

SOEN 287

WEB PROGRAMMING

PHP – 2

A few extras

User defined functions

Pattern matching



A few extras ... Arrays are very useful

```
<?php
```

```
//first make an array of today's desserts
```

```
$desserts = array("Apple pie a la mode",
```

```
    "Rhubarb strawberry pie",
```

```
    "Creme caramel",
```

```
    "Praline almond cheese cake",
```

```
    "Oreo cheese cake",
```

```
    "Carrot cake",
```

```
    "Banana split",
```

```
    "Chocolate cake");
```

```
// Display the content of the array
```

```
foreach ($desserts as $key => $value)
```

```
    print("[ $key ] => $value <br />");
```



arraysHTML.php

A few extras ... Arrays are very useful

...

```
//make the array into a numbered list  
print "<ol>";  
foreach ($desserts as $dessert) {  
    print "    <li>$dessert</li>";  
} // end foreach  
print "</ol>";
```

...

arraysHTML.php

Content of array as a numbered list

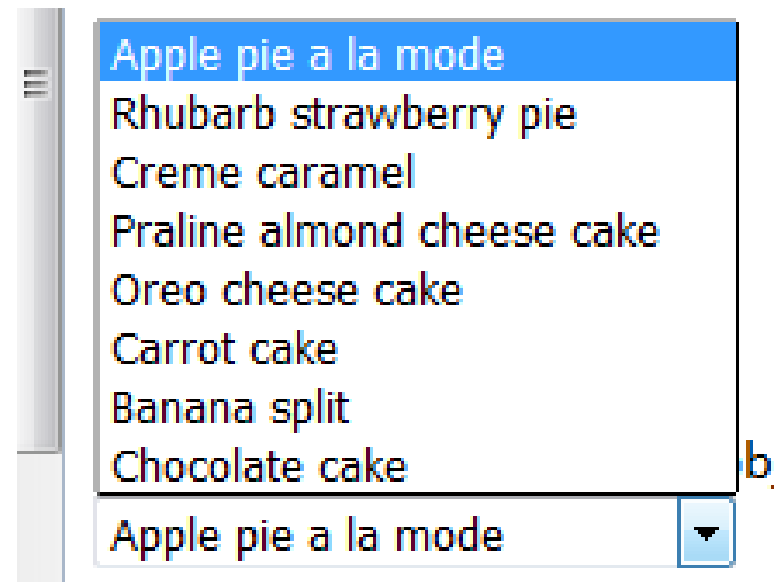
1. Apple pie a la mode
2. Rhubarb strawberry pie
3. Creme caramel
4. Praline almond cheese cake
5. Oreo cheese cake
6. Carrot cake
7. Banana split
8. Chocolate cake

A few extras ... Arrays are very useful

...

```
//make the array into a select object
print "<select name = \"dessert\"><br />";
    foreach ($desserts as $dessert){
        print " <option value =
\"$dessert\">$dessert</option>";
    } // end foreach
print "</select>";
?>
```

arraysHTML.php



A few extras on Output in PHP

Four different ways to output:

1. `print` ✓
2. `echo` ✓
3. `var_dump`
4. `print_r`



print vs echo

- **print:**

- ✓ Outputs a single string
- ✓ Can be used with or without ()
- ✓ Returns 1, so can be used in an expression

- **echo**

- ✓ Outputs one or more strings separated by commas
- ✓ No return value
- ✓ Syntax: () when used with one string, comma when used with many strings

Example: print vs echo

```
$name="Nancy Acemian";
```

```
print $name;
```

```
print "<br />";
```

```
echo $name, "<br />";
```

Or

```
echo ($name) ;
```

```
echo ("<br />") ;
```

OutputStatement.php

Example: print vs echo

```
print $name;  
print "<br />";
```

```
echo $name, "<br />";  
Or
```

```
echo ($ name);
```

```
echo ("<br />");
```

```
echo $name, " ",  
$name, "<br />";
```

```
$cars =  
    array("hisCar" =>  
        "Austin Healey",  
        "myCar" => "BMW",  
        "Camaro",  
        "Datsun",  
        "Ferrari",  
        "kidsCar" => "Hyunda",  
        "Jaguar",  
        "herCar" =>  
            "Lamborghini");
```

OutputStatement.php

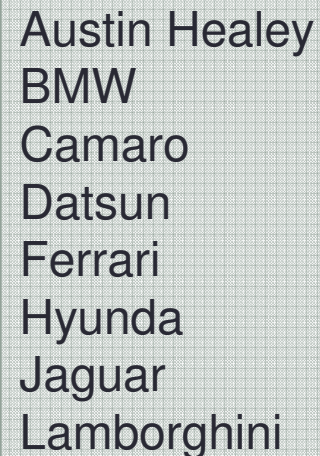
Example: print vs echo

```
foreach ($cars as $x) {  
    print $x;  
    print "<br />"; }
```

```
foreach ($cars as $x) {  
    echo $x , "<br />"; }
```

```
$cars =  
    array("hisCar" =>  
        "Austin Healey",  
        "myCar" => "BMW",  
        "Camaro",  
        "Datsun",  
        "Ferrari",  
        "kidsCar" => "Hyunda",  
        "Jaguar",  
        "herCar" =>  
            "Lamborghini");
```

OutputStatement.php



```
Austin Healey  
BMW  
Camaro  
Datsun  
Ferrari  
Hyunda  
Jaguar  
Lamborghini
```

Example: print vs echo

```
foreach ($cars as $x=>$x_value)
{   print "Key=";
    print $x;
    print ", Value=";
    print $x_value;
    print "<br />"; }
```

```
foreach
($cars as $x=>$x_value) {
    echo "Key=".$x.", Value="
    $x_value."<br>"; }
```

OutputStatement.php

```
$cars =
    array("hisCar" =>
        "Austin Healey",
        "myCar" => "BMW",
        "Camaro",
        "Datsun",
        "Ferrari",
        "kidsCar" => "Hyunda",
        "Jaguar",
        "herCar" =>
            "Lamborghini");
```

```
Key=hisCar, Value=Austin Healey
Key=myCar, Value=BMW
Key=0, Value=Camaro
Key=1, Value=Datsun
Key=2, Value=Ferrari
Key=kidsCar, Value=Hyunda
Key=3, Value=Jaguar
Key=herCar, Value=Lamborghini
```

Example: print_r

```
print_r($cars);
```

```
Array ( [hisCar] => Austin Healey
[myCar] => BMW
[0] => Camaro
[1] => Datsun
[2] => Ferrari
[kidsCar] => Hyunda
[3] => Jaguar
[herCar] => Lamborghini )
```

```
print_r($name);
print_r($num);
```

```
JohnSmith
123
```

```
$cars =
    array("hisCar" =>
        "Austin Healey",
        "myCar" => "BMW",
        "Camaro",
        "Datsun",
        "Ferrari",
        "kidsCar" => "Hyunda",
        "Jaguar",
        "herCar" =>
            "Lamborghini");

$num=123;
```

OutputStatement.php

Example: var_dump

```
var_dump($cars);
```

```
array(8) { ["hisCar"]=>
string(13) "Austin Healey"
["myCar"]=> string(3) "BMW"
[0]=> string(6) "Camaro"
[1]=> string(6) "Datsun"
[2]=> string(7) "Ferrari"
["kidsCar"]=> string(6)
"Hyunda"
[3]=> string(6) "Jaguar"
["herCar"]=> string(11)
"Lamborghini" }
```

```
var_dump($name);
```

```
var_dump($num);
```

```
string(13) "Nancy Acemian" int(123)
```

```
$cars =
    array("hisCar" =>
        "Austin Healey",
        "myCar" => "BMW",
        "Camaro",
        "Datsun",
        "Ferrari",
        "kidsCar" => "Hyunda",
        "Jaguar",
        "herCar" =>
            "Lamborghini");
$num=123;
```

OutputStatement.php

Placing PHP script in markup document

1. Have seen to date imbedded in markup document

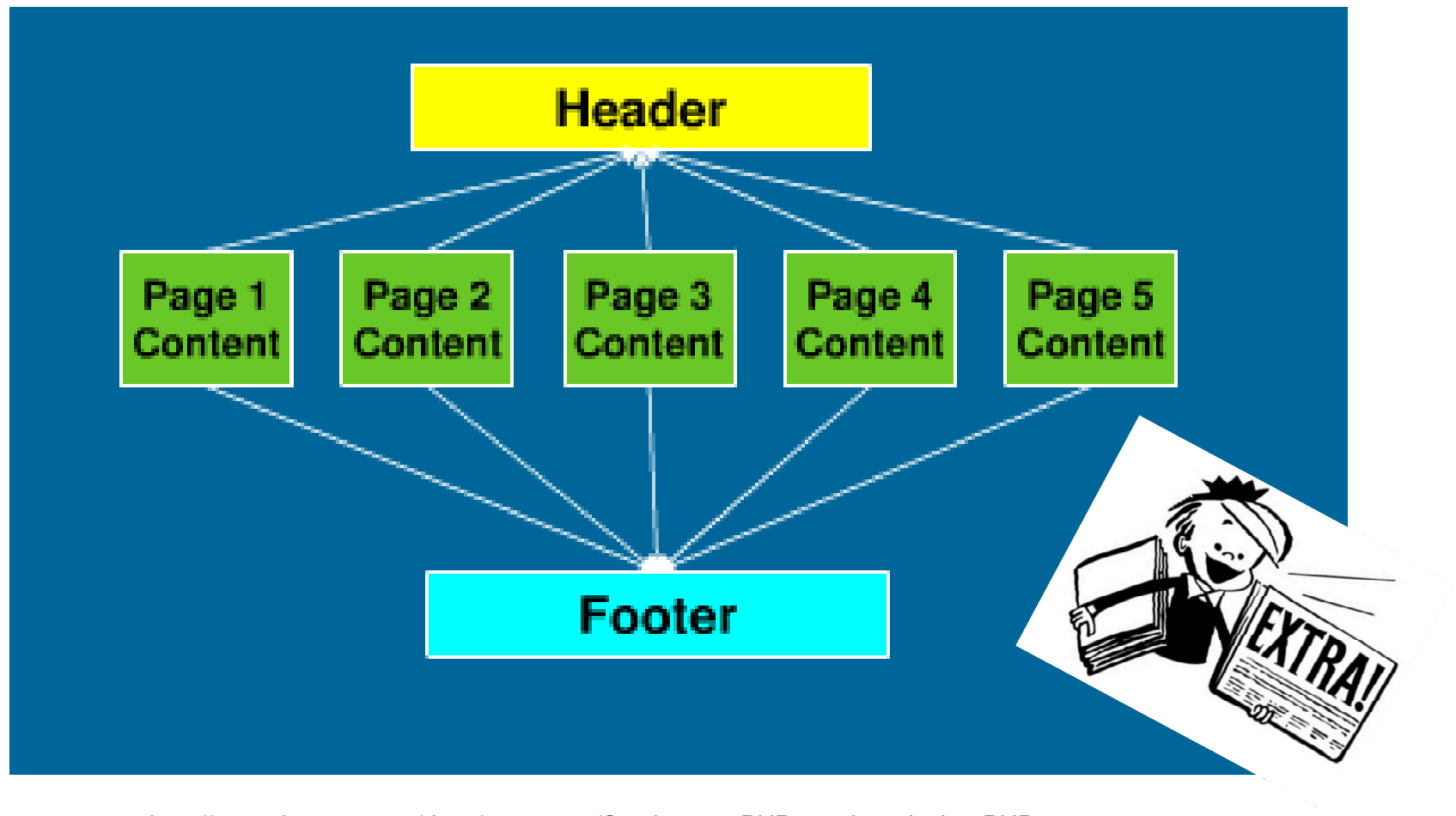
```
<?php  
    ...  
?>
```

2. Can also have in an external file in which case need the
`<include "filename" >` construct

```
<?php  
    <include "file.inc">  
?>
```

Here is an example of how to use it ...

A few extras: Templates for your websites



Templates for your websites ...

Can implement templates for your websites – same header and footer on all pages

Step 1: Create a file with the header information - remove the closing body and html tags.

```
<!DOCTYPE html>
<html lang = "en">
  <head>
    <meta charset="utf-8">
    <title><?php echo $page_title;?></title>
  </head>
  <body>
    
    <hr />
```

[topOfPage.php](#)

Templates for your websites ...

Step 2: Create a file containing the footer info add the closing body and html tags

```
<hr />

<?php echo "<br />", $name ?>
</body>
</html>
```

[bottom.php](#)

Templates for your websites ...

Step 3: Then create the basic template that you will use on all your pages.

Be sure to name the files with a `.php` extension.

[usingTemplate.php](#)

```
<?php
// header
include("topOfPage.php");
?>
```

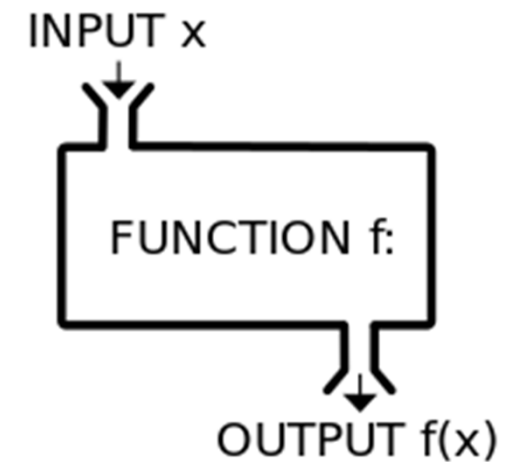
Insert content here

```
<?php
// footer
include("bottom.php");
?>
```

User-Defined Functions

Syntactic form:

```
function function_name(formal_parameters)
{
    ...
}
```



User-Defined Functions

General Characteristics

- Functions need not be defined before they are called
- If you try to redefine a function, it is an error
- Functions can have a variable number of parameters
- Default parameter values are supported

User-Defined Functions

General Characteristics

- Function definitions can be nested
- Function names are NOT case sensitive
- The `return` function is used to return a value;
If there is no `return`, there is no returned value

User-Defined Functions

Parameters

- If the caller sends too many actual parameters, the subprogram ignores the extra ones
- If the caller does not send enough parameters, the unmatched formal parameters are unbound
- The default parameter passing method is pass by value (one-way communication)

User-Defined Functions

Parameters

- To specify **pass-by-reference**, prepend an ampersand to the formal parameter

```
function set_max(&$max, $first, $second) {  
    if ($first >= $second)  
        $max = $first;  
    else  
        $max = $second;  
}
```

User-Defined Functions

Parameters

- If the function does not specify its parameter to be pass by reference, you can prepend an ampersand to the actual parameter in the function call and still get pass-by-reference semantics.

```
someFunction(&$a, $b, 12);
```

User-Defined Functions

Return Values

- Any type may be returned, including objects and arrays, using the `return`
- If a function returns a reference, the name of the function must have a prepended ampersand

```
function &newArray($x) { ... }
```


User-Defined Functions

Scope of Variables

- An undeclared variable in a function has the scope of the function.
- To access a nonlocal variable, it must be declared to be global, as in `global $sum;`
- In a function the declaration `global $sum;` gives access to the variable `$sum` which is declared outside of the function.

User-Defined Functions

Scope of Variables Example

```
<?php
$output = "<p>Hello World!</p>";
function hello1() {
    global $output;
    print $output;
}
function hello2() {
    print $output;
}
hello1();
hello2();
?>
```

How many Hello World
will be printed?

User-Defined Functions

Lifetime of Variables

- Normally, the lifetime of a variable in a function is from its first appearance to the end of the function's execution.
- If want to retain a variable in a function past the execution of script (duration of script) declare it as `static`.

Example of static variable



```
<?php
function myTest() {
    static $x = 0;
    echo $x, "<br />";
    $x++;
}
myTest();
myTest();
myTest();
?>
```

Output:

You try



Write a PHP function with the following specifications:

Parameters

1. An array of numbers (pass by value)
2. Two arrays (pass by reference)

Return value: none

Result:

1st pass by reference contains values >0 of the given array while the 2nd pass by reference contains the values < 0 .

Write the function and code to test it

Pattern Matching

- PHP has two kinds: POSIX and Perl-compatible
- We will focus of Perl-compatible (like tose of JS)
- Will consider only the following 2 functions:
 1. `preg_match(regex, str)`
where
`regex` is the regular expression looking for
`str` is the string to search

Returns a Boolean value

Pattern Matching

- Example of `preg_match`

```
$name="John Smith";  
if(preg_match("/^J"/, $name)  
    echo $name, " begins with J";  
else  
    echo $name, " does not begins with J";
```

pattern.php

Pattern Matching

- Will consider only the following 2 functions:
- `preg_split(regex, str)`
Returns an array of the substrings

→ SHOW `word_table.php`