conquer.plus

JavaScript básico

<title>

AGENDA DO MÓDULO

</title>

Agenda do módulo

Anotações

A linguagem de programação que mais cresce no Brasil é a de JavaScript.

GitHub, 2021.

<title>

MISSÃO DO MÓDULO

</title>

Compreender a importância das **variáveis** no JavaScript e como usá-las na hora de programar.

{01.}
Criando variáveis I

Variáveis:

Estruturas que **armazenam valores ou dados** e que podem ser alteradas posteriormente.

Qualquer tolo pode escrever um código que o computador entenda. Bons programadores escrevem código que os humanos entendem.

Martin Fowler

No que diz respeito à nomenclatura de variáveis, as boas práticas determinam que seja utilizado o formato camelCase.

camelCase: palavras compostas ou frases em que a **primeira letra das palavras** ou abreviatura comece com uma **letra maiúscula,** exceto a primeira letra.

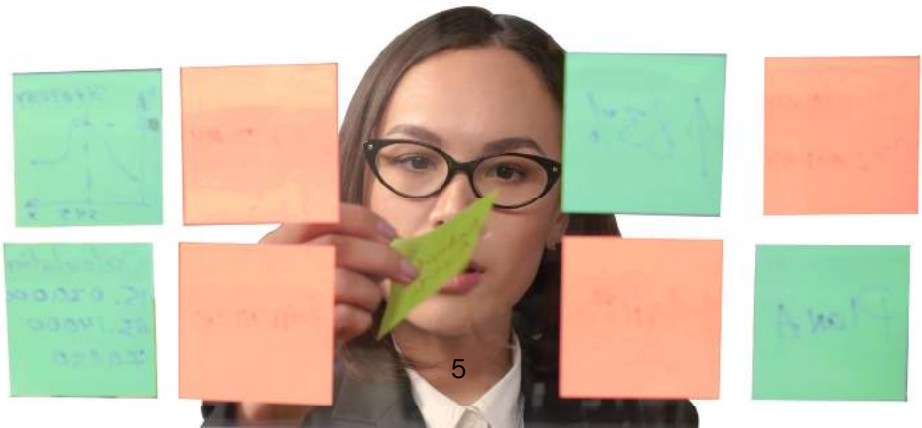
Alguns exemplos são:

- var studentName;
- var clientAge;
- var mathResult.

Outra boa prática é utilizar a nomenclatura de variáveis e a descrição do código, de forma geral, **em inglês.**

Isso possibilitará que o **alcance e a aplicação do seu trabalho seja global.**

Tenha controle sobre suas variáveis.



Existem 3 tipos de escopo em JavaScript:

- Escopo global;
- Escopo de função; e
- Escopo de bloco.

Escopo global

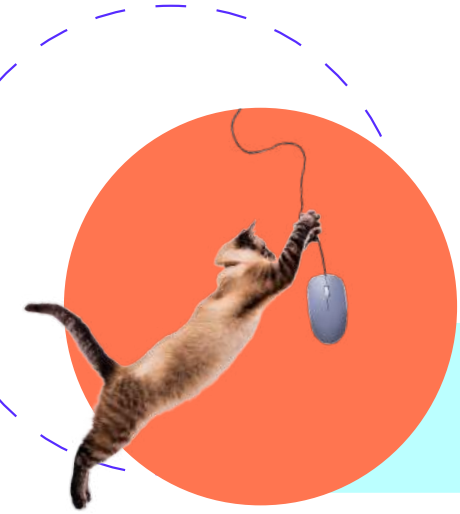
- var
- Variável fora de uma função
- Variável fora das chaves { }
- Usadas em qualquer parte do código

Escopo de função

- Variável dentro de uma função
- Não fica visível para outras partes do código

Escopo de bloco

- Variável declarada dentro das chaves { }, mas não é função
- Não fica visível para outras partes do código
- Não é declarada com var



<title>

PULO DO GATO

</title>

› **Escopo de função** e **escopo de bloco** não são a mesma coisa.

Uma diferença básica é que dentro do escopo de uma função, **podem existir inúmeros escopos de bloco** (como declarações condicionais com if e else)

<title>

MÃO NA MASSA

</title>

› Declare uma variável var nome e atribua seu próprio nome como valor para ela.

› Use o console.log(nome) para ver o resultado.

Você pode usar o console do navegador, como no módulo 1 ou o site **codepen.io/pen.**

{02.}
Criando variáveis II

Com o advento do ES6, foram introduzidas novas keywords para a declaração de variáveis que são **let e const.**

Semelhante ao papel de var, let também é usado para declarar variáveis.
Porém, let pode ter **escopo de bloco.**

Um outro conceito extremamente importante é o **hoisting** ou **"içamento"**.

Trata-se do comportamento de mover declarações para o topo de um escopo.

Const é usado para declarar valores cujo valor é fixo, ou somente de leitura.

Diferentemente de outras linguagens de programação, const em JavaScript não significa que o valor é imutável **(VALOR CONSTANTE)**, mas sim que a referência ao valor é imutável **(REFERÊNCIA CONSTANTE)**.

Por isso você NÃO pode:

- Reatribuir um valor constante
- Reatribuir uma matriz constante
- Reatribuir um objeto constante

Mas caso seu elemento seja uma matriz ou um objeto, você PODE:

- Alterar elementos internos da matriz constante
- Alterar elementos internos do objeto constante

SNAKE_UPPERCASE

Diferentemente das variáveis declaradas com **var e let**, as quais possuem seu nome em **camelCase**, as boas práticas de nomenclatura definem que as variáveis **const** sejam declaradas usando **SNAKE_UPPERCASE**, de forma que todas as letras são maiúsculas separadas por um underline.

<title>

PULO DO GATO

</title>

- Priorize declarar a **variável com const**, a menos que você saiba que o valor será alterado.

Use const ao declarar:

- Uma nova matriz (array)
- Um novo objeto
- Uma nova função
- Um novo RegExp

<title>

MÃO NA MASSA

</title>

- Crie outra variável nome utilizando let e atribua outro valor.
- Dentro do bloco de chaves, imprima “Dentro do bloco o nome é” e concatene a variável nome, faça isso usando console.log.
- Fora do bloco de chaves, imprima “Fora do bloco o nome é” e concatene com a variável nome.

{03.} Tipos de dados: number

Em JavaScript, os valores são tipados, mas não as variáveis.

O que isso quer dizer?

Quer dizer que uma variável declarada com var pode ter um valor numérico atribuído a ela inicialmente, mas no decorrer do código ela pode ter o valor alterado e guardar uma string.

Observação:

Todos os exemplos estão disponíveis na videoaula. Confere lá!

Os valores numéricos ou Number são um tipo primitivo de dados do JavaScript e, diferentemente de outras linguagens de programação, não é necessário especificar **se é um int (inteiro) ou float (flutuante)**. Lembrando que números flutuantes são aqueles com casas decimais.

```
1  const a = 23; //int - inteiro
2
3  const b = 23.5; //float - casas decimais
```

Number()

Uma das alternativas mais utilizadas para realizar a conversão de uma sequência de caracteres numérica em um number, de fato é usar a função Number(). Dessa forma, a string dentro dos parênteses é convertida em um número para ser usado em cálculos.

```
1   const a = "42"; //string
2
3   const b = Number(a);
4   console.log(b) //número
```

NaN (Not a Number)

Além disso, em JavaScript, NaN, Not a Number, é usado para indicar que o valor passado não se trata de um número.

O JavaScript também possui uma lista de built-in methods (métodos nativos) que são ações ou testes que podem ser utilizados em numerais.

Veja alguns dos principais:

<table>

Método	Descrição
isNaN()	Determina se o valor passado é NaN.
isFinite()	Determina se o valor passado é um número finito.
isInteger()	Determina se o valor passado é um inteiro.
isSafeInteger()	Determina se o valor passado é um safe integer.
parseFloat(string)	Converte a string para um número float (é necessário que a string tenha o formato de um número float).
parseInt(string, [radix])	Retorna uma string numérica para um número com notação exponencial.
toExponential(fractionDigits)	Retorna uma string numérica para um número com notação exponencial.
toFixed(digits)	Retorna uma string numérica para um número com notação decimal exponencial.
toPrecision()	Retorna uma string numérica para um número com precisão especificada.
toString([radix])	Retorna uma string numérica em uma base (radix) especificada.
valueof()	Retorna o valor do número.

Além dos métodos, o JavaScript também possui **built-in properties** (propriedades nativas), que são conceitos ou constantes que auxiliam em alguns casos extremos de numerais.

Veja alguns dos principais:

<table>

Propriedade	Descrição
EPSILON	Retorna o menor intervalo entre dois números representáveis.
MAX_SAFE_INTEGER	Retorna o maior safe integer.
MAX_VALUE	Retorna o maior valor possível.
MIN_SAFE_INTEGER	Retorna o menor safe integer.
MIN_VALUE	Retorna o menor valor possível.
NaN	Representa o valor 'Not-a-Number'
NEGATIVE_INFINITY	Representa o infinito negativo.
POSITIVE_INFINITY	Representa o infinito positivo.
prototype	Permite a adição de propriedades a objetos Number.

O JavaScript possui outra biblioteca nativa de grande utilidade que é a **Math**, ela possui alguns recursos poderosos, como o de arredondamento.

<table>

Input	Math.floor	Math.ceil	Math.round	Math.trunc
3.1	3	4	3	3
3.6	3	4	4	3
-1.1	-2	-1	-1	-1
-1.6	-2	-1	-2	-1

<title>

PULO DO GATO

</title>

› Para conhecer a fundo como **usar números** (e datas) em JavaScript, acesse o **MDN Web Docs: https://mzl.la/3wsi5wS.**

<title>

MÃO NA MASSA

</title>

Utilizando a biblioteca **Math,** crie uma variável que gere **números aleatórios** entre 0 e 59 e mostre-os usando **console.log()**.

<title>

PULO DO GATO

</title>

- A biblioteca Math também possui o método **.random**, que é um dos mais úteis para simular aleatoriedade.

Math.random

O método Math.random gera um number float entre 0 e 1 que pode ser trabalhado com os recursos que já conhecemos para que ele adquira o formato que seja necessário.

{04.}
Tipos de dados: string

Agora que já conhecemos um pouco das funcionalidades do numbers em JavaScript, está na hora de falarmos sobre as **strings.**

Por definição, as strings são sequências de caracteres alfanuméricos (letras, números ou símbolos) imutáveis.

Existem três maneiras principais de se converter dados de outros tipos, como number e boolean, em strings:

- Usando o objeto String();
- Usando o objeto new String();
- Usando o método .toString();

Uma outra funcionalidade amplamente utilizada é a concatenação de strings.

Concatenação:
String + String (ou variável).

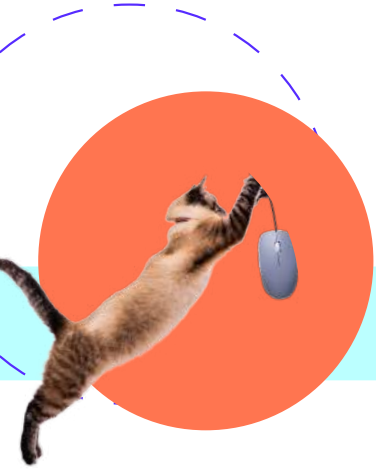
A **concatenação de strings** pode influenciar de modo surpreendentemente intenso o desempenho de seu código.

Nicholas C. Zakas

<title>

PULO DO GATO

</title>



- Sempre que uma variável é concatenada com uma string, ela passa a ser string **apenas durante seu display.**

Outra ferramenta extremamente útil é a interpolação de strings, que permite a manipulação e junção de strings sem a necessidade de utilizar a concatenação, uma vez que para textos mais extensos, usar apenas a concatenação seria mais trabalhoso.

Strings template = Interpolação de Strings $(variável)

Métodos

1. Método .length:
Retorna o tamanho da string, ou seja, quantos caracteres ela possui.

2. Método .charAt():
Nas strings, cada uma das posições de caracteres pode ser acionada por meio do método .charAt(), de forma que o primeiro elemento inicia com o index 0 e segue até o final da string.

3. Método .indexOf():
Este método é o contrário do que foi visto anteriormente, já que usando o .indexOf() deve-se passar o caractere como parâmetro e o seu index (índice) será retornado.

4. Método .split():
Realiza a separação de uma string, tendo um delimitador como guia, e o resultado é uma array em que cada um dos elementos é uma string.

5. Método .trim():
O método .trim() é utilizado para remover os espaços em branco no início ou no fim de uma string. Caso haja espaços entre caracteres, eles não serão removidos.

6. Método .replace():
Este método possui a função de substituir uma substring indicada como primeiro parâmetro, por uma outra string passada como segundo parâmetro.

7. Método .slice():
Ele gera uma substring de acordo com o primeiro e último caractere desejado, os quais devem ser passados como parâmetros do método.

8. Métodos .toUppercase() e .toLowercase():
Os métodos indicados são utilizados para transformar todos os caracteres para maiúsculas (toUppercase) ou para minúsculas (toLowercase).

<title>

MÃO NA MASSA

</title>

Declare uma variável **fullName** que contenha o seu nome completo.

Atribua a outra variável **fullNameLength** o comprimento da variável nome utilizando a propriedade **.length**.

Imprima com **console.log()** o caractere **fullName[fullNameLength]** e depois **fullName[fullNameLength – 1]** e veja o que acontece.

{05.} Tipos de dados: null, undefined e objetos

null

É um valor primitivo do JavaScript.
Geralmente é tratado como uma **ausência intencional de valor**, representando um valor vazio.

O que é diferente de 0 zero, que é um valor e não vazio.

undefined

É um valor global e representa um valor indefinido.
Quer dizer que **quando se declara uma variável, mas não se atribui nenhum valor a ela** (diferente de se atribuir um valor null) ela é do tipo undefined.

No JavaScript podemos utilizar os operadores **==** para checar se duas estruturas são iguais, têm o mesmo valor, e **===** para checar se são idênticas, têm o mesmo valor e são do mesmo tipo.

Objects = Objetos

Os objects (objetos), por sua vez, são estruturas que se diferenciam dos tipos que vimos até agora, pois **podem guardar inúmeros valores primitivos ou até mesmo outros objetos dentro de um.**

Para declarar um object vazio existem duas maneiras:

A primeira é utilizando o construtor **Object()** e a segunda é utilizando a notação literal de objeto com **chaves { }**, que é sempre a mais indicada e mais utilizada.

PULO DO GATO

</title>

› Para acessar os parâmetros dos objetos, use sempre a notação com **colchetes [ ]**.

<title>

MÃO NA MASSA

</title>

Crie um **objeto user** e passe algumas informações pessoais suas, como nome, sobrenome, idade e cidade.

Crie um segundo **objeto computador** e passe as informações da marca e cor do seu computador.

- Adicione esse objeto computador como uma nova propriedade do objeto user.
- Altere o valor de idade para uma idade fictícia.
- Usando **console.log()**, imprima o objeto user no terminal.
- Usando **console.table()**, imprima o objeto user no terminal.
- Imprima apenas o valor de nome no terminal, usando **console.log()**.

{06.}

Coerção explícita e implícita

Quando falamos de coerção estamos falando de uma conversão de um tipo de dado para outro, como de string para number ou vice-versa, por exemplo.

Existem dois tipos de coerção, a implícita e a explícita.

Coerção implícita:

Ocorre quando há conversão de data type de forma automática, por exemplo.

Outra maneira de ocorrer coerção implícita é utilizando o comparador de igualdade **== (loose equality).**

Esse comparador considera apenas o valor.

PULO DO GATO

</title>

› **Evite** ao máximo utilizar a **coerção implícita.**

Coerção explícita:

Possui a mesma essência da coerção implícita, porém nesse caso o programador possui a intenção clara de realizar a conversão de data type.

O modo mais simples de fazer uma conversão do tipo explícita é usar as funções **Boolean()**, **Number()** e **String()**.

David Flanagan

Existem 3 tipos de coerção explícita, que seria para string, para number e para boolean.

<title>

MÃO NA MASSA

</title>

Declare uma variável com o valor "-456.2"v. Usando a coerção explícita, transforme a string em um number.

- Usando **console.log()**, imprima a variável no terminal.
- Usando **typeof()**, imprima o seu tipo de dado (data type) no terminal.

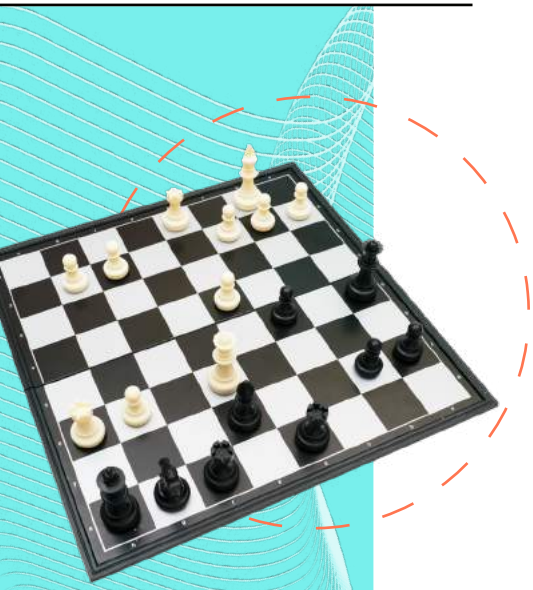
Utilize a coerção implícita concatenando esse number com uma string vazia "".

- Usando **console.log()**, imprima a variável resultante no terminal.
- Imprima novamente o tipo de dado no terminal.

<title>

DESAFIO CONQUER

</title>



```
if (challenge)
   // start
```

<body>

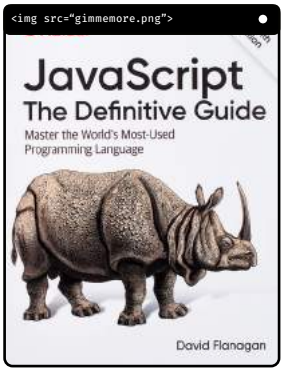
- Adicione seu objUser como mais uma propriedade (key) do objConquer.
- Altere seu campo “nome” em objUser do seu primeiro nome para que contenha o seu nome completo, de forma que o objUser já esteja inserido no objConquer antes da alteração.
- Usando coerção explícita, altere a string “2016” para o formato number, em “fundacao”.
- Imprima seu objeto no console utilizando console.table().

</body>

<title>

QUERO MAIS

</title>



Javascript: The Definitive Guide: *Master the World's Most-Used Programming Language*

E-book: Tutorial sobre JavaScript

Documentação de referência para o JS

Documentação de referência para o ECMA

```
if (interested)
  // gimmemore
```

Anotações