



Desenvolvimento para dispositivos móveis

Prof. Armando Mendes Neto.
armando.mendes@ifc.edu.br

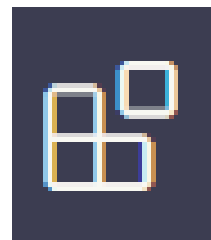
Operadores.

Conteúdo da aula

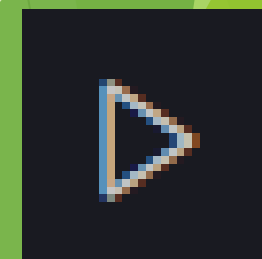
- ▶ Operadores de atribuição.
- ▶ Operadores de comparação.
- ▶ Operadores aritméticos.
- ▶ Operadores lógicos.
- ▶ Precedência de operadores.

Extensão Code Runner

- ▶ Primeiramente, vamos instalar a extensão “Code Runner”.
- ▶ Clique em extensões (ícone na barra lateral a esquerda).
- ▶ Em pesquisar, digite “code runner”.
- ▶ Clique em instalar.



Aparecerá esse ícone para testarmos códigos em JavaScript (Canto superior direito)



Operadores de atribuição

- Um operador de atribuição, atribui um valor ao operando à sua esquerda baseado no valor do operando à direita.

Nome	Operador encurtado	Significado
Atribuição	<code>x = y</code>	<code>x = y</code>
Atribuição de adição	<code>x += y</code>	<code>x = x + y</code>
Atribuição de subtração	<code>x -= y</code>	<code>x = x - y</code>
Atribuição de multiplicação	<code>x *= y</code>	<code>x = x * y</code>
Atribuição de divisão	<code>x /= y</code>	<code>x = x / y</code>
Atribuição de resto	<code>x %= y</code>	<code>x = x % y</code>
Atribuição exponencial	<code>x **= y</code>	<code>x = x ** y</code>

Operadores de atribuição

- Crie a pasta Aula5 e o arquivo “atribuição.js”

```
// Operadores de Atribuição
let n3=20;
n3+=15; //n3 = n3 + 15;
console.log(n3);
/*
Dá para testarmos com todos
+=, -=, *=, /=, %=, **=
*/
```

Faça o teste com
TODOS os
operadores



Operadores de comparação

- Um operador de comparação compara seus operandos e retorna um valor lógico baseado em se a comparação é verdadeira

Operador	Descrição
Igual (<code>==</code>)	Retorna verdadeiro caso os operandos sejam iguais.
Não igual (<code>!=</code>)	Retorna verdadeiro caso os operandos não sejam iguais.
Estritamente igual (<code>===</code>)	Retorna verdadeiro caso os operandos sejam iguais e do mesmo tipo. Veja também object.is e igualdade em JS (en-US) .
Estritamente não igual (<code>!==</code>)	Retorna verdadeiro caso os operandos não sejam iguais e/ou não sejam do mesmo tipo.

Maior que (<code>></code>)	Retorna verdadeiro caso o operando da esquerda seja maior que o da direita.
Maior que ou igual (<code>>=</code>)	Retorna verdadeiro caso o operando da esquerda seja maior ou igual ao da direita.
Menor que (<code><</code>)	Retorna verdadeiro caso o operando da esquerda seja menor que o da direita.
Menor que ou igual (<code><=</code>)	Retorna verdadeiro caso o operando da esquerda seja menor ou igual ao da direita.

Operadores de comparação

- Crie o arquivo “comparação.js”.

```
// Operadores de comparação


let num1=10;
let num2="10";

console.log(num1==num2);
console.log(typeof num1, typeof num2);
console.log(num1===num2);
console.log(num1<=num2);
console.log(num1>num2);
console.log(num1!=num2);
console.log(num1!==num2);
```

Operadores aritméticos

- ▶ Operadores aritméticos tomam valores numéricos (sejam literais ou variáveis) como seus operandos e retornam um único valor numérico.

Operador	Descrição
Módulo (%)	Operador binário. Retorna o inteiro restante da divisão dos dois operandos.
Incremento (++)	Operador unário. Adiciona um ao seu operando. Se usado como operador prefixado (++x), retorna o valor de seu operando após a adição. Se usado como operador pósfixado (x++), retorna o valor de seu operando antes da adição.
Decremento (--)	Operador unário. Subtrai um de seu operando. O valor de retorno é análogo àquele do operador de incremento.

Negação (-)	Operador unário. Retorna a negação de seu operando.
Adição (+)	Operador unário. Tenta converter o operando em um número, sempre que possível.
Operador de exponenciação (**) 	Calcula a base elevada á potência do expoente, que é, base <code>expoente</code>

Operadores aritméticos

- Crie o arquivo “aritméticos.js”.

```
// Operadores Aritméticos  
let n1=10;  
let n2=5;  
  
console.log(n1+n2);  
console.log(n1-n2);  
console.log(n1*n2);  
console.log(n1/n2);  
console.log(n1%n2);  
console.log(n1**n2);
```

Operadores aritméticos

- Crie o arquivo “incrementodecremento.js”.

```
// Incremento e decremento
let i=0;
//i = i+1, i++ ou i+=1;
i+=1;
console.log(i);
i-=1; //i=i-1, i-- ou i-=1;
console.log(i);
console.log(++i);
console.log(i++);
console.log(i);
console.log(i--);
console.log(--i);
```

Operadores lógicos

- ▶ Operadores Lógico são utilizados tipicamente com valores booleanos (lógicos); neste caso, retornam um valor booleano (Verdadeiro ou Falso).

Operador	Utilização
AND lógico (<code>&&</code>)	<code>expr1 && expr2</code>
OU lógico (<code> </code>)	<code>expr1 expr2</code>
NOT lógico (<code>!</code>)	<code>!expr</code>

Operadores lógicos

- Crie o arquivo “lógico1.js”.

```
let a=10;
let b=11;
let c=12;

console.log(`Primeira condição com && = ${a>b && c>a}`);
console.log(`Segunda condição com && = ${c>b && c>a}`);

console.log(`Primeira condição com || = ${a>b || c<a}`);
console.log(`Segunda condição com || = ${a>b || c>a}`);

console.log(`A negação da primeira expressão é = ${!(a>b)}`);
console.log(`A negação da segunda expressão é = ${!(a<b)}`);
```

Operadores lógicos

- Crie o arquivo “lógico2.js”.

```
let idade=18;
let paisPresentes=true;

console.log(`Pode viajar 1: ${((idade>=18 || paisPresentes==true))}`);
console.log(`Pode viajar 2: ${((idade>18 || paisPresentes==true))}`)
console.log(`Pode viajar 3: ${((idade>18 || paisPresentes==false))}`)

let comprouBilhete=true;
console.log(`Pode viajar 4: ${(((idade>18 || paisPresentes==true) && comprouBilhete==false))}`)
console.log(`Pode viajar 5: ${(((idade>=18 || paisPresentes==false) && comprouBilhete==true))}`)
console.log(`Pode viajar 5: ${(((idade>=18 || paisPresentes==true) && comprouBilhete==true))}`)
```

Precedência de operadores

- A *precedência* de operadores determina a ordem em que eles são aplicados quando uma expressão é avaliada.

Precedência:

()

**

/ *

+ -

< >

Precedência de operadores

- Crie o arquivo “precedência.js”

```
// Precedência de Operadores  
let a=1  
let b=2  
let c=3  
let d=4  
let e=2  
console.log((5+a*(2-b+c*(d/e+5))))
```

Por exemplo

Considere que:

A <- 1
B <- 2
C <- 3
D <- 4
E <- 2

$(5+A*(2-B+C*(D/E+5)))$

► $(5+A*(2-B+C*(4/2+5)))$

► $(5+A*(2-B+C*7))$

► $(5+A*(2-B+3*7))$

► $(5+A*(2-B+21))$

► $(5+A*(2-2+21))$

► $(5+A*21)$

► $(5+1*21)$

► $(5+21)$

► (26)

Precedência:

()

**

/ *

+ -

< >

Precedência de operadores



Tabela verdade E, OU, NÃO

1º Valor	Operador	2º Valor	Resultado
V	E	V	V
V	E	F	F
F	E	V	F
F	E	F	F
V	OU	V	V
V	OU	F	V
F	OU	V	V
F	OU	F	F
V	NÃO		F
F	NÃO		V