Engineer

Manifeste d'application web

fichier JSON décrivant un site comme application installable

Ce que c’est

Un manifeste d’application web est un document JSON qui décrit un site web comme une application : son nom, l’adresse à laquelle elle s’ouvre, les adresses qui lui appartiennent, l’apparence de sa fenêtre et les images qui la représentent. Une page désigne le fichier par un élément `link` dont le `rel` vaut `manifest`. Un navigateur proposant l’installation le lit pour déterminer ce que sera la copie installée. La spécification du W3C qui le définit l’appelle simplement manifeste, comme la plupart des échanges courants sur le sujet.

Le fichier est du JSON ordinaire. Son type de média enregistré est `application/manifest+json` et son extension enregistrée `.webmanifest`, même si la spécification accepte tout type de média JSON et si de nombreux sites servent un `manifest.json`. Chaque champ est facultatif. Voici un petit exemple pour un site imaginaire de tables des marées :

```
{
    "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"
        }
    ]
}
```

Trois champs déterminent ce qu'est l'application, par opposition à son apparence. `start_url` suggère la page sur laquelle s'ouvre l'application installée ; un navigateur peut l'ignorer ou permettre à l'utilisateur de la modifier. `scope` est l'ensemble des adresses considérées comme internes à l'application. S'il manque, la spécification prend l'URL de départ sans son nom de fichier, sa chaîne de requête ni son fragment. L'exemple ci-dessus couvre donc tout ce qui se trouve sous `/tides/` . `id` est l'identité de l'application et, s'il manque, reprend l'URL de départ. Les adresses relatives des icônes et l'URL de départ sont résolues par rapport à l'URL du manifeste lui-même, et non à la page qui y renvoie. Un `id` relatif est, lui, résolu par rapport à l'origine de l'URL de départ.

Le reste relève de la présentation. `display` demande `fullscreen` , `standalone` , `minimal-ui` ou `browser` . Un navigateur qui ne peut satisfaire la demande descend dans cette liste dans l'ordre. Chaque icône peut porter un `purpose` : `any` par défaut, `maskable` pour une image conçue pour supporter le recadrage, ou `monochrome` pour une image dont la couleur est supprimée afin de n'utiliser que son canal alpha. `theme_color` colore la fenêtre autour de l'application. Une note complémentaire du W3C ajoute `description` , `categories` et `screenshots` , que les dialogues d'installation et les boutiques peuvent afficher.

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce qu'un manifeste d'application web ? » pour éclairer une décision réelle. Nous pouvons vous aider à relier ce concept au système que vous construisez ou achetez. Des interfaces rapides, utilisables sur téléphone et cohérentes dans tous les navigateurs de vos clients.

Découvrir nos prestations Contactez-nous →

Vérifiez les hypothèses derrière le terme

Appuyez-vous sur l'explication et les réalisations liées pour formuler les questions de votre projet. Nous pouvons examiner les hypothèses avec vous avant le choix d'un outil. Développement cartographique GIS full-stack — cartographie full-stack, de la base au navigateur

Développement cartographique GIS full-stack Contactez-nous →

Pourquoi c'est important

Le manifeste fournit des métadonnées d'installation que le site maîtrise. La spécification laisse la décision d'installer au navigateur : il peut proposer un moyen de le faire, et chaque navigateur fixe ses propres conditions, dont certaines exigent des champs particuliers. Une équipe fournit les données d'entrée, et le navigateur décide de leur usage. La question utile porte donc moins sur la présence d'un manifeste que sur ce qu'il promet.

L'identité est la promesse qui survit au fichier. Lorsqu'un navigateur récupère à nouveau le manifeste, la spécification indique qu'il devrait utiliser un `id` identique comme signal que le nouveau fichier remplace le manifeste d'une application déjà installée. Puisqu'en l'absence d' `id` , l'identité est l'URL de départ, déplacer la page de départ la modifie silencieusement. Le navigateur devrait donc traiter le nouveau fichier comme une application différente de celle présente sur l'appareil du lecteur. Fixer `id` dès la première version coûte une ligne et supprime ce risque.

La portée détermine où s'arrête la fenêtre de l'application. La navigation à l'intérieur de cette portée reste dans la fenêtre installée. Un lien qui en sort s'ouvre tout de même, mais la spécification attend du navigateur qu'il indique au lecteur qu'il se trouve hors de l'application. Une portée plus étroite que le site fait donc apparaître des liens ordinaires entre ses sections comme des sorties.

Les icônes concentrent la plupart des dégâts visibles. Les lanceurs les recadrent en cercles, carrés arrondis et autres formes ; un logo qui remplit tout son carré perd ses bords. La spécification définit la zone sûre d'une icône masquable comme un cercle central dont le rayon vaut deux cinquièmes du côté le plus court de l'icône. Tout ce qui dépasse peut être coupé. Une icône monochrome ne conserve que son canal alpha : dessinée sur un fond opaque, elle devient un carré plein.

Les champs de couleur ont un rôle plus étroit qu'il n'y paraît. `background_color` n'est prévu que pour le moment précédant l'arrivée de la feuille de style. La spécification interdit au navigateur de l'utiliser une fois cette feuille disponible. C'est la couleur de la première image affichée, et de rien ensuite.

À quoi cela ressemble en pratique

Ce site construit son manifeste à partir d'un modèle plutôt que de tenir un fichier à jour manuellement. Sa configuration Hugo enregistre `application/manifest+json` avec le suffixe `webmanifest` et fait du manifeste une sortie de la page d'accueil de chaque langue. Chacune reçoit donc son propre fichier, et l'en-tête HTML de chaque page renvoie au fichier de sa langue. La configuration en consigne la raison : une copie globale ne peut désigner qu'une URL de départ, et un visiteur ayant installé la version danoise du site obtenait une application ouvrant la page d'accueil anglaise.

Le modèle assemble le manifeste sous forme de table associative et le confie à l'encodeur JSON de Hugo. L'échappement et les virgules relèvent donc de l'encodeur, et une virgule manquante ne peut être publiée. Chaque adresse d'icône part de la racine de l'hôte. Le commentaire explique pourquoi : une adresse relative dans `/da/manifest.webmanifest` serait résolue sous `/da/`, et chaque icône danoise donnerait une 404. Les icônes elles-mêmes sont lues sur disque dans trois dossiers, un par usage, avec dix tailles chacun allant de 36 à 512 pixels. La taille est tirée du nom du fichier ; une taille ajoutée par le générateur apparaît donc dans le manifeste sans modification manuelle.

`id` est fixé à l'adresse d'accueil de la langue, et `scope` couvre toute l'origine : un lien vers une autre langue reste ainsi dans la fenêtre installée. Les captures d'écran sont nommées `<lang>-<form_factor>-<WxH>.webp`, et le modèle déduit les trois informations du nom du fichier. L'anglais et le danois comportent chacun une capture large de 1 280 par 800 et une capture étroite de 780 par 1 688. Une langue sans capture ne reçoit aucun champ `screenshots`, plutôt qu'une liste vide.

La page des crédits cite la spécification du W3C parmi les standards suivis par le site et PWABuilder parmi les contrôles externes accessibles à tous : un service qui lit le manifeste et les autres éléments rendant le site installable, depuis les machines de Microsoft.

La page d'installation présente ce même fichier du point de vue du lecteur. Elle explique où chaque navigateur place la commande d'installation et précise que les navigateurs d'Apple ne la proposent jamais spontanément. Firefox dispose d'une fonction distincte de fenêtre d'application web, activée par défaut sous Windows à partir de la version 143, ou de la version 150 pour les installations depuis le Microsoft Store. La documentation d'implémentation de Mozilla indique une prise en charge de Linux désactivée par défaut et une absence de prise en charge de macOS. La page précise aussi ce qu'il faut attendre de ce site : l'application installée dispose d'une icône et de sa propre fenêtre. Le site ne publie ni service worker ni copie hors ligne gérée ; il a donc besoin d'une connexion pour un usage normal.

Où les problèmes surviennent

Chaque icône donne une 404 sous un chemin traduit. Un `src` relatif est résolu par rapport à l'URL du manifeste. Un manifeste servi sous un préfixe de langue demande donc ses icônes sous ce préfixe. Écrivez les adresses d'icône depuis la racine de l'hôte, ou sous forme d'URL complètes.

L'application installée s'ouvre dans la mauvaise langue. Un manifeste ne peut désigner qu'une URL de départ ; tous les lecteurs installent donc la même page d'accueil. Servez un manifeste par langue, chacun avec ses propres `start_url` et `id`, et reliez chaque page au manifeste correspondant.

Les copies installées ignorent le nouveau manifeste. Sans `id`, l'identité est l'URL de départ. Déplacer cette page donne donc au fichier une nouvelle identité, et le navigateur devrait le traiter comme une application différente de celle déjà installée par les lecteurs. Définissez `id` avant la première version et n'y touchez plus ensuite.

Le lanceur recadre le logo. Une icône ayant l'usage `any` est placée ou coupée selon le choix de la plateforme. Ajoutez une icône `maskable` dont tout le contenu important tient dans le cercle central de rayon égal à 40 % du côté le plus court de l'icône.

Modifier le manifeste ne change rien. Le standard HTML n'utilise que le premier élément `link` dont le `rel` contient `manifest`. Si un thème ou une extension a inséré son propre lien plus tôt dans la page, le fichier modifié n'est donc jamais celui qui est lu. Gardez exactement un lien de manifeste par page.

Pour aller plus loin

- Web Application Manifest — chaque champ, le traitement du fichier par le navigateur et les règles d'`id`, de `scope` et des usages des icônes.
- Manifeste d'application web : informations sur l'application — `description`, `categories` et `screenshots`, avec les formats d'écran et les libellés.
- IANA : application/manifest+json — l'enregistrement du type de média et de son extension `.webmanifest`.
- Standard HTML : type de lien manifest — comment le lien de manifeste est récupéré et pourquoi seul le premier compte.
- Utiliser les applications web dans Firefox pour Windows — la fonction Windows et sa disponibilité à partir de la version 143, ou de la version 150 pour les installations depuis le Microsoft Store.
- Documentation source de Firefox : applications web dans Firefox — l'implémentation actuelle : Windows activé par défaut, Linux désactivé par défaut et macOS non pris en charge.

Glossaire

Choisissez selon vos contraintes réelles

Présentez le processus de travail, les interfaces existantes et les utilisateurs qui ont besoin du changement. Nous pouvons définir une livraison testable et une transmission permettant à votre équipe d'assurer la maintenance des éléments livrés.

Consulter les tarifs Contactez-nous →

Rendez la terminologie concrète

Demandez comment « Qu'est-ce qu'un manifeste d'application web ? » s'applique à votre organisation. Un exemple concret tiré de votre système actuel suffit pour engager la discussion.

Découvrir nos prestations

welcome@engineer.company +45 50 10 61 84 Copenhague, Danemark

Emportez cette page avec vous

Fiche de contact

Créé avec soin et amour par D.

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

Droit d'auteur © 2025 – aujourd'hui · Engineer ApS · Tous droits réservés.

Imprimé depuis https://engineer.company/fr/glossary/webmanifest/