VueJS

Introduction

Getting started

- Open the **`vue-intro/start`** folder in Brackets and note the following script tag in **`index.html`** to link up the Vue library:

```
<script src="js/vue.min.js"></script>
```

- That script tag gives us access to Vue, so we are ready to start creating a dynamic page.

Getting started

- Add the following to the body (typing—rather than copying & pasting—is a better way to practice & learn):

```
<div id="app">
        <p>{{ message }}</p>
</div>
```

- There are two important parts here. The ID is what will define the **`app`** in the code.

- The **`message`** is a variable that we can update through code, note it's inside of two curly braces!

Getting started

- Before the closing body add the following inside the **`script`** tags at the bottom of the **`body`**:

```
new Vue({
      el: '#app',
      data: {
            message: 'My First Vue Page'
     }
});
```

- This defines the **`app`** div as the bit that Vue is going to take control of.
- The **`message`** is a variable that is stored in the data of this app.
- Save your page as **`index.html`** and test it

Data Binding

- What you have just done is known as data binding.
- This is one way data binding, but it can be two way, let’s explore that.
- Add this into the “app” div:

```
<input type="text" v-model="message" />
```

Two way data binding

- *Save the page* and test this in the browser.
- Type into the input box to change the title.
- This is just a simple demonstration of the power of Vue.
- Without having to set up any complex event listeners, the page is updated on the fly!

Using it to bind CSS

- Add the following CSS to head section:

```
<style>
    .red {
         background-color: red;
    }
</style>
```

- Now we will dynamically apply this from the input text box

Using it to bind CSS

- Update the message with the red bit of code below:

```
<p v-bind:class="message">
{{ message }}</p>
```

- This binds the message to also control the CSS class. We only have one class called red.
- Save the page and type ‘red’ into the input box.
- This isn’t that useful but it will be the basis of things that we do later on in the term.
- Before we go on, create a different css class and then try applying it via the input.

Binding an input slider

- Save the page and then choose **`File > Save As`**, naming it 'slider.html'

- Remove everything inside the <div id="app">, replace as follows:

```
<input type="range" min="0" max="360" step="5"
value="180" v-model="myValue" />
<p>{{ myValue }}</p>
```

- This creates an input slider that moves from 0 to 360 and moves up by 5. The starting value will be 180.

- The value of the slider is called **`myValue`** this will be displayed in the paragraph.

Binding an input slider

- Change the vue code as shown:

```
new Vue({
      el: '#app',
      data: {
            myValue: 180
      }
});
```

- Save the page and drag the slider up and down.

- It updates the text but wouldn't it be good to use this number to control an image?

Binding an input slider

- Add this to the body under the paragraph:

```
<img src="http://fillmurray.com/300/400"
v-bind:style="[ { filter: 'hue-rotate('
+ myValue + 'deg)' } ]”>
```

- This displays an image on the screen and there’s the CSS style being bound to Vue.

- The filter: hue-rotate CSS is controlled by the **myValue** variable.

Conclusion

- Compare this to the code from the start of the term when we created the same slider with pure JS
- Make sure you understand what data binding is, we will do it again next week