**Criando elementos html pelo JS**

**const** **div** = **document**.**createElement(“div”)**

**div.innerHTML = “ //Conteúdo ”**

**div.setAttribute(“class”, “//nome”)   
container.appendChild(div) //Class dada no html onde se quer app o conteúdo**

**Métodos JavaScript**

**Métodos de Manipulação de Arrays:**

array. **push** (“x”) // Insere um elemento “x” no final

array. **pop** (“x”) // Retira um elemento “x” do final

array. **unshift** (“x”) // Insere um elemento “x” no início

array. **shift** (“x”) // Retira um elemento “x” do início

array. **map** ((x , i) => { // App uma função a cada elemento do array e gera uma novo array com os elementos operados (i = índice da posição do elemento “x”)

//Função

})

array. **indexOf** (“x”) // Retorna à posição do elemento. Se o elemento não existir retorna "-1".

**Métodos De Data:**

getDate() // Pega o Dia do Mês

getDay() // Pega o Dia da semana (número)

getFullYear() // Pega o Ano com 4 dígitos

getHours() // Pega as Horas

getMinutes() // Pega os Minutos

getSeconds() // Pega os Segundos

getMilliseconds() // Pega os Milissegundos

getTime() // Timestamp (milissegundos desde 1 de janeiro de 1970, 00:00:00 UTC)

Date.now() // Timestamp (milissegundos desde 1 de janeiro de 1970, 00:00:00 UTC)

getTimezoneOffset() // Timezone da localidade

**Redirecionamento de Páginas**

window.location = “aula01.html” //redireciona a página para a aula01

window.location = “https://www.google.com.br” //redireciona a página do google

**Métodos Location:**

window.location.**replace**(“https://www.google.com.br”) //substitui a URL corrente pela URL do google a excluindo do histórico (Impossibilitando a volta)

window.location.**assign**(“https://www.google.com.br”) //substitui a URL corrente pela URL do google, porém sem excluir do histórico (Possibilitando a volta)

window.location.**reload**( ) // Recarrega a URL corrente

**Window History:**

window.history.**back**( ) //Volta para a URL anterior gravada no histórico

window.history.**forward**( ) //Avança para a próxima URL gravada no histórico

window.history.**go**(1) //Avança ou volta para a URL indicada por um number no parâmetro

**Caixas de diálogo**

**alert**(“Olá como está?”) //Mostra uma caixa de msg com o texto passado

**confirm**(“Deseja continuar?”) // Mostra uma caixa de msg com o texto passado e retorna um booleano com ‘true’ ou ‘false’

**prompt**(“Digite seu nome.”) // Mostra uma caixa de msg com o texto passado e retorna uma string ou um null

**Conteúdo para Impressão**

window.**print**( ) // vai gerar um print da tela (como um todo) para impressão

//Para imprimir apenas o conteúdo importante, vale criar um media query separado apenas para impressão

**Objetos Literais**

const computador = {    //As {} já indicam que é um objeto

    cpu:"i9",

    ram:"64gb",         //As propriedades devem ser separadas por vírgula

    hd:"2tb"

}

const computadores = [      // Array de Objetos Literais

    {

        cpu:"i9",

        ram:"64gb",

        hd:"2tb"

    },

    {

        cpu:"i7",

        ram:"32gb",

        hd:"2tb"

    },

    {

        cpu:"i5",

        ram:"16gb",

        hd:"1tb"

    }

]

const computador = {

    cpu:"i9",

    ram:"64gb",

    hd:"2tb",

    info:function(){ //Passando métodos

        console.log(`CPU: ${this.cpu}`)

        console.log(`RAM: ${this.ram}`)

        console.log(`HD: ${this.hd}`)

    }

}

computador.info() //Chama o método

//Para Adicionar propriedades temos duas formas:

    computador["monitor"]="22pol"

    computador.placaVideo="rtx"

**Clonando um Objeto Literal:**

//Clonando um objeto literal

const computador = {

    cpu:"i9",

    ram:"64gb",

    hd:"2tb",

    info:function(){

        console.log(`CPU: ${this.cpu}`)

        console.log(`RAM: ${this.ram}`)

        console.log(`HD: ${this.hd}`)

    }

}

const c1 = Object.assign({}, computador)    //C1 passa a ser um objeto exatamente igual a computador

//Mesclando vários Objetos em um só

const o1={obj1: "1"}

const o2={obj2: "2"}

const o3={obj3: "3"}

const o4 = Object.assign(o1, o2, o3)    //o4 será a junção de o1, o2 e o3

**Módulos**

Resumidamente os **módulos em JavaScript** são uma forma de compartilhar informações entre arquivos, dessa forma, todos os arquivos que realizarem a importação desse **módulo** poderão utilizar e usufruir de suas funções ou informações que foram expostas. Dessa maneira, evitamos ter que ficar carregando N arquivos .

**Rodando no Servidor:**

//Informar no arquivo HTML o 'type="module"'

    <script src="modulo01.js" type="module"></script>

**//Arquivo.js 2**

const cursos = []

const carros = []

export {cursos, carros} //Exporta os Arrays para o arquivo.js 1

**//Arquivo.js 1**

import { cursos } from "./modulo02.js";//Recebe os Arrays do Arquivo.js 2

import { carros } from "./modulo02.js";

**Módulo em Classes:**

**Exemplo:**

**//Arquivo.js 3**

class Cursos{

    static cursos = ["JavaScript","HTML","CSS"] //'Static' permite não precisar instanciar a classe para poder utilizá-la

    constructor(){

}

    static getAllCursos=()=>{   //Método que retorna o Array cursos

        return this.cursos

    }

    static getCurso=(i\_curso)=>{    //Método que retorna o curso do índice passado no parâmetro

        return this.cursos[i\_curso]

    }

    static addCurso=(newCurso)=>{   //Método que add um curso novo ao Array cursos

        this.cursos.push(newCurso)

    }

    static RemoveCursos=()=>{       //Método que limpa todo o Array cursos

        this.cursos = []

    }

}

export default Cursos //'default' classifica como padrão e permite alterar o nome da classe no arquivo que irá importar (Não usa as chaves)

**//Arquivo.js 1**

//Como a classe está sendo exportada como ‘default’, podemos alterar o nome da classe 'cursos' para 'c'

import c from "./modulo03.js"

  //Chama os métodos

    c.RemoveCursos()

    c.addCurso("Photoshop")

    console.log(c.getCurso(0))

    console.log(c.getAllCursos())

**Funções**

**Função Symbol:**

//A função Symbol retorna um objeto do tipo Symbol que tem um identificador único. (Não se pode usar esse identificador no DOM)

const s1 = Symbol()

const s2 = Symbol()     //(s1==s2) Irá retornar 'false'

const s3 = Symbol.for("Carlos") //Cria um identificador no registro global

const s4 = Symbol.for("Carlos") //Esse identificador pode ser dado tbm a outro Symbol

                        //(s3==s2) Agora irá retornar 'True'

// 'Symbol.keyFor()' Retorna a string do identificador ou um undefined (trabalha apenas com os registros globais)

console.log(Symbol.keyFor(s1))     //Retornará 'undefined'

console.log(Symbol.keyFor(s3))     //Retornará 'Carlos'