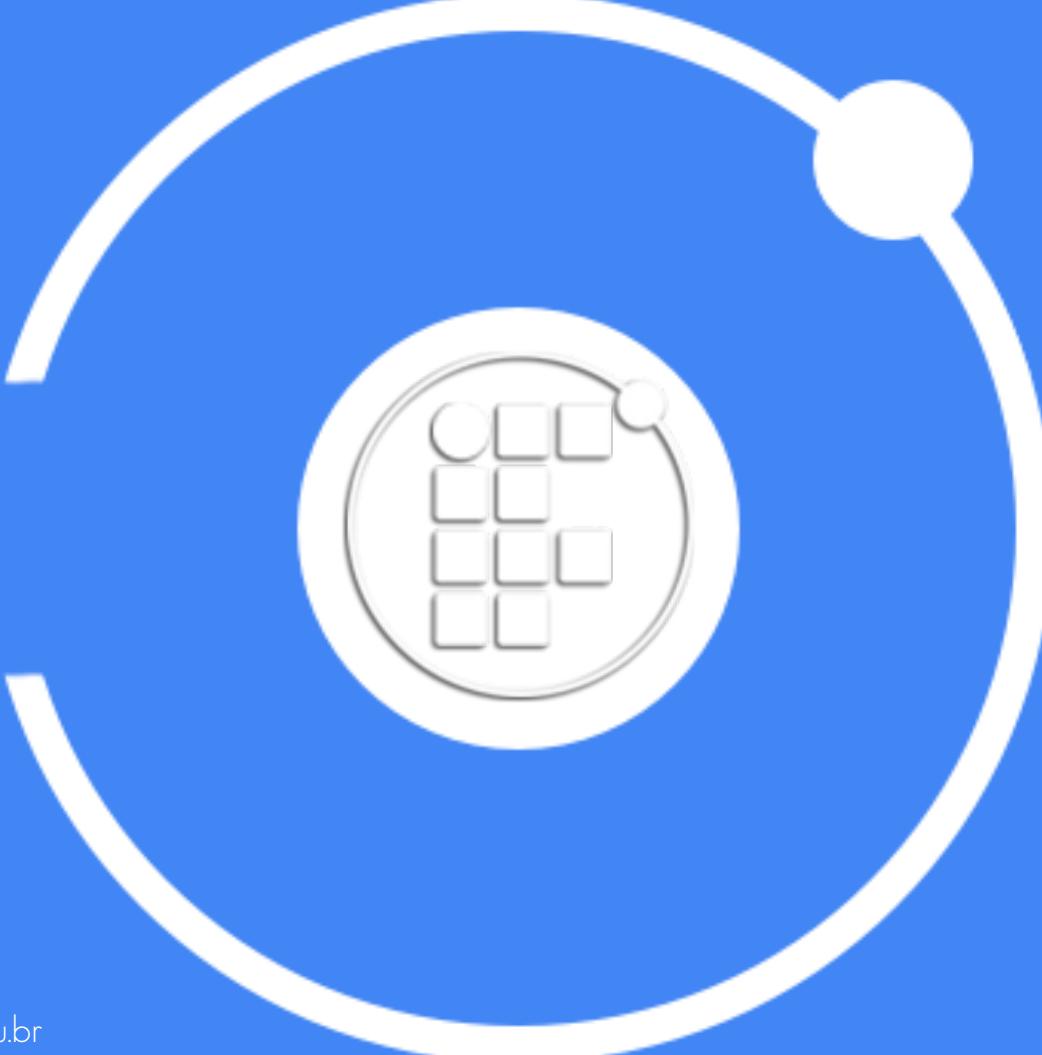


Ionic Framework

TypeScript



Antes vamos falar de



JavaScript

Introdução



- O JavaScript começou com um projeto de Brendan Eich, em abril de 1995, sob o nome Mocha, que pretendia implementar scripts em páginas web através do navegador da NetScape
- No mesmo ano o projeto foi batizado para liveScript.



Introdução



- Na mesma época surgia uma linguagem com prospecção de ser a linguagem do futuro, no caso, o Java.
- O Java era a tecnologia do momento e todos queriam utilizá-la ou portar seu projetos para essa nova linguagem.



Introdução



- Em uma jogada de marketing a Netscape se uniu a Sun Microsystems. A Sun buscava suporte as applets nos navegadores e Netscape promover sua nova linguagem, dessa forma, surgia então, o Javascript.
- O Javascript era na verdade o livescript, que fora apenas rebatizado com intuito de trazer novos desenvolvedores para a linguagem, aproveitando a popularidade da linguagem Java

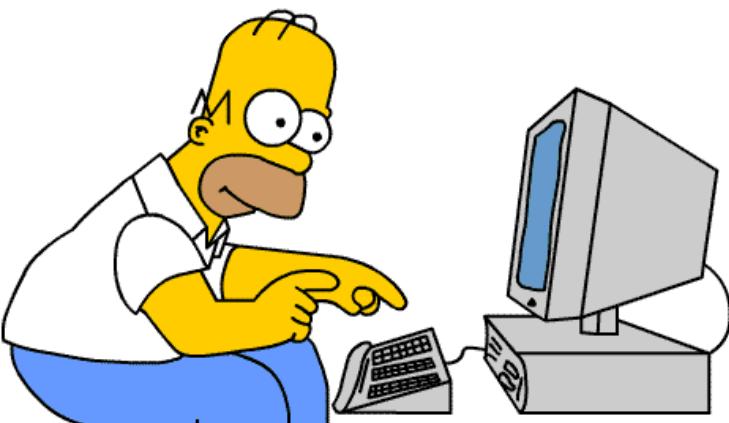


JavaScript™

Introdução



- A disputa pelo mercado de navegadores, se dava especificamente entre NetScape e Microsoft.
- Ao saber do novo recurso do concorrente a Microsoft lançou uma linguagem alternativa chamada JScript, que poderia ser considerada uma “cópia” do JavaScript.



Microsoft

Contextualizando



- Com objetivo de busca de reconhecimento, em 1996 a NetScape anunciou que o JavaScript estaria, a partir de então, submetido aos padrões da Ecma International. Assim surgiu o *ECMAScript* que é a base para linguagens como JavaScript, JScript e ActionScript.
- A ecma é uma associação Europeia dedicada a padronização de sistema de informação, ela já padronizou, por exemplo, a linguagem c#



Contextualizando



- ECMAScript é uma especificação de linguagens de script que o JavaScript implementa, ou seja, é a descrição formal e estruturada de uma linguagem de script.
- Em 2015 foi lançada a ES6 (também chamada de ECMAScript 2015) que define a sexta edição da padronização.
- O principal objetivo da nova versão da especificação foi tornar a linguagem mais flexível, enxuta e fácil de se aprender e trabalhar, tornando-a mais próxima a outras linguagens orientadas a objeto, como Java.



TypeScript

Contextualizando

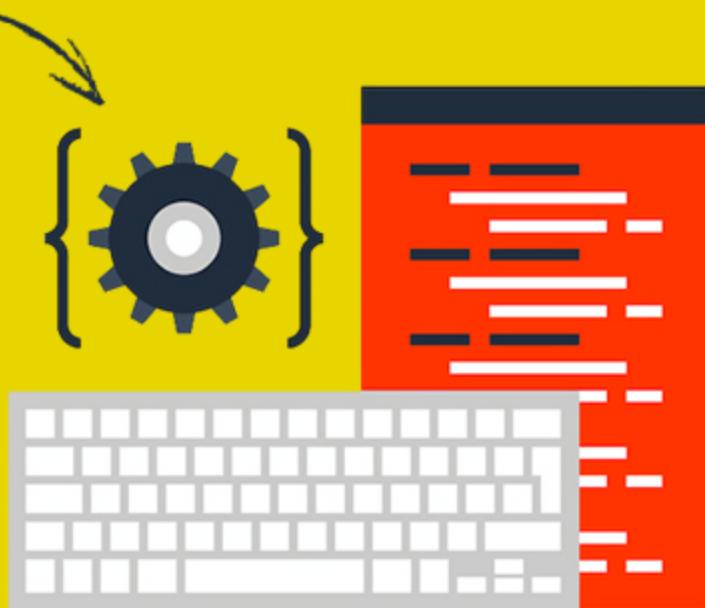


- Criada pela Microsoft, o TypeScript não se trata de uma linguagem nova, mas sim um superset (ou superconjunto) do JavaScript.
- Isto significa que você ainda pode utilizar a sintaxe JavaScript puro pra escrever suas aplicações ou adotar as regras do typescript.
- A partir dessa especificação é possível criar classes, declarar variáveis tipadas, encapsulamento, utilizar Herança entre outros.



Como funciona

TS

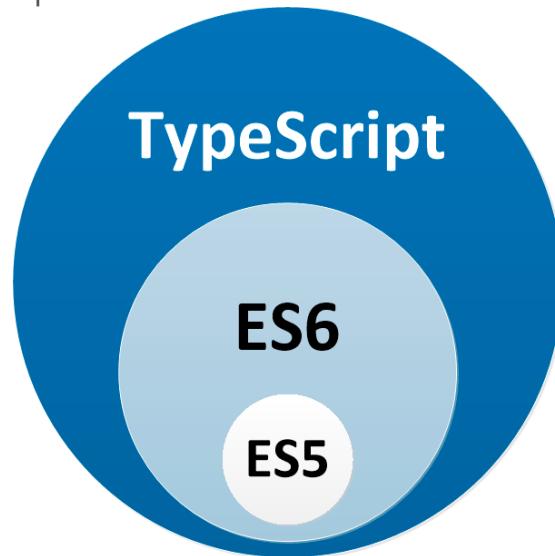


JS

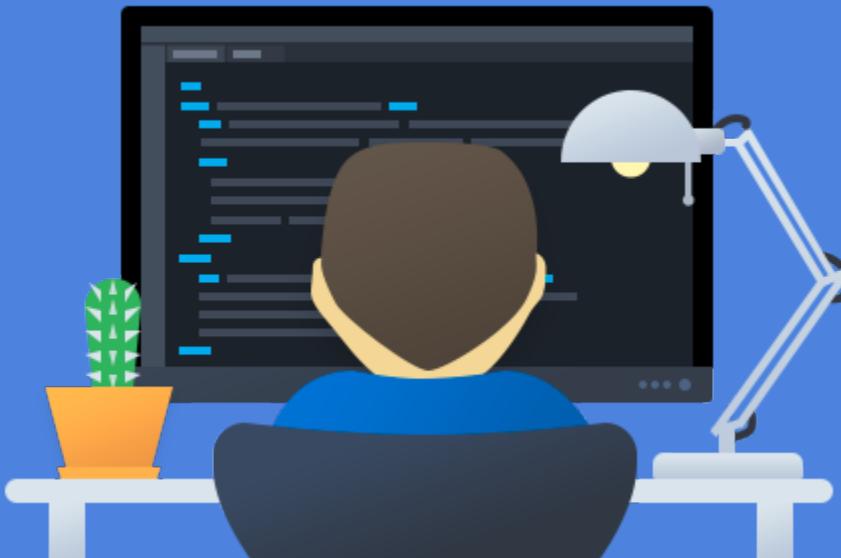
TypeScript



- Quando utilizamos ionic escrevemos nossas lógicas através de typescript, o qual reconhece os padrões da es6, contudo, ao ser transpilado ele é traduzido para es5. Essa abordagem visa manter compatibilidade com maior número de dispositivos possível.
- Possui código fonte aberto
- Suportado pelas ides modernas



Exemplos práticos





```
var soma = function(x, y){  
    return x + y;  
}  
soma(1,2)  
soma(1,"2")
```



```
var soma = function(x:number, y:number){  
    return x + y;  
}  
  
soma(1,2)  
soma(1,"2")
```

Funções simplificadas



```
var mult = function(a,b){  
    return a*b;  
}
```



Tipagem em funções

```
var somaLados = (poligono: { x: number; y: number; }) => {
    return poligono.x + poligono.y;
}
```

Inferência de tipo



```
var numero = 1; //isso é um número
```

Classes



```
class Animal {  
    respira(): string {  
        return "Respirando.... Respirando....";  
    };  
}  
class Cachorro extends Animal {  
    latir(): string {  
        return "Au Au!!";  
    }  
}
```

Classes



```
class Animal {  
    respira(): string {  
        return "Respirando.... Respirando....";  
    };  
}  
class Cachorro extends Animal {  
    latir(): string {  
        return "Au Au!!";  
    }  
}  
var cachorro = new Cachorro();  
var animal = new Animal();  
console.log(animal.respira());  
console.log(cachorro.respira());
```

Construtores



```
class Animal {  
    constructor(public nome: string) {};  
    respira(): string {  
        return this.nome + " Respirando.... Respirando....";  
    };  
}  
class Cachorro extends Animal {  
    late(): string {  
        return this.nome + " Au Au!!";  
    }  
}  
var cachorro = new Cachorro("Link");  
var animal = new Animal("Arroba");
```

Interface de função



```
interface binario {  
    (x: number, y: number): number;  
}  
  
var soma:binario = (x, y) => x + y;
```

Interface de objetos



```
interface Cachorro {  
    nome: string;  
    late: () => void;  
}
```

Interface com parâmetros



```
interface Retangulo {  
    altura: number;  
    largura?: number;  
}
```

Interface para classes



```
interface Calc {  
    soma(n: number): Calc;  
    total(): number;  
}  
interface Calc {  
    subtrai(n: number): Calc;  
}
```

Encapsulamento



```
class Pessoa {  
    private _nome: string;  
    get nome(): string {  
        return this._nome;  
    }  
    set nome(value: string) {  
        this._nome = value;  
    }  
}  
var p = new Pessoa();  
p.nome = "Giovanni";  
console.log(p.nome);
```