e-PG Pathshala

Subject: Computer Science

Paper: Web Technology

Module: JavaScript and Event Handling

Module No: CS/WT/19

Quadrant 1 – e-text

Learning Objectives

In the last module, we tried to understand about HTML DOM object hierarchy and learnt about how to access HTML elements using DOM. Moreover we have leamt to work with DOM objects with few examples.

In this module we will work with HTML forms and event handling. Moreover we will understand JavaScript Event Handling mechanisms and also we will learn about how to work with JavaScript Cookies.

JavaScript - Event Handlers

JavaScript's interaction with HTML is handled through events that occur when the user or the browser manipulates a page. When the page loads, it is called an event. It allows us to execute an action of code when the event occurs. All the event handlers in JavaScript start with the keyword ***on***, and each event handler deals with a certain type of event. (For eg, **onClick**)

How it works:

Step 1:	User moves mouse over object
Step 2:	Event senses that something happened to the object
Step 3:	JavaScript (Event handler) tells the object what to do
Step 4:	Using DOM, the handler locates the object on the web page
Step 5:	Object's image source is changed

Here Step is an action where the user moves the mouse over an object. This action is sensed as an event, and JavaScript handles this event with an event handler. Using DOM, the object is located and the reaction to the event is performed as toggling the source image to anaother one. This is shown in the below diagram.

$$\begin{tikzcd} \text{Action} \arrow[r] & \text{Event} \arrow[r] & \text{JavaScript} \arrow[r] & \text{DOM} \arrow[r] & \text{Reaction} \end{tikzcd}$$

src="button_off.gif   onMouseOver   toggle()   document.img.button1   Src="button_on.gif"

The following is the list of event handlers described along with the Objects on which they can be applied and when they are triggered as given in Table 19.1.

TABLE 19.1: Event Handlers

Event handler	**Applies to**	**Triggered when**
onAbort	Image	The loading of the image is cancelled.
onBlur	Button, Checkbox, FileUpload, Layer, Password, Radio, Reset, Select, Submit, Text, TextArea, Window	The object in question loses focus (e.g. by clicking outside it or pressing the TAB key).
onChange	FileUpload, Select, Text, TextArea	The data in the form element is changed by the user.
onClick	Button, Document, Checkbox, Link, Radio, Reset, Submit	The object is clicked on.
onDblClick	Document, Link	The object is double-clicked on.
onDragDrop	Window	An icon is dragged and dropped into the browser.
onError	Image, Window	A JavaScript error occurs.
onFocus	Button, Checkbox, FileUpload, Layer, Password, Radio, Reset, Select, Submit, Text, TextArea, Window	The object in question gains focus (e.g. by clicking on it or pressing the TAB key).
onKeyDown	Document, Image, Link, TextArea	The user presses a key.
onKeyPress	Document, Image, Link, TextArea	The user presses or holds down a key.
onKeyUp	Document, Image, Link, TextArea	The user releases a key.

onLoad	Image, Window	The whole page has finished loading.
onMouseDown	Button, Document, Link	The user presses a mouse button.
onMouseMove	None	The user moves the mouse.
onMouseOut	Image, Link	The user moves the mouse away from the object.
onMouseOver	Image, Link	The user moves the mouse over the object.
onMouseUp	Button, Document, Link	The user releases a mouse button.
onMove	Window	The user moves the browser window or frame.
onReset	Form	The user clicks the form's Reset button.
onResize	Window	The user resizes the browser window or frame.
onSelect	Text, Textarea	The user selects text within the field.
onSubmit	Form	The user clicks the form's Submit button.
onUnload	Window	The user leaves the page.

Now let us demonstrate some of the event type with examples and the output of it.

onClick Event Type

This event occurs when a user clicks the left button of his mouse.

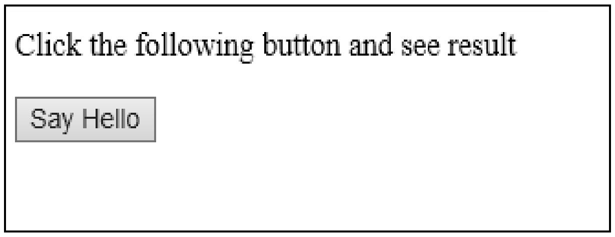
Example 1

```
<html>
<head>
< script type="text/javascript">
 function sayHello()
{
alert("Hello World")
}
</script>
```

```
</head>
<body>
<p>Click the following button and see result</p>
<form>
<input type="button" onclick="sayHello()" value="Say Hello" />
</form>
</body>
</html>
```

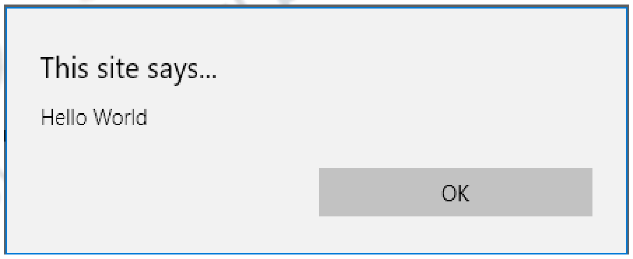
On clicking the button it calls the function sayHello() which alerts a message as shown in FIgure 19.1 and Figure 19.2.

Output:



Click the following button and see result

Figure 19.1 Output of onClick Event Type



This site says...

Hello World

Figure 19.2 Output of onClick Event Type

Example 2

```
<html>
<head>
<title> color world </title>
<script language = "JAVASCRIPT">
function color(f, a)
{
document.bgColor=a;
alert(a);
}
</script>
</head>
<body bgcolor=gray>
<form name=myform>
<input type = button value=red  onClick=color(this.form,"RED")>
<input type = button value=orange  onClick=color(this.form,"ORANGE")>
<input type = button value=green  onClick=color(this.form,"GREEN")>
<input type = button value=violet  onClick=color(this.form,"VIOLET")>
</form>
</body>
</html>
```

This example is to demonstrate the onClick event type which changes the background color of the document on clicking the respective buttons as shown in Figure 19.3 and Figure 19.4.

Output

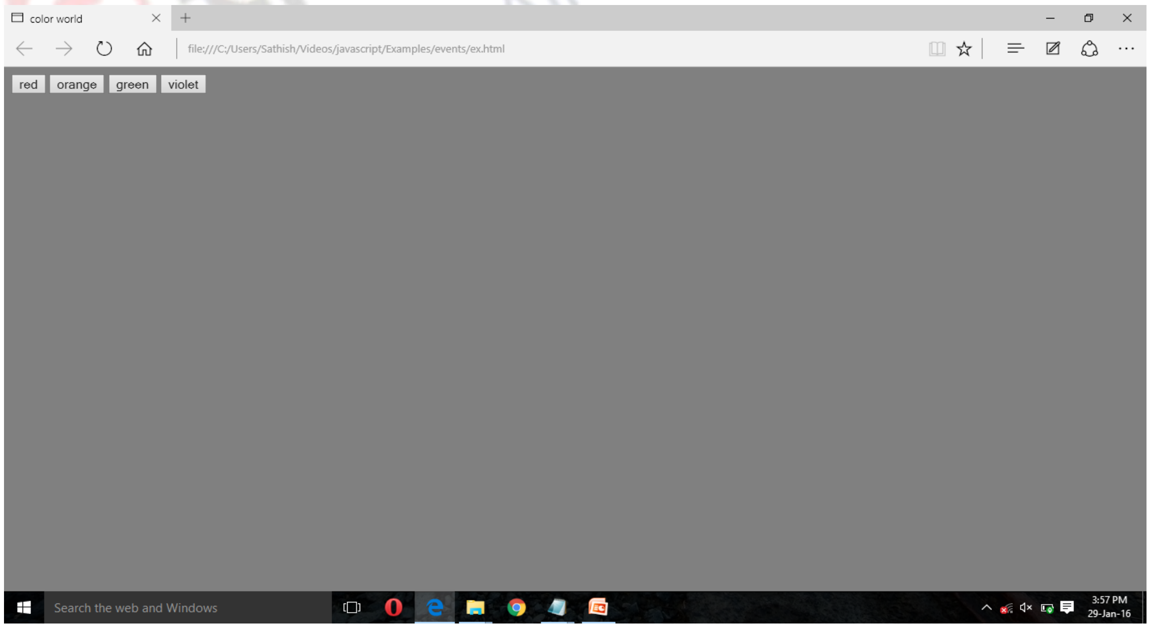
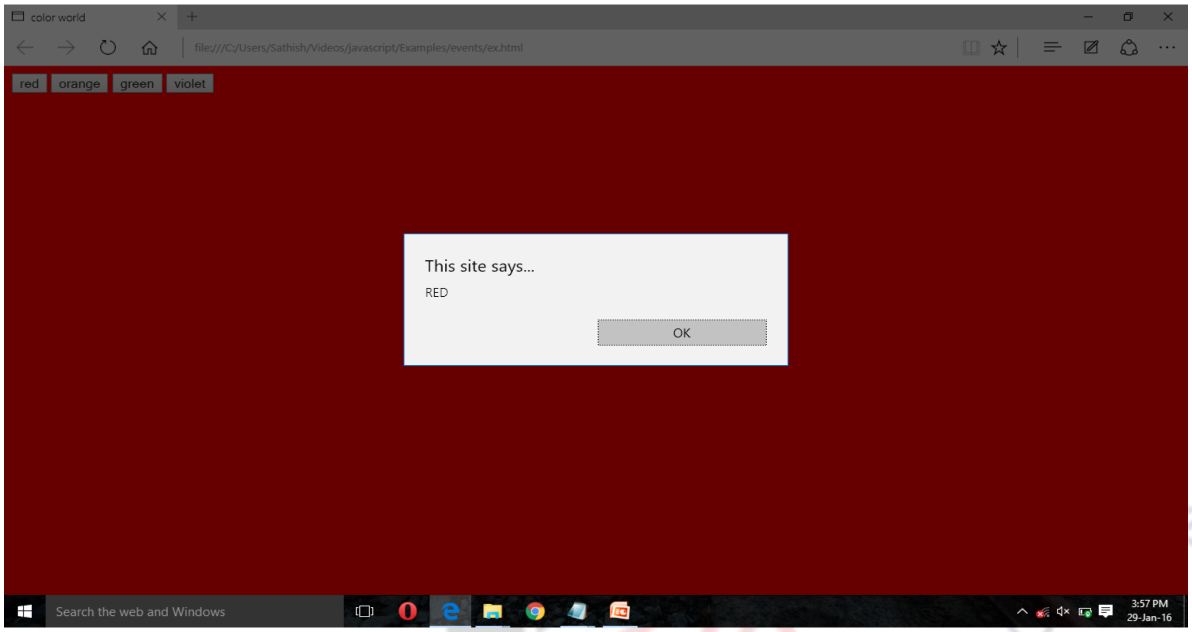

color world
file:///C:/Users/Sathish/Videos/javascript/Examples/events/ex.html
red
orange
green
violet
Search the web and Windows
3:57 PM
29-Jan-16

Figure 19.3 Output of onClick Event Type


color world
file:///C:/Users/Sathish/Videos/javascript/Examples/events/ex.html
red
orange
green
violet
Search the web and Windows
3:57 PM
29-Jan-16

This site says...

RED

OK

Figure 19.4 Output of onClick Event Type

onSubmit Event type

The **onSubmit** is an event that occurs when you try to submit a form.

Syntax:

```
<html>
<head>
<script type="text/javascript">
function validate()
 {
all validation goes here
.........
return either true or false
}
</script>
</head>
<body>
 <form method="POST" action="t.cgi" onsubmit="return validate()">
<input type="submit" value="Submit" />
</form>
</body>
</html>
```

The example demonstates that the form is validated when a submit button is clicked as shown in Figure 19.5.

Output:

Figure 19.5

onMouseOver and onMouseOut Event

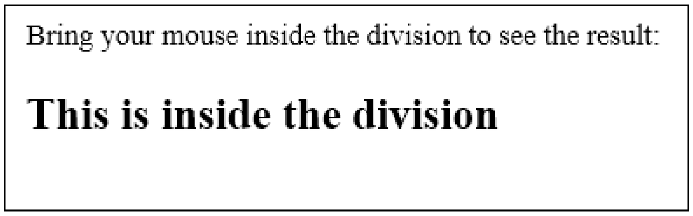
The **onMouseOver** event is triggered when you bring your mouse over any element on the document and the **onMouseOut** event is triggered when you move your mouse out from that element.

Example 1:

```
<html>
<head>
<script type="text/javascript">
 function over() {
document.write ("Mouse Over");
}
function out() {
document.write ("Mouse Out");
}
</script>
</head>
<body>
<p>Bring your mouse inside the division to see the result:</p>
<div onmouseover = "over()" onmouseout= "out()“ >
<h2> This is inside the division </h2>
</div>
</body>
</html>
```

This example demonstrates the onmouseover and onmouseout event types. When the mouse is moved over the text, the function over() is called which displays the message "Mouse Over" and when the mouse is moved out of the text , the function out() is called which displays the message "Mouse Out". The output of this example code is shown in the Figure 19.6 and Figure 19.7.

Output:


Bring your mouse inside the division to see the result:

This is inside the division

Figure 19.6

Mouse Over

Figure 19.7

Example 2:

```
<html>
<head>
<title> image world </title>
</head>
<body >
<a href=ex.html onMouseOver = document.images[0].src = "FLOWER.jpg"
onMouseOut = document.images[0].src = "GRAPES.jpg" >
<img src="GRAPES.jpg" width=160 height=100 border=1>
</a>
</body>
</html>
```

This example toggles the image on thedocument when the mouse is moved over or moved out of the image displayed. The output of this example code is shown in Figure 19.8 and Figure 19.9.

Output:

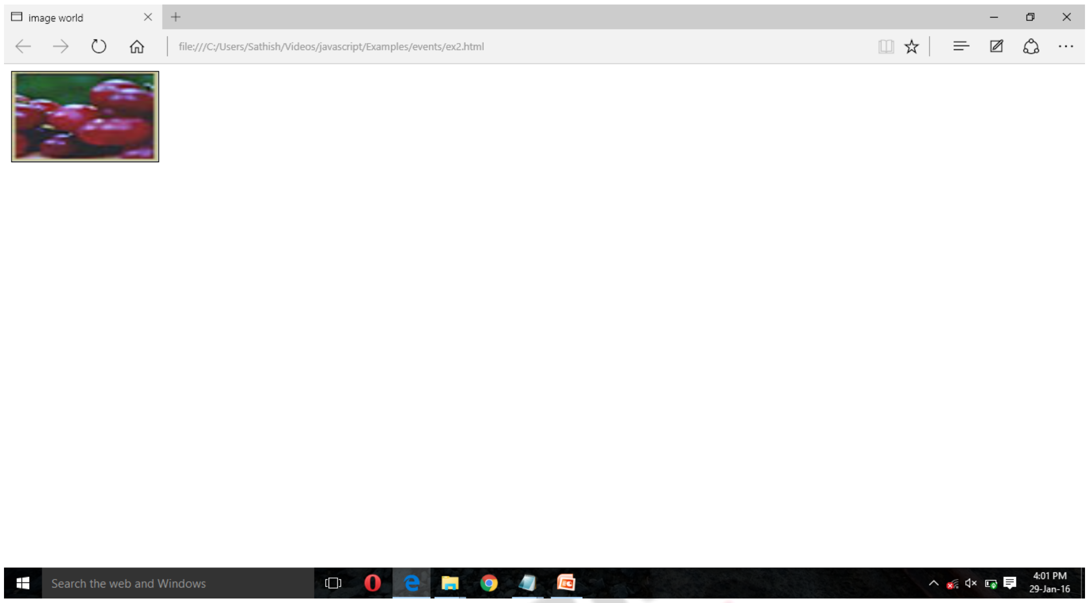


image world
file:///C:/Users/Sathish/Videos/javascript/Examples/events/ex2.html
Search the web and Windows
4:01 PM
29-Jan-16

Figure 19.8

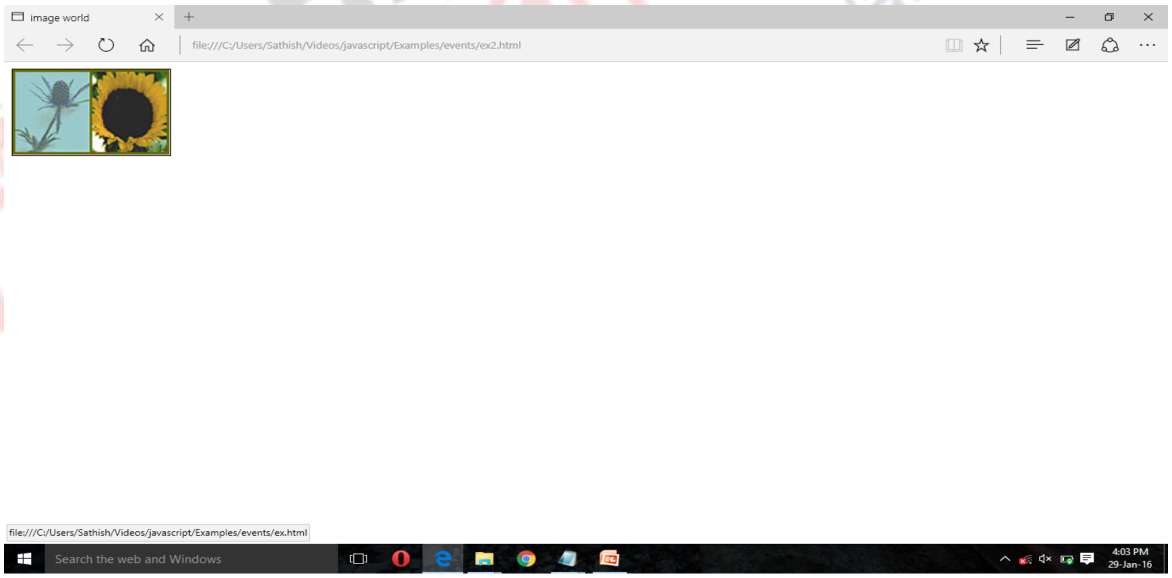


image world
file:///C:/Users/Sathish/Videos/javascript/Examples/events/ex2.html
file:///C:/Users/Sathish/Videos/javascript/Examples/events/ex.html
Search the web and Windows
4:03 PM
29-Jan-16

Figure 19.9

JavaScript Form Validation

HTML form validation can be done by writing a script code using JavaScript.

Example:

An example to validate the name to be filled out without leaving it blank is shown below.

```
function validateForm() {
  var x = document.forms["myForm"]["fname"].value;
  if (x == null || x == "") {
      alert("Name must be filled out");
      return false;
   }
}
```

JavaScript Forms

```
<form name="myForm" action="demo_form.asp"  onsubmit="return
validateForm()" method="post">
Name: <input type="text" name="fname">
        <input type="submit" value="Submit">
</form>
```

Example 1:

```
<html>
<head>
<script type="text/javascript">
function newValue()
{
var x=document.forms.myForm
x[0].value="The mere act of aiming at something makes you big"
x[0].select()
}
</script>
</head>
<body>
<body>
<form name="myForm">
<textarea name="myTextarea" rows="10" cols="20">
The roots of education are bitter but the fruit is sweeter
</textarea><br />
<input type="button" onclick="newValue()" value="New Value">
</form></body></html>
```

This example is to demonstrate about validating a form which has a textarea. The textarea is initially shown with the text. On clicking the button, the function newValue() is called which displays another text. The method select() is used to display the textarea as selected. The output is shown in FIgure 19.10.

Output:

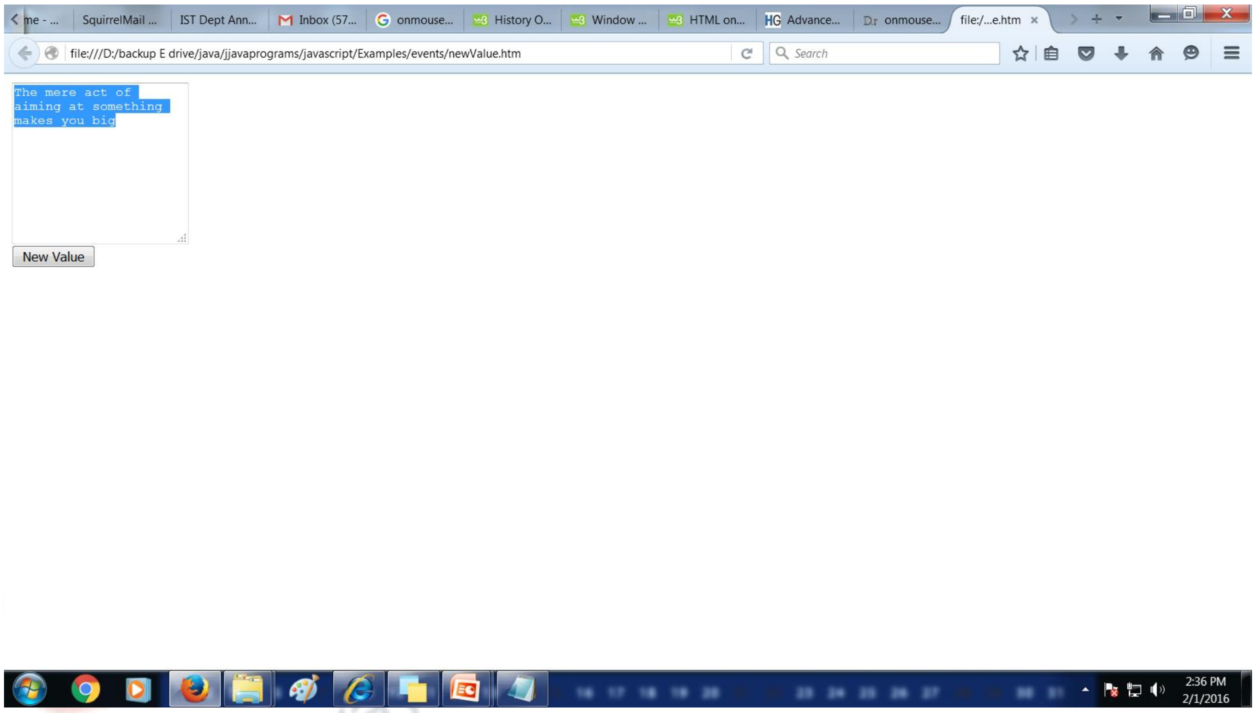

```
The mere act of
aiming at something
makes you big
```
New Value
file:///D:/backup E drive/java/jjavaprograms/javascript/Examples/events/newValue.htm

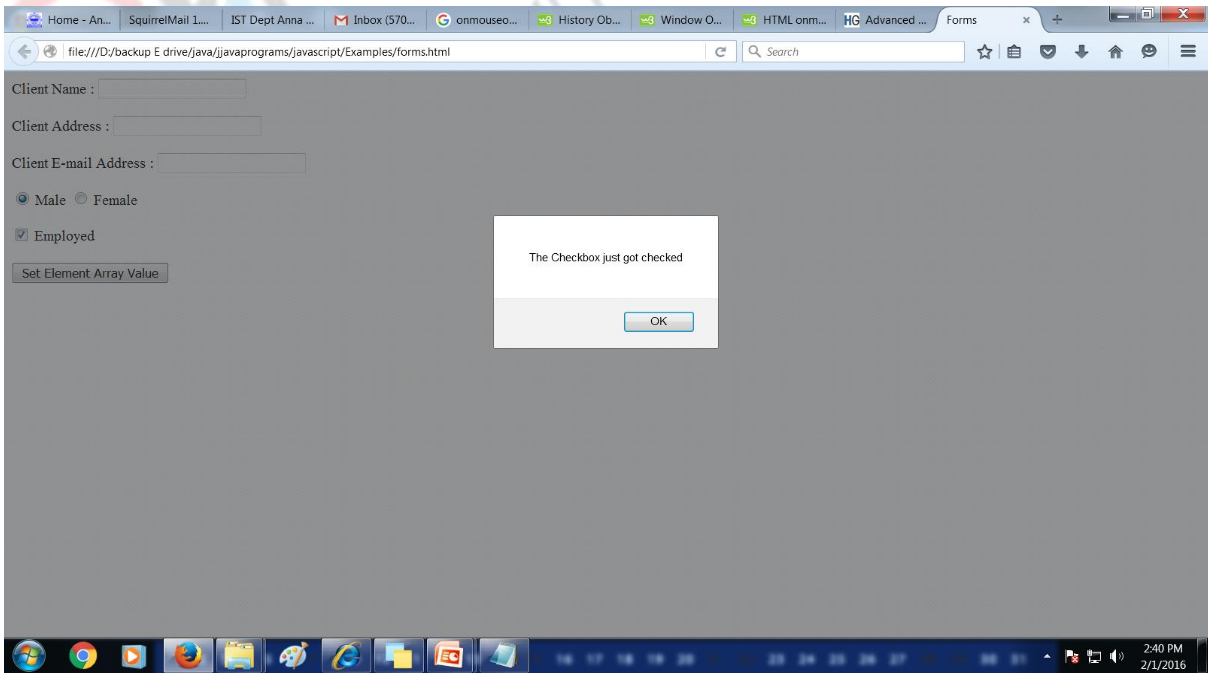
Figure 19.10

Example 2:

```
<html>
<head>
<title> Forms</title>
<!--This code checks the Checkbox When the button is clicked -->
<script Language='JavaScript'>
function Chk(f1)
{
 f1.Check.checked=true;
 window.alert("The Checkbox just got checked");
 f1.Check.checked=false;
 f1.Radio[0].checked=true;
 f1.Radio[1].checked=false;
window.alert("The Radio button just got checked");
}
</script>
</head>
<body>
<form>
Client Name :      <Input Type =Text Name="Text" Value=""><br><br>
Client Address :   <Input Type =Text Name="Text1" Value=""><br><br>
Client E-mail Address : <Input Type =Text Name="Text2" Value=""><br><br>
<Input Type="radio" Name="Radio" Value=""> Male
<Input Type="radio" Name="Radio" Value=""> Female<br><br>
<Input Type="checkbox" Name="Check" Value=""> Employed <br><br>
<Input Type="Button" Name="Bt" Value="Set Element Array Value"
onClick="Chk(this.form)">
</form>
</body>
</html>
```

This example is to demonstrate about validating a form which has a checkbox or a radiobutton. If checkbox is checked or radiobutton is checked, the property "checked" is made true and it will display the message as checked. The output is shown in Figure 19.11.

<u>Output:</u>



file:///D:/backup E drive/java/jjavaprograms/javascript/Examples/forms.html

Client Name :

Client Address :

Client E-mail Address :

● Male ○ Female

✓ Employed

The Checkbox just got checked

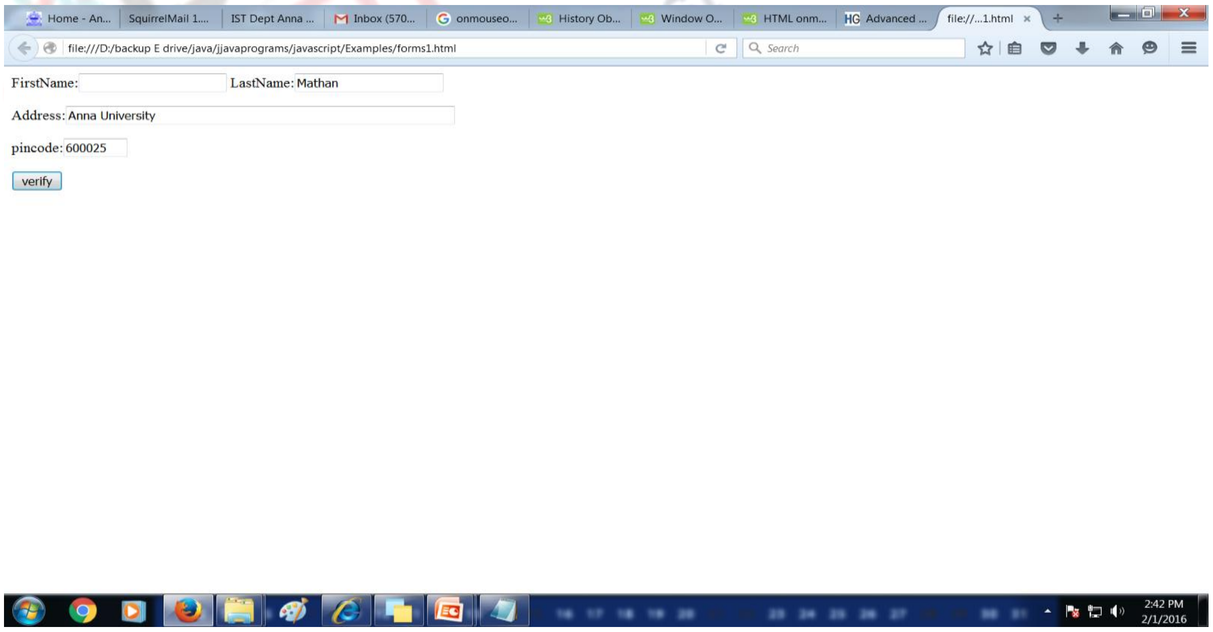
Figure 19.11

Example 3:

```
<html>
<head>
<script language="javascript">
</script>
</head>
<body>
<form>
FirstName:<input type="text" name="Firstname" size=20 onFocus="this.blur();">
LastName:<input type="text" name="Lastname" size=20><p>
Address:<input type="text" name="Address" size=60><p>
pincode:<input type="text" name="Pincode" size=6><p>
<input type="button" name="act" value="verify">
 </form>
</body>
</html>
```

This example code demonstrates the use of function onFocus on a textfield. When a focus is gained on the textfield, it is made to blur using the function blur() which will not allow to enter any data in the textfield. The output is shown in Figure 19.12.

<u>Output:</u>


FirstName: LastName: Mathan

Address: Anna University

pincode: 600025

verify
file:///D:/backup E drive/java/jjavaprograms/javascript/Examples/forms1.html

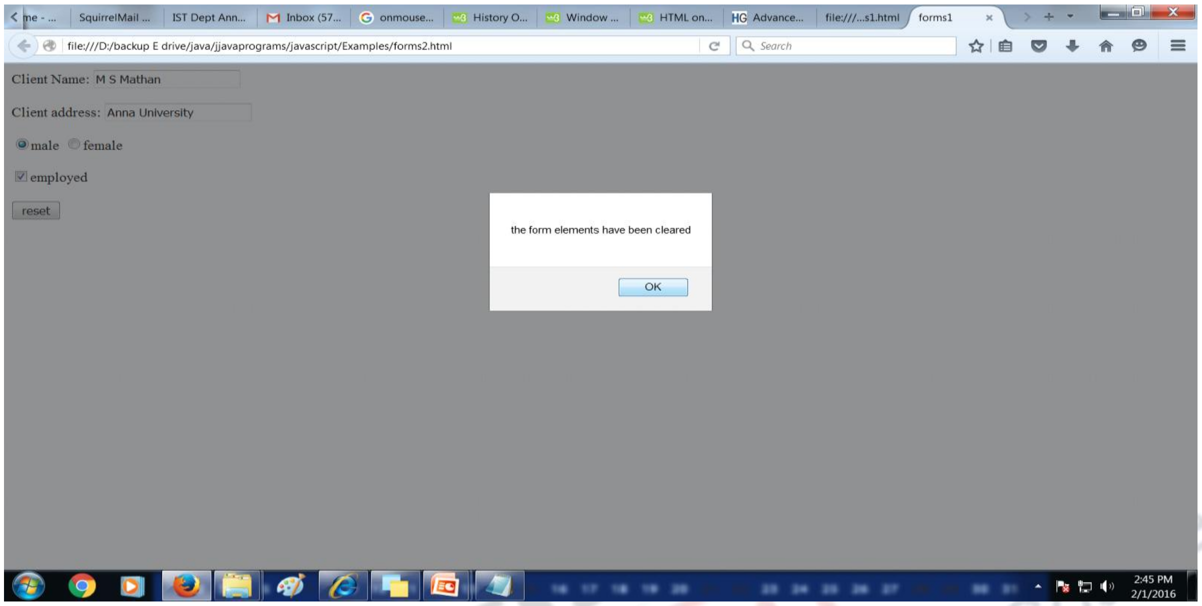
Figure 19.12

Example 4:

```
<html>
<head>
  <title>forms1</title>
<script>
 function func(f1)
  {
   alert("the form elements have been cleared");
  }
</script>
</head>
<body>
<form onReset="func(this.form)">
  Client Name: <input type="text" name="text" value=""><br><br>
  Client address: <input type="text" name="text1" value=""><br><br>
  <input type="radio" name="radio" value="">male
  <input type="radio" name="radio" value="">female<br><br>
  <input type="checkbox" name="check" value="">employed<br><br>
  <input type="reset" name="rst" value="reset">
 </form>
</body>
</html>
```

This example code demonstrates validating a form on a reset button. The event onReset is triggered when a reset button is clicked which clears the form content and displays the message as cleared. The output is shown in Figure 19.13

Output:


file:///D:/backup E drive/java/jjavaprograms/javascript/Examples/forms2.html

Client Name: M S Mathan

Client address: Anna University

● male ○ female

✓ employed

reset

the form elements have been cleared

OK

Figure 19.13

Example 5:

```
<html>
<head>
<title> using text and button objects</title>
<script language="javascript">
  function calculate(form)
     {
       form.results.value= eval(form.entry.value * 10 );
     }
 </script>
</head>
<body>
  <form>
     <input type="text" name="entry" value="">
<input type="button" value="calculate" onClick="calculate(this.form);"
     <br>
<input type="text" name="results" onFocus="this.blur();">
     <br>
  </form>
</body>
</html>
```

This example code demonstrates the use of onClick() and onFocus(). The onFocus is applied on results textfield which will not allow to enter any value using the blur() method. The value entered on the entry field is calculated and shown in the result field. The output of this code is shown in Figure 19.14.

Output:

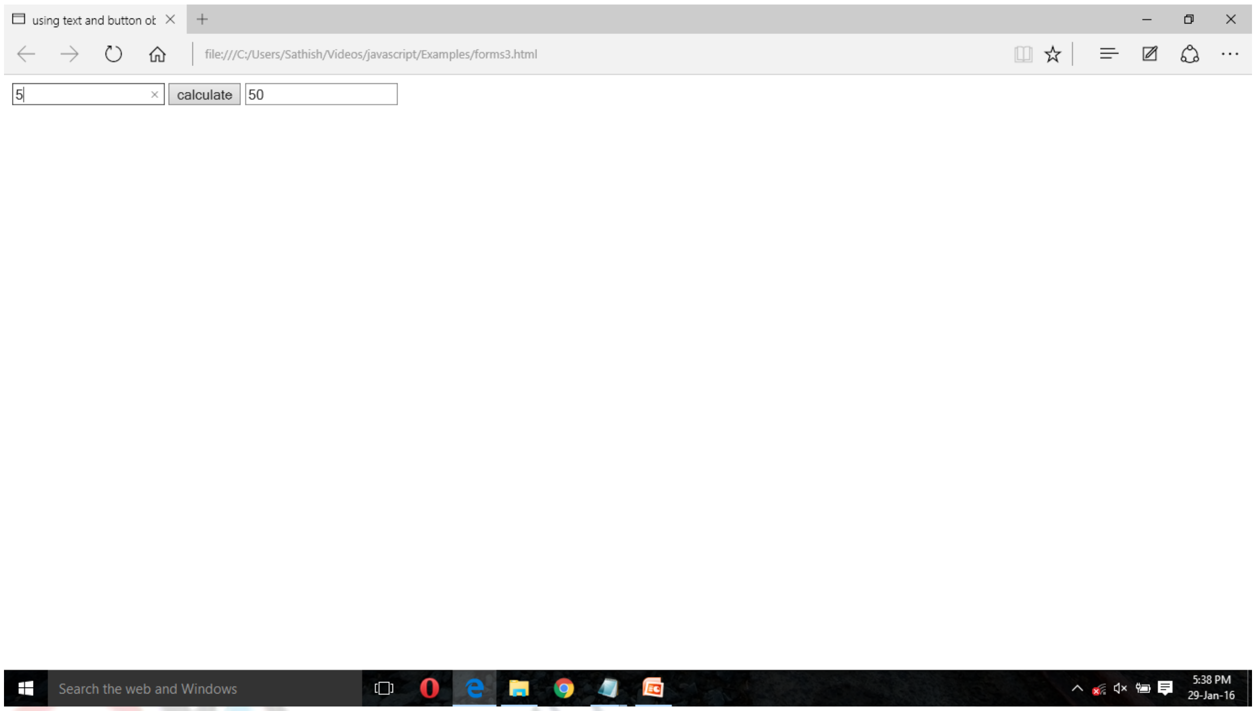

using text and button ob
file:///C:/Users/Sathish/Videos/javascript/Examples/forms3.html
5
calculate
50
Search the web and Windows
5:38 PM
29-Jan-16

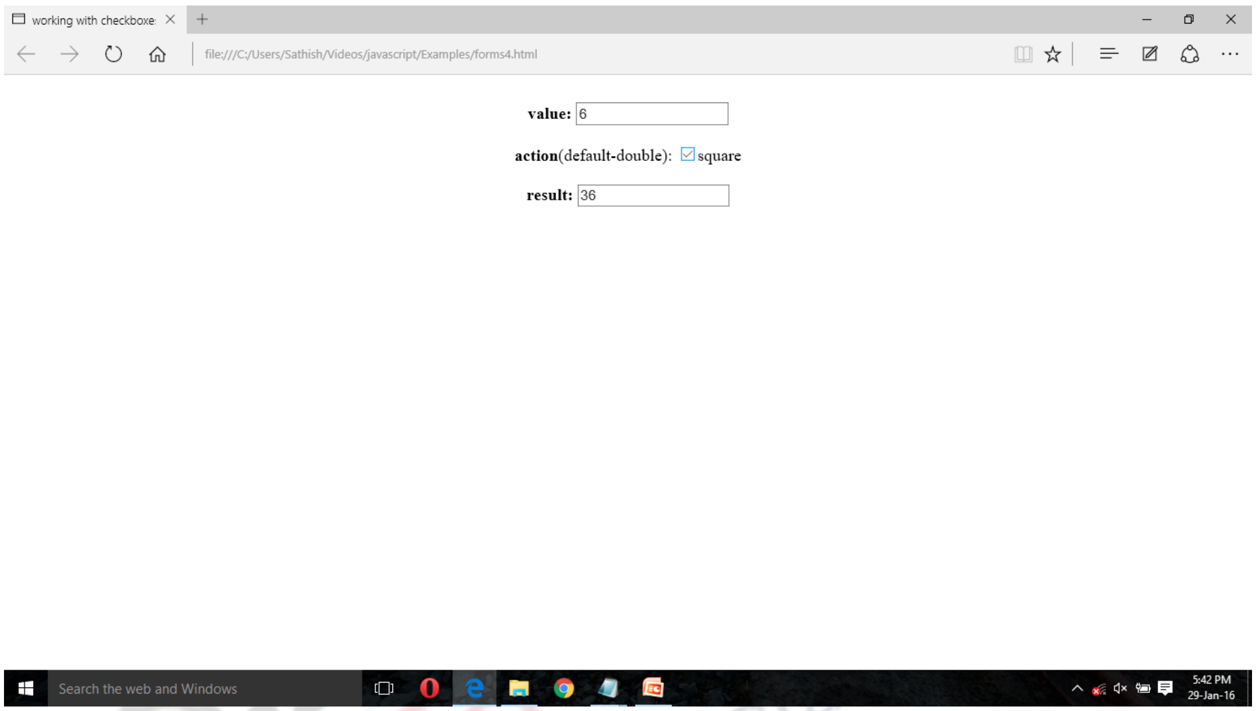
Figure 19.14

Example 6:

```
<html>
<head>
  <title>working with checkboxes</title>
  <script>
    function calculate(form, callingField)
    {
    if(callingField=="result")
    {
    if(form.square.checked)
    {
    form.entry.value= Math.sqrt(form.result.value);
    }
else
{
form.entry.value=form.result.value/2;
}
}
else {
if(form.square.checked)
{
form.result.value = form.entry.value * form.entry.value;
}
else {
form.result.value = form.entry.value*2;
}
}  }
 </script>
 </head>
<body>
   <form>
   <center><br>
   <b> value: </b>
   <input type ="text" name = "entry" value = 0  onChange ="calculate(this.form,this.name);">
         <br><br>
   <b>action</b>(default-double):
   <input type ="checkbox" name = "square"  onClick = "calculate(this.form,this.name);">square
         <br><br>
   <b> result: </b>
 <input type ="text" name = "result" value = 0  onChange = "calculate(this.form,this.name);">
 </center>
 </form>
</body>
</html>
```

This example code demonstrates the use of onClick and onChange event types. There are two textfields (entry and result) and a radiobutton in the form. If the calling field is result field, and if the checkbox is checked then it calculates the squareroot of the value entered in the result field and displays in the entry field else it calculates half of the result value and displays in the entry field. Similarly when the value entered in the entry field changes, based on the checkbox checked or not checked displays the square of the value entered in the entry field or multiplies the value by 2 and displays in the result field. The output of this code is shown in Figure 19.15.

<u>Output:</u>



```
value: 6
action(default-double): ☑ square
result: 36
```

Figure 19.15

JavaScript Cookies

A cookie is used to store information on the user's computer even when the user switches off his/her computer. It is a data that is sent from a web server to a web browser when the user visits a site on a server. It is just a .txt file stored in a user's computer. A cookie can be associated with one or more documents on the web server. More than one cookie can be associated with a document on the web server. Every cookie has a NAME-VALUE pair associated with it. Cookies have an expiration date associated with them.

Cookies are a plain text data record of 5 variable-length fields:

1. **Expires** – The date the cookie will expire. If this is blank, the cookie will expire when the visitor quits the browser.
2. **Domain** -The domain name of your site.

3. **Path** – The path to the directory or web page that set the cookie. This may be blank if you want to retrieve the cookie from any directory or page.
4. **Secure** – If this field contains the word "secure", then the cookie may only be retrieved with a secure server. If this field is blank, no such restriction exists.
5. **Name-Value** - Cookies are set and retrieved in the form of key-value pairs.

Create Cookies

We can create a cookie by assigning a string value to the document.cookie object.

```
document.cookie = "key1=value1;key2=value2;expires=date";
```

Store Cookies

Example

```
<html>
  <head>
     <script type="text/javascript">
        function WriteCookie()
        {
          if( document.myform.customer.value == "" )
                {
            alert("Enter some value!");
            return;
                }
          cookievalue= escape(document.myform.customer.value) + ";";
          document.cookie="name=" + cookievalue;
          document.write ("Setting Cookies : " + "name=" + cookievalue );
        }
        </script>
</head>
  <body>

     <form name="myform" action="">
       Enter name: <input type="text" name="customer"/>
       <input type="button" value="Set Cookie"  onclick="WriteCookie();"/>
     </form>

  </body>
</html>
```

The example shows how a cookie can be created and stored. It retrieves the value from the textfield and stores it as a cookie.

```
<html>
  <head>
 <script type="text/javascript">
      function ReadCookie()
           {
              var allcookies = document.cookie;
             document.write ("All Cookies : " + allcookies );
                    // Get all the cookies pairs in an array
           cookiearray = allcookies.split(';');
                    // Now take key value pair out of this array
           for(var i=0; i<cookiearray.length; i++)
            {
                     name = cookiearray[i].split('=')[0];
                    value = cookiearray[i].split('=')[1];
                    document.write ("Key is : " + name + " and Value is : " + value);
           }
      }
```

The function ReadCookie() retrieve the stored cookie from document.cookie and store in a variable. Then we can use a for loop to read each cookie as a key-value pair.

Summary

This module explains about the JavaScript Event handling mechanisms. This module also explores about working with HTML form validation using HTML DOM and Event handling and finally discusses about JavaScript Cookies.