Geolocation

Is the art of figuring out where you are in the world and optionally sharing that information.

- your IP address
- your wireless network connection
- which cell tower your phone is talking to
- dedicated GPS hardware that calculates latitude and longitude from information sent by satellites in the sky

Geolocation

Privacy concern

permission of the user – browsers, apps, etc.

Geolocation API

Lets you share your location with trusted web sites

Support

IE	Firefox	Safari	Chrome	Opera	iPhone	Android
9.0+	3.5+	5.0+	5.0+	10.6+	3.0+	2.0+

Geolocation API

The geolocation object

The geolocation API is published through the navigator.geolocation object.

If the object exists, geolocation services are available

Geolocation

```
if ("geolocation" in navigator) {
            /* geolocation is available */
      } else {
            alert("geolocation IS NOT available. geolocation
            services are not supported by this browser.");
}
```

Geolocation API

Documentation link by MDN

https://developer.mozilla.org/en-US/docs/WebAPI/Using_geolocation

W3C Geolocation API Specs

http://dev.w3.org/geo/api/spec -source.html

Geolocation API

```
<body>
<p id="ex0">Click the button to see if geolocation is available: </p>
<button onclick ="getLocation()"> check availability</button>
<script>
function getLocation()
        {
        if ("geolocation" in navigator) {
                /* geolocation is available */
                alert("geolocation IS available.");
        } else {
                alert("geolocation IS NOT available. geolocation services are not
        supported by this browser.");
        }
}
</script> </body> </html>
```

School of Design

getCurrentPosition () method

To obtain the user's current location:

- Use getCurrentPosition () method
- This initiates an asynchronous request to detect the user's position, and queries the positioning hardware to get up -to-date information
- When the position is determined, a specified callback routine is executed
- You can optionally provide a second callback to be executed if an error occurs
- A third parameter (optional) is an options interface where you can set the the time to wait for a request and the maximum age of the position returned

getCurrentPosition () method

```
navigator.geolocation.getCurrentPosition (function(position ) {
ExampleFunction( position.coords.latitude,
                              position.coords.longitude );
});
```

This example will cause the ExampleFunction() to execute when the location is obtained.

getCurrentPosition () method

```
<p id="ex1">Click the button to get your coordinates : </p>
<button onclick ="getLocation()">Try It </button>
<script >
var x=document.getElementById("ex1");
function getLocation()
        {
        if (navigator.geolocation )
        {
        navigator.geolocation.getCurrentPosition(showPosition);
        }
        else{x.innerHTML="Geolocation is not supported by this browser.";}
        }
function showPosition(position)
        {
        x.innerHTML="Latitude:" + position.coords.latitude + "<br> Longitude:" +
position.coords.longitude;
        }
</script>
```

School of Design

getCurrentPosition () method

example2.html

Google Map Call

Documentation

https://developers.google.com/maps/

Google Map Call

HTML5 declaration

```
<!DOCTYPE html>
```

CSS declaration

- the map container <div> (#map-canvas) should take up 100% of the height of the HTML body.
- Note: we must specifically declare those percentages for <body> and <html> as well.

```
<style type="text/css">
  html { height: 100% }
  body { height: 100%; margin: 0; padding: 0 }
  #map-canvas { height: 100% }
</style>
```

Google Map Call

Loading the Google Maps API

```
<script type="text/javascript"
  src="https://maps.googleapis.com/maps/api/js?key=API_KEY&sensor=SET_TO_TRUE_OR_FALSE">
</script>
```

URL of a JavaScript file that loads all of the symbols and definitions you need for using the Google Maps API. This script tag is required.

Google Map Call

```
    <script type="text/javascript"
      src="https://maps.googleapis.com/maps/api/js?
key=API_KEY&sensor=SET_TO_TRUE_OR_FALSE">
    </script>
```

Key contains your application's API key

sensor of the URL must be included, and indicates whether this application uses a sensor (such as a GPS locator) to determine the user's location.

- true
- false

Google Map Call

This code instructs the application to

- load the Maps API after the page has fully loaded (using window.onload)
- write the Maps JavaScript API into a <script> tag within the page

instructs the API to only execute the initialize() function after the API has fully loaded by passing callback=initialize to the Maps API bootstrap

Google Map Call

```
function initialize() {
  var mapOptions = {
    zoom: 8,
    center: new google.maps.LatLng(-34.397, 150.644)
  };
  var map = new google.maps.Map(document.getElementById('map-canvas'),
      mapOptions);
}
function loadScript() {
  var script = document.createElement('script');
  script.type = 'text/javascript';
  script.src = 'https://maps.googleapis.com/maps/api/js?v=3.exp&sensor=false&' +
      'callback=initialize';
  document.body.appendChild(script);
}
window.onload = loadScript;
```

Google Map Call

```
<div id="map-canvas" style="width: 100%; height: 100%"></div>
```

define a <div> named "map-canvas"

set its size using style attributes

- Note that this size is set to "100%" which will expand to fit the size on mobile devices.
- You may need to adjust these values based on the browser's screensize and padding.
- Note that the map will always take its size from that of its containing element, so you must always set a size on that <div> explicitly.

Google Map Call

```
var mapOptions = {
  center: new google.maps.LatLng(-34.397, 150.644),
  zoom: 8
};
```

Map options object contains map initialization variables;

center and zoom

The initial resolution at which to display the map is set by the zoom property,

zoom 0 corresponds to a map of the Earth fully zoomed out

higher zoom levels zoom in at a higher resolution

Google Map Call

```
var mapOptions = {
  center: new google.maps.LatLng(-34.397, 150.644),
  zoom: 8
};
```

Offering a map of the entire Earth as a single image would either require an immense map, or a small map with very low resolution.

As a result, map images within Google Maps and the Maps API are broken up into map "tiles" and "zoom levels." At low zoom levels, a small set of map tiles covers a wide area; at higher zoom levels, the tiles are of higher resolution and cover a smaller area.

Google Map Call

```
var map = new google.maps.Map(document.getElementById("map-canvas"),
    mapOptions);
```

variable (map) – assigned to a new Map object

passing in options defined within the mapOptions object literal

These options will be used to initialize the map's properties.

```
Map(mapDiv:Node,
opts?:MapOptions )
```

Creates a new map inside of the given HTML container — which is typically a DIV element — using any (optional) parameters that are passed.

Google Map Call

```
google.maps.event.addDomListener(window, 'load', initialize);
```

Loading the map

Alternatively:

```
<body onload="initialize()">
```

Google Map Call – example_google.html

```
<script src="https://maps.googleapis.com/maps/api/js?v=3.exp&sensor=false"></
script>
    <script>
var map;
function initialize() {
  var mapOptions = {
    zoom: 8,
    center: new google.maps.LatLng(-34.397, 150.644)
  };
  map = new google.maps.Map(document.getElementById('map-canvas'),
      mapOptions);
}
google.maps.event.addDomListener(window, 'load', initialize);
    </script>
  </head>
  <body>
    <div id="map-canvas"></div>
```

Google Map Call – example_google.html

```
<div id="map-canvas" style="width:500px;height:450px;"></div>
```