# Comunicación asíncrona con AJAX

**AJAX** son las siglas de *Asynchronous JavaScript And XML* (JavaScript asíncrono y XML). Hace referencia a una técnica que permite realizar peticiones desde la parte cliente de una web empleando JavaScript, recibir la información de forma asíncrona (es decir, sin quedarse esperando a recibir la respuesta), recoger los datos recibidos y actualizar con ellos una parte de la página, sin necesidad de recargarla en su totalidad.

Para ello, emitiremos peticiones a un servidor web desde JavaScript, y dejaremos definido un método que se ejecutará cuando se reciba la respuesta. Dentro de ese método, típicamente se recogen los datos recibidos y se muestran en una parte de la página. Veremos a continuación cómo definir cada uno de estos pasos.

## 1. Envío de la petición desde JavaScript

Para enviar una petición AJAX desde JavaScript haremos uso de un objeto especial ya predefinido en el lenguaje llamado `XMLHttpRequest`. Debemos inicializarlo, y definir sobre él:

- El recurso del servidor que queremos solicitar, indicando la URL
- Qué función queremos que se ejecute cuando se reciba la respuesta, a través del evento *onreadystatechange* del propio objeto
- Enviar la petición

Aquí vemos un ejemplo donde llamamos a una página llamada *prueba.php* (alojada en la raíz del servidor donde se encuentra el propio archivo JavaScript). El contenido que se reciba se volcará en el interior del elemento con *id* "contenido".

```
const xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
        let elemento = document.getElementById("contenido");
        elemento.innerHTML = this.responseText;
    }
}
xhttp.open("GET", "prueba.php", true);
xhttp.send();
```

El método `open` es el que se encarga de definir a qué recurso queremos acceder (*prueba.php*), y su tercer parámetro a *true* indica que queremos llamarlo de forma asíncrona para no detener el código esperando respuesta. Cuando se invoca el método `send` se envía la petición, y cuando se reciba se ejecutará la función del evento *onreadystatechange*. Dentro de este evento, comprobamos si la petición ha finalizado (`readyState == 4`) y además si lo ha hecho correctamente (`this.status == 200`). Si es así, recogemos la respuesta y la mostramos en el elemento seleccionado.

> **Ejercicio 1:**
>
> Define una carpeta llamada **listacompra** con una página llamada **lista_compra.html** que tenga algún encabezado *h1* que diga "Lista de la compra", y un espacio para cargar una lista no ordenada de elementos. Cuando pulsemos un botón "Cargar lista" llamaremos por AJAX a una página **listado.php**, de la misma carpeta, que devolverá una lista de la compra leída de un fichero de texto, y transformada a lista no ordenada en HTML. Al recibirla, se mostrará la lista en el lugar habilitado de la página HTML.

# 2. AJAX y JSON

**JSON** son las siglas de *JavaScript Object Notation*, una sintaxis propia de Javascript para poder representar objetos como cadenas de texto, y poder así serializar y enviar información de objetos a través de flujos de datos (archivos de texto, comunicaciones cliente-servidor, etc).

Un objeto JavaScript se define mediante una serie de propiedades y valores. Por ejemplo, los datos de una persona (como nombre y edad) podríamos almacenarlos así:

```
let persona = {
    nombre: "Nacho",
    edad: 39
};
```

Este mismo objeto, convertido a JSON, formaría una cadena de texto con este contenido:

```
{"nombre":"Nacho","edad":39}
```

Del mismo modo, si tenemos una colección (vector) de objetos como ésta:

```
let personas = [
    { nombre: "Nacho", edad: 39},
    { nombre: "Mario", edad: 4},
    { nombre: "Laura", edad: 2},
    { nombre: "Nora", edad: 10}
];
```

Transformada a JSON sigue la misma sintaxis, pero entre corchetes:

```
[{"nombre":"Nacho","edad":39}, {"nombre":"Mario","edad":4},
 {"nombre":"Laura","edad":2}, {"nombre":"Nora","edad":10}]
```

JavaScript ofrece un par de métodos útiles para convertir datos a formato JSON y viceversa. Estos métodos son `JSON.stringify` (para convertir un objeto o array JavaScript a JSON) y `JSON.parse` (para el proceso inverso, es decir, convertir una cadena JSON en un objeto JavaScript). Aquí vemos un ejemplo de cada uno:

```
let personas = [
    { nombre: "Nacho", edad: 39},
    { nombre: "Mario", edad: 4},
    { nombre: "Laura", edad: 2},
    { nombre: "Nora", edad: 10}
];

// Convertir array a JSON
let personasJSON = JSON.stringify(personas);
console.log(personasJSON);

// Convertir JSON a array
let personas2 = JSON.parse(personasJSON);
console.log(personas2);
```

JSON puede resultar una herramienta realmente útil combinada con AJAX, ya que va a permitir que el servidor envíe información al cliente en formato JSON (sin un formato específico) y que ese mismo cliente adapte la información y le dé el formato adecuado a sus necesidades. De este modo, un servidor puede proporcionar la misma información a distintos clientes (aplicación web, de escritorio, móvil) y que cada una la adapte visualmente a su entorno particular.

El siguiente fragmento de código define un array de datos en PHP y lo envía en formato JSON, usando la función `json_encode` de PHP:

```
$datos = array(
    "nombre" => "Nacho",
    "telefono" => "611223344",
    "edad" => 43
);
echo json_encode($datos);
```

Suponiendo que el código anterior esté en una página llamada *datos.php*, este otro código invoca a esa página desde JavaScript usando AJAX, recoge el resultado JSON, lo convierte a JavaScript y muestra el resultado en una sección identificada como *contenido*:

```
const xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
        let elemento = document.getElementById("contenido");
        let datos = JSON.parse(this.responseText);
        elemento.innerHTML = "<ul>";
        elemento.innerHTML += "<li>Nombre: " + datos.nombre + "</li>";
        elemento.innerHTML += "<li>Teléfono: " + datos.telefono + "</li>";
        elemento.innerHTML += "<li>Edad: " + datos.edad + "</li>";
        elemento.innerHTML += "</ul>";
    }
}
xhttp.open("GET", "datos.php", true);
xhttp.send();
```

**Ejercicio 2:**

Haz una copia del ejercicio anterior en otra carpeta llamada **listacompraJSON** y añade estos cambios:

- El fichero **listado.php** leerá los datos de la compra de un fichero y los enviará al cliente en formato JSON. En este caso enviaremos un array de objetos JavaScript, donde cada uno tendrá un único campo con el título del artículo
- El fichero **lista_compra.html** pedirá el listado por AJAX, como antes, pero ahora lo que recibirá será un array de datos en formato texto, que deberá convertir a JSON y mostrar en el lugar correspondiente de la página.

## 2.1. Envío de formularios con AJAX

También es posible utilizar AJAX para enviar formularios de forma asíncrona desde el cliente, recoger la respuesta y mostrarla en un lugar de la página sin tener que recargarla toda. Supongamos que tenemos un formulario identificado con `name="form1"`, y queremos enviar sus datos por AJAX a `procesar_formulario.php`. Dicho formulario puede tener establecidos sus mecanismos de validación en el cliente, así que lo que haremos será alterar el comportamiento del evento *submit* para deshacer el comportamiento por defecto y enviar el formulario de otro modo. Para ello, crearemos un objeto de tipo `FormData` de JavaScript, con los datos del formulario, y los enviaremos por AJAX a la página destino, recogiendo la respuesta:

```
let formulario = document.form1;
formulario.addEventListener("submit", function(evento)
{
    evento.preventDefault();
    let datos = new FormData(formulario);
    const xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function() {
        if (this.readyState == 4 && this.status == 200) {
            // Recoger respuesta del servidor y procesarla
            let datosRecibidos = JSON.parse(this.responseText);
            // Mostrar los datos en algún componente
            ...
        }
    }
    xhttp.open("POST", "procesar_formulario.php", true);
    xhttp.send(datos);
});
```

**Ejercicio 3:**

Haz una copia del ejercicio anterior en otra carpeta llamada **listacompraJSONForm** y haz que el fichero **lista_compra.html** tenga ahora, además de lo anterior, un formulario para poder dar de alta nuevos títulos de artículos en la lista de la compra. Cuando enviemos el formulario, desde AJAX crearemos un `FormData` con sus datos y lo enviaremos a otra página **nuevo.php**, que lo añadirá al fichero con la lista de la compra, y redirigirá a **listado.php** para que devuelva la lista actualizada en formato JSON. Al recibir la lista JSON (tras enviar el formulario) se actualizará en *lista_compra.html*. De este modo, la lista se actualizará tanto al pulsar el botón de "Cargar lista" como al enviar el formulario.