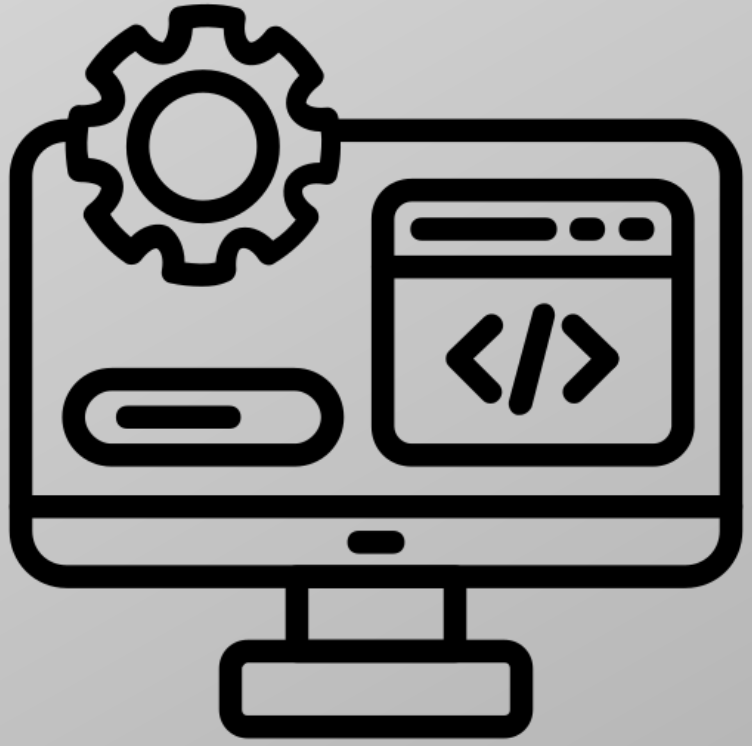


Desenvolvimento BACK-END I

Professor: Douglas Legramante

E-mail: douglas.legramante@ifro.edu.br

INSTITUTO FEDERAL
Rondônia
Campus Vilhena

JavaScript

Aula 03 – Operadores

O que são operadores?

INSTITUTO FEDERAL
Rondônia
Campus Vilhena

Os operadores são símbolos usados para modificar ou até mesmo gerar um novo valor. Na programação, em muitos casos, precisamos que um valor seja gerado a partir da análise, combinação ou comparação entre valores e são os operadores que fazem isso.

```
1 | var atendeAClassificacaoEtária = idade >= 18
```

Operadores

INSTITUTO FEDERAL
Rondônia
Campus
Vilhena

- Aritméticos
- Atribuição
- Incremento
- Comparação, ou relacionais
- Lógicos

Operadores Aritméticos

INSTITUTO FEDERAL
Rondônia
Campus Vilhena

adição (+), subtração (-), multiplicação (*), divisão (/) e módulo (%)

```
1 | var quantidadeVagas = 2 + 5; // resultado: 7
2 | var contratados = 7 - 2; // resultado: 5
3 | var valorContribuicao = 2500 * 0.10; // resultado: 250
4 | var primeiraParcela = 2500 / 2; // resultado: 1250
```

Operadores Aritméticos

INSTITUTO FEDERAL
Rondônia
Campus Vilhena

módulo (%) - Esse operador retorna o resto de uma divisão.

```
1 | var resto1 = 15 % 4; // resultado: 3
2 | var resto2 = 3 % 1.2; // resultado: 0.6
```

Este operador pode ser usado para verificar se um número é par. Para fazer isso, basta verificar se o número dividido por 2 gera um resultado inteiro e resto zero (0).

```
1 | var verificaSeEPar = 20 % 2 == 0;
```

Concatenando strings com o operador de adição (+)

INSTITUTO FEDERAL
Rondônia
Campus Vilhena

Além de ser usado para somar número, o operador de adição (+) também pode juntar strings.

```
1 | console.log("texto" + " e complemento");
2 | // Vai imprimir "texto e complemento"
3 |
4 | console.log("5" + "6");
5 | // Vai imprimir "56"
```

Note que em ambos os exemplos estamos utilizando tipos iguais: strings.

Conversão para strings

INSTITUTO FEDERAL
Rondônia
Campus Vilhena

Quando usamos o operador de adição com tipos diferentes, digamos uma string e um number, o operador de adição primeiro converte todos os valores para string e depois as une.

```
1 | console.log("texto" + 20); // "texto20"
2 | console.log("texto" + true); // "textotrue"
3 | console.log("texto" + null); // "textonull"
4 | console.log("texto" + undefined); // "textoundefined"
5 | console.log( [1,2,3] + 4 ) // "1,2,34"
6 | console.log( {nome:'José'} + 1 ); // “[object Object]1”
```

Conversão para string

Valor original	Valor em string
true	“true”
false	“false”
null	“null”
undefined	“undefined”
array	“valor1, valor2, valor3”
objeto	“[object Object]”

Outros exemplos de conversão de tipos

Somando um valor do tipo number a outros tipos:

```
1  console.log(5 + 20); // 25
2  console.log(5 + true); // 6
3  console.log(5 + null); // 5
4  console.log(true + true); // 2
5  console.log(true + null); // 1
6  console.log(null + null); // 0
7  console.log(null + false); // 0
8  console.log(5 + undefined); // NaN
9  console.log(null + undefined); // NaN
```

Conversão para number

Valor original	Valor em number
true	1
false	0
null	0
undefined	NaN (não é número)

Em qual ordem o JavaScript executa as operações aritméticas?

INSTITUTO FEDERAL
Rondônia
Campus Vilhena

Assim como nas operações matemáticas, os primeiros cálculos feitos são os que possuem os operadores *, / e % e só então + e -.

Caso seja necessário mudar essa ordem, colocamos o que queremos priorizar dentro de parênteses e ele será efetuado primeiro.

Exemplo: (5 + 9) * 5

Nome	Operador encurtado	Significado
Atribuição	x = y	x = y
Atribuição de adição	`x += y`	`x = x + y`
Atribuição de subtração	`x -= y`	`x = x - y`
Atribuição de multiplicação	`x *= y`	`x = x * y`
Atribuição de divisão	`x /= y`	`x = x / y`
Atribuição de resto	`x %= y`	`x = x % y`

Operadores de atribuição aritmética

Existem situações nas quais precisamos gerar um novo valor e atribuí-lo a mesma variável.

Incremento e decremento

INSTITUTO FEDERAL
Rondônia
Campus
Vilhena

Os operadores de incremento `++` e decremento `--`, são usados para adicionar ou subtrair o valor 1 de uma variável.

```
1  var contador = 5;
2  contador++;
3  console.log(contador);// contador agora vale 6
4
5  contador--;
6  console.log(contador);// contador agora vale 5
```

Incremento e decremento

INSTITUTO FEDERAL
Rondônia
Campus
Vilhena

Perceba que **`contador = contador + 1`** ou **`contador += 1`** ou **`contador++`** tem o mesmo efeito, em todos os casos o valor em contador será aumentado em 1.

Assim como **`contador -= 1`** ou **`contador = contador - 1`** ou **`contador --`**, também são iguais, porque em todos os casos o valor em contador será diminuído em 1.

Ordem das operações de incremento ou decremento

Esses operadores podem ser colocados antes ou após o nome de uma variável.

- Quando colocados o operador antes do nome, o valor muda imediatamente:

```
1 | var numero = 9;
2 | console.log(++numero);
```

Neste caso, primeiro o valor será alterado de 9 para 10 e depois ele será impresso.

- Quando o operador é colocado após o nome, o valor muda após a operação atual ser executada.

```
1 | var numero = 9;
2 | console.log(numero++);
```

Neste caso, primeiro o valor 9 será impresso e depois ele será modificado para 10.

Operador	Descrição
Igual (==)	Retorna verdadeiro caso os operandos sejam iguais.
Não igual (!=)	Retorna verdadeiro caso os operandos não sejam iguais.
Estritamente igual (===)	Retorna verdadeiro caso os operandos sejam iguais e do mesmo tipo.
Estritamente não igual (!==)	Retorna verdadeiro caso os operandos não sejam iguais e/ou não sejam do mesmo tipo.
Maior que (>)	Retorna verdadeiro caso o operando da esquerda seja maior que o da direita.
Maior que ou igual (>=)	Retorna verdadeiro caso o operando da esquerda seja maior ou igual ao da direita.
Menor que (<)	Retorna verdadeiro caso o operando da esquerda seja menor que o da direita.
Menor que ou igual (<=)	Retorna verdadeiro caso o operando da esquerda seja menor ou igual ao da direita.

Operadores relacionais

Os operadores de comparação servem para comparar dois valores, retornando um booleano (true ou false).

Operadores ==, !=

O operador de igualdade **==** compara dois valores e retorna true se eles forem iguais.

```
1 | console.log( 21 == 21 ); // vai imprimir true
2 | console.log( 120 == 100); // vai imprimir false
```

Quando precisamos verificar se um valor é diferente do outro, utilizamos o operador de desigualdade **!=**.

```
1 | console.log( 11 != 21 ); // vai imprimir true
2 | console.log( 100 != 100); // vai imprimir false
```

Esses operadores conseguem fazer a comparação, mesmo que os valores sejam de tipos diferentes.

```
1 | console.log(  "20" == 20  ); // true
2 | console.log(  true == 1 ) ; // true
```

Operadores de igualdade e desigualdade estrita

O operador de igualdade estrita (===) assim como o operador de igualdade (==), vai comparar se dois valores são iguais. A diferença entre os dois é que o operador === não faz conversão de tipo, ou seja, ele vai comparar se os dois valores são iguais tanto em valor quanto em tipo.

```
1 | "21" === 21 // o resultado será false
2 | 1 === true // o resultado será false
```

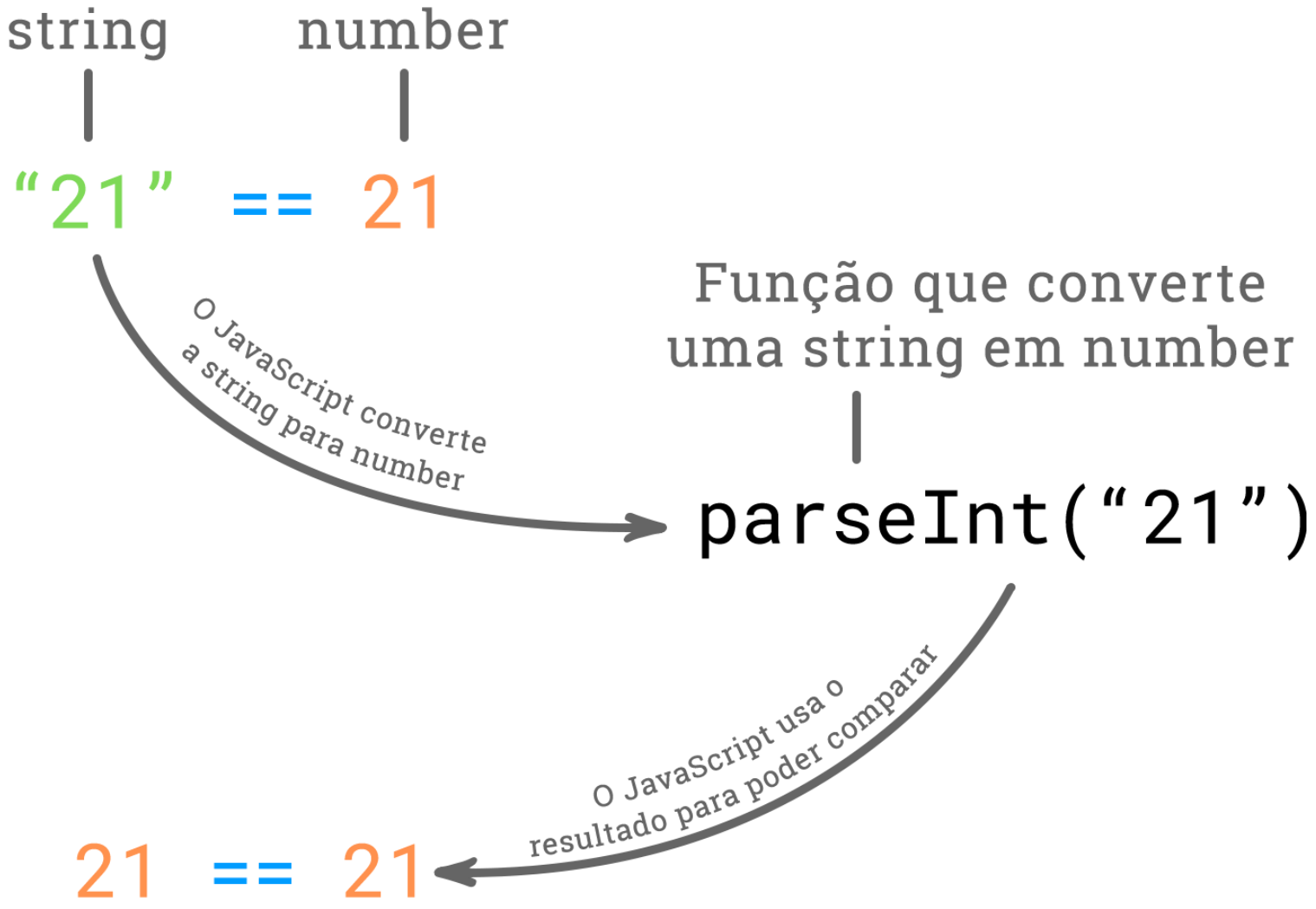
O mesmo vale para o operador de desigualdade estrita (!==).

```
1 | "21" !== 21 // o resultado será true
2 | 1 !== true // o resultado será true
```

Conversão de tipo

Ao utilizar o operador de igualdade o JavaScript vai verificar se os dois valores possuem o mesmo tipo. Se não for esse o caso, o JavaScript vai converter os valores da seguinte forma:

Se um dos valores for um **number** e o outro uma **string**, ele vai converter a string em um number e só então realiza a comparação.



Conversão de tipo

INSTITUTO FEDERAL
Rondônia
Campus Vilhena

- null e undefined são iguais:

```
1 | null == undefined // o resultado será true
2 | null != undefined // o resultado será false
```

- true é convertido para 1 e false para 0:

```
1 | true == 1 // o resultado será true
2 | true != 1 // o resultado será false
3 | false == 0 // o resultado será true
4 | false != 0 // o resultado será false
```

Mesmo que usar os operadores == e != pareça ser mais fácil, podemos ter alguns problemas em nossas validações, já que para eles não importa se os tipos são diferentes. O mesmo não vai ocorrer para os operadores === e !== já que com eles conseguimos ter precisão na nossa comparação.

Operador	Descrição
AND lógico (&&)	Retorna verdadeiro caso ambos operandos sejam verdadeiros; caso contrário, retorna falso.
OU lógico (||)	Retorna verdadeiro caso um dos operandos seja verdadeiro; se ambos forem falsos, retorna falso.
NOT lógico (!)	A negação resulta no valor contrário do operador, ou seja, true será false e false será true

Operadores lógicos

Os operadores lógicos comparam dois valores booleanos e retorna um valor true ou false.

Operadores lógicos

INSTITUTO FEDERAL
Rondônia
Campus Vilhena

É comum usarmos estes operadores lógicos, para combinar operações relacionais:

```
1 | 1 == '1' || "a" == "a" // true
2 | 35 === '35' && 1 == 10 // false
3 | !( 'a' == 1) // true
```

Operador ternário

INSTITUTO FEDERAL
Rondônia
Campus
Vilhena

O operador **ternário** é o único operador JavaScript que possui três operandos. Este operador é frequentemente usado como um atalho para a instrução **if**.

Sintaxe: `condition ? expr1 : expr2`

Se **condition** é true, o operador retornará o valor de expr1; se não, ele retorna o valor de exp2.

```
let idade = 25;
let mensagem = idade >= 18 ? "Acesso permitido" : "Acesso negado";
console.log(mensagem);
```

Exercícios

INSTITUTO FEDERAL
Rondônia
Campus
Vilhena

1. Crie um programa que declare duas variáveis e exiba o resultado da soma, subtração, multiplicação e divisão desses números.
2. Declare duas idades e utilize operadores de comparação para verificar se uma pessoa é mais velha que a outra.
3. Crie uma variável idade com um número e use o operador ternário para verificar se o valor armazenado na variável é maior ou igual a 18. Imprima “Maior de idade” ou “Menor de idade”.
4. Crie uma variável com um número e use o operador ternário para verificar se é par ou ímpar.

Para Nerds

INSTITUTO FEDERAL
Rondônia
Campus Vilhena

MDN Web Docs – Java Script
https://developer.mozilla.org/pt-BR/docs/Web/JavaScript

JavaScript Tutorial
https://www.w3schools.com/js/

Cursos Gratuitos

INSTITUTO FEDERAL
Rondônia
Campus Vilhena

Curso JavaScript Curso em Video
https://www.cursoemvideo.com/curso/javascript/

Curso JavaScript YouTube
https://www.youtube.com/watch?v=McKNP3g6VBA