**JAVASCRIPT**

(Dicas do Zero ao Avançado)

**Usando o Google Chrome para executar comandos Javascript:**

botão direito -> Inspecionar -> Console

**Injetar HTML:**

documet.body.innerHTML = 'Hello';

**Três Maneiras de criar Strings:**

‘...’ “...” `...`

**Escape characters:**

\’ -> para inserir um apóstrofo em uma string criada com ‘...’

\n -> para pular uma linha

**Função Matemática de arredondar:**

Math.round((2095 + 799)/100)

**Função Matemática de Gerar número randômico entre 0 e 1:**

Math.random();

**Emitir um Popup:**

alert ('some \n text');

**Concatenar String com Number em uma mesma expressão:**

alert (`Items (${1+1}): $${(2095 + 799)/100}`);

**Install VS Code Extension “Live Server”:**

Click on extension Button at the left bar and search for “Live Server” and then Install;

Click on Manage Button at the left bar and click “Settings”;

Search for “Live Server” and choose “Chrome” at “Custom Browser”

Return to the code and click with right-button on the code and choose “Open with Live Server”

**Indents on VS Code:**

**Alterar o espaçamento do TAB de 04 espaços para 02 espaços:**

Click on Manage Button at the left bar and click “Settings”;

Search for “tab” -> change tab size from 4 to 2

Return to the code and click on the “spaces: 4” in the botton line, then click in “indent using spaces” and change to “2”

**Wrap Lines: Quebrar as linhas do VS Code automaticamente:**

Click on Manage Button at the left bar and click “Settings”;

Search for “wrap” -> change “Controls how lines should wrap” to “on”

**Variables:**

let -> create a new variable. It is a reserved Word. (ex.: let variable1 = 3;)

**Rules:**

Can’t use special words (ex.: let);

Can’t start with a number;

Can’t use special characters, except $, \_

Use semi-colen (;) in the end of the instruction;

**Naming Conventions:**

camelCase -> cartQuantity (standard name convention used to named variables in JS)

PascalCase -> CartQuantity (Only one feature in JS uses this convention)

kebab-case -> cart-quantity (HTML e CSS and file names)

snake\_case -> cart\_quantity (used in other languages, not in Javascript)

**Outras formas:**

cartQuantity = cartQuantity + 1; -> cartQuantity +=1; -> cartQuantity++;

cartQuantity = cartQuantity + 2; -> cartQuantity += 2;

**Others Operators Shortcuts:**

+=2 -> variable = variable + 2;

-=2 -> variable = variable – 2;

\*=2 -> variable = variable \* 2;

/=2-> variable = variable / 2;

++ -> variable = variable + 1;

-- -> variable = variable – 1;

**Creating Variables:**

let – you should use always you need to change the value of the variable;

var – We don’t use ‘var’ in new JS code.

Const – you should use always you don’t need to change the value of the variable;

Typeof -> return the type of the variable in JS

**Booleans: true / false**

== -> transforma os dois dados no mesmo tipo antes de fazer a comparação. Usado para comparar number com string contendo número. Ex: (5 == ‘5.00’) -> retorna true

=== -> esse não altera o tipo de dado, portanto, se comparar o número 5 com a string ‘5’, retornará false. Ex: (5 === ‘5.00’) -> retorna false;

**Camparison Operators:**

( > < >= <= == === !== !===)

If (condition) { commands} else {commands};

If (condition) {commands} else if (condition) {commands} else {commands};

**Logical Operator:**

&& -> AND

|| -> OR

! -> NOT

If (0) -> retorna false. If (nº <> 0) -> retorna true;

**False Values:**

false, 0, ‘’, NaN, undefined, null (NaN -> not a number)

**Rock/Paper/Scissors Game:**

Rock beats scissors

Paper beats rock

Scissors beats paper

**Functions:**

Let us reuse code.

**Rules for named functions:**

Can’t use special words (ex.: function);

Can’t start with a number;

Can’t use special characters, except $, \_

**Functions can return:**

a value;

a calculation;

a variable.

Undefined (return;)

Ex.: return variable1;

Quando você termina uma função com o return; (undefined), ao encerrar a execução da função, ele retorna para o local onde a função foi chamada. Parecido com o GOSUB do BASIC.

**Como chamar uma função:**

nomeDaFuncao(); ex.: pickComputerMove();

**ou por atribuição a uma variável:**

const computerMove = pickComputerMove();

**Objetcts:**

Used to group many values;

Ex.: const product = {

name: ‘socks’,

price: 1090

};

**DOM (Document Object Model): one of the most important features of JS**

The DOM combines JS and HTML together and gives JS control under the webpage. It is a built-in Object from JS. This object has properties and methods.

Propriedades -> **onclick=”functionJS()”** and **onkeydown=”functionJS()”**

São ”**event listeners**” e executam código JS no caso de:

- um click (onclick )sobre o elemento; ou

- uma tecla pressionada (onkeydown).

Há outros event Listeners:

- onkeyup;

- onscroll;

- onmouseenter

-onmouseleave

And many more.

Você pode dar um console.log(event); -> event é um objeto que possui diversas informações sobre um evento (click, keydown, etc).

No caso do onkeydown(), se você fizer um console.log(event.key) terá acesso a tecla que foi pressionada no objeto “input”.

Document Object -> Represent the web Page;

Window Object -> Represent the Browser.

window.document

window.console.log(‘window’);

window.alert(‘Window’);

O JS possui um shortcut para o Window Object. Por isso, não precisamos digitá-lo ao executar os seus comandos (objetos).

Exercises:

















