

TECNOLOGIA AJAX

El término de Ajax son las siglas de **ASÍNCRONO JAVASCRIPT XML** para indicar que no se trata solo de una tecnología ,sino varias englobadas en un término
EL que comenzó a utilizar masivamente esta tecnología fue GOOGLE, porque tiene la capacidad de dotar a las páginas web de un comportamiento típico de escritorio. Es decir que cuando hacemos, desde una página web, una petición a un servidor web, la respuesta del servidor no necesita que recargue la página completa, sino que tan solo se agregará a la página ya cargada.

Esto lógicamente mejora mucho la experiencia de usuario.

Vamos a entenderlo con un ejemplo:

Seguro que hemos utilizado esta tecnología sin saberlo. Google maps

Sabemos que google maps permite:

- dar una ruta, la más corta.
- Permite navegar por el mapa
- Hacer búsquedas en la casilla de búsqueda para encontrar establecimientos: Ejemplo: gasolineras

Todo esto es tecnología Ajax. Sin Ajax el simple hecho utilizar la lupa, implicaría una nueva carga de la página entera.

gracia a Ajax en lugar de cargar la pagina una y otra vez cada vez que pulsemos , como por ejemplo la lupa, se utilizaría como una aplicación de escritorio: tienes una imagen de fondo, y solo le vas añadiendo la información que va pidiendo.

Otros ejemplos a demás de empleo de Ajax:

Carrito de la compra: ítems que se van cargando al carro y se va actualizando la pagina, sin necesidad de que se recargue otra vez entera

Hoy en dia las empresas medianamente serias utilizan la tecnología AJAX, ej AMAZON

Proceso básico de funcionamiento de este tipo de tecnología

- Si partimos de un ordenador, con por ejemplo un mapa del google maps. Y un servidor remoto. Servidor encargado de proporcionar los datos que voy pidiendo
 1. Primero pedimos que nos cargue el mapa y el servidor nos lo proporcionan
 2. Luego por ejemplo les preguntamos por las gasolineras, esa información la enviamos al servidor .
 3. Cuando al servidor le llega esa información, la procesa y realiza la respuesta
 4. Cuando el navegador recibe esa respuesta, tiene que procesarla, y es ahí donde entra en juego el código javascript , AJAX. Se encarga de procesar la respuesta que nos ha enviado el servidor para **incorporarla** a el mapa



Se trata de un proceso básico, porque cada proceso tiene varios subprocesos que complican todo mucho

VENTAJAS DE UTILIZAR Y NO UTILIZAR AJAX

Cuando vamos navegando por una página web sin AJAX, y hacemos una petición, abre una pagina nueva cada vez. Al tener que abrir una pagina por petición, el tiempo de espera cada vez es mayor, y la experiencia de usuario , que es de lo que se trata, no es satisfactoria.

Si , por el contrario, realizamos peticiones al servidor mediante la tecnología AJAX, el servidor la gestiona y la respuesta se abre en la misma página web desde la que hemos hecho la petición, por lo que no observamos esa recarga



Decíamos que la primera **A**, viene de **Asíncrono (sin necesidad de que esté sincronizado.)** ¿Esto que quiere decir? Precisamente que permite realizar desde el navegador, varias peticiones simultáneas (no le has dado tiempo al servidor a realizar la primera pregunta y que te la devuelva, y ya le estás enviando la segunda).

Si no utilizas la tecnología AJAX, tu no puedes hacer la segunda petición hasta que la primera hno haya sido gestionada del todo por el servidor, es decir que no haya sido pedida, y luego devuelta (**Sin AJAX, seria SINCRONO**)

Cuales son las ventajas de AJAX, que hacen que mejore la experiencia de usuario:

1. Mostrar un nuevo contenido HTML sin recargar la página
2. Enviar un formulario y mostrar una respuesta inmediatamente
3. Hacer login en una pagina web sin abandonar la misma. Ejemplo de casilla para hacer login y contraseña, y cuando te logeas sin abandonar la página, ya reconoce que estás logeado , y te saca tu menú personalizado o lo que sea, pero sin recargar la página
4. Mostrar resultaos automáticos de encuestas: quien no ha hecho una encuesta siguiendo un formulario web, y al enviar a información , sin recargar la página te ofrece tus respuestas o los porcentajes de respuestas...
5. Buscar la información de una base de datos y ver los resultados inmediatamente (carro de la compra)

Antes de comenzar , solo queda saber los elementos fundamentales que forman parte de esta tecnología:

- I. NAVEGADOR **desde** donde lanzamos la petición
- II. SERVIDOR WEB **hacia** donde lanzamos la petición y que a su vez genera la respuesta
- III. CODIGO JAVASCRIPT encargado de **procesar la respuesta** del servidor adecuadamente
- IV. XMLHTTPREQUEST (XHR) . Objeto considerado como una especie de vehículo o puente que se utiliza para lanzar las peticiones al servidor y recibirlas. Se trata de un objeto perteneciente a los navegadores web este objeto se dice que es el CORE AJAX, es decir el núcleo más importante de la tecnología AJAX.
vamos a ver cómo funciona...

PASOS PARA ENVIAR PETICION AL SERVIDOR Y PROCESAR SU RESPUESTA

Debemos tener en cuenta, que el objeto, como objeto que es, tiene sus propiedades y sus métodos

- 1º Por lo tanto cuando queremos hacer una petición de AJAX, lo primero que hacemos es una instancia de este objeto , declarando una variable e instanciar este objeto,, con el operador new junto con el nombre del constructor

Crear instancia de XMLHttpRequest

```
var mi_var=new XMLHttpRequest();
```

- 2º Una vez creada esta instancia del objeto, podemos crear uno de sus métodos, como por ejemplo el método `open()` para especificar que petición se enviará al servidor y donde irá la petición dentro del servidor
- 3º A continuación crear una función para gestionar respuesta, java script. ¿cómo? Modificando el DOM (por ejemplo utilizando función `innerHTML` de `JavaScript`, o la función de `Jquery` `attr.....` en definitiva modificar los elementos de nuestro documento `html`: añadir contenido a un `DIV`, eliminar una etiqueta

4º Recibir la respuesta
esta respuesta puede venir :

- En texto
- En un archivo `html`
- Archivo `json` (`jeison`), de respuesta

La mayoría de las veces esta respuesta se tratará de un texto que debemos gestionar usando `JScript`, utilizando la propiedad **`responseText` del objeto `XHR`**

- ✓ típico código de error que indica que el archivo que le hemos pedido no se encuentra

404 file not found

- ✓ Otro usual que se trata de el código de respuesta que se da al servidor si hay un error interno del servidor (porque se encuentre en mantenimiento en el momento de pedir la petición y la base de datos no está disponible...)

500 internal server error

- ✓ Otro en el que solicitas páginas en el que hay que hacer un login y tu no lo haces.

403 access forbidden

A continuación veremos cómo trabaja la teoría `AJAX` y `JQUERY` en paralelo

EN definitiva los pasos que tenemos que seguir para establecer una comunicación con un servidor remoto utilizando la tecnología `AJAX`, serían:

1. Crear una instancia del objeto `XHR`, que nos permitirá utilizar métodos como `open()` y `send()` del objeto `XHR`, para comunicar con el servidor
2. Crear un código `JS` capaz de modificar el `DOM` de nuestra página, para por ejemplo ubicar la respuesta del servidor donde nos apetezca
3. Y por otro lado podríamos procesar la respuesta recibida, puesto que esta respuesta del servidor se acompaña de códigos internos que podemos procesar con `js`

Se trata de un proceso enormemente complejo, pero que se simplifica utilizando la librería JQuery.

Ventajas de utilizar librería JQuery en colaboración con AJAX:

- Unifica las versiones del navegador. Si no se utiliza la librería nos veríamos obligados a crear varias versiones para navegadores antiguos y modernos
- FUNCION LOAD(), que nos va a simplificar los pasos descritos . Es capaz de realizar una petición al servidor y de gestionar la respuesta que lanza el servidor a esta petición

Ejercicio práctico

- Partimos de un sitio web de tres paginas
- Al pincha en cada uno de los botones se carga la página nueva
- En el index tenemos un div vacío
- Se trata de que se recargue la pagina dentro de este div, en lugar de cargarlo en una página nueva.
- Para hacer esto no es obligatorio utilizar tecnología AJAX, puesto que podemos utiliza iframes
- ¿cómo hacer para que :
 - los vínculos no recarguen el ordenador y
 - además se descargue dentro del contenedor DIV
- Si utilizamos el código que indico a continuación, lo que le estoy es pidiendo que al pulsar en todos los vínculos que están dentro de el id noticias, cargue en el div contenidos_externos, siempre la pagina correspondiente pero no entera , solo #noticia

El load hace un request, y el servidor responde proporcionando un archivo, y después cargaremos ese contenido (el valor se lo dará la línea anterior con el this (ese mismo) de el atributo href de la línea anterior se la dará el

- OJO!! SI no ponemos el return false, se provocará un conflicto porque estará dándole una doble función al clicK (en html y en jquery)
- Cuidado con el espacio en blanco de antes de #noticia
- **CON EL RETURN FALSE INHABILITAMOS EL COMPORTAMIENTO HREF**

-

```
<script>

$(document).ready(function() {

    $("#noticias a").click(function() {

        var url=$(this).attr("href");

        $("#contenidos_externos").load(url + "#noticia");

        return false;

    });

});
```

- para ver esto tendremos que subirlo a servidor

FUNCIONES GET Y POST

Funciones que permiten enviar y recibir de un servidor remoto utilizando la tecnología AJAX

Estas funciones son las encargadas de **procesar la información** que le vamos a enviar al servidor.

Cuando llega esa información al servidor tenemos que utilizar tecnología de servidor (php, asp...) y el servidor hace la búsqueda que lo envía de nuevo al navegador. Aquí es donde entran en juego por segunda vez las funciones get y post. ¿cuál es la sintaxis que tienen estas funciones?

Las dos tienen una sintaxis idéntica

`$.get(url, datos_enviar, función_procesa_respuesta)`

`$.post(url, datos_enviar, función_procesa_respuesta)`

Ambas vienen precedidas de un símbolo dólar, pero este símbolo dólar no apunta a ninguna zona de nuestra página web

Las tres funciones tienen 3 argumentos:

1. el primer argumento es la url del archivo php asp... que procesará la respuesta en el servidor,
2. el segundo argumento son los datos
3. el tercer argumento es el nombre de una función JavaScript que procese la respuesta que le llegue del servidor. Este tercer argumento es opcional, porque a veces lo único que queremos es modificar datos que hay en el servidor (borrar o modificar registros en una base de datos de el servidor),

en estos dos casos yo no necesito procesar la respuesta, caso en que solo enviaremos dos argumentos

EJERCICIO

Se trata de un ejemplo que consiste en que para acceder a una supuesta zona privada vamos a utilizar una contraseña y un usuario

Vamos a imaginarnos que en este caso no se encuentra en una base de datos, sino que se encuentran predeterminados en un archivo PHP.

Se trata de que nos salga un mensaje de que está autorizado, si coinciden, y de que no está autorizado si no coinciden. Utilizando tecnología AJAX

1. como queremos que la acción se desencadene cuando pulsamos el botón de enviar, vamos a Jquery y apuntamos al formulario para que se ejecute el evento submit. Todo lo que programemos dentro de esta función anónima, se va a ejecutar cuando enviemos el formulario
2. Para construir el paquete listo para enviar: nuestro literal (llamado datos_formulario).
Almacenamos dentro de usuario, el valor que hay en el cuadro de texto usuario

```
<script>

$(document).ready(function() {

    $("#login").submit(function() {

        var datosFormulario={usuario:$("#usuario").val(),
                                contra:$("#contra").val() }

    });

});

</script>
```

3. Hasta aquí los datos estarían listos para enviar
4. PROCESO DE ENVIO Y RECEPCION
5. ¿cómo lo enviamos? Con la función GET, así que ahora añadimos:
 - a. Dentro de la función anónima la función get con los tres argumento
 - a.i. **Primero** la url o ruta del archivo del servidor que procesará este paquete que le vamos a enviar. Este literal ...**login.php**
 - a.ii. **Segundo argumento:** El paquete de datos, la información que le vamos a enviar llamado datosFormulario
 - a.iii. **Tercer argumento:** la función que va a gestionar la repuesta del servidor en el caso de que haya éxito. Todavía no existe, pero la vamos a llamar **procesarDatos**.

```
$("#login").submit(function() {

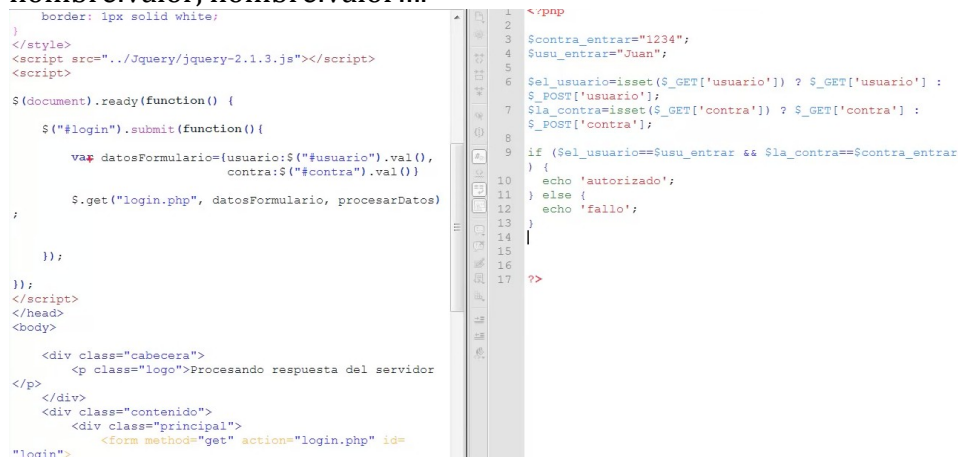
    var datosFormulario={usuario:$("#usuario").val(),
                          contra:$("#contra").val()}

    $.get("login.php", datosFormulario, procesarDatos);

});
```

Lo que hemos hecho con esta instrucción get es enviar la información al servidor.

- a.iv. Aquí entra en juego el archivo PHP partimos de que en el literal que hemos llamado datosFormulario tiene la sintaxis nombre:valor, nombre:valor, nombre:valor....



```
border: 1px solid white;
}
</style>
<script src="../../Jquery/jquery-2.1.3.js"></script>
<script>
$(document).ready(function() {
    $("#login").submit(function() {
        var datosFormulario={usuario:$("#usuario").val(),
                              contra:$("#contra").val()}
        $.get("login.php", datosFormulario, procesarDatos);
    });
});
</script>
</head>
<body>
<div class="cabecera">
<p class="logo">Procesando respuesta del servidor
</p>
</div>
<div class="contenido">
<div class="principal">
<form method="get" action="login.php" id="login">
1  </pnp>
2  $contra_entrar="1234";
3  $usu_entrar="Juan";
4  $usu_entrar="Juan";
5
6  $el_usuario=isset($_GET['usuario']) ? $_GET['usuario'] :
7  $_POST['usuario'];
8  $la_contra=isset($_GET['contra']) ? $_GET['contra'] :
9  $_POST['contra'];
10 if ($el_usuario==$usu_entrar && $la_contra==$contra_entrar) {
11     echo 'autorizado';
12 } else {
13     echo 'fallo';
14 }
15
16
17 ?>
```

El nombre tiene que coincidir con el del get, “usuario” y “contra”

y luego condicional en el que si es positivo devuelve ‘autorizado’, y si es negativo, devuelve ‘fallo’

- a.v. Creamos la Funtion **procesarDatos**
- a.v.1. **Esta función va a recibir unos datos, y estos datos deben almacenarse, y los almacenaremos en el primer argumento (datos_devueltos)**

Rescapitulando: con la función get, lo que hacemos es enviar el paquete de información “datosformulario” a el archivo “login.php”, el servidor procesa este archivo y nos procesa una respuesta “procesarDatos”.

¿Dónde se almacenan los datos de la respuesta?. En el argumento de la función “datos_devueltos”.

Y cual es la respuesta que se almacena, o palabra autorizado o palabra fallo, y ahí es donde podemos crear un condicional para que nos almacene la respuesta en el div “contenidos_externos”, mediante la función html()

por otro lado tenemos que tener en cuenta que el evento submit lo que hace por defecto es cargar la respuesta en una pagina nueva.
para que el formulario no se recargue en una página nueva, hay que incluir un **return false**.

con el return false, le digo que haga todo lo de la función pero **no la tarea que tiene asignada con submit**

a.vi.

```
</style>
<script src="../../Jquery/jquery-2.1.3.js"></script>
<script>

$(document).ready(function() {

    $("#login").submit(function() {

        /*var datosFormulario={usuario:$("#usuario").val(),
                                contra:$("#contra").val()} */

        var datosFormulario=$(this).serialize();

        $.get("login.php", datosFormulario, procesarDatos);

        return false;

    });

    function procesarDatos(datos_devueltos){

        if(datos_devueltos=="autorizado"){

            $("#contenidos_externos").html("<p> Usuario correcto. Bienvenido de nuevo </p>");

        }else{

            $("#contenidos_externos").html("<p> Usuario no autorizado. </p>");

        }

    }

}
```



a.vii. **Probarlo en servidor y nos daría...**

Si es verdadero....

a.viii.

Usuario: Juan

Contraseña: 1234

Enviar

Usuario correcto. Bienvenido de nuevo

si es falso.....

Usuario: wwwwww

Contraseña: wwwwww

Enviar

Usuario no autorizado.

Ahora vamos a ver el código para el caso en el que en el paquete de información, es decir en el literal, se almacenen un montón de datos , utilizando el `serialize()` para ir almacenando todos los datos del

formulario , mediante el this, dentro de "datos formulario"

```
</style>
<script src="../../Jquery/jquery-2.1.3.js"></script>
<script>

$(document).ready(function() {

    $("#login").submit(function() {

        /*var datosFormulario={usuario:$("#usuario").val(),
                                contra:$("#contra").val()} */

        var datosFormulario=$(this).serialize();

        $.get("login.php", datosFormulario, procesarDatos);

        return false;

    });
```

vamos a ver la diferencia entre la función GET Y POST

- la sintaxis es la misma
 - para comunicar con el servidor y procesar la respuesta vale tanto get como post
 - utilizamos get para realizar acciones en el servidor que no implicar modificar nada, es decir acciones de consulta, por ejemplo consultar en una base de datos a ver si existe un registro o no existe
 - el post se utiliza para aquellas acciones en las que si estás modificando algo: borrar un archivo o para actualizar registros (eliminar cambiar o añadir)
- Ojo se suele utilizar el post para modificar información pero si se utiliza el get también te deja. La diferencia es que el post no tiene límites en cuanto a número de caracteres o información que estamos enviando al servidor, mientras que el get si que tiene límite

CONTROLANDO ERRORES DEL SERVIDOR

Sabemos que el navegador envía una señal y este devuelve una respuesta, que para enviar la información que vamos llevar al servidor, necesitamos un objeto literal que va a hacer la función de una especie de paquete. En ese momento pueden surgir varios errores:

- que no haya conexión con internet
- que el servidor esté en mantenimiento o está saturado y no es capaz de procesar ese paquete que le vamos a entregar
- que el archivo de php, encargado de interpretar ese paquete, se eliminara del servidor o se renombró...el caso es que no se encuentra
- que cuando el servidor al dar la respuesta, a lo mejor el servidor se cae, se satura, se reinicia.....
- a lo mejor la conexión a internet sector por motivos desconocidos, y no somos capaces de receptionar el mensaje que envía el servidor

¿cómo informar al usuario de que el proceso, por motivos desconocidos, ha fallado y que lo vuelva a intentar?

mejor con un ejemplo

1. cambiamos el nombre de el archivo php para provocar que no lo encuentre
2. utilizamos la función error. Cuya sintaxis es...(nombre de función que gestione errores. Por ej procesarError)

con este código estamos pidiendo que busque un **login.php**, que le envíe el literal **datosFormulario** y que procese con la función **ProcesarDatos**, y en caso de que de error , se procesa la respuesta con **procesarError**

```
var datosFormulario=$(this).serialize();
$.get("login.php", datosFormulario, procesarDatos).error(ProcesarError);
return false;

});

function procesarDatos(datos_devueltos) {
    if (datos_devueltos=="autorizado") {
        $("#contenidos_externos").html("<p> Usuario correcto. Bienvenido de nuevo </p>");
    } else {
        $("#contenidos_externos").html("<p> Usuario no autorizado. </p>");
    }
}

function ProcesarError() {
    var msgError="Oops!! Ha ocurrido algo inesperado. Por favor, inténtalo más tarde";
    $("#contenidos_externos").html("<p>" + msgError + "</p>");
}
```

Usuario:

Contraseña:

Ooops!! Ha ocurrido algo inesperado. Por favor, inténtalo más tarde

OBJETOS JSON (TAMBIEN LLAMADO JEISON)

Lanzamos una información al servidor se contrasta con una base de datos y cuando la encuentra el servidor empaqueta la respuesta, y esa respuesta llega en diferentes formatos (xml, json, otros...)

En cualquier caso debemos leer el código que vienen en esos paquetes con código javascript

¿qué es un objeto Json?

¿QUÉ ES UN JSON?

```
{  
  +  
  Id: 5  
  Nombre: "Ana",  
  Apellido: "Fernández",  
  Teléfono: "915689874"  
}
```

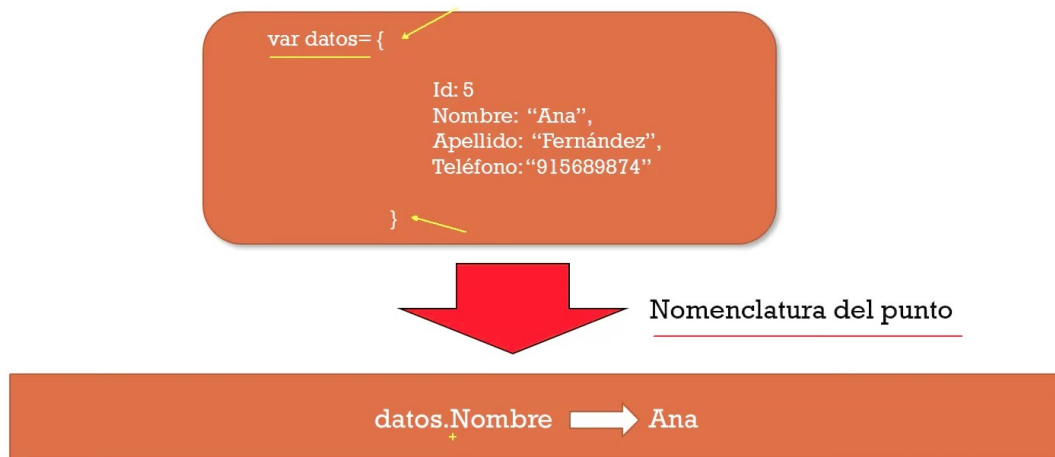
una llave que se abre y otra que se cierra y entre estas llaves, un formato de pares de valores. Nombre: valor, Nombre: valor, Nombre: valor...

Es la respuesta que nos llega del servidor (de una serie de personas, de un objeto de una tienda virtual....

Según el ejemplo, tu le preguntas al servidor por un id, y el te responde con el nombre, apellido y teléfono...y para devolverte toda esta información en base del id, contraseña, DNI....u otro código que le pidas, necesitas un objeto de tipo JSON.

El tema está en **como**, utilizando la tecnología AJAX, acceder a la información de estos objetos JSON. **Utilizando la nomenclatura del punto**

1. Primero asignamos un valor a este paquete, ej *datos*
2. Aplicamos la nomenclatura del punto



Si accedemos al dato nombre, me devolvería su valor *ANA*, datos punto teléfono, devolvería el *teléfono*.

Ejercicio

para este ejercicio deberíamos saber bases de datos y php, pero por si no se conoce, voy a proporcionar una base de datos de tipo JSON, en los que estos objetos ya estarán creados.

1. Partimos de un formulario en el que tenemos un solo cuadro de texto, *usuario*. Se trata de que según el usuario que preguntemos nos proporciona la información del mismo, ej el apellido y la edad. Esos datos que nos devuelve el servidor corresponden a un objeto JSON. Yo pregunto al servidor, el servidor pregunta al objeto si lo tiene y le contesta por completo

Usuario: Antonio

Enviar

Nombre: Antonio
Apellido: Martín
Edad: 51

JSON

2. para hacer este ejercicio partimos de los siguientes archivos:
 - 2.1. pagina html. En head de este archivo tenemos:
 - a.i. la función *ready* y
 - a.ii. la función *submit* para que se desencadene la acción al pulsar el botón enviar
 - a.iii. la variable *datosFormulario*, almacenamos toda la información escrita en el formulario
 - a.iv. función *serialize* para que pase por todos los datos (también podría utilizarse la sintaxis nombre:valor, nombre:valor, nombre:valor...
 - a.v. respecto a la función *get*, tendríamos que cambiarla por la función **getJSON**, con sintaxis idéntica a las funciones *get* y *post* que hemos explicado anteriormente (archivo php con que vas a comunicar, información que le estás enviando, función que se tiene que encargar de procesar la respuesta del servidor)

- a.vi. Por último la función que gestionará la respuesta del servidor
- b. ARCHIVO PHP,
 - archivo que estará en el servidor, y el encargado de procesar la información que lanzamos desde la página html.
 - En el encontramos:
 - b.i. **Variable** que almacene la información que estamos escribiendo en el cuadro de texto usuario. Es decir si escribimos en el formulario maría , ahí se almacena maría, si escribimos juan, se almacena juan....
 - b.ii. **Un array** que almacena un objeto de tipo JSON
 - b.iii. **Un condicional if** para el caso en que el valor de la variable sea juan, te de los datos del mismo dentro de este if también el codificador de el JSON que tiene almacenado en el array, utilizando una función de php que es jsonencode (esto lo suyo es que no sean datos fijos sino que ese archivo php se comunicara con una base de datos y por ultimo con la función echo de php le decimos que nos devuelva, los datos almacenados en el objeto JSON

Ejercicio

¿cómo podemos acceder a la información que nos proporciona este archivo?

1. Primero modificamos la función get por la función getjeison que es capaz de procesar la información respuesta en un formato JSON
2. Contactamos con un archivo **login2.php** y le enviamos la información la información que hay almacenada en **datosFormulario** (que es solo usuario), y la respuesta del servidor la procesa la función **procesarDatos**
3. El objeto JSON se almacena en el parámetro de procesarDatos, llamado datos_devueltos
4. Dentro de la función procesarDatos accedemos a el dato edad del JSON mediante la nomenclatura del punto

```

$(document).ready(function() {

    $("#login").submit(function() {

        /*var datosFormulario={usuario:$("#usuario").val(),
                               contra:$("#contra").val()} */

        var datosFormulario=$(this).serialize();

        $.getJSON("login2.php", datosFormulario, procesarDatos);

        return false;

    });

    function procesarDatos(datos_devueltos){

        $("#contenidos_externos").html("<p>La edad es: " + datos_devueltos.Edad + "</p>");

    }

});
</script>
</head>
<body>

```

Hacer lo mismo con apellido....

```
</style>
<script src="../../Jquery/jquery-2.1.3.js"></script>
<script>

$(document).ready(function() {

    $("#login").submit(function() {

        /*var datosFormulario={usuario:$("#usuario").val(),
                               contra:$("#contra").val()} */

        var datosFormulario=$(this).serialize();

        $.getJSON("login2.php", datosFormulario, procesarDatos);

        return false;

    });

    function procesarDatos(datos_devueltos) {

        $("#contenidos_externos").html("<p>El apellido es: " + datos_devueltos.Apellidd + "</p>");

    }

})
```