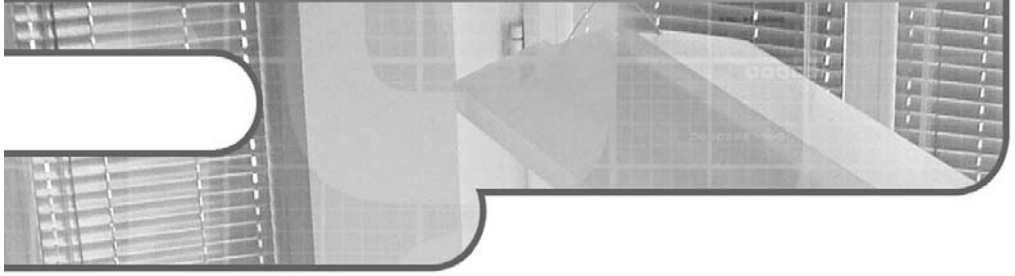

Chapitre 3
Web App Manifest

1. Introduction

La spécification Web App Manifest propose, via l'ajout d'un fichier bien particulier, d'ajouter des métadonnées à une application. Ce fichier se nomme Manifest. Ces données seront ensuite utilisées par les navigateurs Internet dans différentes situations que nous allons décrire dans ce chapitre. En effet, quand ils détecteront ce fichier, ils pourront activer certaines fonctionnalités intéressantes pour l'application. Nous pouvons par exemple ajouter un raccourci vers notre application sur l'écran d'accueil du téléphone des utilisateurs.

Voici un aperçu de ce fichier qui utilise le format JSON.

```
{
  "short_name": "PWA",
  "name": "Progressive Web Apps",
  "icons": [
    {
     "src": "/images/icons-192.png",
     "type": "image/png",
     "sizes": "192x192"
    },
    {
     "src": "/images/icons-512.png",
     "type": "image/png",
     "sizes": "512x512"
```

```
    }
  ],
  "start_url": "/pwa/?from=pwa",
  "background_color": "#3367D6",
  "display": "standalone",
  "scope": "/",
  "theme_color": "#3367D6"
}
```

Le format JSON est un format très utilisé dans le monde web. Il permet de définir, via un couple clé/valeur, des données. Les clés sont au format chaîne de caractères (par exemple, les clés `short_name`, `name` ou encore `icons`). Les valeurs, quant à elles, peuvent être des chaînes de caractères, des objets, des valeurs booléennes, des entiers ou encore des tableaux.

Pour ajouter ce fichier à notre application, nous ajoutons la balise `link` suivante dans la balise `head` de nos pages HTML. Cette balise `link` doit pointer vers un fichier utilisant l'extension .webmanifest.

Dans l'exemple ci-après, le navigateur télécharge le fichier **manifest.webmanifest** se trouvant dans le même répertoire que le fichier HTML qui est en train d'être chargé.

```
<html>
  <head>
  <meta name="viewport" content="width=device-width,
initial-scale=1.0"/>
    <link rel="manifest" href="manifest.webmanifest">
  </head>
  <body>
    ...
  </body>
</html>
```

Remarque

La plupart des navigateurs acceptent également l'extension .json. Si vous préférez utiliser cette extension, assurez-vous que ce fichier sera téléchargé en utilisant le type MIME `application/manifest+json`.

Ce fichier doit être enregistré pour toutes les pages HTML de l'application. Dans le cas d'une Single Page Application, l'opération se limite à la page index.html du projet.

Vous trouverez la spécification officielle sur le site du W3C :
https://www.w3.org/TR/appmanifest/

2. Contenu à définir

Nous allons à présent expliquer les différentes propriétés que nous pouvons définir dans ce fichier. Nous ne sommes pas obligés de définir toutes ces propriétés pour commencer à bénéficier des avantages de ce fichier. Certaines de ces propriétés sont optionnelles. Les navigateurs sont susceptibles d'ajouter dans le futur de nouvelles propriétés afin d'étendre le champ d'action de ce fichier.

– `short_name` et `name` : propriétés permettant de définir respectivement une version courte et une version longue du nom de l'application.

```
{
  "short_name": "PWA",
  "name": "Progressive Web Apps"
}
```

– `start_url` : propriété permettant de définir la page affichée quand l'utilisateur ouvre l'application depuis le raccourci sur l'écran d'accueil du téléphone.

```
{
  "start_url": "/pwa"
}
```

Pour des besoins de statistiques, il est possible d'ajouter un *query parameter* à l'URL afin de déterminer la provenance des utilisateurs dans les outils d'*analytics*.

```
{
  "start_url": "/pwa/?from=pwa"
}
```

– `icons` : jeu d'icônes notamment utilisées lorsque l'application est installée sur l'écran d'accueil. Nous pouvons définir différentes tailles. Le navigateur se charge de choisir la bonne image en fonction de l'emplacement où il souhaite l'utiliser. Il est conseillé de proposer au moins deux icônes aux dimensions 192 x 192 et 512 x 512.

```
{
  "icons": [
   {
    "src": "/images/icons-192.png",
    "type": "image/png",
    "sizes": "192x192"
   },
   {
    "src": "/images/icons-512.png",
    "type": "image/png",
    "sizes": "512x512"
   }
  ]
}
```

Depuis sa version 8, pour répondre aux besoins des constructeurs de téléphones, Android a changé sa façon de travailler avec les icônes en introduisant les *adaptive icons*. Ce nouveau format oblige les icônes de toutes les applications à avoir la même forme. Une zone blanche est ajoutée autour de l'icône si elle ne respecte pas cette norme. Et inversement, si l'image est plus grande que la forme voulue, Android fait la découpe lui-même. Ladite forme n'est pas définie par Android lui-même, mais plutôt par les fabricants de téléphones. C'est pour cela que nous avons des icônes différentes sur Pixel, LG ou encore Samsung.

Ce changement sur Android impacte la façon d'afficher les icônes des Progressive Web Apps. En effet, ces dernières se retrouvaient entourées d'un cercle blanc pas très appréciable à l'œil. Pour remédier à ce problème, la plateforme web a mis en place les *maskable icons*. Deux contraintes doivent être respectées pour créer des maskable icons :

– S'assurer que la partie importante de l'icône se trouve dans une zone correspondant à un cercle occupant 80 % de la taille totale de l'image. Le reste de l'image est éventuellement supprimé par Android en fonction de la forme d'image voulue par le fabricant.

– Ajouter la propriété `purpose` avec la valeur `maskable` lors de la définition de l'icône dans le fichier Manifest.

```
{
  "icons": [
   {
    "src": "/images/icons-192.png",
    "type": "image/png",
    "sizes": "192x192",
  "purpose": "maskable"
   },
   {
    "src": "/images/icons-512.png",
    "type": "image/png",
    "sizes": "512x512",
  "purpose": "maskable"
   }
  ]
}
```

`background_color` : couleur utilisée comme couleur de fond du *splashscreen* (écran d'accueil) lorsque l'application est lancée depuis l'écran d'accueil du téléphone. Cette propriété accepte comme valeur un code couleur au format hexadécimal.

```
{
  "background_color": "#3367D6"
}
```

Il est nécessaire d'avoir un splashscreen, car même si l'application est utilisable hors connexion – les ressources statiques sont donc déjà présentes sur l'appareil –, il faut toujours un temps nécessaire pour que le navigateur puisse lire les différentes ressources et ensuite les exécuter.

`display` : mode d'affichage utilisé lorsque l'application est lancée depuis l'écran d'accueil. Quatre valeurs sont possibles pour cette propriété :

– `browser` : reproduit la même expérience utilisateur qu'au sein d'un navigateur (barre d'URL par exemple).
– `fullscreen` : prend toute la taille de l'écran.
– `standalone` : prend la même taille qu'une application native.

– `minimal-ui` : comme le mode `standalone`, mais ajoute des éléments permettant de faciliter la navigation des utilisateurs (bouton **Retour** par exemple).

```
{
  "display": "standalone"
}
```

`orientation` : propriété qui force l'orientation de l'application. Deux valeurs sont possibles :

– `portrait` : format portrait.

– `landscape` : format paysage.

```
{
  "orientation": "portrait"
}
```

`theme_color` : couleur utilisée par le navigateur pour styliser certains de ses éléments natifs (barre d'adresse par exemple). Cette propriété accepte comme valeur un code couleur au format hexadécimal.

```
{
  "theme_color": "#3367D6"
}
```

Les propriétés que nous venons de présenter sont celles qu'il faut a minima définir pour l'application. D'autres propriétés sont utilisables :

`categories` : tableau permettant de définir des catégories auxquelles l'application appartient. Ces informations seront utilisées dans le futur par les stores.

```
{
  "categories": ["web", "blog"]
}
```

`dir` : propriété permettant d'indiquer le sens de lecture de certaines propriétés du fichier Manifest (`name`, `short_name` et `description`). Cette propriété peut prendre deux valeurs :

– `ltr` : de gauche à droite.

– `Rtl` : de droite à gauche.

```
{
  "dir": "ltr"
}
```

Pour des raisons d'accessibilité, il est recommandé de définir cette propriété.

`iarc_rating_id` : propriété permettant d'indiquer l'identifiant de l'application retourné par l'IARC (*International Age Rating Coalition* - organisme indépendant qui audite une application et indique l'âge minimal requis pour pouvoir l'utiliser). Des applications comme les stores peuvent utiliser cette information pour récupérer cet âge minimal auprès de cet organisme et afficher un logo approprié.

```
{
  "iarc_rating_id": "e84b072d-71b3-4d3e-86ae-31a8ce4e53b7"
}
```

`lang` : propriété permettant d'indiquer la langue de certaines propriétés du fichier Manifest (`name`, `short_name` et `description`). La valeur que nous devons définir doit respecter la liste BCP47 standardisée par l'IETF (*Internet Engineering Task Force*). Dans l'exemple ci-après, nous définissons le français comme langue pour les propriétés citées précédemment.

```
{
  "lang": "fr"
}
```

Pour des raisons d'accessibilité, il est recommandé de définir cette propriété. Si la valeur n'est pas correcte, les synthétiseurs vocaux lisent le contenu écrit dans un certain langage avec éventuellement un accent totalement différent. Cela rend la compréhension très compliquée.