JavaScript possui tipagem dinâmica, no caso ele define o tipo enquanto o programa está rodado, ao contrário de outras linguagens como C, C#, Java, onde o tipo é definido ao declarar uma variável.  
Tipos primitivos JavaScript -> Number, string, null, boolean, undefined

Importante é adicionar o script no HTML para localizar o arquivo, nesse caso na HEAD utilize <script src= “Local”></script>

Utilizar na declaração de variáveis var, let ou const, além disso é importante lembrar que seria Case sensitive, ou seja, JavaScript considera variáveis diferentes se utilizar maiúsculo e minúsculo.

var 🡪 Declara uma variável, opcionalmente, inicializando-a com um valor.

let 🡪 Declara uma variável local de escopo do bloco, opcionalmente, inicializando-a com um valor.

const 🡪 Declara uma constante de escopo de bloco, apenas de leitura.

O essencial é declarar a variável no estilo camelCase, começando com uma primeira palavra minúscula e caso necessite de uma segunda deixar junto e iniciar em maiúscula.

Caso seja necessário alterar os valores da variável, utilizar o Let, se não basta utilizar o const na declaração que funcionará sem problemas e ela não poderá ser alterada.

O tipo objeto seria para juntar informações de variáveis em um mesmo lugar.

let pessoa = {};// Definindo um objeto vazio

Para acessar um valor dentro de um objeto, basta adicionar o nome do objeto.nomeValor.  
Exemplo: pessoa.nome

Para definir o que terá dentro do objeto seria propriedade: O que irá dentro da propriedade (Valor).

Seria a combinação de chave e valor, a chave mais conhecida como identificado;

Exemplo:

let pessoa = {

nome: 'Guilherme',

idade: 20

};

Array 🡪 Conjunto de dados que pode ser acessado por um índice.  
Índice 🡪 Posição exata onde se encontra essa informação.

let familia = []; //Definindo um array vazio

Exemplo:

let familia = [26,45,50,17]; //Definindo um array

console.log(familia[0]);

Não precisa estar definido com apenas um tipo dentro dele, pode inserir tipos diferentes de valores.

Nomear funções deve pensar em um verbo + um substantivo por fins didáticos

Exemplo:

function resetaCor(){

};

Assim basta inserir os parâmetros se necessitar e para chamar a função para utilizar o nome com os parênteses.  
Caso a função precise retornar algo, basta utilizar o return.

Tipos de operadores

Operadores aritméticos(matemáticos)

Operadores de atribuição

Operadores de comparação

Operadores lógicos

Operadores BitWise

Operadores aritméticos

+, -, /, \*, %, \*\* -> Exponencial.

++ -> Incremento

-- -> Decremento

Operador de atribuição

= -> Atribui um valor a uma propriedade.

+=

-=

Operadores de igualdade

=== -> Compara o valor e o tipo são iguais

== -> Compara apenas se o valor é igual ignorando o tipo.

//Operador ternário

//Exemplo: Cliente possui mais de 100 reais, premium, se não normal.

let dinheiro = 100;

let tipo = dinheiro > 100 ? 'premium' : 'normal';

console.log(tipo);

Operadores lógicos

&& -> Retorna true se os dois operandos forem verdadeiros

|| -> Retorna true se um dos operandos for verdadeiro

! -> Retorna à negação do que estiver sendo comparado

Comparações não booleanas

Resultado de uma comparação lógica nem sempre será true ou false.

Falsy: Undefined, null, 0, false, ‘’, NaN

Truthy : Qualquer outro não citado acima.

False || true -> true

False || ‘Guilherme’ -> Guilherme

False || 1 -> 1

False || 1 || 3 -> 1 //Avalia apenas a primeira comparação

Exemplo:

var corPersonalizada = “Vermelho”

var corPadrao = “Azul”

var corPerfil = corPersonalizada || corPadrao

**Comandos**

Console.log(‘’) 🡪 Mostra algo no console do navegador.

Typeof Variável 🡪 É possível verificar o tipo primitivo de uma variável.

**Imersão DEV**

Variáveis -> Referencias para valores definidos que serão utilizados pelo computador.

Console.log -> Utilizado para mostrar algo no console

toFixed -> Utilizado para limitar as casas decimais

function nomeDaFunção(){

} -> Assim terá uma função criada para ser utilizada quando algo for clicado ou para chama-la durante a execução do código.

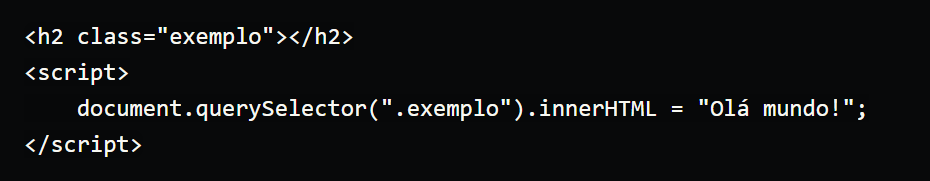
document.getElementById(“ID do elemento”).value -> Retorna o valor de um elemento do HTML de acordo com o ID.

parseInt() -> Transforma a string em inteiro

parseFloat() -> Transforma a string em um número real.

Utilizar o innerHTML quando precisar inserir uma informação dentro de uma tag selecionada no html.

querySelector() -> Retorna o primeiro elemento dentro do documento que foi especificado pelo seletor. O querySelector() aceita os atributos ids, classes e outros parametros de seleção.



getElementById() -> Retorna apenas um elemento selecionando ele pelo ID.