CIP FP cecheste

Desarrollo Web en Entorno Cliente

Javascript

Sintaxis [1]

Cosas a tener en cuenta:

1. No se tienen en cuenta los espacios en blanco.
2. Es "**key sensitive**".
3. No se definen el tipo de variable.
4. No es necesario terminar cada instrucción con ";"-

¿Cómo lo incluímos dentro del código HTML?

1. En un elemento <script></script>

```
<html>
    <head>
        <title>prueba</title>
        <script type="text/javascript">
          /*Código*/
        </script>
    </head>
    <body>
    </body>
</html>
```

¿Cómo lo incluímos dentro del código HTML?

2. Como atributo en cualquier elemento HTML.

```
<html>
    <head>
        <title>prueba</title>    </head>
    <body>
        <div onclick="/*Código*/"> … </div>
    </body>
</html>
```

¿Cómo lo incluímos dentro del código HTML?

3. En un archivo con extensión ".js".

```
<html>
    <head>
        <title>prueba</title>
        <script type="text/javascript" src="archivo.js">
        </script>
    </head>
    <body>
    </body>
</html>
```

Comentarios

```
/*
    Comentario para bloques
    Línea 1
    Línea 2
    Línea 3
*/

// Comentario de una sola línea
```

Variables

1. El primer carácter no puede ser un número.
2. Sólo puede estar formado por letras, números y los símbolos $ (dólar) y _ (guión bajo).

Variables

Nombre	Descripción
var	Declara una variable, iniciándola opcionalmente a un valor. Ámbito global. *Si se inicializa una variable sin indicar nada previamente, se considerará **var***.
let	Declara una variable local en un bloque de ámbito, iniciándola opcionalmente a un valor.
const	Declara una constante de solo lectura en un bloque de ámbito.

Variables

Datos primitivos y básicos
Undefined
Boolean
Number
string
BigInt (Xn)
null
Object
Function

Hoisting

Las variables en **JavaScript** se pueden referenciar antes de ser declaradas sin obtener una excepción.

¿Con qué tipo de declaración de variables conseguimos esto?

var

Hoisting

```
console.log(variable);
```

¿Resultado? Excepción

```
console.log(variable);
let variable = 0;
```

¿Resultado? Excepción

```
console.log(variable);
var variable = 0;
```

¿Resultado? Undefined

Ámbito de las variables

Nombre	Descripción
Globales	Cuando declaras una variable en la raíz de un documento o dentro de un bloque (no función) mediante **var**.
Ámbito de bloque	Dentro de un bloque mediante **let**.

Operadores

1. Unarios

```
X++;
```

1. Binarios

```
1 + 2;
```

1. Ternarios

```
condición ? SI : NO
```

Operadores
Asignación

Nombre	Abreviatura	Operador
Asignación	x = y	x = y
Adición	x += y	x = x + y
Sustracción	x -= y	x = x - y
Multiplicación	x *= y	x = x * y
División	x /= y	x = x / y
Resto	x %= y	x = x % y
Exponenciación	x **= y	x = x**y

¿Cómo funcionan los operadores bit a bit (desplazamientos)? ¿Y los operadores de asignación lógicos? ¿Cuándo se podrían usar?

Operadores
Comparación

Nombre	TRUE	FALSE
Igualdad	1 == 1; 1 == "1"	1 == 2
Desigualdad	1 != 2	1 != 1
Estrictamente iguales	1 === 1	1 === 2; 1 === "1"
Estrictamente desiguales	1 !== 2; 1 !== "1"	1 !== 1
Mayor qué	2 > 1	2 > 3; 2 > 2
Mayor o igual qué	2 >= 1; 2 >= 2	2 >= 3
Menor qué	1 < 2	1 < 1; 1 < 0
Menor o igual qué	1 <= 2; 1 <= 1	2 <= 1

Operadores
Aritméticos

Nombre	Operador	Ejemplo
Suma	+	1 + 2
Resta	-	3 - 1
Multiplicación	*	2 * 3
División	/	4 / 2
Resto	%	4 % 2
Potencia	**	2**3
Incremento (unario)	++	2++; ++2
Decremento (unario)	--	2--; --2

¿Cuál es la diferencia del operador unario antes o después de la variable?

Operadores
Lógicos

Nombre	Operador	Ejemplo
AND	&&	true && false
OR	||	true || false
NOT	!	!true

¿Qué resultado de operaciones o contenido de variables serían interpretados como `FALSE`?

FALSE

FALSE
False
undefined
Null
0
NaN
Cadena vacía (“”)

Operadores
Otros

Nombre	Definición	Ejemplo
typeof	Devuelve el tipo de operando	typeof dato
instanceof	Devuelve TRUE si el operando es del tipo indicado	operando instanceof tipo
Agrupación	Para agrupar expresiones matemáticas.	2 * (3 + 2)

```
let variable = "3";
```

¿Qué devolvería `typeof variable`? y, ¿`variable instanceof Number`?

Estructuras
Condicionales. If

```
if (condicion) {
   sentencia_1;
} else {
   sentencia_2;
}
```

¿Cómo encadenariámos condicionales de tipo `if`?

```
if (condicion_1) {
   sentencia_1;
} else if (condicion_2) {
   sentencia_2;
} else if (condicion_n) {
   sentencia_n;
} else {
   sentencia_final;
}
```

Estructuras

Condicionales. Switch

```
switch (expression) {
    case label_1:
       statements_1
        [break;]
    case label_2:
       statements_2
        [break;]
        …
    default:
       statements_def
}
```

Interacción provisional
Declaración

alert()

```
let mensaje = "Cuidado con el perro";
alert(mensaje);
```

confirm()

```
let mensaje = "¿De verdad quieres eliminarlo?";
let respuesta = confirm(mensaje);
alert(respuesta);
```

prompt()

```
let mensaje = "¿Cómo te llamas?";
let nombre = prompt(mensaje);
alert(nombre);
```

Ejercicios

1. Escribir un programa que pregunte al usuario su edad y muestre por pantalla si es mayor de edad o no.

2. Escribir un programa que almacene la cadena de caracteres contraseña en una variable, pregunte al usuario por la contraseña e imprima por pantalla si la contraseña introducida por el usuario coincide con la guardada en la variable.

3. Escribir un programa que pida al usuario dos números y muestre por pantalla su división. Si el divisor es cero el programa debe mostrar un error.

4. Escribir un programa que pida al usuario un número entero y muestre por pantalla si es par o impar.

5. Para tributar un determinado impuesto se debe ser mayor de 16 años y tener unos ingresos iguales o superiores a 1000 € mensuales. Escribir un programa que pregunte al usuario su edad y sus ingresos mensuales y muestre por pantalla si el usuario tiene que tributar o no.

6. Los alumnos de un curso se han dividido en dos grupos A y B de acuerdo al sexo y el nombre. El grupo A esta formado por las mujeres con un nombre anterior a la M y los hombres con un nombre posterior a la N y el grupo B por el resto. Escribir un programa que pregunte al usuario su nombre y sexo, y muestre por pantalla el grupo que le corresponde.

Arrays
Declaración

```
let arr0 = new Array(element0, element1, ..., elementN);
let arr1 = Array(element0, element1, ..., elementN);
let arr2 = [element0, element1, ..., elementN];
```

Los elementos que lo conforman
no tienen por qué ser homogéneos

Matrices
Declaración

```
let arr0 = new Array(element0, element1, ..., elementN);
let arr1 = Array(element0, element1, ..., elementN);
let arr2 = [element0, element1, ..., elementN];
```

¿Cómo se declarará una matriz?

```
/* Matriz */
let arr2 = [arr, arr1,..., arrN];
```

Además de la heterogeneidad, ¿qué otra característica tienen las matrices?

Métodos de array

¿Cuándo y por qué decimos que una función es destructiva?

Métodos de array

push()

```
frutas = ["manzana", "banana"];
let nuevaLongitud = frutas.push('Naranja');
//Inserta elemento al final
// ["Manzana", "Banana", "Naranja"];
```

Añade un o más elementos al final del array.
Es destructivo. Devuelve la nueva longitud del array

¿Para qué estructura de datos sería interesante su uso?

Métodos de array

pop()

```
let ultimo = frutas.pop(); // elimina y devuelve último elemento
// ["Manzana", "Banana"];
```

Elimina el último elemento del array.
Es destructivo. Devuelve el elemento eliminado

¿Para qué estructura de datos sería interesante su uso?

Métodos de array

shift()

```
let primero = frutas.shift(); // elimina y devuelve primer elemento
// ["Banana"];
```

Igual que pop() pero con el primer elemento.

¿Para qué estructura de datos sería interesante su uso?

Métodos de array
unshift()

```
let nuevaLongitud = frutas.unshift('Fresa'); // añade al inicio
// ["Fresa", "Banana"];
```

Igual que push() pero al principio del array

¿Para qué estructura de datos sería interesante su uso?

Métodos de array
indexOf()

```
frutas.push('Mango');
// ["Fresa", "Banana", "Mango"];
let pos = frutas.indexOf('Banana'); // 1
```

Devuelve el índice de la primera aparición de un valor en el array. Si no existe, devuelve -1.

Métodos de array
lastIndexOf()

```
frutas.push('Banana');
// ["Fresa", "Banana", "Mango", "Banana"];
let pos = frutas.indexOf('Banana'); // 3
```

Devuelve el índice de la última aparición de un valor en el array. Si no existe, devuelve -1.

Métodos de array

splice()

```
frutas.push('Banana');
// ["Fresa", "Banana", "Mango", "Banana"];
let pos = 1;
let cantidad = 1;
let elementoEliminado = frutas.splice(pos, 1, 4, 5); // ["Banana"];
// ["Fresa", "Mango", "Banana", 4, 5];
```

Devuelve y elimina un elemento o elementos desde pos y tantas veces como indiquemos en el segundo parámetro, y los sustituye por los parámetros posteriores.

Métodos de array

slice()

```
let copiaSuperficial = frutas.slice(); // ["Fresa", "Mango"];
```

Igual que `splice` pero sin alterar el original y sin sustituir elementos.

Métodos de array

length()

```
frutas.length; // 2
```

Devuelve el número de elementos del array.

Métodos de array

sort()

```
numeros = [2, 1, 3]
numeros.sort();
console.log(numeros); // [1, 2, 3]
```

Ordena del array.
Es destructivo.

Métodos de array
includes()

```
numeros = [2, 1, 3]
numeros.includes(4);
console.log(numeros); // False
```

Te indica si un elemento está o no en el array.

Métodos de array

reverse()

```
numeros = [2, 1, 3]
numeros.reverse();
console.log(numeros); // [3, 2, 1]
```

Ordena del array en orden descendente.
Es destructivo.

Métodos de array

concat()

```
let numeros = [1, 3, 2];
let pescado = ["mero", "sepia"];
let cosas = numeros.concat(pescado); //Devuelve la
concatenación de los dos arrays
console.log(cosas); //[1, 3, 2, "mero", "sepia"]
```

Concatena dos arrays.
No es destructivo.

Métodos de array
join()

```
let numeros = [1, 3, 2];
let cosas = frutas.join("-"); //Devuelve los elementos separados por el
símbolo indicado en una cadena, por defecto ",".
console.log(cosas); //1-3-2
```

Devuelve todos los elementos de un array
separados por el carácter que indiquemos.
Por defecto será ","

¿Qué pasaría con `toString()`? ¿Tiene un comportamiento similar a `join()` sobre un array? En ese caso, ¿en qué se diferencian?

Arrays
Ejercicios

1. Crear un programa en el que guardes los días de la semana en un array y se le pida al usuario un número entre el 1 y el 7, devolviéndole el nombre del día correspondiente.
2. ~~Realiza un programa al que le pasamos una cadena formada por números separados por asteriscos, y te la devuelva ordenada de mayor a menor y en el mismo formato~~. (PARA CAMBIAR)

Continuará…

String

1. Valor literal.

```
let variable1 = "la Senia";
console.log(typeof variable1);  //string
```

2. Objeto String.

```
let variable2 = new String('la Senia');
console.log(typeof variable2);  //Object
```

¿Qué tipo de dato contendrán `variable1` y `variable2`?

Anticipo…
eval()

¿Conoces la función `eval()`?

Si le pasamos un string que contiene una operación matemática, la calcula y nos devuelve el resultado.

String

1. Valor literal.

```
let variable1 = '2 + 2'; // literal
```

```
eval(variable1);
```

```
// Devuelve 4
```

¿Qué devolverá esta función?

2. Objeto String.

```
let s2 = new String('2 + 2'); // objeto
```

```
eval(s2);
```

```
// Devuelve "2 + 2"
```

¿Qué devolverá esta función?

¿Cuál de los dos es el comportamiento deseado?

String

Podemos acceder a los caracteres como si fuera un array, pero no podemos modificarlos.

```
let mystring = "Hello, World!";
let x = mystring.length;
mystring[0] = 'L' // No produce ningún efecto
mystring[0]; // Esto devuelve "H"
```

El símbolo de escape `\n`, solo funcionará cuando queramos usar la consola javascript del navegador, no en la web. Para ello, deberemos utilizar el elemento HTML correspondiente, `<br>`.

String
concatenar

Para concatenar strings tenemos dos formas:

```
console.log("texto1 " + variable1 + ' ' + variable2);
```

```
console.log(`texto1 ${variable1} ${variable2}`);
```

String
Métodos

Los métodos de la clase String pueden emplearse con un string literal.

Método	Descripción
charAt, charCodeAt, codePointAt	Devuelve el carácter o el código del carácter en la posición especificada en la cadena.
indexOf, lasIndexOf	Devuelve la posición de la subcadena en la cadena o la última posición de una subcadena especificada.
startsWith, endsWith, includes	Devuelve si la cadena empieza, termina o contiene una cadena especificada, o no.
fromCharCode, fromCodePoint	Construye una cadena desde la secuencia de valores Unicode especificada. Tabla Unicode
split	Divide un objeto String en un array de strings separados por substrings. También se puede usar RegExp
slice	Extrae una sección de un string en un array de strings separados por substrings.
substring, substr	Devuelve un substring del string, especificando índice inicial y el índice final.
match, replace, search, test,	Para trabajar con expresiones regulares.
toLowerCase, toUpperCase	Devuelve el string en mayúsculas o minúsculas
repeat	Devuelve un string formado por los elementos del objeto repetidos el número de veces que le indiquemos.
trim	Elimina los espacios en blanco del principio y final del string.

String

Expresiones regulares

Conocidas como RegExp, son un sistema para buscar, capturar o reemplazar texto utilizando patrones. Expresiones regulares

1. ¿Cuál sería la expresión regular para una IP?
2. Cambia los puntos de una cadena por "_".
3. Ayudándote de una expresión regular, cuenta el número de "a" que hay en una cadena.

Para poder probarlo haz uso de los métodos correspondientes. Para introducir y mostrar datos utiliza las funciones `prompt()` y `alert()` respectivamente.

`match, replace, search, test, …`

Number

Sistemas numéricos

En Javascript todos los números están en doble precisión

1. Binario

```
let binario = 0b101;
console.log(binario);
```

2. Octal

```
let octal = 010;
console.log(octal);
```

3. Hexadecimal

```
let hexadecimal = 0xff;
console.log(hexadecimal);
```

Number
Propiedades

Propiedad	Descripción
Number.MAX_VALUE	El número más grande representable.
Number.MIN_VALUE	El número más pequeño representable.
Number.NaN	Valor especial. "No es un número".
Number.NEGATIVE_INFINITY	Valor especial. Infinito negativo.
Number.POSITIVE_INFINITY	Valor especial. Infinito positivo.
Number.MIN_SAFE_INTEGER	Número entero mínimo de seguridad en JavaScript.
Number_MAX_SAFE_INTEGER	Número entero máximo de seguridad en Javascript.

Number
Métodos

Método	Descripción
Number.parseFloat()	Analiza el string que pasamos y devuelve un número en coma flotante. Igual que la función `parseFloat()`.
Number.parseInt()	Analiza el string que pasamos y devuelve un número entero. Igual que la función `parseInt()`.
Number.isFinite()	Determina si el valor pasado es un número finito. Igual que la función `isFinite()`.
Number.isInteger()	Determina si el valor pasado es un entero.
Number.isNaN()	Determina si el valor pasado no es un número. Igual que la función `isFinite()`.
Number.isSafeInteger()	Determina si el valor proporcionado es un número entero seguro.

¿Qué diferencia existe entre un método y una función?

Math
Propiedades

Propiedad	Descripción
Math.E	Constante de Euler, la base de los logaritmos naturales, aproximadamente 2,718
Math.LN2	Logaritmo natural de 2, aproximadamente 0,693
Math.LN10	Logaritmo natural de 10, aproximadamente 2,303
Math.LOG2E	Logaritmo de E con base 2, aproximadamente 1,443
Math.LOG10E	Logaritmo de E con base 10, aproximadamente 0,434
Math.PI	La relación entre la longitud de una circunferencia y su diámetro.
Math.SQRT1_2	Raíz cuadrada de ½.
Math.SQRT2	Raíz cuadrada de 2.

Math
Métodos

Método	Descripción
Math.abs()	Valor absoluto
Math.min()	El menor de una serie de números separados por comas
Math.max()	El mayor de una serie de números separados por comas
Math.random()	Un número aleatorio de 0 a 1
Math.round()	Te redondea un número
Math.trunc()	Te trunca un número
Math.sqrt()	Calcula la raíz cuadrada de un número
Math.sign()	Te devuelve 1 o -1 en función del signo de un número.

1. Realiza un programa que te pida el diámetro de una circunferencia y te devuelva el área.

Date

1. Declaración

```
let d = new Date(); //Hoy
let d = new Date(milliseconds); //A contar desde 01/01/1970
let d = new Date(dateString); //"01/12/1978"; "1978/01/12"
let d = new Date(year, month, day, hours, minutes, seconds, milliseconds);
```

1. Métodos estáticos

```
Date.now(); // Fecha actual en milisegundos
Date.parse(); // Date.parse('01 Jan 1970 00:00:00 GMT'); // Devolvería 0 ms.
Date.UTC(); // Date.UTC(1970, 0, 1, 0, 0, 0)); // Devolvería 0
```

Métodos y Funciones de Date

Estructuras
Repetitivas. while

```
while (condition)
    //instrucciones
...
let n = 0;
let x = 0;
while (n < 3) {
    n++;
    x += n;
}
```

Estructuras

Repetitivas. do … while …

```
do {
    //instrucciones
} while (condición);
...
let n = 0;
let x = 0;
do {
    n++;
    x += n;
} while (n < 3);
```

Estructuras
Repetitivas. for

```
for ([expresión inicial]; [condición]; [expresión de incremento]){
   //instrucciones
}

…

for (let i = 10; i >= 1; i++){
   //instrucciones
}
```

Estructuras

Repetitivas. for … in …

```
for (variable en objeto) {
   //instrucciones
}

…

let resultado = "";
for (let i in obj) {
    resultado += obj + "." + i + " = " + obj[i] + "<br>";
    console.log(resultado);
}
```

Analiza el funcionamiento del bucle

Estructuras

Repetitivas. for … of …

```
for (variable de objeto) {
   //instrucciones
 }
…
let arr = [3, 5, 7];
arr.foo = "hello";
for (let i in arr) {
  console.log(i); // logs "0", "1", "2", "foo"
}
for (let i of arr) {
  console.log(i); // logs "3", "5", "7", “hello”
}
```

Analiza el funcionamiento del bucle

Estructuras
Repetitivas

Continuará…

Alteración del flujo

break

Finaliza el código encerrado en una estructura repetitiva o condicional.

```
switch (expression) {
    case label_1:
       statements_1
        [break;]
        …
    default:
       statements_def;
}
```

Alteración del flujo

continue

Finaliza la iteración en curso y pasa a la siguiente.

```
let i = 0;
let n = 0;
while (i < 5) {
    i++;
    if (i == 3) {
        continue;
    }
    n += i;
}
```

Alteración de flujo

label

Proporciona una sentencia con un identificador que se puede referir al usar las sentencias `break` y `continue`.

```
let i = 0;
let n = 0;
bucle: while (true) {
    i++;
    if (i == 3) {
        break bucle;
    }
    n += i;
}
```

Ejercicios

1. Programa al que le pasemos una fecha en formato “12/01/1978” y nos la devuelva en formato “12 de enero de 1978”.

2. Realiza un juego para adivinar un número. El número a adivinar se escoge de modo aleatorio entre 1 y 1000, y después el usuario debe adivinar el número probando continuamente. El programa le indicará si el número elegido por el usuario es mayor o menor al número que hay que adivinar.

3. Escribir un programa que pida caracteres y muestre ‘VOCAL’ si son vocales y ‘NO VOCAL’ en caso contrario, el programa termina cuando se introduce un espacio.

4. Desarrolla un programa que pida el límite inferior y superior de un intervalo. Si el límite inferior es mayor que el superior lo tiene que indicar y volver a pedir. A continuación, se van introduciendo números hasta que introduzcamos el 0. Cuando termine el programa dará las siguientes informaciones:
   - La suma de los números que están dentro del intervalo (intervalo abierto).
   - Cuantos números están fuera del intervalo.
   - Informa si hemos introducido algún número igual a los límites del intervalo.

5. Realiza un programa al que le pases los colores RGB en decimal, y te devuelva la cadena hexadecimal en el formato de los colores seguros. #xxyyzz

6. Realiza un programa al que le pasamos una cadena formada por números (unidades, decenas y centenas) separados por asteriscos, y te la devuelva ordenada de mayor a menor y en el mismo formato.