**1 – O QUE É REACT?**

* Uma **biblioteca JS** para criação de interfaces;
* Utilizado para construir **SPAs** (Single Page Applications);
* Baseado em **componentes**;
* Utiliza o **JSX** para renderizar HTML;
* E aplica o **Virtual DOM** para realizar as alterações de DOM;
* Podemos **adicionar a um projeto** ou criar um projeto com ele;

**2 – COMO INSTALAR O REACT?**

* Para instalar o React vamos utilizar uma ferramenta chamada **Create React App**;
* Recebemos todos os arquivos da biblioteca e **temos como executá-la**;
* Para utilizar precisamos do **Node e também npm**;
* Esta ferramenta também otimiza o app gerado para **produção**;
* É possível iniciar a aplicação com **npm start**;

**3 – ENTENDENDO O JSX**

* O **JSX** é como um HTML, porém dentro do código Javascript;
* É a principal maneira de escrever HTML como o React;
* Podemos **interpolar variáveis**, inserindo ela entre {};
* É possível também **executar funções** em JSX;
* E aplica o **Virtual DOM** para realizar as alterações de DOM;
* Inserir **valores em atributos de tags** também é válido em JSX;

**4 – COMPONENTES**

* Permite **dividir a aplicação** em partes;
* Os componentes **renderizam JSX**,assim como App.js (que é um componente);
* Precisamos **criar um arquivo** de componente;
* E importa-lo onde precisamos utilizar;
* Normalmente ficam emuma pastachamada **components**;

**5 – PROPS**

* As **props** são valores passados para componentes;
* Podemos deixá-los **dinâmicos**;
* Ou seja, **mudando a execução** por causa do valor da prop;
* O valor é **passado como um atributo** na chamada do componente;
* E precisa ser **resgatado dentro de uma propriedade / argumento** chamada de props na função de definição do componente;
* As props são somente de leitura;

**6 – ADICIONANDO CSS**

* O **CSS** pode ser **adicionado de forma global** na aplicação, por meio do arquivo index.css por exemplo;
* Porém é possível **estilizar a nível de componentes**;
* Utilizamos o **CSS modules** para isso;
* Basta criar um arquivo como: **Componente.module.css**;
* E chamar esse CSS no componente;

**7 – FRAGMENTOS**

* Os **React Fragments** permitem a criação de um componente sem elemento pai
* O propósito é **descomplicar os nós do DOM**
* A sintaxe é **<>** e **</>**, não há um nome para a tag
* Criamos no próprio JSX.

**8 – AVANÇANDO EM PROPS**

* Podendo **definir tipos para as props**, realizando uma espécie de validação
* Definimos em um objeto chamado propTypes no próprio componente
* E ainda há a possibilidade de **definir um um valor padrão**
* Neste caso utilizamos o objeto **defaultProps**

**9 – EVENTOS**

* Os **eventos de React** são os mesmos do DOM
* Ou seja, temos eventos para responder a um clique
* **O evento é atrelado a uma tag** que irá executá-lo
* Geralmente um **método** é atribuído ao evento
* Este método deve ser criado no componente

**10 – USESTATE**

* O **useState** é um hook do React
* Com ele conseguimos **manusear o estado** de um componente de forma simples
* Este hook **funciona muito bem com eventos**
* Podemos **atrelar um evento** a mudança de state

**11 – MÉTODOS POR PROPS**

* Os **métodos** também podem ser passados por props
* Ou seja, um **componente filho pode ativar o método do seu ancestral**
* Vamos acessar o método por meio de um **evento**
* A sintaxe é a mesma de uma props de dados: **props.meuEvento**

**12 – RENDERIZAÇÃO POR CONDIÇÃO**

* Podemos atrelar a exibição de algum a um **if**
* Esta ação é chamada de **renderização condicional**
* Envolvemos as tags em chaves **{}**
* É possível usar o **state** para criar as condições

**13 – RENDERIZAÇÃO DE LISTAS**

* Para renderizar uma lista vamos primeiramente precisar de um **array**
* Depois utilizamos a **função map**, para percorrer cada um dos itens
* Podendo assim renderizar algo na tela
* É possível **unir operadores condicionais** com a renderização de listas

**14 – STATE LIFT**

* **State Lift** é uma técnica utilizada para compartilhar o state
* É normal vários componentes **dependerem do mesmo estado**
* Então precisamos elevar o nível do mesmo a um **componente pai**
* Então centralizamos o state no pai, e **definimos quem usa e quem define** (setState)