Guía: AJAX con jQuery

Conceptos básicos

¿Qué es jQuery?

jQuery es una biblioteca de JavaScript que permite simplificar la manera de escribir código javascript, facilitando tareas como manipular del árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web. Wikipedia

¿Qué es AJAX?

Acrónimo de Asynchronous JavaScript And XML. Ajax permite la ejecución de tareas en el lado del cliente mientras se mantiene la comunicación **asíncrona** con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, mejorando la interactividad, velocidad y usabilidad en las aplicaciones. Wikipedia

Ejemplo de AJAX con PHP

En el siguiente ejemplo se va a confeccionar una pequeña aplicación que permita al usuario consultar si se encuentra en una lista por medio de un formulario sin necesidad de recargar su navegador.

Debemos tener en cuenta que es necesario trabajar sobre un servidor que soporte PHP para que la aplicación se pueda ejecutar sin problemas.

1. Estructura de archivos

Dentro de la carpeta de nuestro proyecto llamada "proyecto-lista" vamos a añadir los archivos "index.html" y "script.js".

```
.proyecto-lista
    |-- index.html
    |-- script.php
```

2. Estructura html

Integrar jQuery

Para integrar la librería jQuery a nuestro proyecto simplemente debemos enlazarla en el archivo index.html por medio de la etiqueta `<script>` indicando la URL de la librería dentro del atributo "src". Esto lo hacemos dentro de las etiquetas `<head>...</head>` de nuestro html. La librería la podemos encontrar en los servidores de Google.

```
<head>
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.0/jquery.min.js"></
```

```
script>
</head>
```

Creando el formulario

Integraremos un pequeño formulario en nuestro index.html dentro del `<body>...</body>` donde el usuario pueda escribir su nombre y hacer la consulta al servidor.

En el atributo "action" de nuestro formulario vinculamos nuestro archivo "script.php" y en el atributo "method" indicamos que el envío de datos se haga por medio de POST.

Dentro del formulario añadiremos dos `<input></input>`, el primero será de tipo "text" y tendrá como nombre "user" y el segundo será de tipo "button", tendrá un id igual a "btn-enviar" y un valor igual a "Enviar".

```
<form action="script.php" method="post">
    Introduce tu nombre:
    <input type="text" name="user">
    <input type="button" id="btn-enviar"    value="Enviar">
</form>
```

Después del formulario vamos a añadir un div con id igual a "resultado", aquí vamos a mostrar el resultado de la petición.

```
<div id="resultado"></div>
```

Hasta el momento, en nuestro archivo index.html deberíamos tener algo similar a esto:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Proyecto Lista</title>
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.0/jquery.min.js"></
script>
</head>
<body>
    <form action="script.php" method="post">
        Introduce tu nombre:
        <input type="text" name="user">
        <input type="button" id="btn-enviar" value="Enviar">
    </form>
    </br>
    <div id="resultado"></div>
</body>
</html>
```

3. Integrando AJAX

jQuery nos proporciona un método que nos permite utilizar AJAX de una manera sencilla, el método lo invocamos con la sintaxis `$.ajax()` el cual recibe como parámetro un JSON que contiene las **propiedades** que indican cómo queremos que se comporte el método. Para este ejemplo utilizaremos sólo algunas de estas propiedades las cuales se explican a continuación.

- **url:** Indicamos la url en forma de cadena a la cual se va realizar la petición. En nuestro caso es "script.php" y la vamos a obtener del formulario utilizando el siguiente código JS/jQuery.

```
var pet = $("form").attr("action");
```

- **type:** Indicamos el método por el cual se va a realizar la petición (POST O GET). En nuestro caso es "POST" y la vamos a obtener del formulario utilizando el siguiente código JS/jQuery.

```
var met = $("form").attr("method");
```

- **data:** Indicamos en forma de JSON cuales son los datos que se van a enviar por medio de la petición al servidor. En nuestro caso vamos a utilizar el método de jQuery `serialize()` el cual vamos a implementar sobre el formulario simplificando bastante la situación. El método va a retornar en formato JSON todos los datos que captura el formulario (ej: nombre del usuario) sin necesidad de tener que especificarlos nosotros mismos. El código para realizar esto es el siguiente.

```
$("form").serialize();
```

- **success:** Aquí especificamos una función la cual recibe como parámetro la información que nos devuelve el archivo php. En nuestro caso será un mensaje que va a indicar si un usuario se encuentra o no en la lista y lo vamos a mostrar en el div con el id igual a "resultado" utilizando la función `text()` de jQuery. El código para hacer esto sería de la siguiente forma.

```
function(info){
    $("#resultado").text(info);
}
```

El método `$.ajax()` lo vamos a añadir después de la integración que hicimos de la librería jQuery, como es código de JavaScript lo vamos a incluir dentro de etiquetas `<script>...</script>`.

Siempre es necesario que escribamos código javascript dentro de la función `on()` para evitar que el código se ejecute antes de que el navegador cargue el DOM y se generen conflictos (Ej: no encuentre elementos de html que se guardan en variables de JS).

No debemos olvidar que debemos disparar el método AJAX con algún evento, en nuestro caso vamos a utilizar el método `click()` de jQuery sobre el botón con id igual a "btn-enviar", este método recibe como parámetro una función la cual se va a ejecutar luego de invocar el evento "click".

Para la función que recibe como parámetro la función `click()` vamos a especificar el argumento "e", este argumento pertenece al objeto evento que estamos realizando y con el fin de evitar que se envíe el formulario por default y se recague la página vamos a invocar el método del evento `preventDefault();`

```
("#btn-enviar").click(function(e){
    e.preventDefault();
});
```

Nuestro código javascript se debería ver así:

```
$(document).on("ready",function(){
    $("#btn-enviar").click(function(e){
        e.preventDefault();
        var pet = $("form").attr("action");
        var met = $("form").attr("method");
        $.ajax({
            url: pet,
            type: met,
            data: $("form").serialize(),
            success: function(info){
                $("#resultado").text(info);
            }
        });
    });
});
```

Con esto terminaríamos de trabajar en el archivo index.html el cual se debería ver de la siguiente manera

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Proyecto Lista</title>
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.0/jquery.min.js"></
script>
    <script>
        $(document).on("ready",function(){
            $("#btn-enviar").click(function(e){
               e.preventDefault();
                var pet = $("form").attr("action");
                var met = $("form").attr("method");
                $.ajax({
                    url: pet,
```

```
                    type: met,
                    data: $("form").serialize(),
                    success: function(info){
                                $("#resultado").text(info);
                             }
                });
            });
        });
    </script>
</head>
<body>
    <form action="script.php" method="post">
        Introduce tu nombre:
        <input type="text" name="user">
        <input type="button" id="btn-enviar" value="Enviar">
    </form>
    </br>
    <div id="resultado"></div>
</body>
</html>
```

4. El archivo PHP

A continuación procederemos a trabajar en nuestro archivo "script.php".

No olvidemos que el fin de nuestra aplicación es indicar al usuario si se encuentra o no en una lista, por lo tanto en nuestro código PHP vamos a recibir el nombre de usuario que se envió vía AJAX en una petición del tipo POST y lo vamos a guardar en la variable "usuario". Recordemos que el atributo nombre (name) de nuestro input es "user". Lo anterior lo realizamos de la siguiente forma.

```
$usuario = $_POST["user"];
```

Luego vamos a comparar dicho nombre de usuario con los nombres de usuario que incluiremos en un array (simulando una tabla en una db), si encontramos una igualdad significa que el usuario sí se encuentra en la lista y por lo tanto el mensaje a mostrar al usuario sería "Si se encuentra en la lista", en caso contrario el mensaje sería "No se ha encontrado en la lista". Este mensaje se envía a través de ajax al usuario, la función que especificamos en la propiedad "succes" es la que realiza esta acción.

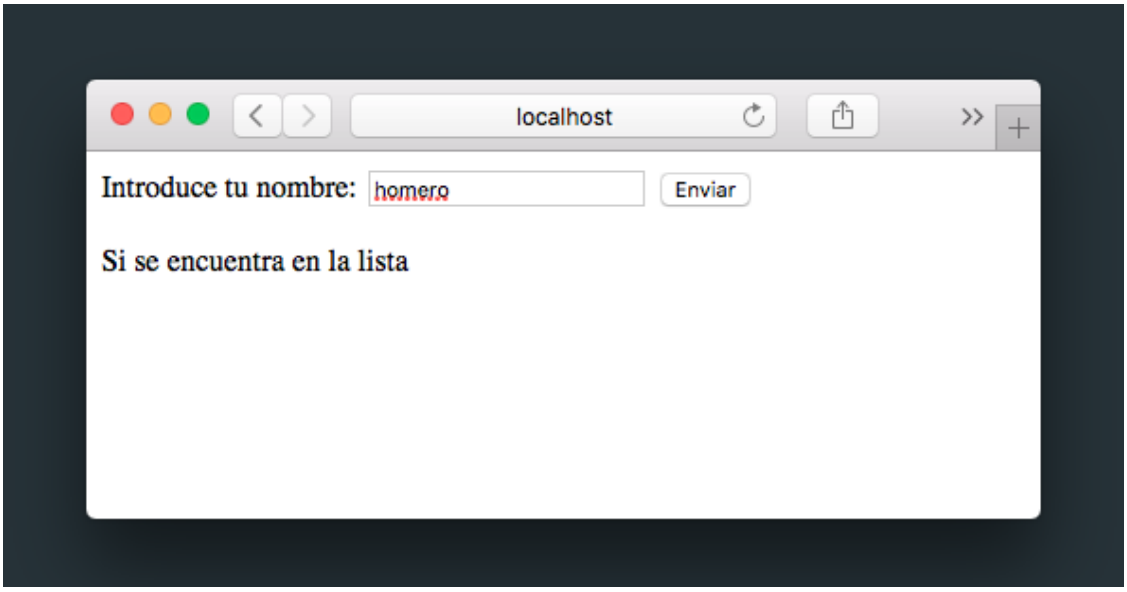
Nuestro archivo "script.php" se debería ver de la siguiente manera:

```
<?php
$usuario = $_POST["user"];

$array = array("pedro","homero","bart");

$resultado = "No se ha encontrado en la lista";

for ($i = 0; $i < count($array); $i++) {
```

```
    if($usuario == $array[$i]){
        $resultado = "Si se encuentra en la lista";
        break;
    }
}

echo $resultado;
?>
```

Si todo sale bien, nuesta aplicación debería funcionar perfectamente sin necesidad de recargar la página al momento de que el usuario realiza la consulta.

Resultado final:



Actividades

1. Permitir que el usuario pueda ingresar su apellido y su edad en el formulario.

2. Validar desde php que el nombre, apellido y edad del usuario no se encuentren vacíos, además validar que el nombre y usuario sean sólo letras y la edad sean sólo números. Si las validaciones no se cumplen mostrar un mensaje de error al usuario.

3. Averiguar para qué sirven el resto de propiedades que recibe como argumento en formato JSON el método `$.ajax`.

3. Utilizar la propiedad "beforeSend" para mostrar un mensaje de "cargando" antes de envíar la respuesta al usuario.

Consultar API jQyery