

# Chapter 3

## PHP Basics

### 3.1 Using PHP Script Tags

This section is on using PHP script tags appropriately in your PHP program.

The beginning PHP tag (`<?php`) has an ending tag (`?>`) and the code in between those tags is a code block or a PHP block.

You can open and close these as many of these code blocks as you want in a webpage.

I am coding regular HTML, I have to make sure that it is not in one of these PHP blocks or it won't behave very well at all.

If I include regular HTML statements in a PHP code block I get an error "unexpected angle bracket."

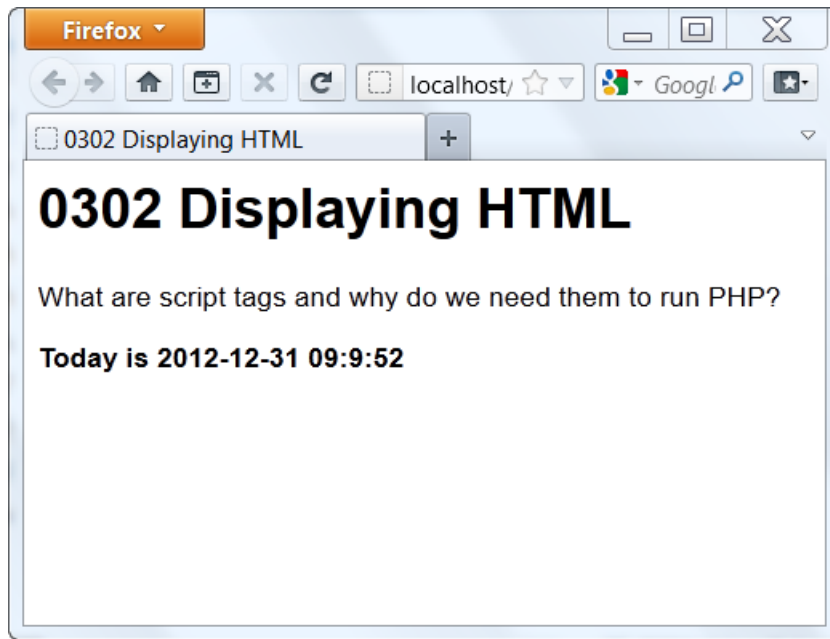
You can't run HTML inside of a PHP block, not by itself. You may generate HTML code from PHP statements like **print** or **echo**. For example...

```
print "<p>This is a paragraph</p>";
```

It's important to let HTML be HTML and let PHP be PHP. They have to appear in the correct places, PHP in the PHP blocks, and HTML anywhere outside of them. It will be interpreted correctly.

### 3.2 Displaying HTML

Below I am displaying a very simple page...



---

Figure 3.2.1 – Simple Web Page

It has a heading and some text. The last line is telling us what is today's date along with the current time.

Now if I want to see the time change, I can just reload the page.

Let's take a look at the PHP code involved in getting this to happen...

```
<?php
print "<html>\n";
print "<head><title>0302 Displaying HTML</title>\n";
print "<style>body {font-family: Arial, Helvetica,
      sans-serif; font-size: 16px;}</style>\n";
print "\n ";
print "<body>\n";
print "    <h1>0302 Displaying HTML</h1>\n";
print "    <p>What are script tags and why do
      we need them to run PHP?</p>\n";
print "\n ";

$today = date('Y-m-d h:g:s');

print "<p><b>Today is $today </b></p>\n";
```

```
print "</body>\n";

print "</html>\n";
?>
```

---

Figure 3.2.2 – Simple Web Page – Using too much PHP code

In the code above, I've done something rather extreme. I have all of the HTML coming out of PHP print statements.

That's not really a good way to do it, but I just wanted to show you that you could and that PHP is basically just writing out HTML to your browser.

You never see PHP code when we do a View-Page-Source, you will just see the resultant HTML code.

The only code I really had to put in a PHP block was the **date** function.

So the code should really look like this...

```
<html>
<head><title>0302 Displaying HTML</title>
<style>body {font-family: Arial, Helvetica, sans-serif;
font-size: 16px;}</style>

<body>
  <h1>0302 Displaying HTML</h1>
  <p>What are script tags and why do
    we need them to run PHP?</p>
  <p><b>

<?php
  $today = date('Y-m-d h:g:s');
  print "<p><b>Today is $today</b></p>";
?>

</b></p>
</body>

</html>
```

## Figure 3.2.3 – Simple Web Page – Using the correct amount of PHP code

Now please don't use the **font** tag (e.g. `<font color="blue">`). The font tag has been deprecated. And the current way to make colors change of tag is to use a **style** tag.

I'm going to use something called an **inline style**. If I had a paragraph like so...

```
<p>Hi there, Steve</p>
```

... and I wanted to turn all the text blue I can apply a style to this paragraph...

```
<p style="color: blue;">Hi there, Steve</p>
```

... and it would display something like this...

Hi there, Steve

When I want to affect a bit of text (within a paragraph, for example) I have to wrap it in a special tag called a **span** tag. So if I did this...

```
<p>Hi <span style="color: blue;">there</span>, Steve</p>
```

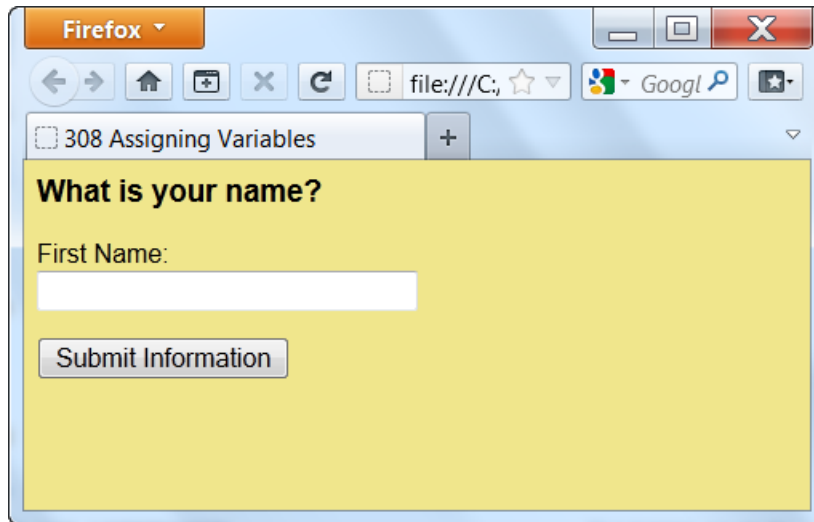
... it would display something like this...

Hi there, Steve

### 3.3 HTML Forms

HTML forms are a way that we can gather user information.

Users of the web page can enter data, click a button and that will call the PHP program that will use that information and process it somehow and return a new Web page back to the browser.

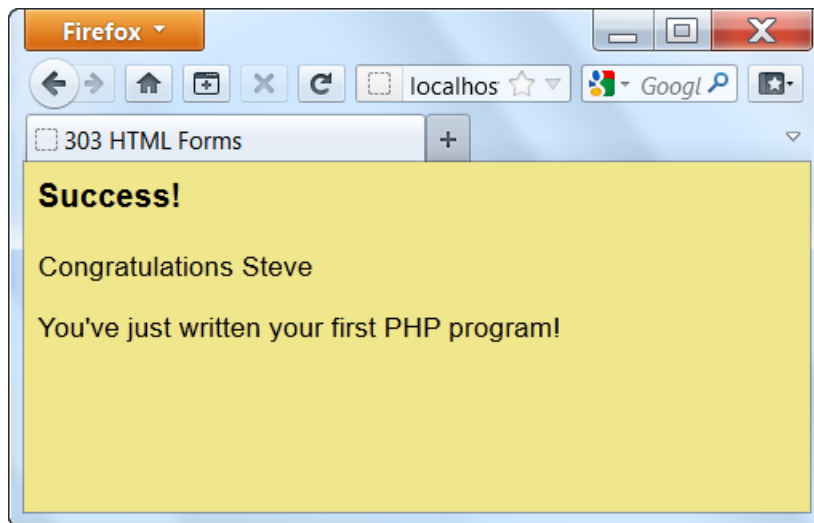


---

Figure 3.3.1 – Simple HTML form

It gives the user a place that enter their name and a button to click.

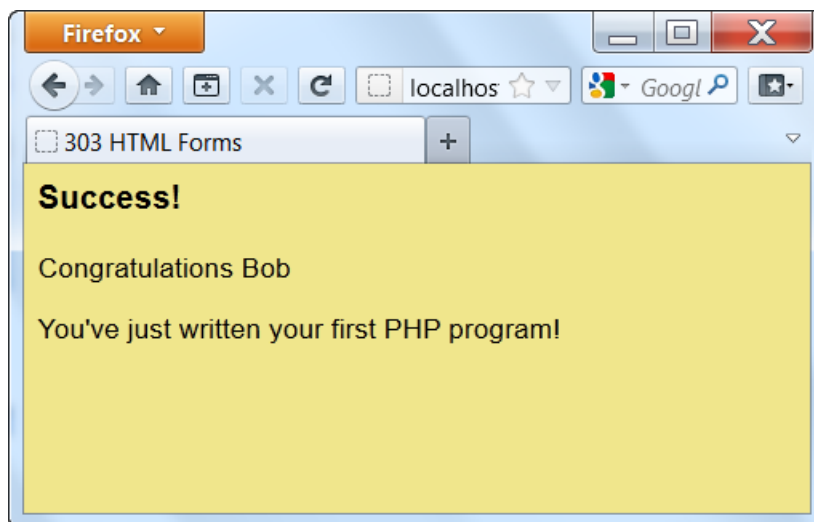
If entered my name, Steve, and clicked the submit information button, it would display like so...



---

Figure 3.3.2 – Response when user entered "Steve"

What if my name wasn't Steve and I was Bob, I wouldn't want to see, "Congratulations Steve", I'd want to see "Congratulations Bob". So let's try that. I'll enter "Bob" and hit the button and there we are...



---

Figure 3.3.2 – Response when user entered "Bob"

Here's code for the HTML file itself...

```
<html>
<head>
    <title>308 Assigning Variables</title>
    <link rel="stylesheet" type="text/css"
href="css/basic.css" />
</head>

<body>

<h3>What is your name?</h3>

<form method="post" action="0303_HTML_Forms.php">

    <p>
        First Name:<br />
        <input type="text" name="firstname" size="30" />
    </p>

    <p>
        <input type="submit" value="Submit Information" />
    </p>

</form>

</body>
</html>
```

---

Figure 3.3.3 – HTML code for a form

The thing that's different from what we've seen before is the beginning and ending **form** tag (<form> and </form>.)

All the elements that are going to be processed on an **HTML form** must be placed inside these two tags.

In fact, it's a good idea to put the ending **form** tag in right after you code the beginning **form** tag... just to make sure you don't forget it.

An **input** tag is used to allow the user to enter data, but it's a special type of **input** tag.

To make this **input** tag display a single line text edit field, the **type** attribute must have a value of "text".

Another attribute I need to set is the **name** attribute. In this example, I gave it a value of "firstname" to describe what kind of data should be entered there. The PHP program that I will call to process this data will need to use "firstname" to find the text the user entered.

Another input tag you need defines the **submit button**. When you make the type equal to "submit" a **submit button** appears on the page.

The **submit button** is a very special button in that a program is invoked when you click it. It knows what program to run by using whatever program is specified for the **action** attribute in the **form** tag. E.g.

```
<form method="post" action="0303_HTML_Forms.php">
```

NOTE: Always make the value of the **method** attribute be equal to "post".

Here is the PHP code for 0303\_HTML\_Forms.php...

```
<html>
<head>
    <title>303 HTML Forms</title>
    <link rel="stylesheet" type="text/css"
        href="css/basic.css" />
</head>

<body>

    <h3>Success!</h3>
```



```
<?php
    $firstname = $_POST['firstname'];
    print "<p>Congratulations $firstname</p>";
?>

<p>You've just written your first PHP program!</p>

</body>
</html>
```

---

Figure 3.3.3 – PHP coded for 0303\_HTML\_Forms.php

There are just two lines of PHP code. The first...

```
$firstname = $_POST['firstname'];
```

... Uses the **\$\_POST** function to retrieve the value entered by the user on the Web page and returns it into a PHP variable named \$firstname.

NOTE: (For you advanced programmers out there.) Technically, the **\$\_POST** is not a function, instead it is an array passed to the PHP program by HTTP request, and we are passing the 'key' to this associative array to retrieve its associated 'value'. If you didn't understand that, don't worry, you can think of the **\$\_POST** as a function and it will work just fine.

The second line of PHP code is ...

```
print "<p>Congratulations $firstname</p>";
```

... which is just printing out a paragraph. Since the \$firstname is enclosed within double quotes (along with the other text), PHP will automatically change \$firstname to whatever value is stored in it (e.g. "Steve" or "Bob".) The term for that is **variable interpolation**... for those of you getting ready for a job interview!

### 3.4 Assigning Values

Please view videos on class Web site for information on this topic.

### 3.5 Appending Text

Please view videos on class Web site for information on this topic.

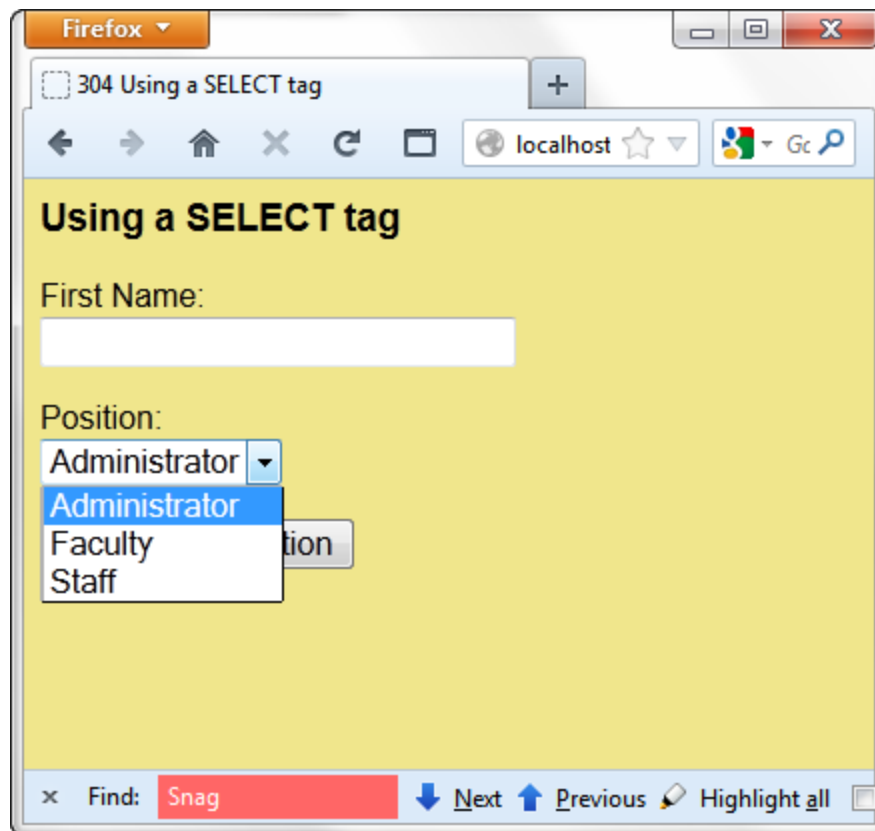
### 3.6 Using the Select Tag

Here you will learn how to use a **select** tag in an HTML form and process it using PHP.

A **select** tag is essentially a drop-down list of different choices we might make

In the example below you would have the choice of "administrator", "faculty", or "staff", for the position that someone held.

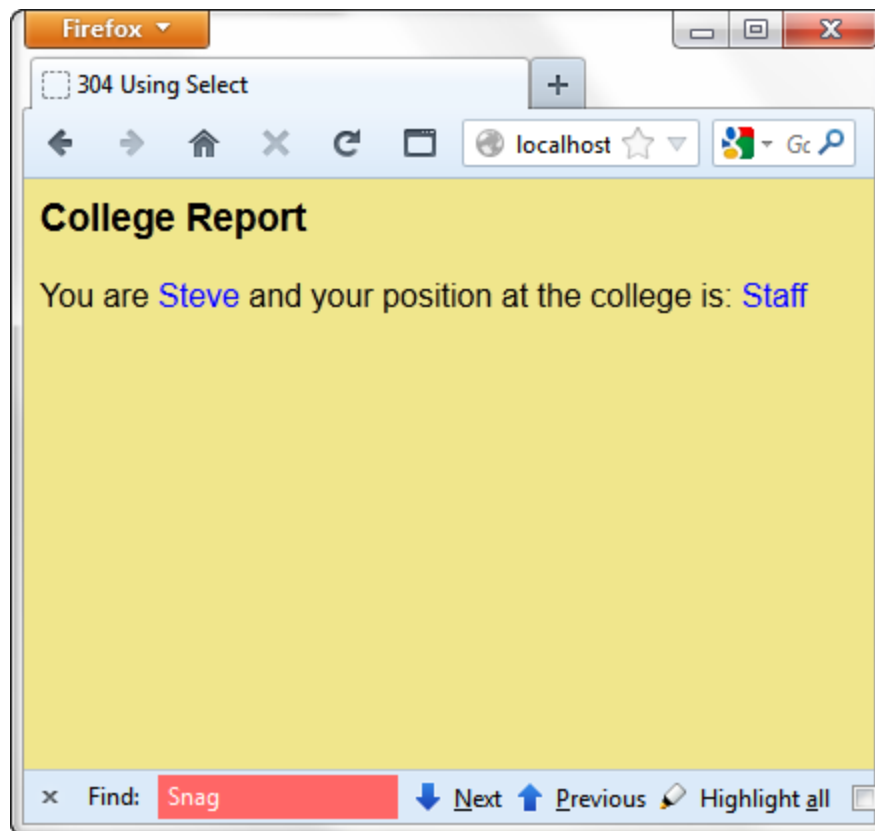
...



---

Figure 3.6.1 – Web page with SELECT-OPTION list

If I enter 'Steve' for my name and select 'staff', then click the submit information, it displays the following...



---

Figure 3.6.2 – Web page response to SELECT-OPTION list

Here's code for the HTMLfile ...

```
<html>
<head>
    <title>305 Using a SELECT tag</title>
    <link rel="stylesheet" type="text/css"
        href="css/basic_2.css" />
</head>

<body>

<h3>Using a SELECT tag</h3>
```

```
<form method="post" action="0304_Using_Select.php">

    <p>
    First Name:<br />
    <input type="text" name="firstname" size="30">
    </p>

    <p>
    Position:<br />
    <bselect name="position" size="1">
        <boption value="Administrator">
            Administrator
        </option>
        <boption value="Faculty">Faculty</option>
        <boption value="Staff">Staff</option>
    </select>
    </p>

    <p>
    <input type="submit" value="Submit Information">
    </p>

</form>

</body>
</html>
```

---

Figure 3.6.3 – HTML code demonstrating a SELECT-OPTION list

The bolded text above shows a complete **select-option** list. In the opening **select** tag you concede that there are two attributes. The first one is the **name** attribute where you should specify the name of this tag so that it can be referenced in the PHP program that is called by the form. The next attribute is the **size**. The size attribute is used to specify how many of the options in the **select-option** list should appear at any one time. In general, we choose "1".

Below the opening **select** tag are specified three options. They are enclosed in the **option** tags. Each **option** tag needs to specify a **value** attribute which will be used by the PHP program depending on which option is selected by the user. The text that appears between the opening **option** tag and the ending **option** tag is what the user

actually sees him a list. The value specified for the **value** attribute does not have to equal the text that is shown to the user. I made them equal in this example, however, I did not have to. Remember, that the PHP program will use the value specified in the **value** attribute, not the text that the user sees.

NOTE: the options will be displayed in the order you list them. If you want to change that, you can put **selected=selected** on the option you wish to appear by default. For example...

```
<option selected=selected value="Faculty">Faculty</option>
```

Remember that you must end your **select-option** list with an ending **select** tag (e.g `</select>`)

Let's look at the PHP program code below...

```
<body>

<h3>College Report</h3>

<?php
    $firstname = $_POST['firstname'];
    $position = $_POST['position'];

    print "<p>You are <span class='textblue'>
        $firstname</span> and ";
    print "your position at the college is:
        <span class='textblue'> $position </span></p>";
?>

</body>
```

---

Figure 3.6.4 – PHP code demonstrating response to a SELECT-OPTION list

In the above code option list you see that...

```
$position = $_POST['position'];
```

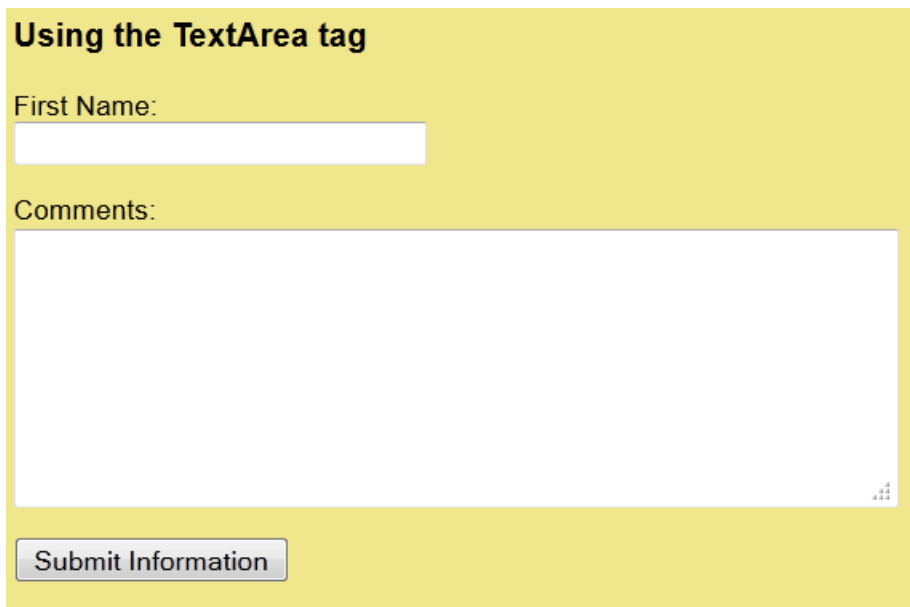
... The `$_POST` is bringing back the value selected in the **select-option** list that was named "position" and placing it in a variable named `$position`.

So if the user had chosen "staff" in the **select-option** list `$position` would be equal to "staff".

### 3.7 Using the TextArea Tag

The text area tag allows a user to enter more than one line of information in a form element and then we can process it with PHP.

Here is an example form with a Textarea...



**Using the TextArea tag**

First Name:

Comments:

---

Figure 3.7.1 – Web page with TextArea

This is the HTML code below...

```
<html>
<head>
    <title>305 Using the TextArea tag</title>
    <link rel="stylesheet" type="text/css"
href="css/basic_2.css" />
</head>

<body>

<h3>Using the TextArea tag</h3>

<form method="post" action="0305_Using_TextArea.php">
    <p>
        First Name:<br />
        <input type="text" name="firstname" size="30">
    </p>

    <p>
        Comments:<br />
        <textarea name="comments" rows="7" cols="50">
    </textarea>
    </p>

    <p>
        <input type="submit" value="Submit Information">
    </p>

</form>

</body>
</html>
```

---

Figure 3.7.2 – HTML code demonstrating a TextArea



This is a little bit different than what we've seen before because the **value** is not specified in the text area tag as an attribute. The **value** is whatever text is found between the `<textarea>` and `</textarea>` tags.

In the example above, it is blank because I want the **textarea** to be empty when it first opens up so I can enter some data.

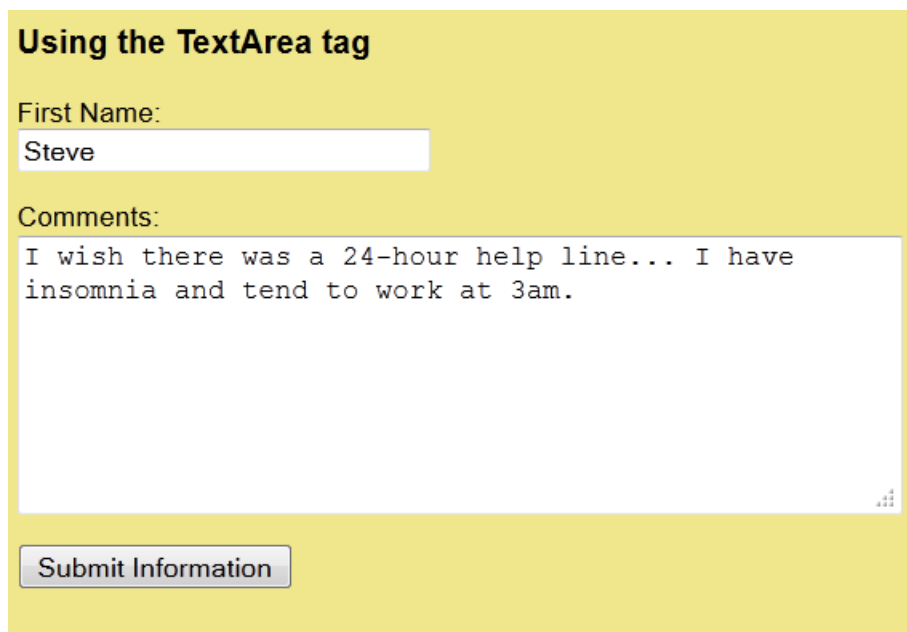
Let's look at the attributes specified on the **textarea** tag.

First there is the **name** attribute which I gave a value of "comments".

Next there are the **rows** and **cols** attributes. The value of the **rows** attribute indicates how many lines of text will display in the textarea. If you add more rows than indicated in the value of the **rows** attribute then a **vertical scrollbar** is automatically generated that allows the user to scroll through all of the text specified.

The value of the **cols** attribute indicates the maximum number of characters can appear on any row.

Here is an example of data filled in the form, just before I click the submit button which will send the form to the PHP program on the server.



**Using the TextArea tag**

First Name:  
Steve

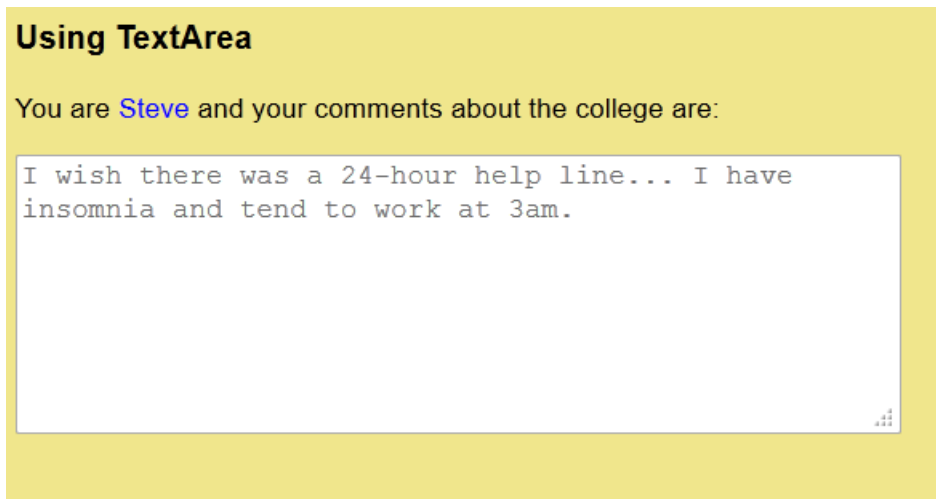
Comments:  
I wish there was a 24-hour help line... I have insomnia and tend to work at 3am.

Submit Information

---

Figure 3.7.3 – HTML Web page with TextArea

After I click the submit button the page returned from the server will look like this...



---

Figure 3.7.4 – HTML Web page returned from server by PHP program

Notice that the text in the **textarea** is grayed out and that there is no submit button.

The text is grayed automatically because I specified...

```
disabled='disabled'
```

... in the `<textarea>` tag.

Look at the PHP code below...

```
<body>

<h3>Using TextArea</h3>

<?php
    $firstname = $_POST['firstname'];
    $comments = $_POST['comments'];
    print "<p>You are <span class='textblue'>
        $firstname</span> and ";
    print "your comments about the college are: </p>";
```

```
        print "<textarea name='comments' rows='7' cols='50'
disabled='disabled' class='textdisabled'>";
        print $comments;
        print "</textarea>";

?>
</body>
```

---

Figure 3.7.6 – PHP code demonstrating response to a textarea

Here is the CSS code ....

```
html {margin:0; padding:0;}

body {
    font-family: Arial, Helvetica, sans-serif;
    color: black;
    background-color: #F0E68C;
}

p {
    font-size: 16px;
}

input {
    font-size: 16px;
}

select {
    font-size: 16px;
}

textarea {
    font-size: 16px;
}

.textblue {
    color: blue;
}
```

---


Figure 3.7.7 – CSS code

Notice that, in the **print** statement, I have created the **textarea** tag with the disabled attribute set and have placed the text the user entered between the <textarea> and </textarea> tags.

### 3.8 Using Radiobuttons

**Radiobuttons** are those little circular buttons that are displayed on a web page that you use to make choice. The unique characteristic of a **radiobutton** is that only one **radiobutton** in a **radiobutton group** can be selected at a time.

In the example below, if I click the part-time radiobutton, the full-time radiobutton is automatically deselected.



Using the RadioButtons

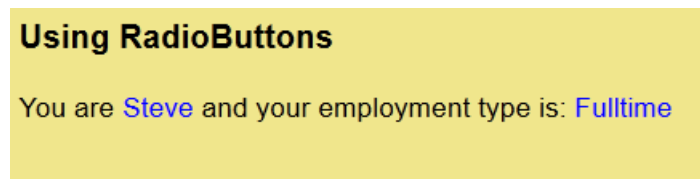
First Name:

Employee Type: ☒ Fulltime ☐ Parttime

---

Figure 3.8.1 – Web page with a radiobutton group

If I click the submit button, it will look like this...



Using RadioButtons

You are Steve and your employment type is: Fulltime

---

Figure 3.8.2 – Web page response to a radiobutton choice

Let's take look at the HTML code...

```
<body>

<h3>Using the RadioButtons</h3>

<form method="post" action="0306_Using_RadioButtons.php">

    <p>
    First Name:<br />
        <input type="text" name="firstname" size="30">
    </p>

    <p>
    Employee Type:
    <input type="radio" name="etype" value="Fulltime"
        checked="checked"> Fulltime
    <input type="radio" name="etype" value="Parttime">
        Parttime
    </p>

    <p>
        <input type="submit" value="Submit Information">
    </p>

</form>
```

---

Figure 3.8.3 – HTML code for radiobutton group

Notice that I give each radiobutton the same exact value for their **name** attribute, "etype". This will enforce the behavior that only allows one **radiobutton** to be selected at a time

I give each radiobutton the same value as what the user sees... but I didn't have to. This behaves the same way as the **option** list on the **select** tag. PHP will use the value specified on the **value** tag for its processing.

Let's look at the PHP code that processes this form...

```
<body>

<h3>Using RadioButtons</h3>

<?php
    $firstname = $_POST['firstname'];
    $etype = $_POST['etype'];

    print "<p>You are <span class='textblue'>
        $firstname</span> and ";
    print "your employment type is: ";
    print "<span class='textblue'> $etype </span></p>";
?>

</body>
```

---

Figure 3.8.4 – PHP code that processes a radiobutton group

## 3.9 Using Checkboxes

Please view videos on class Web site for information on this topic.

## 3.10 Debugging Techniques

Please view videos on class Web site for information on this topic.