

React JS

BIBLIOTECA
FRAMEWORK

- ① Entender a semelhança do react com tags HTML

Exemplo: tag ``

é um componente auto-suficiente

Se eu quero que `` apareça, basta atribuir a propriedade `src` e atribuir um valor a ela, ou seja, um endereço que leva a uma imagem.

Até fazemos a mínima ideia o que é uma tag, esse componente te deve que redar internamente só baixar essa imagem e mostrar no topo. Vai simplesmente dizer que a tag é só isso, ela só dará a sua maquininha interna e fornecerá.

```
<img src = "https://m..." >
```



Componentes no React são a mesma coisa, só que agora temos a oportunidade de abrir o copê desse componente e ver o que há lá dentro, mexer na maquininha interna ou criar seu próprio componente com qualquer comportamento que você quiser.

A boa notícia é que **COMPONENTES** são a **BASE** do React! Só é necessário aprender a fazer isso para habilitar todo o fluxo e maioria dos coisas que querem construir.

- Saber HTML é usar esse modelo muito, só fato agora abrir o copê e mexer na maquininha lá dentro. ☺

② CRIAR NOSSO PRIMEIRO COMPONENTE REACT

Exemplo: componente **CapsLock**

< CapsLock />

Isso, por si só, não faz nada
não imprime nada
mas, qualquer texto passado para ele, vai ser deixado
tudo em caixa alta!

É como agora eu falar com a caixa interna disso?

< CapsLock texto />

Desse modo ele cria
uma propriedade chamada "texto",
assim como a propriedade src daquele img.

Nessa propriedade "texto", ele manda o valor de
"Me dixe um capslock por favor"

< CapsLock texto = "Me dixe um capslock por favor" />

Agora, vamos abrir o capô desse componente...

function capsLock () {

ABRINDO O CAPÔ

F

→ VER A MÁQUINA INTERNAL

Aqui, temos a função f\$. Ou seja, sempre
neste roteiro é a função f\$!

COMPONENTE REACT == FUNÇÃO JAVASCRIPT



Biblioteca vinculável, porque entrega **implementações** ao máximo conhecimento que você SÁ TEM um HTML e JS.
Quanto \oplus você subir o nível de dificuldade, \oplus suas propriedades você vai encontrar, pôr que é **um pão fresco**.

Nos esforçamos ao máximo manter na base o **romântico** do que já conhecemos.

→ Forma essa máquina **metade daí de fio**?
ex: "Me dê um lapis lool por favor!"

PARÂMETROS da função

`function lapisLool()`

RECEBE
VALORES!

Tem vez de, no assinatura da função, ter 1 parâmetro ou mais, propriedade que vem do lado de fora, o React agrupa **TUDO** em 1 único parâmetro, chamado

PROPS

= PROPRIEDADES

! function mapStateToProps (props) {

}

Então, todos os nossos propriedades são
acessíveis por esse único parâmetro, que é um

OBJETO JS

Dessa forma, conseguimos extrair o valor em
uma, acessando de forma tradicional a prop. texto
desse objeto, que é a mesma prop. em vermelho no compo-
nente texto.

function mapDispatchToProps (props) {

const textoInscrito = props.texto

const textoEmPropsLock = textoInscrito

.toUpperCase()

}

→ Agora, vamos fazer ele retornar um HTML com uma
div e o texto em props lock ali no meio..

Basta add um return <div>{ | }</div>

esse represente o ponto VISUAL do nosso componente.

Dá pra, injetar a variável ali dentro, mundo 1
introdução u pronto!

Agora, toda vez que o componente for usado, ele
vai retornar o valor em props.value outra delas.

<LoopsleckText> = "Me deixe um loopsleck por favor" >

```
function Loopsleck(props) {  
  const textoInviado = props.texto  
  const textoFormatado = textoInviado.toUpperCase()  
  return <div> {textoEmLoopsleck} </div>  
}
```

* Vai ser livremente reutilizar componentes dentro de
outros componentes, assim como tags HTML como
de outros tags pt formar um layout, formatação
de texto, inserir imagem, o que vai querer. ☺

⑥ descrever a propriedade CHILDREN de um componente

Porque, um componente Loopsleck é que muita coisa
que possa o texto dentro de sua propriedade não me fornece
uma usabilidade muito boa...

E essa propriedade é:

 (text)

Tentão, pt conseguir vencer o desafio, de problemas possa as ilustrações anteriores ap/ sólido real. ✓

No HTML, quando temos tags que armazenam conteúdo, observam uma estrutura tipo, um tipo de parenteses.

 massinha

A tag strong engloba o texto que está dentro dela. Por natureza, faz o texto ter um

PA RENTE S CO

com essa tag.

Outro exemplo:

→ Parent (pai/mãe)

<div>

<div> Nené </div>

<div> conteúdo </div>

<div>

{ CHILDREN

Então...

```
function Pagina () {  
    return <Lopstake> Show!! </Lopstake>  
}
```

O texto, nesse caso, é o children do nosso componente *

O time da React renderiza isso mesmo exatamente na estrutura, só conseguindo atingir esse nível, que também é enviado pelo props.

Então, não existe mais a PROPRIEDADE texto que vai ser injetada no props. Mas como vamos puxar o que está no CHILDREN desse componente?

Só é! O CHILDREN APARECE É INSERIDO NO PROPS!

Então basta a gente acessar o que está dentro de um props.children, só que só aparece o método nome esperado:

```
function Lopstake (props) {  
    const textoIntrido = props.children;  
    const textoEmLopstake = textoIntrido.toUpperCase();  
    return <div> {textoEmLopstake} </div>
```

function Página () {

{
 retorna <h1>Olá!!</h1>

Sendo no React é um componente!

Um dentro de outro, de mesma forma que um layout elástico, uma página clássica de HTML, por exemplo.

Reset

→ 1 componente que abrange toda a página, que dentro tem 1 componente de menu, que dentro tem 1 componente de botão.

controle visual + controle de máquina interna
da funcionalidade dentro do código componente;
não você quer implementar funcionalidades juntas.

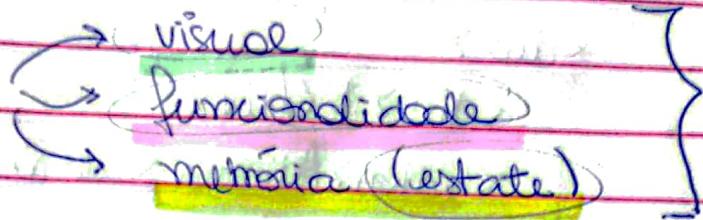
BOMBO é correspondido p/ Já e p/ cá,

giutto com o componente

São peçinhos de LEGO, separados, e encaixar

de várias formas diferentes para formar áudio ou visual diferentes.

COMPONENTE



④ Fazer o componente ter memória (estate)

Visual

funcionalidade

funcionamento do
usuário

ítem

fs

memória dentro
de um componente

Exemplos:

- contador;

- decidir se o componente de botão irá ativar ou não conforme alguma condição;

- manter qualquer dado guardado em memória e esperando (ex: algo digitado em formulário)

Então, vamos criar esse componente chamado **CONTADOR**

function Pagina()

return <contador>

}

{} se voce quiser autoexecutar

E vamos dar vida a ele a partir da função **função** tradicional ...

```

    } function botaoAdicionar() {
        return (
            <div>
                <div></div>
                <button>Adicionar</button>
            </div>
        )
    }

```

Essa é a parte visual,
agora vamos implementar
a funcionalidade:

- (1) Uniquílico o HTML interno.
- (2) Abra tudo com 1 div.
- (3) coloque o 1º div dentro do conteúdo.
- (4) 1 botão que, quando clicado, vai incrementar esse número.

(1) Antes da parte visual do componente, crie a função adicional, que representa a única coisa que faz é o console.log de 1 mensagem.

function botaoAdicionar() {

```

    } function adicionar() {
        console.log('Adicionou');
    }

```

Nos só declarar assim assim não fará nada, então temos que vincular os eventos de onclick do botão.

ON CLICK

no botão ...

return()

B
U
T
O
N

```
<div>
  <div> </div>
  <button onclick> Adiciona </button>
```

<div>

Adiciona </button>

criar todo vez que se elemento
for clicado

No React, a forma

de possuir ls só dentro

do parte visual é

por INTERPOLAÇÃO, mundo chave.

Eventos

<button onclick = {}> Adiciona </button>

{ adicionar contador }

* E, se dizer o console do navegador, somos ver que
ele só diga que eu fize, num controle. Isso não
opõe.

Agora, falta mexer no **ESTADO** do componente ...

Usa State

Preparamos **guardar o valor**

do contador em algum lugar ...

é só somando e aumentando o número ...

state dentro do componente

1º

Forma ERRADA de fazer isso ...

⑤ **Continuar o comportamento declarativo da parte visual**

Se a parte visual é declarativa, basta eu declarar os valores no HTML e iria dizeria sempre refletir o valor que está nessa variável ...

E se declarar que:

contador = contador + 1;

e, logo embaixo, fazer o log, pra acompanhar esse valor.

```
function adicionar contador () {  
    contador = contador + 1;  
    console.log(contador);  
}
```

→ "Para compreender o status da variável var."

Visto daí que, o nº que foi declarado na escritura não muda... mas em "log" de inspeção, o contador está contando.

Nas dessa forma, NÃO funciona o React.



Pensando, a ideia seria fazer como o JQuery, que é o código que, localizar o div e innerHTML, de forma imperativa, fizer:

"Inque o seu texto dentro pt 6,
após 2, use ..."

Nos isso quebra o bloco de você declarar interface e " pronto, dixa só se estudar resenha."

SEPARAR A DECLARAÇÃO DA INTERFACE

DE SUA FUNCIONALIDADE !

importante ☺

Outro jeito: React atualiza só a parte VISUAL o resto
memória a 60 atualizações por segundo, por exemplo.

Assim, refletiu sobre mais, acho dessa forma de
atualização.

Nisso, podemos otimizar! ☺

→ **use State** →
(usa estado)

Isso é um HOOK, um ganchinho que o React
fornecê, que serve literalmente pra usar emponchos num
ponto da sua aplicação e fazer:

"Quando isso acontecer, atualize o valor
de quello corinásser!"

React: Show! atualizada! ☺

- 1º Deixar só a cosa da função;
- 2º importar a funcionalidade useState do React,
que é o HOOK.

`import { useState } from 'react';`

③ chamar o `useState` porque eu quero que o meu componente use `state`, temos memória dentro dele.



`useState` deixa o valor inicial, que no caso do contador vemos sempre com o valor de 1.

■ Ao executar este função, vai retornar 2 coisas:

→ A variável que a gente vai usar para contá-lo, que inicialmente é o valor 1;

→ É uma função especializada em atualizar essa variável. Porque, quando ela for chamada, ela sabe que precisa atualizar a interface também.

Convenção: `useSetCounter()`

setar o valor que a gente quer

[`contador`, `setContador`]

↓ ↓

nova variável função especializada em atualizar
essa variável

Usando o próprio fs, podemos gerar:

Atribuição via DESESTRUTURAÇÃO

→ Destruturalizar esse array de Suposições e atribuir a cada posição um variável independente nesse array.

É só uma forma de extrair dados de variável (ou array) e atribuir diretamente um outros variáveis.

A variável contadora, que tb está associada à parte VISUAL domui componente, é a função especial nextContador() / quando o gente quiser visualizar esse variável e informar ao React que a interface deve ser recalculada.

→ Ok, mas como coloco isso na internet up no mundo real?