ESTUDOS REACTJS

Primeiro passo para começar um projeto Javascript é criar o packge.json, ele pode ser criado com um dos dois comandos abaixo:

* yarn init -y
* Npm init -y

O packege.json é o arquivo onde fica as pricipais informações do projeto e dependências de terceiros, principais bibliotecas são:

* React
  + yarn add react
* React Dom
  + yarn add react-dom
* Babel (converte JS para um JS que os Browsers entendão)
  + yarn add @babel/core @babel/cli @babel/preset-env @babel/preset-react babel-loader @babel/preset-typescript -D
* WebPack
  + yarn add webpack webpack-cli webpack-dev-server -D
* Plugin pra injetar tag scrip no html
  + yarn add html-webpack-plugin -D
* Lib para setar variável de ambiente
  + yarn add cross-env -D
* Libs para importar css para pagina index.html
  + yarn add style-loader css-loader -D
* Loader do webpack para entender arquivo Sass
  + yarn add node-sass sass-loader -D
* Lib para manter o estado da aplicação após o reloader
  + yarn add -D @pmmmwh/react-refresh-webpack-plugin react-refresh
* Adicionar TypeScript ao projeto
  + yarn add typescript @types/react-dom -D

Para o typescript ser inicializado no projeto, deve-se rodar **yarn tsc --init**, este comando cria um arquivo chamado tsconfig.json com algumas configurações.

Para o Babel é necessário criar um aquivo de configuração chamado babel.config.js; No contéudo deste arquivo estará os exports dos módulos do babel.

module.exports = {

presets: [

'@babel/preset-env',

'@babel/preset-typescript',

['@babel/preset-react', {

runtime: 'automatic'

}]

]

}

Para converter o código com JS modermo para um JS entendível pelos browsers deve-se rodar no terminal o comando:

* yarn babel (caminho do arquivo) -o (flag -o indica a saída) (saída do arquivo)
  + yarn babel src/index.js -o dist/bundle.js

Para o WebPack é necessário criar o arquivo de configuração webpack.config.js; No conteúdo do arquivo deve-se configurar os módulos que serão exportados:

const path = require('path');

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

const isDevelopment = process.env.NODE\_ENV != 'production';

module.exports = {

mode: isDevelopment ? 'development' : 'production',

devtool: isDevelopment ? 'eval-source-map' : 'source-map,

entry: path.resolve(\_\_dirname, 'src', 'index.jsx'),

output: {

Path: path.resolve(\_\_dirname, 'dist'),

filename: 'bundle.js'

},

resolve: {

extensions: ['.js','.jsx'],

},

devServer: {

contentBase: path.resolve(\_\_dirname, 'public'),

},

plugins: [

new HtmlWebpackPlugin({

template: path.resolve(\_\_dirname, 'public', 'index.html'),

}),

],

module: {

rules:

[

{

test: /\.jsx$/,

exclude: /node\_modules/,

use: {

loader: 'babel-loader',

options: {

plugins: [

isDevelopment && require.resolve('react-refresh/babel')

].filter(Boolean)

}

},

},

{

test: /\.scss$/,

exclude: /node\_modules/,

use: ['style-loader','css-loader',’sass-loader’],

}

]

}

};

Para executar o webpack na aplicação basta rodar no terminal: yarn webpack.

Para executar o webpack serve deve-se rodar no termina: yarn webpack serve

Para criar uma variável de ambiente deve-se rodar no termial o comando:

* Nome\_da\_variavel=valor\_da\_variavel yarn webpack
  + NODE\_ENV=’development’ yarn webpack

Para não precisar rodar o comando webpack serve todas as vezes que forem necessárias, pode-se configurar um script em package.json, e atribuir um álias para o comando:

{

"name": "EstudoReact",

"version": "1.0.0",

"main": "index.js",

"license": "MIT",

"dependencies": {

"react": "^17.0.2",

"react-dom": "^17.0.2"

},

"scripts" : {

"dev" : "webpack serve",

"build" : "cross-env NODE\_ENV=production webpack"

},

"devDependencies": {

"@babel/cli": "^7.16.0",

"@babel/core": "^7.16.0",

"@babel/preset-env": "^7.16.4",

"@babel/preset-react": "^7.16.5",

"babel-loader": "^8.2.3",

"html-webpack-plugin": "^5.5.0",

"webpack": "^5.65.0",

"webpack-cli": "^4.9.1",

"webpack-dev-server": "^4.6.0"

}

}

Componetes React:

São formas de encapsular em um bloco de código dentro de um único elemento e esse elemento tem suas próprias funcionalidades, estilização e estrutura. É uma forma de organizar a aplicação em vários pedaços que compoem toda a aplicação. E dentro do React, tudo é um componente.

Em termos técnicos um componente é uma função que sempre devolve um HTML.