Web Basics - JavaScript

Lab Book

Document Revision History

Date	Revision No.	Author	Summary of Changes
12-May-2009	1.0	Pradnya Jagtap	Content Creation
05-Oct-2009	1.1	CLS Team	Review
24-Jun-2010	2.0	Anu Mitra	Refinements
03-May-2011	3.0	Karthik M/Anu Mitra	Integration Refinements
21-Apr-2015	4.0	Rathnajothi P	Revamp/Refinement
15-May-2015	5.0	Kavita D Arora	Integration Refinements

Table of Contents

Getting Started

Overview

These Lab book is a guided tour for Learning JavaScript. It contains solved examples and To Do assignments. Follow the steps provided in the solved examples and then work out the 'to do' Assignments given.

Setup Checklist for JavaScript

Here is what is expected on your machine in order for the lab to work.

Minimum System Requirements

- Hardware: Networked PCs with minimum 64 MB RAM and 60 MB HDD.
- Software: Window based Operating System having the latest version of Internet Explorer (IE) or Netscape Navigator installed.

Please ensure that the following is done:

- A text editor like Notepad, Eclipse Luna or Visual Studio 2008 is installed.

Instructions

- For coding standards refer Appendix – A.
- All Lab assignments should follow the coding standards.
- Create a directory by your name in drive <drive> for JavaScript assignments.
- In this directory, create subdirectory javascript_assgn.
- For each lab create directory as lab<lab number>.

Learning More (Bibliography if applicable)

- Beginning JavaScript by Paul Wilton
- JavaScript: The Definitive Guide by David Flanagan
- JavaScript Application Cookbook by Jerry Bradenbaugh

Lab 1:Working with Document Object Model(DOM)

Goals	Understand Window Object Dynamically create windows Handle window events
Time	90 minutes

1 .1: Window object

Create a **prob1.html** web page which has the following items as shown in the figure given below:

- a form that accepts window parameters width, height, title, left and top parameters from text field, and
- two buttons with the labels **New Window** and **Reset** to the web page

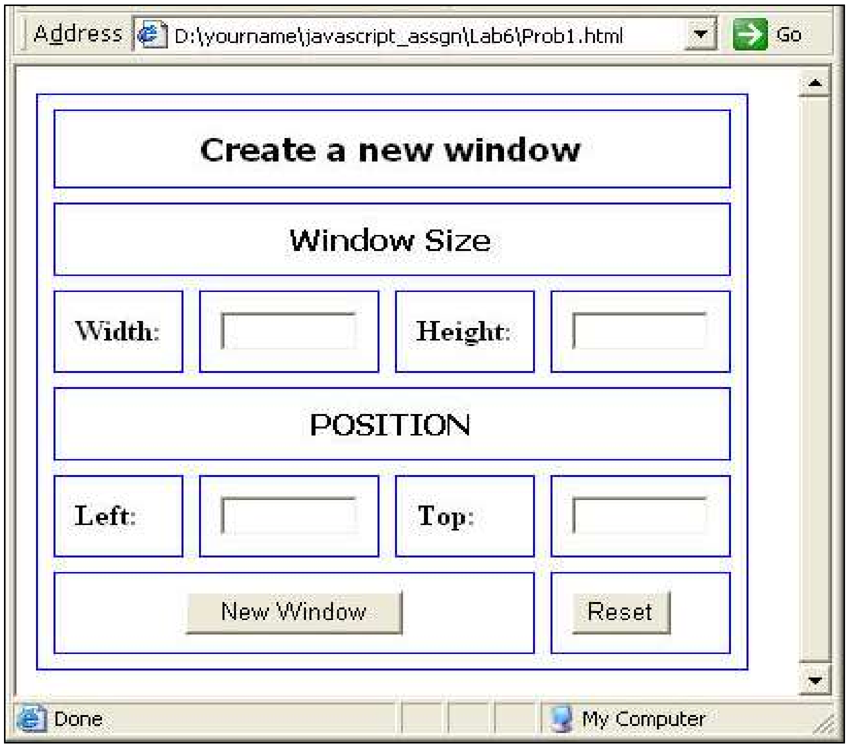
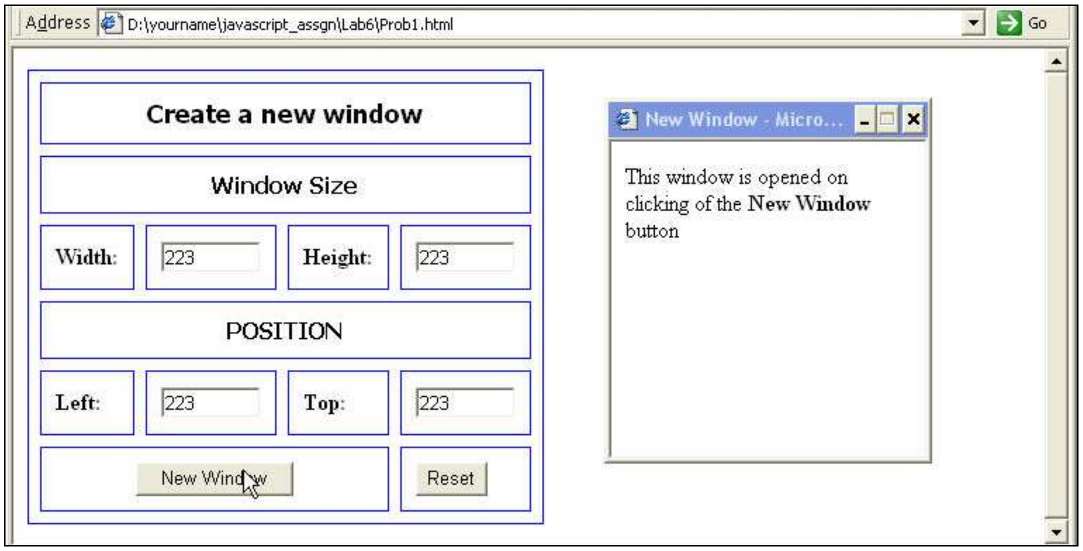


Figure 1: Interface to accept window coordinates

If **Reset** button is clicked, then clear all text fields. If **New Window** button is clicked, then open a new window with specifications entered in the text fields as shown in the figure given below.

Note: By default, the new window opens at the top left corner of the screen.



Create a new window			
Window Size			
Width:	223	Height:	223
POSITION			
Left:	223	Top:	223

This window is opened on clicking of the New Window button

Figure 2: Opening a window

Solution:

Step 1: Complete the following Code and save it as **Prob1.html.**

```
<html>
<head>
<title> window example </title>
</head>
<script>
function nwindow()
{
/*TODO:  get the height, width, left and top from the form object and pass the values to open
method of window along with the name of the html file to be opened in the new window.*/
}
</script>
<body >
<form id="frmlab">
<table border="1" cellspacing="8" cellpadding="10" bordercolor="blue">

// Create Table as shown in fig 6.2
</table>
```

```
</form>
</body>
</html>
```

Example 1: Lab 5: Prob1.html

Step 2: Open **prob1.html** page in the browser, and verify that you get the same output as required.

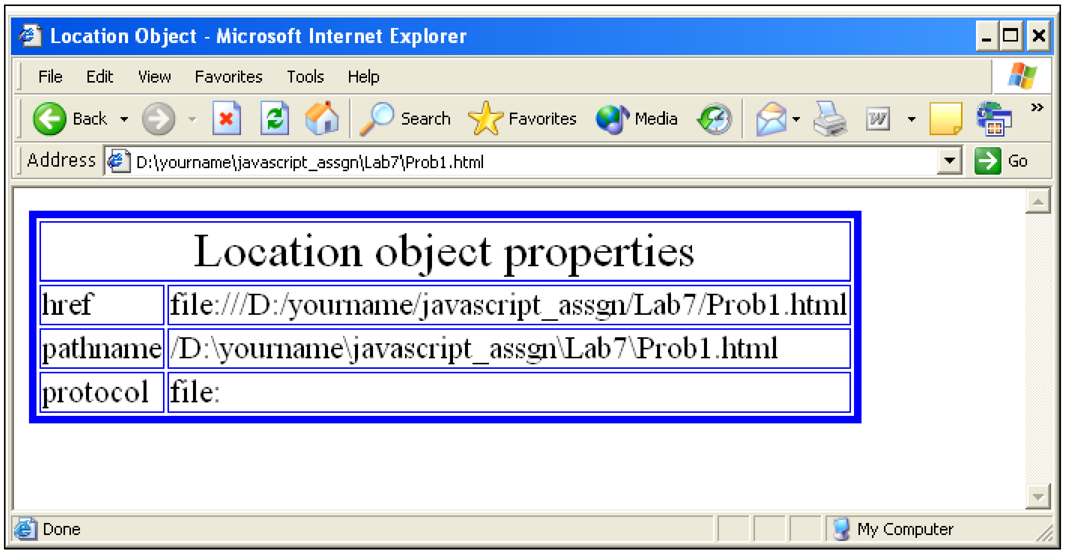
Step 3: Open **prob2.html** page in the browser, and verify that you get the same output as required.

Lab 2:Working with Location Object

Goals	Understand and use Location Object.
Time	20 minutes

2.1: Location Object

Create a web page which will display the properties **href**, **protocol**, and the **pathname** of the location object of your current file.



Location Object - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address D:\yourname\javascript_assgn\Lab7\Prob1.html
Go
Done
My Computer

Location object properties	
href	file:///D:/yourname/javascript_assgn/Lab7/Prob1.html
pathname	/D:\yourname\javascript_assgn\Lab7\Prob1.html
protocol	file:

Figure 3: Location Object Properties

Solution:

Step 1: Write the code and save it as **Prob1.html**.

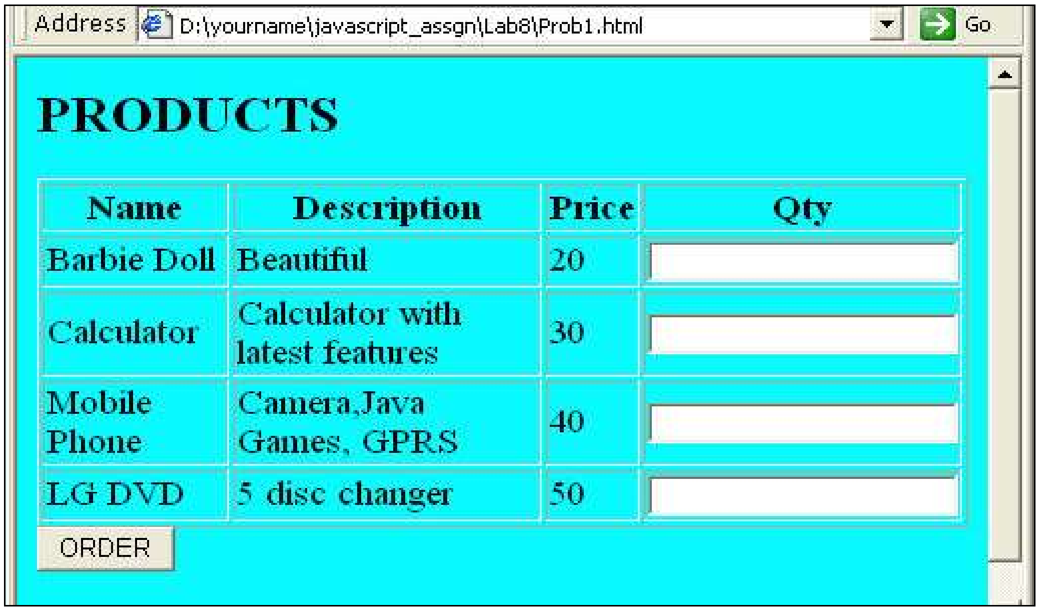
Step 2: Open **prob1.html** page in the browser, and verify that you get the same output as required.

Lab 3:Working with Document Object

Goals	Understand Document Object
Time	120 minutes

3.1: Working with Documents

Create a **prob1.html** web page which displays products available as shown in the following figure. The product details comprise Product Name, Product description, and its price.

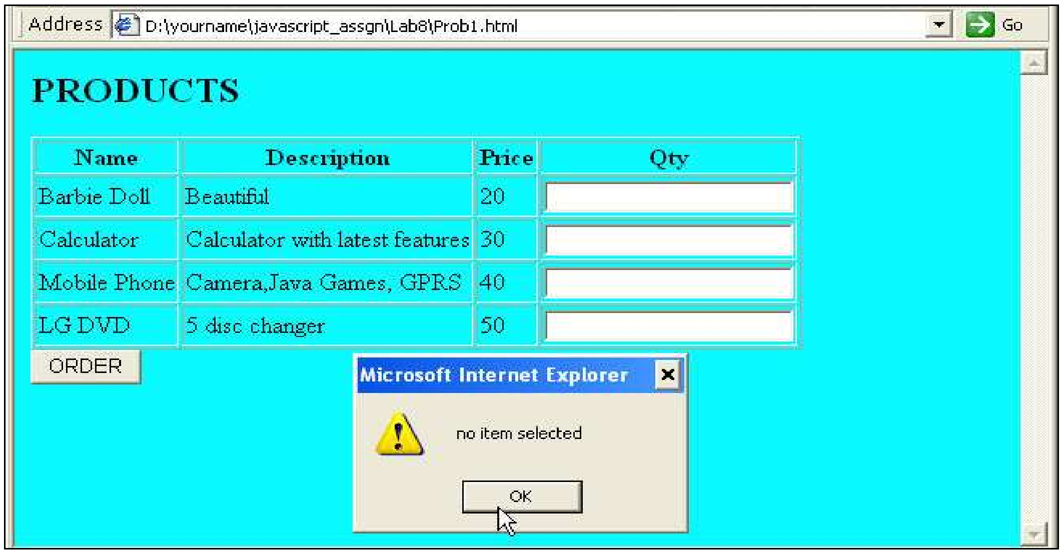


PRODUCTS

Name	Description	Price	Qty
Barbie Doll	Beautiful	20	
Calculator	Calculator with latest features	30	
Mobile Phone	Camera,Java Games, GPRS	40	
LG DVD	5 disc changer	50	

Figure 4: Displaying Products

Users can place orders specifying the quantity of each product. If the user does not enter quantity in any of the text fields, then an error message should be displayed as shown in the figure given below:



Address D:\yourname\javascript_assgn\Lab8\Prob1.html

PRODUCTS

Name	Description	Price	Qty
Barbie Doll	Beautiful	20	
Calculator	Calculator with latest features	30	
Mobile Phone	Camera,Java Games, GPRS	40	
LG DVD	5 disc changer	50	

ORDER

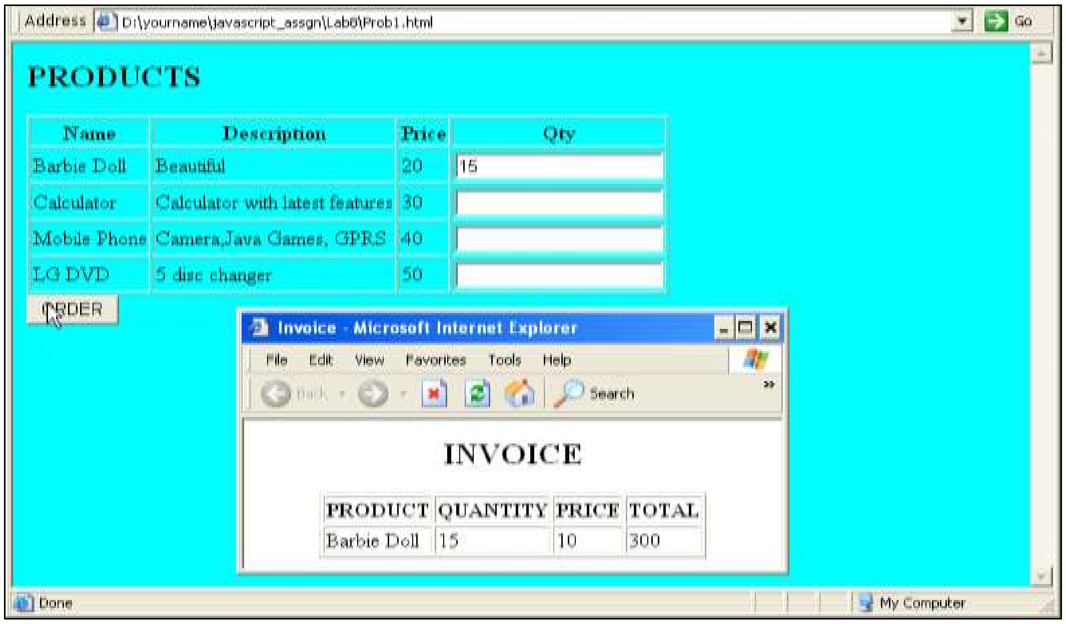
Microsoft Internet Explorer

no item selected

OK

Figure 5: Validating Products

When the user clicks the **Order** button, the invoice for the current products transaction showing the product name, quantity ordered, price and total amount is displayed in a new window as shown in the figure given below:



Address D:\yourname\javascript_assgn\Lab8\Prob1.html

PRODUCTS

Name	Description	Price	Qty
Barbie Doll	Beautiful	20	15
Calculator	Calculator with latest features	30	
Mobile Phone	Camera,Java Games, GPRS	40	
LG DVD	5 disc changer	50	

ORDER

Invoice - Microsoft Internet Explorer

INVOICE

PRODUCT	QUANTITY	PRICE	TOTAL
Barbie Doll	15	10	300

Done

My Computer

Figure 6: Displaying Invoice details in a new window

Solution:

Step 1: Write the code and save it as **Prob1.html**.

Step 2: Open **prob1.html** page in the browser, and verify that you get the same output as required.

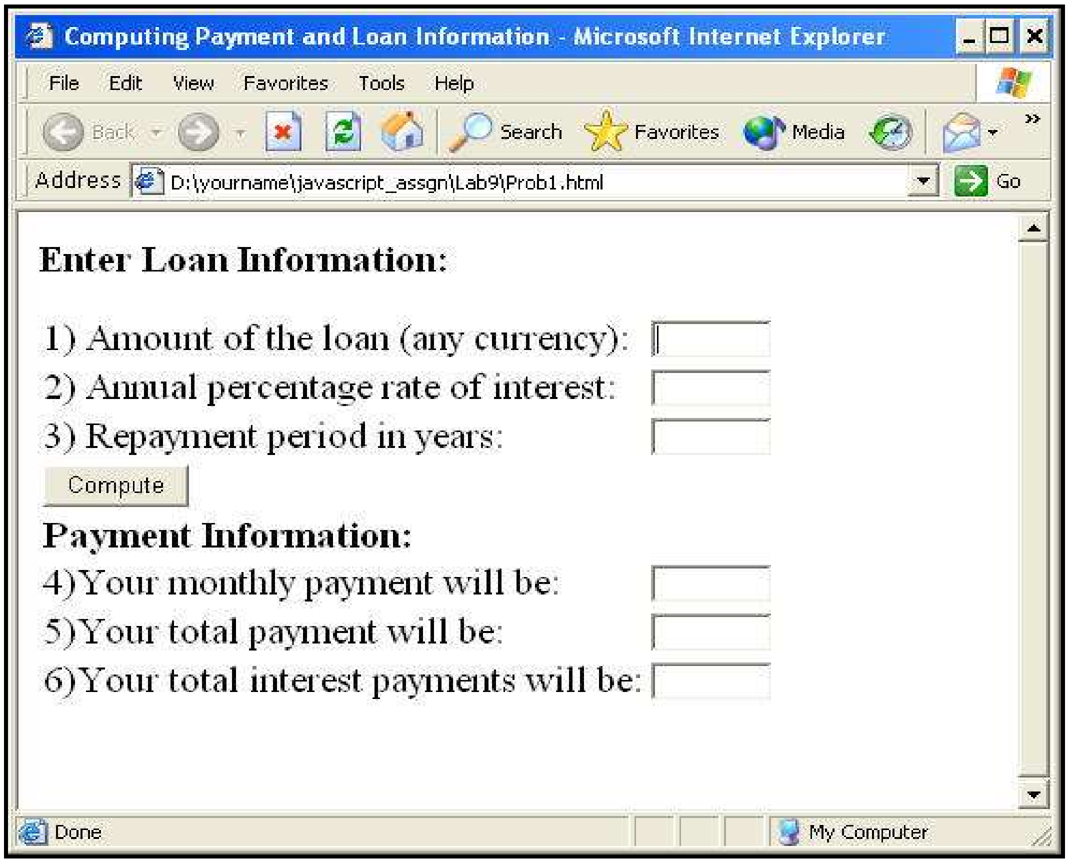
Lab 4:Working with Form Object

Goals	• Understand and use Form Object.
Time	90 minutes

4.1: Form Validation

Create a **prob1.html** web page, as shown below, and calculate **Payment Information** based on **Loan Information**. Validate **Loan information** textfields for numbers. The **Payment Information** textfields should be uneditable. The other constraints are as follows:

- Amount of Loan should not be more than 15 lakhs.
- Repayment period should be between 7 yrs to 15 yrs.


Computing Payment and Loan Information - Microsoft Internet Explorer
File
Edit
View
Favorites
Tools
Help
Back
Search
Favorites
Media
Address
D:\yourname\javascript_assgn\Lab9\Prob1.html
Go
Done
My Computer

Enter Loan Information:

1) Amount of the loan (any currency):

2) Annual percentage rate of interest:

3) Repayment period in years:

Compute

Payment Information:

4)Your monthly payment will be:

5)Your total payment will be:

6)Your total interest payments will be:

Figure 7: Validating Form elements

If the repayment period is not between 7 and 15, then an error message should be displayed next to this control.

Similar kind of error message should be displayed if the amount of loan exceeds 15 lakh.

```
In the function calculatePayment()
/*
TODO:
calculate the monthly payment, total payment, total interest payment on click of the button with
label "compute"
*/
```

Example 2: Lab 8: Prob1.html

Open **prob1.html** page in the browser, and verify that you get the same output as required.

4.2 Validate Field

Create a **prob2.html** page as shown in the below figure.

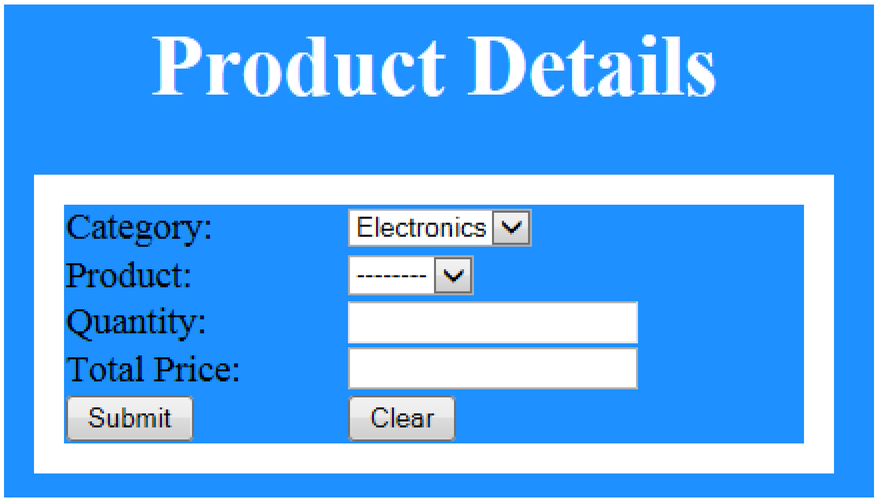


Figure 20: Lab 8.2 Product Details

Data should be prepopulated in category list box (Electronics, Grocery). Based on selection of category, product list need to be populated automatically with values as given in the below table. Also Total price need to be calculated for the entered quantity as per the data in the below table. Total price field should be non-editable field.

Category	Product	Price per quantity in Rupees
Electronics	Television	20000
	Laptop	30000
	Phone	10000
Grocery	Soap	40
	Powder	90

While clicking on submit button, if all the text fields contains valid values then display the filled details in a popup window.

Lab 5: Regular Expressions in JavaScript

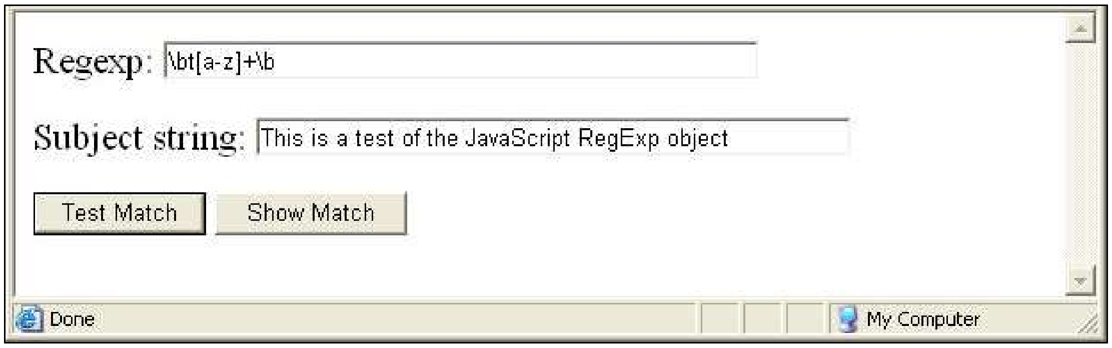
Goals	• Understand Regular Expression object • Use Regular Expression object for validating
Time	90 minutes

5.1: Regular Expression

Create a **prob1.html** page which has two text fields – one for the Regular Expression search pattern and the other for the string in which the pattern has to be checked.

The form has two buttons one **"Test Match"** and the other **"Show Match"**.

Test Match will test the regular expression against the string. **Show Match** will show the matching part of the string.



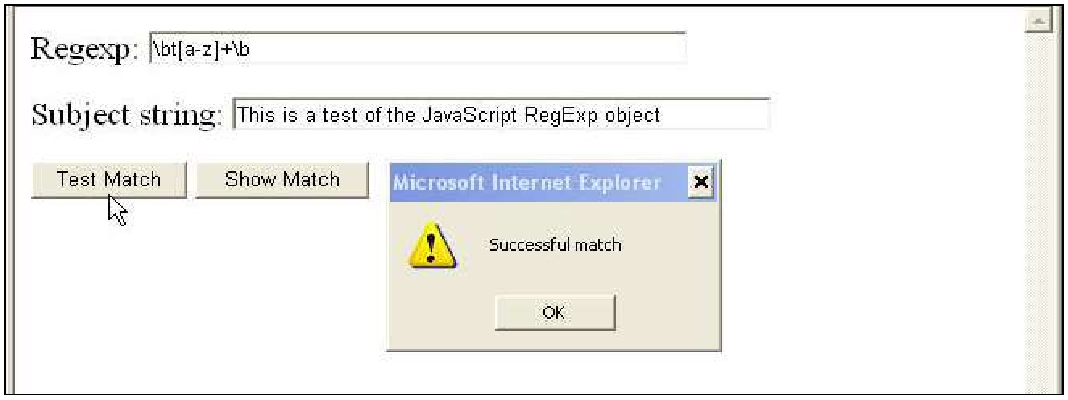
Regexp: \bt[a-z]+\b

Subject string: This is a test of the JavaScript RegExp object

Figure 8: Regular Expression Pattern

Type the regular expression in the first textbox and the string in the second text box. Click the **"Test Match"** button.

If the text in the second text box matches the Regular Expression in the first text box, then it should display a message as shown below.

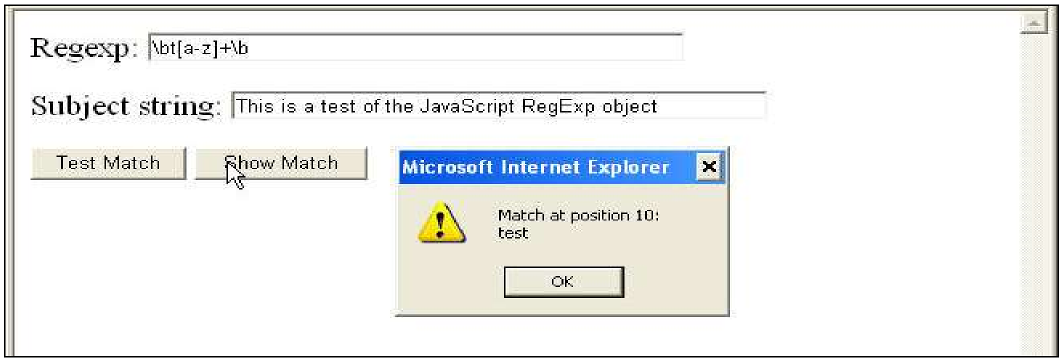

Regexp: \bt[a-z]+\b

Subject string: This is a test of the JavaScript RegExp object

Successful match

Figure 9: Validating Regular Expression Pattern

If you click the **"Show Match"** button, then it should display the match as shown in the figure given below:


Regexp: \bt[a-z]+\b

Subject string: This is a test of the JavaScript RegExp object

Match at position 10:
test

Figure 10: Displaying text matching Regular Expression Pattern

Solution:

Step 1: Write the Code and save it as **Prob1.html.**

Some Tips

In the function demoMatchClick()

/*

TODO:

define a variable re which is the regular expression object, to which the regular expression pattern is passed as an argument. define a variable str which holds the string from subject field match the "str" with "re" using the match method of the string object and display appropriate messages

*/

```
In the function demoShowMatchClick()
/*
TODO:
Store the regular expression pattern in a variable "re" and invoke exec method of regular expression object which takes the matching string as argument and returns the index of the matching string found
*/
```

Example 3: Prob1.html

Step 2: Open **prob1.html** page in the browser, and verify that you get the same output as required.

5.2: Form Validation using Regular Expression

Create a **prob3.html** page as shown in the below figure. Use CSS for designing page The page should be submitted on clicking the **Submit** button when all the form fields are properly validated.

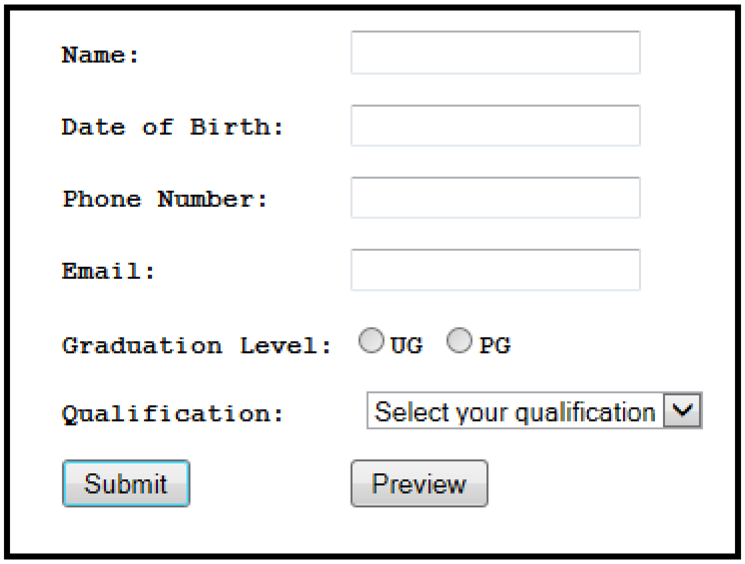


Figure 11: Form Validation using Regular Expression

1. None of the fields should be empty (Use HTML 5 attributes)
2. Name field should be between 3 to 10 characters (Use HTML 5 attributes)
3. Date of Birth format can be either (DD/MM/YY or DD/MM/YYYY) (Use HTML 5 date control)
4. Phone Number should be in xxx-xxxx-xxxx format (Use HTML 5 attributes)
5. Email ID should be valid. (Use HTML 5 attributes)

6. Based on graduation level selected, qualification need to be populated automatically. For an example, if graduation level selected is UG, then qualification should be B.Sc, B.A, B.Com, etc... If graduation level selected is PG, then qualification should be M.A, M.Tech, MCA, MBA, etc… (Call function on onChange event)

7. Calculate age of the person and display all the details in a new popup window when “Preview” button is clicked. Details should be printed in the specified format as given below:

    Name:
    Age:
    Phone Number:
    Email:
    Graduation Level:
    Qualification:

Display the appropriate error message when the condition fails.

Appendices

Appendix A: JavaScript Standards

1. Naming conventions for variables in JavaScript:

- Variables must begin with prefix indicating the type of the variable.
    - All integer must start with "int".
    - All floating data types must start with "flt".
    - All string must start with "str".
    - All object name must start with "obj".
    - All Boolean variables must start with "bln".
    - All variables that store date must start with "dt".
    - All constants must be in upper case with different words separated by underscore (_).
    - All array variables must start with "arr".
    - Apart from these guidelines all variable name must be sensible enough, so that it's purpose can be identified from it's name.

Type	Example
String	strStringName
Boolean	blnPresent
Array	arrArrayName
Object	objObjectName
Date	dtDateName
Integer	intValueInteger
Float	fltValueFloat
Constants	STRING_CONSTANT ARRAY_CONSTANT NUMERIC_CONSTANT

- All HTML elements must be prefixed with appropriate types.
  - TextBox "txt”
  - Image "img"
  - Image map Area “img”
  - option button "opt"
  - CheckBox "chk"
  - DropDown List "lst"
  - Form Name "frm"
  - Buttons "btn"
  - All div tags "div"
  - All class names must start with "cls"
  - All user-defined objects must start with "u"
  - First letter of each variable/function name must be in upper case. Rest all letters must be in lowercase.
  - Use of underscore and digits for naming variables must be avoided.

Tag	Example
Div	divContent
Class	clsInterest
Form	frmContainer
Image	imgMapThis
Button	btnOk
TextBox	txtInterestRate
CheckBox	chkAllow
Option Button	rdbRate
DropDownList	lstState

It must be noted that this naming style does not apply to HTML elements. However, when these elements are accessed in the JavaScript functions, these naming conventions must be followed. This document describes coding convention only for JavaScript.

2. Commenting

- Comments related to a particular line of code should be on the same line after the statement gets over.

```
If (dtToday == “15/07/99”) {            // Is date birthdate?
  alert (“Happy Birthday”);             // Give birthday message
} else {
  alert (“Happy Day”);                                  // Give standard message
}
```

- Over all commenting should consist of two parts – Comment header and Comment footer. Comment header must precede the block of code and Comment footer must follow the block of code.

```
//Function Name:    calculateInterest
//Description:      This function calculates the interest. It accepts the initial investment and
//                  period for which the amount is invested. Rate of interest is fixed.
//                  Formula is
//                  fltInterest = fltAmount * fltPeriod               *fltRATE/100
//                  Dhrumil Dalal
//                  15/07/1999
//Author:           fltAmount – indicated the amount invested
//Start Date:       fltPeriod – Indicates the period of investment
//Input             Calculated interest
Parameters:
//
//Return Value:
Function calculateInterest(fltAmount,fltPeriod){
}


//End of function for calculating the rate of interest
```

3. Documentation

- All variables used in the function must be declared in brief.
- Only one variable declaration per line.
- Describe each variable on the same line and description should not be more than one line.
- All functions must be preceded by comments. Comments must describe the following:
    - Input parameters.
    - Return value.
    - Function logic in brief.

  - Starting date.
  - Name of the author.
  - Revision history.
- After the end of function, there must be block of comment indicating the end of function.

```
//Function Name:              calculateInterest
//Description:                This function calculates the interest. It accepts the initial
//                            investment and period for which the amount is invested.
//                            Rate of interest is fixed. Formula is
//                            fltInterest = fltAmount * fltPeriod          *fltRATE/100
//                            Dhrumil Dalal
//                            15/07/1999
//Author:                     fltAmount – indicated the amount invested
//Start Date:                 fltPeriod – Indicates the period of investment
//Input Parameters:           Calculated interest
//
//Return Value:
Function calculateInterest(fltAmount,fltPeriod){
Var fltRATE = 12.5; // fixed rate of interest
Var fltInterest;  // The variable to store calculated interest
fltInterest = fltAmount * fltPeriod * fltRATE/100 ;
return fltInterest;
}

//End of function for calculating the rate of interest
```

4. 4: Coding Styles

- For statements which may have block of code enclosed in {}, the opening brace "{" must immediately follow the statement and the closing brace "}"must be below the statement. That is to say, the closing brace and first letter of the statement must be same in the column.

```
If (condition) {
  ...
} else {
  if (condition) {
    ...
  } else {
    ...
  }
}

for(intCounter=0; intCounter <= 5; intcounter++){
  //Perform calculation.
  //Display Result}
```

- All statements within corresponding opening and closing brace must be indented. Indentations must be in odd columns.

Column no

```
123456789……………………………………………………………………………
if (condition) {
  ...
} else {
  if (condition) {
    ...
  } else {
    ...
  }
}
```

- Also the code should not extend past the 80th column so that it is required to scroll to the right or left to edit a particular line. In the case of strings which do not fit on one line, it is recommended that temporary variables be used with the string concatenation operator (+=) to construct strings of longer lengths. The following example illustrates this:

```
Column no
123456789………………………………………………………………………80
strMessage  = “Demonstrating the use of …    “;
strMessage  += “prepared on 15-07-1999”;
```

Appendix B: Coding Best Practices

JavaScript Best Practice

The following demonstrate the best practices that should be followed while writing JavaScript code.

1. Inline JavaScript source code

Any JavaScript code that does not write out to the document should be placed within the head of the document.

```
<HTML>
<HEAD>
<SCRIPT>
<!--
function functionName() {
   alert(text);
}

var text = 'Hello World';
//-->
</SCRIPT>
</HEAD>
```

Example 4: Sample Code

This ensures that the browser has loaded the JavaScript function definitions before it is required. It also makes it slightly easier to maintain the JavaScript code if it can always be found in the head of the document.

2. JavaScript Links

Avoid using the **javascript:** protocol as a default URL within a link.

If JavaScript is disabled, then the link will not work. Do not use the following:

```
<SCRIPT>
<!--
function functionName() {
   alert('Hello world');
}
//--></SCRIPT>
<A HREF="javascript:functionName()">text link</A>
```

Example 5: Sample Code

Instead, use JavaScript itself to override the **href** property of the link:

```
<SCRIPT>
<!--
function functionName() {
   alert('Hello world');
}
//-->
</SCRIPT>

<A HREF="default.htm" onClick="this.href='javascript:functionName()'">text link</A>
```

Example 6: Sample Code

3. Avoid Using Void

All browsers do not support the **void** function. Create your own **void** function.
The in built **void()** function is supported since JavaScript 1.1. Therefore it is best to create your own void function rather than rely on JavaScript 1.1 being available.

```
<SCRIPT>
<!--
function myVoid() { } // create a void function
//-->
</SCRIPT>

<A HREF="#" onClick="this.href='javascript:myVoid()'">non functional text link</A>
```

Example 7: Sample Code

4. JavaScript Performance

Avoid writing output multiple times to the document, concatenate the data, and then write all in one go.
With the introduction of Netscape Navigator 4, the rendering of JavaScript generated HTML slowed down considerably.
The following writes the HTML output to the document in one go:

```
<SCRIPT>
<!--
var output = '<P>';
output += 'Last modified: ';
output += document.lastModified;
output += '<\/P>'
document.write(output);
//-->
</SCRIPT>
```

Example 8: Sample Code

5. Select Form Fields

Use the Netscape method to correctly navigate select field properties.
The following technique works in Microsoft Internet Explorer. However it should be avoided.

```
<SCRIPT>
<!--
var property = document.formName.selectName.propertyName
//-->
</SCRIPT>
```

Example 9: Sample Code

Whereas the following will work correctly in all browsers:

```
<SCRIPT>
<!--
var property =
document.formName.selectName.options[document.formName.selectName.options.selectedIndex]
.propertyName
//-->
</SCRIPT>
```

Example 10: Sample Code

6. Changing Location

Do not use the following:

```
<SCRIPT>
<!--
location = 'page.htm';
//-->
</SCRIPT>
```

Example 118: Sample Code

The later approach is confusing as it is not clear whether you are changing the location property of the “window” or the “document object”.

Changing the location using the document is deprecated and causes problems on later browsers.

Use the following:

```
<SCRIPT>
<!--
window.location.href = 'page.htm';
//-->
</SCRIPT>
```

Example 19: Sample Code

7. Opening Windows

While opening a new popup window using JavaScript, there are several points to bear in mind.

To be able to control the popup window from the **opener** window, always retain the returned reference from the window's open method:

```
<SCRIPT>
<!--
var windowHandle = window.open('page.htm','windowName','width=600,height=320');
//-->
</SCRIPT>
```

Example 12: Sample Code

To avoid errors while referring to the **opener** window from the **popup** window, always check for the in-built browser support for the **opener** property. If necessary, provide your own:

```
<SCRIPT>
<!--
var windowHandle = window.open('page.htm','windowName','width=600,height=320');
if (!windowHandle.opener)
   windowHandle.opener = self;
//-->
</SCRIPT>
```

Example 13: Sample Code

While updating the contents of a newly opened window, give the browser time to open the window and to load the initial contents:

```
<SCRIPT>
<!--
function update() {
   windowHandle.document.open();
   windowHandle.document.write('<H1>Hello World<\/H1>');
   windowHandle.document.close();
}

var windowHandle = window.open('page.htm','windowName','width=600,height=320');
if (!windowHandle.opener)
   windowHandle.opener = self;
setTimeout('update()',2000);
//-->
</SCRIPT>
```

Example 14: Sample Code

8. JavaScript Entities

JavaScript Entities are only supported by Netscape Navigator. Avoid their use.
The following will cause errors in other browsers:

```
<HR WIDTH="&{barWidth};%">
```

Example 15: Sample Code

Instead the following can be used.

```
<SCRIPT>
<!--
document.write('<HR WIDTH="' + barWidth + '%">');
//-->
</script>
```

Example 16: Sample Code

Appendix C: Table of Figures

Appendix D: Table of Examples