**Web Engineering (Practical) Lab 2**

In this lab you should learn:

* Basic Introduction about JavaScript
* Some Features of JavaScripts
* Inline(embedded) , Internal and External adding of JavaScript Code

**What is JavaScript?**

JavaScript makes HTML pages more dynamic and interactive.

The **<script>** tag is used to define a client-side script (JavaScript).

The <script> element either contains scripting statements, or it points to an external script file through the **src** attribute.

Common uses for JavaScript are image manipulation, form validation, and dynamic changes of content.

To select an HTML element, JavaScript very often use the document.getElementById(id) method.

### JavaScript can change HTML content

This JavaScript example writes "Hello JavaScript!" into an HTML element with id="demo":

<script>  
document.getElementById("demo").innerHTML = "Hello JavaScript!";  
</script>

### JavaScript can change HTML styles

document.getElementById("demo").style.fontSize = "25px";  
document.getElementById("demo").style.color = "red";

### JavaScript can change HTML attributes

document.getElementById("image").src = "picture.gif";

The HTML <noscript> Tag

The **<noscript>** tag is used to provide an alternate content for users that have disabled scripts in their browser or have a browser that doesn't support client-side scripts:

<script>  
document.getElementById("demo").innerHTML = "Hello JavaScript!";  
</script>  
  
<noscript>Sorry, your browser does not support JavaScript!</noscript>

JavaScript can hide HTML contents

<p id="demo">JavaScript can hide HTML elements.</p>

<button type="button" onclick="document.getElementById('demo').style.display='none'">Click Me!</button>

**Inline (embedded) JavaScript code: Event Handlers.**

Most scripts are activated when some sort of **event** happens. Whenever an event happens on a page, the **browser detects it**. **Scripts that handle events** are referred to, appropriately, as **event handlers**. Event handlers that handle events associated with specific HTML elements on the page can be incorporated into element tags.

* General syntax:
* <someHTMLtag ID="[tagID]" on[Event] = "[JavaScript code]">
* In response to the click of the command button "Hello", **call the alert method of the window object** to display a message box with the messag "Hello World!". (Notice the usage of double and single quotation marks).
* <input type="button" id="hello" value="Hello" onClick = "window.alert('Hello World!')">
* In response to a click on the button "Hello", **call the write method of the document object** to create a new web page displaying "Hello World!" as a h1 centered heading.
* <input type="button" id="hello" value="Hello"
* onClick = "document.write('<center><h1>Hello World!</h1></center>')">
* You can have more than one event handler in the same tag (the example uses **assignment statements** and a keyword **this** that stands for a current element):
* <span id="demo" style="font-size:18pt; font-weight:bold;
* background-color:red; color:gold"
* onMouseover = "this.style.backgroundColor='magenta'"
* onMouseout = "this.style.backgroundColor='#0000ff'"
* onClick = "this.style.visibility='hidden'">
* Point Mouse Here
* </span>

**Point Mouse Here**

* You can have more than one statement in the same event handler:
* / \*On click of cmd\_view\_cart, hide div\_select, display div\_cart,
* and move to top to display the logo: \*/
* <button type="button" id="cmd\_view\_cart" onClick="div\_select.style.display='none';
* div\_cart.style.display='block';
* document.location.href='#top'">
* Add to Shopping Cart</button>
* We could also have reversed the usage of the **two types of quotation marks** and used single quotes to surround the event handling code and double quotes to surround the file name. The important thing is to use the same type of quotation mark to begin and end each part of the code.
* **onLoad** handler in the **body** tag can be used to initialized variable, and element attribute values, as well as to perform initial actions right at the start.
* <body onLoad="alert('The document has been loaded.'); cmd\_view\_cart.disabled='true'">

**Internal JavaScript code: Using <script> tags.**

* Every element in HTML has a tag, and scripts are no different. **Script tags** generally look like this:
* <script language="JavaScript">
* <!-- // hide from the older browsers
* [The actual script commands]
* //--> // stop hiding
* </script>
* Example: Displaying the **lastModified** date of a document.
* <script language="JavaScript">
* <!--
* document.write("<h2>This document was last modified "
* + document.lastModified + "</h2>")
* //-->
* </script>
* Scripts can be inserted **anywhere on a page**, and **in many places** depending on the script writer's preference and the needs of the page.
* When possible, it is a good idea to **keep all scripts in the same place** on an HTML page to make them easier to find and to aid in the debugging process.
* Keeping a script **in the header region** also ensures that the script is loaded before the rest of the page, which can be helpful when you are debugging.
* You can use script **blocks of different languages**, such as VBScript and JScript, **on the same page**.
* <input type="button" name="cmdVBScript" value="VBScript">
* <script language="VBScript">
* <!--
* sub cmdVBScript\_onClick()
* msgBox "This is VBScript message box!", \_
* vbInformation, "Hello from VBScript"
* end sub
* -->
* </script>

**External JavaScript code: Using code stored in a separate .js file.**

1. **First**, you must have JavaScript code stored in a **file** with **js** extension, e.g. external\_js.js
2. function printCurrentDate() {
3. var currentDate = new Date() //create a new Date() object variable.
4. var localDate = currentDate.toLocaleString() //set to the current time zone string
5. var strToday = localDate.substring(0,40) //take first 10 characters from the string
6. document.write("<blockquote><h2>Today is " + strToday + "</h2></blockquote>")

}

1. **Second**, you must **refer in your HTML page** to this external JavaScript file as a **src**:
2. <script language="JavaScript" src="examples/externaljs.js"></script>
3. **Third**, then you must **call the function from your HTML page**:
   * Either from an **internal JavaScript:**.
   * <script language="JavaScript">
   * <!--
   * printCurrentDate()
   * //-->
   * </script>

**Today is 2/7/2017, 9:26:16 AM**

* + Or you may call from an **inline event handler**.
  + <input type="button" value="Print Current Date" onClick="printCurrentDate()" />

1. The **Advantages of external JavaScript:**
   * If you have some script that you would like to **access from many different pages** on your site, you can simply place the script in an external file that is referenced by every such page.
   * An external code can be **reused** by many applications.
   * By using one source, you avoid the problem of accidentally using different versions of your script in different files.
   * Once downloaded, the script file **is cached** (stored locally on the user's machine), which means that the script does not need to be downloaded separately as part of every Web page. This can reduce downloading time substantially. This technique works in Netscape Navigator 3 and later and Internet Explorer 3.02 and later.

**Adding JavaScript to links.**

* General form:
* <a href="javascript:[JavaScript code]">Label to click</a>
* Example:
* <a href="javascript:alert('Hello from a link')">Click to Hello</a>

[Click to Hello](javascript:alert('Hello%20from%20a%20link'))

**Tasks :**

* Use Javascript in all three modes (Internal,External and Inline)
* Use five Events of Javascript in a HTML Form based WebPage