Permissions API for the Web

By Matt Gaunt

Matt is a contributor to Web**Fundamentals**

If you've worked with the Geolocation API before, chances are you've wanted to check if you had permission to use Geolocation without causing a prompt. This simply wasn't possible. You had to request the current position and this would indicate the permission state or cause a prompt to be shown to the user.

Not all APIs work this way. The Notifications API has its own way of allowing you to check the current permission state via Notification.permission.

As the web platform grows in API's, there needs to be a single, standard way for developers to check the status of a permission rather than having to remember how each and every API works. The Permission API, available in Chrome version 43, is intended to be this single, standard way to check the permission status of an API.

permissions.query()

Check the status of a permission using the `permissions.query()` method. This will return a status of granted (you have permission), denied (you are blocked from accessing the API) or prompt (user needs to be prompted). For example:

```
// Check for Geolocation API permissions
navigator.permissions.query({name:'geolocation'})
  .then(function(permissionStatus) {
    console.log('geolocation permission state is ', permissionStatus.state);

    permissionStatus.onchange = function() {
      console.log('geolocation permission state has changed to ', this.state);
    };
  });
```

The query method takes a PermissionDescriptor object, where you define the permission's name. The response is a Promise resolving to a PermissionStatus object. From this object, you can check the state with `permissionStatus.state` for 'granted', 'denied' or 'prompt'. You can also also implement an event handler for `permissionStatus.onchange` and handle changes to the permission state.

Supported PermissionDescriptors

In the above example, we highlight how to query the permission state for geolocation with the following permission descriptor: `{name:'geolocation'}`.

The Notification permission descriptor is similar in that it only requires a name attribute: `{name:'notifications'}`.

Push and midi each have an additional parameter that is specific to that API.

For the push permission, you can supply a `userVisibleOnly` parameter. This indicates whether you wish to show a notification for every push message or be able to send silent push notifications (At the moment Chrome only supports push messages with notifications). You'd use it like so:

```
navigator.permissions.query({name:'push', userVisibleOnly:true})
```

Midi allows a `sysex` parameter. This indicates whether you need to and/or receive system exclusive messages. For midi this would be:

```
navigator.permissions.query({name:'midi', sysex:true})
```

Requesting Permissions

Requesting permission from the user depends on the specific API. For example, geolocation would show a permission prompt when you call `getCurrentPosition()`.

```
navigator.geolocation.getCurrentPosition(function(position) {
  console.log('Geolocation permissions granted');
  console.log('Latitude:' + position.coords.latitude);
  console.log('Longitude:' + position.coords.longitude);
});
```

Whereas notifications would prompt the user when you call `requestPermission()`.

```
Notification.requestPermission(function(result) {
  if (result === 'denied') {
    console.log('Permission wasn\'t granted. Allow a retry.');
    return;
  } else if (result === 'default') {
    console.log('The permission request was dismissed.');
    return;
  }
  console.log('Permission was granted for notifications');
});
```

The point here is that the Permission API allows a consistent way to monitor the status of permissions while being able to support the range of APIs currently on the web.

The big advantage of this is that it allows you to build better experiences for your users, only prompting when it's obvious to the user why you need extra privileges and taking full advantage of these APIs when you know you have been granted permission.

You can find a full set of examples here.

Browser Support

Chrome is the first browser to implement this, Mozilla are planning on shipping this, and Microsoft have shown interest in the API.

Known Issues

- Geolocation will not re-show a prompt if the user dismisses the permission request. The permission status however remains 'prompt'. [crbug.com/476509]