

# COS30043

## Interface Design and Development



## Lecture 4 – View and ViewModel

2022 – Semester 1



1

### Contents

---



- MVVM
- View
- ViewModel

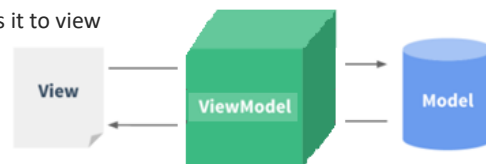


2

# MVVM

MVVM is an architectural pattern that separates an application into three main logical components: the model, the view and the view model

- Model: It holds the data/information of the app which is to be presented to the user for manipulation or interaction.
- View: It is used to render the information to the user.
  - the View doesn't know about the Model and vice-versa.
  - The View passes user input to the VM for processing.
  - The View presents 'state' defined by the VM to user.
- ViewModel: The VM is the link between the Model and the View. It defines all business logics, such as
  - manipulate the data contained in the model
  - handle user interaction
  - Format data in the model and pass it to view

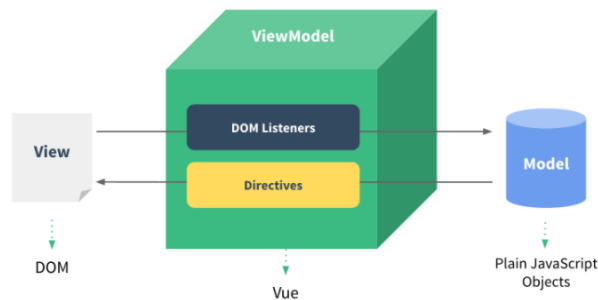


3 - Interface Design and Development, ©

3

# VueJS and MVVM

- Vue.js is a framework for building interactive web interfaces.
- Vue.js is focused on the ViewModel layer of the MVVM pattern. It connects the View and the Model via two way data bindings.



4 - Interface Design and Development, © Swinburne



4

## Contents

---



- MVVM
- View
- ViewModel

5 - Interface Design and Development, © Swinburne



5

## View – Updating Model

---

- View not only shows but also provides user interactive through
  - Input elements
  - v-on:click
- Note that all interactions relate only to updating the values of the model through
  - user Input
  - expression (assignment operator)

6 - Interface Design and Development, © Swinburne



6

## View – Updating View

- Conditional and loop directives updates the view based the value in the model
- Note that the view is only updated if some value in the model changes

- For example,

```
<p>Enter string #1:
  <input type = "text" name = "strVar"
    v-model = "strVar" /></p>
<p>Immediate effect : {{strVar}}</p>
```

- The view change as you update strVar

7 - Interface Design and Development, © Swinburne



7

## View – Design Process

- To update the view, link it to a model
- To update the model, use HTML input element, expression or the Vue instance
- Consider the calendar web app that has 2 display options (Week and Month)

Week	Month
-----	
Mon :	
-----	
Tue :	
-----	

8 - Interface Design and Development, © Swinburne



8

## View – Design Process

---

- Design 2 views
- Create a model to the views e.g. use “sele” to indicate whether month or week is selected
- Link the model to view i.e. v-if, v-else or v-show
- To switch between week and month view, you update the value of “sele”, using
  - Input element, e.g. radio
  - Anchor/Button with v-on:click

9 - Interface Design and Development, © Swinburne



9

## Contents

---

- MVVM
- View
- ViewModel



10 - Interface Design and Development, © Swinburne



10

## ViewModel

---

- Provides the business logic to manage view behaviour
- Prepares/initialises the model for the view
- Responds to user interaction

11 - Interface Design and Development, © Swinburne



11

## ViewModel – Execution

---

- A new instance of the Vue app is created every time it is used
- Prepares/initialises properties and methods
- Mostly assignment/method set operations
  - (Properties) prepare/initialise values to model through ViewModel
  - (Methods) Inject functions to model ( i.e. `data` ), these functions only execute when called/triggered

N.B. **`data`** can be of type `object` or `function`

12 - Interface Design and Development, © Swinburne



12

## View – Template

---

- Link View to ViewModel

```
<html>
:
<body>
  <div id="app">
    {{strVar}}
  </div>
</body>
</html>
```

- In this example `strVar` is accessible by the `myCtrl` function using `this` keyword

13 - Interface Design and Development, © Swinburne



13

## ViewModel – Template

---

- Access Model from ViewModel

```
methods: {
  myCtrl() {
    this.strVar = "new value";
  }
};
```

`myCtrl() {...}`  
is the shorthand for  
`myCtrl: function () { }`

- Multiple functions can be declared inside `methods` object

14 - Interface Design and Development, © Swinburne



14

## ViewModel – Template - HTML

File: ctrldemo.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Template that uses Bootstrap</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width,
                                initial-scale=1.0" />

  <!-- Bootstrap -->
  <link href="css/bootstrap.min.css" rel="stylesheet" />
  <!-- HTML5 Shim and Respond.js IE8 support of HTML5
        elements and media queries -->
  <!-- WARNING: Respond.js doesn't work if you view the
        page via file:// -->
  <!--[if lt IE 9]>
    <script src="js/html5shiv.js"></script>
    <script src="js/respond.min.js"></script>
  <![endif]-->
</head>
```

15 - Interface Design and Development, © Swinburne



15

## ViewModel – Template (continued) - HTML

```
<body >
  <div id="app">
    <p> {{strVar}} </p>
    <button v-on:click="myCtrl">Change</button>
  </div>

  <!-- jQuery - required for Bootstrap plugins) -->
  <script src="js/jquery.min.js"></script>
  <!-- All Bootstrap plug-ins file -->
  <script src="js/bootstrap.min.js"></script>
  <!-- Basic VueJS -->
  <script src="js/vue.min.js"></script>
  <!-- Your Component -->
  <script src="js/app.js"></script>
</body>
</html>
```

16 - Interface Design and Development, © Swinburne



16



## ViewModel – Template (continued) - JS

**File: app.js**

```
Vue.createApp({ //create a new application instance
  data() {
    return {strVar:'Hello World!' }
  },

  methods: {
    myCtrl:function() {
      this.strVar = "Hello New World!";
    }
  }
}).mount('#app')
```

17 - Interface Design and Development, © Swinburne



17

## ViewModel – Methods

- Objects

```
data() {
  return {<object names>}
}
```

- Methods

```
methods: {
  <methodName>:function(<parameters>) {
    <JavaScript program>
    [return <expression>;]
  },
}
```

18 - Interface Design and Development, © Swinburne



18

## ViewModel– Examples (User Method)

- Object (initialisation)

```
data() {  
    return { name: "Dr Caslon Chua.",  
            };  
}
```

- Method (update through method –no parameter)

```
methods:{  
    getName() {  
        return this.name;  
    }  
}
```

**N.B. In this example data is a function, not object**

19 - Interface Design and Development, © Swinburne



19

## ViewModel – Examples (User Method)

- Method (update through method with parameter)

```
methods: {  
    getName(bio) {  
        return (this.name + bio);  
    }  
}
```

- HTML

```
<body id="app">  
    <p> {{ getName("I am a senior lecturer  
at Swinburne") }} </p>  
</body>
```

20 - Interface Design and Development, © Swinburne



20

## ViewModel – Putting it together

### File app.js

```
Vue.createApp ({
  data() {
    return { name: "Dr Caslon Chua.",
    };
  },
  methods: {
    getName: function(bio) {
      return (this.name + bio)
    },
  },
}).mount('#app');
```

### HTML

```
<html>
  <body id="app">
    <div><p>{{getName("I am a senior lecturer at Swinburne")}}</p>
  </div> </body> </html>
```

21 - Interface Design and Development, © Swinburne



21

## ViewModel – watch

- Object (\$watch)

```
watch: {
  <modelNameToWatchforChanges>:
    function (newValue, oldValue) {
      <your code>,
    }
}
```

- \$watch triggers on change when the model value it is watching changes

22 - Interface Design and Development, © Swinburne



22

## ViewModel – Example(Built-in Object)

- HTML

```
<body id="app">
  <input type="text" v-model.lazy="strVar"/>    {{
strVar }}
</div>
</body>
```

- JavaScript

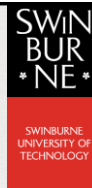
```
Vue.createApp({
  data() { return {strVar: 10} },
  methods: {},
  watch: {
    strVar(newVal, oldVal) {
      alert(oldVal + " " + newVal);
    }
  },
}).mount('#app') ;
```

23 - Interface Design and Development, © Swinburne



23

**WHAT'S NEXT?**  
**– WORKING WITH**  
**MODULES AND ROUTES**



24