Precache Files with workbox-build

This page explains how to use the workbox-build Node module to generate the list of files to precache and add it to your service worker.

Note: You'll need to have Node installed to use workbox-build.

Install `workbox-build`

Start by installing `workbox-build` from npm.

```
$ npm install workbox-build --save-dev
```

Add an Injection Point

Before the files can be "injected" into your service worker, you need to add this line of code to your service worker file:

```
workbox.precaching.precacheAndRoute([]);
```

This piece of code will be replaced by workbox-build with the list of files (See the next section).

Call `injectManifest()`

The final step is to add workbox-build to your build process or script:

```
const workboxBuild = require('workbox-build');

// NOTE: This should be run *AFTER* all your assets are built
const buildSW = () => {
  // This will return a Promise
  return workboxBuild.injectManifest({
    swSrc: 'src/sw.js',
    swDest: 'build/sw.js',
    globDirectory: 'build',
    globPatterns: [
```

```
      '**\/*.{js,css,html,png}',
    ]
  }).then(({count, size, warnings}) => {
    // Optionally, log any warnings and details.
    warnings.forEach(console.warn);
    console.log(`${count} files will be precached, totaling ${size} bytes.`);
  });
}

buildSW();
```

This command will create a list of files to precache, read in your service worker file, inject the manifest and output a new service worker file with the manifest. The result should look something like this:

```
workbox.precaching.precacheAndRoute([
  {
    "url": "basic.html",
    "revision": "7ca37fd5b27f91cd07a2fc68f20787c3"
  },
  {
    "url": "favicon.ico",
    "revision": "1378625ad714e74eebcfa67bb2f61d81"
  },
  {
    "url": "images/hamburger.svg",
    "revision": "d2cb0dda3e8313b990e8dcf5e25d2d0f"
  },

  ...

]);
```

Please ensure that you build the service worker after a file change to ensure your users get the latest files.

Using with Gulp

Using **`workbox-build`** with your Gulp process is simply a case of using the same code as above.

```
const gulp = require('gulp');
const workboxBuild = require('workbox-build');
```

```
gulp.task('service-worker', () => {
  return workboxBuild.injectManifest({
    swSrc: 'src/sw.js',
    swDest: 'build/sw.js',
    globDirectory: 'build',
    globPatterns: [
      '**\/*.{js,css,html,png}',
    ]
  }).then(({count, size, warnings}) => {
    // Optionally, log any warnings and details.
    warnings.forEach(console.warn);
    console.log(`${count} files will be precached, totaling ${size} bytes.`);
  });
});
```

After this you can simply run the task via **`gulp service-worker`** or add it to the end of another Gulp task.

Further Reading

The **`workbox.precaching`** API provides some options for configuration should you need to customize any of the default behaviors. Learn more on the workbox.precaching page.

Last updated May 21, 2018.