Using the Notifications API

- Introduction
- Step 1: Check for Notifications API support
- Step 2: Let the user grant permissions to a website to show notifications
- Step 3: Attach listeners and other actions

This article is outdated and references a deprecated API.

For modern coverage of the Notifications API (which is supported by Chrome, Firefox and Safari), read the MDN docs on Notification. Notify.js also offers a nice abstraction for the API.

Introduction

The Notifications API allows you to display notifications to the user for given events, both passively (new emails, tweets or calendar events) and on user interactions regardless of which tab has focus. There is draft spec but it is not currently in any standard.

You can follow these simple steps to implement notifications in just a few minutes:

Step 1: Check for Notifications API support

We check if `webkitNotifications` is supported. Note that the name of `webkitNotifications` is because it's part of a draft spec. The final spec will have a notifications() function instead.

```
// check for notifications support
// you can omit the 'window' keyword
if (window.webkitNotifications) {
```

```
console.log("Notifications are supported!");
}
else {
console.log("Notifications are not supported for this Browser/OS version yet.");
}
```

Step 2: Let the user grant permissions to a website to show notifications

Any of the constructors we mentioned will throw a security error if the user hasn't manually granted permissions to the website to show notifications. To handle the exception you can use a try-catch statement but you can also use the `checkPermission` method for the same purpose.

```
document.querySelector('#show_button').addEventListener('click', function() {
if (window.webkitNotifications.checkPermission() == 0) { // 0 is PERMISSION_ALLOWED
// function defined in step 2
window.webkitNotifications.createNotification(
    'icon.png', 'Notification Title', 'Notification content...');
} else {
window.webkitNotifications.requestPermission();
}
}, false);
```

If the web application doesn't have permissions to show notifications then the `requestPermission` method will show an infobar:

The notifications permission infobar in Google Chrome

The notifications permission infobar in Google Chrome.

However, it's **very important** to remember that the `requestPermission` method only works in event handlers triggered by a user action, like mouse or keyboard events, in order to avoid unsolicited infobars. In this case, the user action is the click on the button with id "show_button". The snippet above will never work if the user hasn't explicitly clicked on a button or link that triggers the `requestPermission` at some point.

Step 3: Attach listeners and other actions

```
document.querySelector('#show_button').addEventListener('click', function() {
  if (window.webkitNotifications.checkPermission() == 0) { // 0 is PERMISSION_ALLOWED
    // function defined in step 2
    notification_test = window.webkitNotifications.createNotification(
      'icon.png', 'Notification Title', 'Notification content...');
    notification_test.ondisplay = function() { ... do something ... };
    notification_test.onclose = function() { ... do something else ... };
    notification_test.show();
  } else {
    window.webkitNotifications.requestPermission();
  }
}, false);
```

At this point, you might want to encapsulate all these events and actions creating your own Notification class to keep the code cleaner, although this is beyond the scope of this tutorial.

□□□ #3
□ □□□ □□□ 2026-10-31 22:06:19 UTC
□ cocobook-sync □□□ 2026-11-04 17:37:39 UTC