Web Notification Improvements in Chrome 50: Icons, Close Events, Renotify Preferences and Timestamps

By Paul Kinlan

Paul is a Developer Advocate

Push notifications allow you to provide a great app-like experience for your users, alerting them of important and timely updates like incoming chat messages. The notification platform is relatively new in browsers and as more and more use cases and requirements are fleshed out, we are seeing many additions to the APIs for notifications. Chrome 50 (beta in March 2016) is no exception, with no fewer than four new features that give developers more control over notifications. You get the ability to:

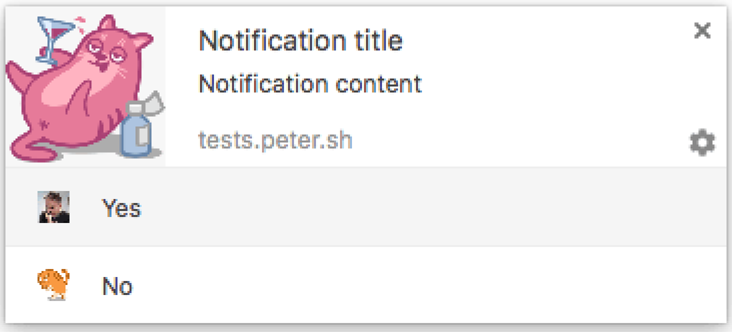
- add icons to notification buttons,
- modify the timestamp to help create a consistent experience,
- track notification close events to help synchronize notifications and provide analytics,
- manage the renotify experience when a notification replaces the currently displayed notification.

Chrome 50 has also added Payloads for Push notifications. To stay up to date with the Notifications API as it's implemented in Chrome, follow the spec and the spec issue tracker.

Create compelling action buttons with custom icons

In a recent post about notification action buttons in Chrome 49, I mentioned that you can't attach images to notification buttons to make them snazzy and appealing, but you could use Unicode characters to inline emojis etc.. Now you don't have to worry: with this recent addition you can now specify an image on the action button:

```
self.registration.showNotification('New message from Alice', {
  actions: [
   {action: 'like', title: 'Like', icon: 'https://example/like.png'},
   {action: 'reply', title: 'Reply', icon: 'https://example/reply.png'}]
});
```



The action icon's appearance differs by platform. For example, on Android the icon will have a dark grey filter applied in Lollipop and above, and a white filter pre-Lollipop, while on desktop it will be full colour. (Note: there is discussion about the future of this on desktop.) Some platforms might not even be able to display action icons, so ensure that you are using the icons to provide context to the action and not as the sole indicator of the intent.

And finally, because the resources must be downloaded, it is good practice to keep the icons as small as possible and to precache them in your install event. (A the time of this writing, fetches of notification resources in Chrome are not yet routed through the service worker.)

Notification close events

A frequently requested feature of notifications is the ability to know when the user has dismissed a notification. We had no way to do that until a recent set of changes to the notification specification added a notificationclose event.

By using the notificationclick and the notificationclose event you can understand how your users are interacting with your notifications. Are they leaving them open for a long time and then actively dismissing them or are they acting on them right away.

One popular use case is to be able to synchronize notifications between devices. If the user dismisses a notification on their desktop device, the same notification on their mobile device should also be dismissed. We don't yet have the ability to do this silently (remember every push message must have a notification displayed), but by using notificationclose it opens up the ability to handle this by allowing you to track the notification state for the user on your server and synchronize that with the other devices as the user uses them.

To use the notificationclose event, register it inside your service worker and it will fire only when the user has actively dismissed a notification, for example, if the user dismisses a specific notification or dismisses all the notifications in their tray (on Android).

If the requireInteraction flag is false or not set, then if the notification is not manually dismissed by the user, but instead automatically by the system, the notificationclose event will not be triggered.

A simple implementation is shown below. When the user dismisses the notification you get access to the notification object from which you can perform custom logic.

```
self.addEventListener('notificationclose', e => console.log(e.notification));
```

You can test this in the Notification Generator; you will get an alert when you close the notification.

Don't annoy your users when you replace an existing notification

I am pretty sure Uncle Ben was talking about the notification system and not the powers of Peter Parker when he said "With great power comes great responsibility". The notification system is a powerful medium for interacting with users. If you abuse their trust they will turn off all notifications and you may lose them entirely.

When you create a notification you can set it to create an audible alert or vibrate to get the attention of the user. Additionally, you can replace an existing notification by reusing its 'tag' attribute on a new notification object.

Prior to Chrome 50, every time you created a notification or replaced an existing one, it would run a vibration pattern or play an audible alert and this could cause frustration for your users. Now In Chrome 50, you now have control over what happens during the renotification via a simple boolean flag called 'renotify'. The new default behaviour when using the same 'tag' for subsequent notifications is to be silent and as the developer you must opt in to "re-notifying" the user by setting the flag to "true".

```
self.registration.showNotification('Oi!', {
  'renotify': true,
  'tag': 'tag-id-1'
});
```

You can try this out in the Notification Generator.

Manage the timestamp displayed to the user

On Android, Chrome's notifications show their create times in the top right corner by default. Unfortunately, this might not be the time that the notification was actually generated by your system. For example, the event might have been triggered when the device was offline, or the notification could be shown for an upcoming meeting. As of Chrome 50, Chrome has added a new 'timestamp' property that enables developers to provide the time that should be displayed in the notification.

```
self.registration.showNotification('Best day evar!', {
  'timestamp': 360370800000
});
```

The timestamp is currently only visible on Chrome for Android. Although it is not visible on desktop, it will affect the notification order on both mobile and desktop.

Note: Be sure to check out the full documentation including best practices for using Web Push Notifications