

## U6. Ajax

### **Parte I. Introducción a Ajax**

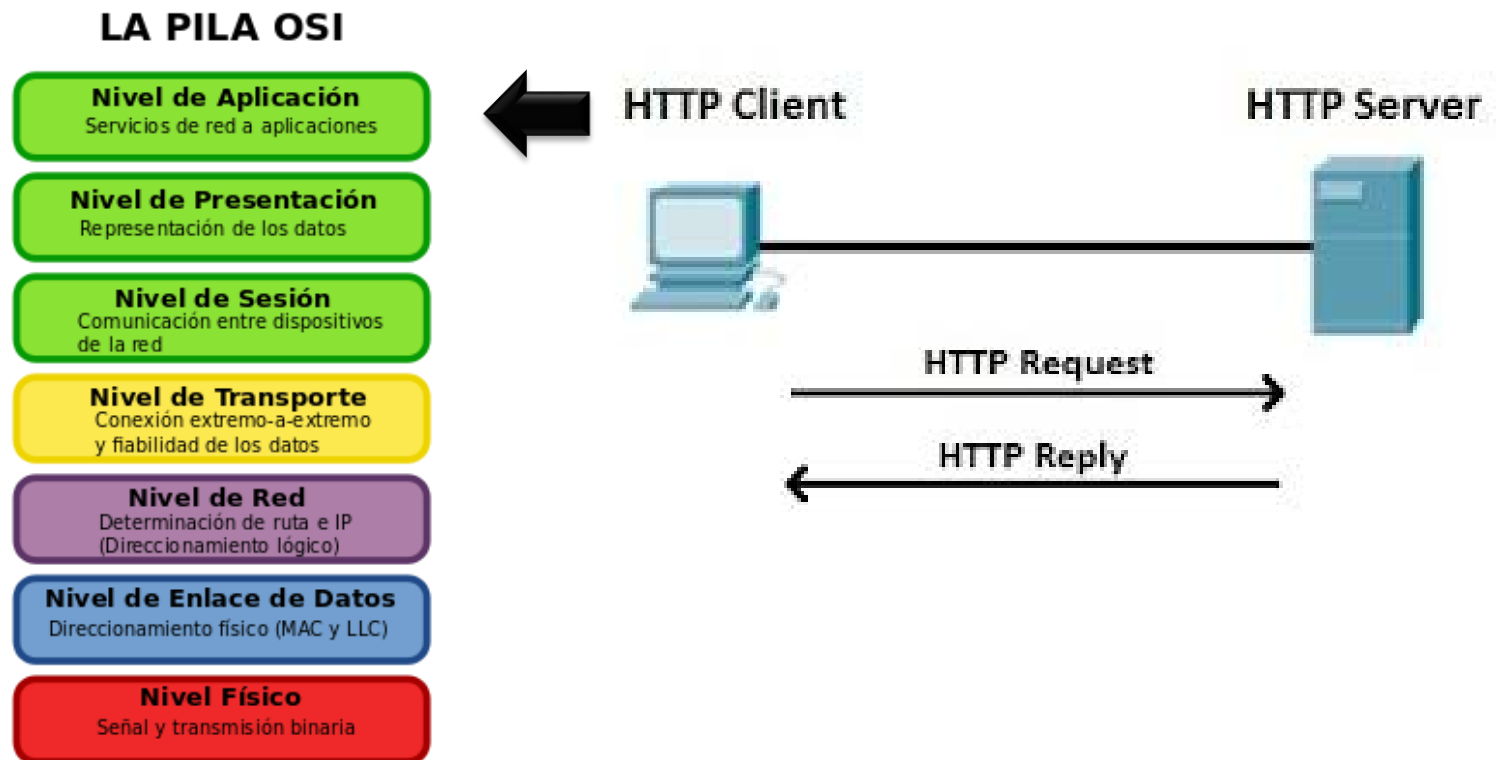
Desarrollo Web en Entorno Cliente

# Contenidos

- El protocolo HTTP
- ¿Qué es Ajax?
- ¿Cómo funciona Ajax?
  - Estructura básica petición Ajax
  - Objeto XMLHttpRequest
- Diferencia entre GET y POST
- Otros formatos

# HTTP

- **Características HTTP:**
  - Protocolo de nivel de aplicación



# HTTP

```
GET /index.html HTTP/1.1
Host: www.example.com
User-Agent: nombre-cliente
Referer: www.google.com
User-Agent: Mozilla/5.0 (X:
Connection: keep-alive
[Línea en blanco]
```

```
HTTP/1.1 200 OK
Date: Fri, 31 Dec 2003 23:59:59 GMT
Content-Type: text/html
Content-Length: 1221

<html lang="eo">
<head>
<meta charset="utf-8">
<title>Título del sitio</title>
</head>
<body>
<h1>Página principal de tuHost</h1>
(Contenido)
```

- **El cliente:**
  - **Solicita una conexión**
  - Envía la petición de un recurso al servidor mediante los métodos **GET** o **POST**.
- **El servidor:**
  - Responde incluyendo la versión del protocolo y un [código de estado](#) y el contenido indicando el tipo de dato **MIME (content-type)**.
  - **Cierra la conexión**

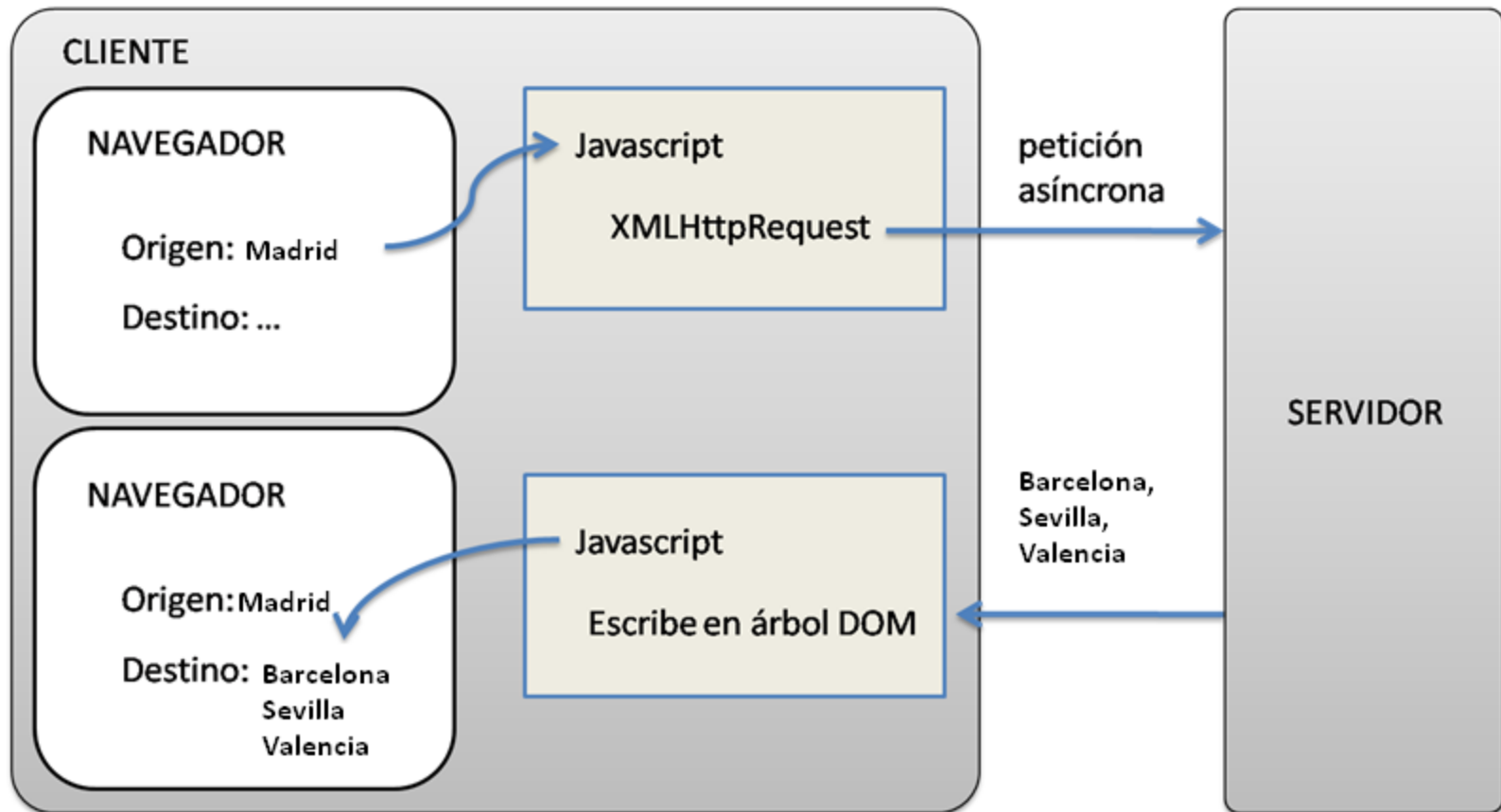
# ¿Qué es Ajax?

- Ajax (Asynchronous JavaScript And XML) no es un lenguaje.
- Es una técnica que requiere de un conjunto de tecnologías:

**Ajax = HTML + XML + DOM + JS + XMLHttpRequest**

- Surge en 1999:  
<http://www.evolutionoftheweb.com/>

# ¿Qué es Ajax?



# ¿Cómo funciona Ajax?

- Estructura básica

1. Función

“obtenerDatosServidor”  
contiene dos parámetros.

2. Se elige el elemento HTML a ser modificado.

3. Se configura una conexión asíncrona con una URL.

4. Se indica la función a ser llamada una vez el estado del objeto cambie.

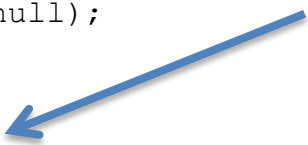
5. Se abre la conexión.

```
<script language = "javascript">

    var objetoXHR = new XMLHttpRequest();

    1 function obtenerDatosServidor(origen, elemento)
    {
    2     var objeto_destino = document.getElementById(elemento);
    3     objetoXHR.open("GET", origen);
    4     objetoXHR.onreadystatechange = respuesta();
    5     objetoXHR.send(null);
    }

    function respuesta(){
        if (objetoXHR.readyState == 4 &&
            objetoXHR.status == 200) {
            objeto_destino.innerHTML = objetoXHR.responseText;
        }
    }
</script>
```



# Objeto XMLHttpRequest

Atributo	Descripción
<code>readyState</code>	Devuelve el estado del objeto como sigue: 0 = sin inicializar, 1 = abierto, 2 = cabeceras recibidas, 3 = cargando y 4 = completado.
<code>responseBody</code>	Devuelve la respuesta como un array de bytes.
<code>responseText</code>	Devuelve la respuesta como una cadena.
<code>responseXML</code>	Devuelve la respuesta como XML. Esta propiedad devuelve un objeto documento XML, que puede ser examinado usando las propiedades y métodos del árbol DOM.
<code>status</code>	Devuelve el estado como un número (p. ej. 404 para "Not Found").
<code>statusText</code>	Devuelve el estado como una cadena (p. ej. "Not Found").



# Objeto XMLHttpRequest

Métodos	Descripción
<code>abort()</code>	Cancela la petición en curso
<code>getAllResponseHeaders()</code>	Devuelve el conjunto de cabeceras HTTP como una cadena.
<code>getResponseHeader(cabecera)</code>	Devuelve el valor de la cabecera HTTP especificada.
<code>open(método, URL [, asíncrono [, nombreUsuario [, clave]]])</code>	<p>Especifica el método, URL y otros atributos opcionales de una petición.</p> <p>El parámetro de <b>método</b> puede tomar los valores "GET", "POST", o "PUT".</p> <p>El parámetro <b>URL</b> puede ser una URL relativa o completa.</p> <p>El parámetro <b>asíncrono</b> especifica si la petición será gestionada asíncronamente o no.</p>
<code>send([datos])</code>	Envía la petición.

# Objeto XMLHttpRequest

Propiedades	Descripción
<code>onreadystatechange</code>	Evento que se dispara con cada cambio de estado.
<code>onabort</code>	Evento que se dispara al abortar la operación.
<code>onload</code>	Evento que se dispara al completar la carga.
<code>onloadstart</code>	Evento que se dispara al iniciar la carga.
<code>onprogress</code>	Evento que se dispara periódicamente con información de estado.

# Diferencia entre GET y POST

Aspectos a valorar	GET	POST
Tamaño petición	Restringido	No restringido
Velocidad	Mayor	Menor
Envío archivos	No	Sí
Visibilidad parámetros	Sí (en la url)	No (forman parte de la información en el cuerpo)
Seguridad	Baja	Alta

# Envío de datos con GET

- El envío de datos con GET se hace añadiendo los datos a la url de la siguiente manera:

*url?dato=valor&dato=valor...*

- Consideraciones:
  1. Existe una longitud máxima de url que limita los datos que podemos enviar.
  2. La información se envía en abierto
  3. El valor se incorpora a la url por lo que debe tener caracteres admitidos en una url.  
Recomendable: [encodeURIComponent](#)

# Envío de datos con GET

- [Ejemplo:](#)

```
<script>
function loadDoc() {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function() {
        if (this.readyState == 4 && this.status == 200) {
            demo.innerHTML = this.responseText;
        }
    };
    xhttp.open("GET", "demo_get2.asp?fname=Henry&lname=Ford",
true);
    xhttp.send();
}
</script>
```

[Ejemplo sugerencias](#)

# Envío de datos con POST

- POST permite enviar contenido en el cuerpo del paquete HTTP (no en la cabecera).
- Indicamos el formato del envío con:  
`setRequestHeader("Content-Type",tipo-mime);`
- Enviamos con:  
`send(datosAenviar)`
- Emplearemos el envío con POST cuando:
  - Enviamos un input del usuario que puede contener caracteres especiales o información confidencial
  - Queremos enviar cantidades grandes
  - Queremos que no se cachee la respuesta

# POST

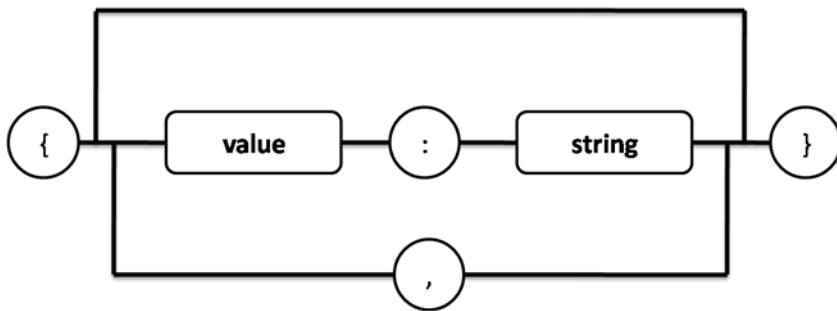
- [Ejemplo recepción](#) con **POST**
- [Ejemplo envío](#) con **POST** (equivale al envío que hemos hecho con get)

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

- Formato por defecto de envío de formulario (se especifica en [enctype](#) sólo si el método es POST):  
application/x-www-form-urlencoded

# JSON

- JSON es Java Script Object Notation
- Surge para el intercambio de datos como alternativa a XML (tiene menos redundancia y una mayor compatibilidad con JS).



1. Un objeto comienza con { y finaliza con }.
2. Cada nombre es seguido por dos puntos, estando los pares nombre/valor separados por una coma.
3. Admite los tipos: string, numeric, booleano, null, objetos y arrays.



# Envío y recepción con JSON

- Si recibimos un fichero JSON tenemos que parsearlo y manipularlo como un objeto JS:  
`var ciudadano = JSON.parse(responseText);`  
`ciudadano[0].nombre; ó bien ciudadano[0]["nombre"];`
- Enviamos un JSON a partir de un objeto JS:  
`xhr.open("POST",url);`  
`xhr.setRequestHeader("Content-Type","application/json");`  
`xhr.send(JSON.stringify(objetoJS));`

# XML

- El lenguaje XML es utilizado para describir y estructurar datos.
- Los navegadores contienen funcionalidades internas para trabajar con documentos XML. Además, alrededor de XML encontramos otras tecnologías: XQuery, XSLT.

```
<?xml version="1.0" encoding="utf-8" ?>
<ciudadano>
  <nombre>pepe</nombre>
  <edad>34</edad>
  <domicilio>calle alcalá 1</domicilio>
  <estudios>
    <estudio>primario</estudio>
    <estudio>secundario</estudio>
  </estudios>
</ciudadano>
```

# Ejemplo JSON vs. XML

<pre>{   'nombre': 'pepe',   'edad': 34,   'domicilio': 'calle alcalá 1',   'estudios': ['primario',                'secundario',                'universitario'] }</pre>	<pre>&lt;ciudadano&gt;   &lt;nombre&gt;pepe&lt;/nombre&gt;   &lt;edad&gt;34&lt;/edad&gt;   &lt;domicilio&gt;     calle alcalá 1   &lt;/domicilio&gt;   &lt;estudios&gt;     &lt;estudio&gt;primario&lt;/estudio&gt;     &lt;estudio&gt;secundario&lt;/estudio&gt;     &lt;estudio&gt;universitario&lt;/estudio&gt;   &lt;/estudios&gt; &lt;/ciudadano&gt;</pre>
---	---

# Envío y Recepción con XML

- Si recibimos un fichero XML:
  - empleamos `xhr.responseXML`
  - lo recorremos con las instrucciones vistas para manipular el DOM
- Podemos crear un fichero XML a partir de un string:

```
var parser = new DOMParser();  
parser.parseFromString(stringXML, "text/xml");
```
- Para enviar un fichero XML:

```
xhr.open("POST",url);  
xhr.setRequestHeader("Content-Type","text/xml");  
xhr.send(fichero.xml)
```