

Bootcamp Full Stack

React

Prof. Raphael Gomide

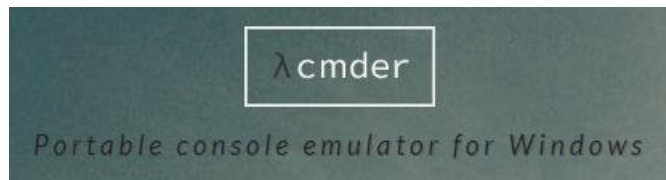


Aula 1. Ambiente de desenvolvimento

☐ Acompanhe o professor.

– Instalação e testes:

- [Visual Studio Code](#) e [extensão para React](#).
- Terminal de comandos – [Cmder](#).
- [Node.js](#) – versão **12.16.2** no **Windows**.
- [Yarn](#) – versão **1.22.1** no **Windows**.



Próxima aula

- ☐ Introdução ao React.



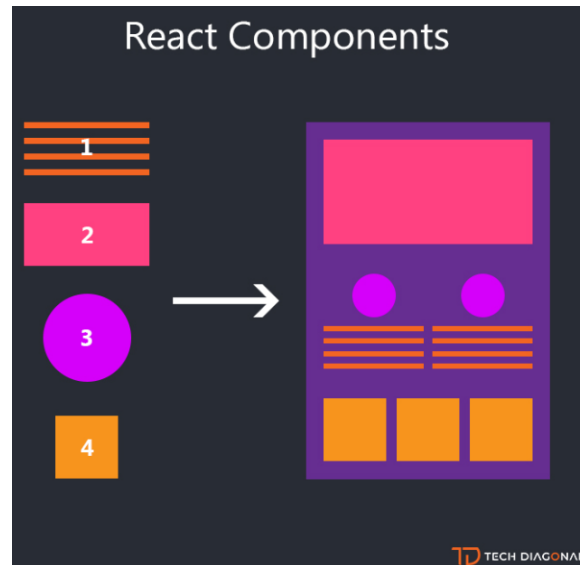
Aula 2. Introdução ao React

☐ React:

- O que é o React?
- Virtual DOM.
- Criação de projetos com o pacote **create-react-app**.

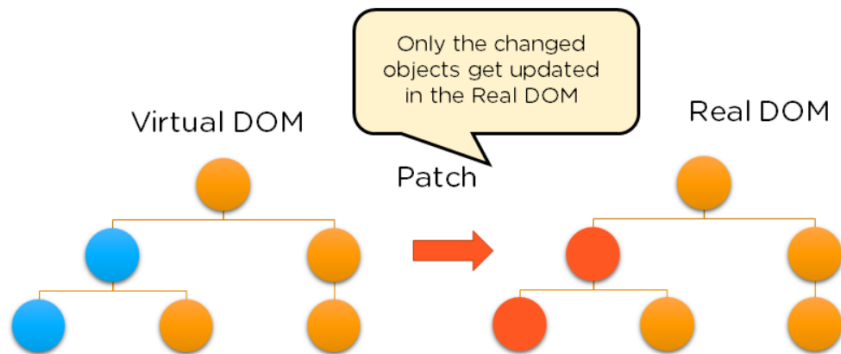
O que é o React?

- “A JavaScript **library** for building user interfaces”.
- Declarativo:
 - Componentes reativos com JSX.
 - **Mais foco** no **estado** do app e **regras de negócio**;
 - **Menos foco** em manipulação do **DOM manual**.
 - Manipulação do DOM performática (Virtual DOM).
- Baseado em componentes:
 - Alto grau de **reutilização de código**.



[Fonte](#)

- Manipulação performática do DOM.
- A manipulação do DOM é considerada uma operação **cara (lenta)**.
- O React só modifica o DOM nos locais que foram realmente alterados.
- Esse processo é mais conhecido como [Reconciliation](#).



[Fonte](#)

- Acompanhe o professor nas seguintes tarefas:
 - Criação de um projeto com **create-react-app**.
 - Navegação pelas principais pastas e arquivos do projeto.
 - Execução do projeto.
 - Modificação do código-fonte do projeto.

- ☑ React – biblioteca para manipulação de interfaces de usuário.
- ☑ Evita manipulação manual do DOM.
- ☑ Manipula o DOM de forma performática.
- ☑ Mais foco no estado e regras de negócio da aplicação.
- ☑ A forma mais fácil e simples de se iniciar um projeto React é com o pacote **create-react-app**.
- ☑ Será sempre utilizada a versão **3.4.1** do pacote **create-react-app** para garantir uma melhor compatibilidade nos projetos.

Próxima aula

☐ Desafio 01.




Aula 3. Desafio 01

☐ Desafio 01.

❑ Criação de um app simples.

- Botão que preenche uma lista não ordenada de itens cujo conteúdo é a data/hora do clique.
- Este projeto será desenvolvido em:
 - JavaScript puro não-performático.
 - JavaScript puro performático.
 - React com Class Components.
 - React com Functional Components + Hooks.
- Acompanhe o professor.

Clique aqui



- sexta-feira, 17/04/2020 11:05:51
- sexta-feira, 17/04/2020 11:05:52
- sexta-feira, 17/04/2020 11:05:52
- sexta-feira, 17/04/2020 11:05:53
- sexta-feira, 17/04/2020 11:05:53
- sexta-feira, 17/04/2020 11:05:53
- sexta-feira, 17/04/2020 11:05:54
- sexta-feira, 17/04/2020 11:05:54
- sexta-feira, 17/04/2020 11:05:55
- sexta-feira, 17/04/2020 11:05:56

- ☑ É possível deixar o app performático com **JavaScript puro**.
Entretanto, isso tende a ser **trabalhoso**.
- ☑ A utilização do **React** torna a **escrita mais declarativa** e mantém o **app performático** quanto à **manipulação do DOM**.
- ☑ A utilização de **React Hooks** pode deixar a **escrita ainda mais declarativa** em comparação aos **Class Components**.

■ Próxima aula

- ❑ Classes com JavaScript.



Aula 4. Classes com JavaScript

☐ Classes com JavaScript:

- Sintaxe.
- Herança.
- Implementação:
 - Criação das classes Animal, Cat e Dog.

- ☑ Classes no JavaScript – açúcar sintático para funções especiais.
- ☑ Suporte ao construtor.
- ☑ Suporte a atributos.
- ☑ Suporte a métodos.
- ☑ Suporte a herança.
- ☑ Utilizada pelo React em Class Components.

- ❑ Class Components – parte 01.



Aula 5. Class Components – parte 01

☐ Class Components – parte 01:

– Acompanhe o professor:

- Criação do projeto **react-counter-01**.
- Os comandos **import** x **export**.
- Herança da classe **Component**.
- Construtor de um **Class Component**.
- O método **render()**.
- JSX (JavaScript and XML).
- Interpolação de dados no JSX.
- Considerações sobre a utilização de CSS no React.



- ✓ **Class Components** – uma das formas de se criar componentes com React.
- ✓ Toda **Class Component** herda de **React.Component**.
- ✓ Se utilizar **construtor**, deve-se invocar **super()**; na primeira instrução.
- ✓ **Class Components** permitem a utilização de **atributos** e **métodos** assim como qualquer classe **JavaScript**.
- ✓ O principal método de um **Class Component** é o **render()**.
- ✓ No React, é utilizado **JSX** para a confecção da **interface gráfica**.
- ✓ Para interpolar expressões JavaScript, utilize chaves **{}**.
- ✓ Será utilizado **CSS Modules** nos projetos.
- ✓ O exemplo apresentado ainda tem muito a ser melhorado.

- ❑ Class Components – parte 02.

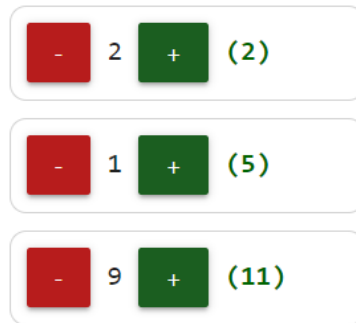


Aula 6. Class Components – parte 02

☐ Class Components – parte 02:

– Acompanhe o professor com explicações de:

- `this.state`.
- O método `this.setState()`.
- Comunicação entre componentes através de props.
- One-way data flow.
- Projeto **react-counter-02**.



- ✓ Para trabalhar com estado, defina valores em **this.state**.
- ✓ O estado deve ser alterado com **this.setState()**.
- ✓ A comunicação entre componentes é feita com **props**.
- ✓ O React implementa a estratégia de **one-way data flow**.
- ✓ Utilize a **prop onClick** para “**escutar**” o **clique** de botões com React.
- ✓ Uma boa prática em **eventos** é utilizar **somente** a **referência** do método. Na **implementação do método**, utilize **arrow functions**.

- ❑ Ciclo de vida de Class Components.



Aula 7. Ciclo de vida de Class Components

❑ Principais métodos do ciclo de vida de um Class Component:

- Acompanhe o professor:
 - O método **componentDidMount**.
 - O método **componentDidUpdate**.
 - O método **componentWillUnmount**.
 - Implementação do projeto **react-lifecycle**.

Abra o console!

Exibir lista



Lista aberta por 5 segundos...



Jasão



Elenise



Izalino



Anania



Suleni



Valente



Kalinka



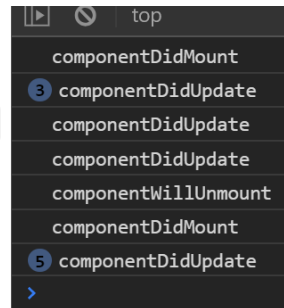
Helen



Aminadabe



Nerci



- ☑ **componentDidMount** – executado **após** o primeiro render() e útil para requisições HTTP, por exemplo.
- ☑ **componentDidUpdate** – executado **após** toda invocação de render() e útil para aplicação de “efeitos colaterais”.
- ☑ **componentWillUnmount** – executado **antes** do componente “morrer” e útil para finalização de objetos, como por exemplo **clearInterval**.
- ☑ Para mais informações sobre os ciclos de vida de **Class Components** no React, acesse [este link](#).

Próxima aula

☐ Desafio 02.



Aula 8. Desafio 02

Desafio 02

- Criação de um app para listar países a partir da API <https://restcountries.eu/rest/v2/all>.
- Input para filtrar os países
- Exibir quantidade de países e soma da população dos países filtrados.
- Acompanhe o professor.



- ☑ Para monitorar inputs com React, é importante definir os atributos **value** e **onChange**.
- ☑ Funções simples, comuns a diversos componentes, podem se situar em módulos isolados (**helpers**). Assim, são mais facilmente reaproveitados.

■ Próxima aula

□ Functional Components.



Aula 9. Functional Components

□ Functional Components:

- Acompanhe o professor:
 - Conversão de **Class Components** para **Functional Components**.
 - Utilização dos seguintes projetos:
 - react-counter-02
 - react-lifecycle
 - desafio-02

- ✓ A **escrita** de **Functional Components** é mais **simples** que a de **Class Components**.
- ✓ Em **Functional Components**, utiliza-se **funções**.
- ✓ Em **Functional Components**, não há **state**. Utiliza-se somente **props**.
- ✓ **Functional Components** são, em regra, somente leitura.
- ✓ As **props** podem ser **desestruturadas** já nos **parâmetros** da **função**.
- ✓ Functional Components retornam, em regra, JSX.
- ✓ **Não** há o método **render()** em **Functional Components**.
- ✓ Não há **ciclo de vida (lifecycle)** em **Functional Components**.
- ✓ **Não** há **this** em **Functional Components**.
- ✓ **Functional Components** utilizam **funções internas (closures)**.

Próxima aula

☐ Desafio 03.

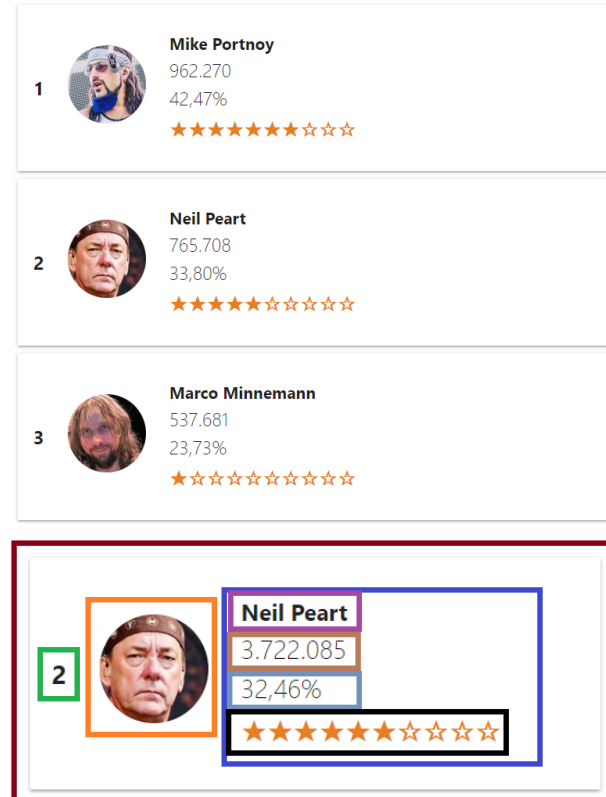


Aula 10. Desafio 03

Desafio 03

- Criação de um app monitorar votações.
- O Back End será fornecido.
 - Votações geradas aleatoriamente a cada 100 milissegundos.
 - Popularidade de 1 a 10 atualizada a cada 10 segundos.
- Quebre os componentes ao máximo.
- Utilize **Functional Components** sempre que possível.
- Acompanhe o professor.

Votação



Próxima aula

☐ React Hooks.



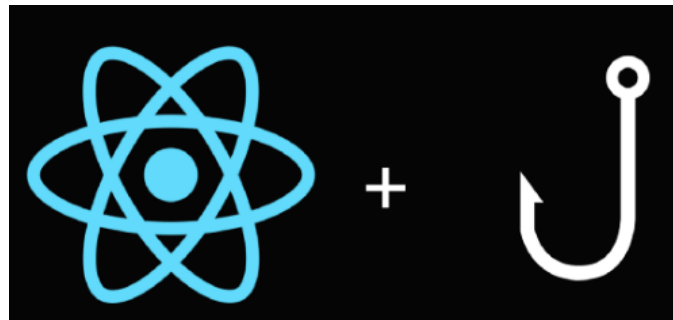
Aula 11. React Hooks

☐ React Hooks

- Introdução.
- O Hook useState.
- O Hook useEffect.
- Migração de projetos.

React Hooks – introdução

- Criado pelo Facebook no fim de 2018.
- Fornece uma escrita ainda mais declarativa.
- Utiliza **closures** e **array destructuring**.
- Permite a utilização de **estado** em **Functional Components**.
- Não pretende (ainda) substituir totalmente as **Class Components**.
- Principais hooks: **useState** e **useEffect**.
- Mais informações [aqui](#).



[Fonte](#)

- Visa substituir **this.state** e **this.setState** de **Class Components**.
- Escrita mais declarativa.
- Utiliza array destructuring.
- Sintaxe padrão: `const [variable, setVariable] = useState(0);`
- Na instrução acima, **variable** representa a **variável** de **estado**.
- Na instrução acima, **setVariable** representa a **função atualizadora**.
- As **funções atualizadoras** só atuam na **variável** ao qual “apontam”.
- **Não** há mais o *merge* de **this.setState**.

- Visa substituir **componentDidMount**, **componentDidUpdate** e **componentWillUnmount** de **Class Components**.
- Escrita mais declarativa.
- Com **useEffect**, não há mais o conceito de montagem do componente (**mounting**) e atualização do componente (**updating**).
- **useEffect** tem um **modelo mental diferente** dos métodos de **ciclo de vida** – a ideia de **useEffect** é **sincronizar** todo o DOM conforme os valores de **props** e **state**. Mais informações [aqui](#).

- **useEffect** permite utilizar um parâmetro extra, conhecido como array de dependências (dependency array ou, simplesmente, **deps**).
 - Quando **não há** o parâmetro, **useEffect** é **invocado após qualquer atualização** – **semelhante a `componentDidUpdate`**.
 - Quando o **parâmetro** é [], **useEffect** é **invocado apenas uma vez** – **semelhante a `componentDidMount`**.
 - Quando o parâmetro está preenchido com [**state1, state2, etc**], **useEffect** é **invocado após a atualização de estado de qualquer uma das variáveis**.
- Quando **há retorno na função** – **useEffect** utiliza o retorno para **eliminar recursos** – **semelhante ao `componentWillUnmount`**.

React Hooks – Migração de projetos

- Acompanhe o professor
- Serão migrados os seguintes projetos:
 - react-counter-02-functional
 - react-desafio-02-functional
 - react-lifecycle-functional
 - react-desafio-03

Próxima aula

☐ Desafio 04.