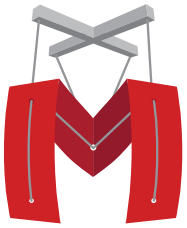

v2.4.4

Marionette.Renderer

The `Renderer` object was extracted from the `ItemView` rendering process, in order to create a consistent and re-usable method of rendering a template with or without data.

Documentation Index

Basic Usage

The basic usage of the `Renderer` is to call the `render` method. This method returns a string containing the result of applying the template using the `data` object as the context.

```
var template = "#some-template";
var data = {foo: "bar"};
var html = Marionette.Renderer.render(template, data);
```

```
// do something with the HTML here
```

If you pass a `template` that coerces to a falsey value, the `render` method will throw an exception stating that there was no template provided.

Pre-compiled Templates

If the `template` parameter of the `render` function is itself a function, the renderer treats this as a pre-compiled template and does not try to compile it again. This allows any view that supports a `template` parameter to specify a pre-compiled template function as the `template` setting.

```
var myTemplate = _.template("<div>foo</div>");
Marionette.ItemView.extend({
  template: myTemplate
});
```

The template function does not have to be any specific template engine. It only needs to be a function that returns valid HTML as a string from the `data` parameter passed to the function.

Custom Template Selection And Rendering

By default, the renderer will take a jQuery selector object as

the first parameter, and a JSON data object as the optional second parameter. It then uses the `TemplateCache` to load the template by the specified selector, and renders the template with the data provided (if any) using Underscore.js templates.

If you wish to override the way the template is loaded, see the `TemplateCache` object.

If you wish to override the template engine used, change the `render` method to work however you want:

```
Marionette.Renderer.render = function(template, data){
  return $(template).tmpl(data);
};
```

This implementation will replace the default Underscore.js rendering with jQuery templates rendering.

If you override the `render` method and wish to use the `TemplateCache` mechanism, remember to include the code necessary to fetch the template from the cache in your `render` method:

```
Marionette.Renderer.render = function(template, data){
  var template = Marionette.TemplateCache.get(template);
  // Do something with the template here
```

```
};
```

Using Pre-compiled Templates

You can easily replace the standard template rendering functionality with a pre-compiled template, such as those provided by the JST or TPL plugins for AMD/RequireJS.

To do this, just override the `render` method to return your executed template with the data.

```
Marionette.Renderer.render = function(template, data){
  return template(data);
};
```

Then you can specify the pre-compiled template function as your view's `template` attribute:

```
var myPrecompiledTemplate = _.template("<div>some template</div>");

Marionette.ItemView.extend({
  template: myPrecompiledTemplate
});
```