

Revisão de Javascript

Labenu_



O que vamos ver hoje?

- Variáveis
- Operadores Aritméticos, Lógicos e Comparadores
- Strings e Arrays
- Funções
- Objetos
- Condicionais
- Laços
- map e filter



we make our site more



Variáveis

Labenu_




Variáveis

- Variáveis são estruturas que permitem **guardar** e **acessar** quaisquer informações no nosso código
- Antes de usarmos estas variáveis, nós precisamos **declará-las** (criá-las)

```
const nome = 23 // const NÃO pode mudar o valor  
let idade = 23 // let pode mudar o valor!  
idade = 24
```



Variáveis

- Uma variável pode receber qualquer tipo de valor (booleano, números, strings, arrays, objetos...)
- Elas devem ter nomes significativos e seguir o padrão camelCase 
 - **Ruim:** Nome
 - **Bom:** nomeDoUsuario



Conversões de Tipos

- Podemos converter strings para números e vice-versa
- Geralmente usamos essas funções de conversão junto com o prompt, que sempre guarda o que o usuário escreveu em forma de **string**!

Número ⇒ String: `toString()`

String ⇒ Número: `Number()`





Exercício 1

- Crie uma variável para guardar o **nome** de um produto e peça para o usuário inserir esse nome
- Crie uma variável para guardar o **preço** de um produto e peça para o usuário inserir esse valor
- O GERENTE FICOU DOIDO! Dê um desconto de 1 real no preço do produto, guardando o novo preço **na mesma variável**

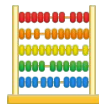


Operadores Aritméticos

Labenu_



Operadores Aritméticos



- `const soma = 20 + 20` `// 40`
- `const subtracao = 25 - 20` `// 5`
- `const multiplicacao = 2 * 20` `// 40`
- `const divisao = 20 / 5` `// 4`
- `const resto = 20 % 3` `// 2`



Operadores Aritméticos



- Uma possível simplificação

```
let resultado = 100
```

```
resultado = resultado + 20    ➡    resultado += 20
```

```
resultado = resultado - 10    ➡    resultado -= 10
```

```
resultado = resultado * 5     ➡    resultado *= 5
```

```
resultado = resultado / 10    ➡    resultado /= 10
```





Exercício 2

- Peça para o usuário **inserir** dois números e guarde-os em variáveis diferentes (**num1** e **num2**)
- Imprima no console:
 - A **soma** do primeiro com o segundo
 - A **subtração** do primeiro pelo segundo
 - A **multiplicação** do primeiro pelo segundo
 - A **divisão** do primeiro pelo segundo
 - O **resto da divisão** do primeiro pelo segundo



Comparadores

Labenu_



Comparadores 🧤🧤

- São operadores que permitem **comparar** variáveis entre si e retornam **true** ou **false**
- São eles:
 - Igual (valor e tipo): **===**
 - Diferente: **!==**
 - Maior e maior igual: **>** e **>=**
 - Menor e menor igual: **<** e **<=**





Exercício 3

- Sorteie um número aleatório entre 1 e 10
- Peça para o usuário inserir um número de 1 a 10
- Imprima no console os dois números dizendo se o que foi digitado pelo usuário é **menor**, **maior** ou **igual** ao sorteado



Operadores Lógicos

Labenu_



Operadores Lógicos

- São operadores especiais usados entre booleanos
- Retornam um valor booleano
- Existem 3 importantes:
 - Operador E: **&&**
 - Operador Ou: **||**
 - Operador Não/Negação: **!**



Operadores Lógicos



a	b	a && b
V	V	V
V	F	F
F	V	F
F	F	F

a	b	a b
V	V	V
V	F	V
F	V	V
F	F	F

a	!a
V	F
F	V





Exercício 4

Um parque de diversões te contratou para fazer um sistema que permite ou proíbe a entrada de pessoas em um brinquedo

MUITO RADICAL

Para entrar, é necessário:

- Ter mais de 18 anos
- Ter mais de 1,60m de altura
- Não ter nenhum problema cardíaco

Faça essas perguntas ao usuário e diga o resultado no console



Strings

Labenu_



Strings

- Podemos declarar strings de 3 maneiras:
 - `"Olá Mundo"` , `'Olá Mundo'` ou ``Olá Mundo``
- Podemos colocar variáveis no meio de 2 jeitos:
 - `"Meu nome é " + nome + " e tenho " + idade + " anos"`
 - ``Meu nome é ${nome} e tenho ${idade} anos``



Protótipo de Strings

- Propriedades
 - **length** ⇒ número de caracteres, conta espaço
- Métodos
 - **toLowerCase()** e **toUpperCase()** ⇒ minúsculo/maiúsculo
 - **trim()** ⇒ retira espaço do início e do fim
 - **includes()** ⇒ true/false se existem caracteres
 - **replaceAll()** ⇒ substitui caracteres por outros





Exercício 5

Dada a frase " **Hoje vou comer cenoura, ebaaa** " realize as operações:

- Imprima a frase inicial juntamente com seu tamanho
- Retire os espaços do início e do fim
- Faça com que possua apenas letras minúsculas
- Verifique se as palavras **comer** e **batata** estão presentes
- Substitua cenoura por batata
- Verifique novamente se possui as palavras **comer** e **batata**



Arrays

Labenu_



O que são arrays?

- Arrays nada mais são do que **listas de elementos**
 - **Ex:** lista de compras, lista de alunos, lista de números da loteria, lista telefônica...
- No javascript, usamos colchetes para agrupar os itens de uma lista:

```
const listaDeCompras = ["batata", "alface", "queijo"]
```

```
const listaDeNumerosMega = [2, 13, 26, 35, 41, 60]
```



Acessando um elemento

- Acessamos os itens através da posição deles na lista!
- Funciona como se fosse uma lista numerada, começando do 0:

Lista de Compras

0. Batata
1. Alface
2. Queijo



Qual é o **item na posição 1**?

Resposta: Alface



Protótipo de Arrays

- Propriedades
 - **length** ⇒ quantidade de itens da lista
- Métodos
 - **push()** ⇒ adiciona item ao fim da lista
 - **pop()** ⇒ retira último item da lista
 - **includes()** ⇒ true/false se existe o item
 - **splice()** ⇒ remover elementos sabendo o índice





Exercício 6

Dada a lista **["batata", "cenoura", "beterraba"]** realize as operações:

- Imprima o segundo item da lista
- Imprima o tamanho da lista
- Adicione o item "mandioca"
- Verifique se há um item chamado cenoura
- Remova o item de índice 1



Funções

Labenu_



O que é uma função?

- Uma função é um **bloco de código** que pode ser **chamado (ou invocado)** a partir de um **nome**

```
1 function calculaArea(altura, largura) {  
2   const area = altura * largura  
3   console.log(area)  
4 }  
5  
6 calculaArea(2, 3)
```

bloco de código

chamada
(invocação)



Parâmetros e Argumentos

- Funções podem receber **entradas**, que podem ser usadas no meio do código

```
1 function calculaArea(altura, largura) {  
2   const area = altura * largura  
3   console.log(area)  
4 }  
5  
6 calculaArea(2, 3)
```

parâmetros

argumentos



Retorno



- Funções podem gerar **saídas**, que podem ser acessadas após a execução

```
1 function calculaArea(altura, largura) {  
2     const area = altura * largura  
3     return area  
4 }  
5  
6 // Atribui retorno à uma variável  
7 const areaCalculada = calculaArea(2, 3)  
8  
9 // Imprime retorno no console  
10 console.log(calculaArea(2, 3))
```

retorno da função

chamadas



Comparação 🙌

Declaração de função

```
1 function somaNumeros (num1, num2) {  
2     return num1 + num2  
3 }
```

Expressões de função

```
1 let somaNumeros = function(num1, num2) {  
2     return num1 + num2  
3 }
```

```
1 let somaNumeros = (num1, num2) => {  
2     return num1 + num2  
3 }
```





Exercício 7

- Crie uma **função** que:
 - **Receba** um array de números e
 - **Retorne** um **novo array** com o último e o primeiro número do array recebido divididos por dois



Objetos

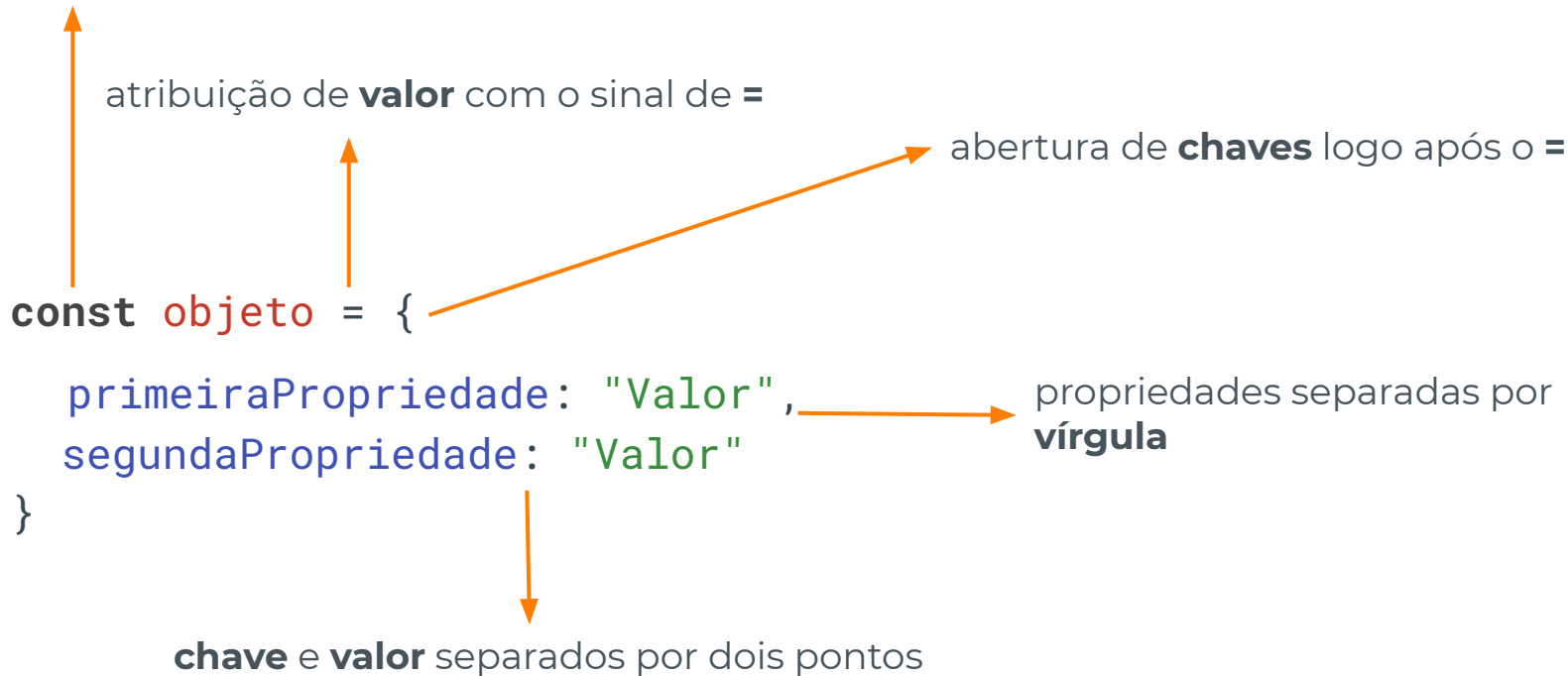
Labenu_



Estrutura padrão de um objeto



declaração com `let` ou `const`
seguido do **nome** do objeto



Acessando e alterando propriedades

- Notação do Ponto

```
const instrutora = {  
  nome: "Amanda Rangel",  
  idade: 27,  
  email: 'mandinha_rock@gmail.com'  
}
```

-  objeto
-  chave
-  valor

```
const nomeDaInstrutora = instrutora.nome;  
console.log(nomeDaInstrutora)
```



Espalhamento ou spread

- Podemos copiar um objeto utilizando o spread operator

```
const usuario = {  
  nome: 'Astrodev',  
  idade: 25,  
  email: 'astrodev@labenu.com.br'  
}
```

```
const novoUsuario = {  
  ...usuario,  
  nome: 'Caio',  
  sobrenome: 'Teixeira'  
}
```





Exercício 8

- Crie um objeto que represente uma pessoa. Essa pessoa precisa ter **nome**, **idade** e **gênero musical preferido**
- Imprima no console as propriedades desse objeto, seguindo o modelo abaixo:

"O nome da pessoa é ____, ela tem ____ anos e gosta muito de ____."

- Crie uma nova pessoa, com mesma idade e gênero musical, mas nome diferente



Conditionais

Labenu_



Bloco if /else 🤔

- **if/else** é a **sintaxe** de programação utilizada para **condicionais**
- Se a **condição for verdadeira** o código **dentro do if** é executado
- Se a **condição for falsa** o código dentro do else é executado



Bloco if / else 🤔

- **if + else:** Todo código da ação vai entre chaves { }

```
1  let condicao = false
2
3  if (condicao){
4      console.log('Entrei no if!')
5  } else {
6      // Como o valor da condição é false,
7      // o código do bloco else será executado
8      console.log('Entrei no else!')
9  }
```



Switch case 🤔

```
1  let paisDeOrigem
2  switch (paisDeOrigem){
3    case 'Brasil':
4      console.log('brasileiro')
5      break
6    case 'EUA':
7      console.log('norte americano')
8      break
9    case 'Inglaterra':
10     console.log('inglês')
11     break
12   default:
13     console.log('nacionalidade não encontrada')
14     break
15 }
```

Verifica o valor de uma **variável**

Os **cases** indicam as condições

Se a variável for igual ao que está no case, o código de dentro será executado





Exercício 9

- Receba um número do usuário
 - Se for par, imprima: é par
 - Senão, imprima: é impar





Exercício 10

- Receba do usuário o nome de um bichinho:
 - Se for um cachorro, retorne: Au au
 - Se for um gato, retorne: Miau
 - Se for uma vaca, retorne: Muuu
 - Se for outro bichinho, retorne: sem barulho reconhecido :(
- Faça utilizando if/else e switch case



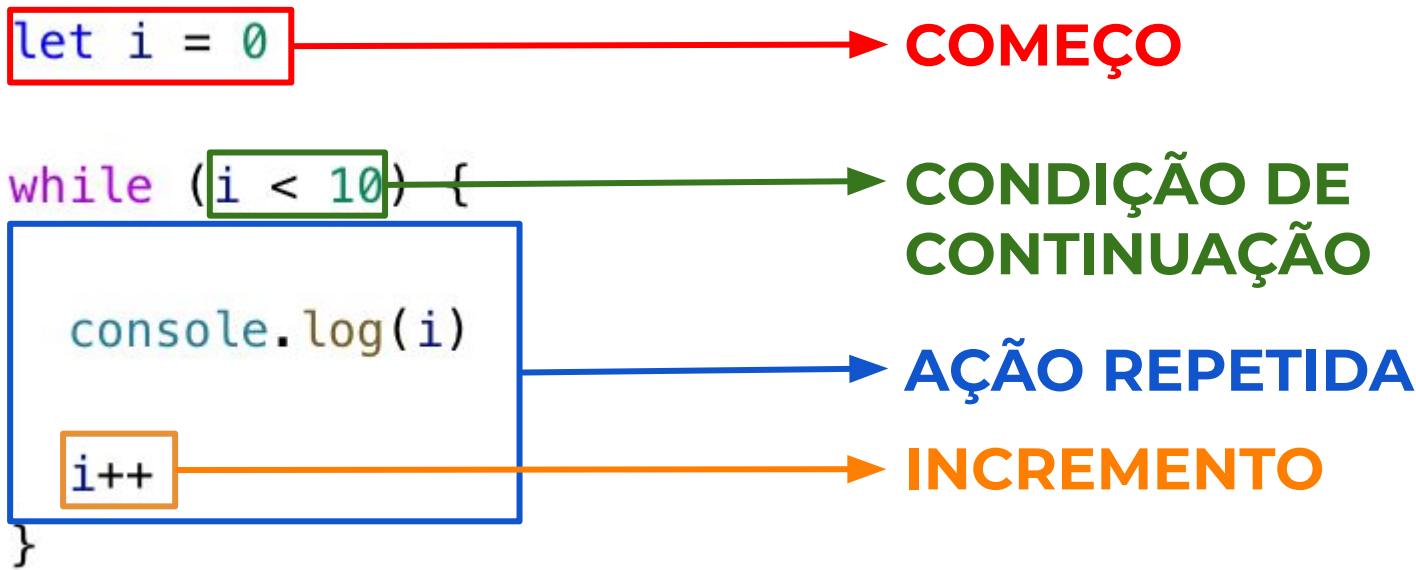
Laços

Labenu_



while

- Exemplo 1 - Imprimindo alguns números



for 🧢

- O laço **for** é uma maneira que permite **simplificar** a escrita de laços que tenham este comportamento

```
for(let i = 0; i < 10; i++) {  
  console.log(i)  
}
```

Diagram illustrating the components of a **for** loop:

- COMEÇO** (Start): `let i = 0;`
- CONDIÇÃO DE CONTINUAÇÃO** (Continuation Condition): `i < 10;`
- INCREMENTO** (Increment): `i++`
- AÇÃO REPETIDA** (Repeated Action): `console.log(i)`



for... of...

- Uma forma de simplificar a leitura dos elementos do array é utilizando o loop **for...of...**

```
const numeros = [14, 67, 89, 15, 23]

for(let numero of numeros) {
  console.log(numero)
}
```





Exercício 11

- Receba um array com números e devolva qual o maior dentro dele
- Ex: Para o array [11, 15, 18, 14, 12, 13], a saída deve ser: "O maior número é 18"
- Faça utilizando as três estruturas de loop vistas





Exercício 12

```
const prof = {  
  nome: "Letícia Chijo",  
  idade: 27,  
  aulasFront: true,  
  aulasBack: false,  
  jogosFavoritos: ["Chrono Trigger", "Undertale", "Hollow Knight"],  
  contaPiada: () => console.log("É pave ou pacume?"),  
  pet: {  
    nome: "Polly",  
    especie: "cachorrinha",  
    raca: "Lhasa Apso",  
    snacksFavoritos: ["biscoito", "maçã", "frango"]  
  }  
}
```





Exercício 12

- Dado o objeto do slide anterior, imprima:

*Oi! Eu me chamo **Letícia Chijo** e tenho **27** anos.*

***Dou/não-dou** aula de front e **dou/não dou** aula de back.*

Meus jogos favoritos são:

*1) **Chrono Trigger***

*2) **Undertale***

*3) **Hollow Knight***

*Tenho uma **cachorrinha** chamada **Polly** que gosta de comer **maçã***

- Faça a Chijo contar uma piada ruim



map e filter

Labenu_



map e filter



Função	Utilização	Retorna um <i>array</i> ?	Tamanho do array
<i>map</i>	Criar um novo <i>array</i> com elementos modificados em relação ao original	Sim	Igual ao original
<i>filter</i>	Criar um novo <i>array</i> com alguns elementos do original	Sim	Igual ou menor que o original



map - Sintaxe

```
const pokemons = [  
  { nome: "Bulbasaur", tipo: "grama" },  
  { nome: "Bellsprout", tipo: "grama" },  
  { nome: "Charmander", tipo: "fogo" },  
  { nome: "Vulpix", tipo: "fogo" },  
  { nome: "Squirtle", tipo: "água" },  
  { nome: "Psyduck", tipo: "água" },  
]  
  
const nomeDosPokemons = pokemons.map((pokemon, indice, array) => {  
  return pokemon.nome  
})
```



filter - Sintaxe



```
const pokemons = [  
  { nome: "Bulbasaur", tipo: "grama" },  
  { nome: "Bellsprout", tipo: "grama" },  
  { nome: "Charmander", tipo: "fogo" },  
  { nome: "Vulpix", tipo: "fogo" },  
  { nome: "Squirtle", tipo: "água" },  
  { nome: "Psyduck", tipo: "água" },  
]  
  
const apenasPokemonsDeGrama = pokemons.filter((pokemon, indice, array) => {  
  return pokemon.tipo === "grama"  
})
```





Exercício 13

Dado um array de produtos onde cada produto possui nome, categoria e preço:

- Retorne um novo array com todos os produtos porém com o 10% de desconto no preço
- Retorne um array apenas com os itens da categoria hortifruti
- Retorne um array apenas com os itens de hortifruti E com 10% de desconto no preço





Exercício 13

```
const produtos = [  
  { nome: "Alface Lavada", categoria: "Hortifruti", preco: 2.5 },  
  { nome: "Guaraná 2l", categoria: "Bebidas", preco: 7.8 },  
  { nome: "Veja Multiuso", categoria: "Limpeza", preco: 12.6 },  
  { nome: "Dúzia de Banana", categoria: "Hortifruti", preco: 5.7 },  
  { nome: "Leite", categoria: "Bebidas", preco: 2.99 },  
  { nome: "Cândida", categoria: "Limpeza", preco: 3.30 },  
  { nome: "Detergente Ypê", categoria: "Limpeza", preco: 2.2 },  
  { nome: "Vinho Tinto", categoria: "Bebidas", preco: 55 },  
  { nome: "Berinjela kg", categoria: "Hortifruti", preco: 8.99 },  
  { nome: "Sabão em Pó", categoria: "Limpeza", preco: 10.80 }  
]
```





Obrigado(a)!