AJAX

5.- AJAX (Request.js)

Este nombre cambia respecto a la versión anterior de "Ajax" pasa a llamarse "AJAX". Algunas propiedades / métodos cambian y desaparecen.

ACENTOS:
Siempre que trabajemos con AJAX nos vamos a encontrar tarde o temprano con el problema de la codificación de la página. Nosotros trabajos en "ISO-8895-1", mientras que AJAX toma por defecto "UTF-8" (adios acentos).

Podríamos pensar que poniendo estas cabeceras en el ASP / PHP al que llamamos mediante AJAX tenemos el problema solucionado. Pero no es así.

ASP

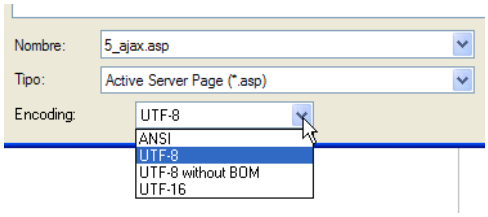
```
<%Response.AddHeader  "charset", "ISO-8859-1"%>
```

PHP

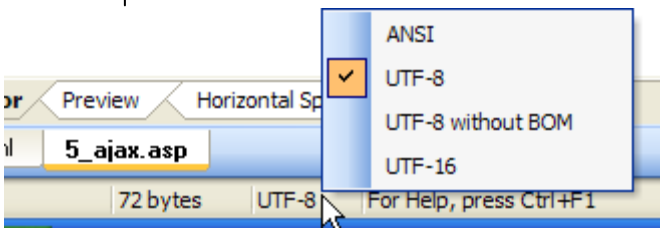
```
<?header("charset: ISO-8859-1")?>
```

En teoria Request lleva una propiedad llamada encoding. Pero tampoco funciona.

Solución: Se debe cambiar la codificación del asp de ANSI a UTF-8 al guardar. Desde WebBuilder.



O desde la barra de estado.



Creando un objeto Ajax

```
var miAjax = new Request({url: 'miurl', opciones : opciones);
```

url: La url del script que recibe la petición.

Opciones: Data

Es la información que le vamos a pasar a nuestro script. Puede ser un string (texto), un formulario o un objeto.

Pasando un String:

```
var miAjax = new Request({
      url: '5_ajax2.asp',
      method: 'get',
      data:'ejemplo=uno&variable=mootools',
      onComplete: function(a){ escribe(a)}
});
miAjax.get();
```

Pasando un formulario:

```
var miAjax = new Request({
      url: '5_ajax2.asp',
      method: 'get',
      data:$( 'form'),
      onComplete: function(a){ escribe(a)}
});
miAjax.get();
```

Donde $('form') es algo como:

```
<form id='formu' name='formu' action='5_ajax2.asp'>
    <input type="hidden" id="kk" name="kk" value="Hola" />
</form>
```

Pasando un Objeto:

```
var miAjax = new Request({
      url: '5_ajax2.asp',
      method: 'get',
      data:  {ejemplo: 'tres', variable: 'mootools' },
      onComplete: function(a){ escribe(a)}
});
miAjax.get();
```

Es una instancia empotrada de Request.

evalScripts

true o false. Si se indica true se evaluarán todos los scripts que vengan en la respuesta de nuestra petición. Se ejecutará todo lo que este entre etiquetas <script>. El valor predefinido es false.

```
var ajax = new Request({
      url: "5_b_ajax.html",
      evalScripts: true,
      onComplete: function(a){ escribe(a); }
}).get();
//Aparecerá un “hola”
```

recibe.html

```
<html>
<head>
<script>
    alert("hola")
</script>
</head>
<body></body>
</html>
```

En recibe.asp podría haber cualquier cosa, incluyendo cualquier etiqueta HTML. Lo único que se evalúa es lo que está entre las etiquetas < script>. Nota: Al parecer no funciona bien ni en ie6, ni en ie7.

evalResponse

true o false. Si se indica true se evalúa toda la respuesta como si fuera un script javascript. El valor predefinido es false.

```
var ajax = new Request({
      url: "5_c_ajax.html",
      evalResponse: true
}).get();

//Aparecerá un “hola”
```

En recibe.html solo puede haber código javascript

```
alert(“hola”)
```

.

Eventos

onComplete

Es la función que se ejecuta cuando llega la respuesta de nuestra petición AJAX.

onSuccess(responseText, responseXML)

Se ejecuta cuando llega la respuesta. Distingue dos tipos de respuesta: de texto y de XML.

onRequest

Se ejecuta cuando se hace el request.

onCancel
Se ejecuta cuando se cancela el request.

onFailure
Se ejecuta cuando el objeto XMLHttpRequest devuelve un error y no se puede completar la petición.

Métodos

getHeader(header)
Devuelve el tipo de cabecera solicitada o null si no existe.

```
var ajax = new Request({
    url: "5_d_ajax.html",
    onComplete: function(a){escribe( this.getHeader('Date'));}
}).get();
```

El código de recibe.asp es:

```
Tue, 04 Nov 2008 17:38:15 GMT
```

y podemos preguntar por (sacado del webDeveloper de Firefox):

Response Headers

Server	Microsoft-IIS/5.1
Date	Wed, 16 Apr 2008 19:34:56 GMT
X-Powered-By	ASP.NET
Content-Length	4
Content-Type	text/html
Cache-Control	private

cancel
Cancela el request que este corriendo en ese momento.

```
Ajax.cancel()
```

Extras

Elemento_formulario.set('send',{opciones}).send()

send es un pequeño gran atajo. Nos permite enviar un formulario con AJAX de manera muy sencilla. La url de destino la toma del action del form al igual que las variables. El resto se puede especificar en las opciones que son las mismas que explique mas arriba.

```
$("formu").set('send', {onComplete: function(a) { escribe(a)}}).send();
```

Donde formu corresponde a:

```
<form id='formu' name='formu' action='5_ajax.asp'>
    <input type="hidden" id="kk" name="kk" value="Hola" />
</form>
```

request.HTML

Una práctica común con AJAX es actualizar el contenido DOM de un elemento con el resultado de lo que nos bajamos en el request. Es decir, substitutir el contenido html (innerhtml) de un elemento de nuestro html.

Con esta extensión Mootools lo hace de forma automática. Tiene las mismas opciones que request y además:

Update: 'id'
Id será el id del elemento que contendrá el nuevo html.

evalScripts: booleano
Se evaluará aquello que encuentre entre <script> (por defecto falso).

evalResponse: booleano
Se evaluará todo lo que obtenga como respueta (por defecto falso).

Ejercicios:

1.- Busca una ley sea bastante larga.

2.- Crea al menos 5 páginas html que cada una tenga parte de la información. No nos vamos a preocupar de los estilos. Llámalas p1.html, p2.html, p3.html, etc.

3.- Crea una página principal, que al arrancar nos muestre tantos marcos pequeños como páginas creaste. Cada marquito albergará un número correspondiente a cada una de las páginas con un enlace.

4.- Al arrancar mostrará en un DIV la información correspondiente a la página p1.html pues entendemos que quiere empezar a leer por esta página.

5.- Cuando acabe de cargar p1.html comenzará la carga de p2.html (la consecutiva) en una capa no visible. Cuando la carga de p2.html esté completa deberá de cambiar el color del número .

6.- Si el usuario pincha:

- sobre un número cuya página asociada esté ya cargada (por ejemplo la 2), la carga será inmediata. "Encendemos" la capa con la página ya cargada. Si la página siguiente no estuviera cargada, se cargará (la página 3) y se cambiará de color el marquito correspondiente.
- Sobre un número cuya página está sin cargar (por ejemplo la 4), se cargará por ajax esa página y se mostrará. Cuando se acabe la carga se empezará a cargar por ajax la página siguiente (la 5) en una capa no visible a la espera del que el usuario pinche en el marquito correspondiente.

Ten presente que tras la última página viene la primera.