background sync and background fetch

Service workers are awesome at what they do. They provide better performance, offline capabilities and the ability to intercept request and provide custom responses.

Out of the box they don't work well with large files or with background processing of files. This is where background sync and background fetch come in... They provides means to do one off sync and fetch large files in the background.

Background Sync

The service worker API has had background sync for a while which, in its current iteration, allows the service worker to defer work until the user has connectivity. This means the user could type a message, hit send, and leave the site knowing that the message will be sent either now, or when they have connectivity.

You can also use background sync to cache individual items in a page independently of the status of the user's network or whether the user is actively engaged with the site.

The code belows assumes that we've already registered a service worker. The code below waits for the service worker to load then registers a sync with a name or tag. We'll use the tag in the service worker code to identify the event that triggered the sync.

If the `navigator.serviceWorker.ready` promise is rejected (most likely because the Operating System rejected the sync) we'll trigger the catch portion of the promise and run the command we wanted to sync.

If the browser doesn't support service workers or, more likely, background sync, we run the command anyway.

```
if ('serviceWorker' in navigator && 'SyncManager' in window) {
  navigator.serviceWorker.ready.then(function(reg) {
    return reg.sync.register('myFirstSync');
```

```
  }).catch(function() {
    postDataFromThePage();
  });
} else {
  postDataFromThePage();
}
```

In the service worker we register a new event to handle the sync requests. We can perform different actions depending on the tag that triggered the request. In the simplest form, we match the sync event tag and perform something... `postDataFromThePage()` will cache resources and perform the tasks we set it up to.

```
self.addEventListener('sync', function(event) {
  if (event.tag == 'myFirstSync') {
    event.waitUntil(postDataFromThePage());
  }
});
```

It's important to know that this is for one off sync operations, not for repeated operations or to fetch large files like audio or video files.

According to caniuse (background sync api) the feature is Chrome-only at the moment, Edge and Firefox have it under development.

Background Fetch

> This API currently works in Chrome 71 with the experimental web features flag enabled or through an origin trial.

Background fetch is different. The API allows you to download large payloads like MP3 or DASH video segments. The download will pause if the user loses connectivity and resume when they get back online. It is also user-initiated; background fetch will

The idea is like this:

1. You tell the browser to perform a group of fetches in the background.
2. The browser fetches those things, displaying progress to the user.
3. Once the fetch has completed or failed, the browser opens your service worker and fires an event to tell you what happened. This is where you decide what to do with the responses, if anything.

If the user closes pages to your site after step 1, that's ok, the download will continue. Because the fetch is highly visible and easily abortable, there isn't the privacy concern of a way-too-long background sync task. Because the service worker isn't constantly running, there isn't the concern that it could abuse the system, such as mining bitcoin in the background.

If the user starts the download while offline, or goes offline during the download, the background fetch will be paused and resumed when there is connectivity.

Background Fetch is an ideal tool if you want to allow users to selectively download longer pieces of content like audio, video or longer chapters or stories in a collection.

Building your background fetch

Because we're asking the user to decide when they want to fetch the content, we can't rely on the service worker itself to do the update.

- Background Sync
    - Background Sync Specification
    - Introducing Background Sync
    - Background Sync by Dean Hume at Ponyfoo
- Background Fetch
    - Introducing Background Fetch
    - Background Fetch API: Get Ready To Use It!
    - Background Fetch Demo App by Jake Archibald