New in Chrome 67

By Pete LePage
Pete is a Developer Advocate

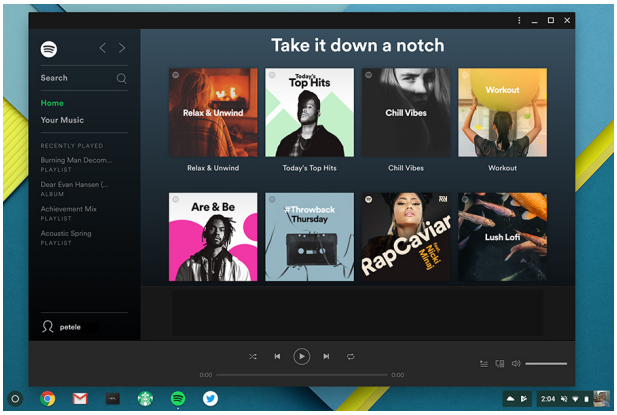
- Progressive Web Apps are coming to the desktop
- The generic sensor API makes it way easier to get access to device sensors like the accelerometer, gyroscope and more.
- And `BigInts` make dealing with big integers way easier.

And there's plenty more!

I'm Pete LePage. Let's dive in and see what's new for developers in Chrome 67!

Note: Want the full list of changes? Check out the Chromium source repository change list.

Desktop PWAs



Desktop Progressive Web Apps are now supported on Chrome OS 67, and we've already started working on support for Mac and Windows. Once installed, they're launched in the same way as other apps, and run in an app window, without an address bar or tabs. Service workers ensure that they're fast, and reliably, the app window experience makes them feel integrated. And they create an engaging experience for your users.

Getting started isn't any different than what you're already doing today. **All of the work you've done for your existing Progressive Web App still applies**, you simply need to consider some additional break points.

If your app meets the standard PWA criteria, Chrome will fire the `beforeinstallprompt` event, but it won't automatically prompt the user. Instead, save the event; then, add some UI - like an install app button - to your app to tell the user your app can be installed. Then, when the user clicks the button, call prompt on the saved event; Chrome will then show the prompt to the user. If they click add, Chrome will add your PWA to their shelf and launcher.

Check out my Google I/O talk where Jenny and I go into detail about the technical and special design considerations you need to think about when building a desktop progressive web app.

And, if you want to start playing with this on Mac or Windows - check out the full Desktop Progressive Web App post for details on how to enable support with a flag.

Generic Sensor API

Sensor data is used in many apps to enable experiences like immersive gaming, fitness tracking, and augmented or virtual reality. This data is now available to web app using the Generic Sensor API.

The API consists of a base Sensor interface with a set of concrete sensor classes built on top. Having a base interface simplifies the implementation and specification process for the concrete sensor classes. For example, the Gyroscope class is super tiny!

```
const sensor = new Gyroscope({frequency: 500});
sensor.start();

sensor.onreading = () => {
    console.log("X-axis " + sensor.x);
    console.log("Y-axis " + sensor.y);
    console.log("Z-axis " + sensor.z);
};
```

The core functionality is specified by the base interface, and Gyroscope merely extends it with three attributes representing angular velocity. Chrome 67 supports the accelerometer, gyroscope, orientation sensor, and motion sensor.

Intel has put together several demos of the generic sensors API and sample code, and they've also updated the Sensors for the Web! post from September with everything you need to know.

`BigInt`s

`BigInt`s are a new numeric primitive in JavaScript that can represent integers with arbitrary precision. Large integer IDs and high-accuracy timestamps can't be safely represented as **`Number`s** in JavaScript, which often leads to real-world bugs (because of which we often end up representing such numbers as strings instead).

```
let max = Number.MAX_SAFE_INTEGER;
// → 9_007_199_254_740_991
max = max + 1;
// → 9_007_199_254_740_992 - Yay!
max = max + 1;
// → 9_007_199_254_740_992 - Uh, no?
```

With **BigInts**, we can safely store and perform integer arithmetic without overflowing. Today, dealing with large integers typically means we have to resort to a library that would emulate **BigInt**-like functionality.

```
let max = BigInt(Number.MAX_SAFE_INTEGER);
// → 9_007_199_254_740_991n
max = max + 9n;
// → 9_007_199_254_741_000n - Yay!
```

When **BigInt** becomes widely available, we'll be able to drop these run-time dependencies in favor of native **BigInts**. Not only is the native implementation faster, it'll help to reduce load time, parse time, and compile time because we won't have to load those extra libraries.

And more!

These are just a few of the changes in Chrome 67 for developers, of course, there's plenty more.

The Credential Management API has been supported since Chrome 51, and provides a framework for creating, retrieving and storing credentials. It did this through two credential types: **PasswordCredential** and **FederatedCredential**. The Web Authentication API adds a third credential type, **PublicKeyCredential**, which allows browsers to authenticate a user with a private/public key pair generated by an authenticator such as a security key, fingerprint reader, or any other device that can authenticate a user. Chrome 67 enables the API using U2F/CTAP 1 authenticators over USB transport on desktop.

Learn more about it in Eiji's Enabling Strong Authentication with WebAuthn post.

Google I/O is a wrap

If you didn't make it to I/O, or may you did, but didn't see all the web talks, check out the Chrome and Web playlist to get caught up on all the latest from Google I/O!

New in DevTools

Be sure to check out New in Chrome DevTools, to learn what's new in for DevTools in Chrome 67.

Subscribe

Then, click the subscribe button on our YouTube channel, and you'll get an email notification whenever we launch a new video, or add our RSS feed to your feed reader.

I'm Pete LePage, and as soon as Chrome 68 is released, I'll be right here to tell you – what's new in Chrome!

Last updated July 2, 2018.