Javascript é uma linguagem interpretada, ou seja, ela está sendo rodada em tempo real pelo navegador. Código disponibilizado instantaneamente (não precisa ser transformado em outra coisa antes de ser enviado para o browser ou console).  
  
Linguagens compiladas (C++) primeiro você coda primeiro, depois passa pelo compilador para que o computador consiga entender o código e aí depois vem o output do que aconteceu com aquelas instruções anteriormente passadas.

JS é uma linguagem multiparadigmas: pode trabalhar com orientação a objeto, programação funcional ou programação estruturada.

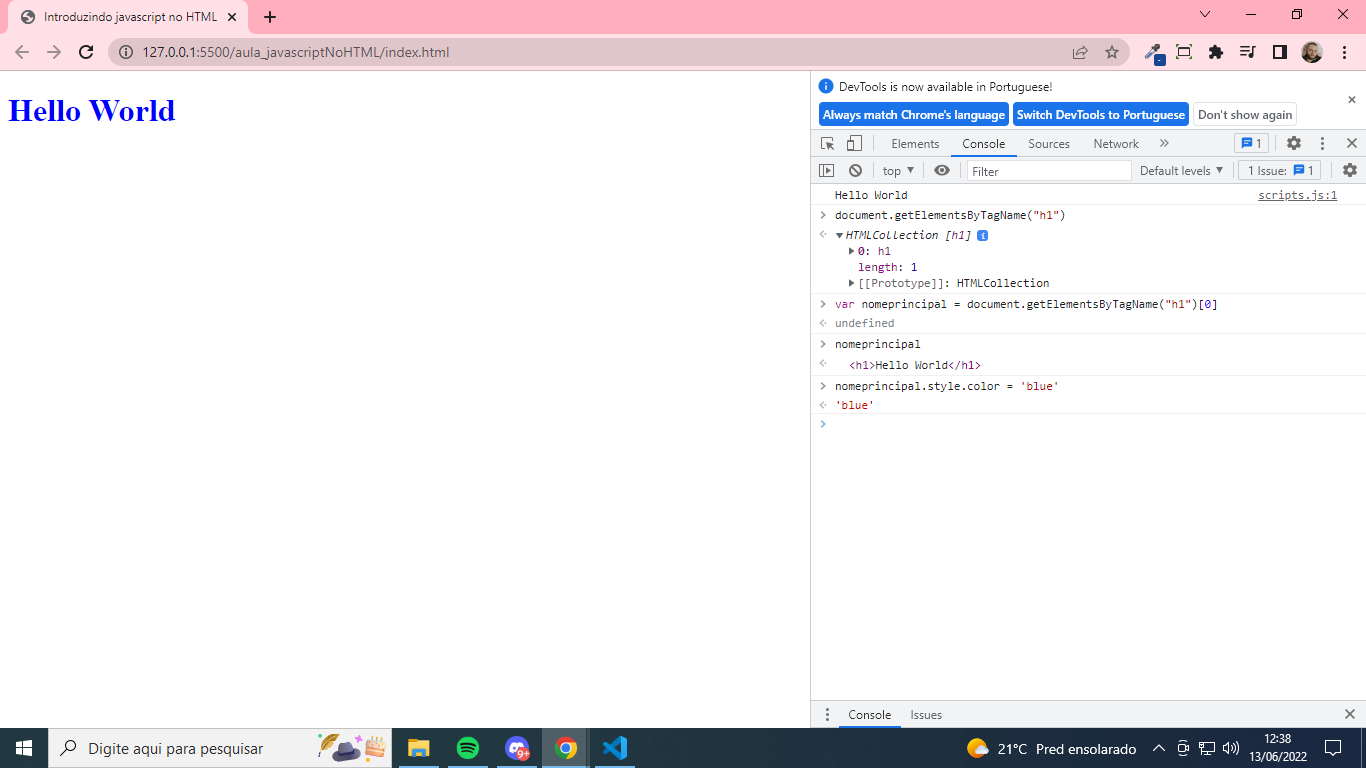
JS segue o padrão ECMAScript (conjunto de normas que vai dizer quais funcionalidades estão sendo lançadas naquela linguagem).

NOTAS   
- a declaração de “variáveis” constantes geralmente é feita com letras maiúsculas

ex: const PRECO = 2;

“variáveis” do tipo const não podem ser manipuladas, ou seja, seus valores são fixos…constantes!

funções - return: é utilizado quando o resultado da função precisa ser manipulado em alguma coisa (geralmente outra função)

Interagindo com elementos HTML e CSS por meio de códigos javascript no console do navegador. ↓  


Case Types (formas de nomear variáveis.

**Camel Case**: someAwesomeVar  
**Snake Case**: some\_awesome\_var  
**kebab Case**: some-awesome-var

**Pascal Case**: SomeAwesomeVar

**Upper Case Snake Case**: SOME\_AWESOME\_VAR (geralmente utilizado para variáveis do tipo CONST)

Atribuição de valores

Valores dinâmicos: Declarar com var e let

o tipo let tem escopo de bloco, ou seja, assim que ele sair do bloco a variável que foi declarada como let vai “parar de funcionar”

ex:

**var** a **=** **1**;

**var** b **=** **2**;

**if** (a **===** **1**) {

**var** a = **11**; //escopo global

**let** b = **22**; //escopo dentro do block if

**console**.**log**(a); //saída vai ser 11

**console**.**log**(b); //saída vai ser 22

}

**console**.**log**(a); //saída vai ser 11

**console**.**log**(b); //saída vai ser 2

Hoisting: atribuição de valor a uma variável antes de declarar ela(só funciona com o tipo var)

Redeclaração:

|  | **var** | **const** | **let** |
| --- | --- | --- | --- |
| **escopo** | global ou local | bloco | bloco |
| **redeclarar** | **sim** | **não** | **não** |
| **reatribuir** | **sim** | **não** | **sim** |
| **hoisting** | **sim** | **não** | **não** |

Javascript é uma linguagem de **tipagem dinâmica**, ou seja, antes de declarar um valor, você não especifica o **tipo** dele.

**Estrutura de dados** em *js*

Dois grandes grupos de dados: Primitivos e não-primitivos

**obs:tipos primitivos não possuem métodos dentro deles.**

**Primitivos**: números, strings(textos), boleanos, nulos(null) e indefinidos(undefined).

**Não-primitivos**: objetos e arrays(listas).

**alguns comandos**  
**typeof** - checar o tipo da variável

**Strings**

umaString**.concat**(outraString); - concatena a primeira string com a segunda

ex: let nome = “Bruno”

let sobrenome = “Henrique”

nome.concat(sobrenome);

saída: “BrunoHenrique”

umaString**.length** - retorna a quantidade de caracteres daquela string

**crase** facilita a utilização de strings com outros tipos de variávels

ex:

forma padrão

nomeCompleto = nome + “ ” + sobrenome;

saída: “Bruno Henrique”

com crase

nomeCompleto = **´${nome} ${sobrenome}´;**

saída: “Bruno Henrique”

algumaString**.split**(“”); - separa os argumentos da string. Retorna uma lista

algumaString**.includes**(“nome”); - verifica se a string possui a palavra nome. Retorna true ou false.

algumaString**.startsWith**(“B”); - verifica se o primeiro elemento da string é o caracter B. Retorna verdadeiro ou falso.

algumaString**.endsWith**(“?”); - verifica se o último elemento da string é o caracter ?. Retorna verdadeiro ou falso.

umNumero**.toString**(); - transforma uma variável do tipo número em uma string

Arrays

let umArray = []; - definindo um array vazio

umArray**.push**(3); - adicionando o número 3 no final do array

saída: 1 (saída foi 1 porque é o tamanho do array)

comando: umaArray;

saída: [3] (saída foi o elemento 3)

umArray.**unshift**(1); - adicionando o número 1 no começo do array

saída: [1, 3]

umArray**.pop**(); - retira o último elemento do array

saída: [1]

umArray**.shift**(); - retira o primeiro elemento do array

saída: []