
FETCH API

USO DE FETCH

- La API Fetch proporciona una interfaz JavaScript para acceder y manipular partes del canal HTTP, tales como peticiones y respuestas. También provee un método global fetch() (en-US) que proporciona una forma fácil y lógica de obtener recursos de forma asíncrona por la red.

Este tipo de funcionalidad se conseguía previamente haciendo uso de XMLHttpRequest.

Fetch proporciona una alternativa mejor que puede ser empleada fácilmente por otras tecnologías como Service Workers (en-US)

HíbridoFetch también aporta un único lugar lógico en el que definir otros conceptos relacionados con HTTP como CORS y extensiones para HTTP.

PETICIONES HTTP CON FETCH

- Fetch es el nombre de una nueva API para Javascript con la cuál podemos realizar peticiones HTTP asíncronas utilizando promesas y de forma que el código sea un poco más sencillo y menos verbose. La forma de realizar una petición es muy sencilla, básicamente se trata de llamar a fetch y pasarle por parámetro la URL de la petición a realizar:

```
// Realizamos la petición y guardamos la promesa
const request = fetch("/robots.txt");

// Si es resuelta, entonces ejecuta esta función...
request.then(function(response) { ... });
```

- El fetch() devolverá una que será aceptada cuando reciba una respuesta y sólo será rechazada si hay un fallo de red o si por alguna razón no se pudo completar la petición. El modo más habitual de manejar las promesas es utilizando .then(). Esto se suele reescribir de la siguiente forma, que queda mucho más simple:

```
fetch("/robots.txt")
  .then(function(response) {
    /** Código que procesa la respuesta **/
  });
```

Al método .then() se le pasa una función callback donde su parámetro response es el objeto de respuesta de la petición que hemos realizado. En su interior realizaremos la lógica que queramos hacer con la respuesta a nuestra petición. A la función fetch(url, options) se le pasa por parámetro la url de la petición y, de forma opcional, un objeto options con opciones de la petición HTTP.

```
// Opciones de la petición (valores por defecto)
const options = {
  method: "GET"
};

// Petición HTTP
fetch("/robots.txt", options)
  .then(response => response.text())
  .then(data => {
    /** Procesar los datos **/
  });
```

EN POCAS PALABRAS…

- En el corazón de Fetch estan las abstracciones de interfaz de cargas de HTTP Requests, Responses, Headers (en-US), y Body, junto a un método global fetch (en-US) para inicializar peticiones de recurso asíncronos. Porque los prinipales componentes de HTTP son abstraídos como objetos Javascript, es sencillo para otras APIs el hacer uso de dicha funcionalidad.
- Service Workers (en-US) es un ejemplo de una API que hace un fuerte uso de Fetch.
- Fetch toma la naturaleza asíncrona de dichas peticiones un paso adelante. La API esta completamente basada en Promise.

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Fetch API HTML Example</title>
  </head>
  <body>
    <div id="content"></div>
    <script>
      // Utilizamos la Fetch API para obtener el contenido de una página HTML
      fetch('https://www.ejemplo.com/pagina.html')
        .then(response => response.text()) // Convertimos la respuesta a texto
        .then(data => {
          const contentDiv = document.getElementById('content');
          contentDiv.innerHTML = data; // Insertamos el contenido en el elemento HTML
        })
        .catch(error => console.error(error)); // Manejamos cualquier error
    </script>
  </body>
</html>
```