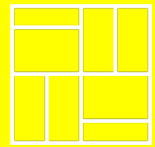
O que é Javascript?

Javascript é uma linguagem de programação interpretada estruturada, que juntamente com HTML e CSS faz parte das três principais tecnologias da World Wide Web.

Esta linguagem permite páginas da Web interactivas tornando-se assim uma parte essencial de tudo o que é desenvolvido para a web, sendo usada pela grande maioria dos sites.



O que é Javascript?

- É a linguagem de programação mais utilizada no mundo;
- É uma linguagem que pode ser usada tanto para frontend como para backend, através de ambientes como o node.js;
- No frontend o javascript tem como funcionalidade adicionar comportamento à página em que está inserido.

RESPONSABILIDADES:

- HTML define uma estrutura e o conteúdo
- CSS define os estilos de cada elemento
- JS adiciona comportamento à página

O que posso fazer com Javascript?

- Manipulação de HTML/DOM;
- Manipulação de CSS;
- Capturar e modificar eventos HTML;
- Efeitos e animações;
- AJAX;
- Plugins;
- Manipular e criar estruturas de dados;
- Aceder a ficheiros e base de dados;
- Gerar todo o tipo de conteúdo.

Conceitos Básicos

- Tag <script>
- Comentários
- Variáveis
- Tipos de dados
- Operadores

Tag <script>

O elemento HTML *<script>* é usado para incluir ou referenciar um script executável.

```
<script>
    // código js
    console.log(“Isto é código js!”)
</script>
```

SOURCE - SRC

Este atributo especifica o url do script externo, podendo ser usado como alternativa ao código directamente embebido no documento HTML.

```
<script src=”URL_DO_FICHEIRO.js”></script>
```

Comentários

```
// isto é um comentário de uma só linha
```

```
/* isto é um comentário de multi-linha que comenta tudo o que
está compreendido entre os asteriscos */
```

```
/*
        // Isto é
        // a combinação
        // de ambos!
*/
```

```
/* Isto já é um erro! /* Estes comentários não se combinam */ SyntaxError */
```

Variáveis

var x = 12

VAR

- Primeiro tipo de variável a ser criado em JS;
- Se inicializada sem valor, é atribuído o valor *undefined;*
- Valor da variável pode ser alterado;
- Declarar este tipo de variável em qualquer lugar no código é o equivalente a declarar no início - *hoisting*.

let x = 12

LET

- Desenvolvido como melhoria do var;
- Se inicializada sem valor, é atribuído o valor *undefined;*
- Valor da variável pode ser alterado;
- Esta variável pertence ao bloco de *scope* onde é declarada.

const x = 12

CONST

- Tem as mesmas características do tipo de variável *let*;
- A única e importante diferença: não pode ser alterada, definindo assim uma constante.

Tipos de Dados

NUMBER

Número inteiro ou decimal

1 1.2 10000

STRING

Sequência de caracteres

“Isto é uma string!”

BOOLEAN

Verdadeiro ou falso

true false

NULL

Valores desconhecidos

null

UNDEFINED

Valores não atribuídos

undefined

OBJECT

Estruturas de dados mais complexas.

```
{
    name : “Vitor”,
    lastName: “Marques”
}
```

Exercício

Declaração de variáveis e tipos

Usando as declaração de variáveis, cria na consola do inspector do browser uma variável para cada tipo de dado.

Para imprimires o valor que criaste na consola podes usar:
console.log(o_nome_da_tua_variável)

Para saberes o tipo da variável podes usar:
console.log(typeof o_nome_da_tua_variável)

Operadores

IGUAL ==

Retorna verdadeiro caso os operandos sejam iguais.

3 == "3" *var1 == 2*

NÃO IGUAL !=

Retorna verdadeiro caso os operandos não sejam iguais.

3 != "4" *3 != 2*

ESTRITAMENTE IGUAL ===

Retorna verdadeiro caso os operandos sejam iguais e do mesmo tipo.

3 === 3

ESTRITAMENTE NÃO IGUAL !==

Retorna verdadeiro caso os operandos não sejam iguais e/ou não sejam do mesmo tipo.

3 !== "3"

Operadores

MAIOR QUE >

Retorna verdadeiro caso o operando da esquerda seja maior que o da direita.

var2 > var1 *"12" > 2*

MAIOR QUE OU IGUAL >=

Retorna verdadeiro caso o operando da esquerda seja maior ou igual ao da direita.

var2 >= var1 *"12" >= 2* *2 >= 2*

MENOR QUE <

Retorna verdadeiro caso o operando da esquerda seja menor que o da direita.

var2 < var1 *2 < "12"*

MENOR QUE OU IGUAL <=

Retorna verdadeiro caso o operando da esquerda seja menor ou igual ao da direita.

var2 <= var1 *2 <= "12"* *2 <= 2*

Operadores

MÓDULO %

Retorna o inteiro restante da divisão dos dois operandos.

11 % 2 retorna 1

INCREMENTO ++

Adiciona um ao seu operando. Se usado como operador prefixado *++x*, retorna o valor de seu operando após a adição. Se usado como operador pós-fixado *x++*, retorna o valor de seu operando antes da adição.

let teste = 3
++teste retorna 4 *teste++ retorna 3*

DECREMENTO --

Subtrai um ao seu operando. O valor de retorno é igual àquele do operador de incremento.

let teste = 3
--teste retorna 2 *teste-- retorna 3*

Operadores

NEGAÇÃO/SUBTRAÇÃO -

Operador usado para proceder à subtração de um valor.
Retorna a negação de seu operando.

let y = 3 -2 // 1

let x = 3
-x retorna -3
-”1000” retorna -1000

ADIÇÃO +

Operador usado para proceder à soma de dois valores.
Tenta converter o operando em um número, sempre que possível.

let y = 3 +2 // 5

+"3" retorna 3.
+true retorna 1

Operadores

MULTIPLICAÇÃO *

Operador usado para proceder à multiplicação de dois valores.

*2 * 3 retorna 6*

DIVISÃO /

Operador usado para proceder à divisão de dois valores.

4 / 2 retorna 2

OPERADOR DE EXPONENCIAÇÃO **

Calcula a base elevada à potência do expoente, que é, $base^{expoente}$

*2 ** 3 retorna 8*

Operadores Lógicos

E / AND &&

Retorna verdadeiro caso ambos operandos sejam verdadeiros; caso contrário, retorna falso

true && true
"Gato" && "Cão" retorna "Cão"

OU / OR ||

Retorna verdadeiro caso ambos os operandos sejam verdadeiro; se ambos forem falsos, retorna falso

false || true retorna true
"Gato" || "Cão" retorna “Gato”

NÃO / NOT !

Retorna falso caso o único operando possa ser convertido para verdadeiro; senão, retorna verdadeiro.

!true retorna false
!false retorna true

Exercício

Operadores

Usando alguns dos operadores e também a declaração de variáveis, cria um script ou ficheiro js para testares os conceitos descritos anteriormente: comentários, variáveis, operadores e tipos de dados

Para imprimires o valor que criaste na consola podes usar:

console.log(o_nome_da_tua_variável)

Para saberes o tipo da variável podes usar:

typeof o_nome_da_tua_variável

ESTRUTURAS DE DECISÃO

Parte das funcionalidade de todas as linguagens de programação é a capacidade da mesma poder executar diversos blocos de código dependendo de alguma condição.

Para esse efeito o javascript dispõe principalmente de duas estruturas de tomada de decisão.

ESTRUTURAS DE DECISÃO

- IF / ELSE IF / ELSE
- SWITCH / CASE

IF / ELSE IF / ELSE

IF

Executa alguma declaração caso a condição lógica for verdadeira

```
if(n < 10) {
    console.log("Este número é pequeno");
```

ELSE IF

Combina declarações, para obter várias condições testadas em sequência

```
} else if(n >= 10 && n < 30){
    console.log("Este número é médio");
```

ELSE

Executa alguma declaração caso a condição lógica for falsa

```
} else {
    console.log("Este número é alto");
}
```

SWITCH / CASE

Uma declaração switch permite que um programa avalie uma expressão e tente associar o valor da expressão ao rótulo de um case. Se uma correspondência é encontrada, o programa executa a declaração associada.

```
switch (expressao) {
        case rotulo_1:
                declaracoes_1
                [break;]
        case rotulo_2:
                declaracoes_2
                [break;]
        ...
        default:
                declaracoes_padrao
                [break;]
}
```

SWITCH / CASE

CASE

Determina cada cláusula a ser avaliada pelo switch. Um case deverá conter um rótulo que corresponda ao valor da expressão a ser avaliada.

DEFAULT

Se nenhuma cláusula é encontrada, o programa determina que a cláusula default será a verdadeira.
Não é obrigatória a sua definição, e por convenção encontra-se no final do switch (não é obrigatório).

BREAK

A instrução break associada a cada cláusula case, garante que o programa sairá do switch assim que a declaração correspondente for executada.

Exercício

Estruturas de Decisão

Usando as estruturas de decisão anteriormente descritas, testa a implementação de cada uma delas: if / else if / else e switch case.

Escreve uma porção de código que quando o valor de n é:

- 1 faça log de ‘Eu sou o único’.
- 2 faça log de ‘Somos pares’.
- 3 faça log de ‘Somos um trio’.

Antes de escreveres o código inicializa a tua variável (p.e):

let n = 1

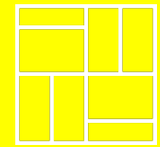
Para imprimires o valor que criaste na consola podes usar:

console.log(o_texto_que_queres_imprimir)

EDIT.

ESTRUTURAS DE REPETIÇÃO

- FOR
- WHILE
- DO / WHILE



FOR

Um ciclo for é repetido até que a condição especificada seja falsa.

```
for(let i = 0; i < 10; i++) {
        console.log(i);
}
```

Um ciclo for é composto por 3 secções

for ([Expressão Inicial]; [Condição]; [Incremento])

EXPRESSÃO INICIAL

Expressão que inicializa um ou mais contadores. Pode conter inicialização da variável.

CONDIÇÃO

Expressão que é avaliada: caso o resultado da condição seja verdadeiro, o ciclo é executado; se o valor da condição for falso, então o ciclo terminará.

INCREMENTO

Expressão que faz actualizar o valor a ser avaliado pela condição do ciclo.

WHILE

Um ciclo while é sempre executado, enquanto a condição especificada seja avaliada como verdadeira.

```
let j = 0;
while(j < 10) {
        console.log(j);
        j++;
}
```

Um ciclo while é composto por 2 secções

```
while (Condição)
        Declaração
```

CONDIÇÃO

Expressão que é avaliada: caso o resultado da condição seja verdadeiro, o ciclo é executado; se o valor da condição for falso, então o ciclo terminará.

DECLARAÇÃO

Código a ser executado se a condição for verdadeira. É aqui que a condição deve ser actualizada.

DO / WHILE

Um ciclo do/while é sempre executado, enquanto a condição especificada seja avaliada como verdadeira. A condição é avaliada depois que o bloco de código é executado, resultando que uma declaração seja executada pelo menos uma vez.

```
let j = 0;
do {
        console.log(j);
        j=j+1;
    } while(j < 10)
```

Um ciclo do/while é composto por 2 secções

```
do
        Declaração
while (Condição);
```

CONDIÇÃO

Expressão que é avaliada: caso o resultado da condição seja verdadeiro, o ciclo é executado; se o valor da condição for falso, então o ciclo terminará.

DECLARAÇÃO

Código a ser executado se a condição for verdadeira. É aqui que a condição deve ser actualizada.

Exercício

Estruturas de Repetição

Usando as estruturas de decisão anteriormente descritas, testa a implementação de cada uma delas: for, while e do while,

Escreve uma porção de código que imprime os números pares entre 0 a 10.

Dicas:

- Dentro da tua estrutura de decisão usa um if para diferenciar números pares.
- Para saberes se um número é par, podes usar o resto da divisão por dois ser 0.
  Vê novamente os operadores.