Sesión 11 – Semana 4

Peticiones y Promesas

Carrera 43 A # 34 - 155. Almacentro. Torre Norte. Oficina 701
Medellín (Antioquia), Colombia

Contenido

1. Asincronía
2. Promesas
3. Peticiones HTTP
    1. Métodos HTTPs
4. AJAX
    1. Métodos de petición `Fetch`

Asincronía

Sincronía vs Asincronía

Sincronía

Cuando comenzamos a programar, normalmente realizamos tareas de forma **síncrona**, llevando a cabo **tareas secuenciales que se ejecutan una detrás de otra**, de modo que el orden o flujo del programa es sencillo y fácil de observar en el código:

```
primera_funcion();    // Tarea 1: Se ejecuta primero
segunda_funcion();    // Tarea 2: Se ejecuta cuando termina primera_funcion()
tercera_funcion();    // Tarea 3: Se ejecuta cuando termina segunda_funcion()
```

Asincronía

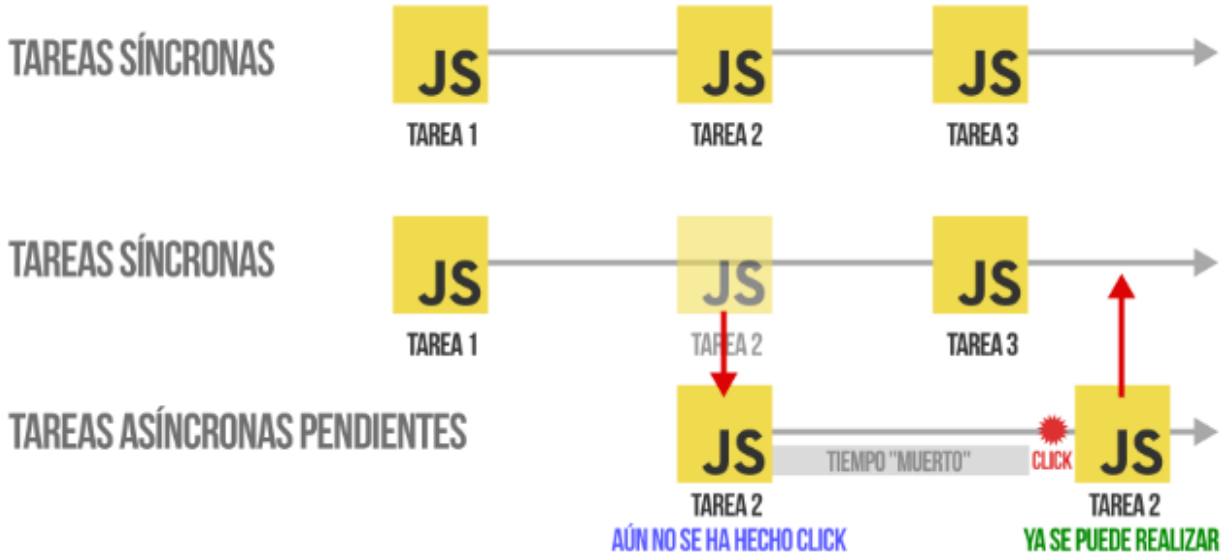
Sincronía vs Asincronía

Asincronía

Ejecución de tareas **que tienen que esperar a que ocurra un determinado suceso** que no depende de nosotros, y reaccionar realizando otra tarea sólo cuando dicho suceso ocurra.

Asincronía

Sincronía vs Asincronía



JavaScript mueve las tareas que dependen de otro factor (por ejemplo un click de un usuario) a una lista de tareas pendientes a las que irá «prestándole atención» a medida que lo necesite, pudiendo continuar y retomar el resto de tareas a continuación de la segunda.

Asincronía

¿Qué es?

Se refiere a la respuesta de una o múltiples tareas que sucederá en el futuro y estas tareas puede que terminen realizándose correctamente (o puede que no) y ciertas tareas pueden depender de otras, por lo que deben respetar un cierto orden. Además, es muy habitual que no sepamos previamente cuanto tiempo va a tardar en terminar una tarea, por lo que necesitamos un mecanismo para controlar todos estos factores: las **promesas**.

Promesas

¿Qué son?

Son un concepto para resolver el problema de asincronía de una forma mucho más elegante y práctica que, por ejemplo, utilizando funciones callbacks directamente. Como su propio nombre indica, una **promesa** es algo que, en principio pensamos que se cumplirá, pero en el futuro pueden ocurrir varias cosas:

1. La promesa **se cumple** (promesa resuelta)
2. La promesa **no se cumple** (promesa se rechaza)
3. La promesa se queda en un estado incierto **indefinidamente** (promesa pendiente)

MAKAIA

WWW.MAKAIA.ORG

Promesas

En JavaScript

Se representan a través de un `objeto`, y cada promesa estará en un estado concreto: pendiente, aceptada o rechazada. Además, cada promesa tiene los siguientes métodos:

Métodos	Descripción
`.then( FUNCTION resolve)`	Ejecuta la función callback **resolve** cuando la promesa se cumple.
`.catch( FUNCTION reject)`	Ejecuta la función callback **reject** cuando la promesa se rechaza.
`.then( FUNCTION resolve, FUNCTION reject)`	Método equivalente a las dos anteriores en el mismo `.then()`.
`.finally( FUNCTION end)`	Ejecuta la función callback **end** tanto si se cumple como si se rechaza.

Promesas

Consumir promesas

Se representan a través de un `objeto`, y cada promesa estará en un estado concreto: pendiente, aceptada o rechazada. Además, cada promesa tiene los siguientes métodos:

```
fetch("/robots.txt")
  .then(function(response) {
    /* Código a realizar cuando se cumpla la promesa */
  })
  .catch(function(error) {
    /* Código a realizar cuando se rechaza la promesa */
  });
```

```
fetch("/robots.txt")
  .then(response => response.text())
  .then(data => console.log(data))
  .finally(() => console.log("Terminado."))
  .catch(error => console.error(data));
```

Peticiones HTTPs

¿Qué es?

La acción por parte del navegador de solicitar a un servidor web un documento o archivo, ya sea un fichero .html, una imagen, una tipografía, un archivo .js, etc. Gracias a dicha petición, el navegador puede descargar ese archivo, almacenarlo en un caché temporal de archivos del navegador y, finalmente, mostrarlo en la página actual que lo ha solicitado.

Métodos HTTPs

1. GET: Solicita una representación de un recurso específico. Las peticiones que usan el método GET sólo deben recuperar datos.
2. POST: se utiliza para enviar una entidad a un recurso en específico, causando a menudo un cambio en el estado o efectos secundarios en el servidor.
3. PUT: reemplaza todas las representaciones actuales del recurso de destino con la carga útil de la petición.
4. DELETE: borra un recurso en específico.
5. PATCH: es utilizado para aplicar modificaciones parciales a un recurso.

AJAX (Asynchronous Javascript and XML)

¿Qué es?

Modalidad de realizar peticiones de forma transparente al usuario, descargando la información y pudiendo tratarla sin necesidad de mostrarla directamente en la página.

Peticiones HTTP con `Fetch`

¿Qué es?

Método para realizar peticiones HTTP asíncronas AJAX 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:

```
fetch('https://jsonplaceholder.typicode.com/todos/1')
```

Retorna una promesa que se resuelve con la respuesta del servidor

Peticiones HTTP con `Fetch`

¿Qué es?

Método para realizar peticiones HTTP asíncronas AJAX 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:

```
fetch('https://jsonplaceholder.typicode.com/todos/1')
```

Retorna una promesa que se resuelve con la respuesta del servidor

Fuente

1. https://lenguajejs.com/javascript/asincronia/que-es/
2. https://jonmircha.com/javascript-asincrono
3. https://developer.mozilla.org/es/docs/Learn/JavaScript/Asynchronous
4. https://lenguajejs.com/javascript/peticiones-http/ajax/
5. https://developer.mozilla.org/es/docs/Web/HTTP/Methods
6. https://lenguajejs.com/javascript/asincronia/promesas/

MAKAIA

WWW.MAKAIA.ORG

Info: comunicaciones@makaia.org

Corporación MAKAIA
Medellín, Colombia
Carrera 43A – 34-155. Almacentro
Torre Norte, Oficina 701
Teléfono: (+574) 448 03 74
Móvil: (+57) 320 761 01 76

@makaiaorg