JavaScript

JavaScript através de uma perspectiva lógica

Professor: **Eduardo Costa**

Histórico da Linguagem

JavaScript, nomeada inicialmente por LiveScript, é uma linguagem de programação criada em por Brendan Eich a pedido da Netscape no ano de 1995.

Na época a empresa SUN Microsystem (Java), hoje sob domínio da Oracle, entrou de cabeça na colaboração do desenvolvimento da linguagem, pois viu vantagem em sua ideia inicial, a validação de formulários web.

Foi criada para ser Server Side, porém acabou como Client Side.

A Microsoft percebeu o poder do JavaScript e ao invés de ajudarem no desenvolvimento acabou desenvolvendo sua própria linguagem, a Jscript para rodar apenas em seus navegadores o Internet Explorer.
Foi uma dor de cabeça para os desenvolvedores da época, pois deveriam dominar as duas linguagens para os seus sites rodarem em todos os navegadores.

A ECMA International, empresa dedicada a padronização de sistemas de informação, criou a especificação ECMA-262, utilizada em tecnologias web em geral e nomeada de ECMAScript.
A Netscape submeteu o JavaScript para as especificações ECMA-262 no final de 1996.
Tanto JavaScript como JScript são baseadas em ECMAScript porém com recursos adicionais.
Atualmente a linguagem ECMAScript adotou o nome JavaScript por motivos comerciais e hoje podemos considerar a linguagem padrão da Web.

Programa

Conceito

Um programa, independente de linguagem, é composto por um conjunto de instruções que são os passos para a resolução de um determinado problema.
A solução se divide em 3 etapas principais:

Exemplo

```
//entrada
    let n1 = prompt("Digite o primeiro número: ");
    let n2 = prompt("Digite o segundo número: ");

    //processamento
    let resultado = n1 + n2;

    //saída
    console.log("Resultado da operação "+ resultado);
```

- Tivemos os dados imputados pelo usuário;
- Processamento do cálculo previamente armazenados;
- E a exibição do resultado processado, para o usuário;
- ENTRADA – PROCESSAMENTO – SAÍDA;

Variável

Conceito

Variável é um recurso computacional que serve para armazenamento de dados temporários na memória.

A grosso modo é composta de um nome e um valor, como no exemplo anterior onde tivemos os números digitados pelo usuário armazenados em duas variáveis para utilização posterior, em um cálculo aritmético.

O valor de uma variável, claramente, varia com o tempo.

Exemplo 1

```
let n1 = parseFloat(prompt("Digite o primeiro número: "));
let n2 = parseFloat(prompt("Digite o segundo número: "));
```

Exemplo 2

Uma variável possui este nome por armazenar valores que podem variar com o tempo de execução do seu programa. Veja:

```
let n1 = 5;
n1 = n1 * 2;
alert(n1);
```

- Na linha 11 definimos uma variável com o nome n1 e valor 5 (não é necessário utilizar a instrução ”var”, devido ao fato de JavaScript ser de tipagem fraca);
- Na linha 13 o valor da variável n1 foi substituído pelo valor que ela tinha multiplicado por 2
- Na linha 15 foi exibido o valor da variável para o usuário. Repare que o valor foi é referente à última atualização da variável;

Regras básicas para definição

- Pode começar apenas com letras, _ (”underlines”) ou $ (”cifrão” apenas no início);
- Pode conter letras, underlines e números (caracteres alfanuméricos), a partir do segundo caracter;
- NÃO PODE conter espaços;
- NÃO PODE começar com números;

```
//Nomes válidos
let n1 = 5;
let nome_completo;
let _n1;
let _;
let $nome;
let $;

//Nomes Inválidos
let 1a;
let nome completo;
let valore$;
let nome _completo;
```

Tipos de Dados

Conceito

Tipos de dados são os possíveis valores que podem ser atribuídos às variáveis. São eles:

Number

Qualquer tipo numérico como real (float) ou inteiro (Integer);

String

cadeia de caracteres imutáveis onde cada caracter está contido em uma posição, da esquerda para a direita, a partir do zero e pode ser utilizado aspas duplas ou simples:

```
// posições 0123456
let nome = "Eduardo";
let sobrenome = ‘Costa’;
```

Bool

Valores que representam o estado verdadeiro (true) ou falso (false)

Undefined

Inexistência ou variável não criada.

Null

Ausência de valores. Neste caso a variável pode receber um valor null ou nunca ter recebido algum valor, porém a variável existe;

NaN

É um tipo especial, na realidade uma propriedade global do JavaScript cujo seu valor especifica que o valor não é um número. É uma propriedade somente leitura. Você não pode configurá-la.
Exemplo: a expressão (”Eduardo” * 10) retorna NaN.

```
let teste = "Eduardo" * 10;
console.log(teste === NaN);
console.log(isNaN(teste));
```

infinity

É um tipo especial que especifica que o valor é infinito. Quando se faz uma divisão por zero em algumas linguagens, é retornado o valor ”não existe divisão por 0”, porém matematicamente esse valor é um valor infinito tendendo a zero. Dessa forma, no JavaScript teremos o mesmo resultado.

```
let x = 5 / 0;
console.log(x); //infinity
```

Object e Arrays

Estruturas de dados formadas por pares de chave e valor

Array

> Tipo de dado composto por índices e valores onde esses valores podem ser qualquer tipo de dado suportado pela linguagem JavaScript, inclusive outro array.

```
// índices     0       1     2
let foo = ["eduardo", 36, 1.67];
```

object

> Tipo de dado composto por pares de chave e valores onde esses valores podem ser qualquer tipo de dado suportado pela linguagem JavaScript, inclusive outro objeto.

```
//objeto pedido
let pedido = {
    codigo : 5563,
    total : 131.98,
    itens : [//array de objetos
        {//item 0 - primeiro item
            "descricao" : "Camiseta Polo",
            valor : 60.98
        },
        {//item 1 - segundo item
            descricao : "Calça Jeans",
            valor : 60.98
        }
    ]
};
```

Veremos mais detalhes sobre arrays e objetos adiante.

Tipos de dados - API

Conceito

Para cada tipo de dado existe uma API que disponibiliza artifícios (métodos/funções) para sua manipulação. Vejamos alguns:

NUMBER

parseInt()

transforma uma string em um inteiro.

parseFloat()

transforma uma string ou um inteiro em um real.

toExponential()

converte o número em exponencial ().

toFixed(n)

fixa um número qualquer em n casas decimais

toString()

transforma um número em uma string.

veja alguns exemplos:

```
//NUMBER API
let numero = 10;
console.log( numero.toExponential(2) );
console.log( numero.toFixed(2));
console.log( numero.value(2) );
```

STRING

nome.charAt(2)

retorna o caracter na posição 2.

nome.charCodeAt(0)

retorna o código ascii do caractere na posição zero.

nome.concat(”@”)

concatena arroba no final de nome.

nome.indexOf(’d’)

retorna o índice da primeira ocorrência do caracter d.

nome.substring(0, 3)

– retorna uma substring a partir do caracter 0, com tamanho 3.

nome.replace(”du”, 33)

substitui as ocorrências de du por 33.

nome.split(’d’)

corta a string retorna um array.

Operadores

Conceito

Os operadores são recursos de uma linguagem de programação que lhe permitem aplicar alguma operação sobre os dados. São utilizados no processamento desses dados gerando assim a saída logicamente adequada àquela situação problema.

Operadores gerais

Segue abaixo a lista de operadores da linguagem JavaScript

OPERADOR	DESCRIÇÃO
+, -, *, /, %	soma, subtração, multiplicação, divisão, módulo (resto da primerira divisão inteira) operadores binários - 2 operandos ex: 2 + 2
+	concatena strings
+, -	operadores unários ex: -5, +12 - apenas um operando
<, >, <=, >=	menor, maior, menor ou igual, maior ou igual
++, --	pré ou pós incremento, pré ou pós decremento
==, !=	compara se é igual, compara se é diferente (faz a conversão e depois a comparação)
===, !==	compara se é igual e do mesmo tipo, compara se não é igual e nem do mesmo tipo
&&, ||	concatenação lógica (lembrar tabela verdade)
? :	Operador ternário ex: **5 < 6 ? console.log("menor") : console.log("maior");**
.	acessa a propriedade ou um método (função) de um objeto
=	operador de atribuição
+=, -=, *=, /=	atribui somando, atribui subtraindo, atribui multiplicando, atribui dividindo (x += y : soma x + y e **atribui o resultado da expressão** à variável x)

Operadores especiais

OPERADOR	DESCRIÇÃO
delete	torna a propriedade ou método de um objeto indefinida
in	verifica a existência de uma propriedade
typeof	verifica o tipo de dado da variável
new	cria um objeto
instanceof	verifica o tipo do objeto
void	retorna valor indefinido

Exemplos

delete

```
let conta = {
    saldo : 1150.75,
    sacar : function ( _valor ) {

        if( (this.saldo == 0) || this.saldo < _valor)
            return "Saldo insuficiente";

        this.saldo -= _valor;
        return "Saque realizado com sucesso";
    },

    depositar : function( _valor ){
        this.saldo += _valor;
        return "Depósito realizado com sucesso"
    }
}

console.log(conta);
console.log(conta.depositar(500.25));
```

```
console.log(conta)
delete conta.saldo;
console.log(conta);
delete conta.sacar;
console.log(conta);
```

in

```
let conta = {
    saldo: 1150.75,
    sacar: function (_valor) {

        if ((this.saldo == 0) || this.saldo < _valor)
            return "Saldo insuficiente";

        this.saldo -= _valor;
        return "Saque realizado com sucesso";
    },

    depositar: function (_valor) {
        this.saldo += _valor;
        return "Depósito realizado com sucesso"
    }
}

//retornará true
console.log('depositar' in saldo);

}
```

typeof

```
let conta = {
    saldo: 1150.75,
    sacar: function (_valor) {

        if ((this.saldo == 0) || this.saldo < _valor)
            return "Saldo insuficiente";

        this.saldo -= _valor;
```

```
        return "Saque realizado com sucesso";
    },

    depositar: function (_valor) {
        this.saldo += _valor;
        return "Depósito realizado com sucesso"
    }
}

console.log(typeof conta); //object
console.log(typeof conta === 'object'); //true

let pais = 'Brasil';
console.log(typeof pais); // 'string'
let numero = 70.35;
console.log(typeof numero); // 'number'
console.log(typeof numero === 'number'); //true
```

typeof

```
let objPessoa = new Object();
objPessoa.nome = "Eduardo Costa";
objPessoa.idade = 34;
objPessoa.envelhecer = function(){
    this.idade++;
}

console.log(objPessoa);// Object {nome: "Eduardo Costa", idade: 34,
envelhecer: function}
console.log(objPessoa instanceof Object);//true
```

void

```
let objPessoa = new Object();
objPessoa.nome = "Eduardo Costa";
objPessoa.idade = 34;
objPessoa.envelhecer = function () {
    this.idade++;
}

objPessoa.fazerNada = function () {
    return void(this.objPessoa);
```

```
}

console.log(objPessoa.fazerNada);
```

Expressões

As expressões montadas nesta seção servem para melhor entendimento das estruturas condicionais bem como programação em geral.

Ordem de precedência dos operadores:

1º Aritméticos

1º () parênteses – primeiro os mais internos
2º * / - multiplicação e divisão na ordem em que aparecer.
3º + - % - adição, subtração ou módulo na ordem em que aparecer.

2º Relacionais

== igual a
> maior que
< menor que
<= menor ou igual a.
>= maior ou igual a
!= diferente de

3º Lógicos

1º && - operador lógico and
2º || - operador lógico or
3º ! operador lógico not

Tabela verdade básica - &&:

true	**&&**	**true**	**=**	**true**
true	&&	false	=	false
false	&&	true	=	false
false	&&	false	=	false

Tabela verdade básica - ||:

true	||	true	= true
true	||	false	= true

false || true = true
false || false = false

Exercícios

Resolva as equações abaixo manualmente, sem a ajuda do computador, pois o intuito é dominar o pensamento relacionado às expressões e assim saber o que fazer exatamente na hora de programa:

a. 5+2
b. 5+2 > 5
c. 5+2 <= 7
d. 5+2 <= 7 == verdadeiro
e. 5+2 != 7 == falso
f. 5+2 != 7 == verdadeiro
g. 5+2 != 7 == verdadeiro ou falso
h. Falso e falso ou verdadeiro
i. 5+2 != 7 / 2 + 3.5 == verdadeiro e falso
j. verdadeiro e verdadeiro e verdadeiro ou verdadeiro e falso
k. ! (50 * (2 + 3) / 2) % 2 == 1
l. 50259 % 2 -1 == 1
m. 5 +2 + 3 - 2 / 4 == (3 + 1) / 2

Estruturas condicionais

Uma estrutura condicional é um desvio de fluxo baseado em uma análise: If(condição) { expressão}

Exemplo

```
let idade = 17;

if (idade >= 18) {
    console.log("Maior de idade");

} else {
```

```
    console.log("Menor de idade");
}

console.log("fim do programa");
```

No exemplo acima, sempre aparecerá a mensagem "*fim do programa"* porém "*Maior de idade"* e *"Menor de idade"* dependerá da condição/resultado da expressão *idade >= 10* que deverá retornar *true* ou *false*. As estruturas condicionais podem ser simples, compostas ou encadeadas.

Simples

Estrutura condicional simples contém apenas um desvio de fluxo. Em resumo é um *if* **sem** o *else* já que este é opcional nesta estrutura;

```
let cor = 'verde';
if (cor == 'verde' ) {
    document.body.style.backgroundColor  = "green";
}
```

Composta

Estrutura condicional composta é a estrutura if else:

```
let cor = 'purple';
if (cor == '' ) {
  //default
    document.body.style.backgroundColor  = "green";
} else {
    document.body.style.backgroundColor  = cor;
}
```

Encadeada

```
//cores da bandeira do Brasil
function myFunction(cor) {
    if (cor == 'green') {
        alert('cor válida: ' + cor);
        document.body.style.backgroundColor = "green";
    } else if (cor == 'yellow') {
```

```
        alert('cor válida: ' + cor);
        document.body.style.backgroundColor = cor;
    } else if (cor == 'blue') {
        alert('cor válida: ' + cor);
        document.body.style.backgroundColor = cor;
    } else if (cor == 'white') {
        alert('cor válida: ' + cor);
        document.body.style.backgroundColor = cor;
    } else {
        alert('cor não correspondente: ' + cor);
        document.body.style.backgroundColor = 'red';
    }
}
```

Estruturas de repetição

Estruturas de repetição são recursos que servem para repetir determinado trecho de código que se repete.

Suponha que você deva calcular a média geral de uma turma de 10 alunos. Para resolver esse problema a solução seria solicitar a nota de cada aluno. Esse trecho deveria ser repetido por 10 vezes e o seu código ficaria relativamente grande e a tarefa seria cansativa. É aqui que entra uma estrutura de repetição onde se escreve o trecho apenas uma vez e o chama por 10 vezes seguidas.

FOR

A estrutura FOR é composta por três partes onde cada uma é separada por ponto e vírgula (;) - a saber:

Variável de controle – é o início da contagem de repetições.
Condição de repetição e parada – enquanto for verdadeira o loop continua a rodar.
Critério de incremento – utilizado para incrementar a variável de controle.

```
for(var i = 0; i <= 5; i++) { //exibe os números de 0 a 5
console.log(i);
}
```

Dica: O **FOR cai bem quando sabemos a quantidade de repetições que devemos executar!**

Temos um problema onde temos que tirar a média de 4 alunos. Sem laço de repetição poderíamos fazer algo parecido com o código:

Exemplo 1

```
let media    = 0;
let nota     = 0;
let qtdAluno = 0;


nota += parseFloat(prompt("Qual a nota do primeiro aluno"));
qtdAluno++;
media = nota / qtdAluno;

nota += parseFloat(prompt("Qual a nota do segundo aluno"));
qtdAluno++;
media = nota / qtdAluno;

nota += parseFloat(prompt("Qual a nota do terceiro aluno"));
qtdAluno++;
media = nota / qtdAluno;

nota += parseFloat(prompt("Qual a nota do quarto aluno"));
qtdAluno++;
media = nota / qtdAluno;

console.log("Média: "+media.toFixed(2)+"\ntotal alunos: "+qtdAluno);
```

Podemos usar esse código normalmente, porém é custoso ter que digitar tudo. Imagina se depois tivesse que aumentar para a média de 20, depois 40, 100 alunos? Se tornaria inviável. Para isso existem os laços de repetição.

Exemplo 2

```
let media = 0;
let nota = 0;
let qtdAluno = 4;

for (let i = 0; i < qtdAluno; i++) {
    nota += parseFloat(prompt("Qual a nota do primeiro aluno"));
}

media = nota / qtdAluno;

console.log("Média: " + media.toFixed(2) + "\ntotal alunos: " +
qtdAluno);
```

WHILE

A estrutura WHILE pode ser comparada a uma estrutura IF simples. Porém o seu bloco de código será executado enquanto a condição for verdadeira. Veja:

Exemplo 1

```
var i = 1;
while  ( i < 6 ) { //exibe os números de 1 a 5
console.log(i);
}
```

ATENÇÃO - No exemplo anterior se você esquecer de incrementar manualmente a variável i, utilizada como controle, o seu bloco de código será executado infinitamente. Isso é conhecido como Loop Infinito. O correto seria:

Exemplo 2

```
var i = 1;
while  ( i < 6 ) { //exibe os números de 1 a 5
console.log(i);
i++
}
```

Dica: **O o while cai bem quando não sabemos com exatidão a quantidade de repetições que devemos executar o bloco de código!**

Funções

Conceito

De forma simples, funções são conjuntos de procedimentos isolados contendo alguma lógica pronta a ser utilizada quando for necessário. Pode ser definida ou incorporada dentro do escopo a ser utilizada. É útil para reutilização de regras lógicas que podem se repetir durante a execução do programa.

sua assinatura é composta da seguinte forma:

Exemplo 1

```
function helloWorld() {
      alert("Olá Mundo");
```

```
}

//exemplo de chamada
helloWorld();
```

Tivemos a palavra reservada ***function*** seguida do nome da função e bem como os parênteses ***()*** e o sinal de chaves ***{}*** que isola a lógica da função;

As funções podem ou não receber parâmetros que são suas variáveis internas e podem ou não retornar algum valor como resultado de seu processamento lógico isolado. O retorno explícito de uma função em JavaScript se dá através da palavra reservada ***return***.

Exemplo 2

```
<body>
    <button onclick="myFunction()">Rodar Exemplos</button>

    <p id="info">Visualize o resultado no console com F12.</p>
</body>
<script>



    // com parâmetro, sem retorno específico*
    function hello(nome) {
        window.alert("Olá" + nome);//undefined
    }

    // com parâmetro, com retorno específico*
    function helloWorld(nome) {
        return "Olá" + nome;//Olá Eduardo
    }

    // com parâmetro opcional, com retorno específico*
    function helloWorldSobrenome(nome, sobrenome = "**sem sobrenome**")
{
        return nome + " " + sobrenome;//Eduardo Eduardo **sem
sobrenome**
    }

    function myFunction() {
        let retorno = "";
```

```
        retorno = hello("Eduardo");
        console.log(retorno);

        retorno = helloWorld("Eduardo");
        console.log(retorno);

        retorno = helloWorldSobrenome("Eduardo");
        console.log(retorno);
    }
</script>
```

* Funções em JavaScript, caso não seja utilizado o ***return***, automaticamente seu retorno será ***undefined***;

Arrays

Conceito

Arrays são estruturas dinâmicas que podem armazenar diversos valores em um único nome de variável;
Antes você tinha uma variável que guardava um único valor por vez, agora você tem uma variável que pode armazenar vários valores. Veja:

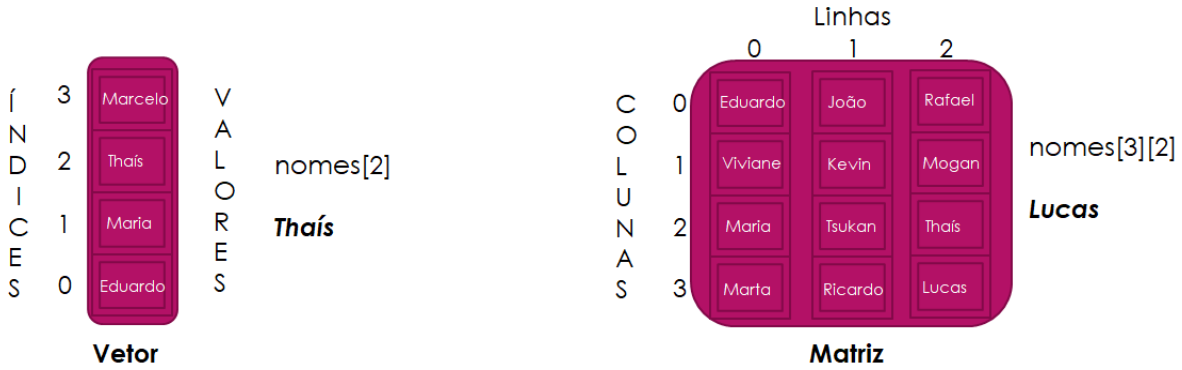
Exemplo 1

```
let nome = "Eduardo";
nome = "Maria";
console.log(nome);

let nomes = new Array()//array tamanho 0 - nenhum elemento
nomes[0] = "Eduardo";
nomes[1] = "Maria";
console.log(nomes);//array tamanho 2 - contém dois elementos
```

```
Maria
▼ (2) ["Eduardo", "Maria"] i
    0: "Eduardo"
    1: "Maria"
    length: 2
  ▶ __proto__: Array(0)
```

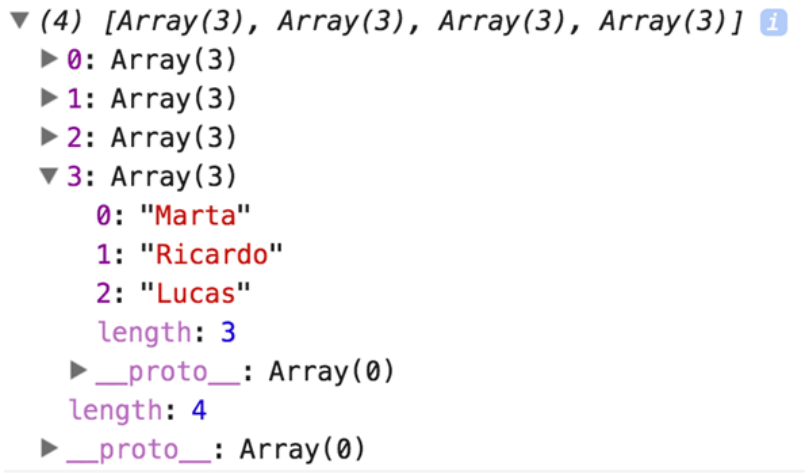
Quando pensamos em arrays devemos imaginar duas situações matemáticas - vetor e matriz;
Um vetor é uma estrutura composta de índices e valores. Veja a imagem abaixo:



Vimos vetores e matrizes. No JavaScript temos array dentro de array. Veja o exemplo do vetor do slide anterior impresso no console.

```
let matriz = [
    ["Eduardo", "João", "Rafael"],//0
    ["Viviane", "Kevin", "Morgan"],//1
    ["Maria", "Tsukan", "Thaís"],//2
    ["Marta", "Ricardo", "Lucas"]//3

];
console.log(matriz);
```



Exemplo 2

Abaixo seguem algumas formas de criar um Array:

```
// forma literal
let aLiteral = []; // array vazio
let valores = ["Marcelo", "Maria", 55, 99.28]; // inserindo valores

// forma orientada a objetos
let aObj  = new Array(); // array vazio
let aObj2 = new Array(5); // array tamanho 5
aObj[0] = "Marcelo";
aObj[1] = "Maria";
aObj.push(55);
aObj.push(99.28);
```

Objetos

Conceito

Objetos, na linguagem JavaScript, são coleções de pares chave/valor, separados por ponto e vírgula “;”, onde a chave se refere a uma propriedade do objeto e o valor a característica da propriedade. O valor pode assumir qualquer tipo de dado em JavaScript. Isso inclui funções ou novos objetos como valores de propriedades.

Os elementos que compõem um objeto, do ponto de vista do conceito de Orientação a Objetos, são: propriedades e métodos, onde os métodos são as funções que utilizamos como valores de uma propriedade deste objeto.

Exemplo 1

```
let objeto = {propriedade : "valor"};
console.log(objeto.propriedade);//valor
```

Exemplo 2

```
//Algumas formas de declarar os objetos

//forma literal
let objLiteral = {};

//utilizando o operador new - comun em outras linguagens
let objOO = new Object();
```

Exemplo 3

```
//exemplo extrapolado. Cada propriedade com um tipo diferente
let objeto2 = {
    prop1 : "valor",
    prop2 : 55,
    prop3 : ["morango", "limão", "abacate"],
    prop4 : function(){
        return "oi, sou um médodo do objeto2";
    },
    prop5 : {
        nome: "Lucas",
        idade: 5,
        dormir: function(){
            console.log("ZZzzZZzz");
        }
    },

    prop6 : 3.14,
    prop7 : null,
    prop8 : undefined,
    prop9 : Infinity
};
```

```
// retorno das propriedades
console.log(objeto.prop1);//"valor"
console.log(objeto2.prop7);//null
console.log(objeto2.prop9);//Infinity
console.log(objeto2.prop4);//"oi, sou um médodo do objeto2"
objeto2.prop5.dormir//imprime no console: ZZzzZZzz
console.log(objeto2.prop5.nome);//Lucas
```

Exemplo 4

Os objetos são dinâmicos e por isso podemos acrescentar ou diminuir propriedades conforme necessidade

```
let pessoa = new Object();// {}
pessoa.nome = "Rafael"; //{nome:"Rafael"}
pessoa.idade = 20; //{nome:"Rafael", idade:20}
pessoa.hobbies = ["filmes", "futebol"]; //{nome:"Rafael", idade:20,
hobbies:["filmes", "futebol"]}
```

Exercícios de lógica de programação

Instruções

Os exercícios de lógica de programação têm a finalidade de desenvolver a habilidade individual de cada aluno. Deverá ser feito em JavaScript puro.
Para cada exercício descreva os passos necessários como forma de comentário, com uma ação por linha, e em seguida elabore as instruções necessárias para a solução do problema.

OBS1: os exercícios devem ser feitos utilizando sempre funções como vimos nos exemplos estudados anteriormente.

OBS2: o desenvolvimento da lógica de programação é individual!

Introdução

1

Faça um programa que receba a idade de uma pessoa em anos e imprima essa idade em: Meses, Dias, Horas, Minutos.

2

Faça um programa que receba o ano de nascimento de uma pessoa e o ano atual. Calcule e imprima:

a. A idade dessa pessoa
b. Essa idade convertida em semana

3

Faça um programa que receba um número e exiba o seu dobro.

4

Faça um programa que receba o nome, o peso e a altura de uma pessoa. Calcule e imprima o nome e o IMC dessa pessoa - IMC = peso / (altura * altura).

5

Faça um programa que receba a medida em centímetros e exiba esse número em polegadas. OBS: Uma polegada equivale a 2.54 centímetros.

6

Faça um programa que receba a medida em polegadas e exiba esse número em centímetros.

7

Faça um programa que receba o nome, cargo e salário de um funcionário. Calcule o salário acrescido de 10%. Ao final exiba o nome, o cargo e o novo salário desse funcionário.

IF simples

8

Faça um programa que receba o nome, cargo e salário de um funcionário. Se o funcionário ganhar abaixo de 1000 reais, calcule o salário acrescido de 10%. Ao final exiba o nome, o cargo e o salário desse funcionário.

9

Uma empresa decide dar aumento de 30% aos funcionários cujo salário é inferior a 500 reais. Escreva um programa que receba o salário de um funcionário e imprima o valor do salário reajustado ou uma mensagem caso o funcionário não tenha direito a aumento.

10

Elabore um programa para cálculo de preços de produtos que solicite o preço de compra do produto e o percentual a ser aplicado em cima desse valor para a venda. Calcule e exiba o preço de venda do produto. Se o percentual a ser aplicado for inferior a 50%, exiba uma mensagem informando ao usuário que o produto será vendido com uma margem muito pequena de lucro.

11

Faça um programa que receba um número inteiro do usuário e informe se este número é positivo ou negativo.

12

Faça um programa que receba um número do usuário e informe se este número é par ou ímpar.

13

Faça um programa que receba um número de usuário e exiba este número, apenas se for par.

14

Faça um programa receba dois valores e imprima qual é o maior número digitado.

IF encadeado

15

Faça um programa que receba 4 notas de um aluno, calcule e imprima a média aritmética das notas e a mensagem de APROVADO para média superior ou igual a 7,0 RECUPERAÇÃO para notas entre 5.0 e 7,0 ou a mensagem de REPROVADO para média inferior a 5,0.

16

Uma empresa decide dar aumento aos funcionários de acordo com o seu cargo, disposto na tabela abaixo:

Cargo	**% Aumento**
Produção	6.5
Administrativo	7.5
Diretoria	12

De acordo com a tabela acima faça um programa que receba o cargo e o salário de um funcionário e calcule e imprima o salário reajustado.

17

Faça um programa que receba um número de usuário e exiba este número, apenas se for par. Caso contrário informe apenas “DIGITE APENAS NÚMEROS PARES”.

18

Faça um programa que receba um número de usuário e exiba este número, apenas se for par. Caso contrário informe apenas “DIGITE APENAS NÚMEROS PARES”.

Código de origem	**Procedência**
1	Sul
2	Norte
3	Leste
4	Oeste
5 ou 6	Nordeste
7, 8 ou 9	Sudeste

10 até 20	Centro-Oeste
acima de 20	Código ainda sem definição

19

Faça um programa que receba a idade de uma pessoa e classifique usando o seguinte critério:

Idade	**Classificação**
0 a 2 anos	Recém-Nascido
Recém-Nascido	Criança
12 a 19 anos	Adolescente
20 a 55 anos	Adulto
Acima de 55 anos	Idoso

20

Faça um programa que receba o nome, salário e código correspondente ao cargo do funcionário e imprima o seu nome, salário, código, cargo, Percentual de aumento e salário com aumento de acordo com o percentual da tabela abaixo:

Código	Cargo	Percentual
1	Escriturário	50%
2	Secretário	35%
3	Caixa	20%
4	Gerente	10%
5	Diretor	O cargo não receberá aumento

FOR

21

Faça um programa que receba 15 números inteiros e imprima na tela
a somatória dos 15 números.

22

Faça um programa que imprima os números ímpares de 0 a 50;

23

Faça um programa que imprima na tela a tabuada de um número qualquer escolhido pelo usuário.

24

Faça um programa que imprima na tela a tabuada de 1 a 10.

25

Uma certa empresa fez uma pesquisa de mercado com 10 pessoas para saber se elas gostaram um determinado produto lançado. Para isso forneceu o sexo do entrevistado e sua resposta (*sim* ou *não*). Faça um programa que calcule e imprima:

A. O número de pessoas que responderam SIM;
B. O número de pessoas que responderam NÃO;
C. O número de mulheres que responderam SIM;
D. A porcentagem de homens que responderam NÃO entre todos
E. os homens analisados.

26

Faça um programa que receba a idade, o peso e o sexo de 10 pessoas. Calcule e imprima:

- Total de Homens;
- Total de Mulheres;
- Média de idade dos Homens;
- Média de idade das mulheres.

While

27

Faça um programa que pergunte para o usuário a quantidade de números que ele quer digitar. Após isso crie um laço que exiba o número que o usuário digitar apenas se for PAR. Utilize a estrutura WHILE.

28

Faça um programa que pergunte para o usuário a quantidade de números que ele quer digitar. Após isso crie um laço que exiba o número que o usuário digitar apenas se for ÍMPAR. Utilize a estrutura WHILE.

29

Faça um programa que imprima na tela a tabuada de um número qualquer escolhido pelo usuário, utilizando a estrutura WHILE.

30

Faça um programa que dê três chances para o usuário acertar a senha e imprima quantas chances ele ainda tem. Para isso você deve utilizar a estrutura WHILE.

31

Altere o exercício anterior para que o usuário tenha a opção de desistir a qualquer momento.

32

Faça um programa que solicite números inteiros ao usuário e some em uma variável chamada de “resultado”. Dê a opção do usuário sair e, após sair, o programa exiba o resultado da somatória dos números digitados. Utilize a estrutura WHILE.

33

Altere o programa anterior para que seja somado apenas os números pares.

34

Altere o programa anterior para que o programa subtraia quando for ímpar e some quando for par.

35

.Faça um programa que solicite ao usuário um número de repetições “x”. Para cada repetição solicite dois números e imprima qual é o maior deles.

36

Faça um programa que solicite ao usuário um número de repetições “x”. Para cada repetição solicite dois números e imprima qual é o menor deles. Exiba a opção de sair para cancelar o laço de repetição.

Arrays

37

Faça um programa que contenha um array tamanho 10. Solicite ao usuário que digite números pares, inteiros, e os guarde no vetor. Para isso utilize a estrutura de repetição WHILE. Ao final, exiba todos os valores com a estrutura de repetição FOR.

38

Faça um programa que carregue 1 array tamanho 6 com números inteiros. Calcule e imprima a quantidade de números ímpares e a quantidade de números pares.

39

Faça um programa que receba a temperatura média de cada mês do ano e armazene essas temperaturas em um vetor. Calcule e imprima a maior e a menor temperatura do ano.

40

Faça um programa que carregue dois vetores de 10 elementos numéricos cada um e imprima a intercalação desses dois.

41

Faça um programa que receba dois vetores (nomes e medias) e armazene. Solicite ao usuário que informe o nome, a nota da prova e a nota do trabalho. Calcule a média e guarde os valores da média e do nome em seus respectivos vetores. Ao final exiba todos os dados.

42

Faça um programa que receba a nota de 10 alunos e armazene essas notas em um vetor. Calcule e imprima:

- A média da Classe;
- A quantidade de alunos APROVADOS (Média >= 7);
- A quantidade de alunos REPROVADOS (Média < 7)

Objetos

43

Faça um programa que crie uma variável objeto vazio, denominada produto. receba do usuário o nome do produto, o preço e o fornecedor e os preencha no objeto vazio criado anteriormente.

44

Faça um programa que crie uma variável objeto vazio, denominada produto. receba do usuário o nome do produto, o preço e o fornecedor e os preencha no objeto vazio criado anteriormente.

A. Crie um objeto denominado estoque com as propriedades cujos valores devem ser informados pelo usuário:

- percentualLucro: float que representa o percentual de lucro, ex 50;
- produtos: array de objetos.

B. Cada produto deve ter as seguintes propriedades com os respectivos tipos de dados:

- descricao : string,
- fornecedor :string;
- valorCompra: float;
- valorVenda : float;
- quantidade: int;

C. O valor de venda do produto deve ser calculado de acordo com o percentual de lucro. Ao final exiba o objeto cadastrado direto no console do chrome.

45

Crie uma função para exibir os produtos em estoque criado no exercício anterior.