DIGITALHOUSE

Estruturas de decisão

Unidade 04

Condicional IF

Com as **condicionais** podemos escrever trechos de código que só executarão **se passar** por nossa verificação.

SINTAXE DO IF

Sempre teremos o comando if seguido de parênteses que engloba a condição, depois do fechamento dos parênteses temos abertura e fechamento de chaves que engloba o código que deve ser executado caso a condição seja verdadeira.

{}

```
if ( condição ) {

// código que será executado caso a condição seja verdadeira

}
```

EXEMPLO

Nesse código, a condição verifica se a variável **diaDaSemana** guarda uma variável igual à “quarta-feira”, se estiver correto, executa a próxima linha com o comando console.log e imprime a mensagem “Hoje é dia de feijoada!”.

Caso a variável tenha um valor diferente do esperado, o código que está entre as chaves será ignorado e a execução continua na linha após o fechamento das chaves.

```
{}
    var diaDaSemana = "quarta-feira"

    if (diaDaSemana == "quarta-feira") {
      console.log("Hoje é dia de feijoada!")
    }
```

Operadores Relacionais

DE COMPARAÇÃO SIMPLES

Compara dois valores, retornando verdadeiro ou falso.

```
10 == 15 // Igualdade → false
10 != 15 // Desigualdade → true
```

DE COMPARAÇÃO ESTRITA

Compara o valor e o tipo de dado também.

```
10 === "10" // Igualdade estrita → false
10 !== 15 // Desigualdade estrita → true
```

No primeiro caso o valor é 10 em ambos casos, mas os tipos de dados são number e string. Como estamos pedindo que ambas coisas sejam iguais, o resultado é **false**.

DE COMPARAÇÃO (CONTINUAÇÃO)

Compara dois valores, retornando verdadeiro ou falso.

{}

```
15 > 15 // Maior que → false
15 >= 15 // Maior ou igual que → true
10 < 15 // Menor que → true
10 <= 15 // Menor ou igual que → true
```

Sempre devemos escrever o símbolo maior (>) o menor (<) antes do igual (>= o <=). Se fizermos o contrário (=> o =<) JavaScript lê primeiro o operador de atribuição = e então ele não sabe o que fazer com o maior (>) ou o menor (<).

Os **operadores** de **comparação** sempre **retornarão** um booleano, ou seja, **true** ou **false**, como resultado.

Condicional IF ELSE

O comando **else** vem para complementar nossa estrutura do **if**. Ele define um comportamento caso a afirmação seja falsa.

SINTAXE DO IF ELSE

O comando **else** complementa o **if**. Veja que devemos colocá-lo logo após o fechamento de chaves do if, e ele tem o próprio bloco de código, também englobado com chaves:

{}

```
if ( condição ) {
// código que será executado caso a condição seja verdadeira
} else {
    // código que será executado caso a condição seja falsa
}
```

EXEMPLO

Nesse código nós imprimimos uma mensagem de acordo com a condição, considerando a possibilidade de ser verdadeiro (entra no if), ou falso (entra no else)

{}

```
var diaDaSemana = "terça-feira"

if (diaDaSemana == "quarta-feira") {
  console.log("Hoje é dia de feijoada!")
} else {
  console.log("Hoje não é dia de feijoada :(")
}
```

Condicional ELSE IF

O comando **ELSE IF** vem para complementar nossa estrutura do **if**. Ele define um segundo ou mais comportamentos antes do else

SINTAXE DO ELSE IF

O comando **ELSE IF** tem uma estrutura muito parecida com o **IF.** Ele leva uma condição seguida de um bloco de código. Vale lembrar que, o ELSE IF sempre vem antes do else

{}

```
if ( condição ) {
    // código que será executado caso a condição seja verdadeira
} else if( condição ) {
    // Segunda condição a ser executada
} else {
    // código que será executado caso a condição seja falsa
}
```

Operadores Lógicos

Os **operadores** nos permitem **manipular o valor** das variáveis, realizar **operações** e **comparar** seus valores

DE ATRIBUIÇÃO

Atribui o valor da direita na variável da esquerda.

```
{}  var idade = 35; // Atribuo o número 35 à idade
```

ARITMÉTICOS

Nos permitem fazer operações matemáticas, retornando o resultado da operação.

```
{}  10 + 15 // Soma → 25
    10 - 15 // Subtração → -5
    10 * 15 // Multiplicação → 150
    15 / 10 // Divisão → 1.5
```

ARITMÉTICOS (CONTINUAÇÃO)

```
15++ // Aumento, é igual a 15 + 1 → 16
15-- // Diminuição, é igual a 15 - 1 → 14
```

```
15 % 5 // Módulo, o resto de dividir 15 entre 5 → 0
15 % 2 // Módulo, o resto de dividir 15 entre 2 → 1
```

O operador de módulo (%) nos retorna o resto de uma divisão.

Os **operadores** aritméticos sempre **retornarão** o **resultado numérico** da **operação** que se está realizando.

DE COMPARAÇÃO SIMPLES

Compara dois valores, retornando verdadeiro ou falso.

```
10 == 15 // Igualdade → false
10 != 15 // Desigualdade → true
```

DE COMPARAÇÃO ESTRITA

Compara o valor e o tipo de dado também.

```
10 === "10" // Igualdade estrita → false
10 !== 15 // Desigualdade estrita → true
```

No primeiro caso o valor é 10 em ambos casos, mas os tipos de dados são number e string. Como estamos pedindo que ambas coisas sejam iguais, o resultado é **false**.

DE COMPARAÇÃO (CONTINUAÇÃO)

Compara dois valores, retornando verdadeiro ou falso.

{}

```
15 > 15 // Maior que → false
15 >= 15 // Maior ou igual que → true
10 < 15 // Menor que → true
10 <= 15 // Menor ou igual que → true
```

JavaScript é uma linguagem que **faz diferença entre caracteres MAIÚSCULOS e minúsculas**. Por isso é importante seguir um padrão na hora de escrever os nomes.

Os **operadores** de **comparação** sempre **retornarão** um booleano, ou seja, **true** ou **false**, como resultado.

LÓGICOS

Permitem combinar valores booleanos, e o resultado também retorna um booleano. Existem três operadores **e** (and), **ou** (or), **negação** (not).

AND (&&) **todos** os valores devem avaliar como **true**.

```
{}    (10 > 15) && (10 != 20) // false
```

FALSE → TRUE → FALSE

```
{}    (12 % 4 == 0) && (12 != 24) // true
```

TRUE → TRUE → TRUE

LÓGICOS (CONTINUAÇÃO)

OR (||) **ao menos um** valor deve avaliar como **true**.

```
(10 > 15) || (10 != 20) // true
```

FALSE → TRUE → TRUE

```
(12 % 4 == 0) && (12 != 24) // true
```

TRUE → TRUE → TRUE

NOT (!) nega a condição, se era true, será false e viceversa

```
!false // true
!(10 > 15) // true
```

Os **operadores lógicos** sempre **retornarão** um booleano, ou seja, **true** ou **false**, como resultado.

CONCATENAÇÃO

Utilizamos para unir strings. Devolve outra cadeia de texto.

```
var nome = 'Teodoro';
var sobrenome = 'García';
var nomeCompleto = nome + ' ' + sobrenome;
```

Se misturarmos outros tipos de dados, eles serão convertidos em cadeias de texto.

```
var linha = 'M';
var assento = 7;
var localização = linha + assento; // 'M7' como string
```