Aluna: Luzaydha Sousa Santos
Atividade: Lista de Atividades Práticas – JavaScript

Esta lista contém 20 exercícios práticos organizados do nível básico ao avançado, baseados no conteúdo do material 'Aprendendo JavaScript' de Grillo & Fortes (2008).

Exercício 1 – [Variáveis e Tipos]

Declare variáveis usando var, let e const com os tipos: string, number, boolean, undefined e null. Exiba os valores no console.log.

Resposta: O material de referência foca na versão ES5 do JavaScript, utilizando a palavra-chave *var* para a declaração de variáveis. As palavras-chave *let* e *const* foram introduzidas em versões posteriores (ES2015/ES6) e não são abordadas no livro.
Numérico (Number), Texto (String), Lógico (Boolean), Indefinido (Undefined) e Nulo (Null).

```
//Declaração de variáveis e tipos de dados, conforme o livro.
// A sintaxe para declarar uma variável é: var nome = valor; [cite: 170]

// Tipo Texto (String) [cite: 202]
var meuNome = O "João"¡;

// Tipo Numérico (Número) [cite: 198, 201]
var idade = 30¡;

// Tipo Lógico (booleano) [cite: 205, 207]
var possuiHabilitacao = verdade verdadeira¡;

// Tipo Indefinido (Undefined) [cite: 208]
// Uma variável declarada sem um valor recebido automaticamente o tipo indefinido. [cite: 209]
var corFavorita¡;

// Tipo Nulo (Null) [cite: 211]
// O tipo nulo representa a ausência intencional de um valor de objeto. [cite: 212]
// No exemplo do livro, ele está associado a uma variável que não foi declarada. [cite: 213]
// Para atribuí-lo diretamente:
var sobrenome = nulo¡;

// Exibindo os valores no console [cite: 171]
consola.O.log log((S)"String:",, meuNome(S)¡;
consola.O.log log((S)"Number:",, idade(S)¡;
consola.O.log log((S)"Boolean:",, possuiHabilitacao(S)¡;
consola.O.log log((S)"Undefined:",, corFavorita(S)¡;
consola.O.log log((S)"Null:",, sobrenome(S)¡;
```

Exercício 2 – [Operadores Aritméticos]

Crie um script que calcule e exiba a soma, subtração, multiplicação, divisão e resto entre dois números.

Resposta: Os operadores são utilizados para a realização de diversas operações, incluindo as aritméticas. A tabela de operadores do livro menciona *Adição (+), Subtração (-), Multiplicação (*), Divisão (/) e Resto (%).*

```
var numeroA = 10;
var numeroB = 5;

// Operações aritméticas
var soma = numeroA + numeroB;
var subtracao = numeroA - numeroB;
var multiplicacao = numeroA * numeroB;
var divisao = numeroA / numeroB;
var resto = numeroA % 3; // Resto de 10 dividido por 3

// Exibindo os resultados no console
console.log("Soma:", soma);
console.log("Subtração:", subtracao);
console.log("Multiplicação:", multiplicacao);
console.log("Divisão:", divisao);
console.log("Resto:", resto);
```

Exercício 3 – [Operadores de Comparação]

Crie um script que compare duas variáveis e imprima mensagens distintas com base no resultado.

Resposta: O livro aborda o uso de operadores lógicos e de comparação dentro de estruturas de decisão como *if* e *else*. A tabela de operadores lista "Maior Que" (>), "Menor Que" (<), "Igualdade" (==), entre outros.

```
var valor1 = 100;
var valor2 = 50;

console.log("Comparando", valor1, "e", valor2);

// A expressão if/else nos habilita a tomar decisões dentro do código [cite: 335]
if (valor1 > valor2) {
  console.log("O valor 1 é maior que o valor 2.");
} else {
  console.log("O valor 1 não é maior que o valor 2.");
}
```

Exercício 4 – [Condicional if/else]

Crie um programa que pergunte a idade de uma pessoa e informe se ela é menor, maior ou idosa (idade > 65).

Resposta: Para solicitar informações do usuário, pode-se usar a função *prompt.* Para múltiplas condições, a estrutura *else if* é a mais adequada, permitindo criar uma cadeia de decisões lógicas.

```
// A função prompt solicita informações do usuário. [cite: 249]
var idade = prompt("Qual é a sua idade?");

// O if/else if/else permite criar diversas decisões lógicas para uma variável. [cite: 4
if (idade < 18) {
  alert("Você é menor de idade.");
} else if (idade >= 18 && idade <= 65) {
  alert("Você é maior de idade.");
} else {
  alert("Você é considerado(a) idoso(a).");
}
```

Exercício 5 – [Operador Ternário]

Reescreva o exercício anterior usando operador ternário para avaliar se a pessoa é maior de idade.

Resposta: O operador ternário não é abordado no livro "Javascript Básico ao Avançado". Ele é uma forma concisa de escrever uma instrução *if/else*. A sintaxe é: *condicao ? valor_se_verdadeiro : valor_se_falso.*

```
var idade = prompt("Qual é a sua idade?");

// O operador ternário não é mencionado no livro.
// A linha abaixo é uma forma abreviada do if/else.
var status = idade >= 18 ? "Maior de idade" : "Menor de idade";

alert(status);
```

Exercício 6 – [Laço for]

Crie um script que imprima os números de 1 a 10 usando for.

Resposta: O laço de repetição *for* é o mais utilizado e sua estrutura consiste em um contador, uma condição e um incremento.

```
// A estrutura do for é: for (contador; condição; incremento) [cite: 750]
for (var i = 1; i <= 10; i++) {
  // O código será executado enquanto a condição do laço retornar verdadeiro. [cite: 762]
  console.log(i);
}
```

Exercício 7 – [Laço while e do...while]

Escreva um script que use while e do...while para contar até 5 e explique a diferença entre os dois.

Resposta: O laço *while* executa um bloco de código enquanto uma condição especificada for verdadeira. O laço *do...while,* que não é abordado no livro, é uma variação que executa o bloco de código uma vez *antes* de verificar a condição.

- ***while****:* A condição é avaliada *antes* da execução do bloco. Se a condição for falsa inicialmente, o bloco nunca é executado.
- ***do...while***: O bloco é executado *pelo menos uma vez* e, só então, a condição é avaliada para determinar se a execução deve continuar.

```
// O laço while não tem um local definido para contador ou incremento [cite: 788]
var contador = 1;

while (contador <= 5) {
  console.log("Contagem:", contador);
  contador++; // O incremento é feito dentro do bloco
}
```

Exercício 8 – [Funções Simples]

Crie uma função chamada saudacao que receba um nome e retorne uma saudação personalizada.

Resposta: Uma função é um bloco de código que pode ser reutilizado. Ela pode receber informações como parâmetros e devolver um novo valor através da instrução

Return.

```
// Para declarar uma função, usamos a palavra reservada 'function' [cite: 520]
function saudacao(nome) {
  // A função vai manipular os parâmetros e depois devolver um novo valor. [cite: 471]
  return "Olá, " + nome + "! Seja bem-vindo(a).";
}

var mensagem = saudacao("Maria");
console.log(mensagem);
```

Exercício 9 – [Funções como Literais]

Implemente uma função anônima atribuída a uma variável para calcular a área de um triângulo.

Resposta: Segundo o livro diferencia *declarações de função* de *expressões de função*. Uma expressão de função consiste em armazenar uma função anônima dentro de uma variável.

```
// Na expressão de função, estamos armazenando o valor da função em uma variável [cite:
var calcularAreaTriangulo = function(base, altura) {
  var area = (base * altura) / 2;
  return area;
};

var resultado = calcularAreaTriangulo(10, 5);
console.log("A área do triângulo é:", resultado);
```

Exercício 10 – [Arrays e Métodos]

Crie um array com 5 frutas e utilize métodos como .push(), .pop(), .slice() e .join().

Resposta: Arrays são usados para agrupar informações em uma única variável. O livro detalha alguns métodos, como *push* (adiciona um item no final) e *pop* (remove o último item). Os métodos *slice()* e *.join()* não são mencionados no material.

```
consola.0.log log((S)"Olá, Mundo!"(S)¡;// A forma mais comum de criar um array é utilizando colchetes [cite: 604]
var frutinhas = [[]A 'Maçã',, 'Banana',, A 'Laranja',, O 'Morango',, 'Uva']¡;
consola.0.log log((S)"Array original:",, frutinhas(S)¡;

// A função push consiste em adicionar um valor depois da última posição
frutinhas.0.empurre((S)0 'Abacaxi'(S)¡;
consola.0.log log((S)"Depois do push:",, frutinhas(S)¡;

// A função pop consiste em removedor a última posição de um array
frutinhas.0.pop((S)(S)¡;
consola.0.log log((S)"Depois do pop:",, frutinhas(S)¡;

//Ó métodô .slice() não é mencionado no livro. Ele retorna uma copa de parte de um array.
var algumasFrutas = frutinhas.0.fatia((S)1 1,, 3 (s)(S)¡; // Pega do índice 1 até o 2 (3 não incluso)
consola.0.log log((S)"Resultado do slice (1, 3):",, algumasFrutas(S)¡;

//Ó métodô .join() não é mencionado no livro. Ele junta todos os elementos de um array em uma string.
var frutasCorda = frutinhas.0.junte-se((S)' - '(S)¡;
consola.0.log log((S)"Resultado do join(' - '):",, frutasCorda(S)¡;
```

Exercício 11 – [Objeto Literal]

Crie um objeto carro com atributos (modelo, ano, motor) e exiba-os no console.

Resposta: Objetos permitem agrupar dados usando um sistema de par chave-valor. Uma das formas de declarar um objeto é usando chaves

```javascript
// Criando um objeto com chaves
var carro = {
  modelo: 'Fusca',
  ano: 1975,
  motor: 1.5
};

// Acessando as propriedades com a notação de ponto [cite: 689]
console.log("Modelo:", carro.modelo);
console.log("Ano:", carro.ano);
console.log("Motor:", carro.motor);
```

Exercício 12 – [Métodos em Objetos]

Adicione ao objeto carro um método consumo(km) que retorne o consumo estimado com base em motor.

Resposta: É possível adicionar funções como propriedades de um objeto, que são chamadas de métodos. Dentro de um método, a palavra-chave

```
var carroça = {
  modeloo¡:: 'Fusca',,
  ano¡:: 1975,,
  motor¡:: 1.5,,
  // Adicionando um métodô ao objeto
  consumo¡:: function (função)((S)km km(S) {
    // 'this' se refere ao objeto atual em missão
    var consumoMédio = km km / isto isto.O.motor¡;
    retorno "O consumo estimado para " + + km km + + "km é de" + + consumoMédio.O.toFixo((S)2 (2)(S) + + "litros."name¡;
  O}
O}¡;

consola.O.log log((S)carroça.O.consumo((S)100))¡;
```

Exercício 13 – [Manipulação de Strings]

Use métodos .charAt(), .indexOf(), .toUpperCase(), .replace() em uma string de exemplo.

Resposta: O livro de referência não possui um capítulo dedicado a métodos de manipulação de Strings. As operações abaixo são funcionalidades padrão do JavaScript para trabalhar com o tipo Texto (String).

```javascript
var frase = "Aprendendo JavaScript";

// .charAt(index): Retorna o caractere em uma posição específica.
var primeiraLetra = frase.charAt(0); // 'A'
console.log("Primeira letra:", primeiraLetra);

// .indexOf(substring): Retorna a posição da primeira ocorrência de um texto.
var posicao = frase.indexOf("Java"); // 10
console.log("Posição de 'Java':", posicao);

// .toUpperCase(): Converte a string para maiúsculas.
var maiuscula = frase.toUpperCase();
console.log("Em maiúsculas:", maiuscula);

// .replace(antigo, novo): Substitui uma parte da string por outra.
var novaFrase = frase.replace("JavaScript", "Python");
console.log("Frase substituída:", novaFrase);
```

Exercício 14 – [Função com Argumento de Função]

Crie uma função que receba outra função como argumento e aplique sobre um número.

Resposta: JavaScript trata funções como "cidadãs de primeira classe", o que permite passá-las como parâmetros para outras funções. A função passada como argumento é chamada de

```
// Função que recebe outra função como argumento (fun) [cite: 144
function aplicarOperacao(numero, operacaoCallback) {
  return operacaoCallback(numero);
}

// Funções que servirão como callback
function dobrar(num) {
  return num * 2;
}

function triplicar(num) {
  return num * 3;
}

// Passando a função 'dobrar' como argumento [cite: 1450]
var resultadoDobro = aplicarOperacao(5, dobrar);
console.log("O dobro de 5 é:", resultadoDobro);

// Passando a função 'triplicar' como argumento
var resultadoTriplo = aplicarOperacao(5, triplicar);
console.log("O triplo de 5 é:", resultadoTriplo);
```

Exercício 15 – [Manipulação do DOM]

Crie uma página HTML com uma . Use JavaScript para alterar o conteúdo e o estilo desta div ao clicar em um botão.

Resposta: O livro "Javascript Básico ao Avançado" foca exclusivamente na linguagem JavaScript (sua sintaxe, lógica e ambiente de execução), não abordando a manipulação do DOM (Document Object Model), que é a forma como o JavaScript interage com os elementos de uma página HTML.

O código a baixo desmostraa prática padrão para essa tarefa:

```
<!TIPO MÉDICO html> >
<html> >
<cabeça de> >
    <título (título)> >Exercitório DOM</título (título)> >
</cabeça de> >
<corpo> >
    <div. id.="a ssg"> >Texto inicial.</div.> >
    <botão id.="meuBotao"> >Clique em Mim</botão> >
    <roteiro src="roteiro.js"> ></roteiro> >
</corpo> >
</html> >
```

```
// Oblendo os elementos da página
var botao = documento.O.getElementById((S)‘meuBotao’(S)¡;
var divMensagem = documento.O.getElementById((S)´msg‘(S)¡;

// Adicionando um "ouvinte de evento" ao botão
botao.O.AdicionarOuvidor de eventos((S)‘clique’,, function (função)((S)(S) {
  // Alterando o conteúdo da div
  divMensagem.O.innerHTML = "O texto foi alterado!"¡;

  // Alterando o estilo da div
  divMensagem.O.estilo.O.cor cor = ‘azul’¡;
  divMensagem.O.estilo.O.fontWeight = ‘ousado’¡;
})¡;
```

Exercício 16 – [Criação Dinâmica de Elementos]

Crie um novo parágrafo usando document.createElement() e o adicione dinamicamente ao body.

Resposta: Assim como o exercício anterior, a criação dinâmica de elementos HTML via JavaScript (manipulação do DOM) não é um tópico coberto pelo livro. A seguir, a solução:

```
// 1. Criar um novo elemento de parágrafo
var novoParagrafo = document.createElement('p');

// 2. Definir o conteúdo do novo parágrafo
novoParagrafo.textContent = "Este parágrafo foi criado dinamicamente com JavaScript.";

// 3. Adicionar o novo parágrafo ao final do <body> da página
document.body.appendChild(novoParagrafo);
```

Exercício 17 – [Função com Propriedade Prototype]

Crie uma função construtora Pessoa(nome, idade) e adicione um método falar() via prototype.

Resposta: O livro aborda extensivamente o uso de funções construtoras para criar "modelos" de objetos. Para evitar a duplicação de métodos em cada instância, a prática correta é adicioná-los à propriedade *prototype* do construtor. Isso faz com que todas as instâncias herdem o método

```
// A primeira letra em maiúsculo é um padrão para identificar um construtor [cite: 1208]
var Pessoa = function(nome, idade) {
  this.nome = nome;
  this.idade = idade;
};

// Adicionando um método ao prototype do construtor [cite: 1234]
Pessoa.prototype.falar = function() {
  console.log("Olá, meu nome é " + this.nome + ".");
};

// Para criar um objeto, usa-se o operador 'new' [cite: 1223]
var joao = new Pessoa('João', 25);
var ana = new Pessoa('Ana', 30);

// Todas as instâncias herdam o método 'falar' do prototype [cite: 1194]
joao.falar();
ana.falar();
```

Exercício 18 – [Tratamento de Exceções]

Crie uma função que gere uma exceção *(throw)* se um número for negativo e trate com *try/catch.*

Resposta: O tratamento de exceções com as palavras-chave *try, catch* e *throw* não é um tópico abordado no material de referência. Abaixo está uma implementação padrão.

```
function verificarNumeroPositivo(numero) {
  if (numero < 0) {
    // 'throw' lança uma exceção, interrompendo o fluxo normal
    throw new Error("O número não pode ser negativo.");
  }
  return "O número " + numero + " é válido.";
}

// O bloco 'try' contém o código que pode gerar uma exceção
try {
  console.log(verificarNumeroPositivo(10));
  console.log(verificarNumeroPositivo(-5)); // Esta linha irá gerar o erro
  console.log("Esta linha não será executada.");
} catch (erro) {
  // O bloco 'catch' é executado se uma exceção for capturada
  console.error("Ocorreu um erro:", erro.message);
}
```

Exercício 19 – [XMLHttpRequest Simples]

Implemente uma chamada AJAX usando XMLHttpRequest para buscar um arquivo .txt local e exibir o conteúdo na página.

Resposta: O livro não cobre APIs de navegador como *XMLHttpRequest,* que é usada para fazer requisições a servidores (AJAX). Esta tecnologia permite que uma página da web seja atualizada de forma assíncrona, trocando dados com um servidor em segundo plano.

```
// Cria uma nova instância do XMLHttpRequest
var xhr = new XMLHttpRequest();

// Configura a requisição: método GET para o arquivo 'arquivo.txt'
xhr.open('GET', 'arquivo.txt', true);

// Função a ser executada quando o estado da requisição mudar
xhr.onreadystatechange = function() {
  // Verifica se a requisição foi concluída (readyState 4) e bem-sucedida (status 200)
  if (xhr.readyState === 4 && xhr.status === 200) {
    // Exibe o conteúdo do arquivo no console
    console.log(xhr.responseText);
    // Poderia também adicionar a um elemento da página:
    // document.body.append(xhr.responseText);
  }
};

// Envia a requisição
xhr.send();
```

Exercício 20 – [Mini Projeto Ajax + DOM]

Desenvolva uma página que carrega dinamicamente notícias de um XML usando AJAX e exibe na tela.

Resposta: Este exercício combina AJAX e manipulação do DOM, dois tópicos que não são cobertos pelo livro "Javascript Básico ao Avançado". A solução envolve buscar um arquivo XML, interpretar (fazer o *parse*) de sua estrutura e, em seguida, criar elementos HTML dinamicamente para exibir os dados. **CODIGO EM: https: //github.com/luzaydha/webdesing/tree/main/U1-SE%C3%87%C3%83O3**