Geolocation

Carlo Taticchi

Luca Tranfaglia

Gian Marco Silieri

Official mascot :

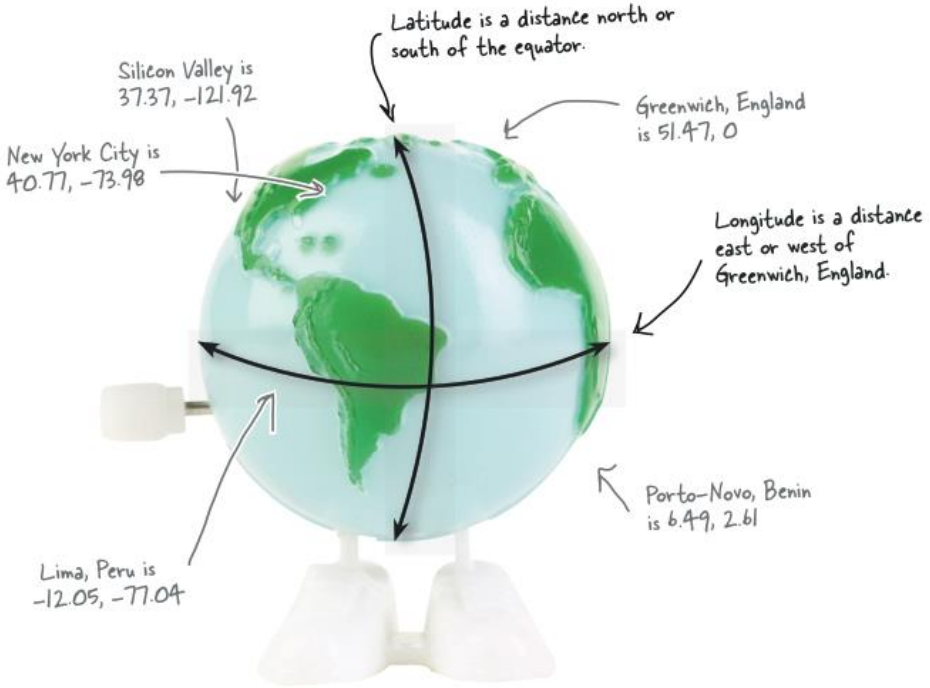
OPI

In this chapter we're going to show you how to create web pages that are LOCATION AWARE with a JavaScript API:Geolocation.

Location, Location, Location

You can give them directions, make suggestions about where they might go, you can know it's raining and suggest indoor activities, you can let your users know who else in their area might be interested in some activity. There's really no end to the ways you can use location information.

With HTML5 (and the Geolocation JavaScript-based API) you can easily access location information in yourpages.



The Lat and Long of it…

Latitude specifies a north/sourth point on the Earth, and longitude, an east/west point.

Latitude is measured from the equator, and longitude is measured from Greenwich, England.

The job of the geolocation API is to give us the coordinates of where we are at any time, using these coordinates

How the Geolocation API determines your location

You don't have to have the newest smartphone to be location aware.
Even desktop browsers are joining the game.

You might ask, how would a desktop browser determine its location if it doesn't have GPS or any other fancy location technologies?

IP ADDRESS

Location information based on your IP address uses an external database to mapthe IP address to a physical location.

However, oftenIP addresses are resolved to locations such as your ISP's local office.

GPS

Global Positioning System, supported by many newer mobile devices, provides extremely accurate location information based on satellites. Location data may include altitude, speed and heading information.
To use it, though, your device has to be able to see the sky, and it can take a long time to get a location.

GPS can also be hard on your batteries.

CELL PHONE

Cell phone triangulation figures out your location based on your distance from one or more cell phone towers (obviously the more towers, the more accurate your location will be). This method can be fairly accurate and works indoors (unlike GPS); It also can be much quicker than GPS.

WI FI

WiFi positioning uses one or more WiFi access points to triangulate your location. This method can be very accurate, works indoors and is fast.
Obviously it requires you are somewhat stationary

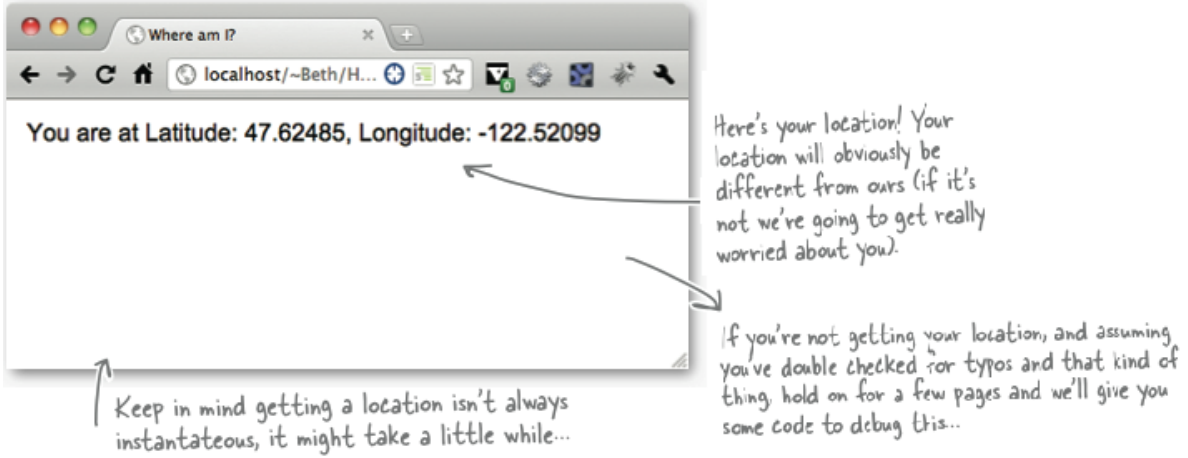
Just where are you anyway?

Well, of course you know where you are, but let's see where your browser thinks you are.

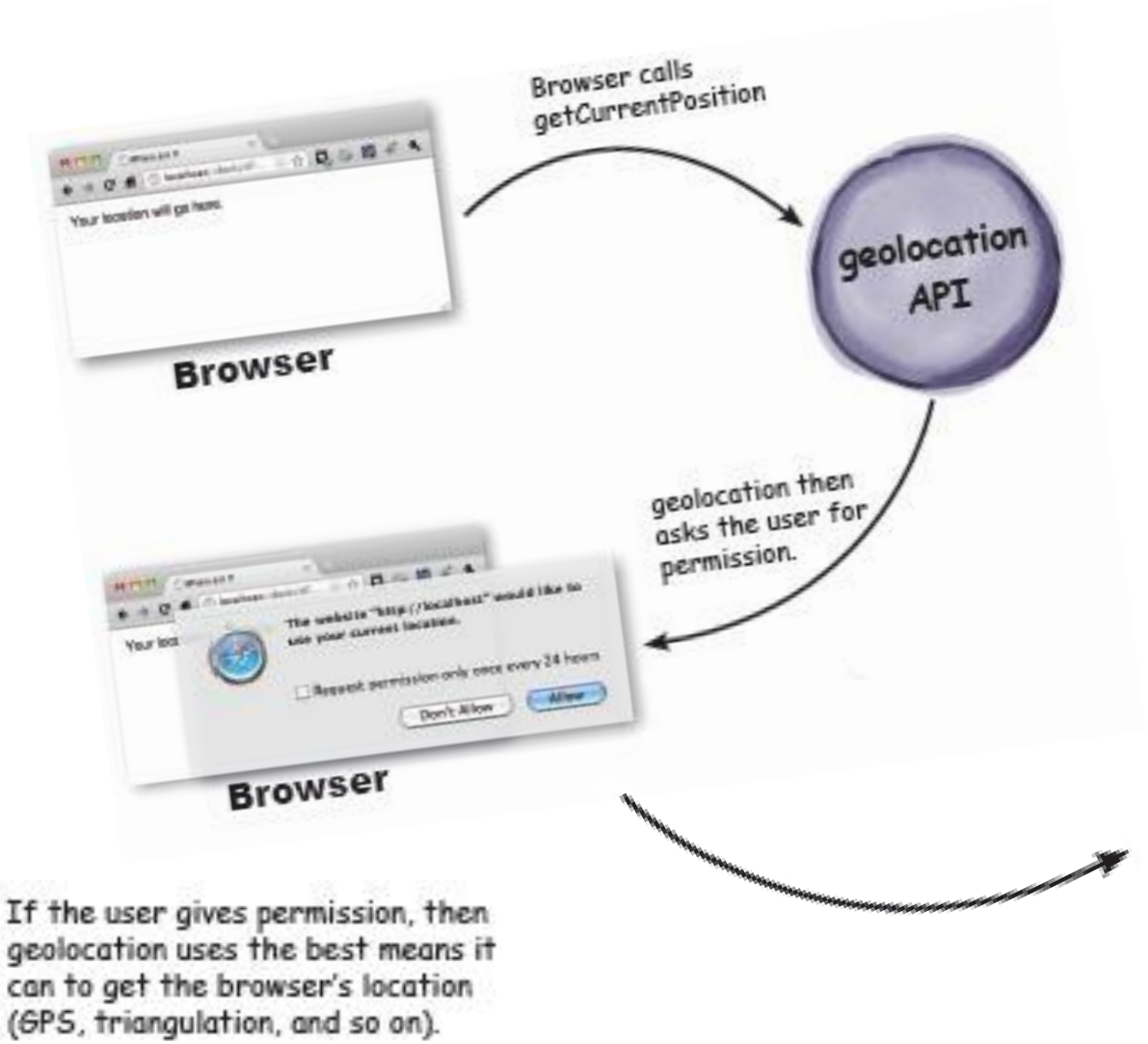
```
<!doctype html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Dove mi trovo?</title>
        <script src="myLoc.js"></script>
        <link rel="stylesheet" href="myLoc.css">
    </head>
    <body>
        <div id="location">
            La tua posizione andrà qui.
        </div>
    </body>
    </html>
```

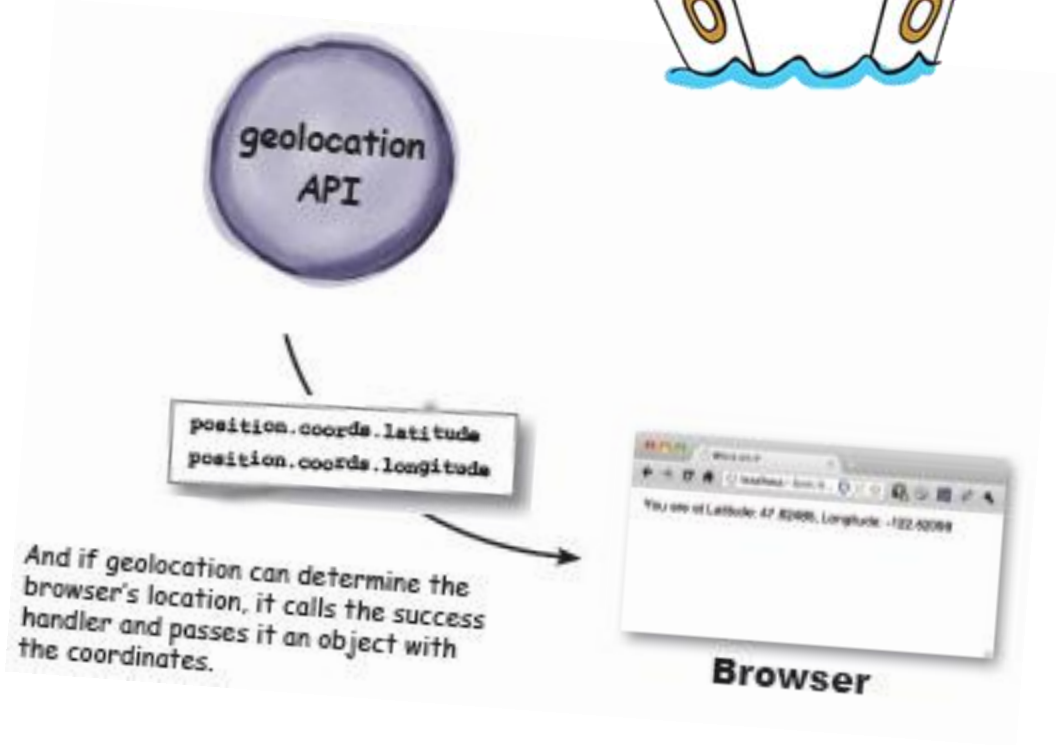
```
window.onload = getMyLocation;
function getMyLocation() {
    if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(displayLocation);
        }
    else {
        alert("Oops, no geolocation support");
    }
}
function displayLocation(position) {
    var latitude = position.coords.latitude;
    var longitude = position.coords.longitude;
    var div = document.getElementById("location");
    div.innerHTML = "Latitudine: " + latitude + ", Longitudine: " + longitude;
}
```

Test drive your location



How it all fits together





Revealing our secret location…

How about we see how far you are from our secret writing location at Wickedly Smart HQ? To do that we need the HQ coordinates and we need to know how to calculate distance between two coordinates.

Writing the code to find the distance

And now let's write the code: all we need to do is pass the coordinates of your location and our location to the `ComputDistance` function.

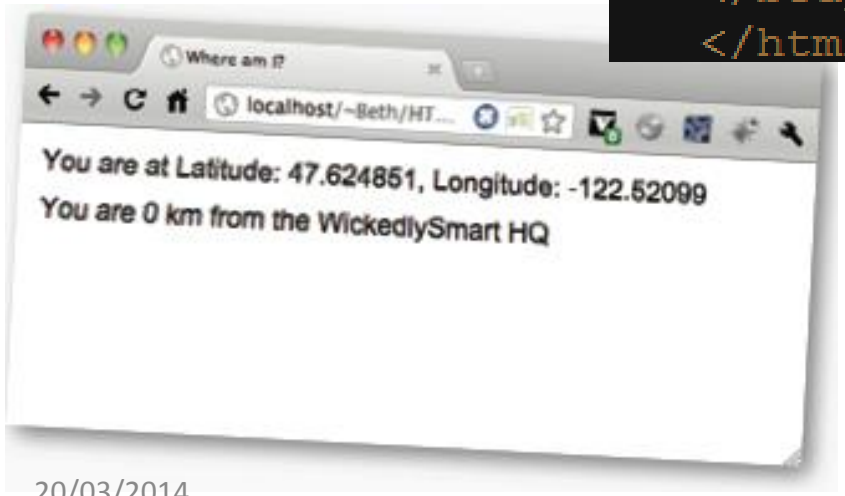
Location-enabled test drive

```
window.onload = getMyLocation;
//Variabile Globale - coordinate Wickedly Smart HQ
var ourCoords = { latitude: 47.624851, longitude: -122.52099 };


function getMyLocation() {
    if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(displayLocation);
        }
    else {
        alert("Oops, no geolocation support");
    }
}

function displayLocation(position) {
    var latitude = position.coords.latitude;
    var longitude = position.coords.longitude;
    var div = document.getElementById("location");
    div.innerHTML = "Latitudine: " + latitude + ", Longitudine: " + longitude;
    var km = computeDistance(position.coords, ourCoords);
    var distance = document.getElementById("distance");
    distance.innerHTML = "You are " + km + " km from the WickedlySmart HQ";
}
```

```
<!doctype html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Dove mi trovo?</title>
        <script src="myLoc.js"></script>
        <link rel="stylesheet" href="myLoc.css">
    </head>
    <body>
        <div id="location">
            La tua posizione andrà qui.
        </div>
        <div id="distance">
            Distance from WickedlySmart HQ will go here.
        </div>
    </body>
    </html>
```

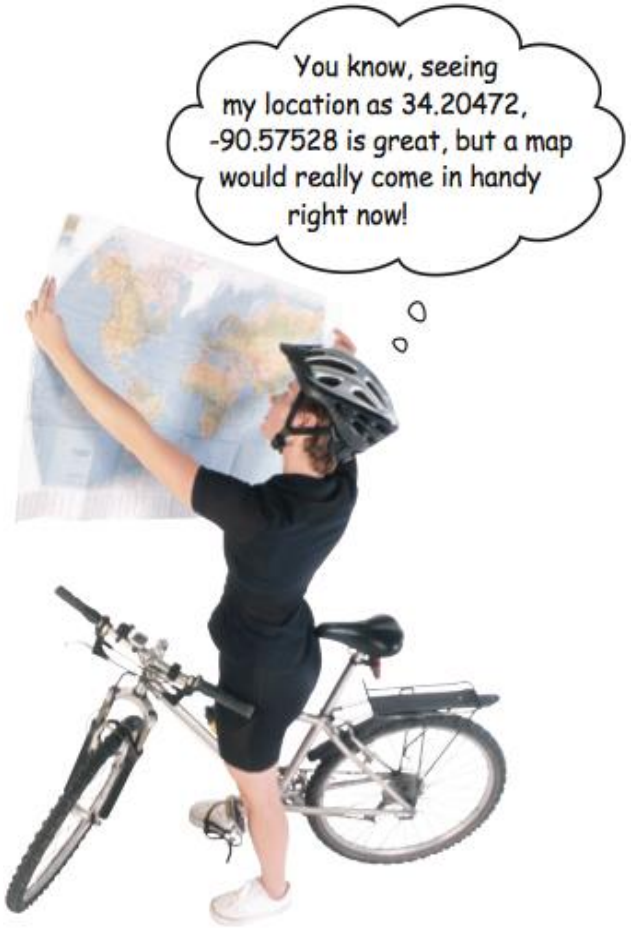


Mapping your position

As we told you up front, the Geolocation API is pretty simple—it gives you a way to find (and as you'll see, track, as well) where you are, but it doesn't provide you with any tools to visualize your location.

To do that we need to rely on a third-party tool, and as you might guess, Google Maps is by far the most popular tool for doing that.

Obviously Google Maps isn't part of the HTML5 spec, but it does interoperate well with HTML5, and so we don't mind a little diversion here and there to show you how to integrate it with the Geolocation API.



How to add a Map to your Page

Create a "div" (like this below) to put our Google Map in HTML page.

HTML

```
<body>
  <div id="location">
    Your location will go here.
  </div>
  <div id="distance">
   Distance from WickedlySmart HQ will go here.
  </div>
  <div id="map">
  </div>
</body>
</html>
```

CSS

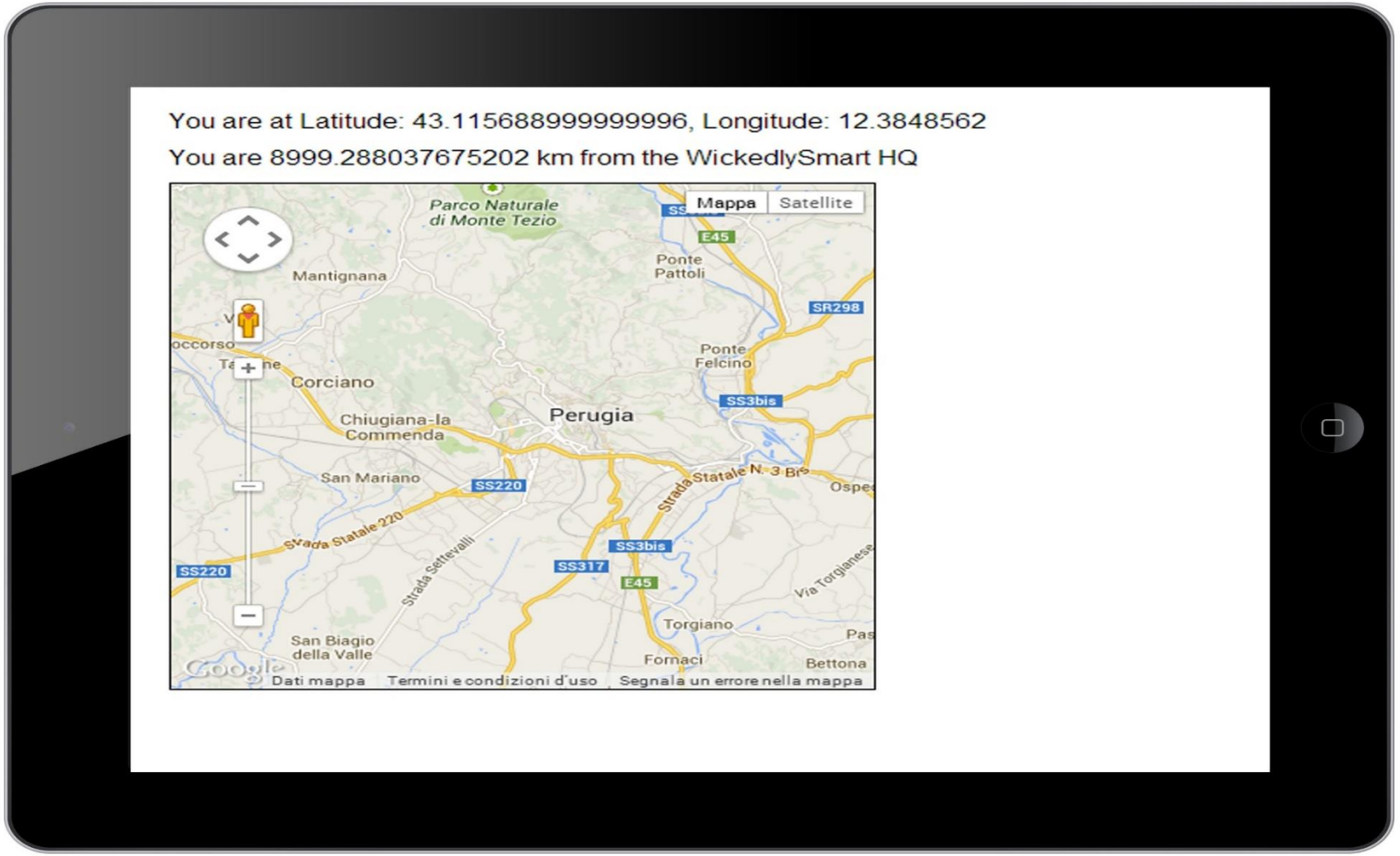
```
#map {
  margin: 5px;
  width: 400px;
  height: 400px;
  border: 1 px solid black;
}
```

JavaScript

```
var map;

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

That's the result!

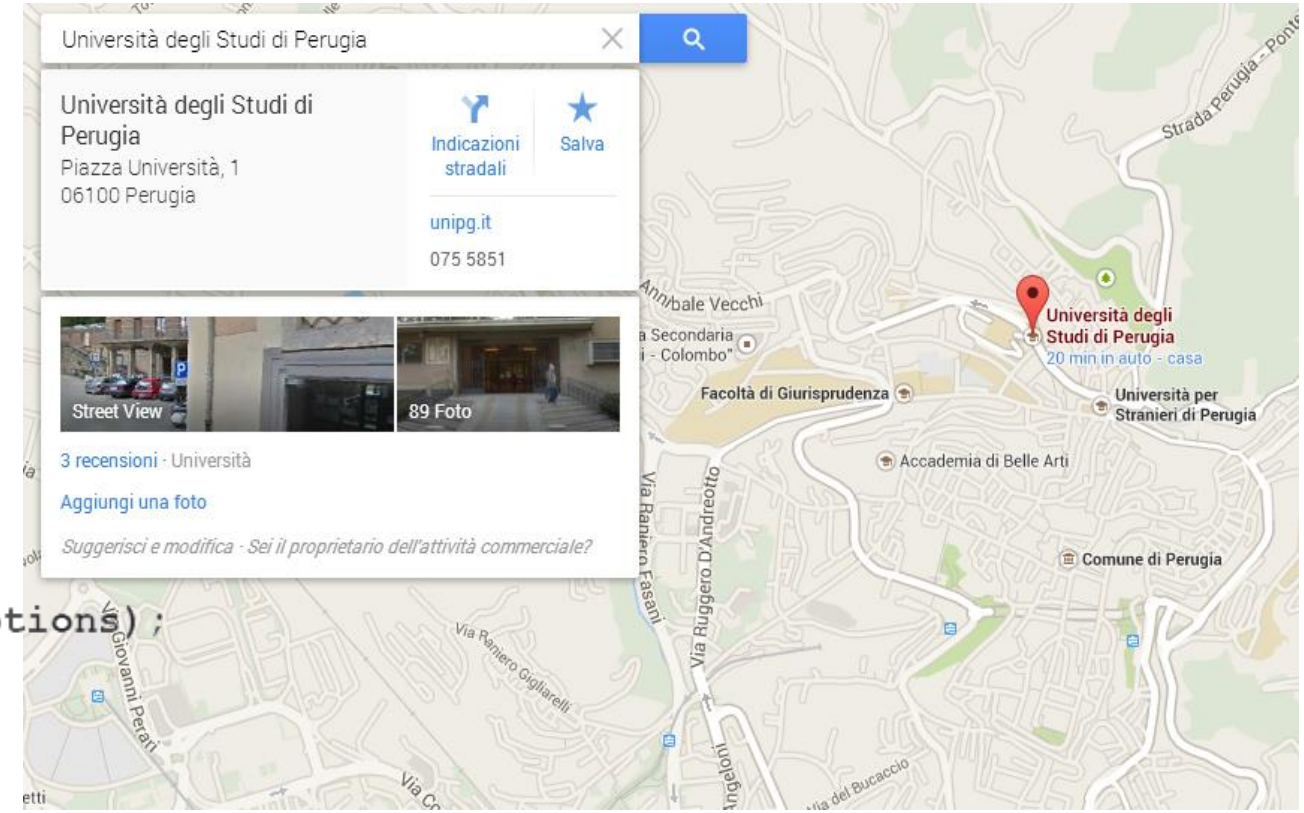


http://wickedlysmart.com/hfhtml5/chapter5/map/myLoc.html

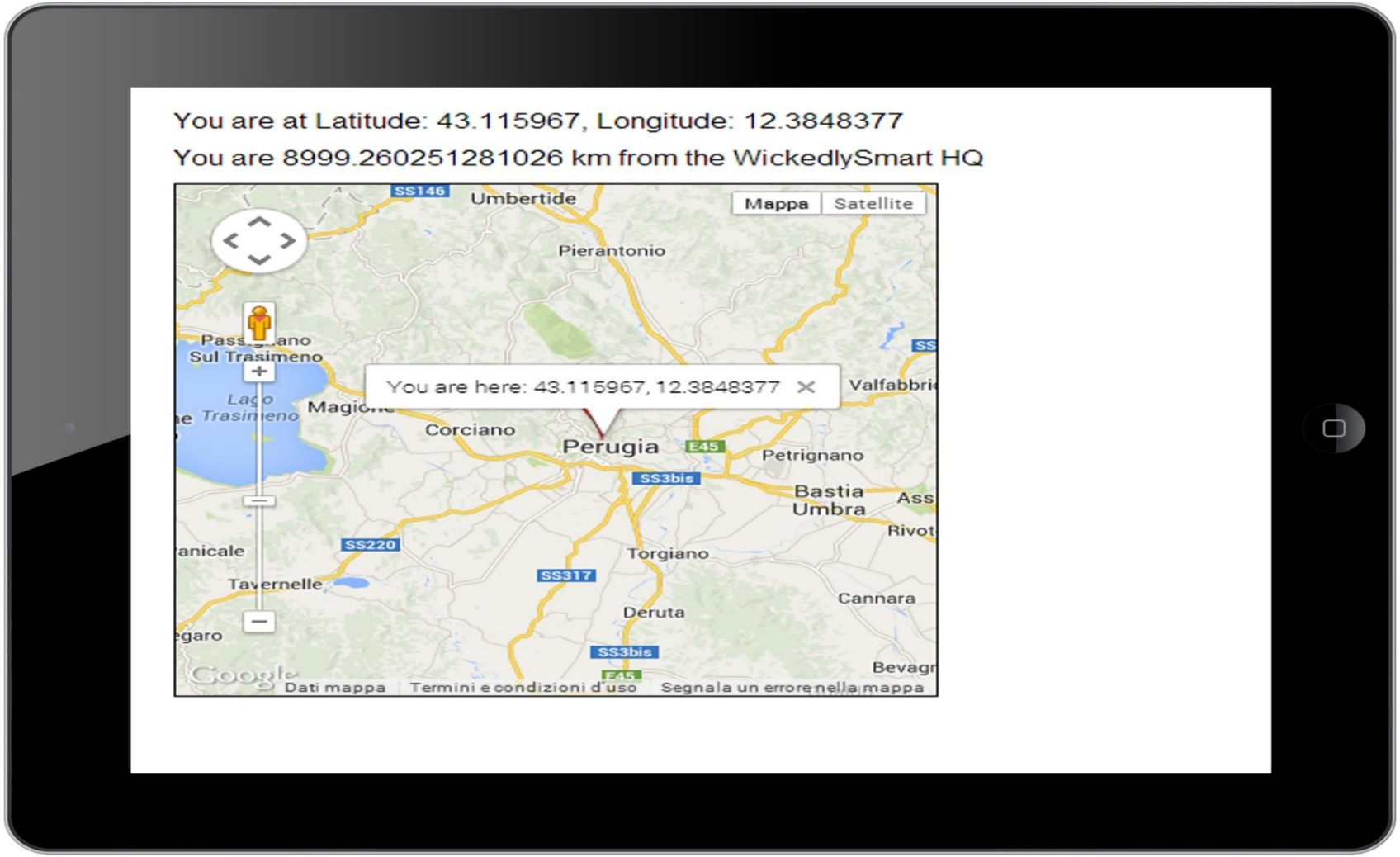
.. And now adding a Google Maker

JavaScript

```
function addMarker(map, latlong, title, content) {
  var markerOptions = {
      position: latlong,
      map: map,
      title: title,
      clickable: true
  };
  var marker = new google.maps.Marker(markerOptions);
}
```



The Google map with maker!



http://wickedlysmart.com/hfhtml5/chapter5/marker/myLoc.html

Google Maps API

Some pointers to where to start:

- Controls: By default, your Google map includes several controls, like the zoom control, a control to switch between Map and Satellite view, and even the Street View control (the little pegman above the zoom control). You can access these controls programmatically from JavaScript to make use of them in your applications.

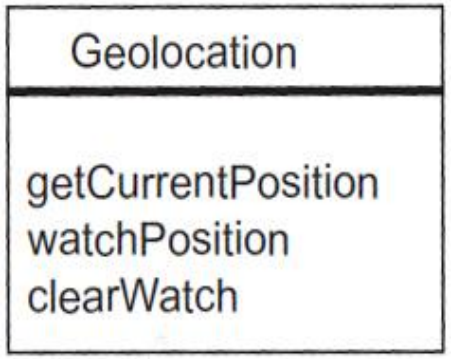
- Services: You have access to directions, and other services, like distance and street view.

- Overlays: Overlays provide check traffic congestion. You can create custom markers, and pretty much anything else you can imagine, using the Google Maps overlay APIs.

All this is available through the Google Maps JavaScript API. Check out the documentation at:

https://developers.google.com/maps/documentation/javascript/?csw=1

Geolocation API

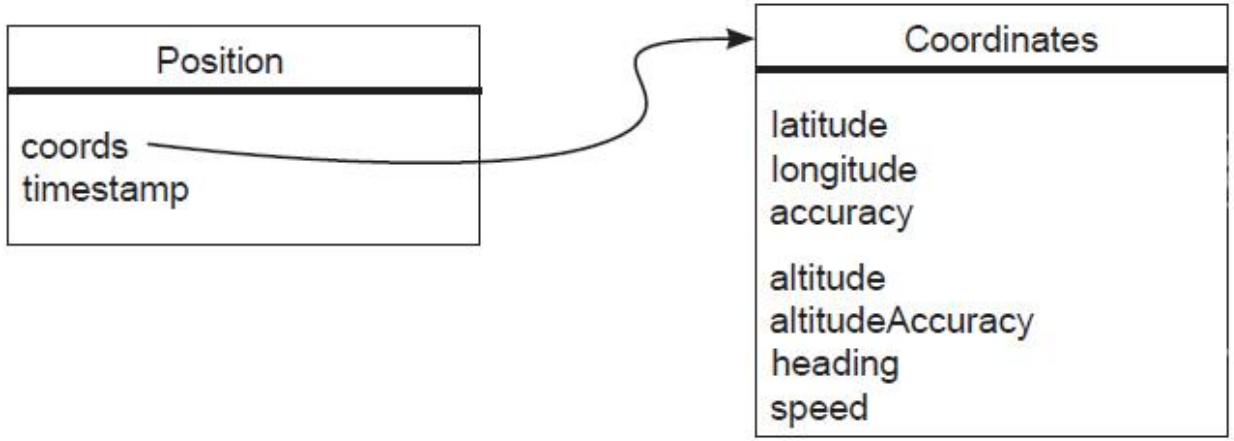


getCurrentPosition(successHandler, errorHandler, positionOptions);

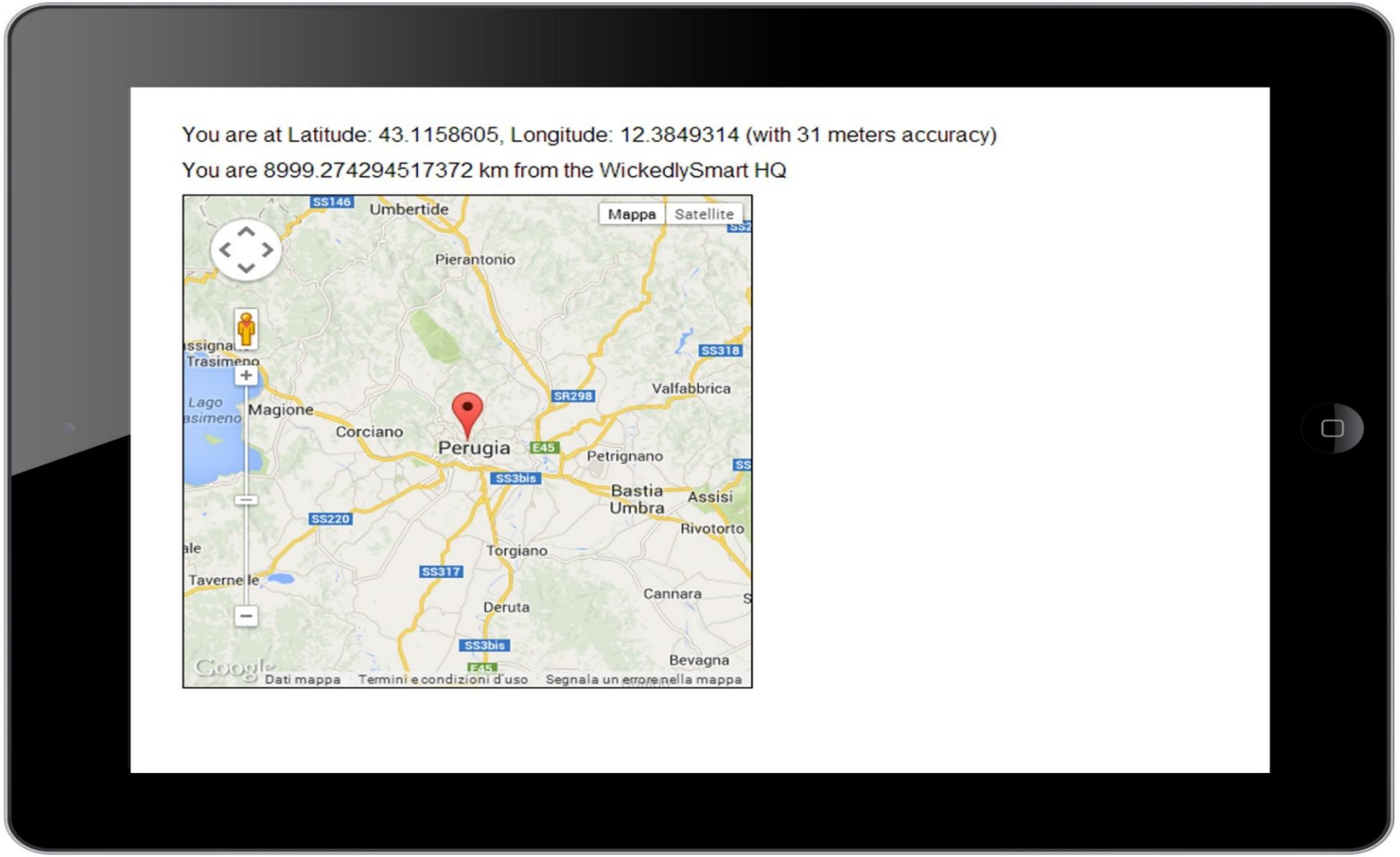
```
function displayLocation(position) {
    var latitude = position.coords.latitude;
    var longitude = position.coords.longitude;
    var div = document.getElementById("location");

    div.innerHTML = "You are at Latitude: " + latitude +
    div.innerHTML += " (with " + position.coords.accuracy

    var km = computeDistance(position.coords, ourCoords);
    var div = document.getElementById("distance");
    distance.innerHTML = "You are " + km + " km from the WickedlySmart HQ";
    showMap(position.coords);
}
```



Accuracy



http://wickedlysmart.com/hfhtml5/chapter5/accuracy/myLoc.html

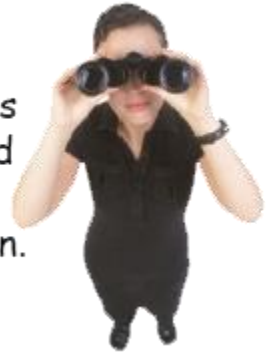
Get off your butt and move around a little

Create an app that tracks your movements in real time.

Browser

1. Your app calls watchPosition, passing in a success handler function.

2. watchPosition sits in the background and constantly monitors your position.



3. When your position changes, watchPosition calls your success handler function to report your new position.

4. watchPosition continues to monitor your position (and report it to your success handler) until you clear it by calling clearWatch.

"Wherever you go, there you are"

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>Wherever you go, there you are</title>
  <script src="myLoc.js"></script>
  <link rel="stylesheet" href="myLoc.css">
</head>
<body>
  <form>
    <input type="button" id="watch" value="Watch me">
    <input type="button" id="clearWatch" value="Clear watch">
  </form>
  <div id="location">
    Your location will go here.
  </div>
  <div id="distance">
    Distance from WickedlySmart HQ will go here.
  </div>
  <div id="map">
  </div>
</body>
</html>
```

We update the HTML to add a form and two buttons: one to start watching your position and one to stop.

Why do we need the buttons?

- users don't want to be tracked all the time.
- another reason:

Handlers and Buttons

```
var watchId = null;

function watchLocation() {
        watchId = navigator.geolocation.watchPosition(
                displayLocation, displayError)
}
```

Success handler

Fail handler

Options

3g
Wifi
GPS

```
var options = {
    enableHighaccuracy: false,
    timeout: Infinity,
    maximumage: 60.000
}
```

```
navigator.geolocation.getCurrentPosition( displayLocation, displayError, options);
```

Scroll the Map

```
function scrollMapToPosition(coords) {
        var latitude = coords.latitude; var longitude = coords.longitude;
        var latlong = new google.maps.LatLng(latitude, longitude);
        map.panTo(latlong);
}
```

Where's Pisa?

```
var cooPisa = new google.maps.LatLng(43.718597, 10.422291);
addMarker(map, cooPisa, title, "Qui c'è il Cnr!");
```

http://wickedlysmart.com/hfhtml5/chapter5/watchmepan/myLoc.html

Summing up

- Geolocation is in HTML5 specs
- Several ways to determine location
- Main methods: getCurrentPosition, watchPosition, clearWatch
- Google Maps API

«That's all folks!»