

Introduction to Vue

SE 331 Component Based Software Development

Before the Vue

- Vue is the Single Page Application

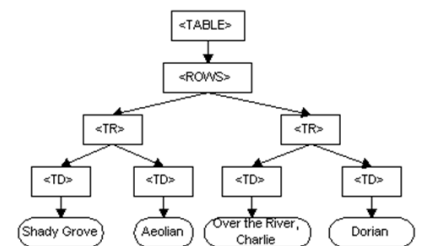
Single Page Web Application

- Many web framework is a single page web application
- Only 1 html file
- The DOM has been changed

DOM

- Document Object Model

```
<TABLE>
  <ROWS>
    <TR>
      <TD>Shady Grove</TD>
      <TD>Aeolian</TD>
    </TR>
    <TR>
      <TD>Over the River, Charlie</TD>
      <TD>Dorian</TD>
    </TR>
  </ROWS>
</TABLE>
```



Single Page Application

- Browser manipulate Dom to visualize
- Java script manipulate Dom in runtime

Vue

- Pronounce like View
- Progressive framework for building User interface
- Core library focused on view layer only

History











- First commit on July 2013
 - By Evan You, a former Google engineer
- 5 years later,
 - Vue.js is the third project on Github in number of the stars



Now

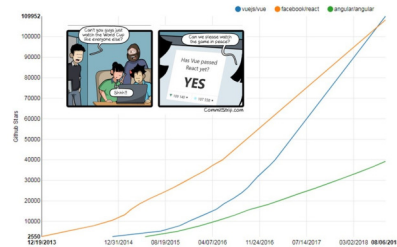
- Gitstar-ranking.com (access on 5th July 2021)

Repositories

Rank	Repository	Star
1	 freeCodeCamp/freeCodeCamp	325,573
2	 996icu/996.ICU	257,723
3	 EbookFoundation/free-programming-books	194,552
4	 vuejs/vue	184,898
5	 jwasham/coding-interview-university	183,352
6	 facebook/react	170,654
7	 sindresorhus/awesome	165,049
8	 kamranahmedse/developer-roadmap	164,516
9	 tensorflow/tensorflow	157,037
10	 twbs/bootstrap	151,519

The Popularity is increase

- Secret of Success?
 - Focus on simplicity
 - Very low learning curve
 - Excellent document
 - Listening to the community



Comparison with other popular frameworks



- Popular, used by top company
- Mature, stable, funded long-term support
- Component orient codebase
- Suitable for modern technological stacks (ES6+ / TypeScript)
- Large ecosystem of components and tools

Working with Component

- Load the components to the application
- Can use the provide source code

```
<head>
  <meta charset="UTF-8" />
  <title>Intro SE331</title>
  <!-- Import Styles -->
  <link rel="stylesheet" href="/assets/styles.css" />
  <!-- Import Vue.js -->
  <script src="https://unpkg.com/vue@3.0.11/dist/vue.global.js"></script>
</head>
```

- The script is in head
 - Confirm that it should be loaded as fast as possible

Our script is at the bottom

- Need to get all Vue component to be loaded
- Need not know all other
 - Tag, variable, code in the html

```
<body>
  <div id="app">
    <h1>Product goes here</h1>
  </div>
  <!-- Import Js -->
  <script src="/main.js"></script>
</body>
```

Creating the Vue App

- Creating the Vue app object
- The object in the memory

```
const app = Vue.createApp({  
  data() {  
    return {  
      product: 'Socks'  
    }  
  }  
})
```

createApp parameter

- The object which consists of functions
- The individual functions should be the same name
- As “Naming Convention”

```
const app = Vue.createApp({  
  data() {  
    return {  
      product: 'Socks'  
    }  
  }  
})
```

Naming Convention

- Rules how to name something
 - Method name
 - Parameter name
- The framework will select the function with the name
 - To used in the framework

In this case

- Create an app which will call method name data
- What data do
 - By specification
 - Return the variable which can be used in the app
 - Data that can be used in the web

```
const app = Vue.createApp({  
  data() {  
    return {  
      product: 'Socks'  
    }  
  }  
})
```

Before using in the html

- Mounting App
- Mounted the created object into the html dom

```
<script>
  const mountedApp = app.mount("#app")
</script>

const app = Vue.createApp({
  data() {
    return {
      product: 'Socks'
    }
  }
})
```

app.mount

- Finding the element which will used the data()
 - This help our html code to be used differently
 - Only the selected element (div) can be used the data in the app
- css selector is used

```
<script>
  const mountedApp = app.mount("#app")
</script>
```

Where is the #app

```
<body>
  <div id="app">
    <h1>Product goes here</h1>
  </div>

  <!-- Import Js -->
  <script src="./main.js"></script>
  <script>
    const mountedApp = app.mount("#app")
  </script>
</body>

const app = Vue.createApp({
  data() {
    return {
      product: 'Socks'
    }
  }
})
```

Binding the data

- Syntax
{{ variable which return from data }}

```
<div id="app">
  <h1>{{product}}</h1>
</div>

const app = Vue.createApp({
  data() {
    return {
      product: 'Socks'
    }
  }
})
```

Binding the attribute

- `{{ }}` is designed only in the content (inside the element)
- Putting to the attribute of html is not good
 - `img = "{{image}}"`
- Using the prefix in the attribute name

v-bind

- Bind the data to the attribute

```
const app = Vue.createApp({
  data() {
    return {
      product: 'Boots',
      image: './assets/images/socks_green.jpg'
    }
  }
})
```

```
<div class="product-image">
  
</div>
```

v-bind

- v-bind separate with :
 - After : should be the attribute name
- The return object in the `data()` function is found
 - Then link data and the attribute value
 - In DOM

```
<div class="product-image">
  
</div>
```

Using the v-bind before attribute name

- Help the code to look better
 - Attribute name contains only characters
- But the whole element is long
 - If we bind several attribute
- The shorthand for v-bind: is :

```
<div class="product-image">
  
</div>
```

```
<div class="product-image">
  
</div>
```

- Make the html to be read easier

Conditional Rendering

- Using the data to select which part should be shown
- if-else
- If -elseif-else

Conditional Rendering

- Provide the data in the data

```
data() {  
  return {  
    product: 'Shoes',  
    image: './assets/images/socks_green.jpg',  
    inStock: true  
  }  
}  
  
<div class="product-info">  
  <h1>{{ product }}</h1>  
  <p v-if="inStock">In Stock</p>  
  <p v-else>Out of Stock</p>  
</div>
```

What happen in Dom

- The element which has not been select has been removed from Dom

```
<div class="product-info">  
  <h1>Shoes</h1>  
  <p>In Stock</p> <!-- $0 -->  
</div>
```

v-show

- The different technique to show/hide the value
- Kept the elements in DOM
 - Hide the elements by add the `display: none;` to the element

```
data() {  
  return {  
    product: 'Shoes',  
    image: './assets/images/socks_green.jpg',  
    inStock: false  
  }  
}  
  
<div class="product-info">  
  <h1>{{ product }}</h1>  
  <p v-show="inStock">In Stock</p>  
  <p v-else>Out of Stock</p>  
</div>  
  
<h1>Shoes</h1>  
<p style="display: none;">In Stock</p> <!-- $0 -->  
<p>Out of Stock</p>
```

v-if v-else-if v-else

- The javascript Boolean expression can be added to the v-if condition

```
<p v-if="inventory > 10">In Stock</p>
<p v-else>Out of Stock</p>
```

```
data() {
  return {
    product: 'Shoes',
    image: './assets/images/socks_green.jpg',
    inStock: true,
    inventory: 100
  }
}
```

- The chain comparison is also provided

```
<div class="product-info">
  <h1>{{ product }}</h1>
  <p v-if="inventory > 10">In Stock</p>
  <p v-else-if="inventory <= 10 && inventory > 0">In Stock</p>
  <p v-else>Out of Stock</p>
</div>
```

The data may not be only the primitive type

- The arrays of objects can also be used
- It array of object can be shown as array
 - Not good for the presentation

```
[ "50% cotton", "30% wool",
  "20% polyester" ]
```

Using the for in command

- Short for -> for each element in array

```
<ul>
  <li v-for="detail in details">{{ detail }}</li>
</ul>
```

```
data() {
  return {
    product: 'Shoes',
    image: './assets/images/socks_green.jpg',
    inStock: true,
    inventory: 100,
    details: ['50% cotton', '30% wool', '20% polyester']
  }
}
```

Array of Objects

- Objects can be put in an array
- Iterate to get each object
- Access each object value by the key of the objects

```
variants: [
  { id: 2234, color: 'green' },
  { id: 2235, color: 'blue' }
]
```

```
<div v-for="variant in variants" :key="variant.id">{{ variant.color }}</div>
```


:key?

- Create a new key attribute
 - Not standard attribute
 - Use as reference to this elements later in the code

```
> <ul></ul>
<div>green</div> == 30
<div>blue</div>
</div>
```

- As it is not standard, the key attribute value is not show in the DOM
 - But keep as state of the DOM node in Vueu

Event handling

- Event that can used in the HTML
- [JavaScript Events \(w3schools.com\)](https://www.w3schools.com/js/js_events.asp)
https://www.w3schools.com/js/js_events.asp

- We can bind the script using v-on

```
<button class="button" v-on:click="addToCart">Add to Cart</button>
```

v-on

- Similar to v-bind
 - v-on: -> event name after semicolon
 - Should be assigned to the javascript method

Defining the methods

- The methods is the function which receive some data, and then manipulate the output
- Define as the object in the parameter of the createApp functions

```
const app = Vue.createApp({
  data() {
    return {
      product: 'Shoes',
      image: './assets/images/socks_green.jpg',
      inStock: true,
      inventory: 100,
      details: ['50% cotton', '30% wool', '20% polyester'],
      variants: [
        { id: 2234, color: 'green' },
        { id: 2235, color: 'blue' }
      ],
      cart: 0
    }
  },
  methods: {
    addToCart() {
      this.cart += 1
    }
  }
})
```

methods attribute

- An object
 - Contain function objects
 - The function can be applied in the app
- Same as data()
 - methods is one of the naming convention of Vue

this keyword

- keyword to emphasize that the variable is in THIS components
- the data in data() is changed or used in the method

```
const app = Vue.createApp({
  data() {
    return {
      product: 'Shoes',
      image: './assets/images/socks_green.jpg',
      inStock: true,
      inventory: 100,
      details: ['50% cotton', '30% wool', '20% polyester'],
      variants: [
        { id: 2234, color: 'green' },
        { id: 2235, color: 'blue' }
      ],
      cart: 0
    }
  },
  methods: {
    addToCart() {
      this.cart += 1
    }
  }
})
```

The shorthand

- v-on make the code a bit long
- Shorthand -> @event_name

```
<button class="button" v-on:click="addToCart">Add to Cart</button>
```

```
<button class="button" @click="addToCart">Add to Cart</button>
```

Q&A

