JS CHEATSHEET II

FTAMBERLINI.DEV.BR
FTAMBERLINIDEV
@FTAMBERLINI

STRINGS

```
#ATRIBUTOS E MÉTODOS
let str = "abcdefghij";
length      //retorna o tamanho da string = 10
str.length  // 10
charAt() // retorna o caractere em uma posição específica
str.charAt(2) // 'c'
charCodeAt() // retorna o código Unicode do caractere na posição indicada
str.charCodeAt(0) // 97
concat() // concatena duas strings
str.concat('XYZ') // 'abcdefghijXYZ'
includes() // verifica se a string contém um texto
str.includes('def') // true
endsWith() // verifica se a string termina com um texto
str.endsWith('hij') // true
startsWith() // verifica se a string começa com um texto
str.startsWith('abc') // true
indexOf() // retorna o índice da primeira ocorrência
str.indexOf('d') // 3
lastIndexOf() // retorna o índice da última ocorrência
str.lastIndexOf('h') // 7
slice() // retorna parte da string entre índices
str.slice(2, 6) // 'cdef'
substring() // retorna parte da string entre índices
str.substring(2, 6) // 'cdef'
toUpperCase() // converte em letras maiúsculas
str.toUpperCase() // 'ABCDEFGHIJ'
toLowerCase() // converte em letras minúsculas
str.toLowerCase() // 'abcdefghij'
trim() // remove espaços no início e no fim
'  abc  '.trim() // 'abc
```

MATH

```
#MÉTODOS
let pi = Math.PI;  //3.141592653589793
Math.round(4.4)     // = 4 arrendondado
Math.round(4.5)     // = 5 arrendondado
Math.pow(2,8)       // = 256 → 2 elevado a 8
Math.sqrt(49)       // = 7 → raiz quadrada
Math.abs(-3.14)     // = 3.14 → absoluto
Math.ceil(3.14)     // = 4 → teto (primeiro inteiro sup.)
Math.floor(3.14)    // = 3 → piso (primeiro inteiro inf.)
pi.toFixed(0)       // = 3 → zero casa decimal
pi.toFixed(2)       // = 3.14 → duas casas decimais
Math.min(0,3,-2,2) // = -2 → menor valor
Math.max(0,3,-2,2) // =  3 → maior valor
Math.sin(0)         // = 0 → seno de 0
Math.cos(pi)        // = -1 → cosseno de pi
Math.log(1)         // = 0 → logaritmo natural
Math.exp(1)         // = 2.7182... (e)
Math.random()       // número randômico entre 0 e 1
Math.floor(Math.random()*5) + 1 //n. aletório entre 1 e 5
```

ESTRUTURA DE REPETIÇÃO

```
# FOR LOOP
//Imprime Tabuada de 9
for (let i = 0; i < 10; i++) {
    console.log(`Tabuada 9 x ${i} = ${i*9}`);
}
//Somando os elementos de um array
let a = [0,1,2,3,4,5]
let soma = 0;
for (let i = 0; i < a.length; i++){
   soma += a[i];
}
console.log(`O somatório é ${soma}`);
//Percorrendo um objeto iterável
const numeros = [10, 20, 30];
for (const n of numeros){
    console.log(n)
}
# WHILE LOOP
//Imprime os 12 primeiros elemento de uma Seq. Fibonacci
let n1=0,n2=1,contador=1,proximo=0;
let fibonacci = n1 + " " + n2
while (contador <= 10) {
    proximo=n1+n2;
    fibonacci+=" " + proximo;
    n1=n2;
    n2=proximo;
    contador++;
}
console.log(fibonacci)

# DO WHILE LOOP
//O do..while sempre executa o bloco pelo menos uma vez
let contador=1;
do {
    console.log("Número:", contador)
    contador++
} while (contador <= 10);

# COMANDOS DE SAÍDA DO LOOP
//O comando continue pula a iteração atual e vai direto
//para a próxima
for (let i = 1; i <= 5; i++) {
  if (i === 3) {
    continue; // Pula o número 3
  }
  console.log(i);
}

//Quando o JavaScript encontra um break, ele sai do laço
//imediatamente, mesmo que ainda existam iterações
//pendentes.
for (let i = 1; i <= 10; i++) {
  if (i === 5) {
    break; // Sai do loop quando i for 5
  }
  console.log(i);
}
```

OPERADORES

```
#ARITMÉTICOS
a = b             //operador de atribuição
a = b + c - d     //operador de adição e subtração
a = b * (c / d )  //operador de multiplicação e divisão
a = 2**3          //operador de exponenciação (2**3=8)
a = 100 % 48      //operador módulo (resto divisão inteira)
                  //100 / 48 = 2 + 4
                  //dividendo / divisor = quociente + resto
a++               //incremento unitário a=a+1;
a--               //decremento unitário a=a-1;
a+=b              //a = a + b (pode ser - * %)
a * (b + c)       //agrupamento

#COMPARAÇÃO & LÓGICOS
a == b            //operador de igualdade
a != b            //operador de diferença
a === b           //operador de igualdade estrita
a !== b           //operador de diferença estrita
a > b  a >= b     //operador (maior/maior ou igual) que
a < b  a <= b     //operador (menor/menor ou igual) que
a && b            //operador E (AND)
a || b            //operador OU (OR)
!a                //operador NÃO (NOT)
typeof a          //operador de tipo (boolean, number...)

#BITWISE
&    AND          //5 & 1 (0101 & 0001)     1 (1)
|    OR           //5 | 1 (0101 | 0001)     5 (101)
~    NOT          //~ 5 (~0101)             10 (1010)
^    XOR          //5 ^ 1 (0101 ^ 0001)     4 (100)
<<   LEFT SHIFT   //5 << 1 (0101 << 1)      10 (1010)
>>   RIGHT SHIFT  //5 >> 1 (0101 >> 1)      2 (10)
>>>  ZERO FILL    // 5 >>> 1 (0101 >>> 1)   2 (00000010)
     RIGHT SHIFT
```

ESTRUTURA DE CONTROLE (IF, SWITCH)

```
# IF
if ((idade >= 18) && (idade < 70 )) {
     voto="OBRIGATORIO"
} else if (idade >= 16){
     voto="FACULTATIVO"
} else {
     voto="PROIBIDO"
}
# SWITCH
switch (new Date().getDay()){
case 6:
     diaSemana = "Sábado";
     break;
case 0:
     diaSemana = "Domingo";
     break;
default:
     diaSemana = "Dia Útil";
}
# OPERADOR TERNÁRIO (?:)
codição ? valorSeVerdadeiro : valorSeFalso
```

JS CHEATSHEET II

FTAMBERLINI.DEV.BR
FTAMBERLINIDEV
@FTAMBERLINI

FUNÇÕES

```
#FUNÇÃO DECLARADA
function soma(a, b) {
   return a + b;
}
x = soma(a, b);
#FUNÇÃO ANÔNIMA
const soma = function(a, b) {
   return a + b;
}
#FUNÇÃO CURTA (ARROW FUNCTION)
const soma = (a,b) => a + b;
```

ARRAY

```
#ATRIBUTOS E MÉTODOS
let arr = [1, 2, 3, 4, 5];
arr[0] // 1 arr[4] //5 arr[5] //undefined
arr.length // 5
push() // adiciona um ou mais elementos ao final
arr.push(6) // retorna 6 (novo comprimento) → arr =
[1,2,3,4,5,6]
pop() // remove o último elemento do array
arr.pop() // retorna 6 → arr = [1,2,3,4,5]
unshift() // adiciona um ou mais elementos no início
arr.unshift(0) // retorna 6 → arr = [0,1,2,3,4,5]
shift() // remove o primeiro elemento
arr.shift() // retorna 0 → arr = [1,2,3,4,5]
concat() // combina dois arrays
arr.concat([6,7]) // [1,2,3,4,5,6,7]
join() // junta os elementos em uma string
arr.join('-') // '1-2-3-4-5'
indexOf() // retorna o índice da primeira ocorrência
arr.indexOf(3) // 2
reverse() // inverte a ordem dos elementos
arr.reverse() // [5,4,3,2,1]
sort() // ordena os elementos (como strings por padrão)
[3,1,2].sort() // [1,2,3]
toString() // converte o array em string
arr.toString() // '1,2,3,4,5'
fill() // preenche o array com um valor
arr.fill(0,2,4) // [1,2,0,0,5]
filter() // cria um novo array com os elementos que
passam no teste
arr.filter(x => x % 2 === 0) // [2,4]
map() // cria um novo array aplicando uma função
arr.map(x => x * 2) // [2,4,6,8,10]
forEach() // executa uma função para cada elemento (sem
retornar)
arr.forEach(x => console.log(x)) // imprime 1 2 3 4 5
reduce() // reduz o array a um único valor (acumulador)
arr.reduce((soma, x) => soma + x, 0) // 15
find() // retorna o primeiro elemento que satisfaz uma
condição
arr.find(x => x > 3) // 4
```

OBJETOS E CLASSES

```
#OBJETO
const pessoa = {              //nome do objeto
   nome:"Pedro",              //atributos
   sobrenome:'da Silva',
   peso:75,
   altura:1.80,
   calcularIMC(){             //método
      return this.peso/this.altura**2
   }
}
//Acessando o atributo do objeto
console.log("Meu nome é " + pessoa.nome);
//Invocando o método do objeto
console.log(`O meu IMC é ${pessoa.calcularIMC()}`);
//Criando um novo atributo
pessoa.idade=20;

#CLASSE (SUPERCLASSE)
class Pessoa {                //definição
   constructor(nome, sobrenome, peso, altura){
      this.nome = nome;
      this.sobrenome = sobrenome;
      this.altura = altura;
      this.#peso = peso       //atributo privado
   }
   //método público para calcular o IMC
   calcularIMC(){
      return this.peso/this.altura**2
   }
   //método público para mostrar nome completo
   nomeCompleto(){
      return `${this.nome} ${this.sobrenome}`;
   }
}
//Criando um objeto a partir da classe
const pessoa = new Pessoa("Ana","Silva",1.70,65);
//Invocando o método do objeto pessoa
console.log(`O meu IMC é ${pessoa.calcularIMC()}`);
//Acessando um atributo do objeto pessoa
console.log("Minha altura é " + pessoa.altura);

#CLASSE COM HERANÇA (SUBCLASSE)
class Aluno extends Pessoa {
   constructor(nome, sobrenome, peso, altura, matricula){
      //chama construtor da classe pai
      super(nome, sobrenome, peso, altura, matricula);
      this.matricula = matricula;
   }
}
//Criando um objeto a partir da classe
const aluno = new Aluno("Ana","Silva",1.70,65,2025020001);
//Invocando o método herdado da classe pai
console.log(`O meu IMC é ${pessoa.calcularIMC()}`);
//Acessando um atributo do objeto aluno
console.log("Minha matricula é " + aluno.matricula);
```

TIPOS DE DADOS (DATA TYPES)

```
#DECLARAÇÃO DE VARIÁVEIS
var a = 1     //escopo global/função
              //permite redeclaração no mesmo bloco
              //é içada (hoisted)
              //evite usar em código moderno

let b = 2     //escopo de bloco e não pode ser redeclarada
              //pode ser reatribuída

const c = 3   //escopo de bloco e não pode ser redeclarada
              //NÃO pode ser reatribuída
            //Atributos de objeto const podem ser alterados

#VALORES E TIPOS
false, true                 //boolean
18, 3.14, NaN               //number (int, dec, not number)
0b10000, 0o000, 0x0010      //number (binary, octa, hexa)
"Maria" , 'Pedro'           //string
undefined, null, Infinity   //special
let a = [1,2,3]             //array
let pessoa={
        nome: "Maria",
        idade: 18}          //object
const dobro = (x) => x * 2;//function
```

ESTRUTURA BÁSICA

```
#SCRIPT NA PÁGINA
<script type="text/javascript">
    console.log("Olá, mundo!");
</script>

#SCRIPT ARQUIVO EXTERNO
<script src="script.js" ></script>

#COMENTÁRIOS
/* Comentário de
   múltiplas linhas */

// Comentário de uma única linha

#SAÍDAS (OUTPUT)
console.log("Olá!")     //escrita no console do navegador
document.write("Olá!") //escrita no arquivo html
alert("Olá!")           //janela de alerta
confirm("Confirma?")    //caixa de diálogo (sim/não)
prompt("Idade?","0")    //caixa de diálogo de entrada
                        //2° argumento é o valor inicial
```