Componentes
Class vs Funcional

Componente de Class vs Funcional

React

```
import { StrictMode } from "react";
import ReactDOM from "react-dom";

import ArrowOla, { ClassOla, FunctionOla } from "./App";

const rootElement = document.getElementById("root");
ReactDOM.render(
     <StrictMode>
          <ArrowOla />
          <ArrowOla />
          <ClassOla />
          <FunctionOla />
     </StrictMode>,
     rootElement
);
```

Olá, Linguagens Script!

Olá, Linguagens Script!

Olá, Linguagens Script!

Olá, Linguagens Script!

Componente de Class vs Funcional

React

```
import React from "react";

class ClassOla extends React.Component {
     render() {
          return <h1>Olá, Linguagens Script!</h1>;
     }
}

const FunctionOla = function () {
     return <h1>Olá, Linguagens Script!</h1>;
};

const ArrowOla = () => <h1>Olá, Linguagens Script!</h1>;

export default ArrowOla;
export { ClassOla, FunctionOla };
```

Componente de Class

Componentes Funcionais

Componente de Class vs Funcional

React

```
import { StrictMode } from "react";
import ReactDOM from "react-dom";
import ArrowOla , { ClassOla} from "./App";

const rootElement = document.getElementById("root");
ReactDOM.render(
     <StrictMode>
          <ArrowOla nome="Linguagens Script" />
          <ArrowOla nome="JavaScript" />
          <ArrowOla nome="React" />
     </StrictMode>,
     rootElement
);
```

Olá, Linguagens Script

Olá, JavaScript

Olá, React

Comp. Class e Funcional > Props

```
import React from "react";

class ClassOla extends React.Component {
      render() {
            return <h1>Ola, {this.props.nome}</h1>;
      }
}

const ArrowOla = (props) => <h1>Ola, {props.nome}</h1>;

export default ArrowOla;
export { ClassOla };
```

Ola, Linguagens Script!

Componente de Class vs Funcional

- **Sintaxe**
  - Escolha depende de gostos e características do programador;
  - O componente de class usa a sintaxe ES6 e extende a componentes React com o método render que retorna elementos React. Componentes funcionais com hooks, são funções JavaScript puras que também retornam elementos React;
- ***Estado e Métodos Lifecycle***
  - Antes do React 16.8 os componentes funcionais eram stateless;
  - Em termos de ciclo de vida, é possível usar o *useEffect* hook em componentes funcionais para obter o mesmo efeito que métodos como *componentDidMount*, *componentDidUpdate* e *componentWillUnmount* dos componentes de classse.
- O **this** e métodos de *binding* deixam de ser necessários em componentes funcionais; Partilha de estados entre componentes class é mais tediosa;

Considerações de Desempenho

- Adicionar **keys** a lista de elementos

```
function NumberList(props) {
  const numbers = props.numbers;
  const listItems = numbers.map((number) =>
    <li key={number.toString()}>
      {number}
    </li>
  );
  return (
    <ul>{listItems}</ul>
  );
```

- **Construir componentes pequenos!**
  - Caso contrário… muitas variáveis de estado no componente, implica, renderização de tudo!

Map JavaScript vs JSX


```
function numberList(props) {
  const numbers = props;
  const listItems = numbers.map((number) =>
                          number % 2 === 0 && number);
  console.log(listItems);
  return listItems;
}
console.log(numberList([1, 2, 3, 4, 5, 6, 7]));
```

```
[false, 2, false, 4, false, 6, false]
```

Map JavaScript vs JSX


```
export function App(props) {
 function numberList(numbers) {
    const listItems = numbers.map( (number) =>
            (number % 2 === 0 && <li key={number.toString()}>
                                    {number}
                                 </li>));
    console.log(listItems)
    return listItems
  }
 return (<> {numberList([1,2,3,4,5,6,7])} </>);
}
```

- 2
- 4
- 6

Considerações de Desempenho

- Evitar montar e desmontar componentes

```
import React, { useState } from "react";
const MountingComponent = () => {
  const [show, setShow] = useState(false);
  return (
    <main>
      <button onClick={() => setShow((show) => !show)}>Show the text</button>
      {show ? <p>I am the text</p> : null}
    </main>
  );
};
```

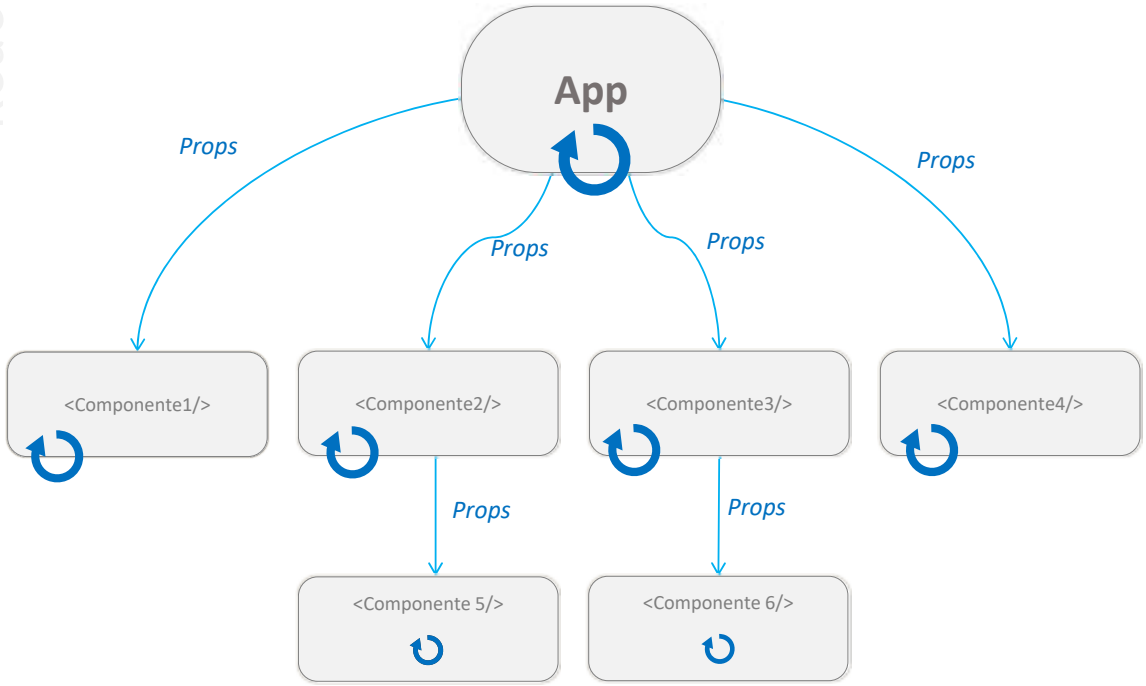
```
const CSSWay = () => {
  const [show, setShow] = useState(false);
  return (
    <main>
      <button onClick={() => setShow((show) => !show)}>Show the text</button>
      <p style={{ opacity: show ? "1" : "0" }}>I am the text</p>
    </main>
  );
};
```

Gestão de Estados

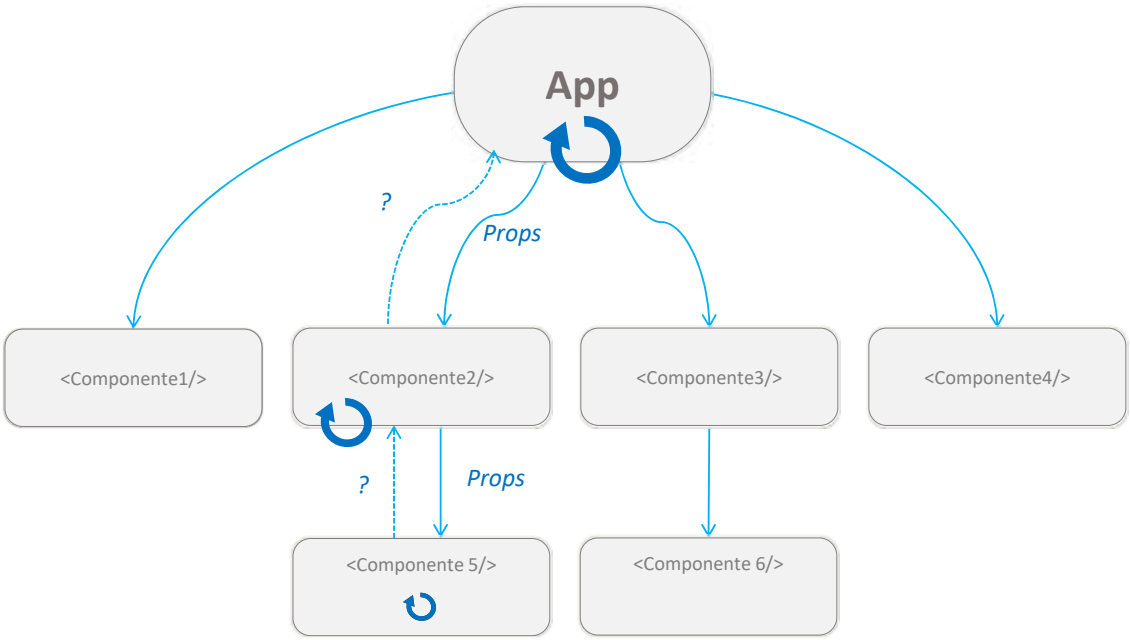
State and props

- Permite armazenar informação que influencia o resultado da renderização.
  - **Props** são uma característica básica do React e permite encadear informações de pai para filho. Portanto, são passados para o componente como parâmetros de funções.
  - **State** gerido dentro do componente,
    como variáveis declaradas dentro da função.

Props vs Estados



Props vs Estados



Gestão de Estados

- Existem dois locais principais no qual os estados podem residir:
  - **Nos componentes**
  - Armazenamento central
- Nos componentes
  - Fluxo de dados fácil de compreender
  - Trabalha de forma facilitada quando a aplicação não é demasiado complexa de forma a que os componentes não precisem partilhar dados e interagir
- Exite um conjunto de outras técnicas para manipulação de dados
  - render props
  - the Context API
  - Redux.

Comunicação entre Componentes

- Passar informação do element pai para o filho: ***Usar props*** (atributos)
  - `<ComponentFilho param={infoParaComponenteFilho} />`
- Passar informação do filho para o pai: ***Callbacks***
  - `paiCallback = (infoDoFilho) =>`
    `{ /* processInfoDoFilho*/};`
  - `<ComponentFilho callback={paiCallback}> />`
- ***React Context / useContext****
  - Variáveis globais para a subárvore de componentes
  - Fornece uma forma de passar dados pela àrvore de componentes sem necessidade de passar propriedades manualmente em todos os níveis*
- ***Redux / useReducer ****

**não será explorado*

Estados

- Componentes necessitam de alterar como resultado de uma interação na aplicação
  - *Clique em determinado botão*
  - *Alterar posição do rato*
  - *Tempo chegou a determinado valor*
- Componentes necessitam de "lembrar" coisas
  - "**Memória**" do Componente
- Valores que estejam directamente refletidos no UI
  - Não deve ser usado para valores "behind the scene" que nao tenham impacto na UI

`useState` `this.state`

Estados


```
import { listaElementos } from "./dados.js";
export default function Galeria() {
  let index = 0;
  function handleClick() {
    index = index + 1;
  }
  let foto = listaElementos[index];
  return (<><button onClick={handleClick}>Next</button>
            <h2><i>{foto.titulo} </i> {foto.nome}</h2>
            <h3>({index + 1} of {listaElementos.length})</h3>
            <img src={foto.url} alt={foto.alt} />
            <p>{foto.descricao}</p></>
  );
}
```

Estados

- *Variáveis Locais:*
  - **não persistem entre renderizações**
  - **não forçam uma nova renderização**
- Atualização do componente com novos dados
  - **Reter** dados entre atualizações (renderizações)
  - **Acionar** nova renderização, com os novos dados
- **Hook useState em Components Funcionais**
  - Variável de estado para reter dados entre as renderização
  - Uma função setter para atualizar a variável de estado e acionar o react para renderizar novamente o componente
  - import { useState } from 'react';

Estados


```
import { listaElementos } from "./dados.js";
export default function Galeria() {
  let index = 0;
  function handleClick() {
    index = index + 1;
  }
  let foto = listaElementos[index];
  return (<><button onClick={handleClick}>Next</button>
            <h2><i>{foto.titulo} </i> {foto.nome}</h2>
            <h3>({index + 1} of {listaElementos.length})</h3>
            <img src={foto.url} alt={foto.alt} />
            <p>{foto.descricao}</p></>
  );
}
```

Estados

React

```
import { listaElementos } from "./dados.js";
export default function Galeria() {
  const [index, setIndex] = useState(0);
  function handleClick() {
    setIndex(index + 1);
  }
  let foto = listaElementos[index];
  return (<><button onClick={handleClick}>Next</button>
            <h2><i>{foto.titulo} </i> {foto.nome}</h2>
            <h3>({index + 1} of {listaElementos.length})</h3>
            <img src={foto.url} alt={foto.alt} />
            <p>{foto.descricao}</p></>
  );
}
```

Array destructuring

Estados: Comp. Funcional

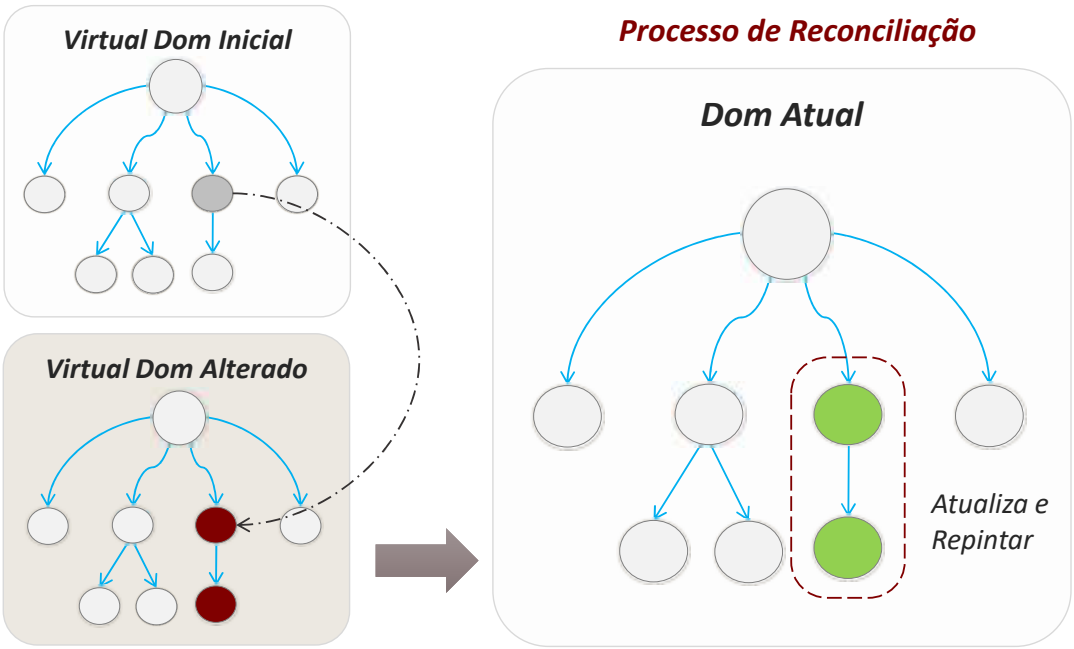
React

```
import { listaElementos } from "./dados.js";
export default function Galeria() {
const [index, setIndex] = useState(0);

function handleClick() {
    console.log(index);
    setIndex(index + 1);
    console.log(index);
}
  let foto = listaElementos[index];
  return (
    <>
      <button onClick={handleClick}>Next</button>
      <h2><i>{foto.titulo} </i> {foto.nome}</h2>
      <h3>({index + 1} of {listaElementos.length})</h3>
      <img src={foto.url} alt={foto.alt} />
      <p>{foto.descricao}</p>
    </>
  );
}
```

DOM vs Virtual Dom

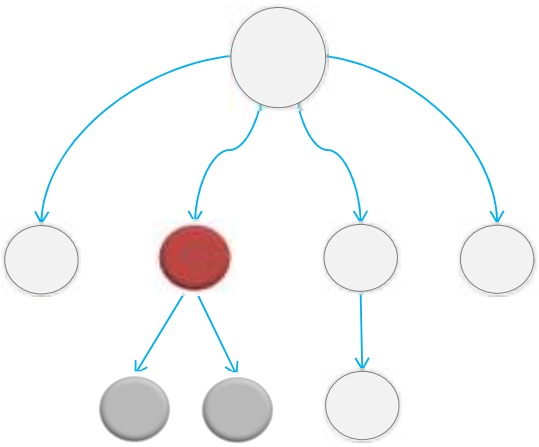
- O React.js possui algoritmos que permitem decidir quando atualizar o DOM no browser melhorando o desempenho das aplicações front-end.



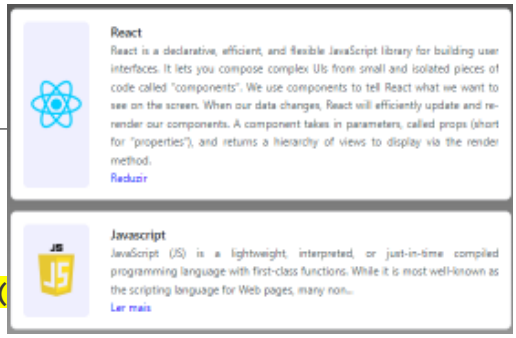
Estados

Partilhar estados entre componentes

lifting state up

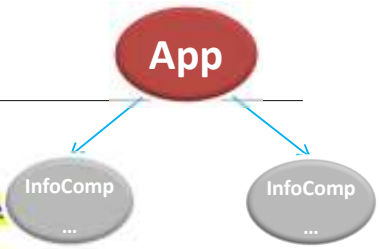


Estados




```
import React, {useState} from "react";
import {getText} from  "../funcs.js";
function InfoComponent(props) {
  const [isAllVisible, setIsAllVisible] = useState(
  const texto = props.children;
 return (
      <div className="wrapper">
        <div className="logo"> <img src={props.src} alt={props.title} /> </div>
        <div className="text">
          <h2>{props.title}</h2>
          <div>
            <p>{getText(texto, maxLength, isAllVisible)}</p>
              {isAllVisible
                ? (<button onClick={()=>setIsAllVisible(false)}>Reduzir</button>)
                  : (<button onClick={()=>setIsAllVisible(true)}>Ler Mais</button>)
                }
          </div>
        </div>
      </div>
  );
}
```


Estados




```
function App () {
  const [allTextVisible, setAllTextVisible] = useState(fal
  const handleAllTextVisible = () => setAllTextVisible(!allTextVisible);
  return (<div id="container">
   <InfoComponent title="React" src="./assets/images/react.png"
        allTextVisible={allTextVisible}  onAllTextVisible={handleAllTextVisible} >
        React is a declarative... </InfoComponent>
   <InfoComponent title="Javascript" src="./assets/images/javascript.png"
         allTextVisible={allTextVisible}   onAllTextVisible={handleAllTextVisible} >
        JavaScript (JS) is a light... </InfoComponent>
    </div>
  );
}
```

Estados


```
function InfoComponent(props) {
  const { title, src, allTextVisible, onAllTextVisible } = props;
  const texto = props.children;
  return (
    <div className="wrapper">
      <div className="logo">
        <img src={src} alt={title} />
      </div>
      <div className="text">
        <h2>{title}</h2>

        <p>{getText(texto, maxLength, allTextVisible)}</p>
           {allTextVisible
           ? (<button onClick={onAllTextVisible}> Reduzir Todos </button>)
           : (<button onClick={onAllTextVisible}> Expandir Todos </button> )}
      </div>
    </div>
  );
}
```