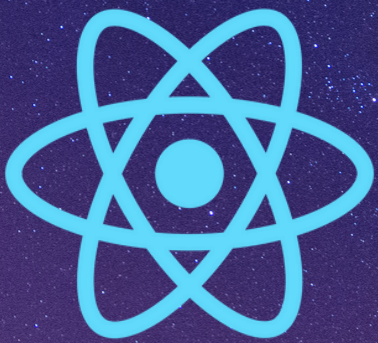
# Fundamentos do React

## TUDO QUE VOCÊ PRECISA PARA CRIAR PROJETOS



POR MATHEUS BATTISTI
HORA DE CODAR

# SOBRE O AUTOR

Matheus Battisti é desenvolvedor há mais de 7 anos e ama a tecnologia da informação como um todo.

Está sempre em busca de um aprendizado constante e evolutivo.

Atua como desenvolvedor Full Stack, desenvolvendo e-commerces e aplicações com arquitetura de microsserviços.

É criador do Hora de Codar, que hoje passa de 30 mil alunos.

Python é uma de suas linguagens preferidas, já utilizou para Data Science como também desenvolvimento web.

Seu objetivo é sempre passar o máximo de conhecimento possível, unindo toda a experiência de seus estudos e também de sua carreira como desenvolvedor.

# INTRODUÇÃO

Este eBook é destinado  a quem deseja conhecer os fundamentos do React.

Vamos de forma teórica e prática, conhecer os principais elementos, que vão determinar o sucesso de suas SPA's.

A ideia é apresentar também um contexto de como você pode utilizar cada recurso a seu favor e também quais as melhores práticas.

Após a leitura, você estará preparado para entender e também criar suas aplicações em React, de forma conciente.

Está pronto? Vamos começar!

# INSTALAÇÃO DO REACT

A instalação do React pode ser feita de diversas maneiras, mas a atual melhor escolha é o **create-react-app**

Um script que podemos executar através do npm, que cria a estrutura base para um projeto em React

Você pode utilizar os comandos

```
npx create-react-app hellworld

cd hellworld

npm start
```

Com estes comandos você tem uma aplicação funcional de React rodando na sua máquina.

E com toda certeza este é seu o primeiro passo dado na direção certa!

# JSX

O **JSX** é a forma que escrevemos HTML no React.

Podemos realizar diversas outras ações, como: imprimir dados de variáveis em nossos templates.

Como também executar códigos de JavaScript, como um if.

Você pode escrever JSX no arquivo App.js do React, ou em qualquer componente, veja na próxima página um exemplo de JSX

# JSX

Exemplo de JSX:

```jsx
function App() {
  const name = "Matheus";

  function sum(a, b) {
    return a + b;
  }

  const url = "https://via.placeholder.com/150";

  return (
    <div className="App">
      <p>Testando o JSX</p>
      <p>Olá {name}</p>
      <p>A soma é: {sum(1, 2)}</p>
      <img src={url} alt="Minha Imagem" />
    </div>
  );
}

export default App;
```

Note que é muito semelhante ao HTML, mas precisamos evitar algumas palavras como class, que pertende ao JavaScript, então colocamos **className**

E outro detalhe: para interportar valores dinâmicos utilizamos {**variável**}

# COMPONENTES

Componentes fazem parte da arquitetura do React.

Em uma aplicação podem existir diversos componentes, que são partes do site em proporções menores.

Por exemplo: barra de navegação, card de produto, título da página.

Todos estes casos podem ser divididos em componentes.

Melhorando nosso código, pois acabamos reaproveitando os componentes em diversos locais.

Na próxima página você encontra um exemplo de componente.

## COMPONENTES

Veja o código a seguir:

```
function Frase() {
  return (
    <div>
      <p>Este componente é apenas uma frase!</p>
    </div>
  );
}

export default Frase;
```

Normalmente os componentes ficam na pasta **components**

Precisamos **criar uma função e exportá-la**.

Note que aqui também utilizamos o JSX, que é retornado entre parenteses, pois possui mais de uma linha de código.

Para utilizar este componente precisamos importá-lo em um arquivo, veja:

# COMPONENTES

Exemplo de importação:

```
1  import Frase from "./Frase";
2
3  function HellWorld() {
4    return (
5      <div>
6        <h1>Meu primeiro componente</h1>
7        <Frase />
8        <Frase />
9      </div>
10   );
11 }
12
13 export default HellWorld;
```

Utilizamos o componente como uma tag, com a sintaxe <Frase />

Note que há o fechamento da tag também

O nome do componente é refletido pelo qual foi importado, como podemos ver na primeira linha do código

# PROPS

Props são os dados que passamos do componente pai para o componente filho.

Na maioria das vezes nós não modificamos estes dados, ou seja, só os utilizamos.

É um recurso extremamente utilizado em React.

Veja:

```
function SayMyName(props) {
  return (
    <div>
      <p>Fala aí {props.nome}, suave?</p>
    </div>
  );
}

export default SayMyName;
```

Aqui o componente SayMyName aceita a props.

E nele utilizamos a props chamada name

Perceba que precisamos declarar na criação da função

## PROPS

Agora veja como passar as props para o componente que as aceita:

```
9         <SayMyName nome="Matheus" />
10        <SayMyName nome="João" />
11        <SayMyName nome="Francisco" />
```

Aqui reutilizamos o componente SayMyName três vezes

E em cada um passamos um nome diferente.

Vemos dois recurso sendo utilizados:

- Passagem de props;
- Reutilização de componentes;

## ADICIONANDO CSS

Para adicionar CSS a nossa aplicação temos acesso a um recurso chamado **CSS Modules**

Basta criar um arquivo com o mesmo nome do componente que queremos colocar estilos

Finalizando com **module.css**

Então se temos um componente chamado Frase.js

Vamos criar o arquivo Frase.module.css

Veja:

# ADICIONANDO CSS

No componente vamos importar o nosso arquivo de CSS

E acessar as classes que foram definidas nele

Inserindo estas nos elementos do nosso componente, e esta é a implementação final:

```
import styles from "./Frase.module.css";

function Frase() {
  return (
    <div className={styles.fraseContainer}>
      <p className={styles.fraseContent}>Este componente é apenas uma frase!</p>
    </div>
  );
}

export default Frase;
```

Importamos o CSS com o nome de styles, o que é um padrão bem utilizado

Acessamos as classes definidas no CSS como propriedades de um objeto

Agora nosso componente tem CSS!

# FRAGMENTS

Sempre que definimos um componente precisamos retornar apenas um elemento

Os fragments resolvem este problema, simplificando o nosso HTML

Podemos então retornar uma espécie de tag especial, que permite um componente retornar dois ou mais elementos

Veja a implementação:

```
function Item(props) {
  return (
    <>
      <li>{props.marca}</li>
    </>
  )
}

export default Item
```

Aqui retornamos um li apenas, com o fragment

Que possui a seguinte sintaxe: <> ... </>

Ou seja, abrimos uma tag de fragment e depois precisamos fechá-la

# TIPOS DE PROPS

Podemos definir o tipo de uma props e também valores default para ela

Isso deixa muito mais sólido nosso programa, limitando o que ele aceita para um melhor funcionamento

Veja o exemplo:

```
import PropTypes from 'prop-types'

function Item(props) {
  return (
    <>
      <li>
        {props.marca} - {props.ano_lancamento}
      </li>
    </>
  )
}

export default Item

Item.propTypes = {
  marca: PropTypes.string.isRequired,
  ano_lancamento: PropTypes.number,
}

Item.defaultProps = {
  marca: 'Faltou a marca',
  ano_lancamento: 1,
}
```

Para definir o tipo precisamos importar prop-types

E depois aplicar as propriedades os tipos que queremos definir

# TIPOS DE PROPS

Para definir um valor default não é necessário importar nada

Porém precisamos criar a propriedade defaultProps e inserir o valor padrão para as que precisamos

Isso faz com que se não passarmos valor nenhum, aqueles valores sejam exibidos no lugar de um valor vazio, o que pode ser interessante em diversos casos

# EVENTOS

Os eventos do React são recursos fundamentais para tratar os envios de dados dos usuários

Como por exemplo o envio de um formulário

Então para cada ação teremos um evento (clique, teclas, formulários e etc.)

Veja um exemplo:

```
function Form() {
  function cadastrarUsuario(e) {
    e.preventDefault()
    console.log('Usuário cadastrado!')
  }

  return (
    <div>
      <h1>Meu cadastro:</h1>
      <form onSubmit={cadastrarUsuario}>
        <div>
          <input type="text" placeholder="Digite seu nome" /
        </div>
        <div>
          <input type="submit" value="Cadastrar" />
        </div>
      </form>
    </div>
  )
}

export default Form
```

# EVENTOS

O evento de submit dispara um método, ou seja, uma função

E nela podemos executar um código baseado na lógica da nossa aplicação

Lidando também com os dados que estamos manipulando no momento

## CURSO COMPLETO

Este curso completo de fundamentos do React está disponível em vídeo também, e ao final criamos um projeto

**Tudo de forma gratuita**, no meu canal de YouTube

O link para a playlist é: **https://youtu.be/FXqX7oof0I4**

Não esqueça de deixar seu like e se inscrever no canal, me ajuda bastante = )

# Obrigado

Enquanto houver pessoas dispostas
a aprender, eu estarei criando
conteúdo para enriquecer ainda
mais os seus conhecimentos

Matheus Battisti

Hora de Codar