

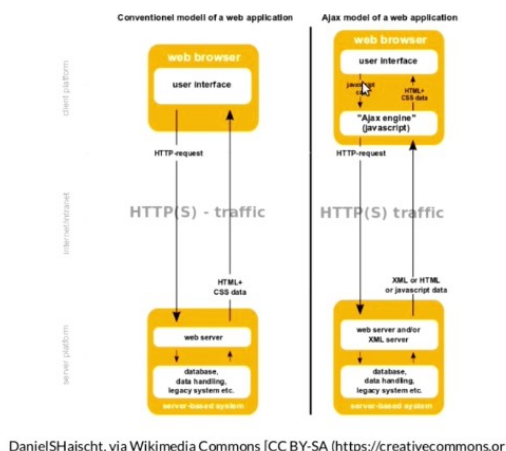
Ver vídeo de OpenWebinar: <https://youtu.be/VFNjPBdgRD0>

Qué es Ajax

Ajax es un conjunto de tecnologías (HTML o XHTML, CSS, JavaScript, DOM, XML y el objeto XMLHttpRequest) que, básicamente, nos permite actualizar ciertas partes de una página web sin tener que recargar toda la página web entera.

En la siguiente imagen comparamos el funcionamiento de Ajax con el modelo tradicional.

2. Funcionamiento



En la parte izquierda vemos cómo funciona la web en el modelo tradicional.

Desde un navegador el usuario hace una petición de una página web al servidor, el servidor le devuelve el HTML y el CSS de esa página y, cuando lo recibe, el navegador lo renderiza.

Sin embargo, cuando estamos trabajando con Ajax y hacemos una petición, el “Ajax engine” de JavaScript hace esa petición de manera asíncrona, sin tener que recargar la página.

En este caso, el servidor procesa la petición, devuelve XML, JSON o HTML al “Ajax engine” de JavaScript, que es el que se encarga de renderizar en el navegador.

Ajax y jQuery

Para trabajar Ajax con jQuery simplemente hay que hacer esta llamada:

```
$function(){  
    //Estructura general SIMPLIFICADA  
    $.ajax(url[,settings]);  
});
```

En esa llamada hay que indicar la URL con la que queremos trabajar y dentro del vector JSON todas las opciones que se quieren utilizar.

Simplemente con eso y con un pequeño conocimiento de jQuery, se pueden realizar llamadas asíncronas, como por ejemplo la que se realiza de forma práctica en el vídeo, y que te permite comprobar la utilidad de las mismas.

Vamos a realizar un pequeño ejemplo:

1.- Creamos una página sencilla denominada “index.php” que tendrá como título “Prueba de Ajax”

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Prueba de Ajax</title>  
</head>  
<body>  
  
</body>  
</html>
```

2.- Descargamos jquery o podemos utilizarlo directamente desde la página web.

Para descarga → <https://jquery.com/download/>

Para utilizar directamente →

<script src="<https://ajax.googleapis.com/ajax/libs/jquery/3.6.3/jquery.min.js>"></script>

En mi caso lo utilizo directamente

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.1/jquery.min.js"></script>
  <title>Prueba de Ajax</title>
</head>
<body>

</body>
</html>

```

3.- Ahora añadimos el siguiente código al final del body:



```

1  <script>
2      function functionName(){
3          var params = {
4              "var1" : "message1",
5              "var2" : "message2",
6              "var3" : "message3"
7          };
8
9          $.ajax({
10             data: params,
11             url: 'page2.php',
12             type: 'POST',
13             beforeSend: function(){
14                 $('#idMessage').html("Message before");
15             },
16             success: function(message_to_show){
17                 $('#idMessage').html("Message after");
18             }
19         });
20     };
21 }
22 </script>

```

4.- Ahora creamos un botón:

```

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.1/jquery.min.js"></script>
  <title>Prueba de Ajax</title>
</head>
<body>

  <input type="button" value="Saludar">|
<script>
  function functionName(){

```

5.- Ahora lo que queremos es que cuando se pulse en el botón, pues se ejecute el código por Ajax. Para ello modificamos el nombre de la función y la llamamos por ejemplo saludar y al botón le ponemos el evento onclick, para que cuando el usuario pinche sobre él ejecute el código que se encuentra en esa función.

Ahora vamos a personalizar nuestro código:


a) Lo primero es indicar que parámetros vamos a pasar a la página page2.php. Esto se hace en "data". En nuestro caso, vamos a modificar los datos que le pasamos indicando nuestro nombre, apellido y teléfono.

El código se te tiene que haber quedado así:




```
1 var params = {  
2     "nombre" : "profesor",  
3     "apellido" : "misApellidos",  
4     "telefono" : "123456789"  
5 };
```

b) Ahora creamos un div con id "idMessage" que será donde la página "page2.php" nos devolverá la información.



```
1 <div id="idMessage"></div>
```

c) También modificamos en la llamada ajax, en success, para indicarle que deseamos que el código que nos muestre vaya ahí.

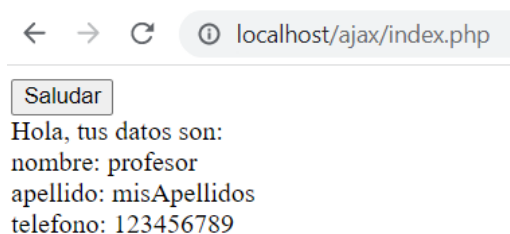


```
1 success: function(message_to_show){  
2     $('#idMessage').html("Message after");  
3 }
```

d) Vamos a crear una nueva página php con el código que queremos que ejecute el servidor, denominada page2.php y que contendrá el siguiente código

```
<?php
    $name = $_REQUEST['nombre'];
    $surname = $_REQUEST['apellido'];
    $phone = $_REQUEST['telefono'];
    echo "Hola, tus datos son:<br>
    nombre: $name<br>
    apellido: $surname<br>
    telefono: $phone";
?>
```

Pues bien, vamos a probar nuestra página para ver que efectivamente no se recarga la página, sino que nos modifica el div con los datos del usuario.



Para ver que efectivamente está funcionando, vete a la opción Network del programador y picha sobre el botón Saludar. Si te fijas, **no se recarga la página index.php**, sino que llama a page2.php y el resultado de esa llamada asíncrona, nos la vuelca en el div.

Saludar

Hola, tus datos son:
 nombre: profesor
 apellido: misApellido
 telefono: 123456789

DevTools is now available in Spanish! [Always match Chrome's language](#) [Switch DevTools to Spanish](#) [Don't show again](#)

Elements Console Sources **Network** Performance Memory Application Security Lighthouse Recorder »

☐ Preserve log ☐ Disable cache No throttling ☐ ☐ ☐

Filter ☐ Invert ☐ Hide data URLs **All** Fetch/XHR JS CSS Img Media Font Doc WS Wasm Manifest Other ☐ Has blocked cookies

☐ Blocked Requests ☐ 3rd-party requests

500 ms 1000 ms 1500 ms 2000 ms 2500 ms 3000 ms 3500 ms 4000 ms 4500 ms 5000 ms 5500 ms

Name	Status	Type	Initiator	Size	Time	Waterfall
index.php	200	document	Other	1.3 kB	7 ms	
jquery.min.js	200	script	index.php	31.1 kB	50 ms	
favicon.ico	200	x-icon	Other	31.2 kB	3 ms	
page2.php	200	xhr	jquery.min.js2	363 B	3 ms	

4 requests 64.0 kB transferred 122 kB resources Finish: 5.39 s DOMContentLoaded: 82 ms Load: 81 ms