**// Funções**

  array.push(element1, element2, ..., elementN);

  // Adiciona elementos ao final do array e retorna o novo comprimento do array.

  array.pop();

  // Remove o último elemento do array e o retorna.

  array.shift();

  // Remove o primeiro elemento do array e o retorna, ajustando os índices dos outros elementos.

  array.unshift(element1, element2, ..., elementN);

  // Adiciona elementos ao início do array e retorna o novo comprimento do array.

  array.concat(array2, array3, ..., arrayN);

  // Combina arrays existentes para criar um novo array.

  array.slice(start, end);

  // Retorna uma cópia superficial de uma parte do array, começando do índice 'start' até 'end - 1'.

  array.splice(start, deleteCount, item1, item2, ..., itemN);

  // Altera o conteúdo de um array removendo, substituindo ou adicionando elementos.

  array.forEach(callback(currentValue, index, array));

  // Executa uma função de retorno de chamada para cada elemento do array.

  array.map(callback(currentValue, index, array));

  // Cria um novo array com os resultados da aplicação de uma função a cada elemento do array.

  array.filter(callback(currentValue, index, array));

  // Cria um novo array com todos os elementos que passam no teste implementado pela função de retorno de chamada.

  array.reduce(callback(accumulator, currentValue, index, array), initialValue);

  // Aplica uma função de retorno de chamada a um acumulador e a cada elemento do array, resultando em um único valor.

  array.sort(compareFunction(a, b));

  // Ordena os elementos de um array (modifica o array) com base em uma função de comparação.

  array.reverse();

  // Inverte a ordem dos elementos em um array.

  // Propriedades

  array.length;

  // Retorna o número de elementos em um array.

  // Exemplo de utilização:

  const myArray = [1, 2, 3, 4, 5];

  myArray.push(6);

  myArray.pop();

  myArray.forEach(item => console.log(item));

  const doubledArray = myArray.map(item => item \* 2);

---------+

  // Métodos e Propriedades de Objetos

  objeto.método(parametros, parametros, ...);   // Descrição do que faz

  objeto.propriedade;                           // Descrição da propriedade

  // Exemplos:

  // Object.keys()

  const myObject = { a: 1, b: 2, c: 3 };

  const keys = Object.keys(myObject);  // Retorna ['a', 'b', 'c']

  // Object.values()

  const values = Object.values(myObject);  // Retorna [1, 2, 3]

  // Object.entries()

  const entries = Object.entries(myObject);  // Retorna [['a', 1], ['b', 2], ['c', 3]]

  // Object.assign()

  const target = {};

  const source = { x: 1, y: 2 };

  Object.assign(target, source);  // Copia propriedades de 'source' para 'target'

  // hasOwnProperty()

  const hasPropA = myObject.hasOwnProperty('a');  // Retorna true

  const hasPropD = myObject.hasOwnProperty('d');  // Retorna false

  // toString()

  const objString = myObject.toString();  // Retorna "[object Object]"

  // constructor

  const constructorFunc = myObject.constructor;  // Retorna a função construtora do objeto

--------++-----

  // Métodos e Propriedades de Strings

  string.método(parametros, parametros, ...);   // Descrição do que faz

  string.propriedade;                           // Descrição da propriedade

  // Exemplos:

  // length

  const myString = "Hello, world!";

  const length = myString.length;  // Retorna o comprimento da string (14)

  // charAt()

  const charAtIndex = myString.charAt(0);  // Retorna 'H'

  // concat()

  const concatenated = myString.concat(" Welcome");  // Retorna "Hello, world! Welcome"

  // indexOf()

  const indexOfComma = myString.indexOf(',');  // Retorna a posição da primeira vírgula (5)

  // slice()

  const sliced = myString.slice(7, 12);  // Retorna "world"

  // toUpperCase()

  const upperCaseString = myString.toUpperCase();  // Retorna "HELLO, WORLD!"

  // replace()

  const replacedString = myString.replace('world', 'universe');  // Retorna "Hello, universe!"

  // trim()

  const stringWithSpaces = "   Hello!   ";

  const trimmedString = stringWithSpaces.trim();  // Retorna "Hello!"

-------+++++

  // Métodos e Propriedades de Números

  numero.método(parametros, parametros, ...);   // Descrição do que faz

  numero.propriedade;                           // Descrição da propriedade

  // Exemplos:

  // toFixed()

  const myNumber = 123.456;

  const fixedNumber = myNumber.toFixed(2);  // Retorna "123.46"

  // toPrecision()

  const preciseNumber = myNumber.toPrecision(4);  // Retorna "123.5"

  // parseInt()

  const parsedInt = parseInt("42");  // Retorna 42

  // parseFloat()

  const parsedFloat = parseFloat("3.14");  // Retorna 3.14

  // Math.round()

  const rounded = Math.round(3.7);  // Retorna 4

  // Math.floor()

  const floored = Math.floor(5.9);  // Retorna 5

  // Math.ceil()

  const ceiling = Math.ceil(2.1);  // Retorna 3

  // Math.max()

  const maxNumber = Math.max(10, 20, 5);  // Retorna 20

  // Math.min()

  const minNumber = Math.min(10, 20, 5);  // Retorna 5

--------++

    // Declaração de Função

  function nomeDaFuncao(parametro1, parametro2, ...) {

    // Descrição do que a função faz

    // ...

  }

  // Função com Parâmetros e Retorno

  function soma(a, b) {

    // Retorna a soma de dois números

    return a + b;

  }

  // Função Anônima (Atribuída a uma Variável)

  const multiplicacao = function(x, y) {

    // Retorna o produto de dois números

    return x \* y;

  };

  // Função de Flecha (Arrow Function)

  const divisao = (numerator, denominator) => {

    // Retorna o resultado da divisão

    if (denominator !== 0) {

      return numerator / denominator;

    } else {

      throw new Error("Divisão por zero não é permitida");

    }

  };

  // Função Auto-Invocada (Immediately Invoked Function Expression - IIFE)

  (function() {

    // Esta função é executada imediatamente após sua definição

    console.log("Esta é uma IIFE!");

  })();

  // Callback - Função Passada como Parâmetro

  function operacaoMatematica(a, b, callback) {

    return callback(a, b);

  }

  // Uso de Funções

  const resultadoSoma = soma(5, 3);  // Resultado: 8

  const resultadoMultiplicacao = multiplicacao(4, 6);  // Resultado: 24

  const resultadoDivisao = divisao(10, 2);  // Resultado: 5

  const resultadoOperacao = operacaoMatematica(7, 2, (x, y) => x - y);  // Resultado: 5

-------++---

  // Criação de uma Promise

  const minhaPromise = new Promise((resolve, reject) => {

    // Lógica assíncrona

    if (condicao) {

      resolve("Sucesso!"); // Ação concluída com sucesso

    } else {

      reject("Erro!");     // Ação falhou

    }

  });

  // Usando uma Promise

  minhaPromise

    .then(resultado => {

      // Ação bem-sucedida

      console.log(resultado);

    })

    .catch(erro => {

      // Ação com erro

      console.error(erro);

    });

  // Exemplo de Uso de Promises Encadeadas

  function obterDadosDaAPI(url) {

    return fetch(url)

      .then(response => response.json())

      .then(data => {

        // Processa os dados obtidos

        return data;

      })

      .catch(error => {

        console.error("Erro ao obter dados da API:", error);

        throw error;

      });

  }

  // Utilizando async/await com Promises

  async function main() {

    try {

      const dados = await obterDadosDaAPI("https://api.exemplo.com/dados");

      console.log("Dados obtidos:", dados);

    } catch (error) {

      console.error("Ocorreu um erro:", error);

    }

  }

  main();

-------++++---

  // Definindo uma Classe com Construtor

  class Pessoa {

    constructor(nome, idade) {

      this.nome = nome;

      this.idade = idade;

    }

    saudacao() {

      console.log(`Olá, meu nome é ${this.nome} e tenho ${this.idade} anos.`);

    }

  }

  // Instanciando um Objeto da Classe Pessoa

  const pessoa1 = new Pessoa("Alice", 25);

  pessoa1.saudacao(); // Saída: Olá, meu nome é Alice e tenho 25 anos.

  // Herança de Classe

  class Estudante extends Pessoa {

    constructor(nome, idade, curso) {

      super(nome, idade);

      this.curso = curso;

    }

    apresentacao() {

      console.log(`Oi, eu sou ${this.nome}, tenho ${this.idade} anos e estudo ${this.curso}.`);

    }

  }

  // Instanciando um Objeto da Classe Estudante

  const estudante1 = new Estudante("Bob", 20, "Ciência da Computação");

  estudante1.apresentacao(); // Saída: Oi, eu sou Bob, tenho 20 anos e estudo Ciência da Computação.

------++

  // Adicionando um Event Listener a um Elemento

  elemento.addEventListener(evento, callback);

  // Exemplo:

  const botao = document.getElementById("meuBotao");

  function cliqueNoBotao() {

    console.log("Botão foi clicado!");

  }

  botao.addEventListener("click", cliqueNoBotao);

  // Removendo um Event Listener de um Elemento

  elemento.removeEventListener(evento, callback);

  // Exemplo:

  botao.removeEventListener("click", cliqueNoBotao);

  // Disparando Eventos Personalizados

  const meuElemento = document.getElementById("meuElemento");

  function meuEventoCustomizado() {

    console.log("Meu evento customizado ocorreu!");

  }

  meuElemento.addEventListener("meuEvento", meuEventoCustomizado);

  // Para disparar o evento:

  const evento = new Event("meuEvento");

  meuElemento.dispatchEvent(evento);