Бесплатная электронная книга

учусь

jade

Free unaffiliated eBook created from
Stack Overflow contributors.

#jade

Около

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

It is an unofficial and free jade 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 jade.

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 нефритом

замечания

Мопс (ранее известный как нефрит) - это высокопроизводительный механизм шаблонов, сильно зависящий от Haml и реализованный с помощью JavaScript для Node.js и браузеров.

1. Производит HTML
2. Поддерживает динамический код
3. Поддерживает повторное использование (DRY)

Главная страница

Репозиторий на GitHub

Версии

версия	Дата выхода
0.0.2	2010-07-03
1.0.0	2013-12-22
1.11.0	2015-06-12

Examples

Установка или настройка

Прежде чем запускать код с помощью Pug, вам нужно иметь некоторые предварительные условия.

Вам нужно будет установить:

- NodeJS с NPM
- ExpressJS (необязательно)

После установки NodeJS вы можете проверить на своем терминале правильную установку:

```
$ node -v
```

В случае успеха он напечатает номер версии Node.

Чтобы установить Pug в ваш проект, предпочтительным и простым способом является NPM (Node Package Manager). Если вы знакомы с этим, просто выполните эту строку кода в терминале:

```
$ npm install pug
```

Если вы хотите установить глобально, вы можете ввести:

```
$ npm install pug-cli -g
```

и работать с

```
$ pug --help
```

Синтаксис

Мопс (старое имя - Jade) - это чистый, непрозрачный синтаксис для написания HTML. Вот простой пример:

```
doctype html
html(lang="en")
  head
    title= pageTitle
    script(type='text/javascript').
      if (foo) bar(1 + 5)
  body
    h1 Pug - node template engine
    #container.col
      if youAreUsingPug
        p You are amazing
      else
        p Get on it!
      p.
        Pug is a terse and simple templating language with a
        strong focus on performance and powerful features.
```

Производит следующий вывод как HTML

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Pug</title>
    <script type="text/javascript">
      if (foo) bar(1 + 5)
    </script>
  </head>
  <body>
    <h1>Pug - node template engine</h1>
    <div id="container" class="col">
      <p>You are amazing</p>
      <p>Pug is a terse and simple templating language with a strong focus on performance and
powerful features.</p>
```

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

Вот правила для преобразования Pug в HTML-код:

1. Отступом текста будет построено дерево HTML. отступы могут использоваться с пробелами или вкладками. Это нельзя смешивать!
2. HTML-теги написаны без `<` и `>` . Атрибуты - это места между круглыми скобками.
3. Комментарий может быть сделан с `//` или `<!-- -->` . Комментарии с `//-` не отображаются в отображаемом HTML.
4. С `#{ }` будет предложена модель: `#{header} #{user.username}` .
5. `#` (Hashtag) без фигурных скобок будет элементом `div` созданным с текстом в качестве идентификатора. Пример `#myID` будет отображаться как `<div id="myID"></div>` .
6. С `.` *(point)* будет `div` сгенерированный с атрибутом класса. Пример: `.myClass` будет отображаться как `<div class="myClass"></div>`
7. С `=` (знак равенства, за которым следует пробел), будет получена переменная. Exaple: `h1= title`
8. А `!=` (Не равно) извлекает переменную без экранирования.
9. А `-` (дефис) позволяет писать JavaScript. Пример: `- console.log("foo");`
10. Ссылка на внешний файл может `script(src="/js/chat.js")` следующим образом: `script(src="/js/chat.js")`
11. С помощью этого `script.` можно использовать встроенный `script.` ,
12. Директива для добавления основного макета: `extends ../layout` .
13. При `layout.pug` происходит вставка с использованием `block content`
14. Использование частичных файлов может осуществляться двумя способами:
    1. частичным `!= partial(template file name/options)` .
    2. Включить: `include ../includes/footer`
15. Обратное включение include `extend` . Это позволяет на странице «html `block` parts» отправлять на страницу макета, например: `extend layout`
16. Конкатенация происходит с символом `+` (плюс) или `#` (hashtag). Плюс используется в JavaScript-коде. Хештег в HTML и отображает содержимое: `p Имя: # {myName}

Использование мопса с Node.js

```
var pug = require('pug');

// compile
var fn = pug.compile('string of pug', options);
var html = fn(locals);

// render
var html = pug.render('string of pug', merge(options, locals));

// renderFile
var html = pug.renderFile('filename.pug', merge(options, locals));
```

Опции

- *filename* Используется в исключениях и требуется при использовании
- *compileDebug* Когда false, *команда* отладки не скомпилирована
- *pretty* Добавить довольно-отступ whitespace для вывода (false по умолчанию)

Прочитайте Начало работы с нефритом онлайн: https://riptutorial.com/ru/jade/topic/1709/начало-работы-с-нефритом

глава 2: harp js

Вступление

Harp - это статический веб-сервер со встроенной предварительной обработкой. Гарп может скомпилировать ваш проект до статических активов, HTML, CSS и JavaScript, без необходимости настройки. Вы также можете использовать Harp в качестве библиотеки узлов для компиляции или работы в качестве сервера.

По умолчанию арфы включают общие, полезные препроцессоры. Он служит Jade (Pug), Markdown, EJS, CoffeeScript, LESS, Sass и Stylus.

Examples

Как настроить арфу

Для начала Harp не требует настройки. Установите арфу в своем терминале с помощью команды: `npm install -g harp` .

```
$ sudo npm install -g harp
$ harp init myproject
$ harp server myproject
```

Прочитайте harp js онлайн: https://riptutorial.com/ru/jade/topic/9863/harp-js

глава 3: Атрибуты

Examples

Нормальные атрибуты

HTML-тег:

Атрибуты тега похожи на html, однако их значения - это обычный JavaScript.

```
a(href='google.com') Google
a(class='button', href='google.com') Google
```

Результат:

```
<a href="google.com">Google</a><a href="google.com" class="button">Google</a>
```

Переменная:

Все нормальные выражения JavaScript отлично работают:

```
- var authenticated = true
body(class=authenticated ? 'authed' : 'anon')
```

Результат:

```
<body class="authed"></body>
```

Многие атрибуты

Если у вас много атрибутов, вы также можете распространять их по многим строкам:

```
input(
  type='checkbox'
  name='agreement'
  checked
)
```

Результат:

```
<input type="checkbox" name="agreement" checked="checked"/>
```

Необязательные атрибуты

По умолчанию все атрибуты экранируются (заменяя специальные символы escape-последовательностями), чтобы предотвратить атаки, такие как межсайтовый скриптинг. Если вам нужно использовать специальные символы, вы можете использовать `!=` Вместо `=` .

Код:

```
div(escaped="<code>")
div(unescaped!="<code>")
```

Результат:

```
<div escaped="&lt;code&gt;"></div>
<div unescaped="<code>"></div>
```

Логические атрибуты

Логические атрибуты отражены Jade и принимают bools, иначе `true` или `false` . Если значение не указано, предполагается, что true.

Код:

```
input(type='checkbox', checked)
input(type='checkbox', checked=true)
input(type='checkbox', checked=false)
input(type='checkbox', checked=true.toString())
```

Результат:

```
<input type="checkbox" checked="checked"/>
<input type="checkbox" checked="checked"/>
<input type="checkbox"/>
<input type="checkbox" checked="true"/>
```

Если doctype является `html` jade, он не должен отражать атрибут и использует краткий стиль (понимаемый всеми браузерами).

Код:

```
doctype html
input(type='checkbox', checked)
input(type='checkbox', checked=true)
input(type='checkbox', checked=false)
input(type='checkbox', checked=true && 'checked')
```

Результат:

```
<!DOCTYPE html>
<input type="checkbox" checked>
<input type="checkbox" checked>
<input type="checkbox">
<input type="checkbox" checked="checked">
```

Атрибуты стиля

Атрибут `style` может быть строкой (как любой нормальный атрибут), но также может быть объектом, который удобен, когда части стиля генерируются JavaScript.

Код:

```
a(style={color: 'red', background: 'green'})
```

Результат:

```
<a style="color:red;background:green"></a>
```

Атрибуты класса

Атрибут `class` может быть строкой (как любой нормальный атрибут), но также может быть массивом имен классов, что удобно при создании из JavaScript.

Код:

```
- var classes = ['foo', 'bar', 'baz']
a(class=classes)
```

```
//- the class attribute may also be repeated to merge arrays
a.bing(class=classes class=['bing'])
```

Результат:

```
<a class="foo bar baz"></a><a class="bing foo bar baz bing"></a>
```

Это также могут быть имена классов сопоставления объектов с истинными или ложными значениями, что полезно для применения условных классов

Код:

```
- var currentUrl = '/about'
a(class={active: currentUrl === '/'} href='/') Home
a(class={active: currentUrl === '/about'} href='/about'
```

Результат:

```
<a href="/">Home</a><a href="/about" class="active">About</a>
```

Классный литерал

Классы могут быть определены с использованием синтаксиса `.classname` :

Код:

```
a.button
```

Результат:

```
<a class="button"></a>
```

Поскольку div - это такой общий выбор тега, он по умолчанию, если вы опускаете имя тега:

Код:

```
.content
```

Результат:

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

Идентификатор

Идентификаторы могут быть определены с использованием синтаксиса `#idname` :

Код:

```
a#main-link
```

Результат:

```
<a id="main-link"></a>
```

Поскольку div - это такой общий выбор тега, он по умолчанию, если вы опускаете имя тега:

Код:

```
#content
```

Результат:

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

и атрибуты

Выраженные «и атрибуты», синтаксис `&attributes` можно использовать, чтобы взорвать объект в атрибуты элемента.

Код:

```
div#foo(data-bar="foo")&attributes({'data-foo': 'bar'})
```

Результат:

```
<div id="foo" data-bar="foo" data-foo="bar"></div>
```

Объект не должен быть литералом объекта. Он также может быть просто переменной, которая имеет объект как его значение (см. Также атрибуты Mixin)

Код:

```
- var attributes = {'data-foo': 'bar'};
div#foo(data-bar="foo")&attributes(attributes)
```

Результат:

```
<div id="foo" data-bar="foo" data-foo="bar"></div>
```

Прочитайте Атрибуты онлайн: https://riptutorial.com/ru/jade/topic/6641/атрибуты

глава 4: случай

Examples

случай

```
- var friends = 10
case friends
  when 0
    p you have no friends
  when 1
    p you have a friend
  default
    p you have #{friends} friends
```

Результат:

```
<p>you have 10 friends</p>
```

Прочитайте случай онлайн: https://riptutorial.com/ru/jade/topic/4012/случай

кредиты

S. No	Главы	Contributors
1	Начало работы с нефритом	Community, David Dias, H. Pauwelyn, Huy Nguyen, Naeem Shaikh
2	harp js	Alice Tribuleva, Mohit Bhardwaj
3	Атрибуты	Huy Nguyen
4	случай	Huy Nguyen