```
<!--

Dan Armendariz
Computer Science 76
Building Mobile Applications
Harvard Extension School

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>

```

```
49.      </article>
50.
51.      </body>
52.  </html>
```

```
<!--

Dan Armendariz
Computer Science 76
Building Mobile Applications
Harvard Extension School

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!");
       }

```

```
49.         // ]]>
50.          </script>
51.
52.     </head>
53.
54.     <body onload="store_test()">
55.
56.     <header>
57.         <h1>Web Storage Example</h1>
58.     </header>
59.
60.     <article>
61.         <header>
62.             <h1>Web storage</h1>
63.         </header>
64.
65.         <p>The contents of the local storage should appear in an alert box.</p>
66.
67.     </article>
68.
69.     </body>
70. </html>
```

```
<!--

Dan Armendariz
Computer Science 76
Building Mobile Applications
Harvard Extension School

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 76
Building Mobile Applications
Harvard Extension School

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 76
Building Mobile Applications
Harvard Extension School

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>
```

```
/*
 Dan Armendariz
 Computer Science 76
 Building Mobile Applications
 Harvard Extension School

 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 76
Building Mobile Applications
Harvard Extension School

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>

```

```
49.     <body onload="geo()">
50.
51.     <header>
52.         <h1>Geolocation Example</h1>
53.     </header>
54.
55.     <article>
56.         <header>
57.             <h1>Location</h1>
58.         </header>
59.
60.         <p>Your location should appear in an alert box, once approved.</p>
61.
62.     </article>
63.
64.     </body>
65. </html>
```

```
<!--

Dan Armendariz
Computer Science 76
Building Mobile Applications
Harvard Extension School

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 76
Building Mobile Applications
Harvard Extension School

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 76
Building Mobile Applications
Harvard Extension School

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>
```

```
 97.             <h1>Location</h1>
 98.         </header>
 99.
100.         <p>Your location should appear below, once approved. <a href="javascript:stop_geo()">Click here to stop.</a></p>
101.
102.         <div id="geo"></div>
103.     </article>
104.
105.     </body>
106. </html>
```

```
<!--

Dan Armendariz
Computer Science 76
Building Mobile Applications
Harvard Extension School

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)
```

```
49.                 navigator.geolocation.getCurrentPosition(print_geo, handler);
50.             else
51.                 alert("Geolocation not supported by your browser!");
52.         }
53.
54.         // geolocation error handler
55.         function handler(err) {
56.             alert("Error #" + err.code + ": " + err.message);
57.         }
58.
59.         // geolocation callback function
60.         function print_geo(pos) {
61.
62.             // fetch coordinates and store into a Google Maps LatLng object
63.             var latlng = new google.maps.LatLng(pos.coords.latitude, pos.coords.longitude);
64.
65.             // set the center of the map to the location
66.             map.setCenter(latlng);
67.
68.             // display a marker at the location
69.             var marker = new google.maps.Marker({
70.                 map: map,
71.                 draggable: false,
72.                 animation: google.maps.Animation.DROP,
73.                 position: latlng
74.             });
75.
76.             // display a circle showing the accuracy radius
77.             var circle = new google.maps.Circle({
78.                 map: map,
79.                 clickable: false,
80.                 fillColor: "#CC0000",
81.                 fillOpacity: 0.15,
82.                 strokeColor: "#FF0000",
83.                 strokeOpacity: 0.25,
84.                 center: latlng,
85.                 radius: pos.coords.accuracy
86.             });
87.
88.             // reset the zoom to show the entirety of the accuracy radius
89.             map.fitBounds(circle.getBounds());
90.
91.         }
92.
93.         // ]]>
94.          </script>
95.
96.     </head>
```

```
 97.
 98.     <body onload="init()">
 99.
100.     <header>
101.         <h1>Geolocation Example</h1>
102.     </header>
103.
104.     <div id="map" style="width:100%; height:100%; background-color: black;"></div>
105.
106.     </body>
107. </html>
```

```
<!--

Dan Armendariz
Computer Science 76
Building Mobile Applications
Harvard Extension School

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();
       }
```

```
49.
50.         // perform geo lookup, but only if the geolocation object exists
51.         function geo() {
52.
53.             var opts = {
54.                 enableHighAccuracy: true,
55.                 timeout: 60000,
56.                 maximumAge: 60000
57.             };
58.
59.             if(navigator.geolocation)
60.                 watchID = navigator.geolocation.watchPosition(map_geo, handler, opts);
61.             else
62.                 alert("Geolocation not supported by your browser!");
63.         }
64.
65.         // geolocation error handler
66.         function handler(err) {
67.             alert("Error #" + err.code + ": " + err.message);
68.
69.             // stop the watch, or iPhones will continuously show an error
70.             navigator.geolocation.clearWatch(watchID);
71.         }
72.
73.         // geolocation callback function
74.         function map_geo(pos) {
75.
76.             // fetch coordinates and store into a Google Maps LatLng object
77.             var latlng = new google.maps.LatLng(pos.coords.latitude, pos.coords.longitude);
78.             var accuracy = pos.coords.accuracy;
79.
80.             // nothing's been placed on the map yet, so let's create them
81.             if(marker == null) {
82.
83.                 // display a marker at the location
84.                 marker = new google.maps.Marker({
85.                     map: map,
86.                     draggable: false,
87.                     position: latlng
88.                 });
89.
90.                 // display a circle showing the accuracy radius
91.                 circle = new google.maps.Circle({
92.                     map: map,
93.                     clickable: false,
94.                     fillColor: "#CC0000",
95.                     fillOpacity: 0.15,
96.                     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>
```

```css
/*
 Dan Armendariz
 Computer Science 76
 Building Mobile Applications
 Harvard Extension School

 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 76
Building Mobile Applications
Harvard Extension School

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 76
Building Mobile Applications
Harvard Extension School

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;
           }
       }


```

```
49.         function init() {
50.             if(!supports_web_storage()) {
51.                 alert("Your browser doesn't seem to support web storage!");
52.                 return;
53.             }
54.
55.
56.
57.             // set some options for our map
58.             var options = {
59.                 zoom: 13,
60.                 center: new google.maps.LatLng(42.375096,-71.105607),
61.                 mapTypeId: google.maps.MapTypeId.ROADMAP
62.             };
63.
64.             // instantiate the map
65.             map = new google.maps.Map(document.getElementById("map"), options);
66.
67.             // make sure we have data
68.             if(localStorage[LOC] == null) {
69.                 alert("No stored data to show; try recording first.");
70.                 return;
71.             }
72.
73.             // retrieve Local Storage data and convert it to a JSON object
74.             track = JSON.parse(localStorage[LOC]);
75.
76.             // create a polyline on the map showing the gps track
77.             var polylineCoordinates = new Array();
78.             for (var i = 0, j = track.length; i < j; i++) {
79.                 polylineCoordinates.push(
80.                     new google.maps.LatLng(track[i].lat, track[i].lng)
81.                 );
82.             }
83.
84.             var polyline = new google.maps.Polyline({
85.                 path: polylineCoordinates,
86.                 strokeColor: "#FF0000",
87.                 strokeOpacity: 0.5,
88.                 strokeWeight: 4,
89.                 map: map
90.             });
91.
92.         }
93.
94.
95.         // ]]>
96.          </script>
```

```
 97.
 98.     </head>
 99.
100.     <body onload="init()">
101.
102.     <div id="map" style="width:100%; height:100%; background-color: black;"></div>
103.
104.     </body>
105. </html>
```

```
<!--

Dan Armendariz
Computer Science 76
Building Mobile Applications
Harvard Extension School

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>
```

```css
/*
 Dan Armendariz
 Computer Science 76
 Building Mobile Applications
 Harvard Extension School

 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;
}
```