

Templates

Options for dynamically building HTML

- JavaScript
 - `var text = document.getElementById("#label");`
 - `text.innerHTML = "Hello world";`
- JQuery
 - `$("label").text("Hello world");`
- Templates
 - Mustache, Pug (Jade), EJS, Handlebars, ...
- Template language + library
 - JSX + React
- Framework
 - Angular, Vue

Building HTML in JQuery

```
var title = "Constructing HTML Elements";
```

```
var container = $("<div>");
```

```
container.addClass("tutorial");
```

```
var h1 = $("<h1>");
```

```
h1.text(title);
```

```
h1.addClass("tutorial-heading");
```

```
container.append(h1);
```

```
$("body").append(container);
```

Building HTML with Strings

```
var title = "Constructing HTML Elements";

var html = [
    '<div class="tutorial">',
        '<h1 class="tutorial-heading">' + title + '<h1>',
    '</div>'
].join("\n");

// html: '<div ...>\n<h1 ...>Constructing HTML Elements<h1>\n</div>'

$("body").append(html);
```

Templates

- Goals
 - Move Javascript out of HTML
 - Separate logic and display
 - Create a readable, maintainable language to write views
 - Modularize the views so you can reuse pieces
 - Use variables in HTML

Move HTML out of the Javascript file:

```
...  
<script id="tutorial-template" type="text/template">  
  <div class="tutorial">  
    <h1 class="tutorial-heading">{{title}}</h1>  
  </div>  
</script>  
<div id="main"></div>  
...
```

HTML

```
var data = {  
  title: "Constructing HTML Elements"  
}  
  
var template = $("#tutorial-template").html();  
  
var html = Mustache.render(template, data);  
$("#main").append(html);
```

JS

Example Template Libraries

- Mustache
 - logic-less templates - no control flow in tags
- Handlebars
 - built on Mustache - some minimal logic
 - client-side example, but can compile templates on the server
- EJS (Embedded JavaScript)
 - Mix JavaScript and HTML together.
- PUG (was Jade)
 - Server-side
 - Different layout for specifying

Handlebars

- HTML plus expressions
- templates included in a `<script>` tag
- Adds a few helper methods: `if`, `each`, `unless`
- Custom helper methods


```
<h1>List of Animals and Sounds</h1>
<div id="animalList">
</div>
```

```
<!-- HANDLEBARS TEMPLATE -->
```

```
<script id="animalTemplate" type="text/x-handlebars-template">
  {{#if animals}}
    <ul>
      {{#each animals}}
        <li class="animal">{{type}} says {{sound}}</li>
      {{/each}}
    </ul>
  {{else}}
    <p>There are no animals.</p>
  {{/if}}
```

```
var source =
  document.getElementById("animalTemplate").innerHTML;

var template = Handlebars.compile(source);

// data
var data = {animals: [
  {type: "Dog", sound: "woof"},
  {type: "Cat", sound: "meow"},
  {type: "Cow", sound: "moo"}
]};

var output = template(data);

document.getElementById("animalList").innerHTML =
  output;
```

Pug (formerly Jade)

- Simple tags - uses indentation and white space for nesting
- Need to learn the syntax
- Can embed Javascript
- Partials
- Pass variables from `render()` call

Pug example

```
//layout.pug
html
  title "Express views"
  body
    include ./partials/header

    block content

    include ./partials/footer
```

```
// index.pug
extends layout

block content
  h1=pageTitle
  p='Our first view'
  - var x = 5;
  div
    ul
      - for (var i=1; i<=x; i++) {
        li= i + ". Hello"
      - }
```

```
var express = require('express');
var path = require('path');
var app = express();

// setup views
app.locals.basedir = __dirname + '/views';
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'pug');

app.get('/', function(req, res) {
  var title = "Home Page"
  res.render('index',
    {"pageTitle": title});
});

app.listen(3000, function() {
  console.log('App is running.');
```

EJS

- Embedded Javascript
- More traditional approach to templates
- Standard HTML
- Plus `<% %>` to enclose JavaScript code
- And `<%= %>` to insert variables

EJS

```
<h1><%= title %></h1>
```

```
<ul>
```

```
  <% for(var i=0; i<languages.length; i++) { %>
```

```
    <li>
```

```
      <%= languages[i] %>
```

```
    </li>
```

```
  <% } %>
```

```
</ul>
```

Client-side/Server-side

- Traditionally, templates are compiled and processed on the server
- As more code is pushed to the client, the templates can also be processed using Javascript on the client

React

- Library that allows you to build components.
- JSX is an HTML-like syntax that compiles to Javascript
 - another template language
- Instead of trying to separate logic and display, separates components