**Curso JavaScript**

**Comentários:**

* // - uma linha
* /\* \*/ - mais de uma linha

**Identificadores:**

* Podem começar com letra, $ ou \_
* Não podem começar com números, conter espaços e ser palavras reservadas
* É possível utilizar letras, números, acentos, símbolos

**Data Types:**

* Number
  + Infinity, NaN
* String
* Boolean
* Null
* Undefined
* Object
  + Array
* Function

Se necessário utilizar o typeof para saber qual o tipo

**String > Número:**

* Number.parseInt(n)
* Number.parseFloat(n)
* Number(n)

**Número > String:**

* String(n)
* n.toString()

**Fomatando string:**

* `${n}` // usa template string, necessário utilizar crase
* + n // usa concatenação
* n.lenght
* n.toUpperCase()
* n.toLowerCase()

**Formatando números:**

* n.toFixed()
* n.toFixed().replace(‘.’, ‘,’)
* n.to.LocaleString(‘pt-BR’, {style: ‘currency’, currency: ‘BRL’’})

**Operadores:**

* aritiméticos
  + +
  + –
  + \*
  + /
  + % //resto da divisão inteira
  + \*\* //ao quadrado
* Atribuição
  + var a = 6 \* 2 / d
  + n += 4
  + n ++ (existe o pré e pós incremento ++n ou n++)
* relacionais
  + >
  + <
  + >=
  + <=
  + ==
  + !=
  + Identidade: ===
* Lógicos
  + ! (negação)
  + && (conjunção)
  + || (disjunção)
* Ternário
  + ?
  + :
  + Ex: teste ? true : false
* Typeof (não será visto nesse curso)
* Precedência: () -> \*\* -> \* / % -> + -
* Precedência: relacionais
* Precedência: ! -> && -> ||

**DOM – Document Object Model**

* **Árvore DOM**
  + window
    - location
    - document
      * html
        + head

meta, title, etc

* + - * + body

h1, p, div, etc

strong

* + - history
* **Métodos de acesso:**
  + Marca: getElementsByTagName()
  + ID: getElementByID()
  + Nome: getElementsByName()
  + Classe: getElementsByClassName()
  + Seletor, mais prático para alterações rápidas
    - querySelector()
    - querySelectorAll()
* **Eventos**
  + Javascript dom events list
  + onclick, mouseenter, mouseout
* **Funções**
  + Serão executadas apenas quando o evento for executado
  + Linhas = bloco
  + function ação(param) {bloco}

**Condições:**

* **Tipos de condição:**
  + Condição **simples**:
    - if (condição) {true}
  + Condição **composta**:
    - if (condição) {true} else {false}
  + Condições **aninhadas**: fazer condição dentro de outra condição
    - if (cond1) {bloco1} else if (cond2) {bloco2} else {bloco3}}
  + Condição **Múltipla**: tem possibilidade de valores múltiplos
    - switch (expressão) {  
      caso valor 1: break  
      caso valor 2: break

caso valor 3: break

caso valor 4: break  
}

**Repetições:**

* Estrutura de repetição com teste lógico no **início**
  + while (condição){}
* Estrutura de repetição com teste lógico no **final**
  + do {} while (condição)
* Estrutura de repetição com **variável de controle**
  + for (inicio; teste; incr){}

**Variáveis Compostas:**

Devem ser capazes de armazenar vários valores em uma mesma estrutura.

* var a = []
  + var a = array (vetor)
  + [] = elementos
  + 0,1,2 = índices
* Um array é uma variável que tem vários elementos. Cada elemento teu seu valor e sua chave de identificação
* Exemplo:
  + let num = [5, 8, 4]
    - índices: 0, 1, 2
* num[3] = 6 = insere um novo elemento no índice escolhido
* num.push(7) = insere um número
* num.length
* num.sort() = coloca os elementos em ordem crescente
* for(let pos in num){} = lê-se “para cada pos em num”. Funciona apenas para arrays e objetos
* num.indexOf() = procura no vetor onde está o valor. Se não tiver o valor será retornado valor -1

**Funções:**

São ações executadas assim que são chamadas ou em decorrência de algum evento. São heterogêneas, podem guardar valores de diferentes tipos.

Uma função pode receber parâmetros e retornar um resultado.

* Nem toda função usa parâmetro, nem toda função usa retornos.
* Caso existam mais de um parâmetro e eu não passar algum, ele ficará como “undefined” e o retorno será NaN.
  + Nesse caso posso passar no parâmetro mesmo o valor
    - **Texto

      O conteúdo gerado por IA pode estar incorreto.**Interface gráfica do usuário

      O conteúdo gerado por IA pode estar incorreto.function soma(*n1*=0, *n2*){}

**Objetos:**

Para declarar uma variável é possível declarar com {}. É possível inserir diretamente um identificador.

São variáveis que guardam valores (atributos) e podem guardar funcionalidades (métodos).

* let amigo = {nome:’José’, sexo:’M’, peso:85.4, engordar(p){}}
  + engordar(p){} = cria um parâmetro “engordar”, que guarda uma função (p){}

**OUTROS:**

* .getHours()
* .getDay()
* .getFullYear()
* **Imagens:**
  + .setAttribute('id', 'foto')
    - .setAttribute('src', 'homem.jpg')
  + res.appendChild(img)
* **Radio button:**
  + Tem que ser pego pelo name, todos os rádios precisam ter o mesmo name - var fsex = document.getElementsByName('sexo')
  + Para selecionar o primeiro, segundo, etc, radio é necessário chamar por colocação - if (fsex[0].checked)
  + Lembrar de colocar *checked em um dos inputs*
* **Elementos HTML**
  + Ao pegar um elemento DOM usando getElementById(), estamos obtendo uma REFERÊNCIA ao elemento HTML, não ao seu VALOR diretamente
  + Caso eu use **.value** retornarei o valor atual como STRING
  + Caso eu use **.valueOf** retornarei o valor PRIMITIVO do objeto
  + .value:
    - Sem usar para elementos HTML
    - Usar para obter o conteúdo do campo
  + .valueOf
    - Se já convertido, não é necessário utilizar
* isNaN(n)