ASSIGNMENT A3

Title:

Perform validations using JavaScript/JQuery.

Problem Statement/Definition:

Perform validations using JavaScript / JQuery on one of the applications from the given list :

1. Online pizza order application
2. Student information system for training & placement department
3. Leave management application
4. Blogging platform
5. Meeting room booking application
6. Exam cell automation application

Objective:

- To understand in depth working of JavaScript and Jquery
- To use JavaScript and Jquery for providing validations, animation and effects to any web application.

S/W Packages and Hardware apparatus used:

- Operating System open source Fedora 20 Networked computer with internet access.
- Editor : IDE : Netbeans 8.1
- Web browser Mozilla Firefox, Google Chrome

Theory:

1. Introduction to JavaScript.

- JavaScript was initially created to “make webpages alive”.
- The programs in this language are called scripts. They can be written right in the HTML and execute automatically as the page loads.Scripts are provided and executed as a plain text. They don‘t need a special preparation or a compilation to run.
- In this aspect, JavaScript is very different from another language called Java.

- When JavaScript was created, it initially had another name: —LiveScript. But Java language was very popular at that time, so it was decided that positioning a new language as a —younger brother of Java would help.
- But as it evolved, JavaScript became a fully independent language, with its own specification called ECMAScript, and now it has no relation to Java at all.
- At present, JavaScript can execute not only in the browser, but also on the server, or actually on any device where there exists a special program called the JavaScript engine.
- The browser has an embedded engine, sometimes it‘s also called a —JavaScript virtual machine.

2. In-browser JavaScript

The modern JavaScript is a safe programming language. It does not provide low-level access to memory or CPU, because it was initially created for browsers which do not require it.
The capabilities greatly depend on the environment that runs JavaScript. For instance, Node.JS supports functions that allow JavaScript to read/write arbitrary files, perform network requests etc.
In-browser JavaScript can do everything related to web page manipulation, interaction with the user and the webserver.
For instance, in-browser JavaScript is able to:

- Add new HTML to the page, change the existing content, modify styles.
- React to user actions, run on mouse clicks, pointer movements, key presses.
- Send requests over the network to remote servers, download and upload files (so-called AJAX and COMET technologies).
- Get and set cookies, ask questions to the visitor, show messages.
- Remember the data on the client-side (local storage).

JavaScript‘s abilities in the browser are limited for the sake of the user‘s safety. The aim is to prevent an evil web page from accessing private information or harming the user‘s data.
The examples of such restrictions are:

- The limitation is for user‘s safety. A page from http://anysite.com which a user has opened must not be able to access another browser tab with the URL http://gmail.com and steal information from there.
- JavaScript can easily communicate over the net to the server where the current page came from. But its ability to receive data from other sites/domains is crippled.

- Though possible, it requires explicit agreement (expressed in HTTP headers) from the remote side.

3. Unique features of JS.

Following are the three unique features of JavaScript:

- Full integration with HTML/CSS.
- Simple things done simply.
- Supported by all major browsers and enabled by default.

These advantages make it the most widespread tool to create browser interfaces.

4. Limitations of JavaScript

We cannot treat JavaScript as a full-fledged programming language. It lacks the following important features −

- Client-side JavaScript does not allow the reading or writing of files. This has been kept for security reason.
- It cannot be used for networking applications because there is no such support available.
- It doesn't have any multithreading or multiprocessor capabilities.
- It is a lightweight, interpreted programming language that allows you to build interactivity into otherwise static HTML pages.

Design:

Login form validation using JavaScript

1.HTML code:

```
<html>
<body>
<form id="form1" runat="server">
<div class="container">
<div class="main">
<h2> Javascript Login Form Validation</h2>
<form id="form_id" method="post" name="myform">
<label> User Name :</label>
<input type="text" name="username" id="username" />
<br><br>
```

```
<label> Password :</label>
<input type="password" name="password" id="password" />
<input type="button" value="Login" id="submit" onclick="validate();" />
</form>
</div> </div> </form></html> </body>
```

2.JavaScript code:

```
<script language = "JavaScript">
function validate() {
var username = document.getElementById("username").value;
var password = document.getElementById("password").value;
if (username == null || username == "") {
alert("Please enter the username.");
return false;
}
if (password == null || password == "") {
alert("Please enter the password.");
return false;
}
alert('Login successful');
}
</script>
```

Test cases:

Sr.no	Test case	Expected output	Output
1.	Do not enter username.	Alert says “Please enter username”	Success
2.	Do not enter a password.	Alert says “Please enter the password”	Success
3.	Enter a password of length less than 6.	Alert says “Minimum password length should be 6.”	Success
4.	Enter a password that does not contain numerals	Alert says “Password should contain at least a numeral”	Success

Conclusion:

Thus, a login form has been validated using JavaScript upon completion of above stated assignment.