Introducción a AJAX

AJAX es el sueño de un desarrollador, porque puedes:

- •Leer datos de un servidor web: una vez cargada la página
- •Actualizar una página web sin volver a cargar la página completa
- •Enviar datos a un servidor web, en segundo plano

Ejemplo AJAX

Let AJAX change this text

Inténtalo tú mismo "

Ejemplo AJAX

Página HTML

```
<!DOCTYPE html>
<html>
<body>

<div id="demo">
  <h2>Let AJAX change this text</h2>
  <button type="button" onclick="loadDoc()">Change Content</button>
</div>

</body>
</html>
```

La página HTML contiene una sección y un <botón>.
La sección se usa para mostrar información de un servidor.
El <botón> llama a una función (si se hace clic).

La función solicita datos de un servidor web y los muestra:

Function loadDoc()

```
function loadDoc() {
  const xhttp = new XMLHttpRequest();
  xhttp.onload = function() {
    document.getElementById("demo").innerHTML = this.responseText;
    }
  xhttp.open("GET", "ajax_info.txt", true);
  xhttp.send();
}
```

Podemos escribir la función anterior con una versión un poco más antigua, en la que en vez de onload se utiliza onreadstatechange y había que comprobar las propiedades readyState y status, pero ambas funciones son equivalentes:

Function loadDoc()

```
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
     document.getElementById("demo").innerHTML = this.responseText;
    }
  }
  xhttp.open("GET", "ajax_info.txt", true);
  xhttp.send();
}
```

¿Qué es AJAX?

AJAX = Asíncrono Javascript And XML.

AJAX no es un lenguaje de programación.

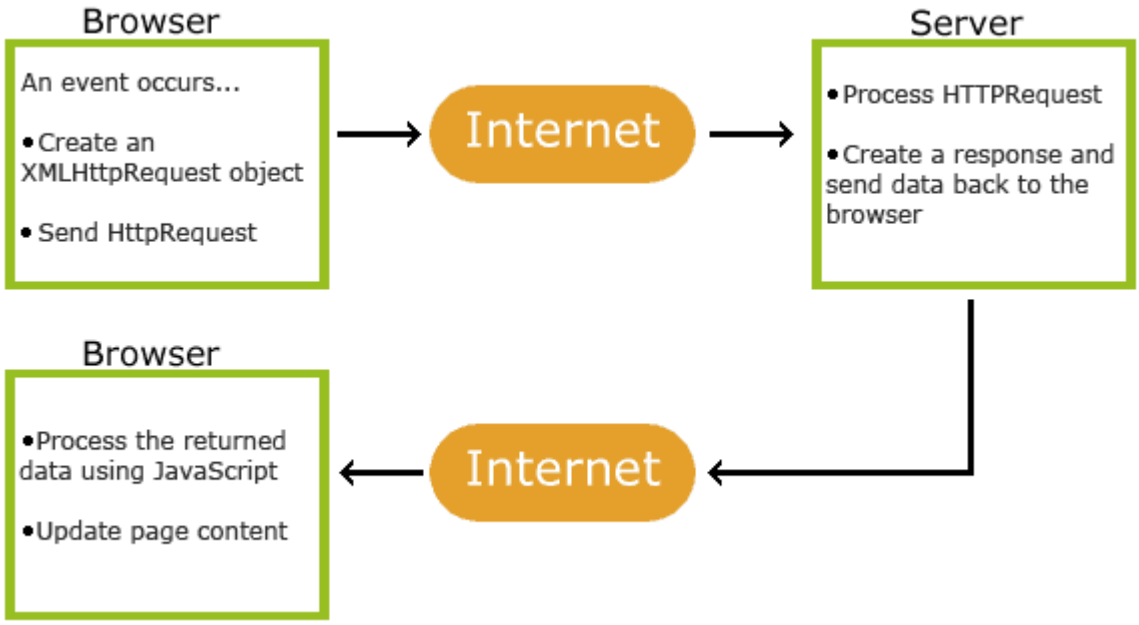
AJAX solo usa una combinación de:

- Un objeto XMLHttpRequest incorporado en un navegador (para solicitar datos desde un servidor web)
- JavaScript y HTML DOM (para mostrar o usar los datos)

AJAX es un nombre engañoso. Las aplicaciones AJAX pueden usar XML para transportar datos, pero es igualmente común transportar datos como texto sin formato o texto JSON.

AJAX permite que las páginas web se actualicen de forma asíncrona mediante el intercambio de datos con un servidor web en segundo plano. Esto significa que es posible actualizar partes de una página web, sin volver a cargar toda la página.

Cómo funciona AJAX



•1. Se produce un evento en una página web (se carga la página, se hace clic en un botón)

•2. Un objeto XMLHttpRequest es creado por JavaScript

•3. El objeto XMLHttpRequest envía una solicitud a un servidor web

•4. El servidor procesa la solicitud

•5. El servidor envía una respuesta a la página web

•6. La respuesta es leída por JavaScript

•7. La acción adecuada (como la actualización de la página) es realizada por JavaScript

AJAX - El objeto XMLHttpRequest

La piedra angular de AJAX es el objeto XMLHttpRequest.

1.Crear un objeto XMLHttpRequest
2.Definir una función de devolución de llamada
3.Abra el objeto XMLHttpRequest
4.Enviar una solicitud a un servidor

El objeto XMLHttpRequest

Todos los navegadores modernos admiten el objeto XMLHttpRequest.

El objeto XMLHttpRequest se puede usar para intercambiar datos con un servidor web detrás de las escenas. Esto significa que es posible actualizar partes de una página web, sin volver a cargar toda la página.

Crear un objeto XMLHttpRequest

Todos los navegadores modernos (Chrome, Firefox, IE7 +, Edge, Safari, Opera) tienen incorporado un objeto XMLHttpRequest.

Sintaxis para crear un objeto XMLHttpRequest:

```
variable = new XMLHttpRequest();
```

Ejemplo

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

Inténtalo tú mismo "

Definir una función de devolución de llamada (callback)

Una función de devolución de llamada es una función que se pasa como parámetro a otra función.

En este caso, la función de devolución de llamada debe contener el código para ejecutar cuando la respuesta esté lista.

```
xhttp.onload = function() {
  // What to do when the response is ready
}
```

Enviar una solicitud

Para enviar una solicitud a un servidor, puede usar los métodos open() y send() del objeto `XMLHttpRequest`:

```
xhttp.open("GET", "ajax_info.txt");
xhttp.send();
```

Ejemplo

```
// Create an XMLHttpRequest object
const xhttp = new XMLHttpRequest();

// Define a callback function
xhttp.onload = function() {
  // Here you can use the Data
}

// Send a request
xhttp.open("GET", "ajax_info.txt");
xhttp.send();
```

Inténtalo tú mismo "

Acceso a través de los dominios

Por razones de seguridad, los navegadores modernos no permiten el acceso a través de dominios.

Esto significa que tanto la página web como el archivo XML que intenta cargar deben estar ubicados en el mismo servidor.

Los ejemplos en W3Schools todos abren archivos XML ubicados en el dominio W3Schools.

Si desea utilizar el ejemplo anterior en una de sus propias páginas web, los archivos XML que cargue deben estar ubicados en su propio servidor.

Métodos del objeto XMLHttpRequest

Method	Description
new XMLHttpRequest()	Creates a new XMLHttpRequest object
abort()	Cancels the current request
getAllResponseHeaders()	Returns header information
getResponseHeader()	Returns specific header information
open(*method, url, async, user, psw*)	Specifies the request *method*: the request type GET or POST *url*: the file location *async*: true (asynchronous) or false (synchronous) *user*: optional user name *psw*: optional password
send()	Sends the request to the server Used for GET requests
send(*string*)	Sends the request to the server. Used for POST requests
setRequestHeader()	Adds a label/value pair to the header to be sent

---

Propiedades del objeto XMLHttpRequest

Property	Description
onreadystatechange	Defines a function to be called when the readyState property changes
readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
responseText	Returns the response data as a string
responseXML	Returns the response data as XML data
status	Returns the status-number of a request 200: "OK" 403: "Forbidden" 404: "Not Found" For a complete list go to the Http Messages Reference
statusText	Returns the status-text (e.g. "OK" or "Not Found")
onload	**Defines a function to be called when the request is recieved (loaded)**

La propiedad onload

Con el objeto `XMLHttpRequest` puede definir una función de devolución de llamada que se ejecutará cuando la solicitud reciba una respuesta.
La función se define en la propiedad `onload` del objeto `XMLHttpRequest`:

Ejemplo

```
xhttp.onload = function() {
  document.getElementById("demo").innerHTML = this.responseText;
}
xhttp.open("GET", "ajax_info.txt");
xhttp.send();
```

Inténtalo tú mismo "

La propiedad onreadystatechange

La propiedad `readyState` contiene el estado de **la petición** de XMLHttpRequest.
La propiedad `onreadystatechange` define una función de devolución de llamada que se ejecutará cuando cambie readyState.
Las propiedades `status` y `statusText` contienen el estado del objeto XMLHttpRequest.

Property	Description
onreadystatechange	Defines a function to be called when the readyState property changes
readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
status	200: "OK" 403: "Forbidden" 404: "Page not found" For a complete list go to the Http Messages Reference
statusText	Returns the status-text (e.g. "OK" or "Not Found")

La función `onreadystatechange` se llama cada vez que cambia readyState.

Cuando `readyState` es 4 y el estado es 200, la respuesta está lista:

Ejemplo

```
function loadDoc() {
  const xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML =
      this.responseText;
    }
  };
  xhttp.open("GET", "ajax_info.txt");
  xhttp.send();
}
```

Inténtalo tú mismo "

El evento `onreadystatechange` se activa cuatro veces (1-4), una vez por cada cambio en el estado listo.

AJAX: envíe una solicitud a un servidor

El objeto XMLHttpRequest se usa para intercambiar datos con un servidor.

Enviar una solicitud a un servidor

Para enviar una solicitud a un servidor, usamos los métodos open () y send () del objeto XMLHttpRequest:

```
xhttp.open("GET", "ajax_info.txt", true);
```

```
xhttp.send();
```

Method	Description
open(*method*, *url*, *async*)	Specifies the type of request *method*: the type of request: GET or POST *url*: the server (file) location *async*: true (asynchronous) or false (synchronous)
send()	Sends the request to the server (used for GET)
send(*string*)	Sends the request to the server (used for POST)

La url - Un archivo en un servidor

El parámetro url del `open()`método es una dirección a un archivo en un servidor:

```
xhttp.open("GET", "ajax_test.asp", true);
```

El archivo puede ser cualquier tipo de archivo, como .txt y .xml, o archivos de secuencias de comandos del servidor como .asp y .php (que pueden realizar acciones en el servidor antes de devolver la respuesta).

Asíncrono: ¿verdadero o falso?

Las solicitudes del servidor deben enviarse de forma asíncrona.
El parámetro asíncrono del método open() debe establecerse en verdadero:

```
xhttp.open("GET", "ajax_test.asp", true);
```

Al enviar de forma asíncrona, JavaScript no tiene que esperar la respuesta del servidor, sino que puede:

- ejecutar otros scripts mientras espera la respuesta del servidor
- tratar la respuesta después de que la respuesta esté lista

El valor predeterminado para el parámetro async es async = true.
Puede eliminar con seguridad el tercer parámetro de su código.
No se recomienda XMLHttpRequest síncrono (async = false) porque JavaScript dejará de ejecutarse hasta que la respuesta del servidor esté lista. Si el servidor está ocupado o lento, la aplicación se colgará o se detendrá.

GET o POST?

GET es más simple y más rápido que POST, y se puede usar en la mayoría de los casos.

Sin embargo, siempre use solicitudes POST cuando:

- Un archivo en caché no es una opción (actualice un archivo o base de datos en el servidor).
- Enviar una gran cantidad de datos al servidor (POST no tiene limitaciones de tamaño).
- Al enviar la entrada del usuario (que puede contener caracteres desconocidos), POST es más robusto y seguro que GET.

Peticiones GET

Una simple solicitud GET:

Ejemplo

```
xhttp.open("GET", "demo_get.asp", true);
xhttp.send();
```

Inténtalo tú mismo "

En el ejemplo anterior, puede obtener un resultado en caché. Para evitar esto, agregue una ID única a la URL:

Ejemplo

```
xhttp.open("GET", "demo_get.asp?t=" + Math.random(), true);
xhttp.send();
```

Inténtalo tú mismo "

Si desea enviar información con el método GET, agregue la información a la URL:

Ejemplo

```
xhttp.open("GET", "demo_get2.asp?fname=Henry&lname=Ford", true);
xhttp.send();
```

Inténtalo tú mismo "

Peticiones POST

Una simple solicitud POST:

Ejemplo

```
xhttp.open("POST", "demo_post.asp", true);
xhttp.send();
```

Inténtalo tú mismo "

Para enviar datos con POST como un formulario HTML, agregue un encabezado HTTP con setRequestHeader(). Especifique los datos que desea enviar en el método send():

Ejemplo

```
xhttp.open("POST", "ajax_test.asp", true);
xhttp.setRequestHeader("Content-type", "application/x-www-form-
urlencoded");
xhttp.send("fname=Henry&lname=Ford");
```

Inténtalo tú mismo "

Method	Description
setRequestHeader(*header, value*)	Adds HTTP headers to the request *header*: specifies the header name *value*: specifies the header value

---

Solicitud sincrónica

Para ejecutar una solicitud síncrona, cambie el tercer parámetro en el método open() a falso:

```
xhttp.open("GET", "ajax_info.txt", false);
```

Algunas veces async = false se usan para pruebas rápidas. También encontrará solicitudes sincrónicas en código JavaScript anterior.

Como el código esperará la finalización del servidor, no hay necesidad de una función onreadystatechange:

Ejemplo

```
xhttp.open("GET", "ajax_info.txt", false);
xhttp.send();
document.getElementById("demo").innerHTML = xhttp.responseText;
```

Inténtalo tú mismo "

No se recomienda XMLHttpRequest sincrónico (async = false) porque el JavaScript dejará de ejecutarse hasta que la respuesta del servidor esté lista. Si el servidor está ocupado o es lento, la aplicación se bloqueará o se detendrá.

La XMLHttpRequest sincrónica está en proceso de ser eliminada del estándar web, pero este proceso puede llevar muchos años.

Se alienta a las herramientas de desarrollo modernas a advertir sobre el uso de solicitudes síncronas y pueden arrojar una excepción InvalidAccessError cuando ocurra.

AJAX - Respuesta del servidor

Propiedades de la respuesta del servidor

Property	Description
responseText	get the response data as a string
responseXML	get the response data as XML data

Métodos de la respuesta del servidor

Method	Description
getResponseHeader()	Returns specific header information from the server resource
getAllResponseHeaders()	Returns all the header information from the server resource

---

La propiedad responseText

La propiedad responseText devuelve la respuesta del servidor como una cadena de JavaScript, y puede usarla en consecuencia:

Ejemplo

```
document.getElementById("demo").innerHTML = xhttp.responseText;
```

Inténtalo tú mismo "

La propiedad responseXML

El objeto XML HttpRequest tiene un analizador XML integrado.

La propiedad responseXML devuelve la respuesta del servidor como un objeto XML DOM.

Usando esta propiedad puede analizar la respuesta como un objeto XML DOM:

Ejemplo

Solicite el archivo cd_catalog.xml y analice la respuesta:

```
xmlDoc = xhttp.responseXML;
txt = "";
x = xmlDoc.getElementsByTagName("ARTIST");
for (i = 0; i < x.length; i++) {
  txt += x[i].childNodes[0].nodeValue + "<br>";
  }
document.getElementById("demo").innerHTML = txt;
xhttp.open("GET", "cd_catalog.xml", true);
xhttp.send();
```

Inténtalo tú mismo "

Ejemplo de AJAX PHP

El siguiente ejemplo muestra cómo una página web puede comunicarse con un servidor web mientras un usuario escribe caracteres en un campo de entrada:

Ejemplo

Start typing a name in the input field below:

First name:

Suggestions:

Ejemplo explicado

En el ejemplo anterior, cuando un usuario escribe un carácter en el campo de entrada, se ejecuta una función llamada "showHint ()".

La función se desencadena por el evento onkeyup.

Aquí está el código HTML:

Ejemplo

```
<html>
<head>
<script>
function showHint(str) {
    if (str.length == 0) {
        document.getElementById("txtHint").innerHTML = "";
        return;
    } else {
        var xmlhttp = new XMLHttpRequest();
        xmlhttp.onreadystatechange = function() {
            if (this.readyState == 4 && this.status == 200) {
```

```
document.getElementById("txtHint").innerHTML = this.responseText;
            }
        };
        xmlhttp.open("GET", "gethint.php?q=" + str, true);
        xmlhttp.send();
    }
}
</script>
</head>
<body>

<p><b>Start typing a name in the input field below:</b></p>
<form>
First name: <input type="text" onkeyup="showHint(this.value)">
</form>
<p>Suggestions: <span id="txtHint"></span></p>
</body>
</html>
```

Inténtalo tú mismo "

Explicación del código:

Primero, verifica si el campo de entrada está vacío (str.length == 0). Si es así, borre el contenido del marcador de posición txtHint y salga de la función.

Sin embargo, si el campo de entrada no está vacío, haga lo siguiente:

- •Crear un objeto XMLHttpRequest
- •Cree la función que se ejecutará cuando la respuesta del servidor esté lista
- •Envíe la solicitud a un archivo PHP (gethint.php) en el servidor
- •Tenga en cuenta que el parámetro q se agrega gethint.php? Q = "+ str
- •La variable str contiene el contenido del campo de entrada

Ejemplos de AJAX

https://www.w3schools.com/js/js_ajax_examples.asp