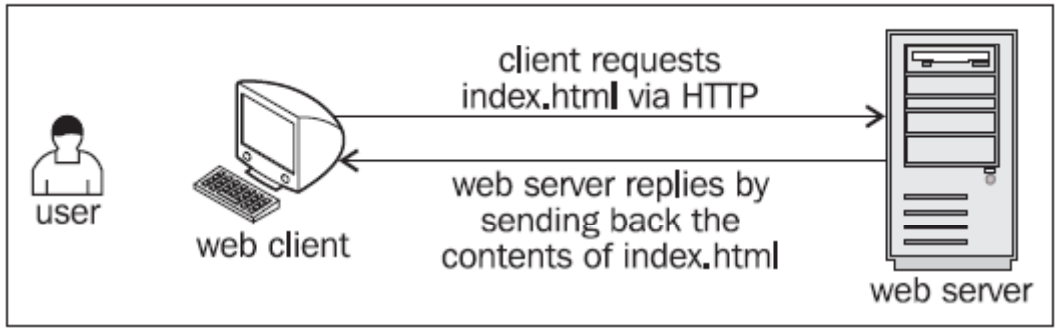
UD3. AJAX

1. HTTP vs HTML
2. Visión general de AJAX
3. DOM
4. Paso de parámetros y manejo de errores
5. Acceso a servidores remotos

1. HTTP vs HTML

- HTTP es un protocolo soportado por todos los navegadores Web.
  - Es el protocolo que se asume por defecto, si escribo www.marca.es , se asume http://www.marca.es
- HTML es un formato de texto estandarizado para pasar contenido Web a los navegadores, generalmente, a través del protocolo HTTP.



1. HTTP vs HTML

- HTML no fue pensado para aplicaciones Web complejas ni de gran interacción con el usuario.
- Si se quiere cambiar sólo una cosa de la pág. Web que ve el cliente ahora mismo, se tiene que hacer una petición al servidor y éste recarga la página HTML por completo.
  - Recordar que cuando trabajo con PHP, en última instancia, lo que ve el usuario es una página HTML.

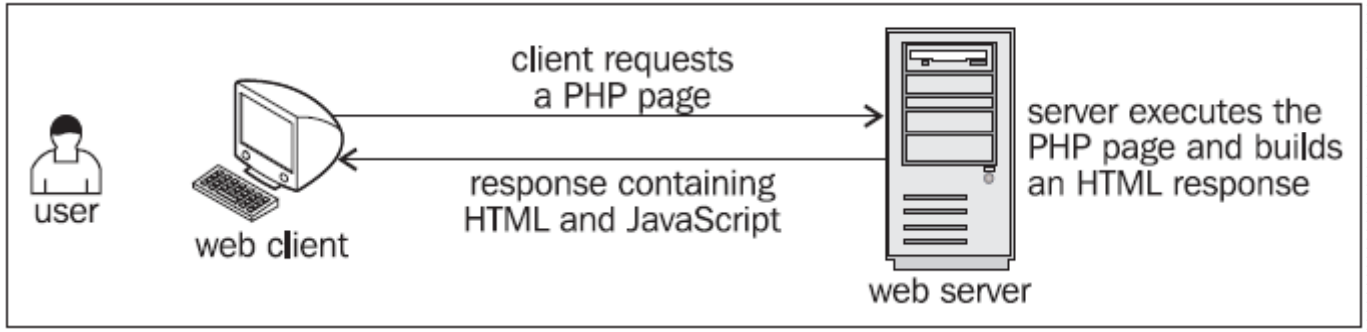
1. HTTP vs HTML

- Dado que la combinación HTTP/HTML es tan limitada, han aparecido tecnologías complementarias:
  - Por el lado de servidor: PHP, JSP, Perl, ASP.NET
  - Por el lado de cliente: Javascript, Flash
- Usando una combinación de todo esto, se pueden llegar a construir aplicaciones Web potentes.

¿Y entonces? ¿Qué nos falta?

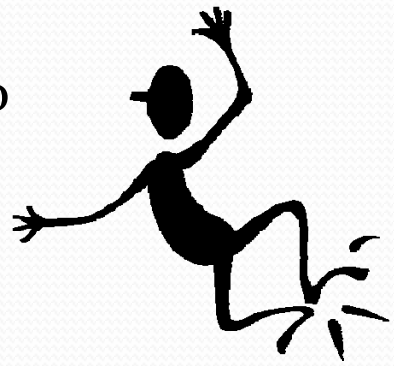
1. HTTP vs HTML

- Parte del mercado, quiere aplicaciones Web todavía mejores
  - Esto les lleva a sobrecargar el cliente con Flash y Applets.
  - Los que no ven factible esta opción, usan la combinación: HTML, JavaScript y PHP
    - Esta combinación tiene un PROBLEMA.



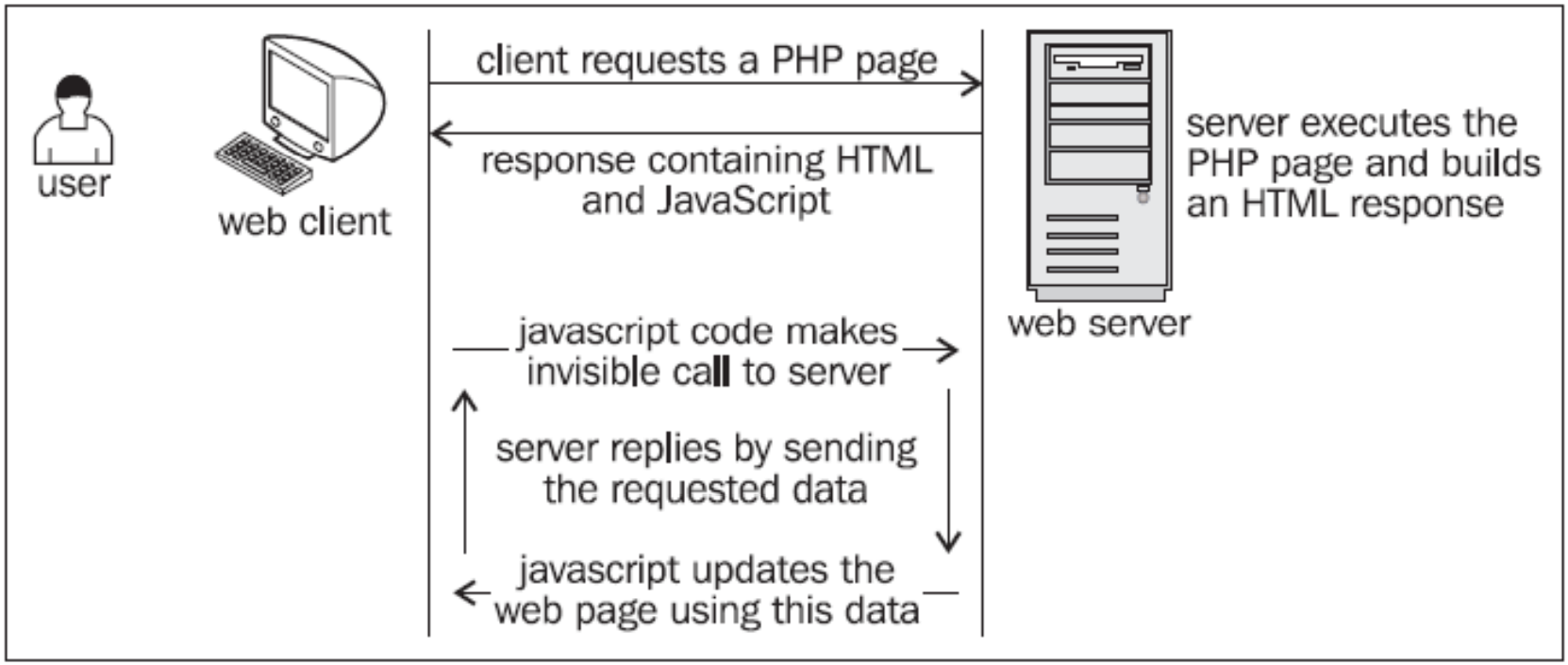
1. HTTP vs HTML

- PROBLEMA
  - Cada vez que el cliente necesita nuevos datos del servidor, necesita hacer una petición HTTP (HTTP Request)
    - La página tiene que esperar contestación del servidor y ser recargada completamente.
    - Esto ralentiza el flujo de trabajado del usuario
  - Una solución a este problema → **AJAX**



2. AJAX – Visión general

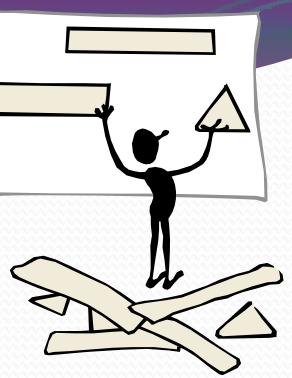
- AJAX – Asynchronous Javascript and XML



2. AJAX – Visión general

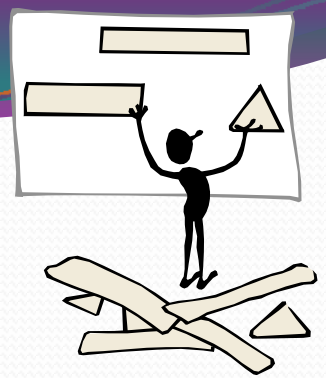
- Ejemplos de uso de AJAX:
  - http://www.gmail.com
  - http://maps.google.com
  - http://www.writely.com
- Como todas las tecnologías, AJAX puede usarse de forma incorrecta.
  - No por el hecho de usar AJAX nuestro portal ya es mejor.

2. AJAX – Visión general



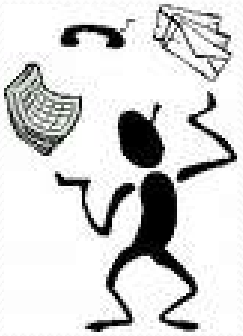
- AJAX es una combinación de estas tecnologías:
  - **JavaScript**
    - Permite manipular partes del HTML mediante DOM (Document Object Model)
  - El objeto **XMLHttpRequest**
    - Permite al JavaScript acceder al servidor de forma asíncrona.
      - El usuario puede seguir trabajando mientras se hacen estas llamadas.
  - Tecnología de servidor → en nuestro caso **PHP**
    - Recoge y trata las peticiones que llegan del cliente a través de JavaScript

2. AJAX – Visión general



- Para que exista **comunicación entre cliente y servidor**, se tienen que pasar los datos de forma que los entiendan las dos partes:
  - El cliente JavaScript envía los datos a través del objeto XMLHttpRequest usando los pares name-value, mediante GET o POST.
    - La forma de tratar esto en servidor ya la sabemos.
  - El servidor responde:
    - Hasta ahora con páginas HTML completas
    - Ahora debemos responder con un código que entienda JavaScript:
      - XML → es el que usaremos
      - JSON (JavaScript Object Notation) → podría ser otra alternativa

2. AJAX – Visión general



Ejemplo 1

Ejercicios

- **Ejercicio 1**
  - Partiendo del ejemplo anterior, crea otro que pregunte al usuario dos números y muestre la multiplicación de los dos.
- **Ejercicio 2**
  - Partiendo del ejemplo anterior, crea otro que pregunte al usuario un número y muestre la tabla de multiplicar de ese número.

3. DOM

- JavaScript es la parte más importante de AJAX.
  - Se supone que sabéis JavaScript del curso pasado.
- La importancia de JavaScript dentro de AJAX es que puede manipular el documento HTML.
  - Para conseguirlo usa DOM
  - DOM tiene la particularidad de poder manipular documentos XML y HTML es un tipo de XML.

3. DOM

- En el **lado cliente**, usaremos DOM para:
  - Manipular páginas HTML
  - Leer e interpretar documentos XML recibidos de servidor
  - Crear nuevos XML
- En el **lado servidor**, lo usaremos para:
  - Crear los XML que enviamos al cliente
  - Leer XML recibidos de otras fuentes

3.1. DOM – En cliente

- **Ejemplo de manipulación de HTML usando DOM**
  - Este ejemplo no usa PHP, podemos probarlo sin arrancar el servidor.

```
<html>
<head>
  <script type="text/javascript">
    var fecha = new Date();
    var hora = fecha.getHours();
    if (hora>=22 || hora<=6)
        document.write("A dormir!");
    else
        document.write("Bienvenido!");
  </script>
</head>
<body></body>
</html>
```

3.1. DOM – En cliente

- ¿Qué ocurre si escribo algo en el <BODY>?
- ¿Tengo manera de intercalar el código de JavaScript con el del BODY?
- La idea es poder intercalar código JavaScript pero, ¿cómo se hace?

3.1. DOM – En cliente

- **Ejemplo de intercalar código usando DOM**
  - Partimos de este HTML

Hola! Puedes escoger entre estos colores:

- Negro
- Naranja
- Rosa

```
<html>
<head></head>
<body>Hola! Puedes escoger entre estos colores:<br/>
<ul><li>Negro</li><li>Naranja</li><li>Rosa</li></ul>
</body>
</html>
```

  - Queremos poder generar la lista de colores de forma dinámica usando AJAX.

3.1. DOM – En cliente

- **Ejemplo de intercalar código usando DOM**
  - Vamos a crear una zona en la que el JavaScript escribirá:

```
<html>
<head></head>
<body>Hola! Puedes escoger entre estos colores:<br/>
<div id="elementoDIV" />
</body>
</html>
```

  - Después de que el JavaScript vuelque el resultado, será así:

```
<html>
<head></head>
<body>Hola! Puedes escoger entre estos colores:<br/>
<div id="elementoDIV" >
<ul><li>Negro</li><li>Naranja</li><li>Rosa</li></ul>
</div>
</body>
</html>
```

3.1. DOM – En cliente

- **Ejemplo de intercalar código usando DOM**
  - ¿Y cómo se hace?
    - Creamos un fichero → ejemplo2.js con este código

```
function process()
{// Crear código HTML
var string = "<ul><li>Negro</li><li>Naranja</li>"
+ "<li>Rosa</li></ul>";
// obtener la referencia al DIV
myDiv = document.getElementById("elementoDIV");
// añadir contenido al elemento DIV
myDiv.innerHTML = string;
}
```

3.1. DOM – En cliente

- **Ejemplo de intercalar código usando DOM**
  - ¿Y cómo se hace?
    - Modificamos el fichero index.html

```
<html>
<head>
<script type="text/javascript" src="ejemplo2.js">
</script>
</head>
<body onload="process()">
Hola! Puedes escoger entre estos colores:<br/>
<div id="elementoDIV" />
</body>
</html>
```

3.1. DOM – En cliente

- **Ejemplo de intercalar más completo**
  - Modificamos el fichero index.html, lo hacemos más sencillo, porque ahora DOM trabajará más:

```
<html>
<head>
<script type="text/javascript" src="ejemplo3.js"></script>
</head>
<body onload="process()">
<div id="elementoDIV" />
</body>
</html>
```

3.1. DOM – En cliente

- **Ejemplo de intercalar más completo**
  - La función process de ejemplo3.js será más elaborada:

```
function process()
{
// Crear el texto que aparece primero
oHello = document.createTextNode
("Hola! Puedes escoger entre estos colores:");
// crear el elemento UL
oUl = document.createElement("ul")
// Crear la primera entrada de la lista
oLiBlack = document.createElement("li");
oBlack = document.createTextNode("Negro");
oLiBlack.appendChild(oBlack);
// Crear la segunda entrada de la lista
oLiOrange = document.createElement("li");
oOrange = document.createTextNode("Naranja");
oLiOrange.appendChild(oOrange);
// Crear la tercera entrada de la lista
oLiPink = document.createElement("li");
oPink = document.createTextNode("Rosa");
oLiPink.appendChild(oPink);
```

```
// Añadimos las entradas en la lista
oUl.appendChild(oLiBlack);
oUl.appendChild(oLiOrange);
oUl.appendChild(oLiPink);
// obtenemos una referencia al elemento DIV
myDiv = document.getElementById("myDivElement");
// añadimos el contenido al DIV
myDiv.appendChild(oHello);
myDiv.appendChild(oUl);
}
```

Ejercicios

- **Ejercicio 3**
  - Usa el código anterior para mostrar el resultado del Ejercicio 2 utilizando listas.

- **Ejercicio 4**
  - Para el ejemplo de los colores, investiga si puede hacerse lo mismo utilizando una tabla en lugar de una lista.

3.2. Trabajar con XML

- Vamos a incluir ahora en el fichero JavaScript la creación del objeto **XMLHttpRequest**, a partir del que se hacen las llamadas a servidor.
- También vamos a crear un fichero XML, que se llamará libros.xml. Lo leeremos usando JavaScript y DOM.
- Este XML simula ser la respuesta que nos daría el servidor.

3.2. Trabajar con XML

- **Libros.xml**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<response>
<libros>
<libro>
<titulo>
PHP and MySQL for Dummies
</titulo>
</libro>
<isbn>
1-924811-32-5
</isbn>
</libro>
<libro>
<titulo>
Beginning PHP 5 and MySQL E-Commerce: From Novice to Professional
</titulo>
<isbn>
1-59059-392-8
</isbn>
</libro>
</libros>
</response>
```

3.2. Trabajar con XML

- Dejamos preparado el HTML que mostrará los libros:

```
<html>
<head>
<script type="text/javascript" src="ejemplo4.js"></script>
</head>
<body onload="process()">
Servidor, dame la lista de libros!
<br/>
<div id="elementoDIV" />
</body>
</html>
```

3.2. Trabajar con XML

- Finalmente, creamos el JavaScript (ejemplo4.js):
  - La parte de createXmlHttpRequestObject, es idéntica al primer ejemplo
  - Process →

```
// hace una llamada HTTP asícrona usando el objeto XMLHttpRequest
function process()
{
    // se ejecuta si no está vacío
    if (xmlHttp)
    {
        // Se intenta conectar al servidor
        try
        {
            //Empezamos a leer el xml
            xmlHttp.open("GET", "libros.xml", true);
            xmlHttp.onreadystatechange = handleServerResponse;
            xmlHttp.send(null);
        }
        // se muestra error en caso de fallo
        catch (e)
        {
            alert("Imposible conectar al servidor:\n" + e.toString());
        }
    }
}
```

3.2. Trabajar con XML

- Finalmente, creamos el JavaScript (ejemplo4.js):
  - handleServerResponse: (1/2)

```
// Se ejecuta automáticamente cuando se recibe un mensaje del servidor
function handleServerResponse()
{
    // Se cumple sólo cuando la transacción (el envío del mensaje) se ha completado
    if (xmlHttp.readyState == 4)
    {
        // el status 200 indica que la transacción se completó correctamente
        if (xmlHttp.status == 200)
        {
            // lee el mensaje del servidor
            var xmlResponse = xmlHttp.responseXML;
            // se obtiene el elemento raíz del XML
            var xmlRoot = xmlResponse.documentElement;
            // Sacamos el array de libros
            titleArray = xmlRoot.getElementsByTagName("titulo");
            isbnArray = xmlRoot.getElementsByTagName("isbn");
```

3.2. Trabajar con XML

- Finalmente, creamos el JavaScript (ejemplo4.js):
  - handleServerResponse: (2/2)

```
            // generamos la salida HTML
            var html = "";
            for (var i=0; i<titleArray.length; i++)
                html += titleArray.item(i).firstChild.data +", "
                + isbnArray.item(i).firstChild.data + "<br/>";
            // obtenemos la referencia del elemento DIV
            myDiv = document.getElementById("elementoDIV");
            // mostramos la salida
            myDiv.innerHTML = "Contestación: <br />" + html;
        }
        // si el HTTP status no es 200, significa que hubo un error
        else
        {
            alert("Problemas al acceder al servidor: " + xmlHttp.statusText);
        }
    }
}
```

3.3. DOM – En servidor

- Antes hemos visto cómo leer un XML situado en el mismo cliente.
- Ahora el XML lo vamos a construir y enviar desde servidor.
- Vamos a crear un fichero PHP que se llamará ejemplo4.php
- Del código que ya tenemos hecho, en process(), modificaremos:
  - xmlHttp.open("GET", “ejemplo4.php", true);

3.3. DOM – En servidor

- En ejemplo4.php, vamos a construir el PHP:
  - Para ver cómo se construye un XML desde PHP
  - Para ver cómo enviarlo

```php
<?php
    // Indicamos que la salida de este PHP no es HTML, sino XML
    header('Content-Type: text/xml');
    // Creamos un XML nuevo
    $dom = new DOMDocument();
    // creamos el nodo <response>
    $response = $dom->createElement('response');
    $dom->appendChild($response);
    // creamos el elemento <libros> y lo añadimos como hijo de <response>
    $books = $dom->createElement('libros');
    $response->appendChild($books);
    // creamos el elemento título del libro
    $title = $dom->createElement('titulo');
    $titleText = $dom->createTextNode
    ('PHP and MySQL for Dummies');
    $title->appendChild($titleText);
    //creamos el elemento ISBN de libro
    $isbn = $dom->createElement('isbn');
    $isbnText = $dom->createTextNode('1-934811-82-5');
    $isbn->appendChild($isbnText);
    //creamos el elemento libro
    $book = $dom->createElement('libro');
    $book->appendChild($title);
    $book->appendChild($isbn);
    // añadimos el elemento libro a la lista de libros.
    $books->appendChild($book);
    // build the XML structure in a string variable
    $xmlString = $dom->saveXML();
    // output the XML string
    echo $xmlString;
?>
```

3.3. DOM – En servidor

- Una vez construido ejemplo4.php, podemos acceder directamente para ver el XML resultante: http://127.0.0.1/AJAX_ej4/ejemplo4.php

```
<?xml version="1.0" ?>
- <response>
  - <libros>
    - <libro>
        <titulo>PHP and MySQL for Dummies</titulo>
        <isbn>1-934811-82-5</isbn>
      </libro>
    </libros>
  </response>
```

Ejercicios

- **Ejercicio 5**
  - A partir del ejemplo pilotos.xml colgado en el GIC y basándote en el ejemplo anterior, muestra los datos del XML.
  - Hazlo de dos formas:
    - Sin hacer acceso al servidor, accediendo directamente a un fichero XML
    - Haciendo acceso al servidor (a un PHP) y construyendo el XML desde allí.

4. Paso de parámetros y manejo de errores en servidor

- En el ejemplo anterior, recibimos un XML del servidor.
- En la vida real, también tendremos que ser capaces de enviarle datos al servidor y de hacer control de errores.
- Vamos a ver un ejemplo de cómo hacerlo.
- Necesitamos un HTML que nos permita introducir los datos: ejemplo5.html

```
<html>
<head>
<script type="text/javascript" src="ejemplo5.js"></script>
</head>
<body>Dividir
<input type="text" id="numero1" name="numero1" /> entre
<input type="text" id="numero2" name="numero2"/>
<input type="button" value="Enviar" onclick="process()" />
<div id="elementoDIV" />
</body>
</html>
```

4. Paso de parámetros y manejo de errores en servidor

- Necesitamos el JavaScript asociado: ejemplo5.js
  - La función process(), idéntica al resto pero colocando este código dentro del TRY

```
// obtiene los dos números dados por el usuario
var numero1 = document.getElementById("numero1").value;

var numero2 = document.getElementById("numero2").value;
// Creamos el querystring
var params = "numero1=" + numero1 + "&numero2=" + numero2;
// hacemos la llamada asíncrona
xmlHttp.open("GET", "ejemplo5.php?" + params, true);
xmlHttp.onreadystatechange = handleServerResponse;
xmlHttp.send(null);
```

4. Paso de parámetros y manejo de errores en servidor

- Necesitamos el JavaScript asociado: ejemplo5.js
  - La función handleServerResponse, idéntica al resto pero cambiando este código dentro del TRY

```
// lee el mensaje del servidor
var xmlResponse = xmlHttp.responseXML;
// captura de errores si no hay respuesta --> IE y Opera
if (!xmlResponse || !xmlResponse.documentElement)
    throw("XML no válido:\n" + xmlHttp.responseText);
// captura de errores con Firefox
var rootNodeName = xmlResponse.documentElement.nodeName;
if (rootNodeName == "parsererror")
    throw("XML no válido:\n" + xmlHttp.responseText);
// se obtiene el elemento raíz del XML
xmlRoot = xmlResponse.documentElement;
// comprobamos que hemos recibido el documento XML que esperábamos
if (rootNodeName != "response" || !xmlRoot.firstChild)
    throw("XML no válido:\n" + xmlHttp.responseText);
// El elemento hijo de <response> es el resultado que tenemos que mostrar.
responseText = xmlRoot.firstChild.data;
// mostramos el mensaje al usuario
myDiv = document.getElementById("elementoDIV");
myDiv.innerHTML = "Resultado dado por el servidor: " + responseText;
```

4. Paso de parámetros y manejo de errores en servidor

- El PHP → ejemplo5.php

```php
<?php
    // cargamos el php que trata los errores
    require('ejemplo5_manejoErrores.php');
    // especificamos que la salida es un documento XML
    header('Content-Type: text/xml');
    // calculamos el resultado
    $numero1 = $_GET['numero1'];
    $numero2 = $_GET['numero2'];
    $result = $numero1 / $numero2;
    // Creamos un documento XML nuevo
    $dom = new DOMDocument();
    // Creamos el nodo <response> y lo añadimos al documento
    $response = $dom->createElement('response');
    $dom->appendChild($response);
    // Añadimos el valos calculado como un nodo de testo hijo del nodo <response>
    $responseText = $dom->createTextNode($result);
    $response->appendChild($responseText);
    // Pasamos la estructura XML a una variable string
    $xmlString = $dom->saveXML();
    // mostramos el texto XML
    echo $xmlString;
?>
```

4. Paso de parámetros y manejo de errores en servidor

- En ejemplo5.php, recogemos los 2 números que nos llegan como parámetro y calculamos la división.
- Estamos añadiendo un fichero para el tratamiento de errores: ejemplo5_manejoErrores.php
- Este php contiene una función de usuario que sustituye al tratamiento de errores por defecto de PHP.
- Sencillamente, lo que hace es transformar el error en algo más legible para el usuario.

4. Paso de parámetros y manejo de errores en servidor

- Ejemplo5_manejoErrores.php

```php
<?php
    // Indicamos que el método de manejo de errores del usuario será el que trate los errores
    set_error_handler('error_handler', E_ALL);

    // función de manejo de errores
    function error_handler($errNo, $errStr, $errFile, $errLine)
    {
        // limpiamos cualquier salida que se haya generado
        if(ob_get_length()) ob_clean();
        // construimos y mostramos el mensaje de error
        $error_message = 'ERRNO: ' . $errNo . chr(10) .
        'TEXT: ' . $errStr . chr(10) .
        'LOCATION: ' . $errFile .
        ', line ' . $errLine;
        echo $error_message;
        // Impedimos que se procesen otros scripts PHP
        exit;
    }
?>
```

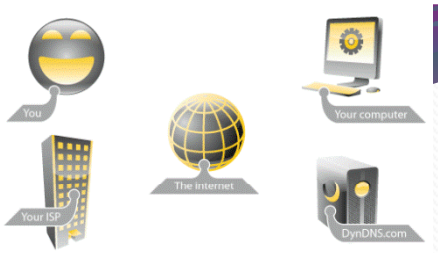
4. Paso de parámetros y manejo de errores en servidor

- Pruebas a hacer:
  - Comprobar que el PHP de cálculo va bien
    - http://127.0.0.1/AJAX_ej5/ejemplo5.php?numero1=9&numero2=3
  - Comprobar que el tratamiento de errores va bien
    - http://127.0.0.1/AJAX_ej5/ejemplo5.php?numero1=9&numero2=0
  - Los mismo desde el HTML
    - Siendo el numero1 9 y el número2, 3 → Da el resultado sin recargar la página
    - Probar poniendo el numero2 a cero → da un error
    - Probar lo mismo pero comentando la línea require('ejemplo5_manejoErrores.php'); de ejemplo5.php

5. Acceso a servidores remotos

- Existe una página que genera números aleatorios:
  - http://www.random.org
  - Para pedir un número aleatorio entre 1 y 100, decimos:
    - http://www.random.org/integers/?num=1&min=1&max=100&base=10&col=1&format=plain&rnd=new
  - Vamos a intentar crear una página HTML que muestre esto:

Servidor, dame un número aleatorio!
Se recibió este número aleatorio del servidor: 53



5. Acceso a servidores remotos

- El JavaScript será este:
  - Esto va dentro del TRY del process()

```
// para navegadores tipo Mozilla, pedimos permiso para acceder a servidores remotos
try
{
    //genera un error que ignoramos si el servidor no es mozilla
    netscape.security.PrivilegeManager.enablePrivilege('UniversalBrowserRead');
}
catch(e) {} // ignorar error
// url de llamada al servidor remoto con sus parámetros necesarios
var serverAddress = "http://www.random.org/integers/";
var serverParams = "num=1" + // cuántos números se quieren generar
"&min=1" + // desde este número
"&max=100"+ // hasta este otro
"&base=10&col=1&format=plain";
//iniciamos el acceso a servidor
xmlHttp.open("GET", serverAddress + "?" + serverParams, true);
xmlHttp.onreadystatechange = handleServerResponse;
xmlHttp.send(null);
```

5. Acceso a servidores remotos

- El JavaScript será este:
  - Esto va dentro del TRY del handleServerResponse()

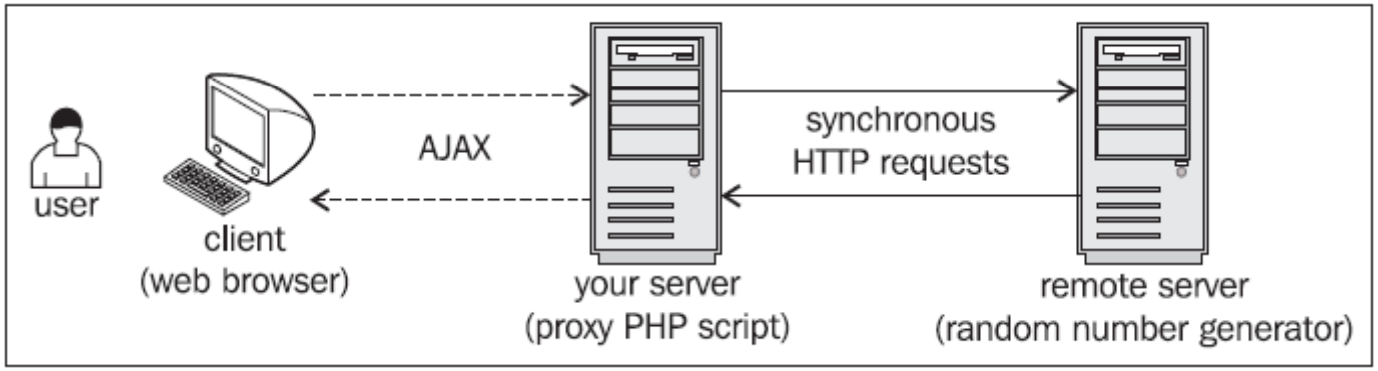
```
// lee el mensaje del servidor
var response = xmlHttp.responseText;
// obtenemos la referencia al elemento DIV
myDiv = document.getElementById("elementoDIV");
// sacamos la salida HTML
myDiv.innerHTML = "Se recibió este número aleatorio del servidor: "
+ response + "<br/>";
```

5. Acceso a servidores remotos

- Si probamos el código, nos tiene que dar un error.
  - Los navegadores (cada uno a su manera) controlan que los JavaScripts no accedan a código que está fuera de la máquina local.
  - Si se intenta, se da un error de seguridad.
  - La única forma de ejecutar este ejemplo, es hacerlo directamente clicando sobre el sistema de ficheros.

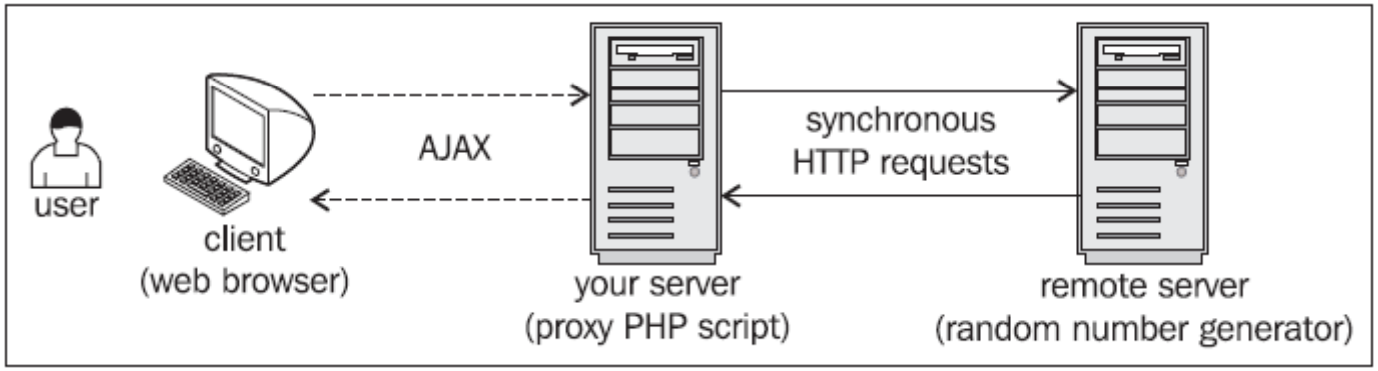
5. Acceso a servidores remotos

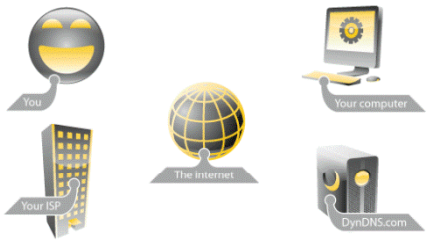
- Esta claro que esta no es la forma correcta de acceder al servidor random.org.
  - Necesitaremos utilizar un script de servidor que nos haga de proxy (a través de PHP)



5. Acceso a servidores remotos

- Esta claro que esta no es la forma correcta de acceder al servidor random.org.
  - Necesitaremos utilizar un script de servidor que nos haga de proxy (a través de PHP)





5. Acceso a servidores remotos

- Para que se nos permita hacer el acceso sin recibir errores de seguridad, hemos de usar un script PHP como este: (proxy.php)

```php
<?php
    // aseguramos que el navegador no tiene nada en caché
    header('Expires: Wed, 23 Dec 1980 00:30:00 GMT');
    header('Last-Modified: ' . gmdate('D, d M Y H:i:s') . ' GMT');
    header('Cache-Control: no-cache, must-revalidate');
    header('Pragma: no-cache');

    // recuperamos los parámetros
    $num = 1; // forzado por nosotros
    $min = $_GET['min'];
    $max = $_GET['max'];

    $serverAddress = "http://www.random.org/integers/";
    $serverParams = "num=" . $num . "&min=". $min . "&max=". $max . "&base=10&col=1&format=plain";

    // recuperar el número generado por el servidor externo
    $randomNumber = file_get_contents($serverAddress . '?' . $serverParams);

    // da el número aleatorio
    echo $randomNumber;
?>
```

Ejercicios

- **Ejercicio 6**
  - A partir del código del proxy dado en la diapo anterior, modifica el código para que se permita hacer accesos al servidor random sin problemas de seguridad.