



Design Web

Prof. Romerito Campos

Plano de Aula

Conteúdo

- Tipos/Valores/Variáveis

JavaScript

Tipos, Valores e Variáveis

Visão geral

- O Javascript divide os tipos em duas categorias:
 - tipos primitivos: números, string de texto, boolean
 - tipos objeto
- Os valores especiais `null` e `undefined` são valores primitivos
- Há um tipo chamado `Symbol`
- **Tudo que não for número, string, boolean, symbol, null ou undefined é considerado objeto**
- A linguagem considera arrays como um tipo de objeto

Visão geral

- Javascript aplica coleta de lixo ([Garbage Collector](#)) automático.
- Há também o conceito de imutabilidade aplicado aos valores aos tipos de dados: **String são imutáveis**

```
let nome = "teste";  
//mostra a letra t  
console.log(nome[0]);  
  
//não altera a string original  
//imutável  
nome[0] = 'b';
```

JavaScript

Números

Números

- Quando um número aparece diretamente no código, ele é chamado de **literal** (aplica-se aos outros tipos).

```
//exemplos de literais  
3.14 //float  
2345.123123 //float  
6.1e10 // 6.1 * 10 elevado a 10  
1000  
100000
```

Aritmética em JavaScript

- A linguagem fornece operadores básicos para aritmética: `+`, `-`, `%`, `**` e etc.
- Além disso, há suporte para operações mais complexas com funções e constantes definidas como propriedades do objeto `Math`.
- [Aqui](#) há um resumo das operações disponíveis

Aritmética em JavaScript

- Em linguagens de programação, os tipos números tem os limites inferior e superior que são capazes de representar.
- Algumas operações podem ultrapassar esses limites gerando **underflow, overflow**.
- Além disso, há também o problema da **divisão por zero**
- JavaScript não lança erros para os problemas citados acima.

Aritmética em JavaScript

- O valor `Infinity` é usado para os casos de underflow, overflow nas operações matemáticas.

```
//acessando o valor infinito  
console.log(Infinity)  
console.log(-Infinity)  
console.log(typeof(Infinity))  
console.log(typeof(-Infinity))  
  
//provando um overflow  
//Number é um objeto assim como Math.  
console.log(Number.MAX_VALUE * 2)
```

Aritmética em JavaScript

- Há outro valor importante que é o **Not-a-Number** ou **NaN**.
- Quando realizamos uma divisão por zero, o resultado é um "não-é-número".

```
//vai produzir um NaN  
console.log( "a" / 0 )
```

- Aplique a função **typeof(NaN)** e veja o tipo de dado ao qual **NaN** pertence.

Javascript

Data e Tempo

Data e Tempo

- A classe `Date` representa e manipula os números que representam uma data.
- Datas em JavaScript são objetos.

```
let timestamp = Date.now(); //string
let agora = new Date(); //objeto
let ms = now.getTime(); //milesecs
```

- Há uma vasta quantidade de métodos para manipular as informações de um objeto do tipo data.

Texto

Texto

- As **strings** são utilizadas para representar texto em JavaScript.
- São imutáveis
- Podem ser acessadas por sintaxe de array e a primeira posição é **0**:

```
let nome = "javascript";  
//mostra o caractere 'a'  
console.log(nome[0]);
```

Texto

- há várias formas de escrever **literais strings**

```
" "  
'testing'  
"3.14"  
'nome="js" '  
"usando contrações como wouldn't"
```

```
console.log('duas\nlinhas');
```

```
//string de uma linha
```

```
"one\  
two\  
tree"
```


Texto

- Sequências de escape - **backslash character (\)**

```
//testar no console do browser  
console.log(\tjavascript)  
console.log(\nteste\n)  
console.log(\u03c0)
```

- Os caracteres de escape permitem aplicar teclas como tab, quebra de linhas entre outras. Também é possível representar emojis.

Texto

- JavaScript fornece um API rica para manipulação de String

```
//remove todos os espaços  
console.log("test  ".trim())  
//repete o 'R' 10 vezes  
console.log("R".repete(10))
```

- Lembre-se que string são **imutáveis**
 - Algumas operações aparentemente modificam a string, mas na verdade geram uma nova cópia: `"teste".toUpperCase()`

Boolean

Boolean

- Há apenas dois valores possíveis
- Qualquer valor JavaScript pode ser convertido para boolean
- Converter `undefined`, `null`, `0`, `-0`, `NaN` e `""` para boolean resulta em **false**.
- Qualquer outro valor, incluindo objetos e arrays, são convertidos para true: `a = []; console.log(Boolean(a))`.

Boolean

- Os operadores lógicos e, ou e a negação são definidos como:
 - `&&` - operador **AND**
 - `||` - operador **OR**
 - `!` - operador **NOT**

```
let a=10, b=11;  
  
if ((a===b && b === 20) || !(a+b < 20)) {  
  
}
```

null e undefined

null e undefined

- **null** indica ausência de qualquer valor na linguagem JavaScript.
- **undefined** indica uma ausência ainda mais profunda de valor.
- **Undefined** é uma constante definida pela linguagem, já **null** é uma palavra reservada.
- Funções sem retorno, devolvem **undefined** como valor de retorno
- Comparar `null == undefined` é true, mas `null === undefined` é false.

Variáveis

Variáveis

- Algo fundamental na programação de computadores consiste em usar **identificadores** para representar valores.
- Os identificadores podem ser **variáveis** ou **constantes**
- É necessário declarar o identificador antes de utilizá-los
- Para declarar uma variável podemos fazer: `let a = 10;`.
- Para declarar uma constante podemos fazer: `const b = 10;`.
Constantes não podem ser modificadas.

Referências

FLANAGAN, D. **JavaScript - The Definitive Guide**. 7. ed. Sebastopol, CA, USA: O'Reilly Media, 2020.