Anotações sobre o React

● Introdução

Esse texto consiste basicamente das minhas anotações e pensamentos enquanto estudo React. Estarei estudando através desse material caso você tenha interesse:

- React Course - Beginner's Tutorial for React JavaScript Library [2022]: https://www.youtube.com/watch?v=bMknfKXIFA8
- Understanding React's UI Rendering Process: https://www.youtube.com/watch?v=i793Qm6kv3U
- React.js Deep Dive: https://www.youtube.com/playlist?list=PLxRVWC-K96b0ktvhd16l3xA6gncuGP7gJ

● O que é React?

React é uma biblioteca JavaScript para construção de interfaces de usuário altamente responsivas e interativas, talvez você esteja se perguntando “mas não dá pra fazer isso só com HTML, CSS e JavaScript?” e é aqui que está o pulo do gato, o que faz o React tão relevante é a forma como ele faz isso, o React funciona baseado em componentes, isso é o React permite que o dev divida a UI em componentes, que são blocos de código independentes que armazenam e controlam a aparência do componente e a sua lógica, agora traduzindo, o React essencialmente junta o HTML, CSS(não diretamente) e JS em um canto só, permitindo que você consiga trabalhar e definir o comportamento de cada componente diretamente no JavaScript, ou seja, vc passa a poder utilizar a lógica do JS para definir o estado do seu HTML, para facilitar você pode pensar os componentes basicamente como um objeto no JS que está diretamente ligada ao HTML.

Outra coisa importante do react é que ele armazena o estado do componente no JS e só atualiza o HTML quando o programador decide que deve atualizar.

Agora vamos aos estudos.

● O básico

React é declarativo, não imperativo
React é como um quebra cabeça
React só renderiza um elemento pai por vez

Sabe quando eu disse pra vc pensar nos componentes basicamente como objetos é porque de fato eles são. Primeiro o React funciona através do JSX(é tipo HTML, mas não é HTML), observe esses pedaços de código.

```
<div className="Foto">
    <h1>Diga xissss</h1>
    <div>
        <p>            o#######o<br></br>
                 o##########o<br></br>
               o##########o<br></br>
              ################<br></br>
             ######    \ ########o<br></br>
            '#;^    _^, / --- \ #####!<br></br>
            ,`     /^_ .-~^~-.__\#<br></br>
           /        ^\/,,@@@,, ; |<br></br>
          |            \!!@@@@@!! ^<br></br>
        #.        .\; '9@@@P'   ^<br></br>
        ###./^ ----,_^^      /@-._<br></br>
                      ^--._,o@@@@@@<br></br>
                         ^;@@@@@@@@@<br></br>
                             ^-;@@@@</p>
    </div>
</div>
```

```
<div>
    <h1>Ian é:</h1>
    <ul>
        <li>incrivel
        </li>
        <li>
            genial
        </li>
        <li>belo</li>
        <li>estranho</li>
    </ul>
</div>
```

Parece HTML, porém é JSX o Javascript interpreta ele basicamente como um objeto, podemos ver isso ao tentarmos renderizar na tela com apenas o JS

```
function Ian() {
        return(
        <div>
            <h1>Ian é:</h1>
            <ul>
                <li>incrivel
                </li>
                <li>
                    genial
                </li>
                <li>belo</li>
                <li>estranho</li>
            </ul>
        </div>
    )
}

var ianJSX = Ian()
document.getElementById("root").append(ianJSX)
document.getElementById("root").append(JSON.stringify(ianJSX))
```

Isso nos dá para nós um objeto

[object Object]{"type":"div","key":null,"ref":null,"props":{"children":[{"type":"h1","key":null,"ref":null,"props":{"children":"Ian é:"},"_owner":null,"_store":{}},{"type":"ul","key":null,"ref":null,"props":{"children":[{"type":"li","key":null,"ref":null,"props":{"children":"incrivel"},"_owner":null,"_store":{}},{"type":"li","key":null,"ref":null,"props":{"children":"genial"},"_owner":null,"_store":{}},{"type":"li","key":null,"ref":null,"props":{"children":"belo"},"_owner":null,"_store":{}},{"type":"li","key":null,"ref":null,"props":{"children":"estranho"},"_owner":null,"_store":{}}]},"_owner":null,"_store":{}}]},"_owner":null,"_store":{}}

porém o react possui a função própria para transformar JSX em HTML e transformá-lo em um DOM, ReactDOM.render(HTML, Local onde deve ser colocado o HTML), funciona tipo um appendChild.

Se voltarmos ao primeiro código e passarmos por essa função

OBS: Pra designar uma classe no react vc precisa criar ela com className e chamar a função com <Nome />. aaaaaa e a função tem q começar com letra maiscula

```
function Foto() {
    return (
        //ASCII FICOU UMA MERDA (ㅜ_ㅜ)
        <div className="Foto">
            <h1>Diga xissss</h1>
            <div>
                <p>            o#######o<br></br>
                         o##########o<br></br>
                       o##############o<br></br>
                     ################<br></br>
                     #####    \ ########o<br></br>
                    '#;^    _^, / --- \ #####!<br></br>
                    ,`     /^_ .-~^~-.__\#<br></br>
                   /        ^\/,,@@@,, ; |<br></br>
                  |            \!!@@@@!! ^<br></br>
                #.       .\; '9@@@P'   ^<br></br>
                ###./^ ----,_^^     /@-._<br></br>
                            ^--._,o@@@@@@<br></br>
                               ^;@@@@@@@@<br></br>
                                 ^-;@@@@</p>
            </div>
        </div>
    )
}

ReactDOM.render(<Foto/>, document.getElementById("root"))
```

O resultado disso será

Diga xissss

```
          o#######o
        o###########o
      o#############o
     #################
     ######    \ ########o
    '#;^    _^, / --- \ #####!
     ;`     /^_ .-~^~-.__\#
    /       ^\/,,@@@,, ; |
   |          \!!@@@@@@!! ^
  #.       .\; '9@@@P'   ^
  ###./^ ----,_^^      /@-._
            ^--._,o@@@@@@@
              ^;@@@@@@@@@@
                ^-;@@@@
```

um HTML perfeitamente construído

● Data driven React

Quando você estiver escrevendo em JSX e quiser indicar para o react interpretar algo como JS vc escreve entre colchetes.

EX:

nome = ian

return(<h1>Hello nome</h1>)

o output será:

Hello nome

porém se vc fizer
nome = ian
return(<h1>Hello {nome}</h1>)

o output será:
Hello ian

inclusive vale tudo de JS entre colchetes, então se o nome for dado pela função getNome() vc poderia escrever

return(<h1>Hello {getNome()}</h1>)

e o output seria:
Hello ian

Agora digamos que eu queira passar parâmetros pra dentro do meu componente, o jeito que você faz isso é basicamente como no HTML. Suponhamos que temos um componente chamado Pessoa que mostra o nome da pessoa e o número dela, para ele ser reutilizável vc precisa passar como parâmetro (no caso de componentes se chama prop) esses dados, para fazer isso a sintaxe seria

```
return(<div>
<Pessoa
nome=”Ian”
numero=”2345678”
/>
<Pessoa
nome=”Zeca”
numero=”757345”
/>
<Pessoa
nome=”Giulia”
numero=”0800777”
/>
</div>)
```

Aqui nós ainda temos um problema vc ta tendo que colocar todos esses parâmetros na mão, mas lembra dos colchetes, suponhamos que vc tenha um array com os dados vc poderia escrever

```
return(<div>
<Pessoa
nome={nome.pop()}
numero={num.pop()}
/>
```

```
<Pessoa
nome={nome.pop()}
numero={num.pop()}
/>
<Pessoa
nome={nome.pop()}
numero={num.pop()}
/>
</div>)
```

E uma coisa importante esses props chegam na função como um objeto, nesse exemplo se a gente tivesse um console.log(props) a saida seria

```
{nome: “Ian”, numero=”2345678”}
{nome: “Zeca”, numero=”757345”}
{nome: “Giulia”, numero=”0800777”}
```

● State

A diferença entre props e state é que props são como variáveis que existem para além da função e são imutáveis e state é como uma variável declarada na função, que naturalmente é gerida pela própria função.

A sintaxe do state é React.useState(valorPraArmazenar), se a gente quiser salvar um nome por exemplo nos vc poderia imaginar que o certo a se fazer é

nome = React.useState(“Ian”)

porem React.useState() retorna um array contendo o valor default do state e uma função então no nosso exemplo

console.log(nome)

nos daria

[“Ian”, f()]

e oq seria essa função? ela é a forma que o react nos dá de alterar o state. então para armazenar a variável corretamente nos desconstruirmos o array na declaração

[nome, setNome] = React.useState(“Ian”)

e caso queiramos alterar o nome na função nos chamamos a função

setNome(“novoNome”).

Se vc tiver a necessidade de mudar o state recursivamente, digamos que queiramos adicionar um sobrenome a nome, é natural pensar em fazer

setNome(nome + “ Simões”)

porem isso está errado (tem a ver com a forma que o react interpreta e renderiza state), oq vc deve fazer é uma callback function, assim

```
setNome(function(prev){
        return prev + “ Simões”})
```

ou usando função com setinha
setNome(prev => prev + ” Simões”)