Vue Instance

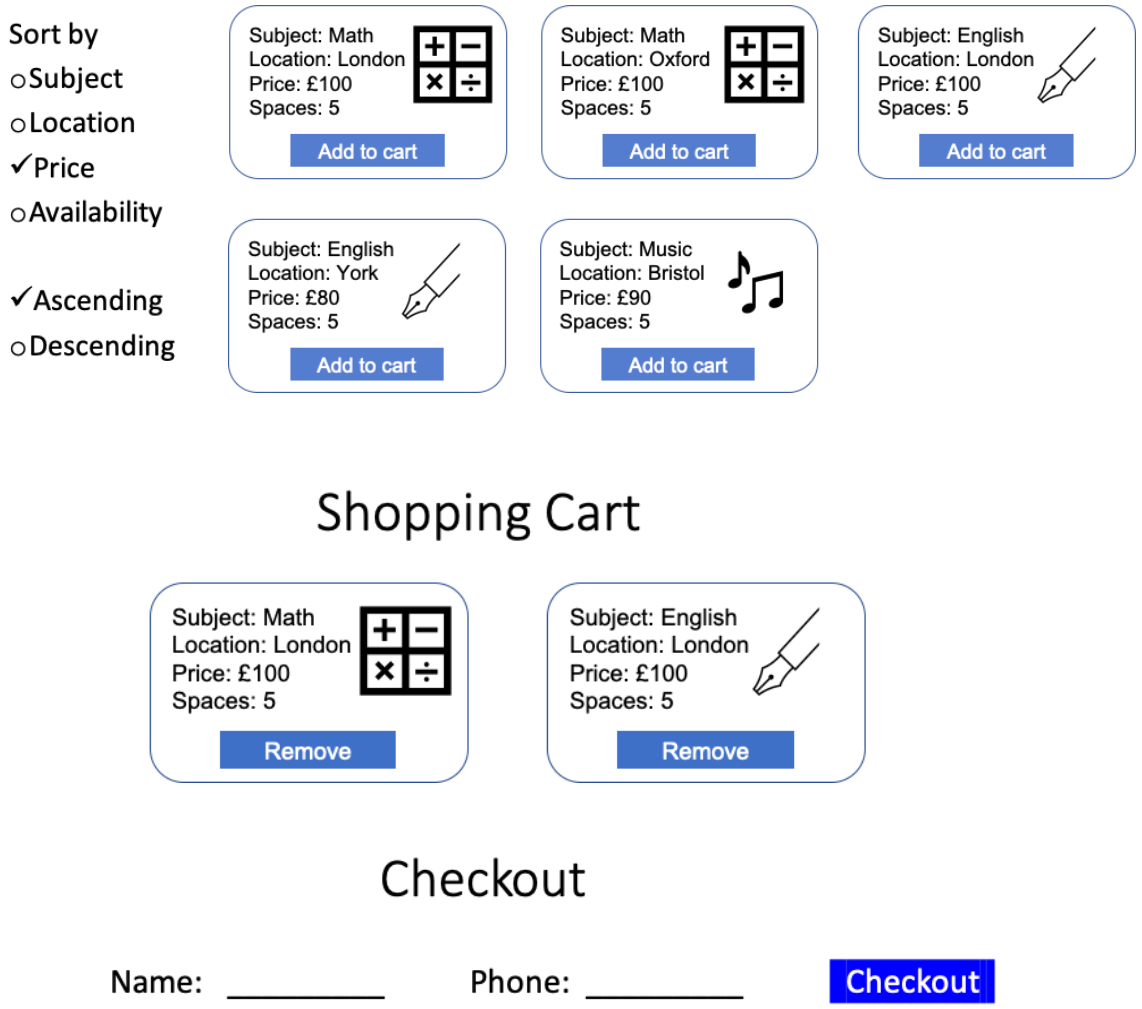
Outline and Learning Objectives

- **CourseWork 1 (CW1) Requirements:**
  - to understand CW1 Requirements
- **Introducing Vue Instance:**
  - To understand the **structure** of the Vue Instance
  - To **create and manage** a Vue Instance
- **Inspecting and Debugging Vue.js apps:**
  - to **inspect** Vue.js with the **console** and **vue-devtools** extension of **Chrome**
  - to **debug** Vue.js with Visual Studio Code
- **Creating Simple Web Pages with Vue.js:**
  - To **add data** to a Vue Instance and **binding data** to web page elements
  - **[Example]** To display one product in a web page with Vue.js
- **Suggestions for Reading**

CourseWork 1 (CW1) Requirements

CW1 Requirements

- **After School Class App**
- **Let's open the module page together:**
  - Handbook



Introducing: Vue Instance

Approach: Project-Based

- We will follow the **example in the textbook** 'Vue.js in Action';
- Build an **online pet accessory shop 'Pet Depot'**
- This is **similar to the after-school class app** for the **coursework**
- The source code is in GitHub: link

The First Step of the “Pet Depot”

```
<html>
    <head>
        <title>Vue.js Pet Depot</title>
        // Load Vue.js from a server (so no local file is needed)
        <script src="https://unpkg.com/vue@2.7.8/dist/vue.js"></script>
    </head>
    <body>
        <!-- The element where Vue will mount -->
        <div id="app"></div>
        <script type="text/javascript">
            let webstore = new Vue({ // The Vue instance
                el: '#app' // The 'option object': DOM mounting point
            });
        </script>
    </body>
</html>
```

The root Vue Instance

- Vue constructor **new Vue()**
- Vue Instance and option object: **new Vue({ /* option object goes here */ })**
  - The **el** option, inside the option object, is used to specify a **Document Object Model (DOM) element** (hence '**el**') where Vue will mount our application.
- The 'option object' is a standard JavaScript (JSON) object
  - **{key1: value1, key2: value2, ...}**
- The 'option object' has several pre-defined 'keys' such as **el, data, and methods**.

The el Property: Mounting Point Selector

- **Any CSS selector** can be used to reference the **mounting point**: element, class, ID, etc.

- If the CSS selector resolves to more than one DOM element, Vue will mount to the **first** matching element.
    - If we invoke the Vue constructor as **new Vue({el: 'div' })**, Vue would mount at **the first div element**.

- If you need to run **multiple Vue instances** on a single page, you could **mount them to different DOM elements** by using unique selectors.

Multiple Vue Instances on One Page

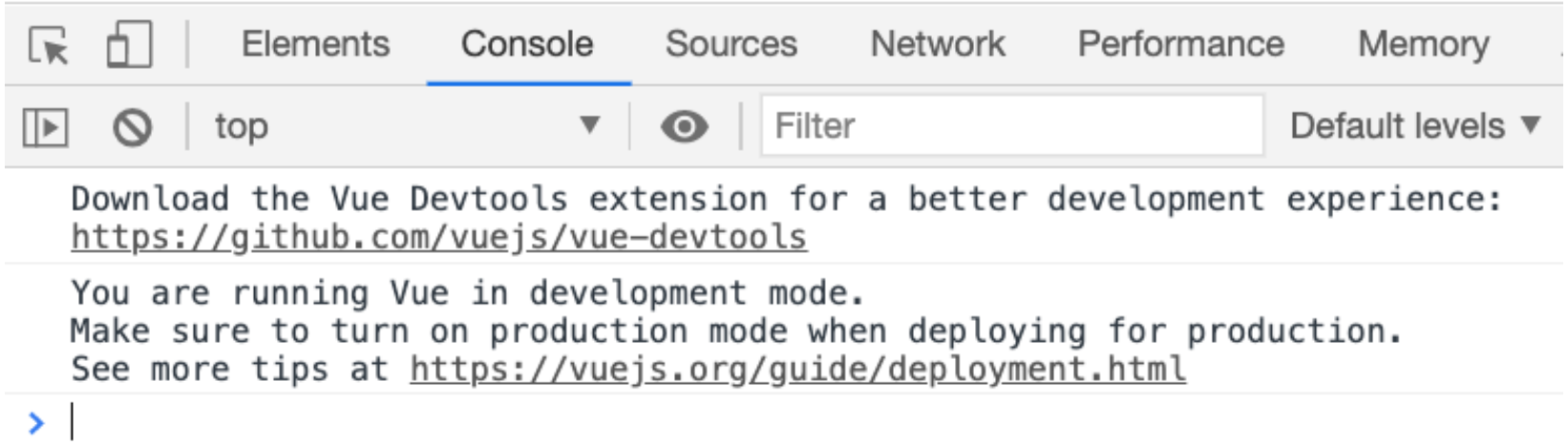
```
<body>
        <div id='app1'></div>
        <div id='app2'></div>
        <div id='app3'></div>

        <script>
                let app1 = new Vue ({el: '#app1', ...})
                let app2 = new Vue ({el: '#app2', ...})
                let app3 = new Vue ({el: '#app3', ...})
        </script>
</body>
```

Inspecting and Debugging Vue.js Apps

Inspecting Vue.js with the Console

- Make sure the Vue app is running
- Open the current 'Pet Depot' app in a browser should show nothing
    - **There is no visible HTML yet**

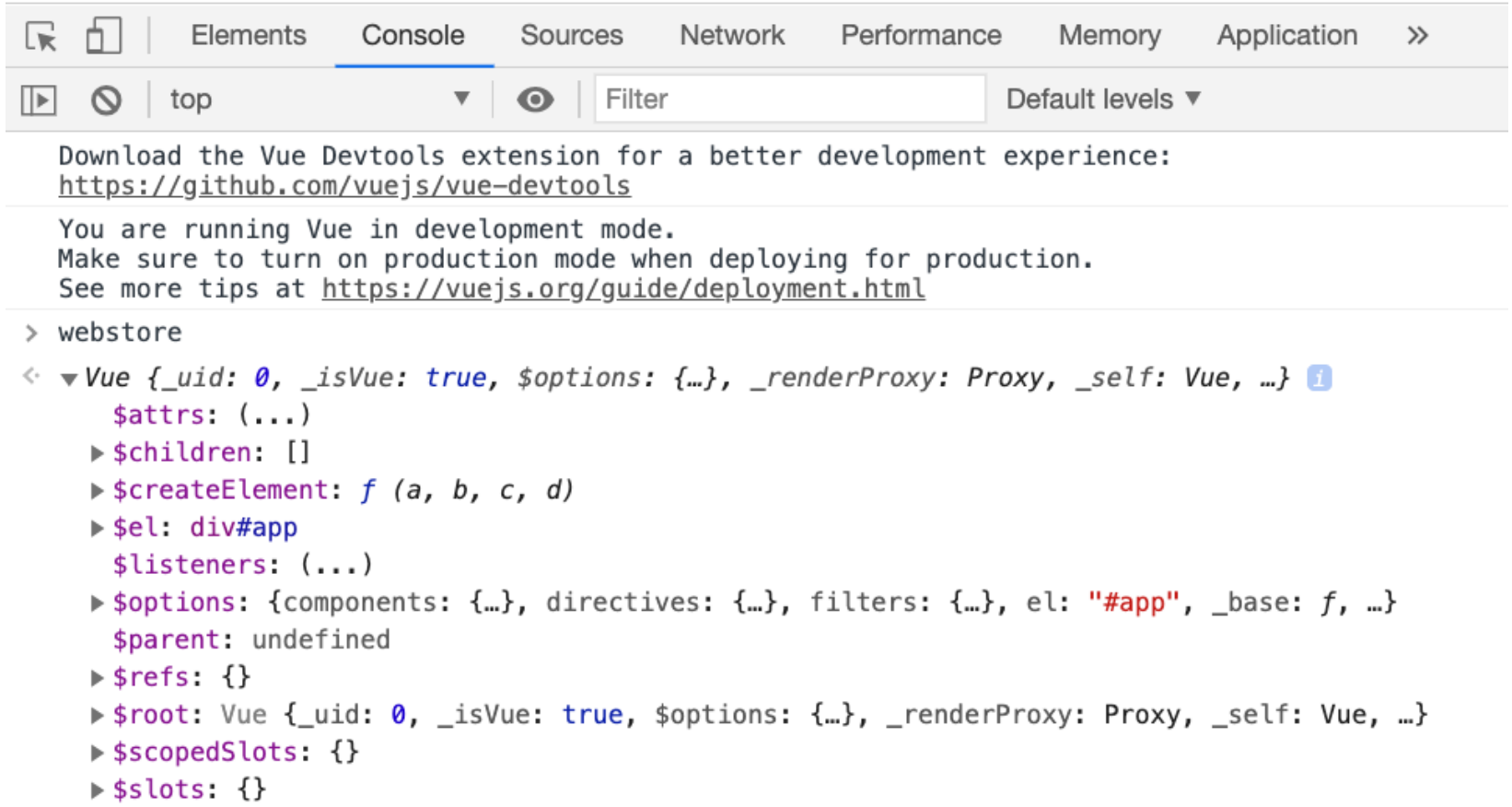
- However, in the browser console (developer tools), there should be message like this:



- This means Vue is running without errors

Using the Console with Vue Variables

- We can check that the Vue instance is using the variable **webstore** in the browser console



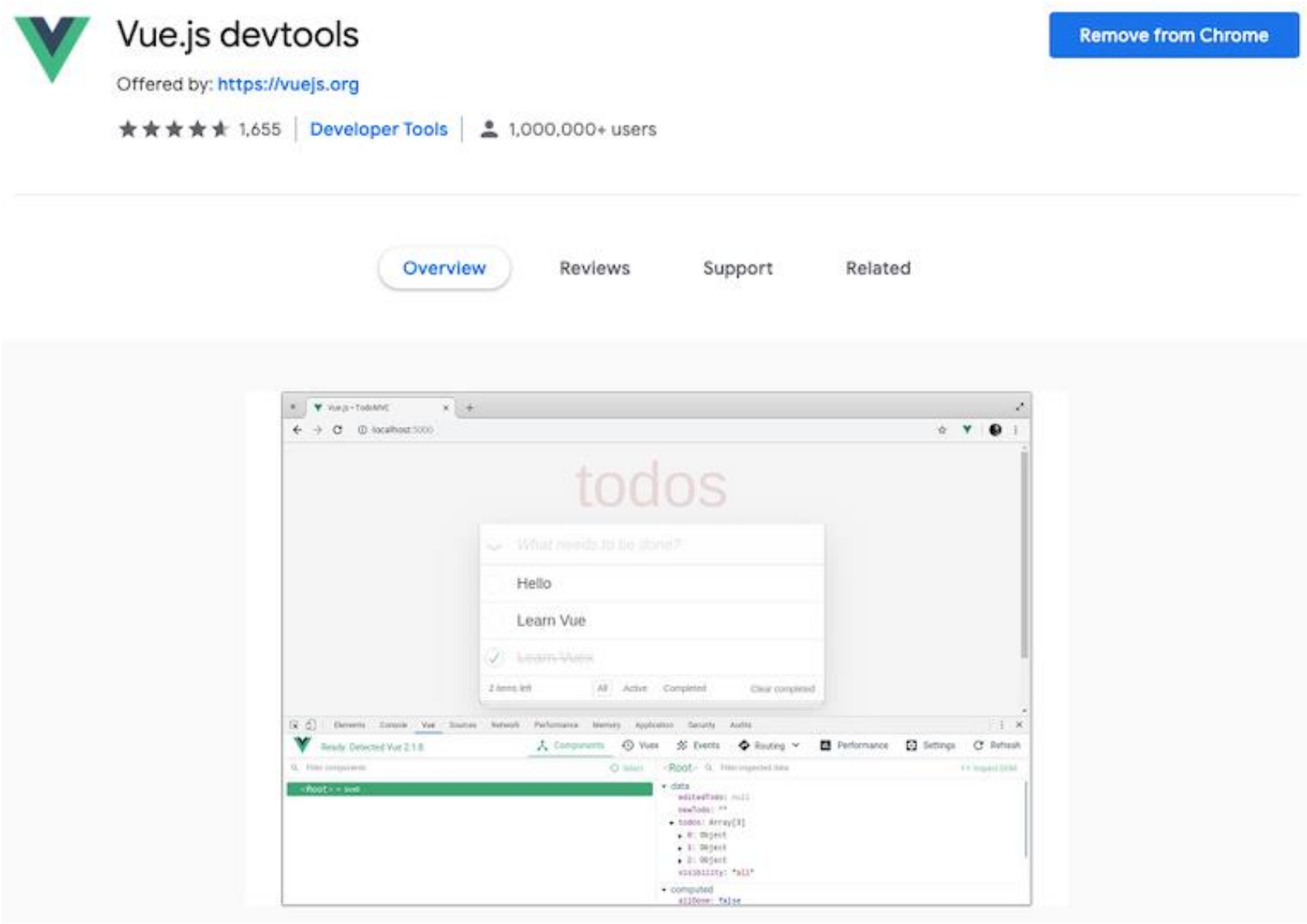
The vue-devtools Extension for Chrome (1)

As suggested in the console message,

vue-devtools (a browser extension)

can help to debug Vue.js code:

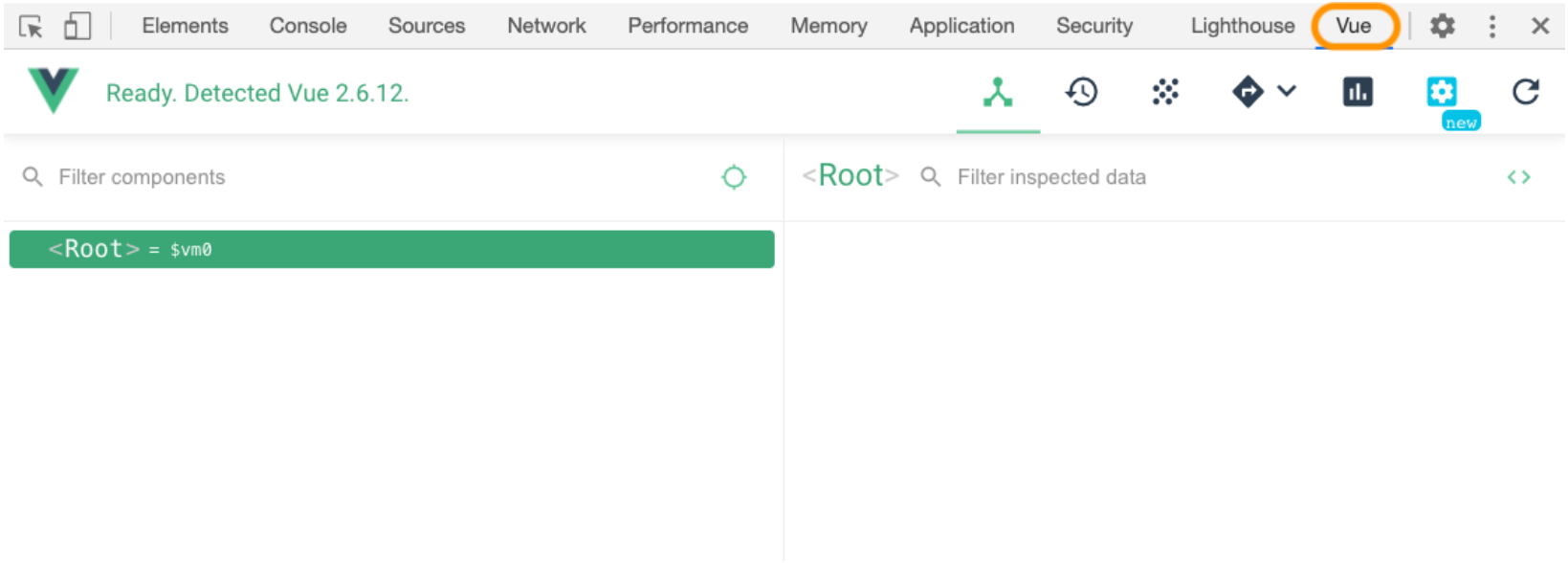
- Go to Google Chrome Store
- Search for vue.js
- Install “Vue.js devtools”
- Direct link to the extension



The vue-devtools Extension for Chrome (2)

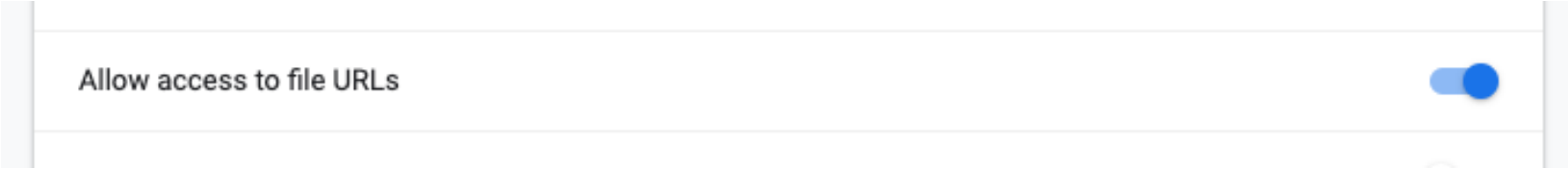
- Once installed, open your Vue.js app, click on the 3 dots (top bar, on the right) -> “More Tools” -> “Developer Tools”

- Then, look for the 'Vue' panel, not the 'console'



The vue-devtools Extension for Chrome (3)

- If this does not work, make sure “Allow access to file URLs” is enabled in the extension settings:
  - go to Chrome Extensions
  - go to the settings of the dev-tools extension
  - enable “Allow access to file URLs”

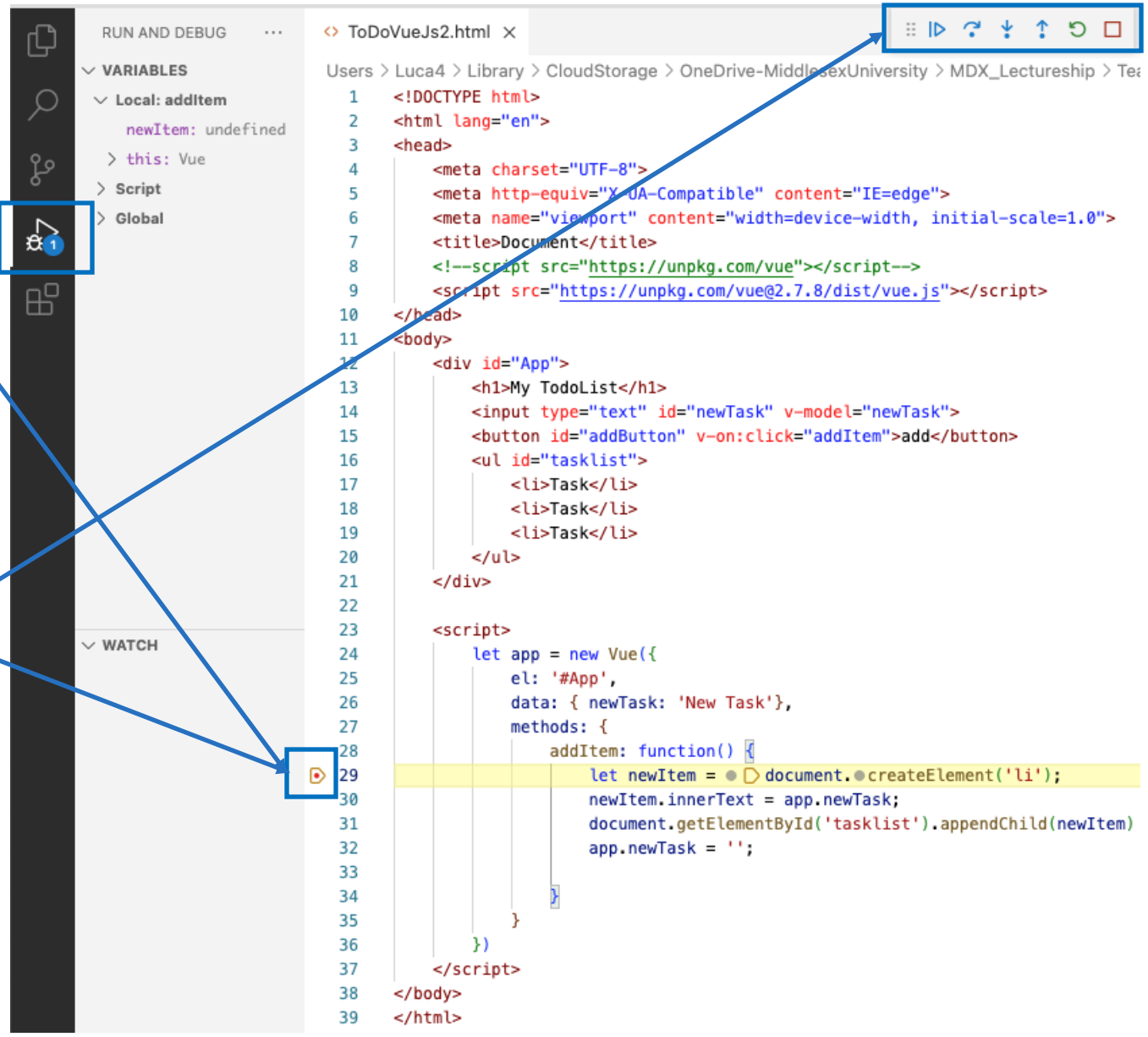


  - and the extension is turned on (when you reload your Vue.js web page)

  - Not this

Debugging Vue.js with Visual Studio Code

- **Add a breakpoint on the left side of your code**, just on the left of a line number code (a red circle will appear)
- Click on the left bar the icon of ”**Run and Debug**”
- Click on ”**Run and Debug” button** (the first time it will ask what browser to use, for example choose “Chrome”)
- **Use the app until the breakpoint is reached** and the Debugger will stop highlighting the current line with yellow
- Use the **buttons** on the top to **“Step over” a line**, to **”Step into”**, etc.
- **Move your cursor over variables** to **inspect** their values



Creating Simple Web Pages with Vue.js

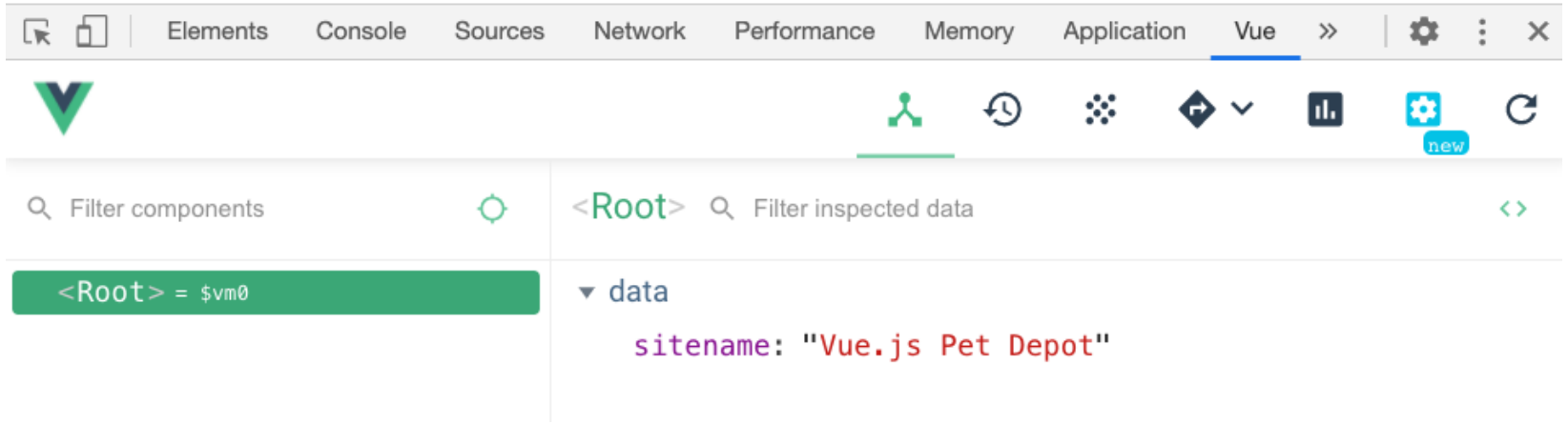
Adding and Managing Data in the Vue Instance

```
<html>
	<head>
		<title>Vue.js Pet Depot</title>
		<script src="https://unpkg.com/vue@2.7.8/dist/vue.js"></script>
	</head>
	<body>
		<div id="app">
			<header>
				<h1 v-text="sitename"></h1>
			</header>
		</div>
		<script type="text/javascript">
			var webstore = new Vue({
				el: '#app', // <=== Don't forget this comma
				data: { // the 'data' option
					// the key 'sitename' matches the value of 'v-text' earlier
					sitename: 'Vue.js Pet Depot’
				}
			});
		</script>
	</body>
</html>
```

Inspecting Data

In the vue-devtools you will be able to inspect and manipulate data

Vue.js Pet Depot



Adding Data to Vue.js for a Product

Add all the information of our product into **data**:


```
data: {
        sitename: "Vue.js Pet Depot",
        product: {
        id: 1001,
        title: "Cat Food, 25lb bag",
        description: "A 25 pound bag of <em>irresistible</em>," + "organic goodness for your cat.",
        price: 2000,
        image: "images/product-fullsize.png"
        }
}
```

How to Display the Information in our App

- **Binding** the **image** with its URL from the data
- **Binding** the **other information** to be displayed
- Notice the usage of the "Mustache" syntax {{ property-name }}

```
<div id="app">
        <header>
                <h1 v-text="sitename"></h1>
        </header>
        <main>
                <figure>
                     <!-- bind 'src' attr to 'product.image' in 'data' -->
                     <img v-bind:src="product.image">
                </figure>
                <h2 v-text="product.title"></h2>
                <p v-text="product.description"></p>


                <!-- The double curly brackets is used instead of 'v-text' -->
                <p>Price: {{product.price}}</p>
        </main>
</div>
```

- Try **v-html**

Next Slide

The Vue Directives v-text and v-html

- Instead of **v-text** , we can use **v-html** to interpret the text as HTML code:
  - ** **
- However, better to keep the usage of **v-html** at the minimum or do not use it at all, due to potential cross-site scripting (XSS) attacks ; it may be used in case the value comes from a source that you can trust at 100% -> in any case better to avoid this approach, there are better ways to achieve the same result **(e.g., using CSS)**

Cat Food, 25lb bag

A 25 pound bag of *irresistible*,organic goodness for your cat.

Price: 2000

The Result

We are now displaying the image and all the other information of our product

Vue.js Pet Depot

Cat Food, 25lb bag

A 25 pound bag of irresistible organic goodness for your cat.

Price: 2000

Suggestions for Reading

Reading

Chapter 2 of the **“Vue.js in Action”** textbook

Questions?