CSF304 – Web Application Development

Session 15: VueJS

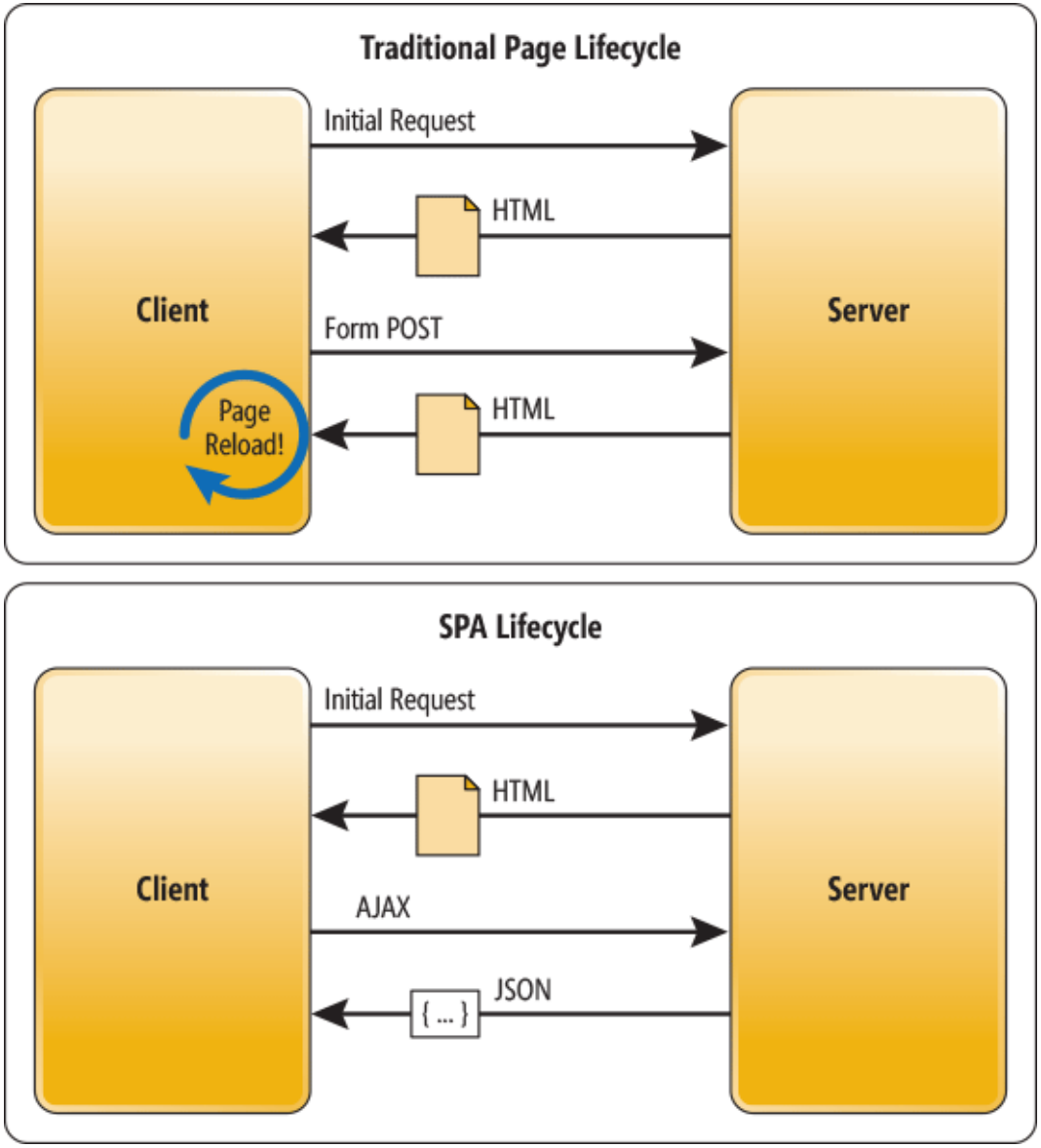
Dr. Liam O'Reilly

Types of Web Apps

I would categorise web apps into (at least 2) categories:

- Traditional style
- Single Page Applications (SPAs)

Traditional Web Apps vs SPA



https://msdn.microsoft.com/en-us/magazine/dn463786.aspx

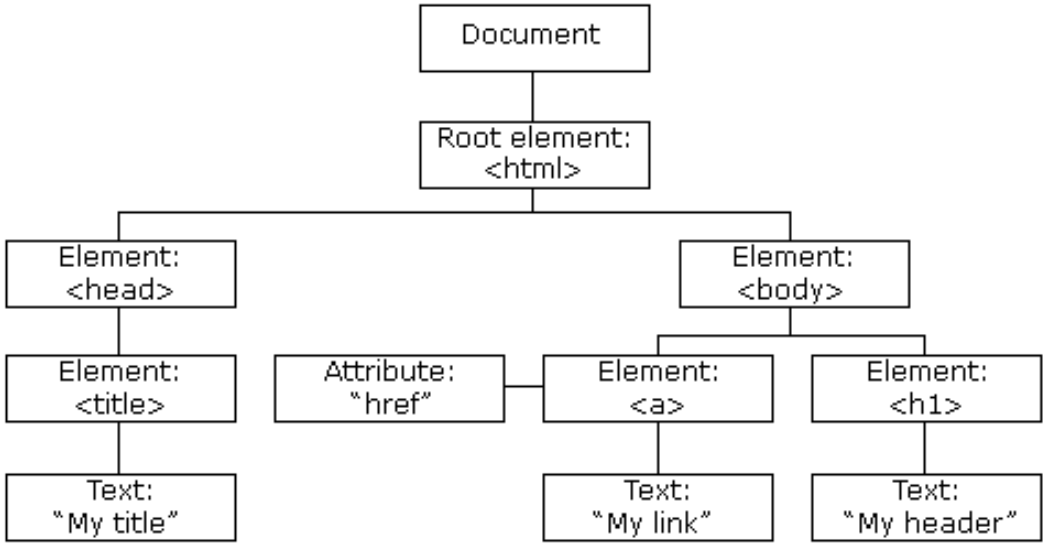
Traditional Web Applications

So far we have looked at “Traditional Web Apps”:

- Get and Post data to URLs.
- Each request brings back a whole webpage.
- Technically, slower than SPAs as you have to send the whole page each time.

HTML DOM

- The HTML DOM (Document Object Model) is a tree of objects that the HTML represents.



- JavaScript can change the DOM.
  - Allows interactive pages.

A Bit of History

- JavaScript:
  - Introduced in December 1995.
  - Scripting language used within HTML.
  - Run on client web browser.
  - Different ways to change the HTML depending on browser.

- jQuery:
  - Lightweight, "write less, do more", JavaScript library.
  - Introduced August 2006.
  - As of June 2018, jQuery is used on 73% of the top 1 million websites, and by 22.4% of all websites (https://en.wikipedia.org/wiki/JQuery)
  - Unified mechanism to manipulate the DOM.

A Bit of History

- Over the last decade many fancy front-end JavaScript frameworks have popped up:
  - Vue.js
  - React.js
  - Angular.js 2
  - …
- These give you a structure to program your front end JavaScript in.

Vue.js

- Vue.js is a progressive JavaScript framework:
  - https://vuejs.org/
  - Progressive means that you can make only parts of your website in Vue.js.
    - Can create website with pieces of Vue.js
    - Can create whole SPAs.
  - See Why Vue.js Video on website above.

- Generally, this is a nice framework.
  - It is data-centric.
  - You specify data and the DOM updates accordingly.
    - In jQuery you manually update the DOM.

Learning Vue.js

- We will learn Vue.js though small examples.
- We will only do the very basics.
  - This could be a whole course in its own right.

- In the next lecture we will integrate it with Laravel.
  - But for now, we will use stand alone web pages (without a web server).

Vue.js Example 1

Vue.js Example 1

Normal HTML5 page.

We load Vue.js from a content delivery network. This is okay for testing and development.

We give an id to an element. This will be used as our main Vue.js container.

We add Javascript which does the following:

- Creates a new instance of VueJs.
- Binds it to the element with the id ‘root’.
- Specifies Vue’s initial data.

```
<!DOCTYPE html>
<html>
<head>
    <title>Vue.js Exmaple</title>
</head>

<body>
    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>

    <div id="root">
        <p>Hello. The message of the day is: {{ message }}</p>
        <p>Hello. The message of the day is: <b>{{ message }}</b></p>
    </div>

    <script>
        var app = new Vue({
            el: "#root",
            data: {
                message: "Vue.js is cool!"
            }
        });
    </script>
</body>
</html>
```

Use {{ … }} to echo out the value of messages

Vue.js Example 1 (Cont.)

- This gives us the web page:

Hello. The message of the day is: Vue.js is cool!

Hello. The message of the day is: **Vue.js is cool!**

- If we change the value of message using the chrome console:

```
> app.message = "Super Cool"
<- "Super Cool"
>
```

- Then the web page updates:

Hello. The message of the day is: Super Cool

Hello. The message of the day is: **Super Cool**

This is what is meant when we say Vue.js is reactive.

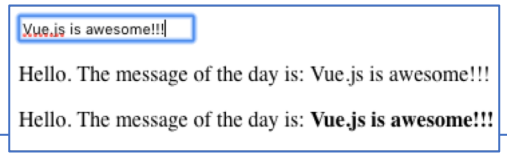
Vue.js Example 2

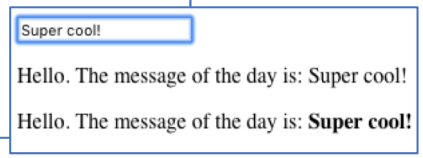
Vue.js Example 2

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

    <div id="root">
        <input type="text" id="input" v-model="message">
        <p>Hello. The message of the day is: {{ message }}</p>
        <p>Hello. The message of the day is: <b>{{ message }}</b></p>
    </div>

    <script>
        var app = new Vue({
            el: "#root",
            data: {
                message: "Vue.js is cool!"
            }
        });
    </script>
</body>
```

The v-model directive is used to create a two-way binding. When the input box's value is changed, then the value of 'message' is updated too. Also vice versa.

The {{ … }} create a one-way binding. When the value of message changes, then the value of {{ … }} will change,





Vue.js Example 3

We Can Also Have Methods

- We can create methods in Vue.js that have access to Vue's data.
- We can bind the methods to events – such as when a button is clicked.

Vue.js Example 3

Create a button that when clicked calls the method reverse.

Create a method named reverse.

The method takes the message string, splits it up to an array of characters, reverses the array, joins the array back together and then stores it back into the message variable.

```
<div id="root">
    <p>{{ message }}</p>
    <button @click="reverse">Reverse Message</button>
</div>
<script>
    var app = new Vue({
        el: "#root",
        data: {
            message: "Why didn't we learn this first?"
        },
        methods: {
            reverse() {
                this.message = this.message.split('').reverse().join('');
            },
        }
    });
</script>
```

Vue.js Example 4

Vue.js Variables Can Hold Structured Data

- Vue.js Variables Can Hold Structured Data such as lists (and much more).
- We can also loop over the lists in the HTML and display them.

Vue.js Example 4

Loop over each item in the list and display it as a list.

```
<div id="root">
    <ul>
        <li v-for="name in names">{{ name }}</li>
    </ul>
</div>

<script>
    var app = new Vue({
        el: "#root",
        data: {
            names: ["Liam", "Phil", "Geoff", "Faron"]
        },
    });
</script>
```

Our data is now a list.

- Liam
- Phil
- Geoff
- Faron

If we use the console to add to the list then the web page (DOM) is updated too.

```
> app.names.push("Bertie")
<- 5
```

- Liam
- Phil
- Geoff
- Faron
- Bertie

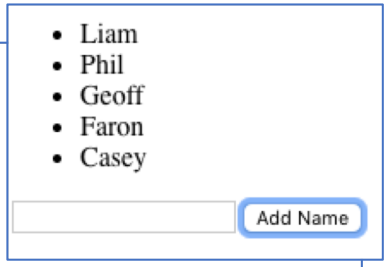
Vue.js Example 5

Vue.js Example 5: Interactive Web Pages

We now have the basics for an interactive web page.

```
<div id="root">
    <ul>
        <li v-for="name in names">{{ name}}</li>
    </ul>

    <input type="text" v-model="newName">
    <button @click="addName"> Add Name</button>
</div>
```

```
<script>
    var app = new Vue({

        el: "#root",

        data: {
            newName: "",
            names: ["Liam", "Phil", "Geoff", "Faron"]
        },

        methods: {
            addName() {
                this.names.push(this.newName);
                this.newName = "";
            }
        }
    });
</script>
```



Create an input field with a two-way binding to newName.

Create a button which when clicked calls the method.

We can declare methods. This adds a new name to the list.

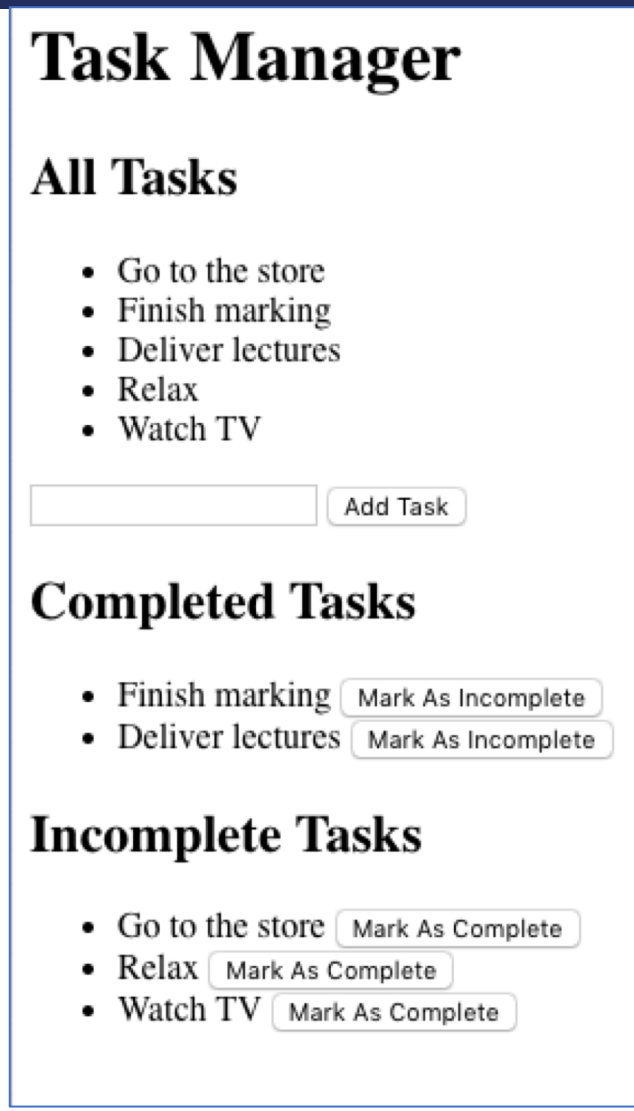
- It takes the value of newname variable, adds it to the list and then clears the variable.

Vue.js Example 6

Vue.js Example 6 – Task Manager

- Example 6 is too large to look at on the slides.
  - Take a look at the code on Blackboard.

- Things to note in the example:
  - Computed properties are values that can be read that are computed based on other values. If the other values change, so do the computed properties.
  - Our data is a list of objects (each with a description and a completed flag).
  - We use steam filtering to separate the completed and incomplete tasks.



Summary

- VueJs is a powerful Java Framework.
  - It is currently popular.
    - But the web technology moves fast! Very fast!
  - It is reactive – you do not manipulate the DOM directly.
- We have only touched on VueJS.
  - Really we have.
- We have looked only at front end in this lecture.
  - Next lecture we combine this with Laravel.