Бесплатная электронная книга

УЧУСЬ HTML

Free unaffiliated eBook created from
Stack Overflow contributors.

#html

Около

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: html

It is an unofficial and free HTML 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 HTML.

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 HTML

замечания

HTML (**H** yper **t** ext **M** arkup **L** anguage) - это XML- совместимая система аннотирования документов с помощью «тегов». Он используется специально для создания контента для веб-страниц и веб-приложений, которые затем могут передаваться по сети.

Помимо текста, текущая версия HTML поддерживает множество различных типов носителей , включая изображения и видео.

Версии

Версия	Спецификация	Дата выхода
1,0	N / A	1994-01-01
2,0	RFC 1866	1995-11-24
3,2	W3C: Спецификация HTML 3.2	1997-01-14
4,0	Спецификация W3C: HTML 4.0	1998-04-24
4,01	Спецификация W3C: HTML 4.01	1999-12-24
5	WHATWG: уровень жизни в формате HTML	2014-10-28
5,1	W3C: Спецификация HTML 5.1	2016-11-01

Examples

Привет, мир

Вступление

HTML (**H** yper **t** ext **M** arkup **L** anguage) использует систему разметки, состоящую из элементов, которые представляют собой конкретный контент. *Разметка* означает, что с HTML вы объявляете *то, что* представлено зрителю, а не *как* оно представлено. Визуальные представления определяются каскадными таблицами стилей (CSS) и реализуются браузерами. Все еще существующие элементы, которые допускают такие ,

например, `font` , «полностью устарели и не должны использоваться авторами» [1] .

HTML иногда называют языком программирования, но он не имеет логики, так что это **язык разметки** . HTML-теги предоставляют семантическое значение и удобочитаемость для содержимого на странице.

Элемент обычно состоит из открывающего тега (`<element_name>`), закрывающего тега (`</element_name>`), который содержит имя элемента, окруженное угловыми скобками, и содержимое между: `<element_name>...content...</element_name>`

Есть некоторые элементы HTML, которые не имеют закрывающего тега или любого содержимого. Они называются недействительными элементами . Элементы Void включают `<img>` , `<meta>` , `<link>` и `<input>` .

Имена элементов можно рассматривать как описательные ключевые слова для содержащегося в них контента, такие как `video` , `audio` , `table` , `footer` .

HTML-страница может состоять из потенциально сотен элементов, которые затем считываются веб-браузером, интерпретируются и визуализируются в человеческом читаемом или звуковом виде на экране.

Для этого документа важно отметить разницу между элементами и тегами:

Элементы: `video` , `audio` , `table` , `footer`

Теги: `<video>` , `<audio>` , `<table>` , `<footer>` , `</html>` , `</body>`

Взгляд элемента

Давайте разложим тег ...

Тег `<p>` представляет общий абзац.

Элементы обычно имеют открывающий тег и закрывающий тег. Начальный тег содержит имя элемента в угловых скобках (`<p>`). Закрывающий тег идентичен открывающему тегу с добавлением косой черты (`/`) между открывающей скобкой и именем элемента (`</p>`).

Затем содержимое может находиться между этими двумя тегами: `<p>This is a simple paragraph.</p>` .

Создание простой страницы

В следующем примере HTML создается простая веб-страница «Hello World» .

HTML-файлы могут быть созданы с помощью любого текстового редактора . Файлы должны быть сохранены с расширением `.html` или `.htm` [2] , чтобы быть распознанными как файлы HTML.

После создания этот файл можно открыть в любом веб-браузере.

```
<!DOCTYPE html>
<html lang="en">

    <head>
        <meta charset="UTF-8">
        <title>Hello!</title>
    </head>

    <body>
        <h1>Hello World!</h1>
        <p>This is a simple paragraph.</p>
    </body>

</html>
```

Простая разбивка страницы

Это теги, используемые в примере:

Тег	Имея в виду
`<!DOCTYPE>`	Определяет версию HTML, используемую в документе. В этом случае это HTML5. Дополнительную информацию см. В разделе « Типы доктринов» .
`<html>`	Открывает страницу. Никакая разметка не должна появляться после закрытия тега (`</html>`). Атрибут `lang` объявляет основной язык страницы с использованием кодов языка ISO (`en` для английского). Дополнительную информацию см. В разделе « Язык контента» .
`<head>`	Открывает раздел главы, который не отображается в главном окне браузера, но в основном содержит информацию *о* документе HTML,

Тег	Имея в виду
	называемом *метаданными* . Он также может содержать импорт из внешних таблиц стилей и скриптов. Закрывающий тег `</head>` .
`<meta>`	Предоставляет браузеру некоторые метаданные о документе. Атрибут `charset` объявляет кодировку символов . Современные HTML-документы должны всегда использовать UTF-8 , хотя это не является обязательным требованием. В HTML `<meta>` не требует закрывающего тега. Дополнительную информацию см. В разделе « Мета-тема» .
`<title>`	Название страницы. Текст, заключенный между этим открытием и закрывающим тегом (`</title>`), будет отображаться на вкладке страницы или в строке заголовка браузера.
`<body>`	Открывает часть документа, отображаемого для пользователей, то есть весь видимый или слышимый контент страницы. После закрытия тега `</body>` содержимое не должно добавляться.
`<h1>`	Заголовок уровня 1 для страницы. См. Заголовки для получения дополнительной информации.
`<p>`	Представляет общий абзац текста.

---

1. ↑ HTML5, 11.2 Несоответствующие функции
2. ↑ `.htm` наследуется от устаревшего ограничения расширения файла с тремя символами DOS .

Прочитайте Начало работы с HTML онлайн: https://riptutorial.com/ru/html/topic/217/начало-работы-с-html

глава 2: ARIA

Синтаксис

- ария-живой
- ария релевантных
- ария-автозаполнение
- ария проверено
- ария-инвалидов
- Aria вспененные
- ария-haspopup
- ария-скрытый
- ария-инвалида
- ария-этикетка
- ария уровня
- ария-многострочный
- ария-multiselectable
- Aria-ориентации
- ария нажиму
- ария-чтение
- ария-требуется
- Aria выбранных
- ария сортировки
- ария-valuemax
- ария-valuemin
- ария-valuenow
- ария-valuetext
- ария-атомная
- ария-занят
- ария-dropeffect
- ария-потащили
- ария-activedescendant
- Aria-контроль
- ария-describedby
- ария-FlowTo
- ария-labelledby
- ария-владеет
- ария-posinset
- ария-SetSize

замечания

ARIA - это спецификация для семантического описания богатых веб-приложений. Стандарты ARIA могут повысить доступность для тех, кто использует вспомогательные технологии (например, устройство чтения с экрана) для доступа к вашему контенту.

Examples

Роль = «тревога»

Сообщение с важной и обычно чувствительной ко времени информацией.

```
<div role="alert" aria-live="assertive">Your session will expire in 60 seconds.</div>
```

> Обратите внимание, что я включил одновременно `role="alert"` и `aria-live="assertive"` . Это синонимичные атрибуты, но некоторые читатели экрана поддерживают только тот или иной. Используя оба одновременно, мы, таким образом, максимизируем шансы, что живая область будет функционировать должным образом.
>
> Источник - Хейдон Пикеринг «Некоторые практические примеры ARIA»

Роль = "alertdialog"

Тип диалога, содержащего предупреждающее сообщение, где начальный фокус переходит к элементу в диалоговом окне.

```
<div role="alertdialog">
  <h1>Warning</h1>
  <div role="alert">Your session will expire in 60 seconds.</div>
</div>
```

Роль = «применение»

Область, объявленная как веб-приложение, в отличие от веб-документа. В этом примере приложение представляет собой простой калькулятор, который может добавить два числа вместе.

```
<div role="application">
  <h1>Calculator</h1>
  <input id="num1" type="text"> + <input id="num2" type="text"> =
  <span id="result"></span>
</div>
```

Роль = «статья»

Раздел страницы, состоящий из композиции, которая образует независимую часть документа, страницы или сайта.

> Установка роли ARIA и / или атрибута aria- *, которая соответствует неявной семантике ARIA по умолчанию, не нужна и не рекомендуется, так как эти свойства уже заданы браузером.

```
<article>
  <h1>My first article</h1>
  <p>Lorem ipsum...</p>
</article>
```

Вы должны использовать `role=article` о не семантических элементах (не рекомендуется, недействительно)

```
<div role="article">
  <h1>My first article</h1>
  <p>Lorem ipsum...</p>
</div>
```

W3C Вход для `role=article`

Роль = «баннер»

Область, которая содержит преимущественно ориентированный на сайт контент, а не контент, специфичный для страницы.

```
<div role="banner">
  <h1>My Site</h1>

  <ul>
    <li><a href="/">Home</a></li>
    <li><a href="/about">About</a></li>
    <li><a href="/contact">Contact</a></li>
  </ul>
</div>
```

Роль = «Кнопка»

Ввод, который позволяет выполнять действия, вызванные пользователем при нажатии или нажатии.

```
<button role="button">Add</button>
```

Роль = «ячейка»

Ячейка в табличном контейнере.

```
<table>
  <thead>
    <!-- etc -->
```

```
  </thead>
  <tbody>
    <td role="cell">95</td>
    <td role="cell">14</td>
    <td role="cell">25</td>
  </tbody>
</table>
```

Роль = «флажок»

Контрольный вход, который имеет три возможных значения: true, false или mixed.

```
<p>
  <input type="checkbox" role="checkbox" aria-checked="false">
  I agree to the terms
</p>
```

Роль = "ColumnHeader"

Ячейка, содержащая информацию заголовка для столбца.

```
<table role="grid">
  <thead>
    <tr>
      <th role="columnheader">Day 1</th>
      <th role="columnheader">Day 2</th>
      <th role="columnheader">Day 3</th>
    </tr>
  </thead>
  <tbody>
    <!-- etc -->
  </tbody>
<table>
```

Роль = «выпадающий»

Представление выбора; обычно похожи на текстовое поле, где пользователи могут вводить опцию, чтобы выбрать параметр, или введите для ввода произвольного текста в качестве нового элемента в списке.

```
<input type="text" role="combobox" aria-expanded="false">
```

Как правило, вы должны использовать JavaScript для создания остальных функций типа или списка.

Роль = «комплементарный»

Поддерживающий раздел документа, предназначенный для дополнения основного содержимого на аналогичном уровне в иерархии DOM, но остается значимым, когда он

отделен от основного контента.

```
<div role="complementary">
  <h2>More Articles</h2>

  <ul>
    <!-- etc -->
  </ul>
</div>
```

Роль = "contentinfo"

Большой воспринимаемый регион, содержащий информацию о родительском документе.

```
<p role="contentinfo">
  Author: Albert Einstein<br>
  Published: August 15, 1940
</p>
```

Роль = «определение»

Определение термина или понятия.

```
<span role="term" aria-labelledby="def1">Love</span>
<span id="def1" role="definition">an intense feeling of deep affection.</span>
```

Роль = «Диалог»

Диалоговое окно представляет собой окно приложения, предназначенное для прерывания текущей обработки приложения, чтобы побудить пользователя ввести информацию или потребовать ответа.

```
<div role="dialog">
  <p>Are you sure?</p>
  <button role="button">Yes</button>
  <button role="button">No</button>
</div>
```

Роль = «каталог»

Список ссылок на членов группы, например статическое оглавление.

```
<ul role="directory">
  <li><a href="/chapter-1">Chapter 1</a></li>
  <li><a href="/chapter-2">Chapter 2</a></li>
  <li><a href="/chapter-3">Chapter 3</a></li>
</ul>
```

Роль = «документ»

Область, содержащая связанную информацию, объявленную как содержимое документа, в отличие от веб-приложения.

```
<div role="document">
  <h1>The Life of Albert Einstein</h1>
  <p>Lorem ipsum...</p>
</div>
```

Роль = «форма»

Ориентированный регион, который содержит коллекцию предметов и объектов, которые в целом объединяются для создания формы.

Использование семантически корректного HTML-элемента `<form>` подразумевает семантику по умолчанию ARIA, то есть `role=form` не требуется, поскольку вы не должны применять контрастирующую роль к уже семантическому элементу, поскольку добавление роли переопределяет естественную семантику элемента.

> Установка роли ARIA и / или атрибута aria- *, которая соответствует неявной семантике ARIA по умолчанию, не нужна и не рекомендуется, так как эти свойства уже заданы браузером.

```
<form action="">
  <fieldset>
    <legend>Login form</legend>
    <div>
      <label for="username">Your username</label>
      <input type="text" id="username" aria-describedby="username-tip" required />
      <div role="tooltip" id="username-tip">Your username is your email address</div>
    </div>
    <div>
      <label for="password">Your password</label>
      <input type="text" id="password" aria-describedby="password-tip" required />
      <div role="tooltip" id="password-tip">Was emailed to you when you signed up</div>
    </div>
  </fieldset>
</form>
```

Вы должны использовать `role=form` для не семантических элементов (не рекомендуется, недействительно)

```
<div role=form>
  <input type="email" placeholder="Your email address">
  <button>Sign up</button>
</div>
```

Роль = «Сетка»

Сетка - это интерактивный элемент управления, который содержит ячейки табличных данных, упорядоченные в строках и столбцах, как таблица.

```
<table role="grid">
  <thead>
    <!-- etc -->
  </thead>
  <tbody>
    <!-- etc -->
  </tbody>
</table>
```

Роль = "GridCell"

Ячейка в сетке или древовидной.

```
<table role="grid">
  <thead>
    <!-- etc -->
  </thead>
  <tbody>
    <tr>
      <td role="gridcell">17</td>
      <td role="gridcell">64</td>
      <td role="gridcell">18</td>
    </tr>
  </tbody>
<table>
```

Роль = «группа»

Набор объектов пользовательского интерфейса, которые не предназначены для включения в сводку страниц или оглавление с помощью вспомогательных технологий.

```
<div role="group">
  <button role"button">Previous</button>
  <button role"button">Next</button>
</div>
```

Роль = «заголовок»

Заголовок для раздела страницы.

```
<h1 role="heading">Introduction</h1>
<p>Lorem ipsum...</p>
```

Роль = "IMG"

Контейнер для коллекции элементов, которые образуют изображение.

```
<figure role="img">
  <img alt="A cute cat." src="albert.jpg">
  <figcaption>This is my cat, Albert.</figcaption>
<figure>
```

Роль = «ссылка»

Интерактивная ссылка на внутренний или внешний ресурс, который при активации активирует пользовательский агент на этот ресурс.

> В большинстве случаев установка роли ARIA и / или атрибута aria- *, которая соответствует неявной семантике ARIA по умолчанию, не нужна и не рекомендуется, так как эти свойства уже заданы браузером.
>
> Источник - https://www.w3.org/TR/html5/dom.html#aria-usage-note

Роль = «список»

Группа неинтерактивных элементов списка.

```
<ul role="list">
  <li role="listitem">One</li>
  <li role="listitem">Two</li>
  <li role="listitem">Three</li>
</ul>
```

Роль = "ListBox"

Виджет, который позволяет пользователю выбирать один или несколько элементов из списка вариантов.

```
<ul role="listbox">
  <li>One</li>
  <li>Two</li>
  <li>Three</li>
</ul>
```

Как правило, вы должны использовать JavaScript для создания функции множественного выбора.

Роль = "ListItem"

Один элемент в списке или каталоге.

```
<ul role="list">
  <li role="listitem">One</li>
  <li role="listitem">Two</li>
  <li role="listitem">Three</li>
</ul>
```

Роль = «Журнал»

Тип живого региона, где новая информация добавляется в значимом порядке, и старая

информация может исчезнуть.

```
<ul role="log">
  <li>User 1 logged in.</li>
  <li>User 2 logged in.</li>
  <li>User 1 logged out.</li>
</ul>
```

Роль = «основной»

Основное содержание документа.

```
<!-- header & nav here -->
<div role="main">
  <p>Lorem ipsum...</p>
</div>
<!-- footer here -->
```

Роль = «бегущая строка»

Тип живого региона, в котором важная информация часто изменяется.

```
<ul role="marquee">
  <li>Dow +0.26%</li>
  <li>Nasdaq +0.54%</li>
  <li>S&amp;amp;P +0.44%</li>
</ul>
```

Роль = «математика»

Содержимое, представляющее собой математическое выражение.

```
<img role="math" alt="y=mx+b" src="slope.png">
```

Роль = «меню»

Тип виджета, который предлагает список вариантов для пользователя.

```
<ul role="menu">
  <li role="menuitem">New</li>
  <li role="menuitem">Open</li>
  <li role="menuitem">Save</li>
  <li role="menuitem">Close</li>
</ul>
```

Роль = "Menubar"

Представление меню, которое обычно остается видимым и обычно представлено горизонтально.

```
<ul role="menubar">
  <li role="menuitem">File</li>
  <li role="menuitem">Edit</li>
  <li role="menuitem">View</li>
  <li role="menuitem">Help</li>
</ul>
```

Роль = "MENUITEM"

Опция в группе вариантов, содержащихся в меню или в строке меню.

```
<ul role="menubar">
  <li role="menuitem">File</li>
  <li role="menuitem">Edit</li>
  <li role="menuitem">View</li>
  <li role="menuitem">Help</li>
</ul>
```

Роль = "menuitemcheckbox"

Контролируемый элемент меню, который имеет три возможных значения: true, false или mixed.

```
<ul role="menu">
  <li role="menuitem">Console</li>
  <li role="menuitem">Layout</li>
  <li role="menuitemcheckbox" aria-checked="true">Word wrap</li>
</ul>
```

Роль = "menuitemradio"

Контролируемое меню в группе ролей menuitemradio, только один из которых может быть проверен одновременно.

```
<ul role="menu">
  <li role="menuitemradio" aria-checked="true">Left</li>
  <li role="menuitemradio" aria-checked="false">Center</li>
  <li role="menuitemradio" aria-checked="false">Right</li>
</ul>
```

Роль = «навигация»

Набор навигационных элементов (обычно ссылок) для навигации по документу или связанным документам.

```
<ul role="navigation">
  <li><a href="/">Home</a></li>
  <li><a href="/about">About</a></li>
  <li><a href="/contact">Contact</a></li>
</ul>
```

Роль = «примечание»

Раздел, содержимое которого является постоянным или вспомогательным для основного содержимого ресурса.

```
<p>Lorem ipsum...</p>
<p>Lorem ipsum...</p>
<p role="note">Lorem ipsum...</p>
```

Роль = «вариант»

Выбираемый элемент в списке выбора.

```
<ul role="listbox">
  <li role="option">Option 1</li>
  <li role="option">Option 2</li>
  <li role="option">Option 3</li>
</ul>
```

Роль = «презентация»

Элемент, чья семантика неявной родной роли не будет сопоставлена с API доступности.

```
<div style="float:left;">Some content on the left.</div>
<div style="float:right;">Some content on the right</div>
<div role="presentation" style="clear:both;"></div> <!-- Only used to clear floats -->
```

Роль = "ProgressBar"

Элемент, который отображает статус выполнения для задач, которые занимают много времени.

```
<progress role="progressbar" value="25" max="100">25%</progress>
```

Роль = «радио»

Проверяемый ввод в группе ролей радио, только один из которых может быть проверен одновременно.

```
<div role="radiogroup">
  <input role="radio" type="radio" aria-checked="true"> One<br>
  <input role="radio" type="radio" aria-checked="false"> Two<br>
  <input role="radio" type="radio" aria-checked="false"> Three
</div>
```

Роль = «область»

Большой воспринимаемый раздел веб-страницы или документа, который, по мнению автора, достаточно важен для включения в сводку или оглавление страницы, например, область страницы, содержащую статистику спортивных событий в реальном времени.

```
<div role="region">
  Home team: 4<br>
  Away team: 2
</div>
```

Роль = "RadioGroup"

Группа переключателей.

```
<div role="radiogroup">
  <input role="radio" type="radio" aria-checked="true"> One<br>
  <input role="radio" type="radio" aria-checked="false"> Two<br>
  <input role="radio" type="radio" aria-checked="false"> Three
</div>
```

Роль = «строка»

Ряд ячеек в табличном контейнере.

```
<table>
  <thead>
    <!-- etc -->
  </thead>
  <tbody>
    <tr role="row">
      <!-- etc -->
    </tr>
  </tbody>
</table>
```

Роль = "rowgroup"

Группа, содержащая один или несколько элементов строки в сетке.

```
<table>
  <thead role="rowgroup">
    <!-- etc -->
  </thead>
  <tbody role="rowgroup">
    <!-- etc -->
  </tbody>
</table>
```

Роль = "rowheader"

Ячейка, содержащая информацию заголовка для строки в сетке.

```
<table role="grid">
  <thead>
    <!-- etc -->
  </thead>
  <tbody>
    <tr>
      <th role="rowheader">Day 1</th>
      <td>65</td>
    </tr>
    <tr>
      <th role="rowheader">Day 2</th>
      <td>74</td>
    </tr>
  </tbody>
</table>
```

Роль = «полоса прокрутки»

Графический объект, который управляет прокруткой содержимого в пределах области просмотра, независимо от того, полностью ли он отображается в области просмотра.

```
<div id="content1">Lorem ipsum...</div>
<div
  role="scrollbar"
  aria-controls="content1"
  aria-orientation="vertical"
  aria-valuemax="100"
  aria-valuemin="0"
  aria-valuenow="25">
    <div class="scrollhandle"></div>
</div>
```

Роль = «Поиск»

Ориентированный регион, который содержит коллекцию предметов и объектов, которые в целом объединяются для создания объекта поиска.

```
<div role="search">
  <input role="searchbox" type="text">
  <button role="button">Search</button>
</div>
```

Роль = "SearchBox"

Тип текстового поля, предназначенного для указания критериев поиска.

```
<div role="search">
  <input role="searchbox" type="text">
  <button role="button">Search</button>
</div>
```

Роль = «разделитель»

Разделитель, который разделяет и выделяет разделы содержимого или группы элементов меню.

```
<p>Lorem ipsum...</p>
<hr role="separator">
<p>Lorem ipsum...</p>
```

Роль = «слайдер»

Пользовательский ввод, в котором пользователь выбирает значение из заданного диапазона.

```
<div
  role="slider"
  aria-valuemax="100"
  aria-valuemin="0"
  aria-valuenow="25">
    <div class="sliderhandle"></div>
</div>
```

Роль = «в полях ввода»

Форма диапазона, которая ожидает, что пользователь будет выбирать из дискретных вариантов.

```
<input
  role="spinbutton"
  aria-valuemax="100"
  aria-valuemin="0"
  aria-valuenow="25"
  type="number"
  value="25">
```

Роль = «Статус»

Контейнер, содержимое которого является консультативной информацией для пользователя, но не является достаточно важным, чтобы оправдывать оповещение, часто, но не обязательно представляемое в виде строки состояния.

```
<div role="status">Online</div>
```

Роль = «переключатель»

Тип флажка, который представляет значения включения / выключения, в отличие от отмеченных / непроверенных значений.

```
<select role="switch" aria-checked="false">
  <option>On</option>
```

```
  <option selected>Off</option>
</select>
```

Роль = «закладка»

Метка группировки, обеспечивающая механизм выбора содержимого вкладки, которое должно быть передано пользователю.

```
<ul role="tablist">
  <li role="tab">Introduction</li>
  <li role="tab">Chapter 1</li>
  <li role="tab">Chapter 2</li>
</ul>
```

Роль = «таблица»

Раздел, содержащий данные, расположенные в строках и столбцах. Роль таблицы предназначена для табличных контейнеров, которые не являются интерактивными.

```
<table role="table">
  <thead>
    <!-- etc -->
  </thead>
  <tbody>
    <!-- etc -->
  </tbody>
</table>
```

Роль = "tablist"

Список элементов табуляции, которые являются ссылками на элементы tabpanel.

```
<ul role="tablist">
   <li role="tab">Introduction</li>
   <li role="tab">Chapter 1</li>
   <li role="tab">Chapter 2</li>
</ul>
```

Роль = "tabpanel"

Контейнер для ресурсов, связанных с вкладкой, где каждая вкладка содержится в списке вкладок.

```
<ul role="tablist">
  <li role="tab">Introduction</li>
  <li role="tab">Chapter 1</li>
  <li role="tab">Chapter 2</li>
</ul>
<div role="tabpanel">
  <!-- etc -->
```

```
</div>
```

Роль = «текстовое поле»

Ввод, который позволяет использовать текст свободной формы в качестве значения.

```
<textarea role="textbox"></textarea>
```

Роль = «Таймер»

Тип живой области, содержащей числовой счетчик, который указывает количество прошедшего времени от начальной точки или время, оставшееся до конечной точки.

```
<p>
  <span role="timer">60</span> seconds remaining.
</p>
```

Роль = «Панель инструментов»

Набор широко используемых функциональных кнопок, представленных в компактной визуальной форме.

```
<ul role="toolbar">
  <li><img alt="New" src="new.png"></li>
  <li><img alt="Open" src="open.png"></li>
  <li><img alt="Save" src="save.png"></li>
  <li><img alt="Close" src="close.png"></li>
</ul>
```

Роль = «подсказка»

Контекстное всплывающее окно, отображающее описание элемента.

```
<span aria-describedby="slopedesc">Slope</span>
<div role="tooltip" id="slopedesc">y=mx+b</div>
```

Как правило, всплывающая подсказка будет скрыта. Используя JavaScript, всплывающая подсказка будет отображаться после задержки, когда пользователь наводится на элемент, который он описывает.

Роль = «дерево»

Тип списка, который может содержать вложенные группы под-уровня, которые могут быть свернуты и расширены.

```
<ul role="tree">
```

```
  <li role="treeitem">
    Part 1
    <ul>
      <li role="treeitem">Chapter 1</li>
      <li role="treeitem">Chapter 2</li>
      <li role="treeitem">Chapter 3</li>
    </ul>
  </li>
  <li role="treeitem">
    Part 2
    <ul>
      <li role="treeitem">Chapter 4</li>
      <li role="treeitem">Chapter 5</li>
      <li role="treeitem">Chapter 6</li>
    </ul>
  </li>
  <li role="treeitem">
    Part 3
    <ul>
      <li role="treeitem">Chapter 7</li>
      <li role="treeitem">Chapter 8</li>
      <li role="treeitem">Chapter 9</li>
    </ul>
  </li>
</ul>
```

Роль = "TreeGrid"

Сетка, строки которой можно развернуть и свернуть так же, как для дерева.

Роль = "TreeItem"

Элемент опции дерева. Это элемент внутри дерева, который может быть расширен или свернут, если он содержит группу подэлементов.

```
<ul role="tree">
  <li role="treeitem">
    Part 1
    <ul>
      <li role="treeitem">Chapter 1</li>
      <li role="treeitem">Chapter 2</li>
      <li role="treeitem">Chapter 3</li>
    </ul>
  </li>
  <li role="treeitem">
    Part 2
    <ul>
      <li role="treeitem">Chapter 4</li>
      <li role="treeitem">Chapter 5</li>
      <li role="treeitem">Chapter 6</li>
    </ul>
  </li>
  <li role="treeitem">
    Part 3
    <ul>
      <li role="treeitem">Chapter 7</li>
      <li role="treeitem">Chapter 8</li>
```

```
      <li role="treeitem">Chapter 9</li>
    </ul>
  </li>
</ul>
```

Прочитайте ARIA онлайн: https://riptutorial.com/ru/html/topic/2734/aria

глава 3: DOCTYPEs

Вступление

Doctypes - сокращение от типа документа - помогает браузерам понять версию HTML, в которой документ написан для лучшей интерпретируемости. Объявления Doctype не являются тегами HTML и находятся в самом верху документа. В этом разделе объясняется структура и декларация различных доктринов в HTML.

Синтаксис

- <! DOCTYPE [строка для версии]>

замечания

Объявление `<!DOCTYPE>` не является тегом HTML. Он используется для указания версии HTML, которую использует документ. Это называется объявлением типа документа (DTD).

Объявление `<!DOCTYPE>` не чувствительно к регистру. Чтобы проверить, действителен ли HTML-код ваших веб-страниц, перейдите в службу проверки W3C .

- Некоторые старые версии IE не поддерживают некоторые теги HTML, если не имеется подходящий doctype.
- Очень *важно,* чтобы объявлялся doctype, чтобы убедиться, что браузер не использует режим quirks. Дополнительная информация о MDN.

Examples

Добавление Doctype

Объявление `<!DOCTYPE>` должно всегда включаться в начало документа HTML, перед `<html>` .

5

См HTML 5 Doctype для подробной информации о HTML 5 DOCTYPE.

```
<!DOCTYPE html>
```

4,01

См. HTML 4.01 Doctypes для получения подробной информации о том, как эти типы отличаются друг от друга.

строгий

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
```

переходный

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

Frameset

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
"http://www.w3.org/TR/html4/frameset.dtd">
```

HTML 4.01 Doctypes

Спецификация HTML 4.01 предоставляет несколько различных типов доктринов, которые позволяют указывать в документе различные типы элементов.

HTML 4.01 Строгий

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
```

Включает все элементы и атрибуты HTML, но **не включает в себя элементы презентации или устаревшие элементы** и **фреймы** .

HTML 4.01 Переходный

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

Включает все элементы и атрибуты HTML, а также устаревшие и устаревшие элементы, но **фреймы не допускаются** .

HTML 4.01 Frameset

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
"http://www.w3.org/TR/html4/frameset.dtd">
```

Включает все элементы и атрибуты HTML, презентационные и устаревшие элементы. Рамки разрешены.

HTML 5 Doctype

HTML5 не основан на SGML и поэтому не требует ссылки на DTD.

HTML 5 Объявление Doctype:

```
<!DOCTYPE html>
```

Нечувствительность к регистру

Per the W3.org HTML 5 `DOCTYPE` Spec :

> DOCTYPE должен состоять из следующих компонентов в следующем порядке:
>
> 1. Строка , которая является ASCII **регистронезависимым** матч для строки `"<!DOCTYPE"` .

поэтому действуют также следующие `DOCTYPE` :

```
<!doctype html>
<!dOCtyPe html>
<!DocTYpe html>
```

В этой статье подробно обсуждается тема: верхний или нижний регистр документа?

Старые Doctypes

HTML 3.2

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
```

HTML 3.2 хорошо поддерживается большинством используемых браузеров. Тем не менее, HTML 3.2 имеет ограниченную поддержку таблиц стилей и не поддерживает функции HTML 4, такие как фреймы и интернационализацию.

HTML 2.0

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
```

HTML 2.0 широко поддерживается браузерами, но не имеет поддержки таблиц, фреймов и интернационализации, а также многих часто используемых элементов и атрибутов презентации.

Прочитайте DOCTYPEs онлайн: https://riptutorial.com/ru/html/topic/806/doctypes

глава 4: HTML 5 кэш

замечания

Файл манифеста представляет собой простой текстовый файл, который сообщает браузеру, что кешировать (и что никогда не кэшировать). Рекомендуемое расширение файла для файлов манифеста: «.appcache» Файл манифеста состоит из трех разделов:

CACHE MANIFEST - Файлы, перечисленные в этом заголовке, будут кэшироваться после их загрузки в первый раз

NETWORK - файлы, перечисленные в этом заголовке, требуют подключения к серверу и никогда не будут кэшироваться

FALLBACK - файлы, перечисленные в этом заголовке, указывают резервные страницы, если страница недоступна

Examples

Основной пример кэша Html 5

это наш файл index.html

```
<!DOCTYPE html>
<html manifest="index.appcache">
<body>
    <p>Content</p>
</body>
</html>
```

то мы создадим файл index.appcache с нижеследующими кодами

```
CACHE MANIFEST
index.html
```

напишите те файлы, которые вы хотите загрузить в кеш-загрузку index.html, затем перейдите в автономный режим и перезагрузите вкладку

Примечание. Эти два файла должны находиться в одной папке в этом примере.

Прочитайте HTML 5 кэш онлайн: https://riptutorial.com/ru/html/topic/8024/html-5-кэш

глава 5: IFrames

параметры

атрибут	подробности
`name`	Задает имя элемента, который будет использоваться с `a` теге , чтобы изменить Iframe в `src` .
`width`	Устанавливает ширину элемента в пикселях.
`height`	Устанавливает высоту элемента в пикселях.
`src`	Задает страницу, которая будет отображаться в кадре.
`srcdoc`	Задает контент, который будет отображаться в кадре, при условии, что браузер поддерживает его. Содержимое должно быть действительным HTML.
`sandbox`	Когда установлено, содержимое iframe рассматривается как уникальное происхождение, и функции, включая скрипты, плагины, формы и всплывающие окна, будут отключены. Ограничения могут быть выборочно расслаблены, добавляя список значений, разделенных пробелом. См. Таблицу в примечаниях для возможных значений.
`allowfullscreen`	Разрешить ли содержимому iframe использовать `requestFullscreen()`

замечания

IFrame используется для встраивания другого документа в текущий HTML-документ.

Вы можете использовать iframes для отображения:

- другие HTML-страницы в одном домене;
- другие HTML-страницы в другом домене (см. ниже - политика одинакового происхождения);
- PDF-документы (хотя IE может иметь некоторые проблемы, этот вопрос может помочь);

Вы ДОЛЖНЫ использовать iframe в качестве последнего средства, поскольку у него есть проблемы с закладок и навигации, и всегда есть лучшие варианты, отличные от iframe. Этот вопрос должен помочь вам лучше понять взлеты и падения iframes.

Политика же происхождения

Некоторые сайты не могут отображаться с использованием iframe, поскольку они применяют политику, называемую политикой «один и тот же источник» . Это означает, что сайт, на котором лежит iframe, должен находиться в том же домене, что и отображаемый.

Эта политика также применяется для управления содержимым, которое находится внутри iFrame. Если iFrame обращается к содержимому из другого домена, вы не сможете получить доступ к содержимому внутри iFrame или манипулировать им.

Элемент iframe на W3C

атрибут `sandbox`

Атрибут `sandbox` при установке добавляет дополнительные ограничения для iframe. Для ослабления этих ограничений можно использовать список токенов, разделенных пробелами.

Значение	подробности
`allow-forms`	Позволяет отправлять формы.
`allow-pointer-lock`	Включает API-интерфейс JavaScript-указателя.
`allow-popups`	Всплывающие окна могут быть созданы с помощью `window.open` или `<a target="_blank"`
`allow-same-origin`	В документе iframe используется реальное происхождение, а не уникальное. Если используется с `allow-scripts` документ iframe может удалить всю песочницу, если она имеет тот же источник, что и родительский документ.
`allow-scripts`	Включает скрипты. Документ iframe и родительский документ могут взаимодействовать друг с другом с помощью API `postMessage()` . Если используется с `allow-same-origin` документ iframe может удалить всю песочницу, если она имеет тот же источник, что и исходный документ.
`allow-top-navigation`	Позволяет содержимому iframe изменять местоположение документа верхнего уровня.

Examples

Основы встроенной структуры

Термин «IFrame» означает «Встроенный кадр». Его можно использовать для включения другой страницы на вашу страницу. Это даст небольшой кадр, который показывает точное содержимое `base.html` .

```
<iframe src="base.html"></iframe>
```

Установка размера рамки

IFrame может быть изменен с использованием атрибутов `width` и `height` , где значения представлены в пикселях (HTML 4.01 разрешено процентное значение, но HTML 5 допускает только значения в пикселях CSS).

```
<iframe src="base.html" width="800" height="600"></iframe>
```

Использование якорей с IFrames

Обычно изменение веб-страницы в iframe инициируется с помощью Iframe, например, щелкая ссылку внутри Ifame. Тем не менее, можно изменить содержимое IFrame из-за пределов IFrame. Вы можете использовать якорь тег , чье `href` атрибут установлен желаемый URL и чей `target` атрибут установлен в Iframe по `name` атрибута.

```
<iframe src="webpage.html" name="myIframe"></iframe>
<a href="different_webpage.html" target="myIframe">Change the Iframe content to
different_webpage.html</a>
```

Использование атрибута «srcdoc»

`srcdoc` может использоваться (вместо атрибута `src`) для указания точного содержимого iframe как целого HTML-документа. Это даст IFrame с текстом «IFrames классные!».

```
<iframe srcdoc="<p>IFrames are cool!</p>"></iframe>
```

Если атрибут `srcdoc` не поддерживается браузером, IFrame вместо этого вернется к использованию атрибута `src` , но если оба атрибута `src` и `srcdoc` присутствуют и поддерживаются браузером, `srcdoc` имеет приоритет.

```
<iframe srcdoc="<p>Iframes are cool!</p>" src="base.html"></iframe>
```

В приведенном выше примере, если браузер не поддерживает атрибут `srcdoc` , он будет отображать содержимое страницы `base.html` .

Песочница

Следующие внедряют ненадежную веб-страницу с включенными ограничениями

```
<iframe sandbox src="http://example.com/"></iframe>
```

Чтобы страница запускала сценарии и отправляла формы, добавьте `allow-scripts` и `allow-forms` к `sandbox attribute` .

```
<iframe sandbox="allow-scripts allow-forms" src="http://example.com/"></iframe>
```

Если есть ненадежный контент (например, комментарии пользователей) в том же домене, что и родительская веб-страница, iframe может использоваться для отключения сценариев, в то же время позволяя родительскому документу взаимодействовать с его содержимым с помощью JavaScript.

```
<iframe sandbox="allow-same-origin allow-top-navigation"
src="http://example.com/untrusted/comments/page2">
```

Родительский документ может добавлять прослушиватели событий и изменять размер IFrame в соответствии с его содержимым. Это, наряду с `allow-top-navigation` , может сделать изолированный iframe, как представляется, частью родительского документа.

Эта песочница не является заменой для дезинфекции, но может использоваться как часть стратегии защиты в глубину .

Также имейте в виду, что эта песочница может быть подорвана злоумышленником, убеждая пользователя напрямую посетить источник iframe. Для уменьшения этой атаки можно использовать HTTP-заголовок политики безопасности контента .

Прочитайте IFrames онлайн: https://riptutorial.com/ru/html/topic/499/iframes

глава 6: SVG

Вступление

SVG означает масштабируемую векторную графику. SVG используется для определения графики для Интернета

Элемент HTML `<svg>` является контейнером для графики SVG.

SVG имеет несколько способов рисования путей, ящиков, кругов, текста и графических изображений.

замечания

SVG - это язык на основе XML для создания масштабируемых векторных изображений. Он может быть написан непосредственно в HTML-документе или встроен из внешних SVG-файлов. Inline SVG может быть изменен и изменен с использованием CSS и JavaScript соответственно.

Поддержка браузеров для SVG варьируется, но может быть установлена здесь .

Более подробную информацию см. В документации по SVG .

Examples

Встраивание внешних SVG-файлов в HTML

Вы можете использовать элементы `<img>` или `<object>` для встраивания внешних элементов SVG. Установка высоты и ширины является необязательной, но настоятельно рекомендуется.

Использование элемента изображения

```
<img src="attention.svg" width="50" height="50">
```

Использование `<img>` не позволяет вам стилировать SVG с помощью CSS или манипулировать им с помощью JavaScript.

Использование элемента объекта

```
<object type="image/svg+xml" data="attention.svg" width="50" height="50">
```

В отличие от `<img>` , `<object>` напрямую импортирует SVG в документ, и поэтому его можно манипулировать с помощью Javascript и CSS.

Встроенный SVG

SVG можно записать непосредственно в HTML-документ. Встроенный SVG можно создавать и манипулировать с помощью CSS и JavaScript.

```
<body>
    <svg class="attention" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink" viewBox="0 0 1000 1000" >
        <path id="attention" d="m571,767l0,-106q0,-8,-5,-13t-12,-5l-108,0q-7,0,-12,5t-
5,13l0,106q0,8,5,13t12,6l108,0q7,0,12,-6t5,-13Zm-1,-208l10,-257q0,-6,-5,-10q-7,-6,-14,-6l-
122,0q-7,0,-14,6q-5,4,-5,12l9,255q0,5,6,9t13,3l103,0q8,0,13,-3t6,-9Zm-7,-522l428,786q20,35,-
1,70q-10,17,-26,26t-35,10l-858,0q-18,0,-35,-10t-26,-26q-21,-35,-1,-70l429,-786q9,-17,26,-
27t36,-10t36,10t27,27Z" />
    </svg>
</body>
```

Вышеуказанный встроенный SVG можно затем использовать с помощью соответствующего класса CSS:

```
.attention {
    fill: red;
    width: 50px;
    height: 50px;
}
```

Результат выглядит следующим образом:

Внедрение SVG с использованием CSS

Вы можете добавить внешние SVG-файлы, используя свойство `background-image` , так же, как и с любым другим изображением.

HTML:

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

CSS:

```
.attention {
    background-image: url(attention.svg);
    background-size: 100% 100%;
    width: 50px;
    height: 50px;
}
```

Вы также можете встроить изображение непосредственно в файл css, используя URL-адрес данных:

```
background-image:
url(data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20xmlns%3Axlink%3D%2
106q0%2C-8%2C-5%2C-13t-12%2C-5l-108%2C0q-7%2C0%2C-12%2C5t-
5%2C13l0%2C106q0%2C8%2C5%2C13t12%2C6l108%2C0q7%2C0%2C12%2C-6t5%2C-13Zm-1%2C-208l10%2C-
257q0%2C-6%2C-5%2C-10q-7%2C-6%2C-14%2C-6l-122%2C0q-7%2C0%2C-14%2C6q-5%2C4%2C-
5%2C12l9%2C255q0%2C5%2C6%2C9t13%2C3l103%2C0q8%2C0%2C13%2C-3t6%2C-9Zm-7%2C-
522l428%2C786q20%2C35%2C-1%2C70q-10%2C17%2C-26%2C26t-35%2C10l-858%2C0q-18%2C0%2C-35%2C-10t-
26%2C-26q-21%2C-35%2C-1%2C-70l429%2C-786q9%2C-17%2C26%2C-27t36%2C-
10t36%2C10t27%2C27Z%22%20%2F%3E%0D%0A%3C%2Fsvg%3E);
```

Прочитайте SVG онлайн: https://riptutorial.com/ru/html/topic/1183/svg

глава 7: TabIndex

параметры

Значение	Имея в виду
отрицательный	элемент будет фокусируемым, но он не должен быть доступен через последовательную навигацию по клавиатуре
0	элемент будет настраиваться и доступен через последовательную навигацию клавиатуры, но относительный порядок определяется соглашением платформы
положительный	элемент должен быть настраиваемым и доступным с помощью последовательной навигации по клавиатуре; это относительный порядок будет определяться значением атрибута: последовательные следуют за увеличением числа `tabindex`

замечания

Максимальное значение для `tabindex` не должно превышать 32767 согласно разделу W3C 17.11.1 Если указанное значение по умолчанию не равно -1

Элемент со значением 0, недопустимое значение или значение `tabindex` должно быть размещено после элементов с положительным индексом в последовательном порядке навигации по клавиатуре.

Examples

Добавить элемент в список табуляции

```
<div tabindex="0">Some button</div>
```

Примечание: Попробуйте использовать родной HTML `button` или `a` тег , где это необходимо.

Удалить элемент из порядка табуляции

```
<button tabindex="-1">This button will not be reachable by tab</button>
```

Элемент будет удален из порядка табуляции, но все равно будет фокусироваться.

Определите пользовательский порядок табуляции (не рекомендуется)

```
<div tabindex="2">Second</div>
<div tabindex="1">First</div>
```

Положительные значения вставляют элемент в позицию порядка табуляции соответствующего значения. Элементы без предпочтений (т.е. `tabindex="0"` или собственные элементы, такие как `button` и `a`) будут добавляться после тех, кто предпочитает.

Положительные значения **не рекомендуются, так** как они нарушают ожидаемое поведение табуляции и могут смущать людей, которые полагаются на устройства чтения с экрана. Попробуйте создать естественный порядок, изменив структуру DOM.

Прочитайте TabIndex онлайн: https://riptutorial.com/ru/html/topic/2594/tabindex

глава 8: Анкеры и гиперссылки

Вступление

Якорные теги обычно используются для связывания отдельных веб-страниц, но их также можно использовать для связи между разными местами в одном документе, часто в оглавлении или даже при запуске внешних приложений. В этом разделе объясняется реализация и применение тегов привязки HTML в разных ролях.

Синтаксис

- `<a href="URL or anchor">Link Text</a>`

параметры

параметр	подробности
`href`	Указывает адрес назначения. Это может быть абсолютный или относительный URL-адрес или `name` якоря. Абсолютный URL - это полный URL-адрес веб-сайта, например http://example.com/ . Относительный URL-адрес указывает на другой каталог и / или документ внутри одного и того же веб-сайта, например `/about-us/` указывает на каталог «about-us» внутри корневого каталога (`/`). При указании на другой каталог без явного указания документа веб-серверы обычно возвращают документ « index.html» внутри этого каталога.
`hreflang`	Указывает язык ресурса, связанный атрибутом `href` (который должен присутствовать вместе с этим). Используйте языковые значения из BCP 47 для HTML5 и RFC 1766 для HTML 4.
`rel`	Задает связь между текущим документом и связанным документом. Для HTML5 значения должны быть определены в спецификации или зарегистрированы в вики-среде Microformats .
`target`	Определяет, где можно открыть ссылку, например, в новой вкладке или окне. Возможные значения: `_blank` , `_self` , `_parent` , `_top` и `framename` (устарели). Принудительное такое поведение не рекомендуется, так как оно нарушает контроль над пользователем через веб-сайт.
`title`	Указывает дополнительную информацию о ссылке. Эта информация чаще всего отображается в виде текста всплывающей подсказки, когда курсор перемещается по ссылке. Этот атрибут не ограничивается ссылками, он

параметр	подробности
	может использоваться почти во всех HTML-тегах.
`download`	Указывает, что цель будет загружаться, когда пользователь нажимает на гиперссылку. Значение атрибута будет именем загруженного файла. Нет ограничений на допустимые значения, и браузер автоматически обнаружит правильное расширение файла и добавит его в файл (.img, .pdf и т. Д.). Если значение опущено, используется исходное имя файла.

Examples

Ссылка на другой сайт

Это основное использование `<a>` **(a** nchor элемент) элемента:

```
<a href="http://example.com/">Link to example.com</a>
```

Он создает гиперссылку на URL `http://example.com/` как указано атрибутом `href` (гипертекстовая ссылка), с текстом привязки «Ссылка на example.com». Это будет выглядеть примерно так:

Ссылка на example.com

Чтобы обозначить, что эта ссылка приводит к внешнему веб-сайту, вы можете использовать `external` тип ссылки:

```
<a href="http://example.com/" rel="external">example site</a>
```

Вы можете ссылаться на сайт, который использует протокол, отличный от HTTP. Например, чтобы ссылаться на FTP-сайт, вы можете сделать это,

```
<a href="ftp://example.com/">This could be a link to a FTP site</a>
```

В этом случае разница заключается в том, что этот тег привязки запрашивает, чтобы браузер пользователя подключался к `example.com` с использованием протокола передачи файлов (FTP), а не протокола передачи гипертекста (HTTP).

Это может быть ссылка на FTP-сайт

Открыть ссылку в новой вкладке / окне

```
<a href="example.com" target="_blank">Text Here</a>
```

Атрибут `target` указывает, где можно открыть ссылку. Установив его на `_blank` , вы сообщаете браузеру открыть его на новой вкладке или в окне (для каждого пользователя).

> **УЯЗВИМОСТЬ БЕЗОПАСНОСТИ ПРЕДУПРЕЖДЕНИЕ!**
>
> Использование `target="_blank"` дает открытому сайту частичный доступ к объекту `window.opener` через JavaScript, который позволяет этой странице затем получить доступ и изменить `window.opener.location` *вашей* страницы и потенциально перенаправить пользователей на вредоносные или фишинг-сайты.
>
> Всякий раз, когда вы используете это для страниц, которые вы не контролируете, добавьте `rel="noopener"` в вашу ссылку, чтобы предотвратить `window.opener` объекта `window.opener` с запросом.
>
> В настоящее время Firefox не поддерживает `noopener` , поэтому вам нужно будет использовать `rel="noopener noreferrer"` для максимального эффекта.

Ссылка на якорь

Якоря могут использоваться для перехода к определенным тегам на странице HTML. Тег `<a>` может указывать на любой элемент с атрибутом `id` . Подробнее о идентификаторах читайте в документации по классам и идентификаторам . Якоря в основном используются для перехода к подразделу страницы и используются в сочетании с тегами заголовков.

Предположим, вы создали страницу (`page1.html`) по многим темам:

```
<h2>First topic</h2>
<p>Content about the first topic</p>
<h2>Second topic</h2>
<p>Content about the second topic</p>
```

После того, как у вас есть несколько разделов, вы можете создать Оглавление в верхней части страницы с быстрыми ссылками (или закладками) к определенным разделам.

Если вы предоставили атрибут `id` своим темам, вы можете ссылаться на них

```
<h2 id="Topic1">First topic</h2>
<p>Content about the first topic</p>
<h2 id="Topic2">Second topic</h2>
<p>Content about the second topic</p>
```

Теперь вы можете использовать якорь в своем оглавлении:

```
<h1>Table of Contents</h1>
    <a href='#Topic1'>Click to jump to the First Topic</a>
    <a href='#Topic2'>Click to jump to the Second Topic</a>
```

Эти привязки также прикреплены к веб-странице, на которой они находятся (`page1.html`). Таким образом, вы можете связывать сайт с одной страницы на другую, ссылаясь на страницу *и* имя привязки.

```
 Remember, you can always <a href="page1.html#Topic1">look back in the First Topic</a> for supporting information.
```

Ссылка, которая запускает JavaScript

Просто используйте протокол `javascript:` для запуска текста как JavaScript вместо того, чтобы открывать его как обычную ссылку:

```
<a href="javascript:myFunction();">Run Code</a>
```

Вы также можете добиться того же, используя атрибут `onclick` :

```
<a href="#" onclick="myFunction(); return false;">Run Code</a>
```

`return false;` необходимо, чтобы ваша страница не прокручивалась вверх, когда нажимается ссылка на `#` . Обязательно включите весь код, который вы хотите запустить до него, так как возвращение прекратит выполнение дополнительного кода.

Также следует отметить восклицательный знак `!` после hashtag, чтобы страница не прокручивалась вверх. Это работает, потому что любой недопустимый slug приведет к тому, что ссылка не будет прокручиваться *нигде* на странице, потому что она не может найти элемент, который он ссылается (элемент с `id="!"`). Вы также можете использовать любой недопустимый `#scrollsNowhere` (например, `#scrollsNowhere`) для достижения такого же эффекта. В этом случае `return false;` не требуется:

```
<a href="#!" onclick="myFunction();">Run Code</a>
```

> **Если вы используете что-либо из этого?**
>
> Ответ почти наверняка ***нет*** . Выполнение JavaScript внутри строки с таким элементом, как это, является довольно плохой практикой. Подумайте об использовании чистых JavaScript-решений, которые ищут элемент на странице и привязывают к нему функцию. Прослушивание события
>
> Также подумайте, действительно ли этот элемент является *кнопкой* вместо *ссылки* . Если это так, вы должны использовать `<button>` .

Ссылка на страницу на том же сайте

Вы можете использовать относительный путь для ссылки на страницы на одном сайте.

```
<a href="/example">Text Here</a>
```

Вышеприведенный пример перейдет к `example` файла в корневом каталоге (`/`) сервера.

Если эта ссылка была на http://example.com , следующие две ссылки приведут пользователя к тому же местоположению

```
<a href="/page">Text Here</a>
<a href="http://example.com/page">Text Here</a>
```

Оба вышеперечисленного перейдут к файлу `page` в корневом каталоге `example.com` .

Ссылка, которая запускает почтовый клиент

Основное использование

Если значение `href` Attribute начинается с `mailto:` он попытается открыть почтовый клиент по клику:

```
<a href="mailto:example@example.com">Send email</a>
```

Это поместит адрес электронной почты `example@example.com` в качестве получателя для вновь созданного письма.

Cc и Bcc

Вы также можете добавлять адреса для cc- или bcc-получателей, используя следующий синтаксис:

```
<a href="mailto:example@example.com?cc=john@example.com&bcc=jane@example.com">Send email</a>
```

Тема и основной текст

Вы также можете заполнить тему и тело для нового письма:

```
<a href="mailto:example@example.com?subject=Example+subject&body=Message+text">Send email</a>
```

Эти значения должны быть закодированы в URL .

Нажав на ссылку с `mailto:` попытается открыть почтовый клиент по умолчанию, указанный в вашей операционной системе, или попросит вас выбрать, какого клиента вы хотите использовать. Не все параметры, указанные после адреса получателя, поддерживаются во всех почтовых клиентах.

Ссылка, набирающая номер

Если значение `href` Attribute начинается с `tel:` ваше устройство набирает номер при нажатии на него. Это работает на мобильных устройствах или на компьютерах / планшетах, работающих под управлением программного обеспечения - например, Skype или FaceTime, - которые могут совершать телефонные звонки.

```
<a href="tel:11234567890">Call us</a>
```

Большинство устройств и программ подскажут пользователю каким-то образом подтвердить номер, который они собираются набрать.

Прочитайте Анкеры и гиперссылки онлайн: https://riptutorial.com/ru/html/topic/254/анкеры-и-гиперссылки

глава 9: Атрибуты данных

Синтаксис

- `<element data-custom-name="somevalue">`

параметры

Значение	Описание
SomeValue	Задает значение атрибута (в виде строки)

Examples

Использование атрибута данных

HTML5 `data-*` атрибуты обеспечивают удобный способ для хранения данных в HTML - элементах. Сохраненные данные могут быть прочитаны или изменены с помощью JavaScript

```
<div data-submitted="yes" class="user_profile">
  … some content …
</div>
```

- Структура атрибута `data-*` - `data-*` , т.е. имя атрибута data появляется после части `data-` . Используя это имя, можно получить доступ к атрибуту.
- Данные в формате строки (включая `json`) могут быть сохранены с использованием атрибута `data-*` .

Поддержка старых браузеров

Атрибуты данных были введены в HTML5, который поддерживается всеми современными браузерами, но старые браузеры до того, как HTML5 не распознают атрибуты данных.

Однако в спецификациях HTML атрибуты, которые не распознаются браузером, должны быть оставлены в покое, и браузер просто игнорирует их при рендеринге страницы.

Веб-разработчики использовали этот факт для создания нестандартных атрибутов, которые являются атрибутами, не входящими в спецификации HTML. Например, атрибут `value` в строке ниже считается нестандартным атрибутом, поскольку спецификации `<img>` не имеют атрибута `value` и не являются глобальным атрибутом:

```
<img src="sample.jpg" value="test" />
```

Это означает, что, хотя атрибуты данных не поддерживаются в старых браузерах, они все еще работают, и вы можете устанавливать и извлекать их с использованием тех же общих `setAttribute` `getAttribute` JavaScript и `getAttribute` , но вы не можете использовать новое свойство `dataset` которое поддерживается только в современных браузерах.

Прочитайте Атрибуты данных онлайн: https://riptutorial.com/ru/html/topic/1182/атрибуты-данных

глава 10: Атрибуты событий HTML

Examples

События HTML-формы

События, вызванные действиями внутри HTML-формы (применяются ко всем элементам HTML, но наиболее часто используются в элементах формы):

атрибут	Описание
ONBLUR	Запускает момент, когда элемент теряет фокус
по изменению	Запускает момент, когда изменяется значение элемента
oncontextmenu	Скрипт запускается при запуске контекстного меню
OnFocus	Запускает момент, когда элемент получает фокус
oninput	Скрипт запускается, когда элемент получает пользовательский ввод
oninvalid	Скрипт запускается, если элемент недействителен
OnReset	Пожары при нажатии кнопки «Сброс» в форме
onsearch	Пожары, когда пользователь что-то пишет в поле поиска (для <input = "search">)
onselect	Пожары после того, как какой-либо текст был выбран в элементе
onsubmit	Пожары при отправке формы

События в клавиатуре

атрибут	Описание
OnKeyDown	Пожары, когда пользователь нажимает клавишу
OnKeyPress	Пожары, когда пользователь нажимает клавишу
OnKeyUp	Пожары, когда пользователь отпускает ключ

Прочитайте Атрибуты событий HTML онлайн: https://riptutorial.com/ru/html/topic/10924/атрибуты-событий-html

глава 11: Включить код JavaScript в HTML

Синтаксис

- `<script type="text/javascript"> //some code </script>`
- `<script type="text/javascript" src="URL"></script>`
- `<script type="text/javascript" src="URL" async>//async code</script>`

параметры

атрибут	подробности
`src`	Указывает путь к файлу JavaScript. Либо относительный, либо абсолютный URL.
`type`	Указывает тип MIME. Этот атрибут требуется в HTML4, но необязательно в HTML5.
`async`	Указывает, что сценарий должен выполняться асинхронно (только для внешних скриптов). Этот атрибут не требует значения (кроме XHTML).
`defer`	Указывает, что сценарий должен быть выполнен, когда страница закончила синтаксический разбор (только для внешних скриптов). Этот атрибут не требует значения (кроме XHTML).
`charset`	Указывает кодировку символов, используемую во внешнем файле сценария, например UTF-8
`crossorigin`	Как элемент обрабатывает запросы с помощью crossorigin
`nonce`	Криптографическое nonce, используемое в *политике безопасности контента,* проверяет CSP3

замечания

Если встроенный JavaScript-код (файл) используется для управления http://stackoverflow.com/documentation/javascript/503/document-object-model-dom Elements, поместите теги `<script></script>` непосредственно **перед закрытием** `</body>` или использовать методы или библиотеки JavaScript (например, jQuery для обработки различных браузеров), что гарантирует, что DOM будет прочитан и готов к манипулированию.

Examples

Связывание с внешним файлом JavaScript

```
<script src="example.js"></script>
```

Атрибут `src` работает как атрибут href на якорях: вы можете указать абсолютный или относительный URL. В приведенном выше примере ссылки на файл внутри одного и того же каталога документа HTML. Обычно это добавляется внутри тегов `<head>` в верхней части html-документа

Непосредственно, включая код JavaScript

Вместо ссылки на внешний файл вы также можете включить JS-код как есть в своем HTML:

```
<script>
// JavaScript code
</script>
```

Включение файла JavaScript, выполняемого асинхронно

```
<script type="text/javascript" src="URL" async></script>
```

Обработка отключенного Javascript

Возможно, клиентский браузер не поддерживает Javascript или отключен Javascript, возможно, из-за соображений безопасности. Чтобы сообщить пользователям, что сценарий должен выполняться на странице, можно использовать тег `<noscript>` . Содержимое `<noscript>` отображается, когда Javascript отключен для текущей страницы.

```
<script>
  document.write("Hello, world!");
</script>
<noscript>This browser does not support Javascript.</noscript>
```

Прочитайте Включить код JavaScript в HTML онлайн: https://riptutorial.com/ru/html/topic/3719/включить-код-javascript-в-html

глава 12: встраивать

параметры

параметры	подробности
`src`	Адрес ресурса
`type`	Тип встроенного ресурса
`width`	Горизонтальное измерение
`height`	Вертикальный размер

Examples

Основное использование

`embed` тег является новым в HTML5. Этот элемент обеспечивает точку интеграции для внешнего (обычно не HTML) приложения или интерактивного контента.

```
<embed src="myflash.swf">
```

Определение типа MIME

Тип MIME должен быть определен с использованием атрибута `type` .

```
<embed type="video/mp4" src="video.mp4" width="640" height="480">
```

Прочитайте встраивать онлайн: https://riptutorial.com/ru/html/topic/8123/встраивать

глава 13: Выходной элемент

параметры

атрибут	Описание
Глобальный	Атрибуты, доступные любому элементу HTML5. Полную документацию по этим атрибутам см. В разделе: Глобальные атрибуты MDN
название	Строка, представляющая имя вывода. В качестве элемента формы на вывод можно ссылаться его имя, используя свойство `document.forms` . Этот атрибут также используется для сбора значений в форме submit.
за	Список разделенных пробелами идентификаторов элементов формы (например, `<inputs id="inp1"> for value is "inp1"`), что вывод предназначен для отображения вычислений.
форма	Строка, представляющая `<form>` которая связана с выходом. Если вывод фактически находится за пределами `<form>` , этот атрибут будет гарантировать, что вывод по-прежнему принадлежит `<form>` и подчиняется коллекциям и отправляет указанную `<form>` .

Examples

Элемент вывода с использованием атрибутов «А» и «Форма»

В следующей демонстрации используется элемент `<output>` `[for]` атрибутов `[for]` и `[form]` . Имейте в виду, что для работы `<output>` **необходим JavaScript** . Встроенный JavaScript обычно используется в формах, как демонстрирует этот пример. Хотя элементы `<input>` являются `type="number"` , их `value` s не является числом, это текст. Поэтому, если вам нужно вычислить `value` s, вы должны преобразовать каждое `value` в число, используя такие методы, как `parseInt()` , `parseFloat()` , `Number()` и т. Д.

Демо-версия

```
    <!--form1 will collect the values of in1 and in2 on 'input' event.-->
    <!--out1 value will be the sum of in1 and in2 values.-->

<form id="form1" name="form1" oninput="out1.value = parseInt(in1.value, 10) +
parseInt(in2.value, 10)">

  <fieldset>

    <legend>Output Example</legend>
```

```
      <input type="number" id="in1" name="in1" value="0">
    <br/>
    +
    <input type="number" id="in2" name="in2" value="0">

  </fieldset>

</form>

<!--[for] attribute enables out1 to display calculations for in1 and in2.-->
<!--[form] attribute designates form1 as the form owner of out1 even if it isn't a
descendant.-->

<output name="out1" for="in1 in2" form="form1">0</output>
```

Элемент вывода с атрибутами

```
<output name="out1" form="form1" for="inp1 inp2"></output>
```

Прочитайте Выходной элемент онлайн: https://riptutorial.com/ru/html/topic/723/выходной-элемент

глава 14: Глобальные атрибуты

параметры

атрибут	Описание
`class`	Определяет одно или несколько имен классов для элемента. См. Классы и идентификаторы .
`contenteditable`	Устанавливает, можно ли редактировать содержимое элемента.
`contextmenu`	Определяет контекстное меню, отображаемое, когда пользователь щелкает правой кнопкой элемент.
`dir`	Устанавливает направление текста для текста внутри элемента.
`draggable`	Устанавливает, можно ли перетаскивать элемент.
`hidden`	Скрывает элемент, который в настоящее время не используется на странице.
`id`	Определяет уникальный идентификатор элемента. См. Классы и идентификаторы .
`lang`	Определяет язык содержимого элемента и его значений текстового атрибута. См. Языки контента .
`spellcheck`	Устанавливает, следует ли проверять / изменять грамматику содержимого элемента.
`style`	Определяет набор встроенных стилей CSS для элемента.
`tabindex`	Устанавливает порядок, в котором элементы на странице перемещаются с помощью ярлыка клавиш табуляции.
`title`	Определяет дополнительную информацию об элементе, как правило, в виде текста всплывающей подсказки при наведении указателя мыши.
`translate`	Определяет, следует ли переводить содержимое элемента.

замечания

Глобальные атрибуты просто присваиваются, которые могут быть применены к любому

элементу во всем документе.

Examples

Contenteditable Attribute

```
<p contenteditable>This is an editable paragraph.</p>
```

После нажатия на абзац его содержимое можно редактировать аналогично текстовому полю ввода.

Когда атрибут `contenteditable` не задан для элемента, элемент наследует его от родителя. Таким образом, весь дочерний текст редактируемого содержимого будет также доступен для редактирования, но вы *можете* отключить его для определенного текста, например:

```
<p contenteditable>
  This is an editable paragraph.
  <span contenteditable="false">But not this.</span>
</p>
```

Обратите внимание, что неотредактируемый текстовый элемент внутри редактируемого элемента по-прежнему будет иметь текстовый курсор, унаследованный от его родителя.

Прочитайте Глобальные атрибуты онлайн: https://riptutorial.com/ru/html/topic/2811/глобальные-атрибуты

глава 15: Заголовки

Вступление

HTML предоставляет не только простые теги абзаца, но и шесть отдельных тегов заголовков для обозначения заголовков разных размеров и толщин. Перечисляется как заголовок 1 по заголовку 6, заголовок 1 имеет самый большой и самый толстый текст, а заголовок 6 - самый маленький и тонкий, вплоть до уровня абзаца. В этом разделе описывается правильное использование этих тегов.

Синтаксис

- `<h1>...</h1>`
- `<h2>...</h2>`
- `<h3>...</h3>`
- `<h4>...</h4>`
- `<h5>...</h5>`
- `<h6>...</h6>`

замечания

- Элемент `h1` - `h6` должен иметь как начальный тег, так и конечный тег. [1]
- Элементы `h1` - `h6` по умолчанию являются элементами уровня блока (стиль CSS: `display: block`). [2]
- Элементы `h1` - `h6` не следует путать с элементом сечения
- Возглавлять теги (`h1` - `h6`) не связаны с `head` метки.
- Допустимый контент: фразирование содержимого
- Различные CSS-стили для заголовков , как правило , отличаются по `font-size` и `margin` . Следующие CSS-настройки для элементов `h1 h6` могут служить ориентацией (характеризуемой как «информативная» W3C)
- Поисковые роботы (код, добавляющий страницу в поисковую систему) автоматически уделяют больше внимания более высокой значимости (h1 имеет больше всего, h2 имеет меньше, h3 - еще меньше, ...) заголовки, чтобы различить, о чем идет речь.

Examples

Использование заголовков

Заголовки могут использоваться для описания предшествующей темы, и они определяются с помощью тегов `<h1>` to `<h6>` . Заголовки поддерживают все глобальные атрибуты .

- `<h1>` определяет наиболее важный заголовок.
- `<h6>` определяет наименее важный заголовок.

Определение заголовка:

```
<h1>Heading 1</h1>
<h2>Heading 2</h2>
<h3>Heading 3</h3>
<h4>Heading 4</h4>
<h5>Heading 5</h5>
<h6>Heading 6</h6>
```

Правильная структура

Поисковые системы и другие **пользовательские агенты** обычно индексируют содержимое страницы на основе элементов заголовка, например, для создания оглавления, поэтому важно использовать правильную структуру заголовков.

В общем, статья должна иметь один элемент `h1` для основного названия, за которым следуют субтитры `h2` - при необходимости снижая слой. Если на более высоком уровне присутствуют элементы `h1` они не используются для описания содержимого более низкого уровня.

Пример документа (дополнительное намерение для иллюстрации иерархии):

```
<h1>Main title</h1>
<p>Introduction</p>

    <h2>Reasons</h2>

        <h3>Reason 1</h3>
        <p>Paragraph</p>

        <h3>Reason 2</h3>
        <p>Paragraph</p>

    <h2>In conclusion</h2>
    <p>Paragraph</p>
```

Прочитайте Заголовки онлайн: https://riptutorial.com/ru/html/topic/226/заголовки

глава 16: Изображений

Синтаксис

- `<img src="" alt="">`

параметры

параметры	подробности
`src`	Указывает URL-адрес изображения
`srcset`	Изображения для использования в разных ситуациях (например, дисплеи с высоким разрешением, небольшие мониторы и т. Д.)
`sizes`	Размеры изображения между точками останова
`crossorigin`	Как элемент обрабатывает запросы с помощью crossorigin
`usemap`	Название используемой карты изображения
`ismap`	Является ли это изображение на стороне сервера
`alt`	Альтернативный текст, который должен отображаться, если по какой-либо причине изображение не может отображаться
`width`	Задает ширину изображения (необязательно)
`height`	Указывает высоту изображения (необязательно)

Examples

Создание изображения

Чтобы добавить изображение на страницу, используйте тег изображения.

Теги изображений (`img`) не имеют закрывающих тегов. Два основных атрибута, которые вы даете тегу `img` : `src` , источник изображения и `alt` , который является альтернативным текстом, описывающим изображение.

```
<img src="images/hello.png" alt="Hello World">
```

Вы также можете получать изображения с веб-URL:

```
<img src="https://i.stack.imgur.com/ALgZi.jpg?s=48&g=1" alt="StackOverflow user Caleb
Kleveter">
```

Примечание. Изображения не привязаны к тексту на странице HTML, изображения связаны с HTML-страницами. Тег `<img>` создает пространство для хранения ссылочного изображения.

Также возможно встроить изображения непосредственно внутри страницы с помощью base64:

```
<img src="..." alt="Hello World">
```

Совет. Чтобы связать изображение с другим документом, просто `<img>` тег `<img>` внутри тегов `<a>` .

Ширина и высота изображения

> **Примечание. Атрибуты ширины и высоты *не* устаревают на изображениях** и никогда не были. Однако их использование было гораздо более строгим.

Размеры изображения могут быть указаны с использованием атрибутов `width` и `height` тега изображения:

```
<img src="images/200x200-img.png" width="200" height="200" alt="A 200x200 image">
```

Указав `width` и `height` изображения, ваша структура дает браузеру подсказку о том, как должна быть выложена страница, даже если вы просто указали фактический размер изображения. Если размеры изображения не указаны, браузеру необходимо будет пересчитать макет страницы после загрузки изображения, что может привести к тому, что страница «скачет» во время загрузки.

4,1

Вы можете предоставить изображение ширину и высоту либо в количестве пикселей CSS, либо в процентах от фактических размеров изображения.

Все эти примеры действительны:

```
<img src="images/50x50-img.png" width="50" height="50" alt="A 50x50 image">
<img src="images/50x50-img.png" width="200" height="200" alt="A 200x200 image">
<img src="images/200x200-img.png" width="50" height="50" alt="A 50x50 image">
<img src="images/200x200-img.png" width="50%" height="50%" alt="A 100x100 image">
```

5

Ширина и высота изображения должны быть указаны в пикселях CSS; процентное значение больше не является допустимым значением. Кроме того, если указаны оба

атрибута, они должны вписываться в одну из трех формул , поддерживающих соотношение сторон. Хотя и действительны, вы не должны использовать атрибуты width и height, чтобы растянуть изображение до большего размера.

Эти примеры действительны:

```
<img src="images/50x50-img.png" width="50" height="50" alt="A 50x50 image">
<img src="images/200x200-img.png" width="50" height="50" alt="A 50x50 image">
<img src="images/50x50-img.png" width="0" height="0" alt="A hidden tracking image">
```

Этот пример не рекомендуется:

```
<img src="images/50x50-img.png" width="200" height="200" alt="A 200x200 image">
```

Эти примеры недействительны:

```
<img src="images/200x200-img.png" width="50%" height="50%" alt="A 100x100 image">
<img src="images/200x200-img.png" width="1" height="200" alt="A 1x200 image">
```

Выбор alt text

Alt-текст используется читателями экрана для пользователей с ослабленным зрением и поисковыми системами. Поэтому важно написать хороший альт-текст для ваших изображений.

Текст должен выглядеть корректно, даже если вы замените изображение его атрибутом alt. Например:

```
<!-- Incorrect -->
<img src="anonymous.png" alt="Anonymous user avatar"/> An anonymous user wrote:
<blockquote>Lorem ipsum dolor sed.</blockquote>
<a href="https://google.com/"><img src="edit.png" alt="Edit icon"/></a> /
<a href="https://google.com/"><img src="delete.png" alt="Delete icon"/></a>
```

Без изображений это будет выглядеть так:

> Анонимный пользователь Аватара пользователя Анонимный пользователь написал:
>
> > Lorem ipsum dolor sed.
>
> Значок редактирования / Удалить значок

Чтобы исправить это:

- Удалите альт-текст для аватара. Это изображение добавляет информацию для зрячих пользователей (легко идентифицируемый значок, чтобы показать, что пользователь анонимный), но эта информация уже доступна в тексте. [1]

- Удалите «значок» из альт-текста для значков. Зная, что это будет икона, если она там, не поможет передать ее фактическую цель.

```
<!-- Correct -->
<img src="anonymous.png" alt=""/> An anonymous user wrote:
<blockquote>Lorem ipsum dolor sed.</blockquote>
<a href="https://google.com/"><img src="edit.png" alt="Edit"/></a> /
<a href="https://google.com/"><img src="delete.png" alt="Delete"/></a>
```

Анонимный пользователь написал:

> Lorem ipsum dolor sed.

Изменить / Удалить

Сноски

[1] Существует семантическая разница между включением атрибута empty alt и его исключением. Атрибут empty alt указывает на то, что изображение *не* является ключевой частью содержимого (как это верно в этом случае - это просто аддитивное изображение, которое не нужно понимать для остальных) и, следовательно, может быть опущено из рендеринга. Однако отсутствие атрибута alt указывает, что изображение *является* ключевой частью содержимого и что для рендеринга просто нет текстового эквивалента.

Отзывчивое изображение с использованием атрибута srcset

Использование srcset с размерами

```
<img sizes="(min-width: 1200px) 580px,
      (min-width: 640px) 48vw,
      98vw"
    srcset="img/hello-300.jpg 300w,
      img/hello-600.jpg 600w,
      img/hello-900.jpg 900w,
      img/hello-1200.jpg 1200w"
    src="img/hello-900.jpg" alt="hello">
```

`sizes` похожи на медиа-запросы, описывающие, сколько пространства занимает изображение в окне просмотра.

- если размер видового окна больше 1200 пикселей, изображение составляет ровно 580 пикселей (например, наш контент центрируется в контейнере с максимальным размером 1200 пикселей в ширину, а изображение занимает половину минус поля).
- если область просмотра находится между 640px и 1200px, изображение занимает 48% окна просмотра (например, масштабирование изображения с нашей страницей и занимает половину ширины окна просмотра минус поля).

- если viewport - это любой другой размер, в нашем случае менее 640 пикселей изображение занимает 98% окна просмотра (например, масштабирование изображения с нашей страницей и полная ширина окна просмотра минус поля). **Состояние носителя должно быть опущено для последнего элемента.**

`srcset` просто сообщает браузеру, какие изображения у нас есть, и каковы их размеры.

- `img/hello-300.jpg` 300.jpg имеет ширину 300 пикселей,
- `img/hello-600.jpg` имеет ширину `img/hello-600.jpg` пикселей,
- `img/hello-900.jpg` имеет ширину 900 пикселей,
- `img/hello-1200.jpg` имеет ширину `img/hello-1200.jpg` пикселей

`src` всегда является обязательным источником изображения. В случае использования с `srcset` , `src` будет служить резервным изображением в случае, если браузер не поддерживает `srcset` .

Использование srcset без размеров

```
<img src="img/hello-300.jpg" alt="hello"
     srcset="img/hello-300.jpg 1x,
       img/hello-600.jpg 2x,
       img/hello-1200.jpg 3x">
```

`srcset` предоставляет список доступных изображений, с устройством пиксел отношения `x` дескриптором.

- если отношение `img/hello-300.jpg` устройства к пикселю равно 1, используйте `img/hello-300.jpg`
- если соотношение сторон устройства равно 2, используйте `img/hello-600.jpg`
- если отношение количества пикселей к устройству равно 3, используйте `img/hello-1200.jpg`

`src` всегда является обязательным источником изображения. В случае использования с `srcset` , `src` будет служить резервным изображением в случае, если браузер не поддерживает `srcset` .

Отзывчивое изображение с использованием элемента изображения

Код

```
<picture>
  <source media="(min-width: 600px)" srcset="large_image.jpg">
  <source media="(min-width: 450px)" srcset="small_image.jpg">
  <img src="default_image.jpg" style="width:auto;">
</picture>
```

использование

Чтобы отображать разные изображения с различной шириной экрана, вы должны включать все изображения, используя тег источника в теге изображения, как показано в приведенном выше примере.

Результат

- На экранах с шириной экрана> 600 пикселей, это показывает large_image.jpg
- На экранах с шириной экрана> 450 пикселей, это показывает small_image.jpg
- На экранах с другой шириной экрана отображается default_image.jpg

Прочитайте Изображений онлайн: https://riptutorial.com/ru/html/topic/587/изображений

глава 17: Использование HTML с CSS

Вступление

CSS предоставляет стили для элементов HTML на странице. Встраиваемый стиль включает использование атрибута стиля в тегах и сильно обескуражен. Внутренние таблицы стилей используют `<style>` и используются для объявления правил для направленных частей страницы. Внешние таблицы стилей можно использовать с помощью `<link>` который берет внешний файл CSS и применяет правила к документу. В этом разделе рассматривается использование всех трех методов вложения.

Синтаксис

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

Examples

Использование внешней таблицы стилей

Используйте `link` атрибут в документе `head` :

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

Вы также можете использовать таблицы стилей, предоставляемые с веб-сайтов через сеть доставки контента, или CDN для краткости. (например, Bootstrap):

```
<head>
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"
crossorigin="anonymous">
</head>
```

Как правило, вы можете найти поддержку CDN для фреймворка на своем веб-сайте.

Внутренняя таблица стилей

Вы также можете включать элементы CSS внутри, используя `<style>` :

```
<head>
    <style type="text/css">
        body {
```

```
            background-color: gray;
        }
    </style>
</head>
```

В программу также можно включить несколько внутренних таблиц стилей.

```
<head>
    <style type="text/css">
        body {
            background-color: gray;
        }
    </style>

    <style type="text/css">
        p {
            background-color: blue;
        }
    </style>
</head>
```

Стиль Inline

Вы можете создать определенный элемент, используя атрибут `style` :

```
<span style="color: red">This text will appear in red.</span>
```

> Примечание. Старайтесь избегать этого - точка CSS заключается в том, чтобы отделить контент от презентации.

Несколько таблиц стилей

Можно загрузить несколько таблиц стилей:

```
<head>
    <link rel="stylesheet" type="text/css" href="general.css">
    <link rel="stylesheet" type="text/css" href="specific.css">
</head>
```

Обратите внимание, что **более поздние файлы и декларации будут отменять предыдущие** . Поэтому, если `general.css` содержит:

```
body {
    background-color: red;
}
```

и `specific.css` содержит:

```
body {
    background-color: blue;
}
```

если оба используются, фон документа будет синим.

Прочитайте Использование HTML с CSS онлайн: https://riptutorial.com/ru/html/topic/4536/использование-html-с-css

глава 18: Карты изображений

Синтаксис

- `<img usemap="#[map-name]">`
- `<map name="[map-name]"></map>`
- `<area>`

параметры

Tag / Атрибут	Значение
`<img>`	Ниже приведены атрибуты изображения для отображения с помощью `<img>` . Применяются обычные атрибуты `<img>` .
`usemap`	`name` карты с символом хеша, предшествующим ей. Например, для карты с `name="map"` изображение должно иметь `usemap="#map"` .
`<map>`	
`name`	Имя карты для ее идентификации. Используется с атрибутом `usemap` изображения.
`<area>`	Ниже указаны атрибуты `<area>` . Когда указано `href` , если ссылка `<area>` a link, `<area>` также поддерживает все атрибуты тега привязки (`<a>`), кроме `ping` . См. Их в документах MDN .
`alt`	Альтернативный текст для отображения, если изображения не поддерживаются. Это необходимо, только если `href` также установлен на `<area>` .
`coords`	Координаты, определяющие выбранную область. Когда `shape="polygon"` , это должно быть установлено в список пар «x, y», разделенных запятыми (т. `shape="polygon" coords="x1, y1, x2, y2, x3, y3, ..."`). Когда `shape="rectangle"` , это должно быть установлено `left, top, right, bottom` . Когда `shape="circle"` , это должно быть установлено в `centerX, centerY, radius` .
`href`	URL-адрес гиперссылки, если указан. Если он опущен, то `<area>` не будет представлять гиперссылку.
`shape`	Форма `<area>` . Может быть установлен по `default` , чтобы выбрать все изображение (нет `coords` не приписывать необходимости), `circle` или `circ`

Tag / Атрибут	Значение
	для окружности, `rectangle` или `rect` для прямоугольника и `polygon` или `poly` для многоугольной области , заданной угловыми точками.

замечания

- Список перечисленных выше параметров изменен из документов MDN: `<map>` и `<area>` .
- Возможно создание координат карты изображения для изображения с более простыми формами (например, во вводном примере выше) с помощью редактора изображений, который показывает координаты (например, GIMP). Однако в общем случае проще использовать генератор карт изображений, например, этот .

Examples

Введение в карты изображений

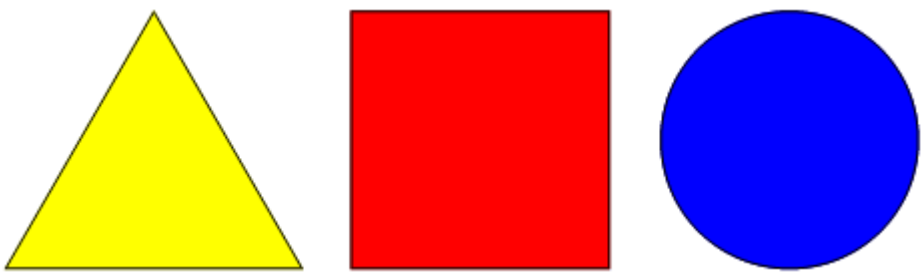
Описание

Карты изображений - это изображение с интерактивными областями, которые обычно действуют как гиперссылки.

Изображение определяется тегом `<img>` , а карта определяется тегом `<map>` тегами `<area>` чтобы обозначать каждую зону, на которую можно щелкнуть. Используйте `usemap` и `name` для привязки изображения и карты.

Основной пример

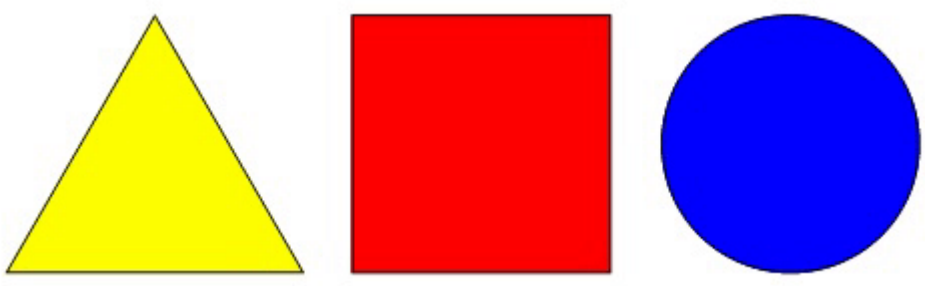
Чтобы создать карту изображения, чтобы каждая из фигур на изображении ниже была нажата:



Код будет выглядеть следующим образом:

```
<img src="http://jaced.com/blogpix/2007/trisquarecircle/002.gif" usemap="#shapes">
<map name="shapes">
    <area shape="polygon" coords="79,6,5,134,153,134">
    <area shape="rectangle" coords="177,6,306,134">
    <area shape="circle" coords="397,71,65">
</map>
```

Вы должны увидеть, что браузер распознает области, когда курсор становится указателем. См. Живую демонстрацию на JSFiddle или см. Демонстрацию ниже:



Прочитайте Карты изображений онлайн: https://riptutorial.com/ru/html/topic/3819/карты-изображений

глава 19: Классы и идентификаторы

Вступление

Классы и идентификаторы облегчают обращение к элементам HTML из сценариев и таблиц стилей. Атрибут class может использоваться в одном или нескольких тегах и используется CSS для стилизации. Однако идентификаторы предназначены для обозначения одного элемента, что означает, что один и тот же идентификатор никогда не должен использоваться дважды. Идентификаторы обычно используются с JavaScript и внутренними ссылками документа и не рекомендуется в CSS. В этом разделе содержатся полезные пояснения и примеры относительно правильного использования атрибутов класса и идентификатора в HTML.

Синтаксис

- class = "class1 class2 class3"
- ID = "UniqueID"

параметры

параметр	подробности
учебный класс	Указывает класс элемента (не уникальный)
Я бы	Указывает идентификатор элемента (уникальный в том же контексте)

замечания

- Оба `class` и `id` являются глобальными атрибутами и поэтому могут быть привязаны к любому элементу HTML.
- Имена классов должны начинаться с буквы (AZ или az), за которой могут следовать буквы, цифры, дефисы и символы подчеркивания.
- В `HTML5` атрибуты class и id могут использоваться для любого элемента. В HTML 4.0.1 они были недоступны для тегов `<base>` , `<head>` , `<html>` , `<meta>` , `<param>` , `<script>` , `<style>` и `<title>` .
- Элемент может иметь один или несколько классов. Классы разделяются пробелами и сами не могут содержать пробелы.
- Элемент может иметь только один идентификатор, и он должен быть уникальным в своем контексте (т.е. веб-странице). Идентификаторы также не могут содержать пробелы.

Examples

Предоставление элемента класса

Классы - это идентификаторы для элементов, которым они назначены. Используйте атрибут `class` чтобы назначить класс элементу.

```
<div class="example-class"></div>
```

Чтобы назначить несколько классов элементу, отделите имена классов пробелами.

```
<div class="class1 class2"></div>
```

Использование классов в CSS

Классы могут использоваться для стилизации определенных элементов без изменения всех элементов такого типа. Например, эти два элемента `span` могут иметь совершенно разные стили:

```
<span></span>
<span class="special"></span>
```

Классы с одинаковым именем могут быть предоставлены любому количеству элементов на странице, и все они получат стиль, связанный с этим классом. Это всегда будет верно, если вы не укажете элемент внутри CSS.

Например, у нас есть два элемента: оба с `highlight` класса:

```
<div class="highlight">Lorem ipsum</div>
<span class="highlight">Lorem ipsum</span>
```

Если наш CSS такой, как показано ниже, то зеленый цвет будет применен к тексту в обоих элементах:

```
.highlight { color: green; }
```

Однако, если мы хотим только нацеливать `div` на `highlight` класса, мы можем добавить определенность, как показано ниже:

```
div.highlight { color: green; }
```

Тем не менее, при стилизации с помощью CSS обычно рекомендуется использовать только классы (например, `.highlight`), а не элементы с классами (например, `div.highlight`).

Как и любой другой селектор, классы могут быть вложенными:

```
.main .highlight { color: red; } /* Descendant combinator */
.footer > .highlight { color: blue; } /* Child combinator */
```

Вы также можете связать селектор классов только с элементами, имеющими комбинацию из нескольких классов. Например, если это наш HTML:

```
<div class="special left menu">This text will be pink</div>
```

И мы хотим покрасить эту определенную часть текста розовым, мы можем сделать следующее в нашем CSS:

```
.special.left.menu { color: pink; }
```

Предоставление элемента идентификатора

Атрибут ID элемента - это идентификатор, который должен быть уникальным во всем документе. Его цель состоит в том, чтобы однозначно идентифицировать элемент при связывании (с использованием привязки), написании сценариев или стилизации (с помощью CSS).

```
<div id="example-id"></div>
```

У вас не должно быть двух элементов с одинаковым идентификатором в одном документе, даже если атрибуты привязаны к двум различным типам элементов. Например, следующий код неверен:

```
<div id="example-id"></div>
<span id="example-id"></span>
```

Браузеры сделают все возможное, чтобы сделать этот код, но неожиданное поведение может возникнуть при стилизации с помощью CSS или добавлении функциональности с помощью JavaScript.

Чтобы ссылаться на элементы по их идентификатору в CSS, префикс ID с `#` .

```
#example-id { color: green; }
```

Чтобы перейти к элементу с идентификатором на заданной странице, добавьте `#` с именем элемента в URL.

```
http://example.com/about#example-id
```

Эта функция поддерживается в большинстве браузеров и не требует дополнительного JavaScript или CSS для работы.

Проблемы, связанные с дублируемыми идентификаторами

Наличие более одного элемента с одним и тем же идентификатором - проблема с устранением неполадок. Парсер HTML обычно пытается отобразить страницу в любом случае. Обычно ошибка не возникает. Но темпы могут оказаться на неправильной веб-странице.

В этом примере:

```
<div id="aDiv">a</div>
<div id="aDiv">b</div>
```

Селекторы CSS все еще работают

```
#aDiv {
    color: red;
}
```

Но JavaScript не справляется с обоими элементами:

```
var html = document.getElementById("aDiv").innerHTML;
```

В этом случае переменная `html` несет только первое содержимое `div ("a")` .

Допустимые значения

Для идентификатора

5

Единственными ограничениями на значение `id` являются:

1. он должен быть уникальным в документе
2. он не должен содержать пробелов
3. он должен содержать хотя бы один символ

Таким образом, значением могут быть все цифры, только одна цифра, только знаки пунктуации, включают специальные символы, что угодно. Просто нет пробелов.

Таким образом, они действительны:

```
<div id="container"> ... </div>
<div id="999"> ... </div>
<div id="#%LV-||"> ... </div>
<div id="____V"> ... </div>
<div id="□□"> ... </div>
<div id="♥"> ... </div>
<div id="{}"> ... </div>
```

```
<div id="©"> ... </div>
<div id="⌂₩¤☆€~¥"> ... </div>
```

Это неверно:

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

Это также неверно, если включено в тот же документ:

```
<div id="results"> ... </div>
<div id="results"> ... </div>
```

4,01

Значение `id` должно начинаться с буквы, которая затем может выполняться только:

- буквы (AZ / az)
- цифры (0-9)
- дефисы ("-")
- подчеркивания ("_")
- colons (":")
- периоды (".")

Ссылаясь на первую группу примеров в вышеприведенном HTML5, допустимо только одно:

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

Они также действительны:

```
<div id="sampletext"> ... </div>
<div id="sample-text"> ... </div>
<div id="sample_text"> ... </div>
<div id="sample:text"> ... </div>
<div id="sample.text"> ... </div>
```

Опять же, если он не начинается с буквы (в верхнем или нижнем регистре), это недействительно.

Для класса

Правила для классов по существу те же, что и для `id` . Разница в том, что значения `class` *не* обязательно должны быть уникальными в документе.

Ссылаясь на приведенные выше примеры, хотя это недопустимо в одном документе:

```
<div id="results"> ... </div>
<div id="results"> ... </div>
```

Это совершенно нормально:

```
<div class="results"> ... </div>
<div class="results"> ... </div>
```

Важное примечание. Как значения идентификатора и класса обрабатываются вне HTML

Имейте в виду, что приведенные выше правила и примеры применяются в контексте HTML.

Использование чисел, знаков препинания или специальных символов в значении `id` или `class` может вызвать проблемы в других контекстах, таких как CSS, JavaScript и регулярные выражения.

Например, хотя в HTML5 допустим следующий `id` :

```
<div id="9lions"> ... </div>
```

... это неверно в CSS:

> ### 4.1.3. Персонажи и случай
>
> В CSS *идентификаторы* (включая имена элементов, классы и идентификаторы в селекторах) могут содержать только символы [a-zA-Z0-9] и символы ISO 10646 U + 00A0 и выше, плюс дефис (-) и подчеркивание (_); ***они не могут начинаться с цифры, двух дефисов или дефисов, за которыми следует цифра*** . (выделено мной)

В большинстве случаев вы можете избегать символов в контекстах, где у них есть ограничения или особый смысл.

Ссылки W3C

- 3.2.5.1 Атрибут `id`
- 3.2.5.7 Атрибут `class`
- 6.2 Основные типы SGML

Прочитайте Классы и идентификаторы онлайн: https://riptutorial.com/ru/html/topic/586/классы-и-идентификаторы

глава 20: Комментарии

Вступление

В комментариях в HTML, как и в других языках программирования, разметки и разметки, другие разработчики имеют информацию о специфике разработки, не затрагивая пользовательский интерфейс. Однако, в отличие от других языков, HTML-комментарии могут использоваться для указания HTML-элементов только для Internet Explorer. В этом разделе объясняется, как писать комментарии HTML и их функциональные приложения.

Синтаксис

- `<!-- Comment text -->`

замечания

Все, начиная с `<!--` и заканчивая `-->` - это комментарий. Комментарии не могут содержать две смежные тире (`--`) и должны заканчиваться ровно двумя тире (т.е. `--->` неверно).

Комментарии не отображаются на веб-странице и не могут быть написаны с помощью CSS. Они могут использоваться разработчиком страницы для создания заметок в HTML или для скрытия определенного контента во время разработки.

Для динамических или интерактивных страниц скрытие и показ контента выполняется с помощью JavaScript и CSS, а не с комментариями HTML.

JavaScript может использоваться для получения содержимого узлов комментариев HTML, и эти узлы могут быть динамически созданы, добавлены и удалены из документа, но это не повлияет на отображение страницы.

Поскольку комментарии HTML являются частью исходного кода страницы, они загружаются в браузер вместе с остальной частью страницы. Исходный код обычно можно просмотреть с помощью опции меню веб-браузера «Просмотр источника» или «Просмотр источника страницы».

Examples

Создание комментариев

HTML-комментарии могут использоваться, чтобы оставлять заметки для себя или других разработчиков о конкретной точке кода. Они могут быть инициированы с помощью `<!--` и заключены с `-->` , например:

```
<!-- I'm an HTML comment! -->
```

Они могут быть встроены внутри другого содержимого:

```
<h1>This part will be displayed <!-- while this will not be displayed -->.</h1>
```

Они также могут охватывать несколько строк, чтобы предоставить дополнительную информацию:

```
<!-- This is a multiline HTML comment.
  Whatever is in here will not be rendered by the browser.
  You can "comment out" entire sections of HTML code.
-->
```

Однако они **не могут** отображаться в другом теге HTML, например:

```
<h1 <!-- testAttribute="something" -->>This will not work</h1>
```

Это приводит к недопустимому HTML, поскольку весь блок `<h1 <!-- testAttribute="something" -->` будет считаться одним стартовым тегом `h1` с некоторой другой недействительной информацией, содержащейся в нем, за которой следует единственная `>` закрывающая скобка, которая ничего не делает.

Для совместимости с инструментами, которые пытаются проанализировать HTML как XML или SGML, тело вашего комментария не должно содержать двух тире `--` .

Условные комментарии для Internet Explorer

Условные комментарии могут использоваться для настройки кода для разных версий Microsoft Internet Explorer. Например, могут быть предоставлены различные классы HTML, теги скриптов или таблицы стилей. Условные комментарии поддерживаются в версиях Internet Explorer с 5 по 9. Старые и новые версии Internet Explorer и все браузеры, отличные от IE, считаются «downlevel» и обрабатывают условные комментарии как обычные комментарии HTML.

Downlevel скрытая

Скрытые в нижнем слое комментарии работают, инкапсулируя весь контент в том, что кажется нормальным комментарием HTML. Только IE с 5 по 9 все равно будет читать его как условный комментарий, и они будут скрывать или отображать содержимое соответственно. В других браузерах контент будет скрыт.

```
<!--[if IE]>
  Revealed in IE 5 through 9. Commented out and hidden in all other browsers.
<![endif]-->
```

```
<!--[if lt IE 8]>
  Revealed only in specified versions of IE 5-9 (here, IE less than 8).
<![endif]-->

<!--[if !IE]>
  Revealed in no browsers. Equivalent to a regular HTML comment.
<![endif]-->

<!--
  For purposes of comparison, this is a regular HTML comment.
-->
```

Downlevel-показало

Они немного отличаются от скрытых комментариев нижнего уровня: только сам условный комментарий содержится в синтаксисе нормального комментария. Браузеры, которые не поддерживают условные комментарии, просто игнорируют их и отображают остальную часть контента между ними.

```
<!--[if IE]>-->
  The HTML inside this comment is revealed in IE 5-9, and in all other browsers.
<!--<![endif]-->

<!--[if IE 9]>-->
  This is revealed in specified versions of IE 5-9, and in all other browsers.
<!--<![endif]-->

<!--[if !IE]>-->
  This is not revealed in IE 5-9. It's still revealed in other browsers.
<!--<![endif]-->
```

Комментирование пробелов между встроенными элементами

Встроенные элементы отображения, обычно такие как `span` или `a` , будут содержать до одного символа пробела до и после них в документе. Чтобы избежать очень длинных строк в разметке (которые трудно читать) и непреднамеренного пробела (что влияет на форматирование), белое пространство может быть прокомментировано.

```
<!-- Use an HTML comment to nullify the newline character below: -->
<a href="#">I hope there will be no extra whitespace after this!</a><!--
--><button>Foo</button>
```

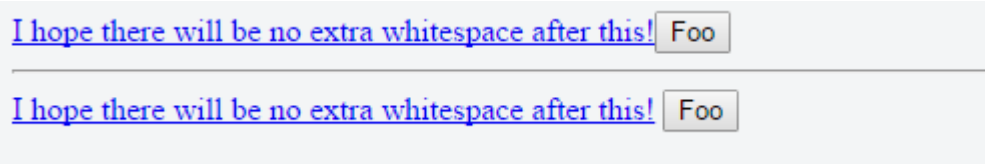
Попробуйте это без комментариев между встроенными элементами, и между ними будет одно пространство. Желательно получить символ пробела.

Пример кода:

```
<!-- Use an HTML comment to nullify the newline character below: -->
<a href="#">I hope there will be no extra whitespace after this!</a><!--
--><button>Foo</button>
<hr>
```

```
<!-- Without it, you can notice a small formatting difference: -->
<a href="#">I hope there will be no extra whitespace after this!</a>
<button>Foo</button>
```

Выход:



Прочитайте Комментарии онлайн: https://riptutorial.com/ru/html/topic/468/комментарии

глава 21: Маркировка компьютерного кода

Синтаксис

- `<pre>Formatted text</pre>`
- `<code>Inline Code</code>`

замечания

Элемент `code` должен использоваться для любой «строки, которую компьютер распознает» (HTML5), например:

- исходный код
- термины из языков разметки / программирования (имена элементов, имена функций и т. д.)
- имена файлов

Связанные элементы

Для переменных используется элемент `var` .

Для выхода компьютера можно использовать элемент `samp` .

Для ввода пользователем можно использовать элемент `kbd` .

Examples

В соответствии с

Если предложение содержит компьютерный код (например, имя элемента HTML), используйте элемент `code` для его маркировки:

```
<p>The <code>a</code> element creates a hyperlink.</p>
```

Блок с

```
 a также
```

Если форматирование (пробел, новые строки, отступы) кода имеет значение, используйте `pre` элемент в сочетании с элементом `code` :

```
<pre>
    <code>
```

```
    x = 42
    if x == 42:
        print "x is …           … 42"
    </code>
</pre>
```

Вам все равно придется избегать символов со специальным значением в HTML (например, `<` with `<`), поэтому для *отображения* блока кода HTML (`<p>This is a paragraph.</p>`) он может выглядеть так:

```
<pre>
    <code>
    &lt;p>This is a paragraph.&lt;/p>
    </code>
</pre>
```

Прочитайте Маркировка компьютерного кода онлайн:
https://riptutorial.com/ru/html/topic/1836/маркировка-компьютерного-кода

глава 22: Маркировочные котировки

замечания

элементы `cite` и `blockquote` не должны использоваться для целей представления разговора, расшифровки разговоров, диалогов в сценариях, записей мгновенных сообщений и других ситуаций, в которых разные игроки играют по очереди в речи.

Examples

В соответствии с

Элемент **`q`** может использоваться для цитаты, которая является частью предложения:

```
<p>She wrote <q>The answer is 42.</q> and everyone agreed.</p>
```

Кавычки

4,01

Котировочные знаки не следует добавлять. Пользовательские агенты должны (в HTML 4.01) соответственно. must (в HTML 4.0) визуализировать их автоматически.

5

Котировки не должны добавляться. Пользовательские агенты будут отображать их автоматически.

Исходный URL (атрибут `cite`)

Атрибут `cite` может использоваться для ссылки на URL-адрес цитируемого источника:

```
<p>She wrote <q cite="http://example.com/blog/hello-world">The answer is 42.</q> and everyone
agreed.</p>
```

Обратите внимание , что браузеры обычно не показывают этот URL, поэтому , если источник имеет значения, вы должны добавить гиперссылка (элемент) в дополнении. `a`

Блок с

Элемент **`blockquote`** может использоваться для цитаты (блочного уровня):

```
<blockquote>
  <p>The answer is 42.</p>
```

```
</blockquote>
```

Исходный URL (атрибут `cite`)

Атрибут `cite` может использоваться для ссылки на URL-адрес цитируемого источника:

```
<blockquote cite="http://example.com/blog/hello-world">
  <p>The answer is 42.</p>
</blockquote>
```

Обратите внимание , что браузеры обычно не показывают этот URL, поэтому , если источник имеет значения, вы должны добавить гиперссылка (элемент) в дополнении (смотрите раздел *Citation / Attribution* о том, где разместить эту ссылку). `a`

Цитирование / Attribution

4,01

Цитирование / атрибуция не должны быть частью элемента `blockquote` :

```
<blockquote cite="http://example.com/blog/hello-world">
  <p>The answer is 42.</p>
</blockquote>
<p>Source: <cite><a href="http://example.com/blog/hello-world" rel="external">Hello
World</a></cite></p>
```

Вы можете добавить элемент `div` чтобы сгруппировать цитату и цитату, но она не способ связывать их семантически.

Элемент **`cite`** может использоваться для ссылки цитируемого источника (но не для имени автора).

5

Цитирование / атрибуция (например, гиперссылка, указывающая исходный URL-адрес), может быть внутри `blockquote` , но в этом случае она должна находиться внутри элемента `cite` (для атрибутов в тексте) или элемента `footer` :

```
<blockquote cite="http://example.com/blog/hello-world">
  <p>The answer is 42.</p>
  <footer>
    <p>Source: <cite><a href="http://example.com/blog/hello-world" rel="external">Hello
World</a></cite></p>
  </footer>
</blockquote>
```

Элемент **`cite`** может использоваться для ссылки цитируемого источника или для имени автора цитаты.

Прочитайте Маркировочные котировки онлайн: https://riptutorial.com/ru/html/topic/2943/маркировочные-котировки

глава 23: Медиа-элементы

параметры

атрибут	подробности
ширина	Устанавливает ширину элемента в пикселях.
рост	Устанавливает высоту элемента в пикселях.
`<source>`	Определяет ресурсы аудио- или видеофайлов
трек	Определяет текстовую дорожку для элементов мультимедиа
управления	Отображает элементы управления
Автовоспроизведение	Автоматическое воспроизведение медиафайлов
петля	Воспроизведение носителя в повторяющемся цикле
приглушенный	Воспроизводит медиа без звука
плакат	Назначает изображение для отображения до загрузки видео

замечания

Поддержка в браузерах

Особенность	Хром	Firefox (Gecko)	Internet Explorer	опера	Сафари
Базовая поддержка	3.0	3,5 (1,9,1)	9,0	10,50	3,1
`<audio>` : PCM в WAVE	(Да)	3,5 (1,9,1)	Никакой поддержки	10,50	3,1
`<audio>` : Vorbis в WebM	(Да)	4.0 (2.0)	Никакой поддержки	10,60	3,1
`<audio>` : Streaming Vorbis / Opus в WebM	?	36,0	?	?	?

Особенность	Хром	Firefox (Gecko)	Internet Explorer	опера	Сафари
через MSE					
`<audio>` : Vorbis в Ogg	(Да)	3,5 (1,9,1)	Никакой поддержки	10,50	Никакой поддержки
`<audio>` : MP3	(Да)	(Да)	9,0	(Да)	3,1
`<audio>` : MP3 в MP4	?	?	?	?	(Да)
`<audio>` : AAC в MP4	(Да)	(Да)	9,0	(Да)	3,1
`<audio>` : Opus в Ogg	27,0	15,0 (15,0)	?	?	?
`<video>` : VP8 и Vorbis в WebM	6,0	4.0 (2.0)	9,0	10,60	3,1
`<video>` : VP9 и Opus в WebM	29,0	28,0 (28,0)	?	(Да)	?
`<video>` : Потоковая передача WebM через MSE	?	42,0 (42,0)	?	?	?
`<video>` : Theora и Vorbis в Ogg	(Да)	3,5 (1,9,1)	Никакой поддержки	10,50	Никакой поддержки
`<video>` : H.264 и MP3 в MP4	(Да)	(Да)	9,0	(Да)	(Да)
`<video>` : H.264 и AAC в MP4	(Да)	(Да)	9,0	(Да)	3,1
любой другой формат	Никакой поддержки	Никакой поддержки	Никакой поддержки	Никакой поддержки	3,1

Examples

Используя ` `и` `элемент для отображения аудио / видеоконтента

Используйте HTML или `<audio>` элемент для встраивания видео / аудио контента в документ. Элемент video / audio содержит один или несколько источников видео / аудио. Чтобы указать источник, используйте либо атрибут src, либо элемент `<source>` ; браузер выберет наиболее подходящий.

Пример аудио-тега:

```
<!-- Simple video example -->
<video src="videofile.webm" autoplay poster="posterimage.jpg">
  Sorry, your browser doesn't support embedded videos,
  but don't worry, you can <a href="videofile.webm">download it</a>
  and watch it with your favorite video player!
</video>

<!-- Video with subtitles -->
<video src="foo.webm">
  <track kind="subtitles" src="foo.en.vtt" srclang="en" label="English">
  <track kind="subtitles" src="foo.sv.vtt" srclang="sv" label="Svenska">
</video>
<!-- Simple video example -->
<video width="480" controls poster="https://archive.org/download/WebmVp8Vorbis/webmvp8.gif" >
  <source src="https://archive.org/download/WebmVp8Vorbis/webmvp8.webm" type="video/webm">
  <source src="https://archive.org/download/WebmVp8Vorbis/webmvp8_512kb.mp4" type="video/mp4">
  <source src="https://archive.org/download/WebmVp8Vorbis/webmvp8.ogv" type="video/ogg">
  Your browser doesn't support HTML5 video tag.
</video>
```

Пример аудио-тега:

```
<!-- Simple audio playback -->
<audio src="http://developer.mozilla.org/@api/deki/files/2926/=AudioTest_(1).ogg" autoplay>
  Your browser does not support the <code>audio</code> element.
</audio>

<!-- Audio playback with captions -->
<audio src="foo.ogg">
  <track kind="captions" src="foo.en.vtt" srclang="en" label="English">
  <track kind="captions" src="foo.sv.vtt" srclang="sv" label="Svenska">
</audio>
```

аудио

HTML5 предоставляет новый стандарт для встраивания аудиофайла на веб-страницу.

Вы можете вставить аудиофайл на страницу с помощью элемента `<audio>` :

```
<audio controls>
  <source src="file.mp3" type="audio/mpeg">
Your browser does not support the audio element.
</audio>
```

видео

Вы можете вставить также видео на веб-страницу с помощью элемента `<video>` :

```
<video width="500" height="700" controls>
  <source src="video.mp4" type="video/mp4">
Your browser does not support the video tag.
</video>
```

Заголовок видео или фон

Добавление видео, которое будет автоматически запускаться в цикле и не имеет элементов управления или звука. Идеально подходит для видео-заголовка или фона.

```
<video width="1280" height="720" autoplay muted loop poster="video.jpg" id="videobg">
  <source src="video.mp4" type="video/mp4">
  <source src="video.webm" type="video/webm">
  <source src="video.ogg" type="video/ogg">
</video>
```

Этот CSS обеспечивает резервную копию, если видео нельзя загрузить. Обратите внимание, что рекомендуется использовать первый кадр видео в качестве плаката video.jpg.

```
#videobg {
  background: url(video.jpg) no-repeat;
  background-size: cover;
}
```

Прочитайте Медиа-элементы онлайн: https://riptutorial.com/ru/html/topic/2111/медиа-элементы

глава 24: Мета-информация

Вступление

Метатеги в документах HTML предоставляют полезную информацию о документе, включая описание, ключевые слова, автора, даты изменений и около 90 других полей. В этом разделе рассматриваются использование и назначение этих тегов.

Синтаксис

- `<meta name="metadata name" content="value">`
- `<meta http-equiv="pragma directive" content="value">`
- `<meta charset="encoding label">`

замечания

Метатег - это тег HTML, используемый для установки метаданных документа HTML. Метатеги должны быть в элементе head. На странице может быть любое количество метатег.

`keywords` метатега обычно не используются роботами. Большинство поисковых систем определяют, какие ключевые слова соответствуют содержанию на веб-страницах. При этом ничего не говорится, что вы больше не должны включать метатег ключевых слов.

Мета-данные страницы в основном используются браузером (например, масштабирование документа) и паутинами паутины, используемыми поисковыми системами (Google, Yahoo !, Bing).

Спецификация дает ряд стандартизованных имен метаданных для использования с `<meta name>` и стандартизованными директивами pragma метаданных для использования с `<meta http-equiv>` . Тем не менее, многие службы в Интернете (веб-сканеры, инструменты разработки, службы социального обмена и т. Д.) Используют форму `<meta name>` в качестве общей точки расширения для метаданных. Некоторые из них перечислены на вики-странице spec .

Examples

Кодировка символов

Атрибут `charset` указывает кодировку символов для HTML-документа и должен быть допустимой кодировкой символов (примеры включают `windows-1252` , `ISO-8859-2` , `Shift_JIS` и `UTF-8`). `UTF-8` (Unicode) является наиболее широко используемым и должен использоваться

для любого нового проекта.

5

```
<meta charset="UTF-8">
```

```
<meta charset="ISO-8859-1">
```

Все браузеры всегда распознавали форму `<meta charset>` , но если вам почему-то нужна ваша страница, чтобы быть действительной HTML 4.01, вы можете использовать следующее:

```
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
```

```
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1">
```

См. Также Стандарт кодирования , чтобы просмотреть все доступные символы кодировки символов, которые распознают браузеры.

Автоматическое обновление

Чтобы обновить страницу каждые пять секунд, добавьте этот `meta` элемент `head` :

```
<meta http-equiv="refresh" content="5">
```

ВНИМАНИЕ! Хотя это действительная команда, рекомендуется не использовать ее из-за ее негативных последствий для пользователя. Обновление страницы слишком часто может привести к ее невосприимчивости и часто прокручивается вверху страницы. Если некоторая информация на странице должна постоянно обновляться, есть намного лучшие способы сделать это, только обновив часть страницы.

Управление компоновкой мобильных устройств

Общие сайты, оптимизированные для мобильных устройств, используют `<meta name="viewport">` следующим образом:

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

Элемент `viewport` дает инструкции браузера о том, как управлять размерами страницы и масштабированием на основе используемого вами устройства.

В приведенном выше примере, `content="width=device-width` означает , что браузер отображает ширину страницы по ширине своего собственного экрана. Таким образом , если этот экран `480px wide` в `480px wide initial-scale=1 480px wide` , в окне браузера будет `480px wide` в `480px wide . initial-scale=1` показывает, что начальный масштаб (который в этом случае

равен 1, означает, что он не масштабируется).

Ниже приведены атрибуты, поддерживаемые этим тегом:

атрибут	Описание
`width`	Ширина виртуального видового экрана устройства. Значения [1] : `device-width` или фактическая ширина в пикселях, например `480`
`height`	Высота виртуального видового экрана устройства. Значения [2] : `device-height` или фактическая ширина в пикселях, например `600`
`initial-scale`	Начальный масштаб при загрузке страницы. 1.0 не масштабируется.
`minimum-scale`	Минимальная сумма, которую посетитель может увеличить на странице. `1.0` не масштабируется.
`maximum-scale`	Максимальная сумма, которую посетитель может увеличить на странице. `1.0` не масштабируется.
`user-scalable`	Позволяет устройству увеличивать и уменьшать масштаб. Значения `yes` или `no` . Если установлено значение no, пользователь не может увеличить веб-страницу. По умолчанию да. Настройки браузера могут игнорировать это правило.

Заметки:

[1] Свойство `width` может быть задано в *пикселях* (`width=600`) или *шириной устройства* (`width=device-width`), которая представляет физическую ширину экрана устройства.

[2] Аналогично, свойство `height` может быть задано в `pixels` (`height=600`) или по `device-height` (`height=device-height`), которая представляет физическую высоту экрана устройства.

Информация о странице

`application-name`

Предоставление имени веб-приложения, которое представляет страница.

```
<meta name="application-name" content="OpenStreetMap">
```

Если это не веб-приложение, метатег `application-name` не должен использоваться.

`author`

Укажите автора страницы:

```
<meta name="author" content="Your Name">
```

Можно указать только одно имя.

description

Задайте описание страницы:

```
<meta name="description" content="Page Description">
```

Метатег `description` может использоваться различными поисковыми системами при индексировании вашей веб-страницы для целей поиска. Обычно описание, содержащееся в метатеге, представляет собой краткое резюме, которое отображается в главном заголовке страницы / сайта в результатах поисковой системы. Обычно Google использует только первые 20-25 слов вашего описания.

generator

```
<meta name="generator" content="HTML Generator 1.42">
```

Определяет один из пакетов программного обеспечения, используемых для создания документа. Только для страниц, где автоматически создается разметка.

keywords

Установите ключевые слова для поисковых систем (разделенные запятой):

```
<meta name="keywords" content="Keyword1, Keyword2">
```

Метатег `keywords` иногда используется поисковыми системами, чтобы знать поисковый запрос, который имеет отношение к вашей веб-странице.
Как правило, это, вероятно, хорошая идея не добавлять слишком много слов, так как большинство поисковых систем, которые используют этот метатег для индексирования, будут индексировать только первые ~ 20 слов. Убедитесь, что вы ставите самые важные ключевые слова.

Роботы

Атрибут `robots` , поддерживаемый несколькими основными поисковыми системами, контролирует, разрешено ли поисковым `robots` индексировать страницу или нет, и следует ли им следовать ссылкам со страницы или нет.

```
<meta name="robots" content="noindex">
```

В этом примере указывается, что все поисковые системы не отображают страницу в

результатах поиска. Другие допустимые значения:

Значение / Директива	Имея в виду
`all`	**По умолчанию.** Эквивалентен `index, follow` . См. Примечание ниже.
`noindex`	Не индексируйте страницу вообще.
`nofollow`	Не следуйте ссылкам на этой странице
`follow`	Ссылки на странице могут быть выполнены. См. Примечание ниже.
`none`	Эквивалентен `noindex, nofollow` .
`noarchive`	Не делайте кешированную версию этой страницы в результатах поиска.
`nocache`	Синоним `noarchive` используемый некоторыми ботами, такими как Bing.
`nosnippet`	Не показывайте фрагмент этой страницы в результатах поиска.
`noodp`	Не используйте метаданные этой страницы из проекта Open Directory для заголовков или фрагментов в результатах поиска.
`notranslate`	Не предлагайте переводы этой страницы в результатах поиска.
`noimageindex`	Не индексируйте изображения на этой странице.
`unavailable_after [RFC-850 date/time]`	Не показывать эту страницу в результатах поиска после указанной даты / времени. Дата / время должны быть указаны в формате RFC 850 .

Примечание. Явное определение `index` и / или `follow` , в то время как действительные значения не требуются, поскольку почти все поисковые системы предполагают, что им разрешено делать это, если это явно не предотвращено. Подобно тому, как работает файл robots.txt, поисковые системы обычно ищут только те вещи, которые им *не позволяют* делать. Только указание на то, что поисковая система не допускает, также предотвращает случайное указание противоположностей (таких как `index, ..., noindex`), которые не все поисковые системы будут обрабатывать одинаково.

Признание номера телефона

Мобильные платформы, такие как iOS, автоматически распознают номера телефонов и превращают их в `tel:` links. Хотя эта функция очень практична, система иногда обнаруживает коды ISBN и другие номера в качестве телефонных номеров.

Для мобильных Safari и некоторых других мобильных браузеров на основе WebKit, чтобы отключить автоматическое распознавание и форматирование номера телефона, вам нужен этот метатег:

```
<meta name="format-detection" content="telephone=no">
```

Социальные медиа

Open Graph - это стандарт для метаданных, который расширяет нормальную информацию, содержащуюся в разметке головы сайта. Это позволяет веб-сайтам, таким как Facebook, показывать более глубокую и богатую информацию о веб-сайте в структурированном формате. Эта информация автоматически отображается, когда пользователи обмениваются ссылками на веб-сайты, содержащие метаданные OG, на Facebook.

Facebook / Open Graph

```
<meta property="fb:app_id" content="123456789">
<meta property="og:url" content="https://example.com/page.html">
<meta property="og:type" content="website">
<meta property="og:title" content="Content Title">
<meta property="og:image" content="https://example.com/image.jpg">
<meta property="og:description" content="Description Here">
<meta property="og:site_name" content="Site Name">
<meta property="og:locale" content="en_US">
<meta property="article:author" content="">
<!-- Facebook: https://developers.facebook.com/docs/sharing/webmasters#markup -->
<!-- Open Graph: http://ogp.me/ -->
```

- Facebook Open Graph Markup
- Протокол Open Graph

Facebook / мгновенные статьи

```
<meta charset="utf-8">
<meta property="op:markup_version" content="v1.0">

<!-- The URL of the web version of your article -->
<link rel="canonical" href="http://example.com/article.html">

<!-- The style to be used for this article -->
<meta property="fb:article_style" content="myarticlestyle">
```

- Статьи на тему:

- Мгновенные статьи: ссылка на формат

Twitter использует собственную разметку для метаданных. Эти метаданные используются в качестве информации для управления отображением твитов, когда они содержат ссылку на сайт.

щебет

```
<meta name="twitter:card" content="summary">
<meta name="twitter:site" content="@site_account">
<meta name="twitter:creator" content="@individual_account">
<meta name="twitter:url" content="https://example.com/page.html">
<meta name="twitter:title" content="Content Title">
<meta name="twitter:description" content="Content description less than 200 characters">
<meta name="twitter:image" content="https://example.com/image.jpg">
```

- Карты Twitter: руководство по началу работы
- Валидатор карт Twitter

Google+ / Schema.org

```
<link href="https://plus.google.com/+YourPage" rel="publisher">
<meta itemprop="name" content="Content Title">
<meta itemprop="description" content="Content description less than 200 characters">
<meta itemprop="image" content="https://example.com/image.jpg">
```

Автоматическое перенаправление

Иногда вашей веб-странице требуется автоматическая переадресация.

Например, чтобы перенаправить на `example.com` через 5 секунд:

```
<meta http-equiv="refresh" content="5;url=https://www.example.com/" />
```

Это строка отправит вас на указанный сайт (в этом случае `example.com` через 5 секунд.

Если вам нужно изменить временную задержку перед перенаправлением, просто изменив число прямо перед вашим `;url=` изменит временную задержку.

Веб-приложение

Вы можете настроить свое веб-приложение или веб-сайт, чтобы добавить ярлык приложения, добавленный на рабочий экран устройства, и запустить приложение в полноэкранном режиме приложения с помощью пункта меню «Добавить в рабочий стол» для Chrome.

Ниже метатеги (ов) откроется веб-приложение в полноэкранном режиме (без адресной

строки).

Android Chrome

```
<meta name="mobile-web-app-capable" content="yes">
```

IOS

```
<meta name="apple-mobile-web-app-capable" content="yes">
```

Вы также можете установить цвет для строки состояния и адресной строки в метатеге.

Android Chrome

```
<meta name="theme-color" content="black">
```

IOS

```
<meta name="apple-mobile-web-app-status-bar-style" content="black">
```

Прочитайте Мета-информация онлайн: https://riptutorial.com/ru/html/topic/1264/мета-информация

глава 25: Навигационные бары

Examples

Базовая панель навигации

Панель навигации - это, по существу, список ссылок, поэтому элементы ul и li используются для привязки навигационных ссылок.

```
<ul>
  <li><a href="#">Home</a></li>
  <li><a href="#">About</a></li>
  <li><a href="#">Contact</a></li>
</ul>
```

Панель навигации HTML5

Чтобы создать навигационную панель с помощью элемента управления HTML5, закрепите ссылки в теге nav.

```
<nav>
  <a href="#">Home</a>
  <a href="#">About</a>
  <a href="#">Contact</a>
</nav>
```

Прочитайте Навигационные бары онлайн: https://riptutorial.com/ru/html/topic/10725/навигационные-бары

глава 26: Пункты

Вступление

Пункты являются самым основным элементом HTML. В этом разделе объясняется и демонстрируется использование элемента абзаца в HTML.

параметры

колонка	колонка
`<p>`	Определяет абзац
` `	Вставляет один разрыв строки
`<pre>`	Определяет предварительно форматированный текст

Examples

HTML-абзацы

Элемент HTML `<p>` определяет **абзац** :

```
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
```

Экран-

Вы не можете быть уверены, как будет отображаться HTML.

Большие или маленькие экраны и измененные окна создадут разные результаты.

С помощью HTML вы не можете изменить вывод, добавив лишние пробелы или дополнительные строки в свой HTML-код.

При отображении страницы браузер удалит лишние пробелы и дополнительные строки:

```
    <p>This is          another     paragraph, extra spaces   will be   removed  by
browsers</p>
```

Прочитайте Пункты онлайн: https://riptutorial.com/ru/html/topic/7997/пункты

глава 27: Пустотелые элементы

Вступление

Не все теги HTML имеют одинаковую структуру. Хотя для большинства элементов требуется тег открытия, закрывающий тег и содержимое, некоторые элементы, известные как элементы void, требуют только открывающего тега, поскольку они сами не содержат никаких элементов. В этом разделе объясняется и демонстрируется правильное использование элементов void в HTML

замечания

Элемент void не может иметь никакого содержимого, но может иметь атрибуты. Элементы Void являются самозакрывающимися, поэтому они не должны иметь закрывающий тег.

В HTML5 следующие элементы недействительны:

- `area`
- `base`
- `br`
- `col`
- `embed`
- `hr`
- `img`
- `input`
- `keygen`
- `link`
- `meta`
- `param`
- `source`
- `track`
- `wbr`

Examples

Недействительные элементы

HTML 4.01 / XHTML 1.0 Strict содержит следующие элементы void:

- `area` - кликабельная, определенная область в изображении
- `base` - указывает базовый URL-адрес, из которого все ссылки основаны
- `br` line break
- `col` - столбец в таблице [устаревший]
- `hr` - горизонтальное правило (строка)
- `img` - изображение
- `input`

- поле, в котором пользователи вводят данные
- `link` - связывает внешний ресурс с документом
- `meta` - предоставляет информацию о документе
- `param` - определяет параметры для плагинов

Стандарты HTML 5 включают все ненулевые теги из предыдущего списка и

- `command` - представляет собой команду, которую пользователи могут вызывать [устаревшие]
- `keygen` - упрощает создание открытых ключей для веб-сертификатов [устарело]
- `source` - указывает источники мультимедиа для элементов `picture` , `audio` и `video`

В приведенном ниже примере **не** указаны элементы void:

```
<div>
    <a href="http://stackoverflow.com/">
        <h3>Click here to visit <i>Stack Overflow!</i></h3>
    </a>
    <button onclick="alert('Hello!');">Say Hello!</button>
    <p>My favorite language is <b>HTML</b>. Here are my others:</p>
    <ol>
        <li>CSS</li>
        <li>JavaScript</li>
        <li>PHP</li>
    </ol>
</div>
```

Обратите внимание, что каждый элемент имеет открывающий тег, закрывающий тег и текст или другие элементы внутри открывающих и закрывающих тегов. Однако теги Void показаны в следующем примере:

```
<img src="https://cdn.sstatic.net/Sites/stackoverflow/company/img/logos/so/so-icon.png" />
<br>
<hr>
<input type="number" placeholder="Enter your favorite number">
```

За исключением тега img, все эти недействительные элементы имеют только открывающий тег. Тег img, в отличие от любого другого тега, имеет self-закрытие `/` перед значком открытого тега. Лучше всего иметь место перед косой чертой.

Прочитайте Пустотелые элементы онлайн: https://riptutorial.com/ru/html/topic/1449/пустотелые-элементы

глава 28: Связывание ресурсов

Вступление

Хотя многие скрипты, значки и таблицы стилей могут быть написаны прямо в разметке HTML, лучше всего использовать эти ресурсы в своем собственном файле и связывать их с вашим документом. В этом разделе рассматриваются ссылки на внешние ресурсы, такие как таблицы стилей и сценарии, в HTML-документ.

Синтаксис

- `<link rel="link-relation" type="mime-type" href="url">`
- `<script src="path-to-script"></script>`

параметры

атрибут	подробности
`charset`	Задает кодировку символов связанного документа
`crossorigin`	Указывает, как элемент обрабатывает запросы на кросс-поиск
`href`	Задает местоположение связанного документа
`hreflang`	Указывает язык текста в связанном документе
`media`	Определяет, на каком устройстве будет отображаться связанный документ, который часто используется при выборе таблиц стилей на основе соответствующего устройства
`rel`	**Требуется** . Задает связь между текущим документом и связанным документом
`rev`	Задает взаимосвязь между связанным документом и текущим документом
`sizes`	Задает размер связанного ресурса. Только когда `rel="icon"`
`target`	Указывает, где должен быть загружен связанный документ
`type`	Задает тип носителя связанного документа
`integrity`	Задает хэш-код с кодировкой base64 (sha256, sha384 или sha512) связанного ресурса, что позволяет браузеру проверить его легитимность.

Examples

Внешняя таблица стилей CSS

```
<link rel="stylesheet" href="path/to.css" type="text/css">
```

Стандартная практика заключается в размещении тегов CSS `<link>` внутри `<head>` в верхней части вашего HTML. Таким образом, CSS будет загружен первым и будет применяться к вашей странице по мере ее загрузки, вместо того, чтобы показывать неустановленный HTML до загрузки CSS. Атрибут `type` не нужен в HTML5, потому что HTML5 обычно поддерживает CSS.

```
 <link rel="stylesheet" href="path/to.css" type="text/css">
```

а также

```
 <link rel="stylesheet" href="path/to.css">
```

... сделать то же самое в HTML5.

Другой, хотя и менее распространенной практикой, является использование оператора `@import` внутри прямого CSS. Как это:

```
<style type="text/css">
    @import("path/to.css")
</style>

<style>
    @import("path/to.css")
</style>
```

JavaScript

синхронный

```
<script src="path/to.js"></script>
```

Стандартная практика заключается в размещении тегов JavaScript `<script>` непосредственно перед закрывающим `</body>` . Загрузка последних скриптов позволяет визуальным эффектам вашего сайта появляться быстрее и не позволяет вашему JavaScript работать с элементами, которые еще не загружены.

Асинхронный

```
<script src="path/to.js" async></script>
```

Другой вариант, когда загружаемый Javascript-код не нужен для инициализации страницы, он может быть загружен асинхронно, что ускоряет загрузку страницы. Используя `async` браузер будет загружать содержимое скрипта параллельно и, как только он будет полностью загружен, прервет разбор HTML, чтобы проанализировать файл Javascript.

Отложенный

```
<script src="path/to.js" defer></script>
```

Отложенные сценарии похожи на сценарии асинхронного программирования, за исключением того, что синтаксический анализ будет выполняться только после полного анализа HTML. Отложенные сценарии гарантированно загружаются в порядке объявления, так же как и синхронные скрипты.

<NoScript>

```
<noscript>JavaScript disabled</noscript>
```

Элемент `<noscript>` определяет контент, который будет отображаться, если пользователь отключил сценарии или браузер не поддерживает использование скриптов. Тег `<noscript>` может быть помещен либо в `<head>` либо в `<body>` .

Favicon

```
<link rel="icon" type="image/png" href="/favicon.png">
<link rel="shortcut icon" type="image/x-icon" href="/favicon.ico">
```

Используйте mime-type `image/png` для файлов PNG и `image/x-icon` для файлов с иконкой (`*.ico`). Разницу см. В этом вопросе SO .

Файл с именем `favicon.ico` в корневом каталоге вашего сайта обычно загружается и применяется автоматически, без необходимости использовать `<link>` . Если этот файл когда-либо изменится, браузеры могут быть медленными и упрямыми в обновлении своего кеша.

Альтернативный CSS

```
<link rel="alternate stylesheet" href="path/to/style.css" title="yourTitle">
```

Некоторые браузеры позволяют применять альтернативные таблицы стилей, если они

предлагаются. По умолчанию они не будут применяться, но обычно они могут быть изменены через настройки браузера:

> Firefox позволяет пользователю выбирать таблицу стилей с помощью подменю «Вид»> «Стиль страницы», Internet Explorer также поддерживает эту функцию (начиная с IE 8), также доступную из меню «Вид»> «Стиль страницы» (по крайней мере, с IE 11), но для Chrome требуется расширение используйте функцию (начиная с версии 48). Веб-страница также может предоставить собственный пользовательский интерфейс, чтобы пользователь мог переключать стили.

(Источник: Документы MDN)

Веб-фид

Используйте атрибут `rel="alternate"` чтобы обеспечить возможность обнаружения ваших фидов Atom / RSS.

```
<link rel="alternate" type="application/atom+xml" href="http://example.com/feed.xml" />
<link rel="alternate" type="application/rss+xml" href="http://example.com/feed.xml" />
```

См. Документы MDN для RSS-каналов и Atomic RSS .

Связать атрибут «media»

```
<link rel="stylesheet" href="test.css" media="print">
```

Media указывает, какую таблицу стилей следует использовать для какого типа носителя. Использование значения `print` будет отображать только эту таблицу стилей для страниц печати.

Значение этого атрибута может быть любым из значений `mediatype` (аналогично запросу в формате CSS).

Пред. И далее

Например, если страница является частью серии статей, можно использовать `prev` и `next` для указания страниц, которые появляются до и после.

```
<link rel="prev" href="http://stackoverflow.com/documentation/java/topics">

<link rel="next" href="http://stackoverflow.com/documentation/css/topics">
```

Совет ресурсов: dns-prefetch, prefetch, prerender

Preconnect

Отношение `preconnect` аналогично `dns-prefetch` в том, что оно разрешит DNS. Тем не менее, он также будет выполнять рукопожатие TCP и опциональное согласование TLS. Это экспериментальная функция.

```
<link rel="preconnect" href="URL">
```

DNS-Prefetch

Сообщает обозревателям, чтобы разрешить DNS для URL-адреса, чтобы все активы с этого URL загружались быстрее.

```
<link rel="dns-prefetch" href="URL">
```

Prefetch

Сообщает обозревателям, что данный ресурс должен быть предварительно загружен, чтобы его можно было загружать быстрее.

```
<link rel="prefetch" href="URL">
```

DNS-Prefetch разрешает только имя домена, тогда как prefetch загружает / сохраняет указанные ресурсы.

Предварительно обрабатывать

Сообщает обозревателям, чтобы они извлекали и отображали URL-адрес в фоновом режиме, чтобы они могли мгновенно доставляться пользователю, когда пользователь переходит к этому URL-адресу. Это экспериментальная функция.

```
<link rel="prerender" href="URL">
```

Прочитайте Связывание ресурсов онлайн: https://riptutorial.com/ru/html/topic/712/связывание-ресурсов

глава 29: Секционные элементы

замечания

Стандарты HTML5 не перечисляют основной элемент как элемент секционирования.

Examples

Элемент статьи

Элемент `<article>` содержит **автономный контент,** такой как статьи, сообщения в блогах, комментарии пользователей или интерактивный виджет, который может быть распространен вне контекста страницы, например, RSS.

- Когда элементы статьи вложены, содержимое внутреннего узла статьи должно быть связано с внешним элементом статьи.

Блог (`section`) с несколькими сообщениями (`article`) и комментариями (`article`) может выглядеть примерно так.

```
<section>
    <!-- Each individual blog post is an <article> -->
    <article>
        <header>
            <h1>Blog Post</h1>
            <time datetime="2016-03-13">13th March 2016</time>
        </header>

        <p>The article element represents a self contained article or document.</p>
        <p>The section element represents a grouping of content.</p>

        <section>
            <h2>Comments <small>relating to "Blog Post"</small></h2>

            <!-- Related comment is also a self-contained article -->
            <article id="user-comment-1">
                <p>Excellent!</p>
                <footer><p>...</p><time>...</time></footer>
            </article>
        </section>
    </article>

    <!-- ./repeat: <article> -->

</section>

<!-- Content unrelated to the blog or posts should be outside the section. -->
<footer>
    <p>This content should be unrelated to the blog.</p>
</footer>
```

Избегайте ненужного использования

Когда основное содержание страницы (исключая заголовки, нижние колонтитулы, панели навигации и т. Д.) - это просто одна группа элементов. Вы можете опустить `<article>` в пользу элемента `<main>` .

```
<article>
    <p>This doesn't make sense, this article has no real `context`.</p>
</article>
```

Вместо этого замените статью элементом `<main>` чтобы указать, что это `main` контент для этой страницы.

```
<main>
    <p>I'm the main content, I don't need to belong to an article.</p>
</main>
```

Если вы используете другой элемент, убедитесь, что вы указали роль `<main>` ARIA для правильной интерпретации и рендеринга между несколькими устройствами и браузерами без HTML5.

```
<section role="main">
    <p>This section is the main content of this page.</p>
</section>
```

Заметки:

- `<main>` потомки элементов не допускаются в `<article>`

Нажмите здесь, чтобы прочитать официальную спецификацию HTML5 для элемента `<article>`

Основной элемент

Элемент `<main>` содержит **основной контент** для вашей веб-страницы. Этот контент уникален для отдельной страницы и не должен появляться в другом месте на сайте. Повторение содержимого, такого как заголовки, нижние колонтитулы, навигация, логотипы и т. Д., Помещается вне элемента.

- Элемент `<main>` должен использоваться только **один раз** на одной странице.
- Элемент `<main>` не должен включаться в качестве потомка `article` , в `aside` , `footer` , элемента `header` или `nav` .

В следующем примере мы показываем **одну запись в блоге** (и соответствующую информацию, такую как ссылки и комментарии).

```
<body>
    <header>
        <nav>...</nav>
    </header>

    <main>
        <h1>Individual Blog Post</h1>
        <p>An introduction for the post.</p>

        <article>
            <h2>References</h2>
            <p>...</p>
        </article>

        <article>
            <h2>Comments</h2> ...
        </article>
    </main>

    <footer>...</footer>
</body>
```

- Сообщение в блоге содержится в элементе `<main>` чтобы указать, что это основной контент для этой страницы (и, следовательно, уникальный на веб-сайте).
- Теги `<header>` и `<footer>` являются *братьями* и *сестрами* для элемента `<main>` .

Заметки:

Спецификация HTML5 распознает элемент `<main>` как элемент **группировки** , а не элемент *секционирования* .

- Атрибуты роли ARIA : `main` *(по умолчанию)* , `presentation`

Добавление атрибута `role="main"` ARIA к **другим элементам,** предназначенным для использования в качестве основного контента, рекомендуется для помощи агентам пользователей, которые не поддерживают HTML5, а также для обеспечения большего контекста для тех, кто это делает.

Элемент `<main>` по умолчанию имеет основную роль, поэтому его не нужно предоставлять.

Нажмите здесь, чтобы прочитать официальную спецификацию HTML5 для элемента `<main>`

Nav Element

Элемент `<nav>` в основном предназначен для использования для разделов, содержащих **основные навигационные блоки** для веб-сайта, это может включать ссылки на другие части веб-страницы *(например, якоря для оглавления)* или на другие страницы целиком.

Встроенные элементы

Далее будет показан встроенный набор гиперссылок.

```
<nav>
    <a href="https://google.com">Google</a>
    <a href="https://www.yahoo.com">Yahoo!</a>
    <a href="https://www.bing.com">Bing</a>
</nav>
```

При необходимости используйте элементы списка

Если содержимое представляет список элементов, используйте элемент списка, чтобы показать это и улучшить работу пользователя.

Обратите внимание на `role="navigation"` , *подробнее об этом ниже.*

```
<nav role="navigation">
    <ul>
        <li><a href="https://google.com">Google</a></li>
        <li><a href="https://www.yahoo.com">Yahoo!</a></li>
        <li><a href="https://www.bing.com">Bing</a></li>
    </ul>
</nav>
```

Избегайте ненужного использования

Элементы `<footer>` могут иметь список ссылок на другие части сайта (FAQ, T & C и т. д.). В этом случае достаточно одного элемента нижнего колонтитула, вам больше не *нужно* обматывать ваши ссылки с помощью элемента `<nav>` в `<footer>` .

```
<!-- the <nav> is not required in the <footer> -->
<footer>
    <nav>
        <a href="#">...</a>
    </nav>
</footer>

<!-- The footer alone is sufficient -->
<footer>
    <a href="#">...</a>
</footer>
```

Заметки:

- `<main>` потомки элементов не допускаются в пределах `<nav>`

Добавление `role="navigation"` Роль ARIA для элемента `<nav>` рекомендуется помогать агентам пользователей, которые не поддерживают HTML5, а также предоставлять больше контекста тем, кто это делает.

```
<nav role="navigation"><!-- ... --></nav>
```

Screen Readers: *(программное обеспечение, которое позволяет слепым или слабовидящим пользователям перемещаться по сайту)*

Пользовательские агенты, такие как устройства чтения с экрана, будут интерпретировать элемент `<nav>` разному в зависимости от их требований.

- Это может придать элементу `<nav>` более высокий приоритет при рендеринге страницы
- Это может задержать визуализацию элемента
- Он может адаптировать страницу определенным образом для адаптации к потребностям пользователя
  пример: сделайте текстовые ссылки в элементах `<nav>` более крупными для тех, кто слабовиден.

Нажмите здесь, чтобы прочитать официальную спецификацию HTML5 для элемента `<nav>`

Элемент раздела

Элемент `<section>` представляет общий раздел для тематического группового содержимого. Как правило, каждый раздел должен быть идентифицирован с элементом заголовка в качестве дочернего элемента `section` .

- Вы можете использовать элемент `<section>` в `<article>` и наоборот.
- Каждый раздел должен иметь *тему* (элемент заголовка, определяющий этот регион)
- Не используйте элемент `<section>` в качестве общего контейнера «стиль».
  Если вам нужен контейнер для стилизации, используйте вместо него `<div>` .

В следующем примере мы показываем **одно сообщение в блоге** с несколькими главами, каждая глава представляет собой раздел *(набор тематически сгруппированного контента, который может быть идентифицирован элементами заголовка в каждом разделе)* .

```
<article>
    <header>
        <h2>Blog Post</h2>
    </header>
    <p>An introduction for the post.</p>
    <section>
        <h3>Chapter 1</h3>
        <p>...</p>
```

```
    </section>
    <section>
        <h3>Chapter 2</h3>
        <p>...</p>
    </section>
    <section>
        <h3>Comments</h3> ...
    </section>
</article>
```

> **Заметки:**
>
> Разработчики должны использовать элемент **статьи,** когда имеет смысл синдицировать содержимое элемента.

Нажмите здесь, чтобы прочитать официальную спецификацию HTML5 для элемента `<main>`

Элемент заголовка

Элемент `<header>` представляет собой вводный контент для его ближайшего содержимого для секреции предков или корневого элемента секции. А `<header>` обычно содержит группу вводных или навигационных средств.

> **Примечание.** Элемент header не является содержимым секции; он не вводит новый раздел.

Примеры:

```
<header>
  <p>Welcome to...</p>
  <h1>Voidwars!</h1>
</header>
```

В этом примере `<article>` имеет `<header>` .

```
<article>
  <header>
    <h1>Flexbox: The definitive guide</h1>
  </header>
  <p>The guide about Flexbox was supposed to be here, but it turned out Wes wasn't a Flexbox
expert either.</p>
</article>
```

Рекомендация W3C

Элемент нижнего колонтитула

Элемент `<footer>` содержит нижнюю часть страницы.

Ниже приведен пример элемента `<footer>` , содержащего тег `p` paragraph.

```
<footer>
    <p>All rights reserved</p>
</footer>
```

Прочитайте Секционные элементы онлайн: https://riptutorial.com/ru/html/topic/311/секционные-элементы

глава 30: Символьные объекты

Examples

Общие специальные символы

Некоторый символ может быть зарезервирован для HTML и не может использоваться напрямую, поскольку это может препятствовать действительным кодам HTML. Например, попытка отображения левого и правого угловых скобок (`<>`) в исходном коде может привести к неожиданным результатам на выходе. Аналогично, пробелы, записанные в исходном коде, могут не отображаться так, как ожидалось, в выходном HTML. Некоторые, например, ☎, недоступны в наборе символов ASCII.

Для этой цели создаются объекты символов. Они имеют вид `&entity_name;` или `&entity_number;` , Ниже перечислены некоторые из доступных HTML-объектов.

символ	Описание	Имя сущности	Номер объекта
""	неразрывное пространство	` `	` `
«<»	меньше, чем	`<`	`&#60;`
«>»	лучше чем	`>`	`&#62;`
«&»	амперсант	`&`	`&#38;`
«-»	em dash	`&mdash;`	`&#8212;`
«-»	en dash	`&ndash;`	`&#8211;`
«©»	авторское право	`&copy;`	`&#169;`
«®»	зарегистрированная торговая марка	`&reg;`	`&#174;`
«™»	товарный знак	`&trade;`	`&#8482;`
«☎»	Телефон	`&phone;`	`&#9742;`

Таким образом, чтобы написать

© 2016 Stack Exchange Inc.

используется следующий HTML-код:

```
<b>&copy; 2016 Stack Exchange Inc.</b>
```

Сущности символов в HTML

При разработке веб-страницы в html требуется много символов и специальных символов, но, как мы знаем, иногда использование символов напрямую может помешать фактическому html-коду, который содержит определенные символы, а также некоторые символы, недоступные на клавиатуре. Таким образом, чтобы избежать конфликта и в то же время, чтобы иметь возможность использовать разные символы в нашем коде, w3 org предоставляет нам «Сущности персонажа».

Объекты символов предопределены с помощью «Имя объекта» - & имя_объекта; и 'Entity Number' - & entity_number; поэтому нам нужно использовать любой из двух символов для отображения требуемого символа на нашей странице.

Список нескольких объектов символов можно найти по адресу https://dev.w3.org/html5/html-author/charref

Простой пример с использованием символьной сущности для «увеличительного стекла»:

```
<input type="text" placeholder="  &#128269; Search"/>
```

который

Прочитайте Символьные объекты онлайн: https://riptutorial.com/ru/html/topic/5229/символьные-объекты

глава 31: Списки

Вступление

HTML предлагает три способа указания списков: упорядоченные списки, неупорядоченные списки и списки описания. В упорядоченных списках используются порядковые последовательности, указывающие порядок элементов списка, неупорядоченные списки используют определенный символ, такой как пуля, чтобы перечислять элементы в не обозначенном порядке, а в списках описания используются отступы для перечисления элементов со своими дочерними элементами. В этом разделе объясняется реализация и комбинация этих списков в разметке HTML.

Синтаксис

- `<ol> ordered list items </ol>`
- `<ul> unordered list items </ul>`
- `<li> list item (ordered and not) </li>`
- `<dl> description list items </dl>`
- `<dt> description list title </dt>`
- `<dd> description list description </dd>`

замечания

Смотрите также

Вы можете добавить свойство CSS типа стиля в `<ul>` , чтобы изменить, какой значок используется для отметки каждого элемента списка, например `<ul style="list-style-type:disc">` . Разрешены следующие типы списка:

- «list-style-type: disc» - это точка по умолчанию.
- «list-style-type: circle» - это незаполненный круг.
- «list-style-type: square» - это заполненный квадрат.
- «Тип списка-типа: нет» не использует никакой отметки.

Вы также можете добавить атрибут типа в `<ol>` , чтобы изменить способ нумерации, например `<ol type="1">` . Разрешены следующие типы:

- Тип = "1" является формой по умолчанию.
- type = "A" использует заглавные буквы в алфавитном порядке
- type = "a" использует строчные буквы в алфавитном порядке
- type = "I" использует римские цифры с прописными буквами
- type = "i" использует римские цифры с строчными буквами

Examples

Неупорядоченный список

Неупорядоченный список может быть создан с помощью `<ul>` и каждый элемент списка может быть создан с помощью `<li>` как показано в следующем примере:

```
<ul>
  <li>Item</li>
  <li>Another Item</li>
  <li>Yet Another Item</li>
</ul>
```

Это создаст маркированный список (который является стандартом по умолчанию):

- Вещь
- Другой предмет
- Еще один пункт

> Вы должны использовать `ul` для отображения списка элементов, где порядок элементов не важен. Если изменение порядка элементов делает неверный список, вы должны использовать `<ol>` .

Упорядоченный список

Упорядоченный список может быть создан с помощью `<ol>` и каждый элемент списка может быть создан с помощью `<li>` как в приведенном ниже примере:

```
<ol>
  <li>Item</li>
  <li>Another Item</li>
  <li>Yet Another Item</li>
</ol>
```

Это приведет к пронумерованному списку (который является стилем по умолчанию):

1. Вещь
2. Другой предмет
3. Еще один пункт

Ручное изменение номеров

Есть несколько способов, которыми вы можете играть, номера которых отображаются в списках в упорядоченном списке. Первый способ - установить начальный номер, используя атрибут `start` . Список начнется с этого определенного числа и продолжит увеличиваться на единицу, как обычно.

```
<ol start="3">
  <li>Item</li>
  <li>Some Other Item</li>
  <li>Yet Another Item</li>
</ol>
```

Это приведет к пронумерованному списку (который является стилем по умолчанию):

3. Вещь
4. Некоторые другие предметы
5. Еще один пункт

Вы также можете явно указать определенный элемент списка на определенный номер. Другие элементы списка после одного с указанным значением будут продолжать увеличиваться на единицу из значения этого элемента списка, игнорируя, где находился родительский список.

```
<li value="7"></li>
```

Также стоит отметить, что, используя атрибут `value` непосредственно в элементе списка, вы можете переопределить существующую систему нумерации упорядоченного списка, перезапустив нумерацию с меньшим значением. Поэтому, если родительский список уже достиг значения 7 и столкнулся с элементом списка со значением 4, этот элемент списка все равно будет отображаться как 4 и продолжить отсчет с этой точки снова.

```
<ol start="5">
  <li>Item</li>
  <li>Some Other Item</li>
  <li value="4">A Reset Item</li>
  <li>Another Item</li>
  <li>Yet Another Item</li>
</ol>
```

Таким образом, в приведенном выше примере будет создан список, который следует шаблону нумерации 5, 6, 4, 5, 6 - начиная с номера ниже предыдущего и дублируя число 6 в списке.

Примечание. Атрибуты `start` и `value` принимают только число - даже если упорядоченный список настроен на отображение в виде римских цифр или букв.

5

Вы можете полностью изменить нумерацию путем добавления `reversed` в вашем `ol` элементе:

```
<ol reversed>
  <li>Item</li>
  <li>Some Other Item</li>
  <li value="4">A Reset Item</li>
```

```
  <li>Another Item</li>
  <li>Yet Another Item</li>
</ol>
```

Обратная нумерация полезна, если вы постоянно добавляете в список, например, с новыми эпизодами или презентациями подкаста, и сначала хотите, чтобы появились самые последние элементы.

Изменение типа цифры

Вы можете легко изменить тип цифры, показанной в марке элемента списка, используя атрибут `type`

```
<ol type="1|a|A|i|I">
```

Тип	Описание	Примеры
`1`	Значение по умолчанию - десятичные числа	1,2,3,4
`a`	В алфавитном порядке (в нижнем регистре)	а, б, в, г
`A`	В алфавитном порядке (в верхнем регистре)	A, B, C, D
`i`	Римские цифры (строчные буквы)	I, II, III, IV
`I`	Римские цифры (в верхнем регистре)	I, II, III, IV

> Вы должны использовать `ol` для отображения списка предметов, где предметы были преднамеренно заказаны, а заказ должен быть выделен. Если изменение порядка элементов НЕ делает неверным список, вы должны использовать `<ul>` .

Описание Список

Список описаний (или *список определений* , который был вызван до HTML5), может быть создан с помощью элемента `dl` . Он состоит из групп имен и значений, где имя указано в элементе `dt` , а значение задается в элементе `dd` .

```
<dl>
  <dt>name 1</dt>
  <dd>value for 1</dd>
  <dt>name 2</dt>
  <dd>value for 2</dd>
</dl>
```

Демо-версия

Группа с именем-именем может иметь более одного имени и / или более одного значения (представляющего альтернативы):

```
<dl>

  <dt>name 1</dt>
  <dt>name 2</dt>
  <dd>value for 1 and 2</dd>

  <dt>name 3</dt>
  <dd>value for 3</dd>
  <dd>value for 3</dd>

</dl>
```

Демо-версия

Вложенные списки

Вы можете вставлять списки для представления подпунктов элемента списка.

```
<ul>
  <li>item 1</li>
  <li>item 2
    <ul>
      <li>sub-item 2.1</li>
      <li>sub-item 2.2</li>
    </ul>
  </li>
  <li>item 3</li>
</ul>
```

- пункт 1
- пункт 2
  - подпункт 2.1
  - подпункт 2.2
- пункт 3

Вложенный список должен быть дочерним элементом элемента `li` .

Вы можете также вставлять различные типы списков:

```
<ol>
    <li>Hello, list!</li>
    <li>
        <ul>
            <li>Hello, nested list!</li>
        </ul>
    </li>
</ol>
```

Прочитайте Списки онлайн: https://riptutorial.com/ru/html/topic/393/списки

глава 32: таблицы

Вступление

Элемент HTML `<table>` позволяет веб-авторам отображать табличные данные (например, текст, изображения, ссылки, другие таблицы и т. Д.) В двумерной таблице со строками и столбцами ячеек.

Синтаксис

- `<table></table>`
- `<thead></thead>`
- `<tbody></tbody>`
- `<tfoot></tfoot>`
- `<tr></tr>`
- `<th></th>`
- `<td></td>`

замечания

Различные элементы таблицы и их атрибуты содержимого вместе определяют модель таблицы. Элемент `<table>` - это элемент контейнера для табличных моделей / табличных данных. Таблицы содержат строки, столбцы и ячейки, данные их потомками. Строки и столбцы образуют сетку; ячейки таблицы должны полностью покрывать эту сетку без перекрытия. В приведенном ниже списке описаны различные элементы таблицы:

- `<table>` - Элемент контейнера для табличных моделей / табличных данных. `<table>` представляет данные с более чем одним измерением в виде таблицы.
- `<caption>` - заголовок или заголовок таблицы (как `figcaption` к `figure`)
- `<col>` - столбец (элемент без содержимого)
- `<colgroup>` - группировка столбцов
- `<thead>` - Заголовок таблицы (только один)
- `<tbody>` - Тело / содержимое таблицы (несколько в порядке)
- `<tfoot>` - `<tfoot>` колонтитул стола (только один)
- `<tr>` - Таблица строк
- `<th>` - ячейка заголовка таблицы
- `<td>` - ячейка данных таблицы

Семантически таблицы предназначены для хранения табличных данных. Вы можете думать об этом как о способе отображения и описания данных, которые имели бы смысл в электронной таблице (столбцах и строках).

Использование таблиц для макета не рекомендуется. Вместо этого используйте правила

CSS для макета и форматирования, включая `display: table` .

Одно замечательное исключение, обычно отображаемое в отрасли относительно использования макета `<table>` относится к электронной почте HTML: некоторые почтовые клиенты, включая Outlook, откатываются назад к более старым механизмам рендеринга после того, как Microsoft потеряет свое монопольное дело против ЕС. Для того, чтобы Microsoft не включила IE в ОС, они просто отбросили механизм рендеринга Outlook к более ранней версии Trident. Этот откат просто не поддерживает современные веб-технологии, поэтому использование макетов на основе `<table>` для электронной почты HTML - единственный способ обеспечить совместимость между браузерами и платформой / клиентом.

Examples

Простая таблица

```
<table>
  <tr>
    <th>Heading 1/Column 1</th>
    <th>Heading 2/Column 2</th>
  </tr>
  <tr>
    <td>Row 1 Data Column 1</td>
    <td>Row 1 Data Column 2</td>
  </tr>
  <tr>
    <td>Row 2 Data Column 1</td>
    <td>Row 2 Data Column 2</td>
  </tr>
</table>
```

Это приведет к тому, что `<table>` состоит из трех полных строк (`<tr>`): одна строка ячеек заголовка (`<th>`) и две строки ячеек содержимого (`<td>`). `<th>` - это *табличные заголовки,* а `<td>` - *табличные данные* . Вы можете поместить все, что хотите, внутри `<td>` или `<th>` .

Заголовок 1 / Столбец 1	**Заголовок 2 / Столбец 2**
Строка данных 1-го столбца 1	Строка данных 1-го столбца 2
Строка данных 2 строки 1	Строка данных 2 строки 2

Закрепление столбцов или строк

Ячейки таблицы могут охватывать несколько столбцов или строк, используя атрибуты `colspan` и `rowspan` . Эти атрибуты могут быть применены к элементам `<th>` и `<td>` .

```
<table>
```

```
    <tr>
        <td>row 1 col 1</td>
        <td>row 1 col 2</td>
        <td>row 1 col 3</td>
    </tr>
    <tr>
        <td colspan="3">This second row spans all three columns</td>
    </tr>
    <tr>
        <td rowspan="2">This cell spans two rows</td>
        <td>row 3 col 2</td>
        <td>row 3 col 3</td>
    </tr>
    <tr>
        <td>row 4 col 2</td>
        <td>row 4 col 3</td>
    </tr>

</table>
```

В результате

row 1 col 1	row 1 col 2	row 1 col 3
This second row spans all three columns		
This cell spans two rows	row 3 col 2	row 3 col 3
	row 4 col 2	row 4 col 3

Обратите внимание, что вам не следует создавать таблицу, в которой как строки, так и столбцы перекрываются, поскольку это недопустимый HTML, и результат обрабатывается по-разному разными веб-браузерами.

`rowspan` = неотрицательное целое число, определяющее количество строк, натянутых на ячейку. Значение по умолчанию этого атрибута равно 1 (`1`). Значение 0 (`0`) означает, что ячейка будет продолжаться от текущей строки до последней строки таблицы (`<thead>` , `<tbody>` или `<tfoot>`).

`colspan` = неотрицательное целое число, определяющее количество столбцов, отнесенных текущей ячейкой. Значение по умолчанию этого атрибута равно 1 (`1`). Значение 0 (`0`) означает, что ячейка будет продолжаться от текущего до последнего столбца группы столбцов `<colgroup>` в которой определена ячейка.

Таблица с темой, tbody, tfoot и заголовком

HTML также таблицы с `<thead>` , `<tbody>` , `<tfoot>` и `<caption>` элементы. Эти дополнительные элементы полезны для добавления семантического значения в ваши таблицы и для обеспечения места для отдельного стили CSS.

При распечатке таблицы, которая не помещается на одну (бумажную) страницу, большинство браузеров повторяют содержимое `<thead>` на каждой странице.

Есть определенный порядок, которому нужно придерживаться, и мы должны знать, что не каждый элемент встает на свои места, как можно было бы ожидать. Следующий пример демонстрирует, как должны быть размещены наши 4 элемента.

```
<table>
 <caption>Table Title</caption> <!--| caption is the first child of table |-->
  <thead> <!--=======================| thead is after caption |-->
    <tr>
      <th>Header content 1</th>
      <th>Header content 2</th>
    </tr>
  </thead>

  <tbody> <!--=======================| tbody is after thead |-->
    <tr>
      <td>Body content 1</td>
      <td>Body content 2</td>
    </tr>
  </tbody>

  <tfoot><!--| tfoot can be placed before or after tbody, but not in a group of tbody. |-->
<!--| Regardless where tfoot is in markup, it's rendered at the bottom. |-->

    <tr>
      <td>Footer content 1</td>
      <td>Footer content 2</td>
    </tr>
  </tfoot>

</table>
```

Результаты следующего примера демонстрируются дважды: в первой таблице отсутствуют стили, вторая таблица имеет несколько примененных свойств CSS: `background-color` , `color` и `border` *. Стили представлены в виде визуального руководства и не являются существенным аспектом данной темы.

Table Title

Header content 1	Header content 2
Body content 1	Body content 2
Footer content 1	Footer content 2

Table Title	
Header content 1	**Header content 2**
Body content 1	Body content 2
Footer content 1	Footer content 2

Элемент	Стили
`<caption>`	Желтый текст на черном фоне.

Элемент	Стили
`<thead>`	Полужирный текст на фиолетовом фоне.
`<tbody>`	Текст на синем фоне.
`<tfoot>`	Текст на зеленом фоне.
`<th>`	Оранжевые границы.
`<td>`	Красные границы.

Группы столбцов

Иногда вы можете применить стиль к столбцу или группе столбцов. Или для семантических целей вы можете группировать столбцы вместе. Для этого используйте `<colgroup>` и `<col>` .

Дополнительный `<colgroup>` позволяет группировать столбцы вместе. `<colgroup>` должны быть дочерними элементами `<table>` и должны появляться после любых элементов `<caption>` и перед любым содержимым таблицы (например, `<tr>` , `<thead>` , `<tbody>` и т. д.).

```
<table>
    <colgroup span="2"></colgroup>
    <colgroup span="2"></colgroup>
    ...
</table>
```

Дополнительный `<col>` позволяет ссылаться на отдельные столбцы или диапазон столбцов без применения логической группировки. Элементы `<col>` являются необязательными, но если они присутствуют, они должны находиться внутри элемента `<colgroup>` .

```
<table>
    <colgroup>
        <col id="MySpecialColumn" />
        <col />
    </colgroup>
    <colgroup>
        <col class="CoolColumn" />
        <col class="NeatColumn" span="2" />
    </colgroup>
    ...
</table>
```

Следующие стили CSS могут применяться к `<colgroup>` и `<col>` :

- `border`
- `background`
- `width`

- `visibility`
- `display` (как на `display: none`)
    - `display: none;` фактически удалит столбцы с экрана, в результате чего таблица будет отображаться так, как если бы эти ячейки не существовали

Для получения дополнительной информации см. Табличные данные HTML5 .

Область охвата

`th` элемента очень часто используются для обозначения заголовков строк и столбцов таблицы, например:

```
<table>
    <thead>
        <tr>
            <td></td>
            <th>Column Heading 1</th>
            <th>Column Heading 2</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <th>Row Heading 1</th>
            <td></td>
            <td></td>
        </tr>
        <tr>
            <th>Row Heading 2</th>
            <td></td>
            <td></td>
        </tr>
    </tbody>
</table>
```

Это можно улучшить для доступности с помощью атрибута `scope` . В приведенном выше примере будут внесены следующие изменения:

```
<table>
    <thead>
        <tr>
            <td></td>
            <th scope="col">Column Heading 1</th>
            <th scope="col">Column Heading 2</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <th scope="row">Row Heading 1</th>
            <td></td>
            <td></td>
        </tr>
        <tr>
            <th scope="row">Row Heading 1</th>
```

```
            <td></td>
            <td></td>
        </tr>
    </tbody>
</table>
```

`scope` известен как *перечислимый атрибут* , что означает, что он может иметь значение из определенного набора возможных значений. Этот набор включает:

- `col`
- `row`
- `colgroup`
- `rowgroup`

Рекомендации:

- https://developer.mozilla.org/en-US/docs/Web/HTML/Element/th#attr-scope
- https://www.w3.org/TR/WCAG20-TECHS/H63.html

Прочитайте таблицы онлайн: https://riptutorial.com/ru/html/topic/274/таблицы

глава 33: Форматирование текста

Вступление

Хотя большинство HTML-тегов используются для создания элементов, HTML также предоставляет теги форматирования в тексте для применения определенных текстовых стилей к частям текста. В этом разделе приведены примеры форматирования текста в формате HTML, такие как выделение, выделение жирным шрифтом, подчеркивание, индекс и разбитый текст.

Синтаксис

- `<abbr>Abbreviation</abbr>`
- `<b>Bold Text</b>`
- `<del>Deleted Text</del>`
- `<em>Emphasized Text</em>`
- `<i>Italic Text</i>`
- `<ins>Inserted Text</ins>`
- `<mark>Marked (or Highlighted) Text</mark>`
- `<s>Stricken Text</s>`
- `<strong>Strong Text</strong>`
- `<sub>Subscript Text</sub>`
- `<sup>Superscript Text</sup>`
- `<u>Underlined Text</u>`

Examples

Жирный, Курсив и Подчеркивание

Жирный текст

Для полужирного текста используйте теги `<strong>` или `<b>` :

```
<strong>Bold Text Here</strong>
```

или же

```
<b>Bold Text Here</b>
```

Какая разница? Семантика. `<strong>` используется для обозначения того, что текст имеет фундаментальное или семантическое *значение* для окружающего текста, а `<b>` указывает на такое значение и просто представляет текст, который должен быть выделен полужирным шрифтом.

Если бы вы использовали `<b>` программа «текст в речь» не произносила бы слово (слова) не иначе, как любое из других слов вокруг него - вы просто обращаете на них внимание, не добавляя никакой дополнительной важности. Используя `<strong>` , тем не менее, одна и та же программа хотела бы произнести эти слова другим голосом, чтобы передать, что текст в какой-то мере важен.

Курсивный текст

Для наложения курсивом используйте теги `<em>` или `<i>` :

```
<em>Italicized Text Here</em>
```

или же

```
<i>Italicized Text Here</i>
```

Какая разница? Семантика. `<em>` используется, чтобы указать, что текст должен иметь дополнительный акцент, который следует подчеркнуть, а `<i>` просто представляет текст, который должен быть удален из обычного текста вокруг него.

Например, если вы хотите подчеркнуть действие внутри предложения, можно сделать это, выделив его курсивом через `<em>` : «Вы уже *представили* бы редактирование уже?»

Но если бы вы идентифицировали книгу или газету, которую обычно выделяли стилистически, вы просто использовали бы `<i>` : «Я был вынужден читать *Ромео и Джульетту* в старшей школе.

Подчеркнутый текст

Хотя сам элемент `<u>` устарел в HTMl 4, он был повторно введен с альтернативным смысловым значением в HTML 5 - для представления неаркулированной, нетекстовой аннотации. Вы можете использовать такой рендеринг для обозначения текста с ошибками на странице или для отметки имени пользователя в Китае.

```
<p>This paragraph contains some <u>mispelled</u> text.</p>
```

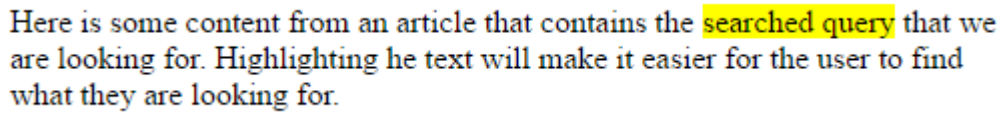
Выделив

Элемент `<mark>` является новым в HTML5 и используется для выделения или выделения текста в документе «из-за его значимости в другом контексте». [1]

Наиболее распространенным примером было бы в результатах поиска, когда пользователь ввел поисковый запрос, и результаты показаны с указанием желаемого запроса.

```
<p>Here is some content from an article that contains the <mark>searched query</mark>
that we are looking for. Highlighting the text will make it easier for the user to
find what they are looking for.</p>
```

Выход:



Обычное стандартное форматирование - черный текст на желтом фоне, но это можно изменить с помощью CSS.

Вставить, удалить или удалить

Чтобы пометить текст как вставленный, используйте `<ins>` :

```
<ins>New Text</ins>
```

Чтобы пометить текст как удаленный, используйте `<del>` :

```
<del>Deleted Text</del>
```

Чтобы пробить текст, используйте `<s>` :

```
<s>Struck-through text here</s>
```

Верхний индекс и подстрочный индекс

Чтобы компенсировать текст вверх или вниз, вы можете использовать теги `<sup>` и `<sub>` .

Чтобы создать надстрочный индекс:

```
<sup>superscript here</sup>
```

Чтобы создать индекс:

```
<sub>subscript here</sub>
```

Сокращение

Чтобы отметить какое-то выражение как аббревиатуру, используйте `<abbr>` :

```
<p>I like to write <abbr title="Hypertext Markup Language">HTML</abbr>!</p>
```

Если присутствует, атрибут `title` используется для представления полного описания такой аббревиатуры.

Прочитайте Форматирование текста онлайн: https://riptutorial.com/ru/html/topic/526/форматирование-текста

глава 34: формы

Вступление

Чтобы группировать элементы ввода и отправлять данные, HTML использует элемент формы для инкапсуляции элементов ввода и представления. Эти формы обрабатывают отправку данных в указанном методе на страницу, обрабатываемую сервером или обработчиком. В этом разделе объясняется и демонстрируется использование форм HTML при сборе и отправке входных данных.

Синтаксис

- `<form method="post|get" action="somePage.php" target="_blank|_self|_parent|_top|framename">`

параметры

атрибут	Описание
`accept-charset`	Задает кодировки символов, которые будут использоваться для отправки формы.
`action`	Указывает, куда отправлять данные формы при отправке формы.
`autocomplete`	Указывает, должна ли форма включать или выключать автозаполнение.
`enctype`	Указывает, как данные формы должны быть закодированы при отправке на сервер (только для метода = "post").
`method`	Указывает метод HTTP, который будет использоваться при отправке данных формы (POST или GET).
`name`	Задает имя формы.
`novalidate`	Указывает, что форма не должна быть проверена при отправке.
`target`	Определяет, где отображать ответ, который получен после отправки формы.

замечания

Элемент `<form>` представляет собой раздел, содержащий элементы, связанные с формой

(например, `<button>` `<fieldset>` `<input>` `<label>` `<output>` `<select>` `<textarea>`), который передает информацию серверу. Требуются теги start (`<form>`) и end (`</form>`).

Examples

предоставление

Атрибут действия

Атрибут action определяет действие, которое должно выполняться при отправке формы, что обычно приводит к скрипту, который собирает представленную информацию и работает с ней. если оставить его пустым, он отправит его в тот же файл

```
<form action="action.php">
```

Атрибут метода

Атрибут метода используется для определения HTTP-метода формы, которая является либо GET, либо POST.

```
<form action="action.php" method="get">
<form action="action.php" method="post">
```

Метод GET в основном используется для *получения* данных, например, для получения сообщения по его идентификатору или имени или для отправки поискового запроса. Метод GET добавит данные формы к URL-адресу, указанному в атрибуте действия.

```
www.example.com/action.php?firstname=Mickey&lastname=Mouse
```

Метод POST используется при отправке данных в скрипт. Метод POST не добавляет данные формы к URL-адресу действия, а отправляет его с использованием тела запроса.

Чтобы правильно отправить данные из формы, необходимо указать имя атрибута имени. В качестве примера давайте отправьте значение поля и задайте его имя в *lastname* :

```
<input type="text" name="lastname" value="Mouse">
```

Дополнительные атрибуты

```
<form action="action.php" method="post" target="_blank" accept-charset="UTF-8"
enctype="application/x-www-form-urlencoded" autocomplete="off" novalidate>

<!-- form elements -->
```

```
</form>
```

Атрибут Target в теге формы

Атрибут target указывает имя или ключевое слово, которое указывает, где отобразить ответ, полученный после отправки формы.

Атрибут target определяет имя или ключевое слово для контекста просмотра (например, вкладка, окно или встроенный фрейм).

Из тега с целевым атрибутом:

```
<form target="_blank">
```

Значения атрибутов

Значение	Описание
_blank	Ответ отображается в новом окне или вкладке
_self	Ответ отображается в том же фрейме (это по умолчанию)
_parent	Ответ отображается в родительском кадре
_Топ	Ответ отображается в полном объеме окна
framename	Ответ отображается в именованном iframe

> Примечание. Атрибут target был ***устарел*** в **HTML 4.01** . Атрибут target ***поддерживается*** в ***HTML5*** .
>
> Фреймы и фреймы не поддерживаются в ***HTML5*** , поэтому значения ***_parent, _top и frameename теперь в основном используются с iframes*** .

Загрузка файлов

Изображения и файлы могут быть загружены / отправлены на сервер, установив атрибут `enctype` тега `form` в `multipart/form-data` . `enctype` указывает, как данные формы будут закодированы при отправке на сервер.

пример

```
<form method="post" enctype="multipart/form-data" action="upload.php">
    <input type="file" name="pic" />
    <input type="submit" value="Upload" />
</form>
```

Группировка нескольких полей ввода

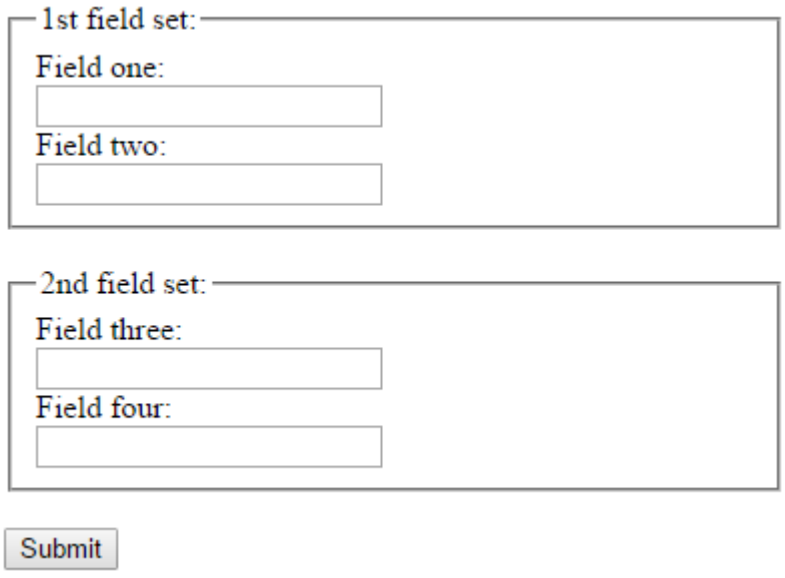
При разработке формы вы можете группировать несколько полей ввода в группу, чтобы помочь организовать макет формы. Это можно сделать, используя тег. Вот пример его использования.

Для каждого набора полей вы можете установить легенду для набора с помощью тега LEGEND TEXT

пример

```
<form>
  <fieldset>
     <legend>1st field set:</legend>
     Field one:<br>
     <input type="text"><br>
     Field two:<br>
     <input type="text"><br>
  </fieldset><br>
  <fieldset>
     <legend>2nd field set:</legend>
     Field three:<br>
     <input type="text"><br>
     Field four:<br>
     <input type="text"><br>
  </fieldset><br>
  <input type="submit" value="Submit">
</form>
```

Результат



Поддержка браузера

Chrome, IE, Edge, FireFox, Safari и последние версии Opera также поддерживают тег

Прочитайте формы онлайн: https://riptutorial.com/ru/html/topic/1160/формы

глава 35: холст

параметры

атрибут	Описание
рост	Определяет высоту холста
ширина	Задает ширину холста

замечания

- Этот тег несовместим с версиями Internet Explorer менее 9. Проверьте caniuse.com на совместимость с браузером.
- `canvas` - это только контейнер для графики, а фактический рисунок графики выполняется с помощью JavaScript.

Examples

Основной пример

Элемент `canvas` был введен в HTML5 для рисования графики.

```
<canvas id="myCanvas">
   Cannot display graphic. Canvas is not supported by your browser (IE<9)
</canvas>
```

Вышеупомянутый элемент создаст прозрачный элемент HTML `<canvas>` размером 300 × 150 пикселей.

Вы можете использовать элемент **canvas** для рисования потрясающих материалов, таких как фигуры, графики, манипуляции изображениями, создания привлекательных игр и т. Д. С помощью **JavaScript** .
Двумерная поверхность *слоя* `canvas` называется `CanvasRenderingContext2D` ; или из `HTMLCanvasElement` с использованием `.getContext("2d")` :

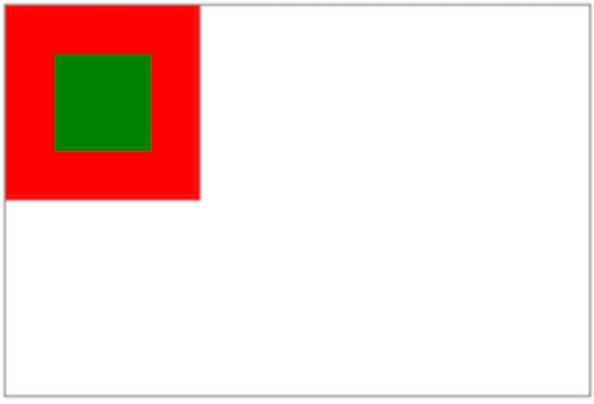
```
var ctx = document.getElementById("myCanvas").getContext("2d");
// now we can refer to the canvas's 2D layer context using `ctx`

ctx.fillStyle = "#f00";
ctx.fillRect(0, 0, ctx.canvas.width, ctx.canvas.height); // x, y, width, height

ctx.fillStyle = "#000";
ctx.fillText("My red canvas with some black text", 24, 32); // text, x, y
```

Пример jsFiddle

Рисование двух прямоугольников на

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <title>Draw two rectangles on the canvas</title>
    <style>
      canvas{
          border:1px solid gray;
      }
    </style>
    <script async>
      window.onload = init; // call init() once the window is completely loaded
      function init(){
        // #1 - get reference to <canvas> element
        var canvas = document.querySelector('canvas');

        // #2 - get reference to the drawing context and drawing API
        var ctx = canvas.getContext('2d');

        // #3 - all fill operations are now in red
        ctx.fillStyle = 'red';

        // #4 - fill a 100x100 rectangle at x=0,y=0
        ctx.fillRect(0,0,100,100);

        // #5 - all fill operations are now in green
        ctx.fillStyle = 'green';

        // #6 - fill a 50x50 rectangle at x=25,y=25
        ctx.fillRect(25,25,50,50);

      }
      </script>
</head>
<body>
  <canvas width=300 height=200>Your browser does not support canvas.</canvas>
</body>
</html>
```

Этот пример выглядит следующим образом:



Прочитайте холст онлайн: https://riptutorial.com/ru/html/topic/1162/холст

глава 36: Элемент Div

Вступление

Элемент div в HTML является элементом контейнера, который инкапсулирует другие элементы и может использоваться для группировки и разделения частей веб-страницы. Сам div сам по себе не представляет ничего, но является мощным инструментом в веб-дизайне. В этом разделе описывается назначение и применение элемента div.

Синтаксис

- `<div>example div</div>`

Examples

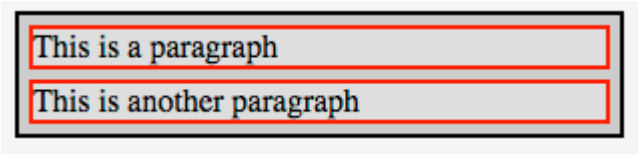
гнездование

Общепринятой практикой является размещение нескольких `<div>` внутри другого `<div>` . Обычно это называют «вложенными» элементами и позволяет дополнительно делить элементы на подразделы или помогать разработчикам с помощью стилей CSS.

`<div class="outer-div">` используется для группировки двух элементов `<div class="inner-div">` ; каждый из которых содержит элемент `<p>` .

```
<div class="outer-div">
  <div class="inner-div">
    <p>This is a paragraph</p>
  </div>
  <div class="inner-div">
    <p>This is another paragraph</p>
  </div>
</div>
```

Это даст следующий результат (стили CSS для ясности):



Вложение встроенных и блочных элементов. В то время как **элементы** вложенности вы должны иметь в виду, что есть встроенные и блокирующие элементы. в то время как элементы блока «добавляют разрыв строки в фоновом режиме», что означает, что другие вложенные элементы отображаются в следующей строке автоматически, встроенные элементы могут быть расположены рядом друг с другом по умолчанию

Избегайте глубокого гнезда `<div>`

Глубокие и часто используемые макеты вложенных контейнеров демонстрируют плохой стиль кодирования.

Закругленные углы или некоторые подобные функции часто создают такой HTML-код. Для большинства браузеров последнего поколения есть CSS3-аналоги. Постарайтесь использовать как можно меньше HTML-элементов для увеличения отношения контента к тегам и уменьшения загрузки страницы, что приведет к лучшему ранжированию в поисковых системах.

Элемент `div` Элемент должен быть не вложен глубже 6 слоев.

Основное использование

Элемент `<div>` обычно не имеет специфического семантического значения сам по себе, просто представляет собой разделение и обычно используется для группировки и инкапсуляции других элементов в документе HTML и разделения их на другие группы контента. Таким образом, каждый `<div>` лучше всего описывается его содержимым.

```
<div>
  <p>Hello! This is a paragraph.</p>
</div>
```

Элемент `div` обычно является элементом уровня блока , что означает, что он разделяет блок документа HTML и занимает максимальную ширину страницы. Браузеры обычно имеют следующее правило CSS по умолчанию:

```
div {
  display: block;
}
```

> Это настоятельно рекомендуется **консорциумом World Wide Web (W3C)** для просмотра элемента div как элемента последней инстанции, поскольку ни один другой элемент не подходит. Использование более подходящих элементов вместо элемента div приводит к улучшению доступности для читателей и упрощению обслуживания для авторов.

Например, сообщение в блоге будет помечено с помощью `<article>` , главы с помощью `<section>` , навигационных средств страницы с использованием `<nav>` и группы элементов управления формой с помощью `<fieldset>` .

Элементы div могут быть полезны для стилистических целей или для обертывания нескольких абзацев внутри раздела, которые все должны быть аннотированы аналогичным образом.

Прочитайте Элемент Div онлайн: https://riptutorial.com/ru/html/topic/1468/элемент-div

глава 37: Элемент метки

Синтаксис

- `<label>Example <input type="radio" name="r"></label>` // Обтекание элемента управления
- `<label for="rad1">Example</label> <input id="rad1" type="radio" name="r">` // Использование `for` атрибута

параметры

Атрибуты	Описание
за	Ссылка на целевой идентификационный элемент. Т.е.: `for="surname"`
форма	**HTML5** , **[Устаревший]** Ссылка на форму, содержащую целевой элемент. Элементы метки ожидаются в элементе `<form>` . Если предоставляется `form="someFormId"` это позволяет помещать ярлык в любом месте документа.

Examples

Основное использование

Простая форма с этикетками ...

```
<form action="/login" method="POST">

    <label for="username">Username:</label>
    <input id="username" type="text" name="username" />

    <label for="pass">Password:</label>
    <input id="pass" type="password" name="pass" />

    <input type="submit" name="submit" />

</form>
```

5

```
<form id="my-form" action="/login" method="POST">

    <input id="username" type="text" name="username" />

    <label for="pass">Password:</label>
    <input id="pass" type="password" name="pass" />

    <input type="submit" name="submit" />
```

```
</form>

<label for="username" form="my-form">Username:</label>
```

О ярлыке

Элемент `<label>` используется для ссылки на элемент действия формы.
В области **пользовательского интерфейса** он используется для облегчения цели / выбора элементов, таких как Type `radio` или `checkbox` .

`<label>` качестве обертки

Он может заключать желаемый элемент действия

```
<label>
    <input type="checkbox" name="Cats">
    I like Cats!
</label>
```

(Нажав на текст, целевой `input` переключит его состояние / значение)

`<label>` как ссылка

Используя атрибут `for` вам не нужно помещать элемент управления в качестве потомка `label` но значение `for` должно соответствовать его идентификатору

```
<input id="cats" type="checkbox" name="Cats">
<label for="cats" >I like Cats!</label>
```

Заметка
Не используйте более одного элемента управления в элементе `<label>`

Прочитайте Элемент метки онлайн: https://riptutorial.com/ru/html/topic/1704/элемент-метки

глава 38: Элемент прогресса

параметры

параметр	Значение
Максимум	Сколько работы требуется в общей сложности
значение	Сколько работы уже выполнено
позиция	Этот атрибут возвращает текущую позицию элемента `<progress>`
этикетки	Этот атрибут возвращает список ярлыков элементов `<progress>` (если они есть)

замечания

Элемент `<progress>` не поддерживается в версиях Internet Explorer менее 10

Элемент `<progress>` - это неправильный элемент, который должен использоваться для чего-то, что является просто калибром, а не для выполнения задачи. Например, показ использования дискового пространства с помощью элемента `<progress>` не подходит. Вместо этого элемент `<meter>` доступен для этого типа прецедентов.

Examples

Прогресс

Элемент `<progress>` является новым в HTML5 и используется для представления хода выполнения задачи

```
<progress value="22" max="100"></progress>
```

Это создает бар, заполненный 22%

Изменение цвета индикатора выполнения

Полосы хода могут быть написаны с помощью переключателя `progress[value]` .

В этом примере индикатор выполнения имеет ширину `250px` и высоту 20 `20px`

```
progress[value] {
```

```
  width: 250px;
  height: 20px;
}
```

Полосы прогресса могут быть особенно сложными для стиля.

Chrome / Safari / Opera

Эти браузеры используют селектор `-webkit-appearance` чтобы `-webkit-appearance` метку прогресса. Чтобы переопределить это, мы можем сбросить внешний вид.

```
progress[value] {
  -webkit-appearance: none;
  appearance: none;
}
```

Теперь мы можем создать стиль контейнера

```
progress[value]::-webkit-progress-bar {
  background-color: "green";
}
```

Fire Fox

Firefox стилирует индикатор выполнения немного по-другому. Мы должны использовать эти стили

```
progress[value] {
  -moz-appearance: none;
  appearance: none;
  border: none;                /* Firefox also renders a border */
}
```

Internet Explorer

Internet Explorer 10+ поддерживает элемент `progress` . Однако он не поддерживает свойство `background-color` . Вместо этого вам нужно использовать свойство цвета.

```
progress[value]  {
  -webkit-appearance: none;
     -moz-appearance: none;
          appearance: none;

  border: none;                       /* Remove border from Firefox */

  width: 250px;
  height: 20px;

  color: blue;
}
```

Отказ HTML

Для браузеров, которые не поддерживают элемент `progress` , вы можете использовать это как обходной путь.

```
<progress max="100" value="20">
    <div class="progress-bar">
        <span style="width: 20%;">Progress: 20%</span>
    </div>
</progress>
```

Браузеры, поддерживающие тег progress, будут игнорировать div, вложенные внутри. Устаревшие браузеры, которые не могут идентифицировать метку прогресса, будут отображать div.

Прочитайте Элемент прогресса онлайн: https://riptutorial.com/ru/html/topic/5055/элемент-прогресса

глава 39: Элементы меню выбора

Синтаксис

- `<select name=""></select>`
- `<datalist id=""></datalist>`
- `<optgroup label="Option Group"></optgroup>`
- `<option value="">Option</option>`

Examples

Выберите Меню

Элемент `<select>` создает раскрывающееся меню, из которого пользователь может выбрать параметр.

```
<select name="">
  <option value="1">One</option>
  <option value="2">Two</option>
  <option value="3">Three</option>
  <option value="4">Four</option>
</select>
```

Изменение размера

Вы можете изменить размер меню выбора с помощью атрибута `size` . Размер 0 или 1 отображает стандартное меню стиля раскрывающегося списка. Размер, превышающий 1, преобразует раскрывающийся список в поле, отображающее много строк, с одним параметром на строку и полосой прокрутки для прокрутки доступных параметров.

```
<select name="" size="4"></select>
```

Многофункциональные меню выбора

По умолчанию пользователи могут выбирать только одну опцию. Добавление атрибута `multiple` позволяет пользователям одновременно выбирать несколько параметров и отправлять все выбранные опции в форме. Использование `multiple` атрибута автоматически преобразует раскрывающееся меню в поле, как если бы он имел определенный размер. Размер по умолчанию, когда это происходит, определяется конкретным браузером, который вы используете, и его невозможно вернуть в раскрывающееся меню стилей, одновременно предоставляя несколько вариантов выбора.

```
<select name="" multiple></select>
```

При использовании `multiple` атрибута существует разница между использованием 0 и 1 для размера, тогда как разница не существует, если не используется атрибут. Использование 0 приведет к тому, что браузер будет вести себя так, как это было запрограммировано по умолчанию. Использование 1 будет явно задавать размер получаемого окна только на одну строку.

Группы вариантов

Вы можете аккуратно группировать свои параметры в меню выбора, чтобы обеспечить более структурированный макет в длинном списке параметров с помощью элемента `<optgroup>` .

Синтаксис очень простой, просто используя элемент с атрибутом `label` для идентификации заголовка для группы и содержащий ноль или более параметров, которые должны быть внутри этой группы.

```
<select name="">
  <option value="milk">Milk</option>
  <optgroup label="Fruits">
    <option value="banana">Bananas</option>
    <option value="strawberry">Strawberries</option>
  </optgroup>
  <optgroup label="Vegetables" disabled>
    <option value="carrot">Carrots</option>
    <option value="zucchini">Zucchini</option>
  </optgroup>
</select>
```

При использовании групп опций не все параметры должны содержаться внутри группы. Кроме того, при отключении группы параметров будут отключены все параметры внутри группы, и невозможно вручную включить один параметр в отключенной группе.

Опции

Параметры внутри меню выбора - это то, что пользователь будет выбирать. Стандартный синтаксис для опции выглядит следующим образом:

```
<option>Some Option</option>
```

Однако важно отметить, что текст внутри самого элемента `<option>` не всегда используется и по существу становится значением по умолчанию для атрибутов, которые не указаны.

Атрибутами, которые управляют фактическим внешним видом и функцией опции, являются `value` и `label` . Метка представляет текст, который будет отображаться в раскрывающемся меню (на что вы смотрите и нажмете, чтобы выбрать его). Значение представляет текст, который будет отправлен вместе с отправкой формы. Если одно из этих значений опущено, оно вместо этого использует текст внутри элемента как значение. Таким образом,

приведенный выше пример может быть «расширен»:

```
<option label="Some Option" value="Some Option">
```

Обратите внимание на отсутствие внутреннего текста и конечного тега, которые не требуются для фактического создания опции внутри меню. Если они были включены, внутренний текст будет проигнорирован, потому что оба атрибута уже указаны и текст не нужен. Тем не менее, вы, вероятно, не увидите много людей, которые пишут их таким образом. Наиболее распространенным способом написания является значение, которое будет отправлено на сервер вместе с внутренним текстом, который в конечном итоге станет атрибутом метки, например:

```
<option value="option1">Some Option</option>
```

Выбор опции по умолчанию

Вы также можете указать определенную опцию, которая будет выбрана в меню по умолчанию, добавив к ней `selected` атрибут. По умолчанию, если в меню не указан параметр, выбранный в меню, первый параметр в меню будет выбран при визуализации. Если несколько атрибутов имеют `selected` атрибут, то последний параметр, присутствующий в меню с атрибутом, будет выбран по умолчанию.

```
<option value="option1" selected>Some option</option>
```

Если вы используете атрибут в меню выбора нескольких опций, то по умолчанию будут выбраны все параметры с атрибутом, и ни один из них не будет выбран, если параметры не имеют атрибута.

```
<select multiple>
  <option value="option1" selected>Some option</option>
  <option value="option2" selected>Some option</option>
</select>
```

DataList

Тег `<datalist>` указывает список предварительно определенных параметров для элемента `<input>` . Он предоставляет функцию автозаполнения для элементов `<input>` . Пользователи будут видеть раскрывающийся список параметров по мере их записи.

```
<input list="Languages">

<datalist id="Languages">
  <option value="PHP">
  <option value="Perl">
  <option value="Python">
  <option value="Ruby">
```

```
  <option value="C+">
</datalist>
```

Поддержка браузера

Хром	край	Mozilla	Сафари	опера
20,0	10,0	4,0	Не поддерживается	9,0

Прочитайте Элементы меню выбора онлайн: https://riptutorial.com/ru/html/topic/722/элементы-меню-выбора

глава 40: Элементы управления вводом

Вступление

Ключевым компонентом интерактивных веб-систем, входных тегов являются HTML-элементы, предназначенные для получения конкретной формы ввода от пользователей. Различные типы элементов ввода могут регулировать введенные данные в соответствии с указанным форматом и обеспечивать безопасность ввода пароля.

Синтаксис

- `<input type="" name="" value="">`

параметры

параметр	подробности
учебный класс	Указывает класс ввода
Я бы	Указывает идентификатор входа
тип	Определяет тип управляющего элемента для отображения. Допустимые значения `hidden` , `text` , `tel` , `url` , `email` , `password` , `date` , `time` , `number` , `range` , `color` , `checkbox` , `radio` , `file` , `submit` , `image` , `reset` и `button` . По умолчанию `text` если не указан, если значение недействительно или если браузер не поддерживает указанный тип.
название	Указывает имя входа
отключен	Логическое значение, указывающее на ввод, должно быть отключено. Отключенные элементы управления не могут редактироваться, не отправляться при отправке формы и не могут получать фокус.
проверено	Когда значением атрибута type является радио или флажок, наличие этого Boolean-атрибута указывает, что элемент управления выбран по умолчанию; в противном случае он игнорируется.
множественный	**HTML5** Указывает, что можно передавать несколько файлов или значений (применяется только к входам типа `file` и `email`)

параметр	подробности
заполнитель	**HTML5** Подсказка пользователю о том, что можно ввести в элемент управления. Текст заполнитель не должен содержать возврат каретки или линейные каналы
автозаполнения	**HTML5** Указывает, будет ли значение элемента управления автоматически завершено браузером.
только для чтения	Логическое значение, указывающее на ввод, не редактируется. Элементы Readonly по-прежнему отправляются при отправке формы, но не получают фокуса. **HTML5:** этот атрибут игнорируется, когда значение атрибута `type` либо установлено на `hidden` , `range` , `color` , `checkbox` , `radio` , `file` или `button` .
требуется	**HTML5** Указывает, что значение должно присутствовать или элемент должен быть проверен, чтобы форма была отправлена
альт	Альтернативный текст для изображений, если они не отображаются.
автофокусировка	Элемент `<input>` должен получать фокус при загрузке страницы.
значение	Задает значение элемента `<input>` .
шаг	Атрибут `step` задает интервалы юридического номера. Он работает со следующими типами ввода: `number` , `range` , `date` , `date datetime-local` , `month` , `time` и `week` .

замечания

Как и в других элементах HTML5 void, `<input>` является самозакрывающимся и может быть записано `<input />` . HTML5 не требует этой косой черты.

Ниже приведены допустимые типы ввода в HTML:

- `button`
- `checkbox`
- `file`
- `hidden`
- `image`
- `password`
- `radio`
- `reset`
- `submit`
- `text` (значение по умолчанию)

5

Ниже перечислены новые типы ввода как часть стандарта HTML 5. Некоторые из этих типов не поддерживаются всеми веб-браузерами. В случае, когда тип не поддерживается, элемент ввода по умолчанию будет иметь тип `text` .

- `color`
- `date`
- `datetime` (Устаревшие и устаревшие)
- `datetime-local`
- `email`
- `month`
- `number`
- `range`
- `search`
- `tel`
- `time`
- `url`
- `week`

Чтобы проверить, какие браузеры поддерживают какие типы, вы можете перейти на сайт caniuse.com .

Examples

Флажок и кнопки радио

обзор

Флажки и переключатели написаны с тегом HTML `<input>` , и их поведение определено в спецификации HTML .

Простейшие кнопки флажка или радио является `<input>` элемента с `type` атрибутом `checkbox` или `radio` , соответственно:

```
<input type="checkbox">
<input type="radio">
```

Один отдельный элемент флажка используется для одной бинарной опции, такой как вопрос «да» или «нет». Флажки являются независимыми, что означает, что пользователь может выбрать столько вариантов, сколько захочет в группе флажков. Другими словами, проверка один флажок *не* снимите другие флажки в CheckBox группе.

Радио-кнопки обычно входят в группы (если они не сгруппированы с другим переключателем, скорее всего, вы должны использовать флажок), идентифицированные с помощью одного и того же `name` на всех кнопках этой группы. Выбор переключателей является *взаимоисключающим* , что означает, что пользователь может выбрать только один выбор из группы переключателей. Когда переключатель установлен, любая другая

радиокнопка с тем же `name` которая была ранее отмечена, не снята.

Пример:

```
<input type="radio" name="color" id="red" value="#F00">
<input type="radio" name="color" id="green" value="#0F0">
<input type="radio" name="color" id="blue" value="#00F">
```

При просмотре радиокнопки отображаются в виде круга (непроверенный) или заполненного круга (отмечен). Флажки отображаются как квадратные (непроверенные) или заполненные квадраты (отмеченные). В зависимости от браузера и операционной системы квадрат иногда имеет закругленные углы.

Атрибуты

флажки и переключатели имеют ряд атрибутов для управления их поведением:

`value`

Как и любой другой элемент ввода, атрибут `value` указывает значение строки, которое нужно связать с кнопкой в случае отправки формы. Тем не менее, флажки и переключатели являются особенными в том , что , когда значение опущено, то по умолчанию `on` при представлении, а не отправив пустое значение. Атрибут `value` не отражается на внешнем виде кнопки.

`checked`

Атрибут `checked` указывает начальное состояние флажка или переключателя. Это логический атрибут и может быть опущен.

Каждый из них является допустимым эквивалентным способом определения проверенного переключателя:

```
<input checked>
<input checked="">
<input checked="checked">
<input checked="ChEcKeD">
```

Отсутствие `checked` атрибута является единственным допустимым синтаксисом для непроверенной кнопки:

```
<input type="radio">
<input type="checkbox">
```

При сбросе `<form>` , флажки и переключатели возвращаются к состоянию их `checked` атрибута.

доступность

Этикетки

Чтобы дать контекст кнопкам и показать пользователям, для каждой кнопки, каждая из них должна иметь метку. Это можно сделать, используя элемент `<label>` чтобы обернуть кнопку. Кроме того, это делает ярлык интерактивным, поэтому вы выбираете соответствующую кнопку.

Пример:

```
<label>
  <input type="radio" name="color" value="#F00">
  Red
</label>
```

или с элементом `<label>` с атрибутом `for` заданным для атрибута `id` кнопки:

```
<input type="checkbox" name="color" value="#F00" id="red">
<label for="red">Red</label>
```

Группы кнопок

Поскольку каждый радиокнопка влияет на остальных в группе, обычно предоставляется ярлык или контекст для всей группы переключателей.

Чтобы предоставить ярлык для всей группы, радиокнопки должны быть включены в элемент `<fieldset>` элементом `<legend>` внутри него.

Пример:

```
<fieldset>
  <legend>Theme color:</legend>
  <p>
    <input type="radio" name="color" id="red" value="#F00">
    <label for="red">Red</label>
  </p>
  <p>
    <input type="radio" name="color" id="green" value="#0F0">
    <label for="green">Green</label>
  </p>
  <p>
    <input type="radio" name="color" id="blue" value="#00F">
    <label for="blue">Blue</label>
  </p>
</fieldset>
```

Флажки также могут быть сгруппированы аналогично, с набором полей и легендой,

идентифицирующей группу связанных флажков. Однако имейте в виду, что флажки *не* должны иметь одно и то же имя, поскольку они не являются взаимоисключающими. Это приведет к тому, что форма отправит несколько значений для одного и того же ключа, и не все серверные языки обрабатывают это одинаково (неопределенное поведение). Каждый флажок должен иметь либо уникальное имя, либо использовать набор квадратных скобок (`[]`), чтобы указать, что форма должна представить массив значений для этого ключа. Какой метод вы выбираете, зависит от того, как вы планируете обрабатывать данные формы на стороне клиента или на стороне сервера. Вы также должны держать легенду коротким, поскольку некоторые комбинации браузеров и считывателей экрана читают легенду перед каждым полем ввода в полевом наборе.

скрытый

```
<input type="hidden" name="inputName" value="inputValue">
```

Скрытый ввод не будет отображаться для пользователя, но его значение будет отправлено на сервер, когда форма будет отправлена, тем не менее.

пароль

```
<input type="password" name="password">
```

Элемент ввода с атрибутом type, значение которого является `password` создает однострочное текстовое поле, подобное `type=text` ввода `type=text` , за исключением того, что текст не отображается, когда пользователь вводит его.

```
<input type="password" name="password" placeholder="Password">
```

Текст-заполнитель отображается в виде обычного текста и автоматически перезаписывается, когда пользователь начинает печатать.

Примечание. Некоторые браузеры и системы изменяют поведение по умолчанию в поле пароля, чтобы также отображать последний типизированный символ в течение короткой продолжительности, например:

Отправить

```
<input type="submit" value="Submit">
```

Ввод ввода создает кнопку, которая отправляет форму, находящуюся внутри, при нажатии.

Вы также можете использовать элемент `<button>` если вам нужна кнопка отправки, которую можно более легко стилизовать или содержать другие элементы:

```
<button type="submit">
  <img src="submit-icon.jpg" /> Submit
</button>
```

файл

```
<input type="file" name="fileSubmission">
```

Входы файлов позволяют пользователям выбирать файл из своей локальной файловой системы для использования с текущей страницей. Если они используются вместе с элементом `form` , они могут использоваться, чтобы позволить пользователям загружать файлы на сервер (дополнительную информацию см. В разделе « Загрузка файлов»).

В следующем примере пользователи могут использовать входной `file` для выбора файла из своей файловой системы и загрузить этот файл в сценарий на сервере с именем `upload_file.php` .

```
<form action="upload_file.php" method="post" enctype="multipart/form-data">
    Select file to upload:
    <input type="file" name="fileSubmission" id="fileSubmission">
    <input type="submit" value="Upload your file" name="submit">
</form>
```

Несколько файлов

Добавив `multiple` атрибутов, пользователь сможет выбрать **несколько** файлов:

```
<input type="file" name="fileSubmission" id="fileSubmission" multiple>
```

Принять файлы

Атрибут Accept определяет типы файлов, которые пользователь может выбрать. Например `.png` , `.gif` , `.jpeg` .

```
<input type="file" name="fileSubmission" accept="image/x-png,image/gif,image/jpeg" />
```

Проверка ввода

Проверка ввода HTML выполняется автоматически браузером на основе специальных атрибутов элемента ввода. Это может частично или полностью заменить проверку ввода

JavaScript. Этот вид проверки может быть обойден пользователем через специально созданные HTTP-запросы, поэтому он не заменяет проверку ввода на стороне сервера. Проверка выполняется только при попытке отправить форму, поэтому все ограниченные входные данные должны быть внутри формы для проверки (если вы не используете JavaScript). Имейте в виду, что входы, которые отключены или доступны только для чтения, не будут инициировать проверку.

Некоторые новые типы ввода (например, `email` , `url` , `tel` , `date` и многие другие) автоматически проверяются и не требуют собственных ограничений проверки.

5

необходимые

Используйте `required` атрибут, чтобы указать, что поле должно быть заполнено, чтобы пройти проверку.

```
<input required>
```

Минимальная / максимальная длина

Используйте `minlength` и `maxlength` чтобы указать требования к длине. Большинство браузеров не позволят пользователю вводить в поле больше, чем *максимальные* символы, что предотвращает их недействительность даже до того, как они попытаются отправить.

```
<input minlength="3">
<input maxlength="15">
<input minlength="3" maxlength="15">
```

Указание диапазона

Используйте атрибуты `min` и `max` чтобы ограничить диапазон чисел, которые пользователь может ввести во вход типа `number` или `range`

```
Marks: <input type="number" size="6" name="marks" min="0" max="100" />
Subject Feedback: <input type="range" size="2" name="feedback" min="1" max="5" />
```

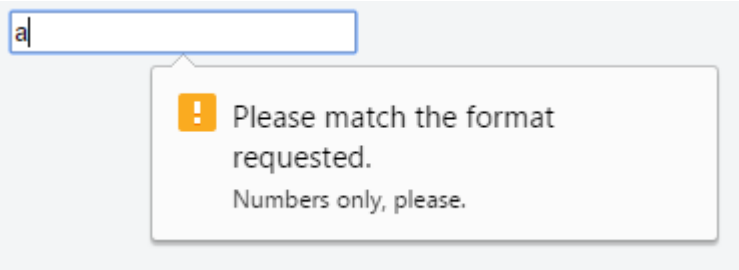
5

Соответствие шаблону

Для большего контроля используйте атрибут `pattern` чтобы указать любое регулярное выражение, которое должно быть сопоставлено, чтобы пройти проверку. Вы также можете указать `title` , который включен в сообщение проверки, если поле не проходит.

```
<input pattern="\d*" title="Numbers only, please.">
```

Вот сообщение, показанное в Google Chrome версии 51 при попытке отправить форму с недопустимым значением внутри этого поля:



Не все браузеры отображают сообщение о недопустимых шаблонах, хотя в большинстве используемых современных браузеров есть полная поддержка.

Проверьте последнюю поддержку CanIUse и внесите соответствующие изменения.

5

Принять тип файла

Для полей ввода типа `file` можно принимать только определенные типы файлов, такие как видео, изображения, аудио, определенные расширения файлов или определенные типы носителей . Например:

```
<input type="file" accept="image/*" title="Only images are allowed">
```

Несколько значений могут быть заданы запятой, например:

```
<input type="file" accept="image/*,.rar,application/zip">
```

Примечание: добавление атрибута `novalidate` к элементу `form` или `formnovalidate` к кнопке отправки, предотвращает проверку элементов формы. Например:

```
<form>
    <input type="text" name="name" required>
    <input type="email" name="email" required>
    <input pattern="\d*" name="number" required>

    <input type="submit" value="Publish"> <!-- form will be validated -->
    <input type="submit" value="Save" formnovalidate> <!-- form will NOT be validated -->
</form>
```

Форма имеет поля, которые необходимы для «публикации» проекта, но не требуются для «сохранения» проекта.

Сброс

```
<input type="reset" value="Reset">
```

Ввод `reset` типа создает кнопку, которая при щелчке сбрасывает все входы в форме, в которой она содержится, до состояния по умолчанию.

- Текст в поле ввода будет сброшен до пустого или его значения по умолчанию (указанного с использованием атрибута `value`).
- Любая опция (ы) в меню выбора будет отменена, если у них нет `selected` атрибута.
- Все флажки и радиокнопки будут отменены, если у них нет атрибута `checked` .

Примечание. Кнопка сброса должна быть внутри или прикреплена (через атрибут `form`) к элементу `<form>` , чтобы иметь какой-либо эффект. Кнопка только сбросит элементы в этой форме.

Число

5

```
<input type="number" value="0" name="quantity">
```

Элемент Input с атрибутом type, значение которого является `number` представляет собой точный элемент управления для установки значения элемента в строку, представляющую число.

Обратите внимание, что это поле не гарантирует наличие правильного номера. Он просто позволяет использовать все символы, которые могут использоваться в любом действительном числе, например, пользователь сможет ввести значение, подобное `e1e-,0` .

телефон

```
<input type="tel" value="+8400000000">
```

Элемент ввода с атрибутом type, значение которого является `tel` представляет собой однострочный текстовый редактор для ввода номера телефона.

Эл. адрес

5

`<input type="email">` используется для полей ввода, которые должны содержать адрес электронной почты.

```
<form>
  <label>E-mail: <label>
  <input type="email" name="email">
</form>
```

Адрес электронной почты может быть автоматически подтвержден при отправке в

зависимости от поддержки браузера.

кнопка

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

Кнопки могут использоваться для инициирования действий на странице, не отправляя форму. Вы также можете использовать элемент `<button>` если вам нужна кнопка, которую можно более легко стилизовать или содержать другие элементы:

```
<button type="button">Button Text</button>
```

Кнопки обычно используются с событием «onclick»:

```
<input type="button" onclick="alert('hello world!')" value="Click Me">
```

или же

```
<button type="button" onclick="alert('hello world!')">Click Me</button>
```

Атрибуты

`[name]`

`name` кнопки, которая отправляется с данными формы.

`[type]`

`type` кнопки.

Возможные значения:

`submit` : кнопка передает данные формы на сервер. Это значение по умолчанию, если атрибут не указан, или если атрибут динамически изменен на пустое или недопустимое значение.

`reset` : кнопка сбрасывает все элементы управления до их начальных значений.

`button` : кнопка не имеет поведения по умолчанию. Он может иметь клиентские сценарии, связанные с событиями элемента, которые запускаются при возникновении событий.

`menu` : кнопка открывает всплывающее меню, определенное через назначенный элемент.

`[value]`

Начальное значение кнопки.

5

Дополнительные атрибуты для кнопок отправки

атрибут	Описание
`form`	Задает идентификатор формы, к которой принадлежит кнопка. Если ни один не указан, он будет принадлежать элементу формы предка (если он существует).
`formaction`	Указывает, куда отправлять данные формы когда форма отправляется с помощью этой кнопки.
`formenctype`	Указывает, как должны быть закодированы данные формы при отправке его на сервер с помощью этой кнопки. Может использоваться только с `formmethod="post"` .
`formmethod`	Указывает используемый HTTP-метод (POST или GET) при отправке форм-данных с помощью этой кнопки.
`formnovalidate`	Указывает, что данные формы не должны проверяться при подаче.
`formtarget`	Определяет, где отображать полученный ответ после отправки формы с помощью этой кнопки.

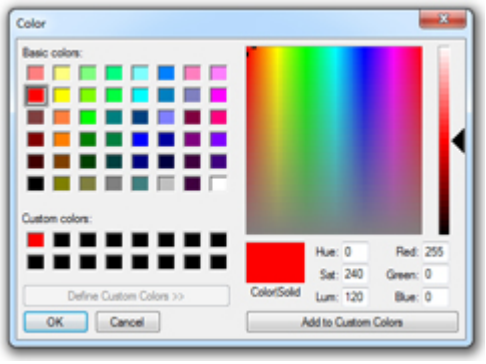
цвет

5

```
<input type="color" name="favcolor" value="#ff0000">
```

В поддерживающих браузерах элемент ввода с атрибутом type, значение которого является `color` создает кнопочный элемент управления с цветом, равным значению атрибута `color` (по умолчанию черный, если значение не указано или является недопустимым шестнадцатеричным форматом).

При нажатии этой кнопки открывается цветной виджет операционной системы, который позволяет пользователю выбирать цвет.



Ошибка для браузеров, которые не поддерживают этот тип ввода, - это обычный `type=text` ввода `type=text` .

#ff0000

Веб-сайт

5

```
<input type="url" name="Homepage">
```

Это используется для полей ввода, которые должны содержать URL-адрес.

В зависимости от поддержки браузера поле `url` может быть автоматически проверено при отправке.

Некоторые смартфоны распознают тип `url` и добавляют «.com» к клавиатуре, чтобы соответствовать вводу url.

Дата

5

```
<input type="date" />
```

На экране появится подсказка для выбора даты. Это не поддерживается в Firefox или Internet Explorer.

DateTime-Local

5

```
<input type="datetime-local" />
```

В зависимости от поддержки браузера на экране появится дата и время, чтобы выбрать дату и время.

Образ

```
<input type="image" src="img.png" alt="image_name" height="50px" width="50px"/>
```

Картинка. Вы должны использовать атрибут src для определения источника изображения и атрибута alt для определения альтернативного текста. Вы можете использовать атрибуты height и width для определения размера изображения в пикселях.

Спектр

5

```
<input type="range" min="" max="" step="" />
```

Элемент управления для ввода числа, точное значение которого не имеет значения.

атрибут	Описание	Значение по умолчанию
мин	Минимальное значение для диапазона	0
Максимум	Максимальное значение для диапазона	100
шаг	Сумма увеличивается с каждым приращением.	1

Месяц

5

```
<input type="month" />
```

В зависимости от поддержки браузера элемент управления отобразит месяц.

Время

5

```
<input type="time" />
```

Ввод `time` отмечает этот элемент как принятие строки, представляющей время. Формат определен в RFC 3339 и должен быть частичным, таким как

```
19:04:39
08:20:39.04
```

В настоящее время все версии Edge, Chrome, Opera и Chrome для Android поддерживают

тип = «время». Более новые версии Android Browser, в частности, 4,4 и выше поддерживают его. Safari для iOS предлагает частичную поддержку, не поддерживая атрибуты min, max и step.

Неделю

5

```
<input type="week" />
```

В зависимости от поддержки браузера будет отображаться элемент управления для ввода номера недельного года и номера недели без часового пояса.

Текст

Самый основной тип ввода и ввод по умолчанию, если не указан `type` . Этот тип ввода определяет однострочное текстовое поле с автоматическим удалением строк из входного значения. Все остальные символы могут быть введены в это. Элементы `<input>` используются в элементе `<form>` для объявления элементов управления вводами, которые позволяют пользователям вводить данные.

Синтаксис

```
<input type="text">
```

или (без указания `type` , используя атрибут по умолчанию):

```
<input>
```

По умолчанию ширина ввода текстового поля составляет 20 символов. Это можно изменить, указав значение атрибута `size` следующим образом:

```
<input type="text" size="50">
```

Атрибут `size` явно отличается от установки ширины с помощью CSS. Использование ширины определяет конкретное значение (в количестве пикселей, процентах от родительского элемента и т. Д.), Что вход всегда должен быть широким. Использование `size` вычисляет величину ширины для распределения на основе используемого шрифта и как широки символы обычно.

Примечание. Использование атрибута `size` не ограничивает количество символов, которые могут быть введены в поле, а только то, насколько широко отображается окно. Для ограничения длины см. « Проверка ввода» .

Поле ввода допускает только одну строку текста. Если вам нужен многострочный

текстовый ввод для значительного объема текста, используйте вместо него элемент `<textarea>` .

Поиск

5

Поиск типа ввода используется для текстового поиска. Он добавит символ лупы рядом с пространством для текста в большинстве браузеров

```
<input type="search" name="googlesearch">
```

DateTime (Глобальный)

Элемент ввода с атрибутом type, значением которого является « **datetime** », представляет собой элемент управления для установки значения элемента для строки, представляющей **глобальную дату и время (с информацией о часовом поясе).**

```
<fieldset>
   <p><label>Meeting time: <input type=datetime name="meeting.start"></label>
</fieldset>
```

Разрешенные атрибуты:

- глобальные атрибуты
- название
- отключен
- форма
- тип
- автозаполнения
- автофокусировка
- список
- мин Макс
- step (float)
- только для чтения
- требуемое значение

Прочитайте Элементы управления вводом онлайн: https://riptutorial.com/ru/html/topic/277/элементы-управления-вводом

глава 41: Языки контента

Синтаксис

- `<element lang="language_code">` <! - Код языка должен быть в формате [ISO 639-1] (https://en.wikipedia.org/wiki/ISO_639-1) ->

замечания

Значение атрибута `lang` должно быть допустимым **тегом языка BCP 47** или **пустой строкой** (если язык неизвестен).

Теги языка BCP 47 перечислены в реестре Subtag Language IANA .

доступность

Соответствующие критерии успеха WCAG 2.0:

- 3.1.1 Язык страницы
- 3.1.2 Язык деталей

Связанные с WCAG 2.0 методы:

- H57: использование атрибутов языка в элементе html
- H58: использование атрибутов языка для идентификации изменений в человеческом языке

Examples

Язык элемента

Атрибут `lang` используется для указания языка содержимого элемента и значений текста атрибута:

```
<p lang="en">The content of this element is in English.</p>
```

```
<p lang="en" title="The value of this attribute is also in English.">The content of this
element is in English.</p>
```

Декларация языка наследуется:

```
<div lang="en">
  <p>This element contains English content.</p>
  <p title="This attribute, too.">Same with this element.</p>
```

```
</div>
```

Элементы с несколькими языками

Вы можете «перезаписать» декларацию языка:

```
<p lang="en">This English sentence contains the German word <span lang="de">Hallo</span>.</p>
```

Обработка атрибутов с разными языками

Вы можете «перезаписать» декларацию языка родительского элемента, введя любой элемент, помимо `basefont applet` , `base` , `basefont` , `br` , `frame` , `frameset` , `hr` , `iframe` , `meta` , `param` , `script` (HTML 4.0) с собственным атрибутом `lang` :

```
<p lang="en" title="An English paragraph">
    <span lang="de" title="A German sentence">Hallo Welt!</span>
</p>
```

Язык базового документа

Хорошей практикой является объявление основного языка документа в элементе `html` :

```
<html lang="en">
```

Если в документе не указан другой атрибут `lang` , это означает, что *все* (то есть содержимое элемента и значения текста атрибута) находится на этом языке.

Если документ содержит части на других языках, эти части должны получить свои собственные атрибуты `lang` для «перезаписывания» декларации языка.

Региональные URL-адреса

Можно добавить атрибут `hreflang` к элементам `<a>` и `<area>` которые создают гиперссылки. В нем указывается язык связанного ресурса. Определенный язык должен быть допустимым тегом языка BCP 47 [1] .

```
<p>
    <a href="example.org" hreflang="en">example.org</a> is one of IANA's example domains.
</p>
```

---

1. ↑ Рабочая группа IETF Network: RFC 5646 *Tags для идентификации языков* , IETF, сентябрь 2009 г.

Прочитайте Языки контента онлайн: https://riptutorial.com/ru/html/topic/737/языки-контента

кредиты

S. No	Главы	Contributors
1	Начало работы с HTML	4444, Abhishek Pandey, aea2002, ahmednawazbutt, Alexander Wigmore, Alexandre N., Amanda Ahn, amflare, Amitay Stern, animuson, Anthony Pham, Boris, bwegs, Callan Heard, ChrisD, CocoaBean, Community, Dave Everitt, Dinidu, dippas, dtyler, duskwuff, Eric Dobbs, Firix, FlyingPiMonster, geek1011, George Bailey, Gerold Broser, H Mirza, H. Pauwelyn, Harish Gyanani, Hemant Kumar, hillary.fraley, Hudson Taylor, ihavemorealts, intboolstring, Isak Combrinck, Jeffrey Lin, JHS, jmmarco, joe_young, John Slegers, Jon Chan, JonasCz, JPB, kelvinelove, Krii, Kurniawantaari, Lahiru Ashan, Lambda Ninja, Léo Martin, Leonidas Menendez, Malcolm, Matt, Matt, MC93, Michael Moriarty, mnoronha, Muntasir, Nishchay, Ortomala Lokni, Persijn, Prateek, Pyloid, Ryan Hilbert, Shannon Young, sideshowbarker, stark, Stelian Matei, Sunny R Gupta, the12, tmg, unor, user3130333, Valor Naram, Willl, Wolfgang, Zaz, zygimantus, Zze
2	ARIA	Bhavya Singh, Paul Sweatte, Shannon Young, Travis, unor, user30796
3	DOCTYPEs	Akshay Anand, Al.G., Angelos Chalaris, Ani Menon, animuson, Chris, Content Solutions, heerfk, mnoronha, pinjasaur, Right leg , Sumner Evans, Thomas Gerot, tmg, unor
4	HTML 5 кэш	Farhad, TricksfortheWeb, Valor Naram
5	IFrames	Adjit, Alexandre N., animuson, ChrisD, dorukayhan, Duh-Wayne-101, Emanuel Vintilă, J F, Ojen, Wojciech Kazior
6	SVG	andreas, Black Mamba, ChrisD, HerrSerker, Patrickdev, Timothy Miller, w5m
7	TabIndex	Content Solutions, Psaniko
8	Анкеры и гиперссылки	Al.G., animuson, Anselm Urban, Anthony Pham, ban17, DawnPaladin, Emil, FlyingPiMonster, insertusernamehere, J F, JHS, joe_young, Jojodmo, Jones Joseph, Lambda Ninja, Matas Vaitkevicius, Nathan Tuggy, Pranav, Prateek, Raystafarian, Robert Columbia, Squidward, Steyn van Esveld, Thomas Gerot, unor, Wolfgang

9	Атрибуты данных	animuson, Community, Faegy, Infuzed guy, James Donnelly, Manish, Nathan Tuggy, Nhan, Racil Hilan, rajarshig, swatchai, unor, Yasir T
10	Атрибуты событий HTML	Paresh Maghodiya
11	Включить код JavaScript в HTML	Alexandre N., andreaem, animuson, Anselm Urban, Charles, Marjorie Pickard, MervS, Roko C. Buljan, Sildoreth, SuperStormer
12	встраивать	Alexandre N.
13	Выходной элемент	J F, Stephen Leppik, zer00ne
14	Глобальные атрибуты	animuson, Becca, unor, Zange-chan
15	Заголовки	Ani Menon, animuson, dippas, Evan, Infuzed guy, joe_young, MervS, Nathan Arthur, Pseudonym Patel, sasha, Thomas Gerot , unor, V-Kopio
16	Изображений	Alex, Alexandre N., andreaem, animuson, Boysenb3rry, Caleb Kleveter, Gabriel Chi Hong Lee, Infuzed guy, ivn, Kake_Fisk, Maximillian Laumeister, Mohd Samir Khan, Mr Lister, Nhan, Shivangi Chaurasia, Stephen Leppik, Wolfgang
17	Использование HTML с CSS	animuson, bdkopen, Christian Ternus, Community, Euan Williams, feeela, Jones Joseph, Michael Moriarty, Thomas Gerot, thouusten
18	Карты изображений	animuson, Lambda Ninja, RamenChef
19	Классы и идентификаторы	Angelos Chalaris, animuson, brandaemon, Caleb Kleveter, Community, Duh-Wayne-101, Emil, Epodax, Evan, GoatsWearHats, Ingrid Stevens, jhnance, JHS, John Slegers, lexith, Luca Putzu, Michael_B, Natalie, Nhan, Richard Hamilton, Simone Carletti, Thomas Gerot, Timothy, Tyler Zika, unor, Wolfgang, xims
20	Комментарии	Ani Menon, animuson, Ashwin Ramaswami, bdkopen, ChrisD, Epodax, JHS, jkdev, RamenChef, Robert Grant, Soaring Code, Squazz, Thomas Gerot, Ulrich Schwarz, Wolfgang
21	Маркировка компьютерного кода	4444, Naveen Gogineni, Shannon Young, SuperStormer, Tot Zam, unor

22	Маркировочные котировки	Content Solutions, mnoronha, unor
23	Медиа-элементы	feeela, Isak Combrinck, LisaMM, Shiva, Yossi Aharon
24	Мета-информация	Abhishek Pandey, Akshit Soota, Alexander Wigmore, Angelos Chalaris, Ani Menon, animuson, Anselm Urban, Bálint, bdkopen , Bookeater, Boris, coliff, Domenic, geek1011, Habel Philip, Hafidz Ilham Aji Permana, Himanshu Vaghela, insertusernamehere, jhoanna, JHS, kelvinelove, m_callens, Matt S, Michael Moriarty, Mr. Alien, Nishchay, Ortomala Lokni, Peter O., Safoor Safdar, Senjuti Mahapatra, Shannon Young, Stas Christiansen, Stephen Leppik, Ted Goas, Thomas Gerot, timmyRS, tmg, unor, VatsalSura, xims
25	Навигационные бары	Community
26	Пункты	Abrar Jahin, Thomas Gerot, Valor Naram
27	Пустотелые элементы	4444, ChrisD, Thomas Gerot, unor
28	Связывание ресурсов	AA2992, animuson, Anselm Urban, Aravind Suresh, Callan Heard, Chris Rutherfurd, cone56, DawnPaladin, Domenic, feeela, Henrique Barcelos, Infuzed guy, JHS, Lambda Ninja, Matas Vaitkevicius, Nhan, Thomas Gerot, unor, V4karian, vladdobra
29	Секционные элементы	Andrew Brooke, Anil, animuson, Hanif Formoly, nalply, Shannon Young, SuperBiasedMan, SuperStormer, Yossi Aharon
30	Символьные объекты	animuson, MervS, stack-learner
31	Списки	animuson, BiscuitBaker, Daniel Käfer, Grace Note, H. Pauwelyn , Jon Ericson, kcpike, Marvin, Matas Vaitkevicius, platy11, Prateek, Pseudonym Patel, Richard Hamilton, Right leg, Sayakiss, Stewartside, Thomas Gerot, tmg, Tom Johnson, unor, Zack
32	таблицы	albert, Alexandre N., animuson, Cedric Zoppolo, Eduardo Molteni, Grant Palin, J F, j08691, JHS, joe_young, Lambda Ninja, Mottie, Mr Lister, Nijin22, Prateek, PrAtik Lochawala, Praveen Kumar, Sildoreth, svarog, Ted Goas, tehciolo, Thomas Landauer, zer00ne
33	Форматирование	animuson, Ben Rhys-Lewis, Emil, gustavohenke, J F, Matas

	текста	Vaitkevicius, Peter L., Raystafarian, Stephen Leppik, Thomas Gerot, unor, Wolfgang
34	формы	Ali Almoullim, Ani Menon, animuson, Aown Muhammad, Chris Rutherfurd, Cullub, Gabriel Chi Hong Lee, Greg T, j08691, Kimmax, Luca langella, Niek Brouwer, Thomas Gerot
35	холст	cone56, Richard Hamilton, Roko C. Buljan, tonethar, Trevor Clarke, user4040648
36	Элемент Div	animuson, Araknid, Black Mamba, Chris, Content Solutions, D M, Faust, feeela, Gal Ratzkin, JHS, MySpeed, S.L. Barth, SuperStormer, Thomas Gerot
37	Элемент метки	Anthony Pham, fredden, Josiah Keller, m_callens, Roko C. Buljan
38	Элемент прогресса	animuson, Content Solutions, Richard Hamilton
39	Элементы меню выбора	Ali Almoullim, amflare, animuson, GentlePurpleRain, Ilyas karim , Mosh Feu, Tot Zam
40	Элементы управления вводом	Abhishek Pandey, Al.G., Alohci, amflare, Amitay Stern, Angelos Chalaris, Ani Menon, animuson, bhansa, Bob, Charlie H, Christophe Strobbe, CN, Community, cone56, Daniel, DawnPaladin, Dipen Shah, Domenic, Druzion, Edvin Tenovimas , Epodax, Franck Dernoncourt, gabe3886, geeksal, H. Pauwelyn, Henrique Barcelos, Huy Nguyen, J F, John Slegers, Kashyap Jha, Lahiru Ashan, Lankymart, Magisch, Marvin, Matas Vaitkevicius, Matt, Maximillian Laumeister, Mike McCaughan, morewry, Mosh Feu, Nathan Arthur, Nil Llisterri, Nishchay, niyasc, NoobCoder, Optimiser, Ortomala Lokni, Pi Programs, Pimgd, Prateek, Prav, Praveen Kumar, Psaniko, QoP, RamenChef, Ranjit Singh, Richard Hamilton, Robert Columbia, Roko C. Buljan, SeinopSys, Sharavnan Kv, Shivangi Chaurasia, SJDS, Squazz, Stephen Leppik, Stewartside, Sunny R Gupta, sv3k, the12, think123, Thomas Gerot, Timon, tmg, Tot Zam, trungk18, Undo, vladdobra, zzzzBov
41	Языки контента	animuson, FelipeAls, Gerold Broser, Isak Combrinck, Muntasir, Shannon Young, unor