New In Chrome 55

By Pete LePage
Pete is a Developer Advocate

Watch on YouTube

- `async` and `await` allows you to write promise-based code as if it were synchronous, but without blocking the main thread.
- Pointer events provide a unified way of handling all input events.
- Sites added to the home screen, are automatically granted the persistent storage permission.

And there's plenty more.

I'm Pete LePage, here's what's new for developers in Chrome 55!

Pointer Events

Pointing at things on the web used to be simple. You had a mouse, you moved it around, sometimes you pushed buttons, and that was it. But this, doesn't work so well on here.

Touch events are good, but to support touch and mouse, you had to support two event models:

```
elem.addEventListener('mousemove', mouseMoveEvent);
elem.addEventListener('touchmove', touchMoveEvent);
```

Chrome now enables unified input handling by dispatching PointerEvents:

```
elem.addEventListener('pointermove', pointerMoveEvent);
```

Pointer events unify the pointer input model for the browser, bringing touch, pens, and mice together into a single set of events. They're supported in IE11, Edge, Chrome, Opera and partially supported in Firefox.

Check out Pointing the Way Forward for more details.

`async` and `await`

Asynchronous JavaScript can be difficult to reason about. Take this function with all it's "lovely" callbacks:

```
function logFetch(url) {
  var xhr = new XMLHttpRequest();
  xhr.onreadystatechange = function (e) {
    if (xhr.readyState === 4) {
      if (xhr.status === 200) {
        console.log(xhr.responseText);
      } else {
        console.error('xhr failed', xhr.statusText);
      }
    }
  };
  xhr.onerror = function (e) {
    console.error(xhr.statusText);
  };
  xhr.open('GET', url);
  xhr.send();
}
```

Re-writing it with `promises` helps avoid the nesting problem:

```
function logFetch(url) {
  return fetch(url)
```

```
    .then(response => response.text())
    .then(text => {
      console.log(text);
    }).catch(err => {
      console.error('fetch failed', err);
    });
}
```

But, Promise-based code can still be difficult to read when there are long chains of asynchronous dependencies.

Chrome now supports the **async** and **await** JavaScript keywords, allowing you to write Promise-based JavaScript that can be as structured and readable as synchronous code.

Instead, our asynchronous function can be simplified to this:

```
async function logFetch(url) {
  try {
    const response = await fetch(url);
    console.log(await response.text());
  }
  catch (err) {
    console.log('fetch failed', err);
  }
}
```

Jake has a great post: Async Functions - making promises friendly with all the details.

Persistent Storage

The persistent storage origin trial is now over. You can now mark web storage as persistent, preventing Chrome from automatically clearing the storage for your site.

```
if (navigator.storage && navigator.storage.persist) {
  navigator.storage.persist().then(granted => {
    if (granted)
      alert("Storage will not be cleared except by explicit user action");
    else
      alert("Storage may be cleared by the UA under storage pressure.");
  });
}
```

In addition, sites that have high engagement, have been added to the home screen or have push notifications enabled are automatically granted the persistence permission.

Check out Chris Wilson's Persistent Storage post for full details and how you can request persistent storage for your site.

CSS Automatic Hyphenation

CSS automatic hyphenation, one of Chrome's most frequently requested layout features is now supported on Android and Mac.

Web Share API

And finally, it's now easier to invoke native sharing capabilities with the new Web Share API, which is available as an origin trial. Paul (Mr. Web Intents) Kinlan has all the details in his Navigator Share post.

Closing

These are just a few of the changes in Chrome 55 for developers.

If you want to stay up to date with Chrome and know what's coming, be sure to subscribe, and be sure to check out the videos from the Chrome Dev Summit for a deeper dive into some of the awesome things the Chrome team is working on.

I'm Pete LePage, and as soon as Chrome 56 is released, I'll be right here to tell you what's new in Chrome!

Subscribe to Chrome Developers on YouTube

Subscribe to our YouTube channel or our RSS feed

Last updated July 2, 2018.