Set up ASP.Net MVC project with React (es6) Instructions:

These instructions will help you setup and install the necessary components for the Mars on-boarding project

Note: Please don't use Semantic-ui-react. Use Semantic ui

Installing React and the other required packages

- Download and install node.js - https://nodejs.org/en/
  - node.js will install the Node Package Manager (npm). We will be using npm to install react and the other packages that are required
- Create a new ASP.NET MVC project
- Create a file called ***react*** under the ***scripts*** folder in the solution explorer
  - This folder will contain all the required files for the project
- Open the command prompt (cmd)
- In the cmd, navigate to the react folder location in the project
  - You can use the **cd** command to navigate to the desired location
- type ***npm --version*** or ***npm*** in the command prompt to check that you installed node.js properly. If not you need to reinstall node.js
- Type ***npm init*** into the cmd (You can use ***npm init -y*** as well for default values.Warning: if default name is ***react it*** can create issues. To fix the issue, choose a different name)
  - Type a name. E.g. mars_onboarding
  - Press enter for everything else
- Install react, babel and webpack (Make sure you are in the react folder in the cmd when installing):
  - npm install react react-dom --save-dev
  - npm install webpack --save-dev
  - **(If prompted would you like to install webpack-cli? Type yes)**
  - npm install --save-dev babel-loader babel-core babel-preset-react babel-preset-env (Don't do this line, do the next line)
  - npm install @babel/core babel-loader @babel/preset-env @babel/preset-react --save-dev
- **Note:** Please look up what ECMAscript 6 (required for react. The latest version of javascript), webpack, react and babel is. It is important to know what these items are since you will be using them. You don't need to know babel, webpack in depth but it is good to know what these are used for.
- In your project, click on the react folder and on the top of the solution explorer there is a button called show all files. Click on that
- Right click on the package.json file and include it in the project

- Create a file called webpack.config.js in the react folder. Copy into file:

```
module.exports = {
      mode: "development",
      context: __dirname,
      entry: {
            customer: "./customer.js",
            product: "./product.js"
      },
      output: {
            path: __dirname + "/dist",
            filename: "[name]_bundle.js"
      },
      watch: true,
      module: {
            rules: [
                  {
                        test: /\.js$/,
                        exclude: /(node_modules)/,
                        use: {
                              loader: 'babel-loader',
                              options: {
                                    presets: ['@babel/preset-env',
'@babel/preset-react']
                              }
                        }
                  }
            ]
      }
}
```

- Create a folder called ***dist*** in the react folder
- Create ***customer.js*** in the react folder
- Some very useful resources (including the tutorials on installing react) are down below. It is recommended that you look at those tutorials and resources
- In the package.json file, type (in the scripts section above **"test":**):

```
"build": "webpack",
```

- The scripts section should look like the one below:

```
"scripts": {
    "build": "webpack",
    "test": "echo \"Error: no test specified\" && exit 1"
  }
```

- To compile your react code, you must open the command prompt, navigate to the react folder and type ***npm run build***. You only need to do this once since there is ***watch: true*** in webpack.config.js. Watch: true means that whenever you save a file in your project, the react code will re-compile again without you having to type npm run build again. Sometimes when adding to the webpack.config.js file, you may need to cancel the command and type npm run build again to include those files in the project
- When you compile your react code, you will see that some files are created in the dist folder. These are the compiled react code (compiled from ES6 Javascript to an earlier version of Javascript that the browser can understand.) Include these files in your view to include the react code. Further learning of React and webpack will be required to understand the process
- To include React:

```
import React from 'react';
import ReactDOM from 'react-dom';
```

- Include the components you want within the braces

Installing and setup for CSS Modules (Optional)

- If you want to learn about CSS Modules, you can try and follow these instructions. Not required since we will be using semantic-ui for our styling
- Type commands in the console. Make sure to be in the react folder of your project:
    - npm install css-loader --save-dev
    - npm install style-loader --save-dev
- In the webpack.config.js file, add a new section in the rules:

```
,{
test: /\.css$/,
      loader: 'style-loader'
},
{
      test: /\.css$/,
      loader: 'css-loader',
      query: {
            modules: true,
            localIdentName: '[name]__[local]___[hash:base64:5]'
```

```
        }
}
```

Resources

- Setting up React with ASP.NET: https://www.youtube.com/watch?v=bnFgGYooDCM&t=859s
- Setting up an ES6 environment for ASP.NET MVC 5: https://www.slightedgecoder.com/2017/05/22/setting-es6-environment-asp-net-mvc-5/
- Installing CSS modules in react and webpack: https://www.youtube.com/watch?v=u9LBSeeJzXc
- Webpack entry points (Important: You need this to understand how to do multiple pages for a react project and using when webpack): https://webpack.js.org/concepts/entry-points/
- Understand how forms work in React: https://reactjs.org/docs/forms.html
- Lifecycle of a react app (Very important): https://engineering.musefind.com/react-lifecycle-methods-how-and-when-to-use-them-2111a1b692b1
- Helped fix some issues: https://www.valentinog.com/blog/react-webpack-babel/