Enhancements

- App shortcuts
- iOS and iPadOS
  - Splash screens
    - Design patterns for iOS startup images
  - Detecting a PWA on Apple mobile platforms
  - Fullscreen support
  - Installation reliability
- Resources

There are many enhancements that can improve the conversion and usage of your PWA.

App shortcuts

App shortcuts is a static list of deep links to your PWA, they are written in your manifest. Web App Manifest spec. It lets you define a list of shortcuts to different parts or features in your PWA, they accelerate navigation to frequently accessed sections.

App shortcuts are available on most desktop operating systems and Android with WebAPK, and they appear in a contextual menu on the app's icon in the home screen, dock, or taskbar, as in the following image:

{% Img src="image/RK2djpBgopg9kzCyJbUSjhEGmnw1/SYm7meKqd286Z62gehX9.png", alt="App shortcuts in Android and macOS.", width="800", height="280" %}

To access this menu, users have to right-click or long-press on the PWA's icon.

{% Aside %} App shortcuts are only available if the user installs your PWA. {% endAside %}

{% Aside 'warning' %} Once the user has installed your PWA, you can't add or remove shortcuts until you update your manifest on the server and the update algorithm is triggered. {% endAside %}

Shortcuts are defined in the `shortcuts` member of the manifest. It takes an array of members with the following properties: `name` : The text that will be shown to the user, typically in a context menu. `start_url`. : The URL the PWA should load when the user starts it from this shortcut. It should be a URL within your PWA scope, and it should deep-link to the feature the `name` or `short_name` describes. `short_name` (Optional) A shorter name used when there is not enough room to display the full value of the `name` field. `description` : (Optional) A description of what this shortcut will do `icons` : (Optional) An array of icon objects with `src`, `type`, `sizes`, and optional `purpose` fields, describing what images should represent the shortcut

You should treat App shortcuts as a best-effort ability. That means that you can't rely on these shortcuts to appear consistently, and even if they appear, you don't know how many shortcuts will appear or if the platform will ignore the icons as its at the discretion of the browsers. A full discussion per platform is out of scope but below you have an idea of how it works on Android and desktop. The best way to deal with this uncertainty is to order the items by priority.

{% Aside 'caution' %} Chrome and Edge on desktop renders the first 10 items, while Chrome on Android only renders the first three shortcuts in the manifest's list, so it's better if you order them by priority. Chrome on Android also adds access to "App Info" and "Site Settings" from the menu after your custom list. {% endAside %}

{% Aside %} If you publish your PWA to app catalogs and stores, you can convert the shortcuts in the manifest to platform-specific implementations using APIs specific to each operating system, such as Quick Actions on iOS and shortcuts on Android. BubbleWrap supports App shortcuts for Android and ChromeOS and PWA Builder for Android, ChromeOS, iOS and iPadOS. {% endAside %}

The following code sample defines different app shortcuts you can try if you install the app on Android with Chrome or on desktop with Edge or Chrome.

{% Glitch 'mlearn-pwa-app-shortcuts' %}

iOS and iPadOS

When publishing PWAs, there are some enhancements that can improve the experience for users on Safari on iOS/iPadOS.

{% Aside 'warning' %} As we mentioned in previous chapters, PWAs on iOS and iPadOS are only installable from Safari; other browsers are not able to install PWAs in the home screen. {% endAside %}

Splash screens

As seen in the Web App Manifest chapter, Android creates splash screens automatically based on the manifest's values. That's not the case for iOS and iPadOS. In these devices, you should define the splash screens in the HTML as static images using `<link>` elements.

These images are known as startup images on Apple devices and they use the `rel` property with `apple-touch-startup-image` value, as in:

```
<link rel="apple-touch-startup-image" href="ios-startup.png">
```

The challenge is that the startup image must have the exact window size that your PWA will have on opening. So, different iOS and iPadOS devices will need different images. More situations need to be covered on the iPad, such as landscape/portrait openings and rendering the PWA in multitask mode (such as 1/3,1/2, or 2/3 of the screen).

You can check an updated list of iOS and iPadOS screen sizes at the Apple Human Interface Guidelines

Different versions of the launch image can be set with a media query inside the `media` attribute:

```
<link rel="apple-touch-startup-image" href="ios-startup.png"
      media="orientation: portrait">
<link rel="apple-touch-startup-image" href="ios-startup-landscape.png"
      media="orientation: landscape">
```

{% Aside 'warning' %} If you want to provide all possible options combining screen definitions, orientations, and multitask mode, you will end up with more than 25 different images you need to create and link in your HTML. If you don't provide a startup image for your user's iOS or iPadOS context, a white screen will appear instead in the opening animation, and while your PWA is loading {% endAside %}

Design patterns for iOS startup images

Defining startup images is hard work, so we have a couple of tools for automated generation and configuration:

- Static generation integrates with your build system, creates all the PNG static images and gives you the HTML code with `<link>` elements to inject into your document. PWA Asset Generator is an example of such a tool.
- Client-side generator, a JavaScript tool that can embed one or more base64 versions of the startup image into `<link>` injected elements based on the current device's type and screen size. You can use an in-memory canvas, render the image and convert it into a `data:` URI with a PNG file. The PWA Compat library is an easy-to-use client-side library that does this by cloning the Android's typical launch screen.

Detecting a PWA on Apple mobile platforms

While you should use Progressive Enhancement and feature detection in your PWA, there may be some edge cases where we need to know if the user is in a PWA on Apple mobile platforms, such as when you want to offer installation instructions or add links to platform-specific apps that are iOS-only.

To avoid reading the user agent string, check the `standalone` property of the `navigator` object. This is a non-standard property, and it's only available on the WebKit engine on iOS and iPadOS.

- If `navigator.standalone` is `undefined` it means the user is not on an iPadOS or iOS device.
- If `navigator.standalone` is `false` it means the user opened the PWA in the browser and is using it there.
- If `navigator.standalone` is `true` it means the user opened the PWA from the home screen and is getting the standalone PWA experience.

Fullscreen support

On Safari on iOS and iPads, only `display: standalone` is supported as a display mode for your PWA's icon. While `display: fullscreen` is not supported as on Android devices, it's possible to use a non-standard meta-tag to make your PWA enter fullscreen mode.

In the next image, you can see at the left a default standalone design with a theme color, and at the right a PWA with a fullscreen iOS mode that lets you render content behind the status bar.

{% Img src="image/RK2djpBgopg9kzCyJbUSjhEGmnw1/SYjZ1upKyQaMuF1nrv7Y.png", alt="A standalone default behavior (left) and a fullscreen iOS screen (right).", width="800", height="216" %}

If you add the following tag in your HTML your PWA on iOS and iPadOS will enter full-screen mode, but it is different from Android.

```
<meta name="apple-mobile-web-app-status-bar-style" content="black-translucent">
```

In this mode, the device's status bar (the top where you see the clock, battery level, and notification icons) is still visible but rendered on top of your content with a transparent background.

{% Aside 'warning' %} The theme color is ignored in this mode, and the notification bar is rendered over a transparent background. Also, your app's [0, 0] pixel is now the [0, 0] pixel of the physical screen and not after the safe area. {% endAside %}

When using this mode, be careful with your design because the operating system will always render the icons in white, so you should always contrast your backgrounds for the top of the screen with light content. Also, it's important to use CSS environment variables to render content in the safe area, as seen in App Design Chapter.

{% Img src="image/RK2djpBgopg9kzCyJbUSjhEGmnw1/WZSHybaKkUvpLcphGJ8y.png", alt="The top 0px of your viewport is below the status bar by default; if you add a black-translucent meta tag, the top 0px of your viewport will match the physical top of the screen", width="800", height="219" %}

Installation reliability

Safari on iOS and iPadOS before 15.4 only loads the manifest file from the network when the user opens the share sheet -using the share icon within the browser- and not when the page loads. Therefore, Safari doesn't check if your website is a PWA until that moment, which can lead to situations where the manifest can't be loaded or takes too much time, and Safari ignores it.

When Safari can't load the manifest on time, pressing "Add to Home Screen" places an icon on the home screen, but does not provide an app experience; it will merely be a shortcut to a Safari tab.

{% Aside 'warning' %} If Safari can't load the manifest, it will fall back to check if your PWA has some deprecated meta tags, such as `apple-mobile-web-app-capable`. You should not use these metatags. They provide a home screen app experience without essential attributes for your PWA, such as honoring the `start_url` or the `scope` attributes, making a terrible app experience. {% endAside %}

Resources

- PWA Asset Generator
- Get things done quickly with app shortcuts
- PWA Compat Library
- Define App Shortcuts

□□□ #2
□ □□□ □□□ 2026-10-31 22:55:20 UTC
□ □□□ □□□ 2026-10-31 22:55:22 UTC