Engineer

Web app manifest

a JSON file describing a site as an installable application

What it is

A web app manifest is a JSON document that describes a website as an application: what it is called, which address it opens at, which addresses belong to it, how its window should look and which pictures stand for it. A page points to the file with a `link` element whose `rel` is `manifest`, and a browser that offers installation reads it to decide what the installed copy will be. The W3C specification that defines it calls it just the manifest, and so does most everyday talk about it.

The file is plain JSON. Its registered media type is `application/manifest+json` and its registered extension is `.webmanifest`, although the specification accepts any JSON media type and plenty of sites serve a `manifest.json`. Every member is optional. A small one for an imaginary tide-table site looks like this:

```
{
    "name": "Harbour Tide Tables",
    "short_name": "Tides",
    "start_url": "/tides/",
    "display": "standalone",
    "background_color": "#ffffff",
    "icons": [
        {
            "src": "/icons/tides-512.png",
            "sizes": "512x512",
            "type": "image/png"
        }
    ]
}
```

Three members decide what the app is, as distinct from how it looks. `start_url` suggests the page the installed app opens at; a browser may ignore it or let the user change it. `scope` is the set of addresses that count as inside the app; when it is missing, the specification takes the start URL with its file name, query and fragment cut off, so the example above owns everything under `/tides/`. `id` is the app's identity, and when it is missing it falls back to the start URL. Relative icon addresses and the start URL resolve against the manifest's own URL rather than the page that linked to it. A relative `id` instead resolves against the start URL's origin.

The rest is presentation. `display` asks for `fullscreen`, `standalone`, `minimal-ui` or `browser`, and a browser that cannot honour the request steps down that list in order. Each icon may carry a `purpose`: `any` by default, `maskable` for a picture drawn to survive cropping, or `monochrome` for one whose colour is thrown away so that only its alpha channel is used. `theme_color` tints the window around the app. A companion W3C note adds `description`, `categories` and `screenshots`, which install dialogs and stores may show.

Use the concept in a decision

Put “What is a web app manifest?” to work in a real decision. We can help relate the concept to the system you are building or buying. Interfaces that load quickly, work on a phone and behave the same in every browser your customers happen to use.

Explore our services Get in touch →

Test the assumptions behind the term

Use the explanation and linked work to frame questions for your project. We can examine the assumptions with you before a tool is chosen. Directed full-stack GIS map development — full-stack mapping, from database to browser

Directed full-stack GIS map development Get in touch →

Why it matters

The manifest supplies installation metadata that the site controls. The specification leaves the decision to install with the browser: it may offer a way to install, and each browser sets its own conditions, some of which require particular members. A team ships the input and the browser decides what to do with it, so the useful question is less whether to have a manifest than what it promises.

Identity is the promise that outlives the file. When a browser fetches the manifest again, the specification says it should use a matching `id` as a signal that the new file replaces the manifest of an app already installed. Because a missing `id` is the start URL, moving the start page quietly changes the identity,

so the browser should treat the new file as a different application from the one on the reader's device. Pinning `id` on the first release costs one line and removes that risk.

Scope decides where the app window ends. Navigation inside the scope stays in the installed window. A link that leaves it still opens, but the specification expects the browser to show the reader that they are outside the app, so a scope narrower than the site makes ordinary links between its sections look like exits.

Icons are where most of the visible damage happens. Launchers crop icons to circles, rounded squares and other shapes, and a logo that fills its whole square loses its edges. The specification defines the safe zone of a maskable icon as a circle in the middle with a radius of two fifths of the icon's shorter side, and anything outside it may be cut. A monochrome icon keeps only its alpha channel, so an icon drawn on an opaque background turns into a solid square.

The colour members are narrower than they look. `background_color` is meant only for the moment before the stylesheet arrives, and the specification forbids the browser from using it once the stylesheet is available. It is the colour of the first frame, and nothing after it.

What it looks like in practice

This site builds its manifest from a template rather than keeping a file by hand. Its Hugo configuration registers `application/manifest+json` with the suffix `webmanifest` and makes the manifest an output of each language's home page, so every language gets its own file, and the head of each page links the file for its own language. The configuration records why: a single global copy can name only one start URL, and a Danish visitor who installed the site got an app that opened the English home.

The template assembles the manifest as a map and hands it to Hugo's JSON encoder, so escaping and commas are the encoder's problem and a missing comma cannot ship. Every icon address starts at the root of the host. Its comment spells out the reason: a relative address in `/da/manifest.webmanifest` would resolve under `/da/`, and every icon in Danish would 404. The icons themselves are read off disk from three folders, one per purpose, ten sizes each from 36 to 512 pixels, with the size taken from the file name, so a size added by the generator appears in the manifest with nothing to edit.

`id` is pinned to the language's home address, and `scope` is the whole origin, so a link into another language stays inside the installed window. Screenshots are named `<lang>-<form_factor>-<WxH>.webp`, and the template reads all three facts off the file name: English and Danish each carry one wide shot at 1280 by 800 and one narrow shot at 780 by 1688. A language without screenshots gets no `screenshots` member at all, rather than an empty list.

The credits page lists the W3C specification among the standards the site follows, and PWABuilder among the outside checks anyone can run: a service that reads the manifest and the rest of what makes the site installable, from Microsoft's machines.

The install page is the reader's half of the same file. It walks through where each browser keeps the install command and notes that Apple's browsers never offer it unprompted. Firefox has a separate web-app window feature enabled by default on Windows from version 143, or version 150 for Microsoft Store installations. Mozilla's implementation documentation lists Linux support as disabled by default and macOS support as unavailable. The page also sets expectations for this site: the installed app is an icon and a window of its own. The site publishes no service worker or managed offline copy, so it needs a connection for normal use.

Where it goes wrong

Every icon 404s on a translated path. A relative `src` resolves against the manifest's URL, so a manifest served from a language prefix asks for icons under that prefix. Write icon addresses from the root of the host, or as full URLs.

The installed app opens in the wrong language. One manifest can name one start URL, so every reader installs the same home page. Serve one manifest per language, each with its own `start_url` and `id`, and link the matching one from each page.

Installed copies ignore the new manifest. With no `id`, the identity is the start URL, so moving the start page gives the file a new identity, and the browser should treat it as a different app from the one readers already have. Set `id` before the first release and leave it alone afterwards.

The launcher crops the logo. An icon with purpose `any` is placed or cut as the platform sees fit. Add a `maskable` icon with everything that matters inside the central circle whose radius is 40% of the icon's shorter side.

Edits to the manifest change nothing. The HTML standard uses only the first `link` element whose `rel` contains `manifest`, so when a theme or plugin has put its own link earlier in the page, the file being edited is never the one read. Keep exactly one manifest link per page.

Further reading

- Web Application Manifest — every member, how a browser processes the file, and the rules for `id`, `scope` and icon purposes.
- Web App Manifest - Application Information — `description`, `categories` and `screenshots`, with form factors and labels.
- IANA: application/manifest+json — the registration of the media type and its `.webmanifest` extension.
- HTML Standard: link type manifest — how the manifest link is fetched and why only the first one counts.
- Use web apps in Firefox for Windows — the Windows feature and its version 143 and Microsoft Store version 150 availability.
- Firefox Source Docs: Web Apps in Firefox — the current implementation: Windows enabled by default, Linux disabled by default, and macOS unsupported.

Glossary

Choose for your actual constraints

Describe the workflow, the existing interfaces and the users who need the change. We can define a testable delivery and a handover that lets your team maintain it.

See what it costs Get in touch →

Make the terminology practical

Ask how “What is a web app manifest?” applies to your organisation. A concrete example from your current system is enough to start the discussion.

Explore our services

welcome@engineer.company +45 50 10 61 84 Copenhagen, Denmark

Take this page with you

Contact card

Carefully Crafted With Love by D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – present · Engineer ApS · All rights reserved.

Printed from https://engineer.company/glossary/webmanifest/