

```
1. <!--
2.
3. Dan Armendariz
4. Computer Science 76
5. Building Mobile Applications
6. Harvard Extension School
7.
8. A basic implementation of web storage.
9.
10. -->
11. <!DOCTYPE html>
12. <html>
13.   <head>
14.     <title>Web Storage Example</title>
15.     <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
16.     <meta name="viewport" content="initial-scale=1.0" />
17.     <link rel="stylesheet" href="style.css" />
18.
19.     <script type="text/javascript">
20.       // 
21.
22.       // demonstrate some local storage capabilities
23.       function store_test() {
24.         // retrieve the value in local storage
25.         alert(localStorage.getItem("foo"));
26.
27.         // set some local storage
28.         localStorage.setItem("foo", "test!");
29.       }
30.
31.       // ]]&gt;
32.     &lt;/script&gt;
33.
34.   &lt;/head&gt;
35.
36.   &lt;body onload="store_test()"&gt;
37.
38.     &lt;header&gt;
39.       &lt;h1&gt;Web Storage Example&lt;/h1&gt;
40.     &lt;/header&gt;
41.
42.     &lt;article&gt;
43.       &lt;header&gt;
44.         &lt;h1&gt;Web storage&lt;/h1&gt;
45.       &lt;/header&gt;
46.
47.       &lt;p&gt;The contents of the local storage should appear in an alert box.&lt;/p&gt;
48.</pre></div>
```

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

```
1. <!--
2.
3. Dan Armendariz
4. Computer Science 76
5. Building Mobile Applications
6. Harvard Extension School
7.
8. A basic implementation of web storage, with some error handling.
9.
10. -->
11. <!DOCTYPE html>
12. <html>
13.   <head>
14.     <title>Web Storage Example</title>
15.     <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
16.     <meta name="viewport" content="initial-scale=1.0" />
17.     <link rel="stylesheet" href="style.css" />
18.
19.     <script type="text/javascript">
20.       // <![CDATA[
21.
22.       // returns true if the web browser supports web storage, false otherwise
23.       function supports_web_storage() {
24.         try {
25.           return 'localStorage' in window && window['localStorage'] !== null;
26.         } catch (e) {
27.           return false;
28.         }
29.       }
30.
31.       // demonstrate some local storage capabilities
32.       function store_test() {
33.         if(!supports_web_storage()) {
34.           alert("Your browser doesn't seem to support web storage!");
35.           return;
36.         }
37.
38.         // retrieve the value in local storage
39.         var data = localStorage.getItem("bar");
40.         if(data == null)
41.           alert("Data is null (nothing in storage for the key 'bar').");
42.         else
43.           alert(data);
44.
45.         // set some local storage
46.         localStorage.setItem("bar", "test!");
47.       }
48.
```

```
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>
```

```
1. <!--
2.
3. Dan Armendariz
4. Computer Science 76
5. Building Mobile Applications
6. Harvard Extension School
7.
8. A basic implementation of web storage, now with more integers and different
9. ways of accessing data.
10.
11. -->
12. <!DOCTYPE html>
13. <html>
14.   <head>
15.     <title>Web Storage Example</title>
16.     <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
17.     <meta name="viewport" content="initial-scale=1.0" />
18.     <link rel="stylesheet" href="style.css" />
19.
20.     <script type="text/javascript">
21.       // 
22.
23.       // returns true if the web browser supports web storage, false otherwise
24.       function supports_web_storage() {
25.         try {
26.           return 'localStorage' in window &amp;&amp; window['localStorage'] !== null;
27.         } catch (e) {
28.           return false;
29.         }
30.       }
31.
32.       // demonstrate some local storage capabilities
33.       function store_test() {
34.         if(!supports_web_storage()) {
35.           alert("Your browser doesn't seem to support web storage!");
36.           return;
37.         }
38.
39.         // retrieve the counter from local storage
40.         var data = localStorage["count"];
41.
42.         // reset the counter if they've not visited before
43.         if(data == null)
44.           data = 0;
45.
46.         // tell the user how many times they've visited
47.         alert("You've visited this page " + data + " times in the past!");
48.</pre></div>
```

```
49.         // set the counter in local storage
50.         localStorage["count"] = ++data;
51.     }
52.
53.     // mess up the counter by replacing it with an actual string
54.     function messup() {
55.         localStorage["count"] = "test";
56.
57.         store_test();
58.     }
59.
60.     // ]]>
61.     </script>
62.
63. </head>
64.
65. <body onload="store_test()">
66.
67. <header>
68.     <h1>Web Storage Example</h1>
69. </header>
70.
71. <article>
72.     <header>
73.         <h1>Web storage</h1>
74.     </header>
75.
76.     <p>The contents of the local storage should appear in an alert box.</p>
77.
78.     <p><a href="javascript:messup()">Mess up counter</a></p>
79.
80. </article>
81.
82. </body>
83. </html>
```

```
1. <!--
2.
3. Dan Armendariz
4. Computer Science 76
5. Building Mobile Applications
6. Harvard Extension School
7.
8. A basic implementation of web storage while protecting the type of data.
9.
10. -->
11. <!DOCTYPE html>
12. <html>
13.   <head>
14.     <title>Web Storage Example</title>
15.     <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
16.     <meta name="viewport" content="initial-scale=1.0" />
17.     <link rel="stylesheet" href="style.css" />
18.
19.     <script type="text/javascript">
20.       // <![CDATA[
21.
22.       // returns true if the web browser supports web storage, false otherwise
23.       function supports_web_storage() {
24.         try {
25.           return 'localStorage' in window && window['localStorage'] !== null;
26.         } catch (e) {
27.           return false;
28.         }
29.       }
30.
31.       // demonstrate some local storage capabilities
32.       function store_test() {
33.         if(!supports_web_storage()) {
34.           alert("Your browser doesn't seem to support web storage!");
35.           return;
36.         }
37.
38.         // Notice the type that this (and all other) data is stored:
39.         alert(typeof localStorage["count"]);
40.
41.         // retrieve the value from local storage (safer)
42.         var data = parseInt(localStorage["count"]);
43.
44.         // reset the counter if they've not visited before
45.         if(data == null || isNaN(data))
46.           data = 0;
47.
48.         // tell the user how many times they've visited
```

```
49.         alert("You've visited this page " + data + " times in the past!");
50.
51.         // set the counter in local storage
52.         localStorage["count"] = ++data;
53.
54.     }
55.
56.     // mess up the counter by replacing it with an actual string
57.     function messup() {
58.         localStorage["count"] = "test";
59.
60.         store_test();
61.     }
62.
63.     // reset the counter at the user's request.
64.     function reset() {
65.         localStorage.removeItem("count");
66.     }
67.
68.     // ]]>
69.     </script>
70.
71. </head>
72.
73. <body onload="store_test()">
74.
75. <header>
76.     <h1>Web Storage Example</h1>
77. </header>
78.
79. <article>
80.     <header>
81.         <h1>Web storage</h1>
82.     </header>
83.
84.     <p>The contents of the local storage should appear in an alert box.</p>
85.
86.     <p><a href="javascript:messup()">Messup counter</a> | <a href="javascript:reset()">Reset Counter</a></p>
87.
88. </article>
89.
90. </body>
91. </html>
```



```
1. <!--
2.
3. Dan Armendariz
4. Computer Science 76
5. Building Mobile Applications
6. Harvard Extension School
7.
8. A demonstration that local storage is not like cookies: by storing
9. lots of data and retrieving without sending to the server.
10.
11. -->
12. <!DOCTYPE html>
13. <html>
14.   <head>
15.     <title>Web Storage Example</title>
16.     <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
17.     <meta name="viewport" content="initial-scale=1.0" />
18.     <link rel="stylesheet" href="style.css" />
19.
20.     <script type="text/javascript">
21.       // 
22.
23.       // returns true if the web browser supports web storage, false otherwise
24.       function supports_web_storage() {
25.         try {
26.           return 'localStorage' in window &amp;&amp; window['localStorage'] !== null;
27.         } catch (e) {
28.           return false;
29.         }
30.       }
31.
32.       // make sure the browser supports web storage
33.       function init() {
34.         if(!supports_web_storage()) {
35.           alert("Your browser doesn't seem to support web storage!");
36.           return;
37.         }
38.
39.         // obtain data in storage
40.         fetch();
41.       }
42.
43.       // fetch the data from local storage and place into text area
44.       function fetch() {
45.         var data = localStorage["MyText"];
46.
47.         if(data == null) {
48.           alert("No text in storage!");</pre></div>
```

```
49.         return;
50.     }
51.
52.     document.getElementById("txt").value = data;
53. }
54.
55. // save the text area into storage
56. function save() {
57.     localStorage["MyText"] = document.getElementById("txt").value;
58. }
59.
60. // remove item from storage
61. function reset() {
62.     localStorage.removeItem("MyText");
63. }
64.
65. // ]]>
66. </script>
67.
68. </head>
69.
70. <body onload="init()">
71.
72. <header>
73.     <h1>Web Storage Example</h1>
74. </header>
75.
76. <article>
77.     <header>
78.         <h1>Web storage</h1>
79.     </header>
80.
81.     <!-- Safari bug: adding the "autofocus" attribute prevents us from writing the value to the text area with JS -->
82.     <textarea rows="50" cols="80" id="txt"></textarea><br />
83.
84.     <input type="button" onclick="save(); return false;" value="Save Data" />
85.
86.     <input type="button" onclick="fetch(); return false;" value="Fetch Data" />
87.
88.     <input type="button" onclick="reset(); return false;" value="Reset" />
89. </article>
90.
91. </body>
92. </html>
```

```
1. /*
2.  Dan Armendariz
3.  Computer Science 76
4.  Building Mobile Applications
5.  Harvard Extension School
6.
7.  Simple CSS layout.
8.
9.  */
10.
11. article,header { display: block; }
12.
13. html, body {
14.     margin: 0;
15.     padding: 0;
16.     background-color: #c99;
17.     font: 12px sans-serif;
18.     height: 100%;
19. }
20.
21. header {
22.     margin: 0;
23.     padding: 0;
24.     text-align: center;
25.     border-bottom: 1px solid black;
26. }
27.
28. article {
29.     border: 1px solid black;
30.     width: 75%;
31.     padding: 1em;
32.     margin: 10px auto;
33.     background-color: white;
34. }
35.
36. article header {
37.     text-align: left;
38.     border-bottom: none;
39. }
40.
41. article h1 {
42.     margin: 0;
43.     padding: 0;
44.     font-size: 20px;
45.     font-weight: bold;
46. }
```

```
1. <!--
2.
3. Dan Armendariz
4. Computer Science 76
5. Building Mobile Applications
6. Harvard Extension School
7.
8. A basic implementation of geolocation.
9.
10. -->
11. <!DOCTYPE html>
12. <html>
13.   <head>
14.     <title>Geolocation Example</title>
15.     <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
16.     <meta name="viewport" content="initial-scale=1.0" />
17.     <link rel="stylesheet" href="style.css" />
18.
19.     <script type="text/javascript">
20.       // <![CDATA[
21.
22.       function geo() {
23.         // perform geo lookup
24.         navigator.geolocation.getCurrentPosition(print_geo);
25.       }
26.
27.       // callback function
28.       function print_geo(pos) {
29.         // display data
30.         alert("Latitude: " + pos.coords.latitude + "\n" +
31.             "Longitude: " + pos.coords.longitude + "\n" +
32.             "Accuracy: " + pos.coords.accuracy + "\n\n" +
33.             "Timestamp: " + pos.timestamp);
34.
35.         /* Other coords properties (availability dependent on browser):
36.          - altitude (meters)
37.          - altitudeAccuracy (meters)
38.          - heading (degrees clockwise from true north)
39.          - speed (meters/second)
40.          If unavailable, the data in these properties will be null.
41.         */
42.       }
43.
44.     // ]]>
45.   </script>
46. </head>
47.
48.
```

```
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>
```

```
1. <!--
2.
3. Dan Armendariz
4. Computer Science 76
5. Building Mobile Applications
6. Harvard Extension School
7.
8. A (better) implementation of geolocation that includes some error handling.
9.
10. -->
11. <!DOCTYPE html>
12. <html>
13.   <head>
14.     <title>Geolocation Example</title>
15.     <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
16.     <meta name="viewport" content="initial-scale=1.0" />
17.     <link rel="stylesheet" href="style.css" />
18.
19.     <script type="text/javascript">
20.       // 
21.
22.       function geo() {
23.         // perform geo lookup, but only if the geolocation object exists
24.         if(navigator.geolocation)
25.           navigator.geolocation.getCurrentPosition(print_geo, handler);
26.         else
27.           alert("Geolocation not supported by your browser!");
28.       }
29.
30.       // error handler
31.       function handler(err) {
32.         alert("Error #" + err.code + ": " + err.message);
33.       }
34.
35.       // callback function
36.       function print_geo(pos) {
37.         // fetch coordinates
38.
39.         // display data
40.         alert("Latitude: "      + pos.coords.latitude + "\n" +
41.             "Longitude: "      + pos.coords.longitude + "\n" +
42.             "Accuracy (in m): " + pos.coords.accuracy + "\n\n" +
43.             "Timestamp: "      + pos.timestamp);
44.
45.         /* Other coords properties (availability dependent on browser):
46.          - altitude (meters)
47.          - altitudeAccuracy (meters)
48.          - heading (degrees clockwise from true north)</pre></div>
```

```
49.         - speed (meters/second)
50.         If unavailable, the data in these properties will be null.
51.     */
52.     }
53.
54. // ]]>
55. </script>
56.
57. </head>
58.
59. <body onload="geo()">
60.
61. <header>
62.     <h1>Geolocation Example</h1>
63. </header>
64.
65. <article>
66.     <header>
67.         <h1>Location</h1>
68.     </header>
69.
70.     <p>Your location should appear in an alert box, once approved.</p>
71.
72. </article>
73.
74. </body>
75. </html>
```

```
1. <!--
2.
3. Dan Armendariz
4. Computer Science 76
5. Building Mobile Applications
6. Harvard Extension School
7.
8. An implementation of geolocation that includes (better) error handling and
9. more legible time stamp.
10.
11. -->
12. <!DOCTYPE html>
13. <html>
14.   <head>
15.     <title>Geolocation Example</title>
16.     <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
17.     <meta name="viewport" content="initial-scale=1.0" />
18.     <link rel="stylesheet" href="style.css" />
19.
20.     <script type="text/javascript">
21.       // <![CDATA[
22.
23.       function geo() {
24.         // set some geo options
25.         var opts = {
26.           enableHighAccuracy: true,
27.           timeout: 60000,
28.           maximumAge: 60000
29.         };
30.
31.         // perform geo lookup, but only if the geolocation object exists
32.         if(navigator.geolocation)
33.           navigator.geolocation.getCurrentPosition(print_geo, handler, opts);
34.         else
35.           alert("Geolocation not supported by your browser!");
36.       }
37.
38.       // error handler
39.       function handler(err) {
40.         switch(err.code) {
41.           case err.PERMISSION_DENIED:
42.             alert("This page is not allowed to view your position. Message: " + err.message);
43.             break;
44.           case err.POSITION_UNAVAILABLE:
45.             alert("Your position is not available. Message: " + err.message);
46.             break;
47.           case err.TIMEOUT:
48.             alert("Timeout when determining your location.");
```



```
49.             break;
50.         default:
51.             alert("Unknown error occurred! Message: " + err.message);
52.     }
53. }
54.
55. // callback function
56. function print_geo(pos) {
57.     // make time stamp a readable format
58.     var d = new Date(pos.timestamp);
59.
60.     // display data
61.     document.getElementById("geo").innerHTML =
62.         "Latitude: "      + pos.coords.latitude + "<br />" +
63.         "Longitude: "    + pos.coords.longitude + "<br />" +
64.         "Accuracy (in m): " + pos.coords.accuracy + "<br />" +
65.         "Timestamp: "    + d.toLocaleString();
66.
67. }
68.
69. // ]]>
70. </script>
71.
72. </head>
73.
74. <body onload="geo()">
75.
76. <header>
77.     <h1>Geolocation Example</h1>
78. </header>
79.
80. <article>
81.     <header>
82.         <h1>Location</h1>
83.     </header>
84.
85.     <p>Your location should appear below, once approved.</p>
86.
87.     <div id="geo"></div>
88. </article>
89.
90. </body>
91. </html>
```

```
1. <!--
2.
3. Dan Armendariz
4. Computer Science 76
5. Building Mobile Applications
6. Harvard Extension School
7.
8. An implementation of geolocation that allows the site to continuously track
9. a user.
10.
11. -->
12. <!DOCTYPE html>
13. <html>
14.   <head>
15.     <title>Geolocation Example</title>
16.     <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
17.     <meta name="viewport" content="initial-scale=1.0" />
18.     <link rel="stylesheet" href="style.css" />
19.
20.     <script type="text/javascript">
21.       // 
22.
23.       // keep track of the watchPosition ID so we can cancel it later.
24.       var watchID = null;
25.
26.       function geo() {
27.         // set some geo options
28.         var opts = {
29.           enableHighAccuracy: true,
30.           timeout: 60000,
31.           maximumAge: 60000
32.         };
33.
34.         // watch the user's location
35.         if(navigator.geolocation)
36.           watchID = navigator.geolocation.watchPosition(print_geo, handler, opts);
37.         else
38.           alert("Geolocation not supported by your browser!");
39.       }
40.
41.       // error handler
42.       function handler(err) {
43.         switch(err.code) {
44.           case err.PERMISSION_DENIED:
45.             alert("This page is not allowed to view your position. Message: " + err.message);
46.             break;
47.           case err.POSITION_UNAVAILABLE:
48.             alert("Your position is not available. Message: " + err.message);</pre></div>
```

```
49.             break;
50.         case err.TIMEOUT:
51.             alert("Timeout when determining your location.");
52.             break;
53.         default:
54.             alert("Unknown error occurred! Message: " + err.message);
55.     }
56.
57.     // stop the watch, or iPhones will continuously show an error
58.     stop_geo();
59. }
60.
61. // callback function
62. function print_geo(pos) {
63.     // make time stamp a readable format
64.     var d = new Date(pos.timestamp);
65.
66.     // display data
67.     document.getElementById("geo").innerHTML =
68.         "Latitude: "      + pos.coords.latitude + "<br />" +
69.         "Longitude: "     + pos.coords.longitude + "<br />" +
70.         "Accuracy (in m): " + pos.coords.accuracy + "<br />" +
71.         "Timestamp: "     + d.toLocaleString();
72. }
73.
74.
75. // stop watching user movement.
76. function stop_geo() {
77.     if(watchID != null) {
78.         navigator.geolocation.clearWatch(watchID);
79.         watchID = null;
80.     } else
81.         alert("Not currently watching the user.");
82. }
83.
84. // ]]>
85. </script>
86.
87. </head>
88.
89. <body onload="geo()">
90.
91. <header>
92.     <h1>Geolocation Example</h1>
93. </header>
94.
95. <article>
96.     <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>
```

```
1. <!--
2.
3. Dan Armendariz
4. Computer Science 76
5. Building Mobile Applications
6. Harvard Extension School
7.
8. A implementation of geolocation and shown on a map.
9.
10. -->
11. <!DOCTYPE html>
12. <html>
13.     <head>
14.         <title>Geolocation Example</title>
15.         <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
16.         <link rel="stylesheet" href="style.css" />
17.
18.         <!-- make this page mobile-friendly by allowing the map to capture zooms -->
19.         <meta name="viewport" content="initial-scale=1.0, user-scalable=no" />
20.
21.         <!-- load the Google Maps API v3 -->
22.         <script type="text/javascript" src="http://maps.google.com/maps/api/js?sensor=false"></script>
23.
24.         <script type="text/javascript">
25.             // 
26.
27.             var map;
28.
29.             // after page load, initialize the Google Map
30.             function init() {
31.
32.                 // set some options for our map
33.                 var options = {
34.                     zoom: 13,
35.                     center: new google.maps.LatLng(42.375096,-71.105607),
36.                     mapTypeId: google.maps.MapTypeId.ROADMAP
37.                 };
38.
39.                 // instantiate the map
40.                 map = new google.maps.Map(document.getElementById("map"), options);
41.
42.                 // perform geo lookup
43.                 geo();
44.             }
45.
46.             // perform geo lookup, but only if the geolocation object exists
47.             function geo() {
48.                 if(navigator.geolocation)</pre></div>
```

```
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>
```

```
1. <!--
2.
3. Dan Armendariz
4. Computer Science 76
5. Building Mobile Applications
6. Harvard Extension School
7.
8. A implementation of geolocation on a map with constant updating.
9.
10. -->
11. <!DOCTYPE html>
12. <html>
13.     <head>
14.         <title>Geolocation Example</title>
15.         <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
16.         <link rel="stylesheet" href="style.css" />
17.
18.         <!-- make this page mobile-friendly by allowing the map to capture zooms -->
19.         <meta name="viewport" content="initial-scale=1.0, user-scalable=no" />
20.
21.         <!-- load the Google Maps API v3 -->
22.         <script type="text/javascript" src="http://maps.google.com/maps/api/js?sensor=false"></script>
23.
24.         <script type="text/javascript">
25.             // 
26.
27.             // vars for Google Maps
28.             var map, marker, circle;
29.
30.             // keep track of the watchPosition ID so we can cancel it later.
31.             var watchID = null;
32.
33.             // after page load, initialize the Google Map
34.             function init() {
35.
36.                 // set some options for our map
37.                 var options = {
38.                     zoom: 13,
39.                     center: new google.maps.LatLng(42.375096,-71.105607),
40.                     mapTypeId: google.maps.MapTypeId.ROADMAP
41.                 };
42.
43.                 // instantiate the map
44.                 map = new google.maps.Map(document.getElementById("map"), options);
45.
46.                 // perform geo lookup
47.                 geo();
48.             }</pre></div>
```



```
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",
```

```
97.         strokeOpacity: 0.25,
98.         center: latlng,
99.         radius: accuracy
100.     });
101.
102.
103.     } else {
104.         // we already have things on the map, let's update them
105.         marker.setPosition(latlng);
106.         circle.setCenter(latlng);
107.         circle.setRadius(accuracy);
108.     }
109.
110.     // set the center of the map to the location
111.     map.setCenter(latlng);
112.
113.     // reset the zoom to show the entirety of the accuracy radius
114.     if (accuracy >= 100)
115.         map.fitBounds(circle.getBounds());
116.     else
117.         // prevent the circle from becoming too small
118.         map.setZoom(17);
119.
120. }
121.
122. // ]]>
123. </script>
124.
125. </head>
126.
127. <body onload="init()">
128.
129. <header>
130.     <h1>Geolocation Example</h1>
131. </header>
132.
133. <div id="map" style="width:100%; height:100%; background-color: black;"></div>
134.
135. </body>
136. </html>
```

```
1. /*
2.  Dan Armendariz
3.  Computer Science 76
4.  Building Mobile Applications
5.  Harvard Extension School
6.
7.  Simple CSS layout.
8.
9. */
10.
11. article,header { display: block; }
12.
13. html, body {
14.     margin: 0;
15.     padding: 0;
16.     background-color: #c99;
17.     font: 12px sans-serif;
18.     height: 100%;
19. }
20.
21. header {
22.     margin: 0;
23.     padding: 0;
24.     text-align: center;
25.     border-bottom: 1px solid black;
26. }
27.
28. article {
29.     border: 1px solid black;
30.     width: 75%;
31.     padding: 1em;
32.     margin: 10px auto;
33.     background-color: white;
34. }
35.
36. article header {
37.     text-align: left;
38.     border-bottom: none;
39. }
40.
41. article h1 {
42.     margin: 0;
43.     padding: 0;
44.     font-size: 20px;
45.     font-weight: bold;
46. }
```

```
1. <!--
2.
3. Dan Armendariz
4. Computer Science 76
5. Building Mobile Applications
6. Harvard Extension School
7.
8. An HTML5 webapp that records a GPS track and stores data into localStorage.
9.
10. -->
11. <!DOCTYPE html>
12. <html>
13.   <head>
14.     <title>GPS Track Recorder</title>
15.     <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
16.     <meta name="viewport" content="initial-scale=1.0" />
17.     <link rel="stylesheet" href="style.css" />
18.
19.     <script type="text/javascript">
20.       // 
21.
22.       // keep track of the watchPosition ID so we can cancel it later.
23.       var watchID = null;
24.
25.       // variable name in localStorage
26.       var LOC = "gpsTrack";
27.
28.       var track = new Array();
29.
30.       function geo() {
31.         // reset localStorage
32.         localStorage.removeItem(LOC);
33.
34.         // set some geo options
35.         var opts = {
36.           enableHighAccuracy: true,
37.           timeout: 60000,
38.           maximumAge: 10000
39.         };
40.
41.         // watch the user's location
42.         if(navigator.geolocation)
43.           watchID = navigator.geolocation.watchPosition(record_geo, handler, opts);
44.         else
45.           alert("Geolocation not supported by your browser!");
46.       }
47.
48.       // error handler</pre></div>
```

```
49.     function handler(err) {
50.         switch(err.code) {
51.             case err.PERMISSION_DENIED:
52.                 alert("This page is not allowed to view your position. Message: " + err.message);
53.                 break;
54.             case err.POSITION_UNAVAILABLE:
55.                 alert("Your position is not available. Message: " + err.message);
56.                 break;
57.             case err.TIMEOUT:
58.                 alert("Timeout when determining your location.");
59.                 break;
60.             default:
61.                 alert("Unknown error occurred! Message: " + err.message);
62.         }
63.
64.         // stop the watch, or iPhones will continuously show an error
65.         stop_geo();
66.     }
67.
68.     // callback function
69.     function record_geo(pos) {
70.         // make time stamp a readable format
71.         var d = new Date(pos.timestamp);
72.
73.         // display data
74.         document.getElementById("geo").innerHTML =
75.             "Latitude: "      + pos.coords.latitude + "<br />" +
76.             "Longitude: "    + pos.coords.longitude + "<br />" +
77.             "Accuracy (in m): " + pos.coords.accuracy + "<br />" +
78.             "Timestamp: "    + d.toLocaleString();
79.
80.         track.push({
81.             lat: pos.coords.latitude,
82.             lng: pos.coords.longitude,
83.             acc: pos.coords.accuracy
84.         });
85.
86.         // save the track into storage
87.         localStorage[LOC] = JSON.stringify(track);
88.
89.     }
90.
91.     // stop watching user movement.
92.     function stop_geo() {
93.         if(watchID != null) {
94.             navigator.geolocation.clearWatch(watchID);
95.             watchID = null;
96.             alert("Stopped recording");
```

```
97.         } else
98.             alert("Not currently watching the user.");
99.     }
100.
101.     // ]]>
102.     </script>
103.
104. </head>
105.
106. <body onload="geo()">
107.
108. <header>
109.     <h1>GPS Track Recorder</h1>
110. </header>
111.
112. <article>
113.     <header>
114.         <h1>Location</h1>
115.     </header>
116.
117.     <p><a href="javascript:stop_geo()">Stop Recording.</a></p>
118.
119.     <p>Your location will appear below, once approved.</p>
120.
121.     <div id="geo"></div>
122.
123.     <p><a href="show.html">Show your track!</a>
124.         (<a href="showraw.html">Or view the raw data in storage</a>)</p>
125.
126. </article>
127.
128. </body>
129. </html>
```

```
1. <!--
2.
3. Dan Armendariz
4. Computer Science 76
5. Building Mobile Applications
6. Harvard Extension School
7.
8. An HTML5 webapp that records a GPS track and stores data into localStorage.
9.
10. -->
11. <!DOCTYPE html>
12. <html>
13.   <head>
14.     <title>Show Recorded GPS Track</title>
15.     <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
16.     <link rel="stylesheet" href="style.css" />
17.     <meta name="apple-mobile-web-app-capable" content="yes" />
18.     <meta name="apple-mobile-web-app-status-bar-style" content="black" />
19.
20.     <!-- make this page mobile-friendly by allowing the map to capture zooms -->
21.     <meta name="viewport" content="initial-scale=1.0, user-scalable=no" />
22.
23.     <!-- load the Google Maps API v3 -->
24.     <script type="text/javascript" src="http://maps.google.com/maps/api/js?sensor=false"></script>
25.
26.
27.     <script type="text/javascript">
28.       // 
29.
30.       // variable name in localStorage
31.       var LOC = "gpsTrack";
32.
33.       // JSON gps track data
34.       var track = null;
35.
36.       // Google Map
37.       var map = null;
38.
39.       // returns true if the web browser supports web storage, false otherwise
40.       function supports_web_storage() {
41.         try {
42.           return 'localStorage' in window &amp;&amp; window['localStorage'] !== null;
43.         } catch (e) {
44.           return false;
45.         }
46.       }
47.
48.</pre></div>
```

```
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>
```

```
1. <!--
2.
3. Dan Armendariz
4. Computer Science 76
5. Building Mobile Applications
6. Harvard Extension School
7.
8. An HTML5 webapp that displays the raw recorded GPS track.
9.
10. -->
11. <!DOCTYPE html>
12. <html>
13.   <head>
14.     <title>Show Raw Recorded GPS Track</title>
15.     <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
16.     <meta name="viewport" content="initial-scale=1.0" />
17.     <link rel="stylesheet" href="style.css" />
18.
19.
20.     <script type="text/javascript">
21.       // 
22.
23.       // variable name in localStorage
24.       var LOC = "gpsTrack";
25.
26.       // JSON gps track data
27.       var track = null;
28.
29.       // returns true if the web browser supports web storage, false otherwise
30.       function supports_web_storage() {
31.         try {
32.           return 'localStorage' in window &amp;&amp; window['localStorage'] !== null;
33.         } catch (e) {
34.           return false;
35.         }
36.       }
37.
38.
39.       function init() {
40.         if(!supports_web_storage()) {
41.           alert("Your browser doesn't seem to support web storage!");
42.           return;
43.         }
44.
45.
46.         // make sure we have data
47.         if(localStorage[LOC] == null) {
48.           document.getElementById("geo").innerHTML = "No GPS track in storage! Try recording first.";</pre></div>
```

```
49.         return;
50.     }
51.
52.     // retrieve Local Storage data and convert it to a JSON object
53.     document.getElementById("geo").innerHTML = localStorage[LOC];
54.
55. }
56.
57.
58. // ]]>
59. </script>
60.
61. </head>
62.
63. <body onload="init()">
64.
65. <header>
66.     <h1>GPS Track Recorder</h1>
67. </header>
68.
69. <article>
70.     <header>
71.         <h1>Raw Track in Storage</h1>
72.     </header>
73.
74.     <div id="geo"></div>
75.
76. </article>
77.
78. </body>
79. </html>
```

```
1. /*
2.  Dan Armendariz
3.  Computer Science 76
4.  Building Mobile Applications
5.  Harvard Extension School
6.
7.  Simple CSS layout.
8.
9. */
10.
11. article,header { display: block; }
12.
13. html, body {
14.     margin: 0;
15.     padding: 0;
16.     background-color: #c99;
17.     font: 12px sans-serif;
18.     height: 100%;
19. }
20.
21. header {
22.     margin: 0;
23.     padding: 0;
24.     text-align: center;
25.     border-bottom: 1px solid black;
26. }
27.
28. article {
29.     border: 1px solid black;
30.     width: 75%;
31.     padding: 1em;
32.     margin: 10px auto;
33.     background-color: white;
34. }
35.
36. article header {
37.     text-align: left;
38.     border-bottom: none;
39. }
40.
41. article h1 {
42.     margin: 0;
43.     padding: 0;
44.     font-size: 20px;
45.     font-weight: bold;
46. }
```