Asking the user for permission

There are times when you're using an API when it would be nice to know what's the permission status is for a given API.

The Permission API works around two concepts: permissions and "powerful features"

The spec defines a powerful feature as:

> A feature of a UA that some code might not be allowed to access, for example, because its environment settings object doesn't satisfy some criteria, or because the user hasn't given permission.
>
> Permissions

To describe the features, the API provides a registry for the features and their descriptors. Currently only Push and midi have additional parameters specific to those APIs.

The powerful features gated with this API are:

- accelerometer
- ambient-light-sensor
- background-fetch
- background-sync
- bluetooth
- camera
- clipboard-write
- device-info
- display-capture
- geolocation
- gyroscope
- magnetometer
- microphone
- midi
- nfc

- notifications
- persistent-storage
- push
- speaker-selection

Working with permissions

There are three basic things we can do with permissions, we can query them, request them, and revoke them.

For these examples, we'll use both push, a second parameter, and geolocation to write promise-based code that takes advantage of these powerful features.

Querying

Using `navigator.permissions.query` we can check individual permissions. It returns a promise that we can further process.

The following code will check the status of the permission and then log in to the console.

We also add an onchange event handler to report when the feature permission changes.

```
navigator.permissions.query({name: 'push', userVisibleOnly: 'true'})
  .then(function(permissionStatus) {
    console.log('push permission state is ', permissionStatus.state);

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

We can also check multiple permissions at the same time using promise.all and do something with each promise.

```
Promise.all([
  navigator.permissions.query({ name: "geolocation" }),
```

```
  navigator.permissions.query({name: 'push', userVisibleOnly: 'true'})
])
.then(([{ state: geoState }, { state: notifState }]) => {
  console.log("Geolocation permission state is:", geoState);
  console.log("Push permission state is:", notifState);
});
```

Code like this would let us know if we need to request permission from the user to use a feature.

requesting

Once we know if we need to ask permission, we need to ask for it. That's where `navigator.permissions.request` comes in. Using the same name that we did when we queried, we tell the browser to ask for permission to use the feature.

```
navigator.permissions.request({name: 'push', userVisibleOnly: 'true'})
  .then(function(permissionStatus) {
    console.log('push permission state is ', permissionStatus.state);

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

The promise will resolve with the status of the request based on user action.

We also add an onchange event handler for the permission status so we know what the user did without having to inspect the result of the promise.

revoking

We can also revoke permissions we've granted before using `navigator.permissions.revoke` to remove the permission we gave earlier.

As with the other methods, we log the result of the request to console and we set up an onchange event listener on permissionStatus to record any change to the permission.

```
navigator.permissions.revoke({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.sta
    };
  });
```

Coding defensively

Because Safari doesn't support the API, we have to code defensively and test if the feature is supported before we use it.

`supportPermissions()` tests if the navigator object has a permissions function available to it.

If the API is available then we can proceed, if not we need to communicate it to the user and, maybe, request permission anyways.

```
function supportsPermissions() {
  return ('permissions' in navigator);
}

if (supportsPermissions()) {
  console.log('we support the permissions API');
} else {
  console.log('we don\'t support the permissions API... Booo!');
}
```

Some final considerations

While we can use the permission API to work with geolocation and navigation, each of these APIs provides its own way to request the user's permission and it's important to know when to use each API.

Geolocation asks for permission when you call

`geolocation.getCurrentPosition` with code like this:

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

Notifications have their own `requestPermission` method that allows you to detect if permission was granted and take action on it.

```
Notification.requestPermission(function(result) {
  if (result === 'denied') {
    console.log('Permission request denied');
    return;
  } else if (result === 'default') {
    console.log('Permission request was dismissed.');
    return;
  }
  console.log('Permission was granted for notifications');
});
```

We can combine both techniques to only run these special permissions requests when permission has not been granted (when the permission status is not denied or granted) and after we explain to the user why we're asking for permission.