**React Commands**

**@ create your own configuration**

We will need NodeJS

NodeJS and NPM

NodeJS is the platform needed for the ReactJS development.

<https://nodejs.org/es/download/>

Windows Installer (.msi)

32-bit

64-bit

================================================

tools to integrate in your project

>Adding webpack loaders for less, sass

>Doing server side rendering

>Using new ES versions

>Adding MobX and Redux

>Making your own configuration just for learning sake

start by initializing a React project from scratch.

=====================================================================

**Step 1 - Create the Root Folder**

Open up the command line or Git bash and create a new directory:

mkdir react-config-tutorial && cd react-config-tutorial

=====================================================================

Initialize NPM project by running:

npm init -y

C:\Users\rafae\Documents\React\react-config-tutorial>npm init -y

Wrote to C:\Users\rafae\Documents\React\react-config-tutorial\package.json:

{

"name": "react-config-tutorial",

"version": "1.0.0",

"description": "",

"main": "index.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

"keywords": [],

"author": "",

"license": "ISC"

}

=====================================================================

**Step 2 - install React and react dom**

Since our main task is to install ReactJS, install it, and its dom packages, using install react and react-dom commands of npm respectively. You can add the packages we install, to package.json file using the --save option.

C:\Users\Tutorialspoint\Desktop\reactApp>npm install react --save

C:\Users\Tutorialspoint\Desktop\reactApp>npm install react-dom --save

Or, you can install all of them in single command as −

C:\Users\username\Desktop\reactApp>npm install react react-dom –save

Now install react

npm install react react-dom –save

=====================================================================

**Step 3 - Install webpack**

Since we are using webpack to generate bundler install webpack, webpack-dev-server and webpack-cli.

**Configuring webpack**

Our first stop will be the webpack. It’s a very popular and powerful tool for configuring not only React, but almost all front-end projects. The core function of webpack is that it takes a bunch of JavaScript files we write in our project and turns them into a single, minified file, so that it will be quick to serve. Starting from webpack 4, we aren’t required to write a configuration file at all to use it, but in this tutorial we will write one so that we can understand it better.

First, let’s do some installation:

npm install webpack webpack-dev-server webpack-cli –save

This will install:

**webpack module** — which include all core webpack functionality

**webpack-dev-server** — this development server automatically rerun webpack when our file is changed

**webpack-cli** — enable running webpack from the command line

=====================================================================

**Step 4 - Install babel**

Install babel, and its plugins babel-core, babel-loader, babel-preset-env, babel-preset-react and, html-webpack-plugin:

npm install babel-core babel-loader babel-preset-env babel-preset-react html-webpack-plugin --save-dev

=====================================================================

**Step 5 - Create the Files**

To complete the installation, we need to create certain files namely, index.html, App.js, main.js, webpack.config.js and, .babelrc. You can create these files manually or, using command prompt.

C:\Users\rafae\Documents\React\react-config-tutorial>type nul > index.html

C:\Users\rafae\Documents\React\react-config-tutorial>type nul > app.js

C:\Users\rafae\Documents\React\react-config-tutorial>type nul > main.js

C:\Users\rafae\Documents\React\react-config-tutorial>type nul > webpack.config.js

C:\Users\rafae\Documents\React\react-config-tutorial>type nul > .babelrc

=====================================================================

**Step 6 - Set Compiler, Server and Loaders**

Open webpack-config.js file and add the following code. We are setting webpack entry point to be main.js. Output path is the place where bundled app will be served. We are also setting the development server to 8001 port. You can choose any port you want.

webpack.config.js

where <command> is one of:

access, adduser, audit, bin, bugs, c, cache, ci, cit,

clean-install, clean-install-test, completion, config,

create, ddp, dedupe, deprecate, dist-tag, docs, doctor,

edit, explore, fund, get, help, help-search, hook, i, init,

install, install-ci-test, install-test, it, link, list, ln,

login, logout, ls, org, outdated, owner, pack, ping, prefix,

profile, prune, publish, rb, rebuild, repo, restart, root,

run, run-script, s, se, search, set, shrinkwrap, star,

stars, start, stop, t, team, test, token, tst, un,

uninstall, unpublish, unstar, up, update, v, version, view,

whoami

npm install babel-core babel-loader babel-preset-env

babel-preset-react html-webpack-plugin --save-dev

const path = require('path');

const HtmlWebpackPlugin = require('html-webpack-plugin');

module.exports = {

entry: './main.js',

output: {

path: path.join(\_\_dirname, '/bundle'),

filename: 'index\_bundle.js'

},

devServer: {

inline: true,

port: 8001

},

module: {

rules: [

{

test: /\.jsx?$/,

exclude: /node\_modules/,

loader: 'babel-loader',

query: {

presets: ['es2015', 'react']

}

}

]

},

plugins:[

new HtmlWebpackPlugin({

template: './index.html'

})

]

}