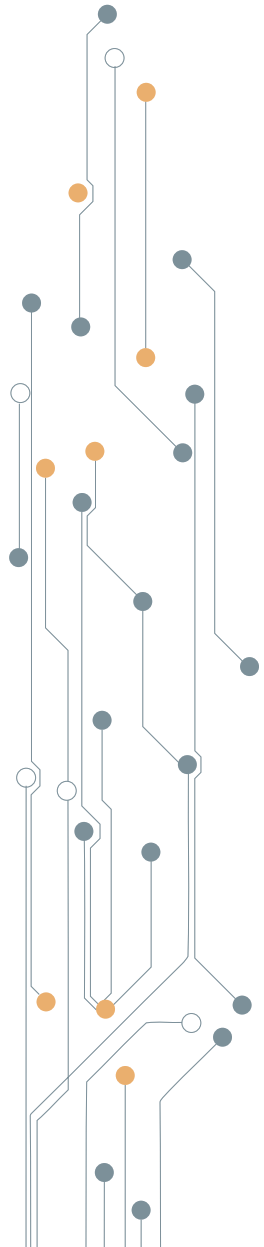
Ajax

Telefónica
EDUCACIÓN DIGITAL

Índice



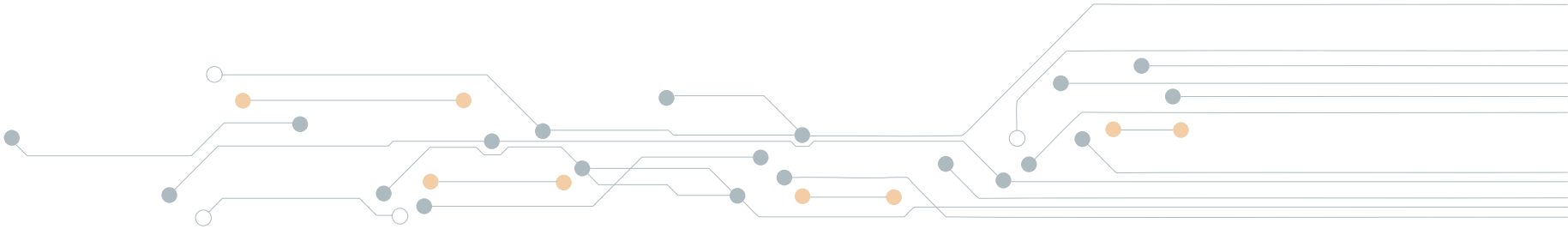
1. Introducción

AJAX se presentó por vez primera en el artículo “Ajax: A New Approach to Web Applications” de Jesse James Garrett en 2005. Anteriormente no existía un término que hiciera referencia a un tipo nuevo de programación web que estaba surgiendo.

Realmente, el término AJAX es un acrónimo (Asynchronous JavaScript + XML).

El artículo define AJAX de la siguiente forma:

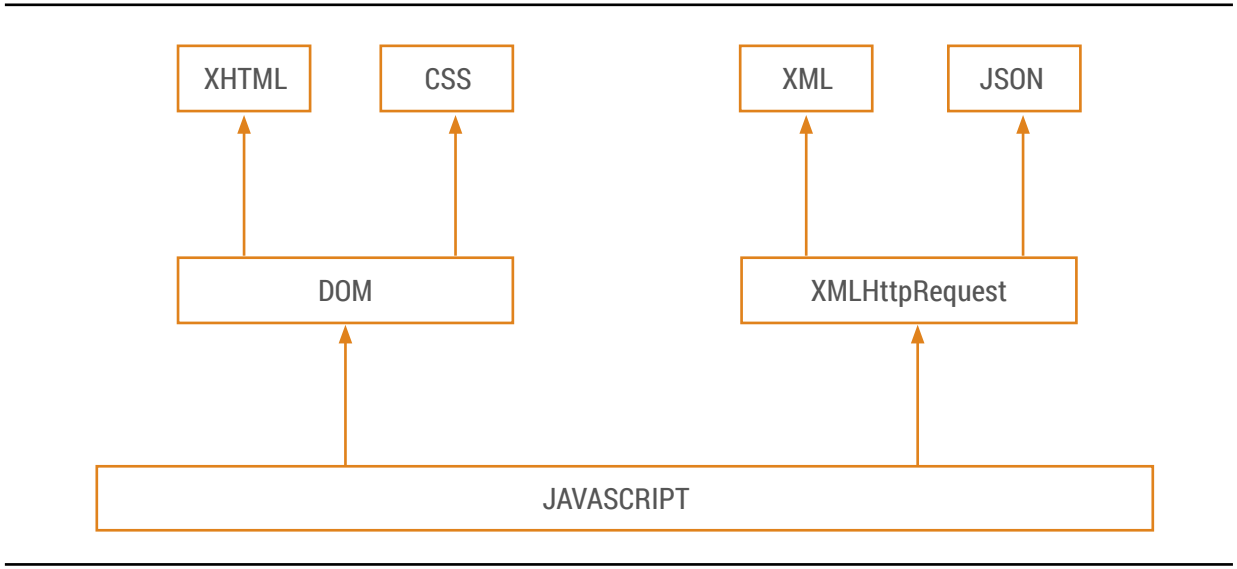
Ajax no es una tecnología en sí mismo. En realidad, se trata de varias tecnologías independientes que se unen de formas nuevas y sorprendentes



2. Funcionamiento y tecnologías implicadas

2.1 Particularidades JSON sobre JavaScript

- **XHTML y CSS**, crea una presentación basada en estándares.
- **XMLHttpRequest**, es el objeto encargado del intercambio asíncrono de información.
- **DOM**, para la manipulación e interacción dinámica de la capa de presentación.
- **XML, XSLT y JSON**, son las tecnologías que constituyen el intercambio y la manipulación de la información.
- **JavaScript**, como unión de todas tecnologías.



2.2| Funcionamiento de AJAX

El desarrollo de aplicaciones AJAX requiere un conocimiento avanzado de todas y cada una de las tecnologías anteriores de cada uno de los componentes.

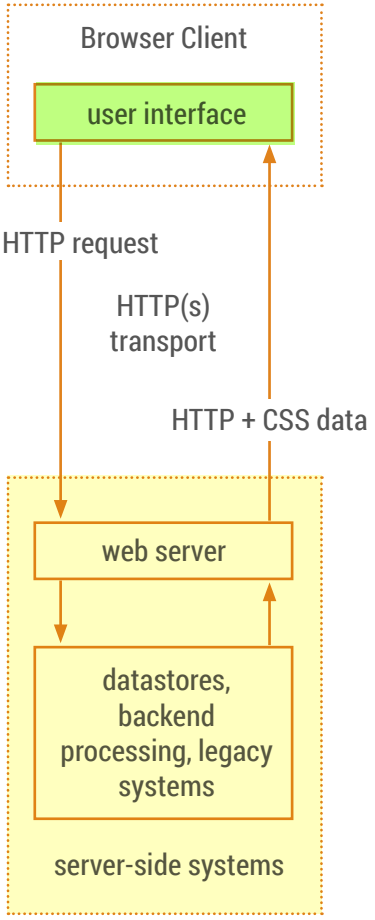
En los antiguos desarrollos web, acciones que realice nuestro cliente sobre la página desencadenaban llamadas al servidor. Y una vez el servidor hubiera acabado con ese tráfico de información devolvía y, por tanto, recargaba, la página web en nuestro cliente.

Viendo el esquema, el de la izquierda muestra el modelo antiguo para el desarrollo web. El de la derecha hace entrever el proceso de AJAX.

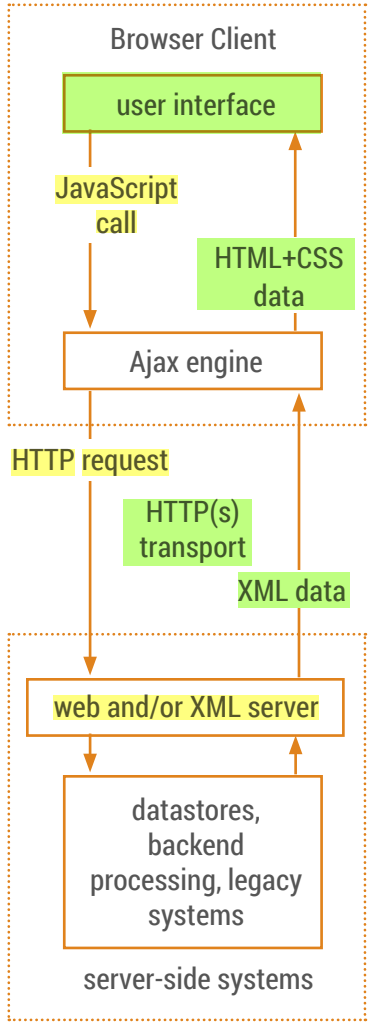
La técnica antigua que permite crear desarrollos web funciona correctamente, pero ralentiza la aplicación y el usuario recibe a cada interacción una petición del servidor que le obliga a recargar la página.

Esto suele ser bastante molesto para el usuario porque a cada cosa que haga se recarga la página por lo que su experiencia con la aplicación es bastante lenta y tediosa.

AJAX mejora la interacción del usuario con nuestra web, evita las recargas constantes, ya que el tráfico de información con el servidor se produce en un segundo plano.



CLASSIC WEB APPLICATION MODEL

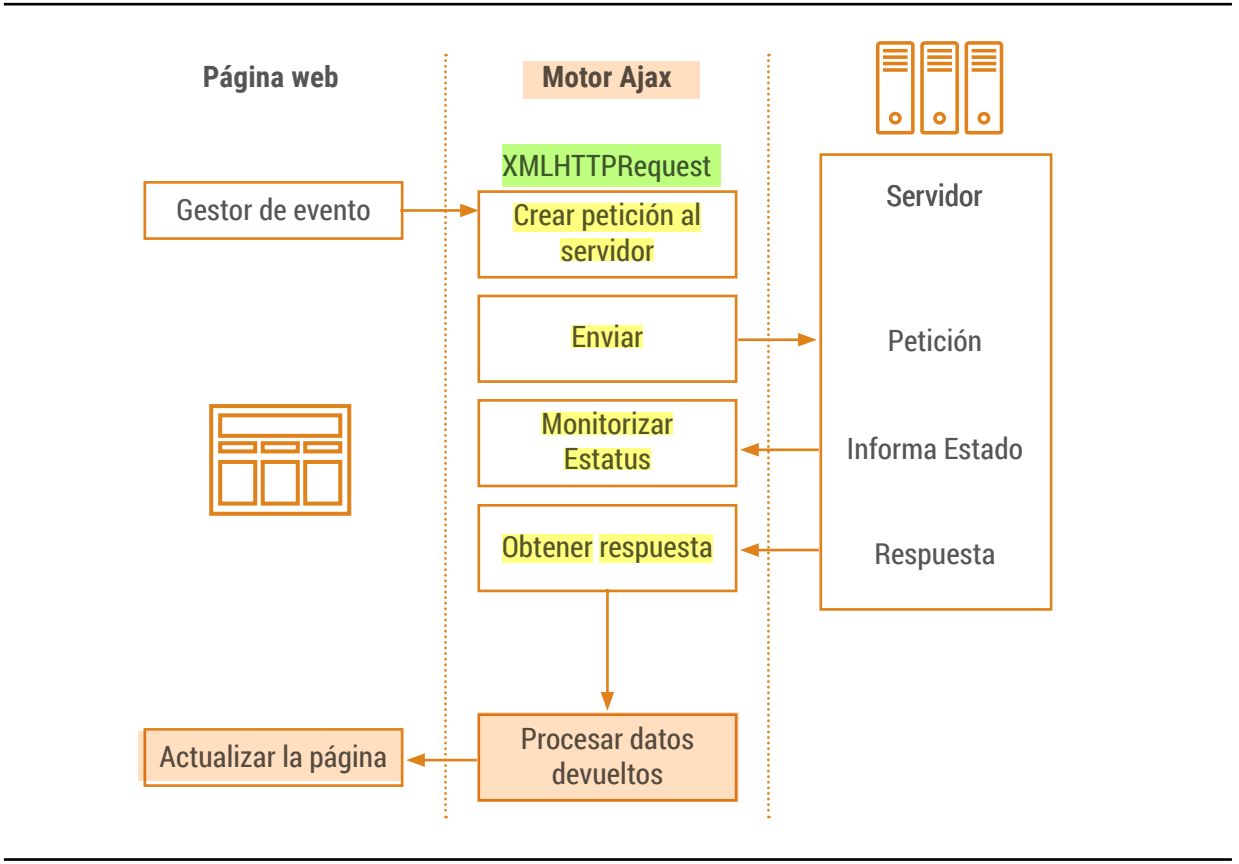


AJAX WEB APPLICATION MODEL

2.3| Peticiones AJAX y manejo de respuestas

El objeto XMLHTTPRequest:

- **Objetivo:** realizar peticiones asíncronas al servidor.
- Es la columna vertebral de todos los desarrollos con AJAX.
- Aceptado por **todos los clientes web** (Microsoft lo mete en IE 5 como un objeto ActiveX).



PROPIEDADES DEL OBJETO XMLHTTPREQUEST

Propiedades	Descripción
Onereadystaetchange onreadystatechange	Determina qué función será llamada cuando la propiedad readyState del objeto cambie
readyState	Número entero que indica el status de la petición: 0= No iniciada 1= Cargando 2= Cargado 3= Interactivo 4=Completado
responseText	Datos devueltos por el servidor en forma de string de texto
responseXML	Datos devueltos por el servidor expresados como un objeto documento.
Status	Código status HTTP devuelto por el servidor: 200= OK (petición correcta) 204= No content (documento sin datos) 301= Moved permanently (recurso movido) 401 = No authorized (necesita autentificación) 403= Forbidden (rechazada por el servidor) 404= Not found (no existe en servidor) 408= Request Timeout (tiempo sobrepasado) 500= Server Error (error en el servidor)
Metodos	
Abort()	Detiene la petición actual
getAllResponseHeaders()	Devuelve todas las cabeceras como un string
getResponseHeaader(x)	Devuelve el valor de la cabecera x como string
Open("Method", "URL","a") URL	Especifica el método HTTP (por ejemplo, GET o POST9, la UR objetivo y si la petición debe ser manejada asíncronamente (Si, a="True" defecto;No, a="false".)
Send (content)	Envía la petición
setRequestHeader("label","Value")	Configura un parámetro y valor label=value y lo asigna a la cabecera para ser enviado con la petición.

Ejemplo

A continuación vamos a ver un pequeño ejemplo práctico, con una petición GET al servidor para conseguir un fichero llamado *ajax.txt.*

```
<!DOCTYPE html>
<html>
<head>
<script>
function loadXMLDoc()
{
var xmlhttp;
if (window.XMLHttpRequest)
  {// code for IE7+, Firefox, Chrome, Opera, Safari
  xmlhttp=new XMLHttpRequest();
  }
else
  {// code for IE6, IE5
  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
xmlhttp.onreadystatechange=function()
  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
    document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
    }
  }
xmlhttp.open("GET","ajax.txt",true);
xmlhttp.send();
}
</script>
</head>
<body>
<div id="myDiv"><h2>Let AJAX change this text</h2></div>
<button type="button" onclick="loadXMLDoc()">Change Content</button>
</body>
</html>
```

VAMOS A ANALIZAR EL CÓDIGO SUPERIOR

El botón de mi html lanza con el evento click la función loadXMLDoc() la cual crea una variable llamada xmlhttp.

El primer condicional nos sirve para saber si nuestro navegador actual puede crear el objeto XMLHttpRequest que dependiendo del navegador lo creará de una manera u otra.

Abrimos la comunicación GET para pedir el fichero ajax.txt y enviamos la petición (se puede enviar tanto asíncrona como síncronamente dependiendo del valor del booleano).

La función onreadystatechange() se ejecutará a cada cambio de estado y cuando la petición sea de 200 y el readystate de 4 significará que se ha completado con éxito (con cualquier otra solución debemos tratar el error o los errores que hemos visto en las tablas anteriores).

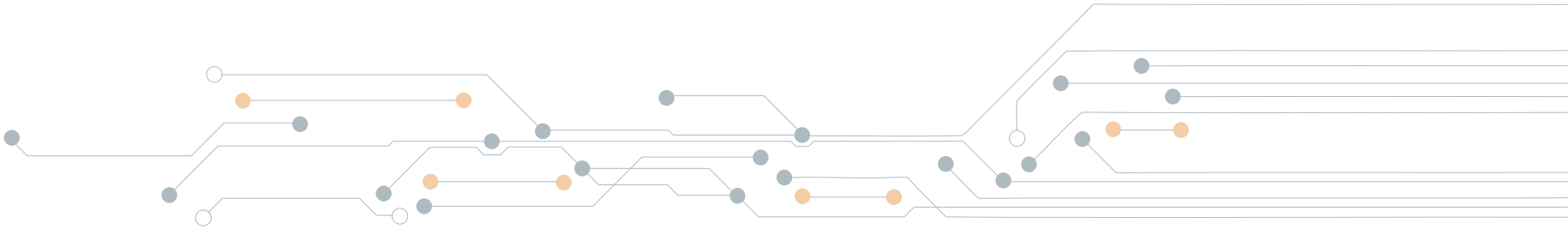
Finalmente seleccionamos el div en cuestión (a partir del API del DOM) y cambiamos su texto plano por el que me ha venido en mi fichero ajax.txt.

2.4 | XML vs JSON

Si comparamos los dos formatos, queda claro que uno no es mejor que otro. Como siempre, de lo que se trata, es saber cómo queremos tratar nuestra información y con qué queremos conectarnos para ese trato.

Si estamos hablando de envío de datos de un extremo a otro (como en el caso de dos servicios web o para dos servidores independientes) XML parece una elección correcta. Aunque, también hay que decir que en el caso de los servicios web han aumentado el uso de REST frente a SOAP, servicios que van enfocados al uso de JSON.

Si son datos que debemos transmitir en una misma aplicación internamente o, aparte, estamos haciendo uso de AJAX, JSON debería ser nuestra elección sin ningún tipo de dudas ya que la devolución del objeto de AJAX es fácilmente convertible.



Telefónica

EDUCACIÓN DIGITAL