ANEXO IV AJAX Y JQUERY

Funciones JQuery para AJAX

Las funciones y utilidades relacionadas con AJAX son parte fundamental de jQuery. El método principal para realizar peticiones AJAX es **$.ajax()** . A partir de esta función básica, se han definido otras funciones relacionadas, de más alto nivel y especializadas en tareas concretas: **$.get(), $.post(), $.load()**, etc.

La sintaxis de **$.ajax()** es muy sencilla:

```
$.ajax(opciones);
```

A continuación se muestra un ejemplo básico realizado con $.ajax():

```
$.ajax({
  url: '/ruta/hasta/pagina.php',
  type: 'POST',
  async: true,
  data: 'parametro1=valor1&parametro2=valor2',
  success: procesaRespuesta,
  error: muestraError
});
```

La siguiente tabla muestra todas las **opciones** que se pueden definir para el método $.ajax():

Opción	**Descripción**
Async	**Indica si la petición es asíncrona.** Su valor por defecto es true, el habitual para las peticiones AJAX
beforeSend	**Permite indicar una función que modifique el objeto XMLHttpRequest antes de realizar la petición**. El propio objeto XMLHttpRequest se pasa como único argumento de la función
complete	**Permite establecer la función que se ejecuta cuando una petición se ha completado** (y después de ejecutar, si se han establecido, las funciones de success o error). La función recibe el objeto XMLHttpRequest como primer parámetro y el resultado de la petición como segundo argumento
contentType	**Indica el valor de la cabecera Content-Type utilizada para realizar la petición.** Su valor por defecto es application/x-www-form-urlencoded

data	**Información que se incluye en la petición**. Se utiliza para enviar parámetros al servidor. Si es una cadena de texto, se envía tal cual, por lo que su formato debería ser parametro1=valor1&parametro2=valor2. También se puede indicar un array asociativo de pares clave/valor que se convierten automáticamente en una cadena tipo *query string*
dataType	**El tipo de dato que se espera como respuesta**. Si no se indica ningún valor, jQuery lo deduce a partir de las cabeceras de la respuesta. Los posibles valores son: **xml**(se devuelve un documento XML correspondiente al valor responseXML), **html**(devuelve directamente la respuesta del servidor mediante el valor responseText),**script** (se evalúa la respuesta como si fuera JavaScript y se devuelve el resultado) y **json** (se evalúa la respuesta como si fuera JSON y se devuelve el objeto JavaScript generado)
error	**Indica la función que se ejecuta cuando se produce un error durante la petición**. Esta función recibe el objeto XMLHttpRequest como primer parámetro, una cadena de texto indicando el error como segundo parámetro y un objeto con la excepción producida como tercer parámetro
ifModified	**Permite considerar como correcta la petición solamente si la respuesta recibida es diferente de la anterior respuesta**. Por defecto su valor es false
processData	**Indica si se transforman los datos de la opción data para convertirlos en una cadena de texto**. Si se indica un valor de false, no se realiza esta transformación automática
success	**Permite establecer la función que se ejecuta cuando una petición se ha completado de forma correcta**. La función recibe como **primer parámetro los datos recibidos del servidor**, previamente formateados según se especifique en la opción dataType, **el segundo parámetro es el estado y el tercero el objeto xmlhttpRequest**.
timeout	**Indica el tiempo máximo, en milisegundos, que la petición espera la respuesta del servidor antes de anular la petición**
type	**El tipo de petición que se realiza**. Su valor por defecto es GET, aunque también se puede utilizar el método POST
url	**La URL del servidor a la que se realiza la petición**

Además de la función $.ajax() genérica, existen varias funciones relacionadas que son versiones simplificadas y especializadas de esa función.

Método load()

El método load() es un método AJAX simple, pero potente. Carga los datos de un servidor y pone los datos devueltos en el seleccionado elemento.

Sintaxis:

```
$(selector).load (url, data, function(response,status,xhr));
```

Parámetros	Descripción
URL	Obligatorio. Especifica la URL que desea solicitar.
Data	Opcional. Especificar los datos que se quieren enviar al servidor con notación de objeto.
function(data,status,xhr)	Opcional. Especifica la function a ejecutar si la petición tiene exito. Los parámetros adicionales son: • *data* – contiene el resultado de la petición. • *status* – contiene el estado de la petición ("success", "notmodified", "error", "timeout", or "parsererror") • *xhr* – contiene el objeto XMLHttpRequest

Ejemplo:

Este es el contenido de nuestro archivo de ejemplo: "demo_test.txt":

```
<h2> jQuery y AJAX es diversión! </ h2>
<p id="p1"> Este es un texto de un párrafo. </ p>
```

//carga el contenido del archivo "demo_test.txt" en un elemento :

```
$("#div1").load ("demo_test.txt").;
```

También es posible añadir un selector de jQuery para el parámetro de URL.

El ejemplo siguiente carga el contenido del elemento con id = "p1", dentro del archivo "Demo_test.txt", en un elemento :

```
$("#div1").load ("demo_test.txt #p1");
```

El siguiente ejemplo muestra un alert después de que el método load() se completa. Si el método load() ha tenido éxito, se muestra "Contenido externo cargado correctamente!", y si no se muestra un mensaje de error:

```
$("button").click (function () {
$("#div1").load ("demo_test.txt", function (responseTxt, statusTxt, xhr) {
        if (statusTxt == "sucess")
                alert ("Contenido externo cargado satisfactoriamente");
        if (statusTxt == "error")
                alert ("Error:" + xhr.status + ":" + xhr.statusText);
        });
});
```

Método $.get()

Sintaxis

```
$.get(URL,data,function(data,status,xhr),dataType)
```

Parámetros	**Descripción**
URL	Obligatorio. Especifica la URL que desea solicitar.
Data	Opcional. En jQuery si queremos enviar datos, como variables en la URL, se especifican con la notación de objetos.
function(data,status,xhr)	Opcional. Especifica la function a ejecutar si la petición tiene exito. Los parámetros adicionales son: • *data* – contiene el resultado de la petición. • *status* – contiene el estado de la petición ("success", "notmodified", "error", "timeout", or "parsererror") • *xhr* – contiene el objeto XMLHttpRequest
dataType	Opcional. Especifica el tipo de datos que el servidor va a devolver. • "xml" - An XML document • "html" - HTML as plain text • "text" - A plain text string • "script" - Runs the response as JavaScript, and returns it as plain text • "json" - Runs the response as JSON, and returns a JavaScript object

Ejemplos

// Petición GET simple

```
$.get("contenido-ajax.html");
```

// Petición GET con función que se ejecuta cuando se procesa la respuesta.

```
$("button").click (function () {
        $.get ("demo_test.asp", function (data, status) {
                alert ("Date:" + data + "\ nStatus:" + status);
        });
});
```

//Petición con envío de datos

```
$.get("recibe-parametros2.php", {nombre: "Evandro", edad: "99"},
function(respuesta){
  $("#miparrafo").html(respuesta);
})
```

//Petición indicando el tipo de respuesta a obtener

```
$("#coniva").click(function(){
  $.get("recibe-parametros-devuelve-json.php", {pais: "ES", precio: 20},
muestraPrecioFinal, "json");
})
$("#siniva").click(function(){
  $.get("recibe-parametros-devuelve-json.php", {pais: "BR", precio: 300},
muestraPrecioFinal, "json");
})
```

Método $.post()

Las peticiones POST se realizan exactamente de la misma forma, por lo que sólo hay que cambiar$.get() por $.post(). La sintaxis de esta función es:

```
$(selector).post(URL,data,function(data,status,xhr),dataType)
```

Ejemplo

```
$("button").click (function () {
        $.post ("demo_test_post.asp",
        {
        nombre: "Pato Donald",
```

```
        ciudad: "Patolandia"
        },
        function (data, status) {
        alert ("Data:" + data + "\ nStatus:" + status);
        });
});
```

El primer parámetro de $.post() es la URL que queremos solicitar ("demo_test_post.asp"). Luego se pasa en algunos datos para enviar junto con la solicitud (nombre y ciudad). La secuencia de comandos ASP en "demo_test_post.asp" lee los parámetros, los procesa, y devuelve un resultado.

El tercer parámetro es una función de devolución de llamada. El primer parámetro de devolución de llamada tiene el contenido de la página solicitada, y el segundo parámetro de devolución de llamada tiene el estado de la solicitud.

El archivo "demo_test_post.asp" contiene:

```
<%
dim fname, city
fname = Request.Form ("name")
city = Request.Form ("city")
Response.Write ("Dear" & fname & ". ")
Response.Write ("Espero que vive bien en" & city & ".")
%>
```

Método $.getJSON()

Es otro método bastante utilizado que recibe una respuesta en JSON. El contenido del objeto nos lo debe enviar el servidor como respuesta, siempre en un JSON que jQuery de manera interna se encargará de interpretar y convertir en un objeto nativo de Javascript, al que podremos acceder para recuperar cualquiera de sus datos.

$(selector).getJSON(url,data,success(data,status,xhr))

Parámetros	Descripción
URL	Obligatorio. Especifica la URL que desea solicitar.
Data	Opcional. Especificar los datos que se quieren enviar al servidor con notación de objeto.
function(data,status,xhr)	Opcional. Especifica la function a ejecutar si la petición tiene exito. Los parámetros adicionales son: • *data* – contiene el resultado de la petición. • *status* – contiene el estado de la petición ("success", "notmodified", "error", "timeout", or "parsererror") • *xhr* – contiene el objeto XMLHttpRequest

Sería el equivalente a:

```
$.ajax({
  dataType: "json",
  url: url,
  data: data,
  success: success
});
```

Ejemplo:

```
$.getJSON( "ajax/test.json", function( data ) {
  var items = [];
  $.each( data, function( key, val ) {
    items.push( "<li id='" + key + "'>" + val + "</li>" );
  });

  $( "<ul/>", {
    "class": "my-new-list",
    html: items.join( "" )
  }).appendTo( "body" );
});
```

JSON

```
{
  "one": "Singular sensation",
  "two": "Beady little eyes",
  "three": "Little birds pitch by my doorstep"
}
```

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

pagina1.html

```
<html>
<head>
<title>Problema</title>
<link rel="StyleSheet" href="estilos.css" type="text/css">
<script type="text/javascript" src="../jquery.js"></script>
<script type="text/javascript" src="funciones.js"></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>
```

funciones.js

```
var x;
x=$(document);
x.ready(inicializarEventos);

function inicializarEventos()
{
  var x;
  x=$("#boton1");
  x.click(presionSubmit);
}

function presionSubmit()
{
  var v=$("#dni").attr("value");
  $.getJSON("pagina1.php",{dni:v},llegadaDatos);
  return false;
}

function llegadaDatos(datos)
{
 $("#resultados").html("Nombre:"+datos.nombre+
                  "<br>"+"Apellido:"+
                  datos.apellido+"<br>"+
                  "Direccion:"+datos.direccion);
```

```
}
```

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'
}";
?>
```