

AJAX

Heri Fernando Londoño

AJAX

El término AJAX es un acrónimo de Asynchronous JavaScript + XML. Se puede traducir como "JavaScript asíncrono + XML."

- No es ninguna tecnología, ni lenguaje de programación
- Es una técnica de desarrollo web que combina varias tecnologías
- Consiguiendo una navegación más ágil y rápida, más dinámica.

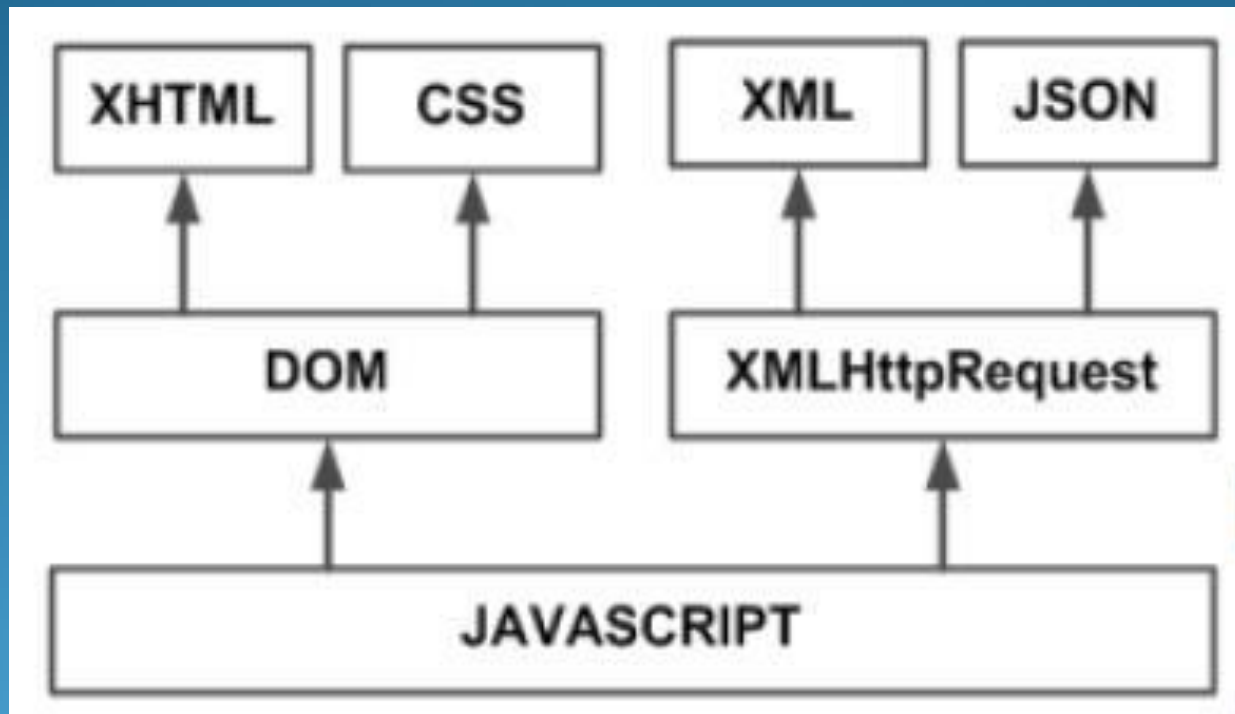
Heri Fernando Londoño

TECNOLOGÍAS EMPLEADAS

- XHTML y CSS, Presentación de datos.
- Document Object Model (DOM). Mostrar e interactuar dinámicamente con la información.
- XML, XSLT y JSON. Intercambio y manipulación de información.
- XMLHttpRequest. Recuperación y envío de datos de modo asíncrono.
- JavaScript. Unir todas las demás tecnologías.

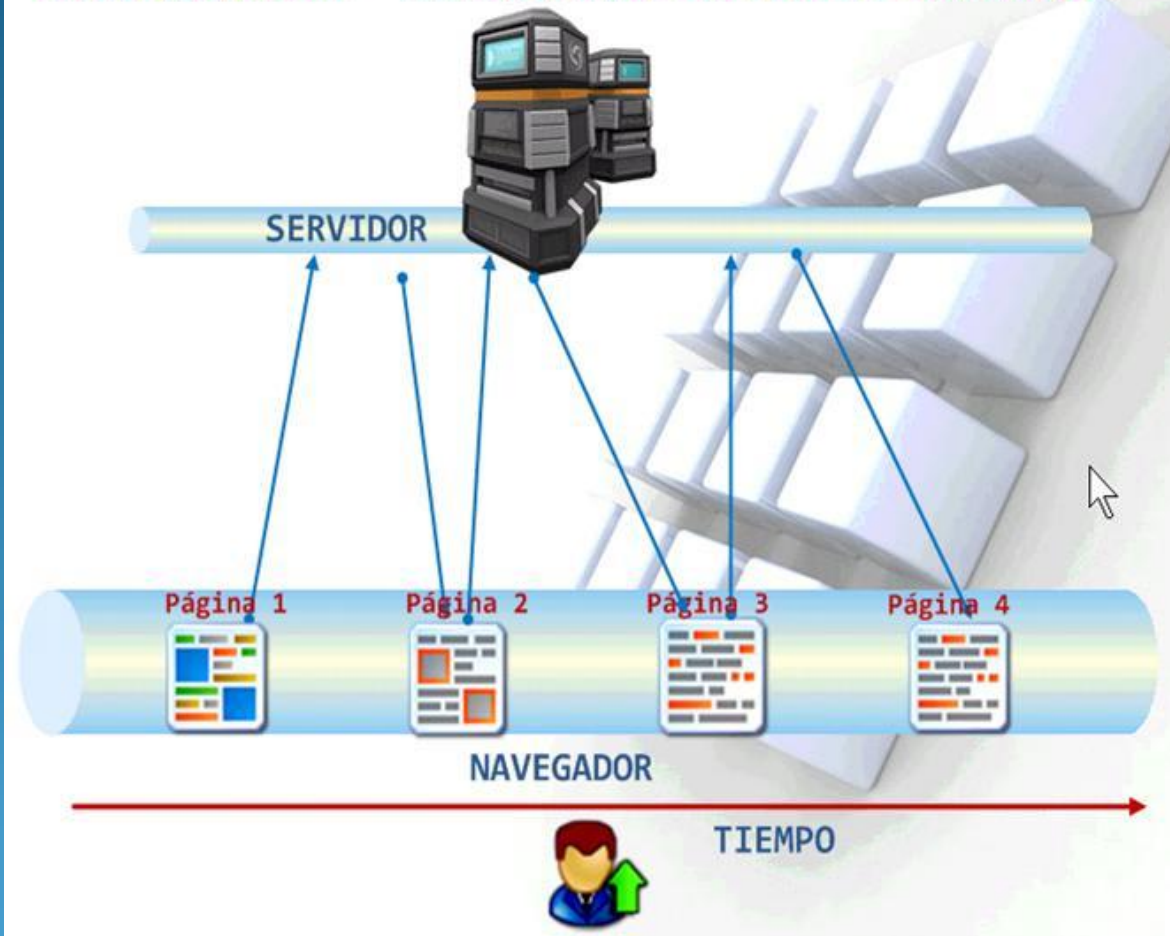
Heri Fernando Londoño

TECNOLOGÍAS EMPLEADAS

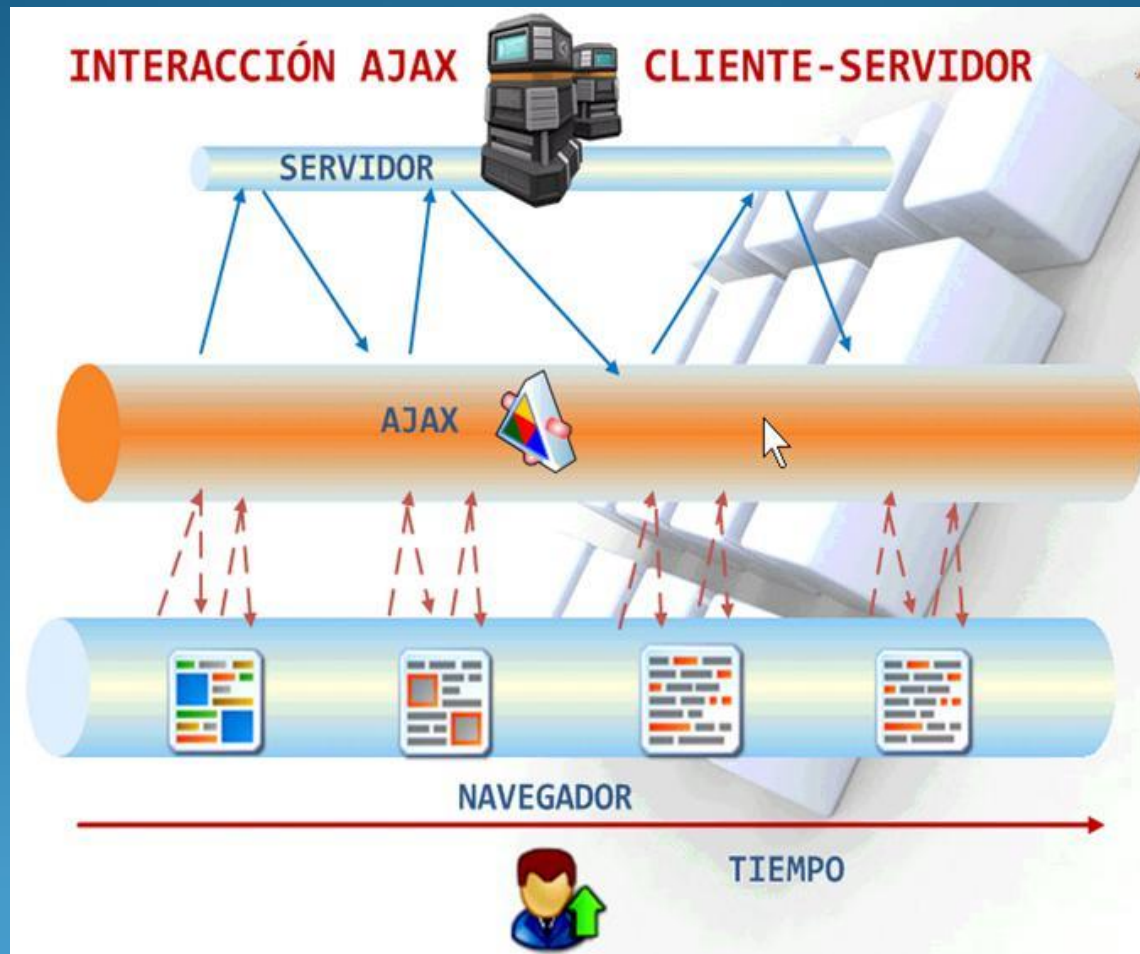


Heri Fernando Londoño

“CONVERSACIÓN” TRADICIONAL CLIENTE-SERVIDOR



Heri Fernando Londoño



Heri Fernando Londoño

VENTAJAS

- Mayor interactividad. Recuperación asíncrona de datos, reduciendo el tiempo de espera del usuario
- Se reduce el tamaño de la información intercambiada
- Portabilidad entre plataformas. No requieren instalación de plugins, applets de Java, ni ningún otro elemento
- Código público

Heri Fernando Londoño

USOS DE AJAX

- Validación de datos de formularios en tiempo real.
- Autocompletado. Correo, nombres, ciudades.
- Operaciones de detalle. Obtener información más detallada de un producto.
- GUI avanzadas
- Refresco de datos
- Actualizar o eliminar registros
- Expandir formularios web
- Devolver peticiones simples de búsqueda

Heri Fernando Londoño

FUNCIONAMIENTO DE AJAX

1. Usuario provoca un evento
2. Se crea y configura un objeto XMLHttpRequest
3. Se asigna un manejador de evento.
4. El objeto XMLHttpRequest realiza una llamada al servidor
5. La petición se procesa en el servidor
6. El servidor retorna el resultado a través de la propiedad responseText o responseXML.
7. Se actualiza la página web con el resultado devuelto

Heri Fernando Londoño

Usuario provoca un evento


- **onabort.** Se produce cuando se detiene la carga de una imagen.
- **onblur.** Se desata cuando un elemento pierde el foco de la aplicación.
- **onchange.** Se desata cuando cambia el estado de un elemento de formulario
- **onclick.** Se produce cuando se da una pulsación o clic al botón del ratón sobre un elemento de la página, generalmente un botón o un enlace.
- **onfocus.** Se produce cuando un elemento de la página o la ventana ganan el foco de la aplicación.
- **onkeydown.** Se produce en el instante que un usuario presiona una tecla.
- **onkeypress.** Ocurre cuando el usuario deja pulsada una tecla un tiempo determinado.

- **onkeyup.** Se produce cuando el usuario deja de apretar una tecla.
- **onload.** Ocurre cuando la página ha terminado de cargar.
- **onmousedown.** Se produce cuando el usuario pulsa sobre un elemento de la página.
- **onmousemove.** Se produce cuando el ratón se mueve por la página.
- **onmouseout.** Ocurre cuando el puntero del ratón sale del área de un elemento de la página.
- **onmouseover.** Ocurre cuando el puntero del ratón entra en el área ocupada por un elemento de la página.
- **onmouseup.** Se produce en el momento que el usuario suelta el botón del ratón.
- **onmove.** Se ejecuta cuando se mueve la ventana del navegador, o un frame.

- **onresize.** Se produce cuando se redimensiona la ventana del navegador, o el frame.
- **onreset.** Se desata en el momento que un usuario hace clic en el botón de reset de un formulario.
- **onselect.** Se ejecuta cuando un usuario realiza una selección de un elemento de un formulario.
- **onsubmit.** Ocurre cuando el visitante da clic sobre el botón de enviar el formulario. Se ejecuta antes del envío propiamente dicho.
- **onunload.** Al abandonar una página, ya sea porque se pulse sobre un enlace que nos lleve a otra página o porque se cierre la ventana del navegador.

Se crea y configura un objeto XMLHttpRequest


Microsoft lo introdujo en IE 5 como un objeto ActiveX



Algunas versiones de IE tienen versión diferente de MSXML

Resto navegadores lo incluyen como objeto nativo de JavaScript

```
var pedir = new ActiveXObject ("Microsoft.XMLHTTP");
```



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

```
var pedir = new ActiveXObject ("msxml2.XMLHTTP");
```

Solución 2 Detectar navegador

```
function obtenerAjax(){  
    if (window.XMLHttpRequest) {  
        xhttp = new XMLHttpRequest();  
    }  
    else {  
        xhttp = new ActiveXObject("Microsoft.XMLHTTP");  
    }  
    return xhttp;  
}  
var ajax = obtenerAjax();
```

El objeto XMLHttpRequest realiza la llamada al servidor

Métodos	Descripción
<code>abort ()</code>	Detiene la petición actual.
<code>getAllResponseHeaders ()</code>	Devuelve todas las cabeceras como un string.
<code>getResponseHeader (x)</code>	Devuelve el valor de la cabecera x como un string
<code>open ('method', 'URL', 'a')</code>	Especifica el método HTTP (por ejemplo, GET o POST), la URL objetivo, y si la petición debe ser manejada asincrónicamente (Sí, a= 'True' defecto; No, a='false'.)
<code>send (content)</code>	Envía la petición, opcionalmente con datos POST
<code>setRequestHeader ('x', 'y')</code>	Configura un par parámetro y valor x=y y lo asigna a la cabecera para ser enviado con la petición.

El objeto XMLHttpRequest realiza la llamada al servidor

Propiedades	Descripción
onreadystatechange	Determina que gestor de evento será llamado cuando la propiedad readyState del objeto cambie.
readyState	Número entero que indica el estatus 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 estatus HTTP devuelto por el servidor
statusText	"Phrase reason" HTTP devuelta por el servidor

200: "OK"
403: "Forbidden"
404: "Not Found"

index.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <script src="script.js" type="text/javascript" defer></script>
  <title>AJAX</title>
</head>
<body>
  <input type="button" value="Solicitar" onclick="peticionServidor()">
  <br>
  En la siguiente línea se muestra el texto del servidor
  <div id="verrespuesta"></div>
</body>
</html>
```

respuesta.php

```
<?php
    echo "Respuesta desde el servidor";
?>
```

script.js

```
function obtenerAjax(){
    if (window.XMLHttpRequest) {
        xhttp = new XMLHttpRequest();
    }
    else {
        xhttp = new ActiveXObject("Microsoft.XMLHTTP");
    }
    return xhttp;
}
var ajax = obtenerAjax();
function peticionServidor(){
    ajax.open("GET", "respuesta.php", true);
    ajax.onreadystatechange = respuestaServidor;
    ajax.send(null);
}
function respuestaServidor(){
    if(ajax.readyState == 4 && ajax.status == 200){
        document.getElementById("verrespuesta").innerHTML = ajax.responseText;
    }
}
```

Envío de parámetros con la petición HTTP

XMLHttpRequest permite enviar parámetros tanto con el método GET como con el método POST de HTTP. En ambos casos, los parámetros se envían como una serie de pares clave/valor concatenados por símbolos &.

La principal diferencia entre ambos métodos es que mediante el método POST los parámetros se envían en el cuerpo de la petición y mediante el método GET los parámetros se concatenan a la URL accedida.

Técnicamente, el método GET tiene un límite en la cantidad de datos que se pueden enviar. Si se intentan enviar más de 512 bytes mediante el método GET, el servidor devuelve un error con código 414 y mensaje Request-URI Too Long ("La URI de la petición es demasiado larga").

Envío de parámetros con la petición HTTP

Cuando se utiliza un formulario, al pulsar sobre el botón de envío del formulario, se crea automáticamente la cadena de texto que contiene todos los parámetros que se envían al servidor. Sin embargo, el objeto XMLHttpRequest no dispone de esa posibilidad y la cadena que contiene los parámetros se debe construir manualmente.

```
function obtenerQueryString()  
{  
    var usuario = document.getElementById("usuario");  
    var contra = document.getElementById("contra");  
    return "usuario=" + encodeURIComponent(usuario.value) +  
        "&contra=" + encodeURIComponent(contra.value) +  
        "&nocache=" + Math.random();  
}
```

Envío de parámetros con la petición HTTP

Si el método va a ser GET, se le agrega el queryString a la dirección url, acompañado de ?.

Si el método va a ser POST, es necesario agregar el método `setRequestHeader` y enviar el queryString como parametro en el método `send`

```
var queryString = obtenerQueryString();  
peticion_http.open("GET", "validaDatos.php?" + queryString, true);  
peticion_http.send(null);  
  
var queryString = obtenerQueryString();  
peticion_http.open("POST", "validaDatos.php", true);  
peticion_http.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");  
peticion_http.send(queryString);
```

Mostrar tabla desde PHP

index.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <script src="script.js" type="text/javascript" defer></script>
  <title>AJAX</title>
</head>
<body>
  <input type="button" value="Solicitar" onclick="peticionServidor()">
  <br>
  En la siguiente línea se muestra la tabla
  <div id="vertabla"></div>
</body>
</html>
```

Mostrar tabla desde PHP

respuesta.php

```
<?php
$productos = array(
    array('nombre' => 'Arroz', 'precio' => 1500),
    array('nombre' => 'Papa', 'precio' => 900),
    array('nombre' => 'Panela', 'precio' => 3500)
);
echo "<table><tr><th>Producto</th><th>Precio</th></tr>";
for($i = 0; $i < count($productos); $i++){
    echo "<tr><td>" . $productos[$i]["nombre"] . "</td>";
    echo "<td>" . $productos[$i]["precio"] . "</td></tr>";
}
echo "</table>";
?>
```

Mostrar tabla desde PHP

script.js

```
function obtenerAjax(){
    if (window.XMLHttpRequest) {
        xhttp = new XMLHttpRequest();
    }
    else {
        xhttp = new ActiveXObject("Microsoft.XMLHTTP");
    }
    return xhttp;
}
var ajax = obtenerAjax();
function peticionServidor(){
    ajax.open("GET", "respuesta.php", true);
    ajax.onreadystatechange = respuestaServidor;
    ajax.send(null);
}
function respuestaServidor(){
    if(ajax.readyState == 4 && ajax.status == 200){
        document.getElementById("vertabla").innerHTML = ajax.responseText;
    }
}
```


Retornar JSON Crear Tabla en JS

index.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <script src="script.js" type="text/javascript" defer></script>
  <title>AJAX</title>
</head>
<body>
  <input type="button" value="Solicitar" onclick="peticionServidor()">
  <br>
  En la siguiente línea se muestra la tabla
  <div id="vertabla"></div>
</body>
</html>
```

Retornar JSON Crear Tabla en JS

respuesta.php

```
<?php
    $productos = array(
        array('nombre' => 'Arroz', 'precio' => 1500),
        array('nombre' => 'Papa', 'precio' => 900),
        array('nombre' => 'Panela', 'precio' => 3500)
    );
    echo json_encode($productos);
?>
```

```
[{"nombre":"Arroz","precio":1500}, {"nombre":"Papa","precio":900}, {"nombre":"Panela","precio":3500}]
```

Retornar JSON Crear Tabla en JS

script.js

```
function obtenerAjax(){
    if (window.XMLHttpRequest) {
        xhttp = new XMLHttpRequest();
    }
    else {
        xhttp = new ActiveXObject("Microsoft.XMLHTTP");
    }
    return xhttp;
}
var ajax = obtenerAjax();
function peticionServidor(){
    ajax.open("GET", "respuesta.php", true);
    ajax.onreadystatechange = respuestaServidor;
    ajax.send(null);
}
function respuestaServidor(){
    if(ajax.readyState == 4 && ajax.status == 200){
        var respuesta = JSON.parse(ajax.responseText);
        cargarEncabezado();
        cargarDatos(respuesta);
    }
}
```

Heri Fernando Londoño

Retornar JSON Crear Tabla en JS

script.js

```
function cargarEncabezado(){  
    var contenedor = document.getElementById("vertabla");  
    var tabla = document.createElement("table");  
    tabla.setAttribute("id", "tabla");  
    contenedor.appendChild(tabla);  
    var fila1 = document.createElement("tr");  
    tabla.appendChild(fila1);  
    var celda11 = document.createElement("th");  
    fila1.appendChild(celda11);  
    var texto11 = document.createTextNode("Producto");  
    celda11.appendChild(texto11);  
    var celda12 = document.createElement("th");  
    fila1.appendChild(celda12);  
    var texto12 = document.createTextNode("Precio");  
    celda12.appendChild(texto12);  
}
```

Retornar JSON Crear Tabla en JS

script.js

```
function cargarDatos(respuesta){  
    tabla = document.getElementById("tabla");  
    for(var i = 0; i < respuesta.length; i++){  
        var fila = document.createElement("tr");  
        tabla.appendChild(fila);  
        var celda1 = document.createElement("td");  
        fila.appendChild(celda1);  
        var nombre = document.createTextNode(respuesta[i].nombre);  
        celda1.appendChild(nombre);  
        var celda2 = document.createElement("td");  
        fila.appendChild(celda2);  
        var precio = document.createTextNode(respuesta[i].precio);  
        celda2.appendChild(precio);  
    }  
}
```