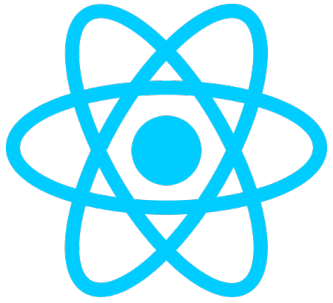
INSTITUTO FEDERAL
Sul-rio-grandense | Câmpus Pelotas

tsi tecnologia em sistemas para internet

DESENVOLVIMENTO FRONT-END II

Introdução ao ReactJS, Principais Conceitos, Propriedades e Estados



Prof. Gill Velleda Gonzales | gillgonzales@ifsul.edu.br | @g1ll
CSTSI - Março - 2025

Desenvolvimento Front-End II

Tópicos

- Conceitos Principais:
- Biblioteca vs *Framework*
- Imperativo vs Declarativo
- Fluxo de Dados
- Arquitetura de Componentes

Biblioteca vs *Framework*

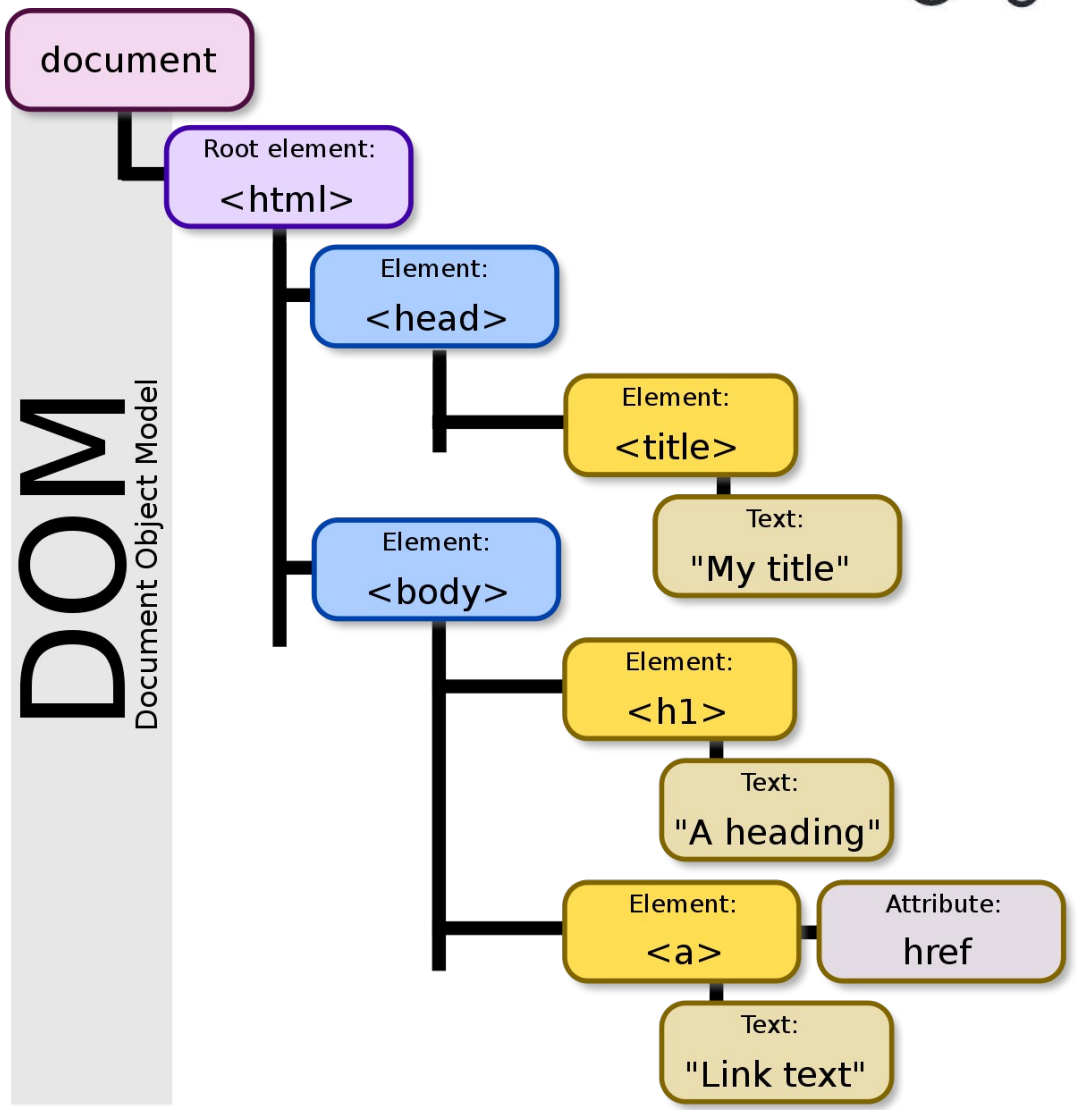
CONCEITOS PRINCIPAIS

- Conforme a própria documentação, ***ReactJS*** é uma **BIBLIOTECA**.
- Mas existem *frameworks* que trazem diversas vantagem e velocidade no desenvolvimento com ***ReactJS***.
- Dentre eles, o mais em voga é o ***NextJs***, da Vercel, para a criação de aplicações web.
- No entanto existem outros, baseados em criação de UI, como o Chacra UI, Tailwind UI, e outros.
- Nesta disciplina focaremos no ***ReactJS***.
- O Objetivo é construir uma base sólida e prepará-los para outros conhecimentos que envolva o ecossistema React.
- Lista de *Frameworks* para React: ***Link***.

Imperativo versos declarativo

CONCEITOS PRINCIPAIS

- **Imperativo:**
  - Alterações **diretas** no ***DOM*** via Javascript de acordo com os eventos que ocorriam durante a interação do usuário.
  - Bibliotecas antigas como ***Jquery*** e até mesmo aquelas voltadas para SPAs (**S**ingle **P**age **A**pplication), como o ***Angular***, permitiam, de formas diferentes, a manipulação direta da árvore de elementos da página, **DOM**.



Fonte: https://blog.codewise.fr/javascript-manipulation-dom

Imperativo versos declarativo

CONCEITOS PRINCIPAIS

- **Imperativo:**
  - Manipulação direta de elementos de **DOM (API DOM)**.

    document.querySelector(‘#container’)

    - ***Link: Exemplo vanilla***
  - Bibliotecas:
    - ***jQuery*** é um exemplo clássico:

      $(‘#container’)

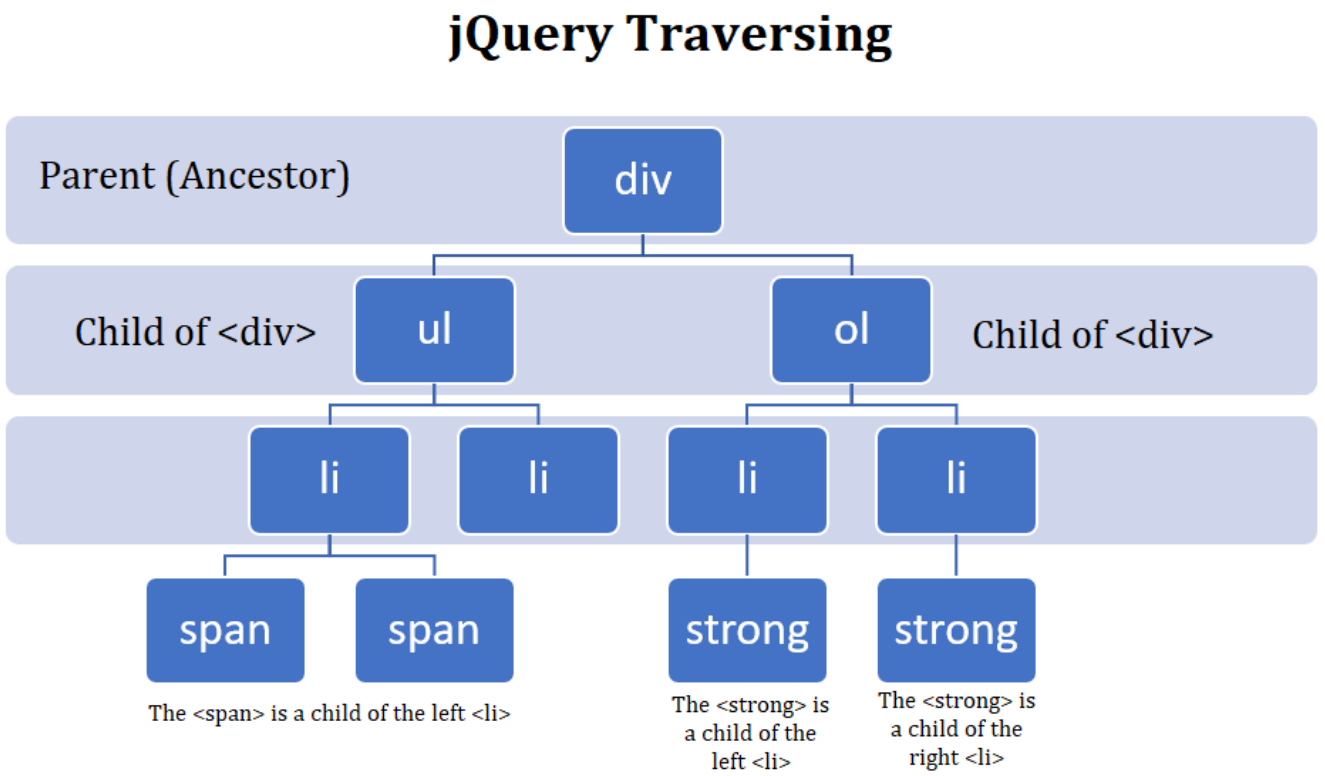
    - ***Link: Exemplo jquery***

```
(function($) {
  $(function(){
    $("button").click(function(e) {
      $.get("/test.json", function(data) {
        $(".list").each(function() {
          $(this).click(function(e) {
            setTimeout(function() {
              alert("Hello World!");
            }, 1000);
          });
        });
      });
    });
  });
})(jQuery);
```

Fonte: https://www.slideshare.net/slideshow/js-for-multidisciplinary-teams/28708948#35

Imperativo versos declarativo

- **Imperativo:**
  - Em aplicação grandes, as ***relações*** entre ***eventos*** e manipulação de **elementos** se tornavam cada vez mais ***complexas***.



Fonte: https://studyopedia.com/jquery/introduction-traversing/

Imperativo versos declarativo

- **Declarativo:**
  - O time do facebook então propôs o **ReactJS**, a primeira biblioteca a ***evitar*** a ***manipulação direta*** do **DOM**.
  - O ***estado*** da aplicação é representado por um Objeto (***Componente***), o **ReactJS** ***atualiza o DOM***, para representar um novo ***estado***.
  - E faz isso através do ***Virtual DOM***, uma cópia do DOM em memória.
  - Nosso código é ***apenas uma DECLARAÇÃO*** de como deverá ser apresentada a aplicação, ou melhor, ***RENDERIZADA***.
  - A estrutura de ***Componentes*** é a base do **ReactJS**.

Imperativo versos declarativo

CONCEITOS PRINCIPAIS

- **Mas o que é um componente?**
  - Tudo em ***ReactJS*** é um **componente**, ou uma parte ***reutilizável*** e modular da página**.**
  - Os elementos principais da página são ***quebrados*** em ***elementos menores*** que se relacionam, também conhecido como ***Atomic Design***.
  - A aplicação é como um grande ***quebra-cabeça (Lego)***, construída por componentes, agrupados para formar estruturas maiores (***Páginas***).
  - Atualmente componentes são definidos por uma função (***function***), mas no início eram utilizadas classes (**class**).
  - Os componentes retornam o ***html*** em formato ***JSX***: ***App.jsx***
    - Javascrit com XML que suporta escrever ***<tags/>*** no código.

Imperativo versos declarativo

CONCEITOS PRINCIPAIS

- **Exemplo de componente e a forma declarativa:**
  - O componente ***Profile*** declara como os elementos são apresentados e com as informações do objeto ***user***.
  - O código ***JSX***, retornado pelo componente, é então ***renderizado***.
  - ***Links:***
    - ***Docs.***
    - ***Exemplo.***

App.js

```
1  const user = {
2    name: 'Hedy Lamarr',
3    imageUrl: 'https://i.imgur.com/yXOvdOSs.jpg',
4    imageSize: 90,
5  };
6
7  export default function Profile() {
8    return (
9      <>
10       <h1>{user.name}</h1>
11       <img
12         className="avatar"
13         src={user.imageUrl}
```

Imperativo versos declarativo

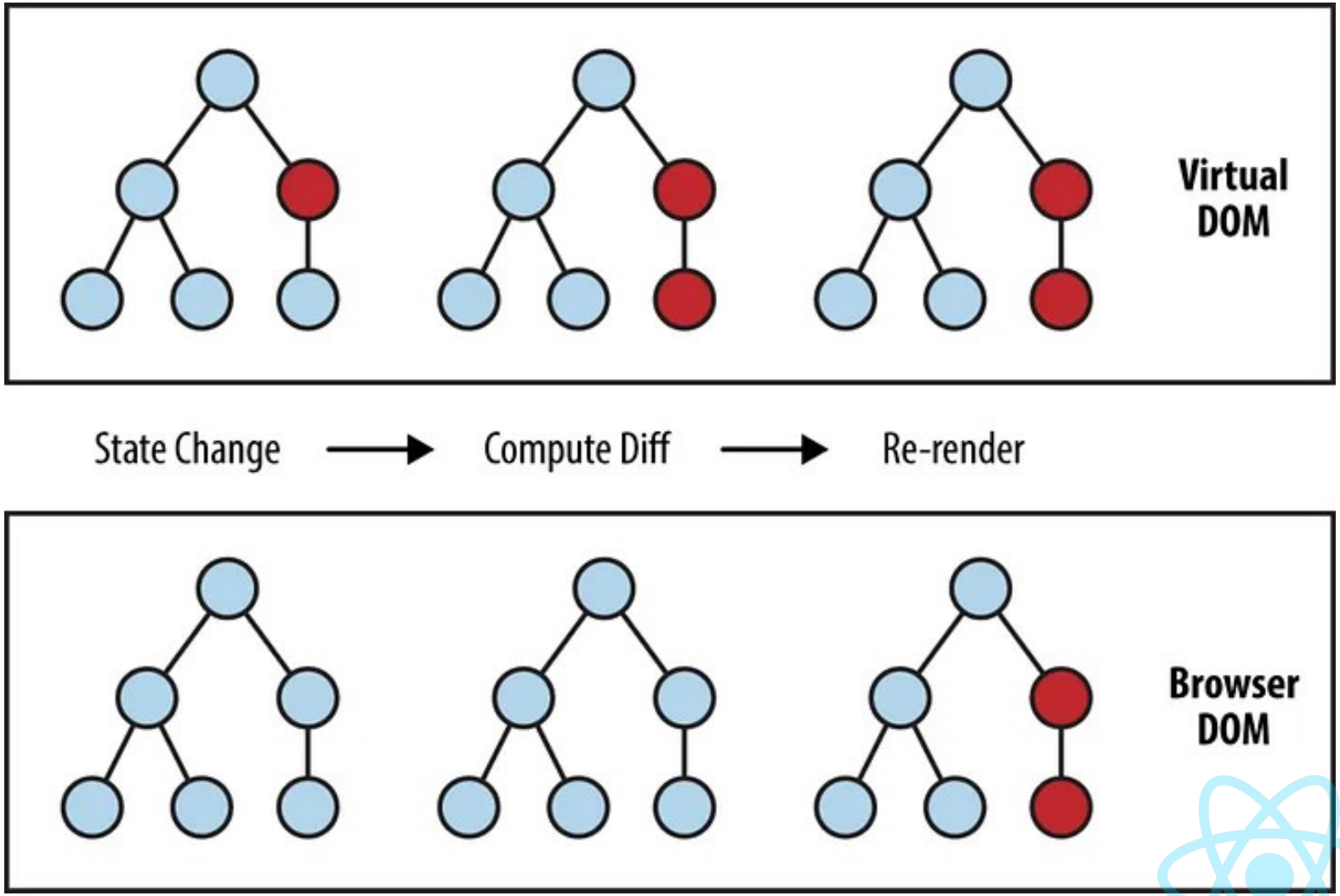
CONCEITOS PRINCIPAIS

- **Regras da declaração via *JSX*:**
  - **1 –** Retorna sempre um ***único elemento*** raiz (**root**).
    - Pode ser usado um fragmento (**fragment**) **<></>** envolvendo o código.
  - **2 –** Todas as tags deverão ser fechadas, **<*img*>** ficará **<*img* />.**
  - **3 –** Atributos do HTML em ***camelCase***, como ***className*** por exemplo.
- Se ficar em dúvida, utilize um conversor de HTML para JSX:
  - ***transform.tools/html-to-jsx***
- ***Exercício 01:***
  - *Corrija o JSX do componente **Bio** deste **link!***

Imperativo versos declarativo

CONCEITOS PRINCIPAIS

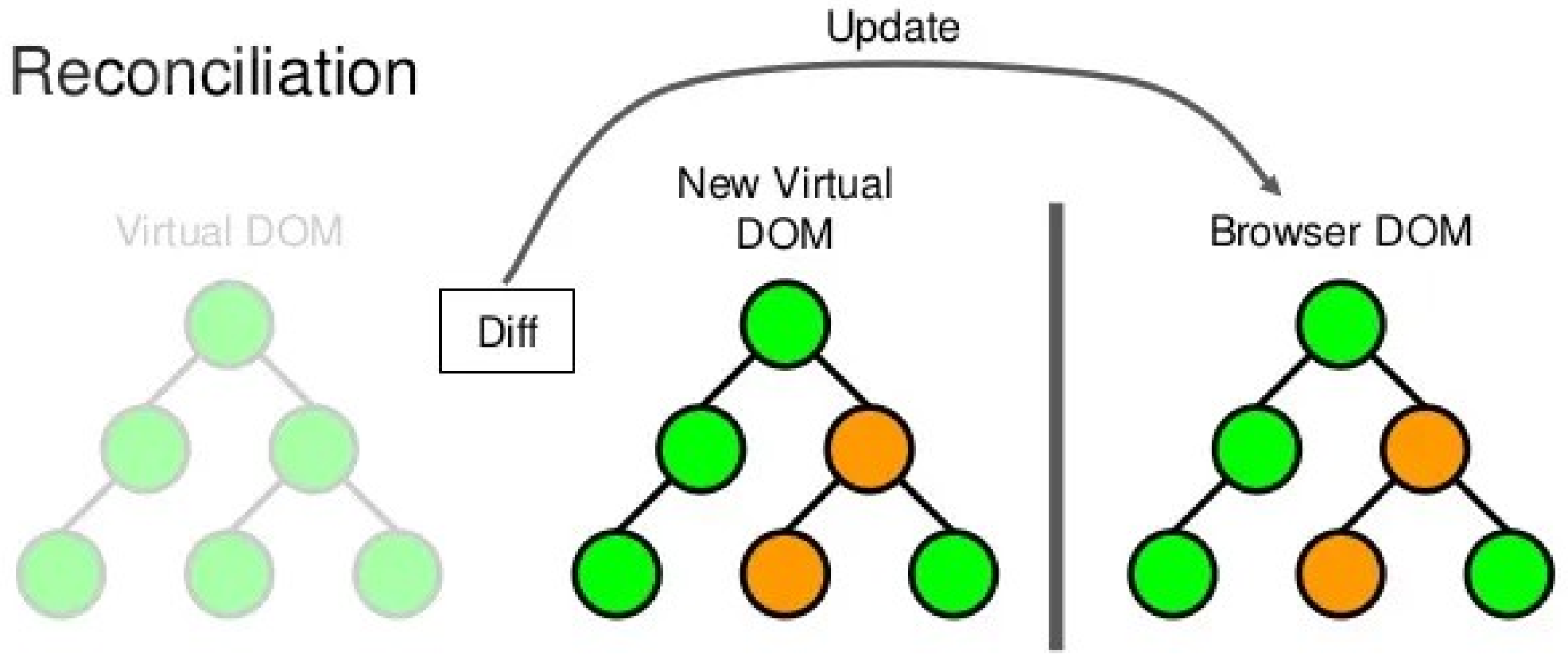
- **Declarativo:**
  - O ***Virtual DOM*** é o que mantém os ***estados*** dos componentes da aplicação.
  - Quando o ***estado*** do componente é ***alterado***, o ***ReactJS*** atualiza o ***DOM Real***, ou seja, no ***navegador***.



Fonte: https://javascript.plainenglish.io/react-the-virtual-dom-comprehensive-guide-acd19c5e327a

Imperativo versos declarativo

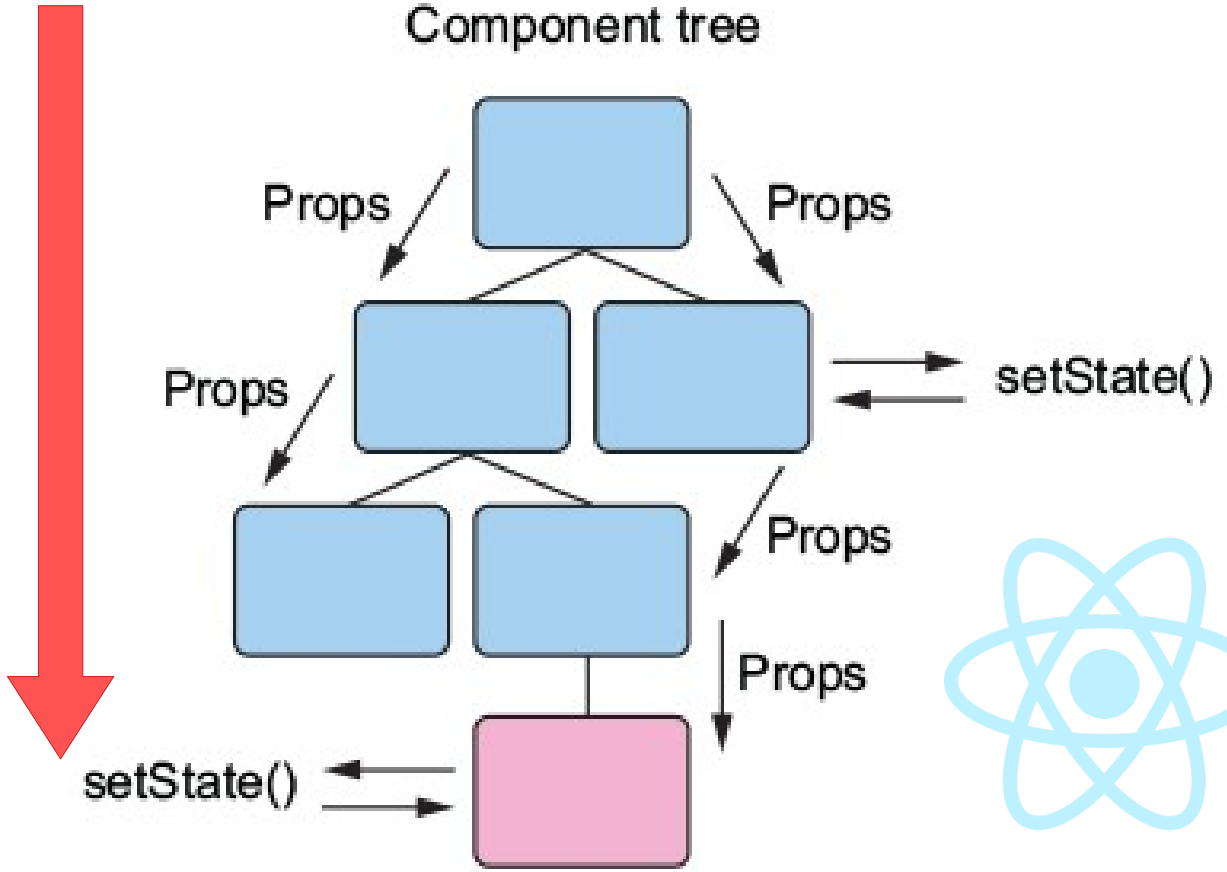
- **Reconciliação entre Virtual DOM e Browser DOM**
  - O ***ReactJS*** garante que o seu algoritmo de reconciliação é eficiente.



Fluxo de Dados

CONCEITOS PRINCIPAIS

- **Propriedades e Estados:**
- Os Componentes possuem propriedades e estados
  - **Propriedades** (***props***): são recebidas e/ou repassadas entre os componentes.
  - As propriedades não podem ser modificadas, são ***constantes***.
  - ***Inicializam*** características de um componente, ou também representam dados a serem mostrados.

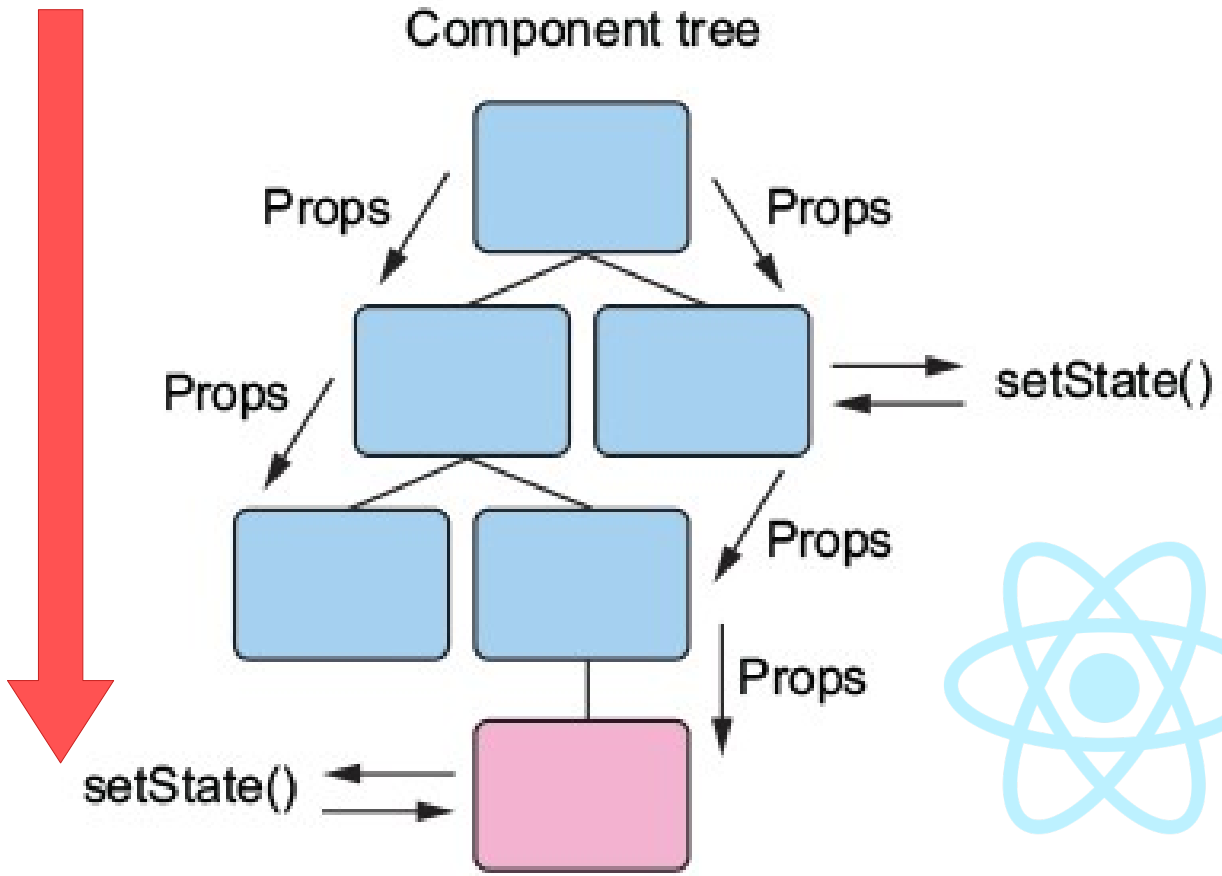


Fonte: https://livebook.manning.com/book/react-in-action/chapter-3/108

Fluxo de Dados

CONCEITOS PRINCIPAIS

- **Propriedades e Estados:**
  - **Estados** (**states**): variáveis do próprio componente, que podem ser modificadas.
  - Quando alterados os estados (***useState***), o componente é obrigatoriamente *atualizado* na tela, ***renderizado***.
  - O ***fluxo de dados*** é sempre **unidirecional** (***one-way***) do componente de maior nível na hierarquia até o menor nível.



Fonte: https://livebook.manning.com/book/react-in-action/chapter-3/108

Fluxo de Dados

CONCEITOS PRINCIPAIS

- **Exemplo:**
  - Veremos o exemplo do componente ***CardImc*** recebendo como propriedades os dados de altura e peso de uma pessoa e mostrando o seu ***Imc***.
  - Os dados estão na propriedade pessoa e são repassados por quem ***"chamou"*** o componente.

CardImc.jsx

```jsx
import './style.css';

export default function CardImc(props) {
  const peso = props.pessoa.peso;
  const alt = props.pessoa.altura;

  const calcImc = () => (peso / alt ** 2).toFixed(2);

  return (
    <div className="imcCard">
      <h1>{props.pessoa.name}:</h1>
      <p>Altura: {alt} m</p>
      <p>Peso: {peso}</p>
      <p>Imc: {calcImc}</p>
    </div>
  );
}
```

Fluxo de Dados

CONCEITOS PRINCIPAIS

- **Primeiro Componente:**
  - Então, as propriedades são passadas pelo componente pai, ***App.jsx***.

- ***Exemplo***

- ***key ?***
  Id do componente.

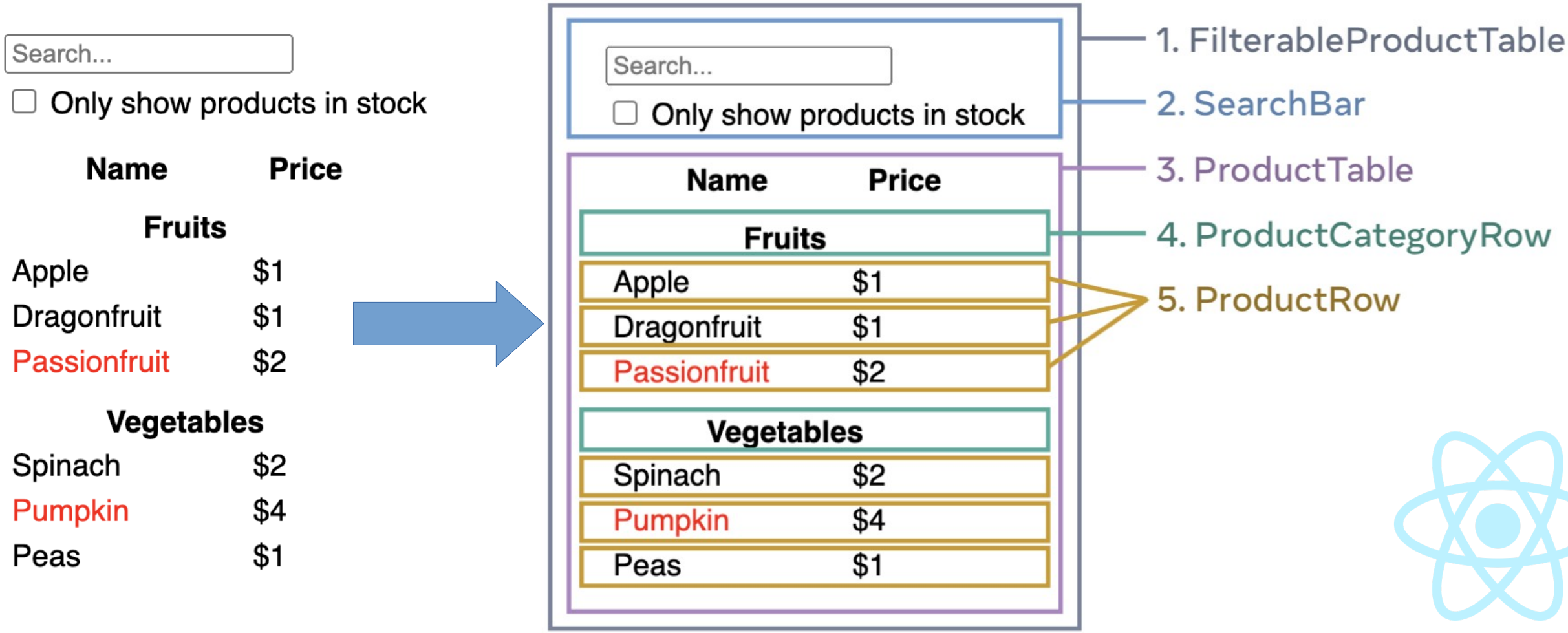
App.jsx | CardImc.jsx

```jsx
import './App.css';
import CardImc from './components/CardImc/CardImc.jsx';

function App() {
  return (
    <>
      <CardImc key="p1" pessoa={{ name: 'Fulano', altura: 1.7, peso: 90 }} />
      <CardImc key="p2" pessoa={{ name: 'Beltrano', altura: 1.8, peso: 75 }} />
      <CardImc key="p3" pessoa={{ name: 'Cicrano', altura: 1.9, peso: 90 }} />
    </>
  );
}

export default App;
```

Arquitetura de Componentes

CONCEITOS PRINCIPAIS

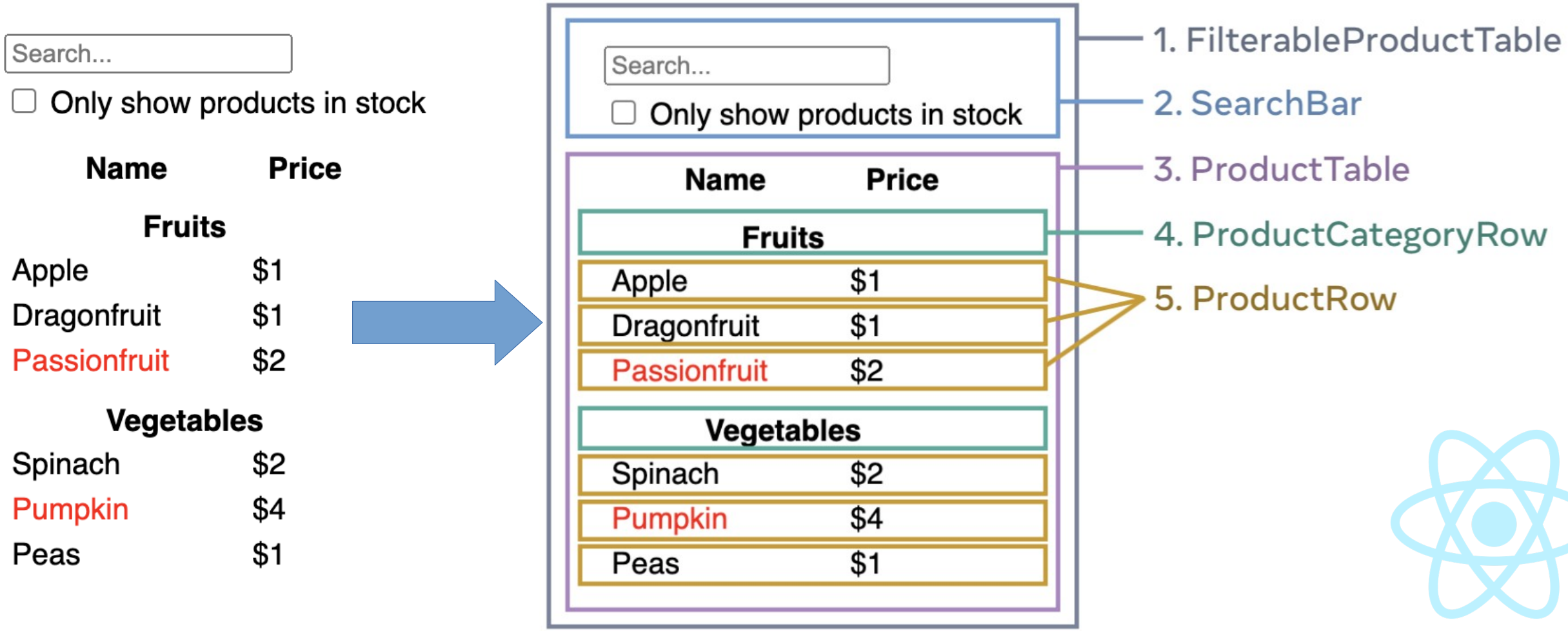
- **Hierarquia de componentes**
  - Divisão da UI em uma arquitetura de componentes de forma hierárquica



Arquitetura de Componentes

CONCEITOS PRINCIPAIS

- **Hierarquia de componentes**
  - Nomes dos componentes seguem o padrão ***PascalCase***



Arquitetura de Componentes

CONCEITOS PRINCIPAIS

- **Passagem de propriedades**
  - Nomear as **propriedades** e repassá-las aos componentes diretamente.
  - **FilterableProductTable** receberá um ***array*** de ***objetos*** ***products***.

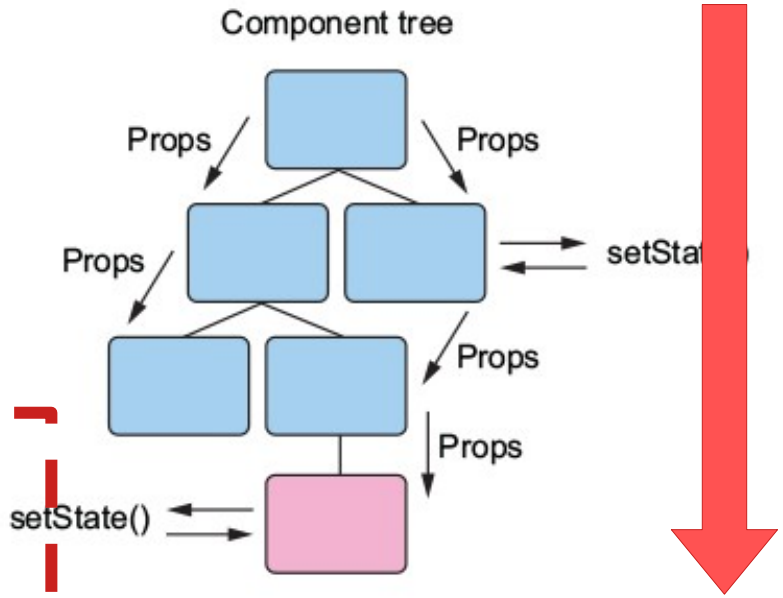
```
const PRODUCTS = [
  { category: 'Fruits', price: '$1', stocked: true, name: 'Apple' },
  { category: 'Fruits', price: '$1', stocked: true, name: 'Dragonfruit' },
  { category: 'Fruits', price: '$2', stocked: false, name: 'Passionfruit' },
  { category: 'Vegetables', price: '$2', stocked: true, name: 'Spinach' },
  { category: 'Vegetables', price: '$4', stocked: false, name: 'Pumpkin' },
  { category: 'Vegetables', price: '$1', stocked: true, name: 'Peas' },
];

export default function App() {
  return <FilterableProductTable products={PRODUCTS} />;
}
```

Arquitetura de Componentes

CONCEITOS PRINCIPAIS

- **Passagem de propriedades**
  - A propriedade ***products*** é então repassada para o componente ***ProductTable***.

```
function FilterableProductTable({ products }) {
  return (
    <div>
      <SearchBar />
      <ProductTable products={products} />
    </div>
  );
}
```



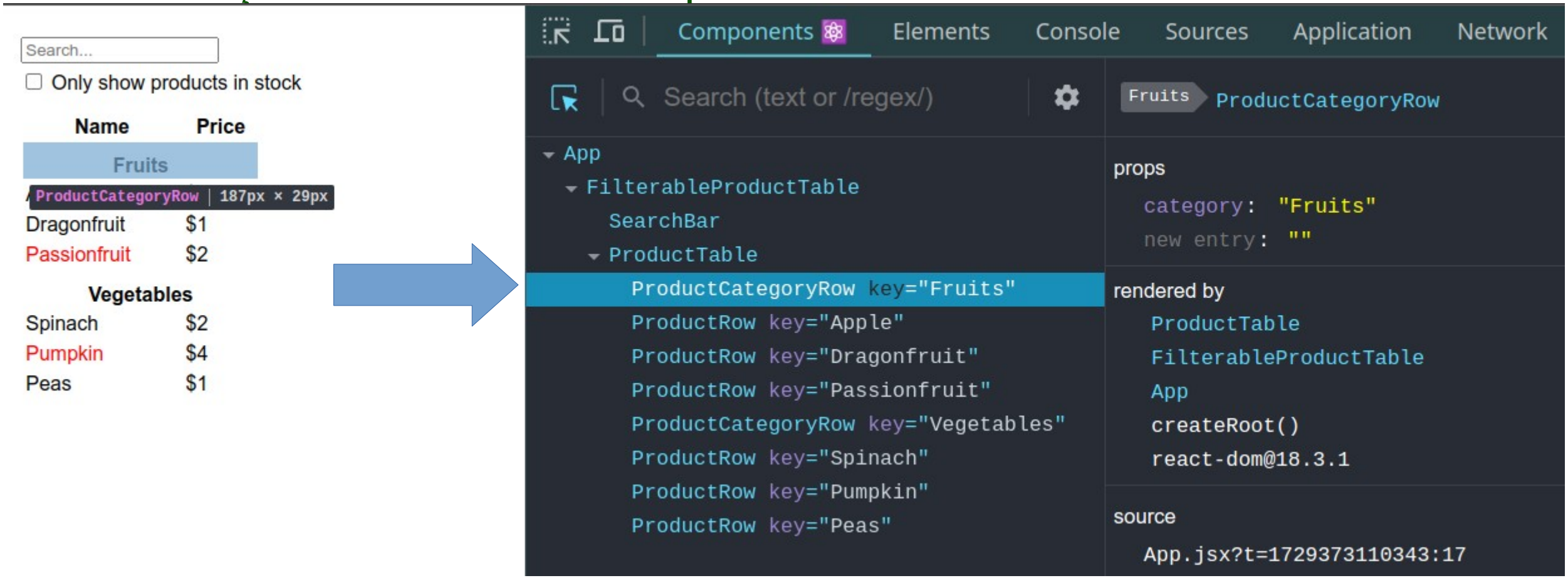
  - Props **desestruturadas** enviadas pela ***App.js***.

```
24  function ProductTable({ products }) {
25    const rows = [];
26    let lastCategory = null;
27
```

Arquitetura de Componentes

CONCEITOS PRINCIPAIS

- **Hierarquia de componentes**
  - Visualização da árvore de componentes na extensão ***React DevTools***:

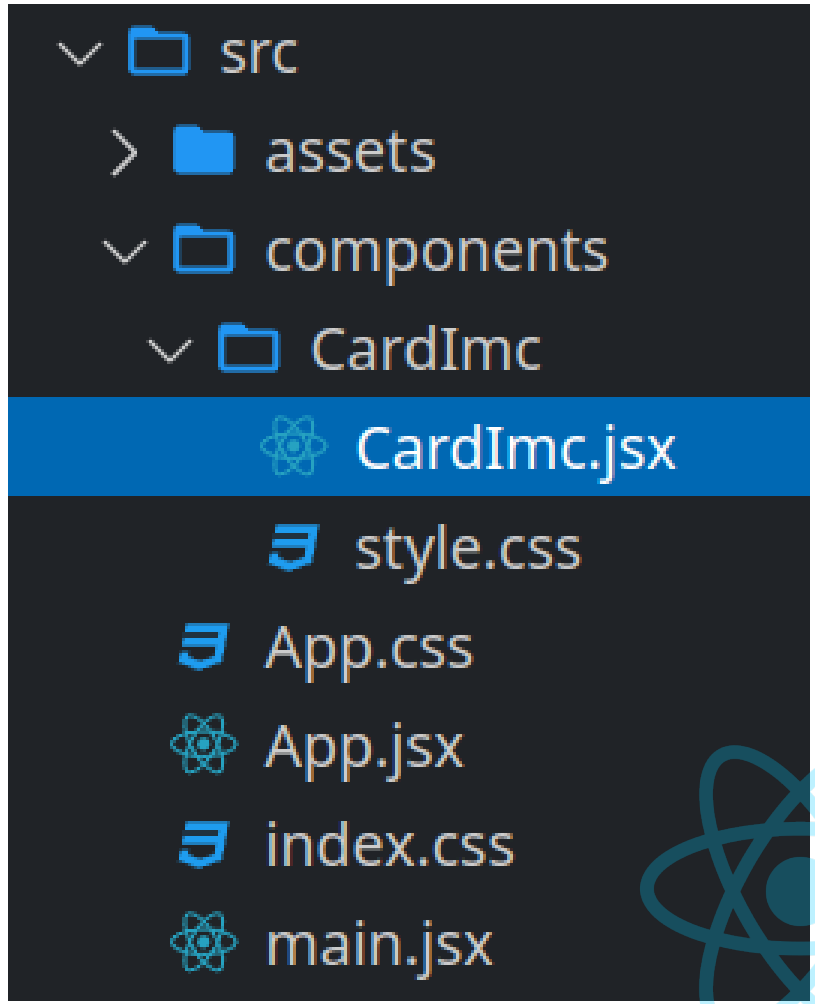


  - **Links: *React DevTools***

Arquitetura de Componentes

CONCEITOS PRINCIPAIS

- **Exercício 02:**
  - Altere o exemplo anterior para que a arquitetura da aplicação se pareça com o padrão do exemplo do ***CardImc***.
  - Ainda seguindo o exemplo do ***CardImc***, crie estilos para cada componente, utilizando ***css* puro** e classes.
  - No componente ***ProductTable***, substitua o método ***forEach*** por ***map***, faça as adaptações necessárias.
  - ***Exemplo de código***



Arquitetura de Componentes

CONCEITOS PRINCIPAIS

- **Exercício 02:**
  - Use o vite para criar o seu projeto.
  - Escolha o template ***react***.
  - ***Exemplo de código***

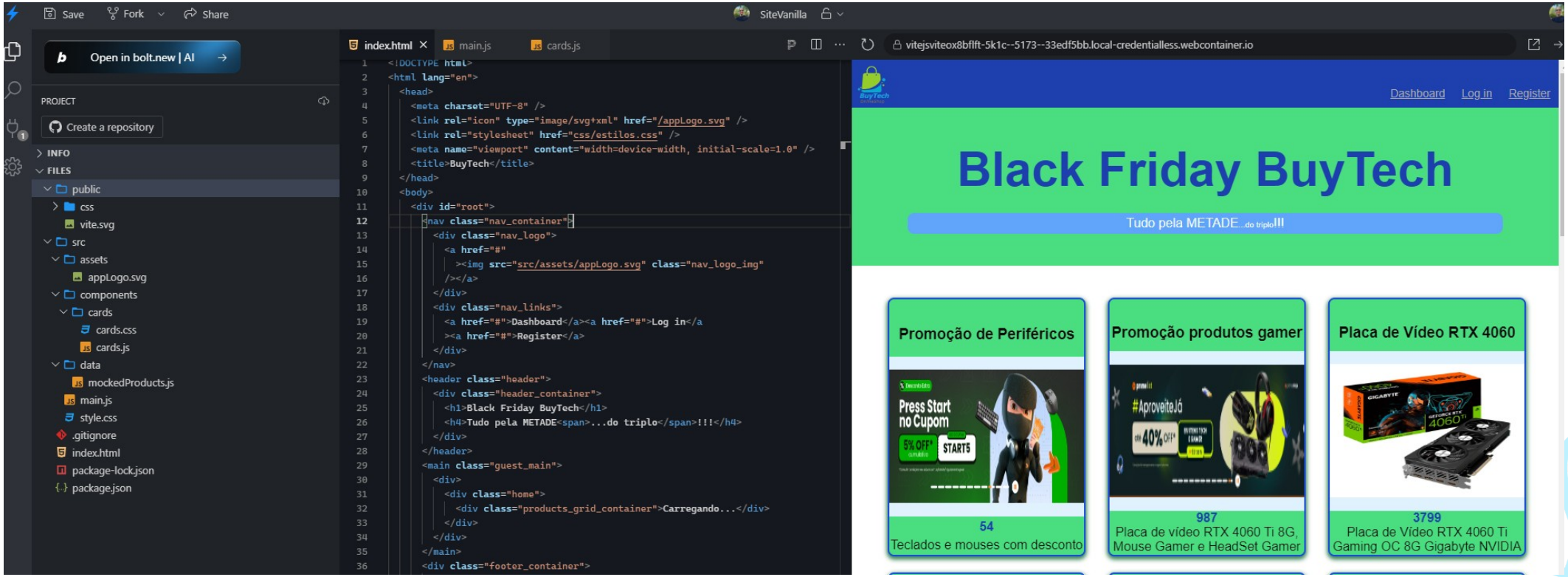
```
/media/Programas/Projetos/ifsul/cstsi/dfe2/prototipos                5s
❯ npm create vite@latest exemplo01Aula02 -- --template react

> npx
> create-vite exemplo01Aula02 --template react

✔ Package name: … exemplo01aula02
```

  - ***Comando:***
    - **$>npm create vite@latest atividade01Aula02 -- --template react**

Arquitetura de Componentes

CONCEITOS PRINCIPAIS

- **Exercício 03:**
  - Crie a versão com ReactJS e de forma "*componentizada*" do projeto abaixo:



  - ***Exemplo Site Vanilla***

- **Atividades**
  - **1)** Enviar os exercícios em uma pasta compactada.
  - **2)** Criar um layout de uma única página usando ***ReactJS***, pode ser do próprio trabalho de ***TCC***, se já o possui, ou uma versão de algum projeto ou exercício do semestre passado de ***DFE1*****.** Enviar a versão vanilla e a nova versão com ReactJs.
  - **Requisitos:**
    - Usar o Vite.
    - Organizar a estrutura em componentes;
    - Manter arquivos css separados por componentes;
    - Possuir os seguintes componentes: ***header***, ***main*** e ***footer***.
    - Não usar páginas como login ou registro.
    - Dados do banco deverão ser “mockados”, como exemplo do ***ProductTable***.

Desenvolvimento Front-End II

Referências da disciplina

- **REFERÊNCIAS**

SILVA, Maurício Samy. **React Aprenda Praticando**. São Paulo: Novatec, 2021.

Quick Start – React: 2022 Meta Platforms, Inc. Disponível em: https://react.dev/learnl. Acesso em: 17/10/2024.

SILVA, Maurício Samy. **CSS3 – Desenvolva aplicações web profissionais com o uso dos poderosos recursos de estilização das CSS3**. São Paulo: Novatec, 2011.

SILVA, Maurício Samy. **HTML5 – A linguagem de marcação que revolucionou a web**. São Paulo: Novatec, 2011.

SILVA, Maurício Samy. **JavaScript: guia do programador.** São Paulo: Novatec, 2010.

ALURA, ***Atomic design no front-end: construa componentes escaláveis e modulares***. Disponível em: https://www.alura.com.br/artigos/atomic-design?srsltid=AfmBOorlQRz0ptT1lJxWIiV4uk0s57NOWNUtO47jIMazuO-6jUJ48myr. Acesso em: 10/03/2025