CIS 4004: Web-Based Information Technology
Spring 2011

Introduction to PHP – Part 2

Instructor : Dr. Mark Llewellyn
markl@cs.ucf.edu
HEC 236, 407-823-2790
http://www.cs.ucf.edu/courses/cis4004/spr2011

Department of Electrical Engineering and Computer Science
University of Central Florida

Functions In PHP

- Functions are at the heart of a well-organized script and will make your code easy to read and reuse.
- Large projects would be unmanageable without functions because the problem of repetitive code would bog down the development process.
- If you haven't had any experience using functions, you can think of a function as an input/output machine. The machine takes the raw materials you feed it (the input) and works with them to produce a product (the output).
- A function accepts values, processes them, and then performs an action (printing to the browser, for example), returns a new value, or both.

Functions In PHP

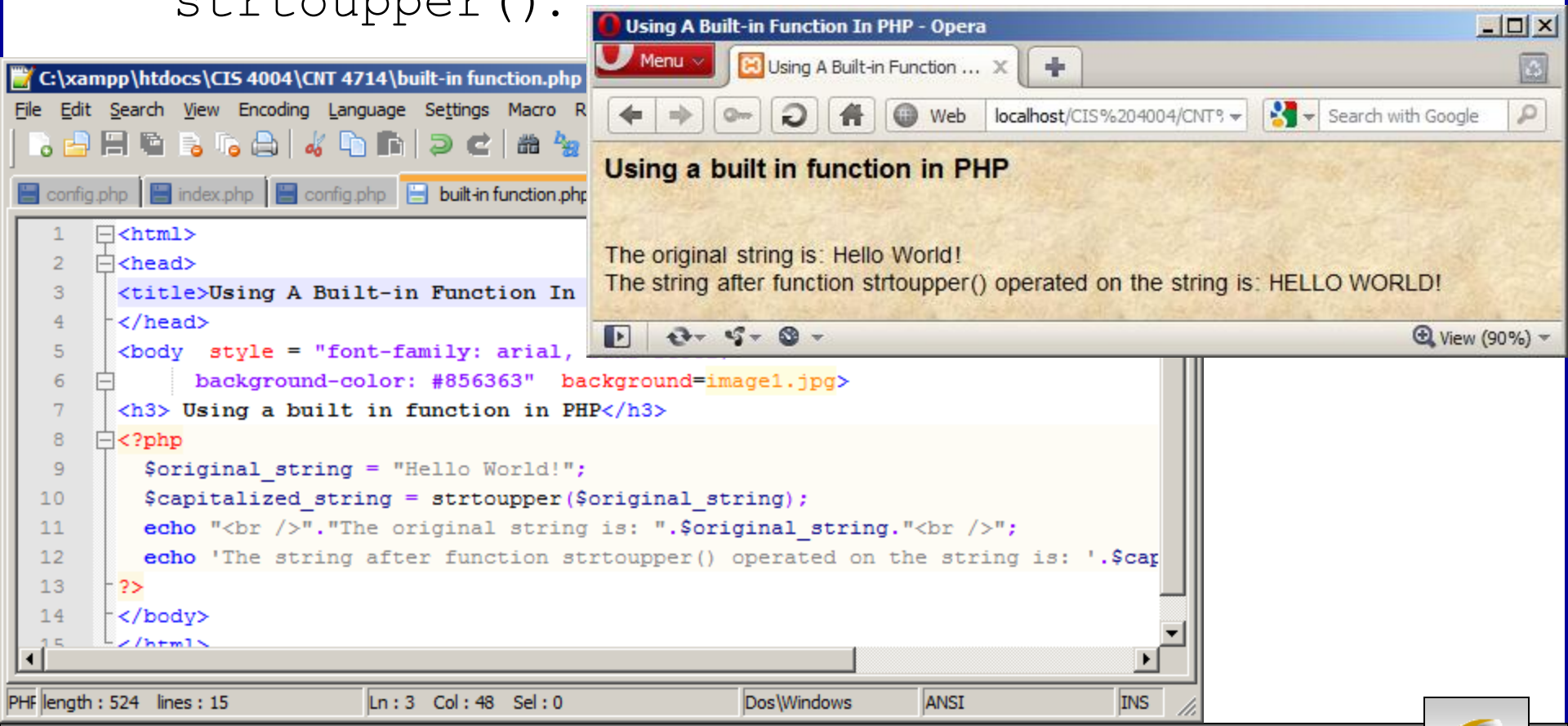
- If you need to bake a cake, you would probably do it yourself, in your own kitchen with your oven. But if you need to bake thousands of cakes, you would probably build or acquire a special cake-baking machine, built for making cakes in massive quantities.
- Similarly, when deciding whether to create a function for reuse, the most important factor to consider is the extent to which it can save you from writing repetitive code.
- If the code you are writing will be used more than once, it is probably best to create a function to represent the code.

Functions In PHP

- A function is a self-contained block of code that can be called by your script.
- When called (or invoked), the function's code is executed and performs a particular task. You can pass values to a function (called arguments), which then uses the values appropriately – storing them, transforming them, displaying them, whatever the function is designed to do. When finished, a function can also pass a value back to the original code that called it into action.
- In PHP, functions come in two flavors – those built in to the language, and those that you define yourself.

Functions In PHP

- PHP has hundreds of built-in functions. Consider the following example that utilizes the built-in function `strtoupper()`.



Functions In PHP

- In the previous example, the function `strtoupper()` is called and passed a variable whose value is represented by a string. The function goes about its business of changing the contents of the string to uppercase letters.
- A function call consists of the function name followed by parentheses. (Note, even a function that has no parameters requires a set of parentheses.) The information being passed to the function (the arguments) are placed between the parentheses.
- For functions that require more than one argument, the arguments are separated by commas:

```
some_function ($an_argument, $another_argument);
```


Functions In PHP

- The `strtoupper()` from the previous example is typical for a function in that it returns a value. Most functions return some information back after they've completed their task – they usually at least tell whether their mission was successful.
- The `strtoupper()` function returns a string value so its usage requires the presence of a variable to accept the returned string, as was the case in the line:

```
$capitalized_string – strtoupper($original_string);
```

- Functions in PHP that return values use a `return` statement within the body of the function. We'll use this in a few more pages when we start constructing our own functions.

Defining Functions In PHP

- You can define your own functions in PHP using the `function` statement:

```
function someFunction($argument1,. . .,argument2) {

        //function code goes here

}
```

- The name of the function follows the function statement and precedes a set of parentheses. If your function requires arguments, you must place the comma-separated variable names within the parentheses. These variables will be filled by the values passed to your function when it is called.

- Even if your function does not require arguments you must still supply the parentheses.

Defining Functions In PHP

- Naming conventions for functions are the same as for normal variables in PHP. As with variables you should apply meaningful names and be consistent in naming and style. Using mixed case in function names is a common convention, thus `myFunction()` instead of `myfunction()` or `my_function()`. (Note: variables names are case sensitive in PHP, function names are not!)

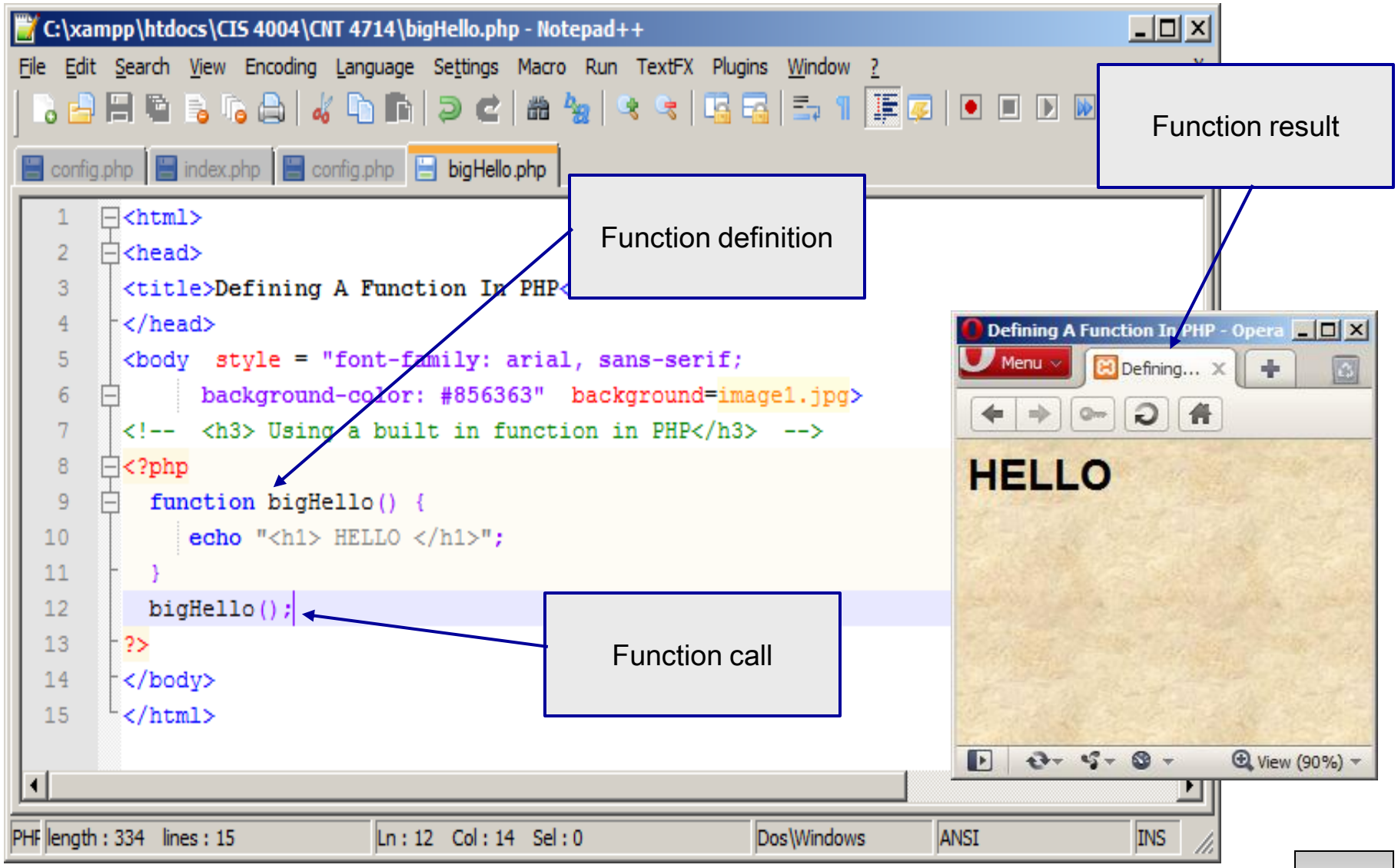
- Let's define a simple function that simply prints out the word "Hello" in big letters.

```
function bigHello() {

  echo "<h1> HELLO </h1>"

}
```

Defining Functions In PHP



Defining Functions In PHP

- For the next example, let's define a function that requires an argument. Actually, let's define two different functions that each take an argument.
- The first function will take a string and print the string with a `<br />` element appended to the string. The second function will do the same, but append two `<br />` elements to the end of the string.

Defining Functions In PHP

*C:\xampp\htdocs\CIS 4004\CNT 4714\functionWithArgument.php - Notepad++

File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ?

config.php | index.php | config.php | functionWithArgument.php

```
2   <head>
3   <title>Defining A Function In PHP With An Argument</title>
4   </head>
5   <body  style = "font-family: arial, sans-serif;
6         background-color: #856363"  background=image1.jpg>
7   <!--  <h3> Using a built in function in PHP</h3>  -->
8   <?php
9     function appendSingleBreak($text) {
10       echo $text."<br />";
11    } //end function appendSingleBreak
12    function appendDoubleBreak($text) {
13       echo $text."<br /><br />";
14    } //end function appendDoubleBreak
15    appendSingleBreak("This is line 1.");
16    appendSingleBreak("This is line 2.");
17    appendSingleBreak("This is line 3.");
18    appendDoubleBreak("This is line 4.");
19    appendDoubleBreak("This is line 5.");
20    appendDoubleBreak("This is line 6.");
21  ?>
22  </body>
23  </html>
```

PHP length : 735 lines : 23 | Ln : 20 Col : 40 Sel : 0 | Dos\Windows ANSI INS

Defining A Function In PHP With...

Menu | Defini...

This is line 1.
This is line 2.
This is line 3.
This is line 4.

This is line 5.

This is line 6.

View (90%)

Defining Functions In PHP

- For the next example, let's define a function that requires two arguments. We'll basically repeat the exercise from the previous example, but in this case rather than writing two different functions that differ only in the number of `<br />` elements they append to a line of text, the new function will have a second argument that represents the number of `<br />` elements to be appended. Clearly this would be more efficient, in terms of code, than creating a different function for each number of
 elements we might want to append.

- In the first version of this example, shown on the next page, I simply repeated the same effect as in the previous version, so the two screen shots from the browser should look identical.

- The second version of this example, shown on page 15, a different effect is produced by the function calls.

C:\xampp\htdocs\CIS 4004\CNT 4714\functionWithTwoArguments.php - Notepad++

File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ? X

config.php | index.php | config.php | functionWithTwoArguments.php

```
2  <head>
3  <title>Defining A Function In PHP With Two Arguments</title>
4  </head>
5  <body  style = "font-family: arial, sans-serif;
6        background-color: #856363"  background=image1.jpg>
7  <!--  <h3> Using a built in function in PHP</h3>  -->
8  <?php
9    function appendBreaks($text, $number) {
10      echo $text;
11      for($i = 1; $i <= $number; $i++){
12         echo "<br />";
13      }
14   } //end function appendBreaks
15   appendBreaks("This is line 1.", 1);
16   appendBreaks("This is line 2.", 1);
17   appendBreaks("This is line 3.", 1);
18   appendBreaks("This is line 4.", 2);
19   appendBreaks("This is line 5.", 2);
20   appendBreaks("This is line 6.", 0);
21 ?>
22 </body>
23 </html>
```

Defining A Function In PHP...

Menu

This is line 1.
This is line 2.
This is line 3.
This is line 4.

This is line 5.

This is line 6.

View (90%)

PHP length : 668 lines : 23 | Ln : 23 Col : 8 Sel : 0 | Dos\Windows | ANSI | INS

```
<head>
<title>Defining A Function In PHP With Two Arguments
</head>
<body  style = "font-family: arial, sans-serif;
      background-color: #856363"  background=image1.j
<!--  <h3> Using a built in function in PHP</h3>  -->
<?php
  function appendBreaks($text, $number) {
     echo $text;
     for($i = 1; $i <= $number; $i++){
        echo "<br />";
     }
  } //end function appendBreaks
  appendBreaks("This is line 1.", 4);
  appendBreaks("This is line 2.", 1);
  appendBreaks("This is line 3.", 3);
  appendBreaks("This is line 4.", 10);
  appendBreaks("This is line 5.", 2);
  appendBreaks("This is line 6.", 0);
?>
</body>
</html>
```

Defining Functions In PHP

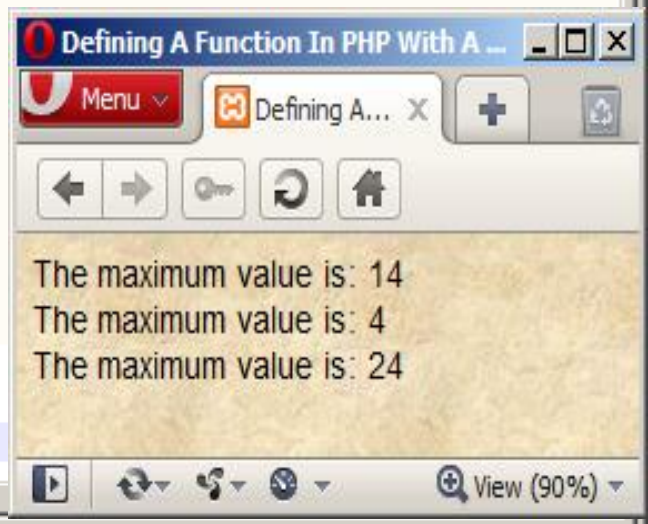
- As a final example of simple function definition, let's construct a function that returns a value.
- In the previous two examples, the string that had the `<br />` elements appended to it was simply printed out in the browser. Sometimes, however, you will want the function to provided a value that you can work with yourself. For example, if the function had returned the appended string, we could have passed that to another function to further process the amended string before it was printed.
- Let's construct a function that will take three integer arguments and determine the largest of the argument values and return this value to the caller.

C:\xampp\htdocs\CIS 4004\CNT 4714\functionWithReturnValue.php - Notepad++

File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ?

config.php | index.php | config.php | functionWithReturnValue.php

```
1  <html>
2  <head>
3  <title>Defining A Function In PHP With A Return Value</title>
4  </head>
5  <body  style = "font-family: arial, sans-serif;
6        background-color: #856363"  background=image1.jpg>
7  <?php
8    function findMax($one, $two, $three) {
9       if (($one >= $two) && ($one  >= $three)) {
10         return $one;
11      } elseif (($two >= $one) && ($two >= $three)) {
12         return $two;
13      } elseif (($three >= $one) && ($three >= $two)) {
14         return $three;
15      }
16   } //end function findMax
17   $currentMax = findMax(12,13,14);
18   echo "The maximum value is: $currentMax <br />";
19   $currentMax = findMax(2,4,3);
20   echo "The maximum value is: $currentMax <br />";
21   $currentMax = findMax(24,12,3);
22   echo "The maximum value is: $currentMax <br />";
23 ?>
24 </body>
25 </html>
```



PHP length : 775 lines : 25 | Ln : 25 Col : 8 Sel : 0 | Dos\Windows | ANSI | INS

Defining Functions In PHP

- The `return` statement can return a value or nothing at all.
- How you arrive at a value passed by a `return` statement can vary.
  - The value can be hard-coded: **`return 4;`**
  - It can be the result of an expression: **`return $a/$b;`**
  - It can be the value returned by yet another function call:

```
return anotherFunction($an_argument);
```

Variable Scope

- A variable that is declared within a function remains local to that function. In other words, that variable is not available outside of the function or within other functions.
- This is referred to as the scope of a variable.
- This also implies that variable names are not required to be unique across functions. Therefore the same variable can be defined in more than one function.
- The following example illustrates the scope of a variable. Notice that both functions `functionOne` and `functionTwo` declare variables named `myInt`.

C:\xampp\htdocs\CIS 4004\CNT 4714\variable scope.php - Notepad++

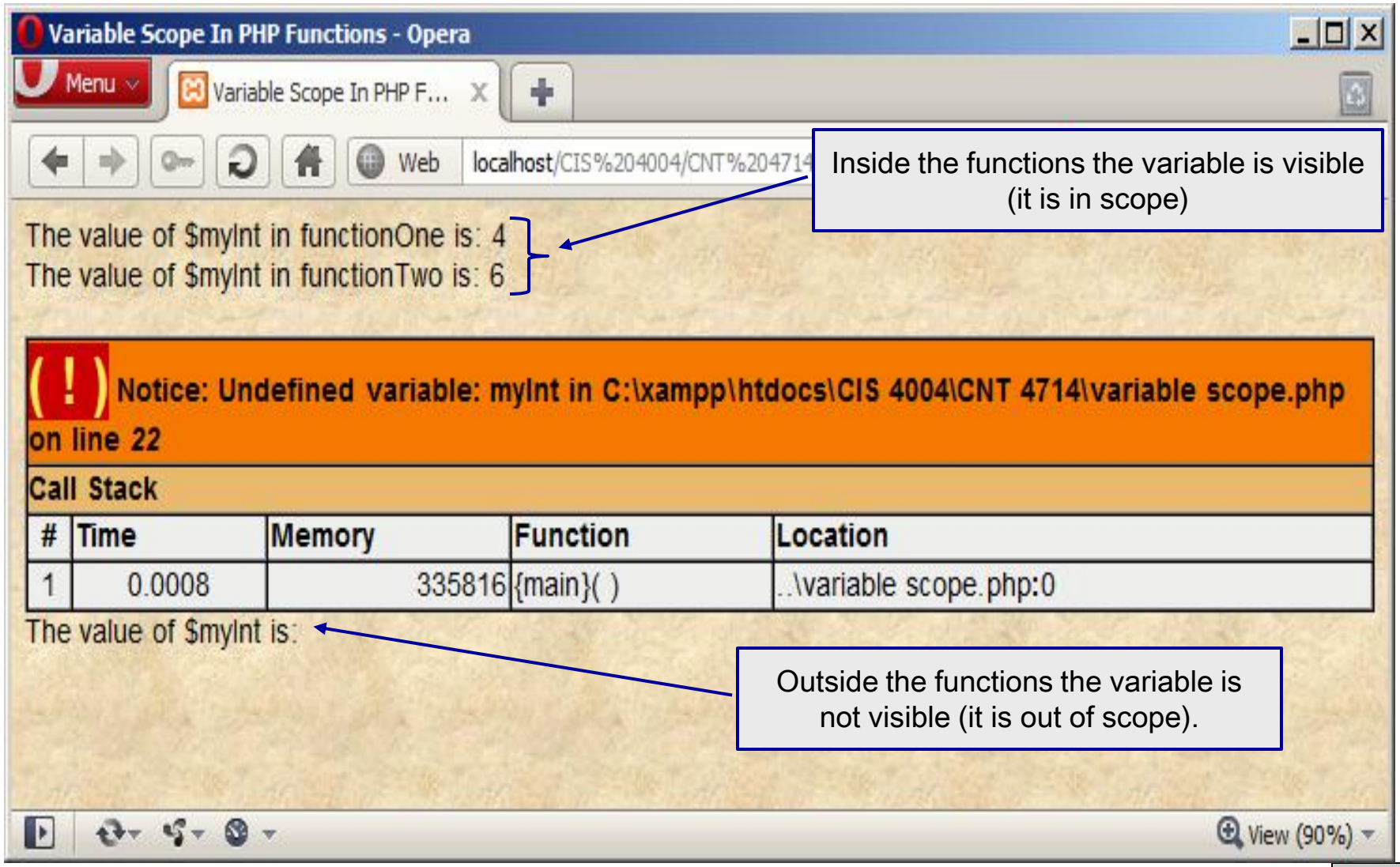
File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ? X

config.php | functionWithReturnValue.php | variable scope.php

```
2   <head>
3   <title>Variable Scope In PHP Functions</title>
4   </head>
5   <body  style = "font-family: arial, sans-serif;
6         background-color: #856363"  background=image1.jpg>
7   <?php
8       function functionOne($one) {
9           $myInt = $one;
10          echo 'The value of '. '$myInt in functionOne is: '. $myInt. "<br />";
11      } //end functionOne
12      function functionTwo($two) {
13          $myInt = $two;
14          echo 'The value of '. '$myInt in functionTwo is: '. $myInt. "<br />";
15      } //end functionTwo
16      $arg1 = 4;
17      $arg2 = 6;
18      functionOne($arg1);
19      functionTwo($arg2);
20      // echo 'The value of $one is: '. $one. "<br />";  //uncomment this line to
21      // echo 'The value of $two is: '. $two. "<br />";  //uncomment this line to
22      echo 'The value of $myInt is: '. $myInt. "<br />";
23  ?>
24  </body>
```

PHP | length : 803 lines : 25 | Ln : 24 Col : 8 Sel : 0 | Dos\Windows | ANSI | INS

Defining Functions In PHP



Variable Scope

- Similar to a variable defined inside a function having no scope outside of the function, a variable declared outside of a function is not accessible from inside the function.
- In general, if a function needs information from outside of the function in order to accomplish its task, the information should be passed as an argument to the function.
- Having said this however, there are times when you might want to access an important variable without passing it in as an argument. This is accomplished in PHP with the `global` statement. The **`global`** statement allows a function to access a variable declared outside of the function. More than one variable can be declared global at one time by separating the variable names with commas.
- The following example illustrates this concept using a variation of the previous example.

C:\xampp\htdocs\CIS 4004\CNT 4714\variable scope V2.php - Notepad++

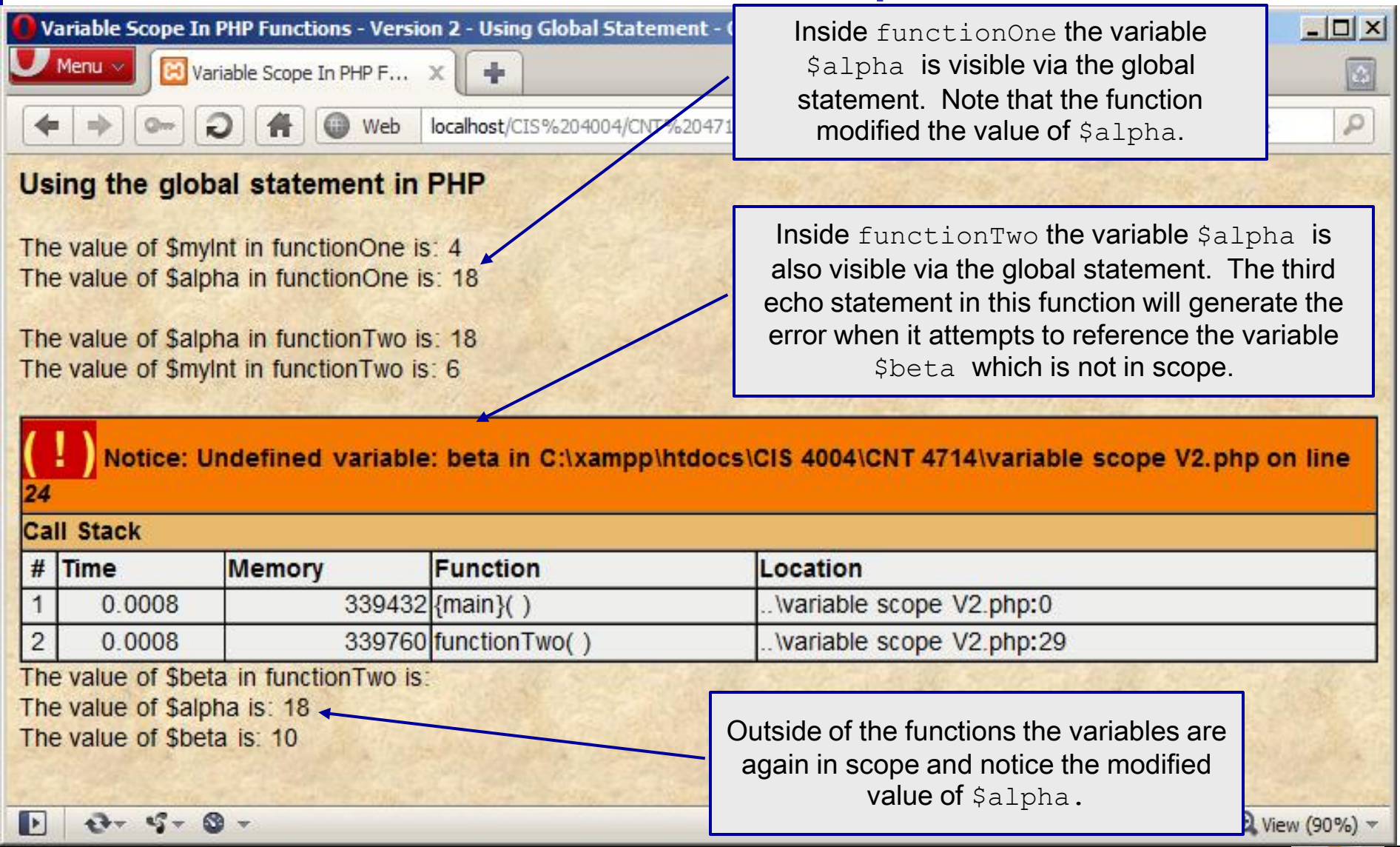
File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ?

config.php | functionWithReturnValue.php | variable scope V2.php

```php
<?php
    $alpha = 14;  //$alpha is declared outside of any functions
    $beta = 10; //$beta too is declared outside of any functions
    function functionOne($one) {
       global $alpha; //allow functionOne access to $alpha
       $myInt = $one;
       $alpha = $alpha + $myInt;
       echo 'The value of '. '$myInt in functionOne is: '. $myInt. "<br />";
       echo 'The value of '. '$alpha in functionOne is: '. $alpha. "<br /> <br /
    } //end functionOne
    function functionTwo($two) {
       global $alpha; //allow functionTwo access to $alpha
       $myInt = $two;
       echo 'The value of '. '$alpha in functionTwo is: '. $alpha. "<br />";
       echo 'The value of '. '$myInt in functionTwo is: '. $myInt. "<br />";
       // $beta is out of scope here - next line generates an error
       echo 'The value of '. '$beta in functionTwo is: '. $beta. "<br />";
    } //end functionTwo
    $arg1 = 4;
    $arg2 = 6;
    functionOne($arg1);
    functionTwo($arg2);
    //both $alpha and $beta are in scope here
    echo 'The value of $alpha is: '. $alpha. "<br />";  //note value of $alpha c
    echo 'The value of $beta is: '. $beta. "<br />";
?>
```

PHP length : 1420 lines : 35 | Ln : 16 Col : 82 Sel : 0 | Dos\Windows | ANSI | INS

Variable Scope



Inside `functionOne` the variable `$alpha` is visible via the global statement. Note that the function modified the value of `$alpha`.

Inside `functionTwo` the variable `$alpha` is also visible via the global statement. The third echo statement in this function will generate the error when it attempts to reference the variable `$beta` which is not in scope.

Outside of the functions the variables are again in scope and notice the modified value of `$alpha`.

Saving State Between Function Calls

- Local variables within functions have a short but happy life – they come into existence when the function is called and die when the execution of the function is finished.
- Occasionally, however, you might want to give a function a rudimentary memory.
- For example, suppose that you'd like a function to keep track of the number of times it has been called so that numbered headings can be created by a script.
- You could of course accomplish this by using the global statement and accessing a variable declared outside of the function. The example on the next page illustrates this technique.

C:\xampp\htdocs\CIS 4004\CNT 4714\saving state with global.php - Notepad++

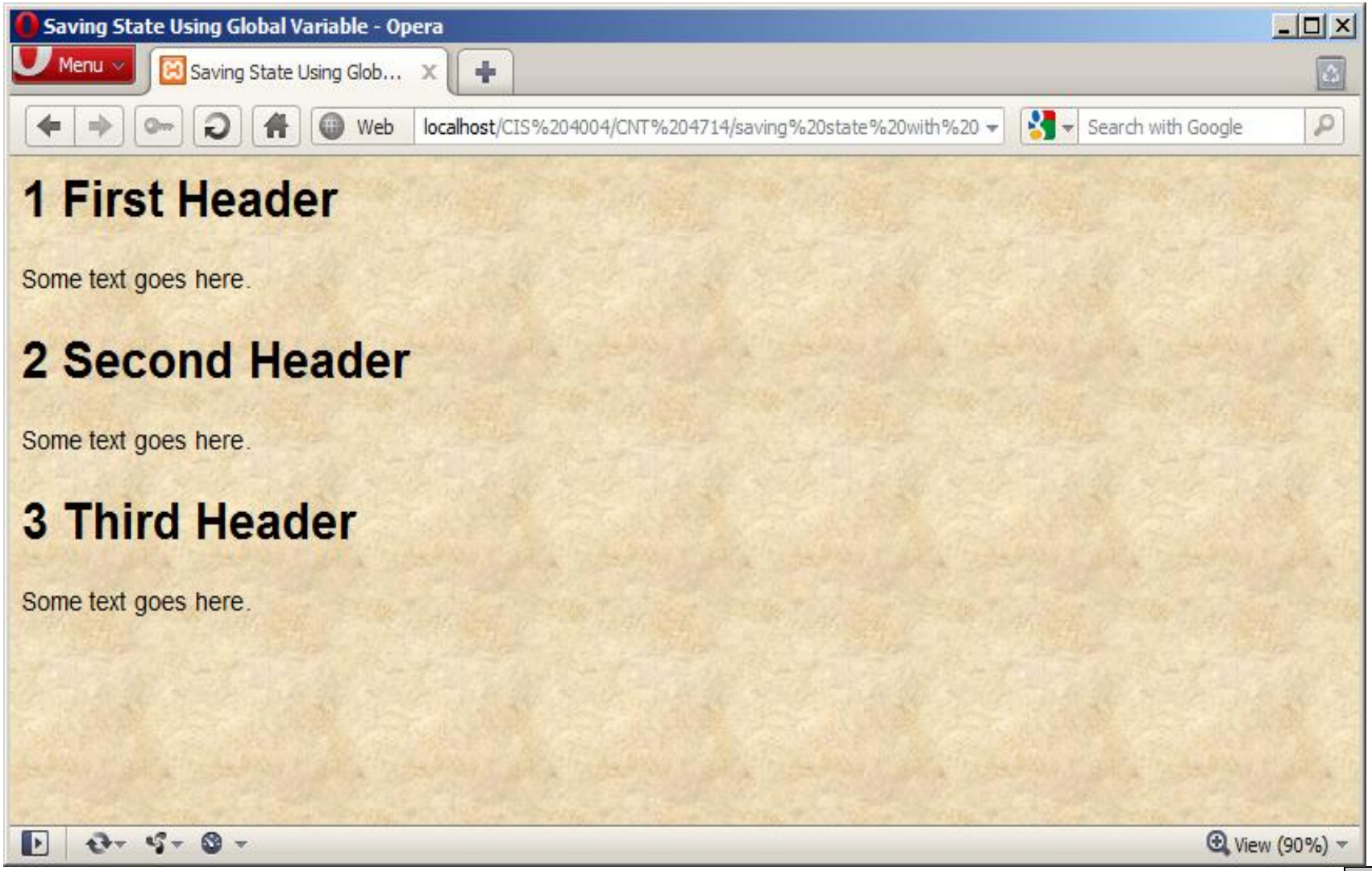
File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ?

config.php | functionWithReturnValue.php | saving state with global.php

```
1  <html>
2  <head>
3  <title>Saving State Using Global Variable</title>
4  </head>
5  <body  style = "font-family: arial, sans-serif;
6        background-color: #856363"  background=image1.jpg>
7  <?php
8      $number_of_calls = 0;  // number of times the function has been called
9      function numberedHeading($txt) {
10        global $number_of_calls; //allow function access to $number_of_calls
11        $number_of_calls++;
12        echo "<h1>".$number_of_calls." ".$txt."</h1>";
13     } //end numberedHeading
14     numberedHeading("First Header");
15     echo "<p> Some text goes here.</p>";
16     numberedHeading("Second Header");
17     echo "<p> Some text goes here.</p>";
18     numberedHeading("Third Header");
19     echo "<p> Some text goes here.</p>";
20 ?>
21 </body>
22 </html>
```

PHP | length : 722 lines : 22 | Ln : 22 Col : 8 Sel : 0 | Dos\Windows | ANSI | INS

Saving State Between Function Calls



Saving State Between Function Calls

- The previous example illustrated providing a function some "memory" through the use of a global variable.
- This is not a very elegant way to achieve this task. **Why?**
- **Answer:** Functions that use the global statement cannot be read as standalone blocks of code. In reading or reusing them, you must look out for the global variables that they manipulate. Failing to do so will render the function useless.
- This is where the `static` statement comes into play in PHP.
- Declaring a variable within a function to be static, the variable remains local to the function and the function remembers its value from execution to execution. The next example illustrates the `static` statement.

*C:\xampp\htdocs\CIS 4004\CNT 4714\saving state using static.php - Notepad++

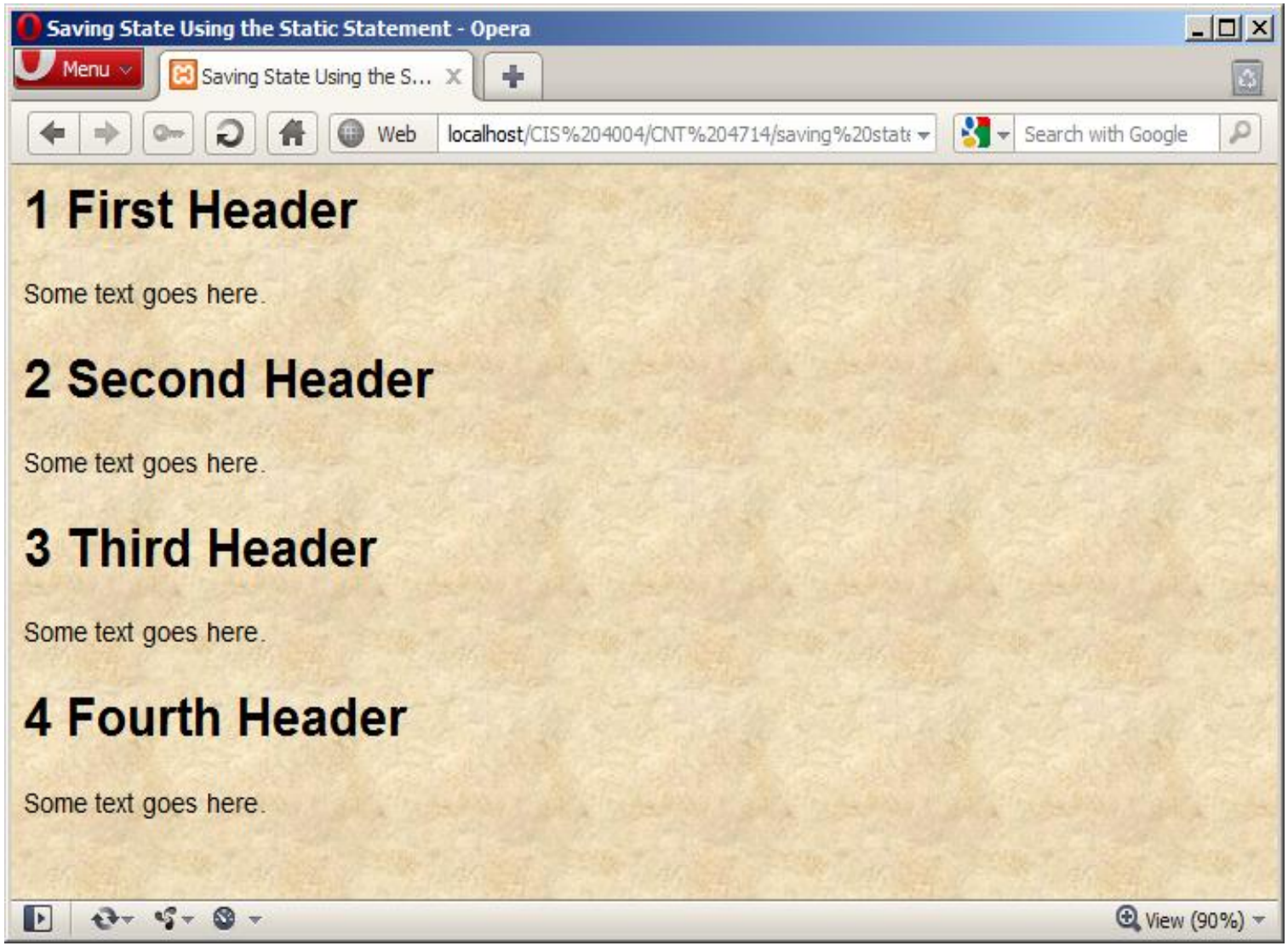
File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ? X

config.php | functionWithReturnValue.php | saving state using static.php

```
1  <html>
2  <head>
3  <title>Saving State Using the Static Statement</title>
4  </head>
5  <body  style = "font-family: arial, sans-serif;
6        background-color: #856363"  background=image1.jpg>
7  <?php
8      function numberedHeading($txt) {
9         static $number_of_calls = 0;  // number of times the function has been ca
10        $number_of_calls++;
11        echo "<h1>".$number_of_calls." ".$txt."</h1>";
12    } //end numberedHeading
13    numberedHeading("First Header");
14    echo "<p> Some text goes here.</p>";
15    numberedHeading("Second Header");
16    echo "<p> Some text goes here.</p>";
17    numberedHeading("Third Header");
18    echo "<p> Some text goes here.</p>";
19    numberedHeading("Fourth Header"); //added to differentiate from previous exa
20    echo "<p> Some text goes here.</p>";
21 ?>
22 </body>
```

PHP | length : 785 lines : 23 | Ln : 19 Col : 85 Sel : 0 | Dos\Windows | ANSI | INS

Saving State Between Function Calls



Setting Default Values For Arguments

- PHP provides a nifty feature to help you construct flexible functions. For functions that require one or more arguments, you can specify that some arguments are optional. This makes your functions more flexible.
- To illustrate the concept of the usefulness of setting default argument values, let's build a function that takes a string of text and an integer that corresponds to the point size in which the string is to be printed in the browser.
- This is shown on the next page.

C:\xampp\htdocs\CIS 4004\CNT 4714\default arguments V1.php - Notepad

File Edit Search View Encoding Language Settings Macro Run TextFX Plu

config.php | functionWithReturnValue.php | default arguments V1.php

```
1  <html>
2  <head>
3  <title>Default Argument Values - V1</title>
4  </head>
5  <body  style = "font-family: arial, sans-serif
6        background-color: #856363"  background=ir
7  <?php
8      function fontWrapper($txt, $fontsize) {
9          echo "<span style=\"font-size:$fontsize\" >".$txt."</span>";
10     } //end numberedHeading
11     fontWrapper("A Heading <br />", "24pt");
12     fontWrapper("Some text in the body <br />", "16pt");
13     fontWrapper("Some smaller text in the body <br />", "12pt");
14     fontWrapper("Some even smaller text in the body <br />", "8pt");
15 ?>
16 </body>
17 </html>
```

PHP | length : 571 lines : 17 | Ln : 17 Col : 8 Sel : 0 | Dos\Windows | ANSI | INS

Default Argument Values - V1 - Opera

Menu | Default Argument...

A Heading

Some text in the body

Some smaller text in the body

Some even smaller text in the body

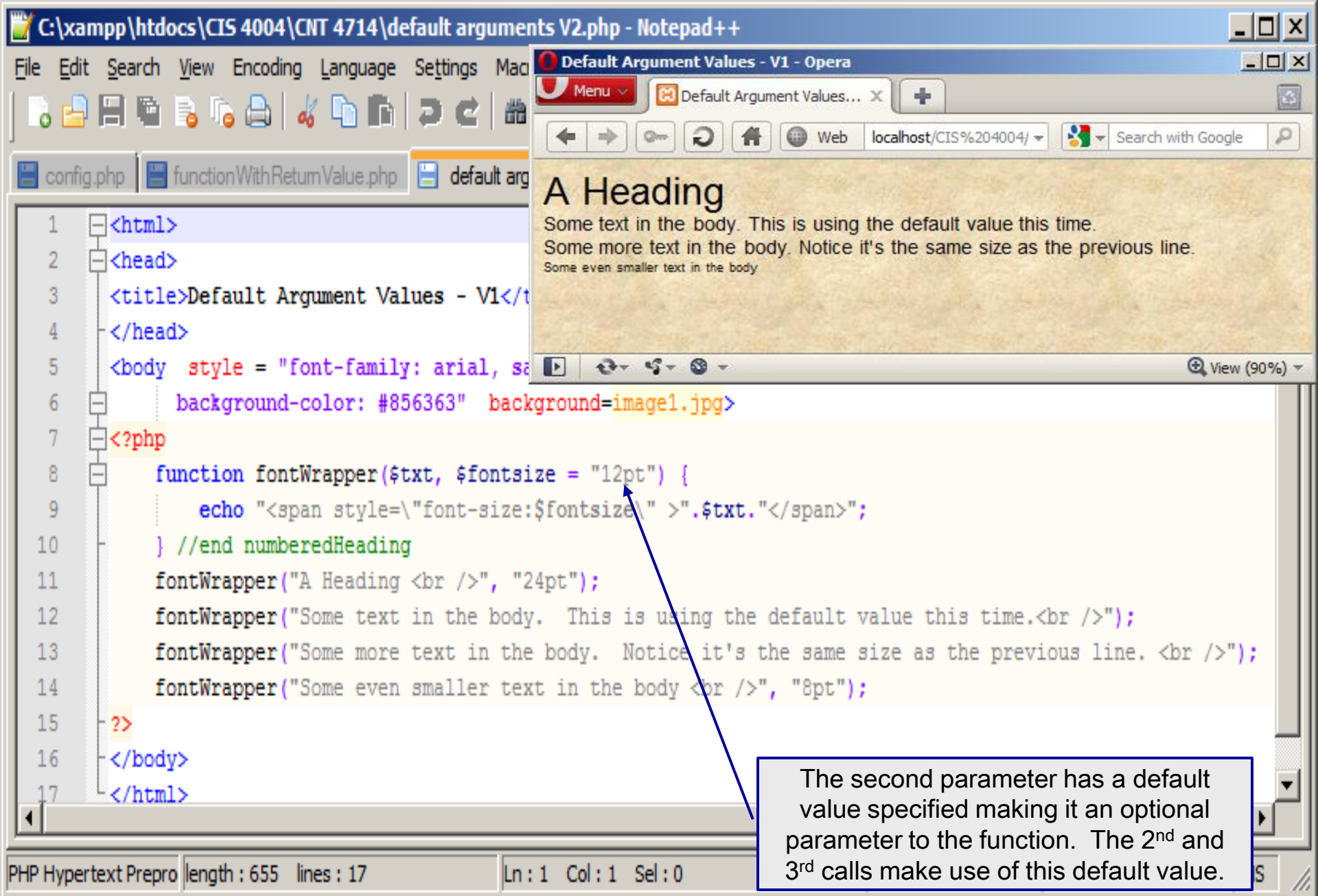
View (90%)

Setting Default Values For Arguments

- The nifty feature that PHP provides is to allow you to assign a value to a function argument within the function definition's parentheses.
- The effect of this is to make the argument value passed from the caller optional as the argument will assume the default value if no value is provided by the caller. The next example modifies the previous example to make use of this feature of PHP.

WARNING

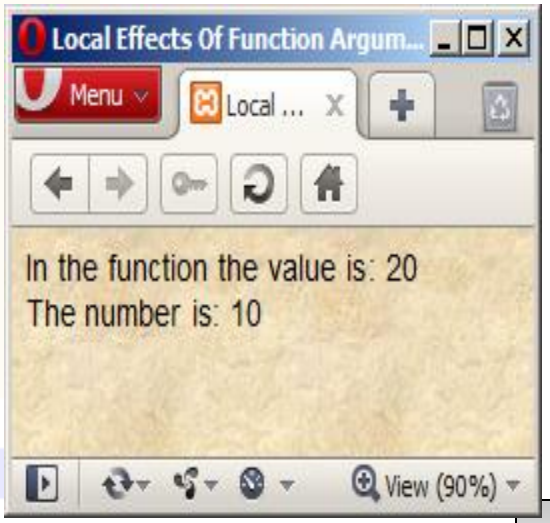
You can create as many optional arguments to a function as you wish. However, the arrangement of the arguments becomes important. Once an optional argument is defined in a function definition, all subsequent arguments must also be optional. In other words, you cannot have the first argument be optional, the second argument required, the third argument optional and so on. The ordering must be: all required arguments followed by all optional arguments.


C:\xampp\htdocs\CIS 4004\CNT 4714\default arguments V2.php - Notepad++
File Edit Search View Encoding Language Settings
config.php
functionWithReturnValue.php
default arg
1 <html>
2 <head>
3 <title>Default Argument Values - V1</t
4 </head>
5 <body style = "font-family: arial, sa
6 background-color: #856363" background=image1.jpg>
7 <?php
8 function fontWrapper($txt, $fontsize = "12pt") {
9 echo "<span style=\"font-size:$fontsize\" >".$txt."";
10 } //end numberedHeading
11 fontWrapper("A Heading
", "24pt");
12 fontWrapper("Some text in the body. This is using the default value this time.
");
13 fontWrapper("Some more text in the body. Notice it's the same size as the previous line.
");
14 fontWrapper("Some even smaller text in the body
", "8pt");
15 ?>
16 </body>
17 </html>
Default Argument Values - V1 - Opera
Menu
Default Argument Values...
Web
localhost/CIS%204004/
Search with Google
A Heading
Some text in the body. This is using the default value this time.
Some more text in the body. Notice it's the same size as the previous line.
Some even smaller text in the body
View (90%)
PHP Hypertext Prepro
length : 655 lines : 17
Ln : 1 Col : 1 Sel : 0
The second parameter has a default value specified making it an optional parameter to the function. The 2nd and 3rd calls make use of this default value.

Passing Variable References To Functions

- When you pass arguments to functions, they are stored as copies in parameter variables. This means that any changes made to these variables by the function is local to the function and are not reflected beyond it.

```
2   <head>
3   <title>Local Effects Of Function Arguments</title>
4   </head>
5   <body  style = "font-family: arial, sans-serif;
6         background-color: #856363"  background=image1.jpg>
7   <?php
8       function add10($num) {
9           $num += 10;
10          echo "In the function the value is: $num <br />";
11      } //end add10
12      $original_num = 10;
13      add10($original_num);
14      echo "The number is: $original_num <br/>";
15  ?>
16  </body>
17  </html>
```



Passing Variable References To Functions

- By default in PHP, variables passed to functions are passed by value. In other words, only local copies of the variables are used by the functions and the original values of the variables are not accessible by the function.
- So how can you allow a function to actually modify a variable sent to it? You must create a reference to the variable.
- The reference operator in PHP is the & (ampersand). Placing an ampersand in front of an argument in a function definition creates a reference to the variable and allows the function to modify the original variable.
- The following example modifies the previous example to make use of passing an argument by reference.

Passing Variable References To Functions

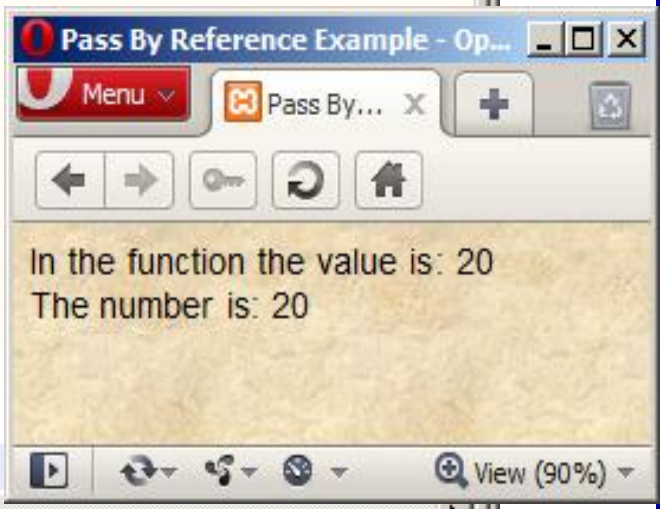
*C:\xampp\htdocs\CIS 4004\CNT 4714\pass by reference.php - Notepad++

File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ?

config.php | functionWithReturnValue.php | pass by reference.php

```
1  <html>
2  <head>
3  <title>Pass By Reference Example</title>
4  </head>
5  <body  style = "font-family: arial, sans-serif;
6        background-color: #856363"  background=image1.jpg>
7  <?php
8      function add10(&$num) {
9          $num += 10;
10         echo "In the function the value is: $num <br />";
11     } //end add10
12     $original_num = 10;
13     add10($original_num);
14     echo "The number is: $original_num <br/>";
15 ?>
16 </body>
17 </html>
```

The argument `$num` is passed by reference since it is preceded with the `&` operator.



PHP | length : 417 lines : 17 | Ln : 17 Col : 8 Sel : 0 | Dos\Windows | ANSI | INS