Tipos de operadores:

1) Atribuição

Operador	Exemplo	Equivalente a
=	x = y	x = y
+=	x += y	x = x+y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y

Posso atribuir valores a uma variável utilizando ***= /= e %=**.

É a mesma ideia de quando usamos +=. Por exemplo: x += y significa x = x+y.

Então se fizer isso utilizando *multiplicação, divisão* ou *módulo* tem a mesma finalidade. Dessa forma atribuo o valor das operações à **x**:

x *=y : o novo valor de x será o resultado da multiplicação dele por **y.**

x /=y : o novo valor de x será o resultado da divisão dele por **y.**

x %=y : o novo valor de x será o resultado do módulo dele por **y.** (Lembrando que módulo retorna o resto da divisão de x por y).

2) Aritméticos

Operador	Descrição
+	Adição
-	Subtração
*	Multiplicação
**	Exponencial
/	Divisão
%	Módulo
++	Incrementar
--	Decrementar

Posso incrementar usando: ++ . Por exemplo: **x=0; x++** então acrescento 1 à x e então x=1.

Para decrementar usamos **– .** Por exemplo: **x=10; x--** então diminuo 1 de x e então x=9.

3) Comparação

Operador	Descrição
==	Igual a
===	Mesmo valor e mesmo tipo
!=	Diferente
!==	Valor e tipos diferentes
>	Maior que
<	Menor que
>=	Maior ou igual
<=	Menor ou igual

2 == 2 comparo valor de uma variável, ou número com o valor de outra variável, podendo ser até uma string por ex.

Já 2 === 2 comparo o valor e o tipo. Então caso uma seja número e outra string, o resultado será falso, pois elas não são totalmente iguais, apenas o valor, pois o tipo é diferente!

Mesma coisa para **!=** e **!==** só que neste caso não vai comparar igualdade, mas sim diferença!

4) Lógicos

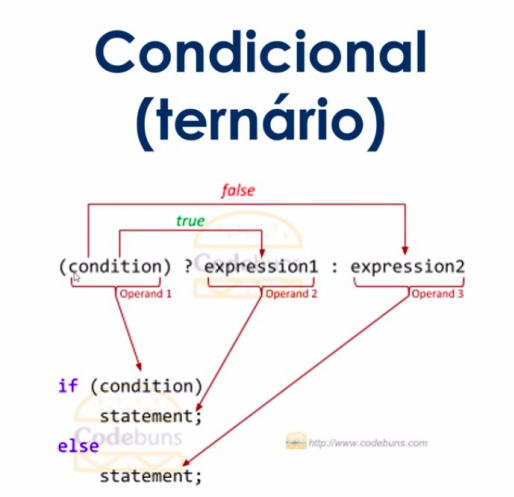
Operador	Descrição
&&	“e” lógico
||	“ou” lógico
!	“não” lógico

&&: comparo se duas afirmações são verdadeiras, ou seja, temos uma afirmação e outra afirmação, então se ela e a próxima forem verdadeiras, será executada determinada instrução.

|| : aqui comparo também duas afirmações, porém neste caso não é necessário as duas serem verdadeiras para executar uma ação, basta uma delas ser para que isso aconteça.

! Acessamos um dado oposto ao que foi determinado. Por exemplo: se uso um não logico seguido de true quero receber um valor falso: **! true**

5) Condicionais



Esse tipo de estrutura é igual a utilização do ifelse, porém é menos verbosa.

Antes da interrogação coloco minha condição, ali estou perguntando se minha condição é verdadeira.

Depois declaro a ação a ser feita, caso seja verdadeira.

E por último, declaro a ação caso seja falsa, como se fosse o *else* mesmo.

Esta expressão está sendo muito mais utilizada do que if else!

Atividade

Crie uma função que recebe dois números como parâmetros. Confira se os números são iguais. Confira se a soma dos números é maior que 10 ou menor que 20.

Retorne uma string dizendo "Os números **num1** e **num2** não/são iguais. Sua soma é **soma**, que é maior/menor que 10 e maior/menor que 20".

Exemplo:

Input: 1, 2

Output: Os números 1 e 2 não são iguais. Sua soma é 3, que é menor que 10 e menor que 20.

Solução

Primeiro vamos criar a função recebendo os dois números e também uma constante para comparar se eles são iguais ou não:

```
function comparaNumeros(num1, num2) {
  const saoIguais = num1 === num2;
  const soma = num1 + num2;
```

a constante é um boolean, vai retornar true ou false da comparação dos números.

Vamos criar também uma constante para guardar o valor da soma dos números.

Para verificar se as variáveis são iguais:

```
if(saoIguais) {
  return "São iguais";
}
return "Não são iguais";
```

Crio a condicional para verificar se os números são iguais e retornar a frase: “São iguais”.

Nem preciso colocar o eles, pois se o if não for verdadeiro ele já vai cair pro próximo return “Não são iguais”.

Podemos usar o **if ternário** para ficar menos verboso:

```
return saoIguais ? "São iguais" : "Não são iguais";
```

Ao invés de isso tudo, Stephany separou tudo em funções auxiliares:

Dividimos o problema em dois: vamos criar a primeira frase “Os numeros x e y não são iguais (ou são iguais)” e depois a segunda frase “Sua soma é z, que é menor que 10 e menor que 20”.

Criando a primeira frase:

Criar uma constante para guardar os valores retornados pela função **criaPrimeiraFrase(num1, num2);**

```
function comparaNumeros(num1, num2) {
  const primeiraFrase = criaPrimeiraFrase(num1, num2);

}
```

Logo em seguida criou a função:

```
function criaPrimeiraFrase(num1, num2) {
  let saoIguais = '';

  if(num1 !== num2) {
    saoIguais = 'não';
  }
```

Com **let** vamos criar uma string auxiliar que pode mudar caso a comparação seja verdadeira ou falsa. Vamos deixá-la vazia, e caso **não** forem iguais vamos colocar *‘não’* como seu valor.

Repare que a frase sempre terá “são iguais” adicionando o “não” na frente caso seja falso:

“Os numeros x e y **não** são iguais” ou “Os numeros x e y são iguais”

Por isso é mais fácil fazer a comparação com **num1 !== num2**.

Agora vamos fazer o **return** dessa função:

```
function criaPrimeiraFrase(num1, num2) {
  let saoIguais = '';

  if(num1 !== num2) {
    saoIguais = 'não';
  }

  return `Os números ${num1} e ${num2} ${saoIguais} são iguais.`
}
```

Se num1 e num2 forem diferentes então **saoIguais = 'não'.**

Se for iguais, **saoIguais = '';** (string vazia)

No **return** coloco a primeira frase que será construída com as variáveis.

Criando a segunda frase:

Dentro da função **comparaNumeros** vou declarar minha constante que irá armazenar o *return* da função **criaSegundaFrase:**

```
function comparaNumeros(num1, num2) {
  const primeiraFrase = criaPrimeiraFrase(num1, num2);
  const segundaFrase = criaSegundaFrase(num1, num2);
}
```

Agora criamos as constantes que irão armazenar os valores para a soma e as comparações a serem feitas:

```
function criaSegundaFrase(num1, num2) {
  const soma = num1 + num2;
  const compara10 = soma > 10;
  const compara20 = soma > 20;
```

As comparações retornarão valores booleanos.

```
let resultado10 = 'menor';
let resultado20 = 'menor';
```

Vamos criar uma variável para retornar o valor 'menor' com **let** e não const, pois queremos que esse valor altere para 'maior' quando for necessário.

```
if(compara10) {
  resultado10 = 'maior';
}

if(compara20) {
  resultado20 = 'maior';
}
```

Agora vamos comparar:

Se **compara10** for verdadeiro (ou seja soma > 10) o valor da variável **resultado10** será 'maior'.

Mesma coisa para **compara20**.

O return fica assim:

```
return `Sua soma é ${soma}, que é ${resultado10} que 10 e ${resultado20} que 20.`
```

Agora, precisamos do *return* para a função **comparaNumeros:**

```
function comparaNumeros(num1, num2) {
  const primeiraFrase = criaPrimeiraFrase(num1, num2);
  const segundaFrase = criaSegundaFrase(num1, num2);

  return `${primeiraFrase} ${segundaFrase}`;
}
```

Vamos chamar a função com **console.log:**

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

E rodar com Node para ver se funcionou.

Abra o terminal no VSCode e digita:

node nomedoarquivo.js

Caso não esteja na pasta certa, é preciso navegar com o terminal (utilizando cd) e ir até a pasta onde o arquivo da atividade está salvo!

```
→  javascript cd 01-sintaxe-e-operadores
→  01-sintaxe-e-operadores node atividade.js
Os números 1 e 2 não são iguais. Sua soma é 3, que é menor que 10 e menor que 20.
→  01-sintaxe-e-operadores
```

Temos alguns probleminhas nesse código, por exemplo: se não inserirmos valor nenhum, dará um erro. Ou seja, ele permite a inserção de qualquer coisa.

Vamos fazer uma validação, checar se o input é um número mesmo, e retornar uma mensagem, para que esse erro não ocorra, forçando o usuário a inserir um número válido:

```
function comparaNumeros(num1, num2) {
  if(!num1 || !num2) return "Defina dois números!";
```

Assim estou dizendo que se o **num1** não existe OU **num2** também não existe, retorne a mensagem “Defina dois números!”.

Neste caso acima, quando o número zero for inserido retornará também essa mensagem, pois essa verificação **!num1** considera: strings vazias, null, undefined, false, 0 e NaN (Not a Number) como não existentes, ou seja, como diferentes de números (!num1).

Precisamos pensar em uma solução que nosso código permita entrada do número 0 sem retornar a mensagem “Defina dois números!”.

Para isso tive que criar condicionais dentro da função **comparaNumeros:**

Primeiro para aceitar valores zero, comparei **num1** e **num2** a 0, trazendo como retorno dessa condição as **const** primeiraFrase e segundaFrase.

Depois fiz uma segunda comparação igual da Stephany, verificando se os números são strings vazias, null, undefined, false ou NaN (Not a Number) com **!num1 || !num2** retornando a msg “Defina um número”.

Em seguida coloquei else para retornar as frases caso nenhuma das condições acima sejam satisfeitas:

```
function comparaNumeros(num1, num2) {
  if(num1 == 0 || num2 == 0){
    const primeiraFrase = criaPrimeiraFrase(num1, num2);
    const segundaFrase = criaSegundaFrase(num1, num2);
    return `${primeiraFrase} ${segundaFrase}`;
  } else if(!num1 || !num2) {
    return "Defina um número!";
  } else {
    const primeiraFrase = criaPrimeiraFrase(num1, num2);
    const segundaFrase = criaSegundaFrase(num1, num2);
    return `${primeiraFrase} ${segundaFrase}`;
  }
}
```