

LENGUAJES INTERPRETADOS EN EL CLIENTE

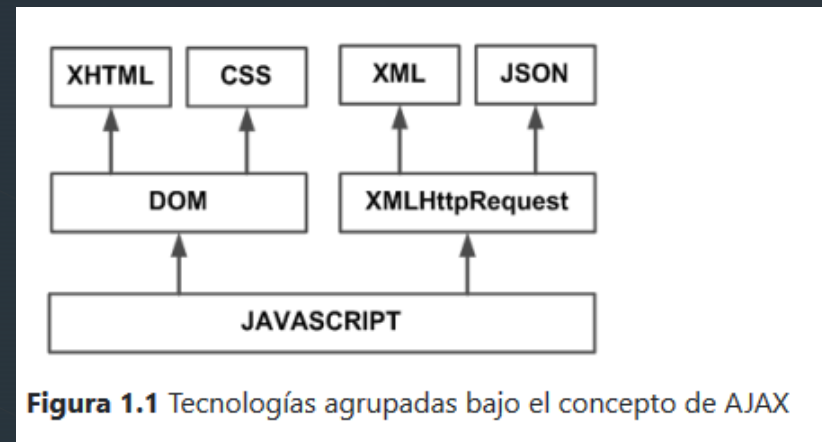


INTRODUCCIÓN A AJAX

- AJAX (Asynchronous JavaScript And XML). Pronunciación (Ajax ó eyax) No es un lenguaje de programación y tampoco es una tecnología en sí mismo. Necesitamos un servidor.
- Con AJAX se consigue una navegación ágil, rápida y dinámica.
- AJAX permite comunicarse con sistemas remotos y/o refrescar partes de una página si necesidad de recargar la página.

Las tecnologías que forman AJAX son:


- XHTML y CSS, para crear una presentación basada en estándares.
- DOM, para la interacción y manipulación dinámica de la presentación.
- XML, XSLT y JSON, para el intercambio y la manipulación de información.
- XMLHttpRequest, para el intercambio asíncrono de información.
- JavaScript, para unir todas las demás tecnologías.




Conceptos Clave

GET vs. POST

- Los dos métodos HTTP más comunes para enviar una petición a un servidor son GET y POST.
- El método GET debe ser utilizado para obtener datos del servidor, pero no modificando. Por otro lado, las solicitudes GET pueden ser almacenadas en la cache del navegador, pudiendo conducir a un comportamiento impredecible si no se lo espera. Generalmente, la información enviada al servidor, es enviada en una cadena de datos (en inglés *query string*).

- 
- El método POST debe ser utilizado para operaciones en donde se está incorporando información al servidor. Por otro lado, este tipo de método no se guarda en la cache del navegador. Además, una cadena de datos puede ser parte de la URL, pero la información tiende a ser enviada de forma separada

- 
- El punto débil de AJAX es que hay que invertir tiempo para escribir código para cada uno de los navegadores que utilizaremos, pero con jQuery se nos permite utilizar AJAX con la siguiente línea de código:
 - **`$.ajax({name:value, name:value, ... })`**

MÉTODOS jQuery para realizar peticiones AJAX al servidor

Method	Description
<u>\$.ajax()</u>	Performs an async AJAX request
<u>\$.ajaxPrefilter()</u>	Handle custom Ajax options or modify existing options before each request is sent and before they are processed by \$.ajax()
<u>\$.ajaxSetup()</u>	Sets the default values for future AJAX requests
<u>\$.ajaxTransport()</u>	Creates an object that handles the actual transmission of Ajax data
<u>\$.get()</u>	Loads data from a server using an AJAX HTTP GET request
<u>\$.getJSON()</u>	Loads JSON-encoded data from a server using a HTTP GET request
<u>\$.parseJSON()</u>	Deprecated in version 3.0, use <u>JSON.parse()</u> instead. Takes a well-formed JSON string and returns the resulting JavaScript value
<u>\$.getScript()</u>	Loads (and executes) a JavaScript from a server using an AJAX HTTP GET request
<u>\$.param()</u>	Creates a serialized representation of an array or object (can be used as URL query string for AJAX requests)
<u>\$.post()</u>	Loads data from a server using an AJAX HTTP POST request
<u>ajaxComplete()</u>	Specifies a function to run when the AJAX request completes
<u>ajaxError()</u>	Specifies a function to run when the AJAX request completes with an error
<u>ajaxSend()</u>	Specifies a function to run before the AJAX request is sent
<u>ajaxStart()</u>	Specifies a function to run when the first AJAX request begins
<u>ajaxStop()</u>	Specifies a function to run when all AJAX requests have completed
<u>ajaxSuccess()</u>	Specifies a function to run when an AJAX request completes successfully
<u>load()</u>	Loads data from a server and puts the returned data into the selected element
<u>serialize()</u>	Encodes a set of form elements as a string for submission
<u>serializeArray()</u>	Encodes a set of form elements as an array of names and values

Ver ejemplos en: https://www.w3schools.com/jquery/jquery_ref_ajax.asp

- Todos los plugins de JQuery necesitan la librería de JQuery para funcionar. La librería es un archivo .js que se puede descargar desde el sitio oficial:

<https://jquery.com/> colocar en una carpeta js y luego vincular con una etiqueta `<script>`

- EJEMPLO:

Colocamos dentro del body:

```
<script src="js/jquery-3.5.1.min.js"></script>
```

Si descargamos el archivo y lo vinculamos de esa manera, debemos subir el archivo a nuestro hosting.

Otra forma de vincular la librería es utilizando el servidor de Google. De esta manera no necesitamos descargarla ni subirla a nuestro servidor. En ese caso, el código es el mismo pero con ruta absoluta:

```
<script  
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jq  
uery.min.js"></script>
```


SINTAXIS DEL MÉTODO \$.ajax()

```
$.ajax(url, { objeto-configurable } ); // o $.ajax( { objeto-configurable } );
```

El objeto configurable contendrá uno o varios de los parámetros siguientes:

- type : tipo de la petición, GET o POST (GET por defecto).
- url : dirección a la que se envía la petición.
- data : datos a enviar al servidor.
- dataType : tipo de datos que se espera obtener del servidor (si no se especifica, jQuery intenta averiguar de qué tipo se trata).
- success : función que se ejecuta cuando se obtiene una respuesta con éxito.
- error : función que se llama si la petición no tiene éxito.

EJEMPLO CON EL MÉTODO \$.ajax()

- La función es un bloque de código JavaScript, en esta se colocan instrucciones para el plugin.
- Se escribe dentro de las etiquetas **<script></script>**. Puede ir debajo de la estructura HTML del plugin, dentro del body o al final del documento.

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js">
</script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $.ajax({url: "demo_test.txt", success: function(result){
      $("#div1").html(result);
    }});
  });
});
</script>
</head>
<body>

<div id="div1"><h2>Let jQuery AJAX Change This Text</h2></div>

<button>Get External Content</button>

</body>
</html>
```

EJEMPLO CON EL MÉTODO \$.get()

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js">
</script>
<script>
$(document).ready(function(){
    $("button").click(function(){
        $.get("demo_test.asp", function(data, status){
            alert("Data: " + data + "\nStatus: " + status);
        });
    });
});
</script>
</head>
<body>

<button>Send an HTTP GET request to a page and get the result back</button>

</body>
</html>
```

XMLHttpRequest object.

- Todos los navegadores modernos admiten el objeto XMLHttpRequest, ya lo tienen incorporado.
- El objeto XMLHttpRequest se puede utilizar para intercambiar datos con un servidor web. Esto significa que es posible actualizar partes de una página web, sin recargar toda la página.

Sintaxis para crear un objeto XMLHttpRequest

- `variable = new XMLHttpRequest();`

Ejemplo

```
<!DOCTYPE html>
<html>
<body>

<h2> XMLHttpRequest </h2>


<p id="demo">AJAX cambiara este texto</p>

<button type="button" onclick="loadDoc()">Cambiar contenido</button>

<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "ajax_info.txt", true);
  xhttp.send();
}
</script>

</body>
</html>
```

La propiedad `XMLHttpRequest.onreadystatechange` contiene el manejador del evento que es invocado cuando se dispara el evento `readystatechange`, lo cual sucede cada vez que cambia el valor de la propiedad `readyState` de `XMLHttpRequest`

- 
- Por razones de seguridad, los navegadores modernos no permiten el acceso entre dominios.
 - Esto significa que tanto la página web como el archivo XML que intenta cargar, deben estar ubicados en el mismo servidor

- Las versiones anteriores de Internet Explorer (5/6) usan un objeto ActiveX en lugar del objeto XMLHttpRequest:
- `variable = new ActiveXObject("Microsoft.XMLHTTP");`

```
<script>
function loadDoc() {
    var xhttp;
    if (window.XMLHttpRequest) {
        // código para navegadores modernos
        xhttp = new XMLHttpRequest();
    } else {
        // código para IE6, IE5
        xhttp = new ActiveXObject("Microsoft.XMLHTTP");
    }
    xhttp.onreadystatechange = function() {
        if (this.readyState == 4 && this.status == 200) {
            document.getElementById("demo").innerHTML = this.responseText;
        }
    };
    xhttp.open("GET", "ajax_info.txt", true);
    xhttp.send();
}
</script>
```

Tutorial:

<https://www.youtube.com/watch?v=bUwadY1yqi0&list=PLK7sa90aSL74eFKxktrvzMeXA1zilxaT>