Chapter 7

Adding Visual Effects to Web Pages

Today, almost every Web developer wants to add visual effects to websites. However, adding complex visual effects to Web pages requires writing many lines of JavaScript code, which is a time-consuming activity. To overcome this issue, jQuery was introduced. It is the JavaScript library supported by HTML that wraps several lines of code into methods that can be called by using a single statement.

This chapter introduces you to jQuery. Further, it discusses jQuery selectors, events, and effects. Also, it discusses how to add the image rollover effect to an image and how to create an image gallery, thereby, making the website more attractive and enhancing the browsing experience of the users.

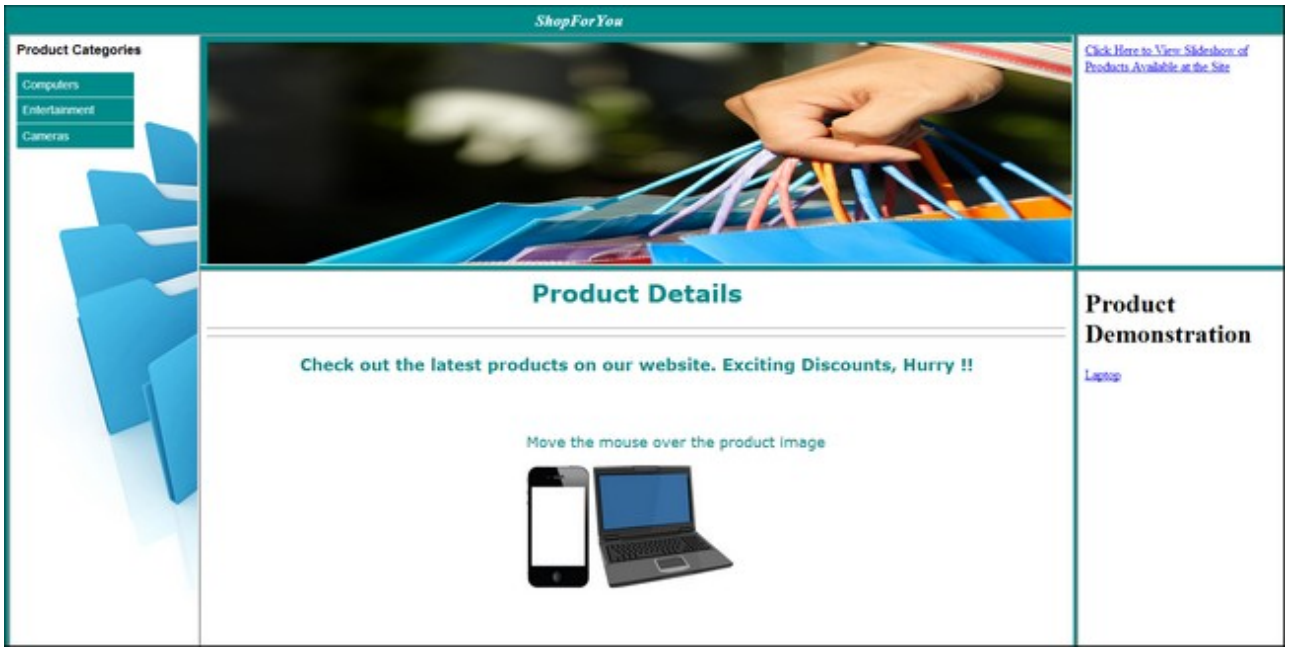
Objectives

In this chapter, you will learn to:

- Explore jQuery
- Add visual effects using jQuery
- Implement image rollover
- Create image gallery

Exploring jQuery

Consider a scenario of the online shopping website, ShopForYou.com. The website should be designed in such a way that it displays the product categories and items in the sliding and collapsible menu and submenu format, as shown in the following figure.



The Sample Web Page

For this, the Web designer has been asked to complete the task quickly and keep the code simple. For this situation, the designer can use the prewritten JavaScript library known as *jQuery*.

jQuery is an open source and cross-browser library of JavaScript codes that was created to make the JavaScript programming simpler. jQuery can be used by all levels of programmers to enhance the look and feel of the Web page. jQuery supports events and can be bound to the HTML elements on the Web page. jQuery bridges the gap between extensive coding and Web designing as it is easy to understand and provides many predefined special effects. With jQuery, a Web designer can add a plethora of impressive visual effects and also extend interactivity on a Web page.

Manipulating HTML Elements by Using jQuery

You can make use of jQuery in a Web page to manipulate HTML elements by downloading the light-weight jQuery JavaScript library and saving it to your system. Once the jQuery JavaScript library is downloaded, it can be referred to in a Web page by using the `<SCRIPT>` tag in the head section of the Web document.

The following syntax is used to specify the jQuery library:

```
<SCRIPT type="text/javascript" src="<path>/jquery-1.8.3.js"> </SCRIPT>
```

In the preceding syntax:

- The `SCRIPT` tag instructs the browser that the HTML document uses a script.
- The `type` attribute specifies the type of scripting used.
- The `src` attribute specifies the name of the jQuery library used. If the jQuery library is stored at the same location as the HTML document, only the name of the library can be given. However, if the jQuery library and the HTML document are stored at different locations, you have to specify the complete path of the jQuery library.

The execution of jQuery code should occur only after the Web page has been fully loaded. To ensure that a Web page is fully loaded before the jQuery code is executed on it, jQuery provides the `document.ready()` function. This function contains the jQuery code to be executed on HTML elements after the Web page has been fully loaded in the browser.

The following syntax is used to specify the `document.ready()` function:

```
<SCRIPT type="text/javascript">
$(document).ready(function(){
// jquery code...
});
</SCRIPT>
```

In the preceding syntax, the dollar sign (`$`) represents start of the jQuery code block and `jQuery code` is the code to be implemented for the rendering of the HTML elements in the browser window. The `document.ready()` function is useful for preventing failure of actions that should be performed on an HTML element. For example, if the jQuery code for hiding an image executes before the document is ready, it will lead to failure. Therefore, to allow execution of the jQuery code only when the document is ready to be displayed in the browser window, all the jQuery methods and code should be declared and defined inside the `document.ready()` function.

The keyword, `jQuery`, can be used in place of the $ sign while writing jQuery code. For example, the preceding syntax can also be written as:

```
jQuery(document).ready(function(){
//jQuery code...
});
```

As jQuery is used to manipulate HTML elements, the elements should be selected first in order to implement the jQuery code on them.

The following jQuery syntax is used to select the HTML elements and perform the required action on them:

```
$(selector).action();
```

In the preceding syntax, `selector` is the element to be manipulated and `action` represents action to be taken on the selected element. jQuery provides many built-in actions that can be applied to the HTML elements, such as hide, fadeout, show, and slideup. Consider the following code to hide all the `<H1>` elements in an HTML document:

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<SCRIPT type="text/javascript" src="jquery-1.8.3.js">
</SCRIPT>
<SCRIPT>
$(document).ready(function(){
$("h1").hide();
});
</SCRIPT>
</HEAD>
<BODY>
...
</BODY>
</HTML>
```

In the preceding code snippet, all the `<H1>` elements will be hidden in the Web page.

jQuery code is written in the <HEAD> section of the document.

Using jQuery Selectors

jQuery uses selectors to select and manipulate an HTML element or a group of elements. You can select an HTML element by:

- Using element name
- Using attribute name
- Using CSS selectors

Using Element Name

jQuery can be used to select an HTML element or a group of HTML elements by using its name. Using jQuery, HTML elements can be selected by referring to their name, ID, or class to which they belong.

Consider the following code snippet to select all the `<H1>` elements in a document:

```
$("h1")
```

In the preceding code snippet, the `<H1>` elements are referred to by the name, `h1`. However, HTML elements

can also be selected by using class or ID. To select an element with a specific ID by using jQuery, you can use the following syntax:

```
$("#Element_ID")
```

In the preceding syntax, `Element_ID` represents ID of the element to be selected. For example, to select an element having the ID, `header`, you can use the following code snippet:

```
$("#header")
```

To select elements with a specific class, you can use the following syntax:

```
$(".Element_class")
```

In the preceding syntax, `.Element_class` represents the class of the elements to be selected. For example, to select elements having the value of the class attribute as `header`, you can use the following code snippet:

```
$(".header")
```

For example, you have created the class selector named `exClass` and the ID selector named `exId` on the `<H1>` tag, as shown below:

```
<H1 class="exClass">
<H1 ID="exId">
```

Now, you want to manipulate the `<H1>` tag with the specified class and ID. You need to use the following code:

```
$("h1.exClass")
$("h1#exId")
```

Here, by using the `$("h1.exClass")` statement, only the `<H1>` header elements specified with the `exClass` value for the class attribute will be selected and by using the `$("h1#exId")` statement, only the `<H1>` header elements that are specified with the `exId` value for the ID attribute will be selected.

jQuery also supports nesting of elements. This allows applying an action on the elements that are nested within other elements. For example, a list item is nested inside a list. Consider the example of the ShopForYou online shopping website where the Web designer wants to implement a menu and submenu. The submenu items must be linked to display their details in a new document.

A menu comprises a list and list items that are denoted by the `<UL>` and `<LI>` tags, respectively. Consider the following code snippet to build a button, menu, and submenu, and to attach links to the submenu items:

```
<BODY>
<P> Product Menu</P>
<H4>Product Categories</H4>
<UL ID="menu">
<LI><A>Laptops</A>
<UL>
<LI><A href="">L-150</A></LI>
<LI><A href="">L-160</A></LI>
<LI><a href="">L-180</A></LI>
<LI><A href="">L-260</A></LI>
<LI><A href="">L-678</A></LI>
</UL>
</LI>
<LI><A>Phones</A>
```

```
<UL>
<LI><A href="">P-1150</A></LI>
<LI><A href="">P-1260</A></LI>
<LI><A href="">P-1180</A></LI>
<LI><A href="">P-2260</A></LI>
<LI><A href="">P-6708</A></LI>
</UL>
</LI>
<LI><A>Cameras</A>
<UL>
<LI><A href="">C-50</A></LI>
<LI><A href="">C-60</A></LI>
<LI><A href="">C-80</A></LI>
<LI><A href="">C-60</A></LI>
<LI><A href="">C-08</A></LI>
</UL>
</LI>
<LI><A>Music Players</A>
<UL>
<LI><A href="">MP-1150</A></LI>
<LI><A href="">MP-1260</A></LI>
<LI><A href="">MP-1180</A></LI>
<LI><A href="">MP-2260</A></LI>
<LI><A href="">MP-6708</A></LI>
</UL>
</LI>
</UL>
<BUTTON>Hide List</BUTTON>
</BODY>
```

Consider the following code snippet to hide all the nested `<UL>` items on the button click of a user, for which `menu` is specified as the ID attribute value:

```
<HEAD>
<SCRIPT src="jquery-1.8.3.js">
</SCRIPT>
<SCRIPT>
$(document).ready(function(){
$("button").click(function(){
$('#menu ul').hide();
});
});
</SCRIPT>
</HEAD>
```

In the preceding code snippet, `#menu` is used to select the HTML element for which the value of the ID attribute is `menu`. `#menu ul` selects all the `<UL>` items nested inside the HTML element for which the `ID` is specified as

menu. When the user clicks on the button, all the nested list items will get hidden, and the menu with items at the first level: Laptops, Phones, Cameras, and Music Players will be visible.

Using Attribute Name

The HTML elements can also be selected by using the name of an attribute. The following syntax is used to select elements by using an attribute name:

```
$("[attribute-name]")
```

In the preceding syntax, attribute-name specifies the name of the attribute.

Consider the following code snippet:

```
$("[src]")
```

In the preceding code snippet, all the elements with the src attribute are selected.

Consider the following code snippet:

```
$("[src='movie.jpg']")
```

In the preceding code snippet, all the elements that have the src attribute as movie.jpg are selected.

Using CSS Selectors

jQuery provides CSS selectors to modify the CSS properties of an HTML element or document.

The following syntax is used to apply CSS selectors to modify the document or element properties:

```
selector.css( PropertyName, PropertyValue );
```

Consider the following code snippet to change the color of the text of the <DIV> element to green:

```
$("div").css("color","green");
```

In the preceding code snippet, all the div elements are selected in the document, and the css color property value is modified to green.

With CSS selectors, you can set multiple properties on an HTML element. The following syntax is used to set multiple CSS properties:

```
css({"propertyname":"value","propertyname":"value",...});
```

Consider the following code snippet to change the color and font size of the text of the <DIV> element:

```
$("div").css("color":"green", "font-size":"200%");
```

In the preceding code snippet, all the div elements are selected in the document, and the css color property value is modified to green and font-size is set to 200%.

jQuery HTML

The jQuery library provides predefined functions to perform modifications on the content of HTML elements.

The following table lists the most commonly-used functions to modify the content of HTML elements.

Function	***Description***	***Syntax***
html()	*Changes the*	*$(selector).html(content)*

	content of HTML elements, such as text in the <P> and <DIV> tags.	
`append()`	*Enables a user to append content after the inside content in the selected element.*	`$(selector).append(content)`
`prepend()`	*Enables a user to append content before the inner content of selected elements.*	`$(selector).prepend(content)`
`remove()`	*Removes the selected element including all the text and nested elements inside it.*	`$(selector).remove()`
`empty()`	*Removes all the text and nested elements of selected elements. However, this method does not remove the element.*	`$(selector).empty()`
`after()`	*Inserts the content after the selected elements.*	`$(selector).after(content)`
`before()`	*Inserts the content before the selected elements.*	`$(selector).before(content)`
`text()`	*Sets or returns the content of selected*	`$(selector).text(content)`

	elements.	
`val()`	*Sets or returns the values of the attributes of selected elements.*	`$(selector).val(value)`
`attr()`	*Sets or returns the values and attributes of selected elements.*	`$(selector).attr(attribute,value)`

The jQuery HTML Functions

Consider the following code for modifying the inner content of HTML elements:

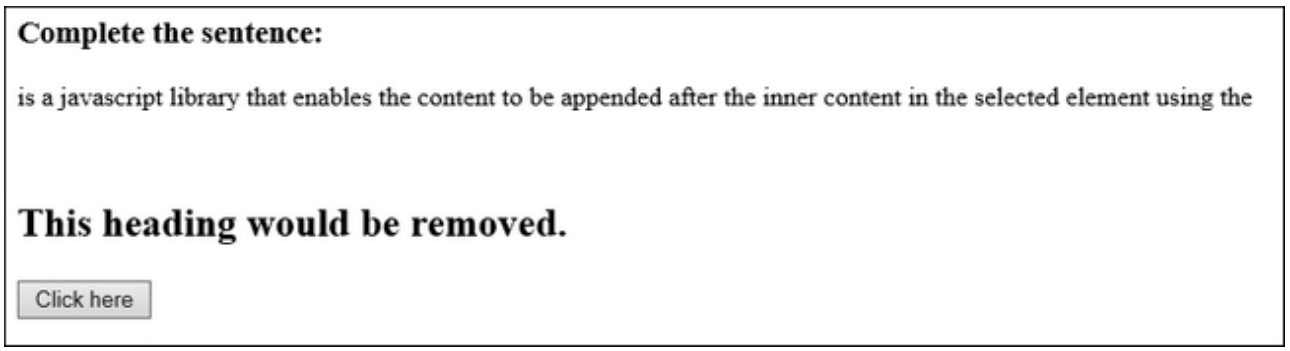
```
<!DOCTYPE HTML><HTML>
<HEAD>
<SCRIPT type="text/javascript" src="jquery-1.8.3.js"></SCRIPT>
<SCRIPT type="text/javascript">
$(document).ready(function(){
$("button").click(function(){
  $("#para").prepend(" <B>jQuery </B>");
  $("#para").append(" <B>append() function</B>.");
  $("#para").after('<H3> This is the newly inserted heading.</H3>');
  $("#newhead").remove();
  });
});
</SCRIPT>
</HEAD>
<BODY>
<DIV align="center">
</DIV>
<H3>Complete the sentence:</H3>
<P ID="para">is a javascript library that enables the
content to be appended after the inner content in the
selected element using the</P><BR>
<H2 ID="newhead">This heading would be removed.</H2>
<BUTTON ID="b">Click here</BUTTON>
</BODY>
</HTML>
```

In the preceding code, when a user clicks the **Click here** button, the content, **jQuery,** is appended before the `<P>` tag having the ID, `para`, by using the `prepend()` function. Similarly, the content, `append()` function, is appended after the same `<P>` tag by using the `append()` function.

In addition, a new `<H3>` tag is inserted after the `<P>` tag having the ID, `para`, by using the `after()` function.

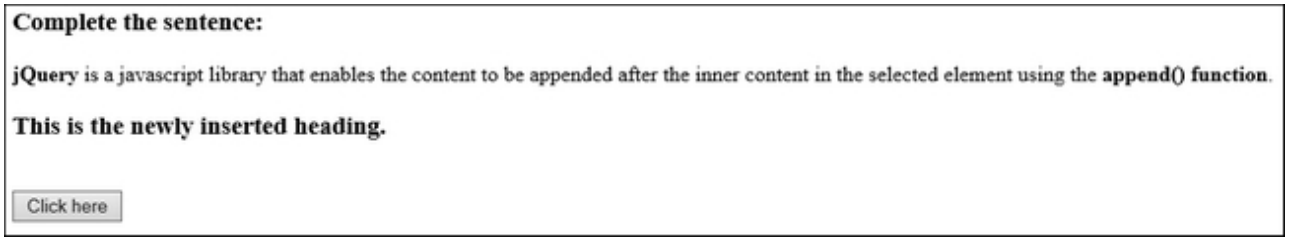
Finally, the heading having the ID, newhead, is removed by using the `remove()` function.

Before the **Click here** button is clicked, the Web page appears, as shown in the following figure.



The Web Page Before the Button is Clicked

After the **Click here** button is clicked, the Web page appears, as shown in the following figure.



The Web Page After the Button is Clicked

The jQuery events are covered in the following topic.

Handling jQuery Events

In jQuery, events are handled by using functions or predefined event methods. An event method is used to detect an event and trigger a function when that event occurs. You can manipulate an element with a function or an event method.

The following table lists some of the event methods provided by jQuery.

Event Method	***Description***
`$(document).load(function)`	*Used to execute a function after the document has finished loading.*
`$(selector).click(function)`	*Used to execute a function on the click event of the selected element.*
`$(selector).dblclick(function)`	*Used to execute*

	a function on the double-click event of the selected element.
`$(selector).mouseenter(function)`	*Used to execute a function when the mouse pointer is moved over the selected element.*
`$(selector).mouseleave(function)`	*Used to execute a function when the mouse pointer is moved out of the selected element.*
`$(selector).keydown(function)`	*Used to execute a function when a key is pressed on the selected element.*
`$(selector).submit(function)`	*Used to execute a function when a form is submitted.*
`$(selector).focus(function)`	*Used to execute a function when the selected element gets focus.*
`$(selector).blur(function)`	*Used to execute a function when the selected element loses focus.*
`$(selector).resize(function)`	*Used to execute a function when the browser window changes size.*

$(selector).unload(function)	*Used to execute a function when the user navigates away from the page.*
$(selector).mousedown(function)	*Used to execute a function when the left mouse button is pressed down on the selected element.*
$(selector).mouseup(function)	*Used to execute a function when the left mouse button is released from the selected element.*

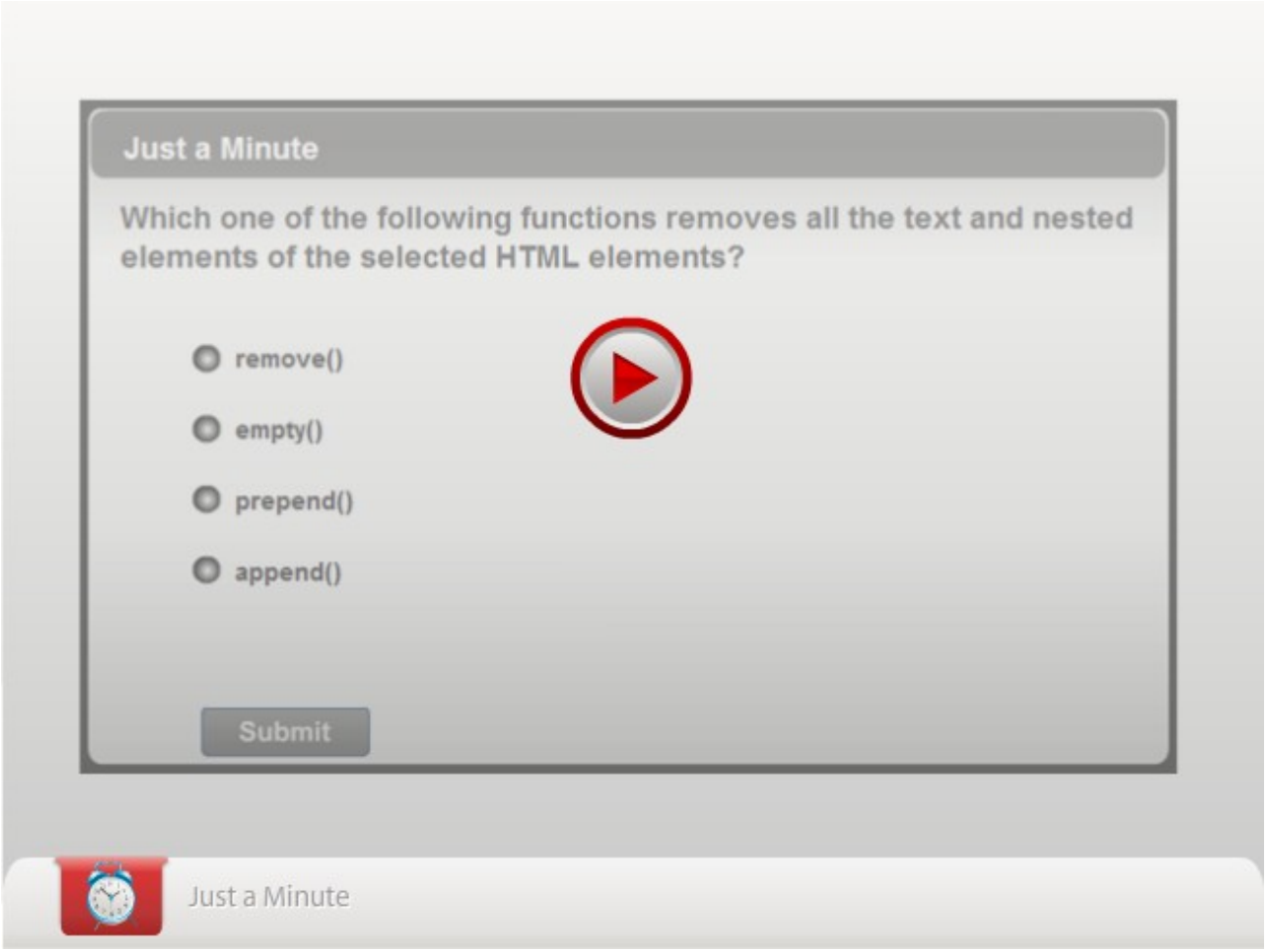
The Event Methods in jQuery

Consider the following code to handle jQuery events:

```
<!DOCTYPE HTML><HTML>
<HEAD>
<SCRIPT type="text/javascript" src="jquery-1.8.3.js"></SCRIPT>
<SCRIPT type="text/javascript">
$(document).ready(function(){
$("#hide_header").click(function(){ /* selects the hide_header button and
binds it to the click function */
$("h1").hide(); //hides the <h1> header
});
});
</SCRIPT>
</HEAD>
<BODY>
<H1>I am the header</H1>
<HR>
<INPUT type="button" ID="hide_header" value='Hide the header'/>
</BODY>
</HTML>
```

In the preceding code, jQuery recognizes a click event when the user clicks the **Hide the header** button. After the event is recognized by jQuery, the function associated with the button is executed and the `<H1>` element is hidden.

There are some predefined functions that can be attached to event methods in jQuery to enrich the visual appearance of a Web document. These functions are referred to as jQuery effects.



Adding Visual Effects Using jQuery

With the help of jQuery, amazing visual effects can be applied to the elements of a Web document. These visual effects help to enrich the browsing experience of a user. Some of the predefined jQuery effects that can be used to add visual appeal to a Web page are:

- Hide
- Show
- Toggle
- Slide
- Fade
- Animate

Implementing Hide Effect

The `hide()` function is used to make an element disappear when an event, such as click or double-click, occurs.

The following syntax is used to hide the selected elements:

```
$(selector).hide(speed)
```

In the preceding syntax, `speed` is an optional parameter that denotes speed with the values, slow, normal, and fast, or the duration in milliseconds at which an element disappears. Consider the following code to implement the hide effect:

```
<!DOCTYPE HTML><HTML>
```

```
<HEAD>
<SCRIPT type="text/javascript" src="jquery-1.8.3.js"></SCRIPT>
<SCRIPT type="text/javascript">
$(document).ready(function(){
$("button").click(function(){
$("h1").hide("slow");
});
});
</SCRIPT>
</HEAD>
<BODY>
<H1>I am the header</H1>
<BUTTON>Hide the header</BUTTON>
</BODY>
</HTML>
```

In the preceding code, the text, **I am the header**, is displayed above the **Hide the header** button. When the **Hide the header** button is clicked, the text, **I am the header,** slowly disappears from the browser window.

Implementing Show Effect

The `show()` function is used to make a hidden element visible when an event occurs. The following syntax is used to show the selected elements, if they are already hidden:

```
$(selector).show(speed)
```

In the preceding syntax, `speed` is an optional parameter. This parameter is used to specify the speed with the values, slow, normal, and fast, or the duration in milliseconds at which an element reappears.

Consider the following code snippet to implement the show effect:

```
<!DOCTYPE HTML><HTML><HEAD>
<SCRIPT type="text/javascript" src="jquery-1.8.3.js"></SCRIPT>
<SCRIPT type="text/javascript">
$(document).ready(function(){
$(".view").click(function(){
$("h3").show(2000);
});
$(".conceal").click(function(){
$("h3").hide();
});
});
</SCRIPT>
</HEAD>
<BODY>
<H3> This text can be hidden and shown. </H3> <BR>
<BUTTON class="view">Click to view</BUTTON>
<BUTTON class="conceal">Click to hide</BUTTON>
```

```
</BODY>
</HTML>
```

In the preceding code snippet, the `<H3>` tag gets hidden in the browser window when the **Click to hide** button is clicked. When the **Click to view** button is clicked, the `<H3>` tag is shown again in 2000 milliseconds (or 2 seconds).

Implementing Toggle Effect

The `toggle ()` function can be used to switch between the show and hide effects of an element. This event can be used to hide or show the element, alternatively, when an event occurs.

The following syntax is used to show or hide the selected elements:

```
$(selector).toggle(speed)
```

In the preceding syntax, `speed` is an optional parameter. It is used to specify the speed with the values, slow, normal, and fast, or the duration in milliseconds at which an element disappears and reappears.

Consider the following code snippet to implement the toggle effect:

```
<!DOCTYPE HTML><HTML>
<HEAD>
<SCRIPT type="text/javascript" src="jquery-1.8.3.js"></SCRIPT>
<SCRIPT type="text/javascript">
$(document).ready(function(){
$(".view").click(function(){
$("h3").toggle('fast');
});
});
</SCRIPT>
</HEAD>
<BODY>
<H3> Show and Hide me Quickly</H3>
<HR>
<BUTTON class="view">Click here</BUTTON>
</BODY>
</HTML>
```

In the preceding code snippet, the `toggle()` function hides and shows the `<H3>` tag, alternately.

Implementing Slide Effect

The slide effect can be used to produce a sliding effect on the selected elements. There are three slide functions, `slideDown()`, `slideUp( )`, and `slideToggle()`, which can be applied on the selected elements. These functions apply a gradual modification on the height of the selected elements to create the slide animation effect on a Web page.

The following table lists the slide effect functions.

Function	***Description***	***Syntax***

slideDown()	*Produces the effect of sliding selected elements in the downward direction.*	*$(selector).slideDown(speed)*
slideUp()	*Produces the effect of sliding selected elements in the upward direction.*	*$(selector).slideUp(speed)*
slideToggle()	*Performs the slideup and slidedown functions, respectively, on alternate clicks.*	*$(selector).slideToggle(speed)*

The Slide Effect Functions

Consider the following code to implement the slide effect:

```
<!DOCTYPE HTML><HTML>
<HEAD>
<SCRIPT type="text/javascript" src="jquery-1.8.3.js"></SCRIPT>
<SCRIPT type="text/javascript">
$(document).ready(function(){
$(".flipbut").click(function(){
  $(".thought").slideToggle();
  });
});
</SCRIPT>
<STYLE type="text/css">
div.thought,.flipbut{ /*stylizing the div element with the class
            attribute specified as "thought",
            and the button with the class attribute
            specified as "flipbut" */}
margin:0px;
padding:5px;
text-align:center;
background:beige;
```

```
border:solid 1px;
}
div.thought
{
height:120px;
display:none;
}
</STYLE>
</HEAD>
<BODY>
<BR><BR>
<BUTTON style="background-color:beige" class="flipbut" text-
align="center">Show/Hide the thought of the day</BUTTON>
<DIV class="thought">
<P>The thought of the day is:</P>
<P> Where there is a will, there is a way!</P>
</DIV>
</BODY>
</HTML>
```

In the preceding code, when the `Show/Hide the thought of the day` button is clicked, toggle functionality is implemented between slide up and slide down on the `<DIV>` element along with the `thought` class.

For the online shopping website, ShopForYou.com, the menu and submenu must be sliding. When the user clicks a product category, the detailed products for that category should be displayed by using the slide toggle effect.

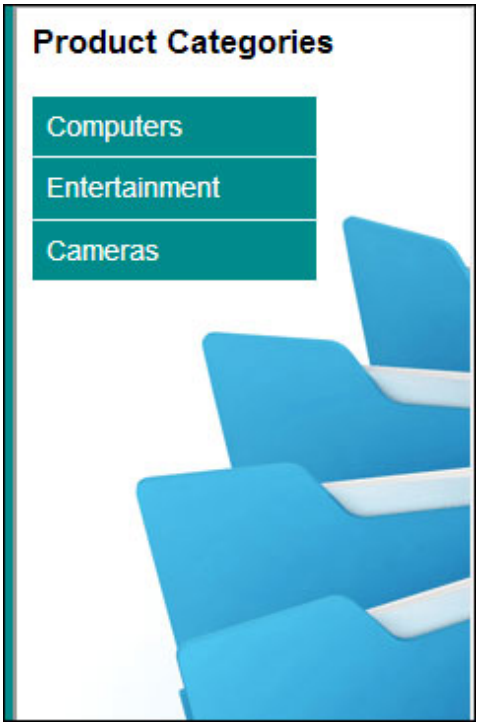
To implement this functionality in the ShopForYou.com website, you need to write the following code in a file and save the file as **menu.html**:

```
<!DOCTYPE HTML><HTML>
<HEAD>
<SCRIPT type="text/javascript" src="jquery-1.8.3.js"></SCRIPT>
<SCRIPT type="text/javascript">
$(document).ready(function(){
$('#menu ul').hide();
  $('#menu li a').mouseenter(function() {
    $(this).next().slideToggle('slow');
    }
   );
});
</SCRIPT>
<STYLE type="text/css">
body{
background-image:url('background.jpg');
}
li a
{
```

```
display:inline;
}
body {
font-family: Arial, sans-serif;
font-size: 0.9em;
background-color:silver;
}
p
{
  line-height: 1.5em;
}
#menu, ul#menu ul { /* stylizing the menu and the list nested in the menu */
  list-style-type:none;
  margin: 0;
  padding: 0;
  width: 10em;
}
ul#menu a { /* stylizing the <a> elements nested in the menu */
  display: inline;
  text-decoration: none;
}
ul#menu li { /* stylizing the list items nested in the menu */
  margin-top: 1px;
}
ul#menu li a { /* stylizing the <a> elements nested in list items of the
menu */
  background: darkcyan;
  color: #fff;
  padding: 0.5em;
}
ul#menu li a:hover { /* stylizing the <a> elements nested in list items of
the menu when mouse is placed or moved over them */
  background: black;
}
ul#menu li ul li a {
  background: grey;
  color: white;
  padding-left: 20px;
}
ul#menu li ul li a:hover {
  background: #aaa;
  color: white;
  border-left: 5px grey solid;
  padding-left: 15px;
}
```

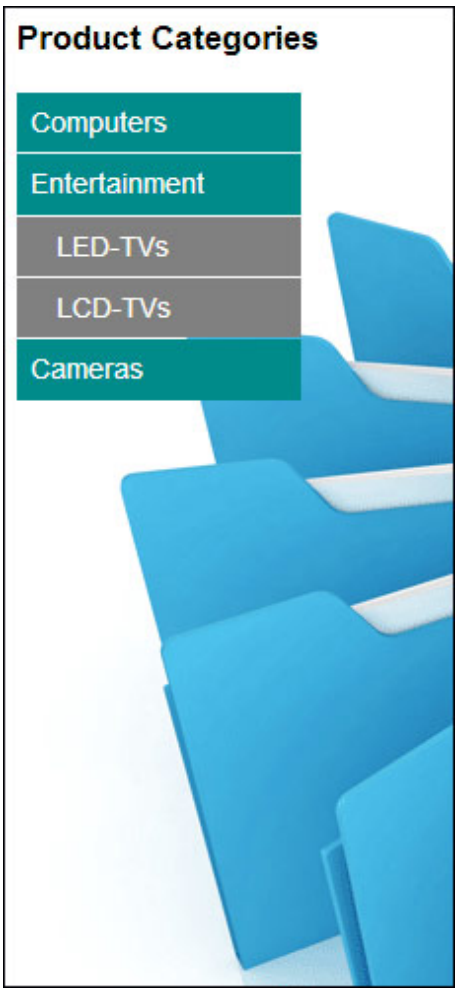
```
.st
{
display:inline;
}
</STYLE>
</HEAD>
<BODY>
<H3>Product Categories</H3>
<UL ID="menu">
<LI class=".st"><A>Computers</A>
<UL>
   <LI><A href="">e-Book Readers</A></LI>
   <LI><A href="">Tablets and i-Pads</A></LI>
<LI><A href="">Laptops</A></LI>
</UL>
</LI>
<LI class=".st"><A>Entertainment</A>
<UL>
   <LI><A href="">LED-TVs</A></LI>
   <LI><A href="">LCD-TVs</A></LI>
</UL>
</LI>
<LI class=".st"><A>Cameras</A>
<UL>
   <LI><A href="">Digital Cameras</A></LI>
   <LI><A href="">Digital SLR Cameras</A></LI>
</UL>
</LI>
</UL>
</BODY>
</HTML>
```

In the preceding code, a menu that displays the **Computers, Entertainment, and Cameras** product categories is created and stylized by using CSS. A product category contains different product items under it. All product items are hidden and only the **Computers, Entertainment, and Cameras** product categories are visible, as shown in the following figure.



The Product Menu

When a product category is clicked, the product items that belong to that specific product category appear to slide down by using the slide toggle effect provided by jQuery. After clicking the **Entertainment** product category, a submenu is displayed, as shown in the following figure.



The Submenu for the Computers Product Category

When the product category is clicked again, the product items appear to slide up.

Implementing Fade Effect

The jQuery fade effect is used to gradually reduce the opacity of the selected elements. You can fade an HTML element in and out of visibility. The following table lists the four functions to produce the fade effect.

Function	***Description***	***Syntax***
`fadeOut()`	*Is used to fade the appearance of the selected elements.*	`$(selector).fadeOut(speed)`
`fadeIn()`	*Is used to restore the appearance of the faded, selected element.*	`$(selector).fadeIn(speed)`
`fadeTo()`	*Is used to specify the opacity for fading the selected element.*	`$(selector).fadeTo(speed,opacity)`
`fadeToggle()`	*Is used to fade the appearance of the selected elements, when they are faded out, and restore the appearance of the selected elements, when they are faded in.*	`$(selector).fadeToggle(speed)`

The Fade Effect Functions

Consider the following code for implementing the fade effect:

```
<!DOCTYPE HTML><HTML>
<HEAD>
<SCRIPT type="text/javascript" src="jquery-1.8.3.js"></SCRIPT>
<SCRIPT type="text/javascript">
$(document).ready(function(){
  $("div").click(function(){
  $(this).fadeOut(1200);
  });
$("button").click(function(){
  $("div").fadeIn(1200);
  });
});
</SCRIPT>
</HEAD>
<BODY>
<DIV style="background:yellow;width:200px">FADE ME AWAY!</DIV><BR>
<BUTTON>Restore</BUTTON>
</BODY>
</HTML>
```

In the preceding code, the `div` tag is referred by using its name. In the click event handler of the div tag, the `this` keyword refers to the current HTML element, that is, the `div` tag. When the user clicks on the `div` tag, the text, `FADE ME AWAY!`, fades away. When the **Restore** button is clicked, the text, `FADE ME AWAY`!, reappears.

Implementing Animate Effect

The jQuery animate effect is used to create custom animations. You can create an animate function by using the following syntax:

```
animate({params},speed,callback);
```

where,

`params` specifies the CSS properties to be animated.

`speed` specifies the duration of the animation effect.

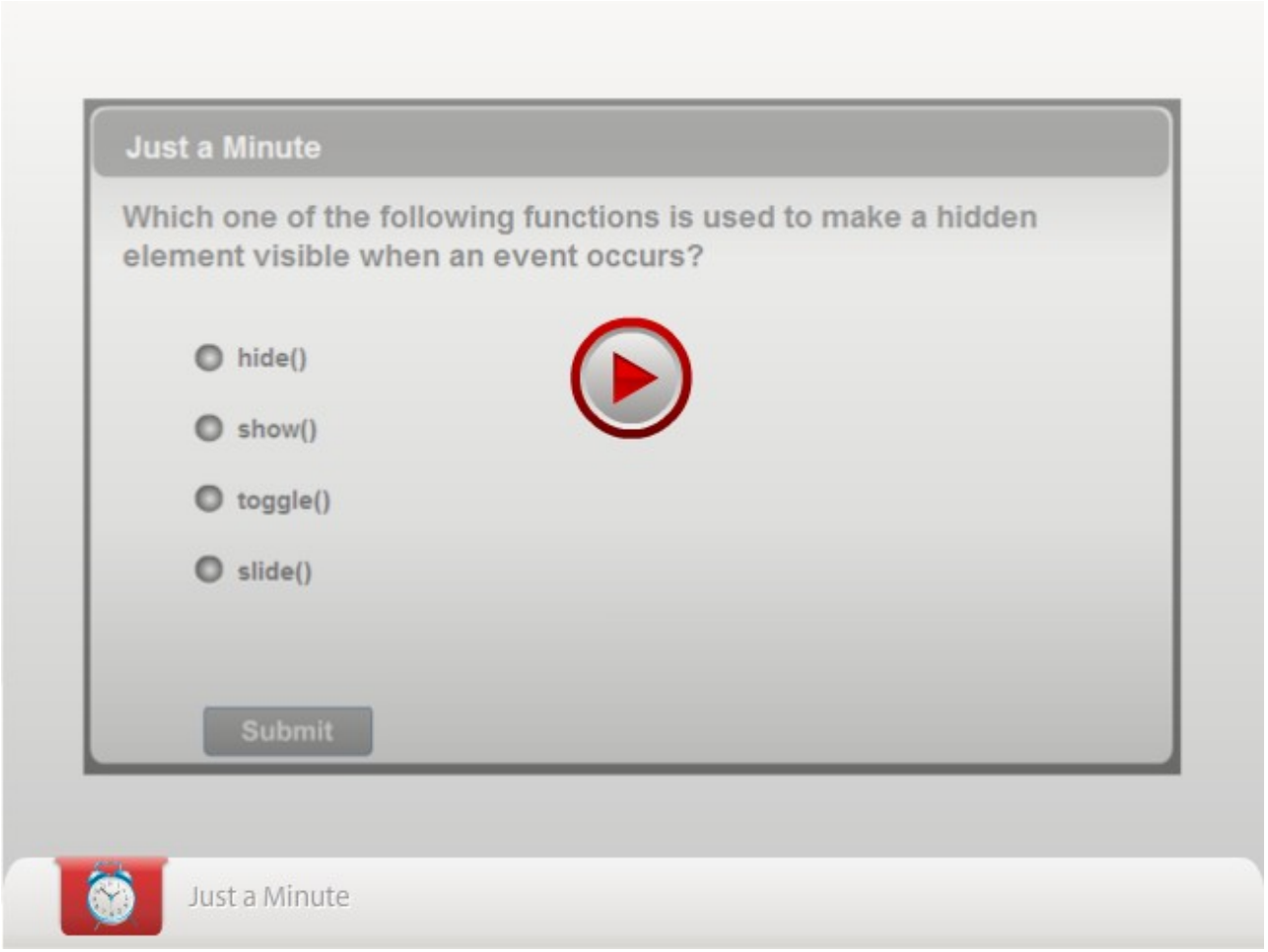
`callback` specifies the function that will be executed after completing the animation. It is an optional parameter.

Consider the following code snippet for applying animation effects on the `<DIV>` element:

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<SCRIPT src="jquery-1.8.3.js">
</SCRIPT>
<SCRIPT>
$(document).ready(function(){
  $("div").mouseenter(function(){
  $("div").animate({
```

```
        center:'250px',
        opacity:'0.5',
        height:'250px',
        width:'250px'
      });
      });
    $("div").mouseleave(function(){
      $("div").animate({
        opacity:'1',
        height:'100px',
        width:'100px'
      });
      });
    });
    </SCRIPT>
    </HEAD>
    <BODY>
    <DIV style="background:#98bf21;height:100px;width:100px;position:absolute;">
    </DIV>
    </BODY>
    </HTML>
```

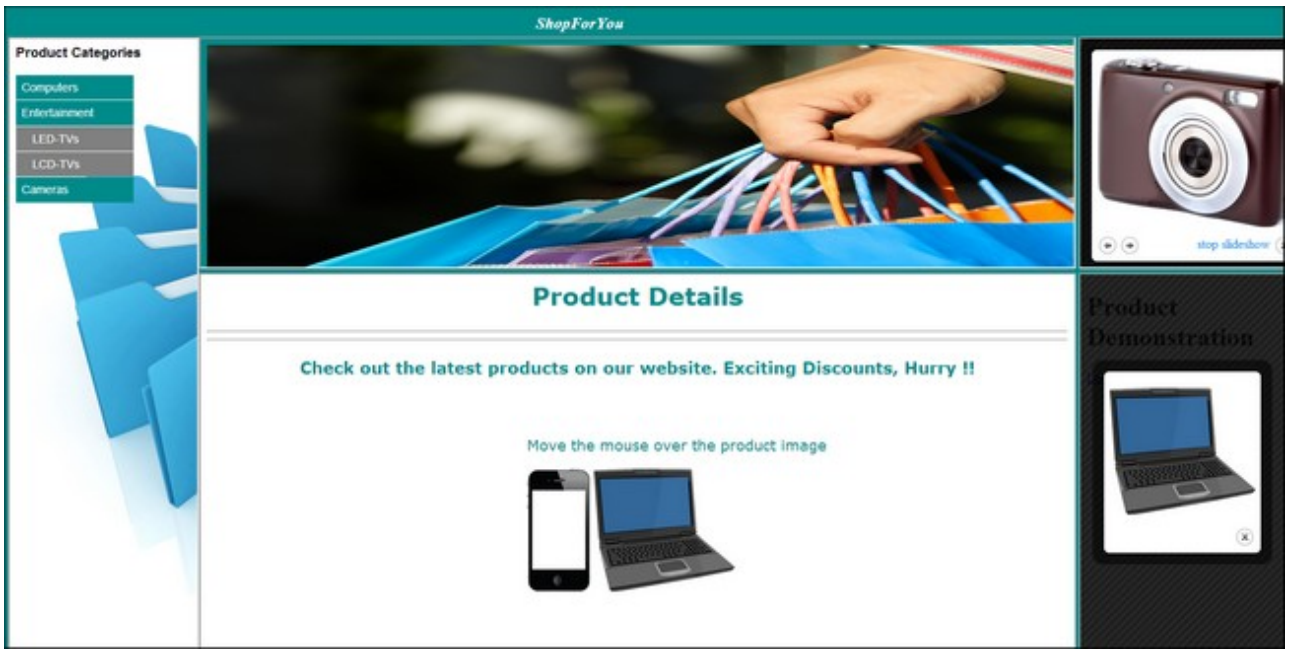
The preceding code will increase the size of the `<DIV>` element according to its specified width and height when the mouse is moved over it and will reduce the size of the `<DIV>` element according to its specified width and height when the mouse is left over it.



Activity 7.1: Exploring jQuery

Implementing Image Rollover

Consider the scenario of the ShopForYou.com website that sells products of various categories, such as phones and laptops. To give customers a preview of the appearance of these products, their images are displayed on the product page. To make the preview of these products appealing to the customers, you may want to add effects to the images displayed, such as when a customer places or moves the mouse pointer on an image, it appears enlarged or appears along with some information about that product. This can be achieved by replacing the image with a larger image of the same product or information about that product as plain text, as shown in the following figure.



The Sample Web Page

Creating Image Rollover

The images can be replaced by other images at runtime by creating the image rollover effect. JavaScript helps you to add the image rollover effect on Web pages.

Image rollover is an effect in which the appearance of an image changes when the mouse pointer is placed or moved on it. The change in the appearance can be achieved by replacing the original image with another image by using JavaScript.

In the ShopForYou.com website, the **Product1.png** and **Product2.png** images display images of the laptop and mobile, respectively, on the product page. You need to add a feature on the website so that when a customer moves or places the mouse pointer over the image of the laptop or mobile, its details are displayed. To accomplish this task, you need to add the image rollover effect to the **Product1.png** and **Product2.png** images.

To implement the rollover effect, you need to write the following code in a file and save it as **imagerollover.html**:

```
<!DOCTYPE HTML><HTML>
<HEAD>
<STYLE>
body{
background-color:white;
}
h2
{
font-family:verdana;
font-size:30px;
color:darkcyan;
text-align:center;
}
h3
{
```

```
text-align:center;
font-family:verdana;
color:darkcyan;
font-size:20px;
}
#para1
{
font-family:verdana;
font-size:18px;
text-align:center;
color:darkcyan;
}
</STYLE>
<SCRIPT>
function over(img,imgsrc)
{
img.src=imgsrc;
}
function out(img,imgsrc)
{
img.src=imgsrc;
}
</SCRIPT>
</HEAD>
<BODY>
<H2> Product Details </H2>
<HR><HR>
<H3> Check out the latest products on our website. Exciting Discounts, Hurry
!!</H3>
<DIV style="position:absolute; top:180px; left:400px;">
<P ID="para1"> Move the mouse over the product image </P>
<IMG ID="productImage1" src="Product1.png" height="150"
onmouseover="over(this,'Product1Details.jpg')" onmouseout="out
(this,'Product1.png')"/>
<IMG ID="productImage2" src="Product2.png" height="150"
onmouseover="over(this,'Product2Details.jpg')" onmouseout="out
(this,'Product2.png')"/>
</DIV>
</BODY>
</HTML>
```

In the preceding code, two functions, `over()` and `out()`, are invoked on the `onmouseover` and `onmouseout` events of the `<IMG>` tag, respectively. The **Product1.png** and **Product2.png** images are added to the Web page by using the `<IMG>` tag. When the customer moves the mouse pointer on the **Product1.png** image, the `over()` function is invoked, which changes the value of the `src` attribute of the image in the Web page to **Product1Details.jpg**. As a result, the **Product1.png** image is replaced with the **Product1Details.jpg** image. However, when the customer moves the mouse away from the **Product1Details.jpg** image, the `out()` function

is invoked, which changes the value of the `src` attribute of the image in the document to the **Product1.png** image. Therefore, the **Product1.png** image is redisplayed on the page. Similarly, the **Product2.png** image is replaced with the **Product2Details.jpg** image.

The following figure shows the output of the preceding code.

The Web Page Displaying the Images of Products

In the preceding Product Details Web page, if the mouse pointer is moved on the image of the laptop, it is replaced by another image that contains information of the product, as shown in the following figure.

The Web Page Displaying the Image Rollover Effect

Creating Backward Compatible Rollover

In most of the browsers, such as Firefox and Internet Explorer, the rollover effect can be created on images by using mouse events. However, in the earlier versions of browsers, mouse events on images cannot be captured. Therefore, the rollover effect cannot be added by using the mouse events on images in these browsers. To overcome this limitation, you need to create a backward compatible rollover.

For example, in the ShopForYou.com website, you need to add the rollover effect compatible for all browsers so that it can produce the desired output for all the customers, irrespective of the browser they are using.

The backward compatible rollover effect helps capture mouse events on images in all the browsers. To implement this effect, you need to use the `<A>` tag and place all the images by using the `<IMG>` tag within it. The

<A> tag can capture mouse events, such as onmouseover and onmouseout, on the images.

Consider the following code snippet that creates the backward compatible rollover effect:

```
<A href="#" onmouseover="over()" onmouseout="out()" >
<IMG ID="image1" src="SampleImage1.gif">
</A>
```

In the preceding code snippet, the <IMG> tag is placed in the <A> tag. The <A> tag captures the mouse events on the SampleImage1.gif image. The captured events can be used to create the rollover effect.

In the preceding examples, you have used only the ID of the image with the src attribute to refer to a single image and replaced it with another image. However, to implement the rollover effect on multiple images, you can use the document.images array for easy implementation. The document.images array stores reference of all the images in the current document. If a reference of any image exists in the document.images array, the document.images array returns true. Therefore, the document.images array can be used to check if the images exist or not.

The various images in the document.images array can be referenced by using either the ID or the index of the image as a subscript of the array. For example, the first image in the document can be referred to by using the index of the image, as shown in the following code snippet:

```
document.images[0]
```

Similarly, the second image is referred to as document.images[1] and so on.

The images in an array can also be referenced by using the ID attribute of the image as a subscript of the array, as shown in the following syntax:

```
document.images [ID];
```

For example, in the ShopForYou.com website, to implement the rollover effect on multiple images and to ensure compatibility with all the browsers, you can use the following code on the product page of the ShopForYou.com website:

```
<!DOCTYPE HTML><HTML>
<HEAD>
<STYLE>
h2
{
font-family:verdana;
font-size:30px;
text-align:center;
}
h3
{
font-family:verdana;
color:orange;
font-size:20px;
text-align:center;
}
#para1
{
font-family:verdana;
```

```
font-size:14px;
color:blue;
}
</STYLE>
<SCRIPT>
function over(img,imgsrc)
{
if(document.images)
{
document.images[img].src = imgsrc;
}
}
function out(img,imgsrc)
{
if (document.images)
{
document.images[img].src = imgsrc;
}
}
</SCRIPT>
</HEAD>
<BODY>
<H2> Product Categories </H2>
<HR><HR>
<H3> Check out the latest products on our website. Exciting Discounts, Hurry
!!
</H3>
<DIV style="position:absolute; top:180px; left:450px;">
<P ID="para1"> Choose the desired product image to view the product details
</P>
<A href="#" onmouseover="over('productImage1','Product1Details.jpg');"
onmouseout="out('productImage1','Product1.png');">
<IMG ID="productImage1" src="Product1.png" border="0"/>
</A>
<A href="#" onmouseover="over('productImage2','Product2Details.jpg');"
onmouseout="out('productImage2','Product2.png');">
<IMG ID="productImage2" src="Product2.png" border="0"/>
</A>
</DIV>
</BODY>
</HTML>
```

In the preceding code, the `<IMG>` tag defines the image source of the laptop and mobile, which are **Product1.png** and **Product2.png**, respectively. This tag is placed within the `<A>` tag. Therefore, the `<A>` tag enables you to capture mouse events on the **Product1.png** and **Product2.png** images. The `over()` and `out()` functions are invoked by using the `<A>` tag. The `over()` function is invoked when the user moves the mouse on

an image. The `out()` function is invoked when the customer moves the mouse away from an image. Both these functions take two parameters, ID and name, of the image to be replaced. The `over()` and `out()` functions check the images in the document by using an array called `document.images`.

When the customer moves the mouse pointer on the **Product1.png** image, the `over()` function is invoked. This function replaces the **Product1.png** image with the Product1Details.jpg image by changing the value of the `src` attribute to the **Product1Details.jpg** image.

When the customer moves the mouse away from the image, the `out()` function is called to reset the value of the `src` attribute to the **Product1.png** image. Similarly, the **Product2.png** image can be replaced with the **Product2Details.jpg** image.

Preloading Images

Preloading images is a technique to load images in the browser cache before the script on the Web page is executed and the Web page is rendered in the browser window. This helps to avoid any delay in loading images from their respective locations, and then displaying these images on the Web page. The `Image` object can be used to preload images in an application. To use the `Image` object, an instance of the `Image` object needs to be created and the actual images need to be attached to the `Image` object in the head section. It ensures that the images are loaded in memory before the body of the page gets loaded.

The browser cache is used to store the downloaded files so that the files are not required to be downloaded again and instead can be used from the browser cache when the user revisits a page.

The following syntax is used to create an instance of the `Image` object:

```
var Imagename= new Image([Width], [Height]);
```

In the preceding syntax:

- `Imagename`: Is the name of the new instance of the `Image` object.
- `[Width]`: Is the width of the image in pixels.
- `[Height]`: Is the height of the image in pixels.

While instantiating the `Image` object, both, the `[Width]` and `[Height]` parameters, are optional.

Consider the following code snippet that creates an instance of the `Image` object:

```
var img = new Image(40,30);
```

In the preceding code snippet, `img` is the name of the instance of the image object. The width and height of the image are assigned the values, `40` and `30`, respectively.

After instantiating an image object, you need to associate the image with the `Image` object. For this, the `src` attribute of the `Image` object is used. The syntax for attaching an image to an instance of the `Image` object is:

```
Imagename.src="ImageURL"
```

In the preceding syntax:

- `Imagename`: Is an instance of the `Image` object.
- `ImageURL`: Is the URL of the image.

Consider the following code snippet that attaches an image with the `Image` object:

```
img.src= SampleImage1.gif;
```

In the preceding code snippet, the `src` attribute of the `Image` object is used to attach the `SampleImage1.gif` image with the `img` object.

If an HTML document contains multiple images, you need to refer to all the images of the document while preloading images.

For example, in the ShopForYou.com website, you can use the `Image` object to preload the images and implement the rollover effect. For this, you can use the following code:

```
<!DOCTYPE HTML><HTML>
<HEAD>
<STYLE>
table{
text-align:center;
}
h2
{
font-family:verdana;
font-size:30px;
text-align:center;
}
h3
{
font-family:verdana;
color:orange;
font-size:20px;
text-align:center;
}
#para1
{
font-family:verdana;
font-size:14px;
color:blue;
text-align:center;
}
</STYLE>
<SCRIPT>
var images;
function preloadimages()
{
images = new
Array('product1.png','product2.png','product1details.jpg','product2details.jpg');
}
function over(a)
```

```
{
  document.images[a].src=images[a+2];
}
function out(a)
{
  document.images[a].src=images[a];
}
</SCRIPT>
</HEAD>
<BODY onLoad=preloadimages()>
<H2> Product Categories </H2>
<HR><HR>
<H3> Check out the latest products on our website. Exciting Discounts, Hurry
!!</H3>
<DIV style="position:absolute; top:180px; left:400px;">
<P ID="para1"> Move the mouse cursor over the product image to view the
product details
</P>
<TABLE border="0">
<TR>
<TD>
<IMG src="product1.png" onmouseover="over(0)" onmouseout="out(0)" />
</TD>
<TD>
<IMG src="product2.png" onmouseover="over(1)" onmouseout="out(1)" />
</TD>
</TR>
</BODY>
</HTML>
```

In the preceding code, an array of images is created in the `preloadimages()` function. This array is passed as a parameter to the `preload()` function. In the `preload()` function, the images are preloaded by using the `Image` object. The images of the document are referred to by using the `document.images` array inside the `over()` and `out()` functions.

Creating Image Gallery

Consider the scenario of ShopForYou.com where an image gallery is used to display images of the products available at the website. An image gallery is a collection of thumbnail pictures or image links. All of these can be clicked individually to provide a large view of the corresponding product on another Web page. This image gallery is implemented by using JavaScript. A disadvantage of this feature is that to preview an image, the customer needs to navigate to the previous page repeatedly.

To avoid repeated navigation from a page to the previous page, you can display the full-size image on the same page. For this, you need to use jQuery plugins to create the image gallery. The jQuery plugins are prewritten and easy to use programs to enhance the jQuery code. By using jQuery plugins, you can create the image gallery by displaying the images dynamically at runtime on the same Web page. The jQuery plugins can be used to create an image gallery by:

- Using colorbox plugin
- Using galleria plugin

The light-weight, colorbox plugin enables displaying a collection of images one by one by using the previous and next buttons. The colorbox plugin can be used to display an image with a variety of transition effects, such as fade or elastic, in the middle of an elegant frame. This frame is known as lightbox and is created at the center of the Web page over a translucent film surrounding it. The colorbox plugin can be used effectively for creating and displaying an image gallery or a slideshow of images on the Web page where the image sequence can be controlled by using the previous and next buttons. To create an elegant image gallery or a slideshow on a Web page, you can download the **colorbox.zip** file that contains the necessary folders and files to implement the colorbox plugin functionalities. Then, you can extract the colorbox folder from the **colorbox.zip** file. The colorbox folder comprises the following folders:

- **colorbox**: This folder comprises the **jquery.colorbox.js** file, which is the colorbox plugin file. This colorbox plugin file is used to implement the lightbox, slideshow, and the transition effects to display images in the image gallery.
- **example1**, **example2**, **example3**, **example4**, and **example5**: These folders contain sample implementation of the colorbox plugin. Each folder illustrates a unique and customized implementation of the colorbox plugin to create an image gallery or a slideshow. Each of these folders contains the following subfolders and files:
    - **colorbox.css**: This style sheet contains the formatting instructions to stylize, position, and decorate the image gallery as it is created by using the colorbox plugin. For example, background of the Web page, border of the image, and the previous, next, and close buttons.
    - **images folder**: This folder contains images that can be used to create and control the image gallery or slideshow. These images can be stylized and positioned on the Web page according to the instructions specified in the colorbox.css style sheet.
    - **index.html**: This Web page is used to illustrate how the image gallery can be displayed by using the elastic, fade, or no transition effects. In addition, this Web page is used to display a slideshow of the images.

Colorbox is a jQuery plugin. Therefore, to implement the functionality provided by this plugin, you need the `jquery-1.8.3.js` file. In addition, you need the jQuery event method, `document.ready()`, which is required to invoke the colorbox plugin to implement the slideshow or image gallery functionality.

Slideshow is used to display images one by one after a defined duration without using the previous and next buttons. However, in the image gallery, the user is required to use the previous and next buttons to scroll through all the images.

Let us consider an example to implement the colorbox plugin and create a lightbox for the image, **Product1.png**. For this, you need to perform the following steps:

1. Create a folder, such as **demo,** in any drive of your system.
2. Copy the downloaded **colorbox** folder to this folder.
3. Create the **jQueryS**, **CSS**, and **contentFiles** folders in the **demo** folder. This is done to ensure that the jquery file (**jquery-1.8.3.js)**, jquery plugin file (**jquery.colorbox.js)**, style sheet (**colorbox.css)**, and images (**Product1.png**) to be used are stored separately for easy implementation.

You can create the folder by any name of your choice. In fact, it is not necessary to create a new folder. This is done to ensure that all your related files are stored at one place. Depending on the storage location of your files, you need to specify the relative or

absolute path of the respective files in the `src` attribute of the `<SCRIPT>` tag.

4. Copy the **jquery.colorbox.js** file from the **colorbox** folder available in the downloaded **colorbox** plugin folder to the **jQueryS** folder.
5. Copy the **colorbox.css** file from the **example1** folder available in the downloaded **colorbox** plugin folder to the **CSS** folder.
6. Copy the downloaded **jquery-1.8.3.js** jQuery file to the **jQueryS** folder.
7. Copy the **Product2.png** file to the **contentFiles** folder.
8. Copy the **images** folder from the **example1** folder available in the downloaded **colorbox** plugin folder to the **CSS** folder.
9. Type the following code in Notepad, and save the file as **LightboxDemo.html** at the same location where you have created the **jQueryS**, **CSS**, and **contentFiles** folders on your computer:

```
<!DOCTYPE HTML>
<HTML>
  <HEAD>
  <LINK media="screen" rel="stylesheet" href="CSS/colorbox.css" />
  <SCRIPT src="jQueryS/jquery-1.8.3.js"></SCRIPT>
  <SCRIPT src="jQueryS/jquery.colorbox.js"></SCRIPT>
  <SCRIPT type="text/javascript">
    $(document).ready(function(){
      $("a").colorbox();
    });
  </SCRIPT>
</HEAD>
<BODY>
  <H1>Product Demonstration</H1>
  <A href="contentFiles/Product2.png">Laptop</A>
</BODY>
</HTML>
```

In the preceding code, `a` represents the `<A>` element defined inside the `<BODY>` element. The `colorbox()` method of the colorbox plugin is used to display the **Product1.png** image inside the lightbox.

10. Open the **LightboxDemo.html** file in Microsoft Internet Explorer.

*Click the **Allow blocked content** button if the following message is displayed: **Internet Explorer restricted this webpage from running scripts of ActiveX controls**.*

11. Click the **Laptop** hyperlink. The slide view of the product image is displayed with the default transition effect by using lightbox, as shown in the following figure.

The Lightbox Displaying Product Image

In the preceding figure, lightbox displays a single image with the default transition effects. However, you can customize the display of images in an image gallery or a slideshow by using the colorbox plugin. For this, the colorbox plugin provides several keys, which need to be passed as parameters to the `colorbox()` method. The following table lists the keys that can be used with the `colorbox()` method.

Key	***Description***
`transition`	*It specifies the type of transition, such as* `fade`*,* `elastic`*, or* `none`*, required for an image on the Web page. Its default value is* `elastic`*.*
`speed`	*It specifies the speed of transition with which the image is displayed. Its default value is* `350 milliseconds`*.*
`width`	*It specifies the total width of the frame in which the image is displayed. Its default value is* `false`*, which means that the width of the frame is not fixed and varies according to the width of the image.*
`height`	*It specifies the total height of the frame in which the image is displayed. Its default value is* `false`*, which means that the height of the frame is not fixed and varies according to the height of the image.*

innerWidth	*It specifies the inner width of the frame in which the image is displayed. This does not include the border and buttons.*
innerHeight	*It specifies the inner height of the frame in which an image is displayed. This does not include the border and buttons. Its default value is* `false`.
current	*It specifies the text that appears when multiple images are displayed by using lightbox, such as* `image 1 of 5` *when the first image in a group of five images is displayed. Its default value is* `image{current} of {total}`. *The {current} is detected and replaced with the actual number of images being displayed while ColorBox runs. {total} is detected and replaced with the actual number of images in the gallery.*
slideshow	*It specifies whether the slideshow of images needs to be created when multiple images are grouped together for preview. Its default value is* `false`.

The Keys of the Colorbox Plugin

Consider the following code to create a lightbox by specifying the height and width of the frame in which the image is displayed:

```
<!DOCTYPE HTML>
<HTML>
  <HEAD>
  <LINK media="screen" rel="stylesheet" href="CSS/colorbox.css" />
  <SCRIPT src="jQueryS/jquery-1.8.3.js"></SCRIPT>
  <SCRIPT src="jQueryS/jquery.colorbox.js"></SCRIPT>
  <SCRIPT type="text/javascript">
    $(document).ready(function(){
```

```
    $("a").colorbox({transition:"fade", height:"250", width:"220"});
      });
    </SCRIPT>
  </HEAD>
  <BODY>
    <H1>Product Demonstration</H1>
    <A href="contentFiles/Product2.png">Laptop</A>
  </BODY>
  </HTML>
```

The preceding code creates a lightbox of height `250 px` and width `220 px` and displays the image inside this lightbox by using the `fade` transition. The output of the preceding code after the `Product1 photo` link is clicked is displayed, as shown in the following figure.

The Lightbox Displaying Product Image

You can also control the display of a set of images in an image gallery with the previous and next buttons. To display a set of images or a group of images by using a lightbox, you need to add a link to each image by using the `<A>` tag and specify the same value for the `rel` attribute of each `<A>` tag defined for the images. The `rel` attribute indicates the relationship between the current document and the path specified by the `href` attribute. By providing the same value for the `rel` attribute for all the images, these can be related to a single group.

Suppose, you need to create a lightbox for the set of images, **Product1.png**, **Product2.png**, and **Product3.png**. For this, you need to perform the following steps:

1. Copy the **Product2.png** and **Product3.png** files to the **contentFiles** folder.
2. Type the following code in Notepad, and save the file as **LightboxGroupDemo.html** at the same location where you have created the **jQueryS**, **CSS**, and **contentFiles** folders on your computer:

```
<!DOCTYPE HTML>
<HTML>
  <HEAD>
  <LINK media="screen" rel="stylesheet" href="CSS/colorbox.css" />
  <SCRIPT src="jQueryS/jquery-1.8.3.js"></SCRIPT>
  <SCRIPT src="jQueryS/jquery.colorbox.js"></SCRIPT>
  <SCRIPT type="text/javascript">
    $(document).ready(function(){
```

```
        $("a").colorbox();
      });
    </SCRIPT>
  </HEAD>
  <BODY>
    <H1>Product Demonstration</H1>
  <A rel="collection" href="contentFiles/Product1.png">Click here to
  view</A>
  <A rel="collection" href="contentFiles/Product2.png"></A>
  <A rel="collection" href="contentFiles/Product3.png"></A>
  </BODY>
  </HTML>
```

In the preceding code, the `rel` attribute of each `<A>` tag defined inside the body tag is specified with the value, `collection`. This ensures that the images having the same value for the `rel` attribute are grouped together so that these images can be displayed as part of a single image gallery.

When you save the preceding code in an HTML file, and then open it in Microsoft Internet Explorer, the **Click here to view** hyperlink is displayed. When you click the hyperlink, the output is displayed with the previous and next buttons, as shown in the following figure.

The Product Image with the Previous and Next Buttons

*Click the **Allow blocked content** button if the following message is displayed:*
***Internet Explorer restricted this webpage from running scripts of ActiveX controls**.*

You can create a slideshow of images by grouping these images by using the colorbox plugin. For this, you need to use the same value for the `rel` attribute for all the images to be displayed in the `<A>` tag and set the `slideshow` key of the `colorbox()` method to true.

Consider the following code that creates a slideshow and saves it as **SlideShow.html**:

```
<!DOCTYPE HTML>
<HTML>
```

```
  <HEAD>
  <STYLE>
body{
Background-color: "silver";}
</STYLE>
  <LINK media="screen" rel="stylesheet" href="css/colorbox.css" />
  <SCRIPT src=" jQueryS/jquery-1.8.3.js"></SCRIPT>
  <SCRIPT src=" jQueryS/jquery.colorbox.js"></SCRIPT>
  <SCRIPT>
    $(document).ready(function(){
      $("a").colorbox({slideshow:true,current:false});

    });
  </SCRIPT>
</HEAD>
<BODY>
<A href=" contentFiles/Product1.png" rel="collection">Click Here to View
Slideshow of Products Available at the Site
</A>
<A href=" contentFiles/Product2.png" rel="collection"></A>
<A href=" contentFiles/Product3.png" rel="collection"></A>
</BODY>
</HTML>
```

In the preceding code, the `slideshow` key is set to true to create a slideshow of the images. The value of the `current` key is set to `false`. This prevents displaying the text depicting the number of the current image, such as image 2 of 5, during the slideshow. The output of the preceding code is shown in the following figure.

The Slideshow of the Images

Using galleria Plugin

The galleria plugin is the jQuery plugin used for creating elegant and professional-looking image galleries. This plugin is used to display an image gallery and create thumbnails of the images to be displayed in the gallery

automatically. When the mouse pointer is moved on a thumbnail image, a larger version of the image is displayed by using a transition effect on the same Web page. Unlike the colorbox plugin, the galleria plugin cannot be used to create a slideshow of images, but the galleria plugin has the ability to create thumbnails of the images to be displayed in the image gallery.

To create an image gallery by using the galleria plugin, you need to download the **galleria.zip** file from the Internet and extract the galleria folder from the **galleria.zip** file at a desired location on your system. The galleria folder comprises the **galleria-1.2.8.js** file, which is the galleria plugin to be used to implement the functionality of creating the image gallery.

Activity 7.2: Creating an Image Gallery

Summary

In this chapter, you learned that:

- jQuery is an open source and cross-browser library of JavaScript codes that was created to make the JavaScript programming simpler.
- You can make use of jQuery in a Web page to manipulate HTML elements by downloading the light-weight jQuery JavaScript library and saving it to your system.
- To ensure that a Web page is fully loaded before the jQuery code is executed on it, jQuery provides the `document.ready()` function.
- Using jQuery, HTML elements can be selected by referring to their name, ID, or class to which they belong.
- jQuery provides CSS selectors to modify the CSS properties of an HTML element or document.
- In jQuery, events are handled by using functions or predefined event methods.
- Some of the predefined jQuery effects that can be used to add visual appeal to a Web page are:
    - Hide
    - Show
    - Toggle
    - Slide
    - Fade
    - Animate
- Image rollover is an effect in which the appearance of an image changes when the mouse pointer is placed or moved on it.
- Preloading images is a technique to load images in the browser cache before the script on the Web page is executed and the Web page is rendered in the browser window.
- The `Image` object can be used to preload images in an application.
- An image gallery is a collection of thumbnail pictures or image links.
- By using jQuery plugins, you can create the image gallery by displaying the images dynamically at runtime on the same Web page.
- The light-weight, colorbox plugin enables displaying a collection of images one by one by using the previous and next buttons.
- The galleria plugin is the jQuery plugin used for creating elegant and professional-looking image galleries.

Reference Reading

Exploring jQuery

Reference Reading: Books	*Reference Reading: URLs*
JavaScript: the complete reference By Thomas A. Powell, Fritz Schneider	*http://www.w3schools.com/jquery/default.asp* *http://jquery.com/*

Adding Visual Effects Using jQuery

Reference Reading: Books	*Reference Reading: URLs*
JavaScript: the complete reference By Thomas A. Powell, Fritz Schneider	*http://api.jquery.com/category/effects/*

Implementing Image Rollover

Reference Reading: Books	*Reference Reading: URLs*
JavaScript: the complete reference By Thomas A. Powell, Fritz Schneider	*http://www.irt.org/articles/js132/*

Creating Image Gallery

Reference Reading: Books	*Reference Reading: URLs*
JavaScript: the complete reference By Thomas A. Powell, Fritz Schneider	*http://monc.se/kitchen/146/galleria-a-javascript-image-gallery*