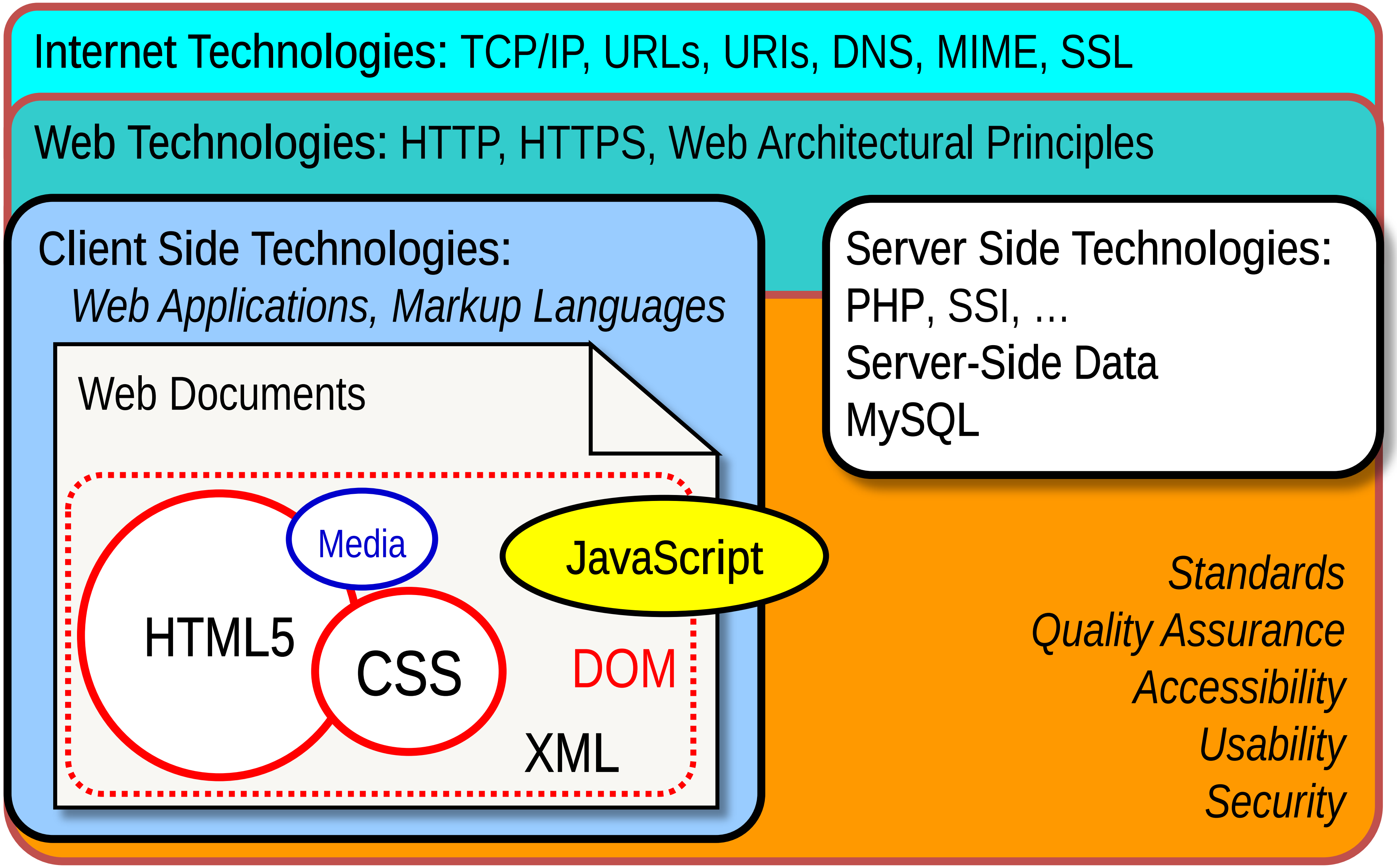
SWINBURNE UNIVERSITY OF TECHNOLOGY

COS10011 Creating Web Applications

Lecture 8 – Server-side Programming PHP: Part 1

Acknowledgement: The contents in this document was adopted from learning material prepared by Alan Colman (Convener, SUT)

Unit of Study Outline


Internet Technologies: TCP/IP, URLs, URIs, DNS, MIME, SSL
Web Technologies: HTTP, HTTPS, Web Architectural Principles
Client Side Technologies:
Web Applications, Markup Languages
Web Documents
Media
HTML5
CSS
JavaScript
DOM
XML
Server Side Technologies:
PHP, SSI, …
Server-Side Data
MySQL
Standards
Quality Assurance
Accessibility
Usability
Security

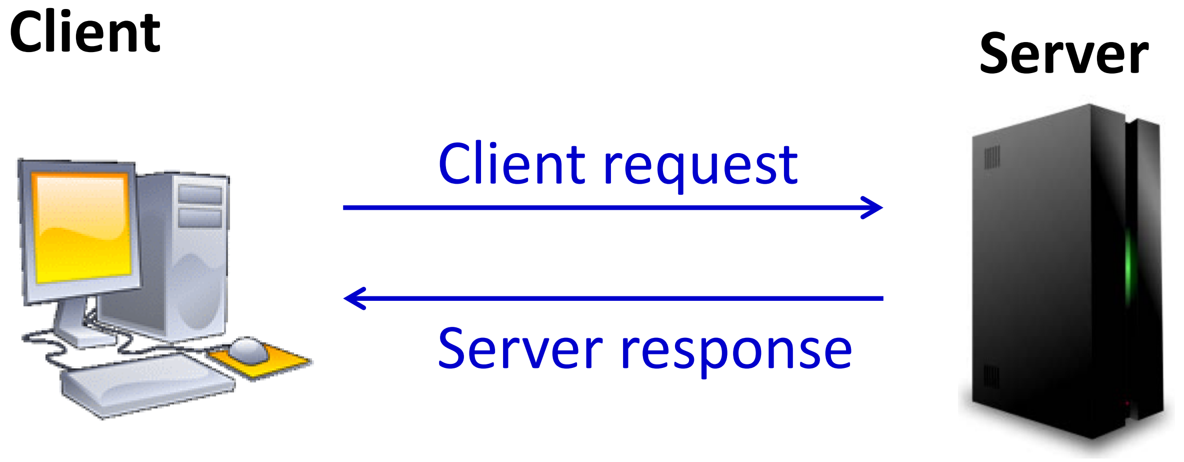
Outline

- Client/Server Architecture
- PHP Scripting
- PHP Variables and Constants
- Data Types
- Arrays
- Expressions
- Functions and Scope
- Control Flow

CLIENT/SERVER ARCHITECTURE

Client/Server Architecture

- A system consisting of a **Client** and a **Server** is known as a two-tier system



The design of a two-tier client/server system

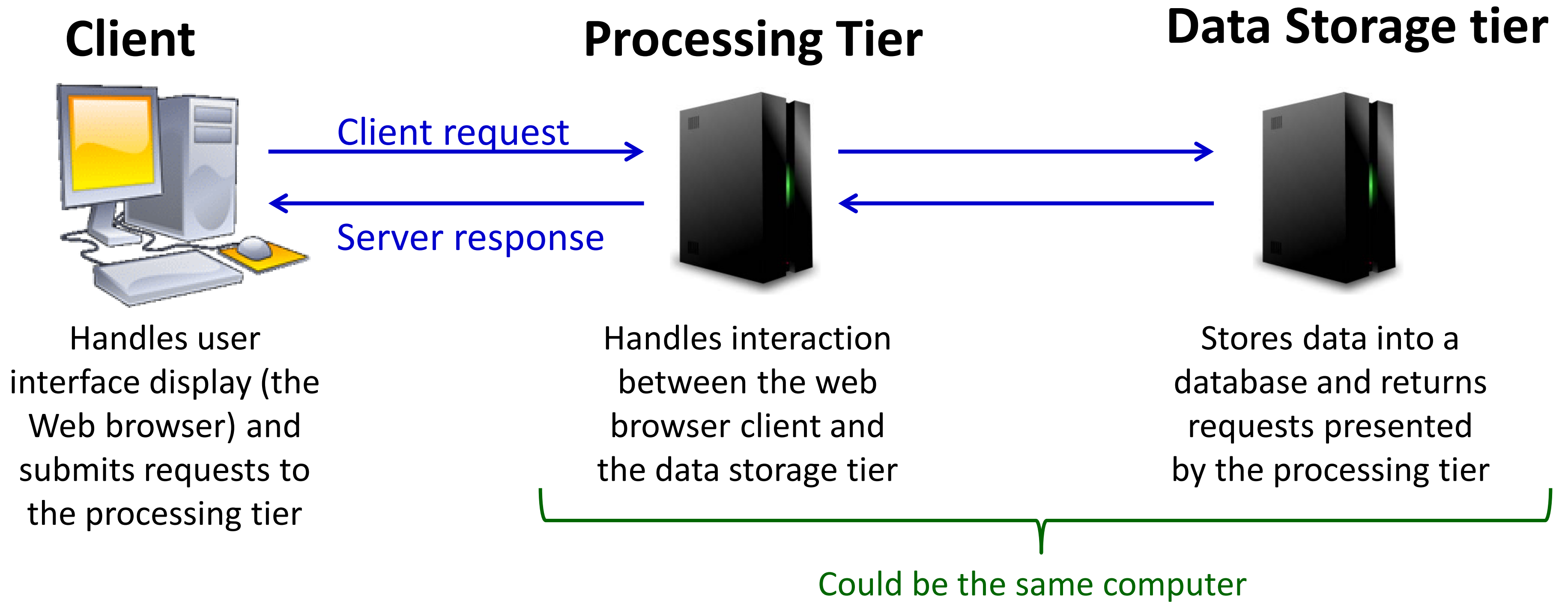
Client/Server Architecture (continued)

- **Client** ("front end"):
  - Presents an interface to the user
  - Gathers information from the user, submits it to a server, then receives, formats, and presents the results returned from the server
- **Server** ("back end"):
  - A computer from which a client requests information
  - Fulfills a request for information by managing the request or serving the requested information to the client
  - Responsible for data storage and management

Client/Server Architecture (continued)

- A **three-tier**, or **multi-tier**, client/server system consists of three distinct pieces:
  - **Client tier**, or **user interface tier**, is the Web browser
  - **Processing tier**, or **middle tier**, handles the interaction between the Web browser client and the **data storage tier**
    - Performs necessary processing or calculations based on the request from the client tier
    - Handles the return of any information to the client tier

Client/Server Architecture (continued)



Handles user interface display (the Web browser) and submits requests to the processing tier

Handles interaction between the web browser client and the data storage tier

Stores data into a database and returns requests presented by the processing tier

The design of a three-tier client/server system

PHP SCRIPTING

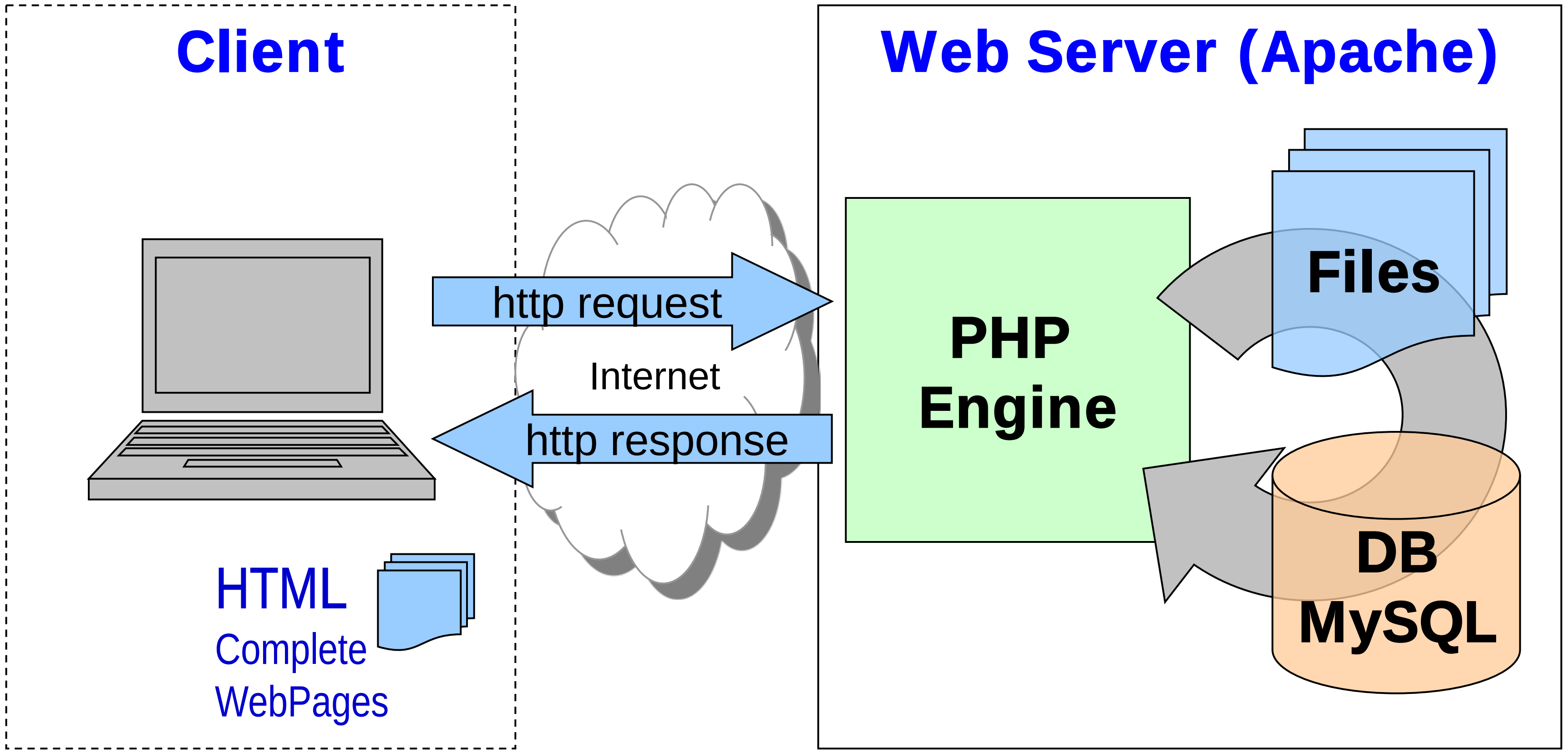
http://php.net/manual/en/langref.php

Server-Side Scripting and PHP

- **Server-side scripting** refers to a scripting language that is executed from a Web server
- **PHP** is a server-side ***embedded scripting language*** that is used to develop interactive web sites
  - Is easy to learn
  - Includes object-oriented programming capabilities
  - Supports many types of databases (MySQL, Oracle, Sybase, ODBC-compliant)

Server-Side Scripting and PHP (continued)

Apache/PHP/MySQL example



Server-Side Scripting and PHP (continued)

What is PHP? http://www.php.net

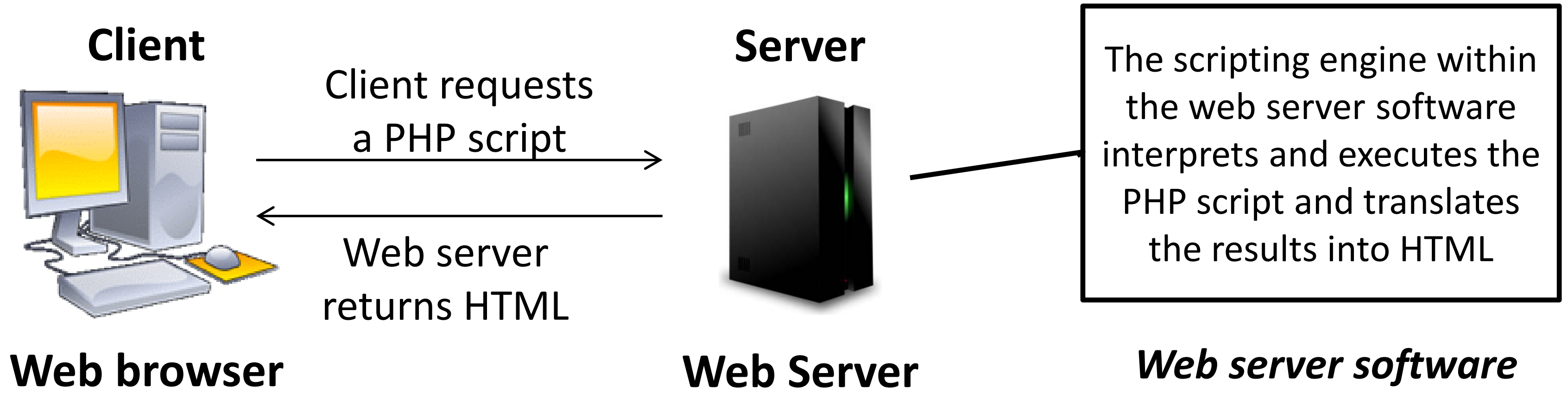
- PHP stands for **P**HP: **H**ypertext **P**reprocessor
- PHP is a server-side scripting language,
- PHP scripts are executed on the server
- PHP supports many databases (MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, etc.)
- PHP is an open source software (OSS)
- PHP is free to download and use
- PHP filename .php

Source: w3schools

Server-Side Scripting and PHP (continued)

- PHP is an ***open source*** programming language
  - *Open source refers to software where source code can be freely used and modified*
- PHP can't access or manipulate a web browser, like JavaScript
- PHP exists and executes solely on a web server, where it performs various types of processing or accesses databases

Server-Side Scripting and PHP (continued)


Client
Client requests a PHP script
Server
The scripting engine within the web server software interprets and executes the PHP script and translates the results into HTML
Web server returns HTML
Web browser
Web Server
Web server software

How a Web server processes a PHP script

- **General rule:**
  Use ***client-side scripting*** to handle user interface processing and light processing, such as form data validation; use ***server-side scripting*** for intensive calculations and data storage.

First PHP Example: first_php.php

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="utf-8" />
        <title>My Website</title>
</head>
<body>
<p>Hello World in unprocessed HTML</p>
<?php
   echo "<p>Hello World in HTML created by PHP</p>"
?>
</body>
</html>
```

Filename must have a PHP extension to be recognised by the pre-processor on the server

PHP code block

Output string in quotes as HTML

PHP Script Blocks

- **Code declaration blocks**
  are separate sections within a web page that are interpreted by the scripting engine
- There are four types of code declaration blocks:
  - **Standard PHP script delimiters**

    <?php statements; **?>**

    Use this coding template
  - (The `<script>` element)

    <script language ="php"> statements; </script>
  - (Short PHP script delimiters)

    <? statements; ?>
  - (ASP-style script delimiters)

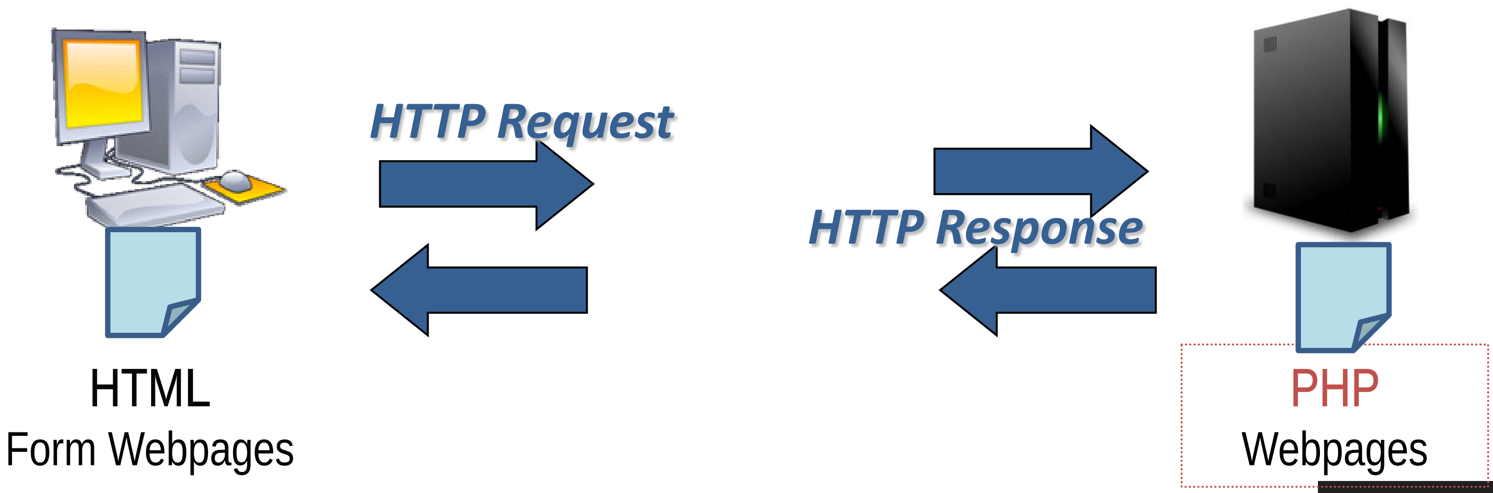
    <% statements; %>

Generating HTML

- To return the results of any processing that occurs within a PHP code block, to the client, you must use an `echo()` or `print()` statement
- The **`echo()`** and **`print()`** **statements** create new text on a Web page that is returned as a response to a client
- `echo` and `print()` statements are virtually identical except:
  - **`print()`** statement accepts only a single argument and returns a value of 1
  - **echo** statement accepts multiple arguments and does not return any value

Generating HTML (continued)

- PHP scripts are executed and only HTML elements are sent back
- If there are no `echo` or `print()` statements the web page will be blank, except if there are HTML codes in the PHP page



Generating HTML (continued)

Example. Given the following PHP file

```
…
<body>
<h1>Multiple Script Sections</h1>
<h2>First Script Section</h2>
<?php echo "<p>Output from the first script
section.</p>"; ?>
<h2>Second Script Section</h2>
<?php echo "<p>Output from the second script
section.</p>";?>
</body>
</html>
```

Beware: Don’t try to HTML validate local source!

Generating HTML (continued)

Example

The following HTML code is sent to the client

```
...
</head>
<body>
<h1>Multiple Script Sections</h1>
<h2>First Script Section</h2>
<p>Output from the first script section.</p>
<h2>Second Script Section</h2>
<p>Output from the second script section.</p>
</body>
</html>
```

Handling quotes

- How would we echo the HTML meta tag

```
<html lang="en">;        ?
```

```
echo "<html lang="en">";    ☹
```

```
echo "<html lang=\"en\">";
```

Escape characters

```
echo "<html lang='en'>";
```

Nested single quotes

SWINBURNE UNIVERSITY OF TECHNOLOGY

PHP Script

PHP script

- uses round brackets ( ) for operator precedence and argument lists
- uses square brackets [ ] for arrays and square bracket notation
- uses curly or brace brackets { } for blocks
- is embedded into an HTML file
- is never sent to a client's Web browser
- is used to dynamically generate a web page

PHP Script (continued)

- A web page document containing PHP code must have an extension of .php
  This is the default extension that most Web servers use to process PHP scripts
- A web page document that does not contain any PHP code should have an **.htm** or **.html** extension

PHP VARIABLES AND CONSTANTS

http://php.net/manual/en/language.variables.php

http://php.net/manual/en/language.constants.php

Example with variables

```
<html>
...
<body>
<h1>Hello World!</h1>
<?php
 echo "<p>";
 $i=1;
 while($i<=5) {
   echo "The number is " . $i . "<br />";
   $i++;
 }
 echo "</p>";
?>
</body>
</html>
```

All variables start with the symbol $.
Note: by default
local scope
(no *var* as in JavaScript)

String concatenation operator in PHP

Would this work?

```
<?php
    for ($i = 1; $i < 7; $i++) {
        echo "<h$i>Heading $i</h$i>";
    }
?>
```

- YES! Variable output can be any HTML – not just text nodes

Interleaving PHP with HTML

Would this work?

```
<?php
    for ($i = 1; $i < 7; $i++) {   echo "<h$i>“
?>
Heading
<?php
    echo "$i</h$i>"; }
?>
```

HTML outside PHP block

- YES! PHP can be arbitrarily interleaved with HTML (but don’t break a string)

Variables and Constants

- The values stored in computer memory are called **variables**
- The values, or data, contained in variables are classified into categories known as **data types**
- The name you assign to a variable is called an **identifier** and it:
  - must begin with a dollar sign ($)
  - can include letters (A to Z, a to z) and numbers (0 to 9) or an underscore (_) … but cannot start with a number
  - cannot include spaces
  - is case sensitive

Variables - Naming

- Suggested naming style for variables

  `$votingAge`

  `or`

  `$voting_age`

- Are the two variable names below referring to the same variable (identifier)?
  - `$firstName`
  - `$FirstName`

PHP is Case Sensitive

Variables - Declaring, initialising,modifying

- Specifying and creating a variable name is called **declaring the variable**
- Assigning a first value to a variable is called **initialising the variable**
- In PHP, you must declare and initialise a variable in the same statement:

```
 $variable_name = value;
```

- You can change the variable’s value at any point

```
 $variable_name = new_value;
```

Variables - Declaring, initialising,modifying

- The data type of a variable (identifiers) or constant depends on the data type of the value assigned to it
  - $unitName = “Creating Web Applications”;
  - $lectureHours = 2;
  - $creditPoints = 12.5;
  - $isCoreUnit = TRUE;

Hint: Give meaningful names

Notice any naming pattern?

Variables – Outputting the Values

- To output the contents of a variable, pass the variable name to the `echo` statement with/without enclosing it in double quotation:

```
$votingAge = 18;
echo $votingAge;
```

Outputting variables in strings

- 3 different techniques

```
<?php
    for ($i = 0; $i < 10; $i++) {
        echo "<p>Number ", $i, "</p>";

        echo "<p>Number " . $i . "</p>";

        echo "<p>Number $i</p>";
    }
?>
```

technique #1 - listing

technique #2 - concatenation

technique #3 - embedded

What would happen here?

```
<?php
    for ($i = 0; $i < 10; $i++) {
        echo "<p>Number $i +1 </p>";
    }
?>
```

Suppose we wanted to add 1 to the variable in the loop

Would not work. Need:

```
        echo "<p>Number ", $i +1, "</p>";
```

Variables – Outputting the Values

- **Note differences if surrounded by double or single quotation marks**

```
echo "<p>The legal voting age is
    $votingAge.</p>";
```

Content of $votingAge will be printed

```
echo '<p>The legal voting age is
    $votingAge.</p>';
```

Text '$votingAge' itself will be printed out.

Hint: If in doubt, separate with commas

```
echo "<p>The legal voting age is ",
           $votingAge, ".</p>”;
```

Constants

- A constant contains information that ***does not change*** during the course of program execution
- Constant names ***do not*** begin with a dollar sign ($)
- Use the **define()** function to create a constant

```
    define("CONSTANT_NAME", value);
```

- The value you pass to the define() function can be a text string, number, or Boolean value
- PHP includes numerous predefined constants that you can use in your scripts
  - e.g. **PHP_INT_MAX**

Constants – Naming

- Suggested naming style for variables

  `PASSING_MARK`

- Which one of the following is a constant?

  `$MAX_ELEMENTS`

  `MAX_ELEMENTS`

Example: use of Constants

do not forget the double quotes

```
<?php
    define ("MAX_ELEMENT", 8);
    echo "<ol>";
    for ($i = 0; $i < MAX_ELEMENT; $i++) {
        echo "<li>item ",($i+1), " </li>";
    }
    echo "</ol>";
?>
```

remember: use list output with embedded calculations

PHP DATA TYPES

http://php.net/manual/en/language.types.php

PHP

PHP is a loosely typed programming language

- **Strongly typed programming languages** require you to declare the data types of variables
  - **Static** or **strong typing** refers to data types that ***do not*** change after they have been declared
  - *C is a strongly typed programming language*
- **Loosely typed programming languages** do not require you to declare the data types of variables
  - **Dynamic** or **loose typing** refers to data types that can change after they have been declared
  - *PHP is a loosely typed programming language*.

Data Types (continued)

- A **data type** is the specific category of information that a variable contains
- Data types that can be assigned only a single value are called **primitive types**

Data Type	Description
Integer	Positive or negative numbers with no decimal places
Floating-point numbers	Positive or negative numbers with decimal places, or expressed in exponential notation
Boolean	Logical value represented by true or false
String	Any sequence of characters
NULL	An empty value

Numeric

PHP supports two numeric data types:

- An **integer** is a positive or negative number with no decimal places (-250, 2, 100, 10,000)
- A **floating-point number** is a number that contains decimal places or that is written in exponential notation (-6.16, 3.17, 2.7541)
  - **Exponential notation**, or **scientific notation**, is short for writing very large numbers or numbers with many decimal places (2.0e11)

Boolean

- A **Boolean** is a value of **true** or **false**
- It decides which part of a program should execute and which part should compare data
- In PHP programming, you can only use true or false
- In other programming languages, you can use integers such as 1 = true, 0 = false

String

- **String** is a sequence of characters
- It is created directly by placing the series of characters between double or single quotes, for example
  - "This is a string"
  - 'This is also a string'

Data Types (continued)

The PHP language supports:

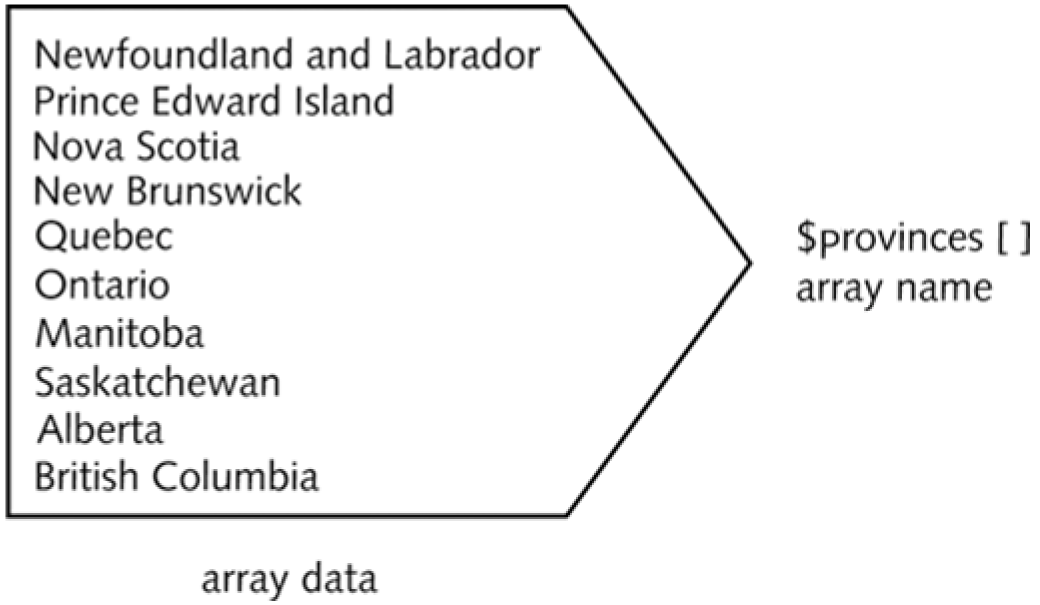
- A **resource** data type – a special variable that holds a reference to an external resource such as a database or XML file
- **Reference** or **composite** data types, which contain multiple values or complex types of information
  - Two reference data types: **arrays** and **objects**

ARRAYS

http://php.net/manual/en/language.types.array.php

Arrays

- An **array** contains a set of data represented by a single variable name



Conceptual example of an array

Declaring and Initialising Indexed Arrays

- An **element** refers to each piece of data that is stored within an array
  - By default, it starts with the number zero (0)
- An **index** is an element's numeric position within the array
  - Referenced by enclosing its index in brackets at the end of the array name:
  - `$provinces[1]`

Creating an 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"
     );
```

Creating an Array (continued)

- Array name and brackets syntax is:

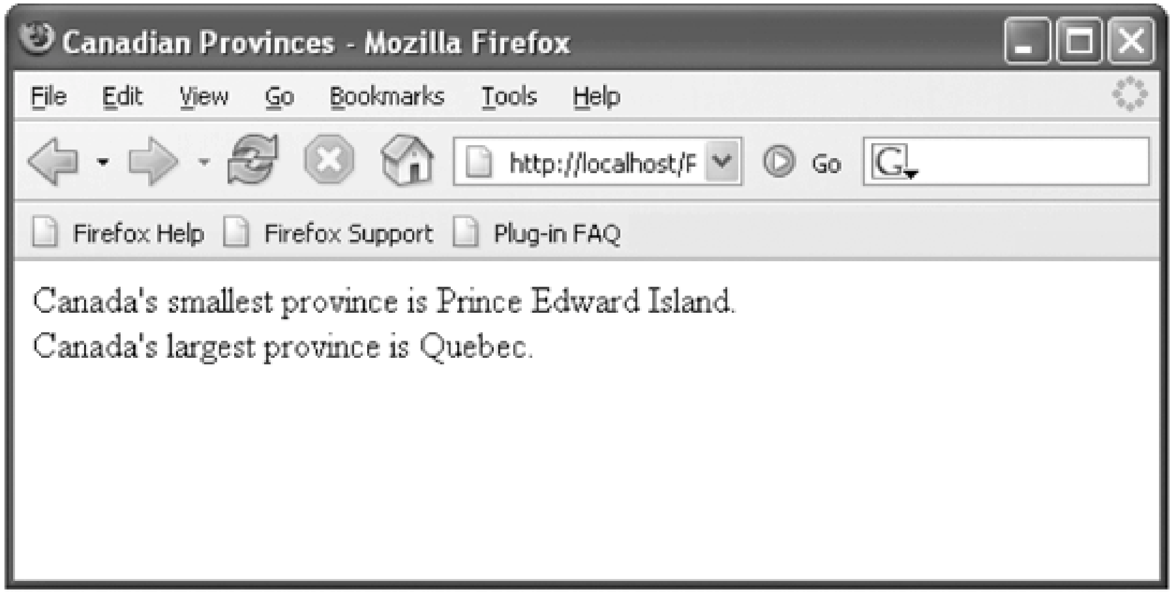
  $array_name[ ]

```
$provinces[] = "Newfoundland and Labrador";
$provinces[] = "Prince Edward Island";
$provinces[] = "Nova Scotia";
$provinces[] = "New Brunswick";
$provinces[] = "Quebec";
$provinces[] = "Ontario";
$provinces[] = "Manitoba";
$provinces[] = "Saskatchewan";
$provinces[] = "Alberta";
$provinces[] = "British Columbia";
```

Note: *In PHP, array elements can be of different data types*

Accessing Element Information

- `echo "<p>Canada's smallest province is $provinces[1].<br />";`
- `echo "Canada's largest province is $provinces[4].</p>";`


Canadian Provinces - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/F
Go
Firefox Help Firefox Support Plug-in FAQ

```
Canada's smallest province is Prince Edward Island.
Canada's largest province is Quebec.
```

Output of elements in the $provinces[ ] array

`count()` Function

- Use the **`count()`** function to find the total number of elements in an array

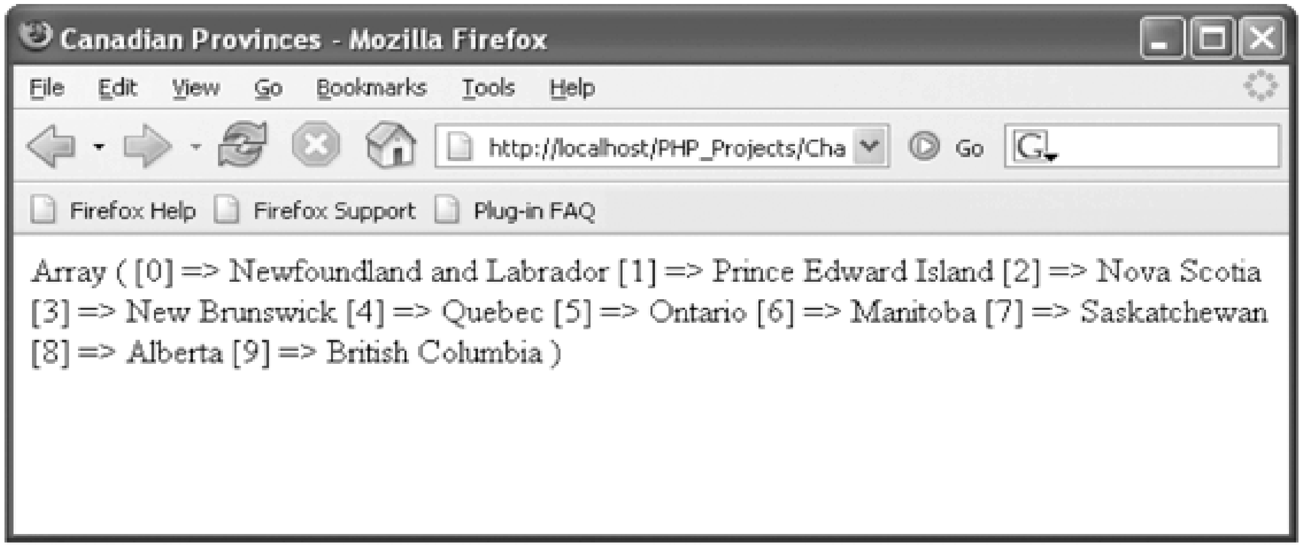
```
$provinces = array("Newfoundland and Labrador",
"Prince Edward Island", "Nova Scotia", "New
Brunswick", "Quebec", "Ontario", " Manitoba",
"Saskatchewan", "Alberta", "British Columbia");

$territories = array("Nunavut", "Northwest
Territories", "Yukon Territory");

echo "<p>Canada has ",
count($provinces), " provinces and ",
count($territories), " territories.</p>";
```

Output:

```
Canada has 10 provinces and 3 territories.
```

`print_r()` Function

- Use to print or return information about variables
- Most useful with arrays because they print the index and value of each element


Canadian Provinces - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/PHP_Projects/Cha
Go
Firefox Help
Firefox Support
Plug-in FAQ

```
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 )
```

Output of the `$provinces[]` array with the `print_r()` function

Modifying Elements

- Include the index for an individual element of the array:

```
$hospitalDepts = array(
    "Anesthesia",           // first element [0]
    "Molecular Biology",    // second element [1]
    "Neurology");           // third element [2]
```

To change the first array element in the `$hospitalDepts[]` array from “Anesthesia” to “Anesthesiology” use:

```
$hospitalDepts[0] = "Anesthesiology";
```

PHP EXPRESSION

http://php.net/manual/en/language.expressions.php

Expressions

- An **expression** is a literal value or variable - that can be evaluated by the PHP scripting engine to produce a result
- **Operands** are variables and literals contained in an expression
- A **literal** is a value such as a literal string or a number
- **Operators** are symbols (e.g. +, *) that are used in expressions to manipulate operands

Expressions (continued)

PHP Operator Types

Operator Type	Description
Array	Performs operations on arrays
Arithmetic	Performs mathematical calculations
Assignment	Assigns values to variables
Comparison	Compares and returns a Boolean value
Logical	Performs Boolean operations on Boolean operands
Special	Performs various tasks, these operators do not fit within other operator categories

- A **binary operator** requires an operand before and after the operator
- A **unary operator** requires a ***single*** operand either before or after the operator

Arithmetic Operators

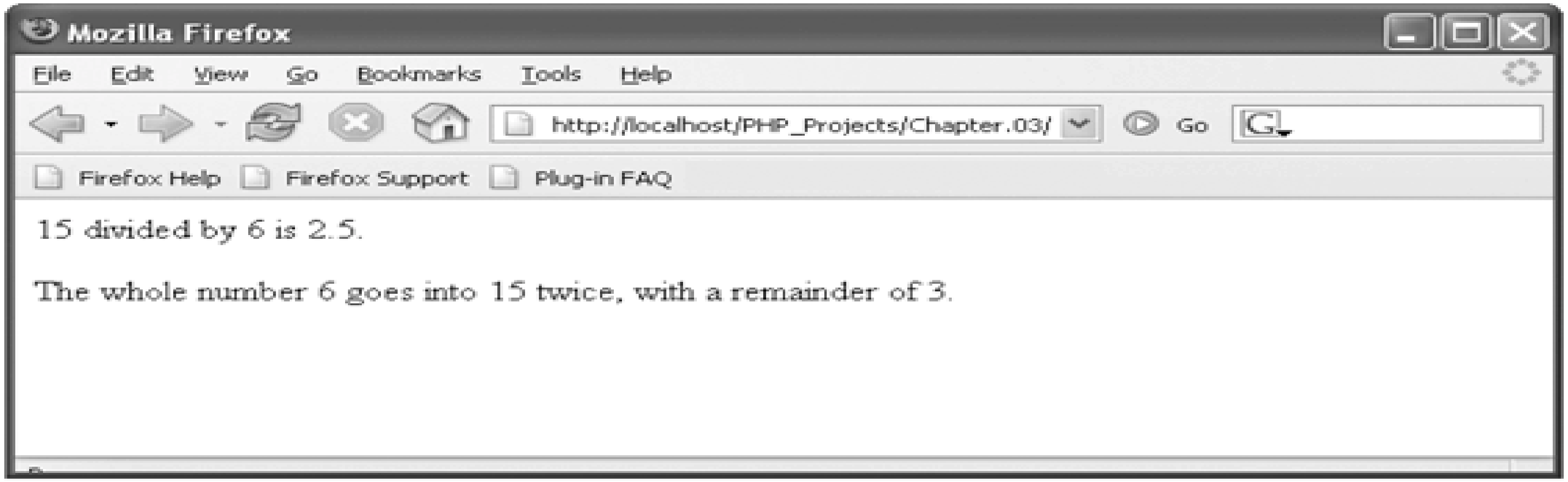
- **Arithmetic operators** are used in PHP to perform mathematical calculations

PHP arithmetic binary operators

Operator	Name	Description
+	Addition	Adds two operands
-	Subtraction	Subtracts one operand from another operand
*	Multiplication	Multiplies one operand by another operand
/	Division	Divides one operand by another operand
%	Modulus	Divides one operand by another operand and returns the remainder

Arithmetic Operators (continued)

```
$divisionResult = 15 / 6;
$modulusResult = 15 % 6;
echo "<p>15 divided by 6 is $divisionResult.</p>"; //
  results to '2.5'
echo "The whole number 6 goes into 15 twice, with a
  remainder of $modulusResult.</p>"; // results to '3'
```


Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/PHP_Projects/Chapter.03/
Go
Firefox Help
Firefox Support
Plug-in FAQ

```
15 divided by 6 is 2.5.

The whole number 6 goes into 15 twice, with a remainder of 3.
```

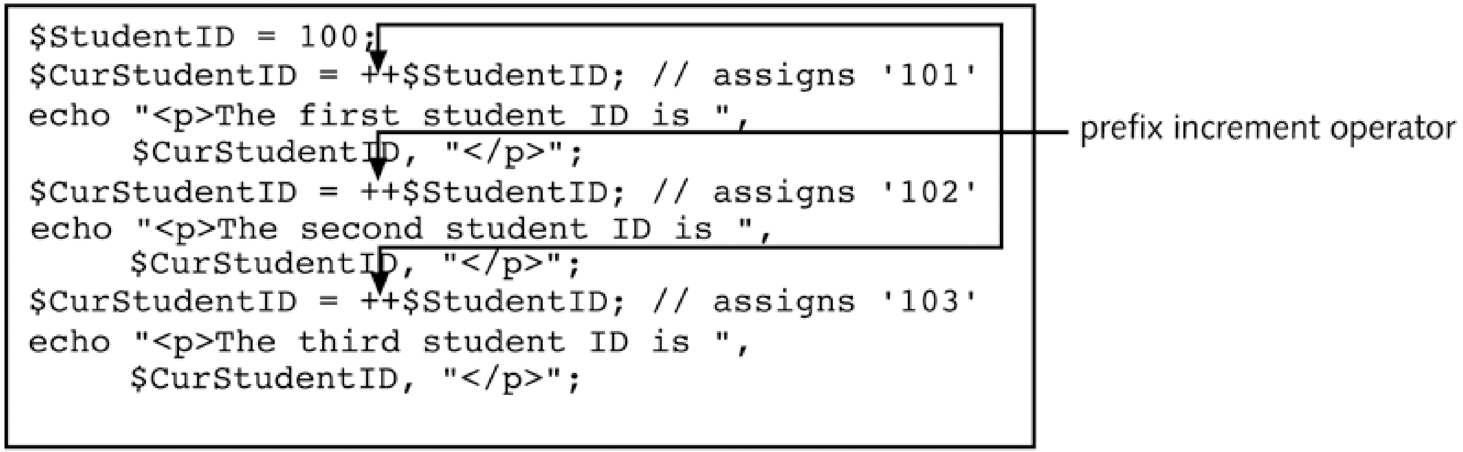
Arithmetic Unary Operators

- The increment (++) and decrement (--) unary operators can be used as prefix or postfix operators
- A **prefix operator** is placed before a variable
- A **postfix operator** is placed after a variable

PHP arithmetic unary operators

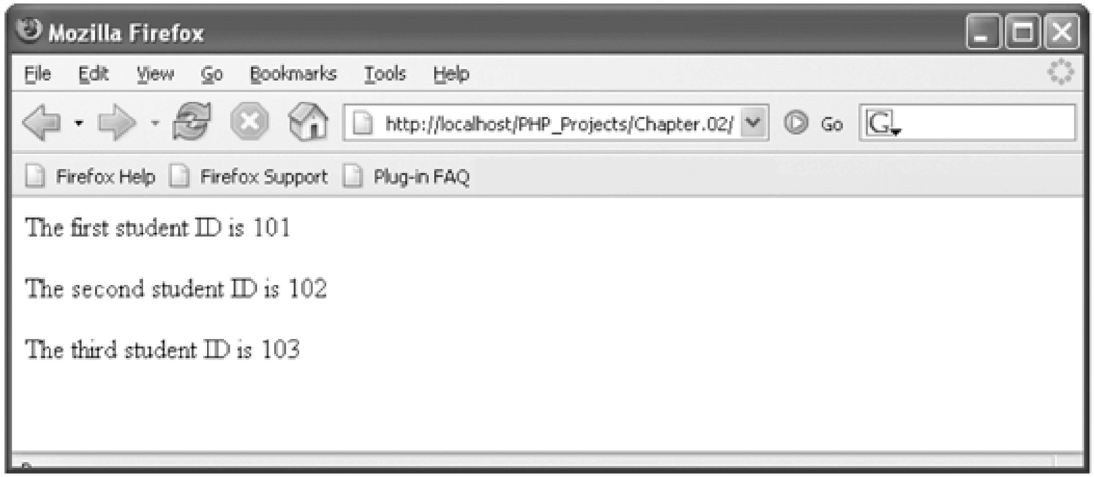
Operator	Name	Description
++	Increment	Increases an operand by a value of one
--	Decrement	decreases an operand by a value of one

Arithmetic Unary Operators (continued)



```
$StudentID = 100;
$CurStudentID = ++$StudentID; // assigns '101'
echo "<p>The first student ID is ",
     $CurStudentID, "</p>";
$CurStudentID = ++$StudentID; // assigns '102'
echo "<p>The second student ID is ",
     $CurStudentID, "</p>";
$CurStudentID = ++$StudentID; // assigns '103'
echo "<p>The third student ID is ",
     $CurStudentID, "</p>";
```

Script that uses the prefix increment operator

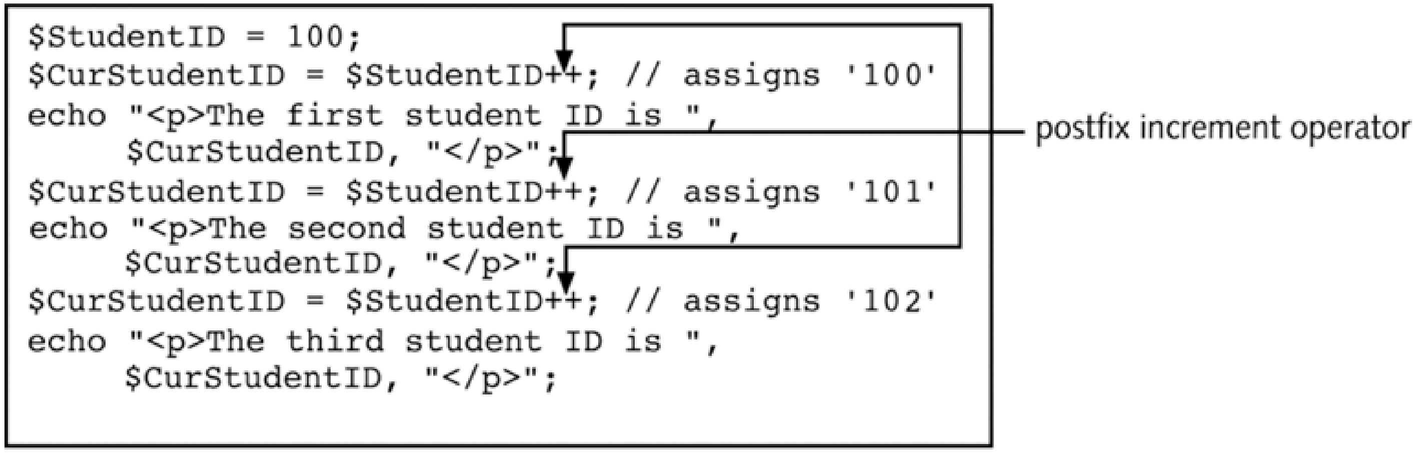


Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/PHP_Projects/Chapter.02/
Go
Firefox Help Firefox Support Plug-in FAQ

```
The first student ID is 101

The second student ID is 102

The third student ID is 103
```

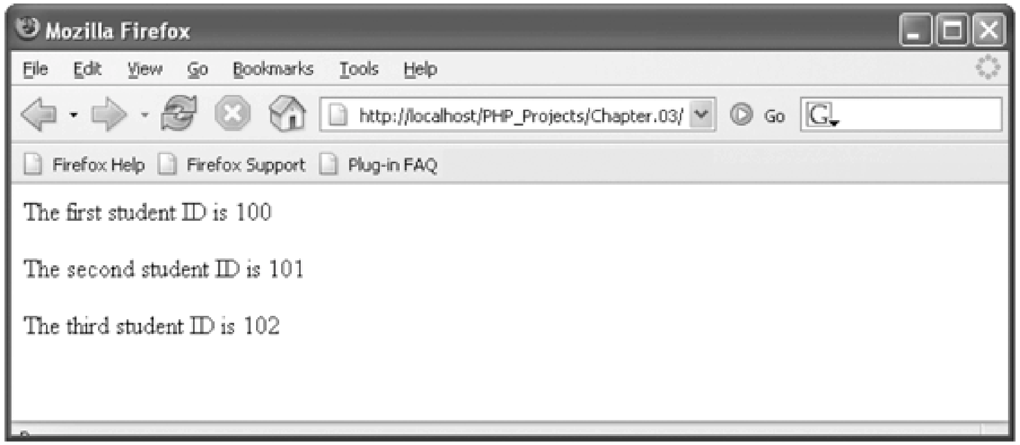
Output of the prefix version of the student ID script

Arithmetic Unary Operators (continued)



```
$StudentID = 100;
$CurStudentID = $StudentID++; // assigns '100'
echo "<p>The first student ID is ",
     $CurStudentID, "</p>";
$CurStudentID = $StudentID++; // assigns '101'
echo "<p>The second student ID is ",
     $CurStudentID, "</p>";
$CurStudentID = $StudentID++; // assigns '102'
echo "<p>The third student ID is ",
     $CurStudentID, "</p>";
```

Script that uses the postfix increment operator



Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/PHP_Projects/Chapter.03/
Go
Firefox Help
Firefox Support
Plug-in FAQ

```
The first student ID is 100

The second student ID is 101

The third student ID is 102
```

Output of the postfix version of the student ID script

Arithmetic Unary Operators (continued)

- What is the difference between prefix increment operator and postfix increment operator ?

Assignment Operators

- **Assignment operators**
  are used for assigning a value to a variable:

```
$myFavoriteSuperHero = "Superman";
$myFavoriteSuperHero = "Batman";
```

- **Compound assignment operators**
  perform mathematical calculations on variables and literal values in an expression, and then assign a new value to the left operand

Assignment Operators (continued)

PHP assignment operators

Operator	Name	description
=	Assignment	Assigns the value of the right operand to the left operand
+=	Compound addition assignment	Adds the value of the right operand to the value of the left operand and assigns the sum to the left operand
-=	Compound subtraction assignment	Subtracts the value of the right operand to the value of the left operand and assigns the difference to the left operand
*=	Compound multiplication assignment	Multiplies the value of the right operand to the value of the left operand and assigns the product to the left operand
/=	Compound division assignment	Divides the value of the right operand to the value of the left operand and assigns the quotient to the left operand
%=	Compound modulus assignment	Divides the value of the right operand to the value of the left operand and assigns the remainder (modulus) to the left operand

Assignment Operators (continued)

```
$x = 100;
$y = 200;
$x += $y;   same as    $x = $x + $y;
```
(Answer: 300)

```
$x = 2;
$y = 6;
$x *= $y;   same as    $x = $x * $y;
```
(Answer: 12)

Comparison and Conditional Operators

- **Comparison operators**
  are used to compare two operands and determine how one operand compares to another.
- A Boolean value of *true* or *false* is returned after two operands are compared
- *The comparison operator compares values, whereas the assignment operator assigns values*
- Comparison operators are used with **conditional statements** and **looping statements**

Comparison and Conditional Operators

(continued)

PHP comparison operators

Operator	Name	Description
==	Equal	Returns true if the operands are equal
===	Strict equal	Returns true if the operands are equal and of the same type
!= or <>	Not equal	Returns true if the operands are not equal
!==	Strict not equal	Returns true if the operands are not equal or not of the same type
>	Greater than	Returns true if the left operand is greater than the right operand
<	Less than	Returns true if the left operand is less than the right operand
>=	Greater than or equal	Returns true if the left operand is greater than or equal to the right operand
<=	Less than or equal	Returns true if the left operand is less than or equal to the right operand

Comparison and Conditional Operators (continued)

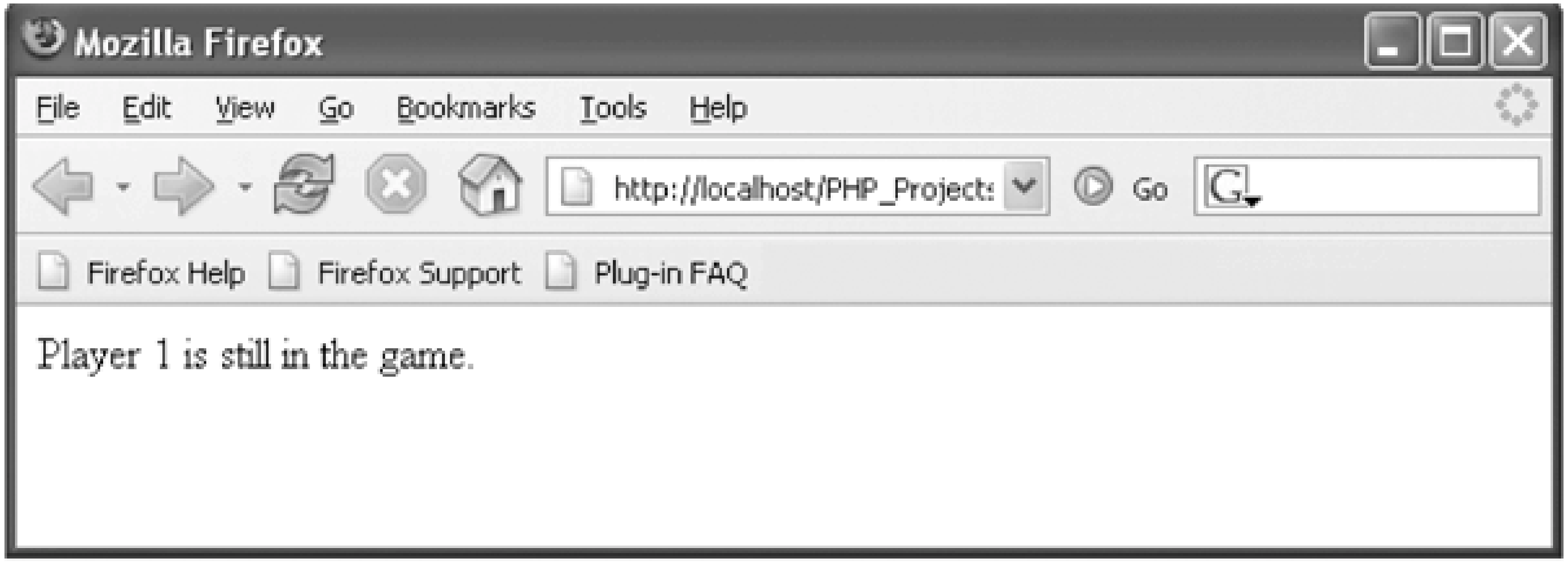
- The **conditional operator** executes one of two expressions, based on the results of a conditional expression
- The syntax for the conditional operator is:

```
conditional expression
      ? expression1 :  expression2;
```

- If the conditional expression evaluates to true, expression1 executes
- If the conditional expression evaluates to false, expression2 executes

Comparison and Conditional Operators
(continued)

```
$blackjackPlayer1 = 20;
($blackjackPlayer1 <= 21)
    ? $result = "Player 1 is still in the game.“
    : $result = "Player 1 is out of the action.";
 echo "<p>", $result, "</p>";
```


Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/PHP_Projects
Go
Firefox Help Firefox Support Plug-in FAQ

```
Player 1 is still in the game.
```

Output of a script with a conditional operator

Logical Operators

- **Logical operators** are used for comparing two Boolean operands for equality
- A Boolean value of true or false is returned after two operands are compared

PHP logical operators

Operator	Name	Description
&&, and	And	Returns true if both the left operand and right operand return a value of true; otherwise, it returns a value of false
||, or	Or	Returns true if either the left operand or right operand returns a value of true; if neither operand returns a value of true, it returns a value of false
!	Not	Returns true if an expression is false and returns false if an expression is true

Special Operators

PHP special operators

Operator	Description
new	Created a new instance of a user-defined or predefined object type
[]	Accesses an element of an array
=>	Specifies the index or key of an array element
,	Separates arguments in a list
?:	Executes one of two expressions based on the results of a conditional expression
instanceof	Returns true is an object is of a specified object type
@	Suppresses any error messages that might be generated by an expression to which it is prepended
(int), (integer), (bool), (boolean), (double), (string), (array), (object)	Casts or transform a variable of one data type into a variable of another data type

***Note**: These Special Operators are introduced throughout this unit as necessary*

Operator Precedence

- **Operator precedence** refers to the order in which operations in an expression are evaluated
- **Associativity** is the order in which operators of equal precedence execute
- Associativity is evaluated on a left-to-right or a right-to-left basis

- *What to do if not certain when you write code?*

Operator Precedence (continued)

Operator precedence in PHP

Operator	Description	Associativity
new	New object	None
[]	Array elements	Right to left
!	Logical Not	Right to left
++	Increment	Right to left
--	Decrement	Right to left
(int) …	Cast	Right to left
@	Suppress error message	Right to left
* / %	Multiplication/division	Left to right
+ - .	Addition/subtraction/string concatenation	Left to right
< <= > >=	Comparison	None
== != <> === !==	Equality	None
&&	Logical And	Left to right
||	Logical Or	Left to right
?:	Conditional	Left to right
= += -= *= /= %=	Assignment	Right to left
and	Logical And	Left to right
or	Logical Or	Left to right
,	List separator	Left to right

FUNCTIONS

http://php.net/manual/en/language.functions.php

Defining Functions

- **Functions** are groups of statements that you can execute as a single unit
- **Function definitions** are the lines of code that make up a function
- Syntax for defining a function is:

```
<?php
function nameOfFunction(parameters) {
      statements;
}
?>
```

Defining Functions (continued)

- Functions, like all PHP code, must be contained within **<?php . . . ?>** tags
- A **parameter** is a variable that is used within a function
- Parameters are placed within the parentheses that follow the function name
- Functions do not have to contain parameters
- The set of curly braces (called **function braces**) contain the function statements

Defining Functions (continued)

- **Function statements** do the actual work of the function and must be contained within the function braces

```
function printNames($name1, $name2, $name3)
{
        echo "<p>$name1</p>";
        echo "<p>$name2</p>";
        echo "<p>$name3</p>";
}
```

Calling Functions

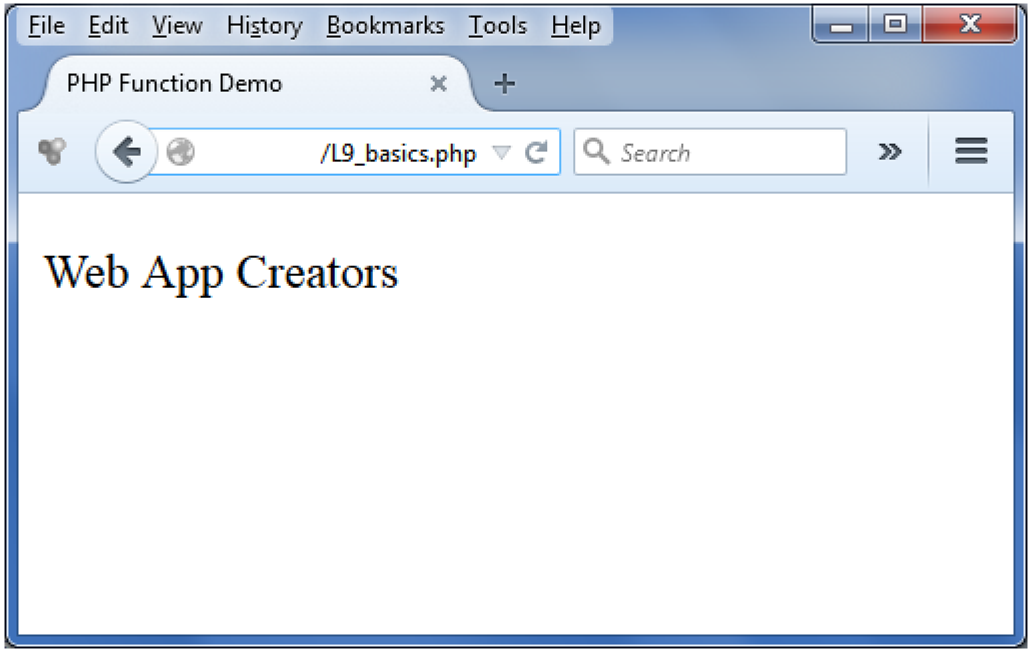
Formal Parameter

Function definition

```
function printCompanyName($companyName) {
        echo "<p>$companyName</p>";
}
printCompanyName("Web App Creators");
```

Actual Parameter

Function invocation



Web App Creators

Output of a call to a custom function

Returning Values

- A **return statement** is a statement that returns a value to the statement that called the function
- A function does not necessarily have to return a value

```
   function averageNumbers($a, $b, $c) {
        $sum = $a + $b + $c;
        $result = $sum / 3;
        return $result;
 }
```

Unlike JavaScript
no var needed

PHP inbuilt (internal) functions

- String functions
  - Examples

    `str_replace()` — Replace all occurrences of the search string with the replacement string

    `htmlspecialchars()` — Convert special characters to HTML entities

    http://php.net/manual/en/ref.strings.php

- Variable Functions
  - Examples

    `is_int()` — Find whether the type of a variable is integer

    `isset()` — Determine if a variable is set and is not NUL

  http://php.net/manual/en/ref.var.php

VARIABLE SCOPE

http://php.net/manual/en/language.variables.scope.php

Understanding Variable Scope

- **Variable scope** is ‘where in your program’ a declared variable can be used
- A variable’s scope can be either global or local
- A **global variable** is one that is declared *outside* a function and is available *to all parts* of your program
- A **local variable** is one that is declared *inside* a function and is only available *within the function* in which it is declared

Understanding Variable Scope (Cont.)

```
<?php
  // all functions usually grouped together
  // in one location
function testScope() {
      $localVariable = "<p>Local variable</p>";
      echo "<p>$localVariable</p>";
                 // prints successfully
}
 $globalVariable = "Global variable";
 testScope();
 echo "<p>$globalVariable</p>";
 echo "<p>$localVariable</p>"; // error message
?>
```

The global Keyword

- With many programming languages, global variables are automatically available to all parts of your program including functions.
- In PHP, we need to use the global keyword to declare a global variable in a function where you would like to use it.

```
<?php
  function testScope() {
        global $globalVariable;
        echo "<p>$globalVariable</p>";
  }
  $globalVariable = "Global variable";
  testScope();
?>
```

If no "global", keyword an error message would be printed out.

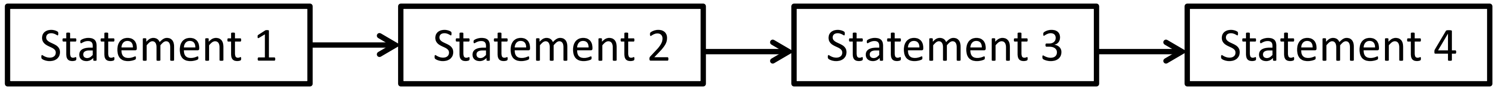
Best Practice: Use local variables. Avoid global.

PHP CONTROL FLOW

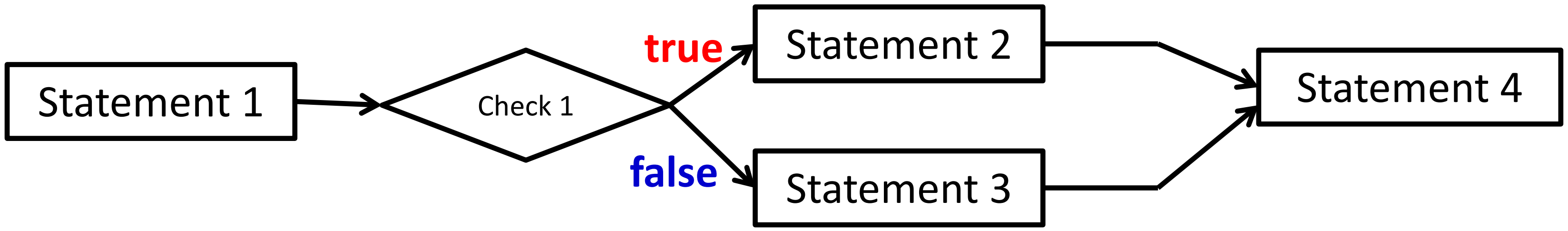
http://php.net/manual/en/language.control-structures.php

Three Models in Programming

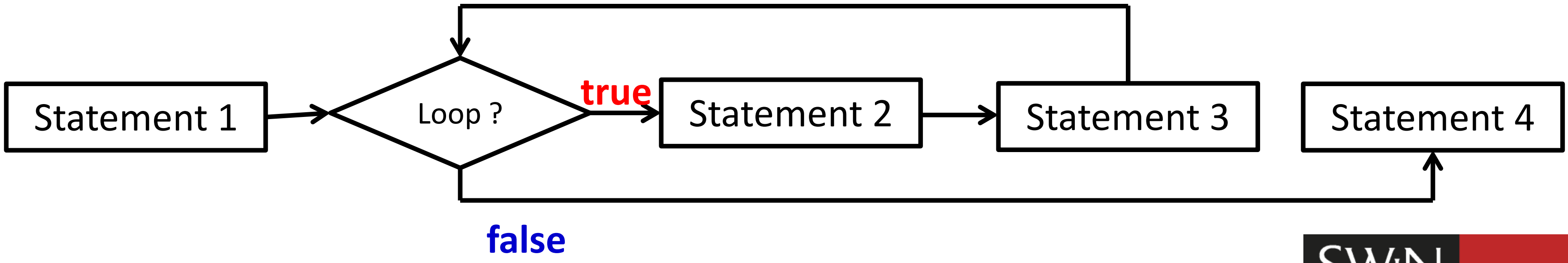
Sequence



Selection



Repetition



Selection

- **Decision making** or **flow control** is the process of determining the order in which statements execute in a program
- The special types of PHP statements used for making decisions are called **decision-making statements** or **decision-making structures**

`if` Statement

- Used to execute specific programming code if the evaluation of a conditional expression returns a value of **true**
- Syntax for a simple `if` statement is:

```
if (conditional expression)
        statement;
```

- Contains three parts:
  - the keyword `if`
  - a conditional expression enclosed within parentheses
  - the executable statements

`if` Statement (continued)

- A **command block** is a group of statements contained within a set of braces
- Each command block must have an opening brace **{**
  and a closing brace **}**

```
$exampleVar = 5;
if ($exampleVar == 5) {    // CONDITION EVALUATES TO 'TRUE'
   echo "<p>The condition evaluates to true.</p>";
   echo '<p>$exampleVar is equal to ', "$exampleVar.</p>";
   echo "<p>Each of these lines will be printed.</p>";
}
echo "<p>This statement will always execute after if.</p>";
```

`if...else` Statement

- An **`if`** statement that includes an **`else`** clause is called an **`if...else` statement**
- An **`else`** clause executes when the condition in an **`if...else`** statement evaluates to **false**
- Syntax for an **`if...else`** statement is:

```
if (conditional expression)
        statement;
else
        statement;
```

`if...else` Statement (continued)

- An `if` statement can be constructed without the `else` clause
- The `else` clause can only be used with an `if` statement

```
$today = "Tuesday";
if ($today == "Monday")
    echo "<p>Today is Monday</p>";
else
    echo "<p>Today is not Monday</p>";
```

Note: Single statements within `if` … `else`, hence no braces needed. ***But Best Practice to include them.***

Nested `if` and `if...else` Statements

- When one decision-making statement is contained within another decision-making statement, they are referred to as nested **decision-making structures**

```
if ($_GET["SalesTotal"] > 50)
    if ($_GET["SalesTotal"] < 100)
        echo "<p>The sales total is "
             ."between 50 and 100.</p>";
```

`switch` Statement

- Controls program flow by executing a specific set of statements depending on the value of an expression
- Compares the value of an expression to a value contained within a special statement called a **case label**
- A **case label** is a specific value that contains one or more statements that execute if the value of the case label matches the value of the switch statement's expression

switch Statement (continued)

- Syntax for the `switch` statement is:

```
switch (expression) {
        case label:
              statement(s);
              break;
        case label:
              statement(s);
              break;
        ...
        default:
              statement(s);
}
```

switch Statement (continued)

```
<?php
  $colour="red";
    switch ($colour) {
    case "red":
          echo "Red!";
          break;
    case "blue":
          echo "Blue!";
          break;
    case "green":
          echo "Green!";
          break;
    default:
      echo "Some other colour!";
}
?>
```

`switch` Statement (continued)

- A `case` label consists of:
  - The keyword case
  - A literal value or variable name (e.g. “Boston”, 75, $var)
  - A colon
- A `case` label can be followed by a single statement or multiple statements
- Multiple statements for a `case` label do not need to be enclosed within a command block
- The `default` **label** contains statements that execute when the value returned by the `switch` statement expression does not match a `case` label
- A `default` label consists of the keyword `default` followed by a colon

Selection – Example with html

- We can mix php coding with html coding.
  For example:

```
<?php if (conditional expression) {
?>
html code block 1 (eg. Show a form)
<?php
} else {
?>
html code block 2 (eg. Hide a form)
<?php
} ?>
```

- It is simpler to use this model for multiple lines of static html, rather than 'echo' all the html code inside the php.

Repetition

- A **loop statement** is a control structure that repeatedly executes a statement or a series of statements while a specific condition is true or until a specific condition becomes true
- There are four types of loop statements:
  - `while` statements
  - `do...while` statements
  - `for` statements
  - `foreach` statements

`while` Statement

- Repeats a statement or a series of statements as long as a given conditional expression evaluates to **`true`**
- Syntax for the **`while`** statement is:

```
while (conditional expression) {
        statement(s);
}
```

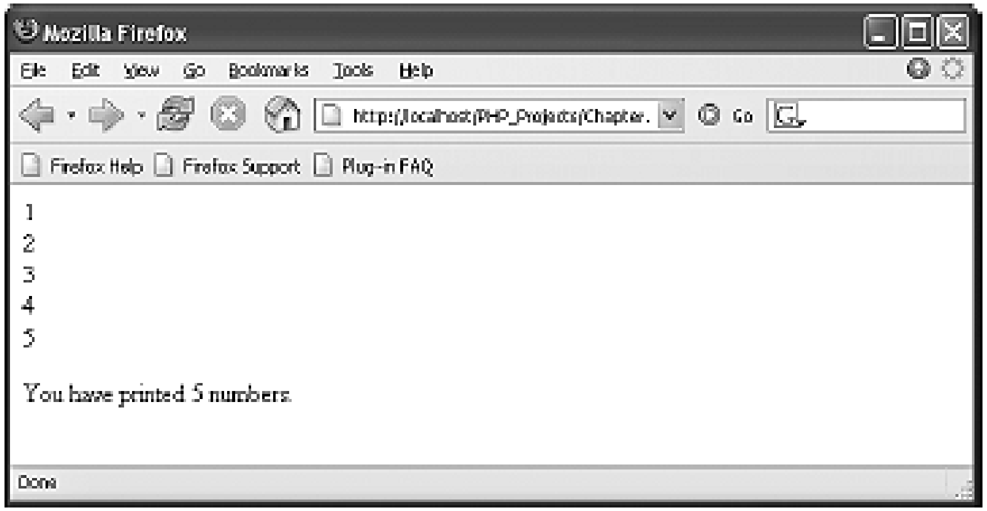
- As long as the conditional expression evaluates to **`true`**, the statement or command block that follows executes repeatedly

`while` Statement (continued)

- Each repetition of a looping statement is called an **iteration**
- A **`while`** statement keeps repeating until its conditional expression evaluates to **`false`**
- A **counter** is a variable that *increments* or *decrements* with each iteration of a loop statement

while Statement (continued)

```
$count = 1;
while ($count <= 5) {
     echo "$count<br />";
     $count++;
}
echo "<p>You have printed 5 numbers.</p>";
```

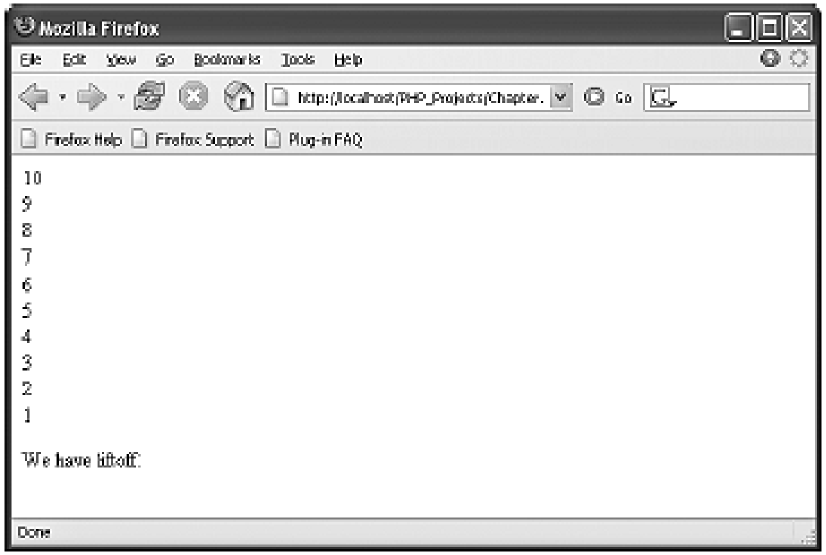

Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/PHP_Projects/Chapter.
Go
Firefox Help Firefox Support Plug-in FAQ
Done

```
1
2
3
4
5

You have printed 5 numbers.
```

Output of a while statement using an increment operator

`while` Statement (continued)

```
$count = 10;
while ($count > 0) {
        echo "$count<br />";
        $count--;
}
echo "<p>We have liftoff.</p>";
```



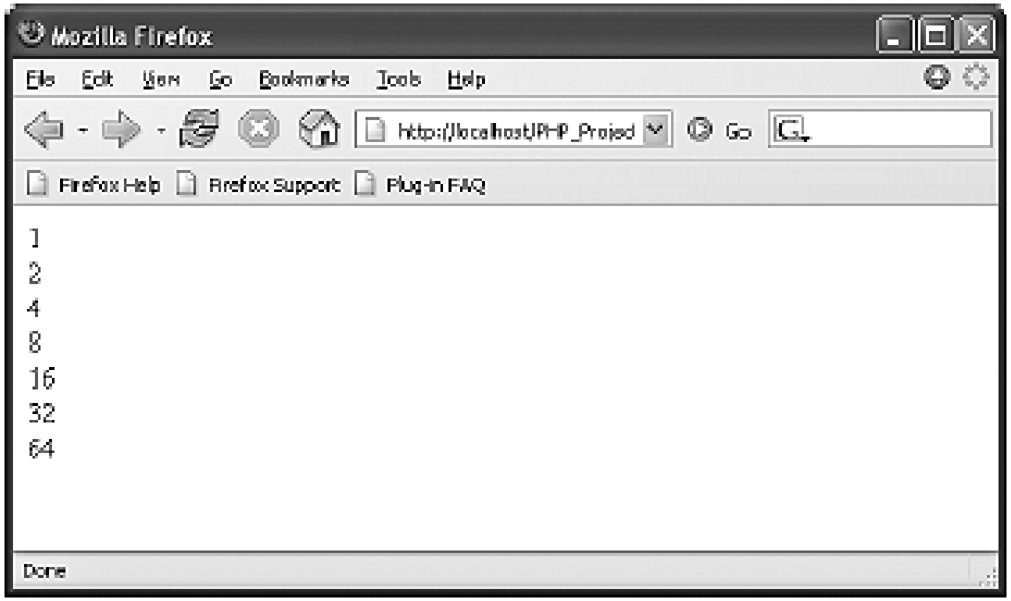
Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/PHP_Projects/Chapter.
Go
Firefox Help
Firefox Support
Plug-in FAQ
Done

```
10
9
8
7
6
5
4
3
2
1

We have liftoff.
```

Output of a `while` statement using a decrement operator

while Statement (continued)

```
$count = 1;
while ($count <= 100) {
      echo "$count<br />";
      $count *= 2;
}
```



Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/PHP_Project
Go
Firefox Help
Firefox Support
Plug-in FAQ
Done

```
1
2
4
8
16
32
64
```

Output of a while statement using the assignment operator *=

`while` Statement (continued)

- In an **infinite loop**, a loop statement never ends because its conditional expression is never false

```
$count = 1;
while ($count <= 10) {
    echo "The number is $count";
}
```

- The `continue` statement
  http://php.net/manual/en/control-structures.continue.php

`do...while` Statement

- Executes a statement or statements once, then repeats the execution as long as a given conditional expression evaluates to **`true`**
- Syntax for the **`do...while`** statement:

```
do {
    statement(s);
} while (conditional expression);
```

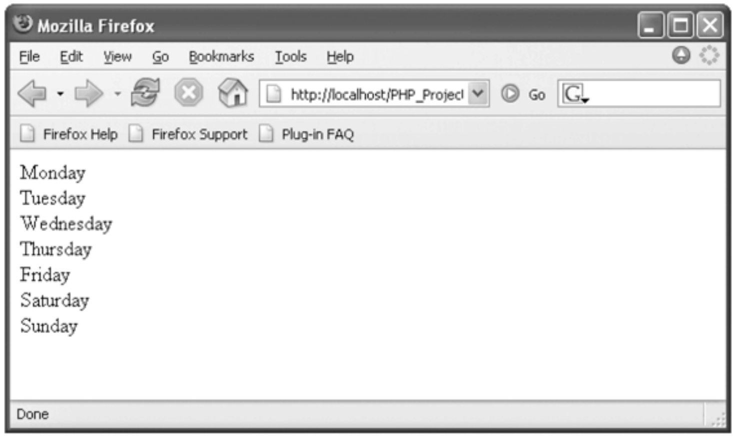
`do...while` Statement (continued)

- **`do...while`** statements *always execute once,* before a conditional expression is evaluated

```
$count = 2;
do {
    echo "<p>The count is equal to"
          . $count . "</p>";
    $count++;
} while ($count < 2);
```

do...while Statement (continued)

```
$daysOfWeek = array("Monday", "Tuesday",
"Wednesday", "Thursday", "Friday", "Saturday",
"Sunday");
$count = 0;
do {
      echo $daysOfWeek[$count], "<br />";
      $count++;
} while ($count < 7);
```


Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/PHP_Project
Go
Firefox Help
Firefox Support
Plug-in FAQ
Done

```
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
```

Output of days of week script in Web browser

`for` Statement

- Used for repeating a statement or a series of statements as long as a given conditional expression evaluates to **`true`**
- If a conditional expression within the **`for`** statement evaluates to true, the **`for`** statement executes and continues to execute repeatedly until the conditional expression evaluates to **`false`**

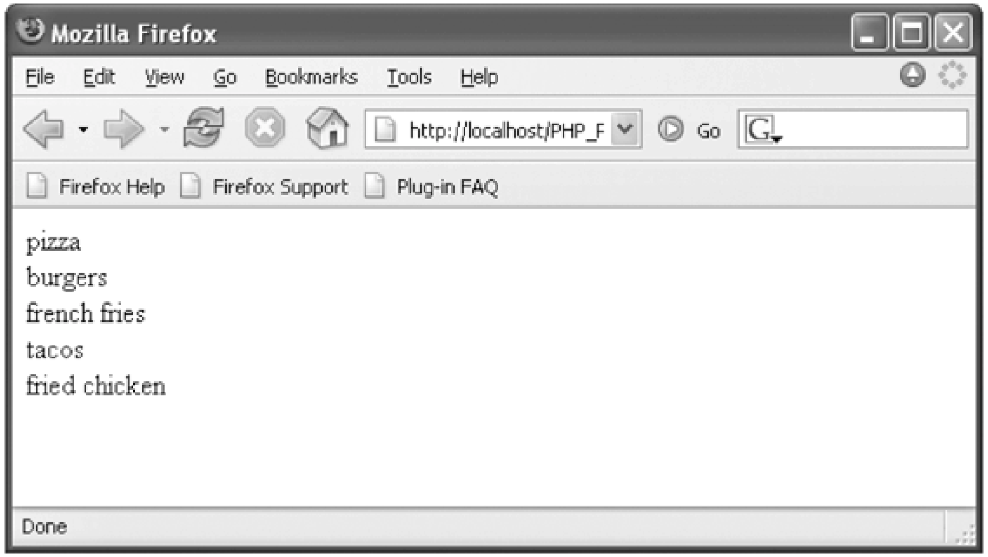
for Statement (continued)

- Can also include code that initialises a counter and changes its value with each iteration
- Syntax of the **for** statement is:

```
for (counter declaration and initialisation;
   condition; update statement) {
      statement(s);
}
```

for Statement (continued)

```
$fastFoods = array("pizza", "burgers", "french fries",
   "tacos", "fried chicken");
for ($count = 0; $count < 5; $count++) {
     echo $fastFoods[$count], "<br />";
}
```



```
pizza
burgers
french fries
tacos
fried chicken
```

Output of fast-foods script

foreach Statement

- Used to iterate or loop through the elements in an **array**

```
$daysOfWeek = array("Monday",
  "Tuesday", "Wednesday", "Thursday",
  "Friday", "Saturday", "Sunday");
   foreach ($daysOfWeek as $day) {
        echo "<p>$day</p>";
```

Note: different from JavaScript:

```
for (variable in collectionOfObject)
{   statements; }
```

`foreach` Statement (continued)

- Used to iterate or loop through the elements in an **array**
- Does not require a counter; instead, you specify an array expression within a set of parentheses following the **`foreach`** keyword
- Syntax for the **`foreach`** statement is:

```
foreach ($array_name as $variable_name) {
     statements;
}
```

Also… Server Side Includes

- SSI provides an easy way to include ***common static content,*** into many pages, such as a standard "header", "nav" or "footer".

- See: http://www.w3schools.com/php/php_includes.asp

SWINBURNE UNIVERSITY OF TECHNOLOGY

COS10011 Creating Web Applications

What's Next?

- PHP part 2

Acknowledgement: The contents in this document was adopted from learning material prepared by Alan Colman (Convener, SUT)