# Javascript

Do básico ao avançado seguindo a MDN web docs

*João Vitor Rezende Moura - 1221184773*

**UNIT - Universidade Tiradentes**

---

**Abstract**

abstract

**Keywords:** Javascript, Front-End, UI, media-queries, web

---

## 1 Primeiros passos com javascript

## 2 Criando elementos em javascript

### 2.1 Condicionais dentro do javascript

Dentro de qualquer linguagem de programação, temos que tomar decisões, sejam elas baseadas nas informações providas pelo usuário, nas informações que conseguimos captar dele, ou seja por algum outro motivo. Esse tipo de escolha, pode ser feito por meio de uma estrutura que existem em todas as linguagens de programação, e que recebe o nome de ***condicionais***.

Elas permitem que a partir de determinada condição que deve ser analisada, executarmos uma determinada ação, ou outra. Um exemplo de código javascript que possui essa determinada estrutura, é a seguinte:

```
if(condição){
  // codigo executado se a condição for verdadeira
}else{
  // código executado se a condição for falsa
}
```

A priori, temos que as palavras `if` e `else` são reservadas dentro do javascript para a construção de estruturas condicionais, portanto temos que dentro dos parênteses, temos a condição que deve ser analisada, e dentro das chaves, os blocos de códigos a serem executados condicionalmente.

A posteriori, temos que as vezes as comparações condicionais de dois fatores podem não ser suficientes para nós conseguirmos analisar o código e executá-lo da maneira desejada, portanto, temos a necessidade de comparar um determinado valor a várias possibilidades as quais ele pode assumir, e assim surge o `else if`:

```
if(primeiraCondicao){
  // executar código do primeiro bloco
}else if(segundaCondicao){
  // executar o segundo bloco de código
}else{
  // executar bloco de código quando condições não forem atingidas
}
```

## 2.2 Aninhando condicionais, e estabelecendo condições

Temos que as condições presentes analisandas dentro das condicionais, devem assumir valores booleanos, ou seja, de verdadeiro ou falso. Entretanto, podemos criar análises numéricas, decimais, e de outros tipos primitivos, que possuem como resultado, um valor booleano, e inserir como a condição que controla o fluxo da condicional, como no seguinte caso:

```
let condicao = 15 < 20

if(condicao){
  // executar código se a condicao for verdadeira
}else{
  // executar código se a condicao for falsa
```

Temos no exemplo anterior, que uma comparação lógica entre dois elementos numéricos gerou um valor booleano, e isso pode ser feito de várias maneiras por meio dos operadores lógicos, ou por meio de operadores de comparação. Os operadores que possibilitam isso, são os de comparação matemática, e operadores booleanos, que são o `&&` e o `||`, que são de disjunção e conjunção respectivamente

Podemos também aninhar condicionais, ou seja, dentro de alguma determinada condicional, fazer outra condicional que pode comparar o mesmo valor, ou um valor que não possua conexão com o primeiro comparado.

## 2.3 Instrução switch

As isntruções condicionais são muito úteis, entretanto, quando temos uma variável que pode assumir diversos valores, isso cria uma cadenia de condicionais que é dificil de manter no código, e difícil de entender, portanto, é recomendando que quando comparamos condicçoes que podem assumir diversos valores, que usemos a estrutura `switch`.

```
switch(expression){
  case choice1:
    // run code and break the comparation
    break;
  case choice2:
    // run code and break the comparation
    break;
  default:
    // only need to run the code actually
}
```

Nesse caso, temos que dois casos existem, e que quebramos a comparação como última instrução do caso o qual o mesmo corresponde. Isso se deve ao fato de que o switch case pode desejar que uma expressão, encaixe em mais de um caso, sendo assim, o mesmo deve passar pelos dois casos, e executar os dois blocos de isntrução, e isso é feito não quebrando a comparação dos casos, ou seja, não adicionando a instrução `break`. Caso nenhum dos casos seja verdadeiro, ou caso o que seja, não possua a instrução de quebrar a comparação, e seja o único verdadeiro, a expressão vai cair no caso padrão, e após executar as instruções, vai quebrar a comparação por ser o último caso propositadamente.

## 2.4 Operador ternário

Temos que as comparações são elementos essenciais dentro da programação, entretanto elas ocupam muito espaço visual, e algumas comparações são tão pequenas que são quase "banais", pra isso existe o operador ternário, que permite uma comparação de um elemento, que pode receber um de dois valores se sua condição for verdadeira, e essa instrução é feita toda em um única linha.

```
(condição) ? se a condição for verdadeira : se a condição for falsa;
```

# 3 Loops