Service Worker

Módulo 05

ServiceWorker

¿Qué es?

Es un proxy de red programable, es decir, una interfaz que me permite interceptar las peticiones http entrantes a mi página.

¿Para qué sirve?

En nuestra aplicación, nos servirá para interceptar las peticiones de todos los archivos y, una vez hecho eso, definir que los archivos se cargen desde la caché en vez de que se traigan desde la red, creando la funcionalidad offline propia de una AWP

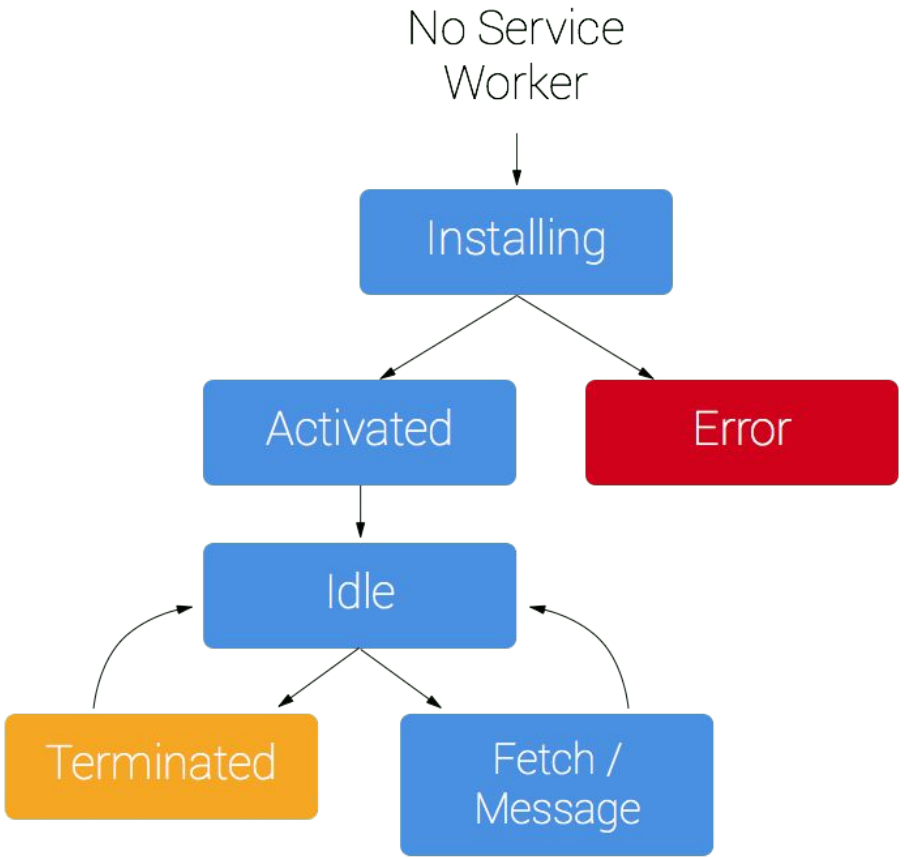
¿Como se usa?

La utilización de un ServiceWorker tiene dos pasos centrales:

1. Crear el archivo encargado de interceptar las peticiones http (serviceWorker.js)
2. Registrar ese archivo como el ServiceWorker activo en la página de inicio.

Ciclo de vida un ServiceWroker

1. Se registra un serviceWorker
2. Al registrarse, se llama al evento ‘install’, que se ejecuta por primera y única vez.
3. Si la instalación fue correcta, se llama al evento ‘activate’. Dicho evento actualiza los datos y pone en funcionamiento el ServiceWorker
4. Una vez activado, el ServiceWroker comienza a interceptar peticiones HTTP a través del evento ‘fetch’.
5. Ocasionalmente puede recibir mensajes de otras APIs



1) Crear el ServiceWroker

En esencia, este archivo contiene los eventos del ciclo de vida: install, activate y fetch principalmente.

En este archivo nosotros podemos definir el código a ejecutar en cada uno de ellos.

Los eventos install y activate son ExtendableEvents, por lo que podemos usar event.waitUntil(*promesa*) para pausar la ejecución hasta que *promesa* esté cumplida.

El evento 'fetch' es un FetchEvent que nos provee el event.respondWith(*response*)

```
self.addEventListener('install', event => {
    // Puedo usar event.waitUtil() para ejecutar una promesa
})

self.addEventListener('activate', event => {
    // Puedo usar event.waitUntil() para ejecutar una promesa
});

self.addEventListener('fetch', event => {
    // Puedo usar event.respondWith() para dar una respuesta
    // personalizada
})
```

Éste es un archivo .js en el root de la app. (por ejemplo, se puede llamar 'serviceWorker.js')

2) Registrar el ServiceWroker

1. No todos los navegadores soportan esta interfaz. Por eso necesitamos primero verificar.
2. Si el navegador lo soporta se procede con la registración. El método regsiter() recibe la url a nuestro archivo service worker y retorna una promesa que, si se resuelve, nos devuevle un objeto registración. Dicho objeto nos permite obtener información global del ServiceWorker (p. ej. a partir de cual url empieza a interceptar peticiones - es el *scope*)

```
if ("serviceWorker" in navigator) {
  navigator.serviceWorker
    .register("urlAlServiceWorker.js")
    .then(function (registracion) {
      console.log(
        "Service Worker registrado correctamente",
        registracion.scope
      );
    });
}
```

Éste fragmento de código en el index de nuestro sitio

¡Muchas gracias!

¡Sigamos trabajando!