

crescere

ReactJS - Introdução

Lucas Soares

O que é React?

- Biblioteca para construção de interfaces
- Utilizado para SPAs (Single Page Application) e PWA (Progressive Web Apps)
- Framework?
- Tudo em Javascript (HTML, CSS...)
- React / ReactJS com (react-dom) / React Native (RN) com Native
- Ambiente de desenvolvimento bem completo

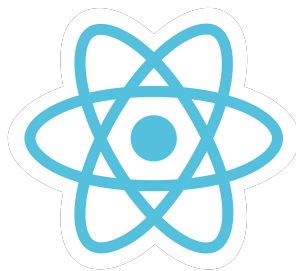
Casos de uso

- Landing pages
- Conteúdo estático
- Pouca ou nenhuma conexão com back-end



- Aplicações mais robustas
- Maior uso de back-end
- Trabalho em equipe
- Utilizar todo conhecimento

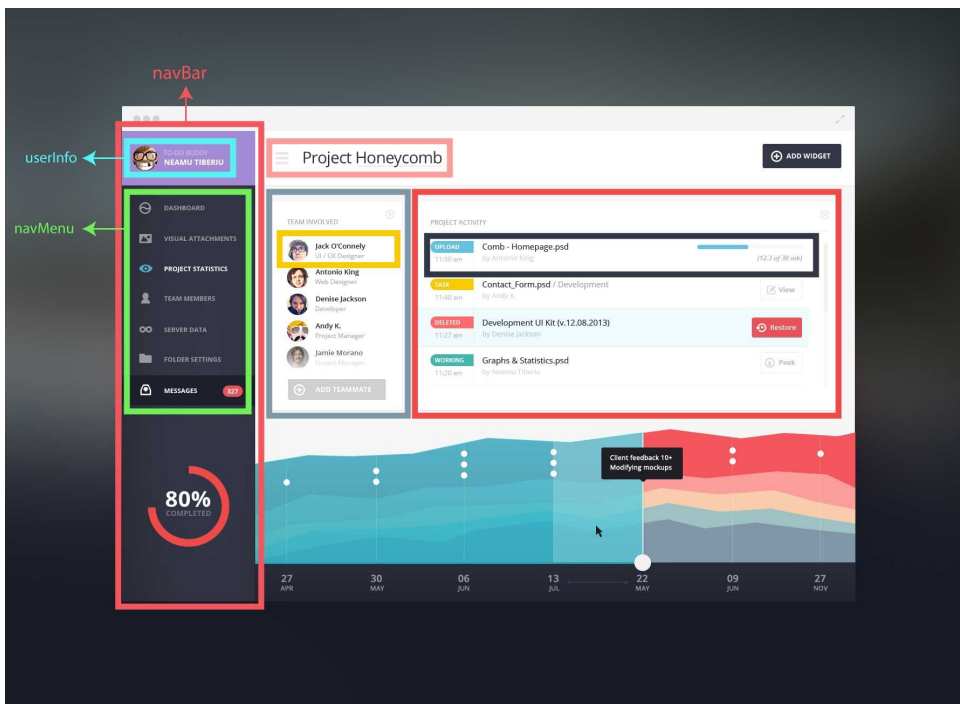
HTML + CSS



Vantagens

- Organização do código
 - Componentes
- Divisão de aplicações
 - Back-end: API
 - Front-end: Interface
- Diversos clientes
- Programação declarativa

Componentes



- Lógica
- Estrutura
- Estilo
- Funcionalidades

JSX

- Escrever **HTML e CSS** dentro do **Javascript**
- Podemos criar nossos próprios elementos
- Criar os nossos componentes

JSX

```
1  function Button() {  
2    return (  
3      <button type="button">  
4        <span class="icon">!</span>  
5      </button>  
6    )  
7  }
```

Imperativo vs Declarativo

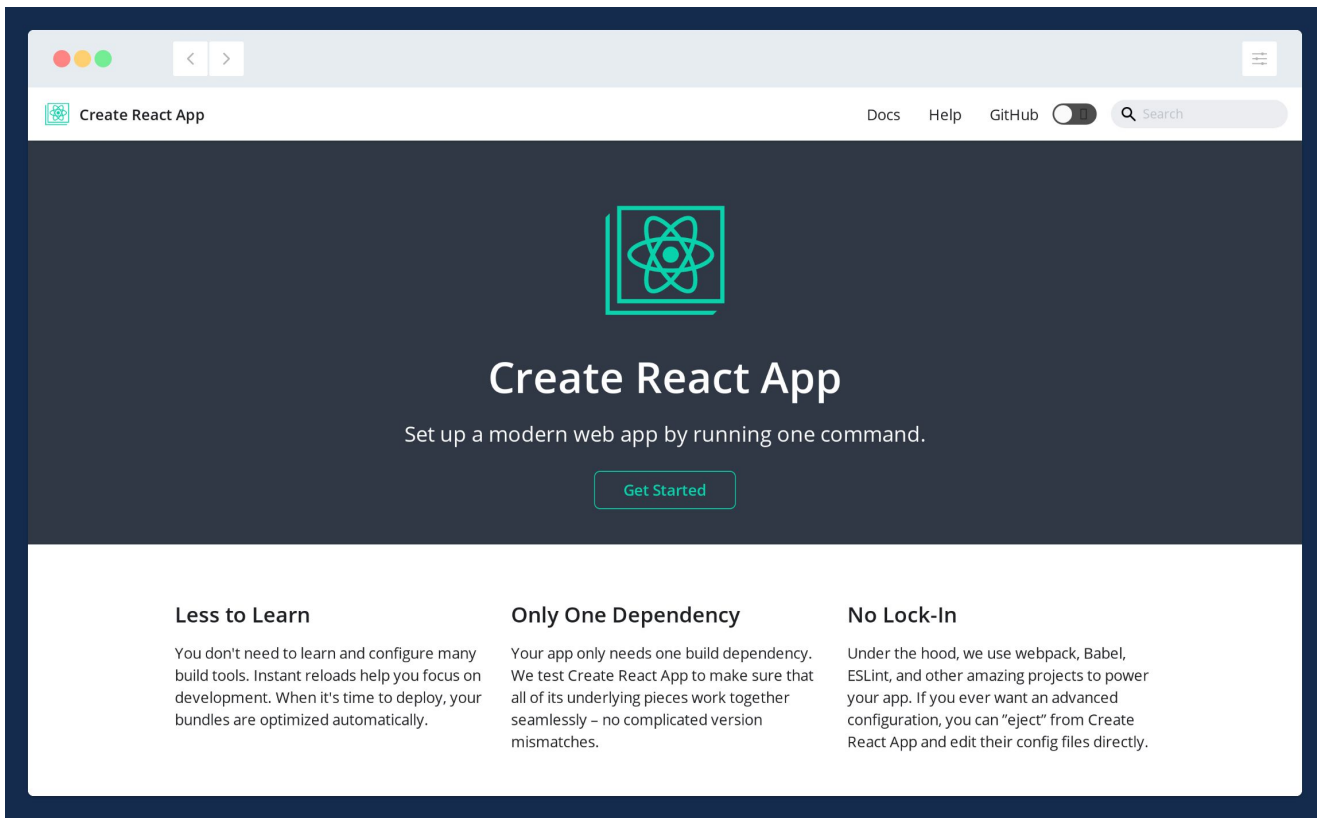
```
1  const notificacoes = 0;
2
3  function montaBadge(num) {
4    if (notificacoes !== 0 && num > 0) {
5      // Adiciona badge
6      // container.appendChild(badge)!! ...
7    }
8    if (notificacoes !== 0 && num > 0) {
9      // Apenas muda o número
10     // badge.innerHTML = num!! ...
11   }
12 }
13 if (notificacoes !== 0 && num !== 0) {
14   // Remove badge
15   // container.removeChild(badge)
16 }
```

```
1  function Badge({ num }) {
2    return (
3      <div id="container">
4        { num > 0 && <div id="badge">{num}</div>
5        <span class="icon">!!</span>
6      </div>
7    );
8  }
```


Babel + Webpack

- Navegador não entende tudo isso
- O **Babel** converte o código para JS de uma forma que o navegador entenda
- O **Webpack** serve para:
 - Arquivo com todo código da aplicação
 - Importar no JS arquivos CSS, imagens e etc
 - Live reload com o **Webpack Dev Server**

Create React App



Create React App - o mais legal

- Configuração do Babel + Webpack
- Estrutura de pastas
- Git inicializado
- Códigos prontos
- Esqueleto do projeto ReactJS (React + React-Dom)

Extensão

[Página inicial](#) > [Extensões](#) > React Developer Tools



React Developer Tools

Remover do Google Chrome

Oferecido por: Facebook

★★★★★ 1.265 | [Ferramentas do desenvolvedor](#) | 2.000.000+ usuários

Visão Geral


Comentários


Suporte

Itens Relacionados



E no final?

 Create React App

Docs Help GitHub 

Welcome

Getting Started

Development

Styles and Assets

Building your App

Installing a Dependency

Importing a Component

Using Global Variables

Adding Bootstrap

Adding Flow

Adding TypeScript

Adding Relay

Adding a Router

Environment Variables

Making a Progressive Web App

Creating a Production Build

Testing

Back-End Integration

Deployment

Advanced Usage

Support

Creating a Production Build

`npm run build` creates a `build` directory with a production build of your app. Inside the `build/static` directory will be your JavaScript and CSS files. Each filename inside of `build/static` will contain a unique hash of the file contents. This hash in the file name enables [long term caching techniques](#).

When running a production build of freshly created Create React App application, there are a number of `.js` files (called *chunks*) that are generated and placed in the `build/static/js` directory:

```
main.[hash].chunk.js
```

- This is your *application code*. `App.js`, etc.

```
[number].[hash].chunk.js
```

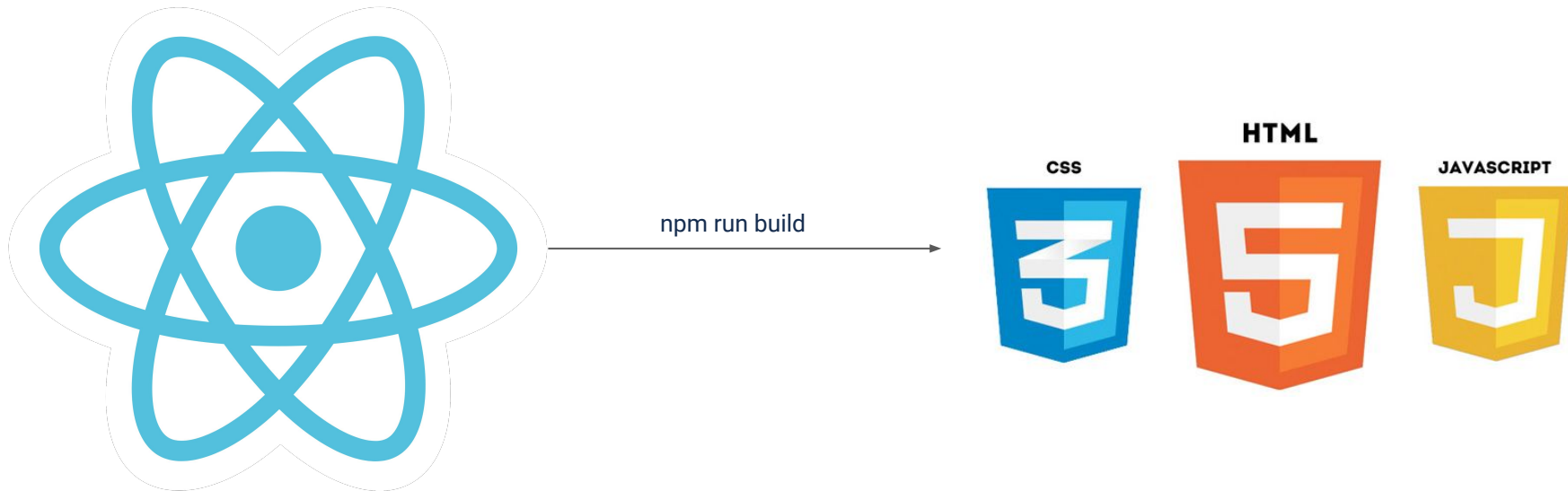
- These files can either be *vendor* code, or [code splitting chunks](#). *Vendor* code includes modules that you've imported from within `node_modules`. One of the potential advantages with splitting your *vendor* and *application* code is to enable [long term caching techniques](#) to improve application loading performance. Since *vendor* code tends to change less often than the actual *application* code, the browser will be able to cache them separately, and won't re-download them each time the app code changes.

```
runtime-main.[hash].js
```

- This is a small chunk of [webpack runtime](#) logic which is used to load and run your application. The contents of this will be embedded in your `build/index.html` file by default to save an additional network request. You can opt out of this by specifying `INLINE_RUNTIME_CHUNK=false` as documented in our [advanced configuration](#), which will load this chunk instead of embedding it in your `index.html`.

Static File Caching
Profiling

E no final?



Deploy

[Docs](#)[Help](#)[GitHub](#)

- Welcome
- Getting Started
- Development
- Styles and Assets
- Building your App
- Testing
- Back-End Integration
 - Proxying in Development
 - Fetching Data
 - Integrating with an API
 - Title & Meta Tags
- Deployment
 - Deployment
- Advanced Usage
 - Custom Templates
 - Can I Use Decorators?
 - Pre-Rendering Static HTML
 - Advanced Configuration
 - Alternatives to Ejecting
- Support

Deployment

`npm run build` creates a `build` directory with a production build of your app. Set up your favorite HTTP server so that a visitor to your site is served `index.html`, and requests to static paths like `/static/js/main.<hash>.js` are served with the contents of the `/static/js/main.<hash>.js` file. For more information see the [production build](#) section.

Static Server

For environments using [Node](#), the easiest way to handle this would be to install [serve](#) and let it handle the rest:

```
npm install -g serve
serve -s build
```

The last command shown above will serve your static site on the port **5000**. Like many of [serve](#)'s internal settings, the port can be adjusted using the `-l` or `--listen` flags:

```
serve -s build -l 4000
```

Run this command to get a full list of the options available:

```
serve -h
```

Other Solutions

You don't necessarily need a static server in order to run a Create React App project in production. It also works well when integrated into an existing server side app.

Static Server

Other Solutions

Serving Apps with Client-Side Routing

Building for Relative Paths

Serving the Same Build from Different Paths

Customizing Environment Variables for Arbitrary Build Environments

AWS Amplify

Azure

Firebase

GitHub Pages

Step 1: Add `homepage` to `package.json`

Step 2: Install `gh-pages` and add `deploy` to `scripts` in `package.json`

Step 3: Deploy the site by running `npm run deploy`

Step 4: For a project page, ensure your project's settings use `gh-pages`

Step 5: Optionally, configure the domain

Notes on client-side routing

Troubleshooting

Heroku

Resolving Heroku Deployment Errors

Netlify

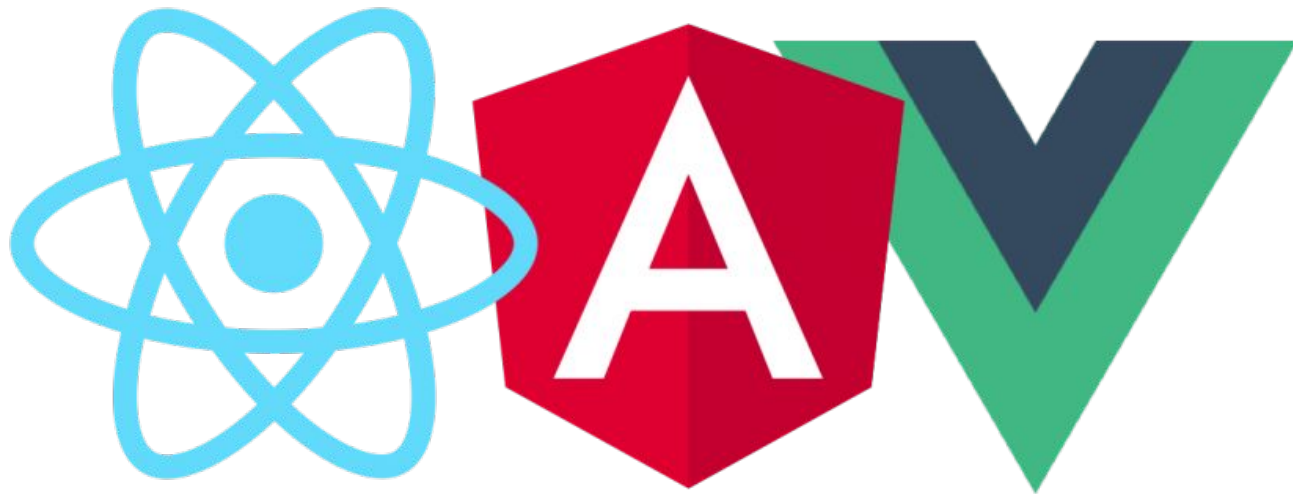
ZEIT Now

Step 1: Installing Now CLI

Step 2: Deploying

Render

Outros: Angular e Vue



Material Extra

- <https://www.youtube.com/watch?v=HN1UjzRSdBk>
- https://www.youtube.com/watch?v=x-4z_u8LcGc&t=2079s
- <https://www.youtube.com/playlist?list=PLBbHLUbqqCrTwIrdix6kl84m4OPE0JexR>
- <https://www.youtube.com/watch?v=H91DhKPjhPk>
- <https://www.youtube.com/watch?v=vyl7JzhdOh4&t=340s>
- <https://www.youtube.com/watch?v=SE76RtsU1VE>
- https://www.youtube.com/playlist?list=PLBNBxpMAbyhUwX2Nh3kbBaBb_2lmtVjTS
- <https://www.youtube.com/watch?v=v1KLEXJFu3A>