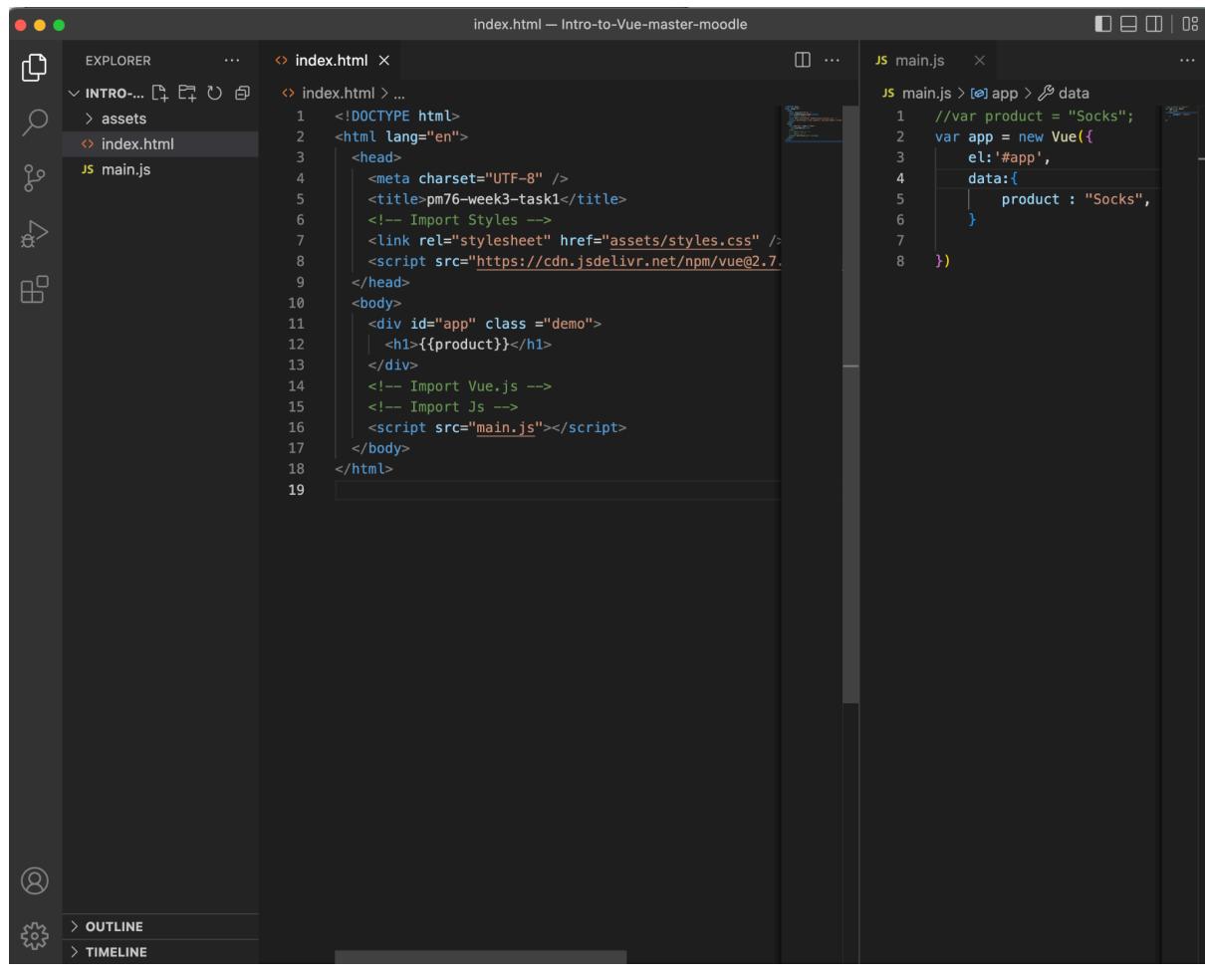




# 1. Appendix1:

## Task 1:



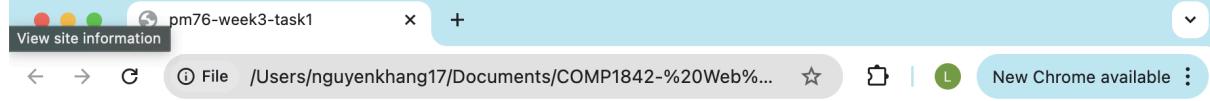
The screenshot shows the VS Code interface with two files open: `index.html` and `main.js`.

**index.html:**

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8" />
<title>pm76-week3-task1</title>
<!-- Import Styles -->
<link rel="stylesheet" href="assets/styles.css" />
<script src="https://cdn.jsdelivr.net/npm/vue@2.7.1/dist/vue.js"></script>
</head>
<body>
<div id="app" class ="demo">
<h1>{{product}}</h1>
</div>
<!-- Import Vue.js -->
<!-- Import Js -->
<script src="main.js"></script>
</body>
</html>
```

**main.js:**

```
//var product = "Socks";
var app = new Vue({
  el:'#app',
  data:{
    product : "Socks",
  }
})
```



**Socks**

*The Code for Task 1*

Challenge 1 - Intro to Vue

Vue Mastery PRO + Follow

HTML

```
1 <div id="app">
2   <div class="product-image">
3     <img src="" />
4   </div>
5
6   <div class="product-info">
7     <h1>{{ product }}</h1>
8   </div>
9
10 </div>
11
12 <script
src="https://cdn.jsdelivr.net/npm/vue@2.5.13/dist/vue.js">
```

CSS

```
1
```

JS

```
1 //Add a description to the
2 data object with the value "A
3 pair of warm, fuzzy socks".
4 Then display the description
5 using an expression in an p
6 element, underneath the h1.
7
8 var app = new Vue({
9   el: '#app',
10  data: {
11    product: 'COMP1842'
12  }
13})
```

**COMP1842**

*Challenge for task 1*

## Task 2:

The screenshot shows the VS Code interface with two files open: `index.html` and `main.js`.

`index.html` content:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Le Nguyen Khang</title>
    <!-- Import Styles -->
    <link rel="stylesheet" href="assets/styles.css" />
  </head>
  <body>
    <div class="nav-bar"></div>
    <div id="app">
      <div class="product">
        <div class="product-image">
          
        </div>
        <div class="product-info">
          <h1>{{ product }}</h1>
        </div>
      </div>
    </div>
    <!-- Import Vue.js -->
    <!-- Import Js -->
    <script src="main.js"></script>
  </body>
</html>
```

`main.js` content:

```
//var product = "Socks";
var app = new Vue({
  el: '#app',
  data: {
    product: "Socks",
    image: "./assets/images/socks_green.jpg"
  }
})
```



*The code for Task 2*

Code editor showing index.html and main.js files.

```
index.html
1 <title>Le Nguyen Khang</title>
2 <!-- Import Styles -->
3 <link rel="stylesheet" href="assets/styles.css" />
4 <script src="https://cdn.jsdelivr.net/npm/vue@2.7.14/dist/vue.js"></script>
5 </head>
6 <body>
7   <div class="nav-bar"></div>
8   <div id="app">
9     <div class="product">
10       <div class="product-image">
11         
12       <!-- Challenge Task 2 -->
13       <a v-bind:href="link"> Click the link here </a>
14     </div>
15   </div>
16 </body>
```

```
main.js
1 var app = new Vue({
2   el: '#app',
3   data: {
4     product: "Socks",
5     image: "./assets/images/socks_green.jpg",
6     link: "https://youtube.com",
7     inventory: 7,
8     details: ["80% cotton", "20% polyester", "Gender-neutral"],
9     variants: [
10       {
11         variantId: 2234,
12         variantColor: "green",
13         variantImage: "./assets/images/socks_green.jpg"
14       },
15     ]
16   }
})
```

Browser preview showing a green sock with a logo and a challenge task link.



Socks

[Click the link here](#)

<https://youtube.com>

*Challenge for task 2*

## Task 3:

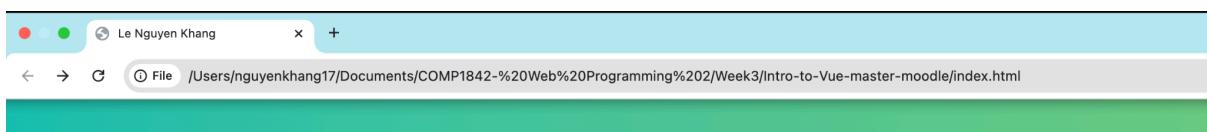
The screenshot shows a code editor interface with two files open: `index.html` and `main.js`.

`index.html` content:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Le Nguyen Khang</title>
    <!-- Import Styles -->
    <link rel="stylesheet" href="assets/styles.css" />
  </head>
  <body>
    <div class="nav-bar"></div>
    <div id="app">
      <div class="product">
        <div class="product-image">
          
          <a v-bind:href="link"> Click the link here</a>
        </div>
        <div class="product-info">
          <h1>{{ product }}</h1>
          <p v-if="inventory > 10">In Stock</p>
          <p v-else-if="inventory <= 10 && inventory > 0 ">Almost Sold Out!</p>
          <p v-else>Out of Stock</p>
        </div>
      </div>
    </div>
    <!-- Import Vue.js -->
    <!-- Import Js -->
    <script src="main.js"></script>
  </body>
</html>
```

`main.js` content:

```
//var product = "Socks";
var app = new Vue({
  el:'#app',
  data:{
    product : "Socks",
    image: "/assets/images/socks_green.jpg",
    link: "https://youtube.com",
    inventory: 7
  }
})
```



### Socks

Almost Sold Out!

[Click the link here](#)

*The code for Task 3*

The screenshot shows the VS Code interface with two tabs open: 'index.html' and 'main.js'. The 'index.html' tab displays the HTML structure for a product page, including a navigation bar, a product section with an image and details, and a footer. The 'main.js' tab shows the Vue.js script defining a component with data properties like 'product' and 'variants', and methods like 'challengeTask2' and 'challengeTask3'.

```

index.html
<!-- Import Styles -->
<link rel="stylesheet" href="assets/styles.css" />
<script src="https://cdn.jsdelivr.net/npm/vue@2.7.14/dist/vue.js"></script>
<div>
  <head>
    <title>Le Nguyen Khang</title>
  </head>
  <div class="nav-bar"></div>
  <div id="app">
    <div class="product">
      <div class="product-image">
        
        <!-- Challenge Task 2 -->
        <a v-bind:href="link"> Click the link here</a>
      </div>
      <div class="product-info">
        <h1>{{ product }}</h1>
        <!-- Challenge Task 3 -->
        <p v-if="inventory">On Sale!</p>
        <!-- v-if="inventory > 10 > To Stockless -->
      </div>
    </div>
  </div>
</div>

```

```

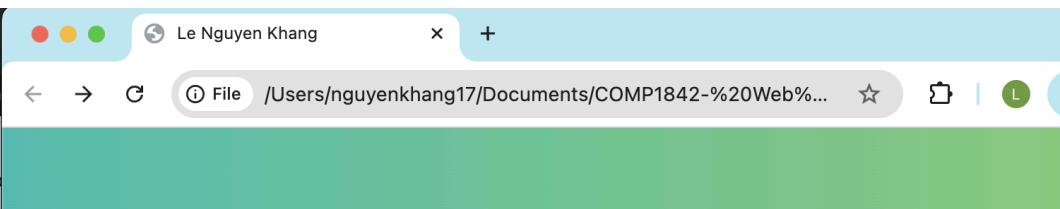
main.js
var app = new Vue({
  el: '#app',
  data: {
    product: "Socks",
    image: "./assets/images/socks_green.jpg",
    //Challenge Task 2
    link: "https://youtube.com",
    // Challenge Task 3
    inventory: true,
    details: ["80% cotton", "20% polyester", "Gender-neutral"],
    variants: [
      {
        variantId: 2234,
        variantColor: "green",
        variantImage: "./assets/images/socks_green.jpg",
      },
      {
        variantId: 2235,
        variantColor: "blue",
        variantImage: "./assets/images/socks_blue.jpg",
      }
    ],
  }
});

```



*Challenge for Task 3*

## Task 4:



The screenshot shows a web browser window titled "Le Nguyen Khang". The address bar displays the URL: "/Users/nguyenkhang17/Documents/COMP1842-%20Web%20Project/index.html". The page content is a product listing for "Socks". The image shows two green socks with a white circular logo featuring a mountain range. The text on the page includes "Almost Sold Out", "80% cotton", "20% polyester", and "Gender-neutral". The browser interface includes standard navigation buttons (back, forward, search) and a toolbar with various icons.

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Le Nguyen Khang</title>
    <!-- Import Styles -->
    <link rel="stylesheet" href="/assets/styles.css" />
    <script src="https://cdn.jsdelivr.net/npm/vue@2.7.14/dist/vue.js"></script>
  </head>
  <body>
    <div class="nav-bar"></div>
    <div id="app">
      <div class="product">
        <div class="product-image">
          
          <!-- <a v-bind:href="#"> Click the link here! </a -->
        </div>
        <div class="product-info">
          <h1>{{ product }}</h1>
          <p v-if="inventory > 10">In Stock</p>
          <p v-else-if="inventory<=10 && inventory > 0">Almost Sold Out</p>
          <p v-else>Out of Stock</p>
          <ul>
            <li v-for="detail in details">{{ detail }}</li>
          </ul>
          <div v-for="variant in variants" :key="variant.variantId">
            <p>{{ variant.variantColor }}</p>
          </div>
        </div>
        <!-- Import Vue.js -->
        <!-- Import JS -->
        <script src="main.js"></script>
      </div>
    </body>
```

main.js

```
var app = new Vue({
  el: '#app',
  data: {
    product: "Socks",
    image: "/assets/images/socks_green.jpg",
    // link: "https://youtube.com",
    inventory: 7,
    details: ["80% cotton", "20% polyester", "Gender-neutral"],
    variants: [
      {
        variantId: 2234,
        variantColor: "green"
      },
      {
        variantId: 2235,
        variantColor: "blue"
      }
    ]
})
```



## Socks

Almost Sold Out

- 80% cotton
- 20% polyester
- Gender-neutral

green

blue

*The code for Task 4*

```

EXPLORER   ...
index.html  JS main.js
assets
index.html
JS main.js

<div class="product-image">
  
  <!-- Challenge Task 2 -->
  <a v-bind:href="link" Click the link here>
</div>

<div class="product-info">
  <h1>{{ product }}</h1>
  <!-- Challenge Task 3 -->
  <p v-if="inventory>10">On Sale!</p>
  <p v-if="inventory > 10">In Stock</p>
  <p v-else-if="inventory<=10 && inventory > 0">Almost Sold Out</p>
  <p v-else>Out of Stock</p>

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

  <div v-for="variant in variants" :key="variant.variantId" >
    <p @mouseover="updateProduct(variant.variantImage)">{{ variant }}</p>
  </div>
  <!-- Challenge Task 4 -->
  <ul>
    <li v-for="size in sizes">{{ size }}</li>
  </ul>

```

```

JS main.js
1 var app = new Vue({
2   el: '#app',
3   data: {
4     product: "Socks",
5     image: "./assets/images/socks_green.jpg",
6     // challenge Task 2
7     // link: "https://youtube.com",
8
9     // Challenge Task 3
10    // inventory: true,
11    inventory: 7,
12    details: ["80% cotton", "20% polyester", "Gender-neutral"],
13    variants: [
14      {
15        variantId: 2234,
16        variantColor: "green",
17        variantImage: "./assets/images/socks_green.jpg",
18      },
19      {
20        variantId: 2235,
21        variantColor: "blue",
22        variantImage: "./assets/images/socks_blue.jpg",
23      }
24    ],
25
26    // Challenge Task 4
27    sizes: ["S", "M", "L", "XL"]
28  }

```

Le Nguyen Khang

File /Users/nguyenkhang17/Documents/COMP1842-%20Web%...

New Chrome available



## Socks

Almost Sold Out

- 80% cotton
- 20% polyester
- Gender-neutral

green

blue

- S
- M
- L
- XL

*Challenge for Task 4*

## **Understanding for Appendix 1:**

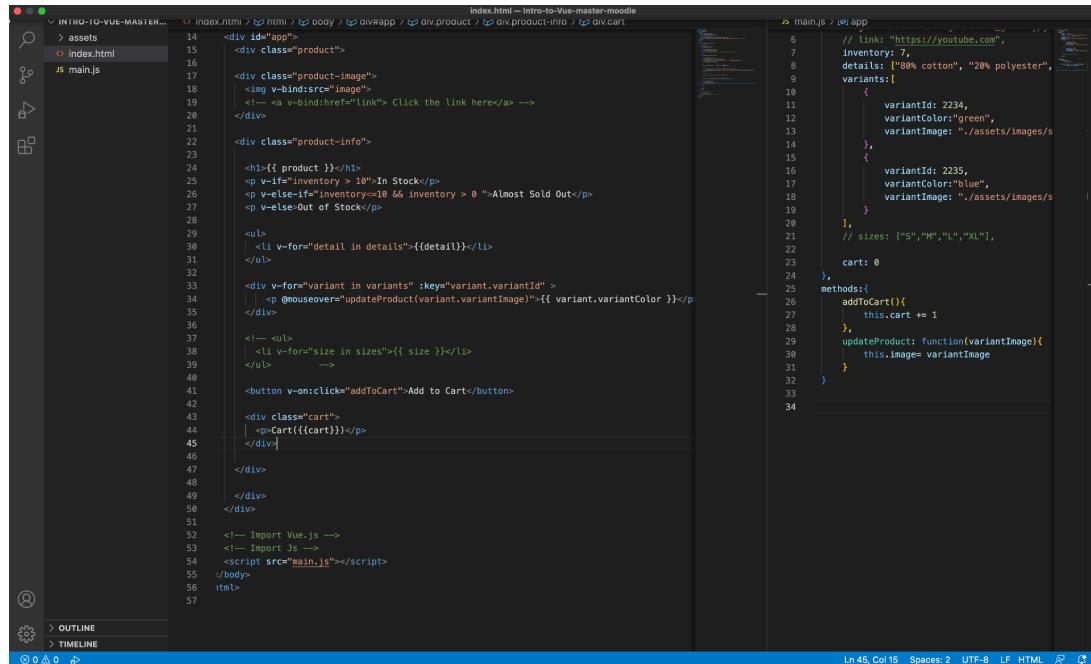
In Vue.js, creating a Vue instance connects data to the DOM, making the interface reactive. This means any changes in the JavaScript data are instantly reflected in the UI without manual updates. Vue's data binding with `{{ }}` allows dynamic data display in HTML, while v-bind enables binding attributes like `src` and `alt` in an image tag to Vue's data. Using shorthand like `:src="image"` simplifies the process, ensuring that when the image source or data changes, the UI automatically updates without reloading the page. This reactivity makes Vue ideal for building interactive, responsive web applications.

For conditional rendering, Vue uses directives like `v-if`, `v-else`, and `v-else-if` to control element display based on conditions. For example, displaying "In Stock" when `inStock` is true or showing an alternative when false. Additionally, `v-show` efficiently toggles element visibility by altering the CSS `display` property without removing elements from the DOM.

Vue's v-for directive makes list rendering straightforward by looping over data arrays and dynamically generating HTML for each item. This reduces the need for hardcoding and makes the application more scalable and maintainable. As data changes, Vue automatically updates the rendered lists, ensuring that the UI stays synchronised with the data model. Overall, Vue simplifies the development of dynamic and interactive interfaces while improving code maintainability and efficiency.

## 2. Appendix 2:

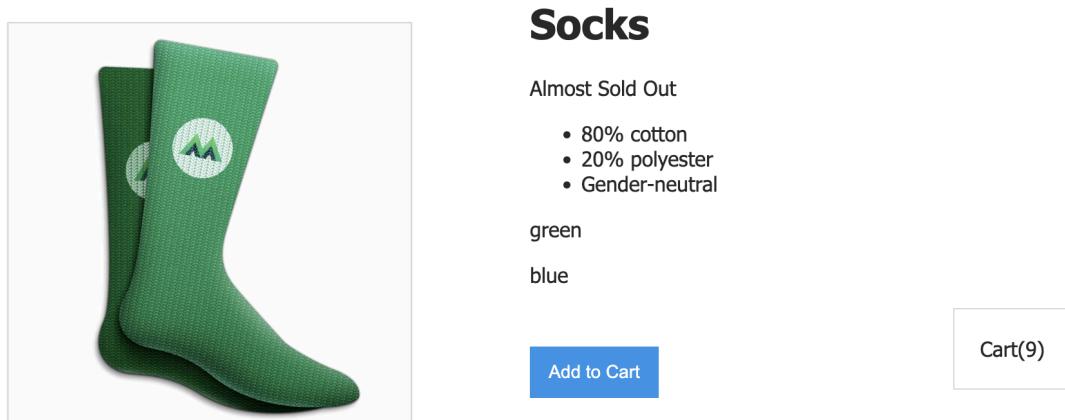
### Task 5:



The screenshot shows a code editor with two tabs open: 'index.html' and 'main.js'. The 'index.html' tab displays the structure of a single-page application. The 'main.js' tab shows the Vue.js component logic.

```
index.html - intro-to-Vue-master-moodle
<div id="app">
  <div class="product">
    <div class="product-image">
      
      <a v-bind:href="link" Click the link here/>
    </div>
    <div class="product-info">
      <h1>{{ product }}</h1>
      <p v-if="inventory > 10">In Stock</p>
      <p v-else-if="inventory<=10 && inventory > 0 ">Almost Sold Out</p>
      <p v-else>Out of Stock</p>
      <ul>
        <li v-for="detail in details">{{detail}}</li>
      </ul>
      <div v-for="variant in variants" :key="variant.variantId" >
        <p @mouseover="updateProduct(variant.variantImage)">{{ variant.variantColor }}</p>
      </div>
      <!-- -->
      <ul v-for="size in sizes">{{ size }}</li>
      </ul>
      <button v-on:click="addToCart">Add to Cart</button>
    </div>
    <div class="cart">
      <p>Cart({{cart}})</p>
    </div>
  </div>
</div>
<!-- Import Vue.js -->
<!-- Import Js -->
<script src="#main.js"></script>
</body>
</html>
```

```
main.js - [app]
6   // link: "https://youtube.com",
7   inventory: 7,
8   details: ["80% cotton", "20% polyester"],
9   variants:[
10     {
11       variantId: 2234,
12       variantColor: "green",
13       variantImage: "./assets/images/s
14     },
15     {
16       variantId: 2235,
17       variantColor: "blue",
18       variantImage: "./assets/images/s
19     }
20   ],
21   // sizes: ["S","M","L","XL"],
22   cart: 0
23 },
24 methods:{
25   addToCart(){
26     this.cart += 1
27   },
28   updateProduct(variantImage){
29     this.image= variantImage
30   }
31 }
32 }
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
```



*The code for Task 5*

The screenshot shows a VS Code interface with two tabs open: `index.html` and `JS main.js`.

**index.html:**

```
<!-- Challenge Task 3 -->
<!-- <p v-if="inventory > 0">On Sale!</p> -->
<p v-if="inventory > 10">In Stock</p>
<p v-else-if="inventory > 10 && inventory < 8">Almost Sold Out</p>
<p v-else>Out of Stock</p>

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

<div v-for="variant in variants" :key="variant.variantId" >
  | <button @mouseover="updateProduct(variant.variantImage)">{{ variant.variантId }} -->
</div>

<!-- Challenge Task 4 -->
<!-- <ul>
  | <li v-for="size in sizes">{{ size }}</li>
</ul> -->

<button v-on:click="addToCart">Add to Cart</button>
<!-- Challenge Task 5 -->
<button v-on:click="decrementCart">Remove from Cart</button>
<div class="cart">
  | <p>Cart({{cart}})</p>
</div>
</div>
</div>
</div>

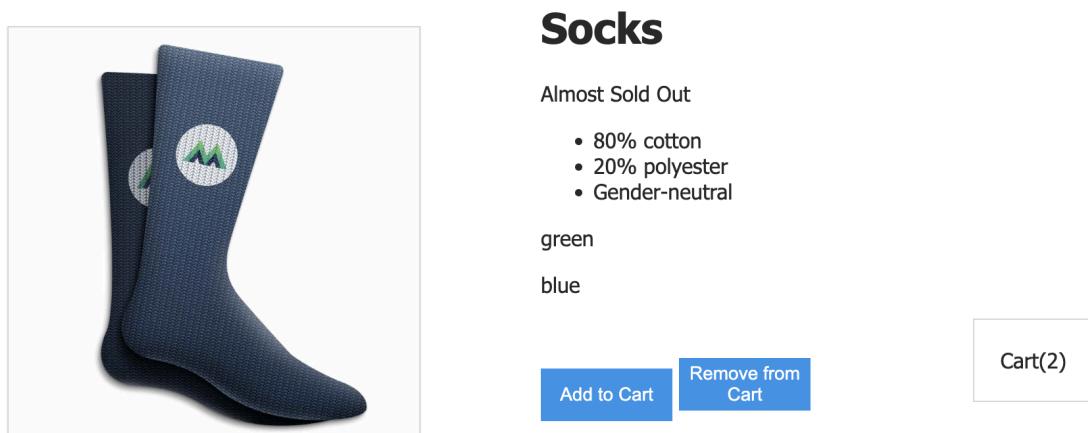
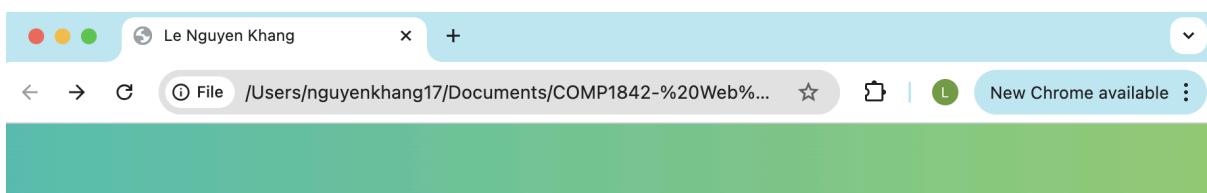
<!-- Import Vue.js -->
<!-- Import Js -->
<script src="main.js"></script>
</body>
</html>
```

**JS main.js:**

```
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62

variantId: 2234,
variantColor: "green",
variantImage: "./assets/images/socks_green.jpg",
},
{
variantId: 2235,
variantColor: "blue",
variantImage: "./assets/images/socks_blue.jpg",
}

// Challenge Task 4
// sizes: ["S","M","L","XL"],
// cart: 0,
methods:{
  addToCart(){
    this.cart += 1
  },
  updateProduct: function(variantImage){
    this.image= variantImage
  },
  decrementCart(){
    this.cart -=1
  }
}
```

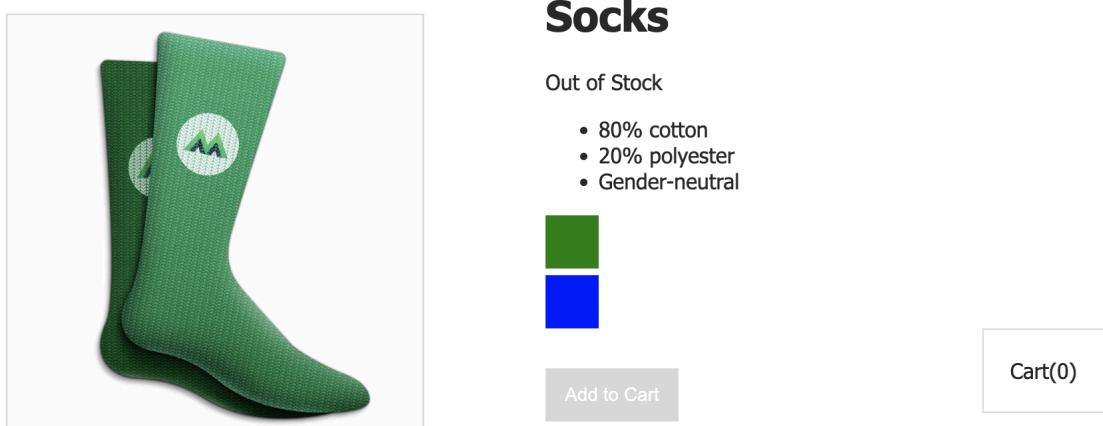


### *Challenge for Task 5*

## Task 6:

The screenshot shows a code editor with two tabs open: 'index.html' and 'main.js'. The 'index.html' tab displays the HTML structure for a product page, including a header, a product image, a product info section with a title and details, a variant selection section, and a cart section. The 'main.js' tab shows the corresponding Vue.js script. It defines a Vue instance 'app' with data properties like 'product', 'image', 'details', 'inStock', 'variants', and 'cart'. It includes methods for adding items to the cart and updating the product's state based on its stock level.

```
index.html
main.js
```



*The code for Task 6*

EXPLORER    ...    index.html    # styles.css

```

INTRO-TO-VUE-MASTER...
  assets
    > images
      # styles.css
    > index.html
  JS main.js

index.html
1  !DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8" />
5    <title>Le Nguyen Khang</title>
6    <!-- Import Styles -->
7    <link rel="stylesheet" href="#{assets}/styles.css" />
8    <script src="https://cdn.jsdelivr.net/npm/vue@2.7.14/dist/vue.js"></script>
9
10 </head>
11
12 <div class="nav-bar"></div>
13
14 <div id="app">
15   <div class="product">
16     <div class="product-image">
17       
18       <!-- Challenge Task 2 -->
19       <!-- <a v-bind:href="link"> Click the link here</a> -->
20     </div>
21
22     <div class="product-info">
23
24       <h1>{{ product }}</h1>
25       <!-- Challenge Task 3 -->
26       <!-- <p v-if="inStock">On Sale!</p> -->
27       <p v-if="inStock">In Stock</p>
28       <!-- <p v-else-if="inStock<=10 && inStock > 0 ">Almost Sold Out</p> -->
29
30       <!-- Challenge Task 6 -->
31       <p v-else :class="{ outOfStock: !inStock }" >Out of Stock</p>
32
33     <ul>
34       <li v-for="detail in details">{{detail}}</li>
35     </ul>
36
37     <div v-for="variant in variants"
38       :key="variant.variantId"
39       class="color-box"
40       :style="{ backgroundColor: variant.variantColor }"
41       @mouseover="updateProduct(variant.variantImage)">
42
43   </div>

```

# styles.css

```

assets > # styles.css > #.activeTab
59   .disabledButton {
60     background-color: #d8d8d8;
61   }
62
63   .review-form {
64     width: 400px;
65     padding: 20px;
66     margin: 40px;
67     border: 1px solid #d8d8d8;
68   }
69
70   input {
71     width: 100%;
72     height: 25px;
73     margin-bottom: 20px;
74   }
75
76   textarea {
77     width: 100%;
78     height: 60px;
79   }
80
81   .tab {
82     margin-left: 20px;
83     cursor: pointer;
84   }
85
86   .activeTab {
87     color: #1E80B0;
88     text-decoration: underline;
89   }
90
91   /* Challenge for Task 6 */
92   .outOfStock {
93     text-decoration: line-through;
94   }

```



# Socks

**Out of Stock**

- 80% cotton
- 20% polyester
- Gender-neutral




Add to Cart

Cart(0)

*Challenge for Task 6*

## Task 7:

```

EXPLORER ... index.html x JS main.js # styles.css ...
INTRO-TO-VUE-MASTER... index.html > HTML > BODY > DIV#APP > DIV.PRODUCT > DIV.PRODUCT-INFO > DIV.CARD
  | <!-- <a v-bind:href="link"> Click the link here</a> -->
  </div>
<div class="product-info">
  <h1>{{ title }}</h1>
  <!-- Challenge Task 3 -->
  <!-- <p v-if="inStock">On Sale!</p> -->
  <p v-if="inStock">In Stock</p>
  <!-- <p v-else-if="inStock<=10 && inStock > 0 ">Almost Sold Out</p>
  <!-- Challenge Task 6 -->
  <!-- <p v-else :class="{ outOfStock: !inStock }">Out of Stock</p>
  <p v-else>Out of Stock</p>

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

  <div v-for="(variant, index) in variants"
    :key="variant.variantId"
    class="color-box"
    :style="{ backgroundColor: variant.variantColor }"
    @mouseover="updateProduct(index)">
    </div>
    <!-- Challenge Task 4 -->
    <!-- <ul>
      <li v-for="size in sizes">{{ size }}</li>
    </ul>
    -->

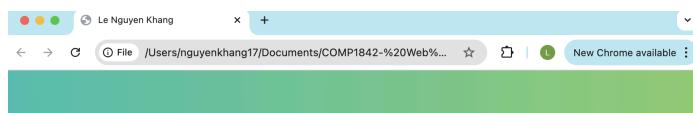
    <button v-on:click="addToCart"
      :disabled="!inStock"
      :class="{ disabledButton: !inStock }">Add to Cart</button>
    <!-- Challenge Task 5 -->
    <!-- <button v-on:click="decrementCart">Remove from Cart</button>
    <div class="cart">
      <p>Cart({{ cart }})</p>
    </div>
  </div>
</div>
</div>

```

```

JS main.js > [app]
variants: [
  {
    variantId: 2234,
    variantColor: "green",
    variantImage: "./assets/images/socks_green.jpg",
    variantQuantity: 10
  },
  {
    variantId: 2235,
    variantColor: "blue",
    variantImage: "./assets/images/socks_blue.jpg",
    variantQuantity: 0
  }
],
// Challenge Task 4
// sizes: ["S", "M", "L", "XL"],
cart: 0,
methods: {
  addToCart() {
    this.cart += 1
  },
  updateProduct(index) {
    this.selectedVariant = index
    console.log(index)
  },
},
computed: {
  title() {
    return this.brand + ' ' + this.product
  },
  image() {
    return this.variants[this.selectedVariant].variantImage
  },
  inStock() {
    return this.variants[this.selectedVariant].variantQuantity
  }
}

```



### Vue Mastery Socks

In Stock

- 80% cotton
- 20% polyester
- Gender-neutral



Add to Cart

Cart(0)



### Vue Mastery Socks

Out of Stock

- 80% cotton
- 20% polyester
- Gender-neutral



Add to Cart

Cart(0)

*The code for Task 7*

```

    var app = new Vue({
      el: '#app',
      data: {
        brand: 'Vue Mastery',
        product: "Socks",
        selectedVariant: 0,
        //Challenge Task 7
        onSale: true,
        //Challenge Task 2-
      },
      details: ["80% cotton", "20% polyester", "Gender-neutral"],
      variants: [
        { ... },
        { ... }
      ],
      // Challenge Task 4-
      cart: 0
    },
    methods: {
      addToCart() {
        ...
      },
      updateProduct: function(index) {
        ...
      }
    },
    // Challenge Task 5
    // decrementCart(){
    //   this.cart -=1
    // }
  },
  computed: {
    // Challenge Task 7
    title() {
      if (this.onSale) {
        return this.brand + ' ' + this.product + ' is on Sale!';
      } else {
        return this.brand + ' ' + this.product;
      }
      // return this.brand + ' ' + this.product
    },
    image() {
      return this.variants[this.selectedVariant].variantImage
    },
    inStock() {
      ...
    }
  }
}

```

The screenshot shows the VS Code interface with the Explorer, Editor, and Terminal panes. The Editor pane displays a file named main.js containing Vue.js code for a product component. The code includes properties like brand, product, and selectedVariant, methods for addToCart and updateProduct, and computed properties for title and image. The code is annotated with comments for challenge tasks 2 through 7.



## Vue Mastery Socks is on Sale!

In Stock

- 80% cotton
- 20% polyester
- Gender-neutral



Add to Cart

Cart(0)

*Challenge for Task 7*

## Understanding for Appendix 2:

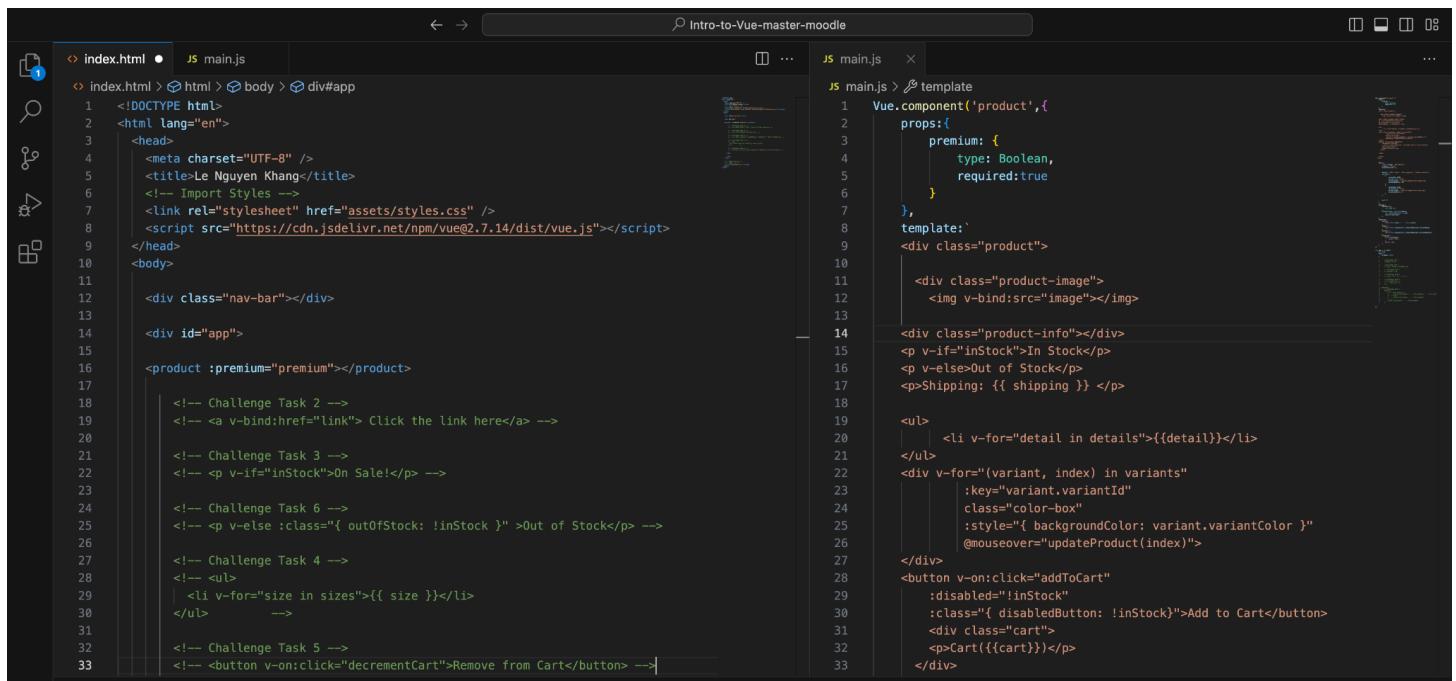
Vue's `v-on` directive is a powerful tool for handling DOM events such as clicks or mouseovers, enabling developers to trigger methods that enhance the interactivity of applications. Using the shorthand `@`, event types like `@click` can easily trigger methods such as `addToCart`, which dynamically updates the data model by referencing properties like `this.cart`. This integration keeps the application reactive and ensures that user interactions are instantly reflected in the UI.

In working with class and style binding, I learned how to dynamically link data-driven styles and classes to HTML elements. For instance, using `v-bind:style` allowed me to change a `div`'s background color based on product variants, while class binding greyed out buttons when a product was out of stock. These bindings ensure that the UI adapts automatically to changes in data, offering a seamless user experience without manual DOM manipulation.

Additionally, Vue's computed properties calculate values dynamically, such as combining product brand and name into a single string, streamlining the logic and reducing redundancy in the data model. By leveraging Vue's reactivity and directives like `v-for` for list rendering, I saw how Vue simplifies complex UI updates, providing cleaner, more maintainable code for scalable applications.

## 3. Appendix 3:

### Task 8:



The screenshot shows a code editor with two tabs: 'index.html' and 'main.js'. The 'index.html' tab displays the following HTML code:

```
1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <meta charset="UTF-8" />
5      <title>Le Nguyen Khang</title>
6      <!-- Import Styles -->
7      <link rel="stylesheet" href="assets/styles.css" />
8      <script src="https://cdn.jsdelivr.net/npm/vue@2.7.14/dist/vue.js"></script>
9
10 </head>
11 <body>
12   <div class="nav-bar"></div>
13
14   <div id="app">
15     <product :premium="premium"></product>
16
17     <!-- Challenge Task 2 -->
18     <!-- <a v-bind:href="link"> Click the link here</a> -->
19
20     <!-- Challenge Task 3 -->
21     <!-- <p v-if="inStock">On Sale!</p> -->
22
23     <!-- Challenge Task 6 -->
24     <!-- <p v-else :class="{ outOfStock: !inStock }" >Out of Stock</p> -->
25
26     <!-- Challenge Task 4 -->
27     <!-- <ul>
28       <li v-for="size in sizes">{{ size }}</li>
29     </ul>
30     -->
31
32     <!-- Challenge Task 5 -->
33     <!-- <button v-on:click="decrementCart">Remove from Cart</button> -->
```

The 'main.js' tab displays the following JavaScript code:

```
1  Vue.component('product', {
2    props: {
3      premium: {
4        type: Boolean,
5        required: true
6      }
7    },
8    template: `
9      <div class="product">
10        <div class="product-image">
11          
12        </div>
13        <div class="product-info">
14          <p v-if="inStock">In Stock</p>
15          <p v-else>Out of Stock</p>
16          <p>Shipping: {{ shipping }}</p>
17
18          <ul>
19            <li v-for="detail in details">{{ detail }}</li>
20          </ul>
21          <div v-for="(variant, index) in variants"
22            :key="variant.variantId"
23            class="color-box"
24            :style="`backgroundColor: variant.variantColor`"
25            @mouseover="updateProduct(index)">
26            </div>
27          <button v-on:click="addCart"
28            :disabled="!inStock"
29            :class="{ disabledButton: !inStock }">Add to Cart</button>
30          <div class="cart">
31            <p>Cart({{ cart }})</p>
32          </div>
33        </div>
34      </div>
35    `
36  })
37
38  new Vue({
39    el: '#app',
40    data: {
41      cart: 0,
42      premium: false,
43      shipping: 5
44    }
45  })
```

```

JS main.js > template
41     data(){
42         variants:[
43             {
44                 variantId: 2234,
45                 variantColor:"green",
46                 variantImage: "./assets/images/socks_green.jpg",
47                 variantQuantity: 10
48             },
49             {
50                 variantId: 2235,
51                 variantColor:"blue",
52                 variantImage: "./assets/images/socks_blue.jpg",
53                 variantQuantity: 0
54             }
55         ],
56         cart: 0
57     },
58     methods:{
59         addToCart(){
60             this.cart += 1
61         },
62         updateProduct: function(index){
63             this.selectedVariant = index
64             console.log(index)
65         },
66     },
67     computed:{
68         title(){
69             return this.brand + ' ' + this.product
70         },
71         image() {
72             return this.variants[this.selectedVariant].variantImage
73         },
74         inStock() {
75             return this.variants[this.selectedVariant].variantQuantity
76         },
77         shipping(){
78             if(this.premium){
79                 return "Free"
80             }
81             return 2.99
82         }
83     }
84 }
85
86 var app = new Vue({
87     el:'#app',
88     data: {
89         premium: false
90     }
91 }
92 )
93
94
95
96
97
98
99
100

```

Le Nguyen Khang

In Stock

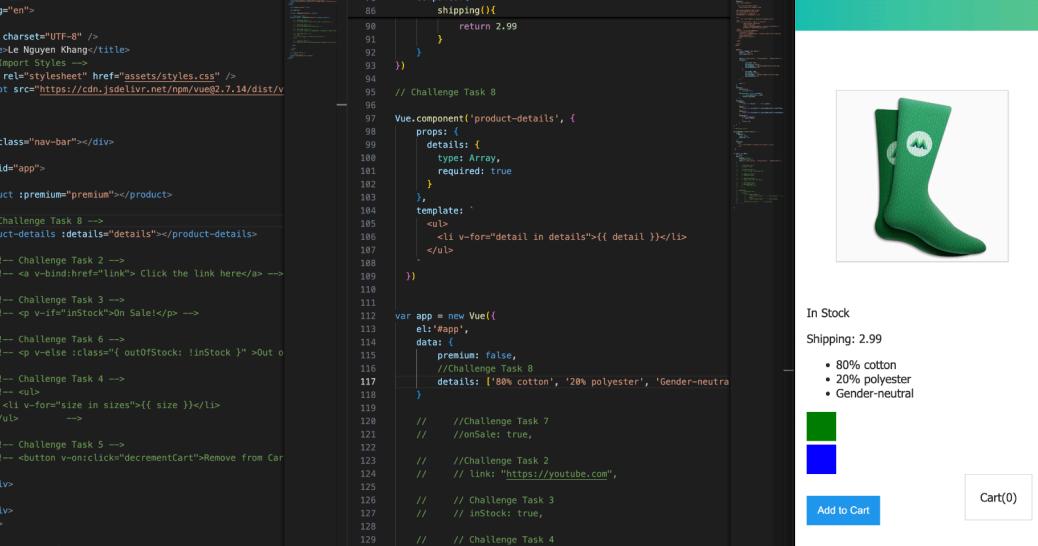
Shipping: 2.99

- 80% cotton
- 20% polyester
- Gender-neutral

Add to Cart

Cart(0)

*The code for Task 8*



The screenshot shows a development environment with two tabs open: 'index.html' and 'js main.js'. The 'index.html' tab contains the HTML structure for a product page, including a header, navigation bar, and a main content area with a 'product' component. The 'js main.js' tab contains the corresponding Vue.js code for the 'product' component, defining props, computed properties like 'shipping()', and a template that includes a list of details. The browser window shows the final rendered product page with two green socks, the text 'In Stock', and 'Shipping: 2.99'. A sidebar on the right provides detailed product information: '80% cotton', '20% polyester', and 'Gender-neutral'. At the bottom, there's a blue 'Add to Cart' button and a box labeled 'Cart(0)'. The overall interface is clean and modern, typical of a Vue.js application.

## *Challenge for Task 8*

## Task 9:



Cart(0)

## Vue Mastery Socks

In Stock

Shipping: 2.99

- 80% cotton
- 20% polyester
- Gender-neutral

Add to Cart

```
index.html
<div class="cart">
  <p>Cart({{ cart.length }})</p>
</div>
<product :premium="premium" @add-to-cart="updateCart"></product>
```

```
main.js
computed: {
  variantQuantity() {
    return this.variants[this.selectedVariant].variantQuantity;
  }
},
methods: {
  addCart() {
    this.$emit('add-to-cart', this.variants[this.selectedVariant].variantId);
  },
  updateProduct(index) {
    this.selectedVariant = index;
    console.log(index);
  }
},
computed: {}
```

```
var app = new Vue({
  el: '#app',
  data: {
    premium: false,
    cart: []
  },
  //Challenge Task 8
  // details: ['80% cotton', '20% polyester', 'Gender-neutral']
}, {
  methods: {
    updateCart(id) {
      this.cart.push(id)
    }
  }
})
// Challenge Task 8
```

Ln 6, Col 27 Spaces: 2 UTF-8 LF HTML

## *The code for Task 9*

```

    63     methods: {
    64       addVariant() {
    65         ...
    66       },
    67       //Challenge Task 9
    68       removeFromCart() {
    69         this.$emit('remove-from-cart', this.variants[this.selectedVariant].variantId)
    70       },
    71       updateProduct(index) {
    72         this.selectedVariant = index
    73         console.log(index)
    74       }
    75     },
    76   },
    77   computed: {
    78     ...
    79   }
    80 }
    81
    82
    83
    84
    85
    86
    87
    88
    89
    90
    91
    92
    93
    94
    95
    96
    97
    98
    99
    100 var app = new Vue({
    101   el:'#app',
    102   data: {
    103     ...
    104   },
    105   methods: {
    106     updateCart(id){
    107       this.cart.push(id)
    108     },
    109     //Challenge Task 9
    110     removeFromCart(id) {
    111       this.cart.pop(id)
    112     }
    113   }
    114 }
    115 )
    116
    117 <!-- Challenge Task 9 -->
    118 <product premium="premium" @add-to-cart="updateCart" @remove-from-cart="removeFromCart"></product>
    119
    120 <!-- Challenge Task 8 -->
  
```

### Challenge code for Task 9

## Task 10:

```

    109   ...
    110   ...
    111   ...
    112   ...
    113   Vue.component('product-review',{
    114     template:
    115       `<form class="review-form" @submit.prevent="onSubmit">
    116         <p v-if="errors.length">
    117           <b>Please correct the following errors:</b>
    118           <ul>
    119             <li v-for="error in errors">{{ error }}</li>
    120           </ul>
    121         </p>
    122         <p>
    123           <label for="name">Name:</label>
    124           <input id="name" v-model="name" placeholder="name">
    125         </p>
    126         <p>
    127           <label for="review">Review:</label>
    128           <textarea id="review" v-model="review" required></textarea>
    129         </p>
    130         <p>
    131           <label for="rating">Rating:</label>
    132           <select id="rating" v-model.number="rating">
    133             <option>5</option>
    134             <option>4</option>
    135             <option>3</option>
    136             <option>2</option>
    137             <option>1</option>
    138           </select>
    139         </p>
    140       </form>
    141     `,
    142     data(){
    143       return{
    144         name:null,
    145         review:null,
    146         rating:null,
    147         errors: []
    148       }
    149     },
    150     methods: {
    151       ...
    152       ...
    153       ...
    154       ...
    155       ...
    156       ...
    157       ...
    158     }
    159   }
  
```

### The code for Task 10

```

    <!-- Product Card -->
    <div>
      In Stock
      Shipping: 2.99
      • 80% cotton
      • 20% polyester
      Gender-neutral
      <img alt="Product image" style="display: block; margin: auto;" />
      <button>Add to Cart
    </div>

    <!-- Review Form -->
    <div>
      Name: 
      Review: 
      Rating: <select></select>
      Would you recommend this product? <input checked="" type="radio"/>
      <br/>
      Yes <input type="radio"/>
      No <input type="radio"/>
      <button>Submit</button>
    </div>

    <!-- Main JS File -->
    <script>
      <!-- Challenge code for Task 10 -->
      <!-- Product Card Logic -->
      <!-- Review Form Logic -->
    </script>
  
```

### *Challenge code for Task 10*

## **Understanding for Appendix 3:**

This task focused on the use of Vue.js components to create a modular, reusable, and maintainable application structure. By encapsulating product-related functionality within a `product` component, the application became more efficient, with a cleaner structure and enhanced reusability. This approach minimises code duplication and facilitates easier updates across multiple instances.

A key aspect of this work involved managing data flow between parent and child components using "props" and "events". The parent component passed a `premium` value to the `product` component, which used it to dynamically adjust shipping costs. Computed properties were also employed to manage conditional logic, ensuring dynamic updates to the UI based on user status.

In addition to downward data flow via props, the child component communicated events like "add-to-cart" back to the parent using `\$emit`. This event-driven model allowed the parent to respond to user actions while maintaining component independence and modularity.

Furthermore, two-way data binding was implemented through Vue's `v-model`, enabling user input in forms to directly modify component data. Custom form validation logic ensured required fields were filled before submission, further

enhancing the application's interactivity and responsiveness. Overall, this work demonstrates the efficient use of Vue's features to create dynamic, interactive applications with clear separation of concerns and streamlined data flow between components.

## 4. MongoDB

### Lesson 4 exercise 1:

The screenshot shows the Studio 3T interface for MongoDB. The left sidebar displays the database structure under 'Open connections'. A search bar at the top right contains the query: `{"name": {"\$ne": "Caerphilly"}, "local\_authority": {"\$not": {"\$in": ["Caerphilly"]}}}`. The results table shows a list of pubs with their IDs, names, addresses, cities, and local authorities. The data includes entries like 'Mark Williams S', 'Brackla Snooker', 'Penarth Road S', 'SNOOKER WORL', etc., from various locations such as Cardiff, Ammanford, Llanelli, and Conwy.

### Lesson 4 exercise 3:

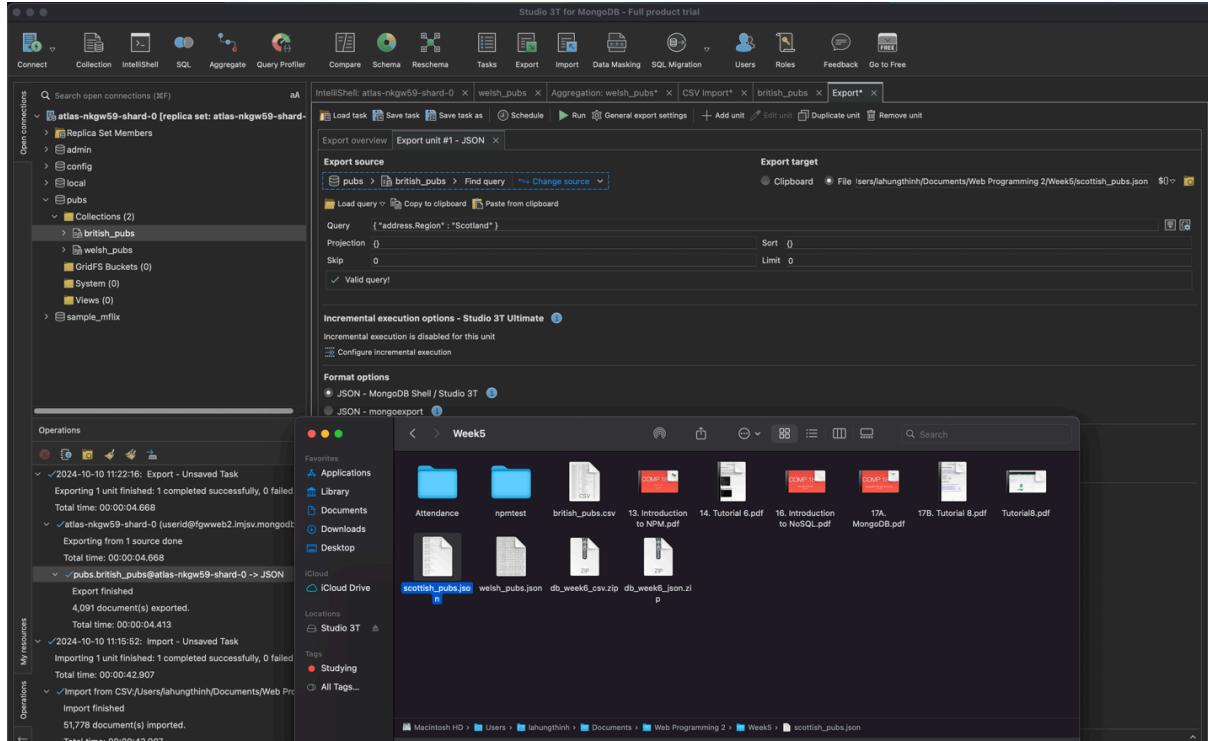
The screenshot shows the Studio 3T interface for MongoDB with an aggregation pipeline. The pipeline stages are:

```

    db.welsh_pubs.aggregate([
      { $match: {} },
      { $group: {} },
      { $project: {} },
      { $sort: {} }
    ])
  
```

The output stage shows a table with columns 'Stage 4: Output' and 'Stage 4: Input'. The output table lists local authorities and their amounts: Powys (9), Vale of Glamorgan (7), Monmouthshire (5), Pembrokeshire (3), Bridgend (3), Carmarthenshire (3), Wrexham (3), Newport (2), Blaenau Gwent (2), and Ceredigion (1). The input table shows the same data with an additional 'amount' column.

## Lesson 5 exercise 3:



## Lesson 6 exercise 2:

NoSQL databases, such as MongoDB, provide a flexible approach to data storage, differing significantly from traditional relational databases. Unlike the rigid table structure of relational databases, NoSQL databases are schema-less, which allows for the storage of diverse data types and structures without predefined schemas. MongoDB is a document-oriented NoSQL database that organizes data into three hierarchical levels: databases, collections, and documents. Collections hold multiple documents, which consist of key-value pairs similar to JSON objects. This document model offers versatility in storing complex, nested data and accommodates changes in data structures over time without the need for database migrations.

A key feature of MongoDB is its support for embedded documents, which allow documents to include nested arrays or objects. This enables complex relationships to be represented within a single document, reducing the need for joins, which are common in relational databases. As a result, MongoDB can manage relationships with fewer collections, enhancing query performance and simplifying data modeling.

MongoDB also includes aggregation capabilities, which are used for data processing and transformation. Its aggregation framework allows for operations like filtering, grouping, and sorting using an aggregation pipeline, which chains multiple stages to manipulate data efficiently. This approach is analogous to SQL's `GROUP BY` but

with greater flexibility and power, enabling complex data transformations and analysis. Overall, MongoDB's design makes it suitable for applications that require fast development cycles, dynamic data structures, and scalability.