React

Framework de SPA 🡪 Ferramenta de trabalho  
React 🡪 Framework de Javascript

* Webpack e Babel
* Node – roda o JS na nossa máquina
* JS por padrão roda apenas no navegador, engine 🡪 responsável por entender e rodar o JS
* Chrome 🡪 V8
* NPM (Node Package manager) – Instala e manipula pacotes na máquina, pacotes (códigos externos que já entregam uma funcionalidade)
* NPX – Create React App
* Package-lock.JSON – relata a versão de cada pacote instalado em nosso projeto
* Package. JSON – tem diversas funcionalidades, as principais são também relatar os pacotes que temos instalados na nossa aplicação, também responsável pelo armazenamento do react-script

O react é todo baseado em componentização.

Webpack agrupa todo código da aplicação, isso é chamado de bundle. Ele auxilia na separação do nosso código em diversos arquivos.  
O webpack também trabalha em conjunto com o NPM na importação de códigos do nosso pacote externo.

Babel transforma o código JS atual em código mais antigo, transforma o JSX em funções react.

JSX difere do HTML, todos os atributos vão estar em camelCase, sempre será usado dentro do return e dentro do return só será passado um único elemento diretamente, porém, utilizando o React.Fragment, será possível passar mais de um elemento,

***<React.Fragment></ React.Fragment>*** ou apenas ***<></>***

Dentro do JSX ainda é possível escrever código Javascript, abrindo chaves dentro de uma div que está incluída no return ***<div>{lista}</>***

Quando estivermos escrevendo uma função em um componente a primeira letra do nome da função será maiúscula.

React Hooks – facilitam a manipulação e a interação com esses estados

Estado e reatividade

UseState – trabalha entre os estados da aplicação, identifica quando um valor é alterado dentro de um componente e recarrega esse componente em tela quando esse valor é alterado

Ciclo de vida de um componente – 1 – Quando ele é construído em tela, 2 – Quando ele é atualizado, 3 – quando ele é removido da tela

UseEffect

Responsável por rodar o código pela primeira vez na tela, atualização do componente (controle feito pelo ‘array’ de dependências (um estado de dependência gerado pela atualização do componente onde o call-back será executado), quando o array está vazio o useEffect será executado apenas na primeira renderização), quando o componente é removido em tela e depois pode ser feito código para ser executado com o ‘return’ para rodar.’

UseRef

Menos utilizado, faz referência e tem acesso a um elemento criado em tela no nosso DOM  
Pode ser utilizado dentro do useEffect

‘Const vídeo = useRef ()’

Ex. useEffect(() => {

      console.log(video.current.play());

    }, [])

Como também através de uma função de call-back

function callBack() {

    console.log(video.current.play());

  }

useMemo & useCallBack

useMemo – Método que vai receber uma função que não vai ser recarregada com a atualização do componente, o seu valor de retorno vai ficar na variável que a criou, ele também pode receber um array de dependência assim como o useEffect, para elementos extremamente pesados

useCallBack – armazena uma função inteira, como se fosse um call-back para ser executada a partir de um evento, só será carregada se houver dependências, também é usado para funções extremamente pesados

Contexto Global / Estado Global – serve basicamente para centralizar valor e estados e compartilhar valores e estados entre componentes que não interagem diretamente, ou seja, um não é pai do outro e um não é filho do outro

Stores ou Contexts – são pastas comumente utilizadas para armazenar valores (Todo componente deve ser iniciado com letra maiúscula)

Rotas

import {BrowserRouter, Router, Route} from "react-router-dom";

O BrowserRouter deve envolver toda parte da aplicação que vai utilizar de rotas, deve ser passado no início, na raiz da aplicação, em App

Id = valor de identificação único pra uma determinada coisa

Banco de dados

Visible -> deve ser deletado