COMP721 Web Development

AUT UNIVERSITY

Week 3: PHP Part 2

Agenda

- *Week 2 review*
- *More about arrays: multi-dimensional and Associative*
- *Array functions, searching and sorting*
- *PHP Superglobals*
- *HTML input types*
- *Strings*
- *PHP as an Object-Oriented Language*

Key features of PHP

- Code Embedded in HTML as a code block:

  <?php statements; ?>

- Dynamic typing: no type declaration is needed for variables; type of a variable is dynamic/changeable
- To strictly compare two variables, use "==="
- Statement names (such as "if", "echo") are case-insensitive but we should stick to lowercase for consistency
- Variables start with "$", names are case-sensitive
- Syntax of expression and control structures very similar to Java and C/C++

PHP Multi-dimensional Arrays

Creating an indexed array

- The `array()` construct syntax is:

[$*array_name* = array(*values*);]

```
$provinces = array(
        "Newfoundland and Labrador",
        "Prince Edward Island",
        "Nova Scotia",
        "New Brunswick",
        "Quebec",
        "Ontario",
        "Manitoba",
        "Saskatchewan",
        "Alberta",
        "British Columbia"
        );
```

Multi-dimensional arrays

Name	Stock	Sold
Volvo	22	18
BMW	15	13
Saab	5	2
Land Rover	17	15

```
$cars = array
  (
  array("Volvo",22,18),
  array("BMW",15,13),
  array("Saab",5,2),
  array("Land Rover",17,15)
  );
```

What's the value of $cars[1][2]?

Visualization of multi-dim arrays (tensors in data science)

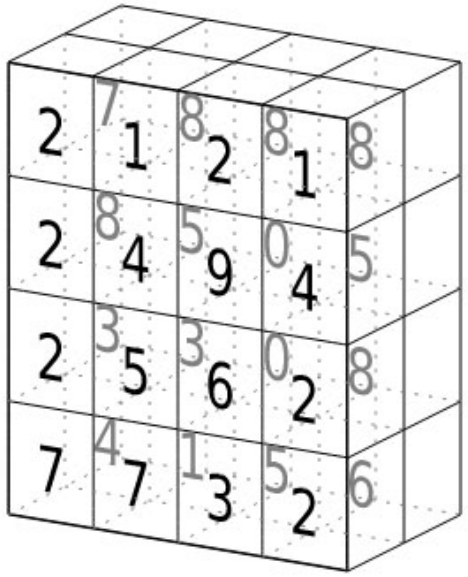
tensor

't'
'e'
'n'
's'
'o'
'r'

tensor of dimensions [6]
(vector of dimension 6)

3	1	4	1
5	9	2	6
5	3	5	8
9	7	9	3
2	3	8	4
6	2	6	4

tensor of dimensions [6,4]
(matrix 6 by 4)



tensor of dimensions [4,4,2]

Tensorflow?

PHP Associative Arrays

Creating an associative array

- The `array()` construct syntax for creating assoc. array is:

```
[$array_name = array(key1 => value1,
key2 => value2, …);]
```

$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

- Example code:

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
echo "Peter is " , $age["Peter"] , " years old.";
?>
```

http://sandbox.onlinephpfunctions.com/

Loop Through an Associative Array

- Example:

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
foreach($age as $x => $x_value){
        echo "Key=" , $x , ", Value=" , $x_value;
        echo "<br>";}
?>
```

Array functions, searching and sorting

Some useful array functions

- The **`array_shift()`** function removes the first element from the beginning of an array
- The **`array_unshift()`** function adds one or more elements to the beginning of an array
- The **`array_pop()`** function removes the last element from the **end** of an array
- The **`array_push()`** function adds one or more elements to the **end** of an array
- The **`array_splice()`** function adds or removes array elements

```
array_splice(array_name, starting element,
elements_to_delete, values_to_insert);
```

Some useful array functions (cont'd)

- The `in_array()` function returns a Boolean value of ***true*** if a given value exists in an array
- The `array_search($value, $array)` function determines whether a given value exists in an array and
  - Returns the ***index*** or ***key*** of the first matching element if the value exists, or
  - Returns ***false*** if the value does not exist
- The `array_key_exists()` function determines whether a given index or key exists
- The `array_slice()` function returns a portion of an array and assigns it to another array

Binary search review

Task 2: Creating a simple "Guessing Game" (2 marks)

The overall task is to create a simple web application that generates and uses **sessions** to store a random number between 0 and 100.

Step 1:

Create a file **`guessinggame.php`** that will be the main page for the game. In this page, a user inputs their guess; the page displays the number of times the user has guessed; whether their number is higher or lower than the generated number; and congratulates them when they guess correctly. (Checking always if the input data is "in-range" and is numeric). It also include a `'Give Up'` link to `giveup.php`, and a `'Start Over'` link to `startover.php`.

Question: what's the minimum number of guesses one needs to make in order to be successful for any random number between 0-100?

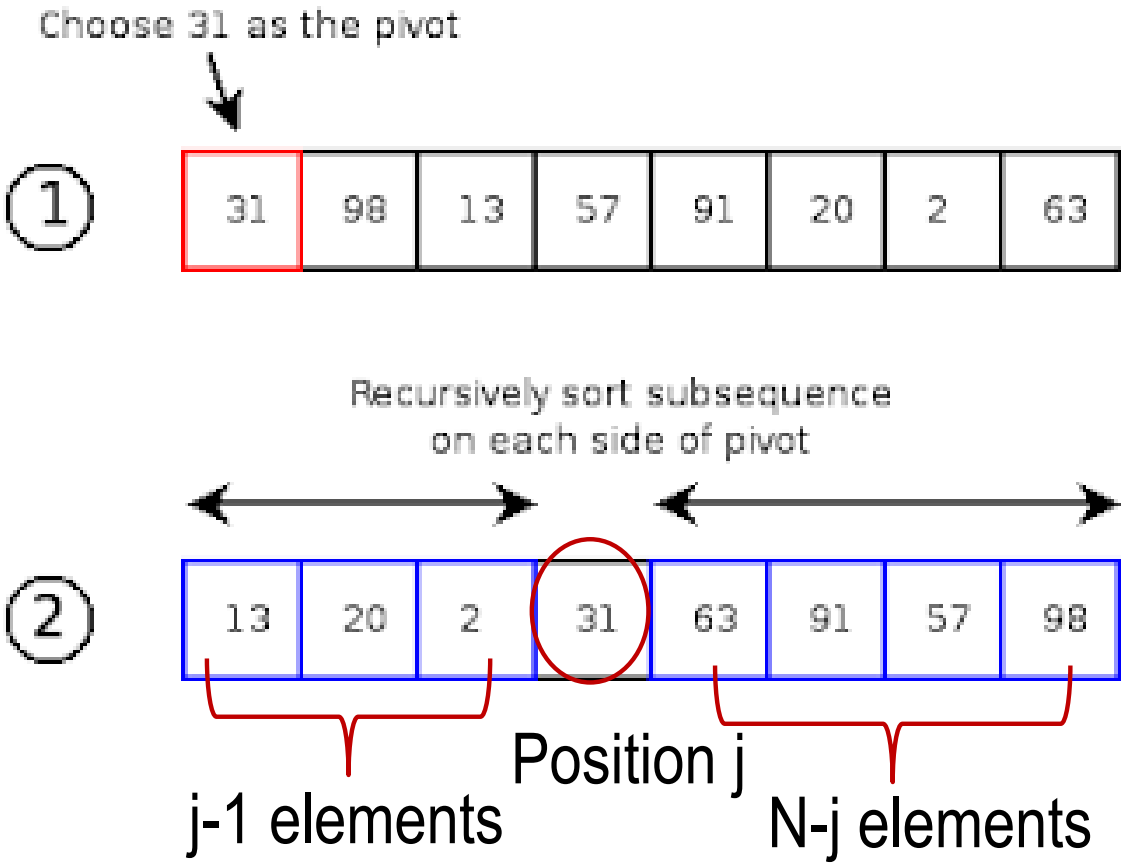
Sorting Arrays

- **`sort()`** and **`rsort()`** for *indexed arrays*
  - `sort()` sorts an indexed array by value and renumbers the indexes
  - `rsort()` performs a reverse sort
- **`ksort()`** and **`krsort()`** for *associative arrays by key*

http://www.php.net/manual/en/array.sorting.php

Special topic: complexity of quick-sort

- A recursive algorithm for sorting numbers
- One iteration needs N+1 compares (N is the input array size)



Time complexity of quick-sort

- If we use C_N to represent the total compares of sorting N elements, then:

$$C_N = N + 1 + C_{j-1} + C_{N-j}, for\ N > 1$$

- But j is a variable, it could be in position 1 to N, so we need to use the average of $C_{j-1} + C_{N-j}$:

$$C_N = N + 1 + \frac{1}{N} \sum_{j=1..N} \left(C_{j-1} + C_{N-j}\right), for\ N > 1$$

Time complexity of quick sort

- When j=N-k, $C_k = C_{N-k}$, so it can be simplified:
- $C_N = N + 1 + \frac{1}{N}\sum_{j=1..N}(C_{j-1} + C_{N-j})\,, for\ N > 1$

$$= N + 1 + \frac{2}{N}\sum_{j=1..N} C_{j-1}$$

- $C_{N-1} = N + \frac{2}{N-1}\sum_{j=1..N-1} C_{j-1}$
- *To remove the denominator:*

$$NC_N - (N-1)C_{N-1} = 2N + 2C_{N-1}$$

- Dividing both sides by $N(N+1)$ to obtain

$$\frac{C_N}{N+1} = \frac{C_{N-1}}{N} + \frac{2}{N+1}$$

Time complexity of quick-sort

- $\frac{C_N}{N+1} = \frac{C_{N-1}}{N} + \frac{2}{N+1}$: now we have the recurrence!
- Expanding/telescoping $\frac{C_{N-1}}{N}$, we obtain:

$$\frac{C_N}{N+1} = \frac{C_1}{2} + 2 \sum_{k=3..N+1} 1/k$$

Where C_1 is 0

- Approximating $\Sigma \frac{1}{\mathrm{k}}$, the harmonic series, to $\ln N$
- We obtain $C_N = O(NlnN)$

Combining Indexed Arrays

- To merge two or more indexed arrays use the **`array_merge()`** function

For example

```
$provinces = array("Newfoundland and Labrador",
   "Prince Edward Island", "Nova Scotia", "New
   Brunswick", "Quebec", "Ontario", "Manitoba",
   "Saskatchewan", "Alberta", "British Columbia");
$territories = array("Nunavut", "Northwest Territories",
   "Yukon Territory");
$canada = array_merge ($provinces, $territories);
print_r($canada);
        //territories appended
```

Output:

```
Array ( [0] => Newfoundland and Labrador [1] => Prince
Edward Island [2] => Nova Scotia [3] => New Brunswick [4]
=> Quebec [5] => Ontario [6] => Manitoba [7] =>
Saskatchewan [8] => Alberta [9] => British Columbia [10]
=> Nunavut [11] => Northwest Territories [12] => Yukon
Territory )
```

Combining Associative Arrays

- + and += works best on associative arrays, especially if the arrays involved do not have any common keys.

Note: only array elements with unique keys are appended. duplicated indexes/keys are ignored

```
$arr1 = array ("one"=>"apple", "two"=>"banana");
$arr2 = array ("three"=>"cherry", "two"=>"grapes");

$arr3 = $arr1 + $arr2;
print_r($arr3);
```

Output: `Array ( [one] => apple [two] => banana [three] => cherry )`

Duplicate keys overwritten

```
$arr4 = array_merge ($arr1, $arr2);
print_r($arr3);
```

Output: `Array ( [one] => apple [two] => grapes [three] => cherry )`

Comparing Arrays

- The **`array_diff()`** function returns an array of elements that exist in one array but not in any other arrays to which it is compared
- The syntax for the `array_diff()` function is:

```
new_array = array_diff($array1, $array2,
$array3, ...);
```

- The **`array_intersect()`** function returns an array of elements that exist in all of the arrays that are compared
- The syntax for the `array_intersect()` function is:

```
new_array = array_intersect($array1,
$array2, $array3, ...);
```

PHP Superglobal arrays

PHP Superglobals

- PHP includes various predefined global arrays, called **superglobals** or autoglobals
- Superglobals contain client, server, and environment information that you can use in your scripts
- Superglobals are **associative arrays** – arrays whose elements are referred to with an alphanumeric key instead of an index number

Key-Value pairs

See ***Predefined Variables, Superglobals and examples:***
http://php.net/manual/en/reserved.variables.php

Using superglobals (continued)

PHP autoglobals

Array	Description
$_COOKIE	An array of values passed to the current script as HTTP cookies
$_ENV	An array of environment information
$_FILES	An array of information about uploaded files
$_GET	An array of values from a form submitted with the GET method
$_POST	An array of values from a form submitted with the POST method
$_REQUEST	An array of all the elements found in the $_COOKIE, $_GET, and $_POST arrays
$_SERVER	An array of information about the Web server that served the current script
$_SESSION	An array of session variables that are available to the current script
$GLOBALS	An array of references to all variables that are defined with global scope

Demo: using $_POST

- Client side html:

```
<form action="process.php" method="post">
   <label>User Name:  <input type="text" name="name"> </label>
   <label><br>Password: <input type="text" name="pwd"></label>
   <input type="submit" value="Submit">
</form>
```

http://jiyu.cmslamp14.aut.ac.nz/

- Server side php:

```
<?php
        // get name and password passed from client
        $name = $_POST["name"];
        $pwd = $_POST["pwd"];
        echo $name." : ".$pwd;
?>
```

Demo using $_GET

- Two ways to send data through HTTP GET
  - Using query string:

    processget.php?name=jianyu&pwd=123456

    Query string: ?name=jianyu&pwd=123456
  - HTML forms using “get” method

```
<form action="process.php" method="get">
   <label>User Name:  <input type="text" name="name"> </label>
   <label><br>Password: <input type="text" name="pwd"></label>
   <input type="submit" value="Submit">
</form>
```

Discussion

Compare HTTP GET & POST methods
(details are in lecture 1 slides…)

	GET	POST
Restrictions on data length		
Restrictions on data type		
Security		
Data visibility		

HTML input types

- text, submit (submit button), reset (reset button), radio, checkbox, button
- HTML5 new types
  - color (mind the spelling!)
  - date (with min/max attributes)
  - time, week, month
  - email, url


```
 Enter a date after 2000-
01-01:
  <input type="date" name="
bday" min="2000-01-02"><br>
```

Form attributes

- pattern: using regular expressions; will be discussed in Week 4
  - Example

```
Country code: <input type="text"
name="country_code" pattern="[A-Za-z]{3}"
title="Three letter country code">
```

- autocomplete, autofocus, min, max, required…

HTML5 datalist Element

- datalist: similar to text input, but has a drop-down list of the pre-defined options as the input data (e.g., google search input box)

```
<form action="/action_page.php">
  <input list="browsers" name="browser">
  <datalist id="browsers">
    <option value="Internet Explorer">
    <option value="Firefox">
    <option value="Chrome">
    <option value="Opera">
    <option value="Safari">
  </datalist>
  <input type="submit">
</form>
```

PHP Strings

PHP is a "hypertext processor"

Form data validation

Use the `empty()` functions to ensure that a variable contains a value

- The `empty()` function determines whether a variable is empty
- Parameter of both functions is the name of the variable you want to check

- `isset()` function is another option

Testing if Form Variables Contain Numeric Values

Use the `is_numeric()` function to test whether a variable contains a numeric string

```
<?php
   // get name and password passed from client
   if (!empty($_GET['height']) && !empty($_GET['weight'])) {
            if (is_numeric($_GET['weight']) &&  is_numeric($_GET['height'])) {
                     $BodyMass = $_GET['weight'] / ($_GET['height']
                              * $_GET['height']) * 703;
                     printf("<p>Your body mass index is %d.</p>",
                              $BodyMass);
            }else
                     echo "<p>You must enter numeric values!</p>";
   }else
            echo "<p>please input both height and weight values!</p>";
?>
```

MANIPULATE STRINGS

Constructing Text Strings

- A text string contains zero or more characters surrounded by double or single quotation marks
- Text strings can be used as literal values or assigned to a variable

```
echo "<p>Dr. Livingstone, I presume?</p>";
$explorer = "Henry M. Stanley";
echo $explorer;
```

- Text strings can also be surrounded with single quotation marks

Note: no data type for a single character in PHP

Constructing Text Strings (continued)

- To include a quoted string within a literal string surrounded by double quotation marks, you surround the quoted string with single quotation marks

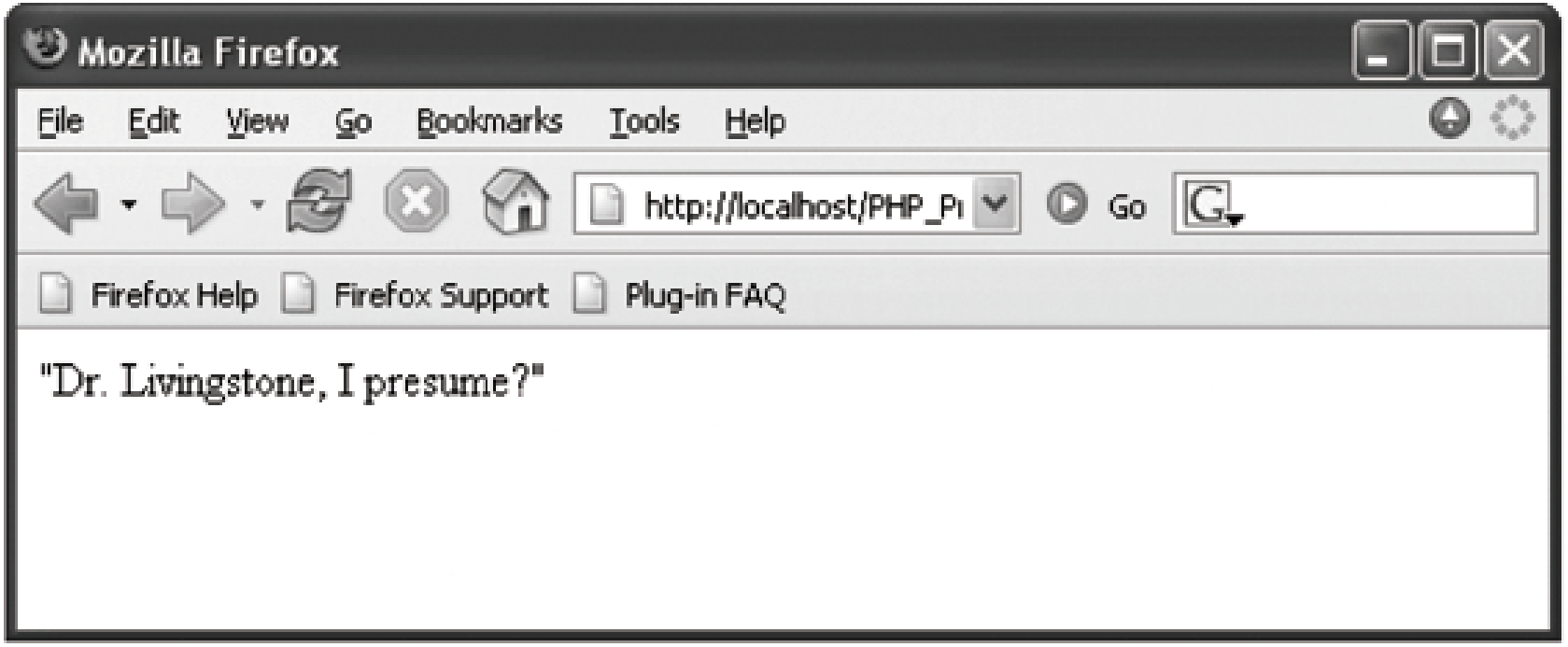
```
$explorerQuote = '<p>"Dr. Livingstone, I presume?"</p>';
```

- To include a quoted string within a literal string surrounded by single quotation marks, you surround the quoted string with double quotation marks

```
$explorerQuote = "<p>'Dr. Livingstone, I presume?'</p>";
```

Constructing Text Strings (continued)

```
$explorerQuote = '<p>"Dr. Livingstone, I presume?"</p>';
echo $explorerQuote;
```



Output of a text string containing double quotation marks

Working with String Operators

In PHP, you use two operators to combine strings

- **Concatenation operator .**

```
$destination = "Paris";
$location = "France";
$destination = "<p>" . $destination . " is in "
        . $location . ".</p>";
echo $destination;
```

- **Concatenation assignment operator .=**

```
$destination = "<p>Paris";
$destination .= "is in France.</p>";
echo $destination;
```

Adding Escape Characters and Sequences

- An escape character tells the compiler or interpreter that the character that follows it has a special purpose
- In PHP, the escape character is the backslash \

```
echo '<p>Marilyn Monroe\'s real name was Norma Jean
  Baker.</p>';
```

- Do not add a backslash before an apostrophe if you surround the text string with double quotation marks

```
echo "<p>Marilyn Monroe's real name was Norma Jean
  Baker.</p>";
```

Adding Escape Characters and Sequences

(continued)

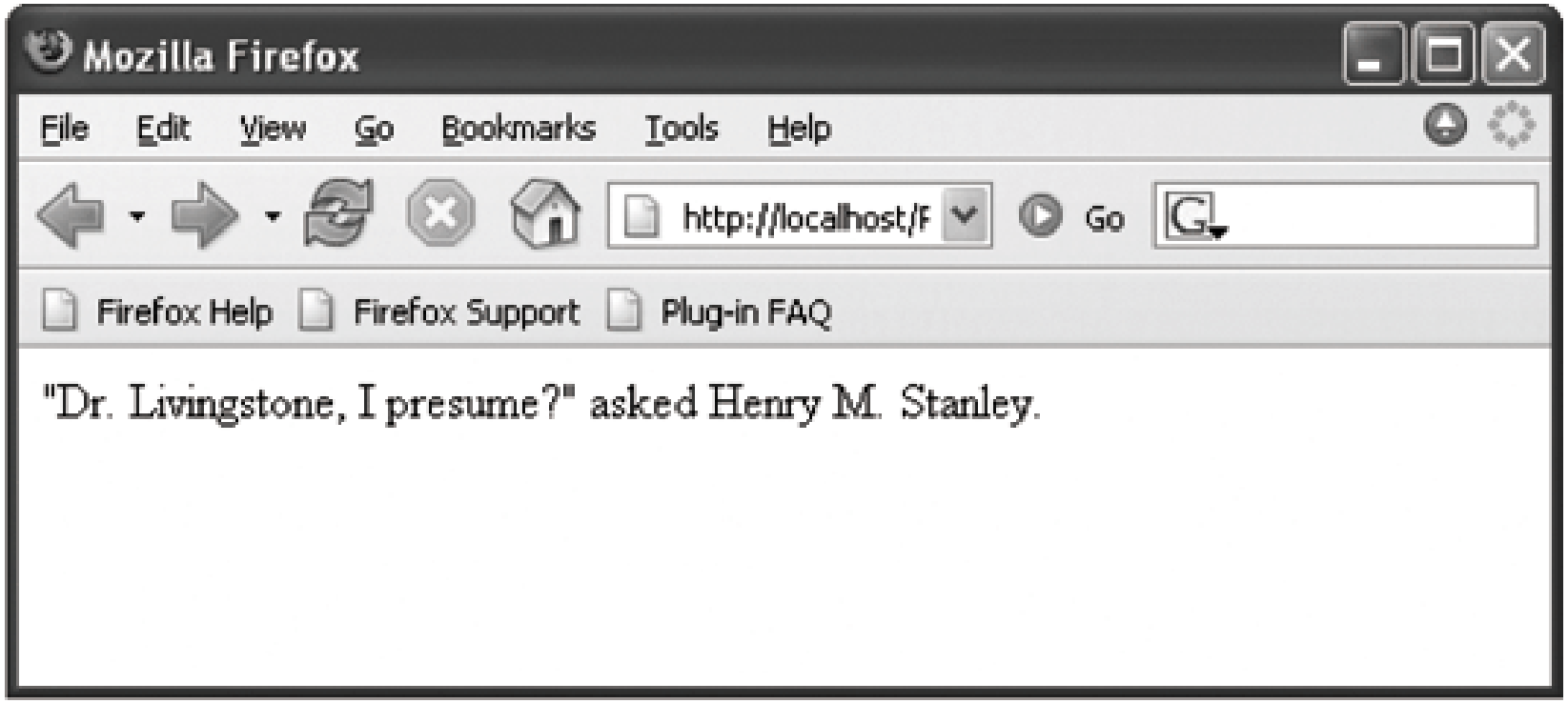
- The escape character combined with one or more other characters is called an escape sequence

Table of PHP escape sequences within double quotation marks

Escape Sequence	Description
`\\`	Inserts a backslash
`\$`	Inserts a dollar sign
`\r`	Inserts a carriage return
`\"`	Inserts a double quotation mark
`\t`	Inserts a horizontal tab
`\n`	Inserts a new line
`\regular expression`	Inserts a character in hexadecimal notation that matches the regular expression

Adding Escape Characters and Sequences (continued)

```
$explorer = "Henry M. Stanley";

echo "<p>\"Dr. Livingstone, I presume?\" asked
   $explorer.</p>";
```



Output of literal text containing double quotation escape sequences

Simple and Complex String Syntax

- **Simple string syntax** uses the value of a variable within a string by including the variable name inside a text string with double quotation marks

```
$vegetable = "broccoli";
echo "<p>Do you have any $vegetable?</p>";
```

How about:

```
echo "<p>Do you have any $vegetables?</p>";
  //causes an error, variable not declared.
```

- When variables are placed within curly braces inside of a string, it is called **complex string syntax**

```
$vegetable = "carrot";
echo "<p>Do you have any {$vegetable}s?</p>";
```

How about:

```
echo "<p>Do you have any {$vegetable}s?</p>";
  //output is: Do you have any carrots?
```

COMPARE STRINGS

Comparing Strings

Using Comparison Operator

```
$loc01 = "Miami is in Florida.";
$loc02 = "Havana is in Cuba.";
if ($loc01 == $loc02)
     echo "<p>Same location.</p>";
else
     echo "<p>Different location.</p>";
```

Comparing Strings (continued)

```
$firstLetter = "A";

$secondLetter = "B";

If ($secondLetter > $firstLetter)

     echo "<p>The second letter is higher in the
       alphabet than the first letter.</p>";

else

     echo "<p>The second letter is lower in the
       alphabet than The first letter.</p>";
```

ASCII American Standard Code for Information Interchange

- Numeric representations of English characters
- ASCII values range from 0 to 255
- Lowercase letters are represented by the values 97 (“a”) to 122 (“z”)
- Uppercase letters are represented by the values 65 (“A”) to 90 (“Z”)
- Since lowercase letters have higher values than uppercase letters, they are evaluated as being “greater” than the uppercase letters

Note: UTF-8 is a strict superset of ASCII with the same physical encoding for ASCII characters

String Comparison Functions

- The `strcasecmp()` function performs a case-insensitive comparison of strings
- The `strcmp()` function performs a case-sensitive comparison of strings

- Both functions accept two parameters representing the strings you want to compare
- Most string comparison functions compare strings based on their ASCII values – returns <0 (if smaller); >0 (if larger); =0 (if same)

Using Similarity Functions to Compare

- The `similar_text()` and `levenshtein()` functions are used to determine the similarity between two strings

- The `similar_text()` function returns the number of characters that two strings have in common
- The `levenshtein()` function returns the number of characters you need to change for two strings to be the same

Using Similarity Functions to Compare

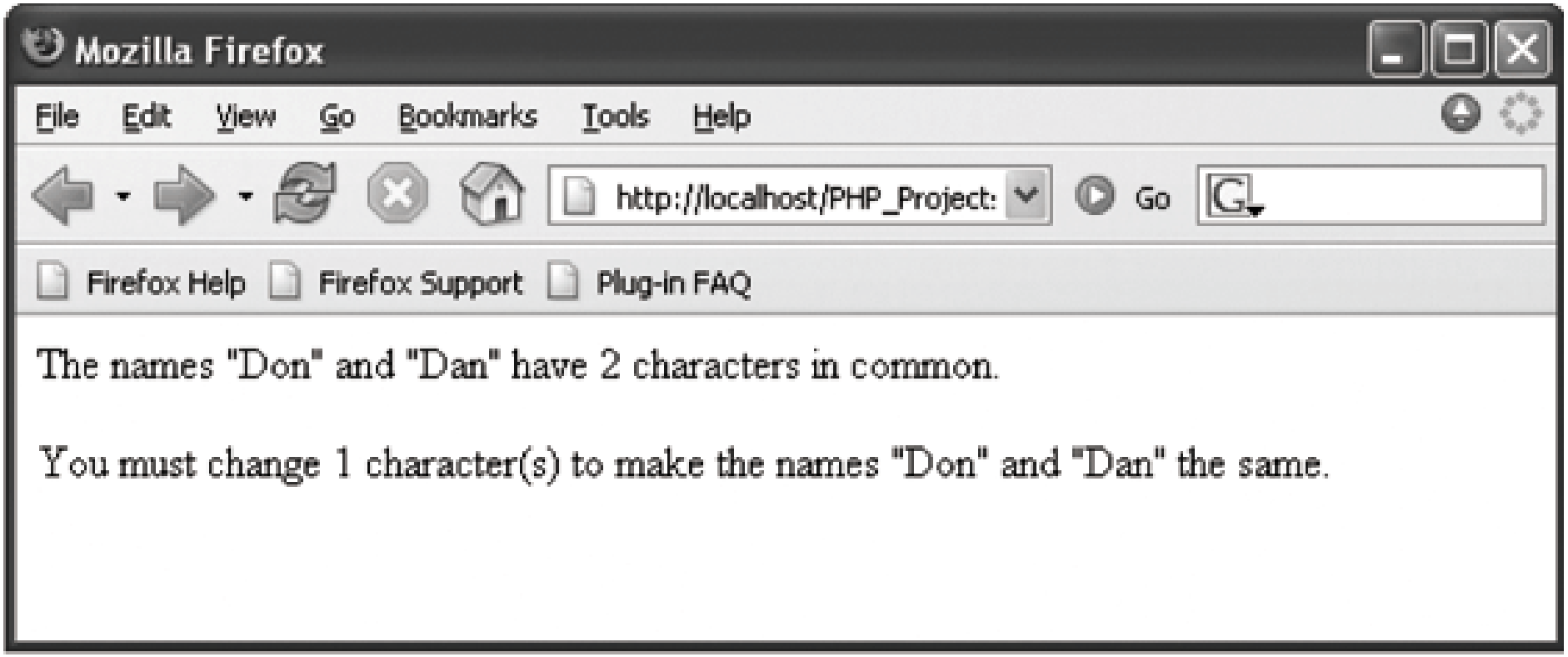
(continued)

- Both functions accept two string arguments representing the values you want to compare

```
$firstName = "Don";

$secondName = "Dan";

echo "<p>The names \"$firstName\" and
   \"$secondName\" have "
   . similar_text($firstName, $secondName)
   . " characters in common.</p>";

echo "<p>You must change "
  . levenshtein($firstName, $secondName)
  . " character(s) to make the names \"$firstName\"
   and \"$secondName\" the same.</p>";
```

Using Similarity Functions to Compare

(continued)



Output of a script with the `similar_text()` and `levenshtein()` functions

PARSE STRINGS

Parsing Strings

- **Parsing** is the act of extracting characters or substrings from a larger string
- When programming, parsing refers to the extraction of information from string literals and variables

Counting Characters and Words in a String

- The most commonly used string counting function is the `strlen()` function, which returns the total number of characters in a string

```
$myStr = ' ab cd ';
echo strlen($myStr); // 7
```

- The `str_word_count()` function returns the number of words inside a string
- Parameter of the `str_word_count()` function can be a literal string or the name of a string variable

```
$bookTitle = "The Cask of Amontillado";

echo "<p>The book title contains " .
   str_word_count($bookTitle) . " words.</p>";
```

Finding and Extracting Characters and Substrings

- There are two types of string search and extraction functions:
- Functions that return a numeric position in a text string
- Functions that return a character or substring

`strpos()` Function

- Performs a case-sensitive search and returns the position of the first occurrence of one string in another string
  Note: begins with a value of 0 // at the first character
- Two parameters for the `strpos()` function:
  - The first is the string you want to search
  - The second contains the characters for which you want to search
- If the search string is not found, the `strpos()` function returns a Boolean value of `false`

`strpos()` Function (Continued)

```
$email = "president@whitehouse.gov";
  echo strpos($email, '@'); //returns 9
  echo strpos($email, 'p'); //returns 0
```

`strchr()` and `strrchr()` Functions

- Parameters of both functions are the string and the character for which you want to search
- Both functions return a substring from the specified characters to the end of the string, *i.e. last portion*
- `strchr()` function starts searching at the beginning of a string
- `strrchr()` function starts searching at the end of a string

Note: Extra ‘r’ means reverse

`substr()` Function

- To extract characters from the beginning or middle of a string, combine the `substr()` function with other functions
- Parameters of the `substr()` function: a text string, the starting position and length of the substring you want to extract

```
$email = "president@whitehouse.gov";
$nameEnd = strpos($email, "@");
echo "<p>The name portion of the e-mail address
  is '" . substr($email, 0, $nameEnd) . "'.</p>";
```

Replacing Characters and Substrings

PHP string replacement functions

Function	Description
`str_ireplace(search_string, replacement_string, string)`	Performs a case-insensitive replacement of all occurrences of specified characters in a string
`str_replace(search_string, replacement_string, string)`	Performs a case-sensitive replacement of all occurrences of specified characters in a string
`substr_replace(string, replacement_string, start_position[, length])`	Replaces characters within a specified portion of a string

Note: Extra 'i' means case-insensitive

str_replace() and str_ireplace() Functions

- The `str_replace()` and `str_ireplace()` functions both accept three parameters:
  - The string you want to search for
  - A replacement string
  - The string in which you want to replace characters

```
$email = "president@whitehouse.gov";
$newEmail = str_replace("president", "vice.president",
   $Email);
echo $newEmail; // prints 'vice.president@whitehouse.gov'
```

Dividing Strings into Smaller Pieces

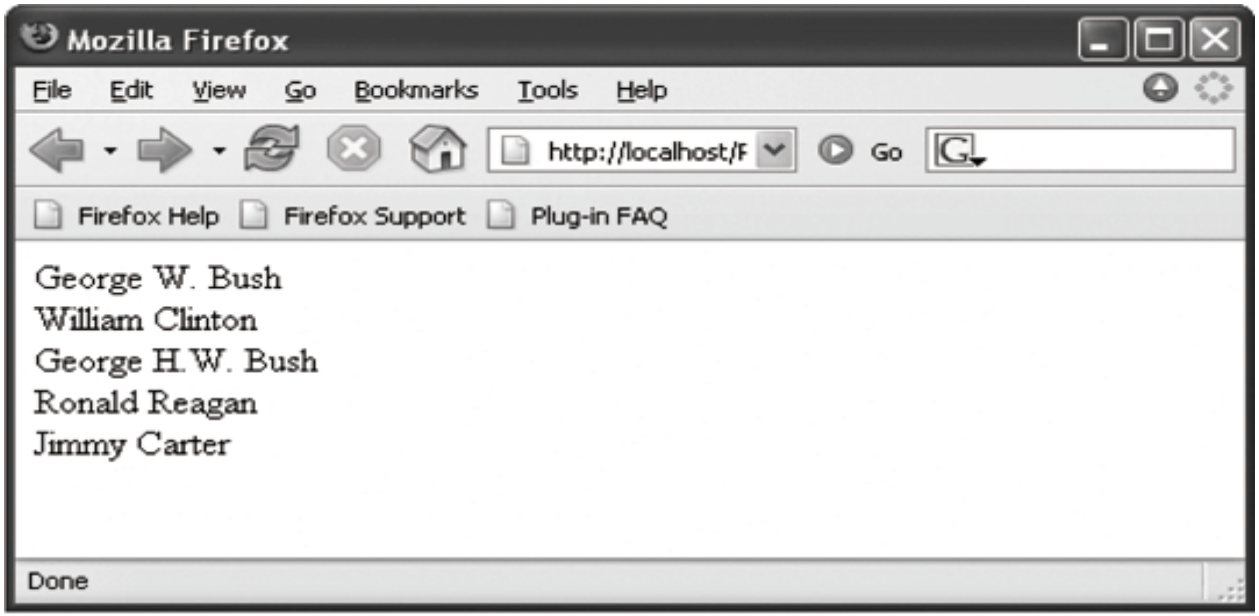
- Use the `strtok()` function to break a string into smaller strings, called **tokens** (one by one)
- The syntax for the **`strtok()`** function is:

```
$variable = strtok(string, separators);
```

- The `strtok()` function returns the entire string if:
  - An empty string is specified as the second argument of the `strtok()` function
  - The string does not contain any of the separators specified
- The `strtok()` function returns tokens one by one

strtok() Function

```
$presidents = "George W. Bush;William Clinton;
   George H.W. Bush;Ronald Reagan;Jimmy Carter";
$president = strtok($presidents, ";");
while ($president != NULL) {
      echo "$president<br/>";
      $president = strtok(";"); //only the separator ";"
   here. The PHP scripting engine keeps track of the
   current token and next token.
}
```



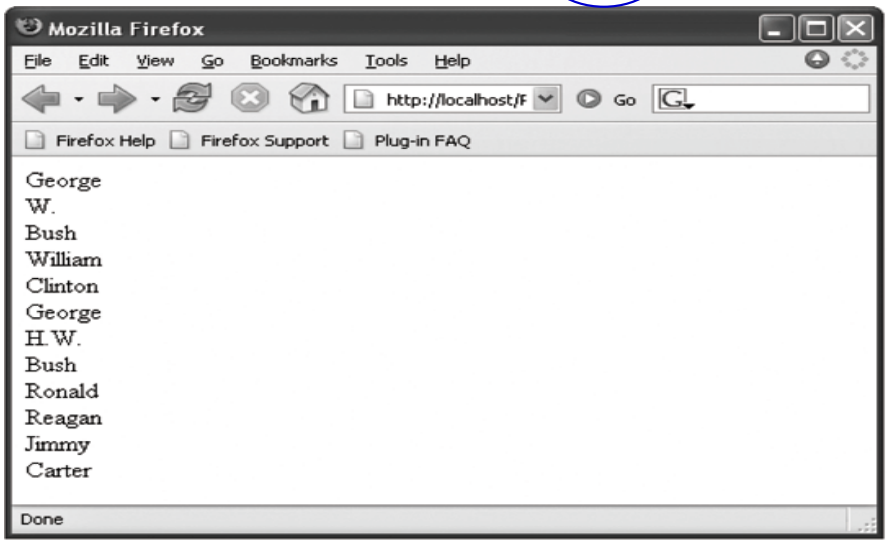
Output of a script that uses `strtok()`

`strtok()` Function (continued)

- `strtok()` divides a string into tokens using any of the characters that are passed

```
$presidents = "George W. Bush;William Clinton;
George H.W. Bush;Ronald Reagan;Jimmy Carter";
$president = strtok($presidents, "; ");
while ($president != NULL) {
   echo "$president<br />";
   $president = strtok("; ");
}
```

Two separators used: ";" and [space]



Output of a script with a `strtok()` function that uses two separators

Converting Between Strings and Arrays

Can also split a string into an array

- The `str_split()` and `explode()` functions 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

Converting Between Strings and Arrays

(continued)

- The `explode()` function splits a string into an indexed array at a specified separator
- The syntax for the `explode()` function is:

```
$array = explode(separators, string);
```

- Note: The order of the arguments for the `explode()` function is the *reverse* of the arguments for the `strtok()` function
- If the string does not contain the specified separators, the entire string is assigned to the first element of the array

Converting Between Strings and Arrays

(continued)

```
$presidents = "George W. Bush;William Clinton;
   George H.W. Bush;Ronald Reagan;Jimmy Carter";
$presidentArray = explode(";", $presidents);

foreach ($presidentArray as $president) {
      echo "$president<br />";
}
```

- Does not separate a string at each character that is included in the separator argument
- Evaluates the characters in the separator argument as a substring
- If you pass to the `explode()` function an empty string as the separator argument, the function returns a value of `false`

`implode()` Function

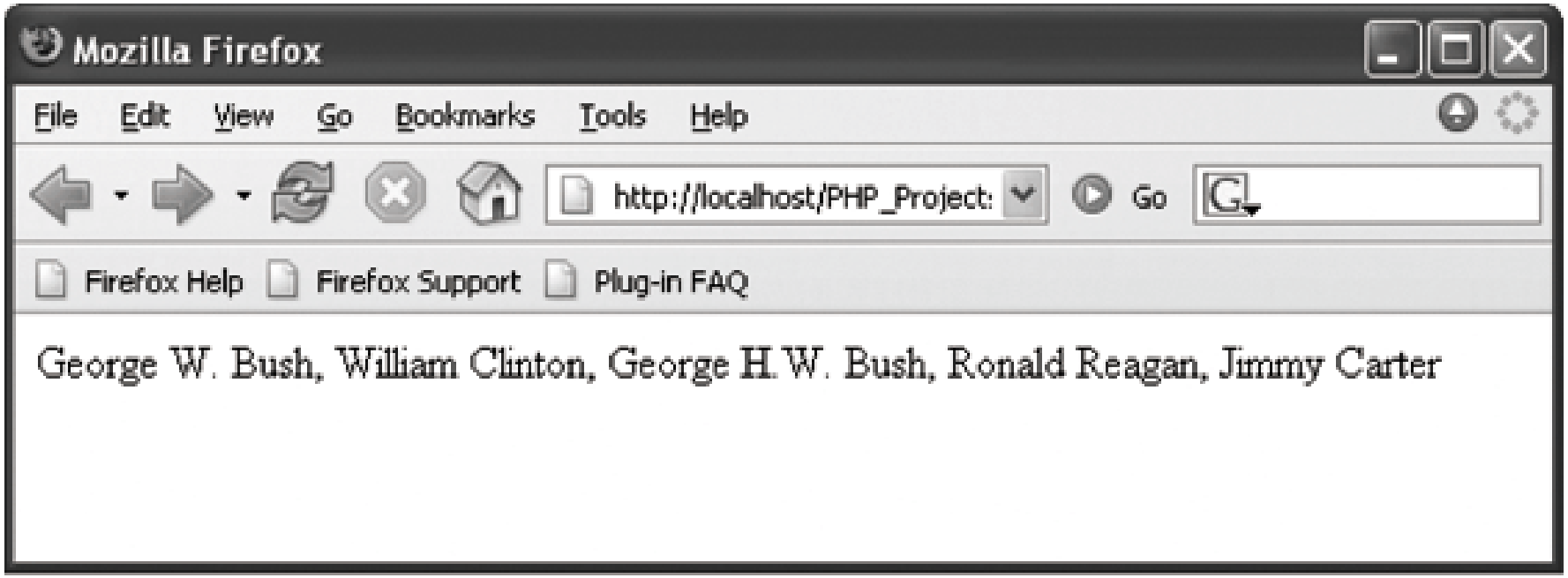
Opposite to `explode()`

- Combines an array's elements into a single string, separated by specified characters
- The syntax is:

```
$variable = implode(separators, array);
```

`implode()` Function (continued)

```
$presidentsArray = array("George W. Bush",
   "William Clinton", "George H.W. Bush", "Ronald
   Reagan", "Jimmy Carter");

$presidents = implode(", ", $presidentsArray);

echo $presidents;
```



Output of a string created with the `implode()` function

Other String Functions

- There are many useful string functions – see http://php.net/manual/en/ref.strings.php for a full list.
- Just a few:
  - `trim()` Strip whitespace (or other characters) from the beginning and end of a string
  - `htmlspecialchars()` Some characters have a special meaning in HTML and have to be converted to HTML Entities if they appear in a HTML document.

'&' (ampersand) becomes '&'
'"' (double quote) becomes '"'
''' (single quote) becomes '''
'<' (less than) becomes '<'
'>' (greater than) becomes '>'

PHP as an Object-Oriented Language

AUT UNIVERSITY

```
class BankAccount {
     private $balance = 0;
     public function setBalance($newValue) {
          $this->balance = $newValue;
     }
     public function getBalance() {
          return $this->balance;
     }
}
if (class_exists("BankAccount")) {
     echo "<p>The BankAccount class is not available!</p>";
} else {
     $checking = new BankAccount();
     $checking->setBalance(100);
     echo "<p>Your checking account balance is "
          . $checking->getBalance() . "</p>";
}
```