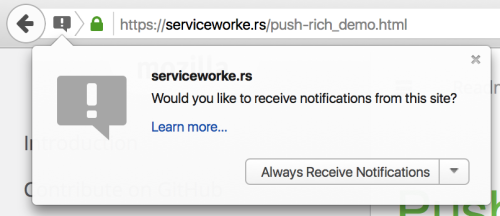
**Web Push**

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.

Web Push benötigt genauso wie die Standortfreigabe oder der Kamerazugriff eine (jederzeit wiederrufbare) Berechtigung, bevor eine Webseite Notifications ausspielen 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, der als zentrale Schaltstelle für Nachrichten fungiert, werden Web-Push-Benachrichtigungen ermöglicht. Jeder Browser-Anbieter betreibt einen eigenen Push-Service zum Schutz der Privatsphäre. 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.

**Service Worker**

Ein Service Worker ist ein JavaScript, das von Web-Browsern im Hintergrund ausgeführt wird. Es stellt die für eine Progressive Web App essentiellen Funktionalitäten wie Caching zur Offline-Verwendbarkeit zur Verfügung. Einmal online aufgerufen, können die Inhalte beim nächsten Besuch der Seite auch dann angezeigt werden, wenn eine schlechte oder sogar gar keine Internetverbindung besteht (Offline-Betrieb). Auch die von nativen Apps bekannten Push-Benachrichtigungen sind mit Service Workern möglich. Service Worker müssen eigens programmiert, im JavaScript der Seite registriert und anschließend installiert werden. Service Worker bedingen HTTPS, weshalb auch jede Progressive Web App mit HTTPS läuft.

**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 auf einem Benutzeragenten geladen ist. Dies ermöglicht Entwicklern asynchrone Benachrichtigungen und Updates für Benutzer, die sich entscheiden, was zu einer besseren Auseinandersetzung mit rechtzeitig neuen Inhalten.

Damit eine App Push-Nachrichten empfangen kann, muss sie einen aktiven Service-Mitarbeiter haben. Wenn der Service-Mitarbeiter 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 Endpunkt und den für das Senden von Daten erforderlichen Verschlüsselungsschlüssel.  
  
Der Service-Mitarbeiter 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 Endpunkt für das Abonnement ist eine eindeutige URL. Die Kenntnis des Endpunkts ist alles, was notwendig ist, um eine Nachricht an Ihre Anwendung zu senden. Die Endpunkt-URL muss daher geheim gehalten werden, oder andere Anwendungen könnten Push-Nachrichten an Ihre Anwendung senden.  
  
Das Aktivieren eines Service-Arbeiters, 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))