

```
1. <!--
2.
3.  Dan Armendariz
4.  Computer Science S-76
5.  Harvard Summer School 2011
6.
7.  A basic implementation of web storage.
8.
9.  -->
10. <!DOCTYPE html>
11. <html>
12.     <head>
13.         <title>Web Storage Example</title>
14.         <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
15.         <meta name="viewport" content="initial-scale=1.0" />
16.         <link rel="stylesheet" href="style.css" />
17.
18.         <script type="text/javascript">
19.             // 
20.
21.             // demonstrate some local storage capabilities
22.             function store_test() {
23.                 // retrieve the value in local storage
24.                 alert(localStorage.getItem("foo"));
25.
26.                 // set some local storage
27.                 localStorage.setItem("foo", "test!");
28.             }
29.
30.             // ]]&gt;
31.         &lt;/script&gt;
32.
33.     &lt;/head&gt;
34.
35.     &lt;body onload="store_test()"&gt;
36.
37.         &lt;header&gt;
38.             &lt;h1&gt;Web Storage Example&lt;/h1&gt;
39.         &lt;/header&gt;
40.
41.         &lt;article&gt;
42.             &lt;header&gt;
43.                 &lt;h1&gt;Web storage&lt;/h1&gt;
44.             &lt;/header&gt;
45.
46.             &lt;p&gt;The contents of the local storage should appear in an alert box.&lt;/p&gt;
47.
48.         &lt;/article&gt;</pre></div>
```

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

```
1. <!--
2.
3.  Dan Armendariz
4.  Computer Science S-76
5.  Harvard Summer School 2011
6.
7.  A basic implementation of web storage, with some error handling.
8.
9.  -->
10. <!DOCTYPE html>
11. <html>
12.   <head>
13.     <title>Web Storage Example</title>
14.     <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
15.     <meta name="viewport" content="initial-scale=1.0" />
16.     <link rel="stylesheet" href="style.css" />
17.
18.     <script type="text/javascript">
19.       // 
20.
21.       // returns true if the web browser supports web storage, false otherwise
22.       function supports_web_storage() {
23.         try {
24.           return 'localStorage' in window &amp;&amp; window['localStorage'] !== null;
25.         } catch (e) {
26.           return false;
27.         }
28.       }
29.
30.       // demonstrate some local storage capabilities
31.       function store_test() {
32.         if(!supports_web_storage()) {
33.           alert("Your browser doesn't seem to support web storage!");
34.           return;
35.         }
36.
37.         // retrieve the value in local storage
38.         var data = localStorage.getItem("bar");
39.         if(data == null)
40.           alert("Data is null (nothing in storage for the key 'bar').");
41.         else
42.           alert(data);
43.
44.         // set some local storage
45.         localStorage.setItem("bar", "test!");
46.       }
47.
48.       // ]]&gt;</pre></div>
```

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

```
1. <!--
2.
3.  Dan Armendariz
4.  Computer Science S-76
5.  Harvard Summer School 2011
6.
7.  A basic implementation of web storage, now with more integers and different
8.  ways of accessing 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.       // 
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 &amp;&amp; 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 counter from local storage
39.         var data = localStorage["count"];
40.
41.         // reset the counter if they've not visited before
42.         if(data == null)
43.           data = 0;
44.
45.         // tell the user how many times they've visited
46.         alert("You've visited this page " + data + " times in the past!");
47.
48.         // set the counter in local storage</pre></div>
```

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

```
1. <!--
2.
3.  Dan Armendariz
4.  Computer Science S-76
5.  Harvard Summer School 2011
6.
7.  A basic implementation of web storage while protecting the type of data.
8.
9.  -->
10. <!DOCTYPE html>
11. <html>
12.   <head>
13.     <title>Web Storage Example</title>
14.     <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
15.     <meta name="viewport" content="initial-scale=1.0" />
16.     <link rel="stylesheet" href="style.css" />
17.
18.     <script type="text/javascript">
19.       // <![CDATA[
20.
21.       // returns true if the web browser supports web storage, false otherwise
22.       function supports_web_storage() {
23.         try {
24.           return 'localStorage' in window && window['localStorage'] !== null;
25.         } catch (e) {
26.           return false;
27.         }
28.       }
29.
30.       // demonstrate some local storage capabilities
31.       function store_test() {
32.         if(!supports_web_storage()) {
33.           alert("Your browser doesn't seem to support web storage!");
34.           return;
35.         }
36.
37.         // Notice the type that this (and all other) data is stored:
38.         alert(typeof localStorage["count"]);
39.
40.         // retrieve the value from local storage (safer)
41.         var data = parseInt(localStorage["count"]);
42.
43.         // reset the counter if they've not visited before
44.         if(data == null || isNaN(data))
45.           data = 0;
46.
47.         // tell the user how many times they've visited
48.         alert("You've visited this page " + data + " times in the past!");
```

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



```
1. <!--
2.
3.  Dan Armendariz
4.  Computer Science S-76
5.  Harvard Summer School 2011
6.
7.  A demonstration that local storage is not like cookies: by storing
8.  lots of data and retrieving without sending to the server.
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.       // returns true if the web browser supports web storage, false otherwise
23.       function supports_web_storage() {
24.         try {
25.           return 'localStorage' in window &amp;&amp; window['localStorage'] !== null;
26.         } catch (e) {
27.           return false;
28.         }
29.       }
30.
31.       // make sure the browser supports web storage
32.       function init() {
33.         if(!supports_web_storage()) {
34.           alert("Your browser doesn't seem to support web storage!");
35.           return;
36.         }
37.
38.         // obtain data in storage
39.         fetch();
40.       }
41.
42.       // fetch the data from local storage and place into text area
43.       function fetch() {
44.         var data = localStorage["MyText"];
45.
46.         if(data == null) {
47.           alert("No text in storage!");
48.           return;</pre></div>
```

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

```
1.  /*
2.   Dan Armendariz
3.   Computer Science S-76
4.   Harvard Summer School 2011
5.
6.   Simple CSS layout.
7.
8.  */
9.
10. article,header { display: block; }
11.
12. html, body {
13.     margin: 0;
14.     padding: 0;
15.     background-color: #c99;
16.     font: 12px sans-serif;
17.     height: 100%;
18. }
19.
20. header {
21.     margin: 0;
22.     padding: 0;
23.     text-align: center;
24.     border-bottom: 1px solid black;
25. }
26.
27. article {
28.     border: 1px solid black;
29.     width: 75%;
30.     padding: 1em;
31.     margin: 10px auto;
32.     background-color: white;
33. }
34.
35. article header {
36.     text-align: left;
37.     border-bottom: none;
38. }
39.
40. article h1 {
41.     margin: 0;
42.     padding: 0;
43.     font-size: 20px;
44.     font-weight: bold;
45. }
```

```
1. <!--
2.
3.  Dan Armendariz
4.  Computer Science S-76
5.  Harvard Summer School 2011
6.
7.  A basic implementation of geolocation.
8.
9.  -->
10. <!DOCTYPE html>
11. <html>
12.   <head>
13.     <title>Geolocation Example</title>
14.     <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
15.     <meta name="viewport" content="initial-scale=1.0" />
16.     <link rel="stylesheet" href="style.css" />
17.
18.     <script type="text/javascript">
19.       // 
20.
21.       function geo() {
22.         // perform geo lookup
23.         navigator.geolocation.getCurrentPosition(print_geo);
24.       }
25.
26.       // callback function
27.       function print_geo(pos) {
28.         // display data
29.         alert("Latitude: " + pos.coords.latitude + "\n" +
30.             "Longitude: " + pos.coords.longitude + "\n" +
31.             "Accuracy: " + pos.coords.accuracy + "\n\n" +
32.             "Timestamp: " + pos.timestamp);
33.
34.         /* Other coords properties (availability dependent on browser):
35.          - altitude (meters)
36.          - altitudeAccuracy (meters)
37.          - heading (degrees clockwise from true north)
38.          - speed (meters/second)
39.          If unavailable, the data in these properties will be null.
40.         */
41.       }
42.
43.       // ]]&gt;
44.     &lt;/script&gt;
45.
46.   &lt;/head&gt;
47.
48.   &lt;body onload="geo()"&gt;</pre></div>
```

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

```

1. <!--
2.
3.  Dan Armendariz
4.  Computer Science S-76
5.  Harvard Summer School 2011
6.
7.  A (better) implementation of geolocation that includes some error handling.
8.
9.  -->
10. <!DOCTYPE html>
11. <html>
12.     <head>
13.         <title>Geolocation Example</title>
14.         <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
15.         <meta name="viewport" content="initial-scale=1.0" />
16.         <link rel="stylesheet" href="style.css" />
17.
18.         <script type="text/javascript">
19.             // 
20.
21.             function geo() {
22.                 // perform geo lookup, but only if the geolocation object exists
23.                 if(navigator.geolocation)
24.                     navigator.geolocation.getCurrentPosition(print_geo, handler);
25.                 else
26.                     alert("Geolocation not supported by your browser!");
27.             }
28.
29.             // error handler
30.             function handler(err) {
31.                 alert("Error #" + err.code + ": " + err.message);
32.             }
33.
34.             // callback function
35.             function print_geo(pos) {
36.                 // fetch coordinates
37.
38.                 // display data
39.                 alert("Latitude: "      + pos.coords.latitude + "\n" +
40.                     "Longitude: "      + pos.coords.longitude + "\n" +
41.                     "Accuracy (in m): " + pos.coords.accuracy + "\n\n" +
42.                     "Timestamp: "      + pos.timestamp);
43.
44.                 /* Other coords properties (availability dependent on browser):
45.                  - altitude (meters)
46.                  - altitudeAccuracy (meters)
47.                  - heading (degrees clockwise from true north)
48.                  - speed (meters/second)
</pre>
</div>
```

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

```
1. <!--
2.
3.  Dan Armendariz
4.  Computer Science S-76
5.  Harvard Summer School 2011
6.
7.  An implementation of geolocation that includes (better) error handling and
8.  more legible time stamp.
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.         // set some geo options
24.         var opts = {
25.           enableHighAccuracy: true,
26.           timeout: 60000,
27.           maximumAge: 60000
28.         };
29.
30.         // perform geo lookup, but only if the geolocation object exists
31.         if(navigator.geolocation)
32.           navigator.geolocation.getCurrentPosition(print_geo, handler, opts);
33.         else
34.           alert("Geolocation not supported by your browser!");
35.       }
36.
37.       // error handler
38.       function handler(err) {
39.         switch(err.code) {
40.           case err.PERMISSION_DENIED:
41.             alert("This page is not allowed to view your position. Message: " + err.message);
42.             break;
43.           case err.POSITION_UNAVAILABLE:
44.             alert("Your position is not available. Message: " + err.message);
45.             break;
46.           case err.TIMEOUT:
47.             alert("Timeout when determining your location.");
48.             break;</pre></div>
```



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

```
1.  <!--
2.
3.  Dan Armendariz
4.  Computer Science S-76
5.  Harvard Summer School 2011
6.
7.  An implementation of geolocation that allows the site to continuously track
8.  a user.
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.       // keep track of the watchPosition ID so we can cancel it later.
23.       var watchID = null;
24.
25.       function geo() {
26.         // set some geo options
27.         var opts = {
28.           enableHighAccuracy: true,
29.           timeout: 60000,
30.           maximumAge: 60000
31.         };
32.
33.         // watch the user's location
34.         if(navigator.geolocation)
35.           watchID = navigator.geolocation.watchPosition(print_geo, handler, opts);
36.         else
37.           alert("Geolocation not supported by your browser!");
38.       }
39.
40.       // error handler
41.       function handler(err) {
42.         switch(err.code) {
43.           case err.PERMISSION_DENIED:
44.             alert("This page is not allowed to view your position. Message: " + err.message);
45.             break;
46.           case err.POSITION_UNAVAILABLE:
47.             alert("Your position is not available. Message: " + err.message);
48.             break;</pre></div>
```

```

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

```

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

```

1.  <!--
2.
3.  Dan Armendariz
4.  Computer Science S-76
5.  Harvard Summer School 2011
6.
7.  A implementation of geolocation and shown on a map.
8.
9.  -->
10. <!DOCTYPE html>
11. <html>
12.     <head>
13.         <title>Geolocation Example</title>
14.         <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
15.         <link rel="stylesheet" href="style.css" />
16.
17.         <!-- make this page mobile-friendly by allowing the map to capture zooms -->
18.         <meta name="viewport" content="initial-scale=1.0, user-scalable=no" />
19.
20.         <!-- load the Google Maps API v3 -->
21.         <script type="text/javascript" src="http://maps.google.com/maps/api/js?sensor=false"></script>
22.
23.         <script type="text/javascript">
24.             // 
25.
26.             var map;
27.
28.             // after page load, initialize the Google Map
29.             function init() {
30.
31.                 // set some options for our map
32.                 var options = {
33.                     zoom: 13,
34.                     center: new google.maps.LatLng(42.375096,-71.105607),
35.                     mapTypeId: google.maps.MapTypeId.ROADMAP
36.                 };
37.
38.                 // instantiate the map
39.                 map = new google.maps.Map(document.getElementById("map"), options);
40.
41.                 // perform geo lookup
42.                 geo();
43.             }
44.
45.             // perform geo lookup, but only if the geolocation object exists
46.             function geo() {
47.                 if(navigator.geolocation)
48.                     navigator.geolocation.getCurrentPosition(print_geo, handler);
</pre>
</div>
```

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

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

```
1. <!--
2.
3. Dan Armendariz
4. Computer Science S-76
5. Harvard Summer School 2011
6.
7. A implementation of geolocation on a map with constant updating.
8.
9. -->
10. <!DOCTYPE html>
11. <html>
12.   <head>
13.     <title>Geolocation Example</title>
14.     <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
15.     <link rel="stylesheet" href="style.css" />
16.
17.     <!-- make this page mobile-friendly by allowing the map to capture zooms -->
18.     <meta name="viewport" content="initial-scale=1.0, user-scalable=no" />
19.
20.     <!-- load the Google Maps API v3 -->
21.     <script type="text/javascript" src="http://maps.google.com/maps/api/js?sensor=false"></script>
22.
23.     <script type="text/javascript">
24.       // <![CDATA[
25.
26.       // vars for Google Maps
27.       var map, marker, circle;
28.
29.       // keep track of the watchPosition ID so we can cancel it later.
30.       var watchID = null;
31.
32.       // after page load, initialize the Google Map
33.       function init() {
34.
35.         // set some options for our map
36.         var options = {
37.           zoom: 13,
38.           center: new google.maps.LatLng(42.375096,-71.105607),
39.           mapTypeId: google.maps.MapTypeId.ROADMAP
40.         };
41.
42.         // instantiate the map
43.         map = new google.maps.Map(document.getElementById("map"), options);
44.
45.         // perform geo lookup
46.         geo();
47.       }
48.
```



```
49. // perform geo lookup, but only if the geolocation object exists
50. function geo() {
51.
52.     var opts = {
53.         enableHighAccuracy: true,
54.         timeout: 60000,
55.         maximumAge: 60000
56.     };
57.
58.     if(navigator.geolocation)
59.         watchID = navigator.geolocation.watchPosition(map_geo, handler, opts);
60.     else
61.         alert("Geolocation not supported by your browser!");
62. }
63.
64. // geolocation error handler
65. function handler(err) {
66.     alert("Error #" + err.code + ": " + err.message);
67.
68.     // stop the watch, or iPhones will continuously show an error
69.     navigator.geolocation.clearWatch(watchID);
70. }
71.
72. // geolocation callback function
73. function map_geo(pos) {
74.
75.     // fetch coordinates and store into a Google Maps LatLng object
76.     var latlng = new google.maps.LatLng(pos.coords.latitude, pos.coords.longitude);
77.     var accuracy = pos.coords.accuracy;
78.
79.     // nothing's been placed on the map yet, so let's create them
80.     if(marker == null) {
81.
82.         // display a marker at the location
83.         marker = new google.maps.Marker({
84.             map: map,
85.             draggable: false,
86.             position: latlng
87.         });
88.
89.         // display a circle showing the accuracy radius
90.         circle = new google.maps.Circle({
91.             map: map,
92.             clickable: false,
93.             fillColor: "#CC0000",
94.             fillOpacity: 0.15,
95.             strokeColor: "#FF0000",
96.             strokeOpacity: 0.25,
```

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

```
1. /*
2.   Dan Armendariz
3.   Computer Science S-76
4.   Harvard Summer School 2011
5.
6.   Simple CSS layout.
7.
8. */
9.
10. article,header { display: block; }
11.
12. html, body {
13.     margin: 0;
14.     padding: 0;
15.     background-color: #c99;
16.     font: 12px sans-serif;
17.     height: 100%;
18. }
19.
20. header {
21.     margin: 0;
22.     padding: 0;
23.     text-align: center;
24.     border-bottom: 1px solid black;
25. }
26.
27. article {
28.     border: 1px solid black;
29.     width: 75%;
30.     padding: 1em;
31.     margin: 10px auto;
32.     background-color: white;
33. }
34.
35. article header {
36.     text-align: left;
37.     border-bottom: none;
38. }
39.
40. article h1 {
41.     margin: 0;
42.     padding: 0;
43.     font-size: 20px;
44.     font-weight: bold;
45. }
```

```
1. <!--
2.
3.  Dan Armendariz
4.  Computer Science S-76
5.  Harvard Summer School 2011
6.
7.  An HTML5 webapp that records a GPS track and stores data into localStorage.
8.
9.  -->
10. <!DOCTYPE html>
11. <html>
12.   <head>
13.     <title>GPS Track Recorder</title>
14.     <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
15.     <meta name="viewport" content="initial-scale=1.0" />
16.     <link rel="stylesheet" href="style.css" />
17.
18.     <script type="text/javascript">
19.       // 
20.
21.       // keep track of the watchPosition ID so we can cancel it later.
22.       var watchID = null;
23.
24.       // variable name in localStorage
25.       var LOC = "gpsTrack";
26.
27.       var track = new Array();
28.
29.       function geo() {
30.         // reset localStorage
31.         localStorage.removeItem(LOC);
32.
33.         // set some geo options
34.         var opts = {
35.           enableHighAccuracy: true,
36.           timeout: 60000,
37.           maximumAge: 10000
38.         };
39.
40.         // watch the user's location
41.         if(navigator.geolocation)
42.           watchID = navigator.geolocation.watchPosition(record_geo, handler, opts);
43.         else
44.           alert("Geolocation not supported by your browser!");
45.       }
46.
47.       // error handler
48.       function handler(err) {</pre></div>
```

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

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

```
1. <!--
2.
3. Dan Armendariz
4. Computer Science S-76
5. Harvard Summer School 2011
6.
7. An HTML5 webapp that records a GPS track and stores data into localStorage.
8.
9. -->
10. <!DOCTYPE html>
11. <html>
12.   <head>
13.     <title>Show Recorded GPS Track</title>
14.     <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
15.     <link rel="stylesheet" href="style.css" />
16.     <meta name="apple-mobile-web-app-capable" content="yes" />
17.     <meta name="apple-mobile-web-app-status-bar-style" content="black" />
18.
19.     <!-- make this page mobile-friendly by allowing the map to capture zooms -->
20.     <meta name="viewport" content="initial-scale=1.0, user-scalable=no" />
21.
22.     <!-- load the Google Maps API v3 -->
23.     <script type="text/javascript" src="http://maps.google.com/maps/api/js?sensor=false"></script>
24.
25.
26.     <script type="text/javascript">
27.       // 
28.
29.       // variable name in localStorage
30.       var LOC = "gpsTrack";
31.
32.       // JSON gps track data
33.       var track = null;
34.
35.       // Google Map
36.       var map = null;
37.
38.       // returns true if the web browser supports web storage, false otherwise
39.       function supports_web_storage() {
40.         try {
41.           return 'localStorage' in window &amp;&amp; window['localStorage'] !== null;
42.         } catch (e) {
43.           return false;
44.         }
45.       }
46.
47.       function init() {</pre></div>
```

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



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

```
1. <!--
2.
3.  Dan Armendariz
4.  Computer Science S-76
5.  Harvard Summer School 2011
6.
7.  An HTML5 webapp that displays the raw recorded GPS track.
8.
9.  -->
10. <!DOCTYPE html>
11. <html>
12.   <head>
13.     <title>Show Raw Recorded GPS Track</title>
14.     <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
15.     <meta name="viewport" content="initial-scale=1.0" />
16.     <link rel="stylesheet" href="style.css" />
17.
18.
19.     <script type="text/javascript">
20.       // <![CDATA[
21.
22.       // variable name in localStorage
23.       var LOC = "gpsTrack";
24.
25.       // JSON gps track data
26.       var track = null;
27.
28.       // returns true if the web browser supports web storage, false otherwise
29.       function supports_web_storage() {
30.         try {
31.           return 'localStorage' in window && window['localStorage'] !== null;
32.         } catch (e) {
33.           return false;
34.         }
35.       }
36.
37.
38.       function init() {
39.         if(!supports_web_storage()) {
40.           alert("Your browser doesn't seem to support web storage!");
41.           return;
42.         }
43.
44.
45.         // make sure we have data
46.         if(localStorage[LOC] == null) {
47.           document.getElementById("geo").innerHTML = "No GPS track in storage! Try recording first.";
48.           return;
```

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

```
1. /*
2.  Dan Armendariz
3.  Computer Science S-76
4.  Harvard Summer School 2011
5.
6.  Simple CSS layout.
7.
8. */
9.
10. article,header { display: block; }
11.
12. html, body {
13.     margin: 0;
14.     padding: 0;
15.     background-color: #c99;
16.     font: 12px sans-serif;
17.     height: 100%;
18. }
19.
20. header {
21.     margin: 0;
22.     padding: 0;
23.     text-align: center;
24.     border-bottom: 1px solid black;
25. }
26.
27. article {
28.     border: 1px solid black;
29.     width: 75%;
30.     padding: 1em;
31.     margin: 10px auto;
32.     background-color: white;
33. }
34.
35. article header {
36.     text-align: left;
37.     border-bottom: none;
38. }
39.
40. article h1 {
41.     margin: 0;
42.     padding: 0;
43.     font-size: 20px;
44.     font-weight: bold;
45. }
```