Upcoming APIs (Part 1)

Cpt S 489

Washington State University

Introduction

- Many of the interesting things happening in the JavaScript world are related to APIs that aren't yet standardized
- Some such things are supported on the major browsers but just aren't officially standardized
- Others are supported on only a few of the major browsers

W3C Status Descriptors

- When reading about a technology/API on MDN, the status column may use one of the W3C status descriptions

Working Draft (WD)

Candidate Recommendation (CR)

Proposed Recommendation (PR)

W3C Recommendation (REC)

(source: https://www.w3.org/2004/02/Process-20040205/tr.html)

W3C Status Descriptors

Working Draft (WD)

A Working Draft is a document that W3C has published for review by the community, including W3C Members, the public, and other technical organizations.

Candidate Recommendation (CR)

A Candidate Recommendation is a document that W3C believes has been widely reviewed and satisfies the Working Group's technical requirements. W3C publishes a Candidate Recommendation to gather implementation experience.

(source: https://www.w3.org/2004/02/Process-20040205/tr.html)

W3C Status Descriptors

Proposed Recommendation (PR)

A Proposed Recommendation is a mature technical report that, after wide review for technical soundness and implementability, W3C has sent to the W3C Advisory Committee for final endorsement.

W3C Recommendation (REC)

A W3C Recommendation is a specification or set of guidelines that, after extensive consensus-building, has received the endorsement of W3C Members and the Director. W3C recommends the wide deployment of its Recommendations. **Note:** W3C Recommendations are similar to the standards published by other organizations.

(source: https://www.w3.org/2004/02/Process-20040205/tr.html)

MDN Status Descriptors

- In short, if it’s not “(ST) Standard”, then, well it’s not an “official” standard.
- Many developers choose to ignore the status and proceed using the feature if it’s supported on the major browsers at this point in time
- Depending on the indented supported platforms/devices, this may or may not suffice for your needs
- Important that you just know how to interpret the documentation

A simple example: battery API

- Purpose of API: To get battery status
- MDN status*: (CR) Candidate recommendation
- Supported on*:
  - Chrome (on desktop or Android)
  - Firefox
- Not supported on*:
  - Safari
  - Chrome on iOS

* = status as of April 2017

A simple example: battery API

- BatteryManager object
- Page on MDN lists members:
  - charging (Boolean)
    - Indicates whether or not the battery is currently charging
  - chargingTime (Number)
  - dischargingTime (Number)
  - level (Number)
    - Represents battery percentage in range [0.0,1.0]

A simple example: battery API

- How to get the BatteryManager object:
- window.navigator.getBattery() function
  - or just navigator.getBattery()
- One caveat: the function does NOT return a BatteryManager object
  - Returns a Promise object that can be used (if all goes well) to get the BatteryManager object

Promise Object

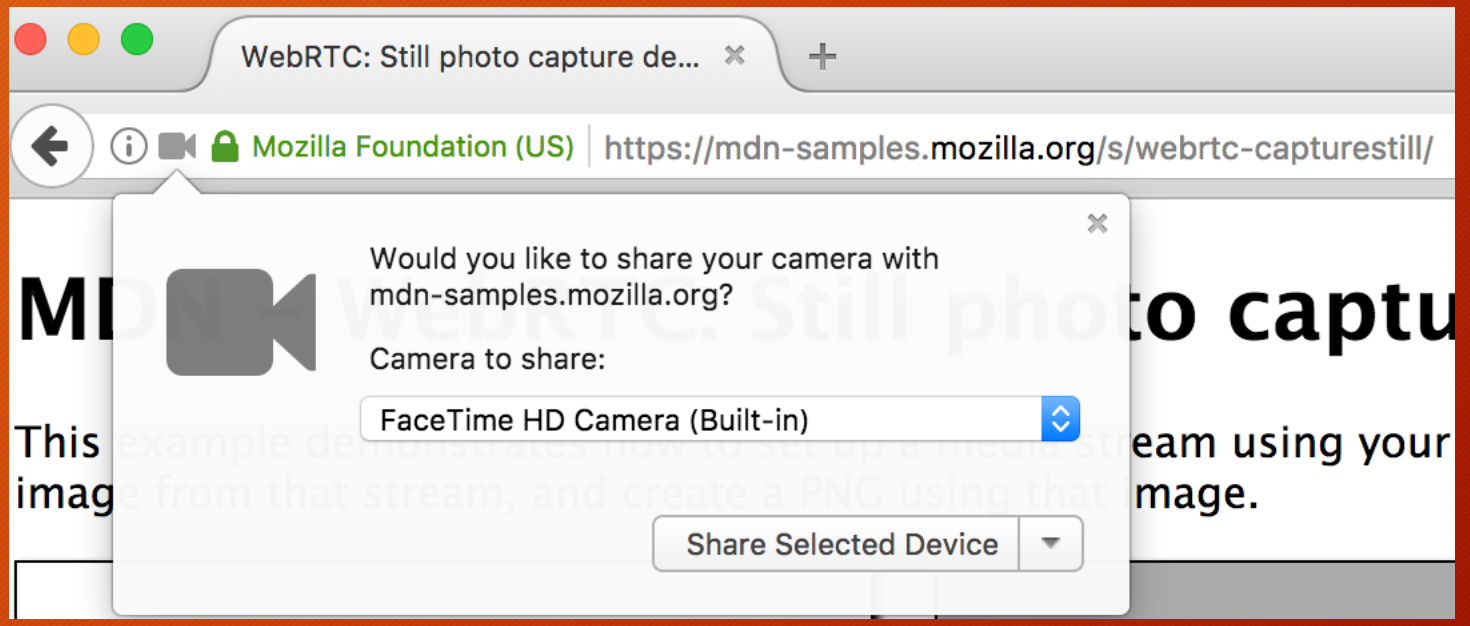
- Promise docmentation page on MDN states: “The Promise object is used for asynchronous computations. A Promise represents a value which may be available now, or in the future, or never.”
- Critical thinking question: Why design an API to return such an object? (discuss)

Promise Object

- Many APIs access “system” functionality (core devices/services)
- User may not want to give browser access to these devices/services
- When promises are used in code, the user may be prompted by the browser to allow access
  - Whenever the user decides, then the promise is resolved (fulfilled or rejected)
  - Hence a lot of these newer APIs giving back Promise objects

Promise Object

- Webcam a good example (hopefully if time permits this semester we will discuss the associated API)



Promise Object

- promiseObject.then function takes 2 functions as parameters
  - First is the function to call if promise is fulfilled
  - Second is the function to call if promise is rejected
- Example (assume promiseObj is an instance of Promise):

```
promiseObj.then(
   function(obj) { console.log(“Got ” + obj+ “ from promise”; },
   function(reason) { console.log(“Promise not fulfilled”); }
);
```

Battery API

- Back to the battery API, the navigator.getBattery() function returns a promise
- When fulfilled, we get back the actual BatteryManager object as the value

```
var p = navigator.getBattery();
p.then(
  function GotBatteryManager(manager) { … }
);
```

Battery API

- In the case of the battery manager, no browsers (that I've seen) ask the user for permission
- They either give access if the browser has the API implemented, or simply don't even implement navigator.getBattery()
- Concept review question: how would you check to see if navigator.getBattery is a function? (discuss)

Geolocation API

- Another API that may not be fully finalized in terms of standards
  - MDN status (as of April 2017): (REC) W3C Recommendation
- But this one seems to be supported currently on all major browsers
- Unlike the battery API:
  - promises are not involved
  - browsers WILL prompt for permission
    - Certain browsers may deny access for various reasons, even if they support the API. For example, if the page is not hosted under a secure (HTTPS) site, then browser will likely reject access to the API.
  - access through already instantiated object: navigator.geolocation
    - This object is an instance of Geolocation object

Geolocation API

- Geolocation.getCurrentPosition function
- Can call with 1 parameter (2 others optional) that is a callback function
- Callback function gets Position object as parameter
- Example:

Code:
```
navigator.geolocation.getCurrentPosition(
   function(pos) {
      console.log(pos.coords.latitude + "," + pos.coords.longitude);
});
```

Output: 46.7319,-117.1542

Geolocation API

- Also supports monitoring position changes via the Geolocation.watchPosition function
- Like the getCurrentPosition function, a callback function is passed
  - Callback invoked “every so often” when the position is being watched
- Remove monitoring with Geolocation.clearWatch
- Try it: write a simple phone app that monitors position and walk around campus to see how it changes
  - Notice accuracy (or lack thereof) and take note of one more thing…

Geolocation API

- Several geolocation APIs, whether we're talking about JavaScript or other languages, support differing levels of accuracy
- A PositionOptions object can be passed to getCurrentPosition and watchPosition
  - enableHighAccuracy Boolean property
  - timeout property
- More accurate location determination may take a lot of device power in comparison with the less accurate method

Geolocation API

- So back to the simple app…

1. Probably set accuracy to high
2. Leave it on as you walk/drive around campus/town
3. Store the data for where you are every few seconds
   - Storage API to be discussed next
4. Analyze data to find out some interesting things
   - Algorithm to approximate percentage of paths traveled more than once
   - (if you have several days worth of data) Algorithm to compare traffic baed on day of week
   - many other interesting things

Geolocation API

- Implement an algorithm to answer the following question using a full day's worth of geolocation data:
- "Where were you at a specific time?"
- Function takes time parameter, returns (x,y) ordered pair for best approximation of location at that time
- Data stored is a sorted array of objects that have properties
  - latitude (Number)
  - longitude (Number)
  - timestamp (Number)
    - Number type and not Date. Why? (discuss)

Geolocation API

- Before you say “just find the timestamp closest to the requested time and return the location,” consider the fact that our day’s worth of location data might have stored a position only once every 5 minutes
  - Can’t always gather data with high frequency due to power limits
  - We want a function that can return a location that could potentially be between 2 data points
- How to implement? (discuss in class)