EBook Gratuito

APPRENDIMENTO

jade

Free unaffiliated eBook created from
Stack Overflow contributors.

#jade

Sommario

Di

Capitolo 1: Iniziare con la giada

Osservazioni

Pug (precedentemente noto come jade) è un motore di template ad alte prestazioni fortemente influenzato da Haml e implementato con JavaScript per Node.js e browser.

1. Produce HTML
2. Supporta il codice dinamico
3. Supporta la riusabilità (DRY)

Pagina iniziale

Repository su GitHub

Versioni

versione	data di rilascio
0.0.2	2010-07-03
1.0.0	2013/12/22
1.11.0	2015/06/12

Examples

Installazione o configurazione

Prima di lanciare il codice con Pug, è necessario disporre di alcuni prerequisiti.

Dovrai installare:

- NodeJS con NPM
- ExpressJS (opzionale)

Dopo aver installato NodeJS, puoi controllare nel tuo terminale l'installazione corretta facendo:

```
$ node -v
```

Se ha successo, stamperà il numero della versione di Node.

Per installare Pug nel tuo progetto, il modo preferito e facile è tramite NPM (Node Package Manager). Se hai familiarità con questo, semplicemente esegui questa linea di codice nel tuo terminale:

```
$ npm install pug
```

Se si desidera eseguire l'installazione a livello globale, è possibile digitare:

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

e corri con

```
$ pug --help
```

Sintassi

Pug (il vecchio nome è Jade) è una sintassi pulita, sensibile allo spazio bianco per scrivere HTML. Qui c'è un semplice esempio:

```
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.
```

Produce il seguente output in formato 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>
```

Ecco le regole per il rendering del codice Pug in HTML:

1. Raggiando il testo, verrà creato l'albero HTML. il rientro potrebbe essere usato con spazi o tabulazioni. Questo non poteva essere mescolato!
2. I tag HTML sono scritti senza `<` e `>` . Gli attributi sono posti tra parentesi tonde.
3. Il commento potrebbe essere fatto con `//` o `<!-- -->` . I commenti con `//-` non sono visibili nel codice HTML visualizzato.
4. Con `#{ }` verrà generato un modello offerto: `#{header} #{user.username}` .
5. Il `#` (hashtag) senza parentesi graffe sarà un elemento `div` creato con il testo come ID. Esempio `#myID` sarà reso come `<div id="myID"></div>` .
6. Con a `.` *(punto)* sarà un `div` generato con un attributo di classe. Esempio: `.myClass` sarà reso come `<div class="myClass"></div>`
7. Con `=` (segno di uguaglianza seguito da uno spazio), verrà recuperata una variabile. Exaple: `h1= title`
8. A `!=` (Non uguale a) ha recuperato una variabile senza eseguire l'escape.
9. A `-` (trattino) consente di scrivere JavaScript. Esempio: `- console.log("foo");`
10. Il collegamento a un file esterno può essere il seguente: `script(src="/js/chat.js")`
11. Lo script inline potrebbe utilizzare questo `script.` .
12. Una direttiva per aggiungere il layout di base: `extends ../layout` .
13. A `layout.pug` avviene l'inserimento usando il `block content`
14. L'uso dei partial può avvenire in due modi:
    1. da parziale `!= partial(template file name/options)` .
    2. Per includi: `include ../includes/footer`
15. L'inverso di include è `extend` . Ciò consente da una pagina "parti di `block` html" di inviare ad una pagina di layout, ad esempio: `extend layout`
16. La concatenazione avviene con il carattere `+` (più) o `#` (hashtag). Il vantaggio è utilizzato al codice JavaScript. L'hashtag in HTML e rende il contenuto: `p Il nome è: # {myName}

Usando pug con 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));
```

Opzioni

- *nomefile* Utilizzato nelle eccezioni e richiesto quando si utilizza include
- *compileDebug* Quando false non viene compilata alcuna strumentazione di debug
- *pretty* Aggiungi spazi vuoti di rientranza in uscita (false di default)

Leggi Iniziare con la giada online: https://riptutorial.com/it/jade/topic/1709/iniziare-con-la-giada

Capitolo 2: arpa js

introduzione

Harp è un server Web statico con preelaborazione integrata. Harp può compilare il tuo progetto con risorse statiche, HTML, CSS e JavaScript, senza alcuna configurazione necessaria. È anche possibile utilizzare Harp come libreria di nodi per la compilazione o l'esecuzione come server.

L'arpa include i preprocessori comuni e utili per impostazione predefinita. Serve Jade (Pug), Markdown, EJS, CoffeeScript, LESS, Sass e Stylus.

Examples

Come configurare Arpa

Arpa non richiede alcuna configurazione per iniziare. Installa Harp nel tuo terminale usando il comando: `npm install -g harp` .

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

Leggi arpa js online: https://riptutorial.com/it/jade/topic/9863/arpa-js

Capitolo 3: Astuccio

Examples

Astuccio

```
- 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
```

Il risultato è:

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

Leggi Astuccio online: https://riptutorial.com/it/jade/topic/4012/astuccio

Capitolo 4: attributi

Examples

Attributi normali

Tag HTML:

Gli attributi dei tag hanno un aspetto simile a html, tuttavia i loro valori sono solo normali JavaScript.

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

Risultato:

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

Variabile:

Anche tutte le normali espressioni JavaScript funzionano bene:

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

Risultato:

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

Molti attributi

Se hai molti attributi, puoi anche distribuirli su più righe:

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

Risultato:

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

Attributi senza escape

Per impostazione predefinita, tutti gli attributi sono sfuggiti (sostituendo caratteri speciali con sequenze di escape) per prevenire attacchi come lo scripting cross-site. Se hai bisogno di usare caratteri speciali puoi usare `!=` Invece di `=` .

Codice:

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

Risultato:

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

Attributi booleani

Gli attributi booleani sono specchiati da Jade e accettano bool, ovvero `true` o `false` . Quando non viene specificato alcun valore, viene assunto true.

Codice:

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

Risultato:

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

Se il doctype è `html` jade sa di non rispecchiare l'attributo e usa lo stile terse (compreso da tutti i browser).

Codice:

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

Risultato:

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

Attributi di stile

L'attributo `style` può essere una stringa (come qualsiasi attributo normale) ma può anche essere un oggetto, il che è utile quando parti di stile sono generate da JavaScript.

Codice:

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

Risultato:

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

Attributi di classe

L'attributo `class` può essere una stringa (come qualsiasi attributo normale) ma può anche essere una matrice di nomi di classi, che è utile quando generata da JavaScript.

Codice:

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

Risultato:

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

Può anche essere un oggetto che associa i nomi di classe a valori veri o falsi, utile per l'applicazione di classi condizionali

Codice:

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

Risultato:

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

Classe letterale

Le classi possono essere definite usando una sintassi `.classname` :

Codice:

```
a.button
```

Risultato:

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

Poiché le div sono una scelta così comune di tag, è l'impostazione predefinita se ometti il nome del tag:

Codice:

```
.content
```

Risultato:

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

ID letterale

Gli ID possono essere definiti usando la sintassi di `#idname` :

Codice:

```
a#main-link
```

Risultato:

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

Poiché le div sono una scelta così comune di tag, è l'impostazione predefinita se ometti il nome del tag:

Codice:

```
#content
```

Risultato:

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

e attributi

Pronunciata "e attributi", la sintassi `&attributes` può essere usata per far esplodere un oggetto in attributi di un elemento.

Codice:

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

Risultato:

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

L'oggetto non deve essere un oggetto letterale. Può anche essere solo una variabile che ha un oggetto come valore (vedi anche Mixin Attributes)

Codice:

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

Risultato:

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

Leggi attributi online: https://riptutorial.com/it/jade/topic/6641/attributi

Titoli di coda

S. No	Capitoli	Contributors
1	Iniziare con la giada	Community, David Dias, H. Pauwelyn, Huy Nguyen, Naeem Shaikh
2	arpa js	Alice Tribuleva, Mohit Bhardwaj
3	Astuccio	Huy Nguyen
4	attributi	Huy Nguyen