SPA's com

ASP. NET Core + React

by José Barbosa | @kidchenko

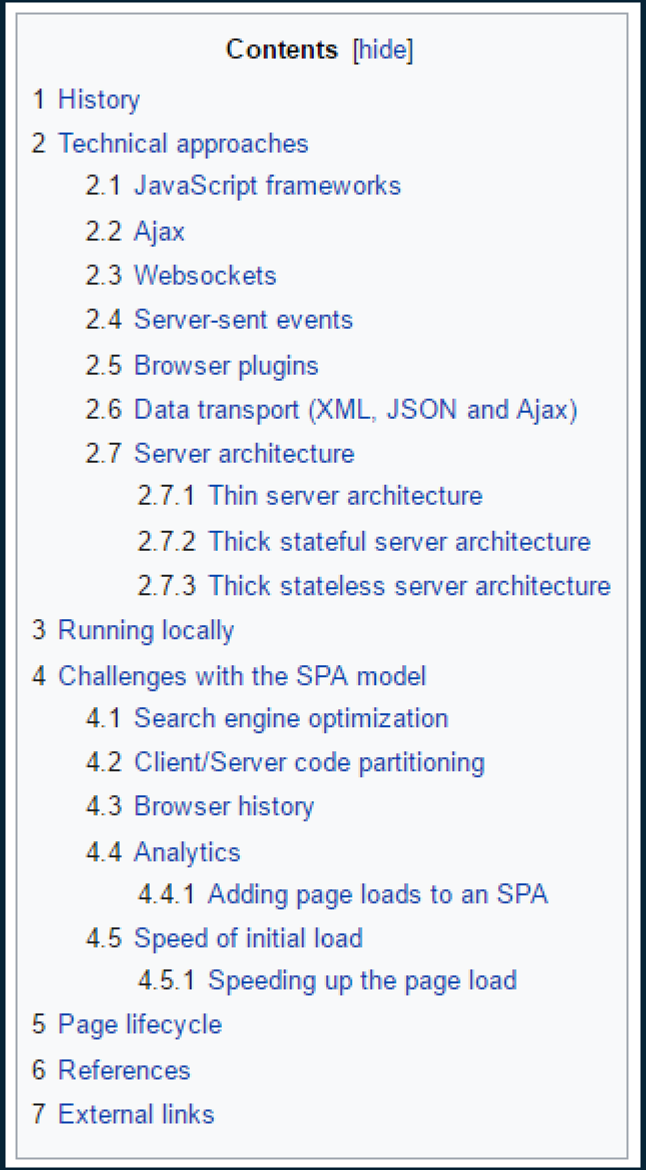
Single

Page

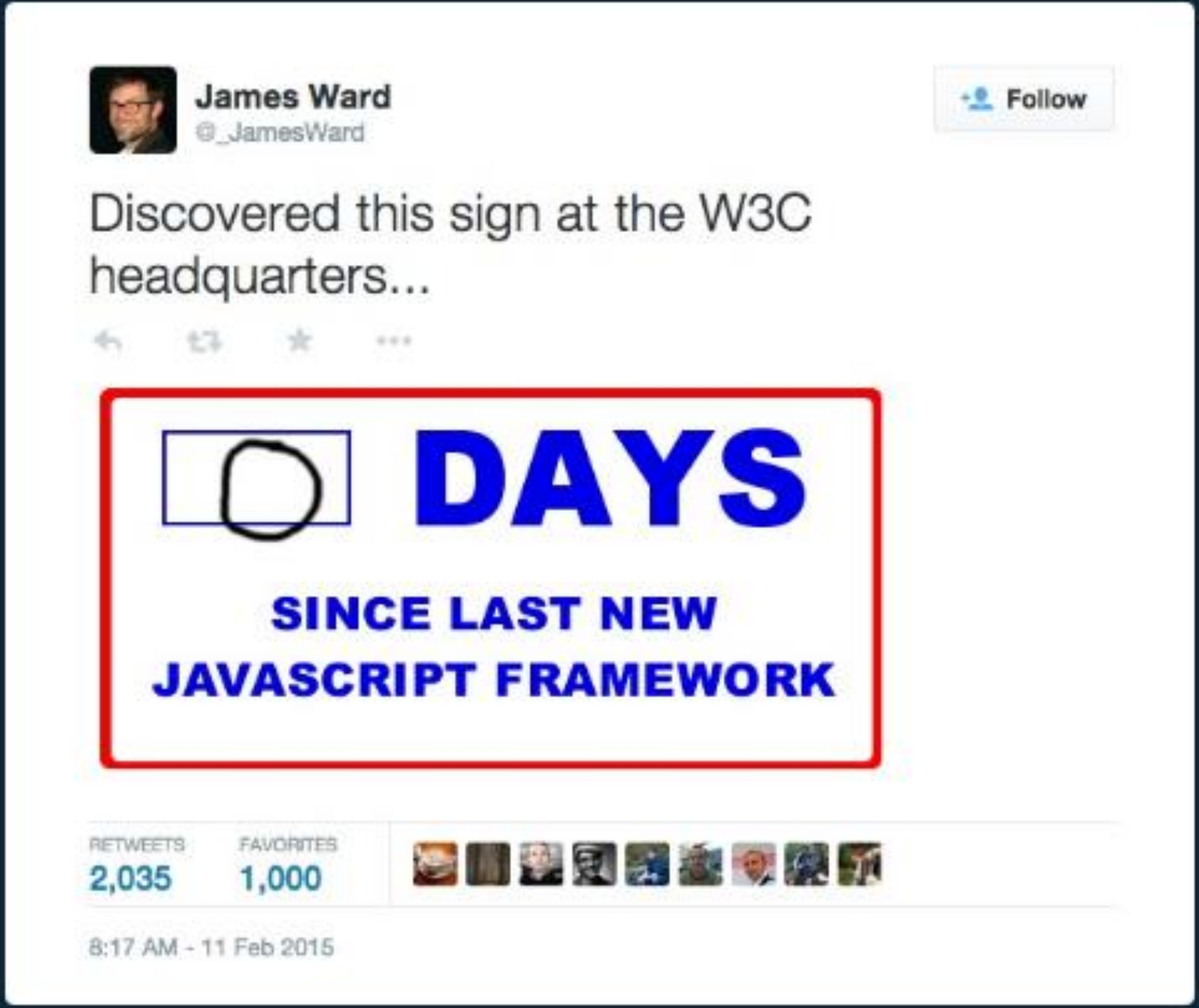
Application

“…In an SPA, either all necessary code – HTML, JavaScript, and CSS – is retrieved with a single page load, or the appropriate resources are dynamically loaded and added to the page as necessary, usually in response to user actions. The page does not reload at any point in the process…”

SPA é uma “aplicação de uma única página”!?

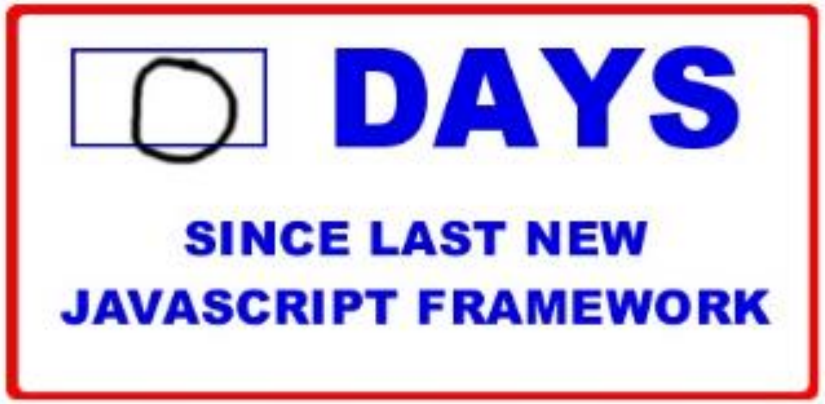


Contents [hide]



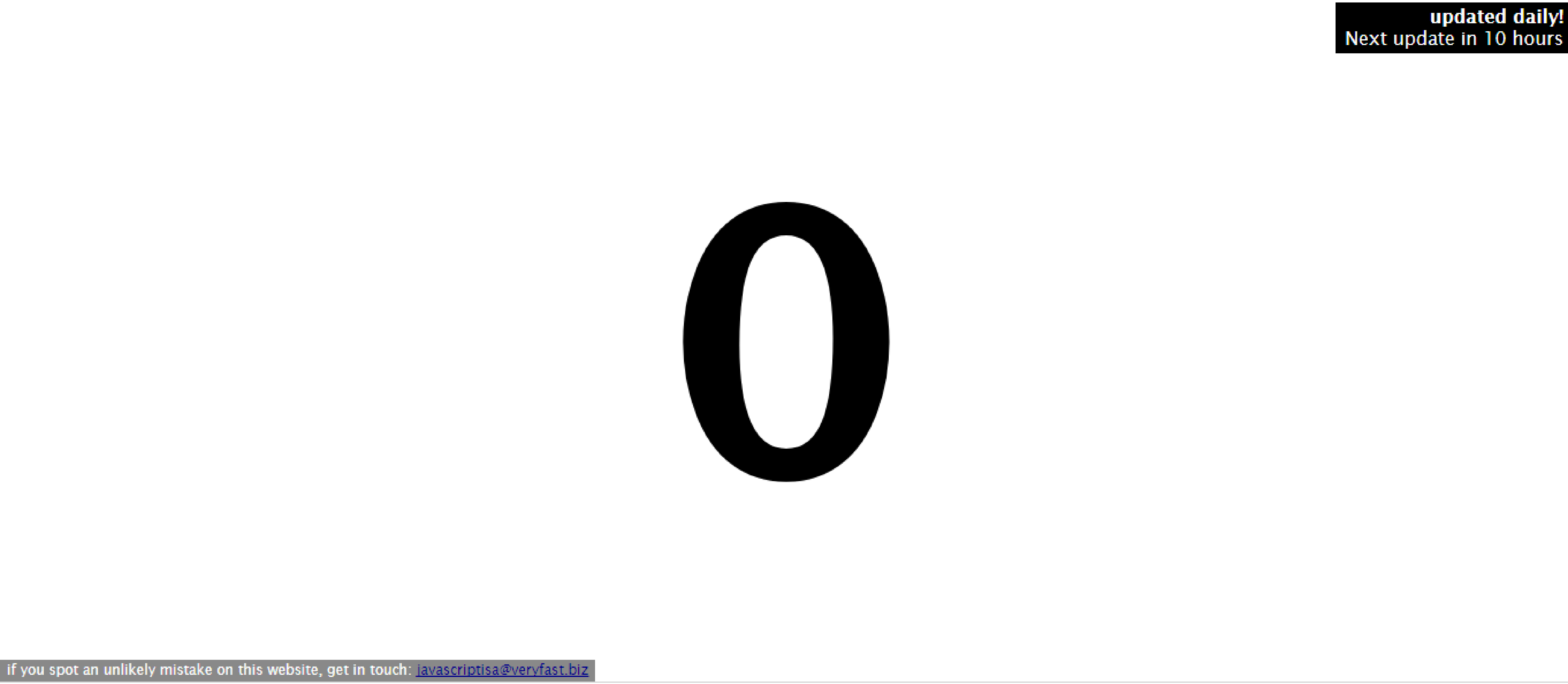
James Ward
@_JamesWard

Discovered this sign at the W3C headquarters...



RETWEETS 2,035 FAVORITES 1,000

8:17 AM - 11 Feb 2015



updated daily!
Next update in 10 hours

0

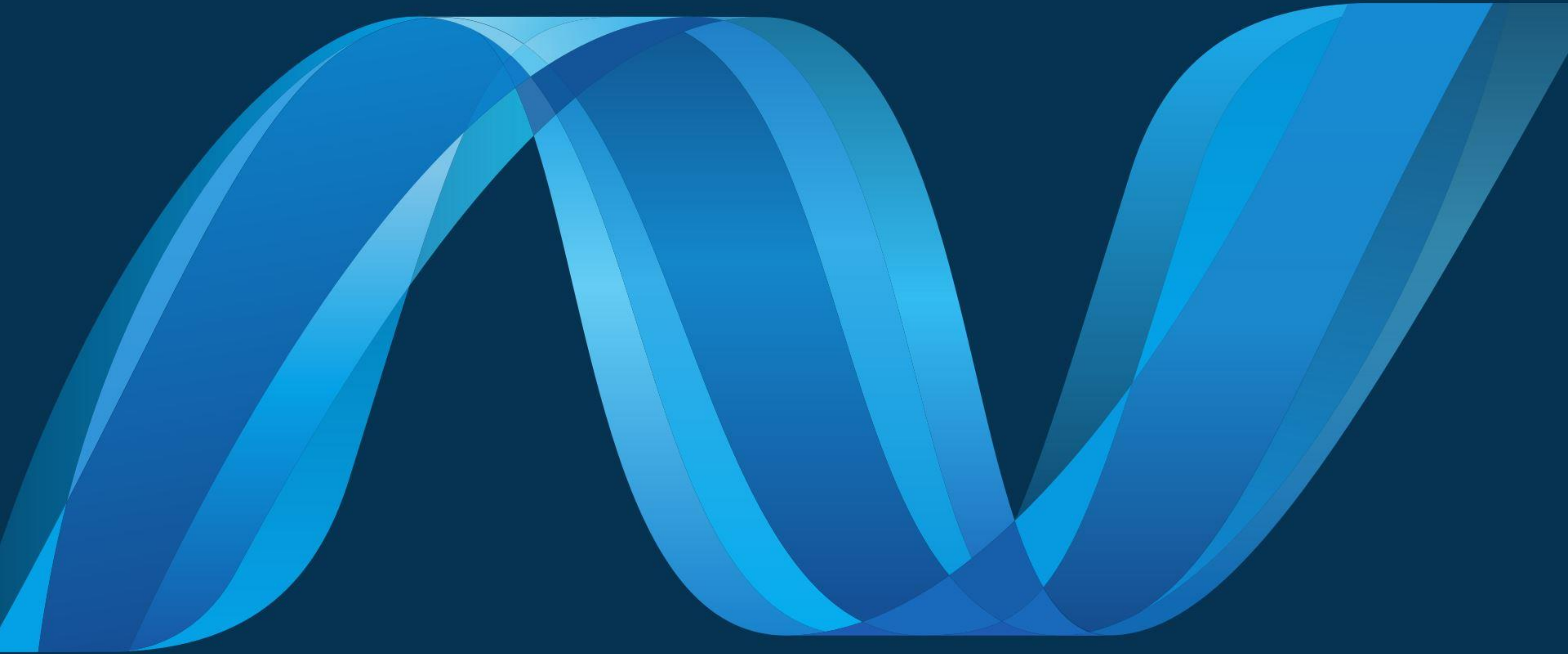
if you spot an unlikely mistake on this website, get in touch: javascriptisa@veryfast.biz

https://dayssincelastjavascriptframework.com/

ember

React

A JAVASCRIPT LIBRARY FOR BUILDING USER INTERFACES

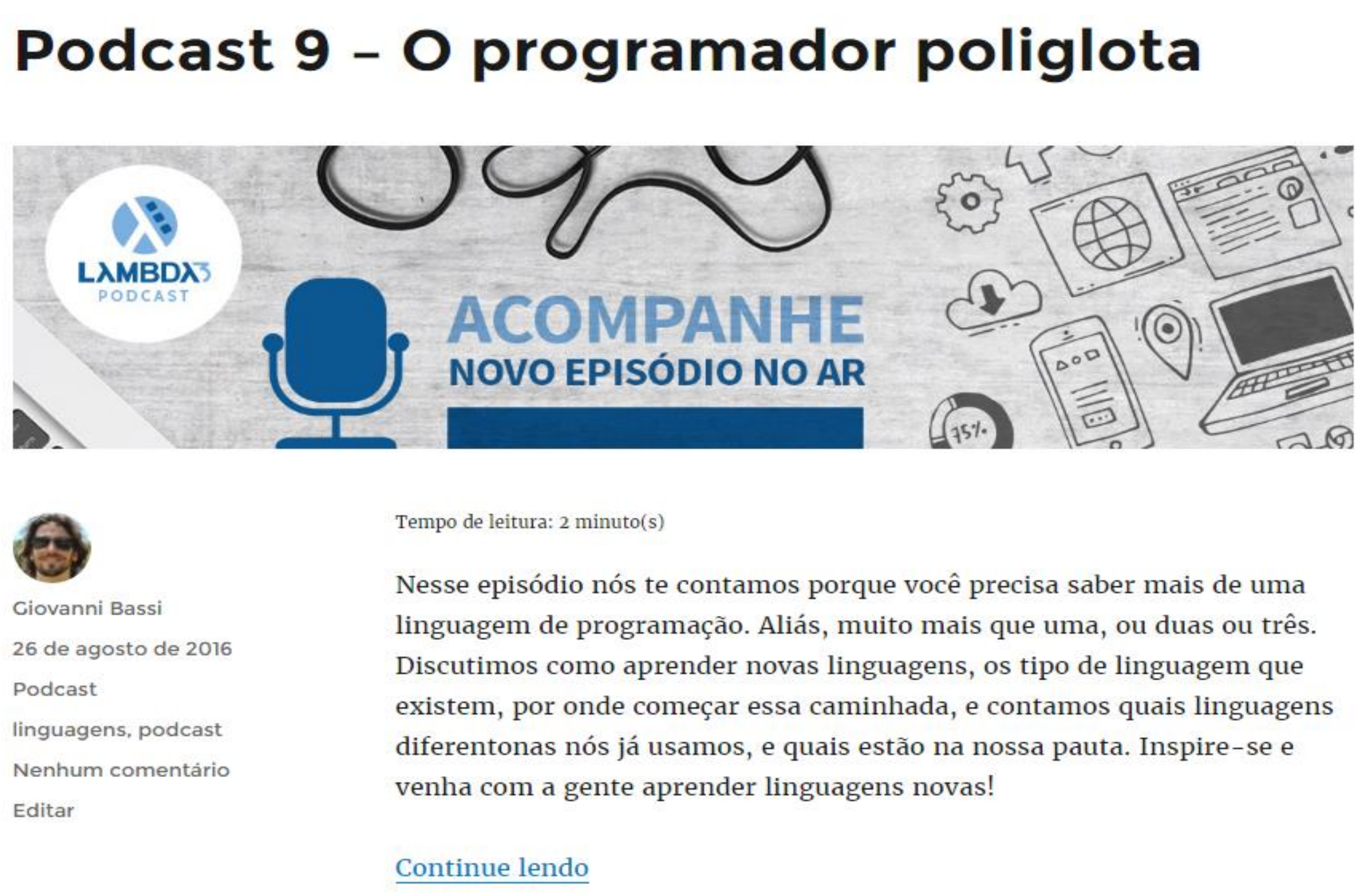


node JS®

Software Engineer;
Agilista;
Empreendedor digital;
Apaixonado por JS;
Ex-Lambda3

@kidchenko

github.com/kidchenko



Podcast 9 – O programador poliglota

Giovanni Bassi
26 de agosto de 2016
Podcast
linguagens, podcast
Nenhum comentário
Editar

Tempo de leitura: 2 minuto(s)

Nesse episódio nós te contamos porque você precisa saber mais de uma linguagem de programação. Aliás, muito mais que uma, ou duas ou três. Discutimos como aprender novas linguagens, os tipo de linguagem que existem, por onde começar essa caminhada, e contamos quais linguagens diferentonas nós já usamos, e quais estão na nossa pauta. Inspire-se e venha com a gente aprender linguagens novas!

Continue lendo

Acompanhe o podcast da Lambda3:

WE'RE HIRING!

ite

DI
VER
SIDA
DE
NA
TI
EU
APOIO

SPA's com

ASP. NET Core + React

What framework do I need?

PureMVC

Epitome

Ractive.js

Backbone

SAPUI5

Atma.js

Polymer

Dojo

Exoskeleton

YUI

cujoJS

Stapes

Maria

DeftJS + ExtJS

KnockoutJS

React

PlastronJS

Kendo UI

Vue.js

AngularJS

ComponentJS

Olives

Agility.js

Dijon

CanJS

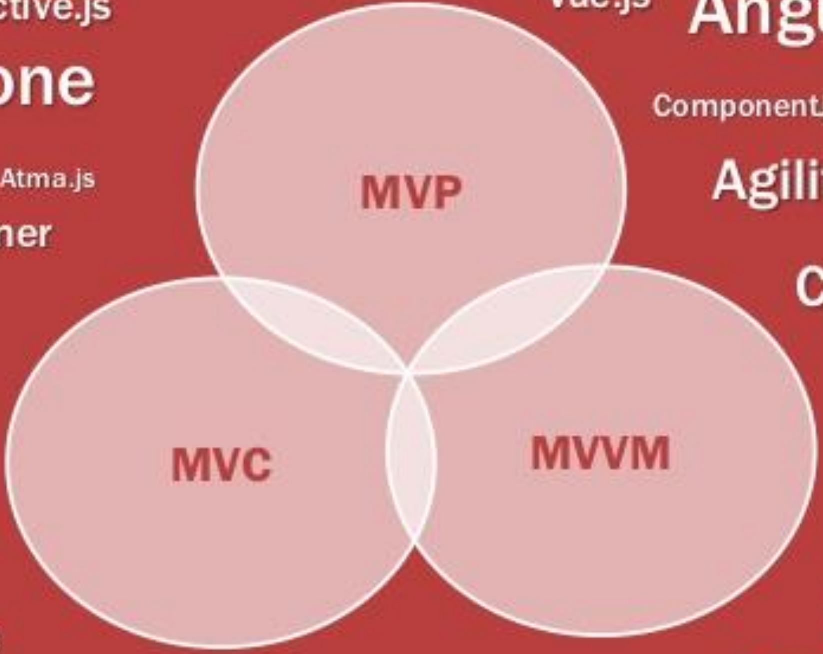
rAppid.js

Montage

DUEL

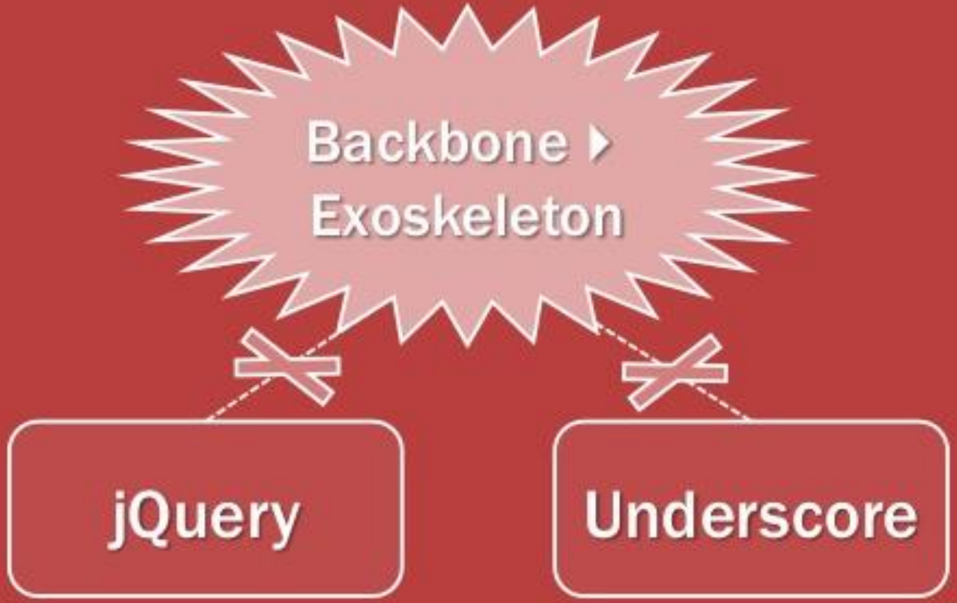
Sammy.js

Ember.js



You have to know then exactly what the framework does

Not really!



ReactJS.NET

REACT ♥ C# AND ASP.NET MVC

ReactJS.NET ma easier to us acebook's React and JSX from C# and other .NET languages, focu g specifically on ASP.NET MVC (a it also wo in other environments). It supports both ASP.NET 4 (with MVC 4 or and ASP.NET Core MVC. It is cross-p for nd can r on Linu .NET ook at see h easy it is to get started with Reac nd R tJS.NE

Latest news: Rea S.NET - .N Core d lots of sn twe (October 2016)

Nops!

On-the-fly JSX to JavaScript compilation

Simply name your file with a `.jsx` extension and link to the file via a `script` tag.

The files will automatically be

```
// /Scripts/HelloWorld.jsx
var HelloWorld = React.createClass({
  render: function() {
    return <div>Hello world!</div>;
  }
```

Traditional MV* Frameworks

Separation of Concerns

Model	Controller	ModelView	View
Data	Display logic (JavaScript)	Tailored Data	Templates (HTML + custom extensions)

Controller, ModelView, and View **are** coupled:
when you change one, you often have to change the others

Angular JS - MV*

Model

```
var myDetails = {
        "firstname" : "Srinivas",
        "lastname": "Nagaram",
        "title" : "UI Developer"
};
```

Data

View

```
<tr>
    <td>{{myDetails.firstname}}</td>
    <td>{{myDetails.lastname}}</td>
    <td>{{myDetails.title}}</td>
</tr>
```

HTML

* Whatever

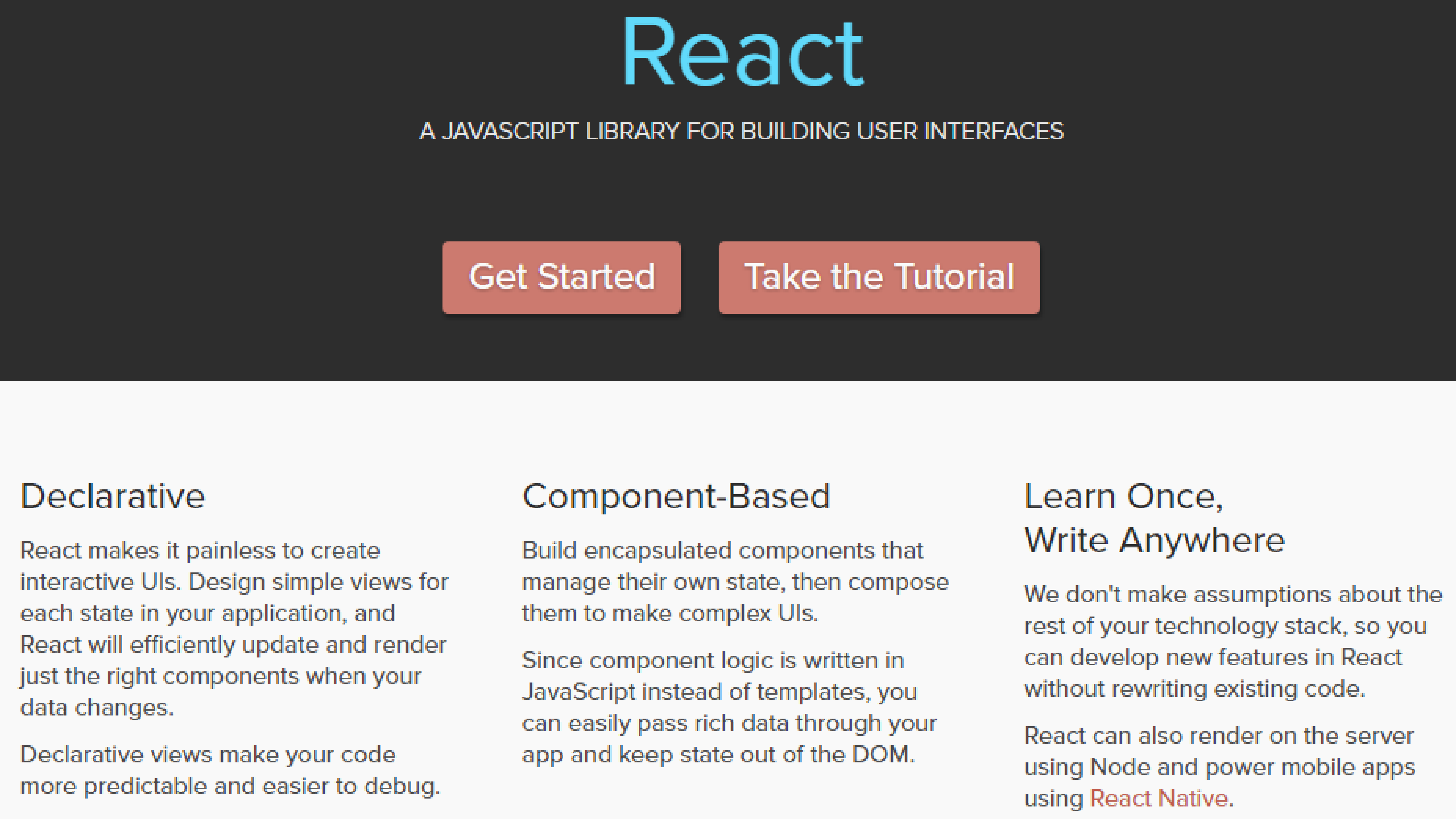
Angular JS does not follow any one of below methods . For this reason we will refer this framework as MV* implementation.

Controller

ViewModel

Presenter

Logic



React

A JAVASCRIPT LIBRARY FOR BUILDING USER INTERFACES

Declarative

React makes it painless to create interactive UIs. Design simple views for each state in your application, and React will efficiently update and render just the right components when your data changes.

Declarative views make your code more predictable and easier to debug.

Component-Based

Build encapsulated components that manage their own state, then compose them to make complex UIs.

Since component logic is written in JavaScript instead of templates, you can easily pass rich data through your app and keep state out of the DOM.

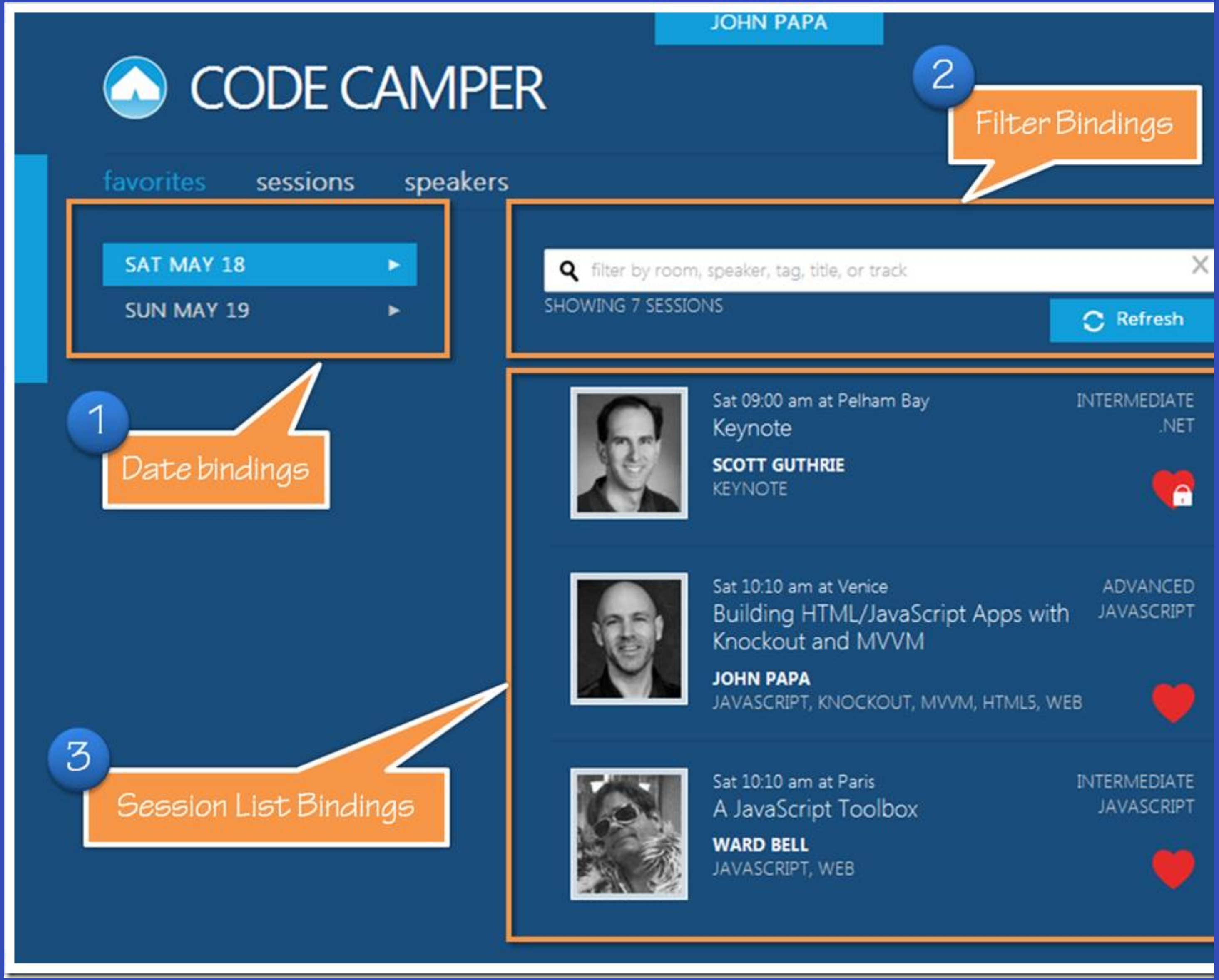
Learn Once, Write Anywhere

We don't make assumptions about the rest of your technology stack, so you can develop new features in React without rewriting existing code.

React can also render on the server using Node and power mobile apps using React Native.

Entendendo o Reactjs

Componentes



Sat 09:00 am at Pelham Bay
Keynote
SCOTT GUTHRIE
KEYNOTE
INTERMEDIATE
.NET

Sat 10:10 am at Venice
Building HTML/JavaScript Apps with Knockout and MVVM
JOHN PAPA
JAVASCRIPT, KNOCKOUT, MVVM, HTML5, WEB
ADVANCED
JAVASCRIPT

Sat 10:10 am at Paris
A JavaScript Toolbox
WARD BELL
JAVASCRIPT, WEB
INTERMEDIATE
JAVASCRIPT

UNIDIRECTIONAL
data flow

```html
<!DOCTYPE html>
<html>
<head>
<script src="https://fb.me/react-15.1.0.min.js"></script>
<script src="https://fb.me/react-dom-15.1.0.min.js"></script>
</head>
<body>
<script>
     ReactDOM.render(React.createElement(
                'div', null, 'Hi React!'), document.body);
</script>
</body>
</html>
```

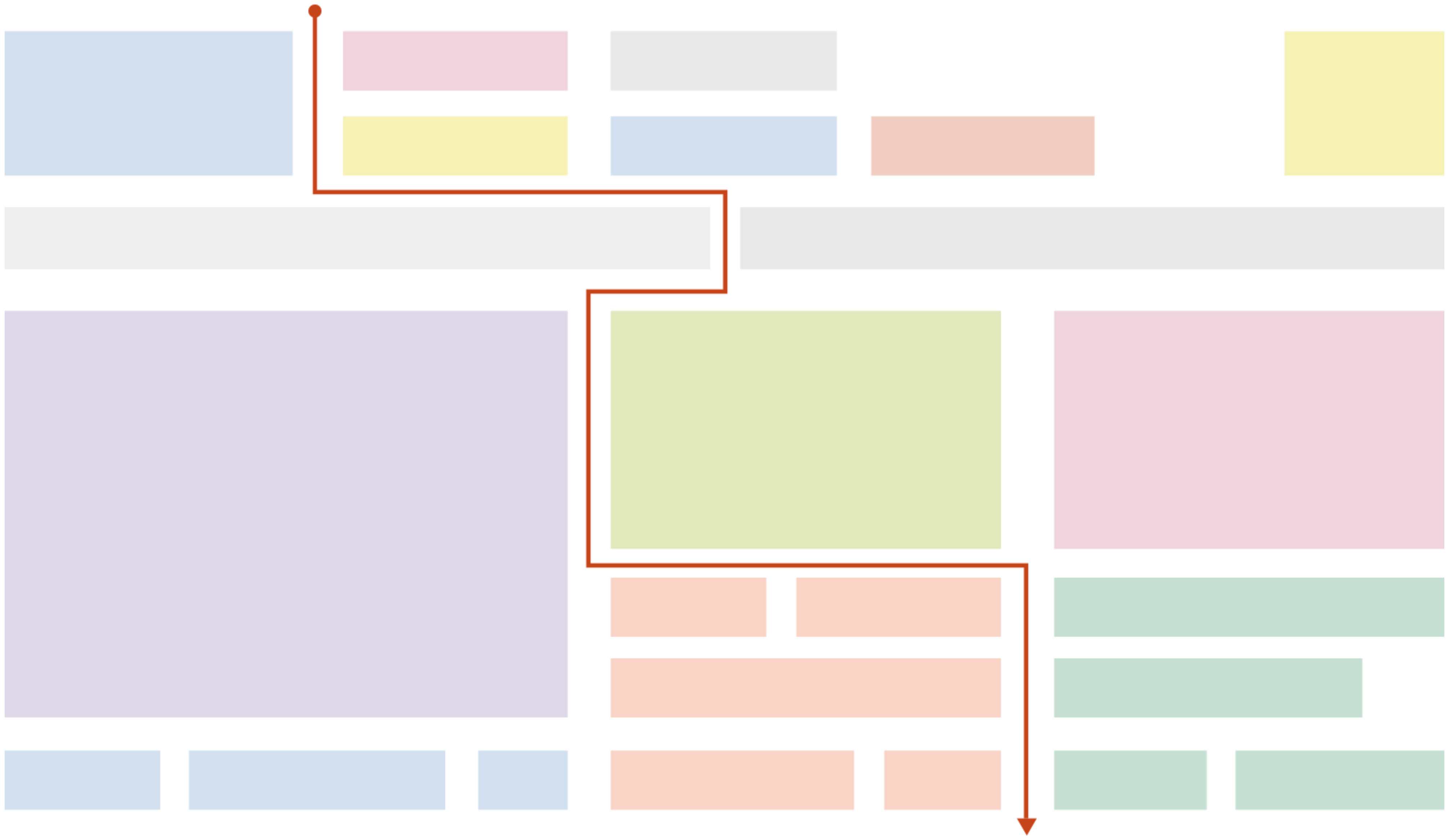
```html
<!DOCTYPE html>
<html>
<head>
<script src="https://fb.me/react-15.1.0.min.js"></script>
<script src="https://fb.me/react-dom-15.1.0.min.js"></script>
</head>
<body>
<script>
     ReactDOM.render(React.createElement(
           'div', null, 'Hi React!'), document.body);
</script>
</body>
</html>
```

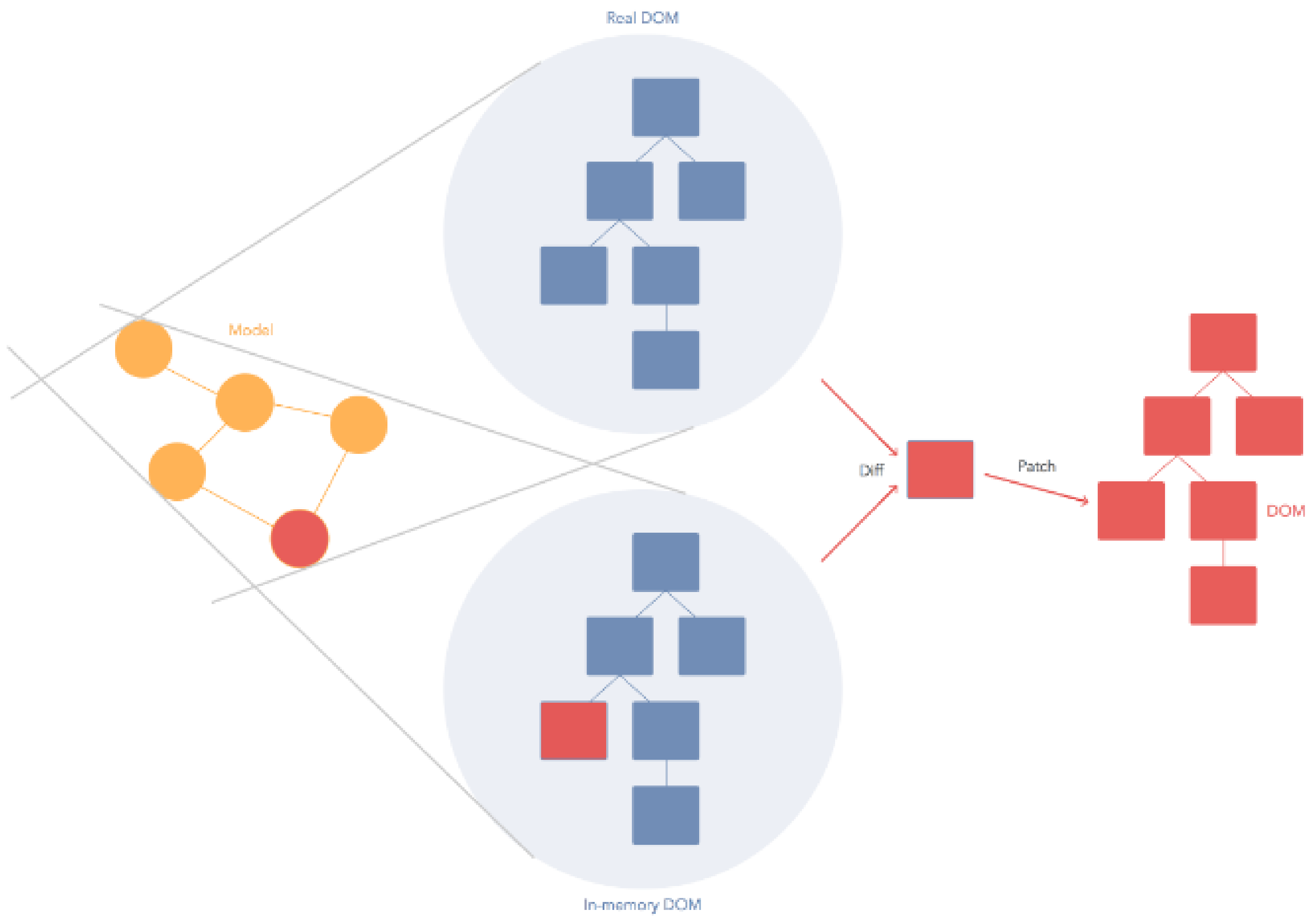
JSX

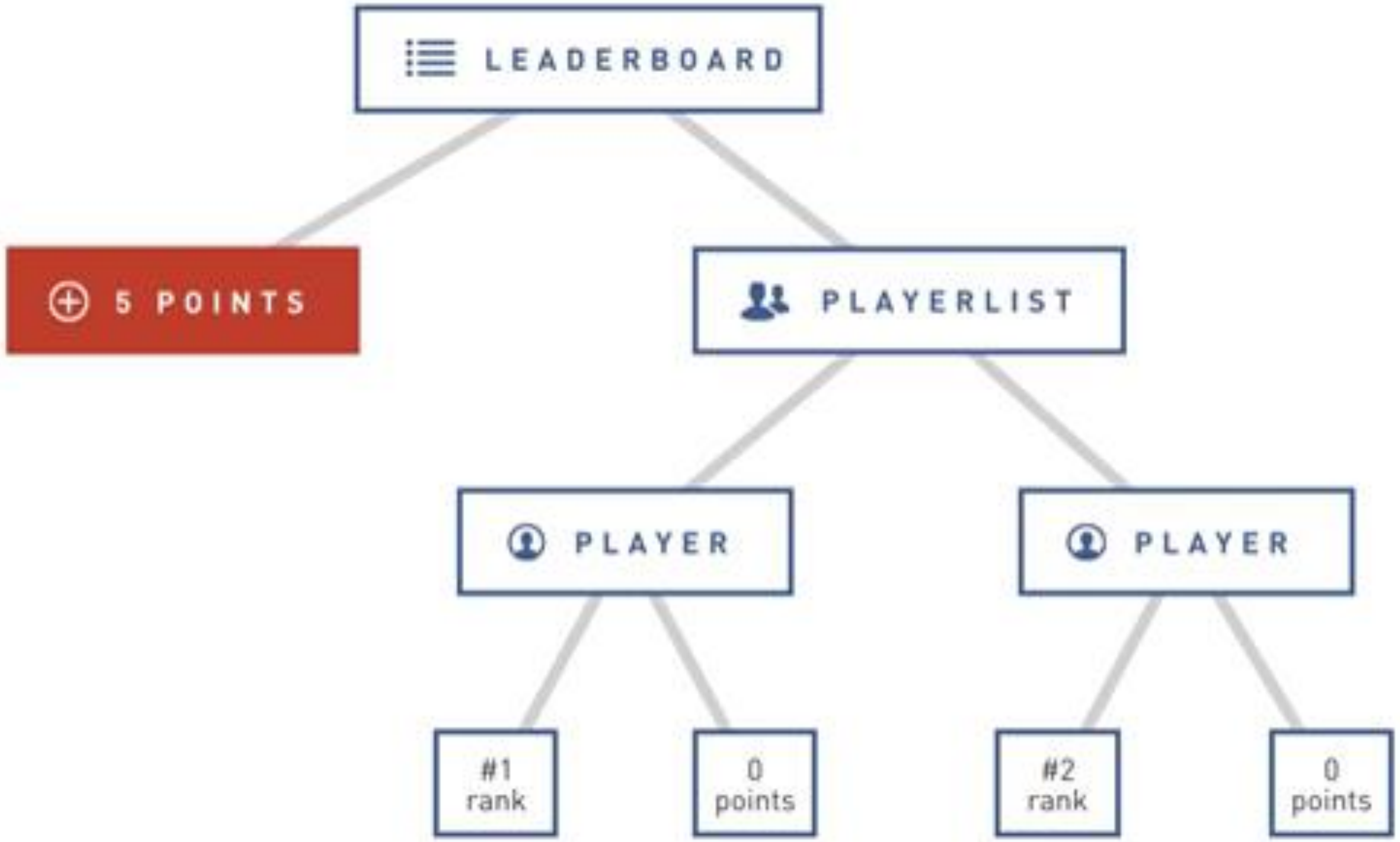
```
<script type="text/jsx">
     var Hi = React.createClass({
          render: function() {
          return (<div>
                    <h1>Hi from react and jsx :)</h1>
                    <p>This is some text</p>
               </div>)
          }
     });
     ReactDOM.render(<Hi />,
document.getElementById('container'));
</script>
```

```
<script type="text/jsx">
     var Hi = React.createClass({
          render: function() {
          return (<div>
                    <h1>Hi from react and jsx :)</h1>
                    <p>This is some text</p>
               </div>)
          }
     });
     ReactDOM.render(<Hi />,
document.getElementById('container'));
</script>
```

Virtual DOM



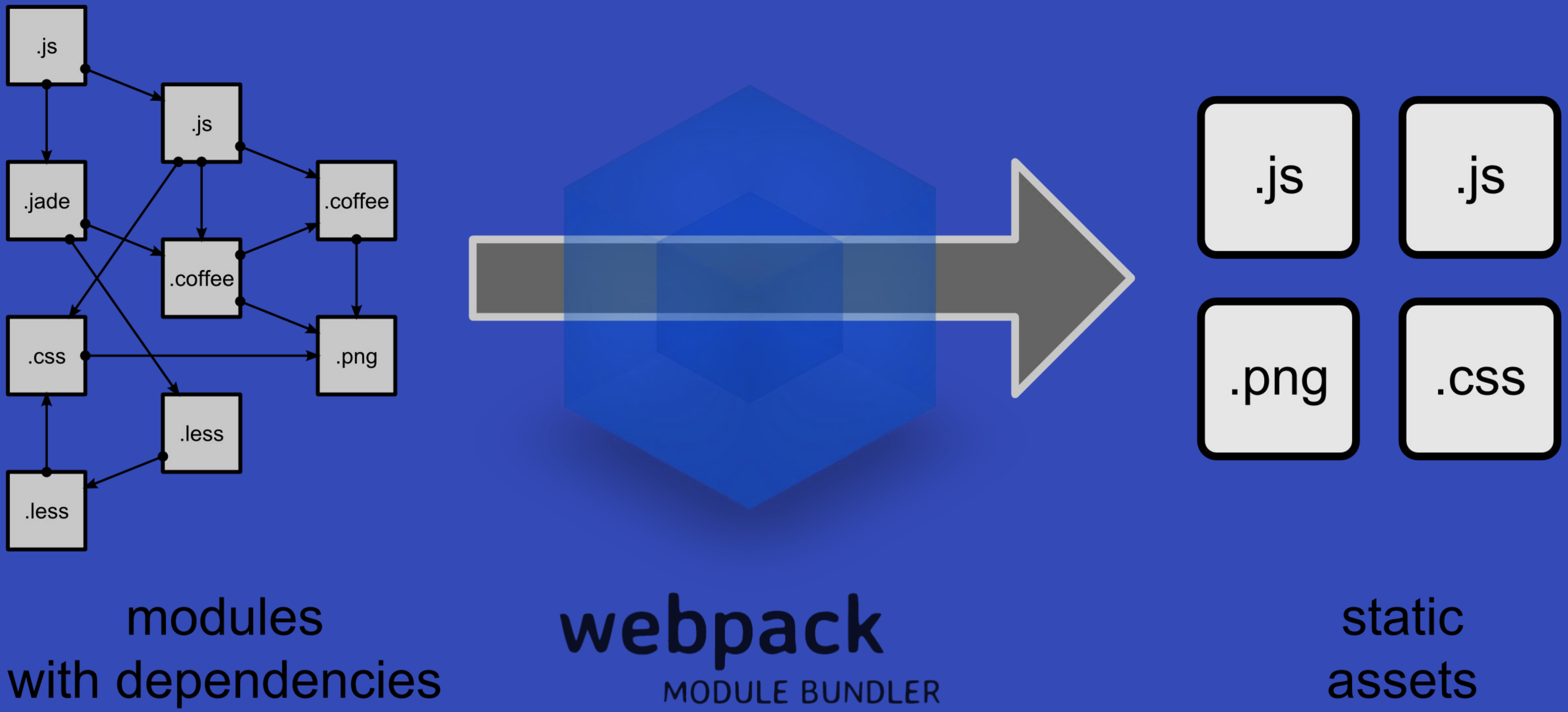

Real DOM
Model
Diff
Patch
DOM
In-memory DOM


LEADERBOARD
5 POINTS
PLAYERLIST
PLAYER
PLAYER
#1 rank
0 points
#2 rank
0 points

Composição

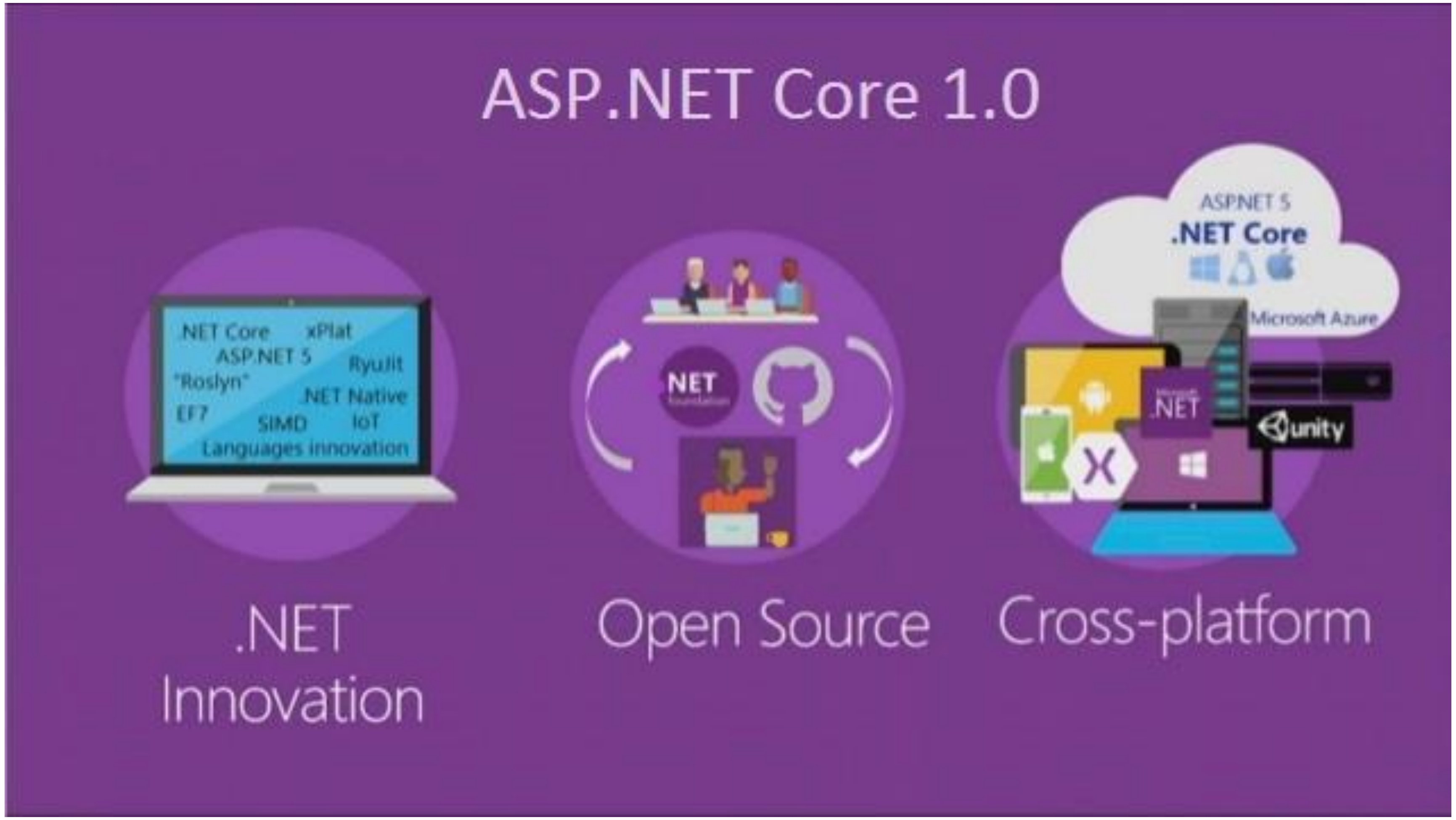
```
var HomePage = React.createClass({
    render: function () {
        return (
            <div>
                <Header />
                <SearchBar />
                <EmployeeList />
            </div>
        );
    }
});
```

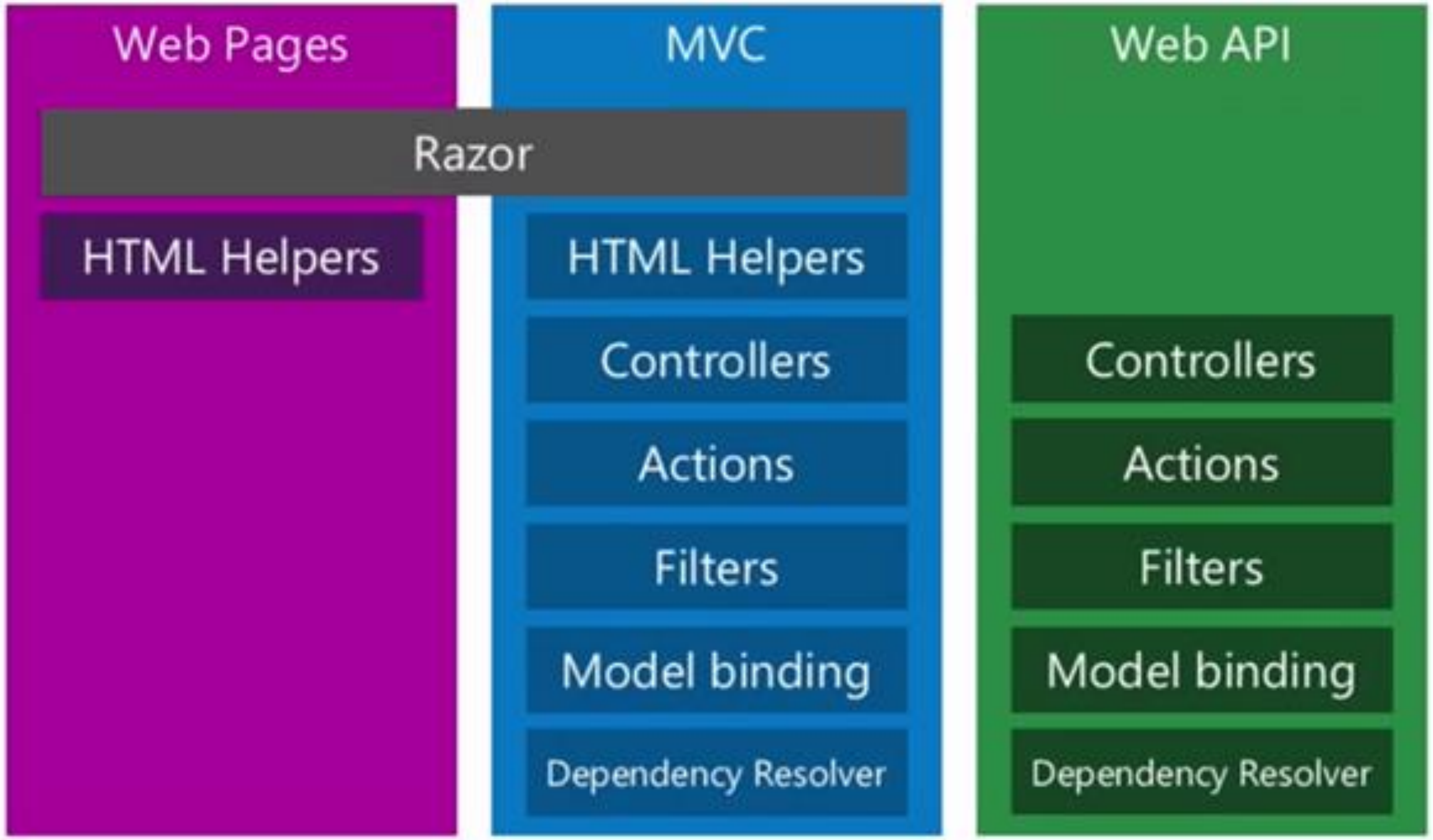
Tooling


.js
.js
.jade
.coffee
.coffee
.css
.png
.less
.less
modules
with dependencies
webpack
MODULE BUNDLER
.js
.js
.png
.css
static
assets

R
m
e

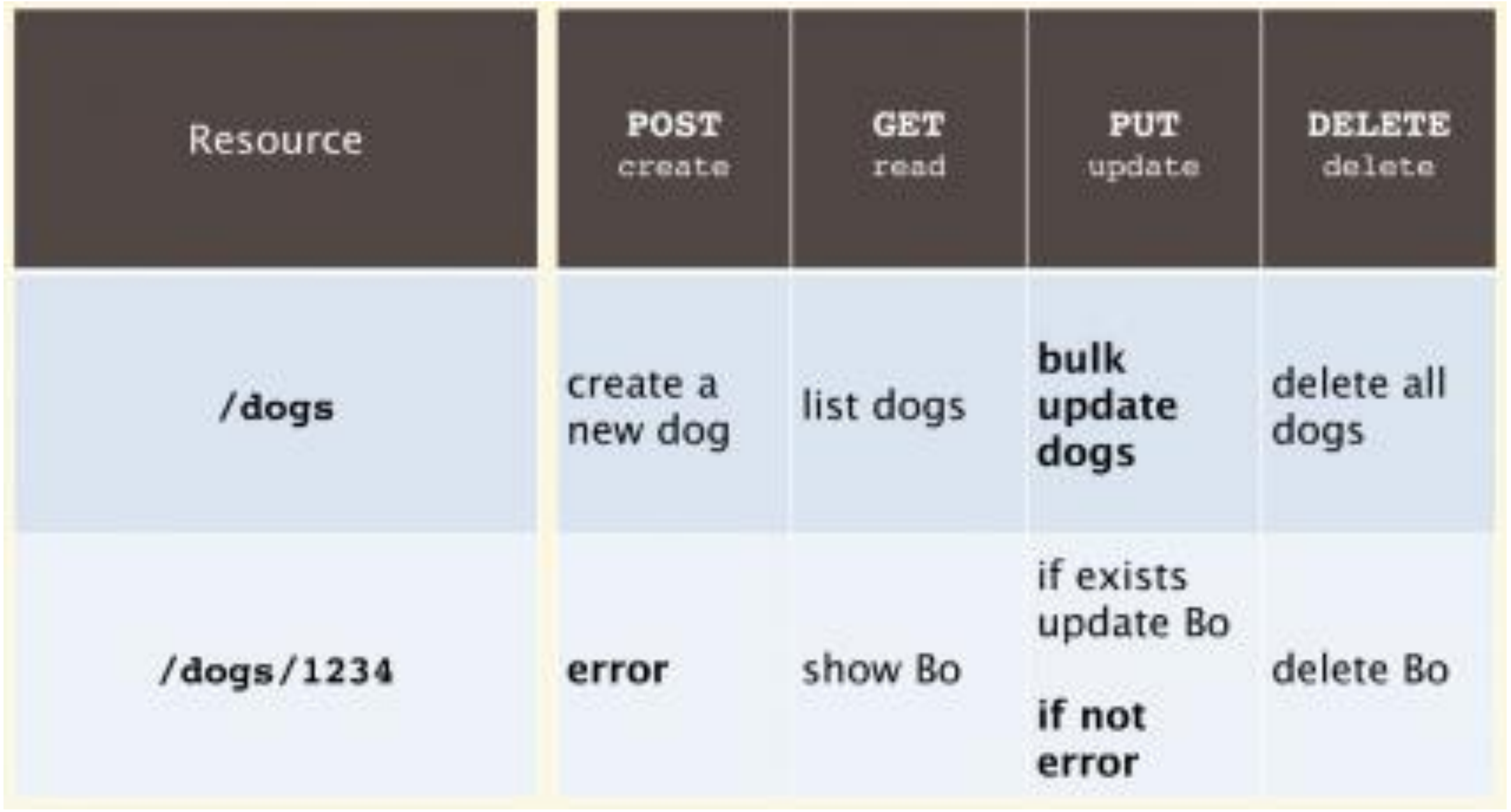
Entendendo o ASP .NET Core


ASP.NET Core 1.0
.NET Core
xPlat
ASP.NET 5
RyuJit
"Roslyn"
.NET Native
EF7
SIMD
IoT
Languages innovation
NET
ASP.NET 5
.NET Core
Microsoft Azure
.NET
unity
.NET Innovation
Open Source
Cross-platform


Web Pages
MVC
Web API
Razor
HTML Helpers
HTML Helpers
Controllers
Actions
Filters
Model binding
Dependency Resolver
Controllers
Actions
Filters
Model binding
Dependency Resolver

Controllers

{ REST }



Resource	**POST** create	**GET** read	**PUT** update	**DELETE** delete
/dogs	create a new dog	list dogs	**bulk update dogs**	delete all dogs

/dogs/1234	**error**	show Bo	if exists update Bo
			
			if not error
				delete Bo

Por quê Reactjs é útil para o ASP .Net Core?

Directory Layout

```
.
├── /.vscode/                   # Visual Studio Code settings
├── /build/                     # The folder for compiled output
├── /client/                    # Client-side app (frontend)
│   ├── /components/            # Common or shared UI components
│   ├── /utils/                 # Helper functions and utility classes
│   ├── /views/                 # UI components for web pages (screens)
│   ├── history.js              # HTML5 History API wrapper used for navigation
│   ├── main.js                 # Entry point that bootstraps the app
│   ├── router.js               # Lightweight application router
│   ├── routes.json             # The list of application routes
│   └── store.js                # Application state manager (Redux)
├── /client.test/               # Unit and integration tests for the frontend app
├── /docs/                      # Documentation to the project
├── /public/                    # Static files such as favicon.ico etc.
│   ├── robots.txt              # Instructions for search engine crawlers
│   └── ...                     # etc.
├── /server/                    # Web server and data API (backend)
│   ├── /Controllers/           # ASP.NET Web API and MVC controllers
│   ├── /Models/                # Entity Framework models (entities)
│   ├── /Views/                 # Server-side rendered views
│   ├── appsettings.json        # Server-side application settings
│   ├── Startup.cs              # Server-side application entry point
│   └── web.config              # Web server settings for IIS
├── /server.test/               # Unit and integration tests for the backend app
|── jsconfig.json               # Visual Studio Code settings for JavaScript
|── package.json                # The list of project dependencies and NPM scripts
|── run.js                      # Build automation script (similar to gulpfile.js)
└── webpack.config.js           # Bundling and optimization settings for Webpack
```

OBRIGADO!
Dúvidas?

@kidchenko

github.com/kidchenko