**JavaScript**

****

...

**Variáveis e tipos primitivos**

**Comentário**

* **//** Comentário de uma linha
* **/\*\*/** Comentário de mais de uma linha

**Identificadores**

São os nomes de cada variável.

* Podem começar por uma letra, $ ou \_
* Não podem começar com números
* É possível usar letras ou números
* É possível usar acentos e símbolos
* Não podem conter espaços
* Não podem ser palavras reservadas

Maiúsculas e minúsculas fazem diferença.

Usar nomes coerentes para as variáveis.

**Tipos primitivos (Data Types)**

Os três principais são os destacados:

* **Number**
  + Infinity
  + NaN (Not a Number)
* **String**
* **Boolean**
* Null
* Undefined
* Object
  + Array
* Function

Como guardar as informações do promp dos scripts?

Crio uma variável para receber as informações. Exemplo:



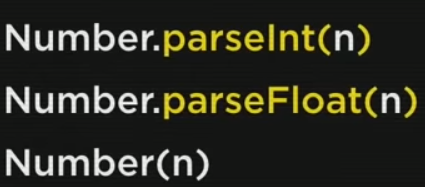
Para usar a variável, podemos usar o seguinte comando:



É necessário concatenar a variável com a String para que ela apareça no alert.

Window.prompt retorna uma string mesmo se for digitado um number

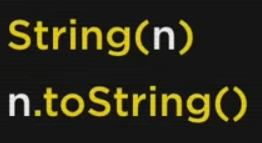
**Para converter string para number, temos três maneiras possíveis:**



A terceira opção deixa a cargo do Javascript decidir se o número vai ser inteiro ou real.

Posso converter direto na variável que recebe a informação ou, posteriormente, na soma dessas variáveis, por exemplo.

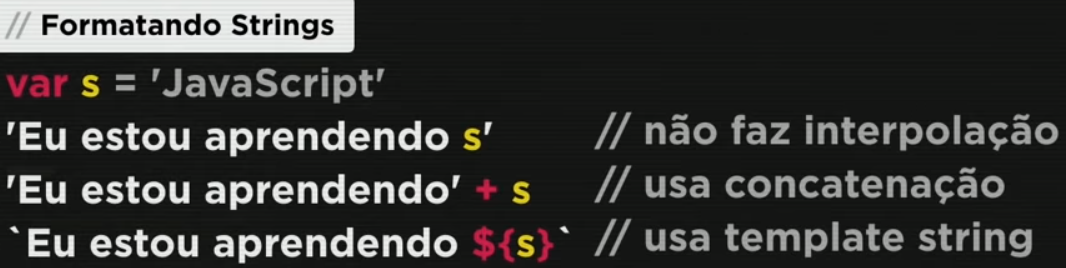
Para converter um número em uma string, temos duas opções:

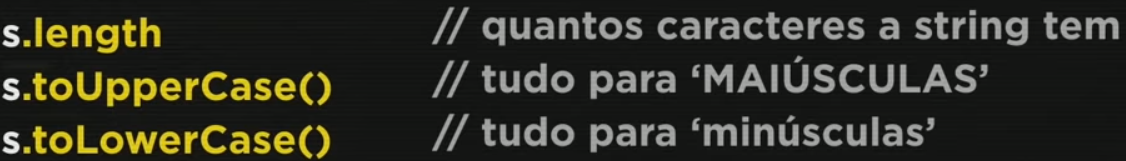


O **n** é o que eu quero converter para string. Exemplo:

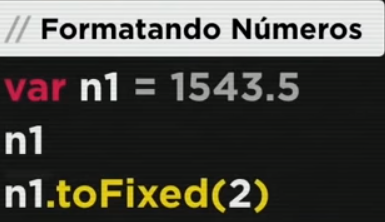


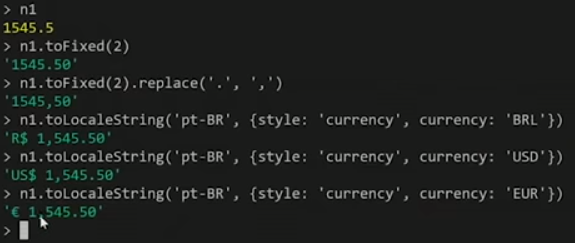
**Formando uma string:**

  
Outras opções de atributos:



Formatando números:



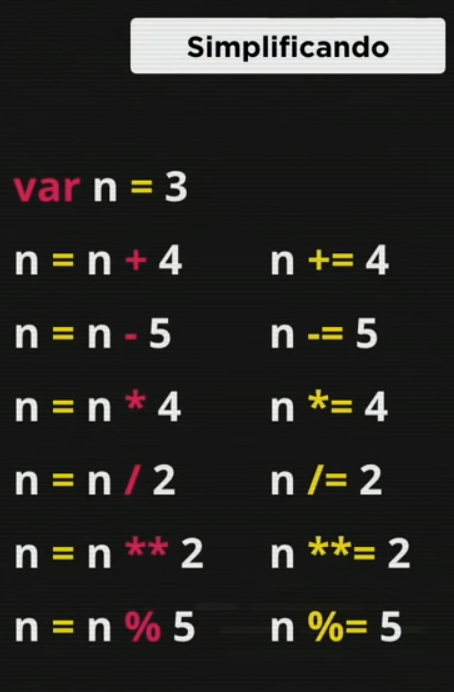


**Operadores**

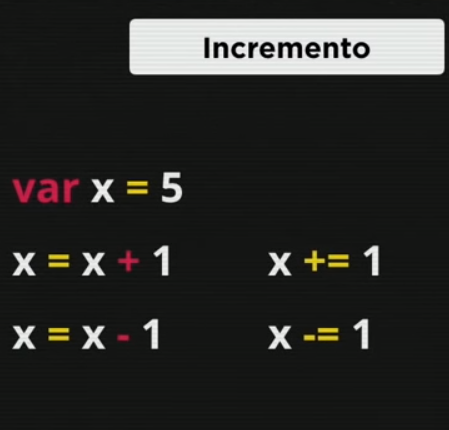
* Aritméticos
  + Soma(+)
  + Subtração(-)
  + Multiplicação(\*)
  + Divisão(/)
  + Divisão inteira (%)
  + Potência(\*\*)
* Atribuição
* Relacionais
* Lógicos
* Ternário

**Ordem de precedência**

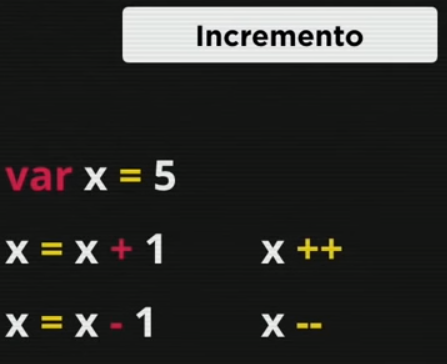
* ()
* \*\*
* \* / % 🡪 se mais de um desses estiver na mesma expressão a ordem de precedência será da esquerda para a direita. Quem aparecer primeiro
* + -



Para incrementar posso usar os seguintes “atalhos”:



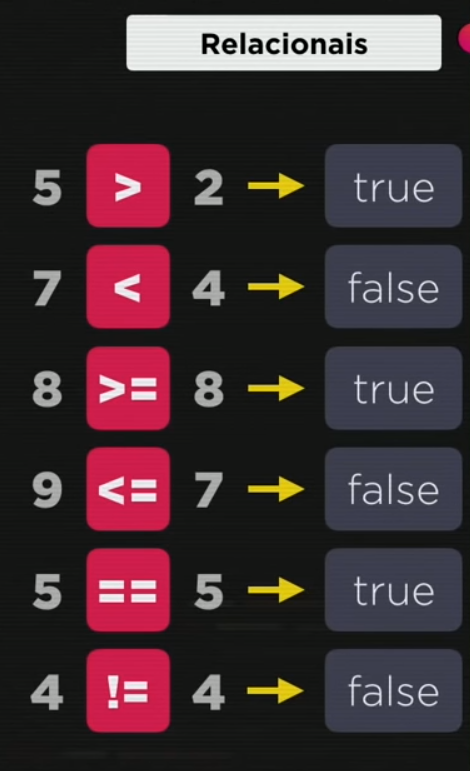
Eles podem ser simplificados desta maneira:

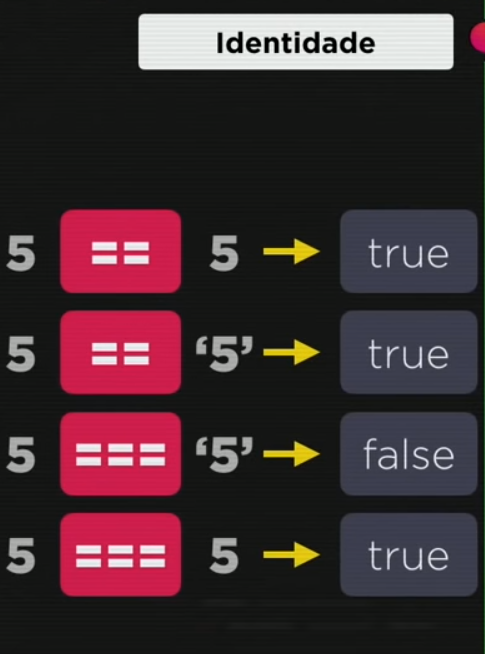


Existe o pré-incremento(quando os operadores vêm antes) e o pós-incremento(quando eles vêm depois). Existe diferença entre eles.

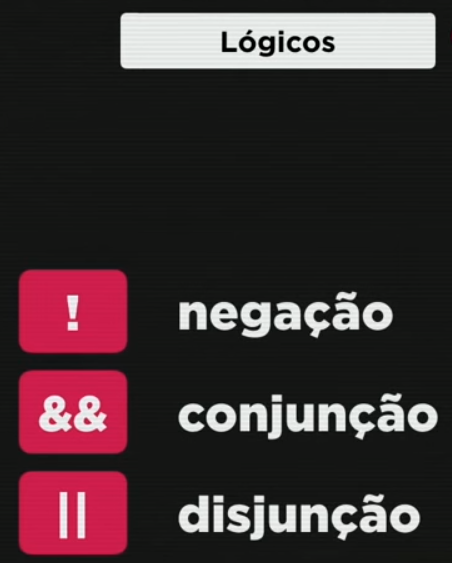
**Operadores relacionais**

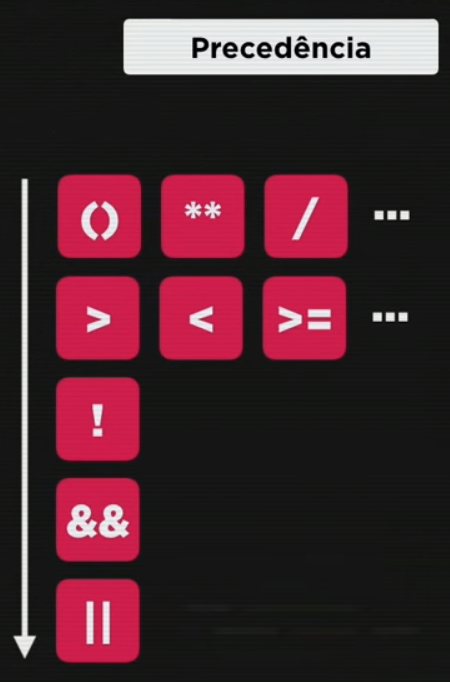
* >
* <
* >=
* <=
* ==
* !=

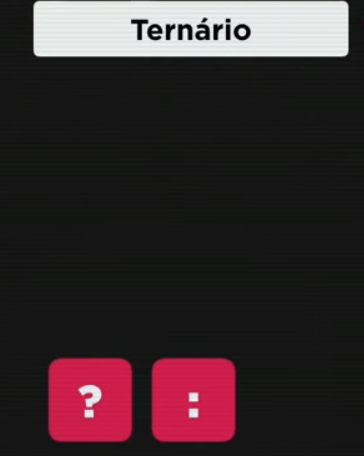




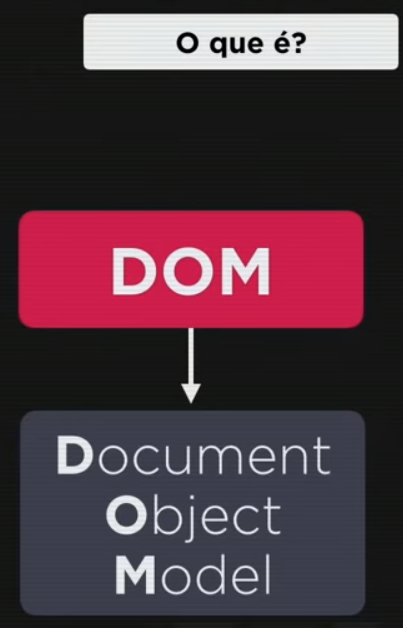
A diferença entre o segundo e terceiro caso é que no segundo ele testa apenas se os valores são iguais, já no segundo ele testa se é o mesmo valor e o mesmo tipo.



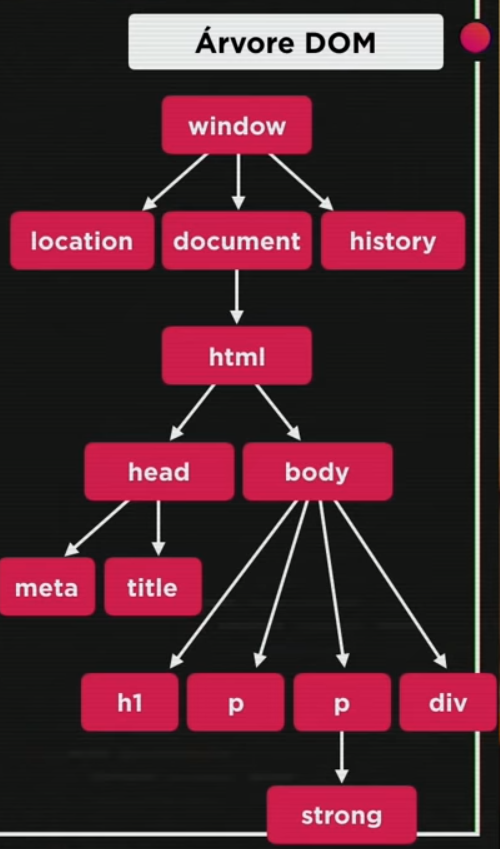




**DOM**

****

Conjunto de objetos dentro do seu computador que vai dar acesso aos componentes internos do seu website.



Exemplo de uma DOM do site criado no curso.

A arvore começa da raiz e essa raiz no navegador a gente chama de **window**.

Dentro do window temos várias outras possibilidades.

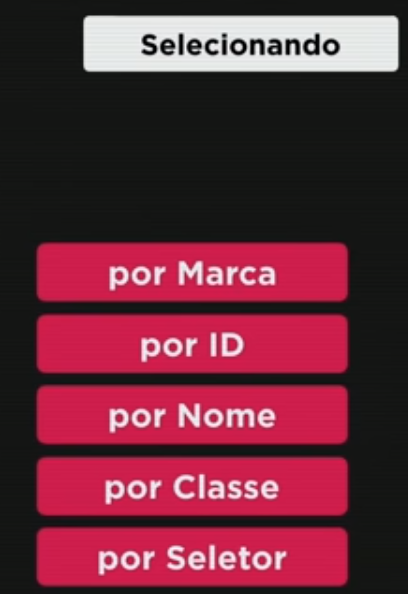
* location: fala qual é a localização do site. Qual é a URL, qual a página atual, qual foi a página anterior...
* document: é o documento atual
* history: vai guardando de onde veio e para onde vai.

Dentro de document temos o html, que é justamente a parte em html do site

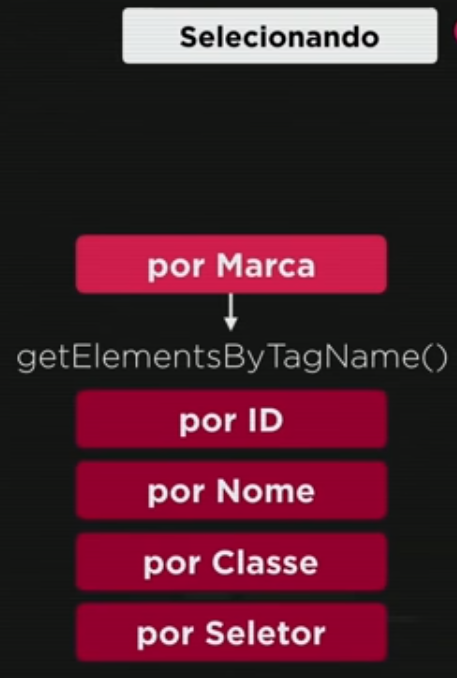
Dentro do html temos dois filhos(chield): head e body.

...

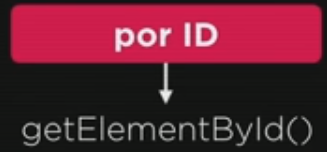
Métodos de acesso a DOM



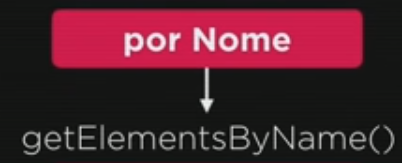
Como seleciona por marca:



Selecionando por ID:

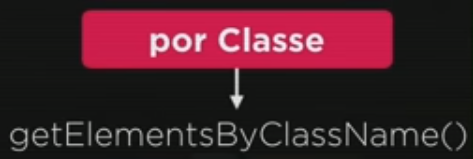


Por Nome:

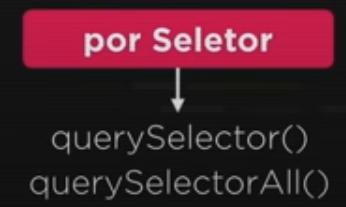


Quando tem Element**s** significa que vai ser mais de um elemento e quando isso acontece eu uso os colchetes no final para definir qual(is).

Por Classe:



Por Seletor:



**Eventos DOM**

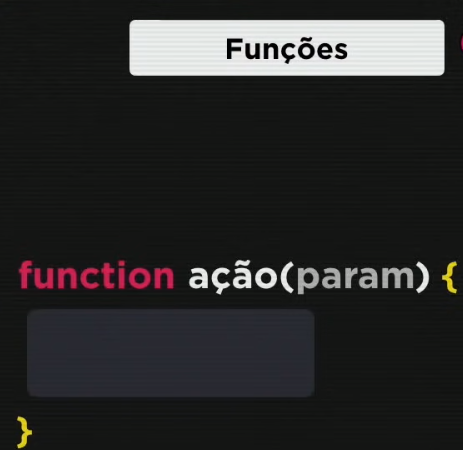
****

* **mouseenter:** quando o mouse entra na div
* **mousemove:** quando o mouse se move dentro da div
* **mousedown:** quando o botão do mouse é apertado e segurado
* **mouseup:** quando o botão do mouse é solto
* **click:** é o clique rápido. Aperta e solta
* **mouseout:** o mouse sai da div

**Existem inúmeros outros eventos**

**Funções**

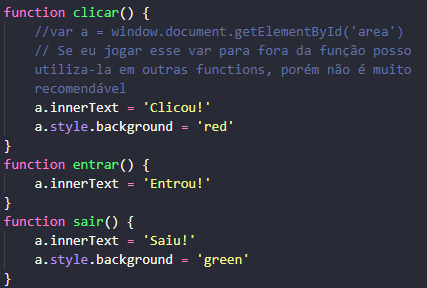
Conjunto de códigos que vão ser executadas quando um evento ocorrer.



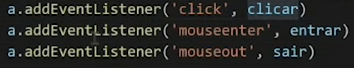
A função pode ser anônima, que é quando não atribuímos um nome para essa função (ação). É possível definir um parâmetro (param).

Esses eventos que estão dentro da div vão ser chamados por JavaScript através de um getElementyById() para modificar algo dentro da div.

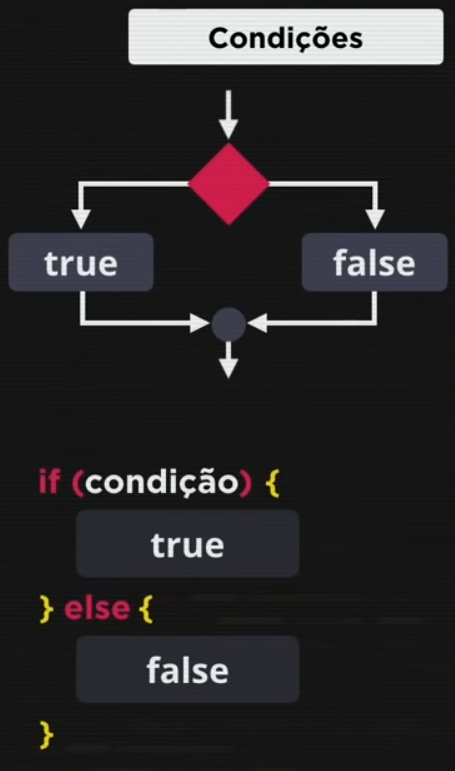


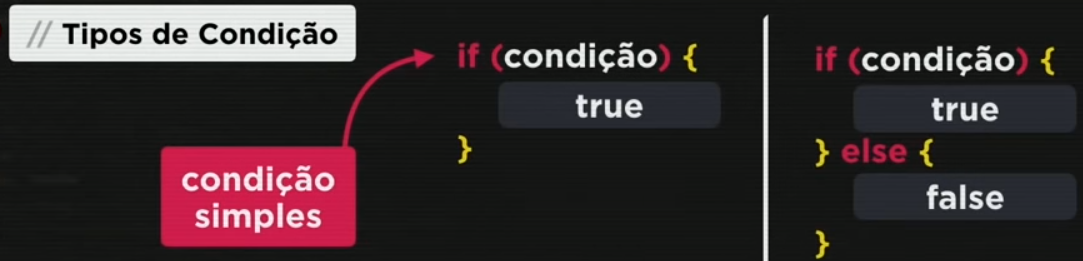


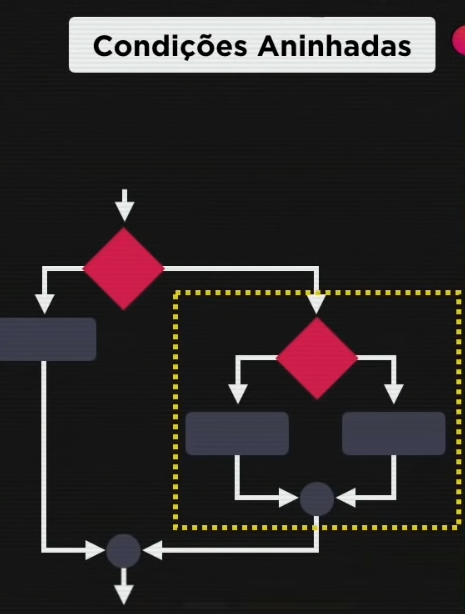
Posso criar ouvidores (listeners) que ficam prestando atenção dentro do próprio JavaScript. Serve como opção a escrita do código no próprio HTML, como foi feita na segunda figura acima, para não deixa o HTML muito poluído.

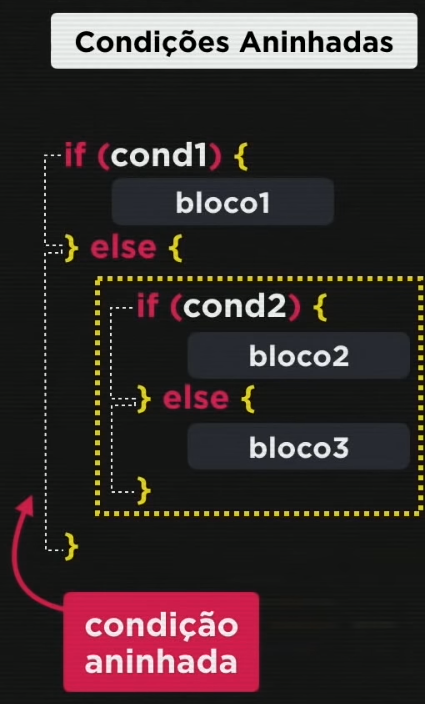


**Condições em JavaScript**





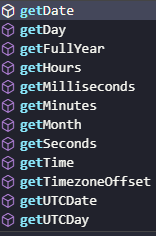




Posso pegar a hora atual do sistema e usar no código.

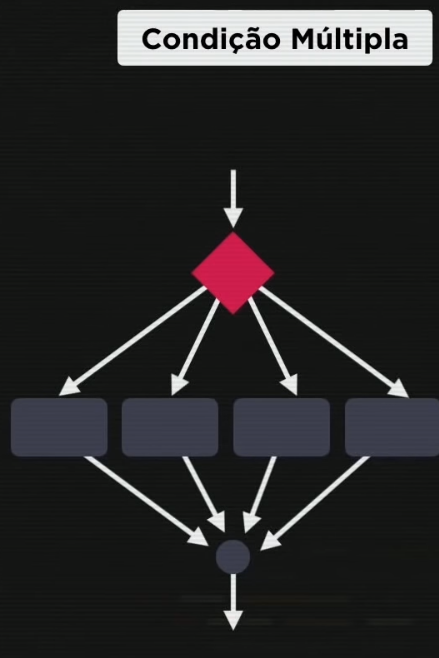


Existem outras opções disponíveis, como:

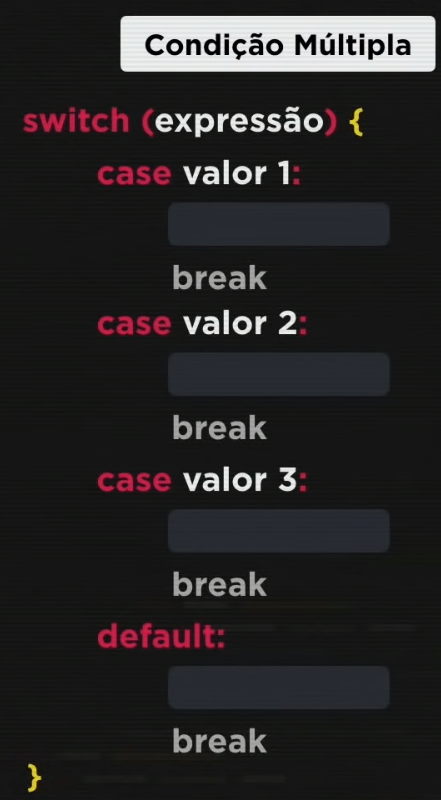


Data, dia, ano, milissegundos, minutos, mês, entre outros.

**Condição múltipla (Switch)**

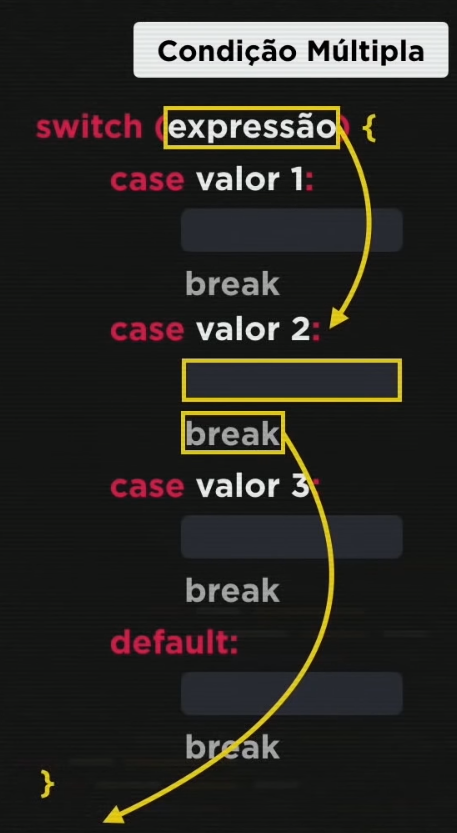


Não serve muito para intervalo de valores, mas para valores fixos/pontuais. Funciona com números inteiros e strings.

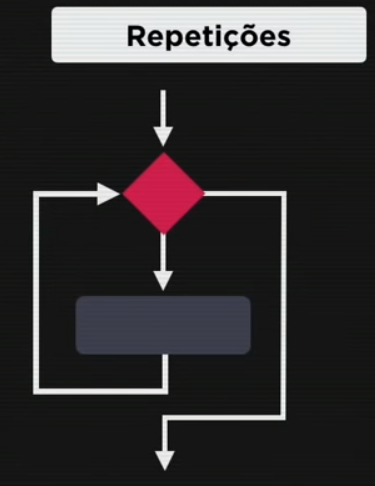


**Default** é como se fosse o else do If. Caso nenhuma condição anterior seja satisfeita ele ira usar o default.

**Break** vai ser utilizado dentro de cada case, obrigatoriamente, sendo opcional no default, mas recomendável que seja usado mesmo assim. Se o break não estiver escrito, mesmo que a condição seja satisfeita, ele continuara executando os demais cases.

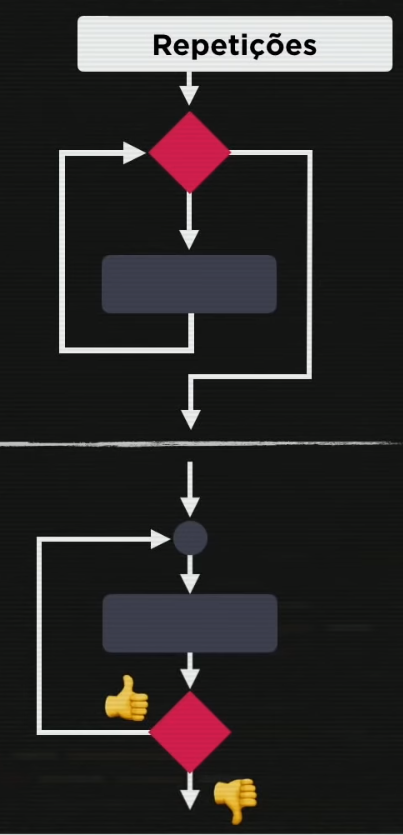


**Repetições**

****

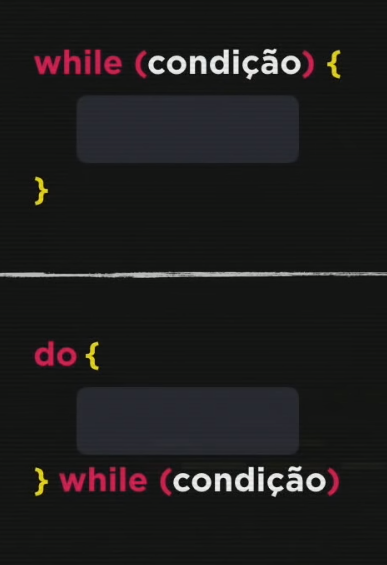
O laço é repetido até que a condição seja satisfeita.

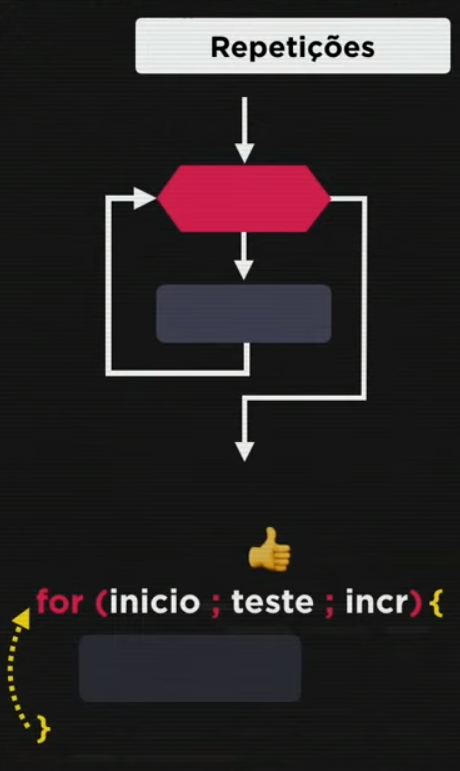
É possível fazer o teste e depois executar o bloco(imagem 1 abaixo) ou fazer o teste e depois executar o bloco(imagem 2 abaixo).



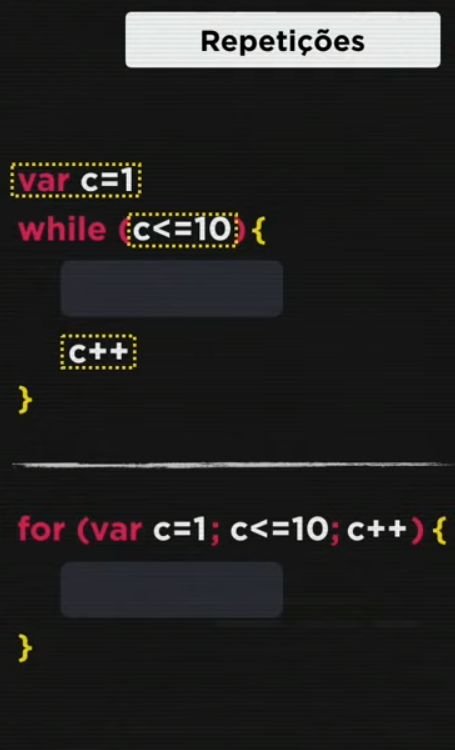
While: estrutura de repetição com teste lógico no início

Do/While: estrutura de repetição com teste lógico no final.

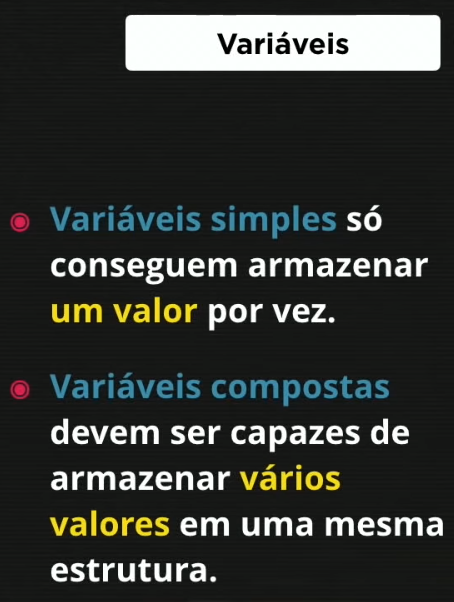




As duas opções abaixo fazem a mesma coisa, só estão escritas de maneiras diferentes.



**Variáveis Compostas**



Array/Vetor/Variável Composta: é uma variável com vários espaços dentro. Para eu não precisar criar 50(exemplo) variáveis, posso criar um array e cada elemento dentro do array vai se tornar uma variável.

Cada elemento dentro de um array/vetor é representado por um índice ou chave.

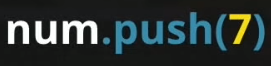


Posso **inclui algo** dentro do array dessa forma:



Com isso estou dizendo que no índice três será incluído o número 6.

Para **colocar na última posição** usamos o comando .push(), por exemplo:



Para descobrir o **comprimento** de um array usamos .length:

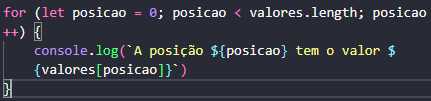


Obs: Não tem parênteses depois do length

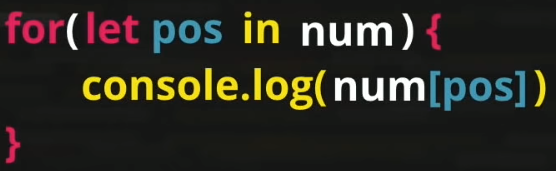
Para colocar os elementos em ordem crescente usamos .sort():



Para mostrar o índice e o valor de cada elemento podemos usar uma estrutura de repetição para não ter que escrever um console.log para tantas quantas forem a quantidade de elementos. Exemplo:



É possível simplificar essa estrutura de repetição desta maneira:



Lê-se: para cada pos(posição) dentro de num mostre o num[pos].

Essa sintaxe do **for** só funciona para arrays e objetos. Todo array em JS é um objeto.

Quando quero buscar um valor dentro de um array uso:



Neste caso, estou buscando o valor 7 dentro do array. Ele vai devolver o índice do valor que eu pedi.

Quando não existe o valor especificado é retornado o valor -1.