Student_____

The Online Grade Book Project

Date: _____

JavaScript Lab 02

Overview In this exercise you will add some client-side field validation to your html forms. Not all fields will need to be validated on the client side, but some will. In this exercise you will learn how to:

1) call a JavaScript function to validate a form, prior to the form being submitted to the server.
2) access a form, and specific form fields from within a JavaScript function.
3) learn more about the parts of an html document and begin to see the document as an object with properties and events.
4) validate one or more form fields.

Part 1 – Setting up a testing environment with a real html document

Step 1: Open an existing html form that you have been working on for the project.

Step 2: Save your html form with a new name, with the suffix _js.html.

Step 2: Make sure that your form name and all of the fields in the form conform to the naming standards that have been passed out.

Step 3: Test your form by changing the action to call php/showparms.php and verify that ALL of your form fields are properly being passed to the php program. If they are not, then fix your html form until all form fields are being property received by the program,

Part 2 – Defining a JavaScript function to validate a form

Step 1: Write the name of your form here: _____

Step 2: Write the name of one of the fields of your form here:_____
(last names are good for testing)

Step 3: Copy the code from the class web site from js_validate_1.txt into the <head> </head> section of your html document.

Step 4: Replace "your-form" with what you wrote from Step 1 above, and replace "your-field" with what you write from Step 2 above – in both places in.

Step 5: Change your <form> tag to include the following
<form …. **onsubmit="return validateForm1()"** method="post">
Keep the action to php/showparns.php
Example: <form name="tf" id="tf"
action = php/showparms.php
onsubmit="return validateForm1()" method="post">

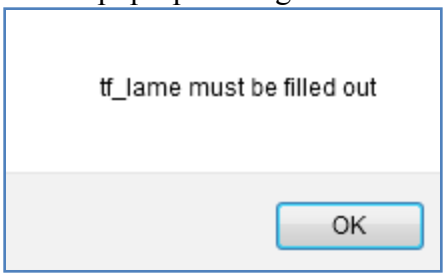
Student_____

The Online Grade Book Project

Date: _____

Part 2 Continued

Step 6: Save your file. Start the Cloud 9 application and preview your html form in a browser. Leave the form blank and press submit – by the way, if you don't have a submit button on your form, then add a submit button to your form.
You should get an alert pop-up message that looks similar to:



Step 7: Close the alert button. Notice that the form was not submitted. Add some data to the form field you wrote down in step 2, and re-submit the form. Because the form

field that you validated is no longer blank, the form will now be submitted to the server.

Part 3: Understand what is going on

Start with the action that triggers the validation:

```
<form name="tf" id="tf"    action = php/showparms.php
                           onsubmit="return validateForm1()" method="post">
```

The process of hitting the submit button triggers an event – the onsubmit event. By adding the "onsubmit" clause, the web browser creates something that will "listen" for an event, which will get triggered when the user hits the submit button. The onsumit will cause the validateForm1() function to be called. The function will return either "true" or "false." If "true" is returned, then the browser will send the form data to the server and the server will execute showparms.php.

If the validateForm1() function returns "false" the browser will not send the form data to the server. The function itself caused the alert message box to appear, so now take a look at the JavaScript.

Looking at the JavaScript:

```
<head>
  <script>
     function validateForm1() {
       var x = document.forms["tf"]["tf_lname"].value;
       if (x == null || x == "") {
          alert("tf_lame must be filled out");
          return false;
        }
       }

  </script>
</head>
```

- The function is within the <script> </script> tags, this will usually work, though some browser may want the exact type of script that is being written, JavaScript is just one of several scripting languages that a browser could support.
- The <script> is in the html <head> tag, this is fairly common. JavaScript could appear almost anywhere in the document, the <head> is common. You may also find JavaScript at the end of an html file because there can be reasons why you would want the entire document to load before doing some actions. For the grade book application having it in the <head> tag is recommended, it will keep it away from the html. It is also possible to store the JavaScript in a separate file, and we may see that in a future lab.

- Reviewing the validateForm1 one line at a time:

```
1) function validateForm1() {
2)   var x = document.forms["tf"]["tf_lname"].value;
3)   if (x == null || x == "") {
4)      alert("tf_lame must be filled out");
5)      return false;
6)     }
7)   }
```

- Line one declares the function name, this function does not accept parameters, but JavaScript functions can take parameters.
- Line 2 declares a variable named x, it primary purpose is to pull out the value that the user typed into the form – it makes the rest of the function easier to use. The html file is viewed as an "object" with a global name of document. This is not anything that you needed to define. As the browser reads your html document, and it finds your form, it adds your form name to a list of all of the other forms in the html file, if there are any. It also keeps a list of all of the fields in the form.

  `document.forms["tf"]["tf_lname"].value`

  will tell the browser to find the tf form, and the field name tf_lname, and to get its value, which will then be stored in variable x.

- Reviewing the validateForm1 one line at a time:

```
1) function validateForm1() {
2)   var x = document.forms["tf"]["tf_lname"].value;
3)   if (x == null || x == "") {
4)      alert("tf_lame must be filled out");
5)      return false;
6)     }
7)   }
```

- Line 4 as you may guess because you have worked with C, tests to see if x (the from field data) is either null – which means the user didn't enter anything, or it is blank. Here the only validation that is really done is checking to see if *anything* had been entered. Some more rigorous checking could be done (did the user enter a ? or number or something that is clearly not a last name?) If either the field is null or blank then statements 4 and 5 (the ones inside the { braces} will be executed.

- Line 4 opens a message box with an alert to the user that there must be data in the form field (in this example, the last name field).

- Line 5 exits the function and returns the value 5, which will get executed after the user closes the alert message box.

- Lines 6 and 7 close the if statement and terminate the function.

- Notice that "true" is not explicitly returned if there is something in the field – it will return true by default.

Additional Notes:

- JavaScript can be unforgiving in that if it does not find a form with the name you specified, or a field with the name you specified, the browser may stop executing any JavaScript.

- Testing each aspect of your JavaScript, isolated from the actions of a php program, will help you to find errors and will save yourself debugging time. We will learn debugging techniques.