# Tema 02 – Console, variáveis, operadores e outros

## Introdução

Olá meus queridxs alunxs, sejam bem vindos do curso de Frontend do Instituto da Oportunidade Social. Nessa aula, você verá alguns assuntos introdutórios de JavaScript, tais como: trabalhar como console, variáveis, operadores, etc.

## Valores e tipos de dados

Dentro do mundo do computador, existem apenas dados. Você pode ler dados, modificar dados, criar novos dados, mas o que nã o são dados não pode ser mencionado. Todos esses dados são armazenados como longas sequências de bits e, portanto, são fundamentalmente semelhantes.

### Valores

Os valores em um computador são quantidade de bits que representam uma informação.

|  |
| --- |
| 1 // Valor do tipo numérico  "Homer" // Valor do tipo string (sequência de caracteres) |

### Números

Valores podem ser o tipo numérico, que representam valores constantes. Como foi dito, o JavaScript tem um único tipo de número. Internamente, é representado como ponto flutuante de 64 bits. Você deve separar a parte fracionária de um número utilizando o ponto:

9.81

E para números muito grandes ou muito pequenos, você pode utilizar a notação científica:

|  |  |
| --- | --- |
| **Notação científica** | **Equivalente** |
| 2.998e8 | 2.998 × 108 = 299,800,000 |
| 6.5e-3 | 6.5 × 10-3 = 0,0065 |

### Números especiais

Existem três valores especiais em JavaScript que são considerados números, mas não se comportam como números normais. Os dois primeiros são **Infinity** e **-Infinity**, que representam os infinitos positivos e negativos. A expressão **Infinity - 1** ainda é infinito e assim por diante. O terceiro é o NaN significa “Not a Number” (“não é um número”), embora seja um valor do tipo numérico. Você obterá este resultado quando, por exemplo, tentar calcular 0/0 (zero dividido por zero).

### Strings

String é um outro tipo de dado, que é usado para representar texto, e devem ser envolvidas utilizando aspas simples ou aspas duplas.

|  |
| --- |
| "All is quiet on New Year's Day"  'A world in white gets underway'  'I want to be with you'  "Be with you, night and day" |

Strings não podem ser divididas, multiplicadas ou subtraídas, mas o operador + pode ser usado para concatenar (juntar, unir) duas ou mais strings. Por exemplo:

|  |
| --- |
| **Concatenando strings** |
| "Instituto" + ' ' + 'da' + " " + "Oportunidade" + ' ' + 'Social' |
| **Equivalente** |
| Instituto da Oportunidade Social |

Observe, que você pode utilizar aspas simples e duplas quando quiser dentro da string.

Você pode inserir caracteres especiais de escape na string, por exemplo:

"Essa é a primeira linha\nE essa é a segunda linha"

A sua string será interpretada como:

|  |
| --- |
| Essa é a primeira linha  E essa é a segunda linha |

O caractere de escape \n indica um nova linha e retorno do curso para o início do parágrafo. A barra invertida (\) é chamada de caractere de escape. Quando uma barra invertida é encontrada em uma string de caracteres, o interpretador examina o próximo caractere e o combina com a barra invertida para formar uma sequência de escape (Tabela 1).

Tabela – Código de escape comuns no JavaScript.

|  |  |
| --- | --- |
| **Código escape** | **Resutado** |
| \n | Nova linha e posiciona o cursor no início da nova linha do console. |
| \t | Tabulação horizontal. Move o cursor do console horizontalmente um espaço de tabulação. |
| \v | Tabulação vertical. Move o cursor do console verticalmente um espaço de tabulação. |
| \\ | Barra invertida. Insere uma barra invertida na string. |
| \' | Aspa simples. Insere uma aspa simples na string. |
| \" | Aspas duplas. Insere uma aspas dupla na string. |
| \b | Backspace. Retorna o cursor uma posição para trás. |

### Valores booleanos

Muitas vezes é útil ter um valor que distingue apenas duas possibilidades, como verdadeiro e falso. Para isso, o JavaScript possui um tipo booleano, que possui apenas dois valores, true (verdadeiro) e false (falso). Existem dois tipos de operadores que retornam um resultado do tipo booleano: operadores de comparação (relacionais) e operadores lógicos.

### Valores vazios

Existem dois valores especiais, escritos como **null** (nulos) e **undefined** (indefinidos), que são usados para denotar a ausência de um valor significativo. Eles próprios são valores, mas não contêm nenhuma informação. Muitas operações na linguagem que não produzem um valor significativo resultam em indefinidas simplesmente porque têm que produzir algum valor.

A diferença de significado entre **undefined** e **null** é um acidente do design do JavaScript, e isso não importa na maioria das vezes. Nos casos em que você realmente precisa se preocupar com esses valores, recomendo tratá-los como geralmente intercambiáveis.

### Conversão automática

JavaScript faz de tudo para aceitar quase qualquer programa que você forneça, até mesmo programas que fazem coisas estranhas. Por exemplo, você pode fazer operações com tipos deferentes de dados:

|  |  |
| --- | --- |
| **Operação** | **Resultado** |
| 8 \* null | 0 |
| "5" – 1 | 4 |
| "5" + 1 | 51 |
| "five" \* 2 | NaN |

Quando um operador é aplicado ao tipo "errado" de valor, o JavaScript silenciosamente converte esse valor para o tipo de que precisa, usando um conjunto de regras que muitas vezes não são o que você deseja ou espera. Isso é chamado de coerção de tipo.

## Operadores

Os operadores especificam uma avaliação/ação a ser executada em um ou mais operandos e podem ser aritméticos, relacionais ou lógicos.

### Operadores aritméticos

A principal coisa a fazer com os números é realizar operações aritméticas. Operações aritméticas como adição ou multiplicação tomam dois valores numéricos e produzem um novo número a partir deles. Por exemplo:

100 + 4 \* 11

Os símbolos + e \* são chamados de operadores aritméticos. A Tabela 2 mostra os operadores aritméticos do JavaScript.

Tabela 2 - Operadores aritméticos do JavaScript.

|  |  |  |
| --- | --- | --- |
| **Categoria** | **Operador** | **Descrição** |
| Operadores aritméticos | + | Adição |
| - | Subtração |
| \* | Multiplicação |
| \*\* | Exponenciação |
| / | Divisão |
| % | Módulo (Resto da divisão inteira) |
| ++ | Incremento |
| -- | Decremento |

Nesse exemplo, primeiro é realizada a multiplicação e não a adição por causa da precedência dos operadores. A ordem de precedência dos operandos aritméticos é:

|  |  |
| --- | --- |
| Shape  Description automatically generated with low confidence | Desse modo a operação:  3 \* 5 + 2  O programa primeiro faz a multiplicação e com o resultado dessa multiplicação realiza a soma. Os parênteses têm a maior precedência, portanto a operação:  3 \* (5 + 2)  Primeiro é realizado a soma dentro dos parênteses e com o resultado dessa soma realiza a multiplicação. |

### Operadores de comparação (relacionais)

Os operadores relacionais são utilizados na realização de comparação entre valores. Por exemplo:

|  |  |
| --- | --- |
| **Operação** | **Resultado** |
| 3 > 2 | true |
| 3 < 2 | false |

Os principais operadores de comparação são mostrados na Tabela 3.

Tabela 3 - Operadores de comparação do JavaScript.

|  |  |  |
| --- | --- | --- |
| **Categoria** | **Operador** | **Descrição** |
| Operadores de comparação | == | Igual |
| != | Diferente |
| < | Menor que |
| <= | Menor ou igual |
| > | Maior que |
| >= | Maior ou igual |

O JavaScript utiliza o operador triplo de igualdade, para garantir a comparação dos valores executando a conversão de tipos. Sendo assim no JavaScript:

|  |  |
| --- | --- |
| **Operação** | **Resultado** |
| 2 == "2" | True |
| 2 === "2" | False |

### Operadores lógicos

Operadores lógicos são usados em programação para concatenar expressões que estabelecem uma relação de comparação entre valores. Os principais operadores lógicos são mostrados na Tabela 4.

Tabela 4 - Operadores lógicos do JavaScript.

|  |  |  |
| --- | --- | --- |
| **Categoria** | **Operador** | **Descrição** |
| Operadores lógicos | && | Lógica “and” ou “e”, que retorna verdadeiro se todos os operandos forem verdadeiros. |
| || | Lógica “or” ou “ou”, que retorna verdadeiro se pelo menos um operando for verdadeiro. |
| ! | Lógica “not” ou “não”, que inverte o valor lógico se é verdadeiro, retorna falso e se é falso retorna verdadeiro. |

### Operador de atribuição

O operador de atribuição ( = ) permite atribuir um valor a uma variável, por exemplo. O JavaScript permite utilizar uma forma contraída do operador de atribuição. A Tabela 5 mostra diversos exemplos do operador de atribuição com os operadores, por exemplo: a instrução x += 2 é equivalente a x = x + 2, ou seja, não precisamos repetir o x novamente.

Tabela 5 - Operador de atribuição.

|  |  |  |
| --- | --- | --- |
| **Operador** | **Exemplo** | **Equivalente** |
| = | x = y | x = y |
| += | x += y | x = x + y |
| -= | x -= y | x = x - y |
| \*= | x \*= y | x = x \* y |
| /= | x /= y | x = x / y |
| %= | x %= y | x = x % y |
| \*\*= | x \*\*= y | x = x \*\* y |
| <<= | x <<= y | x = x << y |
| >>= | x >>= y | x = x >> y |
| >>>= | x >>>= y | x = x >>> y |
| &= | x &= y | x = x & y |
| ^= | x ^= y | x = x ^ y |
| |= | x |= y | x = x | y |
| \*\*= | x \*\*= y | x = x \*\* y |

### Operadores unários

Nem todos os operadores são símbolos. Alguns são escritos como palavras. Um exemplo é o operador **typeof**, que produz um valor de string nomeando o tipo do valor que você forneceu. Por exemplo:

|  |
| --- |
| console.log(typeof 4.5)  // → number  console.log(typeof "x")  // → string |

O typeof de 4.5 retornou que ele é um tipo numérico e de “x” retornou uma string.

Outro operador unário é o símbolo de – (menos), que quando usado em um número transforma o valor de positivo para negativo. Por exemplo:

-8

## Variáveis

Os dados de um programa são armazenados em variáveis. As variáveis são ferramentas indispensáveis na programação, são nelas que colocamos valores para podermos trabalhar com esses dados posteriormente.

|  |
| --- |
| Atenção: O nome de variáveis ou constantes deve começar por letras ou underline ( \_ ) e pode conter números e não pode conter caracteres especiais. O nome também não pode ser uma palavra-chave ou palavra reservada.   * Nomes válidos:   + x, y, ola\_01, \_teste * Nomes inválidos:   + 12\_teste, nome pessoa, 1xx3   Programadores geralmente usam somente letras minúsculas para os nomes de variáveis, mas isso é uma convenção utilizada e não uma regra. |

No JavaScript, podemos declarar variáveis de três maneiras:

* Usando a palavra reservada **var**
* Usando a palavra reservada **let**
* Usando a palavra reservada **const**

A instrução **var** declara uma variável no escopo de uma função ou no escopo global e é opcional inicializar o seu valor.

|  |
| --- |
| var x = 1; // Variável numerica  var nome = "Homer"; // Variável string  var teste; // Variável não inicializada |

A palavra-chave **let** foi introduzida na ES6 em 2015. Variáveis definidas com **let** não podem ser redeclaradas, ou seja, você não pode declarar novamente uma variável com o mesmo nome. Por isso, nos programas mais recentes declarar variáveis utilizando **let** estacada vez mais comum.

|  |  |
| --- | --- |
| **O let não permite isso** | **O var permite isso** |
| let x = "John Doe";  let x = 0;  // SyntaxError: 'x' has already been declared | var x = "John Doe";  var x = 0;  // Sem problemas, 😊 |

A instrução **let** também permite que você declare uma variável no escopo de bloco, ou seja, em uma região delimitada pela abertura e fechamento de chaves { }. Por exemplo:

|  |
| --- |
| {  let z = 10;  // Comandos  } |

A palavra-chave **const** foi também introduzida na ES6 em 2015. Variáveis definidas com **const** não podem ser redeclaradas e não podem ter seu valor alterado, ou seja, você não pode declarar novamente uma variável com o mesmo nome e uma vez inicializada o seu valor será o mesmo até o fim do programa.

|  |
| --- |
| const PI = 3.141592653589793;  PI = 3.14; // Isso produzirá um erro  PI = PI + 10; // Isso também produzirá um erro |

|  |
| --- |
| **Atenção**  Arrays declarados com a palavra-chave constante tem o comportamento um pouco diferente de variáveis, pois eles permitem você alterar e adicionar elementos. Mas esse será um assunto para tratarmos mais para frente. |

|  |  |  |
| --- | --- | --- |
| **Para Saber Mais**   |  |  | | --- | --- | | **Escopo de Variáveis**: O escopo é o conjunto de regras que determinam o uso e a validade de variáveis nas diversas partes do programa. Um escopo define uma região do programa definida pela abertura e fechamento de chaves { }. JavaScript permite criar variáveis em três escopos: global, de função e de bloco. O escopo de função função e de bloco são também chamados de escopo local. No caso, o escopo de uma variável define uma região do código onde a variável é visível, ou seja, ela pode ser acessada.  Por exemplo: a variável **x** foi declarada globalmente e pode ser acessada ou modificada em qualquer lugar do código, pois não há um escopo a delimitando. Então ela pode ser acessada ou modificada pela função ou pelo bloco normalmente. Já a variável **y** está definida localmente no escopo da função **lala** e só vai existir dentro do escopo dessa função, não podendo ser acessada fora da função. O mesmo acontece com a variável **z** só pode ser acessada no escopo do bloco. | A picture containing chart  Description automatically generated | |

### Strict Mode

Você pode “falar” para o interpretador do JavaScript que você quer usar o **strict mode**. O **strict mode** indica que você não pode usar nenhuma variável se a devida declaração. Para ativar esse modo, você deve colocar a string "use strict" no início do arquivo que você irá colocar seu código JavaScript. Vejamos o exemplo:

|  |
| --- |
| "use strict";  x = 3.14; // Isso gerará erro, pois a variável x não foi declarada |

## Objeto console

O objeto console fornece acesso ao console (terminal) de *debugging* do navegador. O console possui diversos métodos, mas vamos aprender com calma quatro métodos do objeto console:

|  |  |
| --- | --- |
| **Método** | **Descrição** |
| clear() | Limpa o console. |
| error() | Envia uma mensagem de erro no console. |
| log() | Envia uma mensagem no console. |
| warn() | Envia uma mensagem de aviso no console. |

Utilizar o objeto console para fazer debug no seu código e visualizar as coisas acontecendo é melhor e mais prático do que usar a mensagem de alerta. Por isso, usaremos esse objeto constantemente nos nossos códigos.

|  |
| --- |
| **Atenção**  Não se preocupe no momento em saber o que é um objeto em JavaScript. Esse conceito será abordado mais adiante com mais detalhes. |

## Vamos praticar

Vamos criar um projeto para testar toda teoria que vimos nesse tema. e. Siga os passos para criar o projeto:

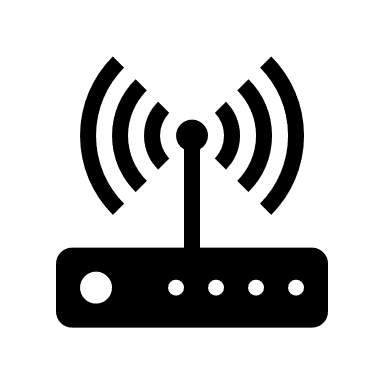
1. Abra o VS Code e escolha um diretório de trabalho para o seu projeto.
2. Crie um diretório para seu projeto com o nome representativo, por exemplo, 02\_JS\_Console\_Variavies\_Operadores.
3. Crie um arquivo dentro do diretório do projeto com o nome index.html.
4. Insira o seguinte código no seu arquivo index.html.

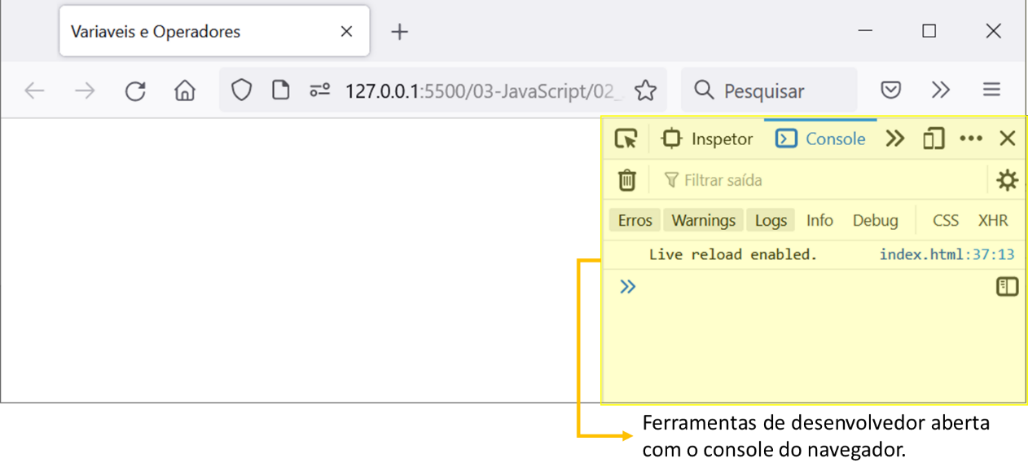
|  |
| --- |
| <!DOCTYPE html>  <html lang="pt-br">      <head>          <meta charset="UTF-8" />          <link rel="shortcut icon" href="#" />          <title>Variaveis e Operadores</title>      </head>      <body>          <script src="main.js"></script>      </body>  </html> |

1. Esse código mostra a marcação <script> sem nenhum código JavaScript entre a abertura e o fechamento da tag, apenas o atributo **src** com o valor main.js. Isso significa que o código JavaScript está em um arquivo externo. Portanto, temos que criar esse novo arquivo **main.js** dentro do diretório do projeto.
2. Ao criar o seu arquivo a hierarquia do diretório do projeto deverá ficar assim:

Text

Description automatically generated

1. Abra o arquivo index.html e clique no botão **Go Live** da extensão **Live Server** e abra a Ferramenta de Desenvolvedor do seu navegador (use o Chrome ou o Firefox para depurar seus projetos). Você pode abrir pela tecla de atalho **F12**.



|  |
| --- |
| **Atenção**  O navegador web abre arquivos com a extensão .html, por isso devemos sempre voltar ao arquivo HTML da nossa aplicação antes de clicar para abrir a extensão **Live Server**. Se você tentar abrir a extensão com um o arquivo de extensão diferente, o navegador irá apresentar a hierarquia de arquivos e diretórios do projeto. |

### Primeiros comandos em JS

Você pode digitar os comandos do JavaScript diretamente no console ou inserir no arquivo **main.js** como iremos fazer nessa prática. Como já estamos com a aplicação aberta no navegador web (já iniciamos o **Live Server**). Então sempre que colocarmos uma instrução em JavaScript utilizando o objeto console, a mensagem será exibida no console do navegador. Vamos começar a programar, siga os passos para ira atualizando o seu arquivo **main.js**.

1. Os primeiros comandos serão mensagens utilizando os métodos **log()**, **error()** e **warn()** do objeto console. Insira os comandos no seu arquivo **main.js**.

|  |  |
| --- | --- |
| 1 | console.log('Hello World!');  console.error('Essa é uma mensagem de erro');  console.warn('Essa é uma mensagem de aviso!'); |
| 2 |
| 3 |

1. Ao salvar o arquivo, você poderá visualizar as mensagens impressas no console.

Graphical user interface, text, application, Word

Description automatically generated

Observe que cada tipo de mensagem é apresentado com cor diferente e a mensagem de erro ativa uma **badge** na barra do console, que indica que algo está errado na sua aplicação.

1. Agora vamos trabalhar com variáveis. O código abaixo declara duas variáveis idade e nome. Vamos usar o método clear() sempre que mudarmos de assuntos, para que nosso console não fique cheio de mensagens.

|  |  |
| --- | --- |
| 6 | // let, const  console.clear();  let idade = 30;  const nome = 'Nico';  console.log(nome);  console.log(idade);  idade = 31;  console.log(idade); |
| 7 |
| 8 |
| 9 |
| 10 |
| 11 |
| 12 |
| 13 |

Na linha 8, declaramos uma variável numérica e, na linha 9, declaramos uma string. Nas linhas 10 e 11, imprimimos o valor das variáveis separadamente. Na linha 12, atualizamos o valor da variável **idade**, lembre-se não é possível alterar o valor da variável **nome**, pois ela foi declarada como **const**. Na linha 13, o valor atualizado da variável idade.

**Atenção**: Observe que estamos colocando os números das linhas nos códigos, isso uma sugestão de onde você deve inserir o código mostrado e, também, para não precisarmos colocar o código completo. Você pode fazer como quiser, apagar o código anterior e substituir pelo novo código mostrado ou manter o código completo inserindo a instrução na linha sugerida.

1. As salvar o arquivo **main.js**, podemos ver o resultado impresso no console.

Graphical user interface, application, Word

Description automatically generated

1. Agora vamos trabalhar com os outros tipos de dados e utilizar o operador unário **typeof** para verificar o tipo de cada uma das variáveis declaradas. Para isso insira o seguinte código no arquivo **main.js**

|  |  |
| --- | --- |
| 14 | // String, Numbers, Boolean, null, undefined  console.clear();  const nome\_pessoa = 'John';  const idade1 = 30;  const rating = 4.5;  const isCool = true;  const x = null;  const y = undefined;  let z;  console.log(typeof nome\_pessoa); // Type of const  console.log(typeof idade1);  console.log(typeof rating);  console.log(typeof isCool);  console.log(typeof x);  console.log(typeof y);  console.log(typeof z); |
| 15 |
| 16 |
| 17 |
| 18 |
| 19 |
| 20 |
| 21 |
| 22 |
| 23 |
| 24 |
| 25 |
| 26 |
| 27 |
| 28 |
| 29 |
| 30 |

Observe na linha 16 que devemos declarar **nome\_pessoa**, pois uma variável **nome** já foi declarada anteriormente como **const** e não podemos usar variáveis com nomes iguais. O mesmo acontece com a variável **idade1** (linha 17).

1. Ao salvar o arquivo **main.js**, você pode ver no console do navegador as mensagens impressas com a cada tipo de dados das variáveis declaradas.

Graphical user interface, text, application

Description automatically generated

Na ordem impressa no console temos:

* Tipo de dados da variável nome\_pessoa: string
* Tipo de dados da variável idade1: number
* Tipo de dados da variável rating: number
* Tipo de dados da variável isCool: boolean
* Tipo de dados da variável x: object
* Tipo de dados da variável y: undefined
* Tipo de dados da variável z: undefined

1. Concatenar strings é muito comum em JavaScript. Você irá fazer isso muitas vezes, portanto preste bastante atenção será que esse assunto for mostrado no material. Vamos inserir o seguinte código para concatenar strings no arquivo **main.js**

|  |  |
| --- | --- |
| 32 | // Concatenar strings  console.clear();  const pessoa = 'Irmão do Jorel';  const idade2 = 30;  console.log('Meu nome é ' + pessoa + ' e eu tenho ' + idade2); |
| 33 |
| 34 |
| 35 |
| 36 |

A linha 36 mostra uma string sendo concatenado com o operador +, nessa maneira de concatenar o texto deve colocado entre aspas simples ou duplas e o nome da variável sem aspas.

1. Ao salvar o arquivo **main.js**, é possível visualizar o resultado no console.

Graphical user interface, application, Word

Description automatically generated

1. Outra forma mais prática de concatenar strings é usar Template Strings. Atualize o arquivo **main.js** com o exemplo abaixo.

|  |  |
| --- | --- |
| 38 | // Template String  const hello = `Meu nome é ${pessoa} e eu tenho ${idade2}`;  console.log(hello); |
| 39 |
| 40 |

A linha 40 contém o que chamamos de Template String. Falaremos mais sobre essa forma prática de concatenar strings, mas uma rápida explicação: você deve colocar a string entre acentos de crase, os nomes das variáveis começando por cifrão ($) e entre chaves { } e não é necessário utilizar o operador +

1. O resultado apresentado no console é o mesmo do anterior.

Graphical user interface, application, Word

Description automatically generated

1. Por fim, vamos ver alguns resultados de uso de operadores de comparação e lógicos. Atualize o arquivo **main.js** com o código abaixo.

|  |  |
| --- | --- |
| 42 | // Operadores de comparação e lógico  console.clear();  let teste1 = 1;  let teste2 = 0;  let valor1 = true;  let valor2 = false;  console.log(teste1 > teste2);  // Operador de comparação  console.log(teste1 < teste2);  // Operador de comparação  console.log(teste1 && teste2); // Operador lógico  console.log(valor1 && valor2); // Operador lógico |
| 43 |
| 44 |
| 45 |
| 46 |
| 47 |
| 48 |
| 49 |
| 50 |
| 51 |
| 52 |

1. Observe o resultado no console do navegador.

Graphical user interface, application, Word

Description automatically generated

**Importante:** Falaremos mais sobres esses testes lógicos na parte de estruturas condicionais e estruturas de repetição. Mas se você quiser se adiantar pode procurar por operações lógicas: AND, OR, NOT e por tabelas verdades.