Service worker communication

Contents ›

Importing `ServiceWorkerModule` into your `AppModule` doesn't just register the service worker, it also provides a few services you can use to interact with the service worker and control the caching of your app.

Prerequisites

A basic understanding of the following:

- Getting Started with Service Workers.

`SwUpdate` service

The `SwUpdate` service gives you access to events that indicate when the service worker has discovered an available update for your app or when it has activated such an update—meaning it is now serving content from that update to your app.

The `SwUpdate` service supports four separate operations:

- Getting notified of *available* updates. These are new versions of the app to be loaded if the page is refreshed.
- Getting notified of update *activation*. This is when the service worker starts serving a new version of the app immediately.
- Asking the service worker to check the server for new updates.
- Asking the service worker to activate the latest version of the app for the current tab.

Available and activated updates

The two update events, `available` and `activated`, are `Observable` properties of `SwUpdate`:

log-update.service.ts

```
@Injectable()
export class LogUpdateService {

  constructor(updates: SwUpdate) {
    updates.available.subscribe(event => {
      console.log('current version is', event.current);
      console.log('available version is', event.available);
    });
    updates.activated.subscribe(event => {
      console.log('old version was', event.previous);
      console.log('new version is', event.current);
    });
  }
}
```

You can use these events to notify the user of a pending update or to refresh their pages when the code they are running is out of date.

Checking for updates

It's possible to ask the service worker to check if any updates have been deployed to the server. The service worker checks for updates during initialization and on each navigation request—that is, when the user navigates from a different address to your app. However, you might choose to manually check for updates if you have a site that changes frequently or want updates to happen on a schedule.

Do this with the `checkForUpdate()` method:

check-for-update.service.ts

```
import { ApplicationRef, Injectable } from '@angular/core';
import { SwUpdate } from '@angular/service-worker';
import { concat, interval } from 'rxjs';
import { first } from 'rxjs/operators';

@Injectable()
export class CheckForUpdateService {

  constructor(appRef: ApplicationRef, updates: SwUpdate) {
    // Allow the app to stabilize first, before starting polling for updates with
`interval()`.
    const appIsStable$ = appRef.isStable.pipe(first(isStable => isStable === true));
    const everySixHours$ = interval(6 * 60 * 60 * 1000);
    const everySixHoursOnceAppIsStable$ = concat(appIsStable$, everySixHours$);

    everySixHoursOnceAppIsStable$.subscribe(() => updates.checkForUpdate());
  }
}
```

This method returns a `Promise` which indicates that the update check has completed successfully, though it does not indicate whether an update was discovered as a result of the check. Even if one is found, the service worker must still successfully download the changed files, which can fail. If successful, the `available` event will indicate availability of a new version of the app.

In order to avoid negatively affecting the initial rendering of the page, `ServiceWorkerModule` waits for up to 30 seconds by default for the app to stabilize, before registering the ServiceWorker script. Constantly polling for updates, for example, with setInterval() or RxJS' interval(), will prevent the app from stabilizing and the ServiceWorker script will not be registered with the browser until the 30 seconds upper limit is reached.

Note that this is true for any kind of polling done by your application. Check the isStable documentation for more information.

You can avoid that delay by waiting for the app to stabilize first, before starting to poll for updates, as shown in the example above. Alternatively, you might want to define a different registration strategy for the ServiceWorker.

Forcing update activation

If the current tab needs to be updated to the latest app version immediately, it can ask to do so with the `activateUpdate()` method:

prompt-update.service.ts

```
@Injectable()
export class PromptUpdateService {

  constructor(updates: SwUpdate) {
    updates.available.subscribe(event => {
      if (promptUser(event)) {
        updates.activateUpdate().then(() => document.location.reload());
      }
    });
  }
}
```

Calling `activateUpdate()` without reloading the page could break lazy-loading in a currently running app, especially if the lazy-loaded chunks use filenames with hashes, which change every version. Therefore, it is recommended to reload the page once the promise returned by `activateUpdate()` is resolved.

More on Angular service workers

You may also be interested in the following:

- Service Worker in Production.