



Бесплатная электронная книга

УЧУСЬ CSS

Free unaffiliated eBook created from
Stack Overflow contributors.

#CSS

.....	1
1: CSS	2
.....	2
.....	2
Examples.....	2
.....	2
.....	2
.....	4
.....	4
CSS @import (CSS-).....	5
@import.....	5
CSS JavaScript.....	5
JavaScript.....	6
JQuery.....	6
.....	6
CSS.....	6
2: 2D-	8
.....	8
.....	8
.....	9
2D Coordiante.....	9
.....	10
:	10
Examples.....	10
.....	10
.....	11
.....	11
.....	12
.....	12
.....	14
3: 3D-	16

.....	16
.....	16
Examples.....	17
3D-.....	17
-.....	18
3D-.....	19
CSS.....	19
HTML.....	19
3D-.....	20
4: CSS-.....	23
.....	23
.....	23
Examples.....	23
.....	23
5:.....	25
.....	25
.....	25
Examples.....	25
.....	25
.....	25
.....	26
.....	26
`will-change`.....	27
.....	27
.....	27
-.....	29
.....	29
6:.....	31
.....	31
.....	32
Examples.....	33
.....	33
.....	33

()..... 35

..... 36

[| |]..... 36

..... 36

..... 37

..... 38

CSS..... 38

HTML..... 38

7: 41

..... 41

Examples..... 41

: 41

..... 41

Flexbox..... 42

..... 42

: 43

..... 44

8: Internet Explorer..... 45

..... 45

Examples..... 45

Internet Explorer 10 45

..... 45

: 45

Internet Explorer 6 Internet Explorer 7..... 46

Internet Explorer 8..... 46

IE6 IE7..... 46

9: (Flexbox)..... 47

..... 47

..... 47

..... 47

Vender..... 47

.....	47
Examples.....	48
.....	48
Holy Grail Flexbox.....	49
flexbox.....	50
(, -.....	52
().....	52
HTML.....	53
CSS.....	53
.....	53
.....	53
: justify-content: center flexbox.....	54
: justify-content: center flexbox.....	55
: align-content: center flexbox.....	55
: align-content: center.....	56
:	57
:	58
.....	59
.....	60
10:	62
.....	62
.....	62
.....	62
.....	63
Examples.....	63
rem.....	63
rems ems.....	64
vh vw.....	65
vmin vmax.....	65
%.....	66
11:	68
.....	

- 68
- 69
- Examples.....70
 -70
 -71
 - MediaType.....71
 -72
 - Viewport.....73
 - Retina.....73
 -74
 -74
 - ,74
 - ,74
 - , («»)......75
 - IE8.....75
 - Javascript.....75
 -75
- 12:**77
 -77
 -77
 -77
- Examples.....77
 - @supports.....77
 -78
- 13:**79
 -79
- Examples.....79
 -79
- CSS**.....79
- ?**.....79

1 -	80
2 -	80
3 -	80
	81
!	81
	82
1:	82
2.	83
3:	83
!important	84
	85
	85
14:	87
	87
Examples	87
()	87
	88
15:	90
	90
	90
Examples	90
	90
	90
16:	91
Examples	91
	91
17:	95
	95
Examples	95
,	95
18:	97
	97

.....	97
.....	97
Examples.....	98
.....	98
-.....	98
19:	100
.....	100
.....	100
.....	100
Examples.....	100
.....	100
.....	101
.....	101
.....	102
20:	104
Examples.....	104
.....	104
HTML	104
CSS	104
.....	104
21:	106
.....	106
.....	106
.....	106
Examples.....	106
.....	106
,	106
.....	107
.....	108
.....	110
.....	111
.....

1:	112
22:	113
.....	113
.....	113
.....	113
.....	113
Examples.....	113
?	113
.....	113
.....	114
.....	116
23:	118
.....	118
.....	118
Examples.....	118
.....	118
.....	119
24:	121
.....	121
Examples.....	121
.....	121
.....	121
25:	123
.....	123
.....	123
Examples.....	123
.....	123
.....	124
26:	125
.....	125
.....

Examples..... 125

 normalize.css..... 125

 125

reset.css..... 126

 126

27: **128**

 128

 128

 129

:..... **129**

Clip-:..... **130**

 Examples..... 130

 ()..... 130

CSS:..... **130**

HTML:..... **130**

 ()..... 131

CSS:..... **131**

HTML..... **131**

 : 132

 132

 133

 , 133

CSS..... **133**

HTML..... **134**

 135

CSS..... **135**

HTML..... **135**

 136

CSS..... **136**

HTML.....	136
28: CSS (CSSOM).....	138
.....	138
Examples.....	138
.....	138
background-image CSSOM.....	138
29:	140
Examples.....	140
.....	140
.....	140
.....	144
.....	145
.....	145
.....	146
.....	146
.....	147
.....	148
30:	150
.....	150
.....	150
.....	150
Examples.....	151
:	151
~	151
:	153
,	153
-x -y.....	154
31:	156
.....	156
.....	156
.....	156

Examples.....	157
.....	157
.....	157
CSS.....	157
HTML.....	157
.....	158
32:	160
.....	160
.....	160
Examples.....	161
.....	161
.....	161
33:	162
.....	162
.....	162
.....	162
Examples.....	162
.....	163
z-.....	163
.....	163
HTML.....	163
CSS.....	163
.....	164
.....	164
.....	165
.....	165
.....	166
34: ().....	167
.....	167
.....	167
.....	167

/	167
Examples.....	168
.....	168
.....	168
.....	168
/	169
-.....	170
35:	173
.....	173
.....	173
Examples.....	173
.....	173
IE	174
36:	175
.....	175
.....	175
Examples.....	175
.....	175
.....	176
.....	177
/	178
.....	179
Clearfix.....	180
Clearfix ().....	180
Clearfix	180
Clearfix IE6 IE7.....	181
DIV float.....	181
.....	183
37: -.....	184
.....	184
.....	184

.....	184
.....	185
Examples.....	185
.....	185
.....	185
38:	187
.....	187
.....	187
.....	188
Examples.....	188
Drop Shadow (box-shadow).....	188
.....	189
.....	189
.....	190
.....	190
39:	192
.....	192
.....	192
.....	192
Examples.....	193
.....	193
.....	193
.....	194
[attribute].....	194
[attribute="value"].....	194
[attribute*="value"].....	194
[attribute~="value"].....	195
[attribute^="value"].....	195
[attribute\$="value"].....	195
[attribute = "value"].....	195
[attribute="value" i].....	196
.....	

196	
0-1-0.....	196
.....	196
.....	196
: selector selector	197
: selector > selector	197
 -: selector + selector	198
Combinator: selector ~ selector	198
.....	199
.....	200
-.....	200
.....	201
:	201
.....	204
.....	205
: ~ ().....	205
boolean	206
boolean	206
CSS	206
.....	207
CSS3:	207
.....	207
, ,	208
A. The: & B.: focus-in CSS	208
-	210
last-of-type.....	211
40:	212
.....	212
.....	212
Examples.....	212
.....	212

41:	214
Examples	214
,	214
.....	214
.....	215
42:	216
.....	216
.....	216
.....	216
Examples	216
.....	216
.....	217
/	217
43:	219
.....	219
Examples	219
.....	219
.....	220
.....	220
44: CSS	222
.....	222
.....	222
.....	222
.....	222
Examples	222
,	222
.....	223
.....	223
45:	224
.....	224
.....	224
.....

224	
Examples.....	225
.....	225
CSS.....	225
HTML.....	225
CSS.....	225
CSS.....	225
HTML.....	226
CSS.....	226
CSS.....	226
HTML.....	227
46:	228
.....	228
.....	228
Examples.....	228
.....	228
.....	229
.....	229
.....	230
.....	230
47:	232
.....	232
.....	232
.....	233
Examples.....	233
.....	233
.....	234
.....	234
.....	235
.....	235
.....	236
.....	

.....	237
.....	237
.....	238
.....	239
.....	239
.....	239
.....	240
.....	240
.....	240
48:	241
.....	241
.....	241
Examples	242
.....	242
.....	242
.....	242
.....	242
.....	244
, div	245
49:	246
.....	246
Examples	246
.....	246
50:	249
.....	249
.....	249
.....	249
Examples	249
.....	249
.....	250

.....	250
RGB / RGBa.....	250
HSL / HSLa.....	251
.....	251
.....	252
.....	253
()	253
- ()	254
.....	254
.....	255
.....	256
.....	256
.....	257
.....	258
.....	258
.....	258
background-attachment:	258
background-attachment:	259
background-attachment: local.....	259
.....	259
.....	260
.....	260
background-origin.....	261
.....	263
.....	265
.....	265
.....	266
contain cover.....	266
contain.....	267
cover.....	268
.....	268

background-blend-mode	269
51:	271
.....	271
.....	271
.....	271
Examples	271
- ()	271
.....	273
52:	275
.....	275
.....	275
.....	275
Examples	275
.....	275
53:	277
.....	277
.....	277
Examples	277
calc ()	277
attr ()	278
()	278
()	278
var ()	278
54:	280
.....	280
Examples	280
.....	280
.....	280
.....	288
.....	288
.....	288

rgb ()	289
.....	289
hsl ()	289
.....	290
.....	290
currentColor	290
.....	290
.....	291
rgba ()	292
.....	292
hsla ()	292
.....	293
55:	294
Examples	294
CSS-	294
.....	294
.....	295
Flexbox	295
:	296
()	297
.....	298
.....	298
.....	299
div	300
.....	300
calc ()	301
.....	301
.....	302
,	303
.....	303
.....	303
.....	303

.....	304
.....	304
.....	305
margin: 0 auto;.....	305
56: CSS	308
.....	308
.....	308
Examples.....	308
BEM.....	308
.....	309
.....	310

Около

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: [css](#)

It is an unofficial and free CSS ebook created for educational purposes. All the content is extracted from [Stack Overflow Documentation](#), which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official CSS.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

глава 1: Начало работы с CSS

замечания

Стили могут быть созданы несколькими способами, что позволяет различной степени повторного использования и объема, когда они указаны в исходном HTML-документе. *Внешние* таблицы стилей можно повторно использовать в HTML-документах. *Встроенные* таблицы стилей применяются ко всему документу, в котором они указаны. *Встроенные* стили применяются только к отдельному HTML-элементу, на котором они указаны.

Версии

Версия	Дата выхода
1	1996-12-17
2	1998-05-12
3	2015-10-13

Examples

Внешний лист стилей

Внешняя таблица стилей CSS может применяться к любому количеству HTML-документов, помещая элемент `<link>` в каждый HTML-документ.

Атрибут `rel <link>` должен быть установлен как `"stylesheet"`, а атрибут `href` - относительный или абсолютный путь к таблице стилей. Хотя использование относительных URL-путей обычно считается хорошей практикой, также могут использоваться абсолютные пути. В HTML5 атрибут `type` **может быть опущен**.

Рекомендуется, чтобы `<link>` помещался в тег `<head>` HTML-файла, чтобы стили загружались до стилей, которые они стилируют. В противном случае **пользователи будут видеть вспышку неравномерного содержимого**.

пример

привет-world.html

```
<!DOCTYPE html>
```



```
<html>
  <head>
    <meta charset="utf-8" />
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>
    <h1>Hello world!</h1>
    <p>I ♥ CSS</p>
  </body>
</html>
```

style.css

```
h1 {
  color: green;
  text-decoration: underline;
}
p {
  font-size: 25px;
  font-family: 'Trebuchet MS', sans-serif;
}
```

Убедитесь, что вы указали правильный путь к вашему CSS-файлу в href. Если файл CSS находится в той же папке, что и ваш HTML-файл, путь не требуется (например, пример выше), но если он сохранен в папке, укажите его так: href="foldername/style.css" .

```
<link rel="stylesheet" type="text/css" href="foldername/style.css">
```

Внешние таблицы стилей считаются лучшим способом обработки CSS. Для этого существует очень простая причина: когда вы управляете сайтом, скажем, на 100 страницах, все контролируются одной таблицей стилей, и вы хотите изменить цвета ссылок с синего на зеленый, намного проще сделать изменение в вашем файле CSS, и пусть изменения «каскад» на всех 100 страницах, чем на 100 отдельных страниц и сделать то же самое изменение 100 раз. Опять же, если вы хотите полностью изменить внешний вид своего сайта, вам нужно только обновить этот файл.

Вы можете загрузить как можно больше CSS-файлов на своей странице HTML.

```
<link rel="stylesheet" type="text/css" href="main.css">
<link rel="stylesheet" type="text/css" href="override.css">
```

Правила CSS применяются с некоторыми базовыми правилами, и порядок имеет значение. Например, если у вас есть файл main.css с некоторым кодом в нем:

```
p.green { color: #00FF00; }
```

Все ваши абзацы с «зеленым» классом будут написаны светло-зеленым цветом, но вы можете переопределить это с другим .css-файлом, просто включив его *после* main.css. Вы можете иметь override.css со следующим кодом, следующим за main.css, например:

```
p.green { color: #006600; }
```

Теперь все ваши абзацы с «зеленым» классом будут написаны более темным зеленым, а не светло-зеленым.

Применяются и другие принципы, такие как «важное» правило, специфика и наследование.

Когда кто-то впервые посещает ваш сайт, их браузер загружает HTML текущей страницы плюс связанный файл CSS. Затем, когда они переходят на другую страницу, их браузеру нужно только загрузить HTML-страницу этой страницы; файл CSS кэшируется, поэтому его не нужно загружать снова. Поскольку браузеры кэшируют внешнюю таблицу стилей, ваши страницы загружаются быстрее.

Внутренние стили

CSS, заключенный в теги `<style></style>` в документе HTML, функционирует как внешняя таблица стилей, за исключением того, что он живет в HTML-документе, а не в отдельном файле, и поэтому может применяться только к документу, в котором он жизни. Обратите внимание, что этот элемент *должен* находиться внутри элемента `<head>` для проверки HTML (хотя он будет работать во всех текущих браузерах, если он помещен в `body`).

```
<head>
  <style>
    h1 {
      color: green;
      text-decoration: underline;
    }
    p {
      font-size: 25px;
      font-family: 'Trebuchet MS', sans-serif;
    }
  </style>
</head>
<body>
  <h1>Hello world!</h1>
  <p>I ♥ CSS</p>
</body>
```

Встроенные стили

Используйте встроенные стили для применения стиля к определенному элементу. Обратите внимание, что это **не** является оптимальным. Рекомендуется использовать правила стиля в `<style>` или внешнем файле CSS, чтобы поддерживать различие между контентом и презентацией.

Встроенные стили переопределяют любой CSS в `<style>` или в таблице внешнего стиля. Хотя это может быть полезно в некоторых случаях, этот факт чаще всего не снижает ремонтпригодность проекта.

Стили в следующем примере применяются непосредственно к элементам, к которым они прикреплены.

```
<h1 style="color: green; text-decoration: underline;">Hello world!</h1>
<p style="font-size: 25px; font-family: 'Trebuchet MS';">I ♥ CSS</p>
```

Встраиваемые стили - это, как правило, самый безопасный способ обеспечения совместимости с различными почтовыми клиентами, программами и устройствами, но может потребовать много времени для написания и немного сложной работы.

Правило CSS @import (одно из CSS-правил)

CSS-правило @import CSS используется для импорта правил стиля из других таблиц стилей. Эти правила должны предшествовать всем другим типам правил, кроме правил @charset; поскольку это не вложенный оператор, @import не может использоваться внутри условных групповых at-правил. @import .

Как использовать @import

Вы можете использовать правило @import следующими способами:

A. С тегом внутреннего стиля

```
<style>
  @import url('/css/styles.css');
</style>
```

B. С внешней таблицей стилей

Следующая строка импортирует файл CSS с именем additional-styles.css в корневой каталог в файл CSS, в котором он отображается:

```
@import '/additional-styles.css';
```

Также возможен импорт внешнего CSS. Обычный вариант использования - файлы шрифтов.

```
@import 'https://fonts.googleapis.com/css?family=Lato';
```

Дополнительный второй аргумент правила @import - это список медиа-запросов:

```
@import '/print-styles.css' print;
@import url('landscape.css') screen and (orientation:landscape);
```

Изменение CSS с помощью JavaScript

Чистый JavaScript

С помощью свойства `style` элемента можно добавлять, удалять или изменять значения свойств CSS с помощью JavaScript.

```
var el = document.getElementById("element");
el.style.opacity = 0.5;
el.style.fontFamily = 'sans-serif';
```

Обратите внимание, что свойства стиля называются в стиле нижнего верблюда. В примере вы видите, что семейство `font-family` CSS становится `fontFamily` в javascript.

В качестве альтернативы непосредственно работать с элементами вы можете создать элемент `<style>` или `<link>` в JavaScript и добавить его в `<body>` или `<head>` документа HTML.

JQuery

Изменение свойств CSS с помощью jQuery еще проще.

```
$('#element').css('margin', '5px');
```

Если вам нужно изменить более одного правила стиля:

```
$('#element').css({
  margin: "5px",
  padding: "10px",
  color: "black"
});
```

jQuery включает два способа изменения правил CSS, имеющих в них дефисы (например, `font-size`). Вы можете поместить их в кавычки или верблюд-футляр в название правила стиля.

```
$('.example-class').css({
  "background-color": "blue",
  fontSize: "10px"
});
```

Смотрите также

- [Документация JavaScript - Чтение и изменение стиля CSS](#) .
- [Документация jQuery - Манипуляция CSS](#)

Списки стилей с CSS

Существует три разных свойства для элементов `list-style-type`: `list-style-type`, `list-style-image` и `list-style-position`, которые должны быть объявлены в этом порядке. Значения по умолчанию: диск, внешний и ни один, соответственно. Каждое свойство может быть объявлено отдельно или с использованием сокращенного свойства в `list-style`.

`list-style-type` определяет форму или тип точки маркера, используемой для каждого элемента списка.

Некоторые из допустимых значений типа `list-style-type`:

- диск
- круг
- площадь
- десятичный
- низший романский
- Верхняя-римской
- НИКТО

(Полный список см. В [Wiki спецификации W3C](#))

Например, чтобы использовать квадратные точки маркера для каждого элемента списка, вы должны использовать следующую пару значений свойства:

```
li {
  list-style-type: square;
}
```

Свойство `list-style-image` определяет, установлен ли значок элемента списка с изображением, и принимает значение `none` или URL-адрес, указывающий на изображение.

```
li {
  list-style-image: url(images/bullet.png);
}
```

Свойство `list-style-position` определяет, где размещать маркер элемента списка, и принимает одно из двух значений: «внутри» или «снаружи».

```
li {
  list-style-position: inside;
}
```

Прочитайте Начало работы с CSS онлайн: <https://riptutorial.com/ru/css/topic/293/начало-работы-с-css>

глава 2: 2D-преобразования

Синтаксис

- **Повернуть преобразование**
- transform: rotate (<angle>)
- **Преобразование перевода**
- transform: translate (<length-or-percent> [, <length-or-percent>]?)
- transform: translateX (<длина или процент>)
- transform: translateY (<длина или процент>)
- **Косовое преобразование**
- transform: skew (<angle> [, <angle>]?)
- transform: skewX (<angle>)
- transform: skewY (<angle>)
- **Трансформация шкалы**
- transform: масштаб (<масштабный коэффициент> [, <масштаб-фактор>]?)
- transform: scaleX (<масштаб-фактор>)
- transform: scaleY (<масштаб-фактор>)
- **Матричное преобразование**
- transform: matrix (<number> [, <number>] {5,5})

параметры

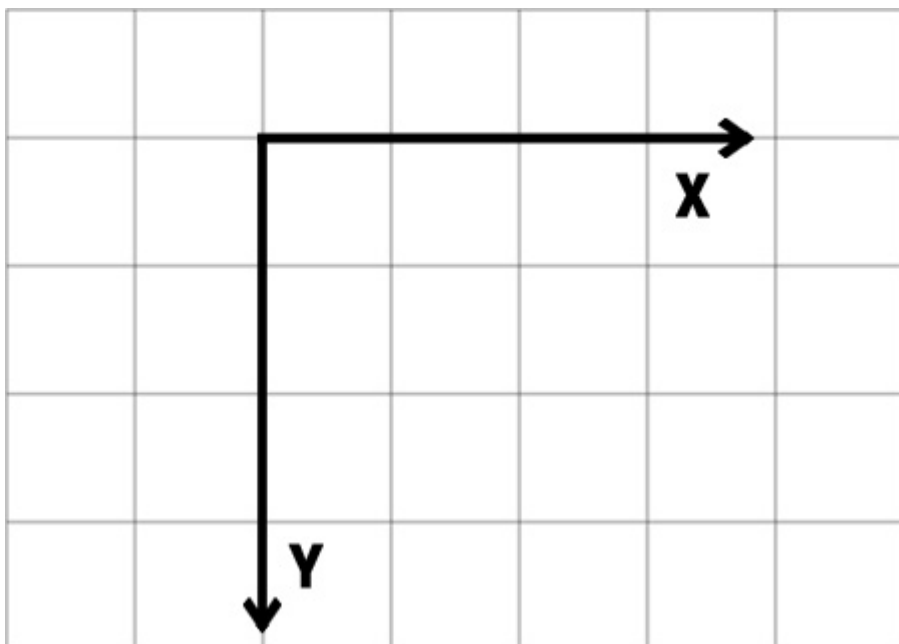
Функция / параметр	подробности
rotate(x)	Определяет преобразование, которое перемещает элемент вокруг неподвижной точки на оси Z
translate(x, y)	Перемещает положение элемента по оси X и Y
translateX(x)	Перемещает положение элемента по оси X
translateY(y)	Перемещает положение элемента по оси Y
scale(x, y)	Изменяет размер элемента по оси X и Y
scaleX(x)	Изменяет размер элемента по оси X
scaleY(y)	Изменяет размер элемента по оси Y
skew(x, y)	Отображение сдвига или трансвекция, искажая каждую точку элемента на определенный угол в каждом направлении

Функция / параметр	подробности
<code>skewX(x)</code>	Отображение горизонтального сдвига, искажающее каждую точку элемента на определенный угол в горизонтальном направлении
<code>skewY(y)</code>	Отображение вертикального сдвига, искажающее каждую точку элемента на определенный угол в вертикальном направлении
<code>matrix()</code>	Определяет двумерное преобразование в виде матрицы преобразования.
угол	Угол поворота или искажения элемента (в зависимости от функции, с которой он используется). Угол может быть в градусах (<code>deg</code>), <code>gradians</code> (<code>grad</code>), радиан (<code>rad</code>) или поворотов (<code>turn</code>). В функции <code>skew()</code> второй угол является необязательным. Если это не предусмотрено, не будет (0) перекоса по оси Y.
Длина или-процентное	Расстояние, выраженное как длина или процент, по которому элемент должен быть переведен. В функции <code>translate()</code> вторая длина или процент необязательна. Если не предусмотрено, то не будет (0) сдвига по оси Y.
масштаб	Число, которое определяет, сколько раз элемент должен быть масштабирован по указанной оси. В функции <code>scale()</code> второй масштабный коэффициент является необязательным. Если это не предусмотрено, первый масштабный коэффициент будет применяться и для оси Y.

замечания

Система 2D Coordiante

Преобразования выполняются в соответствии с системой координат 2D X / Y. Ось X идет справа налево, а ось Y идет вниз, как показано на следующем изображении:



Таким образом, положительный `translateY()` идет вниз, и положительный `translateX()` идет вправо.

Поддержка и префиксы браузера

- IE поддерживает это свойство с IE9 с префиксом `-ms-`. Старые версии и Edge не нуждаются в префиксе
- Firefox поддерживает его с версии 3.5 и нуждается в `-moz-` prefix до версии 15
- Chrome с версии 4 и до версии 34 нуждается в `-webkit-`
- Для Safari необходим префикс `-webkit-` до версии 8
- Opera нуждается в `-o-` префикс для версии 11.5 и `-webkit-` префикс от версии 15 до 22
- Android нуждается в `-webkit-` от версии 2.1 до 4.4.4

Пример префиксного преобразования:

```
-webkit-transform: rotate(45deg);  
-ms-transform: rotate(45deg);  
transform: rotate(45deg);
```

Examples

Поворот

HTML

```
<div class="rotate"></div>
```

CSS


```
.rotate {
  width: 100px;
  height: 100px;
  background: teal;
  transform: rotate(45deg);
}
```

Этот пример повернет div на 45 градусов по часовой стрелке. Центр вращения находится в центре div, 50% слева и 50% сверху. Вы можете изменить центр вращения, установив СВОЙСТВО `transform-origin`.

```
transform-origin: 100% 50%;
```

Вышеприведенный пример установит центр вращения в середину правого бокового конца.

Масштаб

HTML

```
<div class="scale"></div>
```

CSS

```
.scale {
  width: 100px;
  height: 100px;
  background: teal;
  transform: scale(0.5, 1.3);
}
```

Этот пример будет масштабировать div до $100\text{px} * 0.5 = 50\text{px}$ по оси X и до $100\text{px} * 1.3 = 130\text{px}$ по оси Y.

Центр преобразования находится в центре div, 50% слева и 50% сверху.

Переведите

HTML

```
<div class="translate"></div>
```

CSS

```
.translate {
  width: 100px;
  height: 100px;
  background: teal;
  transform: translate(200px, 50%);
}
```

Этот пример переместит div на 200px по оси X и на $100px * 50\% = 50px$ по оси Y.

Вы также можете указать переводы на одной оси.

На оси X:

```
.translate {
  transform: translateX(200px);
}
```

На оси Y:

```
.translate {
  transform: translateY(50%);
}
```

СКОС

HTML

```
<div class="skew"></div>
```

CSS

```
.skew {
  width: 100px;
  height: 100px;
  background: teal;
  transform: skew(20deg, -30deg);
}
```

Этот пример будет искажать div на 20 градусов по оси X и на 30 градусов по оси Y. Центр преобразования находится в центре div, 50% слева и 50% сверху.

См. Результат [здесь](#) .

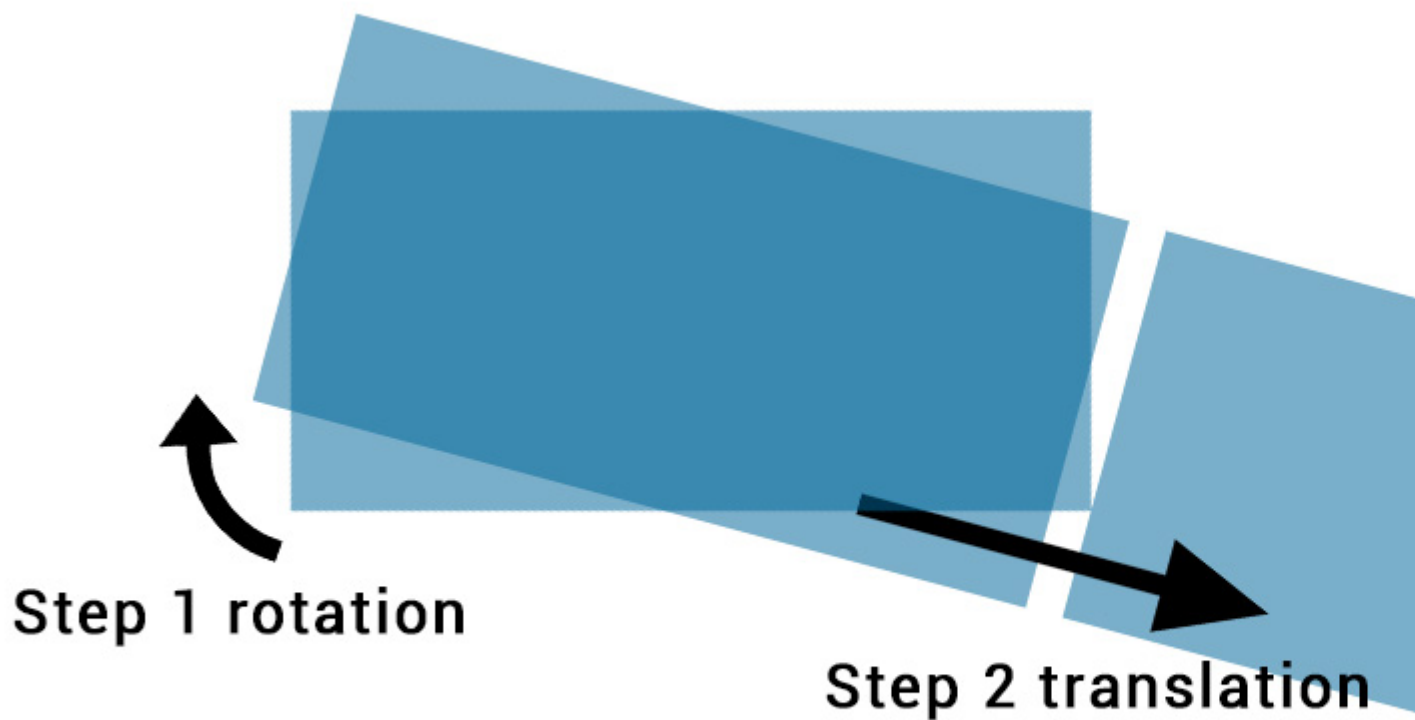
Несколько преобразований

Множественные преобразования могут быть применены к элементу в одном свойстве:

```
transform: rotate(15deg) translateX(200px);
```

Это повернет элемент на 15 градусов по часовой стрелке, а затем переведёт его на 200 пикселей вправо.

В цепных преобразованиях **система координат перемещается вместе с элементом** . Это означает, что перевод не будет горизонтальным, а на оси поверните на 15 градусов по часовой стрелке, как показано на следующем изображении:



Изменение порядка преобразований изменит выход. Первый пример будет отличаться от

```
transform: translateX(200px) rotate(15deg);
```

```
<div class="transform"></div>
```

```
.transform {  
  transform: rotate(15deg) translateX(200px);  
}
```

Как показано на этом изображении:



Step 1 translation

Трансформация происхождения

Преобразования выполняются относительно точки, которая определяется свойством `transform-origin`.

Свойство принимает 2 значения: `transform-origin: XY;`

В следующем примере первый `div` (`.t1`) вращается вокруг верхнего левого угла с `transform-origin: 0 0;` и второй (`.tr`) преобразуется вокруг его верхнего правого угла с `transform-origin: 100% 0`. Вращение применяется **при наведении** :

HTML:

```
<div class="transform origin1"></div>
<div class="transform origin2"></div>
```

CSS:

```
.transform {
  display: inline-block;
  width: 200px;
  height: 100px;
  background: teal;
  transition: transform 1s;
}
```

```
.origin1 {
  transform-origin: 0 0;
}

.origin2 {
  transform-origin: 100% 0;
}

.transform:hover {
  transform: rotate(30deg);
}
```

Значение по умолчанию для свойства `transform-origin` составляет `50% 50%` которое является центром элемента.

Прочитайте 2D-преобразования онлайн: <https://riptutorial.com/ru/css/topic/938/2d-преобразования>

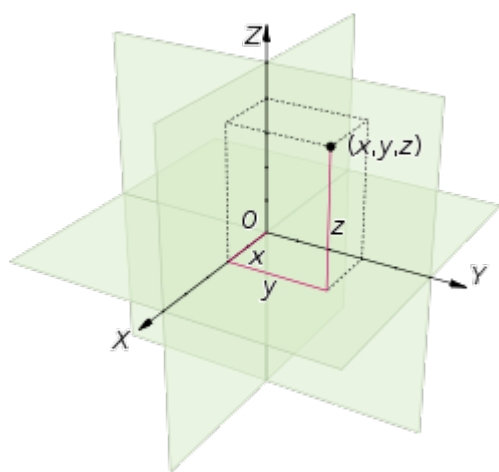
глава 3: 3D-преобразования

замечания

Система координат

3D-преобразования выполнены в соответствии с векторной системой координат (x, y, z) в евклидовом пространстве .

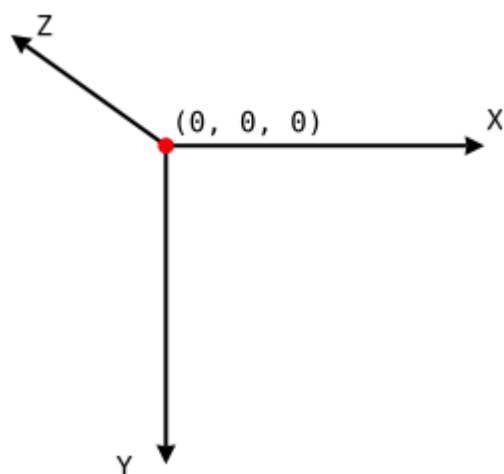
На следующем рисунке показан пример координат в евклидовом пространстве:



В CSS,

- Ось x представляет собой горизонтальную (левую и правую)
- Ось y представляет собой вертикальную (вверх и вниз)
- Ось z представляет собой глубину (вперед и назад / ближе и дальше)

На следующем рисунке показано, как эти координаты переводятся в CSS:



Examples

3D-куб

3D-преобразования могут использоваться для создания множества трехмерных фигур. Вот простой пример куба 3D CSS:

HTML:

```
<div class="cube">
  <div class="cubeFace"></div>
  <div class="cubeFace face2"></div>
</div>
```

CSS:

```
body {
  perspective-origin: 50% 100%;
  perspective: 1500px;
  overflow: hidden;
}
.cube {
  position: relative;
  padding-bottom: 20%;
  transform-style: preserve-3d;
  transform-origin: 50% 100%;
  transform: rotateY(45deg) rotateX(0);
}
.cubeFace {
  position: absolute;
  top: 0;
  left: 40%;
  width: 20%;
  height: 100%;
  margin: 0 auto;
  transform-style: inherit;
  background: #C52329;
  box-shadow: inset 0 0 0 5px #333;
  transform-origin: 50% 50%;
  transform: rotateX(90deg);
  backface-visibility: hidden;
}
.face2 {
  transform-origin: 50% 50%;
  transform: rotateZ(90deg) translateX(100%) rotateY(90deg);
}
.cubeFace:before, .cubeFace:after {
  content: '';
  position: absolute;
  width: 100%;
  height: 100%;
  transform-origin: 0 0;
  background: inherit;
  box-shadow: inherit;
  backface-visibility: inherit;
}
```

```
.cubeFace:before {
  top: 100%;
  left: 0;
  transform: rotateX(-90deg);
}
.cubeFace:after {
  top: 0;
  left: 100%;
  transform: rotateY(90deg);
}
```

Просмотреть этот пример

Дополнительный стиль добавляется в демонстрационную версию, а преобразование применяется при наведении, чтобы просмотреть 6 граней куба.

Следует отметить, что:

- 4 лица выполнены с псевдоэлементами
- [Привязанные преобразования](#)

противоположная сторона-видимость

Свойство `backface-visibility` относится к 3D-преобразованиям.

Благодаря 3D-преобразованиям и свойствам `backface-visibility` вы можете повернуть элемент таким образом, чтобы исходный фронт элемента больше не смотрел на экран.

Например, это отбросит элемент от экрана:

JSFIDDLE

```
<div class="flip">Loren ipsum</div>
<div class="flip back">Lorem ipsum</div>
```

```
.flip {
  -webkit-transform: rotateY(180deg);
  -moz-transform: rotateY(180deg);
  -ms-transform: rotateY(180deg);
  -webkit-backface-visibility: visible;
  -moz-backface-visibility: visible;
  -ms-backface-visibility: visible;
}

.flip.back {
  -webkit-backface-visibility: hidden;
  -moz-backface-visibility: hidden;
  -ms-backface-visibility: hidden;
}
```

Firefox 10+ и IE 10+ поддерживают `backface-visibility` без префикса. Для Opera, Chrome, Safari, iOS и Android все нужно `-webkit-backface-visibility`.

Он имеет 4 значения:

1. **visible** (по умолчанию) - элемент всегда будет отображаться, даже если он не обращен к экрану.
2. **hidden** - элемент не отображается, если он не обращен к экрану.
3. **inherit** - свойство получит свое значение из своего родительского элемента
4. **initial** - устанавливает свойство по умолчанию, которое видно

Указатель компаса или форма иглы с использованием 3D-преобразований

CSS

```
div.needle {
  margin: 100px;
  height: 150px;
  width: 150px;
  transform: rotateY(85deg) rotateZ(45deg);
  /* presentational */
  background-image: linear-gradient(to top left, #555 0%, #555 40%, #444 50%, #333 97%);
  box-shadow: inset 6px 6px 22px 8px #272727;
}
```

HTML

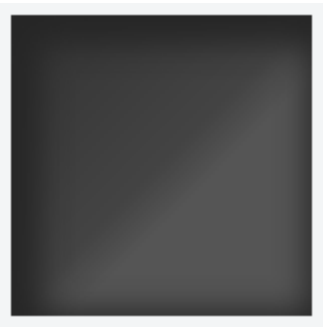
```
<div class='needle'></div>
```

В приведенном выше примере форма стрелки иглы или компаса создается с использованием 3D-преобразований. Обычно, когда мы применяем `rotate` на элементе, вращение происходит только по оси Z, и в лучшем случае мы получим только алмазные формы. Но когда к `rotateY` добавляется `rotateZ` преобразование, элемент сжимается по оси Y и, таким образом, становится похожим на иглу. Чем больше поворот оси Y, тем более сжимается элемент.

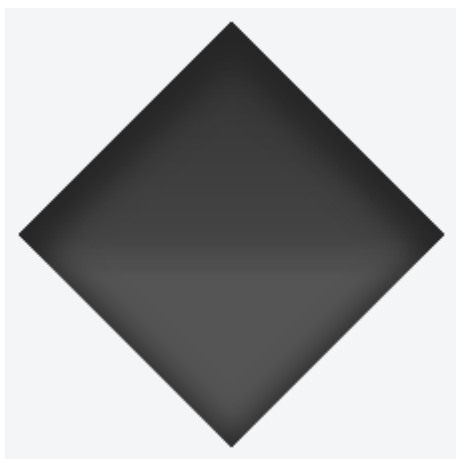
Результатом вышеприведенного примера будет игла, опирающаяся на ее наконечник. Для создания иглы, которая покоится на ее основании, вращение должно быть вдоль оси X, а не вдоль оси Y. Таким образом, значение свойства `transform` должно быть чем-то вроде `rotateX(85deg) rotateZ(45deg);` ,

[Это перо](#) использует аналогичный подход для создания чего-то, что напоминает логотип Safari или циферблат компаса.

Скриншот элемента без преобразования:



Скриншот элемента с только двумерным преобразованием:



Скриншот элемента с 3D-преобразованием:



3D-эффект текста с тенью

HTML:

```
<div id="title">  
  <h1 data-content="HOVER">HOVER</h1>  
</div>
```

CSS:

```
*{margin:0;padding:0;}
```

```
html,body{height:100%;width:100%;overflow:hidden;background:#0099CC;}
#title{
  position:absolute;
  top:50%; left:50%;
  transform:translate(-50%,-50%);
  perspective-origin:50% 50%;
  perspective:300px;
}
h1{
  text-align:center;
  font-size:12vmin;
  font-family: 'Open Sans', sans-serif;
  color:rgba(0,0,0,0.8);
  line-height:1em;
  transform:rotateY(50deg);
  perspective:150px;
  perspective-origin:0% 50%;
}
h1:after{
  content:attr(data-content);
  position:absolute;
  left:0;top:0;
  transform-origin:50% 100%;
  transform:rotateX(-90deg);
  color:#0099CC;
}
#title:before{
  content:'';
  position:absolute;
  top:-150%; left:-25%;
  width:180%; height:328%;
  background:rgba(255,255,255,0.7);
  transform-origin: 0 100%;
  transform: translatez(-200px) rotate(40deg) skewX(35deg);
  border-radius:0 0 100% 0;
}
```

[Просмотреть пример с дополнительным эффектом зависания](#)



В этом примере текст преобразуется, чтобы он выглядел так, как будто он выходит на экран вдали от пользователя.

Тень преобразуется соответствующим образом, чтобы она соответствовала тексту. Поскольку это сделано с псевдоэлементом и атрибутом `data`, он наследует форму преобразований, являющуюся ее родительской (тег `H1`).

Белый «свет» выполнен с `#title` элемента `#title`. Он перекошен и использует граничный радиус для закругленного угла.

Прочитайте 3D-преобразования онлайн: <https://riptutorial.com/ru/css/topic/2446/3d-преобразования>

глава 4: CSS-спрайты

Синтаксис

- // Использование фонового положения
background: url ("sprite-image.png");
background-position: -20px 50px;
- // Сокращение свойств фона
background: url ("sprite-image.png") -20px 50px;

замечания

Для некоторых случаев использования спрайты медленно выходят из употребления, заменяясь значками webfonts или [SVG-изображениями](#) .

Examples

Основная реализация

Что такое спрайт изображения?

Спрайт изображения - это единственный актив, расположенный на листе спрайта изображений. Лист спрайта изображений представляет собой файл изображения, содержащий более одного актива, который можно извлечь из него.

Например:



Изображение выше представляет собой лист спрайтов изображений, и каждая из этих звезд является спрайтом в листе спрайта. Эти справочные листы полезны, потому что они повышают производительность за счет сокращения количества HTTP-запросов, которые браузер может сделать.

Итак, как вы его реализуете? Вот пример кода.

HTML

```
<div class="icon icon1"></div>
<div class="icon icon2"></div>
<div class="icon icon3"></div>
```

CSS

```
.icon {
  background: url("icons-sprite.png");
  display: inline-block;
  height: 20px;
  width: 20px;
}
.icon1 {
  background-position: 0px 0px;
}
.icon2 {
  background-position: -20px 0px;
}
.icon3 {
  background-position: -40px 0px;
}
```

Используя настройку ширины и высоты спрайта и используя свойство `background-position` в CSS (со значением `x` и `y`), вы можете легко извлечь спрайты из листа спрайтов с помощью CSS.

Прочитайте CSS-спрайты онлайн: <https://riptutorial.com/ru/css/topic/3690/css-спрайты>

глава 5: Анимации

Синтаксис

- `transition: <property> <duration> <timing-function> <delay>;`
- `@keyframes <identifier>`
- `[[from | to | <percentage>] [, from | to | <percentage>]* block]*`

параметры

переход	
параметр	подробности
имущество	Либо свойство CSS для перехода, либо <code>all</code> , задающее все свойства, доступные для перехода.
продолжительность	Время перехода, либо в секундах, либо в миллисекундах.
временная функция	Определяет функцию для определения того, как вычисляются промежуточные значения для свойств. Общие значения - это <code>ease</code> , <code>linear</code> и <code>step-end</code> . Для получения более подробной информации ознакомьтесь с упрощенным списком функций .
задержка	Количество времени, в секундах или миллисекундах, для ожидания перед воспроизведением анимации.
@keyframes	
<code>[от к <percentage>]</code>	Вы можете указать заданное время с процентным значением или два процентных значения, то есть <code>10%</code> , <code>20%</code> , в течение периода времени, когда установлены атрибуты набора ключевого кадра.
<code>block</code>	Любое количество атрибутов CSS для ключевого кадра.

Examples

Анимации с переходным свойством

Свойство CSS- `transition` полезное для простых анимаций, позволяет свойствам CSS на основе чисел анимироваться между состояниями.

пример

```
.Example{
  height: 100px;
  background: #fff;
}

.Example:hover{
  height: 120px;
  background: #ff0000;
}
```

[Просмотреть результат](#)

По умолчанию, зависание над элементом с классом `.Example` немедленно приведет к тому, что высота элемента `120px` на `120px` а цвет фона - на красный (`#ff0000`).

Добавляя свойство `transition` , мы можем вызвать эти изменения со временем:

```
.Example{
  ...
  transition: all 400ms ease;
}
```

[Просмотреть результат](#)

`all` значение применяет переход ко всем совместимым (основанным на цифрах) свойствам. Любое имя совместимого свойства (например, `height` или `top`) можно заменить для этого ключевого слова.

`400ms` определяет количество времени, которое занимает переход. В этом случае изменение высоты элемента займет 400 миллисекунд.

Наконец, `ease` - это функция анимации, которая определяет, как воспроизводится анимация. `ease` означает начать медленно, ускорить, а затем снова замедлить. Другие значения являются `linear` , `ease-out` и `ease-in` .

Совместимость между браузерами

Свойство `transition` обычно хорошо поддерживается во всех основных браузерах, за исключением IE 9. Для более ранних версий браузеров Firefox и Webkit используйте префиксы поставщиков, например:

```
.Example{
  transition:      all 400ms ease;
```



```
-moz-transition:    all 400ms ease;
-webkit-transition: all 400ms ease;
}
```

Примечание. Свойство `transition` может анимировать изменения между любыми двумя численными значениями независимо от единицы. Он также может переходить между единицами, такими как `100px 50vh`. Однако он не может переходить между числом и значением по умолчанию или автоматически, например, переходом высоты элемента с `100px` на `auto`.

Увеличение производительности анимации Использование атрибута `will-change``

При создании анимаций и других тяжелых GPU-действиях важно понять атрибут `will-change`.

Оба ключевых кадра CSS и свойство `transition` используют ускорение GPU. Производительность повышается за счет выгрузки вычислений на графический процессор устройства. Это делается путем создания слоев краски (части страницы, которые отображаются отдельно), которые выгружаются на графический процессор для расчета. Свойство `will-change` сообщает браузеру, что будет анимировать, позволяя браузеру создавать небольшие области рисования, тем самым повышая производительность.

Свойство `will-change` принимает список свойств, которые будут анимированы. Например, если вы планируете преобразовать объект и изменить его непрозрачность, вы должны указать:

```
.Example{
  ...
  will-change: transform, opacity;
}
```

Примечание. Использование `will-change` экономно. Установка `will-change` для каждого элемента на странице может привести к снижению производительности, так как браузер может попытаться создать слои краски для каждого элемента, что значительно увеличивает объем обработки проделанного GPU.

Анимации с ключевыми кадрами

Для многоступенчатых анимаций CSS вы можете создавать CSS- `@keyframes`. Ключевые кадры позволяют вам определять несколько точек анимации, называемых ключевыми кадрами, для определения более сложных анимаций.

Основной пример

В этом примере мы создадим базовую фоновую анимацию, которая будет циклически перемещаться между всеми цветами.

```
@keyframes rainbow-background {
  0%      { background-color: #ff0000; }
  8.333%  { background-color: #ff8000; }
  16.667% { background-color: #ffff00; }
  25.000% { background-color: #80ff00; }
  33.333% { background-color: #00ff00; }
  41.667% { background-color: #00ff80; }
  50.000% { background-color: #00ffff; }
  58.333% { background-color: #0080ff; }
  66.667% { background-color: #0000ff; }
  75.000% { background-color: #8000ff; }
  83.333% { background-color: #ff00ff; }
  91.667% { background-color: #ff0080; }
  100.00% { background-color: #ff0000; }
}

.RainbowBackground {
  animation: rainbow-background 5s infinite;
}
```

[Просмотреть результат](#)

Здесь есть несколько разных вещей. Во-первых, фактический синтаксис `@keyframes` .

```
@keyframes rainbow-background{
```

Это задает имя анимации для `rainbow-background` .

```
0%      { background-color: #ff0000; }
```

Это определение ключевого кадра в анимации. Первая часть, `0%` в случае, определяет, где ключевой кадр во время анимации. `0%` означает, что это `0%` от общего времени анимации с самого начала.

Анимация автоматически перейдет между ключевыми кадрами. Таким образом, установив следующий фоновый цвет на `8.333%` , анимация плавно займет `8,333%` времени для перехода между этими ключевыми кадрами.

```
.RainbowBackground {
  animation: rainbow-background 5s infinite;
}
```

Этот код присоединяет нашу анимацию ко всем элементам, которые имеют класс `.RainbowBackground` .

Фактическое свойство анимации принимает следующие аргументы.

- **animation-name** : название нашей анимации. В этом случае `rainbow-background`

- **Продолжительность анимации** : сколько времени займет анимация, в данном случае 5 секунд.
- **animation-iteration-count (Необязательно)** : количество циклов анимации. В этом случае анимация будет продолжаться бесконечно. По умолчанию анимация будет воспроизводиться один раз.
- **animation-delay (Необязательно)** : указывает, как долго ждать анимацию. Значение по умолчанию равно 0 секундам и может принимать отрицательные значения. Например, `-2s` запустит анимацию на 2 секунды в свой цикл.
- **Функция анимации-времени (необязательно)** : задает кривую скорости анимации. По умолчанию он `ease` , когда анимация начинает медленно, быстрее и заканчивается медленнее.

В этом конкретном примере как `0%` и `100%` ключевые кадры определяют `{ background-color: #ff0000; }` . Где бы два или более ключевых кадра не разделяли состояние, можно указать их в одном выражении. В этом случае две строки `0%` и `100%` могут быть заменены на одну строку:

```
0%, 100% { background-color: #ff0000; }
```

Кросс-браузерная совместимость

Для более старых браузеров на основе WebKit вам необходимо использовать префикс поставщика как для объявления `@keyframes` и для свойства `animation` :

```
@-webkit-keyframes{
-webkit-animation: ...
```

Примеры синтаксиса

Наш первый пример синтаксиса показывает свойство сокращения анимации, используя все доступные свойства / параметры:

```
animation: 3s ease-in 1s 2 reverse both
paused slidein;
/* duration | timing-function | delay | iteration-count | direction | fill-mode |
play-state | name */
```

Наш второй пример немного более прост и показывает, что некоторые свойства можно опустить:

```
animation: 3s linear 1s slidein;
/* duration | timing-function | delay | name */
```

Наш третий пример показывает минимальное объявление. Обратите внимание, что имя анимации и продолжительность анимации должны быть объявлены:

```
animation: 3s      slidein;  
/*      duration | name */
```

Также стоит упомянуть, что при использовании сокращения анимации порядок свойств имеет значение. Очевидно, браузер может свести вашу продолжительность с задержкой.

Если краткость не является вашей вещью, вы также можете пропустить стенографическое свойство и выписать каждую собственность отдельно:

```
animation-duration: 3s;  
animation-timing-function: ease-in;  
animation-delay: 1s;  
animation-iteration-count: 2;  
animation-direction: reverse;  
animation-fill-mode: both;  
animation-play-state: paused;  
animation-name: slidein;
```

Прочитайте Анимации онлайн: <https://riptutorial.com/ru/css/topic/590/анимации>

глава 6: бордюры

Синтаксис

- **граница**

- border: border-border border-style border-color | начальная | наследовать;
- border-top: border-width border-style border-color | начальная | наследовать;
- border-bottom: border-width border-style border-color | начальная | наследовать;
- border-left: border-width border-style border-color | начальная | наследовать;
- border-right: border-width border-style border-color | начальная | наследовать;

- **стиль границы**

- пограничный стиль: 1-4 нет | скрытый | пунктирный | пунктирный | твердый | двойной | паз | гребень | вставка | начало | начальная | наследовать;

- **граница радиуса**

- border-radius: 1-4 длина | % / 1-4 длина | % | начальная | наследовать;
- border-top-left-radius: length | % [длина | %] | начальная | наследовать;
- border-top-right-radius: length | % [длина | %] | начальная | наследовать;
- border-bottom-left-radius: length | % [длина | %] | начальная | наследовать;
- border-bottom-right-radius: length | % [длина | %] | начальная | наследовать;

- **границы изображения**

- border-image: border-image-source border-image-slice [border-image-width [border-image-outset]] border-image-repeat
- border-image-source: none | образ;
- border-image-slice: 1-4 номер | % [fill]
- border-image-repeat: 1-2 стрейч | повторить | круглый | пространство

- **границы коллапса**

- пограничный коллапс: отдельный | коллапс | начальная | унаследовать

замечания

Связанные свойства :

- граница
- граница дна
- границы снизу цвет
- Граница-нижний левый-радиус
- границы нижнего правого радиуса
- границы снизу стиль
- Граница дно ширина
- цвет границы
- границы изображения
- границы изображения боковик
- границы изображения повторите
- границы изображения среза
- границы изображения источника
- границы изображения ширина
- границы левого
- границы левого цвета
- границы слева стиль
- Граница левой ширина
- граница радиуса
- граница правой
- границы правого цвета
- границы правой кнопкой стиль
- граница правой ширины

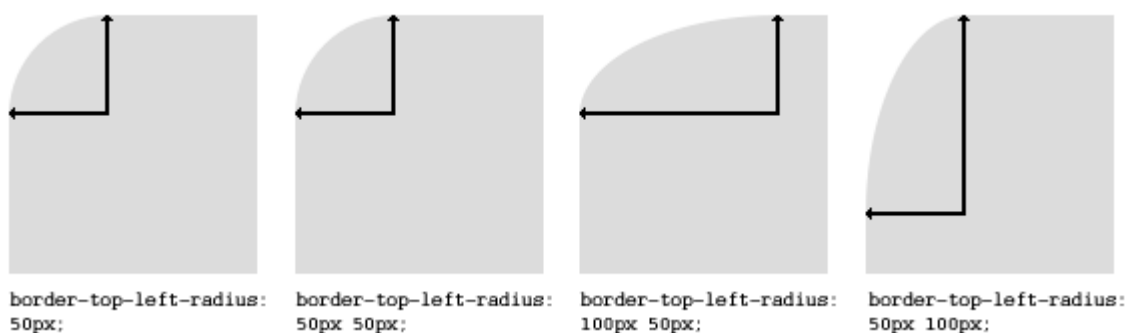
- стиль границы
- границы сверху
- границы верхнего цвета
- границы левого верхнего радиуса
- границы верхнего правого радиуса
- границы топ-стиль
- границы верхней ширины
- ширина рамки

Examples

граница радиуса

Свойство `border-radius` позволяет вам изменить форму базовой модели коробки.

Каждый угол элемента может иметь до двух значений, для вертикального и горизонтального радиуса этого угла (максимум 8 значений).



Первый набор значений определяет горизонтальный радиус. Дополнительный второй набор значений, которому предшествует «/», определяет вертикальный радиус. Если задан только один набор значений, он используется как для вертикального, так и для горизонтального радиуса.

```
border-radius: 10px 5% / 20px 25em 30px 35em;
```

10px - это горизонтальный радиус верхнего левого и нижнего правого. И 5% - это горизонтальный радиус верхнего правого и нижнего левого. Остальные четыре значения после «/» представляют собой вертикальные радиусы для верхнего левого, верхнего правого, нижнего правого и нижнего левого.

Как и во многих свойствах CSS, сокращения могут использоваться для любых или всех возможных значений. Поэтому вы можете указать что угодно от одного до восьми значений. Следующая стенограмма позволяет установить горизонтальный и вертикальный радиус каждого угла в одно и то же значение:

HTML:

```
<div class='box'></div>
```

CSS:

```
.box {  
  width: 250px;  
  height: 250px;  
  background-color: black;  
  border-radius: 10px;  
}
```

Пограничный радиус чаще всего используется для преобразования элементов окна в круги. Установив радиус границы на половину длины квадратного элемента, создается круговой элемент:

```
.circle {  
  width: 200px;  
  height: 200px;  
  border-radius: 100px;  
}
```

Поскольку граничный радиус принимает проценты, обычно используется 50%, чтобы избежать ручного вычисления значения граничного радиуса:

```
.circle {  
  width: 150px;  
  height: 150px;  
  border-radius: 50%;  
}
```

Если значения ширины и высоты не равны, результирующая фигура будет овальной, а не круговой.

Пример граничного радиуса действия браузера:

```
-webkit-border-top-right-radius: 4px;  
-webkit-border-bottom-right-radius: 4px;  
-webkit-border-bottom-left-radius: 0;  
-webkit-border-top-left-radius: 0;  
-moz-border-radius-topright: 4px;  
-moz-border-radius-bottomright: 4px;  
-moz-border-radius-bottomleft: 0;  
-moz-border-radius-topleft: 0;  
border-top-right-radius: 4px;
```



```
border-bottom-right-radius: 4px;  
border-bottom-left-radius: 0;  
border-top-left-radius: 0;
```

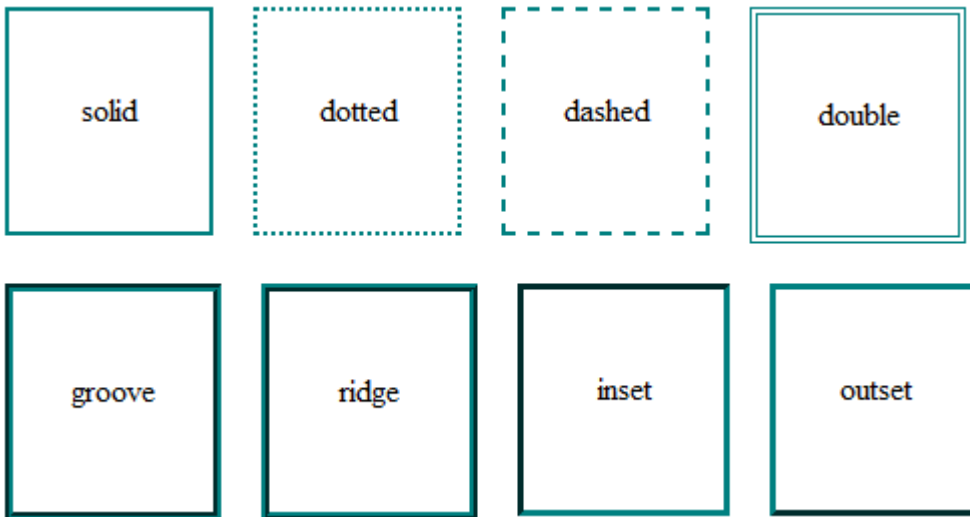
стиль границы

Свойство `border-style` устанавливает стиль границы элемента. Это свойство может иметь от одного до четырех значений (для каждой стороны значения элемента один).

Примеры:

```
border-style: dotted;
```

```
border-style: dotted solid double dashed;
```



`border-style` также может иметь значения `none` и `hidden`. Они имеют тот же эффект, за исключением `hidden` работ для разрешения конфликтов границ для элементов `<table>`. В `<table>` с несколькими границами `none` них `none` имеет наименьшего приоритета (значение в конфликте, граница будет отображаться), а `hidden` имеет самый высокий приоритет (что означает конфликт, граница не будет отображаться).

граница (стенограммы)

В большинстве случаев вы хотите определить несколько свойств границы (`border-width`, `border-style` и `border-color`) для всех сторон элемента.

Вместо того, чтобы писать:

```
border-width: 1px;  
border-style: solid;  
border-color: #000;
```

Вы можете просто написать:

```
border: 1px solid #000;
```

Эти сокращения также доступны для каждой стороны элемента: `border-top` , `border-left` , `border-right` и `border-bottom` . Таким образом, вы можете:

```
border-top: 2px double #aaaaaa;
```

границы изображения

С свойством `border-image` вас есть возможность установить образ, который будет использоваться вместо обычных стилей границы.

`border-image` существу состоит из

- `border-image-source` : путь к изображению, которое будет использоваться
- `border-image-slice` : Задаёт смещение, которое используется для разделения изображения на **девять областей** (четыре **угла** , четыре **края** и **средний**)
- `border-image-repeat` : определяет, как масштабируются изображения для сторон и середины пограничного изображения

Рассмотрим следующий пример, где `border.png` является изображением 90x90 пикселей:

```
border-image: url("border.png") 30 stretch;
```

Изображение будет разделено на девять областей с 30x30 пикселями. Края будут использоваться в качестве углов границы, в то время как сторона будет использоваться между ними. Если элемент выше / шире, чем 30 пикселей, эта часть изображения будет **растянута** . По умолчанию средняя часть изображения будет прозрачной.

внешних пунктов пропуска [влево | вправо | верхняя | нижняя]

Свойство `border-[left|right|top|bottom]` используется для добавления границы к определенной стороне элемента.

Например, если вы хотите добавить границу в левую часть элемента, вы можете сделать следующее:

```
#element {  
  border-left: 1px solid black;  
}
```

границы коллапса

Свойство `border-collapse` применяется только к `table s` (и элементам, отображаемым как `display: table` или `inline-table`), и устанавливает, сворачиваются ли границы таблицы в одну границу или отделяются, как в стандартном HTML.

```
table {
  border-collapse: separate; /* default */
  border-spacing: 2px; /* Only works if border-collapse is separate */
}
```

Также см. [Раздел «Таблицы - оглавление границ»](#)

Несколько границ

Использование контуров:

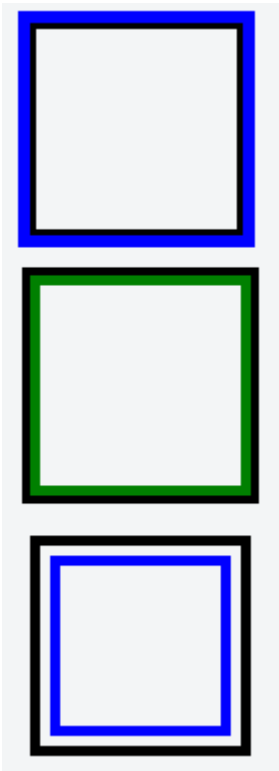
```
.div1{
  border: 3px solid black;
  outline: 6px solid blue;
  width: 100px;
  height: 100px;
  margin: 20px;
}
```

Использование box-shadow:

```
.div2{
  border: 5px solid green;
  box-shadow: 0px 0px 0px 4px #000;
  width: 100px;
  height: 100px;
  margin: 20px;
}
```

Использование псевдоэлемента:

```
.div3 {
  position: relative;
  border: 5px solid #000;
  width: 100px;
  height: 100px;
  margin: 20px;
}
.div3:before {
  content: " ";
  position: absolute;
  border: 5px solid blue;
  z-index: -1;
  top: 5px;
  left: 5px;
  right: 5px;
  bottom: 5px;
}
```



<http://jsfiddle.net/MadalinaTn/bvqpcohm/2/>

Создание многоцветной границы с использованием пограничного изображения

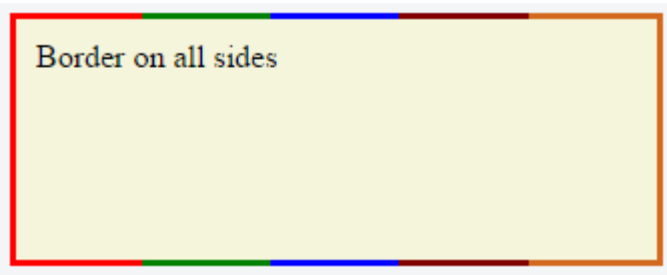
CSS

```
.bordered {  
  border-image: linear-gradient(to right, red 20%, green 20%, green 40%, blue 40%, blue 60%,  
  maroon 60%, maroon 80%, chocolate 80%); /* gradient with required colors */  
  border-image-slice: 1;  
}
```

HTML

```
<div class='bordered'>Border on all sides</div>
```

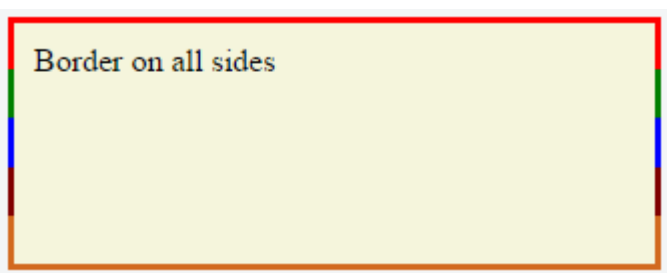
В приведенном выше примере будет создана граница, состоящая из 5 разных цветов. Цвета определяются с помощью `linear-gradient` (вы можете найти дополнительную информацию о градиентах в [документах](#)). Вы можете найти дополнительную информацию о свойстве `border-image-slice` в [примере border-image](#) на той же странице.



(*Примечание: дополнительные свойства были добавлены к элементу для целей презентации*).

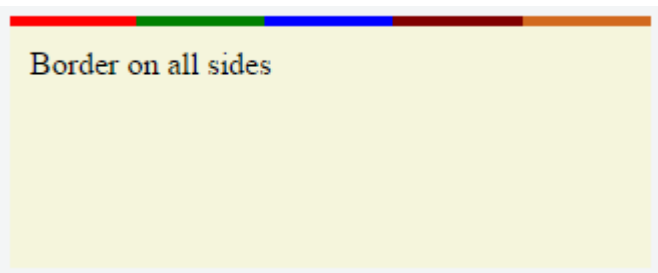
Вы заметили бы, что левая граница имеет только один цвет (начальный цвет градиента), а правая граница также имеет только один цвет (цвет конца градиента). Это связано с тем, как работает свойство `border image`. Это похоже на то, что градиент применяется ко всей коробке, а затем цвета маскируются из областей заполнения и содержимого, что делает его похожим на градиент.

Какая граница (ы) имеет один цвет, зависит от определения градиента. Если градиент - это `to right` градиент, левая граница будет начальным цветом градиента, а правая граница будет цветом конца. Если бы это был `to bottom` градиент, верхняя граница была бы цветом начала градиента, а нижняя граница была бы конечной. Ниже представлен выход цветного градиента `to bottom` 5.



Если граница требуется только на определенных сторонах элемента, то свойство `border-width` можно использовать так же, как и с любой другой обычной границей. Например, добавление кода ниже приведет к созданию границы только в верхней части элемента.

```
border-width: 5px 0px 0px 0px;
```



Обратите внимание, что любой элемент, обладающий свойством `border-image` , **не будет уважать** `border-radius` (то есть граница не будет кривая). Это основано на приведенном ниже описании в спецификации:

Фон фона, но не его пограничное изображение, привязывается к соответствующей кривой (как определено «background-clip»).

Прочитайте бордюры онлайн: <https://riptutorial.com/ru/css/topic/2160/бордюры>

глава 7: Вертикальное центрирование

замечания

Это используется, когда размеры элемента (`width` и `height`) неизвестны или динамичны.

Предпочитаете использовать **Flexbox** по всем другим параметрам, так как он оптимизирован для дизайна пользовательского интерфейса.

Examples

Центрирование с дисплеем: стол

HTML:

```
<div class="wrapper">
  <div class="outer">
    <div class="inner">
      centered
    </div>
  </div>
</div>
```

CSS:

```
.wrapper {
  height: 600px;
  text-align: center;
}
.outer {
  display: table;
  height: 100%;
  width: 100%;
}
.outer .inner {
  display: table-cell;
  text-align: center;
  vertical-align: middle;
}
```

Центрирование с преобразованием

HTML:

```
<div class="wrapper">
  <div class="centered">
    centered
  </div>
</div>
```

CSS:

```
.wrapper {
  position: relative;
  height: 600px;
}
.centered {
  position: absolute;
  z-index: 999;
  transform: translate(-50%, -50%);
  top: 50%;
  left: 50%;
}
```

Центрирование с помощью Flexbox

HTML:

```
<div class="container">
  <div class="child"></div>
</div>
```

CSS:

```
.container {
  height: 500px;
  width: 500px;
  display: flex;           // Use Flexbox
  align-items: center;     // This centers children vertically in the parent.
  justify-content: center; // This centers children horizontally.
  background: white;
}

.child {
  width: 100px;
  height: 100px;
  background: blue;
}
```

Центрирование текста с высотой линии

HTML:

```
<div class="container">
  <span>vertically centered</span>
</div>
```

CSS:

```
.container{
  height: 50px;           /* set height */
  line-height: 50px;     /* set line-height equal to the height */
  vertical-align: middle; /* works without this rule, but it is good having it explicitly
```



```
set */
}
```

Примечание. Этот метод будет только вертикально центрировать *одну строку текста*. Он не будет правильно блокировать элементы блока, и если текст разбивается на новую строку, у вас будет две очень высокие строки текста.

Центрирование с положением: абсолютное

HTML:

```
<div class="wrapper">
  
</div>
```

CSS:

```
.wrapper{
  position:relative;
  height: 600px;
}
.wrapper img {
  position: absolute;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  margin: auto;
}
```

Если вы хотите центрировать другие изображения, тогда вы должны указывать высоту и ширину для этого элемента.

HTML:

```
<div class="wrapper">
  <div class="child">
    make me center
  </div>
</div>
```

CSS:

```
.wrapper{
  position:relative;
  height: 600px;
}
.wrapper .child {
  position: absolute;
  top: 0;
  left: 0;
  right: 0;
```

```
bottom: 0;
margin: auto;
width: 200px;
height: 30px;
border: 1px solid #f00;
}
```

Центрирование с псевдоэлементом

HTML:

```
<div class="wrapper">
  <div class="content"></div>
</div>
```

CSS:

```
.wrapper{
  min-height: 600px;
}

.wrapper:before{
  content: "";
  display: inline-block;
  height: 100%;
  vertical-align: middle;
}

.content {
  display: inline-block;
  height: 80px;
  vertical-align: middle;
}
```

Этот метод лучше всего использовать в тех случаях, когда у вас есть различная высота.

`.content` сосредоточен внутри `.wrapper` ; и вы хотите, `.wrapper` высота `.content` расширилась, когда высота `.wrapper` превысила `.wrapper` высоту `.wrapper` .

Прочитайте Вертикальное центрирование онлайн: <https://riptutorial.com/ru/css/topic/5070/вертикальное-центрирование>

глава 8: Взломы Internet Explorer

замечания

Эти «хаки» могут использоваться для таргетинга на конкретного браузера / клиента. Это можно использовать для устранения различий между браузерами, применяя стили в одной из этих оберток, перечисленных выше.

Examples

Режим высокой контрастности в Internet Explorer 10 и выше

В Internet Explorer 10+ и Edge Microsoft предоставляет селектор « `-ms-high-contrast media` », чтобы отобразить параметр «Высокий контраст» в браузере, который позволяет программисту соответствующим образом корректировать стили своего сайта.

Селектор `-ms-high-contrast` имеет 3 состояния: `active`, `black-on-white` и `white-on-black`. В IE10 не + он также имел `none` состояние, но больше не поддерживаются в Крае идти вперед.

Примеры

```
@media screen and (-ms-high-contrast: active), (-ms-high-contrast: black-on-white) {
  .header{
    background: #fff;
    color: #000;
  }
}
```

Это изменит фон заголовка на белый, а цвет текста на черный, если активен режим высокой контрастности, и он находится в режиме « `black-on-white` ».

```
@media screen and (-ms-high-contrast: white-on-black) {
  .header{
    background: #000;
    color: #fff;
  }
}
```

Как и в первом примере, но это специально выбирает только `white-on-black` состояние и инвертирует цвета заголовка на черный фон с белым текстом.

Дополнительная информация:

[Документация Microsoft](#) на `-ms-high-contrast`

Только Internet Explorer 6 и только Internet Explorer 7

Чтобы настроить таргетинг на Internet Explorer 6 и Internet Explorer 7, запустите свои свойства с помощью * :

```
.hide-on-ie6-and-ie7 {
    *display : none; // This line is processed only on IE6 and IE7
}
```

Не-буквенно-цифровые префиксы (кроме дефиса и подчеркивания) игнорируются в IE6 и IE7, поэтому этот хак работает для любой пары `unprefixed property: value` .

Только Internet Explorer 8

Чтобы настроить таргетинг на Internet Explorer 8, оберните селекторами внутри `@media \0 screen { }` :

```
@media \0 screen {
    .hide-on-ie8 {
        display : none;
    }
}
```

Все, что находится между `@media \0 screen { }` , обрабатывается только I

Добавление поддержки встроенного блока для IE6 и IE7

```
display: inline-block;
```

Свойство `display` со значением `inline-block` не поддерживается Internet Explorer 6 и 7. Обход для этого:

```
zoom: 1;
*display: inline;
```

Свойство `zoom` запускает функцию `hasLayout` элементов и доступно только в Internet Explorer. `*display` гарантирует, что недопустимое свойство выполняется только в затронутых браузерах. Другие браузеры просто игнорируют правило.

Прочитайте Взломы Internet Explorer онлайн: <https://riptutorial.com/ru/css/topic/5056/взломы-internet-explorer>

глава 9: Гибкая компоновка ящиков (Flexbox)

Вступление

Модуль Flexible Box, или просто «flexbox» для краткости, представляет собой коробчатую модель, предназначенную для пользовательских интерфейсов, и позволяет пользователям выравнивать и распределять пространство между элементами в контейнере, так что элементы ведут себя предсказуемо, когда макет страницы должен размещать разные неизвестные размеры экрана. Гибкий контейнер расширяет элементы для заполнения свободного места и сжимает их, чтобы предотвратить переполнение.

Синтаксис

- дисплей: flex;
- flex-direction: row | строка-обратная | столбец | колонного обратное;
- flex-wrap: nowrap | упаковка | наматывается обратное;
- flex-flow: <'flex-direction'> || <'Сгибать-обертывание'>
- justify-content: flex-start | гибкий конец | центр | пространство между | пространство вокруг;
- выравнивающие элементы: flex-start | гибкий конец | центр | исходный | протяжение;
- align-content: flex-start | гибкий конец | центр | пространство между | пространство вокруг | протяжение;
- порядок: <целое>;
- flex-grow: <число>; / * default 0 * /
- flex-shrink: <число>; / * default 1 * /
- flex-basis: <длина> | авто; / * по умолчанию авто * /
- flex: none | [<'flex-grow'> <'flex-shrink'>? || <'flex-basis'>]
- align-self: авто | гибкий старт | гибкий конец | центр | исходный | протяжение;

замечания

Префикс Vendor

- дисплей: -webkit-box; / * Chrome <20 * /
- отображение: -webkit-flex; / * Chrome 20+ * /
- display: -moz-box; /* Fire Fox */
- display: -ms-flexbox; / * IE * /
- дисплей: flex; / * Современные браузеры * /

Ресурсы

- [Полное руководство по Flexbox](#)
- [Решено Flexbox](#)
- [Что такое Flexbox ?!](#)
- [Flexbox через 5 минут](#)
- [Flexbugs](#)

Examples

Липкий нижний колонтитул переменной высоты

Этот код создает липкий нижний колонтитул. Когда содержимое не дойдет до конца окна просмотра, нижний колонтитул прилегает к нижней части окна просмотра. Когда контент проходит мимо нижней части окна просмотра, нижний колонтитул также вытесняется из окна просмотра. [Просмотреть результат](#)

HTML:

```
<div class="header">
  <h2>Header</h2>
</div>

<div class="content">
  <h1>Content</h1>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer nec odio. Praesent libero. Sed cursus ante dapibus diam. Sed nisi. Nulla quis sem at nibh elementum imperdiet. Duis sagittis ipsum. Praesent mauris. Fusce nec tellus sed augue semper porta. Mauris massa. Vestibulum lacinia arcu eget nulla. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Curabitur sodales ligula in libero. </p>
</div>

<div class="footer">
  <h4>Footer</h4>
</div>
```

CSS:

```
html, body {
  height: 100%;
}

body {
  display: flex;
  flex-direction: column;
}

.content {
  /* Include `0 auto` for best browser compatibility. */
  flex: 1 0 auto;
```

```
}

.header, .footer {
  background-color: grey;
  color: white;
  flex: none;
}
```

Макет Holy Grail с использованием Flexbox

Макет Holy Grail - это макет с фиксированным верхним и нижним колонтитулом высоты, а также центр с 3 колонками. 3 столбца включают фиксированную ширину sidenav, центр текущей среды и столбец для другого контента, такого как реклама (центр жидкости появляется сначала в разметке). CSS Flexbox можно использовать для достижения этой простой разметки:

Разметка HTML:

```
<div class="container">
  <header class="header">Header</header>
  <div class="content-body">
    <main class="content">Content</main>
    <nav class="sidenav">Nav</nav>
    <aside class="ads">Ads</aside>
  </div>
  <footer class="footer">Footer</footer>
</div>
```

CSS:

```
body {
  margin: 0;
  padding: 0;
}

.container {
  display: flex;
  flex-direction: column;
  height: 100vh;
}

.header {
  flex: 0 0 50px;
}

.content-body {
  flex: 1 1 auto;

  display: flex;
  flex-direction: row;
}

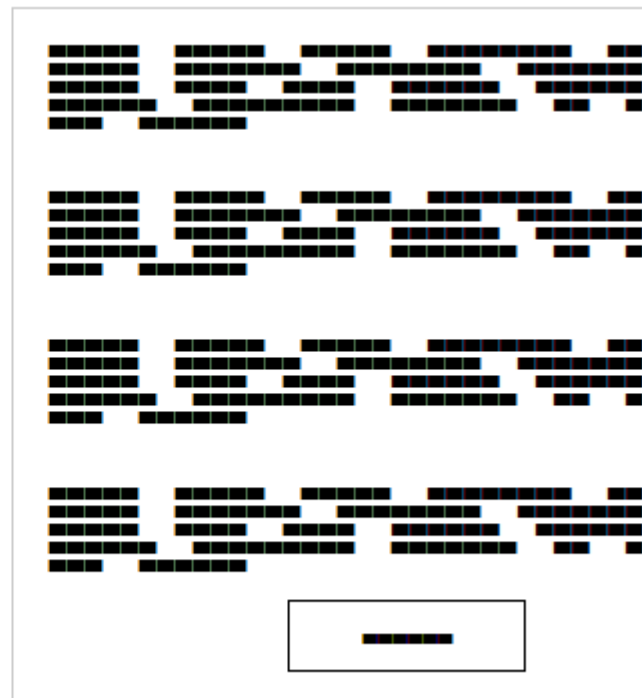
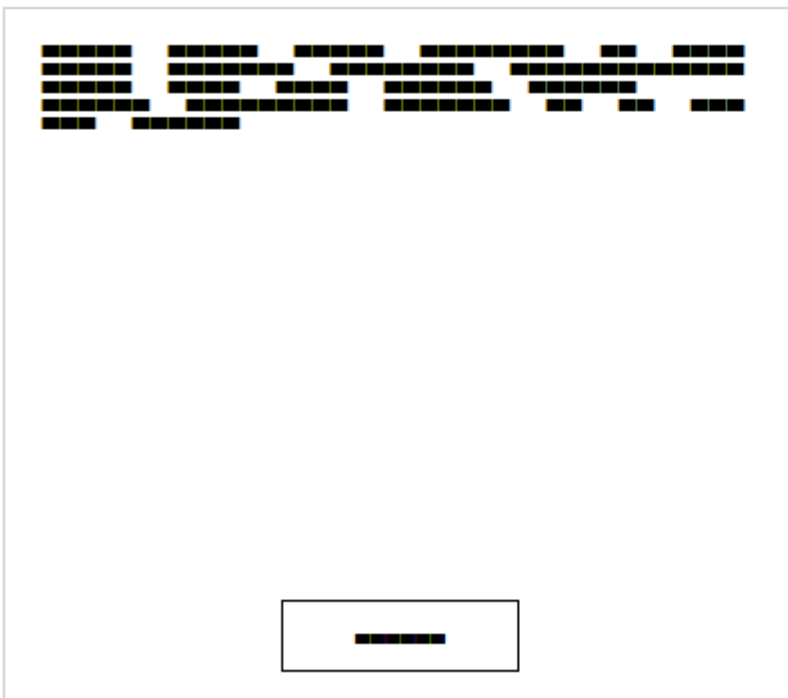
.content-body .content {
  flex: 1 1 auto;
  overflow: auto;
}
```

```
}  
  
.content-body .sidenav {  
  order: -1;  
  flex: 0 0 100px;  
  overflow: auto;  
}  
  
.content-body .ads {  
  flex: 0 0 100px;  
  overflow: auto;  
}  
  
.footer {  
  flex: 0 0 50px;  
}
```

демонстрация

Совершенно выровненные кнопки внутри карточек с flexbox

В наши дни это регулярный шаблон, позволяющий вертикально выровнять **вызов к действиям** внутри своих карточек:



Это может быть достигнуто с помощью специального трюка с `flexbox`

HTML

```
<div class="cards">  
  <div class="card">  
    <p>Lorem ipsum Magna proident ex anim dolor ullamco pariatur reprehenderit culpa esse enim  
    mollit labore dolore voluptate ullamco et ut sed qui minim.</p>  
  </div>  
</div>
```



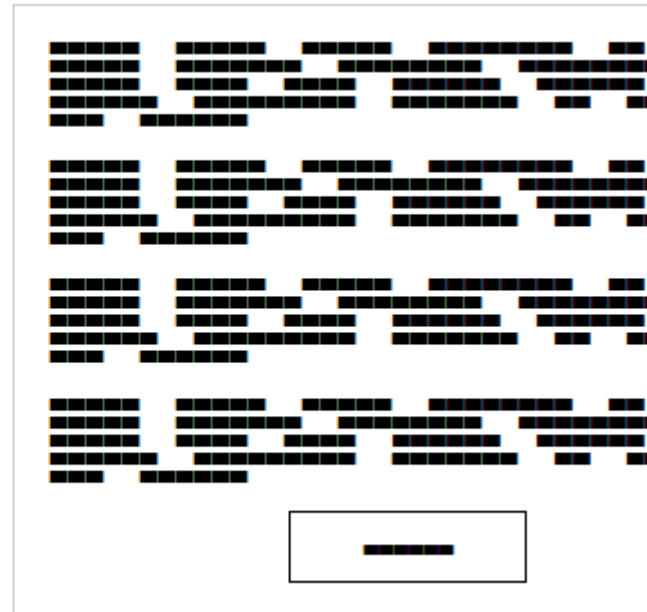
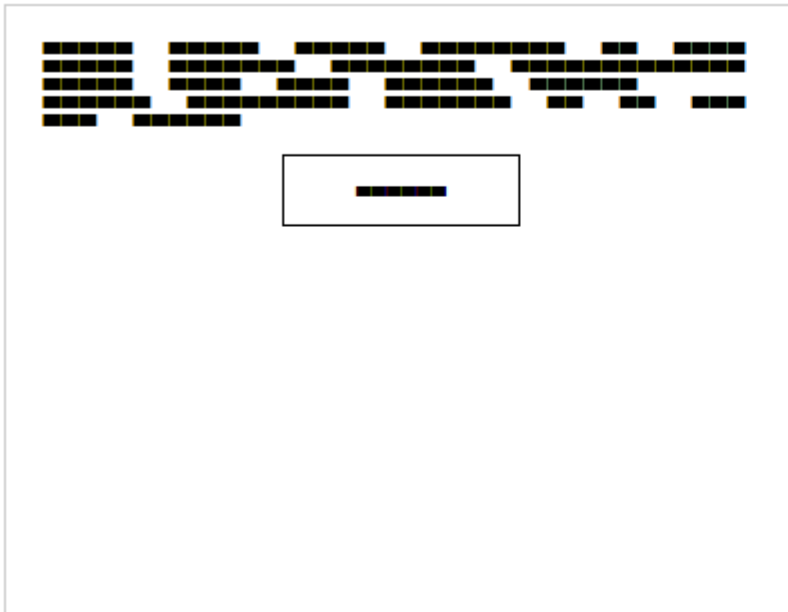
```
<p><button>Action</button></p>
</div>
<div class="card">
  <p>Lorem ipsum Magna proident ex anim dolor ullamco pariatur reprehenderit culpa esse enim mollit labore dolore voluptate ullamco et ut sed qui minim.</p>
  <p>Lorem ipsum Magna proident ex anim dolor ullamco pariatur reprehenderit culpa esse enim mollit labore dolore voluptate ullamco et ut sed qui minim.</p>
  <p>Lorem ipsum Magna proident ex anim dolor ullamco pariatur reprehenderit culpa esse enim mollit labore dolore voluptate ullamco et ut sed qui minim.</p>
  <p>Lorem ipsum Magna proident ex anim dolor ullamco pariatur reprehenderit culpa esse enim mollit labore dolore voluptate ullamco et ut sed qui minim.</p>
  <p><button>Action</button></p>
</div>
</div>
```

Прежде всего, мы используем CSS для применения `display: flex;` к контейнеру. Это создаст 2 столбца, равные по высоте, и содержимое, естественно, внутри него

CSS

```
.cards {
  display: flex;
}
.card {
  border: 1px solid #ccc;
  margin: 10px 10px;
  padding: 0 20px;
}
button {
  height: 40px;
  background: #fff;
  padding: 0 40px;
  border: 1px solid #000;
}
p:last-child {
  text-align: center;
}
```

Макет изменится и станет следующим:



Чтобы переместить кнопки в нижней части блока, нам нужно применить `display: flex;` К самой карте с направлением, установленным в `column`. После этого мы должны выбрать последний элемент внутри карты и установить `margin-top` в `auto`. Это подтолкнет последний абзац к нижней части карты и достигнет требуемого результата.

Финал CSS:

```
.cards {
  display: flex;
}
.card {
  border: 1px solid #ccc;
  margin: 10px 10px;
  padding: 0 20px;
  display: flex;
  flex-direction: column;
}
button {
  height: 40px;
  background: #fff;
  padding: 0 40px;
  border: 1px solid #000;
}
p:last-child {
  text-align: center;
  margin-top: auto;
}
```

Динамическое вертикальное и горизонтальное центрирование
(выравнивание, выравнивание-содержание)

Простой пример (центрирование одного

элемента)

HTML

```
<div class="aligner">
  <div class="aligner-item">...</div>
</div>
```

CSS

```
.aligner {
  display: flex;
  align-items: center;
  justify-content: center;
}

.aligner-item {
  max-width: 50%; /*for demo. Use actual width instead.*/
}
```

Вот [демонстрация](#) .

аргументация

Имущество	Значение	Описание
<code>align-items</code>	<code>center</code>	Это центрирует элементы вдоль оси, отличные от тех, которые указаны с помощью <code>flex-direction</code> , то есть вертикального центрирования для горизонтальной гибкой коробки и горизонтального центрирования для вертикальной гибкой коробки.
<code>justify-content</code>	<code>center</code>	Это центрирует элементы вдоль оси, заданной с помощью <code>flex-direction</code> . То есть, для горизонтальной (<code>flex-direction: row</code>) flexbox, она центрируется по горизонтали и для вертикальной гибкой коробки (<code>flex-direction: column</code>), это центрирует вертикально)

Примеры индивидуальной собственности

Все нижеследующие стили применяются к этой простой схеме:

```
<div id="container">
  <div></div>
  <div></div>
  <div></div>
</div>
```

Где #container - это flex-box .

Пример: justify-content: center на горизонтальной flexbox

CSS:

```
div#container {
  display: flex;
  flex-direction: row;
  justify-content: center;
}
```

Результат:



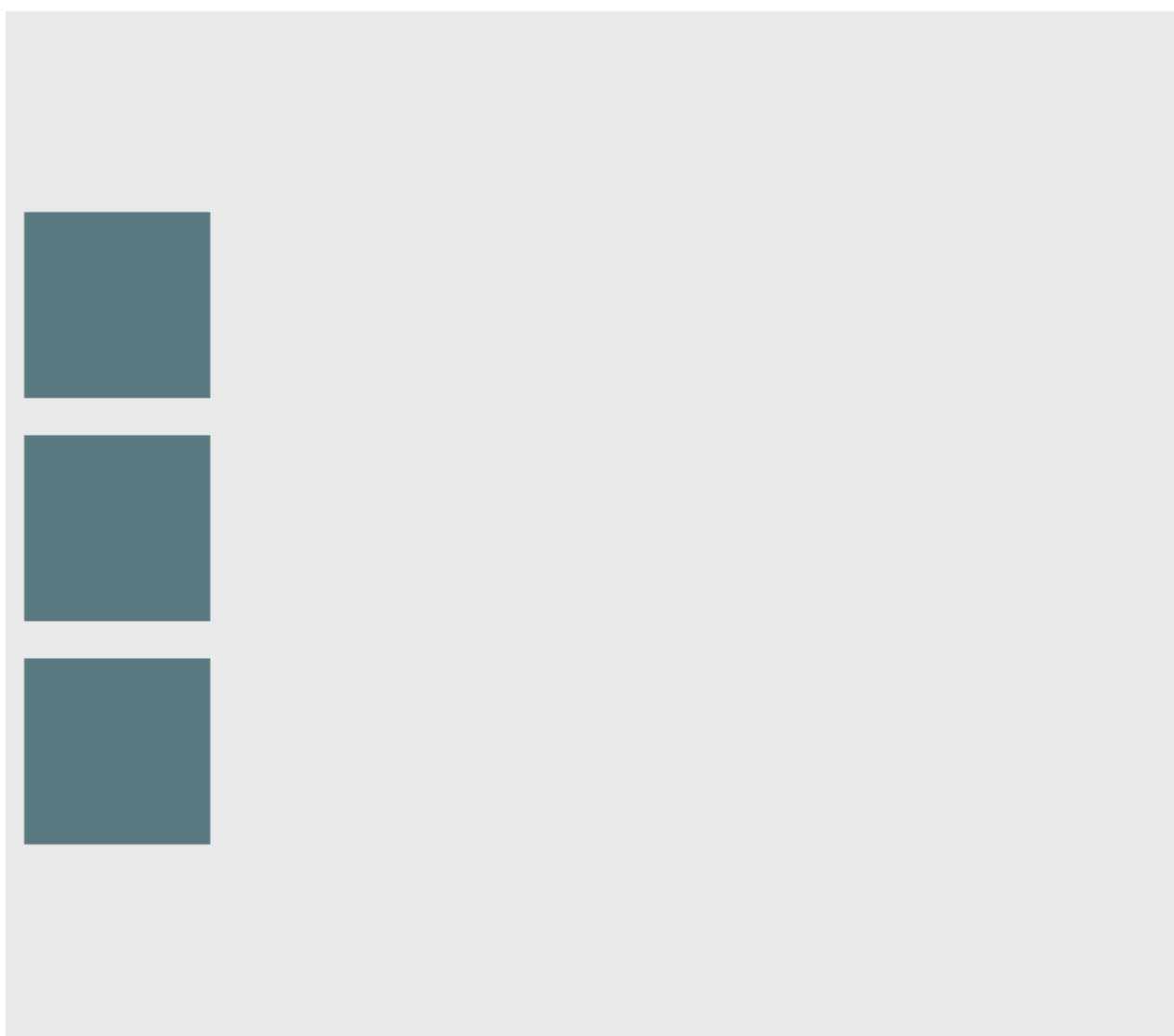
Вот [демонстрация](#) .

Пример: `justify-content: center` на вертикальной flexbox

CSS:

```
div#container {  
  display: flex;  
  flex-direction: column;  
  justify-content: center;  
}
```

Результат:



Вот [демонстрация](#) .

Пример: `align-content: center` на горизонтальной flexbox

CSS:

```
div#container {  
  display: flex;  
  flex-direction: row;  
  align-items: center;  
}
```

Результат:



Вот [демонстрация](#) .

Пример: `align-content: center` на вертикальной гибкой коробке

CSS:

```
div#container {  
  display: flex;  
  flex-direction: column;
```

```
align-items: center;
}
```

Результат:



Вот [демонстрация](#) .

Пример: комбинация для центрирования как на горизонтальной гибкой коробке

```
div#container {
  display: flex;
  flex-direction: row;
  justify-content: center;
  align-items: center;
}
```

Результат:



Вот [демонстрация](#) .

Пример: Комбинация для центрирования как на вертикальной гибкой коробке

```
div#container {  
  display: flex;  
  flex-direction: column;  
  justify-content: center;  
  align-items: center;  
}
```

Результат:



Вот [демонстрация](#) .

Такая же высота на вложенных контейнерах

Этот код гарантирует, что все вложенные контейнеры всегда будут одинаковой высоты. Это делается за счет того, что все вложенные элементы имеют ту же высоту, что и содержательный дед. См. Рабочий пример : <https://jsfiddle.net/3wwh7ewp/>

Этот эффект достигается благодаря свойству `align-items` , установленным на `stretch` по умолчанию.

HTML

```
<div class="container">
  <div style="background-color: red">
    Some <br />
    data <br />
    to make<br />
    a height <br />
```

```
</div>
  <div style="background-color: blue">
    Fewer <br />
    lines <br />
  </div>
</div>
```

CSS

```
.container {
  display: flex;
  align-items: stretch; // Default value
}
```

Примечание. [Не работает в IE версии до 10](#)

Оптимально подходят элементы для их контейнера

Одна из самых приятных функций flexbox - обеспечить оптимальное размещение контейнеров в их родительском элементе.

[Демо-версия](#) .

HTML:

```
<div class="flex-container">
  <div class="flex-item">1</div>
  <div class="flex-item">2</div>
  <div class="flex-item">3</div>
  <div class="flex-item">4</div>
  <div class="flex-item">5</div>
</div>
```

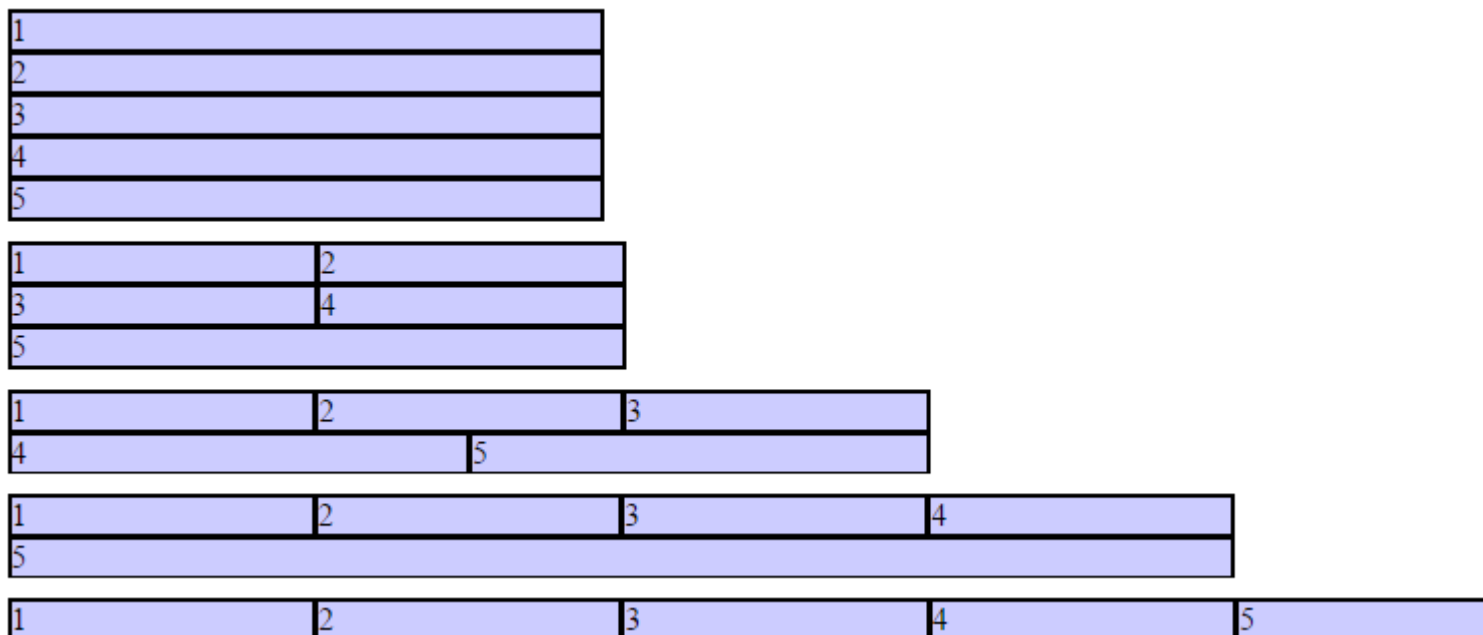
CSS:

```
.flex-container {
  background-color: #000;
  height: 100%;
  display: flex;
  flex-direction: row;
  flex-wrap: wrap;
  justify-content: flex-start;
  align-content: stretch;
  align-items: stretch;
}

.flex-item {
  background-color: #ccf;
  margin: 0.1em;
  flex-grow: 1;
  flex-shrink: 0;
  flex-basis: 200px; /* or % could be used to ensure a specific layout */
}
```

Результат:

Колонки адаптируются по мере изменения размера экрана.



Прочитайте Гибкая компоновка ящиков (Flexbox) онлайн:

<https://riptutorial.com/ru/css/topic/445/гибкая-компоновка-ящиков--flexbox->

глава 10: Единицы длины

Вступление

Измерение расстояния по CSS - это число, за которым следует единица длины (px, em, pc, in, ...)

CSS поддерживает несколько единиц измерения длины. Они абсолютные или относительные.

Синтаксис

- **единичная стоимость**
- 1em

параметры

Единица измерения	Описание
%	Определить размеры в терминах родительских объектов или текущего объекта, зависящего от свойства
Эм	Относительно размера шрифта элемента (2em означает в 2 раза размер текущего шрифта)
рем	Относительно размера шрифта корневого элемента
оч.сл.	Относительно 1% ширины окна просмотра *
В.Х.	Относительно 1% высоты окна просмотра *
Vmin	Относительно 1% меньшего размера окна просмотра
Vmax	Относительно 1% увеличенного размера viewport
см	см
мм	миллиметры
в	дюймы (1 дюйм = 96 пикселей = 2,54 см)
ПВ	пикселей (1px = 1 / 96th of 1in)

Единица измерения	Описание
пт	точек (1pt = 1/72 из 1in)
ПК	picas (1pc = 12 pt)
s	секунд (используется для анимаций и переходов)
Миз	миллисекунды (используются для анимаций и переходов)
бывший	Относительно x-высоты текущего шрифта
ч	Основываясь на ширине нулевого (0) символа
фр	дробная единица (используется для компоновки сетки CSS)

замечания

- Между числом и единицей не может быть пробелов. Однако, если значение равно 0, блок можно опустить.
- Для некоторых свойств CSS допустимы отрицательные длины.

Examples

Размер шрифта с rem

CSS3 вводит несколько новых единиц, включая блок `rem`, который обозначает «root em». Давайте посмотрим, как работает `rem`.

Во-первых, давайте посмотрим на различия между `em` и `rem`.

- **em** : Относительно размера шрифта родителя. Это вызывает проблему усугубления
- **rem** : Относительно размера шрифта корневого или `<html>` элемента. Это означает, что можно объявить один размер шрифта для элемента `html` и определить все единицы `rem` в процентах от этого.

Основная проблема с использованием `rem` для размера шрифта заключается в том, что значения несколько сложны в использовании. Ниже приведен пример некоторых распространенных размеров шрифтов, выраженных в единицах `rem`, при условии, что размер базы составляет 16 пикселей:

- $10px = 0,625rem$
- $12px = 0.75rem$
- $14px = 0.875rem$

- 16px = 1rem (базовый)
- 18px = 1.125rem
- 20px = 1.25rem
- 24px = 1.5rem
- 30px = 1,875rem
- 32px = 2rem

КОД:

3

```
html {
  font-size: 16px;
}

h1 {
  font-size: 2rem;          /* 32px */
}

p {
  font-size: 1rem;         /* 16px */
}

li {
  font-size: 1.5em;       /* 24px */
}
```

Создание масштабируемых элементов с использованием `rem` и `em`

3

Вы можете использовать `rem` определяемый `font-size` вашего тега `html` для стилей, путем установки их `font-size` в значение `rem` и использования `em` внутри элемента для создания элементов, которые масштабируются с вашим глобальным `font-size`.

HTML:

```
<input type="button" value="Button">
<input type="range">
<input type="text" value="Text">
```

Соответствующий CSS:

```
html {
  font-size: 16px;
}

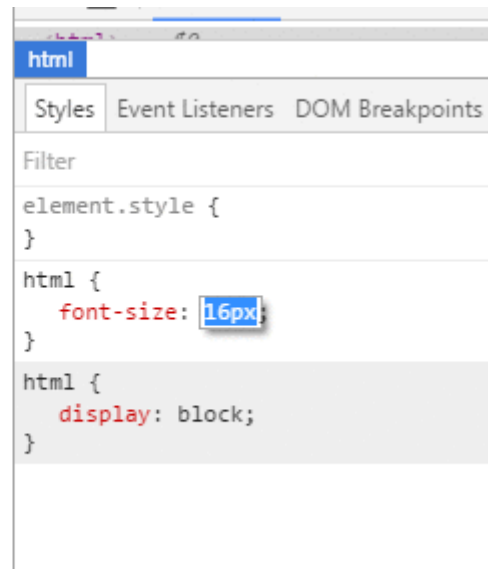
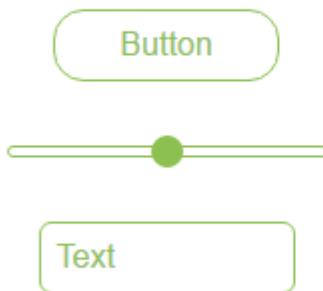
input[type="button"] {
  font-size: 1rem;
  padding: 0.5em 2em;
}

input[type="range"] {
```

```
font-size: 1rem;
width: 10em;
}

input[type=text] {
font-size: 1rem;
padding: 0.5em;
}
```

Возможный результат:



vh и vw

CSS3 представил две единицы для представления размера.

- **vh**, что означает `viewport height` составляет около 1% высоты окна просмотра
- **vw**, который обозначает `viewport width` составляет около 1% от ширины окна просмотра

3

```
div {
width: 20vw;
height: 20vh;
}
```

Выше размер для `div` занимает 20% от ширины и высоты окна просмотра

vmin и vmax

- **vmin**: относительно 1 процента меньшего размера **окна** просмотра
- **vmax**: относительно 1 процента большего размера окна просмотра

Другими словами, 1 vmin равно меньшему из 1 vh и 1 vw

$1 v_{\max}$ равен наибольшему из $1 vh$ и $1 vw$

Примечание : v_{\max} **не поддерживается** в:

- любая версия Internet Explorer
- Safari до версии 6.1

используя процент%

Один из полезных модулей при создании реагирующего приложения.

Его размер зависит от его родительского контейнера.

Уравнение:

(Ширина родительского контейнера) * (Процент (%)) = Выход

Например:

Родитель имеет ширину в **100 пикселей**, а у *ребенка* - **50%** .

На выходе ширина *ребенка* будет равна половине (50%) *родительских* , что составляет **50 пикселей** .

HTML

```
<div class="parent">
  PARENT
  <div class="child">
    CHILD
  </div>
</div>
```

CSS

```
<style>

*{
  color: #CCC;
}

.parent{
  background-color: blue;
  width: 100px;
}

.child{
  background-color: green;
  width: 50%;
}

</style>
```


ВЫХОД



Прочитайте Единицы длины онлайн: <https://riptutorial.com/ru/css/topic/864/единицы-длины>

глава 11: Запросы мультимедиа

Синтаксис

- `@media [not | only] mediatype и (медиа-функция) {/ * Правила CSS для применения * /}`

параметры

параметр	подробности
<code>mediatype</code>	(Необязательно) Это тип носителя. Может быть что -нибудь в пределах <code>all K screen</code> .
<code>not</code>	(Необязательно) Не применяется CSS для данного типа носителя и применяется ко всему остальному.
<code>media feature</code>	Логика для определения прецедента для CSS. Варианты, описанные ниже.
Медиа-функция	подробности
<code>aspect-ratio</code>	Описывает соотношение сторон целевой области отображения устройства вывода.
<code>color</code>	Указывает количество бит на цветной компонент устройства вывода. Если устройство не является цветным устройством, это значение равно нулю.
<code>color-index</code>	Указывает количество записей в таблице поиска цветов для устройства вывода.
<code>grid</code>	Определяет, является ли устройство вывода устройством сетки или растровым устройством.
<code>height</code>	Функция носителя высоты описывает высоту поверхности визуализации выходного устройства.
<code>max-width</code>	CSS не будет применяться на ширине экрана шире, чем указано.
<code>min-width</code>	CSS не будет применяться на ширине экрана, уже указанной выше.
<code>max-height</code>	CSS не будет применяться на высоте экрана выше указанного.

параметр	подробности
<code>min-height</code>	CSS не будет применяться на высоте экрана, меньшей указанной.
<code>monochrome</code>	Указывает количество бит на пиксель на монохромном (серого) устройстве.
<code>orientation</code>	CSS будет отображаться только в том случае, если устройство использует указанную ориентацию. См. Примечания для более подробной информации.
<code>resolution</code>	Указывает разрешение (плотность пикселей) устройства вывода.
<code>scan</code>	Описывает процесс сканирования телевизионных выходных устройств.
<code>width</code>	Средство ширины носителя описывает ширину поверхности отображения выходного устройства (например, ширину окна документа или ширину окна страницы на принтере).
Устаревшие функции	подробности
<code>device-aspect-ratio</code>	Deprecated CSS будет отображаться только на устройствах, соотношение высоты и ширины которых соответствует указанному соотношению. Это deprecated функция и не гарантируется работа.
<code>max-device-width</code>	Deprecated То же, что и <code>max-width</code> но измеряет <code>max-width</code> физического экрана, а не ширину экрана браузера.
<code>min-device-width</code>	Deprecated То же, что и <code>min-width</code> но измеряет <code>min-width</code> физического экрана, а не ширину экрана браузера.
<code>max-device-height</code>	Deprecated То же, что и <code>max-height</code> но измеряет ширину физического экрана, а не ширину экрана браузера.
<code>min-device-height</code>	Deprecated То же, что и <code>min-height</code> но измеряет ширину физического экрана, а не ширину экрана браузера.

замечания

Медиа-запросы поддерживаются во всех современных браузерах, включая Chrome, Firefox, Opera и Internet Explorer 9 и выше.

Важно отметить, что функция `orientation` носителей не ограничивается мобильными

устройствами. Он основан на ширине и высоте окна просмотра (а не на окнах или устройствах).

Ландшафтный режим - это когда ширина области просмотра больше высоты видового экрана.

Портретный режим - это когда высота видового экрана больше ширины видового экрана.

Обычно это означает, что настольный монитор находится в ландшафтном режиме, но иногда это портрет.

В большинстве случаев мобильные устройства сообщают о своем разрешении, а не о реальном размере пикселей, которые могут отличаться из-за плотности пикселей. Для того, чтобы заставить их сообщать их реальный размер пикселя добавить следующее внутри `head` тега:

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

Examples

Основной пример

```
@media screen and (min-width: 720px) {  
  body {  
    background-color: skyblue;  
  }  
}
```

Вышеупомянутый медиа-запрос определяет два условия:

1. Страница должна просматриваться на обычном экране (не на печатной странице, проекторе и т. Д.).
2. Ширина порта просмотра пользователя должна быть не менее 720 пикселей.

Если эти условия выполнены, стили внутри медиа-запроса будут активны, а цвет фона на странице будет синим.

Запросы мультимедиа применяются динамически. Если на странице загружаются условия, указанные в запросе на медиа, CSS будет применяться, но будет немедленно отключен, если условия перестанут выполняться. И наоборот, если условия изначально не выполняются, CSS не будет применяться до тех пор, пока не будут выполнены указанные условия.

В нашем примере, если ширина порта представления пользователя первоначально больше 720 пикселей, но пользователь сжимает ширину браузера, цвет фона перестанет быть

голубым, как только пользователь изменит размер порта представления на менее чем 720 пикселей в ширина.

Использовать по тегу ссылки

```
<link rel="stylesheet" media="min-width: 600px" href="example.css" />
```

Эта таблица стилей по-прежнему загружается, но применяется только на устройствах с шириной экрана более 600 пикселей.

MediaType

Запросы СМИ имеют необязательный параметр `mediatype`. Этот параметр устанавливается непосредственно после `@media` декларации (`@media mediatype`), например:

```
@media print {  
  html {  
    background-color: white;  
  }  
}
```

Приведенный выше код CSS придаст `HTML` элементу DOM белый цвет фона при печати.

Параметр `mediatype` имеет необязательный или `not only` префикс, который будет применять стили ко всему, кроме указанного медиатипа *или* только указанного типа носителя, соответственно. Например, следующий пример кода применит стиль к каждому типу носителя, кроме `print`.

```
@media not print {  
  html {  
    background-color: green;  
  }  
}
```

Точно так же, просто показывая это только на экране, это можно использовать:

```
@media only screen {  
  .fadeInEffects {  
    display: block;  
  }  
}
```

Список `mediatype` может быть лучше понят со следующей таблицей:

Тип носителя	Описание
all	Применить ко всем устройствам

Тип носителя	Описание
screen	Компьютеры по умолчанию
print	Принтеры в целом. Используется для создания печатных версий веб-сайтов
handheld	КПК, мобильные телефоны и карманные устройства с небольшим экраном
projection	Для проектной презентации, например, проекторы
aural	Речевые системы
braille	Брайлевские тактильные устройства
embossed	Вышитые брайлевские принтеры
tv	Телевизионные устройства
tty	Устройства с сеткой символов с фиксированным шагом. Терминалы, портативные устройства.

Использование медиа-запросов для разных размеров экрана

Часто, отзывчивый веб-дизайн включает медиа-запросы, которые являются блоками CSS, которые выполняются только в том случае, если условие выполнено. Это полезно для гибкого веб-дизайна, потому что вы можете использовать медиа-запросы, чтобы указать разные стили CSS для мобильной версии вашего сайта в сравнении с настольной версией.

```
@media only screen and (min-width: 300px) and (max-width: 767px) {
  .site-title {
    font-size: 80%;
  }

  /* Styles in this block are only applied if the screen size is atleast 300px wide, but no
  more than 767px */
}

@media only screen and (min-width: 768px) and (max-width: 1023px) {
  .site-title {
    font-size: 90%;
  }

  /* Styles in this block are only applied if the screen size is atleast 768px wide, but no
  more than 1023px */
}

@media only screen and (min-width: 1024px) {
  .site-title {
```

```
    font-size: 120%;
  }

  /* Styles in this block are only applied if the screen size is over 1024px wide. */
}
```

Ширина против Viewport

Когда мы используем «ширину» с медиа-запросами, важно правильно установить метатеги. Основной метатег выглядит так, и его нужно поместить внутри `<head>` .

```
<meta name="viewport" content="width=device-width,initial-scale=1">
```

Почему это важно?

На основе определения MDN «ширина»

Средство ширины носителя описывает ширину поверхности отображения выходного устройства (например, ширину окна документа или ширину окна страницы на принтере).

Что это значит?

View-port - это ширина самого устройства. Если ваше разрешение экрана говорит, что разрешение 1280 x 720, ширина вашего порта просмотра «1280px».

Чаще всего многие устройства выделяют различное количество пикселей для отображения одного пикселя. Например, iPhone 6 Plus имеет разрешение 1242 x 2208. Но фактическая ширина видового экрана и высота видового экрана - 414 x 736. Это означает, что для создания 1 пикселя используются 3 пикселя.

Но если вы не установили `meta` правильно, он попытается показать вашу веб-страницу с ее родным разрешением, что приведет к уменьшенному просмотру (меньшие тексты и изображения).

Медиа-запросы для экранов сетчатки и не Retina

Хотя это работает только для браузеров на базе WebKit, это полезно:

```
/* ----- Non-Retina Screens ----- */
@media screen
  and (min-width: 1200px)
  and (max-width: 1600px)
  and (-webkit-min-device-pixel-ratio: 1) {
}

/* ----- Retina Screens ----- */
@media screen
  and (min-width: 1200px)
```

```
and (max-width: 1600px)
and (-webkit-min-device-pixel-ratio: 2)
and (min-resolution: 192dpi) {
}
```

Исходная информация

На дисплее есть два типа пикселей. Один - это логические пиксели, а другой - физические пиксели. В основном, физические пиксели всегда остаются неизменными, потому что они одинаковы для всех устройств отображения. Логические пиксели изменяются в зависимости от разрешения устройств для отображения более качественных пикселей. Соотношение пикселей устройства - это соотношение между физическими пикселями и логическими пикселями. Например, MacBook Pro Retina, iPhone 4 и выше сообщают о соотношении пикселей устройства в 2, поскольку физическое линейное разрешение вдвое превышает логическое разрешение.

Причина, по которой это работает только с браузерами на основе WebKit, объясняется:

- Префикс поставщика `-webkit-` перед правилом.
- Это не было реализовано в других машинах, кроме WebKit и Blink.

Терминология и структура

Медиа-запросы позволяют применять правила CSS на основе типа устройства / носителя (например, экрана, печати или портативного компьютера), называемого **типом носителя**, дополнительные аспекты устройства описаны с такими **мультимедийными функциями**, как доступность цветов или размеров видовых экранов.

Общая структура медиа-запроса

```
@media [...] {
  /* One or more CSS rules to apply when the query is satisfied */
}
```

Медиа-запрос, содержащий тип носителя

```
@media print {
  /* One or more CSS rules to apply when the query is satisfied */
}
```

Медиа-запрос, содержащий тип носителя

и функцию мультимедиа

```
@media screen and (max-width: 600px) {  
  /* One or more CSS rules to apply when the query is satisfied */  
}
```

Медиа-запрос, содержащий функцию мультимедиа (и неявный тип носителя «все»)

```
@media (orientation: portrait) {  
  /* One or more CSS rules to apply when the query is satisfied */  
}
```

Медиа-запросы и IE8

Медиа-запросы вообще не поддерживаются в IE8 и ниже.

Обходное решение на основе Javascript

Чтобы добавить поддержку IE8, вы можете использовать один из нескольких решений JS. Например, **ответ** может быть добавлен для добавления поддержки медиа-запросов для IE8 только с помощью следующего кода:

```
<!--[if lt IE 9]>  
<script  
  src="respond.min.js">  
</script>  
<![endif]-->
```

CSS Mediaqueries - это другая библиотека, которая делает то же самое. Код для добавления этой библиотеки в ваш HTML-код будет идентичным:

```
<!--[if lt IE 9]>  
<script  
  src="css3-mediaqueries.js">  
</script>  
<![endif]-->
```

Альтернативный вариант

Если вам не нравится решение на основе JS, вы также должны подумать о добавлении таблицы стилей IE <9, в которой вы настраиваете свой стиль для IE <9. Для этого вы должны добавить следующий код HTML в свой код:

```
<!--[if lt IE 9]>
<link rel="stylesheet" type="text/css" media="all" href="style-ielt9.css"/>
<![endif]-->
```

Замечания :

Технически это еще одна альтернатива: использование **CSS-хаков** для IE <9. Он имеет то же влияние, что и таблица стилей IE <9, но для этого вам не нужна отдельная таблица стилей. Однако я не рекомендую этот параметр, поскольку они создают недопустимый код CSS (что является лишь одной из причин, по которым использование хаков CSS обычно не одобряется сегодня).

Прочитайте Запросы мультимедиа онлайн: <https://riptutorial.com/ru/css/topic/317/запросы-мультимедиа>

глава 12: Запросы функций

Синтаксис

- `@supports [условие] { / * Правила CSS для применения * / }`

параметры

параметр	подробности
<code>(property: value)</code>	Вычисляет true, если браузер может обрабатывать правило CSS. Требуется скобка вокруг правила.
<code>and</code>	Возвращает true, только если выполняются предыдущие и следующие условия.
<code>not</code>	Отрицает следующее условие
<code>or</code>	Возвращает true, если либо предыдущее, либо следующее условие истинно.
<code>(...)</code>	Условия групп

замечания

Функция обнаружения `@supports` с помощью `@supports` поддерживается в Edge, Chrome, Firefox, Opera и Safari 9 и выше.

Examples

Основное использование `@supports`

```
@supports (display: flex) {  
  /* Flexbox is available, so use it */  
  .my-container {  
    display: flex;  
  }  
}
```

С точки зрения синтаксиса `@supports` очень похож на `@media`, но вместо определения размера экрана и ориентации, `@supports` будет определять, может ли браузер обрабатывать заданное правило CSS.

Вместо того, чтобы делать что-то вроде `@supports (flex)`, обратите внимание, что это правило `@supports (display: flex)`.

Обнаружение признаков цепочки

Для того, чтобы обнаружить несколько функций одновременно, использовать `and` оператор.

```
@supports (transform: translateZ(1px)) and (transform-style: preserve-3d) and (perspective: 1px) {
  /* Probably do some fancy 3d stuff here */
}
```

Существует также оператор `or` оператор, а `not` оператор:

```
@supports (display: flex) or (display: table-cell) {
  /* Will be used if the browser supports flexbox or display: table-cell */
}
@supports not (-webkit-transform: translate(0, 0, 0)) {
  /* Will *not* be used if the browser supports -webkit-transform: translate(...) */
}
```

Для получения максимального опыта `@supports` попробуйте группировать логические выражения с круглыми скобками:

```
@supports ((display: block) and (zoom: 1)) or ((display: flex) and (not (display: table-cell))) or (transform: translateX(1px)) {
  /* ... */
}
```

Это будет работать, если браузер

1. Поддерживает `display: block` И `zoom: 1`, или
2. Поддерживает `display: flex` AND NOT `display: table-cell`, или
3. Поддерживает `transform: translateX(1px)`.

Прочитайте Запросы функций онлайн: <https://riptutorial.com/ru/css/topic/5024/запросы-функций>

глава 13: Каскадирование и специфика

замечания

Специфика CSS намеревается продвигать краткость кода, позволяя автору определять некоторые общие правила форматирования для широкого набора элементов, а затем переопределять их для определенного подмножества.

Examples

Каскадный

Каскадирование и специфичность используются вместе для определения конечного значения свойства стилизации CSS. Они также определяют механизмы разрешения конфликтов в наборах правил CSS.

Порядок загрузки CSS

Стили считываются из следующих источников, в следующем порядке:

1. Таблица стилей User Agent (стили, предоставленные поставщиком браузера)
2. Пользовательская таблица стилей (дополнительный стиль, который пользователь установил в своем браузере)
3. Авторская таблица стилей (автор здесь означает создателя веб-страницы / веб-сайта)
 - Возможно, один или несколько файлов `.css`
 - В элементе `<style>` документа HTML
4. Встроенные стили (в атрибуте `style` для HTML-элемента)

Браузер будет искать соответствующий стиль (ы) при рендеринге элемента.

Как разрешаются конфликты?

Когда только один набор правил CSS пытается установить стиль для элемента, тогда конфликта нет, и этот набор правил используется.

Когда несколько наборов правил находятся с конфликтующими настройками, сначала устанавливаются правила Спецификации, а затем каскадные правила используются для определения того, какой стиль использовать.

Пример 1 - Правила специфичности

```
.mystyle { color: blue; } /* specificity: 0, 0, 1, 0 */  
div { color: red; } /* specificity: 0, 0, 0, 1 */
```

```
<div class="mystyle">Hello World</div>
```

Каким будет цвет текста? (наведите указатель мыши на ответ)

синий

Сначала применяются правила специфичности, и тот, который обладает наивысшей спецификой «выигрывает».

Пример 2 - Каскадные правила с идентичными селекторами

Внешний файл css

```
.class {  
  background: #FFF;  
}
```

Внутренний css (в HTML-файле)

```
<style>  
.class {  
  background: #000;  
}  
</style>
```

В этом случае, когда у вас одинаковые селектор, каскад срабатывает и определяет, что последний загружается «выигрывает».

Пример 3 - Правила каскада после правил специфичности

```
body > .mystyle { background-color: blue; } /* specificity: 0, 0, 1, 1 */  
.otherstyle > div { background-color: red; } /* specificity: 0, 0, 1, 1 */
```

```
<body class="otherstyle">  
  <div class="mystyle">Hello World</div>  
</body>
```

Каким будет цвет фона?

красный

После применения правил специфика все еще существует конфликт между синим и красным, поэтому каскадные правила применяются поверх правил специфичности. Каскадирование смотрит на порядок загрузки правил, будь то внутри того же `.css` файла или в коллекции источников стиля. Последний загруженный переопределяет любые предыдущие. В этом случае `.otherstyle > div` правило `.otherstyle > div`.

Заключительная записка

- Селекторная специфика всегда имеет приоритет.
- Таблицы разборки стилей.
- Встроенные стили козырьки все.

Важное заявление!

`!important` объявление используется, чтобы переопределить обычную специфику в таблице стилей, указав более высокий приоритет правилу. Его использование: `property : value !important;`

```
#mydiv {
  font-weight: bold !important;      /* This property won't be overridden
                                     by the rule below */
}

#outerdiv #mydiv {
  font-weight: normal;               /* #mydiv font-weight won't be set to normal
                                     even if it has a higher specificity because
                                     of the !important declaration above */
}
```

Настоятельно рекомендуется избегать использования `!important` (если это абсолютно необходимо), потому что это нарушит естественный поток правил CSS, который может привести к неопределенности в вашей таблице стилей. Также важно отметить, что, когда к одному и тому же правилу применяются несколько `!important` деклараций для одного элемента, тот, который имеет более высокую специфичность, будет применен.

Вот несколько примеров, где использование `!important` объявления может быть оправдано:

- Если ваши правила не должны быть переопределены любым встроенным стилем элемента, который написан внутри атрибута `style` элемента `html`.
- Чтобы предоставить пользователю больше контроля над доступностью веб-страниц, например, увеличивать или уменьшать размер шрифта, переопределяя стиль автора, используя `!important`.
- Для тестирования и отладки с использованием элемента проверки.

Смотрите также:

- **W3C - 6 Назначение значений свойств, каскадирования и наследования - 6.4.2!**
Важные правила

Вычисление специфики выбора

Каждый отдельный CSS-селектор имеет свое значение специфичности. Каждый селектор в последовательности увеличивает общую специфичность последовательности.

Селекторы попадают в одну из трех различных групп специфичности: *A*, *B* и *c*. Когда несколько селекторных последовательностей выбирают данный элемент, браузер использует стили, применяемые последовательностью с наивысшей общей спецификой.

группа	Состоящий из	Примеры
	селекторы id	#foo
<i>B</i>	селекторы классов Селекторы атрибутов псевдоклассы	.bar [title] , [colspan="2"] :hover :nth-child(2)
<i>c</i>	селектор типа псевдо-элементы	div , li ::before , ::first-letter

Группа *A* является наиболее конкретной, за ней следует группа *B*, а затем группа *c*.

Универсальный селектор (`*`) и комбинаторы (например, `>` и `~`) не имеют специфики.

Пример 1: Специфичность различных селекторных последовательностей

```
#foo #baz {} /* a=2, b=0, c=0 */
#foo.bar {} /* a=1, b=1, c=0 */
#foo {} /* a=1, b=0, c=0 */
.bar:hover {} /* a=0, b=2, c=0 */
div.bar {} /* a=0, b=1, c=1 */
:hover {} /* a=0, b=1, c=0 */
[title] {} /* a=0, b=1, c=0 */
.bar {} /* a=0, b=1, c=0 */
div ul + li {} /* a=0, b=0, c=3 */
p::after {} /* a=0, b=0, c=2 */
*::before {} /* a=0, b=0, c=1 */
::before {} /* a=0, b=0, c=1 */
```



```
div {}          /* a=0, b=0, c=1 */
* {}           /* a=0, b=0, c=0 */
```

Пример 2. Какая специфика используется браузером

Представьте себе следующую реализацию CSS:

```
#foo {
  color: blue;
}

.bar {
  color: red;
  background: black;
}
```

Здесь мы имеем селектор идентификаторов, который объявляет `color` как *синий*, и селектор классов, который объявляет `color` *красным* и `background` *черным*.

`#foo` элемента с идентификатором `#foo` и классом `.bar` будут выбраны обоими декларациями. Селекторы идентификаторов имеют специфику группы *A*, а селекторы классов имеют специфику группы *B*. Селектор ID перевешивает любое количество селекторов классов. Из-за этого `color:blue;` из селектора `#foo` и `background:black;` из селектора `.bar` будет применен элемент. Более высокая специфичность селектора ID приведет к тому, что браузер игнорирует `.bar color` селектора `.bar`.

Теперь представьте себе другую реализацию CSS:

```
.bar {
  color: red;
  background: black;
}

.baz {
  background: white;
}
```

Здесь у нас есть два класса селекторов; один из которых объявляет `color` *красным* и `background` *черным*, а другой объявляет `background` *белым*.

`.bar` этими объявлениями будут влиять элементы с `.bar` и `.baz`, однако проблема, которую мы имеем сейчас, состоит в том, что оба `.bar` и `.baz` имеют идентичную специфичность группы *B*. Каскадный характер CSS разрешает это для нас: поскольку `.baz` определяется *после* `.bar`, наш элемент заканчивается *красным* `color` с `.bar` а на *белом* `background` с `.baz`.

Пример 3: Как управлять спецификой

Последний фрагмент из примера 2, приведенный выше, можно использовать, чтобы обеспечить, чтобы наше `.bar color` селектора класса `.bar` использовалось вместо селектора классов `.baz`.

```
.bar {}          /* a=0, b=1, c=0 */
.baz {}          /* a=0, b=1, c=0 */
```

Наиболее распространенным способом достижения этого было бы выяснить, какие другие селекторы могут быть применены к последовательности селектора `.bar`. Например, если `.bar` класс был только когда - либо применительно к `span` элементы, мы могли бы модифицировать `.bar` селектор `span.bar`. Это придало бы ему новую специфику Group C, которая могла бы переопределить `.baz` селектора `.baz`:

```
span.bar {}     /* a=0, b=1, c=1 */
.baz {}         /* a=0, b=1, c=0 */
```

Однако не всегда возможно найти другой общий селектор, который разделяется между любым элементом, который использует класс `.bar`. Из-за этого CSS позволяет нам дублировать селекторы для повышения специфичности. Вместо простого `.bar`, мы можем использовать `.bar.bar` вместо этого (см. [Граматику селекторов, рекомендацию W3C](#)). Это по-прежнему выбирает любой элемент с классом `.bar`, но теперь имеет двойную специфику Group B:

```
.bar.bar {}     /* a=0, b=2, c=0 */
.baz {}         /* a=0, b=1, c=0 */
```

!important и встроенные декларации стиля

Считается, что `!important` флаг в объявлении стиля и стилях, объявленных атрибутом `style` HTML, имеет большую специфичность, чем любой селектор. Если они существуют, декларация стиля, на которую они влияют, будет отменять другие объявления независимо от их специфики. То есть, если у вас есть более одного объявления, которое содержит флаг `!important` для того же свойства, который применяется к одному и тому же элементу. Тогда нормальные правила специфичности будут применяться к этим свойствам в отношении друг к другу.

Поскольку они полностью переопределяют специфику, использование `!important` в большинстве случаев недооценивается. Его следует использовать как можно меньше. Чтобы код CSS был эффективным и поддерживаемым в долгосрочной перспективе, почти всегда лучше повышать специфику окружающего селектора, чем использовать `!important`.

Одно из тех редких исключений, где `!important` не `!important`, заключается в реализации общих вспомогательных классов, таких как `.hidden` или `.background-yellow`, которые должны всегда переопределять один или несколько свойств везде, где они встречаются. И даже тогда вам нужно знать, что вы делаете. Последнее, что вы хотите, при написании

поддерживающего CSS, - иметь `!important` флаги в вашем CSS.

Заключительная записка

Распространенным заблуждением о специфичности CSS является то, что значения группы *A*, *B* и *c* должны быть объединены друг с другом ($a=1, b=5, c=1 \Rightarrow 151$). Это **не** так. Если бы это было так, то 20 селекторов группы *B* или *c* было бы достаточно, чтобы переопределить один селектор группы *A* или *B* соответственно. Эти три группы следует рассматривать как индивидуальные уровни специфичности. Специфичность не может быть представлена одним значением.

При создании таблицы стилей CSS вы должны поддерживать самую низкую специфичность. Если вам нужно сделать специфика немного выше, чтобы перезаписать другой метод, сделайте его выше, но как можно ниже, чтобы сделать его выше. Вам не нужно иметь такой селектор:

```
body.page header.container nav div#main-nav li a {}
```

Это делает будущие изменения более жесткими и загрязняет эту страницу css.

Вы можете рассчитать специфику вашего селектора [здесь](#)

Более сложный пример специфичности

```
div {
  font-size: 7px;
  border: 3px dotted pink;
  background-color: yellow;
  color: purple;
}

body.mystyle > div.myotherstyle {
  font-size: 11px;
  background-color: green;
}

#elmnt1 {
  font-size: 24px;
  border-color: red;
}

.mystyle .myotherstyle {
  font-size: 16px;
  background-color: black;
  color: red;
}
```

```
<body class="mystyle">
  <div id="elmnt1" class="myotherstyle">
    Hello, world!
```

```
</div>
</body>
```

Какими границами, цветами и размерами шрифта будет текст?

размер шрифта:

`font-size: 24;` , так как `#elmnt1` правил `#elmnt1` имеет самую высокую специфичность для рассматриваемого `<div>` , каждое свойство здесь задано.

граница:

`border: 3px dotted red;` , `red` цвет границы берется из `#elmnt1` правил `#elmnt1` , так как он имеет самую высокую специфичность. Другие свойства границы, границы и стиля границы - из набора правил `div` .

фоновый цвет:

`background-color: green;` , `background-color` устанавливается в наборах `div` , `body.mystyle > div.myotherstyle` и `.mystyle .myotherstyle` . Специфические значения (0, 0, 1) против (0, 2, 2) против (0, 2, 0), поэтому средний выигрывает.

цвет:

`color: red;` , Цвет задается в `.mystyle .myotherstyle` правил `div` и `.mystyle .myotherstyle` . Последняя имеет более высокую специфичность (0, 2, 0) и «выигрывает».

Прочитайте Каскадирование и специфика онлайн: <https://riptutorial.com/ru/css/topic/450/каскадирование-и-специфика>

глава 14: Колонны

Синтаксис

- `column-count`: auto | number | inherit | initial | unset;
- ширина столбца: авто | длина;
- `column`: [`column-width`] | [`column-count`];
- `column-span`: none | all | inherit | initial | unset;
- `column-gap`: normal | length | inherit | initial | unset;
- `column-fill`: auto | balance | inherit | initial | unset;
- `column-rule-color`: color | inherit | initial | unset;
- `column-rule-style`: none | hidden | пунктир | пунктир | твердый | двойной | паз | гребень | вставка | начало | наследовать | начальный | unset;
- `column-rule-width`: thin | medium | thick | length | inherit | initial | unset;
- `column-rule`: [`column-rule-width`] | [`column-rule-style`] | [`column-rule-color`];
- `break-after`: auto | always | left | right | recto | verso | page | column | region | avoid | avoid-page | avoid-column | avoid-region;
- `break-before`: auto | always | left | right | recto | verso | page | column | region | avoid | avoid-page | avoid-column | avoid-region;
- `break-in`: auto | избегать | избегать-страницы | избегать-столбца | избегать области;

Examples

Простой пример (количество столбцов)

Многострочный макет CSS упрощает создание нескольких столбцов текста.

Код

```
<div id="multi-columns">Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum</div>
```

```
.multi-columns {  
  -moz-column-count: 2;  
  -webkit-column-count: 2;  
  column-count: 2;  
}
```

Результат

Stack Overflow is a privately held website, the flagship site of the Stack Exchange Network,[4][5][6] created in 2008 by Jeff Atwood and Joel Spolsky. [7][8] It was created to be a more open alternative to earlier Q&A sites such as Experts-Exchange. The name for the website was chosen by voting in April 2008 by readers of Coding Horror, Atwood's popular programming blog.

The website serves as a platform for users to ask and answer questions, and, through membership and active participation, to vote questions and

answers and answers wiki or Di Overflow "badges" awarded receiving given to a badges fo [14] which gamificat or forum. licensed Attribute-

Ширина колонки

Свойство `column-width` задает минимальную ширину столбца. Если количество `column-count` не определено, браузер будет делать столько столбцов, сколько подходит для доступной ширины.

Код:

```
<div id="multi-columns">
  Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum
</div>
```

```
.multi-columns {
  -moz-column-width: 100px;
  -webkit-column-width: 100px;
  column-width: 100px;
}
```

Результат

Stack Overflow is a privately held website, the flagship site of the Stack Exchange Network,[4][5][6] created in 2008 by Jeff Atwood and Joel Spolsky. [7][8] It was created to be a more open alternative to earlier Q&A sites such as Experts-	Exchange. The name for the website was chosen by voting in April 2008 by readers of Coding Horror, Atwood's popular programming blog. The website serves as a platform for users to ask and answer	questions, and, through membership and active participation, to vote questions and answers up or down and edit questions and answers in a fashion similar to a wiki or Digg.[13] Users of Stack Overflow can earn reputation	points and "badges"; for example, a person is awarded 10 reputation points for receiving an "up" vote on an answer given to a question, and can receive badges for their valued contributions, [14] which represents a kind of	gamification of the traditional Q&A site or forum. All user-generated content is licensed under a Creative Commons Attribute-ShareAlike license.
--	--	--	--	--

Прочитайте Колонны онлайн: <https://riptutorial.com/ru/css/topic/3042/колонны>

глава 15: Комментарии

Синтаксис

- /* Комментарий */

замечания

- Комментарии в CSS всегда начинаются с /* и заканчиваются на */
- Комментарии не могут быть вложенными

Examples

Одна линия

```
/* This is a CSS comment */
div {
  color: red; /* This is a CSS comment */
}
```

Несколько строк

```
/*
  This
  is
  a
  CSS
  comment
*/
div {
  color: red;
}
```

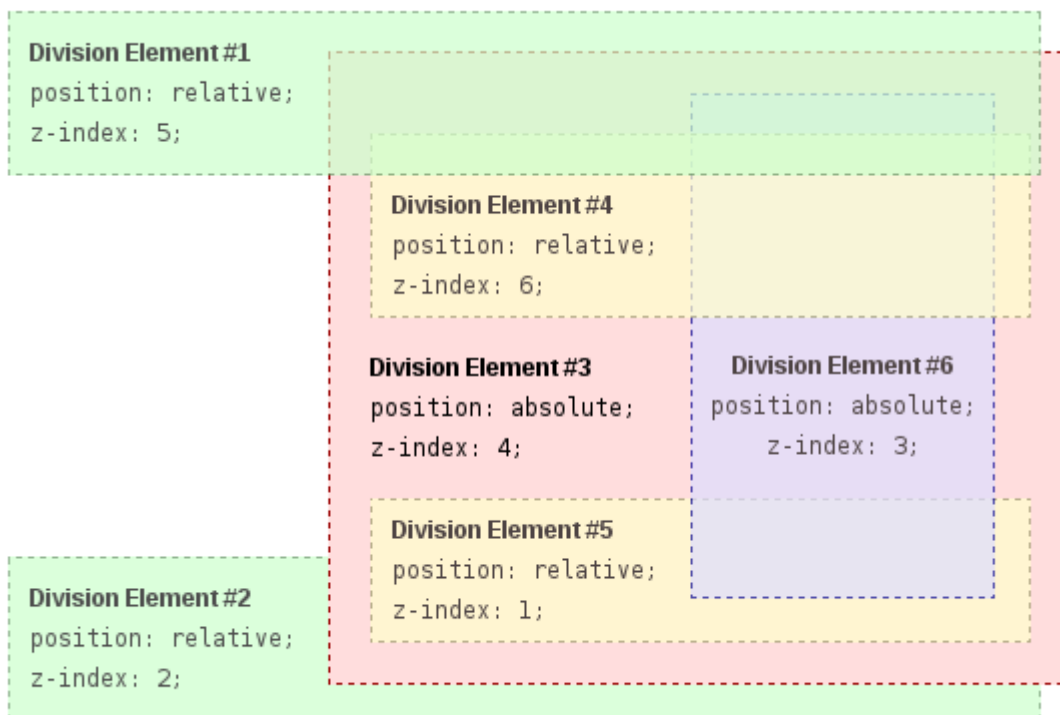
Прочитайте Комментарии онлайн: <https://riptutorial.com/ru/css/topic/1625/комментарии>

глава 16: Контекст стеков

Examples

Контекст стеков

В этом примере каждый позиционируемый элемент создает свой собственный контекст стекирования из-за их позиционирования и значений z-индекса. Иерархия контекстов укладки организована следующим образом:



- корень
 - DIV # 1
 - DIV # 2
 - DIV # 3
 - DIV # 4
 - DIV # 5
 - DIV # 6

Важно отметить, что DIV # 4, DIV # 5 и DIV # 6 являются дочерними элементами DIV # 3, поэтому укладка этих элементов полностью разрешена в DIV # 3. После завершения укладки и рендеринга в DIV # 3 весь элемент DIV # 3 передается для укладки в корневой элемент относительно DIV его дочернего узла.

HTML:

```
<div id="div1">
```

```

<h1>Division Element #1</h1>
<code>position: relative;<br/>
z-index: 5;</code>
</div>
<div id="div2">
  <h1>Division Element #2</h1>
  <code>position: relative;<br/>
  z-index: 2;</code>
</div>
<div id="div3">
  <div id="div4">
    <h1>Division Element #4</h1>
    <code>position: relative;<br/>
    z-index: 6;</code>
  </div>
  <h1>Division Element #3</h1>
  <code>position: absolute;<br/>
  z-index: 4;</code>
  <div id="div5">
    <h1>Division Element #5</h1>
    <code>position: relative;<br/>
    z-index: 1;</code>
  </div>
  <div id="div6">
    <h1>Division Element #6</h1>
    <code>position: absolute;<br/>
    z-index: 3;</code>
  </div>
</div>

```

CSS:

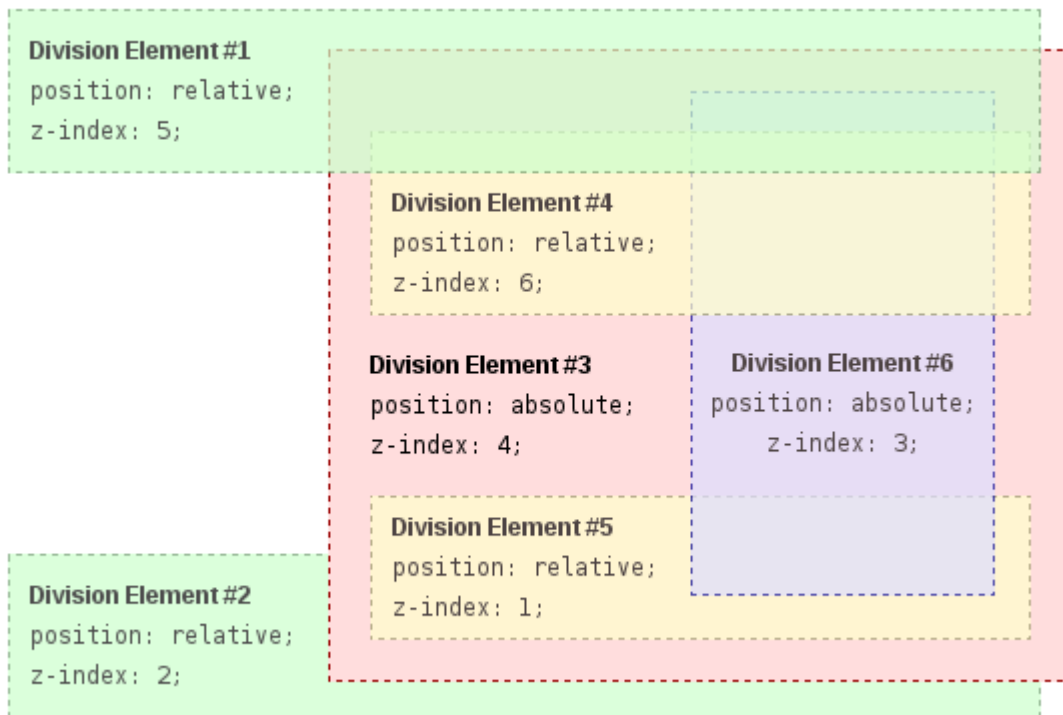
```

* {
  margin: 0;
}
html {
  padding: 20px;
  font: 12px/20px Arial, sans-serif;
}
div {
  opacity: 0.7;
  position: relative;
}
h1 {
  font: inherit;
  font-weight: bold;
}
#div1,
#div2 {
  border: 1px dashed #696;
  padding: 10px;
  background-color: #cfc;
}
#div1 {
  z-index: 5;
  margin-bottom: 190px;
}
#div2 {
  z-index: 2;
}

```

```
#div3 {
  z-index: 4;
  opacity: 1;
  position: absolute;
  top: 40px;
  left: 180px;
  width: 330px;
  border: 1px dashed #900;
  background-color: #fdd;
  padding: 40px 20px 20px;
}
#div4,
#div5 {
  border: 1px dashed #996;
  background-color: #ffc;
}
#div4 {
  z-index: 6;
  margin-bottom: 15px;
  padding: 25px 10px 5px;
}
#div5 {
  z-index: 1;
  margin-top: 15px;
  padding: 5px 10px;
}
#div6 {
  z-index: 3;
  position: absolute;
  top: 20px;
  left: 180px;
  width: 150px;
  height: 125px;
  border: 1px dashed #009;
  padding-top: 125px;
  background-color: #ddf;
  text-align: center;
}
```

Результат:



Источник: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Positioning/Understanding_z_index/The_stacking_context .

Прочитайте Контекст стеков онлайн: <https://riptutorial.com/ru/css/topic/5037/контекст-стеков>

глава 17: Контексты форматирования блоков

замечания

[Контекст форматирования блоков является частью визуального CSS-рендеринга веб-страницы. Это область, в которой происходит компоновка блочных ящиков и в которых поплавки взаимодействуют друг с другом.] [1]

[1]: https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Block_formatting_context MDN

Examples

Использование свойства переполнения со значением, отличным от ВИДИМОГО

```
img{
  float:left;
  width:100px;
  margin:0 10px;
}
.div1{
  background:#f1f1f1;
  /* does not create block formatting context */
}
.div2{
  background:#f1f1f1;
  overflow:hidden;
  /* creates block formatting context */
}
```

```


1 
2 <div class=div1>
3 <p>Lorem ipsum dolor sit amet, cum no paulo mollis pertinacia. Eam in velit graecis, sea mucus insolens ne. Amet doming at has, omnis errem an cum. Eu vim appareat persecuti, ea putant definitionem has, vis ea legendos expetenda. No eros graeci minimum nam, justo augue instructor usu ne. At ludus suscipit disputationi vel.</p>
4
5 <p>Ad case omnis nam, mutat deseruisse persequeris eos ad, in tollit debitis sea. Cu eos munere virtute vituperata. Exerci bonorum sed id, id nec tantas praesent complectitur. Vel cu legendos mediocritatem. Enim liberavisse ei sea.</p>
6 </div>
7
8 
9 <div class=div2>
10 <p>Lorem ipsum dolor sit amet, cum no paulo mollis pertinacia. Eam in velit graecis, sea mucus insolens ne. Amet doming at has, omnis errem an cum. Eu vim appareat persecuti, ea putant definitionem has, vis ea legendos expetenda. No eros graeci minimum nam, justo augue instructor usu ne. At ludus suscipit disputationi vel.</p>
11
12 <p>Ad case omnis nam, mutat deseruisse persequeris eos ad, in tollit debitis sea. Cu eos munere virtute vituperata. Exerci bonorum sed id, id nec tantas praesent complectitur. Vel cu legendos mediocritatem. Enim liberavisse ei sea.</p>
13 </div>

```

```


1 img{
2   float:left;
3   width:100px;
4   margin:0 10px;
5 }
6 .div1{
7   background:#f1f1f1;
8 }
9 .div2{
10  background:#f1f1f1;
11  overflow:hidden;
12  /* creates block formatting c
13 }

```



Lorem ipsum dolor doming at has, omnis expetenda. No eros

Ad case omnis nam, mutat deseruisse p sed id, id nec tantas praesent complecti



Lorem ipsum dolor doming at has, omnis expetenda. No eros

Ad case omnis nam, Exerci bonorum sed sea.

<https://jsfiddle.net/MadalinaTn/qkwwmu6m/2/>

Использование свойства переполнения со значением, отличным от видимого (по умолчанию), создаст новый контекст форматирования блока. Это технически необходимо - если поплавок пересекается с элементом прокрутки, он принудительно перевязывает содержимое.

Этот пример, который показывает, как несколько абзацев будет взаимодействовать с плавающим изображением, похоже на [этот пример](#) , на css-tricks.com.

2 : <https://developer.mozilla.org/en-US/docs/Web/CSS/overflow> MDN

Прочитайте [Контексты форматирования блоков онлайн](#): <https://riptutorial.com/ru/css/topic/5069/контексты-форматирования-блоков>

глава 18: контуры

Синтаксис

- контур: контурная линия с контуром-контуром | начальная | наследовать;
- ширина: средняя | тонкий | толстый | длина | начальная | наследовать;
- контурный стиль: нет | скрытый | пунктирный | пунктирный | твердый | двойной | паз | гребень | вставка | начало | начальная | наследовать;

параметры

параметр	подробности
пунктирный	пунктирная линия
пунктирная	пунктирный контур
твердый	сплошной контур
двойной	двойной контур
паз	Трехмерный рифленый контур, зависит от значения цвета контура
хребет	Трехмерный ребристый контур, зависит от значения контурного цвета
вставка	3D-схема вставки, зависит от значения цвета контура
боковик	3D outset outline, зависит от значения цвета контура
никто	нет набросков
скрытый	скрытый контур

замечания

`outline` теперь описана в [основном пользовательском интерфейсе](#) , уровне 3 модуля CSS (он уже был описан в REC CSS2.1)

Свойство `Outline` определено по умолчанию в браузерах для настраиваемых элементов в `:focus` СОСТОЯНИИ `:focus` .

Его не следует удалять, см. <http://outlinenone.com>, в котором говорится:

Что делает свойство контура?

Он обеспечивает визуальную обратную связь для ссылок, которые имеют «фокус» при навигации по веб-документу с использованием клавиши TAB (или эквивалента). Это особенно полезно для людей, которые не могут использовать мышь или имеют визуальное ухудшение. Если вы удалите контур, вы делаете свой сайт недоступным для этих людей. (...)

Интересные связанные примеры переполнения стека:

- [Как удалить выделение границы в текстовом элементе ввода](#)
- [Как удалить пунктирную схему Firefox на BUTTONS, а также ссылки?](#)

Examples

обзор

Контур - это линия, проходящая вокруг элемента, за пределами границы. В отличие от `border`, контуры не занимают места в коробке. Поэтому добавление контура элемента не влияет на положение элемента или других элементов.

Кроме того, контуры могут быть непрямоугольными в некоторых браузерах. Это может произойти, если `outline` применяется к элементу `span` который содержит текст с различными свойствами `font-size` внутри него. В отличие от границ, контуры *не могут* иметь закругленные углы.

Существенными частями `outline` являются `outline-color`, `outline-style` и `outline-width`.

Определение контура эквивалентно определению границы:

Контур - это линия вокруг элемента. Он отображается по краю элемента. Однако он отличается от пограничной собственности.

```
outline: 1px solid black;
```

план-стиль

Свойство `outline-style` используется для установки стиля контура элемента.

```
p {
  border: 1px solid black;
  outline-color:blue;
  line-height:30px;
}
.p1{
  outline-style: dotted;
}
```



```
.p2{
  outline-style: dashed;
}
.p3{
  outline-style: solid;
}
.p4{
  outline-style: double;
}
.p5{
  outline-style: groove;
}
.p6{
  outline-style: ridge;
}
.p7{
  outline-style: inset;
}
.p8{
  outline-style: outset;
}
```

HTML

```
<p class="p1">A dotted outline</p>
<p class="p2">A dashed outline</p>
<p class="p3">A solid outline</p>
<p class="p4">A double outline</p>
<p class="p5">A groove outline</p>
<p class="p6">A ridge outline</p>
<p class="p7">An inset outline</p>
<p class="p8">An outset outline</p>
```

A dotted outline

A dashed outline

A solid outline

A double outline

A groove outline

A ridge outline

An inset outline

An outset outline

Прочитайте контуры онлайн: <https://riptutorial.com/ru/css/topic/4258/контуры>

глава 19: коробчатого тень

Синтаксис

- box-shadow: none | h-shadow v-shadow blur spread цвет | вставка | initial | inherit;

параметры

параметры	подробности
вставка	по умолчанию тень рассматривается как тень. ключевое слово вставки рисует тень внутри рамки / границы.
Смещение-x	горизонтальное расстояние
Смещение-y	вертикальное расстояние
размытие радиуса	0 по умолчанию. значение не может быть отрицательным. чем больше значение, тем больше становится и становится светлее.
с расширенным радиусом	0 по умолчанию. положительные значения заставят тень расширяться. отрицательные значения уменьшат тень.
цвет	могут быть разных обозначений: ключевое слово цвета, шестнадцатеричное, <code>rgb()</code> , <code>rgba()</code> , <code>hsl()</code> , <code>hsla()</code>

замечания

Поддержка браузера:

- Chrome 10.0
- IE 9.0
- Firefox 4.0 3.5 -moz
- Safari 5.1 3.1 -webkit-
- Opera 10.5

Examples

тени

JSFiddle: <https://jsfiddle.net/UnsungHero97/80qpd7aL/>

HTML

```
<div class="box_shadow"></div>
```

CSS

```
.box_shadow {  
  -webkit-box-shadow: 0px 0px 10px -1px #444444;  
  -moz-box-shadow: 0px 0px 10px -1px #444444;  
  box-shadow: 0px 0px 10px -1px #444444;  
}
```

внутренняя тень

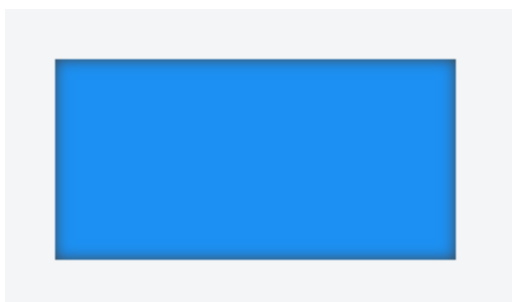
HTML

```
<div class="box_shadow"></div>
```

CSS

```
.box_shadow {  
  background-color: #1C90F3;  
  width: 200px;  
  height: 100px;  
  margin: 50px;  
  -webkit-box-shadow: inset 0px 0px 10px 0px #444444;  
  -moz-box-shadow: inset 0px 0px 10px 0px #444444;  
  box-shadow: inset 0px 0px 10px 0px #444444;  
}
```

Результат:



JSFiddle: <https://jsfiddle.net/UnsungHero97/80qpd7aL/1/>

НИЖНЯЯ ТОЛЬКО ТЕНЬ С ИСПОЛЬЗОВАНИЕМ ПСЕВДОЭЛЕМЕНТА

JSFiddle: <https://jsfiddle.net/UnsungHero97/80qpd7aL/2/>

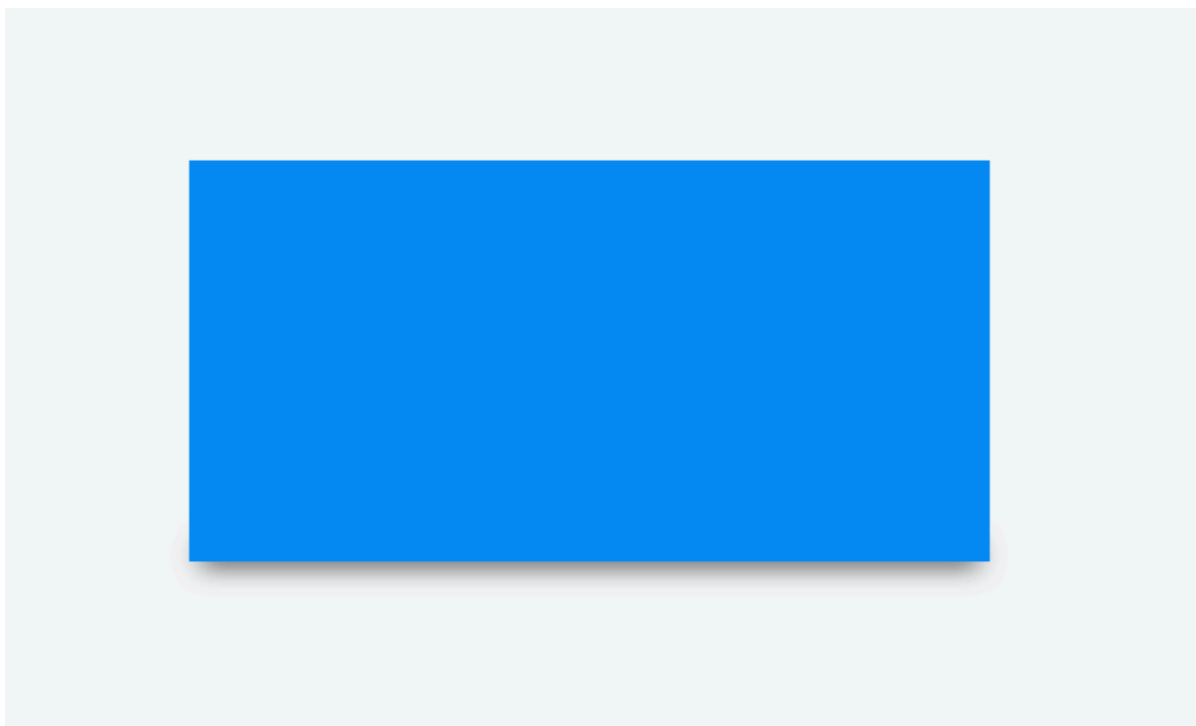
HTML

```
<div class="box_shadow"></div>
```

CSS

```
.box_shadow {
  background-color: #1C90F3;
  width: 200px;
  height: 100px;
  margin: 50px;
}

.box_shadow:after {
  content: "";
  width: 190px;
  height: 1px;
  margin-top: 98px;
  margin-left: 5px;
  display: block;
  position: absolute;
  z-index: -1;
  -webkit-box-shadow: 0px 0px 8px 2px #444444;
  -moz-box-shadow: 0px 0px 8px 2px #444444;
  box-shadow: 0px 0px 8px 2px #444444;
}
```



МНОГОКРАТНЫЕ ТЕНИ

JSFiddle: <https://jsfiddle.net/UnsungHero97/80qpd7aL/5/>

HTML

```
<div class="box_shadow"></div>
```

CSS

```
.box_shadow {  
  width: 100px;  
  height: 100px;  
  margin: 100px;  
  box-shadow:  
    -52px -52px 0px 0px #f65314,  
    52px -52px 0px 0px #7cbb00,  
    -52px 52px 0px 0px #00a1f1,  
    52px 52px 0px 0px #ffbb00;  
}
```



Прочитайте коробчатого тень онлайн: <https://riptutorial.com/ru/css/topic/1746/коробчатого-тень>

глава 20: Макет встроенного блока

Examples

Обоснованная панель навигации

В горизонтально-ориентированной навигационной панели (меню) есть несколько элементов, которые должны быть оправданы. Первый (левый) элемент не имеет левого поля в контейнере, последний (правый) элемент не имеет правого края в контейнере. Расстояние между элементами равно, независимо от ширины отдельного элемента.

HTML

```
<nav>
  <ul>
    <li>abc</li>
    <li>abcdefghijkl</li>
    <li>abcdef</li>
  </ul>
</nav>
```

CSS

```
nav {
  width: 100%;
  line-height: 1.4em;
}
ul {
  list-style: none;
  display: block;
  width: 100%;
  margin: 0;
  padding: 0;
  text-align: justify;
  margin-bottom: -1.4em;
}
ul:after {
  content: "";
  display: inline-block;
  width: 100%;
}
li {
  display: inline-block;
}
```

Заметки

- `nav`, `ul` и `li` были выбраны для их семантического значения «список элементов навигации (меню)». Разумеется, можно использовать и другие теги.
- Псевдоэлемент `:after` псевдоэлемента вызывает дополнительную «линию» в `ul` и, следовательно, дополнительную, пустую высоту этого блока, нажатие другого содержимого вниз. Это решается отрицательным `margin-bottom`, которое должно иметь ту же величину, что и `line-height` (но отрицательная).
- Если страница становится слишком узкой для всех элементов, которые подходят, элементы будут разбиты на новую строку (начиная с правой) и будут оправданы в этой строке. Общая высота меню будет расти по мере необходимости.

Прочитайте Макет встроенного блока онлайн: <https://riptutorial.com/ru/css/topic/3308/макет-встроенного-блока>

глава 21: Маржа

Синтаксис

- `margin: <top & right & bottom & left> ;`
- `margin: <top> , <left & right> , <bottom> ;`
- `margin: <top & bottom> , <left & right> ;`
- `margin: <top> , <right> , <bottom> , <left> ;`
- `margin-top: <top> ;`
- `margin-right: <right> ;`
- `margin-bottom: <bottom> ;`
- `margin-left: <left> ;`

параметры

параметр	подробности
0	установить маржу на none
авто	используется для центрирования, равномерно устанавливая значения с каждой стороны
единиц (например, px)	см. раздел параметров в единицах для списка допустимых единиц
унаследовать	наследовать значение маржи от родительского элемента
начальная	восстановить исходное значение

замечания

Подробнее о «Collapsing Margins»: [здесь](#) .

Examples

Применить маржу на данной стороне

Свойства, специфичные для направления

CSS позволяет указать заданную сторону для применения поля. Четыре свойства, предусмотренные для этой цели:

- margin-left
- margin-right
- margin-top
- margin-bottom

В следующем коде применяется край 30 пикселей в левой части выбранного div.

[Просмотреть результат](#)

HTML

```
<div id="myDiv"></div>
```

CSS

```
#myDiv {  
  margin-left: 30px;  
  height: 40px;  
  width: 40px;  
  background-color: red;  
}
```

параметр	подробности
Левое поле	Направление, в котором применяется маржа.
30px	Ширина поля.

Указание направления использования сокращенной собственности

Свойство стандартного `margin` можно расширить, чтобы указать разные ширины для каждой стороны выбранных элементов. Синтаксис для этого заключается в следующем:

```
margin: <top> <right> <bottom> <left>;
```

В следующем примере применяется маркер нулевой ширины к вершине div, к краю 10px с правой стороны, к краю 50px с левой стороны и к краю 100px с левой стороны. [Просмотреть результат](#)

HTML

```
<div id="myDiv"></div>
```

CSS

```
#myDiv {
  margin: 0 10px 50px 100px;
  height: 40px;
  width: 40px;
  background-color: red;
}
```

Свертывание маржи

Когда два поля касаются друг друга вертикально, они обрушиваются. Когда два поля касаются горизонтально, они не разрушаются.

Пример смежных вертикальных полей:

Рассмотрим следующие стили и разметку:

```
div{
  margin: 10px;
}
```

```
<div>
  some content
</div>
<div>
  some more content
</div>
```

Они будут на 10 пикселей друг от друга, так как вертикальные поля рушится над одним и другим. (Интервал не будет суммой двух полей.)

Пример смежных горизонтальных полей:

Рассмотрим следующие стили и разметку:

```
span{
  margin: 10px;
}
```

```
<span>some</span><span>content</span>
```

Они будут 20px друг от друга, поскольку горизонтальные поля не разваливаются над одним и другим. (Интервал будет суммой двух полей.)

Перекрытие с различными размерами

```
.top{
  margin: 10px;
}
.bottom{
  margin: 15px;
}
```

```
<div class="top">
  some content
</div>
<div class="bottom">
  some more content
</div>
```

Эти элементы будут располагаться на расстоянии 15 пикселей по вертикали. Поля перекрываются настолько, насколько это возможно, но больший запас будет определять расстояние между элементами.

Полученная граница перекрытия

```
.outer-top{
  margin: 10px;
}
.inner-top{
  margin: 15px;
}
.outer-bottom{
  margin: 20px;
}
.inner-bottom{
  margin: 25px;
}
```

```
<div class="outer-top">
  <div class="inner-top">
    some content
  </div>
</div>
<div class="outer-bottom">
  <div class="inner-bottom">
    some more content
  </div>
</div>
```

Каким будет расстояние между двумя текстами? (наведите указатель мыши на ответ)

Интервал будет 25px. Поскольку все четыре поля касаются друг друга, они рухнут, таким образом, используя самый большой запас из четырех.

Теперь, как насчет того, добавим ли мы границы к разметке выше.

```
div{
  border: 1px solid red;
}
```

Каким будет расстояние между двумя текстами? (наведите указатель мыши на ответ)

Интервал будет 59 пикселей! Теперь только поля `.outer-top` и `.outer-bottom` касаются друг друга и являются единственными сложенными полями. Остальные поля разделяются границами. Итак, мы имеем $1\text{px} + 10\text{px} + 1\text{px} +$

15px + 20px + 1px + 25px + 1px. (1px - это границы ...)

Сводные поля между родительскими и дочерними элементами:

HTML:

```
<h1>Title</h1>
<div>
  <p>Paragraph</p>
</div>
```

CSS

```
h1 {
  margin: 0;
  background: #cff;
}
div {
  margin: 50px 0 0 0;
  background: #cfc;
}
p {
  margin: 25px 0 0 0;
  background: #cf9;
}
```

В приведенном выше примере применяется только самый большой запас. Возможно, вы ожидали, что абзац будет расположен 60px из h1 (так как элемент div имеет верхнюю границу поля в 40px, а p имеет верхний край 20px). Этого не происходит, потому что поля сворачиваются вместе, чтобы сформировать один запас.

Горизонтальные элементы центра на странице с использованием поля

Пока элемент является **блоком**, и он имеет **явно заданное значение ширины**, поля могут использоваться для центрирования элементов блока на странице по горизонтали.

Мы добавляем значение ширины, которое меньше ширины окна, и свойство `auto margin` затем распределяет оставшееся пространство влево и вправо:

```
#myDiv {
  width:80%;
  margin:0 auto;
}
```

В приведенном выше примере мы используем объявление сокращенного `margin` чтобы сначала установить 0 в верхние и нижние значения полей (хотя это может быть любое значение), а затем мы используем `auto` чтобы браузер автоматически выделил пространство влево и вправо.

В приведенном выше примере элемент `#myDiv` имеет ширину 80%, что оставляет 20%

оставшегося. Браузер распространяет это значение на оставшиеся стороны так:

$$(100\% - 80\%) / 2 = 10\%$$

Упрощение маржинальной собственности

```
p {
  margin:1px;                /* 1px margin in all directions */
  /*equals to:*/
  margin:1px 1px;
  /*equals to:*/
  margin:1px 1px 1px;
  /*equals to:*/
  margin:1px 1px 1px 1px;
}
```

Другой пример:

```
p{
  margin:10px 15px;         /* 10px margin-top & bottom And 15px margin-right & left*/
  /*equals to:*/
  margin:10px 15px 10px 15px;
  /*equals to:*/
  margin:10px 15px 10px;
  /* margin left will be calculated from the margin right value (=15px) */
}
```

Отрицательные поля

Маржа является одним из нескольких свойств CSS, которые могут быть установлены на отрицательные значения. Это свойство можно использовать для **перекрытия элементов без абсолютного позиционирования** .

```
div{
  display: inline;
}

#over{
  margin-left: -20px;
}

<div>Base div</div>
<div id="over">Overlapping div</div>
```

Пример 1:

Очевидно, что процентное значение маржи по отношению к `margin-left` и `margin-right` будет относиться к его родительскому элементу.

```
.parent {
  width : 500px;
  height: 300px;
}

.child {
  width : 100px;
  height: 100px;
  margin-left: 10%; /* (parentWidth * 10/100) => 50px */
}
```

Но это не тот случай, когда приходит `margin-top` и `margin-bottom`. Оба этих свойства в процентах не относятся к высоте родительского контейнера, а к **ширине** родительского контейнера.

Так,

```
.parent {
  width : 500px;
  height: 300px;
}

.child {
  width : 100px;
  height: 100px;
  margin-left: 10%; /* (parentWidth * 10/100) => 50px */
  margin-top: 20%; /* (parentWidth * 20/100) => 100px */
}
```

Прочитайте Маржа онлайн: <https://riptutorial.com/ru/css/topic/305/маржа>

глава 22: Модель коробки

Синтаксис

- `box-sizing`: параметр ;

параметры

параметр	подробность
<code>content-box</code>	Ширина и высота элемента включают только область содержимого.
<code>padding-box</code>	Ширина и высота элемента включают содержимое и дополнение.
<code>border-box</code>	Ширина и высота элемента включают содержимое, дополнение и границу.
<code>initial</code>	Устанавливает модель окна в состояние по умолчанию.
<code>inherit</code>	Наследует модель поля родительского элемента.

замечания

О подкладке

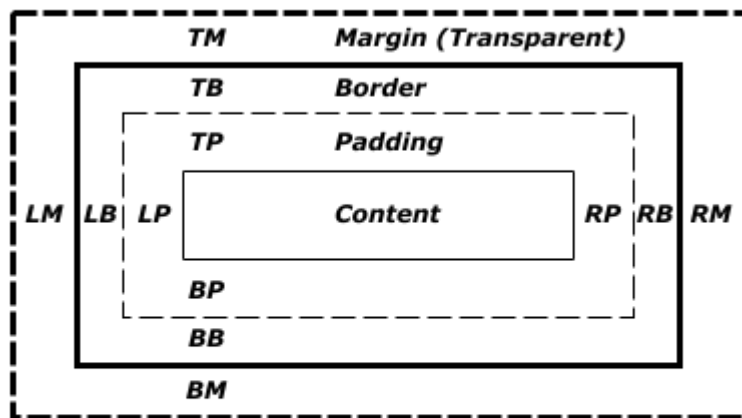
Это значение было реализовано только **Firefox** и, следовательно, не должно использоваться. Он был удален в Firefox версии 50.0.

Examples

Что такое модель коробки?

Края

Браузер создает прямоугольник для каждого элемента в документе HTML. Модель Box описывает, как добавление, граница и маржа добавляются в контент для создания этого прямоугольника.



- Margin edge
- Border edge
- - - Padding edge
- Content edge

Диаграмма из [CSS2.2 Рабочий проект](#)

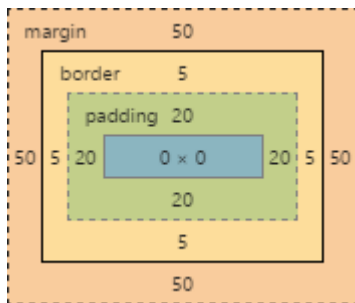
Периметр каждой из четырех областей называется *краем* . Каждое ребро определяет *поле*.

- Самый внутренний прямоугольник - это **поле содержимого** . Ширина и высота этого зависит от отображаемого содержимого элемента (текста, изображений и любых дочерних элементов, которые могут иметь).
- Далее находится поле **прокладки** , определяемое свойством `padding` . Если ширина `padding` не определена, край заполнения равен краю содержимого.
- Затем у нас есть поле **границы** , как определено свойством `border` . Если ширина `border` не определена, край границы равен краю заполнения.
- Самый внешний прямоугольник - **поле поля** , определяемое свойством `margin` . Если ширина `margin` не задана, край края равен граничному краю.

пример

```
div {
  border: 5px solid red;
  margin: 50px;
  padding: 20px;
}
```

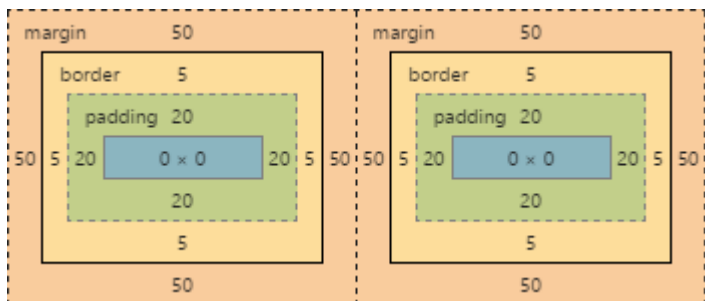
Этот CSS стилирует все элементы `div` чтобы иметь верхнюю, правую, нижнюю и левую границы шириной 5 `5px` ; верхний, правый, нижний и левый поля 50`px` ; и верхнее, правое, нижнее и левое заполнение 20 `20px` . Игнорируя содержимое, наша сгенерированная коробка будет выглядеть так:



Снимок экрана панели стилей элементов Google Chrome

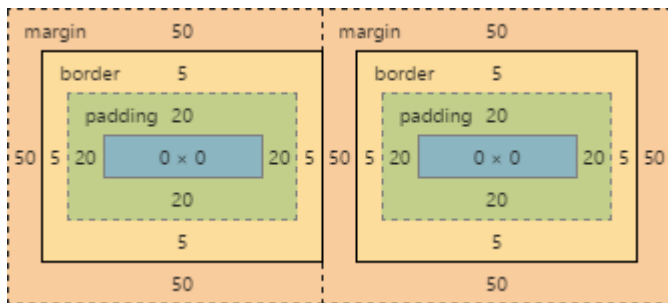
- Поскольку содержимого нет, область содержимого (синий квадрат посередине) не имеет высоты или ширины (0px на 0px).
- Поле заполнения по умолчанию имеет тот же размер, что и поле содержимого, плюс ширина 20 пикселей на всех четырех ребрах, которые мы определяем выше, с атрибутом `padding` (40 пикселей на 40 пикселей).
- Граница границы того же размера, что и поле заполнения, плюс ширина 5 пикселей, которую мы определяем выше, с свойством `border` (50 пикселей на 50 пикселей).
- Наконец, поле поля имеет тот же размер, что и поле рамки, плюс ширина 50 пикселей, которую мы определяем выше, с свойством `margin` (предоставляя нашему элементу общий размер 150 пикселей на 150 пикселей).

Теперь давайте придадим нашему элементу брата с тем же стилем. Браузер смотрит на `Box Model` обоих элементов, чтобы определить, где по отношению к содержимому предыдущего элемента должен быть позиционирован новый элемент:



Содержимое каждого элемента разделяется пробелом 150 пикселей, но коробки с двумя элементами касаются друг друга.

Если мы затем изменим наш первый элемент на отсутствие правого края, край правого края будет находиться в том же положении, что и правый край границы, и теперь наши два элемента будут выглядеть так:



коробчатого проклейки

Модель окна по умолчанию (`content-box`) может быть нелогичной, так как `width / height` для элемента не будет представлять его фактическую ширину или высоту на экране , как только вы начинаете добавлять `padding` и `border` стилей к элементу.

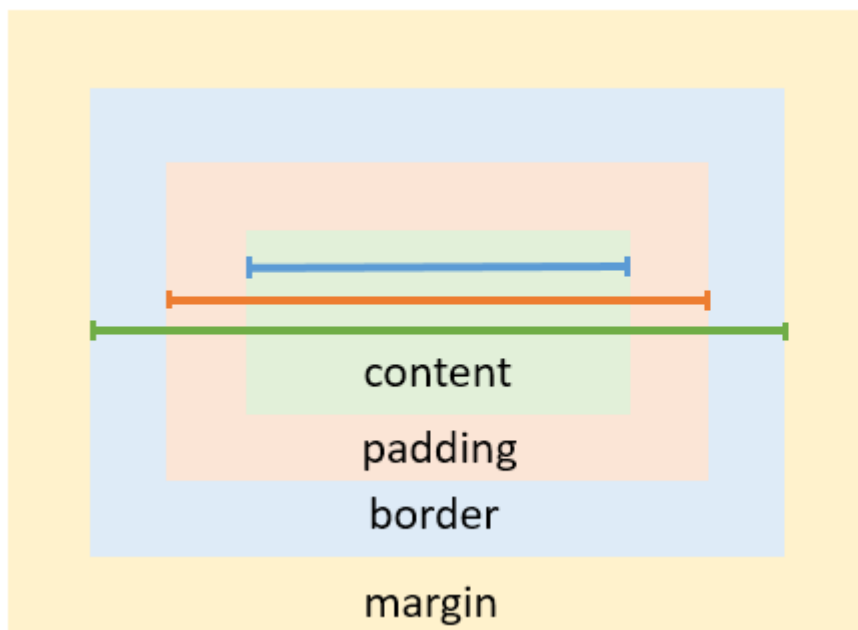
Следующий пример демонстрирует эту потенциальную проблему с `content-box` :

```
textarea {
  width: 100%;
  padding: 3px;
  box-sizing: content-box; /* default value */
}
```

Так как добавление будет добавлено к ширине текстового поля, результирующий элемент будет текстовым полем, который превышает 100%.

К счастью, CSS позволяет нам изменить модель окна с помощью свойства `box-sizing` для элемента. Существует три разных значения доступного свойства:

- `content-box` : общая модель окна - ширина и высота включают только контент, а не дополнение или границу
- `padding-box` : ширина и высота включают в себя содержимое и дополнение, но не границу
- `border-box` : ширина и высота включают в себя содержимое, дополнение и границу



width

■ **content-box:** content

■ **padding-box:** content + padding

■ **border-box:** content + padding + border

Чтобы решить проблему с `textarea` выше, вы можете просто изменить свойство `box-sizing` в `padding-box` или `border-box`. `border-box` наиболее часто используется.

```
textarea {
  width: 100%;
  padding: 3px;
  box-sizing: border-box;
}
```

Чтобы применить определенную модель окна к каждому элементу страницы, используйте следующий фрагмент:

```
html {
  box-sizing: border-box;
}

*, *:before, *:after {
  box-sizing: inherit;
}
```

В этом `box-sizing: border-box;` кодирования `box-sizing: border-box;` не применяется непосредственно к `*`, поэтому вы можете легко переписать это свойство на отдельные элементы.

Прочитайте Модель коробки онлайн: <https://riptutorial.com/ru/css/topic/646/модель-коробки>

глава 23: набивка

Синтаксис

- padding: *length* | initial | inherit | unset;
- padding-top: *length* | initial | inherit | unset;
- padding-right: *length* | initial | inherit | unset;
- padding-bottom: *length* | initial | inherit | unset;
- padding-left: *length* | initial | inherit | unset;

замечания

Свойство padding задает заполняющее пространство со всех сторон элемента. Область заполнения - это пространство между содержимым элемента и его границей. **Отрицательные значения не допускаются** .

1 : <https://developer.mozilla.org/en/docs/Web/CSS/padding> MDN

Также см. Этот [вопрос](#) : «Почему CSS не поддерживает отрицательное дополнение?» и его ответы.

Также, пожалуйста, рассмотрите [модель Box](#) при использовании прокладки. В зависимости от значения размера коробки, добавление элемента может либо добавить к ранее определенной высоте / ширине элемента, либо нет.

Связанные свойства:

[поле](#)

Заполнение встроенных элементов будет применяться только к левому и правому элементам, а не к верхней и нижней части из-за присущих свойств отображения встроенных элементов.

Examples

Прокладка на заданной стороне

Свойство padding задает заполняющее пространство со всех сторон элемента. Область заполнения - это пространство между содержимым элемента и его границей. Отрицательные значения не допускаются.

Вы можете указать сторону отдельно:

- padding-top

- padding-right
- padding-bottom
- padding-left

Следующий код добавит отступы 5px в начало div:

```
<style>
.myClass {
  padding-top: 5px;
}
</style>

<div class="myClass"></div>
```

Сокращение

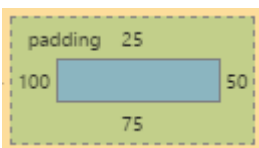
Свойство padding задает заполняющее пространство со всех сторон элемента. Область заполнения - это пространство между содержимым элемента и его границей.

Отрицательные значения не допускаются.

Чтобы сохранить добавление отступов на каждой стороне отдельно (с помощью padding-top, padding-left т. Д.), Вы можете записать его как сокращенное, как показано ниже:

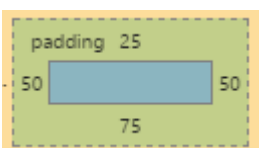
Четыре значения :

```
<style>
  .myDiv {
    padding: 25px 50px 75px 100px; /* top right bottom left; */
  }
</style>
<div class="myDiv"></div>
```



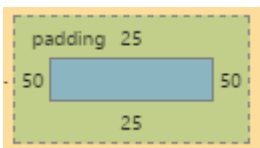
Три значения :

```
<style>
  .myDiv {
    padding: 25px 50px 75px; /* top left/right bottom */
  }
</style>
<div class="myDiv"></div>
```



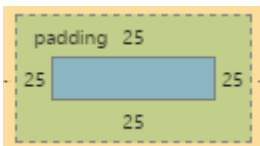
Два значения :

```
<style>
  .myDiv {
    padding: 25px 50px; /* top/bottom left/right */
  }
</style>
<div class="myDiv"></div>
```



Одно значение :

```
<style>
  .myDiv {
    padding: 25px; /* top/right/bottom/left */
  }
</style>
<div class="myDiv"></div>
```



Прочитайте набивка онлайн: <https://riptutorial.com/ru/css/topic/1255/набивка>

глава 24: наследование

Синтаксис

- *свойство*: inherit;

Examples

Автоматическое наследование

Наследование основополагающего механизма CSS, с помощью которого вычисляемые значения некоторых свойств элемента применяются к его «детям». Это особенно полезно, если вы хотите установить глобальный стиль для своих элементов, а не устанавливать указанные свойства для каждого элемента вашей разметки.

Обычными свойствами, которые автоматически наследуются, являются: `font` , `color` , `text-align` , `line-height` .

Предположим, что следующая таблица стилей:

```
#myContainer {
  color: red;
  padding: 5px;
}
```

Это применит `color: red` не только к элементу `<div>` но также к элементам `<h3>` и `<p>` .

Однако из-за характера `padding` его значение **не** будет унаследовано от этих элементов.

```
<div id="myContainer">
  <h3>Some header</h3>
  <p>Some paragraph</p>
</div>
```

Принудительное наследование

Некоторые свойства автоматически не унаследованы от элемента до его «детей». Это связано с тем, что эти свойства обычно являются уникальными для элемента (или выбора элементов), к которому применяется свойство. Обычными такими свойствами являются `margin` , `padding` , `background` , `display` и т. Д.

Однако в любом случае желательно наследование. Чтобы достичь этого, мы можем применить значение `inherit` к свойству, которое должно быть унаследовано. `inherit` значение может быть applied для *любого* свойства CSS и *любой* HTML - элемент.

Предположим, что следующая таблица стилей:

```
#myContainer {
  color: red;
  padding: 5px;
}
#myContainer p {
  padding: inherit;
}
```

Это применит `color: red` для обоих элементов `<h3>` и `<p>` из-за наследования свойства `color`. Тем не менее, элемент `<p>` также наследует значение `padding` от его родителя, потому что это было указано.

```
<div id="myContainer">
  <h3>Some header</h3>
  <p>Some paragraph</p>
</div>
```

Прочитайте наследование онлайн: <https://riptutorial.com/ru/css/topic/3586/наследование>

глава 25: Несколько столбцов

Вступление

CSS позволяет определить, что содержимое элемента переносится на несколько столбцов с пробелами и правилами между ними.

замечания

[Модуль многоуровневой компоновки CSS Уровень 1](#) с 12 апреля 2011 года является Рекомендацией кандидата W3C. С тех пор [было сделано](#) несколько [небольших изменений](#) . Считается, что он находится на [стадии «Стабильный»](#) .

По состоянию на 3 июля 2017 года браузеры Microsoft Internet Explorer 10 и 11 и Edge поддерживают только более старую версию спецификации с использованием префикса поставщика.

Examples

Основной пример

Рассмотрим следующую разметку HTML:

```
<section>
  <p>Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor
  invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et
  justo duo dolores et ea rebum.</p>
  <p> Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem
  ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut
  labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo
  dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor
  sit amet.</p>
  <p>Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor
  invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et
  justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem
  ipsum dolor sit amet.</p>
</section>
```

При использовании следующего CSS содержимое делится на три столбца, разделенных серым столбцом с двумя пикселями.

```
section {
  columns: 3;
  column-gap: 40px;
  column-rule: 2px solid gray;
}
```

Посмотрите [живой пример этого на JSFiddle](#) .

Создание нескольких столбцов

```
<div class="content">
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh
eismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim
ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl
ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in
hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu
feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui
blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla
facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil
imperdiet doming id quod mazim placerat facer possim assum.
</div>
```

Css

```
.content {
-webkit-column-count: 3; /* Chrome, Safari, Opera */
-moz-column-count: 3; /* Firefox */
column-count: 3;
}
```

Прочитайте [Несколько столбцов онлайн: https://riptutorial.com/ru/css/topic/10688/несколько-столбцов](https://riptutorial.com/ru/css/topic/10688/несколько-столбцов)

глава 26: Нормализация стилей браузера

Вступление

Каждый браузер имеет набор стилей CSS по умолчанию, который он использует для рендеринга элементов. Эти стили по умолчанию могут быть несовместимыми в разных браузерах, потому что: спецификации языка неясны, поэтому базовые стили подходят для интерпретации, браузеры могут не следовать указанным спецификациям, или браузеры могут не иметь стилей по умолчанию для новых HTML-элементов. В результате люди могут захотеть нормализовать стили по умолчанию для максимально возможного количества браузеров.

замечания

Сброс Мейера, хотя и эффективен, делает те же самые изменения почти для каждого обычно используемого элемента. Это приводит к тому, что окна инспектора веб-браузера загрязняют окна с одинаковыми прикладными стилями снова и снова и создают больше работы для браузера (больше правил применяются к большему количеству элементов). С другой стороны, метод Normalize намного более сфокусирован и менее широко используется. Это упрощает работу со стороны браузера и приводит к меньшему количеству помех в инструментах проверки браузера.

Examples

normalize.css

Браузеры имеют набор стилей CSS по умолчанию, которые они используют для рендеринга элементов. Некоторые из этих стилей можно даже настроить, используя настройки браузера, чтобы, например, изменять определения шрифта и размера шрифта по умолчанию. В стилях содержится определение того, какие элементы должны быть блочными или встроенными, между прочим.

Поскольку эти стили по умолчанию дают некоторую свободу действий по языковым спецификациям и потому, что браузеры могут не следовать спецификациям, они могут отличаться от браузера к браузеру.

Это где [normalize.css](#) вступает в игру. Он отменяет наиболее распространенные несоответствия и исправляет известные ошибки.

Что оно делает

- Сохраняет полезные значения по умолчанию, в отличие от многих сбросов CSS.
- Нормализует стили для широкого круга элементов.
- Исправляет ошибки и общие непоследовательности браузера.
- Улучшает удобство использования с небольшими изменениями.
- Объясняет, какой код использует подробные комментарии.

Таким образом, включив `normalize.css` в ваш проект, ваш дизайн будет выглядеть более похожим и последовательным в разных браузерах.

Разница в `reset.css`

Возможно, вы слышали об `reset.css`. В чем разница между ними?

В то время как `normalize.css` обеспечивает согласованность, устанавливая разные свойства для унифицированных значений по умолчанию, `reset.css` достигает согласованности, **удаляя** все основные стили, которые может применяться браузером. Хотя сначала это может показаться хорошей идеей, это на самом деле означает, что вы должны сами писать **все** правила, что противоречит тому, чтобы иметь солидный стандарт.

Подходы и примеры

Сброс CSS использует отдельные подходы к умолчанию браузера. Сброс CSS Эрика Мейера уже давно. Его подход сводит на нет многие элементы браузера, которые, как известно, вызывают проблемы сразу же. Ниже приведена его версия (v2.0 | 20110126) Сброс CSS.

```
html, body, div, span, applet, object, iframe,
h1, h2, h3, h4, h5, h6, p, blockquote, pre,
a, abbr, acronym, address, big, cite, code,
del, dfn, em, img, ins, kbd, q, s, samp,
small, strike, strong, sub, sup, tt, var,
b, u, i, center,
dl, dt, dd, ol, ul, li,
fieldset, form, label, legend,
table, caption, tbody, tfoot, thead, tr, th, td,
article, aside, canvas, details, embed,
figure, figcaption, footer, header, hgroup,
menu, nav, output, ruby, section, summary,
time, mark, audio, video {
    margin: 0;
    padding: 0;
    border: 0;
    font-size: 100%;
    font: inherit;
    vertical-align: baseline;
}
```

Сброс CSS Эрика Мейера

Нормализовать CSS на другом и разбираться со многими из них отдельно. Ниже приведен пример из версии (v4.2.0) кода.

```
/**
 * 1. Change the default font family in all browsers (opinionated).
 * 2. Correct the line height in all browsers.
 * 3. Prevent adjustments of font size after orientation changes in IE and iOS.
 */

/* Document
   ===== */

html {
  font-family: sans-serif; /* 1 */
  line-height: 1.15; /* 2 */
  -ms-text-size-adjust: 100%; /* 3 */
  -webkit-text-size-adjust: 100%; /* 3 */
}

/* Sections
   ===== */

/**
 * Remove the margin in all browsers (opinionated).
 */

body {
  margin: 0;
}

/**
 * Add the correct display in IE 9-.
 */

article,
aside,
footer,
header,
nav,
section {
  display: block;
}

/**
 * Correct the font size and margin on `h1` elements within `section` and
 * `article` contexts in Chrome, Firefox, and Safari.
 */

h1 {
  font-size: 2em;
  margin: 0.67em 0;
}
```

Нормализовать CSS

Прочитайте Нормализация стилей браузера онлайн: <https://riptutorial.com/ru/css/topic/1211/нормализация-стилей-браузера>

глава 27: Обрезка и маскировка

Синтаксис

- **вырезка**
- клип-путь: `<clip-source> | [<basic-shape> || <clip-geometry-box>] | никто`
- **маскировка**
- `mask-image: [none | <маска-ссылка>] #`
- маска-режим: `[<маска-режим>] #`
- `mask-repeat: [<стиль повтора>] #`
- `mask-position: [<position>] #`
- `mask-clip: [<geometry-box> | без обрезки] #`
- `mask-origin: [<geometry-box>] #`
- `mask-size: [<bg-size>] #`
- `mask-composite: [<compositing-operator>] #`
- `mask: [<mask-reference> <masking-mode>? || <position> [/ <bg-size>]? || <повторный стиль> || <геометрия> || [<geometry-box> | no-clip] || <compositing-operator>] #`

параметры

параметр	подробности
Клип-источник	URL-адрес, который может указывать на встроенный элемент SVG (или) элемент SVG во внешнем файле, который содержит определение пути клипа.
Основная форма,	Относится к одному из <code>inset()</code> , <code>circle()</code> , <code>ellipse()</code> или <code>polygon()</code> . Используя одну из этих функций, определяется путь обрезки. Эти функции формы работают точно так же, как и в фигурах для плавающих
клип геометрии коробки	Это может иметь один среди <code>content-box</code> , <code>padding-box</code> , <code>border-box</code> , <code>margin-box</code> , <code>fill-box</code> , <code>stroke-box</code> , <code>view-box</code> как значения. Когда это предусмотрено без каких-либо значений для <code><basic-shape></code> , края соответствующего окна используются как путь для отсечения. При использовании с <code><основной формой></code> , это действует в качестве опорного поля для формы.
Маска-справочник	Это может быть <code>none</code> ни образ или ссылочный URL-адрес для источника изображения маски.
повторить	Это указывает, как следует повторять или чередовать маску в осях X

параметр	подробности
стиль	и Y. Поддерживаемые значения: <code>repeat-x</code> , <code>repeat-y</code> , <code>repeat</code> , <code>space</code> , <code>round</code> , <code>no-repeat</code> .
Маска режима	Может быть <code>alpha</code> или <code>luminance</code> или <code>auto</code> и указывает, следует ли рассматривать маску как альфа-маску или маску яркости. Если значение не указано, а ссылка на маску является прямым изображением, то она будет считаться альфа-маской (или), если ссылка на маску является URL-адресом, тогда она будет считаться маской яркости.
позиция	Это определяет положение каждого слоя маски и аналогично поведению свойства <code>background-position</code> . Значение может быть представлено в синтаксисе 1 синтаксиса (например, <code>top</code> , <code>10%</code>) или в синтаксисе значения 2 (например, <code>top right</code> , <code>50%</code> <code>50%</code>).
геометрии коробки	Это указывает поле, в которое должна быть обрезана маска (область маскирования масок), или поле, которое должно использоваться в качестве ссылки для источника маски (область расположения маски) в зависимости от свойства. Список возможных значений: <code>content-box</code> , <code>padding-box</code> , <code>border-box</code> , <code>margin-box</code> , « <code>fill-box</code> , « <code>stroke-box</code> , « <code>view-box</code> . Подробное объяснение того, как работает каждое из этих значений, доступно в спецификации W3C .
BG-размер	Это представляет собой размер каждого слоя маски и имеет тот же синтаксис, что и <code>background-size</code> . Значение может быть длиной или процентом или автоматически, или содержать или содержать. Длина, процент и авто могут быть предоставлены как одно значение или как одно для каждой оси.
компози́нга-оператор	Это может быть любой из <code>add</code> , <code>subtract</code> , <code>exclude</code> , <code>multiply</code> <code>per layer</code> и определяет тип операции компоновки, который должен использоваться для этого слоя, с теми, что ниже него. Подробное объяснение каждого значения доступно в спецификациях W3C .

замечания

CSS Clipping and Masking - это очень новые концепции, поэтому поддержка браузером этих свойств довольно низкая.

Маски:

Как и в момент написания (июль '16), Chrome, Safari и Opera поддерживают эти свойства с `-webkit-` префикса `-webkit-` .

Firefox не требует префиксов, но он поддерживает маски только при использовании с элементами `mask SVG`. Для встроенных элементов `mask SVG` синтаксисом является `mask: url(#msk)` тогда как для использования элементов `mask` во внешнем SVG-файле синтаксис является `mask: url('yourfilepath/yourfilename.svg#msk')` . `#msk` в обоих случаях относится к `id` элемента `mask` , на который ссылается. Как указано в [этом ответе](#) , в настоящее время Firefox не поддерживает какой-либо параметр, отличный от `mask-reference` в свойстве `mask` .

Internet Explorer (и Edge) пока не предлагает поддержку для этого свойства.

Свойство `mask-mode` в настоящее время не поддерживается ни одним браузером с префиксами **или без них** .

Clip-путь:

Как и в момент написания (Jul '16), Chrome, Safari и Opera поддерживают `clip-path` когда путь создается с использованием базовых форм (например, `circle` , `polygon`) или синтаксиса `url(#clipper)` с встроенным SVG. Они не поддерживают отсечение на основе форм, которые являются частью внешних SVG-файлов. Кроме того, они требуют `-webkit` префикса `-webkit` .

Firefox поддерживает только синтаксис `url()` для `clip-path` тогда как Internet Explorer (и Edge) не предлагает поддержки.

Examples

Обрезка (многоугольник)

CSS:

```
div{
  width:200px;
  height:200px;
  background:teal;
  clip-path: polygon(0 0, 0 100%, 100% 50%); /* refer remarks before usage */
}
```

HTML:

```
<div></div>
```


В приведенном выше примере **полигональный** путь отсечения используется для закрепления квадратного (200 x 200) элементов в форме треугольника. Форма вывода является треугольником, потому что путь начинается с (то есть, первые координаты находятся на) 0 0 - это верхний левый угол поля, затем идет до 0 100% - это нижний левый угол поля а затем, наконец, до 100% 50% что не что иное, как правая-средняя точка коробки. Эти пути закрываются автоматически (то есть, отправной точкой будет конечная точка), и поэтому конечная фигура имеет форму треугольника.

Это также можно использовать на элементе с изображением или градиентом в качестве фона.

Пример просмотра

Выход:



Обрезание (круг)

CSS:

```
div{
  width: 200px;
  height: 200px;
  background: teal;
  clip-path: circle(30% at 50% 50%); /* refer remarks before usage */
}
```

HTML

```
<div></div>
```

В этом примере показано, как скопировать div в круг. Элемент обрезается в круг, радиус которого 30% на основе размеров опорной коробки с его центральной точки в центре опорной коробки. Здесь, так как не предоставляется <clip-geometry-box> (другими словами,

ссылочный блок), в качестве ссылочного поля будет использоваться `border-box` элемент.

Форма круга должна иметь радиус и центр с координатами (x, y) :

```
circle(radius at x y)
```

Пример просмотра

Выход:

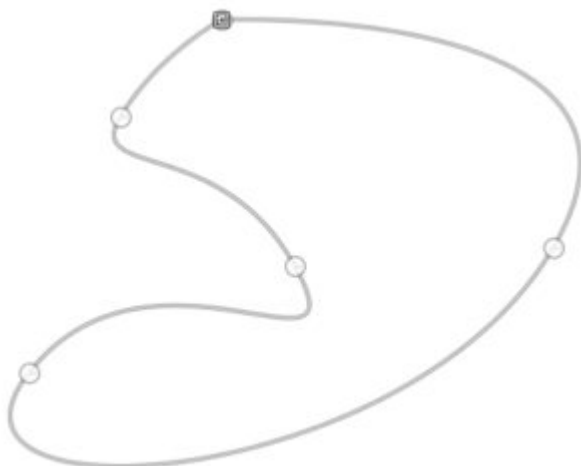


Обрезка и маскировка: обзор и различие

С помощью **Clipping** and **Masking** вы можете сделать некоторые части элементов прозрачными или непрозрачными. Оба могут быть применены к любому элементу HTML.

вырезка

Клипы являются векторными дорожками. Вне этого пути элемент будет прозрачным, внутри непрозрачным. Поэтому вы можете определить свойство `clip-path` для элементов. Каждый графический элемент, который также существует в SVG, вы можете использовать здесь как функцию для определения пути. Примерами являются `circle()` , `polygon()` или `ellipse()` .



пример

```
clip-path: circle(100px at center);
```

Элемент будет виден только внутри этого круга, который расположен в центре элемента и имеет радиус 100 пикселей.

маскировка

Маски похожи на Clips, но вместо определения пути вы определяете маску, какие слои над элементом. Вы можете представить эту маску как изображение, состоящее в основном из двух цветов: черно-белого.

Маска яркости : черный означает, что область непрозрачна, а белый - прозрачный, но есть также полупрозрачная серая область, поэтому вы можете делать плавные переходы.

Альфа-маска : только на прозрачных участках маски элемент будет непрозрачным.



Это изображение, например, можно использовать в качестве маски яркости, чтобы сделать элемент очень плавным переходом справа налево и от непрозрачного до прозрачного.

Свойство `mask` позволяет указать тип маски и изображение, которое будет использоваться как слой.

пример

```
mask: url(masks.svg#rectangle) luminance;
```

Элемент, называемый `rectangle` определенным в `masks.svg` будет использоваться в качестве **маски яркости** для элемента.

Простая маска, которая уменьшает изображение от сплошного до прозрачного

CSS

```
div {  
  height: 200px;  
  width: 200px;  
  background: url(http://lorempixel.com/200/200/nature/1);  
  mask-image: linear-gradient(to right, white, transparent);  
}
```

HTML

```
<div></div>
```

В приведенном выше примере есть элемент с изображением в качестве фона. Маска, применяемая к изображению (с использованием CSS), выглядит так, как будто она исчезает слева направо.

Маскирование достигается с помощью `linear-gradient` который идет от белого (слева) до прозрачного (справа) в качестве маски. Поскольку это альфа-маска, изображение становится прозрачным, когда маска прозрачна.

Выход без маски:



Выход с маской:



Примечание. Как упоминалось в примечаниях, приведенный выше пример будет работать в Chrome, Safari и Opera только при использовании с префиксом `-webkit`. Этот пример (с `linear-gradient` как изображение маски) еще не поддерживается в Firefox.

Использование масок для вырезания отверстия в середине изображения

CSS

```
div {
  width: 200px;
  height: 200px;
  background: url(http://lorempixel.com/200/200/abstract/6);
  mask-image: radial-gradient(circle farthest-side at center, transparent 49%, white 50%); /*
  check remarks before using */
}
```

HTML

В приведенном выше примере прозрачный круг создается в центре с использованием `radial-gradient` и затем он используется в качестве маски для создания эффекта вырезания круга из центра изображения.

Изображение без маски:



Изображение с маской:



Использование масок для создания изображений с неправильной формой

CSS

```
div { /* check remarks before usage */
  height: 200px;
  width: 400px;
  background-image: url(http://lorempixel.com/400/200/nature/4);
  mask-image: linear-gradient(to top right, transparent 49.5%, white 50.5%), linear-
  gradient(to top left, transparent 49.5%, white 50.5%), linear-gradient(white, white);
  mask-size: 75% 25%, 25% 25%, 100% 75%;
  mask-position: bottom left, bottom right, top left;
  mask-repeat: no-repeat;
}
```

HTML

```
<div></div>
```

В приведенном выше примере три изображения с `linear-gradient` (которые при размещении в их соответствующих положениях будут покрывать 100% x 100% от размера контейнера) используются в качестве масок для создания прозрачного треугольного реза в нижней части изображения.

Изображение без маски:



Изображение с маской:



Прочитайте **Обрезка и маскировка онлайн:** <https://riptutorial.com/ru/css/topic/3721/обрезка-и-маскировка>

глава 28: Объектная модель CSS (CSSOM)

замечания

Объектная модель CSS (CSSOM) является отдельной спецификацией.

Текущий проект можно найти здесь: <https://www.w3.org/TR/cssom-1/>

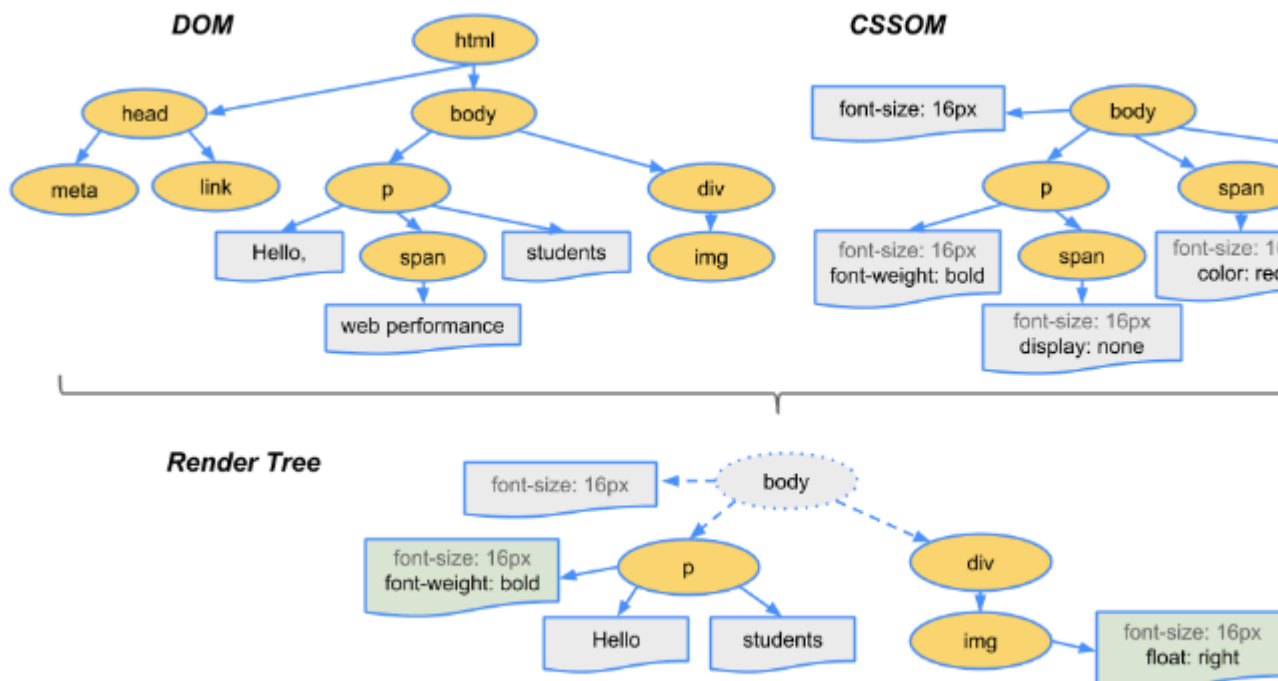
Examples

Вступление

Браузер идентифицирует токены из таблицы стилей и скрывает их в узлах, которые связаны с древовидной структурой. Вся карта всех узлов со связанными стилями страницы будет объектной моделью CSS.

Чтобы отобразить веб-страницу, веб-браузер выполняет следующие шаги.

1. Веб-браузер проверяет ваш HTML и создает DOM (Document Object Model).
2. Веб-браузер проверяет ваш CSS и создает CSSOM (объектная модель CSS).
3. Веб-браузер объединяет DOM и CSSOM для создания дерева рендеринга. Веб-браузер отображает вашу веб-страницу.



Добавление правила background-image через CSSOM

Чтобы добавить правило `background-image` через CSSOM, сначала получите ссылку на правила первой таблицы стилей:

```
var stylesheet = document.styleSheets[0].cssRules;
```

Затем получите ссылку на конец таблицы стилей:

```
var end = stylesheet.length - 1;
```

Наконец, вставьте правило фонового изображения для элемента `body` в конце таблицы стилей:

```
stylesheet.insertRule("body { background-image: url('http://cdn.sstatic.net/Sites/stackoverflow/img/favicon.ico'); }", end);
```

Прочитайте [Объектная модель CSS \(CSSOM\) онлайн: https://riptutorial.com/ru/css/topic/4961/объектная-модель-css--cssom-](https://riptutorial.com/ru/css/topic/4961/объектная-модель-css--cssom-)

глава 29: Одиночные элементы

Examples

Площадь

Чтобы создать квадрат, определите элемент с шириной и высотой. В приведенном ниже примере у нас есть элемент с `width` и `height` по 100 пикселей каждый.



```
<div class="square"></div>
```

```
.square {  
  width: 100px;  
  height: 100px;  
  background: rgb(246, 156, 85);  
}
```

треугольники

Чтобы создать треугольник CSS, определите элемент с шириной и высотой 0 пикселей. Форма треугольника будет сформирована с использованием свойств границы. Для элемента с высотой и шириной 0 четыре границы (сверху, справа, внизу, слева) образуют треугольник. Вот элемент с 0 высотой / шириной и 4 различными цветными границами.



Установив некоторые границы в прозрачные, а другие - на цвет, мы можем создавать различные треугольники. Например, в треугольнике `Up` мы устанавливаем нижнюю границу желаемого цвета, а затем устанавливаем левую и правую границы прозрачными. Вот

изображение с левой и правой границами, слегка затененными, чтобы показать, как формируется треугольник.



Размеры треугольника можно изменить, изменив ширину границы - более высокую, короче, однобокую и т. Д. В приведенных ниже примерах показан треугольник размером 50 × 50 пикселей.

Треугольник - наведение вверх



```
<div class="triangle-up"></div>
```

```
.triangle-up {  
  width: 0;  
  height: 0;  
  border-left: 25px solid transparent;  
  border-right: 25px solid transparent;  
  border-bottom: 50px solid rgb(246, 156, 85);  
}
```

Треугольник - указатель вниз



```
<div class="triangle-down"></div>
```

```
.triangle-down {  
  width: 0;  
  height: 0;  
  border-left: 25px solid transparent;  
  border-right: 25px solid transparent;  
  border-top: 50px solid rgb(246, 156, 85);  
}
```

Треугольник - правый указатель



```
<div class="triangle-right"></div>
```

```
.triangle-right {  
  width: 0;  
  height: 0;  
  border-top: 25px solid transparent;  
  border-bottom: 25px solid transparent;  
  border-left: 50px solid rgb(246, 156, 85);  
}
```

Треугольник - указательный левый



```
<div class="triangle-left"></div>
```

```
.triangle-left {  
  width: 0;  
  height: 0;  
  border-top: 25px solid transparent;  
  border-bottom: 25px solid transparent;  
  border-right: 50px solid rgb(246, 156, 85);  
}
```

Треугольник - Наведение вверх / вправо



```
<div class="triangle-up-right"></div>
```

```
.triangle-up-right {  
  width: 0;  
  height: 0;
```

```
border-top: 50px solid rgb(246, 156, 85);  
border-left: 50px solid transparent;  
}
```

Треугольник - Наведение вверх / влево



```
<div class="triangle-up-left"></div>
```

```
.triangle-up-left {  
  width: 0;  
  height: 0;  
  border-top: 50px solid rgb(246, 156, 85);  
  border-right: 50px solid transparent;  
}
```

Треугольник - указатель вниз / правый



```
<div class="triangle-down-right"></div>
```

```
.triangle-down-right {  
  width: 0;  
  height: 0;  
  border-bottom: 50px solid rgb(246, 156, 85);  
  border-left: 50px solid transparent;  
}
```

Треугольник - указатель вниз / влево



```
<div class="triangle-down-left"></div>
```

```
.triangle-down-left {
  width: 0;
  height: 0;
  border-bottom: 50px solid rgb(246, 156, 85);
  border-right: 50px solid transparent;
}
```

Всплески

Вспышка похожа на звезду, но с точками, простирающимися на меньшее расстояние от тела. Подумайте о форме всплеска, как квадрат с дополнительными, слегка повернутыми, квадратами, расположенными сверху.

Дополнительные квадраты создаются с использованием `::before` и `::after` psuedo-elements.

8-точечная развязка

8-точечная вспышка - это 2 слоистых квадрата. Нижний квадрат - это сам элемент, дополнительный квадрат создается с использованием `::before` псевдоэлементом. Дно повернуто на 20 °, верхний квадрат повернут на 135 °.



```
<div class="burst-8"></div>

.burst-8 {
  background: rgb(246, 156, 85);
  width: 40px;
  height: 40px;
  position: relative;
  text-align: center;
  -ms-transform: rotate(20deg);
  transform: rotate(20deg);
}

.burst-8::before {
  content: "";
  position: absolute;
  top: 0;
  left: 0;
  height: 40px;
  width: 40px;
  background: rgb(246, 156, 85);
  -ms-transform: rotate(135deg);
  transform: rotate(135deg);
}
```

12-точечная всплеск

12-точечный взрыв - это 3 слоистых квадрата. Нижний квадрат - это сам элемент, дополнительные квадраты создаются с использованием `:before` и `:after` псевдоэлементов. Дно повернуто на 0° , следующий квадрат поворачивается на 30° , а верх поворачивается на 60° .



```
<div class="burst-12"></div>

.burst-12 {
  width: 40px;
  height: 40px;
  position: relative;
  text-align: center;
  background: rgb(246, 156, 85);
}

.burst-12::before, .burst-12::after {
  content: "";
  position: absolute;
  top: 0;
  left: 0;
  height: 40px;
  width: 40px;
  background: rgb(246, 156, 85);
}

.burst-12::before {
  -ms-transform: rotate(30deg);
  transform: rotate(30deg);
}

.burst-12::after {
  -ms-transform: rotate(60deg);
  transform: rotate(60deg);
}
```

Круги и эллипсы

Круг

Чтобы создать **круг**, определите элемент с равной `width` и `height` (*квадрат*), а затем установите для свойства `border-radius` этого элемента значение `50%`.



HTML

```
<div class="circle"></div>
```

CSS

```
.circle {  
  width: 50px;  
  height: 50px;  
  background: rgb(246, 156, 85);  
  border-radius: 50%;  
}
```

Эллипс

Эллипс похож на круг, но с разными значениями для `width` и `height` .



HTML

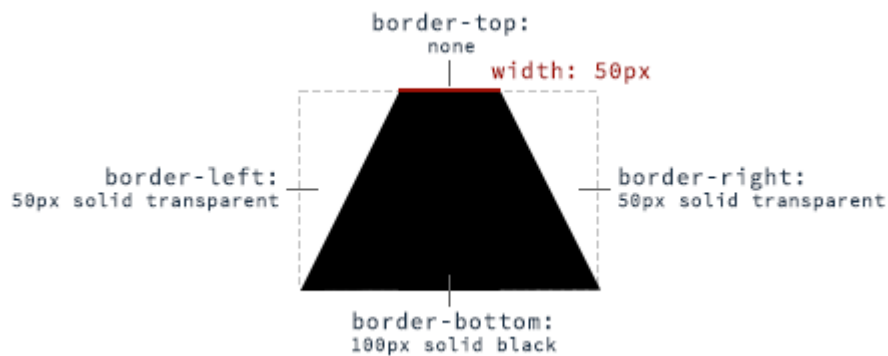
```
<div class="oval"></div>
```

CSS

```
.oval {  
  width: 50px;  
  height: 80px;  
  background: rgb(246, 156, 85);  
  border-radius: 50%;  
}
```

трапеция

Трапеция может быть выполнена блочным элементом с нулевой высотой (высота `0px`), шириной больше нуля и границей, прозрачной, за исключением одной стороны:



HTML:

```
<div class="trapezoid"></div>
```

CSS:

```
.trapezoid {  
  width: 50px;  
  height: 0;  
  border-left: 50px solid transparent;  
  border-right: 50px solid transparent;  
  border-bottom: 100px solid black;  
}
```

При изменении сторон границы можно настроить ориентацию трапеции.

куб

В этом примере показано, как создать куб, используя методы 2D-преобразования `skewX()` и `skewY()`.



HTML:

```
<div class="cube"></div>
```

CSS:

```
.cube {  
  background: #dc2e2e;  
}
```

```

width: 100px;
height: 100px;
position: relative;
margin: 50px;
}

.cube::before {
  content: '';
  display: inline-block;
  background: #f15757;
  width: 100px;
  height: 20px;
  transform: skewX(-40deg);
  position: absolute;
  top: -20px;
  left: 8px;
}

.cube::after {
  content: '';
  display: inline-block;
  background: #9e1515;
  width: 16px;
  height: 100px;
  transform: skewY(-50deg);
  position: absolute;
  top: -10px;
  left: 100%;
}

```

См. Демонстрацию

пирамида

В этом примере показано, как создать **пирамиду** с использованием границ и методов 2D-преобразования `skewY()` и `rotate()` `skewY()` .



HTML:

```
<div class="pyramid"></div>
```

CSS:

```
.pyramid {
```

```
width: 100px;
height: 200px;
position: relative;
margin: 50px;
}

.pyramid::before, .pyramid::after {
  content: '';
  display: inline-block;
  width: 0;
  height: 0;
  border: 50px solid;
  position: absolute;
}

.pyramid::before {
  border-color: transparent transparent #ff5656 transparent;
  transform: scaleY(2) skewY(-40deg) rotate(45deg);
}

.pyramid::after {
  border-color: transparent transparent #d64444 transparent;
  transform: scaleY(2) skewY(40deg) rotate(-45deg);
}
```

Прочитайте **Одиночные элементы онлайн**: <https://riptutorial.com/ru/css/topic/2862/одиночные-элементы>

глава 30: перелив

Синтаксис

- переполнение: видимое | скрытый | прокрутка | авто | начальная | наследовать;

параметры

Значение <code>Overflow</code>	подробности
<code>visible</code>	Показывает все переполняющее содержимое вне элемента
<code>scroll</code>	Скрывает переполняющий контент и добавляет полосу прокрутки
<code>hidden</code>	Скрывает переполненное содержимое, и полосы прокрутки исчезают, и страница фиксируется
<code>auto</code>	То же, что и <code>scroll</code> если содержимое переполняется, но не добавляет полосу прокрутки, если содержимое подходит
<code>inherit</code>	Наследует значение родительского элемента для этого свойства

замечания

Свойство `overflow` указывает, следует ли обрезать контент, отображать полосы прокрутки или растягивать контейнер для отображения содержимого, когда он переполняет свой контейнер уровня блока.

Когда элемент слишком мал, чтобы отображать его содержимое, что происходит? По умолчанию содержимое просто переполняется и отображается вне элемента. Это заставляет вашу работу выглядеть плохо. Вы хотите, чтобы ваша работа выглядела хорошо, поэтому вы устанавливаете свойство переполнения для обработки переполняющего содержимого желательным образом.

Значения для свойства `overflow` идентичны значениям свойств `overflow-x` и `overflow-y`, кроме того, что они применяются вдоль каждой оси

Свойство `overflow-wrap` также известно как свойство `word-wrap`.

Важное примечание. Использование свойства переполнения со значением, отличным от видимого, создаст новый контекст форматирования блока.

Examples

переполнение: прокрутка

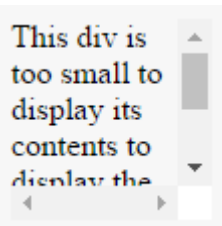
HTML

```
<div>
  This div is too small to display its contents to display the effects of the overflow
  property.
</div>
```

CSS

```
div {
  width:100px;
  height:100px;
  overflow:scroll;
}
```

Результат



Содержимое, указанное выше, обрезается в поле размером 100 пикселей на 100 пикселей, с возможностью прокрутки для просмотра переполненного содержимого.

Большинство настольных браузеров будут отображать как горизонтальные, так и вертикальные полосы прокрутки, независимо от того, обрезано или нет какое-либо содержимое. Это позволяет избежать проблем с появлением и исчезновением полос прокрутки в динамической среде. Принтеры могут печатать переполненный контент.

Переполнение-обертка

`overflow-wrap` сообщает браузеру, что он может сломать строку текста внутри целевого элемента на несколько строк в другом небьющемся месте. Помогает предотвратить длинную строку текста, вызывающую проблемы макета из-за переполнения контейнера.

CSS

```
div {
  width:100px;
  outline: 1px dashed #bbb;
}
```

```
#div1 {
  overflow-wrap:normal;
}

#div2 {
  overflow-wrap:break-word;
}
```

HTML

```
<div id="div1">
  <strong>#div1</strong>: Small words are displayed normally, but a long word like <span
  style="red;">supercalifragilisticexpialidocious</span> is too long so it will overflow past
  the edge of the line-break
</div>

<div id="div2">
  <strong>#div2</strong>: Small words are displayed normally, but a long word like <span
  style="red;">supercalifragilisticexpialidocious</span> will be split at the line break and
  continue on the next line.
</div>
```

#div1: Small words are displayed normally, but a long word like **supercalifragilisticexpialidoc:** is too long so it will overflow past the edge of the line-break

#div2: Small words are displayed normally, but a long word like **supercalifragilisticexpialidocious** will be split at the line break and continue on the next line.

overflow-wrap -
Значение

подробности

normal

Позволяет переполнение слова, если оно длиннее строки

<code>overflow-wrap</code> - Значение	подробности
<code>break-word</code>	Разделит слово на несколько строк, если необходимо
<code>inherit</code>	Наследует значение родительского элемента для этого свойства

переполнение: видимое

HTML

```
<div>
  Even if this div is too small to display its contents, the content is not clipped.
</div>
```

CSS

```
div {
  width:50px;
  height:50px;
  overflow:visible;
}
```

Результат



Содержимое не обрезается и будет отображаться вне поля содержимого, если оно превышает размер его контейнера.

Контекст форматирования блоков, созданный с помощью переполнения

Использование свойства `overflow` со значением, отличным от `visible`, создаст новый **контекст форматирования блока**. Это полезно для выравнивания элемента блока рядом с плавающим элементом.

CSS

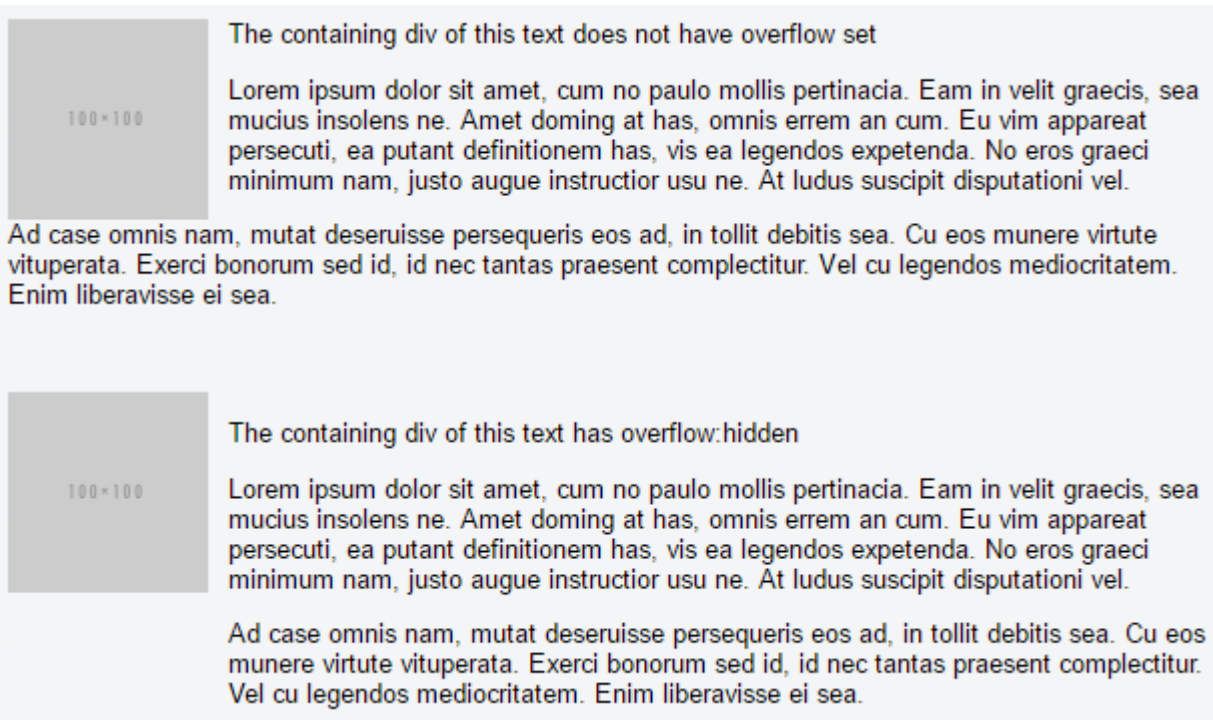
```
img {
  float:left;
  margin-right: 10px;
}
div {
  overflow:hidden; /* creates block formatting context */
}
```

HTML

```

<div>
  <p>Lorem ipsum dolor sit amet, cum no paulo mollis pertinacia.</p>
  <p>Ad case omnis nam, mutat deseruisse persequeris eos ad, in tollit debitis sea.</p>
</div>
```

Результат



The containing div of this text does not have overflow set

100x100

Lorem ipsum dolor sit amet, cum no paulo mollis pertinacia. Eam in velit graecis, sea mucius insolens ne. Amet doming at has, omnis errem an cum. Eu vim appareat persecuti, ea putant definitionem has, vis ea legendos expetenda. No eros graeci minimum nam, justo augue instructor usu ne. At ludus suscipit disputationi vel.

Ad case omnis nam, mutat deseruisse persequeris eos ad, in tollit debitis sea. Cu eos munere virtute vituperata. Exerci bonorum sed id, id nec tantas praesent complectitur. Vel cu legendos mediocritatem. Enim liberavisse ei sea.

The containing div of this text has overflow:hidden

100x100

Lorem ipsum dolor sit amet, cum no paulo mollis pertinacia. Eam in velit graecis, sea mucius insolens ne. Amet doming at has, omnis errem an cum. Eu vim appareat persecuti, ea putant definitionem has, vis ea legendos expetenda. No eros graeci minimum nam, justo augue instructor usu ne. At ludus suscipit disputationi vel.

Ad case omnis nam, mutat deseruisse persequeris eos ad, in tollit debitis sea. Cu eos munere virtute vituperata. Exerci bonorum sed id, id nec tantas praesent complectitur. Vel cu legendos mediocritatem. Enim liberavisse ei sea.

В этом примере показано, как абзацы внутри div с набором свойств `overflow` будут взаимодействовать с плавающим изображением.

переполнение-x и переполнение-y

Эти два свойства работают так же, как свойство `overflow` и принимают одинаковые значения. Параметр `overflow-x` работает только на оси x или слева направо. `overflow-y` работает на оси y или сверху вниз.

HTML

```
<div id="div-x">
  If this div is too small to display its contents,
```



```
    the content to the left and right will be clipped.
</div>

<div id="div-y">
    If this div is too small to display its contents,
    the content to the top and bottom will be clipped.
</div>
```

CSS

```
div {
    width: 200px;
    height: 200px;
}

#div-x {
    overflow-x: hidden;
}

#div-y {
    overflow-y: hidden;
}
```

Прочитайте перелив онлайн: <https://riptutorial.com/ru/css/topic/4947/перелив>

глава 31: Переходы

Синтаксис

- переход: [переход-свойство] [время перехода] [функция-время-переход] [задержка перехода];

параметры

параметр	подробности
переход-недвижимость	Специфический CSS свойство, значение которого изменение должно быть переведен (или) <code>all</code> , если все transitionable свойства нужно будет перенести.
Продолжительность перехода	Продолжительность (или период) в секундах (<code>s</code>) или миллисекундах (<code>ms</code>) , в течение которого переход должен иметь место.
переходное время функция	Функция, которая описывает, как рассчитываются промежуточные значения при переходе. Обычно используемые значения - это <code>ease</code> , <code>ease-in</code> , <code>ease-out</code> , <code>ease-in-out</code> , <code>linear</code> , <code>cubic-bezier()</code> , <code>steps()</code> . Более подробную информацию о различных функциях синхронизации можно найти в спецификациях W3C .
Переход задержки	Количество времени, которое должно было пройти до перехода, может начаться. Может быть указан в секундах (<code>s</code>) или миллисекунд (<code>ms</code>)

замечания

Некоторые старые браузеры поддерживают только свойства `transition` [префиксом](#) :

- `-webkit` : Chrome 25-, Safari 6-, Safari & Chrome для iOS 6.1-, Android 4.3- Browser, Blackberry Browser 7-, UC Browser 9.9- для Android.
- `-moz` : Firefox 15-.
- `-o` : Opera 11.5-, Opera Mobile 12-.

Пример:

```
-webkit-transition: all 1s;
```

```
-moz-transition: all 1s;
-o-transition: all 1s;
transition: all 1s;
```

Examples

Сокращение перехода

CSS

```
div{
  width: 150px;
  height:150px;
  background-color: red;
  transition: background-color 1s;
}
div:hover{
  background-color: green;
}
```

HTML

```
<div></div>
```

Этот пример изменит цвет фона, когда div будет зависать, изменение цвета фона будет длиться 1 секунду.

Переход (длинный)

CSS

```
div {
  height: 100px;
  width: 100px;
  border: 1px solid;
  transition-property: height, width;
  transition-duration: 1s, 500ms;
  transition-timing-function: linear;
  transition-delay: 0s, 1s;
}
div:hover {
  height: 200px;
  width: 200px;
}
```

HTML

```
<div></div>
```

- **transition-property** : указывает свойства CSS, для которых применяется эффект перехода. В этом случае div будет расширяться как по горизонтали, так и по вертикали при зависании.
- **duration-duration** : Определяет продолжительность времени, которое требуется выполнить переходу. В приведенном выше примере переходы высоты и ширины будут занимать 1 секунду и 500 миллисекунд соответственно.
- **Функция времени перехода** : задает кривую скорости эффекта перехода. *Линейное* значение указывает, что переход будет иметь одинаковую скорость от начала до конца.
- **delay-delay** : Задает время, необходимое для ожидания до начала действия перехода. В этом случае высота начнет переход сразу, тогда как ширина будет ждать 1 секунду.

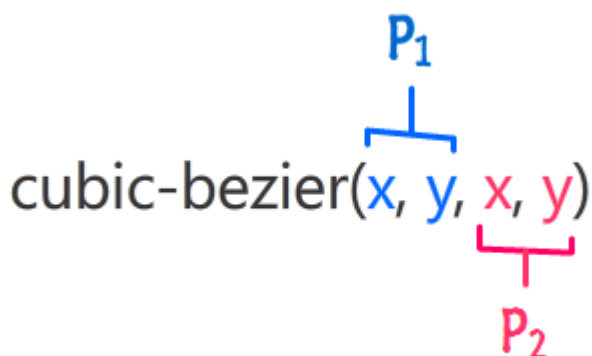
кубических Безье

Функция `cubic-bezier` - это функция времени перехода, которая часто используется для пользовательских и плавных переходов.

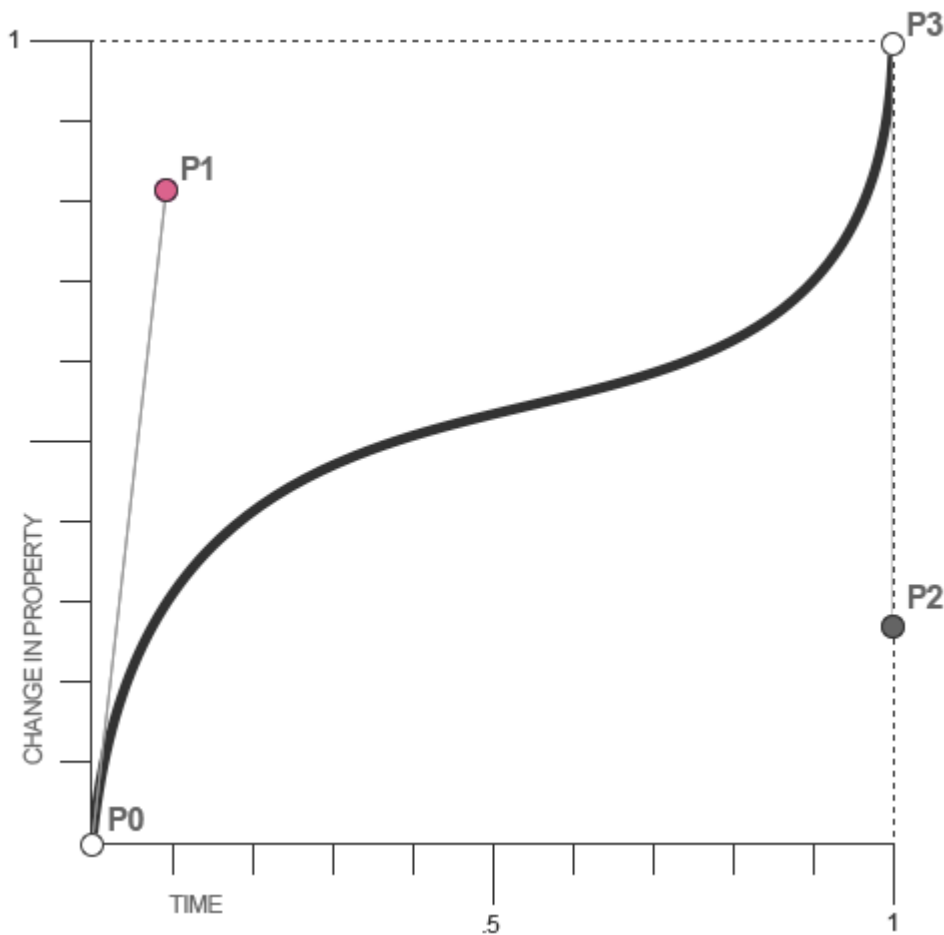
```
transition-timing-function: cubic-bezier(0.1, 0.7, 1.0, 0.1);
```

Функция принимает четыре параметра:

```
cubic-bezier(P1_x, P1_y, P2_x, P2_y)
```



Эти параметры будут отображаться в точки, которые являются частью [кривой Безье](#) :



Для кривых Безье Bzzier P0 и P3 всегда находятся в одном и том же месте. P0 находится в (0,0), а P3 - в (1,1), что означает, что параметры, переданные функции кубического безье, могут быть только между 0 и 1.

Если вы передадите параметры, которые не находятся в этом интервале, функция по умолчанию будет `linear`.

Поскольку кубический-безье является наиболее гибким переходом в CSS, вы можете перевести всю другую функцию синхронизации времени в функции кубического безье:

```
linear : cubic-bezier(0,0,1,1)
ease-in : cubic-bezier(0.42, 0.0, 1.0, 1.0)
ease-out : cubic-bezier(0.0, 0.0, 0.58, 1.0)
ease-in-out : cubic-bezier(0.42, 0.0, 0.58, 1.0)
```

Прочитайте [Переходы онлайн](https://riptutorial.com/ru/css/topic/751/переходы): <https://riptutorial.com/ru/css/topic/751/переходы>

глава 32: Поддержка браузера и префиксы

параметры

Префикс	Браузер (ы)
-webkit-	Google Chrome, Safari, более новые версии Opera 12 и выше, Android, Blackberry и UC-браузеры
-moz-	Mozilla Firefox
-ms-	Internet Explorer, Edge
-o- , -xv-	Опера до версии 12
-khtml-	Konquerer

замечания

Префиксы поставщиков используются для поддержки предварительного просмотра для новых функций CSS, где функциональность еще не рекомендована спецификацией.

Рекомендуется не использовать префиксы поставщиков в производственных средах. Эти префиксы существуют для тестирования новых функциональных возможностей, которые еще не доработаны, а поведение по своей сути является неожиданным. Простое использование префиксов **не** обеспечивает поддержку браузера для старых браузеров, так как вы не можете гарантировать, что функция не изменилась с течением времени, чтобы работать по-другому, и она все равно *может* быть повреждена в тех старых браузерах, которые вы утверждаете для поддержки.

Если поддержка старых браузеров важна, вам следует вместо этого использовать JavaScript или другие решения для имитации эффектов и поистине гарантировать поддержку старых браузеров.

Браузеры будут использовать свои префиксы и игнорировать свойства, которые они не понимают.

ПРИМЕЧАНИЕ . Префиксы должны всегда отображаться перед официальным синтаксисом без предварительного исправления. В противном случае они будут перезаписаны префиксными свойствами, что может быть другой реализацией в конце.

Если браузер поддерживает как непредусмотренную, так и префиксную версию свойства, самое последнее свойство, которое будет объявлено, будет иметь приоритет.

Examples

Переходы

```
div {
  -webkit-transition: all 4s ease;
  -moz-transition: all 4s ease;
  -o-transition: all 4s ease;
  transition: all 4s ease;
}
```

преобразование

```
div {
  -webkit-transform: rotate(45deg);
  -moz-transform: rotate(45deg);
  -ms-transform: rotate(45deg);
  -o-transform: rotate(45deg);
  transform: rotate(45deg);
}
```

Прочитайте Поддержка браузера и префиксы онлайн: <https://riptutorial.com/ru/css/topic/1138/поддержка-браузера-и-префиксы>

глава 33: позиционирование

Синтаксис

- position: static | absolute | fixed | relative | sticky | initial | inherit | unset;
- z-index: авто | *number* | initial | inherit;

параметры

параметр	подробности
статический	Значение по умолчанию. Элементы визуализируются в порядке, как они появляются в потоке документа. Свойства верхнего, правого, нижнего, левого и z-индекса не применяются.
родственник	Элемент позиционируется относительно его нормального положения, поэтому <code>left:20px</code> добавляет 20 пикселей в положение LEFT элемента
фиксированный	Элемент расположен относительно окна браузера
абсолютный	Элемент расположен относительно его первого позиционированного (не статического) элемента предка
начальная	Устанавливает это свойство по умолчанию.
унаследовать	Наследует это свойство от его родительского элемента.
липкий	Экспериментальная особенность. Он ведет себя как <code>position: static</code> внутри своего родителя, пока не будет достигнут заданный порог смещения, тогда он действует как <code>position: fixed</code> .
снята с охраны	Сочетание начального и наследуемого. Больше информации здесь .

замечания

Обычный поток - это поток элементов, если положение элемента *статично*.

1. определяющая *ширина* полезна, потому что в некоторых случаях она препятствует перекрытию содержимого элемента.

Examples

Фиксированная позиция

Определяя положение как фиксированное, мы можем удалить элемент из потока документов и установить его положение относительно окна браузера. Одно очевидное использование - когда мы хотим, чтобы что-то было видно, когда мы прокручиваем нижнюю часть длинной страницы.

```
#stickyDiv {
  position:fixed;
  top:10px;
  left:10px;
}
```

Перекрывающиеся элементы с z-индексом

Чтобы изменить позиции по умолчанию, расположенные по **порядку** расположения **стека** (свойство `position` установлено на `relative`, `absolute` или `fixed`), используйте свойство `z-index`.

Чем выше z-индекс, тем выше он находится в контексте стекирования (по оси z).

пример

В приведенном ниже примере значение z-индекса 3 помещается зеленым сверху, z-индекс 2 ставит красный под ним, а z-индекс 1 ставит под ним синий.

HTML

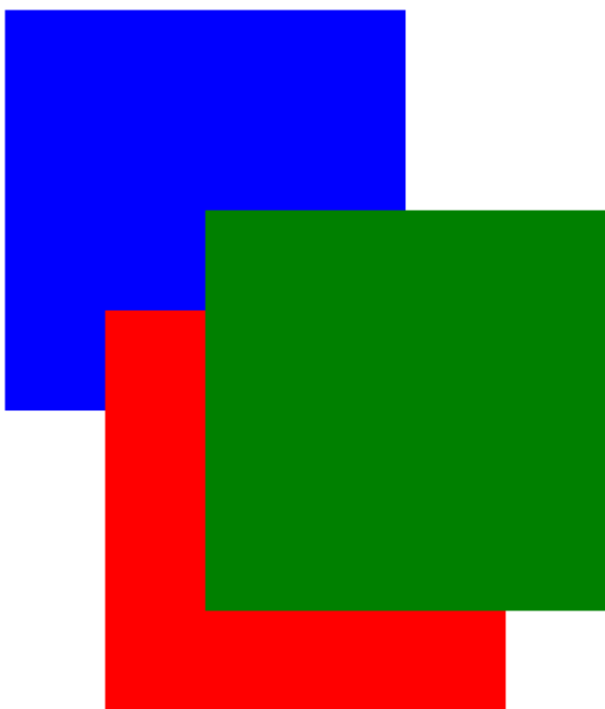
```
<div id="div1"></div>
<div id="div2"></div>
<div id="div3"></div>
```

CSS

```
div {
  position: absolute;
  height: 200px;
  width: 200px;
}
div#div1 {
  z-index: 1;
  left: 0px;
  top: 0px;
  background-color: blue;
}
div#div2 {
```

```
z-index: 3;
left: 100px;
top: 100px;
background-color: green;
}
div#div3 {
z-index: 2;
left: 50px;
top: 150px;
background-color: red;
}
```

Это создает следующий эффект:



См. Рабочий пример в [JSFiddle](#) .

Синтаксис

```
z-index: [ number ] | auto;
```

параметр	подробности
number	Целое число. Более высокое число выше в стеке <code>z-index</code> . 0 - значение по умолчанию. Отрицательные значения допустимы.
auto	Дает элементу тот же контекст стека, что и его родитель. (По умолчанию)

замечания

Все элементы расположены в 3D-оси в CSS, включая ось глубины, измеряемую свойством `z-index`. `z-index` работает только с позиционируемыми элементами: (см .: [Почему z-index нуждается в определенной позиции для работы?](#)). Единственное значение, в котором оно игнорируется, - это значение по умолчанию, `static` .

Читайте о свойстве `z-index` и `Stacking Contexts` в [спецификации CSS](#) в многоуровневой презентации и в сети [разработчиков Mozilla](#) .

Относительная позиция

Относительное позиционирование перемещает элемент по отношению к тому, где он находился бы в *нормальном потоке*. Свойства эффекта:

1. Топ
2. оставил
3. право
4. низ

используются для указания того, как далеко перемещать элемент из того места, где он находился бы в нормальном потоке.

```
.relpos{
  position:relative;
  top:20px;
  left:30px;
}
```

Этот код будет перемещать поле, содержащее элемент с атрибутом `class = "relpos"` 20px down и 30px справа от того места, где он находился бы в нормальном потоке.

Абсолютная позиция

Когда используется абсолютное позиционирование, поле желаемого элемента выводится из *нормального потока* и больше не влияет на положение других элементов на странице. Свойства смещения:

1. Топ
2. оставил
3. право
4. низ

указать, что элемент должен появиться относительно его следующего нестатического содержащего элемента.

```
.abspos{
  position:absolute;
  top:0px;
  left:500px;
}
```

Этот код будет перемещать поле, содержащее элемент с атрибутом `class="abspos"` down 0px и right 500px относительно его содержащего элемента.

Статическое позиционирование

Позиция элемента по умолчанию является `static`. Прочитать [MDN](#) :

Это ключевое слово позволяет элементу использовать нормальное поведение, то есть оно выложено в текущем положении в потоке. Свойства верхнего, правого, нижнего, левого и z-индекса не применяются.

```
.element{
  position:static;
}
```

Прочитайте позиционирование онлайн: <https://riptutorial.com/ru/css/topic/935/>
позиционирование

глава 34: Пользовательские свойства (переменные)

Вступление

Переменные CSS позволяют авторам создавать повторно используемые значения, которые могут использоваться в документе CSS.

Например, в CSS часто используется для повторного использования одного цвета в документе. До CSS-переменных это означало бы многократно использовать одно и то же значение цвета во всем документе. С помощью CSS-переменных значение цвета может быть присвоено переменной и указано в нескольких местах. Это упрощает изменение значений и является более смысловым, чем использование традиционных значений CSS.

Синтаксис

- `: root { } / * псевдокласс, позволяющий более глобальное определение переменных * /`
- `--variable-name: значение ; / * определить переменную * /`
- `var (- variable-name, default-value) / * использовать определенную переменную со значением значения по умолчанию * /`

замечания

Переменные CSS в настоящее время считаются экспериментальной технологией.

ПОДДЕРЖКА / СОВМЕСТИМОСТЬ БРАУЗЕРА

Firefox: версия **31** + (по умолчанию включена)

[Дополнительная информация от Mozilla](#)

Chrome: версия **49** + (по умолчанию включена) .

«Эта функция может быть включена в Chrome версии 48 для тестирования, включив функцию `experimental Web Platform` . Введите этот `chrome://flags/` в адресной строке Chrome».

IE: Не поддерживается.

Edge: [Under Development](#)

Safari: версия **9.1+**

Examples

Переменный цвет

```
:root {
  --red: #b00;
  --blue: #4679bd;
  --grey: #ddd;
}
.Bx1 {
  color: var(--red);
  background: var(--grey);
  border: 1px solid var(--red);
}
```

Переменные размеры

```
:root {
  --W200: 200px;
  --W10: 10px;
}
.Bx2 {
  width: var(--W200);
  height: var(--W200);
  margin: var(--W10);
}
```

Переменный каскадный

CSS-переменные каскадом во многом аналогичны другим свойствам и могут быть легко скопированы.

Вы можете определять переменные несколько раз, и только определение с наивысшей специфичностью будет применяться к выбранному элементу.

Предполагая этот HTML:

```
<a class="button">Button Green</a>
<a class="button button_red">Button Red</a>
<a class="button">Button Hovered On</a>
```

Мы можем написать этот CSS:

```
.button {
  --color: green;
  padding: .5rem;
  border: 1px solid var(--color);
  color: var(--color);
}

.button:hover {
```

```
--color: blue;
}

.button_red {
  --color: red;
}
```

И получите этот результат:



Допустимы / Инвалидам

Именование При указании переменных CSS он содержит только буквы и тире, как и другие свойства CSS (например: line-height, -moz-box-sizing), но он должен начинаться с двойных тире (-)

```
//These are Invalids variable names
--123color: blue;
--#color: red;
--bg_color: yellow
--$width: 100px;

//Valid variable names
--color: red;
--bg-color: yellow
--width: 100px;
```

Переменные CSS чувствительны к регистру.

```
/* The variable names below are all different variables */
--pcolor: ;
--Pcolor: ;
--pColor: ;
```

Пустое пространство Vs

```
/* Invalid */
--color;;

/* Valid */
--color: ; /* space is assigned */
```

сцеплений

```
/* Invalid - CSS doesn't support concatenation*/
.logo{
  --logo-url: 'logo';
  background: url('assets/img/' var(--logo-url) '.png');
}
```

```

/* Invalid - CSS bug */
.logo{
  --logo-url: 'assets/img/logo.png';
  background: url(var(--logo-url));
}

/* Valid */
.logo{
  --logo-url: url('assets/img/logo.png');
  background: var(--logo-url);
}

```

Осторожно при использовании единиц

```

/* Invalid */
--width: 10;
width: var(--width)px;

/* Valid */
--width: 10px;
width: var(--width);

/* Valid */
--width: 10;
width: calc(1px * var(--width)); /* multiply by 1 unit to convert */
width: calc(1em * var(--width));

```

С медиа-запросами

Вы можете повторно устанавливать переменные в медиа-запросах и каскадировать эти новые значения везде, где они используются, что невозможно с препроцессорными переменными.

Здесь медиа-запрос изменяет переменные, используемые для настройки очень простой сетки:

HTML

```

<div></div>
<div></div>
<div></div>
<div></div>

```

CSS

```

:root{
  --width: 25%;
  --content: 'This is desktop';
}
@media only screen and (max-width: 767px){
  :root{
    --width:50%;
    --content: 'This is mobile';
  }
}

```



```
}
@media only screen and (max-width: 480px){
  :root{
    --width:100%;
  }
}

div{
  width: calc(var(--width) - 20px);
  height: 100px;
}
div:before{
  content: var(--content);
}

/* Other Styles */
body {
  padding: 10px;
}

div{
  display: flex;
  align-items: center;
  justify-content: center;
  font-weight:bold;
  float:left;
  margin: 10px;
  border: 4px solid black;
  background: red;
}
```

Вы можете попробовать [изменить размер](#) окна в этой [демо-версии CodePen](#)

Вот анимированный снимок экрана об изменении размера:

This is desktop

This is desktop

This is desktop

This is desktop

Прочитайте Пользовательские свойства (переменные) онлайн:

<https://riptutorial.com/ru/css/topic/1755/пользовательские-свойства--переменные->

глава 35: помутнение

Синтаксис

- `opacity`: число (* строго от 0 до 1) | наследовать | начальная | снята с охраны;

замечания

Если вы не хотите применять непрозрачность, вы можете использовать это вместо этого:

фон: `rgba (255, 255, 255, 0,6);`

Ресурсы:

- MDN: <https://developer.mozilla.org/en/docs/Web/CSS/opacity> ;
- Прозрачность W3C: свойство «непрозрачность»: <https://www.w3.org/TR/css3-color/#transparency>
- Поддержка браузера: <http://caniuse.com/#feat=css-opacity>

Examples

Свойство непрозрачности

Прозрачность элемента может быть задана с использованием свойства `opacity` . Значения могут быть от 0.0 (прозрачные) до 1.0 (непрозрачные).

Пример использования

```
<div style="opacity:0.8;">  
  This is a partially transparent element  
</div>
```

Стоимость имущества	прозрачность
<code>opacity: 1.0;</code>	непрозрачный
<code>opacity: 0.75;</code>	25% прозрачный (75% непрозрачный)
<code>opacity: 0.5;</code>	50% прозрачный (50% непрозрачный)
<code>opacity: 0.25;</code>	75% прозрачный (25% непрозрачный)
<code>opacity: 0.0;</code>	прозрачный

Совместимость IE для непрозрачности

Чтобы использовать `opacity` во всех версиях IE, порядок:

```
.transparent-element {  
  /* for IE 8 & 9 */  
  -ms-filter:"progid:DXImageTransform.Microsoft.Alpha(Opacity=60)"; // IE8  
  /* works in IE 8 & 9 too, but also 5, 6, 7 */  
  filter: alpha(opacity=60); // IE 5-7  
  /* Modern Browsers */  
  opacity: 0.6;  
}
```

Прочитайте помутнение онлайн: <https://riptutorial.com/ru/css/topic/2864/помутнение>

глава 36: Поплавки

Синтаксис

- ясно: нет | левый | право | оба | inline-start | рядный конец;
- поплавок: левый | право | нет | inline-start | рядный конец;

замечания

Поскольку float подразумевает использование макета блока, в некоторых случаях он изменяет вычисленное значение отображаемых значений [1]

[1]: <https://developer.mozilla.org/en-US/docs/Web/CSS/float> MDN

Examples

Поплавок изображения внутри текста

Самое основное использование float - это перенос текста вокруг изображения. В приведенном ниже коде будут отображаться два абзаца и изображение, при этом второй абзац течет вокруг изображения. Обратите внимание, что он всегда является содержимым *после того*, как перемещаемый элемент перемещается вокруг перемещаемого элемента.

HTML:

```
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer nec odio. Praesent libero. Sed cursus ante dapibus diam. Sed nisi. Nulla quis sem at nibh elementum imperdiet. Duis sagittis ipsum. Praesent mauris. Fusce nec tellus sed augue semper porta. Mauris massa. Vestibulum lacinia arcu eget nulla. </p>



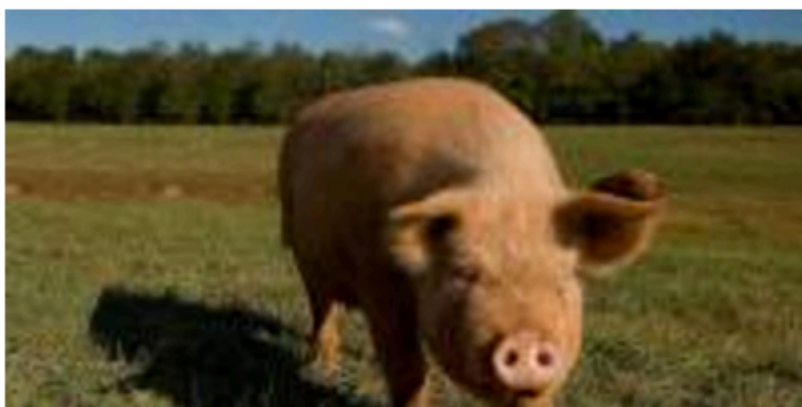
<p>Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Curabitur sodales ligula in libero. Sed dignissim lacinia nunc. Curabitur tortor. Pellentesque nibh. Aenean quam. In scelerisque sem at dolor. Maecenas mattis. Sed convallis tristique sem. Proin ut ligula vel nunc egestas porttitor. Morbi lectus risus, iaculis vel, suscipit quis, luctus non, massa. Fusce ac turpis quis ligula lacinia aliquet. </p>
```

CSS:

```
img {
  float:left;
  margin-right:1rem;
}
```

Это будет выход

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer nec odio. Praesent libero. Sed cursus ante dapibus diam. Sed nisi. Nulla quis sem at nibh elementum imperdiet. Duis sagittis ipsum. Praesent mauris. Fusce nec tellus sed augue semper porta. Mauris massa. Vestibulum lacinia arcu eget nulla.



Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Curabitur sodales ligula in libero.

Sed dignissim lacinia nunc. Curabitur tortor. Pellentesque nibh. Aenean quam. In scelerisque sem at dolor. Maecenas mattis. Sed convallis tristique sem. Proin ut ligula vel nunc egestas porttitor. Morbi lectus risus, iaculis vel, suscipit quis, luctus non, massa. Fusce ac turpis quis ligula lacinia aliquet.

[Codepen Link](#)

Простая комбинация столбцов с фиксированной шириной

Простой двухколоночный макет состоит из двух элементов с фиксированной шириной, плавающих. Обратите внимание, что в этом примере боковая панель и область содержимого не имеют одинаковой высоты. Это одна из сложнейших частей с многоколоночными макетами с использованием поплавков и требует обходных решений, чтобы сделать несколько столбцов одинаковой высоты.

HTML:

```
<div class="wrapper">

<div class="sidebar">
  <h2>Sidebar</h2>

  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer nec odio.</p>
</div>

<div class="content">
  <h1>Content</h1>

  <p>Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos
himenaeos. Curabitur sodales ligula in libero. Sed dignissim lacinia nunc. Curabitur tortor.
Pellentesque nibh. Aenean quam. In scelerisque sem at dolor. Maecenas mattis. Sed convallis
tristique sem. Proin ut ligula vel nunc egestas porttitor. Morbi lectus risus, iaculis vel,
suscipit quis, luctus non, massa. Fusce ac turpis quis ligula lacinia aliquet. </p>
</div>

</div>
```

CSS:

```
.wrapper {
  width:600px;
  padding:20px;
  background-color:pink;

  /* Floated elements don't use any height. Adding "overflow:hidden;" forces the
  parent element to expand to contain its floated children. */
  overflow:hidden;
}

.sidebar {
  width:150px;
  float:left;
  background-color:blue;
}

.content {
  width:450px;
  float:right;
  background-color:yellow;
}
```

Простая комбинация столбцов с фиксированной шириной

HTML:

```
<div class="wrapper">
  <div class="left-sidebar">
    <h1>Left Sidebar</h1>
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. </p>
  </div>
  <div class="content">
    <h1>Content</h1>
```

```
<p>Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Curabitur sodales ligula in libero. Sed dignissim lacinia nunc. Curabitur tortor. Pellentesque nibh. Aenean quam. In scelerisque sem at dolor. Maecenas mattis. Sed convallis tristique sem. Proin ut ligula vel nunc egestas porttitor. Morbi lectus risus, iaculis vel, suscipit quis, luctus non, massa. </p>
</div>
<div class="right-sidebar">
  <h1>Right Sidebar</h1>
  <p>Fusce ac turpis quis ligula lacinia aliquet.</p>
</div>
</div>
```

CSS:

```
.wrapper {
  width:600px;
  background-color:pink;
  padding:20px;

  /* Floated elements don't use any height. Adding "overflow:hidden;" forces the
  parent element to expand to contain its floated children. */
  overflow:hidden;
}

.left-sidebar {
  width:150px;
  background-color:blue;
  float:left;
}

.content {
  width:300px;
  background-color:yellow;
  float:left;
}

.right-sidebar {
  width:150px;
  background-color:green;
  float:right;
}
```

Двухколоночный ленивый / жадный макет

Этот макет использует один плавающий столбец для создания двухколоночного макета без определенной ширины. В этом примере левая боковая панель «ленива», поскольку она занимает всего лишь столько места, сколько нужно. Другой способ сказать это, что левая боковая панель «обернута термоусадочной пленкой». Правильный столбец содержимого «жадный», поскольку он занимает все оставшееся пространство.

HTML:

```
<div class="sidebar">
  <h1>Sidebar</h1>
  
```



```
</div>

<div class="content">
<h1>Content</h1>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer nec odio. Praesent libero. Sed cursus ante dapibus diam. Sed nisi. Nulla quis sem at nibh elementum imperdiet. Duis sagittis ipsum. Praesent mauris. Fusce nec tellus sed augue semper porta. Mauris massa. Vestibulum lacinia arcu eget nulla. </p>
<p>Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Curabitur sodales ligula in libero. Sed dignissim lacinia nunc. Curabitur tortor. Pellentesque nibh. Aenean quam. In scelerisque sem at dolor. Maecenas mattis. Sed convallis tristique sem. Proin ut ligula vel nunc egestas porttitor. Morbi lectus risus, iaculis vel, suscipit quis, luctus non, massa. Fusce ac turpis quis ligula lacinia aliquet. Mauris ipsum. Nulla metus metus, ullamcorper vel, tincidunt sed, euismod in, nibh. </p>
</div>
```

CSS:

```
.sidebar {
  /* `display:table;` shrink-wraps the column */
  display:table;
  float:left;
  background-color:blue;
}

.content {
  /* `overflow:hidden;` prevents `.content` from flowing under `.sidebar` */
  overflow:hidden;
  background-color:yellow;
}
```

скрипка

ОЧИСТИТЬ СОБСТВЕННОСТЬ

Свойство `clear` напрямую связано с поплавками. Значения свойств:

- `none` - По умолчанию. Позволяет плавающие элементы с обеих сторон
- `left` - плавающие элементы не разрешены с левой стороны
- `справа` - плавающие элементы не допускаются с правой стороны
- `оба` - не допускаются плавающие элементы на левой или правой стороне
- `initial` - Устанавливает это свойство по умолчанию. Читайте об исходных
- `inherit` - Наследует это свойство от его родительского элемента. Читайте о наследовании

```
<html>
<head>
<style>
img {
  float: left;
}

p.clear {
  clear: both;
}
```

```
}
</style>
</head>
<body>


<p>Lorem ipsoum Lorem ipsoum Lorem ipsoum Lorem ipsoum Lorem ipsoum Lorem ipsoum Lorem ipsoum
Lorem ipsoum Lorem ipsoum Lorem ipsoum Lorem ipsoum Lorem ipsoum </p>
<p class="clear">Lorem ipsoum Lorem ipsoum Lorem ipsoum Lorem ipsoum Lorem ipsoum Lorem ipsoum Lorem ipsoum
Lorem ipsoum Lorem ipsoum Lorem ipsoum Lorem ipsoum Lorem ipsoum Lorem ipsoum </p>

</body>
</html>
```

Clearfix

Ручка clearfix - популярный способ содержать поправки (N. Gallagher aka @necolas)

Не следует путать с `clear` собственностью, `clearfix` этого *понятия* (что также связана с поправками, таким образом, возможной путаница). Чтобы *содержать поправки*, вы должны добавить `.cf` или `.clearfix` в контейнер (**родительский** `.cf` и `.clearfix` ЭТОТ класс с помощью нескольких правил, описанных ниже.

3 версии с немного отличающимися эффектами (источники: [новый взлом micro clearfix](#) Н. Галлахера и [clearfix, загруженный](#) TJ Koblentz):

Clearfix (с кратковременным сглаживанием оставшихся поправков)

```
.cf:after {
  content: "";
  display: table;
}

.cf:after {
  clear: both;
}
```

Clearfix также предотвращает свертывание верхнего края содержащихся поправков

```
/**
 * For modern browsers
 * 1. The space content is one way to avoid an Opera bug when the
 *    contenteditable attribute is included anywhere else in the document.
```

```
* Otherwise it causes space to appear at the top and bottom of elements
* that are clearfixed.
* 2. The use of `table` rather than `block` is only necessary if using
* `:before` to contain the top-margins of child elements.
*/
.cf:before,
.cf:after {
  content: " "; /* 1 */
  display: table; /* 2 */
}

.cf:after {
  clear: both;
}
```

Clearfix с поддержкой устаревших браузеров IE6 и IE7

```
.cf:before,
.cf:after {
  content: " ";
  display: table;
}

.cf:after {
  clear: both;
}

/**
 * For IE 6/7 only
 * Include this rule to trigger hasLayout and contain floats.
 */
.cf {
  *zoom: 1;
}
```

[Codepen с эффектом clearfix](#)

Другой ресурс: [все, что вы знаете о clearfix, неверно](#) (clearfix и BFC - Контекст блока форматирования, в то время как hasLayout относится к устаревшим браузерам IE6, возможно, 7)

Встроенный DIV с использованием float

Элемент `div` является элементом уровня блока, то есть он занимает всю ширину страницы, а братья и сестры помещают один ниже другого независимо от их ширины.

```
<div>
  <p>This is DIV 1</p>
```

```
</div>
<div>
  <p>This is DIV 2</p>
</div>
```

Вывод следующего кода будет



This is DIV 1

This is DIV 2

Мы можем сделать их в строке, добавив свойство `float` CSS в `div` .

HTML:

```
<div class="outer-div">
  <div class="inner-div1">
    <p>This is DIV 1</p>
  </div>
  <div class="inner-div2">
    <p>This is DIV 2</p>
  </div>
</div>
```

CSS

```
.inner-div1 {
  width: 50%;
  margin-right:0px;
  float:left;
  background : #337ab7;
  padding:50px 0px;
}

.inner-div2 {
  width: 50%;
  margin-right:0px;
  float:left;
  background : #dd2c00;
  padding:50px 0px;
}

p {
  text-align:center;
}
```



This is DIV 1

[Codepen Link](#)

Использование свойства переполнения для очистки поплавок

Установка значения `overflow` для `hidden` , `auto` или `scroll` к элементу очистит все поправки внутри этого элемента.

Примечание: использование `overflow:scroll` всегда будет показывать `overflow:scroll`

Прочитайте Поплавки онлайн: <https://riptutorial.com/ru/css/topic/405/поплавки>

глава 37: Псевдо-элементы

Вступление

Псевдоэлементы, как и псевдоклассы, добавляются к селекторам CSS, но вместо описания специального состояния они позволяют вам определять область и стиль определенных частей элемента html.

Например, псевдо-элемент `:: first-letter` нацелен только на первую букву элемента блока, заданного селектором.

Синтаксис

- `selector :: pseudo-element {свойство: значение}`

параметры

Псевдо-элемент	Описание
<code>::after</code>	Вставить содержимое после содержимого элемента
<code>::before</code>	Вставлять содержимое перед содержимым элемента
<code>::first-letter</code>	Выбирает первую букву каждого элемента
<code>::first-line</code>	Выбирает первую строку каждого элемента
<code>::selection</code>	Соответствует части элемента, выбранного пользователем.
<code>::backdrop</code>	Используется для создания фона, который скрывает базовый документ для элемента в стеке верхнего уровня
<code>::placeholder</code>	Позволяет вам стиль текста заполнителя элемента формы (экспериментальный)
<code>::marker</code>	Для применения атрибутов списка в данном элементе (Experimental)
<code>::spelling-error</code>	Представляет текстовый сегмент, который браузер помечен как неправильно написанный (Experimental)
<code>::grammar-error</code>	Представляет текстовый сегмент, который браузер помечен как грамматически некорректный (экспериментальный)

замечания

- Иногда вы видите двойные двоеточия (:: :) вместо одного (: . Это способ отделить псевдо-классы от псевдо-элементов, но некоторые старые браузеры , такие как Internet Explorer 8 поддерживает **только** одного двоеточие (:) для псевдо-элементов.
- В селекторе можно использовать только один псевдоэлемент. Он должен появиться после простых селекторов в инструкции.
- Псевдоэлементы не являются частью DOM, поэтому не могут быть нацелены на `:hover` или другие пользовательские события.

Examples

Псевдо-элементы

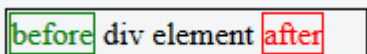
Псевдоэлементы добавляются в селекторы, но вместо описания специального состояния они позволяют вам стилизовать определенные части документа.

Атрибут `content` необходим для рендеринга псевдоэлементов; однако атрибут может иметь пустое значение (например, `content: ""`).

```
div::after {
  content: 'after';
  color: red;
  border: 1px solid red;
}

div {
  color: black;
  border: 1px solid black;
  padding: 1px;
}

div::before {
  content: 'before';
  color: green;
  border: 1px solid green;
}
```



Псевдоэлементы в списках

Псевдоэлементы часто используются для изменения внешнего вида списков (в основном для неупорядоченных списков, `ul`).

Первый шаг - удалить пули по умолчанию:

```
ul {
  list-style-type: none;
}
```

Затем вы добавляете пользовательский стиль. В этом примере мы создадим градиентные поля для пуль.

```
li:before {
  content: "";
  display: inline-block;
  margin-right: 10px;
  height: 10px;
  width: 10px;
  background: linear-gradient(red, blue);
}
```

HTML

```
<ul>
  <li>Test I</li>
  <li>Test II</li>
</ul>
```

Результат



Прочитайте Псевдо-элементы онлайн: <https://riptutorial.com/ru/css/topic/911/псевдо-элементы>

глава 38: Свойство фильтра

Синтаксис

- filter: none (значение по умолчанию)
- filter: initial (по умолчанию - none);
- filter: inherit (по умолчанию - родительское значение);
- фильтр: размытие (px)
- фильтр: яркость (число |%)
- фильтр: контраст (число |%)
- filter: drop-shadow (vertical-shadow-px vertical-shadow-px shadow-blur-px shadow- - цвет распространения)
- filter: greyscale (число |%)
- фильтр: оттенок-поворот (град)
- фильтр: инвертировать (число |%)
- фильтр: непрозрачность (число |%)
- фильтр: насыщенный (число |%)
- фильтр: сепия (число |%)

параметры

Значение	Описание
размытие (x)	Размывает изображение по пикселям x.
Яркость (x)	Уменьшает изображение при любом значении выше 1,0 или 100%. Ниже этого изображения будет затемнено.
Контраст (x)	Обеспечивает более контрастность изображения при любом значении выше 1,0 или 100%. Ниже этого изображения будет менее насыщенным.
drop-shadow (h, v, x, y, z)	Дает изображение тень. h и v могут иметь отрицательные значения. x, y и z необязательны.
полутонное (x)	Показывает изображение в оттенках серого с максимальным значением 1,0 или 100%.
Оттенок поворота (x)	Применяет оттенок-поворот к изображению.
инвертировать (x)	Инвертирует цвет изображения с максимальным значением 1,0 или 100%.

Значение	Описание
Непрозрачность (x)	Устанавливает, насколько непрозрачно / прозрачно изображение с максимальным значением 1.0 или 100%.
насыщают (x)	Насыщает изображение при любом значении выше 1,0 или 100%. Ниже этого изображения начнет дезатуироваться.
сепия (x)	Преобразует изображение в сепию с максимальным значением 1,0 или 100%.

замечания

1. Поскольку фильтр является экспериментальной функцией, вы должны использовать префикс `-webkit`. Он может меняться в синтаксисе и поведении, но изменения, вероятно, будут небольшими.
2. Он может не поддерживаться в более старых версиях основных браузеров. В мобильных браузерах он может быть полностью неподдерживаемым.
3. Из-за относительно ограниченной поддержки попробуйте использовать `box-shadow` вместо `filter: drop-shadow()`. Используйте `opacity` вместо `filter: opacity()`.
4. Он может быть анимирован через Javascript / jQuery. Для Javascript используйте `object.style.WebkitFilter`.
5. Проверьте [W3Schools](#) или [MDN](#) для получения дополнительной информации.
6. W3Schools также имеет [демонстрационную страницу](#) для всех типов значений фильтра.

Examples

Drop Shadow (вместо этого используйте вместо него тень `box-shadow`)

HTML

```
<p>My shadow always follows me.</p>
```

CSS

```
p {
  -webkit-filter: drop-shadow(10px 10px 1px green);
  filter: drop-shadow(10px 10px 1px green);
}
```

Результат

My shadow always follows me.
My shadow always follows me.

Множественные значения фильтра

Чтобы использовать несколько фильтров, отделите каждое значение пробелом.

HTML

```
<img src='donald-duck.png' alt='Donald Duck' title='Donald Duck' />
```

CSS

```
img {  
  -webkit-filter: brightness(200%) grayscale(100%) sepia(100%) invert(100%);  
  filter: brightness(200%) grayscale(100%) sepia(100%) invert(100%);  
}
```

Результат



Оттенок

HTML

```
<img src='donald-duck.png' alt='Donald Duck' title='Donald Duck' />
```

CSS

```
img {  
  -webkit-filter: hue-rotate(120deg);  
  filter: hue-rotate(120deg);  
}
```

Результат



Инvertировать цвет

HTML

```
<div></div>
```

CSS

```
div {  
  width: 100px;  
  height: 100px;  
  background-color: white;  
  -webkit-filter: invert(100%);  
  filter: invert(100%);  
}
```

Результат



Переходит от белого к черному.

ПЯТНО

HTML

```
<img src='donald-duck.png' alt='Donald Duck' title='Donald Duck' />
```

CSS

```
img {  
  -webkit-filter: blur(1px);  
  filter: blur(1px);  
}
```

Результат



Заставляет тебя тереть очки.

Прочитайте Свойство фильтра онлайн: <https://riptutorial.com/ru/css/topic/1567/свойство-фильтра>

глава 39: Селекторы

Вступление

Селекторы CSS идентифицируют определенные HTML-элементы в качестве целей для стилей CSS. В этом разделе рассказывается о том, как селектор CSS нацелен на элементы HTML. Селекторы используют широкий диапазон из более чем 50 методов выбора, предлагаемых языком CSS, включая элементы, классы, идентификаторы, псевдоэлементы и псевдоклассы и шаблоны.

Синтаксис

- `# id`
- `, имя_класса`
- `: псевдо-имя класса`
- `:: псевдо-elementname`
- `[attr] / * имеет атрибут attr . * /`
- `[attr = " value "] / * имеет атрибут attr , а его значение точно « value » . * /`
- `[attr ~ = " значение "] / * имеет атрибут attr , а его значение при разбиении по пробелу содержит « значение » . * /`
- `[attr | = " value "] / * имеет атрибут attr , а его значение точно « value » , или его значение начинается с « value - » . * /`
- `[attr ^ = " value "] / * имеет атрибут attr , а его значение начинается с " value " . * /`
- `[attr $ = " value "] / * имеет атрибут attr , а его значение заканчивается « значением » . * /`
- `[attr * = " значение "] / * имеет атрибут attr , а его значение содержит « значение » . * /`
- `имя элемента`
- `*`

замечания

- Иногда вы видите двойные двоеточия (`::`) вместо одного (`:`). Это способ отделить [псевдоклассы](#) от [псевдоэлементов](#) .
- Старые браузеры, как Internet Explorer 8, поддерживают **только** один двоеточие (`:`) для определения псевдо-элементов.
- В отличие от псевдоклассов, для каждого селектора может использоваться только один псевдоэлемент, если он присутствует, он должен появляться после последовательности простых селекторов, которые представляют объекты селектора (будущая версия [спецификации W3C](#) может допускать множественные псевдоэлементы на селектор).

Examples

Селекторы атрибутов

обзор

Селекторы атрибутов могут использоваться с различными типами операторов, которые соответствующим образом меняют критерии выбора. Они выбирают элемент, используя наличие заданного атрибута или значения атрибута.

Селектор (1)	Соответствующий элемент	Выбирает элементы ...	Версия CSS
[attr]	<div attr>	С атрибутом attr	2
[attr='val']	<div attr="val">	Где атрибут attr имеет значение val	2
[attr~='val']	<div attr="val val2 val3">	Где val появляется в список attr разделенных attr	2
[attr^='val']	<div attr="val1 val2">	Где значение attr <i>начинается с</i> val	3
[attr\$='val']	<div attr="sth aval">	Где значение attr <i>заканчивается значением</i> val	3
[attr*='val']	<div attr="somevalhere">	Где attr содержит val где угодно	3
[attr ='val']	<div attr="val-sth etc">	Где значение attr равно точно val , или начинается с val и сразу затем - (U + 002D)	2
[attr='val' i]	<div attr="val">	Если attr имеет значение val , игнорируя буквенный корпус val .	4 (2)

Заметки:

1. Значение атрибута может быть окружено кавычками или двойными кавычками. Никакие кавычки вообще не могут работать, но это не действует в соответствии со стандартом CSS и не рекомендуется.

2. Не существует единой интегрированной спецификации CSS4, поскольку она разделяется на отдельные модули. Тем не менее, существуют модули уровня 4. [См. Поддержку браузера](#) .

подробности

`[attribute]`

Выбирает элементы с заданным атрибутом.

```
div[data-color] {  
  color: red;  
}
```

```
<div data-color="red">This will be red</div>  
<div data-color="green">This will be red</div>  
<div data-background="red">This will NOT be red</div>
```

[Живая демонстрация на JSBin](#)

`[attribute="value"]`

Выбирает элементы с заданным атрибутом и значением.

```
div[data-color="red"] {  
  color: red;  
}
```

```
<div data-color="red">This will be red</div>  
<div data-color="green">This will NOT be red</div>  
<div data-color="blue">This will NOT be red</div>
```

[Живая демонстрация на JSBin](#)

`[attribute*="value"]`

Выбирает элементы с заданным атрибутом и значением, где данный атрибут содержит заданное значение где угодно (в качестве подстроки).

```
[class*="foo"] {  
  color: red;  
}
```

```
<div class="foo-123">This will be red</div>  
<div class="foo123">This will be red</div>  
<div class="bar123foo">This will be red</div>  
<div class="barfoo123">This will be red</div>  
<div class="barfo0">This will NOT be red</div>
```


Живая демонстрация на JSBin

```
[attribute~="value"]
```

Выбирает элементы с заданным атрибутом и значением, где данное значение отображается в списке, разделенном пробелами.

```
[class~="color-red"] {  
  color: red;  
}
```

```
<div class="color-red foo-bar the-div">This will be red</div>  
<div class="color-blue foo-bar the-div">This will NOT be red</div>
```

Живая демонстрация на JSBin

```
[attribute^="value"]
```

Выбирает элементы с заданным атрибутом и значением, где данный атрибут начинается со значения.

```
[class^="foo-"] {  
  color: red;  
}
```

```
<div class="foo-123">This will be red</div>  
<div class="foo-234">This will be red</div>  
<div class="bar-123">This will NOT be red</div>
```

Живая демонстрация на JSBin

```
[attribute$="value"]
```

Выбирает элементы с заданным атрибутом и значением, где данный атрибут заканчивается заданным значением.

```
[class$="file"] {  
  color: red;  
}
```

```
<div class="foobar-file">This will be red</div>  
<div class="foobar-file">This will be red</div>  
<div class="foobar-input">This will NOT be red</div>
```

Живая демонстрация на JSBin

```
[attribute|= "value"]
```

Выбирает элементы с заданным атрибутом и значением, где значение атрибута является

точно заданным значением или является точно заданным значением, за которым следует – (U + 002D)

```
[lang]="EN" {  
  color: red;  
}
```

```
<div lang="EN-us">This will be red</div>  
<div lang="EN-gb">This will be red</div>  
<div lang="PT-pt">This will NOT be red</div>
```

Живая демонстрация на JSBin

```
[attribute="value" i]
```

Выбирает элементы с заданным атрибутом и значением, где значение атрибута может быть представлено как `Value`, `VALUE`, `vAlUe` или любая другая возможность `vAlUe` регистра.

```
[lang="EN" i] {  
  color: red;  
}
```

```
<div lang="EN">This will be red</div>  
<div lang="en">This will be red</div>  
<div lang="PT">This will NOT be red</div>
```

Живая демонстрация на JSBin

Специфика селекторов атрибутов

0-1-0

То же, что и селектор классов и псевдоклассы.

```
*[type=checkbox] // 0-1-0
```

Обратите внимание, что это означает, что селектор атрибутов может использоваться для выбора элемента по его идентификатору на более низком уровне специфичности, чем если бы он был выбран с помощью [селектора ID](#): `[id="my-ID"]` нацелен на тот же элемент, что и `#my-ID` но с более низкой специфичностью.

Подробнее см. [Раздел «Синтаксис»](#) .

Комбинаторы

обзор

селектор	Описание
<code>div span</code>	Селектор потомков (все <code> s</code> , которые являются потомками <code><div></code>)
<code>div > span</code>	Селектор детей (все <code> s</code> , которые являются прямым дочерним элементом <code><div></code>)
<code>a ~ span</code>	Общий селектор Sibling (все <code> s</code> , которые являются братьями и сестрами после <code><a></code>)
<code>a + span</code>	Смежный селектор (все <code> s</code> , которые сразу после <code><a></code>)

Примечание. Селектор селектора отображает целевые элементы, которые появляются после них в исходном документе. CSS, по своей природе (он каскадирует), не может ориентироваться на *предыдущие* или *родительские* элементы. Однако, используя свойство `flex order`, [предыдущий селектор для сиблинга может быть смоделирован на визуальных носителях](#) .

Комбинатор потомков: `selector selector`

Комбинатор потомков, представленный по меньшей мере одним пространственным символом (), выбирает элементы, которые являются потомками определенного элемента. Этот комбинатор выбирает **всех** потомков элемента (из дочерних элементов вниз).

```
div p {
  color:red;
}
```

```
<div>
  <p>My text is red</p>
  <section>
    <p>My text is red</p>
  </section>
</div>

<p>My text is not red</p>
```

[Живая демонстрация на JSBin](#)

В приведенном выше примере первые два элемента `<p>` выбираются, поскольку они оба являются потомками `<div>` .

Детский комбинатор: `selector > selector`

Ребенок (`>`) комбинатор используется для выбора элементов , которые являются **детьми**, или **прямыми потомками**, указанного элемента.

```
div > p {
  color:red;
}
```

```
<div>
  <p>My text is red</p>
  <section>
    <p>My text is not red</p>
  </section>
</div>
```

[Живая демонстрация на JSBin](#)

Вышеприведенный CSS выбирает только первый элемент `<p>` , так как он является единственным абзацем, непосредственно выходящим из `<div>` .

Второй элемент `<p>` не выбран, потому что он не является прямым дочерним элементом `<div>` .

Смежный сиблинг-комбинатор: `selector + selector`

Смежный смежный (`+`) комбинатор выбирает элемент-сиблинг, который немедленно следует за определенным элементом.

```
p + p {
  color:red;
}
```

```
<p>My text is not red</p>
<p>My text is red</p>
<p>My text is red</p>
<hr>
<p>My text is not red</p>
```

[Живая демонстрация на JSBin](#)

В приведенном выше примере выбираются только те элементы `<p>` которым *непосредственно предшествует* другой элемент `<p>` .

Общий сиблинг Combinator: `selector ~ selector`

Комбинатор общего родства (~) выбирает *всех* братьев и сестер, следующих за указанным элементом.

```
p ~ p {
  color:red;
}
```

```
<p>My text is not red</p>
<p>My text is red</p>
<hr>
<h1>And now a title</h1>
<p>My text is red</p>
```

[Живая демонстрация на JSBin](#)

В приведенном выше примере выбираются все элементы `<p>` которым *предшествует* другой элемент `<p>` , независимо от того, находятся ли они непосредственно рядом.

Селектора имен классов

Селектор имен классов выбирает все элементы с именем целевого класса. Например, имя класса `.warning` будет выбирать следующий элемент `<div>` :

```
<div class="warning">
  <p>This would be some warning copy.</p>
</div>
```

Вы также можете комбинировать имена классов с целевыми элементами более конкретно. Давайте рассмотрим пример выше, чтобы продемонстрировать более сложный выбор класса.

CSS

```
.important {
  color: orange;
}
.warning {
  color: blue;
}
.warning.important {
  color: red;
}
```

HTML

```
<div class="warning">
  <p>This would be some warning copy.</p>
```

```
</div>

<div class="important warning">
  <p class="important">This is some really important warning copy.</p>
</div>
```

В этом примере все элементы с `.warning` класса будут иметь синий цвет текста, элементы с `.important` класса с имеют оранжевый цвет текста, а также все элементы, которые имеют **КАК** `.important` и `.warning` имени класса будут иметь красный текст цвет.

Обратите внимание, что внутри CSS объявление `.warning.important` не имеет пробелов между двумя именами классов. Это означает, что он будет найти только элементы, которые содержат оба названия класса `warning` и `important` в своем `class` атрибута. Эти имена классов могут быть в любом порядке элемента.

Если бы пространство было включено между двумя классами в объявлении CSS, оно будет выбирать только элементы, которые имеют родительские элементы с `.warning` классов `.warning` и дочерними элементами с именами `.important class`.

Селекторы идентификаторов

Селекторы идентификаторов выбирают элементы DOM с целевым идентификатором. Для выбора элемента по определенному идентификатору в CSS используется префикс `#`.

Например, следующий элемент HTML `div` ...

```
<div id="exampleID">
  <p>Example</p>
</div>
```

... можно выбрать с помощью `#exampleID` в CSS, как показано ниже:

```
#exampleID {
  width: 20px;
}
```

Примечание. Спецификации HTML не позволяют использовать несколько элементов с одинаковым идентификатором

Псевдо-классы

Псевдоклассы - это **ключевые слова**, которые позволяют выбирать на основе информации, которая находится за пределами дерева документов или которая не может быть выражена другими селекторами или комбинаторами. Эта информация может быть связана с определенным состоянием (**состояние** и **динамические** псевдо-классы), в местах (**структурные** и **нацелены** на псевдо-классы), к отрицанию прежнего (**отрицание** псевдо-класса) или на языках (**языки** псевдо-класса). Примеры включают, была ли соблюдена

ссылка (`:visited`), мышь над элементом (`:hover`), флажок установлен (`:checked` отмечен) и т. Д.

Синтаксис

```
selector:pseudo-class {  
  property: value;  
}
```

Список псевдоклассов:

название	Описание
<code>:active</code>	Применяется к любому элементу, который активирован (т.е. нажат) пользователем.
<code>:any</code>	Позволяет создавать наборы связанных селекторов, создавая группы, которые включенные элементы будут соответствовать. Это альтернатива повторению всего селектора.
<code>:target</code>	Выбирает текущий активный элемент <code>#news</code> (нажата на URL-адрес содержащий это имя привязки)
<code>:checked</code>	Применяется к радио, флажкам или элементам опций, которые отмечены или переключается в состояние «включено».
<code>:default</code>	Представляет собой элемент пользовательского интерфейса, который по умолчанию используется для группы аналогичные элементы.
<code>:disabled</code>	Применяется к любому элементу пользовательского интерфейса, находящемуся в отключенном состоянии.
<code>:empty</code>	Применяется к любому элементу, у которого нет детей.
<code>:enabled</code>	Применяется к любому элементу пользовательского интерфейса, который находится в разрешенном состоянии.
<code>:first</code>	Используется в сочетании с правилом <code>@page</code> , это выбирает первую страницу в печатный документ.
<code>:first-child</code>	Представляет собой любой элемент, который является первым

название	Описание
	дочерним элементом его родителя.
<code>:first-of-type</code>	Применяется, когда элемент является первым из выбранного типа элемента внутри его родителя. Это может быть или не быть первым ребенком.
<code>:focus</code>	Применяется к любому элементу, который имеет фокус пользователя. Это может быть дано клавиатура пользователя, события мыши или другие формы ввода.
<code>:focus-within</code>	Может использоваться для выделения целого раздела, когда один элемент внутри него сфокусирован. Он соответствует любому элементу, который соответствует совпадению псевдокласса: фокус или имеет направленный потомок.
<code>:full-screen</code>	Применяется к любому элементу, отображаемому в полноэкранном режиме. Он выбирает весь стек элементов, а не только элемента верхнего уровня.
<code>:hover</code>	Применяется к любому элементу, наводимому указательным устройством пользователя, но не активирован.
<code>:indeterminate</code>	Применяет элементы интерфейса радио или флажка, которые не проверяются ни не контролируются, но находятся в неопределенном состоянии. Это может быть связано с атрибут элемента или DOM-манипуляция.
<code>:in-range</code>	Псевдокласс класса <code>:in-range</code> соответствует, когда элемент имеет его атрибут значения внутри указанных ограничений диапазона для этого элемента. Это позволяет странице дать обратную связь, что значение, определенное в настоящее время использование элемента находится в пределах диапазона.
<code>:invalid</code>	Применяется к элементам <code><input></code> , значения которых недопустимы в соответствии с тип, указанный в атрибуте <code>type=</code> .
<code>:lang</code>	Применяется к любому элементу, у которого есть элемент <code><body></code> обозначенный <code>lang=</code> attribute. Для того чтобы псевдокласс был действительным, он должен содержать действительный двух- или трехбуквенный код языка .

название	Описание
<code>:last-child</code>	Представляет любой элемент, который является последним дочерним элементом его родителя.
<code>:last-of-type</code>	Применяется, когда элемент является последним из выбранного типа элемента внутри его родитель. Это может быть или не быть последним ребенком.
<code>:left</code>	Используется в сочетании с правилом <code>@page</code> , это выбирает все левое страниц в печатном документе.
<code>:link</code>	Применяется к любым ссылкам, которые не были посещены пользователем.
<code>:not()</code>	Применяется ко всем элементам, которые не соответствуют значению, переданному (<code>:not(p)</code> или <code>:not(.class-name)</code> например. Оно должно иметь значение, которое должно быть действителен и может содержать только один селектор. Однако вы можете объединить несколько <code>:not</code> селекторов.
<code>:nth-child</code>	Применяется, когда элемент является n элементом его родителя, где n может быть целым числом, математическим выражением (например, $n+3$) или ключевыми словами <code>odd</code> или <code>even</code> .
<code>:nth-of-type</code>	Применяется, когда элементом является n элемент его родительского элемента тот же тип элемента, где n может быть целым числом, математическим выражение (например, $n+3$) или ключевые слова <code>odd</code> или <code>even</code> .
<code>:only-child</code>	Псевдокласс класса <code>:only-child</code> представляет любой элемент который является единственным ребенком его родителя. Это то же самое, что и <code>:first-child:last-child</code> или <code>:nth-child(1):nth-last-child(1)</code> , но с более низкой специфичностью.
<code>:optional</code>	<code>:optional</code> псевдо-класс CSS представляет любой элемент который не имеет требуемого атрибута, установленного на нем. Это позволяет формы, чтобы легко указывать необязательные поля и соответственно их стиль.
<code>:out-of-range</code>	Псевдокласс класса <code>:out-of-range</code> соответствует, когда элемент имеет

название	Описание
	<p>свой value за пределами указанных ограничений диапазона для этого элемента.</p> <p>Он позволяет странице дать обратную связь, что значение, определяемое в настоящее время с помощью элемент выходит за пределы диапазона. Значение может быть вне диапазона, если оно либо меньше, либо больше максимального и минимального значений.</p>
<code>:placeholder-shown</code>	Экспериментальный. Применяется к любому элементу формы, отображающему текст заполнителя.
<code>:read-only</code>	Применяется к любому элементу, который не редактируется пользователем.
<code>:read-write</code>	Применяется к любому элементу, который редактируется пользователем, например, <code><input></code> .
<code>:right</code>	Используется в сочетании с правилом <code>@page</code> , это выбирает все правильные страницы в печатный документ.
<code>:root</code>	соответствует корневому элементу дерева, представляющего документ.
<code>:scope</code>	CSS-псевдокласс сопоставляет элементы, которые являются ссылочными точка для выбора.
<code>:target</code>	Выбирает текущий активный элемент <code>#news</code> (нажата на URL-адрес содержащий это имя привязки)
<code>:visited</code>	Применяется к любым ссылкам, которые были посещены пользователем.

`:visited` pseudoclass не может быть использован для большинства стилей во многих современных браузерах больше , потому что это дыра в безопасности. См. Эту [ссылку](#) для справки.

Основные селекторы

селектор	Описание
*	Универсальный селектор (все элементы)

селектор	Описание
div	Селектор тегов (все элементы <div>)
.blue	Селектор классов (все элементы с классом blue)
.blue.red	Все элементы с классом blue и red (тип составного селектора)
#headline	Селектор ID (элемент с атрибутом «id», установленным в headline)
:pseudo-class	Все элементы с псевдоклассом
::pseudo-element	Элемент, который соответствует псевдоэлементу
:lang(en)	Элемент, который соответствует: lang-декларации, например
div > p	дочерний селектор

Примечание . Значение идентификатора должно быть уникальным на веб-странице. Нарушением [стандарта HTML](#) является использование значения идентификатора более одного раза в одном и том же дереве документов.

Полный список селекторов можно найти в [спецификации CSS Selectors Level 3](#) .

Как стиль ввода диапазона

HTML

```
<input type="range"></input>
```

CSS

эффект	Псевдоселектор
Большой палец	input [type=range]::-webkit-slider-thumb, input [type=range]::-moz-range-thumb, input [type=range]::-ms-thumb
трек	input [type=range]::-webkit-slider-runnable-track, input [type=range]::-moz-range-track, input [type=range]::-ms-track
OnFocus	input [type=range]:focus
Нижняя часть дорожки	input [type=range]::-moz-range-progress, input [type=range]::-ms-fill-lower (невозможно в браузерах WebKit в настоящее время - требуется JS)

Глобальное логическое значение с флажком: проверено и ~ (общий

комбинированный блок)

С помощью селектора вы можете легко реализовать глобальное доступное логическое значение без использования JavaScript.

Добавить boolean как флажок

В самом начале вашего документа добавьте столько булевых, сколько хотите, с уникальным `id` и `hidden` набором атрибутов:

```
<input type="checkbox" id="sidebarShown" hidden />
<input type="checkbox" id="darkThemeUsed" hidden />

<!-- here begins actual content, for example: -->
<div id="container">
  <div id="sidebar">
    <!-- Menu, Search, ... -->
  </div>

  <!-- Some more content ... -->
</div>

<div id="footer">
  <!-- ... -->
</div>
```

Измените значение boolean

Вы можете переключить логическое значение, добавив `label` с набором атрибутов `for` :

```
<label for="sidebarShown">Show/Hide the sidebar!</label>
```

Доступ к логическому значению с помощью CSS

Обычный селектор (например `.color-red`) указывает свойства по умолчанию. Их можно переопределить, следуя селекторам `true / false` :

```
/* true: */
<checkbox>:checked ~ [sibling of checkbox & parent of target] <target>

/* false: */
<checkbox>:not(:checked) ~ [sibling of checkbox & parent of target] <target>
```

Обратите внимание, что `<checkbox>`, `[sibling ...]` и `<target>` должны быть заменены соответствующими селекторами. `[sibling ...]` может быть конкретным селектором (часто, если вы ленивы) просто `*` или ничего, если цель уже является братом этого флага.

Примерами приведенной выше структуры HTML являются:

```
#sidebarShown:checked ~ #container #sidebar {
  margin-left: 300px;
}

#darkThemeUsed:checked ~ #container,
#darkThemeUsed:checked ~ #footer {
  background: #333;
}
```

В бою

Смотрите [эту скрипту](#) для реализации этих глобальных логических элементов.

CSS3: пример выбора диапазона

```
<style>
input:in-range {
  border: 1px solid blue;
}
</style>

<input type="number" min="10" max="20" value="15">
<p>The border for this value will be blue</p>
```

Псевдокласс класса `:in-range` соответствует, когда элемент имеет свой атрибут значения внутри указанных ограничений диапазона для этого элемента. Он позволяет странице дать обратную связь, что значение, определяемое в настоящее время с помощью элемента, находится в пределах диапазона. [1]

Детский псевдокласс

«Псевдо-класс: `nth-child (a + b)` CSS соответствует элементу, у которого есть дочерние элементы + `b-1` перед ним в дереве документов, для заданного положительного или нулевого значения для `n`» - MDN: [nth-child](#)

псевдо-селектор	1	2	3	4	5	6	7	8	9	10
<code>:first-child</code>	✓									
<code>:nth-child(3)</code>			✓							

псевдо-селектор	1	2	3	4	5	6	7	8	9	10
:nth-child(n+3)			✓	✓	✓	✓	✓	✓	✓	✓
:nth-child(3n)			✓			✓			✓	
:nth-child(3n+1)	✓			✓			✓			✓
:nth-child(-n+3)	✓	✓	✓							
:nth-child(odd)	✓		✓		✓		✓		✓	
:nth-child(even)		✓		✓		✓		✓		✓
:last-child										✓
:nth-last-child(3)								✓		

Выберите элемент, используя свой идентификатор, без высокой специфичности идентификатора

Этот трюк помогает вам выбрать элемент с использованием идентификатора в качестве значения для селектора атрибутов, чтобы избежать высокой специфичности селектора идентификаторов.

HTML:

```
<div id="element">...</div>
```

CSS

```
#element { ... } /* High specificity will override many selectors */
[id="element"] { ... } /* Low specificity, can be overridden easily */
```

A. The: не пример псевдокласса & B.: focus-in CSS псевдокласс

A. Синтаксис представлен выше.

Следующий селектор соответствует всем `<input>` элементам в HTML-документе, которые не отключены и не имеют класса `.example` :

HTML:

```
<form>
  Phone: <input type="tel" class="example">
  E-mail: <input type="email" disabled="disabled">
  Password: <input type="password">
```

```
</form>
```

CSS:

```
input:not([disabled]):not(.example){
  background-color: #ccc;
}
```

Псевдокласс класса `:not()` также будет поддерживать разделенные запятыми селектора в Selectors Level 4:

CSS:

```
input:not([disabled], .example){
  background-color: #ccc;
}
```

[Живая демонстрация на JSBin](#)

См. Синтаксис фона [здесь](#) .

В. Псевдокласс класса CSS.

HTML:

```
<h3>Background is blue if the input is focused .</p>
<div>
  <input type="text">
</div>
```

CSS:

```
div {
  height: 80px;
}
input{
  margin:30px;
}
div:focus-within {
  background-color: #1565C0;
}
```

```
div {
  height: 80px;
}
input{
  margin:30px;
}
div:focus-within {
  background-color: #1565C0;
}
```

Background is blue if the input is focused .



#

:focus-within CSS pseudo-class 📄 - UNOFF

The **:focus-within** pseudo-class matches elements that either themselves match **:focus** or that have descendants which match **:focus**.

Current aligned

Usage relative

Date relative

Show all

IE	Edge *	Firefox	Chrome	Safari	Op
		52	49		
	14	53	58		4
11	15	54	¹ 59	10.1	¹ 4
	16	55	60	11	¹ 4
		56	61	TP	¹ 4
		57	62		

Notes

Known issues (0)

Resources (11)

Feedback

¹ Can be enabled via the "Experimental Web Platform Features" flag

Пример селектора псевдо-класса с единственным дочерним элементом

`:only-child` CSS псевдо-класс представляет собой любой элемент, который является единственным потомком своего родителя.

HTML:

```
<div>
  <p>This paragraph is the only child of the div, it will have the color blue</p>
</div>

<div>
  <p>This paragraph is one of the two children of the div</p>
  <p>This paragraph is one of the two children of its parent</p>
</div>
```

CSS:


```
p:only-child {
  color: blue;
}
```

В приведенном выше примере выбирается элемент `<p>` который является единственным дочерним элементом из его родителя, в этом случае `<div>` .

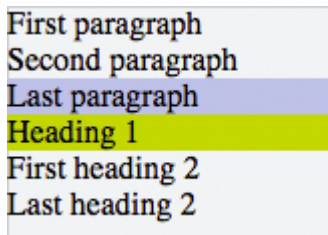
[Живая демонстрация на JSBin](#)

Селектор `last-of-type`

`:last-of-type` выбирает элемент, который является последним дочерним элементом определенного типа его родителя. В приведенном ниже примере `css` выбирает последний абзац и последний заголовок `h1` .

```
p:last-of-type {
  background: #C5CAE9;
}
h1:last-of-type {
  background: #CDDC39;
}
```

```
<div class="container">
  <p>First paragraph</p>
  <p>Second paragraph</p>
  <p>Last paragraph</p>
  <h1>Heading 1</h1>
  <h2>First heading 2</h2>
  <h2>Last heading 2</h2>
</div>
```



First paragraph
Second paragraph
Last paragraph
Heading 1
First heading 2
Last heading 2

[jsFiddle](#)

Прочитайте Селекторы онлайн: <https://riptutorial.com/ru/css/topic/611/селекторы>

глава 40: сетка

Вступление

Макет сетки - это новая и мощная система компоновки CSS, которая позволяет легко разделить содержимое веб-страницы на строки и столбцы.

замечания

[Модуль модуляции сетки CSS 1-го уровня](#), по состоянию на 9 сентября 2016 года, Рекомендацию кандидата W3C. Он считается находящимся на стадии тестирования (<https://www.w3.org/Style/CSS/current-work>).

По состоянию на 3 июля 2017 года браузеры Microsoft Internet Explorer 10 и 11 и Edge поддерживают только более старую версию спецификации с использованием префикса поставщика.

Examples

Основной пример

Имущество	Возможные значения
дисплей	сетка / встроенная сетка

CSS Grid определяется как свойство отображения. Он применяется только к родительскому элементу и его непосредственным детям.

Рассмотрим следующую разметку:

```
<section class="container">
  <div class="item1">item1</div>
  <div class="item2">item2</div>
  <div class="item3">item3</div>
  <div class="item4">item4</div>
</section>
```

Самый простой способ определить структуру разметки выше как сетку - просто установить ее свойство `display` в `grid`:

```
.container {
  display: grid;
}
```

Однако выполнение этого неизбежно приведет к развалу всех дочерних элементов друг над другом. Это связано с тем, что дети в настоящее время не знают, как позиционировать себя в сетке. Но мы можем прямо сказать им.

Сначала нам нужно сообщить элементу `.container` сколько строк и столбцов будет составлять его структуру, и мы можем сделать это, используя свойства `grid-columns` и `grid-rows` (обратите внимание на плюрализацию):

```
.container {
  display: grid;
  grid-columns: 50px 50px 50px;
  grid-rows: 50px 50px;
}
```

Тем не менее, это все равно не помогает нам, потому что нам нужно дать заказ каждому дочернему элементу. Мы можем сделать это, указав значения `grid-row` и `grid-column` которые будут сообщать, где он находится в сетке:

```
.container .item1 {
  grid-column: 1;
  grid-row: 1;
}
.container .item2 {
  grid-column: 2;
  grid-row: 1;
}
.container .item3 {
  grid-column: 1;
  grid-row: 2;
}
.container .item4 {
  grid-column: 2;
  grid-row: 2;
}
```

Предоставляя каждому элементу значение столбца и строки, он идентифицирует порядок элементов в контейнере.

Посмотрите рабочий пример [JSFiddle](#) . Вам нужно будет посмотреть это в IE10, IE11 или Edge, чтобы они работали, поскольку в настоящее время это единственные браузеры, поддерживающие Grid Layout (с префиксом поставщика `-ms-`) или включение флага в Chrome, Opera и Firefox в соответствии с [caniuse](#) для упорядочения протестировать их.

Прочитайте сетка онлайн: <https://riptutorial.com/ru/css/topic/2152/сетка>

глава 41: Спектакль

Examples

Используйте трансформирование и непрозрачность, чтобы избежать компоновки триггеров

Изменение некоторого атрибута CSS приведет к тому, что браузер будет синхронно вычислять стиль и макет, что плохо, если вам нужно анимировать со скоростью 60 кадров в секунду.

НЕ

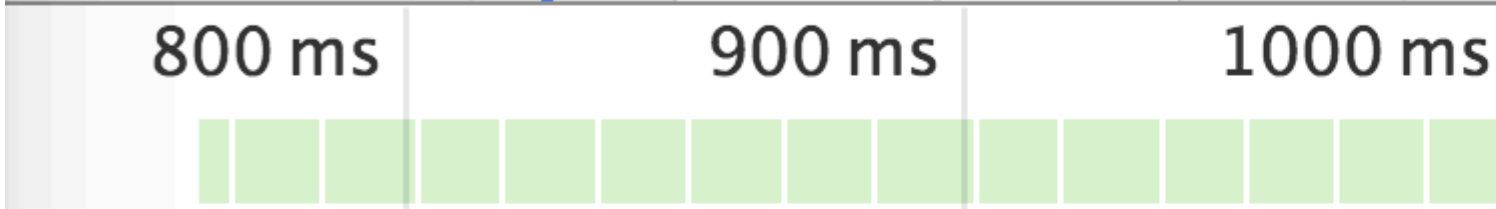
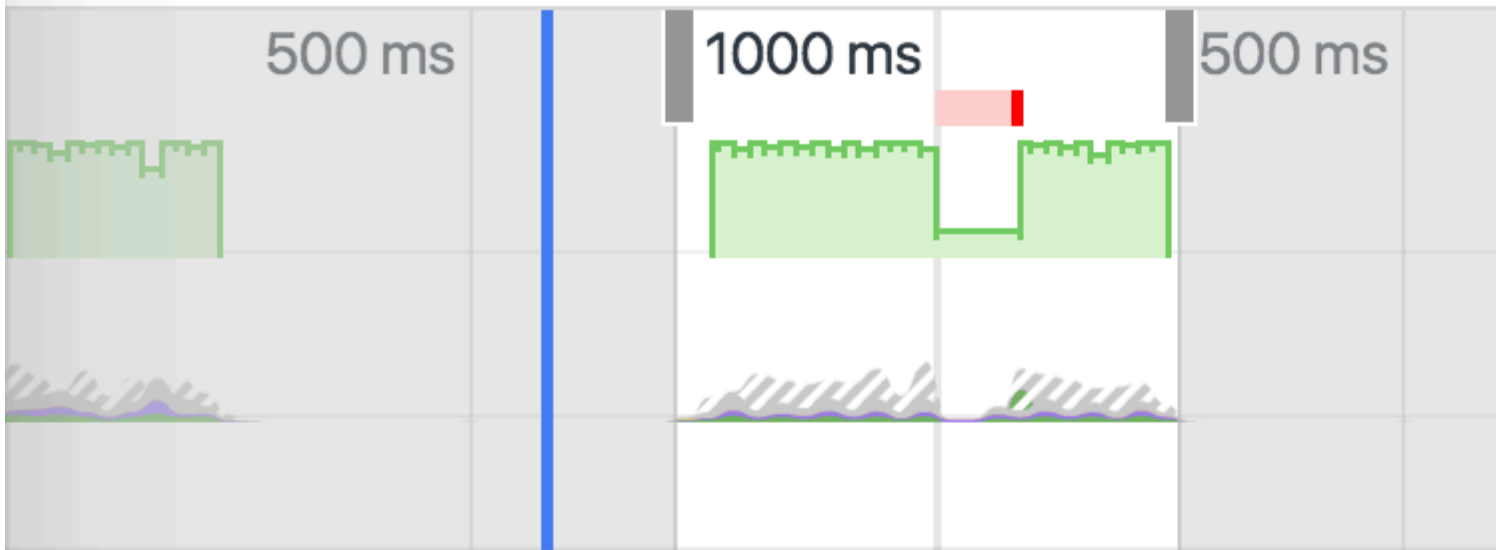
Анимация с `left` и `top` расположением триггера.

```
#box {
  left: 0;
  top: 0;
  transition: left 0.5s, top 0.5s;
  position: absolute;
  width: 50px;
  height: 50px;
  background-color: gray;
}

#box.active {
  left: 100px;
  top: 100px;
}
```

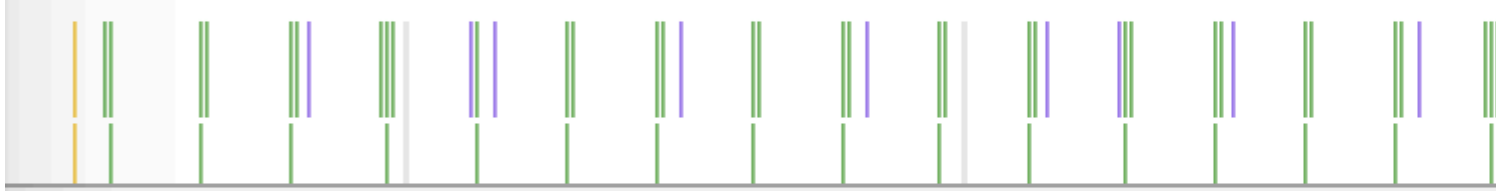
Демонстрация заняла **11,7 м** для рендеринга, **9,8 м** для рисования

● | Capture: Network JS Pro



▶ Interactions

▼ Main



Summary **Bottom-Up** Call Tree Event Log

Group by Category ▼

Self Time		Total Time		Activ
11.7 ms	54.2 %	11.7 ms	54.2 %	▼
4.2 ms	19.5 %	4.2 ms	19.5 %	
3.9 ms	18.2 %	3.9 ms	18.2 %	
1.8 ms	8.3 %	1.8 ms	8.3 %	

глава 42: Стили списка

Синтаксис

- `list-style: list-style-type | list-style-position | list-style-image | начальная | наследовать;`

параметры

Значение	Описание
Список стилей типа	тип маркера списка.
список-стиль-позиция	определяет, где разместить маркер
список-стиль-изображение	указывает тип маркера списка
начальная	присваивает этому свойству значение по умолчанию
унаследовать	наследует это свойство от своего родительского элемента

замечания

Хотя `list-style-type` - это свойство, которое применяется только к элементам списка (обычно ``), оно часто указывается для тега списка (`` или ``). В этом случае элементы списка наследуют свойство.

Examples

Тип пули или нумерация

Конкретно для тегов `` в неупорядоченном списке (``):

```
list-style: disc;           /* A filled circle (default) */
list-style: circle;       /* A hollow circle */
list-style: square;       /* A filled square */
list-style: '-';          /* any string */
```

Конкретно для тегов `` в упорядоченном списке (``):

```
list-style: decimal;       /* Decimal numbers beginning with 1 (default) */
list-style: decimal-leading-zero; /* Decimal numbers padded by initial zeros (01, 02, 03, ... 10)
```

```
*/
list-style: lower-roman;          /* Lowercase roman numerals (i., ii., iii., iv., ...) */
list-style: upper-roman;         /* Uppercase roman numerals (I., II., III., IV., ...) */
list-style-type: lower-greek;    /* Lowercase roman letters (α., β., γ., δ., ...) */
list-style-type: lower-alpha;    /* Lowercase letters (a., b., c., d., ...) */
list-style-type: lower-latin;    /* Lowercase letters (a., b., c., d., ...) */
list-style-type: upper-alpha;    /* Uppercase letters (A., B., C., D., ...) */
list-style-type: upper-latin;    /* Uppercase letters (A., B., C., D., ...) */
```

Неспецифические:

```
list-style: none;                /* No visible list marker */
list-style: inherit;            /* Inherits from parent */
```

Положение пули

Список состоит из `` элементов внутри содержащего элемента (`` или ``). Оба элемента списка и контейнер могут иметь поля и `padding`s, которые влияют на точное положение содержимого элемента списка в документе. Значения по умолчанию для поля и отступов могут отличаться для каждого браузера. Чтобы получить один и тот же макет кросс-браузера, их необходимо установить конкретно.

Каждый элемент списка получает поле «маркер», содержащее маркер маркера. Этот ящик можно разместить внутри или вне поля списка.

```
list-style-position: inside;
```

помещает пулю в элемент `` , подталкивая содержимое вправо по мере необходимости.

```
list-style-position: outside;
```

помещает пулю слева от элемента `` . Если в заполнении содержащего элемента недостаточно места, поле маркера будет расширяться влево, даже если оно упадет со страницы.

Отображение результата `inside` и `outside` позиционирования: [jsfiddle](#)

Удаление пуль / номеров

Иногда список должен просто не отображать никаких маркеров или цифр. В этом случае не забудьте указать `margin` и `padding`.

```
<ul>
  <li>first item</li>
  <li>second item</li>
</ul>
```

CSS

```
ul {  
  list-style-type: none;  
}  
li {  
  margin: 0;  
  padding: 0;  
}
```

Прочитайте Стили списка онлайн: <https://riptutorial.com/ru/css/topic/4215/стили-списка>

глава 43: Стилирование курсора

























Синтаксис

- курсор: авто | default | нет | контекстное меню | Помощь | указатель | прогресс | подождите | ячейка | перекрестье | текст | вертикальный текст | псевдоним | копия | переместить | без капли | не разрешено | e-resize | n-resize | ne-resize | nw-resize | s-resize | se-resize | sw-resize | w-resize | ew-resize | ns-resize | nesw-resize | nwse-resize | col-resize | размер строки | all-scroll | увеличить | Уменьшение | захват | захват;

Examples

Изменение типа курсора

```
cursor: value;
```

	default		n-resize		not-allowed
	crosshair		ne-resize		no-drop
	hand		e-resize		vertical-text
	pointer		se-resize		all-scroll
	Cross browser		s-resize		col-resize
	move		sw-resize		row-resize
	text		w-resize		
	wait		nw-resize		
	help		progress		

Примеры:

Значение	Описание
никто	Для элемента не отображается курсор
авто	По умолчанию. Браузер устанавливает курсор
Помогите	Курсор указывает, что доступна помощь.
Подождите	Курсор указывает, что программа занята
переехать	Курсор указывает, что что-то должно быть перемещено
указатель	Курсор - это указатель и указывает ссылку

указатель событий

Свойство `pointer-events` позволяет контролировать, как элементы HTML реагируют на события `mouse` / `touch`.

```
.disabled {
  pointer-events: none;
}
```

В этом примере,

«none» предотвращает все параметры щелчка, состояния и курсора в указанном HTML-элементе [[1]]

Другими допустимыми значениями для элементов HTML являются:

- `авто`;
- `унаследовать`.

1. <https://css-tricks.com/almanac/properties/p/pointer-events/>

Другие источники:

- <https://developer.mozilla.org/en-US/docs/Web/CSS/pointer-events>
- <https://davidwalsh.name/pointer-events>

каретка цвета

Свойство CSS цвета каретки определяет цвет каретки, видимый индикатор точки вставки в элементе, где текст и другой контент вставляются путем ввода или редактирования пользователем.

HTML

```
<input id="example" />
```

CSS

```
#example {
  caret-color: red;
}
```

Ресурсы:

- <https://developer.mozilla.org/en-US/docs/Web/CSS/caret-color>

Прочитайте [Стилизация курсора онлайн](https://riptutorial.com/ru/css/topic/1742/): <https://riptutorial.com/ru/css/topic/1742/>

[стилирование-курсора](#)

глава 44: Структура и форматирование правила CSS

замечания

Для удобства чтения сохраняйте все декларации с отступом на один уровень от их селектора и закрывающей фигурной скобкой на своей собственной линии. Добавьте одно место после селекторов и двоеточий и всегда помещайте точку с запятой после окончательной декларации.

Хорошо

```
p {  
  color: maroon;  
  font-size: 16px;  
}
```

Плохой

```
p{  
  color: maroon;  
font-size:16px }
```

Один лайнер

Если есть только одно или два объявления, вы можете уйти с этим. Не рекомендуется для большинства случаев. Всегда будьте последовательными, когда это возможно.

```
p { color: maroon; font-size: 16px; }
```

Examples

Правила, селекторы и блоки объявлений

Правило CSS состоит из **селектора** (например, `h1`) и **блока объявлений** (`{ }`).

```
h1 { }
```

Списки свойств

Некоторые свойства могут принимать несколько значений, которые все вместе называются **списком свойств** .

```
/* Two values in this property list */
span {
  text-shadow: yellow 0 0 3px, green 4px 4px 10px;
}

/* Alternate Formatting */
span {
  text-shadow:
    yellow 0 0 3px,
    green 4px 4px 10px;
}
```

Несколько селекторов

Когда вы группируете селектор CSS, вы применяете одни и те же стили к нескольким различным элементам, не повторяя стили в таблице стилей. Используйте запятую для разделения нескольких сгруппированных селекторов.

```
div, p { color: blue }
```

Таким образом, синий цвет применяется ко всем элементам `<div>` и всем элементам `<p>` . Без запятой только `<p>` элементы, являющиеся дочерним элементом `<div>` , будут красными.

Это также относится ко всем типам селекторов.

```
p, .blue, #first, div span{ color : blue }
```

Это правило применяется к:

- `<p>`
- элементы `blue` класса
- элемент с идентификатором `first`
- каждый `` внутри `<div>`

Прочитайте [Структура и форматирование правила CSS онлайн](https://riptutorial.com/ru/css/topic/4313/структура-и-форматирование-правила-css):

<https://riptutorial.com/ru/css/topic/4313/структура-и-форматирование-правила-css>

глава 45: Счетчики

Синтаксис

- counter-set: [<counter-name> <integer>?] + | никто
- counter-reset: [<counter-name> <integer>?] + | никто
- counter-increment: [<counter-name> <integer>?] + | никто
- counter (<counter-name> [, <counter-style>]?)
- счетчики (<counter-name>, <connector-string> [, <counter-style>]?)

параметры

параметр	подробности
Счетчик-имя	Это имя счетчика, который необходимо создать или увеличить или распечатать. Это может быть любое пользовательское имя, которое желает разработчик.
целое число	Это целое число является необязательным значением, которое при представлении рядом с именем счетчика будет представлять начальное значение счетчика (в параметрах <code>counter-set</code> , свойства <code>counter-reset</code>) или значение, по которому счетчик должен увеличиваться (в <code>counter-increment</code>).
НИКТО	Это начальное значение для всех 3 <code>counter-*</code> свойств. Когда это значение используется для <code>counter-increment</code> , это влияет на значение ни одного из счетчиков. Когда это используется для двух других, счетчик не создается.
счетчик стиле	Это указывает стиль, в котором должно отображаться значение счетчика. Он поддерживает все значения, поддерживаемые свойством <code>list-style-type</code> . Если <code>none</code> не используется, значение счетчика вообще не печатается.
Соединитель-строка	Это представляет строку, которая должна быть помещена между значениями двух разных уровней счетчика (например, «.» В «2.1.1»).

замечания

Счетчики не являются новой темой в CSS. Он был частью спецификаций уровня CSS (2), и, следовательно, имеет очень высокую поддержку браузера.

Все браузеры, кроме IE6 и IE7, поддерживают CSS-счетчики.

Examples

Применение стилей римских цифр к выходу счетчика

CSS

```
body {
  counter-reset: item-counter;
}

.item {
  counter-increment: item-counter;
}

.item:before {
  content: counter(item-counter, upper-roman) ". "; /* by specifying the upper-roman as style
the output would be in roman numbers */
}
```

HTML

```
<div class='item'>Item No: 1</div>
<div class='item'>Item No: 2</div>
<div class='item'>Item No: 3</div>
```

В приведенном выше примере вывод счетчика будет отображаться как I, II, III (римские числа) вместо обычных 1, 2, 3, поскольку разработчик явно указал стиль счетчика.

Количество каждого элемента с помощью счетчика CSS

CSS

```
body {
  counter-reset: item-counter; /* create the counter */
}

.item {
  counter-increment: item-counter; /* increment the counter every time an element with class
"item" is encountered */
}

.item-header:before {
  content: counter(item-counter) ". "; /* print the value of the counter before the header and
```

```
append a "." to it */
}

/* just for demo */

.item {
  border: 1px solid;
  height: 100px;
  margin-bottom: 10px;
}
.item-header {
  border-bottom: 1px solid;
  height: 40px;
  line-height: 40px;
  padding: 5px;
}
.item-content {
  padding: 8px;
}
```

HTML

```
<div class='item'>
  <div class='item-header'>Item 1 Header</div>
  <div class='item-content'>Lorem Ipsum Dolor Sit Amet....</div>
</div>
<div class='item'>
  <div class='item-header'>Item 2 Header</div>
  <div class='item-content'>Lorem Ipsum Dolor Sit Amet....</div>
</div>
<div class='item'>
  <div class='item-header'>Item 3 Header</div>
  <div class='item-content'>Lorem Ipsum Dolor Sit Amet....</div>
</div>
```

В приведенном выше примере номера каждого элемента отображаются на странице и добавляется номер элемента перед его заголовком (с использованием свойства `content` элемента `.item-header :before` псевдо). Живая демонстрация этого кода доступна [здесь](#) .

Внедрение многоуровневой нумерации с помощью счетчиков CSS

CSS

```
ul {
  list-style: none;
  counter-reset: list-item-number; /* self nesting counter as name is same for all levels */
}
li {
  counter-increment: list-item-number;
}
li:before {
  content: counters(list-item-number, ".") " "; /* usage of counters() function means value of
```



```
counters at all higher levels are combined before printing */
}
```

HTML

```
<ul>
  <li>Level 1
    <ul>
      <li>Level 1.1
        <ul>
          <li>Level 1.1.1</li>
        </ul>
      </li>
    </ul>
  </li>
  <li>Level 2
    <ul>
      <li>Level 2.1
        <ul>
          <li>Level 2.1.1</li>
          <li>Level 2.1.2</li>
        </ul>
      </li>
    </ul>
  </li>
  <li>Level 3</li>
</ul>
```

Вышеприведенный пример многоуровневой нумерации с использованием счетчиков CSS. Он использует **самонастраивающуюся** концепцию счетчиков. Self nesting - это понятие, в котором, если элемент уже имеет счетчик с заданным именем, но ему необходимо создать другое, он создает его как дочерний элемент существующего счетчика. Здесь второй уровень `ul` уже наследует счетчик `list-item-number` из его родительского `list-item-number` но затем должен создать свой собственный номер `list-item-number` (для его дочерних `list-item-number li`) и поэтому создает номер `list-item-number[1]` (счетчик для второй уровень) и устанавливает его под `list-item-number[0]` (счетчик для первого уровня). Таким образом, он достигает многоуровневой нумерации.

Выход выводится с использованием функции `counters()` вместо функции `counter()` потому что функция `counters()` предназначена для префикса значения всех счетчиков более высокого уровня (родительского) при печати результата.

Прочитайте Счетчики онлайн: <https://riptutorial.com/ru/css/topic/2575/счетчики>

глава 46: таблицы

Синтаксис

- таблица-макет: *авто* | фиксированный;
- пограничный коллапс: *отдельный* | разрушаться;
- border-spacing: <длина> | <длина> <длина>;
- empty-cells: *show* | скрывать;
- сторона заголовка: *верх* | низ;

замечания

Эти свойства применяются как к элементам `<table>` (*), так и к элементам HTML, отображаемым как `display: table` или `display: inline-table`

(*) `<table>`, очевидно, изначально обозначены UA / браузерами как `display: table`

HTML-таблицы семантически допустимы для табличных данных. Не рекомендуется использовать таблицы для макета. Вместо этого используйте CSS.

Examples

Стол-макет

Свойство `table-layout` изменяет алгоритм, используемый для компоновки таблицы.

Ниже пример двух таблиц, которые установлены в `width: 150px`:

First name	Last name	Homeworld
Luke	Skywalker	Tatooine
Leia	Organa	Alderaan

First name	Last name	Homeworld
Luke	Skywalker	Tatooine
Leia	Organa	Alderaan

В таблице слева есть `table-layout: auto` а в правой - `table-layout: fixed`. Первый шире, чем указанная ширина (210px вместо 150px), но содержимое подходит. Последний принимает заданную ширину 150 пикселей, независимо от того, происходит ли переполнение содержимого или нет.

Значение	Описание
<i>авто</i>	Это значение по умолчанию. Он определяет макет таблицы, определяемый содержимым его «ячеек».

Значение	Описание
фиксированный	Это значение устанавливает, что макет таблицы определяется по свойству width, предоставленному в таблицу. Если содержимое ячейки превышает эту ширину, ячейка не будет изменять размер, а вместо этого позволит переполнение содержимого.

границы коллапса

Свойство `border-collapse` определяет, следует ли разделять или объединять границы таблиц.

Ниже приведен пример двух таблиц с разными значениями для свойства `border-collapse` :

First name	Last name	Homeworld
Luke	Skywalker	Tatooine
Leia	Organa	Alderaan

First name	Last name	Homeworld
Luke	Skywalker	Tatooine
Leia	Organa	Alderaan

Таблица слева имеет `border-collapse: separate` а таблица справа имеет `border-collapse: collapse` .

Значение	Описание
отдельный	Это значение по умолчанию. Это делает границы таблицы отдельно друг от друга.
коллапс	Это значение устанавливает, что границы таблицы сливаются вместе, а не различаются.

границы разнос

Свойство `border-spacing` определяет интервал между ячейками. Это не имеет никакого эффекта, если `border-collapse` будет `separate` .

Ниже пример двух таблиц с разными значениями для свойства `border-spacing` :

First name	Last name	Homeworld
Luke	Skywalker	Tatooine
Leia	Organa	Alderaan

First name	Last name	Homeworld
Luke	Skywalker	Tatooine
Leia	Organa	Alderaan

В таблице слева есть `border-spacing: 2px` (по умолчанию), а справа - `border-spacing: 8px` .

Значение	Описание
<длина>	Это поведение по умолчанию, хотя точное значение может различаться между браузерами.
<длина> <длина>	Этот синтаксис позволяет задавать отдельные горизонтальные и вертикальные значения соответственно.

пустые клетки-

Свойство `empty-cells` определяет, должны ли отображаться ячейки без содержимого. Это не имеет никакого эффекта, если `border-collapse` будет `separate`.

Ниже пример с двумя таблицами с разными значениями, установленными в свойстве `empty-cells`:

First name	Last name	Homeworld
Luke		Tatooine
Leia	Organa	

First name	Last name	Homeworld
Luke		Tatooine
Leia	Organa	

В таблице слева есть `empty-cells: show` а справа - `empty-cells: hide`. Первый отображает пустые ячейки, а последний - нет.

Значение	Описание
<i>шоу</i>	Это значение по умолчанию. Он отображает ячейки, даже если они пусты.
скрывать	Это значение скрывает ячейку вообще, если в ячейке нет содержимого.

Дополнительная информация:

- <https://www.w3.org/TR/CSS21/tables.html#empty-cells>
- <https://developer.mozilla.org/en-US/docs/Web/CSS/empty-cells>
- <http://codepen.io/SitePoint/pen/yfhtq>
- <https://css-tricks.com/almanac/properties/e/empty-cells/>

Надпись на стороне

Свойство `caption-side` определяет вертикальное позиционирование элемента `<caption>` внутри таблицы. Это не имеет никакого эффекта, если такой элемент не существует.

Ниже пример с двумя таблицами с разными значениями, установленными для свойства `caption-side`:

Star Wars figures		
First name	Last name	Homeworld
Luke	Skywalker	Tatooine
Leia	Organa	Alderaan

First name	Last name	Homeworld
Luke	Skywalker	Tatooine
Leia	Organa	Alderaan

Star Wars figures

Таблица слева имеет `caption-side: top` а справа `caption-side: bottom`.

Значение	Описание
Топ	Это значение по умолчанию. Он помещает надпись над столом.
низ	Это значение помещает подпись ниже таблицы.

Прочитайте таблицы онлайн: <https://riptutorial.com/ru/css/topic/1074/таблицы>

глава 47: Типографика

Синтаксис

- font: [*font-style*] [*font-variant*] [*font-weight*] *font-size* [/ *line-height*] *font-family* ;
- font-style: *font-style*
- font-variant: *font-variant*
- font-weight: *font-weight* ;
- font-size: *font-size* ;
- line-height: *line-height* ;
- font-family: *font-family* ;
- цвет: *цвет* ;
- quotes: *none* | *string* | *initial* | *inherit* ;
- font-stretch: *font-stretch* ;
- text-align: *text-align* ;
- text-indent: *length* | *initial* | *inherit* ;
- text-overflow: *clip* | *ellipsis* | *string* | *initial* | *inherit* ;
- text-transform: *none* | *заглавные буквы* | *прописные буквы* | *строчные* | *начальные* | *наследования* ;
- text-shadow: *h-shadow* *v-shadow* *blur-radius* *color* | *none* | *initial* | *inherit* ;
- font-size-adjust: *число* | *none* | *initial* | *inherit* ;
- стретч-стрит: *сверхконденсат* | *конденсированный* | *конденсированный* | *полуконденсированный* | *нормальный* | *полурасширенный* | *расширенный* | *экстра-расширенный* | *ультра-расширенный* | *initial* | *inherit* ;
- Дефис: *нет* | *руководство* | *авто* ;
- tab-size: *number* | *length* | *initial* | *inherit* ;
- расстояние между буквами: *normal* | *length* | *initial* | *inherit* ;
- word-spacing: *normal* | *length* | *initial* | *inherit* ;

параметры

параметр	подробности
стиль шрифта	<i>italics</i> или <i>oblique</i>
вариант шрифта	<i>normal</i> или <i>small-caps</i>
начертание шрифта	<i>normal</i> , <i>bold</i> или числовой от 100 до 900.
размер шрифта	Размер шрифта, заданный в % , px , em или любом другом действительном измерении CSS

параметр	подробности
<i>высота линии</i>	Высота линии, заданная в % , px , em или любом другом действительном измерении CSS
<i>семейство шрифтов</i>	Это определение имени семьи.
<i>цвет</i>	Любое допустимое цветовое представление CSS , такое как red , #00FF00 , hsl(240, 100%, 50%) и т. Д.
<i>Шрифт растяжения</i>	Использовать ли закрытое или расширенное лицо из шрифта. Допустимые значения являются normal , ultra-condensed , extra-condensed , condensed , semi-condensed , semi-expanded , expanded , extra-expanded ИЛИ ultra-expanded
<i>выравнивания текста</i>	start , end , left , right , center , justify , match-parent
<i>текст-отделка</i>	none , underline , overline , line-through , initial , inherit ;

замечания

- Свойство `text-shadow` не поддерживается версиями Internet Explorer менее 10.

Examples

Размер шрифта

HTML:

```
<div id="element-one">Hello I am some text.</div>
<div id="element-two">Hello I am some smaller text.</div>
```

CSS:

```
#element-one {
  font-size: 30px;
}

#element-two {
  font-size: 10px;
}
```

Текст внутри `#element-one` будет размером 30px , а текст в `#element-two` будет размером 10px .

Шрифт

С синтаксисом:

```
element {
  font: [font-style] [font-variant] [font-weight] [font-size/line-height] [font-family];
}
```

Вы можете иметь все стили, связанные с `font` в одном объявлении с сокращением `font` . Просто используйте свойство `font` и поместите свои значения в правильном порядке.

Например, чтобы все элементы `p` выделены жирным шрифтом с размером шрифта 20px и с использованием Arial в качестве семейства шрифтов, вы обычно его кодировали следующим образом:

```
p {
  font-weight: bold;
  font-size: 20px;
  font-family: Arial, sans-serif;
}
```

Однако с сокращением шрифта он может быть сжат следующим образом:

```
p {
  font: bold 20px Arial, sans-serif;
}
```

Обратите внимание : поскольку `font-style` , `font-variant` , `font-weight` и `line-height` являются необязательными, три из них пропущены в этом примере. Важно отметить, что использование ярлыка **сбрасывает** другие атрибуты, не указанные. Другим важным моментом является то, что двумя необходимыми атрибутами для ярлыка шрифта для работы являются `font-size` `font-family` . Если они не включены, ярлык игнорируется.

Начальное значение для каждого из свойств:

- `font-style: normal;`
- `font-variant: normal;`
- `font-weight: normal;`
- `font-stretch: normal;`
- `font-size: medium;`
- `line-height: normal;`
- `font-family` - **ЗАВИСИТ ОТ ПОЛЬЗОВАТЕЛЯ**

Шрифты

```
font-family: 'Segoe UI', Tahoma, sans-serif;
```

Браузер попытается применить шрифт «Segoe UI» к символам в элементах, на которые

ссылается вышеуказанное свойство. Если этот шрифт недоступен или шрифт не содержит глифа для требуемого символа, браузер возвращается к Таhома и, при необходимости, шрифту sans-serif на компьютере пользователя. Обратите внимание, что любые имена шрифтов с более чем одним словом, такие как «Segoe UI», должны иметь одинарные или двойные кавычки вокруг них.

```
font-family: Consolas, 'Courier New', monospace;
```

Браузер попытается применить грань шрифта «Консола» к символам в элементах, на которые ссылается вышеуказанное свойство. Если этот шрифт недоступен или шрифт не содержит глифа для требуемого символа, браузер вернется к «Courier New» и, при необходимости, любому моноширинному шрифту на компьютере пользователя.

Межбуквенное расстояние

```
h2 {  
  /* adds a 1px space horizontally between each letter;  
   also known as tracking */  
  letter-spacing: 1px;  
}
```

Свойство letter-spacing используется для указания пробела между символами в тексте.

! Расстояние между буквами также поддерживает отрицательные значения:

```
p {  
  letter-spacing: -1px;  
}
```

Ресурсы: <https://developer.mozilla.org/en-US/docs/Web/CSS/letter-spacing>

Текстовое преобразование

Свойство text-transform позволяет вам изменять капитализацию текста. Допустимыми значениями являются: uppercase , uppercase capitalize , lowercase , initial , inherit И none

CSS:

```
.example1 {  
  text-transform: uppercase;  
}  
.example2 {  
  text-transform: capitalize;  
}  
.example3 {  
  text-transform: lowercase;  
}
```

HTML

```
<p class="example1">
  all letters in uppercase <!-- "ALL LETTERS IN UPPERCASE" -->
</p>
<p class="example2">
  all letters in capitalize <!-- "All Letters In Capitalize (Sentence Case)" -->
</p>
<p class="example3">
  all letters in lowercase <!-- "all letters in lowercase" -->
</p>
```

Отступ текста

```
p {
  text-indent: 50px;
}
```

Свойство `text-indent` указывает, сколько текста горизонтального пространства должно быть перемещено до начала первой строки текстового содержимого элемента.

Ресурсы:

- [Отступы только в первой строке текста в абзаце?](#)
- <https://www.w3.org/TR/CSS21/text.html#propdef-text-indent>
- <https://developer.mozilla.org/en-US/docs/Web/CSS/text-indent>

Текстовое оформление

Свойство `text-decoration` используется для установки или удаления украшений из текста.

```
h1 { text-decoration: none; }
h2 { text-decoration: overline; }
h3 { text-decoration: line-through; }
h4 { text-decoration: underline; }
```

`text-decoration` можно использовать в сочетании с `text-decoration-style` и `text-decoration-color` как сокращенное свойство:

```
.title { text-decoration: underline dotted blue; }
```

Это сокращенная версия

```
.title {
  text-decoration-style: dotted;
  text-decoration-line: underline;
  text-decoration-color: blue;
}
```

Следует отметить, что следующие свойства поддерживаются только в Firefox

- текст-отделка цвета
- текст-отделка линия
- текст-отделка-стиль
- текст-отделка-скип

Переполнение текста

Свойство `text-overflow` имеет дело с тем, как переполненный контент должен сигнализироваться пользователям. В этом примере `ellipsis` представляет сжатый текст.

```
.text {
  overflow: hidden;
  text-overflow: ellipsis;
}
```

К сожалению, `text-overflow: ellipsis` работает только в одной строке текста. Невозможно поддерживать эллипсис на последней строке стандартного CSS, но это может быть достигнуто при использовании нестандартных реализаций гибких пакетов только для веб-китов.

```
.giveMeEllipsis {
  overflow: hidden;
  text-overflow: ellipsis;
  display: -webkit-box;
  -webkit-box-orient: vertical;
  -webkit-line-clamp: N; /* number of lines to show */
  line-height: X;      /* fallback */
  max-height: X*N;     /* fallback */
}
```

Пример (открыть в Chrome или Safari):

<http://jsfiddle.net/csYjC/1131/>

Ресурсы:

<https://www.w3.org/TR/2012/WD-css3-ui-20120117/#text-overflow0>

Межстрочный интервал

Свойство `word-spacing` определяет поведение интервалов между тегами и словами.

Возможные значения

- положительная или отрицательная *длина* (с использованием `em` `px` `vh` `cm` и т. д.) или *процент* (с использованием `%`)
- ключевое слово `normal` использует интервал слова шрифта по умолчанию
- ключевое слово `inherit` принимает значение из родительского элемента

CSS

```
.normal    { word-spacing: normal; }  
.narrow    { word-spacing: -3px; }  
.extensive { word-spacing: 10px; }
```

HTML

```
<p>  
  <span class="normal">This is an example, showing the effect of "word-spacing".</span><br>  
  <span class="narrow">This is an example, showing the effect of "word-spacing".</span><br>  
  <span class="extensive">This is an example, showing the effect of "word-spacing".</span><br>  
</p>
```

Интернет-Демо

[Попробуй сам](#)

Дальнейшее чтение:

- [слово-интервал - MDN](#)
- [word-spacing - w3.org](#)

Направление текста

```
div {  
  direction: ltr; /* Default, text read from left-to-right */  
}  
.ex {  
  direction: rtl; /* text read from right-to-left */  
}  
.horizontal-tb {  
  writing-mode: horizontal-tb; /* Default, text read from left-to-right and top-to-bottom. */  
}  
.vertical-rtl {  
  writing-mode: vertical-rl; /* text read from right-to-left and top-to-bottom */  
}  
.vertical-ltr {  
  writing-mode: vertical-rl; /* text read from left-to-right and top to bottom */  
}
```

Свойство `direction` используется для изменения горизонтального направления текста элемента.

Синтаксис: `direction: ltr | rtl | initial | inherit;`

Свойство `writing-mode` изменяет выравнивание текста, чтобы его можно было читать сверху вниз или слева направо, в зависимости от языка.

Синтаксис: `direction: horizontal-tb | vertical-rl | vertical-lr;`

Вариант шрифта

Атрибуты:

нормальный

Атрибут по умолчанию для шрифтов.

капитель

Устанавливает каждую букву в верхний регистр, **но** делает строчные буквы (из исходного текста) меньше по размеру, чем буквы, которые изначально заглавные.

CSS:

```
.smallcaps{
  font-variant: small-caps;
}
```

HTML:

```
<p class="smallcaps">
  Documentation about CSS Fonts
  <br>
  aNd ExAmPlE
</p>
```

Output:

DOCUMENTATION ABOUT CSS FONTS
AND EXAMPLE

Примечание. Свойство `font-variant` является сокращением для свойств: `font-variant-caps`, `font-variant-numeric`, `font-variant-alternates`, `font-variant-ligatures` и `font-variant-east-asian`.

Цитаты

Свойство `quotes` используется для настройки кавычек открытия и закрытия тега `<q>` .

```
q {
  quotes: "«" "»";
}
```

Текстовая тень

Чтобы добавить тени в текст, используйте свойство `text-shadow` . Синтаксис выглядит следующим образом:

```
text-shadow: horizontal-offset vertical-offset blur color;
```

Тень без радиуса размытия

```
h1 {  
  text-shadow: 2px 2px #0000FF;  
}
```

Это создает эффект синей тени вокруг заголовка

Тень с радиусом размытия

Чтобы добавить эффект размытия, добавьте параметр `blur radius`

```
h1 {  
  text-shadow: 2px 2px 10px #0000FF;  
}
```

Несколько теней

Чтобы выделить элемент несколькими тенями, разделите их запятыми

```
h1 {  
  text-shadow: 0 0 3px #FF0000, 0 0 5px #0000FF;  
}
```

Прочитайте Типография онлайн: <https://riptutorial.com/ru/css/topic/427/типография>

глава 48: Управление макетами

Синтаксис

- display: none | inline | блок | список | inline-list-item | встроенный блок | встроенный стол | стол | настольная ячейка | стол-стол | table-column-group | table-footer-group | table-header-group | стол-строка | table-row-group | flex | inline-flex | сетка | встроенная сетка | запуск | рубин | рубиновая основа | рубиновый текст | рубиновый-базовый контейнер | ruby-text-container | содержание;

параметры

Значение	эффект
none	Скройте элемент и не позволяйте ему занимать пространство.
block	Блочный элемент, занимающий 100% доступной ширины, перерыв после элемента.
inline	Встроенный элемент не занимает ширины, не разбивается на элемент.
inline-block	Принимая специальные свойства как от встроенных, так и от блочных элементов, нет разрыва, но может иметь ширину.
inline-flex	Отображает элемент как гибкий контейнер inline-уровня.
inline-table	Элемент отображается в виде таблицы встроенного уровня.
grid	Ведет себя как элемент блока и выдает его содержимое в соответствии с моделью сетки.
flex	Ведет себя как элемент блока и выдает его содержимое в соответствии с моделью flexbox.
inherit	Наследуйте значение от родительского элемента.
initial	Сбросьте значение до значения по умолчанию, взятого из поведения, описанного в спецификациях HTML, или из таблицы стилей по умолчанию для браузера / пользователя.
table	Ведет себя подобно элементу <code>table</code> HTML.
table-cell	Пусть элемент ведет себя как элемент <code><td></code>

Значение	эффект
<code>table-column</code>	Пусть элемент ведет себя как элемент <code><col></code>
<code>table-row</code>	Пусть элемент ведет себя как элемент <code><tr></code>
<code>list-item</code>	Пусть элемент ведет себя как элемент <code></code> .

Examples

Свойство отображения

Свойство CSS `display` является основополагающим для управления макетом и потоком HTML-документа. Большинство элементов имеют `display` умолчанию значение как `block` и `inline` (хотя некоторые элементы имеют другие значения по умолчанию).

В соответствии

`inline` элемент занимает ровно столько ширины по мере необходимости. Он складывается горизонтально с другими элементами одного и того же типа и может не содержать других не-встроенных элементов.

```
<span>This is some <b>bolded</b> text!</span>
```

This is some **bolded** text!

Как показано выше, два `inline` элемента, `` и `` , являются строчными (отсюда и название) и не прерывают поток текста.

блок

Элемент `block` занимает максимальную доступную ширину своего «родительского элемента». Он начинается с новой строки и, в отличие от `inline` элементов, не ограничивает тип элементов, которые он может содержать.

```
<div>Hello world!</div><div>This is an example!</div>
```

Hello world!
This is an example!

Элемент `div` по умолчанию является блочным уровнем, и, как показано выше, два элемента `block` вертикально сложены и, в отличие от `inline` элементов, поток текста прерывается.

Встроенный блок

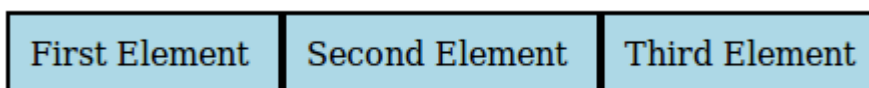
Значение `inline-block` дает нам лучшее из обоих миров: оно смешивает элемент с потоком текста, позволяя нам использовать `padding`, `margin`, `height` и аналогичные свойства, которые не оказывают заметного влияния на `inline` элементы.

Элементы с этим отображаемым значением действуют так, как если бы они были обычным текстом и в результате влияли правила, управляющие потоком текста, такие как `text-align`. По умолчанию они также сокращаются до наименьшего размера, чтобы разместить их содержимое.

```
<!--Inline: unordered list-->
<style>
li {
  display : inline;
  background : lightblue;
  padding:10px;

  border-width:2px;
  border-color:black;
  border-style:solid;
}
</style>

<ul>
<li>First Element </li>
<li>Second Element </li>
<li>Third Element </li>
</ul>
```



```
<!--block: unordered list-->
<style>
li {
  display : block;
  background : lightblue;
  padding:10px;

  border-width:2px;
  border-color:black;
  border-style:solid;
}
</style>

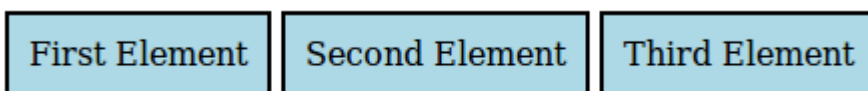
<ul>
<li>First Element </li>
<li>Second Element </li>
<li>Third Element </li>
</ul>
```



```
<!--Inline-block: unordered list-->
<style>
li {
  display : inline-block;
  background : lightblue;
  padding:10px;

  border-width:2px;
  border-color:black;
  border-style:solid;
  }
</style>

<ul>
<li>First Element </li>
<li>Second Element </li>
<li>Third Element </li>
</ul>
```



НИКТО

Элемент, которому присвоено значение `none` для его свойства отображения, вообще не будет отображаться.

Например, давайте создадим `div`-элемент с идентификатором `myDiv` :

```
<div id="myDiv"></div>
```

Теперь это можно пометить как не отображаемое с помощью следующего правила CSS:

```
#myDiv {
  display: none;
}
```

Когда элемент установлен для `display:none`; браузер игнорирует каждое другое свойство макета для этого конкретного элемента (как для `position` и для `float`). Для этого элемента не будет показано поле, и его существование в `html` не влияет на положение следующих

элементов.

Обратите внимание, что это отличается от установки свойства `visibility` на `hidden`. Настройка `visibility: hidden;` для элемента не будет отображаться элемент на странице, но элемент все равно займет пространство в процессе рендеринга, как если бы он был виден. Это повлияет на то, как на странице отображаются следующие элементы.

Значение `none` для свойства отображения обычно используется вместе с JavaScript, чтобы отображать или скрывать элементы по своему усмотрению, устраняя необходимость фактического удаления и повторного создания.

Чтобы получить старую структуру таблицы, используя `div`

Это стандартная структура таблицы HTML

```
<style>
  table {
    width: 100%;
  }
</style>

<table>
  <tr>
    <td>
      I'm a table
    </td>
  </tr>
</table>
```

Вы можете сделать такую же реализацию, как это

```
<style>
  .table-div {
    display: table;
  }
  .table-row-div {
    display: table-row;
  }
  .table-cell-div {
    display: table-cell;
  }
</style>

<div class="table-div">
  <div class="table-row-div">
    <div class="table-cell-div">
      I behave like a table now
    </div>
  </div>
</div>
```

Прочитайте Управление макетами онлайн: <https://riptutorial.com/ru/css/topic/1473/управление-макетами>

глава 49: Установка и размещение объектов

замечания

Internet Explorer не поддерживает свойства `object-fit` и место `object-position`.

Examples

Объект посадки

Свойство **object-fit** определяет, как элемент будет вписываться в поле с установленной высотой и шириной. Обычно применяется к изображению или видео, Object-fit принимает следующие пять значений:

FILL

```
object-fit:fill;
```

original image



object-fit: fill;



Заполнить растягивает изображение, чтобы оно соответствовало содержимому, независимо от исходного соотношения сторон изображения.

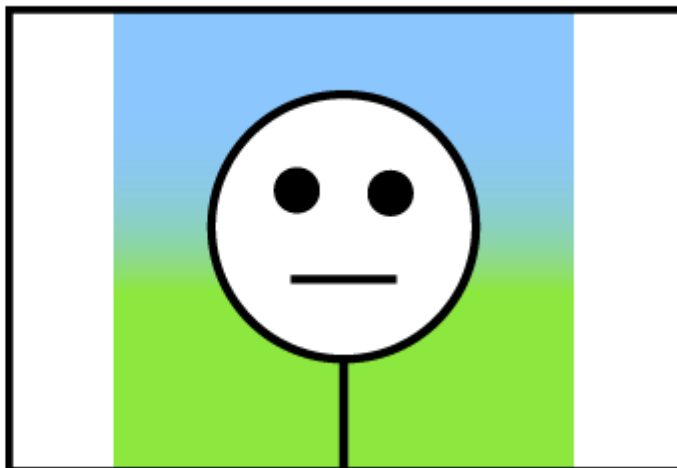
СОДЕРЖАТЬ

```
object-fit:contain;
```

original image



object-fit: contain;



Contain соответствует изображению на высоте или ширине окна при сохранении соотношения сторон изображения.

ПОКРЫТИЕ

```
object-fit: cover;
```

original image



object-fit: cover;



Покрытие заполняет всю коробку изображением. Соотношение сторон изображения сохраняется, но изображение обрезается до размеров окна.

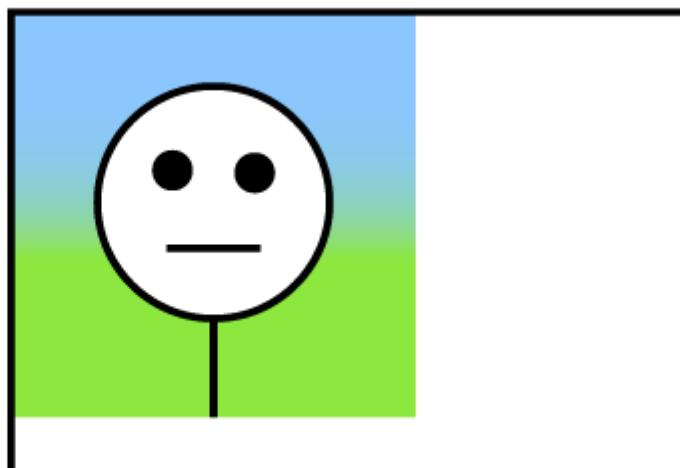
НИКТО

```
object-fit: none;
```

original image



object-fit: none;



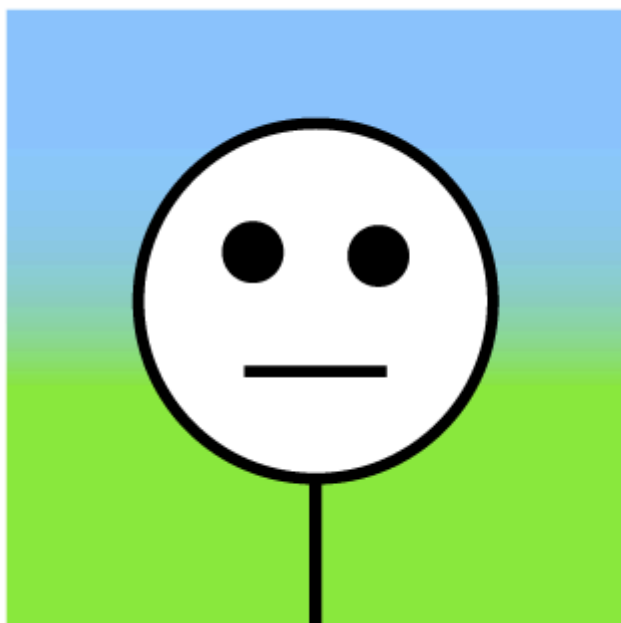
None игнорирует размер поля и не изменяется.

УМЕНЬШАТЬ

```
object-fit: scale-down;
```

Масштабирование уменьшает или уменьшает размер объекта как `none` или как `contain`. Он отображает любой вариант с меньшим размером изображения.

original image



object-fit: scale-down;



Прочитайте [Установка и размещение объектов онлайн](https://riptutorial.com/ru/css/topic/5520/установка-и-размещение-объектов):

<https://riptutorial.com/ru/css/topic/5520/установка-и-размещение-объектов>

глава 50: Фон

Вступление

С помощью CSS вы можете устанавливать цвета, градиенты и изображения в качестве фона элемента.

Можно указать различные комбинации изображений, цветов и градиентов, а также настроить размер, положение и повторение (среди прочего) этих.

Синтаксис

- `background-color`: `color` | прозрачный | начальная | наследовать;
- `background-image`: `url` | нет | начальная | наследовать;
- `background-position`: значение;
- `background-size`: `<bg-size>` [`<bg-size>`]
- `<bg-size>`: авто | длина | обложка | содeржат | начальная | наследовать;
- `background-repeat`: `repeat` | `repeat-x` | `repeat-y` | `no-repeat` | начальная | наследовать;
- `background-origin`: `padding-box` | пограничный ящик | контент-бокс | начальная | наследовать;
- `background-clip`: `border-box` | прокладка | контент-бокс | начальная | наследовать;
- фон-приложение: прокрутка | исправлено | местные | начальная | наследовать;
- `background`: `bg-color` `bg-image` `position` / `bg-size` `bg-repeat` `bg-origin` `bg-clip` `bg-attachment` `initial` | наследовать;

замечания

- Градиенты CSS3 не будут работать в версиях Internet Explorer менее 10.

Examples

Фоновый цвет

Свойство `background-color` задает цвет фона элемента с использованием значения цвета или с помощью ключевых слов, например, `transparent`, `inherit` или `initial`.

- **прозрачный**, указывает, что цвет фона должен быть прозрачным. Это значение по умолчанию.
- **inherit**, наследует это свойство от своего родительского элемента.
- **initial**, присваивает этому свойству значение по умолчанию.

Это может быть применено ко всем элементам и `::first-letter` **псевдо-элементам** `::first-letter / ::first-line`.

Цвета в CSS можно указать **разными способами**.

Названия цветов

CSS

```
div {
  background-color: red; /* red */
}
```

HTML

```
<div>This will have a red background</div>
```

- Приведенный выше пример является одним из нескольких способов, которыми CSS должен представлять один цвет.
-

Кодовые обозначения

Hex-код используется для обозначения компонентов RGB цвета в шестнадцатеричной нотации base-16. `#ff0000`, например, ярко-красный, где красный компонент цвета составляет 256 бит (ff), а соответствующие зеленые и синие части цвета - 0 (00).

Если оба значения в каждом из трех пар RGB (R, G и B) совпадают, тогда код цвета можно укоротить на три символа (первая цифра каждого спаривания). `#ff0000` можно сократить до `#f00`, а `#ffffff` можно сократить до `#fff`.

Hex-нотация нечувствительна к регистру.

```
body {
  background-color: #de1205; /* red */
}

.main {
  background-color: #00f; /* blue */
}
```

RGB / RGBA

Другой способ объявить цвет - использовать RGB или RGBA.

RGB означает красный, зеленый и синий и требует трех отдельных значений от 0 до 255,

расположенных между скобками, которые соответствуют десятичным значениям цвета соответственно красного, зеленого и синего.

RGBA позволяет добавить дополнительный альфа-параметр между 0.0 и 1.0 для определения непрозрачности.

```
header {
  background-color: rgb(0, 0, 0); /* black */
}

footer {
  background-color: rgba(0, 0, 0, 0.5); /* black with 50% opacity */
}
```

HSL / HSLa

Другой способ объявить цвет - использовать HSL или HSLa и похож на RGB и RGBA.

HSL обозначает оттенок, насыщенность и легкость, а также часто называется HLS:

- Оттенок - это градус на цветовом колесе (от 0 до 360).
- Насыщенность - это процент от 0% до 100%.
- Легкость также составляет от 0% до 100%.

HSLa позволяет добавить дополнительный альфа-параметр между 0.0 и 1.0 для определения непрозрачности.

```
li a {
  background-color: hsl(120, 100%, 50%); /* green */
}

#p1 {
  background-color: hsla(120, 100%, 50%, .3); /* green with 30% opacity */
}
```

Взаимодействие с фоновым изображением

Следующие утверждения эквивалентны:

```
body {
  background: red;
  background-image: url(partiallytransparentimage.png);
}

body {
  background-color: red;
  background-image: url(partiallytransparentimage.png);
}
```

```
body {
  background-image: url(partiallytransparentimage.png);
  background-color: red;
}

body {
  background: red url(partiallytransparentimage.png);
}
```

Все они приведут к отображению красного цвета под изображением, где части изображения прозрачны или изображение не отображается (возможно, в результате `background-repeat`).

Обратите внимание, что следующее не эквивалентно:

```
body {
  background-image: url(partiallytransparentimage.png);
  background: red;
}
```

Здесь значение `background` переопределяет `background-image` .

Для получения дополнительной информации о `background` собственности см [фона стенографии](#)

Изображение на заднем плане

Свойство `background-image` используется для указания фонового изображения для всех сопоставленных элементов. По умолчанию это изображение выложено для покрытия всего элемента, за исключением поля.

```
.myClass {
  background-image: url('/path/to/image.jpg');
}
```

Чтобы использовать несколько изображений в качестве `background-image` , определите разделяемый запятыми `url()`

```
.myClass {
  background-image: url('/path/to/image.jpg'),
                  url('/path/to/image2.jpg');
}
```

Изображения будут складываться в соответствии с их порядком с первым объявленным изображением поверх других и так далее.

Значение	Результат
<code>url('/path/to/image.jpg')</code>	Укажите путь (ы) фонового изображения или ресурс

Значение	Результат
	изображения, указанный в схеме URI данных (апострофы могут быть опущены), отдельные кратные по запятой
none	Нет фонового изображения
initial	Значение по умолчанию
inherit	Наследовать значение родителя

Больше CSS для фонового изображения

Эти следующие атрибуты очень полезны и почти существенны.

```
background-size:    xpx ypx | x% y%;
background-repeat: no-repeat | repeat | repeat-x | repeat-y;
background-position: left offset (px/%) right offset (px/%) | center center | left top | right bottom;
```

Фоновые градиенты

Градиенты - это новые типы изображений, добавленные в CSS3. В качестве изображения градиенты задаются с использованием свойства `background-image` или сокращением `background`.

Существует два типа функций градиента: линейный и радиальный. Каждый тип имеет не повторяющийся вариант и повторяющийся вариант:

- `linear-gradient()`
- `repeating-linear-gradient()`
- `radial-gradient()`
- `repeating-radial-gradient()`

линейный градиент ()

`linear-gradient` имеет следующий синтаксис

```
background: linear-gradient( <direction>?, <color-stop-1>, <color-stop-2>, ...);
```

Значение	Имея в виду
<direction>	Может быть аргументом, как <code>to top</code> , <code>to bottom</code> , <code>to right</code> или <code>to left</code> ; или угол как <code>0deg</code> , <code>90deg</code> Угол начинается от вершины и вращается по часовой стрелке. Может указываться в градусах , градусах , радах или оборотах . Если опустить, градиент течет сверху вниз

Значение	Имея в виду
<code><color-stop-list></code>	Список цветов, необязательно следуя каждому из них на процент или длину , чтобы отобразить его. Например, <code>yellow 10% , rgba(0,0,0,.5) 40px , #fff 100% ...</code>

Например, это создает линейный градиент, начинающийся справа и переход от красного к синему

```
.linear-gradient {
  background: linear-gradient(to left, red, blue); /* you can also use 270deg */
}
```

Вы можете создать `diagonal` градиент, объявив горизонтальную и вертикальную начальную позицию.

```
.diagonal-linear-gradient {
  background: linear-gradient(to left top, red, yellow 10%);
}
```

В градиенте можно указать любое количество остановок цвета, разделив их запятыми. В следующих примерах будет создан градиент с 8 остановками цвета

```
.linear-gradient-rainbow {
  background: linear-gradient(to left, red, orange, yellow, green, blue, indigo, violet)
}
```

радиально-градиент ()

```
.radial-gradient-simple {
  background: radial-gradient(red, blue);
}

.radial-gradient {
  background: radial-gradient(circle farthest-corner at top left, red, blue);
}
```

Значение	Имея в виду
<code>circle</code>	Форма градиента. Значения - это <code>circle</code> или <code>ellipse</code> , по умолчанию - <code>ellipse</code> .
<code>farthest-corner</code>	Ключевые слова, описывающие, насколько велика конечная форма. Значения <code>closest-side</code> , <code>farthest-side</code> , <code>closest-corner</code> , <code>farthest-corner</code>
<code>top left</code>	Устанавливает положение центра градиента так же, как и <code>background-position</code> .

Повторяющиеся градиенты

Повторяющиеся функции градиента принимают те же аргументы, что и приведенные выше примеры, но нарисуйте градиент на фоне элемента.

```
.bullseye {
  background: repeating-radial-gradient(red, red 10%, white 10%, white 20%);
}
.warning {
  background: repeating-linear-gradient(-45deg, yellow, yellow 10%, black 10%, black 20% );
}
```

Значение	Имея в виду
-45deg	Угловой блок . Угол начинается от вершины и вращается по часовой стрелке. Может указываться в градусах , градусах , радах или оборотах .
to left	Направление градиента, по умолчанию - <code>to bottom</code> . Синтаксис: <code>to [y-axis(top OR bottom)] [x-axis(left OR right)]</code> т.е. <code>to top right</code>
yellow 10%	Цвет, необязательно сопровождаемый процентом или длиной, чтобы отобразить его. Повторяется два или более раз.

Обратите внимание, что вместо названий цветов могут использоваться цветовые коды HEX, RGB, RGBA, HSL и HSLa. Названия цветов использовались для иллюстрации. Также обратите внимание, что синтаксис радиального градиента намного сложнее линейного градиента, и здесь показана упрощенная версия. Полное описание и спецификации см. В [Документах MDN](#)

Сокращение фона

Свойство `background` можно использовать для установки одного или нескольких связанных с фоном свойств:

Значение	Описание	CSS Ver.
<code>background-image</code>	Фоновое изображение для использования	1+
<code>background-color</code>	Цвет фона для применения	1+
<code>background-position</code>	Положение фонового изображения	1+
<code>background-size</code>	Размер фонового изображения	3+

Значение	Описание	CSS Ver.
background-repeat	Как повторить фоновое изображение	1+
background-origin	Как фон установлен (игнорируется , когда background-attachment является fixed)	3+
background-clip	Как фон окрашивается по отношению к content-box , border-box или padding-box	3+
background-attachment	Как работает фоновое изображение, независимо от того, прокручивается ли оно вместе с его содержащим блоком или имеет фиксированное положение в области просмотра	1+
initial	Устанавливает свойство для значения по умолчанию	3+
inherit	Наследует значение свойства от родительского	2+

Порядок значений не имеет значения, и каждое значение является необязательным

Синтаксис

Синтаксис фоновой сокращенной декларации:

```
background: [<background-image>] [<background-color>] [<background-position>]/[<background-size>] [<background-repeat>] [<background-origin>] [<background-clip>] [<background-attachment>] [<initial|inherit>];
```

Примеры

```
background: red;
```

Просто установите background-color с red значением.

```
background: border-box red;
```

Установка background-clip в рамку-рамки и background-color в красный.

```
background: no-repeat center url("somepng.jpg");
```

Устанавливает background-repeat не повторяющийся, background-origin в центр и background-image изображение.

```
background: url('pattern.png') green;
```

В этом примере `background-color` элемента будет установлен на `green` с помощью `pattern.png`, если он доступен, наложен на цвет, повторяясь так часто, как необходимо, чтобы заполнить элемент. Если `pattern.png` включает любую прозрачность, тогда `green` цвет будет виден за ним.

```
background: #000000 url("picture.png") top left / 600px auto no-repeat;
```

В этом примере у нас есть черный фон с изображением «picture.png» сверху, изображение не повторяется ни одной из осей и расположено в верхнем левом углу. Положение / после позиции должно включать в себя размер фонового изображения, которое в этом случае устанавливается как ширина `600px` и автоматически для высоты. Этот пример может хорошо работать с изображением функции, которое может исчезать в сплошной цвет.

ПРИМЕЧАНИЕ. Использование свойства shorthand background сбрасывает все ранее заданные значения свойств фона, даже если значение не задано. Если вы хотите изменить ранее заданное значение свойства фона, вместо этого используйте свойство longhand.

Фоновая позиция

Свойство `background-position` используется для указания начальной позиции для фонового изображения или градиента

```
.myClass {  
  background-image: url('path/to/image.jpg');  
  background-position: 50% 50%;  
}
```

Позиция устанавливается с использованием координаты **X** и **Y** и устанавливается с использованием любого из единиц, используемых в CSS.

Единица измерения	Описание
<i>значение %</i> <i>значение %</i>	Процент для горизонтального смещения относительно (<i>ширина области фонового позиционирования - ширина фонового изображения</i>) . Процент для вертикального смещения относительно (<i>высота области фонового позиционирования - высота фонового изображения</i>) Размер изображения - размер, заданный размером <code>background-size</code> .
<i>значение px</i> <i>значение px</i>	Смещение фонового изображения по длине, заданной в пикселях относительно левого верхнего угла области фонового позиционирования

Единицы в CSS могут быть указаны различными способами (см. [Здесь](#)).

Свойства позиции фокуса

В дополнение к сокращенному свойству выше, можно также использовать свойства `background-position-x` и `background-position-y` . Это позволяет вам управлять положениями `x` или `y` отдельно.

ПРИМЕЧАНИЕ. Это поддерживается во всех браузерах, кроме Firefox (версии 31-48) [2](#) . Firefox 49, который будет выпущен в сентябре 2016 года, *будет* поддерживать эти свойства. До тех пор [в этом ответе «Переполнение стека» есть взлома Firefox.](#)

Справочное приложение

Свойство `background-attachment` устанавливает, фиксировано ли фоновое изображение или прокручивается с остальной частью страницы.

```
body {
  background-image: url('img.jpg');
  background-attachment: fixed;
}
```

Значение	Описание
свиток	Фон прокручивается вместе с элементом. Это значение по умолчанию.
фиксированный	Фон фиксируется относительно окна просмотра.
местный	Фон прокручивается вместе с содержимым элемента.
начальная	Устанавливает это свойство по умолчанию.
унаследовать	Наследует это свойство от его родительского элемента.

Примеры

background-attachment: прокрутка

Поведение по умолчанию, когда тело прокручивается, прокручивает фон:


```
body {
  background-image: url('image.jpg');
  background-attachment: scroll;
}
```

background-attachment: исправлено

Фоновое изображение будет зафиксировано и не будет перемещаться при прокрутке тела:

```
body {
  background-image: url('image.jpg');
  background-attachment: fixed;
}
```

background-attachment: local

Фоновое изображение div будет прокручиваться при прокрутке содержимого div.

```
div {
  background-image: url('image.jpg');
  background-attachment: local;
}
```

Повторение фона

Свойство `background-repeat` устанавливает, если / как фоновое изображение будет повторяться.

По умолчанию фоновое изображение повторяется как по вертикали, так и по горизонтали.

```
div {
  background-image: url("img.jpg");
  background-repeat: repeat-y;
}
```

Вот как `background-repeat: repeat-y` **ВЫГЛЯДИТ** так:



Цвет фона с непрозрачностью

Если вы установите `opacity` элемента, это повлияет на все его дочерние элементы. Чтобы установить непрозрачность только на фоне элемента, вам придется использовать цвета RGBA. Следующий пример будет иметь черный фон с 0,6 непрозрачностью.

```
/* Fallback for web browsers that don't support RGBA */
background-color: rgb(0, 0, 0);

/* RGBA with 0.6 opacity */
background-color: rgba(0, 0, 0, 0.6);

/* For IE 5.5 - 7*/
filter: progid:DXImageTransform.Microsoft.gradient(startColorstr=#99000000,
endColorstr=#99000000);

/* For IE 8*/
-ms-filter: "progid:DXImageTransform.Microsoft.gradient(startColorstr=#99000000,
endColorstr=#99000000)";
```

Несколько фоновых изображений

В CSS3 мы можем складывать несколько фона в один и тот же элемент.

```
#mydiv {
  background-image: url(img_1.png), /* top image */
                  url(img_2.png), /* middle image */
                  url(img_3.png); /* bottom image */
  background-position: right bottom,
                     left top,
                     right top;
  background-repeat: no-repeat,
                   repeat,
                   no-repeat;
}
```

Изображения будут складываться поверх друг друга с первым фоном сверху и последним фоном сзади. `img_1` будет сверху, `img_2` и `img_3` будут внизу.

Мы также можем использовать свойство стенографии фона для этого:

```
#mydiv {
  background: url(img_1.png) right bottom no-repeat,
             url(img_2.png) left top repeat,
             url(img_3.png) right top no-repeat;
}
```

Мы также можем складывать изображения и градиенты:

```
#mydiv {
  background: url(image.png) right bottom no-repeat,
             linear-gradient(to bottom, #fff 0%, #000 100%);
}
```

- [демонстрация](#)

Свойство `background-origin`

Свойство `background-origin` указывает, где находится фоновое изображение.

Примечание. Если для свойства `background-attachment` установлено значение `fixed`, это свойство не действует.

Значение по умолчанию: `padding-box`

Возможные значения:

- `padding-box` - позиция относительно поля заполнения
- `border-box` - позиция относительно поля границы
- `content-box` - позиция относительно поля содержимого
- `initial`
- `inherit`

CSS

```

.example {
  width: 300px;
  border: 20px solid black;
  padding: 50px;
  background: url(https://static.pexels.com/photos/6440/magazines-desk-work-workspace-
medium.jpg);
  background-repeat: no-repeat;
}

.example1 {}

.example2 { background-origin: border-box; }

.example3 { background-origin: content-box; }

```

HTML

```

<p>No background-origin (padding-box is default):</p>

<div class="example example1">
  <h2>Lorem Ipsum Dolor</h2>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod
tincidunt ut laoreet dolore magna aliquam erat volutpat.</p>
  <p>Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis
nisl ut aliquip ex ea commodo consequat.</p>
</div>

<p>background-origin: border-box:</p>
<div class="example example2">
  <h2>Lorem Ipsum Dolor</h2>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod
tincidunt ut laoreet dolore magna aliquam erat volutpat.</p>
  <p>Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis
nisl ut aliquip ex ea commodo consequat.</p>
</div>

<p>background-origin: content-box:</p>
<div class="example example3">
  <h2>Lorem Ipsum Dolor</h2>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod
tincidunt ut laoreet dolore magna aliquam erat volutpat.</p>
  <p>Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis
nisl ut aliquip ex ea commodo consequat.</p>
</div>

```

Результат:

No background-origin (padding-box is default):



background-origin: border-box:



background-origin: content-box:



указывает область рисования фона.

Значение по умолчанию: `border-box`

Ценности

- `border-box` - значение по умолчанию. Это позволяет фону полностью простираться до внешнего края границы элемента.
- `padding-box` зажимает фон на внешнем краю прокладки элемента и не позволяет ему растягиваться на границе;
- `content-box` фиксирует фон на краю окна содержимого.
- `inherit` применяет настройку родителя к выбранному элементу.

CSS

```
.example {
  width: 300px;
  border: 20px solid black;
  padding: 50px;
  background: url(https://static.pexels.com/photos/6440/magazines-desk-work-workspace-medium.jpg);
  background-repeat: no-repeat;
}

.example1 {}

.example2 { background-origin: border-box; }

.example3 { background-origin: content-box; }
```

HTML

```
<p>No background-origin (padding-box is default):</p>

<div class="example example1">
  <h2>Lorem Ipsum Dolor</h2>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.</p>
  <p>Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.</p>
</div>

<p>background-origin: border-box:</p>
<div class="example example2">
  <h2>Lorem Ipsum Dolor</h2>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.</p>
  <p>Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.</p>
</div>

<p>background-origin: content-box:</p>
<div class="example example3">
  <h2>Lorem Ipsum Dolor</h2>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod
```

```
tincidunt ut laoreet dolore magna aliquam erat volutpat.</p>
```

```
<p>Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis  
nisl ut aliquip ex ea commodo consequat.</p>  
</div>
```

Размер фона

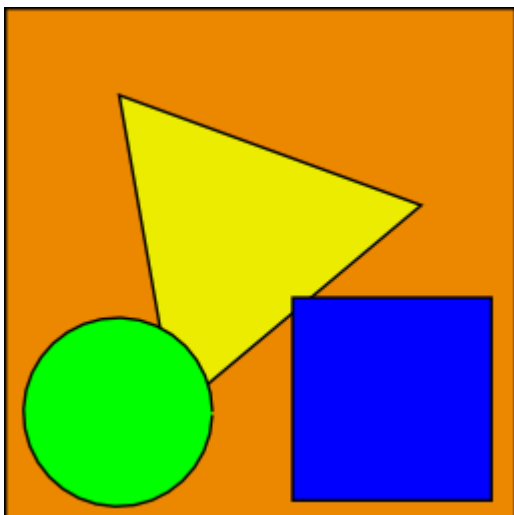
Общий обзор

Свойство `background-size` позволяет управлять масштабированием `background-image`. Он принимает до двух значений, которые определяют масштаб / размер результирующего изображения в вертикальном и горизонтальном направлениях. Если свойство отсутствует, считается ее `auto` как в `width` и `height`.

`auto` будет поддерживать соотношение сторон изображения, если оно может быть определено. Высота необязательна и может считаться `auto`. Таким образом, на изображении 256 px x 256 px все последующие настройки `background-size` давали бы изображение с высотой и шириной 50 px:

```
background-size: 50px;  
background-size: 50px auto; /* same as above */  
background-size: auto 50px;  
background-size: 50px 50px;
```

Итак, если мы начали со следующего изображения (имеющего упомянутый размер 256 px x 256 px),



мы получим 50 пикселей x 50 пикселей на экране пользователя, который находится на фоне нашего элемента:



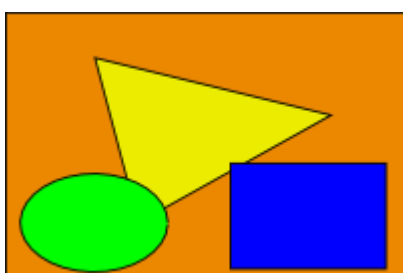
Можно также использовать процентные значения для масштабирования изображения по отношению к элементу. Следующий пример привел бы к обратному изображению 200 px × 133 px:

```
#withbackground {
  background-image: url(to/some/background.png);

  background-size: 100% 66%;

  width: 200px;
  height: 200px;

  padding: 0;
  margin: 0;
}
```



Поведение зависит от `background-origin`.

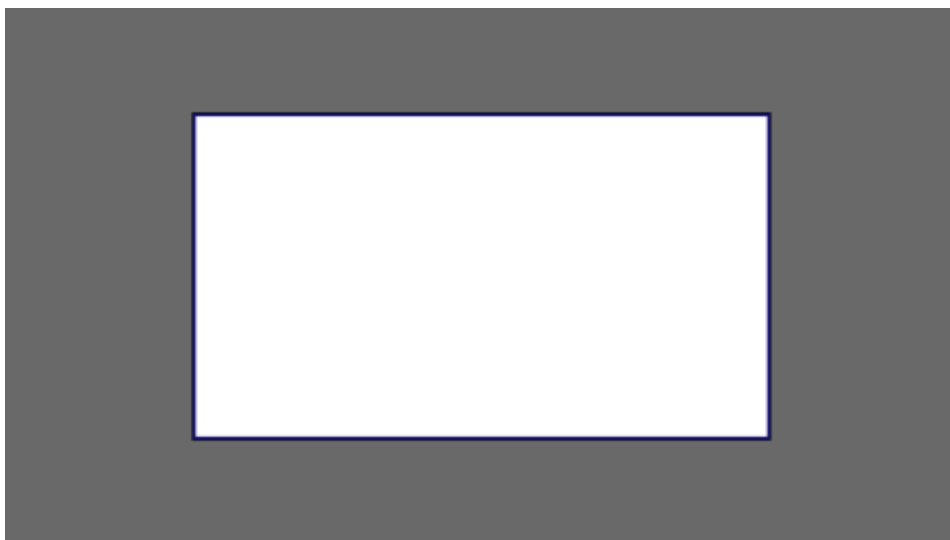
Сохранение соотношения сторон

Последний пример в предыдущем разделе потерял первоначальное соотношение сторон. Круг попал в эллипс, квадрат в прямоугольник, треугольник в другой треугольник.

Продолжительность или процентный подход недостаточно гибки, чтобы поддерживать соотношение сторон в любое время. `auto` не помогает, поскольку вы можете не знать, какое измерение вашего элемента будет больше. Однако, чтобы полностью покрывать определенные области с изображением (и правильным соотношением сторон) или содержать изображение с правильным соотношением сторон полностью в фоновом режиме, значения, `contain` и `cover` обеспечивают дополнительную функциональность.

Яйца для `contain` и `cover`

Извините за плохой каламбур, но мы собираемся использовать [фотографию дня для Biswarup Ganguly](#) для демонстрации. Допустим, что это ваш экран, а серая область находится за пределами видимого экрана. Для демонстрации мы собираемся принять соотношение 16 × 9.



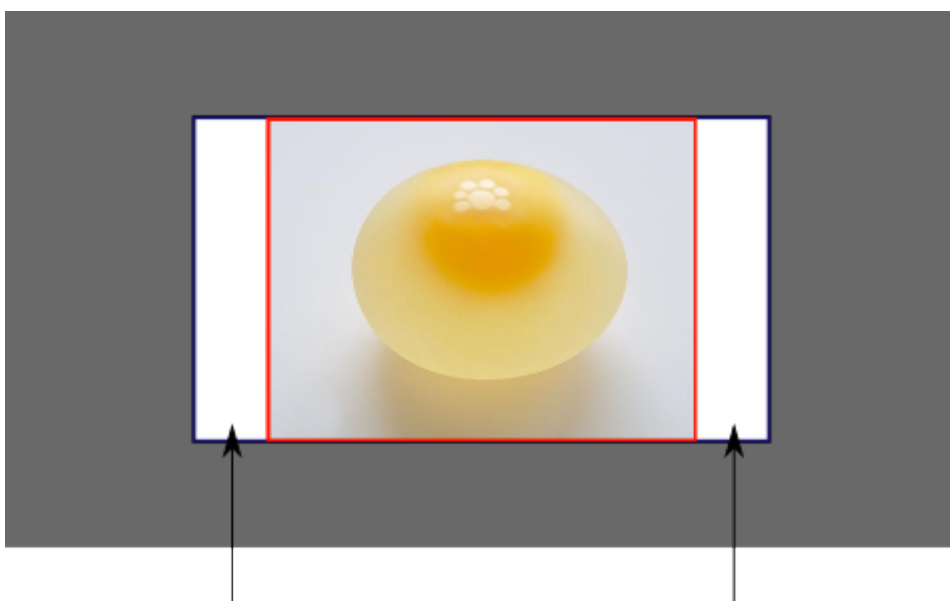
Мы хотим использовать вышеупомянутую картину дня в качестве фона. Однако по какой-то причине мы обрезали изображение до 4x3. Мы могли бы установить свойство `background-size` для определенной фиксированной длины, но мы сосредоточимся на `contain` и `cover`. Обратите внимание, что я также предполагаю, что мы не изменяли ширину и / или высоту `body`.

`contain`

```
contain
```

Масштабируйте изображение, сохраняя при этом его внутреннее соотношение сторон (если оно имеется) до самого большого размера, чтобы его ширина и высота могли вписываться в область фонового позиционирования.

Это гарантирует, что фоновое изображение всегда полностью содержится в области фонового позиционирования, однако в этом случае может быть пустое пространство, заполненное `background-color`:

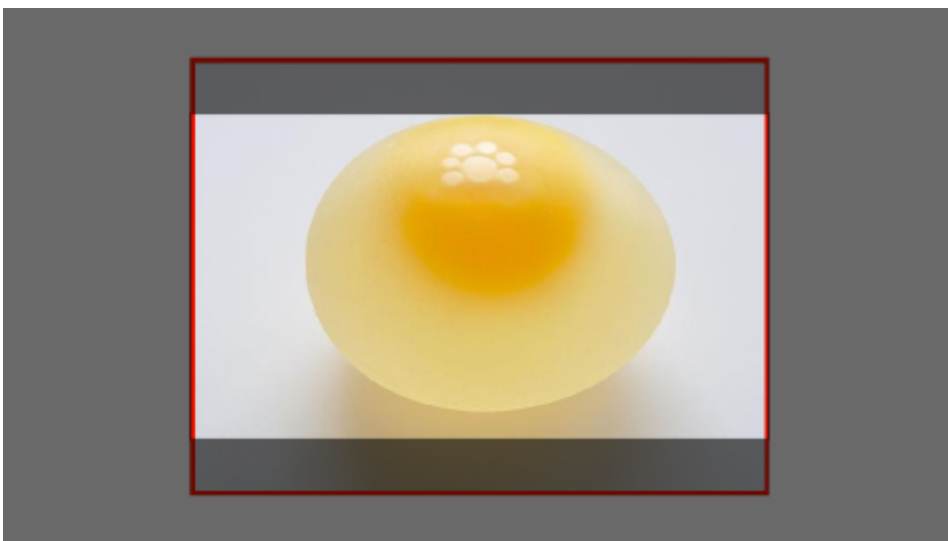


cover

cover

Масштабируйте изображение, сохраняя при этом его внутреннее соотношение сторон (если оно есть) до наименьшего размера, так что его ширина и высота могут полностью перекрывать область фонового позиционирования.

Это гарантирует, что фоновое изображение покрывает все. Не будет видимого `background-color`, но в зависимости от соотношения экрана большая часть вашего изображения может быть отключена:



Демонстрация с фактическим кодом

```
div > div {
  background-image: url(http://i.stack.imgur.com/r5CAq.jpg);
  background-repeat: no-repeat;
  background-position: center center;
  background-color: #ccc;
  border: 1px solid;
  width: 20em;
  height: 10em;
}
div.contain {
  background-size: contain;
}
div.cover {
  background-size: cover;
}
/*****
Additional styles for the explanation boxes
*****/

div > div {
  margin: 0 1ex 1ex 0;
  float: left;
}
div + div {
```

```

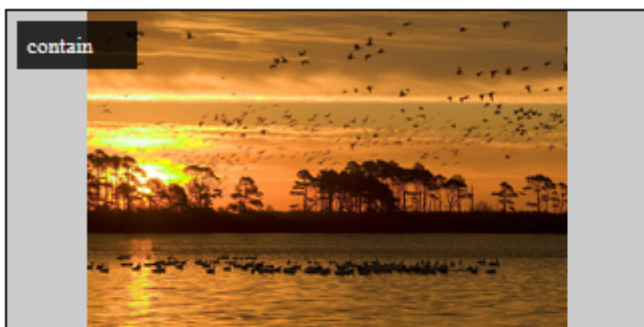
clear: both;
border-top: 1px dashed silver;
padding-top: 1ex;
}
div > div::after {
background-color: #000;
color: #fefefe;
margin: 1ex;
padding: 1ex;
opacity: 0.8;
display: block;
width: 10ex;
font-size: 0.7em;
content: attr(class);
}

```

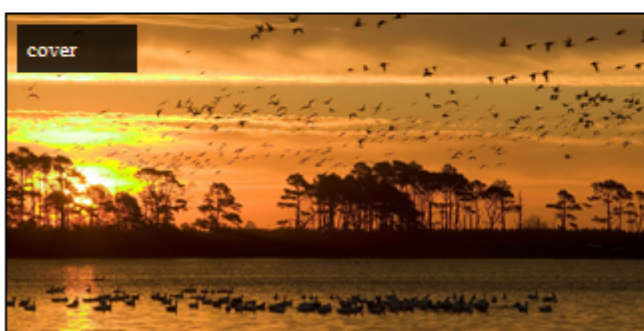
```

<div>
  <div class="contain"></div>
  <p>Note the grey background. The image does not cover the whole region, but it's fully
  <em>contained</em>.
  </p>
</div>
<div>
  <div class="cover"></div>
  <p>Note the ducks/geese at the bottom of the image. Most of the water is cut, as well as a
  part of the sky. You don't see the complete image anymore, but neither do you see any
  background color; the image <em>covers</em> all of the <code>&lt;div&gt;</code>.
  </p>
</div>

```



Note the **grey background**. The image does not cover the whole region, but it's fully *contained*.



Note the ducks/geese at the bottom of the image. Most of the water is cut, as well as a part of the sky. You don't see the complete image anymore, but neither do you see any background color; the image *covers* all of the `<div>`.

Свойство background-blend-mode

```

.my-div {
width: 300px;
height: 200px;
background-size: 100%;
background-repeat: no-repeat;
}

```

```
background-image: linear-gradient(to right, black 0%,white 100%),
url('https://static.pexels.com/photos/54624/strawberry-fruit-red-sweet-54624-medium.jpeg');
background-blend-mode:saturation;
}
```

```
<div class="my-div">Lorem ipsum</div>
```

См. Результат здесь: <https://jsfiddle.net/MadalinaTn/y69d28Lb/>

CSS Синтаксис: background-blend-mode: normal | умножить | экран | наложение | темные | свет | цвет-додж | насыщение | цвет | светимость;

Прочитайте Фон онлайн: <https://riptutorial.com/ru/css/topic/296/фон>

глава 51: Формы для поплавок

Синтаксис

- форма снаружи: нет | [`<basic-shape>` || `<shape-box>`] | `<Изображение>`
- `margin-size`: `<длина>` | `<Процент>`
- `shape-image-threshold`: `<number>`

параметры

параметр	подробности
никто	Значение <code>none</code> означает, что область с плавающей точкой (область, которая используется для обертывания содержимого вокруг элемента <code>float</code>) не изменяется. Это значение по умолчанию / начальное значение.
Основная форма,	Относится к одному из <code>inset()</code> , <code>circle()</code> , <code>ellipse()</code> или <code>polygon()</code> . Используя одну из этих функций и ее значения, определяется форма.
форма ящик	Относится к одному из <code>margin-box</code> , <code>border-box</code> , <code>padding-box</code> <code>content-box</code> . Когда предоставляется только <code><shape-box></code> (без <code><basic-shape></code>), это поле является формой. Когда он используется вместе с <code><basic-shape></code> , это действует как ссылочный блок.
образ	Когда изображение предоставляется как значение, форма вычисляется на основе альфа-канала указанного изображения.

замечания

Поддержка браузера для модуля CSS Shapes очень ограничена на данный момент времени.

Он поддерживается в Chrome v37+ и Opera 24+ без префиксов браузера / поставщика. Safari поддерживает его с v7.1+, но с префиксом `-webkit-`.

Он еще не поддерживается в IE, Edge и Firefox.

Examples

Форма снаружи с основной формой - круг ()

С помощью свойства CSS `shape-outside` можно определить значения формы для области с плавающей точкой, чтобы встроенный контент обертывался вокруг формы, а не в поле `float`.

CSS

```
img:nth-of-type(1) {
  shape-outside: circle(80px at 50% 50%);
  float: left;
  width: 200px;
}
img:nth-of-type(2) {
  shape-outside: circle(80px at 50% 50%);
  float: right;
  width: 200px;
}
p {
  text-align: center;
  line-height: 30px; /* purely for demo */
}
```

HTML

```

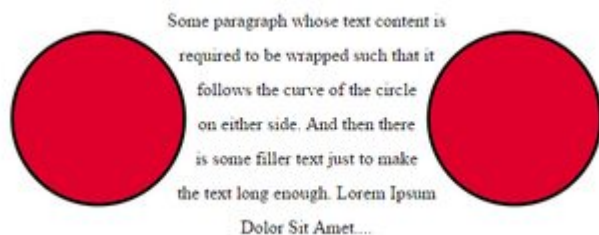

<p>Some paragraph whose text content is required to be wrapped such that it follows the curve
of the circle on either side. And then there is some filler text just to make the text long
enough. Lorem Ipsum Dolor Sit Amet...</p>
```

В приведенном выше примере оба изображения на самом деле являются квадратными изображениями, а когда текст помещен без свойства `shape-outside`, он не будет течь по кругу с обеих сторон. Он будет обтекать только содержащую коробку изображения. С `shape-outside` находящейся `shape-outside` с плавающей точкой, переопределяется как *круг*, и содержимое создается для обтекания этого *воображаемого круга*, который создается с использованием `shape-outside`.

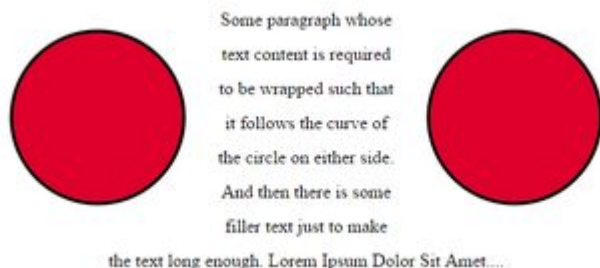
Воображаемая окружность, которая используется, чтобы повторно определить область поплавок представляет собой окружность с радиусом 80 пикселей, проведенным из центра, средней точки отсчета поля изображения.

Ниже приведены несколько скриншотов, чтобы проиллюстрировать, как содержимое будет обернуто, когда используется `shape-outside` и когда она не используется.

Выход с `shape-outside`



Выход без shape-outside



Форма края

shape-margin свойство CSS добавляет *маржу* shape-outside .

CSS

```
img:nth-of-type(1) {
  shape-outside: circle(80px at 50% 50%);
  shape-margin: 10px;
  float: left;
  width: 200px;
}
img:nth-of-type(2) {
  shape-outside: circle(80px at 50% 50%);
  shape-margin: 10px;
  float: right;
  width: 200px;
}
p {
  text-align: center;
  line-height: 30px; /* purely for demo */
}
```

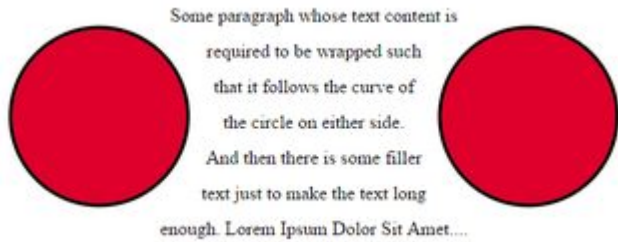
HTML

```


<p>Some paragraph whose text content is required to be wrapped such that it follows the curve
of the circle on either side. And then there is some filler text just to make the text long
enough. Lorem Ipsum Dolor Sit Amet....</p>
```

В этом примере маркер 10px добавляется вокруг **формы** с использованием shape-margin . Это создает немного больше пространства между *воображаемым кругом*, который определяет область с плавающей точкой и фактическое содержимое, которое течет.

Выход:



Прочитайте [Формы для поплавков онлайн: https://riptutorial.com/ru/css/topic/2034/формы-для-поплавков](https://riptutorial.com/ru/css/topic/2034/формы-для-поплавков)

глава 52: фрагментация

Синтаксис

- page-break-after: авто | всегда | избегать | левый | право | начальная | наследовать;
- page-break-before: auto | всегда | избегать | левый | право | начальная | наследовать;
- перерыв страницы: авто | избегать | начальная | наследовать;

параметры

Значение	Описание
авто	По умолчанию. Автоматические разрывы страниц
всегда	Всегда вставляйте разрыв страницы
избежать	Избегайте разрыва страницы (если возможно)
оставил	Вставьте разрывы страниц, чтобы следующая страница была отформатирована как левая страница
право	Вставьте разрывы страниц, чтобы следующая страница была отформатирована как правая страница
начальная	Устанавливает это свойство по умолчанию.
унаследовать	Наследует это свойство от его родительского элемента.

замечания

В CSS нет свойства разрыва страницы. Только 3 свойства (**перерыв страницы, перерыв страницы, перерыв страницы**).

Связанные: `orphans` , `widows` .

Examples

Разрыв страницы печати

```
@media print {  
  p {  
    page-break-inside: avoid;  
  }  
}
```

```
h1 {
  page-break-before: always;
}
h2 {
  page-break-after: avoid;
}
}
```

Этот код делает 3 вещи:

- он предотвращает разрыв страницы внутри любых тэгов, что означает, что абзац никогда не будет разбит на две страницы, если это возможно.
- он заставляет перерыв страницы перед всеми заголовками h1, что означает, что перед каждым возникновением h1 произойдет разрыв страницы.
- он предотвращает разрывы страниц сразу после любого h2

Прочитайте фрагментация онлайн: <https://riptutorial.com/ru/css/topic/4316/фрагментация>

глава 53: функции

Синтаксис

- `<calc()> = calc(<calc-sum>)`
- `<calc-sum> = <calc-product> [['+' | '-'] <calc-product>]*`
- `<calc-product> = <calc-value> ['*' <calc-value> | '/' <number>]*`
- `<calc-value> = <number> | <dimension> | <percentage> | (<calc-sum>)`

замечания

Для `calc()` операторов « - » и « + » требуется `calc()` пространство, но не операторы « * » или « / ».

Все устройства должны быть одного типа; например, попытка умножить высоту на длительность, например, является недопустимой.

Examples

функция `calc()`

Принимает математическое выражение и возвращает числовое значение.

Это особенно полезно при работе с различными типами единиц (например, вычитание значения `px` из процента) для вычисления значения атрибута.

`+`, `-`, `/`, и `*` все могут использоваться, а круглые скобки могут быть добавлены, чтобы указать порядок операций, если это необходимо.

Используйте `calc()` для вычисления ширины элемента `div`:

```
#div1 {
  position: absolute;
  left: 50px;
  width: calc(100% - 100px);
  border: 1px solid black;
  background-color: yellow;
  padding: 5px;
  text-align: center;
}
```

Используйте `calc()` для определения положения фонового изображения:

```
background-position: calc(50% + 17px) calc(50% + 10px), 50% 50%;
```

Используйте `calc()` для определения высоты элемента:

```
height: calc(100% - 20px);
```

функция attr ()

Возвращает значение атрибута выбранного элемента.

Ниже представлен элемент `blockquote`, который содержит символ внутри атрибута `data-*` который CSS может использовать (например, внутри `::before` и `::after` псевдоэлемента), используя эту функцию.

```
<blockquote data-mark=""></blockquote>
```

В следующем блоке CSS символ добавляется до и после текста внутри элемента:

```
blockquote[data-mark]::before,  
blockquote[data-mark]::after {  
  content: attr(data-mark);  
}
```

функция линейного градиента ()

Создает изображение, представляющее линейный градиент цветов.

```
linear-gradient( 0deg, red, yellow 50%, blue);
```

Это создает градиент, идущий снизу вверх, с цветами, начинающимися с красного, затем желтым на 50% и заканчивающимся синим.

функция радиального градиента ()

Создает изображение, представляющее собой градиент цветов, излучающих из центра градиента

```
radial-gradient(red, orange, yellow) /*A gradient coming out from the middle of the  
gradient, red at the center, then orange, until it is finally yellow at the edges*/
```

Функция var ()

Функция `var ()` позволяет использовать переменные CSS.

```
/* set a variable */  
:root {  
  --primary-color: blue;  
}  
  
/* access variable */  
selector {
```

```
color: var(--primary-color);  
}
```

Эта функция в настоящее время разрабатывается. Проверьте caniuse.com для последней поддержки браузера.

Прочитайте функции онлайн: <https://riptutorial.com/ru/css/topic/2214/функции>

глава 54: Цвета

Синтаксис

- цвет: #rgb
- цвет: #rrggbb
- color: rgb [a] (<красный>, <зеленый>, <синий> [, <альфа>])
- color: hsl [a] (<hue>, <saturation%>, <lightness%> [, <alpha>])
- цвет: **colorkeyword** / * зеленый, синий, желтый, оранжевый, красный, ..etc * /

Examples

Цветные ключевые слова

Большинство браузеров поддерживают цветные ключевые слова, чтобы указать цвет. Например, чтобы задать `color` синего элемента, используйте ключевое слово `blue` :





















```
.some-class {  
  color: blue;  
}
```

Ключевые слова CSS не чувствительны к регистру - `blue` , `Blue` и `BLUE` будут `#0000FF` в `#0000FF` .





















Цветные ключевые слова





















Название цвета	Значение Hex	Значения RGB	цвет
AliceBlue	# F0F8FF	RGB (240248255)	
AntiqueWhite	# FAEBD7	RGB (250235215)	
вода	# 00FFFF	RGB (0255255)	
Аквамарин	# 7FFFD4	RGB (127255212)	
лазурь	# F0FFFF	RGB (240255255)	
Бежевый	# F5F5DC	RGB (245245220)	
фора	# FFE4C4	RGB (255228196)	





















Название цвета	Значение Hex	Значения RGB	цвет
черный	# 000000	RGB (0,0,0)	
BlanchedAlmond	#FFEBCD	RGB (255,235,205)	
синий	# 0000FF	RGB (0,0,255)	
Сине-фиолетовый	# 8A2BE2	RGB (138,43,226)	
коричневый	# A52A2A	RGB (165,42,42)	
BurlyWood	# DEB887	RGB (222,184,135)	
CadetBlue	# 5F9EA0	RGB (95,158,160)	
картезианский монастырь	# 7FFF00	RGB (127,255,0)	
Шоколад	# D2691E	RGB (210,105,30)	
коралловый	# FF7F50	RGB (255,127,80)	
CornflowerBlue	# 6495ED	RGB (100,149,237)	
Кукурузные рыльца	# FFF8DC	RGB (255,248,220)	
малиновый	# DC143C	RGB (220,20,60)	
Сяан	# 00FFFF	RGB (0,255,255)	
Темно-синий	# 00008B	RGB (0,0,139)	
DarkCyan	# 008B8B	RGB (0,139,139)	
DarkGoldenRod	# B8860B	RGB (184,134,11)	
Темно-серый	# A9A9A9	RGB (169,169,169)	
Темно-серый	# A9A9A9	RGB (169,169,169)	
Темно-зеленый	# 006400	RGB (0,100,0)	





















Название цвета	Значение Hex	Значения RGB	цвет
DarkKhaki	# BDB76B	RGB (189,183,107)	
DarkMagenta	# 8B008B	RGB (139,0,139)	
DarkOliveGreen	# 556B2F	RGB (85,107,47)	
DarkOrange	# FF8C00	RGB (255,140,0)	
DarkOrchid	# 9932CC	RGB (153,50,204)	
Темно-красный	# 8B0000	RGB (139,0,0)	
DarkSalmon	# E9967A	RGB (233,150,122)	
DarkSeaGreen	# 8FBC8F	RGB (143,188,143)	
DarkSlateBlue	# 483D8B	RGB (72,61,139)	
DarkSlateGray	# 2F4F4F	RGB (47,79,79)	
DarkSlateGrey	# 2F4F4F	RGB (47,79,79)	
DarkTurquoise	# 00CED1	RGB (0,206,209)	
DarkViolet	# 9400D3	RGB (148,0,211)	
Темно-розовый	# FF1493	RGB (255,20,147)	
DeepSkyBlue	# 00BFFF	RGB (0,191,255)	
DimGray	# 696969	RGB (105,105,105)	
DimGrey	# 696969	RGB (105,105,105)	
DodgerBlue	# 1E90FF	RGB (30,144,255)	
огнеупорный кирпич	# B22222	RGB (178,34,34)	
FloralWhite	# FFFAF0	RGB (255,250,240)	


Название цвета	Значение Hex	Значения RGB	цвет
Зеленый лес	# 228B22	RGB (34,139,34)	
фуксия	# FF00FF	RGB (255,0,255)	
Gainsboro	#DCDCDC	RGB (220,220,220)	
GhostWhite	# F8F8FF	RGB (248,248,255)	
Золото	# FFD700	RGB (255,215,0)	
золотарник	# DAA520	RGB (218,165,32)	
Серый	# 808080	RGB (128,128,128)	
Серый	# 808080	RGB (128,128,128)	
зеленый	# 008000	RGB (0,128,0)	
Желто-зеленый	# ADFF2F	RGB (173,255,47)	
Нектар	# F0FFF0	RGB (240,255,240)	
Ярко-розовый	# FF69B4	RGB (255,105,180)	
IndianRed	# CD5C5C	RGB (205,92,92)	
Индиго	# 4B0082	RGB (75,0,130)	
слоновая кость	# FFFFF0	RGB (255,255,240)	
Хаки	# F0E68C	RGB (240,230,140)	
лаванда	# E6E6FA	RGB (230,230,250)	
LavenderBlush	# FFF0F5	RGB (255,240,245)	
LawnGreen	# 7CFC00	RGB (124,252,0)	
LemonChiffon	#FFFACD	RGB (255,250,205)	

Название цвета	Значение Hex	Значения RGB	цвет
Светло-синий	# ADD8E6	RGB (173216230)	
LightCoral	# F08080	RGB (240128128)	
LightCyan	# E0FFFF	RGB (224255255)	
LightGoldenRodYellow	# FAFAD2	RGB (250250210)	
Светло-серый	# D3D3D3	RGB (211211211)	
Светло-серый	# D3D3D3	RGB (211211211)	
Светло-зеленый	# 90EE90	RGB (144238144)	
Светло-розовый	# FFB6C1	RGB (255182193)	
LightSalmon	# FFA07A	RGB (255160122)	
LightSeaGreen	# 20B2AA	RGB (32178170)	
LightSkyBlue	# 87CEFA	RGB (135206250)	
LightSlateGray	# 778899	RGB (119136153)	
LightSlateGrey	# 778899	RGB (119136153)	
LightSteelBlue	# B0C4DE	RGB (176196222)	
Светло-желтого	# FFFFE0	RGB (255255224)	
Лайм	# 00FF00	RGB (0,255,0)	
LimeGreen	# 32CD32	RGB (50,205,50)	
Белье	# FAF0E6	RGB (250240230)	
фуксин	# FF00FF	RGB (255,0,255)	
каштановый	# 800000	RGB (128,0,0)	

Название цвета	Значение Hex	Значения RGB	цвет
MediumAquaMarine	# 66CDAA	RGB (102,205,170)	
MediumBlue	# 0000CD	RGB (0,0,205)	
MediumOrchid	# BA55D3	RGB (186,85,211)	
MediumPurple	# 9370DB	RGB (147,112,219)	
MediumSeaGreen	# 3CB371	RGB (60,179,113)	
MediumSlateBlue	# 7B68EE	RGB (123,104,238)	
MediumSpringGreen	# 00FA9A	RGB (0,250,154)	
MediumTurquoise	# 48D1CC	RGB (72,209,204)	
MediumVioletRed	# C71585	RGB (199,21,133)	
Темно-синий	# 191970	RGB (25,25,112)	
MintCream	# F5FFFA	RGB (245,255,250)	
MistyRose	# FFE4E1	RGB (255,228,225)	
мокасин	# FFE4B5	RGB (255,228,181)	
NavajoWhite	#FFDEAD	RGB (255,222,173)	
военно-морской флот	# 000080	RGB (0,0,128)	
OldLace	# FDF5E6	RGB (253,245,230)	
оливковый	# 808000	RGB (128,128,0)	
OliveDrab	# 6B8E23	RGB (107,142,35)	
оранжевый	# FFA500	RGB (255,165,0)	
Оранжево-красный	# FF4500	RGB (255,69,0)	

Название цвета	Значение Hex	Значения RGB	цвет
орхидея	# DA70D6	RGB (218,112,214)	
PaleGoldenRod	# EEE8AA	RGB (238,232,170)	
Бледно-зеленый	# 98FB98	RGB (152,251,152)	
PaleTurquoise	# AFEEEE	RGB (175,238,238)	
PaleVioletRed	# DB7093	RGB (219,112,147)	
PapayaWhip	# FFEFD5	RGB (255,239,213)	
Peachpuff	# FFDAB9	RGB (255,218,185)	
Перу	# CD853F	RGB (205,133,63)	
розовый	# FFC0CB	RGB (255,192,203)	
слива	# DDA0DD	RGB (221,160,221)	
PowderBlue	# B0E0E6	RGB (176,224,230)	
Пурпурный	# 800080	RGB (128,0,128)	
RebeccaPurple	# 663399	RGB (102,51,153)	
красный	# FF0000	RGB (255,0,0)	
RosyBrown	# BC8F8F	RGB (188,143,143)	
Королевский синий	# 4169E1	RGB (65,105,225)	
SaddleBrown	# 8B4513	RGB (139,69,19)	
Лосось	# FA8072	RGB (250,128,114)	
Sandybrown	# F4A460	RGB (244,164,96)	
Цвет морской волны	# 2E8B57	RGB (46,139,87)	

Название цвета	Значение Hex	Значения RGB	цвет
Морская ракушка	# FFF5EE	RGB (255,245,238)	
охра	# A0522D	RGB (160,82,45)	
Серебряный	# C0C0C0	RGB (192,192,192)	
Голубое небо	# 87CEEB	RGB (135,206,235)	
SlateBlue	# 6A5ACD	RGB (106,90,205)	
Шифер серый	# 708090	RGB (112,128,144)	
SlateGrey	# 708090	RGB (112,128,144)	
Снег	# FFFAFA	RGB (255,250,250)	
SpringGreen	# 00FF7F	RGB (0,255,127)	
Стальной синий	# 4682B4	RGB (70,130,180)	
загар	# D2B48C	RGB (210,180,140)	
чирок	# 008080	RGB (0,128,128)	
чертополох	# D8BFD8	RGB (216,191,216)	
Помидор	# FF6347	RGB (255,99,71)	
Бирюзовый	# 40E0D0	RGB (64,224,208)	
Виолетта	# EE82EE	RGB (238,130,238)	
пшеница	# F5DEB3	RGB (245,222,179)	
белый	# FFFFFFFF	RGB (255,255,255)	
Белый дым	# F5F5F5	RGB (245,245,245)	
желтый	# FFFF00	RGB (255,255,0)	

Название цвета	Значение Hex	Значения RGB	цвет
Желто-зеленый	# 9ACD32	RGB (154,205,50)	

В дополнение к названным цветам также есть ключевое слово `transparent` , которое представляет собой полностью прозрачный черный цвет: `rgba(0,0,0,0)`

Шестнадцатеричное значение

Фон

Цвета CSS также могут быть представлены в виде шестигранного триплета, где элементы представляют красные, зеленые и синие компоненты цвета. Каждое из этих значений представляет собой число в диапазоне от `00` до `FF` или от `0` до `255` в десятичной нотации. Можно использовать строчные и / или строчные шестнадцатеричные значения (т. `#3fc` = `#3FC` = `#33ffcc`). Браузер интерпретирует `#369` как `#336699` . Если это не то, что вы намеревались, а скорее хотите `#306090` , вам нужно указать это явно.

Общее количество цветов, которое может быть представлено шестнадцатеричной нотацией, составляет 256^3 или `16 777 216`.

Синтаксис

```
color: #rrggbb;
color: #rgb
```

Значение	Описание
rr	<code>00 - FF</code> для количества красного
gg	<code>00 - FF</code> для количества зеленого
bb	<code>00 - FF</code> для количества синего

```
.some-class {
  /* This is equivalent to using the color keyword 'blue' */
  color: #0000FF;
}

.also-blue {
  /* If you want to specify each range value with a single number, you can!
  This is equivalent to '#0000FF' (and 'blue') */
  color: #00F;
}
```

Шестнадцатеричная нотация используется для указания значений цвета в цветовом формате RGB в соответствии с «**Числовыми значениями цвета**» W3C .

В Интернете доступно множество инструментов для поиска шестнадцатеричных (или просто шестнадцатеричных) значений цвета.

Найдите « **шестнадцатеричную цветовую палитру** » или « **шестнадцатеричный подборщик цветов** » с вашим любимым веб-браузером, чтобы найти кучу вариантов!

Значения шестнадцатеричных значений всегда начинаются с знака фунта (#), длиной до шести «цифр» и не зависят от регистра: то есть, они не заботятся о капитализации. #FFC125 и #ffc125 имеют один и тот же цвет.

rgb () Обозначение

RGB - это аддитивная цветовая модель, которая представляет цвета как смеси красного, зеленого и синего света. По сути, представление RGB является десятичным эквивалентом шестнадцатеричного обозначения. В шестнадцатеричном формате каждое число колеблется от 00-FF, что эквивалентно 0-255 в десятичном и 0% -100% в процентах.

```
.some-class {
  /* Scalar RGB, equivalent to 'blue'*/
  color: rgb(0, 0, 255);
}

.also-blue {
  /* Percentile RGB values*/
  color: rgb(0%, 0%, 100%);
}
```

Синтаксис

```
rgb(<red>, <green>, <blue>)
```

Значение	Описание
<red>	целое число от 0 до 255 или процент от 0 - 100%
<green>	целое число от 0 до 255 или процент от 0 - 100%
<blue>	целое число от 0 до 255 или процент от 0 - 100%

hsl () Обозначение

HSL обозначает **оттенок** («какой цвет»), **насыщенность** («сколько цветов») и **легкость** («сколько белого»).

Оттенок представлен как угол от 0 ° до 360 ° (без единиц), а насыщенность и легкость представлены в процентах.

```
p {
  color: hsl(240, 100%, 50%); /* Blue */
}
```

Синтаксис

```
color: hsl(<hue>, <saturation>%, <lightness>%);
```

Значение	Описание
<hue>	(без единиц), где 0 ° красный, 60 ° - желтый, 120 ° - зеленый, 180 ° - голубой, 240 ° - синий, 300 ° - пурпурный, а 360 ° красный
<saturation>	указанная в процентах, где 0% полностью ненасыщенна (оттенки серого) и 100% полностью насыщена (ярко окрашена)
<lightness>	указано в процентах, где 0% полностью черный, а 100% полностью белый

Заметки

- Насыщение 0% всегда производит оттенок серого; изменение оттенка не влияет.
- Легкость 0% всегда производит черный цвет, а 100% всегда производит белый; изменение оттенка или насыщенности не влияет.

currentColor

`currentColor` возвращает вычисленное значение цвета текущего элемента.

Использовать в одном элементе

Здесь `currentColor` оценивается как красный, поскольку свойство `color` установлено на `red`:

```
div {
  color: red;
  border: 5px solid currentColor;
  box-shadow: 0 0 5px currentColor;
}
```


В этом случае указание `currentColor` для границы, скорее всего, избыточно, так как исключение должно давать идентичные результаты. Используйте свойство `currentColor` внутри свойства `border` внутри одного и того же элемента, если оно будет перезаписано иначе из-за **более конкретного** селектора.

Так как это вычисленный цвет, граница будет зеленой в следующем примере из-за второго правила, переопределяющего первое:

```
div {
  color: blue;
  border: 3px solid currentColor;
  color: green;
}
```

Унаследовано от родительского элемента

Цвет родителя наследуется, здесь `currentColor` оценивается как «синий», делая синий цвет дочернего элемента.

```
.parent-class {
  color: blue;
}

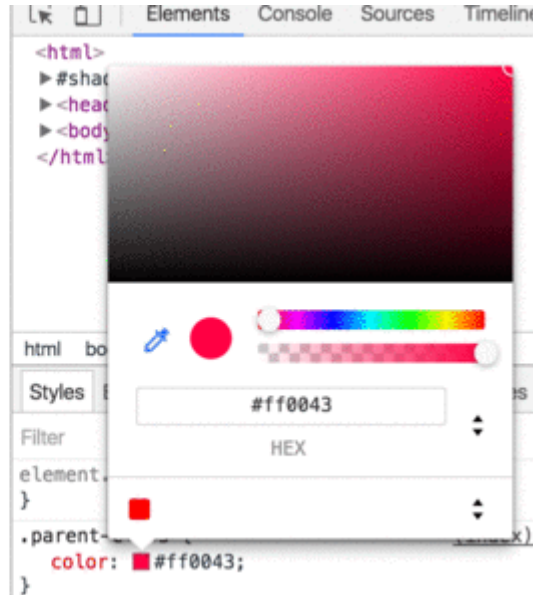
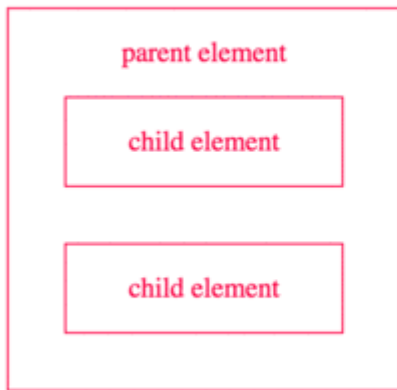
.parent-class .child-class {
  border-color: currentColor;
}
```

`currentColor` также может использоваться другими правилами, которые обычно не наследуются от свойства цвета, такого как фоновый цвет. В приведенном ниже примере показано, как дети используют цвет, установленный в родительском в качестве фона:

```
.parent-class {
  color: blue;
}

.parent-class .child-class {
  background-color: currentColor;
}
```

Возможный результат:



rgba () Обозначение

Подобно нотации [rgb \(\)](#) , но с дополнительным значением альфа (непрозрачность).

```
.red {
  /* Opaque red */
  color: rgba(255, 0, 0, 1);
}

.red-50p {
  /* Half-translucent red. */
  color: rgba(255, 0, 0, .5);
}
```

Синтаксис

```
rgba(<red>, <green>, <blue>, <alpha>);
```

Значение	Описание
<red>	целое число от 0 до 255 или процент от 0 - 100%
<green>	целое число от 0 до 255 или процент от 0 - 100%
<blue>	целое число от 0 до 255 или процент от 0 - 100%
<alpha>	число от 0 до 1, где 0.0 полностью прозрачно и 1,0 полностью непрозрачно

hsla () Обозначение

Подобно нотации [hsl \(\)](#) , но с добавленным значением альфа (непрозрачность).

```
hsla(240, 100%, 50%, 0)      /* transparent */
hsla(240, 100%, 50%, 0.5)  /* half-translucent blue */
hsla(240, 100%, 50%, 1)    /* fully opaque blue */
```

Синтаксис

```
hsla(<hue>, <saturation>%, <lightness>%, <alpha>);
```

Значение	Описание
<hue>	(без единиц), где 0 ° красный, 60 ° - желтый, 120 ° - зеленый, 180 ° - голубой, 240 ° - синий, 300 ° - пурпурный, а 360 ° красный
<saturation>	процент, где 0% полностью ненасыщен (оттенки серого) и 100% полностью насыщен (ярко окрашен)
<lightness>	процент, где 0% полностью черный, а 100% полностью белый
<alpha>	число от 0 до 1, где 0 полностью прозрачно и 1 полностью непрозрачно

Прочитайте Цвета онлайн: <https://riptutorial.com/ru/css/topic/644/цвета>

глава 55: Центрирование

Examples

Использование CSS-преобразования

[Преобразования CSS](#) основаны на размере элементов, поэтому, если вы не знаете, насколько высокий или широкий ваш элемент, вы можете установить его на 50% сверху и слева от относительного контейнера и перевести его на 50% слева и вверх центрировать его вертикально и горизонтально.

Имейте в виду, что с помощью этой методики элемент может заканчиваться визуализацией на нецелой границе пикселя, что делает его размытым. См. [Этот ответ в SO](#) для обходного пути.

HTML

```
<div class="container">
  <div class="element"></div>
</div>
```

CSS

```
.container {
  position: relative;
}

.element {
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
}
```

[Просмотреть пример в JSFiddle](#)

СОВМЕСТИМОСТЬ КРЕСТОВЫХ БРАУЗЕРОВ

Свойству transform необходимы префиксы, поддерживаемые старыми браузерами. Префиксы необходимы для Chrome <= 35, Safari <= 8, Opera <= 22, Android Browser <= 4.4.4 и IE9. CSS-преобразования не поддерживаются IE8 и более ранними версиями.

Ниже приведено общее объявление преобразования для предыдущего примера:

```
-webkit-transform: translate(-50%, -50%); /* Chrome, Safari, Opera, Android */
-ms-transform: translate(-50%, -50%); /* IE 9 */
transform: translate(-50%, -50%);
```

Для получения дополнительной информации см. [Canluse](#) .

ДОПОЛНИТЕЛЬНАЯ ИНФОРМАЦИЯ

- Элемент позиционируется в соответствии с первым нестатическим родителем (`position: relative` , `absolute` или `fixed`). Узнайте больше в этой [скрипке](#) и этой [теме документации](#) .
- Для центрирования по горизонтали используйте `left: 50%` и `transform: translateX(-50%)` . То же самое касается центрирования по вертикали: центр с `top: 50%` и `transform: translateY(-50%)` .
- Использование нестатических элементов ширины / высоты с помощью этого метода центрирования может привести к появлению срезающего элемента. Это чаще всего происходит с элементами, содержащими текст, и может быть исправлено путем добавления: `margin-right: -50%;` и `margin-bottom: -50%;` , Просмотрите эту [скрипку](#) для получения дополнительной информации.

Использование Flexbox

HTML:

```
<div class="container">
  
</div>
```

CSS:

```
html, body, .container {
  height: 100%;
}
.container {
  display: flex;
  justify-content: center; /* horizontal center */
}
img {
  align-self: center; /* vertical center */
}
```

[Просмотреть результат](#)

HTML:

```

```

CSS:

```
html, body {
```

```
height: 100%;
}
body {
  display: flex;
  justify-content: center; /* horizontal center */
  align-items: center;    /* vertical center */
}
```

[Просмотреть результат](#)

См [Динамическая горизонтального и вертикального центровку](#) под [Flexbox](#) документации для получения более подробной информации о Flexbox и какие стили означают.

Поддержка браузера

Flexbox поддерживается всеми основными браузерами, [за исключением версий IE до 10](#) .

Некоторые недавние версии браузера, такие как Safari 8 и IE10, требуют [префиксов поставщиков](#) .

Для быстрого создания префиксов существует [инструмент Autoprefixer](#) , сторонний инструмент.

Для старых браузеров (например, IE 8 и 9) [доступен Polyfill](#) .

Для более подробного просмотра поддержки браузера Flexbox см. [Этот ответ](#) .

Использование позиции: абсолютная

Работа в старых браузерах (IE >= 8)

Автоматические поля, в сочетании со значениями нуля для `left` и `right` или `top` и `bottom` смещений, будут центрировать абсолютно позиционированные элементы внутри своего родителя.

[Просмотреть результат](#)

HTML

```
<div class="parent">
  
</div>
```

CSS

```
.parent {
  position: relative;
  height: 500px;
}
```

```
.center {
  position: absolute;
  margin: auto;
  top: 0;
  right: 0;
  bottom: 0;
  left: 0;
}
```

Элементы, которые не имеют своей собственной неявной ширины и высоты, как изображения, будут нуждаться в этих значениях.

Другие ресурсы: [Абсолютное центрирование в CSS](#)

Техника элемента призрак (взлома Михала Чернова)

Этот метод работает, даже когда размеры контейнера неизвестны.

Настройте элемент «призрак» внутри центрированного контейнера, который будет на 100% высотой, затем используйте `vertical-align: middle` как для элемента, так и для центрированного элемента.

CSS

```
/* This parent can be any width and height */
.block {
  text-align: center;

  /* May want to do this if there is risk the container may be narrower than the element
  inside */
  white-space: nowrap;
}

/* The ghost element */
.block:before {
  content: '';
  display: inline-block;
  height: 100%;
  vertical-align: middle;

  /* There is a gap between ghost element and .centered,
  caused by space character rendered. Could be eliminated by
  nudging .centered (nudge distance depends on font family),
  or by zeroing font-size in .parent and resetting it back
  (probably to 1rem) in .centered. */
  margin-right: -0.25em;
}

/* The element to be centered, can also be of any width and height */
.centered {
  display: inline-block;
  vertical-align: middle;
  width: 300px;
  white-space: normal; /* Resetting inherited nowrap behavior */
}
```

HTML

```
<div class="block">
  <div class="centered"></div>
</div>
```

Использование выравнивания текста

Наиболее распространенным и простым типом центрирования является текст строк в элементе. CSS имеет правило `text-align: center` для этой цели:

HTML

```
<p>Lorem ipsum</p>
```

CSS

```
p {
  text-align: center;
}
```

Это не работает для центрирования целых элементов блока. `text-align` управляет только выравниванием встроенного контента, такого как текст в его родительском блочном элементе.

Подробнее о `text-align` в разделе « [Типография](#) ».

Центрирование по отношению к другому элементу

Мы увидим, как центрировать контент в зависимости от высоты близкого элемента.

Совместимость: IE8 +, все другие современные браузеры.

HTML

```
<div class="content">
  <div class="position-container">
    <div class="thumb">
      
    </div>
    <div class="details">
      <p class="banner-title">text 1</p>
      <p class="banner-text">content content content content content content content content
content content content content content content content content</p>
      <button class="btn">button</button>
    </div>
  </div>
</div>
```

CSS


```
.content * {
  box-sizing: border-box;
}
.content .position-container {
  display: table;
}
.content .details {
  display: table-cell;
  vertical-align: middle;
  width: 33.333333%;
  padding: 30px;
  font-size: 17px;
  text-align: center;
}
.content .thumb {
  width: 100%;
}
.content .thumb img {
  width: 100%;
}
```

Ссылка на [JSFiddle](#)

Основными пунктами являются `.thumb`, `.details` и `.position-container` container:

- В `.position-container` должен `display: table`.
- `.details` должны иметь реальную ширину `width: ...` и `display: table-cell`, `vertical-align: middle`.
- `.thumb` должна иметь `width: 100%` если вы хотите, чтобы она заняла все оставшееся пространство, и на нее будет влиять ширина `.details`.
- Изображение (если у вас есть изображение) внутри `.thumb` должно иметь `width: 100%`, но это не обязательно, если у вас есть правильные пропорции.

Вертикальное выравнивание с тремя строками кода

Поддерживается IE11 +

[Посмотреть результат](#)

Используйте эти 3 строки для вертикального выравнивания практически всего. Просто убедитесь, что `div / image`, к которому вы применяете код, имеет родителя с высотой.

CSS

```
div.vertical {
  position: relative;
  top: 50%;
  transform: translateY(-50%);
}
```

HTML

```
<div class="vertical">Vertical aligned text!</div>
```

Вертикально выравнивать изображение внутри div

HTML

```
<div class="wrap">
  
</div>
```

CSS

```
.wrap {
  height: 50px; /* max image height */
  width: 100px;
  border: 1px solid blue;
  text-align: center;
}
.wrap:before {
  content: "";
  display: inline-block;
  height: 100%;
  vertical-align: middle;
  width: 1px;
}
img {
  vertical-align: middle;
}
```

Горизонтальное и вертикальное центрирование с использованием таблицы

Можно легко центрировать дочерний элемент, используя свойство отображения `table`.

HTML

```
<div class="wrapper">
  <div class="parent">
    <div class="child"></div>
  </div>
</div>
```

CSS

```
.wrapper {
  display: table;
  vertical-align: center;
  width: 200px;
  height: 200px;
}
```

```
background-color: #9e9e9e;
}
.parent {
display: table-cell;
vertical-align: middle;
text-align: center;
}
.child {
display: inline-block;
vertical-align: middle;
text-align: center;
width: 100px;
height: 100px;
background-color: teal;
}
```

Используя calc ()

Функция calc () является частью нового синтаксиса в CSS3, в котором вы можете вычислить (математически), какой размер / позиция занимает ваш элемент, используя различные значения, такие как пиксели, проценты и т. Д. Примечание: - всякий раз, когда вы используете эту функцию , всегда занимайтесь пространством между двумя значениями calc(100% - 80px) .

CSS

```
.center {
position: absolute;
height: 50px;
width: 50px;
background: red;
top: calc(50% - 50px / 2); /* height divided by 2*/
left: calc(50% - 50px / 2); /* width divided by 2*/
}
```

HTML

```
<div class="center"></div>
```

Вертикально выровнять элементы динамической высоты

Применение css интуитивно не дает желаемых результатов, потому что

- vertical-align:middle **не** применяется к элементам уровня блока
- margin-top:auto И margin-bottom:auto **используемые значения будут вычисляться как НОЛЬ**
- margin-top:-50% **процентных значений на основе процентов вычисляются относительно ширины** содержащего блока

Для самой широкой поддержки браузера используется обходной путь со вспомогательными элементами:

HTML

```
<div class="vcenter--container">
  <div class="vcenter--helper">
    <div class="vcenter--content">
      <!--stuff-->
    </div>
  </div>
</div>
```

CSS

```
.vcenter--container {
  display: table;
  height: 100%;
  position: absolute;
  overflow: hidden;
  width: 100%;
}
.vcenter--helper {
  display: table-cell;
  vertical-align: middle;
}
.vcenter--content {
  margin: 0 auto;
  width: 200px;
}
```

[jsfiddle](#) из [оригинальный вопрос](#) . Этот подход

- работает с динамическими элементами высоты
- уважает поток контента
- поддерживается устаревшими браузерами

Использование высоты линии

Вы также можете использовать `line-height` чтобы центрировать вертикально одну строку текста внутри контейнера:

CSS

```
div {
  height: 200px;
  line-height: 200px;
}
```

Это довольно уродливо, но может быть полезно внутри элемента `<input />` . Свойство `line-height` работает только тогда, когда текст, который будет центрирован, охватывает одну строку. Если текст переносится на несколько строк, результирующий вывод не будет центрирован.

Центрирование вертикально и горизонтально, не беспокоясь о высоте или ширине

Следующая методика позволяет добавить ваш контент в элемент HTML и центрировать его как по горизонтали, так и по вертикали, **не беспокоясь о его высоте или ширине** .

Внешний контейнер

- должен иметь `display: table;`

Внутренний контейнер

- должен иметь `display: table-cell;`
- должен иметь `vertical-align: middle;`
- должен иметь `text-align: center;`

Поле содержимого

- должен иметь `display: inline-block;`
- следует перенастроить горизонтальное выравнивание текста, например. `text-align: left;` или `text-align: right;` , если вы хотите, чтобы текст был центрирован

демонстрация

HTML

```
<div class="outer-container">
  <div class="inner-container">
    <div class="centered-content">
      You can put anything here!
    </div>
  </div>
</div>
```

CSS

```
body {
  margin : 0;
}

.outer-container {
  position : absolute;
  display: table;
  width: 100%; /* This could be ANY width */
  height: 100%; /* This could be ANY height */
  background: #ccc;
}
```

```
.inner-container {
  display: table-cell;
  vertical-align: middle;
  text-align: center;
}

.centered-content {
  display: inline-block;
  text-align: left;
  background: #fff;
  padding: 20px;
  border: 1px solid #000;
}
```

Смотрите также [эту скрипку](#) !

Центрирование с фиксированным размером

Если размер вашего контента фиксирован, вы можете использовать абсолютное позиционирование до 50% с `margin` который уменьшает половину ширины и высоты вашего контента:

HTML

```
<div class="center">
  Center vertically and horizontally
</div>
```

CSS

```
.center {
  position: absolute;
  background: #ccc;

  left: 50%;
  width: 150px;
  margin-left: -75px; /* width * -0.5 */

  top: 50%;
  height: 200px;
  margin-top: -100px; /* height * -0.5 */
}
```

Горизонтальное центрирование с фиксированной шириной

Вы можете центрировать элемент горизонтально, даже если вы не знаете высоту содержимого:

HTML

```
<div class="center">
```

```
Center only horizontally
</div>
```

CSS

```
.center {
  position: absolute;
  background: #ccc;

  left: 50%;
  width: 150px;
  margin-left: -75px; /* width * -0.5 */
}
```

Вертикальное центрирование с фиксированной высотой

Вы можете центрировать элемент по вертикали, если знаете высоту элемента:

HTML

```
<div class="center">
  Center only vertically
</div>
```

CSS

```
.center {
  position: absolute;
  background: #ccc;

  top: 50%;
  height: 200px;
  margin-top: -100px; /* width * -0.5 */
}
```

Использование margin: 0 auto;

Объекты могут быть центрированы с использованием `margin: 0 auto;` если они являются блочными элементами и имеют определенную ширину.

HTML

```
<div class="containerDiv">
  <div id="centeredDiv"></div>
</div>

<div class="containerDiv">
  <p id="centeredParagraph">This is a centered paragraph.</p>
</div>

<div class="containerDiv">
  
</div>
```

CSS

```
.containerDiv {
  width: 100%;
  height: 100px;
  padding-bottom: 40px;
}

#centeredDiv {
  margin: 0 auto;
  width: 200px;
  height: 100px;
  border: 1px solid #000;
}

#centeredParagraph {
  width: 200px;
  margin: 0 auto;
}

#centeredImage {
  display: block;
  width: 200px;
  margin: 0 auto;
}
```

Результат:



This is a centered paragraph.



Пример JSFiddle: [центрирование объектов с полем: 0 auto;](#)

Прочитайте [Центрирование онлайн](#): <https://riptutorial.com/ru/css/topic/299/центрирование>

глава 56: Шаблоны проектирования CSS

Вступление

Эти примеры предназначены для документирования CSS-специфических шаблонов проектирования, таких как [BEM](#) , [OOCSS](#) и [SMACSS](#) .

Эти примеры НЕ предназначены для документирования фреймворков CSS, таких как [Bootstrap](#) или [Foundation](#) .

замечания

Эти примеры предназначены для документирования CSS-специфических методологий / шаблонов проектирования.

Эти методологии включают, но не ограничиваются следующим:

- [BEM](#)
- [OOCSS](#)
- [SMACSS](#)

Эти примеры НЕ предназначены для документирования фреймворков CSS, таких как [Bootstrap](#) или [Foundation](#) . Хотя вы можете включить примеры того, как применять одну или несколько методологий / шаблонов CSS с CSS-структурой, эти примеры должны сосредоточиться на методологиях / шаблонах проектирования с этой конкретной структурой и на использовании самой структуры.

Examples

BEM

[BEM](#) обозначает `Blocks, Elements and Modifiers` . Это методология, первоначально задуманная российской технологической компанией [Yandex](#) , но которая приобрела немалую тягу среди американских и западноевропейских веб-разработчиков.

Как следует из этого, методология BEM - это все, что касается компонентности вашего кода HTML и CSS в трех типах компонентов:

- **Блоки:** автономные объекты, которые имеют смысл сами по себе

Примерами являются `header` , `container` , `menu` , `checkbox` и `textbox`

- **Элементы:** часть блоков, которые не имеют отдельного значения и семантически

привязаны к их блокам.

Примерами являются `menu item list item`, `checkbox caption` И `header title`

- **Модификаторы:** флаги на блоке или элементе, используемые для изменения внешнего вида или поведения

Примерами являются `disabled`, `highlighted`, `checked`, `fixed`, `size big` И `color yellow`

Цель ВЕМ - оптимизировать читаемость, удобство обслуживания и гибкость вашего CSS-кода. Способ достижения этого - применить следующие правила.

- Стили блоков никогда не зависят от других элементов на странице
- Блоки должны иметь простое, короткое имя и избегать `_` или `-` символов
- При стилизации элементов используйте селектор формата `blockname__elementname`
- При модификаторах стилей используйте селектора формата `blockname--modifiername` И `blockname__elementname--modifiername`
- Элементы или блоки, которые имеют модификаторы, должны наследовать все от блока или элемента, который он модифицирует, кроме свойств, которые должен модифицировать модификатор

Пример кода

Если вы примените ВЕМ к вашим элементам формы, ваши селекторы CSS должны выглядеть примерно так:

```
.form { } // Block
.form--theme-xmas { } // Block + modifier
.form--simple { } // Block + modifier
.form__input { } // Block > element
.form__submit { } // Block > element
.form__submit--disabled { } // Block > element + modifier
```

Соответствующий HTML должен выглядеть примерно так:

```
<form class="form form--theme-xmas form--simple">
  <input class="form__input" type="text" />
  <input class="form__submit form__submit--disabled" type="submit" />
</form>
```

Прочитайте Шаблоны проектирования CSS онлайн: <https://riptutorial.com/ru/css/topic/10823/шаблоны-проектирования-css>

кредиты

S. No	Главы	Contributors
1	Начало работы с CSS	adamboro , animuson , Ashwin Ramaswami , awe , Boysenb3rry , Chris , Community , csx.cc , darrylyeo , FelipeAls , Gabriel R. , Garconis , Gerardas , GoatsWearHats , G-Wiz , Harish Gyanani , Heri Hehe Setiawan , J Atkin , Jmh2013 , joe_young , Jose Gomez , Just a student , Lambda Ninja , Marjorie Pickard , Nathan Arthur , patelarpan , RamenChef , Rocket Risa , Saroj Sasmal , ScientiaEtVeritas , selvassn , Sverri M. Olsen , Teo Dragovic , Todd , TylerH , Vivek Ghaisas , Xinyang Li , ZaneDickens
2	2D-преобразования	Charlie H , Christiaan Maks , Harry , Hors Sujet , John Slegers , Luke Taylor , Madalina Taina , mnoronha , PaMaDo , Praveen Kumar , RamenChef , web-tiki , zer00ne
3	3D-преобразования	Harry , Luka Kerr , Madalina Taina , mnoronha , Mr. Meeseeks , Nhan , RamenChef , web-tiki
4	CSS-спрайты	Elegant.Scripting , Jmh2013 , RamenChef , Ted Goas , Ulrich Schwarz
5	Анимации	Aeolingamenfel , apaul , Dex Star , Jasmin Solanki , Nathan Arthur , RamenChef , Richard Hamilton , TylerH
6	бордюры	andreas , Cassidy Williams , doctorsherlock , FelipeAls , Gnietschow , Harry , jaredsk , Madalina Taina , Nobal Mohan , RamenChef , ScientiaEtVeritas , Trevor Clarke
7	Вертикальное центрирование	animuson , AVAVT , bocanegra , Chiller , Chris , jaredsk , leo_ap , mmativ , patelarpan , Phil , Praveen Kumar , RamenChef
8	Взломы Internet Explorer	Aeolingamenfel , animuson , Elizaveta Revyakina , feeela , John Slegers , LiLacTac , Praveen Kumar , RamenChef , ScientiaEtVeritas , Timothy Miller , TylerH
9	Гибкая компоновка ящиков (Flexbox)	Ahmad Alfy , Asim K T , FelipeAls , fzzylogic , James Donnelly , Jef , Lambda Ninja , Marc-Antoine Leclerc , Nathan Arthur , Nhan , Ori Marash , RamenChef , Randy , Squazz , takeradi , Ted Goas , Timothy Morris , TylerH
10	Единицы длины	4dgaurav , A B , animuson , Epodax , geeksal , J F , Marc , Milche Patern , Ortomala Lokni , RamenChef , Richard Hamilton , rmondasilva , Robert Koritnik , Robotnicka , ScientiaEtVeritas , StefanBob , Stewartside , Thomas Altmann , Toby , user2622348 ,

		vladdobra , Zakaria Acharki , zer00ne
11	Запросы мультимедиа	amflare , Chathuranga Jayanath , darrylyeo , Demeter Dimitri , dodopok , James Donnelly , Jmh2013 , joe_young , joejoe31b , John Slegers , Matas Vaitkevicius , Mattia Astorino , Maximillian Laumeister , Nathan Arthur , Praveen Kumar , RamenChef , ScientiaEtVeritas , srikarg , Teo Dragovic , Viktor
12	Запросы функций	Andrew Myers , RamenChef
13	Каскадирование и специфика	amflare , Arjan Einbu , brandaemon , DarkAjax , dippas , dmkerr , geeksal , Grant Palin , G-Wiz , James Donnelly , jehna1 , John Slegers , kingcobra1986 , Matas Vaitkevicius , Nathan Arthur , Nhan , Ortomala Lokni , RamenChef , ScientiaEtVeritas , Sunnyok , Ze Rubeus
14	Колонны	Brett DeWoody , Madalina Taina , RamenChef
15	Комментарии	animuson , bdkopen , coderfin , Madalina Taina , Nick
16	Контекст стеков	Nemanja Trifunovic , RamenChef
17	Контексты форматирования блоков	Madalina Taina , Milan Laslop , RamenChef
18	контуры	Arif , Casey , Chathuranga Jayanath , Daniel Nugent , FelipeAls , Madalina Taina , RamenChef , ScientiaEtVeritas
19	коробчатого тень	Hristo , Madalina Taina , RamenChef
20	Макет встроенного блока	Marten Koetsier , RamenChef
21	Маржа	Arjan Einbu , cdm , Chris Spittles , Community , J F , Madalina Taina , Mr_Green , Nathan Arthur , RamenChef , rejnev , Sun Qingyao , Sunnyok , Tot Zam , Trevor Clarke
22	Модель коробки	Arjan Einbu , Ben Rhys-Lewis , Benolot , BiscuitBaker , FelipeAls , James Donnelly , Lambda Ninja , Nathan Arthur , Ortomala Lokni , RamenChef , ScientiaEtVeritas , Sergej , V-Kopio
23	набивка	Andy G , CalvT , Felix A J , Madalina Taina , Mehdi Dehghani , Nathan , Paul Sweatte , pixelbandito , RamenChef , StefanBob , Will DiFruscio
24	наследование	Chris , mnoronha , RamenChef
25	Несколько	bipon , Sebastian Zartner

	столбцов	
26	Нормализация стилей браузера	andre mcgruder , Confused One , Grant Palin , MMachinegun , mnoronha , RamenChef , SeinopSys
27	Обрезка и маскировка	Andre Lopes , Harry , RamenChef , ScientiaEtVeritas , web-tiki
28	Объектная модель CSS (CSSOM)	Alohci , animuson , feeela , Paul Sweatte , RamenChef , rishabh dev
29	Одиночные элементы	andreas , animuson , Brett DeWoody , Chiller , J F , Nick , RamenChef , ScientiaEtVeritas , TylerH
30	перелив	Andrew , bdkopen , jgh , Madalina Taina , Miles , Qaz , RamenChef , ScientiaEtVeritas , Ted Goas , Zac
31	Переходы	Christiaan Maks , dippas , Harry , Praveen Kumar , RamenChef , Richard Hamilton , ScientiaEtVeritas , Sergey Denisov , web-tiki
32	Поддержка браузера и префиксы	Andrew , animuson , Braiam , Nhan , Obsidian , RamenChef , ScientiaEtVeritas , Shaggy , TylerH
33	позиционирование	Abhishek Singh , Alohci , animuson , Blunderfest , CalvT , Cassidy Williams , Chetan Joshi , GingerPlusPlus , Jacob Gray , Lambda Ninja , Madalina Taina , Matthew Beckman , Mattia Astorino , Rahul Nanwani , RamenChef , Sourav Ghosh , Stephen Leppik , Theodore K. , TylerH
34	Пользовательские свойства (переменные)	animuson , Brett DeWoody , Community , Daniel Käfer , Muthu Kumaran , Obsidian , RamenChef , RedRiderX , TylerH
35	помутнение	Andrew , animuson , Hillel Tech , Madalina Taina , RamenChef
36	Поплавки	demonofthemist , FelipeAls , Madalina Taina , Nathan Arthur , RamenChef , vishak
37	Псевдо-элементы	4dgaurav , ankit , Chris , Community , dippas , dmnsgn , geek1011 , geeksal , Gofilord , kevin vd , Marcatectura , Milche Patern , Nathan , Pat , Praveen Kumar , RamenChef , ScientiaEtVeritas , Shaggy , Stewartside , Sunnyok
38	Свойство фильтра	Jeffery Tang , Nathan , RamenChef
39	Селекторы	75th Trombone , A.J. , Aaron , abaracedo , Ahmad Alfy , Alex Filatov , amflare , Anil , animuson , Araknid , Arjan Einbu , Ashwin

		Ramaswami, BoltClock, Cerbrus, Charlie H, Chris, Chris Nager, Clinton Yeboah, Community, CPython, darrylyeo, Dave Everitt, David Fullerton, Demeter Dimitri, designcise, Devid Farinelli, Devon Bernard, Dinidu, dippas, Erenor Paz, Felix Edelmann, Felix Schütz, flyingfish, Forty, fracz, Frits, gandreadis, geeksal, George Bailey, George Grigorita, H. Pauwelyn, HansCz, henry, Hugo Buff, Hynes, J Atkin, J F, Jacob Gray, James Donnelly, James Taylor, Jasha, jehna1, Jmh2013, joejoe31b, Joël Bonet Rodríguez, John Slegers, Kurtis Beavers, Madalina Taina, Marc, Mark Perera, Matas Vaitkevicius, Matsemann, Michael_B, Milan Laslop, Naeem Shaikh, Nathan Arthur, Nick, Ortomala Lokni, Persijn, Praveen Kumar, RamenChef, rdans, Richard Hamilton, Rion Williams, Robert Koritnik, RockPaperLizard, RoToRa, Sbats, ScientiaEtVeritas, Shaggy, Siavas, Stewartside, sudo bangbang, Sumner Evans, Sunnyok, ThatWeirdo, theB, Thomas Gerot, TylerH, xpy, Yury Fedorov, Zac, Zaffy, Zaz, Ze Rubeus, Zze
40	сетка	Chris Spittles, FelipeAls, Jonathan Zúñiga, Mike McCaughan, Nhan, Praveen Kumar, RamenChef, Sebastian Zartner
41	Спектакль	mnoronha, RamenChef, TrungDQ
42	Стили списка	animuson, Madalina Taina, Marten Koetsier, RamenChef, Ted Goas
43	Стилизация курсора	cone56, Madalina Taina, ScientiaEtVeritas, Squazz
44	Структура и форматирование правила CSS	Alon Eitan, darrylyeo, Marjorie Pickard
45	Счетчики	Harry, RamenChef
46	таблицы	Casper Spruit, Chris, FelipeAls, JHS, Madalina Taina
47	Типография	Alex Morales, Alohci, andreas, Arjan Einbu, ChaoticTwist, Evgeny, Felix A J, Goulven, Hynes, insertusernamehere, James Donnelly, joe_young, Jon Chan, Madalina Taina, Michael Moriarty, Nathan Arthur, Nhan, Praveen Kumar, RamenChef, Richard Hamilton, rmondasilva, Ryan, Sourav Ghosh, Suhaib Janjua, Ted Goas, Toby, ToniB, Trevor Clarke, user2622348, Vlusion, Volker E.
48	Управление макетами	Arjun Iv, Chathuranga Jayanath, Chris, Jmh2013, Kevin Katzke, Kurtis Beavers, Madalina Taina, mnoronha, Niek Brouwer, RamenChef, Sander Koedood, ScientiaEtVeritas, SeinopSys

49	Установка и размещение объектов	4444 , Miles , RamenChef
50	Фон	4444 , Ahmad Alfy , animuson , Asim K T , Ben Rhys-Lewis , Boris , CalvT , cdm , Charlie H , CocoaBean , Dan Devine , Dan Eastwell , Daniel G. Blázquez , Daniel Stradowski , Darthstroke , designcise , Devid Farinelli , dippas , fcalderan , FelipeAls , Goose , Horst Jahns , Hynes , Jack , Jacob Gray , James Taylor , John Slegers , Jon Chan , Jonathan Zúñiga , Kevin Montrose , Louis St-Amour , Madalina Taina , Maximillian Laumeister , Michael Moriarty , Mr. Alien , mtb , Nate , Nathan Arthur , Nhan , Persijn , Praveen Kumar , RamenChef , Richard Hamilton , ScientiaEtVeritas , Sergey Denisov , Shaggy , Sourav Ghosh , Stewartside , Stratboy , think123 , Timothy , Trevor Clarke , TylerH , Zac , Zeta , Zze
51	Формы для поплавков	animuson , Harry , RamenChef , ScientiaEtVeritas
52	фрагментация	animuson , dodopok , Madalina Taina , Milan Laslop , RamenChef
53	функции	animuson , Brett DeWoody , cone56 , dodopok , H. Pauwelyn , haim770 , jaredsk , khawarPK , Kobi , RamenChef , SeinopSys , TheGenie OfTruth , TylerH
54	Цвета	andreas , animuson , Arjan Einbu , Brett DeWoody , Community , cuervoo , darrylyeo , designcise , H. Pauwelyn , Jasmin Solanki , John Slegers , Kuhan , Marc , Michael Moriarty , Miro , Nathan Arthur , niyasc , RamenChef , Richard Hamilton , ScientiaEtVeritas , SeinopSys , Stewartside , user007 , Wolfgang , X-27
55	Центрирование	abaracedo , Alex Morales , Alohci , andreas , animuson , apaul , Christiaan Maks , Daniel Käfer , Devid Farinelli , Diego V , dippas , Eliran Malka , Emanuele Parisio , Euan Williams , F. Müller , Farzad YZ , Felix A J , geek1011 , insertusernamehere , JedaiCoder , jehna1 , JHS , John Slegers , Jonathan Argentiero , Kilian Stinson , Kyle Ratliff , Lambda Ninja , Lucia Bentivoglio , Luke Taylor , Madalina Taina , maho , Maxouhell , Michael_B , Mifeet , Mod Proxy , Mohammad Usman , Nathan Arthur , Nhan , o.v. , Ortomala Lokni , paaacman , Paul Kozlovitch , Praveen Kumar , Sandeep Tuniki , ScientiaEtVeritas , Sergej , Siavas , smonff , Someone , Stewartside , Sunnyok , Taylor , TylerH , web-tiki , Ze Rubeus , zeel
56	Шаблоны проектирования CSS	John Slegers