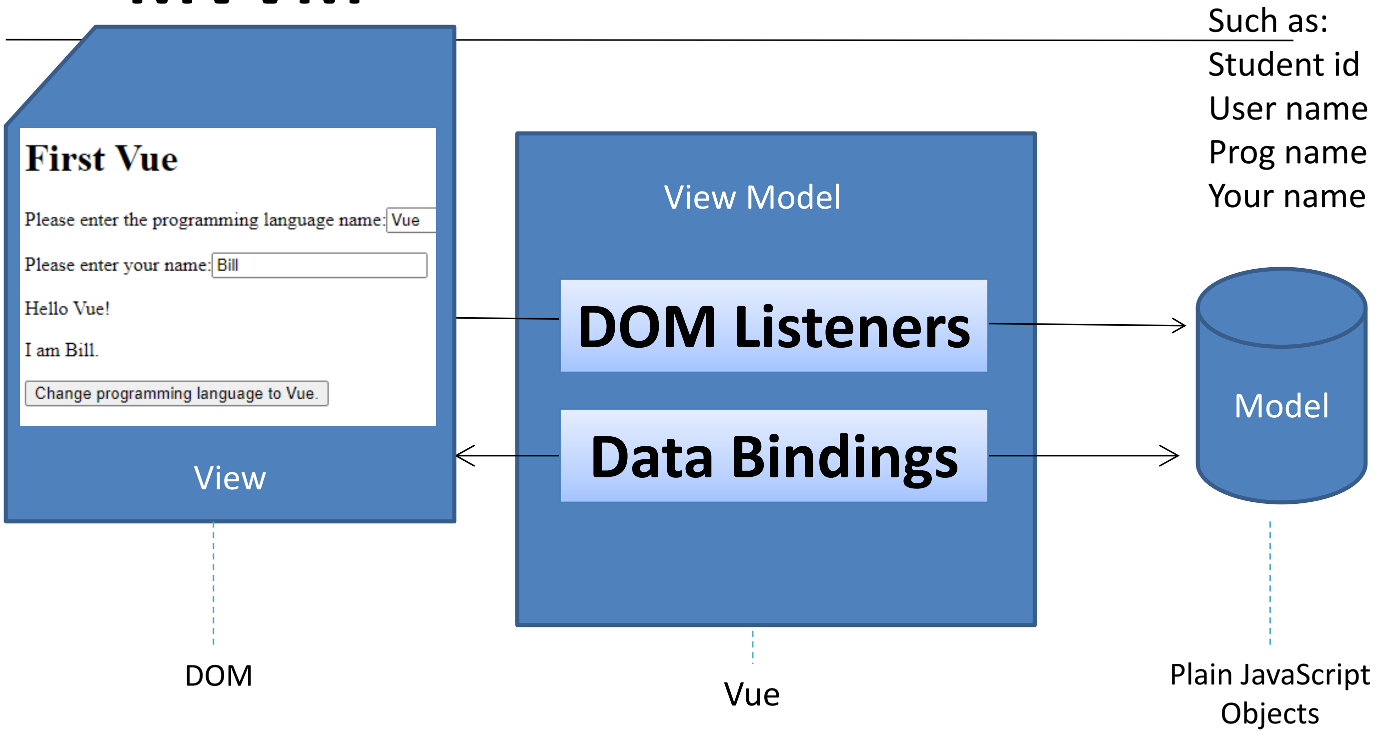
COS30043 Interface Design and Development

Lecture 3 – VueJS Data Binding and Directives

2022 – Semester 1

Contents

MVVM



First Vue

Please enter the programming language name: Vue

Please enter your name: Bill

Hello Vue!

I am Bill.

Change programming language to Vue.

Model-View-View Model (MVVM)

- an architectural pattern that separates an application into three main logical components: the model, the view and the view model
- one of the most frequently used industry-standard frontend web development framework to create scalable and extensible projects

Model Component

- An object representing the
  - data related logic that the user works with
  - data that is being transferred between the View and Controller components
- For example, a Student object will retrieve the student information from the database, edit it and update it back to the database

View Component

- all the user interface
- For example, the Student view would include UI components such as text boxes, dropdowns that user interacts with

View Model Component

- interfaces between Model and View components to process business logic and requests, edit data using the Model component and interact with the View to generate the output. The view binds to the view model to send and receive information from the model.
- For example, the Student View Model handles (input) all the interactions from the Student View, (process) update the database using the Student Model, and (output) interact to generate the view of the Student data

Contents

VueJS

- Vue is a JavaScript framework for building user interfaces.
- a fully client-side framework
- has templating based on bidirectional UI data binding, where the HTML template is compiled in the browser
  - Compilation step creates pure HTML and the browser regenerates it into the view. This is continuously repeated for subsequent page views.
- controller and model **state** are maintained within the client browser, thus new pages are generated without any interaction with a server.

Single-page Application (SPA)

- a web application or web site that is contained in a single web page to provide user experience similar to desktop applications
- all code, HTML, CSS and JavaScript, is retrieved on initial page load, and appropriate resources are dynamically loaded as necessary based on response to user actions
- the entire page does not reload nor control transfer to another page
- interaction involves dynamic communication with the web server in the background

Challenges with the SPA model

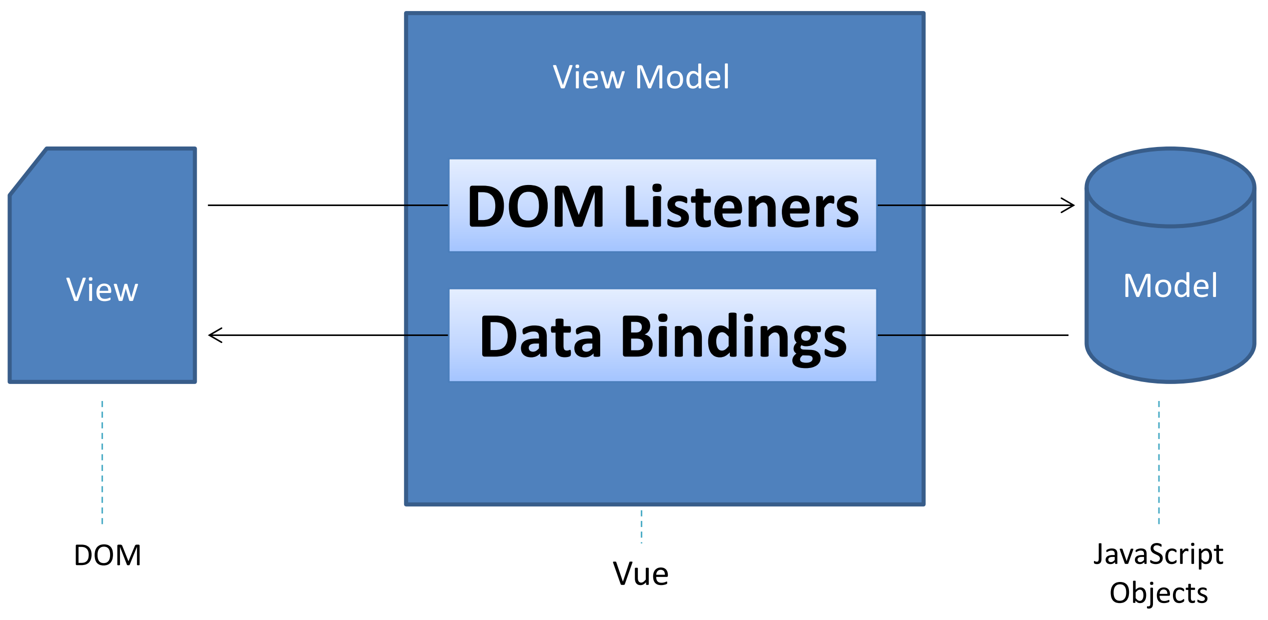
- Search engine optimization
  - where are the codes
- Client/Server code partitioning
  - where to place the logic code
- Browser history
  - what happens to the back button
- Analytics
  - no new page load to trigger analytics
- Page Load
  - long initial load time

Contents

Data Binding

- an automatic way of updating the view whenever the model changes and vice versa
- templates, HTML along with any additional markup or directives, are compiled on the browser that generates a live view
  - Any changes to the view are immediately reflected in the model, and any changes in the model are propagated to the view.
  - view becomes an instant projection of your model.

Data Binding – Two Way



Contents

DIRECTIVES (MODEL – DATA/VARIABLES)

v-bind
v-model
v-once

Directives

- Directives are instruction for VueJS to do things in a certain way.
- Markers on a DOM element that tell VueJS's **HTML compiler** to
  - attach a specified behaviour to that DOM element (e.g. via event listeners); or
  - transform the DOM element and its children
- custom and reusable HTML-like elements and attributes starting with v-

`The directives mentioned in this presentation are some of the many directives available in Vue. We can also create custom directives.`

Directives (Expression and Operators)

- Assignment =
  `varA = 3; varB = 2;` `returns 2`
- Arithmetic + , - , *, /, %
  `varA * varB` `returns 6`
- Comparison <, <=, >, >=, ==, !=, ===, !==
  `varA > varB` `returns true`
- Logical &&, ||
  `(varA > 1) && (varA < 5)` `returns true`
- Inline if:
  `(varA == 1 ? 10 : 20)` `returns 20`
- Concatenation +
  `('A' + 'B')` `returns 'AB'`

Directives (Model – Data/Variables)

- Array

```
students = ['Amy', 'Ben']
students[0]
```

- Object array

```
studentObjs = [{name:'Amy',age:24},
               {name:'Ben',age:23}]
studentObjs[0].name
studentObjs[0].age
```

Directives (Model – Data/Variables)

- String, numeric and boolean variable

```
varA = 'string'
varB = 2
varC = true
```

- Object variable

```
unit.code = 'COS30043'
unit.desc = 'IDD'
```

Note: ' is often used as " is used to enclose expressions in the HTML code

Directives (Model – Data/Variables)

- v-bind: Binds one or more attributes dynamically, or a component prop to an expression

In html:

```
<div id="app">
      <a v-bind:href ="someUrl">website</a>
      <img v-bind:src="imageSrc">
</div>
```

In JavaScript:

```
Vue.createApp({
      data() {return {
            someUrl: "https://swin.edu.au",
            imageSrc: "flower.jpg",
            }
      }
}).mount('#app');
```

Directives (Model – Data/Variables)

- v-model

  creates a two-way binding on a form input element or a component. The form input elements include <input>,<select> and <textarea>.

In html:

```
<div id="app">
      <input v-model= "message" >
      <p>Message is: {{ message }}</p>
</div>
```

In JavaScript:

```
Vue.createApp({
      data() {return {
            message: "Hello Vue!",
            }
      }
}).mount('#app');
```

Directives (Model – Data/Variables)

- v-once

It is used to render the element and component once. On re-rendering, the element/component and its children are treated as static content and skipped.

```
<ul>
<li v-for="i in list" v-once>{{i}}</li>
</ul>
```

DIRECTIVES (VIEW – CONDITIONAL)

v-if
v-else & v-else-if
v-show

KNOW ING

Directives (v – if)

- v-If: removes or recreates a portion of the DOM tree based on a Boolean expression

In html:

```
<div id="app">
      <div v-if="myVar == 'dogs'">
            <h3>Dogs</h3>
            <p>Welcome to a world of dogs.</p>
      </div>
</div>
```

In JavaScript:

```
Vue.createApp({
      data() {return {
            myVar: "dogs",
            }
      }
}).mount('#app');
```

Directives (v– else & v-else-if)

v-else indicates an else block for v-if

```
<div v-if="Math.random() > 0.5"> Congratulations!
</div>
<div v-else> Better luck next time </div>
```

v-else-if indicates an else-if block for v-if. It can be included multiple times

```
<div v-if="myVar == 'dogs'">
        <h3>Dogs</h3>
        <p>Welcome to a world of dogs.</p>
</div>
<div v-else-if="myVar == 'tuts'">
        <h3>Tutorials</h3>
        <p>Learn from examples.</p>
</div>
<div v-else>
        <p>Select topic from the dropdown.</p>
</div>
```

Directives – if versus else-if

- v-if conditionally renders an element based on the truthy-ness of the value of the expression. It expects an expression to render the element
- v-else does not expect an expression. It must be preceded by v-if or v-else-if.
- v-else-if expects an expression and must be preceded by v-if or v-else-if

Directives (v – show)

- v-show

It conditionally displays an element. The element is rendered and remains in the DOM. v-show only toggles the *display* CSS property of the element

```
      <h1 v-show="ok">Hello Mr. Chua!</h1>
```

v-if: creates or removes an element based on the condition. If the condition is false, the element is not rendered (doesn’t exist in the DOM).

v-show: the element is rendered and remains in the DOM, only show or hide the element (using css) based on the condition.

DIRECTIVES (VIEW – LOOP)

v-for

Directives (View – Loop Array)

- v-for: This directive renders a list of items based on an array

  In html:
```
<div id="app">
    <ul>
         <li v-for="s in students" >
              {{s}}
         </li>
    </ul>
</div>
```
  In JavaScript:
```
Vue.createApp({
     data() {
          return {
               students: ["Amy", "Bill]
          }
     }
}).mount('#app');
```

DIRECTIVES (VIEW – FUNCTION-LIKE)

v-on

Directives

- v-on

  Executes Vue expression on clicks

  – Can be implemented using v-on:click or @click

```
<div class="col-md-4">
    <h2>Click Events</h2>
    <input type="button" value="Click for your
lucky number" v-on:click="num = 7"> {{num}}

    <input type="button" value="Click for your
random number" @click="num2= num2+num"> {{num2}}
</div>
```

DIRECTIVES (ADDITIONAL SUPPORT)

transition

Directives (Additional Support)

Vue provides a transition wrapper component, allowing you to add entering/leaving transitions for any element or component in the following contexts:

- Conditional rendering (using v-if)
- Conditional display (using v-show)
- Dynamic components
- Component root nodes

Directives (Additional Support)

```
Example:
#HTML
  <p><input type="checkbox" id="checked" v-model="checked" /></p>
  <transition name="fade">
      <div v-show="checked">
        <span>Show:</span>
        I show up when your checkbox is checked.
      </div>
  </transition>

#CSS
•             .fade-enter-active,
•             .fade-leave-active {
•               transition: opacity 3s ease;
•             }

•             .fade-enter-from,
•             .fade-leave-to {
•               opacity: 0;
•             }
```

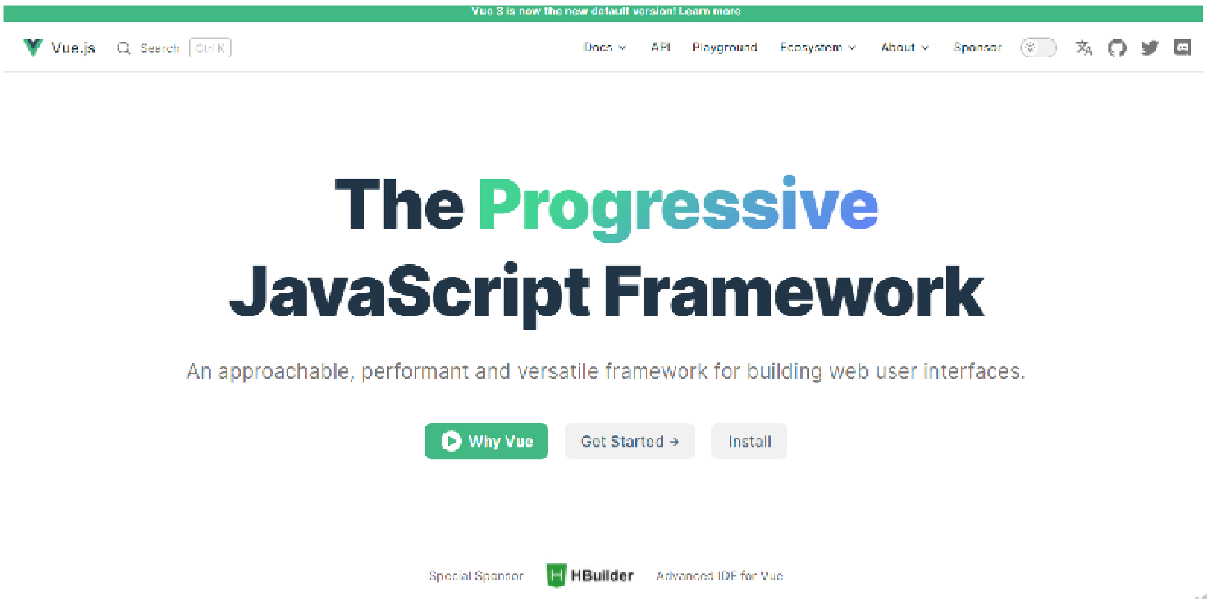
Contents

- Model-View-ViewModel
- What is VueJS
- Data Binding
- Directives
- Install VueJS

GETTING STARTED

Software Installation

- Go to https://vuejs.org/
- Click on Install


Vue 3 is now the new default version! Learn more

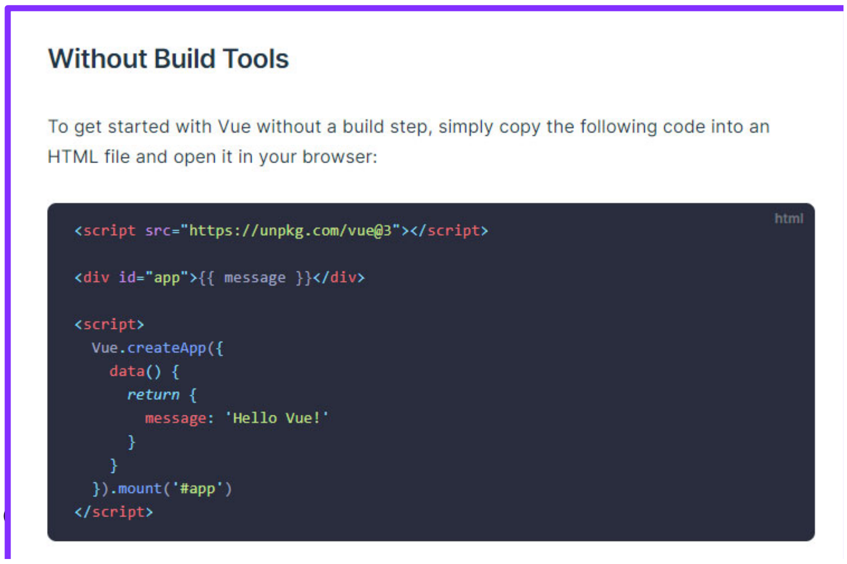
The Progressive JavaScript Framework

An approachable, performant and versatile framework for building web user interfaces.

Software Installation (Link to CDN)

Link to vue js version 3, you can start to use it. For example:

```
<div id="app">{{ message }}</div>
….
<script src="https://unpkg.com/vue@3"></script>
<script>
 Vue.createApp({
  data() {
   return {
    message: 'Hello Vue!'
   }
  }
 }).mount('#app')
</script>
```



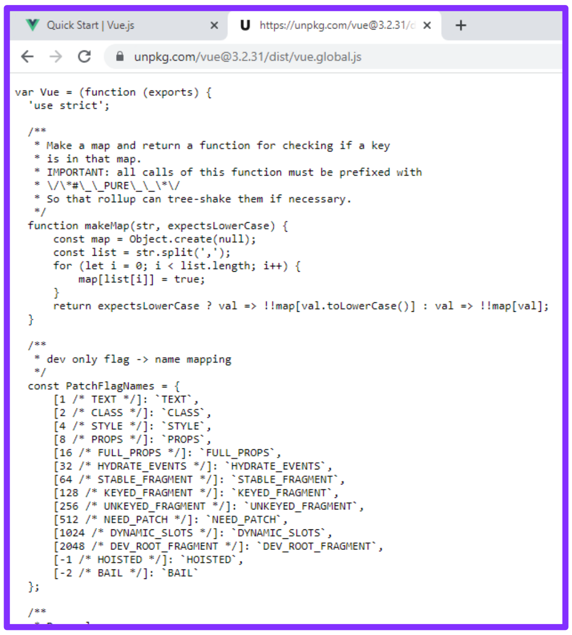
Without Build Tools

To get started with Vue without a build step, simply copy the following code into an HTML file and open it in your browser:

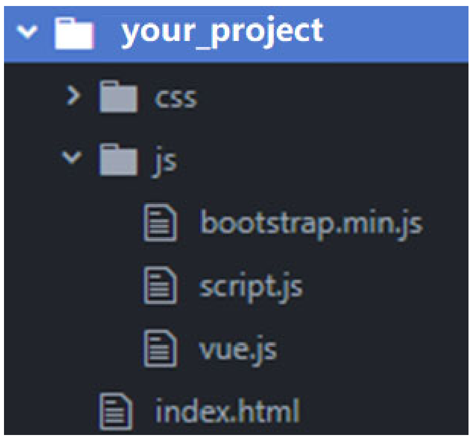
```html
<script src="https://unpkg.com/vue@3"></script>

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

<script>
  Vue.createApp({
    data() {
      return {
        message: 'Hello Vue!'
      }
    }
  }).mount('#app')
</script>
```

Software Installation (Download to computer)

- You can also download vuejs to your own computer. Type https://unpkg.com/vue@3 in a browser, copy and paste the code, save to a file named vue.js in an appropriate folder.

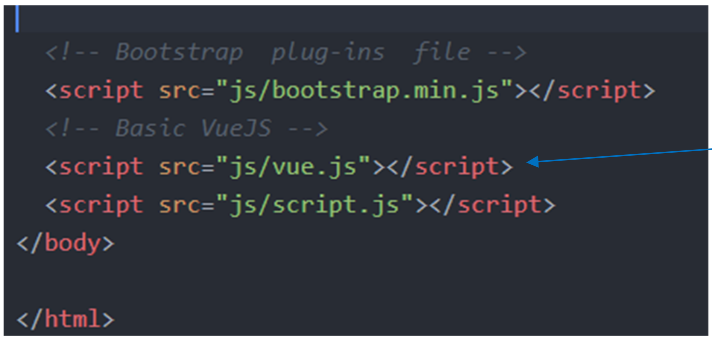

Quick Start | Vue.js
https://unpkg.com/vue@3.2.31/d
unpkg.com/vue@3.2.31/dist/vue.global.js

```
var Vue = (function (exports) {
  'use strict';

  /**
   * Make a map and return a function for checking if a key
   * is in that map.
   * IMPORTANT: all calls of this function must be prefixed with
   * \/\*#\_\_PURE\_\_\*\/
   * So that rollup can tree-shake them if necessary.
   */
  function makeMap(str, expectsLowerCase) {
      const map = Object.create(null);
      const list = str.split(',');
      for (let i = 0; i < list.length; i++) {
          map[list[i]] = true;
      }
      return expectsLowerCase ? val => !!map[val.toLowerCase()] : val => !!map[val];
  }

  /**
   * dev only flag -> name mapping
   */
  const PatchFlagNames = {
      [1 /* TEXT */]: `TEXT`,
      [2 /* CLASS */]: `CLASS`,
      [4 /* STYLE */]: `STYLE`,
      [8 /* PROPS */]: `PROPS`,
      [16 /* FULL_PROPS */]: `FULL_PROPS`,
      [32 /* HYDRATE_EVENTS */]: `HYDRATE_EVENTS`,
      [64 /* STABLE_FRAGMENT */]: `STABLE_FRAGMENT`,
      [128 /* KEYED_FRAGMENT */]: `KEYED_FRAGMENT`,
      [256 /* UNKEYED_FRAGMENT */]: `UNKEYED_FRAGMENT`,
      [512 /* NEED_PATCH */]: `NEED_PATCH`,
      [1024 /* DYNAMIC_SLOTS */]: `DYNAMIC_SLOTS`,
      [2048 /* DEV_ROOT_FRAGMENT */]: `DEV_ROOT_FRAGMENT`,
      [-1 /* HOISTED */]: `HOISTED`,
      [-2 /* BAIL */]: `BAIL`
  };

  /**
```



Software Installation (Download to computer)

- Load the JS file in the HTML file using the script tag.



```
  <!-- Bootstrap  plug-ins  file -->
  <script src="js/bootstrap.min.js"></script>
  <!-- Basic VueJS -->
  <script src="js/vue.js"></script>
  <script src="js/script.js"></script>
</body>

</html>
```

Load the vue.js file

Software Installation (Download to computer)

Folder Contents (Simplified)

```
framework/
├── css/
│    ├── style.css          ← used for custom css
│    ├── bootstrap.min.css
│    ├── bootstrap.min.css.map   ← used for debugging
│    ├── bootstrap-theme.min.css
│    └── bootstrap-theme.min.css.map

├── js/
│    ├── vue.js
│    ├── custom.js  ← used for custom js
│    ├── bootstrap.min.js
│
├── index.html
```

WHAT'S NEXT?
– VIEW AND VIEW MODEL