



# Introdução à Linguagem JavaScript

Paradigmas de Linguagens de Programação

**Gabriel Marques de Amaral Gravina**

**Ausberto S. Castro Vera**

27 de setembro de 2021



Copyright © 2021 Gabriel Marques de Amaral Gravina e Ausberto S. Castro Vera

UENF - UNIVERSIDADE ESTADUAL DO NORTE FLUMINENSE DARCY RIBEIRO

CCT - CENTRO DE CIÊNCIA E TECNOLOGIA

LCMAT - LABORATÓRIO DE MATEMÁTICAS

CC - CURSO DE CIÊNCIA DA COMPUTAÇÃO

*Primeira edição, Maio 2019*



## Sumário

<b>1</b>	<b>Introdução</b>	<b>5</b>
1.1	Aspectos históricos da linguagem JavaScript	5
1.2	Áreas de Aplicação da Linguagem	5
1.2.1	NodeJS	6
1.2.2	Orientação a objetos	6
1.2.3	Programação Funcional	6
<b>2</b>	<b>Conceitos básicos da Linguagem JavaScript</b>	<b>7</b>
2.1	Estrutura Léxica	7
2.2	Operadores	8
2.3	Variáveis e Constantes	8
2.3.1	Tipos Primitivos	9
2.3.2	Tipos de Objeto	10
2.4	Estrutura de Controle e Funções	10
2.4.1	O comando IF	10
2.4.2	Laços de Repetição	11
2.4.3	For	12
2.4.4	Do ... While	12
<b>3</b>	<b>Programação Orientada a Objetos com JavaScript</b>	<b>15</b>
3.1	Classes e Objetos	15
3.2	Operadores ou Métodos	15
3.3	Herança	15
3.4	Estudo de Caso:	15

<b>Bibliografia</b> .....	<b>17</b>
<b>Index</b> .....	<b>19</b>



## 1. Introdução

A linguagem de programação JavaScript é a “linguagem da web”. Seu uso é dominante na internet e praticamente quase todos os sites a utilizam. Além disso, smartphones, tablets e vários outros dispositivos têm interpretadores de JavaScript embutidos. Isso a torna uma das linguagens mais utilizadas dos dias atuais e uma das linguagens mais usadas por desenvolvedores de software. É importante dizer que, embora o nome sugira, JavaScript é uma linguagem completamente diferente e independente da linguagem Java. Mesmo assim, suas sintaxes tem traços de semelhança, mas nada além disso.

Por ser uma linguagem fácil de ser aprendida e fortemente tolerante, permitiu que usuários pudessem ter suas necessidades atendidas de forma cômoda e eficiente. A linguagem é de alto-nível, dinâmica e interpretada. Além disso, é adequada para orientação de objeto e programação funcional. É uma linguagem não tipada – ou seja, suas variáveis não tem um tipo específico e seus tipos não são importantes para a linguagem. Baseado no livro [Fla20].

### 1.1 Aspectos históricos da linguagem JavaScript

A linguagem foi criada na NETSCAPE por Brendan Eich. Tecnicamente, JavaScript é uma marca registrada da Sun Microsystems (atualmente Oracle) usada para descrever a implementação da língua pela Netscape (atualmente Mozilla). Na época, a Netscape enviou a linguagem para a padronização da ECMA – European Computer Manufacturer’s Association, sua versão padronizada ficou conhecida como “ECMAScript”. Na prática, todos chamam a linguagem apenas de JavaScript. De acordo com [Fla20].

### 1.2 Áreas de Aplicação da Linguagem

A linguagem JavaScript é completamente versátil e tem aplicações nos variados ambientes, seja no client-side ou no server-side. Nesta seção falarei de algumas aplicações e paradigmas da programação que podem ser implementados em JavaScript. De acordo com [Fla20].

### 1.2.1 NodeJS

A linguagem foi criada para ser utilizada em navegadores da web, e esse segue sendo seu ambiente mais comum de execução até hoje. Enfim, o ambiente do navegador permite a linguagem obter a entrada de usuários e fazer requests HTTP. Porém, em 2010 outro ambiente foi criado para executar código em JavaScript. O NodeJS, popularmente conhecido como Node, tinha a ideia de invés de manter a linguagem presa a um navegador, permitir que a linguagem tivesse acesso ao sistema operacional. Isso proporcionou a utilização da linguagem no lado do servidor, invés de se limitar apenas ao navegador. Atualmente, o Node tem grande popularidade na implementação de servidores web. Baseado no livro [Fla20].

### 1.2.2 Orientação a objetos

A linguagem é orientada a objeto, porém apresenta algumas diferenças que valem ser mencionadas. Na linguagem, as classes são baseadas no mecanismo de herança de protótipos. Se dois objetos herdam do mesmo objeto protótipo, então diz-se que são instâncias de uma mesma classe. Membros, ou instâncias da classe, tem suas propriedades para manter e também métodos que definem seu comportamento. Este comportamento é definido pela classe e compartilhado para todas as instancias. Retirado do artigo da documentação da linguagem, em: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Classes>

### 1.2.3 Programação Funcional

Basicamente, a programação funcional é um paradigma da programação que visa produzir software através de funções puras, evitando compartilhamento de estados, dados mutáveis e efeitos colaterais. Embora JavaScript não seja uma linguagem de programação funcional como Haskell ou Lisp, o fato da linguagem poder manipular funções como objeto significa que técnicas de programação funcional podem ser implementadas na linguagem. Os métodos de array do ECMAScript 5, como `map()` e `reduce()` satisfazem bem o estilo de programação funcional. Retirado do livro [Pow15].





## 2. Conceitos básicos da Linguagem JavaScript

Neste capítulo é serão apresentados os principais conceitos da linguagem JavaScript, sua estrutura léxica, operadores, laços de repetição entre outros tópicos. Os livros básicos e recomendados o estudo da Linguagem JavaScript são: [Fla20], [Pow15] entre outros.

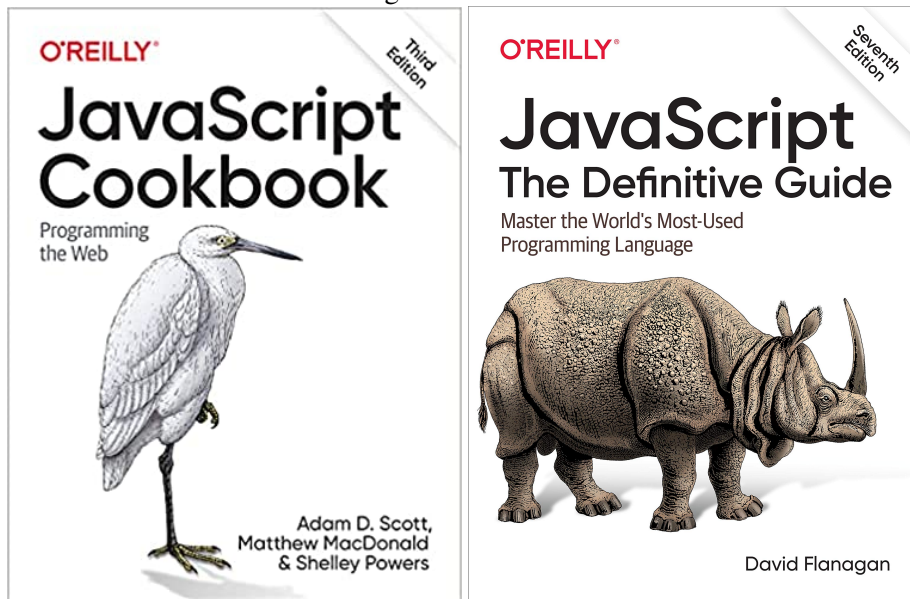
### 2.1 Estrutura Léxica

A linguagem JavaScript é feita utilizando o set de caracteres Unicode, que dá suporte a praticamente todas as linguagens utilizadas atualmente no mundo. Essa é uma linguagem case sensitive, ou seja, os nomes de variáveis, funções e outros identificadores devem ser sempre utilizados de maneira consistente, ao contrário do que acontece no html, por exemplo. Além disso, o JavaScript ignora os espaços e as quebras de linha, com algumas exceções. Isso permite que os programas sejam identados de maneira que façam o código ser legível e fácil de entender. Falando em tornar o código legível, os comentários em JavaScript podem ser feitos de duas formas: uma delas são os comentários de uma só linha, que utilizam `///` e a outra são os comentários de múltiplas linhas, que ignorarão tudo que está dentro dos caracteres. Observe o exemplo abaixo:

```
1  /* Explicacao do codigo  
2    O codigo abaixo realiza... */  
3  var helloWorld = function(){  
4      console.log('Hello World!');  
5  }  
6  helloWorld();
```

\*No JavaScript o uso de vírgulas é opcional.

Figura 2.1: Melhores Livros



Fonte: O autor

## 2.2 Operadores

```

1      ++                //incremento
2      --                //decremento
3      !                 //inverte valores em booleano
4      ==                //testa igualdade
5      !=                //testa desigualdade
6      //testa por igualdade estrita, ou seja, o tipo tambem tem que ser o
mesmo
7      ===
8      ||                //OR
9      &&                //AND
10     =                 //Atribui um valor a uma variavel
11     *=, /=, %=, +=... //Faz uma atribuicao e um calculo
12     //Menor que, maior que, menor ou igual que, maior ou igual que
13     <, >, <=, >=
14

```

## 2.3 Variáveis e Constantes

Com base no livro[Fla20], uma variável é, de forma resumida, um nome simbólico para um valor armazenado no computador. Quando chamamos uma variável, estamos acessando o valor guardado por ela.

Na linguagem JavaScript, existem dois tipos de variáveis: as primitivas e as de objeto. Para se declarar uma no JavaScript é necessário utilizar a palavra reservada "var" seguida de seu nome. Abaixo encontram-se exemplos da declaração de variáveis no JavaScript:

```

1      //E possivel declarar uma variavel vazia
2      var a;
3      var b = 100;

```



```
4     var name = "Lucas";
5
6     //Tambem e possivel declarar multiplas variaveis numa so linha
7     var A = 0, B = 1, C = 2;
8
9     //Variaveis tambem podem ser criadas dentro de lacos de repeticao
10    (for var i = 0; i<10; i++){
11        console.log(i)
12    }
13
```

### 2.3.1 Tipos Primitivos

Os tipos primitivos do JS incluem números, strings de textos e valores booleanos (true e false). Além disso, existem também os tipos especiais "null" e "undefined", que são valores primitivos, porém não são números, strings ou booleanos. Nesse sentido, cada um é considerado membro de um tipo especial.

#### 2.3.1.1 Números

Uma fator da linguagem JavaScript que é incomum em outras línguas é que não há distinção entre inteiros e floats, sendo todos os números representados como floats. A linguagem armazena os números utilizando o formato de floats de 64 bits, podendo armazenar números grandes com precisão considerável.

#### 2.3.1.2 Strings

De acordo com [Fla20], uma string é uma sequência imutável de valores de 16 bits, onde cada um representa geralmente um caractere Unicode. O tamanho da string dependerá de quantos desses valores ela contém. Para incluir uma string num programa, basta colocar aspas (simples ou duplas). Por exemplo:

```
1     //E uma string vazia
2     ''
3
4     "10.24"
5     //Utilizacao da combinacao de aspas simples e duplas
6     '0 numero "8" e par'
7
8     mensagem01a = "Ola, seja bem vindo"
9     //Printa o conteudo da variavel no console do navegador
10    console.log(mensagem01a)
11
12    /*compara o valor da variavel e retorna true ou false.
13    No caso, retornara true*/
14    a = "01a"
15    a == "01a"
16
17
18
```

#### 2.3.1.3 Booleanos

Conforme [Pow15], um valor booleano é um valor que representa verdade ou falsidade. Deste modo, só há dois possíveis valores para um booleano. No JavaScript, as palavras reservadas para os

booleanos são "true" e "false", e são geralmente o resultado de uma comparação. Observe o exemplo:

```
1   a = 10
2   b = 3
3   a == b
4   false
5
6   a == 10
7   true
8
```

#### 2.3.1.4 Tipos Especiais

Consoante a [Fla20], "null" é uma palavra reservada que geralmente indica ausência de um valor. Se utilizarmos o comando "typeof" no "null", veremos que será retornado "object", o que significa que "null" é algo que indica a ausência de objeto. Resumindo, pode ser utilizado para indicar que não há valor em uma variável, string ou objeto.

Por isso, "null" e "undefined" costumam ser definidos como um único objeto do seu tipo.

#### 2.3.2 Tipos de Objeto

Segundo [Fla20], qualquer valor que não seja um número, string, objeto ou null e undefined é um objeto, ou seja, é uma coleção de propriedades onde cada uma tem um nome e valor.

##### 2.3.2.1 Globais

Os objetos globais são aqueles que podem ser usados em todo programa escrito em JavaScript. Quando um interpretador da linguagem inicia, ele cria novos objetos globais e os dá as propriedades que o definem. Algumas das propriedades globais existentes são: undefined, Infinity, NaN. Além de propriedades e funções globais, no JavaScript existem também constructor functions, como: Date(), Object() e objetos globais, como o Math e JSON.

São exemplos de funções globais:

```
1   isNaN()           //Retorna se um valor e um numero ou nao.
2   parseInt()        //Recebe o conteudo de uma string e converte para.
   inteiro
3   parseFloat()       //Recebe uma string e a converte para float.
4   eval()             //Avalia codigo representado por uma string.
5   isFinite()         //Verifica se um numero e finito.
6
7
```

## 2.4 Estrutura de Controle e Funções

De acordo com [Fla20], uma estrutura de controle dita a ordem em que instruções serão executadas. Estruturas muito conhecidas em outras linguagens estão presentes também no JavaScript.

### 2.4.1 O comando IF

O comando IF funciona para fazer com que o JavaScript execute expressões condicionalmente. Isso significa que o computador somente executará uma determinada instrução caso a condição seja verdadeira. Caso a seja falsa, o programa executará outro do bloco de código. O comando IF na

linguagem toma a seguinte forma:

```
1 //Sintaxe
2 if(condicao){
3     //realiza instrucao A
4 }else{
5     //realiza outra instrucao
6 }
7
8 //-----Exemplo-----
9 var nome = "Marcos"
10
11 if(nome == Marcos){
12     console.log("Bem vindo, Marcos!")
13 }else{
14     console.log("Apenas Marcos pode ler esta mensagem!")
15 }
16
```

É possível também utilizar IFs dentro de outros IFs, como no exemplo abaixo:

```
1
2 if(animal == cachorro){
3     console.log("Um cachorro e um animal")
4     if(cachorro == panda){
5         console.log("Um panda e um cachorro")
6     }else{
7         console.log("Um panda nao e um cachorro")
8     }
9 }
10
11
```

### 2.4.2 Laços de Repetição

Laços de repetição executam uma instrução até que uma determinada condição seja verdadeira. Na linguagem JavaScript, os laços de repetição são o 'while', 'do ... while' e o 'for'.

#### 2.4.2.1 While

Os laços de repetição "while" tem a seguinte sintaxe no JavaScript:

```
1 while(condicao == true){
2     execute...
3 }
4
5 //O progrma abaixo conta de 0 a 999
6 vai i = 0;
7
8 while(i < 1000){
9     console.log(i)
10    i++

```

```
11 }  
12  
13
```

É importante que o laço "while" atinga em algum momento uma condição de saída. Caso contrário, o programa continuará executando indefinidamente. No exemplo abaixo, temos um programa sem condição de saída.

```
1 while(Bolsonaro == Horrroso){  
2     console.log("0 presidente e horroroso")  
3 }  
4 //0 programa printara a mensagem acima indefinidamente  
5
```

### 2.4.3 For

Geralmente, os laços que utilizam o "for" são mais simples de serem lidos. Isso devido ao fato de poderem executar uma variável inicial, testá-la e incrementá-la em uma única linha. Na linguagem, o laço for funciona com a seguinte sintaxe:

```
1 for(inicia variavel; testa condicao; incrementa){  
2     realiza instrucao  
3 }  
4  
5 //-----Exemplo-----  
6 //0 programa abaixo calcula fatoriais  
7 var fatorial = 10;  
8 var resultado = fatorial;  
9 var multiplicadorInicial = fatorial - 1  
10  
11 for(var i = multiplicadorInicial; i > 1; i--){  
12     resultado = resultado * i;  
13 }  
14  
15 console.log(resultado)  
16  
17
```

### 2.4.4 Do ... While

Ao contrário do "while" e do "for", o "dowhile" verifica a condição apenas no final da função. A sintaxe do "dowhile" no JavaScript é a listada abaixo:

```
1 do  
2     instrucao  
3 while(condicao == true)  
4  
5 //-----EXEMPLO-----  
6 //0 programa abaixo conta de 1 a 100  
7 var i = 0;
```

```
8   do
9       i++
10      console.log(contador)
11  while(i<100)
12
13
```

Caso o código acima fosse executado com o "while", o programa contaria apenas de 1 a 99, já que a checagem no início impediria o programa de fazer mais uma iteração.







## 3. Programação Orientada a Objetos com JavaS

### 3.1 Classes e Objetos

```
1 class NomeClasse:  
2  
3     def metodo1:  
4  
5     def Metodo2:  
6  
7
```

### 3.2 Operadores ou Métodos

### 3.3 Herança

### 3.4 Estudo de Caso:





## Referências Bibliográficas

- [Fla20] David Flanagan. *JavaScript : the definitive guide : master the world's most-used programming language*. O'Reilly Media, Sebastopol, CA, 2020. Citado 6 vezes nas páginas 5, 6, 7, 8, 9 e 10.
- [Pow15] Shelley Powers. *JavaScript cookbook : [programming the web]*. O'Reilly Media, Sebastopol, CA, 2015. Citado 3 vezes nas páginas 6, 7 e 9.



**Disciplina:** Paradigmas de Linguagens de Programação 2021

**Linguagem:** Linguagem JavaScript

**Aluno:** Gabriel Marques de Amaral Gravina

### Ficha de avaliação:

Aspectos de avaliação (requisitos mínimos)	Pontos
Elementos básicos da linguagem (Máximo: 01 pontos) <ul style="list-style-type: none"><li>• Sintaxe (variáveis, constantes, comandos, operações, etc.)</li><li>• Usos e áreas de Aplicação da Linguagem</li></ul>	
Cada elemento da linguagem (definição) com exemplos (Máximo: 02 pontos) <ul style="list-style-type: none"><li>• Exemplos com fonte diferenciada ( Courier , 10 pts, azul)</li></ul>	
Mínimo 5 exemplos completos - Aplicações (Máximo : 2 pontos) <ul style="list-style-type: none"><li>• Uso de rotinas-funções-procedimentos, E/S formatadas</li><li>• Menu de operações, programas gráficos, matrizes, aplicações</li></ul>	
Ferramentas (compiladores, interpretadores, etc.) (Máximo : 2 pontos) <ul style="list-style-type: none"><li>• Ferramentas utilizadas nos exemplos: pelo menos DUAS</li><li>• Descrição de Ferramentas existentes: máximo 5</li><li>• Mostrar as telas dos exemplos junto ao compilador-interpretador</li><li>• Mostrar as telas dos resultados obtidos nas ferramentas</li><li>• Descrição das ferramentas (autor, versão, homepage, tipo, etc.)</li></ul>	
Organização do trabalho (Máximo: 01 ponto) <ul style="list-style-type: none"><li>• Conteúdo, Historia, Seções, gráficos, exemplos, conclusões, bibliografia</li></ul>	
Uso de Bibliografia (Máximo: 01 ponto) <ul style="list-style-type: none"><li>• Livros: pelo menos 3</li><li>• Artigos científicos: pelo menos 3 (IEEE Xplore, ACM Library)</li><li>• Todas as Referências dentro do texto, tipo [ABC 04]</li><li>• Evite Referências da Internet</li></ul>	
Conceito do Professor (Opcional: 01 ponto)	
Nota Final do trabalho:	

*Observação:* Requisitos mínimos significa a *metade* dos pontos