edspencer / jaml

JavaScript Haml — Read more

http://edspencer.net/2009/11/jaml-beautiful-html-generation-for-javascript.html

- HTTP

- Git Read-Only

https://github.com/edspen

Read-Only access

version 0.1.0

erikvold (author)
January 29, 2011
commit ccb8b471bfb06b846c99
tree c44b4b354621b0339e2f
parent 15ea81c738882298ef27
jaml /

name	age	history message
examples/	January 23, 2010	CommonJS compatibility + Jakefile to automate buil... [tlrobinson]
lib/	January 29, 2011	version 0.1.0 [erikvold]
specs/	January 29, 2011	Closes #17 Allowing users to specify the this obje... [erikvold]
src/	January 29, 2011	Closes #17 Allowing users to specify the this obje... [erikvold]
.gitmodules	October 20, 2009	Initials [Ed Spencer]
Jakefile	January 23, 2010	CommonJS compatibility + Jakefile to automate buil... [tlrobinson]
Jaml-all.js	January 29, 2011	version 0.1.0 [erikvold]
README.textile	January 27, 2011	README edit for url to jasmine-node [erikvold]
package.json	January 29, 2011	version 0.1.0 [erikvold]

README.textile

Jaml: beautiful HTML generation for JavaScript

Jaml tries to emulate Ruby's Haml library, making it easy to generate HTML in your JavaScript projects.

Examples

Something Simple

Registering a template is easy:

```
Jaml.register('simple', function() {
  div(
    h1("Some title"),
```

```
    p("Some exciting paragraph text"),
    br(),

    ul(
      li("First item"),
      li("Second item"),
      li("Third item")
    )
  );
});
```

So is rendering it:

```
Jaml.render('simple');
```

Here's the output (yes, the indentation really is that pretty):

```
<div>
  <h1>Some title</h1>
  <p>Some exciting paragraph text</p>
  <br />
  <ul>
    <li>First item</li>
    <li>Second item</li>
    <li>Third item</li>
  </ul>
</div>
```

Templating

Usually we want to inject data into templates – let's see how to do that:

```
Jaml.register('product', function(product) {
  div({cls: 'product'},
    h1(product.title),

    p(product.description),

    img({src: product.thumbUrl}),
    a({href: product.imageUrl}, 'View larger image'),

    form(
      label({for: 'quantity'}, "Quantity"),
      input({type: 'text', name: 'quantity', id: 'quantity', value: 1}),

      input({type: 'submit', value: 'Add to Cart'})
    )
  );
});
```

And now to render it:

```
//this is the product we will be rendering
var bsg = {
  title      : 'Battlestar Galactica DVDs',
```

```
  thumbUrl   : 'thumbnail.png',
  imageUrl   : 'image.png',
  description: 'Best. Show. Evar.'
};

Jaml.render('product', bsg);
```

Which gives us:

```
<div class="product">
  <h1>Battlestar Galactica DVDs</h1>
  <p>Best. Show. Evar.</p>
  <img src="thumbnail.png" />
  <a href="image.png">View larger image</a>
  <form>
    <label for="quantity">Quantity</label>
    <input type="text" name="quantity" id="quantity" value="1"></input>
    <input type="submit" value="Add to Cart"></input>
  </form>
</div>
```

Collections and partials

We can reuse templates inside other templates. Here we make a Category template to hold more than one product:

```
Jaml.register('category', function(category) {
  div({cls: 'category'},
    h1(category.name),
    p(category.products.length + " products in this category:"),

    div({cls: 'products'},
      Jaml.render('product', category.products)
    )
  );
});
```

Now we render it with a couple of products:

```
//here's a second product
var snowWhite = {
  title      : 'Snow White',
  description: 'not so great actually',
  thumbUrl   : 'thumbnail.png',
  imageUrl   : 'image.png'
};

//and a category
var category = {
  name    : 'Doovde',
  products: [bsg, snowWhite]
}

Jaml.render('category', category);
```

Which gives us:

```
<div class="category">
  <h1>Doovde</h1>
  <p>2 products in this category:</p>
  <div class="products"><div class="product">
  <h1>Battlestar Galactica DVDs</h1>
  <p>Best. Show. Evar.</p>
  <img src="thumbnail.png" />
  <a href="image.png">View larger image</a>
  <form>
    <label for="quantity">Quantity</label>
    <input type="text" name="quantity" id="quantity" value="1"></input>
    <input type="submit" value="Add to Cart"></input>
  </form>
</div>
<div class="product">
  <h1>Snow White</h1>
  <p>not so great actually</p>
  <img src="thumbnail.png" />
  <a href="image.png">View larger image</a>
  <form>
    <label for="quantity">Quantity</label>
    <input type="text" name="quantity" id="quantity" value="1"></input>
    <input type="submit" value="Add to Cart"></input>
  </form>
</div>
</div>
</div>
```

Error handling

If the requested template does not exist, the renderer will return null:

```
=> Jaml.render('missing');
=> null
```

Jaml Tests

You can run the Jaml test suite using either node.js at the command line or via a webpage-based runner.

Run the tests in a browser

Jasmine must be checked out in a directory alongside jaml:

```
git clone https://github.com/pivotal/jasmine.git
ls
=> jaml jasmine
```

…then open specs/index.html in your browser.

Run the tests using node.js

1) Install node.js.

2) Check out sconover's jasmine-node alongside jaml:

```
git clone https://github.com/sconover/jasmine-node.git
ls
=> jaml jasmine-node
```

3) Run the suite:

```
$ node specs/suite.js
Started
.........................

Finished in 0.018 seconds
X tests, Y assertions, 0 failures
```


Markdown Cheat Sheet

Format Text

Headers

```
# This is an <h1> tag
## This is an <h2> tag
###### This is an <h6> tag
```

Text styles

```
*This text will be italic*
```

```
_This will also be italic_
**This text will be bold**
__This will also be bold__

*You **can** combine them*
```

Lists

Unordered

```
* Item 1
* Item 2
  * Item 2a
  * Item 2b
```

Ordered

```
1. Item 1
2. Item 2
3. Item 3
   * Item 3a
   * Item 3b
```

Miscellaneous

Images

```
![GitHub Logo](/images/logo.png)
Format: ![Alt Text](url)
```

Links

```
http://github.com - automatic!
[GitHub](http://github.com)
```

Blockquotes

```
As Kanye West said:
> We're living the future so
> the present is our past.
```

Code Examples in Markdown

Syntax highlighting with GFM

````
```javascript
function fancyAlert(arg) {
  if(arg) {
    $.facebox({div:'#foo'})
  }
}
```
````

Or, indent your code 4 spaces

```
Here is a Python code example
without syntax highlighting:

    def foo:
      if not bar:
        return true
```

Inline code for comments

```
I think you should use an
`<addr>` element here instead.
```