UD 2 – INTRODUCION AL LENGUAJE JAVASCRIPT

Índice

2.1 Reglas para los identificadores de variables.....1
2.2. Declarar variables: var, let y const.....2
2.3. Identificar un tipo de dato.....3
2.4. Tipo de dato: undefined.....3
2.5. Tipos de datos: Diferencias en Undefined y Null.....3
2.6. Tipos de datos: String.....7
2.7. Tipos de Datos: Numéricos.....8
2.8. Comparaciones y operadores lógicos.....9
2.9. Comentarios.....10
2.10. Mensajes de confirmación al visitar Páginas Web.....10
2.11. Mejorar la experiencia del Usuario.....11
2.12. Ejercicios.....11
2.13 Ejercicios Ampliación.....14

2.1 Reglas para los identificadores de variables

En Javascript no podemos declarar el tipo de dato de nuestra variable como en otros lenguajes de programación. Por tanto, es el dato almacenado por la variable el que nos determina el tipo de una variable. Además se permite una variable en tiempo de ejecución almacene diferente tipos de datos. Por lo tanto habrá que tener especial atención a realizar ciertas operaciones dependiendo del tipo de dato involucrado sobre todo si tenemos cadenas de texto o números enteros.

Las reglas para identificar variables especifican:

- Pueden formarse mediante letras, números y símbolos ($ y _)
- Además el primer carácter no puede ser número ni contener @.

Ejemplo:
let nombre; CORRECTO
let nombreApellidos; CORRECTO
let minombre_10; CORRECTO
let $minombre; CORRECTO
let 10_minombre; DECLARACION INCORRECTA
let mi@nombre; DECLARACION INCORRECTA

2.2. Declarar variables: var, let y const

Se procede a detallar los diferentes tipos de declaración de variables:

- var declara una variable de ámbito global o local para la función, sin importar el ámbito de bloque. Permite hoisting.
- let declara una variable de ámbito: global, local para la función ó de bloque. Es reasignable y no permite hoisting.
- const declara una variable de ámbito global, local para la función o de bloque. No es reasignable, pero es mutable. No permite hoisting.

En general, let sería todo lo que se necesita dentro de un bloque, función o ámbito global. const sería para variables que no van sufrir una reasignación. var se puede relegar para cuando necesitemos hacer hoisting, vamos, casi nunca.

¿Qué es Hoisting?. Comportamiento que se puede entender como «la variable i ha sido declarada en el programa pero en el momento de intentar acceder a ella aún no tenía un valor asignado.

```
console.log(z); // ReferenceError: x is not defined
 var i; // Variable declarada pero valor no inicializado
 console.log(i); // undefined
 console.log(x); //undefined
 var x = 1;
 i = 1;
 console.log(i); // 1
 console.log(b); // ReferenceError: Cannot access 'b' before initialization
 let b = 1;
```

En los bucles for es habitual definir una variable sobre la que iterar. Esto se puede hacer tanto con let como con var. Aunque en el caso de let el alcance de las variables solamente es dentro del bucle. Lo que permite evitar la aparición de problemas.

```
 /*Con var, en ambos bucles la variable es la misma y por lo tanto la del segundo bloque machaca
la primera*/
 for (var i = 0; i < 2; i++) {
   console.log("Tercero", i);
   for (var i = 0; i < 2; i++) {
     console.log("Cuarto", i);}
 }
/*Con let es como si i fuese dos variables diferentes. Let tiene ambito detro del bloque */
 for (let i = 0; i < 2; i++) {
   console.log("Primer", i);
   for (let i = 0; i < 2; i++) {
     console.log("Segundo", i);}
 }
```

El uso de let evita que se alteren los valores de una variable de forma accidental, algo difícil de depurar. Además Let evita redeclarar una variable dos veces.

El uso de var desborda el bloque y conserva el valor para lo bueno y para lo malo, al ser la misma variable machaca el valor que tuviera anteriormente:

```
var mensaje = "Hola Mundo!";
 if (true) {
   //var mensaje = "Se ha cambiado el mensaje variable global";
   let mensaje = "Se ha cambiado el mensaje";
 }
 console.log(mensaje);
```

Si se necesita emplear una variable Let fuera de su alcance, se resolverá con un return:

```
function test() {
let result = 1;
result++;
return result;
}
let obj = test();
console.log(obj);
```

2.3. Identificar un tipo de dato

En JavaScript se puede consultar el tipo de dato con la función **typeof(lavariable).** Como se ha comentando anteriormente la consulta solo se valida en el momento de ejecutarla, debido a que podría tomar otros tipos de datos distintos.

```
let minombre;
let a = null;
let b = "cadena";
let c = 15;
let d = true;
alert(typeof minombre); //undefined
alert(typeof a); //objetc
alert(typeof b); //string
alert(typeof c); //number
alert(typeof d); //boolean
```

2.4. Tipo de dato: undefined

Se emplea cuando una variable ha sido declarada pero no ha sido inicializada y por lo tanto no tiene asignado ningún valor.

```
let minombre;
alert(typeof minombre); //undefined
```

2.5. Tipos de datos: Diferencias en Undefined y Null

La clave para entender la diferencia entre valores Undefined y null, radica en la noción de existencia. Si una variable es:

- **undefined**: para Javascript, no existe. Tiene dos explicaciones, por un lado no ha sido declarada o en ningún momento se le asignó un valor.
- **null**: para Javascript, la variable existe. En algún momento, explícitamente, la variable se estableció a null. No tiene un valor asignado.

Las principales confusiones vienen de que, en Javascript, las expresión:

undefined == null (igualdad abstracta)

es **verdadera** por efecto del "type coercion" (mostrar tabla).

Como ya explicamos, se recuerda que el operador == no funciona en Javascript cómo están acostumbrados la mayoría de los programadores. El operador de comparación al uso en Javascript es **===** y la expresión: (igualdad estricta)

undefined === null

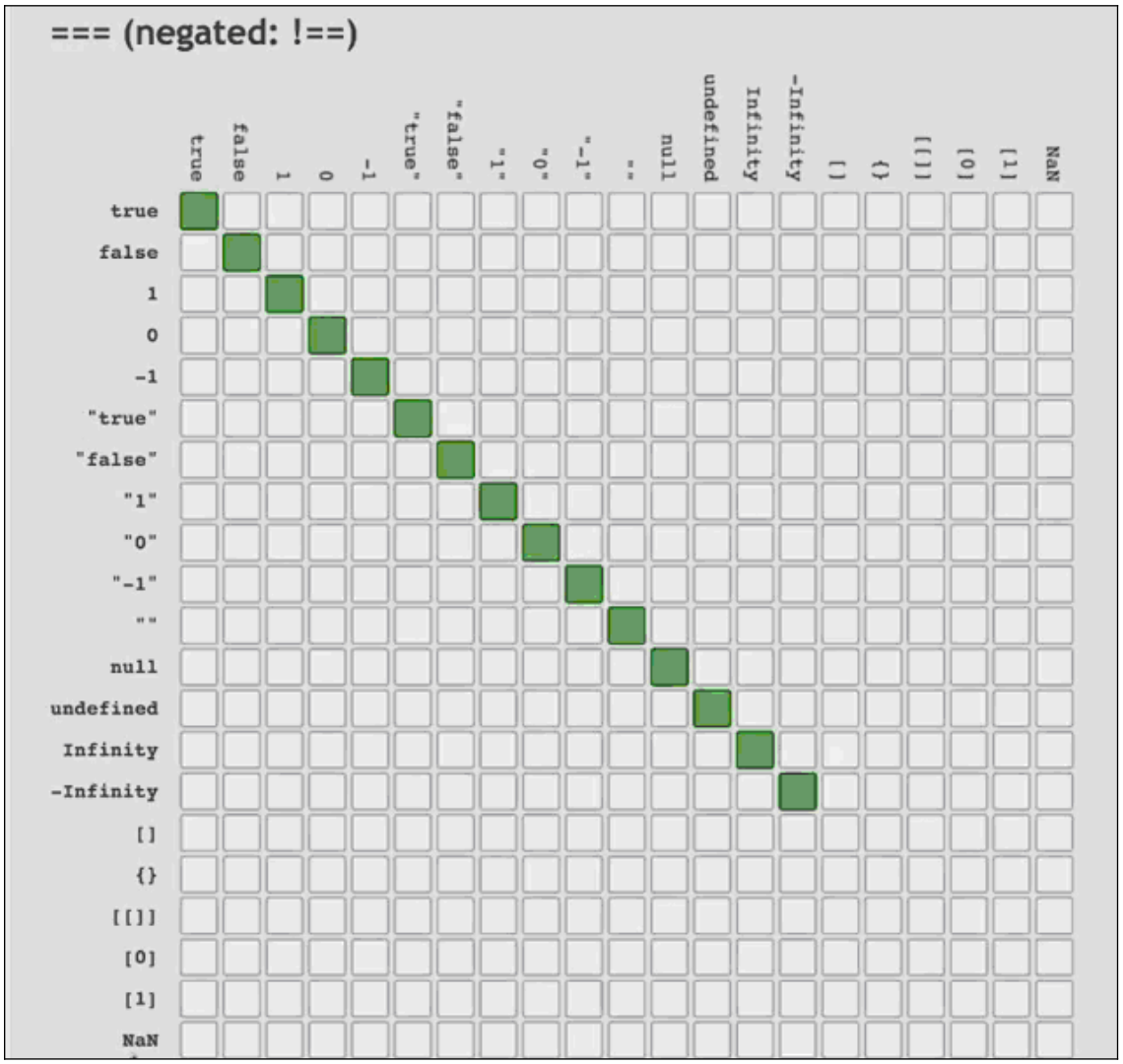
es **falsa** como cabía esperar.

```
console.log(null==undefined)  //true
console.log(null===undefined)  //false
```

Por tanto existen diferencias entre == y ===. Para comprenderlas al completo, tenemos que tener en cuenta que existe una regla general y excepciones que se mostraran en la siguiente tabla

Regla general:

- Si los operadores son del mismo tipo. Igual y estrictamente igual es lo mismo.
- Si un operador es de tipo cadena y el otro número. Convierte la cadena a número.
- Si un operador es booleano y el otro no. Convierte el no booleano a número y compara.



NaN no es igual ni estrictamente igual a NaN

```
console.log(NaN===NaN)  //false
console.log(NaN==NaN)    //false
```

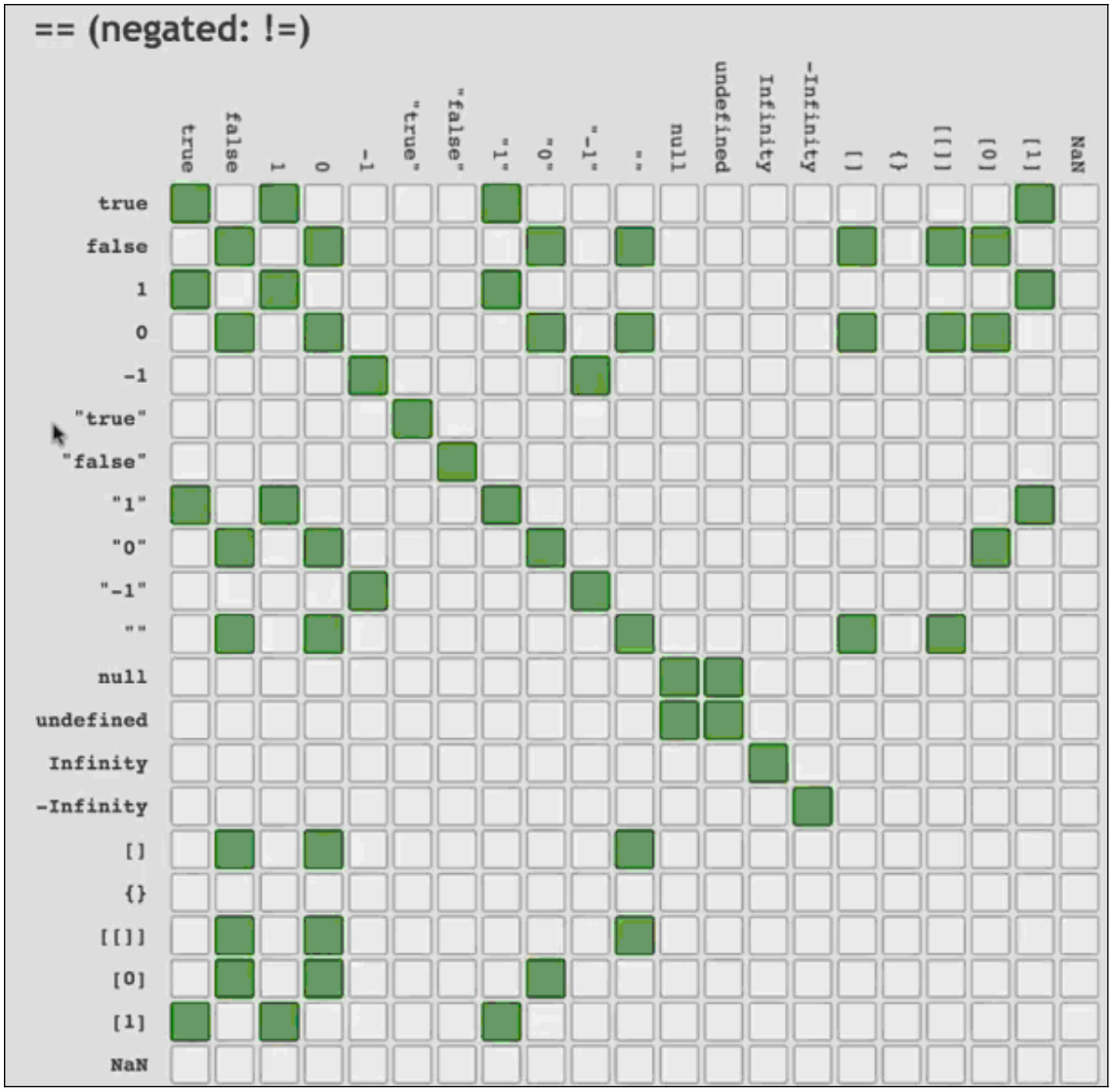
El valor inicial de NaN es Not-A-Number (No es Un Número)
NaN nunca es equivalente con cualquier otro número, incluido el mismo NaN.
No puedes chequear el valor de un not-a-number comparándolo con Number.NaN. Usar la función isNaN() para aquello.
Muchos métodos de JavaScript (como son el Number constructor, parseFloat y parseInt) retornan NaN si el valor especificado en el parámetro no puede ser parseado como un número.
Ejemplo: Valores de otro tipo pueden ser convertidos a números usando la función Number().
new Number(value);
var b = Number('123'); // b === 123 es true

Por lo tanto, tampoco puedes chequear el valor de un not-a-number comparándolo con Number.NaN.

Algunas excepciones importantes:

- ✓ Undefined es igual a Null

Resto de excepciones, ver a continuación:



2.6. Tipos de datos: String

Una cadena puede ser cualquier texto dentro de comillas. Puede usar comillas simples o dobles:

let a = "Letra";
let a = 'Letra';

Puede utilizar comillas dentro de una cadena, siempre que no coincidan con las comillas que rodean la cadena:

let comilla = "Instituto Escuela Secundaria";
let comilla = "Instituto Escuela Secundaria 'IES'";
let comilla = 'Instituto Escuela Secundaria "IES"';

Ejemplo:
Crea un Script y sin utilizar ninguna variable, tiene que mostrar un aviso con la siguiente frase:
1) A la gente de Zaragoza se les llama "Maños"

2) Nacimos en los 70's

Ejemplo: Ahora de forma conjunta
3) A la gente de Zaragoza se les llama "Maños" Nacimos en los 70's

4) ¿Sin quitar las comillas como se soluciona en JavaScritp?

```
alert('"Maños" de los 70's');
```

5) Concatenar palabras en una misma frase

```
let b = "cadena";
console.log(b + " " + b);
```

6) Calcular longitud de un string (variable.length)

```
var txt = "SUPERCALIFRAGILISTICOESPIALIDOSO";
alert(txt.length);
```

7) Cambiar color a la página

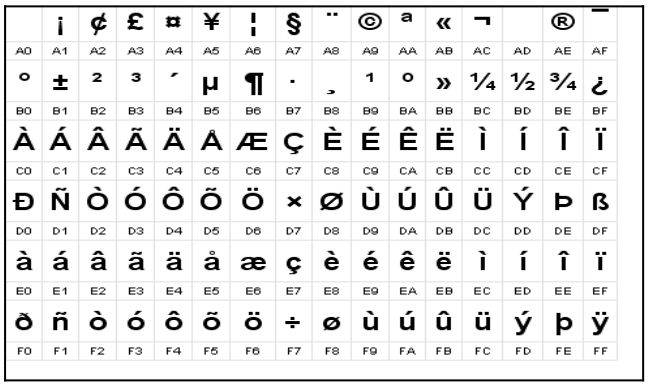
```
document.body.style.backgroundColor = "red";
```

8) Cambiar de color una sección.

Ayuda: En HTML, crear un div incluyendo los elementos que se deseen dentro.

9) Cadenas de escape
Permiten introducir dentro de una cadena caracteres especiales como saltos de líneas, acentos, tabulaciones, etc. Entre las cadenas de escape más conocidas podemos mencionar:
n: Salto de línea

\r: Retorno de carro
\t: Tabulación horizontal
\v: Tabulación vertical
\': Comilla simple o apostrofe
\": Comilla doble
\\: Barra invertida
\xdd: Caracter especial especificado por dos dígitos hexadecimanes *dd (ver tabla a continuación)*
NOTA: también se puede poner en unicode. Ej: \u0008



	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬		®	¯
A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF
à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF
ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ
F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF

Ejemplo: alert('\xE6');

10) Ejercicio propuesto:
Crear un script que emita un aviso en el que detalle la siguiente la afirmación:
<El número "µ" tiene un valor de 3,14….>

11) Ejercicio:
Crea un script que muestre un texto con 4 cadenas de escape diferentes.

2.7. Tipos de Datos: Numéricos

Conversión de variables a números
Hay 3 métodos de JS que se pueden utilizar para convertir variables en números:

- El Number()método
- El parseInt()método
- El parseFloat()método

Estos métodos no son métodos numéricos, sino métodos JavaScript globales.

Métodos globales de JavaScript
Los métodos globales de JavaScript se pueden utilizar en todos los tipos de datos de JS.

Estos son los métodos más relevantes cuando se trabaja con números:
Number(): Devuelve un número, convertido a partir de su argumento.

parseFloat(): Analiza su argumento y devuelve un número de punto flotante.
parseInt(): Analiza su argumento y devuelve un entero

https://www.w3schools.com/js/js_number_methods.asp

2.8. Comparaciones y operadores lógicos

JavaScript Arithmetic Operators

Arithmetic operators are used to perform arithmetic on numbers:

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
**	Exponentiation (ES2016)
/	Division
%	Modulus (Division Remainder)
++	Increment
--	Decrement

https://www.w3schools.com/js/js_comparisons.asp

2.9. Comentarios

Existen dos formas para realizar comentarios al código de JS:

- De linea: //

Ejemplo:
var minombre; // esto es un variable

- Varias lineas: /* */

Ejemplo:
/* comentario de varias líneas
Con el ejemplo de una variable */

2.10. Mensajes de confirmación al visitar Páginas Web

✓ Mensajes de confirmación para poder validar o extraer información al usuario acerca de la actividad relacionada con el sitio web. Nos permite además acercarnos al usuario y mejorar su experiencia de la página.

```
let confirmacion = confirm("¿Te ha gustado nuestra página Web?");
//Detectamos si el usuario acepto el mensaje
if (confirmacion) {
  alert("¡Gracias por confiar en nosotros!");
}
//Detectamos si el usuario rechaza el mensaje
else {
  alert("¡Intentaremos mejorarla proximamente!");
}
```

✓ Además se emplean para certificar determinados sitos web que son seguros y verificar su legitimad frente a la normativa vigente.

```
<body>
<a href="http://www.videojuegos.com/" onclick="return confirmar()">
  Ir a Videojuegos</a>
<script type="text/javascript">
  let minombre;
  let a = null;
  function confirmar() {
    return confirm(
      "Esta pagina tiene contenido para adulto. ¿Eres mayor de 18 años?"
    );
  }
    </script>
    </body>
```

2.11. Mejorar la experiencia del Usuario

Para continuar mejorando la experiencia en la Web, emplearemos scripts para solicitar información al usuario y realizar avisos personalizados. Muy empleado en zonas de registro de la pagina o al abrir determinados menús del sitio.

```
let nombre = prompt("Introduce tu nombre");
if (nombre != null) {
  alert("Hola " + nombre + "! Gracias por darte de alta");
}
```

2.12. Ejercicios

Ejercicio: Crea un fichero op1.js que incluirá las siguientes cuestiones:

1) Crea una aplicación que solicitar el nombre al usuario y lo guardará en una variable denominada nombre. Solicitar el primer apellido al usuario y lo guardará en una variable denominada apellido.
   Almacenaremos en una nueva variable denominada fullName, el nombre y primer apellido registrado separados por un espacio.
   Solicitar la edad al usuario y lo guardas en una variable denominada edad. Calcular y asignar a una nueva variable Nacimiento, el año de nacimiento del usuario (sin tener en cuenta el mes de nacimiento.
   Mostrar en el cuadro de resultados del editor la siguiente información (una en cada línea):
   - Nombre completo: (valor de la variable fullName)
   - Año de nacimiento: (valor de la variable year)

2) Calcula el exponencial de cualquier número

3) Crea un script para calcular números pares e impares. Los números se introducen por teclado y emplear el módulo %.

   Solución: parimpar.html

4) Detectar si un número es múltiplo de otro número.

5) Crea un Array que almacene los 12 meses del año y muestra cada mes empleando un for

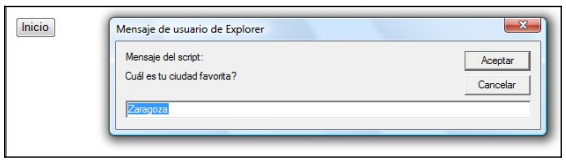
6) Compara dos números enteros para saber cual es mayor. Además comprueba si son positivos

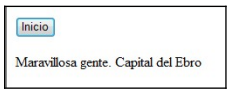
7) Muestra en el documento HTML los 30 primeros números

8) Muestra en el documento HTML el factorial de un número que se inserta por teclado. Declara dos variables: para el número y para el resultado. Emplea un for.

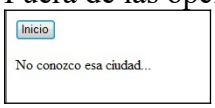
   Solución: factorial.html

9) Crea un script para que al accionar un botón, solicite insertar el nombre de una ciudad. Muestra en una ventana, la elección múltiple de 3 ciudades con switch. Tienes que declarar las siguientes ciudades (Zaragoza, Barcelona, Madrid) y que cada una tenga su propio mensaje.





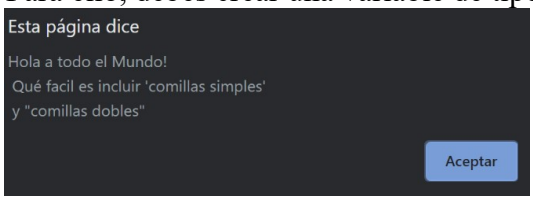
Fuera de las opciones aparece el siguiente mensaje:



   Solución: Elección multiple-respuesta.html

Ejercicio: Crea un fichero op2.js que incluirá las siguientes cuestiones:

1) Realiza una aplicación Web, que mediante JavaScript muestre el siguiente aviso. Para ello, debes crear una variable de tipo String llamada mensaje.



2) Realiza una aplicación Web, que mediante JavaScript haga divisiones en binario con desplazamiento a la derecha.
Ejemplo Decimal: 40 / 16 vs Ejemplo binario: 40 >> 4.

3) Realiza una aplicación Web, que mediante JavaScript haga multiplicaciones en binario con desplazamiento a la izquierda.
Ejemplo Decimal: 26 x 4 vs Ejemplo binario: 26 << 2.

4) Realiza una aplicación Web, que mediante JavaScript muestre por consola, el máximo y el valor mas cercano a cero posible con Javascript. Ayuda: emplea Number.MaxValue y Number.MinValue. Basándote en lo anterior, muestra un valor infinito.

5) Dadas las dos siguientes cadenas:
OVNI = “OBJETO VOLADOR NO IDENTIFICADO"
Info = “En un lugar de la mancha”
Realiza una aplicación Web, que mediante JavaScript compruebe si las cadenas están escritas en minúsculas, mayúsculas ó ambas. Emplea: toUpperCase() y toLowerCase()
De manera adicional, mejora la aplicación anterior, para poder evaluar cualquier cadena de texto que introduce el usuario.

2.13 Ejercicios Ampliación

1) Crea una aplicación Web que mediante JavaScript calcule la nota de la primera evaluación de la asignatura de Desarrollo Web Entorno Cliente.
   Crea dos variables: NotaProyecto y NotaExamen que se introducirán por teclado.
   Sacar por pantalla la media resultante y su calificación. Contemplar cada uno de los siguientes casos:
   - Si NotaProyecto o NotaExamen es menor de 4,5. Calificación: Suspenso.
   - Si NotaProyecto y NotaExamen es mayor o igual de 4,5, pero la media resultante inferior a 5. Calificación: Suspenso.
   - NotaProyecto y NotaExamen es mayor o igual de 4,5 y la media resultante entre 5 y menor de 7. Calificacion: Aprobado.
   - NotaProyecto y NotaExamen es mayor o igual de 4,5 y la media entre 7 y menor de 9. Calificacion: Notable.
   - NotaProyecto y NotaExamen es mayor o igual de 4,5 y la media mayor o igual que 9. Calificación: Sobresaliente.

   Recordar que las notas de proyecto y de examen van desde 0 a 10 (ambos incluidos).

2) Realiza una aplicación Web, que mediante JavaScript verifique el DNI. Se solicitará dos entradas al usuario.
   - Número de DNI
   - Letra de DNI

   Se mostrará por pantalla si el DNI es correcto o incorrecto (Numero y letra).
   Se contemplarán las siguientes excepciones:
   - Longitud del número incorrecta.
   - Letra no introducida
   - Carácter de la letra incorrecto.

   Condiciones:
   - El archivo de JavaScript se llamará dni.js y la aplicación Web se llamará CalculoDNI
   - Tendrás que crearte un array llamado “abcedario” con las siguiente información:

   ['T', 'R', 'W', 'A', 'G', 'M', 'Y', 'F', 'P', 'D', 'X', 'B', 'N', 'J', 'Z', 'S', 'Q', 'V', 'H', 'L', 'C', 'K', 'E']
   - Número de DNI negativos y superiores a 99.999.999 no existen.
   - Para el cálculo de la letra, emplea el módulo del total de posiciones del array.

3) Empleando el fichero operaciones.js, realiza una aplicación Web que contenga la siguiente información:

Javascript

TABLA DE MULTIPLICAR DEL 7
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
7 x 10 = 70

TABLA DE SUMAR DEL 8
8 + 1 = 9
8 + 2 = 10
8 + 3 = 11
8 + 4 = 12
8 + 5 = 13
8 + 6 = 14
8 + 7 = 15
8 + 8 = 16
8 + 9 = 17
8 + 10 = 18

TABLA DE DIVIDIR DEL 9
9 / 1 = 9
9 / 2 = 4.5
9 / 3 = 3
9 / 4 = 2.25
9 / 5 = 1.8
9 / 6 = 1.5
9 / 7 = 1.2857142857142858
9 / 8 = 1.125
9 / 9 = 1
9 / 10 = 0.9

125 / 8 con desplazamiento de bits

15

40 x 4 con desplazamiento de bits

160

25 / 2 con desplazamiento de bits

12

10 x 16 con desplazamiento de bits

160

Emplea los siguientes archivos para su resolución:

- La aplicación Web se llama “OperacionesBits”
- El fichero de Javascript se llama “Operaciones”

Condiciones:

- Utilizar los tres tipos de bucles que hay en JavaScript (un bucle diferente para cada número).
- No se puede borrar ninguna línea de código de los archivos.
- No se puede modificar los nombres de los archivos.
- Se escribe el código que se necesite debajo de cada comentario.