Notifying you of Changes to Notifications

By Matt Gaunt

Matt is a contributor to Web**Fundamentals**

Firstly, I apologize for that awful title, but I couldn't not.

In Chrome 44 Notfication.data and ServiceWorkerRegistration.getNotifications() are added and open up / simplify some common use cases when dealing with notifications with push messages.

Notification Data

Notification.data allows you to associate a JavaScript object with a Notification.

What this basically boils down to, is when you receive a push message, you can create a notification with some data, then in the notificationclick event you can get the notification that was clicked and get its data.

For example, creating a data object and adding it to your notification options like so:

```
self.addEventListener('push', function(event) {
  console.log('Received a push message', event);

  var title = 'Yay a message.';
  var body = 'We have received a push message.';
  var icon = '/images/icon-192x192.png';
  var tag = 'simple-push-demo-notification-tag';
  var data = {
    doge: {
        wow: 'such amaze notification data'
    }
  };

  event.waitUntil(
    self.registration.showNotification(title, {
      body: body,
      icon: icon,
      tag: tag,
      data: data
    })
```

```
  );
});
```

Means we can get the information in the notificationclick event:

```
self.addEventListener('notificationclick', function(event) {
  var doge = event.notification.data.doge;
  console.log(doge.wow);
});
```

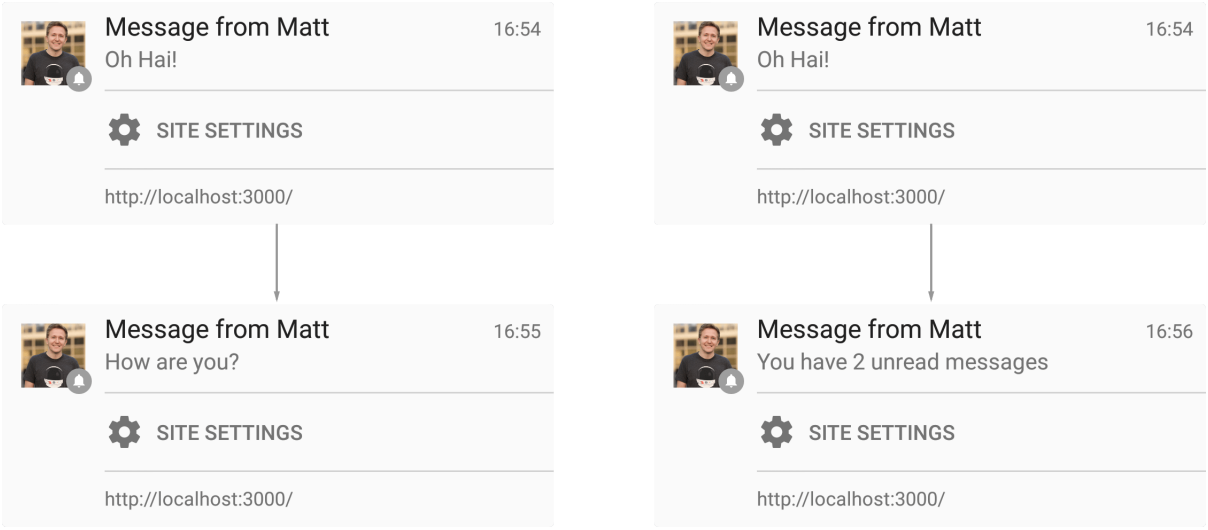
Before this, you had to stash data in IndexDB or put something on the end of the icon URL - eek.

ServiceWorkerRegistration.getNotifications()

One common request from developers working on push notifications is to have better control over the notifications that they display.

An example use case would be a chat application where a user sends multiple messages and the recipient displays multiple notifications. Ideally the web app would be able to notice you have several notifications which haven't been viewed and collapse them down into a single notification.

Without getNotifications() the best you can do is replace the previous notification with the latest message. With getNotifications(), you can "collapse" the notifications if a notification is already displayed - leading to a much better user experience.



The code to do this is relatively simple. Inside your push event, call ServiceWorkerRegistration.getNotifications() to get an array of current Notifications and from there decide the right behaviour, whether that's collapsing all notifications or by using the Notification.tag.

```
function showNotification(title, body, icon, data) {
  var notificationOptions = {
    body: body,
    icon: icon ? icon : 'images/touch/chrome-touch-icon-192x192.png',
    tag: 'simple-push-demo-notification',
    data: data
  };

  self.registration.showNotification(title, notificationOptions);
  return;
}

self.addEventListener('push', function(event) {
  console.log('Received a push message', event);

  // Since this is no payload data with the first version
  // of Push notifications, here we'll grab some data from
  // an API and use it to populate a notification
  event.waitUntil(
    fetch(API_ENDPOINT).then(function(response) {
      if (response.status !== 200) {
        console.log('Looks like there was a problem. Status Code: ' +
          response.status);
        // Throw an error so the promise is rejected and catch() is executed
        throw new Error();
      }

      // Examine the text in the response
      return response.json().then(function(data) {
        var title = 'You have a new message';
        var message = data.message;
        var icon = 'images/notification-icon.png';
        var notificationTag = 'chat-message';

        var notificationFilter = {
          tag: notificationTag
        };
        return self.registration.getNotifications(notificationFilter)
          .then(function(notifications) {
            if (notifications && notifications.length &gt; 0) {
              // Start with one to account for the new notification
              // we are adding
```

```
              var notificationCount = 1;
              for (var i = 0; i &lt; notifications.length; i++) {
                var existingNotification = notifications[i];
                if (existingNotification.data &&
                  existingNotification.data.notificationCount) {
                  notificationCount +=
existingNotification.data.notificationCount;
                } else {
                  notificationCount++;
                }
                existingNotification.close();
              }
              message = 'You have ' + notificationCount +
                ' weather updates.';
              notificationData.notificationCount = notificationCount;
            }

            return showNotification(title, message, icon, notificationData);
          });
      });
    }).catch(function(err) {
      console.error('Unable to retrieve data', err);

      var title = 'An error occurred';
      var message = 'We were unable to get the information for this ' +
        'push message';

      return showNotification(title, message);
    })
  );
});

self.addEventListener('notificationclick', function(event) {
  console.log('On notification click: ', event);

  if (Notification.prototype.hasOwnProperty('data')) {
    console.log('Using Data');
    var url = event.notification.data.url;
    event.waitUntil(clients.openWindow(url));
  } else {
    event.waitUntil(getIdb().get(KEY_VALUE_STORE_NAME,
event.notification.tag).then(function(url) {
      // At the moment you cannot open third party URL's, a simple trick
      // is to redirect to the desired URL from a URL on your domain
      var redirectUrl = '/redirect.html?redirect=' +
        url;
      return clients.openWindow(redirectUrl);
    }));
```

```
  }
});
```

The first thing to highlight with this code snippet is that we filter our notifications by passing a filter object to getNotifications(). This means we can get a list of notifications for a specific tag (in this example for a particular conversation).

```
var notificationFilter = {
  tag: notificationTag
};
return self.registration.getNotifications(notificationFilter)
```

Then we look over the notifications which are visible and check to see if there is a notification count associated with that notification and increment based on that. This way if there is one notification telling the user there are two unread messages, we would want to point out that there are three unread messages when a new push arrives.

```
var notificationCount = 1;
for (var i = 0; i &lt; notifications.length; i++) {
  var existingNotification = notifications[i];
  if (existingNotification.data && existingNotification.data.notificationCount) {
    notificationCount += existingNotification.data.notificationCount;
  } else {
    notificationCount++;
  }
  existingNotification.close();
}
```

A subtlety to highlight is that you need to call close() on the notification to ensure the notification is removed from the notification list. This is a bug in Chrome since each notification is replaced by the next one because the same tag is used. At the moment this replacement isn't being reflected in returned array from getNotifications().

This is only one example of getNotifications() and as you can imagine, this API opens up a range of other use cases.

NotificationOptions.vibrate

As of Chrome 45 you can specify a vibration pattern when creating a notification. On devices that support the Vibration API - currently only Chrome for Android - this allows you to customize the vibration pattern that will be used when the notification is displayed.

A vibration pattern can either be an array of numbers, or a single number which is treated as an array of one number. The values in the array represent times in milliseconds, with the even indices (0, 2, 4, ...) being how long to vibrate, and the odd indices being how long to pause before the next vibration.

```
self.registration.showNotification('Buzz!', {
  body: 'Bzzz bzzzz',
  vibrate: [300, 100, 400] // Vibrate 300ms, pause 100ms, then vibrate 400ms
});
```

Remaining Common Feature Requests

The one remaining common feature request from developers is the ability to close a notification after a certain time period or the ability to send a push notification with the purpose of just closing a notification if it's visible.

At the moment there isn't a way you can do this and nothing in the spec that will allow it :(but the Chrome engineering team are aware of this use case.

Android Notifications

On desktop you can create a notification with the following code:

```
new Notification('Hello', {body: 'Yay!'});
```

This was never supported on Android due to restrictions of the platform: specifically, Chrome can't support the callbacks on the Notification object, such as onclick. But it's used on desktop for displaying notifications for web apps you may currently have open.

The only reason I mention it is that originally, a simple feature detection like the one below would help you support desktop and not cause any errors on Android:

```
if (!'Notification' in window) {
  // Notifications aren't supported
  return;
}
```

However, with push notification support now on Chrome for Android, notifications can be created from a ServiceWorker, but not from a web page, meaning this feature detect is no

longer appropriate. If you try and create a notification on Chrome for Android you'll receive this error message:

Uncaught TypeError: Failed to construct 'Notification': Illegal constructor. Use ServiceWorkerRegistration.showNotification() instead

The best way to feature detect for Android and desktop at the moment is to do the following:

```
function isNewNotificationSupported() {
    if (!window.Notification || !Notification.requestPermission)
        return false;
    if (Notification.permission == 'granted')
        throw new Error('You must only call this \*before\* calling
Notification.requestPermission(), otherwise this feature detect would bug the
user with an actual notification!');
    try {
        new Notification('');
    } catch (e) {
        if (e.name == 'TypeError')
            return false;
    }
    return true;
}
```

This can be used like so:

```
if (window.Notification && Notification.permission == 'granted') {
    // We would only have prompted the user for permission if new
    // Notification was supported (see below), so assume it is supported.
    doStuffThatUsesNewNotification();
} else if (isNewNotificationSupported()) {
    // new Notification is supported, so prompt the user for permission.
    showOptInUIForNotifications();
}
```

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

Last updated July 2, 2018.