



ACADEMY

LAPIDANDO TALENTOS



CONHECENDO O ANGULAR 10

AULA 1 – INTRODUÇÃO E CONCEITOS

Eder Franco @ FPF Tech



CONHECENDO O ANGULAR ~~10~~

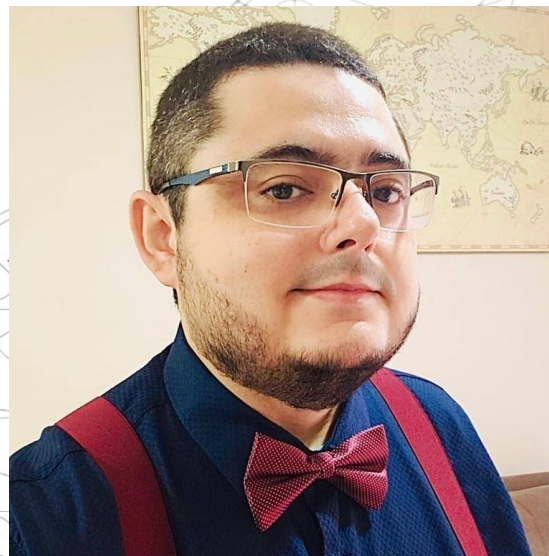
AULA 1 – INTRODUÇÃO E CONCEITOS

Eder Franco @ FPF Tech

Agenda

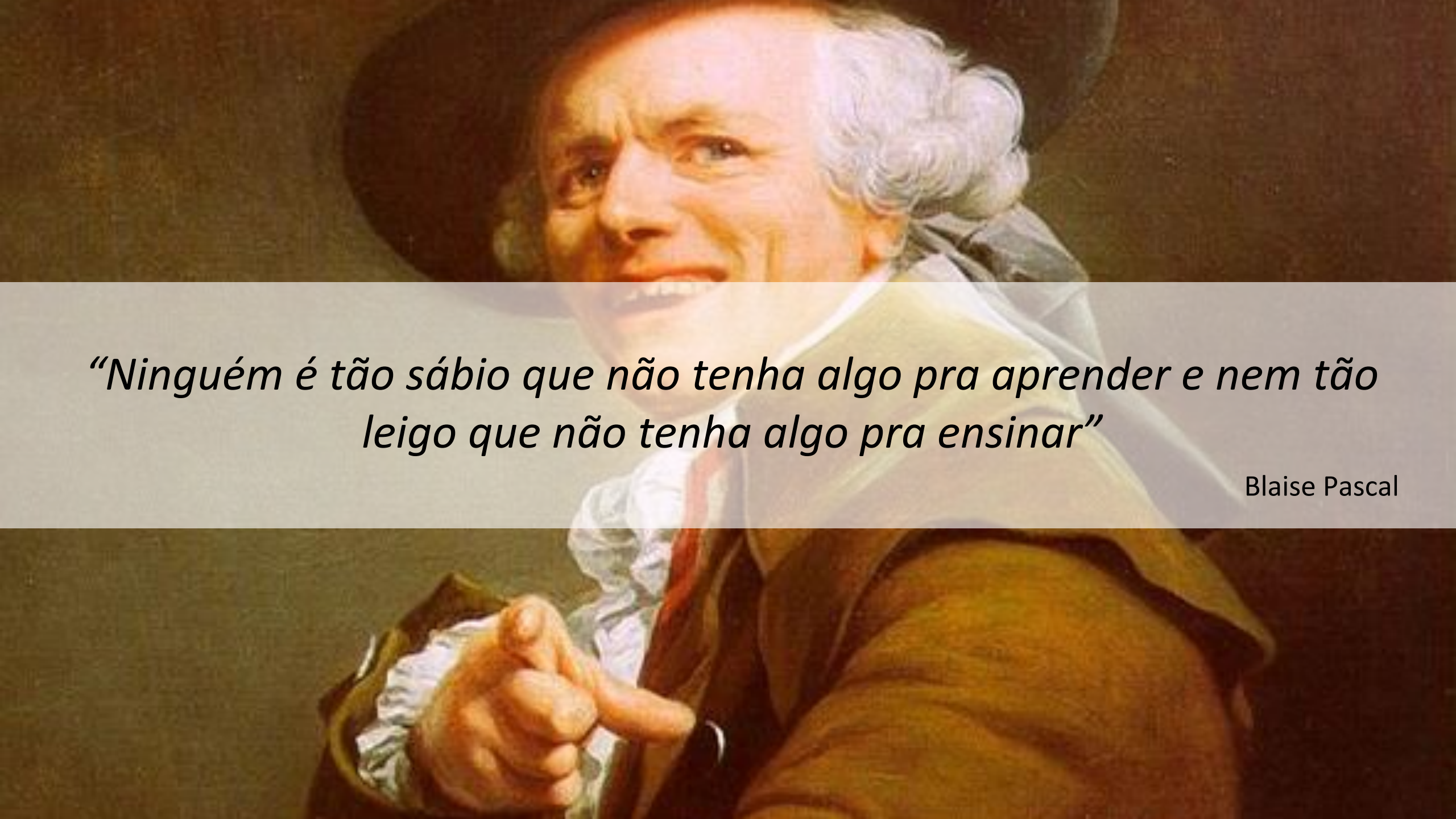
1. Revisão de conceitos sobre a linguagem Javascript;
2. Cenário atual no desenvolvimento front-end: frameworks e ferramentas;
3. Principais recursos das novas versões do ECMAScript:
 - *Arrow functions*;
 - Criação de variáveis e constantes;
 - Resolução de escopo;
 - Classes orientadas a objetos;
 - Interpolação de *strings*;
4. Referências

Quem?



Programador e professor, notívago e viciado em café.
Cristão, nerd, gamer e amigo da vizinhança.

<https://www.linkedin.com/in/efranco23>

A portrait of Blaise Pascal, a French philosopher, mathematician, and scientist. He is depicted from the chest up, wearing a dark hat and a brown coat over a white ruffled shirt and a red cravat. He has a serious expression and is pointing his right index finger towards the viewer. The background is dark and indistinct.

“Ninguém é tão sábio que não tenha algo pra aprender e nem tão leigo que não tenha algo pra ensinar”

Blaise Pascal

Mas antes...

Pré-requisitos:

- Conhecimentos consolidados em desenvolvimento web (HTML e CSS);
- Sólidos conhecimentos em linguagem Javascript e requisições assíncronas;
- Conhecimentos em programação orientada a objetos e modularização;
- Conhecimentos básicos em Angular JS (desejável).

Mas antes...

#ficaadica

- Não tenho estes conhecimentos! E agora?
 - Estude o material dos cursos anteriores de front-end realizados na Academy.
 - Acompanhe atentamente a revisão de conceitos (a seguir).
 - Realize com atenção os exemplos conduzidos em sala de aula (os principais conceitos serão revisados).

Alinhando as expectativas

Neste curso nós vamos:

- Revisar o histórico da linguagem JS e sua importância no mercado atual;
- Conversar sobre os alguns frameworks de JS e suas características;
- Conhecer algumas funcionalidades importantes trazidas pelas novas edição do ECMAScript;

Alinhando as expectativas

Neste curso nós vamos:

- Compreender o que os pré-processadores de Javascript podem fazer por nós;
- Conhecer algumas ferramentas de produtividade em desenvolvimento front-end;
- Conhecer a linguagem Typescript e seus principais componentes;

Alinhando as expectativas

Neste curso nós vamos:

- Entender a evolução do framework AngularJS;
- Entender as principais diferenças entre o AngularJS e o Angular;
- Conhecer a arquitetura e principais componentes do Angular;
- Praticar a utilização do Angular criando uma projeto de SPA consumindo uma API pré-definida;

Alinhando as expectativas

Neste curso nós **não** vamos:

- Aprofundar-se em teorias básicas (conceitos de OO, por exemplo);
- Realizar *benchmark* de ferramentas e frameworks;
- Desvirtuar os tópicos ou desviar o contexto;
- Programar APIs;
- Defender *causas*.



Nice! High Five

Revisão de conceitos sobre JS

Javascript

- Linguagem de script criada pela Netscape em 1996 (Mocha, LiveScript);
- Não tem qualquer relação com JAVA (o nome foi apenas uma jogada comercial);
- Padronizada pela ECMA (European Computers Manufacturers Association);

Javascript

- Os principais navegadores implementam muitas funcionalidades do ECMAScript 6, mas o ECMAScript 5 segue como padrão mais estável e mais utilizado.
- É possível programar com Javascript fora do ambiente de um *browser*, utilizando NodeJS (não será objeto deste curso, mas utilizaremos alguns dos seus recursos e bibliotecas).

Revisão de conceitos

Javascript

- Principais versões:
 - 1996 - JavaScript 1 (Netscape)
 - 1997 – ECMAScript 1
 - 1999 – ECMAScript 2
 - 2000 – ECMAScript 3
 - 2011 – ECMAScript 5.1
 - 2015 – ECMAScript 6 (ES6)
 - 2016 – ECMAScript 7
 - 2017 – ECMAScript 8
 - 2018 – ECMAScript 9
 - 2019 – ECMAScript 10
 - 2020 – ECMAScript11

Cenário atual e frameworks



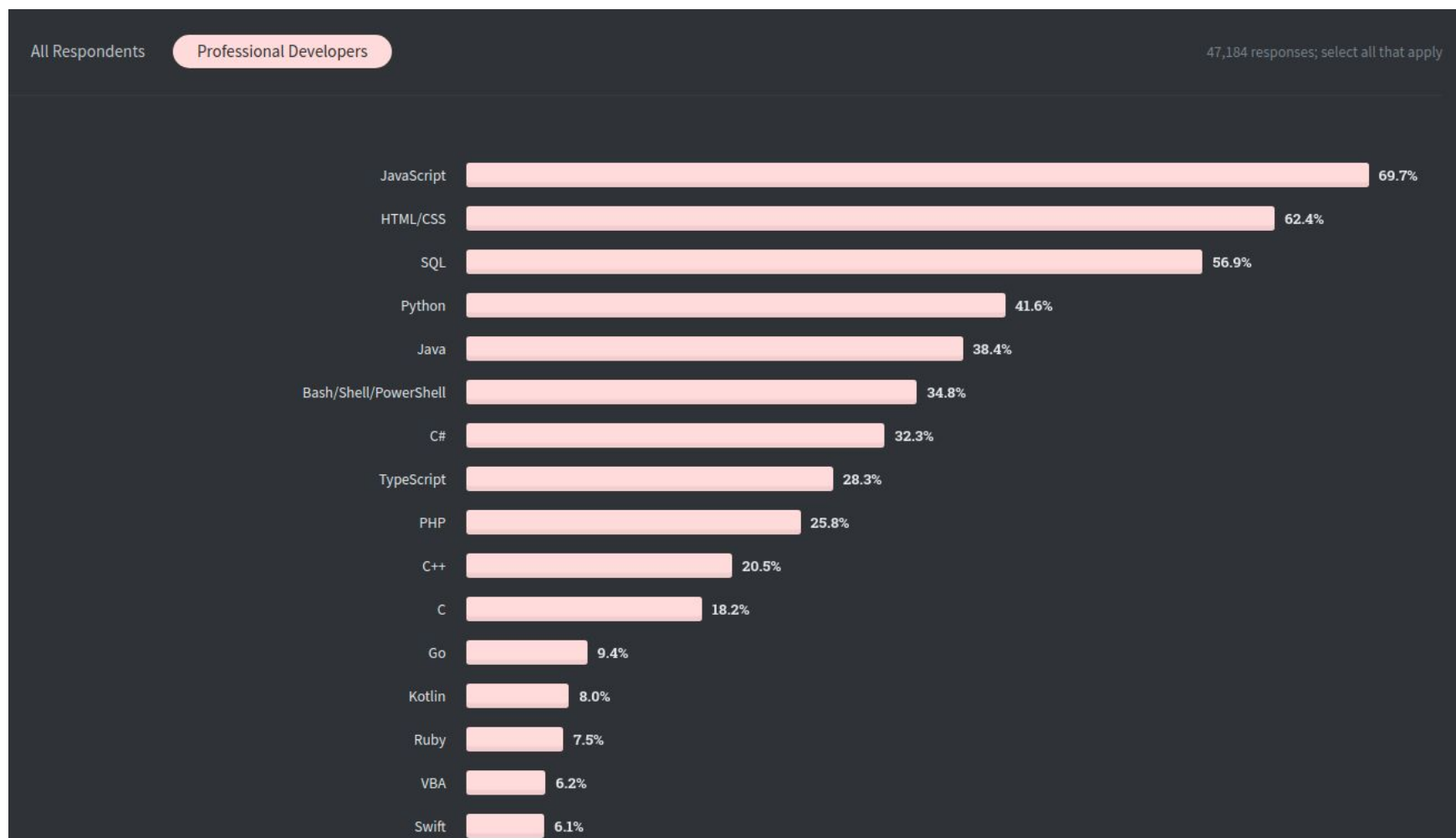
JavaScript: The road so far...

Cenário atual e frameworks

Cenário atual

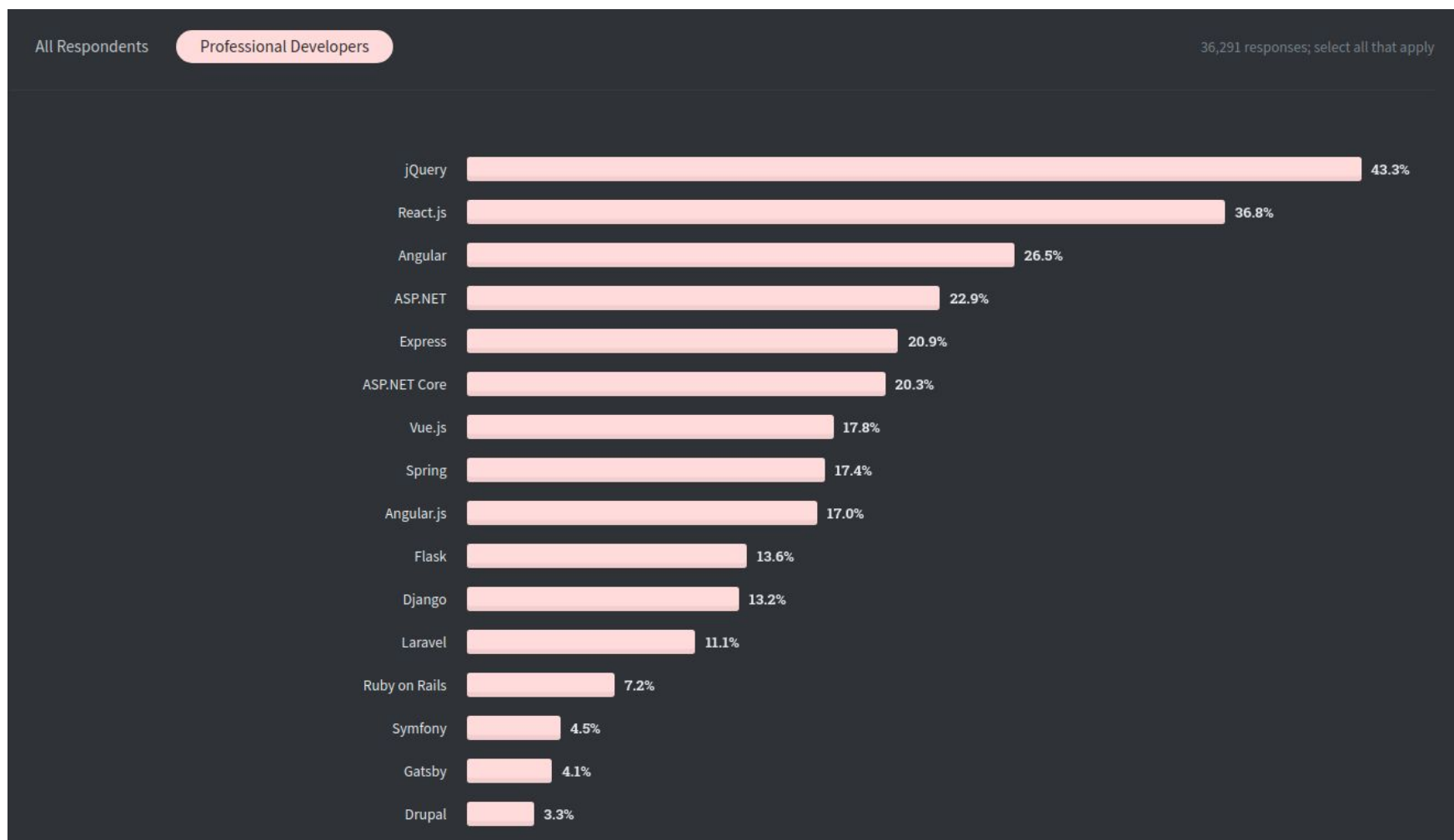
- Javascript rompeu as barreiras dos browsers com o Node.JS;
- ES6 e versões posteriores trouxeram mais maturidade à linguagem;
- Grande popularidade dos webapps responsivos;
- *Single Page Applications* e *Progressive Webapps (PWA)*;
- Constante e crescente popularidade, impulsionada pelo NodeJS.

Cenário atual e frameworks



[Stackoverflow Developer Survey 2020](#) - Programming Languages

Cenário atual e frameworks

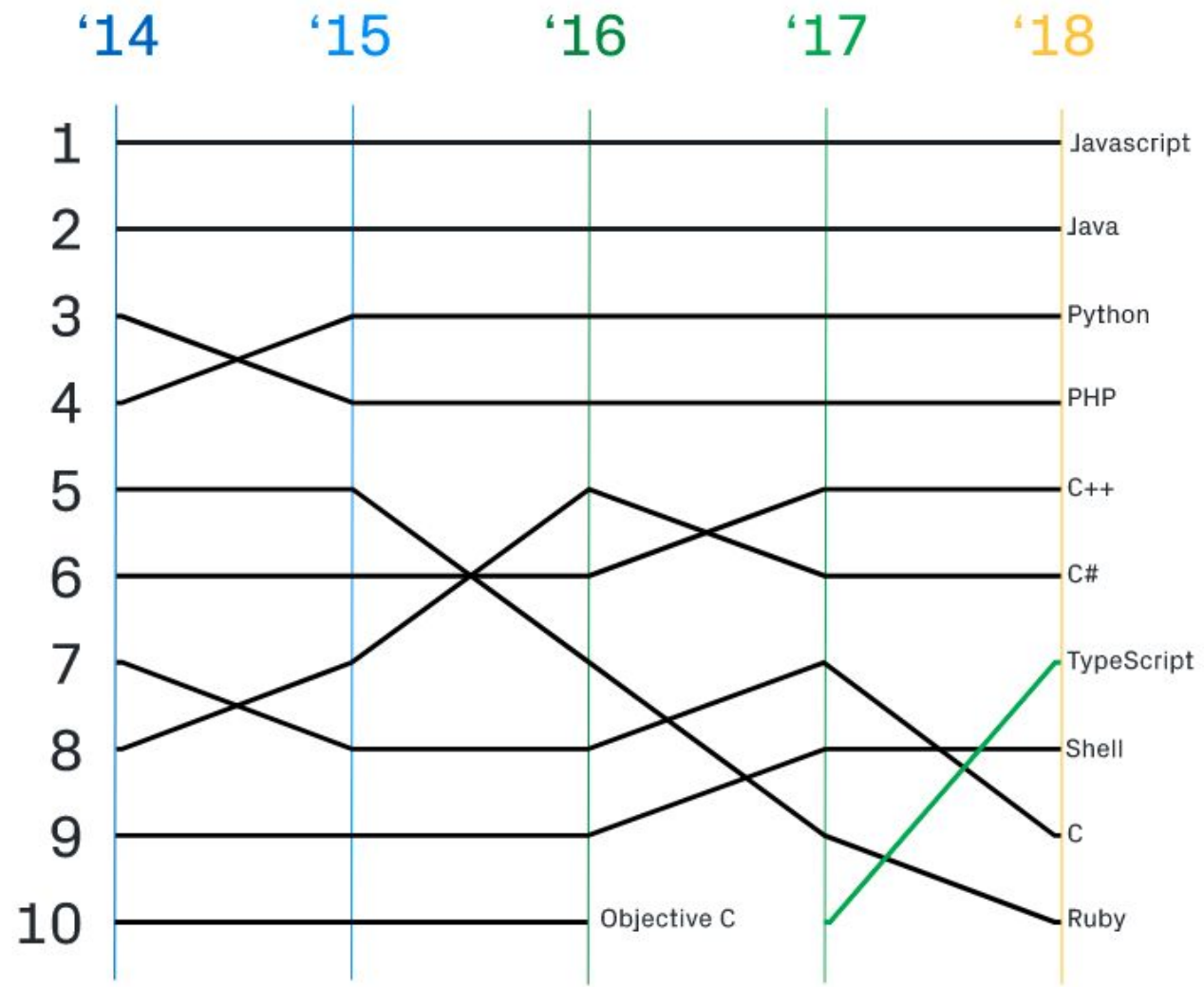


[Stackoverflow Developer Survey 2020](#) - Web Frameworks

Top languages over time

You're coding on GitHub in hundreds of programming languages, but JavaScript still has the most contributors in public and private repositories, organizations of all sizes, and every region of the world.

This year, TypeScript shot up to #7 among top languages used on the platform overall, after making its way in the top 10 for the first time last year. TypeScript is now in the top 10 most used languages across all regions GitHub contributors come from—and across private, public, and open source repositories. *



Fastest growing languages

We're seeing trends toward more statically typed languages focused on type safety and interoperability: Kotlin, TypeScript, and Rust are growing fast this year.

In addition, the number of contributors writing HCL, a human readable language for DevOps, has more than doubled since 2017. Popular in machine learning projects, Python is at #8. And there are 1.5x more contributors writing Go this year than last year. *

		Growth in contributors
1	Kotlin	2.6x
2	HCL	2.2x
3	TypeScript	1.9x
4	PowerShell	1.7x
5	Rust	1.7x
6	CMake	1.6x
7	Go	1.5x
8	Python	1.5x
9	Groovy	1.4x
10	SQLPL	1.4x

Cenário atual e frameworks

Era uma vez...

- jQuery
 - Não é um framework, mas destacamos aqui devido à sua importância para o passado recente;
 - Criado em 2006, é uma biblioteca de funções JS com várias soluções implementadas;
 - Grande facilidade para manipulação do DOM;
 - *Faça mais, escreva menos!*
 - Download: <https://jquery.com/>



Cenário atual e frameworks

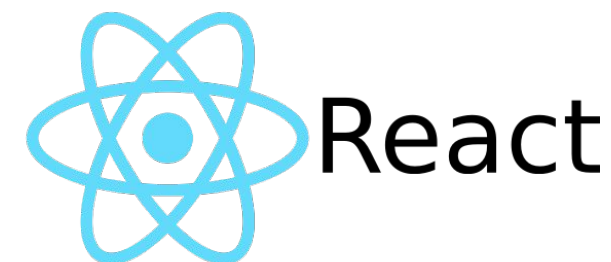
Passado recente (e presente...)

- Angular JS (1.x)
 - Criado em 2009 e mantido pelo Google;
 - Framework completo;
 - Programação em camadas (MVVM, MVC... MVW?);
 - Sincronização de dados entre HTML e JS (*two way data binding*);
 - Download: <https://angularjs.org/>



Cenário atual e frameworks

Principais frameworks de front-end



- React
 - Criado e mantido pelo Facebook desde 2013;
 - Melhor desempenho em vários aspectos;
 - Foco maior em *templates* e bom gerenciamento do DOM;
 - Grande popularidade, comunidade madura;
 - Download: <https://react-cn.github.io/react/downloads.html>

Cenário atual e frameworks

Principais frameworks de front-end

- Vue JS
 - Criado em 2014;
 - Implementa o padrão MVVM;
 - Mais simples que o Angular JS e similares;
 - Leve e rápido para criação de SPAs;
 - Download: <http://vuejs.org/>



Vue.js

Cenário atual e frameworks

Principais frameworks de front-end

- Aurelia
 - Lançando em setembro de 2014 por ex-membros de projetos concorrentes, como o Angular;
 - Arquitetura moderna, boa modularização;
 - Recursos semelhantes do Angular;
 - Pouca necessidade de dependências externas;
 - Download: <http://aurelia.io/>



Cenário atual e frameworks

Principais frameworks de front-end

- Angular
 - Lançando em setembro de 2016, a nova versão do Angular é um projeto diferente e mais maduro;
 - Mantido pelo Google, foi criado em TypeScript;
 - Radicalmente diferente do Angular JS (1.x), não possui compatibilidade com código da versão anterior;
 - Download: <https://angular.io/>



Cenário atual e frameworks

Ferramentas e produtividade

- SASS
- Typescript
- Node JS
- Grunt e Gulp
- Bower

Cenário atual e frameworks

Ferramentas e produtividade

- SASS
- Typescript
- Node JS
- ~~Grunt e Gulp~~
- ~~Bower~~

Cenário atual e frameworks

Ferramentas e produtividade

- Yeoman e Fountain JS
- Webpack e SystemJS
- Angular CLI
- Yarn

Cenário atual e frameworks

Ferramentas e produtividade

- ~~Yeoman e Fountain JS~~
- Webpack e SystemJS
- Angular CLI
- Yarn

A man with short brown hair, wearing a white t-shirt, is seen from the back and side, standing in a grocery store aisle. He is looking at shelves filled with various bottles of oil and vinegar. The shelves are well-stocked, and the lighting is bright. A semi-transparent grey box is overlaid on the center of the image, containing text.

Para refletir:

*O que pode funcionar bem para o nosso contexto?
Qual é a melhor opções para os nossos próximos projetos?*

Novas versões do ECMAScript

Novas versões do ECMAScript

Problema

- Browsers não estão preparados para suportar novas versões do ECMAScript;
 - Compatibilidade total para o ECMAScript 5, parcial para o ES6;
- Necessidade de utilizar novos recursos da linguagem levou ao surgimento de pré-processadores e linguagens que estendem o javascript;

Novas versões do ECMAScript

Problema

- Alguns recursos novos do ECMAScript:
 - Arrow functions;
 - Melhor gerenciamento de escopo na criação de variáveis;
 - Classes orientadas a objetos;
 - Interpolação de strings;

Novas versões do ECMAScript

Arrow functions

- Formas para escrever funções mais curtas;
- Sempre são funções anônimas;
- Resolvem o problema de escopo quando utilizamos *this*;
- Compatibilidade: <https://mzl.la/2hUUSwa>

Novas versões do ECMAScript

Arrow functions

- Forma geral:

```
([param], [param]) => {  
    statements  
}
```

```
param => expression
```


Novas versões do ECMAScript

Arrow functions

- Exemplos:

```
// ES5  
var vazia = function() {};  
  
// ES6  
let vazia = () => {};
```

Novas versões do ECMAScript

Arrow functions

- Exemplos:

```
19 // ES5
20 var maior = function(a, b) {
21     if( a > b ){
22         return a;
23     } else {
24         return b;
25     }
26 }
```

```
1 // ES6
2 let maior = (a, b) => {
3     if( a > b ){
4         return a;
5     } else {
6         return b;
7     }
8 }
```

Novas versões do ECMAScript

Arrow functions

- Exemplos:

```
var quadrado = function(num) {  
    return num * num;  
};  
  
// ES6  
let quadrado = a => a * a;  
  
quad(3); // 9  
quad(4); // 16
```

Novas versões do ECMAScript

Criação de variáveis e constantes

- Problema:
 - `var` – Define uma variáveis globalmente ou no escopo inteiro de uma função:
 - Exemplo:
 - Uma variável declarada com `var` dentro de uma função está limitada ao seu escopo, mas uma variável declarada dentro de um bloco como `if` “vazará” para fora deste bloco;

Novas versões do ECMAScript

Criação de variáveis e constantes

- **let** – permite declarar variáveis limitando seu escopo ao bloco, expressão ou instrução onde é usada (inverso do var);
- **const** – Declaração específica para constantes;

Novas versões do ECMAScript

Classes orientadas a objetos

- Até o ES5 o JS era uma linguagem OO baseada em protótipos:
 - Veja mais em: <https://mzl.la/2l2EDC1>
- Para criar algo parecido com uma classe, era necessário manipular o *prototype* de um objeto (Object.prototype);

Novas versões do ECMAScript

Classes com protótipos

- Exemplos:
 - Criar uma classe:

```
var Pessoa = function () {};  
  
var pessoa1 = new Pessoa();  
var pessoa2 = new Pessoa();
```

Novas versões do ECMAScript

Classes com protótipos

- Exemplos:
- Construtor:

```
1  var Pessoa = function () {  
2      console.log("exemplar criado");  
3  }  
4  
5  var pessoa1 = new Pessoa();  
6  var pessoa2 = new Pessoa();
```


Novas versões do ECMAScript

Classes com protótipos

- Exemplos:
- Atributos:

```
1  var Pessoa = function(nome) {  
2      this.nome = nome;  
3      console.log('Exemplar de Pessoa criado');  
4  };  
5  
6  var pessoa1 = new Pessoa('Alice');  
7  var pessoa2 = new Pessoa('Bob');
```

Novas versões do ECMAScript

Classes com protótipos

- Exemplos:
- Métodos:

```
1  var Pessoa = function (genero) {
2      this.genero = genero;
3      alert('Pessoa instanciada');
4  }
5
6  Pessoa.prototype.dizerOla = function()
7  {
8      alert ('hello');
9  };
10
11  var pessoa1 = new Pessoa('Masculino');
12  var pessoa2 = new Pessoa('Feminino');
13
14  // Chamando o método dizerOla em Pessoa .
15  pessoa1.dizerOla(); // hello
```

Novas versões do ECMAScript

Classes no ES6

- A partir do ES6 tornou-se possível criar classes com uma notação mais convencional (utilizando a palavra *class*);
- Esta nova notação é mais parecida com outras linguagens de programação OO e facilita a criação de atributos e métodos, e a utilização de conceitos de OO, como visibilidade de métodos e herança;
- Veja mais em: <https://mzl.la/2yEfsJ7>

Novas versões do ECMAScript

Classes no ES6

- Exemplos:

```
1  class Pessoa {  
2      constructor(genero, nome) {  
3          this.genero = genero;  
4          this.nome = nome;  
5      }  
6  
7      dizerOla() {  
8          alert("Hello");  
9      }  
10 }  
11  
12 let pessoa = new Pessoa("Masculino", "Eder");  
13 console.log(pessoa.genero);  
14 console.log(pessoa.nome);  
15 pessoa.dizerOla();
```

Novas versões do ECMAScript

Classes no ES6

- Exemplos:

```
var leao = new Leao();  
leao.miar(); // Aslan diz: MIAU!  
leao.rugir(); // Aslan ruge.
```

```
18  class Gato {  
19      constructor(nome) {  
20          this.nome = nome;  
21      }  
22  
23      miar() {  
24          console.log(this.nome + ' diz: MIAU!');  
25      }  
26  }  
27  
28  class Leao extends Gato {  
29      constructor(nome) {  
30          super(nome);  
31      }  
32      rugir() {  
33          console.log(this.nome + ' ruge.');34      }  
35  }
```

Novas versões do ECMAScript

Interpolação de strings

- Exemplo:

```
1  // ES5
2  var nome = "Eder";
3  var sobrenome = "Franco";
4  var nome_completo = nome + ' ' + sobrenome;
5  console.log(nome_completo); // Eder Franco
6
7
8  var nome = "Eder";
9  var sobrenome = "Franco";
10 var nome_completo = `${nome} ${sobrenome}`;
11 console.log(nome_completo); // Eder Franco
```


Novas versões do ECMAScript

Interpolação de strings

- Exemplo:

```
// ES6 (erro no ES5)
var saudacao = `
Olá, ${nome_completo}!
Isso é uma string com múltiplas
linhas!
`;
console.log(saudacao);
```

```
Olá, Eder Franco!
Isso é uma string com múltiplas
linhas!
```

Novas versões do ECMAScript

Algumas soluções

BABEL



LiveScript 

Novas versões do ECMAScript

Algumas soluções

- Babel: compila JS mais moderno para ES5;
- CoffeeScript, LiveScript e TypeScript são linguagens com sintaxe própria e que são compiladas para gerarem código compatível para os browsers;
- Lista de linguagens que compilam para JS:
 - <http://bit.ly/2yDu4sj>

**TO BE
CONTINUED...** 

Referências

Referências

- Medium. **Top Javascript framework topics to learn.** Disponível em: <https://medium.com/javascript-scene/top-javascript-frameworks-topics-to-learn-in-2017-700a397b7111>. Acesso em 31/03/2017.
- Site Point. **Top Javascript Frameworks, Libraries and Tools.** Disponível em: <https://www.sitepoint.com/top-javascript-frameworks-libraries-tools-use/>. Acesso em 31/03/2017.
- Hackernoon. **Five best Javascript frameworks.** Disponível em: <https://hackernoon.com/5-best-javascript-frameworks-in-2017-7a63b3870282>. Acesso em 31/03/2017.
- W3C. **Javascript Web APIs.** Disponível em: <http://www.w3.org/standards/webdesign/script>. Acesso em 31/03/2017.

Referências

- ARAÚJO, Leandro. **Programação Front-End com Javascript e jQuery**. Aulas 1 a 6. Manaus: BuriTech, 2014.
- Auth0. **A Brief History of JavaScript**. Disponível em <https://auth0.com/blog/a-brief-history-of-javascript/> Acesso em: 07/04/2017.
- UFRJ / NCE. **Características da Linguagem JavaScript**. Disponível em <http://www.nce.ufrj.br/ginape/js/conteudo/introducao/caracteristicas.htm> Acesso em: 07/04/2017.
- MDN. **JavaScript Básico**. Disponível em https://developer.mozilla.org/pt-BR/docs/Aprender/Getting_started_with_the_web/Javascript_básico Acesso em: 07/04/2017.

Referências

- CAMPOS, Daniel. **Yarn: A evolução do NPM**. Disponível em <https://tableless.com.br/yarn-evolucao-do-npm/> Acesso em 25/11/2017.
- Stack Overflow. **Developr Survey Results 2017**. Disponível em <https://insights.stackoverflow.com/survey/2017> Acesso em: 07/10/2017.
- Kangax. **ES6 compat table**. Disponível em <http://kangax.github.io/compat-table/es6/> Acesso em: 07/10/2017.
- ECMA International. **Standard ECMA-262**. Disponível em <https://www.ecma-international.org/publications/standards/Ecma-262.htm> Acesso em: 07/10/2017.
- Jashkenas. **List of languages that compiles to JS**. Disponível em <https://github.com/jashkenas/coffeescript/wiki/list-of-languages-that-compile-to-js> Acesso em: 07/10/2017.

Referências

- VENA, Stefano. **BazaarJS: Javascript preprocessors**. Disponível em <https://www.leanpanda.com/blog/2015/08/26/coffeescript-ecmascript-6-typescript/>
Acesso em 07/10/2017.
- MDN. **Object.prototype**. Disponível em https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Object.prototype Acesso em: 07/10/2017.
- MDN. **Arrow Functions**. Disponível em https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Functions/Arrow_functions Acesso em: 07/10/2017.
- MDN. **Let**. Disponível em <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Statements/let>
Acesso em: 07/10/2017.

Referências

- MDN. **Const.** Disponível em <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Statements/const> Acesso em 07/10/2017.
- PRADO, Felipe. **Diferenças entre VAR, LET e CONST.** Disponível em <http://www.matera.com/br/2017/05/09/javascript-6-diferenca-entre-var-let-e-const/> Acesso em: 07/10/2017.
- ILEGBODY, Ben. **Learning ES6 Classes.** Disponível em <http://www.benmvp.com/learning-es6-classes/> Acesso em: 07/10/2017.
- MDN. **Classes.** Disponível em <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Classes> Acesso em: 07/10/2017.

Eder Martins Franco

eder.franco@fpf.br
efranco23@gmail.com



facebook.com/efranco23
linkedin.com/in/efranco23
classroom.google.com