Web Programming

PHP Basics:

- Introduction to PHP
  - a PHP file, PHP workings, running PHP.
- Basic PHP syntax
  - variables, operators, if...else...and switch, while, do while, and for.
- Some useful PHP functions
- How to work with
  - HTML forms, cookies, files, time and date.
- How to create a basic checker for user-entered data

Server-Side Dynamic Web Programming

- **CGI is one of the most common approaches to server-side programming**
  - Universal support: (almost) Every server supports CGI programming. A great deal of ready-to-use CGI code. Most APIs (Application Programming Interfaces) also allow CGI programming.
  - Choice of languages: CGI is extremely general, so that programs may be written in nearly any language. Perl is one of the most popular, but C, C++, Ruby, and Python are also used for CGI programming.
  - Drawbacks: A separate process is run every time the script is requested. A distinction is made between HTML pages and code.

- **Other server-side alternatives try to avoid the drawbacks**
  - Server-Side Includes (SSI): Code is embedded in HTML pages, and evaluated on the server while the pages are being served. Add dynamically generated content to an existing HTML page, without having to serve the entire page via a CGI program.
  - Active Server Pages (ASP and ASP.NET, Microsoft) : The ASP engine is integrated into the web server so it does not require an additional process. It allows programmers to mix code within HTML pages instead of writing separate programs. (Drawback(?) Must be run on a server using Microsoft server software.)
  - Java Servlets (Sun): As CGI scripts, they are code that creates documents. These must be compiled as classes which are dynamically loaded by the web server when they are run.
  - Java Server Pages (JSP): Like ASP, another technology that allows developers to embed Java in web pages.

PHP

- **developed in 1995 by Rasmus Lerdorf (member of the Apache Group)**
  - originally designed as a tool for tracking visitors at Lerdorf's Web site
  - within 2 years, widely used in conjunction with the Apache server
  - developed into full-featured, scripting language for server-side programming
  - free, open-source
  - server plug-ins exist for various servers
  - now fully integrated to work with mySQL databases

- **PHP is somewhat similar to JavaScript, only it's a server-side language**
  - PHP code is embedded in HTML using tags
  - when a page request arrives, the server recognizes PHP content via the file extension (`.php` or `.phtml`)
  - the server executes the PHP code, substitutes output into the HTML page
  - the resulting page is then downloaded to the client
  - user never sees the PHP code, only the output in the page

- **The acronym PHP means (in a slightly recursive definition)**
  - PHP: Hypertext Preprocessor

What do You Need?

- **Our server supports PHP**
  - You don't need to do anything special!
  - You don't need to compile anything or install any extra tools!
  - Create some .php files in your web directory - and the server will parse them for you.

- **Most servers support PHP**
  - Download PHP for free here: http://www.php.net/downloads.php
  - Download MySQL for free here: http://www.mysql.com/downloads/index.html
  - Download Apache for free here: http://httpd.apache.org/download.cgi

(<u>Note:</u> All of this is already present on the CS servers, so you need not do any installation yourself to utilize PHP on our machines.)

Help with PHP

•Loads of information, including help on individual PHP functions may be found at

http://uk.php.net/

Basic PHP syntax

A PHP scripting block always starts with `<?php` and ends with `?>`. A PHP scripting block can be placed (almost) anywhere in an HTML document.

```
<html>
<!-- hello.php COMP519 -->
<head><title>Hello World</title></head>
<body>
 <p>This is going to be ignored by the PHP interpreter.</p>

   <?php echo '<p>While this is going to be parsed.</p>'; ?>

  <p>This will also be ignored by the PHP preprocessor.</p>

   <?php print('<p>Hello and welcome to <i>my</i> page!</p>');
    ?>

  <?php

   //This is a comment

  /*
   This is
   a comment
   block
  */
  ?>

</body>
</html>
```

view the output page

`print` and `echo` for output

`a semicolon (;)` at the end of each statement

// for a single-line comment

/* and */ for a large comment block.

The server executes the print and echo statements, substitutes output.

Scalars

All variables in PHP start with a $ sign symbol. A variable's type is determined by the context in which that variable is used (i.e. there is no strong-typing in PHP).

```
<html><head></head>
<!-- scalars.php COMP519 -->
<body>  <p>
<?php
$foo = true; if ($foo) echo "It is TRUE! <br /> \n";
$txt='1234'; echo "$txt <br /> \n";
$a = 1234; echo "$a <br /> \n";
$a = -123;
echo "$a <br /> \n";
$a = 1.234;
echo "$a <br /> \n";
$a = 1.2e3;
echo "$a <br /> \n";
$a = 7E-10;
echo "$a <br /> \n";
echo 'Arnold once said: "I\'ll be back"', "<br /> \n";
$beer = 'Heineken';
echo "$beer's taste is great <br /> \n";
$str = <<<EOD
Example of string
spanning multiple lines
using "heredoc" syntax.
EOD;
echo $str;
?>
</p>
</body>
</html>
```

view the output page

Four scalar types:

boolean
  true or false

integer,

float,
  floating point numbers

string
  single quoted
  double quoted

Arrays

An array in PHP is actually an ordered map. A map is a type that maps values to keys.

```
<?php
$arr = array("foo" => "bar", 12 => true);
echo $arr["foo"]; // bar
echo $arr[12];    // 1
?>
```

`array()` = creates arrays

key = either an integer or a string.

value = any PHP type.

```
<?php
array(5 => 43, 32, 56, "b" => 12);
array(5 => 43, 6 => 32, 7 => 56, "b" => 12);
?>
```

if no key given (as in example), the PHP interpreter uses (maximum of the integer indices + 1).

if an existing key, its value will be overwritten.

```
<?php
$arr = array(5 => 1, 12 => 2);
foreach ($arr as $key => $value) { echo $key, '=>',
                                        $value);   }
$arr[] = 56;     // the same as $arr[13] = 56;
$arr["x"] = 42; // adds a new element
unset($arr[5]); // removes the element
unset($arr);     // deletes the whole array
$a = array(1 => 'one', 2 => 'two', 3 => 'three');
unset($a[2]);
$b = array_values($a);
?>
```

view the output page

can set values in an array

`unset()` removes a key/value pair

`array_values()` makes reindexing effect (indexing numerically)

*Find more on arrays

Constants

A constant is an identifier (name) for a simple value. A constant is case-sensitive by default. By convention, constant identifiers are always uppercase.

```php
<?php

// Valid constant names
define("FOO",     "something");
define("FOO2",     "something else");
define("FOO_BAR", "something more");

// Invalid constant names  (they shouldn't start
//      with a number!)

define("2FOO",     "something");

// This is valid, but should be avoided:
// PHP may one day provide a "magical" constant
// that will break your script

define("__FOO__", "something");

?>
```

You can access constants anywhere in your script without regard to scope.

Operators

- **Arithmetic Operators:** +, -, *,/ , %, ++, --
- **Assignment Operators:** =, +=, -=, *=, /=, %=

Example	Is the same as
x+=y	x=x+y
x-=y	x=x-y
x*=y	x=x*y
x/=y	x=x/y
x%=y	x=x%y

- **Comparison Operators:** ==, !=, >, <, >=, <=
- **Logical Operators:** &&, ||, !
- **String Operators**: . and .= (for string concatenation)

```
$a = "Hello ";
$b = $a . "World!"; // now $b contains "Hello World!"

$a = "Hello ";
$a .= "World!";
```

Conditionals: if else

Can execute a set of code depending on a condition

```
<html><head></head>
<!-- if-cond.php COMP519 -->
<body>

<?php
$d=date("D");
echo $d, "<br/>";
if ($d=="Fri")
     echo "Have a nice weekend! <br/>";
else
     echo "Have a nice day! <br/>";

$x=10;
if ($x==10)
{
     echo "Hello<br />";
     echo "Good morning<br />";
}

?>

</body>
</html>
```

if (condition)
code to be executed if condition is true;
else
code to be executed if condition is false;

date() is a built-in PHP function that can be called with many different parameters to return the date (and/or local time) in various formats

In this case we get a three letter string for the day of the week.

view the output page

Conditionals: switch

Can select one of many sets of lines to execute

```
<html><head></head>
<body>
<!-- switch-cond.php COMP519 -->
<?php
$x = rand(1,5);  // random integer
echo "x = $x <br/><br/>";
switch ($x)
{
case 1:
  echo "Number 1";
  break;
case 2:
  echo "Number 2";
  break;
case 3:
  echo "Number 3";
  break;
default:
  echo "No number between 1 and 3";
  break;
}
?>

</body>
</html>
```

```
switch (expression)
{
case label1:
  code to be executed if
expression = label1;
  break;
case label2:
  code to be executed if
expression = label2;
  break;
default:
  code to be executed
  if expression is different
  from both label1 and label2;
  break;
}
```

view the output page

Looping: while and do-while

Can loop depending on a condition

```
<html><head></head>
<body>

<?php
$i=1;
while($i <= 5)
{
  echo "The number is $i <br />";
  $i++;
}
?>

</body>
</html>
```

view the output page

```
<html><head></head>
<body>

<?php
$i=0;
do
{
  $i++;
  echo "The number is $i <br />";
}
while($i <= 10);
?>

</body>
</html>
```

view the output page

loops through a block of code if, and as long as, a specified condition is true

loops through a block of code once, and then repeats the loop as long as a special condition is true (so will always execute at least once)

Looping: for and foreach

Can loop depending on a "counter"

```
<?php
for ($i=1; $i<=5; $i++)
{
echo "Hello World!<br />";
}
?>
```

loops through a block of code a specified number of times

view the output page

```
<?php
$a_array = array(1, 2, 3, 4);
foreach ($a_array as $value)
{
   $value = $value * 2;
   echo "$value <br/> \n";
}
?>
```

```
<?php
$a_array=array("a","b","c");
foreach ($a_array as $key => $value)
{
   echo $key . " = " . $value . "\n";
}
?>
```

loops through a block of code for each element in an array

User Defined Functions

Can define a function using syntax such as the following:

```
<?php
function foo($arg_1, $arg_2, /* ..., */ $arg_n)
{
   echo "Example function.\n";
   return $retval;
}
?>
```

Can also define conditional functions, functions within functions, and recursive functions.

Can return a value of any type

```
<?php
function square($num)
{
   return $num * $num;
}
echo square(4);
?>
```

```
<?php
function small_numbers()
{
   return array (0, 1, 2);
}
list ($zero, $one, $two) = small_numbers();
echo $zero, $one, $two;
?>
```

```
<?php
function takes_array($input)
{
   echo "$input[0] + $input[1] = ", $input[0]+$input[1];
}
 takes_array(array(1,2));
?>
```

view the output page

Variable Scope

The scope of a variable is the context within which it is defined.

```
<?php
$a = 1; /* limited variable scope */
function Test()
{
   echo $a;
  /* reference to local scope variable */
}
Test();
?>
```

The scope is local within functions, and hence the value of $a is undefined in the "echo" statement.

```
<?php
$a = 1;
$b = 2;
function Sum()
{
   global $a, $b;
   $b = $a + $b;
}
Sum();
echo $b;
?>
```

`global`

refers to its global version.

```
<?php
function Test()
{
   static $a = 0;
   echo $a;
   $a++;
}
Test1();
Test1();
Test1();
?>
```

`static`

does not lose its value.

view the output page

Including Files

The include() statement includes and evaluates the specified file.

```
//  vars.php
<?php

$color = 'green';
$fruit = 'apple';

?>

// test.php
<?php

echo "A $color $fruit"; // A

include 'vars.php';

echo "A $color $fruit"; // A green apple

?>
```

view the output page

```
<?php

function foo()
{
   global $color;

   include ('vars.php‘);

   echo "A $color $fruit";
}

/* vars.php is in the scope of foo() so     *
 * $fruit is NOT available outside of this  *
 * scope.  $color is because we declared it *
 * as global.                               */

foo();                      // A green apple
echo "A $color $fruit";   // A green

?>
```

view the output page

*The scope of variables in “included” files depends on where the “include” file is added!

You can use the include_once, require, and require_once statements in similar ways.

PHP Information

The `phpinfo()` function is used to output PHP information about the version installed on the server, parameters selected when installed, etc.

```
<html><head></head>
<!- info.php COMP519
<body>
<?php
// Show all PHP information
phpinfo();
?>
<?php
// Show only the general information
phpinfo(INFO_GENERAL);
?>
</body>
</html>
```

view the output page

INFO_GENERAL	The configuration line, php.ini location, build date, Web Server, System and more
INFO_CREDITS	PHP 4 credits
INFO_CONFIGURATION	Local and master values for php directives
INFO_MODULES	Loaded modules
INFO_ENVIRONMENT	Environment variable information
INFO_VARIABLES	All predefined variables from EGPCS
INFO_LICENSE	PHP license information
INFO_ALL	Shows all of the above (default)

Server Variables

The `$_SERVER` array variable is a reserved variable that contains all server information.

```
<html><head></head>
<body>

<?php
echo "Referer: " . $_SERVER["HTTP_REFERER"] . "<br />";
echo "Browser: " . $_SERVER["HTTP_USER_AGENT"] . "<br />";
echo "User's IP address: " . $_SERVER["REMOTE_ADDR"];
?>

<?php
echo "<br/><br/><br/>";
echo "<h2>All information</h2>";
foreach ($_SERVER as $key => $value)
  {
    echo $key . " = " . $value . "<br/>";
  }
?>

</body>
</html>
```

view the output page

$_SERVER info on php.net

The $_SERVER is a super global variable, i.e. it's available in all scopes of a PHP script.

File Open

The `fopen("file_name","mode")` function is used to open files in PHP.

r	Read only.	r+	Read/Write.
w	Write only.	w+	Read/Write.
a	Append.	a+	Read/Append.
x	Create and open for write only.	x+	Create and open for read/write.

```
<?php
$fh=fopen("welcome.txt","r");
?>
```

For w, and a, if no file exists, it tries to create it (use with caution, i.e. check that this is the case, otherwise you'll overwrite an existing file).

```
<?php
if
( !($fh=fopen("welcome.txt","r")) )
exit("Unable to open file!");
?>
```

For x if a file exists, this function fails (and returns 0).

If the `fopen()` function is unable to open the specified file, it returns 0 (false).

File Workings

`fclose()` **closes a file.**

`fgetc()` **reads a single character**

`fwrite(), fputs ()`
writes a string with and without \n

`feof()` **determines if the end is true.**

`fgets()` **reads a line of data**

`file()` **reads entire file into an array**

```
<?php
$myFile = "welcome.txt";
if (!($fh=fopen($myFile,'r')))
exit("Unable to open file.");
while (!feof($fh))
{
$x=fgetc($fh);
echo $x;
}
fclose($fh);
?>
```

```
<?php
$lines = file('welcome.txt');
foreach ($lines as $l_num => $line)
{
 echo "Line #{$l_num}:"
.$line.”<br/>”;
}
?>
```

```
<?php
$myFile = "welcome.txt";
$fh = fopen($myFile, 'r');
$theData = fgets($fh);
fclose($fh);
echo $theData;
?>
```

```
<?php
$myFile = "testFile.txt";
$fh = fopen($myFile, 'a') or
die("can't open file");
$stringData = "New Stuff 1\n";
fwrite($fh, $stringData);
$stringData = "New Stuff 2\n";
fwrite($fh, $stringData);
fclose($fh);
?>
```

Form Handling

Any form element is automatically available via one of the built-in PHP variables (provided that HTML element has a “name” defined with it).

```
<html>
<-- form.html COMP519 -->
<body>
<form action="welcome.php" method="post">
Enter your name: <input type="text" name="name" /> <br/>
Enter your age: <input type="text" name="age" /> <br/>
<input type="submit" /> <input type="reset" />
</form>
</body>
</html>
```

```
<html>
<!-- welcome.php COMP 519 -->
<body>

Welcome <?php echo $_POST["name"]."."; ?><br />
You are <?php echo $_POST["age"]; ?> years old!

</body>
</html>
```

`$_POST`
contains all POST data.

`$_GET`
contains all GET data.

view the output page

Cookie Workings

`setcookie(name,value,expire,path,domain)` **creates cookies.**

```
<?php
setcookie("uname", $_POST["name"], time()+36000);
?>
<html>
<body>
<p>
Dear <?php echo $_POST["name"] ?>, a cookie was set on this
page! The cookie will be active when the client has sent the
cookie back to the server.
</p>
</body>
</html>
```

view the output page

NOTE:
setcookie() **must appear BEFORE** <html> **(or any output) as it's part of the header information sent with the page.**

```
<html>
<body>
<?php
if ( isset($_COOKIE["uname"]) )
echo "Welcome " . $_COOKIE["uname"] . "!<br />";
else
echo "You are not logged in!<br />";
?>
</body>
</html>
```

view the output page

`$_COOKIE`
contains all COOKIE data.

`isset()`
finds out if a cookie is set

use the cookie name as a variable

Getting Time and Date

`date()` and `time ()` **formats a time or a date.**

```
<?php
//Prints something like: Monday
echo date("l");

//Like: Monday 15th of January 2003 05:51:38 AM
echo date("l jS \of F Y h:i:s A");

//Like: Monday the 15th
echo date("l \\t\h\e jS");
?>
```

view the output page

date() returns a string formatted according to the specified format.

```
<?php
$nextWeek = time() + (7 * 24 * 60 * 60);
                   // 7 days; 24 hours; 60 mins; 60secs
echo 'Now:       '. date('Y-m-d') ."\n";
echo 'Next Week: '. date('Y-m-d', $nextWeek) ."\n";
?>
```

view the output page

time() returns current Unix timestamp

*Here is more on date/time formats: http://uk.php.net/manual/en/function.date.php

Required Fields in User-Entered Data

A multipurpose script which asks users for some basic contact information and then checks to see that the required fields have been entered.

```
<html>
<!-- form_checker.php COMP519 -->
<head>
<title>PHP Form example</title>
</head>
<body>
<?php
/*declare some functions*/
```

Print Function

```
function print_form($f_name, $l_name, $email, $os)
{
?>

 <form action="form_checker.php" method="post">
 First Name: <input type="text" name="f_name" value="<?php echo $f_name?>" /> <br/>
 Last Name <b>*</b>:<input type="text" name="l_name" value="<?php echo $l_name?>" /> <br/>
 Email Address <b>*</b>:<input type="text" name="email" value="<?php echo $email?>" /> <br/>
 Operating System: <input type="text" name="os" value="<?php echo $os?>" /> <br/><br/>
 <input type="submit" name="submit" value="Submit" /> <input type="reset" />
 </form>

<?php
}  //**  end of "print_form" function
```

Check and Confirm Functions

```
function check_form($f_name, $l_name, $email, $os)
{
  if (!$l_name||!$email){
  echo "<h3>You are missing some required fields!</h3>";
  print_form($f_name, $l_name, $email, $os);
  }
  else{
   confirm_form($f_name, $l_name, $email, $os);
  }
}  //** end of "check_form" function
```

```
function confirm_form($f_name, $l_name, $email, $os)
{
?>

<h2>Thanks! Below is the information you have sent to us.</h2>
<h3>Contact Info</h3>

<?php
echo "Name: $f_name $l_name <br/>";
echo "Email: $email <br/>";
echo "OS: $os";
}   //** end of "confirm_form" function
```

Main Program

```
/*Main Program*/

if (!$_POST["submit"])
{
?>

 <h3>Please enter your information</h3>
 <p>Fields with a "<b>*</b>" are required.</p>

<?php
 print_form("","","","");
}
else{
 check_form($_POST["f_name"],$_POST["l_name"],$_POST["email"],$_POST["os"]);
}
?>


</body>
</html>
```

view the output page

Learning Outcomes

In the last lectures you have learned

- What is PHP and what are some of its workings.
- Basic PHP syntax
  - variables, operators, if...else...and switch, while, do while, and for.
- Some useful PHP functions
- How to work with
  - HTML forms, cookies, files, time and date.
- How to create a basic checker for user-entered data.