```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Using JavaScript String Functions</title>
<!--
  cssdweb.edu/StudentResources/CodeSamples/UsingJavaScriptStringFunctions_1.html
  Copyright (c) 2022 by Jesse Heines.  All rights reserved.  May be freely
    copied or excerpted for educational purposes with credit to the author.
  updated by JMH on May 18, 2022 at 5:58 PM
-->
  <style>
    p, blockquote, ol {
      margin-top: 0 ;
      margin-bottom: 0 ;
    }
    pre, code {
      margin-bottom: 0 ;
      font-size: 1.1em ;
      font-style: normal ;
    }
  </style>
</head>

<body>
  <h3>JavaScript String Functions Demonstration</h3>

  <pre>(1) var str="New Hampshire Dept. of Corrections Special School District" ;</pre>
  <p>Original string:&nbsp;<strong>
  <script>
    var str="New Hampshire Dept. of Corrections Special School District" ;
    document.writeln( str ) ;
  </script>
  </strong></p>

  <pre>(2) var length1 = str.length ;</pre>
  <p>The original string contains <strong>
  <script>
    var length1 = str.length ;
    document.writeln( length1 ) ;
  </script>
  </strong> characters.</p>
  <blockquote><em>Note that the string length (character count) includes spaces.&nbsp;
    Also note that <code>length</code> is a <strong>property</strong>, not a
    function.&nbsp; This is why <code>length</code> is not followed by
    parentheses.</em></blockquote>

  <pre>(3) var space1 = str.indexOf( " " ) ;</pre>
  <p>The first space is at character position:&nbsp;<strong>
  <script>
    var space1 = str.indexOf( " " ) ;
    document.writeln( space1 ) ;
  </script>
  </strong></p>
  <blockquote><em>Note that the character position is 0-based!</em></blockquote>

  <pre>(4) var word1 = str.substr( 0, space1 ) ;</pre>
  <p>The first word is:&nbsp;<strong>
  <script>
    var word1 = str.substr( 0, space1 ) ;
    document.writeln( word1 ) ;
  </script>
  </strong></p>
```

```
<blockquote><em>The</em> <code>substr</code> <em>parameters are the starting position
  and the number of characters.</em></blockquote>

<pre>(5) var space2 = str.indexOf( " ", space1+1 ) ;</pre>
<p>The second space is at character position:&nbsp;<strong>
<script>
  var space2 = str.indexOf( " ", space1+1 ) ;
  document.writeln( space2 ) ;
</script>
</strong></p>
<blockquote><em>The second parameter tells JavaScript where to start the character
  search.</em></blockquote>

<pre>(6) var word2 = str.substr( space1+1, space2-space1 ) ;</pre>
<p>The second word &mdash; extracted using the <code>substr</code> function
  &mdash; is:&nbsp;<strong>
<script>
  var word2 = str.substr( space1+1, space2-space1 ) ;
  document.writeln( word2 ) ;
</script>
</strong></p>
<blockquote><em>Remember that the second parameter to the</em> <code>substr</code>
  <em>function is the <strong>number of characters</strong>, not the ending space.&nbsp;
  This is why we have to subtract</em> <code>space1</code> <em>from</em>
  <code>space2</code>.</blockquote>

<pre>(7) var word2b = str.substring( space1+1, space2 ) ;</pre>
<p>The second word &mdash; extracted using the <code>substring</code> function &mdash;
  is:&nbsp;<strong>
<script>
  var word2b = str.substring( space1+1, space2 ) ;
  document.writeln( word2b ) ;
</script>
</strong></p>
<blockquote><em>The second parameter to the</em> <code>substring</code> <em>function is
  indeed the ending space.&nbsp; This is why we do <strong>not</strong> subtract</em>
  <code>space1</code> <em>from</em> <code>space2</code>.</blockquote>

<pre>(8) var phrase1 = word2 + " " + word1 ;</pre>
<p>Concatenating <code>word1</code> onto <code>word2</code> yields:&nbsp;<strong>
<script>
  var phrase1 = word2 + " " + word1 ;
  document.writeln( phrase1 ) ;
</script>
</strong></p>
<blockquote><em>Remember that the second parameter is the number of characters, not the
  ending space.&nbsp; This is why we have to subtract</em> <code>space1</code>
  <em>from</em> <code>space2</code>.</blockquote>

<pre>(9) var arrWords = str.split( " " ) ;</pre>
<p>We can get all the words at once using the <code>split</code> function:</p>
<strong>
<script>
  var arrWords = str.split( " " ) ;
  document.writeln( "<ol start='0'>" ) ;
  for ( var k = 0 ; k < arrWords.length ; k++ ) {
    document.writeln( "<li>" + arrWords[k] + "</li>" ) ;
  }
  document.writeln( "</ol>" ) ;
</script>
</strong>
<blockquote><em>Note that array indexes are 0-based!</em></blockquote>

<pre>(10) document.writeln( arrWords[5] + " " + arrWords[7] + " " + arrWords[3] + " " +
  arrWords[1] + " " + arrWords[0] + " " + arrWords[6] + " " + arrWords[4] ) ;</pre>
<p>We can then print them any way we want!&nbsp;<strong></p>
<script>
  document.writeln( "&nbsp; &nbsp; &nbsp; &nbsp; " +
      arrWords[5] + " " + arrWords[7] + " " + arrWords[3] + " " +
      arrWords[1] + " " + arrWords[0] + " " + arrWords[6] + " " + arrWords[4] ) ;
</script>
</strong>

<pre>(11) var phrase2 = str.replaceAll( " ", "|" ) ;</pre>
<p>Replacing all spaces with vertical bars yields:&nbsp;<strong>
<script>
  var phrase2 = str.replaceAll( " ", "|" ) ;
  document.writeln( phrase2 ) ;
</script>
</strong></p>
<blockquote><em>The first parameter is the search string and the second is the
  replacement string.</em></blockquote>

<pre>(12) var bStartsWithNew = str.startsWith( "New" ) ;</pre>
<p>Testing whether the string starts with <strong>New</strong> yields:&nbsp;<strong>
<script>
  var bStartsWithNew = str.startsWith( "New" ) ;
  document.writeln( bStartsWithNew ) ;
</script>
</strong></p>

<pre>(13) var bEndsWithNew = str.startsWith( "Hampshire" ) ;</pre>
<p>Testing whether the string ends with <strong>Hampshire</strong> yields:&nbsp;<strong>
<script>
  var bEndsWithNew = str.startsWith( "Hampshire" ) ;
  document.writeln( bEndsWithNew ) ;
</script>
</strong></p>

<pre>(14) var strUpper = str.toUpperCase() ;</pre>
<p>Converting the string to all uppercase yields:&nbsp;<strong>
<script>
  var strUpper = str.toUpperCase() ;
  document.writeln( strUpper ) ;
</script>
</strong></p>

<pre>(15) var strLower = str.toLowerCase() ;</pre>
<p>Converting the string to all lowercase yields:&nbsp;<strong>
<script>
  var strLower = str.toLowerCase() ;
  document.writeln( strLower ) ;
</script>
</strong></p>

<pre>(16) var bIncludesSchool1 = str.includes( "School" ) ;</pre>
<p>Testing whether the string includes (contains) <strong>School</strong>
 yields:&nbsp;<strong>
<script>
  var bIncludesSchool1 = str.includes( "School" ) ;
  document.writeln( bIncludesSchool1 ) ;
</script>
</strong></p>
<blockquote><em>Note that the <code>includes</code> function is case-sensitive.&nbsp;
  This is verified by the next example.</em></blockquote>

<pre>(17) var bIncludesSchool2 = str.includes( "school" ) ;</pre>
<p>Testing whether the string includes <strong>school</strong> yields:&nbsp;<strong>
```

```
  <script>
    var bIncludesSchool2 = str.includes( "school" ) ;
    document.writeln( bIncludesSchool2 ) ;
  </script>
  </strong></p>
</body>
</html>
```

JavaScript String Functions Demonstration

```
(1) var str="New Hampshire Dept. of Corrections Special School District" ;
```

Original string: **New Hampshire Dept. of Corrections Special School District**

```
(2) var length1 = str.length ;
```

The original string contains **58** characters.

> *Note that the string length (character count) includes spaces. Also note that* `length` *is a* ***property****, not a function. This is why* `length` *is not followed by parentheses.*

```
(3) var space1 = str.indexOf( " " ) ;
```

The first space is at character position: **3**

> *Note that the character position is 0-based!*

```
(4) var word1 = str.substr( 0, space1 ) ;
```

The first word is: **New**

> *The* `substr` *parameters are the starting position and the number of characters.*

```
(5) var space2 = str.indexOf( " ", space1+1 ) ;
```

The second space is at character position: **13**

> *The second parameter tells JavaScript where to start the character search.*

```
(6) var word2 = str.substr( space1+1, space2-space1 ) ;
```

The second word — extracted using the `substr` function — is: **Hampshire**

> *Remember that the second parameter to the* `substr` *function is the* ***number of characters****, not the ending space. This is why we have to subtract* `space1` *from* `space2`.

```
(7) var word2b = str.substring( space1+1, space2 ) ;
```

The second word — extracted using the `substring` function — is: **Hampshire**

> *The second parameter to the* `substring` *function is indeed the ending space. This is why we do* ***not*** *subtract* `space1` *from* `space2`.

```
(8) var phrase1 = word2 + " " + word1 ;
```

Concatenating `word1` onto `word2` yields: **Hampshire New**

> *Remember that the second parameter is the number of characters, not the ending space. This is why we have to subtract* `space1` *from* `space2`.

```
(9) var arrWords = str.split( " " ) ;
```

We can get all the words at once using the `split` function:

0. New
1. Hampshire
2. Dept.
3. of
4. Corrections
5. Special
6. School
7. District

> *Note that array indexes are 0-based!*

```
(10) document.writeln( arrWords[5] + " " + arrWords[7] + " " + arrWords[3] + " " +
    arrWords[1] + " " + arrWords[0] + " " + arrWords[6] + " " + arrWords[4] ) ;
```

We can then print them any way we want!

Special District of Hampshire New School Corrections

```
(11) var phrase2 = str.replaceAll( " ", "|" ) ;
```

Replacing all spaces with vertical bars yields:
New|Hampshire|Dept.|of|Corrections|Special|School|District

The first parameter is the search string and the second is the replacement string.

```
(12) var bStartsWithNew = str.startsWith( "New" ) ;
```

Testing whether the string starts with **New** yields: **true**

```
(13) var bEndsWithNew = str.startsWith( "Hampshire" ) ;
```

Testing whether the string ends with **Hampshire** yields: **false**

```
(14) var strUpper = str.toUpperCase() ;
```

Converting the string to all uppercase yields: **NEW HAMPSHIRE DEPT. OF CORRECTIONS SPECIAL SCHOOL DISTRICT**

```
(15) var strLower = str.toLowerCase() ;
```

Converting the string to all lowercase yields: **new hampshire dept. of corrections special school district**

```
(16) var bIncludesSchool1 = str.includes( "School" ) ;
```

Testing whether the string includes (contains) **School** yields: **true**

Note that the `includes` *function is case-sensitive. This is verified by the next example.*

```
(17) var bIncludesSchool2 = str.includes( "school" ) ;
```

Testing whether the string includes **school** yields: **false**