Full Stack Development Lab

Aim:

Week 7: Write a Program on Java Script Functions and Events

Description:

JavaScript Events

The change in the state of an object is known as an **Event**. In html, there are various events which represents that some activity is performed by the user or by the browser. When javascript code is included in HTML, js react over these events and allow the execution. This process of reacting over the events is called **Event Handling**. Thus, js handles the HTML events via **Event Handlers**.

For example, when a user clicks over the browser, add js code, which will execute the task to be performed on the event.

Some of the HTML events and their event handlers are:

Mouse events:

Event Performed	Event Handler	Description
click	onclick	When mouse click on an element
mouseover	onmouseover	When the cursor of the mouse comes over the element
mouseout	onmouseout	When the cursor of the mouse leaves an element
mousedown	onmousedown	When the mouse button is pressed over the element
mouseup	onmouseup	When the mouse button is released over the element
mousemove	onmousemove	When the mouse movement takes place.

Keyboard events:

Event Performed	Event Handler	Description
Keydown & Keyup	onkeydown & onkeyup	When the user press and then release the key

Form events:

Event Performed	Event Handler	Description
focus	onfocus	When the user focuses on an element
submit	onsubmit	When the user submits the form
blur	onblur	When the focus is away from a form element
change	onchange	When the user modifies or changes the value of a form element

Window/Document events

Event Performed	Event Handler	Description
load	onload	When the browser finishes the loading of the page
unload	onunload	When the visitor leaves the current webpage, the browser unloads it
resize	onresize	When the visitor resizes the window of the browser

Click Event

```
<html>
<head> Javascript Events </head>
<body>
<script language="Javascript" type="text/Javascript">
    <!--
    function clickevent()
    {
        document.write("This is GRIET Website");
    }
    //-->
</script>
<form>
<input type="button" onclick="clickevent()" value="Who's this?"/>
</form>
</body>
</html>
```

Output:practice in LAB

MouseOver Event

```
<html>
<head>
<h1> Javascript Events </h1>
</head>
<body>
<script language="Javascript" type="text/Javascript">
    <!--
    function mouseoverevent()
    {
        alert("This is JavaTpoint");
    }
    //-->
</script>
<p onmouseover="mouseoverevent()"> Keep cursor over me</p>
</body>
```

```
</html>
```

Output:practice in LAB

Focus Event

```
<html>
<head> Javascript Events</head>
<body>
<h2> Enter something here</h2>
<input type="text" id="input1" onfocus="focusevent()"/>
<script>
<!--
    function focusevent()
    {
        document.getElementById("input1").style.background=" aqua";
    }
//-->
</script>
</body>
</html>
```

Output:practice in LAB

Keydown Event

```
<html>
<head> Javascript Events</head>
<body>
<h2> Enter something here</h2>
<input type="text" id="input1" onkeydown="keydownevent()"/>
<script>
<!--
    function keydownevent()
    {
        document.getElementById("input1");
        alert("Pressed a key");
    }
//-->
</script>
</body>
</html>
```

Output:practice in LAB

Load event

```
<html>
<head>Javascript Events</head>
</br>
<body onload="window.alert('Page successfully loaded');">
<script>
<!--
document.write("The page is loaded successfully");
//-->
</script>
</body>
</html>
```

Output:practice in LAB

7.2 JavaScript addEventListener()

The **addEventListener()** method is used to attach an event handler to a particular element. It does not override the existing event handlers. Events are said to be an essential part of the JavaScript. A web page responds according to the event that occurred. Events can be user-generated or generated by API's. An event listener is a JavaScript's procedure that waits for the occurrence of an event.

The addEventListener() method is an inbuilt function of JavaScript. We can add multiple event handlers to a particular element without overwriting the existing event handlers.

Syntax

```
element.addEventListener(event, function, useCapture);
```

Although it has three parameters, the parameters ***event*** and ***function*** are widely used. The third parameter is optional to define. The values of this function are defined as follows.

Parameter Values

event: It is a required parameter. It can be defined as a string that specifies the event's name.

Note: Do not use any prefix such as "on" with the parameter value. For example, Use "click" instead of using "onclick".

function: It is also a required parameter. It is a JavaScript function which responds to the event occur.

useCapture: It is an optional parameter. It is a Boolean type value that specifies whether the event is executed in the bubbling or capturing phase. Its possible values are **true** and **false**. When it is set to true, the event handler executes in the capturing phase. When it is set to false, the handler executes in the bubbling phase. Its default value is **false**.

Let's see some of the illustrations of using the addEventListener() method.

Code:

```
<!DOCTYPE html>
<html>
<body>
<p> This is an example of adding multiple events to the same element. </p>
<p> Click the following button to see the effect. </p>
<button id = "btn"> Click me </button>
<p id = "para"></p>
<p id = "para1"></p>
<script>
function fun() {
   alert("Welcome to the GRIET college");
}
```

```

function fun1() {
   document.getElementById("para").innerHTML =  "This is second function";

}
function fun2() {
   document.getElementById("para1").innerHTML =  "This is third function";
}
var mybtn = document.getElementById("btn");
mybtn.addEventListener("click", fun);
mybtn.addEventListener("click", fun1);
mybtn.addEventListener("click", fun2);
</script>
</body>
</html>
```

7.3 JavaScript dblclick event

The **dblclick** event generates an event on double click the element. The event fires when an element is clicked twice in a very short span of time. We can also use the JavaScript's **addEventListener()** method to fire the double click event.

In HTML, we can use the **ondblclick** attribute to create a double click event.

Syntax

Now, we see the syntax of creating double click event in HTML and in javascript (without using **addEventListener()** method or by using the **addEventListener()** method).

In HTML

```
<element ondblclick = "fun()">
```

In JavaScript

```
object.ondblclick = function() { myScript };
```

In JavaScript by using the addEventListener() method

```
object.addEventListener("dblclick", myScript);
```

Let's see some of the illustrations to understand the double click event.

Example - Using ondblclick attribute in HTML

In this example, we are creating the double click event using the HTML **ondblclick** attribute.

```
<!DOCTYPE html>
<html>
<head>
</head>

<body>
<h1 id = "heading" ondblclick = "fun()"> Hello world :):) </h1>
<h2> Double Click the text "Hello world" to see the effect. </h2>
<p> This is an example of using the <b> ondblclick </b> attribute. </p>
<script>
function fun() {
document.getElementById("heading").innerHTML = " Welcome to the GRIET College ";
}
</script>
</body>
</html>
```

Output:practice in LAB

7.4 JavaScript onload

In JavaScript, this event can apply to launch a particular function when the page is fully displayed. It can also be used to verify the type and version of the visitor's browser. We can check what cookies a page uses by using the **onload** attribute.

In HTML, the onload attribute fires when an object has been loaded. The purpose of this attribute is to execute a script when the associated element loads.

In HTML, the **onload** attribute is generally used with the **<body>** element to execute a script once the content (including CSS files, images, scripts, etc.) of the webpage is completely loaded. It is not necessary to use it only with <body> tag, as it can be used with other HTML elements.

The difference between the **document.onload** and **window.onload** is: **document.onload** triggers before the loading of images and other external content. It is fired before the **window.onload**.

While the **window.onload** triggers when the entire page loads, including CSS files, script files, images, etc.

Syntax

```
window.onload = fun()
```

Let's understand this event by using some examples.

Example1

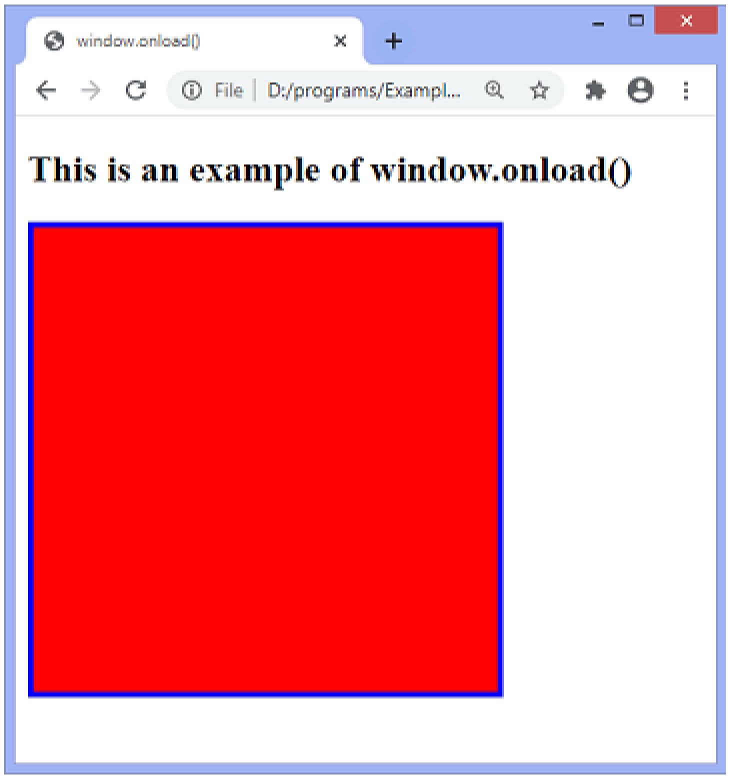
In this example, there is a div element with a height of 200px and a width of 200px. Here, we are using the **window.onload()** to change the background color, width, and height of the **div** element after loading the web page.

The background color is set to **'red'**, and width and height are set to **300px** each.

```
<!DOCTYPE html>
<html>
<head>
<meta charset = " utf-8">
<title> window.onload() </title>
<style type = "text/css">
#bg{
width: 200px;
height: 200px;
border: 4px solid blue;
}
</style>
<script type = "text/javascript">
window.onload = function(){
document.getElementById("bg").style.backgroundColor = "red";
document.getElementById("bg").style.width = "300px";
document.getElementById("bg").style.height = "300px";
}
</script>
</head>
<body>
<h2> This is an example of window.onload() </h2>
<div id = "bg"></div>
</body>
</html>
```

Output

After the execution of the code and loading of the page, the output will be -



This is an example of window.onload()

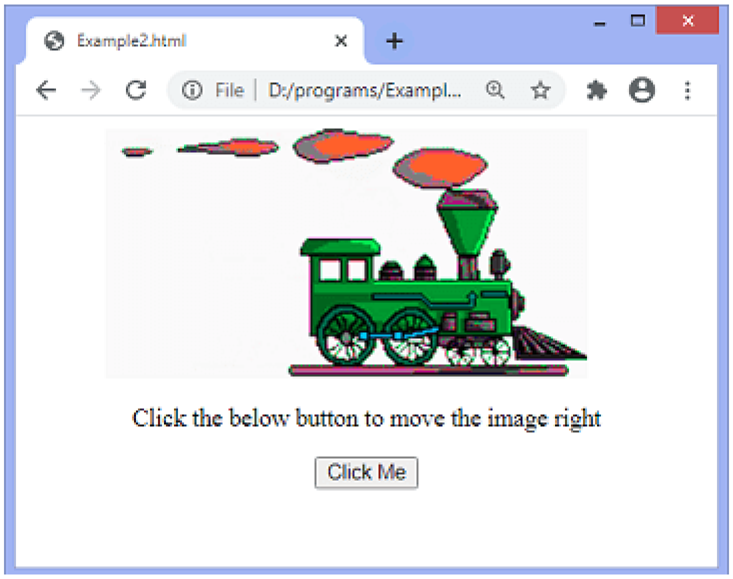
Example2

In this example, we are implementing a simple animation by using the properties of the DOM object and functions of javascript. We use the JavaScript function getElementById() for getting the DOM object and then assign that object into a global variable.

```
<html>
  <head>
    <script type = "text/javascript">

          var img = null;
         function init(){
             img = document.getElementById('myimg');
            img.style.position = 'relative';
             img.style.left = '50px';
         }
          function moveRight(){
            img.style.left = parseInt(
            img.style.left) + 100 + 'px';
        }
         window.onload = init;

    </script>
  </head>

  <body>
    <form>
      <img id = "myimg" src = "train1.png" />
       <center>
         <p>Click the below button to move the image right</p>
       <input type = "button" value = "Click Me" onclick = "moveRight();" />
    </center>
    </form>
  </body>

</html>
```

Output

After the successful execution of the above code, the output will be -



Click the below button to move the image right

Result: Thus, in the above programs successfully executed without errors Using Java script functions and events in eclipse editor**.**