Progressive Web App (PWA)

Vous souhaitez développer une application dédiée aux mobiles et vous hésitez encore entre une application web et une application native …

Incontestablement, chacune de ces solutions offrent avantages et inconvénient :

L'**application native** est développée spécifiquement pour un OS, de ce fait :

- Elle permet d'exploiter les fonctionnalités natives du terminal (gyroscope, GPS, caméra, …)
- Elle peut être utilisée sans connexion internet
- Elle engendre des couts supplémentaires de développement (autant d'application que d'OS) et de maintenance

L'**application web** est un site web responsive, de ce fait :

- Un seul développement suffit, les couts de développement et de maintenance sont donc minimisés.
- Elle est responsive et s'adapte donc, automatiquement, à toutes les tailles d'écran
- Pas besoin de passer par les stores pour la distribuer.
- Il n'est pas possible de gérer les notifications push

Une alternative semble pouvoir tirer avantages de ces deux solutions :

Depuis 2015, une nouvelle approche est proposée : **les 'Progressive Web Apps' (PWA).**

Il s'agit d'une web app améliorée qui utilise une technologie avancée du web (une combinaison d'*application shell* et de de futurs standards comme les *Service Workers*) et permet d'offrir, à l'utilisateur, une application pour smartphone beaucoup plus immersive qu'une simple web app.

Les avantages sont considérables :

- Progressives : elles sont construites avec l'amélioration progressive comme un principe de base et fonctionnent pour chaque utilisateur, quel que soit le choix du navigateur ;
- Responsives : s'adaptent à tout facteur de forme à savoir bureau, mobile, tablette, etc. ;
- Indépendantes de la connectivité : capables de fonctionner hors ligne ou sur les réseaux de faible qualité grâce aux service workers ;
- Sûres : services via HTTPS pour prévenir l'espionnage et assurer que le contenu n'a pas été altéré ;
- Elles peuvent être partagées facilement via une URL ;
- Elles disposent d'une icône qui peut être affichée sur l'écran d'accueil.

Une liste d'avantages qui relance vraiment le débat sur l'utilité de développer une application purement native. D'autant que, si on regarde de plus près, un utilisateur sollicite, en moyenne, sur un mois, 25 applications mobiles et consulte plus d'une centaine de sites web …

Seul ombre au tableau, cette approche est encore un peu 'verte', actuellement, tous les navigateurs ne prennent pas en charge ces technologies même si Chrome, Firefox et Opera intègrent déjà ces futurs standards.

Zoom sur les Service Workers :

"*il vaut mieux le résultat d'hier instantanément qu'attendre trop longtemps celui d'aujourd'hui.*" (SFEIR Mag)

Véritable chef d'orchestre, cette technologie est indispensable pour votre PWA.

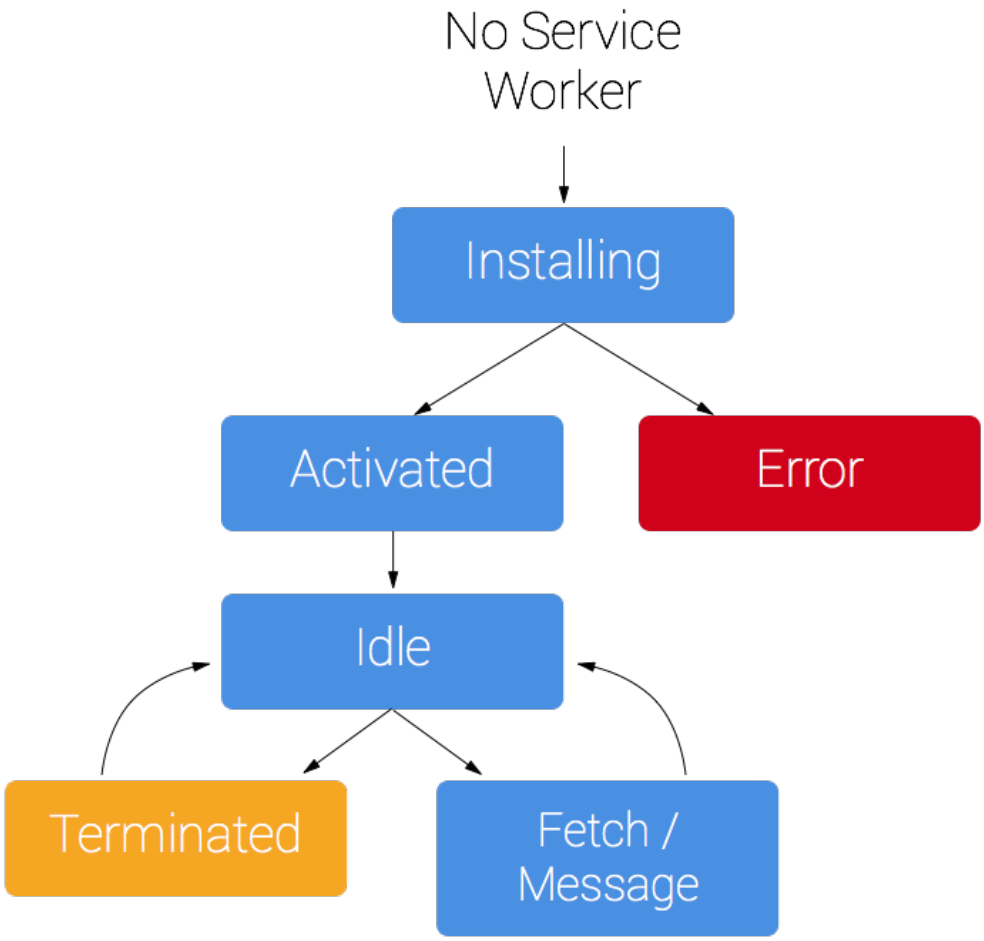
Il s'agit d'un script qui est chargé en même temps que votre page et qui s'exécute en dehors du contexte de votre page web.

Un service worker est un quelque sorte un proxy cache côté client : il va intercepter toutes vos requêtes serveur et y répondre soit en utilisant les données en provenance du réseau soit en utilisant un cache dédié. Il rend donc l'application utilisable offline.

Ce principe permet également de pallier aux problèmes de connectivité : perte ou faible connectivité.

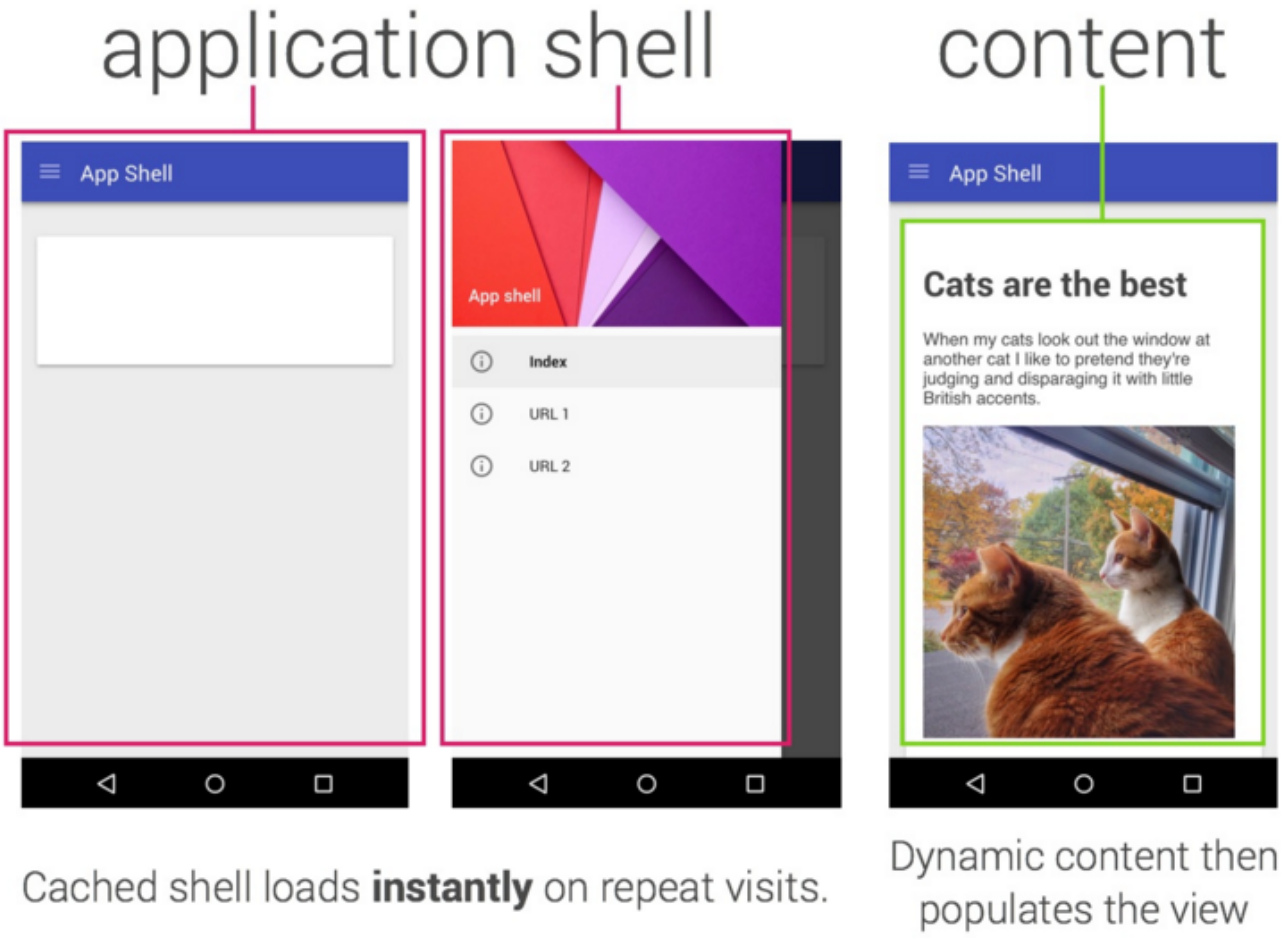
Concrètement, un service worker possède un cycle de vie indépendant de votre page web :

- Pour installer un service worker il doit être enregistré dans le script de votre application web (**register**)
- Une fois installé, le Service Worker va s'executer dans un thread (**activate**)
- Après activation, le service peut intercepter les évènements **fetch** et **message** émis respectivement par une requete serveur ou un appel via l' API postMessage



Zoom sur l’architecture application shell :

L’objectif est de mettre en cache la « coque » de votre application hors ligne et d’en remplir son contenu, en utilisant JS, quand la coque est chargée (gain en rapidité) :



Quelques liens sur le web :

- Un blog très bien réalisé sur le sujet
- La page Google dédiée au PWA