

React

Uma biblioteca Javascript para criar interfaces

Allan Ramos

Desenvolvedor na Pagar.me

Github \ Twitter \ Medium: allangrds

Mas antes disso,

precisamos recapitular

AJAX

por Google em 2005



Bootstrap

por Twitter em 2011

Breadcrumbs

[Home](#) / [Library](#) / [Data](#)

Pagination

« 1 2 3 4 5 »

Pagers

Previous

Next

? Older

Newer ?

Tabs

Section 1

Section 2

Section 3

I'm in Section A.

Pills

Home

Profile

Dropdown ▼

Disabled link

Nav Lists

Home

Library

Applications

Help

Labels

Default

Success

Warning

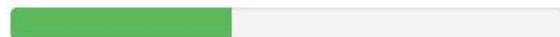
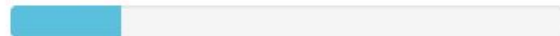
Danger

Info

Badges

Default

Progress bars



Single Page Application

Uma **single page application** é uma aplicação web que carrega o site apenas uma vez.



[Home](#)[Download / Install](#)[Tutorials](#)[Live examples](#)[Documentation](#)[Forum](#)[Source](#)

Knockout.

Simplify dynamic JavaScript UIs with the Model-View-View Model (MVVM) pattern

Download
v3.4.2 - 22kB min+gz



[release notes](#)

Download RC
v3.5.0 RC



[release notes](#)

Key concepts



Declarative Bindings

Easily associate DOM elements with model data using a concise, readable syntax



Automatic UI Refresh

When your data model's state changes, your UI updates automatically



Dependency Tracking

Implicitly set up chains of relationships between model data, to transform and combine it



Templating

Quickly generate sophisticated, nested UIs as a function of your model data

More features

Free open source (MIT license)

New: Interactive tutorials

Get started with knockout.js quickly, learning to build *single-page applications*.

Framework Knockout.js por Steve Sanderson em 2011

Experiência do Usuário

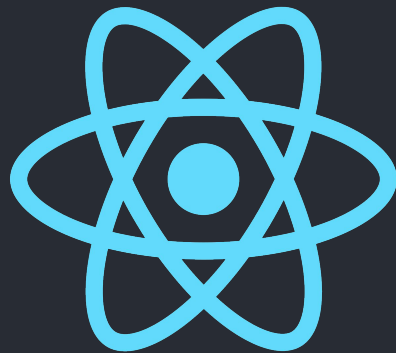
Performance





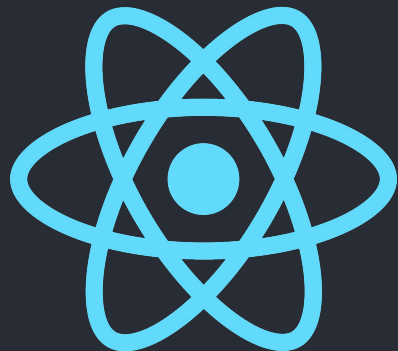
Cuidado

SPA não é
bala de prata



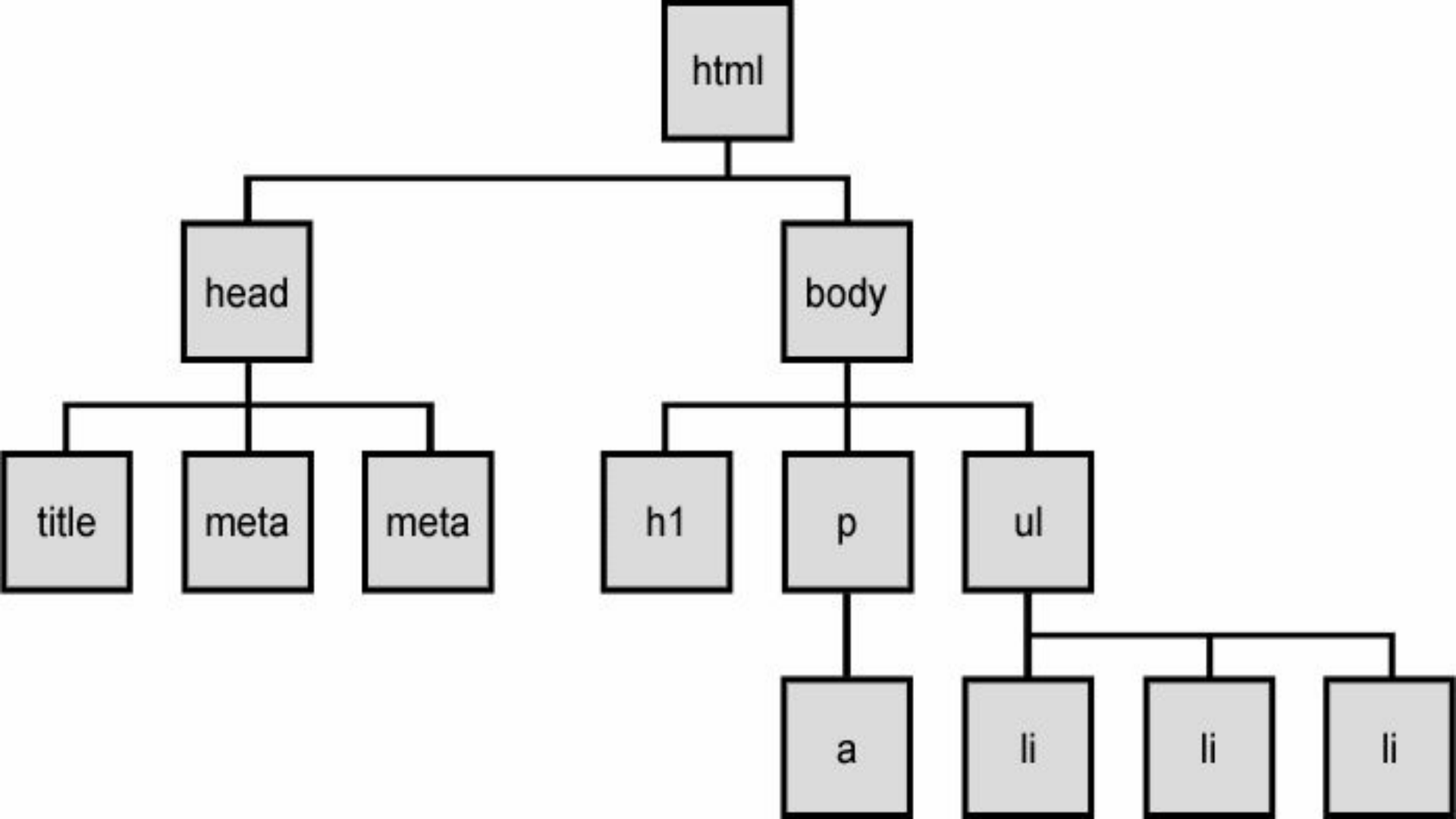
React

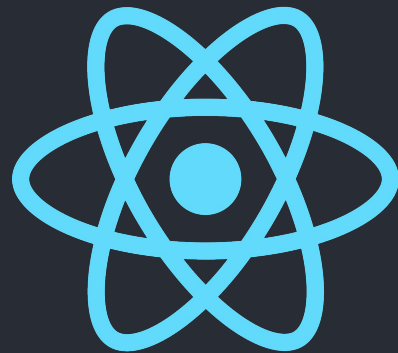
Uma biblioteca Javascript para criar interfaces



DOM por Virtual DOM

Manipular o DOM é custoso





Componentes

Reaproveite o que pode se repetir

Um **componente** é uma **parte visual** que
pode ser **reutilizada**



```
1 class Title extends React.Component {  
2   render() {  
3     return React.createElement('h1', null, 'Olá Mundo');  
4   }  
5 }
```

<https://codepen.io/anon/pen/pOEoYg?editors=1010>

Explicando a estrutura

- Title - nome da classe;
- extends - especifica que aquela classe herdará atributos/comportamentos de outra;
- React.Component - Classe pai;
- render - função que renderiza conteúdo;

Eu **não** preciso de **componentes**, tenho
as tags do **HTML** e o **JS**

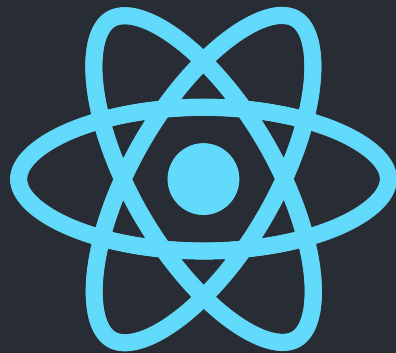


```
1 # Quando usamos o HTML
2
3 <button class="btn btn-dark">
4     Botão
5 </button>
6
7 <button class="btn btn-light">
8     Botão
9 </button>
10
```



```
1 # Quando usamos o HTML
2
3 <button class="btn btn-dark">
4     Botão
5 </button>
6
7 <button class="btn btn-light">
8     Botão
9 </button>
10
11 -----
12
13 # Quando usamos o React
14
15 <Button
16     type="dark"
17     title="Botão"
18 />
```

- Facilita a leitura;
- Permite ter comportamento padrão;
- Permite ter estilos dinâmicos;
- Abstrai comportamento.



JSX

Uma extensão da sintaxe do Javascript



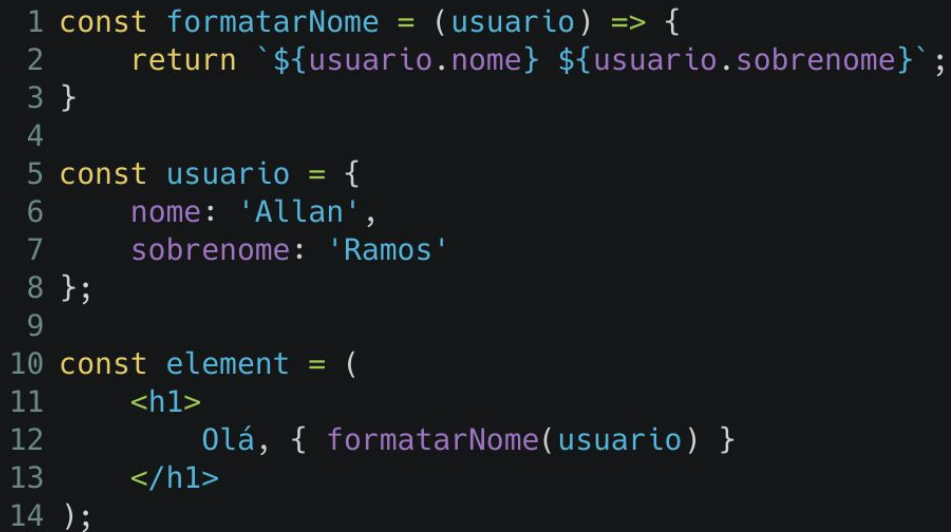
```
1 const element = <h1>Hello, world!</h1>;
```

Nem string, nem HTML. JSX!



```
const nome = 'Allan Ramos';  
const element = <h1>Olá, { nome }</h1>;
```

Uma variável dentro de uma expressão JSX



```
1 const formatarNome = (usuario) => {  
2     return `${usuario.nome} ${usuario.sobrenome}`;  
3 }  
4  
5 const usuario = {  
6     nome: 'Allan',  
7     sobrenome: 'Ramos'  
8 };  
9  
10 const element = (  
11     <h1>  
12         Olá, { formatarNome(usuario) }  
13     </h1>  
14 );
```

Uma função sendo chamada dentro de uma expressão JSX

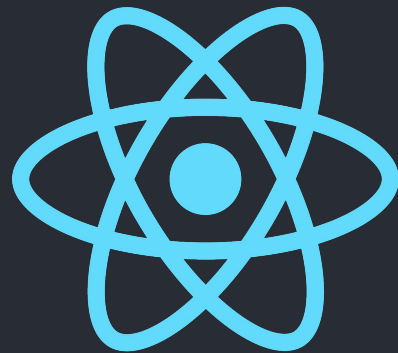


```
class Welcome extends React.Component {  
  render() {  
    return React.createElement('h1', null, 'Olá Mundo!');  
  }  
}
```

React sem JSX



```
class Welcome extends React.Component {  
  render() {  
    return <h1>Olá Mundo!</h1>;  
  }  
}
```



State

O state(estados) é qualquer tipo de informação que poderá ser manipulada pelo componente.

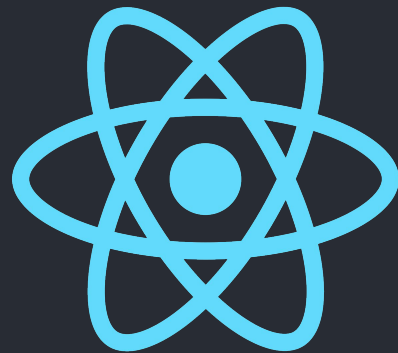
```
class Tela extends React.Component {
  constructor() {
    super();
    this.state = { numero: 0 };

    this.aumentaNumero = this.aumentaNumero.bind(this);
    this.diminuiNumero = this.diminuiNumero.bind(this);
  }

  aumentaNumero() {
    this.setState({
      numero: ++this.state.numero
    });
  }

  diminuiNumero() {
    this.setState({
      numero: --this.state.numero
    });
  }

  render() {
    return (
      <div>
        <p>Número: { this.state.numero }</p>
        <button onClick={this.aumentaNumero}>Aumenta</button>
        <button onClick={this.diminuiNumero}>Diminui</button>
      </div>
    );
  }
}
```

Self closed tags

Eles parecem tags HTML



```
1 <Imagem src="meulogo.png" alt="Meu Logo" />
2
3 <Botao>
4     Clique aqui
5 </Botao>
```

Imagem - Self closed component



**Cuidado com
as versões**



```
{  
  "scripts": {  
    ...  
  },  
  "devDependencies": {  
    "eslint": "^4.19.1",  
    "execa": "0.10.0",  
  },  
}
```

#partiuDocs

<https://reactjs.org/>

#partiuFazer

<https://github.com/facebook/create-react-app>



```
1 npm install --global create-react-app  
2 create-react-app ola-mundo
```

Instalando o projeto create-react-app



```
1 cd ola-mundo  
2 npm start
```

Rodando o projeto

MY-APP

- node_modules
- public
 - favicon.ico
 - index.html
 - manifest.json
- src
 - App.css
 - App.js
 - App.test.js
 - index.css
 - index.js
 - logo.svg
 - registerServiceWorker.js
- .gitignore
- package.json
- README.md
- yarn.lock

Explicando a estrutura

- public/
 - assets
 - index.html
- src/
 - Código do projeto



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

Explicando a estrutura

- ReactDOM - Biblioteca responsável por manipular o DOM;
- React - Biblioteca responsável por todo o ecossistema React;
- index.css - Estilo da aplicação;
- App - Componente;
- ReactDOM.render - Função que insere o App na div com id “root”.

Vamos fazer algo
juntos?

Live Coding

1. Deverão existir dois inputs numéricos;
2. Deverá existir um botão “Calcular”;
3. Ao clicar no botão, deverá aparecer o seguinte texto:
 - a. “Olá <nome>, o resultado da soma é <resultado>”

#partiuExercício

TODO

- Tarefa #1
- Tarefa #2

What needs to be done?

Add #3

Exercício

1. Será uma mini aplicação de TodoList. Ela deverá permitir que os usuários adicionem novas tarefas;
2. Deverá conter um input;
3. Deverá conter um botão;
4. Deverá mostrar a lista de tarefas conforme for clicado no botão.

Referências

<http://blog.locaweb.com.br/artigos/desenvolvimento-artigos/o-que-e-single-page-application/>

<https://vizir.com.br/2016/08/quando-e-porque-desenvolver-uma-spa-single-page-application/>

<http://paislee.io/evolution-of-the-single-page-application/>

<https://medium.com/@pshrmn/demystifying-single-page-applications-3068d0555d46>

Referências

<https://scotch.io/@anitashah/what-problems-does-reactjs-solve-when-must-you-select-reactjs>

<https://code.tutsplus.com/pt/tutorials/getting-started-with-react-and-jsx--cms-27352>

<https://reactjs.org/docs/react-without-jsx.html>

<https://reactjs.org/docs/introducing-jsx.html>