



SILVANO GIL

- **Introducción a AJAX**

- Introducción a la Web
- Componentes de AJAX
- Usos de AJAX

- **Características de AJAX**

- Cómo funciona AJAX
- Beneficios de Ajax
- Elección del Framework

- **Elementos básicos AJAX**

- DOM
- Selectores y DOM
- Manejo de nodos
- JSON
- XML
- Javascript y DOM

- **Ajax Accesible (hijax)**

- Javascript y Limitaciones
- Hijax

- **Firebug**

- Manejo de DOM
- Uso de Red
- Manejo de Profiling

- **Ajax y Frameworks**

- JQuery
- Ext-JS



# Introducción a la Web

---

- El término AJAX se presentó por primera vez en el artículo "[Ajax: A New Approach to Web Applications](#)" publicado por Jesse James Garrett el 18 de Febrero de 2005. Hasta ese momento, no existía un término normalizado que hiciera referencia a un nuevo tipo de aplicación web que estaba apareciendo.
- En realidad, el término AJAX es un acrónimo de *Asynchronous JavaScript + XML*, que se puede traducir como "JavaScript asíncrono + XML".
- El artículo define AJAX de la siguiente forma:  
*Ajax no es una tecnología en sí mismo. En realidad, se trata de varias tecnologías independientes que se unen de formas nuevas y sorprendentes*

## Introducción al AJAX

---

# **Componentes de Ajax**

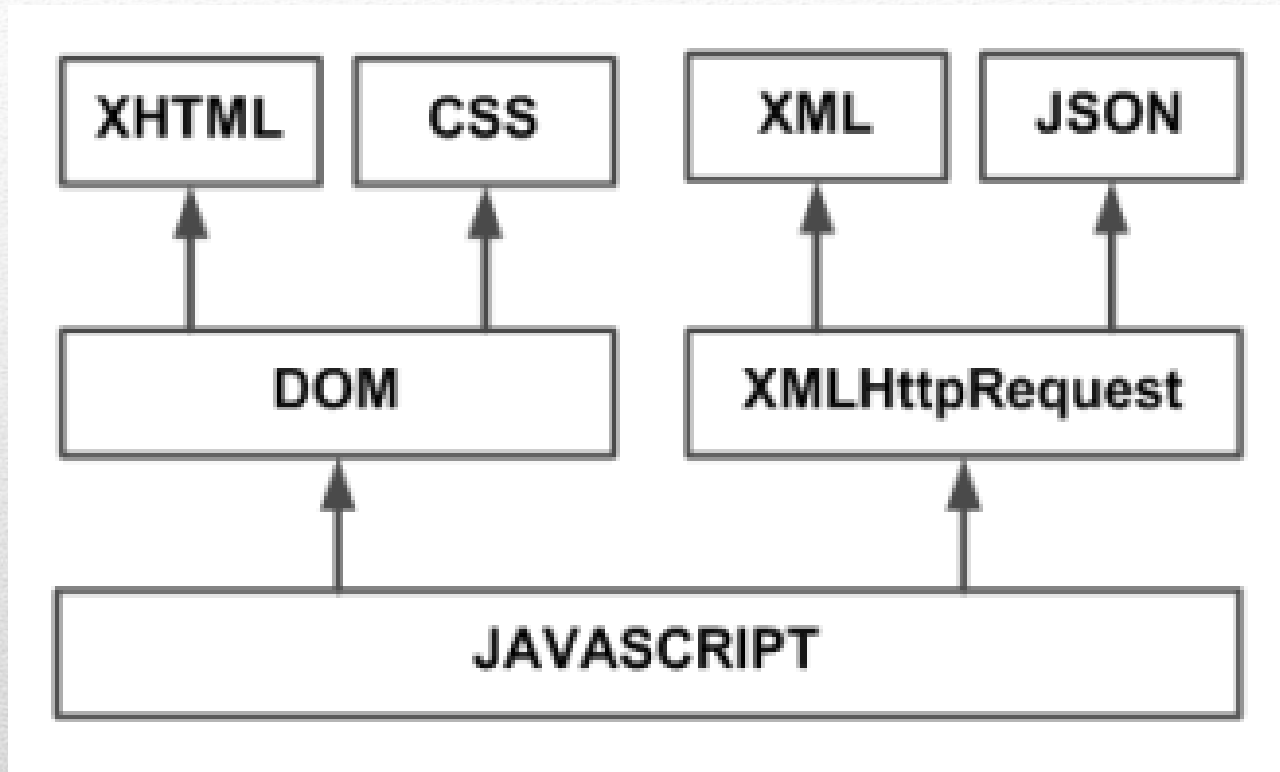
- Las tecnologías que forman AJAX son:
  - XHTML y CSS, para crear una presentación basada en estándares.
  - DOM, para la interacción y manipulación dinámica de la presentación.
  - XML, XSLT y JSON, para el intercambio y la manipulación de información.
  - XMLHttpRequest, para el intercambio asíncrono de información.
  - JavaScript, para unir todas las demás tecnologías.

## **Introducción al AJAX**

---



# Componentes de Ajax



## Introducción al AJAX

# Usos de Ajax

- Desde su aparición, se han creado cientos de aplicaciones web basadas en AJAX. En la mayoría de casos, AJAX puede sustituir completamente a otras técnicas como Flash. Además, en el caso de las aplicaciones web más avanzadas, pueden llegar a sustituir a las aplicaciones de escritorio.
- A continuación se muestra una lista de algunas de las aplicaciones más conocidas basadas en AJAX:
  - Gestores de correo electrónico: [Gmail](#), [Yahoo Mail](#), [Windows Live Mail](#).
  - Cartografía: [Google Maps](#), [Yahoo Maps](#), [Windows Live Local](#).
  - Aplicaciones web y productividad: [Google Docs](#), [Zimbra](#), [Zoho](#).
  - Otras: [Netvibes](#) [metapágina], [Digg](#) [noticias], [Meebo](#) [mensajería], [30 Boxes](#) [calendario], [Flickr](#) [fotografía].

## Introducción al AJAX



# **Cómo funciona Ajax**

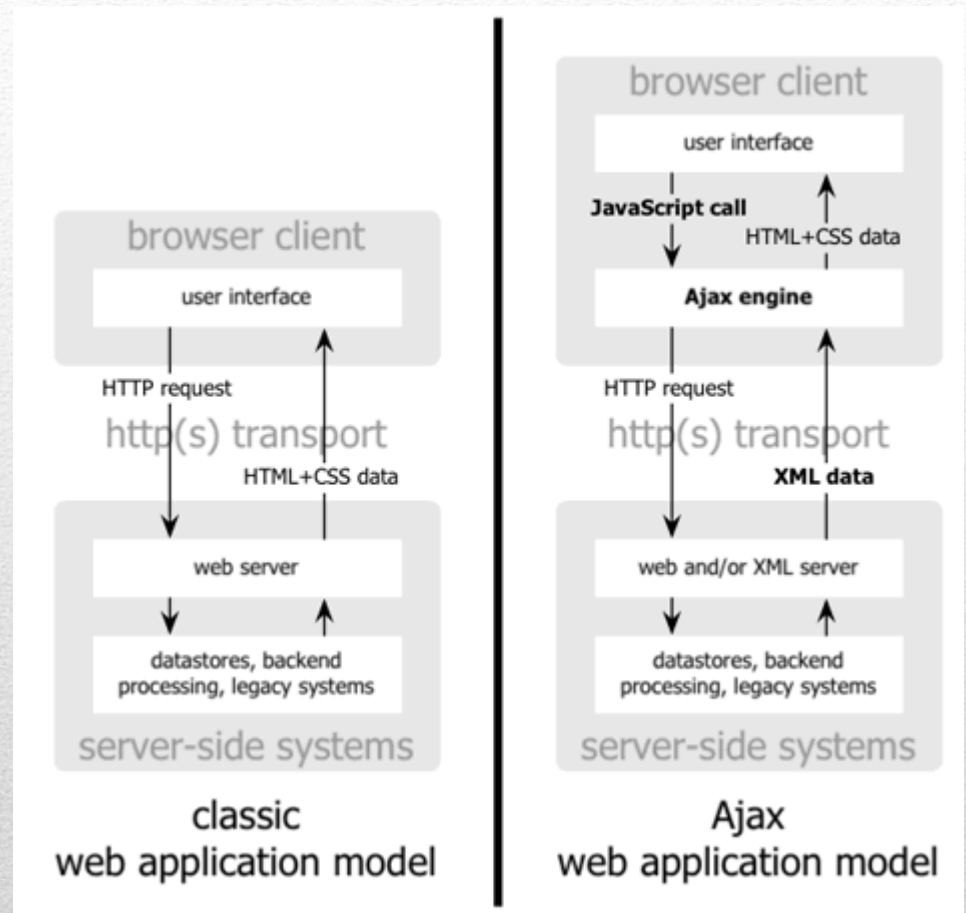
- Desarrollar aplicaciones AJAX requiere un conocimiento avanzado de todas y cada una de las tecnologías anteriores de cada uno de los componentes.
- En las aplicaciones web tradicionales, las acciones del usuario en la página (pinchar en un botón, seleccionar un valor de una lista, etc.) desencadenan llamadas al servidor. Una vez procesada la petición del usuario, el servidor devuelve una nueva página HTML al navegador del usuario.

## **Características de AJAX**

---

# Cómo funciona Ajax

- En el siguiente esquema, la imagen de la izquierda muestra el modelo tradicional de las aplicaciones web. La imagen de la derecha muestra el nuevo modelo propuesto por AJAX:



## Características de AJAX



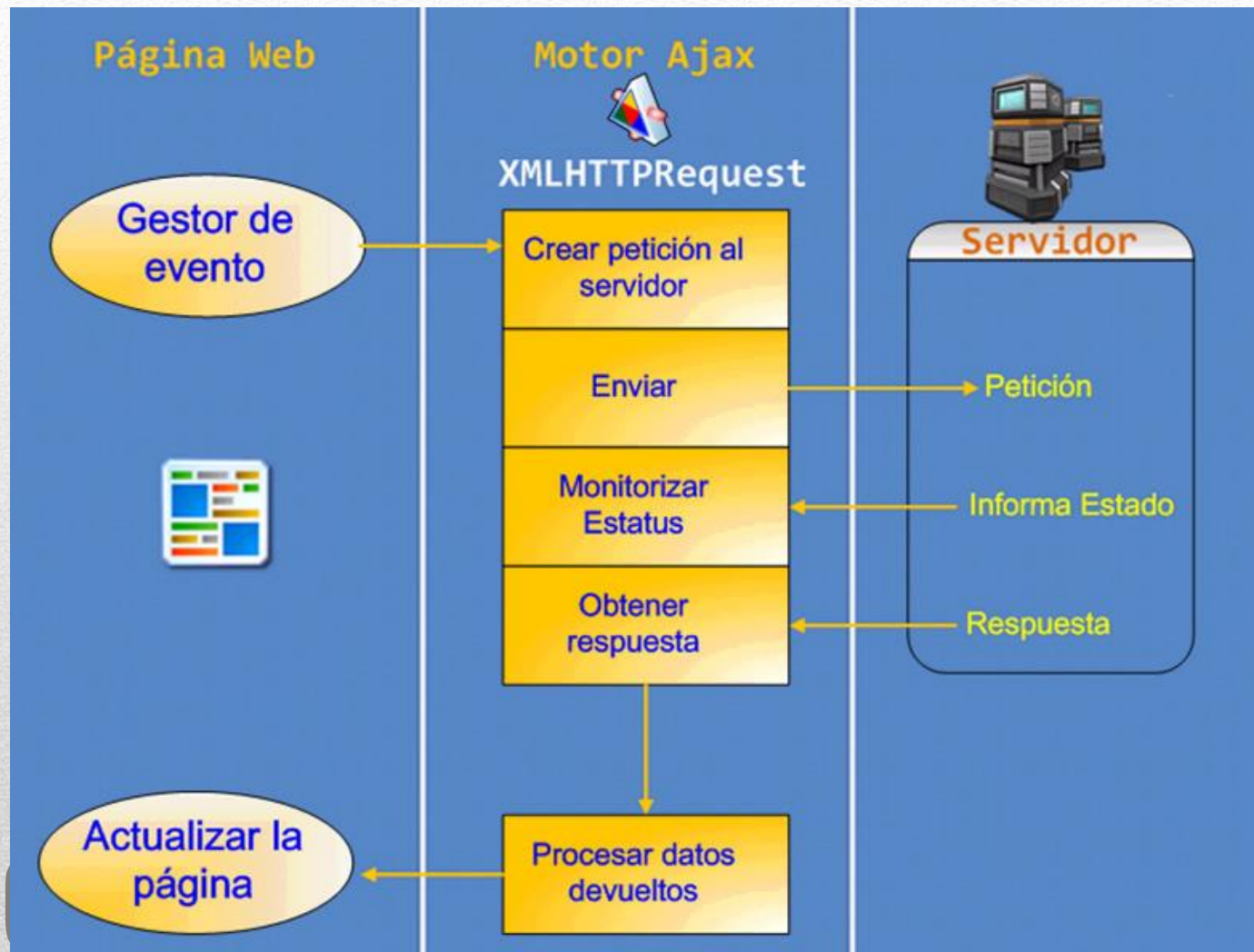
# **Cómo funciona Ajax**

- Esta técnica tradicional para crear aplicaciones web funciona correctamente, pero no crea una buena sensación al usuario. Al realizar peticiones continuas al servidor, el usuario debe esperar a que se recargue la página con los cambios solicitados. Si la aplicación debe realizar peticiones continuas, su uso se convierte en algo molesto.
- AJAX permite mejorar completamente la interacción del usuario con la aplicación, evitando las recargas constantes de la página, ya que el intercambio de información con el servidor se produce en un segundo plano.

## **Características de AJAX**

---

# Cómo funciona Ajax





# Cómo funciona Ajax

## Respuestas del servidor (HTTP Response)

Código de Status	Razón	Explicación
200	OK	Petición correcta
204	No Content	Documento sin datos
301	Moved Permanently	Recurso Movido
401	Not Authorized	Necesita autenticación
403	Forbidden	Rechazada por servidor
404	Not Found	No existe en servidor
408	Request Timeout	Tiempo sobrepasado
500	Server Error	Error en el servidor

# Cómo funciona Ajax

- El objeto **XMLHttpRequest**:
  - Su objetivo es hacer peticiones asíncronas al servidor.
  - Es la columna vertebral de todas las aplicaciones AJAX.
  - Es admitido por todos los navegadores.
  - Microsoft lo introdujo en IE 5 como un objeto ActiveX

## Características de AJAX

---



# Cómo funciona Ajax

## Propiedades del objeto XMLHttpRequest

Propiedades	Descripción
onreadystatechange	Determina que función será llamada cuando la propiedad readyState del objeto cambie.
readyState	Número entero que indica el status de la petición: 0 = No iniciada 1 = Cargando 2 = Cargado 3 = Interactivo 4 = Completado
responseText	Datos devueltos por el servidor en forma de string de texto
responseXML	Datos devueltos por el servidor expresados como un objeto documento.
status	Código estatus HTTP devuelto por el servidor: 200 = OK (Petición correcta) 204 = No Content (Documento sin datos) 301 = Moved Permanently (Recurso Movido) 401 = Not Authorized (Necesita autenticación) 403 = Forbidden (Rechazada por servidor) 404 = Not Found (No existe en servidor) 408 = Request Timeout (Tiempo sobrepasado) 500 = Server Error (Error en el servidor)

# Cómo funciona Ajax

## Métodos del objeto XMLHttpRequest

Propiedades	Descripción
<code>abort()</code>	Detiene la petición actual.
<code>getAllResponseHeaders()</code>	Devuelve todas las cabeceras como un string.
<code>getResponseHeader(x)</code>	Devuelve el valor de la cabecera x como un string.
<code>open('method', 'URL', 'a' )</code>	Especifica el método HTTP (por ejemplo, GET o POST), la URL objetivo, y si la petición debe ser manejada asíncronamente (Si, a='True' defecto; No, a='false'.)
<code>send(content)</code>	Envía la petición
<code>setRequestHeader( 'label' , 'value' )</code>	Configura un par parámetro y valor label=value y lo asigna a la cabecera para ser enviado con la petición.



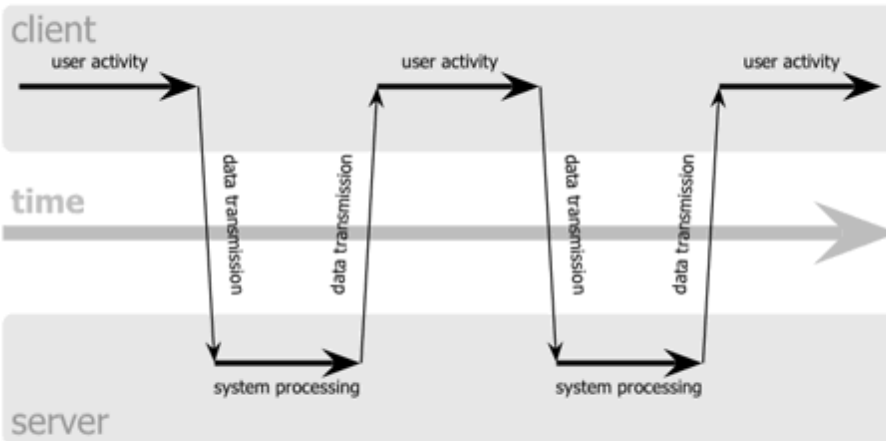
# **Beneficios de Ajax**

- Las aplicaciones construidas con AJAX eliminan la recarga constante de páginas mediante la creación de un elemento intermedio entre el usuario y el servidor. La nueva capa intermedia de AJAX mejora la respuesta de la aplicación, ya que el usuario nunca se encuentra con una ventana del navegador vacía esperando la respuesta del servidor.
- El siguiente esquema muestra la diferencia más importante entre una aplicación web tradicional y una aplicación web creada con AJAX. La imagen superior muestra la interacción síncrona propia de las aplicaciones web tradicionales. La imagen inferior muestra la comunicación asíncrona de las aplicaciones creadas con AJAX.

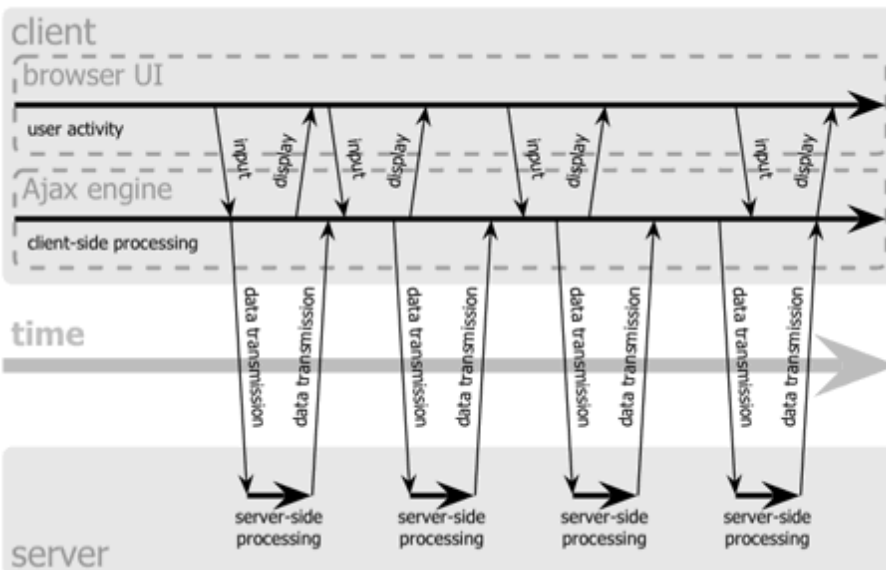
## **Características de AJAX**

# Beneficios de Ajax

classic web application model (synchronous)



Ajax web application model (asynchronous)



- Las peticiones HTTP al servidor se sustituyen por peticiones JavaScript que se realizan al elemento encargado de AJAX.
- Las peticiones más simples no requieren intervención del servidor, por lo que la respuesta es inmediata.
- Si la interacción requiere una respuesta del servidor, la petición se realiza de forma asíncrona mediante AJAX.
- En este caso, la interacción del usuario tampoco se ve interrumpida por recargas de página o largas esperas por la respuesta del servidor.

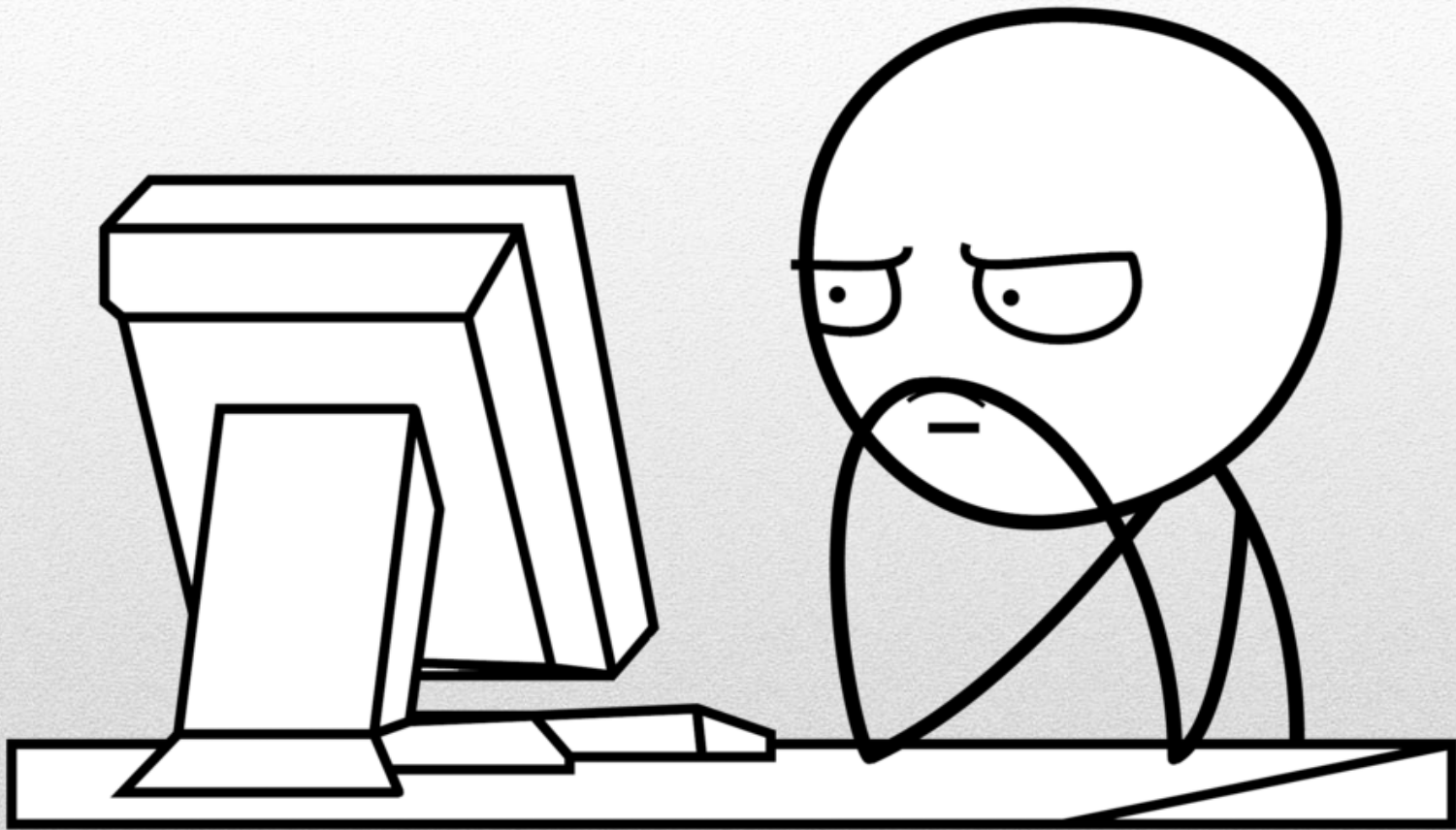


# **Elección del framework**

- Los frameworks de Ajax facilitan y aceleran el desarrollo de las aplicaciones.
- La elección del framework específico depende de tus necesidades y restricciones:
  - Para quienes desean implementar Ajax a través de scripting en el cliente, Prototype y Dojo son muy buenas opciones, que además de la funcionalidad básica de Ajax, proveen capacidades para extender las estructuras base de Javascript y del DOM, con la ventaja agregada de su independencia de la plataforma en el servidor.
  - Para los que desarrollan en ASP.NET, sin lugar a dudas la opción natural es Microsoft ASP.NET AJAX debido a su completa integración con el modelo de controles de servidor de ASP.NET.
  - Los programadores de Java tienen las opciones de DWR y Google Web Toolkit, entre las más conocidas.
- Con esto, he presentado tecnologías representativas (por cierto, todas gratuitas), que a pesar de ser muy distintas entre sí, todas ellas facilitan y aceleran el desarrollo de aplicaciones con Ajax.

## **Características de AJAX**

# EJERCICIO



**AJAX**



# DOM

- Cuando se definió el lenguaje XML, surgió la necesidad de procesar y manipular el contenido de los archivos XML mediante los lenguajes de programación tradicionales. XML es un lenguaje sencillo de escribir pero complejo para procesar y manipular de forma eficiente. Por este motivo, surgieron algunas técnicas entre las que se encuentra DOM.
- DOM o Document Object Model es un conjunto de utilidades específicamente diseñadas para manipular documentos XML. Por extensión, DOM también se puede utilizar para manipular documentos XHTML y HTML.
- Técnicamente, DOM es una API de funciones que se pueden utilizar para manipular las páginas XHTML de forma rápida y eficiente.

# DOM

- Antes de poder utilizar sus funciones, DOM transforma internamente el archivo XML original en una estructura más fácil de manejar formada por una jerarquía de nodos. De esta forma, DOM transforma el código XML en una serie de nodos interconectados en forma de *árbol*.
- El árbol generado no sólo representa los contenidos del archivo original (mediante los nodos del árbol) sino que también representa sus relaciones (mediante las ramas del árbol que conectan los nodos).
- Aunque en ocasiones DOM se asocia con la programación web y con JavaScript, la API de DOM es independiente de cualquier lenguaje de programación. De hecho, DOM está disponible en la mayoría de lenguajes de programación comúnmente empleados.



# DOM

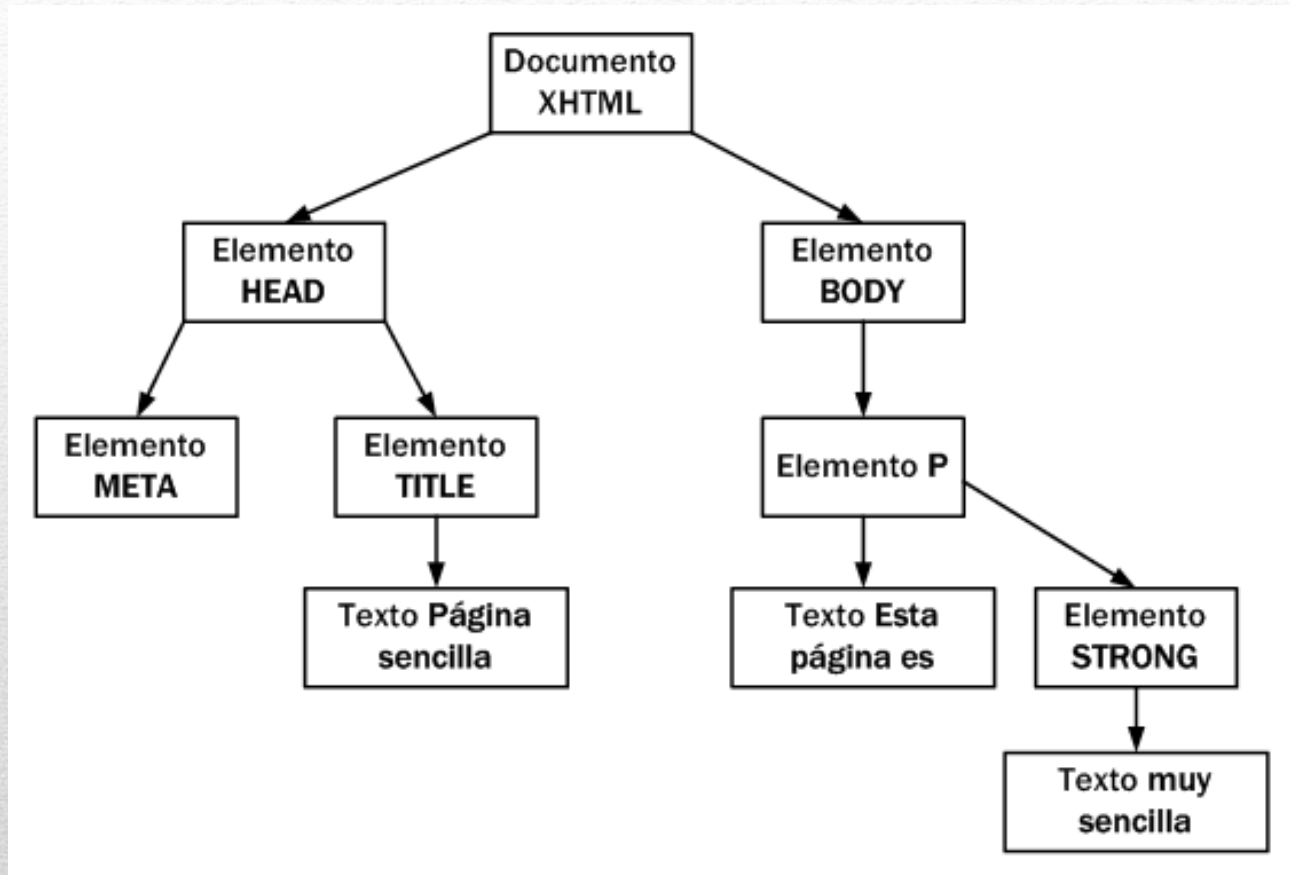
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3
.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" /
  >
    <title>Página sencilla</title>
  </head>

  <body>
    <p>Esta página es <strong>muy sencilla</strong></p>
  </body>
</html>
```

## Elementos básicos de AJAX 21

---

# DOM



## Elementos básicos de AJAX <sup>22</sup>



# DOM

- Antes de poder utilizar la API de DOM, se construye de forma automática el árbol para poder ejecutar de forma eficiente todas esas funciones. De este modo, para utilizar DOM es imprescindible que la página web se haya cargado por completo, ya que de otro modo no existe el árbol de nodos y las funciones DOM no pueden funcionar correctamente.
- La ventaja de emplear DOM es que permite a los programadores disponer de un control muy preciso sobre la estructura del documento HTML o XML que están manipulando. Las funciones que proporciona DOM permiten añadir, eliminar, modificar y reemplazar cualquier nodo de cualquier documento de forma sencilla.
- La primera especificación de DOM (*DOM Level 1*) se definió en 1998 y permitió homogeneizar la implementación del DHTML o *HTML dinámico* en los diferentes navegadores, ya que permitía modificar el contenido de las páginas web sin necesidad de recargar la página entera.

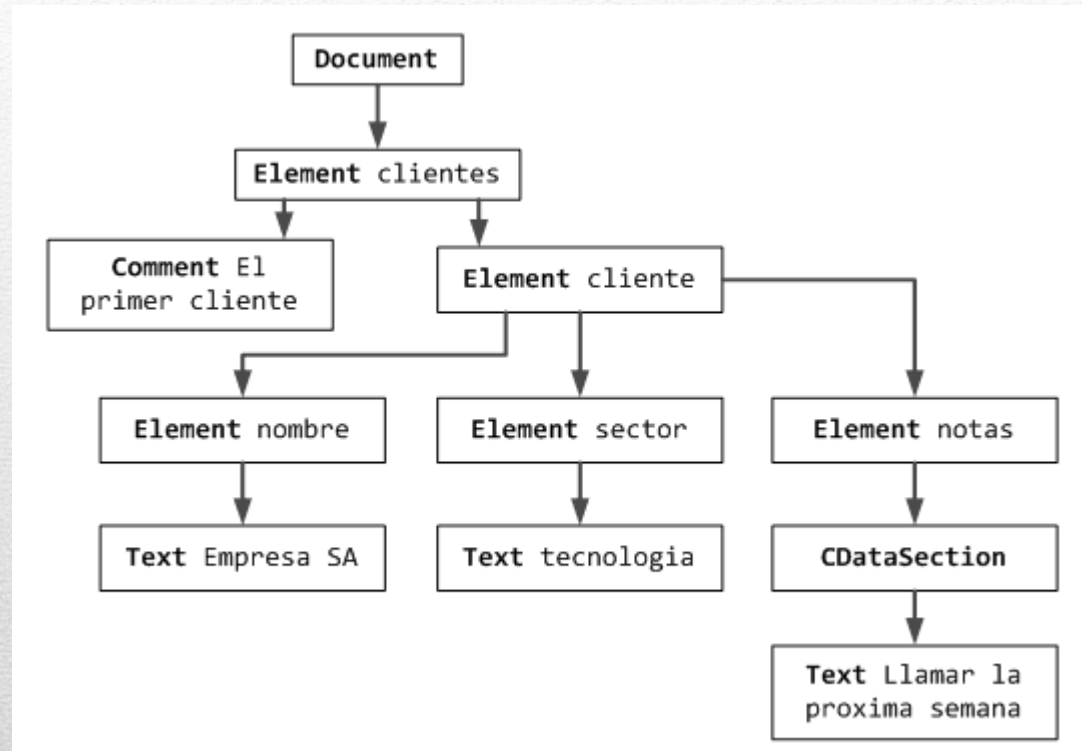
# Selectores y DOM

- Los documentos XML y HTML tratados por DOM se convierten en una jerarquía de nodos. Los nodos que representan los documentos pueden ser de diferentes tipos. A continuación se detallan los tipos más importantes:
  - Document: es el nodo raíz de todos los documentos HTML y XML. Todos los demás nodos derivan de él.
  - DocumentType: es el nodo que contiene la representación del DTD empleado en la página (indicado mediante el DOCTYPE).
  - Element: representa el contenido definido por un par de etiquetas de apertura y cierre (<etiqueta>...</etiqueta>) o de una etiqueta *abreviada* que se abre y se cierra a la vez (<etiqueta/>). Es el único nodo que puede tener tanto nodos hijos como atributos.
  - Attr: representa el par nombre-de-atributo/valor.
  - Text: almacena el contenido del texto que se encuentra entre una etiqueta de apertura y una de cierre. También almacena el contenido de una sección de tipo CDATA.
  - CDATASection: es el nodo que representa una sección de tipo <![CDATA[ ]]>.
  - Comment: representa un comentario de XML.
  - Se han definido otros tipos de nodos pero que no son empleados habitualmente: DocumentFragment, Entity, EntityReference, ProcessingInstruction y Notation.



# Selectores y DOM

```
<?xml version="1.0"?>
<clientes>
  <!-- El primer cliente -->
  <cliente>
    <nombre>Empresa SA</nombre>
    <sector>Tecnologia</sector>
    <notas><![CDATA[
      Llamar la proxima semana
    ]]></notas>
  </cliente>
</clientes>
```



## Elementos básicos de AJAX 25

# Selectores y DOM

- Un buen método para comprobar la transformación que sufren las páginas web y visualizar la jerarquía de nodos creada por DOM es utilizar la utilidad "*Inspector DOM*" (o "*DOM Inspector*") del navegador Mozilla Firefox.
- La utilidad se puede encontrar en el menú *Herramientas* y además de mostrar visualmente la jerarquía de nodos, permite acceder fácilmente a toda la información de cada nodo y muestra en la página web el contenido al que hace referencia el nodo actual.



# Selectores y DOM

- Una vez que DOM ha creado de forma automática el árbol completo de nodos de la página, ya es posible utilizar sus funciones para obtener información sobre los nodos o manipular su contenido.
- JavaScript crea el objeto Node para definir las propiedades y métodos necesarios para procesar y manipular los documentos.
- En primer lugar, el objeto Node define las siguientes constantes para la identificación de los distintos tipos de nodos:

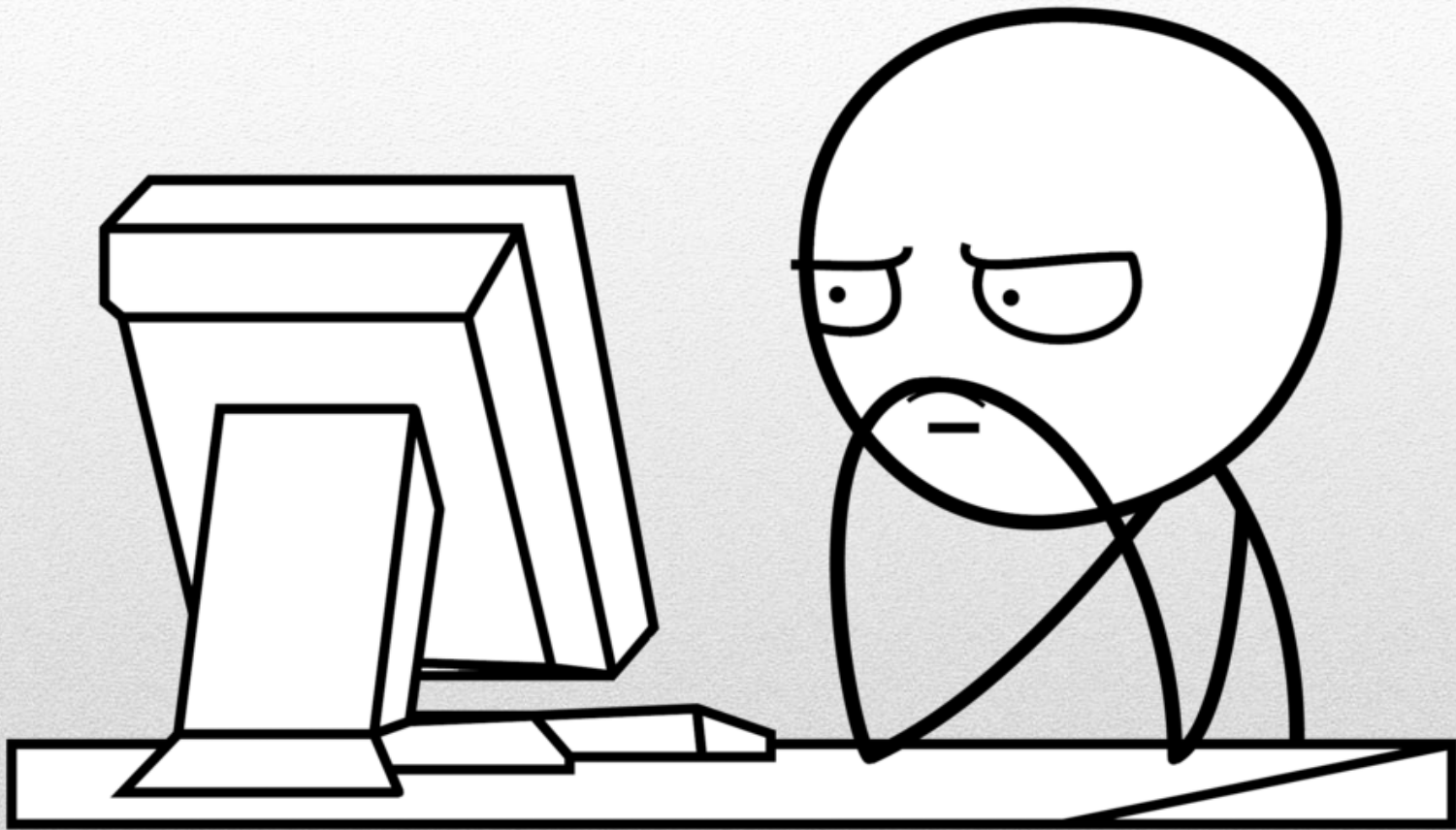
- `Node.ELEMENT_NODE = 1`
- `Node.ATTRIBUTE_NODE = 2`
- `Node.TEXT_NODE = 3`
- `Node.CDATA_SECTION_NODE = 4`
- `Node.ENTITY_REFERENCE_NODE = 5`
- `Node.ENTITY_NODE = 6`
- `Node.PROCESSING_INSTRUCTION_NODE = 7`
- `Node.COMMENT_NODE = 8`
- `Node.DOCUMENT_NODE = 9`
- `Node.DOCUMENT_TYPE_NODE = 10`
- `Node.DOCUMENT_FRAGMENT_NODE = 11`
- `Node.NOTATION_NODE = 12`

Propiedad/Método	Valor devuelto	Descripción
<code>nodeName</code>	<code>String</code>	El nombre del nodo (no está definido para algunos tipos de nodo)
<code>nodeValue</code>	<code>String</code>	El valor del nodo (no está definido para algunos tipos de nodo)
<code>nodeType</code>	<code>Number</code>	Una de las 12 constantes definidas anteriormente
<code>ownerDocument</code>	<code>Document</code>	Referencia del documento al que pertenece el nodo
<code>firstChild</code>	<code>Node</code>	Referencia del primer nodo de la lista <code>childNodes</code>
<code>lastChild</code>	<code>Node</code>	Referencia del último nodo de la lista <code>childNodes</code>
<code>childNodes</code>	<code>NodeList</code>	Lista de todos los nodos hijo del nodo actual
<code>previousSibling</code>	<code>Node</code>	Referencia del nodo hermano anterior o <code>null</code> si este nodo es el primer hermano



<code>hasChildNodes()</code>	Boolean	Devuelve <code>true</code> si el nodo actual tiene uno o más nodos hijo
<code>attributes</code>	NamedNodeMap	Se emplea con nodos de tipo <code>Element</code> . Contiene objetos de tipo <code>Attr</code> que definen todos los atributos del elemento
<code>appendChild(nodo)</code>	Node	Añade un nuevo nodo al final de la lista <code>childNodes</code>
<code>removeChild(nodo)</code>	Node	Elimina un nodo de la lista <code>childNodes</code>
<code>replaceChild(nuevoNodo, anteriorNodo)</code>	Node	Reemplaza el nodo <code>anteriorNodo</code> por el nodo <code>nuevoNodo</code>
<code>insertBefore(nuevoNodo, anteriorNodo)</code>	Node	Inserta el nodo <code>nuevoNodo</code> antes que la posición del nodo <code>anteriorNodo</code> dentro de la lista <code>childNodes</code>

# EJERCICIO



**DOM**



# JSON

- JSON es un acrónimo de *JavaScript Object Notation*, un formato ligero originalmente concebido para el intercambio de datos en Internet. Se considera un subconjunto de la notación literal para representar objetos, Arrays, cadenas, booleanos y números en Javascript.
- Su popularización llegó sobre 2001 gracias al apoyo incondicional de [Douglas Crockford](#). Yahoo! ayudó en gran manera a su difusión a raíz de la inclusión de este formato en algunos de sus servicios web más innovadores. En diciembre de 2006, Google comenzaría a ofrecer sus feeds en JSON para su protocolo web [GData](#).
- Pese a que JSON se basa en la notación Javascript, está considerado como un lenguaje independiente de formato de datos cuya especificación es descrita en [RFC4627](#).

# JSON - Anatomía

```
{
  "id" : "0001",
  "type" : "donut",
  "name" : "Cake",
  "image" : {
    "url" : "images/0001.jpg",
    "width" : 200,
    "height" : 200
  },
  "thumbnail" : {
    "url" : "images/thumbnails/0001.jpg",
    "width" : 32,
    "height" : 32
  },
  "dateEntry" : "2010-12-05"
}
```

- NOTA: Para comprobar la validez de un marcado JSON existen un par de herramientas online de gran valor: [JSONFormatter](#) y [JSONLint](#). En ambos sitios, podemos pegar nuestro código para que sea validado y reformateado.



# JSON

- **Particularidades de JSON sobre Javascript**

- Algunas de las particularidades o reglas del formato JSON a tener en cuenta son:
- Los pares nombre-valor van siempre delimitados por comillas, independientemente de si se tratan de nombres válidos en Javascript que podrían aparecer sin ellas.
- JSON puede representar seis tipos de valores: objetos, Arrays, números, cadenas, booleanos y null.
- Las fechas no son reconocidas como un tipo de objeto propio.
- Los números en JSON no pueden ir precedidos de ceros salvo en el caso de notación decimal ( *Ejem: 0.001* ).
- En definitiva, como JSON es considerado un lenguaje independiente, sus objetos deben ser considerados como cadenas Javascript, no como objetos nativos.

# JSON

- **Usando JSON en Javascript**
- El verdadero potencial de JSON es su integración con Javascript. Esto permite un fácil intercambio de datos entre aplicaciones mediante peticiones XHR al servidor a la vez que abre posibilidades ilimitadas a nuevos paradigmas de gestión como los emergentes sistemas No-SQL.
- Para permitir que un objeto JSON sea utilizado por Javascript, debemos *parsearlo* (interpretarlo) mediante el uso del comando *eval()* tal y como se muestra a continuación:

```
var myCakes = eval('(' + cakeJSON + ')');  
alert(myCakes.name); // donut  
alert(myCakes.image.width); // 200
```



# JSON

- **Usando JSON en Javascript**
- A través de **eval()**, interpretamos el objeto JSON y lo convertimos en una entidad (objeto) Javascript; esto permite acceder a sus propiedades directamente como con cualquier otro objeto.  
Obsérvese que es necesario un doble paréntesis para evitar la ambigüedad con la que Javascript interpreta las llaves con las que comienza el objeto JSON.
- *NOTA: Para acceder a las propiedades del objeto, utilizamos la notación con punto en lugar de corchetes tal y como recomiendan los expertos:*

```
alert( myCakes[thumbnail][height] ); // Peor  
alert( myCakes.thumbnail.height ); // Mejor
```

# JSON

- **Usando JSON en Javascript**
- Las transacciones XHR, por seguridad, están limitadas únicamente al ámbito del dominio que hace la petición, por lo tanto, cuando se recibe una respuesta, podemos estar 100% seguros de que ésta, proviene del propio dominio. Sin embargo, eso no tiene por que librarnos de un error del servidor o de una redirección maliciosa que nuestro eval() a ciegas puede convertir en desastre.
- Siempre es útil recordar ese mantra que inunda foros y artículos y que suele ser lo primero que se graba a fuego en la conciencia de un programador Javascript: *eval is evil*. Para evitar todos los problemas derivados de una incorrecta interpretación de nuestro código, podemos hacer uso de algunas herramientas de terceros.



# JSON

## Objetos Literales

```
var oCliente = {  
  dni : "44035648",  
  nombres : "Maria",  
  apellidos : "Jimenez",  
  edad : 22,  
  activo : true  
};
```

```
alert(oCliente.dni); // Muestra: 44035648  
alert(oCliente.nombres); // Muestra: Maria  
alert(oCliente.apellidos); // Muestra: Jimenez  
alert(oCliente.edad); // Muestra: 22  
alert(oCliente["activo"]); // Muestra: true
```

Almacenan  
información en pares  
nombre : valor

color : "rojo",

Se puede acceder a  
Estas propiedades

Mediante el nombre del  
objeto y la sintaxis de punto.

```
alert(oCliente.activo);
```

Mediante corchetes y nombre  
de la propiedad

```
alert(oCliente["activo"]);
```



# JSON

## Arrays Literales

```
var aNombres = ["Hugo", "Paco", "Luis"];
```

```
alert(aNombres[0]); // Muestra Hugo  
alert(aNombres[1]); // Muestra Paco  
alert(aNombres[2]); // Muestra Luis
```

Formato de datos  
muy ligero

[ y ]

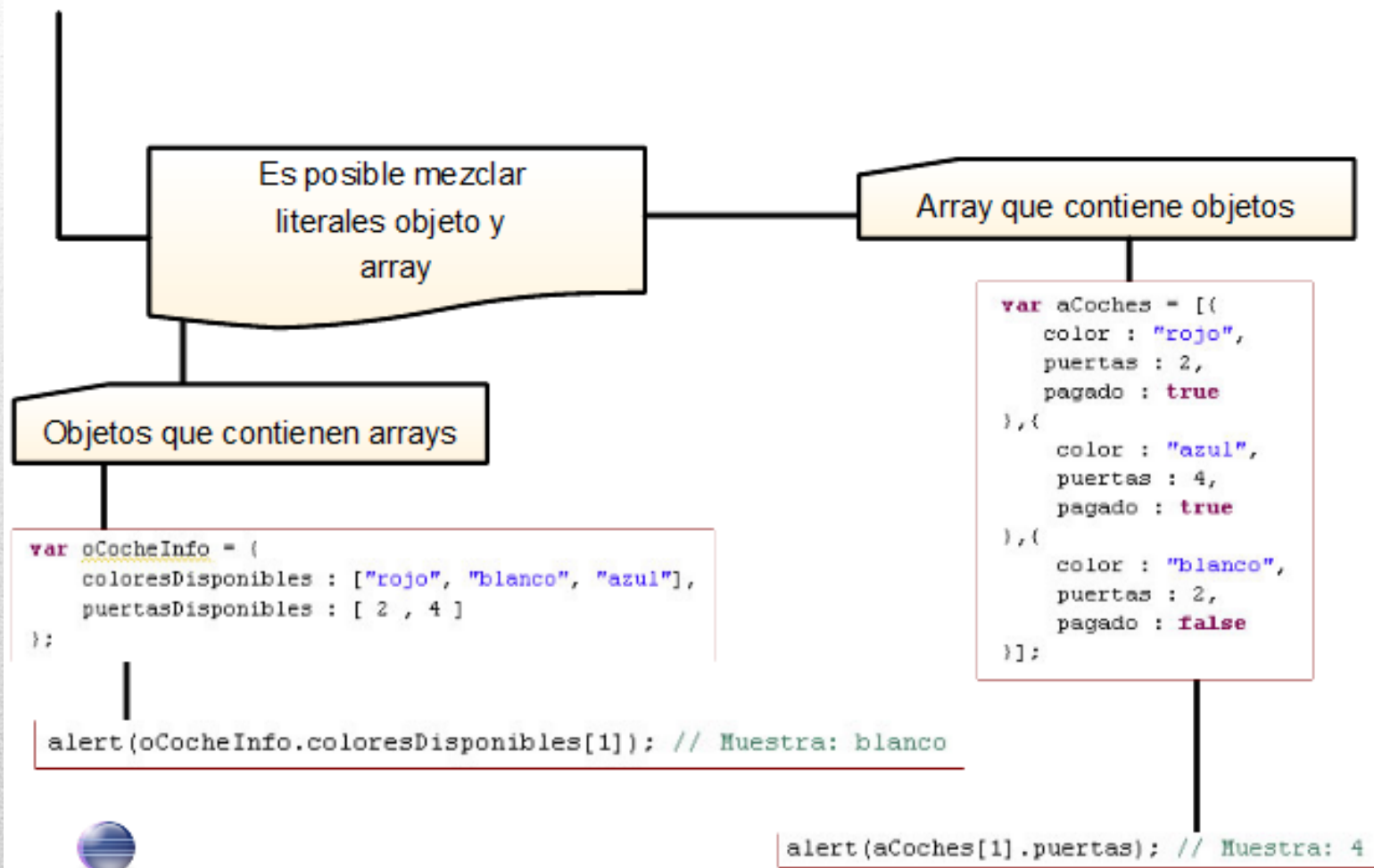
Y por encerrar lista de valores  
separados por comas

[ "string", 24 , true, null ]



# JSON

## Mezclar Literales



# JSON

- **Parseando JSON de forma segura**
- Existen diversas formas que permiten prescindir de eval() delegando la tarea en métodos más seguros.
- En JQuery, tenemos por ejemplo el método parseJSON que comprueba la integridad del marcado antes de evaluarlo. Mootools ofrece también su propio método, JSON.encode(), que realiza una tarea similar al anterior ejemplo de jQuery.
- Sin embargo, con la llegada del ECMAScript 5, se ha implementado un nuevo objeto JSON basado en la API programada por el propio Douglas Crockford. Sus métodos más interesantes son parse() y stringify().



# JSON – parse( )

- **JSON.parse** ofrece un *eval()* seguro a través de su filtrado mediante expresiones regulares. Si la cadena no es un objeto JSON válido, devuelve un error de sintaxis y el eval no es llamado.
- La sintaxis de este comando es la siguiente:  
*JSON.parse( string \$JSONString [, function \$reviver ] );*
- El primer parámetro recoge la cadena JSON mientras que el segundo, opcional, acepta una función para el tratamiento de dicha cadena. Esta función recoge dos parámetros (clave y valor) y por cada registro del objeto Javascript que estamos componiendo, se evalúan sus pares según el criterio definido.

# JSON – parse()

- Este método resulta muy práctico para, por ejemplo, convertir las cadenas de fechas en objetos Javascript:

```
function dateReviver(key, value) {  
  if (typeof value === 'string') {  
    var a = /^(\\d{4})-(\\d{2})-(\\d{2})$/.exec(value);  
    if (a) {  
      return new Date(Date.UTC(+a[1], +a[2] - 1, +a[3]));  
    }  
  }  
  return value;  
};
```

```
var myObj = JSON.parse( myStringJSON, dateReviver);  
myObj.dateEntry; //Sat Dec 04 2010 16:00:00 GMT-0800 (Pacific Standard Time)
```



# JSON – stringify( )

- **JSON.stringify** hace la operación contraria. Su sintaxis es la siguiente:  
*JSON.stringify( obj \$value [, \$replacer] [, @space] );*
- El primer parámetro *\$value* suele ser un objeto Javascript (o un array) y es a partir del cual, obtenemos la cadena JSON correspondiente.
- El segundo parámetro, *\$replacer*, si es una función, actúa básicamente como el *\$reviver* anterior pero de modo inverso. Sin embargo, si le proporcionamos un array, actúa como una lista blanca de aquellas propiedades del objeto que serán serializadas.
- El tercer parámetro, *\$space*, puede ser un número o una cadena. En el primer caso, representa el número de espacios en blanco con el que será indentado cada nivel de nuestro JSON. Si usamos una cadena, cada uno de los pares aparecerá indentado con el carácter que hayamos definido. Por lo general, usaremos los caracteres de escape Javascript para indicar tabulaciones o saltos de línea (el más común será `'\t'` para indicar una tabulación horizontal).

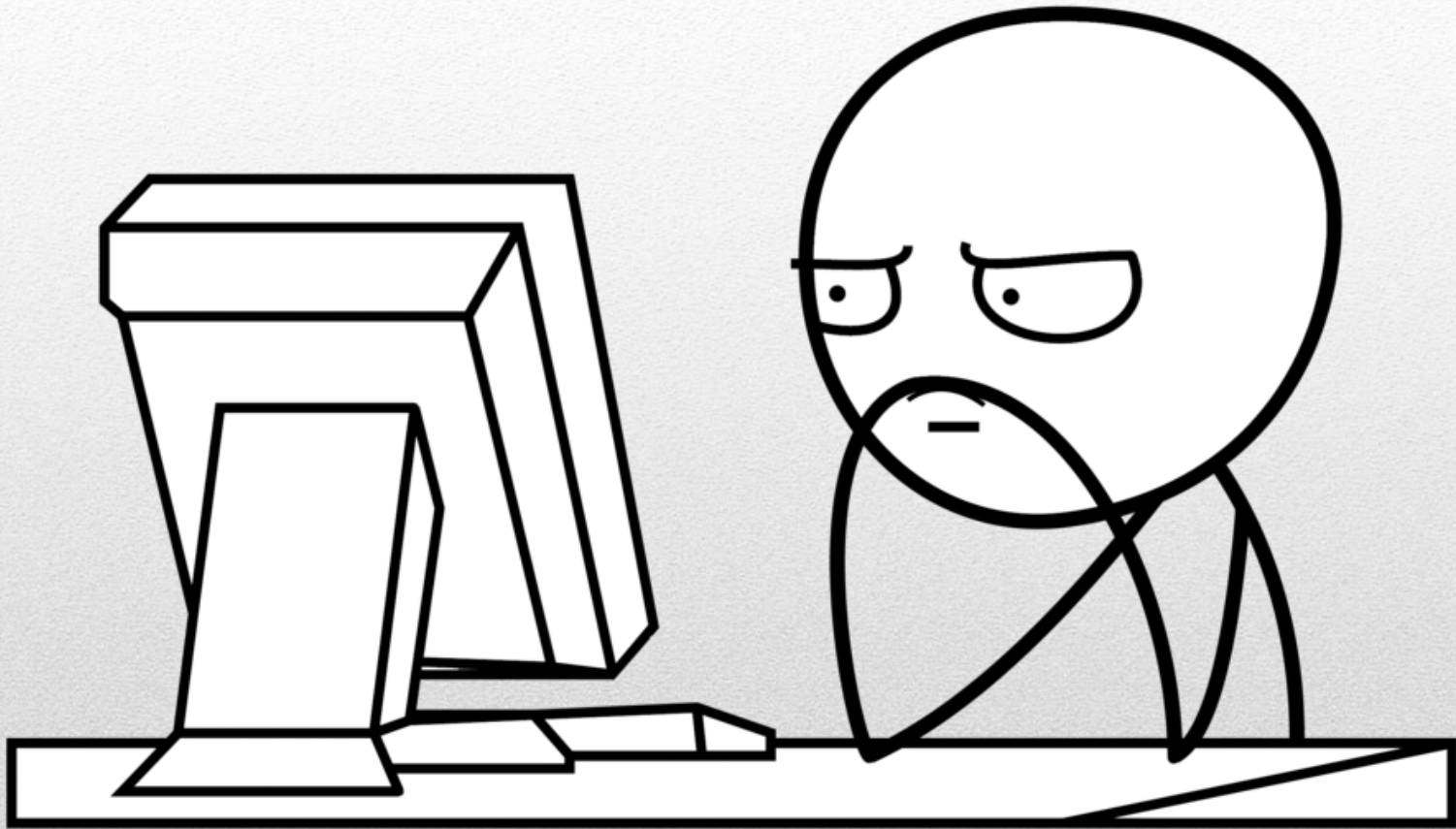
# JSON – stringify( )

- Como podemos ver, el uso de un *\$replacer* resulta interesante para filtrar aquellos pares *clave-valor* que no nos interesan en un momento dado.
- Podemos encontrar ambas funciones en la mayoría de navegadores modernos con la excepción de IE7. [Asen Bozhilov](#) ha compilado una tabla comparativa con las diferentes formas en que los fabricantes han implementada JSON.parse.

```
JSON.stringify(myObj, ['id','type','name'], '\t')  
/*  
{  
  "id" : "0001",  
  "type" : "donut",  
  "name" : "Cake"  
}  
*/
```



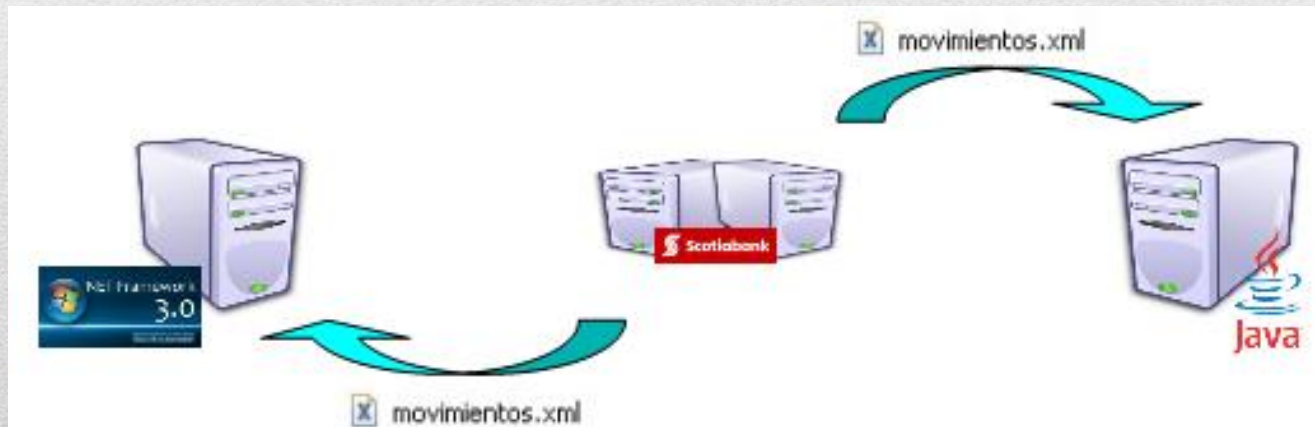
# EJERCICIO



## JSON

# XML

- *Extensible Markup Language* (lenguaje de marcas ampliable)
- XML no ha nacido sólo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas.



## Elementos básicos de AJAX 46



# XML - Estructura

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE movimientos SYSTEM "movimientos.dtd" [<!ELEMENT
movimientos (movimiento)*>]>

<movimientos>
  <movimiento>
    <fecha> 06/09/2008 </fecha>
    <descripcion> Retiro por cajero </descripcion>
    <monto> -100.00 </monto>
  </movimiento>
  <movimiento>
    <fecha> 05/09/2008 </fecha>
    <descripcion> Transferencia de otra cuenta </descripcion>
    <monto> 320.00 </monto>
  </movimiento>
</movimientos>
```

**movimientos.xml**

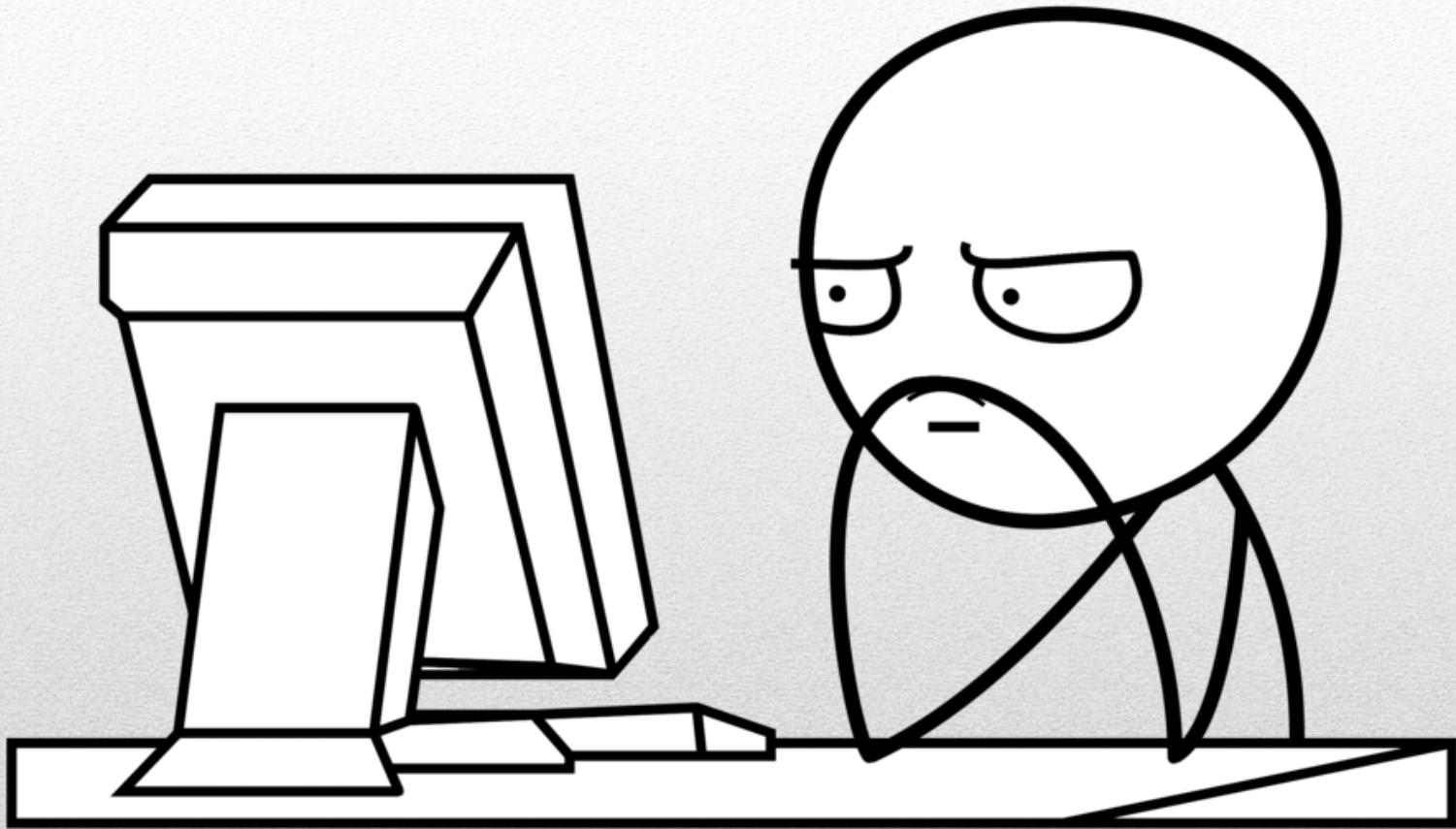
Y el DTD(Document Type Definition) para este XML sería:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Este es el DTD de Edit_Mensaje -->
<!ELEMENT movimiento (fecha, descripcion, monto)*>
  <!ELEMENT fecha (#PCDATA)>
  <!ELEMENT descripcion (#PCDATA)>
  <!ELEMENT monto (#PCDATA)>
```

**movimientos.dtd**

## Elementos básicos de AJAX 47

# EJERCICIO

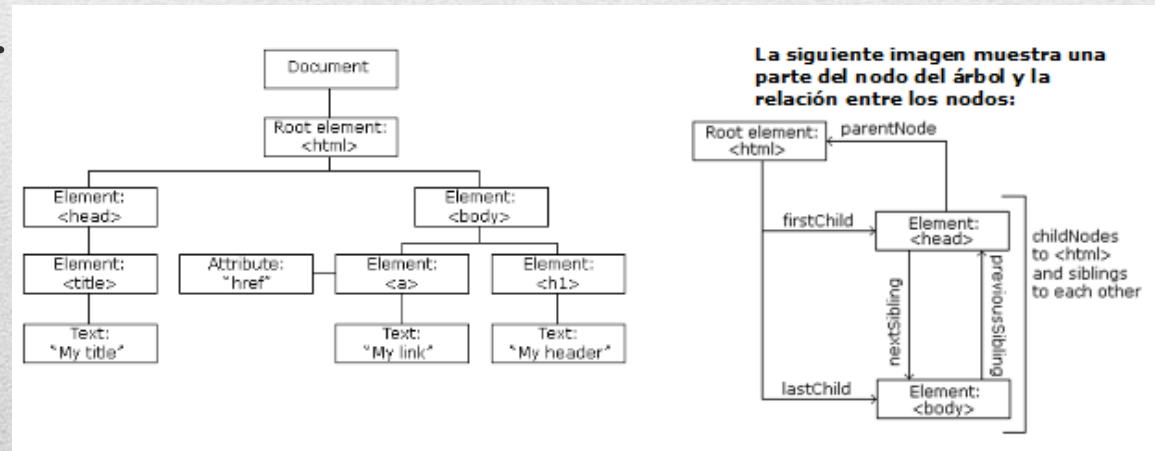


**XML**



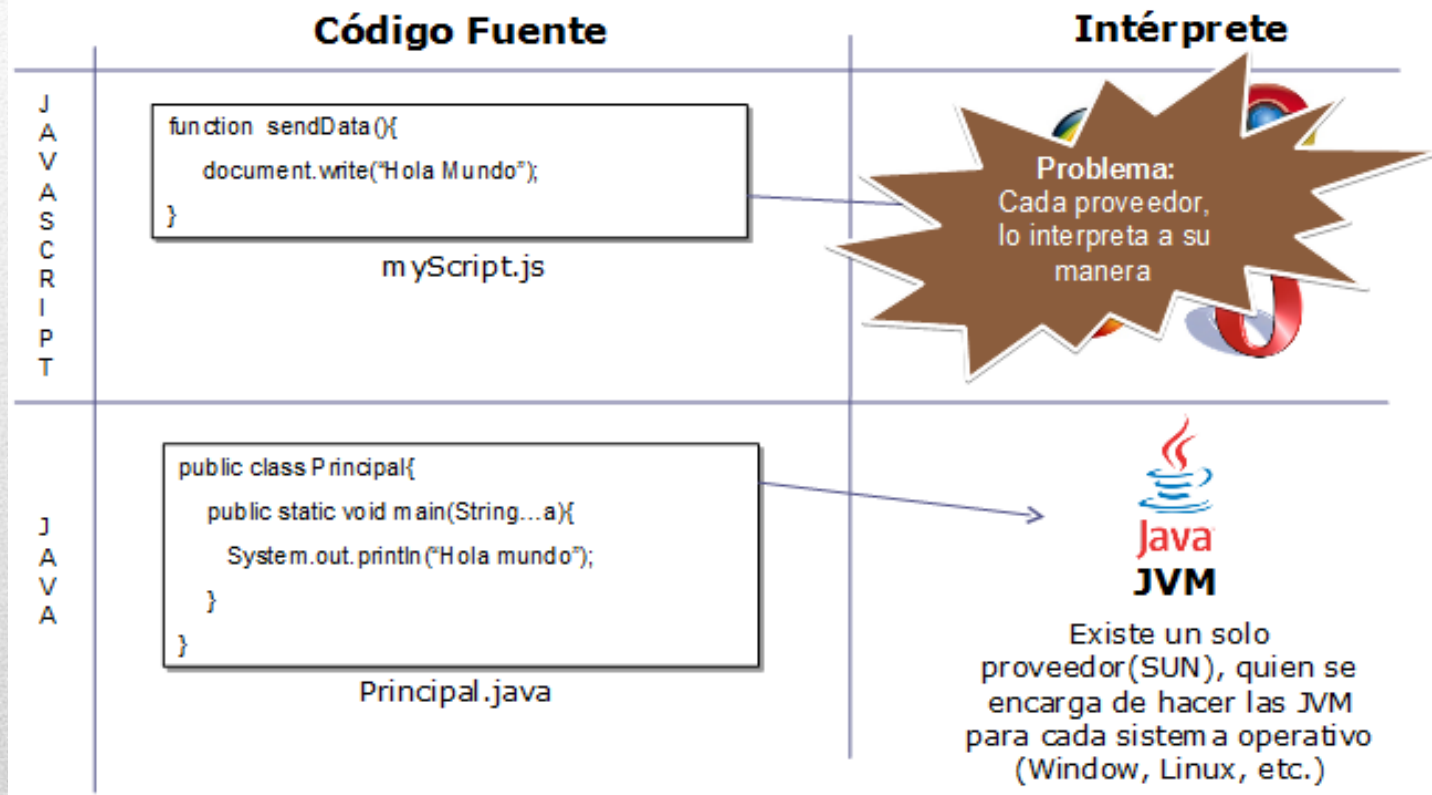
# Javascript y DOM

- El DOM es una plataforma neutral, y el lenguaje que permite a los programas y scripts acceder y actualizar dinámicamente el contenido, la estructura y el estilo de los documentos.
- El documento puede ser procesado y los resultados de la transformación se pueden incorporar de nuevo en la misma página.



# Javascript y limitaciones

El problema de toda la vida...



## AJAX accesible ( hijax )



# Hijax

- Bajo el término de **Hijax**, se encuentra una estrategia del uso del Ajax que tiene como objetivo lograr páginas web accesibles.
- La técnica se basa en lo que se denomina en inglés **progressive enhancement** y **graceful degradation**, dos estrategias que permiten que un sistema informático, en este caso una página web, funcione correctamente aún en el caso de que falte algún componente.
- En el caso de Hijax, la estrategia para lograr el *progressive enhancement* es la siguiente:
  - Diseñar una web al "estilo antiguo", con enlaces y acciones de formularios que envían información al servidor y este devuelve una página completa con cada petición.
  - A continuación utilizar Javascript para capturar los enlaces y las acciones de formularios para enviar la información mediante XMLHttpRequest. De este modo se puede seleccionar que parte de la página se pueda actualizar de forma individual en vez de tener que recargar toda la página.

## AJAX accesible ( hijax )

# Hijax - Ejemplo

- Imaginaros que tenemos el siguiente enlace:

`<a href="/curso/ejemplo/">Mi Ejemplo</a>`

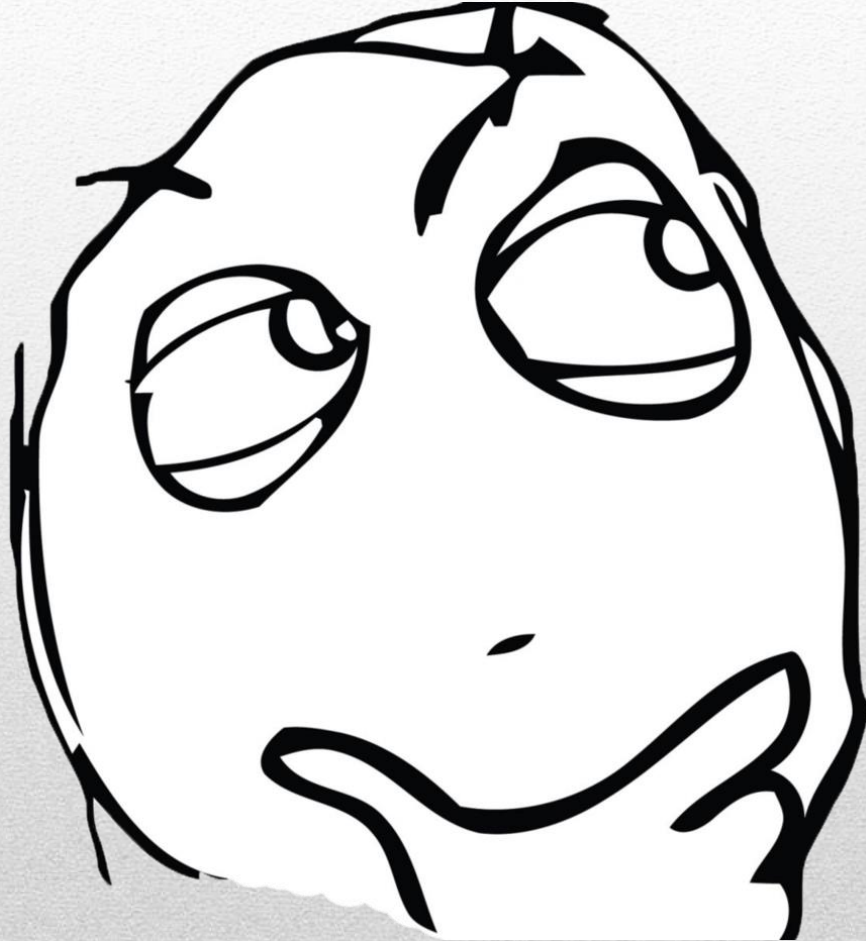
- Y el siguiente código:

```
$(document).ready(function(){
    $('a').click(function() {
        $.ajax({type: 'POST',
            url: 'ajax/' + $(this).attr('href'),
            error: function(xhr, ajaxOptions, thrownError) {
                // Código si error.
            },
            success: function(response) {
                // Código si la operación se ha realizado correctamente.
                return false;
            }
        })
        return false;
    });
});
```

## AJAX accesible ( hijax )



# **TURNO DE DEBATE**



**HIJAX**

# **FIREFOX**

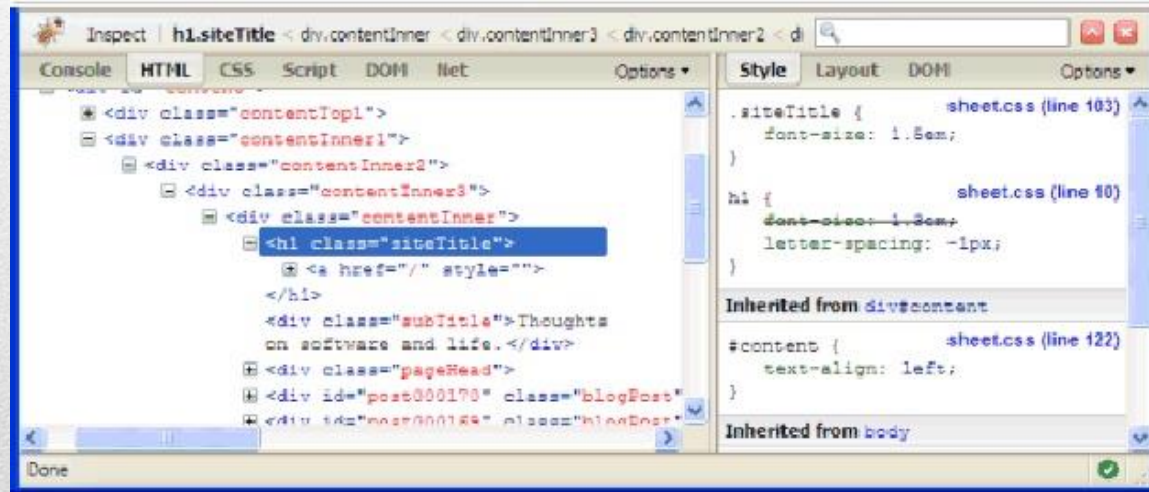
- Si te dedicas profesionalmente al diseño de páginas y aplicaciones web, seguramente tu navegador preferido para trabajar es Firefox. Si no lo conoces, puedes descargar gratuitamente Firefox desde su sitio web oficial:  
<http://www.mozilla.com/firefox/>
- Las principales ventajas de Firefox desde el punto de vista del diseñador y creador de páginas web es que respeta los estándares del W3C mucho más que los navegadores de la familia Internet Explorer. Además, permite instalar pequeños añadidos, llamados extensiones, que añaden funcionalidades al navegador. Una extensión se puede considerar como un pequeño programa que se instala dentro del navegador y que añade alguna característica interesante que el navegador no incorpora de serie. Lo mejor de las extensiones de Firefox es que existen cientos de extensiones, prácticamente todas son gratuitas y casi todas son realmente útiles. El sitio web de Firefox incluye una sección especial llamada “Complementos” en la que se puede encontrar el listado completo de extensiones disponibles para Firefox:  
<http://ad-dons.mozilla.org/es-ES/firefox/>
- A continuación se muestra una pequeña selección de algunas de las extensiones más interesantes para los diseñadores de páginas web.

## **Firebug**



# FIREBUG

- Firebug es probablemente la extensión más útil y completa de todas las que están relacionadas con el diseño web. No importa si tu especialidad es XHTML, CSS, JavaScript, DOM o Ajax, ya que Firebug proporciona toda la información posible sobre cada uno de estos temas. Como Firebug tiene tantas opciones, lo mejor es instalar la extensión y probarla en tus proyectos web.



- Descargar Firebug: <https://addons.mozilla.org/firefox/1843/>

## Firebug

# FIREBUG

- Una vez instalado podemos notar el icono de la extensión en la barra de estado del navegador. Abrimos alguna página web y le damos clic al icono de insecto para iniciarlo.



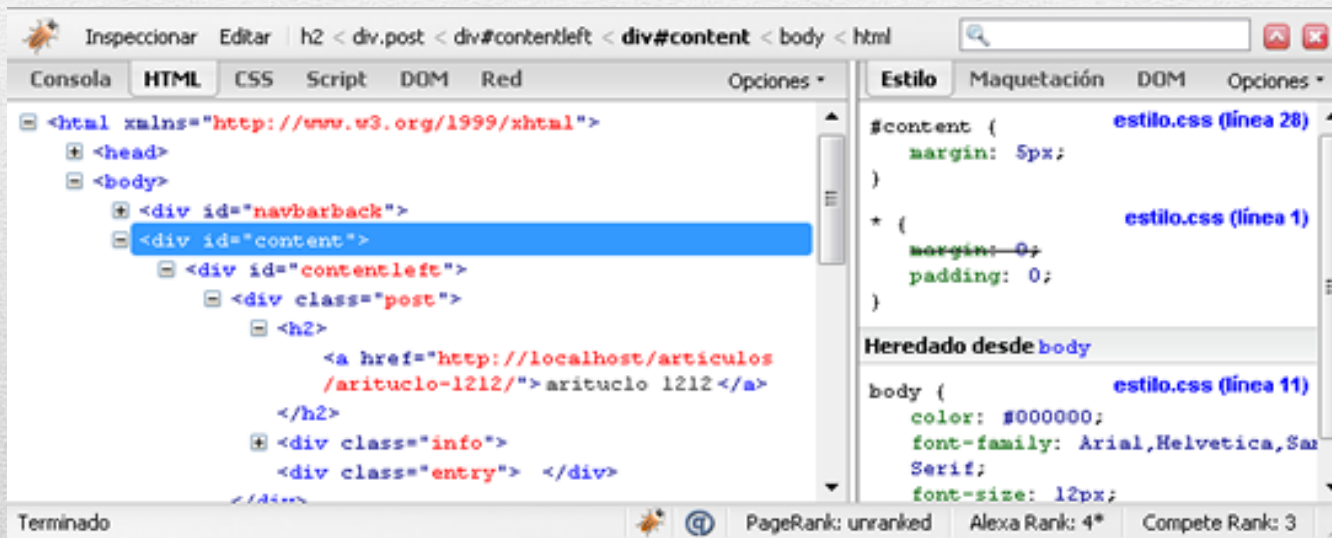
- Por defecto Firebug esta deshabilitado para cada sitio web. Así que habilitamos las 3 opciones: Consola, Script y Red. Recomendamos hacer esto para nuestro sitio local de desarrollo (localhost).
- Luego notaremos la siguiente barra de en el área donde apareció en mensaje anterior.

# Firebug



# Manejo del DOM

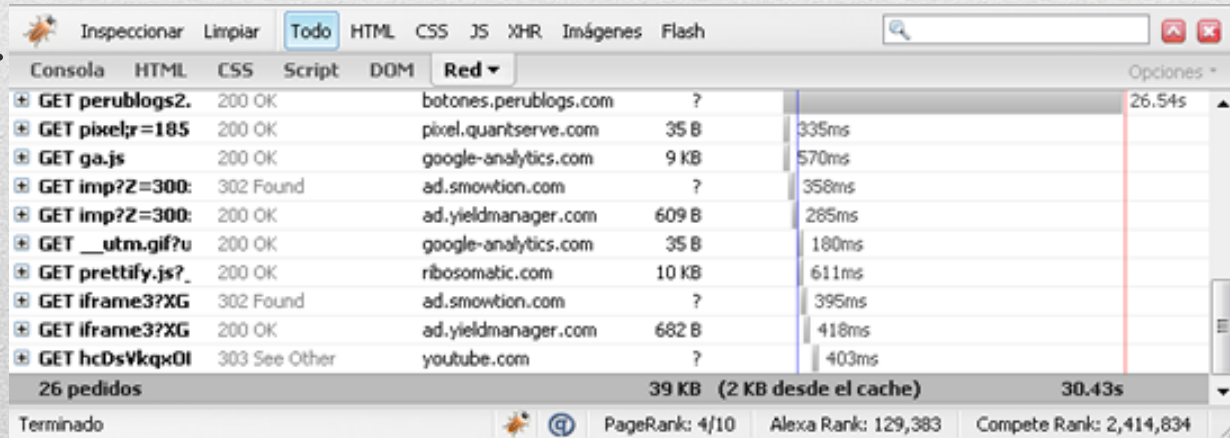
- **Inspeccionar.** Nos permite ver la estructura del documento HTML y navegar a través de este. Podemos ver incluso en el panel derecho los estilos CSS asociados con cada elemento HTML de la página actual, así como los Scripts y el DOM.



# Firebug

# Uso de red

- **Red.** Esta pestaña nos permite monitorear la red. Podemos ver los archivos que relacionados con nuestra página web (css, imágenes, etc.); ver si fueron cargados correctamente o no (muestra código relacionado 202, 301, etc.); la fuente url; el tamaño del archivo y el tiempo de carga.
- Al finalizar la carga de la página web reporta el total de pedidos, el tamaño de toda la página (junto con la que hay en cache) y el tiempo total de carga.



Consola	HTML	CSS	Script	DOM	Red	Opciones
+	GET	perublogs2.	200 OK		botones.perublogs.com	?
+	GET	pixel;r=185	200 OK		pixel.quantserve.com	35 B
+	GET	ga.js	200 OK		google-analytics.com	9 KB
+	GET	imp?Z=300:	302 Found		ad.smowtion.com	?
+	GET	imp?Z=300:	200 OK		ad.yieldmanager.com	609 B
+	GET	__utm.gif?u	200 OK		google-analytics.com	35 B
+	GET	prettify.js?	200 OK		ribomatic.com	10 KB
+	GET	iframe3?XG	302 Found		ad.smowtion.com	?
+	GET	iframe3?XG	200 OK		ad.yieldmanager.com	682 B
+	GET	hcDsVkkxOI	303 See Other		youtube.com	?
26 pedidos					39 KB (2 KB desde el cache)	30.43s
Terminado					PageRank: 4/10	Alexa Rank: 129,383
					Comete Rank: 2,414,834	

Firebug

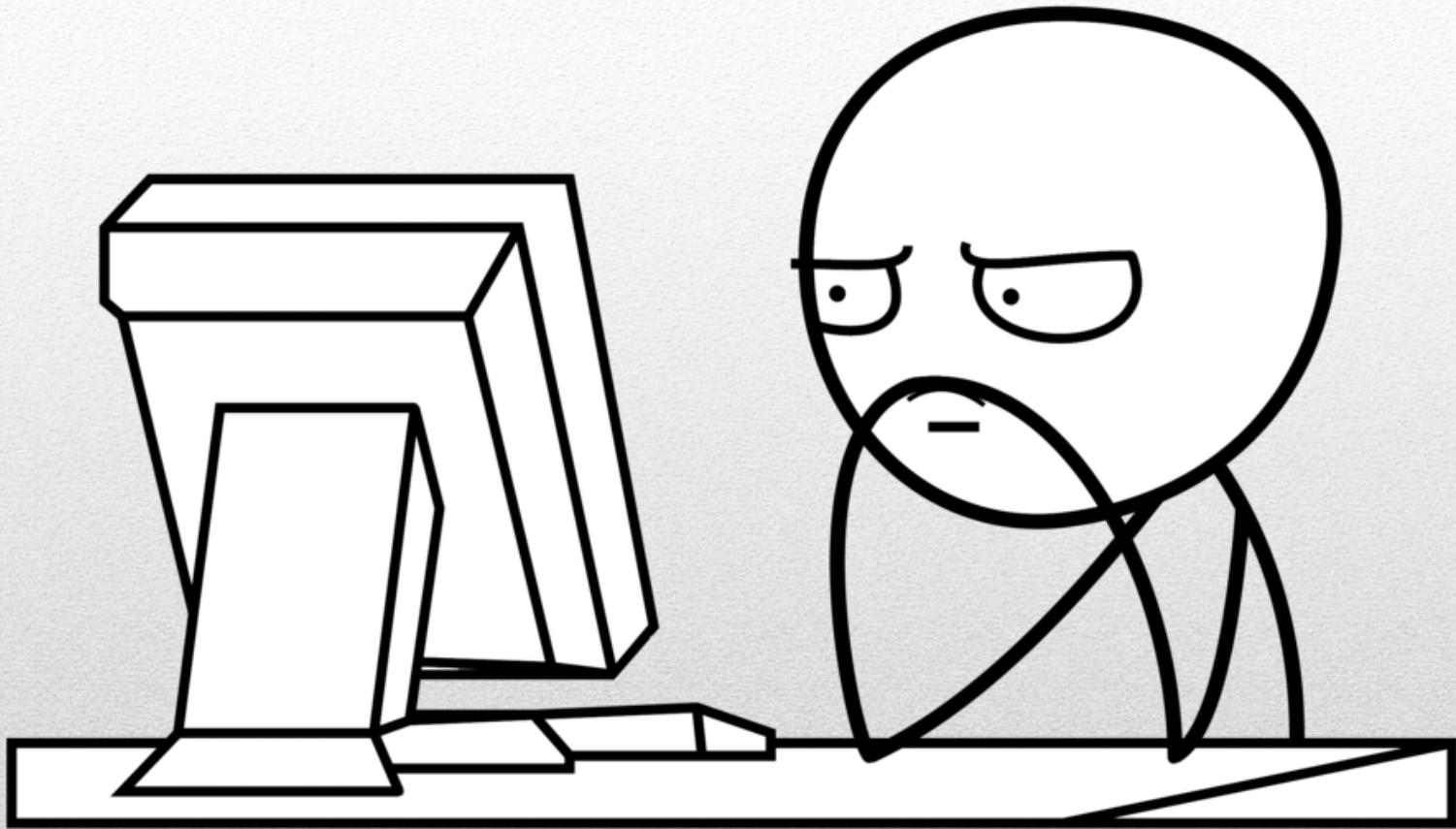


# Manejo de Profiling

- En software, el término **profiling** se refiere a realizar un análisis del rendimiento de un programa, investigar su comportamiento utilizando información reunida en la ejecución del programa. El objetivo último es mejorarlo optimizando aquellas partes que ofrezcan peores resultados.
- El desarrollo en JavaScript no es algo sencillo (sobre todo su *debuggeo*), pero gracias a herramientas como **Firebug** esta tarea resulta bastante más fácil.
- Una de sus múltiples opciones es precisamente el **profiler**:
  - Gracias al cual podremos obtener valiosa información sobre el número de llamadas que se realizan a cada función, el tiempo que consume... y poder así detectar posibles fallos de rendimiento.

## Firebug

# EJERCICIO



## FIREBUG



# JQuery

- JQuery es una librería de JavaScript para acceder a los objetos del DOM de un modo simplificado.
- El autor de esta librería es John Resig que además trabaja para Mozilla Corporation.
- Las aplicaciones en internet son cada vez más complejas, ya que incorporan efectos visuales, drag and drop, auto-completar, animaciones etc. el desarrollar todos estos conceptos desde cero puede resultar complicado sobretodo si tenemos que presentar la solución con muy poco tiempo, en este tipo de situaciones el empleo de librerías como el JQuery nos facilitan el desarrollo de la aplicación. Otra ventaja paralela es despreocuparnos cuando codificamos en la compatibilidad de navegadores, ya que la librería resolverá esto.

## Ajax y frameworks

---

# JQuery

- Para utilizar la librería como dijimos debemos descargarla del sitio oficial y en cada página que lo requiera agregar:

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

- Del sitio oficial de JQuery descargaremos la versión descomprimida que ocupa alrededor de 60 Kb (es el archivo jquery.js) y cuando subamos nuestro sitio es recomendable descargar la versión comprimida que tiene un peso de 20 Kb.



# JQuery

- La librería JQuery en resumen nos aporta las siguientes ventajas:
  - Nos ahorra muchas líneas de código.
  - Nos hace transparente el soporte de nuestra aplicación para los navegadores principales.
  - Nos provee de un mecanismo para la captura de eventos.
  - Provee un conjunto de funciones para animar el contenido de la página en forma muy sencilla.
  - Integra funcionalidades para trabajar con AJAX.

# JQuery

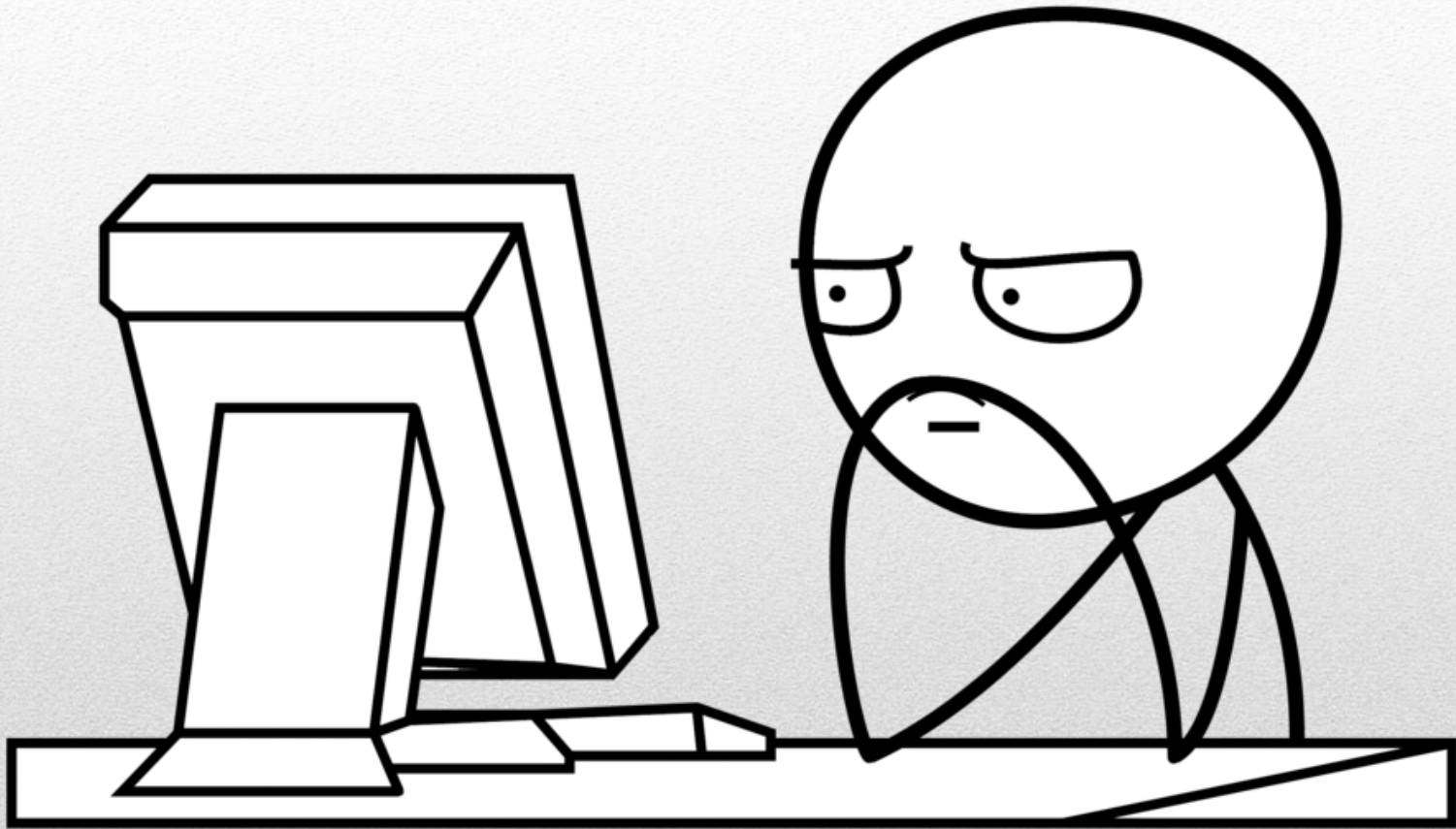
- Selecciones:
  - ID
  - Tipo de elementos
  - Clase
- text() / text(valor) / val() / val(valor)
- attr(nombre\_prop) / attr(nombre\_prop, valor) / removeAttr(nombre)
- addClass(nombre\_clase) / removeClass(nombre\_clase)
- html() / html(código\_html)
- Eventos (click, dblclick, blur,...)
- Manipulación de elementos del DOM / Iteración de elementos
- show() / hide() / fadeIn() / fadeOut() /  
fadeOut(velocidad,valor\_opacidad) /  
fadeTo(velocidad,valor\_opacidad,función) / toggle()

## Ajax y frameworks

---



# EJERCICIO



## JQUERY

# Ext-JS

- **Ext JS** (pronunciado como "ekst" ) es una biblioteca de JavaScript para el desarrollo de aplicaciones web interactivas usando tecnologías como [AJAX](#), [DHTML](#) y [DOM](#). Fue desarrollada por [Sencha](#).
- Originalmente construida como una extensión de la biblioteca [YUI](#) por Jack Slocum, en la actualidad puede usarse como extensión para las biblioteca [JQuery](#) y [Prototype](#). Desde la versión 1.1 puede ejecutarse como una aplicación independiente.



# Ext-JS - Versiones

---

- **Ext JS 1.1.** Liberación final: 1 de agosto de 2007. Puede ejecutarse como una aplicación independiente (stand-alone).
- **Ext JS 2.0.** Liberada el 4 de diciembre de 2007. Su atractivo radicaba en ofrecer interfaces y elementos similares a las aplicaciones de escritorio. También incluía documentación de su API y ejemplos. No tiene compatibilidad con su versión anterior, la 1.X por lo que era forzoso efectuar un plan de migración.
- **Ext JS 2.0.1.** Liberación final: 23 de enero de 2008. Principalmente arreglaba varios errores detectados en la versión 2.0.
- **Ext JS 2.1.** Liberación final: 21 de abril de 2008. Soporte para REST.
- **Ext JS 2.2.** Liberación final: 4 de agosto de 2008.
- **Ext JS 3.0.** Liberación final: 10 de agosto de 2009. Mejoras en consistencia y manejo de memoria. Es la primera versión que aparece con el nombre Ext JS.
- **Ext JS 3.1.** Liberación final: 17 de diciembre de 2009. Mejoras en desempeño en Internet Explorer. Optimización de Layouts. Nuevos componentes como el TreeGrid.
- **Ext JS 3.2.** Liberación final: 7 de abril de 2010. Nuevos elementos como SliderTip, SliderField. Mejorar de desempeño en Box Layouts, AnchorLayout y ColumnLayout. Ordenamiento y filtrado múltiple en elementos Store. Transiciones animadas para elementos DataView.
- **Ext JS 3.3.** Liberación final: 11 de octubre de 2010. Agregó los elementos PivotGrid, ActionColumn y nuevos componentes para el manejo de Calendarios.
- **Ext JS 4.0.** Liberación final: 26 de abril de 2011. Incluye una refactorización de todo el framework entre lo que cabe destacar una nueva estructura de clases y carga dinámica de objetos, paquete de datos, nuevos gráficos y temas.
- Todas las versiones suelen tener compatibilidad con la versión anterior a excepción de la 1.X.

## Ajax y frameworks

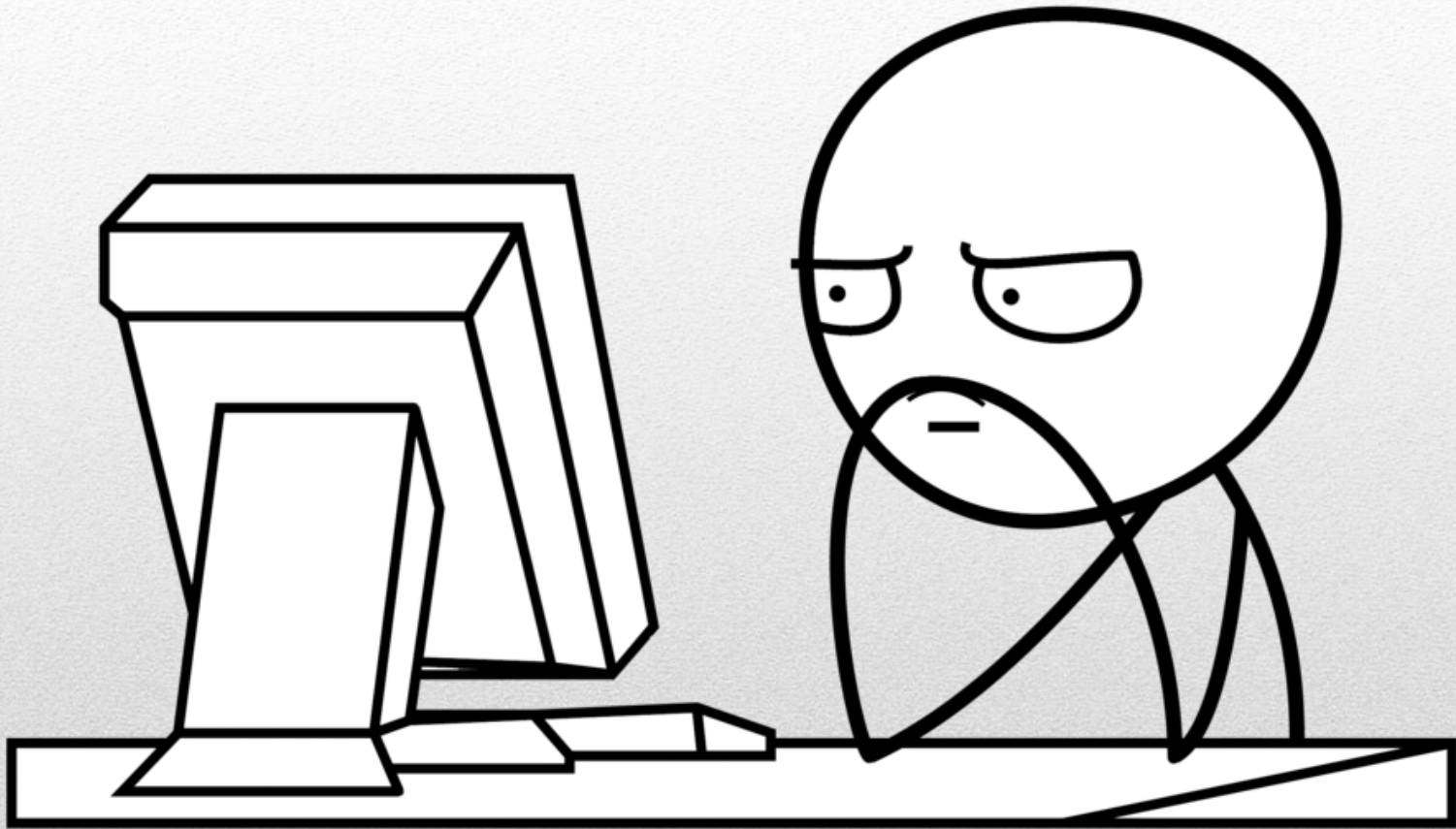
---

# Ext-JS - Funcionalidades

- Dispone de un conjunto de componentes (widgets) para incluir dentro de una aplicación web, como:
  - Cuadros y áreas de texto.
  - Campos para fechas.
  - Campos numéricos.
  - Combos.
  - Radiobuttons y checkboxes.
  - Editor HTML.
  - Elementos de datos (con modos de sólo lectura, datos ordenables, columnas que se pueden bloquear y arrastrar, etc.).
    - Árbol de datos.
    - Pestañas.
- Barra de herramientas.
  - Menús al estilo de Windows.
  - Paneles divisibles en secciones.
  - Sliders.
  - Gráficos
- Varios de estos componentes están capacitados para comunicarse con el servidor usando AJAX. También contiene numerosas funcionalidades que permiten añadir interactividad a las páginas HTML, como:
  - Cuadros de diálogo.
  - quicktips para mostrar mensajes de validación e información sobre campos individuales.



# EJERCICIO



**EXT-JS**



# **DUDAS Y PREGUNTAS**

---



# BIBLIOGRAFIA



Ajax, rich internet applications y desarrollo web para programadores

autor: de DEITEL, HARVEY M.

**AJAX**  
LOS MEJORES TRUCOS



Ajax. los mejores trucos

autor: de PERRY, BRUCE W.



Introduccion a ajax con php (programacion)

autor: de BABIN, LEE



Asp.net con ajax

autor: de VAVILALA, RAMA KRISHNA y GALLO, ALESSANDRO



Ajax con asp.net (wrox)

autor: de VV.AA.



Ajax (manual imprescindible)

autor: de MELLADO DOMINGUEZ, JAVIER



Ajax en j2ee

autor: de MARTIN SIERRA, ANTONIO



Ajax

autor: de VV.AA. y FAWCETT, JOE y MCPEAK, JEREMY

# **BIBLIOGRAFIA**

- **Libros**

- Christian Gross - Ajax Patterns and Best Practices - Ed. Apress, 2006
- Ryan Asleson y Nathaniel T. Schutta - Foundations of Ajax - Ed. Apress, 2006

- **Pequeños o grandes tutoriales y donde buscarlos**

- Sang Shin 10-Week Free AJAX Programming
- <http://www.javapassion.com/ajaxcodecamp/>
- Sergio Gálvez JSP(Java Server Pages) [www.sicuma.uma.es](http://www.sicuma.uma.es)
- Ángel Barbero Tutorial de XML [www.lawebdelprogramador.com](http://www.lawebdelprogramador.com)
- Lola Cárdenas Curso de JavaScript [www.lawebdelprogramador.com](http://www.lawebdelprogramador.com)
- José C. García Curso de Hojas de Estilo [www.lawebdelprogramador.com](http://www.lawebdelprogramador.com)

- **Webs**

- W3C [www.w3schools.com](http://www.w3schools.com)
- W3 <http://www.w3.org/>
- Web estilo <http://www.webestilo.com/html/>





**fsgilp@gmail.com**

---