

Módulo 01

01 variáveis

```
~ // var, let, const
  // var -> ES5 - cross-browser
  // let e const -> ES2015
```

02 variáveis parte 2

```
let teste = "minha string" // string
teste = 10 // let não permite recriar variável o var permite

const teste2 = 15 // não consegue reatribuir um valor na const, não pode alterar

let teste3 // declarou variável sem valor
teste3 = 20
console.log(teste3)
```

03 tipos de dados primitivos

```
// strings, number (int, float), boolean

let minhaVar = "um texto aqui" // string
let minhaVar2 = 'outro texto'
var minhaVar3 = `template literals`

let idade = 40 // number
let msg = "Eu possuo " + idade + " anos"
let msg2 = `Eu possuo ${idade} anos`
```

04 number

```
01modulo > 03numberjs > ...
1 let idade = 40 // number
2 let msg = "Eu possuo " + idade + " anos"
3 let msg2 = `Eu possuo ${idade} anos`
4
5 console.log(typeof msg) // o typeof mostra o tipo da variável
6
7 const n1 = 10 // number
8 const n2 = 1.50 //number
9
10 console.log(`n1: ${n1} , tipo: ${typeof n1}`)
11 console.log(`n1: ${n2} , tipo: ${typeof n2}`)
```

05 boolean

```
const isValid = true // true ou false

console.log(`tipo: ${typeof isValid}, valor: ${isValid}`)
```

06 undefined e null

```
// undefined, null, symbol ES2015

let varTest // undefined variável declarada mais não utilizada
let varTest2 = null // null valor nulo o tipo é object

console.log(typeof varTest2)
```

07 conversão entre tipos

```
1 let n1 = 10
2 let n2 = "2.5"
3 n2 = parseInt(n2) // converte a string para int 2
4 // se tem 12a ele tira o a em diante e se for a12 ã consegue converter e fica NaN
5
6 n2 = parseFloat(n2) // converte a string para float 2.5
7
8 console.log(n1 * n2) // 20
9 console.log(n1 + n2) // 102 ele concatena com o mais
10
11 // tipos de conversão parseFloat, parseInt, Number()
```

08 conversão entre tipos parte 2

```
01modulo > 07conversaoEntreTipos.js > ...
1  let n1 = 15
2
3  n1 = n1 + "" // converte para string
4  n1 = n1.toString() // converte para string decimal de 0 até 9
5  n1 = n1.toString(16) // converte para hexadecimal 0 ate F
6  n1 = n1.toString(2) // converte para binário 0 e 1
7
```

09 operadores aritméticos

```
1  // +, -, *, /, %, **
2
3  let n1 = 10
4  let n2 = 5
5
6  console.log(n1 + n2) // 15
7  console.log(n1 - n2) // 5
8  console.log(n1 / n2) // 2
9  console.log(n1 * n2) // 50
10 console.log(n1 % n2) // 0 o resto da divisão
11 console.log(3 ** 2) // 9
```

10 operadores de atribuição

```
1  let n3 = 20
2  n3 = n3 + 15
3  n3 += 15
4
5  console.log(n3) // 35
```

11 incremento e decremento

```
01modulo > 11incrementoDecremento.js > ...
1  let i = 0
2  i++ // 1 primeiro soma depois incrementa
3  i-- // 0
4
5  ++i // soma e incrementa
6  --i
```

12 operadores de comparação

```
01modulo > 12operadoresComparacao.js > ...
1  /*
2   | valor igualdade ==
3   | valor igualdade e tipo ===
4   | >, <, >=, <=, !=, !==
5   */
6
7  let n1 = 10
8  let n2 = '10'
9
10 console.log(n1 == n2) // true
11 console.log(n1 === n2) // false
12 console.log(n1 > n2) // false
13 console.log(n1 < n2) // false
14 console.log(n1 >= n2) // true
15 console.log(n1 <= n2) // true
16 console.log(n1 != n2) // false
17 console.log(n1 !== n2) // true
```

13 operadores lógicos

Exp A	Exp B	AND (&&)	OR ()	Exp A	NOT A (!A)
true	true	true	true	true	false
true	false	false	true	false	true
false	true	false	true		
false	false	false	false		

14 operadores lógico parte 2

```
modulo > .js 12operadoresLogicos.js > ...
1 let idade = 18
2 let paisPresente = true
3 let comprouBilhete = false
4 const podeViajar = idade >= 18 || paisPresente && comprouBilhete
5
6 console.log(`Pode viajar: ${podeViajar}`)
```

15 precedência de operadores oque vem entre os (aqui) faz primeiro

```
1 let idade = 18
2 let paisPresente = true
3 let comprouBilhete = false
4 const podeViajar = (idade >= 18 || paisPresente) && comprouBilhete
5
6 console.log(`Pode viajar: ${podeViajar}`)
```

16 condicional if e else

```
01modulo > .js 13ifElse.js > ...
1 let idade = 18
2 let paisPresente = true
3 let comprouBilhete = false
4 const podeViajar = (idade >= 18 || paisPresente) && comprouBilhete
5
6 if (comprouBilhete) {
7   console.log("Sim comprou o bilhete")
8   if (idade >= 18) {
9     console.log("Pode viajar sozinho")
10  } else {
11    console.log("Pode viajar somente acompanhado com os pais")
12  }
13 } else {
14   console.log("Não comprou o bilhete")
15 }
```

17 operador ternário

```
01modulo > .js 14operadoresTernario.js > ...
1 let idade = 18
2 let msg = idade >= 18 ? 'Sim' : 'Não'
3 console.log(msg) // Sim
```


18 valores falsy e truthy

```
01modulo > .js 15falsyTruthy.js > ...
// falsy values: 0, "", NaN, undefined, null
// truthy values: tirando os 5 de cima o resto é tudo truthy

let value = ''

console.log(value ? 'truthy' : 'falsy') // falsy
```

19 curto circuito

```
01 modulo >  16curtoCircuito.js > ...  
1   let n = 0  
2  
3   n = n || 10 // por zero ser falso ele mostra o 10  
4  
5   console.log(n) // 10  
6  
7   let isValid = false  
8  
9   isValid && console.log('é valido') // não executa  
10  isValid || console.log('é valido') // executa é valido
```

20 condicional switch

21 repetção

22 break vc continue

23 funções

24 funções que retornam valores

25 funções que recebem parâmetros

Módulo 03

01 Sistema léxico

Palavras Reservadas

break	debugger	finally	new	typeof
case	default	for	return	var
catch	delete	function	super	void
class	do	if	switch	while
const	else	import	this	with
continue	export	in	throw	yield
	extends	instanceof	try	

Palavras Reservadas para o futuro

enum [↗]	implements	abstract	int	volatile
	interface	boolean	long	
await	let	byte	native	
	package	char	short	
	private	double	synchroniz	
	protected	final	ed	
	public	float	throws	
	static	goto	transient	

```

1 let nome = 'José'
2 console.log(2 + 2) // 4
3
4 // um comentário
5
6 /*
7  Comentário
8  de
9  várias
10  linhas
11 */
12
13 let nome2 = 'Maria'
14 alert(nome2)

```

02 Sistema léxico 2

```

'use strict'
// deixa o use strict na primeira linha
x = 10

function foo() {
  x = 10
}
foo()
//console.log(x)

function dobrar(n1, n1) {
  return n1 * n1
}

console.log(dobrar(5, 7))

```

03 Sistema Léxico 3

```

// o js não é estritamente tipado

let x = 10
x = 'outra string'
console.log('outra string')

function teste() {
  console.log(this) // é o obj windows
}
teste()

```

Módulo 04

01 introdução

```

1 teste()
2
3 function teste() { // é hosting é jogado pra cima na inicialização do programa
4   console.log('teste')
5 }
6
7 const test = function teste2() { // isso não funciona por que nao é hosting
8   console.log('teste2')
9 }
10
11 // o hosting funciona com o var, mais não funciona com o let e const
12 console.log(minhaVar)
13 var minhaVar = 'teste 03'

```

02 funções auto-invocáveis

```
(function() {  
  let isValid = true  
  console.log('Modal', isValid)  
  
  function init() {  
    console.log('init modal')  
  }  
  
  init()  
})();
```

```
(function() {  
  let isValid = true  
  console.log('Menu', isValid)  
  
  function init() {  
    console.log('init menu')  
  }  
  
  init()  
})();
```

```
<body>  
  <script src="./menu.js"></script>  
  <script src="./moda.js"></script>  
</body>
```

03 parâmetro para funções auto-invocáveis

```
(function(n1, n2) {  
  let isValid = true  
  console.log('Menu', isValid, n1, n2)  
  
  function init() {  
    console.log('init menu')  
  }  
  
  init()  
})(10, 20)
```

04 use strict da erro para não poluir o escopo global

```
hulo > 02autoInvocaveis > menu.js > ...  
(function(n1, n2) {  
  'use strict'  
  let isValid = true  
  console.log('Menu', isValid, n1, n2)  
  
  function init() {  
    console.log('init menu')  
  }  
  
  init()  
})(10, 20)
```

05 arguments

```
1 function somar(n1, n2) {
2   return n1 + n2
3 }
4 console.log(somar(10, 20)) // o 10 e 20 é o argumento
5
6 function somar2(arr) {
7   let total = 0
8   for (let i = 0; i < arr.length; i++) {
9     total += arr[i]
10  }
11  return total
12 }
13 console.log(somar2([3, 4, 5])) //12
14
15 function somar3() {
16   let total = 0
17   for (let i = 0; i < arguments.length; i++) {
18     total += arguments[i]
19   }
20   return total
21 }
22 console.log(somar3(3, 4, 5, 9, 50)) // 71
```

06 arguments vs arrow function

```
1 const somar = function() { // o arguments não funciona com () =>
2   let total = 0
3   for (let i = 0; i < arguments.length; i++) {
4     total += arguments[i]
5   }
6   return total
7 }
8 console.log(somar(3, 4, 5, 9, 50)) // 71
```

07 propriedade name

```
function somar2() {
  console.log('soma 2 números')
}

console.log(somar2.name) // somar2
```

08 objetos de primeira classe

```
function fn(cb) {
  console.log('executar ação de callback')
  console.log(typeof cb)
  typeof cb === 'function' && cb()
}

function callback() {
  console.log('função passada por parâmetro')
}

fn(callback)

const objeto = {
  callback
}

objeto.callback()

function fn2(n) {
  return function(n1) {
    return n * n1
  }
}

const funcao2 = fn2(10)
const mult = funcao2(2)
console.log(mult) // 20
```

09 desafio calcular média

```
04modulo > 07calculaMedia.js > <function> > calculaMedia
1 (function() {
2   function calculaMedia() {
3     let total = 0
4     let qtd = arguments.length // quantos parâmetros tem
5
6     for (let i = 0; i < qtd; i++) {
7       if (typeof arguments[i] !== 'number') {
8         throw Error('only numbers')
9       }
10
11       total += arguments[i]
12     }
13
14     return total / qtd
15   }
16
17   let media1 = calculaMedia(6, 8)
18   let media2 = calculaMedia(6, 8, 9)
19   let media3 = calculaMedia(4, 4, 4)
20   console.log(media1)
21   console.log(media2)
22   console.log(media3)
23 })()
```

10 desafio calcular imc

```
04modulo > 08calculaIMC.js > classificaIMC
2 Muito abaixo do peso 16 a 16,9 kg/m2
3 Abaixo do peso 17 a 18,4 kg/m2
4 Peso normal 18,5 a 24,9 kg/m2
5 Acima do peso 25 a 29,9 kg/m2
6 Obesidade Grau I 30 a 34,9 kg/m2
7 Obesidade Grau II 35 a 40 kg/m2
8 Obesidade Grau III maior que 40 kg/m2
9 */
10
11 function calcularIMC(peso, altura) {
12   if (peso === undefined || altura === undefined) {
13     throw Error('Need two parameters: weight and height')
14   }
15
16   return peso / (altura * altura)
17 }
18
19 function classificaIMC(imc) {
20   if (imc > 16 && imc < 16.9) {
21     return 'Muito abaixo do peso'
22   } else if (imc > 17 && imc < 18.4) {
23     return 'Abaixo do peso'
24   } else if (imc > 18.5 && imc < 24.9) {
25     return 'Peso normal'
26   } else if (imc > 25 && imc < 29.9) {
27     return 'Acima do peso'
28   } else if (imc > 30 && imc < 34.9) {
29     return 'Obesidade Grau I'
30   } else if (imc > 35 && imc < 40) {
31     return 'Obesidade Grau II'
32   } else {
33     return 'Obesidade Grau III'
34   }
35 }
36
37 let imc = calcularIMC(59, 1.75)
38 console.log('IMC: ' + imc)
39 console.log(classificaIMC(imc))
```

11 callback

```
04modulo > 09callback.js > teste
1 const teste = function(cb) {
2   console.log('função teste')
3
4   if (typeof cb === 'function') {
5     cb('passado por parâmetro para cb') // executa a callback
6   }
7 }
8
9 const fn = function(param) {
10   console.log('função anonima de callback')
11   console.log(param)
12 }
13
14 //fn(30)
15
16 teste(fn)
```


12 calcular imc com callback

```
1 function calcularIMC(peso, altura, callback) {
2   if (peso === undefined || altura === undefined) {
3     throw Erro('Need two parameters: weight and height')
4   }
5
6   let imc = peso / (altura * altura)
7
8   if (typeof callback === 'function') {
9     return callback(imc)
10  }
11
12  return imc
13 }
14
15 function classificaIMC(imc) {
16   if (imc > 16 && imc < 16.9) {
17     return 'Muito abaixo do peso'
18   } else if (imc > 17 && imc < 18.4) {
19     return 'Abaixo do peso'
20   } else if (imc > 18.5 && imc < 24.9) {
21     return 'Peso normal'
22   } else if (imc > 25 && imc < 29.9) {
23     return 'Acima do peso'
24   } else if (imc > 30 && imc < 34.9) {
25     return 'Obesidade Grau I'
26   } else if (imc > 35 && imc < 40) {
27     return 'Obesidade Grau II'
28   } else {
29     return 'Obesidade Grau III'
30   }
31 }
32
33 let imc = calcularIMC(59, 1.75)
34 let imc2 = calcularIMC(60, 1.65, classificaIMC)
35 console.log(imc)
36 console.log(imc2)
```

Módulo 05

01 oque é DOM

O que é o DOM?

Uma API disponibilizada nos browser
que editar o que é mostrado na tela

document.getElementById()
getElementsByTagName()
getElementsByClassName()
querySelector()
querySelectorAll()

02 selecionar elemento HTML com Javascript

```
<h1 id="title1">Curso Javascript Completo 2018</h1>

<main class="cmain" id="idmain">
  <h2>subtitulo</h2>
  <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. </p>
</main>

<section class="csection" id="idsection">
  <h3>subtitulo</h3>
  <p class="paragrafo2">Lorem ipsum dolor sit amet, consectetur adipisicing elit. </p>
  <p class="paragrafo2">Lorem ipsum dolor sit amet, consectetur adipisicing elit. </p>
</section>

<script>
  // seleciona o elemento do id title1
  document.getElementById('title1').textContent = 'Novo testo' // textContent para adicionar novo texto
  document.querySelector('#title1').textContent = 'Outro novo texto' // o mesmo de cima porém manda o # para id e . para class
  document.getElementsByClassName('paragrafo2')[0].textContent = 'oi pr' // retorna um html[] manda o índice[0]
  document.querySelectorAll('.paragrafo2')[1].textContent = 'mudou por query' // seleciona todas as classe
</script>
```

03 selecionar na árvore do DOM

```
<main class="cmain" id="idmain">
  <h2>subtitulo</h2>
  <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. </p>
</main>

<section class="csection" id="idsection">
  <h3>subtitulo</h3>
  <p class="paragrafo2">Lorem ipsum dolor sit amet, consectetur adipisicing elit. </p>
  <p class="paragrafo2">Lorem ipsum dolor sit amet, consectetur adipisicing elit. </p>
</section>

<script>
  let teste = document.getElementById('idmain')
  // seleciona o elemento pelo nome da tag
  teste.getElementsByTagName('p')[0].textContent = 'editado por arvore'
  document.querySelector('#idmain p') // selecionano pelo query
  document.querySelector('#idsection p')[0] // selecionano pelo query
  document.querySelector('#idsection p')[1] // selecionano pelo query
</script>
```

04 lembrar onde inserir os nosso scripts

```
<section class="csection" id="idsection">
  <h3>subtitulo</h3>
  <p class="paragrafo2">Lorem ipsum dolor sit amet, consectetur adipisicing elit. </p>
  <p class="paragrafo2">Lorem ipsum dolor sit amet, consectetur adipisicing elit. </p>
</section>

<script>
  document.querySelector('h1').textContent = 'Editado com query'
</script>
</body>
</html>
```

05 exercício saudação textContent serve tanto para definir e recuperar propriedades

```
05modulo > 04desafioSaudacao > saudacao.js > ...
1  (function() {
2    const nomeUsuario = 'José Romário'
3    const elemento = document.querySelector('.top-bar p')
4
5    elemento.textContent = `Bem-vindo(a), ${nomeUsuario}`
6  })()

ulo > 04desafioSaudacao > saudacao.js > <function>
(function() {
  const nomeUsuario = 'José Romário'
  const elemento = document.querySelector('.top-bar p')

  elemento.textContent += nomeUsuario
})()
```

06 inserir tags HTML o `textContent` não entende html, o `innerHTML` entende o html

```
(function() {  
  const nomeUsuario = 'José Romário'  
  const elemento = document.querySelector('.top-bar p')  
  
  elemento.innerHTML += `<strong>${nomeUsuario}<strong>`  
})()
```

07 esconder elementos o `parentElement` é o elemento pai, `children` é os filhos

```
(function() {  
  const nomeUsuario = null  
  const elemento = document.querySelector('.top-bar p')  
  
  if (nomeUsuario) {  
    elemento.innerHTML += `<strong>${nomeUsuario}<strong>`  
  } else {  
    // permite escrever o css dentro do javascript  
    // elemento.style.display = 'none'  
    elemento.parentElement.style.display = 'none' // parentElement é o elemento pai  
  }  
})()
```

08 remover elementos

```
elemento.remove() // remove o elemento da página
```

para remover do explorer

```
const elementToRemove = elemento.parentElement // pega o pai  
elementToRemove.removeChild(elementToRemove) // apartir do pai remove o filho  
}
```

09 criar elemento

```
(function() {  
  const nomeUsuario = 'Pedro'  
  
  if (nomeUsuario) {  
    const topBarElemento = document.createElement('div') // cria elemento div  
    topBarElemento.className = 'top-bar' // a div recebe a classe top-bar  
    topBarElemento.innerHTML = `<p>Olá, ${nomeUsuario}</p>` // insira um html dentro do elemento div  
  
    // inserir o elemento div em algum lugar  
    const elementoPai = document.querySelector('.hero') // o elemento pai  
  
    elementoPai.insertBefore(topBarElemento, elementoPai.firstChild) // 1 novo elemento, 2 a referencia  
  }  
})()
```

10 simular o cadastro de e-mail

```
const txtEmail = document.getElementById('txtEmail')  
const msgFeed = document.getElementById('newsletterFeedback')  
  
function cadastrarEmail() {  
  const email = txtEmail.value  
  msgFeed.innerHTML += `0 ${email} cadastrado com sucesso`  
}
```

11 habilitar ou desabilitar um input onblur executa um função quando perde o foco

```
const txtEmail = document.getElementById("txtEmail")

function editarEmail() {
  txtEmail.disabled = false
  txtEmail.focus()
}

function disableEmail() {
  txtEmail.disabled = true
}
```

12 propriedades

```
<form action="#">
  <input type="text" id="txtNome" placeholder="nome" maxLength="30" />
  <br />
  <input type="email" id="txtEmail" placeholder="e-mail" class="input input-email" />
  <br />
  <input type="checkbox" id="contrato"><br />
  <label for="contrato">Li e aceito o Contrato</label>
  <textarea name="" id="" cols="30" rows="10" readOnly>
    Este contrato...
  </textarea>
</form>
```

```
1 const txtNome = document.querySelector("#txtNome")
2 const txtEmail = document.querySelector("#txtEmail")
3 const labelContrato = document.querySelector('label[for="contrato"]')
4
5 txtNome.value = 'Romário' // seta o valor para
6 txtNome.disabled = true // desabilita o input
7
8 txtEmail.readOnly = true
9
10 console.log(txtNome.value) // Romário
11 console.log(labelContrato.htmlFor) // contrato
12 console.log(txtNome.maxLength) // 30
13 console.log(txtEmail.className) // input input-email
```

13 desafio checkbox

```
<script>
  function enableOrDisableButton() {
    const isChecked = document.querySelector("#contrato").checked
    const buttonDisabled = document.querySelector('input[type="submit"]')

    if (isChecked) {
      buttonDisabled.disabled = false
    } else {
      buttonDisabled.disabled = true
    }
  }
</script>
```

Módulo 06

01 métodos de array

Array Literais = const arr = []

Array Construtores = const arr = new Array()

```
06modulo > 01metodosDeArray > indexjs > mapNumeros
1 // array literal
2 const array = [1, 5, 10, "olá", true]
3
4 // retorna true caso satisfaça toda condição
5 let soNumero = array.every(function(elemento) {
6   return typeof elemento === "number" // false
7 })
8
9 // cria um nova array filtrando o dados que você deseja
10 let filtrarNumeros = array.filter(function(elemento, indice, array) {
11   return typeof elemento === "number" // [1, 5, 10]
12 })
13
14 // usa o forEach quando quer fazer um interação dentro do array
15 let foreachNumeros = array.forEach(function(elemento, indice, array) {
16   // console.log('indice: ${indice} = valor: ${elemento}')
17 })
18
19 // retorna uma nova array com os dados alterados
20 let mapNumeros = filtrarNumeros.map(function(elemento, indice, array) {
21   return elemento * elemento // 1, 25, 100
22 })
```

02 indexOf, lastIndexOf, includes, find, findIndex

```
let arr = [4, 5, 10, 20, 35, 43, 5, 4]

// retorna o índice do elemento
let arrIndexOf = arr.indexOf(20) // 3, caso não encontre o valor retorna -1
let arrLastIndexOf = arr.lastIndexOf(5) // 6, caso não encontre o valor retorna -1
let arrIncludes = arr.includes(20) // true, retorna true caso tenha o elemento

// retorna o primeiro valor que satisfaça a minha condição
let arrFind = arr.find(function(elemento) {
  return elemento > 10 // 20, se não encontrar retorna undefined
})

// retorna o índice do primeiro elemento encontrado que satisfaça a condição
let arrFindIndex = arr.findIndex(function(elemento) {
  return elemento > 10 // 3
})
```

03 concat, join, toString

```
06 módulo > 03arrays.js > ...
1 let arr1 = [1, 2, 3]
2 let arr2 = [5, 6, 7]
3
4 let arrToString = arr1.toString() // 1, 2, 3 retorna um string do valor de dentro do array
5 let arrJoin = arr1.join(" - ") // 1 - 2 - 3 retorna um string com o separador dos elementos de sua escolha
6 let arrConcat = arr1.concat(arr2) // [1, 2, 3, 5, 6, 7] junta o array 2 no array 1 e armazena num novo array
7
```

04 push, pop, shift, unshift, slice, splice

```
06 módulo > 04arrays.js > ...
1 let arr = [1, 3, 4, 7, 9];
2
3 arr.push(12); // acrescenta valor no final do array, e retorna o índice e modifica a arr original
4 arr.pop(); // remove o último elemento e retorna esse elemento e modifica a arr original
5 arr.shift(); // remove o primeiro elemento e retorna esse elemento e modifica a arr original
6 arr.unshift(3, 12, 55); // adiciona elemento no início do array e retorna o índice e modifica a arr original
7
8 let arr2 = [0, 1, 2, 3, 4, 5, 6, 7, 8];
9
10 let newArr2 = arr2.slice(2, 5); // recorta um pedaço do array 1pr o início, 2pr até não altera a array original
11 let newArr4 = arr2.splice(1, 3, "Olá mundo"); // modifica a array original, 1pr o início, 2pr remove 3 elementos, 3pr adiciona novo elemento, e retorna os valores removidos
12
13 console.log(arr2);
14 console.log(newArr2);
```

05 reverse, reduce

```
06 módulo > 05arrays.js > nomeOrder > nomeReduce() callback
1 let arr = [1, 4, 5, 8];
2
3 console.log(arr.reverse()); // inverte a ordem do array e altera a array original
4
5 // o reduce faz uma interação em cada elemento do array e retorna um único valor
6 let soma = arr.reduce((acumulador, valorAtual, indice, arrayOriginal) => {
7   return acumulador + valorAtual;
8 })
9
10 let nomes = ["Maria", "Pedro", "Luiz", "Souza", "Marcos"];
11
12 let nomeOrder = nomes.reduce((acc, atual) => {
13   let primeiraLetra = atual[0];
14
15   if (acc[primeiraLetra]) {
16     acc[primeiraLetra]++;
17   } else {
18     acc[primeiraLetra] = 1;
19   }
20
21   return acc; // { M: 2, P: 1, L: 1, S: 1 }
22 }, {}) // valor de retorno
23
24 console.log(nomeOrder);
```

06 desafio

```
06desafio.js > ...
1  const numeros = [1, 3, 4, 1, 4, 5, 3, 5, 8, 9]
2  // retorno [1, 3, 4, 5, 8, 9]
3
4  let numerosUnicos = numeros.reduce((acc, atual) => {
5
6    if (acc.indexOf(atual) < 0) {
7      acc.push(atual)
8    }
9
10   return acc
11 }, [])
12
13 console.log("números únicos = ", numerosUnicos); // [1, 3, 4, 5, 8, 9]
```

07 array.from vs array.of

08 spread operator

09 destructuring

10 for of

11 desafio

12 prototype, call e apply