CIS 4004: Web-Based Information Technology Spring 2011

Introduction to PHP – Part 5 – Form Handling

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

Form Handling In PHP

- This set of notes will focus on handling forms in PHP. If you need a refresher on generating forms and the various XHTML elements that might appear in a form, I suggest you go back and review the XHTML notes Advanced XHTML & CSS – Tables And Forms that we covered earlier in the course.

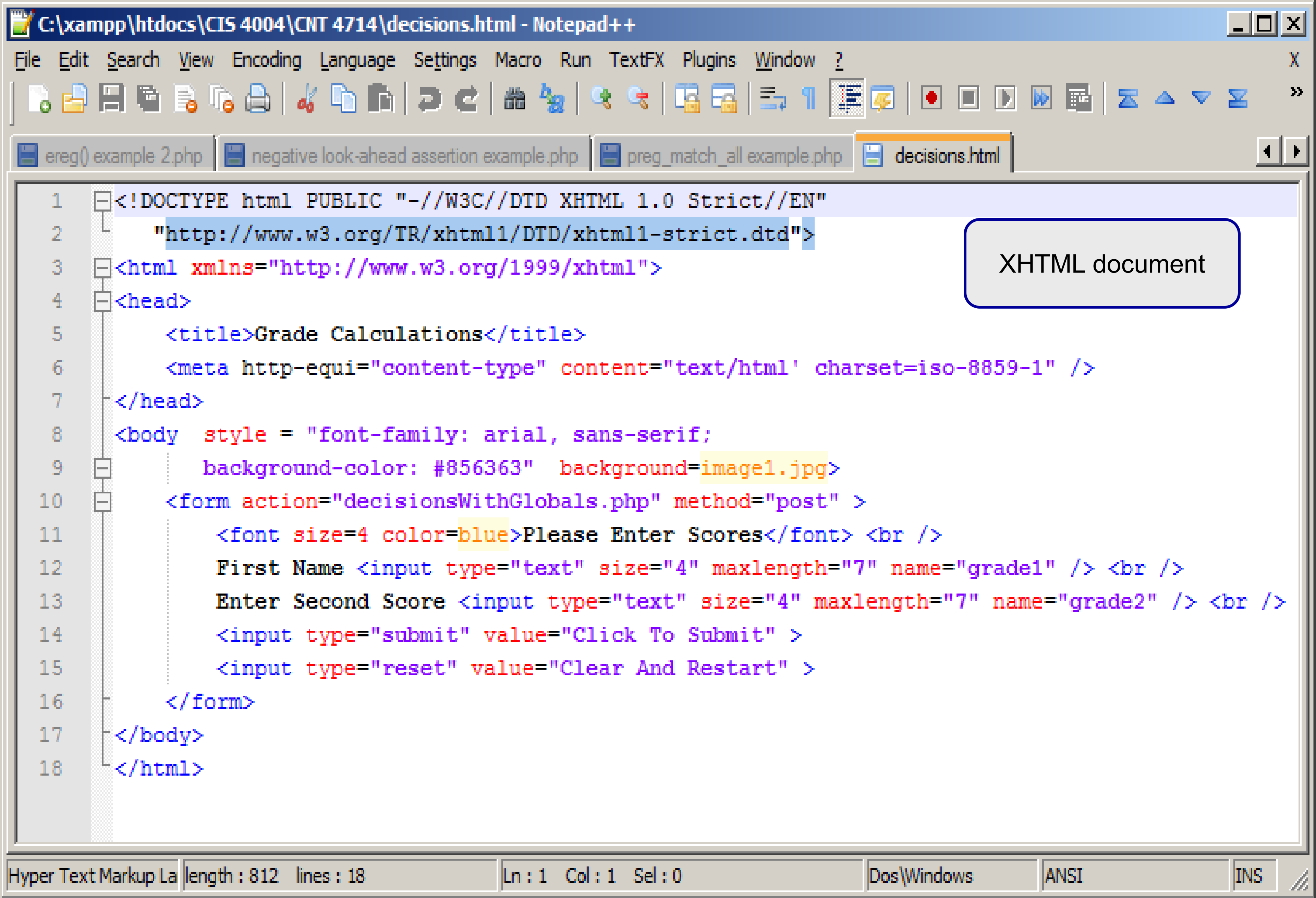
- What we want to do here is focus on the PHP side of things and not the XHTML side.

- Recall in the earlier set of notes that when we created a form, the action attribute simply caused an email to be sent to the address specified by the action attribute. What we want to do here though is cause a PHP script to be executed and the data from the form made available to the script.

Form Handling In PHP

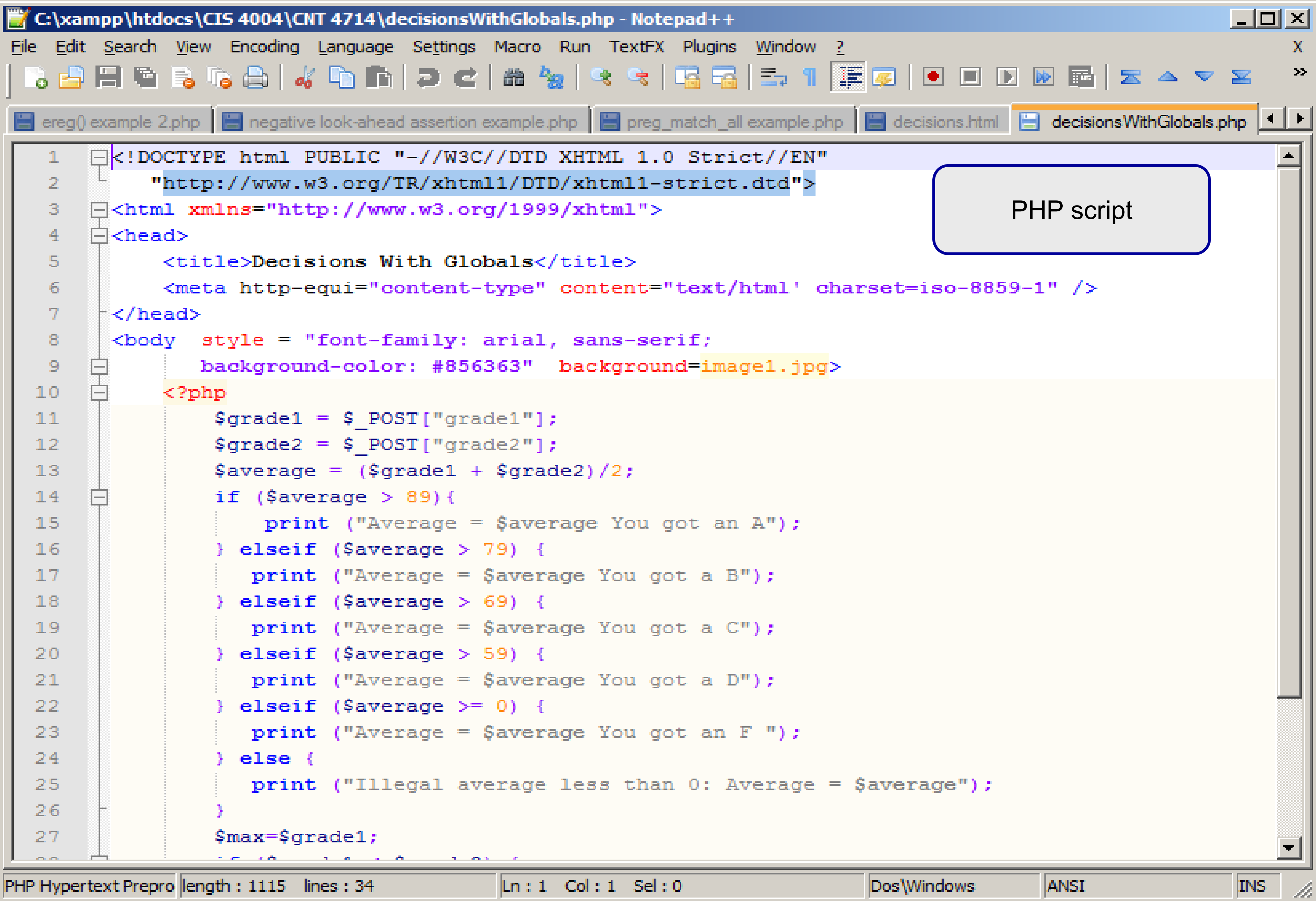
- Depending on what method was used to submit the form data to the PHP script (either GET or POST), PHP has two superglobal arrays, called `$_GET` and `$_POST`, that will be used to store the data contained in the form.
- `$_GET` and `$_POST` are associative arrays that contain, as key values, the names of the form elements as specified by their `name` attribute and their associated values as submitted by the user.
- You may recall that we saw an example of this in the introductory set of PHP notes. That example is duplicated on the next two pages to refresh your memory.

Note: The term superglobal refers to variables that are generated by the PHP and are always in scope. This means that regardless of where you are in a PHP script, whether it be inside a function or elsewhere, a superglobal variable is always available without needing to use the global statement to bring the variable into the current scope.


C:\xampp\htdocs\CIS 4004\CNT 4714\decisions.html - Notepad++
File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ?
ereg() example 2.php
negative look-ahead assertion example.php
preg_match_all example.php
decisions.html
Hyper Text Markup La
length : 812 lines : 18
Ln : 1 Col : 1 Sel : 0
Dos\Windows
ANSI
INS

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Grade Calculations</title>
    <meta http-equi="content-type" content="text/html' charset=iso-8859-1" />
</head>
<body  style = "font-family: arial, sans-serif;
       background-color: #856363"  background=image1.jpg>
    <form action="decisionsWithGlobals.php" method="post" >
        <font size=4 color=blue>Please Enter Scores</font> <br />
        First Name <input type="text" size="4" maxlength="7" name="grade1" /> <br />
        Enter Second Score <input type="text" size="4" maxlength="7" name="grade2" /> <br />
        <input type="submit" value="Click To Submit" >
        <input type="reset" value="Clear And Restart" >
    </form>
</body>
</html>
```

XHTML document



```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Decisions With Globals</title>
    <meta http-equi="content-type" content="text/html' charset=iso-8859-1" />
</head>
<body  style = "font-family: arial, sans-serif;
       background-color: #856363"  background=image1.jpg>
    <?php
        $grade1 = $_POST["grade1"];
        $grade2 = $_POST["grade2"];
        $average = ($grade1 + $grade2)/2;
        if ($average > 89){
            print ("Average = $average You got an A");
        } elseif ($average > 79) {
           print ("Average = $average You got a B");
        } elseif ($average > 69) {
           print ("Average = $average You got a C");
        } elseif ($average > 59) {
           print ("Average = $average You got a D");
        } elseif ($average >= 0) {
           print ("Average = $average You got an F ");
        } else {
           print ("Illegal average less than 0: Average = $average");
        }
        $max=$grade1;
```

PHP script

Form Handling In PHP

- For reasons of internationalization, `get` has been deprecated as a value for the `method` attribute in XHTML `form` elements. So we will focus exclusively on the `post` value for the `method` attribute and thus the `$_POST` superglobal array.

- In earlier versions of PHP (those prior to 4.1.0), PHP created, by default, standard variable names to represent the values contain within the superglobal arrays. Current versions, by default, do not enable this property for security reasons. It is still possible to turn this behavior on by setting the `register_globals` directive to on in the PHP configuration file (`php.ini`), it should be discouraged.

Form Handling In PHP

- A safer approach to this is to use the import_request_variables() function. This function will create global-scope variables for the values stored in the relevant superglobal arrays.

- The syntax for this function is:

```
import_request_variables($types [, $prefix])
```

- Where `$types` represents a string indicating the types of variables to import and should consist of any combination (not case sensitive) of the letters `P`, `G`, and `C`. These letters represent `$_POST`, `$_GET`, and `$_COOKIE`, respectively. The second optional parameter, `$prefix`, if provided, should be a string representing what to prefix to the start of every variable created.

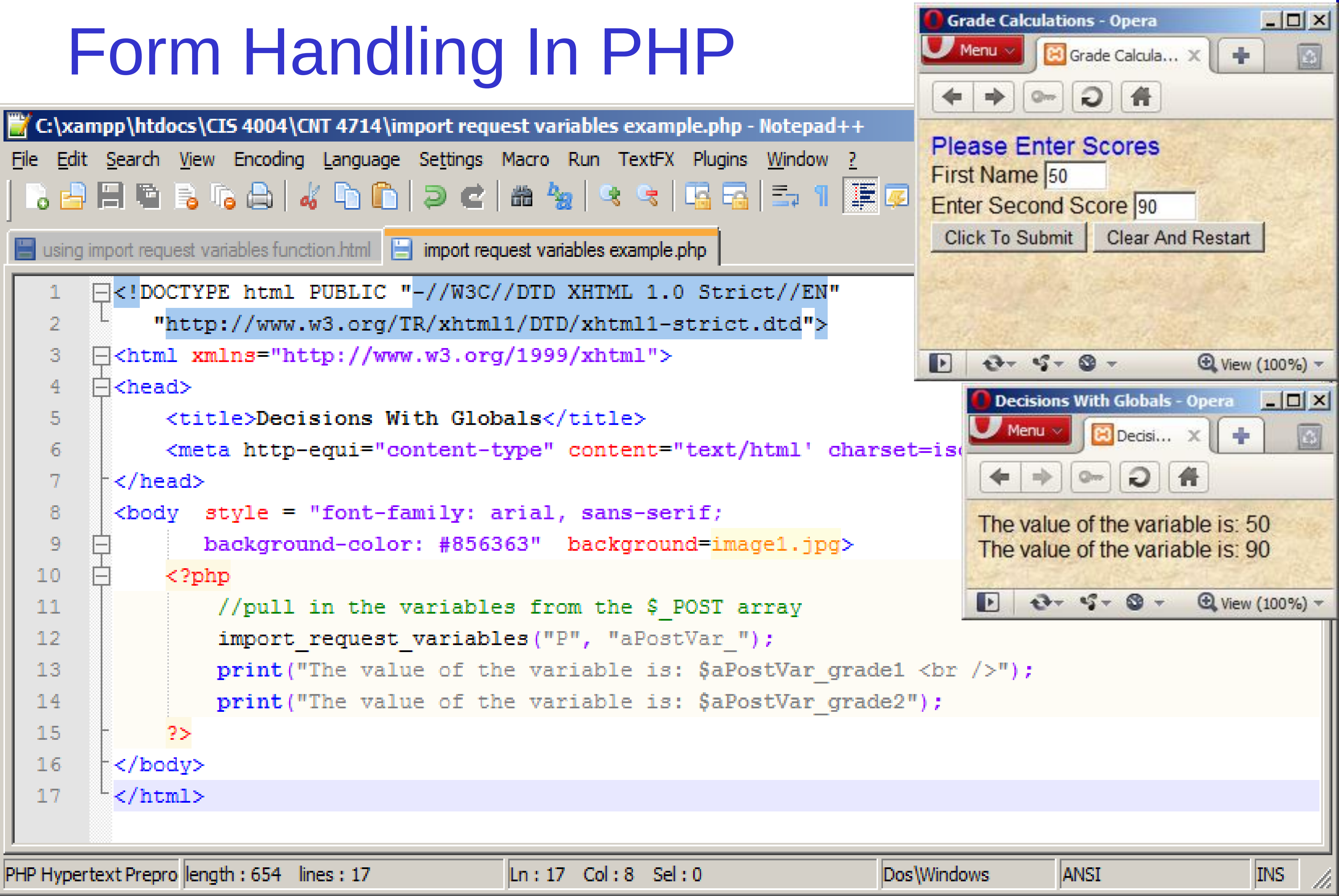
Form Handling In PHP

- Note, that while this function is safer than using the `register_globals` configuration directive, it does not protect you from the security risks associated with working with user data. It is always recommended that data received from the user be sanitized before use.

- It is also important to note that the `import_request_variables()` function imports variables into the global scope only. Therefore, it should never be used from within a function.

- An example of the `import_request_variables()` function is shown on the next page.

Form Handling In PHP



Grade Calculations - Opera
Menu
Grade Calcula...
View (100%)

Please Enter Scores

First Name 50

Enter Second Score 90

Click To Submit
Clear And Restart

C:\xampp\htdocs\CIS 4004\CNT 4714\import request variables example.php - Notepad++
File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ?
using import request variables function.html
import request variables example.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Decisions With Globals</title>
    <meta http-equi="content-type" content="text/html' charset=is
</head>
<body  style = "font-family: arial, sans-serif;
       background-color: #856363"  background=image1.jpg>
    <?php
        //pull in the variables from the $_POST array
        import_request_variables("P", "aPostVar_");
        print("The value of the variable is: $aPostVar_grade1 <br />");
        print("The value of the variable is: $aPostVar_grade2");
    ?>
</body>
</html>
```

```
The value of the variable is: 50
The value of the variable is: 90
```

Securing Hidden Elements

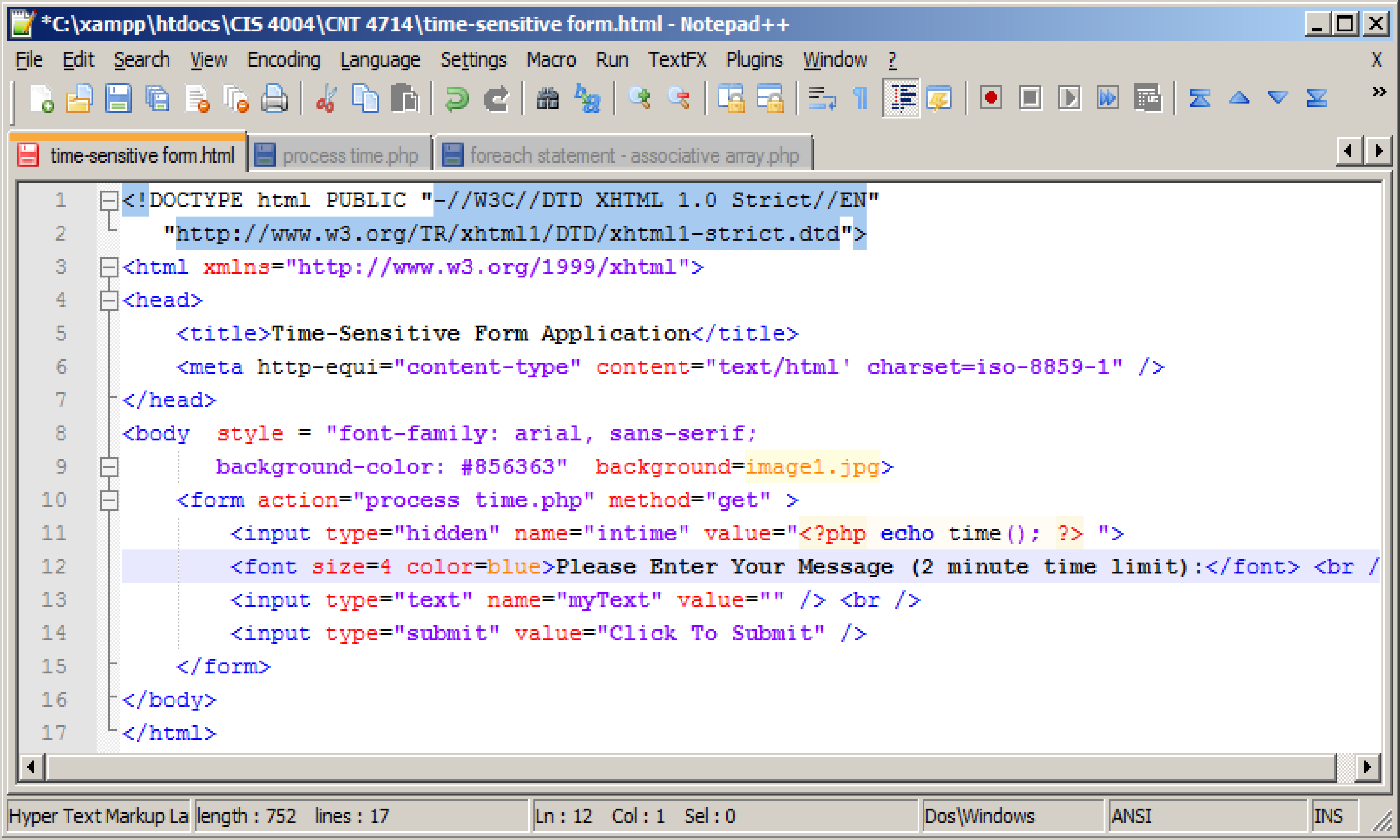
- Take a look at the following example (which does use a GET method value).

- This script creates a time-sensitive form. If the user does not click the submit button within 2 minutes of starting the form in their browser, the form will be considered invalid.

- Basically, a hidden field is used to store the time when the form is first displayed to the user, when the user clicks the submit button to submit the form, the PHP script checks the current time and determines if more than 2 minutes has elapsed. If so the a message is displayed indicating that the form is invalid.

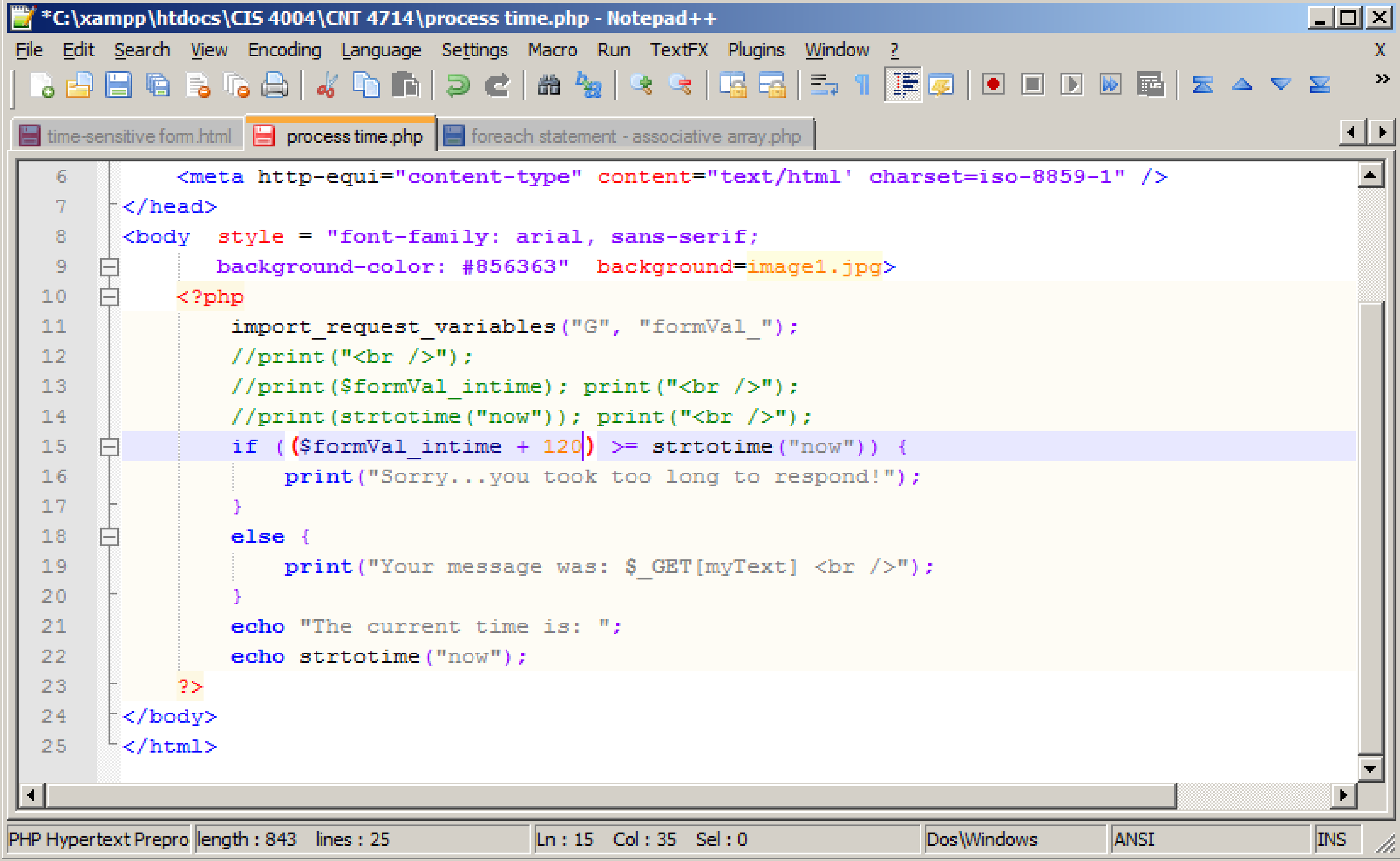
- The XHTML document is on page 11 and the PHP script is on page 12.

Securing Hidden Elements


*C:\xampp\htdocs\CIS 4004\CNT 4714\time-sensitive form.html - Notepad++
File
Edit
Search
View
Encoding
Language
Settings
Macro
Run
TextFX
Plugins
Window
?
X
time-sensitive form.html
process time.php
foreach statement - associative array.php
Hyper Text Markup La
length : 752   lines : 17
Ln : 12   Col : 1   Sel : 0
Dos\Windows
ANSI
INS

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Time-Sensitive Form Application</title>
    <meta http-equi="content-type" content="text/html' charset=iso-8859-1" />
</head>
<body  style = "font-family: arial, sans-serif;
       background-color: #856363"  background=image1.jpg>
    <form action="process time.php" method="get" >
        <input type="hidden" name="intime" value="<?php echo time(); ?> ">
        <font size=4 color=blue>Please Enter Your Message (2 minute time limit):</font> <br /
        <input type="text" name="myText" value="" /> <br />
        <input type="submit" value="Click To Submit" />
    </form>
</body>
</html>
```

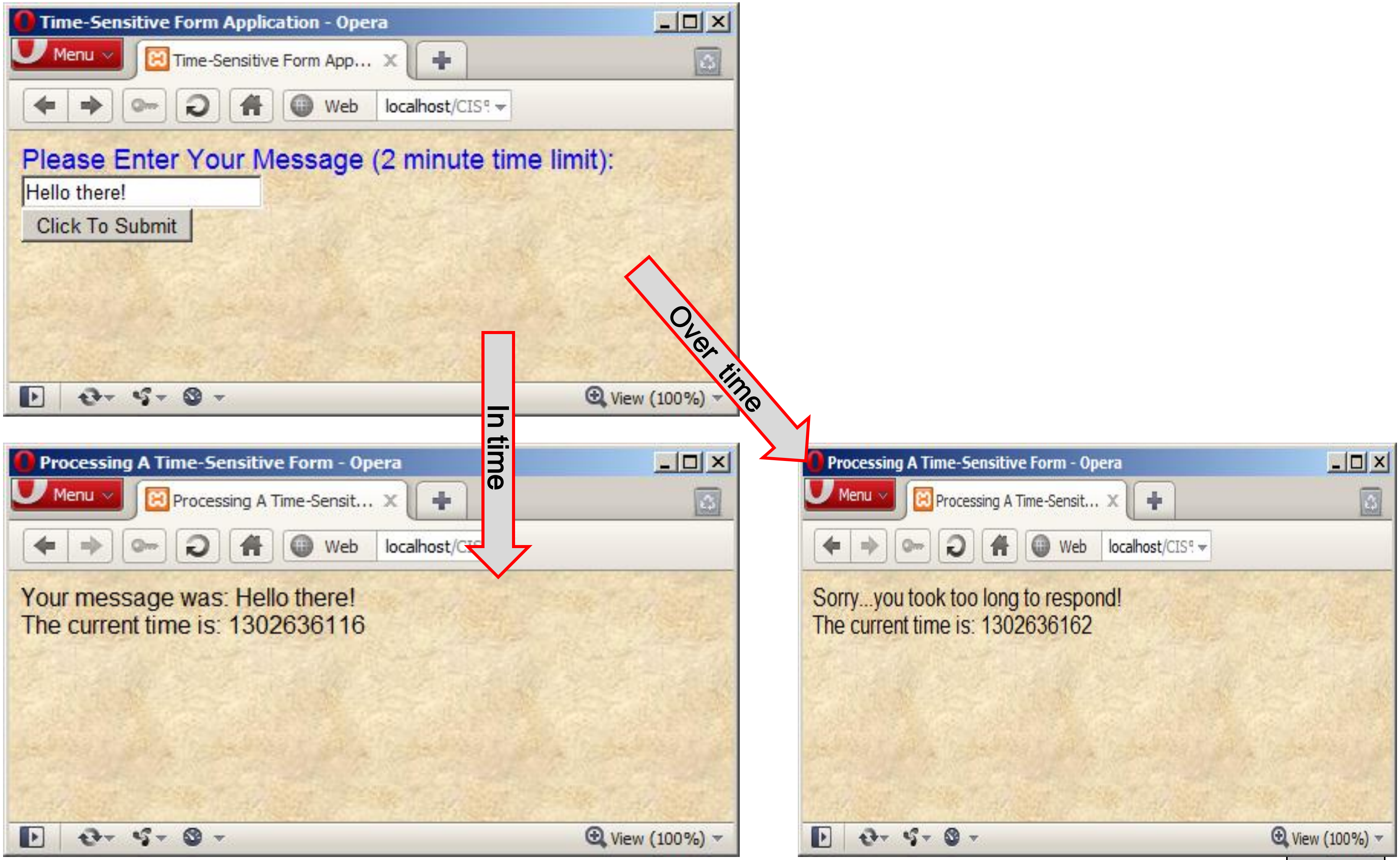
Securing Hidden Elements



```
    <meta http-equi="content-type" content="text/html' charset=iso-8859-1" />
</head>
<body  style = "font-family: arial, sans-serif;
       background-color: #856363"  background=image1.jpg>
    <?php
        import_request_variables("G", "formVal_");
        //print("<br />");
        //print($formVal_intime); print("<br />");
        //print(strtotime("now")); print("<br />");
        if ( ($formVal_intime + 120) >= strtotime("now")) {
            print("Sorry...you took too long to respond!");
        }
        else {
            print("Your message was: $_GET[myText] <br />");
        }
        echo "The current time is: ";
        echo strtotime("now");
    ?>
</body>
</html>
```

PHP Hypertext Prepro | length : 843 lines : 25 | Ln : 15 Col : 35 Sel : 0 | Dos\Windows | ANSI | INS

Securing Hidden Elements


Time-Sensitive Form Application - Opera
Menu
Time-Sensitive Form App...
Web
localhost/CIS
View (100%)
Processing A Time-Sensitive Form - Opera
Processing A Time-Sensit...
In time
Over time

Please Enter Your Message (2 minute time limit):

Hello there!

Click To Submit

Your message was: Hello there!
The current time is: 1302636116

Sorry...you took too long to respond!
The current time is: 1302636162

Securing Hidden Elements

- One of the reasons the get method value is a potential security problem is that it is possible for a somewhat savvy user of your site to send a false time value via the query string that is appended to the URL which invokes the PHP script.

- Notice on the previous page that I printed the time value out. This time value is appended to the URL (called a query string) and the URL would look like:

```
http:// … process time.php?time=1302636166
```

- For the case where the user waited too long to submit, what they could have done was submitted a URL like the one below which explicitly sets the time to be only 20s greater than the time in the query string:

```
http:// … process time.php?time=1302636186
```

Securing Hidden Elements

- While the previous example used the `$_GET` superglobal and illustrated the vulnerability of the method, fear not, for there are better ways to secure the hidden elements of your forms and protect them from a malicious user.

- The secret to data validation in this case is the MD5 algorithm. This algorithm is used to create a message digest (a sort of digital fingerprint) of the data provided to it.

- MD5 stands for Message-Digest algorithm 5. This is a widely used cryptographic hash function that uses a 128-bit (16-byte) hash value. MD5 was “cryptographically broken” in 2007 and is no longer considered safe for applications like SSL and digital signatures, but it will still suffice for our application here.

Securing Hidden Elements

- Like the fingerprints found on a person, the digital fingerprint generated by the MD5 algorithm is unique to the string that it represents.
- The MD5 algorithm is not collision resistant, so there is a small chance (1 in 3.40x10^{38}) that two strings will produce an identical fingerprint, for all practical purposes they will be unique. However, it is a predictable algorithm in that the same string always produces the same fingerprint.
- In PHP the MD5 algorithm is implemented by the `md5()` function, which has the following syntax:

```
md5($string);
```

- Where `$string` represents the string to generate the fingerprint for. The function returns a 32-character fingerprint based on the data in `$string`.

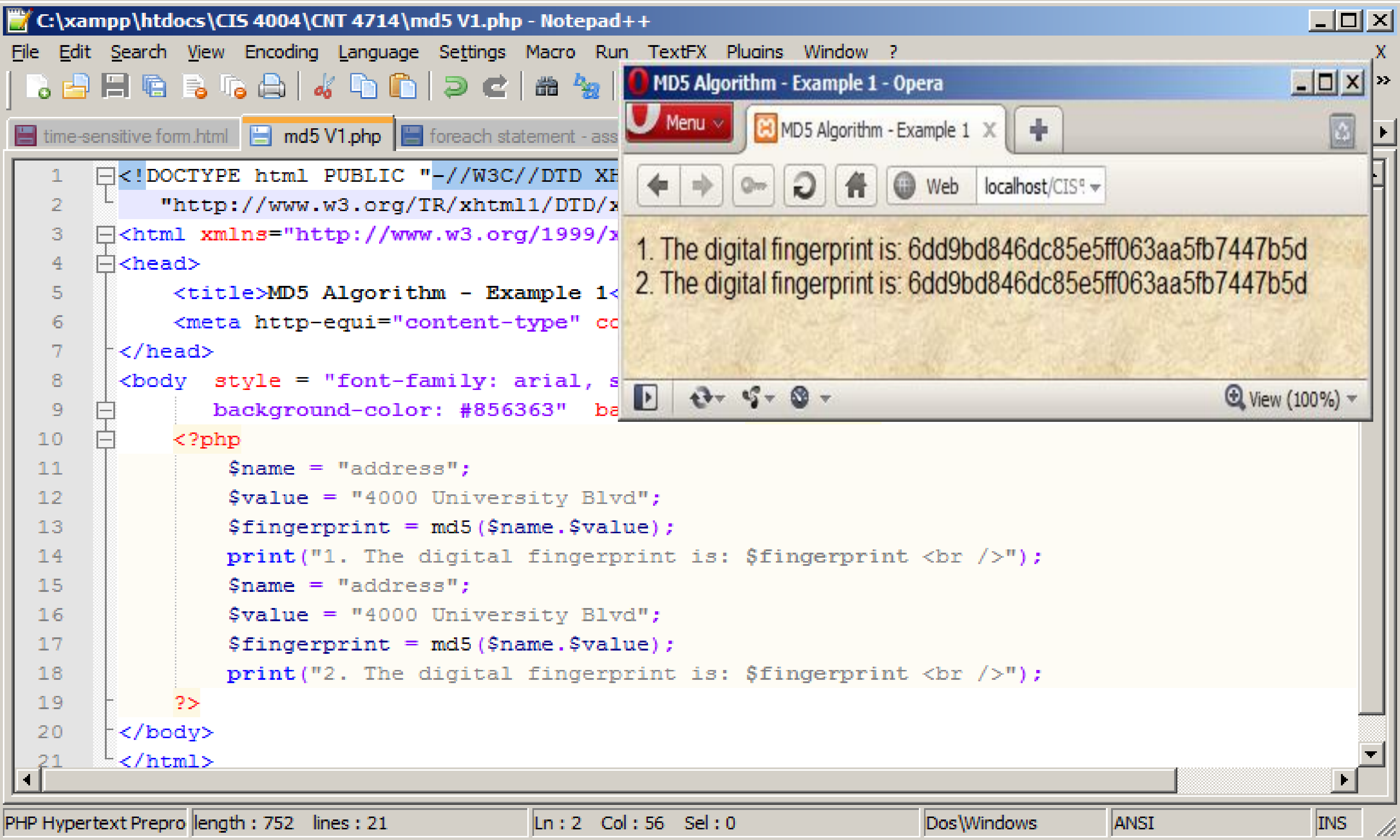
Securing Hidden Elements

- How does the MD5 algorithm help in this case? By creating digital fingerprint values for each hidden form element in your document and then checking those fingerprint values when the form is submitted, you can be confident that the data submitted was actually valid.

- When creating an MD5 fingerprint for this sort of purpose, its important to remember that one of the major benefits of this type of algorithm can also be its downfall. Because the algorithm is completely predictable, simply using some combination of the provided `$name` and `$value` parameters could be hazardous.

- The example on the following page illustrates this, with two calls to the `md5()` function using the same string, it produces the same digital fingerprint.

Securing Hidden Elements


C:\xampp\htdocs\CIS 4004\CNT 4714\md5 V1.php - Notepad++
File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ?
time-sensitive form.html
md5 V1.php
foreach statement - ass
MD5 Algorithm - Example 1 - Opera
Menu
MD5 Algorithm - Example 1
Web
localhost/CIS
View (100%)
PHP Hypertext Prepro
length : 752 lines : 21
Ln : 2 Col : 56 Sel : 0
Dos\Windows
ANSI
INS

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XH
   "http://www.w3.org/TR/xhtml1/DTD/x
<html xmlns="http://www.w3.org/1999/x
<head>
    <title>MD5 Algorithm - Example 1<
    <meta http-equi="content-type" co
</head>
<body  style = "font-family: arial, s
       background-color: #856363"  ba
    <?php
        $name = "address";
        $value = "4000 University Blvd";
        $fingerprint = md5($name.$value);
        print("1. The digital fingerprint is: $fingerprint <br />");
        $name = "address";
        $value = "4000 University Blvd";
        $fingerprint = md5($name.$value);
        print("2. The digital fingerprint is: $fingerprint <br />");
    ?>
</body>
</html>
```

```
1. The digital fingerprint is: 6dd9bd846dc85e5ff063aa5fb7447b5d
2. The digital fingerprint is: 6dd9bd846dc85e5ff063aa5fb7447b5d
```

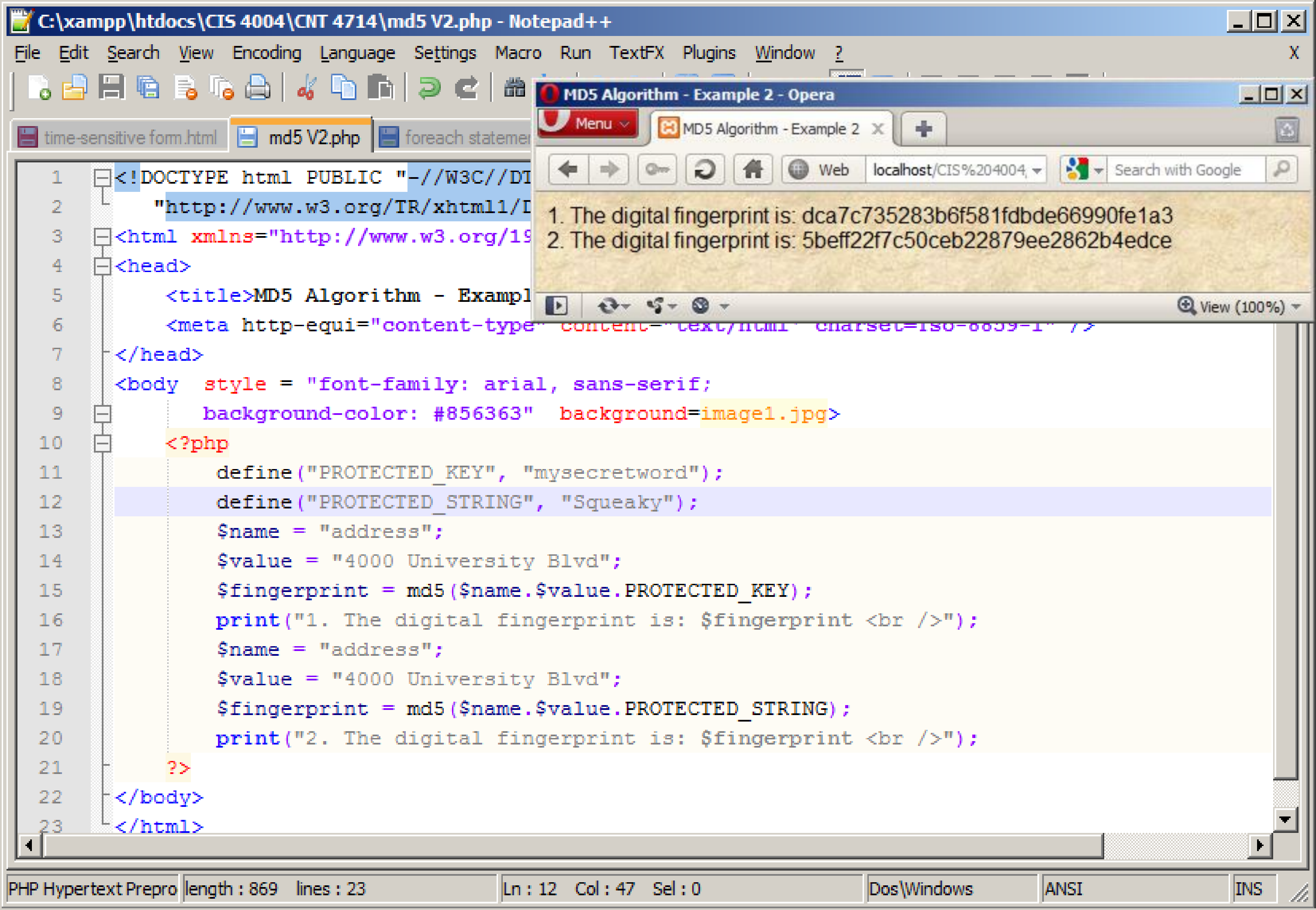
Securing Hidden Elements

- For the MD5 fingerprint to be truly unique, a value completely unknown to the outside user must be included in the creation of the digital fingerprint.

- The following example illustrates this, by again creating a digital fingerprint for the same string twice, but each time with a different additional string appended to the string.

- The additional string would be defined using a constant in PHP let's call one of them `PROTECTED_KEY`, and the other one `PROTECTED_STRING.` Recall that constants are created using the PHP `define` statement.

- Notice now, that the two identical strings produce different digital fingerprints.

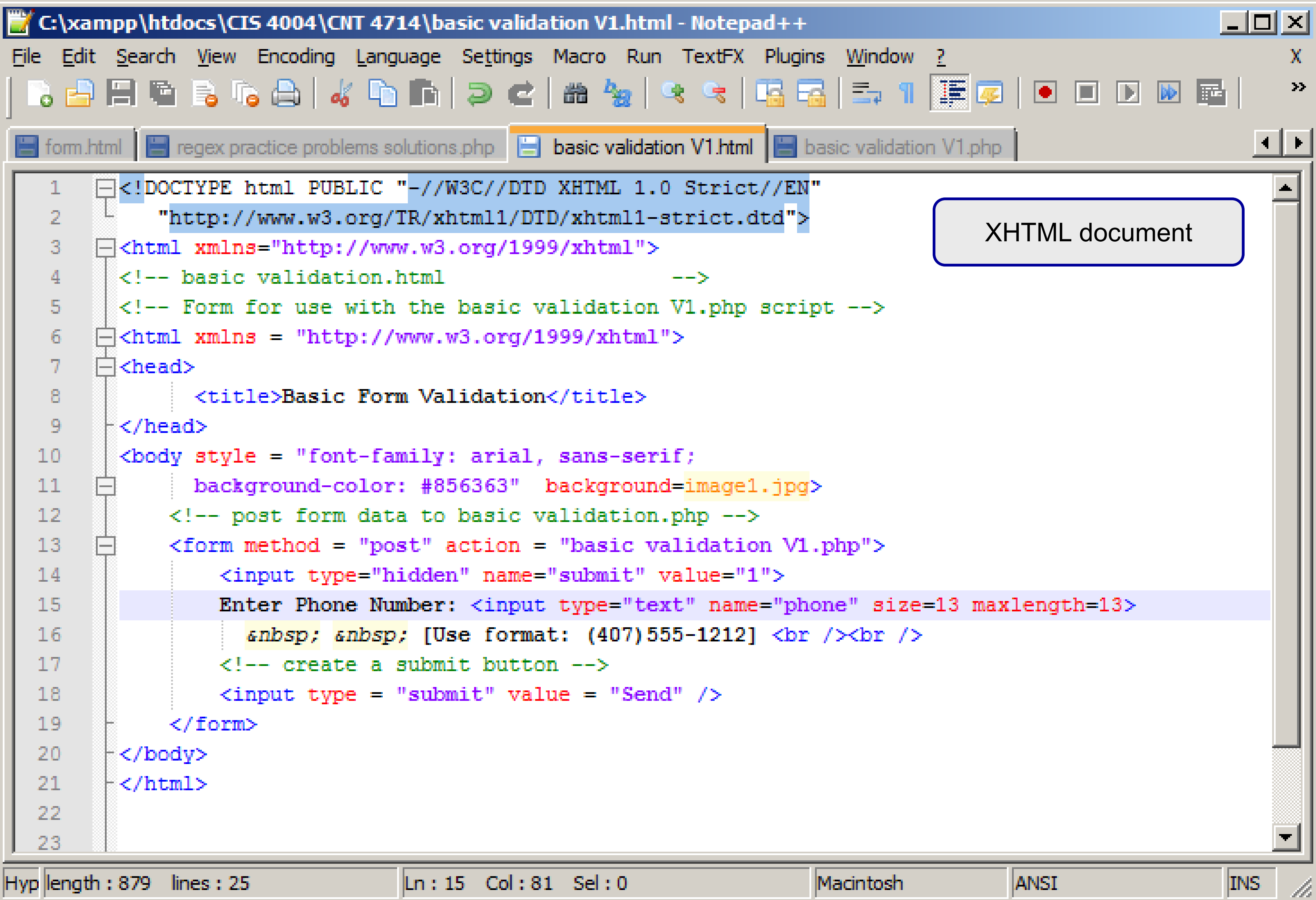

C:\xampp\htdocs\CIS 4004\CNT 4714\md5 V2.php - Notepad++
File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ?
time-sensitive form.html
md5 V2.php
foreach statemer
MD5 Algorithm - Example 2 - Opera
Menu
MD5 Algorithm - Example 2
Web
localhost/CIS%204004
Search with Google
View (100%)
PHP Hypertext Prepro
length : 869 lines : 23
Ln : 12 Col : 47 Sel : 0
Dos\Windows
ANSI
INS

```
<!DOCTYPE html PUBLIC "-//W3C//DT
    "http://www.w3.org/TR/xhtml1/D
<html xmlns="http://www.w3.org/19
<head>
    <title>MD5 Algorithm - Exampl
    <meta http-equi="content-type" content="text/html" charset=iso-8859-1" />
</head>
<body  style = "font-family: arial, sans-serif;
       background-color: #856363"  background=image1.jpg>
    <?php
        define("PROTECTED_KEY", "mysecretword");
        define("PROTECTED_STRING", "Squeaky");
        $name = "address";
        $value = "4000 University Blvd";
        $fingerprint = md5($name.$value.PROTECTED_KEY);
        print("1. The digital fingerprint is: $fingerprint <br />");
        $name = "address";
        $value = "4000 University Blvd";
        $fingerprint = md5($name.$value.PROTECTED_STRING);
        print("2. The digital fingerprint is: $fingerprint <br />");
    ?>
</body>
</html>
```

```
1. The digital fingerprint is: dca7c735283b6f581fdbde66990fe1a3
2. The digital fingerprint is: 5beff22f7c50ceb22879ee2862b4edce
```

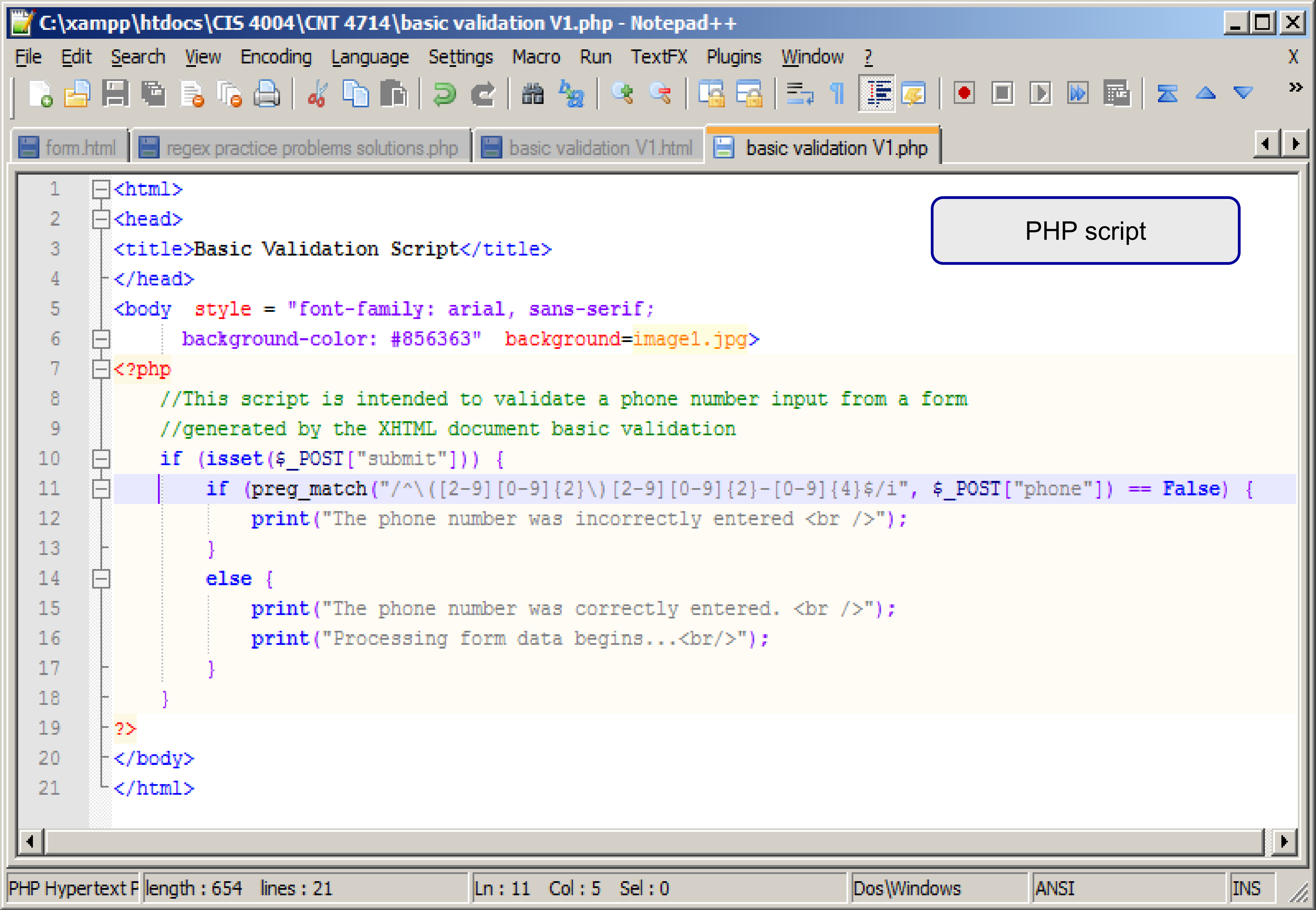
Basic Form Processing And Validation

- In the simplest sense, form validation and processing is nothing more than working with the appropriate superglobal array (`$_GET` or `$_POST`) to do something in your PHP script.
- However, for a form of any complexity, often a considerable amount of effort goes into the validation of the data coming in from the form.
- For anything beyond the most elementary validation, usually all form validation is done via regular expressions.
- The example on the next page illustrates basic form validation using a regular expression.


C:\xampp\htdocs\CIS 4004\CNT 4714\basic validation V1.html - Notepad++
File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ?
form.html
regex practice problems solutions.php
basic validation V1.html
basic validation V1.php
Hyp length : 879 lines : 25
Ln : 15 Col : 81 Sel : 0
Macintosh
ANSI
INS

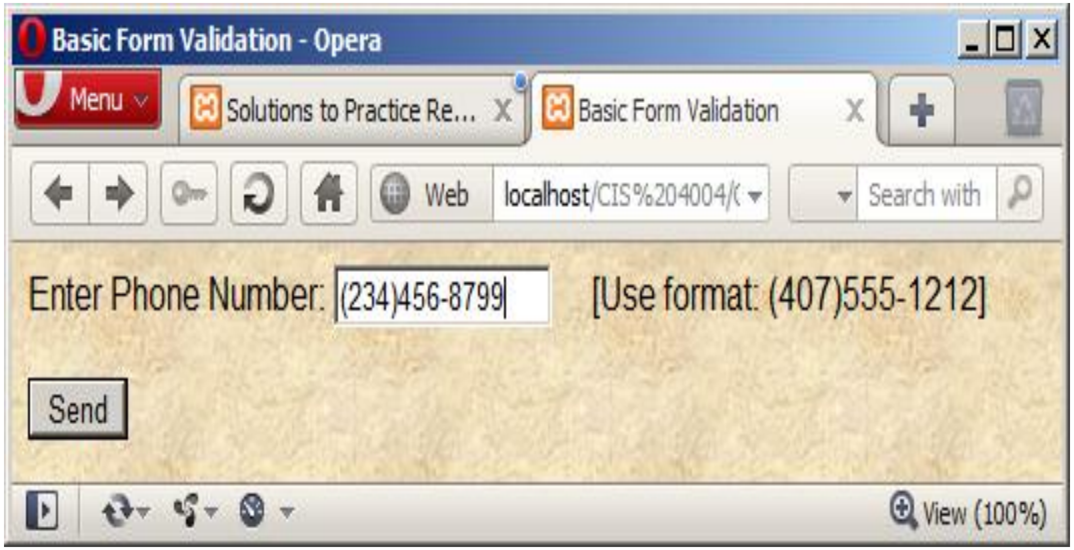
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!-- basic validation.html                  -->
<!-- Form for use with the basic validation V1.php script -->
<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
      <title>Basic Form Validation</title>
</head>
<body style = "font-family: arial, sans-serif;
      background-color: #856363"  background=image1.jpg>
    <!-- post form data to basic validation.php -->
    <form method = "post" action = "basic validation V1.php">
        <input type="hidden" name="submit" value="1">
        Enter Phone Number: <input type="text" name="phone" size=13 maxlength=13>
          &nbsp; &nbsp; [Use format: (407)555-1212] <br /><br />
        <!-- create a submit button -->
        <input type = "submit" value = "Send" />
    </form>
</body>
</html>
```

XHTML document


C:\xampp\htdocs\CIS 4004\CNT 4714\basic validation V1.php - Notepad++
File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ?
form.html
regex practice problems solutions.php
basic validation V1.html
basic validation V1.php
PHP Hypertext P length : 654 lines : 21
Ln : 11 Col : 5 Sel : 0
Dos\Windows
ANSI
INS

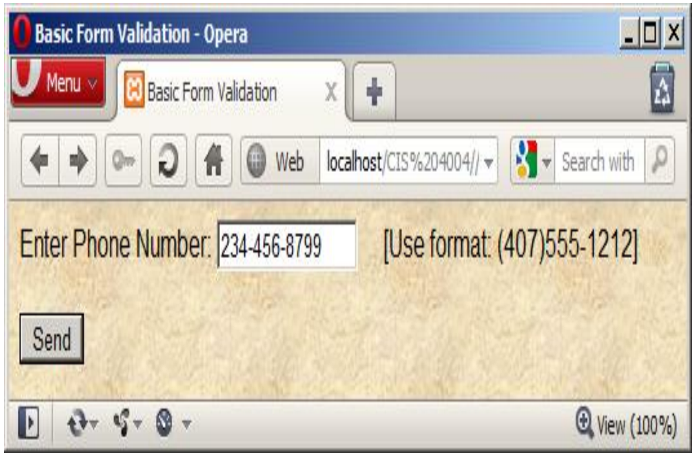
PHP script

```
<html>
<head>
<title>Basic Validation Script</title>
</head>
<body  style = "font-family: arial, sans-serif;
      background-color: #856363"  background=image1.jpg>
<?php
    //This script is intended to validate a phone number input from a form
    //generated by the XHTML document basic validation
    if (isset($_POST["submit"])) {
        if (preg_match("/^\([2-9][0-9]{2}\)[2-9][0-9]{2}-[0-9]{4}$/i", $_POST["phone"]) == False) {
            print("The phone number was incorrectly entered <br />");
        }
        else {
            print("The phone number was correctly entered. <br />");
            print("Processing form data begins...<br/>");
        }
    }
?>
</body>
</html>
```



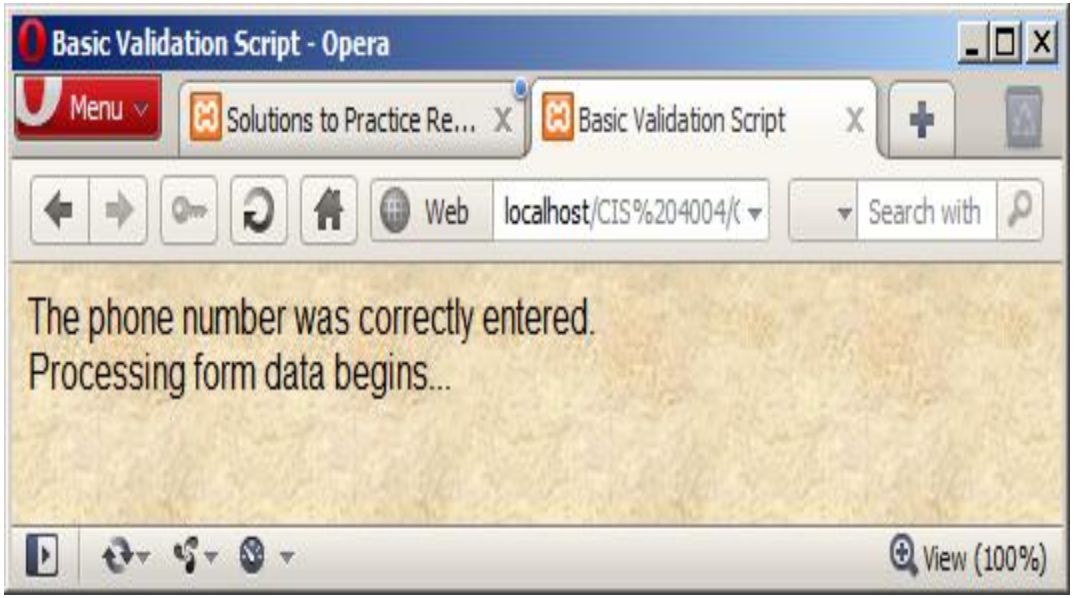
Enter Phone Number: (234)456-8799 [Use format: (407)555-1212]

Send

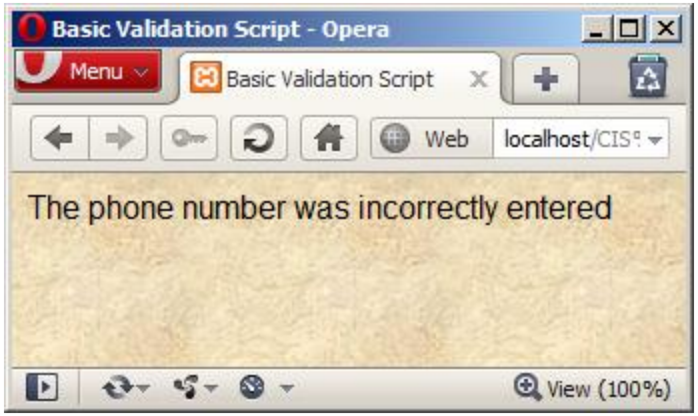


Enter Phone Number: 234-456-8799 [Use format: (407)555-1212]

Send



The phone number was correctly entered.
Processing form data begins...



The phone number was incorrectly entered

Basic Form Processing And Validation

- Because form validation is such an application-specific subject (every situation is different in some aspect), there is little benefit to discuss more about the general validation and processing of forms.

- Instead, the approach we'll take is to create a form-processing/data validation architecture that is general enough to use on any form without sacrificing flexibility.

- We'd also like to be able to separate the validation related code from the presentation related code. To this end, we'll develop three different files, the XHTML front-end form, and two PHP scripts on which is responsible for validating the form and the second which is responsible for validating the user-supplied data.

Basic Form Processing And Validation

- I’ve named the files I created for this as follows:

  ```
  big form example - frontend.html
  formvalidation.php
  userdefinedvalidation.php
  ```

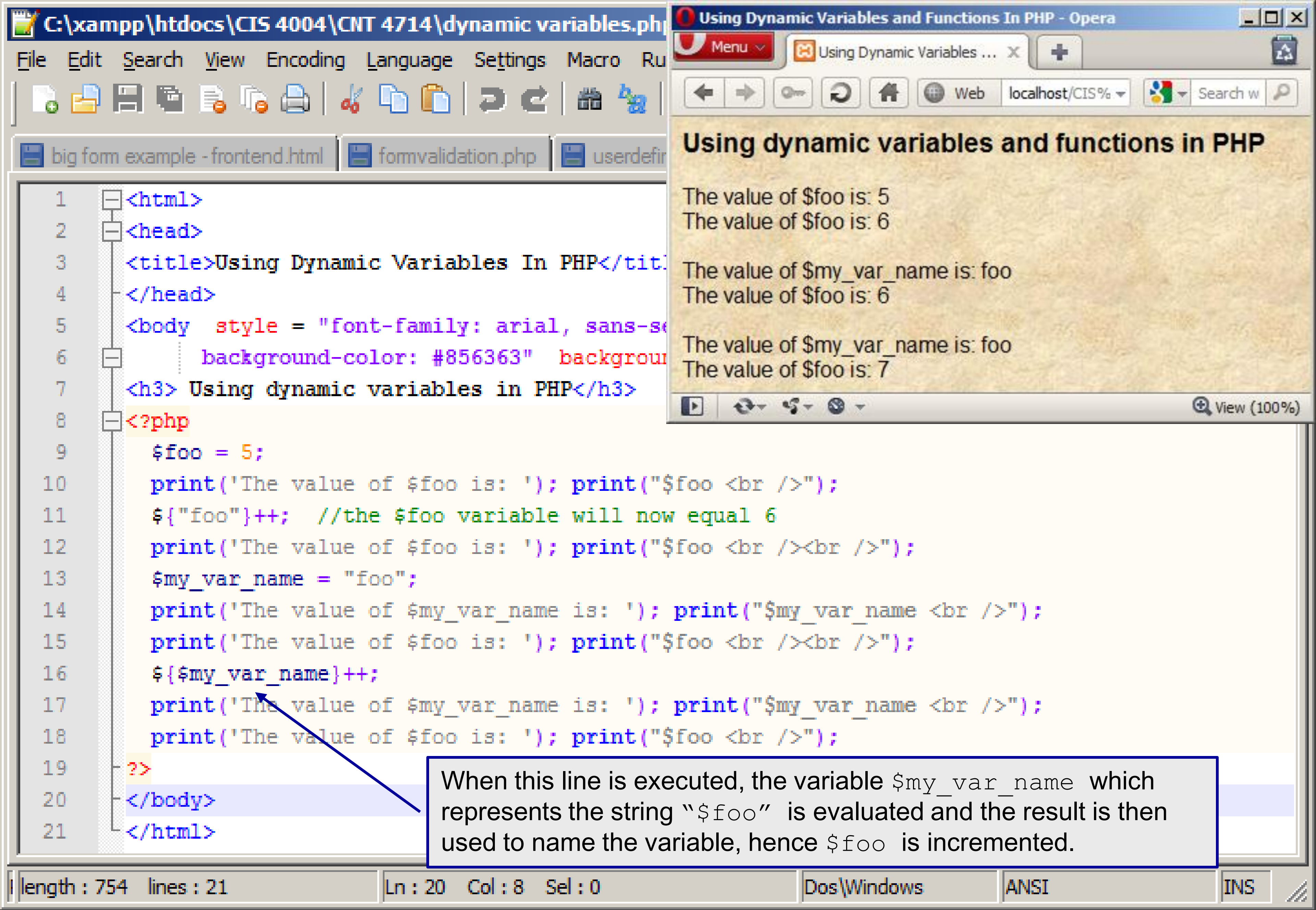
- To create such generic form processing scripts will require some fairly advanced PHP concepts that we’ll introduce along the way.

- The form validation script works through a combination of hidden form elements and dynamic functions calls. As we’ve already dealt with hidden form elements, we’ll take our first aside and look at dynamic functions in PHP.

An Aside – Dynamic Variables And Functions In PHP

- PHP supports both dynamic variables and dynamic functions.
- A dynamic variable is a variable whose actual identifier is unknown until the script is executed. Think of a dynamic variable as a "variable variable".
- The syntax for defining a dynamic variable is: `${<expression>}`

  where `expression` represents any valid PHP expression that evaluates to a value that follows the rules for what constitutes a variable name.
- The example script on the next page illustrates a dynamic variable.


C:\xampp\htdocs\CIS 4004\CNT 4714\dynamic variables.php
File Edit Search View Encoding Language Settings Macro Ru
big form example - frontend.html
formvalidation.php
userdefir
length : 754 lines : 21
Ln : 20 Col : 8 Sel : 0
Dos\Windows
ANSI
INS
Using Dynamic Variables and Functions In PHP - Opera
Menu
Using Dynamic Variables ...
Web
localhost/CIS%
Search w
View (100%)

```
<html>
<head>
<title>Using Dynamic Variables In PHP</titl
</head>
<body  style = "font-family: arial, sans-se
       background-color: #856363"  backgrou
<h3> Using dynamic variables in PHP</h3>
<?php
  $foo = 5;
  print('The value of $foo is: '); print("$foo <br />");
  ${"foo"}++;  //the $foo variable will now equal 6
  print('The value of $foo is: '); print("$foo <br /><br />");
  $my_var_name = "foo";
  print('The value of $my_var_name is: '); print("$my_var_name <br />");
  print('The value of $foo is: '); print("$foo <br /><br />");
  ${$my_var_name}++;
  print('The value of $my_var_name is: '); print("$my_var_name <br />");
  print('The value of $foo is: '); print("$foo <br />");
?>
</body>
</html>
```

Using dynamic variables and functions in PHP

```
The value of $foo is: 5
The value of $foo is: 6

The value of $my_var_name is: foo
The value of $foo is: 6

The value of $my_var_name is: foo
The value of $foo is: 7
```

When this line is executed, the variable `$my_var_name` which represents the string "`$foo`" is evaluated and the result is then used to name the variable, hence `$foo` is incremented.

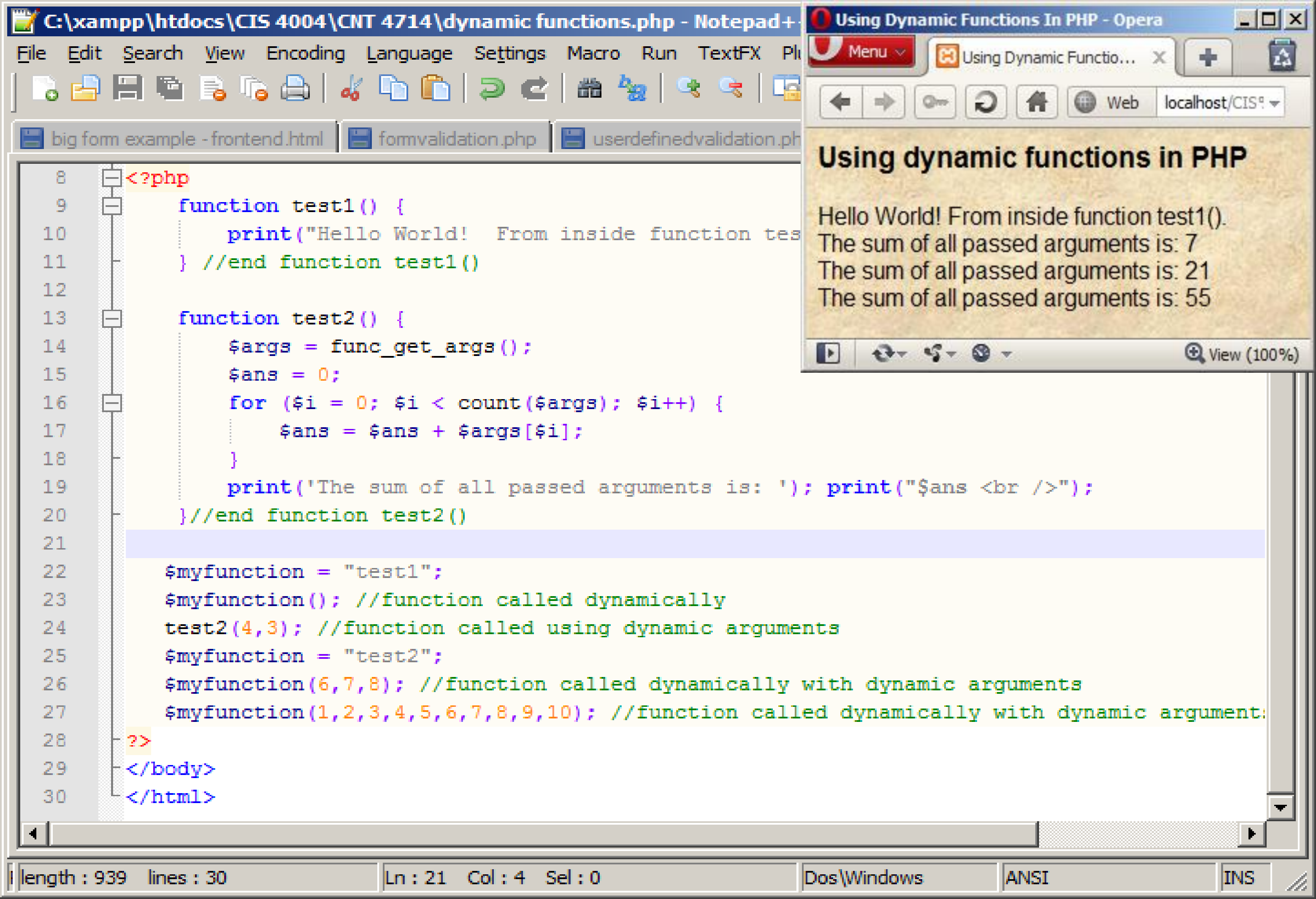
An Aside – Dynamic Variables And Functions In PHP

- Dynamic functions are particularly useful in PHP for validating form data as you'll shortly see.

- To execute a function whose name you do not know until runtime, you simply append a parameter list to the end of any variable.

- PHP also supports dynamic function parameters, meaning that the function definition does not define the number of parameters that will be passed to it, rather it is dynamically determined.

- There are two built-in supporting functions for this in PHP: `func_num_args()` returns the total number of arguments passed to the current function and `func_get_args()` is used to return an indexed array containing the values of each parameter.


C:\xampp\htdocs\CIS 4004\CNT 4714\dynamic functions.php - Notepad++
File Edit Search View Encoding Language Settings Macro Run TextFX Pl
big form example - frontend.html
formvalidation.php
userdefinedvalidation.ph
length : 939 lines : 30
Ln : 21 Col : 4 Sel : 0
Dos\Windows
ANSI
INS
Using Dynamic Functions In PHP - Opera
Menu
Using Dynamic Functio...
Web
localhost/CIS
View (100%)

```
<?php
    function test1() {
        print("Hello World!  From inside function tes
    } //end function test1()

    function test2() {
        $args = func_get_args();
        $ans = 0;
        for ($i = 0; $i < count($args); $i++) {
            $ans = $ans + $args[$i];
        }
        print('The sum of all passed arguments is: '); print("$ans <br />");
    }//end function test2()

   $myfunction = "test1";
   $myfunction(); //function called dynamically
   test2(4,3); //function called using dynamic arguments
   $myfunction = "test2";
   $myfunction(6,7,8); //function called dynamically with dynamic arguments
   $myfunction(1,2,3,4,5,6,7,8,9,10); //function called dynamically with dynamic argument
?>
</body>
</html>
```

Using dynamic functions in PHP

```
Hello World! From inside function test1().
The sum of all passed arguments is: 7
The sum of all passed arguments is: 21
The sum of all passed arguments is: 55
```

Basic Form Processing And Validation

- Getting back to the form processing side of things, the hidden form elements will be used by the PHP script to provide both a human-friendly description of each field element (useful when an error occurs) and to identify those form fields that are "required".

- Specifically, for any given form element with a name of `<name>`, the description of that field is defined as being stored in a hidden element by the name of `<name>_desc`.

- The second hidden form element that the form-processing script uses is called `required` and should contain a comma-separated list of required elements.

Basic Form Processing And Validation

- The `value` attribute of each visible element in the form should be populated with it s associated value from the superglobal (`$_POST`).

- This is done primarily so that if the form is submitted and not processed for whatever reason (an error has occurred) the user will not have to retype the form.

- The next issue of concern is how to deal with validation errors that may occur when the form is submitted. This will be handled in the form validation script through two global variables: `$form_errors` and `$form_errorlist`. When the form validation script attempts to validate the data submitted to it, upon an error, it creates these two dynamic variables.

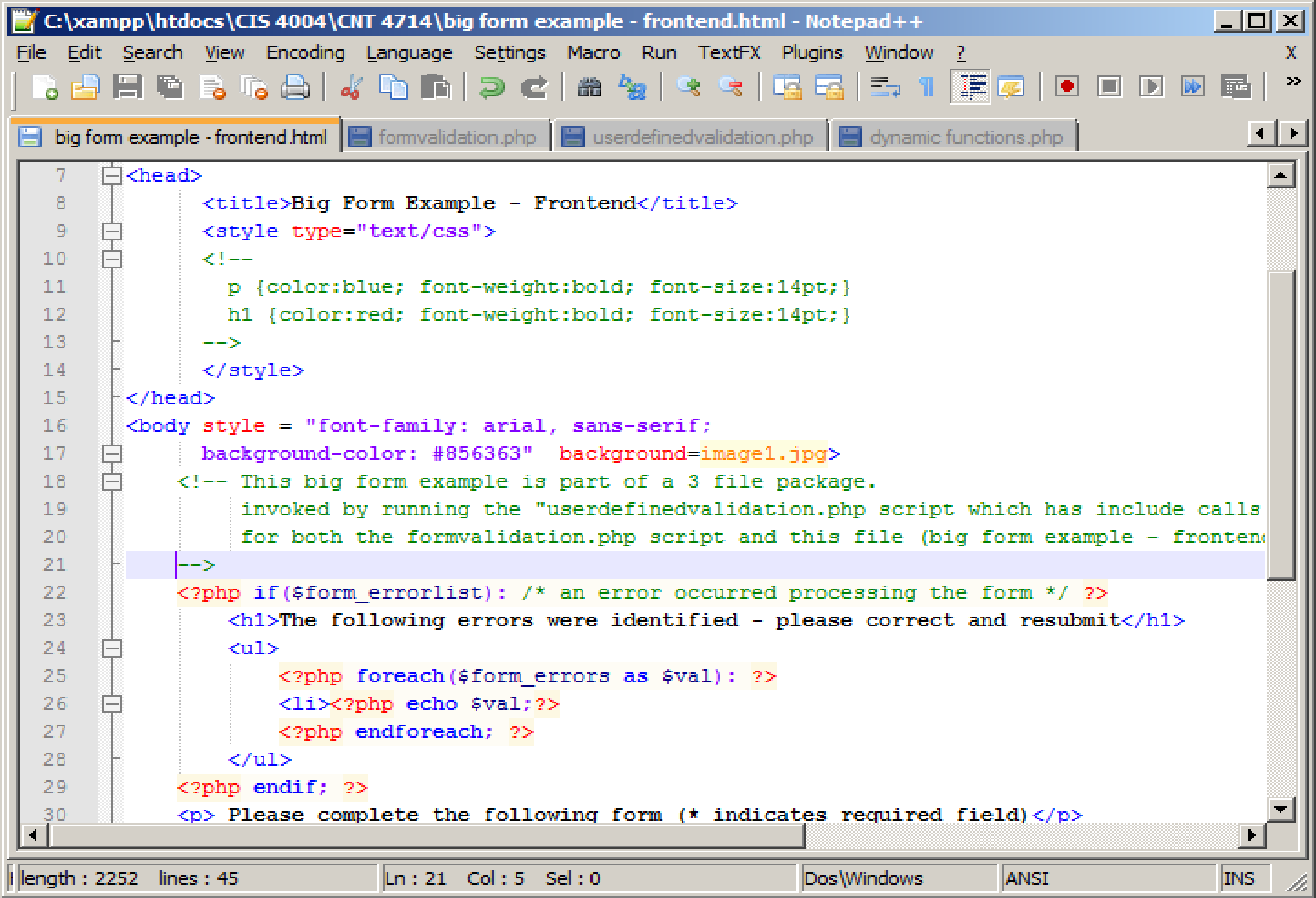
Basic Form Processing And Validation

- The first variable `$form_errors` is a Boolean value indicating whether an error occurred during the validation, and the second `$form_errorlist` is an array of error messages that occurred during validation.

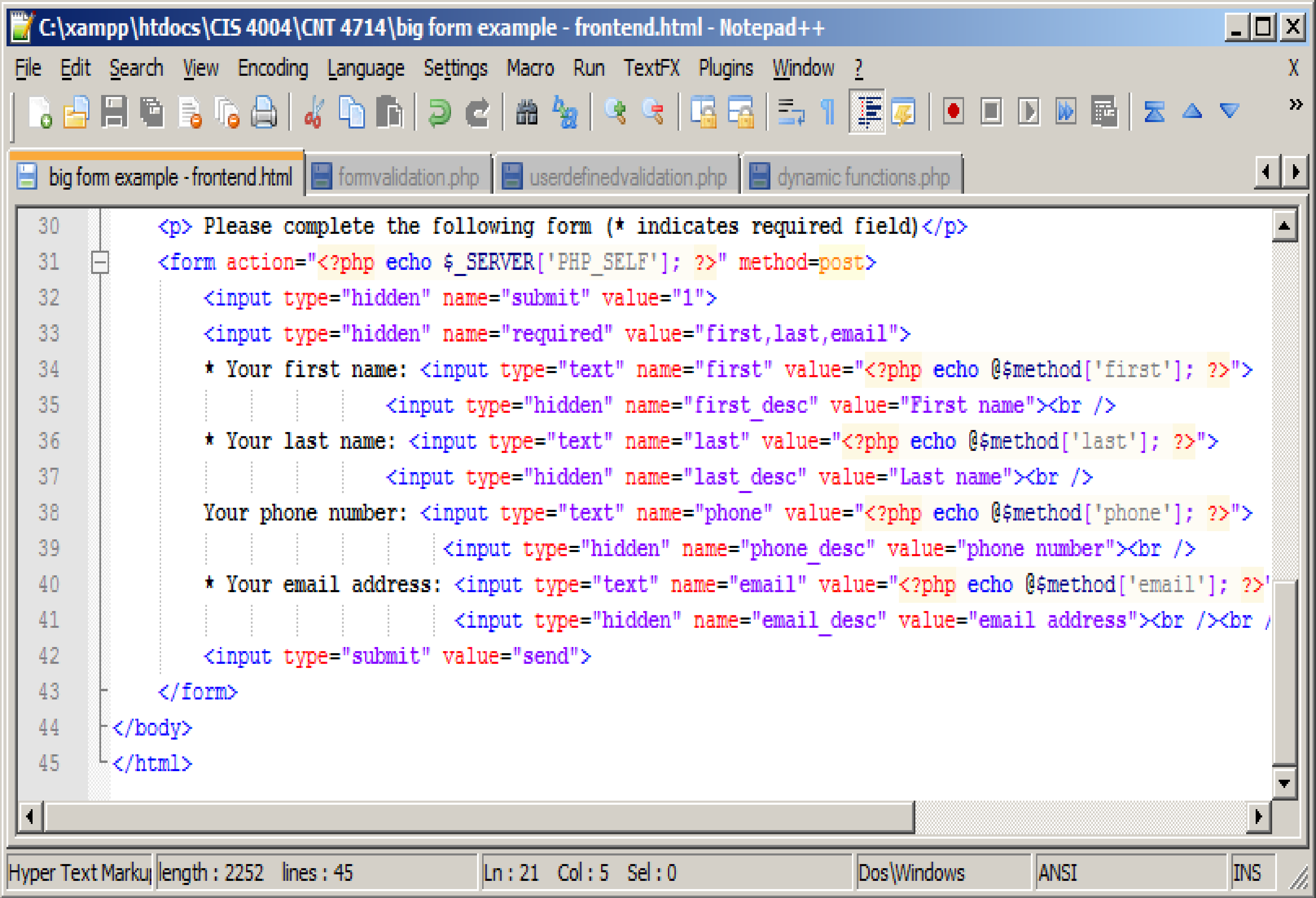
- How these variables are used in your script to display validation errors to the user is subjective and can be done in many different ways.

- What I've chosen to do in the example is to use a little script at the top of the XHTML document that will display an unordered list (bulleted list) of all the errors that occurred and need attention from the user.

- The next two pages illustrate the complete XHTML front-end.


C:\xampp\htdocs\CIS 4004\CNT 4714\big form example - frontend.html - Notepad++
File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ?
big form example - frontend.html
formvalidation.php
userdefinedvalidation.php
dynamic functions.php
length : 2252 lines : 45
Ln : 21 Col : 5 Sel : 0
Dos\Windows
ANSI
INS

```
<head>
      <title>Big Form Example - Frontend</title>
      <style type="text/css">
      <!--
        p {color:blue; font-weight:bold; font-size:14pt;}
        h1 {color:red; font-weight:bold; font-size:14pt;}
      -->
      </style>
</head>
<body style = "font-family: arial, sans-serif;
      background-color: #856363"  background=image1.jpg>
    <!-- This big form example is part of a 3 file package.
         invoked by running the "userdefinedvalidation.php script which has include calls
         for both the formvalidation.php script and this file (big form example - frontend
    -->
    <?php if($form_errorlist): /* an error occurred processing the form */ ?>
      <h1>The following errors were identified - please correct and resubmit</h1>
      <ul>
          <?php foreach($form_errors as $val): ?>
          <li><?php echo $val;?>
          <?php endforeach; ?>
      </ul>
    <?php endif; ?>
    <p> Please complete the following form (* indicates required field)</p>
```


C:\xampp\htdocs\CIS 4004\CNT 4714\big form example - frontend.html - Notepad++
File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ?
big form example - frontend.html
formvalidation.php
userdefinedvalidation.php
dynamic functions.php
Hyper Text Marku
length : 2252 lines : 45
Ln : 21 Col : 5 Sel : 0
Dos\Windows
ANSI
INS

```
    <p> Please complete the following form (* indicates required field)</p>
    <form action="<?php echo $_SERVER['PHP_SELF']; ?>" method=post>
        <input type="hidden" name="submit" value="1">
        <input type="hidden" name="required" value="first,last,email">
        * Your first name: <input type="text" name="first" value="<?php echo @$method['first']; ?>">
                        <input type="hidden" name="first_desc" value="First name"><br />
        * Your last name: <input type="text" name="last" value="<?php echo @$method['last']; ?>">
                        <input type="hidden" name="last_desc" value="Last name"><br />
        Your phone number: <input type="text" name="phone" value="<?php echo @$method['phone']; ?>">
                             <input type="hidden" name="phone_desc" value="phone number"><br />
        * Your email address: <input type="text" name="email" value="<?php echo @$method['email']; ?>'
                              <input type="hidden" name="email_desc" value="email address"><br /><br /
        <input type="submit" value="send">
    </form>
</body>
</html>
```

Basic Form Processing And Validation

- The next step is to deal with the validation of the form.
- I’ve divided the form validation script into three separate functions: `add_error()`, `_process_form()`, and `validate_form()`.
- The function `validate_form()` does most of the work, with the other two functioning as support or helper functions.
- The `add_error()` function uses the two dynamic variables `$form_errors` and `$form_errorlist`. The function itself is quite simple and simply adds errors when they occur to the `$form_errorlist`. This function appears at the top of the form validation script as seen on page 43.

Basic Form Processing And Validation

- The main part of the form validation script is in the `validate_form()` function. This function takes a single parameter, which is a reference to the superglobal array to validate. When executed, the function attempts to perform a number of tasks in the interest of validating the data.

- During the course of this operation, if any validation errors occur, `validate_form()` calls the `add_error()` function with an appropriate error message and thus populates the error variables.

- When executed, `validate_form()` starts by first processing the `required` hidden field and checks to make sure that all required fields are not empty (the user has supplied values for all required fields).

Basic Form Processing And Validation

- Following that check, the function attempts to process each individual form element according to the following rules:
  - If the element is named `submit`, `required`, or ends in `_desc`, it is ignored.
  - For all other elements, the function attempts to call the function `<name>_validate()`, where `<name>` is the name of the current element.
- Unless defined by the user, the `<name>_validate()` functions do not exist. These function are your responsibility to create to validate each individual form element (or at least the elements you are concerned with validating).

Basic Form Processing And Validation

- These functions should accept two parameters, the value submitted and a description of the field taken from the `<name>_desc` element.

- These functions should return `true` if the submitted value is valid or return an error message upon failure to validate the user supplied value.

  – For example, if you were validating a form element whose name attribute is phone (for a phone number), then the following function would need to be constructed:

```
function phone_validate($data, $desc) {
   $regex = “/^\([2-9][0-9]{2}\)[2-9][0-9]{2}-[0-9]{4}/i”;
   if preg_match($regex, $data) != 1) {
        return(“The ‘$desc’ field isn’t valid!”);
   }
   return true;
}
```

Basic Form Processing And Validation

- Assuming that the `validate_form()` function executes and does not encounter any errors, it then calls the `_process_form()` function.

- This function is designed to clean up any nonrequired form elements (the `_desc`, `submit`, and `required` hidden elements) and call the function `process_form()`.

- As with the validation functions we just discussed, the `process_form()` function must be defined by you and is designed to allow you to actually perform whatever action was desired after a successful validation. You might send an email, or submit data to a database, or any number of actions.

- This function accepts a single parameter, an array of the submitted data, and has no return value.

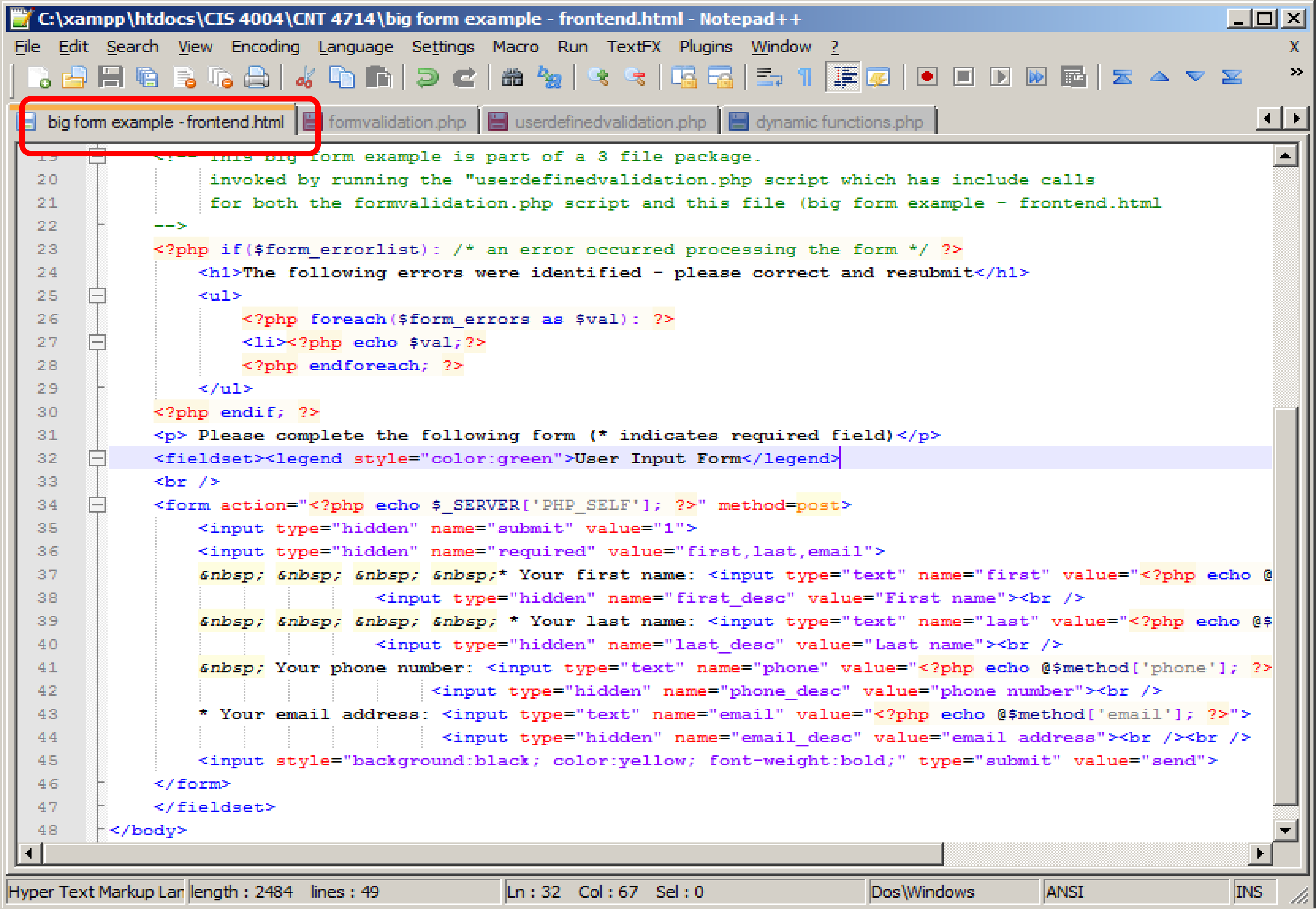
Basic Form Processing And Validation

- If this function does not exist (i.e., you don't' create one), nothing will happen to the submitted data upon a successful validation.

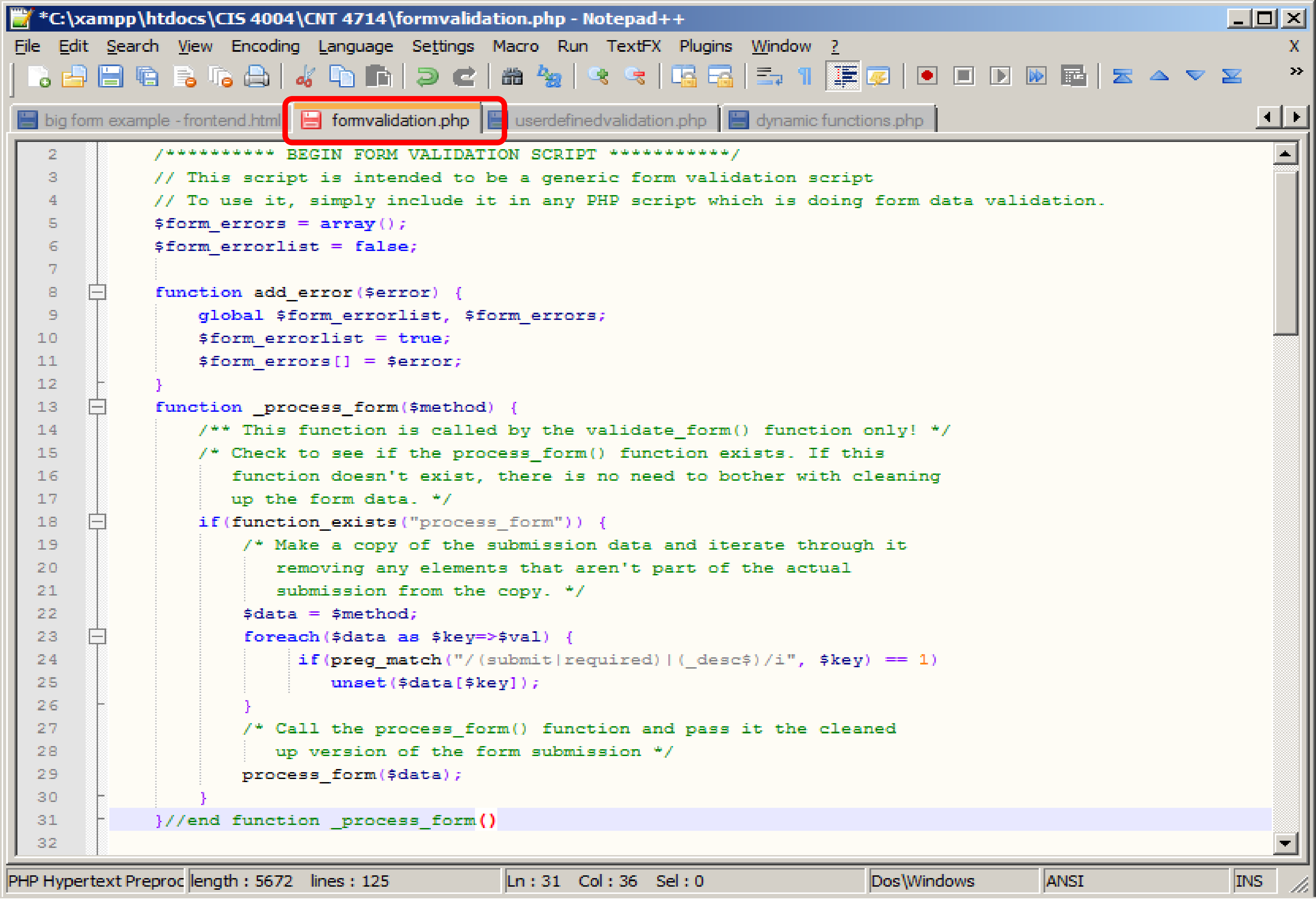
- In the example, I've simply printed out the data from the form, but I put a commented line in to show how you might email the data to yourself. The example function appears at the end of the script named `userdefinedvalidation.php` shown beginning on page 47.

- Look at the markup and scripts for this application beginning on the next page.
  - big form example – frontend.html document – page 42
  - formvalidation.php script – page 43
  - userdefinedvalidation.php script – page 47

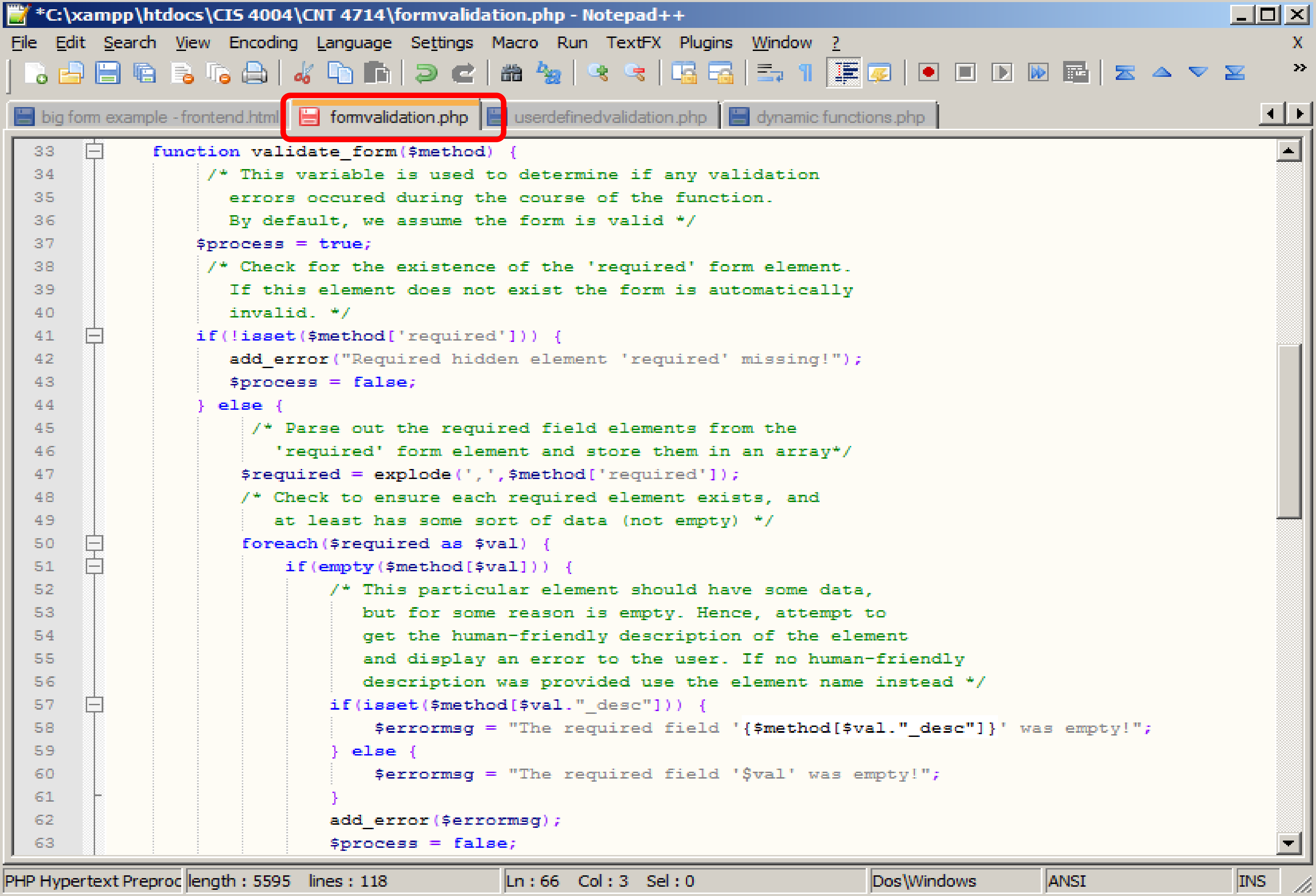
- The execution of these is shown beginning on page 49.


C:\xampp\htdocs\CIS 4004\CNT 4714\big form example - frontend.html - Notepad++
File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ?
big form example - frontend.html
formvalidation.php
userdefinedvalidation.php
dynamic functions.php
Hyper Text Markup Lan
length : 2484 lines : 49
Ln : 32 Col : 67 Sel : 0
Dos\Windows
ANSI
INS

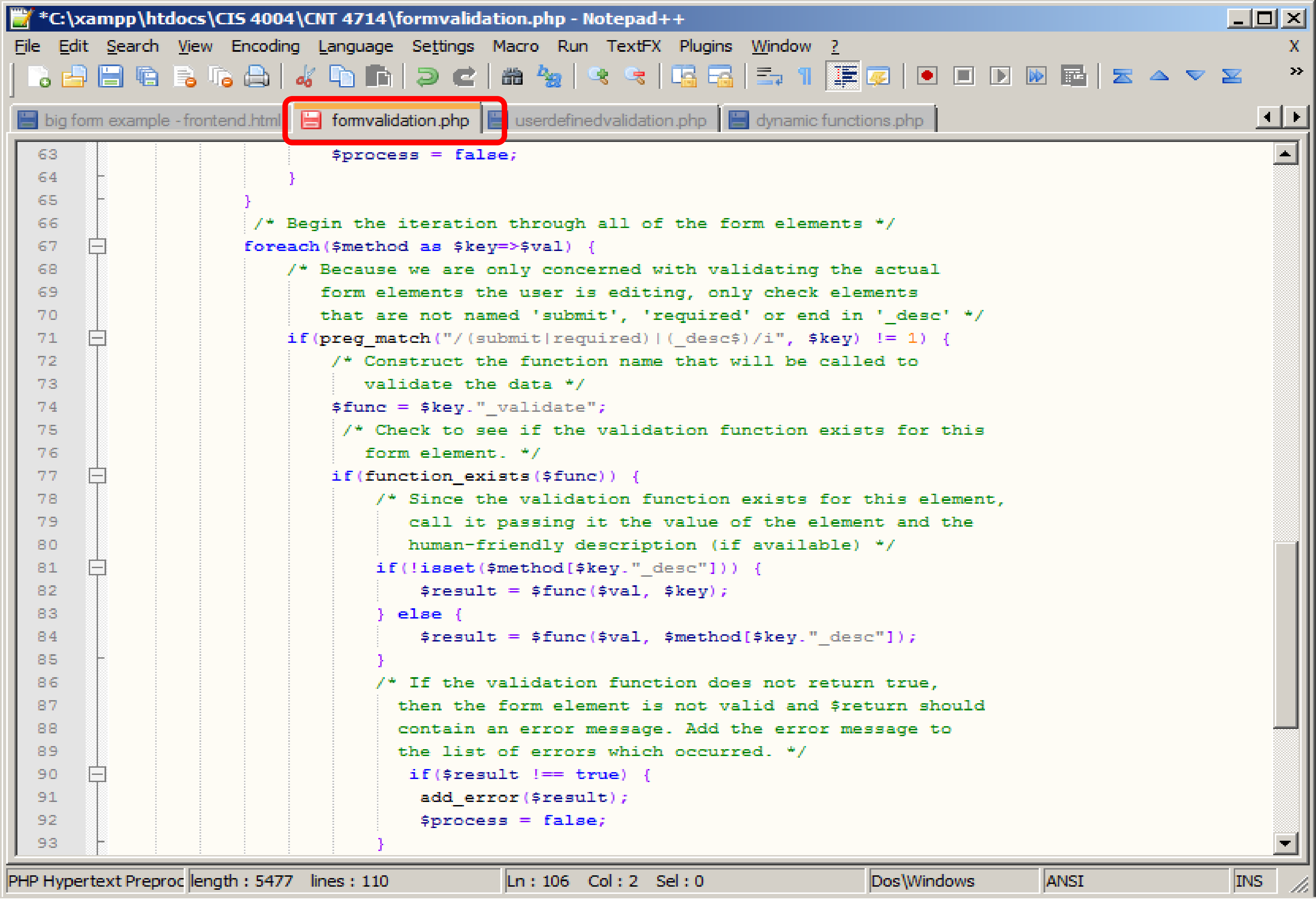
```
  <!-- This big form example is part of a 3 file package.
       invoked by running the "userdefinedvalidation.php script which has include calls
       for both the formvalidation.php script and this file (big form example - frontend.html
  -->
  <?php if($form_errorlist): /* an error occurred processing the form */ ?>
    <h1>The following errors were identified - please correct and resubmit</h1>
    <ul>
      <?php foreach($form_errors as $val): ?>
      <li><?php echo $val;?>
      <?php endforeach; ?>
    </ul>
  <?php endif; ?>
  <p> Please complete the following form (* indicates required field)</p>
  <fieldset><legend style="color:green">User Input Form</legend>
  <br />
  <form action="<?php echo $_SERVER['PHP_SELF']; ?>" method=post>
    <input type="hidden" name="submit" value="1">
    <input type="hidden" name="required" value="first,last,email">
    &nbsp; &nbsp; &nbsp; &nbsp;* Your first name: <input type="text" name="first" value="<?php echo @
            <input type="hidden" name="first_desc" value="First name"><br />
    &nbsp; &nbsp; &nbsp; &nbsp; * Your last name: <input type="text" name="last" value="<?php echo @$
            <input type="hidden" name="last_desc" value="Last name"><br />
    &nbsp; Your phone number: <input type="text" name="phone" value="<?php echo @$method['phone']; ?>
               <input type="hidden" name="phone_desc" value="phone number"><br />
    * Your email address: <input type="text" name="email" value="<?php echo @$method['email']; ?>">
               <input type="hidden" name="email_desc" value="email address"><br /><br />
    <input style="background:black; color:yellow; font-weight:bold;" type="submit" value="send">
  </form>
  </fieldset>
</body>
```


*C:\xampp\htdocs\CIS 4004\CNT 4714\formvalidation.php - Notepad++
File
Edit
Search
View
Encoding
Language
Settings
Macro
Run
TextFX
Plugins
Window
?
big form example - frontend.html
formvalidation.php
userdefinedvalidation.php
dynamic functions.php
PHP Hypertext Preproc
length : 5672   lines : 125
Ln : 31   Col : 36   Sel : 0
Dos\Windows
ANSI
INS

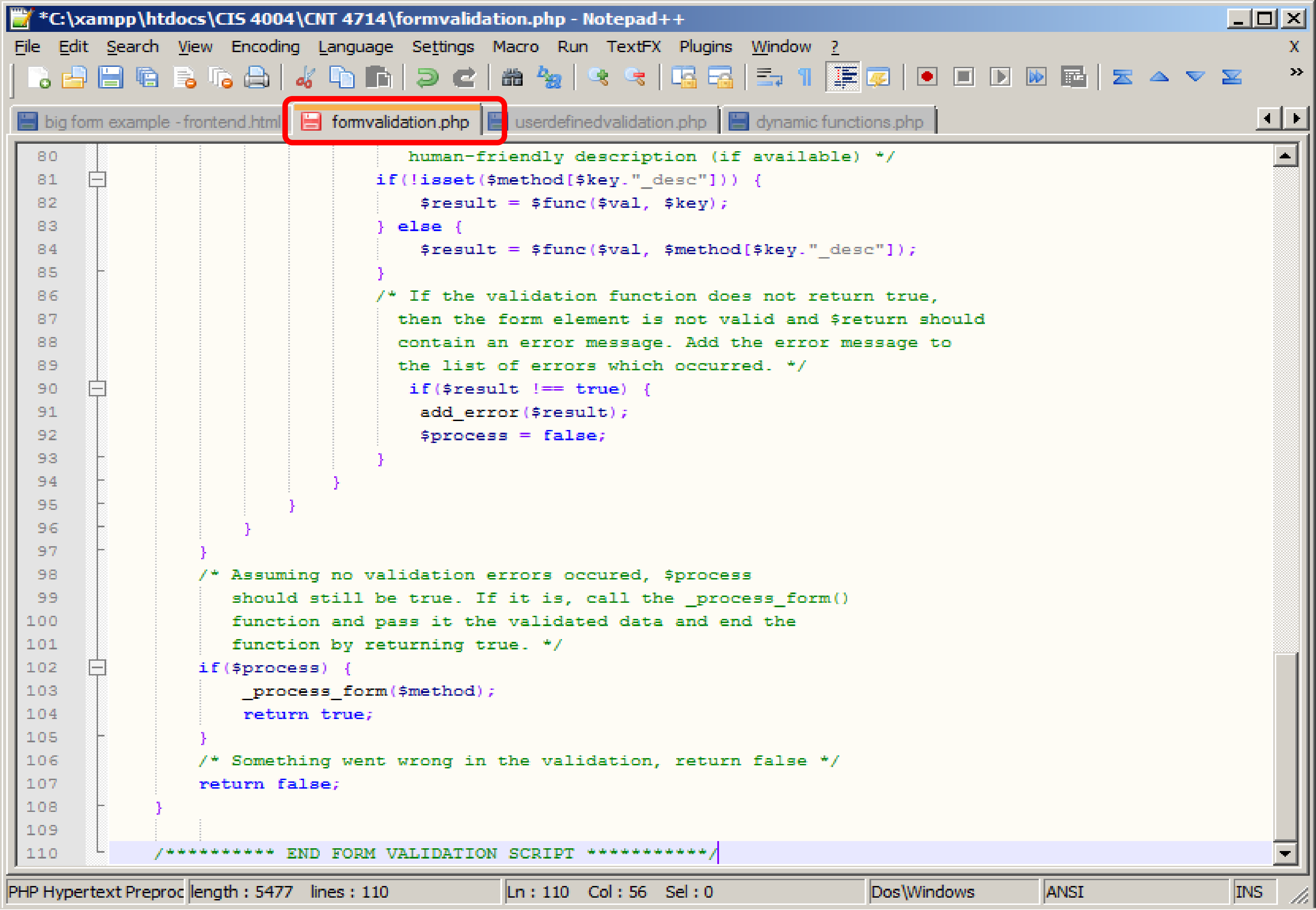
```php
/*********** BEGIN FORM VALIDATION SCRIPT ************/
// This script is intended to be a generic form validation script
// To use it, simply include it in any PHP script which is doing form data validation.
$form_errors = array();
$form_errorlist = false;

function add_error($error) {
    global $form_errorlist, $form_errors;
    $form_errorlist = true;
    $form_errors[] = $error;
}
function _process_form($method) {
    /** This function is called by the validate_form() function only! */
    /* Check to see if the process_form() function exists. If this
       function doesn't exist, there is no need to bother with cleaning
       up the form data. */
    if(function_exists("process_form")) {
        /* Make a copy of the submission data and iterate through it
           removing any elements that aren't part of the actual
           submission from the copy. */
        $data = $method;
        foreach($data as $key=>$val) {
            if(preg_match("/(submit|required)|(_desc$)/i", $key) == 1)
               unset($data[$key]);
        }
        /* Call the process_form() function and pass it the cleaned
           up version of the form submission */
        process_form($data);
    }
}//end function _process_form()
```


*C:\xampp\htdocs\CIS 4004\CNT 4714\formvalidation.php - Notepad++
File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ?
big form example - frontend.html
formvalidation.php
userdefinedvalidation.php
dynamic functions.php
PHP Hypertext Preproc
length : 5595 lines : 118
Ln : 66 Col : 3 Sel : 0
Dos\Windows
ANSI
INS

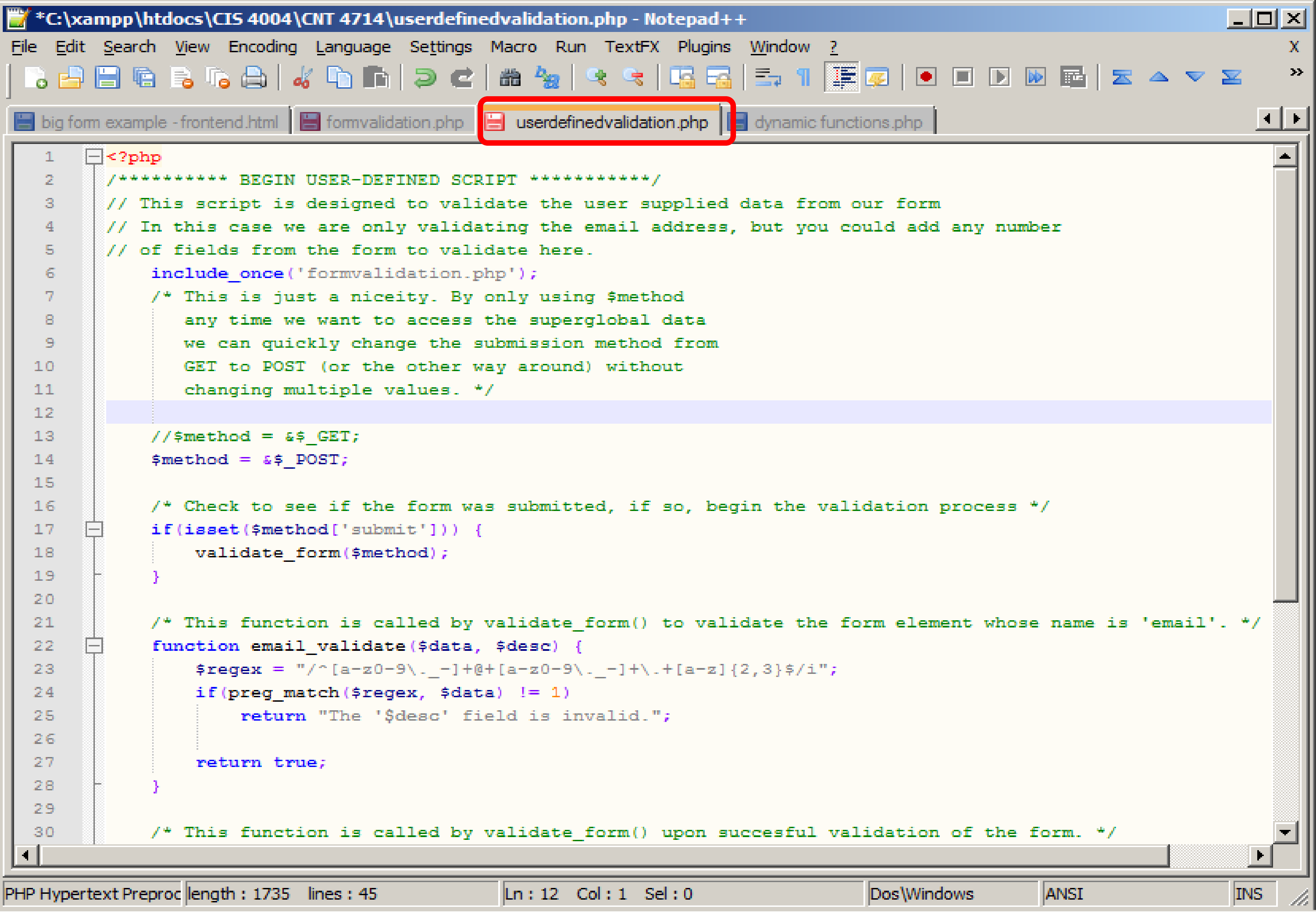
```php
function validate_form($method) {
     /* This variable is used to determine if any validation
       errors occured during the course of the function.
       By default, we assume the form is valid */
    $process = true;
     /* Check for the existence of the 'required' form element.
       If this element does not exist the form is automatically
       invalid. */
    if(!isset($method['required'])) {
       add_error("Required hidden element 'required' missing!");
       $process = false;
    } else {
         /* Parse out the required field elements from the
           'required' form element and store them in an array*/
        $required = explode(',',$method['required']);
        /* Check to ensure each required element exists, and
           at least has some sort of data (not empty) */
        foreach($required as $val) {
            if(empty($method[$val])) {
                /* This particular element should have some data,
                   but for some reason is empty. Hence, attempt to
                   get the human-friendly description of the element
                   and display an error to the user. If no human-friendly
                   description was provided use the element name instead */
                if(isset($method[$val."_desc"])) {
                    $errormsg = "The required field '{$method[$val."_desc"]}' was empty!";
                } else {
                    $errormsg = "The required field '$val' was empty!";
                }
                add_error($errormsg);
                $process = false;
```


*C:\xampp\htdocs\CIS 4004\CNT 4714\formvalidation.php - Notepad++
File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ?
big form example - frontend.html
formvalidation.php
userdefinedvalidation.php
dynamic functions.php
PHP Hypertext Preproc
length : 5477 lines : 110
Ln : 106 Col : 2 Sel : 0
Dos\Windows
ANSI
INS

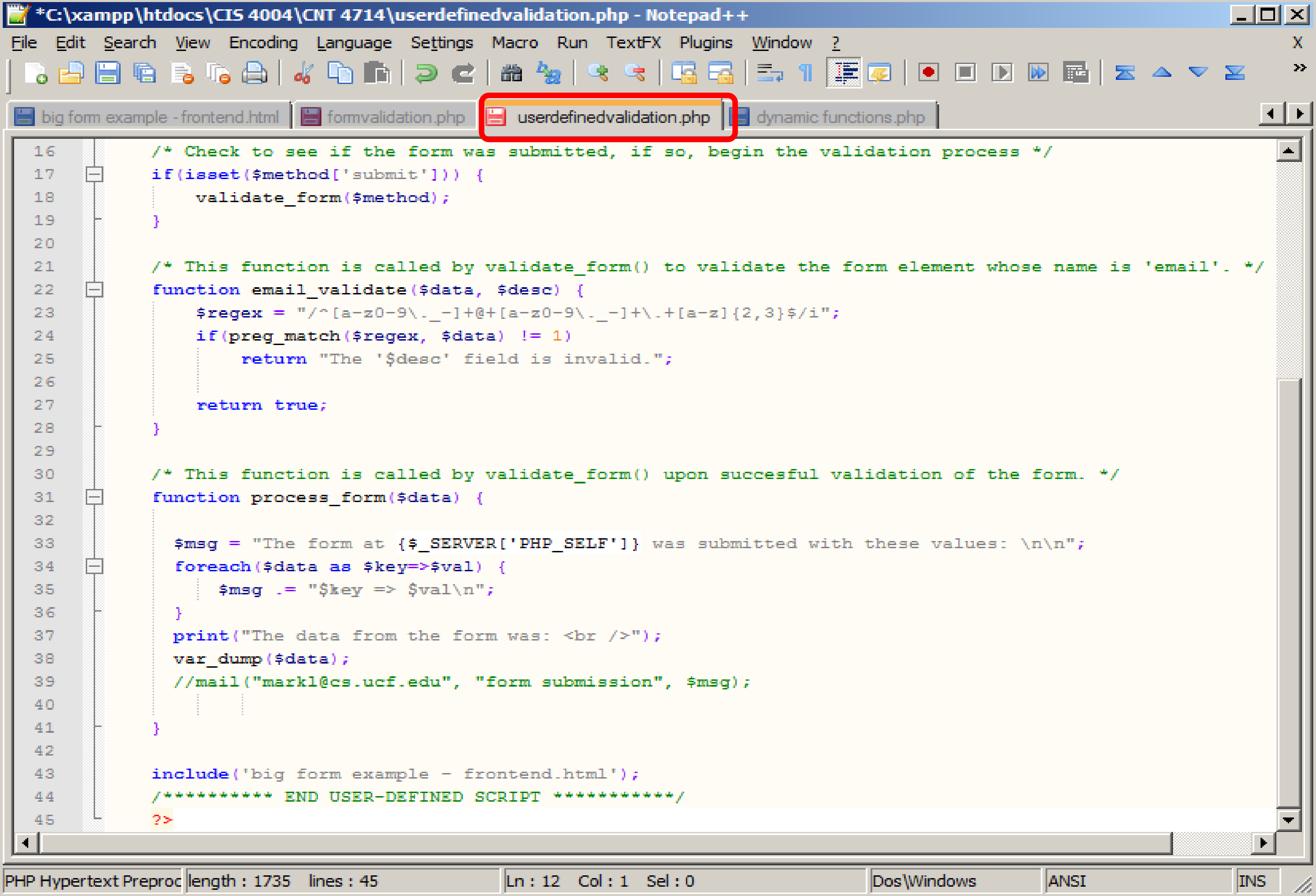
```
                    $process = false;
                }
            }
             /* Begin the iteration through all of the form elements */
            foreach($method as $key=>$val) {
                /* Because we are only concerned with validating the actual
                   form elements the user is editing, only check elements
                   that are not named 'submit', 'required' or end in '_desc' */
                if(preg_match("/(submit|required)|(_desc$)/i", $key) != 1) {
                    /* Construct the function name that will be called to
                       validate the data */
                    $func = $key."_validate";
                     /* Check to see if the validation function exists for this
                       form element. */
                    if(function_exists($func)) {
                        /* Since the validation function exists for this element,
                           call it passing it the value of the element and the
                           human-friendly description (if available) */
                        if(!isset($method[$key."_desc"])) {
                            $result = $func($val, $key);
                        } else {
                            $result = $func($val, $method[$key."_desc"]);
                        }
                        /* If the validation function does not return true,
                          then the form element is not valid and $return should
                          contain an error message. Add the error message to
                          the list of errors which occurred. */
                           if($result !== true) {
                            add_error($result);
                            $process = false;
                        }
```


*C:\xampp\htdocs\CIS 4004\CNT 4714\formvalidation.php - Notepad++
File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ?
big form example - frontend.html
formvalidation.php
userdefinedvalidation.php
dynamic functions.php
PHP Hypertext Preproc
length : 5477 lines : 110
Ln : 110 Col : 56 Sel : 0
Dos\Windows
ANSI
INS

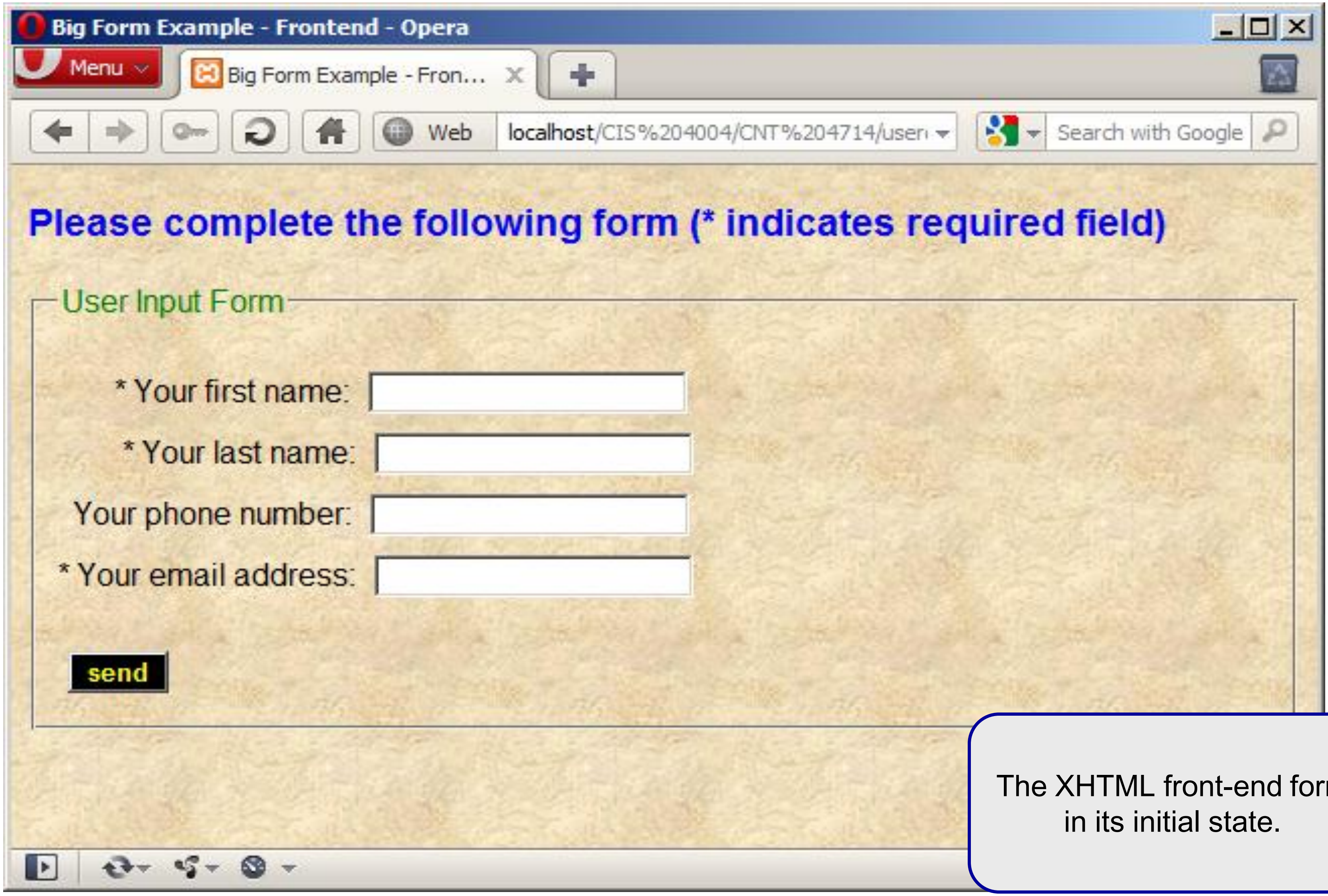
```
                           human-friendly description (if available) */
                        if(!isset($method[$key."_desc"])) {
                            $result = $func($val, $key);
                        } else {
                            $result = $func($val, $method[$key."_desc"]);
                        }
                        /* If the validation function does not return true,
                          then the form element is not valid and $return should
                          contain an error message. Add the error message to
                          the list of errors which occurred. */
                           if($result !== true) {
                            add_error($result);
                            $process = false;
                        }
                    }
                }
            }
        }
        /* Assuming no validation errors occured, $process
           should still be true. If it is, call the _process_form()
           function and pass it the validated data and end the
           function by returning true. */
        if($process) {
            _process_form($method);
            return true;
        }
        /* Something went wrong in the validation, return false */
        return false;
    }

    /*********** END FORM VALIDATION SCRIPT ************/
```


*C:\xampp\htdocs\CIS 4004\CNT 4714\userdefinedvalidation.php - Notepad++
File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ?
big form example - frontend.html
formvalidation.php
userdefinedvalidation.php
dynamic functions.php
PHP Hypertext Preproc
length : 1735 lines : 45
Ln : 12 Col : 1 Sel : 0
Dos\Windows
ANSI
INS

```php
<?php
/*********** BEGIN USER-DEFINED SCRIPT ************/
// This script is designed to validate the user supplied data from our form
// In this case we are only validating the email address, but you could add any number
// of fields from the form to validate here.
    include_once('formvalidation.php');
    /* This is just a niceity. By only using $method
       any time we want to access the superglobal data
       we can quickly change the submission method from
       GET to POST (or the other way around) without
       changing multiple values. */

    //$method = &$_GET;
    $method = &$_POST;

    /* Check to see if the form was submitted, if so, begin the validation process */
    if(isset($method['submit'])) {
        validate_form($method);
    }

    /* This function is called by validate_form() to validate the form element whose name is 'email'. */
    function email_validate($data, $desc) {
        $regex = "/^[a-z0-9\._-]+@+[a-z0-9\._-]+\.+[a-z]{2,3}$/i";
        if(preg_match($regex, $data) != 1)
            return "The '$desc' field is invalid.";

        return true;
    }

    /* This function is called by validate_form() upon succesful validation of the form. */
```



```
    /* Check to see if the form was submitted, if so, begin the validation process */
    if(isset($method['submit'])) {
        validate_form($method);
    }

    /* This function is called by validate_form() to validate the form element whose name is 'email'. */
    function email_validate($data, $desc) {
        $regex = "/^[a-z0-9\._-]+@+[a-z0-9\._-]+\.+[a-z]{2,3}$/i";
        if(preg_match($regex, $data) != 1)
            return "The '$desc' field is invalid.";

        return true;
    }

    /* This function is called by validate_form() upon succesful validation of the form. */
    function process_form($data) {

      $msg = "The form at {$_SERVER['PHP_SELF']} was submitted with these values: \n\n";
      foreach($data as $key=>$val) {
          $msg .= "$key => $val\n";
      }
      print("The data from the form was: <br />");
      var_dump($data);
      //mail("markl@cs.ucf.edu", "form submission", $msg);

    }

    include('big form example - frontend.html');
    /*********** END USER-DEFINED SCRIPT ************/
    ?>
```

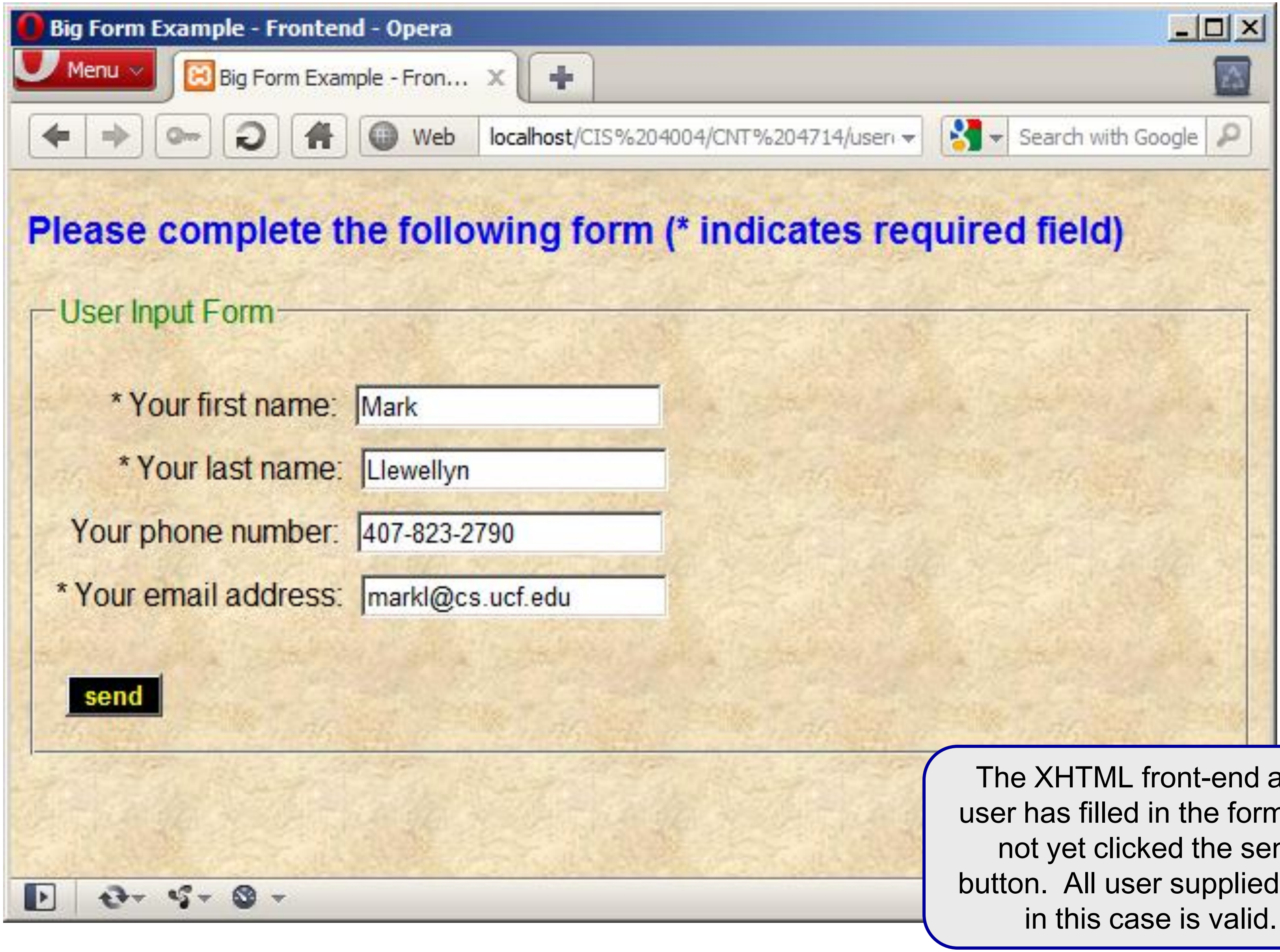

Big Form Example - Frontend - Opera
Menu
Big Form Example - Fron...
Web
localhost/CIS%204004/CNT%204714/user
Search with Google

Please complete the following form (* indicates required field)

User Input Form

* Your first name:	
* Your last name:	
Your phone number:	
* Your email address:	

The XHTML front-end form in its initial state.


Big Form Example - Frontend - Opera
Menu
Big Form Example - Fron...
Web
localhost/CIS%204004/CNT%204714/user
Search with Google

Please complete the following form (* indicates required field)

User Input Form

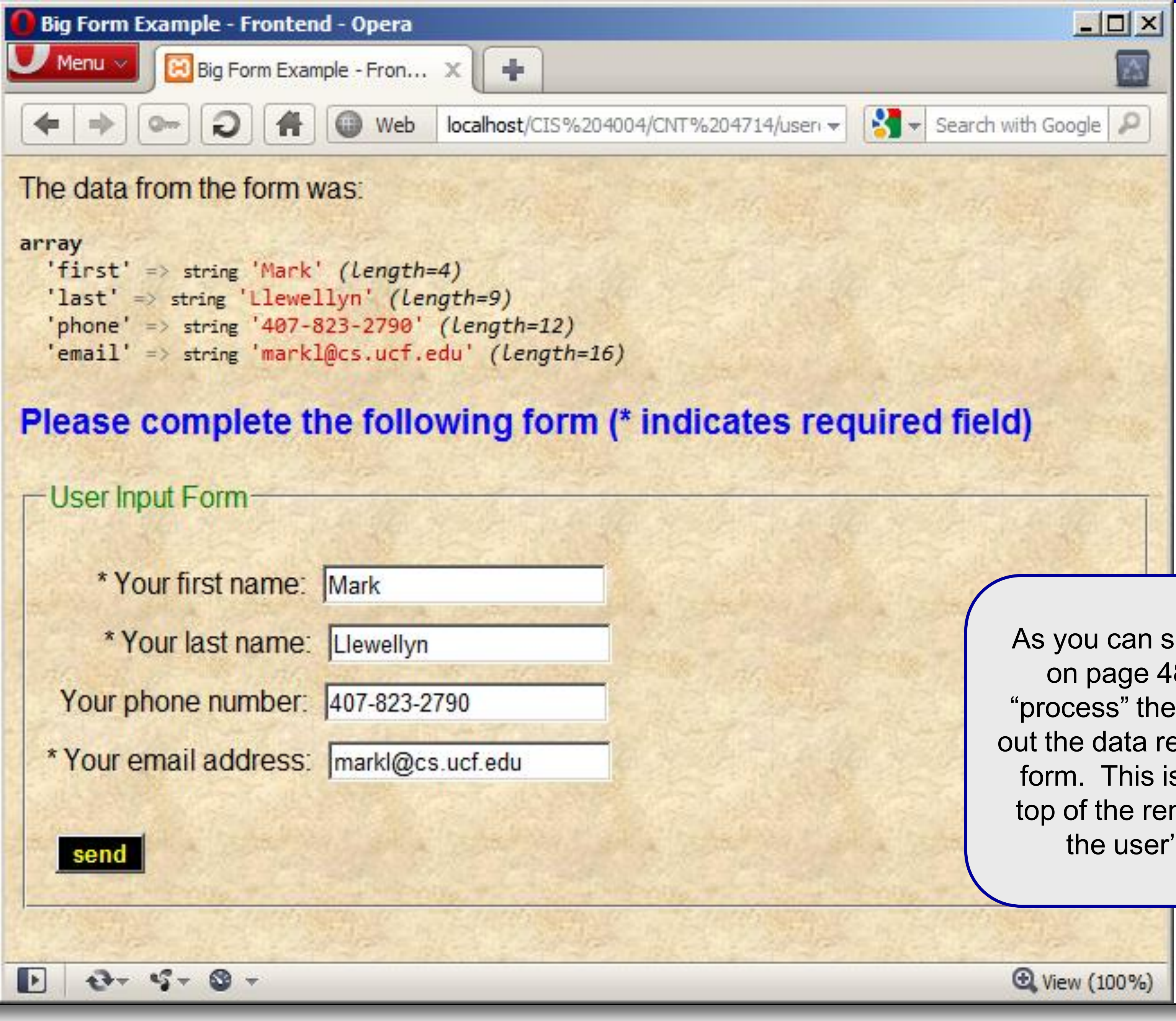
* Your first name: Mark

* Your last name: Llewellyn

Your phone number: 407-823-2790

* Your email address: markl@cs.ucf.edu

The XHTML front-end after user has filled in the form, but not yet clicked the send button. All user supplied data in this case is valid.


Big Form Example - Frontend - Opera
Menu
Big Form Example - Fron...
Web
localhost/CIS%204004/CNT%204714/user
Search with Google
View (100%)

The data from the form was:

```
array
  'first' => string 'Mark' (length=4)
  'last' => string 'Llewellyn' (length=9)
  'phone' => string '407-823-2790' (length=12)
  'email' => string 'markl@cs.ucf.edu' (length=16)
```

Please complete the following form (* indicates required field)

User Input Form

* Your first name:	Mark
* Your last name:	Llewellyn
Your phone number:	407-823-2790
* Your email address:	markl@cs.ucf.edu

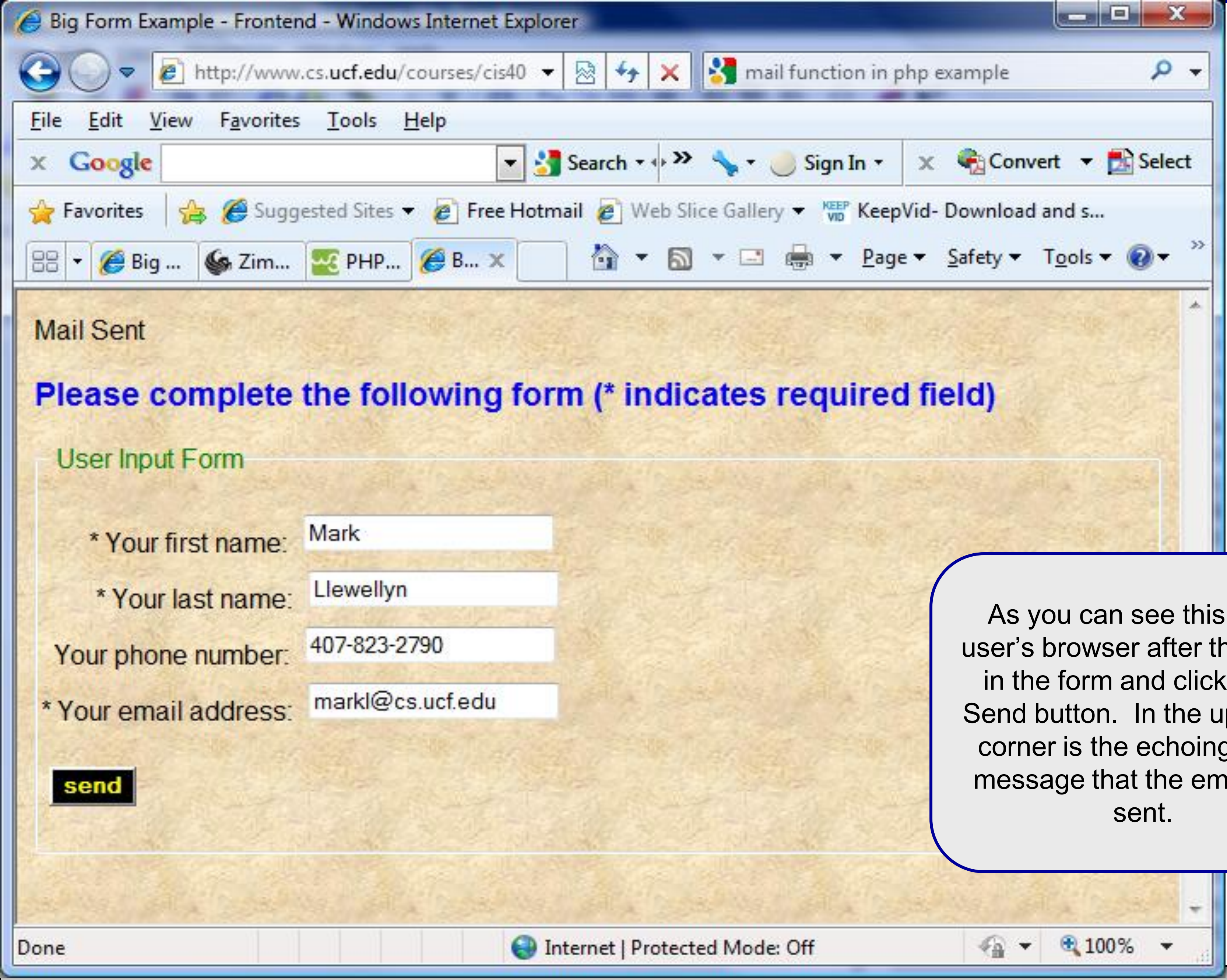
send

As you can see in the script on page 48, all I did to "process" the form was print out the data received from the form. This is shown at the top of the rendered page in the user's browser.

Basic Form Processing And Validation

- If we modify the `userdefinedvalidation()` function so that rather than echoing the form data back to the user, we send an email, the function `process_form()` will look like the following:

```
/* This function is called by validate_form() upon successful validation of the form. */
     function process_form($data) {
      $msg = "The form at {$_SERVER['PHP_SELF']} was submitted with these values: \n\n";
      foreach($data as $key=>$val) {
          $msg .= "$key => $val\n";
      }
            //print("The data from the form was: <br />");
            //var_dump($data);
            $from="My Form Example";
            $headers = "From:" . $from;
      mail("markl@cs.ucf.edu", "form submission", $msg, $headers);
            echo "Mail Sent";

    }
```


Big Form Example - Frontend - Windows Internet Explorer
http://www.cs.ucf.edu/courses/cis40
mail function in php example
File Edit View Favorites Tools Help
Google Search Sign In Convert Select
Favorites Suggested Sites Free Hotmail Web Slice Gallery KeepVid- Download and s...
Big ... Zim... PHP... B...
Page Safety Tools
Done Internet | Protected Mode: Off 100%

Mail Sent

Please complete the following form (* indicates required field)

User Input Form

* Your first name: Mark

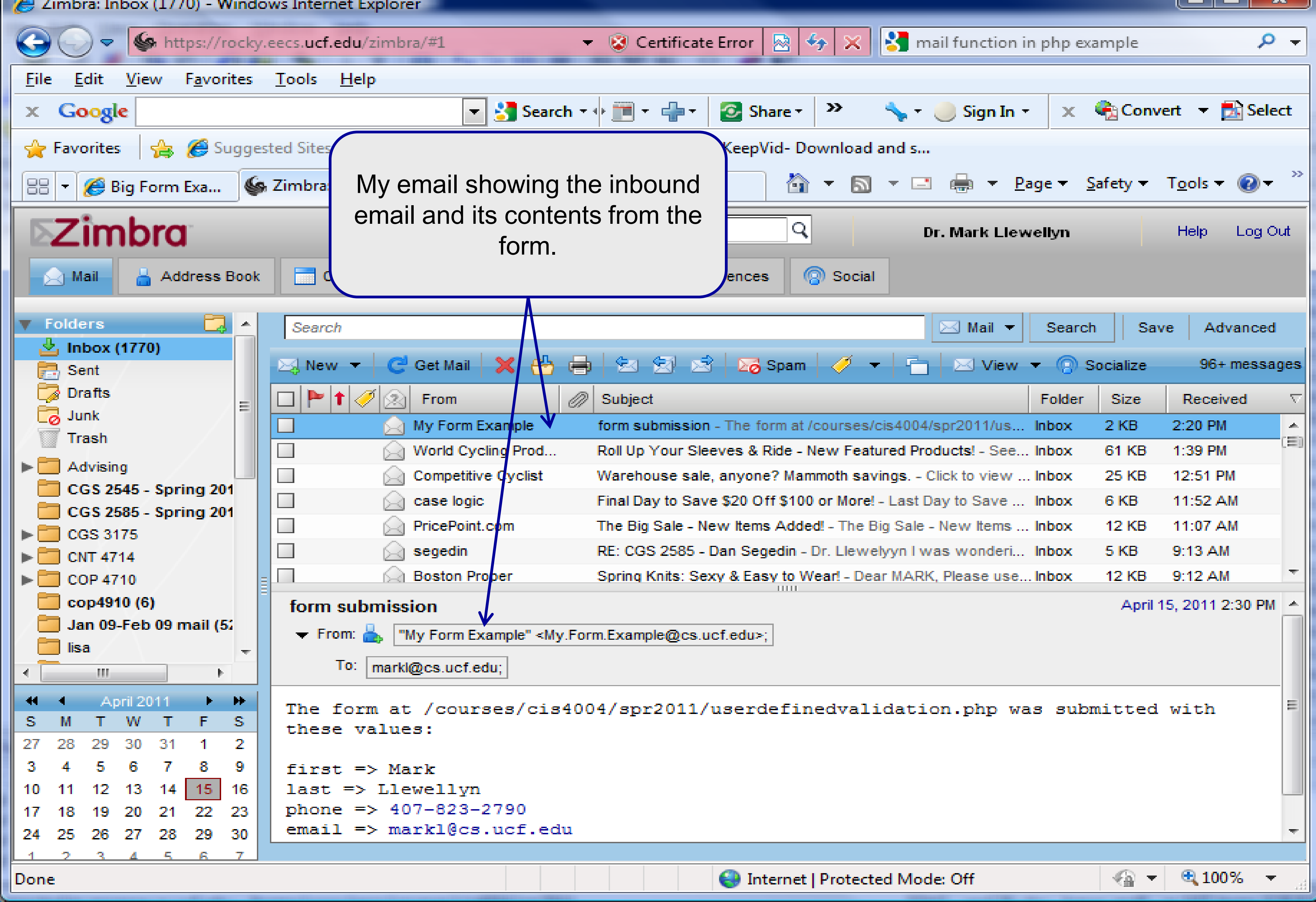
* Your last name: Llewellyn

Your phone number: 407-823-2790

* Your email address: markl@cs.ucf.edu

send

As you can see this is the user's browser after they filled in the form and clicked the Send button. In the upper left corner is the echoing of the message that the email was sent.



My email showing the inbound email and its contents from the form.

From	Subject	Folder	Size	Received
My Form Example	form submission - The form at /courses/cis4004/spr2011/us...	Inbox	2 KB	2:20 PM
World Cycling Prod...	Roll Up Your Sleeves & Ride - New Featured Products! - See...	Inbox	61 KB	1:39 PM
Competitive Cyclist	Warehouse sale, anyone? Mammoth savings. - Click to view ...	Inbox	25 KB	12:51 PM
case logic	Final Day to Save $20 Off $100 or More! - Last Day to Save ...	Inbox	6 KB	11:52 AM
PricePoint.com	The Big Sale - New Items Added! - The Big Sale - New Items ...	Inbox	12 KB	11:07 AM
segedin	RE: CGS 2585 - Dan Segedin - Dr. Llewelyyn I was wonderi...	Inbox	5 KB	9:13 AM
Boston Proper	Spring Knits: Sexy & Easy to Wear! - Dear MARK, Please use...	Inbox	12 KB	9:12 AM

form submission April 15, 2011 2:30 PM

From: "My Form Example" <My.Form.Example@cs.ucf.edu>;

To: markl@cs.ucf.edu;

```
The form at /courses/cis4004/spr2011/userdefinedvalidation.php was submitted with
these values:

first => Mark
last => Llewellyn
phone => 407-823-2790
email => markl@cs.ucf.edu
```