

Full Name / Student ID	Huynh Thanh Tuyen / 001323067
Submission Title	COS1103 Task 1 & 2
Course Name	Web Programming 2
Course Code	COS1103

## Task 1 (Lesson 1-4)

### Lesson 1: The Vue Instance

The screenshot shows a browser window with the URL `127.0.0.1:5500/Week3/task1.html`. The page content is "Socks". The browser's developer tools are open, showing the following code:

```

index.html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" type="text/css" href="assets/styles.css">
</head>
<body>
    <div id="app">
        <h1>{{ product }}</h1>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/vue@2.7.14/dist/vue.js"></script>
    <script src="main.js"></script>
</body>
</html>

```

```

main.js
var app = new Vue({
    el: '#app',
    data: {
        product: "Socks",
        brand: "Vue Mastery",
        onSale: true,
        selectedVariant: 0,
        altText: "A pair of socks",
        inventory: 100,
        details: ["80% cotton", "20% polyester", "Gender-neutral"],
        variants: [
            {
                variantId: 2234,
                variantColor: "green",
                variantImage: "./assets/images/socks_green.jpg",
                variantQuantity: 10,
            },
            {

```

## Challenge of lesson 1:

The screenshot shows a code editor interface with several files open. On the left, the file tree (EXPLORER) shows a project structure with folders LAB, Week1, Week2, Week3, assets, index.html, main.js, and task1.html. The main area displays the content of task1.html and main.js.

**task1.html:**

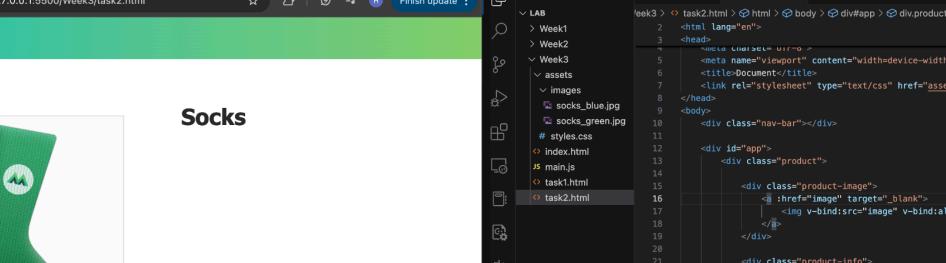
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" type="text/css" href="assets/styles.css">
</head>
<body>
    <div id="app">
        <h1>{{ product }}</h1>
        <p>{{ description }}</p>
    </div>
</body>
</html>
```

**main.js:**

```
var app = new Vue({
    el: '#app',
    data: {
        product: "Socks",
        description: "A pair of warm, fuzzy socks",
        brand: "Vue Mastery",
        onSale: true,
        selectedVariant: 0,
        altText: "A pair of socks",
        inventory: 100,
        details: ["80% cotton", "20% polyester", "Gender-neutral"],
        variants: [
            {
                variantId: 2234,
                variantColor: "green",
                variantImage: "./assets/images/socks_green.jpg",
                variantQuantity: 10,
            },
        ],
    },
})
```

At the bottom of the interface, there are status bars showing "Ln 12, Col 26" and "Port: 3000".

## Lesson 2: Attribute Binding



The screenshot shows a browser window with two tabs open. The active tab is titled 'task2.html' and displays a product page for 'Socks'. The page features a large image of a green sock with a white logo. A modal window is overlaid on the page, containing the text 'Finish update'.

The left side of the interface shows a file explorer with a tree view of files. The 'LAB' folder contains 'Week1', 'Week2', and 'Week3'. 'Week3' contains 'assets' (with 'images' subfolder), 'index.html', 'main.js', 'task1.html', and 'task2.html'. The 'task2.html' file is currently selected.

The right side of the interface shows the code editor with the content of 'task2.html'. The code includes HTML for a navigation bar, a product container, and a product image, along with some CSS and JS. The JS section defines a Vue.js application 'app' with a 'data' object containing product details like 'product': 'Socks', 'description': 'A pair of warm, fuzzy socks', and 'variants': [ { 'variantId': 2234, 'variantColor': 'green', 'variantImage': '/assets/images/socks\_green.jpg', 'variantInventory': 30 } ].

## Challenge of lesson 2

The screenshot shows a browser window with a Vue.js application. The application displays a pair of green socks with a white mountain logo. The code editor on the right shows the file structure and code for task2.html. The code includes a Vue component for the socks, which uses an image and some descriptive text.

```
index.html
task2.html x
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <link rel="stylesheet" type="text/css" href="assets/styles.css">
</head>
<body>
  <div class="nav-bar"></div>
  <div id="app">
    <div class="product">
      <div class="product-image">
        <a href="#" target="_blank">
          
        </a>
      </div>
      <div class="product-info">
        <h1>{{ product.name }}</h1>
      </div>
    </div>
  </div>
</body>
<script src="https://cdn.jsdelivr.net/npm/vue@2.7.14/dist/vue.js"></script>
<script src="main.js"></script>
```

```
JS main.js x
var app = new Vue({
  el: '#app',
  data: {
    product: "Socks",
    description: "A pair of warm, fuzzy socks",
    image: "assets/images/socks_green.jpg",
    altText: "A pair of socks",
    brand: "Vue Mastery",
    onSale: true,
    selectedVariant: 0,
    inventory: 100,
    details: ["80% cotton", "20% polyester", "Gender-neutral"],
    variants: [
      {
        variantId: 2234,
        variantColor: "green",
        variantImage: "assets/images/socks_green.jpg",
        variantQuantity: 10,
```

## Lesson 3: Conditional Rendering



```
EXPLORER
LAB
> Week1
> Week2
> Week3
  > assets
    > images
      < socks_blue.jpg
      < socks_green.jpg
    > # styles.css
  > index.html
  > JS main.js
  > task1.html
  > task2.html
  > task3.html

index.html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Lab</title>
  </head>
  <body>
    <div id="app">
      <div class="product">
        <div class="product-image">
          <a href="#" target="_blank">
            
          </a>
        </div>
        <div class="product-info">
          <h1>{{ product }}</h1>
          <p v-if="inStock">> {{ inventory }} 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>
  </body>
</html>
```

```
task3.html
<script src="https://cdn.jsdelivr.net/npm/vue@2.7.14/dist/vue.js"></script>
<script src="main.js"></script>
```

```
JS main.js
Week3 > JS main.js > [e] app > ⚡ data > ⚡ inventory
1 var app = new Vue({
  2   data: [
  3     {
  4       product: "Socks",
  5       description: "A pair of warm, fuzzy socks",
  6       image: "/assets/images/socks_green.jpg",
  7       altText: "A pair of socks",
  8       inStock: true,
  9       inventory: 100,
 10      // brand: "Vue Mastery",
 11      // onSale: true,
 12      // selectedVariant: 0,
 13      // details: ["80% cotton", "20% polyester", "Gender-neutral"],
 14      // variants: [
 15      //   {
 16      //     variantId: 2234,
 17      //     variantColor: "green",
 18      //     variantImage: "/assets/images/socks_green.jpg",
 19      //     variantQuantity: 10,
```



```
EXPLORER
LAB
> Week1
> Week2
> Week3
  > assets
    > images
      < socks_blue.jpg
      < socks_green.jpg
    > # styles.css
  > index.html
  > JS main.js
  > task1.html
  > task2.html
  > task3.html

index.html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Lab</title>
  </head>
  <body>
    <div id="app">
      <div class="product">
        <div class="product-image">
          <a href="#" target="_blank">
            
          </a>
        </div>
        <div class="product-info">
          <h1>{{ product }}</h1>
          <p v-if="inStock">> {{ inventory }} 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>
  </body>
</html>
```

```
task3.html
<script src="https://cdn.jsdelivr.net/npm/vue@2.7.14/dist/vue.js"></script>
<script src="main.js"></script>
```

```
JS main.js
Week3 > JS main.js > [e] app > ⚡ data > ⚡ inventory
1 var app = new Vue({
  2   data: [
  3     {
  4       product: "Socks",
  5       description: "A pair of warm, fuzzy socks",
  6       image: "/assets/images/socks_green.jpg",
  7       altText: "A pair of socks",
  8       inStock: true,
  9       inventory: 10,
```

Socks

Out of Stock



```

<div class="product-image">
  <a href="image" target="_blank">
    
  </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>

```

```

var app = new Vue({
  data: {
    product: "Socks",
    description: "A pair of warm, fuzzy socks",
    image: "/assets/images/socks_green.jpg",
    altText: "A pair of socks",
    inStock: true,
    inventory: 0,
    // brand: "Vee Mastery",
    // onSale: true,
    // selectedVariant: 0,
    // details: ["80% cotton", "20% polyester", "Gender-neutral"],
    // variants: [
    //   {
    //     variantId: 2234,
    //     variantColor: "green",
    //     variantImage: "/assets/images/socks_green.jpg",
    //     variantQuantity: 10,
    
```

## Challenge of lesson 3

In Stock

On Sale!



```

<div class="product-image">
  <a href="image" target="_blank">
    
  </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>
  <span v-show="onSale">On Sale!</span>
</div>

```

```

var app = new Vue({
  data: {
    product: "Socks",
    description: "A pair of warm, fuzzy socks",
    image: "/assets/images/socks_green.jpg",
    altText: "A pair of socks",
    inStock: true,
    inventory: 100,
    onSale: true,
    // brand: "Vee Mastery",
    // selectedVariant: 0,
    // details: ["80% cotton", "20% polyester", "Gender-neutral"],
    // variants: [
    //   {
    //     variantId: 2234,
    //     variantColor: "green",
    //     variantImage: "/assets/images/socks_green.jpg",
    //     variantQuantity: 10,
    
```

Screenshot of a development environment showing a Vue.js application for a product page.

**Browser View:**

- URL: 127.0.0.1:5500/Week3/task3.html
- Page Content: A green sock with a white mountain logo. Below it is the text "Socks" and "In Stock".

**Code Editor (VS Code):**

- EXPLORER:** Shows files: index.html, JS main.js, task1.html, task2.html, task3.html.
- EDITOR:** Task3.html (HTML code):
 

```
<html lang="en">
<head>
</head>
<body>
  <div class="nav-bar"></div>
  <div id="app">
    <div class="product">
      <div class="product-image">
        <a :href="image" target="_blank">
          
        </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>
        <span v-show="onSale">On Sale</span>
      </div>
    </div>
  </div>
</body>
</html>
```
- JS main.js:**

```
Week3 > JS main.js > (el) app > data > onSale
var app = new Vue({
  ...
  data: {
    product: "Socks",
    description: "A pair of warm, fuzzy socks",
    image: "./assets/images/socks_green.jpg",
    altText: "A pair of socks",
    inStock: true,
    inventory: 100,
    onSale: false,
    // brand: "Vue Mastery",
    // selectedVariant: 0,
    // details: ["80% cotton", "20% polyester", "Gender-neutral"],
    // variants: [
    //   {
    //     variantId: 2234,
    //     variantColor: "green",
    //     variantImage: "./assets/images/socks_green.jpg",
    //     variantQuantity: 10,
    //   }
  }
})
```

## Lesson 4: List Rendering

Screenshot of a development environment showing a Vue.js application for a product page with list rendering.

**Browser View:**

- URL: 127.0.0.1:5500/Week3/task4.html
- Page Content: A green sock with a white mountain logo. Below it is the text "Socks" and "In Stock". Underneath is a bulleted list: "• 80% cotton", "• 20% polyester", and "• Gender-neuetral". Further down are the colors "green" and "blue".

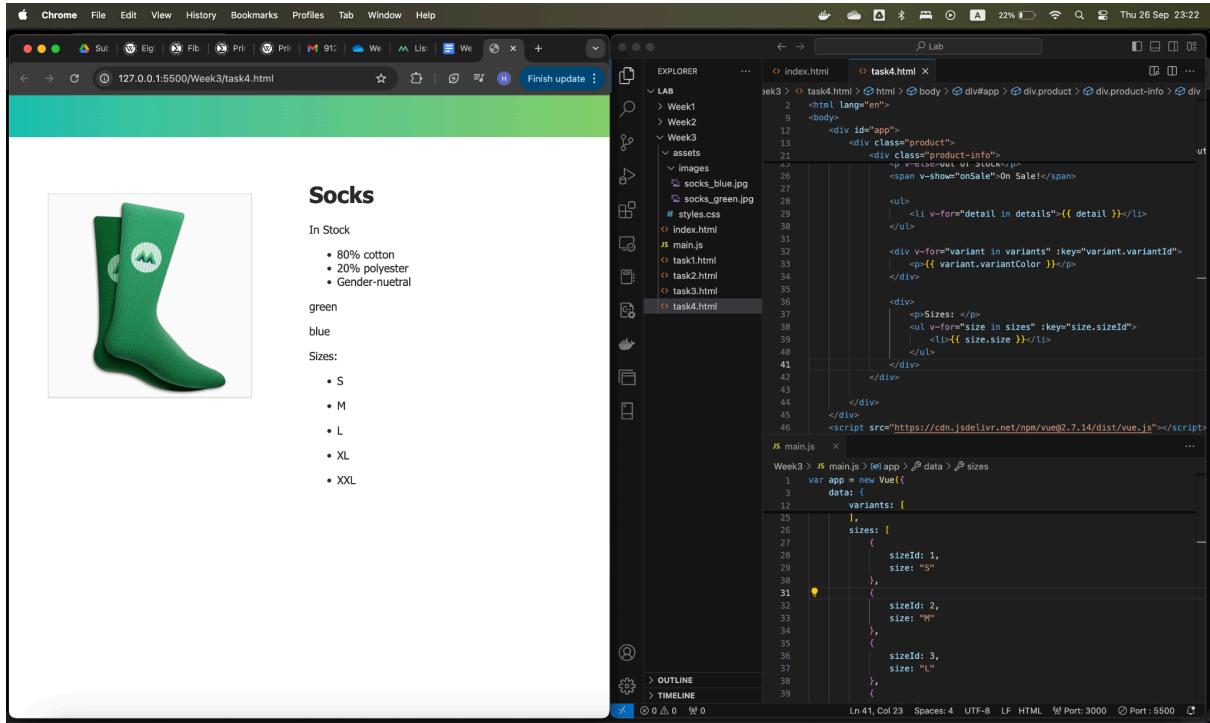
**Code Editor (VS Code):**

- EXPLORER:** Shows files: index.html, JS main.js, task1.html, task2.html, task3.html, task4.html.
- EDITOR:** Task4.html (HTML code):
 

```
<html lang="en">
<head>
</head>
<body>
  <div id="app">
    <div class="product">
      <div class="product-image">
        <a :href="image" target="_blank">
          
        </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>
        <span v-show="onSale">On Sale</span>
      </div>
      <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>
  </div>
</body>
</html>
```
- JS main.js:**

```
Week3 > JS main.js > (el) app > data > variants > variantColor
var app = new Vue({
  ...
  data: {
    ...
    variants: [
      {
        variantId: 2234,
        variantColor: "green",
        variantImage: "./assets/images/socks_green.jpg",
        variantQuantity: 10,
      },
      {
        variantId: 2235,
        variantColor: "blue",
        variantImage: "./assets/images/socks_blue.jpg",
        variantQuantity: 10,
      },
    ],
  }
})
```

## Challenge of lesson 4



The screenshot shows a Vue.js application running in a browser. The application displays a product page for socks. On the left, there is a large image of a green sock with a small logo on the toe. To the right of the image, the product name "Socks" is displayed, along with the text "In Stock". Below this, there is a bulleted list of product details: "• 80% cotton", "• 20% polyester", and "• Gender-neutral". Further down, there is a section titled "Sizes:" with a list of size options: "green", "blue", "S", "M", "L", "XL", and "XXL". The background of the application is light blue.

The browser's developer tools are open, showing the code editor with three files:

- index.html**: The main HTML file containing the structure of the page.
- task4.html**: A component file containing Vue.js template code for rendering the product details and sizes.
- main.js**: The script file containing the Vue.js instance configuration.

The code editor shows the following code for **task4.html**:

```
<div id="app">
  <div class="product">
    <div class="product-info">
      <p>On Sale</p>
      <span v-show="onSale">On Sale!</span>
    </div>
    <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>
      <p>Sizes: </p>
      <ul v-for="size in sizes" :key="size.sizeId">
        <li>({ size.size })</li>
      </ul>
    </div>
  </div>
</div>
```

The code editor shows the following code for **main.js**:

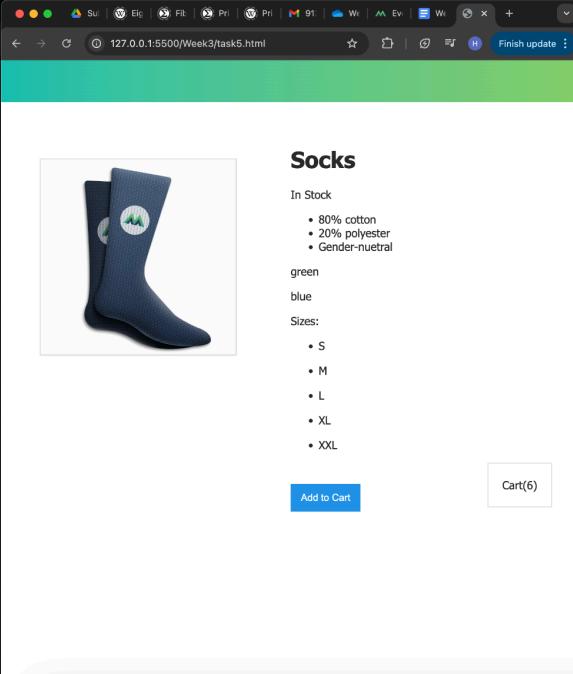
```
var app = new Vue({
  data: {
    variants: [
      {
        sizes: [
          {
            sizeId: 1,
            size: "S"
          },
          {
            sizeId: 2,
            size: "M"
          },
          {
            sizeId: 3,
            size: "L"
          }
        ]
      }
    ]
  }
})
```

## Evaluation

From lesson 1, I understood how to create a Vue instance for managing a Vue app and HTML elements rendered by the VueJS framework. In lesson 2, I have just learned how to bind data to html tags to create dynamic data to display on the html; in addition, Vue instance has a place to define data properties, which helps developers to manage data easily. For lesson 3, I understood how to use conditional statements to render HTML tags according to conditional logic. This statement can assist in rendering only necessary HTML tags for business logic instead of many HTML tags. Finally, list rendering is a helpful feature that helps developers create list data with a for loop instead of creating many duplicate HTML tags manually.

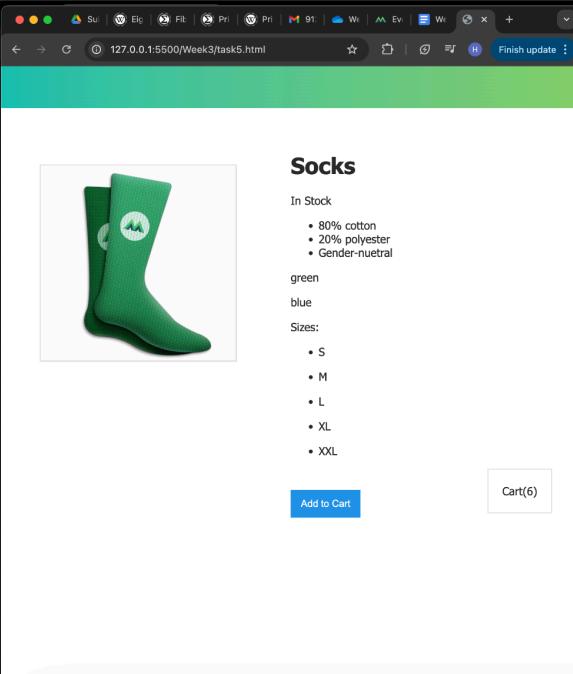
# Task 2 (Lesson 5-7)

## Lesson 5: Event Handling



The screenshot shows a web application interface. On the left, there is a product card for 'Socks'. The card features a blue sock with a green mountain logo. Below the image, the text 'In Stock' is followed by a bulleted list: '• 80% cotton', '• 20% polyester', and '• Gender-neutral'. A dropdown menu shows 'green' selected. The size options are 'S', 'M', 'L', 'XL', and 'XXL'. At the bottom of the card is a blue 'Add to Cart' button. To the right of the card, a sidebar displays the current inventory count as 'Cart(6)'. The background shows the code editor with files for 'index.html', 'task5.html', and 'main.js'.

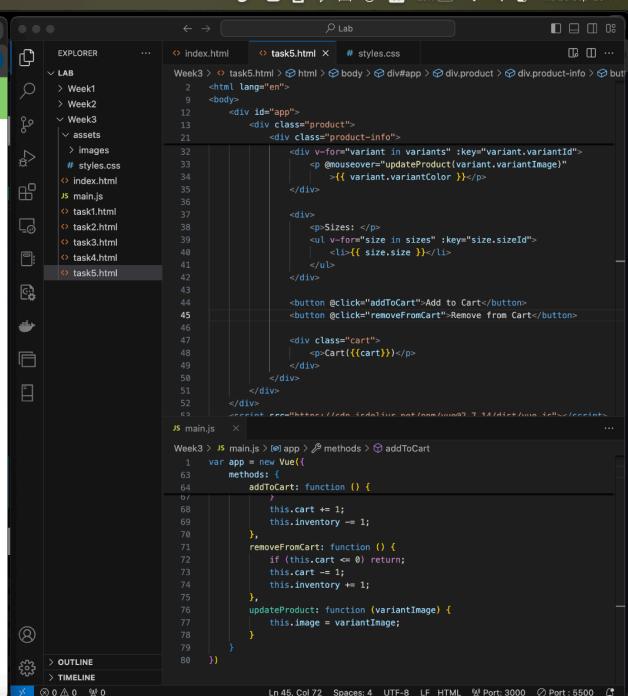
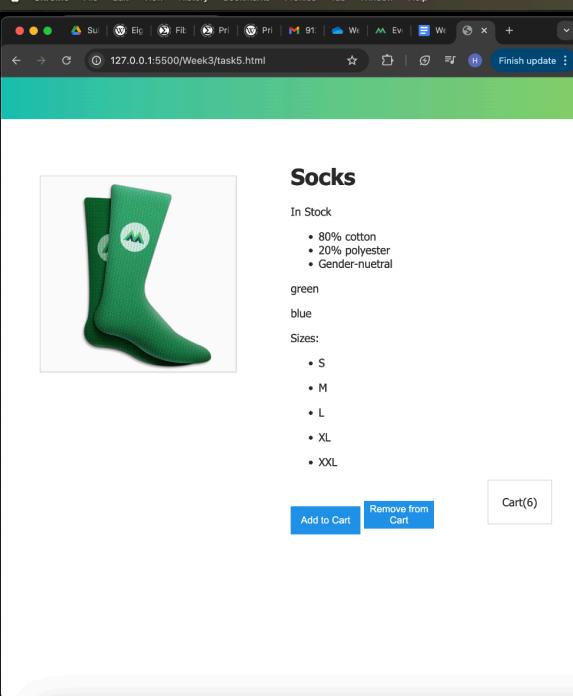
```
Week3 > task5.html > # body > div#app > div.product > div.product-info
Week3 > index.html > # body > div#app > div.product > div.product-info
Week3 > main.js > [el] app > <methods> > addToCart
1 var app = new Vue({
2   // ...
3   methods: {
4     addToCart: function () {
5       if (this.inventory <= 0) {
6         return
7       }
8       this.cart += 1;
9       this.inventory -= 1;
10    },
11    removeFromCart: function () {
12      if (this.cart == 0) return;
13      this.cart -= 1;
14      this.inventory += 1;
15    },
16    updateProduct: function (variantImage) {
17      this.image = variantImage;
18    }
19  }
20 }
21 
```



This screenshot shows the same web application as the previous one, but the socks are now green. The product card, inventory details, size dropdown, and 'Add to Cart' button are all present. The sidebar shows 'Cart(6)'.

```
Week3 > task5.html > # body > div#app > div.product > div.product-info
Week3 > index.html > # body > div#app > div.product > div.product-info
Week3 > main.js > [el] app > <methods> > addToCart
1 var app = new Vue({
2   // ...
3   methods: {
4     addToCart: function () {
5       if (this.inventory <= 0) {
6         return
7       }
8       this.cart += 1;
9       this.inventory -= 1;
10    },
11    removeFromCart: function () {
12      if (this.cart == 0) return;
13      this.cart -= 1;
14      this.inventory += 1;
15    },
16    updateProduct: function (variantImage) {
17      this.image = variantImage;
18    }
19  }
20 }
21 
```

## Challenge of lesson 5

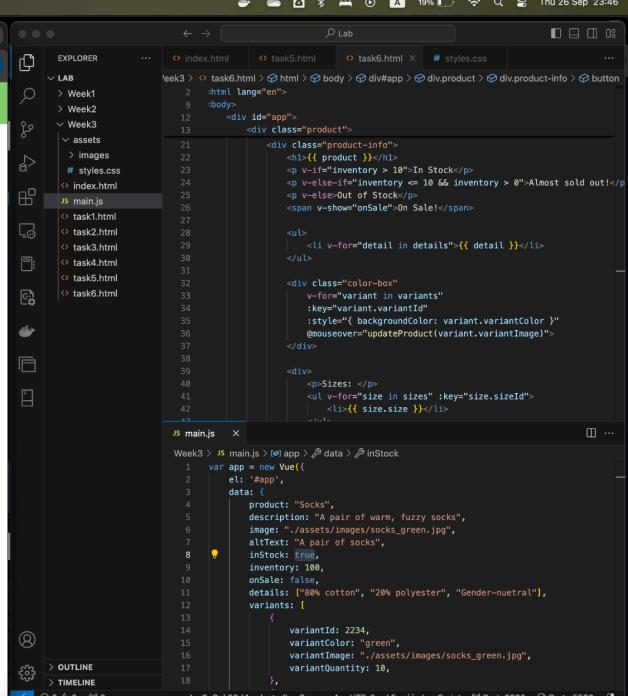
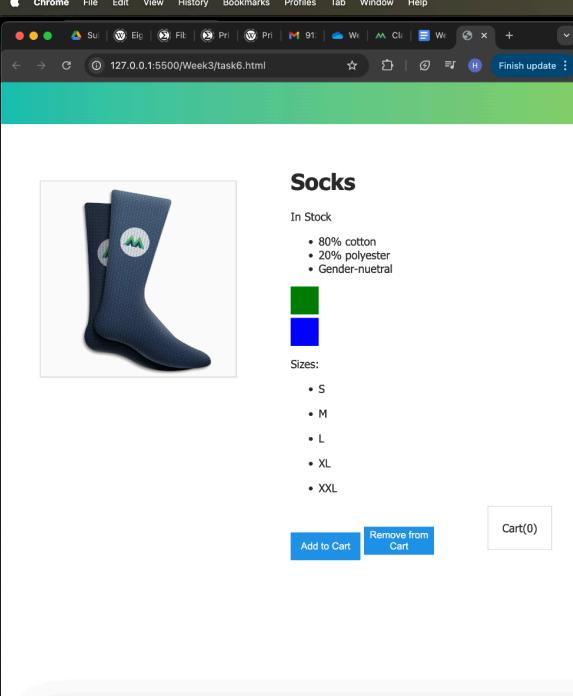


The screenshot shows a web browser displaying a product page for 'Socks'. The product image is a pair of green socks with a small logo on the toe. The description section says 'In Stock' and lists the following details:

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

The size selection section shows radio buttons for S, M, L, XL, and XXL. A 'Cart(6)' button is visible. The code editor on the right shows the generated HTML, CSS, and JavaScript for this page.

## Lesson 6: Class & Style Binding



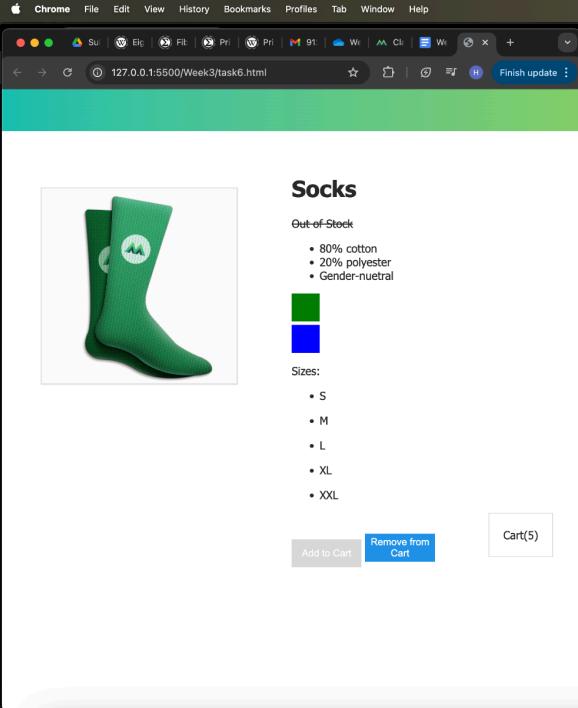
The screenshot shows a web browser displaying a product page for 'Socks'. The product image is a pair of blue socks with a small logo on the toe. The description section says 'In Stock' and lists the following details:

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

The size selection section shows radio buttons for S, M, L, XL, and XXL. A 'Cart(0)' button is visible. The code editor on the right shows the generated HTML, CSS, and JavaScript for this page, including class and style bindings.



## Challenge of lesson 6



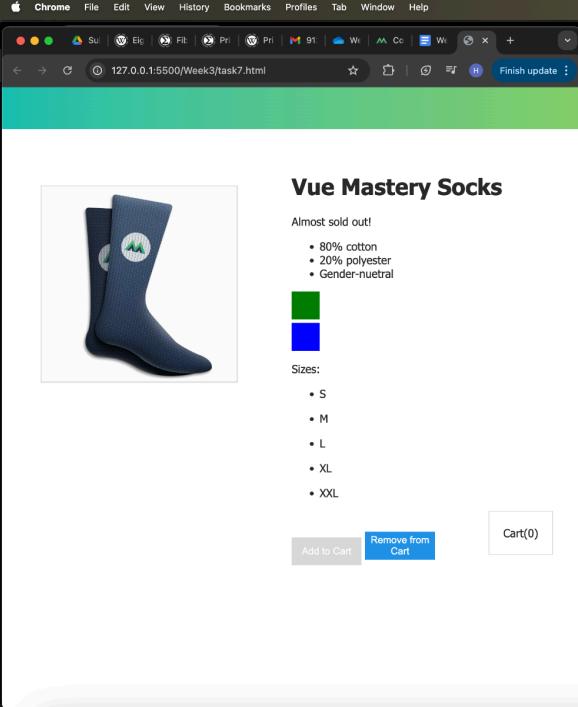
The screenshot shows a product page for 'Socks'. The page includes a large image of a green sock with a logo, a list of material details (80% cotton, 20% polyester, Gender-neutral), and a color swatch. Below the image is a list of sizes (S, M, L, XL, XXL). At the bottom, there are 'Add to Cart' and 'Remove from Cart' buttons, and a 'Cart(5)' button.

Code Editor View:

```
index.html
<div id="app">
  <div class="product">
    <div class="product-image">
      <a href="#" target="_blank">
        
      </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 :class="{ outOfStock: !inStock }">Out of Stock!</p>
      <span v-show="onSale">On Sale!</span>
    </div>
    <ul>
      <li v-for="detail in details">{{ detail }}</li>
    </ul>
    <div class="color-box">
      v-for="variant in variants"
      :key="variant.variantId"
      :style="{ backgroundColor: variant.variantColor }"
    </div>
  </div>
</div>
```

```
main.js
var app = new Vue({
  el: '#app',
  data: {
    cart: 0,
    inStock: true
  },
  methods: {
    addToCart: function () {
      if (this.inventory <= 0) {
        this.inStock = false;
        return;
      }
      this.cart += 1;
      this.inventory -= 1;
    },
    removeFromCart: function () {
      if (this.cart <= 0) return;
      this.cart -= 1;
      this.inventory += 1;
      if (this.inventory > 0) this.inStock = true;
    },
    updateProduct: function (variantImage) {
      this.image = variantImage;
    }
  }
})
```

## Lesson 7: Computed Properties



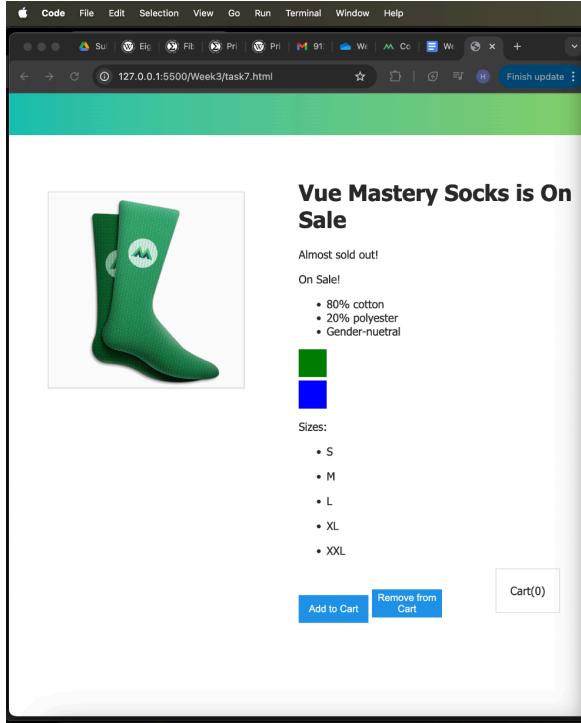
The screenshot shows a product page for 'Vue Mastery Socks'. The page includes a large image of a dark blue sock with a logo, a list of material details (80% cotton, 20% polyester, Gender-neutral), and a color swatch. Below the image is a list of sizes (S, M, L, XL, XXL). At the bottom, there are 'Add to Cart' and 'Remove from Cart' buttons, and a 'Cart(0)' button.

Code Editor View:

```
index.html
<div id="app">
  <div class="product">
    <div class="product-image">
      <img alt="Vue Mastery Socks" />
    </div>
    <div class="product-info">
      <h1>{{ title }}</h1>
      <p v-if="inventory > 10">In Stock</p>
      <p v-else-if="inventory <= 10 && inventory > 0">Almost sold out!</p>
      <p v-else :class="{ outOfStock: !inStock }">Out of Stock!</p>
      <span v-show="onSale">On Sale!</span>
    </div>
    <ul>
      <li v-for="detail in details">{{ detail }}</li>
    </ul>
    <div class="color-box">
      v-for="variant, index in variants"
      :key="variant.variantId"
      :style="{ backgroundColor: variant.variantColor }"
      @mouseover="updateProduct(index)"
    </div>
  </div>
</div>
```

```
main.js
var app = new Vue({
  el: '#app',
  data: {
    cart: 0,
    brand: "Vue Mastery",
    title: "Almost sold out!",
    inventory: 5,
    onSale: false,
    variants: [
      {
        variantId: 1,
        variantImage: "https://i.imgur.com/1QvZGfI.png",
        variantQuantity: 5
      }
    ],
    details: [
      "80% cotton",
      "20% polyester",
      "Gender-neutral"
    ]
  },
  computed: {
    title() {
      return this.onSale ? this.brand + " " + this.product + " is On Sale!" : this.title;
    },
    image() {
      return this.variants[this.selectedVariant].variantImage;
    },
    inStock() {
      return this.variants[this.selectedVariant].variantQuantity;
    }
  },
  methods: {
    addToCart() {
      this.cart++;
      this.inventory--;
    },
    removeFromCart() {
      this.cart--;
      this.inventory++;
    },
    updateProduct(index) {
      this.selectedVariant = index;
    }
  }
})
```

## Challenge of lesson 7



The screenshot shows a Vue.js application running in a browser. The page displays a green sock with a logo, the text "Vue Mastery Socks is On Sale", and a message "Almost sold out! On Sale!". It lists the following details:

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

A color swatch shows green and blue squares. Below the details, there's a size selection section with options: S, M, L, XL, XXL. At the bottom, there are "Add to Cart" and "Remove from Cart" buttons, and a "Cart(0)" button.

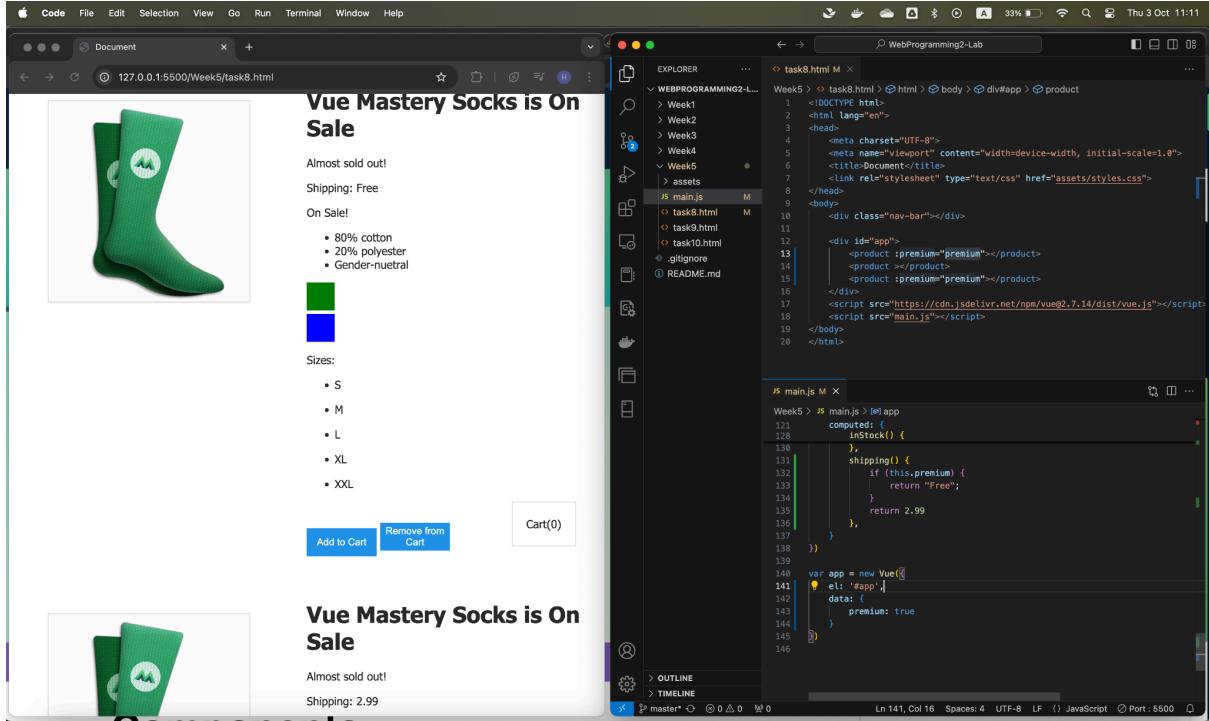
The right side of the image shows the code editor interface with two tabs: "index.html" and "main.js". The "index.html" tab shows the HTML structure of the page, including the product details and a "Cart" section. The "main.js" tab shows the JavaScript code defining the Vue instance and its data properties. A tooltip is visible over the "variant" variable in the code, showing its value: "uzzy socks", "inventory: 5", "onSale: true", "details: [80% cotton, 20% polyester, Gender-neutral]", and "variants: [ { variantId: 2234, variantColor: 'green', variantImage: '/assets/images/socks\_green.jpg', variantQuantity: 10, } ]".

## Evaluation

In lesson 5, I learned how to use event handling effectively with VueJS. With this mechanism, it helps me to manage events better. For example, I define an event listener on the HTML tags, then I create an event handler in the methods property of the Vue instance to handle incoming events from the HTML tags. It helps manage the event signal mechanism efficiently. Moreover, class and style binding help to add CSS dynamically following data. For example, I can change the background colour of the colour box following the data of socks. With the computed properties, it helps to create computed data based on other data dynamically; in addition, it can help to save memory in creating many variables to store data. Finally, lessons 5–7 help understand more about event handling in processing user interactions.

# Task 3 (Lesson 8 - 10)

## Lesson 8: Components



Vue Mastery Socks is On Sale

Almost sold out!

Shipping: Free

On Sale!

- 80% cotton
- 20% polyester
- Gender-neutra

Sizes:

- S
- M
- L
- XL
- XXL

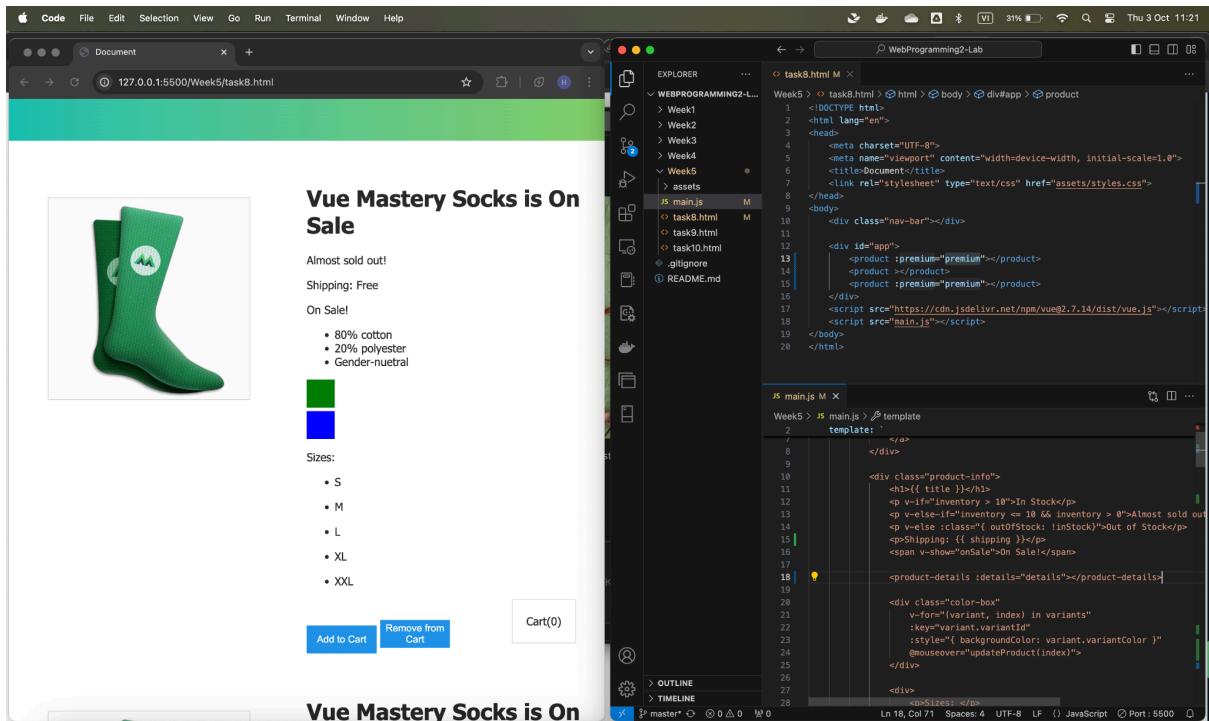
Add to Cart Remove from Cart Cart(0)

Vue Mastery Socks is On Sale

Almost sold out!

Shipping: 2.99

## Challenge of lesson 8



Vue Mastery Socks is On Sale

Almost sold out!

Shipping: Free

On Sale!

- 80% cotton
- 20% polyester
- Gender-neutra

Sizes:

- S
- M
- L
- XL
- XXL

Add to Cart Remove from Cart Cart(0)

Vue Mastery Socks is On Sale

# Evaluation

## Lesson 9: Communicating Event



The screenshot shows a web browser window with the URL `127.0.0.1:5500/Week5/task9.html`. The page displays a product listing for 'Vue Mastery Socks'. The title is 'Vue Mastery Socks is On Sale'. Below the title, it says 'Almost sold out!' and 'Shipping: Free'. A section titled 'On Sale!' lists the following details:

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

A small image of a green sock with a white 'M' logo is shown. Below the image, there are two color swatches: a green square and a blue square. The size options listed are S, M, L, XL, and XXL. At the bottom of the product card are two buttons: 'Add to Cart' and 'Remove from Cart'. To the right of the browser window is a code editor showing the source code for `task9.html` and `main.js`.

```
task9.html
<html lang="en">
  <head>
    <link href="#/static/styles.css" rel="stylesheet">
  </head>
  <body>
    <div id="app">
      <div class="cart">
        <p>Cart({ cart.length })</p>
      </div>
      <product>
        <img alt="Vue Mastery Sock" data-vv="image" data-vv-error="Image required" data-vv-validation="valid" data-vv-validation-error="" data-vv-validation-type="required"/>
        <div>
          <h3>Vue Mastery Socks is On Sale</h3>
          <div>Almost sold out!</div>
          <div>Shipping: Free</div>
          <div>On Sale!</div>
          <ul>
            <li>• 80% cotton</li>
            <li>• 20% polyester</li>
            <li>• Gender-neutral</li>
          </ul>
          <div>
            <img alt="Green Sock" data-vv="color" data-vv-error="Color required" data-vv-validation="valid" data-vv-validation-error="" data-vv-validation-type="required"/>
            <img alt="Blue Sock" data-vv="color" data-vv-error="Color required" data-vv-validation="valid" data-vv-validation-error="" data-vv-validation-type="required"/>
          </div>
          <div>Sizes:</div>
          <ul>
            <li>• S</li>
            <li>• M</li>
            <li>• L</li>
            <li>• XL</li>
            <li>• XXL</li>
          </ul>
          <div>
            <button>Add to Cart</button>
            <button>Remove from Cart</button>
          </div>
        </div>
      </product>
    </div>
  </body>
</html>
```

```
main.js
var app = new Vue({
  el: '#app',
  data: {
    premium: true,
    cart: []
  },
  methods: {
    updateCart(id) {
      if (this.inventory <= 0) {
        return
      }
      this.cart.push(id);
      this.inventory -= 1;
    }
  }
})
```

## Challenge of lesson 9:



The screenshot shows a web browser window with the URL `127.0.0.1:5500/Week5/task9.html`. The page displays a product listing for 'Vue Mastery Socks'. The title is 'Vue Mastery Socks is On Sale'. Below the title, it says 'Almost sold out!' and 'Shipping: Free'. A section titled 'On Sale!' lists the following details:

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

A small image of a green sock with a white 'M' logo is shown. Below the image, there are two color swatches: a green square and a blue square. The size options listed are S, M, L, XL, and XXL. At the bottom of the product card are two buttons: 'Add to Cart' and 'Remove from Cart'. To the right of the browser window is a code editor showing the source code for `task9.html` and `main.js`.

```
task9.html
<html lang="en">
  <head>
    <link href="#/static/styles.css" rel="stylesheet">
  </head>
  <body>
    <div id="app">
      <div class="cart">
        <p>Cart(2)</p>
      </div>
      <product>
        <img alt="Vue Mastery Sock" data-vv="image" data-vv-error="Image required" data-vv-validation="valid" data-vv-validation-error="" data-vv-validation-type="required"/>
        <div>
          <h3>Vue Mastery Socks is On Sale</h3>
          <div>Almost sold out!</div>
          <div>Shipping: Free</div>
          <div>On Sale!</div>
          <ul>
            <li>• 80% cotton</li>
            <li>• 20% polyester</li>
            <li>• Gender-neutral</li>
          </ul>
          <div>
            <img alt="Green Sock" data-vv="color" data-vv-error="Color required" data-vv-validation="valid" data-vv-validation-error="" data-vv-validation-type="required"/>
            <img alt="Blue Sock" data-vv="color" data-vv-error="Color required" data-vv-validation="valid" data-vv-validation-error="" data-vv-validation-type="required"/>
          </div>
          <div>Sizes:</div>
          <ul>
            <li>• S</li>
            <li>• M</li>
            <li>• L</li>
            <li>• XL</li>
            <li>• XXL</li>
          </ul>
          <div>
            <button>Add to Cart</button>
            <button>Remove from Cart</button>
          </div>
        </div>
      </product>
    </div>
  </body>
</html>
```

```
main.js
var app = new Vue({
  el: '#app',
  data: {
    premium: true,
    cart: [
      {id: "2234", quantity: 1, add: "updateCart", remove: "removeCart"}, 
      {id: "2235", quantity: 1, add: "updateCart", remove: "removeCart"}]
  },
  methods: {
    updateCart(id) {
      console.log(this.cart);
      this.cart.push(id);
      this.inventory -= 1;
    },
    removeCart(id) {
      if (this.cart <= 0) return;
      while (this.cart.indexOf(id) !== -1) {
        this.cart.splice(this.cart.indexOf(id), 1);
      }
      this.inventory += 1;
    }
  }
})
```

## Lesson 10: Forms & v-model

The screenshot shows a web browser window with a product page and an adjacent code editor.

**Product Page:**

- Image: A green shoe.
- Text: "80% cotton", "20% polyester", "Gender-neutral".
- Color swatches: Green and blue.
- Text: "Sizes:"
  - S
  - M
  - L
  - XL
  - XXL
- Buttons: "Add to Cart" and "Remove from Cart".

**Reviews Section:**

- Name: Tuyễn
- Rating: 4
- Review: as

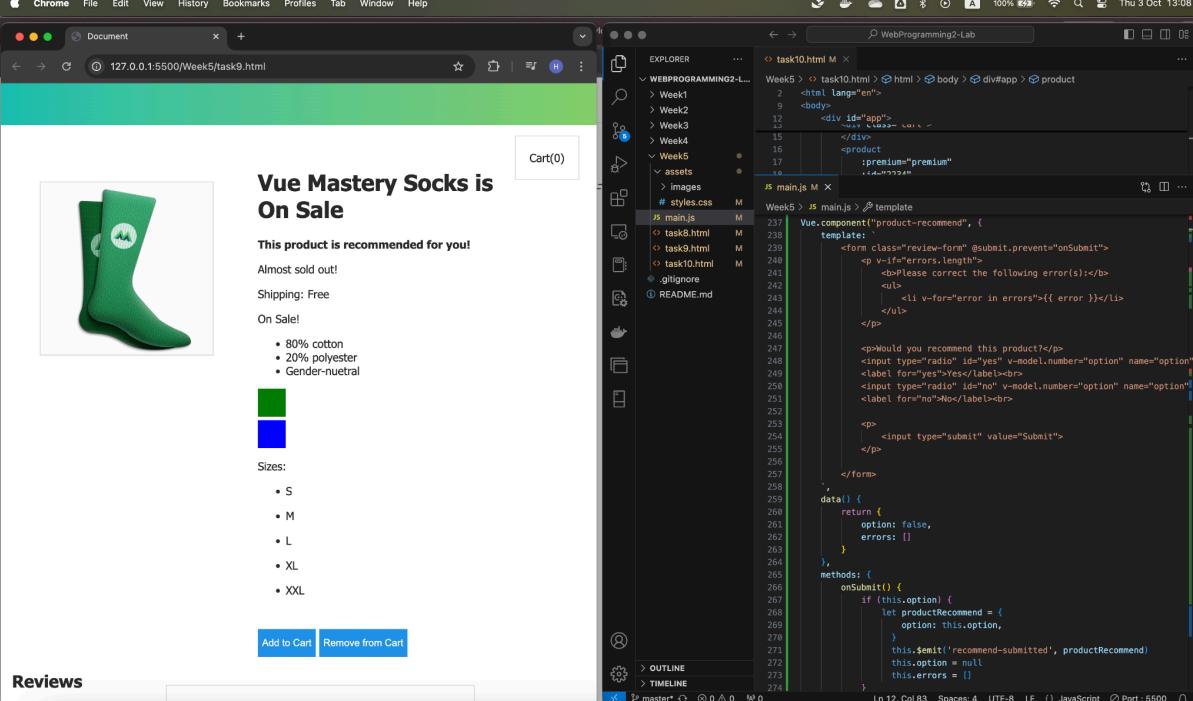
**Code Editor (task10.html):**

```
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>WebProgramming2-Lab</title>
    <link href="https://cdn.jsdelivr.net/npm/vuetify@2.7.14/dist/vuetify.min.css" rel="stylesheet" />
    <script src="https://cdn.jsdelivr.net/npm/vue@2.7.14/dist/vue.js"></script>
    <script src="main.js"></script>
  </head>
  <body>
    <div id="app">
      <product>
        <!-- Premium Product -->
        <!-- Premium Product -->
        <!-- Premium Product -->
      </product>
    </div>
  </body>
</html>
```

**Code Editor (main.js):**

```
methods: {
  onSubmit() {
    if (!this.name || !this.review || !this.rating) {
      let productReview = {
        name: this.name,
        review: this.review,
        rating: this.rating
      }
      this.$emit('review-submitted', productReview)
      this.name = null
      this.review = null
      this.rating = null
      this.errors = []
    } else {
      if (!this.name) this.errors.push("Name required.");
      if (!this.review) this.errors.push("Review required.");
      if (!this.rating) this.errors.push("Rating required.");
    }
  }
},
```

# Challenge of lesson 10

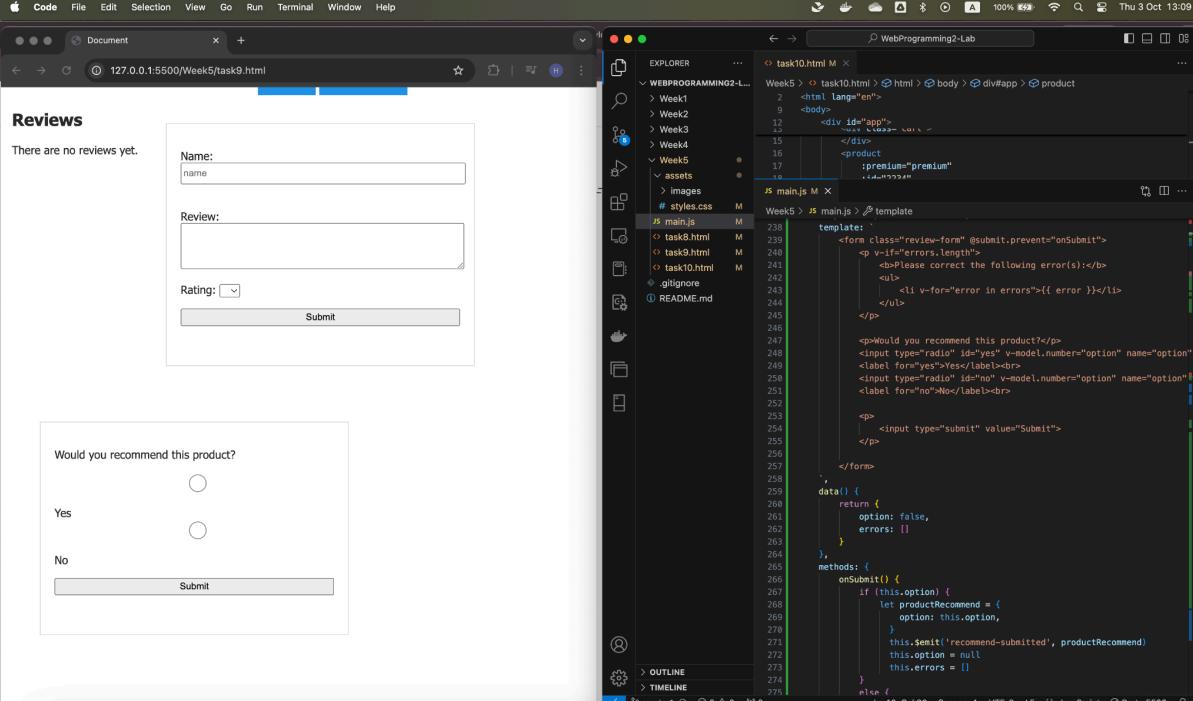


The screenshot shows a web browser window with the URL `127.0.0.1:5500/Week5/task9.html`. The page displays a product listing for 'Vue Mastery Socks is On Sale'. It features a green sock with a mountain logo, a short description, and a 'Cart(0)' button. Below the product image is a section titled 'This product is recommended for you!' followed by a note 'Almost sold out!', 'Shipping: Free', and 'On Sale!'. A bulleted list indicates the product is 80% cotton, 20% polyester, and gender-neutral. There are small green and blue squares representing color swatches. A 'Sizes:' section lists sizes S, M, L, XL, and XXL. At the bottom are 'Add to Cart' and 'Remove from Cart' buttons. The browser's developer tools are open, showing the project structure in the Explorer panel and the main.js file in the code editor. The main.js file contains Vue component logic for a 'product-recommend' component.

```
EXPLORER task10.html M
Week5 > task10.html > html > body > div#app > product
> Week1
> Week2
> Week3
> Week4
> Week5
> assets
> images
# styles.css M
JS main.js M
<html lang="en">
  <body>
    <div id="app">
      <product>
        <product>
          :premium="premium"
        </product>
      </product>
    </div>
  </body>
</html>
```

```
task10.html M
<html lang="en">
  <body>
    <div id="app">
      <product>
        <product>
          :premium="premium"
        </product>
      </product>
    </div>
  </body>
</html>
```

```
JS main.js M
237   vue.component('product-recommend', {
238     template:
239       <form class="review-form" @submit.prevent="onSubmit">
240         <p v-if="errors.length">
241           <b>Please correct the following error(s):</b>
242           <ul>
243             <li v-for="error in errors">{{ error }}</li>
244           </ul>
245         </p>
246         <p>Would you recommend this product?</p>
247         <input type="radio" id="yes" v-model.number="option" name="option">
248         <label for="yes">Yes</label><br>
249         <input type="radio" id="no" v-model.number="option" name="option">
250         <label for="no">No</label><br>
251
252         <p>
253           <input type="submit" value="Submit">
254         </p>
255       </form>
256     },
257     data() {
258       return {
259         option: false,
260         errors: []
261       }
262     },
263     methods: {
264       onSubmit() {
265         if (this.option) {
266           let productRecommend = {
267             option: this.option,
268           }
269           this.$emit('recommend-submitted', productRecommend)
270           this.option = null
271           this.errors = []
272         }
273       }
274     }
275   },
```



The screenshot shows a web browser window with the URL `127.0.0.1:5500/Week5/task9.html`. The page displays a 'Reviews' section with a form for leaving a review. It includes fields for 'Name', 'Review', 'Rating' (with a dropdown menu), and a 'Submit' button. Below the form is a question 'Would you recommend this product?' with 'Yes' and 'No' radio buttons. The browser's developer tools are open, showing the project structure in the Explorer panel and the main.js file in the code editor. The main.js file contains Vue component logic for a 'product-recommend' component.

```
EXPLORER task10.html M
Week5 > task10.html > html > body > div#app > product
> Week1
> Week2
> Week3
> Week4
> Week5
> assets
> images
# styles.css M
JS main.js M
<html lang="en">
  <body>
    <div id="app">
      <product>
        <product>
          :premium="premium"
        </product>
      </product>
    </div>
  </body>
</html>
```

```
task10.html M
<html lang="en">
  <body>
    <div id="app">
      <product>
        <product>
          :premium="premium"
        </product>
      </product>
    </div>
  </body>
</html>
```

```
JS main.js M
237   vue.component('product-recommend', {
238     template:
239       <form class="review-form" @submit.prevent="onSubmit">
240         <p v-if="errors.length">
241           <b>Please correct the following error(s):</b>
242           <ul>
243             <li v-for="error in errors">{{ error }}</li>
244           </ul>
245         </p>
246         <p>Would you recommend this product?</p>
247         <input type="radio" id="yes" v-model.number="option" name="option">
248         <label for="yes">Yes</label><br>
249         <input type="radio" id="no" v-model.number="option" name="option">
250         <label for="no">No</label><br>
251
252         <p>
253           <input type="submit" value="Submit">
254         </p>
255       </form>
256     },
257     data() {
258       return {
259         option: false,
260         errors: []
261       }
262     },
263     methods: {
264       onSubmit() {
265         if (this.option) {
266           let productRecommend = {
267             option: this.option,
268           }
269           this.$emit('recommend-submitted', productRecommend)
270           this.option = null
271           this.errors = []
272         }
273       }
274     }
275   },
```

## Evaluation

In lesson 8, I learned how to create and use Vue components. Components help break down the user interface into smaller pieces, making the app more manageable. For example, I can pass data to these

components using props, allowing different parts of the app to communicate with each other. This helps manage the UIs by dividing them into sections with their own functionality. In lesson 9, I learned about communicating events between components. I used \$emit to send custom events from child to parent components, allowing the parent to listen for these events and respond. In lesson 10, I learned how to work with forms in Vue using the v-model directive for two-way data binding. For example, I could bind input fields directly to data in the Vue instance. Additionally, I learned how to handle form submissions and perform validation making data management more efficient when processing user inputs. These lessons deepen my understanding of building interactive Vue.js applications with components, communication between components, and handling user input.

## Tutorial 8 - Week 6

### Task 2

#### End of lesson 4 exercise 1

The screenshot shows the Studio 3T interface for MongoDB. The top navigation bar includes: Studio 3T, File, Edit, Database, Collection, Index, Document, GridFS, View, Help. Below the bar are various tool icons: Connect, Collection, IntellisShell, SQL, Aggregate, Query Profiler, Compare, Schema, Reschema, Tasks, Export, Import, Data Masking, SQL Migration, Users, Roles, Feedback, and Go to Free. A message at the top states: "Enjoy a 30-day full product trial. Once the trial is complete, you will be switched to Studio 3T Free or you can switch now by disabling the trial in My License (in the Help Menu)." The main workspace shows a tree view of databases and collections under "Open connections". The "welsh\_pubs" collection is selected, displaying its documents. The results table shows fields: \_id, fsa\_id, name, address, and location. The table lists numerous documents with IDs like 5d34c59c098cc, 523718, 522955, etc. To the right of the results table is a "Visual Query Builder" panel with sections for Match all (\$and), Projection, and Sort.

## End of lesson 4 exercise 3

The screenshot shows the Studio 3T interface for MongoDB. On the left, the sidebar displays the connection to 'atlas-1166r6-shard-0' and the 'welsh\_pubs' collection. The main area shows a table of documents with columns: \_id, fsa\_id, name, and local\_authority. A large portion of the screen is occupied by the 'Visual Query Builder' panel on the right, which contains a query editor with the following code:

```
Query v { "name": {"$ne": "Caerphilly"} }
Projection { "fsa_id": 1, "name": 1, "city": 1, "local_authority": 1 }
Sort { "address.city": 1 }
```

The 'Visual Query Builder' panel also includes sections for 'Match all (\$and)' and 'Projection' with checkboxes for 'fsa\_id', 'name', 'city', and 'local\_authority'.

## End of lesson 5 exercise 3

The screenshot shows the Studio 3T interface for MongoDB. The left sidebar shows the same connection and collection as the previous screenshot. The main area now displays an aggregation pipeline in the 'Pipeline' tab:

```
Pipeline + Add stage × Stage 1: $match
Stage 2: $group
Stage 3: $project
Stage 4: $sort
```

The 'Stage 3: \$project' stage has the following code highlighted:

```
1 { "local_authority": "$_id.local_authority",
  2   "amount": "SCOUNT(*)",
  3   "_id": NumberInt(0)
  4 }
```

Below the pipeline, the 'Pipeline Output' tab shows the resulting data in a table:

local_authority	amount
Ipwys	9
Vale of Glamorgan	7
Monmouthshire	5
Carmarthenshire	3
Pembrokeshire	3
Bridgend	3
Wrexham	3

The bottom status bar indicates '1 document selected'.

## End of lesson 6 exercise 2

```
5     "address" : {
6         "Region" : "Scotland",
7         "address" : "10-12 Castle Street, Forfar",
8         "city" : "Kirriemuir"
9     },
10    "fsa_id" : NumberInt(450355),
11    "local_authority" : "Angus",
12    "location" : {
13        "latitude" : "56.644842",
14        "longitude" : "-2.888037"
15    },
16    "name" : "10 Cafe Bar"
17},
18{
19    "_id" : ObjectId("5d684ad118b29c04ecba9161"),
20    "address" : {
21        "Postcode" : "DG1 2BY",
22        "Region" : "Scotland",
23        "address" : "102 English Street, Dumfries, Dumfriesshire",
24        "city" : "Dumfries"
25    },
26    "contact" : {
27        "phone" : "01387 252654"
28    },
29    "fsa_id" : NumberInt(453070),
30    "local_authority" : "Dumfries and Galloway",
31    "location" : {
32        "latitude" : "55.868668",
33        "longitude" : "-3.608535"
34    },
35    "name" : "102 Club"
36},
37{
38    "_id" : ObjectId("5d684ad118b29c04ecba9162"),
39    "address" : {
40        "Postcode" : "KA9 2JT",
41        "Region" : "Scotland",
42        "address" : "43 Berelands Road, Prestwick South Ayrshire",
43        "city" : "Prestwick"
44    },
45    "fsa_id" : NumberInt(488728),
46    "local_authority" : "South Ayrshire",
47    "location" : {
48        "latitude" : "55.5006",
49        "longitude" : "-4.602813"
50    },
51    "name" : "1860 Club"
52}
```

## Conclusion

In the fourth session, MongoDB and Mongo Atlas offered a cloud database that was straightforward to use and study for novice users. Additionally, it offers security configurations like network and database access. In addition, the tutorial offers fundamental understanding of database, collection, and document management. In addition, MongoDB offers a solid query build that includes software development kits for Python, Node.js, Ruby, and several additional languages, as well as SQL statements like SQL databases and mongoshells. Lastly, it shares a significant benefit with NoSQL in that collections may be created dynamically without a database schema being predefined.

MongoDB offers a wide range of capabilities for aggregating data across multiple programming languages, as demonstrated in lesson 5. Furthermore, Studio3T offers a visual aggregate editor for data analysis. Because Mongo offers a pipeline for aggregating data, developers are able to set up a comprehensive pipeline that includes query, group, project, and sorting capabilities.

MongoDB offers strong features in lesson 6 that make it simple to import and export data from data sources. Developers may effortlessly import data into collections using sizable CSV and JSON files. Furthermore, Mongo makes data exporting effortless. Instead of exporting all of the data from the database, developers can export data from queries using studio3T.

# Appendix

## Git Link

<https://github.com/huynhthanhtuyen456/WebProgramming2-Lab>