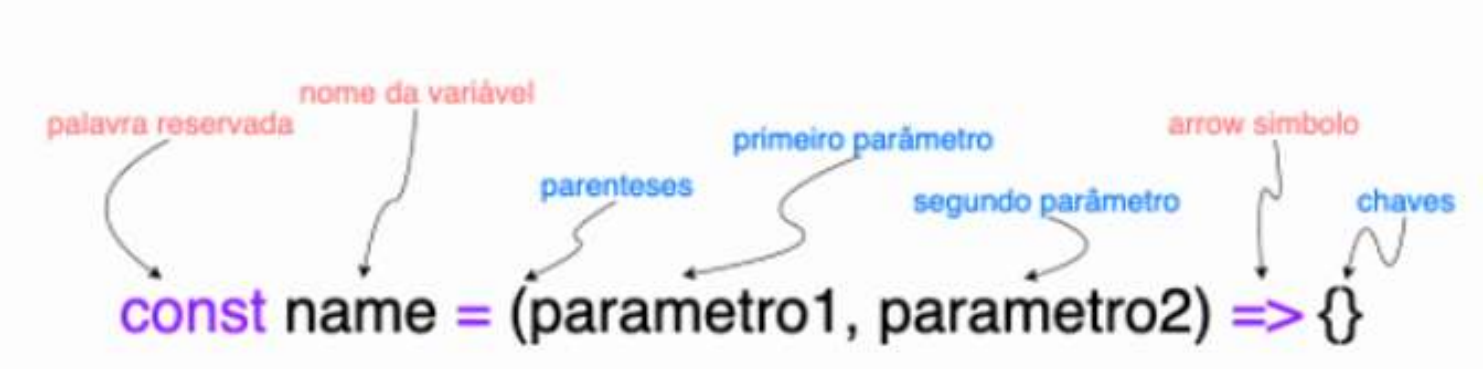
React ES6

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

- **ECMAScript** é uma especificação de linguagem, ou seja, ela define os padrões para uma linguagem de programação, e o JavaScript é a implementação desses padrões;
  - Es6_1.html,
  - Es6_2.html,
  - Es6_3html,
  - Es6_4.html
  - Es6_5.html
  - Es6_6.html
  - Es6_7.html
  - Es6_8.html
  - React_Arrow.html



- **addEventListener** é a maneira de registrar uma espera de evento;

- https://developer.mozilla.org/pt-BR/docs/Web/API/Event

A função .map()

O **.map()** é um método de array em JavaScript que permite iterar sobre uma lista e transformar seus elementos, retornando um **novo array**.

Ele é muito usado no React para renderizar listas dinamicamente.

```
> const numeros = [1, 2, 3, 4];
<· undefined
> dobrados = numeros.map((num) => num * 2);
<·  ▸ (4) [2, 4, 6, 8]
>
```

A função .map()

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

Sintaxe Básica:

```
const novoArray = arrayOriginal.map((elemento, indice, array) => {
  return novoValor;
});
```

- elemento: item atual do array.
- índice *(opcional)*: posição do item no array.
- array *(opcional)*: o próprio array original.
- Retorna um **novo array** com os elementos modificados.

A função .map()

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

Exemplo com react

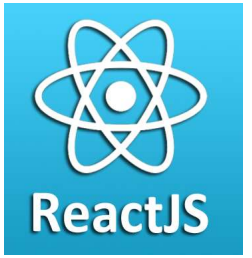
```
const nomes = ["Ana", "Carlos", "Beatriz"];

function ListaNomes() {
  return (
    <ul>
      {nomes.map((nome, index) => (
        <li key={index}>{nome}</li>
      ))}
    </ul>
  );
}
```

O React precisa de uma chave única (key) para identificar os elementos ao re-renderizar a lista.

Atividade: Testar este trecho de código diretamente no HTML, fazendo os ajustes que você achar necessário.

React JSX



2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

- **JSX** é um acrônimo para JavaScript extention que permite escrever código JavaScript parecido com HTML;
- **JSX** é uma sintaxe semelhante a HTML usada por **react**;
- Com o **JSX**, você pode escrever expressões dentro de chaves {}
- O código HTML deve ser agrupado em UM elemento de nível superior;

React JSX

- Para escrever HTML em várias linhas, coloque o HTML entre parênteses;

- Não podemos usar instruções if else dentro do **JSX**, mas podemos usar expressões condicionais (ternárias)

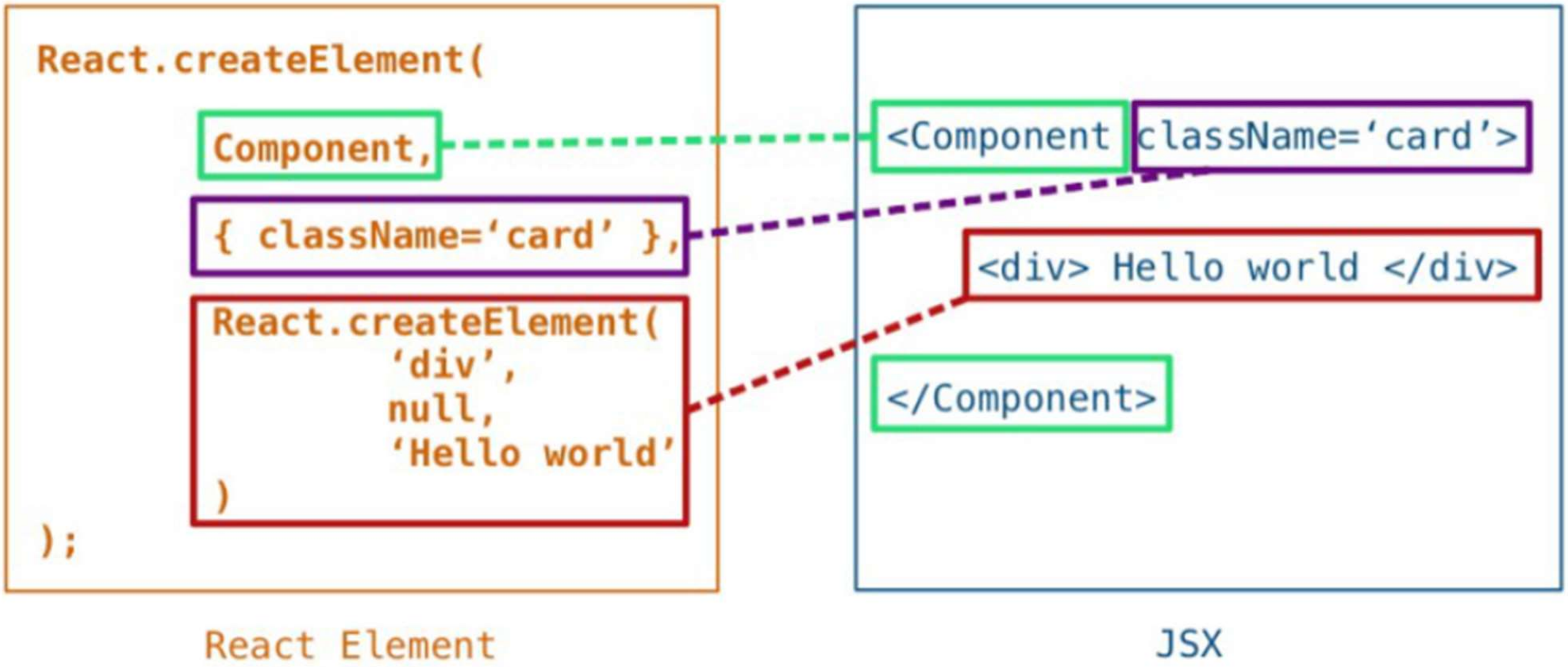
  - Elementos aninhados REACT_JSX_Nested.html
  - Atributos; REACT_JSX_Atributos.html
  - Expressôes Java Script; REACT_JSX_Expressões.html
  - Estilo; REACT_JSX_Estilo.html
  - Comentários; REACT_JSX_Comentarios.html

React JSX

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

Sem JSX

Com JSX



React JSX

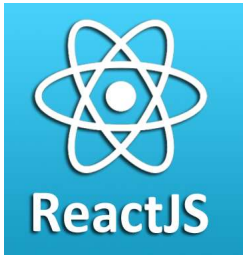
Com JSX

```
<MyButton color="blue" shadowSize={2}>
  Click Me
</MyButton>
```

Sem JSX

```
React.createElement(
  MyButton,
  {color: 'blue', shadowSize: 2},
  'Click Me'
)
```

React JSX



2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

Com JSX

```
const myelement = <h1>I Love JSX!</h1>;

ReactDOM.render(myelement, document.getElementById('root'));
```

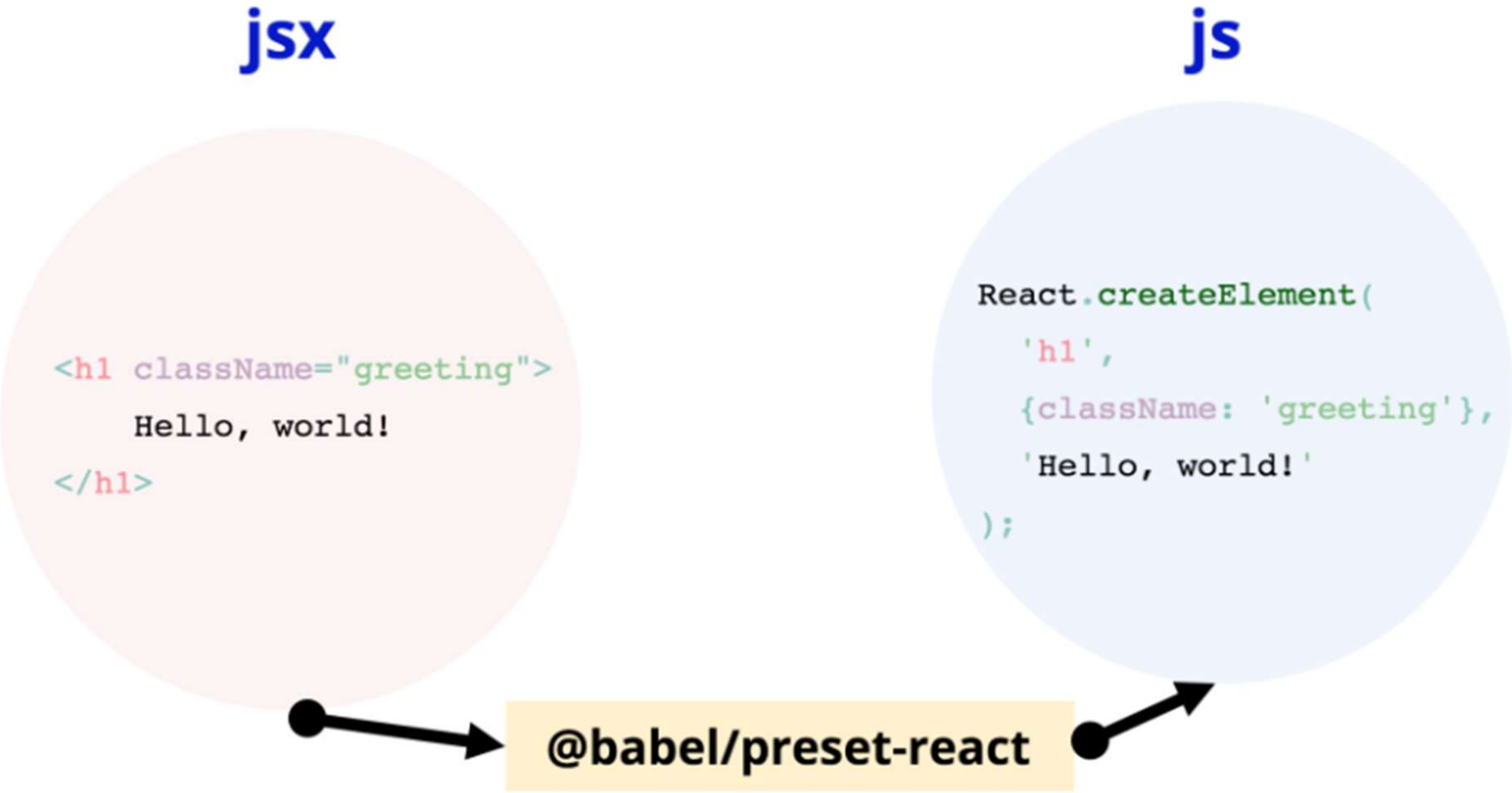
Sem JSX

```
const myelement = React.createElement('h1', {}, 'I do not use JSX!');

ReactDOM.render(myelement, document.getElementById('root'));
```

React JSX

2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207



React Components

- **Components** são como funções que retornam HTML;
- **Components** são de dois tipos: Class e Function;
- O nome do **Components** deve começar por Letra maiúscula;
- Deve incluir a sentença: *extends React.Component*;
- Requer o método *render(),* este método retorna HTML;
  - Component_Class1.html
  - Component_Class2.html
  - Component_Function.html
  - Component_Function2.html

React Components

2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207

- Se houver uma função *constructor()* no seu **componente**, essa função será chamada quando o **componente** for iniciado;
  - Component_Class_constructor1.html
  - Component_Class_constructor2.html

- **Components in Components**
  - Component_Class2.html
  - Component_Class3.html

React Props

2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207

- **Props** são argumentos passados para os componentes React;
- **Props** = Propriedades.
- **Props** são como argumentos de funções em JavaScript e atributos em HTML;
  - Props1.html,
  - Props2.html,
  - Props3.html,
  - Props4.html
  - Props5.html
- **Props** são imutaveis;
- Default **Props**
  - Props6.html
  - Props7.html
  - Props8.html
  - Props9.html *
  - Props10.html *

React Props

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

Componente Sem argumentos

```
function HelloWorld() {
  return <span>Hello, World!</span>;
}

// Render
<HelloWorld />

// Output
<span>Hello, World!</span>
```

Com argumentos

```
function Hello(props) {
  return <div>Hello, {props.who}!</div>;
}
```

```
// Render
<Hello who="Earth" />

// Output
<div>Hello, Earth!</div>
```

React Props

Com argumentos

```
class HelloAsClass extends Component {
  render() {
    return <div>Hello, {this.props.who}!</div>;
  }
}

// Render
<HelloAsClass who="Earth" />

// Output
<div>Hello, Earth!</div>
```

React State

- O **state** (estado) é um objeto que armazena dados que podem mudar ao longo do tempo em um componente. Ele permite que o React **reaja a mudanças** e **re-renderize** a interface quando necessário.

- Quando o objeto **state** é alterado, o componente é renderizado novamente.

React State

- O **state** é inicializado no construtor;
  - State1.html,
- O **state** pode conter quantas propriedades forem necessárias;
  - State2.html
  - State3.html
  - State4.html

React State

2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207

Usando **state** em um Componente de Classe

Em **componentes de classe**, o state precisa ser inicializado no constructor e atualizado com **this.setState().**

```
class Contador extends React.Component {
  constructor(props) {
    super(props);
    this.state = { numero: 0 };
  }

  aumentar = () => {
    this.setState({ numero: this.state.numero + 1 });
  };

  render() {
    return (
      <div>
        <h1>{this.state.numero}</h1>
        <button onClick={this.aumentar}>Aumentar</button>
      </div>
    );
  }
}
```

- O estado inicial é { numero: 0 }.
- O botão chama this.setState() para atualizar o número.
- O React **re-renderiza** o componente quando o estado muda.

React State

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

Usando **state** com funções

Nos **componentes funcionais**, usamos o **Hook useState()** para lidar com o estado.

```
import { useState } from "react";

function Contador() {
  const [numero, setNumero] = useState(0);

  return (
    <div>
      <h1>{numero}</h1>
      <button onClick={() => setNumero(numero + 1)}>Aumentar</button>
    </div>
  );
}
```

- useState(0) define o estado inicial como 0.
- setNumero(numero + 1) atualiza o estado.
- O componente **re-renderiza** quando numero muda.

React State

Atividade:

Coloque o código abaixo para funcionar em uma página HTML, fazendo os ajustes que você achar necessário.

```
import React, { useState } from "react";
export default function SwitchMode(){
    const [mode, SetMode] = useState('claro');

    return (
        <div>
            <button onClick={() =>{
                const newState = mode === 'claro' ? 'escuro' : 'claro';

                SetMode(newState)
            }}>
                {mode === 'claro' ? '🌞' : '🌙'}
            </button>
        </div>
    )
}
```

React State

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

RESUMO:

- ✅ O state armazena informações dinâmicas dentro do componente.
- ✅ Em **classes**, usamos this.state e this.setState().
- ✅ Em **funções**, usamos o Hook useState().
- ✅ Sempre que o state muda, o React **atualiza a UI automaticamente**.

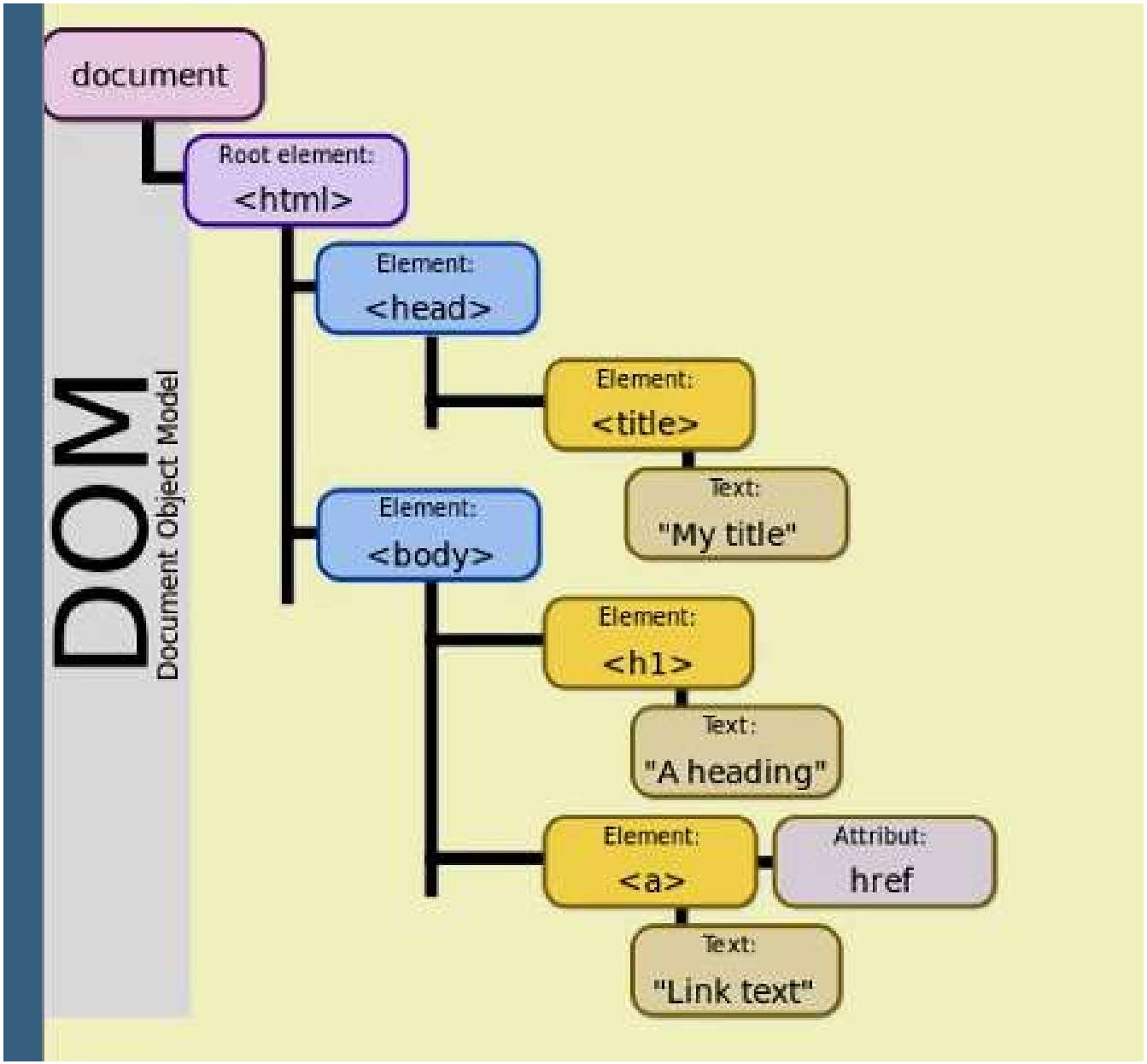
React Diferença entre State e Props

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

State	Props
Gerenciado dentro do componente.	Passado de fora do componente.
Pode ser modificado dentro do componente.	Não pode ser alterado pelo componente que o recebe.
Usado para armazenar e atualizar informações dinâmicas.	Usado para passar dados e configurações para componentes filhos.

Document Object Model

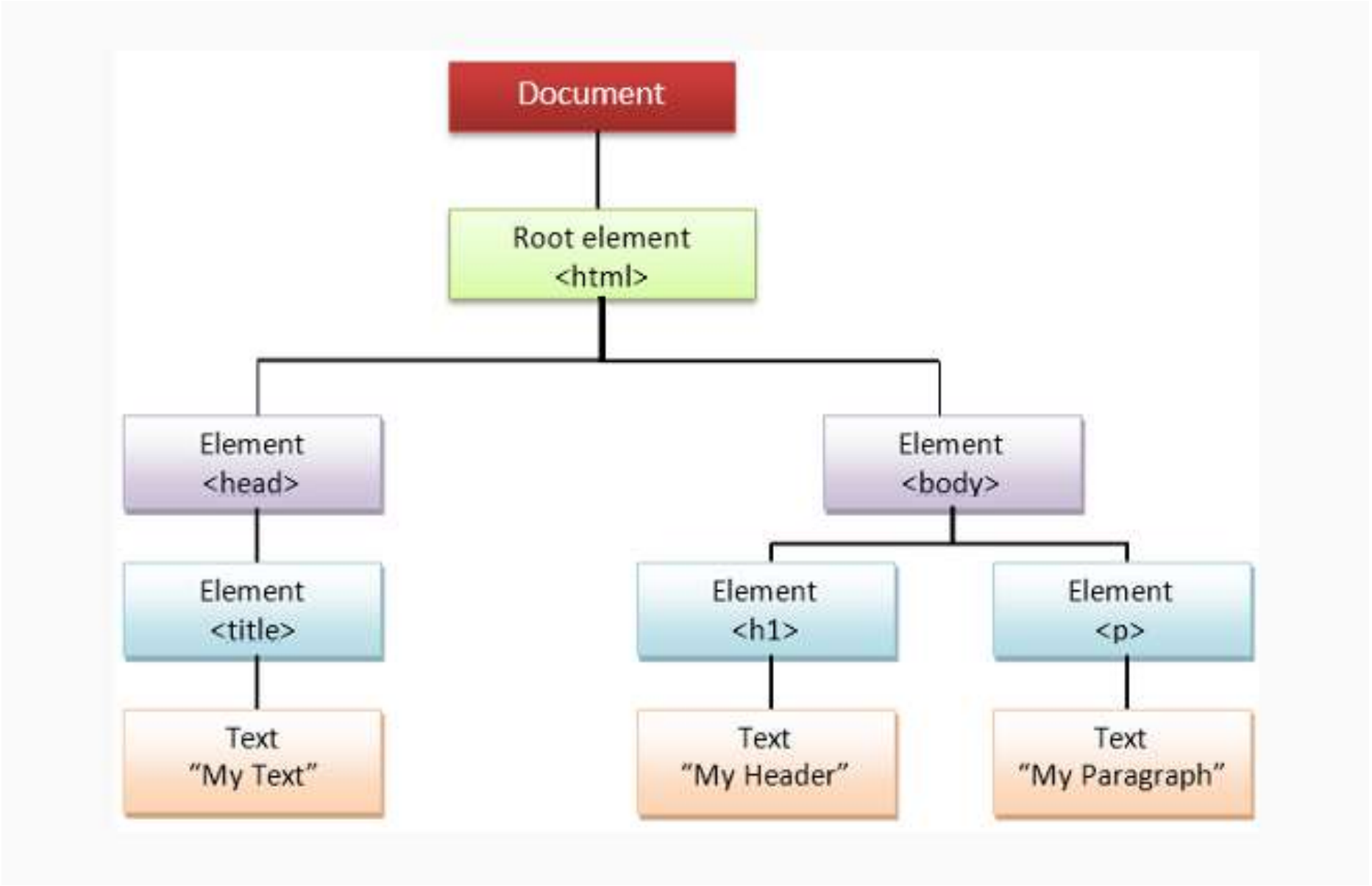
2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207



O DOM (**Document** Object Model) é uma interface que representa como os documentos HTML são lidos pelo seu browser;

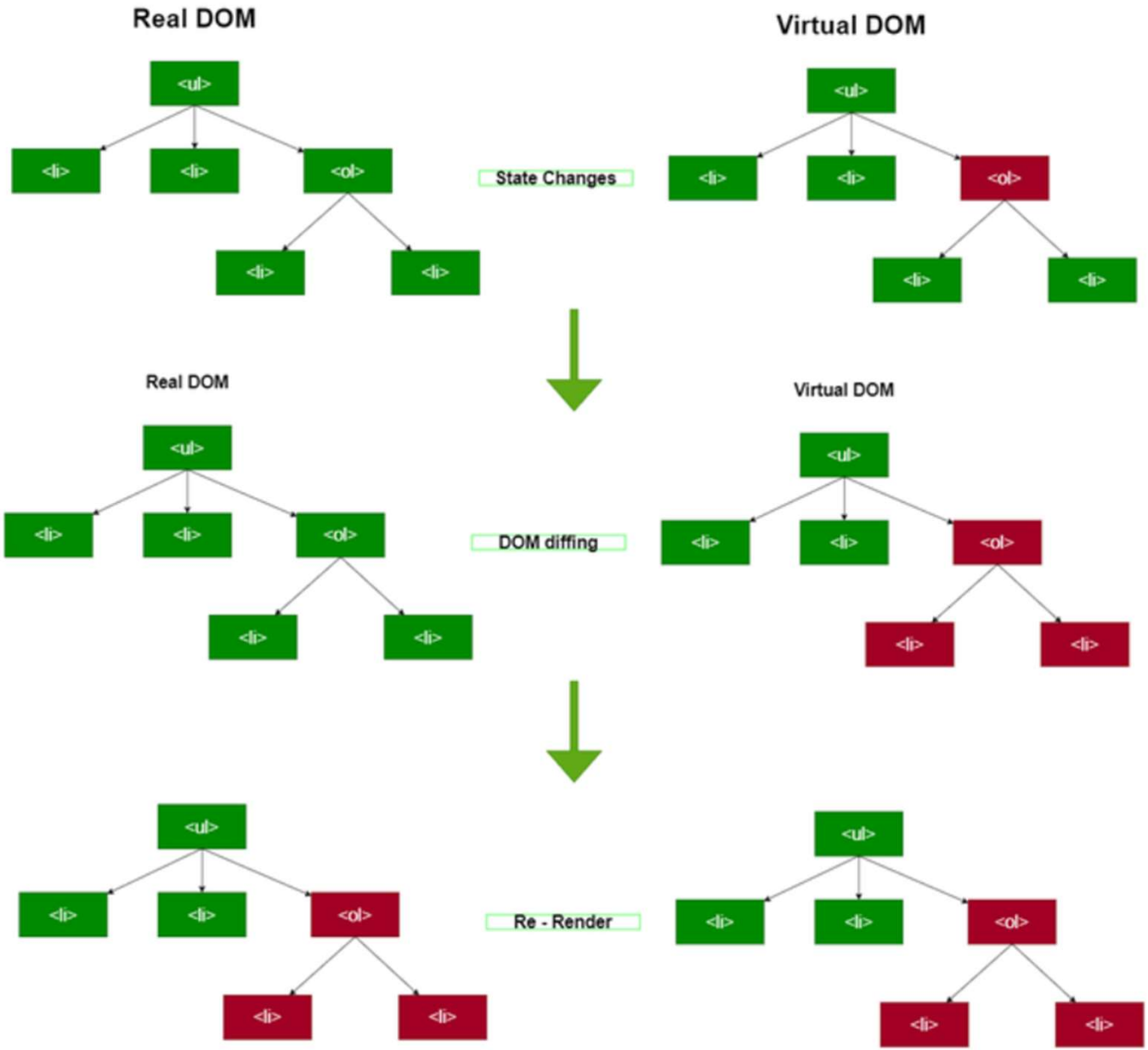
Document Object Model

2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207



Document Object Model

2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207



React Events

2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207

- Assim como o HTML, o React pode executar ações com base nos eventos do usuário;

- React tem os mesmos eventos que HTML;

- Os eventos react são gravados na sintaxe camelCase:
  - on**C**lick ao invés de onclick.

- Manipuladores de eventos do react são escritos dentro de chaves;
  - onClick={shoot} ao invés de onClick="shoot()"
    - (Event1.html) à (Event6.html)

REACT – Atividade

2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657

- Atividade
  - Alterar o TicTacToe para:

1. Exibir a página conforme abaixo;
2. Ao clicar em um número exibir X no seu lugar;

Próximo Jogador: XPróximo Jogador: X

0	1	2
3	4	5
6	7	8

0	1	2
3	4	5
6	7	8

REACT – Atividade

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657

- Atividade
  - Documentar o código do TicTacToe :