

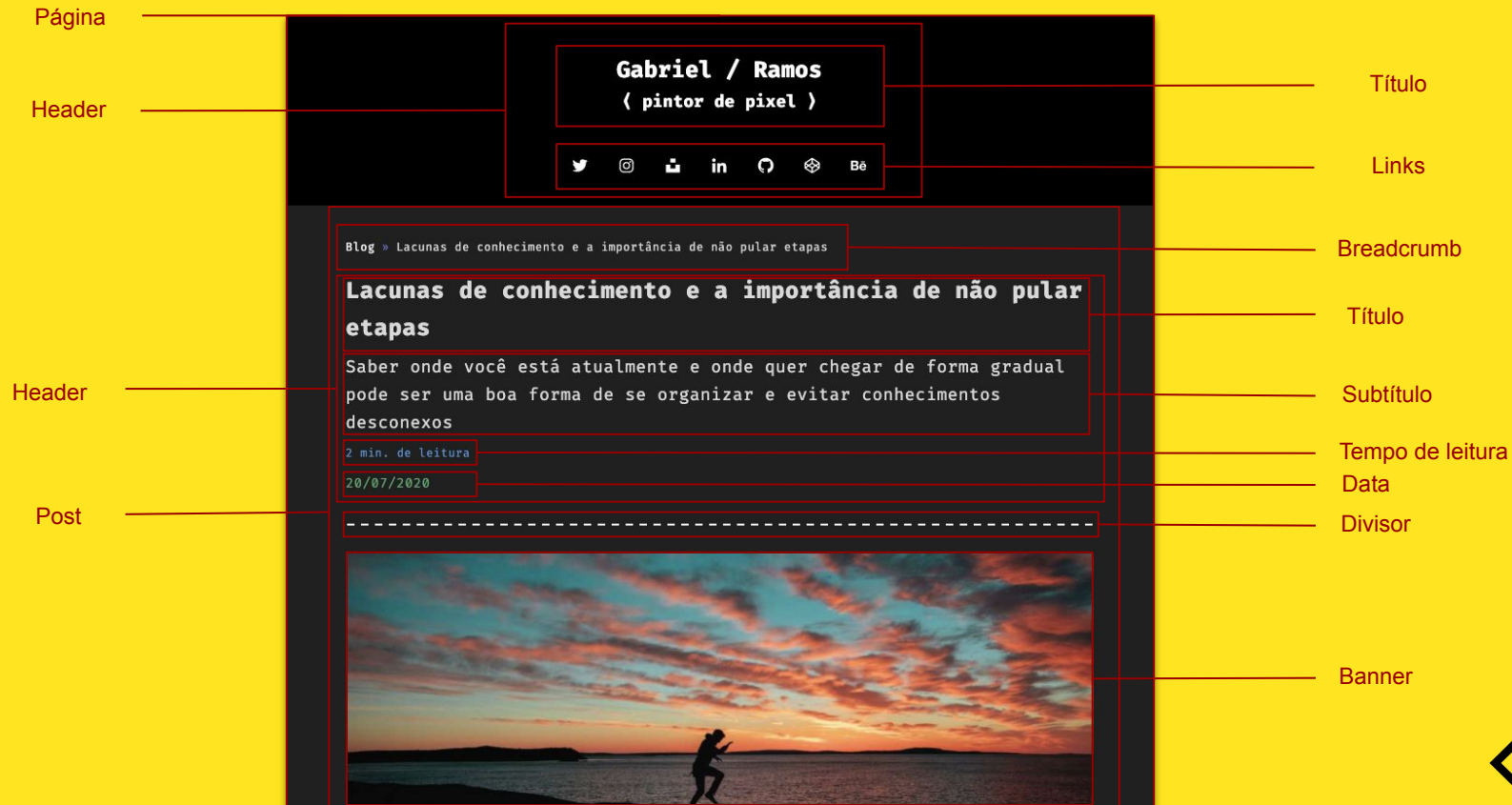


React Hooks

Quais os ganhos de utilizar ferramentas como
React?

Bibliotecas como o **React** nos permitem escrever interfaces de simples forma reutilizável

Estrutura baseada em Componentes



Mas nem toda **interface** é estática... E na grande maioria das vezes precisamos **gerenciar estados**

Tá...
Mas o que é **estado**?

click

carregamento de
dados

Interação com
uma página

Podemos imaginar **estado** como uma **relação entre
nossa interface e eventos**

abrir/fechar um
menu

alteração de um
formulário

O que aparece na tela ao clicar no **botão**?

```
import React from 'react';

let contagem = 1;

const Contador = () => {
  const onClick = () => {
    contagem++;
  }

  return (
    <div>
      <button onClick={onClick}>
        clique para contar
      </button>
    </div>
  )
}

export default Contador;
```


Estado também está relacionado com a **re-renderização** de um componente

Hooks são uma nova adição ao React 16.8. Eles permitem que você use o state e outros recursos do React sem escrever uma classe.

Hook de Estado: **useState**

Armazena um valor de estado e disparar uma re-renderização quando modificado

useState

importar o hook

```
import React, { useState } from 'react';
```

```
const Menu = () => {
```

```
  const [open, setOpen] = useState(false);
```

```
  // const estado = useState(false);
```

```
  // const open = estado[0];
```

```
  // const setOpen = estado[1];
```

Executar o hook
passando um valor de
estado inicial

Criar duas variáveis com o
retorno dessa função
(exemplo com desestruturação
do array de retorno)

Essa função retorna um array com 2
valores:

- O valor de estado
- Uma função para alterar esse valor

useState

Cria uma função para mudar o estado (no **onClick** do botão abaixo)

Caso **open** seja **true**, renderiza a lista

```
import React, { useState } from 'react';

const Menu = () => {
  const [open, setOpen] = useState(false);
  const toggle = () => setOpen(!open)

  return (
    <nav>
      <button onClick={toggle}>Abrir</button>
      {open && (
        <ul>
          <li>Home</li>
          <li>Sobre</li>
          <li>Contato</li>
        </ul>
      )}
    </nav>
  );
}

export default Menu;
```

Cria as variáveis de estado

Indica função para ser executada no click (**toggle**)

Hook de efeitos colaterais: **useEffect**

Sincroniza valores de estado com qualquer efeito colateral

useEffect

importar o hook

```
import React, { useState, useEffect } from 'react';

const Menu = () => {
  const [open, setOpen] = useState(false);
  const toggle = () => setOpen(!open)

  useEffect(() => {
  }, []);
```

Executar o hook passando:

- Função a ser executada
- Array de dependências (utilizado para disparar a função novamente)
 - Se o array for vazio, o **useEffect** é disparado somente ao criar o componente
 - Se não passar nada (nem um array vazio) o **useEffect** será disparado quando qualquer variável de estado mudar

useEffect

Executa a função **fazQualquerCoisaAi** sempre que o valor de **open** for alterado

```
import React, { useState, useEffect } from 'react';
```

```
const fazQualquerCoisaAi = valor => {  
  console.log('Executando qualquer coisa');  
  console.log(`O valor é ${valor}`);  
}
```

Função que faz qualquer coisa

```
const Menu = () => {  
  const [open, setOpen] = useState(false);  
  const toggle = () => setOpen(!open)
```

```
    useEffect(() => {  
      fazQualquerCoisaAi(open);  
    }, [open]);
```

```
    return (  
      <nav>  
        <button onClick={toggle}>abrir</button>  
        {open && (  
          <ul>  
            <li>Home</li>  
            <li>Sobre</li>  
            <li>Contato</li>  
          </ul>  
        )}  
      </nav>  
    );  
  }  
}
```

```
export default Menu;
```




“Com grandes poderes, vêm grandes responsabilidades”

Ben, Tio

Regras dos Hooks

- Utilize **hooks** apenas em nível superior (ao declarar seu componente), ou seja, **não** utilize em:
 - condicionais
 - loops
 - funções aninhadas
- Utilize **hooks** apenas dentro de funções React

* não seguir essas regras pode causar **sérios problemas e comportamentos inesperados** nos seus componentes *



A light-colored dog is captured mid-jump, with its front paws raised high and its head tilted back. The entire scene is set against a solid, bright yellow background, which also serves as the backdrop for the text.

DÚVIDAS

Issaê

Links úteis:

- [Introdução aos hooks](#)