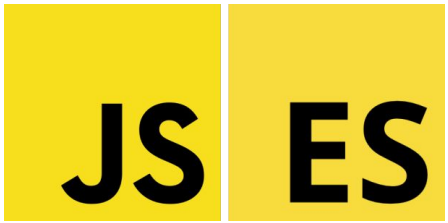
centro universitário FEI

CCP150
DES. DE APLICATIVOS MÓVEIS

Prof. Isaac
Prof. Rafael Alves

DESENVOLVIMENTO DE APLICAÇÕES MOBILE

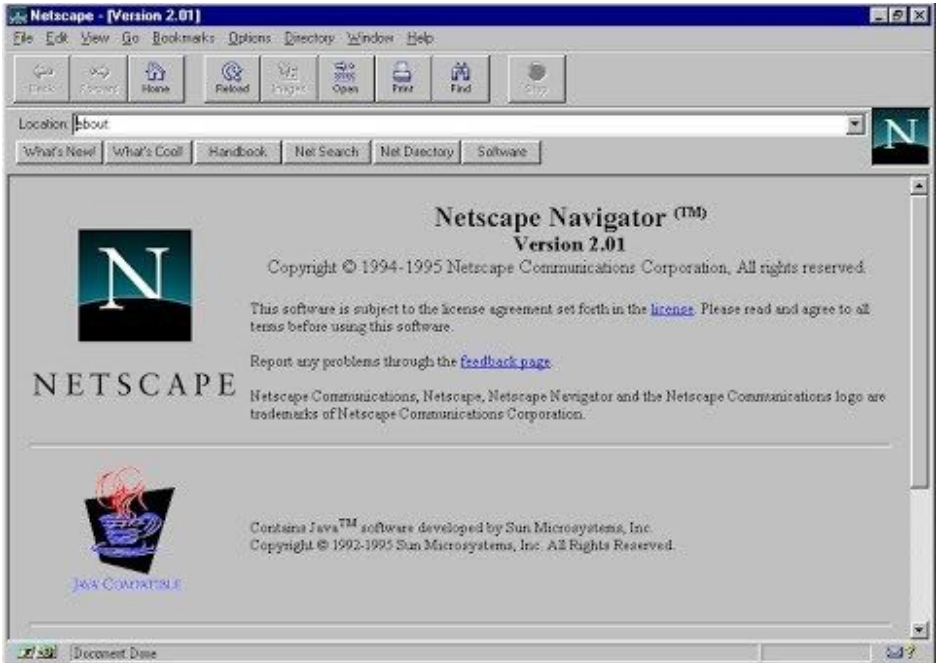
ECMAScript 6



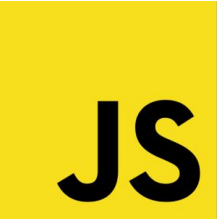
Explicando o JavaScript...

- Desenvolvido pela Netscape na década de 90

Netscape Navigator:

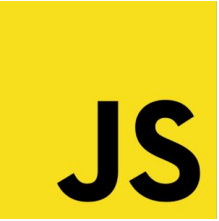


Explicando o JavaScript...



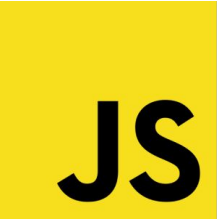
- Netscape e Sun Microsystems (Java) fizeram, na época, um acordo para o desenvolvimento de linguagem de programação para o Navegador
- Netscape decide que a linguagem do navegador deveria complementar o Java (ter também sintaxe semelhante)
- Nome inicial durante o desenvolvimento: **Mocha**
- Chamada oficialmente de **LiveScript** nos primeiros lançamentos beta do Netscape Navigator 2.0

Explicando o JavaScript...



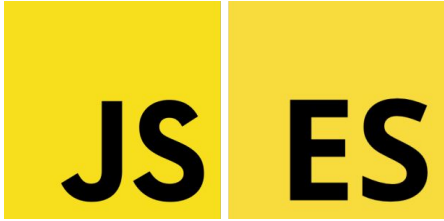
- Poucos meses depois, ainda nas versões beta do Netscape Navigator 2.0, a linguagem mudou de nome para **JavaScript**
- A mudança foi uma jogada de marketing para dar ao JavaScript o status que o Java tinha na época
- O nome causou e causa confusão até hoje, dando a impressão de que a linguagem é somente uma derivação do Java

Explicando o JavaScript...



- Logo após (1996), a Microsoft fez engenharia reversa no JavaScript e lançou o **JScript** no **Internet Explorer 3**
- Então, a Netscape decidiu normatizar a linguagem através da organização **ECMA International** (originalmente, *European Computer Manufacturers Association*), companhia que era especializada em padrões e normativas

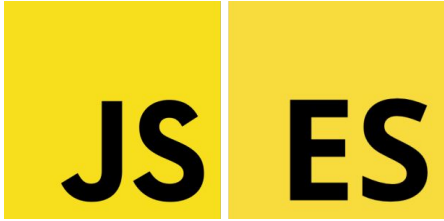
ECMAScript 6



- Os trabalhos em cima da normativa ECMA-262 começaram em Novembro de 1996
- O nome JavaScript já era patenteado pela Sun Microsystems (hoje Oracle) e não poderia ser usado
- Portanto, o nome composto por ECMA e JavaScript foi usado, resultando em **ECMAScript**

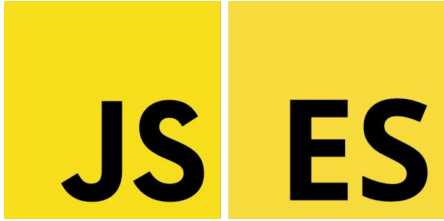
Fontes: https://www.mundojs.com.br/2018/01/27/o-que-e-ecmascript-es6-es8/
http://shipit.resultadosdigitais.com.br/blog/javascript-1-uma-breve-historia-da-linguagem/

ECMAScript 6



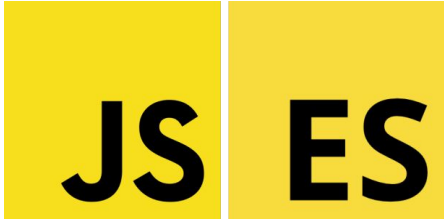
- Simplificando, o **ECMAScript é o padrão**
- **ECMA-262** é a especificação desse padrão
- O JavaScript é a implementação (dialeto) mais popular desse padrão
- Conhecida também por: ES6, ECMAScript 2015 ou JavaScript 6
- O **ECMAScript 6 adicionou novas funcionalidades** à linguagem JavaScript

ECMAScript 6



- **Algumas das novas funcionalidades:**
  - let
  - const
  - Classes
  - Arrow functions
  - Parâmetros default
  - Operador exponencial (**)
  - Operador de expansão (...)

let e const



- Declaração de variáveis no contexto de bloco
- Podemos declarar variáveis de 4 formas no JavaScript:
  - ***'nada'***
  - ***var***
  - ***let***
  - ***const***

variáveis

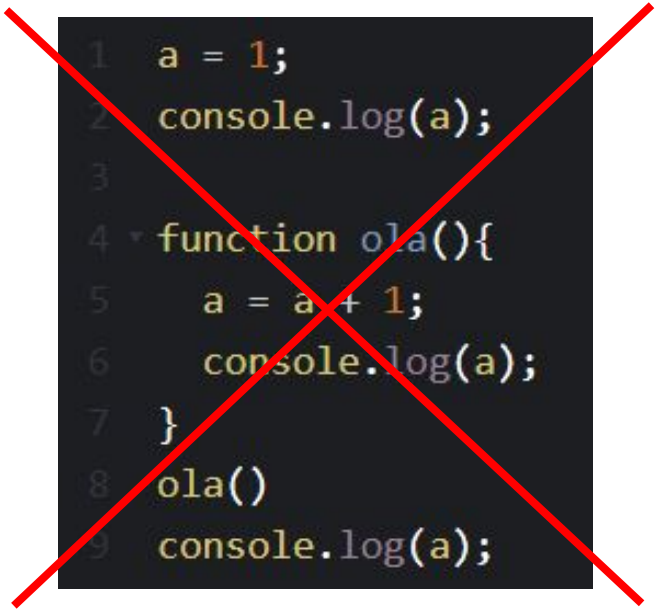
- ***'nada' -** variável global (não recomendado!)*

```
1  a = 1;
2  console.log(a);
3
4  function ola(){
5    a = a + 1;
6    console.log(a);
7  }
8  ola()
9  console.log(a);
```

Escopo da variável a

variáveis

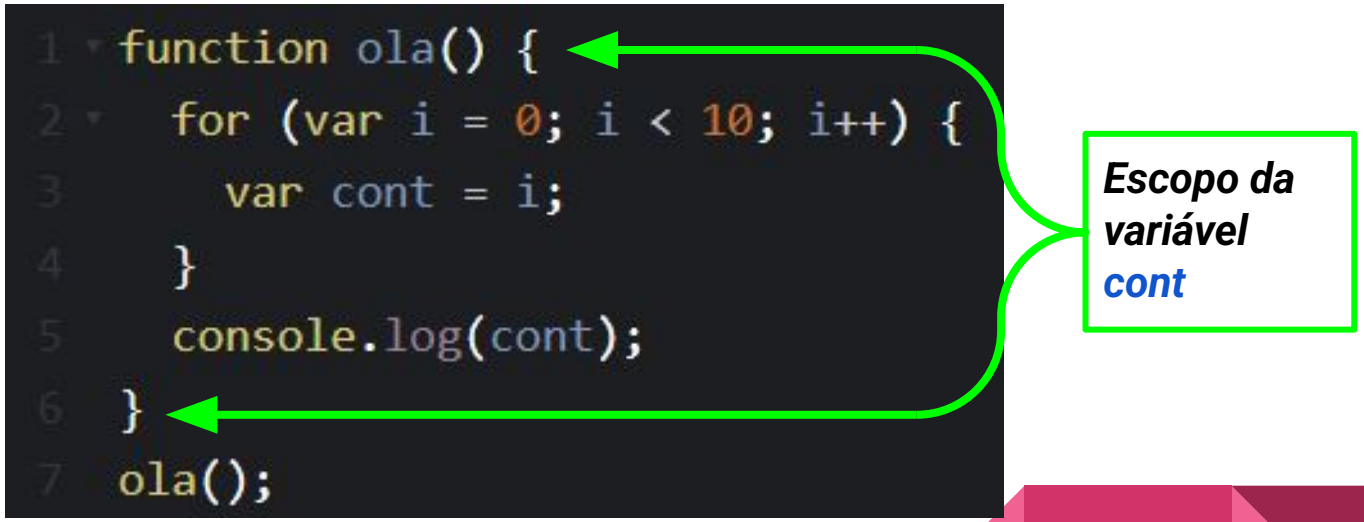
- ***'nada' -** variável global (não recomendado!)*



variáveis

JS ES

- ***var -*** *variável com escopo de função*



variáveis

JS ES

- ***let -*** *variável com escopo de bloco*

```
function ola() {
  for (let i = 0; i < 10; i++) {
    let cont = i;
  }
  console.log(cont);
Uncaught ReferenceError: cont is not defined
}
ola();
```

variáveis

- ***let -*** *variável com escopo de bloco*

```
function ola() {
  for (let i = 0; i < 10; i++) {
    let cont = i;
    console.log(cont);
  }
}
ola();
```

Escopo da variável cont

variáveis

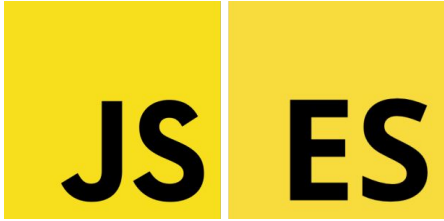
JS ES

- ***const -*** *igual ao let, só que as variáveis são read-only*

```
1  const a = 1;
2  a = 2;
```

```
Uncaught TypeError: Assignment to constant variable.
```

Classes



- **Importante avanço no JavaScript**
- Possibilita Orientação a Objetos
  - Herança
  - Polimorfismo

Classes

- **Declaração:**

```
class Pessoa{
  constructor(nome){
    this.nome = nome;
  }

  getName(){
    return this.nome;
  }

  setName(nome){
    this.nome = nome;
  }
}
```

- **Instância:**

```
pessoa = new Pessoa("Fulano");
console.log(pessoa.getName());
```

https://codepen.io/danilo-perico/pen/LYZYXEb?editors=0011

Classes

- **Declaração**
  syntax sugar

```
class Pessoa{
  constructor(nome){
    this._nome = nome;
  }

  get nome(){
    return this._nome;
  }

  set nome(nome){
    this._nome = nome;
  }
}
```

O ***underscore*** antes da propriedade ***nome*** é uma convenção que indica que essa variável deve ser mantida privada, embora o JavaScript não impeça o acesso direto a ela.

https://codepen.io/danilo-perico/pen/GRqRwpQ?editors=0011

Classes

- **Declaração *atributo private***

No JavaScript atual (***ECMAScript 2022***), você pode usar a sintaxe **#** para criar variáveis privadas que são verdadeiramente ***inacessíveis fora da classe***.

```
class Pessoa{
  #nome = ""
  #idade = 0
  constructor(nome, idade){
    this.#nome = nome;
    this.#idade = idade;
  }

  get nome(){
    return this.#nome;
  }

  get idade(){
    return this.#idade;
  }

  set nome(nome){
    this.#nome = nome;
  }
}
```

```
pessoa1 = new Pessoa("Fulano", 25);
pessoa1.nome = "Novo"
console.log(pessoa1.nome);
console.log(pessoa1.idade);
```

https://codepen.io/Isaac-Jesus-Silva/pen/xxmzpaj

Classes

- **Declaração *atributo static***

```
class Test {
  static contador = 0;
}

teste01 = new Test()

console.log(teste01.contador)
console.log(Test.contador)
```

```
Console
undefined
0
```

No JavaScript atual (***ECMAScript 2022***), você pode usar a palavra ***static*** para criar variáveis static que não são das instâncias da classe ***assim para acessá las você deve usar o nome da classe***.

https://codepen.io/Isaac-Jesus-Silva/pen/rNoqaWN

Classes

- **Declaração *métodos static***

```
class Test {
  static printHello(){
    console.log('Hello this is a greeting from a static method');
  }
}

Test.printHello();
```

No JavaScript atual (***ECMAScript 2022***), você pode usar a palavra ***static*** para criar métodos static que não são das instâncias da classe ***assim para acessá las você deve usar o nome da classe***.

https://codepen.io/Isaac-Jesus-Silva/pen/ExGdamP?editors=0012

Classes - Herança

JS ES

```
class Aluno extends Pessoa{
  constructor(nome, ra){
    super(nome);
    this._ra = ra;
  }
}
```

```
aluno = new Aluno("Fulano", 123456);
console.log(aluno.nome);
```

Classes - Herança

- Crie a classe Aluno Herdando a classe Pessoa.
- Crie o atributo ra como privado
- Adicione os métodos get e set do ra.
- Crie um objeto do tipo aluno e testes os métodos get e set ra.

https://codepen.io/Isaac-Jesus-Silva/pen/xxmzpaj

Arrow Functions

JS ES

- Sintaxe alternativa para declaração de funções

Função convencional

```
var sum = function(x, y) {
  return x + y;
};

console.log(sum(1, 2));
```

Função de seta (Arrow Function)

```
const sum = (x, y) => {
  return x + y
}

console.log(sum(1, 2))
```

https://codepen.io/danilo-perico/pen/oNLgNMo?editors=0012

Arrow Functions

JS ES

- Pode ainda ficar melhor, se só existir a linha do ***return*** no corpo da função:

```
const sum = (x, y) => x + y

console.log(sum(1, 2))
```

Parâmetros Default

JS ES

- Permite a atribuição de um valor padrão à um ou mais parâmetros

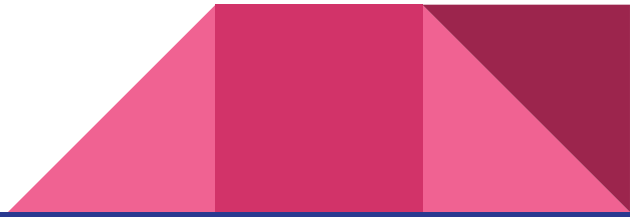
```
function f(a=2, b=3){
   return a+b;
}

console.log(f(1,2)); // saída será 3 pois a=1 e b=2
console.log(f(1));   // saída será 4 pois a=2 e b=3(default)
console.log(f());    // saída será 5 pois a=2(default) e b=3(default)
```

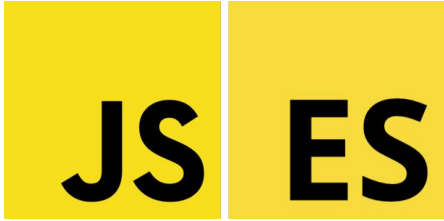
https://codepen.io/danilo-perico/pen/rNLaadp?editors=0012

Operador Exponencial

- Substitui o ***Math.pow(a,b)***
- Por ***a**b***



Operador de Expansão ...



- Utilizado para copiar um vetor (array) ou objeto
- Pode concatenar valores em um vetor
- Sintaxe simples e poderosa

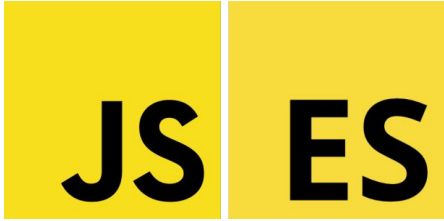
Operador de Expansão ...

JS ES

```
let v = [1,2,3,4];
let v2 = [...v]; // operador de expansão criando um segundo vetor
let v3 = [0, ...v]; // inserção do item 0 na primeira posição e cópia do vetor v
let v4 = [0, ...v, 5]; // inserção do item 0 e 5 e cópia do vetor v
let v5 = [2, ...v.slice(1,4), 9]; // concatena 2, sub vetor [2,3,4] e 9
console.log(v);
console.log(v2);
console.log(v3);
console.log(v4);
console.log(v5);
```

https://codepen.io/danilo-perico/pen/KKMwwJq?editors=0011

ECMAScript 6

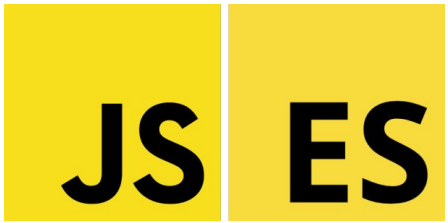


- As novidades do ES6 podem ser vistas no site:

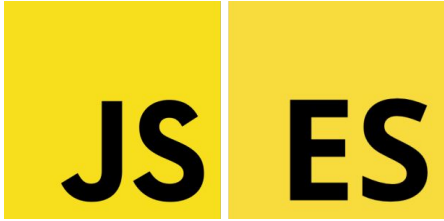
  http://es6-features.org/

- O **React** e o **React Native** suportam automaticamente o **ES6**

Revisão de JS

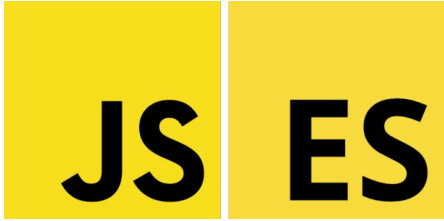


JavaScript



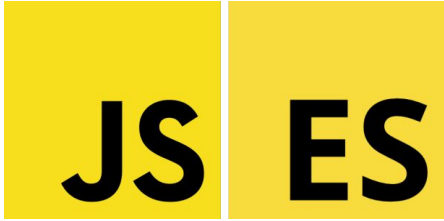
- Revisão de 2 conceitos importantes:
  - ***this***
  - ***bind( )***

JavaScript



- ***this***
  - Pronome utilizado quando precisamos nos referir ao objeto atual
  - Dependendo do contexto, ou do escopo, ***this*** pode referenciar diferentes objetos!
  - **JavaScript tem contexto dinâmico**

JavaScript



- ***this - Exemplo***
  - Podemos chamar várias vezes uma função em vários contextos diferentes
  - Exemplo: função ***play***

https://codepen.io/danilo-perico/pen/Bazyjyg?editors=0011

JavaScript

- ***this - Exemplo***
  - Perceba que no exemplo anterior, da função ***play***, temos muita repetição de código
  - Podemos criar a função ***play*** somente uma vez

https://codepen.io/danilo-perico/pen/LYZEGNm?editors=0012

JavaScript

- ***this e bind( )***
  - Mas agora a função ***play*** só funciona com a variável ***filme*** no contexto global
  - Porém, podemos alterar o contexto utilizando a função ***bind( )***

```
let playGlobal = play
let playCinema = play.bind(cinema)
let playNetflix = play.bind(netflix)

playGlobal()
playCinema()
playNetflix()
```

Exemplos inspirados no site:
https://imasters.com.br/javascript/javascript-entendendo-o-de-uma-vez-por-todas

JavaScript

- ***this e bind( )***
  - Mas agora a função ***play*** só funciona com a variável ***filme*** no contexto global
  - Com arrow function não podemos alterar o contexto utilizando a função ***bind( )***

https://codepen.io/danilo-perico/pen/BazyjpB?editors=0011

Exemplos inspirados no site:
https://imasters.com.br/javascript/javascript-entendendo-o-de-uma-vez-por-todas

PRÁTICA

JavaScript

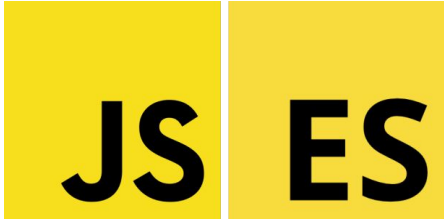
1. Refatore o código a seguir para que declaração do objeto VideoGame seja uma classe de acordo com o ES6. Não se esqueça de criar o construtor e invocar um objeto desta classe

```
function VideoGame(marca, nControles, tipoMidia) {
  this.marca = marca;
  this.nControles = nControles;
  this.tipoMidia = tipoMidia;
}

var playstation = new VideoGame('sony', '2', 'dvd');
console.log(playstation);
// { marca: 'sony', nControles: '2', tipoMidia: 'dvd' }
```

Fonte: ECMAScript 6 - Entre de cabeça do futuro do JavaScript. Diego Martins de Pinho

JavaScript



1. Inclua o atributo ***ligado***

   Inclua os métodos: ***jogar()***, ***ligar(estado)*** e ***salvarJogo()***;