Extra Notes - Web Development - Location

- Dr Nick Hayward

Basic Geolocation A basic example of using the Geolocation API with HTML and jQuery.

Contents

- Intro
- Setup
- Geolocation check
- Geolocation output
- update UX

intro Further details on the Geolocation API can be found on the MDN site

- MDN Geolocation API - https://developer.mozilla.org/en-US/docs/Web/API/Geolocation/Using_geolocation

setup Create a basic app `index.html` page, and then add the required JS files

- public/js/index.js - app logic and code
- public/utils/jquery.js - JS library for jQuery

geolocation check Add a button, for example, to the UI to allow a user to request geolocation. e.g.

```
<!-- geolocation check -->
<button id="check-location">check location</button>
```

and then add a listener for a button click, which we can use to query the geolocation api and user's location. e.g.

```
// button for location selector
var locButton = $('#check-location')

// event listener for geolocation button
locButton.on('click', () => {
  if (!navigator.geolocation) {
    return alert('geolocation not supported by your current web browser...');
  }

  // navigator object - getCurrentPosition accepts success and fail functions
  navigator.geolocation.getCurrentPosition( (position) => { // success
    console.log(position);
  }, () => { // error
    alert('unable to get location...');
  });
});
```

So, we can call the `getCurrentPosition()` method on the navigator object, specifying a success and fail callback. The position parameter in the success callback contains the location details for the current user, e.g.

```
position.coords.latitude
position.coords.longitude
```

This is what we need to be able to plot a user's location on a map, for example.

geolocation output We can output the user's location to the DOM, for example, and then plot their position on map &c.

```
// output location coordinates
var locOutput = $('#location-output');
...
// navigator object - getCurrentPosition accepts success and fail functions
navigator.geolocation.getCurrentPosition( (position) => { // success
  // console.log(position);
  // li element for position output
  var li = $('<li>');
  // set text for coordinates
  li.text(`latitude = ${position.coords.latitude} & longitude = ${position.coords.longitude}`);
  // append new element to DOM placeholder
  locOutput.append(li);
}, () => { // error
  alert('unable to get location...');
});
...
```

update UX - button visibility As the geolocation query may take a few seconds to complete, we need to ensure that a user does not repeatedly click the `check location` button.

So, we can add an attribute to disable the button, e.g. in the listener for the click event on the location button

```
// event listener for geolocation button
locButton.on('click', () => {
  if (!navigator.geolocation) {
    return alert('geolocation not supported by your current web browser...');
  }
  locButton.attr('disabled', 'disabled');
  // navigator object - getCurrentPosition accepts success and fail functions
  navigator.geolocation.getCurrentPosition( (position) => { //
    // remove disabled attribute
    locButton.removeAttr('disabled');
    // li element for position output
    var li = $('<li>');
    // set text for coordinates
    li.text(`latitude = ${position.coords.latitude} & longitude = ${position.coords.longitude}`);
    // append new element to DOM placeholder
    locOutput.append(li);
  }, () => { // error
    // remove disabled attributed - show location button to allow user to try again &c.
    locButton.removeAttr('disabled');
    alert('unable to get location...');
  });
});
```

As the app gets the current location, the button is `disabled` . Then, the `disabled` attribute is removed as the location's current position is returned and rendered.

If an error occurs &c., the `disabled` attribute will again be removed to allow a user to retry the location request.

update UX - dynamic button text We can also modify the text of the button whilst a query is being performed.

We can add a call to the `text()` method as we set the attribute for `disabled` on the `check location` button, e.g.

```
locButton.attr('disabled', 'disabled').text('checking location...');
```

and then ensure that the button text is reverted to original as we remove the `disabled` attribute from the button, e.g.

```
// remove disabled attribute - show location button to allow user to try again &c.
locButton.removeAttr('disabled').text('check location');
```