

# Componentes de Classes

DEVinHouse

Parcerias para desenvolver a sua carreira

**SENAI**

<LAB365>

# Componentes de Classe

Componentes de classe são componentes que possuem um alto nível de poder dentro da aplicação, pois além gerenciar o próprio estado, herdam os chamados métodos de ciclo de vida do React, lidam com partes lógicas da aplicação e manipulam eventos através de métodos que podem ser invocados em qualquer lugar do componente ou em seus filhos.

# Exemplo 1

```
1  import { Component } from 'react'
2
3  class ClassBasedCase1 extends Component {
4      render() {
5          return <h3>Olá, Mundo!</h3>
6      }
7  }
8
9  export default ClassBasedCase1;
```

# Constructor - Componentes de Classe

O **constructor** para um componente React é chamado antes que este componente seja montado. Por meio dele definimos também o recebimento das props mas, diferentemente dos modelos funcionais, precisamos declarar um comando ***"super(props)"*** antes da inicialização dos outros recursos do componente.

## Exemplo - Constructor

```
constructor(props) {  
  super(props);  
  this.state = { text: 'Senai' };  
}
```

Mas o que é o método “***super(props)***” falado anteriormente?

## Exemplo - Constructor

```
constructor(props) {  
  super(props);  
  this.state = { text: 'Senai' };  
}
```

No JavaScript, o **super** refere-se ao construtor da classe pai.  
(No nosso exemplo, ele refere-se à implementação de **extends Component** da classe.)

# Exemplo - Constructor

```
4  export default class Home extends Component {  
5  
6  
7  constructor(props) {  
8    // Aqui não podemos usar o "this"  
9    super(props);  
10   // Aqui podemos  
11   this.state = { text: 'Senai' };  
12 }  
13
```



*É importante lembrar que você não pode usar o **this** em um construtor até que você tenha chamado o construtor pai. Isso vale para qualquer código Javascript e é um dos conceitos da herança de classes.*

O componente React passa pelas seguintes fases ou ciclos de vida:

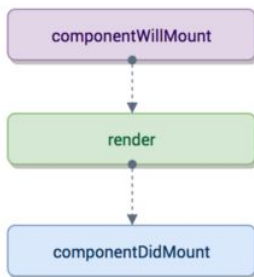
- Inicialização
- Montagem
- Atualização
- Desmontagem

# Exemplo - Constructor

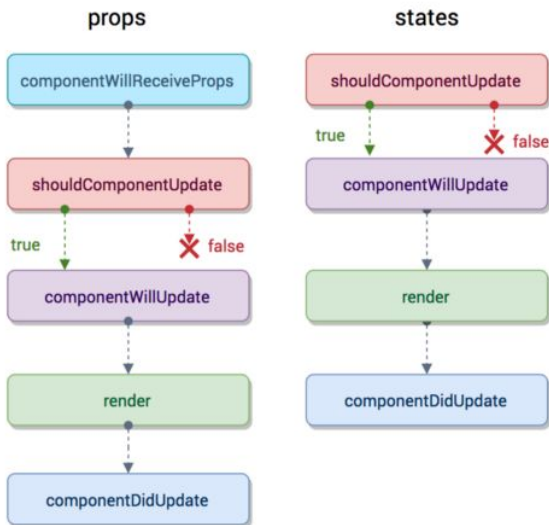
## Initialization

setup props and state

## Mounting



## Updation



## Unmounting

componentWillUnmount

# Inicialização - Ciclo de vida

Nesta fase, o componente React se prepara para sua inicialização, configurando os estados iniciais e props padrões se houverem.

```
6   constructor(props) {  
7     super(props);  
8     this.state = { text: 'Senai' };  
9   }
```

# Inicialização - Ciclo de vida

Nesta fase, o componente React se prepara para sua inicialização, configurando os estados iniciais e props padrões se houverem.

```
6   constructor(props) {  
7     super(props);  
8     this.state = { text: 'Senai' };  
9   }
```

# Montagem - Ciclo de vida

Depois de preparar com todas as necessidades básicas, estado e props, o nosso Componente React está pronto para ser montado no DOM do navegador. Esta fase fornece métodos que podem ser invocados antes e depois da montagem dos componentes. Os métodos que são chamados nesta fase são:

- **componentWillMount \*** - executado antes do componente ser montado no DOM.
- **componentDidMount** - executado após o componente ser montado no DOM.

## Exemplo - componentDidMount

```
11  componentDidMount() {  
12    // Aqui podemos realizar buscar dados de uma API.  
13    console.log('O componente foi montado.')  
14  }
```

# componentDidMount x useEffect

```
componentDidMount() {  
  console.log('O componente foi montado.')  
}
```



```
useEffect(() => {  
  console.log('O componente foi montado.')  
}, [])
```

Exemplos de ciclo de vida para verificar se o componente foi montado. Comparação entre funcionais e componente de classe.



# Atualização - Ciclo de vida

Esta fase começa quando o componente React já nasceu no navegador e cresce recebendo novas atualizações. O componente pode ser atualizado de duas maneiras, através do envio de novas props ou a atualização do seu estado.

# Atualização - Ciclo de vida

- **shouldComponentUpdate** - quando o componente recebe novas props ou estado (this.state). Retorna true/false, por padrão é true.
- **componentWillUpdate** - Este método é usado apenas para preparar a próxima renderização, diz que o componente está próximo de ser atualizado.
- **componentDidUpdate** - É executado quando o novo componente (já atualizado) foi atualizado no DOM.

# Desmontagem - Ciclo de vida

Nesta fase, o componente não é mais necessário e será desmontado do DOM. O método que se chama nesta fase é o seguinte:

- **componentWillUnmount** - Este método é o último método no ciclo de vida. Isso é executado imediatamente antes de o componente ser removido do DOM.

# Desmontagem - Ciclo de vida

```
componentWillUnmount() {  
  /**  
   * Caso seja uma ação de Logout podemos:  
   * - Limpar os dados do usuário  
   * - Resetar o storage e tokens de autenticação  
   */  
  console.log('O componente será desmontado')  
}
```

# componentWillUnmount x useEffect

```
componentWillUnmount() {  
  console.log('Componente sendo desmontado.')  
}
```



```
useEffect(() => {  
  return () => {  
    console.log('Componente sendo desmontado.')  
  };  
})
```

Exemplos de ciclo de vida para verificar se o componente será desmontado. Comparação entre funcionais e componente de classe.

# Gestão de Estado - this.state

O state dos componentes baseados em classe funciona da mesma forma que o hook `useState` mas tem uma forma de declaração diferente. Primeiramente definimos o `this.state` após o método “`super`” no constructor.

```
constructor(props) {  
  super(props);  
  this.state = { text: 'Senai' };  
  // Equivalente ao Hook:  
  // const [text, setText] = useState('Senai');  
}
```

# Gestão de Estado - this.state

```
13   changeText(event) {
14     this.setState({
15       text: event.target.value
16     });
17     // Equivalente a setText(event.target.value)
18   }
19
20   render() {
21     return (
22       <div>
23         <span><b>Texto digitado: </b>{this.state.text}</span>
24         /* Equivalente a {text} */
25
26         <input value={this.state.text} onChange={this.changeText}></input>
27         /* Equivalente a value={text} onChange={setText((e) => setText(e.target.value))} */
28       </div>
29     )
30   }
31 };
```

# Classname Dinâmica

DEVinHouse

Parcerias para desenvolver a sua carreira

**SENAI**


<LAB365>



***className*** é o atributo de um componente para atribuir classes css a ele, seu uso é simples e existe a possibilidade de adicionar um conteúdo dinâmico, como alteração de cor de um botão quando ele está ativo.

## className - Uso

```
16   const botaoAtivo = 1;  
17  
18   return (  
19     <div className='div-padrao'>  
20       <button className={botaoAtivo ? 'btn-ativo' : 'btn-inativo'}>Enviar</button>  
21     </div>  
22   )
```



Este modelo de vinculação de classe funciona como um if. Se a condição passar, adiciona o primeiro item ao className, caso a condição não seja validada o segundo item será adicionado.

## Exercício 1 - Componente de Perfil (15min)

Construir um componente, seja ele com modelo de classe ou funcional, para receber por props uma imagem, um nickname e uma url.

Ao lado deixo um modelo que pode ou não ser seguido para renderizar este componente.

### Usuários Github



@yanestevesufjf

<https://api.github.com/users/yanestevesufjf>

# INTERVALO DE AULA

## **DEV!**

Finalizamos o nosso primeiro período de hoje. Que tal descansar um pouco?!

Nos vemos em 20 minutos.

**Início:** 20:00

**Retorno:** 20:20



## Exercício 2 - Componente de Perfil

Modificar o exercício 1, adicionando um campo de pesquisa no componente Pai da renderização, adicionar um evento para realizar uma chamada para a API do Github a cada tecla digitada no input. Adicionar o retorno dos dados no array de usuários iniciado com useState.

### Usuários Github



@yanestevesufjf

<https://api.github.com/users/yanestevesufjf>

- Porque devemos escrever o super(props) - <https://overreacted.io/pt-br/why-do-we-write-super-props/>
- Métodos de ciclo de vida - <https://medium.com/creditas-tech/m%C3%A9todos-do-ciclo-de-vida-de-componentes-reactjs-um-mergulho-profundo-332ed7b3b782>
- Repo com os códigos das aulas - <https://github.com/yanestevesufjf/clamed-react>

## AVALIAÇÃO DOCENTE

O que você está achando das minhas aulas neste conteúdo?

[Clique aqui](#) ou escaneie o QRCode ao lado para avaliar minha aula.

Sinta-se à vontade para fornecer uma avaliação sempre que achar necessário.





# DEVinHouse

Parcerias para desenvolver a sua carreira

**OBRIGADO!**



<LAB365>



# AGENDA

- Informações do template
- Exemplo com texto (Light mode)
- Exemplo com texto (Dark mode)
- Exemplo com imagens (Light mode)
- Exemplo com imagens (Dark mode)

# INFORMAÇÕES DO TEMPLATE

- Título da Apresentação:
  - Fonte: Ubuntu Bold
  - Formato: Maiúsculo
  - Tamanho: 34
  - Cor: Branco
- Título do Slide:
  - Fonte: Ubuntu Bold
  - Formato: Maiúsculo
  - Tamanho: 22
  - Cor: Branco
- Parágrafos:
  - Fonte: Open Sans Normal
  - Tamanho: 14 a 18
  - Cores: Branco (Dark Mode) ou Preto (Light Mode)
- Marcadores de tópicos:
  - Formatos: Símbolos ou Alfanuméricos
  - Cor: Laranja
- Padrão de Cores:
  - Cinza - #868584
  - Preto - #1C1C19
  - Branco - #FAFAFA
  - Laranja - #F08305
  - Rosa - #c71d81
  - Azul - #0e1d8e

## EXEMPLO COM TEXTO (LIGHT MODE)

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.

## EXEMPLO COM TEXTO (DARK MODE)

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.

# EXEMPLO COM IMAGENS (LIGHT MODE)

## Título

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.



## Título

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.

# EXEMPLO COM IMAGENS (DARK MODE)

## Título

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.



## Título

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.