



# Declarações em JavaScript

Prof. Romerito Campos

# Declarações em JavaScript

## Conteúdo

- Expressões (primária, object e array, function, acesso a propriedades)

# Intro

- Declarações em JS são finalizadas com `;`.
- Declarações são avaliadas de modo que algo aconteça, enquanto expressões produzem um valor.
- Programs em JavaScript são sequências de declarações.
  - Exemplos: condicionais, loops, jumps. Jumps: `break`, `return` e `throw`.

# Declarações de expressões

- Operação de atribuição: `nome = "a" + "b"`.
- Operador de incremento: `cont++`.
- Remover uma propriedade de um objeto: `delete obj.nome;`.
- Expressões que têm efeitos colaterais, como mudar o valor de algo, são chamadas de expressões de declaração (`statement`).

# Declarações vazias e compostas.

- Uma declaração vazia é simplesmente `;`.
- Declaração em bloco:

```
{  
    a = Math.sqrt (10);  
    a = a++;  
    console.log(a);  
}
```

# Condicionais

- `if`, `if else`, `switch`,
- Tomada de decisão

```
if (a >= 10) {  
    b = a;  
}  
if (a === 10) {  
    b = a;  
}
```

# Condicionais

- Exemplo de if-else

```
let a = 10;  
if (a === 10) {  
    console.log('algo')  
} else if (a === 20) {  
    console.log('algo')  
} else {  
    console.log('algo')  
}
```

# Condicionais

- Exemplo de `switch`

```
let nome = 'joao'

switch (nome) {
  case 'joao':
    console.log(nome);
    break
  case 'maria':
    console.log(nome);
    break
  default:
    console.log('valor padrão');
}
```



# Loops

- Há várias opções de laço `for`.
- `while`
- `do/while`
- `for`
- `for/of`
- `for/in`

# while

- Primeiro avalia a expressão e se for **true**, executa o corpo do laço.

```
let a = true;  
while () {  
    //loop infinito  
}
```

```
while (a) {  
    //executa apenas uma vez  
    a = !a;  
}
```

## do/while

- Nesta variação, pelo menos uma execução do corpo do laço é executada.

```
let a = true;  
do {  
    //print do a uma vez  
    console.log(a);  
} while(a)
```

```
let a = true;  
do {  
    //loop infinito  
} while (a);
```

# for

- Laço clássico.

```
for (inicializar; testar; incrementar) {  
    // Corpo  
}
```

```
//adicionar 10 números ao array 'a'.  
a = [];  
for (let i = 0; i<=10>; i++) {  
    a[i] = i;  
}
```

```
let a = [];  
//observe incremento  
for (let i = 0; i<10; a[i] = i++);
```

## for/of:

- Incluído no ES 6
- Trabalha com objetos "iteráveis"
- E campos de objetos: ARRAYS, Strings, sets, MAPS
- `let a = [1, 2, 3, 4]`
- `for (let i of a) {console.log(i);}`
- Objetos não são por padrão iteráveis  
Exemplo de uso de `Object.keys(obj)` e `Object.values(obj)`.
- Exemplo com string

# for/of

- Iteração sobre array e objetos

*//exemplo com array*

```
let a = [1, 2, 3, 4, 5]
for (let i of a) {
  console.log(i)
}
```

*//exemplo com objeto*

```
let a = {y: 'jose', x: 'teste'}
for (let i of Object.values(a)) {
  console.log(i)
}
```

## for/in:

- Similar ao `for/of`, mas não necessita que o objeto seja iterável

```
let obj = {a: 'rc', b: 'da'};  
for (let i in obj) {  
    console.log(obj[i])  
}
```

# Jumps

- Comuns em muitas linguagens
- Mudam o fluxo do programa com um "pulo"
- Exemplo: `break`, `return`



# Declaração Rotulada

- Você pode rotular parte do código a partir de um nome e "saltar" para este bloco
  - Somente é útil por declarações (statement) que têm corpo como loops e condicionais.

```
pulo: while (true) {  
  console.log('r')  
  continue pulo;  
}
```

# break

- Usado para encerrar prematuramente um laço ou switch

```
for (let i=0; i < 100; i++) {  
    if (i == 9) {  
        break;  
    }  
}
```

# continue

- Similar ao comando `break`, mas não interrompe o laço. Apenas passa para a próxima iteração.

```
for (let i=0; i < 100; i++) {  
  if (i % 2 == 0) {  
    continue;  
  }  
}
```

# return

- Aparece apenas no corpo de funções

```
function teste() {  
    return 0;  
}
```

# Referências

FLANAGAN, D. **JavaScript - The Definitive Guide**. 7. ed. Sebastopol, CA, USA: O'Reilly Media, 2020.