Diseño y Programación Web

Tema 5

JavaScript (AJAX)

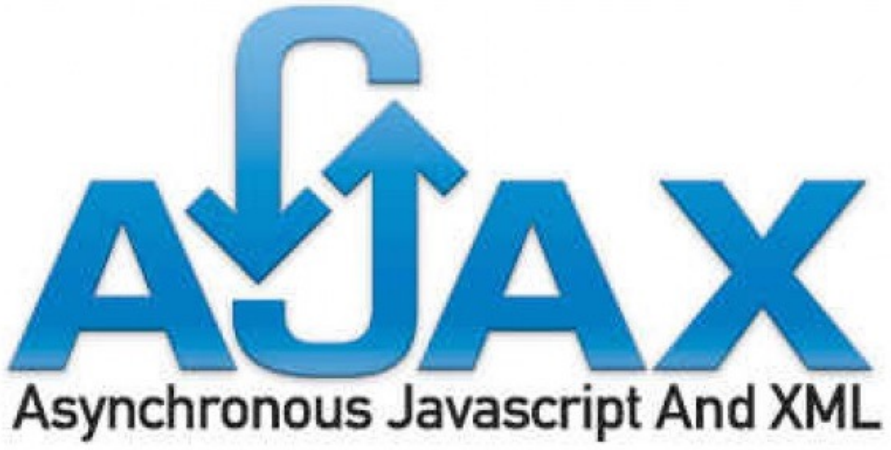
Enrique Moguel
enrique@unex.es

AJAX

- ***A**synchronous **J**avaScript **A**nd **X**ML*

- Conjunto de técnicas de desarrollo web utilizadas en el lado del cliente para crear aplicaciones asíncronas.

- Permiten a las aplicaciones enviar y recibir datos de un servidor sin interferir en la visualización y comportamiento de la página que se muestra.



AJAX

- En una aplicación web convencional:
  - La interacción con el usuario sigue el patrón "*Click, wait and refresh*".
  - Es necesario refrescar la página para cada evento, transmisión de datos, ...
  - El usuario debe esperar por la respuesta.

AJAX

- Problemas de este tipo de aplicaciones web convencionales:
  - Respuesta lenta.
  - El contexto operacional se pierde.
    - Se pierde la información en la pantalla.
    - Se pierde la posición del scroll.
  - No se retroalimenta al usuario de forma inmediata.
- Para suplir estos defectos surgen las **tecnologías RIA**.

AJAX

- **DHTML**:
  - DHTML = JavaScript + DOM + CSS.
  - Utilizado para crear aplicaciones interactivas.

- No permite comunicación asíncrona.
  - Se requiere la recarga completa de la página.

AJAX

- **AJAX**:
  - DHTML más comunicación asíncrona a través del objeto **XMLHttpRequest**.

  - Pros:
    - Muy asentado en la industria.
    - Existen muchos *framework* y herramientas.
    - No es necesario descargar aplicaciones ni añadir *plug-ins* al navegador.

  - Contras:
    - Incompatibilidades entre buscadores.
    - JavaScript es difícil de mantener y depurar.

AJAX

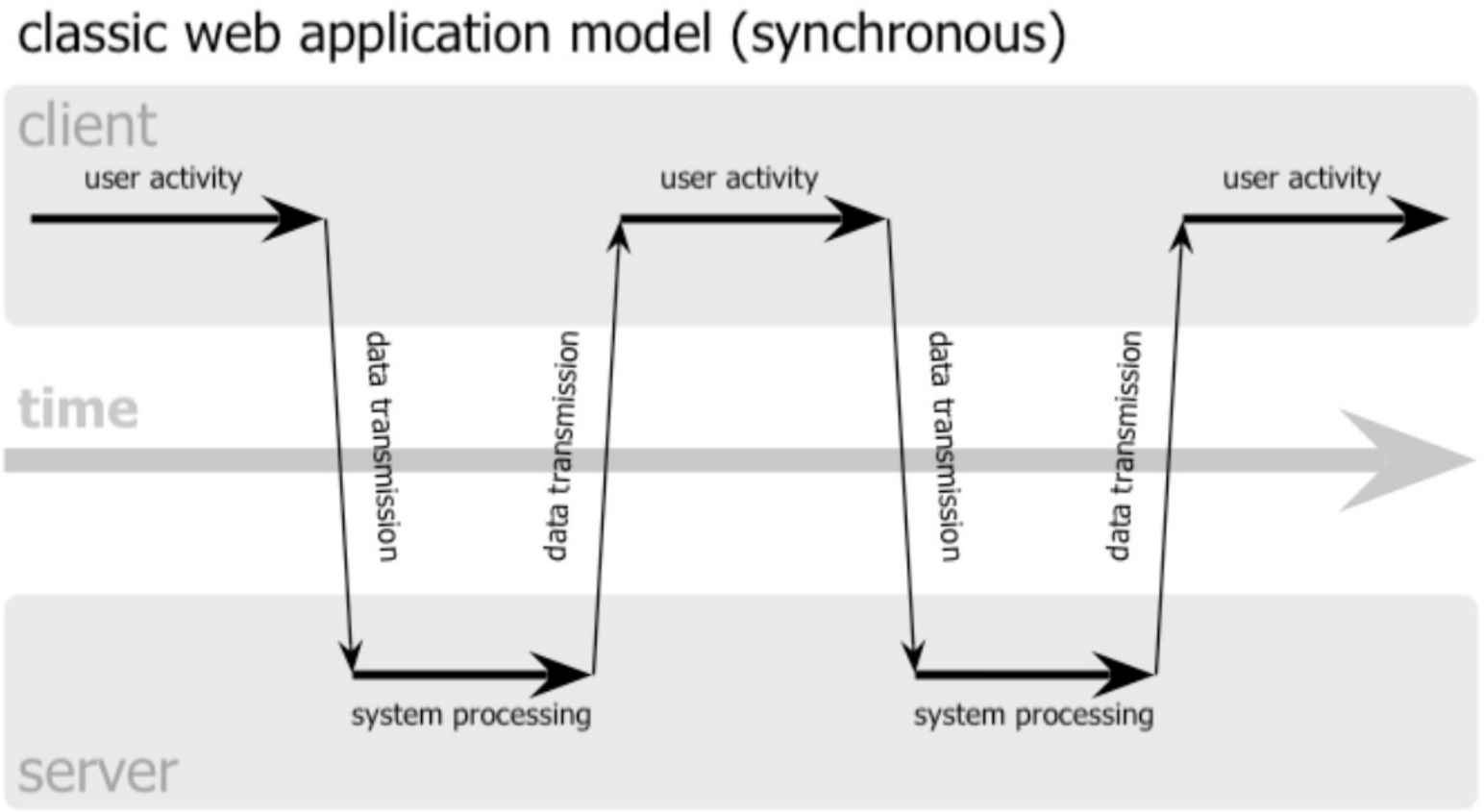
- Proporciona una interacción intuitiva y natural con el usuario.
  - No sigue el patrón "*Click, wait and refresh*".
  - El movimiento de ratón puede ser suficiente para disparar los eventos.

- Permite actualizar una página parcialmente sin necesidad de recargar la página completa.
  - Solo los elementos que contienen nueva información son actualizados, ofreciendo una respuesta más rápida.
  - El resto de la página no cambia, se mantiene el contexto operacional.

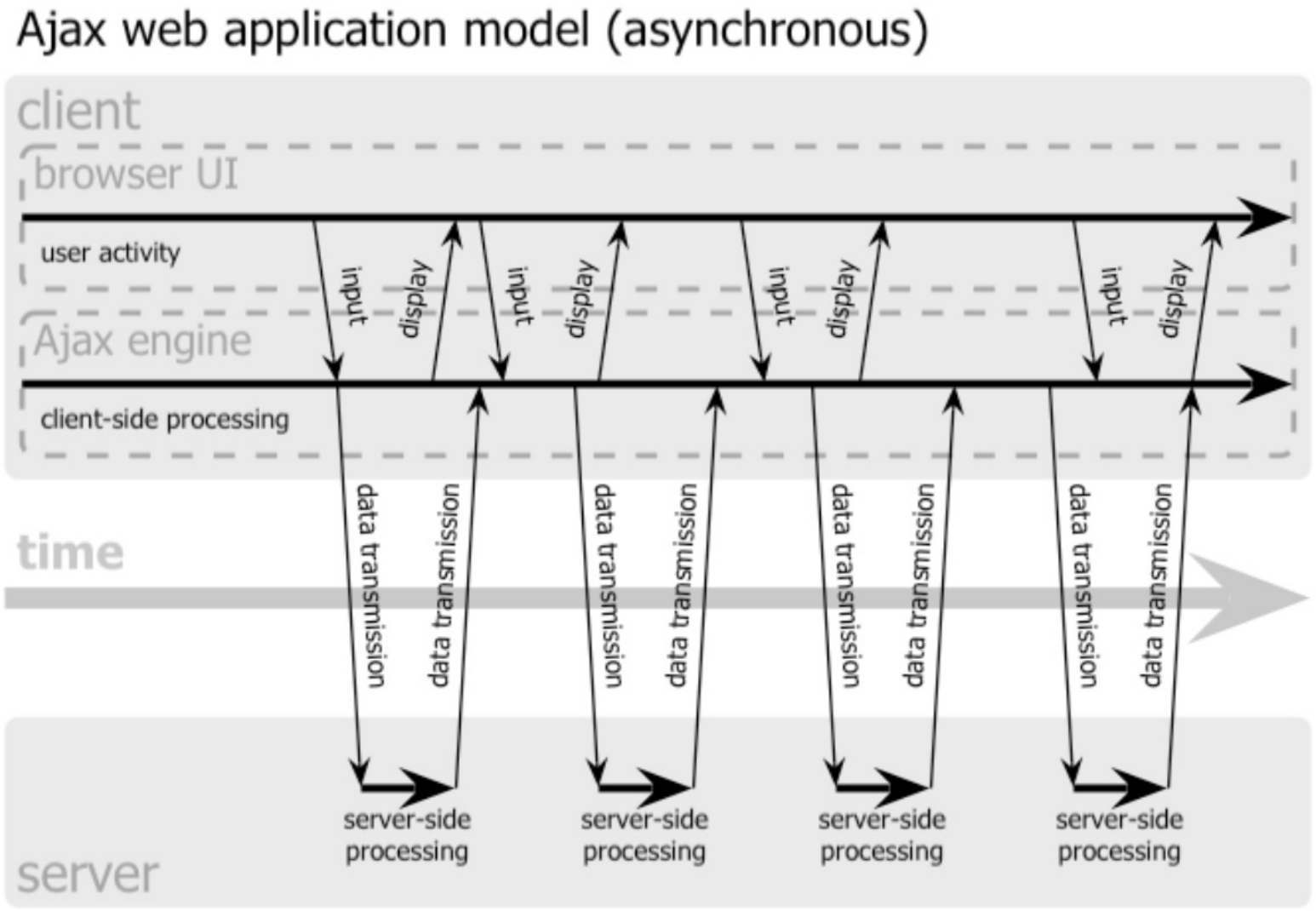
AJAX

- La comunicación asíncrona remplaza el modelo síncrono de petición-respuesta.
- El usuario puede continuar usando la aplicación mientras se solicita información al servidor.

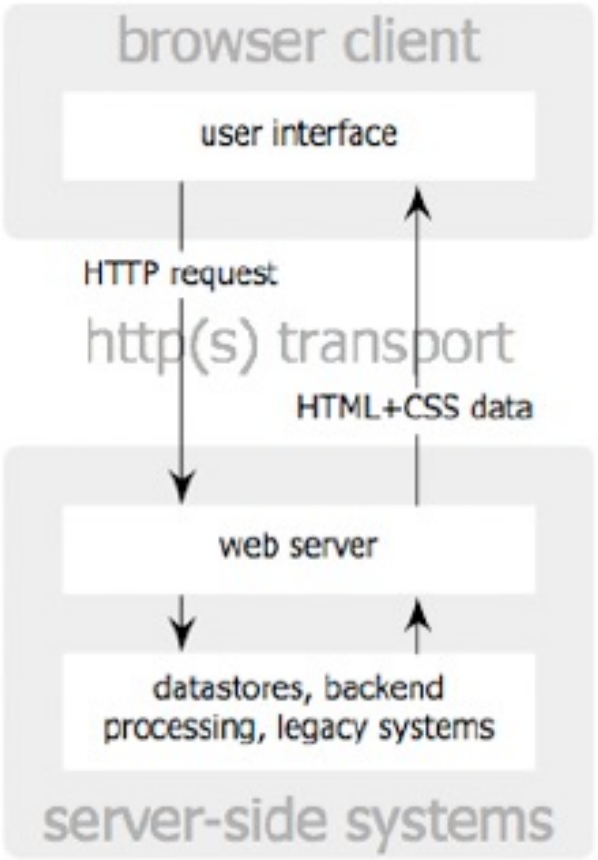
AJAX


classic web application model (synchronous)
client
user activity
user activity
user activity
data transmission
data transmission
data transmission
data transmission
time
system processing
system processing
server

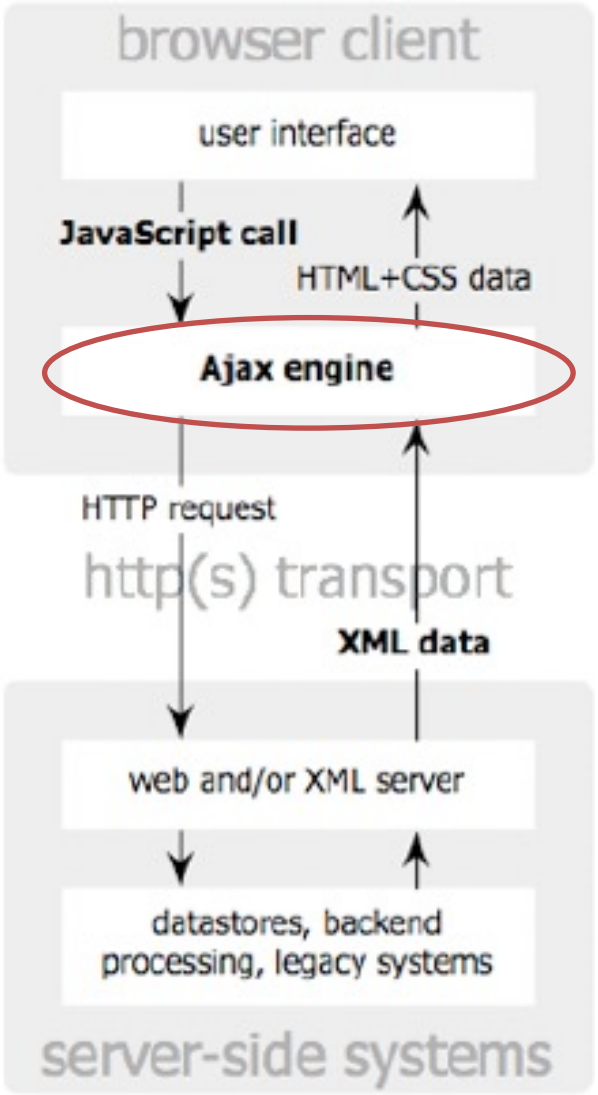
AJAX


Ajax web application model (asynchronous)
client
browser UI
user activity
input
display
input
display
input
display
input
display
Ajax engine
client-side processing
data transmission
data transmission
data transmission
data transmission
data transmission
data transmission
data transmission
data transmission
time
server-side processing
server-side processing
server-side processing
server-side processing
server

AJAX



classic
web application model



Ajax
web application model

AJAX

- AJAX:

- Conocimientos necesarios:
  - **JavaScript**;
  - XMLHttpRequest;
  - **HTML**;
  - **CSS**;
  - JSON;
  - **DOM**;

XMLHttpRequest

- Objeto JavaScript.
- Incluido en los navegadores modernos.
- Se comunica con el servidor siguiendo el protocolo HTTP.
- Realiza una comunicación asíncrona sin interrumpir las operaciones del usuario.

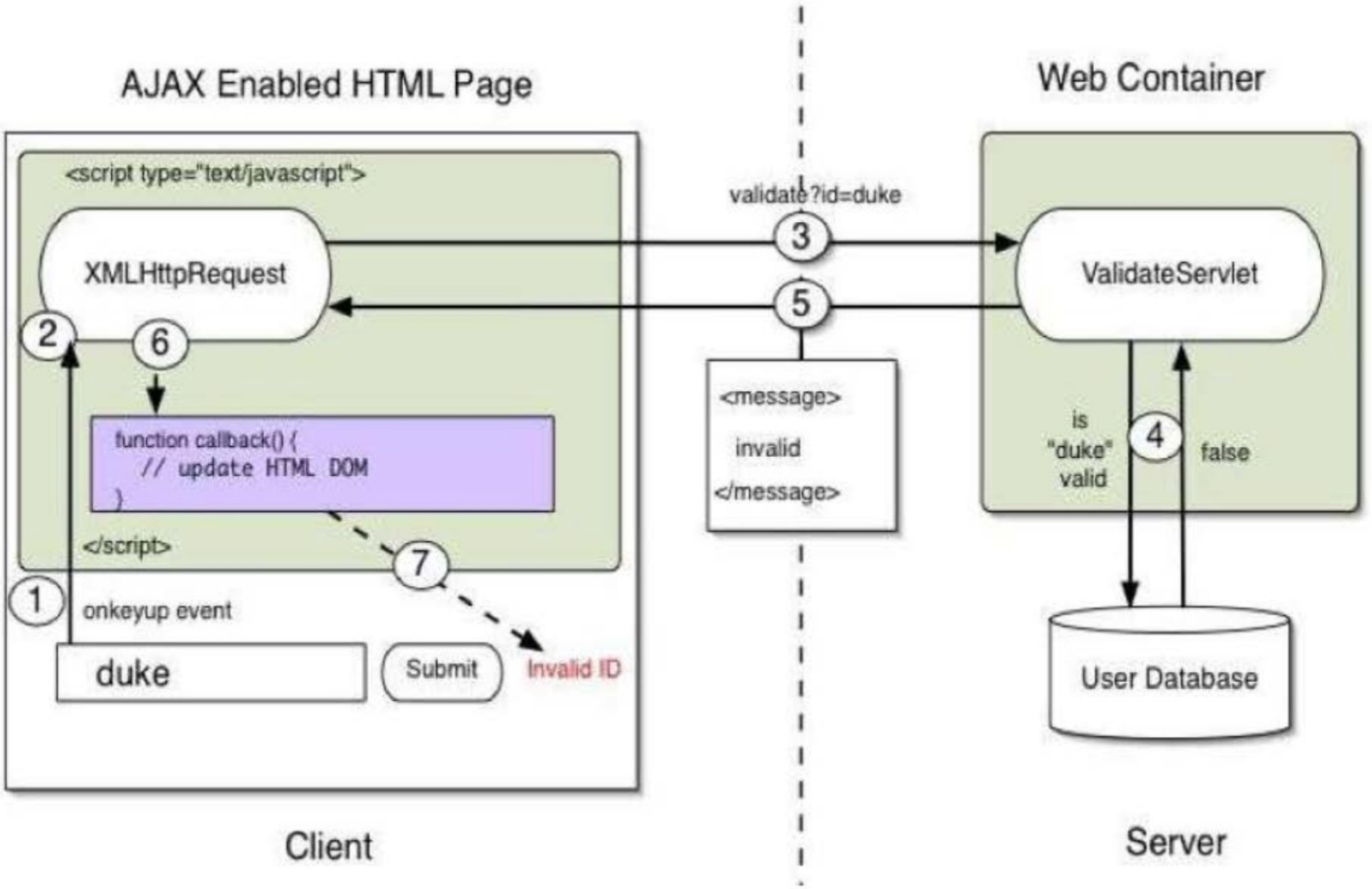
XMLHttpRequest

¿XML?

¿Y JSON?

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

XMLHttpRequest



XMLHttpRequest

- Pasos de una operación AJAX:
  - Se produce un evento en el cliente.
  - Se crea el objeto **XMLHttpRequest**.
  - Se configura el objeto **XMLHttpRequest**.
  - El objeto **XMLHttpRequest** realiza una petición asíncrona.
  - El servidor devuelve la respuesta a la petición realizada.
  - El objeto **XMLHttpRequest** llama a la función de *callback* y procesa el resultado.
  - Se utiliza DOM para actualizar el HTML.

XMLHttpRequest

- Se produce un evento en el cliente.
- Una función JavaScript es ejecutada para manejar el **evento**.

```
<input type="text"
       size="20"
       id="userid"
       name="id“ onkeyup="validateUserId();">
```

XMLHttpRequest

- Se crea el objeto XMLHttpRequest.

```
function validateUserId() {
        initRequest();
        …
}
var req;
function initRequest() {
        if (window.XMLHttpRequest) {
                req = new XMLHttpRequest();
        } else if (window.ActiveXObject) {
                isIE = true;
                req = new
ActiveXObject("Microsoft.XMLHTTP");
}}
```

XMLHttpRequest

- El objeto XMLHttpRequest realiza una petición asíncrona.

```
function validateUserId() {
    initRequest();
    req.onreadystatechange = processRequest;
}
```

XMLHttpRequest

- Se configura el objeto XMLHttpRequest.

```
function validateUserId() {
        initRequest();
        req.onreadystatechange = processRequest;
        if (!target)
                target = document.getElementById("userid");
        var url = "validate?id=" + target.value;
        req.open("GET", url, true);
        req.send(null);
}
```

XMLHttpRequest

- El servidor devuelve la respuesta a la petición realizada.

```
public void doGet(HttpServletRequest request,
                                HttpServletResponse response) {
    String targetId = request.getParameter("id");
    if ((targetId != null) && !accounts.containsKey(targetId.trim())) {
        response.setContentType("text/xml");
        response.setHeader("Cache-Control", "no-cache");
        response.getWriter().write("<valid>true</valid>");
    } else {
        response.getWriter().write("<valid>false</valid>");
    }
}
```

XMLHttpRequest

- El objeto XMLHttpRequest llama a la función de *callback* y procesa el resultado.

- El objeto ha sido configurado para llamar al método ***processRequest()*** cuando se produzca un cambio en ***readyState***.

```
function processRequest() {
    if (req.readyState == 4) {
        if (req.status == 200) {
            var message = ...;
```

XMLHttpRequest

- Se utiliza DOM para actualizar el HTML.
- Se utiliza también para informar al usuario de la respuesta producida por el servidor.

XMLHttpRequest

- **readyState**: Estado en el que se encuentra el objeto:
  - 0 : Sin inicializar.
  - 1 : Abierto.
  - 2 : Enviado.
  - 3 : Recibiendo.
  - 4 : Completado .

XMLHttpRequest

- **responseText**:
  - Contiene la respuesta en forma de cadena de texto.

- **responseXML**:
  - Contiene la respuesta en forma de documento XML. Contiene la representación en memoria, puede utilizarse DOM.

- **Status**:
  - Estado del protocolo HTTP devuelto por el servidor.
  - Correcto 200.
  - Not found 404.

XMLHttpRequest

- **statusText**:
  - Devuelve el estado devuelto por el servidor en una cadena de texto.
  - Ok.
  - Not found.

XMLHttpRequest

- **abort()**:
  - Cancela la petición actual.

- **getAllResponseHeaders()**:
  - Devuelve el conjunto de cabeceras HTTP como una cadena.

- **getResponseHeader (headerName)**:
  - Devuelve el valor de la cabecera HTTP especificada.

- **send (content)**:
  - Envía la petición incluyendo una cadena o un objeto DOM.

XMLHttpRequest

- **open**(method, URL), **open**(method, URL, async).
  - method: Método HTTP a utilizar (GET, POST, PUT, ...).
  - URL: Dirección absoluta o relativa a la que realizar la petición.
  - Async: Indica si la petición debe realizarse de forma asíncrona.

XMLHttpRequest

- **setRequestHeader(label,value)**:
  - Añade un par clave-valor a la cabecera HTTP que se va a enviar.

- **Onreadystatechange**:
  - Especifica un manejador de evento para gestionar los cambios en el estado del objeto.

XMLHttpRequest

- **Ejemplos:**

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

https://www.studentstutorial.com/ajax/introduction

- **jQuery + AJAX:**

https://www.w3schools.com/jquery/ajax_ajax.asp