Add to Home Screen

By Pete LePage
Pete is a Developer Advocate

Add to Home Screen, sometimes referred to as the web app install prompt makes it easy for users to install your Progressive Web App on their mobile or desktop device. When installed, it adds your PWA to to their launcher, and runs it like any other installed app.

Chrome handles most of the heavy lifting for you, and on Android, Chrome will generate a WebAPK creating an even more integrated experience for your users.

What is the criteria?

In order for a user to be able to install your Progressive Web App, it needs to meet the following criteria:

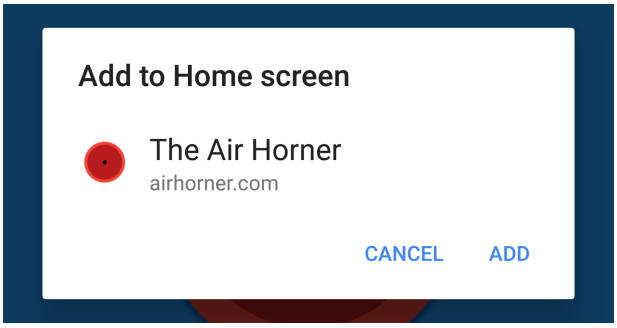
- The web app is not already installed
- Meets a user engagement heuristic (currently, the user has interacted with the domain for at least 30 seconds)
- Includes a web app manifest that includes:
  - `short_name` or `name`
  - `icons` must include a 192px and a 512px sized icons
  - `start_url`
  - `display` must be one of: `fullscreen`, `standalone`, or `minimal-ui`
- Served over **HTTPS** (required for service workers)
- Has registered a service worker with a `fetch` event handler

When these criteria are met, Chrome will fire a `beforeinstallprompt` event that you can use to prompt the user to install your Progressive Web App.

Note: Other browsers have different criteria for installation, or to trigger the `beforeinstallprompt` event, check their respective sites for full details: Edge, Firefox, Opera, Samsung Internet, and UC Browser.

If the web app manifest includes **`related_applications`** and has **`"prefer_related_applications": true`**, the native app install prompt will be shown instead.

Show the add to home screen dialog



Add to Home Screen dialog on Android

In order to show the Add to Home Screen dialog, you need to:

1. Listen for the **`beforeinstallprompt`** event
2. Notify the user your app can be installed with a button or other element that will generate a user gesture event.
3. Show the prompt by calling **`prompt()`** on the saved **`beforeinstallprompt`** event.

Note: Chrome 67 and earlier showed an add to home screen banner, the banner was removed in Chrome 68.

Listen for `beforeinstallprompt`

If the add to home screen criteria are met, Chrome will fire a **`beforeinstallprompt`** event, that you can use to indicate your app can be 'installed', and then prompt the user to install it.

When the **`beforeinstallprompt`** event has fired, save a reference to the event, and update your user interface to indicate that the user can add your app to their home screen.

```
let deferredPrompt;

window.addEventListener('beforeinstallprompt', (e) => {
  // Prevent Chrome 67 and earlier from automatically showing the prompt
  e.preventDefault();
  // Stash the event so it can be triggered later.
```

```
  deferredPrompt = e;
});
```

Notify the user your app can be installed

The best way to notify the user your app can be installed is by adding a button or other element to your user interface. **Don't show a full page interstitial or other elements that may be annoying or distracting.**

```
window.addEventListener('beforeinstallprompt', (e) => {
  // Prevent Chrome 67 and earlier from automatically showing the prompt
  e.preventDefault();
  // Stash the event so it can be triggered later.
  deferredPrompt = e;
  // Update UI notify the user they can add to home screen
  btnAdd.style.display = 'block';
});
```

Success: you may want to wait before showing the prompt to the user, so you don't distract them from what they're doing. For example, if the user is in a check-out flow, or creating their account, let them complete that before interrupting them with the prompt.

Show the prompt

To show the add to home screen prompt, call `prompt()` on the saved event from within a user gesture. It will show a modal dialog, asking the user to to add your app to their home screen.

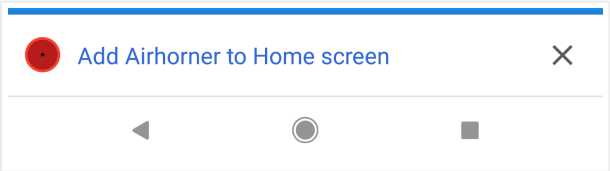
Then, listen for the promise returned by the `userChoice` property. The promise returns an object with an `outcome` property after the prompt has shown and the user has responded to it.

```
btnAdd.addEventListener('click', (e) => {
  // hide our user interface that shows our A2HS button
  btnAdd.style.display = 'none';
  // Show the prompt
  deferredPrompt.prompt();
  // Wait for the user to respond to the prompt
  deferredPrompt.userChoice
    .then((choiceResult) => {
      if (choiceResult.outcome === 'accepted') {
        console.log('User accepted the A2HS prompt');
```

```
      } else {
        console.log('User dismissed the A2HS prompt');
      }
      deferredPrompt = null;
    });
});
```

You can only call **`prompt()`** on the deferred event once, if the user dismissed it, you'll need to wait until the **`beforeinstallprompt`** event is fired on the next page navigation.

The mini-info bar



The mini-infobar

The mini-infobar is an interim experience for Chrome on Android as we work towards creating a consistent experience across all platforms that includes an install button into the omnibox.

The mini-infobar is a Chrome UI component and is not controllable by the site, but can be easily dismissed by the user. Once dismissed by the user, it will not appear again until a sufficient amount of time has passed (currently 3 months). The mini-infobar will appear when the site meets the add to home screen criteria, regardless of whether you **`preventDefault()`** on the **`beforeinstallprompt`** event or not.

Feedback

Determine if the app was successfully installed

To determine if the app was successfully added to the users home screen *after* they accepted the prompt, you can listen for the **`appinstalled`** event.

```
window.addEventListener('appinstalled', (evt) => {
  app.logEvent('a2hs', 'installed');
});
```

Detecting if your app is launched from the home screen

`display-mode` media query

The **`display-mode`** media query makes it possible to apply styles depending on how the app was launched, or determine how it was launched with JavaScript.

To apply a different background color for the app above when being launched from the home screen with "**`display`**: "**`standalone`**", use conditional CSS:

```
@media all and (display-mode: standalone) {
  body {
    background-color: yellow;
  }
}
```

It's also possible to detect if the **`display-mode`** is standalone from JavaScript:

```
if (window.matchMedia('(display-mode: standalone)').matches) {
  console.log('display-mode is standalone');
}
```

Safari

To determine if the app was launched in **`standalone`** mode in Safari, you can use JavaScript to check:

```
if (window.navigator.standalone === true) {
  console.log('display-mode is standalone');
}
```

Updating your app's icon and name

If you change any of the properties in your manifest, those changes will be reflected to the user after they've run your app again.

Tip: Icons may be cached, so it may be helpful to change the filenames when updating icons or other graphics.

Test your add to home screen experience

You can manually trigger the `beforeinstallprompt` event with Chrome DevTools. This makes it possible to see the user experience, understand how the flow works or debug the flow. If the PWA criteria aren't met, Chrome will throw an exception in the console, and the event will not be fired.

Caution: Chrome has a slightly different install flow for desktop and mobile. Although the instructions are similar, testing on mobile **requires** remote debugging, without it, it will use the desktop install flow.

Chrome for Android

1. Open a remote debugging session to your phone or tablet.
2. Go to the **Application** panel.
3. Go to the **Manifest** tab.
4. Click **Add to home screen**

Chrome OS

1. Open Chrome DevTools
2. Go to the **Application** panel.
3. Go to the **Manifest** tab.
4. Click **Add to home screen**

Dogfood: To test the install flow for Desktop Progressive Web Apps on Mac or Windows, you'll need to enable the `#enable-desktop-pwas` flag.

Will `beforeinstallprompt` be fired?

The easiest way to test if the **`beforeinstallprompt`** event will be fired, is to use Lighthouse to audit your app, and check the results of the User Can Be Prompted To Install The Web App test.

Last updated July 24, 2018.