Nesse curso damos uma introdução sobre o funcionamento do React. Uma coisa que não ficou tão clara no curso mas que vale a pena anotar, o **react não é uma aplicação que voce baixa e permite que o computador utilize-o como é para linguagens, é uma biblioteca que você baixa quando especificamente na pasta que vai utilizar, no aplicativo que esta sendo feito, isso significa que para cada programa que o react vai ser usado, é necessário baixar o react para dentro daquele programa, permitindo que o javascript faça uso dele**.

O **ReactJS** é uma **biblioteca** JavaScript para criar uma interface de usuário, podendo ser aclopada em qualquer projeto, **ela não é um framework**, e sua função principal é a criação de interface para o usuário. Começou seu uso no Facebook, antes de 2012, ficando em open-source e liberado pra uso em 2013.

Ele é **modular**, com pequenas peças, **components**, que são utilizadas pelo usuário de acordo com sua necessidade. O ReactJS é muitas vezes tratado como framework, mostrando o quanto ele é capaz, mas vale relembrar, NÃO É.

React é uma linguagem **declarativa**, ela esta preocupada apenas com o que é exebido na interface do usuário, **ele apenas declara o que acontece**, podendo ser utilizado em qualquer lugar, mas geralmente é usado para o front-end.

O React não separa técnologias com marcações em arquivos separados, mas **separa conceitos e mantem o código pouco acoplado, chamando de components**. O **JSX** é uma sintax sugar para o React.createElement, tornando simples o processo, mas não é obrigatório.

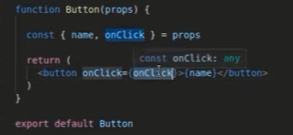
O Browser é incapaz de interpretar o JSX, precisando de um transpilador para que a aplicação entenda o código, geralmente utiliza-se o **Babel**.

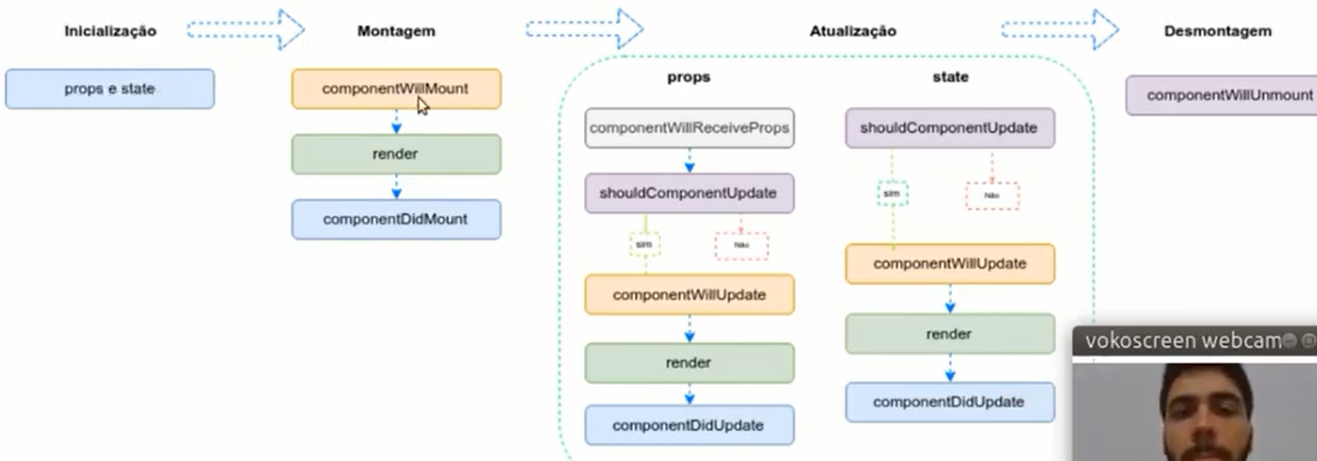
O React é uma **single page application**, ele faz com que fique tudo em uma única página html

O react possue um **DOM virtual** por fora do browser, o que traz vantagens de performace e evita erros, simplesmente deixando de renderizar aquilo que não funciona sem compremeter o funcionamento do restante.

observando que para o uso do react é necessário importar os seguintes imposts do próprio react. O instrutor guiou um pouco como instala mas ele fez da maneira mais confusa possível, caso não funcione na primeira vez, procurar na net. Os arquivos JSX **são os arquivos que o React usa para renderizar** seus nomes sempre começam com letra maiúscula, como identificador prático, coisas como botões, para que mantenhamos um padrão de botão e assim melhorando a performace, podem ser criados utilizando arquivos jsx dentro de /assets e importando-os para o script principal, lembrando que ele também precisará dos imposts do react e reactDOM e **usara a linguagem do JavaScript**.

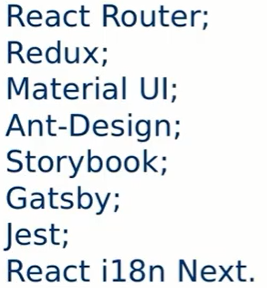
Um componente de jsx deve ser preferencialmente feito de forma abstrata e sem detalhe, para que suas caracteristicas especificas sejam dadas pelo script no qual ele sera utilizado.

**Tudo que estiver dentro de um componente é acessado através do children**, em outras palavras, sempre que usamos um prop que pertence a outra coisa, devemos avisar que é uma children. Além disso, em geral ao tratar de funções dentro do react, utilizamos muito **arrow functions para fazer nossas funções**.

**Estado** e **ciclo de vida** são termos importantes para o React, principalmente com o funcionamento de **roots**.

A imagem acima mostra a ordem de acontecimento das instruções dentro do React, note que o **componentWillMount** esta depreciado, e pode não estar mais na versão atual.

Inicialização Montagem Atualização desmontagem

O ReactDOM sera atualizado a cada renderização. Nenhum componente pai ou filho devem saber **se outro componente possui estado ou não**. O **estado** é apenas **local** **ao componente**, e caso seja necessário **enviar algum atributo para outro** **é utilizado os props**.

Lembrete, o método **render** é a principal forma de uso dos componentes do React, é como ele reconhece o que deve renderizar na página.

O react não é ideal de ser utilizado sozinho, então se utilizao outras bibliotecas que fazem parte do seu **ecossistema** que facilitam seu uso ainda mais, exemplos estão na imagem ao lado.

**Webpack**, o module bundler é um **empacotador de módulos para aplicações JS**, ele gera bundlers que serão utilizados no HTML, em ES5.

<https://webpack.js.org/>

ele suporta **fontes**, **CSS**, **imagens**, **HTML**, **JS** e **plugins**.

Na configuração do webpack existe alguns conceitos principais:

**Entry –** utilizando grafos, o webpack precisa **de um ponto de entrada para buscar os módulos e suas dependências**.

**Output –** para determinar quais os bundlers serão exportados para o arquivo final que o webpack vai emitir.

**Loader –** para permitir que o Webpack gerencia arquivos não-JS.

**Plugins** – podem ser utilizados para otimizar pacotes, verificação e gestão de scripts, etc.

**Mode –** utilizado para abordagens de configuração 0, podendo ser três: **production**, para otimização internas, **development**, que executa 3 plugins especificos que são plugins básicos, o **ugliflyJsPlugin**, **ModuleConcatenationPlugin** e **NoEmitOnErrorsPlugin** (se não passar nada, esse será selecionado),e **none** para nenhuma otimização.

Para a criação do webpack, seguimos os seguintes comandos:

Npm i –D webpack webpack-cli

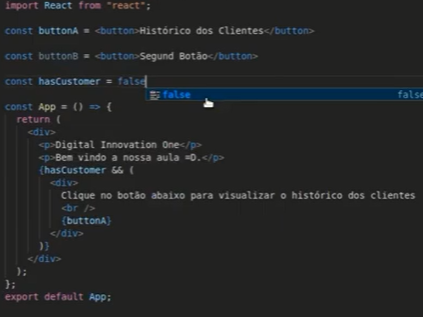
“build”: “webpack –mode *production*”

Npm i @babel/core babel-loader @babel/preset-env @babel/preset-react –save-dev

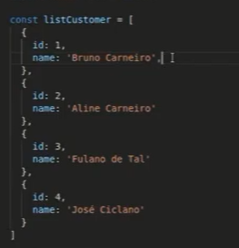
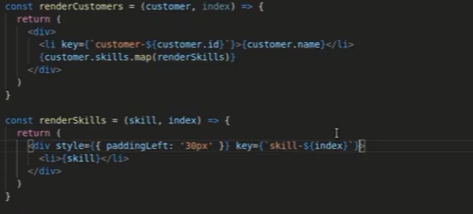


A imagem acima mostra a instalação do Eslint.

A **renderização condicional** é uma capacidade do react que permite renderizar apenas alguns componentes distintos que encapsulam o comportamento que você precisa, de acordo do **estado** de sua aplicação. Pensa em um if que checa o estado para da aplicação para renderizar algo específico.

São variáveis de elemento, com **if inline** e o operador lógico **&&**, **if else** com operador lógico **condicional**. Ele evita que um componente não seja/seja renderizado, conforme o **estado.**

Também é possivel fazer listas arrays, renderizando multiplos componentes, listas básicas, chaves e extraindo componentes de chaves, as chaves são unicas e utilizadas dentro dos elementos apenas. Um exemplo de lista pode ser visto abaixo. Na próxima página pelo jeito lol

cada item da lista (de elementos irmãos) **só pode ter uma única chave** (Key).

O react tambem permite manipular eventos através de suas diversas funções.