Características do JavaScript

JavaScript é uma linguagem dinâmica, ou seja, a linguagem não é compilada, e sim interpretada e executada diretamente. Além disso, é uma linguagem de tipagem fraca, portante o interpretador de código não exige que as variáveis recebam tipos ou que o tipo se mantenha o mesmo ao longo de toda a execução do código.

Tipos em JavaScript

Os principais tipos dispoíveis na linguagem são:

- String: valores de texto
- Number: valores numéricos de ponto flutuante
- Boolean: valores verdadeiros ou falsos (true ou false)

```
// String
const a = "";

// Number
const b = 1;

// Boolean
const c = true;
```

Esses são os tipos de valor, ou seja, a variável a qual o valor é atribuído recebe diretamente o valor atribuído. Isso quer dizer que, caso o valor de uma variável seja atribuído a outra, e a primeira variável seja alterada posteriormente, o valor da segunda variável se mantém inalterado:

```
let var1 = "primeira";

let var2 = var1;

console.log(var1, var2);
// "primeira", "primeira"

var1 = "alterado";

console.log(var1, var2);
```

```
// "alterado", "primeira"
```

O tipo de uma variável pode ser o próprio tipo do valor atribuído a ela, undefined em caso de variáveis declaradas mas não definidas e null em caso de variáveis vazias;

Além disso, o tipo de uma variável determina quais operações podem ser realizadas com ela, como por exemplo, dados do tipo Number, comparar se um Booleano é verdadeiro ou dividir uma String em segmentos menores.

Além dos tipos mencionados acima, existem também:

- Array: listas de valores
- Object: estruturas com diversos pares de chaves e valores

```
// Array
const d = ["", 1, true];

// Object
const e = {
  chave: "valor"
}
```

Esses são os tipos referenciados, nesse caso, a variável onde o valor foi atribuído recebe na verdade uma referência daquele valor na memória. Diferente dos tipos de valor, se uma das variáveis tiver o valor de uma chave modificado, todas as variáveis que possuem aquela referência serão alteradas junto:

```
const obj1 = {
  chave: "valor inicial",
};

const obj2 = obj1;

console.log(obj1, obj2);
// { chave: "valor inicial" } { chave: "valor inicial" }

obj2.chave = "valor final";

console.log(obj1, obj2);
// { chave: "valor final" } { chave: "valor final" }
```

Objetos

Apesar do tipo Object ser um tipo distinto dos outros, tudo é tratado como objeto, portante, possuem propriedades e funções que podem ser acessadas, como o tamanho de Strings e Arrays através da propriedade "length". Os únicos objetos quenão podem ter propriedades são Null e Undefined, por se tratarem de tipos vazios.

Operadores Lógicos

Os operadores lógicos servem para realizar uma comparação entre valores e determinar se a comparação for verdadeira ou falsa:

! Indica a negação do valor ou operação realizada

> Compara se o valor da esquerda é maior que o da direita
>= Compara se o valor da esquerda é maior ou igual ao da direita
< Compara se o valor da esquerda é menor que o da direita
<= Compara se o valor da esquerda é menor ou igual ao da direita
|| Verifica se um dos valores comparados é verdadeiro
&& Verifica se todos os valores comparados são verdadeiros

Operadores de Atribuição, Incrementação e Decrementação

+= Atribuir a uma variável o valor dela somada ao número atribuído
-= Atribuir a uma variável o valor dela subtraíndo do número atribuído
*= Atribuir a uma variável o valor dela multiplicada pelo número atribuído
/= Atribuir a uma variável o valor dela dividida pelo número atribuído
++ Incrementar o valor de um número em 1
-- Reduzir o valor de um número em 1

Observe os exemplos abaixo:

```
let a = 1;
a += 3;

console.log(a);
// 4

let b = 5;
b -= 3;

console.log(b);
```

```
// 2

let c = 3;
c *= 3;

console.log(c);
// 9

let d = 45;
d /= 3;

console.log(d);
// 15

let e = 0;
e++;
console.log(e);
// 1
e--;
console.log(e);
// 0
```

Note que os operadores de incremento e decremento não precisam de uma atribuição, eles automáticamente atualizam o valor da variável.

Loops e Estruturas

Existem algumas estruturas que servem para modificar o curso de execução do código, sendo elas:

if

A estrutura "if" serve para realizar testes lógicos, caso o teste seja verdadeiro, o código dentro do escopo da condicional é executado:

```
const a = 1;
const b = 2;

if (a > b) {
  console.log("O valor " + a + " é maior que o valor " + b);
}
```

Também podemos adicionar uma condicional para caso a condicional seja falsa, utilizando o "else":

```
const a = 1;
const b = 2;

if (a > b) {
  console.log("O valor " + a + " é maior que o valor " + b);
} else {
  console.log("O valor " + a + " é menor que o valor " + b);
}
```

Por fim, podemos encadear diversos "ifs" antes de uma condição "else" final, utilizando o "else if":

```
const a = 1;
const b = 2;

if (a > b) {
  console.log("O valor " + a + " é maior que o valor " + b);
} else if(a == b) {
  console.log("O valor " + a + " é igual ao valor " + b);
} else {
  console.log("O valor " + a + " é menor que o valor " + b);
}
```

Um detalhe importante sobre o if else encadeado é que, caso uma condição seja verdadeira, o código dessa condição é executado, e as outras condições não serão testadas, caso a ordem de escrita das condições não esteja correta, isso pode levar a comportamentos inexperados no código.

switch

O "switch" age semelhante a um if else encadeado, porém, não o utilizados para fazer comparações booleanas, e sim para com a igualdade direta entre uma variável e algumas possibilidades, podendo também utilizar de um valor padrão caso nenhum dos valores comparados seja encontrado:

```
const dia = new Date().getDay();

switch (dia) {
  case 0:
    console.log("Domingo");
    break;
  case 1:
    console.log("Segunda-feira");
```

```
    break;
  case 2:
    console.log("Terça-feira");
    break;
  case 3:
    console.log("Quarta-feira");
    break;
  case 4:
    console.log("Quinta-feira");
    break;
  case 5:
    console.log("Sexta-feira");
    break;
  case 6:
    console.log("Sábado");
    break;
  default:
    console.log("Dia inválido");
    break;
}
```

Um detalhe importante sobre a sintaxe do switch é que precisamos utilizar um "break" ao final de cada "case".

for

O for serve para realizar um número determinado de iterações com base em um contador, a declaração do loop for é dividida em três partes:

1. Contador
2. Comparador
3. Incrementador

Um loop for é escrito da seguinte forma:

```
for (let contador = 0; contador < 10; contador = contador + 1) {
  // código executado em loop
}
```

O loop for é útil especialmente em situações onde precisamos lidar com arrays, podemos utilizar o contador para acessar um novo índice do array a cada iteração.

while

O loop while executa um trecho de código enquanto uma condição for verdadeira, diferente do for, ele recebe somente a condição a ser testada:

```
let contador = 0;
while (contador < 10) {
  // código do loop;
}
```

Loops infinitos

É importante prestar atenção na forma com que escrevemos nossos loops, pois caso algum erro seja cometido nesse processo, podemos criar um loop infinito que irá impedir a execução do resto do nosso código. Loops infinitos podem ocorrer por exemplo, quando não alterados o valor atribuído na condicional do while ou se utilizarmos um operador de decremneto no contador de um loop for em uma situação em que um operador de incremento deveria ser utilizado.

Break e Continue

As palavras reservadas break e continue podem ser utilizadas dentro de loops para interromper a execução do loop, ou pular a execução atual. Vamos supor o seguinte exemplo:

"Escreva um código que printa no console todos os números de 0 a 100, porém, se o valor for par o código pula a execução, e encerra a execução ao atingir um número impar maior que 26."

O código seria assim:

```
for (let i = 0; i ≤ 100; i++) {
  if (i % 2 === 1 && i > 26) {
    break;
  } else if (i % 2 === 0) {
    continue;
  } else {
    console.log(i);
  }
}
```

Note que, mesmo tendo um limite de 100, ao atingir o valor 27 a execução do loop é interrompida.

Desafios

Fáceis

1. Escreva uma função que soma 2 valores
2. Escreva uma função que subtrai 2 valores
3. Escreva uma função que multiplica 2 valores
4. Escreva uma função que divide 2 valores

Médios

1. Escreva uma função que soma os valores de um array
2. Escreva uma função que recebe uma string e uma letra a ser buscada, e retorna o índice da primeira ocorrência da letra desejada

Difícil

1. Escreva uma função que recebe um texto e:

- transforma todos os letras em minúsculas
- retorna quantas vezes cada letra aparece

Dica: uma string é um array de caracteres

Referências

MDN Web Docs