Using Vue with ASP.NET Core By Example

Lab 7 Instructions

Shawn Wildermuth, Instructor

1. Add Webpack to the Project
    a. Open the **package.json** file.
    b. Create a new section after '**dependencies'** called '**devDependencies'**:

```
{
  "version": "1.0.0",
  "name": "asp.net",
  "private": true,
  "dependencies": {
    "jquery": "~3.3.1",
    "popper.js": "1.14.1",
    "bootstrap": "4.0.0",
    "bootswatch": "4.0.0",
    "@fortawesome/fontawesome-free-webfonts": "1.0.5",
    "vue": "2.5.16",
    "axios": "0.18.0",
    "vee-validate": "2.0.8",
    "vuex": "3.0.1",
    "vue-router": "3.0.1"
  },
  "devDependencies": {
  }
}
```

    c. Inside the **devDependendies**, add **webpack** and **webpack-cli** packages:

```
  "devDependencies": {
    "webpack": "^4.9.1",
    "webpack-cli": "^2.1.4"
  }
```

    d. Open a **console** and type "**webpack --version**" to see if it is install correctly.

```
D:\courses\vue-webpage\VueByExampleCourse\Labs\Lab7\after>webpack --version
4.9.1
```

    e. Back in Visual Studio, create a new file at the **root** of the project called **webpack.config.js**.
    f. Export an empty **object** in this new file:

```
// webpack.config.js
module.exports = {
}
```

g. Add an **entry** pointing at the **app.js**:

```
// webpack.config.js
module.exports = {
  entry: "./wwwroot/js/app.js"
}
```

h. Add a **mode** to specify **development**:

```
// webpack.config.js
module.exports = {
  entry: "./wwwroot/js/app.js",
  mode: 'development'
}
```

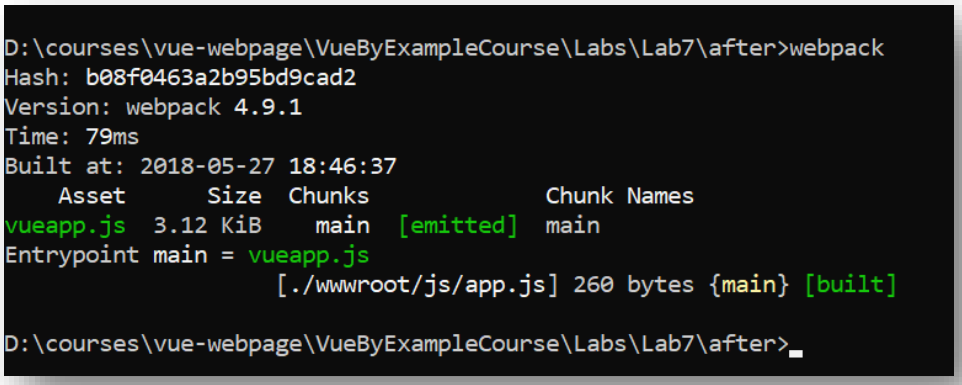
i. Add a new variable at the top called **path** and use **require** to get the **path** object:

```
// webpack.config.js
let path = require("path");

module.exports = {
  entry: "./wwwroot/js/app.js",
  mode: 'development'
}
```

j. Finally, add an **output** to point at a **dist** folder in **wwwroot**:

```
// webpack.config.js
let path = require("path");

module.exports = {
  entry: "./wwwroot/js/app.js",
  mode: 'development',
  output: {
    filename: 'vueapp.js',
    path: path.resolve(__dirname, './wwwroot/dist')
  }
}
```

k. Go back to console, and type **webpack** again and see if create your first web package:



l. Finally, open the **/Views/Root/Index.cshtml**.

m. Replace all the **scripts** with the single **vueapp.js** file:

```
@section scripts {
  <script src="~/dist/vueapp.js"></script>
}
<div id="index-view">
  <router-view></router-view>
</div>
```

n. If you run the project now, it won't work. Let's get it working next.

2. Convert to Module Loading
    a. Open the **app.js** file in **/wwwroot/js**.
    b. Import **Vue**, **VueRouter** and **VeeValidate** to enable them to be used (and hint webpack that it needs them):

```
// app.js
import Vue from "vue";
import VueRouter from "vue-router";
import VeeValidate from "vee-validate";
```

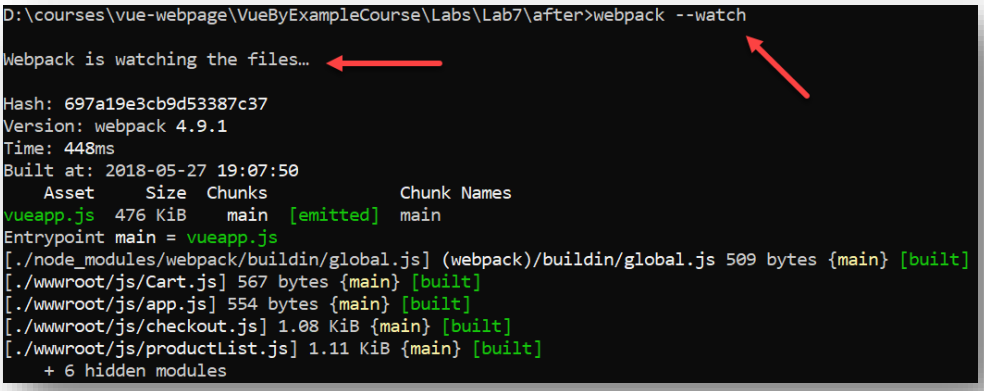
    c. Import the two views and the **DataStore** using the relative syntax:

```
// app.js
import Vue from "vue";
import VueRouter from "vue-router";
import VeeValidate from "vee-validate";

import ProductList from "./productList";
import Checkout from "./checkout";
import store from "./store";

Vue.use(VueRouter);
Vue.use(VeeValidate);
```

    d. Open the console and type **webpack** to see if it loads more than the one file like last time:

```
D:\courses\vue-webpage\VueByExampleCourse\Labs\Lab7\after>webpack
Hash: 8292ae5eab290ee20641
Version: webpack 4.9.1
Time: 425ms
Built at: 2018-05-27 18:53:02
    Asset     Size  Chunks             Chunk Names
vueapp.js  474 KiB    main  [emitted]  main
Entrypoint main = vueapp.js
[./node_modules/webpack/buildin/global.js] (webpack)/buildin/global.js 509 bytes {main} [built]
[./wwwroot/js/app.js] 444 bytes {main} [built]
[./wwwroot/js/checkout.js] 1.06 KiB {main} [built]
[./wwwroot/js/productList.js] 1.09 KiB {main} [built]
    + 6 hidden modules
```

e. Add the **--watch** flag to have it compile as we change the files:



f. Open the **productList.js** file.
g. Import the **Vue** object like we did in the **app.js**.
h. Change the creation of the object to **export default** so that the import in **app.js** returns this object instead of having a global object:

```
// productList.js
import Vue from "vue";

export default Vue.component("product-list", {4
```

i. Open the **checkout.**js and repeat this same behavior:

```
// checkout.js
import Vue from "vue";

export default Vue.component("checkout", {
```

j. Open the **store.js**.
k. Repeat this again:

```
// store.js
import Vuex from "vuex";

export default new Vuex.Store({
```

l. Add importing **axios** since the **store** needs that object too:

```
// store.js
import Vuex from "vuex";
import axios from "axios";

export default new Vuex.Store({
```

m. Import Vue as well so we can use the Vuex to add the plugin:

```
// store.js
import Vue from "vue";
import Vuex from "vuex";
import axios from "axios";

Vue.use(Vuex);
```

n. Open the **cart.js** and repeat it again:

```
// cart.js
import Vue from "vue";

export default Vue.component("the-cart", {
```

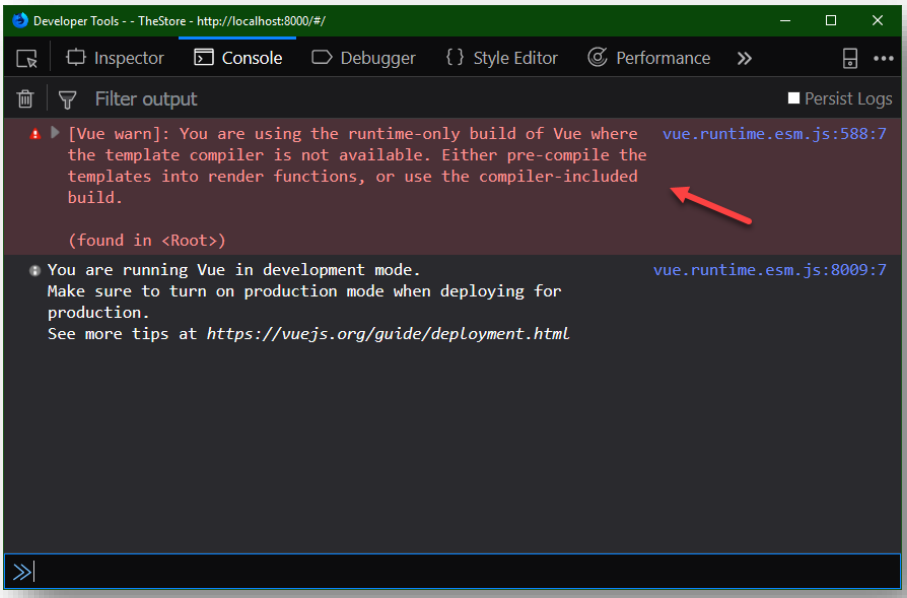
o. In order to import it, open the **app.js** file.
p. Add **Cart** as an imported object:

```
// app.js
import Vue from "vue";
import VueRouter from "vue-router";
import VeeValidate from "vee-validate";

import ProductList from "./productList";
import Checkout from "./checkout";
import DataStore from "./datastore";
import Cart from "./Cart";
```

q. In the **Vue** object, add a new **component** section and set the **Cart** in there:

```
var vm = new Vue({
  el: "#index-view",
  store: DataStore,
  components: {
    Cart
  },
  router: new VueRouter({ routes })
});
```

r. If you run the project, you'll see it still fails to display.
s. Hit F12 and look at the console to see a problem with the runtime version of Vue:



The version of Vue you're using can't compile the templates. So we need to change which version of Vue the imports are finding.

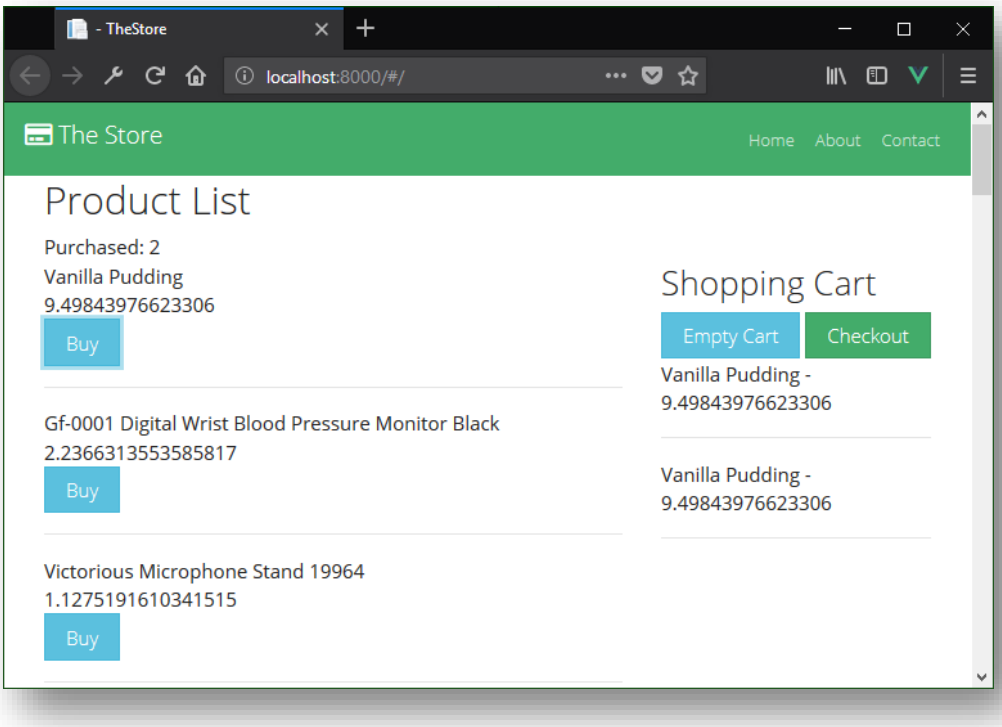
t. Open the web.config.js file.

u. Add a section called resolve and create an alias for anything ending with Vue:

```
// webpack.config.js
let path = require("path");

module.exports = {
  entry: "./wwwroot/js/app.js",
  mode: 'development',
  output: {
    filename: 'vueapp.js',
    path: path.resolve(__dirname, './wwwroot/dist')
  },
  resolve: {
    alias: {
      "vue$": 'vue/dist/vue.esm.js'
    }
  }
}
```

v. Open the console again and stop and restart '**webpack --watch**'. Watch only sees changes in your files, not in the configuration file itself.

w. Run it again and you should see your application work in the browser:



3. Add Babel and use ECMAScript
    a. Open the **package.json**
    b. Add the **babel** packages we need to the **devDependencies**:

```
  "devDependencies": {
    "webpack": "^4.9.1",
    "webpack-cli": "^2.1.4",
    "babel-loader": "7.1.4",
    "babel-core": "6.26.3",
    "babel-preset-env": "1.6.1"
  }
```

    c. In order to have babel compile our project to ES5, you'll need to specify to use the babel loader.
    d. Start by opening the **webpack.config.js** file.
    e. Add a new **module** section:

```
// webpack.config.js
let path = require("path");

module.exports = {
  entry: "./wwwroot/js/app.js",
  mode: 'development',
  output: {
    filename: 'vueapp.js',
    path: path.resolve(__dirname, './wwwroot/dist')
  },
  resolve: {
    alias: {
      "vue$": 'vue/dist/vue.esm.js'
    }
  },
  module: {

  }
}
```

    f. Then add a new section inside **module** called **rules** which is an **array** of rules:

```
  module: {
    rules: [
    ]
  }
```

g. Finally, add a **rule** for babel specifying:
   i. Test to find .js files
   ii. Exclude to stop it from running babel through any NPM packages.
   iii. Specify the loader as the babel loader.
   iv. And specify options from the environment:

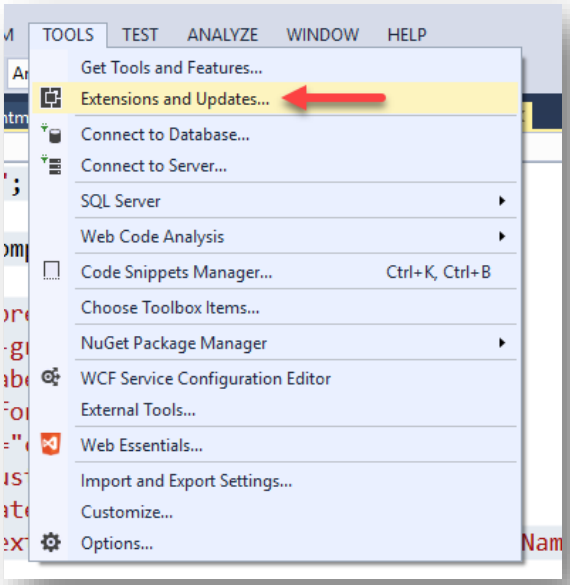
```
module: {
  rules: [
    {
      test: /\.js$/,
      exclude: /node_modules/,
      loader: "babel-loader",
      options: {
        presets: ['env']
      }
    }
  ]
}
```

h. Open the checkout.js.
i. Replace the ES5 usage with some ES6 usage knowing that babel will support the syntax on all browsers:

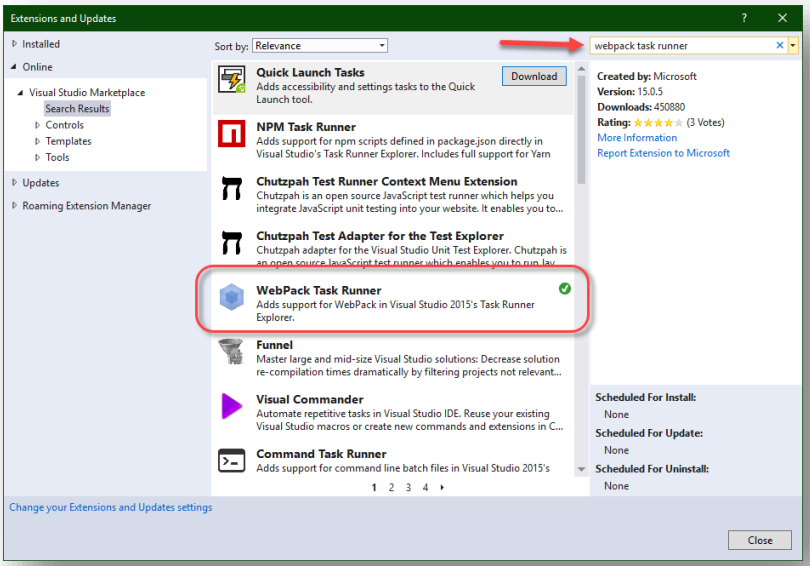
```
// checkout.js
import Vue from "vue";

export default Vue.component("checkout", {
  template: `...`,
  data() {
    return {
      customer: {}
    };
  },
  methods: {
    onSave() {
      alert(JSON.stringify(this.customer));
    }
  }
});
```

j. Stop and restart webpack again.
k. Run this to see the website is still working as expected.

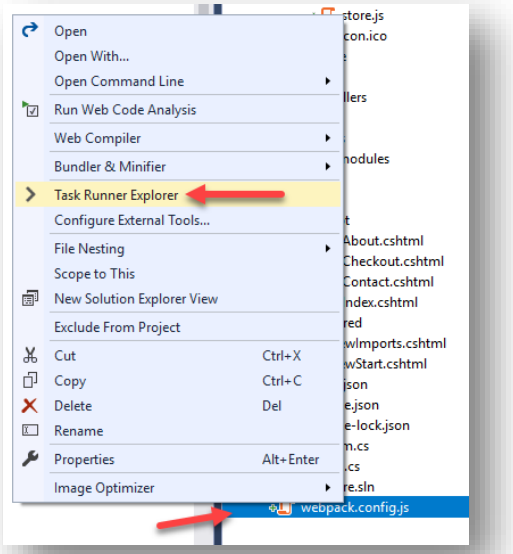
4. Add Webpack support in Visual Studio
    a. Open the **Extensions and Updates** in the **Tools** menu:



    b. Search for "**webpack task runner**" to enable Visual Studio to run webpack for you:

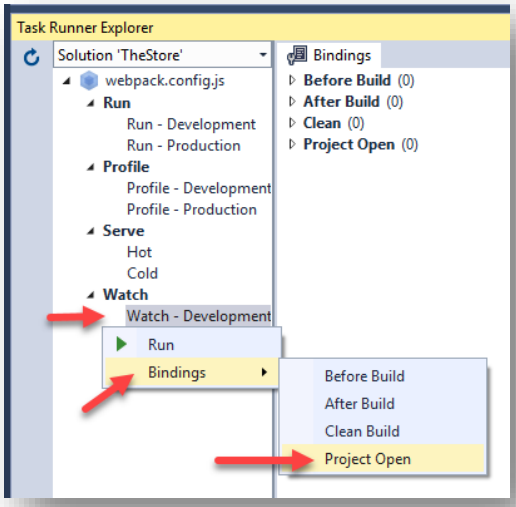


c. Once that is installed, right-click the **webpack configuration file** and pick "**Task Runner Explorer**":



d. Find the Development version of Watch in the Task Runner Explorer.

e. Right-click the "**Watch - Development**", and pick "Project Open" to have webpack watch run when the project is open:



f. Finally, double cick the “Watch – Development” to make it start running: