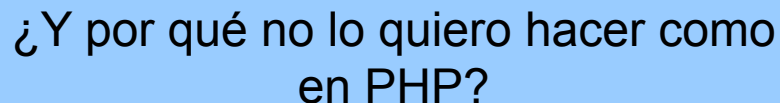


TEMA 3

Interfaces Interactivas

Objetivos

- Crear interfaces web que interactúen con el usuario de forma compleja:
 - Adición/supresión de elementos.
 - Reacción a acciones del usuario:
 - P. ej. Movimiento de ratón.
 - Todo esto sin necesidad de generar una nueva página web con procesamiento en el servidor (Como hacemos en PHP).



¿Y por qué no lo quiero hacer como en PHP?

3.1 JavaScript Básico

3.1 JavaScript Básico

1. Inmersión en JavaScript
2. El lenguaje JavaScript
3. Programación estructurada
4. Vectores
5. Funciones
6. Ámbitos de código y variables
7. Funciones propias del lenguaje

3.1.1 Inmersión en JavaScript

- DHTML
- JavaScript
- ¿Qué puedo hacer con JavaScript?
- Primer programa en JavaScript
- Entrada y salida básica

DHTML

- Dynamic HTML: Conjunto de técnicas para crear páginas web interactivas, en contraste con las páginas web estáticas.
- DHTML= HTML + CSS + JavaScript + DOM.
 - HTML: Definición y estructuración del contenido.
 - CSS: Presentación visual.
 - JavaScript: Lenguaje de programación. Define las acciones que se realizan según determinadas condiciones.
 - DOM: Document Object Model. Los diferentes elementos de una página web modelados/representados como objetos, lo que permite interactuar con ellos y modificarlos desde un lenguaje orientado a objetos.

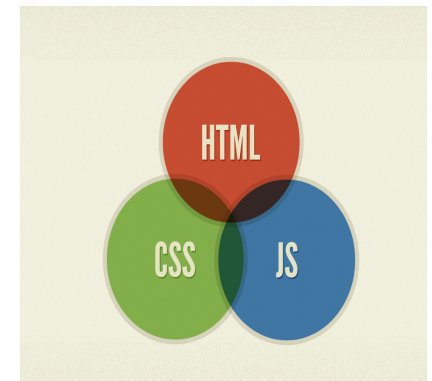
JavaScript



- ¿Qué es?
 - Lenguaje de programación ligero (Script).
 - Lenguaje para añadir interactividad a páginas web (Originalmente).
 - Hay usos en servidor, PDF, aplicaciones de escritorio, etc.
 - Creado por Netscape para su navegador (1995).
 - Interpretado → Multiplataforma.
 - Orientado a prototipos. Extensión por clonación.
 - http://es.wikipedia.org/wiki/Programaci%C3%B3n_basada_en_prototipos
 - Estándar ECMA e ISO. ECMAScript.
 - No es un lenguaje propietario → No hay que pagar licencia.
- ¿Qué no es?
 - Orientado a objetos (estrictamente). No tiene herencia.
 - No tiene compilador → No hay errores en tiempo de compilación.
 - No es Java → No tiene nada que ver con Java.
 - Aunque parece que la parte de Java del nombre viene de la “moda”.
 - ¿Seguro?
<http://www.2ality.com/2011/03/javascript-how-it-all-began.html>
 - Un lenguaje de propósito general, rápido y potente.

¿Qué puedo hacer con JavaScript?

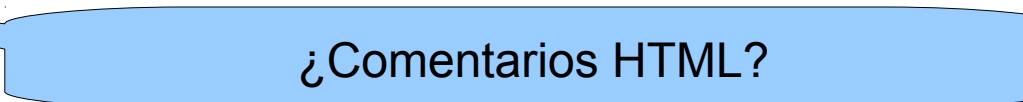
- Crear contenido en una página.
- Modificar contenido de una página.
- Reaccionar a eventos (P.ej. Ratón).
- Validar datos (Formularios).
- Detectar el navegador del usuario (y reaccionar en función de ello).
- Crear, leer y manipular cookies.
- Y miles de cosas más...



Primer programa en JavaScript

EJ1

```
<html>
  <body>
    <script type="text/javascript">
      <!--
        document.write("Hola Mundo");
      //-->
    </script>
  </body>
</html>
```



¿Comentarios HTML?

Entrada y salida básica

EJ2

```
<html>
  <body>
    <script type="text/javascript">
      <!--
        var nombre = prompt(
          "Dime tu nombre", "");
        document.write(  "Hola " + nombre);
      //-->
    </script>
  </body>
</html>
```

- Punto y coma opcional (si la sentencia se divide en varias líneas es necesario). Incluir el punto y coma es una buena práctica.

3.1.2 El lenguaje JavaScript

- Variables
- Palabras reservadas
- Literales
- Caracteres de escape
- Operadores
- Precedencia de operadores
- Conversión de tipos
- Comentarios

Variables

- Nombres de variables:
 - Deben comenzar por una letra o guión bajo.
 - El resto pueden ser cifras, letras o guiones bajos.
 - No puede coincidir con una palabra reservada.
- JavaScript es un lenguaje sensible a mayúsculas → Los nombres de las variables también.
- Para **declarar una variable** se emplea la palabra reservada *var*:
`var variable;`
 - Cuidado porque funcionan sin declarar
- El valor inicial será *null* / *undefined* si no se inicializa:
`var variable=5;`
- No se indica el tipo de la variable. Se adapta automáticamente.

Palabras reservadas

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

Algunas más:

- Dependientes del navegador.
- Nombres de algunos objetos comunes.

<http://www.javascripter.net/faq/reserved.htm>

Literales

- Números:
 - Enteros:
 - Decimales: 54, -89
 - Hexadecimales: 0xFA34, -0xB0A3
 - Octales: 0769, -0564
 - Reales:
 - Simples: 6.78
 - Notación científica: 6.34E5, -45E-10
- Cadenas de caracteres: Comillas dobles o simples.
 - “Hola”.
 - ‘Cadena’.
 - Pueden alternarse las comillas para usarlas dentro de la cadena: ‘Ella dijo “Hola”’, “Juan me dijo ‘adelante’”.
- Valores booleanos: *true*, *false*.



Sin significado especial

Caracteres de escape

Character	Meaning
\b	Backspace
\f	Form feed
\n	New line
\r	Carriage return
\t	Tab
\'	Single quote or apostrophe (')
\"	Double quote (")
\\	Backslash (\)
\XXX	XXX is an octal number
\xXX	XX is a hexadecimal number

Operadores aritméticos

Operator	Description	Example	Result
+	Addition	x=2 y=2 x+y	4
-	Subtraction	x=5 y=2 x-y	3
*	Multiplication	x=5 y=4 x*y	20
/	Division	15/5 5/2	3 2.5
%	Modulus (division remainder)	5%2 10%8 10%2	1 2 0
++	Increment	x=5 x++	x=6
--	Decrement	x=--y	x=4

Operadores de asignación y comparación

Operator	Example	Is The 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

Operator	Description	Example
==	is equal to	5==8 returns false
===	is equal to (checks for both value and type)	x=5 y="5" x==y returns true x===y returns false
!=	is not equal	5!=8 returns true
>	is greater than	5>8 returns false
<	is less than	5<8 returns true
>=	is greater than or equal to	5>=8 returns false
<=	is less than or equal to	x<=8 is true

Operadores lógicos

Operator	Description	Example
&&	and	x=6 y=3 (x < 10 && y > 1) returns true
	or	x=6 y=3 (x==5 y==5) returns false
!	not	!(x==y) is true

Operación con cadenas

Operator	Description	Example
+	concatenation	"Hello "+ "World" returns "Hello World"
==	is equal to	"a"=="b" returns false "hi"=="hi" return true
!=	is not equal	"hi"!="hello" returns true
>	is greater than	"hello">"hi" returns false
<	is less than	"hello"<"hi" returns true
>=	is greater than or equal to	"hello">="hi" returns false
<=	is leaser than or equal to	"hello"<= "hi" returns true

Precedencia de operadores

A igual nivel, se evalúa de izquierda a derecha

1. Incremento/decremento
2. Lógico NOT/Signos negativo y positivo
3. Multiplicación/División/Módulo
4. Suma/Resta
5. Operadores relacionales
6. Lógico AND
7. Lógico OR
8. Asignación

https://developer.mozilla.org/en-US/docs/JavaScript/Reference/Operators/Operator_Precedence

Conversión entre tipos

- Conversión automática al operar.
- De cadena a numérico:
 - Sumar 0
 - `var cad="123"; var num = cad + 0;`
- De número a cadena:
 - Concatenar la cadena vacía
 - `var num=123; var cad = num + "";`

Comentarios

- **Una sola línea:**

```
// Comentario de una sola línea  
var a = 5+b;
```

- **Multilínea:**

```
/* Esto es un comentario  
multilínea. Las líneas son  
ignoradas */  
var a = 5+b;
```

3.1.3 Programación estructurada

- Estructura condicional:
 - Simple
 - Caso contrario
 - Compleja
 - Basada en casos
- Bucles:
 - Simple
 - Postcondición
 - Contador

Estructura condicional simple

```
<script type="text/javascript">
  var d=new Date();
  var time=d.getHours();
  if (time<12) {
    document.write(
      "<b>Buenos dias</b>"
    );
  }
</script>
```


Estructura condicional caso contrario

```
<script type="text/javascript">
  var d=new Date();
  var time=d.getHours();
  if (time<10) {
    document.write("<b>Buenos dias</b>");
  }
  else{
    document.write("<b>Buenas tardes</b>");
  }
</script>
```

Estructura condicional compleja

```
<script type="text/javascript">
    var d=new Date();
    var time=d.getHours();
    if (time<10) {
        document.write("<b>Buenos dias</b>");
    }
    else if (time<20) {
        document.write("<b>Buenas tardes</b>");
    }
    else{
        document.write("<b>Buenas noches</b>");
    }
</script>
```

Estructura condicional basada en casos

```
<script type="text/javascript">
  var d=new Date();
  var theDay=d.getDay(); // 0 domingo, 1 lunes, ...
  switch (theDay) {
    case 5:
      document.write("Viernes al fin!");break;
    case 6:
      document.write("Sabado!");break;
    case 0:
      document.write("Descanso dominguero");break;
    default:
      document.write("Esperando el finde...");
  }
</script>
```

Bucle simple

EJ3

```
<html>
  <body>
    <script type="text/javascript">
      var i=0;
      while (i<=10) {
        document.write("El numero es " + i);
        document.write("<br />");
        i=i+1;
      }
    </script>
  </body>
</html>
```

Bucle postcondición

EJ4

```
<html>
  <body>
    <script type="text/javascript">
      var i=0;
      do {
        document.write("El numero es " + i);
        document.write("<br />");
        i=i+1;
      } while(i<0);
    </script>
  </body>
</html>
```

Bucle con contador

EJ5

```
<html>
  <body>
    <script type="text/javascript">
      for(var i=0;i<=10;i++){
        document.write("El numero es " + i);
        document.write("<br />");
      }
    </script>
  </body>
</html>
```

3.1.4 Vectores

- Vectores unidimensionales
- Vectores multidimensionales
- Recorrido de vectores

Vectores

Alternativa no recomendada
porque complica la sintaxis

```
<html>
  <body>
    <script type="text/javascript">
      var mycars = [] // new Array();
      mycars[0] = "Saab";
      mycars[1] = "Volvo";
      mycars[2] = "BMW";

      for (i=0; i<mycars.length; i++) {
        document.write(mycars[i] + "<br />")
      }
    </script>
  </body>
</html>
```

Se puede indicar el
tamaño del vector
por si queremos
prereservar

- **Se pueden indicar los elementos en el momento de creación**
http://www.w3schools.com/js/tryit.asp?filename=tryjs_array

Matrices

- <http://www.desarrolloweb.com/articulos/ejemplos/javascript/arraymultidimensional.html>
- Se construyen como vectores de vectores



¿Hay alguien que no es familiar con este concepto?

Recorrido de vectores

EJ6

```
<html>
  <body>
    <script type="text/javascript">
      var mycars = new Array();
      mycars[0] = "Saab";
      mycars[1] = "Volvo";
      mycars[2] = "BMW";
      for (var x in mycars) {
        document.write(mycars[x] + "<br />");
      }
    </script>
  </body>
</html>
```

Cuidado porque
esto en algunos casos
puede ser una mala idea

En caso de duda... Nuestro amigo lenght de toda la vida
http://www.w3schools.com/js/tryit.asp?filename=tryjs_array_loop

3.1.5 Funciones

- Declaración
- Invocación
- Parámetros
- Devolución de valores
- Recursividad

Funciones: Declaración e invocación

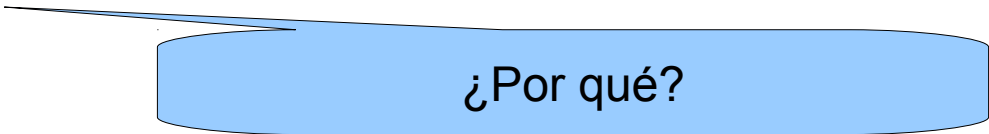
EJ7

```
<html>
  <body>
    <script type="text/javascript">
      function fecha_de_hoy() {
        var fecha = new Date();
        document.write(fecha.getDate()+"-");
        document.write(fecha.getMonth()+1+"-");
        document.write(fecha.getFullYear()+1900);
      }

      document.write("Bienvenido, hoy es: ");
      fecha_de_hoy();
    </script>
  </body>
</html>
```

Parámetros

- Sin tipo
- Paso por valor (excepto cuando se pasan objetos).



¿Por qué?

```
function fecha_de_hoy(separador) {  
    var fecha = new Date();  
    document.write(fecha.getDate()+separador);  
    document.write(fecha.getMonth()+1+separador);  
    document.write(fecha.getFullYear()+1900);  
}
```

Valor devuelto

- Sin tipo

```
function prod(a,b) {  
    x=a*b;  
    return x;  
}  
  
...  
var resultado = prod(5,3);
```

Recursividad

- Está permitida la recursividad
 - ¡Cuidado con el tamaño de la pila!

EJ8

```
<html>
  <body>
    <script type="text/javascript">
      function factorial(n){
        if(n!=1) return n*factorial(n-1);
        else return 1;
      }

      var n = prompt("Introduce un número","");
      document.write(
        "El factorial de "+n+" es " + factorial(n));
    </script>
  </body>
</html>
```

4.1.6 Ámbitos de código y variables

- Dónde colocar el código
- Tipos de variables
- Detección del tipo de variables

Dónde colocar el código Javascript

- En la cabecera:
 - Para funciones y variables globales.
- Cuerpo:
 - Llamadas a funciones y código que se desea ejecutar a la carga de una determinada parte de la página.
- Archivo externo:
 - Librerías de funciones / código compartido con otras páginas

Dónde colocar el código Javascript (2)

Ej31

<html>

<head>

<script type="text/javascript">

function fecha_de_hoy(){

var fecha = new Date();

document.write(fecha.getDate()+"-");

document.write(fecha.getMonth()+1+"-");

document.write(fecha.getFullYear()+1900);

}

</script>

</head>

<body>

Bienvenido, hoy es:

<script type="text/javascript">

fecha_de_hoy();

</script>

</body>

</html>

Declaración

Uso

Librerías externas

```
function fecha_de_hoy(){  
    var fecha = new Date();  
    document.write(fecha.getDate()+"-");  
    document.write(fecha.getMonth()+1+"-");  
    document.write(fecha.getFullYear()+1900);  
}
```

EJ32.html

<html>

<head>

<script type="text/javascript" src="ej32.js"></script>

</head>

<body>

Bienvenido, hoy es:

<script type="text/javascript">
 fecha_de_hoy();
</script>

</body>

</html>

Autocierre de la etiqueta script

- `<script src="archivo.js" type="text/javascript" />`
 - Sólo lo acepta chrome.
 - Es un error según el estándar para XHTML
 - http://www.w3.org/TR/xhtml1/#C_3
- `<script src="archivo.js" type="text/javascript">`
`</script>`
 - Lo estándar y ampliamente aceptado.

Tipos de variable

- Tipos de variable
 - Numéricas: Enteros y reales.
 - Booleanas.
 - Cadenas
 - *Funciones*
 - *Objetos*
- Ámbito:
 - Global: Toda la página.
 - Local: Función o bloque (estructura de control).
- **No es necesario declarar una variable antes de su uso (uso de *var*), pero entonces será una variable global (Mala práctica).**

Detección de tipos

- Operador *typeof* que devuelve una cadena indicando el tipo de variable:
 - ‘number’
 - ‘string’
 - ‘boolean’
 - ‘function’
 - ‘object’
 - ‘undefined’. Sin inicializar

Detección de tipos

```
var variable
...
if(typeof variable=='number')
    variable++;
else if(typeof variable=='string')
    variable = variable + ' ' + variable;
...
```

3.1.7 Funciones propias del lenguaje

- *isNaN*: Detección de número inválidos.
- *parseInt*: Lectura de un número desde una cadena.
- *parseFloat*: Lectura de un real desde una cadena.

Detección de números inválidos

EJ11

```
<html>
  <body>
    <script type="text/javascript">
      var num="";
      do{
        num=prompt("Introduce un número",num);
      }while(isNaN(num));
      document.write("El numero es: " + num);
    </script>
  </body>
</html>
```

Lectura de números desde cadenas

EJ12

```
<html>
  <body>
    <script type="text/javascript">
      var cad=prompt("Introduce un número", "");
      var entero = parseInt(cad);
      var real = parseFloat(cad);
      document.write("El numero es: " +
        entero + " y/o " + real);
    </script>
  </body>
</html>
```

- `parseInt` puede tener un argumento adicional para indicar la base {16,10,8,2}
- En cuando no se puede convertir la cadena se ignora el resto.
- Si el primer carácter no se puede convertir se devuelve el valor *NaN*.

Ejercicio 1

- Cree un script en JS que pida al usuario, mediante un cuadro de diálogo, un número.
- El programa seguirá pidiendo números hasta que se introduzca una expresión no numérica.
- Al finalizar el programa mostrará cada uno de los números introducidos en dos tablas. La primera contendrá los números que se han introducido que son primos y la otra los que no.
- Se valorará la claridad y comentarios del código, así como el formato de la salida generada (p. ej. usar tablas y algo de CSS – sin pasarse -).