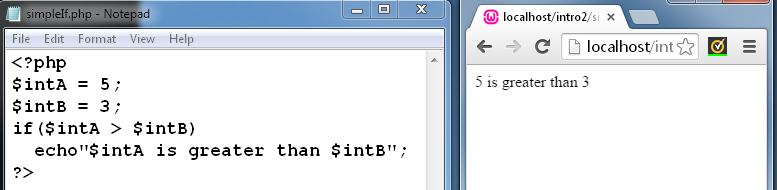
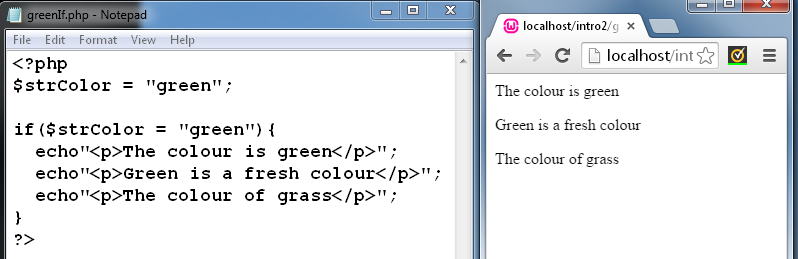
Flow of Control in PHP

# The if construct

This is the major construct in form validation.

**EXAMPLE 1:**

Using the brace characters we can set a number of statements relating to the condition.



Using the brace characters we can specify any number of statements that we wish to execute conditionally.

# The ‘else’ statement

Sometimes we want to express an alternative -when the if statement is false:

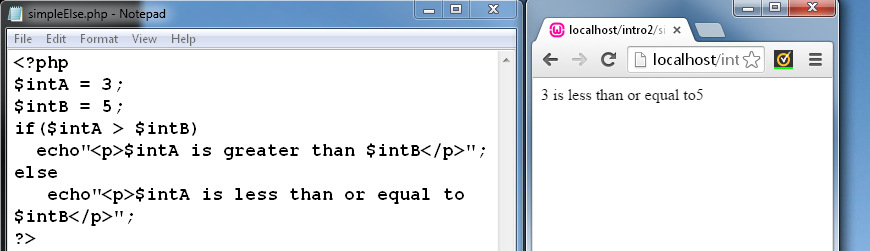
if(expression)

Statement to be executed if expression is true;

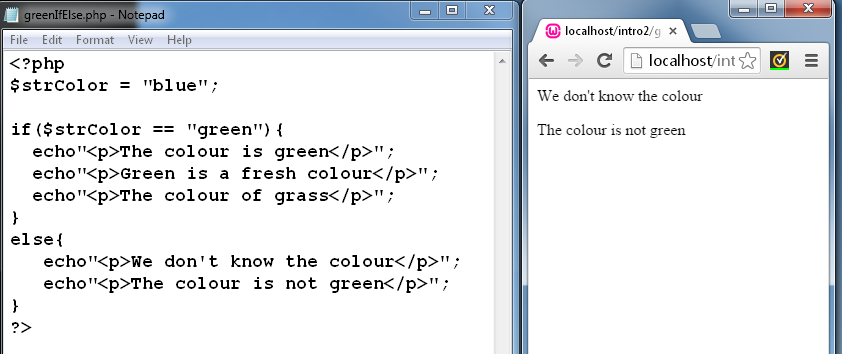
else

statement to be executed if expression is false;

**Simple example:**



**The ‘if-else’ construct employs braces to group together multiple statements for conditional execution.**



# The ‘elseIf’ statement

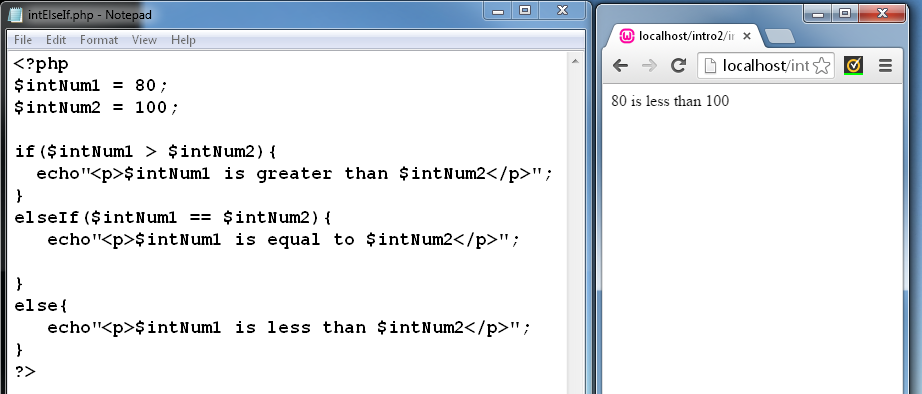
if(expression)

Statement to be executed if expression1 is true

elseIf(expression 2)

statement to be executed if expression2 is true

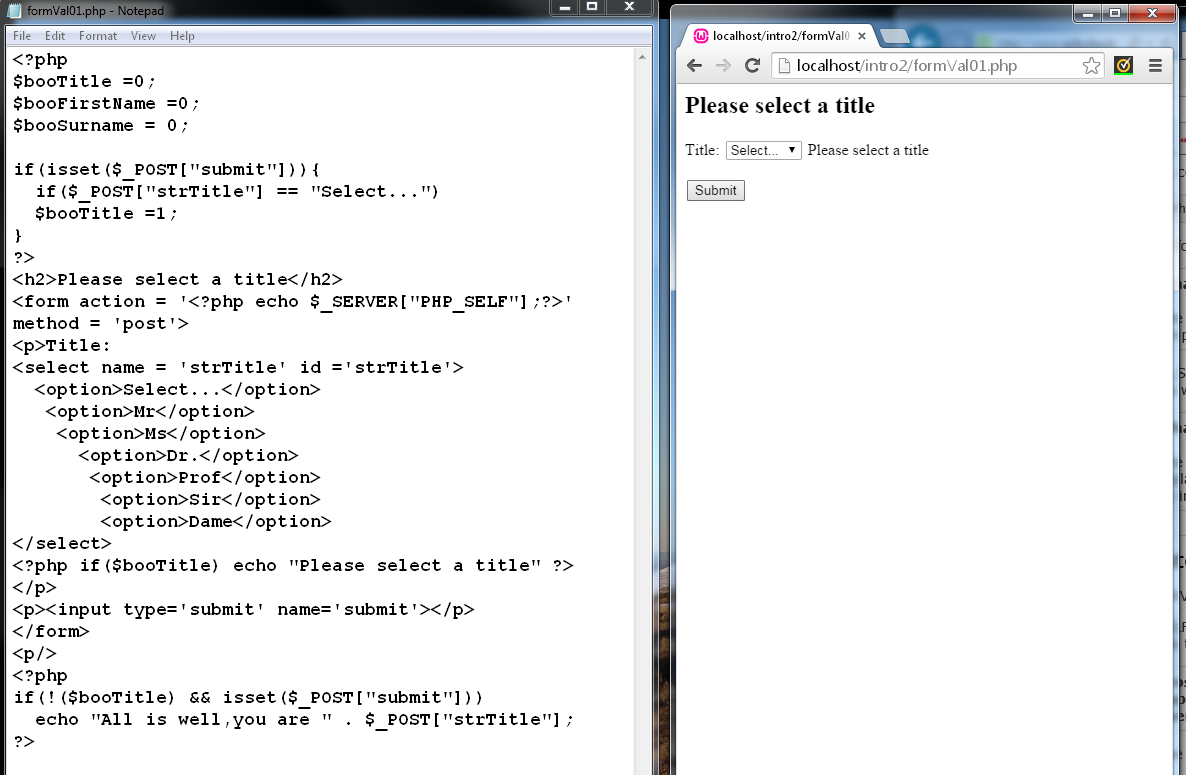
**Example 2:**



Simple Form Validation- using Boolean variables.

This example checks whether data has been entered or not. It works for text and select boxes using a PHP if statement.

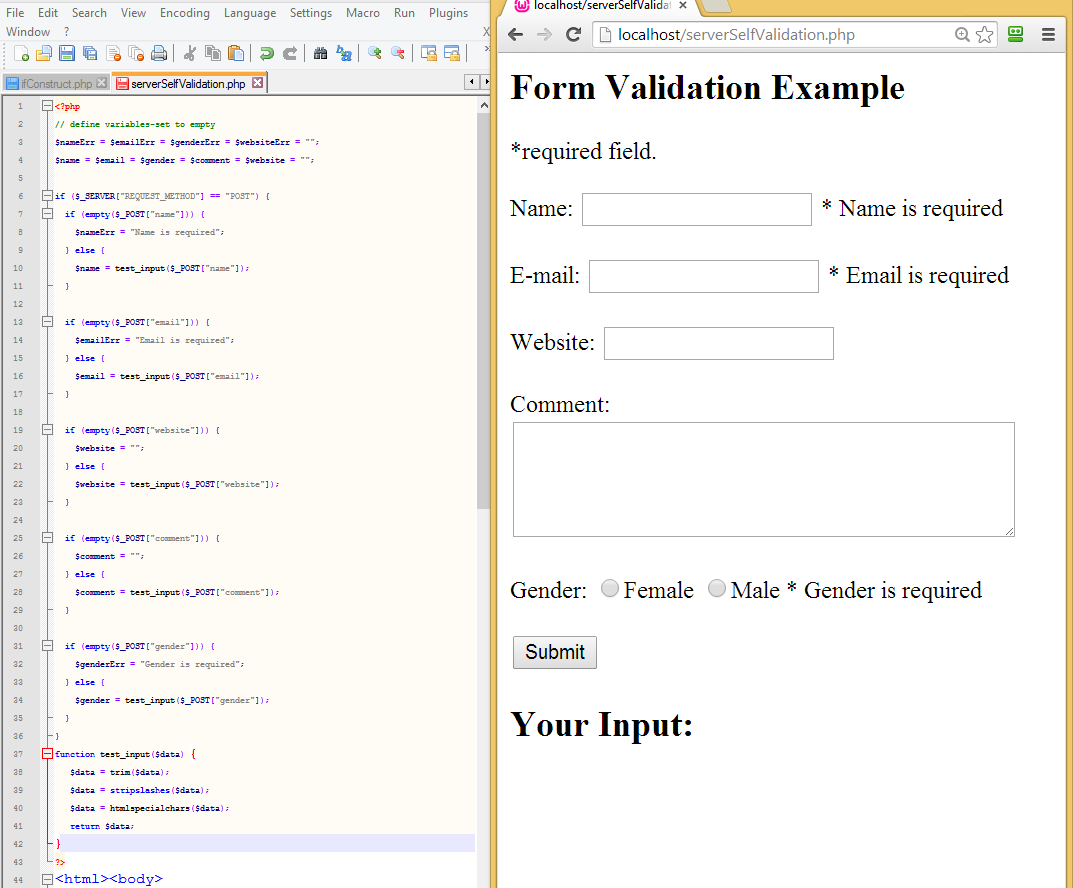
EXAMPLE 3:



Whole scripts, validated for presence/absence of data in fields

The PHP code at the top of the page includes If statements and the function to convert data to special characters.

EXAMPLE 4:

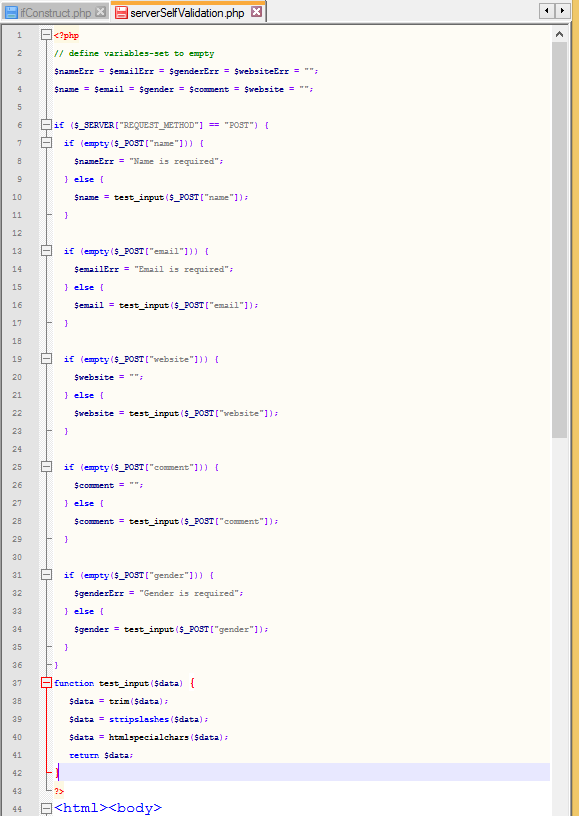


The HTML Section of the page starts below the PHP validation code.

Note <span> tags are included that indicate fields that are required with an asterisk.

The comment that “field is required” after validation is written to these tags via the PHP $Err variables that are present.

If there is no error, they remain empty. If there is an error, a message is loaded into the variable and displayed via the echo command.



Form Validation using $\_SERVER[“PHP\_SELF”] variable.

Some develops warn against using global variables, as they are more open to hackers.

If PHP\_SELF is used in your page then a user can enter a (/) and some Cross Site Scripting (XSS) commands to execute. This means it is a method that may attract hackers.

How To Avoid $\_SERVER["PHP\_SELF"] Exploits?

$\_SERVER["PHP\_SELF"] exploits can be avoided by using the htmlspecialchars() function.

The form code should look like this:

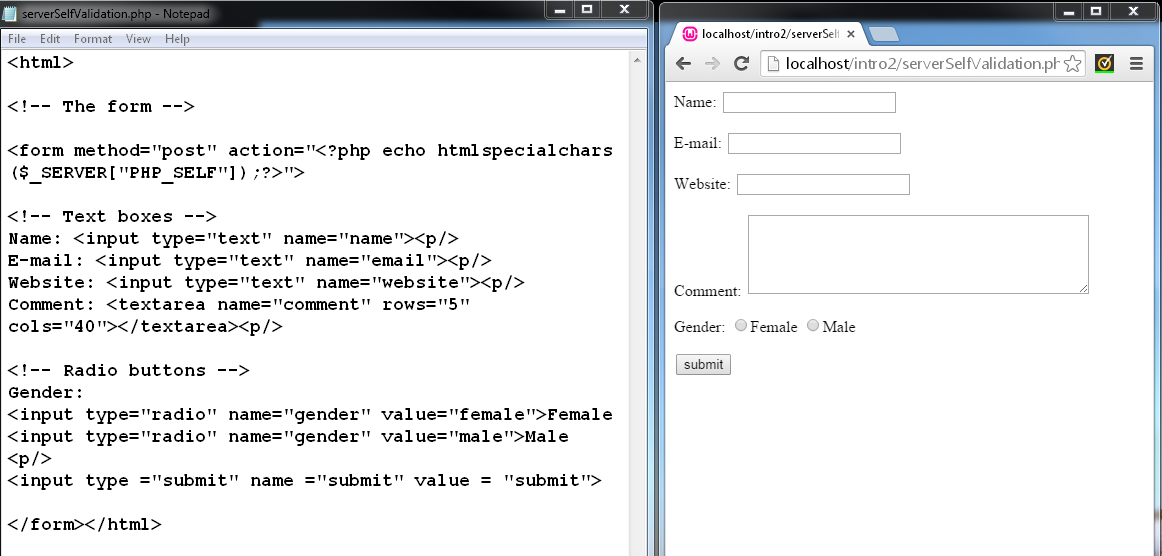
<form method="post" action="<?php echo htmlspecialchars($\_SERVER["PHP\_SELF"]);?>">

The htmlspecialchars() function converts special characters to HTML entities. Now if the user tries to exploit the PHP\_SELF variable, it will result in the following output:

<form method="post" action="test\_form.php/&quot;&gt;&lt;script&gt;alert('hacked')&lt;/script&gt;">

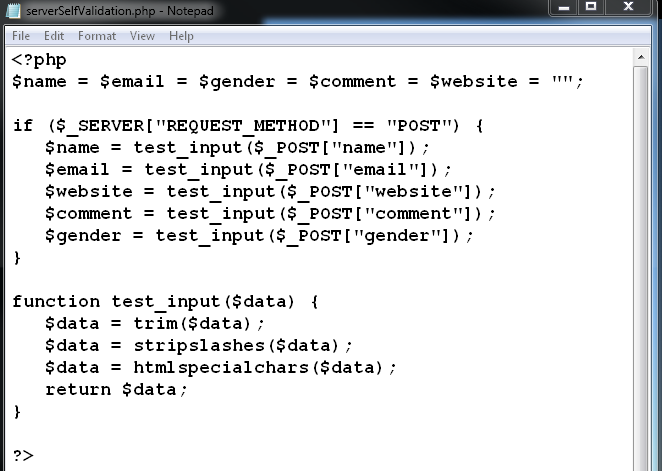
The exploit attempt fails, and no harm is done!

**Building the HTML Form**

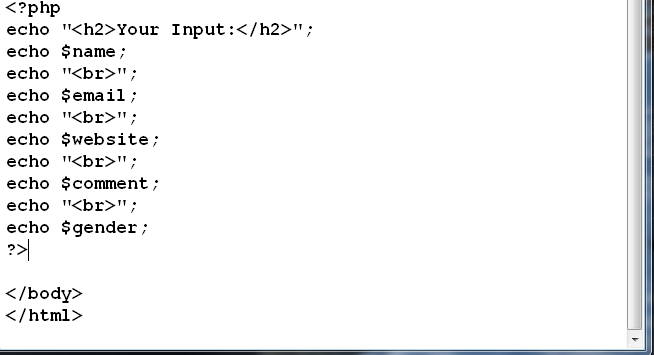


**Define variables and set their value to empty. Then write an if statement to populate the PHP variables with user data.**

**(The php code is at the top of the page.)**



Now you can call up the input and print it to the bottom of the page.

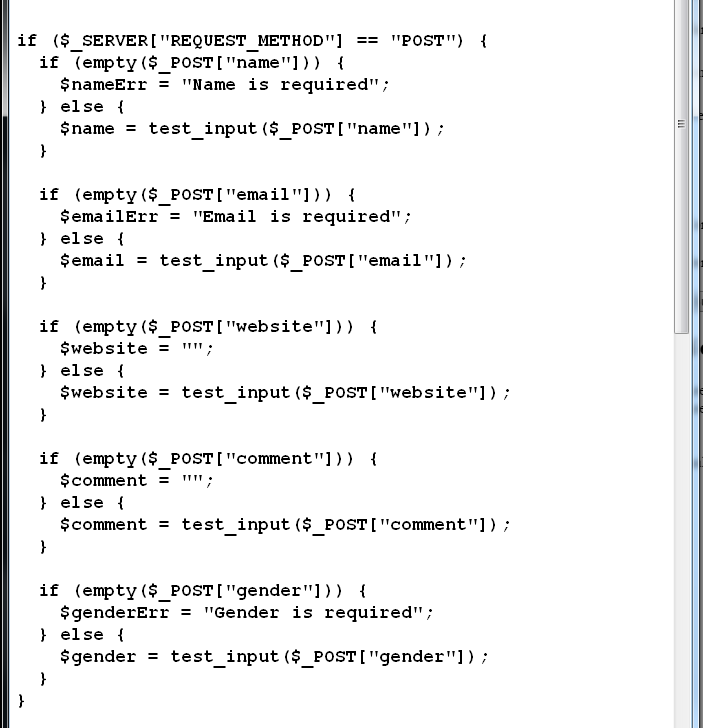


Your input should show up at the bottom of the form.

Now we set error variables (for when fields are empty)

$nameErr = $emailErr = $genderErr = $websiteErr = "";

We also expand our if statement to not only populating variables with filled out fields, but also generating error messages if they are empty.



**More sophisticated Validations**.

PHP - Validate Name

The code below shows a simple way to check if the name field only contains letters and whitespace. If the value of the name field is not valid, then store an error message:

$name = test\_input($\_POST["name"]);  
if (!preg\_match("/^[a-zA-Z ]\*$/",$name)) {  
  $nameErr = "Only letters and white space allowed";   
}

PHP - Validate E-mail

The easiest and safest way to check whether an email address is well-formed is to use PHP's filter\_var() function.

In the code below, if the e-mail address is not well-formed, then store an error message:

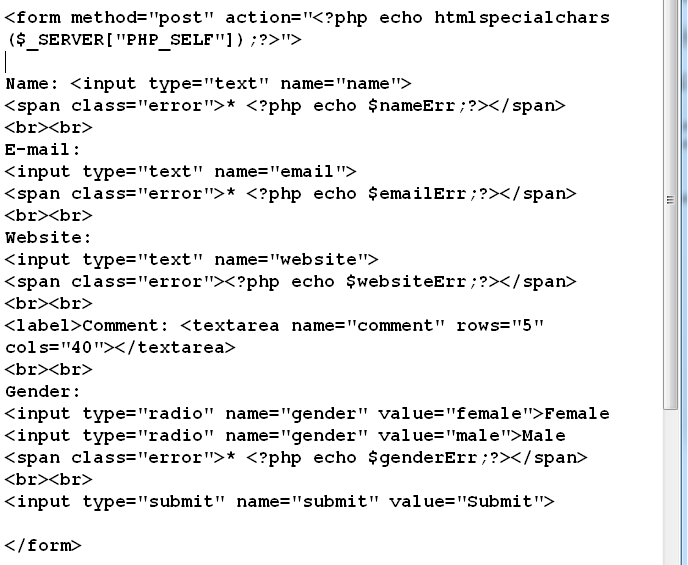
$email = test\_input($\_POST["email"]);  
if (!filter\_var($email, FILTER\_VALIDATE\_EMAIL)) {  
  $emailErr = "Invalid email format";   
}

PHP - Validate URL

The code below shows a way to check if a URL address syntax is valid (this regular expression also allows dashes in the URL). If the URL address syntax is not valid, then store an error message:

$website = test\_input($\_POST["website"]);  
if (!preg\_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-9+&@#\/%?=~\_|!:,.;]\*[-a-z0-9+&@#\/%=~\_|]/i",$website)) {  
  $websiteErr = "Invalid URL";   
}

In order to display error messages, we need to adjust the HTML code by adding some php error statements beside the form elements.



Now test your form by leaving some fields empty. You will see that asterisks are also placed inside the span tags beside those form elements that are required fields.

# Exercises:

1. Write a file that compares two bank balances using an else If statement. (EXAMPLE 1)

Use an appropriate variable type for your data. Return whether balance1 is greater than, equal to or less than balance 2.

Save as **BankBalance.php**

1. Use a PHP form to collect Title, First Name, Surname, Email, Preferred Colour, Size and Number of T-shirts ordered. (EXAMPLE 4)

Add an image of a T-shirt to the page as a banner image.

\*note for a more elegant email validation you will have to set the pattern-but Boolean is sufficient for this example.

Save as **PHP-formVal01.php -**

1. Build a User Form and Validation (like the one shown on page 4) that contains name, email and URL. Validate the name for letters and white space, the email for correct format and the URL for correct format. Call the file **PHP\_Validation02.php**