Bibliografia recomendada:

- JavaScript: O Guia Definitivo
- JavaScript: Guia do Programador
- Guia da Mozilla
- Guia da ECMA

Operadores:

- aritméticos (sempre lembrar da ordem precedência);
  - +; adição
  - -; subtração
  - *; multiplicação
  - /; divisão real
  - %; resto da divisão divisão inteira
  - **; potência
- obs: em javascript, também temos a simplificação da autoatribuição, sendo ela igual a do python: n += 4; n -= 2; n*= 2…
- incremento;
  - x++ ou ++x;
  - x-- ou -- x;
- relacionais;
  - >; maior que
  - <; menor que
  - >=; maior ou igual
  - <=; menor ou igual
  - ==; igual
    - ===; operador de igualdade restrita - igual para caractere e tipo
  - !=; diferente
  - sempre retornam valores booleanos
- lógicos;
  - !; negação - !true = false; !false = true
  - &&; conjunção (and)
  - ||; disjunção (or)
  - exemplo:
    - var status = (idade >= 18) ? 'Adulto' : 'Menor';
    - aqui, a variável status vai guardar a string correspondente.
- ternário;
  - trabalha com ‘?:’
  - teste ? true : false - media>=7 ? ‘Aprovado’ : ‘Reprovado’
  - basicamente: [condição] ? [valor_vardadeiro] : [valor_falso]
- ordem de precedência em uma expressão condicional:
  - aritméticos;
  - relacionais;
  - ! - negação;
  - && - and;
  - || - or;

Comandos:

- window.alert(‘Mensagem’); - gera uma janela de alerta

window.confirm(‘Mensagem’); - gera uma janela com confirmação ou não
window.prompt(‘Mensagem’); - gera uma janela com prompt de entrada
document.write(‘Mensagem’) - escreve no documento (html)

Variáveis:
- var nomeDaVariavel = valor;
- definir variável numérica:
  - var num1 = 10;
  - temos números ‘especiais’, sendo eles o infinity e NaN (not a number)
- definir variável booleana:
  - var valor = true;

Datatypes:
- podemos usar o typeof para identificar o tipo da variável
- number:
  - temos números ‘especiais’, sendo eles o infinity e NaN (not a number)
  - Number.parseInt(n) - converte para inteiro
  - Number.parseFloat(n) - converte para float
  - posso utilizar somente Number(n) - o js define a partir da entrada
- string
  - string(n) - posso converter number para string OU
  - n.toString()
- boolean
- null
- undefined
- object
  - array
- function

Atributos:
- variavel.length - qual o tamanho da string ou do array
- variavel.replace(‘oqTenho’, ‘oqQuero’) - troca um caractere por outro
- variavel.toUpperCase() - transforma todas as letras em maiúsculas
- variavel.toLowerCase() - transforma todas as letras em minúsculas
- numero.toFixed(2) - formata o numero para dois pontos flutuantes

DOM:
- o que é? document object model - é um conjunto de objetos dentro do navegador que permite o acesso aos componentes internos do seu website
- window.document.write(‘mensagem’) - escreve uma mensagem no documento
  - window.document.write(window.document.charset) - também podemos colocar outras interações dentro do document.write
    - exemplos de interações:
      - window.document.URL;
      - window.document.navigader.appName;

Maneiras de acessar os elementos do DOM:
- por Marca:

p1 = getElementsByTagName()[qualParágrafoDesejaSelecionar]
document.write(pi.innerText) - o innerText pega o que está escrito dentro do parágrafo selecionado
temos uma pequeno detalhe entre innerText e innerHTML
innerText seleciona apenas o texto do parágrafo
innerHTML seleciona o texto e as tags que estão modificando esse parágrafo, trazendo o resultado igual o do parágrafo original
também é possível adicionar style diretamente pela variável que aponta para o parágrafo: p1.style.color = 'blue'

resumo geral de como usar acessar elementos na prática:

```
<div id="msg" name="teste" class="sim">Click em mim</div>
// maneiras diferentes de selecionar a mesma div
var d1 = window.document.getElementsByTagName('div')[0]
var d1 = window.document.getElementById('msg')
var d1 = window.document.getElementsByName('teste')[0]
var d1 = window.document.getElementsByClassName('sim')[0]
var d1 = window.document.querySelector('div#msg')
```

por ID
getElementById()
por Nome
getElementsByName()
por Classe
getElementsByClassName()
por Seletor (mais recomendado atualmente)
querySelector()
querySelectorAll()

Árvore DOM (elementos):

- window
  - location
  - document
    - html
      - head
        - meta
        - title, etc
      - body, etc…
        - h1
        - p
        - p
          - strong
        - div
  - history, etc…

Eventos DOM

- mouseenter
- mousemove
- mousedown

mouseup
click
mouseou

Funções
estrutura básica

```
function nomeDaFuncao() {
    bloco de código
}
```

podemos disparar funções diretamente do html (segundo o guia da mozilla é válido mas não é recomendado pois deixa o html poluído)
exemplo de utilização das funções diretamente do html

```
<div id="area" onclick="clicar()" onmouseenter="entrar()"
var a = window.document.querySelector('div#area')
function clicar() {
    a.innerText = 'Clicou!'
}

function entrar() {
    a.style.background = 'blue'
}

function sair() {
    a.style.background = 'rgb(83, 184, 83)'
}
```

Funções - Avançando
são ações executadas assim que são chamadas ou em decorrência de algum evento
uma função pode receber parâmetros e retornar um resultado
estrutura básica

```
function acao(parametros) {
    bloco de código
    return res
}

acao(5)
```

exemplo simples

```
function parImpar(n) {
    if (n % 2 === 0) {
        return 'par'
    } else {
        return 'impar'
    }
}

let res = parImpar(5)
```

```
// Ao colocar esses valores você pode impedir o programa de quebrar caso não
receba dois parâmetros
function soma(n1=0, n2=0) {

posso colocar uma funcao dentro de uma variável
    let v = function(x) {
       return x * 2
    }

    console.log(v(5))

recursividade - função chama ela mesma
    function fatorial(n) {
       if (n === 1) {
          return 1
       } else {
          return n * fatorial(n-1)
       }
    }

    console.log(fatorial(5))
```

Condições

- estrutura básica
  - if (condicao) {
    bloco de código se true
    }
    else {
    bloco de código se false
    }
- tipos de condição
  - simples
    só conta com o bloco do if
  - composta
    conta o bloco do if e do else
  - aninhadas
    if (condicao1) {
    bloco de código se true
    }
    else if (condicao2 {
    bloco de código se for false e true
    }
    else {
    bloco de código se for false false}
  - múltipla
    switch (expressão) {
    case valor1:

bloco de código
break
case valor2
bloco de código
break
case valor3
bloco de código
break
default
bloco de código
break
}
dica: default funciona de maneira semelhante ao else
obs: para funcionamento ideal, é necessário utilizar o break, pois caso entre no valor2 ele encerrar no valor2 com o break, caso contrário, iria iria continuar executando os blocos
obs: switch só funciona com números inteiros e caracteres únicos

Estruturas de controle

estrutura de repetição com teste lógico no início - while

```
while (condicao) {
    bloco de código
}
```

estrutura de repetição com teste lógico no final - do while

```
do {
    bloco de código
}
while (condicao)
```

estrutura for

```
for (inicio; teste; incremento) {
    bloco de código
}
```

Variáveis compostas

funcionam de maneira semelhantes às listas do python
devem ser capazes de armazenar vários valores em uma mesma estrutura
como declarar uma variável composta

```
let num = [1, 2, 3]
```

uma array é composta por elemento, que é um par que agrupa o espaço colocado na memória, o valor dentro dele e o índice;
como adicionar elementos dentro de uma array

```
// Adiciona um elemento na posição exata
num[3] = 6;
console.log(num)

// Adiciona um elemento na última posição
num.push(7)
console.log(num)
```

num.length - retorna o tamanho da array *notar que nesse caso não precisa de parênteses
num.sort() - coloca os valores em ordem crescente
como passar por todos os itens de uma array

```
for (let i in num) {
    console.log(num[i])
}
OU
for (let pos = 0; pos<num.length; pos++) {
    console.log(num[pos])
}
```

como buscar a posição de um item exato
num.indexOf(7) - retornando a chave (index) onde esse valor está
se o valor não existir na array, ela retorna -1

declarando um objeto
let amigo = {nome:'Leo', idade:21}
posso colocar funções dentro de um objeto
let amigo = {nome:'Leo', idade:21, aumentarIdade(idd){}}

Obs:
para podermos utilizar a template string, invés de aspas deveremos usar crases
n1.toLocaleString('pt-BR', {style: 'currency' , currency: 'BRL'}) - formata o valor para a moeda de localização global, nesse caso pt-BR formata em BRL;

```
// podemos ver a diferença entre innerText e innerHTML aqui:
// window.alert(p1.innerHTML)
// window.alert(p1.innerText)
```

para adicionarmos MAIS texto em uma única div com o innerHTML ou innerText podemos seguir a estrutura

```
espaco.innerHTML=`<p>Sua velocidade atual é <strong> ${velo}km/h </strong> </p>`
espaco.innerHTML += `<p>Você está acima do limite de velocidade. <strong>MULTADO!</strong></p>`
```

sempre lembrando do +=

como pegar a hora do sistema:

```
var agora = new Date()
var hora = agora.getHours()
```

também é possível pegar os minutos, dia, ano, etc, apena trocando o getHours

como pegar o resultado de um input:radio

```
var sexo = document.querySelector('input[name=sexo]:checked').value
```

lembrando que o name muda de acordo com o radio

como definir um label para uma radio

```
        <input type="radio" name="sexo" id="homem" value="homem" checked>
        <label for="homem">Maculino</label>
        lembrar que o for do label tem que ser o id do radio
como inserir um elemento dinamicamente com js (exemplo para adicionar imagem)
        var img = document.createElement(‘img’) // cria o elemento
        img.setAttribute(‘id’, ‘foto’) // adiciona o id=’foto’ na tag img
        img.setAttribute(‘src’, ‘foto.png’)
        res.appendChild(img) // res é uma div já coletada anteriormente

principais diferenças entre while e do while
        while - verifica primeiro o teste lógico e executa o bloco de comando
        do while - primeiro executa o bloco de comando e verifica o teste lógico

como remover um elemento dinamicamente com js
        campoReset.removeChild(document.querySelector('input#b-reset'))
        se você já tiver definido o elemento a ser removido anteriormente, pode
        passa-lo diretamente no removeChild, assim
                campoReset.removeChild(campoFilhoDefinido)

como identificar se um input está vazio
        let txtNum = document.querySelector('input#num');
        if (txtNum.value.length === 0) {}

como arredondar um valor
        ${media.toFixed(2)} // arredonda para duas casas

como saber se um número já foi adicionado na lista
        numeros = [1, 2, 3]
        num = 5
        if (numero.includes(num))

como limpar um input após clicar no botao
        let txtNum = document.querySelector('input#num')
        txtNum.value = ''

como acessar o elemento diretamente pelo for
        for (let n of numeros) {
           soma += n
           // o n já assume o valor do elemento sem precisar fazer numeros[n]
        }
```