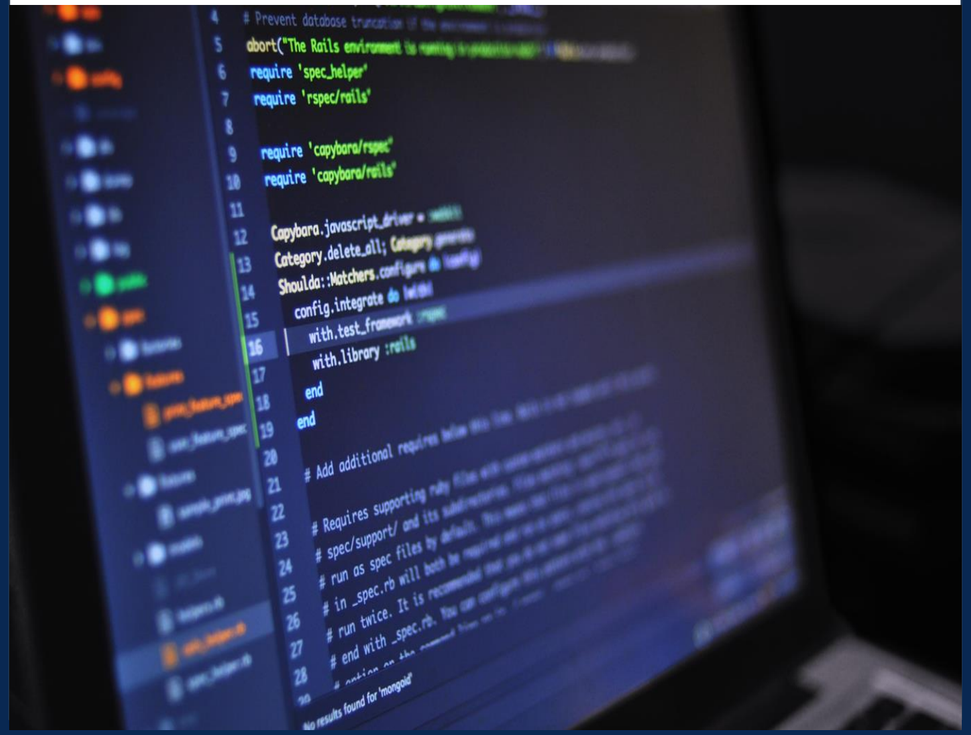
# Bloque TypeScript



# Tema 8: AJAX

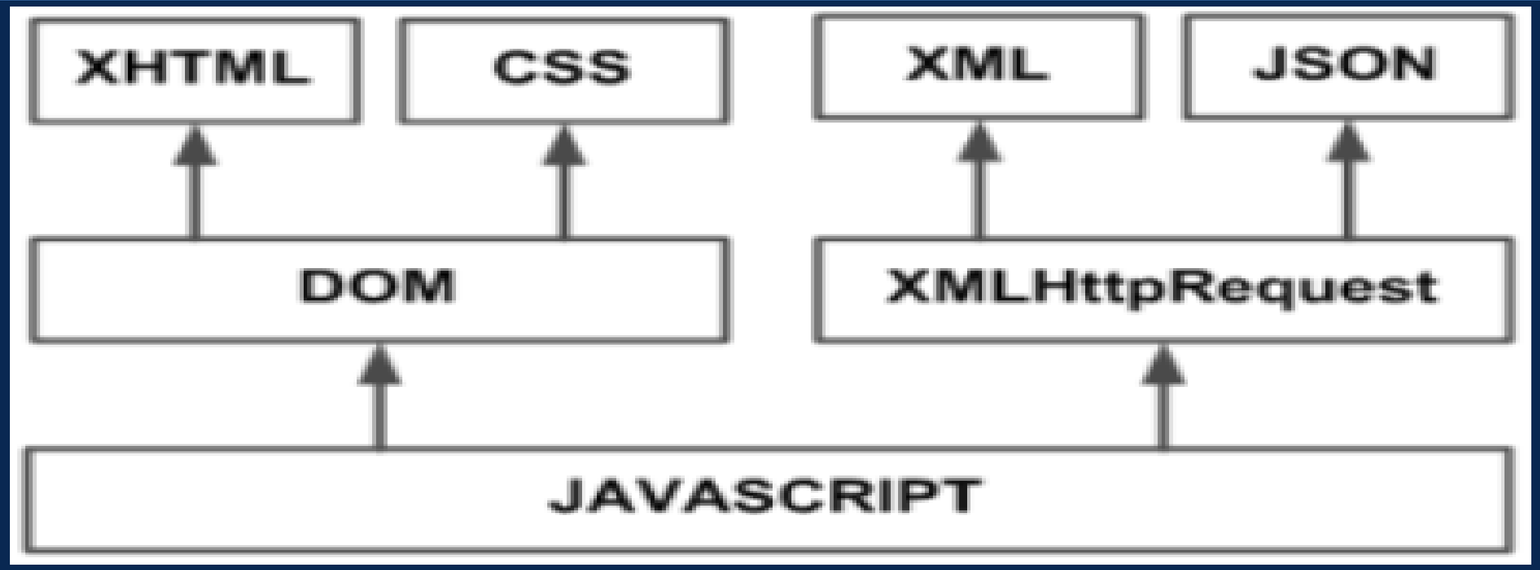
un evento es lo que se produce cuando el navegador detecta determinados eventos; por ejemplo, cuando se pulsa en un enlace, se activa un botón o se carga una página.

# I N T R O D U C C I Ó N

AJAX: ***A**synchronous **J**avaScript **A**nd **X**ML*

Representa el soporte de los navegadores para acceder, de modo asíncrono, a datos en el servidor utilizando JavaScript.

Actualmente también se entiende por AJAX las bibliotecas de componentes gráficos programados con HTML Dinámico.



un evento es lo que se produce cuando el navegador detecta determinados eventos; por ejemplo, cuando se pulsa en un enlace, se activa un botón o se carga una página.

# A J A X


- Las peticiones AJAX utilizan un objeto **XMLHttpRequest**.
- Permite realizar con *JavaScript* una petición HTTP al servidor y esperar a que se reciban los datos.
- La clave de la programación con AJAX es la **petición asíncrona de datos al servidor**:
  - Se solicitan datos al servidor, pero el navegador no queda bloqueado a la espera.
  - Cuando llegan los datos, una función *JavaScript* se encarga de procesarlos.

un evento es lo que se produce cuando el navegador detecta determinados eventos; por ejemplo, cuando se pulsa en un enlace, se activa un botón o se carga una página.

# A J A X


La función que recibe los datos del servidor se conoce como “**callback**”.

La función callback es llamada cada vez que cambia el estado de la petición HTTP:

Cuando el estado de la petición es completo (4) y el código de respuesta de HTTP es correcto (200) se pueden obtener los datos de la petición en texto normal o estructurado en un árbol DOM.

un evento es lo que se produce cuando el navegador detecta determinados eventos; por ejemplo, cuando se pulsa en un enlace, se activa un botón o se carga una página.

# X M L H T T P R E S Q U E S T


API que se encuentra implementado en el navegador y que proporciona los métodos y propiedades necesarios para la comunicación con el servidor mediante HTTP.

Este objeto nos permitirá hacer peticiones asíncronas y se encargará de avisarnos cuando se reciba su respuesta.

un evento es lo que se produce cuando el navegador detecta determinados eventos; por ejemplo, cuando se pulsa en un enlace, se activa un botón o se carga una página.

# X M L H T T P R E S Q U E S T

## ATRIBUTOS

| ATRIBUTO | DESCRIPCIÓN |
|---|---|
| readyState | Devuelve el estado del objeto como sigue:<br><br>0 = sin inicializar<br>1 = abierto<br>2 = cabeceras recibidas<br>3 = cargando<br>4 = completado. |
| responseBody | Devuelve la respuesta como un array de bytes. |
| responseText | Devuelve la respuesta como una cadena. |
| responseXML | Devuelve la respuesta como XML.<br><br>Esta propiedad devuelve un objeto documento XML, que puede ser examinado usando las propiedades y métodos del árbol del Document  Object Model. |
| status | Devuelve el estado como un número (p. ej. 404 para "Not Found" y 200 para "OK"). |
| statusText | Devuelve el estado como una cadena (p. ej. "Not Found" o "OK"). |

un evento es lo que se produce cuando el navegador detecta determinados eventos; por ejemplo, cuando se pulsa en un enlace, se activa un botón o se carga una página.

# X M L H T T P R E S Q U E S T


## MÉTODOS

| MÉTODO | DESCRIPCIÓN |
|---|---|
| abort() | Cancela la petición en curso. |
| getAllResponseHeaders() | Devuelve el conjunto de cabeceras HTTP como una cadena. |
| getResponseHeader(nombreCabecera) | Devuelve el valor de la cabecera HTTP especificada. |
| open (método, URL, [asíncrono[nombreUsuario [clave]]] ) | Especifica el método, URL y otros atributos opcionales de una petición.<br><br>El parámetro de método puede tomar los valores "GET", "POST", o "PUT" ("GET" y "POST" son dos formas para solicitar datos, con "GET" los parámetros de la petición se codifican en la URL y con "POST" en las cabeceras de HTTP).<br><br>El parámetro URL puede ser una URL relativa o completa.<br><br>El parámetro asíncrono especifica si la petición será gestionada asíncronamente o no. Un valor true indica que el proceso del script continúa después del método send(), sin esperar a la respuesta, y false indica que el script se detiene hasta que se complete la operación, tras lo cual se reanuda la ejecución.<br><br>En el caso asíncrono se especifican manejadores de eventos, que se ejecutan ante cada cambio de estado y permiten tratar los resultados de la consulta una vez que se reciben, o bien gestionar eventuales errores. |
| send([datos]) | Envía la petición. |
| setRequestHeader( etiqueta, valor ) | Añade un par etiqueta/valor a la cabecera HTTP a enviar. |

un evento es lo que se produce cuando el navegador detecta determinados eventos; por ejemplo, cuando se pulsa en un enlace, se activa un botón o se carga una página.

# X M L H T T P R E S Q U E S T

## EVENTOS

| PROPIEDAD | DESCRIPCIÓN |
|---|---|
| onreadystatechange | Evento que se dispara con cada cambio de estado. |
| onabort | Evento que se dispara al abortar la operación. |
| onload | Evento que se dispara al completar la carga. |
| onloadstart | Evento que se dispara al iniciar la carga. |
| onprogress | Evento que se dispara periódicamente con información de estado. |

un evento es lo que se produce cuando el navegador detecta determinados eventos; por ejemplo, cuando se pulsa en un enlace, se activa un botón o se carga una página.

# INICIALIZAR XMLHTTPREQUEST


```
function obtainXMLHttpRequest()
{
    let httpRequest;
    if (window.XMLHttpRequest){
        //El explorador implementa la interfaz de forma nativa
        httpRequest = new XMLHttpRequest();
    }
    else if (window.ActiveXObject){
        //El explorador permite crear objetos ActiveX
        try {
            httpRequest = new ActiveXObject("MSXML2.XMLHTTP");
        } catch (e) {
            try {
                httpRequest = new ActiveXObject("Microsoft.XMLHTTP");
            } catch (e) {}
        }
    }
    // Si no se puede crear, devolvemos false. En caso contrario, devolvemos el  objeto
    if (!httpRequest){
        return false;
    }
    else{
        return httpRequest;
    }
}
```

un evento es lo que se produce cuando el navegador detecta determinados eventos; por ejemplo, cuando se pulsa en un enlace, se activa un botón o se carga una página.

# EVENTO ONREADYSTATECHANGE


3 métodos del objeto:

1. **open(Metodo,URL,true)**. Este método recibe qué método de comunicación utiliza en la petición (GET o POST) y que URL se aplica.
2. **send([datos])**. Este método ha de hacerse posteriormente a un open, nunca antes. Envía la información a la URL especificada en open.
3. **setRequestHeader** que indicará el formato de las cabeceras enviadas.

También debemos explicar el atributo readyState: este atributo puede poseer los siguientes valores:

**0** al inicializarse el objeto.
**1** al abrirse una conexión (al usar el método open).
**2** al hacer una petición (uso de send).
**3** mientras se está recibiendo información de la petición.
**4** cuando la petición se ha completado.

un evento es lo que se produce cuando el navegador detecta determinados eventos; por ejemplo, cuando se pulsa en un enlace, se activa un botón o se carga una página.

# AJAX -PETICIÓN HTTP

TS

```
// Abrimos la conexión
httpRequest.open("POST","ajax.php",true);
// Indicamos como seran las cabeceras
httpRequest.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
// Definimos el comportamiento de onreadystatechange
httpRequest.onreadystatechange=function(){
    if (httpRequest.readyState==1) {
            document.getElementById('capa').innerHTML="CARGANDO...";
    }
    // Si se ha completado
    if (httpRequest.readyState==4) {
        // Si es correcto el status
        if (httpRequest.status==200){
        // de httpRequest.responseText obtenemos la cadena con la respuesta
                document.getElementById('capa').innerHTML=httpRequest .responseText;
        }
    }
}
// Enviamos la acción
httpRequest.send("accion="+accion);
```