School of Computer Science

Web and Database Computing 2019

Lecture 18: Introduction to Client-side JS Frameworks

adelaide.edu.au

seek LIGHT

What are Client Side Frameworks?

(and why you want to use them)

JavaScript is great ...

- Allows us to have dynamic content on client side
  - Interactive page elements
  - Respond to events e.g. mouse over menu, clicks, etc.
  - Content changes without reloading page
  - Content changes based on user, browser, location etc.

... but writing DOM code can be a pain.

```
// Store results here
var values = [];

// Get all checkboxes
var checkboxes = document.getElementsByClassName('box');

// Check if ticked and if so add value to array
for(var i=0; i<checkboxes.length; i++){
      if(checkboxes[i].checked){
            value.push(checkboxes[i].value);
            break;
      }
}
```

Getting the values of checkboxes shouldn't be this hard!

There's gotta be a better way!

jQuery; the original solution

```
// Store results here
var values = [];

// jQuery
$('.box:checked').each(function() { value.push($( this ).val()); });
```

- A Javascript library, designed to simplify repetetive and common Javascript tasks.
- Full of shortcuts to easily access and manipulate the DOM tree and make iteratative tasks easier.
- Fixed browser incompatibilities.
- Based on CSS selectors, similar to `document.querySelector();`

Simply load the library as a script:

```
<script src="https://code.jquery.com/jquery-3.4.0.min.js"></script>
```

jQuery; the original solution

```
$( 'css selector' )                                  // Select elements

$( 'css selector' ).hide();                          // Hide/Show elements
$( 'css selector' ).show();                          // (display style)

$( 'css selector' ).text();                          // Get/set text/html content
$( 'css selector' ).html();

$( 'css selector' ).css('property');                 // Get CSS content of element,
$( 'css selector' ).css('property','value');         // set for multiple

$( 'css selector' ).width();                         // Change common CSS properties
$( 'css selector' ).width('10px');
$( 'css selector' ).outerWidth('10px');

$( '<div><p>some html</p></div>' );                  // Create new elements

$( 'css selector' ).append( '<p>html</p>' );         // Modify the DOM tree
```

What could we do better?

- Minimise directly changing DOM
    - Keep page elements that we want to change accessible as JavaScript objects.
    - Treat complex components composed of multiple HTML elements as a single object that can be easily manipulated.
    - Change parts of components by updating the properties of the object.
- Store page data as state information
    - Update page elements automatically when state changes.
- Use templates/placeholders to improve code reusabiility.

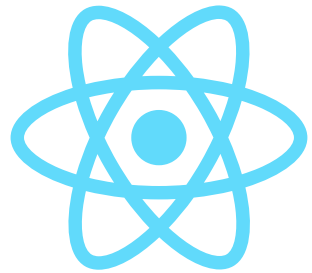
Introducing Client-side JS Frameworks

Current Client-side JS Frameworks

Angular

- Developed by Google
- Original Front-end Framework
- Google, Micorsoft, PayPal, The Guardian, Nike, HBO, Sony
  Source: https://www.madewithangular.com/
- Complex to learn and use (but has gotten better), heavy (144K)

React



- Developed by Facebook
- Currently most popular framework
- Used by Facebook, Airbnb, Dropbox, Netflix, Reddit
  Source: https://madewithreact.com/
- Easier to use but still a challenge to learn, medium (117K)

Current Client-side JS Frameworks

Vue.js

- Open Source
- Gaining popularity
- Used by Baidu, Tencent,Xiaomi, DJI, Nintendo, Sainsbury's
  Source: https://madewithvuejs.com/
- Easy to learn/use & lightweight (88K)

We will be using Vue

Getting started with Vue.js

You can follow along in the lecture slides,
but also following the guide at https://vuejs.org/v2/guide/

Adding Vue to our website

Add a script tag in our document head:

Store the .js file on our site:

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

Or use a CDN (recommended for performance)

```
<script src="https://cdn.jsdelivr.net/npm/vue"></script>
```

Basic Templating

Vue HTML Result Edit in JSFiddle

```
var appdiv = new Vue({
  el: "#app",
  data: {
    text: "Hello"
  }
});
```

https://jsfiddle.net/ian_knight_uofa/3289b16w/3/

Basic Templating; What's happening?

- Use placeholders in our HTML
  - Represented using 'moustaches'

```
{{ placeholder }}
```

- Create a Vue instance
  - Use CSS selector to choose the element to apply the Vue instance to.

```
new Vue({ el: "selector", ... });
```

  - Name the data that will replace the placeholders

```
data: { placeholder:"value" }
```

Vue HTML Result Edit in JSFiddle

```
var appdiv = new Vue({
  el: "#app",
  data: {
    text: "Hello"
  }
});
```

Basic Templating; What's happening?

- Modify the data properties of the Vue instance to automatically update the page.

```
var appdiv = new Vue( ... );
appdiv.text = 'Hi';
```

Vue HTML Result Edit in JSFiddle

```
var appdiv = new Vue({
  el: "#app",
  data: {
    text: "Hello"
  }
});
```

Why Bother?

This all may seem unnecessarily complex, so why bother?

- We're just getting started
- Consider the following example:

Vue HTML Result Edit in JSFiddle

```
var appdiv = new Vue({
  el: "#app",
  data: {
    text: "Hello"
  }
});
```

https://jsfiddle.net/ian_knight_uofa/3289b16w/3/

Templating with objects

- Data properties can be objects as well:

```
<p id="para">{{ obj1.prop1 }} and {{ obj1.prop2 }}</p>
```

```
new Vue({ el: "#para",
          data:  {
                obj1 : { prop1: 'value1',
                         prop2: 2 }
          }
        });
```

Attributes and Style

Vue HTML CSS Result Edit in JSFiddle

```
var vcolour = new Vue({
  el: "#app",
  data: {
    text: "red"
  }
});
```

https://jsfiddle.net/ian_knight_uofa/5v1yw3Lr/3/

Attributes and Style; What's happening?

- Moustache notation only works for text.

```
<h1 id="{{ doesnt_work }}">{{ works_fine }}</h1>
```

- For attributes, replace the desired attribute with `v-bind:attribute_name`.

```
<h1 v-bind:id="now_it_works">{{ works_fine }}</h1>
```

Vue HTML CSS Result Edit in JSFiddle

```
var colour = new Vue({
  el: "#app",
  data: {
    text: "red"
  }
});
```

Attributes and Style; What's happening?

- Classes and styles are special.
- We can use a JavaScript object to specify multiple classes

```
<h1 id="classexample" v-bind:class="{ 'bold_headings': bold_class }">Text</h1>
```

```
new Vue({  el: "#classexample",  data: { bold_class: true }});
```

  - Where the class `bold_headings` will be included if data property `bold_class` is true.
- We can do the same with Styles:

```
<h1 id="styleexample" v-bind:class="{ 'font-family': font }">Some text</h1>
```

```
new Vue({  el: "#styleexample",  data: { font: 'sans-serif' }});
```

  - Where the style `font-family` will be given the value of the data property `font`.

Vue HTML CSS Result Edit in JSFiddle

```
var vcolour = new Vue({
  el: "#app",
  data: {
    text: "red"
  }
});
```

Dynamic Data

We can manipulate the same data to present in different ways:

Vue HTML Result Edit in JSFiddle

```
var vm = new Vue({
  el: '#example',
  data: {
    message: 'Hello'
  },
  computed: {
    // a computed getter
    reversedMessage: function () {
      // `this` points to the vm instance
      return this.message.split('').reverse().join('')
    }
  }
});
```

https://jsfiddle.net/ian_knight_uofa/wvszc3pt/4/

Dynamic Data; What's happening?

- The return value of a computed function can be used in place of a regular data property.
- A computed function that references a data property of the Vue instance will be run any time that data property is changed.

```
computed: {
    reversedMessage: function () {
        console.log('boop'+this.d2);
        return this.message.split('').reverse().joi
    }
}
```

- Modifying the data properties of the Vue automatically runs the function, updating the computed properties.

```
{{ reversedMessage }}
```

Vue HTML Result Edit in JSFiddle

```
var vm = new Vue({
  el: '#example',
  data: {
    message: 'Hello'
  },
  computed: {
    // a computed getter
    reversedMessage: function () {
      // `this` points to the vm instance
      return this.message.split('').reverse().
    }
  }
});
```


THE UNIVERSITY
of ADELAIDE

What's happening

Due:

- Prac Exercise 5 due Fri.
- Prac Exercise 6 due Monday week.

This week:

- More Vue.js
- Google Maps API
- Review lecture Friday
  If you have client-side concepts that you want covered, post in the survey.

Further learning:

- Review Vue.js Guide