FIAP

GRADUAÇÃO

45697056

TDS

Responsive Web Development

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

Prof. Luís Carlos lsilva@fiap.com.br

45697056

INTRODUÇÃO

React

ArrowFunction x Function

45697056

Antes de mergulharmos de vez no mundo dos componentes reativos. Vamos dar uma pequena olhada neste assunto que na minha opinião é bastante importante.

Existe muito mito em torno das ArrowFunction, então vamos acabar com isso.

1 – Sim ArrowFunction é diferente de Function.

Ex:

```
1   function sum(a, b) {
2       return a + b
3   }
4
5   const sum = (a, b) => {
6       return a + b
7   }
8
```

ArrowFunction x Function

45697056

No exemplo anterior podemos ver claramente que as diferenças se iniciam na declaração.

Equanto a function, necessita da palavra chave function e do nome da função a arrow function apenas levou a declaração de constante, o nome o sinal de atribuição e a seta.

Isso mesmo, declaração, uma ArrowFunction tem que ser declarada como variavel ou constante.

Além das funções padrões que trabalhamos, temos as funções anônimas, aquelas quais, utilizamos para executarem ações direto dentro de blocos de declarações:

Ex:

```
10  const elementos = document.querySelector('p');
11  elementos.addEventListener('click', function(){
12      console.log('Função-Anônima')
13  });
```

ArrowFunction x Function

45697056

No exemplo anterior nós vimos um bloco de declaração clásico com uma chamada de função anônima, vamos ver o mesmo bloco agora com ArrowFunction:

Ex:

```
15  const elementos = document.querySelector('p');
16  elementos.addEventListener('click', () => {
17      console.log('ArrowFunction')
18  });
19
20  //Ou, eliminando as chaves
21
22  const elementos = document.querySelector('p');
23  elementos.addEventListener('click', () => console.log('ArrowFunction') );
24
25  //O que poderia ser feito no primeiro exemplo.
26  const sum = (a, b) => a + b
```

ATENÇAO: Somente podemos eliminar as chaves se tivermos uma única linha de instrução.

REACT

45697056

O React é uma biblioteca Javascript criada pelo Facebook, utilizada para criar interfaces para usuários, com uma particularidade de renderizar somente a parte da tela que é necessária, aumentando assim em muito a performance da página.

Seu foco principal é transformar a experiencia do usuário mais eficiente, tornando a aplicação mais leve e performática, permitindo a reusábilidade de componentes.

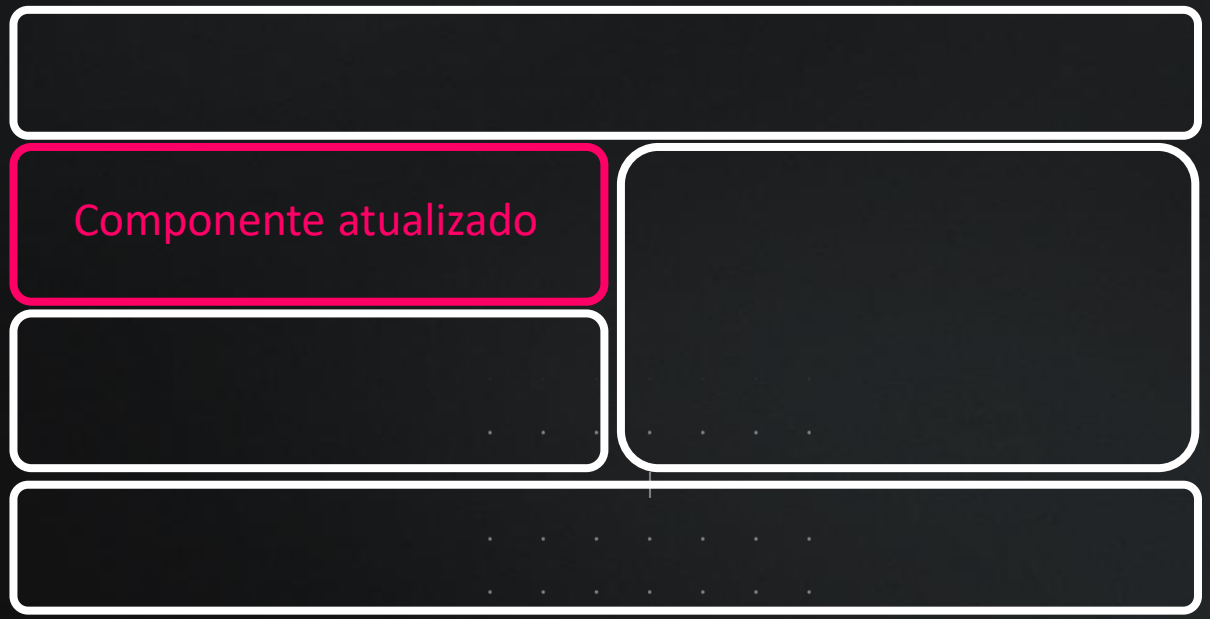
No React tudo é Javascript, até os elementos HTML são criados por ele através do JSX, uma extensão de sintaxe que nos permite trazer a criação dos elementos HTML para dentro do Javascript.

Para que as mudanças dos componentes da tela sejam harmoniosas, ele utiliza o Virtual DOM (VDOM) que gerencia os componentes em memória e sincroniza com o DOM real, utilizando a biblioteca do ReactDOM. Isso aumenta a performance, melhorando até a classificação nos motores de busca.

REACT

45697056

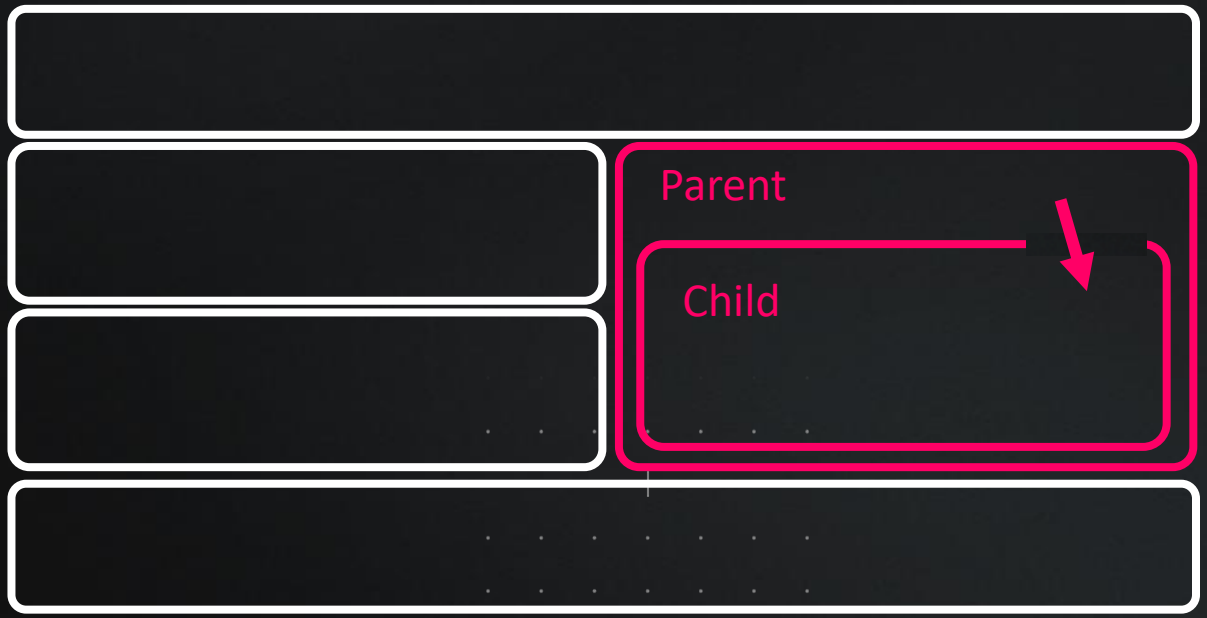
O React, por trabalhar com componentes, só precisa carregar a parte da página que foi alterada, mantendo as demais partes, deixando o trabalho mais rápido.



REACT

45697056

Ele trabalha com um fluxo unidirecional, em um único sentido, ou “One-way data flow”. As informações devem sempre vir do elemento pai para o elemento filho.

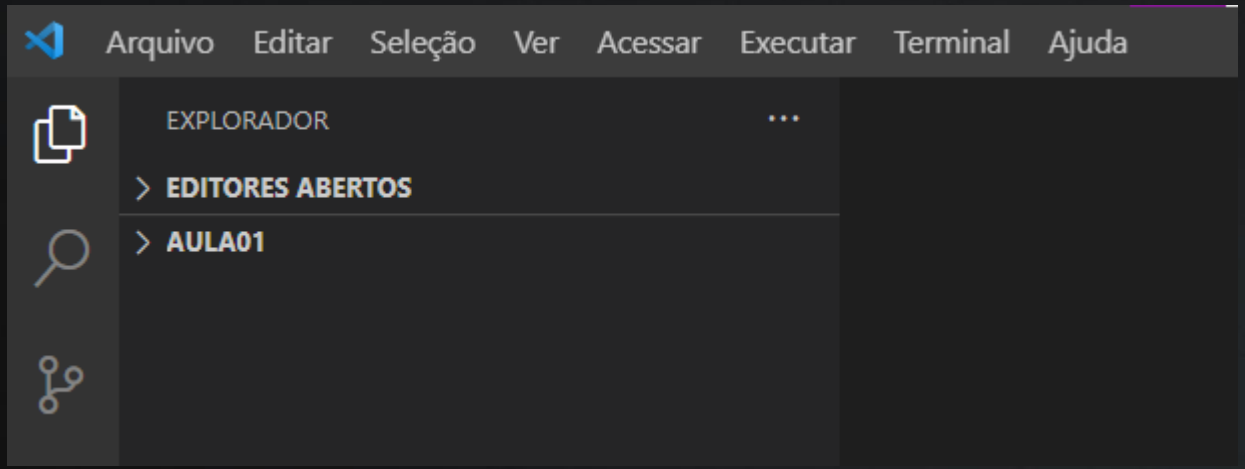


REACT

45697056

Na apostila passada, instalamos o Node.js, vamos usar ele para criar nossas aplicações diretamente do computador, sem estruturas online, ok?

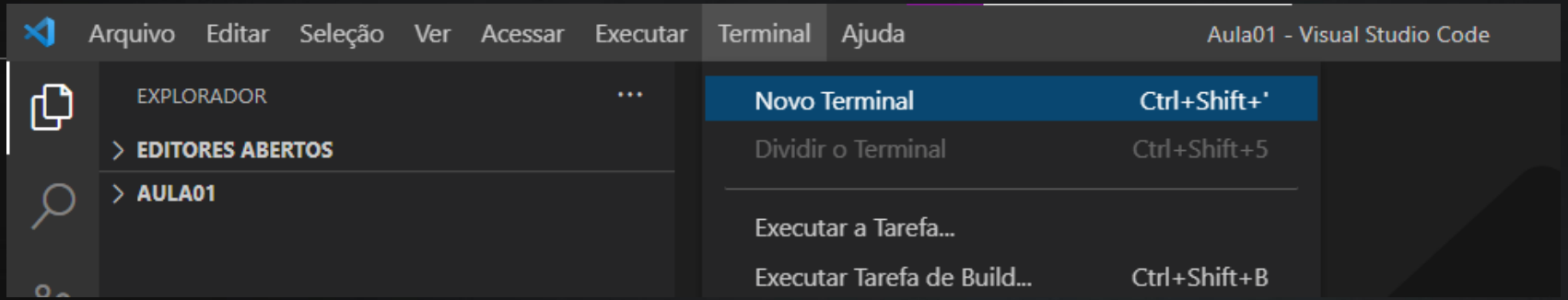
- Vamos abrir o VS Code e criar uma pasta na área de trabalho, chame de Aula01.



REACT

45697056

- Abra o terminal do VS Code, pode usar o atalho: crtl + shift + ‘ ou Menu Terminal -> Novo terminal.



- Obs. Se quiser também pode usar o prompt de comando, basta acessar a nossa pasta Aula01.

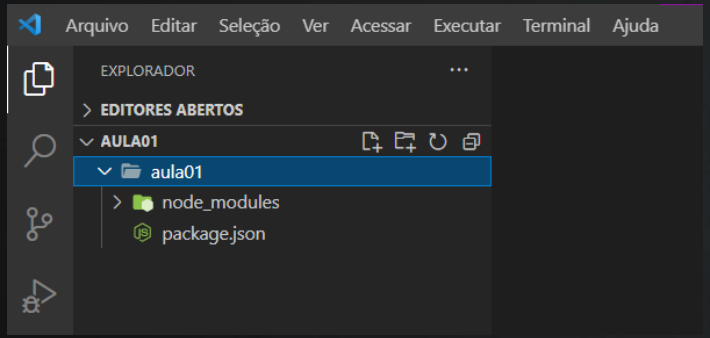
REACT

45697056

- Agora vamos usar o npx, que é um executor de pacotes para criar nossa primeira aplicação. Digite o seguinte comando no prompt do terminal:

```
npx create-react-app aula01
```

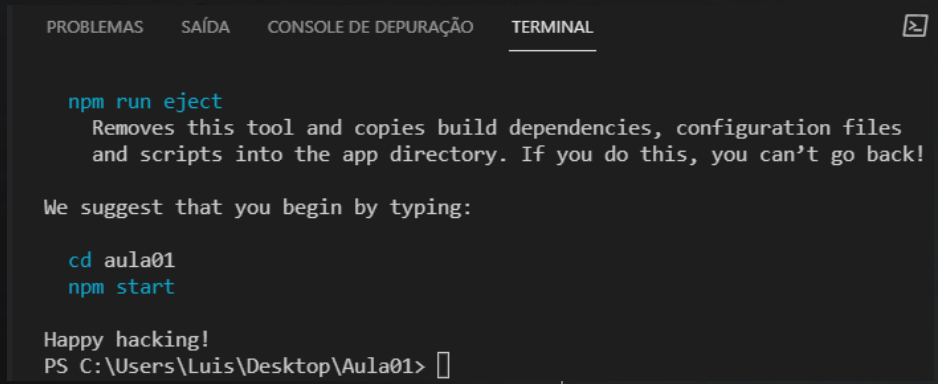
A criação pode demorar alguns minutos, enquanto ele instala, deixe a nossa pasta Aula01 aberta para visualizar a criação dos arquivos em tempo real.



REACT

45697056

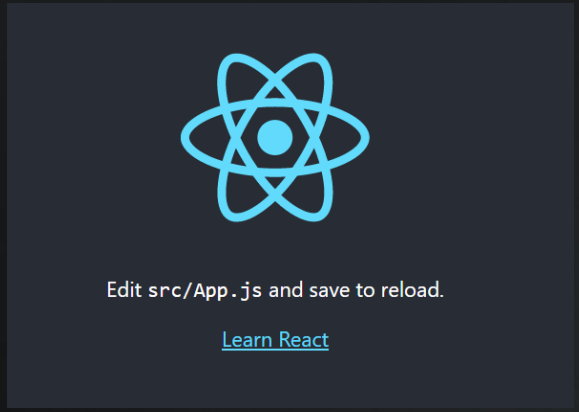
- Ao término da instalação, se for concluída com sucesso, devemos receber uma mensagem sugerindo que iniciemos a aplicação.



REACT

45697056

- Agora vamos subir nossa aplicação, digite cd aula01 [enter] para abrir a pasta e em seguida npm start.



- Esta é a página inicial da nossa aplicação.

REACT

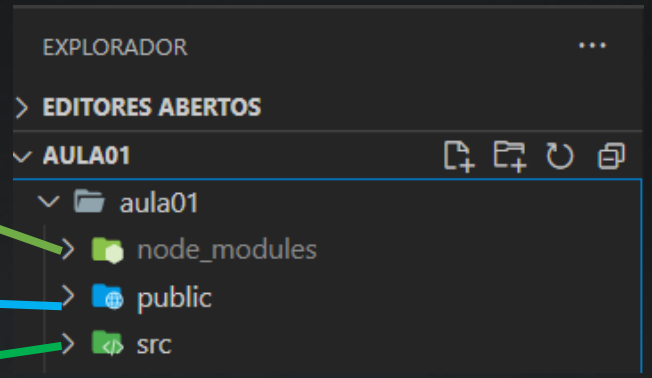
45697056

- Na pasta do nosso projeto temos 3 subpastas que foram criadas com o projeto:

Node Modules: são bibliotecas do Node que estão disponível para uso na aplicação

Public: contém os arquivos que são carregados no lado do cliente

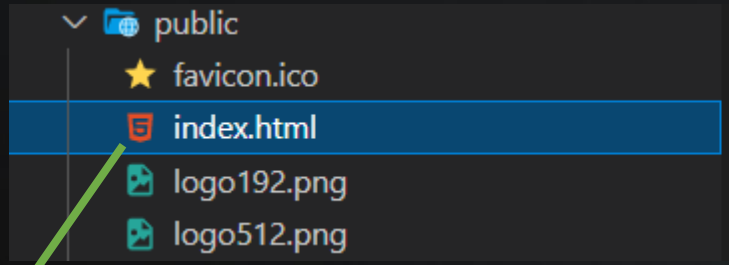
Src: contém os arquivos que estão do lado do servidor



REACT

45697056

- O React é carregado na página index.html, dentro da div com id “root” e é a partir dai que o javascript carrega todos os componentes da aplicação.



Arquivo index.html na pasta public

```
28  </head>
29  <body>
30    <noscript>You need to e
31    <div id="root"></div>
32    <!--
```

Div “root”, onde o javascript carrega toda a aplicação

REACT

45697056

- Já na pasta SRC, o index.js é o responsável por carregar toda a nossa aplicação na div de id “root” que está no index.html. Repare que ele está carregando o app.js nele.

```
6
7   ReactDOM.render(
8     <React.StrictMode>
9       <App />
10    </React.StrictMode>,
11    document.getElementById('root')
12  );
```

REACT

45697056

- O arquivo app.js é nosso arquivo que está dando origem ao componente principal da aplicação, repare que todos os componentes da tela estão sendo criados nele dentro de uma função que recebe o seu nome. Esta função retorna todos os elementos da tela que serão criados no html.
- Isso é possível por causa do jsx.

```jsx
import logo from './logo.svg';
import './App.css';

function App() {
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <p>
          Edit <code>src/App.js</code> and save to reload.
        </p>
        <a
          className="App-link"
          href="https://reactjs.org"
          target="_blank"
          rel="noopener noreferrer"
        >
          Learn React
        </a>
      </header>
    </div>
  );
}

export default App;
```

REACT

45697056

Vamos limpar o conteúdo do arquivo app.js e vamos criar um novo conteúdo do zero.

```
App.js M
aula01 > src > JS App.js > ...
1  import React from 'react' //importação da biblioteca do React para o arquivo
2
3  function App(){            //função com o nome do arquivo
4
5    return(                  //a função deve retornar o conteúdo que será exibido
6      <>
7        <h1>"Olá Mundo!"</h1>
8      </>
9    )
10 }
11
12 export default App;   // instrução para chamar a função
13
14
```

```
<html lang="en">
▶ <head>…</head>
▼ <body>
    <noscript>You need to enable JavaScript to run
    this app.</noscript>
••• ▼ <div id="root"> == $0
      <h1>"Olá Mundo!"</h1>
    </div>
    <!--
          This HTML file is a template.
          If you open it directly in the browser, you
    will see an empty page.
```

localhost:3000

"Olá Mundo!"

REACT

45697056

Podemos ter tantos componentes quanto necessário, mas todos eles devem estar dentro de um componente principal, senão teremos erro no código. Este elemento principal pode ser uma tag como uma DIV, por exemplo, ou uma tag vazia simplesmente para marcar o conteúdo, como temos abaixo:

```
import React from 'react'

export default function App(){

  return(
    <>
      <h1>TDS FIAP - RWD </h1>
      <a href="#">FIAP</a>
      <br/>
      <a href="#">GOOGLE</a>
    </>
  )
}
```

Podemos fazer a exportação da função na mesma linha

Tags self close precisam de barra de fechamento “obrigatório”

REACT

45697056

Todas as expressões javascript realizadas dentro do jsx devem ser feitas dentro de chaves “{ }”, para serem aceitas

```jsx
import React from 'react'

export default function App(){

  const aluno = 'Matheus Ramalho'
  const curso = 'ADS'

  return(
    <>
      <h1>TDS FIAP - RWD </h1>
      <p>Alunos: {aluno}</p>
      <p>Curso: {curso}</p>
    </>
  )
}
```

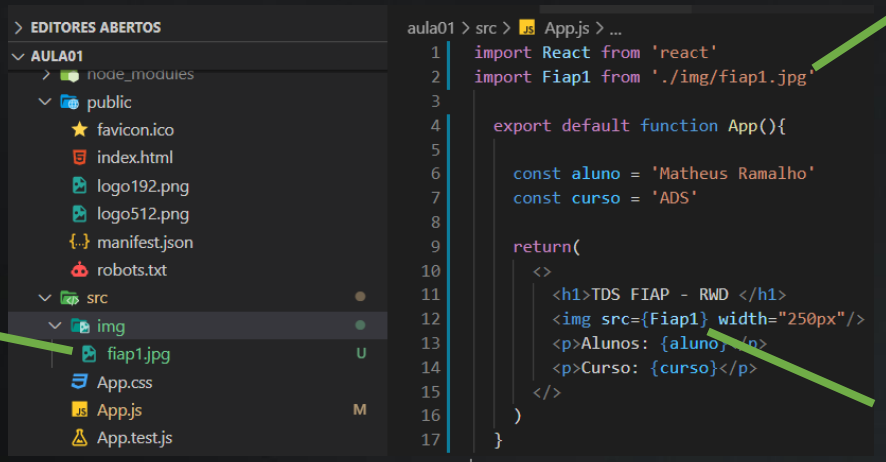
Código javascript da função antes do return

Valores expressos pelas constantes entre chaves

REACT

45697056

Para inserir elementos como imagens devemos importar antes de inserir no componente, lembrando sempre se usar as chaves:



Import da imagem

Imagem salva na pasta src/img

Inserindo a imagem na tag img

OBS. Se a imagem estiver na pasta public, o mapeamento será do modo convencional

REACT

45697056

Como falamos a pouco, o React trabalha com componentes, então vamos criar alguns para ver como funciona. Dentro da pasta src crie uma pasta chamada componentes e dentro dela um arquivo chamado ListaAluno.js. Nela insira o seguinte código:

```
import React from 'react'

export default function ListaAlunos(){

    return(
        <ul>
            <li>Huguinho</li>
            <li>Zézinho</li>
            <li>Luizinho</li>
        </ul>
    )
}
```

REACT

45697056

Agora para conseguirmos utilizar o arquivo ListaAlunos.js como um componente devemos importa-lo dentro de App.js e inserir ele no componente principal, conforme abaixo:

```
import React from 'react'
import Fiap1 from './img/fiap1.jpg'
import Lista from './compomentes/ListaAlunos'

export default function App(){

  const aluno = 'Matheus Ramalho'
  const curso = 'ADS'

  return(
    <>
      <h1>TDS FIAP - RWD </h1>
      <img src={Fiap1} width="250px"/>
      <p>Alunos: {aluno}</p>
      <p>Curso: {curso}</p>
      <Lista/>
    </>
  )
}
```

Importando o arquivo ListaAlunos.js

Inserindo no componente principal, criando uma tag com seu nome

Exercício

45697056

1. Crie uma nova aplicação chamada exercício;
2. Limpe o conteúdo do arquivo App.js
3. Crie um componente chamado Cabecalho.js e insira uma tag header com um h1 e um parágrafo;
4. Crie um componente chamado Carros.js e insira uma imagem de carro e uma lista com 5 modelos de carro.
5. Crie um componente chamado Parceiros.js e insira um h2 e 4 links.

REACT

45697056

Como comentamos no início os dados, se necessário, devem fluir do pai para seus filhos, este é o fluxo natural. Para conseguirmos passar estes dados vamos utilizar ‘props’. Ele é passado do pai para seus filhos como um objeto e dentro do filho podemos acessar seus valores.

Para enviar os valores para o filho devemos criar atributos em sua tag que está no componente pai. Vamos enviar os nomes dos alunos daquela lista que criamos a pouco:

```
export default function App(){

  const aluno = 'Matheus Ramalho'
  const curso = 'ADS'
  const alunos = ['Luís','Alexandre','Allen']

  return(
    <>
      <h1>TDS FIAP - RWD </h1>
      <img src={Fiap1} width="250px"/>
      <p>Alunos: {aluno}</p>
      <p>Curso: {curso}</p>
      <Lista alunos={alunos} />
    </>
  )
}
```

Dados contidos no pai

Enviando os dados para o filho

REACT

45697056

Para o filho poder utilizar os dados enviados pelo pai devemos colocar o parâmetro props na função ListaAlunos e, como um objeto, chamar seus valores.

```jsx
import React from 'react'

export default function ListaAlunos(props){

    return(
        <ul>
            <li>{props.alunos[0]}</li>
            <li>{props.alunos[1]}</li>
            <li>{props.alunos[2]}</li>
        </ul>
    )
}
```

Parâmetro props

Utilização dos valores enviados pelo pai

REACT

45697056

Assim como passamos dados do pai para o filho, também podemos passar funções, vamos enviar uma função para ver as diferenças. Crie a função no componente pai conforme abaixo:

```
export default function App(){

  const aluno = 'Matheus Ramalho'
  const curso = 'ADS'
  const alunos = ['Luís','Alexandre','Allen']

  const novoAluno = () => 'Fernanda'

  return(
    <>
      <h1>TDS FIAP - RWD </h1>
      <img src={Fiap1} width="250px"/>
      <p>Alunos: {aluno}</p>
      <p>Curso: {curso}</p>
      <Lista alunos={alunos} maisAluno={novoAluno}/>
    </>
  )
}
```

Criação de uma função

Enviando a função para o filho

REACT

45697056

Para a utilização da função herdada do pai só não devemos esquecer dos parênteses, o resto é igual as variáveis:

```
import React from 'react'

export default function ListaAlunos(props){

    return(
        <ul>
            <li>{props.alunos[0]}</li>
            <li>{props.alunos[1]}</li>
            <li>{props.alunos[2]}</li>
            <li>{props.maisAluno()}</li>
        </ul>
    )
}
```

Utilizando a função

Exercício

45697056

1. Volte para a aplicação exercício que criamos a pouco;
2. Passe os modelos dos carros do componente pai para o componente filho.
3. Crie uma função multiplicando dois números no componente pai e passe para o componente Parceiros, executando em uma span logo abaixo do h2;

DUVIDAS