BGSU MAIL 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 Mail Forms, the web developer simply develops an HTML form according to the specification given in this document.

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.

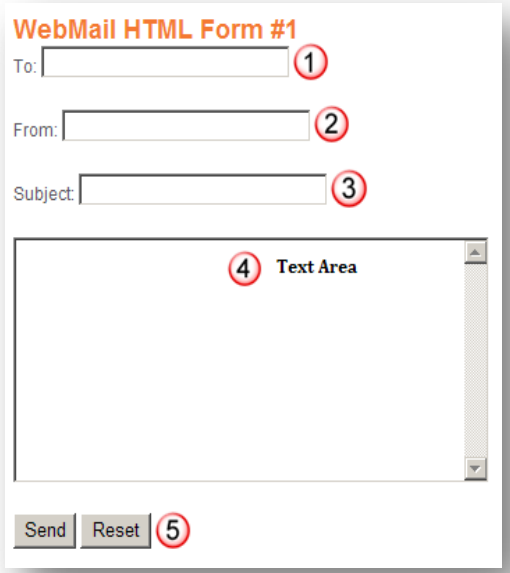
This document describes the process of creating a form along with specific requirements for using BGSU Mail Forms. Therefore, the information contained in this document, in part, pertain to forms in general.

If you are not comfortable modifying and creating basic HTML, this document may contain concepts you are unfamiliar with. Visit http://w3schools.com/html/default.asp for a basic introduction before continuing.

Contents

QUICK TOURS

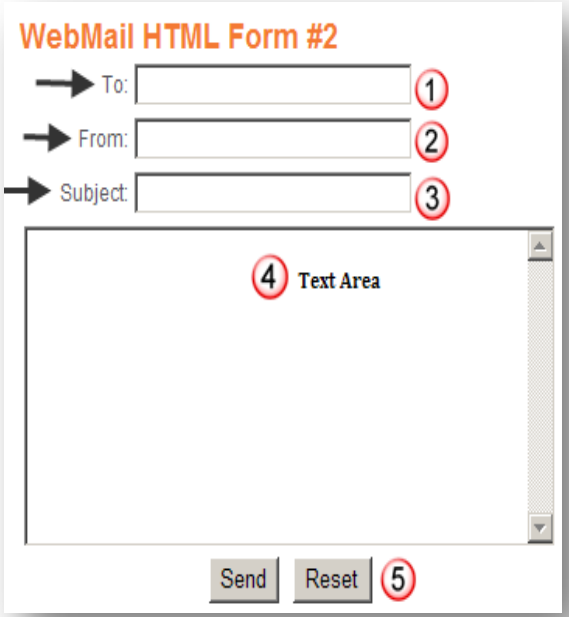
Example 1: consists of three text fields: mailTo, mailFrom, and mailSubject, on text area (Message), a submit form, and a reset button. All the user has to do is type the desired information into the form, press the Send button, and the e-mail is sent.




```
<html>
<head>
<title>Web Mail #1</title>
</head>
<body>
<!-- Replace "test.htm" with "submit.htm" to activate this form -->
<form method="post" action="https://webapp.bgsu.edu/mailform/test.htm">
<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>
```

QUICK TOURS

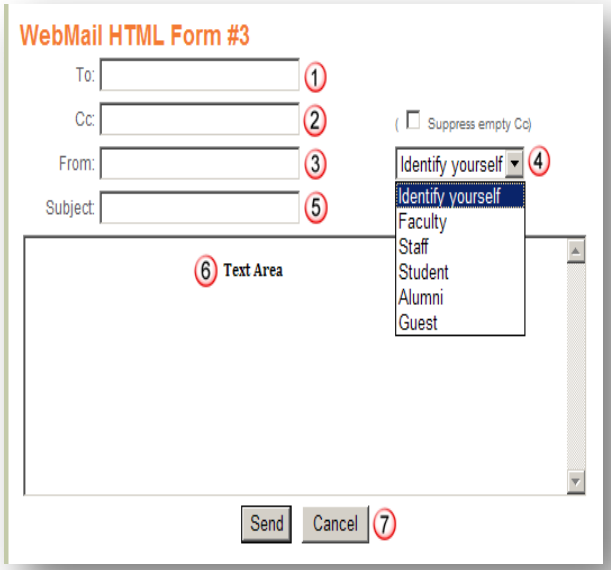
Example 2: is somewhat an improvement over example 1; the number and type of form elements remain the same, but the form has been formatted to look better on the screen (using an HTML table). Some JavaScript code has been added to check the data for validity before it is sent to the server.




```
<html>
<head> <title>Web Mail #2</title> <script type="text/javascript"
src="http://www.bgsu.edu/scripts/checkForm.js"> </script>
</head> <body>
<!-- Replace "test.htm" with "submit.htm" to activate this form -->
<form method="post" onsubmit="return checkForm(this )"
 action="https://webapp.bgsu.edu/mailform/test.htm"> <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">  <input type="reset">
</td> </tr>
</table> </form>
</body> </html>
```

QUICK TOURS

Example 3: in the 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 man have. (NOTE: some of the JavaScript code may have been omitted from the listing below.)




```
<html>
<head> <title>Web Mail #2</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>
<!-- Replace "test.htm" with "submit.htm" to activate this form -->
<form method="post" onsubmit="return checkForm(this )"
 action="https://webapp.bgsu.edu/mailform/test.htm"> <table>
<caption><big><strong>BGSU Web Mail</strong></big></caption>
<tr>
<td>To:</td>
<td><input type= "text" name= "mailTo" size= "24"></td>
</tr>

<tr>
<td>Cc:</td> <td><input type= "text" name="mailFrom"size= "24"></td>
<td><input type="checkbox" name="_supressEmptyFields"><small>(Suppress empty
CC)</small></td>
</tr>

<tr>
<td>From:</td> <td><input type= "text" name= "mailFrom" size= "24"> </td>
<td> <select name= "group" size= "1"> <option>
Identify yourself</option> <option>Staff</option>
<option>Student</option> <option>Alumni</option> <option>Guest</option>
</select> </td>
</tr>

<tr>
<td>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">   <input
type="reset" value="Cancel"> </td>
</tr> </table>
</form> </body>
</html>
```

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.

Form Tags

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 pages where the **<form>** is to begin:

<form method="post" action="http://webapp.bgsu.edu/mailform/submit.htm">

mailTo

Another necessary component of a BGSU form is the e-mail address of the individual receiving the form data. Must be inside the BGSU domain, well formed e-mail address. If using more than one address use a comma as a separator. 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. 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.

mailFrom

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 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="24">

A mailFrom address must be a well formed e-mail address. To add a requirement that a mailFrom address be within the BGSU domain, **mailFrom$** can be used:

From:<input name="mailFrom$" size="24">

Send

Once all 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 tag:

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

To place an actual word on the button replace the value "Send E-mail" with something else

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

OTHER TAGS

The elements in this section are important and highly recommended, but not required.

mailCC

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. If more than one address use a comma as a separator. To add this item, include the following line in your form:

Cc:<input name="mailCc" value= "someoneelse@bgsu.edu" size="30">

A mailCc address must be a well formed BGSU e-mail address. E-mail addresses outside the bgsu.edu domain will be rejected.

mailSubject

The subject of the e-mail sent from the form should also be specified in the form. This value is determined by the creator of the form or entered by the user. To set a predetermined value, include this line in the form:

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

suppressEmptyFields

When a user presses the submit button, all the input data is sent to the server, even empty fields. The script processes these empty fields by default. Thus empty fields appear in the output and in the e-mail. To suppress empty fields from being displayed in the output and e-mail, include the following line:

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

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

redirect URL

When the user presses the submit button and completes the submission process described on the Form Submission page, a standard HTML page is returned to the browser. To redirect the output include a line within the form:

<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, any valid URL may be used. The URL must be an absolute URL..

Reset

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 include the following line where you want the button to appear:

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

<input type=”reset” value=”Reset”>

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 rest button.

HTML FORM ELEMENT TEXT

Many input types exist for the user to enter information into a form. These range from simple text fields to scrollable lists. Below are some of the possible input types with brief descriptions and illustrations of each. *The examples are taken from a pizza order.*

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">

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.

Example Text Box

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

Text Area

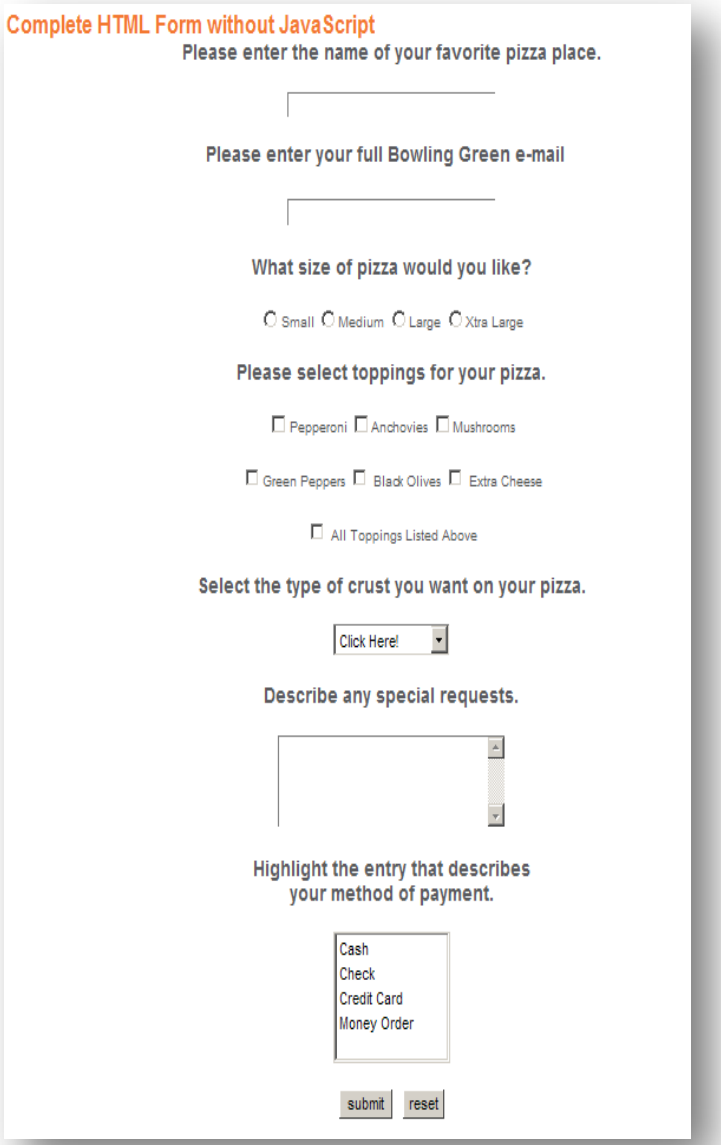
A text area is very similar to a text field, however, a text area allows for the entry of multiple lines of text. To incorporate such a fields into an input element, include the following tag:

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

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 that it does on a Macintosh system. When using this input type in a form, make sure the end tag **(</textarea>)** is included.

Example Text Area Box

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



HTML FORM ELEMENT RADIO BUTTONS

Below is a continuation of some of the possible input types with brief descriptions and illustrations of each. *The examples are taken from a pizza order.*

Radio Buttons

To allow a 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 same form as a small circle, which becomes filled when selected. To include a set of radio buttons that allow the user to answer yes or no to a question, place the following lines in the form:

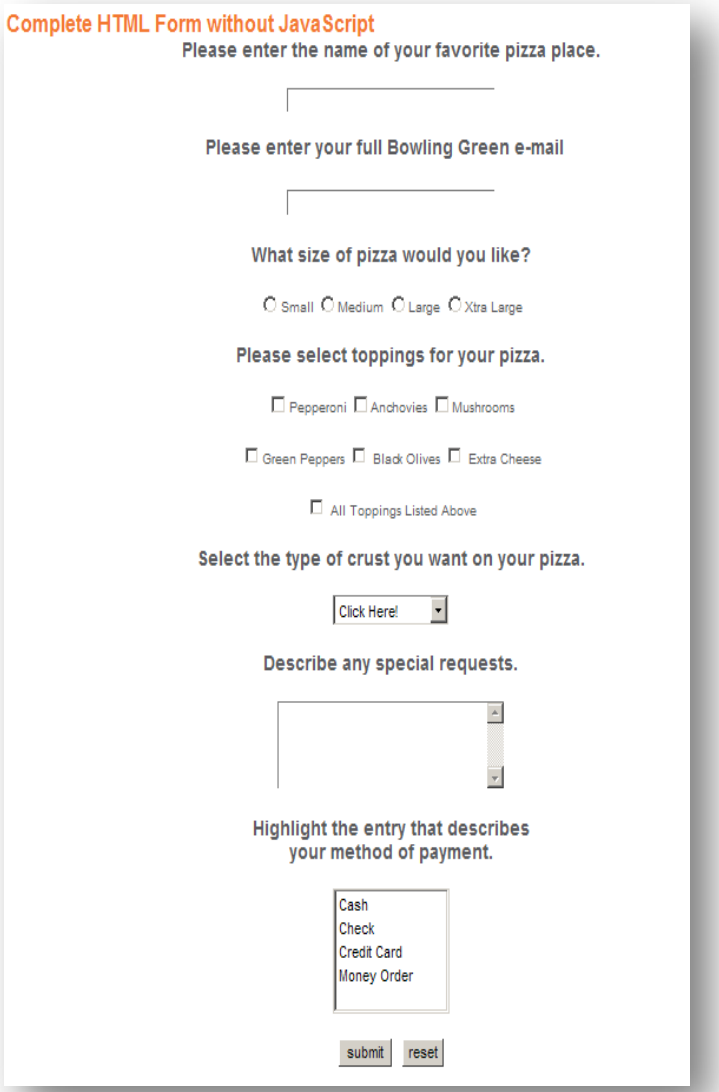
<input type="radio" name="Answer" value="Yes">Yes
<input type="radio" name="Answer" value="No">No

Example Radio Buttons

What size pizza would you like?
<input type="radio" name="Size" value="Small">Small
<input type="radio" name="Size" value="Medium"<Medium
<input type="radio" name="Size" value="Large">Large">Large
<input type="radio" name="Size" value="Xtra Large">Xtra Large

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



HTML FORMS ELEMENTS CHECKBOXES

Below is a continuation of some of the possible input types with brief descriptions and illustrations of each. *The examples are taken from a pizza order.*

Checkboxes

To allow a 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:

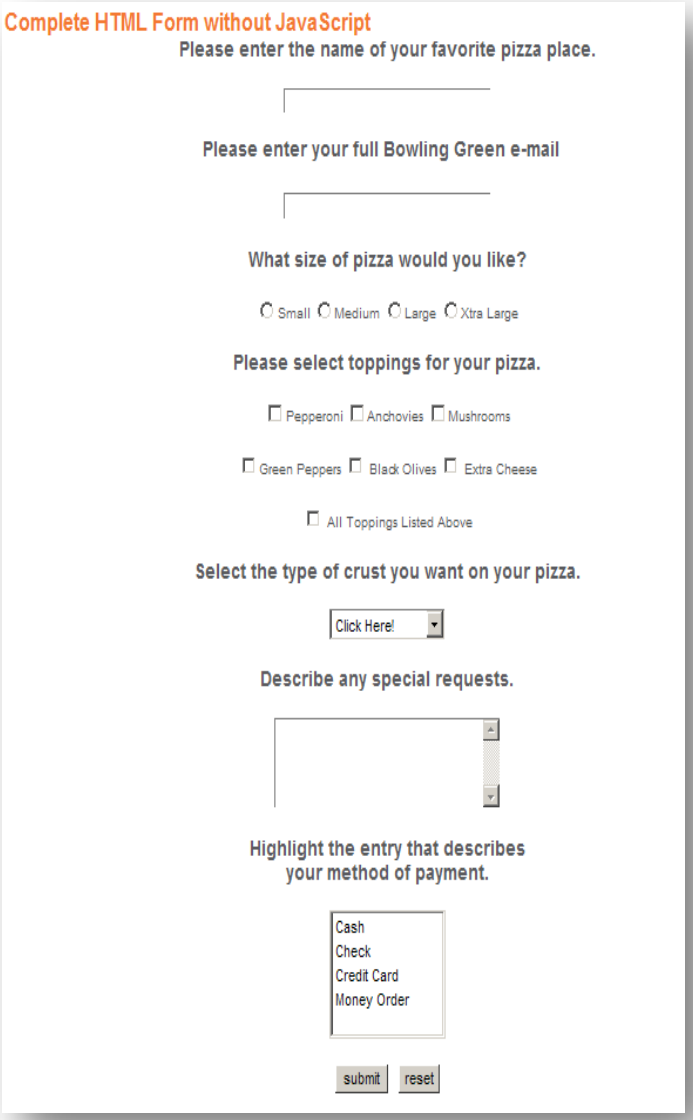
<input type="checkbox" name=" name_of_item" value="Item">Item

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.

Examples Checkboxes

Please select toppings for your pizza.
<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
<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
<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.



HTML FORM ELEMENT POP UP MENU

Below is a continuation of some of the possible input types with brief descriptions and illustrations of each. *The examples are taken from a pizza order.*

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 **<options>** 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.

Example Pop-Up

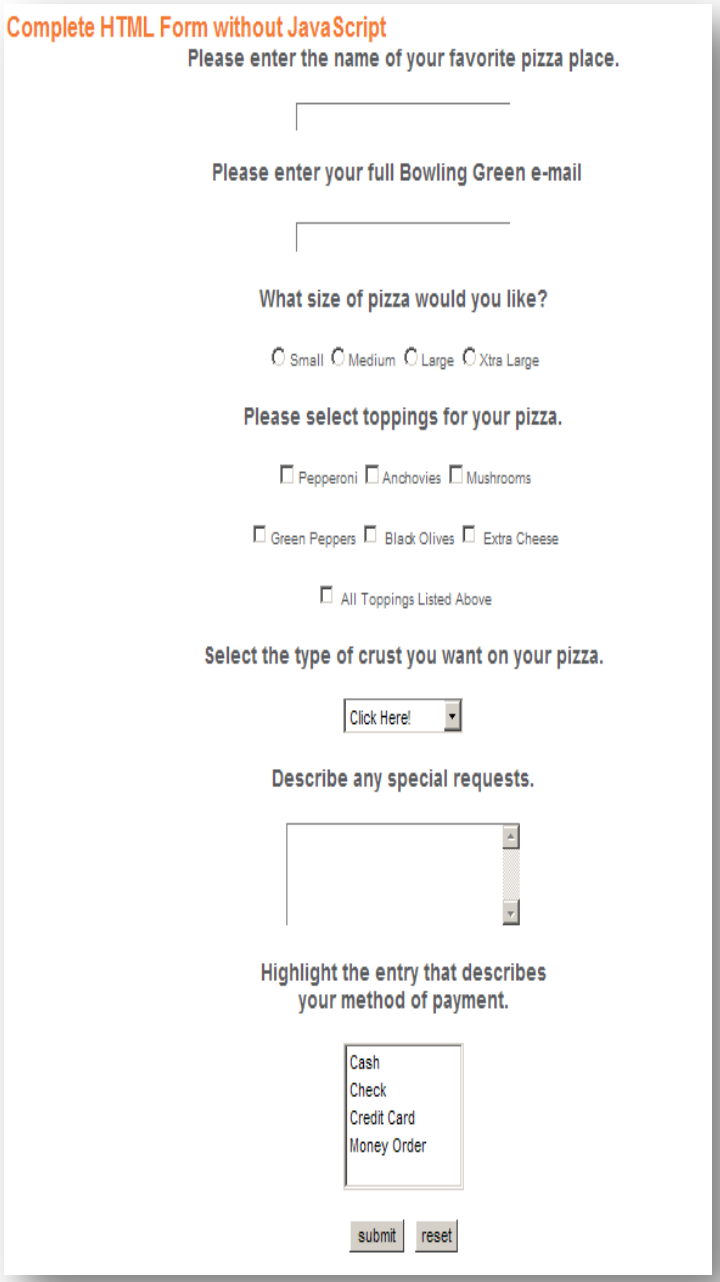
```
<b>Select the type of crust you want on your pizza</b>
<select name="Crust_Type">
<option>Click Here!
<option value="Thin">Thin Crust
<option value="Thick">Thick Crust
<option value="Pan">Pan Pizza
<option value="Stuffed">Stuffed Crust
</select>
```

Activate Pop-Up

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.

Example Scrollabe List

```
<b>Highlights the entry that describes<BR>your method of payment.<b>
<selectname="Payment_Type"size="5">
<optionvalue="Cash">Cash
<optionvalue="Check">Check
<optionvalue="Credit Card">Credit Card
<optionvalue="Money Order">Money Order
</select>
```



JAVASCRIPT & 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.

Script Tag

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 permits simple error checking in forms. To learn more about JavaScript, check out one of the numerous JavaScript tutorials on the web. (see Additional Information page for pointers to online resources).

<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.

Additions to Form Tag

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

<form method="post"name="myForm"
action="https://webapp.bgsu.edu/mailform/test.htm"
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.

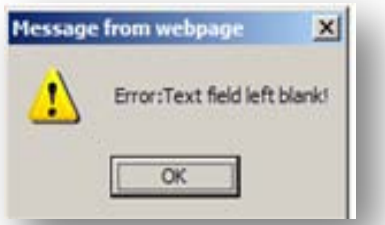
checkForm Function

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 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: (JavaScript statements to actually validate the input fields are not shown.)

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

Blank Text Field

If a 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. Below is an example of how to check for a blank field in the submitted form. It should be placed somewhere in the 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 test field or text area to check. Not that myForm corresponds to the name given in the <form> tag. The code will check for a text length of zero for the 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.

JAVASCRIPT & CATCHING ERRORS II

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.

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.

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-mailAddress!");
window.document.myForm.mailFrom.focus();
window.document.myForm.mailFrom.select();
return false;
}
if (mailFrom.indesOf(".", 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 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.

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=https://webapp.bgsu.edu/mailform/test.htm
TO
<form method="post" action=https://webapp.bgsu.edu/mailform/submit.htm

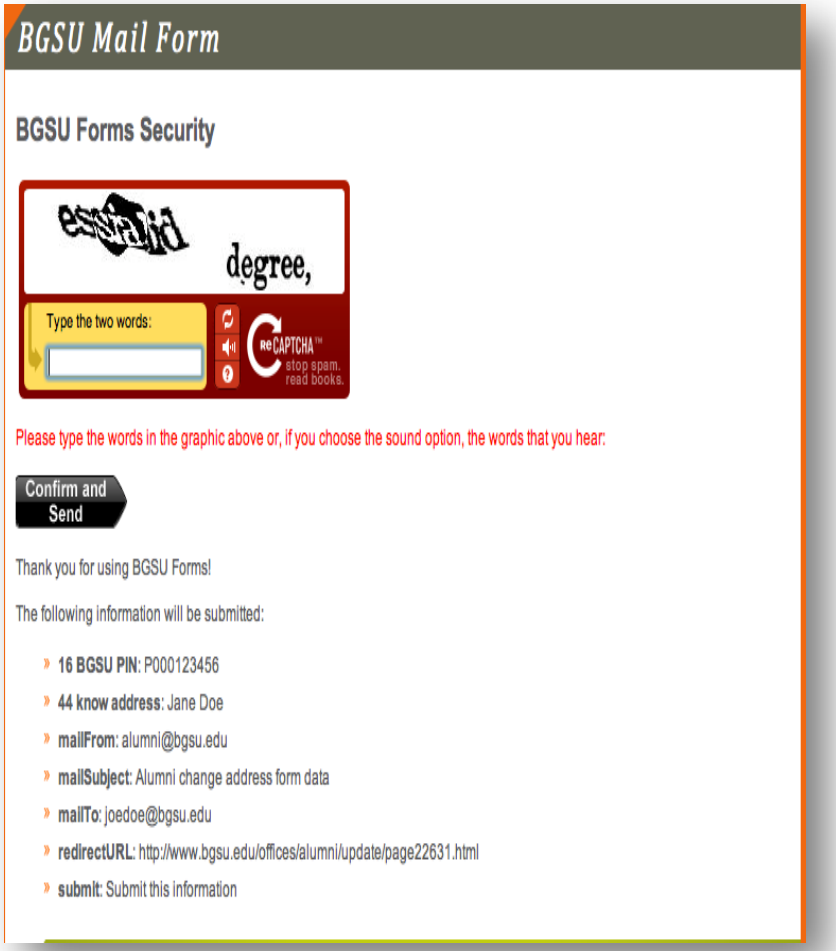
If the form includes JavaScript, the additional attributes described in "*Entry Too Long",* addition to the Form tag, remain the same. After changing the value of the **action** attribute form **"test.htm"** to **"submit.htm"**, fill out the form and press the submit button. This should generate an e-mail 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.

FORM SUBMISSION

Before users can submit a form, the data must be entered and the submit button pressed. A confirmation page with a security feature is then displayed. Users will have to type a security code they see in a distorted image or several words they hear from an ambient audio recording.
A list of the data items they entered in the form is redisplayed in the confirmation page. (Form fields with names beginning with, "_", i.e. "_first_name" do not display on the confirmation page.)
Pressing the "Confirm and Send" button will test the security code entered and, if valid, the form data will be sent to the recipient. See image below.

Why do we need security codes or BGSUMAILForm

Some sophisticated external users, in the past, have been able to exploit the BGSUMailForms script to automatically send out spam. To prevent this, BGSU decided to add a second level validation using a security code. This second level validation will reject an automated form submissions, therefore reducing spam. The form script will continue to function as before, except it requires users to read and/or listen, then type in the security code in the text box provided. Lastly, the user will confirm and send the data.
The security code is based on reCAPTCHA. http://www.wikipedia.org/



ADDITIONAL INFORMATION

Special Form Fields

Action Attribute:
action="https://webapp.bgsu.edu/mailform/submit.htm"

Mail To: Recipient of email:
mailTo - Required. Must be inside of bgsu domain, well-formed e-mail address. If more than one address exists use a comma as a separator (i.e. tjacobi@bgsu.edu, ffalcon@bgsu.edu)

Mail From: sender of e-mail (as perceived by the recipient)
mailFrom - Required, well-formed e-mail address
mailFrom$ - used to require a BGSU e-mail address

Mail Subject: subject message of e-mail
mailSubject

Mail Carbon Copy: secondary recipient of e-mail
mailCc: must be inside of bgsu domain, well-formed e-mail address. If more than one address exists, use a comma as a separator

Suppress empty fields: used to suppress display and emailing of input fields with no data entered.
Suppress EmptyFields - Values "true" or "yes" to activate it

Redirect URL: web page to redirect user after successful submission of form.
redirectURL - any valid URL

Self-Help

http://www.bgsu.edu/its/tsc/self-help/page10730.html
http://www.bgsu.edu/its/tsc/self-help/page13216.html
http://www.bgsu.edu/scripts
http://www.htmlgoodies.com/tutorials/forms/article/php/3479121
http://www.w3schools.com/html/default.asp.

For questions associated with BGSU hardware, software, network connections, BGSU accounts, class accounts, or computer accounts consult the Technical Support Center at 419-372-0999 (tsc@bgsu.edu) Inquiries may also be reported to TSC using the TSC Online Submission from at:

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