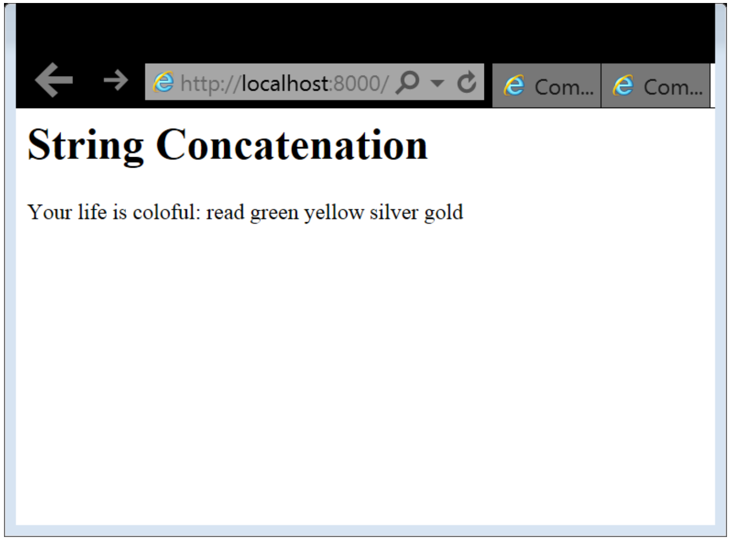
String Manipulation

String operators

- The concatenation operator (.) combines several string variables and literal strings.
- You can also combine strings using the concatenation assignment operator (.=)
- Example

```
<html>
   <head>
      <title>String Concatenation</title>
   </head>
   <body>
      <h1>String Concatenation</h1>
      <?php
         $colors = array("read", "green", "yellow", "silver", "gold");
         $message = "Your life is coloful: ";
         foreach ($colors as $color) {
            $message .= " " . $color;
         }
         echo("<p>$message</p>");
      ?>
   </body>
</html>
```



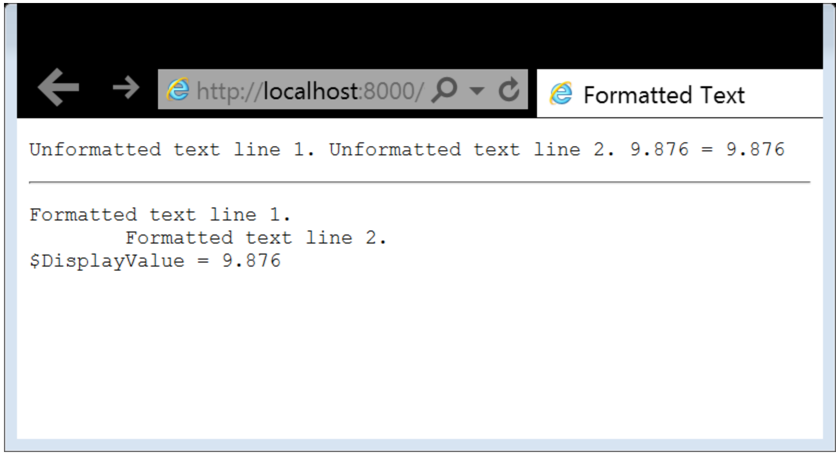
String Concatenation

Your life is coloful: read green yellow silver gold

The <pre> element

- Normally, the Web browser will treat all new lines, carriage returns, and tabs as spaces. Using the <pre> tag tells the Web browser not to convert those charac- ters to spaces.
- Example

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Formatted Text</title>
  </head>
  <body>
    <?php
    $DisplayValue=9.876;
    echo "<pre>\n";
    echo "Unformatted text line 1. ";
    echo "Unformatted text line 2. ";
    echo "$DisplayValue = $DisplayValue";
    echo "</pre>\n";
    echo "<hr>";
    echo "<pre>\n";
    echo "Formatted text line 1. \r\n";
    echo "\tFormatted text line 2. \r\n";
    echo "\$DisplayValue = $DisplayValue";
    echo "</pre>\n";
    ?>
  </body>
</html>
```



```
Unformatted text line 1. Unformatted text line 2. 9.876 = 9.876
```

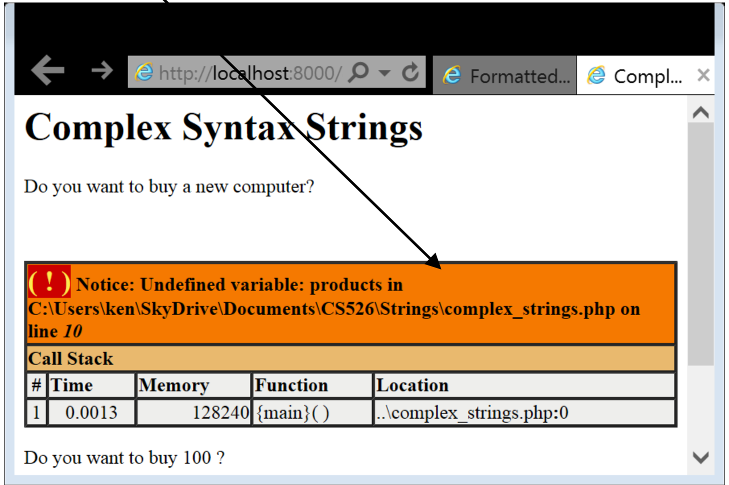
```
Formatted text line 1.
        Formatted text line 2.
$DisplayValue = 9.876
```

Complex String Syntax

- When the PHP scripting engine encounters a dollar sign within a text string, it attempts to evaluate any characters that follow the dollar sign as part of the variable name until it comes to a character that is not allowed in an identifier, such as a space.
- Example

```
<html>
   <head>
      <title>Complex Syntax Strings</title>
   </head>
   <body>
      <h1>Complex Syntax Strings</h1>
      <?php
      $product = "computer";
      echo "<p>Do you want to buy a new $product?</p><br>";
      echo "<p>Do you want to buy 100 $products?</p><br>"; // undefined $products
      echo "<p>Do you want to buy 100 {$product}s?</p><br>";
      ?>
   </body>
</html>
```

This is a correct way



Complex Syntax Strings

Do you want to buy a new computer?

(!) Notice: Undefined variable: products in C:\Users\ken\SkyDrive\Documents\CS526\Strings\complex_strings.php on line *10*

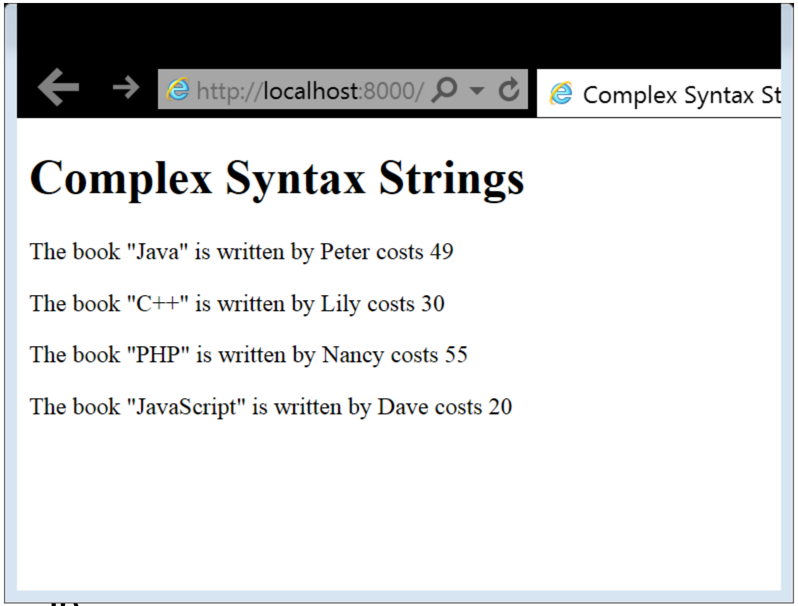
Call Stack

#	Time	Memory	Function	Location
1	0.0013	128240	{main}( )	..\complex_strings.php:0

Do you want to buy 100 ?

The complex_string_syntax.php

```
<!DOCTYPE html>
<html>
<head>
   <title>Complex Syntax Strings</title>
</head>
<body>
   <h1>Complex Syntax Strings</h1>
   <?php
   $books = array("Java", "C++", "PHP", "JavaScript");
   $authors = array("Peter", "Lily", "Nancy", "Dave");
   $prices = array(49, 30, 55, 20);
   for ($i = 0; $i < count($books); $i++) {
      echo "<p>The book \"{$books[$i]}\" is written by {$authors[$i]} costs
{$prices[$i]}</p>";
   }
   ?>
</body>
</html>
```



Complex Syntax Strings

The book "Java" is written by Peter costs 49

The book "C++" is written by Lily costs 30

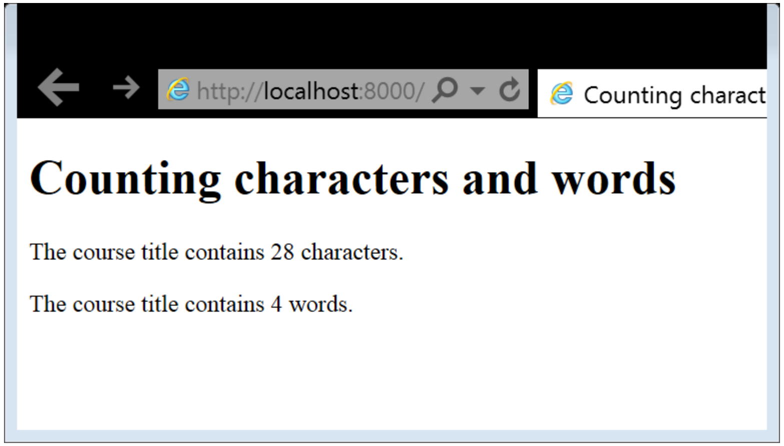
The book "PHP" is written by Nancy costs 55

The book "JavaScript" is written by Dave costs 20

String handling functions

- The **strlen()** function that returns the total number of characters in a string.
- The **str_word_count**() function, which returns the number of words in a string.
- Example

```
<!DOCTYPE html>
<html>
   <head>
      <title>Counting characters and words</title>
   </head>
   <body>
      <h1>Counting characters and words</h1>
      <?php
      $courseTitle = "Advanced PHP Web Programming";
      echo "<p>The course title contains " . strlen($courseTitle) . " characters.</p>";
      echo "<p>The course title contains " . str_word_count($courseTitle) . "
words.</p>";
      ?>
   </body>
</html>
```



Counting characters and words

The course title contains 28 characters.

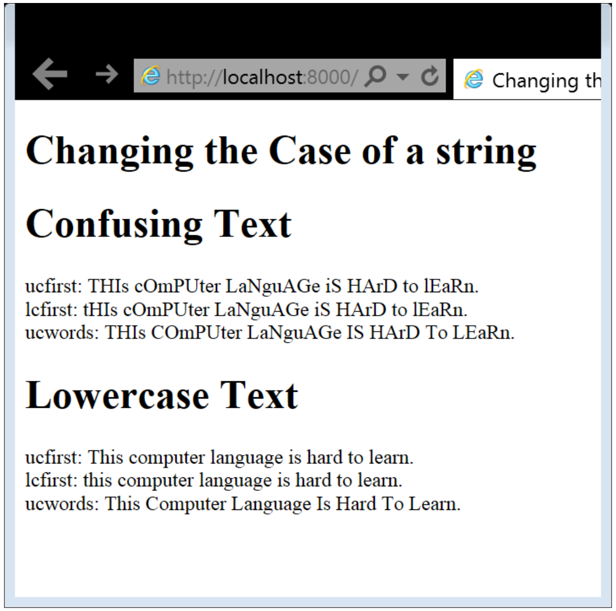
The course title contains 4 words.

Changing the case of a string

- The **strtoupper**() function converts all of the letters in a string to uppercase.
- The **strtolower**() function converts all of the letters in a string to lowercase.
- The **ucfirst()** function ensures that the first character of a string is uppercase.
- The **lcfirst**() function converts the first character of a string to lowercase
- The **ucwords**() function converts the first character of each word in a string to uppercase

The change_case.php

```
<!DOCTYPE html>
<html>
  <head>
    <title>Changing the Case of a string</title>
  </head>
  <body>
    <h1>Changing the Case of a string </h1>
    <?php
    $someText = "tHIs cOmPUter LaNguAGe iS HArD to lEaRn.";
    echo "<h1>Confusing Text</h1>\n";
    echo "ucfirst: " . ucfirst($someText) . "<br />\n";
    echo "lcfirst: " . lcfirst($someText) . "<br />\n";
    echo "ucwords: " . ucwords($someText) . "<br />\n";

    $LowercaseText = strtolower($someText);
    echo "<h1>Lowercase Text</h1>\n";
    echo "ucfirst: " . ucfirst($LowercaseText) . "<br />\n";
    echo "lcfirst: " . lcfirst($LowercaseText) . "<br />\n";
    echo "ucwords: " . ucwords($LowercaseText) . "<br />\n";
    ?>
  </body>
</html>
```



Changing the Case of a string

Confusing Text

```
ucfirst: THIs cOmPUter LaNguAGe iS HArD to lEaRn.
lcfirst: tHIs cOmPUter LaNguAGe iS HArD to lEaRn.
ucwords: THIs COmPUter LaNguAGe IS HArD To LEaRn.
```

Lowercase Text

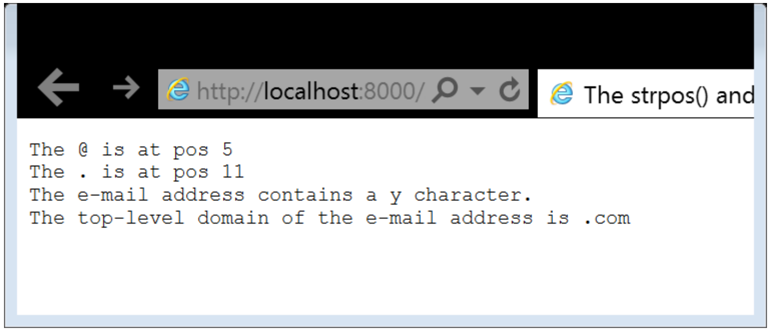
```
ucfirst: This computer language is hard to learn.
lcfirst: this computer language is hard to learn.
ucwords: This Computer Language Is Hard To Learn.
```

String parsing

- The term "parsing" refers to the act of dividing a string into logical component substrings or tokens. E.g. you may need to extract the name portion of the e-mail address or domain name.
- The **strpos**() function that performs a case-sensitive search and returns the position of the first occurrence of a substring within a string.
- The **strchr**() function that returns the last portion of a string, starting with a specified character. It starts searching at the beginning of a string.
- The **strrchr**() function that returns the last portion of a string, starting with a specified character. It starts searching at the end of a string.

Example of strpos()and strchr()

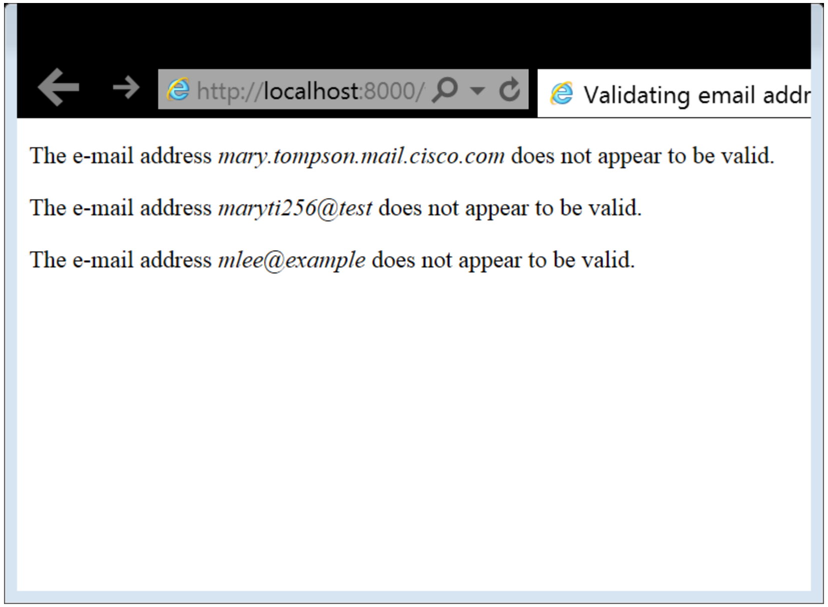
```
<!DOCTYPE html>
<html>
   <head>
      <title>The strpos() and strchr() function</title>
   </head>
   <body>
      <?php
      $email = "peter@yahoo.com";
      echo "<pre>";
      echo "The @ is at pos " . strpos($email, '@') . "\n";
      echo "The . is at pos " . strpos($email, '.') . "\n";

      if (strpos($email, '@') !== FALSE)
         echo "The e-mail address contains a y character.\n";
      else
         echo "<p>The e-mail address does not contain a y character.\n";

      echo "The top-level domain of the e-mail address is " . strrchr($email, ".")
. "\n";
      echo "</pre>";
      ?>
   </body>
</html>
```



```
The @ is at pos 5
The . is at pos 11
The e-mail address contains a y character.
The top-level domain of the e-mail address is .com
```

Example of validating email addresses

```
<!DOCTYPE html>
<html>
   <head>
      <title>Validating email addresses</title>
   </head>
   <body>
      <?php
      $emailAddresses = array(
         "peter.lee@yahoo.com",
         "mary.tompson.mail.cisco.com",
         "john.hart@mail.invalid",
         "alan.buck@test",
         "peterli256@example.com",
         "maryti256@test",
         "mlee@example",
         "mlee@example.net",
         "king.a.deer@example.org");

      function validateAddress($Address) {
         if (strpos($Address, '@') !== FALSE && strpos($Address, '.') !== FALSE)
            return TRUE;
         else
            return FALSE;
      }

      foreach ($emailAddresses as $address) {
         if (validateAddress($address) == FALSE)
            echo "<p>The e-mail address <em>$address</em> does not appear to be
valid.</p>\n";
      }
      ?>
   </body>
</html>
```



The e-mail address *mary.tompson.mail.cisco.com* does not appear to be valid.

The e-mail address *maryti256@test* does not appear to be valid.

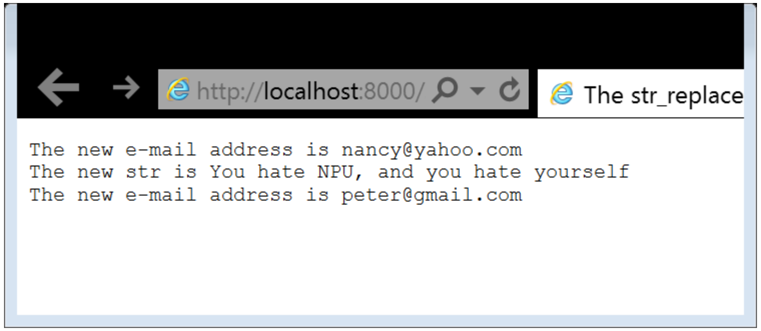
The e-mail address *mlee@example* does not appear to be valid.

Replacing characters and substrings

- **The str_replace**() and **str_ireplace**() functions both accept three arguments: the string you want to search for, a replacement string, and the string in which you want to replace characters.
- The **substr_replace**() function allows you to replace characters within a specified portion of a string. You pass to the substr_replace() function the string you want to replace, the replacement text, and the starting and ending positions of the characters you want to replace. If you do not include the last argument, the substr_replace() function replaces all the characters from the starting position to the end of the string.

Example of str_replace() and substr_replace() function

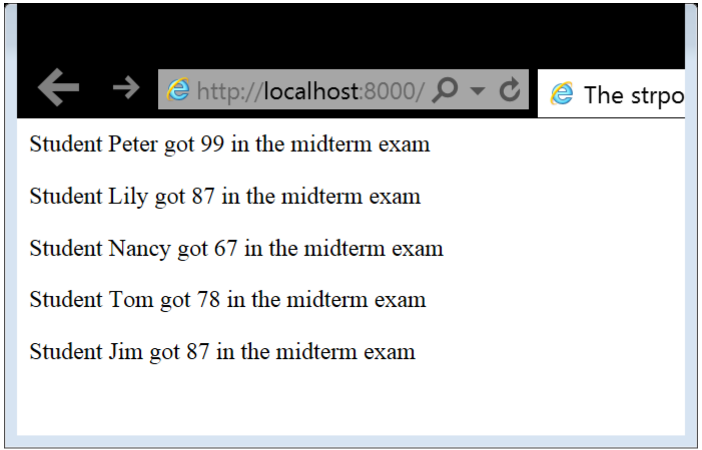
```
<!DOCTYPE html>
<html>
   <head>
      <title>The str_replace() and substr_replace() function</title>
   </head>
   <body>
      <?php
      $email = "peter@yahoo.com";
      $newEmail = str_replace("peter", "nancy", $email);
      echo "<pre>";
      echo "The new e-mail address is $newEmail\n";
      $str = "You love NPU, and you love yourself";
      $newStr = str_replace("love", "hate", $str);
      echo "The new str is $newStr\n";
      $start = strpos($email, "y");
      $end = strpos($email, ".");
      $length = $end - $start;
      $newEmail = substr_replace($email, "gmail", $start, $length);
      echo "The new e-mail address is $newEmail\n";
      echo "</pre>";
      ?>
   </body>
</html>
```



```
The new e-mail address is nancy@yahoo.com
The new str is You hate NPU, and you hate yourself
The new e-mail address is peter@gmail.com
```

Example of a message template

```
<html>
  <head>
    <title>The strpos() and strchr() function</title>
  </head>
  <body>
    <?php
    $students = array("Peter", "Lily", "Nancy", "Tom", "Jim");
    $scores = array(99, 87, 67, 78, 87);
    $messageTemplate = "<p>Student [NAME] got [SCORE] in the midterm exam</p>";
    foreach ($students as $studentNo => $name) {
      $tempMessage = str_replace("[NAME]", $name, $messageTemplate);
      $message = str_replace("[SCORE]", $scores[$studentNo], $tempMessage);
      echo $message;
    }
    ?>
  </body>
</html>
```



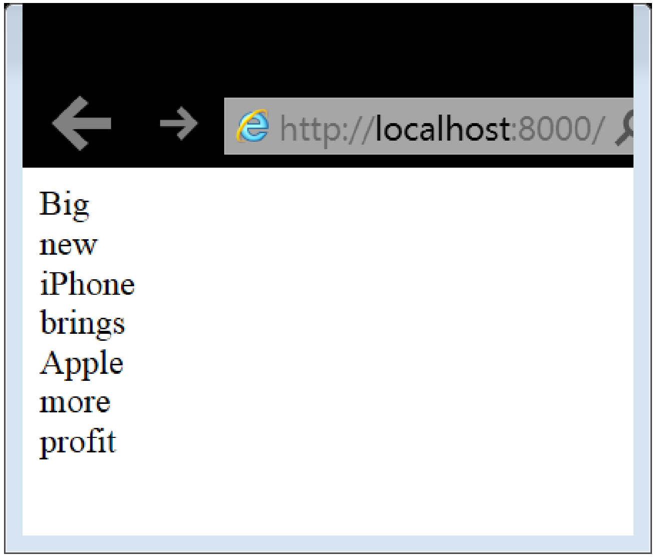
```
Student Peter got 99 in the midterm exam

Student Lily got 87 in the midterm exam

Student Nancy got 67 in the midterm exam

Student Tom got 78 in the midterm exam

Student Jim got 87 in the midterm exam
```

Dividing strings into parts

- If a string that contains multiple data elements (tokens) separated by a common delimiter, you can split the string into its individual elements using strtok() function.
- The syntax of the strtok() function is
  - $variable = strtok(string, separators);
  - The strtok() function assigns to $variable the token (substring) from the beginning of the string to the first separator.
  - To assign the next token to $variable, you call the strtok() function again, but only pass to it a single argument containing the separator.

Example of strtok() function

```
<!DOCTYPE html>
<html>
  <head>
    <title>The strtok() function</title>
  </head>
  <body>
    <?php
    $news = "Big new iPhone brings Apple more profit";
    $word = strtok($news, " ");
    while ($word != NULL) {
      echo "$word<br />";
      $word = strtok(" ");
    }
    ?>
  </body>
</html>
```



```
Big
new
iPhone
brings
Apple
more
profit
```

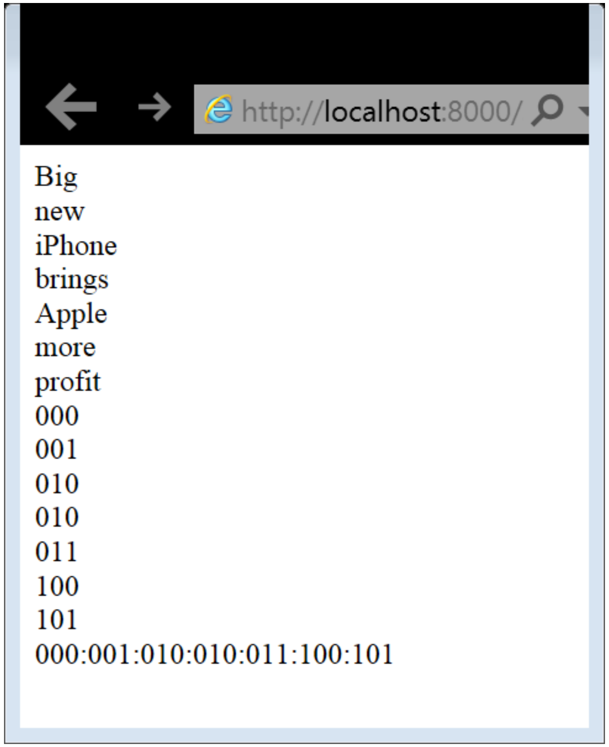
Converting between arrays and strings

- The **str_split**() or **explode**() function to split a string into an indexed array.
  - The **str_split()** function splits each character in a string into an array element, using the syntax
    - $array = str_split(string[, length]);.
    - The length argument represents the number of characters you want assigned to each array element.
  - The **explode()** function splits a string into an indexed array at a specified separator. The syntax for the explode() function is
    - $array = explode(separator, string);
    - Be sure to notice that the order of the arguments for the explode() function is the reverse of the arguments for the strtok() function.
- The opposite of the explode() function is the implode() function, which combines an array's elements into a single string, separated by specified characters. The syntax for the **implode()** function is

  $variable = implode(separator, array);

The explode(), str_split(), and implode() function

```
<!DOCTYPE html>
<html>
   <head>
      <title>The str_split and explode() function</title>
   </head>
   <body>
      <?php
      $news = "Big new iPhone brings Apple more profit";
      $wordsArray = explode(" ", $news);
      foreach ($wordsArray as $word) {
         echo "$word<br />";
      }
      $codes = "000001010010011100101";
      $codesArray = str_split($codes, 3);
      foreach ($codesArray as $code) {
         echo "$code<br />";
      }
      $codesStr = implode(":", $codesArray);
      echo "$codesStr<br />";
      ?>
   </body>
</html>
```



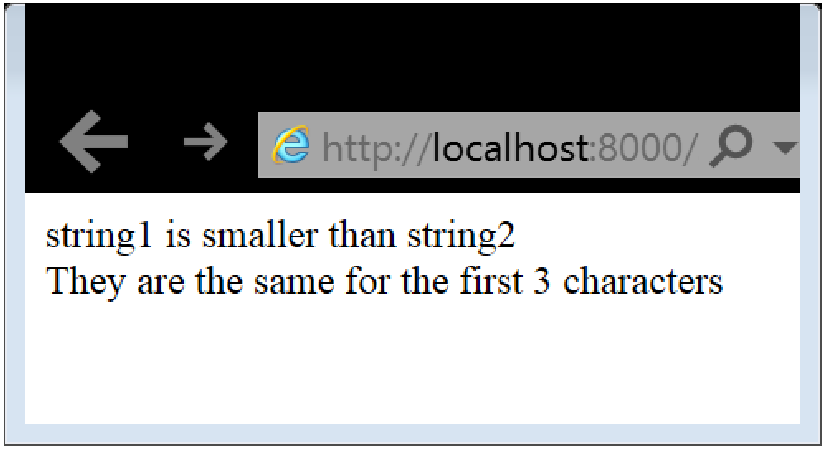
```
Big
new
iPhone
brings
Apple
more
profit
000
001
010
010
011
100
101
000:001:010:010:011:100:101
```

String comparison function

- The **strcasecmp**() and **strcmp**(). The only difference between the two is that the strcasecmp() function performs a case-insensitive comparison of strings, whereas the strcmp() function performs a case-sensitive comparison.
  - $result= strcmp(string1, string2);.
  - If the ASCII value in string1 is less than that of string2, the functions return a value less than 0, usually –1. However, if the ASCII value of the character in string2 is greater than the ASCII value of the corresponding character in string1, the functions return a value greater than 0, usually 1. If they are the same, it returns 0.
- The strncmp() and strncasecmp() functions are very similar to the strcmp() and strcasecmp() functions, except that you need to pass a third integer argument representing the number of characters you want to compare in the strings.

Example of strcmp() function

```
<!DOCTYPE html>
<html>
  <head>
    <title>The strcmp() function</title>
  </head>
  <body>
    <?php
    $string1 = "ABCDEF";
    $string2 = "ABCF";
    $result = strcmp($string1, $string2);
    if ($result > 0)
      echo "string1 is greater than string2<br />";
    else if ($result < 0)
      echo "string1 is smaller than string2<br />";
    else
      echo "They are the same<br />";
    if (strncmp($string1, $string2, 3) == 0)
      echo "They are the same for the first 3 characters<br />";
    ?>
  </body>
</html>
```



```
string1 is smaller than string2
They are the same for the first 3 characters
```