Promise.prototype.finally

By Mathias Bynens

V8 JavaScript whisperer

`Promise.prototype.finally` is enabled by default in V8 v6.3.165+ and Chrome 63+. It allows registering a callback to be invoked when a promise is settled (i.e. either fulfilled, or rejected).

Imagine you want to fetch some data to show on the page. Oh, and you want to show a loading spinner when the request starts, and hide it when the request completes. When something goes wrong, you show an error message instead.

```
const fetchAndDisplay = ({ url, element }) => {
  showLoadingSpinner();
  fetch(url)
    .then((response) => response.text())
    .then((text) => {
      element.textContent = text;
      hideLoadingSpinner();
    })
    .catch((error) => {
      element.textContent = error.message;
      hideLoadingSpinner();
    });
};

fetchAndDisplay({
  url: someUrl,
  element: document.querySelector('#output')
});
```

If the request succeeds, we display the data. If something goes wrong, we display an error message instead.

In either case we need to call `hideLoadingSpinner()`. Until now, we had no choice but to duplicate this call in both the `then()` and the `catch()` block. With `Promise.prototype.finally`, we can do better:

```
const fetchAndDisplay = ({ url, element }) => {
  showLoadingSpinner();
```

```
  fetch(url)
    .then((response) => response.text())
    .then((text) => {
      element.textContent = text;
    })
    .catch((error) => {
      element.textContent = error.message;
    })
    .finally(() => {
      hideLoadingSpinner();
    });
};
```

Not only does this reduce code duplication, it also separates the success/error handling phase and the cleanup phase more clearly. Neat!

Currently, the same thing is possible with **`async`/`await`**, and without **`Promise.prototype.finally`**:

```
const fetchAndDisplay = async ({ url, element }) => {
  showLoadingSpinner();
  try {
    const response = await fetch(url);
    const text = await response.text();
    element.textContent = text;
  } catch (error) {
    element.textContent = error.message;
  } finally {
    hideLoadingSpinner();
  }
};
```

Since async and **`await`** are strictly better, my recommendation remains to use them instead of vanilla promises. That said, if you prefer vanilla promises for some reason, **`Promise.prototype.finally`** can help make your code simpler and cleaner.

Last updated July 2, 2018.