



BGSU Forms

WEB DOCUMENT
WEBDO1

Updated 6/20/2007

This document discusses how to use BGSU Forms, Version 3, a web application that e-mails the contents of a web form to a specified user.

This document should be used by BGSU faculty, staff, and students who want to add interactive forms to their web documents.

This document contains

- an introduction to BGSU Forms
- a Quick Tour of BGSU Forms
- instructions for using BGSU Forms to collect and process input data
- a definition of special fields in BGSU Forms
- instructions for testing and implementing a BGSU Form
- a description of various HTML form input types
- examples how to use JavaScript to enhance forms

The conventions used in this document are

- Bold Title Case** = an action to be taken by the reader
- underline = name of an object on the screen

Additional sources include

- HTML
 - <http://www.htmlgoodies.com/primers/html/>
- HTML Goodies - Form Tutorials
 - <http://www.htmlgoodies.com/tutorials/forms/article.php/3479121>
- JavaScript
 - <http://www.w3schools.com/js/default.asp>
- JavaScript Examples
 - <http://www.developer.com/lang/jscript/>
- BGSU Forms Examples
 - <http://www.bgsu.edu/its/tsc/self-help/page13216.html>
- This document
 - <http://www.bgsu.edu/downloads/cio/file9322.pdf>

Table of Contents

1.0 Introduction	3
2.0 Quick Tour	3
2.1 Quick Tour Example 1	3
2.2 Quick Tour Example 2	4
2.3 Quick Tour Example 3	6
3.0 Required Tags	8
3.1 Form Tag	8
3.2 mailTo Address	8
3.3 mailFrom Address	9
3.4 Send Button	9
3.5 Naming Items in a Form	10
4.0 Other Tags	10
4.1 mailCc Address	10
4.2 mailSubject Field	10
4.3 suppressEmptyFields Field	11
4.4 redirectURL Field	11
4.5 Reset Button	11
5.0 HTML Form Elements	12
5.1 Text Field	12
5.2 Text Area	12
5.3 Radio Buttons	13
5.4 Checkboxes	14
5.5 Pop-up Menu or Scrollable List	15
6.0 JavaScript and Catching Errors	16
6.1 Script Tag	16
6.2 Additions to Form Tag	17
6.3 checkForm Function	17
6.3.1 Blank Text Field	17
6.3.2 Entry Too Long	18
6.3.3 Invalid E-mail Address	18
7.0 Finishing the Form	18
8.0 The Form Submission Procedure	19
8.1 Why do we need security code on BGSU Forms?	20
9.0 Additional Help	20

Table of Figures

Figure 2.1 Quick Tour Example 1	3
Figure 2.2 Quick Tour Example 2	5
Figure 2.3 Quick Tour Example 3	8
Figure 3.4 Submit Button	10
Figure 4.4 Reset Button	12
Figure 5.1 Text Box	12
Figure 5.2 Text Area Box	13
Figure 5.3 Radio Buttons	13
Figure 5.4 Checkboxes	14
Figure 5.6 Pop-up Menu	15
Figure 5.7 Activated Pop-up Menu	15
Figure 5.8 Scrollable List	16
Figure 8.1 Forms Submission with Security Code	19

1.0 Introduction

Many web developers want to add interactivity to their pages by allowing users to submit responses to a form. The information gathered from a form may serve a variety of purposes from simple responses to statistical analysis. Such information is submitted via e-mail by the click of a button when a form is completed.

BGSU Forms is a tool written exclusively for use with BGSU's web servers. This tool is easy to use: no knowledge of server-side scripting is required. Client-side JavaScript may be added to the form for data validation if desired. To use BGSU Forms, the web developer simply develops an HTML form according to the specifications given in this document.

This document describes the process of creating a form along with specific requirements for using BGSU Forms. Therefore, the information contained below may, in part, pertain to forms in general. However, some of it is directly applicable to BGSU Forms.

For examples of BGSU Forms using all of the input types described in this document and the HTML code used to generate them, see URL:

<http://www.bgsu.edu/its/tsc/self-help/page13216.html>

We encourage you to browse these examples to learn more about BGSU Forms.

2.0 Quick Tour

This section contains a Quick Tour of BGSU Forms. We present three carefully graded examples. Each example lists the code needed to produce the HTML form. The form itself is also shown. The last two examples incorporate JavaScript, but some of the JavaScript code may have been deleted from the listing for brevity.

2.1 Quick Tour Example 1

The following HTML document is perhaps the simplest example of BGSU Forms:

```
<html>

<head>
  <title>Web Mail #1</title>
</head>
<body>
  <!-- Replace "test3" with "submit3" to activate this form -->
  <form method="post"
    action="http://webapps.bgsu.edu/cgi-bin/bgsuform/test3">
    <h1>BGSU Web Mail</h1>
    <p>To: <input type="text" name="mailTo" size="24"></p>
    <p>From: <input type="text" name="mailFrom" size="24"></p>
    <p>Subject: <input type="text" name="mailSubject" size="24"></p>
    <p><textarea rows="10" cols="40" name="Message"></textarea></p>
    <p><input type="submit" value="Send"> <input type="reset"></p>
  </form>
</body>

</html>
```

Figure 2.1 Quick Tour Example 1

BGSU Web Mail

To:

From:

Subject:

Example 1 consists of three text fields (**mailTo**, **mailFrom**, and **mailSubject**), one text area (**Message**), a submit button, and a reset button. All the user has to do is type the desired information into the form, press the **Send** button, and the e-mail is sent.

In this example, the names of the three text fields (**mailTo**, **mailFrom**, and **mailSubject**) are extremely important. The server recognizes these names as the To, From, and Subject lines of an e-mail message, respectively. In particular, **mailTo** and **mailFrom** are *required* fields for any given form that uses BGSU Forms.

2.2 Quick Tour Example 2

The second example is somewhat of an improvement over the previous example. The number and type of form elements remain the same, but the form has been formatted to look better on the screen (using an HTML table). Also, some JavaScript code has been added to check the data for validity before it is sent to the server.

Figure 2.2 Quick Tour Example 2

The image shows a web form titled "BGSU Web Mail". It contains three text input fields labeled "To:", "From:", and "Subject:". Below these is a large, empty text area with a vertical scrollbar on the right and a horizontal scrollbar at the bottom. At the bottom of the form are two buttons: "Send" and "Reset".

```
html>
```

```
<head>
  <title>Web Mail #2</title>
  <script type="text/javascript"
    src="http://www.bgsu.edu/scripts/checkForm.js">
  </script>
</head>
<body>
<!-- Replace "test3" with "submit3" to activate this form -
<form method="post" onsubmit="return checkForm( this )"
action="http://webapps.bgsu.edu/cgi-bin/bgsuform/test3">
<table>
  <caption><big><strong>BGSU Web Mail</strong></big></caption>
<tr>
  <td align="right">To:</td>
<td><input type="text" name="mailTo" size="24"></td>
</tr>
<tr>
  <td align="right">From:</td>
<td><input type="text" name="mailFrom" size="24"></td>
</tr>
<tr>
<td align="right">Subject:</td>
<td><input type="text" name="mailSubject" size="24"></td>
</tr>
```

```

<tr>
  <td align="center" colspan="2">
    <textarea rows="10" cols="40" name="message"></textarea>
  </td>
</tr>
<tr>
  <td align="center" colspan="2">
    <input type="submit" value="Send">&nbsp;&nbsp;&nbsp;<input type="reset">
  </td>
</tr>
</table>
</form>
</body>

</html>

```

2.3 Quick Tour Example 3

The third and final example in this Quick Tour is similar to the previous two examples, but more input fields have been added. Once the submit button is pressed, the user's input to each field in the form will be sent via e-mail. There is no practical limit to the number of input fields a form may have. (Note: Some of the JavaScript code may have been omitted from the listing below.)

```

<html>

<head>
  <title>Web Mail #3</title>
  <script type="text/javascript"
    src="http://www.bgsu.edu/scripts/checkForm.js">
  </script>
  <script type="text/javascript">
    // See BGSU Forms example online for the JavaScript code
  </script>
</head>
<body onload="document.webForm.mailTo.focus()">
  <!-- Replace "test3" with "submit3" to activate this form -->
  <form name="webForm" method="post" onsubmit="return checkForm( this )"
    action="http://webapps.bgsu.edu/cgi-bin/bgsuform/test3">
    <table>
      <caption><big><strong>BGSU Web Mail</strong></big></caption>
      <tr>
        <td align="right">To:</td>
        <td><input type="text" name="mailTo" size="24"></td>
      </tr>
      <tr>
        <td align="right">Cc:</td>
        <td><input type="text" name="mailCc" size="24"></td>
      <td>
        <small>
          (<input type="checkbox" name="_suppressEmptyFields"
            value="true">&nbsp;Suppress empty Cc)
        </small>
      </td>
      </tr>
    </table>
  </form>

```

```

<td align="right">From:</td>
<td><input type="text" name="mailFrom" size="24"></td>
<td>
  <select name="group" size="1">
    <option selected value="">Identify yourself</option>
    <option>Faculty</option>
    <option>Staff</option>
    <option>Student</option>
    <option>Alumni</option>
    <option>Guest</option>
  </select>
</td>
</tr>
<tr>
<td align="right">Subject:</td>
<td><input type="text" name="mailSubject" size="24"></td>
</tr>
<tr>
<td align="center" colspan="3">
  <textarea rows="10" cols="60" name="message"></textarea>
</td>
</tr>
<tr>
<td align="center" colspan="3">
  <input type="submit" value="Send">&nbsp;&nbsp;&nbsp;<input type="reset"
value="Cancel">
</td>
</tr>
</table>
</form>
</body>
</html>

```

Figure 2.3 Quick Tour Example 3

BGSU Web Mail

To:

Cc: (Suppress empty Cc)

From: Identify yourself

Subject:

3.0 Required Tags

As with any HTML document, forms incorporate standard HTML tags. However, certain tags and attributes are required for a BGSU form to work.

3.1 Form Tag

The most important tag used when creating or adding a form is the **<form>** tag. During the initial development of a form, add the following line to the web page where the form is to begin:

```
<form method="post" action="http://webapps.bgsu.edu/cgi-bin/bgsuform/test3">
```

An ending tag (**</form>**) is also required at the end of the form.

This version of the **<form>** tag allows for the use of the test version of the BGSU Forms script. No e-mail will be sent when this test script is used. After creating and testing a form (see Section 7.0, Testing a Form Locally), the **<form>** tag must be modified slightly to actually send e-mail. See Section 7.0 for details.

See BGSU Forms Examples online for complete working examples. If the form is to include JavaScript for data validation, see Section 6.2, Additions to Form Tag, for additional arguments required.

3.2 mailTo Address

Another necessary component of a BGSU form is the e-mail address of the individual receiving the form data. This address will appear on the To line of the e-mail message that is sent. To do this, use the following technique:

```
<input type="hidden" name="mailTo" value="nobody@bgsu.edu">
```

This line should be placed after the **<form>** tag mentioned in the Section 3.1. Setting the **type** attribute to **"hidden"** causes the input element to be invisible on the web page. The **name** attribute must be **mailTo** exactly as shown, otherwise it will not be possible to process the form. Lastly, replace **nobody@bgnet.bgsu.edu** with the full e-mail address of the person who is to receive the information from the form.

Note: A **mailTo** address must be a well-formed BGNET e-mail address. E-mail addresses outside the bgsu.edu domain will be rejected by BGSU Forms.

Note: Previous versions of BGSU Forms referred to the **mailTo** address as the **sendto** address. For backward compatibility with existing forms, the **sendto** address is still supported by Version 3 of BGSU Forms, but a warning is displayed. The **sendto** address will be removed in the next release of BGSU Forms, so users are advised not to use it.

3.3 mailFrom Address

The e-mail address of the user who fills out the form must also be given. This address will appear on the From line of the e-mail message that is sent. Any reply sent regarding the original message will also go to this e-mail address. To add this item to your form, include the following tag:

From: <input name="mailFrom" size="30">

The above line will create a text field (see Section 5.1, Text Field) for the client to enter an e-mail address. To ensure entry of the e-mail address by the user, JavaScript may be used (see Section 6.0, JavaScript and Catching Errors). Also, BGSU Forms requires e-mail addresses to be fully qualified as defined by published standards (e.g., ffalcon@bgnet.bgsu.edu as opposed to just `ffalcon`) so developers may want to add a cautionary note along with any request to enter an e-mail address in a form.

Note: By default, a **mailFrom** address may be an arbitrary (but well formed) e-mail address. If a developer wants to add a further requirement that a **mailFrom** address be within the BGSU domain, **mailFrom\$** can be used instead of **mailFrom**:

From: <input name="mailFrom\$" size="30">

Note: Previous versions of BGSU Forms referred to the **mailFrom** address as the **email** address. For backward compatibility with existing forms, the **email** address is still supported by Version 3 of BGSU Forms, but a warning is displayed. The **email** address will be removed in the next release of BGSU Forms, so users are advised not to use it.

Note: The **mailFrom** address was an optional field in previous versions of BGSU Forms. In Version 3, however, **mailFrom** is a *required* field.

3.4 Send Button

Once all of the information is entered into a form, the user must have a way to send the results to the server. This is accomplished through a submit button. To use such a button, enter the following line after the **<form>** tag:

<input type="submit" value="Send E-mail">

BGSU Forms

Replace the value "**Send E-mail**" with whatever words you want to display on the actual button. See Figure 3.4 for an illustration of a simple submit button and the HTML code used to create it.

Figure 3.4 Submit Button



```
<input type="submit" value="submit">
```

See Section 8.0, The Forms Submission Procedure, for more details about the Form Submission Process used for BGSU Forms.

3.5 Naming Items in a Form

Each form element must have a unique name associated with it. This name is used to distinguish the various input areas in the form and to make the e-mail output more readable. To name a form element, include an attribute/value pair in every tag:

```
name="name_of_item"
```

Replace **name_of_item** with the actual element name. Do not use spaces in the name field. Instead, use underscores to create multi-word names. The underscores will appear as spaces in the e-mail message generated when the form is submitted. For example, if **first_name** is a name, "first name: " would appear next to the entered value in the e-mail. Moreover, an underscore placed before the item name, such as **_name_of_item**, will make the item invisible in the web page returned to the user. For example, if the user is not supposed to see the e-mail address to which the form is sent, **_mailTo** could be used instead of **mailTo**.

4.0 Other Tags

The previous section discussed the required elements of a BGSU Form. In contrast, the elements discussed in this section are important and highly recommended, but not required.

4.1 mailCc Address

The e-mail address of a "carbon copy" recipient may be entered into the form. This will appear in the Cc line of the e-mail message that is sent. To add this item, include the following line in your form:

```
Cc: <input name="mailCc" size="30">
```

The above line will create a text field (see Section 5.1, Text Field) for the user to enter an e-mail address. BGSU Forms requires e-mail addresses to be fully qualified (e.g., `ffalcon@bgnet.bgsu.edu`) so developers may want to add a cautionary note along with any request to enter an e-mail address on a BGSU form.

Note: A **mailCc** address must be a well-formed BgNet e-mail address. E-mail addresses outside the `bgsu.edu` domain will be rejected by BGSU Forms.

Note: The **mailCc** address is a new feature in Version 3 of BGSU Forms.

4.2 mailSubject Field

The subject of the e-mail sent from the form should also be specified in the form. This value is determined by the maker of the form or entered by the user. To set a

predetermined value, add the following line to the form:

```
<input type="hidden" name="mailSubject" value="Subject of E-mail">
```

Be sure to replace "Subject of E-mail" with the desired subject. Alternatively, to have the user enter this information while completing the form, include this line in the form:

```
Subject: <input name="mailSubject" size="30">
```

Often the subject will be set by the maker of the form as in the first example. However, the second option is also possible. With no subject entered by the maker or the user, the subject of the e-mail sent will automatically be set to some default value.

Note: Previous versions of BGSU Forms referred to the **mailSubject** address as the **subject** address. For backward compatibility with existing forms, the **subject** address is still supported by Version 3 of BGSU Forms, but a warning is displayed. The **subject** address will be removed in the next release of BGSU Forms, so users are advised not to use it.

4.3 suppressEmptyFields Field

When the user presses the submit button, all the input data is sent to the server, even empty fields. The BGSU Forms script processes these empty fields by default. Thus empty fields appear in the output. If you would like to suppress empty fields from being displayed in the output, simply include the following line somewhere after the **<form>** tag:

```
<input type="hidden" name="_suppressEmptyFields" value="true">
```

By including this line, no empty fields will appear in the output.

Note: The **suppressEmptyFields** field is a new feature in Version 3 of BGSU Forms.

4.4 redirectURL Field

When the user presses the submit button and completes the submission process described in section 8.0, a standard HTML page is returned to the browser.

If you would like to redirect the output, simply include a line something like the following within the **<form>** tags:

```
<input type="hidden" name="redirectURL" value="http://www.bgsu.edu/">
```

As a result, the BGSU home page will be loaded into the user's browser after the submit button is pressed. Of course, any valid URL may be used. The only restriction is that the URL must be an *absolute* URL.

Note: The **redirectURL** field is a new feature in Version 3 of BGSU Forms.

4.5 Reset Button

A reset button will remove any responses the user previously entered into the form. Inclusion of a reset button is not necessary but is helpful for the user. To insert a reset button, type the following line where you want the button to appear:

```
<input type="reset" value="Clear">
```

BGSU Forms

The **type** attribute must be set to "reset", but the **value** can be anything. In general, "Clear" or "Reset" seem to be the most descriptive labels for a reset button. See Figure 4.4 for an illustration of a reset button and the HTML code used to generate it.

Figure 4.4 Reset Button



```
<input type="reset" value="reset">
```

5.0 HTML Form Elements

Many input types exist for the user to enter information into a form. These range from simple text fields to scrollable lists. Following are some of the possible input types with brief descriptions and illustrations of each. The examples are taken from a pizza order form. See <http://www.bgsu.edu/its/tsc/self-help/page13216.html> for the complete working example.

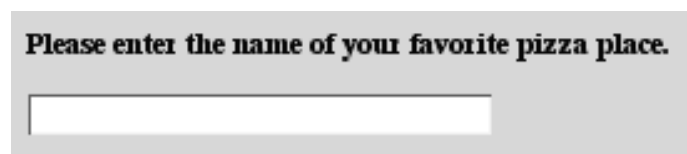
5.1 Text Field

The text field is probably the simplest form element. It allows the user to fill in a single line of text. This could be used for the client's name or e-mail address. To add a text box include the following line:

```
<input type="text" name="name_of_item" size="30">
```

replacing "name_of_item" appropriately (see Section 3.5, Naming Items in a Form). The size may vary depending on the width of the desired field. A default value can be added to the text field by inserting **value="default_value"** in the tag. This will become the default value and will print in the e-mail generated by the form even if the user enters nothing in the field. See Figure 5.1 for an illustration of a text field and the HTML code required to use it in a form.

Figure 5.1 Text Box



```
<b>Please enter the name of your favorite pizza place.</b>  
<input type="text" name="Name" size="30">
```

Note: Pixel widths are generally greater on PCs than Macintosh systems, so a text field will appear larger on a form viewed on a PC than on a Macintosh.

5.2 Text Area

A text area is very similar to a text field; however, a text area allows for the entry of multiple lines of text. To incorporate such a field into an input element, include the following tag:

```
<textarea name="name_of_item" rows="4" cols="30"></textarea>
```

The attributes of the tag should be self-explanatory. The size of the box can be adjusted by changing the number of rows or columns. Note, however, that pixel widths are wider on PCs than Macintosh systems. This means that the text area included in the form will look much larger on a PC than it does on a Macintosh system. When using this input type in a form, make sure the end tag (`</textarea>`) is included. See Figure 5.2 for an illustration of a text area and the HTML code used to generate it.

Figure 5.2 Text Area Box

```
<b>Describe any special requests.</b>
<textarea name="Request" rows="4" cols="25"></textarea>
```

5.3 Radio Buttons

To allow the user to select **only one** item from a pair or group, a radio button is used. When creating a button to be associated with other radio buttons, the **name** must be exactly the same for each, but the **value** must differ. Each radio button appears on the form as a small circle, which becomes filled when selected. To include a set radio buttons that allow the user to answer yes or no to a question, place the following lines in the form:

```
<input type="radio" name="Answer" value="Yes">Yes
<input type="radio" name="Answer" value="No">No
```

As described in Section 3.5, the value given to **name** attribute must be unique for each set or group of radio buttons used in the form. See Figure 5.3 for an example of radio buttons used in a form.

Figure 5.3 Radio Buttons

```
<b> What size of pizza would you like? </b>
```

```
<input type="radio" name="Size" value="Small">Small
<input type="radio" name="Size" value="Medium">Medium
<input type="radio" name="Size" value="Large">Large
<input type="radio" name="Size" value="Xtra Large">Xtra Large
```

BGSU Forms

Note that the item assigned to the **value** field only requires quotes when it includes a space as in "Xtra Large" in the example above. It is optional (but recommended) to use quotes when there are no spaces.

A feature of radio buttons is the capability of having a default button checked upon entry to the form. This is done by adding the word **checked** to the input tag of the corresponding button. For example, "Large" could be selected as the default size in the above example by editing the corresponding line as follows:

```
<input type="radio" name="Size" value="Large" checked>Large
```

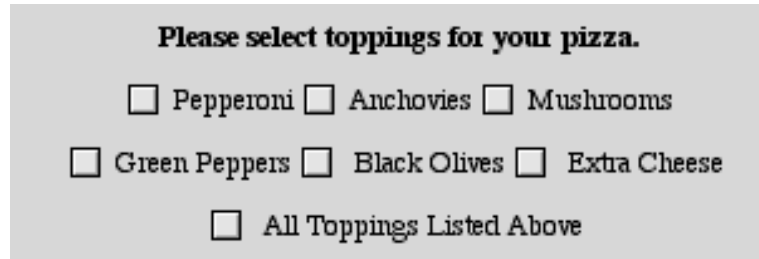
5.4 Checkboxes

To allow the user to be able to select or deselect any number of related items, checkboxes are required. With checkboxes, **one or more** items may be selected. Each checkbox appears on the form as a square box and includes a checkmark when it is selected. The default value on every checkbox is unselected. To set up a checkbox, include the following line:

```
<input type="checkbox" name="name_of_item" value="Item">Item
```

Edit "**name_of_item**" above and replace "**Item**" with the words to appear next to the checkbox and in the e-mail message generated by the form. When using a group of associated checkboxes, the **name** would be the same for each but the **value** would be unique. See Figure 5.4 for an example.

Figure 5.4 Checkboxes



Please select toppings for your pizza.

Pepperoni Anchovies Mushrooms

Green Peppers Black Olives Extra Cheese

All Toppings Listed Above

```
<b>Please select toppings for your pizza.</b>
```

```
<P>
```

```
<input type="checkbox" name="Toppings" value="Pepperoni">Pepperoni
```

```
<input type="checkbox" name="Toppings" value="Anchovies">Anchovies
```

```
<input type="checkbox" name="Toppings" value="Mushrooms">Mushrooms
```

```
<P>
```

```
<input type="checkbox" name="Toppings" value="Green Peppers">Green Peppers
```

```
<input type="checkbox" name="Toppings" value="Black Olives"> Black Olives
```

```
<input type="checkbox" name="Toppings" value="Extra Cheese"> Extra Cheese
```

```
<P>
```

```
<input type="checkbox" name="Toppings" value="All"> All Toppings Listed Above
```

As with radio buttons, one or more checkboxes may be selected by default by adding **checked** to the input tag.

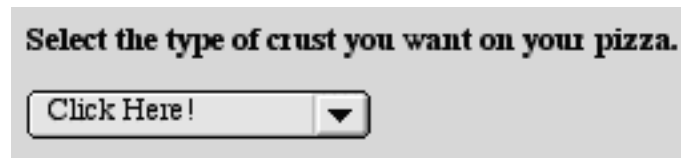
5.5 Pop-up Menu or Scrollable List

A pop-up menu or scrollable list allows the user to choose one or more options from a group of options. The HTML code used to generate these input types is very similar. To create a pop-up menu, for example, use a `<select>` tag with a sequence of `<option>` tags as illustrated below:

```
<select name= "Month">
  <option selected>January
  <option>February
  <option>March
  <option>April
  <option>May
  <option>June
</select>
```

These options let the user choose one of the first six months of the year. To have a particular option initially appear in the menu, use `<option selected>` instead of `<option>` beside that item. See Figure 5.6 for an example of a pop-up menu.

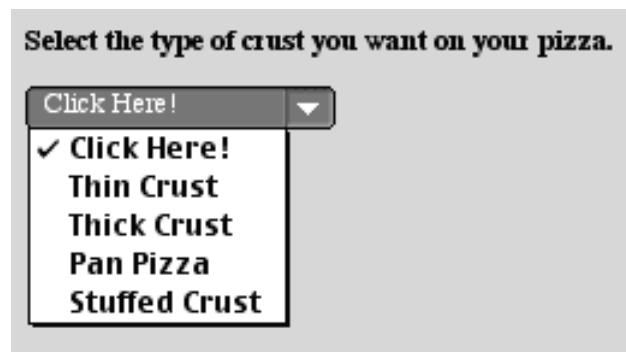
Figure 5.6 Pop-up Menu



```
<b>Select the type of crust you want on your pizza</b>
<select name="Crust_Type">
  <option>Click Here!
  <option value="Thin">Thin Crust
  <option value="Thick">Thick Crust
  <option value="Pan">Pan Pizza
  <option value="Stuffed">Stuffed Crust
</select>
```

A common interface design technique is to prompt the user for input in one of the option items. In the example above, "Click Here!" is listed as the first option. Figure 5.7 illustrates what the client sees after clicking on the pop-up menu.

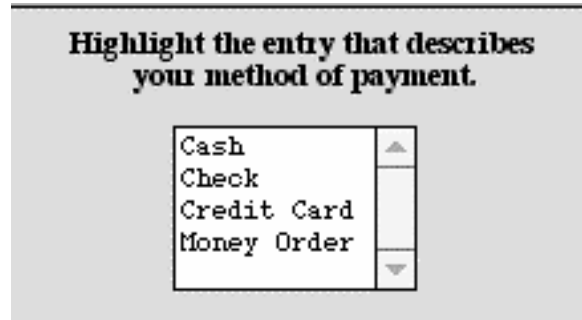
Figure 5.7 Activated Pop-up Menu



BGSU Forms

By making some minor alterations to the HTML code used to create a pop-up menu, users can select from a scrollable list of options. See Figure 5.8.

Figure 5.8 Scrollable List



```
<b>Highlight the entry that describes <BR>
your method of payment.<b>
```

```
<select name="Payment_Type" size="5" >
  <option value="Cash">Cash
  <option value="Check">Check
  <option value="Credit Card">Credit Card
  <option value="Money Order">Money Order
</select>
```

Note: Pop-up menus and scrollable lists are analogous to radio buttons and checkboxes, respectively.

6.0 JavaScript and Catching Errors

In order to ensure that the data entered by the user is correct, JavaScript may be used. Validation can occur as the user enters the data or just before the form is submitted. While such scripting is not necessary, it aids in the usability of a form. Some programming experience will make scripting easier, but it is not essential.

The following script fragments are relatively simple. In fact, JavaScript is a very powerful scripting language. While only a small portion of the capabilities of JavaScript will be shown here, the following explanations permit simple error checking in forms. To learn more about JavaScript, check out one of the numerous JavaScript tutorials on the web (see the cover page of this document for pointers to online resources).

6.1 Script Tag

To begin a JavaScript script, start with the following line:

```
<script type="text/javascript">
```

After this line, include whatever functions you've written to check the form and then finish the script with

```
</script>
```

The `<script>` tags usually appear in the `<head>` of the HTML document, but JavaScript code can be placed in the `<body>` as well.

6.2 Additions to Form Tag

In order for the various functions defined in the JavaScript script to work properly, the `<form>` tag must include additional attributes and values:

```
<form method="post" name="myForm"
      action="http://webapps.bgsu.edu/cgi-bin/bgsuform/test3"
      onsubmit="return checkForm()" onreset="return clearForm()">
```

The values associated with the **name**, **onsubmit** and **onreset** attributes are chosen by the developer. The **name** attribute ("myForm") gives the form a name to use when writing the JavaScript code. The **onsubmit** and **onreset** attributes control the execution of certain JavaScript functions, which are invoked when the submit or reset buttons are pressed, respectively.

6.3 checkForm Function

By creating a **checkForm** function, the information submitted through a form may be checked for accuracy and thoroughness on the client side before actually being processed by the server. If a problem is found, the user will be notified immediately of the error, then allowed to correct the entry and re-submit the form. With the `<form>` tag above (see Section 6.2, Additions to Form Tag), the **checkForm** function will be executed when the submit button is pressed.

Adding a **checkForm** function, or any function, in JavaScript is relatively simple. Like most HTML tags, each function needs a beginning and an end. Also, each function must be placed between the `<script>` and `</script>` tags. A **checkForm** starts like this:

```
function checkForm() {
  // insert code here
  return true;
}
```

Unfortunately, adding the above code does not accomplish any error checking. The following sections show code that can be included in the **checkForm** function to check for a variety of errors.

6.3.1 Blank Text Field

If the user fails to fill in part of a form, the results may not be complete. JavaScript allows the web developer to ensure entry of text. The following section is an example of how to check for a blank field in the submitted form. It should be placed somewhere in the **checkForm** function (see Section 6.3, checkForm Function).

```
var entry = window.document.myForm.name_of_item.value;
if ( entry.length == 0 ) {
  window.alert ( "Error: Text field left blank!" );
  window.document.myForm.name_of_item.focus();
  return false;
}
```

In order to use this section of code, replace **name_of_item** with the name of the text field or text area to check. Note that **myForm** corresponds to the **name** given in the `<form>` tag (see Section 6.2 Additions to Form Tag). The code above will check for a text length of zero for the

BGSU Forms

specified field (**name_of_item** in this case). If this field is empty, an alert window will appear, the empty box will be highlighted, and the cursor will be placed in the box for subsequent input.

6.3.2 Entry Too Long

For some items in a form, entry should be limited to a certain number of characters. For example, a box for entry of a middle initial should allow either zero or one character. The following shows how to check for an errant length greater than one.

```
var initial = window.document.myForm.middleInitial.value;
if ( initial.length > 1 ) {
    window.alert ( "Error: Middle initial too long: " + initial );
    window.document.myForm.middleInitial.focus();
    window.document.myForm.middleInitial.select();
    return false;
}
```

The above assumes that the name of the text field is **middleInitial** and the form name is **myForm**. The expression "+ initial" in the **window.alert** line will output the value of the **initial** variable in the warning window.

6.3.3 Invalid E-mail Address

When an e-mail address is entered, it should generally contain both the 'at' sign (@) and a period (.) after the sign. The following code uses these criteria to perform a primitive check for a well-formed e-mail address.

```
var mailFrom = window.document.myForm.mailFrom.value;
var atsignPos = mailFrom.indexOf( "@", 0 );
if ( atsignPos == -1 ) {
    window.alert ( "Error: No @ in E-mail Address!" );
    window.document.myForm.mailFrom.focus();
    window.document.myForm.mailFrom.select();
    return false;
}
if ( mailFrom.indexOf( ".", atsignPos ) == -1 ) {
    window.alert ( "Error: No . after @ in E-mail Address!" );
    window.document.myForm.mailFrom.focus();
    window.document.myForm.mailFrom.select();
    return false;
}
```

The first if statement checks for an 'at' sign (@) in the e-mail address. After an 'at' sign (@) is found, the second if statement checks for a period after this sign. This code is more advanced than the previous examples.

7.0 Finishing the Form

After developing a form written in HTML, it must be tested locally. This means that the page should be opened in a web browser while the file is still on the hard drive. In the Rhythmyx Content Management System (CMS), this testing can be done while doing a Preview of the page. Fill in the form like a user would and click the submit button. The page that appears will show if the results are correct and what the client will see when the form is actually submitted. If the desired response is not produced edit the form's HTML code so that the displayed output is what's desired.

Once the form is debugged and tested, change the `<form>` tag from:

```
<form method="post" action="http://webapps.bgsu.edu/cgi-bin/bgsuform/test3">
```

to

```
<form method="post" action="http://webapps.bgsu.edu/cgi-bin/bgsuform/submit3">
```

If the form includes JavaScript, the additional attributes described in Section 6.2, Addition to the Form Tag, remain the same. After changing the value of the **action** attribute from **"test3"** to **"submit3"**, fill out the form and press the submit button. This should generate an e-mail message. Check this e-mail message to see that the form worked. If an e-mail message is not received, check that the form, especially the **mailTo** field, is correct.

8.0 The Form Submission Procedure

To insure that the BGSU Forms software is being used for the intended purpose (to collect forms data), an enhancement has been made to the forms submission process. In addition to pressing the submit button when the form is complete, the client will also have to type the code they see in a security image and click another button to confirm and send their data to the intended recipient. If the client just presses the submit button and does nothing else, their form data will not be processed. So, in instructions you provide to your constituents, you need to describe what they need to do so you will receive their forms data.

The forms submission process will work like this:

1. The client will enter data into the form and then press the submit button.
2. The client will see an image containing a security code, a text box to enter the code, a confirmation button followed by a list of the data items that were entered in this form.

Figure 8.1 Forms Submission with Security Code

BGSU Forms

e x n 5

For security reasons, please enter the code in the graphic above:

(If you have difficulty reading the above graphic, please contact Technology Support Center (TSC) at 419-372-0999)

Confirm and Send

Thank you for using BGSU Forms!

The following information will be submitted:

- Comments: xx
- Full Name: Freddy Falcon
- Other Contact: ball games
- Phone: 2-0000
- Topic: I need to contact you
- Webpage: http://www.bgsu.edu/
- Who: BGSU Student
- mailFrom: webmaster@bgnet.bgsu.edu
- mailSubject: Web Development Feedback Form
- mailTo: webdev@bgnet.bgsu.edu
- submit: Submit

3. The client must then enter the code and press the confirm and send button before the data is sent to the recipient.

BGSU Forms

8.1 Why do we need security code on BGSU Forms?

Some sophisticated users have been able to exploit the BGSU forms script to automatically send out spam. To prevent this, we decided to add a second level validation using a security code. This second level validation will reject any automated form submission, therefore reducing spam. The form script will continue to function as before, except it requires users to read the security code, type the code in the box provided, confirm and send. This security code is based on CAPTCHA. See Wikipedia at <http://www.wikipedia.org/> for a definition.

9.0 Additional Help

For questions associated with BGSU hardware, software, network connections, BGNet accounts, class accounts or other computer accounts, clients should consult the TSC self-help pages at URL:

<http://www.bgsu.edu/its/tsc/self-help/>

Clients can also contact the Technology Support Center (TSC) in Hayes Hall, room 110, or by phone at 419-372-0999. Problems can be also reported to the TSC by using the TSC Online Submission form at URL:

<http://www.bgsu.edu/its/tsc/page9500.html>

Comments, corrections, or suggestions concerning this document can also be reported to the TSC. Photocopying of this document is encouraged. Reprints of the document's content are permitted if credit is given and a copy is sent to ITS Documentation, 265 Hayes Hall, Bowling Green State University, Bowling Green, OH 43403.