CS1320
Creating Modern Web and Mobile Applications
Lecture 10:
Front End Frameworks
Frameworks

- **Web applications are a lot of work**
  - Much of it is busy work

- **Routine chores and programming**
  - Administration
  - URL -> command mapping
  - Preparing HTML pages from data
  - Data structure mapping

- **Frameworks try to simplify this**
  - And a new one is developed every week
Frameworks

• Front End Frameworks
  o Provide common facilities in the front end
  o Simplify coding the front end
  o VUE, React, Angular

• Back End Frameworks
  o Provide common facilities in the back end
  o Express, Flask, Django, …

• Complete Frameworks
  o Provide integrated facilities for both the front and back end
  o Content management systems

• Mobile Frameworks
  o Provide common facilities for mobile front ends
Frameworks

• Most frameworks are based on a Model-View-Controller architecture

• Separation of Concerns
  ○ Presentation and contents: html and css
  ○ Data and display: MVC
  ○ Structure and application specifics
Model-View-Controller
Model-View-Controller

• Basic idea is to separate the display, the data, and the logic
  ○ Each can be change independent of the others

• Exactly how this is done various from case to case
  ○ Some do it with a common data abstraction
  ○ Some do it with callbacks
  ○ Some automate the controller: MVVM
  ○ All call themselves MVC

• Different people mean different thing
Front End Frameworks

• What are the problems with JavaScript in the front end
  ○ Need to modify the DOM as the data changes
  ○ Need to create new HTML/CSS as the data changes
  ○ Need to change the data as the HTML changes (user inputs)

• Where is the messy, repetitive work
  ○ Creating the same HTML over and over for new data
  ○ Updating the HTML as the data changes

• How can this be simplified
Templating or Scripting

- The content of a dynamic web page will depend on data
  - The results of a search, the set of relevant items
  - How something is displayed may depend on its value (bar graph)
- Want to generate (and modify) a page based on computed values
  - Values might be computed in the front end
  - Values might be passed from the back end to the front end
  - Values might be computed in the back end
- Templating does a lot of this for you
Back End Templating

• The back end needs to generate HTML files
  ○ These often depend on data

• Templates are html files that are expanded based on data
  ○ Can use data to replace values
  ○ Can iterate over data to generate html for each instance
  ○ Can generate different html based on the data values

• Very convenient way of generating complex data-based pages

• We’ll get back to this as we cover the back end
  ○ Moustache (handlebars)
Front End Templating

• We can do the same thing in the front end
  ○ Values might change based on user input
  ○ New or changed values might be obtained from the back end
  ○ In either case we want to update the current page (DOM) accordingly

• Different approaches to this
  ○ REACT - programming model that generates HTML
  ○ Angular - HTML with inserts that are replaced & templating constructs
  ○ VUE - HTML with inserts that are replaced & templating constructs
VUE

• HTML-centric framework
• Simple approach to inserting data values into HTML dynamically
• MVVM model to handle automatic updates to the DOM as values change
• Notion of components to allow easier creation of complex HTML
  ○ Component is a DOM tree structure that can be filled in
  ○ Language for creating these (requires a preprocessor)
• Extensions for animation, mixins, routing, ...
  ○ It can do a lot, but basic VUE is generally good enough
MVVM (Model-View, View-Model)
VUE Example

• vueex.html demo
VUE Basics

• Write the HTML, have VUE fill in values
  ▪ `<div id="app"> {{ message }} </div>`
  ▪ `{{ <expression> }}` is replaced by the value `<expression>`
  ▪ Where `<expression>` is computed against a particular context

• VUE works in the front end
  ○ `let app = new Vue( {`;
    ▪ `el : '#app',`
    ▪ `data: { message: 'Hello World' }`
    ▪ `methods: { ... } ... };`
  ○ EL - selector of the element this applies to
  ○ DATA - context for evaluating expressions
  ○ METHODS - methods that can be used in the HTML

• Can access and set the data
  ○ `app.message = 'Good Night'`
  ○ Will change the DOM automatically
VUE Binding

• `{{ expr }}` is the simplest binding
• Can also bind using v-bind prefix
  o `<span v-bind:title='message'> ... </span>`
  o Use instead of `title='{{message}}'`
  o Short cut :title

• v-bind is a VUE directive
  o VUE directives start with v-
  o Other VUE directives provide additional facilities
VUE Conditionals

• Can control HTML based on values
  ○ `<div id='app'>`
    ▪ `<span v-if='seen'>This HTML only exists if seen is true</span>`
    ▪ `<span v-else>This HTML only exists if seen is false</span>`
    ▪ `</div>`
  ○ let app = new Vue( { el: '#app', data: { seen : true } } );

• Can change app.seen
  ○ Causes the HTML to change accordingly
VUE Loops

• Can create HTML for each element of a collection
  o `<div id='app'><ol><li v-for='todo in todos'>{{ todo.text }}</li></ol></div>`
  o let app = new Vue( { el: ‘app’,
    ■ data: { todos : [ “Lab 1”, “Prelab 2”, “Assignment 1”, “Project Meeting” ] } } )

• This will generate a li element for each element of the todos array

• Changing the todos array will update the HTML accordingly
  o Add a new element, remove an element
VUE Events

- Can tie HTML actions to methods on the VUE object
- `<div id='app'><p>{{msg}}</p> <button v-on:click='reverse'>Reverse</button> </div>`
  - Short cut: @click
  - v-on:event.modifier
- `let app=new Vue({el: '#app',
  data: { msg: 'Hello World' },
  methods: { reverse: function() {
    this.msg = this.msg.split('').reverse().join('')
  } }
})`

- v-on directive ties click to the appropriate method in the VUE object
VUE Model

• Can also tie input fields to VUE variables directly
• `<div id='app'><p>{{msg}}</p><input v-model='msg'></div>`
  - The input field maps to and from the variable msg
• `let app = new Vue( { el: '#app', data: { } });`
  - Typing in the input field changes msg
  - Changing msg changes the displayed paragraph
VUE Components

• Can create HTML macros with encapsulated methods

• Vue.component(‘name’, {
  o props: [ list of bound variables ]
  o methods: { … }, data: { … }
  o template: <html as a string> }

• Use this as <name></name>

• Can be combined
  o <name v-for=’t in array’ v-bind:var=’t’ v-bind:key=’t.id’></name>

• Doesn’t work in all contexts (e.g. tables,…)
  o <tr is=’name’ …></tr>
VUE Example

• vuesimple.html as the design
• vueex.html as the result - look at the code
• vueex1.html as the result with components - look at the code
REACT

• A JavaScript-centric front end framework
• You write JavaScript and embed the HTML into it
• Requires a preprocessor
  ○ Generally run in the back end
  ○ Generates both HTML and JavaScript
Creating Dynamic HTML Pages

• We now have multiple approaches
  o Doing most of the work in the back end (generate templated page)
    ▪ Once we have a back end
  o Doing most of the work in the front end (generate page based on data)
    ▪ Single page application

• All require some control of the HTML from JavaScript
  o Can use JavaScript directly
    ▪ Best for simple things - hide and show elements, changing classes, styles
  o Using jQuery
    ▪ Useful if you are doing this a lot or there are multiple elements
  o Using VUE
    ▪ Useful if the changes are complex and data dependent
What to Use

• Will depend on the overall application
  ○ We will get back to this once we have a better understanding of
    ▪ The back end
    ▪ Communication between the front and back end
    ▪ What types of interaction to support
  ○ Discussing Web Application Architectures

• For now
  ○ Become familiar with the basics (JavaScript or jQuery)
  ○ Become familiar with front end templating (VUE)
  ○ Understand their capabilities and limits so you can make a wise choice later
Next Time

- Lab 3: Front End Frameworks
- Homework:
  - PreLab 3: to familiarize yourself with JavaScript
Model-View-Controller

Data Display

Data Management

Data Store

Web Browser

JavaScript

Back End