

```
1. <?php
2.
3.     /**
4.      * counter.php
5.      *
6.      * David J. Malan
7.      * malan@harvard.edu
8.      *
9.      * Implements a counter. Demonstrates sessions.
10.     */
11.
12.     // enable sessions
13.     session_start();
14.
15.     // check counter
16.     if (isset($_SESSION["counter"]))
17.     {
18.         $counter = $_SESSION["counter"];
19.     }
20.     else
21.     {
22.         $counter = 0;
23.     }
24.
25.     // increment counter
26.     $_SESSION["counter"] = $counter + 1;
27.
28. ?>
29.
30. <!DOCTYPE html>
31.
32. <html>
33.     <head>
34.         <title>counter</title>
35.     </head>
36.     <body>
37.         You have visited this site <?= $counter ?> time(s).
38.     </body>
39. </html>
```

```
1. <!--
2.
3. ajax-0.html
4.
5. Gets stock quote from quote.php via Ajax with jQuery, displaying price with alert.
6.
7. David J. Malan
8. malan@harvard.edu
9.
10. -->
11.
12. <!DOCTYPE html>
13.
14. <html>
15.   <head>
16.     <script src="http://code.jquery.com/jquery-latest.min.js"></script>
17.     <script>
18.
19.       /**
20.        * Gets a quote via JSON.
21.        */
22.       function quote()
23.       {
24.         var url = 'quote.php?symbol=' + $('#symbol').val();
25.         $.getJSON(url, function(data) {
26.           alert(data.price);
27.         });
28.       }
29.
30.     </script>
31.     <title>ajax-0</title>
32.   </head>
33.   <body>
34.     <form onsubmit="quote(); return false;">
35.       Symbol: <input autocomplete="off" id="symbol" type="text"/>
36.       <br/><br/>
37.       <input type="submit" value="Get Quote"/>
38.     </form>
39.   </body>
40. </html>
```

```
1. <!--
2.
3. ajax-1.html
4.
5. Gets stock quote from quote.php via Ajax with jQuery, displaying price with alert.
6.
7. David J. Malan
8. malan@harvard.edu
9.
10. -->
11.
12. <!DOCTYPE html>
13.
14. <html>
15.     <head>
16.         <script src="http://code.jquery.com/jquery-latest.min.js"></script>
17.         <script>
18.
19.             $(function() {
20.
21.                 $('#quote').submit(function() {
22.
23.                     var url = 'quote.php?symbol=' + $('#symbol').val();
24.                     $.getJSON(url, function(data) {
25.                         alert(data.price);
26.                     });
27.                     return false;
28.
29.                 });
30.
31.             });
32.
33.         </script>
34.         <title>ajax-1</title>
35.     </head>
36.     <body>
37.         <form id="quote">
38.             Symbol: <input autocomplete="off" id="symbol" type="text"/>
39.             <br/><br/>
40.             <input type="submit" value="Get Quote"/>
41.         </form>
42.     </body>
43. </html>
```

```
1. <!--
2.
3. ajax-2.html
4.
5. Gets stock quote from quote.php via Ajax with jQuery, embedding result in page itself.
6.
7. David J. Malan
8. malan@harvard.edu
9.
10. -->
11.
12. <!DOCTYPE html>
13.
14. <html>
15.   <head>
16.     <script src="http://code.jquery.com/jquery-latest.min.js"></script>
17.     <script>
18.
19.       /**
20.        * Gets a quote via JSON.
21.        */
22.       function quote()
23.       {
24.         var url = 'quote.php?symbol=' + $('#symbol').val();
25.         $.getJSON(url, function(data) {
26.           $('#price').html(data.price);
27.         });
28.       }
29.
30.     </script>
31.     <title>ajax-2</title>
32.   </head>
33.   <body>
34.     <form onsubmit="quote(); return false;">
35.       Symbol: <input autocomplete="off" id="symbol" type="text"/>
36.       <br/>
37.       Price: <span id="price">to be determined</span>
38.       <br/><br/>
39.       <input type="submit" value="Get Quote"/>
40.     </form>
41.   </body>
42. </html>
```

```
1.
2. <!--
3.
4. blink.html
5.
6. Flashes a greeting.
7.
8. Computer Science 50
9. David J. Malan
10.
11. -->
12.
13. <!DOCTYPE html>
14.
15. <html>
16.   <head>
17.     <script>
18.
19.       // toggles visibility of greeting
20.       function blink()
21.       {
22.         var div = document.getElementById('greeting');
23.         if (div.style.visibility == "hidden")
24.         {
25.           div.style.visibility = "visible";
26.         }
27.         else
28.         {
29.           div.style.visibility = "hidden";
30.         }
31.       }
32.
33.       // blink every 500ms
34.       window.setInterval(blink, 500);
35.
36.     </script>
37.     <style>
38.
39.       #greeting
40.       {
41.         font-size: 96pt;
42.         margin: 240px;
43.         text-align: center;
44.       }
45.
46.     </style>
47.     <title>blink</title>
48.   </head>
```

```
49.     <body>
50.         <div id="greeting">
51.             hello, world
52.         </div>
53.     </body>
54. </html>
```

```
1. <!DOCTYPE html>
2.
3. <html>
4.   <head>
5.     <script>
6.
7.       function greet()
8.       {
9.         alert('hello, ' + document.getElementById('name').value + '!');
10.      }
11.
12.    </script>
13.    <title>dom-0</title>
14.  </head>
15.  <body>
16.    <form id="demo" onsubmit="greet(); return false;">
17.      <input id="name" placeholder="Name" type="text"/>
18.      <input type="submit"/>
19.    </form>
20.  </body>
21. </html>
```

```
1. <!DOCTYPE html>
2.
3. <html>
4.   <head>
5.     <title>dom-1</title>
6.   </head>
7.   <body>
8.     <form id="demo">
9.       <input id="name" placeholder="Name" type="text"/>
10.      <input type="submit"/>
11.    </form>
12.    <script>
13.
14.      document.getElementById('demo').onsubmit = function() {
15.        alert('hello, ' + document.getElementById('name').value + '!');
16.        return false;
17.      };
18.
19.    </script>
20.  </body>
21. </html>
```



```
1. <!DOCTYPE html>
2.
3. <html>
4.   <head>
5.     <script src="http://code.jquery.com/jquery-latest.min.js"></script>
6.     <script>
7.
8.       $(document).ready(function() {
9.         $('#demo').submit(function(event) {
10.           alert('hello, ' + $('#name').val() + '!');
11.           event.preventDefault();
12.         });
13.       });
14.
15.     </script>
16.     <title>dom-2</title>
17.   </head>
18.   <body>
19.     <form id="demo">
20.       <input id="name" placeholder="Name" type="text"/>
21.       <input type="submit"/>
22.     </form>
23.   </body>
24. </html>
```

```
1.
2. <!--
3.
4. form-0.html
5.
6. A form without client-side validation.
7.
8. David J. Malan
9. malan@harvard.edu
10.
11. -->
12.
13. <!DOCTYPE html>
14.
15. <html>
16.     <head>
17.         <title>form-0</title>
18.     </head>
19.     <body>
20.         <form action="register.php" method="get">
21.             Email: <input name="email" type="text"/>
22.             <br/>
23.             Password: <input name="password" type="password"/>
24.             <br/>
25.             Password (again): <input name="confirmation" type="password"/>
26.             <br/>
27.             I agree to the terms and conditions: <input name="agreement" type="checkbox"/>
28.             <br/><br/>
29.             <input type="submit" value="Register"/>
30.         </form>
31.     </body>
32. </html>
```

```

1. <!--
2. <!--
3.
4. form-1.html
5.
6. A form with client-side validation.
7.
8. David J. Malan
9. malan@harvard.edu
10.
11. -->
12.
13. <!DOCTYPE html>
14.
15. <html>
16.     <head>
17.         <title>form-1</title>
18.     </head>
19.     <body>
20.         <form action="register.php" id="registration" method="get">
21.             Email: <input name="email" type="text"/>
22.             <br/>
23.             Password: <input name="password" type="password"/>
24.             <br/>
25.             Password (again): <input name="confirmation" type="password"/>
26.             <br/>
27.             I agree to the terms and conditions: <input name="agreement" type="checkbox"/>
28.             <br/><br/>
29.             <input type="submit" value="Register"/>
30.         </form>
31.         <script>
32.
33.             var form = document.getElementById('registration');
34.
35.             // onsubmit
36.             form.onsubmit = function() {
37.
38.                 // validate email
39.                 if (form.email.value == '')
40.                 {
41.                     alert('You must provide your email address!');
42.                     return false;
43.                 }
44.
45.                 // validate password
46.                 else if (form.password.value == '')
47.                 {
48.                     alert('You must provide a password!');

```

```
49.         return false;
50.     }
51.
52.     // validate confirmation
53.     else if (form.password.value != form.confirmation.value)
54.     {
55.         alert('Passwords do not match!');
56.         return false;
57.     }
58.
59.     // validate agreement
60.     else if (!form.agreement.checked)
61.     {
62.         alert('You must agree to the terms and conditions!');
63.         return false;
64.     }
65.
66.     // valid!
67.     return true;
68.
69.     };
70.
71.     </script>
72. </body>
73. </html>
```

```
1.
2. <!--
3.
4. form-2.html
5.
6. A form with client-side validation using jQuery.
7.
8. David J. Malan
9. malan@harvard.edu
10.
11. -->
12.
13. <!DOCTYPE html>
14.
15. <html>
16.     <head>
17.         <script src="http://code.jquery.com/jquery-latest.min.js"></script>
18.         <script>
19.
20.             // onready
21.             $(document).ready(function() {
22.
23.                 // onsubmit
24.                 $('#registration').submit(function() {
25.
26.                     // validate email
27.                     if ($('#registration input[name=email]').val() == '')
28.                     {
29.                         alert('You must provide your email address!');
30.                         return false;
31.                     }
32.
33.                     // validate password
34.                     else if ($('#registration input[name=password]').val() == '')
35.                     {
36.                         alert('You must provide a password!');
37.                         return false;
38.                     }
39.
40.                     // validate confirmation
41.                     else if ($('#registration input[name=password]').val() != $('#registration input[name=confirmation]').val())
42.                     {
43.                         alert('Passwords do not match!');
44.                         return false;
45.                     }
46.
47.                     // validate agreement
48.                     else if (!$('#registration input[name=agreement]').is(':checked'))
```

```
49.         {
50.             alert('You must agree to the terms and conditions!');
51.             return false;
52.         }
53.
54.         // valid!
55.         return true;
56.
57.     });
58.
59. });
60.
61. </script>
62. <title>form-2</title>
63. </head>
64. <body>
65.     <form action="register.php" id="registration" method="get">
66.         Email: <input name="email" type="text"/>
67.         <br/>
68.         Password: <input name="password" type="password"/>
69.         <br/>
70.         Password (again): <input name="confirmation" type="password"/>
71.         <br/>
72.         I agree to the terms and conditions: <input name="agreement" type="checkbox"/>
73.         <br/><br/>
74.         <input type="submit" value="Register"/>
75.     </form>
76. </body>
77. </html>
```

```
1. <!--
2.
3. geolocation-0.html
4.
5. Geolocates a user.
6.
7. David J. Malan
8. malan@harvard.edu
9.
10. -->
11.
12. <!DOCTYPE html>
13.
14. <html>
15.     <head>
16.         <script>
17.
18.             function callback(position)
19.             {
20.                 alert(position.coords.latitude + ', ' + position.coords.longitude);
21.             }
22.
23.             function geolocate()
24.             {
25.                 if (typeof(navigator.geolocation) != 'undefined')
26.                 {
27.                     navigator.geolocation.getCurrentPosition(callback);
28.                 }
29.                 else
30.                 {
31.                     alert('Your browser does not support geolocation!');
32.                 }
33.             }
34.
35.         </script>
36.         <title>geolocation-0</title>
37.     </head>
38.     <body onload="geolocate()"></body>
39. </html>
```

```
1. <!--
2.
3. geolocation-1.html
4.
5. Geolocates a user, demonstrating an anonymous function.
6.
7. David J. Malan
8. malan@harvard.edu
9.
10. -->
11.
12. <!DOCTYPE html>
13.
14. <html>
15.     <head>
16.         <script>
17.
18.             function geolocate()
19.             {
20.                 if (typeof(navigator.geolocation) != 'undefined')
21.                 {
22.                     navigator.geolocation.getCurrentPosition(function(position) {
23.                         alert(position.coords.latitude + ', ' + position.coords.longitude);
24.                     });
25.                 }
26.                 else
27.                 {
28.                     alert('Your browser does not support geolocation!');
29.                 }
30.             }
31.
32.         </script>
33.         <title>geolocation-1</title>
34.     </head>
35.     <body onload="geolocate()"></body>
36. </html>
```



```
1. <?php
2.
3.     // open connection to Yahoo
4.     $handle = @fopen("http://download.finance.yahoo.com/d/quotes.csv?f=snl1&s={$_GET["symbol"]}", "r");
5.     if ($handle === false)
6.     {
7.         // trigger (big, orange) error
8.         trigger_error("Could not connect to Yahoo!", E_USER_ERROR);
9.         exit;
10.    }
11.
12.    // download first line of CSV file
13.    $data = fgetcsv($handle);
14.    if ($data === false || count($data) == 1)
15.    {
16.        return false;
17.    }
18.
19.    // close connection to Yahoo
20.    fclose($handle);
21.
22.    // ensure symbol was found
23.    if ($data[2] === "0.00")
24.    {
25.        return false;
26.    }
27.
28.    // prepare stock as an associative array
29.    $stock = [
30.        "symbol" => $data[0],
31.        "name" => $data[1],
32.        "price" => $data[2],
33.    ];
34.
35.    // output stock as JSON
36.    header("Content-Type: application/json");
37.    print(json_encode($stock));
38.
39. ?>
```

```
1. <!DOCTYPE html>
2.
3. <html>
4.   <head>
5.     <title>register</title>
6.   </head>
7.   <body>
8.     You are registered! (Well, not really.)
9.   </body>
10. </html>
```

```
1. <!DOCTYPE html>
2.
3. <html>
4.   <head>
5.     <meta name="viewport" content="width=device-width">
6.     <script>
7.
8.       function greet()
9.       {
10.         if (typeof(localStorage) !== 'undefined')
11.         {
12.           if (typeof(localStorage['counter']) === 'undefined')
13.           {
14.             localStorage['counter'] = 0;
15.           }
16.           alert('You have visited ' + localStorage['counter'] + ' time(s) before!');
17.           localStorage['counter']++;
18.         }
19.         else
20.         {
21.           alert('Your browser does not support localStorage!');
22.         }
23.       }
24.
25.     </script>
26.     <title>storage</title>
27.   </head>
28.   <body onload="greet()"></body>
29. </html>
```

1. foo

```
1. <?php
2.
3. /**
4.  * dictionary.php
5.  *
6.  * David J. Malan
7.  * malan@harvard.edu
8.  *
9.  * Implements a dictionary in a (non-object-oriented) way that
10. * mimics Problem Set 6's implementation in C. However, an
11. * object-oriented design would be better in PHP.
12. */
13.
14. // size of dictionary
15. $size = 0;
16.
17. // hash table
18. $table = [];
19.
20. /**
21.  * Returns true if word is in dictionary else false.
22.  */
23. function check($word)
24. {
25.     global $table;
26.     if (isset($table[strtolower($word)]))
27.     {
28.         return true;
29.     }
30.     else
31.     {
32.         return false;
33.     }
34. }
35.
36. /**
37.  * Loads dictionary into memory. Returns true if successful else false.
38.  */
39. function load($dictionary)
40. {
41.     global $table, $size;
42.     if (!file_exists($dictionary) && is_readable($dictionary))
43.     {
44.         return false;
45.     }
46.     foreach (file($dictionary) as $word)
47.     {
48.         $table[chop($word)] = true;
```

```
49.         $size++;
50.     }
51.     return true;
52. }
53.
54. /**
55.  * Returns number of words in dictionary if loaded else 0 if not yet loaded.
56.  */
57. function size()
58. {
59.     global $size;
60.     return $size;
61. }
62.
63. /**
64.  * Unloads dictionary from memory. Returns true if successful else false.
65.  */
66. function unload()
67. {
68.     return true;
69. }
70.
71. ?>
```

```
1. #!/bin/env php
2. <?php
3.
4. /*****
5.  * speller.php
6.  *
7.  * David J. Malan
8.  * malan@harvard.edu
9.  *
10. * Implements a spell-checker.
11. *****/
12.
13. require("dictionary.php");
14.
15. // maximum length for a word
16. // (e.g., pneumonoultramicroscopicsilicovolcanoconiosis)
17. define("LENGTH", 45);
18.
19. // default dictionary
20. define("WORDS", "/home/cs50/pset6/dictionaries/large");
21.
22. // check for correct number of args
23. if ($argc != 2 && $argc != 3)
24. {
25.     print("Usage: speller [dictionary] text\n");
26.     return 1;
27. }
28.
29. // benchmarks
30. $ti_load = 0.0; $ti_check = 0.0; $ti_size = 0.0; $ti_unload = 0.0;
31.
32. // determine dictionary to use
33. $dictionary = ($argc == 3) ? $argv[1] : WORDS;
34.
35. // load dictionary
36. $before = microtime(true);
37. $loaded = load($dictionary);
38. $after = microtime(true);
39.
40. // abort if dictionary not loaded
41. if (!$loaded)
42. {
43.     print("Could not load $dictionary.\n");
44.     return 1;
45. }
46.
47. // calculate time to load dictionary
48. $ti_load = $after - $before;
```

```
49.
50. // try to open file
51. $file = ($argc == 3) ? $argv[2] : $argv[1];
52. $fp = fopen($file, "r");
53. if ($fp === false)
54. {
55.     print("Could not open $file.\n");
56.     return 1;
57. }
58.
59. // prepare to report misspellings
60. printf("\nMISSPELLED WORDS\n\n");
61.
62. // prepare to spell-check
63. $word = "";
64. $index = 0; $misspellings = 0; $words = 0;
65.
66. // spell-check each word in file
67. for ($c = fgetc($fp); $c !== false; $c = fgetc($fp))
68. {
69.     // allow alphabetical characters and apostrophes (for possessives)
70.     if (preg_match("/[a-zA-Z]/", $c) || ($c == "'" && $index > 0))
71.     {
72.         // append character to word
73.         $word .= $c;
74.         $index++;
75.
76.         // ignore alphabetical strings too long to be words
77.         if ($index >= LENGTH)
78.         {
79.             // consume remainder of alphabetical string
80.             while (($c = fgetc($fp)) !== false && preg_match("/[a-zA-Z]/", $c));
81.
82.             // prepare for new word
83.             $index = 0; $word = "";
84.         }
85.     }
86.
87.     // ignore words with numbers (like MS Word)
88.     else if (ctype_digit($c))
89.     {
90.         // consume remainder of alphabetical string
91.         while (($c = fgetc($fp)) !== false && preg_match("/[a-zA-z0-9]/", $c));
92.
93.         // prepare for new word
94.         $index = 0; $word = "";
95.     }
96.
```



```
97.         // we must have found a whole word
98.     else if ($index > 0)
99.     {
100.         // update counter
101.         $words++;
102.
103.         // check word's spelling
104.         $before = microtime(true);
105.         $misspelled = !check($word);
106.         $after = microtime(true);
107.
108.         // update benchmark
109.         $ti_check += $after - $before;
110.
111.         // print word if misspelled
112.         if ($misspelled)
113.         {
114.             print("$word\n");
115.             $mispellings++;
116.         }
117.
118.         // prepare for next word
119.         $index = 0; $word = "";
120.     }
121. }
122.
123. // close file
124. fclose($fp);
125.
126. // determine dictionary's size
127. $before = microtime(true);
128. $n = size();
129. $after = microtime(true);
130.
131. // calculate time to determine dictionary's size
132. $ti_size = $after - $before;
133.
134. // unload dictionary
135. $before = microtime(true);
136. $unloaded = unload();
137. $after = microtime(true);
138.
139. // abort if dictionary not unloaded
140. if (!$unloaded)
141. {
142.     print("Could not load $dictionary.\n");
143.     return 1;
144. }
```

```
145.    // calculate time to determine dictionary's size
146.    $ti_unload = $after - $before;
147.
148.    // report benchmarks
149.    printf("\nWORDS MISSPELLED:    %d\n", $mispellings);
150.    printf("WORDS IN DICTIONARY:  %d\n", $n);
151.    printf("WORDS IN TEXT:        %d\n", $words);
152.    printf("TIME IN load:         %.2f\n", $ti_load);
153.    printf("TIME IN check:        %.2f\n", $ti_check);
154.    printf("TIME IN size:         %.2f\n", $ti_size);
155.    printf("TIME IN unload:       %.2f\n", $ti_unload);
156.    printf("TOTAL TIME:          %.2f\n\n", $ti_load + $ti_check + $ti_size + $ti_unload);
157.
158.    ?>
```

1. foo bar