**OPERADORES**

**OPERADORES ARITMÉTICOS**

Os operadores aritméticos são utilizados na computação para cálculos matemáticos. Os resultados de cálculos aritméticos são os que você já conhece. Um console.log() do Javascript tem o poder de resolver o que estiver contido dentro dos parênteses e exibir o resultado, ou o erro, no console do navegador. Dessa forma, um *console.log(8+2)* consegue imprimir normalmente o valor do cálculo no console. O mesmo acontece quando qualquer operador aritmético for utilizado.

Os operadores que precisam de atenção nesta subunidade são os *exponencial (^)* e *módulo(%)*. Uma vez que não temos costume de representá-los de forma corriqueira como os demais operadores.

O operador exponencial *o famoso “elevado a”* nas nossas aulas, com o JS, irá ser representado por \*\* como no exemplo *console.log(2\*\*3) // Resultado 8*

O operador (^) será utilizado como operador lógico o qual veremos mais a seguir.

O operador módulo quer dizer “*resto da divisão”.* Ou seja, 8 módulo 2 tem resto 4. Exemplo: *console.log(12%8) // resultado 4*

Um detalhe importante é que o operador de Soma também é utilizado para trabalhar com strings ou caracteres realizando a *concatenação* como no exemplo: console.log(“Ao infinito e “ + “além”)

Incremento:

O operador de incremento (decremento) incrementa (decrementa) (adiciona *- ou subtrai -*  um a) seu operando e retorna um valor;

* Se usado na forma posfixa, com o operador depois do operando (por exemplo, x++ ou x--), então ele retorna o valor antes de incrementá-lo ou decrementá-lo.
* Se usado na forma prefixa, com o operador antes do operando (por exemplo, ++x ou --x), então ele retorna o valor depois de incrementá-lo ou decrementá-lo.

var a = 4;

console.log(a++) // retorna 4

console.log(a) // retorna 5

console.log(++a) // retorna 5

console.log(a) // retorna 5

console.log(a--) // retorna 4

console.log(a) // 3

console.log(++a)// retorna 3

console.log(a) // retorna 3

Na computação, os operadores aritméticos utilizam a mesma ordem de prioridade da matemática para resolver as expressões. Como segue nos exemplos:

console.log(((2 \* 8 ) - 2) /2) // prioridade está na utilização dos parênteses

console.log(2 \* 8 - 2 / 2) // prioridade: divisão, multiplicação e subtração

console.log(2 + 8 - 2 + 2) // a prioridade da soma e subtração é igual, então realiza-se a expressão da esquerda para a direita

**OPERADORES RELACIONAIS**

Sempre retornará um valor lógico. Que, como já sabemos, é um VERDADEIRO ou FALSO.

console.log(2\*4 == 24/3) // true

console.log ((14%4) < 1) // false

console.log((2+(8%7)) >=3) // true

Atenção para o uso correto dos parênteses e para a leitura natural, quando não houver prioridades definidas, que se dará sempre da esquerda para a direita.

**OPERADORES LÓGICOS**

A multiplicação lógica E, que no Português Estruturado (ou representação binária *bitwise AND*) é representada por ^, e você pode confundir com o operador aritmético de multiplicação \* mas não vai (!!!), é representada pelas linguagens de programação, mais comumente, pelo operador lógico &&.

Observe em console.log (true && false) // false

A Adição Lógica OU, que no Português Estruturado (ou representação binária bitwise XOR) é representado por +, e você pode confundir com o operador aritmético + mas não vai (!!!), é representada pelas linguagens de programação, mais comumente, pelo operador lógico ||. Perceba abaixo:

console.log(true || false) // true

console.log(false || false) // false

Como as expressões são avaliadas da esquerda para a direita, os valores são testados em “curto circuito”, que sugere que:

* falso && (*qualquer coisa)* é avaliado como falso através de curto-circuito.
* true || (*qualquer coisa)* é avaliado como verdadeiro através de curto-circuito.

Ordem de precedência entre os operadores lógicos também é um ponto importante para nos atentarmos, principalmente porque não vamos lidar sempre com expressões de 2 valores. Basicamente a ordem padrão é *da esquerda para a direita*, mas quando os parênteses são usados, a história muda. O que está entre parênteses ganha prioridade. Perceba abaixo:

O resultado de *console.log (false && (true || true ))* é *false*

O resultado de *console.log (false && true || true )* é *true*

Observe a ordem de precedência da segunda expressão e veja que && tem precedência sobre ||. Seria como se a expressão abaixo estivesse sendo resolvida:

O resultado de *console.log ((false && true) || true )* é *true*

Então, possui parênteses? Não. Olha pra relação de precedência dos operadores. Sim. Executa o que está dentro do parêntese primeiro.