

MANUAL

De Ajax

Ajax Ya

Extraído de <http://www.ajaxya.com.ar/>

<i>Orden del concept o</i>	<i>Concepto</i>	<i>Detalle del concepto</i>	<i>Pagina</i>
---	------------------------	--	----------------------

1	Qué es AJAX?		
2	Ventajas y desventajas de AJAX.		
3	Un ejemplo con AJAX.		
4	El mismo ejemplo sin AJAX.		
5	Objeto XMLHttpRequest		
6	Pasando datos al servidor por el método GET.		
7	Pasando datos al servidor por el método POST.		
8	Recuperando datos mediante la propiedad.responseText del objeto XMLHttpRequest		
9	Recuperando datos mediante la propiedad.responseXML del objeto XMLHttpRequest		
10	Propiedades.onreadystatechange y readyState.		
11	Propiedades status y.statusText.		
12	Método abort del objeto XMLHttpRequest		
13	Mostrar un gif animado mientras se envían y reciben los datos del servidor.		
14	Paginación con AJAX		
15	Cargar un control de tipo select		
16	Mostrar un tooltip con datos recuperados del servidor en forma asincrónica		
17	Autocompletar un control de tipo text		
18	Encuesta con AJAX		
19	Qué es JSON?		
20	Recuperar datos del servidor en formato JSON		
21	De PHP a JSON (utilizando la librería JSON.php)		

22	<i>De JavaScript a JSON (utilizando la librería json.js)</i>		
23	<i>De JSON a JavaScript (utilizando la librería json.js)</i>		
24	<i>De JSON a PHP (utilizando la librería JSON.php)</i>		
25	<i>Pizarra interactiva multiusuario</i>		

1 - Qué es AJAX?

AJAX son las siglas de Asynchronous JavaScript And XML. No es un lenguaje de programación sino un conjunto de tecnologías (HTML-JavaScript-CSS-DHTML-PHP/ASP.NET/JSP-XML) que nos permiten hacer páginas de internet más interactivas.

La característica fundamental de AJAX es permitir actualizar parte de una página con información que se encuentra en el servidor sin tener que refrescar completamente la página. De modo similar podemos enviar información al servidor.

La complejidad se encuentra en que debemos dominar varias tecnologías:

- HTML o XHTML
- CSS
- JavaScript
- DHTML Básicamente debemos dominar todos los objetos que proporciona el DOM.
- XML Para el envío y recepción de los datos entre el cliente y el servidor.
- PHP o algún otro lenguaje que se ejecute en el servidor (ASP.Net/JSP)

En este curso suponemos que domina las tecnologías mencionadas, en caso de no ser así recomiendo recorrer los cursos de [HTML Ya](#), [JavaScript Ya](#), [CSS Ya](#), [PHP Ya](#) y [DHTML Ya](#), luego todo lo nuevo que aparezca lo haremos explicando lentamente a través de ejemplos.

2 - Ventajas y desventajas de AJAX.

Ventajas

1. Utiliza tecnologías ya existentes.
2. Soportada por la mayoría de los navegadores modernos.
3. Interactividad. El usuario no tiene que esperar hasta que lleguen los datos del servidor.
4. Portabilidad (no requiere plug-in como Flash y Applet de Java)
5. Mayor velocidad, esto debido que no hay que retornar toda la página nuevamente.
6. La página se asemeja a una aplicación de escritorio.

Desventajas

1. Se pierde el concepto de volver a la página anterior.
2. Si se guarda en favoritos no necesariamente al visitar nuevamente el sitio se ubique donde nos encontrábamos al grabarla.
3. La existencia de páginas con AJAX y otras sin esta tecnología hace que el usuario se desoriente.
4. Problemas con navegadores antiguos que no implementan esta tecnología.
5. No funciona si el usuario tiene desactivado el JavaScript en su navegador.
6. Requiere programadores que conozcan todas las tecnologías que intervienen en AJAX.
7. Dependiendo de la carga del servidor podemos experimentar tiempos tardíos de respuesta que desconciertan al visitante.

3 - Un ejemplo con AJAX.

Confeccionaremos un ejemplo donde veremos que aparecen muchos conceptos, no se preocupe si no los comprende en su totalidad ya que los mismos se verán en forma detallada a lo largo de este curso.

La idea fundamental de este ejercicio es conocer como debemos estructurar nuestras páginas y ver que introduce de nuevo el empleo de AJAX.

Confeccionaremos un problema muy sencillo, imaginemos que tenemos una lista de hipervínculos con los distintos signos del horóscopo y queremos que al ser presionado no recargue la página completa sino que se envíe una petición al servidor y el mismo retorne la información de dicho signo, luego se actualice solo el contenido de un div del archivo HTML.

Este problema se puede resolver muy fácilmente si refrescamos la página completamente al presionar el hipervínculo, pero nuestro objetivo es actualizar una pequeña parte de la página y más precisamente el div que debe mostrar los datos del signo seleccionado.

Si bien nuestra página solo contendrá los hipervínculos a los distintos signos en un caso real la página puede contener muchos otros elementos HTML con imágenes, otros hipervínculos etc. los cuales no deberán sufrir cambios (ni parpadeo) ya que solo se modificará el elemento div respectivo mediante DHTML.

Esta actualización parcial de la página tiene muchas ventajas:

- Reducimos el ancho de banda requerido al no tener que recuperar toda la página.
- Agilizamos la actualización de la página.
- Reducimos el parpadeo de la página.
- Hacemos más natural la navegación del sitio.

La mayoría de los problemas requieren los siguientes archivos como mínimo:

1. El archivo HTML (es la página que se ve en el navegador)
2. El archivo JS (contiene todas las rutinas JavaScript que permiten actualizar dinámicamente la página HTML (mediante DHTML) y las rutinas que permiten comunicarse con el servidor para el envío y recepción de información.
3. La hoja de estilo, es decir el archivo CSS
4. La página que contiene el script que se ejecuta en el servidor(en nuestro caso emplearemos el lenguaje PHP)

Comencemos a presentar los distintos archivos para resolver este problema:

pagina1.html

```
<html>
<head>
<title>Problema</title>
<script src="funciones.js" language="JavaScript"></script>
<link rel="StyleSheet" href="estilos.css" type="text/css">
</head>
<body>
<h1>Signos del horóscopo.</h1>
<div id="menu">
<p><a id="enlace1" href="pagina1.php?cod=1">Aries</a></p>
<p><a id="enlace2" href="pagina1.php?cod=2">Tauro</a></p>
<p><a id="enlace3" href="pagina1.php?cod=3">Geminis</a></p>
<p><a id="enlace4" href="pagina1.php?cod=4">Cancer</a></p>
<p><a id="enlace5" href="pagina1.php?cod=5">Leo</a></p>
<p><a id="enlace6" href="pagina1.php?cod=6">Virgo</a></p>
<p><a id="enlace7" href="pagina1.php?cod=7">Libra</a></p>
<p><a id="enlace8" href="pagina1.php?cod=8">Escorpio</a></p>
<p><a id="enlace9" href="pagina1.php?cod=9">Sagitario</a></p>
<p><a id="enlace10" href="pagina1.php?cod=10">Capricornio</a></p>
<p><a id="enlace11" href="pagina1.php?cod=11">Acuario</a></p>
<p><a id="enlace12" href="pagina1.php?cod=12">Piscis</a></p>
</div>
<div id="detalles">Seleccione su signo.</div>
</body>
</html>
```

Esta página contiene HTML puro. Es importante notar que debemos incorporar los dos archivos externos .css y .js mediante los elementos HTML respectivos:

```
<script src="funciones.js" language="JavaScript"></script>
<link rel="StyleSheet" href="estilos.css" type="text/css">
```

La hoja de estilo solo tiene el objetivo de mejorar la presentación en la página de los doce hipervínculos de los signos del horóscopo. Puede probar de eliminar el archivo .css mediante el borrado del elemento link del archivo HTML y el problema debería continuar funcionando, por supuesto con una presentación mucho más pobre.

Podemos observar que cada hipervínculo solicita la misma página al servidor pero pasándole como parámetro un valor distinto, con esto podremos detectar en el servidor que signo a elegido el operador.

El segundo archivo contiene las reglas de estilo que se definen para el archivo HTML:

estilos.css

```
#menu {
  font-family: Arial;
  margin: 5px;
}

#menu p {
  margin: 0px;
  padding: 0px;
}

#menu a {
  display: block;
  padding: 3px;
  width: 160px;
  background-color: #f7f8e8;
  border-bottom: 1px solid #eee;
  text-align: center;
}

#menu a:link, #menu a:visited {
  color: #f00;
  text-decoration: none;
}

#menu a:hover {
  background-color: #369;
  color: #fff;
}

#detalles {
  background-color: #ffc;
  text-align: left;
  font-family: verdana;
  border-width: 0;
  padding: 5px;
  border: 1px dotted #fa0;
  margin: 5px;
}
```

No haremos un análisis de estas reglas ya que corresponden al tema 28 del curso de [CSS Ya](#) "*Creación de un menú vertical configurando las pseudoclases*", puede refrescar los conceptos allí. Inclusive si todavía conoce poco de CSS y no quiere estudiarlo por ahora puede anular el archivo no incorporándolo en la página HTML suprimiendo el elemento link.

Ahora viene uno de los puntos claves donde debemos prestar más atención, esto se encuentra en las rutinas JavaScript que debemos implementar para comunicarnos con el servidor, además de lo ya conocido de DHTML para añadir elementos HTML en forma dinámica.

Veamos el archivo en su totalidad y expliquemos en forma muy global (recuerde que a lo largo de este curso iremos profundizando todos estos conceptos de comunicación con el servidor):

funciones.js

```

addEvent(window,'load',inicializarEventos,false);

function inicializarEventos()
{
    var ob;
    for(f=1;f<=12;f++)
    {
        ob=document.getElementById('enlace'+f);
        addEvent(ob,'click',presionEnlace,false);
    }
}

function presionEnlace(e)
{
    if (window.event)
    {
        window.event.returnValue=false;
        var url=window.event.srcElement.getAttribute('href');
        cargarHoroscopo(url);
    }
    else
    if (e)
    {
        e.preventDefault();
        var url=e.target.getAttribute('href');
        cargarHoroscopo(url);
    }
}

var conexion1;
function cargarHoroscopo(url)
{
    if(url=="")
    {
        return;
    }
    conexion1=crearXMLHttpRequest();
    conexion1.onreadystatechange = procesarEventos;
    conexion1.open("GET", url, true);
    conexion1.send(null);
}

function procesarEventos()
{
    var detalles = document.getElementById("detalles");
    if(conexion1.readyState == 4)
    {
        detalles.innerHTML = conexion1.responseText;
    }
    else
    {
        detalles.innerHTML = 'Cargando...';
    }
}

```

```

//*****
//Funciones comunes a todos los problemas
//*****
function addEvent(elemento,nomevento,funcion,captura)
{
    if (elemento.attachEvent)
    {
        elemento.attachEvent('on'+nomevento,funcion);
        return true;
    }
    else
    if (elemento.addEventListener)
    {
        elemento.addEventListener(nomeevento,funcion,captura);
        return true;
    }
    else
        return false;
}

function crearXMLHttpRequest()
{
    var xmlHttp=null;
    if (window.ActiveXObject)
        xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
    else
        if (window.XMLHttpRequest)
            xmlHttp = new XMLHttpRequest();
    return xmlHttp;
}

```

En este punto si es indispensable haber realizado el curso de [DHTML Ya](#) para entender como registramos los eventos para los doce hipervínculos. Recordemos que siempre llamaremos a la función addEvent (que se encuentra codificada en el mismo archivo) para hacer compatible nuestro código con el navegador IE (recordemos que no cumple los estándares referente a eventos).

Lo primero que se ejecuta es la llamada a la función inicializarEventos() inmediatamente luego que la página se a cargado por completo en el navegador:

```

function inicializarEventos()
{
    var ob;
    for (f=1;f<=12;f++)
    {
        ob=document.getElementById('enlace'+f);
        addEvent(ob, 'click', presionEnlace, false);
    }
}

```

En esta función registramos el evento click para los doce enlaces de los signos del horóscopo. Para facilitar la codificación recordemos que todos tienen casi el mismo nombre, difieren por un número al final. Luego dentro de un for rescatamos la referencia a cada enlace y registramos el evento click indicando que se debe llamar a la función presionEnlace.

La función presión enlace:

```
function presionEnlace(e)
{
    if (window.event)
    {
        window.event.returnValue=false;
        var url=window.event.srcElement.getAttribute('href');
        cargarHoroscopo(url);
    }
    else
    {
        if (e)
        {
            e.preventDefault();
            var url=e.target.getAttribute('href');
            cargarHoroscopo(url);
        }
    }
}
```

Primero detecta que navegador se trata y procede a desactivar el evento por defecto para el hipervínculo, luego llama a la función cargarHoroscopo pasandole como referencia la url que contiene el hipervínculo.

Todo lo comentado hasta acá se estudió en cursos anteriores: HTML, JavaScript, CSS, DHTML.

Veamos ahora la función cargarHoroscopo:

```
var conexion1;
function cargarHoroscopo(url)
{
    if(url=='')
    {
        return;
    }
    conexion1=crearXMLHttpRequest()
    conexion1.onreadystatechange = procesarEventos;
    conexion1.open("GET", url, true);
    conexion1.send(null);
}
```

Previo a la definición de esta función definimos una variable global llamada conexion1 que será utilizada en esta y la siguiente función.

La función recibe como parámetro la url a la que debe hacer la petición de datos. Lo primero que verificamos que el parámetro no llegue vacío, en caso de estar vacío salimos con el comando return.

El siguiente paso es llamar a la función crearXMLHttpRequest que crea y retorna un objeto de la clase XMLHttpRequest (luego veremos que este objeto nos permite comunicarnos con el servidor de forma asincrónica):

```
conexion1=crearXMLHttpRequest()
```

Esta función se encuentra codificada más abajo dentro del mismo archivo y tiene por objetivo retornar un objeto de la clase XMLHttpRequest.

La creación del objeto de la clase XMLHttpRequest se implementa separada en otra función porque depende del navegador que se trate la sintaxis cambia:

```
function crearXMLHttpRequest()
```

```

{
    var xmlHttp=null;
    if (window.ActiveXObject)
        xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
    else
        if (window.XMLHttpRequest)
            xmlHttp = new XMLHttpRequest();
    return xmlHttp;
}

```

Como podemos observar verificamos si se trata del navegador IE, en caso afirmativo la creación del objeto XMLHttpRequest es:

```

    if (window.ActiveXObject)
        xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");

```

Si no se trata del navegador IE verificamos si existe la clase XMLHttpRequest en el objeto window, en caso afirmativo creamos un objeto de dicha clase:

```

    if (window.XMLHttpRequest)
        xmlHttp = new XMLHttpRequest();

```

Por la diferencia en la forma de crear un objeto de la clase XMLHttpRequest hemos movido esta actividad a esta función.

La función retorna la referencia del objeto creado:

```

    return xmlHttp;

```

Retornemos a la función cargarHoroscopo y veamos que hacemos con el objeto de la clase XMLHttpRequest que acabamos de crear:

```

conexion1=crearXMLHttpRequest()
conexion1.onreadystatechange = procesarEventos;
conexion1.open("GET", url, true);
conexion1.send(null);

```

La propiedad onreadystatechange se inicializa con la referencia de una función que será la encargada de procesar los datos enviados por el servidor, veremos el código de esta función más adelante.

Seguidamente llamamos al método open que tiene tres parámetros:

- Primero el método de envío de datos (GET o POST) Recordemos que si los datos se envían como parámetros (como es nuestro ejemplo) debemos indicar que utilizamos el método GET
- El segundo parámetro es la url y la página que procesará los datos que le enviemos.
- El tercer parámetro indicamos si se procesarán los datos de forma asíncrona (true) o síncrona (false)

Por último nos falta llamar al método send para que comience el proceso:

```

conexion1.send(null);

```

Nos queda explicar la función `procesarEventos` que se ejecuta cada vez que el objeto `conexion1` de la clase `XMLHttpRequest` cambia de estado. Tengamos en cuenta que los estados posibles de este objeto son:

- 0 No inicializado.
- 1 Cargando.
- 2 Cargado.
- 3 Interactivo.
- 4 Completado.

Para conocer el estado del objeto debemos acceder a la propiedad `readyState` que almacena alguno de los cinco valores que enunciamos.

Nuestra función `procesarEventos` es:

```
function procesarEventos()
{
    var detalles = document.getElementById("detalles");
    if(conexion1.readyState == 4)
    {
        detalles.innerHTML = conexion1.responseText;
    }
    else
    {
        detalles.innerHTML = 'Cargando...';
    }
}
```

Decíamos que cuando la propiedad `readyState` almacena 4 significa que todos los datos han llegado desde el servidor, luego mediante el método `responseText` recuperamos la información enviada por el servidor. Luego cualquier otro valor que contenga la propiedad `readyState` mostramos dentro del `div` el mensaje 'cargando...'.

Es seguro que muchas dudas han surgido de este primer pantallazo de AJAX, pero no se preocupe a medida que avancemos en el curso se irán aclarando e internalizando.

Pero todavía nos queda la página que contiene el programa en el servidor, en nuestro caso empleamos el lenguaje PHP (tener en cuenta que podemos emplear otro lenguaje de servidor para esto)

Veamos el código de esta página:

```
if ($_REQUEST['cod']==1)
    echo "<strong>Aries:</strong> Hoy los cambios serán físicos, personales, de
carácter, Te sentirás
impulsivo y tomarás iniciativas. Período en donde considerarás unirte a
agrupaciones de
beneficencia, o de ayuda a los demás.";
if ($_REQUEST['cod']==2)
    echo "<strong>Tauro:</strong> Hoy los cambios serán privados, íntimos.
Recuerdos. Ayuda,
solidaridad. Asuntos en lugares de retiro. Tu cónyuge puede aportar buen status a
tu vida o apoyo a
tu profesión.";
if ($_REQUEST['cod']==3)
    echo "<strong>Géminis:</strong> Los asuntos de hoy tienen que ver con las
amistades, reuniones,
actividades con ellos. Día esperanzado, ilusiones. Mucha energía sexual y fuerza
emocional. Deseos
difíciles de controlar.";
if ($_REQUEST['cod']==4)
    echo "<strong>Cancer:</strong> Este día la profesión y las relaciones con
superiores y con tu
madre serán de importancia. Actividad en relación a estos temas. Momentos
positivos con compañeros
de trabajo. Actividad laboral agradable.";
if ($_REQUEST['cod']==5)
    echo "<strong>Leo:</strong> Este día los estudios, los viajes, el extranjero y la
espiritualidad
serán lo importante. Pensamientos, religión y filosofía también. Vivencias
kármicas de la época te
vuelven responsable tomando decisiones.";
if ($_REQUEST['cod']==6)
    echo "<strong>Virgo:</strong> Para este día toma importancia tu vida sexual,
tal vez miedos, temas
legales, juicios o herencias. Experiencias extrañas. Hay karma de prueba durante
este período en tu
parte psicológica, generándose algunos replanteos.";
if ($_REQUEST['cod']==7)
    echo "<strong>Libra:</strong> Hoy todo asunto tiene que ver con tu pareja,
también con socios, con
la gente o el público. Ellos serán lo más importante del día. Ganancias a través de
especulaciones o
del juego. Actividades vocacionales artísticas.";
if ($_REQUEST['cod']==8)
    echo "<strong>Escorpio:</strong> Hoy todo asunto tiene que ver con temas de
trabajo y de salud.
Presta atención a ambos. Experiencias diversas con compañeros. Durante este
período tendrás muchos
recursos para ganar dinero.";
if ($_REQUEST['cod']==9)
    echo "<strong>Sagitario:</strong> Durante este día se vivirán cambios en
relación a los noviazgos
```

```

o a los hijos. Creatividad, actividad, diversiones y salidas. Período de encuentros
con personas o
situaciones que te impresionan.";
if ($_REQUEST['cod']==10)
    echo "<strong>Capricornio:</strong> Los cambios del día tienen que ver con tu
hogar, con la
convivencia y con el padre. Asuntos relativos al carácter en la convivencia. El
karma de
responsabilidad de estos momentos te acercará al mundo de lo desconocido,
mucha madurez y contacto
con el más allá.";
if ($_REQUEST['cod']==11)
    echo "<strong>Acuario:</strong> Hoy todo asunto tiene que ver con el entorno
inmediato, hermanos y
vecinos, con la comunicación, los viajes cortos o traslados frecuentes. El hablar y
trasladarse será
importante hoy. Mentalidad e ideas activas.";
if ($_REQUEST['cod']==12)
    echo "<strong>Piscis:</strong> Durante este día se vivirán cambios en la
economía, movimientos en
los ingresos, negocios, valores. Momentos de gran fuerza y decisión
profesionales, buscarás el
liderazgo.";
?>

```

Mediante el vector asociativo \$_REQUEST recuperamos el valor del parámetro cod y mediante una serie de if verificamos si almacena el valor 1 procedemos a generar un texto referente al signo Aries, si tiene un 2 generamos un texto referente al signo Tauro y así sucesivamente.

Hay que tener en cuenta que no se estará enviando una página HTML completa, por eso no tiene los elementos Head, Body etc. sino es más bien un archivo de texto que luego será añadido en forma dinámica al div de la página HTML.

Debe quedar claro que los datos se podrían haber rescatado perfectamente de una base de datos, pero por simplicidad hemos dispuesto estos 12 if y generado el texto respectivo. Veremos más adelante problemas que acceden a bases de datos.

Hasta acá el primer problema de AJAX. Le recomiendo pasar a la sección de "Problemas Resueltos" y ejecutar este ejercicio, releer nuevamente estos conceptos y tratar de hacer modificaciones sencillas al problema.

4 - El mismo ejemplo sin AJAX.

Volveremos a confeccionar el mismo problema que muestra una lista de hipervínculos con los distintos signos del horóscopo, a diferencia del problema expuesta en el concepto anterior ahora no emplearemos AJAX, es decir recargaremos la página completamente al presionar alguno de los hipervínculos.

Como debemos recargar la página y actualizar en el servidor los datos del signo del horóscopo seleccionado confeccionaremos solo una página php.

```

<html>
<head>
<title>Problema</title>
<link rel="StyleSheet" href="estilos.css" type="text/css">
</head>
<body>
<h1>Signos del horóscopo.</h1>
<div id="menu">
<p><a id="enlace1" href="pagina1.php?cod=1">Aries</a></p>
<p><a id="enlace2" href="pagina1.php?cod=2">Tauro</a></p>
<p><a id="enlace3" href="pagina1.php?cod=3">Geminis</a></p>
<p><a id="enlace4" href="pagina1.php?cod=4">Cancer</a></p>
<p><a id="enlace5" href="pagina1.php?cod=5">Leo</a></p>
<p><a id="enlace6" href="pagina1.php?cod=6">Virgo</a></p>
<p><a id="enlace7" href="pagina1.php?cod=7">Libra</a></p>
<p><a id="enlace8" href="pagina1.php?cod=8">Escorpio</a></p>
<p><a id="enlace9" href="pagina1.php?cod=9">Sagitario</a></p>
<p><a id="enlace10" href="pagina1.php?cod=10">Capricornio</a></p>
<p><a id="enlace11" href="pagina1.php?cod=11">Acuario</a></p>
<p><a id="enlace12" href="pagina1.php?cod=12">Piscis</a></p>
</div>
<div id="detalles">
<?php
if (!isset($_REQUEST['cod']))
    echo "Seleccione su signo.";
if ($_REQUEST['cod']==1)
    echo "<strong>Aries:</strong> Hoy los cambios serán físicos,
    personales, de carácter, Te sentirás impulsivo y tomarás
    iniciativas. Período en donde considerarás unirte a agrupaciones
    de beneficencia, o de ayuda a los demás.";
if ($_REQUEST['cod']==2)
    echo "<strong>Tauro:</strong> Hoy los cambios serán privados,
    íntimos. Recuerdos. Ayuda, solidaridad. Asuntos en lugares de retiro. Tu
    cónyuge puede aportar buen status a tu vida o apoyo a tu profesión.";
if ($_REQUEST['cod']==3)
    echo "<strong>Géminis:</strong> Los asuntos de hoy tienen
    que ver con las amistades, reuniones, actividades con ellos. Día esperanzado,
    ilusiones. Mucha energía sexual y fuerza emocional. Deseos difíciles
    de controlar.";
if ($_REQUEST['cod']==4)
    echo "<strong>Cancer:</strong> Este día la profesión
    y las relaciones con superiores y con tu madre serán de importancia. Actividad
    en relación a estos temas. Momentos positivos con compañeros de
    trabajo. Actividad laboral agradable.";
if ($_REQUEST['cod']==5)
    echo "<strong>Leo:</strong> Este día los estudios, los
    viajes, el extranjero y la espiritualidad serán lo importante. Pensamientos,
    religión y filosofía también. Vivencias kármicas de
    la época te vuelven responsable tomando decisiones.";
if ($_REQUEST['cod']==6)
    echo "<strong>Virgo:</strong> Para este día toma importancia
    tu vida sexual, tal vez miedos, temas legales, juicios o herencias. Experiencias
    extrañas. Hay karma de prueba durante este período en tu parte psicológica,
    generándose algunos replanteos.";

```

```

if ($_REQUEST['cod']==7)
    echo "<strong>Libra:</strong> Hoy todo asunto tiene que ver con
    tu pareja, también con socios, con la gente o el público. Ellos
    serán lo más importante del día. Ganancias a través
    de especulaciones o del juego. Actividades vocacionales artísticas.";
if ($_REQUEST['cod']==8)
    echo "<strong>Escorpio:</strong> Hoy todo asunto tiene que ver
    con temas de trabajo y de salud. Presta atención a ambos. Experiencias
    diversas con compañeros. Durante este período tendrás muchos
    recursos para ganar dinero.";
if ($_REQUEST['cod']==9)
    echo "<strong>Sagitario:</strong> Durante este día se
    vivirán cambios en relación a los noviazgos o a los hijos. Creatividad,
    actividad, diversiones y salidas. Período de encuentros con personas o
    situaciones que te impresionan.";
if ($_REQUEST['cod']==10)
    echo "<strong>Capricornio:</strong> Los cambios del día
    tienen que ver con tu hogar, con la convivencia y con el padre. Asuntos relativos
    al carácter en la convivencia. El karma de responsabilidad de estos momentos
    te acercará al mundo de lo desconocido, mucha madurez y contacto con el
    más allá.";
if ($_REQUEST['cod']==11)
    echo "<strong>Acuario:</strong> Hoy todo asunto tiene que ver
    con el entorno inmediato, hermanos y vecinos, con la comunicación, los
    viajes cortos o traslados frecuentes. El hablar y trasladarse será importante
    hoy. Mentalidad e ideas activas.";
if ($_REQUEST['cod']==12)
    echo "<strong>Piscis:</strong> Durante este día se vivirán
    cambios en la economía, movimientos en los ingresos, negocios, valores.
    Momentos de gran fuerza y decisión profesionales, buscarás el liderazgo.";
?>
</div>
</body>
</html>

```

Como podemos observar los hipervínculos llaman a la misma página:

```
href="pagina1.php?cod=1">Aries</a></p>
```

Luego el código PHP que se ejecuta en el servidor verifica el valor que llega como parámetro y muestra el detalle del signo del horóscopo seleccionado:

```

if ($_REQUEST['cod']==1)
    echo "<strong>Aries:</strong> Hoy los cambios serán físicos,
    personales, de carácter, Te sentirás impulsivo y tomarás
    iniciativas. Período en donde considerarás unirte a agrupaciones
    de beneficencia, o de ayuda a los demás.";

```

En caso de no llegar parámetros a la página (normalmente la primer vez que cargamos la página) el primer if se verifica verdadero:

```

if (!isset($_REQUEST['cod']))
    echo "Seleccione su signo.";

```

Si comparamos este ejemplo con el anterior veremos que utilizar AJAX reduce la cantidad de información que pedimos al servidor, también evitamos la recarga completa de la página (imaginemos un sitio que contiene muchos elementos el redibujado es lento y engorroso)

La hoja de estilo no tiene cambios con respecto al problema anterior:

```
#menu {
  font-family: Arial;
  margin: 5px;
}

#menu p {
  margin: 0px;
  padding: 0px;
}

#menu a {
  display: block;
  padding: 3px;
  width: 160px;
  background-color: #f7f8e8;
  border-bottom: 1px solid #eee;
  text-align: center;
}

#menu a:link, #menu a:visited {
  color: #f00;
  text-decoration: none;
}

#menu a:hover {
  background-color: #369;
  color: #fff;
}

#detalles {
  background-color: #ffc;
  text-align: left;
  font-family: verdana;
  border-width: 0;
  padding: 5px;
  border: 1px dotted #fa0;
  margin: 5px;
}
```

5 - Objeto XMLHttpRequest

El objeto XMLHttpRequest es un elemento fundamental para la comunicación asincrónica con el servidor. Este objeto nos permite enviar y recibir información en formato XML y en general en cualquier formato (como vimos en el ejercicio anterior retornando un trozo de archivo HTML)

La creación de un objeto de esta clase varía si se trata del Internet Explorer de Microsoft, ya que este no lo incorpora en JavaScript sino que se trata de una ActiveX:

```
if (window.ActiveXObject)
    xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
```

En cambio en FireFox y otros navegadores lo incorpora JavaScript y procedemos para su creación de la siguiente manera:

```
if (window.XMLHttpRequest)
    xmlhttp = new XMLHttpRequest();
```

Como hemos visto en el problema anterior siempre implementaremos una función que nos retorne un objeto XMLHttpRequest haciendo transparente el proceso en cuanto a navegador donde se esté ejecutando:

```
function crearXMLHttpRequest()
{
    var xmlhttp=null;
    if (window.ActiveXObject)
        xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
    else
        if (window.XMLHttpRequest)
            xmlhttp = new XMLHttpRequest();
    return xmlhttp;
}
```

Es decir la función crearXMLHttpRequest se encargará de retornarnos un objeto de la clase XMLHttpRequest.

Las **propiedades** principales del objeto XMLHttpRequest son:

- *onreadystatechange* Almacena el nombre de la función que se ejecutará cuando el objeto XMLHttpRequest cambie de estado.
- *readyState* Almacena el estado del requerimiento hecho al servidor, pudiendo ser:
 - 0 No inicializado (el método open no a sido llamado)
 - 1 Cargando (se llamó al método open)
 - 2 Cargado (se llamó al método send y ya tenemos la cabecera de la petición HTTP y el status)
 - 3 Interactivo (la propiedad responseText tiene datos parciales)
 - 4 Completado (la propiedad responseText tiene todos los datos pedidos al servidor)
- *responseText* Almacena el string devuelto por el servidor, luego de haber hecho una petición.
- *responseXML* Similar a la anterior (responseText) con la diferencia que el string devuelto por el servidor se encuentra en formato XML.

Los **métodos** principales del objeto XMLHttpRequest son:

- *open* Abre un requerimiento HTTP al servidor.
- *send* Envía el requerimiento al servidor.

Confeccionaremos otro problema para fijar conceptos vistos hasta el momento.

Confeccionar una página que muestre una imagen y permita calificarla con un valor entre 1 y 10. Permitir ingresar el nombre del visitante. Disponer de un control de tipo select para seleccionar el valor. Luego al presionar un botón enviar el valor seleccionado utilizando el objeto XMLHttpRequest al servidor donde almacenaremos en un archivo de texto el nombre del

visitante y el puntaje. Retornar luego todos los votos hasta el momento. Actualizaremos la página HTML con todos los nombres y votos hasta el momento.

El archivo HTML es:

paginal.html

```
<html>
<head>
<title>Problema</title>
<script src="funciones.js" language="JavaScript"></script>
</head>
<body>
<h1>Vote esta foto</h1>
<p>

</p>
Nombre: <input type="text" id="nombre" size="20"> <br>
Voto: <select id="voto">
<option value="0" selected>seleccione</option>
<option value="1">1</option>
<option value="2">2</option>
<option value="3">3</option>
<option value="4">4</option>
<option value="5">5</option>
<option value="6">6</option>
<option value="7">7</option>
<option value="8">8</option>
<option value="9">9</option>
<option value="10">10</option>
</select> <br>
<input type="button" id="boton1" value="votar">
<div id="resultados"></div>
</body>
</html>
```

Lo primero que podemos observar es que no utilizaremos una hoja de estilo para reducir la complejidad del problema y concentrarnos en la lógica.

Si vemos utilizamos controles de tipo input, select y button, pero no disponemos ningún formulario. Esto se debe a que los datos ingresados se enviarán en forma asíncrona mediante el objeto XMLHttpRequest.

Otro punto a destacar que a cada control le definimos la propiedad id, esta es de suma importancia para poder accederla desde JavaScript. No definimos la propiedad name ya que no se enviarán los datos por medio de formulario.

Nuestro archivo con las funciones JavaScript es:

funciones.js

```
addEvent(window,'load',inicializarEventos,false);

function inicializarEventos()
{
  var ob=document.getElementById('boton1');
  addEvent(ob,'click',presionBoton,false);
}
```

```

}

function presionBoton(e)
{
    var ob1=document.getElementById('voto');
    var ob2=document.getElementById('nombre');
    cargarVoto(ob1.value,ob2.value);
}

var conexion1;
function cargarVoto(voto,nom)
{
    conexion1=crearXMLHttpRequest();
    conexion1.onreadystatechange = procesarEventos;
    conexion1.open('GET','pagina1.php?puntaje='+voto+'&nombre='+nom, true);
    conexion1.send(null);
}

function procesarEventos()
{
    var resultados = document.getElementById("resultados");
    if(conexion1.readyState == 4)
    {
        resultados.innerHTML = conexion1.responseText;
    }
    else
    {
        resultados.innerHTML = 'Cargando...';
    }
}

//*****
//Funciones comunes a todos los problemas
//*****
function addEvent(elemento,nomevento,funcion,captura)
{
    if (elemento.attachEvent)
    {
        elemento.attachEvent('on'+nomevento,funcion);
        return true;
    }
    else
    if (elemento.addEventListener)
    {
        elemento.addEventListener(nomeevento,funcion,captura);
        return true;
    }
    else
        return false;
}

function crearXMLHttpRequest()
{
    var xmlhttp=null;
    if (window.ActiveXObject)

```

```

xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
else
    if (window.XMLHttpRequest)
        xmlHttp = new XMLHttpRequest();
return xmlHttp;
}

```

Como podemos observar tiene mucho en común con el primer ejemplo de AJAX que habíamos desarrollado.

Lo primero que hacemos es inicializar el evento load con la función inicializarEventos, en esta inicializamos el evento click del único botón que contiene la página:

```

addEvent(window, 'load', inicializarEventos, false);

function inicializarEventos()
{
    var ob=document.getElementById('boton1');
    addEvent(ob, 'click', presionBoton, false);
}

```

Cuando se presiona el botón se dispara la función presionBoton, donde obtenemos la referencia a los dos controles (select y text) que tienen almacenados los valores. Llamamos finalmente a la función cargarVoto:

```

function presionBoton(e)
{
    var ob1=document.getElementById('voto');
    var ob2=document.getElementById('nombre');
    cargarVoto(ob1.value, ob2.value);
}

```

La función cargarVoto recibe como parámetro el valor del voto y el nombre del visitante, seguidamente llama a la función crearXMLHttpRequest.

Por último inicializamos la propiedad onreadystatechange y llamamos a los métodos open y send. En el método open pasamos los dos datos en la cabecera de la petición de página.

```

var conexion1;
function cargarVoto(voto, nom)
{
    conexion1=crearXMLHttpRequest()
    conexion1.onreadystatechange = procesarEventos;
    conexion1.open('GET', 'paginal.php?puntaje='+voto+'&nombre='+nom,
true);
    conexion1.send(null);
}

```

Nos queda la función procesarEventos, que cuando la propiedad readyState del objeto XMLHttpRequest tiene un valor 4 (proceso completado) recupera el valor de la propiedad responseText con la información que se retornó desde el servidor:

```

function procesarEventos()
{
    var resultados = document.getElementById("resultados");
    if(conexion1.readyState == 4)
    {

```

```

    resultados.innerHTML = conexion1.responseText;
}
else
{
    resultados.innerHTML = 'Cargando...';
}
}

```

El archivo PHP que se ejecuta en el servidor es el siguiente:

paginal.php

```

<?php
header('Content-Type: text/html; charset=ISO-8859-1');
$ar=fopen("puntaje.txt","a") or
    die("No se pudo abrir el archivo");
fputs($ar,"Nombre:".$_REQUEST['nombre']."<br>");
fputs($ar,"Voto:".$_REQUEST['puntaje']."<br><br>");
fclose($ar);
$ar=fopen("puntaje.txt","r") or
    die("No se pudo abrir el archivo");
while (!feof($ar))
{
    $linea=fgets($ar);
    echo $linea;
}
fclose($ar);
?>

```

Lo primero que hacemos es abrir el archivo para agregar datos, es decir no borramos los votos existentes (puede probar de cambiar "a" de append por "w" que crea el archivo):

```

$ar=fopen("puntaje.txt","a") or
    die("No se pudo abrir el archivo");

```

Luego recuperamos los parámetros que llegan a la página y los grabamos:

```

fputs($ar,"Nombre:".$_REQUEST['nombre']."<br>");
fputs($ar,"Voto:".$_REQUEST['puntaje']."<br><br>");

```

Cerramos y abrimos nuevamente el archivo, pero ahora con el objetivo de leerlo:

```

fclose($ar);
$ar=fopen("puntaje.txt","r") or
    die("No se pudo abrir el archivo");

```

Por último generamos el archivo a retornar al navegador:

```

while (!feof($ar))
{
    $linea=fgets($ar);
    echo $linea;
}

```

Con este segundo ejemplo debemos poder identificar que partes son comunes al problema anterior.

6 - Pasando datos al servidor por el método GET.

Para indicar cual es el método de envío de los datos al servidor lo hacemos en el primer parámetro del método open del objeto XMLHttpRequest:

```
conexion1.open('GET','paginal.php?puntaje='+voto+'&nombre='+nom,
true);
```

En este ejemplo vemos como indicamos que el envío de los datos se hace por el método GET. Si lo hacemos de esta forma tenemos que tener mucho cuidado en la codificación del segundo parámetro del método open donde indicamos el nombre de la página a pedir.

Seguido al nombre de la página debe ir el signo de interrogación, el nombre del parámetro, luego un igual y el valor del parámetro. En caso de haber más de un parámetro debemos separarlos mediante el caracter ampersand.

Por último el tercer parámetro del método open normalmente se pasa el valor true indicando que el requerimiento de la página es asíncrona (esto permite al visitante continuar interactuando con la página sin que se congele hasta llegar la solicitud)

Confeccionaremos un problema similar al anterior, es decir que nos permita calificar una foto con un valor de 1 al 10. La diferencia es que la calificación será por medio de una lista de hipervínculos del 1 al 10. Además haremos que la barra de selección de la calificación cambie de color cuando ingresamos con el mouse.

Veremos que la mayor dificultad se encuentra en la codificación de esta última característica más que en la comunicación asíncrona.

El archivo HTML es (pagina1.html):

```
<html>
<head>
<title>Problema</title>
<script src="funciones.js" language="JavaScript"></script>
<link rel="StyleSheet" href="estilos.css" type="text/css">
</head>
<body>
<h1>Vote esta foto</h1>
<p>

</p>
<ul class="voto" id="votofoto1">
<li><a href="pagina1.php?voto=1">1</a></li>
<li><a href="pagina1.php?voto=2">2</a></li>
<li><a href="pagina1.php?voto=3">3</a></li>
<li><a href="pagina1.php?voto=4">4</a></li>
<li><a href="pagina1.php?voto=5">5</a></li>
```

```

<li><a href="pagina1.php?voto=6">6</a></li>
<li><a href="pagina1.php?voto=7">7</a></li>
<li><a href="pagina1.php?voto=8">8</a></li>
<li><a href="pagina1.php?voto=9">9</a></li>
<li><a href="pagina1.php?voto=10">10</a></li>
</ul>
<br>
<div id="resultados"></div>
<a href="votos.txt">Ver resultados</a>
</body>
</html>

```

La foto se encuentra en el directorio inmediatamente superior por eso indicamos el ../ :

```

```

Para no complicar más el problema el archivo de resultados se almacena en un archivo de texto llamado votos.txt (al presionar el hipervínculo procedemos a mostrarlo en el navegador, esto nos podrá permitir verificar si realmente el voto se registró en el servidor):

```
<a href="votos.txt">Ver resultados</a>
```

Cada hipervínculo pasa como parámetro la calificación respectiva:

```

<li><a href="pagina1.php?voto=1">1</a></li>
<li><a href="pagina1.php?voto=2">2</a></li>
<li><a href="pagina1.php?voto=3">3</a></li>

```

La hoja de estilo queda definida como sigue (estilos.css):

```

.voto {
    padding:0px;
}
.voto a {
    float:left;
    width:15px;
    text-decoration:none;
    text-align:center;
    color:#f00;
    background-color:#f7f8e8;
    border-right:1px solid white;
    font-size:13px;
}

.voto li {
    display:inline;
}

```

Como la lista de hipervínculos los queremos todos en la misma línea inicializamos la propiedad display con el valor inline:

```

.voto li {
    display:inline;
}

```

Luego donde se encuentra la complejidad mayor es en funciones.js:

```
addEvent(window,'load',inicializarEventos,false);

function inicializarEventos()
{
  var ref=document.getElementById('votofoto1');
  var vec= ref.getElementsByTagName('li');
  var vec2=ref.getElementsByTagName('a');
  for(f=0;f<vec2.length;f++)
  {
    addEvent(vec[f],'mouseover',entrar,false);
    addEvent(vec[f],'mouseout',salir,false);
    addEvent(vec2[f],'click',presionBoton,false);
  }
}

function entrar(e)
{
  var ref;
  if (window.event)
    ref=window.event.srcElement;
  else
    if (e)
      ref=e.target;
  var ob=document.getElementById('votofoto1');
  var vec=ob.getElementsByTagName('li');
  for(f=0;f<ref.firstChild.nodeValue;f++)
  {
    vec[f].firstChild.style.background='#f00';
    vec[f].firstChild.style.color='#fff';
  }
}

function salir(e)
{
  var ref;
  if (window.event)
    ref=window.event.srcElement;
  else
    if (e)
      ref=e.target;
  var ob=document.getElementById('votofoto1');
  var vec=ob.getElementsByTagName('li');
  for(f=0;f<ref.firstChild.nodeValue;f++)
  {
    vec[f].firstChild.style.background='#f7f8e8';
    vec[f].firstChild.style.color='#f00';
  }
}

function presionBoton(e)
{
  var ref;
```

```

    if (window.event)
    {
        window.event.returnValue=false;
        ref=window.event.srcElement;
    }
    else
    if (e)
    {
        e.preventDefault();
        ref=e.target;
    }
    cargarVoto(ref.firstChild.nodeValue);
}

var conexion1;
function cargarVoto(voto)
{
    conexion1=crearXMLHttpRequest();
    conexion1.onreadystatechange = procesarEventos;
    var aleatorio=Math.random();
    conexion1.open('GET','pagina1.php?voto='+voto+"&aleatorio="+aleatorio,
    true);
    conexion1.send(null);
}

function procesarEventos()
{
    var resultados = document.getElementById("resultados");
    if(conexion1.readyState == 4)
    {
        resultados.innerHTML = 'Gracias.';
    }
    else
    {
        resultados.innerHTML = 'Procesando...';
    }
}

//*****
//Funciones comunes a todos los problemas
//*****
function addEvent(elemento,nomevento,funcion,captura)
{
    if (elemento.attachEvent)
    {
        elemento.attachEvent('on'+nomevento,funcion);
        return true;
    }
    else
    if (elemento.addEventListener)
    {
        elemento.addEventListener(nomeevento,funcion,captura);
        return true;
    }
}

```

```

    else
        return false;
}

function crearXMLHttpRequest()
{
    var xmlHttp=null;
    if (window.ActiveXObject)
        xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
    else
        if (window.XMLHttpRequest)
            xmlHttp = new XMLHttpRequest();
    return xmlHttp;
}

```

La primera función que se ejecuta cuando se termina de cargar por completo la página es inicializarEventos:

Obtenemos la referencia al div que contiene la foto:

```
var ref=document.getElementById('votofoto1');
```

Luego obtenemos a partir de la referencia de dicho div todos los elementos li (list item) y a (anchor):

```
var vec= ref.getElementsByTagName('li');
var vec2=ref.getElementsByTagName('a');
```

Pasamos a la asignación de eventos a estos elementos dentro de un for, a los elementos de tipo li le definimos los eventos mouseover y mouseout, luego a los elementos a le definimos el evento click:

```
for(f=0;f<vec2.length;f++)
{
    addEvent(vec[f], 'mouseover', entrar, false);
    addEvent(vec[f], 'mouseout', salir, false);
    addEvent(vec2[f], 'click', presionBoton, false);
}
```

La función entrar se ejecuta cuando la flecha del mouse se dispone dentro de algún elemento de la lista. Luego dentro de un for cambiamos el color de todos los elementos menores e incluyendo al que emitió el evento:

```
function entrar(e)
{
    var ref;
    if (window.event)
        ref=window.event.srcElement;
    else
        if (e)
            ref=e.target;
    var ob=document.getElementById('votofoto1');
    var vec=ob.getElementsByTagName('li');
    for(f=0;f<ref.firstChild.nodeValue;f++)
    {
        vec[f].firstChild.style.background='#f00';
        vec[f].firstChild.style.color='#fff';
    }
}
```

```

    }
}

```

El algoritmo es similar cuando sacamos la flecha del mouse de un elemento de tipo li:

```

function salir(e)
{
    var ref;
    if (window.event)
        ref=window.event.srcElement;
    else
        if (e)
            ref=e.target;
    var ob=document.getElementById('votofoto1');
    var vec=ob.getElementsByTagName('li');
    for(f=0;f<ref.firstChild.nodeValue;f++)
    {
        vec[f].firstChild.style.background='#f7f8e8';
        vec[f].firstChild.style.color='#f00';
    }
}

```

La función presionBoton se ejecuta cuando alguno de los elementos a (anchor) es presionado:

```

function presionBoton(e)
{
    var ref;
    if (window.event)
    {
        window.event.returnValue=false;
        ref=window.event.srcElement;
    }
    else
        if (e)
        {
            e.preventDefault();
            ref=e.target;
        }
    cargarVoto(ref.firstChild.nodeValue);
}

```

En la función presionBoton desactivamos la acción por defecto de un elemento "a" llamando a la función e.preventDefault() o en su defecto si nos encontramos con el navegador IE inicializamos la propiedad returnValue con false (window.event.returnValue=false).

Procedemos luego a llamar a la función cargarVoto pasando como referencia el valor contenido en el elemento "a".

Veamos ahora la función cargarVoto:

```

var conexion1;
function cargarVoto(voto)
{
    conexion1=crearXMLHttpRequest();
    conexion1.onreadystatechange = procesarEventos;
    var aleatorio=Math.random();
    conexion1.open('GET','paginal.php?
voto='+voto+"&aleatorio="+aleatorio, true);
    conexion1.send(null);
}

```

```
}
```

Creamos un objeto de la clase XMLHttpRequest llamando a la función crearXMLHttpRequest(). Inicializamos la propiedad onreadystatechange con el nombre de la función que se disparará cada vez que el objeto XMLHttpRequest cambie de estado.

Ahora llamamos a la función open del objeto XMLHttpRequest indicando:

```
var aleatorio=Math.random();
conexion1.open('GET','pagina1.php?
voto='+voto+"&aleatorio="+aleatorio, true);
```

El método de conexión utilizado, en este caso es 'GET', como segundo parámetro la página que procesará el dato enviado al servidor (en nuestro caso enviamos el valor entero comprendido entre 1 y 10, más precisamente el que seleccionó el operador) El dato a enviar se llama voto y almacena el contenido de la variable voto que llega como parámetro a esta función.

Otra cosa que vamos a incorporar es el envío de un valor aleatorio. Esto es necesario si el navegador está configurado que recupere las páginas del cache. Imaginemos que votamos y asignamos el valor 10, luego si volvemos a seleccionar el 10 puede recuperar la pagina1.php? voto=10 del cache y no actualizar el archivo de texto con el voto seleccionado. La solución mas sencilla es enviar un parámetro con un valor aleatorio, lo cual el navegador interpretará que se trata de otra página.

Recordemos que el tercer parámetro de la función open indica que el proceso se efectúe en forma asincrónica (esto si le pasamos el valor true)

La función procesarEventos obtiene una referencia al div llamado detalles y muestra inicialmente el texto: 'Cargando...', luego cuando el servidor informa que los datos se registraron pasa a mostrar el texto: 'Gracias'.

```
function procesarEventos()
{
    var resultados = document.getElementById("resultados");
    if(conexion1.readyState == 4)
    {
        resultados.innerHTML = 'Gracias.';
    }
    else
    {
        resultados.innerHTML = 'Procesando...';
    }
}
```

Por último la página que se ejecuta en el servidor (pagina1.php) tiene por objetivo registrar en un archivo de texto el valor seleccionado por el visitante:

```
<?php
header('Content-Type: text/html; charset=ISO-8859-1');
$ar=fopen("votos.txt","a") or
    die("No se pudo abrir el archivo");
fputs($ar,$_REQUEST['voto']."-\\n");
fclose($ar);
?>
```

7 - Pasando datos al servidor por el método POST.

Podemos enviar los datos por el método GET, como hemos visto hasta ahora, pero también podemos enviar los datos por el método POST.

El método POST se utiliza cuando hay que enviar mucha información al servidor.

Hay varios puntos a tener en cuenta para cuando codificamos los datos para el envío por el método POST:

1. Cuando llamamos al método open del objeto XMLHttpRequest como primer parámetro indicamos el string 'POST'
2. `conexion1.open('POST','pagina1.php', true);`
3. Llamamos al método setRequestHeader indicando que los datos a enviarse están codificados como un formulario.
4. `conexion1.setRequestHeader("Content-Type","application/x-www-form-urlencoded");`
5. Llamamos al método send del objeto XMLHttpRequest pasando los datos:
6. `conexion1.send("nombre=juan&clave=z80");`

Podemos concatenar datos extraídos de un formulario y enviarlos a través del método send.

Confeccionaremos un problema completo para probar la funcionalidad del envío de datos por el método POST.

Haremos una serie de páginas que nos permitan enviar los datos cargados en un formulario al servidor en forma asincrónica y por lo tanto no tendremos que refrescar la página luego de enviado los datos. El formulario solicitará que ingrese el nombre y comentarios del sitio.

El archivo HTML es el siguiente (pagina1.html):

```
<html>
<head>
<title>Problema</title>
<script src="funciones.js" language="JavaScript"></script>
</head>
<body>
<form action="pagina1.php" method="post" id="formulario">
Nombre: <input type="text" name="nombre" id="nombre"
size="20"><br>
Comentarios: <br>
<textarea name="comentarios" id="comentarios" rows="10"
cols="50"></textarea><br>
```

```

<input type="submit" value="Enviar" id="enviar">
<div id="resultados"></div>
<a href="comentarios.txt">Ver resultados</a>
</form>
</body>
</html>

```

Este archivo HTML no tiene nada de especial, básicamente es un formulario con un control de tipo input, otro de tipo textarea y finalmente un botón para el envío de los datos al servidor. Además contiene un div vacío para mostrar mensajes sobre el resultado del envío de los datos al servidor.

Disponemos un hipervínculo al archivo de texto que almacenará los datos cargados, esto con el objetivo de poder controlar si los datos realmente se cargaron en el servidor. En un problema más grande seguramente los datos los almacenaremos en una base de datos.

El archivo funciones.js es:

```

addEvent(window,'load',inicializarEventos,false);

function inicializarEventos()
{
    var ref=document.getElementById('formulario');
    addEvent(ref,'submit',enviarDatos,false);
}

function enviarDatos(e)
{
    if (window.event)
        window.event.returnValue=false;
    else
        if (e)
            e.preventDefault();
    enviarFormulario();
}

function retornarDatos()
{
    {
        var cad="";
        var nom=document.getElementById('nombre').value;
        var com=document.getElementById('comentarios').value;
        cad='nombre='+encodeURIComponent(nom)
        + '&comentarios='+encodeURIComponent(com);
        return cad;
    }
}

var conexion1;
function enviarFormulario()
{
    {
        conexion1=crearXMLHttpRequest();
        conexion1.onreadystatechange = procesarEventos;
        conexion1.open('POST','pagina1.php', true);
        conexion1.setRequestHeader("Content-Type","application/x-www-form-
        urlencoded");
    }
}

```

```

    conexion1.send(retornarDatos());
}

function procesarEventos()
{
    var resultados = document.getElementById("resultados");
    if(conexion1.readyState == 4)
    {
        resultados.innerHTML = 'Gracias.';
    }
    else
    {
        resultados.innerHTML = 'Procesando...';
    }
}

//*****
//Funciones comunes a todos los problemas
//*****
function addEvent(elemento,nomevento,funcion,captura)
{
    if (elemento.attachEvent)
    {
        elemento.attachEvent('on'+nomevento,funcion);
        return true;
    }
    else
    if (elemento.addEventListener)
    {
        elemento.addEventListener(nomeevento,funcion,captura);
        return true;
    }
    else
        return false;
}

function crearXMLHttpRequest()
{
    var xmlhttp=null;
    if (window.ActiveXObject)
        xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
    else
        if (window.XMLHttpRequest)
            xmlhttp = new XMLHttpRequest();
    return xmlhttp;
}

```

Lo primero que hacemos en la función inicializarEventos es:

```

var ref=document.getElementById('formulario');
addEvent(ref,'submit',enviarDatos,false);

```

Obtener la referencia del formulario e inicializar el evento submit para poder capturar el momento en que los datos se enviarán al servidor. Es decir la función enviarDatos se ejecutará cuando el operador presione el botón de tipo submit.

La función enviarFormulario:

```
var conexion1;  
function enviarFormulario()  
{  
    conexion1=crearXMLHttpRequest()  
    conexion1.onreadystatechange = procesarEventos;  
    conexion1.open('POST','pagina1.php', true);  
    conexion1.setRequestHeader("Content-Type","application/x-www-form-  
urlencoded");  
    conexion1.send(retornarDatos());  
}
```

Inicializa el objeto XMLHttpRequest indicando en el método open que los datos se enviarán mediante el comando POST.

Luego llamamos al método setRequestHeader indicando el tipo de contenido a enviar al servidor.

Finalmente llamamos al método send con los datos. Estos datos los recuperamos del formulario llamando a la función retornarDatos() que a continuación la explicaremos.

```
function retornarDatos()  
{  
    var cad='';  
    var nom=document.getElementById('nombre').value;  
    var com=document.getElementById('comentarios').value;  
    cad='nombre='+encodeURIComponent(nom)  
+ '&comentarios='+encodeURIComponent(com);  
    return cad;  
}
```

Obtenemos el valor contenido en cada control (input y textarea) y luego concatenamos nombre del dato y valor separándolos por el carácter ampersand.

Otra cosa importante es llamar a la función JavaScript encodeURIComponent para codificar los datos ingresados por el visitante y se puedan enviar correctamente al servidor.

La función procesarEventos:

```
function procesarEventos()  
{  
    var resultados = document.getElementById("resultados");  
    if(conexion1.readyState == 4)  
    {  
        resultados.innerHTML = 'Gracias.';  
    }  
    else  
    {  
        resultados.innerHTML = 'Procesando...';  
    }  
}
```

Dijimos ya que esta función se ejecuta cada vez que cambia el estado del objeto XMLHttpRequest. Cuando retorna un 4 significa que el envío de datos se efectuó en forma correcta, mostramos en ese momento el mensaje 'Gracias'.

El último archivo: pagina1.php

```
<?php
header('Content-Type: text/html; charset=ISO-8859-1');
$ar=fopen("comentarios.txt","a") or
    die("No se pudo abrir el archivo");
fputs($ar,"Nombre:".$_REQUEST['nombre']."\n");
fputs($ar,"Comentarios:".$_REQUEST['comentarios']."\n\n");
fclose($ar);
?>
```

8 - Recuperando datos mediante la propiedad responseText del objeto XMLHttpRequest

Ahora nos concentraremos en la propiedad responseText del objeto XMLHttpRequest. Esta propiedad almacena el valor devuelto por el servidor.

Normalmente accederemos a la propiedad responseText cuando el objeto XMLHttpRequest nos informa que toda la información fue remitida por el servidor, esto ocurre cuando la propiedad readyState del objeto XMLHttpRequest almacena el valor 4.

Lo más común es que tengamos un código similar al siguiente:

```
function procesarEventos()
{
    var detalles = document.getElementById("comentarios");
    if(conexion1.readyState == 4)
    {
        detalles.innerHTML = conexion1.responseText;
    }
    else
    {
        detalles.innerHTML = 'Cargando...';
    }
}
```

Confeccionaremos un ejemplo para identificar donde utilizar la propiedad responseText. El problema consiste en mostrar una lista de hipervínculos que representan los comentarios de distintas fecha. El objetivo es rescatar todos los comentarios para la fecha seleccionada por el visitante y su posterior visualización sin tener que recargar nuevamente la página.

El archivo pagina1.html es:

```
<html>
<head>
<title>Problema</title>
<script src="funciones.js" language="JavaScript"></script>
<link rel="StyleSheet" href="estilos.css" type="text/css">
</head>
<body>
<h2>Seleccione la fecha:</h2>
<p>
<div id="fecha">
<a href="pagina1.php?fecha=10/03/2007">ver comentarios del
10/03/2007</a><br>
```

```

<a href="pagina1.php?fecha=11/03/2007">ver comentarios del
11/03/2007</a><br>
<a href="pagina1.php?fecha=12/03/2007">ver comentarios del
12/03/2007</a><br>
</div>
<div class="recuadro" id="comentarios">Comentarios:</div>
</body>

```

Cada hipervínculo dispone como parámetro la fecha de la cual queremos recuperar los comentarios.

Disponemos un div donde visualizaremos los comentarios de la fecha seleccionada.

La hoja de estilo estilos.css es:

```

.recuadro {
  background-color: #ffffcc;
  text-align:left;
  font-family:verdana;
  border-width:0;
  padding:5px;
  border: 1px dotted #ffaa00;
}

```

Luego el archivo funciones.js es:

```

addEvent(window,'load',inicializarEventos,false);

function inicializarEventos()
{
  var ref;
  ref=document.getElementById('fecha');
  var vec=ref.getElementsByTagName('a');
  for(f=0;f<vec.length;f++)
  {
    addEvent(vec[f],'click',presionEnlace,false);
  }
}

function presionEnlace(e)
{
  if (window.event)
  {
    window.event.returnValue=false;
    var url=window.event.srcElement.getAttribute('href');
    verComentarios(url);
  }
  else
  if (e)
  {
    e.preventDefault();
    var url=e.target.getAttribute('href');
    verComentarios(url);
  }
}

```

```

}

var conexion1;
function verComentarios(url)
{
    if(url=="")
    {
        return;
    }
    conexion1=crearXMLHttpRequest();
    conexion1.onreadystatechange = procesarEventos;
    conexion1.open("GET", url, true);
    conexion1.send(null);
}

function procesarEventos()
{
    var detalles = document.getElementById("comentarios");
    if(conexion1.readyState == 4)
    {
        detalles.innerHTML = conexion1.responseText;
    }
    else
    {
        detalles.innerHTML = 'Cargando...';
    }
}

//*****
//Funciones comunes a todos los problemas
//*****
function addEvent(elemento,nomevento,funcion,captura)
{
    if (elemento.attachEvent)
    {
        elemento.attachEvent('on'+nomevento,funcion);
        return true;
    }
    else
    if (elemento.addEventListener)
    {
        elemento.addEventListener(nomeevento,funcion,captura);
        return true;
    }
    else
        return false;
}

function crearXMLHttpRequest()
{
    var xmlHttp=null;
    if (window.ActiveXObject)
        xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
    else
        if (window.XMLHttpRequest)
            xmlHttp = new XMLHttpRequest();
}

```

```
return xmlhttp;  
}
```

La función inicializarEventos:

```
var ref;  
ref=document.getElementById('fecha');  
var vec=ref.getElementsByTagName('a');  
for(f=0;f<vec.length;f++)  
{  
    addEvent(vec[f], 'click', presionEnlace, false);  
}
```

Obtiene la referencia a todas las a (anchor) contenidas en el div 'fecha', luego mediante un for inicializa el evento click para cada hipervínculo.

La función presionEnlace se dispara cuando se presiona alguno de los hipervínculos:

```
function presionEnlace(e)  
{  
    if (window.event)  
    {  
        window.event.returnValue=false;  
        var url=window.event.srcElement.getAttribute('href');  
        verComentarios(url);  
    }  
    else  
        if (e)  
        {  
            e.preventDefault();  
            var url=e.target.getAttribute('href');  
            verComentarios(url);  
        }  
}
```

Primero desactivamos el evento por defecto y llamamos a la función verComentarios pasando como referencia el parámetro del hipervínculo.

La función verComentarios :

```
var conexion1;  
function verComentarios(url)  
{  
    if(url=='')  
    {  
        return;  
    }  
    conexion1=crearXMLHttpRequest()  
    conexion1.onreadystatechange = procesarEventos;  
    conexion1.open("GET", url, true);  
    conexion1.send(null);  
}
```

Llama a la función crearXMLHttpRequest que retorna un objeto de tipo XMLHttpRequest y procede a almacenarlo en la variable conexion1 (que al ser una variable global podrá utilizarse y accederse desde otra función)

Las siguientes tres líneas son similares a problemas que hemos realizado anteriormente. Primero inicializamos la propiedad `onreadystatechange` con el nombre de la función que procesará los eventos del objeto `XMLHttpRequest`.

Como segundo paso llamamos al método `open` indicando que enviaremos la fecha mediante el método `GET`, el siguiente parámetro el nombre de la página a llamar junto con la fecha a enviar y por último pasamos el valor `true` indicando que emplearemos comunicación asincrónica.

El tercer paso es llamar al método `send`. Como parámetro a este método indicamos el valor `null` ya que los datos viajan en la misma cabecera de petición de página (ej. 'pagina1.php? fecha=22/11/2007')

La función `procesarEventos` es donde accedemos a la propiedad `responseText`:

```
function procesarEventos()
{
    var detalles = document.getElementById("comentarios");
    if(conexion1.readyState == 4)
    {
        detalles.innerHTML = conexion1.responseText;
    }
    else
    {
        detalles.innerHTML = 'Cargando...';
    }
}
```

En esta función mientras la propiedad `readyState` del objeto `XMLHttpRequest` almacene un valor distinto a 4 significa que el proceso no a finalizado. Cuando identificamos que tiene el valor 4 procedemos a modificar el `div` con el valor devuelto por el servidor:

```
detalles.innerHTML = conexion1.responseText;
```

Para simplificar el problema en el servidor evitamos rescatarlos de una base de datos (como ocurre en la realidad), luego el contenido de esta página es (pagina1.php):

[illegible]

[illegible]

9 - Recuperando datos mediante la propiedad responseXML del objeto XMLHttpRequest

La propiedad `responseXML` a diferencia de la propiedad `responseText` recupera los datos como XML y debemos recorrerlo mediante las funciones del DOM.

Es necesario que el programa que se ejecute en el servidor estructure los datos en formato XML.

Para probar el funcionamiento de esta propiedad del objeto XMLHttpRequest implementaremos una página que muestre en un control select los nombres de una serie de países. Cuando se seleccione uno y se presione un botón recuperaremos de dicho país el nombre de la capital, su superficie, cantidad de habitantes y su idioma.

El servidor generará un archivo con formato XML y lo retornará al cliente.

El primer archivo (pagina1.html):

```
<html>
<head>
<title>Problema</title>
<script src="funciones.js" language="JavaScript"></script>
</head>
<body>
<h1>Datos de paises.</h1>
<p>Seleccione el pais
<select id="pais">
<option value="0" selected>seleccione</option>
<option value="Argentina">Argentina</option>
<option value="Brasil">Brasil</option>
<option value="Chile">Chile</option>
</select><br>
<input type="button" id="boton1" value="Recuperar">
</p>
<div id="resultados"></div>
</body>
</html>
```

Este archivo HTML no tiene nada nuevo.

El archivo que contiene las funciones JavaScript (funciones.js):

```
addEvent(window,'load',inicializarEventos,false);

function inicializarEventos()
{
    var ob=document.getElementById('boton1');
    addEvent(ob,'click',presionBoton,false);
}

function presionBoton(e)
{
    var ob1=document.getElementById('pais');
    recuperarDatos(ob1.value);
}

var conexion1;
function recuperarDatos(pais)
{
    conexion1=crearXMLHttpRequest();
    conexion1.onreadystatechange = procesarEventos;
    conexion1.open('GET','pagina1.php?pa='+pais, true);
    conexion1.send(null);
}

function procesarEventos()
{
    var resultados = document.getElementById("resultados");
    if(conexion1.readyState == 4)
    {
        var xml = conexion1.responseXML;
```

```

var capital=xml.getElementsByTagName('capital');
var superficie=xml.getElementsByTagName('superficie');
var idioma=xml.getElementsByTagName('idioma');
var poblacion=xml.getElementsByTagName('poblacion');
resultados.innerHTML='Capital='+capital[0].firstChild.nodeValue + '<br>' +
    'Superficie='+superficie[0].firstChild.nodeValue + '<br>' +
    'Idioma='+idioma[0].firstChild.nodeValue + '<br>' +
    'Poblacion='+poblacion[0].firstChild.nodeValue ;

}
else
{
    resultados.innerHTML = 'Cargando...';
}
}

//*****
//Funciones comunes a todos los problemas
//*****
function addEvent(elemento,nomevento,funcion,captura)
{
    if (elemento.attachEvent)
    {
        elemento.attachEvent('on'+nomevento,funcion);
        return true;
    }
    else
    if (elemento.addEventListener)
    {
        elemento.addEventListener(nomeevento,funcion,captura);
        return true;
    }
    else
        return false;
}

function crearXMLHttpRequest()
{
    var xmlhttp=null;
    if (window.ActiveXObject)
        xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
    else
        if (window.XMLHttpRequest)
            xmlhttp = new XMLHttpRequest();
    return xmlhttp;
}

```

En la primer función que se ejecuta enlazamos la función a ejecutar al presionar el botón:

```

var ob=document.getElementById('boton1');
addEvent(ob, 'click',presionBoton,false);

```

Cuando se presiona el botón se ejecuta la función presionBoton:

```

function presionBoton(e)
{
    var obl=document.getElementById('pais');

```

```

    recuperarDatos (obl.value);
}

```

Recuperamos el país seleccionado en el control select y llamamos a la función recuperarDatos pasando como parámetro dicho país.

La función recuperarDatos crea un objeto de la clase XMLHttpRequest y abre una conexión con el servidor mediante el método GET pasando como parámetro el nombre del país seleccionado:

```

var conexion1;
function recuperarDatos(pais)
{
    conexion1=crearXMLHttpRequest()
    conexion1.onreadystatechange = procesarEventos;
    conexion1.open('GET','paginal.php?pa='+pais, true);
    conexion1.send(null);
}

```

Como sabemos la propiedad onreadystatechange se inicializa con el nombre de la función que procesará los cambios de estado del objeto XMLHttpRequest. En este caso el algoritmo de la función es:

```

function procesarEventos()
{
    var resultados = document.getElementById("resultados");
    if(conexion1.readyState == 4)
    {
        var xml = conexion1.responseXML;
        var capital=xml.getElementsByTagName('capital');
        var superficie=xml.getElementsByTagName('superficie');
        var idioma=xml.getElementsByTagName('idioma');
        var poblacion=xml.getElementsByTagName('poblacion');
        resultados.innerHTML='Capital='+capital[0].firstChild.nodeValue +
'<br>' +
                                'Superficie='+superficie[0].firstChild.nodeValue + '
'<br>' +
                                'Idioma='+idioma[0].firstChild.nodeValue +
'<br>' +
                                'Poblacion='+poblacion[0].firstChild.nodeValue ;
    }
    else
    {
        resultados.innerHTML = 'Cargando...';
    }
}

```

Cuando los datos han llegado completamente, es decir la propiedad readyState del objeto XMLHttpRequest almacena el valor 4 procedemos a cargar los datos dentro del div 'resultados'.

Primero almacenamos la información en la variable xml:

```

var xml = conexion1.responseXML;

```

Ahora podemos recorrer el archivo XML mediante las funciones que provee el DOM.

```

var capital=xml.getElementsByTagName('capital');

```

Esta línea genera un vector con las referencias a todos los elementos llamados 'capital' (En nuestro caso hay uno solo)

Como se trata de un vector luego debemos accederlo por medio de un subíndice (indicamos el valor cero ya que contiene solo un elemento):

```
capital[0].firstChild.nodeValue
```

Como sabemos el texto contenido entre las marcas se trata de otro nodo y lo podemos acceder mediante el método firstChild. Por último para acceder a la información de dicho nodo de texto lo hacemos mediante la propiedad nodeValue (Tener en cuenta que todos estos conceptos se ven detenidamente en el tutorial de [DHTML](#))

El último archivo que integra este problema es el que se ejecuta en el servidor (pagina1.php):

```
<?php
if ($_REQUEST['pa']=='Argentina')
{
    $superficie=2700000;
    $capital="Buenos Aires";
    $idioma="Castellano";
    $poblacion=38000000;
}
if ($_REQUEST['pa']=='Brasil')
{
    $superficie=8500000;
    $capital="Brasilia";
    $idioma="Portugues";
    $poblacion=163000000;
}
if ($_REQUEST['pa']=='Chile')
{
    $superficie=750000;
    $capital="Santiago";
    $idioma="Castellano";
    $poblacion=15000000;
}

$xml="<?xml version=\"1.0\"?>\n";
$xml="<pais>\n";
$xml="<superficie>$superficie</superficie>\n";
$xml="<capital>$capital</capital>\n";
$xml="<idioma>$idioma</idioma>\n";
$xml="<poblacion>$poblacion</poblacion>\n";
$xml="</pais>\n";
header('Content-Type: text/xml');
echo $xml;
?>
```

Nuevamente para simplificar el problema no hemos almacenado los datos de los países en una base de datos. Mediante una serie de if verificamos de qué país se trata e inicializamos cuatro variables.

Por último creamos un string organizando la información con formato XML.

Mediante la función header informamos al navegador que se trata de un archivo XML.

Finalmente procedemos a imprimir la variable \$xml luego de enviar la cabecera al navegador.

10 - Propiedades onreadystatechange y readyState.

Dos propiedades fundamentales del objeto XMLHttpRequest son onreadystatechange y readyState.

El objetivo de cada una es:

- *onreadystatechange* Almacena el nombre de la función que se ejecutará cuando el objeto XMLHttpRequest cambie de estado.
- *readyState* Almacena el estado del requerimiento hecho al servidor, pudiendo ser:
 - 0 No inicializado (el método open no a sido llamado)
 - 1 Cargando (se llamó al método open)
 - 2 Cargado (se llamó al método send y ya tenemos la cabecera de la petición HTTP y el status)
 - 3 Interactivo (la propiedad responseText tiene datos parciales)
 - 4 Completado (la propiedad responseText tiene todos los datos pedidos al servidor)

Normalmente cuando creamos un objeto de la clase XMLHttpRequest inicializamos la propiedad onreadystatechange con el nombre de la función que procesará los datos enviados por el servidor:

```
function cargarHoroscopo(url)
{
    if(url=='')
    {
        return;
    }
    conexion1=crearXMLHttpRequest();
    conexion1.onreadystatechange = procesarEventos;
    conexion1.open("GET", url, true);
    conexion1.send(null);
}
```

Por otro lado dentro de la función que previamente fue asignada a la propiedad onreadystatechange verificamos el estado de la propiedad readyState:

```
function procesarEventos()
{
    var detalles = document.getElementById("detalles");
    if(conexion1.readyState == 4)
    {
        detalles.innerHTML = conexion1.responseText;
    }
    else
    {
        detalles.innerHTML = 'Cargando...';
    }
}
```

Para ver el paso de estados del objeto XMLHttpRequest implementaremos una aplicación que calcule el cuadrado de un número que ingresamos por teclado. Y además mostraremos mediante un alert el estado actual de la propiedad readyState.

pagina1.html>

```
<html>
<head>
<title>Problema</title>
<script src="funciones.js" language="JavaScript"></script>
</head>
<body>
<form action="pagina1.php" method="post" id="formulario">
Ingrese nro: <input type="text" name="nro" id="nro"
size="10"><br>
<input type="submit" value="Calcular el cuadrado" id="enviar">
<div id="resultados"></div>
</form>
</body>
</html>
```

funciones.js

```
addEvent(window,'load',inicializarEventos,false);

function inicializarEventos()
{
    var ref=document.getElementById('formulario');
    addEvent(ref,'submit',enviarDatos,false);
}

function enviarDatos(e)
{
    {
        if (window.event)
            window.event.returnValue=false;
        else
            if (e)
                e.preventDefault();
        enviarFormulario();
    }
}

var conexion1;
function enviarFormulario()
{
    {
        conexion1=crearXMLHttpRequest()
        conexion1.onreadystatechange = procesarEventos;
        var num=document.getElementById('nro').value;
        alert('Valor de la propiedad readyState:'+conexion1.readyState);
        conexion1.open('GET','pagina1.php?numero='+num, true);
        conexion1.send(null);
    }
}

function procesarEventos()
{
    {
        alert('Valor de la propiedad readyState:'+conexion1.readyState);
    }
}
```

```

var resultados = document.getElementById("resultados");
if(conexion1.readyState == 4)
{
    resultados.innerHTML = conexion1.responseText;
}
else
    if (conexion1.readyState==1 || conexion1.readyState==2 ||
conexion1.readyState==3)
    {
        resultados.innerHTML = 'Procesando...';
    }
}

//*****
//Funciones comunes a todos los problemas
//*****
function addEvent(elemento,nomevento,funcion,captura)
{
    if (elemento.attachEvent)
    {
        elemento.attachEvent('on'+nomevento,funcion);
        return true;
    }
    else
        if (elemento.addEventListener)
        {
            elemento.addEventListener(nomeevento,funcion,captura);
            return true;
        }
        else
            return false;
}

function crearXMLHttpRequest()
{
    var xmlhttp=null;
    if (window.ActiveXObject)
        xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
    else
        if (window.XMLHttpRequest)
            xmlhttp = new XMLHttpRequest();
    return xmlhttp;
}

```

Como podemos observar si accedemos a la propiedad readyState antes de llamar a los métodos open y send la misma almacena el valor cero:

```

conexion1=crearXMLHttpRequest()
conexion1.onreadystatechange = procesarEventos;
var num=document.getElementById('nro').value;
alert('Valor de la propiedad
readyState:'+conexion1.readyState); //Muestra cero

```

Cuando llamamos al método open se ejecuta por primera vez la función que inicializamos en la propiedad onreadystatechange, al mostrar la propiedad readyState veremos que almacena un uno:

```
conexion1.open('GET','pagina1.php?numero='+num, true);
```

Cuando llamamos al método send se ejecuta por segunda vez la función que inicializamos en la propiedad onreadystatechange, al mostrar la propiedad readyState veremos que continua con el valor uno.

Luego la función:

```
function procesarEventos()
{
    alert('Valor de la propiedad
readyState:'+conexion1.readyState);
    var resultados = document.getElementById("resultados");
    if(conexion1.readyState == 4)
    {
        resultados.innerHTML = conexion1.responseText;
    }
    else
        if (conexion1.readyState==1 || conexion1.readyState==2 ||
conexion1.readyState==3)
        {
            resultados.innerHTML = 'Procesando...';
        }
}
```

mostrará los valores 2,3 y 4.

Recordemos:

- 0 No inicializado (el método open no a sido llamado)
- 1 Cargando (se llamó al método open)
- 2 Cargado (se llamó al método send y ya tenemos la cabecera de la petición HTTP y el status)
- 3 Interactivo (la propiedad.responseText tiene datos parciales)
- 4 Completado (la propiedad.responseText tiene todos los datos pedidos al servidor)

pagina1.php

```
<?php
$cuadrado=$_REQUEST['numero']*$_REQUEST['numero'];
echo $cuadrado;
?>
```

11 - Propiedades status y statusText.

Estas dos propiedades todavía no las habíamos utilizado. Veremos que las mismas nos permiten hacer un código más correcto.

- **status** Esta propiedad almacena el código del estado de la petición HTTP. Entre otros valores que puede retornar el servidor: 200 es el valor para una conexión exitosa, 404

página inexistente. Esta propiedad de solo lectura está disponible cuando readyState toma los valores 3 o 4.

- **statusText** Almacena el texto de la petición HTTP enviado por el servidor.

Con estas nuevas propiedades veremos que lo más correcto cuando readyState contiene el valor 4 debemos además verificar si la propiedad status almacena el valor 200 (es decir el servidor procesó correctamente la petición)

Para probar estas propiedades confeccionaremos el problema del concepto anterior (retornar del servidor el cuadrado de un número que ingresamos por teclado), introduciremos un error (el método open llamará a una página php inexistente) y veremos cual es el mensaje en pantalla, luego agregaremos la verificación de la propiedad status.

pagina1.html

```
<html>
<head>
<title>Problema</title>
<script src="funciones.js" language="JavaScript"></script>
</head>
<body>
<form action="pagina1.php" method="post" id="formulario">
Ingrese nro:<input type="text" name="nro" id="nro"
size="10"><br>
<input type="submit" value="Calcular el cuadrado" id="enviar">
<div id="resultados"></div>
</form>
</body>
</html>
```

funciones.js

```
addEvent(window,'load',inicializarEventos,false);

function inicializarEventos()
{
    var ref=document.getElementById('formulario');
    addEvent(ref,'submit',enviarDatos,false);
}

function enviarDatos(e)
{
    {
        if (window.event)
            window.event.returnValue=false;
        else
            if (e)
                e.preventDefault();
        enviarFormulario();
    }
}

var conexion1;
function enviarFormulario()
{
    {
        conexion1=crearXMLHttpRequest();
        conexion1.onreadystatechange = procesarEventos;
```

```

var num=document.getElementById('nro').value;
conexion1.open('GET','paginax.php?numero='+num, true);
conexion1.send(null);
}

function procesarEventos()
{
    var resultados = document.getElementById("resultados");
    if(conexion1.readyState == 4)
    {
        resultados.innerHTML = conexion1.responseText;
    }
    else
        if (conexion1.readyState==1 || conexion1.readyState==2 ||
conexion1.readyState==3)
        {
            resultados.innerHTML = 'Procesando...';
        }
}

//*****
//Funciones comunes a todos los problemas
//*****
function addEvent(elemento,nomevento,funcion,captura)
{
    if (elemento.attachEvent)
    {
        elemento.attachEvent('on'+nomevento,funcion);
        return true;
    }
    else
        if (elemento.addEventListener)
        {
            elemento.addEventListener(nomeevento,funcion,captura);
            return true;
        }
        else
            return false;
}

function crearXMLHttpRequest()
{
    var xmlhttp=null;
    if (window.ActiveXObject)
        xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
    else
        if (window.XMLHttpRequest)
            xmlhttp = new XMLHttpRequest();
    return xmlhttp;
}

```

En este archivo podemos ver el error que hemos introducido:

```

conexion1.open('GET','paginax.php?numero='+num, true);

```

Hemos pasado como nombre de página: paginax.php donde debíamos disponer pagina1.php.

pagina1.php

```
<?php
$cuadrado=$_REQUEST['numero']*$_REQUEST['numero'];
echo $cuadrado;
?>
```

Cuando ejecute estas páginas deberá ver un mensaje parecido a: "Page Not Found".

Luego modifique el archivo funciones.js verificando el estado de la propiedad status:

```
function procesarEventos()
{
    var resultados = document.getElementById("resultados");
    if(conexion1.readyState == 4)
    {
        if (conexion1.status==200)
        {
            resultados.innerHTML = conexion1.responseText;
        }
        else
        {
            resultados.innerHTML='';
            alert(conexion1.statusText);
        }
    }
    else
    {
        if (conexion1.readyState==1 || conexion1.readyState==2 ||
conexion1.readyState==3)
        {
            resultados.innerHTML = 'Procesando...';
        }
    }
}
```

Es decir que no solo debemos verificar que la propiedad readyState tenga almacenado un 4, sino que la propiedad status retorne un 200. En este caso por el else solamente borramos el div donde aparece el texto 'Procesando...' y mostramos en un alert el texto del error devuelto por el servidor .

12 - Método abort del objeto XMLHttpRequest

El objeto XMLHttpRequest tiene un método llamado abort que tiene por objetivo detener la conexión establecida.

Hay situaciones donde el servidor se encuentra saturado y no puede devolver una petición. En estas situaciones es bueno mostrar un mensaje al usuario del sitio que indique el problema, además mediante este método (abort) cancelamos la petición.

Confeccionaremos el primer problema planteado en este tutorial:

Confeccionar un problema que muestre una lista de hipervínculos con los distintos signos del horóscopo y luego al ser presionado no recargue la página completa sino que se envíe una petición al servidor y el mismo retorne la información de dicho signo, luego se actualice solo el contenido de un div del archivo HTML.

Le agregaremos que si en tres segundos el estado de petición no a finalizado mostraremos un mensaje al usuario indicando que el servidor se encuentra saturado.

Para simular el efecto de saturación del servidor utilizaremos en el programa PHP la llamada a la función sleep que tiene por objetivo detener una determinada cantidad de segundos la ejecución del programa PHP.

Nuestra página HTML es (pagina1.html):

```
<html>
<head>
<title>Problema</title>
<script src="funciones.js" language="JavaScript"></script>
<link rel="StyleSheet" href="estilos.css" type="text/css">
</head>
<body>
<h1>Signos del horóscopo.</h1>
<div id="menu">
<p><a id="enlace1" href="pagina1.php?cod=1">Aries</a></p>
<p><a id="enlace2" href="pagina1.php?cod=2">Tauro</a></p>
<p><a id="enlace3" href="pagina1.php?cod=3">Geminis</a></p>
<p><a id="enlace4" href="pagina1.php?cod=4">Cancer</a></p>
<p><a id="enlace5" href="pagina1.php?cod=5">Leo</a></p>
<p><a id="enlace6" href="pagina1.php?cod=6">Virgo</a></p>
<p><a id="enlace7" href="pagina1.php?cod=7">Libra</a></p>
<p><a id="enlace8" href="pagina1.php?cod=8">Escorpio</a></p>
<p><a id="enlace9" href="pagina1.php?cod=9">Sagitario</a></p>
<p><a id="enlace10" href="pagina1.php?cod=10">Capricornio</a></p>
<p><a id="enlace11" href="pagina1.php?cod=11">Acuario</a></p>
<p><a id="enlace12" href="pagina1.php?cod=12">Piscis</a></p>
</div>
<div id="detalles">Seleccione su signo.</div>
</body>
</html>
```

El archivo estilos.css:

```
#menu {
  font-family: Arial;
  margin:5px;
}

#menu p {
  margin:0px;
```

```

padding:0px;
}

#menu a {
display: block;
padding: 3px;
width: 160px;
background-color: #f7f8e8;
border-bottom: 1px solid #eee;
text-align:center;
}

#menu a:link, #menu a:visited {
color: #f00;
text-decoration: none;
}

#menu a:hover {
background-color: #369;
color: #fff;
}

#detalles {
background-color: #ffc;
text-align:left;
font-family:verdana;
border-width:0;
padding:5px;
border: 1px dotted #fa0;
margin:5px;
}

```

El archivo funciones.js

```

addEvent(window,'load',inicializarEventos,false);

function inicializarEventos()
{
var ob;
for(f=1;f<=12;f++)
{
ob=document.getElementById('enlace'+f);
addEvent(ob,'click',presionEnlace,false);
}
}

function presionEnlace(e)
{
if (window.event)
{
window.event.returnValue=false;
var url=window.event.srcElement.getAttribute('href');
cargarHoroscopo(url);
}
else

```

```

    if (e)
    {
        e.preventDefault();
        var url=e.target.getAttribute('href');
        cargarHoroscopo(url);
    }
}

var conexion1;
var tiempo;
function cargarHoroscopo(url)
{
    if(url=="")
    {
        return;
    }
    conexion1=crearXMLHttpRequest();
    conexion1.onreadystatechange = procesarEventos;
    conexion1.open("GET", url, true);
    conexion1.send(null);
    tiempo=setTimeout("finDeEspera()",3000);
}

function procesarEventos()
{
    var detalles = document.getElementById("detalles");
    if(conexion1.readyState == 4)
    {
        clearTimeout(tiempo);
        detalles.innerHTML = conexion1.responseText;
    }
    else
    if(conexion1.readyState == 1)
    {
        detalles.innerHTML = 'Cargando...';
    }
}

function finDeEspera()
{
    conexion1.abort();
    detalles.innerHTML = 'Intente nuevamente más tarde, el servidor esta sobrecargado.';
}
//*****
//Funciones comunes a todos los problemas
//*****
function addEvent(elemento,nomevento,funcion,captura)
{
    if (elemento.attachEvent)
    {
        elemento.attachEvent('on'+nomevento,funcion);
        return true;
    }
    else
    if (elemento.addEventListener)

```

```

    {
        elemento.addEventListener(nomevento,funcion,captura);
        return true;
    }
    else
        return false;
}

function crearXMLHttpRequest()
{
    var xmlHttp=null;
    if (window.ActiveXObject)
        xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
    else
        if (window.XMLHttpRequest)
            xmlHttp = new XMLHttpRequest();
    return xmlHttp;
}

```

En este archivo tenemos que cuando comenzamos la conexión con el servidor inicializamos un temporizador para que si pasan más de 3 segundos sin responder el servidor proceda a abortar esa conexión e informe al visitante.

```
tiempo=setTimeout("finDeEspera()",3000);
```

La variable tiempo guarda una referencia al temporizador, con el objetivo de poderlo detener si la respuesta demora menos de 3 segundos.

```

if(conexion1.readyState == 4)
{
    clearTimeout(tiempo);
    detalles.innerHTML = conexion1.responseText;
}

```

Aca podemos ver que si la petición finaliza procedemos a desabilitar el temporizador llamando a la función clearTimeout.

```

function finDeEspera()
{
    conexion1.abort();
    detalles.innerHTML = 'Intente nuevamente más tarde, el servidor esta sobrecargado.';
}

```

La función finDeEspera se ejecutará si pasan 3 segundos sin finalizar el envío de datos del servidor. Aca es donde procedemos a llamar al método abort del objeto XMLHttpRequest. Mostramos además un mensaje al usuario del sitio.

Por último nuestro archivo pagina1.php:

```

<?php
sleep(4);
if ($_REQUEST['cod']==1)
    echo "<strong>Aries:</strong> Hoy los cambios serán físicos, personales, de carácter, Te sentirás

```

impulsivo y tomarás iniciativas. Período en donde considerarás unirte a agrupaciones de beneficencia, o de ayuda a los demás.";

```
if ($_REQUEST['cod']==2)
    echo "<strong>Tauro:</strong> Hoy los cambios serán privados, íntimos. Recuerdos. Ayuda, solidaridad. Asuntos en lugares de retiro. Tu cónyuge puede aportar buen status a tu vida o apoyo a tu profesión.";
```

```
if ($_REQUEST['cod']==3)
    echo "<strong>Géminis:</strong> Los asuntos de hoy tienen que ver con las amistades, reuniones, actividades con ellos. Día esperanzado, ilusiones. Mucha energía sexual y fuerza emocional. Deseos difíciles de controlar.";
```

```
if ($_REQUEST['cod']==4)
    echo "<strong>Cancer:</strong> Este día la profesión y las relaciones con superiores y con tu madre serán de importancia. Actividad en relación a estos temas. Momentos positivos con compañeros de trabajo. Actividad laboral agradable.";
```

```
if ($_REQUEST['cod']==5)
    echo "<strong>Leo:</strong> Este día los estudios, los viajes, el extranjero y la espiritualidad serán lo importante. Pensamientos, religión y filosofía también. Vivencias kármicas de la época te vuelven responsable tomando decisiones.";
```

```
if ($_REQUEST['cod']==6)
    echo "<strong>Virgo:</strong> Para este día toma importancia tu vida sexual, tal vez miedos, temas legales, juicios o herencias. Experiencias extrañas. Hay karma de prueba durante este período en tu parte psicológica, generándose algunos replanteos.";
```

```
if ($_REQUEST['cod']==7)
    echo "<strong>Libra:</strong> Hoy todo asunto tiene que ver con tu pareja, también con socios, con la gente o el público. Ellos serán lo más importante del día. Ganancias a través de especulaciones o del juego. Actividades vocacionales artísticas.";
```

```
if ($_REQUEST['cod']==8)
    echo "<strong>Escorpio:</strong> Hoy todo asunto tiene que ver con temas de trabajo y de salud. Presta atención a ambos. Experiencias diversas con compañeros. Durante este período tendrás muchos recursos para ganar dinero.";
```

```
if ($_REQUEST['cod']==9)
    echo "<strong>Sagitario:</strong> Durante este día se vivirán cambios en relación a los noviazgos o a los hijos. Creatividad, actividad, diversiones y salidas. Período de encuentros con personas o situaciones que te impresionan.";
```

```
if ($_REQUEST['cod']==10)
    echo "<strong>Capricornio:</strong> Los cambios del día tienen que ver con tu hogar, con la convivencia y con el padre. Asuntos relativos al carácter en la convivencia. El karma de
```

```

responsabilidad de estos momentos te acercará al mundo de lo desconocido,
mucha madurez y contacto
con el más allá.";
if ($_REQUEST['cod']==11)
    echo "<strong>Acuario:</strong> Hoy todo asunto tiene que ver con el entorno
inmediato, hermanos y
vecinos, con la comunicación, los viajes cortos o traslados frecuentes. El hablar y
trasladarse será
importante hoy. Mentalidad e ideas activas.";
if ($_REQUEST['cod']==12)
    echo "<strong>Piscis:</strong> Durante este día se vivirán cambios en la
economía, movimientos en
los ingresos, negocios, valores. Momentos de gran fuerza y decisión
profesionales, buscarás el
liderazgo.";
?>

```

Hay que tener en cuenta que para poder simular el efecto de saturación del servidor hemos llamado a la función sleep(4) para que detenga la ejecución del programa durante 4 segundos, tiempo suficiente para que el programa JavaScript del navegador aborte la conexión e informe de la situación al usuario.

Pruebe luego de quitar la función sleep y compruebe el funcionamiento correcto del sitio.

13 - Mostrar un gif animado mientras se envían y reciben los datos del servidor.

Hasta ahora mientras se actualiza la página mostramos un texto: 'Procesando...', es muy común utilizar un gif animado que represente tal operación.

Haremos una serie de páginas que nos permitan enviar los datos cargados en un formulario al servidor en forma asincrónica y mostraremos un gif animado mientras dura el envío de datos. El formulario solicitará que ingrese el nombre y sus comentarios.

pagina1.html

```

<html>
<head>
<title>Problema</title>
<script src="funciones.js" language="JavaScript"></script>
</head>
<body>
<form action="pagina1.php" method="post" id="formulario">
Nombre: <input type="text" name="nombre" id="nombre"
size="20"><br>
Comentarios: <br>
<textarea name="comentarios" id="comentarios" rows="10"
cols="50"></textarea><br>
<input type="submit" value="Enviar" id="enviar">
<span id="resultados"></span><br>
<a href="comentarios.txt">Ver resultados</a>
</form>

```

```
</body>
</html>
```

Hemos dispuesto un elemento span donde insertaremos el gif animado. Utilizamos un span para que aparezca la imagen al lado del botón submit.

funciones.js

```
addEvent(window,'load',inicializarEventos,false);

function inicializarEventos()
{
    var ref=document.getElementById('formulario');
    addEvent(ref,'submit',enviarDatos,false);
}

function enviarDatos(e)
{
    if (window.event)
        window.event.returnValue=false;
    else
        if (e)
            e.preventDefault();
    enviarFormulario();
}

function retornarDatos()
{
    var cad='';
    var nom=document.getElementById('nombre').value;
    var com=document.getElementById('comentarios').value;
    cad='nombre='+encodeURIComponent(nom)
    + '&comentarios='+encodeURIComponent(com);
    return cad;
}

var conexion1;
function enviarFormulario()
{
    conexion1=crearXMLHttpRequest();
    conexion1.onreadystatechange = procesarEventos;
    conexion1.open('POST','pagina1.php', true);
    conexion1.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
    conexion1.send(retornarDatos());
}

function procesarEventos()
{
    var resultados = document.getElementById("resultados");
    if(conexion1.readyState == 4)
    {
        if (conexion1.status==200)
            resultados.innerHTML = 'Gracias.';
    }
}
```

```

else
    alert(conexion1.statusText);
}
else
{
    resultados.innerHTML = '';
}
}

//*****
//Funciones comunes a todos los problemas
//*****
function addEvent(elemento,nomevento,funcion,captura)
{
    if (elemento.attachEvent)
    {
        elemento.attachEvent('on'+nomevento,funcion);
        return true;
    }
    else
    if (elemento.addEventListener)
    {
        elemento.addEventListener(nomeevento,funcion,captura);
        return true;
    }
    else
        return false;
}

function crearXMLHttpRequest()
{
    var xmlhttp=null;
    if (window.ActiveXObject)
        xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
    else
        if (window.XMLHttpRequest)
            xmlhttp = new XMLHttpRequest();
    return xmlhttp;
}

```

Lo nuevo en este problema es:

```
resultados.innerHTML = '';
```

Es decir insertamos una imagen dentro del elemento span. La imagen se encuentra en el directorio inmediatamente superior a donde se encuentra esta página (por eso disponemos ../ previo al nombre del archivo)

pagina1.html

```

<?php
header('Content-Type: text/html; charset=ISO-8859-1');
$ar=fopen("comentarios.txt","a") or

```

```
die("No se pudo abrir el archivo");
fputs($ar,"NHP grabado: ".$_REQUEST['nombre']."\n");
fputs($ar,"Comentarios: ".$_REQUEST['comentarios']."\n\n");
fclose($ar);
sleep(1);
?>
```

En la página PHP grabamos en un archivo de texto los datos y mediante la función sleep detenemos la ejecución del programa en el servidor una determinada cantidad de segundos (esto para poder apreciar en el navegador el gif, en la realidad no hay que llamar a sleep)

14 - Paginación con AJAX

Un lugar donde puede ayudar el uso de AJAX es en la paginación de datos mientras otro recurso en la página se está ejecutando. Confeccionaremos una página que muestre un video e inmediatamente en la parte inferior mostraremos los comentarios del video paginados.

Sin utilizar AJAX estamos obligados a recargar completamente la página lo que haría imposible ver el video y recorrer los comentarios en forma completa (considerando que solo parte de los comentarios están en la página)

Veamos y expliquemos cada uno de los archivos que intervienen.

pagina1.html

```
<html>
<head>
<title>Problema</title>
<script src="funciones.js" language="JavaScript"></script>
</head>
<body>
<object width="425" height="350"><param name="movie"
value="http://www.youtube.com/v/60og9gwKh1o"></param><param
name="wmode" value="transparent"></param><embed
src="http://www.youtube.com/v/60og9gwKh1o" type="application/x-shockwave-
flash"
wmode="transparent" width="425" height="350"></embed></object>
<br>
<div id="detalles">
</div>
</body>
</html>
```

Este archivo HTML no tiene nada de especial. Podemos decir que hemos incrustado un video del sitio youtube.

También es importante notar que hemos dispuesto un div vacío cuyo id se llama detalles. Es dentro de este div donde iremos mostrando los comentarios en forma asincrónica.

El archivo funciones.js

```

addEvent(window,'load',inicializarEventos,false);

function inicializarEventos()
{
    cargarPagina('pagina2.php');
}

function presionEnlace(e)
{
    if (window.event)
    {
        window.event.returnValue=false;
        var url=window.event.srcElement.getAttribute('href');
        cargarPagina(url);
    }
    else
    if (e)
    {
        e.preventDefault();
        var url=e.target.getAttribute('href');
        cargarPagina(url);
    }
}

var conexion1;
function cargarPagina(url)
{
    if(url=="")
    {
        return;
    }
    conexion1=crearXMLHttpRequest();
    conexion1.onreadystatechange = procesarEventos;
    conexion1.open("GET", url, true);
    conexion1.send(null);
}

function procesarEventos()
{
    {
        var detalles = document.getElementById("detalles");
        if(conexion1.readyState == 4)
        {
            detalles.innerHTML = conexion1.responseText;
            var ob1=document.getElementById('sig');
            if (ob1!=null)
                addEvent(ob1,'click',presionEnlace,false);
            var ob2=document.getElementById('ant');
            if (ob2!=null)
                addEvent(ob2,'click',presionEnlace,false);
        }
        else
        {

```

```

    detalles.innerHTML = '    ';
}
}

//*****
//Funciones comunes a todos los problemas
//*****
function addEvent(elemento,nomevento,funcion,captura)
{
    if (elemento.attachEvent)
    {
        elemento.attachEvent('on'+nomevento,funcion);
        return true;
    }
    else
    if (elemento.addEventListener)
    {
        elemento.addEventListener(nomeevento,funcion,captura);
        return true;
    }
    else
        return false;
}

function crearXMLHttpRequest()
{
    var xmlHttp=null;
    if (window.ActiveXObject)
        xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
    else
        if (window.XMLHttpRequest)
            xmlHttp = new XMLHttpRequest();
    return xmlHttp;
}

```

Como sabemos la primer función que se ejecuta al terminar de cargar la página es inicializarEventos:

```

function inicializarEventos()
{
    cargarPagina('pagina2.php');
}

```

Dentro de esta función llamamos a la función cargarPagina con el nombre del archivo php que se ejecuta en el servidor y tiene por objetivo enviarnos los comentarios.

La función cargarPagina es la que crea un objeto de la clase XMLHttpRequest:

```

var conexion1;
function cargarPagina(url)
{
    if(url=='')
    {
        return;
    }
}

```

```

conexion1=crearXMLHttpRequest();
conexion1.onreadystatechange = procesarEventos;
conexion1.open("GET", url, true);
conexion1.send(null);
}

```

La función procesarEventos:

```

function procesarEventos()
{
    var detalles = document.getElementById("detalles");
    if(conexion1.readyState == 4)
    {
        detalles.innerHTML = conexion1.responseText;
        var ob1=document.getElementById('sig');
        if (ob1!=null)
            addEvent(ob1,'click',presionEnlace,false);
        var ob2=document.getElementById('ant');
        if (ob2!=null)
            addEvent(ob2,'click',presionEnlace,false);
    }
    else
    {
        detalles.innerHTML = '    ';
    }
}

```

Cuando la propiedad readyState tiene un cuatro procedemos a cargar el div "detalles" con el trozo de HTML que se generó en el servidor:

```

detalles.innerHTML = conexion1.responseText;

```

Otra cosa muy importante es inicializar el evento click para los hipervínculos siguiente y anterior que contiene todo listado con paginación:

```

var ob1=document.getElementById('sig');
if (ob1!=null)
    addEvent(ob1,'click',presionEnlace,false);
var ob2=document.getElementById('ant');
if (ob2!=null)
    addEvent(ob2,'click',presionEnlace,false);

```

Es importante notar los if. Esto debido a que cuando no hay más registros que mostrar no habrá un hipervínculo. Imagine que la primer página no tiene que tener un hipervínculo "Anteriores".

Por último tenemos el archivo pagina2.php:

```

<?php
header('Content-Type: text/html; charset=ISO-8859-1');
if (isset($_REQUEST['pos']))
    $inicio=$_REQUEST['pos'];
else
    $inicio=0;
$conexion=mysql_connect("localhost","root","z80")
    or die("Problemas en la conexion");
mysql_select_db("bdajax",$conexion)

```

```

    or die("Problemas en la seleccion de la base de datos");
$registros=mysql_query("select * from comentarios limit $inicio,3",
$conexion) or die("Problemas en el select:".mysql_error());
$impresos=0;
while ($reg=mysql_fetch_array($registros))
{
    $impresos++;
    echo "Nombre:".$reg['nombre']."<br>";
    echo "Fecha:".$reg['fecha']."<br>";
    echo "Comentarios:".$reg['descripcion']."<br>";
    echo "<br>";
}
mysql_close($conexion);
if ($inicio==0)
    echo "anteriores ";
else
{
    $anterior=$inicio-3;
    echo "<a href=\"pagina2.php?pos=$anterior\" id=\"ant\">Anteriores
</a>";
}
if ($impresos==3)
{
    $proximo=$inicio+3;
    echo "<a href=\"pagina2.php?pos=$proximo\" id=\"sig\">Siguientes</a>";
}
else
    echo "siguientes";
?>

```

Lo primero que hacemos es verificar si llega el parámetro pos, como habíamos visto desde el inicializar eventos indicabamos cargar la página pagina2.php sin especificar parámetros. Es decir la variable \$inicio se inicializa con 0. \$inicio nos sirve para saber a partir de cual registro debemos rescatar.

```

if (isset($_REQUEST['pos']))
    $inicio=$_REQUEST['pos'];
else
    $inicio=0;

```

Luego de conectarnos y seleccionar la base de datos procedemos a efectuar un select utilizando el limit para rescatar hasta tres registros (es decir página de tamaño 2):

```

$registros=mysql_query("select * from comentarios limit $inicio,3",
$conexion) or die("Problemas en el select:".mysql_error());

```

Mediante un while recorreremos los registros rescatados y los imprimimos con el comando echo. Además utilizamos un contador para saber cuantos registros se imprimieron:

```

$impresos=0;
while ($reg=mysql_fetch_array($registros))
{
    $impresos++;
    echo "Nombre:".$reg['nombre']."<br>";
    echo "Fecha:".$reg['fecha']."<br>";
    echo "Comentarios:".$reg['descripcion']."<br>";
    echo "<br>";
}

```

```
}
```

Si \$inicio vale cero significa que no hay registros anteriores, en caso que sea distinto a cero creamos un hipervínculo y pasamos como parámetro el valor de \$inicio menos 3:

```
if ($inicio==0)
    echo "anteriores ";
else
{
    $anterior=$inicio-3;
    echo "<a href=\"pagina2.php?pos=$anterior\" id=\"ant\">Anteriores
    </a>";
}
```

Para el hipervínculo de "Siguietes" procedemos de forma similar, si \$impresos vale 3 significa que posiblemente haya más registros que mostrar, por lo que creamos un hipervínculo y pasamos como parámetro el valor de \$inicio más 3.

```
if ($impresos==3)
{
    $proximo=$inicio+3;
    echo "<a href=\"pagina2.php?pos=$proximo\"
id=\"sig\">Siguietes</a>";
}
else
    echo "siguietes";
```

15 - Cargar un control de tipo select

Confeccionaremos un problema que contenga dos controles de tipo select. En el primero almacenaremos una lista de carreras de estudio ("Analista de Sistemas", "Telecomunicaciones" y "WebMaster")

Cuando se seleccione una carrera enviaremos una petición al servidor para que retorne todas las materias que tiene esa carrera y procederemos a la carga del segundo select.

El archivo HTML es el siguiente (pagina1.html):

```
<script src="funciones.js" language="JavaScript"></script>
</head>
<body>
<h1>Prueba de cargar un control de tipo select recuperando datos del servidor
mediante AJAX</h1>
Seleccione la carrera:
<select id="carreras" name="carreras">
<option value="0">Seleccionar....</option>
<option value="1">Analista de sistemas</option>
<option value="2">Telecomunicaciones</option>
<option value="3">WebMaster</option>
</select> <span id="espera"></span> <br>
Materias de la carrera:
<select id="materias" name="materias">
</select> <br>
</body>
```

</html>

Luego el archivo que contiene las funciones de JavaScript (funciones.js) es:

```
addEvent(window,'load',inicializarEventos,false);

function inicializarEventos()
{
    var select1=document.getElementById('carreras');
    addEvent(select1,'change',mostrarMaterias,false);
}

var conexion1;
function mostrarMaterias(e)
{
    var codigo=document.getElementById('carreras').value;
    if (codigo!=0)
    {
        conexion1=crearXMLHttpRequest();
        conexion1.onreadystatechange = procesarEventos;
        conexion1.open('GET','pagina1.php?cod='+codigo, true);
        conexion1.send(null);
    }
    else
    {
        var select2=document.getElementById('materias');
        select2.options.length=0;
    }
}

function procesarEventos()
{
    if(conexion1.readyState == 4)
    {
        var d=document.getElementById('espera');
        d.innerHTML = "";
        var xml = conexion1.responseXML;
        var pals=xml.getElementsByTagName('materia');
        var select2=document.getElementById('materias');
        select2.options.length=0;
        for(f=0;f<pals.length;f++)
        {
            var op=document.createElement('option');
            var texto=document.createTextNode(pals[f].firstChild.nodeValue);
            op.appendChild(texto);
            select2.appendChild(op);
        }
    }
    else
    {
        var d=document.getElementById('espera');
        d.innerHTML = '';
    }
}
```

```

//*****
//Funciones comunes a todos los problemas
//*****
function addEvent(elemento,nomevento,funcion,captura)
{
    if (elemento.attachEvent)
    {
        elemento.attachEvent('on'+nomevento,funcion);
        return true;
    }
    else
    if (elemento.addEventListener)
    {
        elemento.addEventListener(nomeevento,funcion,captura);
        return true;
    }
    else
        return false;
}

function crearXMLHttpRequest()
{
    var xmlHttp=null;
    if (window.ActiveXObject)
        xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
    else
        if (window.XMLHttpRequest)
            xmlHttp = new XMLHttpRequest();
    return xmlHttp;
}

```

En la primer función que se ejecuta luego de haberse cargado completamente la página definimos el evento change para el primer select:

```

var select1=document.getElementById('carreras');
addEvent(select1,'change',mostrarMaterias,false);

```

Es decir cuando hagamos la selección de un item del primer select se dispara la función mostrarMaterias.

La función mostrar materias:

```

function mostrarMaterias(e)
{
    var codigo=document.getElementById('carreras').value;
    if (codigo!=0)
    {
        conexion1=crearXMLHttpRequest()
        conexion1.onreadystatechange = procesarEventos;
        conexion1.open('GET','paginal.php?cod='+codigo, true);
        conexion1.send(null);
    }
    else
    {
        var select2=document.getElementById('materias');
        select2.options.length=0;
    }
}

```

```
}
```

Rescata el valor del primer select (es decir donde estan almacenadas los nombres de carreras. Si está seleccionada procede a crear un objeto de tipo XMLHttpRequest y le pasa como parámetro el código de la carrera respectiva.

En caso de seleccionar el primer item del select (contiene el texto "Seleccionar....") procedemos a borrar el contenido del segundo select.

La función procesarEventos:

```
function procesarEventos()
{
    if(conexion1.readyState == 4)
    {
        var d=document.getElementById('espera');
        d.innerHTML = '';
        var xml = conexion1.responseXML;
        var pals=xml.getElementsByTagName('materia');
        var select2=document.getElementById('materias');
        select2.options.length=0;
        for(f=0;f<pals.length;f++)
        {
            var op=document.createElement('option');
            var texto=document.createTextNode(pals[f].firstChild.nodeValue);
            op.appendChild(texto);
            select2.appendChild(op);
        }
    }
    else
    {
        var d=document.getElementById('espera');
        d.innerHTML = '';
    }
}
```

Rescata el contenido del archivo XML retornado del servidor:

```
var xml = conexion1.responseXML;
```

Como sabemos el archivo XML contiene un conjunto de materias:

```
var pals=xml.getElementsByTagName('materia');
```

Mediante un for las rescatamos y las agregamos al segundo select:

```
for(f=0;f<pals.length;f++)
{
    var op=document.createElement('option');
    var texto=document.createTextNode(pals[f].firstChild.nodeValue);
    op.appendChild(texto);
    select2.appendChild(op);
}
```

Nos queda el programa que se ejecuta en el servidor para generar el archivo XML con las materias de la carrera seleccionada por el visitante (para facilitar el problema no hemos dispuesto esta información en una base de datos.

```

<?php
$car=$_REQUEST['cod'];
if ($car==1)
{
    $materias=array('Programacion I','Analisis Matematico',
                    'Estructura de las Organizaciones','Ética Profesional');
}
if ($car==2)
{
    $materias=array('Fundamentos de Fisica','Analisis Matematico 1',
                    'Ingles Tecnico I','Sistemas de Comunicaciones I
');
}
if ($car==3)
{
    $materias=array('Informatica I','Multimedia I','Bases de Datos');
}

$xml="<?xml version=\"1.0\"?>\n";
$xml.="<materias>\n";
for($f=0;$f<count($materias);$f++)
{
    $xml.="<materia>".$materias[$f]."</materia>\n";
}
$xml.="</materias>\n";
header('Content-Type: text/xml');
echo $xml;
?>

```

Según el código de carrera se genera un vector \$materias con todas las materias de esa carrera.

Luego generamos la estructura del archivo XML respectivo y disponemos en la cabecera de la petición que se trata de un archivo XML:

```

header('Content-Type: text/xml');
echo $xml;

```

16 - Mostrar un tooltip con datos recuperados del servidor en forma asincrónica

Para aplicar varios de los conceptos vistos hasta ahora veamos una implementación de un Tooltip con DHTML y recuperando la información a mostrar del servidor en forma asincrónica. Recordemos que un Tooltip es un mensaje que aparece sobre un objeto cuando disponemos la flecha del mouse sobre el mismo, este recuadro nos informa el objetivo de dicho control.

El archivo HTML muestra un conjunto de div (cuadrados), cuando pasamos la flecha del mouse sobre el cuadrado aparecerá un mensaje, este mensaje lo buscaremos en el servidor.

El archivo HTML es pagina1.html:

```

<html>
<head>
<title>Problema</title>
<script lenguaje="javascript" src="funciones.js"
type="text/javascript"></script>
<link rel="StyleSheet" href="estilos.css" type="text/css">
</head>
<body>
<p>Entre con el mouse al recuadro.</p>
<div class="cuadradito" id="c1"></div>
<div class="cuadradito" id="c2"></div>
<div class="cuadradito" id="c3"></div>
<div class="cuadradito" id="c4"></div>
</body>
</html>

```

Luego tenemos el archivo estilos.css:

```

.cuadradito{
background-color: #f00;
height: 50px;
width: 50px;
margin: 3px;
z-index: -1;
}

#divmensaje {
background-color: yellow;
position: absolute;
visibility: hidden;
padding: 5px;
width: 250px;
z-index: 100;
}

```

El archivo funciones.js:

```

addEvent(window,'load',inicializarEventos,false);

function inicializarEventos()
{
var vec=document.getElementsByTagName('div');
for(f=0;f<vec.length;f++)
{
addEvent(vec[f],'mouseover',mostrarToolTip,false);
addEvent(vec[f],'mouseout',ocultarToolTip,false);
addEvent(vec[f],'mousemove',actualizarToolTip,false);
}
var ele=document.createElement('div');
ele.setAttribute('id','divmensaje');
vec=document.getElementsByTagName('body');
vec[0].appendChild(ele);
}

```

```

function mostrarToolTip(e)
{
    var d = document.getElementById("divmensaje");
    d.style.visibility = "visible";
    if (window.event)
        e=window.event;
    d.style.left = e.clientX + document.body.scrollLeft + 15;
    d.style.top = e.clientY + document.body.scrollTop + 15;
    var ref;
    if (window.event)
        ref=window.event.srcElement;
    else
        if (e)
            ref=e.target;
    recuperarServidorTooltip(ref.getAttribute('id'));
}

function actualizarToolTip(e)
{
    if (window.event)
        e=window.event;
    var d = document.getElementById("divmensaje");
    d.style.left = e.clientX + document.body.scrollLeft + 15;
    d.style.top = e.clientY + document.body.scrollTop + 15;
}

function ocultarToolTip(e)
{
    var d = document.getElementById("divmensaje");
    d.style.visibility = "hidden";
}

var conexion1;
function recuperarServidorTooltip(codigo)
{
    {
        conexion1=crearXMLHttpRequest();
        conexion1.onreadystatechange = procesarEventos;
        conexion1.open('GET','pagina1.php?cod='+codigo, true);
        conexion1.send(null);
    }
}

function procesarEventos()
{
    {
        var d = document.getElementById("divmensaje");
        d.style.visibility = "visible";
        if(conexion1.readyState == 4)
        {
            d.innerHTML=conexion1.responseText;
        }
        else
        {
            d.innerHTML = '' ;
        }
    }
}

```

```

//*****
//Funciones comunes a todos los problemas
//*****
function addEvent(elemento,nomevento,funcion,captura)
{
    if (elemento.attachEvent)
    {
        elemento.attachEvent('on'+nomevento,funcion);
        return true;
    }
    else
    if (elemento.addEventListener)
    {
        elemento.addEventListener(nomeevento,funcion,captura);
        return true;
    }
    else
        return false;
}

function crearXMLHttpRequest()
{
    var xmlhttp=null;
    if (window.ActiveXObject)
        xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
    else
        if (window.XMLHttpRequest)
            xmlhttp = new XMLHttpRequest();
    return xmlhttp;
}

```

Primero obtenemos la referencia a todos los div de la página:

```
var vec=document.getElementsByTagName('div');
```

Definimos las funciones que se disparan para los eventos mouseover, mouseout y mousemove:

```

for (f=0;f<vec.length;f++)
{
    addEvent (vec[f], 'mouseover',mostrarToolTip,false);
    addEvent (vec[f], 'mouseout',ocultarToolTip,false);
    addEvent (vec[f], 'mousemove',mostrarToolTip,false);
}

```

Podemos ver que enlazamos la misma función para los eventos mouseover y mousemove, esto debido a que en dicha función mostramos el Tooltip en la coordenada actual del mouse.

Por otro lado la función inicializarEventos crea un div y lo añade al árbol de objetos, este nos servirá para mostrar el mensaje:

```

var ele=document.createElement('div');
ele.setAttribute('id','divmensaje');
vec=document.getElementsByTagName('body');
vec[0].appendChild(ele);

```

La función mostrarToolTip obtiene la referencia del div que muestra el mensaje (hasta este momento está oculto) y cambia la propiedad visibility a "visible", luego llama a la función recuperarServidorToolTip(ref.getAttribute('id')).

```

function mostrarToolTip(e)
{

```

```

var d = document.getElementById("divmensaje");
d.style.visibility = "visible";
if (window.event)
    e=window.event;
d.style.left = e.clientX + document.body.scrollLeft + 15;
d.style.top = e.clientY + document.body.scrollTop + 15;
var ref;
if (window.event)
    ref=window.event.srcElement;
else
    if (e)
        ref=e.target;
recuperarServidorTooltip(ref.getAttribute('id'));
}

```

La función `ocultarTooTip` solo oculta el div del mensaje.

```

function ocultarToolTip(e)
{
    var d = document.getElementById("divmensaje");
    d.style.visibility = "hidden";
}

```

La función `recuperarServidorTooltip` recibe el valor del atributo `id` del div donde se encuentra la flecha del mouse. Se crea un objeto de la clase `XMLHttpRequest` y se envía el código del div respectivo.

```

var conexion1;
function recuperarServidorTooltip(codigo)
{
    conexion1=crearXMLHttpRequest();
    conexion1.onreadystatechange = procesarEventos;
    conexion1.open('GET','paginal.php?cod='+codigo, true);
    conexion1.send(null);
}

```

Luego la función `procesarEventos` carga el div y procede hacerlo visible:

```

function procesarEventos()
{
    var d = document.getElementById("divmensaje");
    d.style.visibility = "visible";
    if(conexion1.readyState == 4)
    {
        d.innerHTML=conexion1.responseText;
    }
    else
    {
        d.innerHTML = '    ';
    }
}

```

Por último el programa del servidor genera un trozo de HTML dependiendo del código de div enviado (`pagina1.php`):

```

<?php
if ($_REQUEST['cod']=='c1')

```

```

{
  echo "<p>Primer tooltip.</p>";
  echo "<p>aaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaa aaaaaa
aaaaaaaaaa";
  echo "aaaaaaaaaaaaaaaaaaaaa aaaaaaaaa aaaaaaaaaaaaaa aaaaaaaaaaaaaa";
  echo "aaaaaaaaaaaaaaaaaaaaa aaaaaaaa aaaaaaaaaaaaaa aaaaaaaa aaaaaa";
  echo "aaaaaaaaaaaaaaaaaaaaa aaaaa aaaaaaaaaaaaaaaaaaaaaa aaaaaa aaaaaaaa";
  echo "aaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaa aaaaaa
aaaaaaaaaaaaaaaaaaaaa</p>";
}
if ($_REQUEST['cod']=='c2')
{
  echo "<p>Segundo tooltip.</p>";
  echo "<p>bbbbbbbbb bbbbbbbbbbb bbbbbbbb bbbbbbbbbbbbbb bbbbbbbbbbb
bb";
  echo "bbbbbbbbb bbbbbbbbbbb bbbbbbbbbbbbbb bbbbbbbbbbb bbbbbbbbbbb
bbb</p>";
}
if ($_REQUEST['cod']=='c3')
  echo "<p>Tercer tooltip.</p>";
if ($_REQUEST['cod']=='c4')
  echo "<p>Cuarto tooltip.</p>";
?>

```

En la realidad estos datos generalmente se rescatan de una base de datos, pero para concentrarnos en lo esencial hemos dejado esto de lado.

17 - Autocompletar un control de tipo text

Implementaremos con AJAX el concepto de autocompletar o también conocido como lista de sugerencias. A medida que tipemos caracteres dentro de un control de tipo text mostraremos un conjunto de palabras que comienzan con los caracteres ingresados hasta el momento.

En la realidad los datos se extraen de una base de datos pero para simplificar el problema no lo haremos en este caso.

Veremos los distintos archivos que intervienen para solucionar el problema:

pagina1.html

```

<html>
<head>
<title>Problema</title>
<script lenguaje="javascript" src="funciones.js"
type="text/javascript"></script>
<link rel="StyleSheet" href="estilos.css" type="text/css">
</head>
<body>
<form action="#">
Ingrese una palabra que comience con a:<br>
<input type="text" id="palabra" autocomplete="off"><br>

```

```
<div id="resultados"></div>
<input type="submit" value="buscar">
</form>
</body>
</html>
```

Esta página no tiene nada nuevo. Es importante notar que hemos dispuesto los elementos para incorporar la hoja de estilo y el archivo donde se encuentran las funciones en JavaScript:

```
<script lenguaje="javascript" src="funciones.js"
type="text/javascript"></script>
<link rel="StyleSheet" href="estilos.css" type="text/css">
```

La hoja de estilo es estilos.css:

```
#resultados {
  position: absolute;
  background: #ff0;
}
```

Es importante notar que la clase #resultados hemos definido la propiedad position con el valor absolute. Esto significa que el cuadro donde aparecerá la lista de palabras estará superpuesta a otro contenido de la página. Además el color de fondo es amarillo.

El archivo funciones.js

```
addEvent(window,'load',inicializarEventos,false);

function inicializarEventos()
{
  var ob=document.getElementById('palabra');
  addEvent(ob,'keyup',presionTecla,false);
}

var conexion1;
function presionTecla(e)
{
  conexion1=crearXMLHttpRequest();
  conexion1.onreadystatechange = procesarEventos;
  palabra=document.getElementById('palabra').value;
  conexion1.open('GET','pagina1.php?palabra='+palabra, true);
  conexion1.send(null);
}

function procesarEventos()
{
  var resultados = document.getElementById("resultados");
  if(conexion1.readyState == 4)
  {
    if (conexion1.status==200)
    {
      var xml = conexion1.responseXML;
      var pals=xml.getElementsByTagName('palabra');
      resultados.innerHTML="";
    }
  }
}
```

```

        for(f=0;f<pals.length;f++)
        {
            resultados.innerHTML = resultados.innerHTML + pals[f].firstChild.nodeValue
+ '<br>';
        }
    }
    else
        alert(conexion1.statusText);
}
else
{
    resultados.innerHTML = '    ';
}
}

//*****
//Funciones comunes a todos los problemas
//*****
function addEvent(elemento,nomevento,funcion,captura)
{
    if (elemento.attachEvent)
    {
        elemento.attachEvent('on'+nomevento,funcion);
        return true;
    }
    else
        if (elemento.addEventListener)
        {
            elemento.addEventListener(nomeevento,funcion,captura);
            return true;
        }
        else
            return false;
}

function crearXMLHttpRequest()
{
    var xmlHttp=null;
    if (window.ActiveXObject)
        xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
    else
        if (window.XMLHttpRequest)
            xmlHttp = new XMLHttpRequest();
    return xmlHttp;
}

```

En la función inicializarEventos:

```

var ob=document.getElementById('palabra');
addEvent(ob, 'keyup', presionTecla, false);

```

Enlazamos la función presionTecla para cuando ocurre el evento keyup del único control text que contiene la página. Es decir que cada vez que el usuario presione una tecla, al momento de soltarla se ejecuta la función presionTecla.

La función presionTecla:

```
var conexion1;  
function presionTecla(e)  
{  
    conexion1=crearXMLHttpRequest();  
    conexion1.onreadystatechange = procesarEventos;  
    palabra=document.getElementById('palabra').value;  
    conexion1.open('GET', 'pagina1.php?palabra='+palabra, true);  
    conexion1.send(null);  
}
```

crea un objeto de la clase XMLHttpRequest, extrae el contenido del text y procede a efectuar la petición al servidor pasando mediante el método GET la cadena ingresada hasta ese momento.

Luego la función procesarEventos:

```
function procesarEventos()  
{  
    var resultados = document.getElementById("resultados");  
    if(conexion1.readyState == 4)  
    {  
        if (conexion1.status==200)  
        {  
            var xml = conexion1.responseXML;  
            var pals=xml.getElementsByTagName('palabra');  
            resultados.innerHTML='';  
            for(f=0;f<pals.length;f++)  
            {  
                resultados.innerHTML = resultados.innerHTML +  
pals[f].firstChild.nodeValue + '<br>';  
            }  
        }  
        else  
            alert(conexion1.statusText);  
    }  
    else  
    {  
        resultados.innerHTML = '    ';  
    }  
}
```

Cuando la propiedad readyState informa que llegaron todos los datos y además la propiedad status retorna un 200 procedemos a rescatar los datos mediante la propiedad responseXML.

Por último procedemos a mostrar la lista de palabras dentro del div resultados.

Nos queda el archivo que se ejecuta en el servidor y nos retorna el archivo XML con la lista de palabras (pagina1.php):

```
<?php  
$pal=$_REQUEST['palabra'];  
$vec=array('alma','algo','amo','aro','animo','arbol','abrir');  
if (strlen($pal)>0)  
{  
    for($f=0;$f<count($vec);$f++)
```

```

    {
    if ($pal==substr($vec[$f],0,strlen($pal)))
        $veciguales[]=$vec[$f];
    }
}
$xml="<?xml version=\"1.0\"?>\n";
$xml.="<palabras>\n";
if (isset($veciguales))
{
    for($f=0;$f<count($veciguales);$f++)
    {
        $xml.="<palabra>".$veciguales[$f]."</palabra>\n";
    }
}
$xml.="</palabras>\n";
header('Content-Type: text/xml');
echo $xml;
?>

```

Las palabras las tenemos en un vector (en la realidad generalmente se extraen de una base de datos):

```
$vec=array('alma','algo','amo','aro','animo','arbol','abrir');
```

Almacenamos en un vector todas las que comienzan con la cadena ingresada hasta el momento:

```

    for($f=0;$f<count($vec);$f++)
    {
    if ($pal==substr($vec[$f],0,strlen($pal)))
        $veciguales[]=$vec[$f];
    }

```

Generamos una cadena cuyo contenido es XML:

```

$xml="<?xml version=\"1.0\"?>\n";
$xml.="<palabras>\n";
if (isset($veciguales))
{
    for($f=0;$f<count($veciguales);$f++)
    {
        $xml.="<palabra>".$veciguales[$f]."</palabra>\n";
    }
}
$xml.="</palabras>\n";

```

Informamos al navegador que enviaremos un archivo XML:

```
header('Content-Type: text/xml');
```

Y por último indicamos la información a enviar:

```
echo $xml;
```

18 - Encuentra con AJAX

Confeccionaremos un nuevo problema donde podemos utilizar AJAX. Vamos a desarrollar una pequeña aplicación para hacer una encuesta.

La característica principal es que cuando el operador haga su selección procederemos a enviar la selección al servidor y generaremos un gráfico en forma dinámica en el servidor y procederemos a actualizar en forma asincrónica solo una parte de la página.

pagina1.php

```
<body>
<h1>Implementación de una encuesta empleando AJAX.</h1>
<form action="pagina1.php" method="post" id="formulario">
<fieldset>
<h2>Que lenguaje utiliza para la implementación de páginas
dinámicas?</h2>
<div id="encuesta">
<input type="radio" id="radio1" name="radio">PHP<br>
<input type="radio" id="radio2" name="radio">ASP<br>
<input type="radio" id="radio3" name="radio">JSP<br>
<input type="submit" value="Enviar" id="enviar">
</fieldset>
</div>
</form>
</body>
</html>
```

funciones.js

```
addEvent(window,'load',inicializarEventos,false);

function inicializarEventos()
{
    var ref=document.getElementById('formulario');
    addEvent(ref,'submit',enviarDatos,false);
}

function enviarDatos(e)
{
    {
        if (window.event)
            window.event.returnValue=false;
        else
            if (e)
                e.preventDefault();
        if (document.getElementById('radio1').checked)
            enviarSeleccion(1);
        else
            if (document.getElementById('radio2').checked)
                enviarSeleccion(2);
            else
                if (document.getElementById('radio3').checked)
                    enviarSeleccion(3);
    }
}

var conexion1;
```

```

function enviarSeleccion(cod)
{
    conexion1=crearXMLHttpRequest();
    conexion1.onreadystatechange = procesarEventos;
    var aleatorio=Math.random();
    conexion1.open('GET','pagina1.php?codigo='+cod+"&aleatorio="+aleatorio,
true);
    conexion1.send(null);
}

function procesarEventos()
{
    var encuesta = document.getElementById("encuesta");
    if(conexion1.readyState == 4)
    {

        encuesta.innerHTML = '      ';
    }
    else
    {

        encuesta.innerHTML = '      ';
    }
}

//*****
//Funciones comunes a todos los problemas
//*****
function addEvent(elemento,nomevento,funcion,captura)
{
    if (elemento.attachEvent)
    {
        elemento.attachEvent('on'+nomevento,funcion);
        return true;
    }
    else
    if (elemento.addEventListener)
    {
        elemento.addEventListener(nomeevento,funcion,captura);
        return true;
    }
    else
        return false;
}

function crearXMLHttpRequest()
{
    var xmlHttp=null;
    if (window.ActiveXObject)
        xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
    else
        if (window.XMLHttpRequest)
            xmlHttp = new XMLHttpRequest();
    return xmlHttp;
}

```

```
}
```

Procedemos a capturar el evento submit para enviar los datos en forma asincrónica:

```
function inicializarEventos()
{
    var ref=document.getElementById('formulario');
    addEvent(ref,'submit',enviarDatos,false);
}
```

Cuando se presiona el botón submit procedemos a llamar a la función enviarSeleccion con el número de opción seleccionada:

```
function enviarDatos(e)
{
    if (window.event)
        window.event.returnValue=false;
    else
        if (e)
            e.preventDefault();
    if (document.getElementById('radio1').checked)
        enviarSeleccion(1);
    else
        if (document.getElementById('radio2').checked)
            enviarSeleccion(2);
        else
            if (document.getElementById('radio3').checked)
                enviarSeleccion(3);
}
```

La función enviarSeleccion procede a crear un objeto de la clase XMLHttpRequest y envía el número de opción seleccionada de la encuesta y un valor aleatorio para que no rescate el navegador una página que se encuentre en la cache de la computadora:

```
var conexion1;
function enviarSeleccion(cod)
{
    conexion1=crearXMLHttpRequest();
    conexion1.onreadystatechange = procesarEventos;
    var aleatorio=Math.random();
    conexion1.open('GET','paginal.php?
codigo='+cod+"&aleatorio="+aleatorio, true);
    conexion1.send(null);
}
```

La función procesarEventos carga la imagen generada dinámicamente en el servidor. Esto se hace cuando el objeto XMLHttpRequest nos informa que los datos fueron generados completamente:

```
function procesarEventos()
{
    var encuesta = document.getElementById("encuesta");
    if(conexion1.readyState == 4)
    {

        encuesta.innerHTML = '      ';
    }
    else
```

```

{
    encuesta.innerHTML = '      ';
}
}

```

Por último el archivo pagina1.php

```

<?php
$preg=$_REQUEST['codigo'];
$conexion=mysql_connect("localhost","root","z80")
    or die("Problemas en la conexion");
mysql_select_db("bdajax",$conexion)
    or die("Problemas en la seleccion de la base de datos");
if ($preg==1)
    $registros=mysql_query("update encuesta
        set pregunta1=pregunta1+1
        where codigo=1",$conexion) or
        die("Problemas en el select:".mysql_error());
if ($preg==2)
    $registros=mysql_query("update encuesta
        set pregunta2=pregunta2+1
        where codigo=1",$conexion) or
        die("Problemas en el select:".mysql_error());
if ($preg==3)
    $registros=mysql_query("update encuesta
        set pregunta3=pregunta3+1
        where codigo=1",$conexion) or
        die("Problemas en el select:".mysql_error());
$registro=mysql_query("select * from encuesta where codigo=1",$conexion)
    or die(mysql_error());
$reg=mysql_fetch_array($registro);
$barra=new Barra(500,300);
$barra->fijarprofundidad(20);
$barra->sumar($reg['pregunta1'], "PHP");
$barra->sumar($reg['pregunta2'], "ASP");
$barra->sumar($reg['pregunta3'], "JSP");
$barra->graficar();

class Barra {
    var $ancho;
    var $alto;
    var $imagen;
    var $profundidad;
    var $colorfondo;
    var $vectorcolorfondo;
    var $colorbarra;
    var $vectorcolorbarra;
    var $colorvalores;
    //almacena los valores y los titulos de cada barra
    var $datos;
    function sumar($valor,$titulo)
    {
        $indice=count($this->datos);
        $this->datos[$indice]['valor']=$valor;
        $this->datos[$indice]['titulo']=$titulo;
    }
}

```

```

}
function fijarcolorfondo($rojo,$verde,$azul)
{
  $this->vectorcolorfondo[0]=$rojo;
  $this->vectorcolorfondo[1]=$verde;
  $this->vectorcolorfondo[2]=$azul;
  $this->colorfondo=ImageColorAllocate ($this->imagen,$this-
>vectorcolorfondo[0],
  $this->vectorcolorfondo[1],$this->vectorcolorfondo[2]);
  ImageFill($this->imagen,0,0,$this->colorfondo);
}

function fijarcolorbarra($rojo,$verde,$azul)
{
  $this->vectorcolorbarra[0]=$rojo;
  $this->vectorcolorbarra[1]=$verde;
  $this->vectorcolorbarra[2]=$azul;
  $this->colorbarra=ImageColorAllocate($this->imagen,$this-
>vectorcolorbarra[0],
  $this->vectorcolorbarra[1],$this->vectorcolorbarra[2]);
}

function fijarprofundidad($profundidad)
{
  $this->profundidad=$profundidad;
}

//Metodo privado
function mayor()
{
  $primera=true;
  $may=0;
  foreach ($this->datos as $val)
  {
    if ($primera)
    {
      $primera=false;
      $may=$val['valor'];
    }
    if ($val['valor']>$may)
      $may=$val['valor'];
  }
  return $may;
}

function graficarsombraizquierda($columna,$y1,$y2)
{
  $rojo=$this->vectorcolorbarra[0]-90;
  if ($rojo<0)
    $rojo=0;
  $verde=$this->vectorcolorbarra[1]-90;
  if ($verde<0)
    $verde=0;
  $azul=$this->vectorcolorbarra[2]-90;
  if ($azul<0)
    $azul=0;
  $colorsombra=imageColorAllocate($this->imagen,$rojo,$verde,$azul);

```

```

$puntos[]=$columna;
$puntos[]=$y1;
$puntos[]=$columna;
$puntos[]=$y2;
$puntos[]=$columna+$this->profundidad;
$puntos[]=$y2-$this->profundidad;
$puntos[]=$columna+$this->profundidad;
$puntos[]=$y1-$this->profundidad;
imagefilledpolygon($this->imagen,$puntos,4,$colorsombra);
$colorbordebarra=imageColorAllocate($this->imagen,0,0,0);
imagepolygon($this->imagen,$puntos,4,$colorbordebarra);
}

function graficarsombrasuperior($columna,$y1,$anchobarra)
{
    $rojo=$this->vectorcolorbarra[0]-40;
    if ($rojo<0)
        $rojo=0;
    $verde=$this->vectorcolorbarra[1]-40;
    if ($verde<0)
        $verde=0;
    $azul=$this->vectorcolorbarra[2]-40;
    if ($azul<0)
        $azul=0;
    $colorsombra=imageColorAllocate($this->imagen,$rojo,$verde,$azul);
    $puntos[]=$columna;
    $puntos[]=$y1;
    $puntos[]=$columna+$anchobarra;
    $puntos[]=$y1;
    $puntos[]=$columna+$anchobarra+$this->profundidad;
    $puntos[]=$y1-$this->profundidad;
    $puntos[]=$columna+$this->profundidad;
    $puntos[]=$y1-$this->profundidad;
    imagefilledpolygon($this->imagen,$puntos,4,$colorsombra);
    $colorbordebarra=imageColorAllocate($this->imagen,0,0,0);
    imagepolygon($this->imagen,$puntos,4,$colorbordebarra);
}

function Barra($ancho,$alto)
{
    $this->ancho=$ancho;
    $this->alto=$alto;
    $this->imagen=imageCreate($this->ancho,$this->alto);
    $this->vectorcolorfondo[0]=0;
    $this->vectorcolorfondo[1]=0;
    $this->vectorcolorfondo[2]=255;
    $this->colorfondo=ImageColorAllocate($this->imagen,$this->vectorcolorfondo[0],
        $this->vectorcolorfondo[1],$this->vectorcolorfondo[2]);
    ImageFill($this->imagen,0,0,$this->colorfondo);
    $this->vectorcolorbarra[0]=255;
    $this->vectorcolorbarra[1]=255;
    $this->vectorcolorbarra[2]=0;
    $this->colorbarra=ImageColorAllocate($this->imagen,$this->vectorcolorbarra[0],
        $this->vectorcolorbarra[1],$this->vectorcolorbarra[2]);

```

```

$this->profundidad=10;
$this->colorvalores=ImageColorAllocate($this->imagen,0,0,0);
}
function graficar()
{
    $may=$this->mayor();
    $anchobarra=($this->ancho-110)/count($this->datos);
    $x1=10;
    $y1=$this->alto-50;
    $colorbordebarra=ImageColorAllocate($this->imagen,0,0,0);
    foreach($this->datos as $reg)
    {
        $altura=($reg['valor']/$may)*($this->alto-80);
        imagefilledrectangle($this->imagen,$x1,$y1-$altura,$x1+$anchobarra,
$y1,$this->colorbarra);
        imagerectangle($this->imagen,$x1,$y1-$altura,$x1+$anchobarra,
$y1,$colorbordebarra);
        ImageString($this->imagen,2,$x1+3,$y1,$reg['titulo'],$this->colorbarra);
        $this->graficarsombraizquierda($x1+$anchobarra,$y1-$altura,$y1);
        $this->graficarsombrasuperior($x1,$y1-$altura,$anchobarra);
        ImageString($this->imagen,2,$x1+3,$y1-$altura+5,$reg['valor'],$this-
>colorvalores);
        $x1=$x1+$anchobarra+(100/count($this->datos));
    }
    Header ("Content-type: image/png");
    ImagePNG ($this->imagen,"encuesta.png");
    ImageDestroy($this->imagen);
}
}
?>

```

No analizaremos mucho este archivo ya que no es objetivo del curso el aprendizaje de PHP.

Básicamente la clase Barra nos permite crear un gráfico de barra creando un objeto de dicha clase y pasando los datos respectivos:

```

$barra=new Barra(500,300);
$barra->fijarprofundidad(20);
$barra->sumar($reg['pregunta1'], "PHP");
$barra->sumar($reg['pregunta2'], "ASP");
$barra->sumar($reg['pregunta3'], "JSP");
$barra->graficar();

```

19 - Qué es JSON?

JSON es el acrónimo de JavaScript Object Notation.

JSON es un formato alternativo de envío y recepción de datos, es decir reemplaza a XML o el envío de texto plano.

Este formato de datos es más liviano que XML.

Veremos que hace el código más sencillo ya que utiliza el código JavaScript como modelo de datos.

Tenemos que repasar un poco como se definen los array literales y objetos literales en JavaScript, ya que serán las estructuras para la transmisión de datos:

```
var usuario=['juan',26];
```

Como vemos los elementos de un array literal se encierran entre corchetes y los valores contenidos van separados por coma.

Cuando definimos un array literal no le indicamos a cada elemento de que tipo se trata, podemos almacenar cadenas (entre comillas), números, valores lógicos (true,false) y el valor null.

Para acceder a los elementos de un array literal lo hacemos por su nombre y entre corchetes indicamos que elementos queremos acceder:

```
alert(usuario[0]); //Imprimimos el primer elemento del array
alert(usuario[1]); //Imprimimos el segundo elemento del array
```

Veamos como definimos los objetos literales en JavaScript:

```
var persona={
  'nombre':'juan',
  'clave':'xyz',
  'edad':26
};
```

Los objetos literales se definen por medio de pares "nombre":"valor"

Todo objeto literal tiene un nombre, en el ejemplo le llamé persona. Un objeto literal contiene una serie de propiedades con sus respectivos valores. Todas las propiedades del objetos se encuentran encerradas entre llaves. Luego cada propiedad y valor se las separa por dos puntos. Y por último cada propiedad se las separa de las otras propiedades por medio de la coma.

Para acceder a las propiedades del objeto literal lo podemos hacer de dos formas:

```
alert(persona.nombre); //Imprime el valor de la propiedad nombre del
objeto persona.
```

La segunda forma es indicando la propiedad entre corchetes:

```
alert(persona['nombre']);
```

Luego se pueden combinar objetos literales y array literales:

```
var persona={
  'nombre':'juan',
  'edad':22,
  'estudios':['primario','secundario']
};
```

Como podemos ver podemos crear estructuras de datos complejas combinando objetos literales y array literales.

Luego para acceder a los distintos elementos tenemos:

```
alert(persona.nombre);  
alert(persona.estudios[0]);  
alert(persona.estudios[1]);
```

La sintaxis de JSON difiere levemente de lo visto anteriormente:

```
{  
  'nombre':'juan',  
  'edad':22,  
  'estudios':['primario','secundario']  
}
```

Como podemos ver en la sintaxis JSON no aparecen variables, sino directamente indicamos entre llaves las propiedades y sus valores.

También hemos eliminado el punto y como luego de la llave final. El resto es igual.

Ahora veamos la diferencia entre JSON y XML utilizando este ejemplo:

XML:

```
<persona>  
<nombre>juan</nombre>  
<edad>22</edad>  
<estudios>  
<estudio>primario</estudio>  
<estudio>secundario</estudio>  
</estudios>  
</persona>
```

Y como vimos en JSON:

```
{  
  'nombre':'juan',  
  'edad':22,  
  'estudios':['primario','secundario']  
}
```

Podemos ver que es mucho más directa la definición de esta estructura (de todos modos cuando la estructura a representar es muy compleja XML sigue siendo la opción principal)

Como podemos ver si tenemos que transmitir estas estructuras por internet la notación JSON es más liviana.

Otra ventaja es como recuperamos los datos en el navegador, como vimos si se trata de un archivo XML llamamos al método requestXML y luego accedemos por medio del DOM

En cambio con JSON al llegar el archivo procedemos a generar una variable en JavaScript que recree el objeto literal, esto mediante la función eval:

```
var persona=eval('(' + conexion1.responseText + ')');
```

Ya veremos en el próximo concepto como recuperar los datos del servidor mediante el objeto XMLHttpRequest

Para probar y generar un objeto a partir de una notación JSON haremos el siguiente problema:

Confeccionar una página que contenga un botón. Al ser presionado evaluar un string que almacena un objeto literal con notación JSON. El objeto literal debe representar las características de una computadora (procesador, memoria ram, capacidad de cada disco duro)

Mostrar los datos mediante el método alert

Hay que tener bien en cuenta que en este problema no hay nada de AJAX ya que no nos comunicaremos con el servidor para el envío de datos.

pagina1.html

```
<html>
<head>
<title>Problema</title>
<script src="funciones.js" language="JavaScript"></script>
</head>
<body>
<h1>Evaluar una variable que contiene notación JSON.</h1>
<input type="button" id="boton1" value="Ver">
</body>
</html>
```

funciones.js

```
addEvent(window,'load',inicializarEventos,false);

function inicializarEventos()
{
    var ob=document.getElementById('boton1');
    addEvent(ob,'click',presionBoton,false);
}

function presionBoton(e)
{
    var cadena="{ 'microprocesador':'pentium'," +
                " 'memoria':1024," +
                " 'discos':[80,250]" +
                " }";
    var maquina=eval('(' + cadena + ')');
    alert('microprocesador:'+maquina.microprocesador);
    alert('Memoria ram:'+maquina.memoria);
    alert('Capacidad disco 1:'+maquina.discos[0]);
    alert('Capacidad disco 2:'+maquina.discos[1]);
}

//*****
//Funciones comunes a todos los problemas
//*****
function addEvent(elemento,nomevento,funcion,captura)
{
    if (elemento.attachEvent)
    {
        elemento.attachEvent('on'+nomevento,funcion);
    }
}
```

```

    return true;
}
else
    if (elemento.addEventListener)
    {
        elemento.addEventListener(nomevento,funcion,captura);
        return true;
    }
    else
        return false;
}

```

Cuando se presiona el botón se ejecuta la función `presionBoton`. En esta lo primero que hacemos definimos un string que contiene un objeto con notación JSON:

```

var cadena="{ 'microprocesador':'pentium'," +
            " 'memoria':1024," +
            " 'discos':[80,250]" +
            " }";

```

Seguidamente pasamos a evaluar este string:

```

var maquina=eval('(' + cadena + ')');

```

Ahora si tenemos un objeto JavaScript. Esto se logra utilizando la función `eval` de JavaScript. Es importante que siempre al string que contiene la notación JSON le debemos anteceder el paréntesis de apertura y finalizarlo con el paréntesis de cerrado (esto para que JavaScript no tenga problemas con las llaves de apertura y cierre de la notación JSON).

Una vez que tenemos el objeto en JavaScript procedemos a acceder a sus atributos:

```

alert('microprocesador:'+maquina.microprocesador);
alert('Memoria ram:'+maquina.memoria);
alert('Capacidad disco 1:'+maquina.discos[0]);
alert('Capacidad disco 2:'+maquina.discos[1]);

```

20 - Recuperar datos del servidor en formato JSON

Para ver como recuperamos datos del servidor en formato JSON en lugar de texto plano o XML implementaremos el siguiente ejemplo:

Confeccionar un sitio que permita ingresar el documento de una persona y nos retorne su apellido, nombre y lugar donde debe votar.

Para reducir el tamaño del problema y concentrarnos en la forma de transmisión de datos y su posterior recuperación en el navegador no implementaremos una base de datos en el servidor (en la realidad los datos de los votantes se encontrarían en una tabla)

El archivo `pagina1.html`:

```

<html>
<head>
<title>Problema</title>
<script src="funciones.js" language="JavaScript"></script>

```

```

</head>
<body>
Ingrese dni (solo existen los valores 1,2 y 3):<input type="text"
name="dni" id="dni" size="10"><br>
<input type="button" value="Enviar" id="boton1">
<div id="resultados"></div>
</body>
</html>

```

El archivo html no tiene nada especial. Definimos el div resultados para mostrar posteriormente los datos devueltos por el servidor.

El archivo funciones.js:

```

addEvent(window,'load',inicializarEventos,false);

function inicializarEventos()
{
    var ob=document.getElementById('boton1');
    addEvent(ob,'click',presionBoton,false);
}

function presionBoton(e)
{
    var ob=document.getElementById('dni');
    recuperarDatos(ob.value);
}

var conexion1;
function recuperarDatos(dni)
{
    {
        conexion1=crearXMLHttpRequest();
        conexion1.onreadystatechange = procesarEventos;
        conexion1.open('GET','pagina1.php?dni='+dni, true);
        conexion1.send(null);
    }
}

function procesarEventos()
{
    {
        var resultados = document.getElementById("resultados");
        if(conexion1.readyState == 4)
        {
            var datos=eval("(" + conexion1.responseText + ")");
            var salida = "Apellido:"+datos.apellido+"<br>";
            salida=salida+"Nombre:"+datos.nombre+"<br>";
            salida=salida+"Dirección donde debe votar:"+datos.direccion;
            resultados.innerHTML = salida;
        }
        else

        {
            resultados.innerHTML = "Cargando...";
        }
    }
}

```

```

//*****
//Funciones comunes a todos los problemas
//*****
function addEvent(elemento,nomevento,funcion,captura)
{
    if (elemento.attachEvent)
    {
        elemento.attachEvent('on'+nomevento,funcion);
        return true;
    }
    else
    if (elemento.addEventListener)
    {
        elemento.addEventListener(nomeevento,funcion,captura);
        return true;
    }
    else
        return false;
}

function crearXMLHttpRequest()
{
    var xmlHttp=null;
    if (window.ActiveXObject)
        xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
    else
        if (window.XMLHttpRequest)
            xmlHttp = new XMLHttpRequest();
    return xmlHttp;
}

```

Cuando se presiona el botón enviar se ejecuta la función:

```

function presionBoton(e)
{
    var ob=document.getElementById('dni');
    recuperarDatos(ob.value);
}

```

En esta función recuperamos el documento ingresado por el operador y llamamos a la función recuperarDatos pasando como parámetro el dni de la persona.

La función:

```

var conexion1;
function recuperarDatos(dni)
{
    conexion1=crearXMLHttpRequest();
    conexion1.onreadystatechange = procesarEventos;
    conexion1.open('GET','paginal.php?dni='+dni, true);
    conexion1.send(null);
}

```

crea un objeto de la clase XMLHttpRequest y procede a realizar una comunicación asincrónica con el servidor pasando por parámetro GET el dni ingresado por el visitante al sitio.

Luego la función procesarEventos se ejecuta para cada uno de los estados de la petición:

```
function procesarEventos()
{
    var resultados = document.getElementById("resultados");
    if(conexion1.readyState == 4)
    {
        var datos=eval("(" + conexion1.responseText + ")");
        var salida = "Apellido:"+datos.apellido+"<br>";
        salida=salida+"Nombre:"+datos.nombre+"<br>";
        salida=salida+"Dirección donde debe votar:"+datos.direccion;
        resultados.innerHTML = salida;
    }
    else
    {
        resultados.innerHTML = "Cargando...";
    }
}
```

Cuando la propiedad readyState tiene almacenado un 4 procedemos a recuperar el string devuelto por la conexión como veremos un poco más abajo se trata de un string que almacena los datos en formato JSON.

Procedemos mediante la función eval a generar un objeto en JavaScript y posteriormente mostramos los datos accediendo a los atributos de dicho objeto.

Como podemos observar es muy fácil acceder luego a la información devuelta por el servidor.

Por último el archivo pagina1.php:

```
<?php
header('Content-Type: text/txt; charset=ISO-8859-1');
$nombre="";
$apellido="";
$direccion="";
if ($_REQUEST['dni']=='1')
{
    $nombre='Juan';
    $apellido='Rodriguez';
    $direccion='Colon 123';
}
if ($_REQUEST['dni']=='2')
{
    $nombre='Ana';
    $apellido='Maldonado';
    $direccion='Lima 245';
}
if ($_REQUEST['dni']=='3')
{
    $nombre='laura';
    $apellido='Pueyrredon';
    $direccion='Laprida 785';
}

echo "{
    'nombre': '$nombre',
    'apellido': '$apellido',
    'direccion': '$direccion'
}
```

```
?>    }";
```

Como dijimos para concentrarnos en JSON no extraemos la información de una base de datos. Por medio de tres if verificamos que número de documento se trata y procedemos a inicializar tres variables.

Lo más interesante es como procedemos a generar la salida con formato JSON:

```
echo "{
    'nombre': '$nombre',
    'apellido': '$apellido',
    'direccion': '$direccion'
}";
```

Recordemos que todo objeto JSON debe ir entre llaves y cada atributo le debe seguir el caracter dos puntos y el valor de dicho atributo.

21 - De PHP a JSON (utilizando la librería JSON.php)

En el concepto anterior habíamos visto como generar un archivo con formato JSON en el servidor y enviárselo al cliente (navegador). Esta metodología de generar el string con formato JSON puede ser muy engorroso cuando las estructuras comienzan a ser más complejas.

Se cuenta con una clase que transforma vectores y clases de PHP a formato JSON en forma automática.

Para ver esta librería llamada [JSON.php](#) confeccionaremos un problema que rescate un conjunto de registros de una tabla MySQL y seguidamente los transforme en formato JSON. En el navegador mediante eval generaremos un vector JavaScript y procederemos a mostrarlo.

Como podemos observar mientras sea más compleja la estructura a generar, más propenso a errores será la generación.

El problema que resolveremos es el siguiente:

Se tiene una tabla llamada "perifericos" donde almacenamos el código, descripción y precio de distintos periféricos de computadoras. Generar un archivo JSON en el servidor y proceder a mostrar los datos de los periféricos en el navegador.

pagina1.html:

```
<html>
<head>
<title>Problema</title>
<script src="funciones.js" language="JavaScript"></script>
</head>
<body>
<h2>Recuperar datos del servidor almacenados en una base de datos en formato
JSON utilizando la librería JSON.php</h2>
<br>
<input type="button" value="Recuperar" id="boton1">
<div id="resultados"></div>
```

```
</body>
</html>
```

Este archivo no tiene nada nuevo.

Lo más interesante y nuevo se presenta en el archivo pagina1.php:

```
<?php
$conexion=mysql_connect("localhost","root","z80")
    or die("Problemas en la conexion");
mysql_select_db("bdajax",$conexion)
    or die("Problemas en la seleccion de la base de datos");
$registros=mysql_query("select codigo,descripcion,precio from perifericos",
$conexion)
    or die("Problemas en el select".mysql_error());
while ($reg=mysql_fetch_array($registros))
{
    $vec[]=$reg;
}

require('../JSON.php');
$json=new Services_JSON();
$cad=$json->encode($vec);
echo $cad;
?>
```

Lo primero que hacemos es guardar todos los registros de la tabla perifericos en el vector llamado \$vec, esto mediante el ciclo:

```
while ($reg=mysql_fetch_array($registros))
{
    $vec[]=$reg;
}
```

Ahora debemos incluir el archivo llamado "JSON.php" y crear un objeto de la clase services_JSON:

```
require('../JSON.php');
$json=new Services_JSON();
```

El archivo yo lo almacené en una carpeta que se encuentra inmediatamente en el nivel superior, por eso la sintaxis ../

Luego el método que automáticamente convierte el vector PHP en formato JSON es:

```
$cad=$json->encode($vec);
```

Ahora ya tenemos en la variable \$cad el contenido del vector en formato JSON.

Procedemos seguidamente a la salida de esta cadena:

```
echo $cad;
```

Tengamos en cuenta que las cuatro líneas presentadas anteriormente nos evitan tener que generar la cadena en formato JSON en forma manual como podemos ver seguidamente:

```
$cad='[';
while ($reg=mysql_fetch_array($registros))
{
    $cad.="{'codigo':'".$reg['codigo']."'"},";
    $cad.="{'descripcion':'".$reg['descripcion']."'"},";
    $cad.="{'precio':'".$reg['precio']."'"},";
}
$cad.=']';
echo $cad;
```

Esta metodología es engorrosa y propensa a errores.

Luego el archivo JavaScript es:

```
addEvent(window,'load',inicializarEventos,false);

function inicializarEventos()
{
    var ob=document.getElementById('boton1');
    addEvent(ob,'click',presionBoton,false);
}

var conexion1;
function presionBoton(e)
{
    conexion1=crearXMLHttpRequest();
    conexion1.onreadystatechange = procesarEventos;
    conexion1.open('GET','pagina1.php', true);
    conexion1.send(null);
}

function procesarEventos()
{
    var resultados = document.getElementById("resultados");
    if(conexion1.readyState == 4)
    {
        alert('Cadena en formato JSON: '+conexion1.responseText);

        var datos=eval("(" + conexion1.responseText + ")");
        var salida="";
        for(f=0;f<datos.length;f++)
        {
            salida += 'Codigo:'+datos[f].codigo+"<br>";
            salida += 'Descripcion:'+datos[f].descripcion+"<br>";
            salida += 'Precio:'+datos[f].precio+"<br><br>";
        }
        resultados.innerHTML = salida;
    }
    else
    {
        resultados.innerHTML = "Cargando...";
    }
}
```

```

//*****
//Funciones comunes a todos los problemas
//*****
function addEvent(elemento,nomevento,funcion,captura)
{
    if (elemento.attachEvent)
    {
        elemento.attachEvent('on'+nomevento,funcion);
        return true;
    }
    else
    if (elemento.addEventListener)
    {
        elemento.addEventListener(nomevento,funcion,captura);
        return true;
    }
    else
        return false;
}

function crearXMLHttpRequest()
{
    var xmlhttp=null;
    if (window.ActiveXObject)
        xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
    else
        if (window.XMLHttpRequest)
            xmlhttp = new XMLHttpRequest();
    return xmlhttp;
}

```

La función procesarEventos rescata los datos enviados por el servidor y los muestra inicialmente tal como llegan en una ventana mediante el comando alert:

```

alert('Cadena en formato JSON: '+conexion1.responseText);

```

Podemos ver que se trata de un string JSON correctamente formado.

Ahora sí mediante el comando eval procedemos a generar un vector JavaScript y mostramos los datos dentro de un div:

```

var datos=eval("(" + conexion1.responseText + ")");
var salida='';
for(f=0;f<datos.length;f++)
{
    salida += 'Codigo: '+datos[f].codigo+"<br>";
    salida += 'Descripcion: '+datos[f].descripcion+"<br>";
    salida += 'Precio: '+datos[f].precio+"<br><br>";
}
resultados.innerHTML = salida;

```

Para este problema posiblemente es más fácil generar un trozo de HTML en el servidor y en el navegador solo mostrarlo, pero hay muchas situaciones que necesitamos recuperar una estructura de datos del servidor y proceder a su procesamiento en el navegador.

22 - De JavaScript a JSON (utilizando la librería json.js)

Ahora veremos otra librería, pero esta librería es en JavaScript, la misma nos permite convertir un objeto o array de JavaScript en formato JSON. Así de modo inverso nos permite generar un objeto o array JavaScript a partir de un string JSON y esto de modo seguro.

Esta librería se encuentra en [json.js](#)

Solo debemos descargarla y agregarla en nuestra página. Para poder probarlos en este sitio yo la incorporé en una carpeta que se encuentra inmediatamente en el nivel superior donde se almacenan las páginas que está probando. Es decir que para hacer uso en estos ejemplos debemos tipear:

```
<script src="../../json.js" language="JavaScript"></script>
```

La primera función que veremos es la que nos permite convertir un objeto o array a una cadena JSON. Para esto implementaremos un pequeño ejercicio donde al presionar un botón definimos un objeto que almacena tres atributos, de los cuales uno es una array. Luego convertimos dicho objeto a formato JSON con un método contenido en la librería json.js.

pagina1.html:

```
<html>
<head>
<title>Problema</title>
<script src="../../json.js" language="JavaScript"></script>
<script src="funciones.js" language="JavaScript"></script>
</head>
<body>
<h2>Convertir un objeto JavaScript a formato JSON</h2>
<input type="button" value="Convertir" id="boton1">
</body>
</html>
```

Es importante notar que hemos insertado dos archivos externos codificados en JavaScript:

```
<script src="../../json.js" language="JavaScript"></script>
<script src="funciones.js" language="JavaScript"></script>
```

Nuestro archivo con las funciones en JavaScript (funciones.js):

```
addEvent(window,'load',inicializarEventos,false);

function inicializarEventos()
{
    var ref=document.getElementById('boton1');
    addEvent(ref,'click',mostrarConversion,false);
}

function mostrarConversion(e)
{
    var obj={
        nombre:'juan',
```

```

    edad:25,
    sueldos:[1200,1700,1990]
  };

  var cadena=obj.toJSONString();
  alert(cadena);
}

//*****
//Funciones comunes a todos los problemas
//*****
function addEvent(elemento,nomevento,funcion,captura)
{
  if (elemento.attachEvent)
  {
    elemento.attachEvent('on'+nomevento,funcion);
    return true;
  }
  else
  if (elemento.addEventListener)
  {
    elemento.addEventListener(nomeevento,funcion,captura);
    return true;
  }
  else
    return false;
}

```

Lo más importante lo podemos encontrar en la función mostrarConversion:

```

function mostrarConversion(e)
{
  var obj={
    nombre:'juan',
    edad:25,
    sueldos:[1200,1700,1990]
  };

  var cadena=obj.toJSONString();
  alert(cadena);
}

```

Primero definimos un objeto en JavaScript:

```

var obj={
  nombre:'juan',
  edad:25,
  sueldos:[1200,1700,1990]
};

```

El objeto obj contiene tres atributos (nombre,edad y sueldo), de los cuales el último es un array con tres elementos.

Luego simplemente llamando al método toJSONString retorna un string con el contenido del objeto pero codificado en formato JSON:

```
var cadena=obj.toJSONString();  
alert(cadena);
```

23 - De JSON a JavaScript (utilizando la librería json.js)

Habíamos visto que para convertir una cadena que contiene información en formato JSON solo debíamos utilizar la función eval de JavaScript. Esto lo podemos seguir utilizando siempre y cuando la información del archivo JSON provenga de nuestro sitio.

Tengamos en cuenta si el archivo JSON proviene de un sitio que no tenemos la seguridad que nos envíe solo objetos y array literales, nuestro sitio está en peligro, ya que eval también ejecutará funciones en JavaScript si el archivo JSON las tiene.

Hay una solución para el problema que hemos planteado y se encuentra en utilizar un método contenido en la librería json.js que verifica que el archivo JSON solo contenga datos.

Habíamos resuelto este problema con el siguiente código:

```
var datos=eval("(" + conexion1.responseText + ")");
```

Si utilizamos la librería json.js debemos hacer:

```
var datos=conexion1.responseText.parseJSON();
```

Resolveremos el mismo problema sobre la base de datos de periféricos. Luego en el navegador emplearemos la librería json.js para convertir el archivo JSON a JavaScript.

pagina1.html

```
<html>  
<head>  
<title>Problema</title>  
<script src="../json.js" language="JavaScript"></script>  
<script src="funciones.js" language="JavaScript"></script>  
</head>  
<body>  
<h2>Recuperar datos del servidor almacenados en una base de datos en formato  
JSON utilizando la librería JSON.php y en el cliente utilizando la librería  
json.js</h2>  
<br>  
<input type="button" value="Recuperar" id="boton1">  
<div id="resultados"></div>  
</body>  
</html>
```

Es importante importar la librería json.js:

```
<script src="../json.js" language="JavaScript"></script>  
<script src="funciones.js" language="JavaScript"></script>
```

```

addEvent(window,'load',inicializarEventos,false);

function inicializarEventos()
{
    var ob=document.getElementById('boton1');
    addEvent(ob,'click',presionBoton,false);
}

var conexion1;
function presionBoton(e)
{
    conexion1=crearXMLHttpRequest();
    conexion1.onreadystatechange = procesarEventos;
    conexion1.open('GET','pagina1.php', true);
    conexion1.send(null);
}

function procesarEventos()
{
    var resultados = document.getElementById("resultados");
    if(conexion1.readyState == 4)
    {
        alert('Cadena en formato JSON: '+conexion1.responseText);

        var datos=conexion1.responseText.parseJSON();
        var salida="";
        for(f=0;f<datos.length;f++)
        {
            salida += 'Codigo: '+datos[f].codigo+"<br>";
            salida += 'Descripcion: '+datos[f].descripcion+"<br>";
            salida += 'Precio: '+datos[f].precio+"<br><br>";
        }
        resultados.innerHTML = salida;
    }
    else
    {
        resultados.innerHTML = "Cargando...";
    }
}

//*****
//Funciones comunes a todos los problemas
//*****
function addEvent(elemento,nomevento,funcion,captura)
{
    if (elemento.attachEvent)
    {
        elemento.attachEvent('on'+nomevento,funcion);
        return true;
    }
    else
    {
        if (elemento.addEventListener)
        {

```

```

        elemento.addEventListener(nomevento,funcion,captura);
        return true;
    }
    else
        return false;
}

function crearXMLHttpRequest()
{
    var xmlHttp=null;
    if (window.ActiveXObject)
        xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
    else
        if (window.XMLHttpRequest)
            xmlHttp = new XMLHttpRequest();
    return xmlHttp;
}

```

En la función procesar eventos debemos llamar al método parseJSON:

```
var datos=conexion1.responseText.parseJSON();
```

Esto lo podemos hacer ya que la librería añade esta funcionalidad a los string.

el archivo pagina1.php no tiene cambios:

```

<?php
$conexion=mysql_connect("localhost","root","z80")
or die("Problemas en la conexion");
mysql_select_db("bdajax",$conexion) or
die("Problemas en la seleccion de la base de datos");

$registros=mysql_query("select codigo,descripcion,precio from perifericos",
$conexion)
or die("Problemas en el select".mysql_error());

while ($reg=mysql_fetch_array($registros))
{
    $vec[]=$reg;
}

require('../JSON.php');
$json=new Services_JSON();
$cad=$json->encode($vec);
echo $cad;
?>

```

24 - De JSON a PHP (utilizando la librería JSON.php)

Ahora nos queda ver como en el servidor recibir datos con formato JSON y proceder a generar una clase en PHP mediante la librería JSON.php.

La sintaxis es:

```
$json=new Services_JSON();
$cad=$json->decode(stripslashes($_REQUEST['cadena']));
echo 'Nombre:'. $cad->nombre;
...
...
```

Es decir la clase Services_JSON tiene un método llamado decode que recibe como parámetro una cadena con datos codificados en JSON y procede a retornar la información en un objeto.

Confeccionaremos un problema que envíe desde el navegador información en formato JSON y en el servidor la convertiremos en un objeto de PHP y procederemos a generar un trozo de HTML que retornaremos al navegador para que lo muestre.

El archivo pagina1.html

```
<html>
<head>
<title>Problema</title>
<script src="../json.js" language="JavaScript"></script>
<script src="funciones.js" language="JavaScript"></script>
</head>
<body>
<h2>Enviar datos desde el navegador en formato JSON y proceder a la
decodificación en una clase PHP con la librería JSON.php</h2>
<input type="button" value="Enviar" id="boton1">
<div id="resultados"></div>
</body>
</html>
```

El archivo con las funciones JavaScript es:

```
addEvent(window,'load',inicializarEventos,false);

function inicializarEventos()
{
    var ref=document.getElementById('boton1');
    addEvent(ref,'click',botonPresionado,false);
}

function botonPresionado(e)
{
    var obj={
        nombre:'juan',
        edad:25,
        sueldos:[1200,1700,1990]
    };
    var cadena=obj.toJSONString();
    enviarDatos(cadena);
}

var conexion1;
function enviarDatos(cadena)
{

```

```

    conexion1=crearXMLHttpRequest();
    conexion1.onreadystatechange = procesarEventos;
    conexion1.open('GET','pagina1.php?cadena='+cadena, true);
    conexion1.send(null);
}

function procesarEventos()
{
    var resultados = document.getElementById("resultados");
    if(conexion1.readyState == 4)
    {
        resultados.innerHTML = conexion1.responseText;
    }
    else
    {
        resultados.innerHTML = "Cargando...";
    }
}

//*****
//Funciones comunes a todos los problemas
//*****
function addEvent(elemento,nomevento,funcion,captura)
{
    if (elemento.attachEvent)
    {
        elemento.attachEvent('on'+nomevento,funcion);
        return true;
    }
    else
    if (elemento.addEventListener)
    {
        elemento.addEventListener(nomeevento,funcion,captura);
        return true;
    }
    else
        return false;
}

function crearXMLHttpRequest()
{
    var xmlhttp=null;
    if (window.ActiveXObject)
        xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
    else
        if (window.XMLHttpRequest)
            xmlhttp = new XMLHttpRequest();
    return xmlhttp;
}

```

Cuando se presiona el botón procedemos a partir de un objeto JavaScript generar una cadena en formato JSON, luego procedemos a llamar a la función enviarDatos:

```

var obj={
    nombre:'juan',
    edad:25,

```

```

    sueldos:[1200,1700,1990]
};
var cadena=obj.toJSONString();
enviarDatos(cadena);

```

La función `enviarDatos` procede a crear un objeto de la clase `XMLHttpRequest` y pasa mediante el parámetro `GET` la cadena en formato JSON:

```

var conexion1;
function enviarDatos(cadena)
{
    conexion1=crearXMLHttpRequest();
    conexion1.onreadystatechange = procesarEventos;
    conexion1.open('GET','pagina1.php?cadena='+cadena, true);
    conexion1.send(null);
}

```

Solo queda mostrar el trozo de datos en HTML que retorna el servidor:

```

function procesarEventos()
{
    var resultados = document.getElementById("resultados");
    if(conexion1.readyState == 4)
    {
        resultados.innerHTML = conexion1.responseText;
    }
    else
    {
        resultados.innerHTML = "Cargando...";
    }
}

```

Lo nuevo se centra en el archivo `pagina1.php`:

```

<?php
require('../JSON.php');
$json=new Services_JSON();
$cad=$json->decode(stripslashes($_REQUEST['cadena']));
echo 'Nombre:'. $cad->nombre;
echo '<br>';
echo 'Edad:'. $cad->edad;
echo '<br>';
echo 'Primer sueldo:'. $cad->sueldos[0];
echo '<br>';
echo 'Segundo sueldo:'. $cad->sueldos[1];
echo '<br>';
echo 'Tercer sueldo:'. $cad->sueldos[2];
?>

```

El método que convierte una cadena con formato JSON en un objeto de PHP es:

```

$cad=$json->decode(stripslashes($_REQUEST['cadena']));

```

Recordemos que con la función `stripslashes` sacamos los caracteres de escape (barras invertidas para las comillas entre otras)

Luego podemos fácilmente acceder a los atributos del objeto con la sintaxis:

```

echo 'Nombre:'. $cad->nombre;
echo '<br>';
echo 'Edad:'. $cad->edad;
echo '<br>';
echo 'Primer sueldo:'. $cad->suellos[0];
echo '<br>';
echo 'Segundo sueldo:'. $cad->suellos[1];
echo '<br>';
echo 'Tercer sueldo:'. $cad->suellos[2];

```

25 - Pizarra interactiva multiusuario

El último ejemplo que implementaré y utilizaré JSON para la comunicación entre el cliente y el servidor será una "pizarra interactiva multiusuario", básicamente desarrollaremos una aplicación que muestre un tablero con letras que se puedan mover con el mouse. Lo interesante será que cada un cierto tiempo nos comunicaremos con el servidor e informaremos las letras que se han desplazado dentro de la ventana, esto permitirá que cualquier otro usuario que esté ejecutando en ese momento la misma página verá el desplazamiento que efectuó otra persona.

Para probar si realmente funciona esta característica podemos ejecutar el "problema resuelto" utilizando el FireFox y el Internet Explorer. Podremos observar como se sincronizan las posiciones de las letras dentro de la ventana (si un usuario mueve una letra hacia la derecha, luego de algunos segundos todos los otros usuarios verán reflejado el cambio en sus navegadores).

Veamos los distintos archivos que intervienen (pagina1.html):

```

<html>
<head>
<title>Problema</title>
<script src="../../json.js" language="JavaScript"></script>
<script src="funciones.js" language="JavaScript"></script>
</head>
<body style="background: #eee">
<div><strong>Puede desplazar las letras con el mouse para escribir
palabras que serán vistas por otros usuarios que visiten la página
en este momento o más tarde.</strong></div>
<div id="letras"></div>
</body>
</html>

```

Hay que tener en cuenta que emplearemos JSON para la transferencia de datos entre el cliente y el servidor por lo que debemos disponer:

```

<script src="../../json.js" language="JavaScript"></script>

```

Este archivo no tiene nada de especial toda la complejidad se encuentra en el archivo funciones.js que lo incorporamos con la siguiente línea:

```

<script src="funciones.js" language="JavaScript"></script>

```

Ahora el archivo donde se encuentra toda la complejidad del código que se ejecuta en el cliente está en funciones.js:

```

addEvent(window,'load',inicializarEventos,false);

function desactivarSeleccion(e)
{
    return false
}

var conexion1;
function inicializarEventos()
{
    document.onmousedown=desactivarSeleccion;
    document.onmousemove=desactivarSeleccion;

    conexion1=crearXMLHttpRequest();
    conexion1.onreadystatechange = procesarEventos;
    conexion1.open('GET','pagina1.php', true);
    conexion1.send(null);
}

var datos;
var datosNuevos;
var datosMovil;
var vectorLetras=new Array();
var reloj=null;
var relojGeneral=null;
var pasos=0;
function procesarEventos()
{
    if(conexion1.readyState == 4)
    {
        datos=conexion1.responseText.parseJSON();
        crearLetras();
        relojGeneral=window.setInterval(actualizarCoordenadas, 5000);
    }
}

function crearLetras()
{
    for(f=0;f<datos.length;f++)
    {
        var ob=document.createElement('div');
        ob.style.left=datos[f].x+'px';
        ob.style.top=datos[f].y+'px';
        ob.style.width='17px';
        ob.style.height='17px';
        ob.style.background='#eee';
        ob.style.position='absolute';
        ob.style.fontSize='18px';
        ob.style.padding='3px';
        ob.style.cursor='pointer';
        ob.id='div'+f;
        ob.style.textAlign='center';
        var x=document.getElementById('letras');
        x.appendChild(ob);
    }
}

```

```

var ref=document.getElementById('div'+f);
ref.innerHTML=datos[f].letra;
vectorLetras[f]=new Recuadro(ob,datos[f].letra,f+1,datos[f].x,datos[f].y);
}
}

function letrasMovidas(cod,x,y) {
this.codigo=cod;
this.x=x;
this.y=y;
}

function actualizarCoordenadas()
{
var let=new Array();
var con=0;
for(f=0;f<vectorLetras.length;f++)
{
if (datos[f].x!=vectorLetras[f].retornarX() ||
    datos[f].y!=vectorLetras[f].retornarY())
{
datos[f].x=vectorLetras[f].retornarX();
datos[f].y=vectorLetras[f].retornarY();
let[con]=new
    letrasMovidas(datos[f].codigo,vectorLetras[f].retornarX(),vectorLetras[f].re
tornarY());
con++;
}
}
var aleatorio=Math.random();
var cadena=let.toJSONString();
conexion1=crearXMLHttpRequest();
conexion1.onreadystatechange = procesarEventosContinuos;
conexion1.open('GET','pagina2.php?letras='+cadena+"&aleatorio="+aleatorio,
true);
conexion1.send(null);
}

function procesarEventosContinuos()
{
if(conexion1.readyState == 4)
{
datosNuevos=conexion1.responseText.parseJSON();
datosMovil=conexion1.responseText.parseJSON();

var cambios=false;
for(f=0;f<datosNuevos.length;f++)
{
if (datosNuevos[f].x!=datos[f].x ||
    datosNuevos[f].y!=datos[f].y)
{
datosMovil[f].x=datos[f].x;
datosMovil[f].y=datos[f].y;
cambios=true;
}
}
}
}

```

```

    if (cambios)
    {
        if (reloj==null)
            reloj=window.setInterval(moverLetras, 5);
        clearInterval(relojGeneral);
        pasos=20;
    }
}

function moverLetras()
{
    var cambios=false;
    pasos--;
    for(f=0;f<datosNuevos.length;f++)
    {
        if (datosNuevos[f].x!=datos[f].x ||
            datosNuevos[f].y!=datos[f].y)
        {
            cambios=true;
            var dx=Math.abs(datosNuevos[f].x-datos[f].x);
            var avancex;
            if ((datosNuevos[f].x-datos[f].x)>0)
                avancex=Math.round(dx/20);
            else
                avancex=Math.round(-dx/20);
            datosMovil[f].x=parseInt(datosMovil[f].x)+avancex;

            var dy=Math.abs(datosNuevos[f].y-datos[f].y);
            var avancey;
            if ((datosNuevos[f].y-datos[f].y)>0)
                avancey=Math.round(dy/20);
            else
                avancey=Math.round(-dy/20);
            datosMovil[f].y=parseInt(datosMovil[f].y)+avancey;

            cambios=true;
            if (pasos==0)
            {
                vectorLetras[f].fijarX(datosNuevos[f].x);
                vectorLetras[f].fijarY(datosNuevos[f].y);
            }
            else
            {
                vectorLetras[f].fijarX(datosMovil[f].x);
                vectorLetras[f].fijarY(datosMovil[f].y);
            }
        }
    }
    if (pasos==0)
    {
        clearInterval(reloj);
        reloj=null;
        relojGeneral=window.setInterval(actualizarCoordenadas, 5000);
    }
}

```

```

//Drag and Drop

Recuadro=function(div)
{
    tX=0;
    tY=0;
    difX=0;
    difY=0;
    addEvent(div,'mousedown',inicioDrag,false);

    function coordenadaX(e)
    {
        if (window.event)
            return event.clientX+document.body.scrollTop;
        else
            return e.pageX;
    }

    function coordenadaY(e)
    {
        if (window.event)
            return event.clientY+document.body.scrollTop;
        else
            return e.pageY;
    }

    function inicioDrag(e)
    {
        if (window.event)
            e=window.event;
        var eX=coordenadaX(e);
        var eY=coordenadaY(e);
        var oX=parseInt(div.style.left);
        var oY=parseInt(div.style.top);
        difX=oX-eX;
        difY=oY-eY;
        addEvent(document,'mousemove',drag,false);
        addEvent(document,'mouseup',soltar,false);

    }

    function drag(e)
    {
        if (window.event)
            e=window.event;
        tX=coordenadaY(e)+difY+'px';
        tY=coordenadaX(e)+difX+'px'
        div.style.top=tX;
        div.style.left=tY;
    }

    function soltar(e)

```

```

{
  if (window.event)
  {
    document.detachEvent('onmousemove',drag);
    document.detachEvent('onmouseup',soltar);
  }
  else
  {
    document.removeEventListener('mousemove',drag,false);
    document.removeEventListener('mouseup',soltar,false);
  }
  actualizarCoordenadas();
}

this.retornarX=function()
{
  return parseInt(div.style.left);
}

this.retornarY=function()
{
  return parseInt(div.style.top);
}

this.fijarX=function(xx)
{
  div.style.left=xx+'px';
}

this.fijarY=function(yy)
{
  div.style.top=yy+'px';
}
}

//*****
//Funciones comunes a todos los problemas
//*****
function addEvent(elemento,nomevento,funcion,captura)
{
  if (elemento.attachEvent)
  {
    elemento.attachEvent('on'+nomevento,funcion);
    return true;
  }
  else
  if (elemento.addEventListener)
  {
    elemento.addEventListener(nomeevento,funcion,captura);
    return true;
  }
  else
    return false;
}

```

```
function crearXMLHttpRequest()
{
    var xmlhttp=null;
    if (window.ActiveXObject)
        xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
    else
        if (window.XMLHttpRequest)
            xmlhttp = new XMLHttpRequest();
    return xmlhttp;
}
```

La primera función que se ejecuta es inicializarEventos donde tenemos:

```
document.onmousedown=desactivarSeleccion;
document.onmousemove=desactivarSeleccion;
```

Con estas dos asignaciones desactivamos la posibilidad de seleccionar texto dentro de la página, esto es para que no se puedan seleccionar las letras.

Luego:

```
conexion1=crearXMLHttpRequest();
conexion1.onreadystatechange = procesarEventos;
conexion1.open('GET','paginal.php', true);
conexion1.send(null);
```

creamos un objeto de la clase crearXMLHttpRequest para recuperar la posición de cada letra. Veremos más adelante que tenemos una base de datos donde almacenamos las letras y las coordenadas de cada una.

La función procesarEventos:

```
function procesarEventos()
{
    if(conexion1.readyState == 4)
    {
        datos=conexion1.responseText.parseJSON();
        crearLetras();
        relojGeneral=window.setInterval(actualizarCoordenadas, 5000);
    }
}
```

cuando los datos se han enviado por completo del servidor procedemos a rescatarlos y generar un objeto literal en JavaScript llamando a la función parseJSON(). Llamamos seguidamente a la función crearLetras() y finalmente creamos un timer o alarma para que se dispare cada 5 segundos, veremos luego que tiene por objetivo recuperar las coordenadas de las letras almacenadas en el servidor:

```
relojGeneral=window.setInterval(actualizarCoordenadas, 5000);
```

La función crearLetras:

```
function crearLetras()
{
    for (f=0;f<datos.length;f++)
    {
```

```

var ob=document.createElement('div');
ob.style.left=datos[f].x+'px';
ob.style.top=datos[f].y+'px';
ob.style.width='17px';
ob.style.height='17px';
ob.style.background='#eee';
ob.style.position='absolute';
ob.style.fontSize='18px';
ob.style.padding='3px';
ob.style.cursor='pointer';
ob.id='div'+f;
ob.style.textAlign='center';
var x=document.getElementById('letras');
x.appendChild(ob);
var ref=document.getElementById('div'+f);
ref.innerHTML=datos[f].letra;
vectorLetras[f]=new
Recuadro(ob,datos[f].letra,f+1,datos[f].x,datos[f].y);
}
}

```

crea elementos HTML de tipo "div" y los dispone en las coordenadas que acabamos de recuperar del servidor:

```

ob.style.left=datos[f].x+'px';
ob.style.top=datos[f].y+'px';

```

El ancho y el alto son fijos:

```

ob.style.width='17px';
ob.style.height='17px';

```

Definimos un id distinto a cada uno:

```

ob.id='div'+f;

```

Por último lo añadimos a la página:

```

ref.innerHTML=datos[f].letra;
vectorLetras[f]=new
Recuadro(ob,datos[f].letra,f+1,datos[f].x,datos[f].y);

```

y creamos un objeto de la clase Recuadro que nos permitirá desplazarlo con el mouse (esta clase se estudió en el curso de DHTML Ya.

La función actualizarCoordenadas se dispara cada 5 segundos o inmediatamente después que un usuario desplaza una letra en la pantalla:

```

function actualizarCoordenadas()
{
var let=new Array();
var con=0;
for (f=0;f<vectorLetras.length;f++)
{
if (datos[f].x!=vectorLetras[f].retornarX() ||
datos[f].y!=vectorLetras[f].retornarY())
{
datos[f].x=vectorLetras[f].retornarX();

```

```

        datos[f].y=vectorLetras[f].retornarY();
        let[con]=new
            letrasMovidas(datos[f].codigo,vectorLetras[f].retornarX(),vec
torLetras[f].retornarY());
        con++;
    }
}
var aleatorio=Math.random();
var cadena=let.toJSONString();
conexion1=crearXMLHttpRequest();
conexion1.onreadystatechange = procesarEventosContinuos;
conexion1.open('GET','pagina2.php?
letras='+cadena+"&aleatorio="+aleatorio, true);
conexion1.send(null);
}

```

Dentro del for identificamos si alguna de las letras fue desplazada con el mouse:

```

if (datos[f].x!=vectorLetras[f].retornarX() ||
    datos[f].y!=vectorLetras[f].retornarY())

```

En caso afirmativo actualizamos la estructura datos:

```

    datos[f].x=vectorLetras[f].retornarX();
    datos[f].y=vectorLetras[f].retornarY();

```

y además creamos una componente de la clase letrasMoviles:

```

        let[con]=new
            letrasMovidas(datos[f].codigo,vectorLetras[f].retornarX(),vec
torLetras[f].retornarY());

```

Este vector let tiene los cambios efectuados en pantalla para ser enviados al servidor.

Fuera del for creamos un objeto de la clase XMLHttpRequest y procedemos a enviar los datos al servidor:

```

    conexion1.open('GET','pagina2.php?
letras='+cadena+"&aleatorio="+aleatorio, true);

```

Recordemos que para convertir el vector de JavaScript a JSON lo hacemos:

```

    var cadena=let.toJSONString();

```

La función procesarEventosContinuos:

```

function procesarEventosContinuos()
{
    if(conexion1.readyState == 4)
    {
        datosNuevos=conexion1.responseText.parseJSON();
        datosMovil=conexion1.responseText.parseJSON();

        var cambios=false;
        for(f=0;f<datosNuevos.length;f++)
        {
            if (datosNuevos[f].x!=datos[f].x ||
                datosNuevos[f].y!=datos[f].y)

```

```

        {
            datosMovil[f].x=datos[f].x;
            datosMovil[f].y=datos[f].y;
            cambios=true;
        }
    }
    if (cambios)
    {
        if (reloj==null)
            reloj=window.setInterval(moverLetras, 5);
        clearInterval(relojGeneral);
        pasos=20;
    }
}
}

```

Recupera las coordenadas actuales de las letras que se encuentran registradas en el servidor:

```

datosNuevos=conexion1.responseText.parseJSON();
datosMovil=conexion1.responseText.parseJSON();

```

Utilizamos dos variables ya que una la utilizaremos para ir desplazando lentamente la letra por la pantalla.

Dentro de un for verificamos si hay coordenadas distintas entre las que administra nuestro navegador y las registradas en el servidor:

```

    if (datosNuevos[f].x!=datos[f].x ||
        datosNuevos[f].y!=datos[f].y)
    {

```

En caso de haber diferencias:

```

    if (cambios)
    {
        if (reloj==null)
            reloj=window.setInterval(moverLetras, 5);
        clearInterval(relojGeneral);
        pasos=20;
    }

```

desactivamos el timer relojGeneral y activamos un timer para desplazar lentamente las letras entre la posición actual y la registrada en el servidor (la función moverLetras se dispara cada 5 milisegundos.

La función moverLetra:

```

function moverLetras()
{
    var cambios=false;
    pasos--;
    for(f=0;f<datosNuevos.length;f++)
    {
        if (datosNuevos[f].x!=datos[f].x ||
            datosNuevos[f].y!=datos[f].y)
        {
            cambios=true;
            var dx=Math.abs(datosNuevos[f].x-datos[f].x);
            var avancex;

```

```

        if ((datosNuevos[f].x-datos[f].x)>0)
            avancex=Math.round(dx/20);
        else
            avancex=Math.round(-dx/20);
        datosMovil[f].x=parseInt(datosMovil[f].x)+avancex;

        var dy=Math.abs(datosNuevos[f].y-datos[f].y);
        var avancey;
        if ((datosNuevos[f].y-datos[f].y)>0)
            avancey=Math.round(dy/20);
        else
            avancey=Math.round(-dy/20);
        datosMovil[f].y=parseInt(datosMovil[f].y)+avancey;

        cambios=true;
        if (pasos==0)
        {
            vectorLetras[f].fijarX(datosNuevos[f].x);
            vectorLetras[f].fijarY(datosNuevos[f].y);
        }
        else
        {
            vectorLetras[f].fijarX(datosMovil[f].x);
            vectorLetras[f].fijarY(datosMovil[f].y);
        }
    }
}
if (pasos==0)
{
    clearInterval(reloj);
    reloj=null;
    relojGeneral=window.setInterval(actualizarCoordenadas, 5000);
}
}

```

desplaza las letras que han cambiado de posición. Esta función se ejecuta 20 veces hasta que la variable global pasos almacene el valor 0.

Luego tenemos los dos archivos que se ejecutan en el servidor (pagina1.php):

```

<?php
$conexion=mysql_connect("localhost","root","z80")
or die("Problemas en la conexion");
mysql_select_db("bdajax",$conexion) or die("Problemas en la seleccion
de la base de datos");
$registros=mysql_query("select letra,x,y,codigo from letras",$conexion)
or die("Problemas en el select".mysql_error());

while ($reg=mysql_fetch_array($registros))
{
    $vec[]=$reg;
}
mysql_close($conexion);
require('../JSON.php');
$json=new Services_JSON();
$cad=$json->encode($vec);

```

```
echo $cad;  
?>
```

Recupera de la tabla letras las coordenadas y letras propiamente dichas que serán mostradas en el servidor:

```
$registros=mysql_query("select letra,x,y,codigo from letras",  
$conexion)  
or die("Problemas en el select".mysql_error());
```

Guardamos los datos en un vector:

```
while ($reg=mysql_fetch_array($registros))  
{  
    $vec[]=$reg;  
}
```

Generamos un archivo con formato JSON para que se envíe al cliente:

```
require('../JSON.php');  
$json=new Services_JSON();  
$cad=$json->encode($vec);  
echo $cad;
```

Por último nos queda el archivo que llamamos cada 5 segundos para indicarle las novedades dentro del navegador (si el usuario desplazó alguna letra) y recuperar las novedades registradas en el servidor:

```
<?php  
require('../JSON.php');  
$json=new Services_JSON();  
$cad=$json->decode(stripSlashes($_REQUEST['letras']));  
$conexion=mysql_connect("localhost","root","z80")  
or die("Problemas en la conexion");  
mysql_select_db("bdajax",$conexion) or die("Problemas en la seleccion  
de la base de datos");  
$registros=mysql_query("select letra,x,y,codigo from letras",$conexion)  
or die("Problemas en el select".mysql_error());  
for($f=0;$f<count($cad);$f++)  
{  
    mysql_query("update letras set x=".$cad[$f]->x.",y=".$cad[$f]->y."  
    where codigo=".$cad[$f]->codigo,$conexion) or die("Problemas en el  
select".mysql_error());  
}  
$registros=mysql_query("select x,y,codigo from letras",$conexion) or  
die("Problemas en el select".mysql_error());  
while ($reg=mysql_fetch_array($registros))  
{  
    $vec[]=$reg;  
}  
mysql_close($conexion);  
$json=new Services_JSON();  
$cad=$json->encode($vec);  
echo $cad;  
?>
```

Primero recuperamos los datos enviados por el navegador y generamos un vector asociativo en PHP a partir de los datos que llegan en formato JSON:

```
$cad=$json->decode(stripSlashes($_REQUEST['letras']));
```

Modificamos las coordenadas de las letras:

```
$registros=mysql_query("select letra,x,y,codigo from letras",
$conexion)
or die("Problemas en el select".mysql_error());
for($f=0;$f<count($cad);$f++)
{
    mysql_query("update letras set x=".$cad[$f]->x.",y=".$cad[$f]->y."
        where codigo=".$cad[$f]->codigo,$conexion) or die("Problemas en el
        select".mysql_error());
}
```

Por último recuperamos todas las letras y sus coordenadas y las enviamos nuevamente al cliente (navegador) que las solicitó:

```
$registros=mysql_query("select x,y,codigo from letras",$conexion) or
die("Problemas en el select".mysql_error());
while ($reg=mysql_fetch_array($registros))
{
    $vec[]=$reg;
}
mysql_close($conexion);
$json=new Services_JSON();
$cad=$json->encode($vec);
echo $cad;
```