* JavaScript
* O que é?

A linguagem JavaScript é uma linguagem de programação considerada dinâmica ( Por permitir que uma variável seja atribuido por um valor de um tipo e por permitir que o tipo dessa variável seja alterado por uma outra atribuição) e interpretada.   
A característica interpetrada se refere a uma linguagem de programação em que o código fonte dessa linguagem é executado por um programa de computador, chamado interpretador, que é executado pelo sistema operacional ou processador. Ele lê o código linha por linha em vez de ler ele todo de uma só vez.

O JavaScript executa diretamente no navegador. E ele surgiu como o navegador Netscape. E segue as especificações ECMAScript.   
  
O JavaScript pode ser adcionado de duas maneiras na página. Uma por meio de um script externo, que é um link no HTML direcionando para o documento JavaScript. O elemento Script não é um elemento vazio, então ele precisa de uma tag de abertura e uma tag de fechamento.   
Ex:

<Script scr=”./js/meu-script.js”> </Script>  
  
  
E a outra por meio do Script Inline, onde o conteúdo do Script fica dentro da tag <Script> não usando o atributo SRC.  
  
<Script>

Console.log (Este é um Script)

</Script>

Obs: As boas práticas indicam que é melhor usar um script externo. Pois o tamanho desse script pode ser muito grande para ficar dentro do html.

O JavaScript é executado na sequencia que ele é definido no HTML. Então o navegador processa o HTML, como resposta da requisição e sempre que há um elemento Script, ele vai executar esse elemento Script e só depois que ele executar o Script que ele vai continuar processando o documento.

* Sintaxe Básica de um Script  
    
  No JavaScript eu começo fazendo a declaração de uma variável atráves da palavra chave *Var* , seguida de um nome para essa variável, uso o operador = e atribuo um valor entre ‘’.   
  O uso do ponto e vírgula (;) no final da minha variável é opicional. Porém é importante seguir um padrão na escrita do código.   
    
  Var *mensagem =* ‘Olá’ ;

É possível atribuir a variável para um outro valor:

mensagem= ‘Olá Danilo”;   
  
Para imprimir essa váriavel é necessário usar uma função:   
  
 Alert (mensagem);

É assim que fica um documento JavaScript:   
  
 Var *mensagem =* ‘Olá’ ;  
mensagem= ‘Olá Danilo”;  
  
 Alert (mensagem);

* Tipos primitivos   
    
  São três tipos principais que são usados com frequência:   
    
  🡪 Var vNumber , que aceita tanto números inteiros como números decimais.   
    
  EX: Var vNumber= 5   
    
  🡪 Var String, é definido por qualquer sequência de carácteres. Elas podem ser escritas tanto entre aspas simples ‘’, como entre aspas duplas “”. Mas é importante manter uma consistência na escrita do código.   
    
    
  🡪 Var vBoolean, recebe apenas dois tipos de valores, o True ou false.   
    
  Obs: Dentro do Console, no navegador, é possível checar que tipo de variável eu estou trabalhando. Basta eu colocar (typeof e o nome da minha variável) e ele vai me trazer que tipo de variável eu estou trabalhando.
* Objetos – Em JavaScript, um objeto é uma entidade independente, com propriedades e tipos ue definem suas características. Elas são escritas entre chaves {}. Todos os dados que são colocados dentro do meu objeto podem ser alterados ou apagados por novas variáveis.   
  Quando eu solicito ao console uma variável que não está completa ou que foi deletada, ele me responde com a palavra “Undefined”.   
  Como em outros programas, é possível usar o valor Null em JavaScript. Ele representa um valor nulo ou vazio para aquela variável. Porém não confundir com o Undefined. Pois eles não são a mesma coisa.   
    
  Obs: Diferença entre Null e Undefined – Segundo a documentação ECMAScript, o valor undefined é utilizado quando uma variável não possui valor. Já o Null é utilizado quando se quer intencionalmente dizer que há uma ausência de qualquer valor de objeto para aquela variável.   
    
  🡪Array – Um Array no JavaScript é um objeto com um contrutor único, com uma sintaxe literal e com um conjunto adcional de propriedades e de metódos herdados de um protótipo de array. Eles são usados para armazenar vários valores em uma única variável, e são escritos entre colchetes [].   
    
  Obs: Uma diferença importante entre tipos primitivos e objetos no JavaScript é que os Objetos primitivos copiam o seu valor. Já os objetos tem o valor como uma referência.  
    
  Ex:   
  Var primitivo = 1;  
  Var primitivo2 = primitivo1 // copia o valor   
  primitivo2 =2   
    
  console.log(primitivo1); // Imprime 1 como valor   
  console.log(primitivo2); //Imprime 2 como valor   
    
    
  var objeto1 = {x: 1};  
  var objeto2 = objeto1 //copia a referência  
  objeto2.x = 2   
    
  console.log(objeto1.x); //Imprime 2 como valor   
  console.log(objeto2.x); //Imprime 2 como valor
  + Operadores e expressões no JavaScript   
      
    🡪Operadores Lógicos – São usados para realizar operações lógicas. Elas podem ser do tipo Negação (!), tipo E lógico (&&) e OU lógico (||). Os operandos devem ser lógicos, verdadeiro ou falso. Também podem operar sobre expressões lógicas, ou seja, que retornem valores verdadeiros ou falso.   
      
    E lógico (&&) - O Operador E lógico recebe dois operandos e retorna verdadeiro se, e somente se ambos os operandos sejam verdadeiros . Retorna falso, caso contrário.   
      
    Ou lógico (||) – Para o operador Ou lógico (||) retornar verdadeiro, basta que um dos operandos seja verdadeiro. Ele também retorna verdadeiro caso os dois operando sejam verdadeiros. Retorna falso, se os dois forem falsos.   
      
    Negação (!) – O operador de negação (!) é um operando unário, isto é, opera sobre apenas um operando. Ele nega, inverte o valor lógico do operando.   
      
    OBS: Diferente de quando eu uso o boolean ou o Sting, os operadores lógicos funcionam diferente para as variáveis numéricas. O número 0 siginifca FALSE, enquanto que qualquer outro número é TRUE.   
    No ou lógico, Qualquer número que estiver com ZERO vai ser a resposta da variável.   
    EX:   
    8||0   
    8  
      
    Porém quando os dois números forem TRUE, ele vai ter como resposta o primeiro operador.   
      
    EX: 8||5  
     8  
      
    Já o E lógico, se ambos os operadores forem TRUE, ele vai retornar o segundo operador.   
    EX: 8||5  
     5  
      
    E se a operação tiver algum número que seja FALSE, ele vai retornar o número FALSE.   
      
    EX 8||0   
     0  
      
      
      
      
      
    🡪 Operadores Aritiméticos – São operadores usados nas contas aritiméticas da matemática:   
      
      
    X+Y - Adição   
    X-Y - Subtração   
    +X – Converte para número (Se já não for)  
    -X – Inversão de sinal  
    X/Y – Divisão   
    X\*Y – Multiplicação  
    X%Y – Resto da divisão de X por Y  
    X \*\* Y – Exponenciação  
      
    OBS: O operador Aritimético adição também pode ser usado para somar Strings. Chamado Concatenação. O resultado que ele produz pode ser visto como uma soma. Este operador irá avaliar os valores dos operandos, que podem ser mais de dois e retornar uma string que é a junção desses valores. Vale notar que os operandos não precisam necessariamente ser todos do tipo String.   
      
    EX: Var numitens = 10;   
      
    var mensagem = ‘O número de itens no pedido é’ + numitens + ‘.’  
      
    O Valor da variável ‘mensagem’ será: O número de itens no pedido é 10.  
      
      
    Os operadores podem ser escritos com atribuições:   
      
    X + = y - Adição   
    X - = y – Subtração  
    X / = y – Divisão   
    X \* = y – Multiplicação  
    X % = y – Resto da divisão de x por y   
    X \*\* = y – Exponenciação   
      
    Essa forma de escrever pode ser entendida como uma abreviação:   
      
    EX: X=5  
    X + = 2;   
    X = 7;   
      
    É o equivalente a X= X+2   
      
      
    🡪 Operadores de Incremento e Decremento -   
      
    X ++ - Retorna X e depois incrementa   
    X - - Retorna X e depois decrementa   
    ++X – Incrementa e depois retorna X  
    --X - Decrementa e depois retorna X  
      
    EX: Var X = 1;   
    X++   
    X = 1  
    X= 2   
      
    - Nesse primeiro exemplo, ele primeiro trouxe o valor original de X e depois aumentou, mudando o valor de X.   
      
    ++X   
    X= 3   
      
    -Nesse ele aumenta o valor de X antes de trazer o valor da variável.   
      
      
    🡪 Operadores de Comparação:   
      
    X < Y - Menor   
    X > Y – Maior   
    X <= Y – Menor ou igual   
    X >= y – Maior ou igual   
      
      
      
    🡪 Operadores de igualdade:   
      
    X==y – Igual   
    X != y - Diferente  
    X === y – Exatamente Igual   
    X ! == y - Não Extamente igual   
      
      
    🡪 Precedência de operadores – A precedência de operadores determina a ordem em que os operadores são processados. Operadores com maior precedência são processados primeiro.
  + Funções – Funções são blocos de construção fundamentais em JavaScript. Uma Função é um procedimento de JavaScript, um conjunto de instruções que executa uma tarefa ou calcula um valor.   
    A definição de uma função, também chamada de Declaração da Função, consiste no uso da palavra chave “function”, seguida por:   
      
    -Nome da Função   
    - Lista de argumentos para a função, entre parênteses e separado por vírgulas.   
    -Declarações JavaScript que definem a função, ente chaves {}
  + Escopo – O Escopo em JavaScript se refere à visibilidade de variáveis e funções dentro de um programa. Podemos determinar que no JavaScript existem dois tipos de escopo, o globla e o Local, quando uma variável está fora da função, ela é uma variável global.   
    Quando ela está dentro da função, ela se torna uma variável Local.
  + Comandos de Decisão – Os comandos de decisão permitem implementar desvios condicionais, isto é escolher qual comando, ou conjunto de comandos, será executado de acordo com a avaliação de uma condição.   
      
    O IF é uma estrutura controle utilizada para tomar decisões. Realiza uma ou mais operações em função de uma expressão na forma de condição. Basicamente primeiro é dada uma condição e caso seja positiva, ele executa a ação. A estrutura básica de IF é:   
      
    IF (Condição) {código a ser executado}  
      
    IF (A>B) {  
    Return A}  
      
      
    O ELSE serve basicamente para suprir a parte negativa da condição IF, ou seja, quando a condição no IF for falsa, o ELSE assume e executa a ação.  
    Sua sintaxe é igual a do IF, mas com uma pequena diferença:   
      
    IF (Condição) {código executado se verdadeiro}  
    Else {Código executado se falso}  
      
    IF (A>B) {  
    Return A}   
    Else {  
    Return B}  
      
    🡪 Operador Ternário – O Operador condicional ternário é o único operador JavaScript que possui três operandos. Este operador é frequentemente usado como um atalho para a instrução IF  
      
    Function menor (A, B, C)  
    Var r = (A < B && A < C) ? a : (B < A && B < C) ? B : C;  
    return r  
      
    🡪 Switch Case - O Switch Case é uma estrutura de decisão usada quando precisamos testar condições para determinar qual função será executada em seguida. Essa expressão nos permite substituir múltiplos if else, tornando o código criado mais simples, legível e fácil de manter.   
      
    EX:   
    function formatmes(mes) {  
    Switch (mes) {  
    Case 1:   
     return “Janeiro”;   
      
    Case 2:  
    Return “Fevereiro”;
  + Comandos de repetição - Estruturas de repetição também conhecidas como laços (loops), são utilizados para executar, repetidamente, uma instrução ou bloco de instrução enquanto determinada condição estiver sendo satisfeita.   
    Qualquer que seja a estrutura de repetição, ela conterá quatro elementos fundamentais:   
      
    **Inicialização** – A Inicialização compõe-se de todo código que determina a condição inicial de repetição.   
      
    **Condição** - A Condição é uma expressão booleana avaliada após cada leitura do corpo e determina se uma nova leitura deve ser feita ou se a estrutura de repetição deve ser encerrada.  
      
    **Corpo** – O Corpo compõe-se de todas as instruções que são executadas repetidamente.   
      
    **Interação** - A interação é a instrução que deve ser executada depois do corpo e antes de uma nova repetição.   
      
    **While** – Este termo é usado para construir uma estrutura de repetição que executa, repetidamente, uma única instrução ou um bloco delas “enquanto” uma expressão booleana for verdadeira. A palavra reservada While sempre será seguida de um par de parênteses, que determinam a condição desta estrutura de repetição.  
      
    EX:   
    Console.log (Imprimindo de 1 a 5)  
      
    Var contador1 = 1;  
    While(contador1 <= 5) {  
    console.log(contador1);  
    contador1++  
    }  
      
    Obs: Foi usado um operador de incremento para que a variável não entrasse num looping infinito.   
      
      
    **Do While** – A estrutura de repetição Do-While é uma variação da estrutura While. Existe uma diferença sutil, porém importante, entre elas. Em While, a condição é testada antes da primeira execução das instruções que compõem seu corpo. Desse modo, se a condição for falsa na primeira vez em que for avaliada, as instruções não serão executadas nenhuma vez.  
    No Do-While, por outro lado, a condição somente é avaliada depois que suas instruções são executadas pela primeira vez, assim, mesmo que a condição seja falsa antes dele iniciar, suas instruções serão executadas pelo menos uma vez.  
      
    EX:   
      
    Console.log (Imprimindo de 1 a 5)  
      
    Var contador2 = 1;  
    do{ console.log(contador2);  
    contador2++;   
    }  
    While(contador2 <= 5)   
      
    **FOR** – O for é uma estrutura de repetição compacta. Seus elementos de inicialização, condição e interação são reunidos na forma de um cabeçalho e o corpo é disposto em seguida.   
      
    EX:  
      
    Console.log (‘Imprimindo de 1 a 5’)  
    for( var contador3 = 1; contador3 <= 5; contador3 ++){  
    Console.log(Contador3);  
    }