AJAX

Asynchronous JavaScript And XML

Valeriano Moreno

2023

INTRO

AJAX es sinónimo de usar JavaScript desde el navegador, en el rol de cliente, para comunicarnos con un servidor del que obtener o adjuntar información

INTRO - ARQUITECTURA

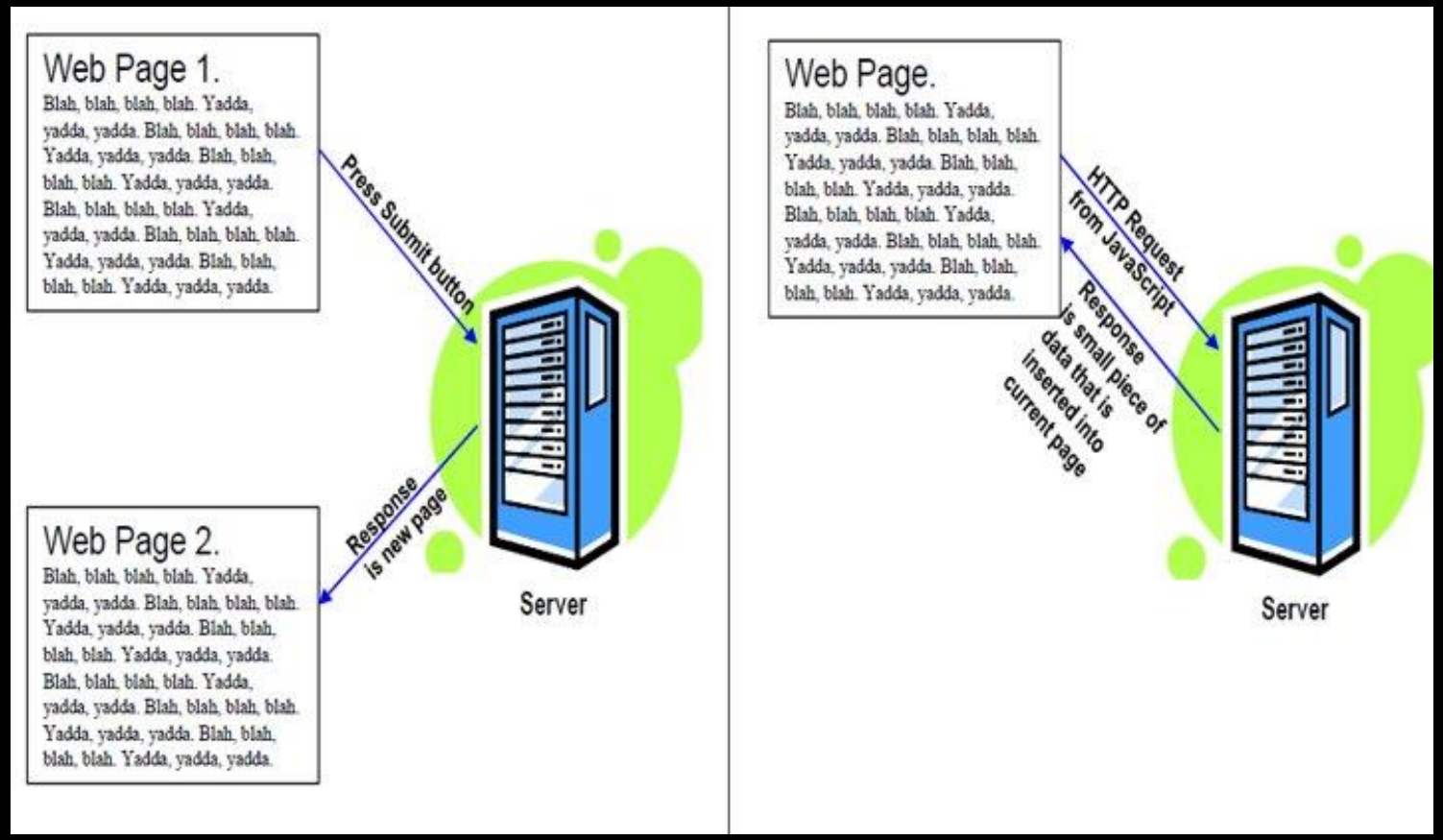
La tecnología tiene pues cabida en el modelo CLIENTE-SERVIDOR de 2 o 3 capas

El cliente será siempre un navegador, bien a veces puro o embebido en una aplicación híbrida

INTRO - TECNOLOGÍAS

Su uso se ha intensificado en las aplicaciones SPA, pudiendo aparecer no sólo en webs estándar, sino muchas veces internamente, envuelto en frameworks como Angular, React, Cordova o Ionic.

INTRO - EVOLUCIÓN WEB



Modelo CLÁSICO

Modelo AJAX

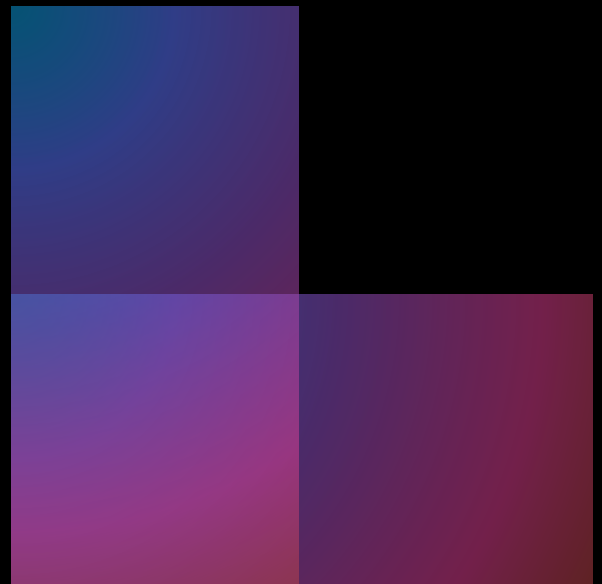
INTRO - EJEMPLO

CICLO GENERAL AJAX

JS llama al Servidor

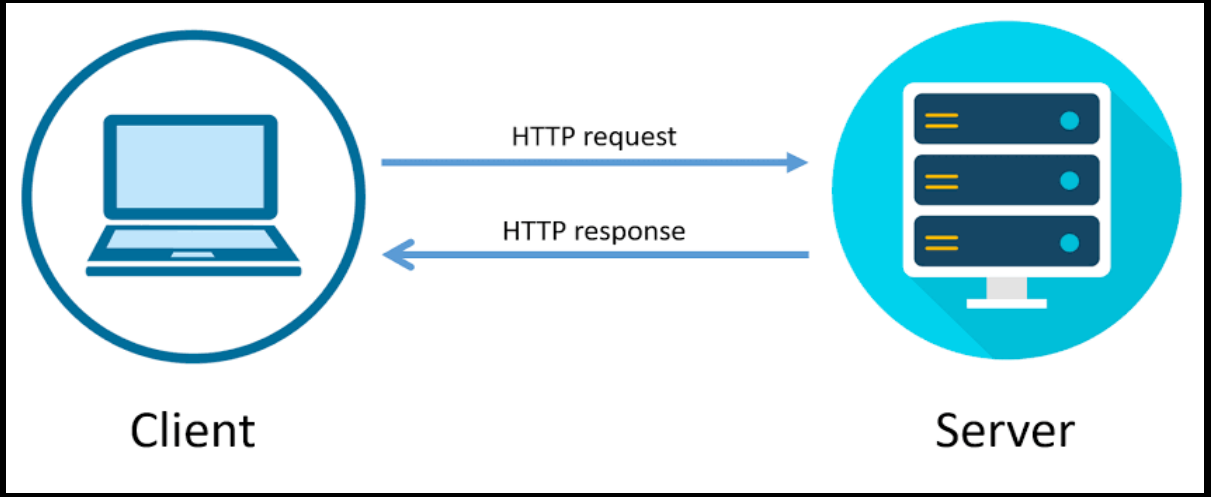
El servidor recibe la petición, la procesa y genera la respuesta

El cliente recibe la respuesta, actualiza la IU – si procede - controlando posibles errores



HTTP AJAX

AJAX se ejecuta sobre HTTP, por lo que la conocer el protocolo es indispensable



INFORMACIÓN

Generalmente, la información intercambiada entre el cliente y servidor es codificada en texto en formato XML o JSON

XML VS JSON

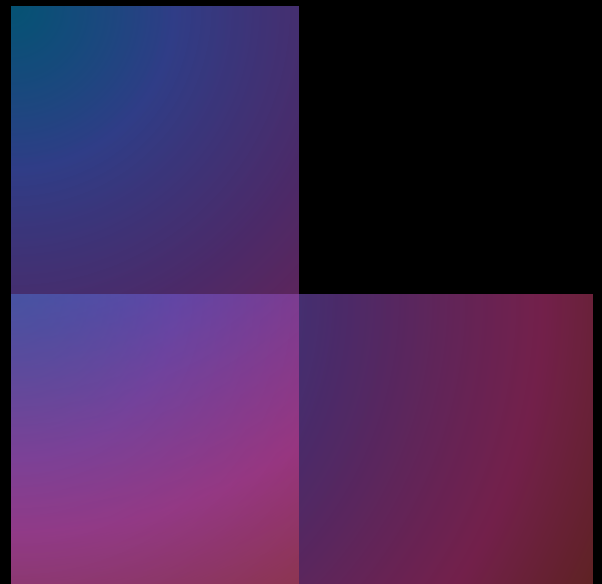
XML y JSON son formatos alternativos de representación textual de información jerarquizada

La principal diferencia es que JSON es un poco más económico

XML VS JSON

```
<menu id="file" value="File"> <popup>
<menuitem value="New" onclick="CreateNewDoc()"
/> <menuitem value="Open" onclick="OpenDoc()" />
<menuitem value="Close" onclick="CloseDoc()" />
</popup> </menu>
```

```
{"menu": { "id": "file", "value": "File", "popup": {
"menuitem":    [ {"value": "New",
"onclick": "CreateNewDoc()"}, {"value": "Open",
"onclick": "OpenDoc()"}, {"value": "Close", "onclick":
"CloseDoc()"} ] } } }
```



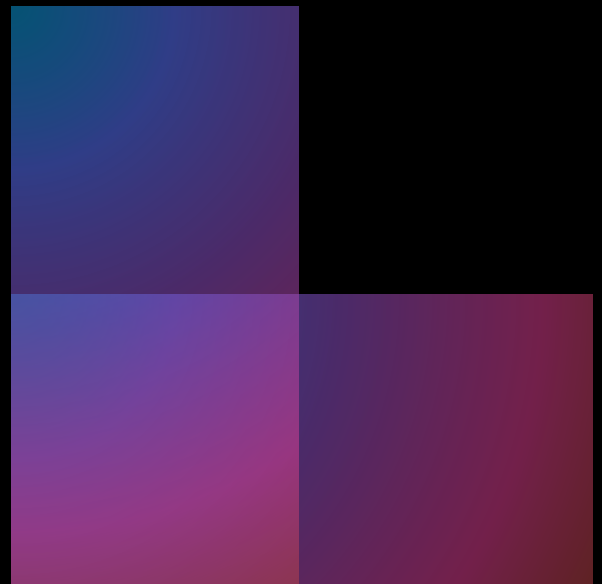
(DE) SERIALIZACIÓN

Para trabajar con el formato JSON existe un objeto predefinido en el navegador, que incorpora los métodos para convertir un objeto en texto y viceversa

(DE) SERIALIZACIÓN

JSON.parse() Deserializar

JSON.stringify() Serializar



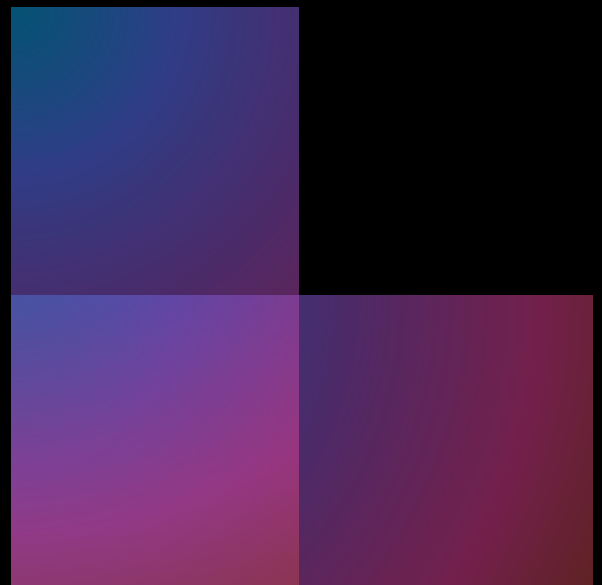
APIS

Como hemos comentado, en esencia, todo se traduce al intercambio petición-respuesta de HTTP entre el cliente y el servidor

Para tal fin, hay diversas APIS o métodos

APIS

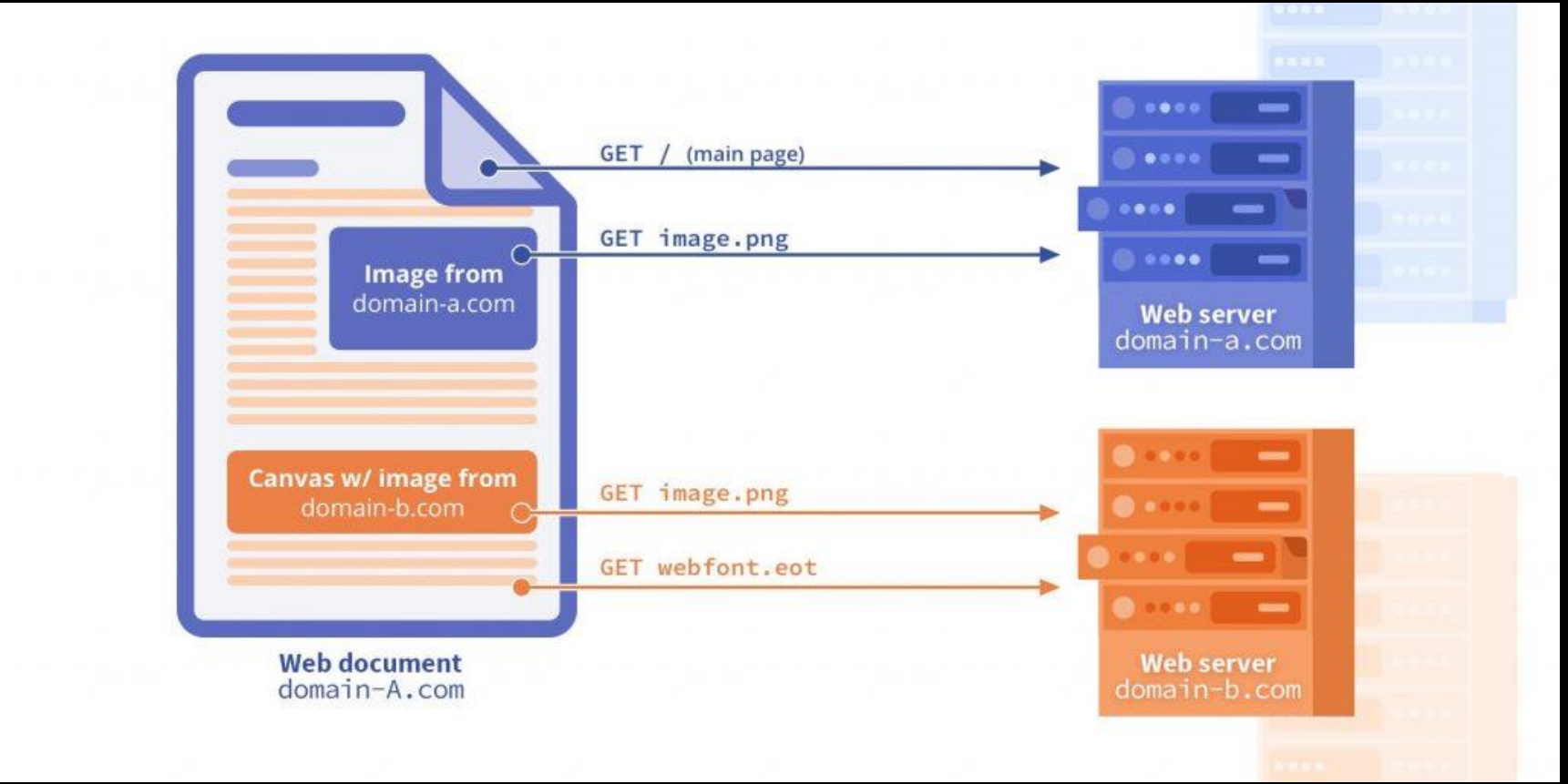
- XMLHttpRequest -estándar-
- Fetch (await) -estándar-
- Axios -librería-
- Observables (TScriptAngular) -librería-



CORS

Al acceso a orígenes cruzados o CORS es un escenario a menudo conflictivo en el que un script, descargado de un dominioA, intenta acceder a un servidor de un dominioB

CORS

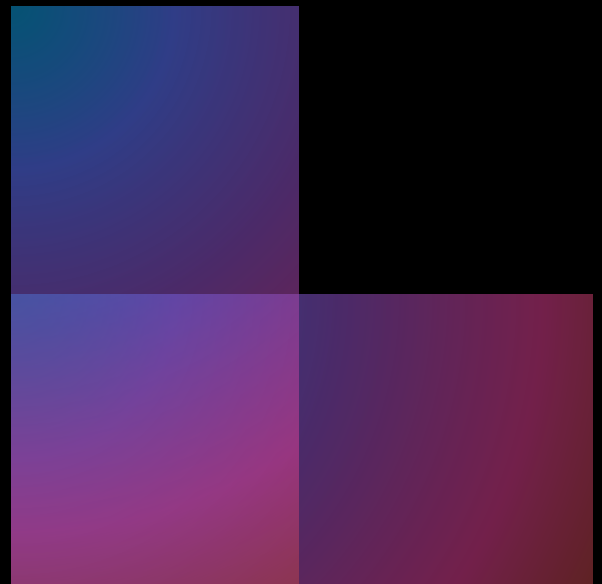

GET / (main page)
GET image.png
Image from
domain-a.com
Canvas w/ image from
domain-b.com
GET image.png
GET webfont.eot
Web document
domain-A.com
Web server
domain-a.com
Web server
domain-b.com

CORS

El problema se manifiesta en el lado del cliente, pero tiene su solución definitiva en el lado del servidor, por lo que confunde en su apariencia y complejidad a muchos programadores aún expertos

SOLUCIONES CORS

- Plugins
- Servidor Intermedio
- Configuración servidor
- JSONP



Hilos / Threads

Aunque la tecnología AJAX genera un hilo independiente del principal donde se gestiona la comunicación, podemos mejorar el rendimiento de nuestras aplicaciones envolviendo AJAX en un WebWorker aparte del script principal

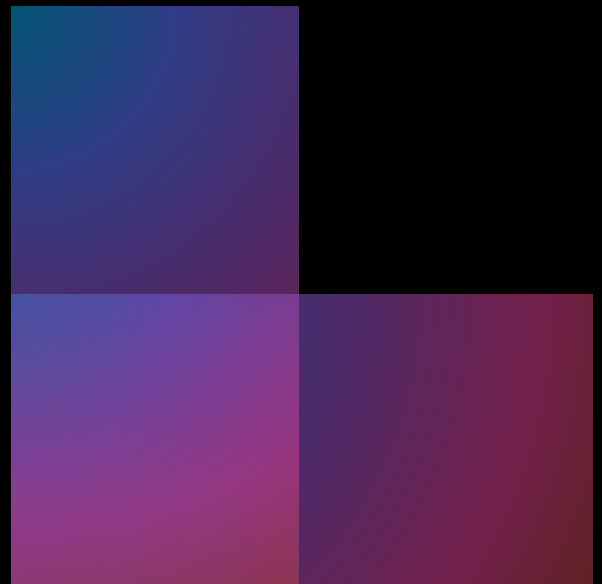
WebWorker

Un WebWorker es un mecanismo que ofrece el API de JS para ejecutar procesos en hilos independientes del principal

WebWorker

Para comunicar el hilo principal con el del WebWorker, existen EVENTOS predefinidos

Hay que tener en cuenta que desde el WebWorker no tengo acceso al DOM, ni a algunas propiedades del objeto window



Richardson Maturity Model

Criterio de madurez de APIS web que van desde el 0 hasta el 3

Los servicios REST se consideran un nivel de madurez 3 o superior

ENLACES DE INTERÉS

- MDN - AJAX
- MDN - XMLHttpRequest
- MDN – Fetch
- AXIOS
- MDN - Async Await

ENLACES DE INTERÉS

- Promesas
- Funciones Flecha
- MDN – CORS
- Can I USE
- JSONP
- WebWorker