```
<!--

 Dan Armendariz
 Computer Science E-76
 Harvard Extension School
 Spring 2011

 A basic implementation of web storage.

-->
<!DOCTYPE html>
<html>
    <head>
        <title>Web Storage Example</title>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <meta name="viewport" content="initial-scale=1.0" />
        <link rel="stylesheet" href="style.css" />

        <script type="text/javascript">
        // <![CDATA[

        // demonstrate some local storage capabilities
        function store_test() {
            // retrieve the value in local storage
            alert(localStorage.getItem("foo"));

            // set some local storage
            localStorage.setItem("foo", "test!");
        }

        // ]]>
        </script>

    </head>

    <body onload="store_test()">

    <header>
        <h1>Web Storage Example</h1>
    </header>

    <article>
        <header>
            <h1>Web storage</h1>
        </header>
```

```

        <p>The contents of the local storage should appear in an alert box.</p>

    </article>

    </body>
</html>
```

```
<!--

 Dan Armendariz
 Computer Science E-76
 Harvard Extension School
 Spring 2011

 A basic implementation of web storage, with some error handling.

-->
<!DOCTYPE html>
<html>
    <head>
        <title>Web Storage Example</title>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <meta name="viewport" content="initial-scale=1.0" />
        <link rel="stylesheet" href="style.css" />

        <script type="text/javascript">
        // <![CDATA[

        // returns true of the web browser supports web storage, false otherwise
        function supports_web_storage() {
            try {
                return 'localStorage' in window && window['localStorage'] !== null;
            } catch (e) {
                return false;
            }
        }

        // demonstrate some local storage capabilities
        function store_test() {
            if(!supports_web_storage()) {
                alert("Your browser doesn't seem to support web storage!");
                return;
            }

            // retrieve the value in local storage
            var data = localStorage.getItem("bar");
            if(data == null)
                alert("Data is null (nothing in storage for the key 'bar').");
            else
                alert(data);

            // set some local storage
```

```
                localStorage.setItem("bar", "test!");
        }

        // ]]>
        </script>

    </head>

    <body onload="store_test()">

    <header>
        <h1>Web Storage Example</h1>
    </header>

    <article>
        <header>
            <h1>Web storage</h1>
        </header>

        <p>The contents of the local storage should appear in an alert box.</p>

    </article>

    </body>
</html>
```

```
<!--

 Dan Armendariz
 Computer Science E-76
 Harvard Extension School
 Spring 2011

 A basic implementation of web storage, now with more integers and different
 ways of accessing data.

-->
<!DOCTYPE html>
<html>
    <head>
        <title>Web Storage Example</title>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <meta name="viewport" content="initial-scale=1.0" />
        <link rel="stylesheet" href="style.css" />

        <script type="text/javascript">
        // <![CDATA[

        // returns true of the web browser supports web storage, false otherwise
        function supports_web_storage() {
            try {
                return 'localStorage' in window && window['localStorage'] !== null;
            } catch (e) {
                return false;
            }
        }

        // demonstrate some local storage capabilities
        function store_test() {
            if(!supports_web_storage()) {
                alert("Your browser doesn't seem to support web storage!");
                return;
            }

            // retrieve the counter from local storage
            var data = localStorage["count"];

            // reset the counter if they've not visited before
            if(data == null)
                data = 0;

```

```
            // tell the user how many times they've visited
            alert("You've visited this page " + data + " times in the past!");

            // set the counter in local storage
            localStorage["count"] = ++data;
        }

        // mess up the counter by replacing it with an actual string
        function messup() {
            localStorage["count"] = "test";

            store_test();
        }

        // ]]>
        </script>

    </head>

    <body onload="store_test()">

    <header>
        <h1>Web Storage Example</h1>
    </header>

    <article>
        <header>
            <h1>Web storage</h1>
        </header>

        <p>The contents of the local storage should appear in an alert box.</p>

        <p><a href="javascript:messup()">Mess up counter</a></p>

    </article>

    </body>
</html>
```

```
<!--

 Dan Armendariz
 Computer Science E-76
 Harvard Extension School
 Spring 2011

 A basic implementation of web storage while protecting the type of data.

-->
<!DOCTYPE html>
<html>
    <head>
        <title>Web Storage Example</title>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <meta name="viewport" content="initial-scale=1.0" />
        <link rel="stylesheet" href="style.css" />

        <script type="text/javascript">
        // <![CDATA[

        // returns true of the web browser supports web storage, false otherwise
        function supports_web_storage() {
            try {
                return 'localStorage' in window && window['localStorage'] !== null;
            } catch (e) {
                return false;
            }
        }

        // demonstrate some local storage capabilities
        function store_test() {
            if(!supports_web_storage()) {
                alert("Your browser doesn't seem to support web storage!");
                return;
            }

            // Notice the type that this (and all other) data is stored:
            alert(typeof localStorage["count"]);

            // retrieve the value from local storage (safer)
            var data = parseInt(localStorage["count"]);

            // reset the counter if they've not visited before
            if(data == null || isNaN(data))
```

```
                data = 0;

            // tell the user how many times they've visited
            alert("You've visited this page " + data + " times in the past!");

            // set the counter in local storage
            localStorage["count"] = ++data;

        }

        // mess up the counter by replacing it with an actual string
        function messup() {
            localStorage["count"] = "test";

            store_test();
        }

        // reset the counter at the user's request.
        function reset() {
            localStorage.removeItem("count");
        }

        // ]]>
        </script>

    </head>

    <body onload="store_test()">

    <header>
        <h1>Web Storage Example</h1>
    </header>

    <article>
        <header>
            <h1>Web storage</h1>
        </header>

        <p>The contents of the local storage should appear in an alert box.</p>

        <p><a href="javascript:messup()">Messup counter</a> | <a href="javascript:reset()">Reset Counter</a></p>

    </article>

    </body>
```

```
</html>
```

```
<!--

 Dan Armendariz
 Computer Science E-76
 Harvard Extension School
 Spring 2011

 A demonstration that local storage is not like cookies: by storing
 lots of data and retrieving without sending to the server.

-->
<!DOCTYPE html>
<html>
    <head>
        <title>Web Storage Example</title>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <meta name="viewport" content="initial-scale=1.0" />
        <link rel="stylesheet" href="style.css" />

        <script type="text/javascript">
        // <![CDATA[

        // returns true of the web browser supports web storage, false otherwise
        function supports_web_storage() {
            try {
                return 'localStorage' in window && window['localStorage'] !== null;
            } catch (e) {
                return false;
            }
        }

        // make sure the browser supports web storage
        function init() {
            if(!supports_web_storage()) {
                alert("Your browser doesn't seem to support web storage!");
                return;
            }

            // obtain data in storage
            fetch();
        }

        // fetch the data from local storage and place into text area
        function fetch() {
            var data = localStorage["MyText"];

            if(data == null) {
                alert("No text in storage!");
                return;
            }

            document.getElementById("txt").value = data;
        }

        // save the text area into storage
        function save() {
            localStorage["MyText"] = document.getElementById("txt").value;
        }

        // remove item from storage
        function reset() {
            localStorage.removeItem("MyText");
        }

        // ]]>
        </script>

    </head>

    <body onload="init()">

    <header>
        <h1>Web Storage Example</h1>
    </header>

    <article>
        <header>
            <h1>Web storage</h1>
        </header>

        <!-- Safari bug: adding the "autofocus" attribute prevents us from writing the value to the text area with
JS -->
        <textarea rows="50" cols="80" id="txt"></textarea><br />

        <input type="button" onclick="save(); return false;" value="Save Data" />

        <input type="button" onclick="fetch(); return false;" value="Fetch Data" />

        <input type="button" onclick="reset(); return false;" value="Reset" />
    </article>
```

```

    </body>
</html>
```

```css
/*
 Dan Armendariz
 Computer Science E-76
 Harvard Extension School
 Spring 2011

 Simple CSS layout.

*/

article,header { display: block; }

html, body {
    margin: 0;
    padding: 0;
    background-color: #c99;
    font: 12px sans-serif;
    height: 100%;
}

header {
    margin: 0;
    padding: 0;
    text-align: center;
    border-bottom: 1px solid black;
}

article {
    border: 1px solid black;
    width: 75%;
    padding: 1em;
    margin: 10px auto;
    background-color: white;
}

article header {
    text-align: left;
    border-bottom: none;
}

article h1 {
    margin: 0;
    padding: 0;
    font-size: 20px;
    font-weight: bold;
```

```css
}
```

```
<!--

 Dan Armendariz
 Computer Science E-76
 Harvard Extension School
 Spring 2011

 A basic implementation of geolocation.

-->
<!DOCTYPE html>
<html>
    <head>
        <title>Geolocation Example</title>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <meta name="viewport" content="initial-scale=1.0" />
        <link rel="stylesheet" href="style.css" />

        <script type="text/javascript">
        // <![CDATA[

        function geo() {
            // perform geo lookup
            navigator.geolocation.getCurrentPosition(print_geo);
        }

        // callback function
        function print_geo(pos) {
            // display data
            alert("Latitude: "  + pos.coords.latitude + "\n" +
                  "Longitude: " + pos.coords.longitude + "\n" +
                  "Accuracy: "  + pos.coords.accuracy + "\n\n" +
                  "Timestamp: " + pos.timestamp);

            /* Other coords properties (availability dependent on browser):
               - altitude (meters)
               - altitudeAccuracy (meters)
               - heading (degrees clockwise from true north)
               - speed (meters/second)
               If unavailable, the data in these properties will be null.
            */
        }

        // ]]>
        </script>
```

```

    </head>

    <body onload="geo()">

    <header>
        <h1>Geolocation Example</h1>
    </header>

    <article>
        <header>
            <h1>Location</h1>
        </header>

        <p>Your location should appear in an alert box, once approved.</p>

    </article>

    </body>
</html>
```

```
<!--

 Dan Armendariz
 Computer Science E-76
 Harvard Extension School
 Spring 2011

 A (better) implementation of geolocation that includes some error handling.

-->
<!DOCTYPE html>
<html>
    <head>
        <title>Geolocation Example</title>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <meta name="viewport" content="initial-scale=1.0" />
        <link rel="stylesheet" href="style.css" />

        <script type="text/javascript">
        // <![CDATA[

        function geo() {
            // perform geo lookup, but only if the geolocation object exists
            if(navigator.geolocation)
                navigator.geolocation.getCurrentPosition(print_geo, handler);
            else
                alert("Geolocation not supported by your browser!");
        }

        // error handler
        function handler(err) {
            alert("Error #" + err.code + ": " + err.message);
        }

        // callback function
        function print_geo(pos) {
            // fetch coordinates

            // display data
            alert("Latitude: "        + pos.coords.latitude + "\n" +
                  "Longitude: "       + pos.coords.longitude + "\n" +
                  "Accuracy (in m): " + pos.coords.accuracy + "\n\n" +
                  "Timestamp: "       + pos.timestamp);

            /* Other coords properties (availability dependent on browser):
```

```
               - altitude (meters)
               - altitudeAccuracy (meters)
               - heading (degrees clockwise from true north)
               - speed (meters/second)
               If unavailable, the data in these properties will be null.
            */
        }

        // ]]>
        </script>

    </head>

    <body onload="geo()">

    <header>
        <h1>Geolocation Example</h1>
    </header>

    <article>
        <header>
            <h1>Location</h1>
        </header>

        <p>Your location should appear in an alert box, once approved.</p>

    </article>

    </body>
</html>
```

```
<!--

 Dan Armendariz
 Computer Science E-76
 Harvard Extension School
 Spring 2011

 An implementation of geolocation that includes (better) error handling and
 more legible time stamp.

-->
<!DOCTYPE html>
<html>
    <head>
        <title>Geolocation Example</title>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <meta name="viewport" content="initial-scale=1.0" />
        <link rel="stylesheet" href="style.css" />

        <script type="text/javascript">
        // <![CDATA[

        function geo() {
            // set some geo options
            var opts = {
                enableHighAccuracy: true,
                timeout: 60000,
                maximumAge: 60000
            };

            // perform geo lookup, but only if the geolocation object exists
            if(navigator.geolocation)
                navigator.geolocation.getCurrentPosition(print_geo, handler, opts);
            else
                alert("Geolocation not supported by your browser!");
        }

        // error handler
        function handler(err) {
            switch(err.code) {
                case err.PERMISSION_DENIED:
                        alert("This page is not allowed to view your position. Message: " + err.message);
                        break;
                case err.POSITION_UNAVAILABLE:
                        alert("Your position is not available. Message: " + err.message);
```

```
                        break;
                case err.TIMEOUT:
                        alert("Timeout when determining your location.");
                        break;
                default:
                    alert("Unknown error occurred! Message: " + err.message);
            }
        }

        // callback function
        function print_geo(pos) {
            // make time stamp a readable format
            var d = new Date(pos.timestamp);

            // display data
            document.getElementById("geo").innerHTML =
                "Latitude: "        + pos.coords.latitude + "<br />" +
                "Longitude: "       + pos.coords.longitude + "<br />" +
                "Accuracy (in m): " + pos.coords.accuracy + "<br />" +
                "Timestamp: "       + d.toLocaleString();

        }

        // ]]>
        </script>

    </head>

    <body onload="geo()">

    <header>
        <h1>Geolocation Example</h1>
    </header>

    <article>
        <header>
            <h1>Location</h1>
        </header>

        <p>Your location should appear below, once approved.</p>

        <div id="geo"></div>
    </article>

    </body>
```

```
</html>
```

```
<!--

 Dan Armendariz
 Computer Science E-76
 Harvard Extension School
 Spring 2011

 An implementation of geolocation that allows the site to continuously track
 a user.

-->
<!DOCTYPE html>
<html>
    <head>
        <title>Geolocation Example</title>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <meta name="viewport" content="initial-scale=1.0" />
        <link rel="stylesheet" href="style.css" />

        <script type="text/javascript">
        // <![CDATA[

        // keep track of the watchPosition ID so we can cancel it later.
        var watchID = null;

        function geo() {
            // set some geo options
            var opts = {
                enableHighAccuracy: true,
                timeout: 60000,
                maximumAge: 60000
            };

            // watch the user’s location
            if(navigator.geolocation)
                watchID = navigator.geolocation.watchPosition(print_geo, handler, opts);
            else
                alert("Geolocation not supported by your browser!");
        }

        // error handler
        function handler(err) {
            switch(err.code) {
                case err.PERMISSION_DENIED:
                        alert("This page is not allowed to view your position. Message: " + err.message);
                        break;
                case err.POSITION_UNAVAILABLE:
                        alert("Your position is not available. Message: " + err.message);
                        break;
                case err.TIMEOUT:
                        alert("Timeout when determining your location.");
                        break;
                default:
                    alert("Unknown error occurred! Message: " + err.message);
            }

            // stop the watch, or iPhones will continuously show an error
            stop_geo();
        }

        // callback function
        function print_geo(pos) {
            // make time stamp a readable format
            var d = new Date(pos.timestamp);

            // display data
            document.getElementById("geo").innerHTML =
                "Latitude: "        + pos.coords.latitude + "<br />" +
                "Longitude: "       + pos.coords.longitude + "<br />" +
                "Accuracy (in m): " + pos.coords.accuracy + "<br />" +
                "Timestamp: "       + d.toLocaleString();

        }

        // stop watching user movement.
        function stop_geo() {
            if(watchID != null) {
                navigator.geolocation.clearWatch(watchID);
                watchID = null;
            } else
                alert("Not currently watching the user.");
        }

        // ]]>
        </script>

    </head>

    <body onload="geo()">

```

```
    <header>
        <h1>Geolocation Example</h1>
    </header>

    <article>
        <header>
            <h1>Location</h1>
        </header>

        <p>Your location should appear below, once approved. <a href="javascript:stop_geo()">Click here to stop.</a
></p>

        <div id="geo"></div>
    </article>

    </body>
</html>
```

```
<!--

 Dan Armendariz
 Computer Science E-76
 Harvard Extension School
 Spring 2011

 A implementation of geolocation and shown on a map.

-->
<!DOCTYPE html>
<html>
    <head>
        <title>Geolocation Example</title>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <link rel="stylesheet" href="style.css" />

        <!-- make this page mobile-friendly by allowing the map to capture zooms -->
        <meta name="viewport" content="initial-scale=1.0, user-scalable=no" />

        <!-- load the Google Maps API v3 -->
        <script type="text/javascript" src="http://maps.google.com/maps/api/js?sensor=false"></script>

        <script type="text/javascript">
        // <![CDATA[

        var map;

        // after page load, initialize the Google Map
        function init() {

            // set some options for our map
            var options = {
                zoom: 13,
                center: new google.maps.LatLng(42.375096,-71.105607),
                mapTypeId: google.maps.MapTypeId.ROADMAP
            };

            // instantiate the map
            map = new google.maps.Map(document.getElementById("map"), options);

            // perform geo lookup
            geo();
        }

```

```
        // perform geo lookup, but only if the geolocation object exists
        function geo() {
            if(navigator.geolocation)
                navigator.geolocation.getCurrentPosition(print_geo, handler);
            else
                alert("Geolocation not supported by your browser!");
        }

        // geolocation error handler
        function handler(err) {
            alert("Error #" + err.code + ": " + err.message);
        }

        // geolocation callback function
        function print_geo(pos) {

            // fetch coordinates and store into a Google Maps LatLng object
            var latlng = new google.maps.LatLng(pos.coords.latitude, pos.coords.longitude);

            // set the center of the map to the location
            map.setCenter(latlng);

            // display a marker at the location
            var marker = new google.maps.Marker({
                map: map,
                draggable: false,
                animation: google.maps.Animation.DROP,
                position: latlng
            });

            // display a circle showing the accuracy radius
            var circle = new google.maps.Circle({
                map: map,
                clickable: false,
                fillColor: "#CC0000",
                fillOpacity: 0.15,
                strokeColor: "#FF0000",
                strokeOpacity: 0.25,
                center: latlng,
                radius: pos.coords.accuracy
            });

            // reset the zoom to show the entirety of the accuracy radius
            map.fitBounds(circle.getBounds());

        }

        // ]]>
        </script>

    </head>

    <body onload="init()">

    <header>
        <h1>Geolocation Example</h1>
    </header>

    <div id="map" style="width:100%; height:100%; background-color: black;"></div>

    </body>
</html>
```

```
<!--

 Dan Armendariz
 Computer Science E-76
 Harvard Extension School
 Spring 2011

 A implementation of geolocation on a map with constant updating.

-->
<!DOCTYPE html>
<html>
    <head>
        <title>Geolocation Example</title>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <link rel="stylesheet" href="style.css" />

        <!-- make this page mobile-friendly by allowing the map to capture zooms -->
        <meta name="viewport" content="initial-scale=1.0, user-scalable=no" />

        <!-- load the Google Maps API v3 -->
        <script type="text/javascript" src="http://maps.google.com/maps/api/js?sensor=false"></script>

        <script type="text/javascript">
        // <![CDATA[

        // vars for Google Maps
        var map, marker, circle;

        // keep track of the watchPosition ID so we can cancel it later.
        var watchID = null;

        // after page load, initialize the Google Map
        function init() {

            // set some options for our map
            var options = {
                zoom: 13,
                center: new google.maps.LatLng(42.375096,-71.105607),
                mapTypeId: google.maps.MapTypeId.ROADMAP
            };

            // instantiate the map
            map = new google.maps.Map(document.getElementById("map"), options);

            // perform geo lookup
            geo();
        }

        // perform geo lookup, but only if the geolocation object exists
        function geo() {

            var opts = {
                enableHighAccuracy: true,
                timeout: 60000,
                maximumAge: 60000
            };

            if(navigator.geolocation)
                watchID = navigator.geolocation.watchPosition(map_geo, handler, opts);
            else
                alert("Geolocation not supported by your browser!");
        }

        // geolocation error handler
        function handler(err) {
            alert("Error #" + err.code + ": " + err.message);

            // stop the watch, or iPhones will continuously show an error
            navigator.geolocation.clearWatch(watchID);
        }

        // geolocation callback function
        function map_geo(pos) {

            // fetch coordinates and store into a Google Maps LatLng object
            var latlng = new google.maps.LatLng(pos.coords.latitude, pos.coords.longitude);
            var accuracy = pos.coords.accuracy;

            // nothing’s been placed on the map yet, so let’s create them
            if(marker == null) {

                // display a marker at the location
                marker = new google.maps.Marker({
                    map: map,
                    draggable: false,
                    position: latlng
                });

                // display a circle showing the accuracy radius
```

```
                circle = new google.maps.Circle({
                    map: map,
                    clickable: false,
                    fillColor: "#CC0000",
                    fillOpacity: 0.15,
                    strokeColor: "#FF0000",
                    strokeOpacity: 0.25,
                    center: latlng,
                    radius: accuracy
                });


            } else {
                // we already have things on the map, let’s update them
                marker.setPosition(latlng);
                circle.setCenter(latlng);
                circle.setRadius(accuracy);
            }

            // set the center of the map to the location
            map.setCenter(latlng);

            // reset the zoom to show the entirety of the accuracy radius
            if (accuracy >= 100)
                map.fitBounds(circle.getBounds());
            else
                // prevent the circle from becoming too small
                map.setZoom(17);

        }

        // ]]>
        </script>

    </head>

    <body onload="init()">

    <header>
        <h1>Geolocation Example</h1>
    </header>

    <div id="map" style="width:100%; height:100%; background-color: black;"></div>

    </body>
```

```
</html>
```

```
/*
 Dan Armendariz
 Computer Science E-76
 Harvard Extension School
 Spring 2011

 Simple CSS layout.

*/

article,header { display: block; }

html, body {
    margin: 0;
    padding: 0;
    background-color: #c99;
    font: 12px sans-serif;
    height: 100%;
}

header {
    margin: 0;
    padding: 0;
    text-align: center;
    border-bottom: 1px solid black;
}

article {
    border: 1px solid black;
    width: 75%;
    padding: 1em;
    margin: 10px auto;
    background-color: white;
}

article header {
    text-align: left;
    border-bottom: none;
}

article h1 {
    margin: 0;
    padding: 0;
    font-size: 20px;
    font-weight: bold;
}
```

```
<!--

 Dan Armendariz
 Computer Science E-76
 Harvard Extension School
 Spring 2011

 An HTML5 webapp that records a GPS track and stores data into localStorage.

-->
<!DOCTYPE html>
<html>
    <head>
        <title>GPS Track Recorder</title>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <meta name="viewport" content="initial-scale=1.0" />
        <link rel="stylesheet" href="style.css" />

        <script type="text/javascript">
        // <![CDATA[

        // keep track of the watchPosition ID so we can cancel it later.
        var watchID = null;

        // variable name in localStorage
        var LOC = "gpsTrack";

        var track = new Array();

        function geo() {
            // reset localStorage
            localStorage.removeItem(LOC);

            // set some geo options
            var opts = {
                enableHighAccuracy: true,
                timeout: 60000,
                maximumAge: 10000
            };

            // watch the user's location
            if(navigator.geolocation)
                watchID = navigator.geolocation.watchPosition(record_geo, handler, opts);
            else
                alert("Geolocation not supported by your browser!");
        }

        // error handler
        function handler(err) {
            switch(err.code) {
                case err.PERMISSION_DENIED:
                        alert("This page is not allowed to view your position. Message: " + err.message);
                        break;
                case err.POSITION_UNAVAILABLE:
                        alert("Your position is not available. Message: " + err.message);
                        break;
                case err.TIMEOUT:
                        alert("Timeout when determining your location.");
                        break;
                default:
                    alert("Unknown error occurred! Message: " + err.message);
            }

            // stop the watch, or iPhones will continuously show an error
            stop_geo();
        }

        // callback function
        function record_geo(pos) {
            // make time stamp a readable format
            var d = new Date(pos.timestamp);

            // display data
            document.getElementById("geo").innerHTML =
                "Latitude: "        + pos.coords.latitude + "<br />" +
                "Longitude: "       + pos.coords.longitude + "<br />" +
                "Accuracy (in m): " + pos.coords.accuracy + "<br />" +
                "Timestamp: "       + d.toLocaleString();

            track.push({
                    lat: pos.coords.latitude,
                    lng: pos.coords.longitude,
                    acc: pos.coords.accuracy
            });

            // save the track into storage
            localStorage[LOC] = JSON.stringify(track);

        }

```

```
        // stop watching user movement.
        function stop_geo() {
            if(watchID != null) {
                navigator.geolocation.clearWatch(watchID);
                watchID = null;
                alert("Stopped recording");
            } else
                alert("Not currently watching the user.");
        }

        // ]]>
        </script>

    </head>

    <body onload="geo()">

    <header>
        <h1>GPS Track Recorder</h1>
    </header>

    <article>
        <header>
            <h1>Location</h1>
        </header>

        <p><a href="javascript:stop_geo()">Stop Recording.</a></p>

        <p>Your location will appear below, once approved.</p>

        <div id="geo"></div>

        <p><a href="show.html">Show your track!</a>
          (<a href="showraw.html">Or view the raw data in storage</a>)</p>

    </article>

    </body>
</html>
```

```
<!--

 Dan Armendariz
 Computer Science E-76
 Harvard Extension School
 Spring 2011

 An HTML5 webapp that records a GPS track and stores data into localStorage.

-->
<!DOCTYPE html>
<html>
    <head>
        <title>Show Recorded GPS Track</title>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <link rel="stylesheet" href="style.css" />
        <meta name="apple-mobile-web-app-capable" content="yes" />
        <meta name="apple-mobile-web-app-status-bar-style" content="black" />

        <!-- make this page mobile-friendly by allowing the map to capture zooms -->
        <meta name="viewport" content="initial-scale=1.0, user-scalable=no" />

        <!-- load the Google Maps API v3 -->
        <script type="text/javascript" src="http://maps.google.com/maps/api/js?sensor=false"></script>


        <script type="text/javascript">
        // <![CDATA[

        // variable name in localStorage
        var LOC = "gpsTrack";

        // JSON gps track data
        var track = null;

        // Google Map
        var map = null;

        // returns true of the web browser supports web storage, false otherwise
        function supports_web_storage() {
            try {
                return 'localStorage' in window && window['localStorage'] !== null;
            } catch (e) {
                return false;
            }
```

```
        }


        function init() {
            if(!supports_web_storage()) {
                alert("Your browser doesn't seem to support web storage!");
                return;
            }



            // set some options for our map
            var options = {
                zoom: 13,
                center: new google.maps.LatLng(42.375096,-71.105607),
                mapTypeId: google.maps.MapTypeId.ROADMAP
            };

            // instantiate the map
            map = new google.maps.Map(document.getElementById("map"), options);

            // make sure we have data
            if(localStorage[LOC] == null) {
                alert("No stored data to show; try recording first.");
                return;
            }

            // retrieve Local Storage data and convert it to a JSON object
            track = JSON.parse(localStorage[LOC]);

            // create a polyline on the map showing the gps track
            var polylineCoordinates = new Array();
            for (var i = 0, j = track.length; i < j; i++) {
                polylineCoordinates.push(
                    new google.maps.LatLng(track[i].lat, track[i].lng)
                );
            }

            var polyline = new google.maps.Polyline({
                path: polylineCoordinates,
                strokeColor: "#FF0000",
                strokeOpacity: 0.5,
                strokeWeight: 4,
                map: map
            });

        }


        // ]]>
        </script>

    </head>

    <body onload="init()">

    <div id="map" style="width:100%; height:100%; background-color: black;"></div>

    </body>
</html>
```

```
<!--

 Dan Armendariz
 Computer Science E-76
 Harvard Extension School
 Spring 2011

 An HTML5 webapp that displays the raw recorded GPS track.

-->
<!DOCTYPE html>
<html>
    <head>
        <title>Show Raw Recorded GPS Track</title>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <meta name="viewport" content="initial-scale=1.0" />
        <link rel="stylesheet" href="style.css" />


        <script type="text/javascript">
        // <![CDATA[

        // variable name in localStorage
        var LOC = "gpsTrack";

        // JSON gps track data
        var track = null;

        // returns true of the web browser supports web storage, false otherwise
        function supports_web_storage() {
            try {
                return 'localStorage' in window && window['localStorage'] !== null;
            } catch (e) {
                return false;
            }
        }


        function init() {
            if(!supports_web_storage()) {
                alert("Your browser doesn't seem to support web storage!");
                return;
            }

            // make sure we have data
            if(localStorage[LOC] == null) {
                document.getElementById("geo").innerHTML = "No GPS track in storage! Try recording first.";
                return;
            }

            // retrieve Local Storage data and convert it to a JSON object
            document.getElementById("geo").innerHTML = localStorage[LOC];

        }


        // ]]>
        </script>

    </head>

    <body onload="init()">

    <header>
        <h1>GPS Track Recorder</h1>
    </header>

    <article>
        <header>
            <h1>Raw Track in Storage</h1>
        </header>

        <div id="geo"></div>

    </article>

    </body>
</html>
```

```
/*
 Dan Armendariz
 Computer Science E-76
 Harvard Extension School
 Spring 2011

 Simple CSS layout.

*/

article,header { display: block; }

html, body {
    margin: 0;
    padding: 0;
    background-color: #c99;
    font: 12px sans-serif;
    height: 100%;
}

header {
    margin: 0;
    padding: 0;
    text-align: center;
    border-bottom: 1px solid black;
}

article {
    border: 1px solid black;
    width: 75%;
    padding: 1em;
    margin: 10px auto;
    background-color: white;
}

article header {
    text-align: left;
    border-bottom: none;
}

article h1 {
    margin: 0;
    padding: 0;
    font-size: 20px;
    font-weight: bold;
}
```