JavaScript
parte 1

Carolina Caballero

Agenda

- Introducción a Javascript
- Operadores
- Tipos de Datos
- Arrays

neitek_

Introducción a Javascript

JavaScript

- Lenguaje de scripting
- Orientado a Objetos
- Interpretado
- Se ejecuta en casi cualquier navegador
- Es débilmente tipeado
- Uso del lado del cliente
- Permite efectos, eventos, mostrar mensajes, validaciones, etc.

Versiones JavaScript

- JavaScript fue inventado por Brendan Eich en1995, se convirtió en estándar ECMA en 1997.
- Nombre oficial : ECMAScript .
- Las versiones ECMAScript se abrevian: ES1, ES2, ES3, ES5 y ES6
- A partir de 2016 las versiones llevan el año (ECMAScript 2016 / 2017 / 2018 …)

Ver	Official Name	Description
ES1	ECMAScript 1 (1997)	First edition
ES2	ECMAScript 2 (1998)	Editorial changes
ES3	ECMAScript 3 (1999)	Added regular expressions Added try/catch Added switch Added do-while
ES4	ECMAScript 4	Never released
ES5	ECMAScript 5 (2009) Read More	Added "strict mode" Added JSON support Added String.trim() Added Array.isArray() Added Array iteration methods Allows trailing commas for object literals

Versiones JavaScript

ES6	ECMAScript 2015 Read More	Added let and const Added default parameter values Added Array.find() Added Array.findIndex()
	ECMAScript 2016 Read More	Added exponential operator (**) Added Array.includes()
	ECMAScript 2017 Read More	Added string padding Added Object.entries() Added Object.values() Added async functions Added shared memory
	ECMAScript 2018 Read More	Added rest / spread properties Added asynchronous iteration Added Promise.finally() Additions to RegExp

Posibilidades y Limitaciones

- Entorno limitado
- No acceso a recursos fuera del dominio
- Solo cierra ventanas que el script mismo haya abierto
- Ventanas no demasiado grandes
- Ventanas no demasiado pequeñas
- Ventanas fuera de la vista del usuario (depende del navegador)
- No se puede acceder a archivos del sistema
- No se puede acceder a preferencias del navegador
- Se puede detener el script si toma demasiado tiempo

Posibilidades y Limitaciones

- Crear y leer cookies dentro del mismo dominio
- Acceso limitado a objetos potencialmente peligrosos
- Confirmación del usuario antes de realizar ciertas acciones
- Elementos descargados de otros servidores bloqueados

Sintaxis

- No se toman en cuenta los espacios en blanco y las nuevas líneas.
- Se distinguen las mayúsculas y minúsculas (case sensitive).
- No se define el tipo de variables.
- No es necesario terminar cada secuencia con punto y coma (pero es altamente recomendado).
- Se pueden incluir comentarios.

Primer Script

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
 <HEAD>
  <script type="text/javascript">
  <!--
    alert('Hola!');
  -->
  </script>

 </HEAD>
 <BODY>
 </BODY>
</HTML>
```

- File: Sample01.html

Primer Script

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
 <HEAD>
  <script type="text/javascript" src="/js/archivo.js"></script>
 </HEAD>
 <BODY>
 </BODY>
</HTML>
```

- File: Sample02.html

Comentarios

```
<script type="text/javascript">
// Comentario de una línea

/*
    Comentario de
    múltiples líneas
*/
</script>
```

Variables

- Las variables en JS son contenedores para almacenar datos.

```
var numero1 = 3;
var numero2 = 2;
var resultado = numero1 + numero2;

var num1 = 1;
var num2 = 2;
resultado = num1 + num2

var num1 = 1;
var num2;
num2 = 2;
var resultado = num1 + num2;
```

Variables

- Deben tener nombre único (identifier).
- Sólo puede estar formado por letras, números y los símbolos $ (peso o dólar) y _ (guion bajo).
- El primer carácter no puede ser un número.
- No puede usarse palabras reservadas

```
var $numero1;
var _$letra;
var $$$otroNumero;
var $_a__$4;
```

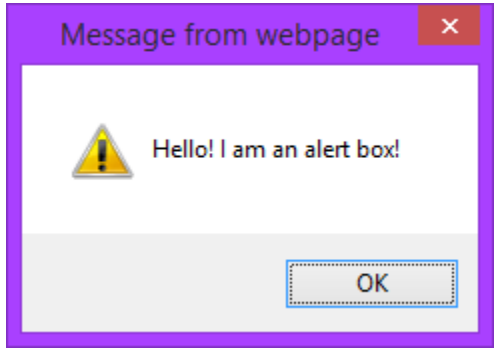
Palabras reservadas

abstract	arguments	await*	boolean
break	byte	case	catch
char	class*	const	continue
debugger	default	delete	do
double	else	enum*	eval
export*	extends*	false	final
finally	float	for	function
goto	if	implements	import*
in	instanceof	int	interface
let*	long	native	new
null	package	private	protected
public	return	short	static
super*	switch	synchronized	this
throw	throws	transient	true
try	typeof	var	void
volatile	while	with	yield

Tipos de mensajes

- Mensaje **alert**.
  - Abre una ventana de dialogo.
  - Tiene únicamente el botón OK.
  - Recibe un parametro (opcional) para desplegar en la ventana de dialogo.

```
alert("Hello! I am an alert box!!");
```



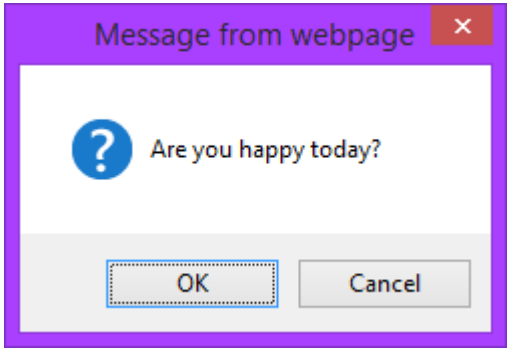
Tipos de mensajes

- Mensaje **confirm**.
  - Abre una ventana de confirmación.
  - Tiene los botones OK y Cancel.
  - Recibe un parametro (opcional) para desplegar en la ventana de dialogo.
  - Regresa true si se presiona OK, o false en cualquier otro caso.

```
<script>

  if (confirm("Are you happy today?") )
  {   alert ("=)");  }
  else
  {   alert (":(");  }

</script>
```



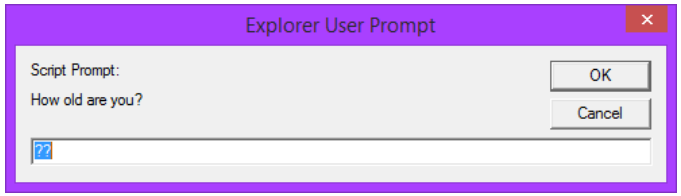
Tipos de mensajes

- Mensaje **prompt**.
  - Abre una ventana de pregunta.
  - Tiene los botones OK y Cancel y un campo texto.
  - Recibe 2 parametros, el primero es obligatorio, es el texto que se va a desplegar en la ventana, el segundo es opcional y es el valor default del campo de entrada.
  - Pide la captura de una respuesta al usuario, que se regresa como respuesta a la llamada de **prompt**

```
<script>

  var res = "";
  res = prompt ("How old are you?", "??");
  alert (res);

</script>
```



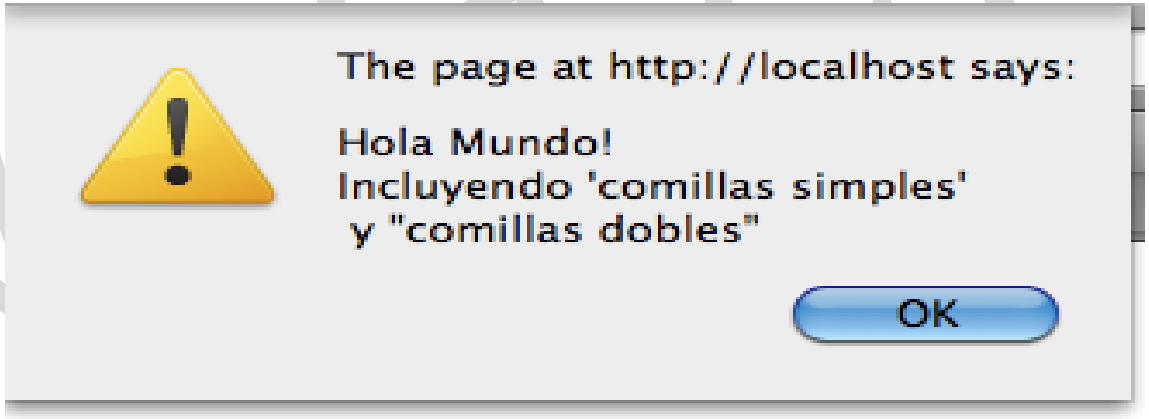
- File: Sample03.html

Ejercicio

- Escribe el codigo JS que haga las siguientes preguntas:
  - 1.-Como te llamas?
  - 2.-Eres mayor de edad?
  - 3a.-Cual es tu pelicula favorita?
  - 3b.-Cual es tu caricatura favorita?
- Guarda las respuestas en diferentes variables.
- Si la respuesta de la pregunta #2 es verdadera, entonces hacer la pregunta 3a, de lo contrario hacer la pregunta 3b.
- Desplegar un resumen de las preguntas y respuestas en un mensaje de alerta.

Ejercicio

- Crear un script que muestre el siguiente mensaje:



neitek_

Agenda

- Introducción a Javascript
- Operadores
- Tipos de Datos
- Arrays

Operadores

Operadores de Asignación

```
<script type="text/javascript">

  var numero1 = 3;
  var numero2 = 4;

  // Error
  5 = numero1;

  // Ahora, la variable numero1 vale 5
  numero1 = 5;

  // Ahora, la variable numero1 vale 4
  numero1 = numero2;

</script>
```

Operadores de Asignación

Operator	Example	Same As
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y

Operadores Aritméticos

```
<script type="text/javascript">

 // + Suma
 // - Resta
 // / División
 // * Multiplicación
 // % Modulo

  var numero1 = 10;
  var numero2 = 5;

  resultado = numero1 / numero2;   // resultado = 2
  resultado = 3 + numero1;         // resultado = 13
  resultado = numero2 - 4;         // resultado = 1
  resultado = numero1 * numero 2;  // resultado = 50
  resultado =  numero1 % 8;        // resultado = 2

</script>
```

- File: Sample04.html

Operadores Aritméticos

Operador	Descripción
++X	Incrementa x en una unidad y devuelve el valor
X++	Devuelve el valor de x y lo deja incrementado en una unidad
--X	Decrementa x en una unidad y devuelve el valor
X--	Devuelve el valor de x y lo deja decrementado en una unidad

Operadores de Comparación

==	Equal	x == y
===	Strict equal	x === y
!=	Unequal	x != y
!==	Strict unequal	x !== y

Operadores de Comparación

Operator	Description	x = 5	Comparing	Returns
==	equal to		x == 8	false
			x == 5	true
			x == "5"	true
===	equal value and equal type		x === 5	true
			x === "5"	false
!=	not equal		x != 8	true
!==	not equal value or not equal type		x !== 5	false
			x !== "5"	true
			x !== 8	true

Operadores Lógicos

Operador	Descripción
&&	Y lógico (devuelve verdadero si los dos operandos son verdaderos, y falso en caso contrario)
||	O lógico (devuelve verdadero si uno de los dos operandos o ambos son verdaderos, y falso en caso contrario)
!	No lógico (devuelve verdadero si el operando es falso, y falso si es verdadero)

Operadores Lógicos

x = 6 and **y = 3**

Operator	Description	Example
&&	and	(x < 10 && y > 1) is true
||	or	(x == 5 || y == 5) is false
!	not	!(x == y) is true

Operadores de Bits

Operation	Result	Same as	Result
5 & 1	1	0101 & 0001	0001
5 | 1	5	0101 | 0001	0101
~ 5	10	~0101	1010
5 << 1	10	0101 << 1	1010
5 ^ 1	4	0101 ^ 0001	0100
5 >> 1	2	0101 >> 1	0010
5 >>> 1	2	0101 >>> 1	0010

Operadores de Comparación

Operador	Descripción
==	Devuelve verdadero si los dos operandos son iguales
!=	Devuelve verdadero si los dos operadores son distintos
>	Devuelve verdadero si el primer operando es mayor que el segundo
<	Devuelve verdadero si el primer operando es menor que el segundo
>=	Devuelve verdadero si el primer operando es mayor o igual que el segundo
<=	Devuelve verdadero si el primer operando es menor o igual que el segundo

Operadores de Comparación

Case	Value
2 < 12	true
2 < "12"	true
2 < "John"	false
2 > "John"	false
2 == "John"	false
"2" < "12"	false
"2" > "12"	true
"2" == "12"	false

Operadores Condicionales

- Operador condicional (ternario) ? :

```
variablename = (condition) ? value1:value2
```

```
var voteable = (age < 18) ? "Too young":"Old enough";
```

Ejercicio

- El factorial de un número entero n es una operación matemática que consiste en multiplicar todos los factores n x (n-1) x (n-2) x ... x 1. Así, el factorial de 5 (escrito como 5!) es igual a:

$$5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$$

- Utilizando la estructura **for**, crear un script que calcule el factorial de un número entero y despliegue la respuesta detallada.

neitek_

Agenda

- Introducción a Javascript
- Operadores
- Tipos de Datos
- Arrays

Tipos de Datos

Tipos de datos

Tipo	Ejemplo
Número	Cualquier número 12, 22.5, 2e8
Cadena	"Hola" o 'Hola'
Booleano	true o false
Objeto	Array, etc
Función	
NULL	
undefined	

Tipos de datos Dinámicos

```
var x;               // Now x is undefined
var x = 5;           // Now x is a Number
var x = "John";      // Now x is a String
```

Características en Strings

```
var carName = "Volvo XC60";   // Using double quotes
var carName = 'Volvo XC60';   // Using single quotes
```

```
var answer = "It's alright";             // Single quote inside double quotes
var answer = "He is called 'Johnny'";    // Single quotes inside double quotes
var answer = 'He is called "Johnny"';    // Double quotes inside single quotes
```

Funciones para Strings

lenght calcula la longitud de una cadena.

```
var mensaje = "Hola Mundo";
var numLetras = mensaje.length; //numeroLetras = 10
```

+ concatena cadenas

```
var mensaje1 = "Hola";
var mensaje2 = " Mundo";
var mensaje = mensaje1 + mensaje2; // mensaje = "Hola Mundo"
```

concat() concatena cadenas

```
var mensaje1 = "Hola";
var mensaje2 = mensaje1.concat(" Mundo");
// mensaje2 = "Hola Mundo"
```

Funciones para Strings

toUpperCase() transforma a mayúsculas todos los caracteres

```
var mensaje1 = "Hola";
var mensaje2 = mensaje1.toUpperCase(); // mensaje2 = "HOLA"
```

toLowercase() transforma a minúsculas todos los caracteres

```
var mensaje1 = "HolA";
var mensaje2 = mensaje1.toLowerCase(); // mensaje2 = "hola"
```

Funciones para Strings

charAt(posicion)

```
var mensaje = "Hola";
var letra = mensaje.charAt(0); // letra = H
letra = mensaje.charAt(2);     // letra = l
```

indexOf(caracter)

```
var mensaje = "Hola";
var posicion = mensaje.indexOf('a'); // posicion = 3
posicion = mensaje.indexOf('b');     // posicion = -1
```

lastIndexOf(caracter)

```
var mensaje = "Hola";
var posicion = mensaje.lastIndexOf('a'); // posicion = 3
posicion = mensaje.lastIndexOf('b');     // posicion = -1
```

Funciones para Strings

search(stringToFind)

```
var mensaje = "Hola Mundo";
var porcion = mensaje.substring(2); // porcion = "la Mundo"
porcion = mensaje.substring(7);     // porcion = "ndo"
porcion = mensaje.substring(-2);    // porcion = "Hola Mundo"
porcion = mensaje.substring(1, 8);  // porcion = "ola Mun"
porcion = mensaje.substring(3, 4);  // porcion = "a"
porcion = mensaje.substring(5, 0);  // porcion = "Hola "
porcion = mensaje.substring(0, 5);  // porcion = "Hola "
```

split(separador)

```
var mensaje = "Hola Mundo, soy una cadena de texto!";
var palabras = mensaje.split(" ");
// palabras = ["Hola", "Mundo,", "soy", "una", "cadena", "de", "texto!"];
```

Mecanismos de Escape

- \n o \r Una nueva línea
- \t Un tabulador
- \' Una comilla simple
- \" Una comilla doble
- \\ Una barra inclinada

```
var x = 'It\'s alright';
var y = "We are the so-called \"Vikings\" from the north."
```

Características en Números

```
var x1 = 34.00;     // Written with decimals
var x2 = 34;        // Written without decimals
```

```
var y = 123e5;      // 12300000
var z = 123e-5;     // 0.00123
```

Propiedades de Números

Property	Description
constructor	Returns the function that created JavaScript's Number prototype
MAX_VALUE	Returns the largest number possible in JavaScript
MIN_VALUE	Returns the smallest number possible in JavaScript
NEGATIVE_INFINITY	Represents negative infinity (returned on overflow)
NaN	Represents a "Not-a-Number" value
POSITIVE_INFINITY	Represents infinity (returned on overflow)
prototype	Allows you to add properties and methods to an object

Métodos para Números

Method	Description
isFinite()	Checks whether a value is a finite number
isInteger()	Checks whether a value is an integer
isNaN()	Checks whether a value is Number.NaN
isSafeInteger()	Checks whether a value is a safe integer
toExponential(x)	Converts a number into an exponential notation
toFixed(x)	Formats a number with x numbers of digits after the decimal point
toPrecision(x)	Formats a number to x length
toString()	Converts a number to a string
valueOf()	Returns the primitive value of a number

Funciones para Números

isNaN()

```
var numero1 = 0;
var numero2 = 0;
if(isNaN(numero1/numero2)) {
  alert("La división no está definida para los números indicados");
} else {
  alert("La división es igual a => " + numero1/numero2);
}
```

Funciones para Números

toFixed(digitos)

```
var numero1 = 4564.34567;
numero1.toFixed(2); // 4564.35
numero1.toFixed(6); // 4564.345670
numero1.toFixed(); // 4564
```

Ejercicios

- Capura de un string, convertir cada primer letra de cada palabra en mayúscula, desplegar valor resultante.

- Capturar un número, desplegar el número en forma invertida.

Agenda

- Introducción a Javascript
- Operadores
- Tipos de Datos
- Arrays

Arrays

Arrays

- Los JS Arrays se escriben con [].
- Los elementos del Array se separan con comas.
- El primer elemento del array se encuentra en la posición 0.

```
var cars = ["Saab", "Volvo", "BMW"];
```

Reemplazan esto:	Por esto:

x0=0;
x1=1;
x2=2;
x3=3;
x4=4;
x5=5;
	x[0]=0;
	x[1]=1;
	x[2]=2;
	x[3]=3;
	x[4]=4;
	x[5]=5;

Creación de Arrays

- var meses=[];
- var meses = new Array();
- var meses = new Array(12);

Inicialización de Arrays

- var meses= ["Ene", "Feb", "Mar", "Abr", "May", "Jun", "Jul", "Ago", "Sep", "Oct", "Nov", "Dic"];

- var meses = [];
- meses[0] = "Ene";
- meses[1] = "Feb";
- meses[2] = "Mar“;
- meses[11] = "Dic";

meses[3] .. meses[10] es undefined

Elementos de array

- Los arrays en JS son objetos.
- Es posible tener valores de diferentes tipo en el mismo array.
- Un array puede contener:
  - Tipos de datos básicos (strings, números, booleanos)
  - Objetos
  - Funciones
  - Arrays

Recorriendo Arrays

```
<script>

    var x = [1,2,3,4,5,6,7,8]
    for(i=0; i<x.length; i++) {
       document.writeln(x[i]+'<br>');
    }

</script>
```

1
2
3
4
5
6
7
8

```
<script>

    var cars = ["Saab", "Volvo", "BMW"];
    document.getElementById("demo").innerHTML = cars;

</script>
```

Saab,Volvo,BMW

- File: Sample05.html

Arrays – propiedades

- length: retorna el número de elementos del arreglo.

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.length;                          // the length of fruits is 4
```

- File: Sample06.html

Arrays – métodos

- toString(): convierte un arreglo en cadena separada por comas.

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo").innerHTML = fruits.toString();
```

```
Banana,Orange,Apple,Mango
```

- join(*separator*): es como el toString, solo que se puede especificar el separador.

```
var fruits = ["Banana", "Orange","Apple", "Mango"];
document.getElementById("demo").innerHTML = fruits.join(" * ");
```

```
Banana * Orange * Apple * Mango
```

- File: Sample06.html

Arrays – métodos

- pop(): elimina el ultimo elemento, retorna el elemento eliminado.

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.pop();                 // Removes the last element ("Mango") from fruits
```

- push(): agrega al final un elemento, retorna el nuevo length del arreglo.

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.push("Kiwi");          //  Adds a new element ("Kiwi") to fruits
```

- File: Sample06.html

Arrays – métodos

- shift(): elimina el primer elemento del array, retorna el elemento eliminado.

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.shift();            // Removes the first element "Banana" from fruits
```

- unshift(): agrega un elemento al inicio del array, retorna el nuevo length del array.

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.unshift("Lemon");    // Adds a new element "Lemon" to fruits
```

  - File: Sample06.html

Arrays – métodos

- delete(): elimina un elemento del arreglo, puede producir huecos en el array con valores *undefined*.

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
delete fruits[0];           // Changes the first element in fruits to undefined
```

- splice(): agrega elementos en una posición determinada

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.splice(2, 0, "Lemon", "Kiwi");
```

Banana,Orange,Lemon,Kiwi,Apple,Mango

- File: Sample06.html

Arrays – métodos

- concat(): crea un nuevo arreglo uniendo uno o mas arreglos

```
var myGirls = ["Cecilie", "Lone"];
var myBoys = ["Emil", "Tobias","Linus"];
var myChildren = myGirls.concat(myBoys);
```

```
var arr1 = ["Cecilie", "Lone"];
var arr2 = ["Emil", "Tobias","Linus"];
var arr3 = ["Robin", "Morgan"];
var myChildren = arr1.concat(arr2, arr3);
```

```
var arr1 = ["Cecilie", "Lone"];
var myChildren = arr1.concat(["Emil", "Tobias","Linus"]);
```

- File: Sample07.html

Arrays – métodos

- slice(): crea un nuevo arreglo a partir de una posicion del arreglo original.

```
var fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];
var citrus = fruits.slice(1);
```

Orange,Lemon,Apple,Mango

```
var fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];
var citrus = fruits.slice(1, 3);
```

Orange,Lemon

- File: Sample07.html

Arrays – métodos

- sort(): ordena alfabeticamente el arreglo

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.sort();              // Sorts the elements of fruits
```

```
var points = [40, 100, 1, 5, 25, 10];
points.sort(function(a, b){return a - b});
```

- File: Sample08.html

Arrays – métodos

- reverse(): pone en orden inverso

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.sort();              // Sorts the elements of fruits
fruits.reverse();           // Reverses the order of the elements
```

- File: Sample08.html

Valor máximo en array

- Encontrar el mayor valor de un arreglo:

```
function myArrayMax(arr) {
    return Math.max.apply(null, arr);
}
```

```
function myArrayMax(arr) {
    var len = arr.length
    var max = -Infinity;
    while (len--) {
        if (arr[len] > max) {
            max = arr[len];
        }
    }
    return max;
}
```

- File: Sample09.html

Array multidimensionales

- No es un tipo de dato especial, es en realidad un array de arrays.

```
var x=[0,1,2,3,4,5];
var y=[x];
document.writeln(y[0][3]); // Salida: 2
```

Los Arrays son asignados por Referencia

```
var miArray = [ 'cero', 'uno', 'dos' ];
var nuevoArray = miArray;

nuevoArray[1] = 'cambio';

document.writeln(miArray[1]); // Salida: cambio
```

```
var miArray = [ 'cero', 'uno', 'dos', 'tres' ];
document.writeln(miArray[1]); // Salida: uno

function pasadoPorReferencia(refArray) {
   refArray[1] = 'cambio';
}

pasadoPorReferencia(miArray);
document.writeln(miArray[1]); // Salida: cambio
```

Ejercicios

- Hacer función para encontrar el valor mínimo de un arreglo.
- Hacer una función para crear una matriz para las tablas de multiplicar. Pedir como dato de entrada el valor máximo de tabla. Llenar el arreglo con los valores correspondientes. Desplegar tablas de multiplicación en forma alineada.
- Los arrays son pasados por referencia por default, escribir un script para evitar este comportamiento para que no se sobreescriba el array original.

Continua en JS_2