**Ignite Curso de React.js**

1. Criar pasta do projeto

2. npm init -y (cria o package.json, onde vai contar todas as informações principais do projeto)

3. npm add react (para adicionar o react)

4. npm add react-dom (React-dom é o react para web)

DOM é a árvore de elementos da aplicação, é o HTML convertido na sintaxe de objeto)

5. Criar estrutura de pastas:

5.1. Pasta src (Source)- onde ficará nosso código javascript. nj

5.2. Pasta public (Asset) - onde ficará arquivos públicos, como o index.html, ícones.

5.2.1. Criar index.html

6. Configurar biblioteca Babel (Babel converte o código de uma maneira que todos os browsers e ambientes conseguem ler):

6.1. npm add @babel/core @babel/cli @babel/presente-env -D (-D significa que só vai ser instalado no ambiente de desenvolvimento e não irá para produção)

6.1.1. babel core é a biblioteca babel;

6.1.2. babel cli para executar o babel via linha de comando;

6.1.3. babel preset-env é um extensão do babel que identifica o ambiente que o app está sendo executado e converte o código da melhor maneira.

6.2. Criar um arquivo chamado babel.config.js (onde vai ser exportado a configuração (os módulos) do babel)

7. Criar arquivo index.js dentro da pasta scr.

7.1. Adicionar código de teste:  
 const user = {

Name: “Ashley”

}

Console.log(user.address?.street)

\*”?” verifica se é null

7.2. npx babel src/index.js --out-file dist/bundle.js (após isso o código será convertido.)

8. Dentro de index.js importar o React de “react” e criar um function App() retornando um código html (ex: <h1>Hello World</h1>

9. npm add @babel/preset-react -D para que ele consiga entender o código react.

\* transformar index.js em index.jsx, pois o x indica o uso do html no javascript.

10. Adicionar webpack no projeto.

10.1. npm add webpack webpack-cli -D

10.2. adicionar um arquivo webpack.config.js

10.2.1. nesse arquivo será necessário exportar os objetos de entrada e de saída.

10.2.2. exportar as extensões para que o resolve consiga ler (.jsx)

10.2.3 excluir todos os arquivos de node-modules.

10.2.4. adicionar a integração do webpack com o babel usando o babel-loader.

11. Criar um novo arquivo chamado app.jsx e passar a função App() criado anteriormente para ele, porém exportando a função.

12. no index.jsx será necessário importar a função App do arquivo app.jsx

13. npx wepack para gerar um novo bundle.js

14. no index.html na pasta public vamos adicionar um div no body com o id=”root”

15. adicionar html no root usando a função render

16. adicionar o bundle.js no index.html usando uma tag script.

17. adicionar o plugin html-webpack-plugin, pois ele vai referenciar o bundle.js no index, sem precisar adicionar manualmente.

18. Adicionar o Webpack server, para que gerar um servidor local, com auto save, sem a necessidade de recarregar o browser

19.Criar atalhos de script para não precisar rodar o webpack server e sim npm run dev ou npm run build.

20. Importar css no app.js e adicionar uma regra para leitura do css e adiciono as bibliotes style-loader e css-loader

21. Adicionar SASS no projeto com node-sass e sass-loader, para programar com arquivos pré processado.

REACT – COMPONENTES

1. Componentes no React são como as tags no HTML
2. Componente é uma função que começa com a primeira letra maiúscula.
3. Quando um arquivo é um componente sempre o começar com a letra maiúscula.
4. Apenas um componente por arquivo.
5. Para incluir variáveis dentro das tags html, é possível usar interpolação com duas chaves ”{}”.

REACT – PROPRIEDADES

1. Propriedades são semelhantes a atributos no HTML.