**Service Worker**

Ein Service Worker ist eine W3C-Standard-Webtechnik bei der JavaScript-Code im Hintergrund von Web-Browsern ausgeführt wird. Mit Hilfe von Service Worker ist es möglich, essentielle Funktionalitäten wie Caching zur Offline-Verwendbarkeit (z.B. bei Ausfall der Internetverbindung) von Web-Anwendungen, Aktualisierungen von Inhalten im Hintergrund, aber auch die von nativen Apps bekannten Push-Benachrichtigungen(Push-Notifications) zu ermöglichen. Dies findet alles im Hintergrund des Browsers statt und macht somit eine Installation von Software oder Software-Diensten unnötig.

Der Service Worker kann als Proxy fungieren und zum anderen vom Server gesendete Benachrichtigungen, selbst dann empfangen, wenn gerade keine Web-Page der entsprechenden Domain / Web-App geöffnet ist.

**Eigenschaften eines Service Worker:**

Ein Service-Worker ist eine Art von Web-Worker.

Der Service-Worker wird auf dem registrierenden Service-Worker-Client ausgeführt.

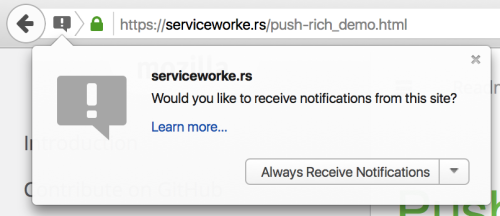
Ein Service-Worker mehrere Zustände analysiert, installiert, aktiviert um zu sehen, in welchem Zustand er sich befindet, weiterhin besitzt ein Service-Worker eine eigene Skript url (eine URL) und eine Service-Worker-Registrierung, die sich selbst enthält. Service Worker bedingen HTTPS, weshalb auch jede Progressive Web App mit HTTPS läuft.

**Push API / Web-Push**

Bei Web Push handelt es sich um eine Erweiterung des bekannten Service-Worker-Standards. Solange der Browser geöffnet ist, können Benachrichtigungen von Webseiten empfangen werden, selbst wenn der eigentliche Tab nicht geöffnet ist. So kann man E-Mail-Tab schließen und trotzdem über eingehende Mails informiert werden. Da keine zusätzlichen Apps oder Text-Nachrichten für direkte Notifications nötig sind, ergibt sich ein großer Vorteil für Speichernutzung, Performance und Akkulaufzeit von Mobilgeräten.

Web Push benötigt genauso wie die Standortfreigabe oder der Kamerazugriff eine (jederzeit wiederrufbare) Berechtigung, bevor eine Webseite Notifications anzeigen kann.

* Hier kommt unser Bild rein

[](https://entwickler.de/wp-content/uploads/2016/01/Screen-Shot-2016-01-25-at-00.20.13-500x216.png)

Durch eine ständige Verbindung zu einem Push Service in unserem Fall „Firebase Cloud Messaging“ , der als zentrale Schaltstelle für Nachrichten fungiert, werden Web-Push-Benachrichtigungen ermöglicht. Ursprünglich betrieb jeder Browser-Anbieter einen eigenen Push-Service zum Schutz der Privatsphäre. Erst kürzlich wurden aber GCM (Google Cloud Messaging Push Service von Google) und Firebase (Mozilla Firefox Push Service) zu Firebase Cloud Messaging zusammengelegt.

So erhält jede Webseite einen anderen, anonymen Web Push Identifier zur Verhinderung von seitenübergreifenden Zuordnungen. Zudem verschlüsselt der Browser die Nutzerdaten über ein Public-Key-Verfahren und meldet sich auch nur beim Push-Dienst an, wenn der User das Push-Verfahren erlaubt hat.

**Push API**

Die Push-API bietet Webanwendungen die Möglichkeit, von einem Server gesendete Nachrichten zu empfangen, unabhängig davon, ob die Webanwendung im Vordergrund oder sogar aktuell geladen ist.

Damit eine App, Push-Nachrichten empfangen kann, muss sie einen aktiven Service-Worker haben. Wenn der Service-Worker aktiv ist, kann er Push-Benachrichtigungen über PushManager.subscribe () abonnieren.  
  
Die resultierende PushSubscription enthält alle Informationen, die die Anwendung benötigt, um eine Push-Nachricht zu senden: einen Endpoint und den für das Senden von Daten erforderlichen Verschlüsselungsschlüssel.  
  
Der Service-Worker wird nach Bedarf gestartet, um eingehende Push-Nachrichten zu behandeln, die an den ServiceWorkerGlobalScope.onpush- Eventhandler übergeben werden. Dies ermöglicht es Apps, auf Nachrichten zu reagieren, die empfangen werden, beispielsweise durch Anzeigen einer Benachrichtigung (mit ServiceWorkerRegistration.showNotification ())…  
  
Jedes Abonnement für einen Service-Worker ist eindeutig. Der Endpoint für das Abonnement ist eine eindeutige URL. Die Kenntnis des Endpoints ist alles, was notwendig ist, um eine Nachricht an die Anwendung zu senden. Die Endpoint-URL muss daher geheim gehalten werden, oder andere Anwendungen könnten Push-Nachrichten an die Anwendung senden.  
  
Das Aktivieren eines Service-Workers, um eine Push-Nachricht zu liefern, kann zu einer erhöhten Ressourcennutzung, insbesondere der Batterie, führen. Verschiedene Browser haben unterschiedliche Schemata für die Handhabung - es gibt derzeit keinen Standardmechanismus. Firefox ermöglicht es, eine begrenzte Anzahl von Push-Nachrichten an eine Anwendung zu senden, obwohl Push-Nachrichten, die Benachrichtigungen generieren, von dieser Grenze ausgenommen sind. Die Grenze wird jedes Mal aktualisiert, wenn die Website besucht wird. Im Vergleich dazu gibt Chrome kein Limit an, erfordert aber, dass jede Push-Nachricht eine Benachrichtigung ausgibt.

Funktionsweise

1. Anfordern von Berechtigungen für Webbenachrichtigungen oder sonstige verwendete Berechtigungen.
2. Registrieren eines Service-Workers, um die Seite zu steuern.
3. Abonnieren des Push-Messaging-Service mit PushManager.subscribe ().
4. Aufrufen des mit der Subskription verknüpften Endpunktes und generieren eines öffentlichen Clientschlüssel (PushSubscription.endpoint und PushSubscription.getKey ().
   * getKey () derzeit nur experimentell für Firefox.
5. Senden der Details an den Server, so dass er Push-Nachricht senden kann, wenn erforderlich.
6. Der Key kann aus dem Endpoint kann mittels des Befehls Fetch abgetrennt werden. Und dann gespeichert werden (da wir Google Chrome Nutzen 🡪 und nicht nur Firefox)
7. Auf der Serverseite wird der Endpunkt und alle anderen erforderlichen Details wie die Sender ID und Geräte ID gespeichert, so dass sie verfügbar sind, wenn eine Push-Nachricht an einen Push-Absender gesendet werden muss.
8. Um eine Push-Nachricht zu senden, muss ein HTTP-POST an die Endpoint-URL gesendet werden. Die Anforderung muss einen TTL-Header enthalten, der begrenzt, wie lange die Nachricht in der Warteschlange stehen soll, wenn der Benutzer nicht online ist. Um Nutzdaten in Ihre Anfrage einzubinden, müssen Sie verschlüsselt werden (mit dem öffentlichen Schlüssel des Clients (public key))

Payload Verschlüsselung