Fundamentos de Programación

JavaScript

Fundamentos

IES La Nía

JavaScript

- Lenguaje de programación de *script*
- Incrustado en los navegadores web
- Interpretado, no se compila
- Sencillo, pero potente
- Soporta orientación a objetos

Uso en el Navegador

- Dentro de un documento HTML, el código JavaScript se escribe entre etiquetas `<script>` y `</script>`
  - En el `head` o antes de cerrar `body`

- Podemos probarlo con las **DevTools:** Herramientas del Desarrollador y/o Consola
  - Mejor en *Firefox* o *Chrome*
- O mediante jsbin.com

Ejemplo

```
<!DOCTYPE html>
<html lang="es">
<head>
	<title>Hola Aspe</title>
	<meta charset="utf-8" />
	<script>
		console.log("Hola Aspe desde la consola");
		alert("Hola Aspe desde alert");
	</script>
</head>
<body></body>
</html>
```

Variables

- Almacenan un dato
- Se definen con la *keyword* **`var`**
- Se le puede asignar valores
- Si olvidamos **`var`**, el programa no falla, pero puede provocar un comportamiento inesperado
  - *Hoisting*
- Las variables pueden ser de tipo entero, flotante, cadena, booleano u objeto/array

Identificadores

- Es el nombre de la variables
- Los tipos básicos se escriben en minúsculas. Si es un objeto, comienza por mayúscula
- Las constantes en MAYÚSCULAS
- Notación *camelCase*

```
var dato;
var datoConVariasPalabras;
var PI = 3.1415;
var Clase = new Object();
```

Instrucciones

- Cada instrucción se separa con `;`
- Puede contener asignaciones, comentarios e instrucciones de:
  - Control / condicionales
  - Iterativas / bucles
  - Tratamiento de excepciones

Asignación

- Al declarar una variable no se le asigna ningún tipo. Se le asigna al asignarle un valor

```
var num = 4;
var edad = 33;
var suma = num + edad;
var resta = edad - num;
var producto = num * edad;
var division = num / edad;
var resto = edad % num;
var formula = num * 2 + ((edad - 2) / 2);
```

Comentarios

- Permiten introducir texto en mitad del código y que no se interprete

- `//` → comentarios de una línea

- `/*`

  → comentarios de varias líneas

  `*/`

Cadenas

- Entre comillas dobles o simples

```
"" // cadena vacía
'probando'
"3.14"
'nombre="miFormulario"'
"comemos en el McDonald's"
```

- Para concatenar cadenas → operador **+**
- La propiedad **`length`** permite averiguar el tamaño de una cadena.

```
var msj = "Hola " + "Mundo";
var tam = msj.length;
```

Números

- Enteros y reales

```
var diez = 10; // entero
var pi = 3.14; // real
```

- Operaciones: la suma **+**, la resta **-**, el producto *****, la división **/** y el resto **%**
- Operadores: **++** (incremendo), **--** (decremento)
- Objeto **Math** permite otras operaciones como la potencia, redondear, obtener el mayor o el menor, etc..

Operaciones aritméticas

Suma	+	Resta	-
Multiplicación	*	División	/
Resto	%		

Operador	Expresión	Equivalencia	Descripción
+=	`x += y`	`x = x+y`	Incrementa `x` en `y` unidades
-=	`x -= y`	`x = x-y`	Decrementa la `x` en `y` unidades
*=	`x *= y`	`x = x*y`	Multiplica el valor de `x` por `y`
/=	`x /= y`	`x = x/y`	Divide el valor de `x` por `y`
%=	`x %= y`	`x = x %y`	Asigna el resto de `x/y` a `x`
++	`x++`	`x = x + 1`	Incrementa `x` en uno
--	`x--`	`x = x - 1`	Decrementa `x` en uno

Funciones predefinidas para mensajes → leer/escribir

- **`alert`**`(mensaje);` → muestra un mensaje
- `var respuesta =` **`prompt`**`(mensaje);` → muestra un mensaje y recupera un valor de tipo texto
- `var respuesta =` **`confirm`**`(mensaje);` → muestra un mensaje, recupera un valor *booleano* con `true` si OK, o `undefined` si cancela;

Consola

- Para mostrar la información a la hora de probar una aplicación, es más cómodo trabajar con la consola.

  `console.log(mensaje)` → muestra mensaje

  `console.error(mensaje)` → muestra mensaje en rojo

  `console.clear()` → limpia la pantalla

Ejercicios

0) (`ej100holaMundo`) Muestra por pantalla un mensaje con “Hola Mundo”

   (`ej100R1`) Tras leer un nombre, saludar a dicho nombre

   (`ej100R2`) No utilizar ninguna variable

Planteamiento de problemas

- Todo programa tiene:
  - entrada de datos
  - procesamiento
  - salida

- Sustantivos → entrada, salida
- Verbos → procesamiento

Ejercicios

1) (`ej101cadena`) A partir de una frase, mostrar cuantos caracteres contiene.
2) (`ej102madlib`) A partir de un nombre, un verbo, un adjetivo y un adverbio, crea una historia que contenga dichos elementos:

   Entrada: perro / caminar / azul / rápidamente

   Salida: ¿ Te gusta caminar con tu perro azul rápidamente ?

   1) (ej102R) Crea un *mad libs* más extenso.

Funciones de transformación de texto a número

- `función` **`parseInt`**`(dato);` → De tipo cadena (u otro) a entero
- `función` **`parseFloat`**`(dato);` → De cadena (u otro) a real

```
var cadena = "3.14";
var pi = parseFloat(cadena);
var tres = parseInt(pi);
var suma = pi + tres;
```

Ejercicios

3) (`ej103edad`) A partir de una edad, mostrar la edad que tendrá dentro de 10 años y hace 10 años.
4) (`ej104precio`) A partir del precio base, calcular el precio de venta de un producto sabiendo que tiene unos impuestos del 21% y 3 euros de gastos de envío

Ejercicios

5) (`ej105mates`) A partir de dos números, muestra todas las operaciones matemáticas sencillas (+, -, *, /, %).
6) (`ej106impuestos`) A partir de una cantidad y un porcentaje de IVA, mostrar su IVA y el precio total.
7) (`ej107dinero`) A partir de una cantidad de dinero, mostrar su descomposición en billetes (500, 200, 100, 50, 20, 10, 5) y monedas (2, 1), para que el número de elementos sea mínimo.

Ejercicios

8) (`ej108jubilacion`) Crea una calculadora para jubilación, de manera que a partir de una edad y la edad de jubilación, muestre cuantos años quedan para jubilarse y el año de jubilación.

```
var ahora = new Date();
var anyo = ahora.getFullYear();
```

Booleanos

- Permite guardar valores verdaderos o falsos, es decir, **`true`** o **`false`**.
- Los operadores que trabajan con booleanos son la conjunción **`&&`**, la disyunción **`||`** y la negación `!`.
- Además, tenemos el operador ternario:

```
(cond) ? valorVerdadero : valorFalso.
```

```
var esBooleano = true;
```

Tablas lógicas

A	*B*	***!****A*	*A* **&&** *B*	*A* **||** *B*
false	false	true	false	false
false	true	true	false	true
true	false	false	false	true
true	true	false	true	true

Conversión de tipos

- Si sumamos un número y un texto, el número se pasa a texto y se concatena el resultado.
  - Si pasamos un texto incorrecto a número → `NaN` (*Not a Number*)
  - `isNaN(dato)` → `true` si el dato no es numérico

- Si un valor es `0, -0, null, false, NaN, undefined,` o una cadena vacía (`""`), entonces es falso.

Comparando Valores

- Para comparar valores haremos uso de
  - **==** → igualdad con conversión de tipos
  - **===** → igualdad sin conversión de tipos
  - **!=** → diferencia con conversión de tipos
  - **!==** → diferencia sin conversión de tipos
- Devuelven `true` o `false`

```
var verdadero = ("1" == true);
var falso = ("1" === true);
```

Ejercicios

9) (`ej109iguales`) Crea un programa que a partir de dos números, indique mediante un *booleano* si son iguales.

10) (`ej110numero`) Crea un programa que lea un dato e indique mediante un *booleano* si es un número.

11) (`ej111par`) Crea un programa que a partir de un número, indique si es par mediante un booleano

Función

- Fragmento que permite reutilizar código.
- Son como pequeños programas dentro del programa principal
- Se declaran una vez y se invoca muchas veces.
- Reciben los datos por los parámetros, separados por comas
  - Pueden existir funciones sin parámetros
- Devuelven un sólo valor mediante `return`
  - Si la función no hace `return`, el valor es `undefined`

Función – Declaración e Invocación

- Declaración:
  - **`function`** *`nombreFuncion(parámetros)`* `{`

```
    // código de la función
    return valor;
}
```

```
1 function saluda(nombre) {
2   return "Hola" + nombre;
3 }
4
5 saluda("Adrián");
6 saluda("Marina");
```

- Invocación
  - `nombreFuncion(argumentos)`
    - `argumentos` separados por comas, pueden o no coincidir con los parámetros

Ejercicios

12) (`ej112fdoble`) Crea una función que devuelva el doble de un número.

13) (`ej113fsuma`) Crea una función que sume dos números.

14) (`ej114fpar`) Crea una función llamada `esPar` que devuelva un booleano indicando si el número recibido es par.

15) (`ej115fimpar`) Crea una función llamada `esImpar` que utilizando la función anterior devuelva si un número es impar.

Alcance

- Las variables definidas fuera de las funciones tienen alcance **global** → accesibles desde cualquier función
- Los parámetros de una función y las variables declaradas dentro de una función (se conocen como variables locales) sólo son accesibles desde dentro de la misma función → alcance de **función**

- **No** usar variables globales dentro de una función

Ejemplo Alcance

```
var global = 10;

function saluda() {
  var local = 5;

  console.log("Dentro de saluda local:" + local);
  console.log("Dentro de saluda global:" + global);

  return "Hola";
}

console.log(global);
console.log(local); // error
console.log(saluda());
```