

How to create React project?

This document introduces 3 ways of creating react application with .net.

Method 1: using create-react-app

Method 2: using .net framework 4.6 with react(if you're not familiar with react and core 2, use this method)

Method 3: [using core 2 with react\(Mars project structure\)](#)

You properly will use method 2 for the onboarding task.

Pre-requirement:

- [Node.js](#) (npm installed on the local machine)
- Visual studio **2017**/ Visual studio code

Basic commands:

- cd -> change directory
- cd .. -> back to previous directory
- Dir -> display current directory
- Mkdir -> create new file in directory
- --help -> lists and briefly describes every system command.

Note:

Cli = command line language

Npm = node package manager

Npx = [take a look at this link](#)

[Babel](#) = convert old version JavaScript to es6(new version of javascript)

[Webpack](#) = bundling packages

*When you want to run the package in npm/ use the npm command, you simply open the CMD and type "npm xxx xxx". When you want to run dotnet app, simply open the cmd and type "dotnet xxx xxx"

Tools for react development:

[React-chrome](#)

Great online resources:

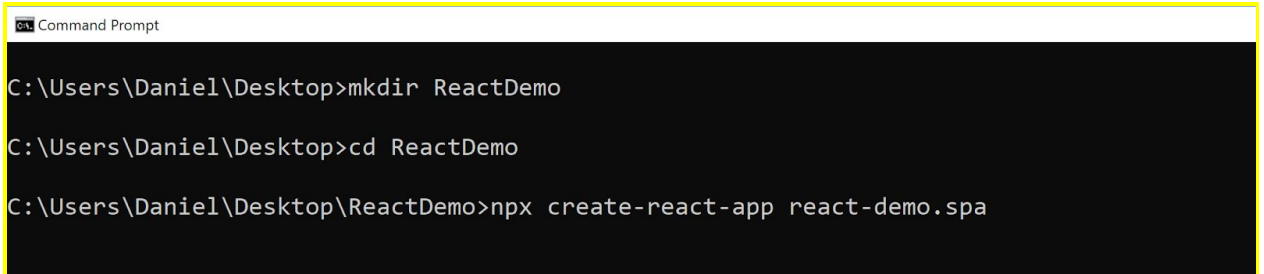
[react](#)

Method 1: Using Create-react-app (perfect with VSCode)

Using [create-react-app](#) that provided by facebook

1. Open cmd and navigate to the directory that you like the create the app, in this case I chose desktop.
2. Make a new directory name “reactDemo” and cd to this directory.
3. Type “npx create-react-app your-app-name.spa”, press enter.(spa means single page application)

(We've created a react SPA here and we will have to create the API after)



```
Command Prompt

C:\Users\Daniel\Desktop>mkdir ReactDemo

C:\Users\Daniel\Desktop>cd ReactDemo

C:\Users\Daniel\Desktop\ReactDemo>npx create-react-app react-demo.spa
```

4. After this command, the npm will automatically download the template for you. And you can see all available command for npm.

```
npm start
  Starts the development server.

npm run build
  Bundles the app into static files for production.

npm test
  Starts the test runner.

npm run eject
  Removes this tool and copies build dependencies, configuration files
  and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:

  cd react-demo.spa
  npm start

Happy hacking!
```

5. Now create the Core 2 API. Using the same CMD, type “dotnet new -i yourappname”. You will see result as ablow:

| Templates | Short Name | Language | Tags |
|--|-------------|--------------|---------------------|
| Console Application | console | [C#], F#, VB | Common/Console |
| Class library | classlib | [C#], F#, VB | Common/Library |
| Unit Test Project | mstest | [C#], F#, VB | Test/MSTest |
| xUnit Test Project | xunit | [C#], F#, VB | Test/xUnit |
| ASP.NET Core Empty | web | [C#], F# | Web/Empty |
| ASP.NET Core Web App (Model-View-Controller) | mvc | [C#], F# | Web/MVC |
| ASP.NET Core Web App | razor | [C#] | Web/MVC/Razor Pages |
| ASP.NET Core with React.js with ES6 | reactes6 | [C#] | Web/MVC/SPA |
| ASP.NET Core with Angular | angular | [C#] | Web/MVC/SPA |
| ASP.NET Core with React.js | react | [C#] | Web/MVC/SPA |
| ASP.NET Core with React.js and Redux | reactredux | [C#] | Web/MVC/SPA |
| ASP.NET Core Web API | webapi | [C#], F# | Web/WebAPI |
| global.json file | globaljson | | Config |
| NuGet Config | nugetconfig | | Config |
| Web Config | webconfig | | Config |
| Solution File | sln | | Solution |
| Razor Page | page | | Web/ASP.NET |
| MVC ViewImports | viewimports | | Web/ASP.NET |
| MVC ViewStart | viewstart | | Web/ASP.NET |

Examples:

```
dotnet new mvc --auth Individual
dotnet new classlib --framework netcoreapp2.0
dotnet new --help
```



6. Type “dotnet new webapi -o yourappname.api” to create an API.

```
C:\Users\Daniel\Desktop\ReactDemo>dotnet new webapi -o react-deom.api
```

7. Make sure you are in the root folder which is ReactDemo. Then type “code .” to open visual studio code with 2 folders.

◀ **REACTDEMO**

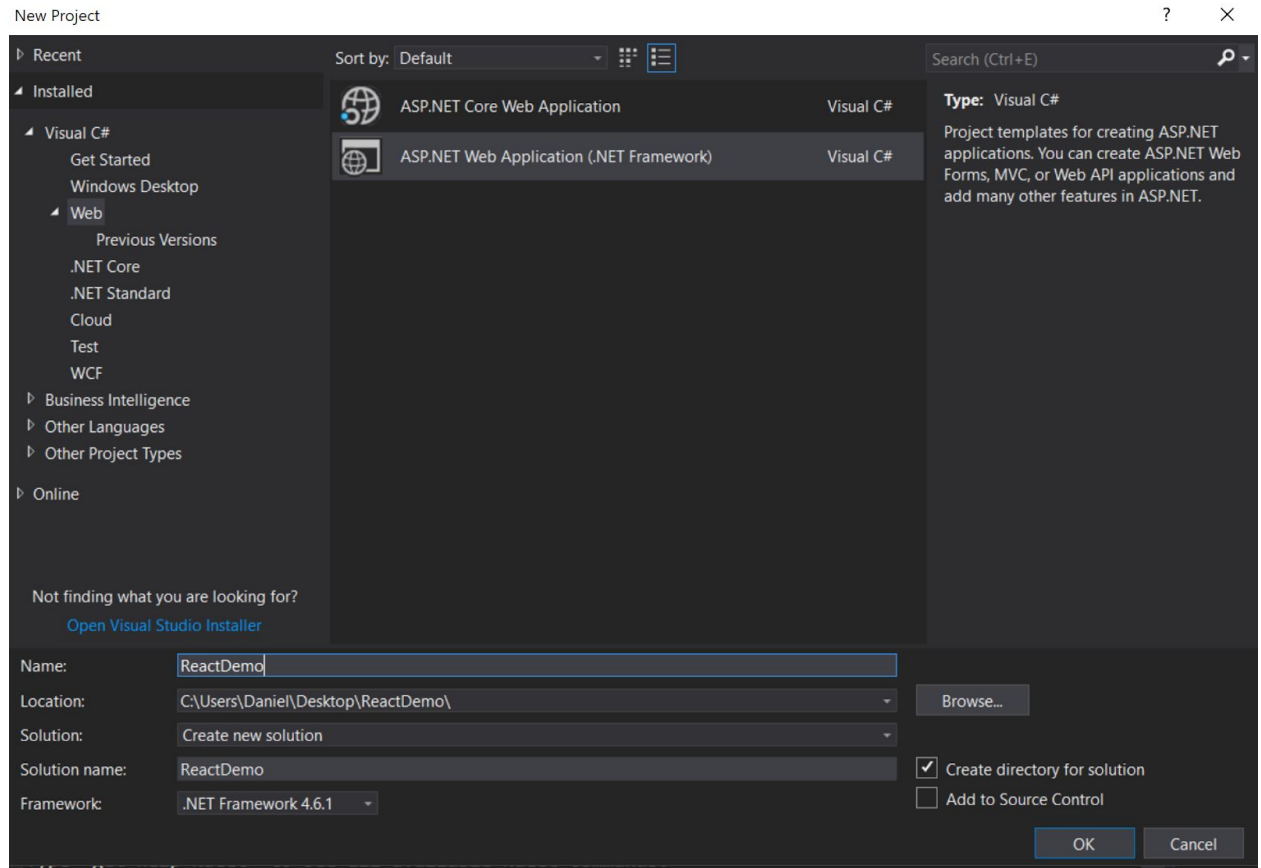


- ▶  react-demo.spa
- ▶  react-deom.api

Method 2: Using .net framework 4.xx with MVC(VS 2017)

(this is a little bit complex)

1. Create a new project, select the .NET framework, click next.



2. Select MVC, click ok.
3. Expand the Scripts folder and create a new folder call "react".
4. Open the CMD and "cd" to the react folder directory.

5. Type “npm init”. This will help you to create a package.json which help you save all dependencies(packages) that you’ll have to use.

```
C:\Users\Daniel\Desktop\ReactDemo\ReactDemo\ReactDemo\Scripts\react>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

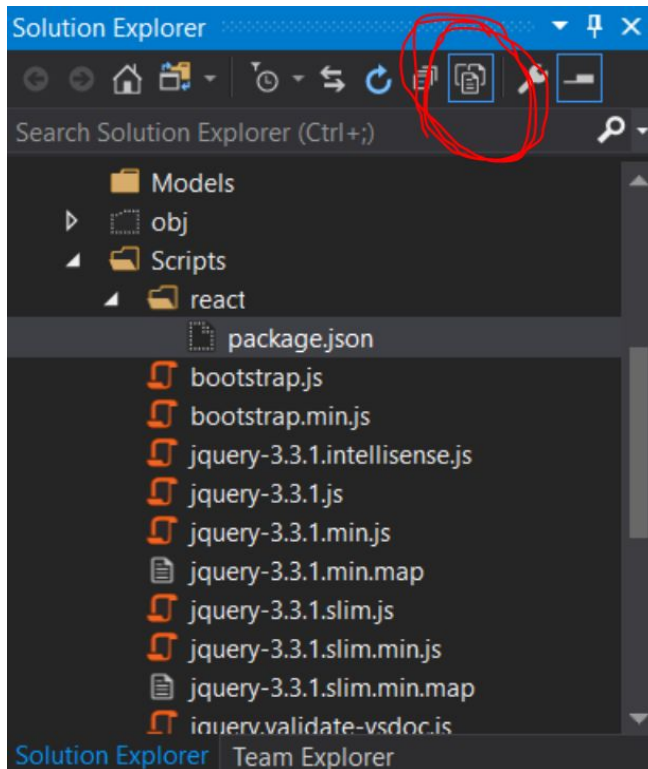
Press ^C at any time to quit.
package name: (react)
```

6. Rename the package name to something else, i rename it to “react-onboarding”, press enter, keep pressing enter until you see entry point. Entry point is the entry point that frontend would looking for. You can either keep index.js or change that.
7. Press enter, until you see the author, put your name in and press enter.

```
Press ^C at any time to quit.
package name: (react) react-onboarding
version: (1.0.0)
description:
entry point: (index.js) app.js
test command:
git repository:
keywords:
author: daniel
license: (ISC)
About to write to C:\Users\Daniel\Desktop\ReactDemo\ReactDemo\ReactDemo\Scripts\react\package.json:
{
  "name": "react-onboarding",
  "version": "1.0.0",
  "description": "",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "daniel",
  "license": "ISC"
}

Is this OK? (yes)
```

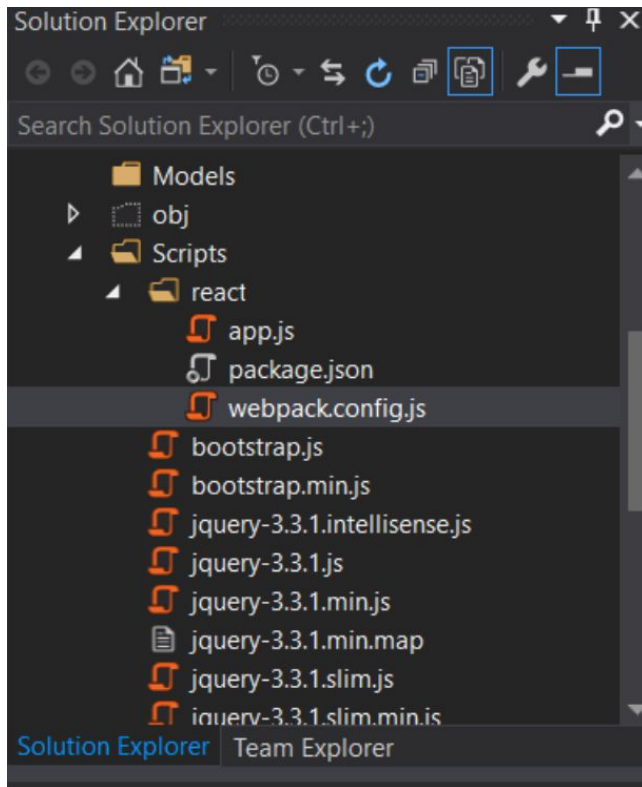
8. Go back to VS 2017, you will see nothing showing up even you created the new file.



- 9.
10. Click the show all files icon. Right click package.json and select “include in project”.
11. Back to CMD, type “`npm install --save-dev react react-dom`”.
12. After the installation, in CMD, type “`npm i @babel/core babel-loader @babel/preset-env @babel/preset-react --save-dev`”

```
npm install webpack webpack-dev-server webpack-cli --save -g
```

13. Back to VS 2017, you will see lots of changes in package.json file. Under react folder create 2 javascript files; "app.jsx" and "webpack.config.js"



*.js and .jsx are similar, if you want to use .js, change the webpack setting to .js otherwise use .jsx. The webpack will convert that for you, so it really not that matter! Add below script into webpack.config.js:

```
module.exports = {
  context: __dirname,
  mode: 'development',
  entry: {
    app: "./app.js",
  },
  output: {
    path: __dirname + "/dist",
    filename: "[name].bundle.js"
  },
  watch: true,
  resolve: {
    extensions: [".jsx", ".js"]
  },
}
```



```

module: {
  rules: [
    {
      test: /\.jsx?$/,
      exclude: /(node_modules)/,
      use: {
        loader: 'babel-loader',
        options: {
          presets: ['@babel/preset-env', '@babel/preset-react']
        }
      }
    }
  ]
}
};

```

14. Open package.json and change to(watch and wbp):

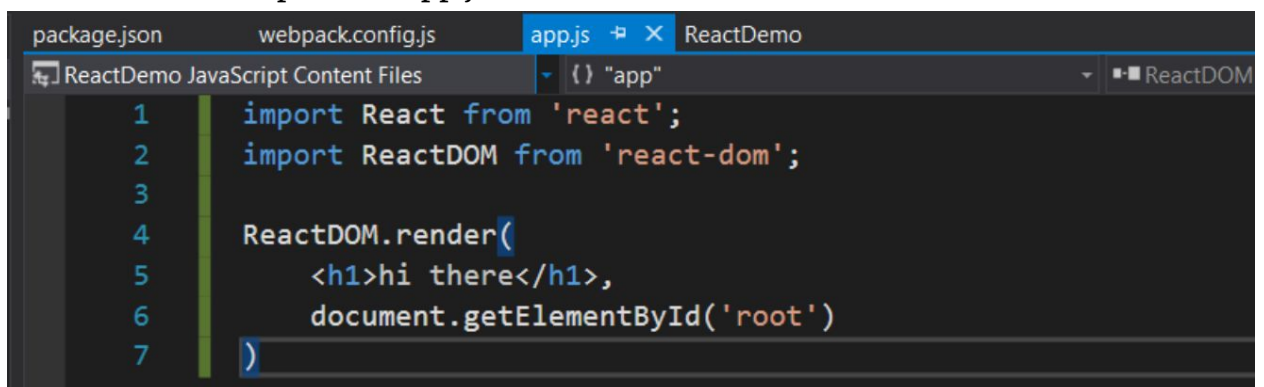
```

"scripts"
  "test" "echo \"Error: no test specified\" && exit 1"
  "wbp" "webpack --config webpack.config.js"

"watch" true

```

- 15.
16. Go to CMD, run the webpack by putting the command: “npm run wbp”.
17. The CMD would ask you to install one more package, just type “webpack-cli” and press enter. And the webpack is running.
18. Put some react script in the app.js file:



```

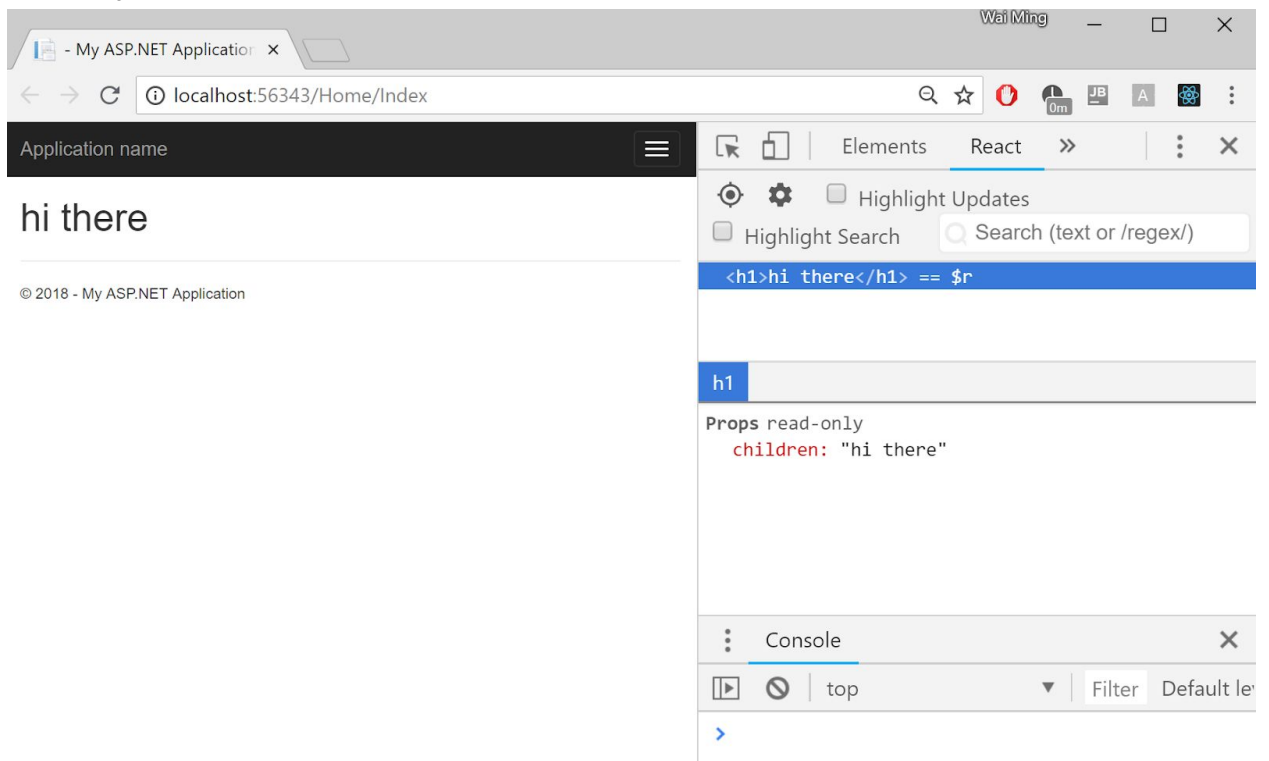
package.json  webpack.config.js  app.js  ReactDemo
ReactDemo JavaScript Content Files  {} "app"  ReactDOM
1  import React from 'react';
2  import ReactDOM from 'react-dom';
3
4  ReactDOM.render(
5    <h1>hi there</h1>,
6    document.getElementById('root')
7  )

```

19. Navigate to Views -> Home -> Index.cshtml. Change code to:

```
Index.cshtml  ApplicationInsights.config  package.json  webpack.config.js  app.js  ReactDemo
1  <div id="root"></div>
2
3  @section scripts {
4
5      <script src="~/Scripts/react/dist/bundle.js"></script>
6  }
```

20. Run the application. If you see the following, means you are setup the react correctly.



Method 3: Using .net core 2 with MVC(VS 2017)

(Please do a manual update for the babe by yourself, otherwise, it may not work)

This required you have a bit knowledge in core 2.

[using core 2 with react\(Mars project structure\)](#)

NPM instal commands:

- npm install --save-dev react react-dom react-tag-input
- npm install --save-dev webpack
- npm install --save-dev babel-loader babel-core babel-preset-react babel-preset-env

Code in webpack.config.js:

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

}
}
]