Kostenloses eBook

LERNEN

jade

Free unaffiliated eBook created from
Stack Overflow contributors.

#jade

Inhaltsverzeichnis

Über

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

Kapitel 1: Erste Schritte mit Jade

Bemerkungen

Pug (früher bekannt als Jade) ist eine leistungsstarke Template-Engine, die stark von Haml beeinflusst und mit JavaScript für Node.js und Browser implementiert wurde.

1. Erzeugt HTML
2. Unterstützt dynamischen Code
3. Unterstützt die Wiederverwendbarkeit (TROCKEN)

Startseite

Repository auf GitHub

Versionen

Ausführung	Veröffentlichungsdatum
0,0,2	2010-07-03
1.0.0	2013-12-22
1.11.0	2015-06-12

Examples

Installation oder Setup

Bevor Sie mit Pug Code programmieren können, müssen Sie einige Voraussetzungen haben.

Sie müssen installieren:

- NodeJS mit NPM
- ExpressJS (optional)

Nach der Installation von NodeJS können Sie die korrekte Installation in Ihrem Terminal überprüfen:

```
$ node -v
```

Bei Erfolg wird die Nummer der Node-Version gedruckt.

Um Pug in Ihrem Projekt zu installieren, ist der bevorzugte und einfache Weg der NPM (Node Package Manager). Wenn Sie damit vertraut sind, führen Sie einfach diese Codezeile in Ihrem

Terminal aus:

```
$ npm install pug
```

Wenn Sie global installieren möchten, können Sie Folgendes eingeben:

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

und mit laufen

```
$ pug --help
```

Syntax

Pug (alter Name ist Jade) ist eine saubere, Whitespace-sensitive Syntax zum Schreiben von HTML. Hier ist ein einfaches Beispiel:

```
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.
```

Erzeugt folgende Ausgabe als 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>
```

Hier sind die Regeln zum Rendern von Pug in HTML-Code:

1. Durch Einrücken des Textes wird der HTML-Baum erstellt. Einrücken kann mit Leerzeichen oder Tabulatoren verwendet werden. Das konnte nicht gemischt werden!
2. HTML-Tags werden ohne `<` und `>` . Attribute sind Stellen zwischen runden Klammern.
3. Mit `//` oder `<!-- -->` Kommentar abgegeben werden. Kommentare mit `//-` sind im gerenderten HTML- `//-` nicht sichtbar.
4. Mit `#{ }` wird ein angebotenes Modell generiert: `#{header} #{user.username}` .
5. Das `#` (Hashtag) ohne geschweifte Klammern wird ein `div` Element mit dem Text als ID erstellt. Beispiel `#myID` wird als `<div id="myID"></div>` .
6. Mit einem `.` *(Punkt)* wird ein `div` mit einem Klassenattribut generiert. Beispiel: `.myClass` wird als `<div class="myClass"></div>`
7. Mit `=` (Gleichheitszeichen gefolgt von einem Leerzeichen) wird eine Variable abgerufen. Beispiel: `h1= title`
8. A `!=` (Ungleich) eine Variable abgerufen, ohne zu entkommen.
9. Mit einem `-` (Bindestrich) können Sie JavaScript schreiben. Beispiel: `- console.log("foo");`
10. Das Verknüpfen mit einer externen Datei kann folgendermaßen erfolgen: `script(src="/js/chat.js")`
11. Inline-Skript könnte dieses `script.` .
12. Eine Anweisung zum Hinzufügen des `extends ../layout` : `extends ../layout` .
13. Bei `layout.pug` geschieht das Einsetzen durch Verwendung `block content`
14. Die Verwendung von Teilweisen kann auf zwei Arten erfolgen:
    1. durch partiell `!= partial(template file name/options)` .
    2. Mit include: `include ../includes/footer`
15. Die Umkehrung von include ist `extend` . Dies ermöglicht es, von einer Seite "HTML- `block` " beispielsweise an eine Layoutseite zu senden: `extend layout`
16. Verketten geschieht mit dem Zeichen `+` (Plus) oder `#` (Hashtag). Das Plus wird im JavaScript-Code verwendet. Der Hashtag in HTML und stellt den Inhalt dar: `p Der Name lautet: #{meinName}

Pug mit Node.js verwenden

```
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));
```

Optionen

- *Dateiname* Wird in Ausnahmen verwendet und ist bei Verwendung von Includes erforderlich
- *compileDebug* Bei false werden keine Debug-Instrumente kompiliert
- *pretty* pretty-Einbuchtung Leerzeichen ausgeben (false per Default)

Erste Schritte mit Jade online lesen: https://riptutorial.com/de/jade/topic/1709/erste-schritte-mit-jade

Kapitel 2: Attribute

Examples

Normale Attribute

HTML-Tag:

Tag-Attribute sehen ähnlich aus wie HTML, ihre Werte sind jedoch nur reguläres JavaScript.

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

Ergebnis:

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

Variable:

Alle normalen JavaScript-Ausdrücke funktionieren ebenfalls gut:

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

Ergebnis:

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

Viele Attribute

Wenn Sie viele Attribute haben, können Sie sie auch über viele Zeilen verteilen:

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

Ergebnis:

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

Nicht maskierte Attribute

Standardmäßig werden alle Attribute mit Escapezeichen versehen (Ersetzen von Sonderzeichen durch Escape-Sequenzen), um Angriffe wie Cross Site Scripting zu verhindern. Wenn Sie Sonderzeichen verwenden müssen, können Sie `!=` Anstelle von `=` .

Code:

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

Ergebnis:

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

Boolesche Attribute

Boolesche Attribute werden von Jade gespiegelt und akzeptieren Bools, die als " `true` oder " `false` . Wenn kein Wert angegeben wird, wird "true" angenommen.

Code:

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

Ergebnis:

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

Wenn der Doctype `html` weiß Jade, das Attribut nicht zu spiegeln, und verwendet den knappen Stil (der von allen Browsern verstanden wird).

Code:

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

Ergebnis:

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

Stilattribute

Das `style` Attribut kann eine Zeichenfolge sein (wie jedes normale Attribut), es kann jedoch auch ein Objekt sein. Dies ist praktisch, wenn Teile des Styles von JavaScript generiert werden.

Code:

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

Ergebnis:

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

Klassenattribute

Das `class` kann eine Zeichenfolge sein (wie jedes normale Attribut), es kann jedoch auch ein Array von Klassennamen sein, was praktisch ist, wenn es aus JavaScript generiert wird.

Code:

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

Ergebnis:

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

Es kann sich auch um ein Objekt handeln, das Klassennamen True- oder False-Werten zuordnet, was zum Anwenden von bedingten Klassen nützlich ist

Code:

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

Ergebnis:

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

Klasse wörtlich

Klassen können mit der Syntax `.classname` definiert werden:

Code:

```
a.button
```

Ergebnis:

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

Da div eine solche gewöhnliche Wahl des Tags sind, ist dies die Standardeinstellung, wenn Sie den Tag-Namen weglassen:

Code:

```
.content
```

Ergebnis:

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

ID-Literal

IDs können mit einer `#idname` Syntax definiert werden:

Code:

```
a#main-link
```

Ergebnis:

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

Da div eine solche gewöhnliche Wahl des Tags sind, ist dies die Standardeinstellung, wenn Sie den Tag-Namen weglassen:

Code:

```
#content
```

Ergebnis:

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

& Attribute

Ausgesprochen "und Attribute", kann die Syntax `&attributes` verwendet werden, um ein Objekt in Attribute eines Elements zu zerlegen.

Code:

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

Ergebnis:

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

Das Objekt muss kein Objektliteral sein. Es kann auch nur eine Variable sein, die ein Objekt als Wert hat (siehe auch Mixin-Attribute).

Code:

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

Ergebnis:

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

Attribute online lesen: https://riptutorial.com/de/jade/topic/6641/attribute

Kapitel 3: Fall

Examples

Fall

```
- 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
```

Ergebnis ist:

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

Fall online lesen: https://riptutorial.com/de/jade/topic/4012/fall

Kapitel 4: Harfe js

Einführung

Harp ist ein statischer Webserver mit integrierter Vorverarbeitung. Harp kann Ihr Projekt aus statischen Assets, HTML, CSS und JavaScript zusammenstellen, ohne dass eine Konfiguration erforderlich ist. Sie können Harp auch als Node-Bibliothek zum Kompilieren oder als Server verwenden.

Harp enthält standardmäßig die gängigen, nützlichen Vorprozessoren. Es dient Jade (Pug), Markdown, EJS, CoffeeScript, LESS, Sass und Stylus.

Examples

Harp einrichten

Harp erfordert keine Konfiguration, um loszulegen. Installieren Sie Harp in Ihrem Terminal mit dem Befehl: `npm install -g harp` .

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

Harfe js online lesen: https://riptutorial.com/de/jade/topic/9863/harfe-js

Credits

S. No	Kapitel	Contributors
1	Erste Schritte mit Jade	Community, David Dias, H. Pauwelyn, Huy Nguyen, Naeem Shaikh
2	Attribute	Huy Nguyen
3	Fall	Huy Nguyen
4	Harfe js	Alice Tribuleva, Mohit Bhardwaj