

JavaScript

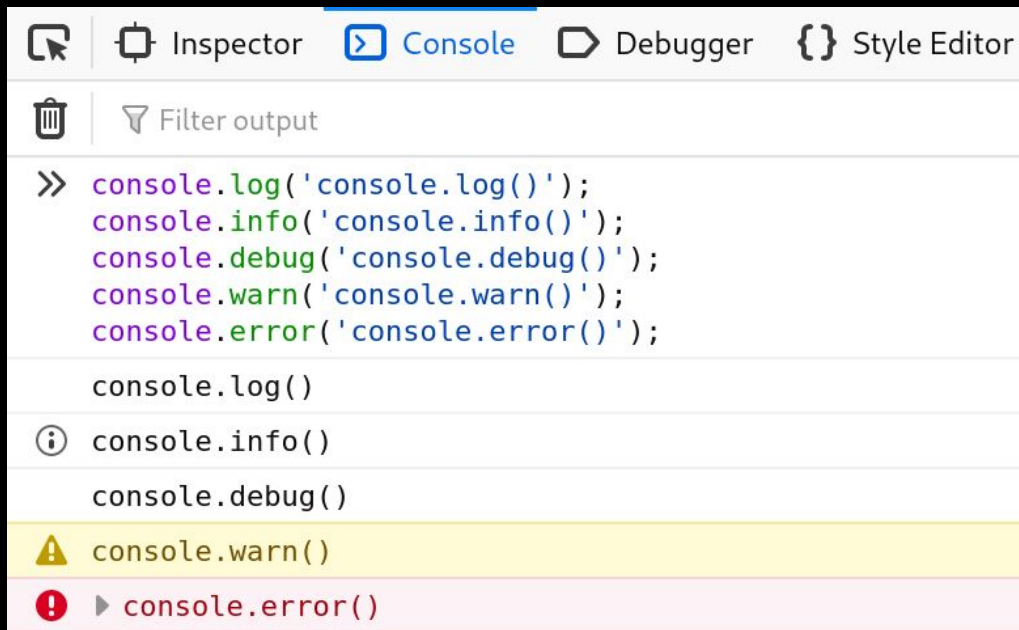
Caio Oliveira
Web Dev

JS

Agenda

1. Iniciando com JS
 - a. Console
 - b. Comentários
 - c. Tipos de dados
 - d. Tratamento de dados
 - e. Concatenação de strings
2. Propriedades e métodos
3. Operadores
4. Objetos pré-definidos
5. Iniciando com funções

O console



The screenshot shows a web browser's developer console with the 'Console' tab selected. The console displays a series of log messages corresponding to the execution of various JavaScript console methods. The messages are color-coded: log (blue), info (green), debug (purple), warn (yellow), and error (red). The 'warn' and 'error' messages are highlighted with yellow and red backgrounds, respectively.

```
>> console.log('console.log()');  
console.info('console.info()');  
console.debug('console.debug()');  
console.warn('console.warn()');  
console.error('console.error()');
```

console.log()
console.info()
console.debug()
console.warn()
console.error()



Dica

É necessário utilizar o ponto vírgula?

Não, porém torna o código mais legível e te acostuma caso você aprenda uma outra linguagem que utiliza.

Comentários



Pra que serve?

Servem para **explicar o que o código** está fazendo, deixar **instruções** para outros desenvolvedores utilizando o código e adicionar outras **anotações úteis**.

**Pra que
serve?**

São muito utilizados no
processo de **debugging**.

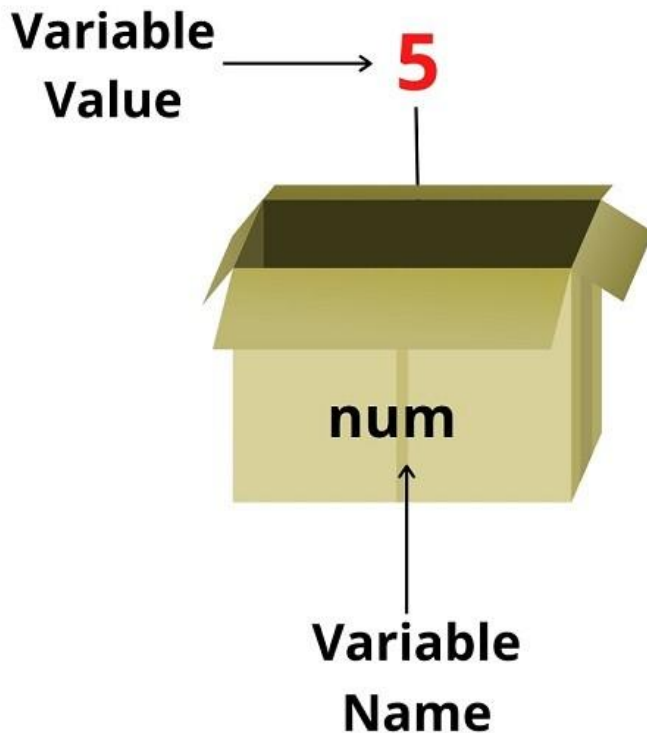
Debug

É o processo realizado para encontrar e corrigir erros.


```
// Comentário em linha única
```

```
/* Comentário  
   em múltiplas  
   linhas */
```

Variáveis



Regras para dar nome às variáveis



Podem começar com letra, \$ ou _



Não podem começar com números



É possível usar acentos e símbolos



Não pode conter espaço



Não pode ser palavra reservada. Ex: *function*, *var*...

**O que não é regra, mas é
convenção?**

Convenções são boas práticas ou guias de estilo da comunidade para a linguagem de programação.



Dica

Vai utilizar nome composto?

Quando a variável possui mais de uma palavra, a convenção é utilizar o chamada **camelCase**, onde a primeira letra é minúscula e as próximas começam com letra maiúscula. Ex: **nomeCompleto**

Fique atento!



A é diferente de a. O JS é **case sensitive**.



Escolha **nomes coerentes** para as variáveis



Evite colocar nomes genéricos como n1, n2, x, y

JavaScript Coding Conventions

Coding conventions are **style guidelines for programming**. They typically cover:

- Naming and declaration rules for variables and functions.
- Rules for the use of white space, indentation, and comments.
- Programming practices and principles.

Coding conventions **secure quality**:

- Improve code readability
- Make code maintenance easier

Coding conventions can be documented rules for teams to follow, or just be your individual coding practice.

https://www.w3schools.com/js/js_conventions.asp

JavaScript

Programming Language

LET

vs

CONST

vs

VAR



Antes de entender a diferença entre elas nós temos que entender, o que é escopo e hoisting...

FLAP

SCHOOL





SÃO PAULO

FIAP

SCHOOL



Scope

```
let year = '2020';
```

Global Scope

```
function theYear() {  
  let text = "The year is"  
  return text + " " + year;  
}
```

Function Scope

```
if(10 < 20) {  
  let greeting = "hi";  
  return greeting  
}
```

Block Scope

FIAP

SCHOOL

Hoisting?!



Hoisting

O Hoisting permite que você **execute funções antes das suas declarações**. Na prática, inicialmente as declarações de funções são colocadas na memória durante a fase de compilação e, mesmo assim, permanecem no mesmo lugar que estão digitadas.

var

- ✓ Respeita o escopo da função
- ✓ Pode ter seu valor alterado e redefinido
- ✓ Pode ser acessada antes de ter um valor atribuído por conta do *hoisting*.

let

- ✓ Respeita o escopo do bloco.
- ✓ Pode ter seu valor alterado, mas não pode ser redefinida. Porém pode ser duplicada se não fizer parte do mesmo escopo.
- ✓ No momento do *hoisting* ela é reconhecida, porém retorna *Reference Error*.

const


- ✓ Também respeita o escopo do bloco.
- ✓ Não pode ter seu valor alterado, nem redefinido.
- ✓ Deve ser inicializada no momento da declaração.
- ✓ Pode ter suas propriedades alteradas.

var

var apple = 



a thing in a box named "apple"

apple = 



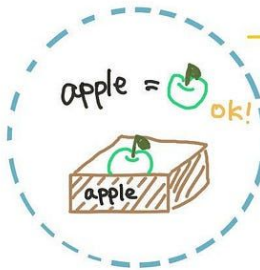
you can swap item later

let

let apple = 



a thing in a box named "apple" w/ protection shield



~~apple =  NG~~

you can swap item only if you ask inside of the shield

const

const apple = 



a thing in LOCKED cage named "apple"



~~apple =  NG~~

you can't swap item later.



apple.multiply(3) OK!

... but you can ask the item to change itself (if the item has method to do that)



var



let



const

**TODAS AS
VARIÁVEIS
SÃO IGUAIS?**



Tipos de Dado Primitivos

String

"hello world"

'abacaxi'

`5 patinhos`

Number

20

3.1415

-18

-9.04

Infinity

NaN

Boolean

true

false

Object

{}

Outros

null

undefined



Dica

Como saber o tipo da variável?

Nós temos um operador que faz isso chamado **typeof**. Ex:

```
typeof "John"  
String
```


JS

typeof function

typeof 'this is a string'

'string'

typeof 12345

'number'

typeof (11 === 11)

'boolean'

Como eu guardo o que o usuário digita em uma variável?



Prompt



Puxando de um input dentro do HTML

Como eu guardo o que o usuário digita em uma variável?

```
let mensagem = prompt("Qual é o  
seu nome?")
```

Como eu guardo o que o usuário digita em uma variável?

```
let mensagem =  
document.querySelector('#txtNome').value
```

FIAP

SCHOOL

Tratamento de dados



Qual é o resultado desse código?

```
<script>
  let numero1 = window.prompt("Digite um número")
  let numero2 = window.prompt("Digite outro número")

  resultado = numero1 + numero2

  alert(resultado)
</script>
```

FIAP

SCHOOL

**Deu ruim!
E como a
gente
resolve?**



```
let numero1 =
```

```
Number.parseInt(window.prompt("Dig  
ite um número"))
```

```
Number.parseFloat(window.prompt("D  
igite outro número"))
```


E se eu quiser converter um número para string?

1ª Alternativa

```
String(window.prompt("Digite um  
número"))
```

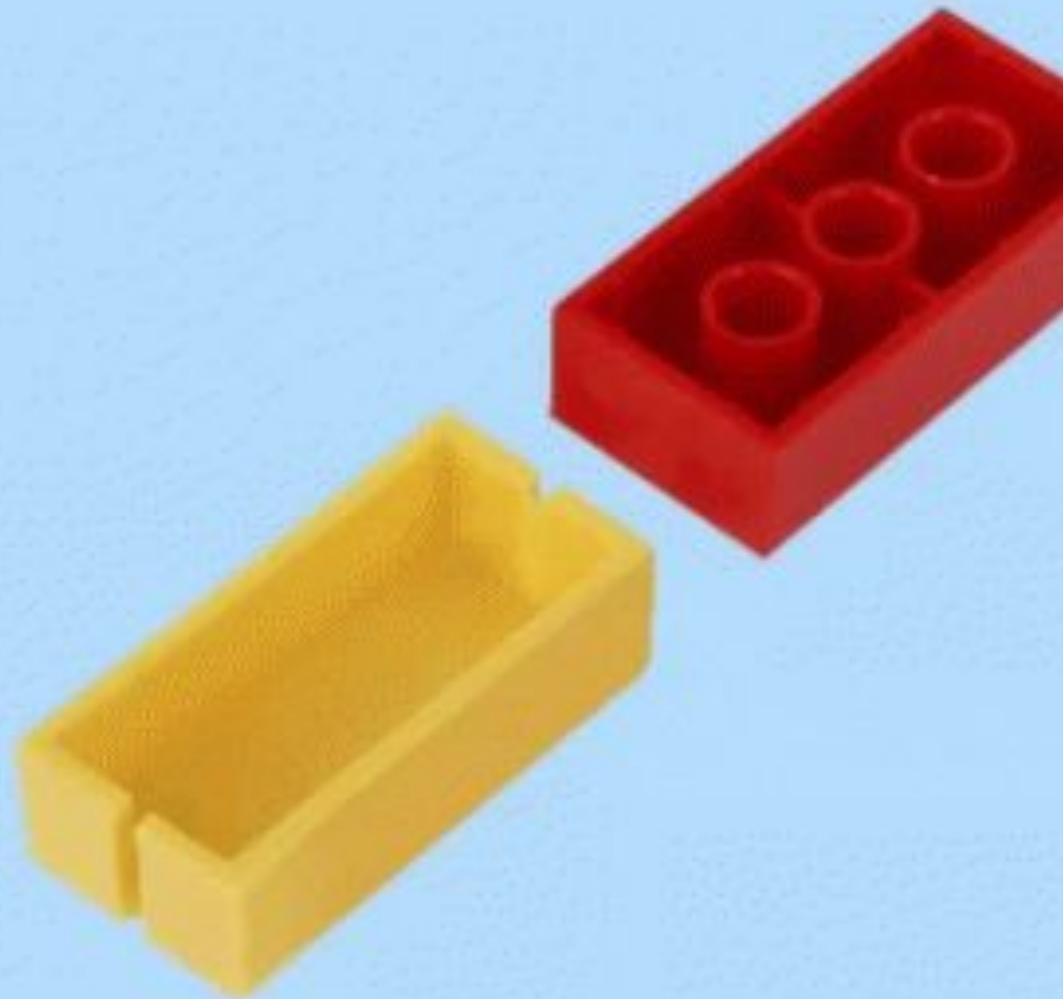
2ª Alternativa

```
numero1.toString()
```

FIAP

SCHOOL

Concatenação





```
let name = 'Francisco';
```

```
let age = 20;
```

```
console.log('The age of ' + name + ' is ' + age);
```

No ES6 ganhamos uma forma mais fácil de fazer isso, a `template string`



`\$ {...}`

```
<script>
  let numero1 = Number.parseInt(window.prompt("Digite um número"))
  let numero2 = Number.parseInt(window.prompt("Digite outro número"))

  resultado = numero1 + numero2

  alert(`O resultado é : ${resultado}`)
</script>
```

Template Strings



Permitem quebra de linha



Permite interpolação



Aceita variáveis,
operadores (soma, divisão,
if/else ternário) e funções

FIAP

SCHOOL

Propriedades e métodos



Propriedades e métodos

Quando nós inserimos um novo pedaço de informação no JS, ele é salvo como uma instância do tipo de dado.

House Class



House Instance



House Instance



Propriedades e métodos

Todos os dados têm acesso à propriedades e métodos específicos que são passados para cada instância.

Nós podemos acessar essas propriedades e métodos usando o **ponto(.)**

Quais são as propriedades das strings?

`nome.length`

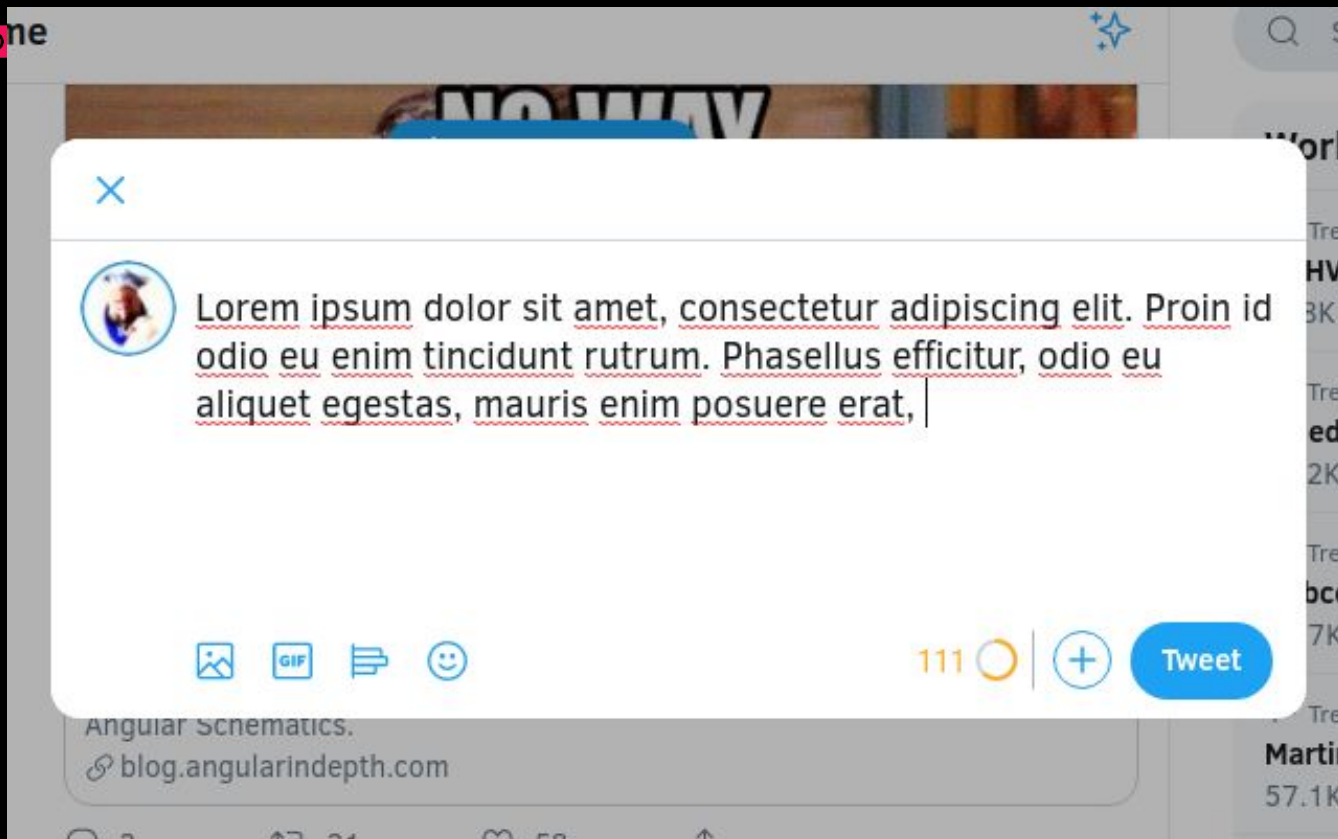
quantos caracteres a string tem

`nome.toUpperCase()`

**transforma todos os caracteres em
maiúscula**

FIAP

SCHO



**E quais são as
propriedades dos
números?**


```
numero1.toFixed(2)
```

Define que o número terá 2 casas decimais

```
nome.toLocaleString('pt-BR',{style: 'currency',  
currency:'BRL'})
```

Formata o dado para moeda em reais

FIAP

SCHOOL

Documentação



Documentação

É qualquer tipo de recurso que te ajuda a aprender sobre as **funcionalidades e características** de uma tecnologia específica.

Desenvolvedores usam a documentação quando eles precisam aprender sobre uma linguagem específica.

FIAP

SCHOOL

No começo ela
pode assustar
um pouco...

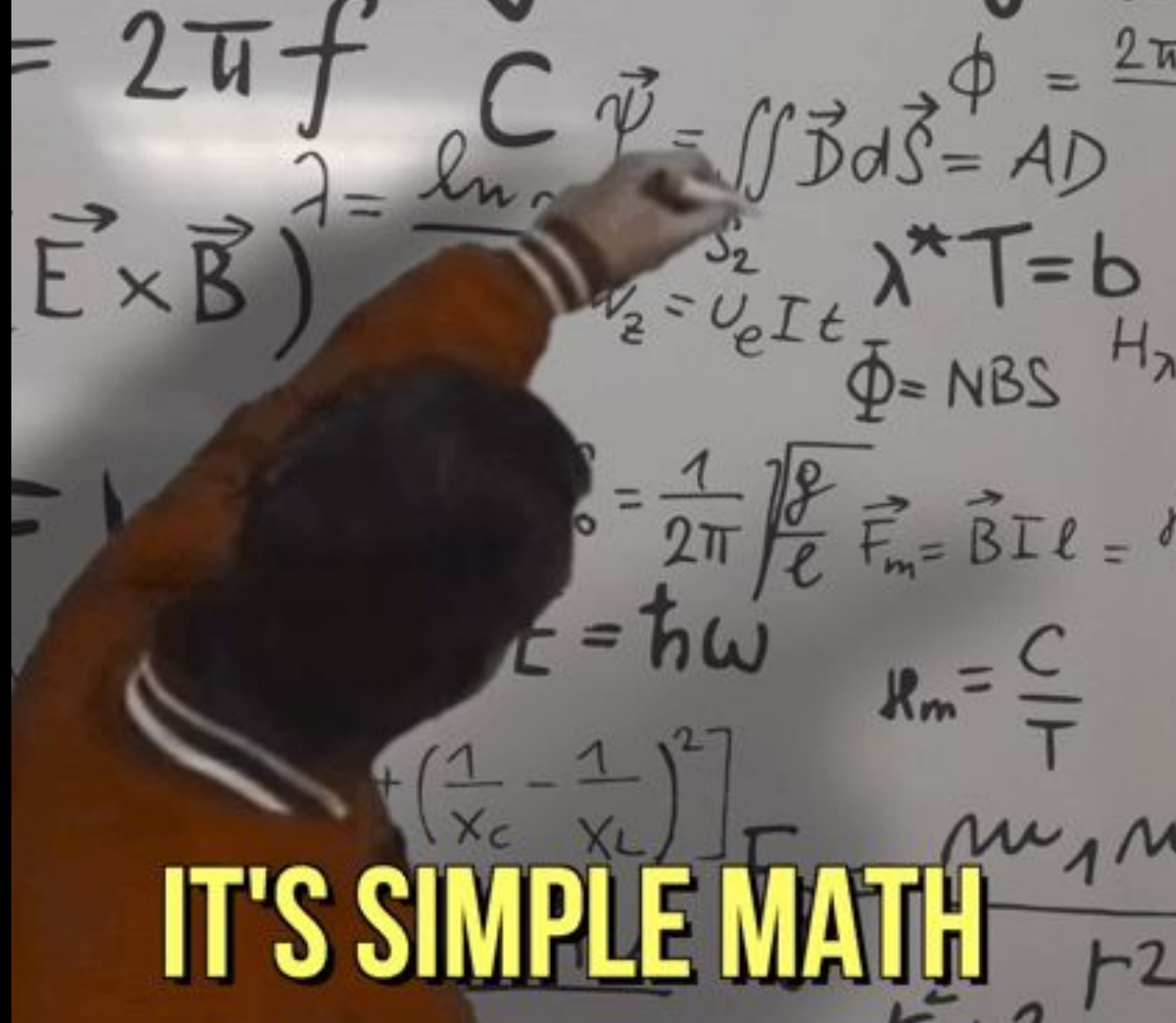


RUNNNNNNN

FIAP

SCHOOL

Operadores



IT'S SIMPLE MATH

Tipos de operadores



Aritméticos



Atribuição



Relacionais



Lógicos



Ternários

Aritméticos

5	+	2	→	7
5	-	2	→	3
5	*	2	→	10
5	/	2	→	2.5
5	%	2	→	1
5	**	2	→	25

FIAP

SCHOOL

$$(5 + 3) * 2$$

Atribuição

Operador	Equivalente a	Utilização
=	$x = y$	$x = y$
+=	$x = x + y$	$x += y$
*=	$x = x * y$	$x *= y$
/=	$x = x / y$	$x /= y$
%=	$x = x \% y$	$x \% = y$

FIAP

SCHOOL



Relacionais

Operador	Descrição
==	Igual a
===	Mesmo valor e tipo
!=	Diferente
!==	Diferente em valor e tipo
<	Menor que
>	Maior que
<=	Menor ou igual a
>=	Maior ou igual a

5 == 5



true

5 == '5'



true

5 === '5'



false

Lógicos

Operador	Descrição
&&	'e' lógico
	'ou' lógico
!	'não' lógico

**Mas nem só de dados
primitivos vive o JS...**

Objetos pré-definidos



Objetos pré-definidos

Além dos dados primitivos, o JS tem objetos.

Alguns deles já vem pré-definidos na linguagem, como o **console**.

Eles têm suas próprias propriedades e métodos que podemos utilizar.

Objetos pré-definidos



Math



Date



Array



String



Number



Boolean

Math

`math.random()` - irá retornar um número decimal aleatório, maior que zero e menor que um.

`math.floor()` - arredonda o número para baixo

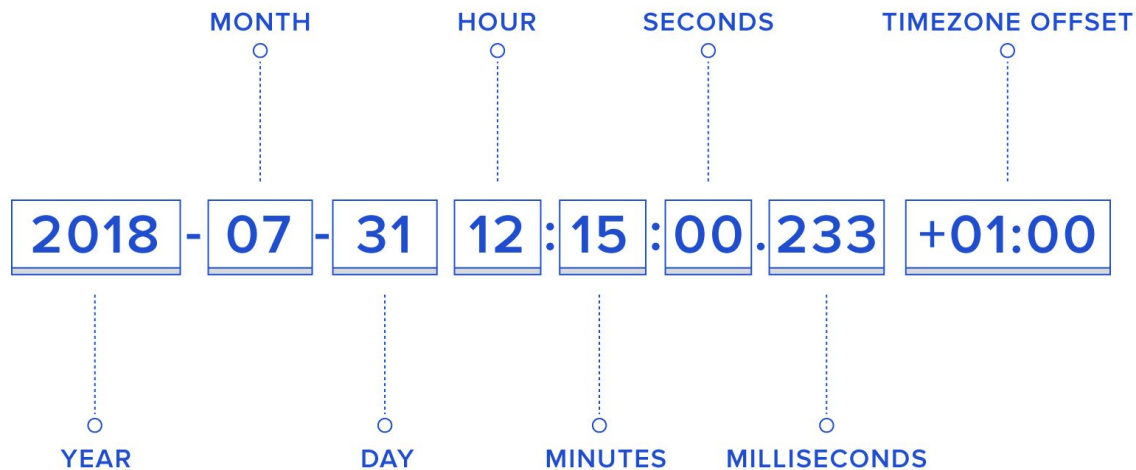
`math.ceil()` - arredonda o número para cima

`math.round()` - arredonda para o valor inteiro mais próximo

Number

`Number.isInteger(17.9)` - retorna true ou false para caso o número seja inteiro ou não

Date



Array

JS

Array Methods



- toString()
- join()
- concat()
- splice()
- slice()
- indexOf()
- lastIndexOf()
- flat()
- forEach()
- map()
- filter()
- reduce()
- some()
- every()
- find()
- findIndex()
- sort()






.map(=>) => 

.filter() => 

.every() => false

.some() => true

.fill(, 1) => 

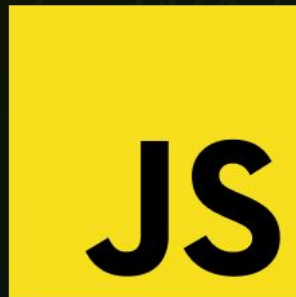
.findIndex(el => el===) => 2

.find() => 

.reduce((acc, cur)=>acc+cur)=> 

O que são
funções?

function () {



? !

}

O que são funções?

É um bloco de código reutilizável que agrupa vários comandos para realizar um objetivo específico.



```
function name() {  
    //.....  
}
```



By: Chinwendu Enyinna



NAME

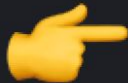


PARAMETERS



```
function addNumbers(a, b) {
```

BODY



```
}
```

Arrow Functions

É uma nova forma de escrever funções introduzida no ES6 de forma mais curta.

Utiliza o símbolo () =>

Elimina a necessidade de escrever a palavra **function**.

ZERO PARAMETERS

```
const functionName = () => {};
```

ONE PARAMETER

```
const functionName = paramOne => {};
```

TWO OR MORE PARAMETERS

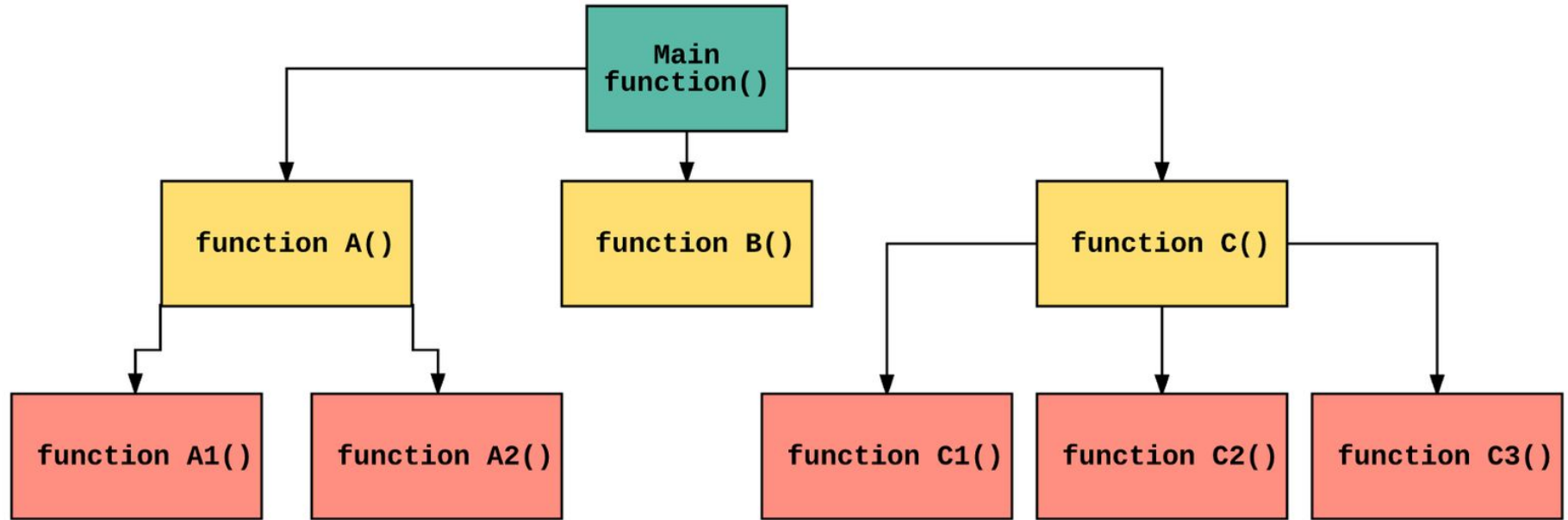
```
const functionName = (paramOne, paramTwo) => {};
```

SINGLE-LINE BLOCK

```
const sumNumbers = number => number + number;
```

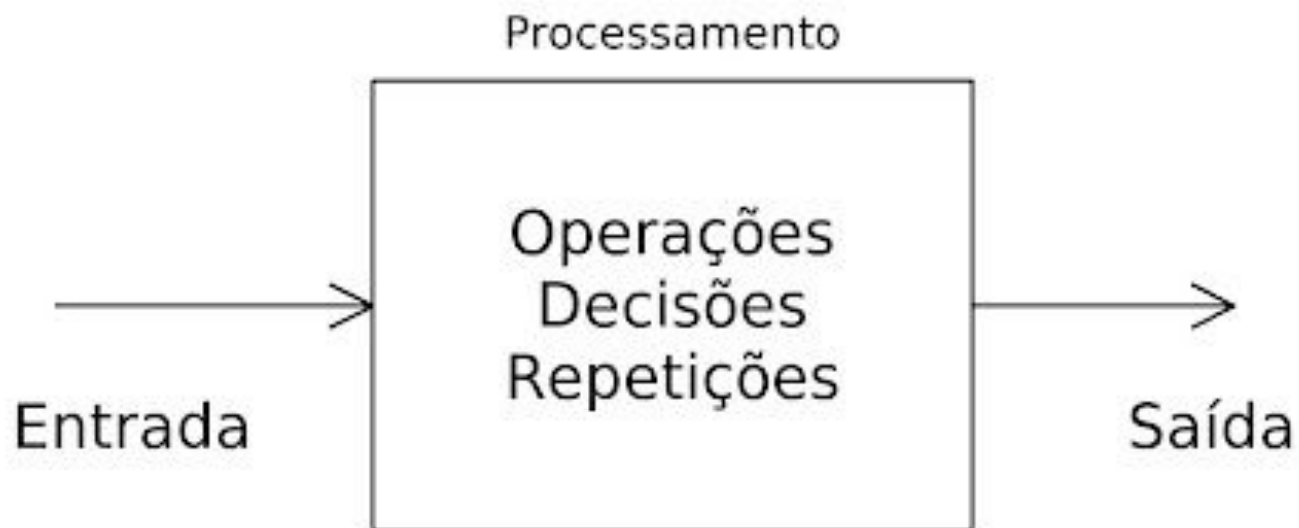
MULTI-LINE BLOCK

```
const sumNumbers = number => {  
  const sum = number + number;  
  return sum; } — RETURN STATEMENT  
};
```



Vantagens de utilizar o escopo local

- Fará seu código mais legível e organizado já que ele estará dividido em seções
- Deixará seu código mais claro já que fica muito mais fácil identificar quais variáveis estão associadas com diferentes partes do programa
- Torna a manutenção do seu código mais fácil, já que ele se torna modular
- Irá economizar memória no seu código porque as variáveis irão “morrer” assim que o bloco for executado



FIA.P

SCHOOL

Hands on!



Desafio 1

Uma livraria está fazendo uma promoção para os livros do Harry Potter: de R\$22 por R\$15, isso se levar os 7 livros (ou mais).

Faça um programa para ajudar o vendedor a informar o preço total com base no número de livros adquiridos.

Desafio 2

Numa avenida movimentada, o limite de velocidade é de 60km/h.

Faça um programa que, com base na velocidade informada, notifique ao guarda o valor da multa (caso haja). A multa é de R\$100 por quilômetro ultrapassado.

Desafio 3

Faça um programa que indique o maior valor entre 3 números.

Desafio Final

Será que conseguimos
fazer sair o Jokeipo? 🙄