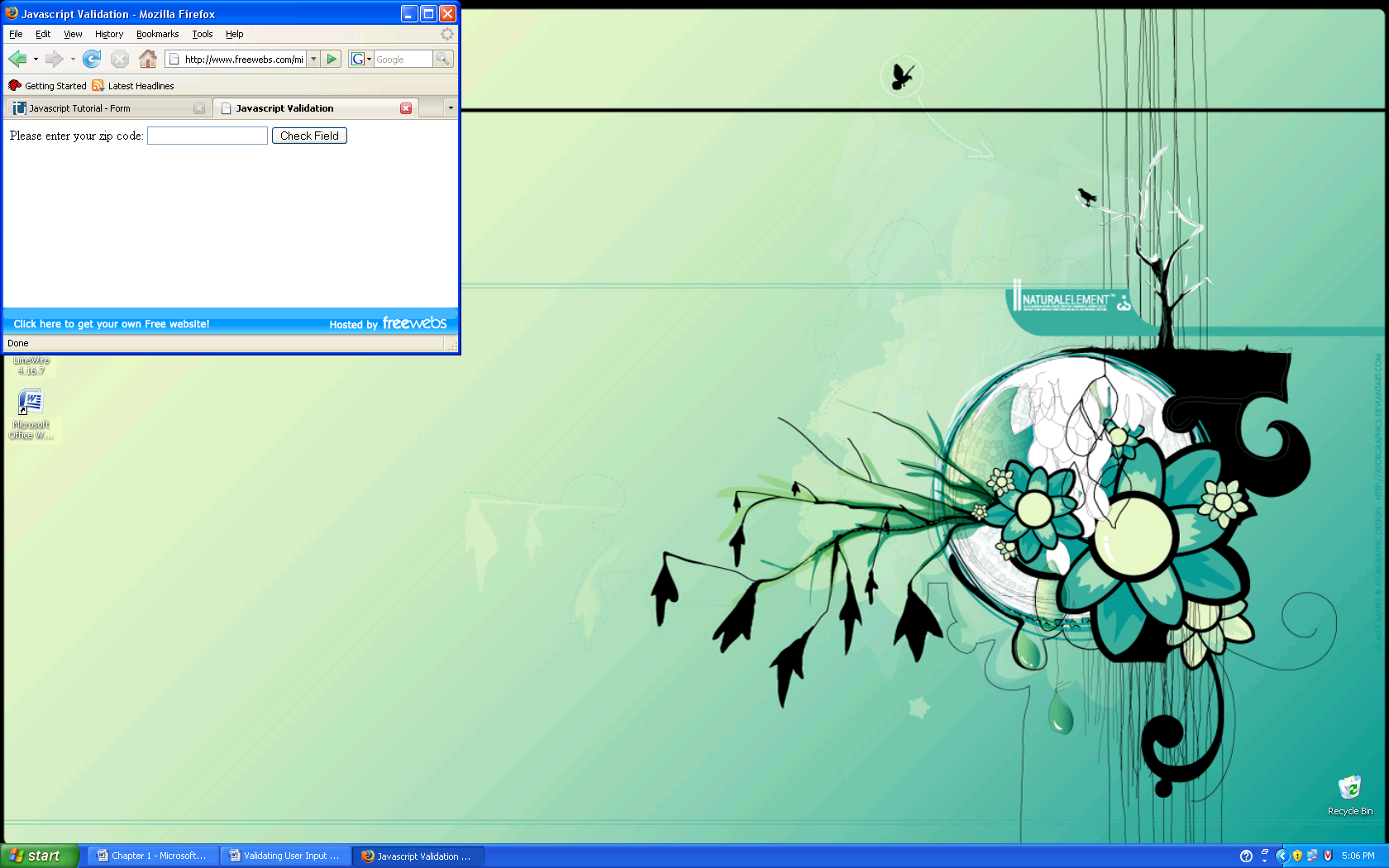
Validating User Input

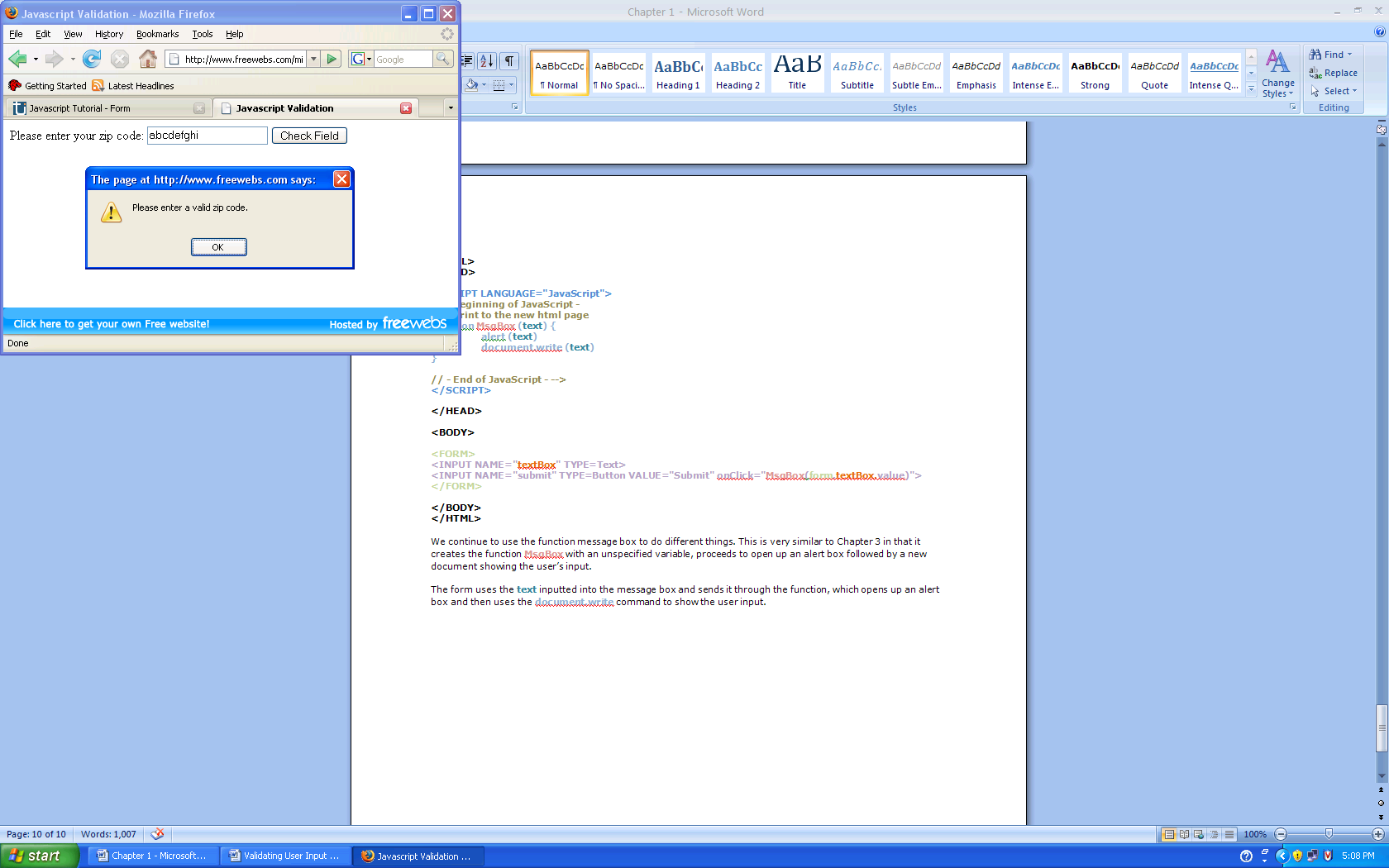
One of the strongest features of Javascript is actually validating what users send in.

# Example 1: Validating Zip codes

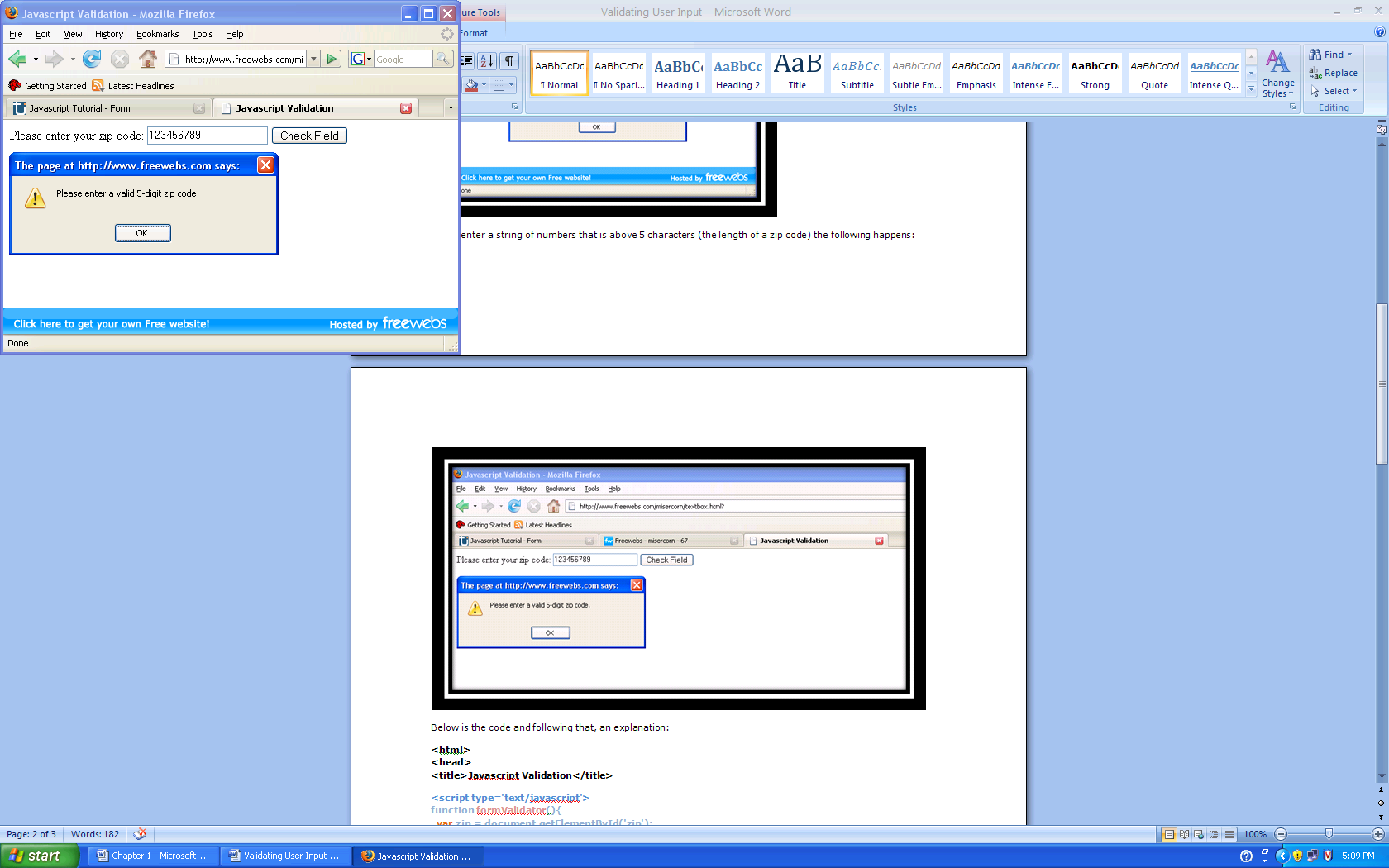
To do so, we utilize a variety of scripts. Below are screenshots of this particular script in action.



If you enter something that is not numbers, the following happens:



If you enter a string of numbers that is above 5 characters (the length of a zip code) the following happens:



Below is the code and following that, an explanation:

<html>  
<head>  
<title>Javascript Validation</title>

<script type=’text/javascript’>  
function formValidator(){  
var zip = document.getElementById(‘zip’);

if(isNumeric(zip, “Please enter a valid zip code.”)){  
 if(lengthRestriction(zip, 5, 5)){  
 return true;  
}  
}  
return false;  
}  
function isNumeric(elem, helperMsg){  
varnumericExpression = /^[0-9]+$/;  
 if(elem.value.match(numericExpression)){  
 return true;  
 }else{  
 alert(helperMsg);  
elem.focus();  
 return false;  
}  
}  
function lengthRestriction(elem, min, max){  
varuInput = elem.value;  
 if(uInput.length>= min&&uInput.length<= max){  
 return true;  
 }else{  
 alert(“Please enter a valid “ +max+ “-digit zip code.”);  
elem.focus();  
 return false;

}  
}  
</script>

<formonsubmit=’return formValidator()’>  
Please enter your zip code:   
<input type=’text’ id=’zip’>  
<input type=’submit’ value=’Check Field’>  
</form>

</body>  
</html>

**A regular expression** (regex or regexp for short) is a special text string for describing a search pattern. Regular expressions are wildcards on steroids.

Ref: http://www.regular-expressions.info/quickstart.html

wildcard notations such as \*.txt to find all text files in a file manager. The regex equivalent is .\*\.txt$.

Eleven characters with special meanings are often called "metacharacters".:

* the opening square bracket [
* the backslash \
* the caret ^,
* the dollar sign $
* the period or dot .
* the vertical bar or pipe symbol |
* the question mark ?,
* the asterisk or star \*
* the plus sign +
* the opening round bracket ( and the closing round bracket )

Anchors do not match any characters. They match a position.

^ matches at the start of the string

$ matches at the end of the string

Shorthand Character Classes

\d matches a single character that is a digit,

\w matches a "word character" (alphanumeric characters plus underscore)

\s matches a whitespace character (includes tabs and line breaks)

Character Sets

use a hyphen inside a character class to specify a range of characters

[0-9] matches a single digit between 0 and 9. You can use more than one range

[0-9a-fA-F] matches a single hexadecimal digit, case insensitively.

combine ranges and single characters. [0-9a-fxA-FX] matches a hexadecimal digit or the letter X

Repetition

* a repetition operator like {2,4}: repeat 2, 3 or 4 times
* The asterisk or star (\*) tells the engine to attempt to match the preceding token zero or more times.
* The plus (+) tells the engine to attempt to match the preceding token once or more

More frequently used expressions can be found at http://www.regular-expressions.info/examples.html

In order for all of the validation scripts to be applied to this one field, we use a master **formValidator** function. Because our field is a zip code field, the input needs to be all numbers, and must be 5 characters in length for it to be considered a valid zip code. In order to check for this, we create the variable “zip.” Variables are used to store information for later use. We use the **getElementbyId** method in order to easily access something with an id. We assign our input an id, this time “zip” in order for the **getElementbyId**method to work. Next we create some comparisons. The following if-then statements reference from more elaborate functions that come later on, and is basically a “master check list” of sorts. If “zip” fails either the “**isNumeric**” or “**lengthRestriction**” functions, then it will not submit. If zip passes all the tests, then it “returns true” and sends.

The function “**isNumeric**” is used to check to see if the user’s input is all numbers. We get lucky here, as Javascript has a built in match system that compares the user entered stuff with our specified standards. We define our standards by creating a variable called “numericExpression” and say that it has to be from 0-9 (which are all the numbers that can be used).A regular expression /^[0-9]+$/ will only match if the string is all numbers and is at least one character long. And by using the handy, already available Javascript match, we can compare the two instantly. If it passes this test, then it moves on to the next test. If it fails, then there is an alert box that says “Please enter a valid zip code.”

Our next function is to check if the user’s input is exactly 5 digits. We create a function, **lengthRestriction**, and define its standards: the input must be between a specified max and min, stated earlier. If it fails this test, then an alert box will appear that asks the user to “Please enter a valid 5-digit zip code).

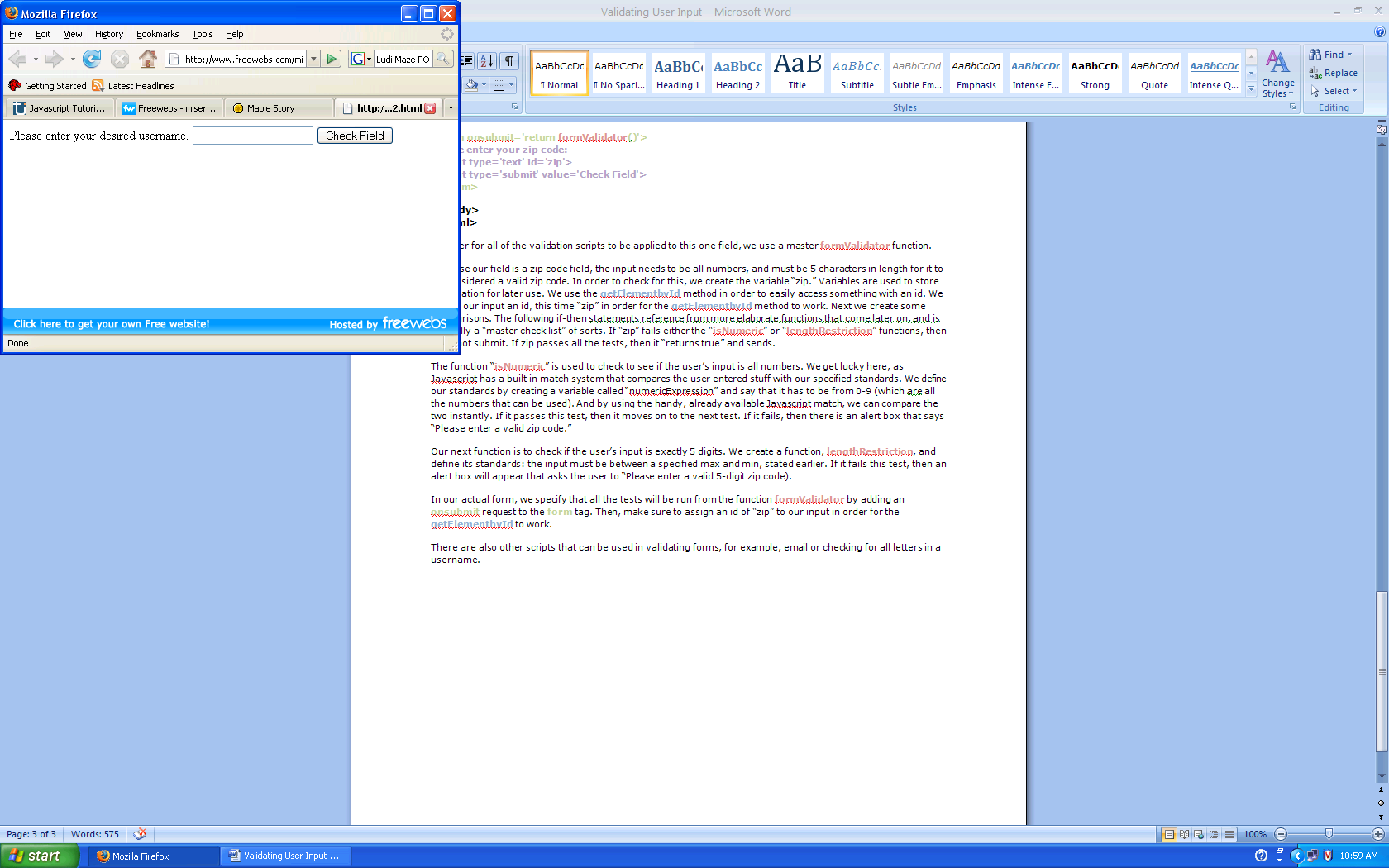
In our actual form, we specify that all the tests will be run from the function **formValidator** by adding an **onsubmit**request to the **form** tag. Then, make sure to assign an id of “zip” to our input in order for the **getElementbyId** to work.

# Example 2: Validating username (letters only)

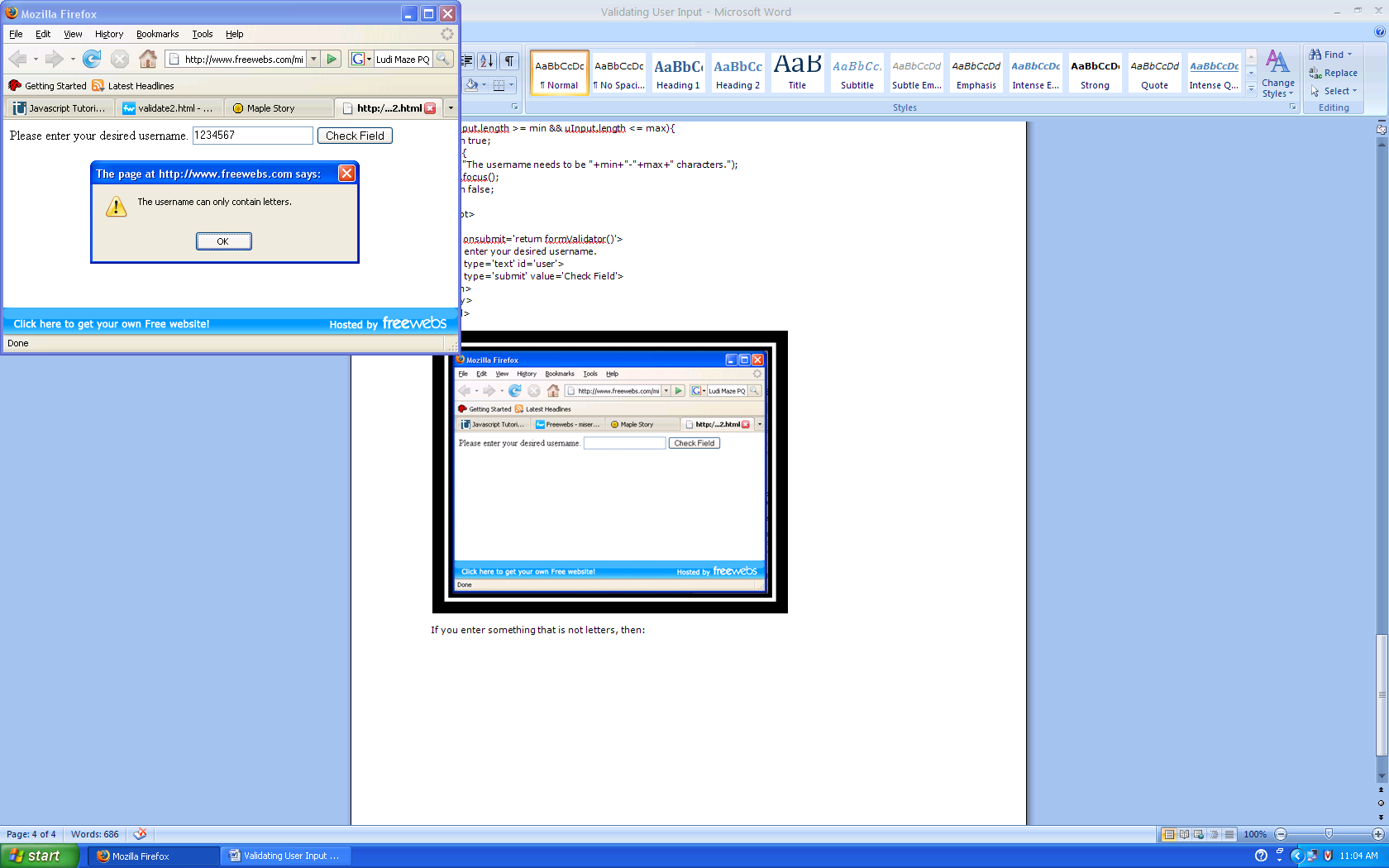
There are also other scripts that can be used in validating forms, for example, checking for all letters in a username. The following script is very similar to the top one, with modifications for a username field:

<script type=’text/javascript’>  
functionformValidator(){  
var user = document.getElementById(‘user’);  
 if(isAlphabet(user, “The username can only contain letters.”)){  
 if(lengthRestriction(user, 6, 8)){  
 return true;  
}}  
return false;  
}  
functionisAlphabet(elem, helperMsg){  
varalphaExp = /^[a-zA-Z]+$/;  
 if(elem.value.match(alphaExp)){  
 return true;  
 }else{  
 alert(helperMsg);  
elem.focus();  
 return false;}  
}  
function lengthRestriction(elem, min, max){  
varuInput = elem.value;  
 if(uInput.length>= min &&uInput.length<= max){  
 return true;  
 }else{  
 alert(“The username needs to be “+min+”-“+max+” characters.”);  
elem.focus();  
 return false;  
}}  
</script>  
  
<form onsubmit=’return formValidator()’>  
Please enter your desired username.  
<input type=’text’ id=’user’>  
<input type=’submit’ value=’Check Field’>  
</form>  
</body>  
</html>

The only script difference is that we change the comparative expression from numbers to letters.



If you enter something that is not letters, then:

  
  
If you enter less than six characters or more than eight, this happens:

# Example 3: Validating email address

The function for checking email is slightly more complex because an email has several components:  
1) A name, containing a mixture of letters and numbers and symbols.  
2) An ‘@’ symbol.  
3) Another name; the provider’s, consisting of letters, numbers, and symbols.  
4) A period.  
5) A top level domain consisting of 2-4 letters. (com, net, gov, us, etc.)  
We must take all of these components into consideration when writing the match expression.

<html>  
<head>  
<title>Javascript Validation</title>

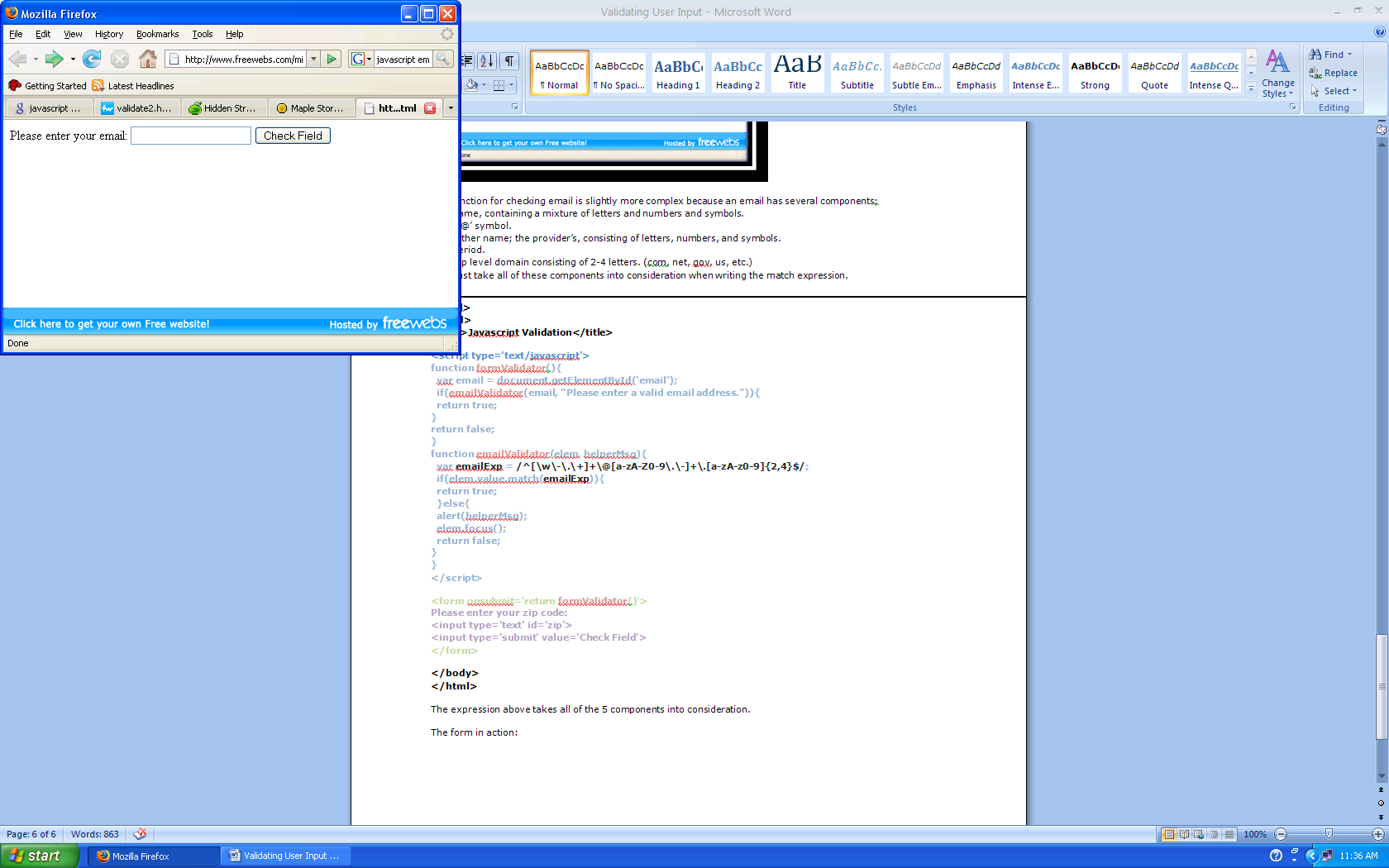
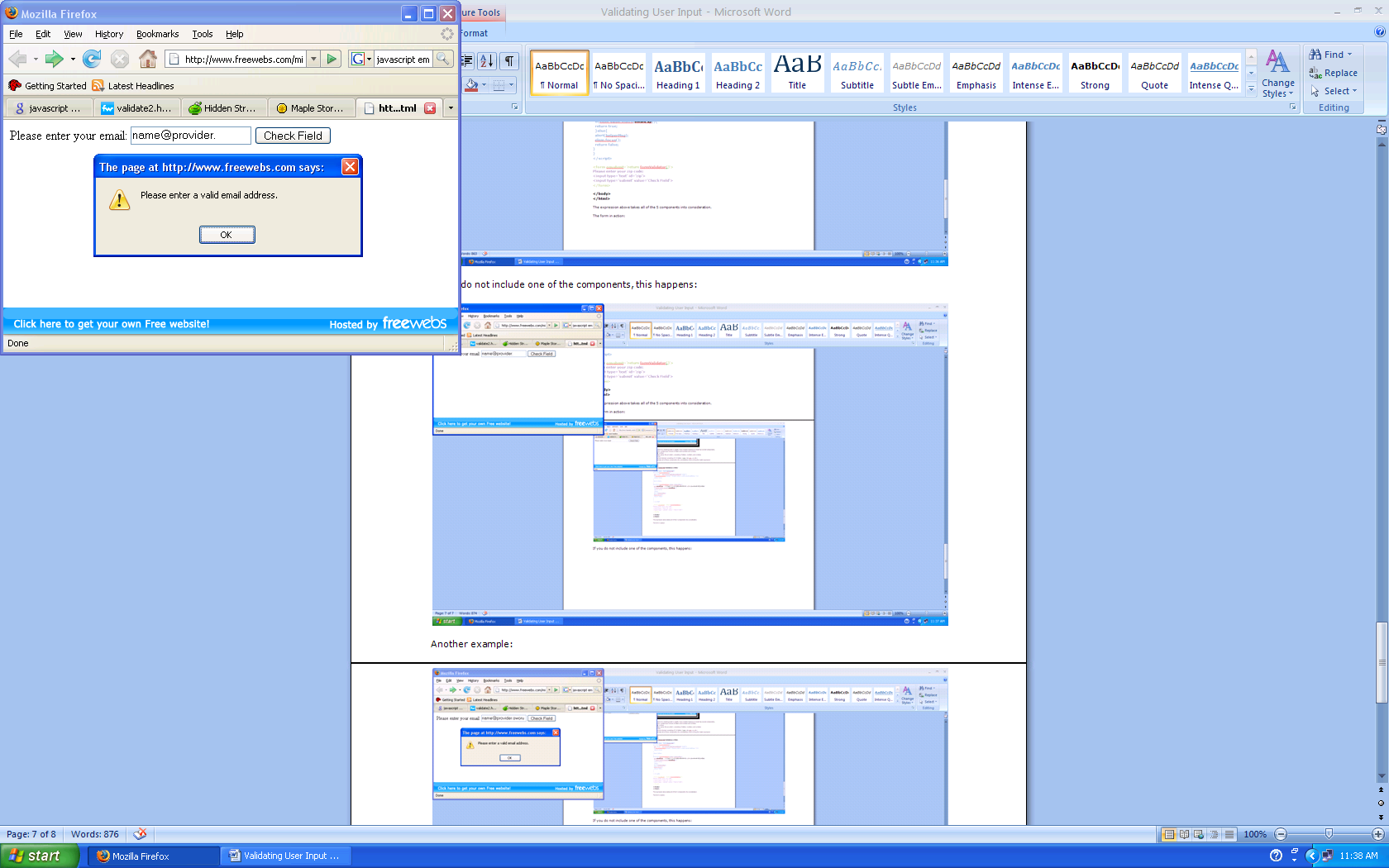
<script type=’text/javascript’>  
function formValidator(){  
var email = document.getElementById(‘email’);  
 if(emailValidator(email, “Please enter a valid email address.”)){  
 return true;  
}  
return false;  
}  
function emailValidator(elem, helperMsg){  
var emailExp = /^[\w\-\.\+]+\@[a-zA-Z0-9\.\-]+\.[a-zA-z0-9]{2,4}$/;  
 if(elem.value.match(emailExp)){  
 return true;  
 }else{  
 alert(helperMsg);  
elem.focus();  
 return false;  
}  
}  
</script>

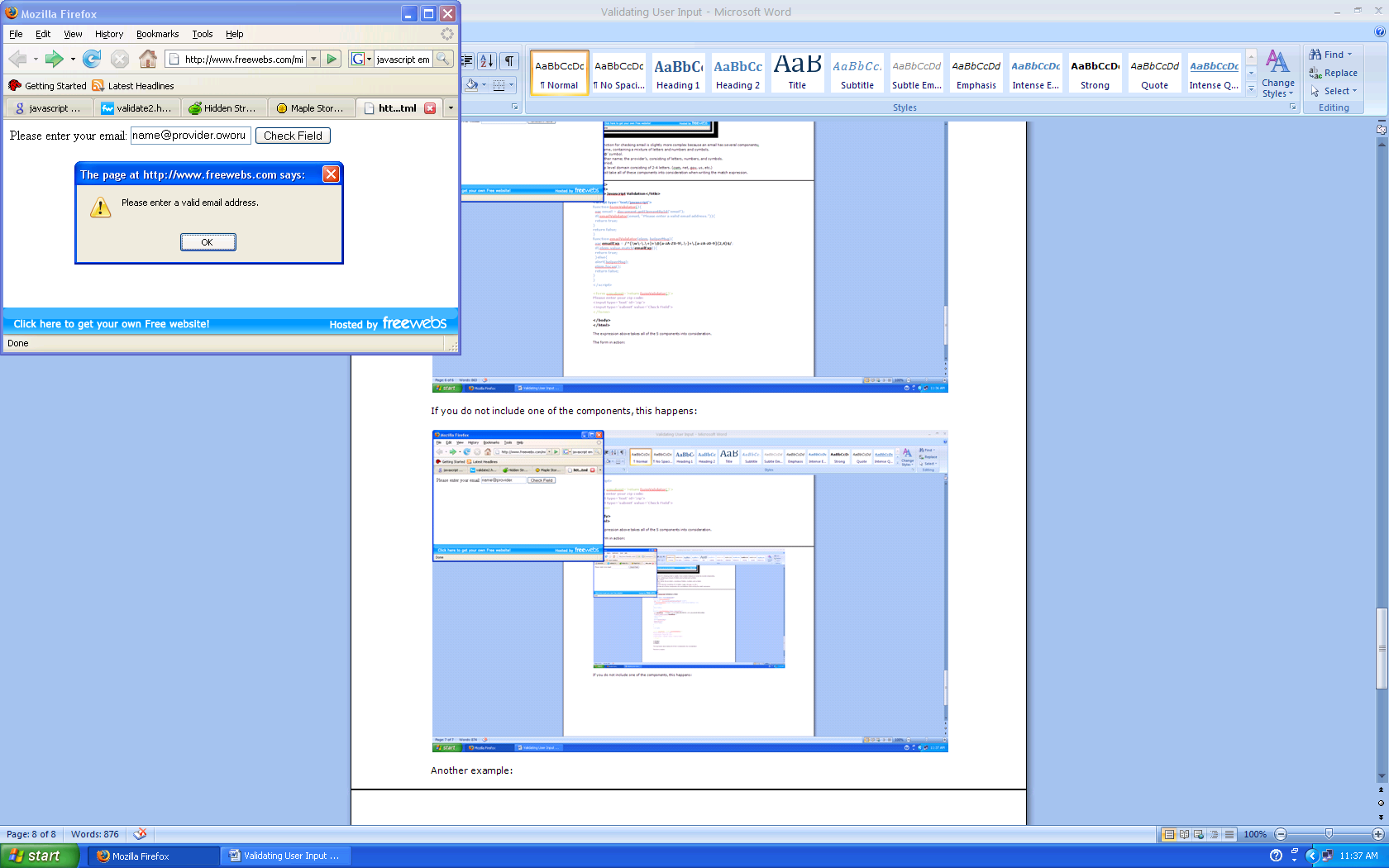
<form onsubmit=’return formValidator()’>  
Please enter your zip code:   
<input type=’text’ id=’zip’>  
<input type=’submit’ value=’Check Field’>  
</form>

</body>  
</html>

The expression above takes all of the 5 components into consideration.

The form in action:

  
  
If you do not include one of the components, this happens:  
  
  
Another example:



# Example 4: Integrating Javascript and PHP

After the javascript of the client side validating the user input fields, the values can be passed to a server script for processing and storing the values, such as in a database. PHP and ASP are two most popular scripting languages. We will use php in the following.

## 4.1 Using onsubmit to validate values and pass them to php page

The onSubmit event is used to validate ALL form fields before submitting it to a php server page. Below is a simple example of how to use the onSubmit event. The JavascriptValidateForm() function will be called when the user clicks the submit button in the form. If the field values are not validated, the submitting to php page will be cancelled. The function ValidateForm() returns either true or false. If it returns true the form will be submitted to a php page, otherwise the submit will be cancelled:

<form method="post" action="FormAccept.php"

onsubmit="return ValidateForm()">

Following is the Javascript in a html file, zipcode\_onsubmit.html, for validating zip code and submit to a Serverscipt.php if the alidation is successful:

<html>

<head>

<title>Javascript Validation</title>

<script type='text/javascript'>

functionformValidator(){

var zip = document.getElementById("zip");

if(isNumeric(zip, "Please enter a valid zip code.")){

if(lengthRestriction(zip, 5, 5)){

document.getElementById("zip").value= zip.value;

document.getElementById("result").innerHTML= "zipcode verified";

return true;

}

}

return false;

}

functionisNumeric(elem, helperMsg){

varnumericExpression = /^[0-9]+$/;

if(elem.value.match(numericExpression)){

return true;

}else{

alert(helperMsg);

elem.focus();

return false;

}

}

functionlengthRestriction(elem, min, max){

varuInput = elem.value;

if(uInput.length>= min &&uInput.length<= max){

return true;

}else{

alert("Please enter a valid " +max+ "-digit zip code.");

elem.focus();

return false;

}

}

</script>

</head>

<formonsubmit='return formValidator()' method="get" action="Serverscript.php">

Please enter your zip code and click Submit:

<input type='text' name='zip' id="zip">

<input type='submit' value='Submit'>

</form>

</body>

</html>

After the zip code is validated, it will be submitted to the php page, Serverscript.php, which is listed below:

<html>

<body>

<?php

$resp=$\_REQUEST["zip"];

echo "Verrified zip code ".$resp." is received and saved in zipCode.txt!";

$file = fopen("zipCode.txt","w+");

fwrite($file, $resp);

fclose($file);

?>

<script language="javascript">

varmyWindow=window.open('','','width=200,height=200')

var txt1="<html><body>The verified and saved result: </br>";

var txt2="</body></html>";

myWindow.document.writeln(txt1);

myWindow.document.writeln("zip code: "+ "<?=$resp;?>"+ "</br>");

myWindow.document.write(txt2);

myWindow.document.close();

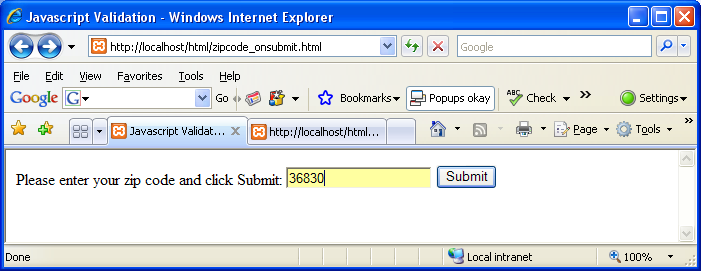
var t=setTimeout("myWindow.close()",3000);

</script>

</body>

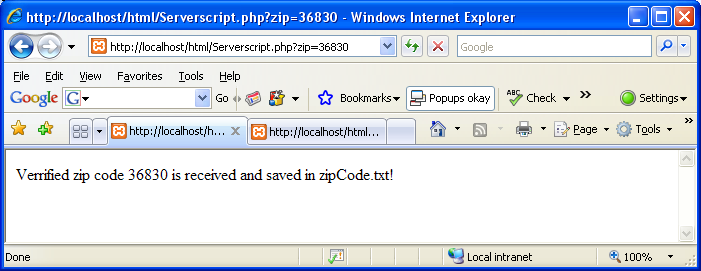
</html>

The following image shows the html page.

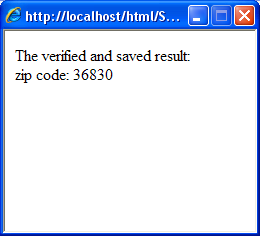


After click submit, the values are validated by the JavaScript function formValidate

This phpscrit displays a new page to show the received zipcode, save it in a file called zipCode.txt and popup a window to show the received zip code if you allow popup windows (this popup window disappears automatically!). Note that the submitted zip code is displayed in the URL as shown below since the get method is used.



Below is the popup window if you allow popup windows that will disappear automatically after a few seconds.



## 4.2 Using onblur to validate values and pass them to php page

There is a need for user to closely interact with a complicated form. For example, a typo in credit card number can be caught as soon as a user finishes typing, a user can readily correct the mistake before marching to the next field. This kind of interactivity is essential for web-based services and applications. The event of onblur is ideal for this purpose because when a user takes one of the following actions:

* Hit a tab
* Move mouse to another field
* Hit any other part of the page

after finishing typing a field that will be validated by corresponding JavaScript function formValidator. A simplified code looks as below:

<input type='text' name='zip' id="zip" onblur='formValidator()'>

Blow is the html file zipcode\_client.html which contains the JavaScript formValidator:

<html>

<head>

<title>Javascript Validation</title>

<script type='text/javascript'>

functionformValidator(){

var zip = document.getElementById("zip");

if(isNumeric(zip, "Please enter a valid zip code.")){

if(lengthRestriction(zip, 5, 5)){

document.getElementById("zip").value= zip.value;

document.getElementById("result").innerHTML= "zipcode verified";

return true;

}

}

return false;

}

functionisNumeric(elem, helperMsg){

varnumericExpression = /^[0-9]+$/;

if(elem.value.match(numericExpression)){

return true;

}else{

alert(helperMsg);

elem.focus();

return false;

}

}

functionlengthRestriction(elem, min, max){

varuInput = elem.value;

if(uInput.length>= min &&uInput.length<= max){

return true;

}else{

alert("Please enter a valid " +max+ "-digit zip code.");

elem.focus();

return false;

}

}

</script>

</head>

<form method="post" action="Serverscript.php">

Please enter your zip code and hit tab key:

<input type='text' name='zip' id="zip" onblur='formValidator()'>

<p id="result"> Validation status: unknown</p>

Click submit after the validation is successful!

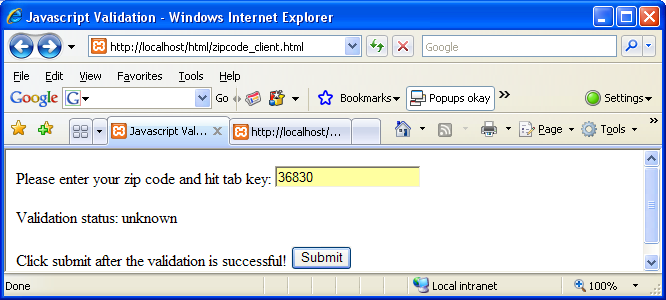
<input type='submit' value='Submit'>

</form>

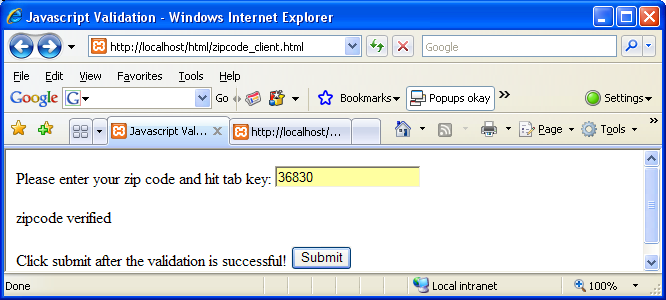
</body>

</html>

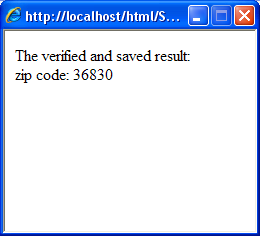
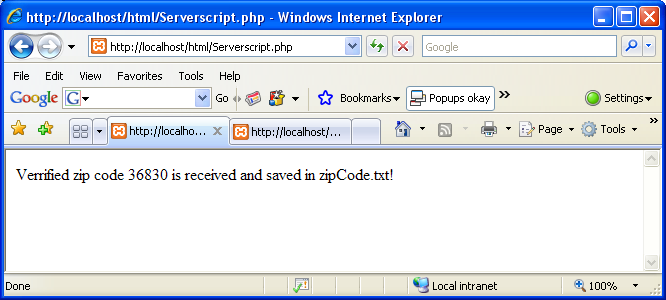
The following image shows the zip code validation status is unknown before a user hit tab or click somewhere else:



after the user hit the tab key, the validation result is shown as following:



When the user is ready to submit, one can click the submit button. The following figures show the php script Serverscript actions:



Note that the post method is used in zipcode\_client.html so that no zip code is shown in the URL.

## 4.3 Using onblur to validate two values and pass them to php page

The following html page 2-f.html verifies both the username and zipcode:

<script type='text/javascript'>

functionuserValidator(){

var user = document.getElementById("user");

if(isAlphabet(user, "The username can only contain letters.")){

if(lengthRestriction(user, 6, 8)){

document.getElementById("user").value= user.value;

document.getElementById("userResult").innerHTML= "username verified!";

return true;

}}

return false;

}

functionisAlphabet(elem, helperMsg){

varalphaExp = /^[a-zA-Z]+$/;

if(elem.value.match(alphaExp)){

return true;

}else{

alert(helperMsg);

elem.focus();

return false;}

}

functionlengthRestriction(elem, min, max){

varuInput = elem.value;

if(uInput.length>= min &&uInput.length<= max){

return true;

}else{

alert("The username needs to be "+min+"-"+max+" characters.");

elem.focus();

return false;

}}

functionzipValidator(){

var zip = document.getElementById("zip");

if(isNumeric(zip, "Please enter a valid zip code.")){

if(lengthRestriction(zip, 5, 5)){

document.getElementById("zip").value= zip.value;

document.getElementById("zipResult").innerHTML= "zip code verified!";

return true;

}

}

return false;

}

functionisNumeric(elem, helperMsg){

varnumericExpression = /^[0-9]+$/;

if(elem.value.match(numericExpression)){

return true;

}else{

alert(helperMsg);

elem.focus();

return false;

}

}

</script>

<body>

<html>

<form action="2-f.php" method="get">

Please enter your desired username (6-8 letters) and click the next field:

<input type='text' name="user" id="user" onblur='userValidator()'>

<p id="userResult"> username validation status unknown! </p>

Please enter your zip code and hit tab key:

<input type='text' name='zip' id="zip" onblur='zipValidator()'>

<p id="zipResult"> zip code validation status unknown! </p>

Click submit when the verifications are complete!

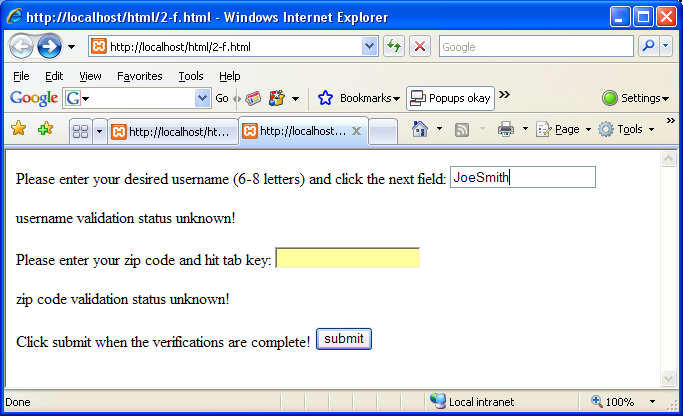
<input type='submit' value='submit'>

</form>

</body>

</html>

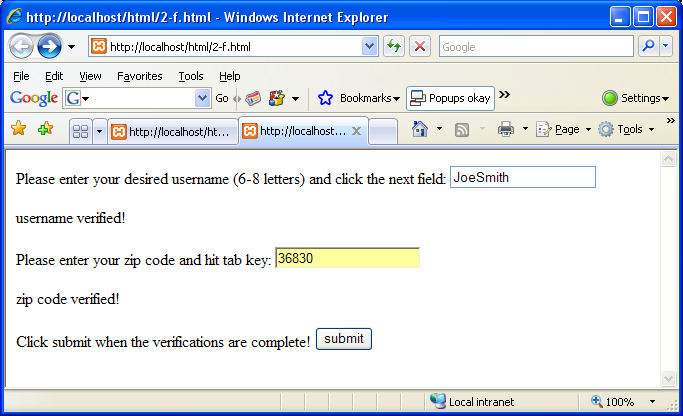
The following figure shows that a username was entered:



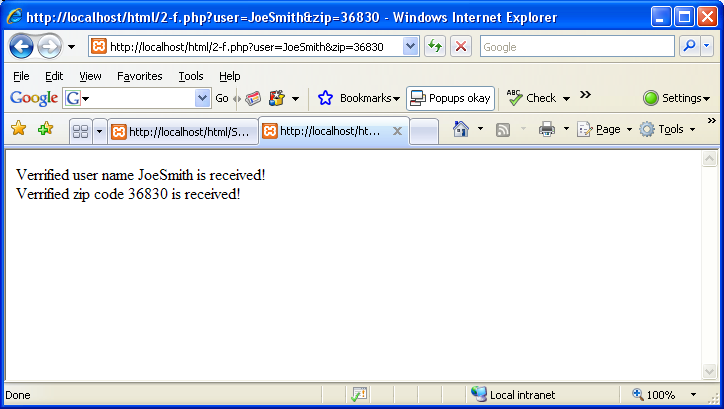
as soon as a user clicks the zip code field, the previously entered username will be validated by JavaScript function userValidator. The validation result is shown below:

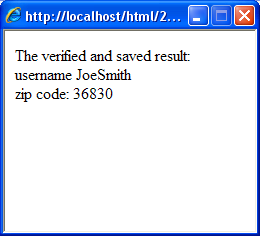


The user can keep typing zipcode and then hit tab key. The zip code validation is carried out by zipValidator function as shown below:



When the user is ready, one can click the submit button and the 2-f.php will receive the two fields as shown below:





Note that the get method is used to pass both fields to the php page. The following is the 2-f.php file:

<html>

<body>

<?php

$user\_r=$\_REQUEST["user"];

$zip\_r=$\_REQUEST["zip"];

echo "Verrified user name ".$user\_r. " is received!"."</br>";

echo "Verrified zip code ".$zip\_r. " is received!"."</br>";

$file = fopen("test.txt","w+");

fwrite($file, "<username>".$user\_r."</username>"."\n");

fwrite($file, "<zipcode>".$zip\_r."</zipcode> "."\n");

fclose($file);

// echo $response

?>

<script language="javascript">

varmyWindow=window.open('','','width=200,height=200')

// varnewDoc=myWindow.document.open("text/html","replace");

var txt1="<html><body>The verified and saved result: </br>";

var txt2="</body></html>";

myWindow.document.writeln(txt1);

myWindow.document.writeln("username "+ "<?=$user\_r;?>"+"</br>");

myWindow.document.writeln("zip code: "+ "<?=$zip\_r;?>");

myWindow.document.write(txt2);

myWindow.document.close();

var t=setTimeout("myWindow.close()",5000);

</script>

</body>

</html>

The username and zip code are saved in the file test.txt using xml format as shown below:

<username>JoeSmith</username>

<zipcode>36830</zipcode>

## 4.4 Using onchange to collect values and pass them to php page

In the previous cookie lab, we learned an example about cookie and JavaScript which use onchange method to pass value to php.

Now I would like to explain this example detailed but you do not need do it again in this lab report.

First, we use following html codes to collect the user name, and then data will be passed to JavaScript when the content is changed or user clicks the submit button.

<td>Enter your name: </td>

<td>

<input type="text" size="30" id="userName" onchange="GetInfo(this.value)"/>

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

</td>

Second, we create GetXmlHttpObject() to pass the data to setcookie.php.

function GetInfo(str)

{

name=str

xmlHttp=GetXmlHttpObject();

if (xmlHttp==null)

{

alert ("Your browser does not support AJAX!");

return;

}

var url="setcookie.php";

url=url+"?q="+str;

url=url+"&sid="+Math.random();

xmlHttp.onreadystatechange=stateChanged;

xmlHttp.open("GET",url,true);

xmlHttp.send(null);

}

Third, we use the setcookie.php to receive and process the data in this example; then we set cookies for each user and echo the cookie value.

<?php

$q=$\_GET["q"];

echo($q);

echo "<br />";

if ($q=="A")

{

setcookie("userA","$q",time()+3600);

if (isset($\_COOKIE["userA"]))

{

echo "Welcome " . $\_COOKIE["userA"] . "!<br />";

}

else

echo "Welcome friend!<br />";

}

else if ($q=="B")

{

setcookie("userB","$q",time()+3600);

if (isset($\_COOKIE["userB"]))

{

echo "Welcome " . $\_COOKIE["userB"] . "!<br />";

}

else

echo "Welcome friend!<br />";

}

else if ($q=="C")

{

setcookie("userC","$q",time()+3600);

if (isset($\_COOKIE["userC"]))

{

echo "Welcome " . $\_COOKIE["userC"] . "!<br />";

}

else

echo "Welcome friend!<br />";

}

else

{

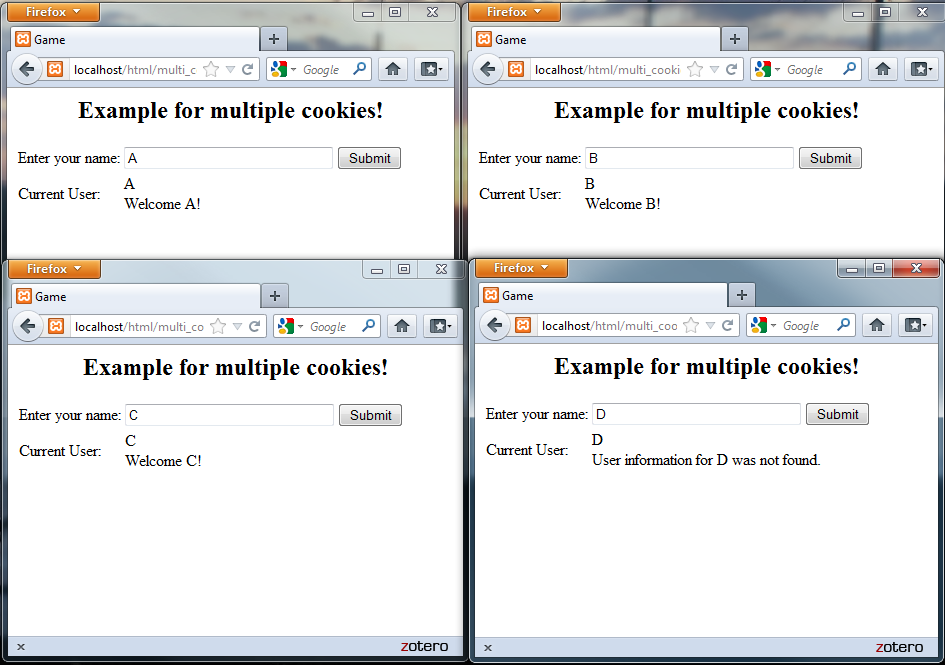
echo("User information for ");

echo($q);

echo(" was not found.");

}

?>



On Your Own Exercise

Write two codes, show how to run codes (screen captures) and submit codes using zipped format:

1. 4-f.html (contains a JavaScript): key functions

Verifying the followings using on blur and submit the four fields to 4-f.php

1. First name
2. Last name
3. Telephone number (10 digits including area code)
4. Address (including State and zip code)
5. 4-f.php: key functions
6. receive those 4 fields
7. acknowledge the 4 fields
8. save the fields using xml format in a file user profile.txt