# Exposing Sensors in JS

## case study: screen-adjusted Device Orientation API

W3C TPAC October 2014
Geolocation Working Group Meeting
timvolodine@google.com

# Exposing sensors in JavaScript

- Current approach is somewhat ad-hoc
  - ondeviceorientation events on window (with immediate callback)
    ```
    window.addEventListener('deviceorientation', onOrientation)
    ```
  - *null-events* when sensor not available
- Expose a Sensor object
  - onchange event handler
  - maybe also onconnect / ondisconnect
  - more specific handlers
  - specify desired update frequency
  - provide synchronous access to latest value
    - e.g. requestAnimationFrame

# Exposing sensors in JavaScript

- How to expose the Sensor object?
  - new Sensor [1]:
    - `var light = new sensors.AmbientLight({ frequency: 100 });`
    - `light.value of sensor is “null” until updated`
    - `error event when permission denied`
    - `sensors.Temperature.requestValue().then(...);`
  - Using a promise to obtain Sensor object [2].
    - ```
      navigator.getDeviceOrientationSensor(“high”)
               .then(function(sensor){..},
                     function(error){..});
      ```
    - `reject when not available/permission denied`
    - `sensor.orientation contains latest values`

[1] https://github.com/rwaldron/sensors
[2] http://goo.gl/9wC7PI
      ```

# Exposing Sensors in Javascript

```
partial interface Navigator {
 Promise<DeviceOrientationSensor> getDeviceOrientationSensor(SamplingFrequencyType);
};

[ActiveDOMObject]
interface Sensor : EventTarget {
 readonly attribute double samplingFrequency;
 attribute EventHandler onchange;
};

interface DeviceOrientationSensor : Sensor {
 readonly attribute OrientationData orientation;
 DOMMatrix getOrientationAsRotationMatrix();
};
```

# Exposing Sensors in Javascript

- Battery considerations
  - desired sensor update frequency:
    - `enum SamplingFrequencyType { "low", "normal", "high" };`
  - stop when page is not visible.
- Privacy considerations
  - e.g. fingerprinting and monitoring keystrokes (crbug.com/421691).
  - limit data precision.
  - `navigator.requestAccelerometerPermission()` API ?
  - current idea is to request permissions in the browser i.e. renderer doesn't know about it.

# Screen Adjusted Device Orientation

- Goal: provide *device screen orientation* in 3D space.
- Handle using a library in JavaScript:
  - https://github.com/richtr/Full-Tilt-JS
- Using quaternions:
  - Let *q* be the device orientation quaternion,
  - let *a* be the screen orientation angle.
  - Then $q_{screen} = q * [\, cos(-a/2)\ sin(-a/2)*e_z \,]$
  - same as $q_{screen} = [\, cos(-a/2)\ sin(-a/2)*n \,] * q$
- Provide ‘native’ support
  - Screen Orientation API isn’t really widely available.
  - Developers need to find out for themselves.

# Screen Adjusted Device Orientation

- Screen Orientation API + Device Orientation API
  - screen orientation is a property of window
  - can change independently of device orientation

```
partial interface DeviceOrientationEvent
{
  readonly attribute double? screenAlpha;
  readonly attribute double? screenBeta;
  readonly attribute double? screenGamma;
}
```

  - a separate “virtual” sensor would combine two APIs in blink

```
navigator.getScreenAdjustedDeviceOrientationSensor(..)
```