Modern Technology

CS-546 – WEB PROGRAMMING

JavaScript Templating

What do we mean by templating?

Last week, we were able to create elements from JavaScript Objects by manually making a series of jQuery objects.

There are many libraries that make this much easier by allowing you to generate HTML from an HTML string and some sort of object.

For the purpose of this course, I suggest the **Handlebars** library to perform templating.

What is Handlebars?

Handlebars is a very popular templating library. It takes some template string through a variable or a template tag and then proceeds to parse it into an **Abstract Syntax Tree**, which it will then populate with data.

This allows you to parse a template once, and then reuse it many times with new data.

Many frameworks, like AngularJS and Ember, use the same general syntax as Handlersbars, despite having different internal workings.

How do I use Handlebars?

After your template is referenced, you will pass data to it:

```
<script id="my-template" type="text/x-handlebars-template">
<div class="entry">
  <h1>{{title}}</h1>
  <div class="body">
    {{body}}
  </div>
</div>
</script>
```

```
var template = '"<div class="entry">  <h1>{{title}}</h1>  <div class="body">    {{body}}  </div></div>"';
undefined
console.log(template)
"<div class="entry">  <h1>{{title}}</h1>  <div class="body">    {{body}}  </div></div>"
```

How do I use Handlebars?

After your template is referenced, you will pass data to it, compile it, and insert the resulting HTML:

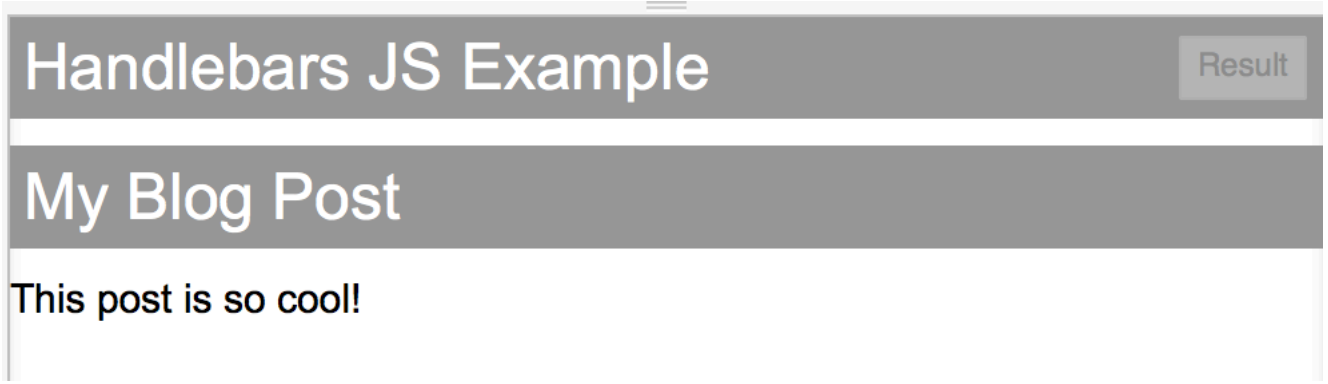
```
var source = $("#my-template").html();
var template = Handlebars.compile(source);

var data = {
  title: "My Blog Post",
  body: "This post is so cool!"
};

$('body').append(template(data));
```

How do I use Handlebars?

This results in a compiled HTML version with your data in it.

Handlebars also supports creating custom ways to display variables, such as parsing users to print their full names from a JSON Object that has their first name and last name.

This will allow you to define a series of templates, compile the sources, and then create new HTML easily with that data.



Modern Tech

What's new in Web Dev Technology?

You now have build steps in frontend technology

- JavaScript and CSS are compiled

Servers often act entirely as APIs

- They now often offload work to other machines to do

There are many other technologies besides PHP to use on the web now.

What will I see in a modern web app?

A modern web application will often have a focus on:

- User experience
- Fast loading time
- API Style Design

For the user-facing portion, you will often be creating Single Page Applications.

The backend portion of web applications will now often act as APIs, meaning they will simply take queries and respond to them accordingly.

After the user-facing portion of the query is handled, the backend will often offload work to **worker roles**, which are servers that do heavy duty computation (such as OCR, collecting data from other APIs, and so on) that will perform the bulk of heavy duty work.

Modern Backends

As we said before, modern backends are now often just APIs that have *other* components performing work for them. These APIs are written in any number of languages, but very commonly are:

- NodeJS
- Ruby (using the Rails framework)
- C# / ASP.Net / Web API

NodeJS

NodeJS is a JavaScript environment that runs on your desktop, rather than on your browser. It has the syntax of JavaScript that you are used to, with additional features.

NodeJS has a massive package manager, allowing you to easily set it up and running; you can download a package that runs a server for you, and create a self-contained web server in under 5 minutes.

NodeJS runs off a module system, meaning you require modules that you will need to perform your code and export modules that you want made available to other portions of your application.

The learning curve for NodeJS is minimal, as you have been programming in JavaScript throughout the semester.

Ruby (On Rails)

Ruby is an elegant programming language that focuses on code readability and making it as descriptive as possible. Ruby has many ways to perform the same task; it is the epitome of a DAMP coding language, as it is meant to be readable before performant.

Ruby, like JavaScript, is loosely typed!

Ruby on Rails (Rails) is a very popular MVC Framework for Ruby; it provides a server and an easy to use setup to quickly create a web application where the data is organized neatly.

Ruby's got a fairly decent learning curve, as the language and framework are very opinionated on how things should be done and you often have to learn how to do something "The Ruby Way".

C# / ASP.NET MVC / Web API

Microsoft's C# is an Object Oriented Programming . It is **strongly typed and compiled**!

C# is designed to run work with the Microsoft .NET framework, meaning there is a large amount of tools and libraries available to use.

ASP.NET is an open source server-side application framework; this allows you to use any language available on the .NET Framework to write a web application.

ASP.NET Web API is a framework that allows you to easily interpret and respond to HTTP Calls; it is very low level, and as such it is very customizable.

Because it is compiled, it generally responds and runs faster, as it does not need to be interpreted each time the page is hit.

Offloading Labor onto Workers

As we've said, it is very common to offload heavy processing labor onto workers. These workers can be written and run in any language you desire, as they are not interacting with your user.

One such example could be running Optical Character Recognition on images. You would achieve this through the following steps:

- Upload an image file through the API (10 seconds)
- Offload the OCR work to a worker (1 second)
- Return an ID that will lead the user to the result (1 second)
- The worker will then perform OCR work (180 seconds)
- The worker will place detected words in a database entry related to that ID (5 seconds)
- The user, at some point, will be able to see results

This accomplishes two things:

- The user will not be left waiting for the result, but will have to re-fetch it at some point.
- Your web server will not be using up valuable resources performing this computation; a worker will!

JS and CSS Compilers

Babel

Babel is a plugin-based JavaScript compiler.

- You can create a new plugin in order to accomplish any task, such as changing **true/false** to **!0** and **!1.**

Popular plugins are:

- React
- ES2015 to ES5 safe code
- JSX
- Async Generators

Babel is becoming more and more common as compiling JavaScript becomes more and more popular. It allows you to write next generation code that works today.

TypeScript

TypeScript, an open source project spearheaded by Microsoft that allows you to write a superset of JavaScript that is type constrained.

It allows for static type checking modern features such as lexical binding, classes, and mixins.

Typescript on left, JS on right.

```typescript
class Greeter {
    greeting: string;
    constructor(message: string) {
        this.greeting = message;
    }
    greet() {
        return "Hello, " + this.greeting;
    }
}

var greeter = new Greeter("world");
```

```javascript
var Greeter = (function () {
    function Greeter(message) {
        this.greeting = message;
    }
    Greeter.prototype.greet = function () {
        return "Hello, " + this.greeting;
    };
    return Greeter;
})();
var greeter = new Greeter("world");
```

LESS

LESS is a CSS Pre-Processor; it allows you to write CSS with additional features such as variables, functions, and such; LESS must be compiled into proper CSS.

```less
@base: #f938ab;

.box-shadow(@style, @c) when (iscolor(@c)) {
  -webkit-box-shadow: @style @c;
  box-shadow:         @style @c;
}
.box-shadow(@style, @alpha: 50%) when (isnumber(@alpha)) {
  .box-shadow(@style, rgba(0, 0, 0, @alpha));
}
.box {
  color: saturate(@base, 5%);
  border-color: lighten(@base, 30%);
  div { .box-shadow(0 0 5px, 30%) }
}
```

```css
.box {
  color: #fe33ac;
  border-color: #fdcdea;
}
.box div {
  -webkit-box-shadow: 0 0 5px rgba(0, 0, 0, 0.3);
  box-shadow: 0 0 5px rgba(0, 0, 0, 0.3);
}

```

SASS

SASS is another CSS Preprocessor, with all the features of LESS. It supports more and easier programming, such as conditional statements.

SASS is currently edging out LESS and becoming more commonly used.

SCSS

```scss
@mixin border-radius($radius) {
  -webkit-border-radius: $radius;
    -moz-border-radius: $radius;
     -ms-border-radius: $radius;
        border-radius: $radius;
}

.box { @include border-radius(10px); }
```

CSS Compiled in 0.002s

```css
.box {
  -webkit-border-radius: 10px;
  -moz-border-radius: 10px;
  -ms-border-radius: 10px;
  border-radius: 10px;
}
```

JSX

JSX is a drafted XML-Like syntax proposed for JavaScript extensions that allows programmers to more easily create XML-Like data (such as HTML) inside of their JavaScript through the use of a pre-processor or compiler of some sort. Babel is commonly used to compile JSX.

An example of JSX is on the left, a compiled JavaScript for React version on the right.

```
var Hello = React.createClass({
    render: function() {
        return <div>Hello {this.props.name}
</div>;
    }
});

ReactDOM.render(
    <Hello name="World" />,
    document.getElementById('container')
);

```

```
var Hello = React.createClass({displayName: 'Hello',
    render: function() {
        return React.createElement("div", null, "Hello
", this.props.name);
    }
});

ReactDOM.render(
    React.createElement(Hello, {name: "World"}),
    document.getElementById('container')
);

```

Compiling our work

Different tech systems allow you to compile these types of languages differently.

- Ruby has gems that compile
- Node has libraries that will compile them for you.

It is becoming more and more common to use a Task Runner, such as **Gulp**, and plugins for that task runner to compile code for you.

This also allows you to add many additional steps, such as minifying and concatenating your scripts, creating source maps, and so on.

Common Build Tasks

In frontend development, you often perform two additional steps:

- **Minification**: The process of renaming all variables, stripping all whitespace, and so on to drastically reduce the size of your JavaScript or CSS file
- **Concatenation**: The process of taking multiple files and combining them into one file.

Task runners like Gulp make it easy to perform these tasks to ensure that your user downloads the smallest file possible. Using Gulp and such, you could create a build script that automatically:

- Watches for a LESS file changes
- On file change, recompile all your LESS files
- Copy all the files to a folder called `css/*`
- Combine their output
- Output combined version to file `css/site.css`
- Minify the output further
- Output combined and minified version to `css/site.min.css`

JavaScript Frameworks

React

React is a new library that is rapidly becoming popular that allows programmers to easily create *components* to their web application.

React renders **extraordinarily fast**, due to creating a virtual copy of the DOM and only changing things that update; it essentially runs a fast difference check.

If you are looking for any frontend technology to experiment with, I suggest you check out React.

AngularJS

AngularJS is a full fledged, single paged application framework. It forces you to organize your code into modules, classes, services, factories, templates, and so on.

By forcing you to split-up your code, AngularJS makes it very easy to create large scale applications by forcing good patterns to be followed. AngularJS allows you to update data in your JavaScript code and have the view automatic refresh without calling re-rendering functions.

AngularJS 1.x is currently still being developed and maintained.

AngularJS is backed by Google, so it's pretty well supported by top tier programmers.

Angular 2

Angular 2 is essentially a different framework from Angular 1.x, but is a spiritual successor. It will take the best parts of Angular 1.x, such as the components handled in directives, and branch off of those more powerfully.

It takes away parts that did not end up being as good as they were expected to be, such as two way data binding.

Angular 2 is written in TypeScript, and in order to most easily write Angular 2 code you will want to write it in Typescript.

Other Frameworks

Ember.js is similar to Handlebars in that it relies heavily on templating using **{{ var_name }}** syntax**,** but takes on components similar to Angular and React to make reusable components. Like Angular, it will auto-update JavaScript data in your view.

Meteor is a whole entire framework that allows updates to exist in real time synchronization between the client, the server, and the database using websockets. It is an **entire** technology stack.

Frontend Boilerplates

Bootstrap 3

Twitter Bootstrap 3 is an HTML, CSS, and JavaScript framework that acts as a good launching point to quickly get a website up and running and not worry about the design or common layout elements, such as modals and alerts.

Bootstrap 3 offers:

- A responsive layout; as your screen size increases, so too will the layout
- Many accessibility enhancements out of the box; people with screen readers and such will be able to easily interpret Bootstrap pages
- A grid; you can design elements based on rows and columns easily.
- A barebone style for most common elements that is designed to be uniform throughout your website.

Bootstrap 4

Bootstrap 4 builds off of the Bootstrap 3, but will be making several changes:

- Moved from LESS to SASS
- Improvements to the Grid
- Greater degree of customizability
- Rewritten JS Plugins for better modularity
- Tons more

Bootstrap 4 is currently in open alpha, meaning you can download it and start using it, but expect changes to Bootstrap 4 to be breaking towards the code or layout you write.

Zurb's Foundation

Zurb's Foundation is another responsive framework; it was designed to be mobile first, meaning that its set of rules focus on smaller devices and add more rules in for larger devices.

Zurb has recently pushed to be much more accessibility oriented.

Zurb provides one page that shows every aspect of Foundation, so you can see what it looks like and how it's used very easily:

- http://foundation.zurb.com/sites/docs/kitchen-sink.html#accordion

Semantic UI

Semantic UI is another frontend framework that is growing in popularity due to its ease of use. It provides a series of class names that are easy to understand such as ***button***, to make something look like a button, or ***divided grid***, to make a grid that has visible divider lines.

Semantic UI, like Bootstrap and Foundation, provides HTML, CSS, and JavaScript components.

Single Page Applications

What is a Single Page Application?

A Single Page Application (**SPA**) is a web application where the user never *actually* leaves the page, but sees more data by requesting relevant data via AJAX calls and rendering accordingly.

This has many upsides:

- Allows for quick navigation from app state to app state due to only downloading data, not layout
- You only need to request the data you want

But also several downsides:

- Slower initial page load as you download an application
- Has to account for more application states, such as network interruptions
- Rendering HTML on the client side can be costly (CPU wise)
- Added complexity
- Relies on the user to have a browser that supports all the features you use

React

With the help of several other libraries, it is becoming more and more common to create a SPA with React.

Out of the box, React allows you to create reusable web components that have self contained logic.

With some added code to handle the state of your application, you can therefore use React to change your page **very quickly** each time that your user navigates to somewhere else, or new data is created.

AngularJS

AngularJS was built for SPAs; you define pages in terms of directives (components) or controllers that isolate and contain all their logic, and place it in a relevant view.

AngularJS is essentially the de-facto SPA framework at the moment. There are hundreds of jobs open entirely just for AngularJS developers.

Mobile

What does mobile development have to do with web development?

As web developers, your skills can be easily repurposed to be used on mobile **and** desktops applications.

There are frameworks that allow you to create a WebView (think, a browser) and create an entire application that runs in the context of a user's phone that appears to be a native application.

These 'web applications' often suffer a performance hit, however, as they are working within the limitations of a browsers; there is often a lot of overhead you don't care about running.

What is Cordova?

Apache Cordova is a framework that allows you to create and package HTML, CSS, and JavaScript web pages inside of a WebView (an HTML/CSS/JS renderer running inside of your phone).

Cordova allows you to interface with common device functions, such as limited manipulation of the file system. You may then access these functions through a JavaScript API.

Cordova allows you to create **cross-platform-applications**, as these plugins can often work on multiple different phone operating systems such as Android, iOS, and WP8.

- Plugins have two components: a JavaScript API call, and native code that is written to perform these tasks

What is Ionic?

Ionic further wraps Cordova by providing an entire AngularJS ecosystem to work in to create cross-platform mobile devices.

It exposes many Cordova plugins as AngularJS services, making it even **easier** to create a fully fleshed out application running on a variety of devices.

Ionic also provides users with view-related plugins that allow for easily creating a more native-feeling application, such as having slideout menus and such.