MATERIAL EXTRA

Tipos Primitivos

growdev

TIPAGEM DINÂMICA

Tipagem de variáveis/constantes está ligado a declarar o tipo de dado que será armazenado em um variável. Exemplo para armazenar a idade, declaramos que esta é uma variável que armazena um número, para armazenar o nome, declaramos uma variável que armazena texto.

Tipagem dinâmica é uma característica de determinadas linguagens de programação, que não exigem declarações de tipos de dados, pois são capazes de escolher que tipo utilizar dinamicamente.
Não confunda tipagem dinâmica com ausência de tipos. Na verdade, ela possui tipos, a diferença está na capacidade da linguagem escolher o tipo automaticamente.

Estrutura de dados

Todas as linguagens de programação têm sua própria estrutura de dados embutida, mas essa estrutura frequentemente difere uma da outra.

Tipos de dados no JavaScript

A última versão ECMAScript define sete tipos de dados. Seis deles são chamados de tipos primitivos:

- Boolean
- Null
- Undefined
- Number
- String

Um primitivo (um valor primitivo, tipo de dados primitivo) é um dado que não é estruturado, ou seja, não é representado através de uma estrutura de Objeto e, por consequência, não possui métodos. Na maior parte do tempo, um valor primitivo é representado diretamente através do mais baixo nível da implementação de uma linguagem.

Boolean

Um booleano, em ciência da computação, é um tipo de dado lógico que pode ter apenas um de dois valores possíveis: **verdadeiro** (`true`) ou **falso** (`false`).

```
const maiorDeIdade = true
const ativo = false
```

Null

Em ciência da computação, um valor nulo representa uma referência que aponta, geralmente de maneira intencional, para um objeto ou endereço de memória inválido ou inexistente. O significado do valor nulo varia entre as implementações das linguagens. O valor `null` é um literal em JavaScript que representa um valor nulo ou "vazio" (“sem conteúdo”).

```
// usuario é conhecido e existe, mas não aponta para nenhum tipo ou valor no momento da sua inicialização
let usuario = null
```

Undefined

Uma variável que não foi atribuída a um valor específico, assume o valor undefined (indefinido). O mesmo acontece com **parâmetros** de funções para o qual não existem **argumentos**, assim como a ausência de valor de retorno em uma função.

```
// foo é declarada mas não lhe é atribuída um valor
let foo

console.log(foo) // undefined
```

```
function bar(parametro) {
   console.log(parametro) // undefined
}

let foo = bar()  // argumento inexistente na chamada da função
console.log(foo) // undefined
```

Number

Em outras linguagens de programação diferentes tipos numéricos podem existir, por exemplo: Integers (Inteiros), Floats (Pontos Flutuantes), Doubles (Dobros), ou Bignums. Porém, de acordo com os padrões ECMAScript, existe somente um tipo numérico na linguagem: o **Number**. Este tipo armazena números inteiros ou números de ponto flutuante (com casas decimais).

```
let idadePessoa = 28
let alturaPessoa = 1.72
const PI = 3.14159
```

String

Strings são úteis para guardar dados que podem ser representados em forma de texto. Pode ser um único ou um conjunto de caracteres alfanuméricos.

```
let nomePessoa = "João da Silva";
let cidade = "Porto Alegre";
let categoriaCNH = "B";
```

O comprimento de uma String é o número de caracteres dela. Podemos checar o tamanho de uma String usando a propriedade **length** de uma variável deste tipo.

```
let nomePessoa = "João da Silva";
let cidade = "Porto Alegre";
let categoriaCNH = "B";

console.log(nomePessoa.length); // 13
console.log(cidade.length); // 12
console.log(categoriaCNH.length); // 1
```

Cada caractere na String ocupa uma posição. O primeiro caractere estará no índice 0, o próximo no índice 1 e assim por diante. Para acessar um caractere individual em uma String, podemos utilizar o método **charAt**.

```
let nomePessoa = "Maria";
console.log(nomePessoa.charAt(0)); // "M"
console.log(nomePessoa.charAt(1)); // "a"
console.log(nomePessoa.charAt(2)); // "r"
console.log(nomePessoa.charAt(3)); // "i"
console.log(nomePessoa.charAt(4)); // "a"
```

REFERÊNCIAS

1. Versões ECMAScript
2. Estrutura de dados do JavaScript
3. Sintaxe e Tipos - JavaScript

BONS ESTUDOS