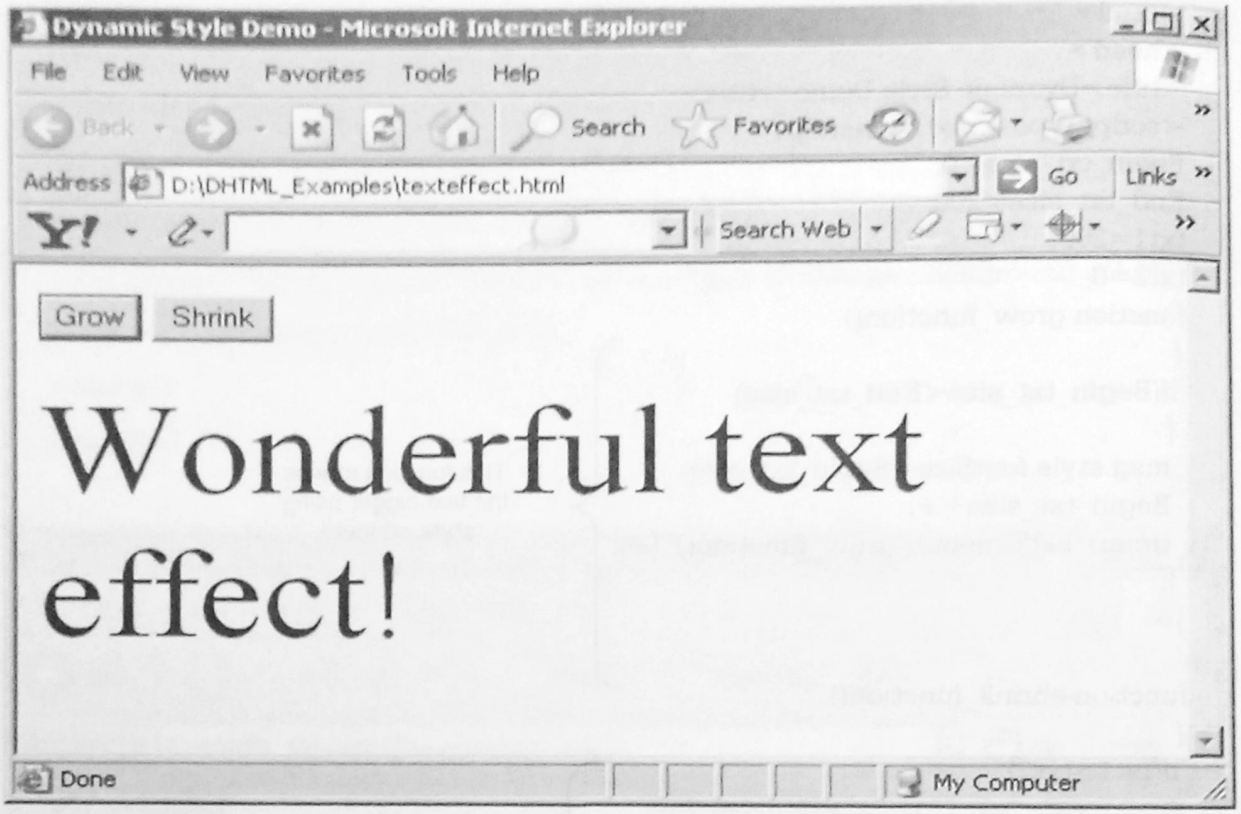
Output



In above program, the **setTimeout** is used to set the time interval between the text change event. To stop the timing interval the **clearTimeout()** function is used.

4.15 Events

- **Event** is an activity that represents a change in the environment. For example mouse clicks, pressing a particular key of keyboard represent the events. Such events are called **intrinsic events**.
- **Event handler** is a script that gets executed in response to these events. Thus event handler enables the web document to respond the user activities through the browser window.
- JavaScript support this special type of programming in which events may occur and these events get responded by executing the event handlers. This type of programming is called event-driven programming.
- Events are specified in lowercase letters and these are case-sensitive.
- The process of connecting event handler to an event is called **event registration**. The event handler registration can be done using two methods -
    - Assigning the tag attributes.
    - Assigning the handler address to object properties.

4.15.1 Events, Attributes and Tags

On occurrence of events the tag attribute must be assigned with some used defined functionalities. This will help to execute certain action on occurrence of particular event.

Commonly used events and tag attributes are enlisted in the following table -

Events	Intrinsic event attribute	Meaning	Associated tags
blur	onblur	Losing the focus.	<button> <input> <a> <textarea> <select>
change	onchange	On occurrence of some change.	<input> <textarea> <select>
click	onclick	When user clicks the mouse button.	<a> <input>
dblclick	ondblclick	When user double clicks the mouse button.	<a> <input> <button>
focus	onfocus	When user acquires the input focus.	<a> <input> <select> <textarea>
keyup	onkeyup	When user releases the key from the keyboard.	Form elements such as input,button,text,textarea and so on.
keydown	onkeydown	When user presses the key down	Form elements such as input,button,text,textarea and so on.
keypress	onkeypress	When user presses the key.	Form elements such as input,button,text,textarea and so on.
mousedown	onmousedown	When user clicks the left mouse button.	Form elements such as input,button,text,textarea and so on.
mouseup	onmouseup	When user releases the left mouse button.	Form elements such as input,button,text,textarea and so on.
mousemove	onmousemove	When user moves the mouse.	Form elements such as input,button,text,textarea and so on.
mouseout	onmouseout	When the user moves the mouse away from some element	Form elements such as input,button,text,textarea and so on.

mouseover	onmouseover	When the user moves the mouse away over some element	Form elements such as input,button,text,textarea and so on.
load	onload	After getting the document loaded.	<body>
reset	onreset	When the reset button is clicked.	<form>
submit	onsubmit	When the submit button is clicked.	<form>
select	onselect	On selection.	<input> <textarea>
unload	onunload	When user exits the document.	<body>

The use of these tag attributes for handling the events is illustrated by following code sample

```
<input type ="button" name="My_button"
onclick="My_fun()"; />
```

Tag attribute **Event handler**

That means when the user clicks the **button** then as an event handler a user defined function **My_fun()** gets called. Basically in this function user writes the instructions that need to be executed on the button click event.

4.15.2 Handling Events from the Body Elements

To understand how events works in JavaScript let us put some form components on the JavaScript. The **onload** event gets activated as soon as the web page gets loaded in the browser's window. Following script along with its output illustrate the **onload** tag attribute.

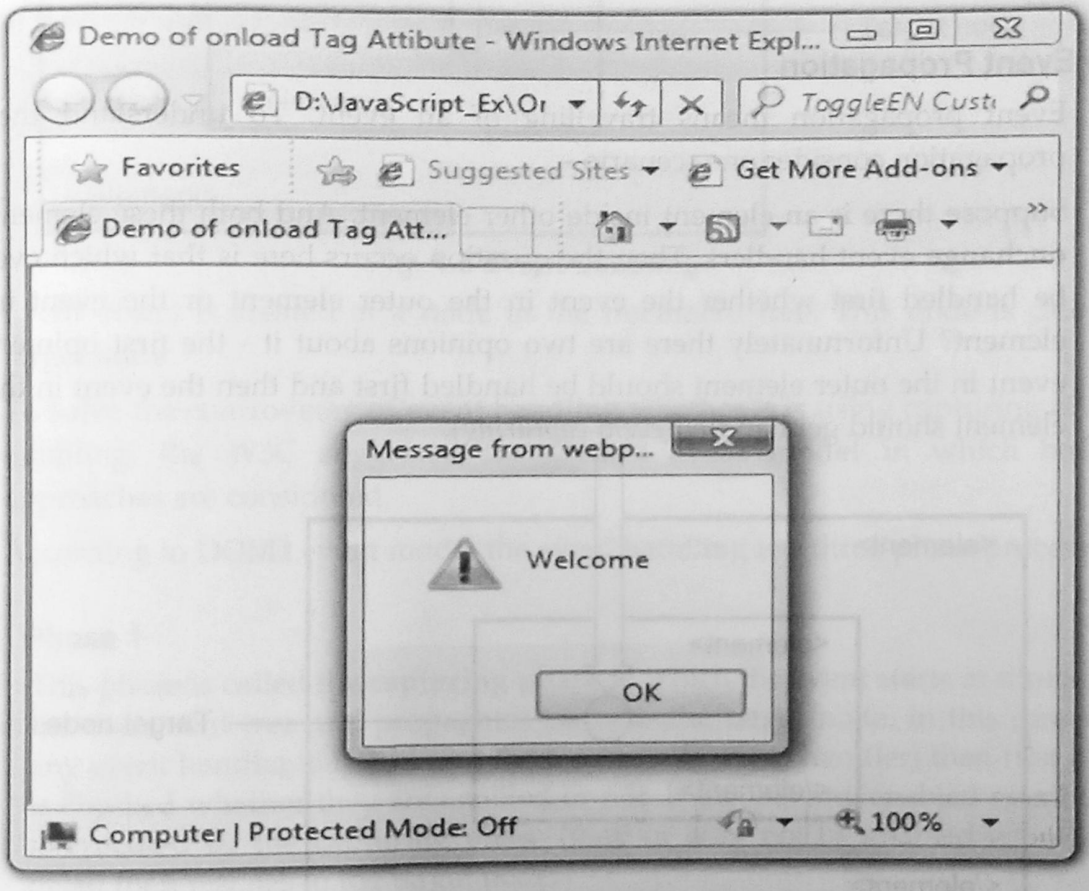
JavaScript[OnloadDemo.html]

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
 <title>Demo of onload Tag Attibute</title>
 <script type="text/javascript">
  function my_fun()
  {
  //This message will be displayed on page loading
```

```
alert("Welcome");
 }
</script>
</head>
<body onload="my_fun()">
</body>
</html>
```

When web document gets loaded on the browser window then **my_fun()** will be called

Output



4.15.3 The DOM2 Event Model

The DOM2 event model provides the generic event system in which the events can be registered. The event flow can be represented in a hierarchical manner. Using DOM2 event model information of each event can be obtained.

The DOM2 provides the method for capturing the events so that one can perform specific action in response to them. It also provides an **Event** and **MouseEvent** objects. These are basically the interfaces which contain information specific to a given event. This information can then be used by your event processing code.

Following table enlists the Event Interface and types of various Events that can be triggered.

Event interface	Various events
Event	blur,change,abort,error,focus,load,reset,resize,select,scroll,unload,submit
MouseEvent	mousedown, mouseup, mouseover, mousemove, mouseout, click

In DOM0 the connection between even and event handler is simple. But in DOM2 the event model is complicated one.

4.15.4 Event Propagation

- Event propagation means travelling of an event. To understand the event propagation consider one scenario -

  Suppose there is an element inside other element. And both these elements have **onchange** event handlers. Then the question occurs here is that which event will be handled first whether the event in the outer element or the event in inner element? Unfortunately there are two opinions about it - the first opinion is the event in the outer element should be handled first and then the event in the inner element should get handled(*event capturing*).

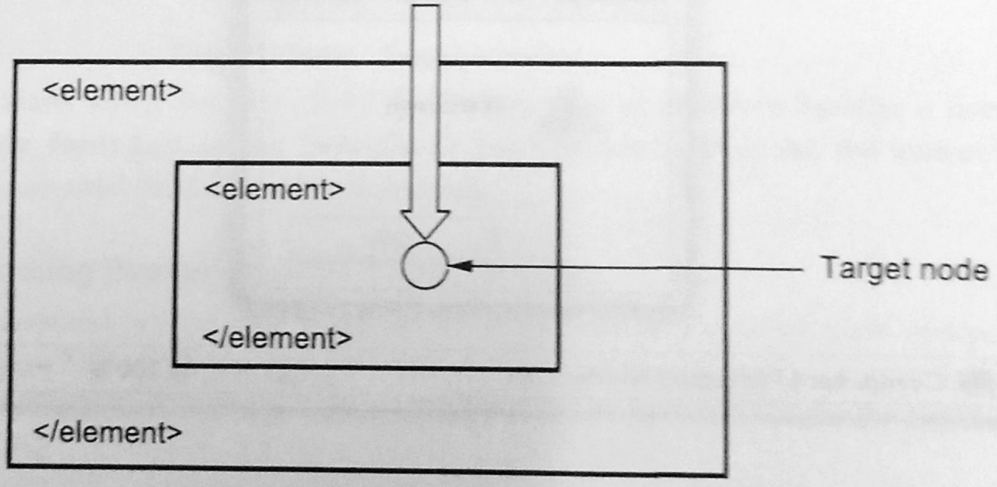


Fig. 4.4 Event capturing

According to other opinion, first directly the event which is present in the inner element should be handled and then the event in the outer element must be handled(*event bubbling*). The DOM0 supports this kind of event handling.

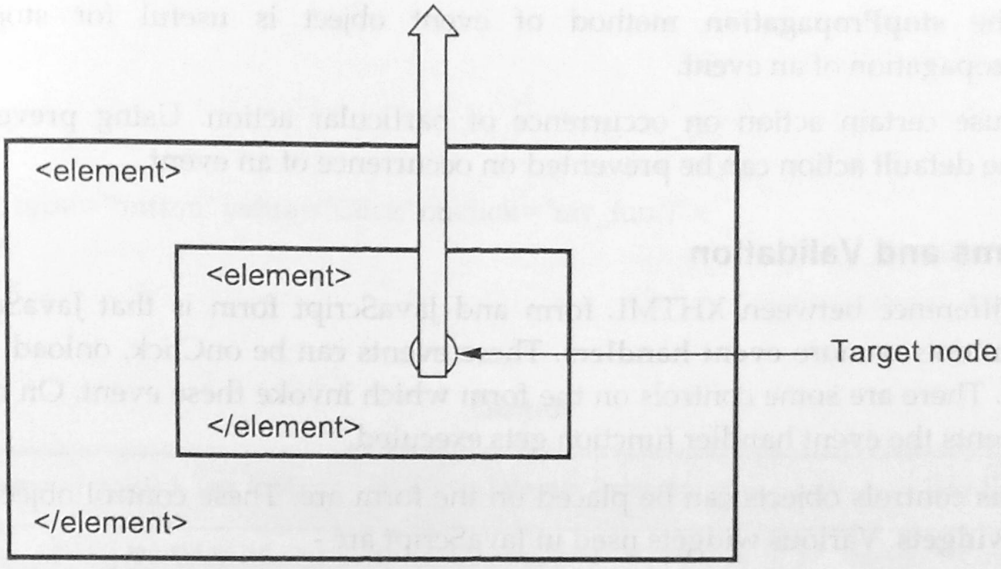


Fig. 4.5 Event bubbling

- Event object is created at a node in the document tree. This node is called the **target node**
- To solve the controversy of event handling whether it is using capturing or using bubbling, the W3C suggested the **DOM2 event model** in which both the approaches are considered.
- According to DOM2 event model the event handling is a three-phase process

Phase 1

This phase is called the **capturing phase** in which the event starts at root node in the document tree and propagates towards the target node. In this direction if any event handler is found (not the target node event handler) then first of all it is checked whether they are enabled or not. If there is any enabled event then it is executed by the capturing phase (may or may not be enabled event). Thus event then reaches to the target node.

Phase 2

In this phase the handlers registered to the target node get executed.

Phase 3

This phase is called the **bubbling phase** in which the event starts at the target node and moves up to the document tree node. In this phase the event bubbles up the document tree to the root node. On this trip if any event handler registered for event gets executed by the bubbling phase (may or may not be enabled event).

- There are some events that can be bubbled or captured. All the mouse events can be bubbled but load and unload events do not get bubbled.

- The **stopPropagation** method of event object is useful for stopping the propagation of an event.

Events cause certain action on occurrence of particular action. Using **preventDefault** method the default action can be prevented on occurrence of an event.

4.16 Forms and Validation

The difference between XHTML form and JavaScript form is that JavaScript form relies upon one or more **event handlers**. These events can be onClick, onload, onSubmit and so on. There are some controls on the form which invoke these event. On occurrence of such events the event handler function gets executed.

Various controls objects can be placed on the form are. These control objects are also called as **widgets**. Various widgets used in JavaScript are -

- Text box
- Password
- Push button
- Radio button
- Check box

Let us discuss some of these widgets while processing the form.

4.16.1 Button

For handling the event using button element we have used the tag attribute **onclick**. The idea is that whenever we click the button some event handler must be called. This event handler can be a user defined function in which certain set of instructions get executed.

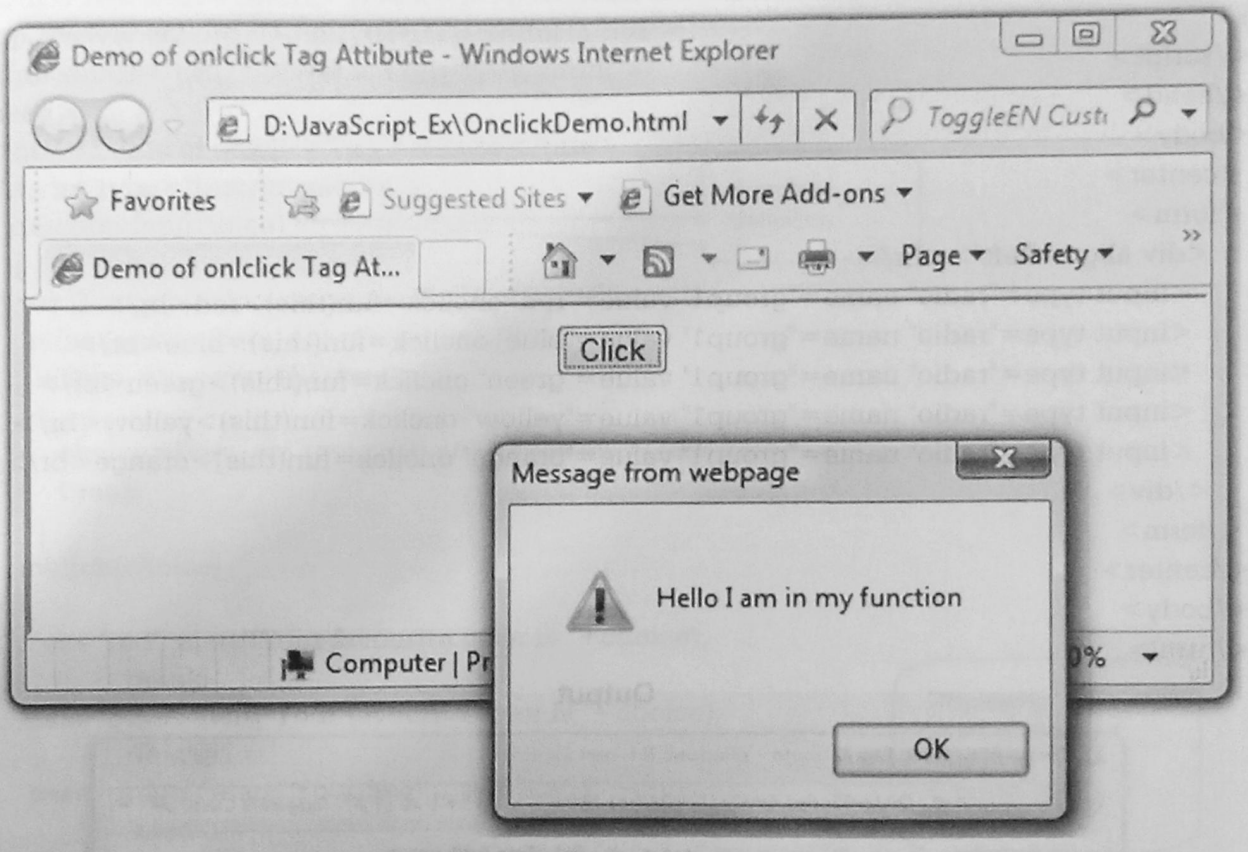
Following is a simple JavaScript in which on the button click we have called a function my_fun(). This is a simple function in which we have displayed some message using alert popup box.

JavaScript[OnclickDemo.html]

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
 <title>Demo of onlclick Tag Attibute</title>
<script type="text/javascript">
 function my_fun()
 {
  alert("Hello I am in my function");
 }
```

```
</script>
</head>
<body>
 <center>
 <form>
   <input type="button" value="Click" onclick="my_fun()">
 </form>
</center>
</body>
</html>
```

Output



In above script we have declared the form using <form> tag. Inside this tag we defined the input type="button" for defining the widget **button**. The **value** denotes the string to be written on this widget. Then **onclick** event occurs which can be then handled using the event handler function **my_fun()**

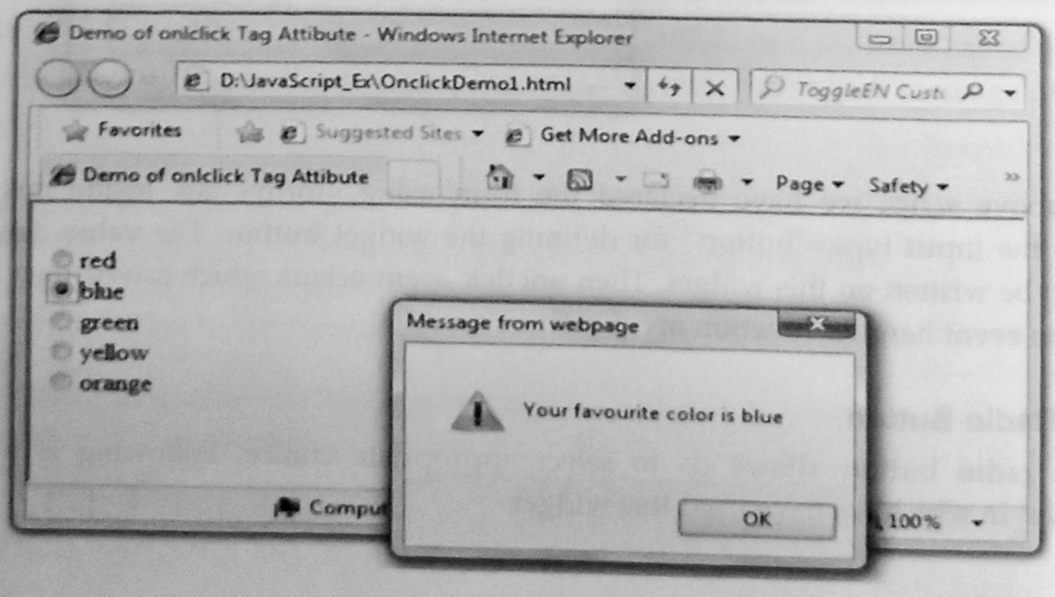
4.16.2 Radio Button

The **radio** button allows us to select appropriate choice. Following is a sample JavaScript in which we have used this widget -

JavaScript[OnclickDemo1.html]

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
 <title>Demo of onlclick Tag Attibute</title>
<script type="text/javascript">
function fun(form_obj)
{
 alert("Your favourite color is "+form_obj.value);
}
</script>
</head>
<body>
 <center>
 <form>
    <div align="left"><br/>
      <input type="radio" name="group1" value="red" onclick=fun(this)>red<br/>
      <input type="radio" name="group1" value="blue" onclick=fun(this)>blue<br/>
      <input type="radio" name="group1" value="green" onclick=fun(this)>green<br/>
      <input type="radio" name="group1" value="yellow" onclick=fun(this)>yellow<br/>
      <input type="radio" name="group1" value="orange" onclick=fun(this)>orange<br/>
    </div>
 </form>
</center>
</body>
</html>
```

Output



In above script the event handler is registered by assigning its call to **onclick** attribute of radio buttons. The specific button that is clicked is identified by sending the parameter to the event handler function **fun**. We have passed the corresponding object as the specific parameter using **this** pointer. In the event handler routine using the value of sent object the particular button click can be identified.

We can modify the above script in order to access the element by using the DOM object. This DOM object can be obtained using the method **getElementById.**

JavaScript[onclickDemo1.html]

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Demo of onlclick Tag Attibute</title>
  <script type="text/javascript">
    function fun(choice)
    {
     var Dom_obj=document.getElementById("form1");
     for(var i=0;i<Dom_obj.group1.length;i++)
      if(Dom_obj.group1[i].checked)
      {
        choice=Dom_obj.group1[i].value;
        break;
      }
     switch(choice)
     {
     case "red": alert("Your favourite color is "+choice);
                 break;
     case "blue": alert("Your favourite color is "+choice);
                 break;
     case "green": alert("Your favourite color is "+choice);
                 break;
     case "yellow": alert("Your favourite color is "+choice);
                 break;
     case "orange": alert("Your favourite color is "+choice);
                 break;
     }
    }
  </script>
</head>
<body>
 <center>
 <form id="form1">
    <div align="left"><br/>
      <input type="radio" name="group1" value="red" />red<br/>
```

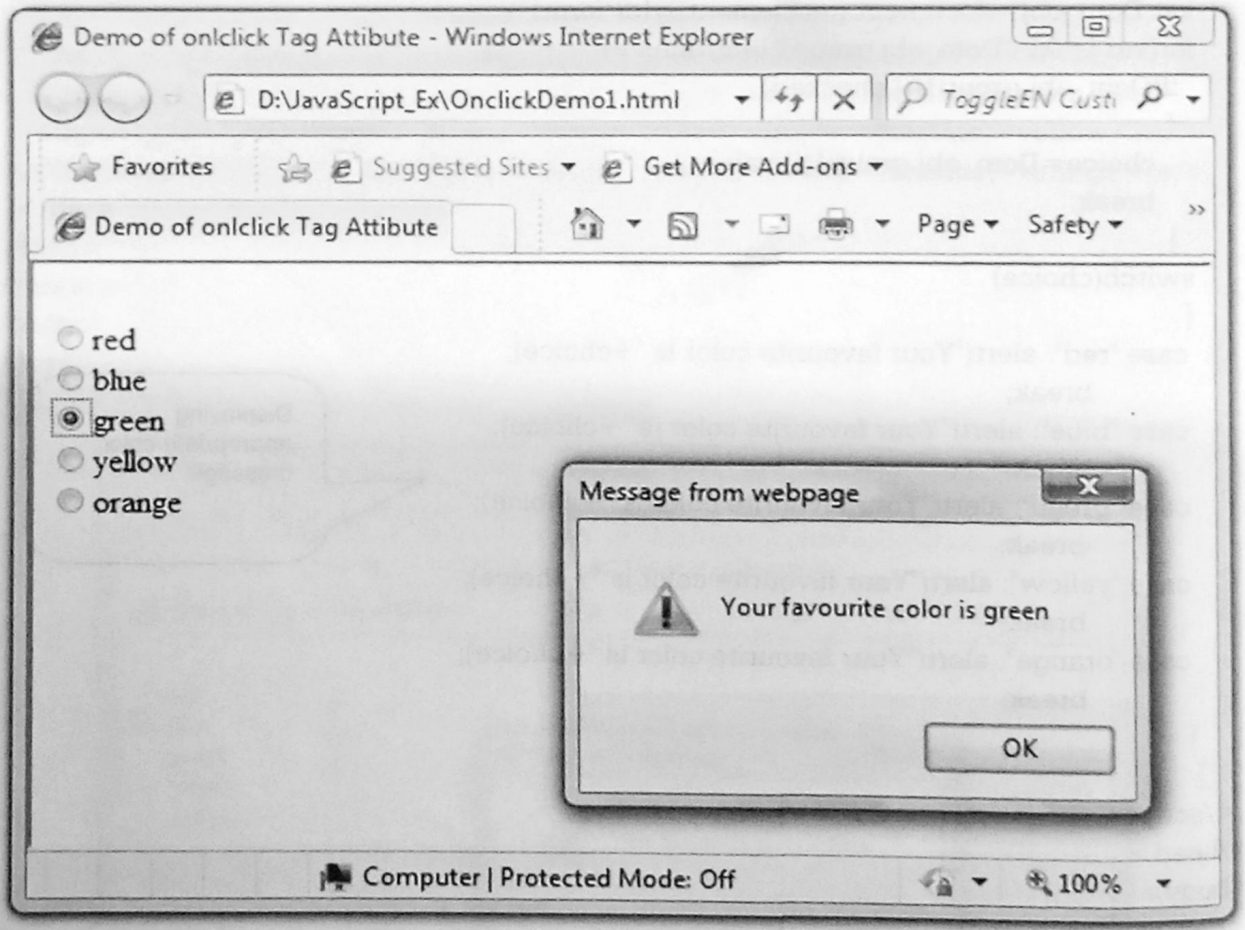
Function definition

Displaying appropriate color message

```
    <input type="radio" name="group1" value="blue" />blue<br/>
    <input type="radio" name="group1" value="green" />green<br/>
    <input type="radio" name="group1" value="yellow" />yellow<br/>
    <input type="radio" name="group1" value="orange" />orange
  </div>
</form>
<script type="text/javascript">
  var Dom_obj=document.getElementById("form1");
  Dom_obj.elements[0].onclick=fun;
  Dom_obj.elements[1].onclick=fun;
  Dom_obj.elements[2].onclick=fun;
  Dom_obj.elements[3].onclick=fun;
  Dom_obj.elements[4].onclick=fun;
</script>
</center>
</body>
</html>
```

Call to the function **onclick**. That means registering the event

Output



Just execute the above script, select different radio buttons and accordingly you will get the appropriate message using the alert box.

4.16.3 Text Box and Password Element

There are various events such as focus, change, blur and select which can be used for the form elements such as text box and password. We will discuss these events and their handling using some illustrative examples

4.16.3.1 The focus event and blur() Method

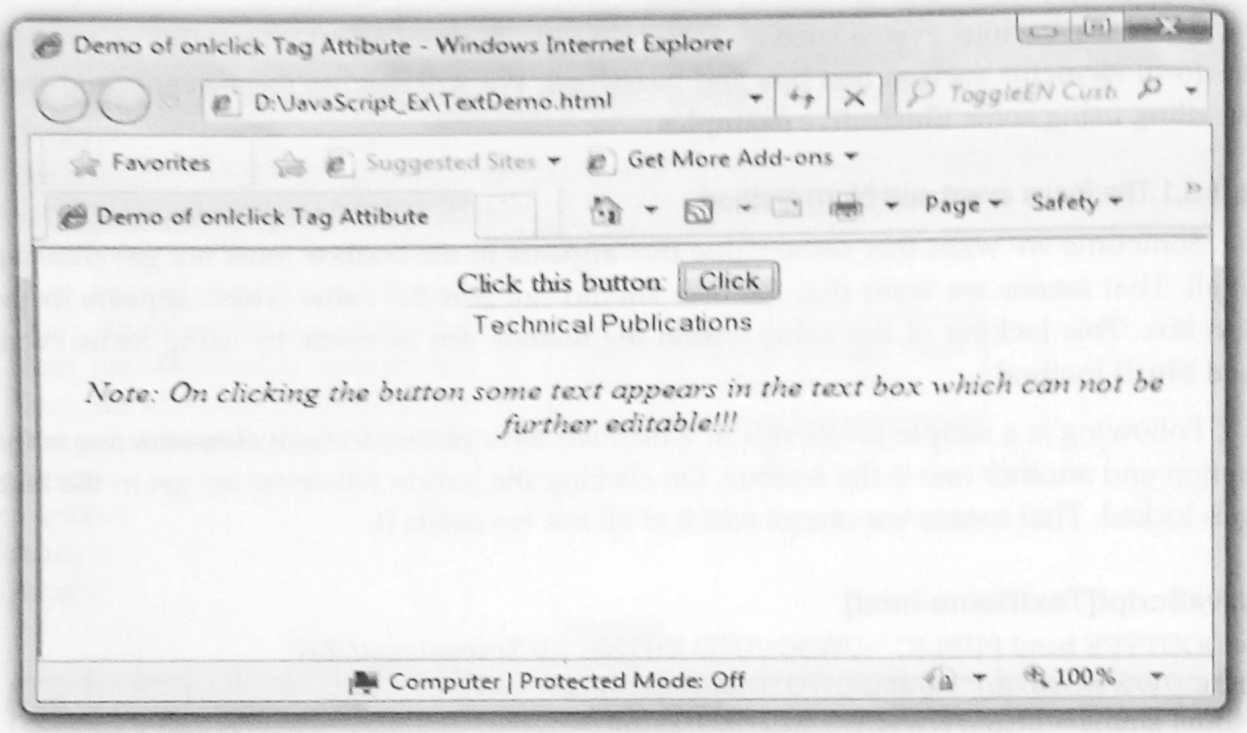
Sometime we want that some value that appears in the textbox must not get changed at all. That means we want that the user should not edit the value which appears in the text box. This locking of the value within the textbox can be made by using focus event and **blur()** method.

Following is a simple JavaScript in which we have placed to form elements one is the button and another one is the textbox. On clicking the button whatever we get in the text, gets locked. That means we cannot edit it at all nor we delete it.

JavaScript[TextDemo.html]

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
 <title>Demo of onlclick Tag Attibute</title>
<script type="text/javascript">
 function my_fun()
 {
  document.getElementById("mytext").value="Technical Publications";
  }
</script>
</head>
<body>
 <center>
 <form id="form1">
 <label>Click this button:
  <input type="button" value="Click" onclick="my_fun();"/><br/>
</label>
 <input type="text" value="" id="mytext" onfocus="this.blur();"/>
 </form>
<p><em>
 Note: On clicking the button some text appears in the text box
 which can not be further editable!!!
</em></p>
</center>
</body>
</html>
```

Output



4.16.3.2 Password Verification

We have used password verification process that comes along the web document. In this process user has to enter the password correctly for two times. If both the passwords are matching then the password is verified. Normally this facility is given when user creates his web account.

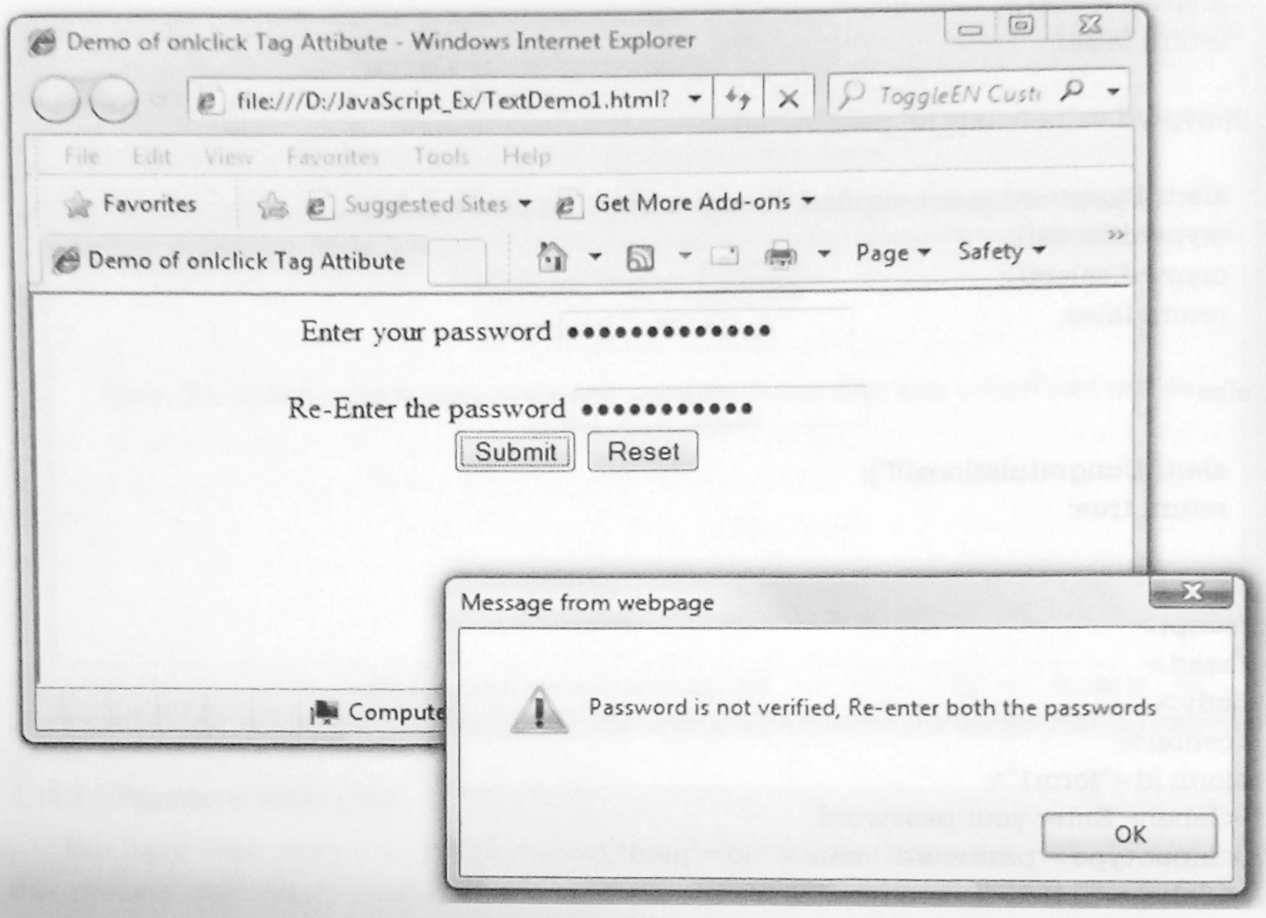
In the following JavaScript we have used two textboxes which are of **password** type. That means whatever we type in these boxes appear in the form of dots. We will compare the entries in the two text boxes; if those are not same then the alert message will be displayed.

JavaScript[TextDemo.html]

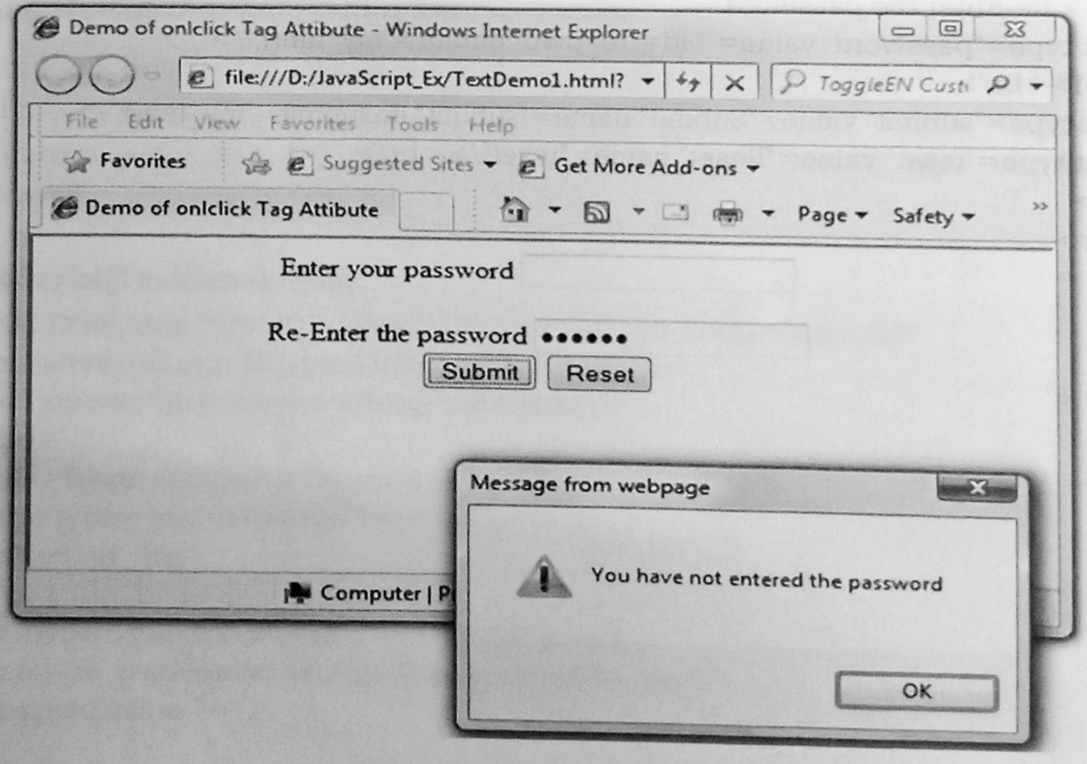
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
 <title>Demo of onlclick Tag Attibute</title>
<script type="text/javascript">
 function my_fun()
 {
  var mypwd=document.getElementById("pwd");
  var my_re_pwd=document.getElementById("re_pwd");
  if(mypwd.value=="")
  {
```

```
    alert("You have not entered the password");
    mypwd.focus();
    return false;
   }
   if(mypwd.value!=my_re_pwd.value)
   {
    alert("Password is not verified, Re-enter both the passwords");
    mypwd.focus();
    mypwd.select();
    return false;
   }
   else
   {
    alert("Congratulations!!!");
    return true;
   }
  }
</script>
</head>
<body>
 <center>
 <form id="form1">
  <label> Enter your password
  <input type="password" value="" id="pwd" />
  </label>
  <br/><br/>
  <label> Re-Enter the password
  <input type="password" value="" id="re_pwd" onblur="my_fun();"/>
  </label><br/>
  <input type="submit" value="Submit" name="submit" onsubmit="my_fun();"/>
   <input type="reset" value="Reset" name="reset"/><br/>
 </form>
</center>
</body>
</html>
```

Output(Run1)



Output(Run2)



Output(Run3)

Review Questions

1. *Explain the JavaScript execution environment.*
2. *What is document object model?*
3. *What are the ways by which we can access the elements in JavaScript?*
4. *What is event and event handling?*
5. *Explain the terms-events, attributes and tags?*
6. *Write a JavaScript illustrating the event handling from body elements.*
7. *Write a JavaScript illustrating the event handling using button elements.*
8. *Exaplain the focus and blur methods.*
9. *Write a JavaScript for verifying the password entered by the user.*
10. *"JavaScript is used in validation techniques" -Comment on it.*
11. *Explain DOM2 Event model.*
12. *Expalin the process of event propagation.*
13. *What are the drawbacks of DOM0 that are overcome in DOM2?*
14. *What is navigator object?*
15. *Explain the technique of DOM tree traversal and modification.*