[Creating a React Application 2](#_Toc514168993)

[Development Environment contains: 2](#_Toc514168994)

[How to use Node commands in Windows Installation. 2](#_Toc514168995)

[How to install React 2](#_Toc514168996)

[Bootstrap 4](#_Toc514168997)

[Gulpjs 4](#_Toc514168998)

[Appendix 17](#_Toc514168999)

[APP.JS 17](#_Toc514169000)

[APP.TEST.JS 18](#_Toc514169001)

[INDEX.JS 18](#_Toc514169002)

[GULPFILE.JS 18](#_Toc514169003)

[INDEX.HTML 19](#_Toc514169004)

[\_CONFIG.SCSS 20](#_Toc514169005)

[DEFAULT.SCSS 22](#_Toc514169006)

[FOOTER.JS 23](#_Toc514169007)

[HEADER.JS 24](#_Toc514169008)

[CONTACT.JS 25](#_Toc514169009)

[HOMEPAGE.JS 25](#_Toc514169010)

[PRODUCTS.JS 26](#_Toc514169011)

# Creating a React Application

## Development Environment contains:

1. Nodejs.org
2. Reactjs.org
3. Gulpjs.com
4. Sass-lang.com
5. Bootstrap

#### How to use Node commands in Windows Installation.

To use the Node command open start and use : Nodejs Command Prompt which is created after Nodejs installation on your localhost.

Create for example on Desktop (or wherever you like) a folder for our application which will contain all the files of our React App.

Mine is ReactTutorial and I created it on my Desktop.

#### How to install React

Open : [add-react-to-a-new-app](https://reactjs.org/docs/add-react-to-a-new-app.html)

In the NodeJS command prompt we shall change directory and go to our ReactTutorial folder we just created on our Desktop.

My path is : C:\Users\Xhani\Desktop\ReactTutorial>

Now we can Globally install our ReactJS (you only need to install this once) :

Npm install –g create-react-app

Now we are going to create our react app

Create-react-app my-react-project

My-react-project in the command above is the title I want to give to my app (**shall be only lower case, and valid characters**)

The above command will install all the necessary dependencies and prepare the project for us.

After the creation of the project folder and files is done, we navigate to our newly created folder

C:\Users\Xhani\Desktop\ReactTutorial>cd my-react-project

C:\Users\Xhani\Desktop\ReactTutorial\my-react-project>npm start

After completion it will start our server and it will open in our Browser our app

Then we can navigate to our src folder: my-react-project\src and delete App.css, Logo.svg, index.css files.

Navigate to our public folder inside my-react-project :

Here we see favicon.ico & index.html files. (we can update our favicon file later on with whatever we would like)

Later we open our Editor (in my case I am using Visual Studio Code) and open our App.js file and delete the code from:

<div className=”App-header”>

Up to:

To get started, edit <code>src/App.js</> and save to reload

</p>

Remove from the top of the file the line:

Import logo from ‘./logo.svg’;

Import ‘./App.css’;

Later we go to Index.js file and delete:

Import ‘./index.css’;

At this point we can see that on our Application opened in the Browser we see no more errors but just a blank page.

This is good!!!

Go on to the next part of the project.

#### Bootstrap

We navigate to the page: https[://getbootstrap.com/docs/4.1/getting-started/download/](https://getbootstrap.com/docs/4.1/getting-started/download/)

We go the Bootstrap CDN part and copy the link :

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.0/css/bootstrap.min.css" integrity="sha384-9gVQ4dYFwwWSjIDZnLEWnxCjeSWFphJiwGPXr1jddIhOegiu1FwO5qRGvFXOdJZ4" crossorigin="anonymous">

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.0/js/bootstrap.min.js" integrity="sha384-uefMccjFJAIv6A+rW+L4AHf99KvxDjWSu1z9VI8SKNVmz4sk7buKt/6v9KI65qnm" crossorigin="anonymous"></script>

Later on our Editor we go to our public folder and to our Index.html file and paste the above code

Now because we want to use SASS we need to use a task runner (gulpjs) to compile the CSS

#### Gulpjs

We can go to <https://gulpjs.com/> and run the following commands:

Npm install gulp-cli –g

Npm install gulp –D

Create a file named : gulpfile.js in the project’s directory (the path to my file)

C:\Users\Xhani\Desktop\ReactTutorial\my-react-project\gulpfile.js

Later on we will need to install some gulp plugins :

Gulp-sass

Site: <https://www.npmjs.com/package/gulp-sass>

npm i gulp-sass

npm install gulp-sass --save-dev

Gulp-clean-css

Site: <https://www.npmjs.com/package/gulp-clean-css>

npm i gulp-clean-css

npm install gulp-clean-css --save-dev

Gulp-uglify

Site: <https://www.npmjs.com/package/gulp-uglify>

npm i gulp-uglify

npm install --save-dev gulp-uglify

Gulp-rename (0 dependencies) that’s why we use npm I gulp-rename

Site: <https://www.npmjs.com/package/gulp-rename/>

npm i gulp-rename

Gulp-changed

Site: <https://www.npmjs.com/package/gulp-changed>

npm i gulp-changed

npm install --save-dev gulp-changed

Installing SASS :

Npm install –g sass

In our my-react-project\src folder we are going to create two new folders named:

scss

css

Then in gulpfile.js we are adding this code in order to create the tasks for gulp in order to take the input from scss and outputting to css as well as creating a watch which looks for any changes happening in scss (it works exactly like the supervisor / nodemon for nodejs )

'use strict';

// Dependencies

var gulp = require('gulp');

var sass = require('gulp-sass');

var minifyCSS = require('gulp-clean-css');

var uglify = require('gulp-uglify');

var rename = require('gulp-rename');

var changed = require('gulp-changed');

//////////////

// - SCSS/CSS

//////////////

var SCSS\_SRC = './src/Assets/scss/\*\*/\*.scss';

var SCSS\_DEST = './src/Assets/css';

// Compile SCSS

// pipe allows to chain tasks (gulp method)

gulp.task('compile\_scss', function(){

gulp.src(SCSS\_SRC)

.pipe(sass().on('error', sass.logError))

.pipe(minifyCSS())

.pipe(rename({ suffix: 'min'}))

.pipe(changed(SCSS\_DEST))

.pipe(gulp.dest(SCSS\_DEST));

});

// Detect changes in SCSS

gulp.task('watch\_scss', function(){

gulp.watch(SCSS\_SRC, ['compile\_scss']);

});

// Run tasks

gulp.task('default', ['watch\_scss']);

we can run on terminal the command :

Comment: this command is just what we need to run once as above in the Run tasks section of our code we see that in the function gulp.task we can call the default gulp in order to run the watcher.

gulp

If we have written our code correctly the we should see no errors on the terminal after running gulp.

Then we can create a file in our scss folder we created before and call it :

Default.scss

We add some code in order to see if the gulp.task is working.

NOTE!!! We need to stop first the gulp.task we run before

In Windows terminal for termination of batch process we use : Ctrl + Shift + C

After terminating it we can check by writing sample code in our default.scss file:

body{

font-size: 10px;

}

Then we can see that gulp creates a file for us in the Assets\css folder named default.min.css

body{font-size:10px}

So we can see gulp is working perfectly looking for any changes in the scss folder files.

We need to import our defaultmin.css file path into our App.js. So add this line :

import './Assets/css/default.min.css';

right after the first import of the file.

Now we can go to index.html file and include google fonts for instance from : fonts.google.com

I included the following :

<link href="https://fonts.googleapis.com/css?family=**Open+Sans:400,400i,700,700i**" rel="stylesheet">

Our App.js file will act as a bridge between the React world and the client side of the browser. All components will be put in our App.js file.

We are going to include another folder in our src folder called components

Then in components folder we are going to create another folder called headerComponent

In headerComponent folder we are going to create a new file called header.js

In the header.js file we copy the contents of App.js file (here are the copied contents with the alterations made for the header.js file)

import React, { Component } from 'react';

class Header extends Component {

render() {

return (

<header>

</header>

);

}

}

export default Header;

Then we need to import the header Component we created in the App.js file as this:

// components

import Header from './components/headerComponent/header.js';

In order to call and show the header on our browser we need to add this code inside the divs

<div className="App">

<Header />

</div>

But for now it is showing just empty space because we have not done anything yet in our header.js file.

We can add the following code in yellow letters in our header.js file:

import React, { Component } from 'react';

class Header extends Component {

render() {

return (

<header>

<div className="logo">

LOGO

</div>

<nav>

<ul>

<li>

<a href="#">Home</a>

</li>

<li>

<a href="#">Products</a>

</li>

<li>

<a href="#">Contacts</a>

</li>

</ul>

</nav>

</header>

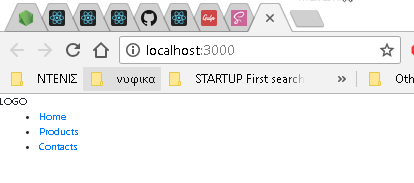
);

}

}

export default Header;

Then we can see on our Browser that the App.js file renders the content in the header.js file and outputs it like this:



Now we would like to add a footer for our app.

We go to our src/components folder and add a new folder footerComponents

And in our footerComponents folder we create a new file called footer.js

Add the following code in footer.js:

import React, { Component } from 'react';

class Footer extends Component {

render() {

return (

<footer>

</footer>

);

}

}

export default Footer;

In App.js file we have to import the footer Component:

import Footer from './components/footerComponent/footer';

and add the following code (in yellow cases) in order to print it in the Browser:

class App extends Component {

render() {

return (

<div className="App">

<Header />

<Footer />

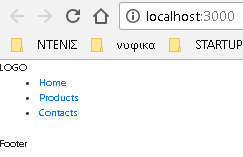
</div>

);

}

}

We write Footer in our footer.js file inside the <Footer></Footer> elements and here is the result it prints out:



Now we are going to create a new Component folder in the components folder called pages

Plus create a new file inside pages folder called homePage.js and add the following code:

import React, { Component } from 'react';

class Homepage extends Component {

render() {

return (

<div className="wrap">

Home Page Content Here

</div>

);

}

}

export default Homepage;

In the App.js file we must import the Homepage component:

import Homepage from './components/pages/homePage';

And add <Homepage /> between the other two Elements (<Header /> and <Footer />)

class App extends Component {

render() {

return (

<div className="App">

<Header />

<Homepage />

<Footer />

</div>

);

}

}

Then we are going to create a new file in scss folder called: \_config.scss and we write the following code:

/\*

vars

\*/

$black: #010000;

$white: #ffffff;

$lightGrey: #e6e6e6;

$darkGrey: #333;

$red: #ff1755;

/\* Fonts -

regular 400

regular 400 italic

bold 700

bold 700 italic

\*/

body {

font-family: 'Open Sans', sans-serif;

font-size: 18px;

color: $black;

font-weight: 400;

font-style: normal;

}

/\*

Mixins

\*/

@mixin translateY($e){

-webkit-transform: translateY($e);

-moz-transform: translateY($e);

transform: translateY($e);

}

@mixin transition($e){

-webkit-transition: $e;

-moz-transition: $e;

transition: $e;

}

Then in the default.scss file we add :

@import "\_config.scss";

header{

position: relative;

display: block;

width: 100%;

background: $black;

color: $white;

height: 90px;

.logo {

position: absolute;

top: 50%; left: 15px;

@include translateY(-50%);

}

nav,

nav ul,

nav ul li{

margin: 0;

padding: 0;

}

nav{

position: absolute;

top: 50%; right: 15px;

@include translateY(-50%);

}

}

In order to style our content areas , Header, Home Page Content, Footer we begin creating css rules in our default.scss file

The final code will be in the Appendix

After Styling we can see how to incorporate a Router for React, for navigating from page to page for our App.

We have to install react-router-dom

Npm i –dev---save react-router-dom

Now we have to import react-router-dom in our App.js file adding this code:

import {

BrowserRouter as Router,

Route,

Link

} from 'react-router-dom';

Next create a products.js file in pages folder

We add the following code in the products.js file:

import React, { Component } from 'react';

class Products extends Component {

render() {

return (

<div className="container-fluid">

<h1>

Products

</h1>

<p>Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.</p>

</div>

);

}

}

export default Products;

We also need to import this component in our App.js file:

import Products from './components/pages/products';

**NOTE**

\*As a result we can see that we must have a different component for every different page we create. \*

Now in the App.js file we don’t need anymore the <Homepage /> element. And we are going to wrap the parent div className=”App” in a Router as well as add code in it:

class App extends Component {

render() {

return (

<Router>

<div className="App">

<Header />

<Route exact path='/' component={Homepage} />

<Route exact path='/Products' component={Products} />

<Footer />

</div>

</Router>

);

}

}

By this way if we enter manually the address localhost:3000/Products in our browser we would be served the page Products.

But because we don’t want to do it manually we are going to make some configurations on our header.js file and import Link (Import link because in React we don’t use Anchor tags)

import {

Link

} from 'react-router-dom';

So the code becomes like this:

import React, { Component } from 'react';

import {

Link

} from 'react-router-dom';

class Header extends Component {

render() {

return (

<header>

<div className="logo">

LOGO

</div>

<nav>

<ul>

<li className="first">

<a href="#">Home</a>

</li>

<li>

<Link to="/Products">Products</Link>

<a href=”#”>Products</a>

</li>

<li className="last">

<a href="#">Contacts</a>

</li>

</ul>

</nav>

</header>

);

}

}

export default Header;

NOTE !!!

As we can se the <Link to=”/Products”>Products</Link> took the place of the anchor tag <a href=”#”>Products</a> because in React the Link compiles to an Anchor tag so it is the same thing but also because anchor tags are not use in React.

# Appendix

## APP.JS

import React, { Component } from 'react';

import {

BrowserRouter as Router,

Route,

Link

} from 'react-router-dom';

// components

import Header from './components/headerComponent/header';

import Footer from './components/footerComponent/footer';

import Homepage from './components/pages/homePage';

import Products from './components/pages/products';

import Contact from './components/pages/contact';

// includes

import './Assets/css/default.min.css';

class App extends Component {

render() {

return (

<Router>

<div className="App">

<Header />

<Route exact path='/' component={Homepage} />

<Route exact path='/Products' component={Products} />

<Route exact path='/Contact' component={Contact} />

<Footer />

</div>

</Router>

);

}

}

export default App;

## APP.TEST.JS

import React from 'react';

import ReactDOM from 'react-dom';

import App from './App';

it('renders without crashing', () => {

const div = document.createElement('div');

ReactDOM.render(<App />, div);

ReactDOM.unmountComponentAtNode(div);

});

## INDEX.JS

import React from 'react';

import ReactDOM from 'react-dom';

import App from './App';

import registerServiceWorker from './registerServiceWorker';

ReactDOM.render(<App />, document.getElementById('root'));

registerServiceWorker();

## GULPFILE.JS

'use strict';

// Dependencies

var gulp = require('gulp');

var sass = require('gulp-sass');

var minifyCSS = require('gulp-clean-css');

var uglify = require('gulp-uglify');

var rename = require('gulp-rename');

var changed = require('gulp-changed');

//////////////

// - SCSS/CSS

//////////////

var SCSS\_SRC = './src/Assets/scss/\*\*/\*.scss';

var SCSS\_DEST = './src/Assets/css';

// Compile SCSS

// pipe allows to chain tasks (gulp method)

gulp.task('compile\_scss', function(){

gulp.src(SCSS\_SRC)

.pipe(sass().on('error', sass.logError))

.pipe(minifyCSS())

.pipe(rename({ suffix: '.min'}))

.pipe(changed(SCSS\_DEST))

.pipe(gulp.dest(SCSS\_DEST));

});

// Detect changes in SCSS

gulp.task('watch\_scss', function(){

gulp.watch(SCSS\_SRC, ['compile\_scss']);

});

// Run tasks

gulp.task('default', ['watch\_scss']);

## INDEX.HTML

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<meta name="theme-color" content="#000000">

<!--

manifest.json provides metadata used when your web app is added to the

homescreen on Android. See https://developers.google.com/web/fundamentals/engage-and-retain/web-app-manifest/

-->

<link rel="manifest" href="%PUBLIC\_URL%/manifest.json">

<link rel="shortcut icon" href="%PUBLIC\_URL%/favicon.ico">

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.0/css/bootstrap.min.css" integrity="sha384-9gVQ4dYFwwWSjIDZnLEWnxCjeSWFphJiwGPXr1jddIhOegiu1FwO5qRGvFXOdJZ4" crossorigin="anonymous">

<link href="https://fonts.googleapis.com/css?family=Open+Sans:400,400i,700,700i" rel="stylesheet">

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.0/js/bootstrap.min.js" integrity="sha384-uefMccjFJAIv6A+rW+L4AHf99KvxDjWSu1z9VI8SKNVmz4sk7buKt/6v9KI65qnm" crossorigin="anonymous">

</script>

<!--

Notice the use of %PUBLIC\_URL% in the tags above.

It will be replaced with the URL of the `public` folder during the build.

Only files inside the `public` folder can be referenced from the HTML.

Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC\_URL%/favicon.ico" will

work correctly both with client-side routing and a non-root public URL.

Learn how to configure a non-root public URL by running `npm run build`.

-->

<title>React App</title>

</head>

<body>

<noscript>

You need to enable JavaScript to run this app.

</noscript>

<div id="root"></div>

<!--

This HTML file is a template.

If you open it directly in the browser, you will see an empty page.

You can add webfonts, meta tags, or analytics to this file.

The build step will place the bundled scripts into the <body> tag.

To begin the development, run `npm start` or `yarn start`.

To create a production bundle, use `npm run build` or `yarn build`.

-->

</body>

</html>

## \_CONFIG.SCSS

/\*

vars

\*/

$black: #010000;

$white: #ffffff;

$lightGrey: #e6e6e6;

$darkGrey: #333;

$red: #ff1755;

/\*

Mixins

\*/

@mixin translateY($e){

-webkit-transform: translateY($e);

-moz-transform: translateY($e);

transform: translateY($e);

}

@mixin transition($e){

-webkit-transition: $e;

-moz-transition: $e;

transition: $e;

}

/\* Fonts -

regular 400

regular 400 italic

bold 700

bold 700 italic

\*/

body {

font-family: 'Open Sans', sans-serif;

font-size: 18px;

color: $black;

font-weight: 400;

font-style: normal;

}

a{

text-decoration: none;

@include transition(.4s ease-in-out);

cursor: pointer;

&:hover{

color: $red;

@include transition(.4s ease-in-out);

}

}

.container-fluid{

line-height: 24px;

}

## DEFAULT.SCSS

@import "\_config.scss";

header{

position: relative;

display: block;

width: 100%;

background: $black;

color: $white;

height: 90px;

-webkit-box-shadow: 0 8px 6px -6px #999;

-moz-box-shadow: 0 8px 6px -6px #999;

box-shadow: 0 8px 6px -6px #999;

.logo {

position: absolute;

top: 50%; left: 15px;

@include translateY(-50%);

}

nav,

nav ul,

nav ul li{

margin: 0;

padding: 0;

}

nav{

position: absolute;

top: 50%; right: 15px;

@include translateY(-50%);

ul li{

list-style-type: none;

display: inline-block;

margin-right: 20px;

&.last{

margin: 0;

}

a{

font-size: 19px;

line-height: normal;

color: white;

}

}

}

}

.tftable {

font-size:12px; color:#333333; width:100%;border-width: 1px; border-color: #729ea5;border-collapse: collapse;

}

.tftable th {

font-size:12px; background-color:#acc8cc;border-width: 1px; padding: 8px; border-style: solid;border-color: #729ea5; text-align:left;

}

.tftable tr {

background-color:#d4e3e5;

}

.tftable td {

font-size:12px; border-width: 1px; padding: 8px;border-style: solid; border-color: #729ea5;

}

.tftable tr:hover {

background-color:#3bc089;

}

## FOOTER.JS

import React, { Component } from 'react';

class Footer extends Component {

render() {

return (

<footer>

Footer

</footer>

);

}

}

export default Footer;

## HEADER.JS

import React, { Component } from 'react';

import {

Link

} from 'react-router-dom';

class Header extends Component {

render() {

return (

<header>

<div className="logo">

LOGO

</div>

<nav>

<ul>

<li className="first">

<Link to="/">Home</Link>

</li>

<li>

<Link to="/Products">Products</Link>

</li>

<li className="last">

<Link to="/Contact">Contact</Link>

</li>

</ul>

</nav>

</header>

);

}

}

export default Header;

## CONTACT.JS

import React, { Component } from 'react';

class Contact extends Component {

render() {

return (

<div className="container-fluid">

<h1>

Contact

</h1>

<table class="tftable" border="1">

<tr><th>Header 1</th><th>Header 2</th><th>Header 3</th><th>Header 4</th><th>Header 5</th></tr>

<tr><td>Row:1 Cell:1</td><td>Row:1 Cell:2</td><td>Row:1 Cell:3</td><td>Row:1 Cell:4</td><td>Row:1 Cell:5</td></tr>

<tr><td>Row:2 Cell:1</td><td>Row:2 Cell:2</td><td>Row:2 Cell:3</td><td>Row:2 Cell:4</td><td>Row:2 Cell:5</td></tr>

<tr><td>Row:3 Cell:1</td><td>Row:3 Cell:2</td><td>Row:3 Cell:3</td><td>Row:3 Cell:4</td><td>Row:3 Cell:5</td></tr>

<tr><td>Row:4 Cell:1</td><td>Row:4 Cell:2</td><td>Row:4 Cell:3</td><td>Row:4 Cell:4</td><td>Row:4 Cell:5</td></tr>

<tr><td>Row:5 Cell:1</td><td>Row:5 Cell:2</td><td>Row:5 Cell:3</td><td>Row:5 Cell:4</td><td>Row:5 Cell:5</td></tr>

<tr><td>Row:6 Cell:1</td><td>Row:6 Cell:2</td><td>Row:6 Cell:3</td><td>Row:6 Cell:4</td><td>Row:6 Cell:5</td></tr>

</table>

</div>

);

}

}

export default Contact;

## HOMEPAGE.JS

import React, { Component } from 'react';

class Homepage extends Component {

render() {

return (

<div className="container-fluid">

<h1>

Home Page Content

</h1>

<p>Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.</p>

</div>

);

}

}

export default Homepage;

## PRODUCTS.JS

import React, { Component } from 'react';

class Products extends Component {

render() {

return (

<div className="container-fluid">

<h1>

Products

</h1>

<p>Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.</p>

</div>

);

}

}

export default Products;