Passo a passo de como começar um projeto em react !

**Caso queira pular alguns passos, esse é o script do package.json**

{

  "name": "ignite-reactjs",

  "version": "1.0.0",

  "main": "index.js",

  "license": "MIT",

  "dependencies": {

    "css-loader": "^6.2.0",

    "react": "^17.0.2",

    "react-dom": "^17.0.2",

    "style-loader": "^3.2.1"

  },

  "scripts": {

    "test": "jest",

    "dev": "webpack serve",

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

  },

  "devDependencies": {

    "@babel/cli": "^7.14.8",

    "@babel/core": "^7.15.0",

    "@babel/preset-env": "^7.15.0",

    "@babel/preset-react": "^7.14.5",

    "@babel/preset-typescript": "^7.15.0",

    "@pmmmwh/react-refresh-webpack-plugin": "0.5.0-rc.4",

    "@types/react": "^17.0.20",

    "@types/react-dom": "^17.0.9",

    "babel-loader": "^8.2.2",

    "cross-env": "^7.0.3",

    "html-webpack-plugin": "^5.3.2",

    "node-sass": "^6.0.1",

    "react-refresh": "^0.10.0",

    "sass-loader": "^12.1.0",

    "typescript": "^4.4.2",

    "webpack": "^5.51.1",

    "webpack-cli": "^4.8.0",

    "webpack-dev-server": "^4.1.0"

  }

}

**PASSO 1**

- Criar pasta onde será salvo o projeto

- Abrir a pasta no VS Code

- Inicializar o package.json yarn init -y ou npm init -y.

**PASSO 2**

- yarn add react

- yarn add react-dom

**PASSO 3**

- Criar pasta src e public

- Criar um index.html na pasta public

- Dentro do index.html inicializar o código html:5

- Inserir no <body/>

 <div *id*="root"></div>

- Criar arquivo chamado App.jsx na pasta src

- Inserir dentro do App.jsx

export function App() {

  return (

    <>

    </>

  )

}

- Criar arquivo index.jsx dentro da src

- Inserir dentro do index.jsx

import React from 'react';

import { render } from 'react-dom';

import { App } from './App'

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

**PASSO 4 - BABEL**

- yarn add @babel/code @babel/cli @babel/preset-env -D

- Criar um arquivo chamado babel.config.js dentro da pasta src

- Inserir no arquivo babel.config.js

- yarn add babel/preset-react -D

- yarn add babel-loader -D

*module*.*exports* = {

  presets: [

    '@babel/preset-env',

    '@babel/preset-typescript',

    ['@babel/preset-react', {

      runtime: 'automatic'

    }]

  ]

}

- yarn babel src/index.jsx –out-file dist/bundle.js

**PASSO 5 – WEBPACK**

- yarn add webpack webpack-cli webpack-dev-server -D

- Criar um arquivo chamado webpack.config.js na pasta src

- Inserir no arquivo webpack.config.js

const path = require('path')

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

const ReactRefreshWebpackPlugin = require('@pmmmwh/react-refresh-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.tsx'),

  output: {

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

    filename: 'bundle.js'

  },

  resolve: {

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

  },

  devServer:{

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

    hot: true,

  },

  plugins: [

    isDevelopment && **new** ReactRefreshWebpackPlugin(),

**new** HtmlWebpackPlugin({

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

    })

  ].filter(*Boolean*),

  module: {

    rules: [

      {

        test: /\.(j|t)sx$/,

        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']

      }

    ],

  },

};

**PASSO 6**

- yarn add html-webpack-plugin -D

- yarn add cross-env -D

**PASSO 7**

-  "scripts": {

    "test": "jest",

    "dev": "webpack serve",

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

  },

- yarn dev para denvolvimento e yarn build para produção, teste com yarn dev, sua aplicação deve abrir.

**PASSO 8**

- yarn add style-loader css-loader -D

- yarn add node-sass -D

- yarn add sass-loader -D

**PASSO 9**

- yarn add -D @pmmmwh/react-refresh-webpack-plugin react-refresh

**PASSO 10**

- yarn add typescript -D

- yarn tsc –init

- yarn add @types/react-dom -D

- ao criar o arquivo tsconfig.json você deve colocar esse script no lugar, ele já está otimizado com as principais funções.

{

  "compilerOptions": {

    "lib": ["dom","dom.iterable","esnext"],

    "jsx": "react-jsx",

    "moduleResolution": "node",

    "resolveJsonModule": true,

    "isolatedModules": true,

    "allowJs": true,

    "noEmit": true,

    "allowSyntheticDefaultImports": true,

    "esModuleInterop": true,

    "forceConsistentCasingInFileNames": true,

    "strict": true,

    "skipDefaultLibCheck": true,

    "skipLibCheck": true

  },

  "include" : ["src"]

}