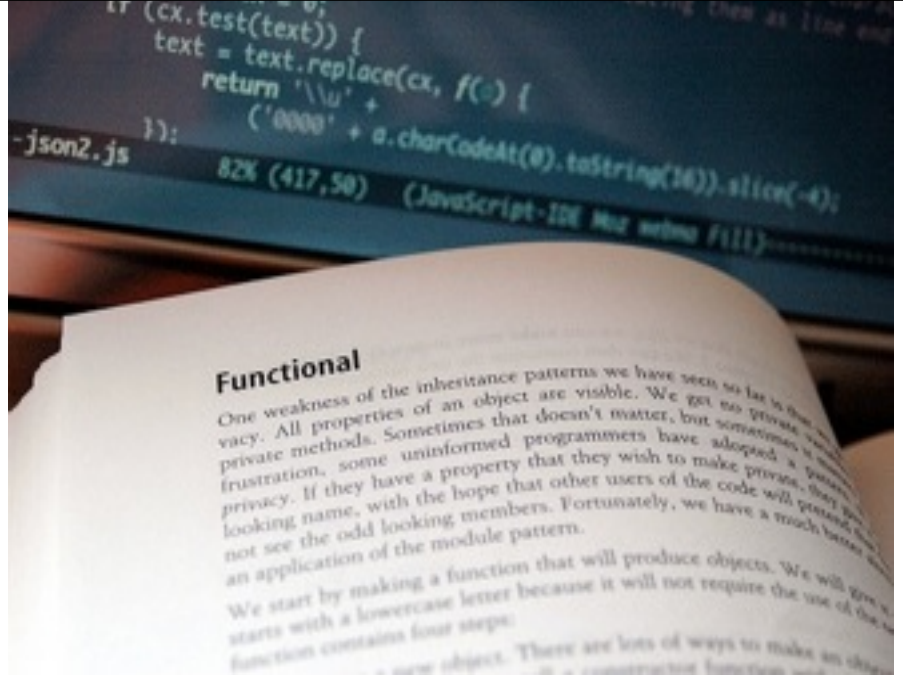
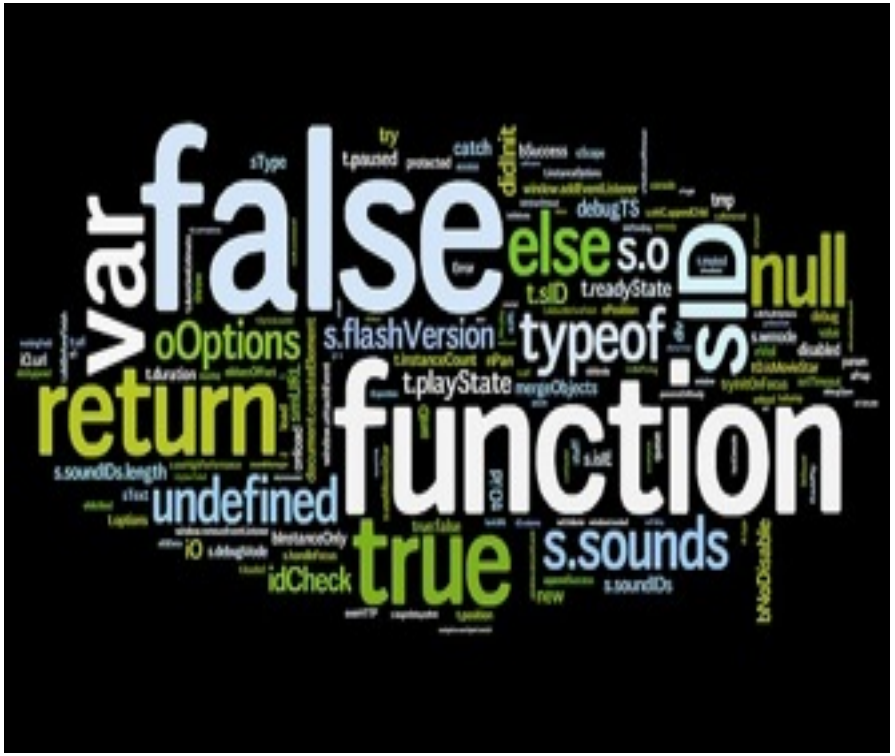
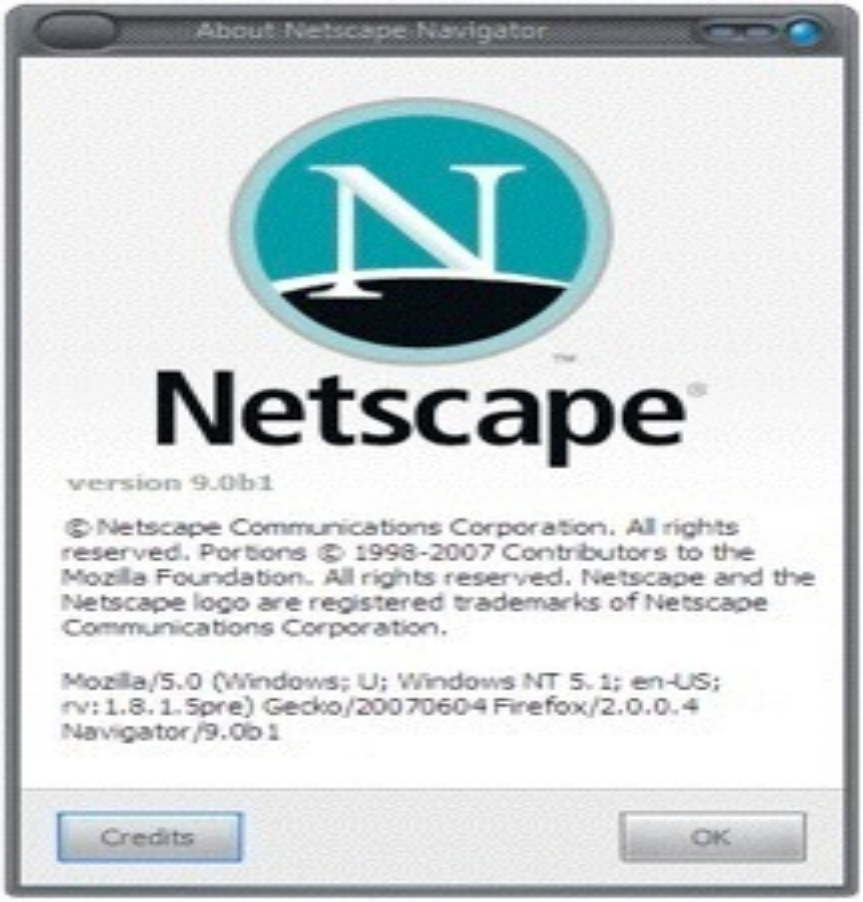




Tema 3

JavaScript

Parte I:
El núcleo del lenguaje.
Interfaz básico con el navegador.



Introducción a Javascript

3.1

Conceptos previos

¿Qué es javascript? ¿En qué se diferencia de otros lenguajes de programación web?

JavaScript

- Javascript tiene
  - Sintaxis superficialmente similar a C++ o Java, mucho menos restrictiva
  - Visible el código fuente, ya que es interpretado, no compilado
    - No obstante, el fuente se puede ofuscar
    - Las implementaciones modernas hacen una compilación JIT (*just-in-time*) a código máquina
  - Orientación a objetos:
    - Distinta filosofía que C++ o Java: en JavaScript no existen clases.
    - Los objetos son colecciones de métodos y propiedades.
- Cuidado, no es del todo lo que parece. Parece C “raro” o “Java para no-programadores”, pero introduce elementos sofisticados de varios paradigmas “alternativos”
  - Programación funcional: *lambdas*, *closures*
  - OO no convencional: herencia basada en prototipos

¿Qué se puede hacer con JavaScript?

- Se puede usar para:
  - Interactuar con el navegador: abrir ventanas, saltar a otra URL,…
  - Interactuar con el documento HTML: verificar formularios, añadir/eliminar contenido, hacer animaciones…
  - Abrir conexiones con el servidor (AJAX)
  - Dibujar en la página, en 2D e incluso en 3D
- No se suele usar para :
  - Trabajar directamente con bases de datos. Esto se suele hacer desde el servidor, que accede él mismo y nos devuelve los datos.
- Estas capacidades no vienen del lenguaje en sí, sino de la forma en la que está actualmente integrado en los navegadores
  - De hecho, existen librerías para acceder a bases de datos locales con SQL

```
var Torrent = Class.create();

Torrent.prototype = {
  initialize: function(xml) {
    var xmlString = new XMLSerial
    var torrentNode = xml.getElem
    var statsNode   = xml.getElem

    this.objectID = getText(torre
    this.name = getText(torrentNo
    this.displayName = (this.name
                        this.name
                        this.name
                        this.name
    this.hash = getText(torrentNo
```


Introducción a Javascript

3.2

El núcleo de Javascript

JavaScript como lenguaje, independientemente de la web

El léxico de JavaScript

- JavaScript es descendiente de la familia C, C++, Java
  - Diferencia mayúsculas/minúsculas
  - Comentarios tipo C/C++ /* Comentario */ //comentario

- El **;** es opcional (si cada sentencia está en una línea), aunque es recomendable para evitar problemas, ya que el intérprete lo inserta automáticamente

```
a = 3
b = 4
```

Declaración de variables

- No obligatoria, aunque aconsejable. Si se usa una variable sin declarar, se asume implícitamente como **global**
- Las variables no tienen tipo fijo, se declaran simplemente con la palabra clave **var**
- El valor de una variable declarada pero no inicializada es un valor especial llamado undefined

```
var a;
b = 27;          //válido
a = 3;
a = "hola";    //válido
var c,d;
console.log(c)   //undefined
console.log(d)   //error
```

Tipos de datos

- Numérico (enteros y reales)
- Booleano (`true==1, false==0`)
- Clases básicas del sistema
  - String
  - Object (tipo base para los objetos)
  - function
  - Array
  - Date
  - RegExp (expresión regular)

Operadores y sentencias

- Idem a C++/Java
  - Operadores aritméticos y lógicos (eso sí, no se pueden redefinir)
  - Sentencias: bucles, condicionales, etc.
- Igualdad (==): emplea conversión de tipos
- Identidad(===): sin conversión

```
if ("1"==true)     //esto es cierto
...
if ("1"===true)     //pero esto no
```

- **delete** existe, pero no significa lo mismo que en C++, lo veremos cuando hablemos de objetos

Operador de asignación

- Datos primitivos: por valor (copia)
- Objetos: por referencia (ambas variables apuntan al mismo objeto)
  - Aunque existen punteros, no se pueden emplear como en C++ (desplazarse por la memoria, obtener la direccion, etc.)

```
var a,b;
a = new Array();   // un array es un objeto que se
                   // crea con el constructor Array()
a[0] = 1;
b = a;             // b referencia al array a
a[0] = 100;
alert(b[0]);       // muestra el valor 100
```

Funciones

- Se definen con la palabra clave **function**
- Los parámetros no tienen tipo (como es lógico)
- Es recomendable declarar las variables locales, para no coincidir con una global

```
var res = 1;
function suma (arg1, arg2)  {
      var res = arg1 + arg2;
      return res;
}
```

- Las funciones son objetos, y por tanto se pueden asignar a variables o pasar como parámetros

```
function operar(arg1,arg2,op) {
  return op(arg1,arg2)
}
operar(2,2,suma);
```

Trabajo con objetos

- Creación, inserción de propiedades y borrado dinámico

```
var Homer;
Homer = new Object();
Homer.nombre = "Homer Simpson";
Homer.edad = 34;
Homer.casado = true;
delete Homer.edad    //(Homer.edad==undefined)
```

- Acceso

```
Homer.edad = 40;      //Notación "clásica"
Homer["edad"] = 40;  //Como si fuera un array
var prop = prompt("¿Qué propiedad quieres?");
alert(Homer[prop]);
```

Literales

- Permiten definir un objeto como un conjunto de propiedades `nombre:valor` separadas por comas
  - Si el nombre de la propiedad contiene espacios, guiones, etc. o es una palabra reservada, debe ir entre comillas

```
var homer = {
  “nombre completo”: "Homer Simpson",
  tels: [                  //Los arrays se definen con el corchete
    "555-123456",
    "555-654321"
  ],
  ocupacion: {    //Una propiedad puede ser un objeto
      puesto: "operario",
      lugar: "central nuclear de Springfield"
  }
}
```

JSON

- Notación basada en literales para poder representar objetos como texto plano
  - Ciertos valores no son representables de modo estándar, como `undefined`, fechas, expresiones regulares,...
- Formato estándar para intercambiar objetos con el servidor en aplicaciones AJAX

```
var json = '{"nombre":"Homer", "edad": 34}'
var homer = JSON.parse(json)          //cadena JSON -> objeto
//idem pero peligroso, eval permite ejecutar código arbitrario
var homer = eval("(" + json + ")")
console.log(JSON.stringify(homer))  //objeto->cadena JSON
```

Constructores

- **Cualquier función** se puede usar como un **constructor**, invocándola con `new`
  - En ese caso, `this` en la función referencia al objeto recién creado
  - Cuidado, si nos olvidamos de poner el `new`, `this` es el objeto global

```
//Se usa la convención de mayúsculas porque
//vamos a usar la función como constructor
//pero es una función normal y corriente
function Persona(nombre) {
  this.nombre = nombre
  //recordar que las funciones son objetos
  this.saludar = function() { return "hola"}
}

var p1 = new Persona("Pepe")
console.log(p1.nombre)    //Pepe
console.log(p1.saludar()) //hola
```

Prototipos

- **Todas las funciones** tienen una propiedad por defecto llamada `prototype`, inicialmente vacía, es un objeto al que se pueden añadir propiedades

```
function Persona(nombre) {
   this.nombre = nombre
}

Persona.prototype.saludar = function() {
   return "hola"
}
```

- Todos los objetos construídos con la función “heredan” el prototipo.
- Al referenciar una propiedad si el objeto no la tiene directamente se busca en el prototipo

```
var p1 = new Persona(“pepe”)
console.log(p1.nombre)     //pepe
console.log(p1.saludar())  //hola
```

Extender los objetos nativos

- Podemos extender la funcionalidad de los objetos del sistema usando `prototype`

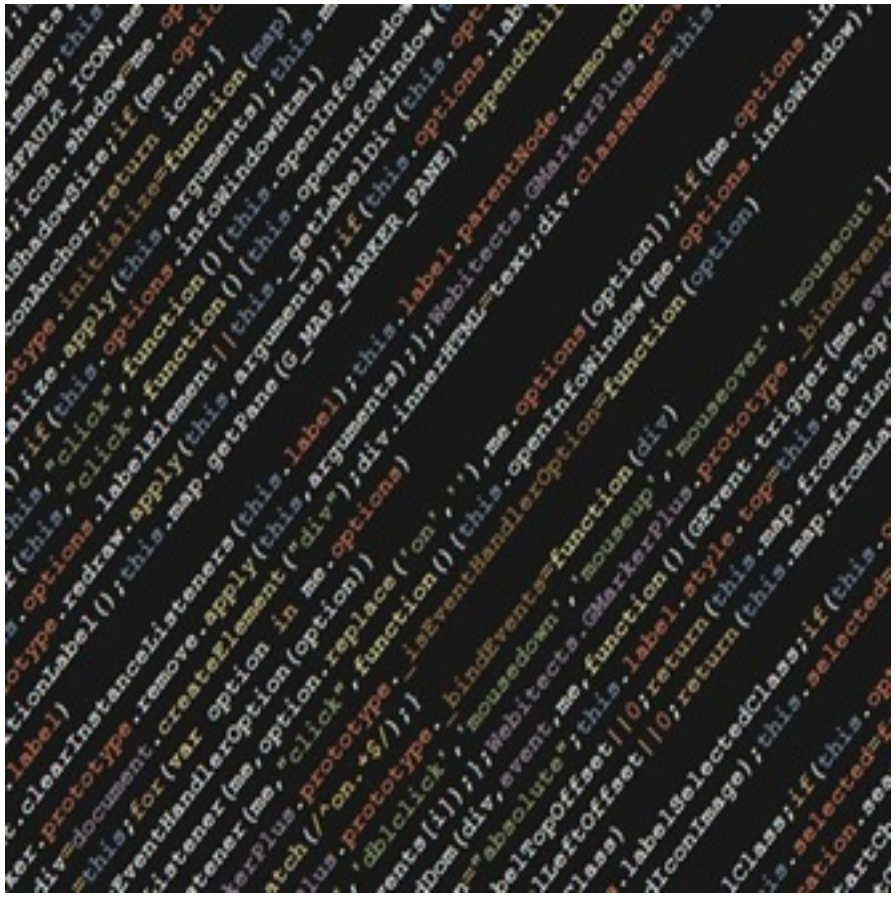
```
Date.prototype.formatear = function() {
            return this.getDate()+"/"+(this.getMonth()
+1)+"/"+this.getFullYear();
}
```

```
//El constructor sin parámetros devuelve la fecha actual
ahora = new Date();
console.log(ahora.formatear());
```

Arrays

- **Dispersos**: no todas las posiciones tienen que contener datos. Las vacías tienen **`undefined`**
- **Heterogéneos**: cada posición puede tener un tipo distinto

```
var a=new Array()
a[5]=4
a[7]="hola"
alert(a.length)    //8!!
```



3.3

Scripts en páginas web

¿Cómo metemos código Javascript en el HTML?

Introducción a Javascript

Insertar Javascript en el HTML

- En etiquetas **script**
- El ámbito de las variables es la página entera
  - Las variables no se pueden compartir entre páginas
- El código se ejecuta a medida que se va leyendo

```
<html>
<head>
    <script type="text/javascript">  //type obligatorio en HTML4
        function ahora() {             //pero no en HTML5
           var h = new Date();
           return h.toLocaleString();   }
        var verFecha = true;
   </script>
</head>
<body>
   <script type="text/javascript">
        if (verFecha)  alert("Fecha y hora: " + ahora());
   </script>
</body>
</html>
```

Otras formas de incluir código

- En un fichero aparte (.js). Similar al #include de C

```
<script src="prog.js"> </script>
```

  - Se recomienda ponerlo al final de la página para que el navegador ya haya renderizado algo y no aparezca en blanco mientras se carga el script
- En un manejador de evento
  - El código se ejecuta al producirse el evento, no al leer la página

```
<input type="button" value="pulsa aqui"
       onClick="alert('hola')">
```

- Como una URL **javascript:**
  - Para poder ejecutar código en respuesta a un click en un enlace

```
<a href="javascript:alert('hola')"> ¡Hola! </a>
```

Introducción a Javascript

3.4

Interfaz con el navegador (Browser Object Model o BOM)

API orientado a objetos para interactuar con el navegador

APIs de JavaScript para la web

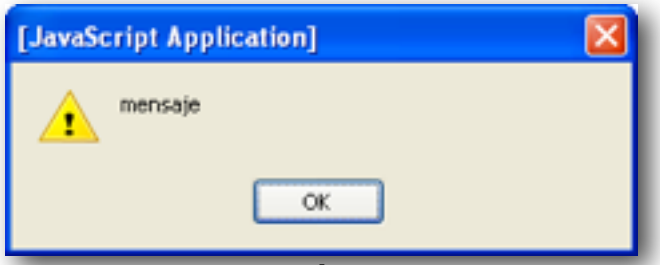
- En web nos puede interesar interactuar con:
  - El navegador:
    - Obtener las propiedades y capacidades
    - Mover la ventana, abrir un popup (cada vez menos),…
    - Hacer que salte a otra dirección
  - El propio HTML: (el API se llama DOM, lo veremos en las clases siguientes)
    - Añadir texto, etiquetas, quitar etiquetas
    - Reordenar partes del documento (por ejemplo una tabla por columnas)
    - Hacer animaciones

Objeto global: window

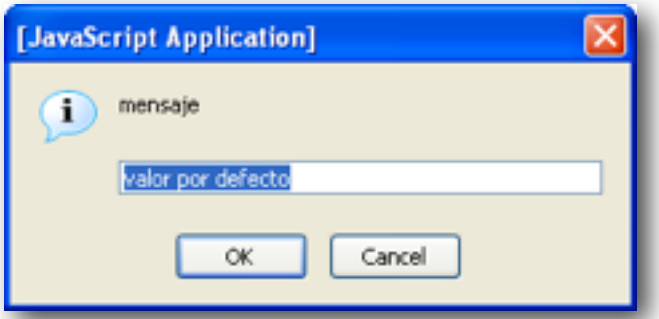
- El objeto **window** determina el contexto de ejecución
  - Las variables "globales" son propiedades de `window`

```
a = 7;
alert (window.a)    //7
```

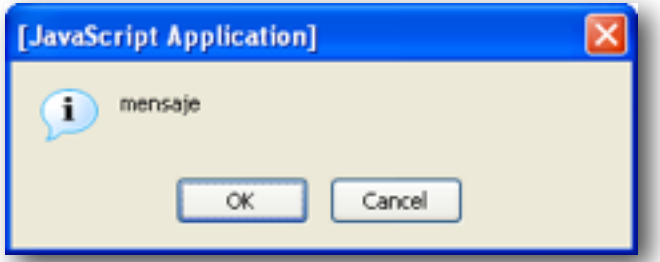
- Cuadros de diálogo modales
  - `alert(mensaje)`
  - `prompt(mensaje, valor_por_defecto)`: devuelve cadena introducida o `null` si se ha pulsado cancelar
  - `confirm(mensaje)`: devuelve `true` o `false` según si se pulsa Aceptar o Cancelar



alert



prompt



confirm

Objeto window: algunas propiedades

Propiedad	Significado
`closed`	valor booleano que indica si la ventana ha sido cerrada
`status`	texto de la barra de estado (para un mensaje temporal)
`defaultStatus`	texto por defecto de la barra de estado (es reemplazado momentáneamente cuando se pasa el ratón por un enlace u otros objetos)
`name`	nombre de la ventana
`opener`	referencia al objeto `window` que creó la ventana actual
`parent`	si la ventana actual es un frame, referencia a la ventana que lo contiene. En caso contrario, es lo mismo que `window`
`top`	si la ventana actual es un frame, referencia a la ventana de nivel superior que lo contiene. En caso contrario, es lo mismo que `window`
`self, window`	la propia ventana

Objeto navigator

Propiedad	Significado
`appName`	Nombre común del navegador. Ejemplos: `Netscape`, `Microsoft Internet Explorer`
`appVersion`	Número de versión e información adicional. Ejemplo: en Navigator 4.6, versión inglesa para Windows tiene el valor: `4.6 [en] (Win98; I)`
`userAgent`	La información que envía el navegador en la cabecera http `USER-AGENT`

- En el pasado, el objeto Navigator se usaba para ejecutar un código u otro dependiendo de la compatibilidad.
  - Pero eso era cuando solo existían Navigator y Explorer
  - En la actualidad es más sencillo comprobar si el método o propiedad que necesitamos existe. Si no, será **`undefined`** (==false)

```
if (document.all) {
      …     //Estamos en Internet Explorer
}
```

Objeto location

Propiedad	Significado
`href`	cadena que representa la URL completa
`protocol`	solo la parte del protocolo (ej, http:)
`host`	solo el nombre del host
`pathname`	trayectoria hasta el recurso, incluido el mismo
`search`	parte de la URL que sigue al carácter "?",(incluido) si está presente.

- Por ejemplo, para hacer que el navegador salte a la página "login.htm"

```
location.href="login.htm"
```

Referencias

- **“Javascript enlightenment”** Cody Lindley
  - Disponible en http://www.javascriptenlightenment.com
- **“Javascript: the definitive guide”**, David Flanagan
  - La referencia más clásica y exhaustiva sobre Javascript, cubre no solo el lenguaje, también los APIs de la web
- **“Secrets of the Javascript Ninja”**, John Resig
  - Desde lo básico a conceptos avanzados. Muy bien explicado por John Resig, el creador de jQuery