HTML

Web workers, geolocation, touch events

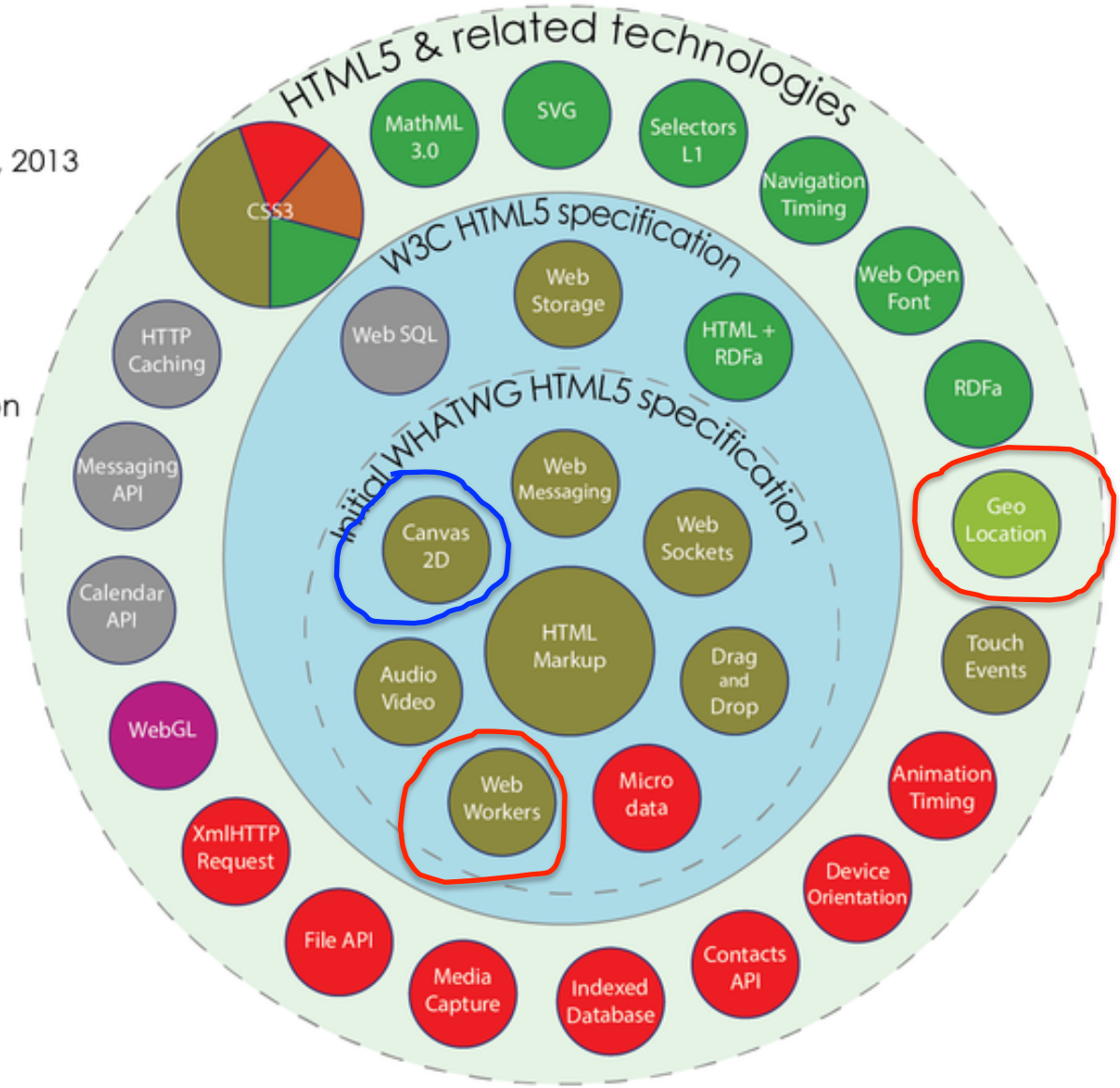
Overview

- Web Workers
  - Single-threaded woes
  - Web worker threads
    - Using, limitations
  - Examples
    - Fibonacci calculation
    - Mandelbrot set
- Geolocation
  - Privacy issues
  - JavaScript API
  - Example pages

HTML5

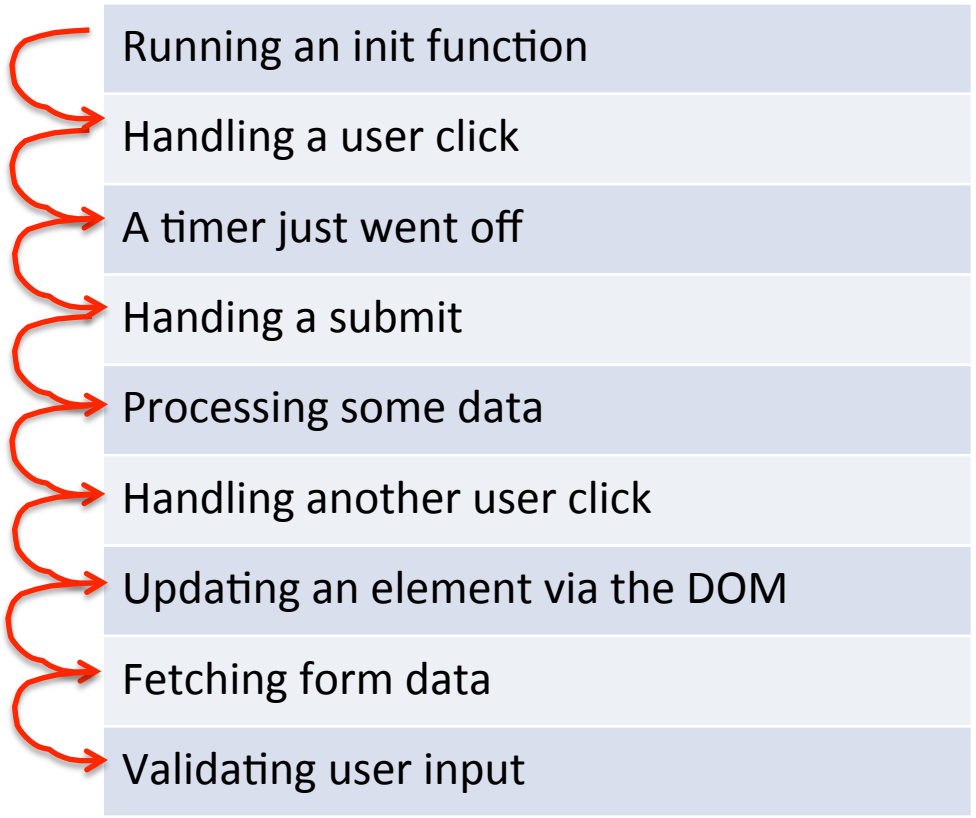
Taxonomy & Status on January 20, 2013

- W3C Recommendation
- Proposed Recommendation
- Candidate Recommendation
- Last Call
- Working Draft
- Non-W3C Specifications
- Deprecated



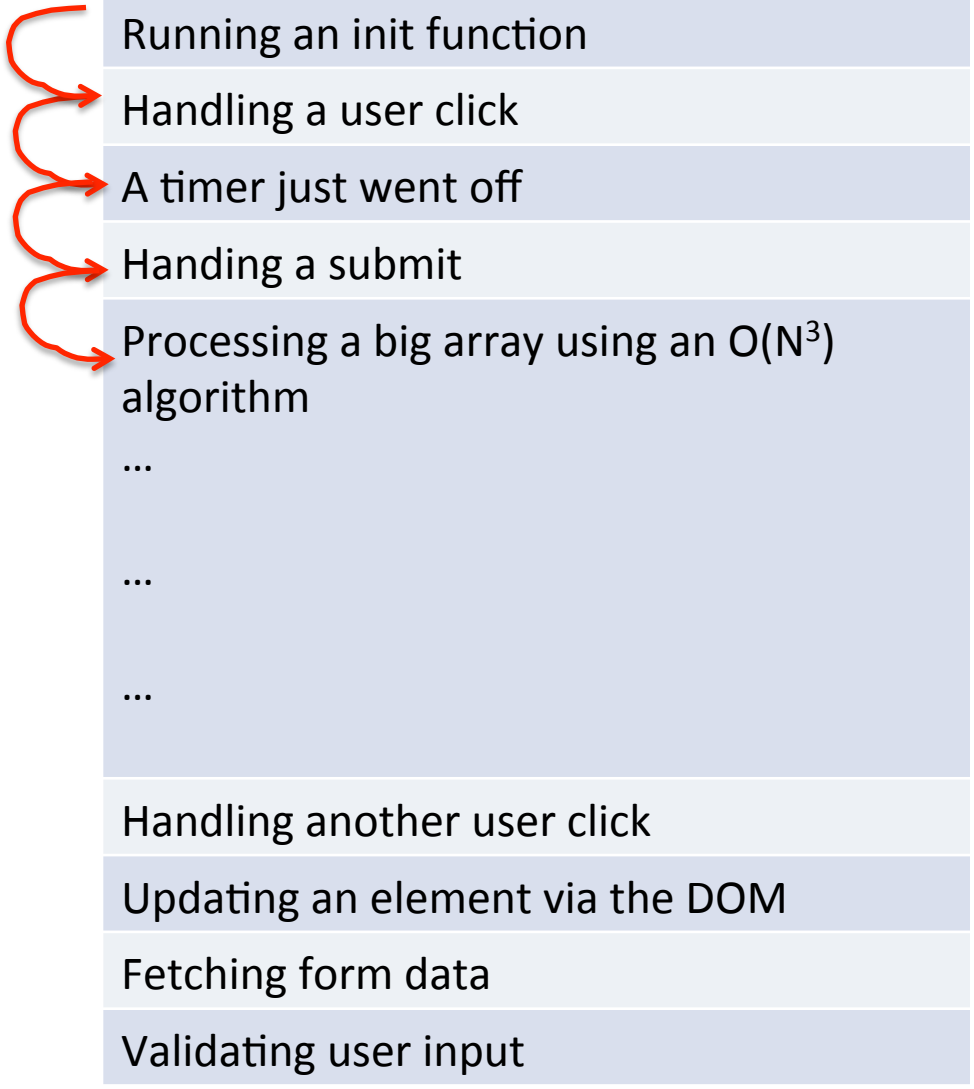
One thread to rule them all

- Prior to HTML5:
  - Single JavaScript thread running your page



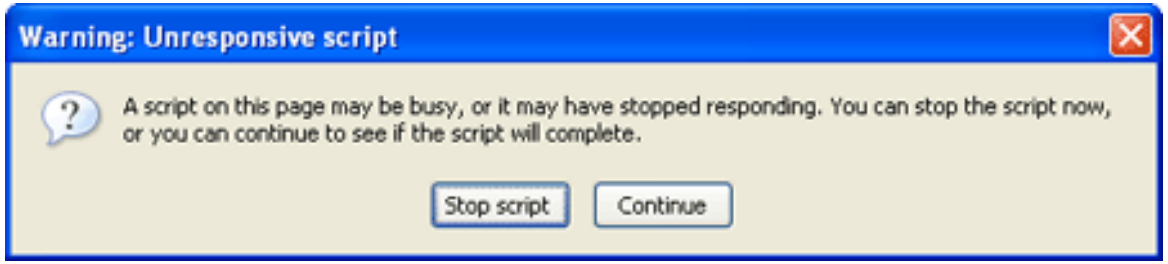
One thread to rule them all

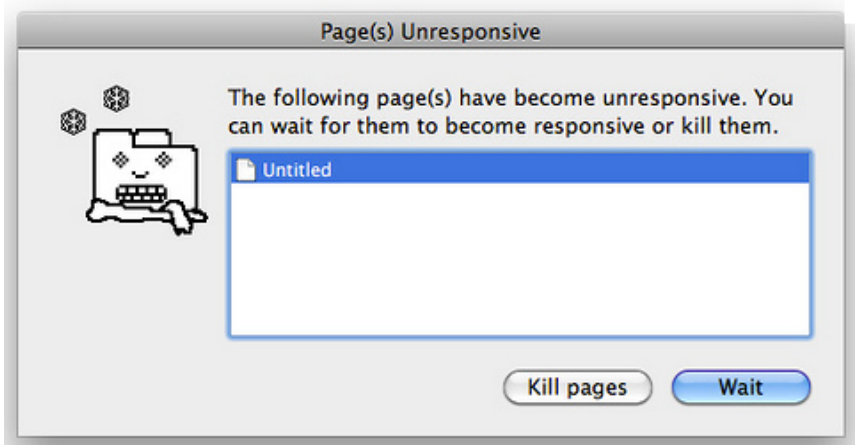
- Problem: Time consuming / blocking task

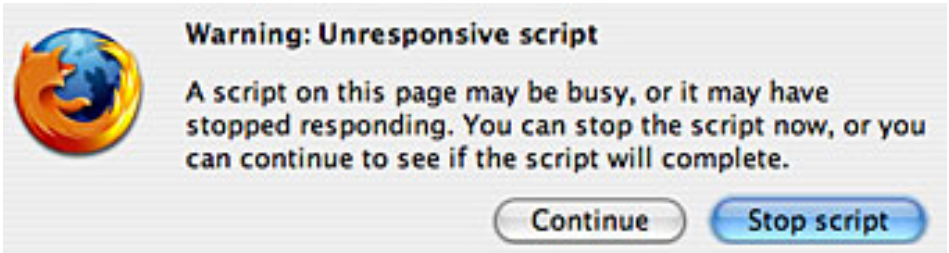


Single-threaded pros/cons

- Prior to HTML5:
  - Single-thread running all JavaScript
    - Pro: Easy to code and understand
    - Pro: No concurrency issues
    - Con: Page can become unresponsive to user





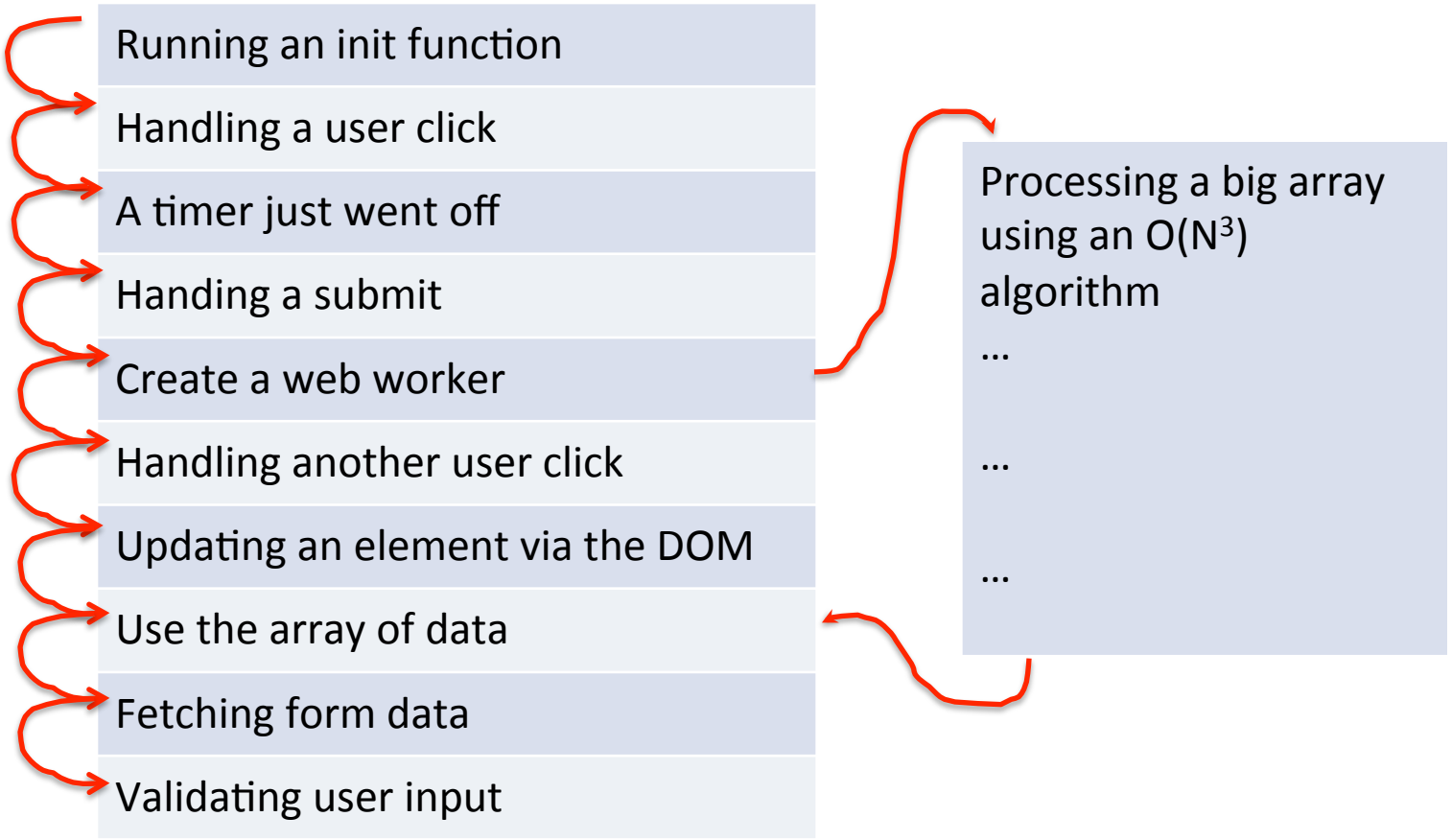


Web workers

- Multiple JavaScript threads running in browser
  - Offload time-consuming computational tasks
  - Better utilization of modern multi-core processors
  - Threading *may* model your problem better
    - Simplifying your code
- Type types
  - Dedicated workers
    - Linked to the creating page
  - Shared workers
    - Shared between all pages on a domain

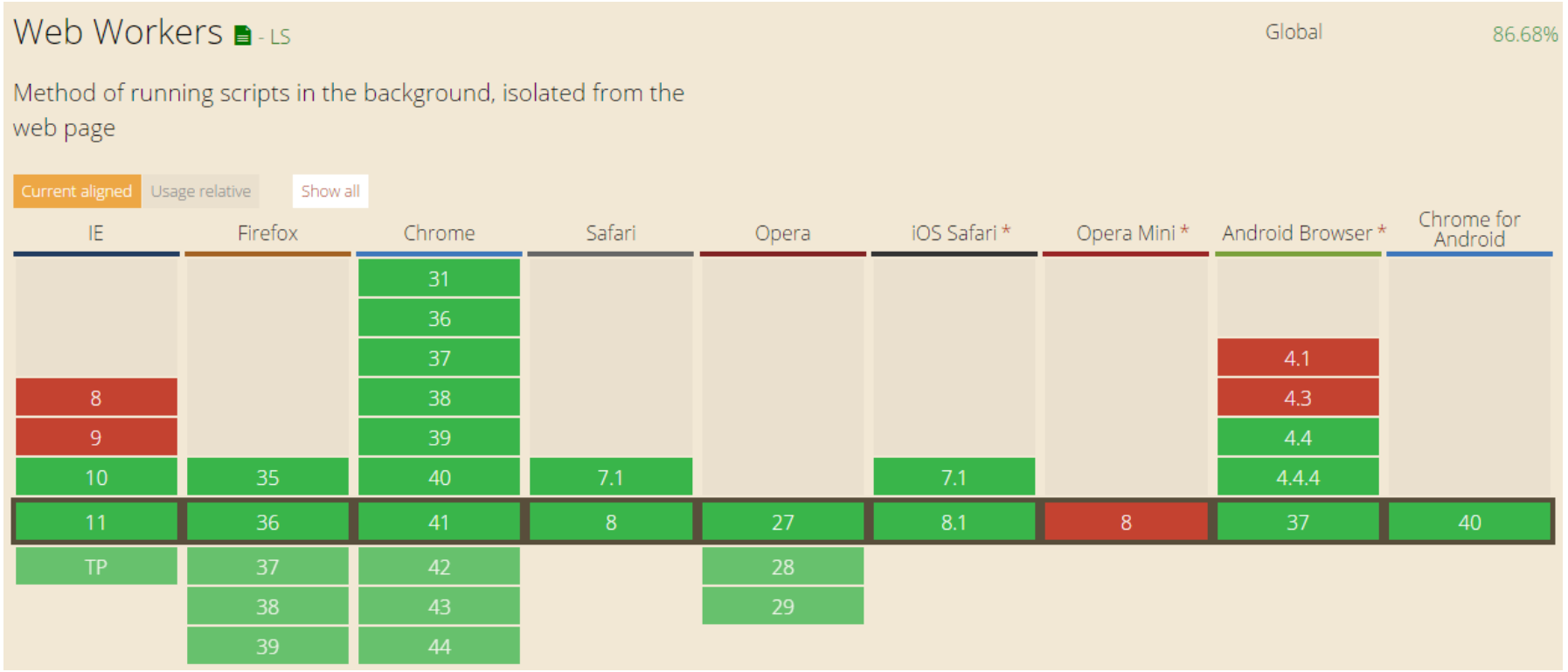
One thread to rule them all

- Problem: Time consuming / blocking task



Web workers

- How well are they supported?



http://caniuse.com/#search=worker

```
if (!window["Worker"])
{
    alert("No support for web workers!");
}
```

JavaScript code to test if browser supports web workers.

Web worker details

- Creating:
  - Worker is defined its own JavaScript file
- Limitations:
  - Workers don't have access to many things:
    - No access to DOM
    - No access to variables or functions in main program
    - No access to many runtime objects
      - e.g. window, document, parent
- Process:
  - Main program sends message to worker
  - Worker does work
  - Worker sends message back with results

Web worker example

- Goal:
  - Multithreaded Fibonacci calculator
  - Using simple (slow, wasteful) recursive algorithm
  - User types number, hits button to start calculation
  - Results appear in `<div>` as they arrive
- Single threaded version
- Multi-threaded version

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Fibonacci calculator</title>
<script>
    function goButton()
    {
        var num = document.getElementById("num").value;
        result = fib(num);
        document.getElementById("results").innerHTML +=
           "fib(" + num + ") = " + result + "<br />";
    }
    function fib(n)
    {
        if (n == 0)
            return 0;
        if (n == 1)
            return 1;
        return fib(n - 1) + fib(n - 2);
    }
</script>
</head>

<body>
<input type="text" size="10" id="num" /><br />
<input type="button" value="Go" onClick="goButton();"/>
<div id="results"></div>
</body>
</html>
```

fib1.html
Single threaded version

NOTE:
This is a stupendously inefficient way to calculate this!

fib2.html
Web Worker version

```
<script>
  function goButton()
  {
    if (!window["Worker"])
    {
       alert("No support for web workers!");
       return;
    }

    var num = document.getElementById("num").value;
    var worker = new Worker("fib.js");
    worker.onmessage = addResult;
    worker.postMessage(num);
 }
 function addResult(event)
 {
    document.getElementById("results").innerHTML +=
     "fib(" + event.data.num + ") = " +
     event.data.result + "<br />";
 }
</script>
```

fib.js
Worker JavaScript

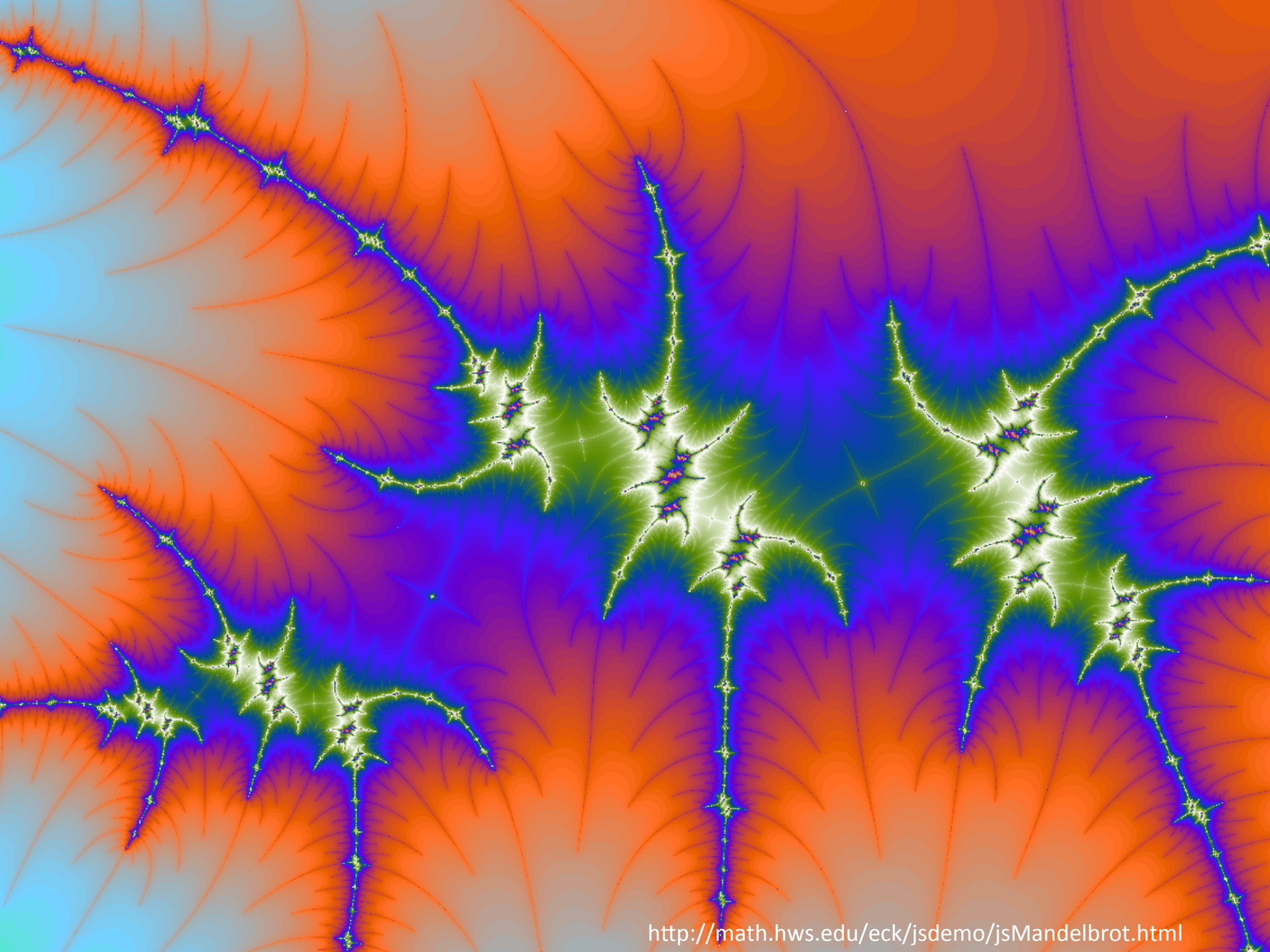
```
onmessage = startWork;

function startWork(event)
{
  var n   = event.data;
  var r   = fib(n);
  var msg = {num: n, result: r};
  postMessage(msg);
}

function fib(n)
{
  if (n == 0)
    return 0;
  if (n == 1)
    return 1;
  return fib(n - 1) +
    fib(n - 2);
}
```

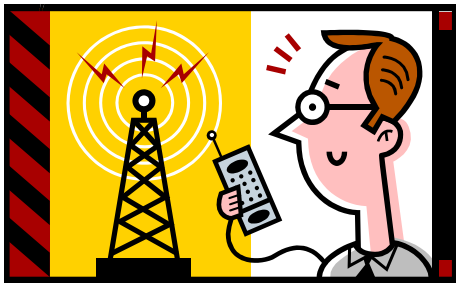
Other web worker details

- Killing a worker
  - `worker.terminate()` from main program
  - Worker can stop itself: `close()`
- Importing other JavaScript files:
  - Workers must use function: `importScripts()`
  - Also used for JSONP (JSON with padding)
    - For doing cross-domain Ajax
- Workers can also:
  - Spawn their own workers
  - Use `setInterval()` to schedule periodic work


http://math.hws.edu/eck/jsdemo/jsMandelbrot.html

Geolocation

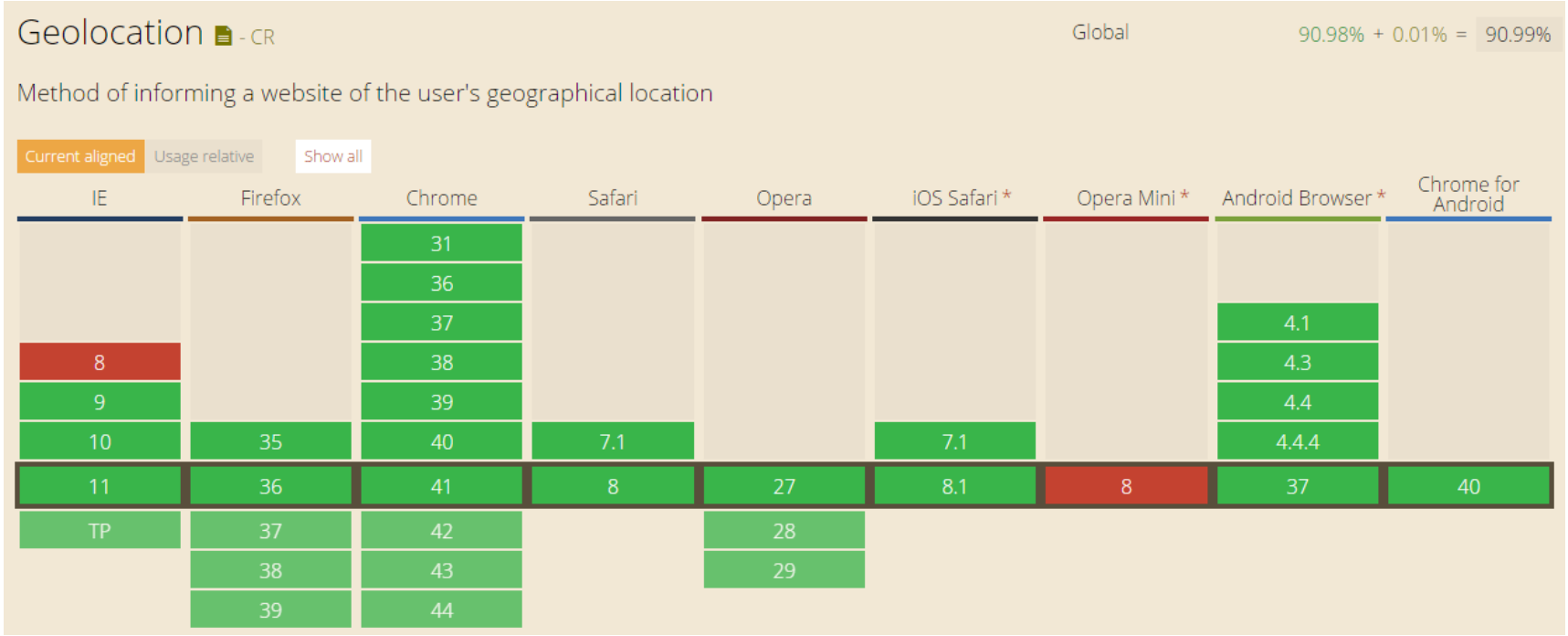
- Not part of W3C HTML5 spec
  - Another W3C standard
  - Widely supported
- High-level interface to device location info
  - Global Positioning System (GPS)
  - Or location inferred from network signal
    - IP address
    - RFID
    - WiFi, Bluetooth
    - GSM/CDMA cell IDs
    - User input



- One shot request or repeated updates

Geolocation

- How well is it supported?
  - Almost every modern desktop/mobile browser

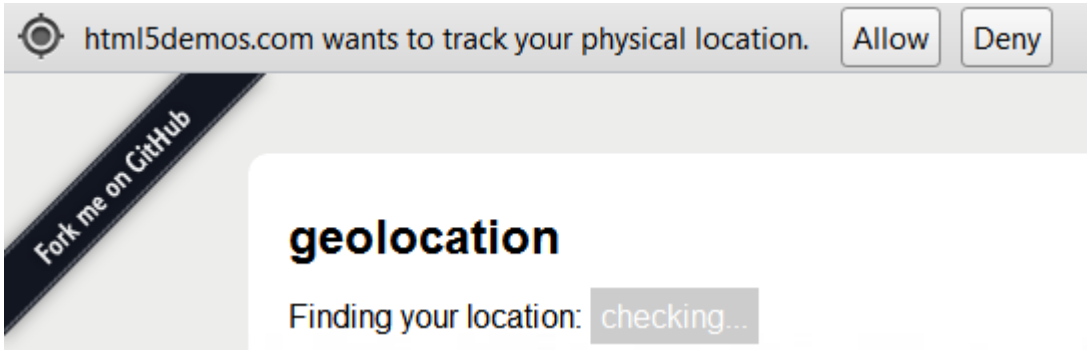



```
if (!navigator.geolocation)
{
   alert("No support for geolocation!");
}
```

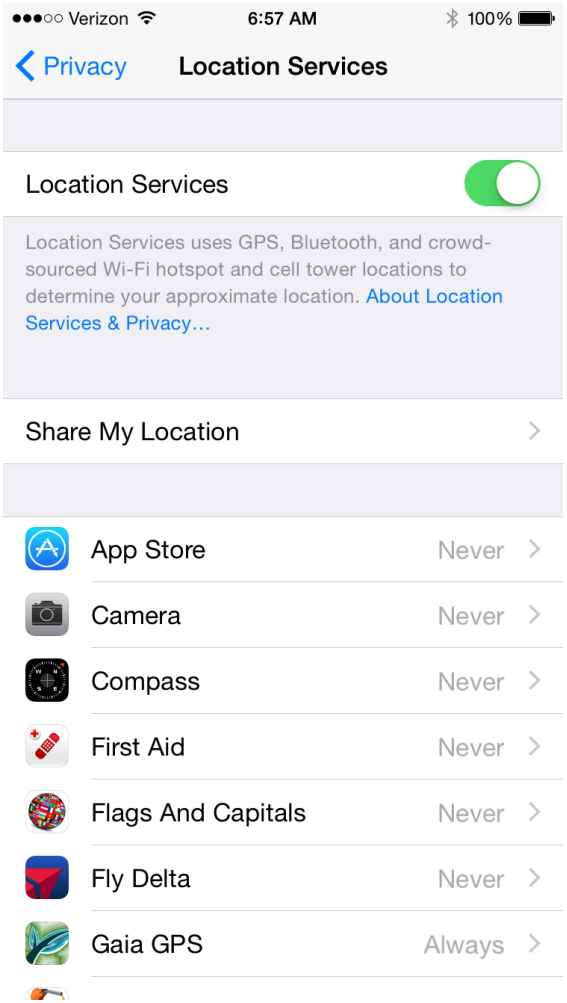
JavaScript code to test if browser supports geolocation.

Privacy issues

- User permission required



Desktop browser asking for permission



Location privacy settings on iPhone

Privacy issues

- WiFi access point databases
  - Send AP MAC / GPS / Cell IDs to central database
  - Apple/Google/Microsoft phones also stored history on device
  - Sometimes people's phones are an AP

http://petewarden.github.com/iPhoneTracker/

https://wigle.net/stats
http://www.skyhookwireless.com/

Related web services

- Often combined with the Google Maps API:

http://gmaps-samples-v3.googlecode.com/svn/trunk/map_events/map_events.html

http://earth-api-samples.googlecode.com/svn/trunk/demos/drive-simulator/index.html

http://maps.googleapis.com/maps/api/streetview?size=600x600&location=40.720032,%20-73.988354&heading=235&sensor=false

http://maps.googleapis.com/maps/api/elevation/json?locations=39.7391536,-104.9847034&sensor=true_or_false

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Where am I?</title>
<script>

window.onload = getMyLocation;

var options = { enableHighAccuracy: false, timeout: Infinity, maximumAge: 0 };

function getMyLocation()
{
    if (navigator.geolocation)
        navigator.geolocation.getCurrentPosition(displayLocation, displayError, options);
    else
        alert("No geolocation support!");
}

...

</script>
</head>
<body>
<div id="location">
</div>
</body>
```

geo1.html

```
function displayLocation(position)
{
    var lat              = position.coords.latitude;
    var long             = position.coords.longitude;
    var accuracy         = position.coords.accuracy;
    var timestamp        = position.timestamp;

    var altitude         = position.coords.altitude;
    var altitudeAccuracy = position.coords.altitudeAccuracy;
    var heading          = position.coords.heading;
    var speed            = position.coords.speed;

    var div  = document.getElementById("location");
    div.innerHTML  = "Latitude: "          + lat       + "<br>";
    div.innerHTML += "Longitude: "         + long      + "<br>";
    div.innerHTML += "Accuracy: "          + accuracy  + "<br>";
    div.innerHTML += "Timestamp: "         + timestamp + "<br><br>";

    div.innerHTML += "Altitude: "          + altitude         + "<br>";
    div.innerHTML += "Altitude accuracy: " + altitudeAccuracy + "<br>";
    div.innerHTML += "Heading: "           + heading          + "<br>";
    div.innerHTML += "Speed: "             + speed            + "<br>";
}

function displayError(error)
{
    var errorTypes = {0: "Unknown error", 1: "Permission denied by user",
                      2: "Position is not available", 3: "Request timed out"};
    var errorMessage = errorTypes[error.code];
    if (error.code == 0 || error.code == 2)
       errorMessage = errorMessage + " " + error.message;
    var div = document.getElementById("location");
    div.innerHTML = errorMessage;
}
```

geo1.html (continued)

Other features

- Measuring speed of location fix
  - geo2.html
    - Asks for high accuracy
    - Will not accept cached result, `maximumAge = 0`
    - `timeout = 10`, increase by 10ms on timeout

```
var options = { enableHighAccuracy: true, timeout:10, maximumAge: 0 };

function getMyLocation()
{
    if (navigator.geolocation)
        navigator.geolocation.getCurrentPosition(displayLocation,
                                                 displayError,
                                                 options);
    else
        alert("No geolocation support!");
}
```

Other features

- Updates whenever user moves
  - geo3.html
    - `watchPosition` instead of `getCurrentPosition`

```
var options = { enableHighAccuracy: true, timeout:0, maximumAge: 0 };

function getMyLocation()
{
    if (navigator.geolocation)
        navigator.geolocation.watchPosition(displayLocation,
                                            displayError,
                                            options);
    else
        alert("No geolocation support!");
}
```

geo4.html: adding Google maps

```
...
<style>
div#map
{
   margin: 5px;
   width: 400px;
   height: 400px;
   border: 1px solid black;
}
</style>

<script src="https://maps.google.com/maps/api/js?sensor=true"></script>
<script>
...
var map = null;

function showMap(coords)
{
   var googleLatAndLong = new google.maps.LatLng(coords.latitude, coords.longitude);
   var mapOptions = {zoom: 10, center: googleLatAndLong, mapTypeId: google.maps.MapTypeId.HYBRID};
   var mapDiv = document.getElementById("map");
   map = new google.maps.Map(mapDiv, mapOptions);
}

function displayLocation(position)
{
   ...
   showMap(position.coords);
}
...
<div id="map"></div>
```

geo5.html: adding pins

High accuracy: GPS, Wi-Fi, and mobile networks

Nexus 4

Update #633

geo5.html: adding pins

Battery saving: Wi-Fi and mobile networks

Nexus 4

Update #93

geo5.html: adding pins

Wi-Fi, Bluetooth
GPS, 3G

iPhone 4s

Update #276

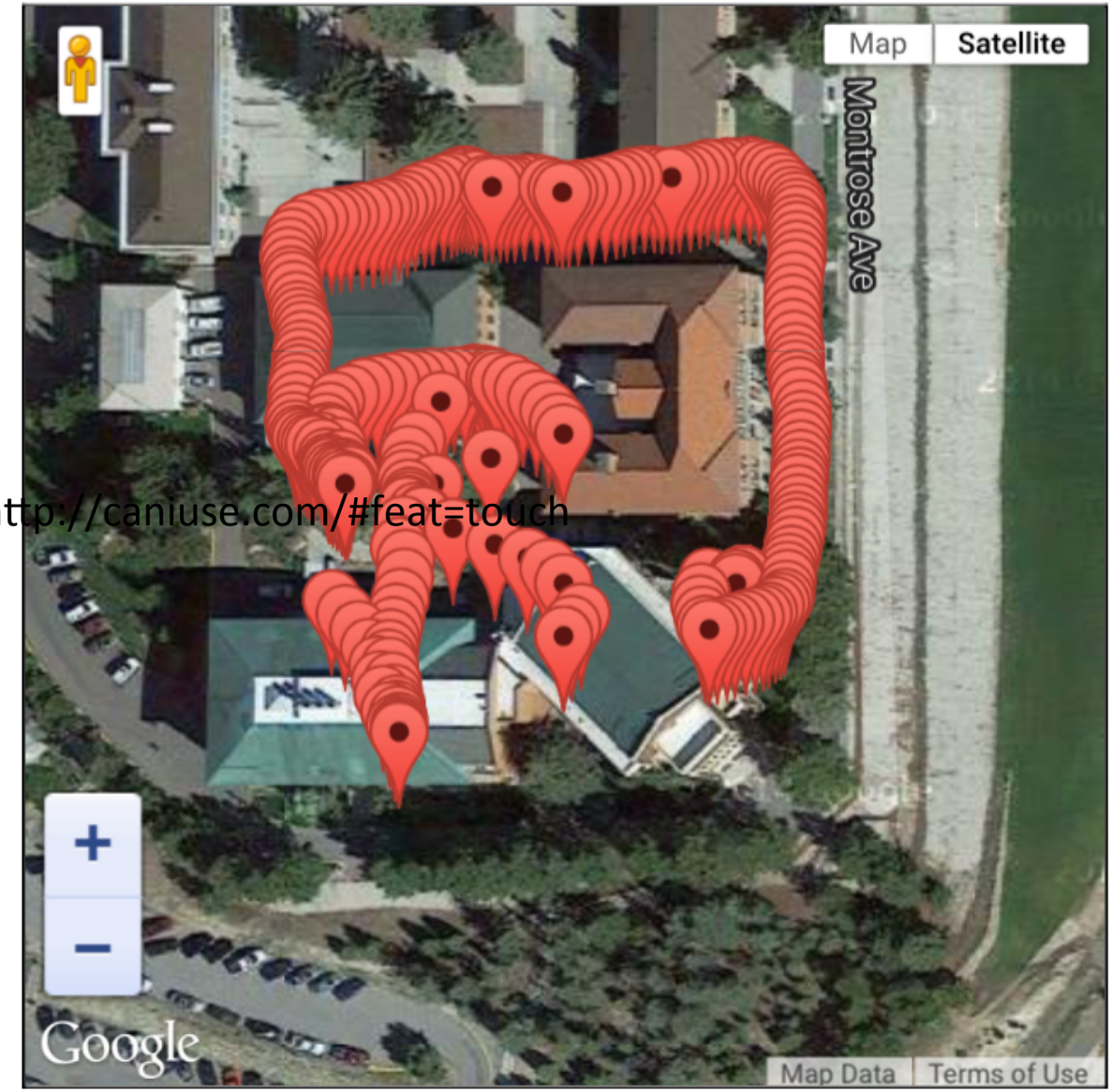
geo5.html: adding pins

Wi-Fi, Bluetooth, GPS, 4G

iPhone 6

Update #342

http://caniuse.com/#feat=touch



Touch events

- Multi-touch interaction
  - All modern mobile devices
  - Increasing on the desktop
- Touch Events API
  - A separate W3C standard
  - `Touches, TouchEvents, TouchLists`

Attribute	Description
`identifier`	unique numeric identifier
`target`	the DOM element that is the event target of the touch, even if the touch has moved outside of this element
`screenX`	horizontal coordinate relative to screen
`screenY`	vertical coordinate relative to screen
`clientX`	horizontal coordinate of point relative to viewport, excluding scroll offset
`clientY`	vertical coordinate of point relative to viewport, excluding scroll offset
`pageX`	horizontal coordinate of point relative to viewport, including scroll offset
`pageY`	vertical coordinate of point relative to viewport, including scroll offset

The main touch events defined in the specification are outlined in the table below. The target of the event is the element in which the touch was detected, even if the touch has moved outside this element.

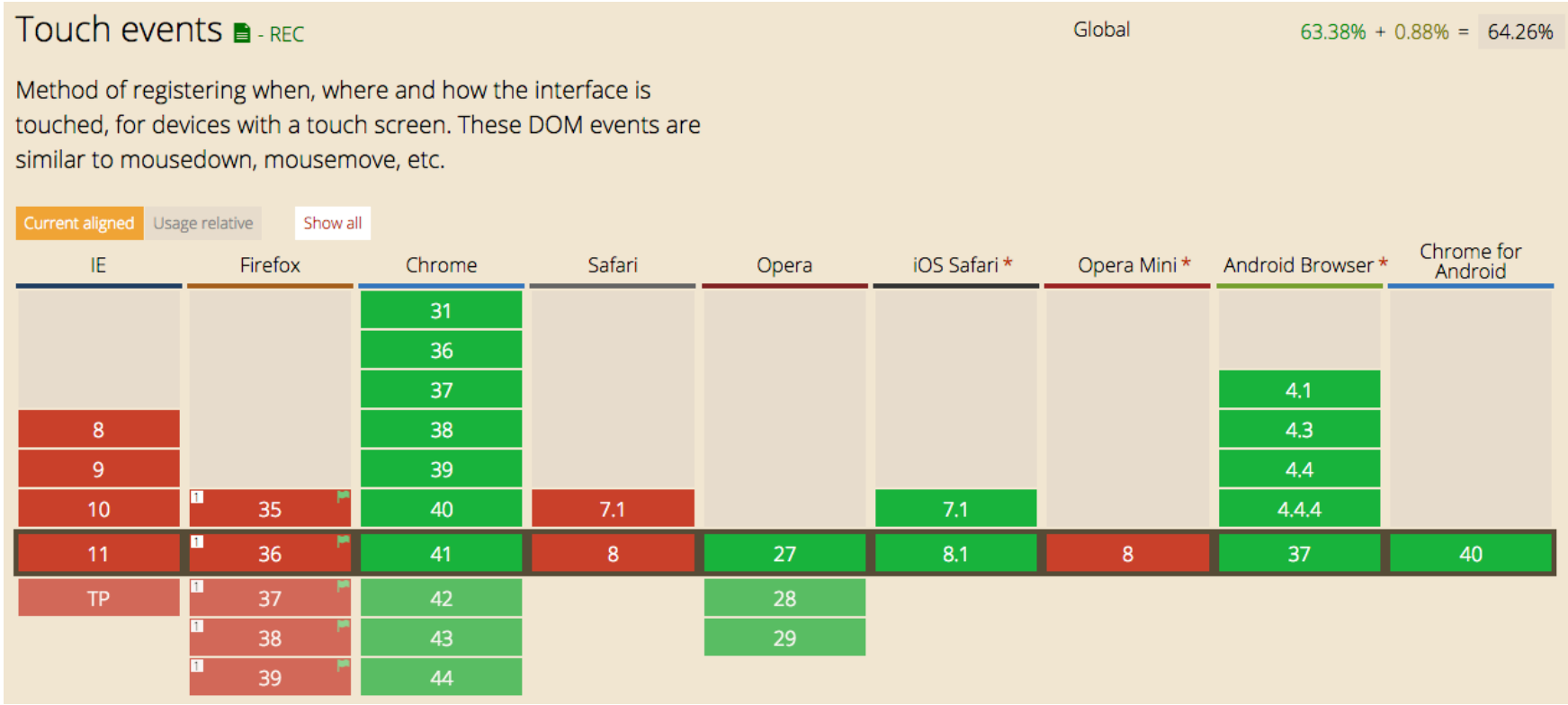
Event	Description
`touchstart`	triggered when a touch is detected
`touchmove`	triggered when a touch movement is detected
`touchend`	triggered when a touch is removed e.g. the user's finger is removed from the touchscreen
`touchcancel`	triggered when a touch is interrupted, e.g. if touch moves outside of the touch-capable area

Three lists of touches, *TouchLists*, are associated with each of the events. These are:

List	Description
`touches`	all current touches on the screen
`targetTouches`	all current touches that started on the target element of the event
`changedTouches`	all touches involved in the current event

Touch events

- How well is it supported?



http://caniuse.com/#feat=touch

Conclusions

- HTML5 + other associated APIs
  - Enable new and different web apps
    - Location-based services
  - Making browser apps more like desktop apps
    - Threading now supported
  - Making browser apps more like mobile apps
    - Multi-touch support