Lecture 6
Making Your HTML Location Aware: Geolocation [2]

Samkeun Kim <skim@hknu.ac.kr>

http://cyber.hknu.ac.kr/

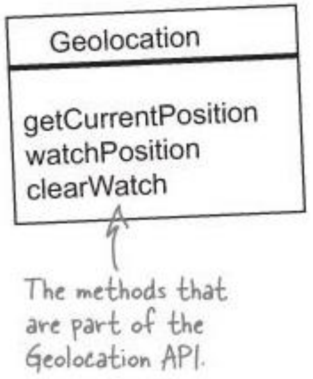
국립한경대학교
HANKYONG NATIONAL UNIV.

Meanwhile back at the Geolocation API…

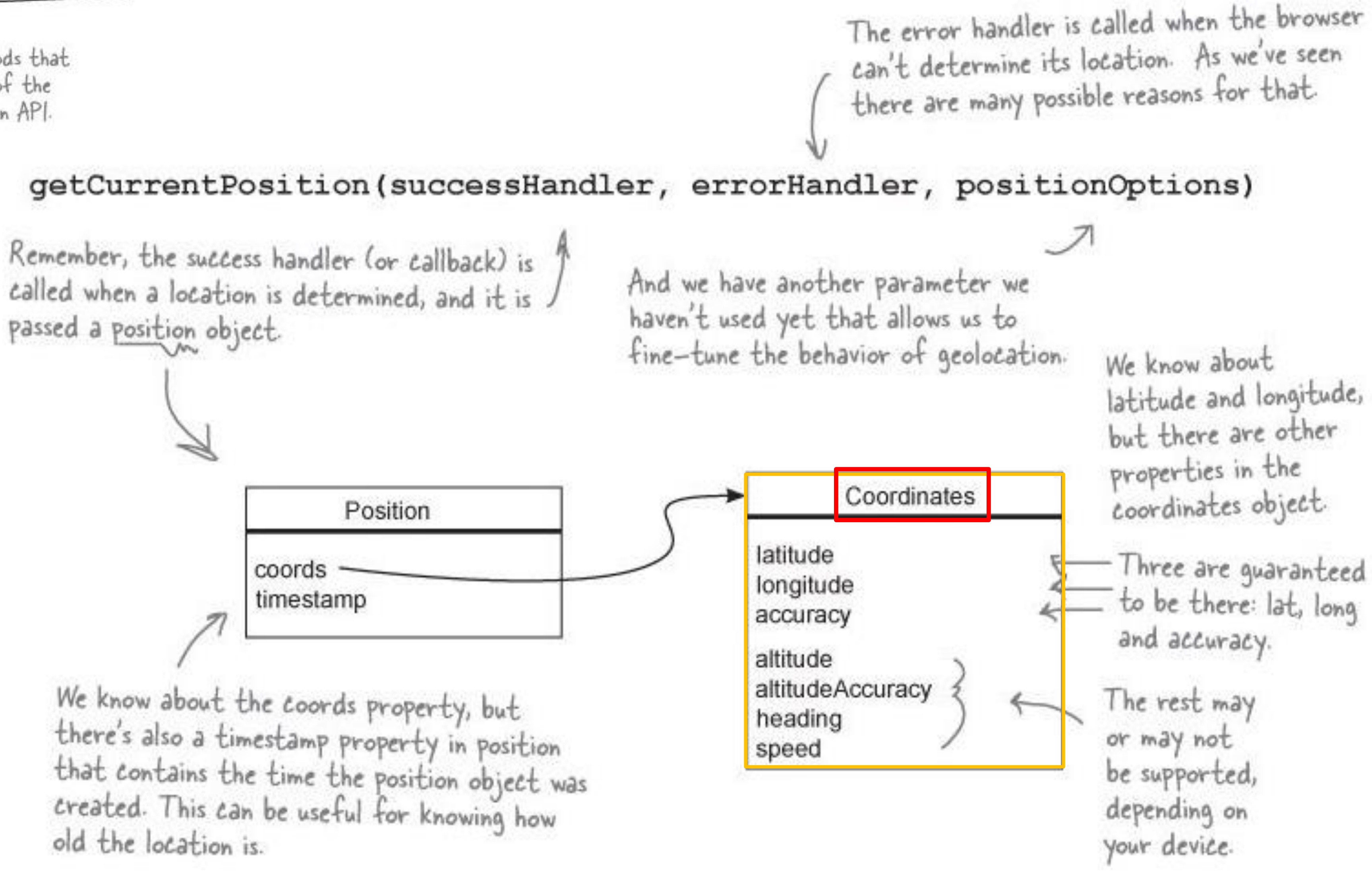
Geolocation API is actually really simple, having **just three** methods:

- getCurrentPosition
- **watchPosition**
- **clearWatch**

Let's take another look at **getCurrentPosition** and at some related objects, like the **Position** and **Coordinates** objects.



The methods that are part of the Geolocation API.



```
getCurrentPosition(successHandler, errorHandler, positionOptions)
```

The error handler is called when the browser can't determine its location. As we've seen there are many possible reasons for that.

Remember, the success handler (or callback) is called when a location is determined, and it is passed a position object.

And we have another parameter we haven't used yet that allows us to fine-tune the behavior of geolocation.

We know about latitude and longitude, but there are other properties in the coordinates object.

Three are guaranteed to be there: lat, long and accuracy.

The rest may or may not be supported, depending on your device.

We know about the coords property, but there's also a timestamp property in position that contains the time the position object was created. This can be useful for knowing how old the location is.

Can we talk about your accuracy?

Depending on the method the browser uses, you might know your location to **within 10 meters**, complete with your speed, heading and altitude.

The **Geolocation API gives us the accuracy, in meters**, of the location, to within a **95% confidence level.**

The accuracy information is part of the **coordinates** object.

Let's pull it out and use it in the displayLocation function.

Can we talk about your accuracy?

```
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;
    div.innerHTML += " (with " + position.coords.accuracy + " meters 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);
}
```

Here we use the accuracy property of position, and append onto the end of the <div>'s innerHTML.

실습과제 6-1 Accuracy Test

Make sure you've got this one liner added to your code, and load the page.

Now you can see how accurate your location is. Be sure to try this on any device you have.

https://ksamkeun.000webhostapp.com/html5/ch5/accuracy/myLoc.html

스마트폰으로 테스트 해보기!!

“Wherever you go, there you are”

The origin of this phrase has been hotly debated.

Some claim the first real mention of it was in the film Buckaroo Banzai, others draw its origin from Zen Buddhist text, still others cite various books, movies and popular songs.

No matter the source, it’s here to stay, and even more so after this chapter because we’re going to turn it into a little web app named **“Wherever you go, there you are.”**

Yes, there is an app for that!

But, we’re going to need a little participation from you, the reader, because for this one you’ll have to (excuse us for saying this) get off your butt and move around a little.

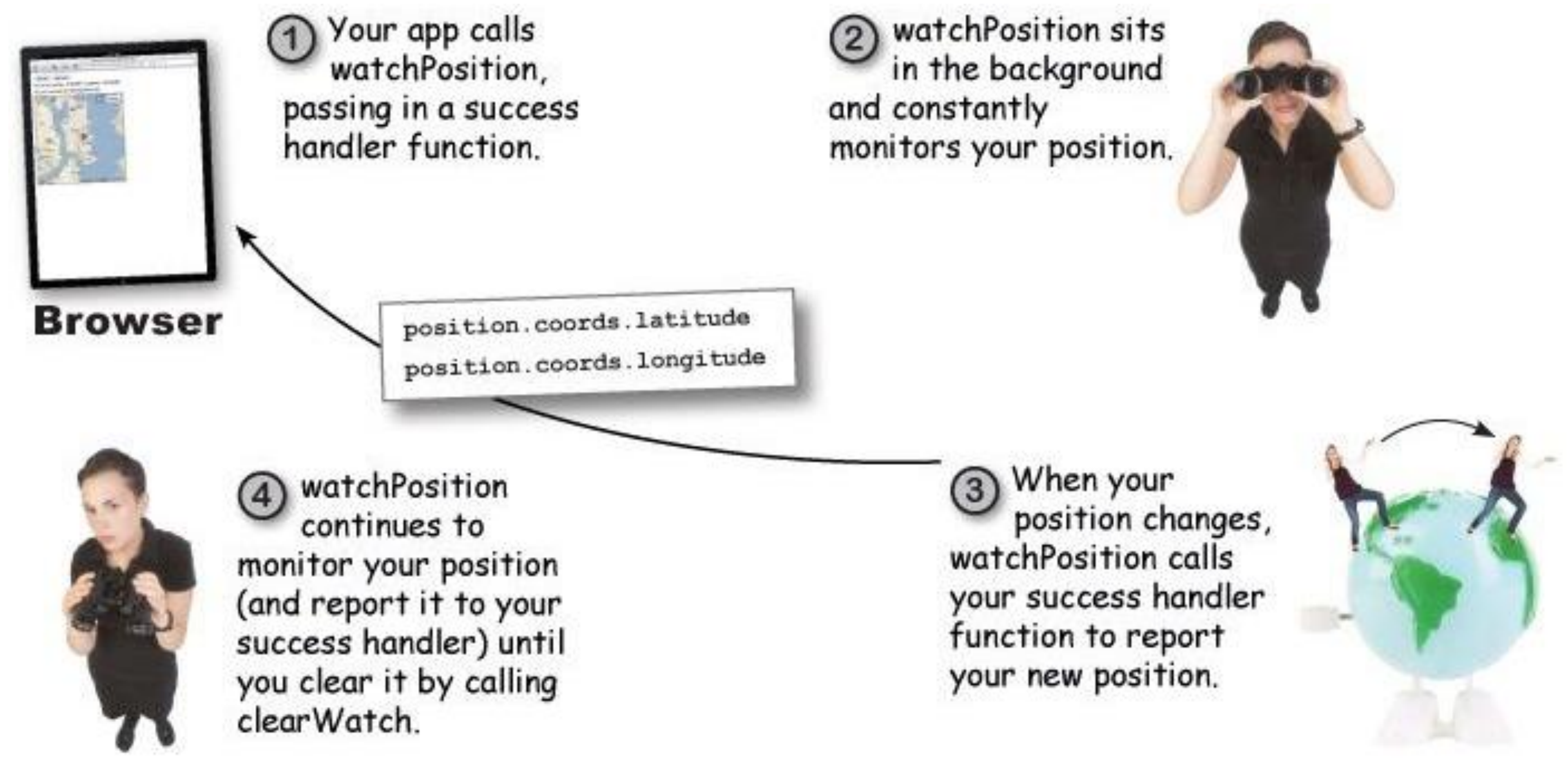
What we’re going to do is extend our current code so that it **tracks your movements in real time.**

To do that we’re going to bring everything together, including the last two methods in the Geolocation API, and create an app that tracks you, in near real time.

How we're going to track your movements

Geolocation API has a watchPosition method

- 사용자의 움직임을 watching하고 있다가 그 위치가 바뀌면 바뀐 위치를 기록한다
- 위치가 바뀔 때마다 반복적으로 success handler를 호출한다


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.
position.coords.latitude
position.coords.longitude

Getting the app started

위치 트래킹을 시작시키고 스톱시킬 수 있는 두 개의 버튼을 추가해 보자.

Why do we need the buttons?

Two reasons:

1. Users don't want to be tracked all the time and they usually **want some control** over that.
2. Constantly checking your position is an **energy-intensive operation** on a mobile device and if it's left on all the time, it will cause your battery life to suffer.

So, first, we'll update the HTML to add a form and two buttons:

- **one to start watching your position**
- **one to stop**

Tracking a user in real time can be a real battery drainer. Make sure you give the user information about their tracking, and some control over it, too.

Getting the app started

```
<script src="https://maps.google.com/maps/api/js?key=YOUR API KEY"></script>
```

```
<!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're adding a form element with two buttons, one to start the watch, with an id of "watch", and one to clear the watch, with an id of "clearWatch".

We're going to reuse our old <div>s to report on the real-time location information.

We'll come back and worry about the Google map in a bit...

Reworking our old code…

두 개의 버튼 클릭에 대한 **두 개의 핸들러**를 만들어 보자:

- **watchLocation** for the watch button
- **clearWatch** for the clear button

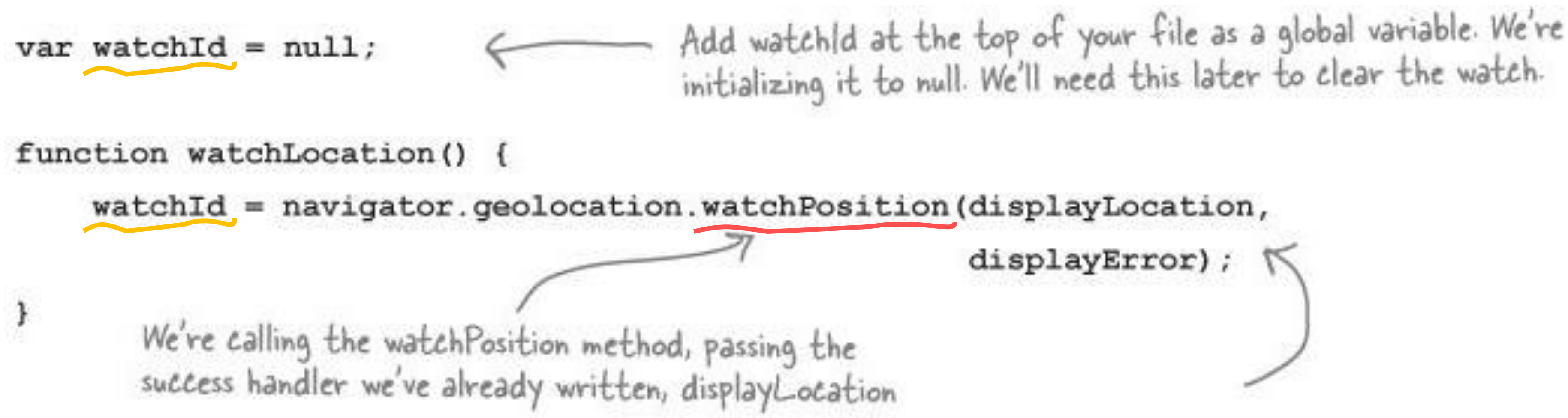
If the browser supports geolocation, we'll add our button click handlers. No point in adding them if geolocation isn't supported.

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

We're going to call watchLocation to start the watch, and clearWatch to stop it.

Writing the **watchLocation** handler

- Uses the **geolocation.watchPosition** method to start watching their position
- Has two parameters, a **success** handler and an **error** handler
- Returns a watchId (global variable)



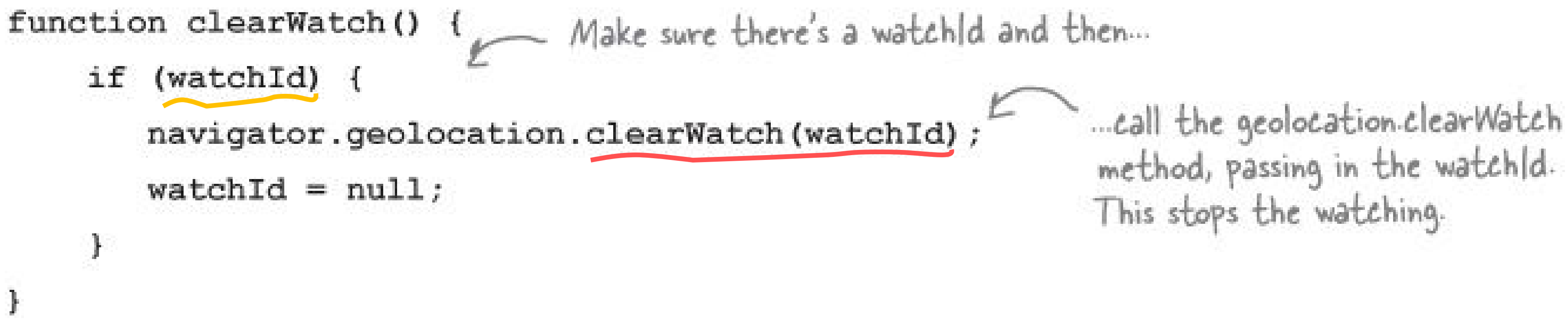
```
var watchId = null;

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

Add watchId at the top of your file as a global variable. We're initializing it to null. We'll need this later to clear the watch.

We're calling the watchPosition method, passing the success handler we've already written, displayLocation

Writing the **clearWatch** handler

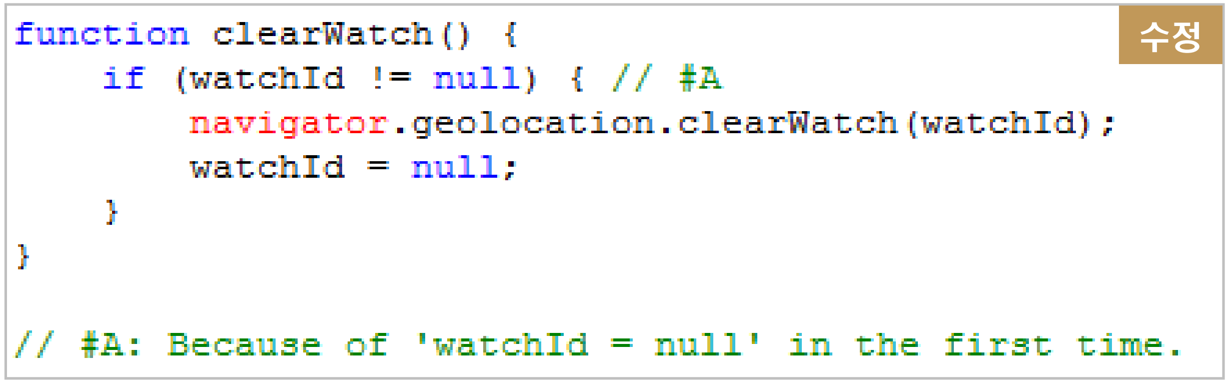
- Now let's write the handler to **clear** the watching activity
- Need to take the watchId and pass it to the **geolocation.clearWatch** method



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

Make sure there's a watchId and then...

...call the geolocation.clearWatch method, passing in the watchId. This stops the watching.



수정

```
function clearWatch() {
    if (watchId != null) { // #A
        navigator.geolocation.clearWatch(watchId);
        watchId = null;
    }
}

// #A: Because of 'watchId = null' in the first time.
```

We still need to make a small update to **displayLocation**…

- Calls **showMap**
- Otherwise, **showMap has already been called and we don't need to call it again**

```
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;
    div.innerHTML += " (with " + position.coords.accuracy + " meters accuracy)";

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

    if (map == null) {
        showMap(position.coords);
    }
}
```

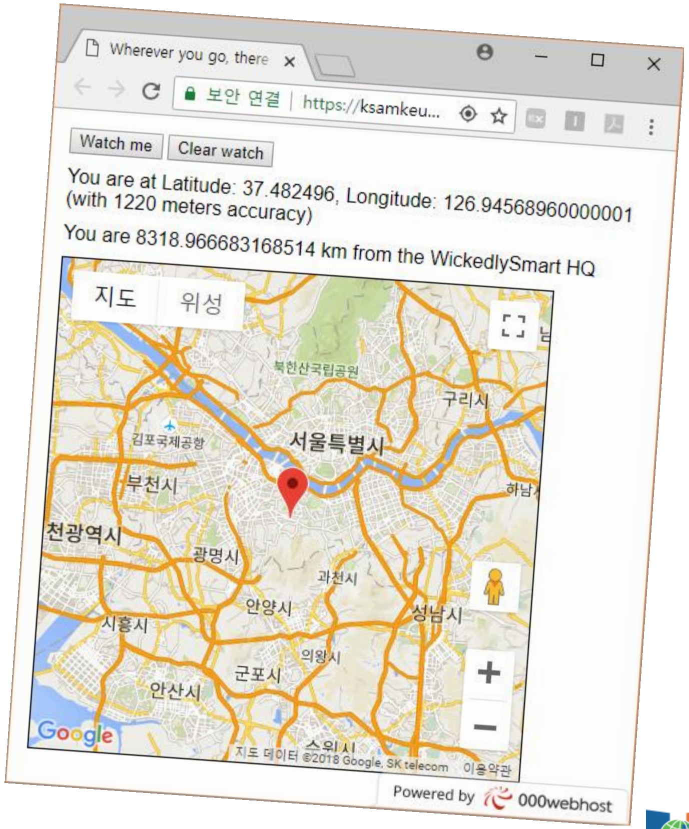
If we haven't called showMap already, then call it, otherwise we don't need to call it every time displayLocation is called.

실습과제 6-2 Time to get moving!

Make sure you've got all the new code typed in and reload your page, **myLoc.html**. Now, to truly test this page you're going to need to "relocate" to have your position updated. So take a walk, **jump on your bike**, get in the car, or use whatever your favorite mode of transportation might be.

It almost goes without saying that if you're running this on your desktop, this app is going to be pretty boring (since you can't take it with you), so you really need to use a mobile device for this test. And, if you need help getting to a hosted version with your mobile device, we've placed a copy of this code at:

https://ksamkeun.000webhostapp.com/html5/ch5/watchme/myLoc.html



You are at Latitude: 37.482496, Longitude: 126.94568960000001 (with 1220 meters accuracy)

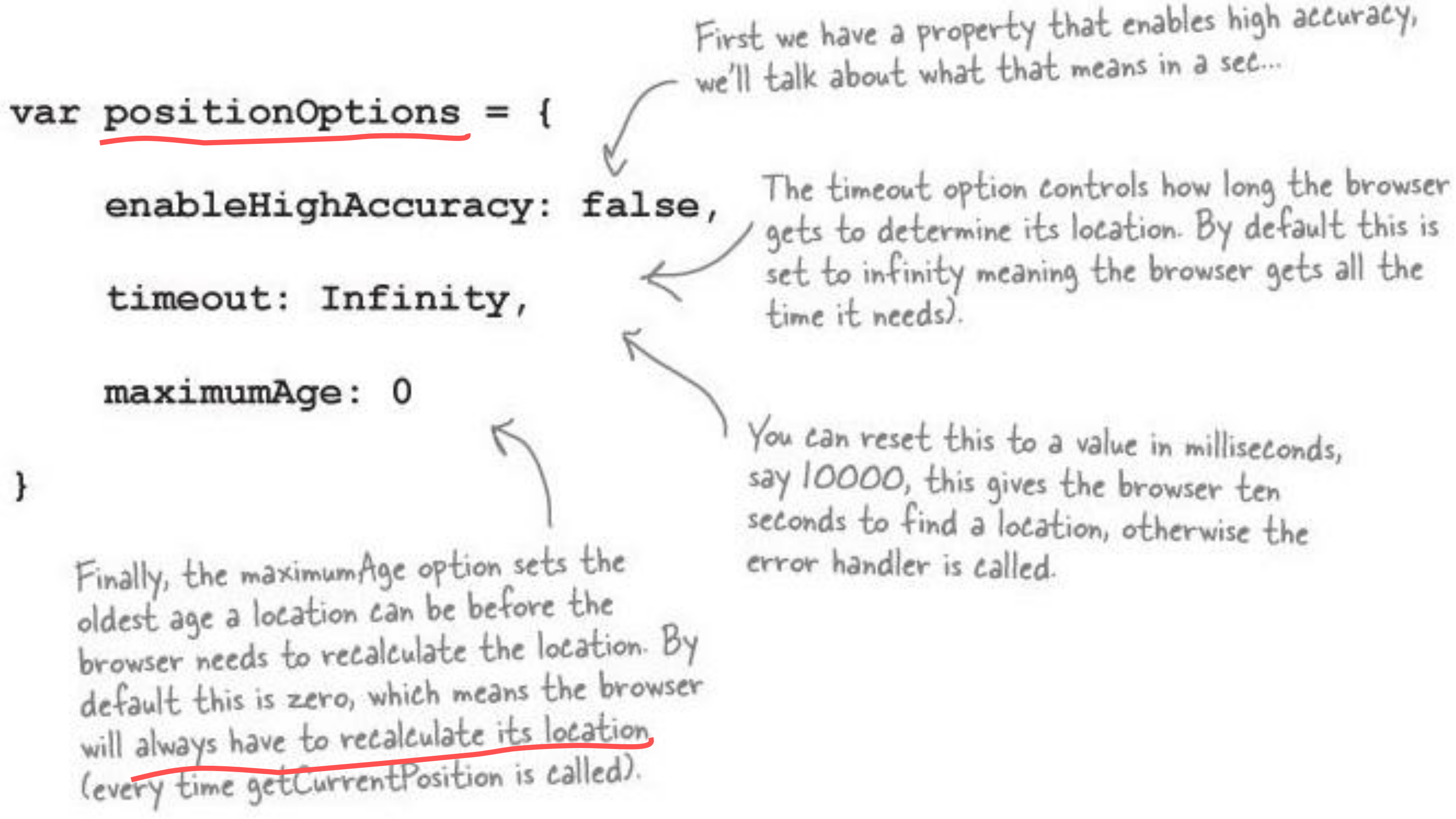
You are 8318.966683168514 km from the WickedlySmart HQ

You've got some Options…

Third parameter of getCurrentPosition: **positionOptions**

- ✓ 위치(경위도) 값을 계산하는 방법을 제어할 수 있다

Let's look at the **three parameters** along with their default values:



```
var positionOptions = {

    enableHighAccuracy: false,

    timeout: Infinity,

    maximumAge: 0

}
```

First we have a property that enables high accuracy, we'll talk about what that means in a sec…

The timeout option controls how long the browser gets to determine its location. By default this is set to infinity meaning the browser gets all the time it needs).

You can reset this to a value in milliseconds, say 10000, this gives the browser ten seconds to find a location, otherwise the error handler is called.

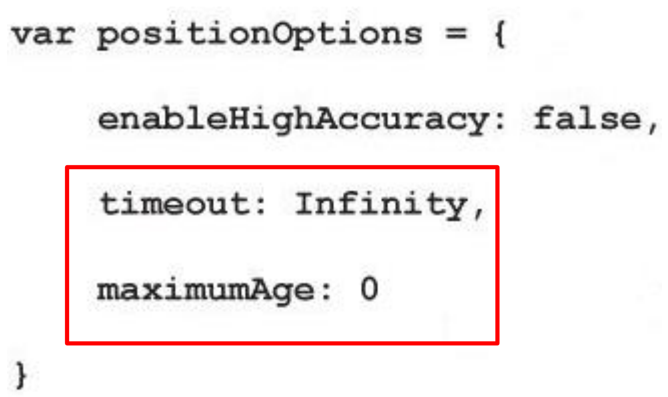
Finally, the maximumAge option sets the oldest age a location can be before the browser needs to recalculate the location. By default this is zero, which means the browser will always have to recalculate its location (every time getCurrentPosition is called).

Can we talk about your accuracy, again?

- Geolocation API has an accuracy property
- 정밀도(accuracy)는 브라우저 구현 방식에 따라 달라질 수 있다.
- 정밀도와 전력소비 간에 약간의 거래(deal)가 있을 수 있다.
- 높은 정밀도를 요구하지 않는 경우라면 빠르고 적은 전력소비로 응답할 수 있다

  예: 앱 사용자가 '서울'에 있다.
- 그러나 사용자가 위치해 있는 거리(street)를 알 필요가 있다면 그 정보를 얻기 위해 API는 GPS를 켜고 많은 전력소비를 감수해야 한다.
- enableHighAccuracy 옵션을 이용하면 얻을 수 있는 가장 정확한 위치를 얻을 수 있다.
- 명심해야 할 사항은 이 옵션은 브라우저가 더 정확한 위치 정보를 제공할 수 있음을 보장해 주지 않는다는 사실이다.

The world of timeouts and maximum age…



```
var positionOptions = {
    enableHighAccuracy: false,
    timeout: Infinity,
    maximumAge: 0
}
```

timeout:

- 사용자의 위치를 결정하는데 얼마의 시간이 걸릴지를 브라우저에게 말해준다
- 브라우저가 이 옵션(timeout)에 지정된 milliseconds 시간 안에 새로운 위치를 결정할 수 없으면 에러 핸들러가 호출된다
- By default, this option is set to Infinity

maximumAge:

- 위치가 얼마나 오래 보존될 수 있는 지를 브라우저에게 말해준다
- 그래서 브라우저가 위치를 결정하는데 60초 걸렸고 maximumAge가 90000(90초)으로 설정되어 있다면 getCurrentPosition의 호출은 기존의 캐시에 있는 위치를 반환할 것이다
- But if the maximumAge was set to **30** seconds, the browser would be forced to determine a **new position.**

How to specify options

enableHighAccuracy와 maximumAge를 60초로 설정하는 예:

```
var options = {enableHighAccuracy: true, maximumAge: 60000};
```

Are you starting to see that JavaScript really rocks? Well, at least we think it does.

Options를 getCurrentPosition 또는 watchPosition에 전달한다:

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

Here, we're just passing our options along using the options variable.

Or, options 객체를 inline으로 작성할 수도 있다:

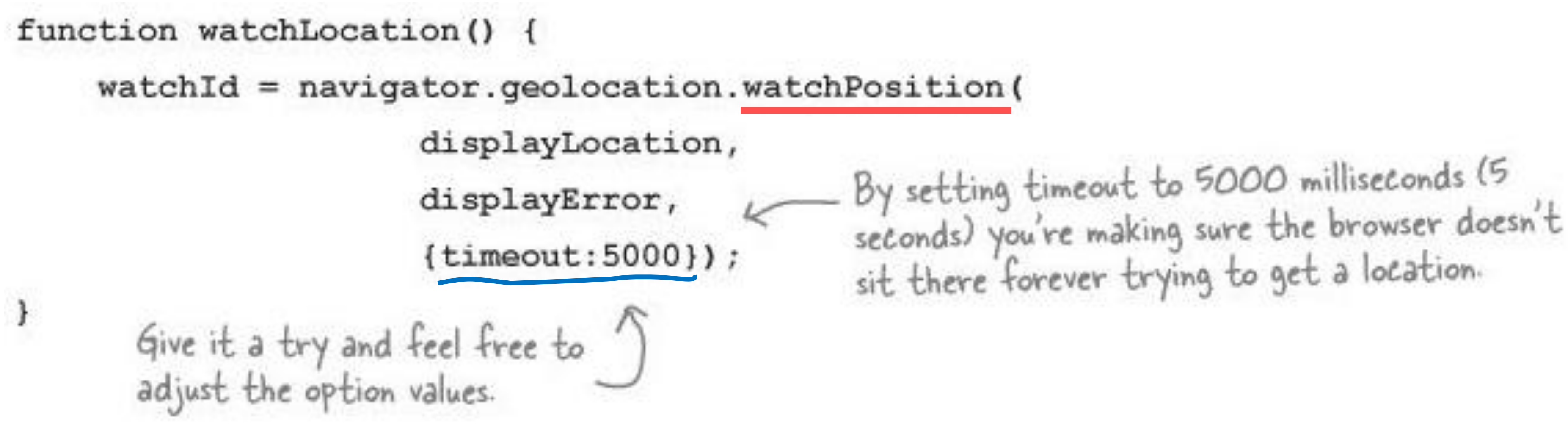
```
navigator.geolocation.getCurrentPosition(
                        displayLocation,
                        displayError,
                        {enableHighAccuracy: true, maximumAge: 60000});
```

You'll see this technique used a lot in JavaScript code.

Here are the options, written as a literal object right in the function call! Some would argue this is easier and more readable as code.

Test Drive Diagnostics Checkup

When you ran the diagnostics before, did you get the test case where you waited and waited and nothing happened? That's 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. Well, 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.** Here's how:



```
function watchLocation() {
   watchId = navigator.geolocation.watchPosition(
                      displayLocation,
                      displayError,
                      {timeout:5000});
}
```

By setting timeout to 5000 milliseconds (5 seconds) you're making sure the browser doesn't sit there forever trying to get a location.

Give it a try and feel free to adjust the option values.

TRY THIS AT HOME (PUSHING GEO TO THE LIMIT)

device here

time here

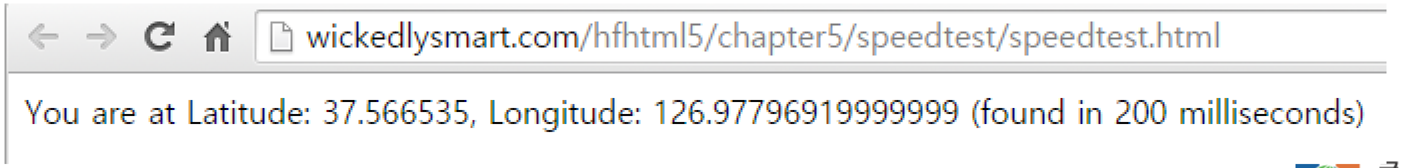
ON ________ FOUND IN ______________ milliseconds

ON ________ FOUND IN ______________ milliseconds

ON ________ FOUND IN ______________ milliseconds

ON ________ FOUND IN ______________ milliseconds

https://wickedlysmart.com/hfhtml5/chapter5/speedtest/speedtest.html



wickedlysmart.com/hfhtml5/chapter5/speedtest/speedtest.html

You are at Latitude: 37.566535, Longitude: 126.97796919999999 (found in 200 milliseconds)

Let's finish this app!

사용자가 움직일 때 맵이 사용자의 위치를 중심으로 유지하는 함수를 작성해 보자:

✓ **Drops a new marker** each time we get a new position

Okay, we're going to call this function scrollMapToPosition and we're going to pass it a position's coordinates.

The coordinates are going to be your latest new position, so we're going to center the map on that location, and drop a marker there too.

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

    map.panTo(latlong);

    addMarker(map, latlong, "Your new location", "You moved to: " +
                                latitude + ", " + longitude);
}
```

First let's grab the new lat and long, and create a google.maps.LatLng object for them.

The panTo method of the map takes the LatLng object and scrolls the map so your new location is at the center of the map.

Finally, we'll add a marker for your new location using the addMarker function we wrote earlier, passing in the map, the LatLng object, a title and some content for the new marker.

Integrating our new function

위치가 변할 때마다 **scrollMapToPosition**을 호출하도록 **displayLocation** 함수를 업데이트 한다:

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

    if (map == null) {
        showMap(position.coords);
    } else {
        scrollMapToPosition(position.coords);
    }
}
```

The first time displayLocation is called, we need to draw the map and add the first marker.

After that, all we need to do is add a new marker to the existing map.

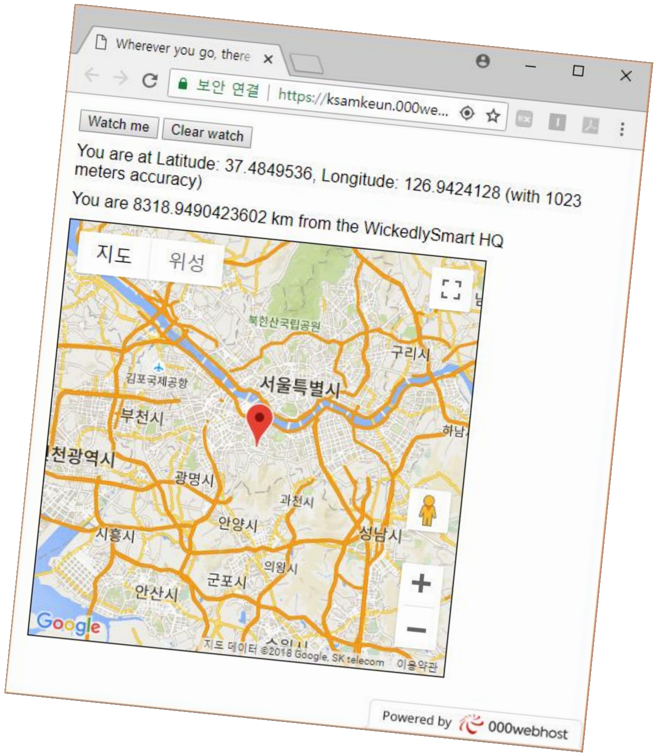
실습과제 6-3 And one more time...

Reload your page and start moving around... is your map following you? You should see a trail of markers being added to your map as you move (unless you're sitting at your desktop!).

So, we submit this application as solid proof that "wherever you go, there you are."

[자신의 스마트폰 이용] 학교와 집을 오갈 때 추적 경로(일부분만)가 나타나도록 하시오.

https://ksamkeun.000webhostapp.com/html5/ch5/watchmepan/myLoc.html



You are at Latitude: 37.4849536, Longitude: 126.9424128 (with 1023 meters accuracy)

You are 8318.9490423602 km from the WickedlySmart HQ

실습과제 6-4

Google Maps JavaScript API Code Samples:
https://developers.google.com/maps/documentation/javascript/examples/

위 사이트 예제 중 '지도에 그리기(Drawing on the Map)' 섹션에서 적어도 10개 이상의 예제를 선택하여 수행하시오.

(오른쪽 마우스 클릭 > 한국어로 번역 선택!)

Q & A