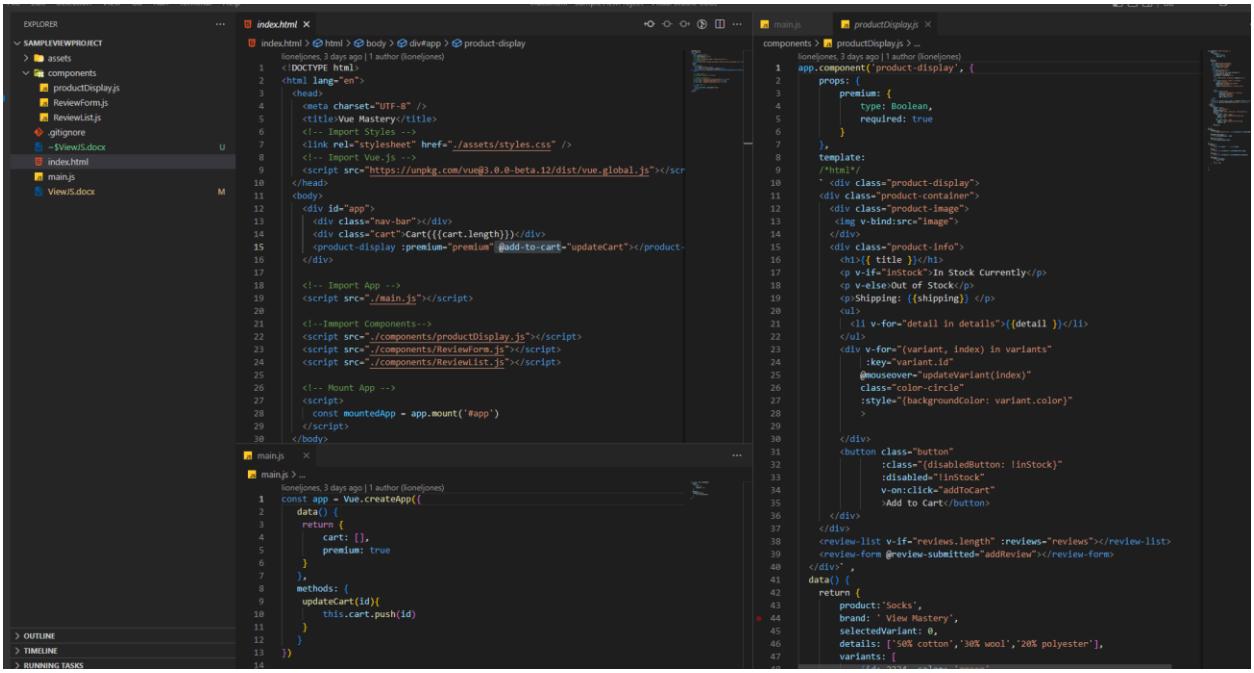
ViewJS

The first coarse:
https://www.vuemastery.com/courses/intro-to-vue-3/intro-to-vue3/

Overview of Vue:
Another good IDE for Vue
https://www.dcloud.io/hbuilderx.html?hmsr=vue-en&hmpl=&hmcu=&hmkw=&hmci=
VueMastery.com website
https://www.vuemastery.com/courses-path/beginner/
https://vuejs.org/guide/introduction.html

Features/Differences/Project Structure
Project Structure(s)
CDN Based Project Structure



This is created using the CDN version of vue
https://vuejs.org/guide/quick-start.html#with-build-tools
The example allows you to just include the script link from the official docs in the link above
The Basic project file structure:
You have a main.js file and an Index.html file
The main.js file is where you create your application using Vue.createApp
The Index.html file makes a reference to the main.js file and mounts the app you create in main.js
The index.html file is where you make a reference to each component in your application that you build via the import script tag imports
In the <div = “app”>, this is where you make reference to our imported components.

Components: (CDN Based Project)
In a CDN based project, you create your components as regular javascript files with the .js extension.
Withing the component, you a "**Template" section**. The template section contains the markup for it's component that will be rendered (The HTML). You also have the data method that houses your properties for our component (data). You also have a methods section for your methods that you want to use in your application. You also can have "**computed**" properties. Computed properties are properties that can have more that just a value, they can have "computations" or "operations" associated with them.
Tools:
VSCode has a "live" server that you can use to view real-time changes to your components as you code. You can also make changes in the console's browser tools to change properties to test out state changes in realtime.

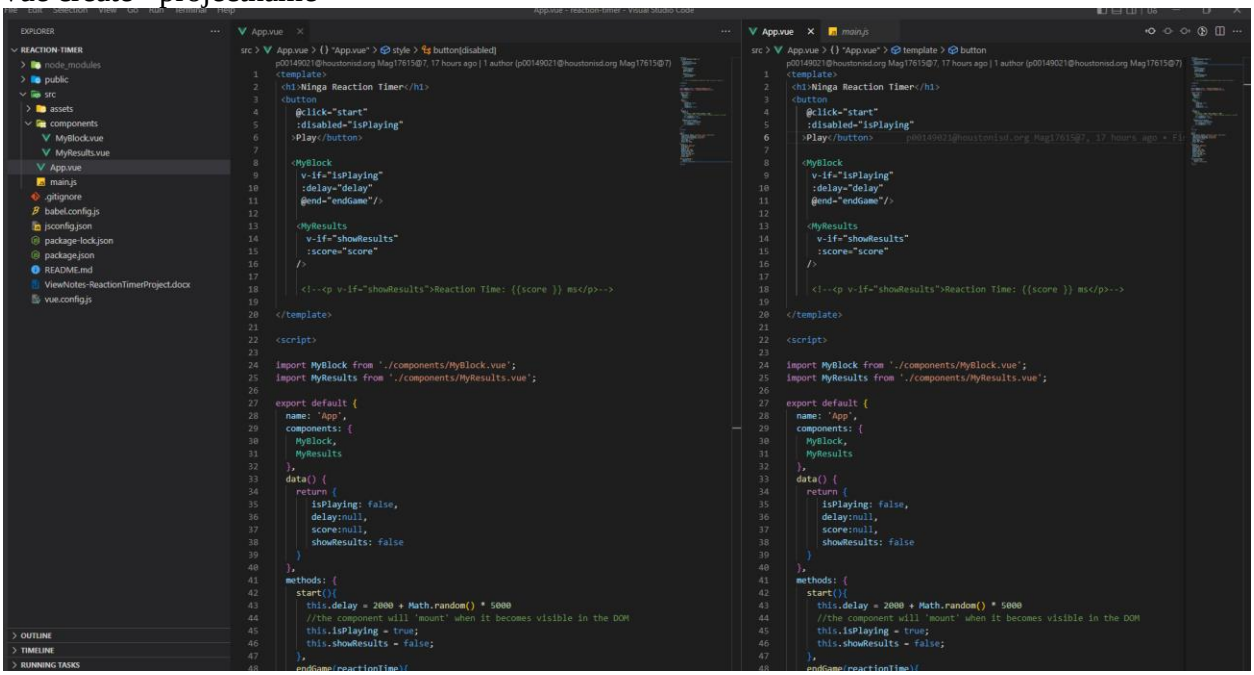
CLI Based project (Minified)
This project type is created with the vue cli
https://vuejs.org/guide/quick-start.html#with-build-tools

To create a build-tool-enabled Vue project on your machine, your command line (without the `>` sign):

```
> npm init vue@latest
```

Installing the cli on windows
https://docs.microsoft.com/en-us/windows/dev-environment/javascript/vue-on-windows
npm install -g @vue/cli

To create a project
vue create <projectname>



This type of project is designed based on SFC (Single File Components)
https://vuejs.org/guide/scaling-up/sfc.html

Why SFC

While SFCs require a build step, there are numerous benefits in return:

- Author modularized components using familiar HTML, CSS and JavaScript syntax
- Colocation of inherently coupled concerns
- Pre-compiled templates
- Component-scoped CSS
- More ergonomic syntax when working with Composition API
- More compile-time optimizations by cross-analyzing template and script
- IDE support with auto-completion and type-checking for template expressions
- Out-of-the-box Hot-Module Replacement (HMR) support

SFC is a defining feature of Vue as a framework, and is the recommended approach for using Vue in the following scenarios:

- Single-Page Applications (SPA)
- Static Site Generation (SSG)
- Any non-trivial frontend where a build step can be justified for better development experience (DX).

That said, we do realize there are scenarios where SFCs can feel like overkill. This is why Vue can still be used via plain JavaScript without a build step. If you are just looking for enhancing largely static HTML with light interactions, you can also check out petite-vue, a 6 kB subset of Vue optimized for progressive enhancement.

The project is still mainly the same with some exceptions:
You have a main.js file (the same as the CDN), this is where you create your app and mount it. (In CDN, you mount in the Index.html file). But this step is done in one file (main.js)
In the CLI (SFC) approach, you create components much like the way you create components in frameworks like React.js
The files use the .vue extension.
The components have a Template section as well as a data() function as well.
The components are imported as references the way other Javascript frameworks like React and Angular references them. (Standard es6 syntax) for importing
The export default method is used to make the components “exportable” and “importable”
Tools:
There are some good Tools as well for the SFC way as well. The Vue Tools for Chrome are very good for analyzing the state and structure and performance of your application.

Data-Binding
Vue uses Data-binding like angular does (Two way data-binding)
https://vueschool.io/lessons/user-inputs-vue-devtools-in-vue-3?friend=vuejs
This is different from React that uses a Virtual DOM

https://stackoverflow.com/questions/32712605/is-reacts-data-binding-really-one-way-it-seems-like-it-is-two-way

Angular:
Controller <--> View
React:
Component State --> DOM --> New Component State --> New DOM

Documentation:
View has "EXCELLENT!!!!" documentation. They have a lot of good training videos (paid and free).

Tip: Extensions to Install
Tip: Using the console to change reactive properties
Tip: Adding Methods
Tip: Adding a computed property
Tip: Creating components
Tip: Adding props to our re-usable component
Tip: Emitting events to child components (i:e button events) – event binding
Tip: Form and Field binding
Tip: Alerts
Tip: Importing components and the scope made available to all components
Tip: The no-brainer deployment
Tip: Spinning up a live server when coding with the CDN in vs code (as well as other extensions to install)
Tip: VModel (Two way binding), modifiers etc.
********** USING THE VIEW CLI**********
Tip: Creating a view project using the CLI (Seeing the VUE Chrome Extension work as well)
Tip: CSS <script> tag in App.vue

Tip: Another tip on emitting methods
Tip: Passing props (just a refresher)
Tip: Bootstrap for Vue
Tip: Routing with vue
Tip: Building your project for production
Tip: Testing the application to run from your build file from VS Code
Tip: AXIOS with VUE

Tip: Bootsrap for vue

https://bootstrap-vue.org/docs/components/table
Everything you can do with bootstrap (tables, elements etc..) is documented in the URL above.

Tip: Extensions to Install
Live Server
es6-string-html
Vetur

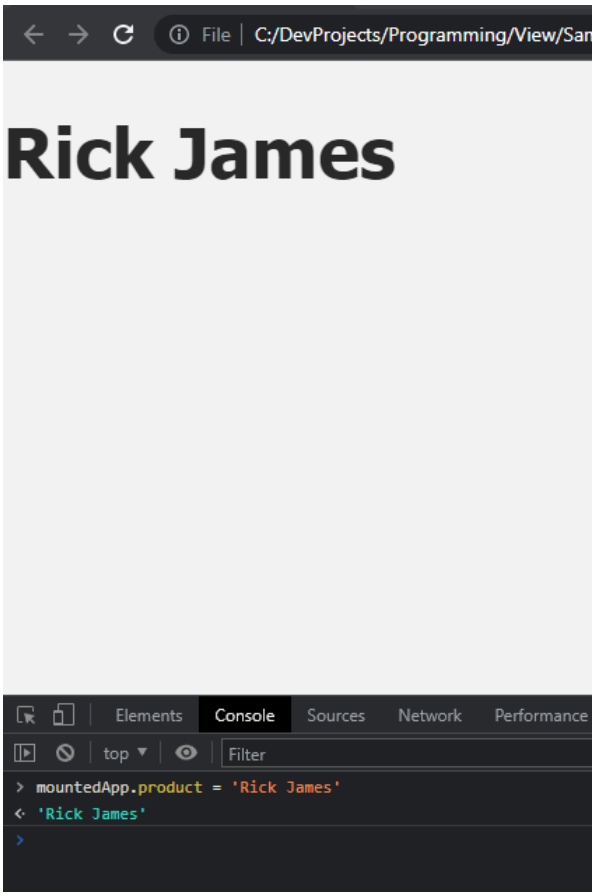
Tip: Using the console to change reactive properties
When you browse your app, you can open developer tools and change a property reactively

index.html

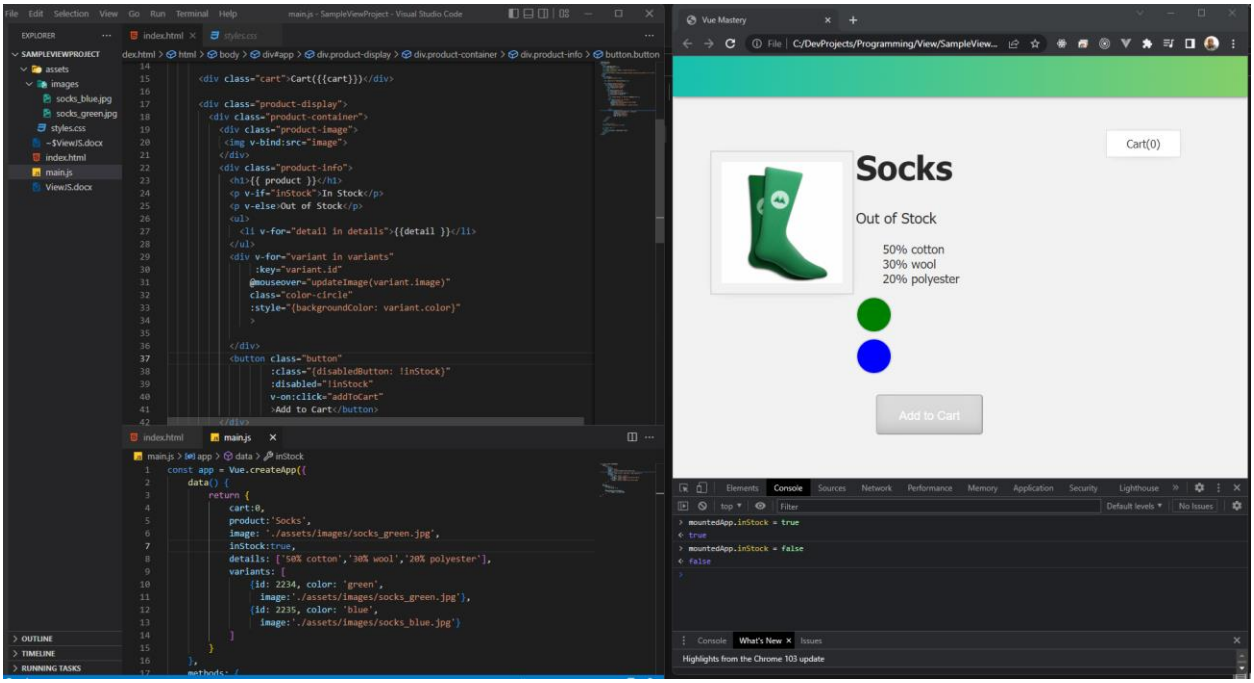
```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Vue Mastery</title>
    <!-- Import Styles -->
    <link rel="stylesheet" href="./assets/styles.css" />

    <!-- Import Vue.js -->
    <script src="https://unpkg.com/vue@3.0.0-beta.12/dist/vue.global.js"></script>
  </head>
  <body>
    <div id="app">
      <h1>{{product}}</h1>
    </div>

    <!-- Import App -->
    <script src="./main.js"></script>

   <!--Mount App-->
   <script>
    const mountedApp = app.mount('#app')
   </script>

  </body>
</html>
```

main.js

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



This works with any property on our data object for setting properties using the chrome console



Tip: Adding Methods

```
@mouseover="updateImage(variant.image)"
```

main.js

main.js > app > data > inStock

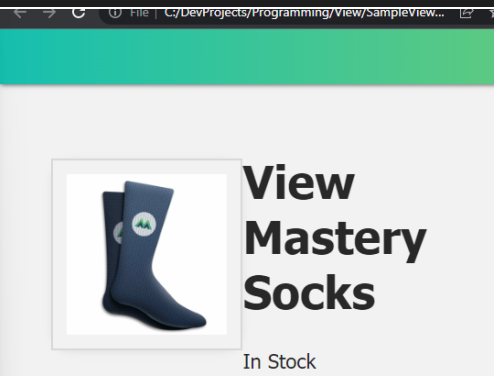
```js
const app = Vue.createApp({
    data() {
        return {
            cart:0,
            product:'Socks',
            image: './assets/images/socks_green.jpg',
            inStock:true,
            details: ['50% cotton','30% wool','20% polyester'],
            variants: [
                {id: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

```
                @mouseover="updateImage(variant.image)"
                class="color-circle"
                :style="{backgroundColor: variant.color}"
                >

            </div>
            <button class="button"
                    :class="{disabledButton: !inStock}"
                    :disabled="!inStock"
                    v-on:click="addToCart"
                    >Add to Cart</button>
        </div>
    </div>
  </div>
</div>

<!-- Import App -->
<script src="./main.js"></script>
```

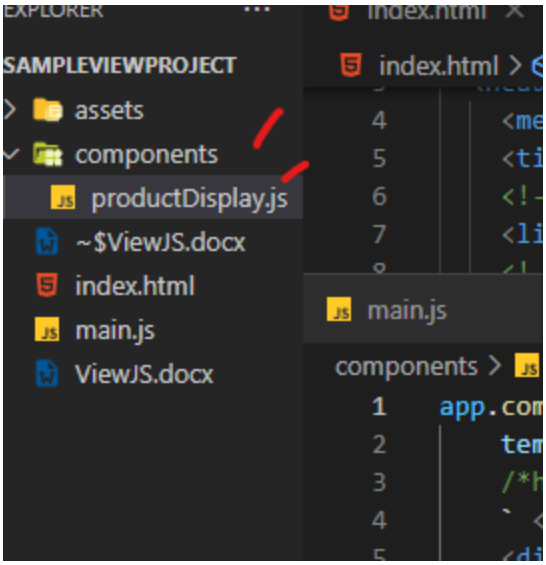
Tip: Adding a computed property

```
main.js
main.js > app > computed > title
                    image:'./assets/images/socks_blue.jpg'}
                ]
            }
        },
        methods: {
            addToCart() {
                this.cart += 1;
            },
            updateImage(variantImage) {
                this.image = variantImage
            }
        },
        computed : {
            title() {
                return this.brand + ' ' + this.product
            }
        }
    })

```

```
index.html > html > body > div#app > div.product-display > div.product-container > div.product-info
    <div id="app">
      <div class="nav-bar"></div>

      <div class="cart">Cart({{cart}})</div>

      <div class="product-display">
        <div class="product-container">
          <div class="product-image">
           <img v-bind:src="image">
          </div>
          <div class="product-info">
            <h1>{{ title }}</h1>
            <p v-if="inStock">In Stock</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.id"
```



Tip: Creating components


EXPLORER
SAMPLEVIEWPROJECT
assets
components
productDisplay.js
~$ViewJS.docx
index.html
main.js
ViewJS.docx
index.html
main.js
components

index.html × styles.css

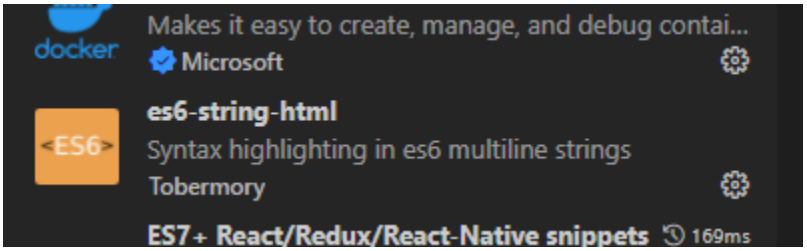
index.html > html > body > div#app > product-display

main.js productDisplay.js ×

components > productDisplay.js > ...

```
1   app.component('product-display', {
2       template:
3       /*html*/
4       ` <div class="product-display">
5       <div class="product-container">
6         <div class="product-image">
7          <img v-bind:src="image">
8         </div>
9         <div class="product-info">
10          <h1>{{ title }}</h1>
11          <p v-if="inStock">In Stock</p>
12          <p v-else>Out of Stock</p>
13          <ul>
14            <li v-for="detail in details">{{detail }}</li>
15          </ul>
16          <div v-for="(variant, index) in variants"
17               :key="variant.id"
18              @mouseover="updateVariant(index)"
19              class="color-circle"
20              :style="{backgroundColor: variant.color}"
21              >
22
23          </div>
24          <button class="button"
25                  :class="{disabledButton: !inStock}"
26                  :disabled="!inStock"
27                  v-on:click="addToCart"
28                  >Add to Cart</button>
29        </div>
30      </div>
31    </div>` ,
32    data() {
33      return {
34          product:'Socks',
35          brand: ' View Mastery',
36          selectedVariant: 0,
37          details: ['50% cotton','30% wool','20% polyester'],
38          variants: [
39              {id: 2234, color: 'green',
40               image:'./assets/images/socks_green.jpg',
41               quantity: 50
42              },
43              {id: 2235, color: 'blue',
44               image:'./assets/images/socks_blue.jpg',
```

One thing to note, you have to have the /*html*/ directive in order to get the markup not think it is a standard string. This was made possible by using the VS Code Extension:



```
View  Go  Run  Terminal  Help        ReviewForm.js - SampleViewProject - Visual Studio Code
index.html   productDisplay.js   ReviewForm.js ×   main.js
components > ReviewForm.js > template
1   app.component('review-form', {
2       template:
3       /*html*/
4       `<form class="review-form" @submit.prevent="onSubmit">
5       <h3>Leave a review</h3>
6       <label for="name">Name:</label>
7       <input id="name" >
8
9       <label for="review">Review:</label>
10      <textarea id="review"></textarea>
11
12      <label for="rating">Rating:</label>
13        <option>5</option>
14        <option>4</option>
15        <option>3</option>
16        <option>2</option>
17        <option>1</option>
18      </select>
19
20
21      <input class="button" type="submit" value="Submit">
22
23    </form>`
24  })
```

Above we cut and pasted our markup and code from our main.js and pasted it into our component

Also note:

In the main.js file, we still need to have the code:

```
30    </body>
```

main.js × productDisplay.js

main.js > app

```
1   const app = Vue.createApp({
2       data() {
3        return {
4            cart: 0
5        }
6       },
7       methods: {
8
9       }
10  })
11
```

This is the default method for vue

And now in our Index.html file

```html
<!-- Import Styles -->
<link rel="stylesheet" href="./assets/styles.css" />
<!-- Import Vue.js -->
<script src="https://unpkg.com/vue@3.0.0-beta.12/dist/vue.global.js"></script>
</head>
<body>
<div id="app">
<div class="nav-bar"></div>
<div class="cart">Cart({{cart}})</div>
<product-display></product-display>
</div>

<!-- Import App -->
<script src="./main.js"></script>

<!--Immport Components-->
<script src="./components/productDisplay.js"></script>

<!-- Mount App -->
<script>
const mountedApp = app.mount('#app')
</script>
</body>
</html>
```

We import the component via script tag and reference the tag like in angular

Tip: Adding props to our re-usable component

index.html productDisplay.js main.js

components > productDisplay.js > template

```
app.component('product-display', {
    props: {
        premium: {
            type: Boolean,
            required: true
        }
    },
    template:
    /*html*/
    ` <div class="product-display">
    <div class="product-container">
      <div class="product-image">
```

Go Run Terminal Help main.js - SampleViewProject - Visual Studio

index.html productDisplay.js main.js

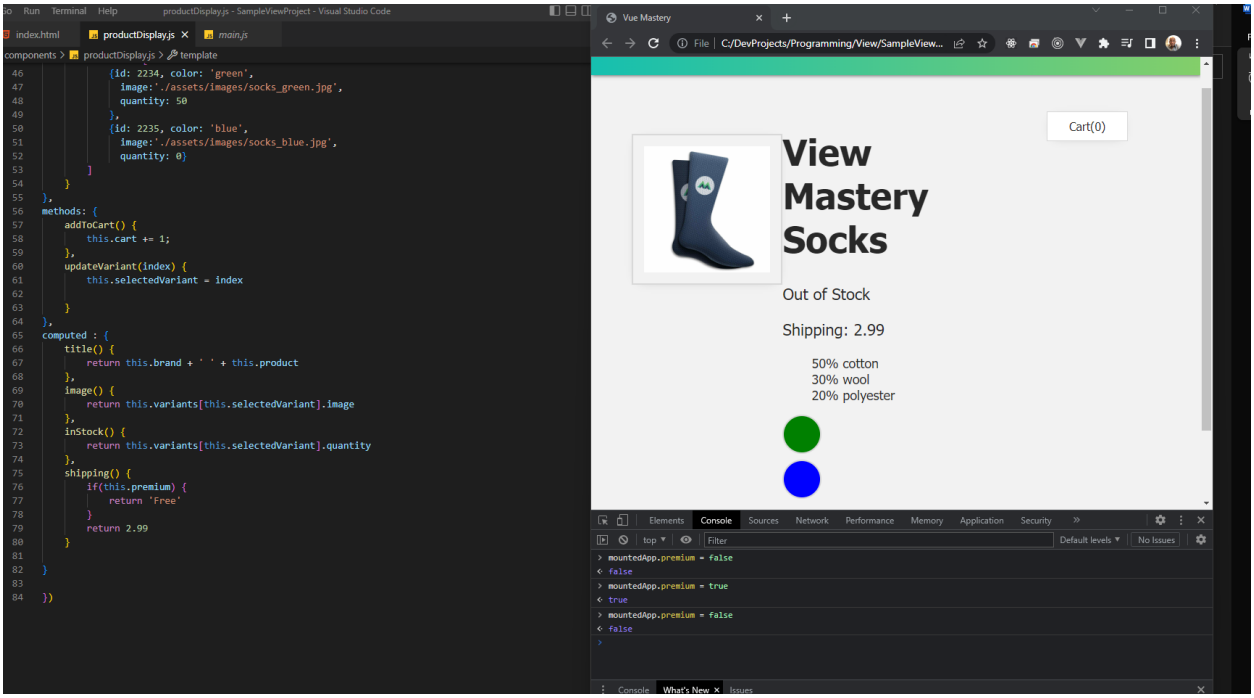
main.js > ...

```
const app = Vue.createApp({
    data() {
     return {
         cart: 0,
         premium: true
     }
    },
    methods: {

    }
})
```

```
    },
    template:
    /*html*/
    ` <div class="product-display">
    <div class="product-container">
      <div class="product-image">
       <img v-bind:src="image">
      </div>
      <div class="product-info">
        <h1>{{ title }}</h1>
        <p v-if="inStock">In Stock</p>
        <p v-else>Out of Stock</p>
        <p>Shipping: {{shipping}} </p>
        <ul>
          <li v-for="detail in details">{{detail }}</li>
        </ul>
        <div v-for="(variant, index) in variants"
             :key="variant.id"
            @mouseover="updateVariant(index)"
            class="color-circle"
            :style="{backgroundColor: variant.color}"
            >
```

```
            return this.variants[this.selectedVariant].image
        },
        inStock() {
            return this.variants[this.selectedVariant].quantity
        },
        shipping() {
            if(this.premium) {
                return 'Free'
            }
            return 2.99
        }

    }

```

To check



Tip: Emitting events to child components (i:e button events) – event binding
In order to fix the button event, we have to create a event binder using emit
Over in our component:

View Go Run Terminal Help productDisplay.js - SampleViewProject - Visual Studio Code

index.html productDisplay.js main.js

components > productDisplay.js > methods > addToCart

```
38      </div>` ,
39      data() {
40        return {
41            product:'Socks',
42            brand: ' View Mastery',
43            selectedVariant: 0,
44            details: ['50% cotton','30% wool','20% polyester'],
45            variants: [
46                {id: 2234, color: 'green',
47                  image:'./assets/images/socks_green.jpg',
48                  quantity: 50
49                },
50                {id: 2235, color: 'blue',
51                  image:'./assets/images/socks_blue.jpg',
52                  quantity: 0}
53            ]
54        }
55    },
56    methods: {
57        addToCart() {
58            this.$emit('add-to-cart')
59        },
60        updateVariant(index) {
61            this.selectedVariant = index
62
63        }
64    },
65    computed : {
66        title() {
67            return this.brand + ' ' + this.product
68        },
```

In our index.html file:

```html
index.html > html > body
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Vue Mastery</title>
    <!-- Import Styles -->
    <link rel="stylesheet" href="./assets/styles.css" />
    <!-- Import Vue.js -->
    <script src="https://unpkg.com/vue@3.0.0-beta.12/dist/vue.global.js"></script>
  </head>
  <body>
    <div id="app">
      <div class="nav-bar"></div>
      <div class="cart">Cart({{cart}})</div>
      <product-display :premium="premium" @add-to-cart="updateCart"></product-display>
    </div>

    <!-- Import App -->
    <script src="./main.js"></script>

    <!--Immport Components-->
    <script src="./components/productDisplay.js"></script>

    <!-- Mount App -->
    <script>
      const mountedApp = app.mount('#app')
    </script>
  </body>
</html>
```

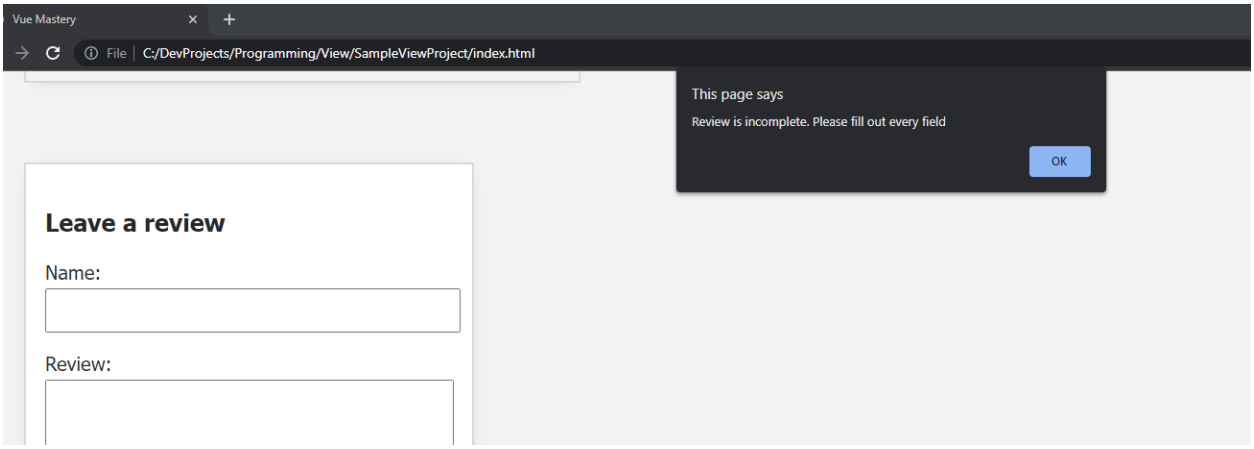
Then in our main.js file for our methods:

```
index.html   productDisplay.js   main.js ×
main.js > ...
const app = Vue.createApp({
    data() {
     return {
         cart: 0,
         premium: true
     }
    },
    methods: {
     updateCart(){
         this.cart +=1
     }
    }
})
```

it's pretty straight forward

Tip: Alerts

```
        review: '',
        rating: null
    }
  },
  methods: {
   onSubmit(){

        if(this.name ==='' || this.review === '' || this.rating === null){
            alert('Review is incomplete. Please fill out every field');
            return
        }

        let productReview = {
            name: this.name,
            review: this.review,
            rating: this.rating
        }
        this.$emit('review-submitted',productReview)

        //reset
        this.name = ''
        this.review = ''
        this.rating = null
    }

  }
})
```



Tip: Form and field binding:

Two way binding accomplished as shown below

ReviewForm.js

components > ReviewForm.js > ...

```
4       `<form class="review-form" @submit.prevent="onSubmit">
5       <h3>Leave a review</h3>
6       <label for="name">Name:</label>
7       <input id="name" v-model="name" >
8
9       <label for="review" >Review:</label>
10      <textarea id="review" v-model="review"></textarea>
11
12
13      <label for="rating">Rating:</label>
14      <select id="rating" v-model.number="rating">
15        <option>5</option>
16        <option>4</option>
17        <option>3</option>
18        <option>2</option>
19        <option>1</option>
20      </select>
21
22
23      <input class="button" type="submit" value="Submit">
24
25    </form>`,
26    data() {
27     return {
28         name:'',
29         review: '',
30         rating: null
31     }
32    },
33    methods: {
34     onSubmit(){
35
36         if(this.name ==='' || this.review === '' || this.rating === null){
37             alert('Review is incomplete. Please fill out every field');
38             return
39         }
40
41         let productReview = {
42             name: this.name,
43             review: this.review,
44             rating: this.rating
45         }
46         this.$emit('review-submitted',productReview)
47
48         //reset
49         this.name = ''
50         this.review = ''
51         this.rating = null
```

Ln 32, Col 6 Spaces: 4 UTF-8 CRLF {

Tip: Importing components and the scope made available to all components

One interesting thing about imports in Vue:
You import all of your components in the index.html file only

ReviewForm.js index.html ×

index.html > ...

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Vue Mastery</title>
    <!-- Import Styles -->
    <link rel="stylesheet" href="./assets/styles.css" />
    <!-- Import Vue.js -->
    <script src="https://unpkg.com/vue@3.0.0-beta.12/dist/vue.global.js"></script>
  </head>
  <body>
    <div id="app">
      <div class="nav-bar"></div>
      <div class="cart">Cart({{cart.length}})</div>
      <product-display :premium="premium" @add-to-cart="updateCart"></product-display>
    </div>

    <!-- Import App -->
    <script src="./main.js"></script>

    <!--Immport Components-->
    <script src="./components/productDisplay.js"></script>
    <script src="./components/ReviewForm.js"></script>
    <script src="./components/ReviewList.js"></script>

    <!-- Mount App -->
    <script>
      const mountedApp = app.mount('#app')
    </script>
  </body>
</html>
```

Then in any of your components, you can reference them

productDisplay.js ReviewList.js

components > productDisplay.js > template

```
25            @mouseover="updateVariant(index)"
26            class="color-circle"
27            :style="{backgroundColor: variant.color}"
28            >
29
30        </div>
31        <button class="button"
32                :class="{disabledButton: !inStock}"
33                :disabled="!inStock"
34                v-on:click="addToCart"
35                >Add to Cart</button>
36      </div>
37    </div>
38    <review-list v-if="reviews.length" :reviews="reviews"></review-list>
39    <review-form @review-submitted="addReview"></review-form>
40  </div>` ,
41  data() {
42    return {
43        product:'Socks',
44        brand: ' View Mastery',
45        selectedVariant: 0,
```

Tip: The no-brainer deployment

Shit is a no-brainer, just copy over the file structure:

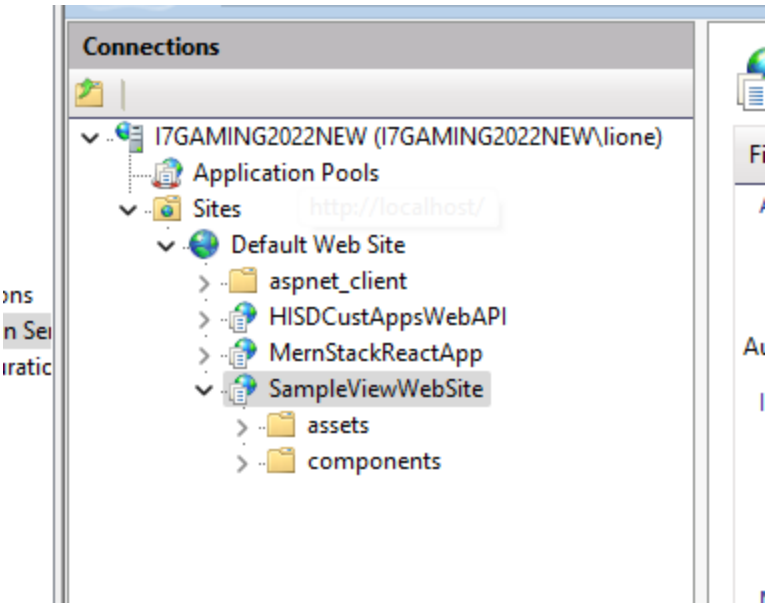
Sort View

This PC > OS (C:) > DevProjects > Programming > WebSites > SampleViewWebSite

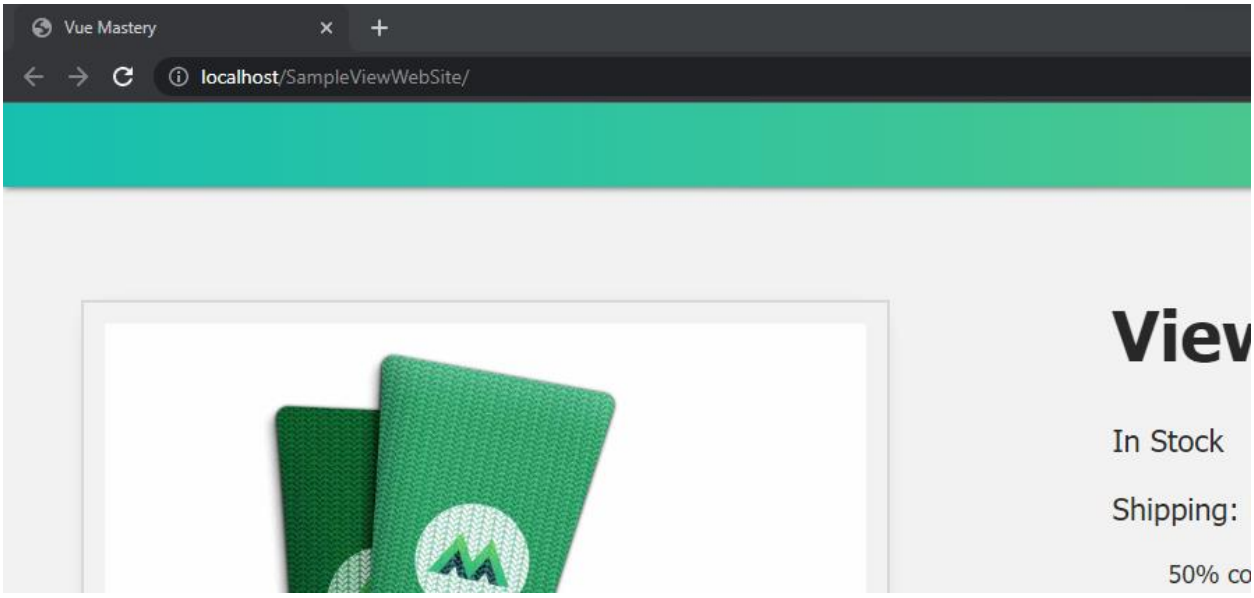
Name	Date modified	Type	Size
assets	7/3/2022 6:38 PM	File folder	
components	7/3/2022 6:38 PM	File folder	
.gitignore	6/30/2022 6:00 PM	Git Ignore Source ...	1 KB
index.html	7/3/2022 1:37 PM	Chrome HTML Do...	1 KB
main.js	7/3/2022 1:07 PM	JavaScript File	1 KB

r_html_theme

Add application in IIS

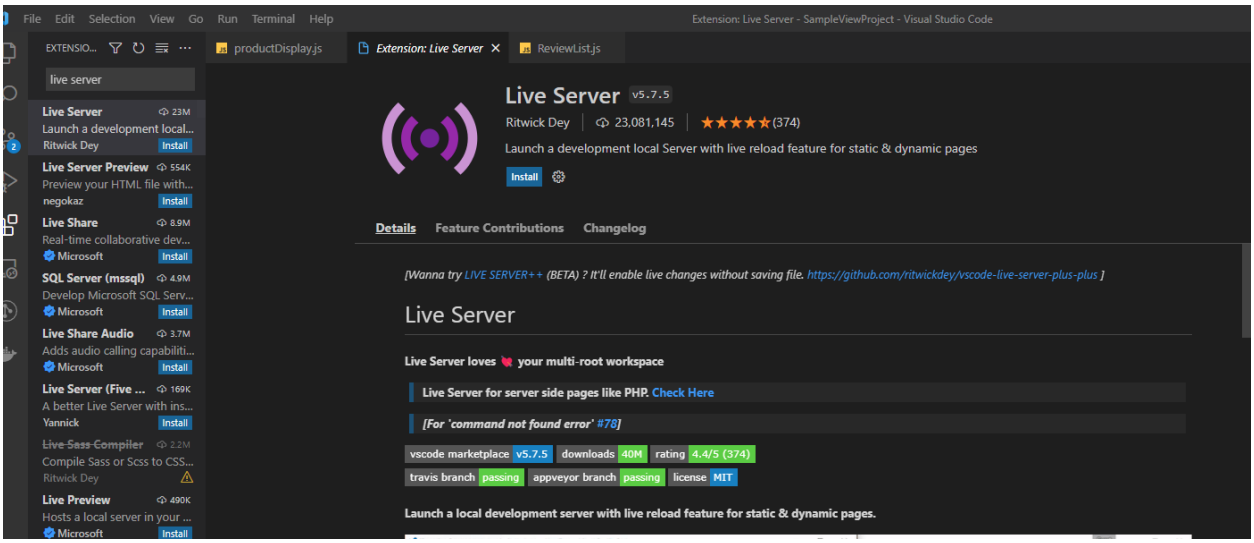


And just browse:

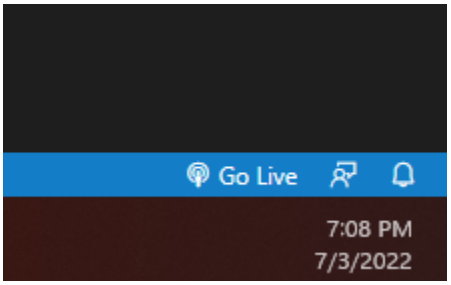


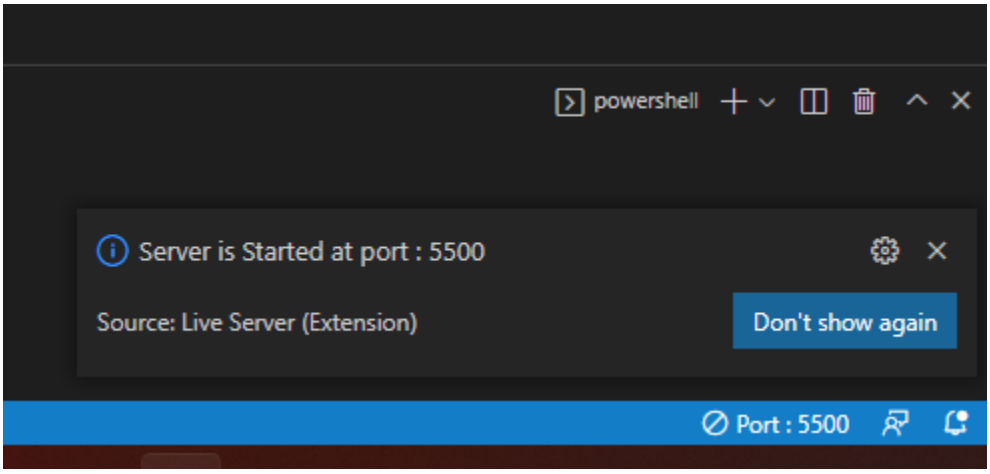
Tip: Spinning up a live server when coding with the CDN in vs code

In VS code add this extension



To use the extension, click on the icon at the bottom tool-bar





Then right-click on the index.html file:

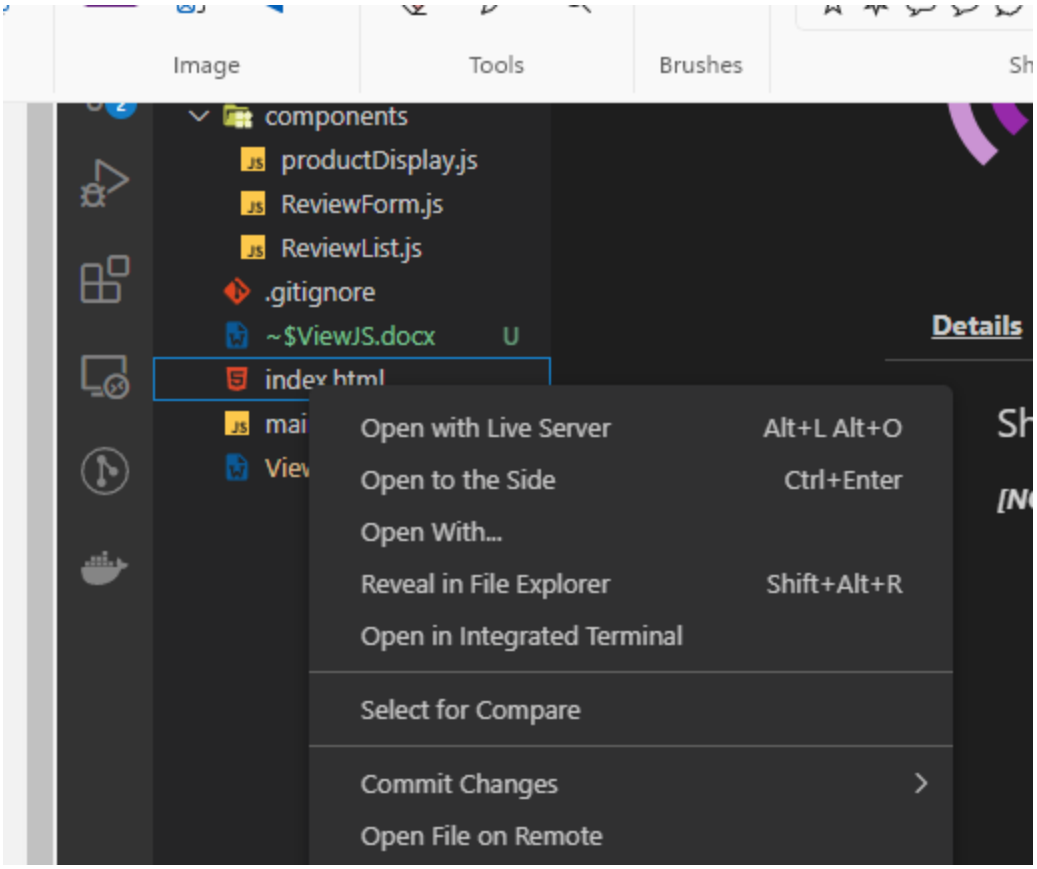
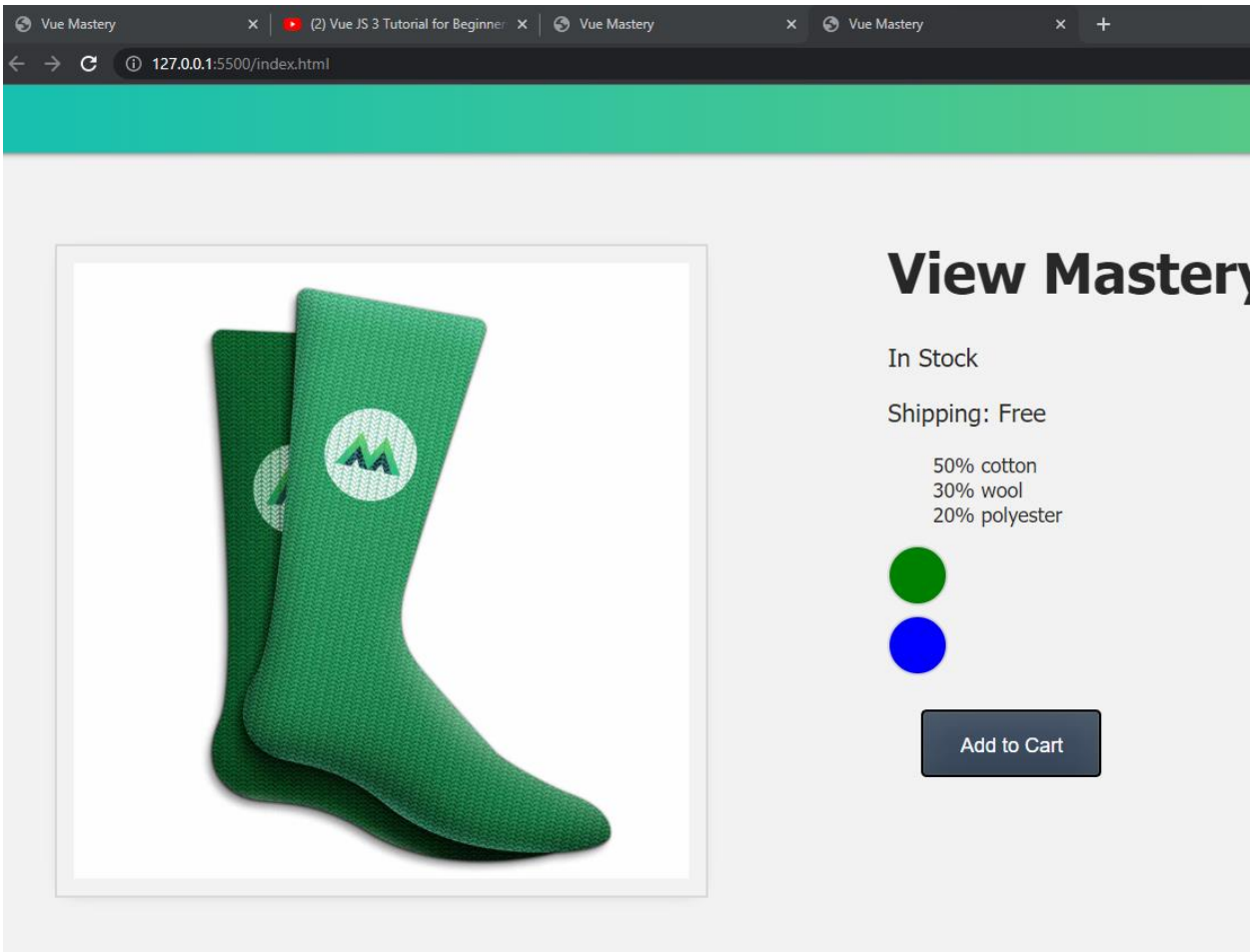
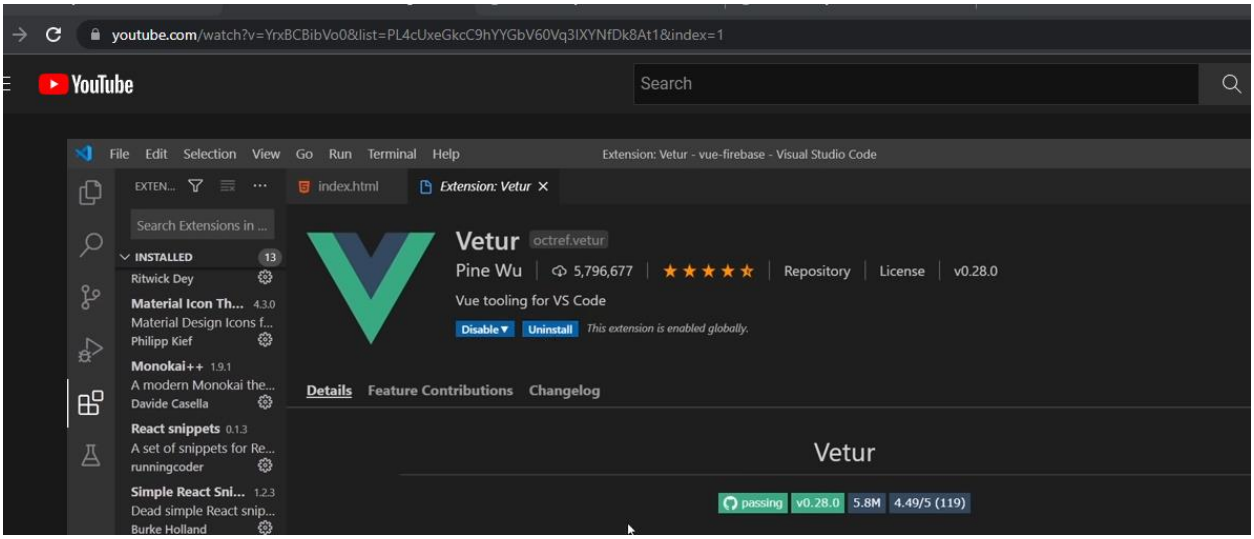


Image
Tools
Brushes
components
productDisplay.js
ReviewForm.js
ReviewList.js
.gitignore
~$ViewJS.docx
U
index.html
Details
Open with Live Server
Alt+L Alt+O
Open to the Side
Ctrl+Enter
Open With...
Reveal in File Explorer
Shift+Alt+R
Open in Integrated Terminal
Select for Compare
Commit Changes
Open File on Remote

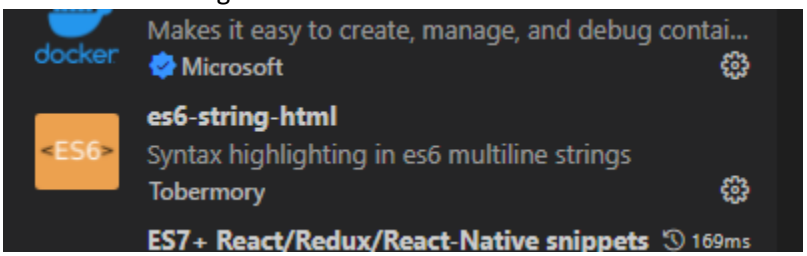


And as you make changes, they will be reflected in your page immediately. SWEET!!!!

Another Extension to install as well



And also don’t forget:



Tip: Creating a view project using the CLI
This is from the video series
https://www.youtube.com/watch?v=GWRvrSqnFbM&list=PL4cUxeGkcC9hYYGbV60Vq3IXYNfDk8At1&index=4
To install vue (provided you have node installed)
Using Powershell (it's better)
npm install -g @vue/cli

```
Windows PowerShell
npm WARN deprecated apollo-tracing@0.15.0: The `apollo-tracing` package is no longer part of Apollo Server 3. See h
//www.apollographql.com/docs/apollo-server/migration/#tracing for details
npm WARN deprecated uuid@3.4.0: Please upgrade  to version 7 or higher.  Older versions may use Math.random() in ce
 circumstances, which is known to be problematic.  See https://v8.dev/blog/math-random for details.
npm WARN deprecated source-map-resolve@0.5.3: See https://github.com/lydell/source-map-resolve#deprecated
npm WARN deprecated source-map-url@0.4.1: See https://github.com/lydell/source-map-url#deprecated
npm WARN deprecated resolve-url@0.2.1: https://github.com/lydell/resolve-url#deprecated
npm WARN deprecated urix@0.1.0: Please see https://github.com/lydell/urix#deprecated
C:\Users\lione\AppData\Roaming\npm\vue -> C:\Users\lione\AppData\Roaming\npm\node_modules\@vue\cli\bin\vue.js

> core-js-pure@3.23.3 postinstall C:\Users\lione\AppData\Roaming\npm\node_modules\@vue\cli\node_modules\core-js-pur
> node -e "try{require('./postinstall')}catch(e){}"

Thank you for using core-js ( https://github.com/zloirock/core-js ) for polyfilling JavaScript standard library!

The project needs your help! Please consider supporting of core-js:
> https://opencollective.com/core-js
> https://patreon.com/zloirock
> bitcoin: bc1qlea7544qtsmj2rayg0lthvza9fau63ux0fstcz

Also, the author of core-js ( https://github.com/zloirock ) is looking for a good job -)


> @apollo/protobufjs@1.2.2 postinstall C:\Users\lione\AppData\Roaming\npm\node_modules\@vue\cli\node_modules\@apoll
tobufjs
> node scripts/postinstall

+ @vue/cli@5.0.6
added 896 packages from 534 contributors in 55.874s
PS C:\Users\lione>
: Creating a view project using the CLI
```

To create a new project
vue create modal-project

```
> OS (C:) > DevProjects > Programming > View                    Search View
Command Prompt
Microsoft Windows [Version 10.0.22000.739]
(c) Microsoft Corporation. All rights reserved.

C:\Users\lione>cd C:\DevProjects\Programming\View

C:\DevProjects\Programming\View>vue create modal-project
```

```
Also, the author of core-js ( https://github.com/zloirock ) is looking for a good job -)

npm
Vue CLI v5.0.6
   Creating project in C:\DevProjects\Programming\View\modal-project.
  Initializing git repository...
  Installing CLI plugins. This might take a while...

[                .......] / finalize:chalk: sill finalize C:\DevProjects\Programming\View\modal-project
```

```
Command Prompt

added 846 packages from 463 contributors and audited 847 packages in 17.601s

88 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

   Invoking generators...
   Installing additional dependencies...

added 94 packages from 62 contributors and audited 941 packages in 6.868s

98 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

   Running completion hooks...

   Generating README.md...

   Successfully created project modal-project.
   Get started with the following commands:

 $ cd modal-project
 $ npm run serve


C:\DevProjects\Programming\View>
```

As shown above, you can run the project with the command:
npm run serve
And as you can see, the project is like the one on code-sandbox on my iPAD

File Edit Selection View Go Run Terminal Help — HelloWorld.vue - modal-project - Visual Studio Code

EXPLORER

MODAL-PROJECT
- node_modules
- public
  - favicon.ico
  - index.html
- src
  - assets
    - logo.png
  - components
    - HelloWorld.vue
  - App.vue
  - main.js
- .gitignore
- babel.config.js
- jsconfig.json
- package-lock.json
- package.json
- README.md
- vue.config.js

OUTLINE

HelloWorld.vue

src > components > HelloWorld.vue > {} "HelloWorld.vue" > template

You, 2 minutes ago | 1 author (You)

```vue
<template>
  <div class="hello">
    <h1>{{ msg }}</h1>
    <p>
      For a guide and recipes on how to configure / customize this project,<br>
      check out the
      <a href="https://cli.vuejs.org" target="_blank" rel="noopener">vue-cli documentation</a>.
    </p>
    <h3>Installed CLI Plugins</h3>
    <ul>
      <li><a href="https://github.com/vuejs/vue-cli/tree/dev/packages/%40vue/cli-plugin-babel" target="_blank" rel="noopener">babel</a></li>
      <li><a href="https://github.com/vuejs/vue-cli/tree/dev/packages/%40vue/cli-plugin-eslint" target="_blank" rel="noopener">eslint</a></li>
    </ul>
    <h3>Essential Links</h3>
    <ul>
      <li><a href="https://vuejs.org" target="_blank" rel="noopener">Core Docs</a></li>
      <li><a href="https://forum.vuejs.org" target="_blank" rel="noopener">Forum</a></li>
      <li><a href="https://chat.vuejs.org" target="_blank" rel="noopener">Community Chat</a></li>
      <li><a href="https://twitter.com/vuejs" target="_blank" rel="noopener">Twitter</a></li>
      <li><a href="https://news.vuejs.org" target="_blank" rel="noopener">News</a></li>
    </ul>
    <h3>Ecosystem</h3>
    <ul>
      <li><a href="https://router.vuejs.org" target="_blank" rel="noopener">vue-router</a></li>
      <li><a href="https://vuex.vuejs.org" target="_blank" rel="noopener">vuex</a></li>
      <li><a href="https://github.com/vuejs/vue-devtools#vue-devtools" target="_blank" rel="noopener">vue-devtools</a></li>
      <li><a href="https://vue-loader.vuejs.org" target="_blank" rel="noopener">vue-loader</a></li>
      <li><a href="https://github.com/vuejs/awesome-vue" target="_blank" rel="noopener">awesome-vue</a></li>
    </ul>
  </div>
</template>

<script>
export default {
  name: 'HelloWorld',
  props: {
    msg: String
  }
}
</script>

<!-- Add "scoped" attribute to limit CSS to this component only -->
<style scoped>
h3 {
  margin: 40px 0 0;
```

MASTER_DEVELOPMENT_LAN… | modal-project

Not secure | 192.168.50.60:8080

Welcome to Your Vue.js App

For a guide and recipes on how to configure / customize this project,
check out the vue-cli documentation.

Installed CLI Plugins

babel eslint

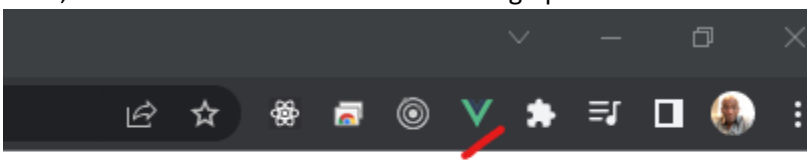
Essential Links

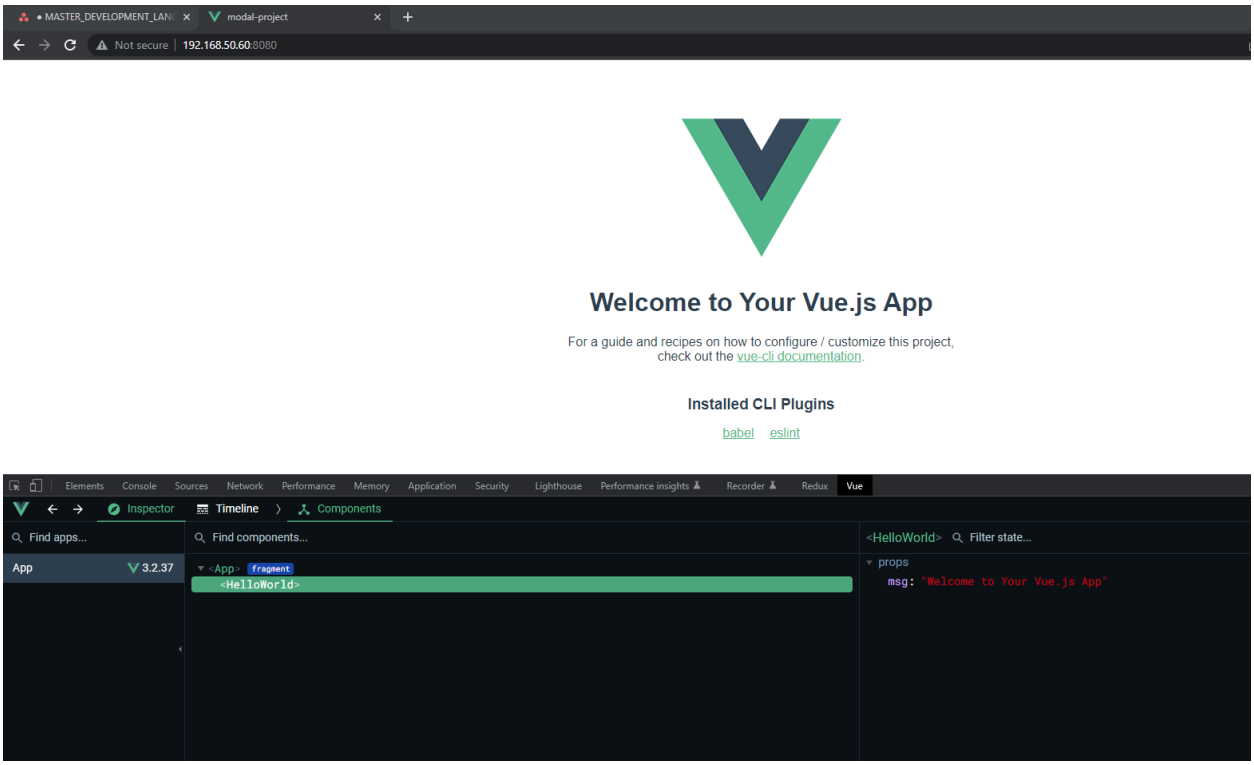
Core Docs Forum Community Chat Twitter News

Ecosystem

vue-router vuex vue-devtools vue-loader awesome-vue

Also , the Vue Chrome extension is showing up as well

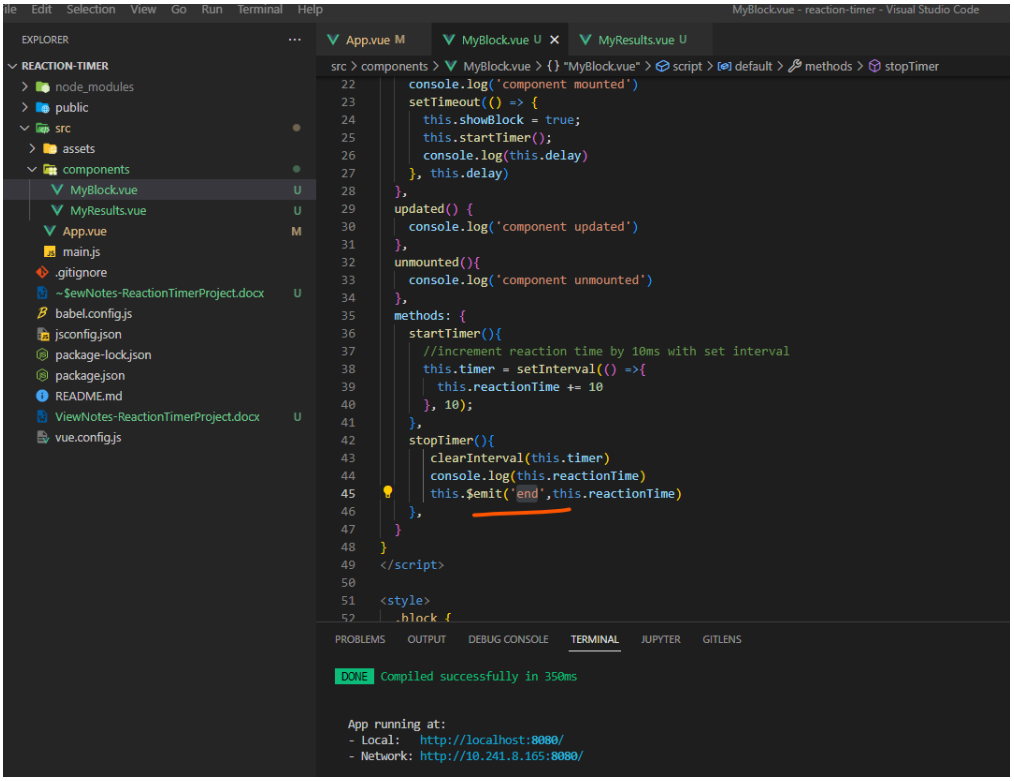




Tip: CSS <style> tag in App.vue
One thing to note, in the App.vue file there is a <style> tag, any .css you place in there will be global to all components (see my asana note on code sandbox to see what I am taking about)

This is a good video that explains how to use global styles as well
https://www.youtube.com/watch?v=KM1U6DqZf8M&list=PL4cUxeGkcC9hYYGbV60Vq3IXYNfDk8At1&index=5

Tip: Another tip on emitting methods
If you look below, we are emitting a method that we will name 'end' in the parent component. Notice how we are passing data along with it.



Now over in our parent component:

Notice below how we are making reference to the emitter by its name (‘end’)

We are pointing it to the method “endgame”

In the method, we are specifying an argument as a parameter for the function. When created the emitter, we passed in some data, so what that means is that the method create automatically receives that data as it’s parameter.

EXPLORER

REACTION-TIMER

- node_modules
- public
- src
  - assets
  - components
    - MyBlock.vue U
    - MyResults.vue U
  - App.vue M
  - main.js
- .gitignore
- ~$ewNotes-ReactionTimerProject.docx U
- babel.config.js
- jsconfig.json
- package-lock.json
- package.json
- README.md
- ViewNotes-ReactionTimerProject.docx U
- vue.config.js

OUTLINE

App.vue M | MyBlock.vue U | MyResults.vue U

src > App.vue > {} "App.vue" > template > p

```vue
<button
  @click="start"
  :disabled="isPlaying"
 >Play</button>
 <MyBlock
   v-if="isPlaying"
   :delay="delay"
   @end="endGame"/>
   <p>Reaction Time: {{score }}</p>      You, 5 minutes ago • Uncommitted changes
</template>

<script>

import MyBlock from './components/MyBlock.vue'

export default {
  name: 'App',
  components: {
    MyBlock,
  },
  data() {
    return {
      isPlaying: false,
      delay:null,
      score:null
    }
  },
  methods: {
    start(){
      this.delay = 2000 + Math.random() * 5000
      //the component will 'mount' when it becomes visible in the DOM
      this.isPlaying = true;
    },
    endGame(reactionTime){
      this.score = reactionTime;
      this.isPlaying = false;
    }
  }
}
```

PROBLEMS | OUTPUT | DEBUG CONSOLE | TERMINAL | JUPYTER | GITLENS

```
DONE  Compiled successfully in 350ms

App running at:
- Local:   http://localhost:8080/
```

Tip: Passing props (just a refresher)

We declare props

Help MyResul

App.vue M | MyResults.vue U | MyBlock.vue U

src > components > MyResults.vue > {} "MyResults.vue" > script

```vue
<template>
  <div>
    <p >Reaction Time: {{score }} ms</p>
  </div>
</template>

<script>
export default {
  name:'MyResults',
  props:['score'],
  data() {
    return {

    }
  },
}
</script>

<style>

</style>
```

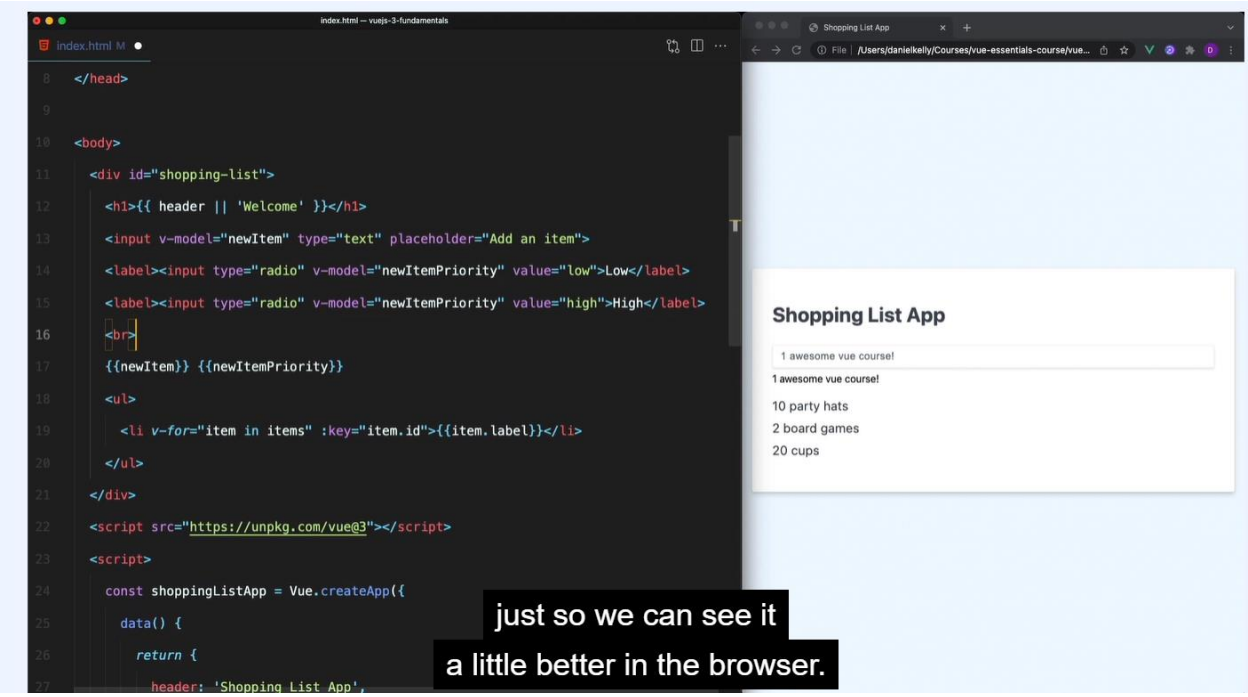
Then to receive the prop from the parent:

src > App.vue > {} "App.vue" > template

```
      :disabled="isPlaying"
    >Play</button>
          You, 4 minutes ago • Uncommitted changes
    <MyBlock
      v-if="isPlaying"
      :delay="delay"
      @end="endGame"/>

      <MyResults
        v-if="showResults"
        :score="score"
      />

        <!--<p v-if="showResults">Reaction Time: {{score }} ms</p>-->

  </template>

  <script>

  import MyBlock from './components/MyBlock.vue';
  import MyResults from './components/MyResults.vue';

  export default {
    name: 'App',
    components: {
      MyBlock,
      MyResults
    },
    data() {
      return {
          isPlaying: false,
          delay:null,
          score:null,
          showResults: false
      }
    },
    methods: {
      start(){
        this.delay = 2000 + Math.random() * 5000
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER GITLENS

Tip: VModel (Two way binding), modifiers etc.
https://vueschool.io/lessons/user-inputs-vue-devtools-in-vue-3?friend=vuejs
This is a “VERY” video on vmodel and other features working with model/view in your application



Tip: Routing with vue
https://www.vuemastery.com/blog/vue-router-a-tutorial-for-vue-3/?gclid=Cj0KCQjw5ZSWBhCVARIsALERCvyeRNuUS6qrbYZ3sSelC4Pad_blHQWUgx9pb7i9TcAzH0vIo5aMOcUaArXpEALw_wcB
Routing is done the similar to “REACT” (SWEETTTT!!), see the link above for a sample creating of how to install and use routing in Vue.js

1. **Install the Vue 3 Router from the Command Line**

```
$ npm i vue-router@next
```

2. **Add a routing directory & configuration file**

/src/router/index.js

```
import { createWebHistory, createRouter } from "vue-router";
import Home from "@/views/Home.vue";
import About from "@/views/About.vue";

const routes = [
  {
    path: "/",
    name: "Home",
    component: Home,
  },
  {
    path: "/about",
    name: "About",
    component: About,
  },
];

const router = createRouter({
  history: createWebHistory(),
  routes,
});

export default router;
```

Notice that we create our `routes` in an array, where we specify for each route a few important items:

Tip: Building your project for production
npm run build
it builds the dist folder

File Edit Selection View Go Run Terminal Help

EXPLORER

REACTION-TIMER

- dist
  - css
  - js
  - favicon.ico
  - index.html
- node_modules
- public
- src
- .gitignore
- babel.config.js
- jsconfig.json
- package-lock.json
- package.json
- README.md
- ViewNotes-ReactionTi...
- vue.config.js

App.vue

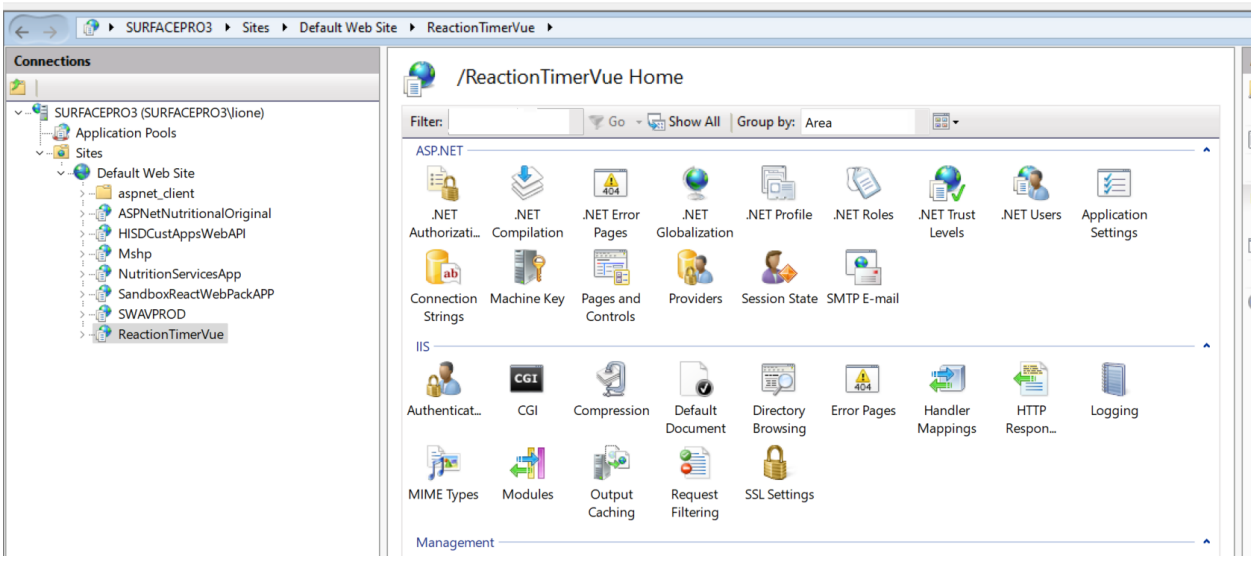
src > App.vue

p00149021@houstonisd.org Mag17615@7, 2

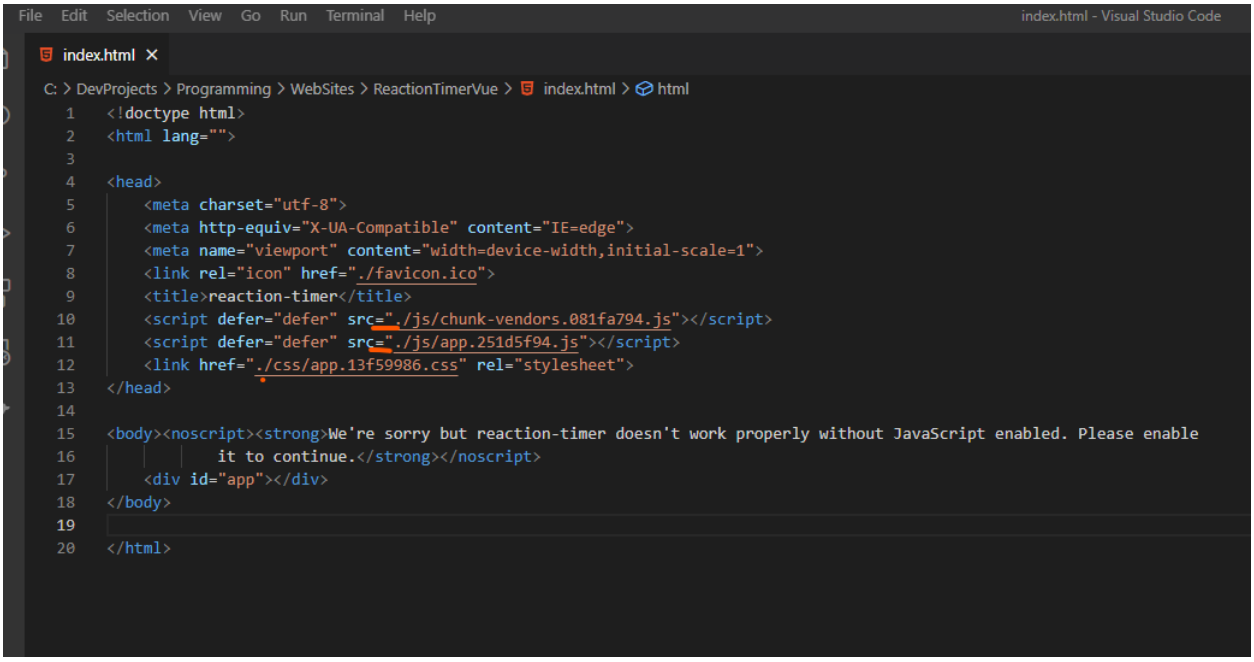
```vue
<template>
  <h1>Ninga Reaction Timer</h1>
  <button
     @click="start"
     :disabled="isPlaying"
    >Play</button>

    <MyBlock
      v-if="isPlaying"
      :delay="delay"
      @end="endGame"/>

     <MyResults
       v-if="showResults"
       :score="score"
     />

      <!--<p v-if="showResults">

</template>
```

This PC > Local Disk (C:) > DevProjects > Programming > WebSites > ReactionTimerVue

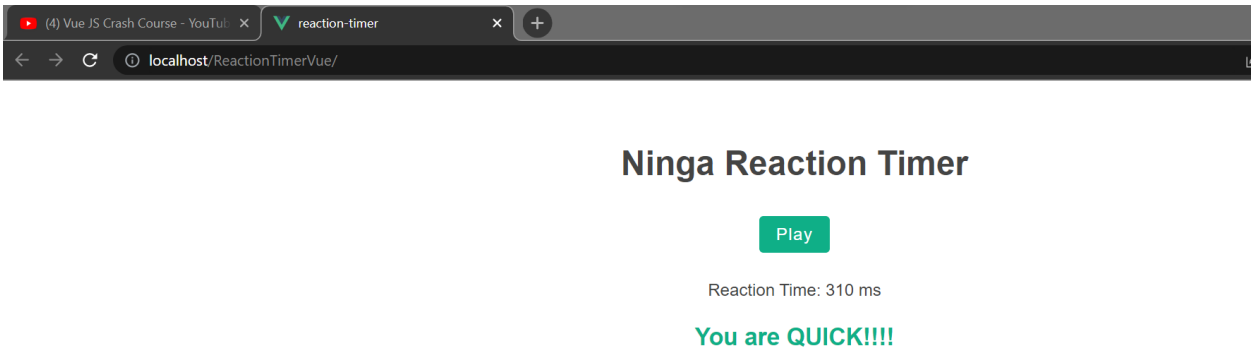
Name	Date modif
css	7/6/2022 1:
js	7/6/2022 1:
favicon.ico	7/6/2022 1:
index.html	7/6/2022 1:



But for some reason, you have to change the relative paths (just like when you build a regular react project without webpack)



And it works



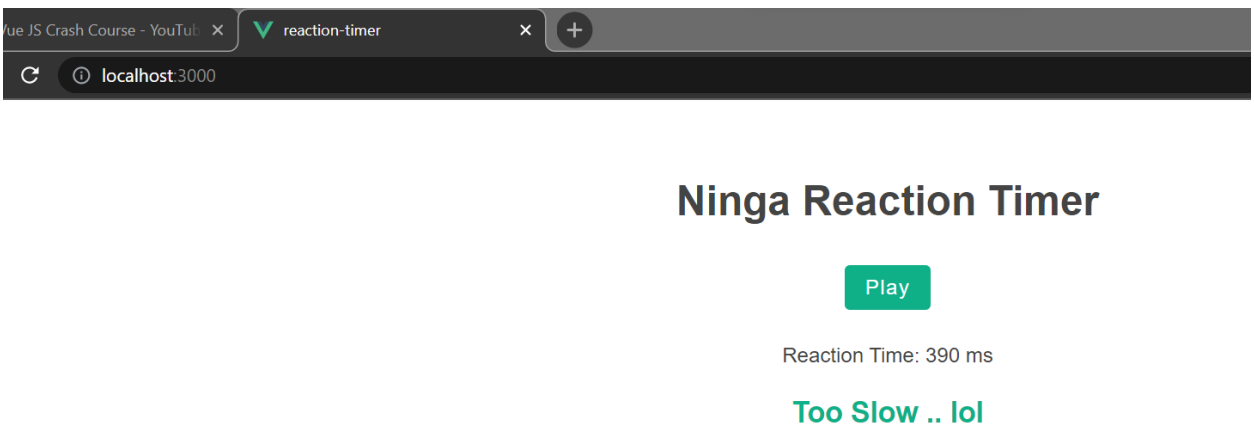
Tip: Testing the application to run from your build file from VS Code
This allows you to test your dist folder's code before you deploy
Run this command:
npm I -g serve

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    JUPYTER    GITLENS

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\DevProjects\Programming\vueProjects\reaction-timer> npm i -g serve
C:\Users\lione\AppData\Roaming\npm\serve -> C:\Users\lione\AppData\Roaming\npm\node_modules\serve\bin\serve.js
+ serve@13.0.2
added 90 packages from 40 contributors in 5.284s
PS C:\DevProjects\Programming\vueProjects\reaction-timer>
```

Then run the following command:
Serve the dist folder
serve -s dist

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    JUPYTER    GITLENS

C:\Users\lione\AppData\Roaming\npm\serve -> C:\Users\lione\AppData\Roaming\npm\node_modules\serve\bin\serve.js
+ serve@13.0.2
added 90 packages from 40 contributors in 5.284s
PS C:\DevProjects\Programming\vueProjects\reaction-timer> serve -s dist

   Serving!

   - Local:            http://localhost:3000
   - On Your Network:  http://10.241.8.165:3000

   Copied local address to clipboard!
```

And it runs



Tip: AXIOS with VUE

https://v2.vuejs.org/v2/cookbook/using-axios-to-consume-apis.html?redirect=true

minute. First, we'd install axios with either npm/yarn or through a CDN link.

There are a number of ways we can request information from the API, but it's nice to first find out what the shape of the data looks like, in order to know what to display. In order to do so, we'll make a call to the API endpoint and output it so we can see it. We can see in the CoinDesk API documentation, that this call will be made to `https://api.coindesk.com/v1/bpi/currentprice.json`. So first, we'll create a data property that will eventually house our information, and we'll retrieve the data and assign it using the `mounted` lifecycle hook:

JS

```
new Vue({
  el: '#app',
  data () {
    return {
      info: null
    }
  },
  mounted () {
    axios
      .get('https://api.coindesk.com/v1/bpi/currentprice.json')
      .then(response => (this.info = response))
  }
})
```

HTML

```
<div id="app">
  {{ info }}
</div>
```