**REFERENCIAS PARA ESTUDAR**

Developer.mozilla.org -> guias e referencias de javascript da empresa que o criou

Ecma-> ecmainternational.org -> ECMA262( referencias direto da ecma para javascript)

**DECLARAR VARIAVEIS**

Var nome\_da variável = valor;

Ou

Let nome\_da\_variavel = valor;

**PODEMOS** ou **NÃO** usar ; no JS

As regras para nome são as mesmas das outras linguagens

Podem começar com $ ou \_

São sensitive case

Não é preciso declarar o tipo da variável

**TIPOS PRIMITIVOS**

* Number
  + Infinity
  + NaN
* String
* Boolean
* Null
* Undefined
* Object
  + array
* function

O **tipo number** representa tanto o INT quanto o FLOAT

As variáveis trocam de tipo de acordo com o que recebem, assim como o PHP

**ASPAS**

`` (crase) (shift+acento agudo)

‘’ (aspas simples)

“” (aspas duplas)

Todas são utilizáveis

**COMANDOS**

typeof -> verifica qual o tipo da variável

alert ou window.alert(‘’) -> exibe um alerta na tela

confirm ou window.confirm(‘’) -> exibe uma confirmação na tela (botão OK e CANCELAR)

prompt ou window.prompt(‘’) -> exibe uma caixa de dialogo (caixa de texto)

O comando prompt SEMPRE retorna uma string, portanto, é preciso CONVERTER caso esteja trabalhando com números

Number.parseInt ou parseInt(n) ->converte string para number int

Number.parseFloat ou parseFloat(n) -> converte string para number float

Number(n) -> converte string para o number que o JS detectar que foi digitado (ou int ou float)

String(n)-> converte number para string

n.toString->converte a variável para string

s.length -> RETORNA quantos caracteres a string tem

s.toUpperCase() -> torna a string toda ‘MAIUSCULA’

s.toLowerCase()-> torna a string toda ‘minúscula’

document.write(‘’) -> escreve no documento (pagina)

é possível usar TAGs HTML dentro dele

n.toFixed(qnt\_casas\_decimais) -> declara quantas casas decimais quer em um numero

var n = 2.5

n.toFixed(2) -> 2.50

n.replace(‘caracter\_substituido’,’caracter\_substituto’)

var n = 2.5

n.replace(‘.’,’,’) -> 2,5 //substituiu ‘ponto’ por ‘virgula’

n.toLocaleString(‘pt-br’,{style: ‘currency’, currency: ’BRL’ }) -> converter um valor numérico para valor em real

é possível trocar o BRL por outras moedas

console.log(‘’) -> exibir mensagens no console (node)

var.innerHTML = ’’ -> altera oq tem dentro da estrutura html desse elemento

podendo inserir tags html dentro junto ao texto

var.innerText = ‘’ -> altera somente o texto desse elemento

**CONCATENAÇÃO**

**Quando o JS encontra STRING+STRING ele concatena**

**Quando encontra NUMBER+NUMBER ele soma**

**TIPOS DE CONCATENAÇÃO**

Var s = ‘javascript’

‘eu estou aprendendo s’ //não faz interpolação (concatenação)

‘eu estou aprendendo’ + s //usa concatenação

`eu estou aprendendo ${s}` //usa template string

(É NECESSARIO USAR CRASE INVES DE ASPAS)

O template string é um jeito mais fácil de fazer concatenação

É possível usar funções dentro do template

${s.length}

**OPERADORES**

**ARITMÉTICOS**

Usados para fazer calculo

+ soma

- subtração

\* multiplicação

/ divisão

% modulo (resto)

\*\* potenciação

**ATRIBUIÇÃO**

**Atribuição simples**

= recebe

**Auto-atribuição**

Var n = 3

n += 4

Funciona pra todas as operações

n/= 2

mesma coisa que(n = n/2)

...

**Incremento**

I++ (soma 1)

i— (subtrai 1)

ou

++i

--I

\*faz diferença

**RELACIONAIS**

**>**maior

**<** menor

**>=** maior ou igual

**<=** menor ou igual

**==** igual

**!=** diferente

**===** identidade (testa se dois valores são totalmente idênticos, incluindo **o TIPO**)

**Para o JS**

**5==’5’ é TRUE**

**Porém, 5===’5’ é FALSE**

**!===** extritamente diferente (se são diferentes incluindo **o TIPO**)

Todos retornam valores BOOLEAN (TRUE ou FALSE)

**LOGICOS**

**!** negação

**&& ‘e’ (conjunção)**

**||** ‘ou’ (disjunção) **SÃO PROCESSADOS NESSA ORDEM**

**TERNÁRIOS**

**?**

**:**

Teste ? true : false

Teste -> teste logico

True -> oq acontece se for verdadeiro

False-> oq acontece se for falso

**Exemplo1**

a>7?'aprovado':'reprovado'

‘reprovado’

**Exemplo2**

b = 5%2 ==1.5 ? 'foi':'n deu'

‘foi’

B receberá ‘foi’ se 5%2 for igual a 1.5, se não for, receberá ‘n deu’

**Document Object Model (DOM)**

Somente usado dentro do navegador

É organizado em arvores

Começando na raiz chamado **window**

Dentro dele há vários objetos

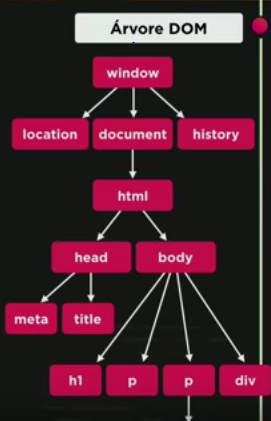
Por exemplo

Location -> url da pagina

Document -> html (parent)

* head(child)
  + meta
  + title
  + outras coisas
* body(child)
  + tudo que tem no body

History -> de onde veio, pra onde vai



**SELECIONANDO ELEMENTOS**

\*todos que estão no plural tem que usar ‘[]’ para identificar uma tag especifica que aparece varias vezes

**POR MARCA(tag)**

getElementsbyTag(‘tag’)

**POR ID**

GetElementbyId(‘id’)

**POR NOME**

getElementsbyName(‘name’)  
**POR CLASSE**

getElementsbyClass(‘class’)

**POR SELETOR**

document.querySelectro(‘div#id’)

document.querySelectro(‘div.class’)

**ALTERAR STYLE**

Var d = document.getElementbyId(‘id’)

d.style.parte\_do\_CSS = ‘valor’

**EVENTOS DOM**

Tudo que pode acontecer com um elemento

Nem todo elemento é compatível com todos os eventos

Mdn events reference (bibliografia da mozila para todos os eventos js)

**Eventos de mouse**

**Mouseenter**

Mouse entrou no elemento

Onmouseenter=’funcao()’

**Mousemove**

Mouse se movendo dentro do elemento

**Mousedown**

Mouse clicou e segurou no elemento

**Mouseup**

Mouse soltou o click no elemento

**Click**

Mouse fez o click completo no elemento

Onclick=’função’

**Mouseout**

Mouse saiu do elemento

Onmouseout=’funcao()’

**FUNÇÕES**

Function ação(param){

}

Existem funções anônimas

Function(){

}

**LISTENERS**

Variável.addEventListener(‘evento’,função)

Área.addEventListener(‘mouseenter’,entrar)

\*sem parênteses na função

\*sem ‘on’ no evento

**Data e hora**

var agora = new Date()

var hora = agora.getHours()

o objeto date pode ser usado para pegar horas e datas

o getDay retorna um numero correspondente ao dia

0- Domingo

1- segunda

2- terça

3- quarta

4- quinta

5- sexta

6- sabado

**Variaveis compostas**

Armazenam mais de um valor – ARRAYs

**Declaração**

Var a =[]

Ou

Var a = [a,b,c]

Arrays são vetores, com vários elementos, identificados por um índice chave, e cada elemento tem o seu conteúdo

**Aceita valores heterogêneos, de tipos diferentes em cada elemento**

a.push(2) -> insere um elemento no array

a.length -> mostra quantos elementos tem o array

(não usa parênteses, pois é um atributo, não um metodo)

a.sort() -> ordena em ordem crescente os elementos do vetor

a.indexOf(b) -> retorna qual o índice desse elemento

(se não houver esse valor no vetor, retornará -1)

**laço especial para vetores**

for(var i in valores){

    console.log(valores[i])

}

N é preciso declarar ‘i’ como ‘0’

**FUNÇÕES 2 (parâmetros)**

São ações executadas assim que são chamadas ou em decorrência de algum evento

Uma função pode receber parâmetros e retornam algum resultado

**Function** ação(param){

Return res

}

ação(5)

**é possível definir valores padrão caso algum parâmetro não seja mandado**

function soma(n1=0,n2=0){

    return n1+n2

}

console.log(soma(5))

**no caso, retornará 5 pois o segundo parâmetro não foi enviado**

**variáveis também podem receber funções**

let v = function(){

    console.log('teste')

}

v()

let a = function(x){

    return x\*2

}

console.log(a(2))

**‘v’ retornara ‘teste’**

**‘a’ retornará 4**

**Funções podem chamar funções (inclusive elas mesmas)**

//RECURSIVIDADE

function fatorial(n){

    if(n==1){

        return 1

    }

    else{

        return n\* fatorial(n-1)

    }

}

**OBJETOS**

Declaração

Let amigo = {}

Nos objetos é possível nomear o índice do ‘vetor’

Let amigo = {

Nome: ‘jose’,

Sexo:’m’,

Peso:80,

Engordar(p){}

}

**Objetos podem guardar funções**