Form Handling Examples

Example1:

A program to convert values between Fahrenheit and Celsius

The formula used is *Fahrenheit = (9 / 5) * Celsius + 32*

---

ccc.html File

```
<html>
  <head>
    <title>Temperature Converter</title>
  </head>
  <body>
    <h3>Enter either Fahrenheit or Celsius and click on Convert</h3>
    <form name="cc" method="post" action="convert.php">
       <b>Fahrenheit</b>
       <input type="text" name="f" size="7"/><br/>
       <b>Celsius</b>
       <input type="text" name="c" size="7"/><br/>
       <input type="submit" value="convert"/><br/>
    </form>
  </body>
</html>
```

convert.php file

```
<?php

   $f = $c = "";

   $f = $_POST['f'];

   $c = $_POST['c'];

   if ($f != '')
    {
      $c = intval((5 / 9) * ($f - 32));
      $out = "$f °f equals $c °c";
          }

   elseif($c != '')
    {
      $f = intval((9 / 5) * ($c + 32));
      $out = "$c °c equals $f °f";
         }
   else $out = "";

echo "<h2>$out</h2>";?>
```

isset function

isset ($var); Determines if a variable is set and is not **NULL**. **Returns true** if set and **false** if not. Meaning that, it returns **TRUE** if **var** exists and has value, and returns **FALSE** if it doesn't have a value or has a **NULL** value.

Example2: convert2.php

```
<?php

 $f = $c = "";

 if (isset($_POST['f']))
   $f = $_POST['f'];

 if (isset($_POST['c']))
   $c = $_POST['c'];

   if ($f != '')
    {
      $c = intval((5 / 9) * ($f - 32));
      $out = "$f °f equals $c °c";
          }

   elseif($c != '')
    {
      $f = intval((9 / 5) * ($c + 32));
      $out = "$c °c equals $f °f";
         }
   else $out = "";

echo $out; ?>

 <html>
  <head>
    <title>Temperature Converter</title>
  </head>
 <body>
    <form name="cc" method="post" action="convert2.php">
       <b>Fahrenheit</b>
       <input type="text" name="f" size="7" value=""/><br/>
       <b>Celsius</b>
       <input type="text" name="c" size="7" value=""/><br/>
       <input type="submit" value="convert"/><br/>
    </form>

  </body>
</html>
```

Example3:

```
<?php

 $f = $c = "";

 if (isset($_POST['f']))
   $f = $_POST['f'];

 if (isset($_POST['c']))
   $c = $_POST['c'];

 if ($f != '')
   {
     $c = intval((5 / 9) * ($f - 32));
     $out = "$f °f equals $c °c";
          }

 elseif($c != '')
    {
      $f = intval((9 / 5) * ($c + 32));
      $out = "$c °c equals $f °f";
         }
 else $out = "";

echo $out;?>

 <html>
  <head>
    <title>Temperature Converter</title>
  </head>
 <body>
<form name="cc" method="post"
                action="<?php echo $_SERVER["PHP_SELF"];?>">
       <b>Fahrenheit</b>
       <input type="text" name="f" size="7" value=""/><br/>
       <b>Celsius</b>
       <input type="text" name="c" size="7" value=""/><br/>
       <input type="submit" value="convert"/><br/>
    </form>


  </body>
</html>
```

Example4:

How to handle data from select menu, when using multiple selections.
In this example, suppose the user select the choices: **one, three and five.**

ss.html

```
<form action=”mypage.php” method=”POST”>
<SELECT multiple name="options">
<option value=’1’>One</option>
<option value=’2’>Two</option>
<option value=’3’>Three</option>
<option value=’4’>Four</option>
<option value=’5’>Five</option>
</select>
<input type=”submit” name=”Submit” value="Send” >
</form>
```

If you were to attempt to acces the values in mypage.php like so:

CODE 1

```
echo $_POST['options'];
```

Or by using **CODE 2**

```
$myoptions=$_POST['options'];

echo $myoptions[0];
echo $myoptions[1];
echo $myoptions[2];
etc...
```

The output would be: 5

This effectively **overwrites** the previous value that was assigned to $options.

In order to get every option that was selected you need to **add a pair of brackets** at the end of the Select box's name like so:

CODE

```
...
<SELECT multiple name="options[]">
...
```

The brackets tell PHP that the submitted data is in actuality an Array.
So instead of overwriting the values that are sent, they are added to subsequent positions in the array.
options[]=1
options[]=3
options[]=5
etc...
This will now allow us to access every value that was selected, from the array.

CODE

```
$myoptions=$_POST['options'];
```

```
Then:
```

```
echo $myoptions[0];
echo $myoptions[1];
echo $myoptions[2];
etc...
```

Would echo out as:

1
3
5
etc...

Use count function:

Should we need to find out how many options were selected we could issue a count() function on our array:

CODE

```
$selecteditems=count($myoptions);
```