FIAP

GRADUAÇÃO

45697056

# TDS

# Responsive Web Development

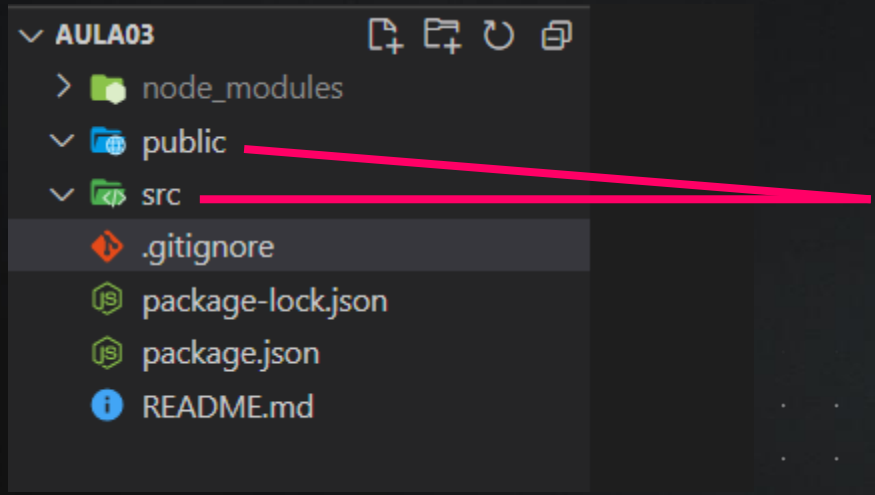
Prof. Alexandre Carlos profalexandre.jesus@fiap.com.br

Prof. Luís Carlos lsilva@fiap.com.br

# Novo projeto c/ pasta src  e public do zero

45697056

Para iniciarmos nossos projeto da aula de hoje, vamos criar um novo projeto chamado react-aula4 e vamos apagar todos os arquivos das pastas src e public.

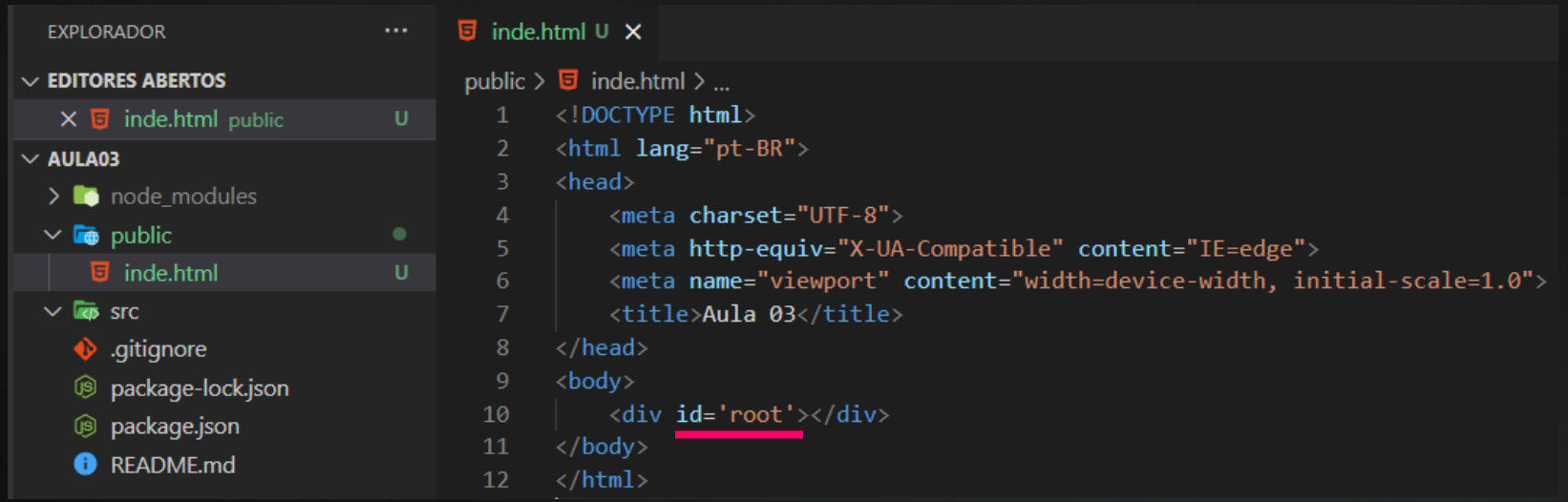



Apagamos todos os arquivos da pasta src e public

# Novo projeto c/ pasta src  e public do zero

React

Na pasta public devemos criar novamente o arquivo index.html, não podemos esquecer de criar uma div com o id= ‘root’.



# Novo projeto c/ pasta src e public do zero

React

O próximo passo será recriar o arquivo index.js na pasta scr. Insira o código abaixo para que os componentes que vamos criar a seguir possam ser renderizados no arquivo index.html.

index.js M

src > index.js

```
1  import React from 'react'
2  import ReactDOM from 'react-dom'
3
4  ReactDOM.render(
5      <h1>Conteúdo de Index.js</h1>,
6      document.getElementById('root')
7  )
8
```

Nos permite usar o JSX

Nos permite usarmos o VDOM

Método para renderizar os componentes na tela

Conteúdo que será renderizado

Local onde será renderizado

Repare em nossa página no navegador, agora só temos a tag h1.

# Novo projeto c/ pasta src e public do zero

React

Por boa prática, vamos criar um novo componente chamado App.js, ele será o nosso componente principal. A princípio vamos apenas colocar um h1 dentro, repita o código abaixo:

index.js M | App.js M

src > App.js > ...

```
1  import React from 'react'
2
3  export default () =>{
4
5      return(
6      <h1>Conteúdo de App.js</h1>
7      )
8  }
```

Podemos usar uma arrow function para deixar o código mais leve.

# Novo projeto c/ pasta src e public do zero

45697056

Para o nosso componente App ser reproduzido na tela agora temos que inserir ele no index.js.

index.js M | App.js M

src > index.js

```
1  import React from 'react'
2  import ReactDOM from 'react-dom'
3  import App from './App'
4
5  ReactDOM.render(<App/>,document.getElementById('root'))
6
```

Repare que agora conseguimos até deixar em apenas uma linha.


# Trabalhando com Classes

45697056

Em react temos duas formas de criar componentes, através de funções como temos usado e também através de classes. Apesar de componentes criados como funções tragam recursos mais atuais, é importante aprendermos a trabalhar com componentes de classes para entendermos melhor o funcionamento do react.

# Trabalhando com Classes

45697056

Na pasta componentes, crie um arquivo chamado PrimeiraClasse.js e insira o código abaixo:

```
import React, { Component } from 'react'

class PrimeiraClasse extends Component{

    render(){
        return(
            <>
              <p>Primeira Classe Funcionando</p>
            </>
        )
    }
}
export default PrimeiraClasse
```

Estende component

É uma classe JS

O return deve ficar dentro do método render

Não se esqueça em chamar no App

# Trabalhando com Classes

Nas classes devemos criar um construtor para trabalhar, em javascript ele chama constructor e como herdamos de componente, devemos chamar o super para passar o props:

```
import React, { Component } from 'react'

class PrimeiraClasse extends Component{

        constructor(props){
            super(props)
        }

    render(){
        return(
            <>
              <p>Primeira Classe Funcionando</p>
              <p>Mensagem: {this.props.msg}</p>
            </>
        )
    }
}
export default PrimeiraClasse
```

Precisamos criar um construtor

Devemos usar o super

Como estamos na classe precisamos usar this

Não se esqueça em chamar no App

# Trabalhando com Classes

45697056

Agora vamos passar uma mensagem para ele:

exemplo > src > App.js > ...

```
1  import React from 'react'
2  import PrimeiraClasse from './componentes/PrimeiraClasse'
3
4
5  export default ()=>{
6
7      return(
8          <>
9          <h1>Trabalhando com classes</h1>
10         <PrimeiraClasse msg='Agora mudou um pouco'/>
11         </>
12     )
13 }
```

O atributo deve ser msg para ficar igual a chamada do props

# Trabalhando com Classes

Vamos fazer mais um exemplo, crie um arquivo chamado Funcionario.js dentro da pasta componentes:

Se a variável estiver dentro do construtor usar this

Não se esqueça em chamar no App

```
45697056
import React, { Component } from "react";

export default class Funcionario extends Component {
  constructor(props) {
    super(props);
    this.nome='João'
  }
   state={
        cargo:'auxiliar',
        turno:'noite'
    }

  render() {
    return (
      <>
        <h2>Funcionário</h2>
        <p>Nome: {this.nome}</p>
        <p>Cargo: {this.state.cargo}</p>
        <p>Turno: {this.state.turno}</p>
      </>
    );
  }
}
```

# Trabalhando com Classes

Vamos utiliazar o setState para ver como ele se comporta:

Repare que devemos chamar diretamente dentro dele

Não se esqueça em chamar no App

```
45697056
export default class Funcionario extends Component {
  constructor(props) {
    super(props);
    this.nome='João'
   }
   state={
        cargo:'auxiliar',
        turno:'Noite'
   }

    mudarTurno(){
        this.setState({turno:(this.state.turno == 'Noite'?'Manhã':'Noite')})
    }

  render() {
    return (
      <>
        <h2>Funcionário</h2>
        <p>Nome: {this.nome}</p>
        <p>Cargo: {this.state.cargo}</p>
        <p>Turno: {this.state.turno}</p>
        <button onClick={()=>this.mudarTurno()}>Adicional</button>
      </>
    ); }}
```


# Trabalhando com Classes

45697056

Outra forma de trabalharmos com o setState é usando uma arrow function internamente, assim garantimos mais a renderização quando ele muda o valor de state:

```
mudarTurno(){
    this.setState(
        (state)=>(
        {turno:(state.turno == 'Noite'?'Manhã':'Noite')})
    )
}
```

# Trabalhando com Classes



Agora vamos usar props e state juntamente, vamos adicionar mais um atributo em nossa classe chamado adicional, uma função para aumentar os adicionais e um botão para chamar. O valor será enviado através de props pelo App:

```
state={
    cargo:'auxiliar',
    turno:'Noite',
    adicional: 0
}
```

```
adicional(){
    this.setState((state,props)=>(
        {adicional:state.adicional + props.add}
    ))
}

render() {
  return (
    <>
      <h2>Funcionário</h2>
      <p>Nome: {this.nome}</p>
      <p>Cargo: {this.state.cargo}</p>
      <p>Turno: {this.state.turno}</p>
      <p>Adicionais:{this.state.adicional}</p>
      <button onClick={()=>this.mudarTurno()}>Mudar Turno</button>
      <button onClick={()=>this.adicional()}>Adicional</button>
    </>
  ); }}
```


# Trabalhando com Classes

Na classe App, vamos passar um valor por atributo, para usarmos na incrementação:

```
return(
    <>
    <h1>Trabalhando com classes</h1>
    <PrimeiraClasse msg='Agora mudou um pouco'/>
    <Funcionario add={1}/>
    </>
)
```


# Exercício

45697056

1 – Crie um novo projeto React chamado exercicio4, limpando todo o conteúdo das pastas public e src e recriando os arquivos index.html, index.js e App.js.

2 – Utilizando classes para criar componentes, crie um componente chamado Aluno.js e nele crie parágrafos para receber nome, curso, turma e rm deste aluno.

3 – Em App, crie 3 componentes alunos e em cada um passe através de props os valores de alunos.

# DUVIDAS