

GEOLOCATION

Why to use geolocation

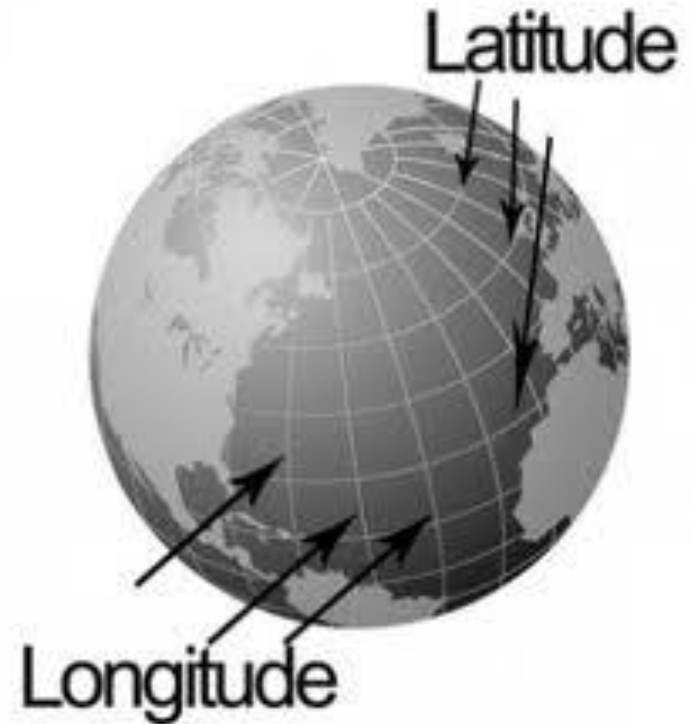
Knowing where your users are can add a lot to a web experience:

- 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.

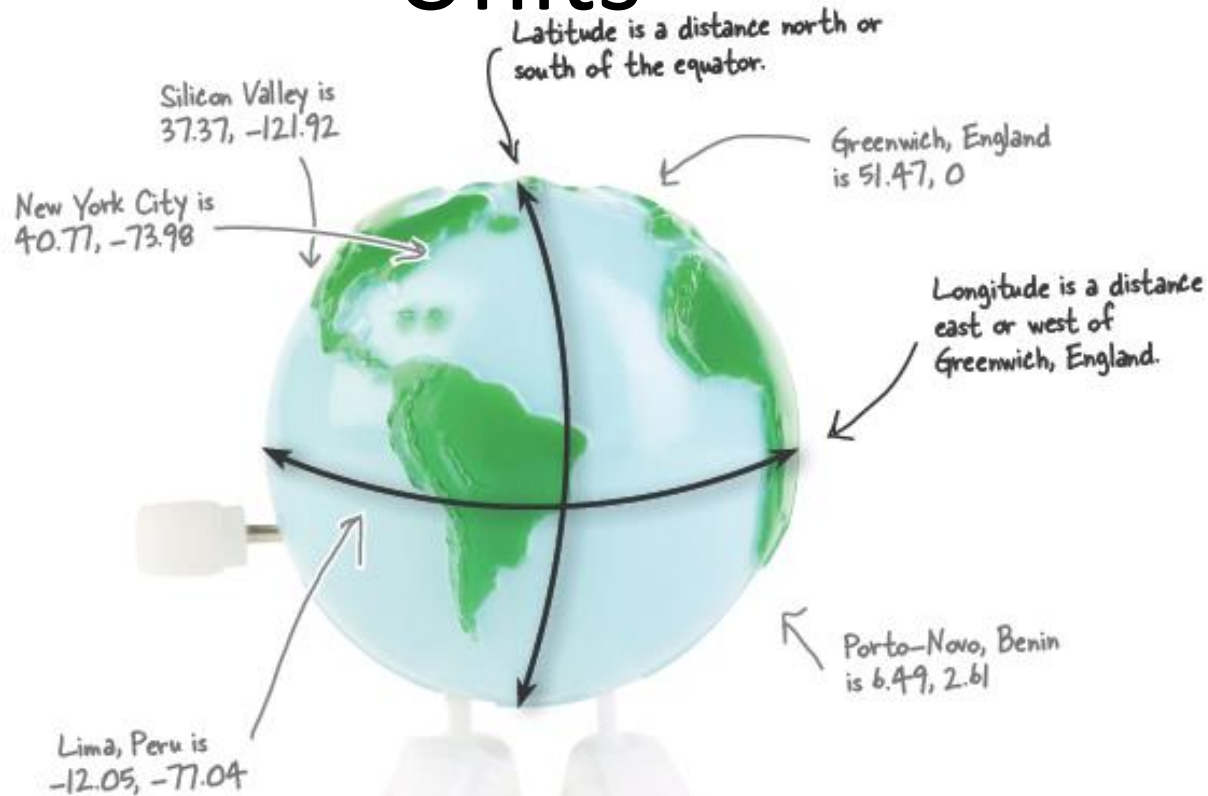
With HTML5 (and the Geolocation JavaScript-based API) you can easily access location information in your pages

Coordinates

- To know where you are, you need a coordinate system on the Earth's surface: latitude and longitude
- Latitude specifies a north/south 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, in terms of latitude and longitude



Units



- You've probably seen latitude and longitude specified in degrees/ minutes/ seconds, such as (47°38'34", 122°32'32"), and in decimal values, such as (47.64, -122.54).
- With the Geolocation API we always use decimal values. To convert from degrees to decimals:

```
function degreesToDecimal(degrees, minutes, seconds) {  
    return degrees + (minutes / 60.0) + (seconds / 3600.0);  
}
```

How browsers determine location

- Browsers are using different ways to determine where you are, some more accurate than others.

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 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.

IP Address

- Location information based on your IP address uses an external database to map the IP address to a physical location.
- The advantage of this approach is that it can work anywhere; however, often IP addresses are resolved to locations such as your ISP's local office.
- Think of this method as being reliable to the city or sometimes neighborhood level.

Cell phone triangulation

- Cell phone triangulation figures out your location based on your distance from one or more cell phone towers (the more towers, the more accurate the location).
- This method can be fairly accurate and works indoors (unlike GPS); it also can be much quicker than GPS.
- Then again, if you're in the middle of nowhere with only one cell tower, your accuracy is going to suffer

WiFi

- 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.

What method is my browser using?

- The browser can use *any* of these means to determine your location.
- In fact, a smart browser might first use cell phone triangulation, if it is available, to give you a rough idea of location, and then later give you a more accurate location with WiFi or GPS.
- Based on the accuracy, you can determine how useful the location is going to be for you

Browser support

Well supported by all modern browsers:

Internet Explorer 9, Firefox, Chrome, Safari and
Opera

Adding geolocation: HTML page

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Where am I?</title>
    <script src="myLoc.js"></script>
    <link rel="stylesheet" href="myLoc.css">
  </head>
  <body>
    <div id="location">
      Your location will go here.
    </div>
  </body>
</html>
```

Adding geolocation: JavaScript

```
window.onload = getMyLocation;
```

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

Handler: **displayLocation**

```
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 + "  
        Longitude: " + longitude;  
}
```

Test drive

- [link](#)


Geolocation API

navigator.

geolocation.

```
getCurrentPosition(successHandler,  
                    errorHandler, options)
```

Position is an object that is passed into your success handler by the geolocation API



```
function displayLocation(position) {  
}
```


Error handler

```
function displayError(error)
{
    var errorTypes = {
        0: "Unknown error",
        1: "Permission denied",
        2: "Position is not available",
        3: "Request timeout"    };

    var errorMessage = errorTypes[error.code];
    if (error.code == 0 || error.code == 2) {
        errorMessage = errorMessage + " " + error.message; }

    var div = document.getElementById("location");
    div.innerHTML = errorMessage;
}
```

Test: [link](#)

To test on mobile device: put your files on the server.

Try:

[http://csci.viu.ca/~barskym/teaching/WP2013/
GEOLOCATION_LECTURE/latlong/myLoc.html](http://csci.viu.ca/~barskym/teaching/WP2013/GEOLOCATION_LECTURE/latlong/myLoc.html)

Calculating distance between two locations

```
function degreesToRadians(degrees) {  
  var radians = (degrees * Math.PI)/180;  
  return radians;  
}
```

- To compute the distance between two points on a sphere use the Haversine equation:

```
function computeDistance(startCoords, destCoords) {  
  var startLatRads = degreesToRadians(startCoords.latitude);  
  var startLongRads = degreesToRadians(startCoords.longitude);  
  var destLatRads = degreesToRadians(destCoords.latitude);  
  var destLongRads = degreesToRadians(destCoords.longitude);  
  var radius = 6371; // radius of the Earth in km  
  
  var distance = Math.acos(Math.sin(startLatRads) * Math.sin(destLatRads) +  
    Math.cos(startLatRads) * Math.cos(destLatRads) *  
    Math.cos(startLongRads - destLongRads)) * radius;  
  return distance;  
}
```

Test: [link](#)

Quiz: What does this code do

```
if (km < 0.1) {  
    distance.innerHTML = "You're on fire!";  
} else {  
    if (prevKm < km) {  
        distance.innerHTML = "You're getting hotter!";  
    } else {  
        distance.innerHTML = "You're getting colder...";  
    }  
}  
prevKm = km;
```

Visualizing location: Google map API

- Geolocation API is simple—it gives you a way to find 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. Google Maps is by far the most popular tool for doing that.

Add script and map element

```
<script  
src="http://maps.google.com/maps/api/js?sensor=true">  
</script>
```

```
<body>
```

```
...
```

```
  <div id="map">
```

```
  </div>
```

```
</body>
```

```
</html>
```

Setting map parameters

```
var googleLatAndLong = new
    google.maps.LatLng(latitude, longitude);

var mapOptions = {
    zoom: 10,    //0 -21
    center: googleLatAndLong,
    mapTypeId: google.maps.MapTypeId.ROADMAP
                //SATELLITE, HYBRID
};
```

Creating a map

```
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);  
}
```

Another constructor from Google,
which takes as parameters an
HTML element and map options

Test: [link](#)

Sticking a pin in it

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

Adding a marker with a pop-up information window

```
function addMarker(map, latlong, title, content) {  
...  
    var infoWindowOptions = {  
        content: content,  
        position: latlong  
    };  
    var infoWindow = new  
        google.maps.InfoWindow(infoWindowOptions);  
    google.maps.event.addListener  
        (marker, "click", function() {  
            infoWindow.open(map);  
        });  
}
```

Add to showMap

```
var title = "Your Location";  
var content = "You are here: " + coords.latitude + ", " +  
              coords.longitude;  
addMarker(map, googleLatAndLong, title, content);
```

Test: [link](#)

More Google maps API

- **Controls:** By default, your Google map includes several controls, like the zoom control, the pan 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.
- **Overlays:** Overlays provide another view on top of a Google map; say, a heat map overlay. If you're commuting, you can check traffic congestion with the traffic overlay. You can create custom overlays, like custom markers, your photos, and pretty much anything else you can imagine, using the Google Maps overlay APIs.
- **Services:** Ever looked up directions in Google Maps? If so, then you've used the Directions service. You have access to directions, as well as other services, like distance and street view through the Google Maps services APIs.

<http://code.google.com/apis/maps/documentation/javascript/>

Geolocation object

geolocation

getCurrentPosition
watchPosition
clearWatch

getCurrentPosition(successHandler, errorHandler, positionOptions)

passes

position

coords
timestamp

Coordinates

latitude
longitude
accuracy

altitude
altitudeAccuracy
heading
speed

These may be not supported by your device

Accuracy

- Finding your location isn't an exact science. Depending on the method the browser uses, you may know only the state, city, or city block you're on.
- With more advanced devices you might know your location to within 10 meters, complete with your speed, heading and altitude.

```
div.innerHTML = "You are at Latitude: " + latitude  
                + ", Longitude: " + longitude;  
div.innerHTML += " (with " + position.coords.accuracy  
                + " meters accuracy)";
```

Test: [link](#)

geolocation

getCurrentPosition

watchPosition

clearWatch

watchPosition method

The watchPosition method looks just like the getCurrentPosition method, but behaves a little differently: it repeatedly calls your success handler each time your position changes.

Tracking position

```
if (navigator.geolocation) {  
    navigator.geolocation.getCurrentPosition();  
    var watchButton =  
        document.getElementById("watch");  
    watchButton.onclick = watchLocation;  
    var clearWatchButton =  
        document.getElementById("clearWatch");  
    clearWatchButton.onclick = clearWatch;  
}  
else {  
    alert("Oops, no geolocation support");  
}
```


On button clicks: watch location, or clear watch

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

function clearWatch() {
    if (watchId) {
        navigator.geolocation.clearWatch(watchId);
        watchId = null;
    }
}
```

Test (on smart phone):

http://csci.viu.ca/~barskym/teaching/WP2013/GEOLOCATION_LECTURE/watchmepan/myLoc.html

The world of timeout and maximum age

- Timeout

This option tells the browser how **long** it gets to determine the user's location. Note that if the user is prompted to approve the location request, the timeout doesn't start until then. If the browser can't determine a new location within the number of milliseconds specified in the timeout, the error handler is called. *By default, this option is set to Infinity.*

- MaximumAge

This option tells the browser how **old** the location can be. So, if the browser has a location that was determined 60 seconds ago, and maximumAge is set to 90000 (90 seconds), then a call to `getCurrentPosition` would return the existing, cached position (the browser would not try to get a new one). But if the maximumAge was set to 30 seconds, the browser would be forced to determine a new position.

Quiz: Who does what

<p>{maximumAge:600000}</p> <p>A</p>	<p>I want only cached positions less than 10 minutes old. If there aren't any cached positions less than 10 minutes old, I ask for a new position, but only if I can get one in 1 second or less</p>	<p>1</p>
<p>{timeout:1000, maximumAge:600000}</p> <p>B</p>	<p>I'll use a cached position if the browser has one that's less than 10 minutes old, otherwise, I want a fresh position.</p>	<p>2</p>
<p>{timeout:0, maximumAge:Infinity}</p> <p>C</p>	<p>I want only fresh positions. The browser can take as long it wants to get me one.</p>	<p>3</p>
<p>{timeout:Infinity, maximumAge:0}</p> <p>D</p>	<p>I want only cached positions. I'll take one of any age. If there is no cached position at all, then I call the error handler. No new positions for me! I'm for offline use.</p>	<p>4</p>

If your location is not displayed

- That is most likely because of the infinite timeout. In other words the browser will wait forever to get a location as long as it doesn't encounter some error condition.
- Now you know how to fix that, because we can force the Geolocation API to be a little more expedient by setting its timeout value.