**JavaScript**

Cliente: Quem usa os dados.

Server: Quem fornece os dados.

O Javascript para a interatividade (clientes) ou seja, primeiramente ele era usado somente para front-end, porém, atualmente ele vem sendo usado também no servidor, ou seja, no back-end.

Para criar um script interno basta abrir a tag <script></script> dentro do body do html. Geralmente abrimos essa tag no final do documento.

**Comandos simples**

<script>

Alert ('Meu primeiro script!') //Serve para exibir uma mensagem no topo da tela.

confirm ('Está gostando de JS?') //Exibe também uma mensagem, porém existe um botão 'OK' e um botão 'CANCELAR'.

prompt ('Qual é seu nome?') //Exibe uma barra de escrita, como se fosse interagir com o usuario.

</script>

Também existem alguns comandos para tratamentos de dados, tais eles como:

<script>

        var nome = prompt (`Qual é seu nome? `)

        document.write(`Ola, <strong>${nome}</strong>! Seu nome tem ${nome.length} letras. </br>`)

        document.write(`Seu nome em maiúsculo fica ${nome.toUpperCase()}`)

</script>

Uma breve explicação é que, nome.length serve para contar a quantidade de caracteres presente no conteúdo da variável. Já o nome.toUpperCase serve para que todos os caracteres da variável fique em maiúsculo.

No exemplo também e possível ver o document.write que serve para escrever a frase no documento html.

Podemos também ‘Ler’ o salário de uma pessoa com um comando que já arruma a questão de ponto e virgula e até demonstra a moeda em que o salário e pago, veja:

Texto

Descrição gerada automaticamente com confiança média

**Variáveis**

Para entendermos o que é uma variável, basta pensarmos em um estacionamento, onde nele obviamente existem vagas e cada uma dessas vagas tem uma nomenclatura específica, exemplo:

Vagas para carro: C1, C2, C3

Vagas para caminhão: T1, T2, T3

Vagas para moto: M1, M2, M3

Percebe-se que C, T e M são indicadores (nome da variável), entre tudo esses indicadores possuem algumas regras:

* Podem começar com: letra, $ ou \_
* Não podem começar com números
* É possível usar: Letras, números, acentos e símbolos
* Não pode conter espaço (para representar um espaço usasse \_)
* Não podem ser palavras reservadas ou palavras que são nome de outros comandos (alert, prompt etc.)

Agora para relacionarmos esse estacionamento com as variáveis da programação pensamos da seguinte forma, o computador guarda dados como se fosse carros, cada um com sua nomenclatura, e os indicadores citados acima servem para denominar esses dados. Exemplo:

<script>

        var n1 = 10

        var n2 = 20

</script>

Nesses exemplos percebe-se que atribuímos dois valores diferentes para cada variável (variável pode ser representada por: var, let ou const).

Para testarmos essa variável usamos o NODEJS, ou algum outro terminal, veja:

Forma

Descrição gerada automaticamente com confiança média

Lembrando que sempre que criamos uma variável o sinal de igual (=) é lido como ‘recebe’, ou seja, a variável n1 recebe o valor 10 e a variável n2 recebe o valor 20.

Agora que vimos como criar uma variável podemos juntá-las com alguns outros comandos visto anteriormente, veja:

<script>

        var nome = (prompt(`Digite seu nome`))

        var idade = (prompt(`Digite sua idade`))

        var nota = (prompt(`Digite sua nota`))

</script>

Nesse caso, o valor da variável será igual a oque foi escrito na barra do prompt. Lembrando que todo valor escrito dentro de um prompt será considerado uma string (conjunto de letras) mesmo sendo um number (conjunto de números). Ou seja, se quisermos criar outra variável que demonstre o resultado da soma de duas variáveis teremos que transformar essa string em um número, temos que usar o seguinte comando:

<script>

        var n1 = Number (prompt('Digite um número'))

        var n2 = Number (prompt('Digite outro número'))

</script>

Agora com o resultado da nossa variável tendo se tornado um número podemos criar uma variável que soma esses dois valores escolhidos anteriormente, veja:

<script>

        var n1 = Number.parseFloat (prompt('Digite um número'))

        var n2 = Number (prompt('Digite outro número'))

        var soma = n1 + n2

</script>

**Juntando variáveis em uma frase genérica**

Voltando a usar outro comando apreendido anteriormente (alert) podemos juntar algumas variáveis nessa mensagem usando o + que serve para concatenação, veja:

<script>

        var nome = prompt('Qual é seu nome?')

        alert('É um prazer em te conhecer, ' + nome + '!')

</script>

Lembrando que toda string precisa estar entre aspas simples (‘’), aspas compostas (“”) ou a crase (``) que serve para um formato específico que veremos a seguir.

Quando queremos juntar várias variáveis em uma única frase podemos facilitar nosso trabalho colocando entre crase (``) e usando o placeholder ${x}, veja no exemplo a seguir onde relembramos as variáveis criadas anteriormente:

<script>

        var n1 = Number.parseFloat (prompt('Digite um número'))

        var n2 = Number (prompt('Digite outro número'))

        var soma = n1 + n2

        alert(`A soma de ${n1} e ${n2} e igual a ${soma}`)

</script>

Sendo assim a mensagem exibida no alert terá todas as informações obtidas pelas variáveis.

**DataTypes**

|  |
| --- |
| Number (infinity, NaN) |
| String |
| Boolean |
| null |
| Undefinid |
| Object (array) |
| Function |

**Operadores**

|  |
| --- |
| Aritméticos |
| Atribuição |
| Relacionais |
| Lógicos |
| Ternário |

**Aritméticos/Atribuição**

Esses são os operários aritméticos em Javascript, seus significados são:

(+) Serve como adição.

(-) Serve como subtração.

(\*) Serve para multiplicação.

(/) Serve para divisão de número real, ou aqueles que seu resultado da algo com virgula.

(%) Resto da divisão, ou seja, aquele número que sobra para que não tenha um resultado com virgula.

(\*\*) Serve para potenciação.

No caso demonstrado abaixo e possível entender que a princípio (N) tinha um valor, porém logo em seguida esse valor foi alterado (**auto-atribuições**) em uma conta em que o resultado dessa conta se torna o valor atual da variável, podendo nos relembrar de novo o exemplo do estacionamento onde um carro sai para entrada de outro carro na vaga. Vale lembrar que a ordem usada para calcular com esses operários e padrão a da matemática ensinada em sala de aula. Primeiro oque está dentro de parênteses (x) depois, multiplicação ou divisão e só então resolvemos adição ou subtração. Exemplo:

Forma

Descrição gerada automaticamente com confiança média

Veja que o valor da variável (n) foi alterado pela adição que foi dita logo em seguida. Podemos também usar algumas simplificações, veja:

Forma

Descrição gerada automaticamente com confiança média

Ou seja, toda vez que botamos algum sinal aritmético atras do sinal = (recebe) o próprio Javascript faz a conta e altera o valor da variável.

**Relacionais**

Esses são os operários de relação em Javascript, seus significados são:

(>) Mostra que o número da esquerda é maior que o da direita.

(<) Mostra que o número da direita é maior que o da esquerda.

(>=) Mostra que o número da esquerda é maior ou igual ao da direita.

(<=) Mostra que o número da direita é maior ou igual ao da esquerda.

(==) Mostra que o número da esquerda é igual ao da direita.

(!=) Mostra que o número da esquerda e diferente do número da direita.

Lembrando que essas relações podem ser tanto falsas (false) quanto verdadeiras (true), vejamos alguns testes no NodeJS:

Forma

Descrição gerada automaticamente com confiança média

**Identidade**

Precisamos dos operários de igualdade porque mesmo comparando uma string com um número o Javascript demonstra sendo verdadeiro, já que eles são idênticos, porém existe uma forma de reverter essa situação usando o operário (===), para entender melhor veja no exemplo a seguir feito o NodeJS:

Forma

Descrição gerada automaticamente com confiança média

Vejamos que na variável n1 foi atribuído um valor do tipo number, já na variável n2 atribuímos o mesmo valor só que do tipo string, porém quando usamos == para ver se elas são iguais o Javascript confirma porque são idênticas e depois quando usamos === para confirmar sua identidade o Javascript reconhece que existe uma diferença e fala que a relação é falsa.

**Lógicos**

Como operários lógicos temos (!) que demonstra negação (&&) que demonstra conjunção ‘e’ e (II) que demonstra uma disjunção ‘ou’.

Por ser algo lógicos podemos entender da seguinte forma.

Calculadora preta com letras brancas

Descrição gerada automaticamenteCalculadora preta com letras brancas em fundo preto

Descrição gerada automaticamenteTexto

Descrição gerada automaticamente

**Ternário**

Os operadores ternários servem para indicar o que vai acontecer caso o valor for real ou falso veja no exemplo a seguir.

Forma

Descrição gerada automaticamente com confiança média

Ou seja, a primeira casa depois do ? é a resposta caso a relação seja verdadeira, já o segundo é a resposta caso a relação seja falsa.

**DOM (Document Object Model)**

|  |
| --- |
| Por marca |
| Por ID |
| Por Nome |
| Por Classe |
| Por Seletor |

Na tabela a cima vemos 5 tipos de formas para trabalhar com DOM, cada uma com sua caracteristica em especifico, veja.

**Por marca:** Serve para selecionar elementos por sua tag, porém é possivel ver no codigo que se trata de algo no plural, ou seja, após o (x) temos que colocar entre colchetes [x] o numero do paragrafo, lembrando que essa contagem começa do 0.

<script>

        document.getElementsByTagName(p)[0]

</script>

**Por ID:** Serve para selecionar elementos atraves de seu ID, essa ferramenta não nescessita de [x] ja que ID é uma propriedade unica.

<script>

        document.getElementById(redessociais)

</script>

**Por nome:** Serve para selecionar elementos através de seu nome.

<script>

        document.getElementsByClassName(Instagram)

</script>

**Por classe:** Funciona igual o por ID, porém ele se refere ao nome da classe, não o do ID.

<script>

        document.getElementsByClassName(container)

</script>

**Por seletor:** Serve como coringa, serve para todos os anteriores.

<script>

        document.querySelector()

        document.querySelectorAll()

</script>

**Funções**

Para usarmos funções no Javascript existem duas maneiras, sendo elas pelo próprio script ou pelo HTML, agora explicando como funciona, primeiramente temos que escolher o tipo da função ou em outras palavras o evento que tem que acontecer para que a função seja ativada, veja melhor no exemplo abaixo:

<script>

        var a = document.getElementById('area')

        var corpo = document.body

        function clicar(){

            a.style.background = 'blue'

            a.style.color = 'white'

            corpo.style.background = 'black'

        }

</script>

No exemplo vemos que primeiramente a div com ID área foi atribuída com a função de click ou seja, toda vez que essa div for clicada algo ira acontecer, agora analisando o script vemos que primeiramente a variável a foi definida como a div área, já a variável corpo foi definida como o body, sendo assim ao analisar a function vemos que o estilo do fundo da variável a ira se tornar azul com escritas brancas e na outra variável o fundo também ira mudar, porem para preto.

Diagrama

Descrição gerada automaticamente com confiança média Relógio digital com números

Descrição gerada automaticamente com confiança média

Tela preta com letras brancas

Descrição gerada automaticamente