Unit 5

Regular Expression, Rollover and Frames

Marks: 14 (R-2, U-6, A-6)

Course Outcome: Create interactive web pages using regular expressions for validations.

Unit Outcome:
1) Compose relevant regular expression for the given character pattern search.
2) Develop JavaScript to implement validations using the given regular expression.
3) Create frame based on the given problem.
4) Create window object a per the given problem.
5) Develop JavaScript for creating rollover effect for the given situation.

Topics and Sub-topics:

5.1 Regular Expression
- 5.1.1 What is Regular Expression:
- 5.1.2 Language of Regular Expression
- 5.1.3 Finding Non-Matching Character
- 5.1.4 Entering a Range of Character
- 5.1.5 Matching Digits and Non-digits
- 5.1.6 Matching Punctuation and Symbols
- 5.1.7 Matching Words
- 5.1.8 Replace Text using a Regular Expression
- 5.1.9 Returned the Match Character
- 5.1.10 Regular Expression Object Properties

5.2 Frames
- 5.2.1 Create a frame
- 5.2.2 Invisible borders of frame
- 5.2.3 Calling a child window
- 5.2.4 Changing a content and focus of a child window
- 5.2.5 Writing to a child window
- 5.2.6 Accessing elements of another child window.

5.3 Rollover
- 5.3.1Creating rollover
- 5.3.2 Text Rollover
- 5.3.3 multiple actions for rollover

5.3.4 more efficient rollover

5.1 Regular Expression

A regular expression is an object that describes a pattern of characters.

The JavaScript **RegExp** class represents regular expressions, and both String and **RegExp** define methods that use regular expressions to perform powerful pattern-matching and search-and-replace functions on text.
A regular expression is very similar to a mathematical expression, except a regular expression tells the browser how to manipulate text rather than numbers by using special symbols as operators.

A **Regular Expression** is a sequence of characters that constructs a search pattern. When you search for data in a text, you can use this search pattern to describe what you are looking for.

```
/^[Reg]ular[Ex]pression$/
```

A regular expression can be a **single character** or a more complicated pattern. It can be used for any type of text search and text replace operations. A Regex pattern consist of simple characters, such as /abc/, or a combination of simple and special characters, such as **/ab*c/** or **/example(d+).d*/.**

In JavaScript, a regular expression is an object that describes a pattern of characters. The JavaScript **RegExp** class represents regular expressions, and both String and RegExp define methods. It uses regular expressions to perform **pattern-matching** and **search-and-replace** functions on text.

Syntax:
A regular expression is defined with the **RegExp ()** constructor as:

```
var pattern = new RegExp(pattern, attributes);
```

or simply

```
var pattern = /pattern/attributes;
```

Here,

- **Pattern –** A string that specifies the pattern of the regular expression or another regular expression.
- **Attributes –** An optional string containing attributes that specify global, case-insensitive, and multi-line matches.

5.1.1 Language of Regular Expression

Brackets

Brackets ([ ]) have a special meaning when used in the context of regular expressions.

They are used to find a range of characters.

Expression	Description
[abc]	Find any character between the brackets
[^abc]	Find any character NOT between the brackets
[0-9][A-Z][a-z][p-r][pqr]	Find any character between the brackets (any digit)
[^0-9]	Find any character NOT between the brackets (any non-digit)
(x|y)	Find any of the alternatives specified

The [abc] expression is used to find any character between the brackets.

The characters inside the brackets can be any characters or span of characters:

- [abcde..] - Any character between the brackets
- [A-Z] - Any character from uppercase A to uppercase Z
- [a-z] - Any character from lowercase a to lowercase z
- [A-z]- Any character from uppercase A to lowercase z

Syntax

```
new RegExp("[abc]")
or simply:
/[abc]/
```

Syntax with modifiers

```
new RegExp("[abc]", "g")
or simply:
/[abc]/g
```

Code: Do a global search for the characters "i" and "s" in a string:

```
<html>
<body>
<p>Click the button to do a global search for the characters "i" and "s" in a string.</p>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
function myFunction()
{
  var str = "Do you know if this is all there is?";
  var patt1 = /[is]/gi;
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML = result;
}
</script>
</body>
</html>
```

Output:

Click the button to do a global search for the characters "i" and "s" in a string.

Try it

i,i,s,i,s,i,s

5.1.2 Finding non-matching Characters

The [^abc] expression is used to find any character NOT between the brackets.
The characters inside the brackets can be any characters or span of characters:

- ✔ [abcde..] - Any character between the brackets
- ✔ [A-Z] - Any character from uppercase A to uppercase Z
- ✔ [a-z] - Any character from lowercase a to lowercase z
- ✔ [A-z]- Any character from uppercase A to lowercase z

- Sometimes a JavaScript application prohibits certain characters from appearing within text entered into a form, such as a hyphen (-); otherwise, the character might inhibit processing of the form by the CGI program running on the web server.
- You can direct the browser to search for illegal character(s) by specifying the illegal character(s) within brackets and by placing the caret (^) as the first character in the bracket.
- Let's see how this works in the following example:
  /[^\-]/
- In this case, the browser is asked to determine whether the text does not contain the hyphen.
- The caret asks the browser to determine whether the following character(s) do not appear in the text.
- To find the hyphen in text, you need to escape the hyphen with the backslash, like so \-.
- Suppose you wrote the following regular expression and the browser didn't find the hyphen in the text. The browser responds with a false—this is because you are telling the browser to determine whether the hyphen appears in the text. If the hyphen appears, the browser would respond with a true.

/[\-]/

- However, by placing a caret in the regular expression, as shown next, the browser responds with a true if the hyphen is not found in the text. This is because you are telling the browser to determine whether the hyphen does not appear in the text.
  /[^\-]/

Syntax

```
new RegExp("[^xyz]")
or simply:
/[^xyz]/
```

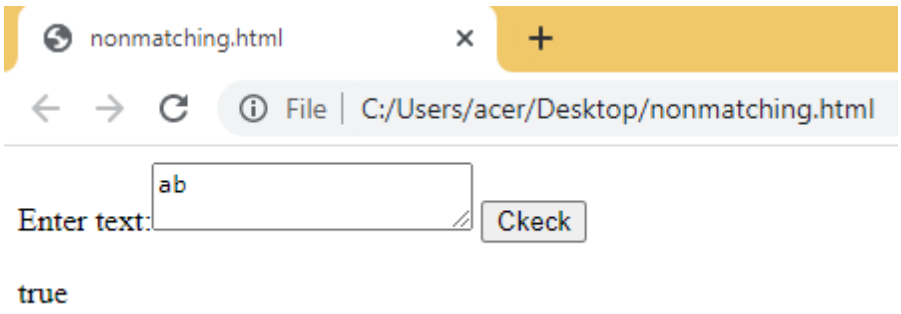
Syntax with modifiers

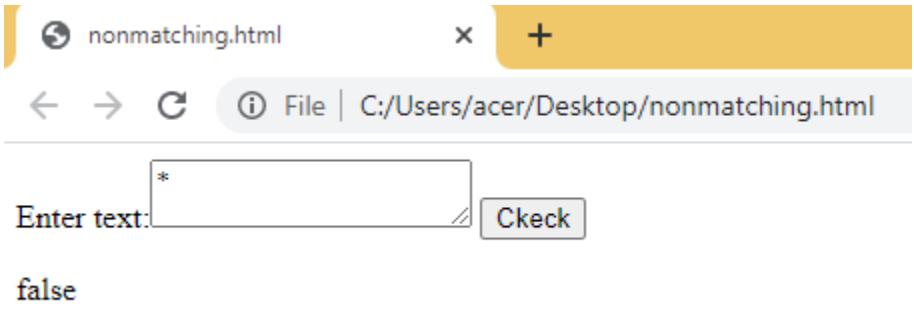
```
new RegExp("[^xyz]", "g")
or simply:
/\[^xyz]/g
```

Example:

```
<html> <script>
function check()
{
var exp=/[^\*]/;
var res=exp.test(document.getElementById("txt1").value);
document.getElementById("demo1").innerHTML=res;
}
</script>
<body>
Enter text:<textarea id="txt1"></textarea>
<input type="button" onclick="check()" value="Ckeck">
<p id="demo1"></p>
</body>
</html>
```

Output:





Code: Do a global search for characters that are NOT "i" and "s" in a string:

```
<html>
<body>
<p>Click the button to do a global search for characters that are NOT "i" and "s" in a string.</p>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
function myFunction() {
  var str = "Do you know if this is all there is?";
  var patt1 = /[^is]/gi;
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML = result;
}
```

```
</script>
</body>
</html>
```

Output:

Click the button to do a global search for characters that are NOT "i" and "s" in a string.

Try it

D,o, ,y,o,u, ,k,n,o,w, ,f, ,t,h, , ,a,l,l, ,t,h,e,r,e, ,?

The [^0-9] expression is used to find any character that is NOT a digit.
The digits inside the brackets can be any numbers or span of numbers from 0 to 9.

Syntax

```
new RegExp("[^0-9]")
or simply:
/[^0-9]/
```

Syntax with modifiers

```
new RegExp("[^0-9]", "g")
or simply:
/\[^0-9]/g
```

Code: Do a global search for numbers that are NOT "1" and “s” in a string.

```
<html>
<body>
<p>Click the button to do a global search for numbers that are NOT "1" and "s"in a string.</p>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
function myFunction()
```

```
{
  var str = "12121212This is JavaScript";
  var patt1 = /[^1s]/gi;
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML = result;
}
</script>
</body>
</html>
```

Output:

Click the button to do a global search for numbers that are NOT "1" and "s"in a string.

Try it

2,2,2,2,T,h,i, ,i, ,J,a,v,a,c,r,i,p,t

5.1.3 Entering a range of characters

The [0-9] expression is used to find any character between the brackets.
The digits inside the brackets can be any numbers or span of numbers from 0 to 9.
Note: can used for alphabets [a-z] and [A-Z] or [p-r] or [P-R] or [pqr] or [PQR]

Syntax

```
new RegExp("[0-9]")
or simply:
/[0-9]/
```

Syntax with modifiers

```
new RegExp("[0-9]", "g")
or simply:
/\[0-9]/g
```

Code: Do a global search for the number "1" and “s” in a string:

```
<html>
<body>
```

```
<p>Click the button to do a global search for the number "1" and “s” in a string.</p>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
function myFunction() {
  var str = "12121212This is Javascript";
  var patt1 = /[1s]/g;
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML = result;
}
</script>
</body>
</html>
```

Output:

Click the button to do a global search for the number "1" and "s" in a string.

Try it

1,1,1,1,s,s,s

The (x|y) expression is used to find any of the alternatives specified.
The alternatives can be of any characters.
Syntax:

```
new RegExp("(x|y)")
or simply:
/(x|y)/
```

Syntax with modifiers:

```
new RegExp("(x|y)", "g")
or simply:
/(x|y)/g
```

Code: Do a global search to find any of the specified alternatives (0|5|7):

```
<html>  <body>
<p>Click the button to do a global search for any of the specified alternatives (0|5|7).
</p>   <button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
function myFunction()
{
  var str = "01234567890123456789";
  var patt1 = /(0|5|7)/g;
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML = result;
}
</script>   </body>   </html>
```

Output:

Click the button to do a global search for any of the specified alternatives (0|5|7).

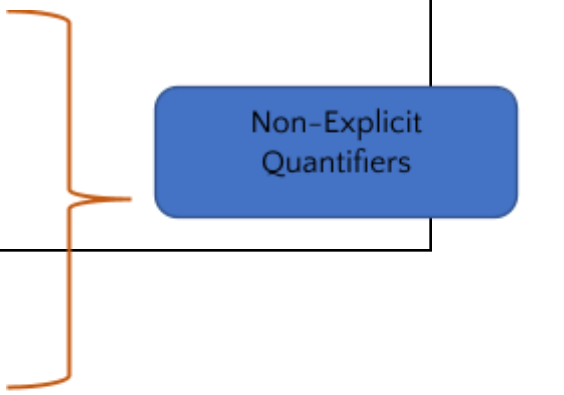
Try it

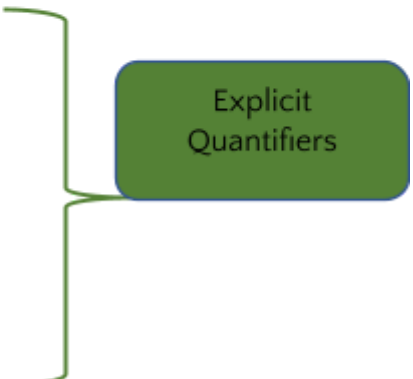
0,5,7,0,5,7

Quantifiers

The frequency or position of bracketed character sequences and single characters can be denoted by a special character. Each special character has a specific connotation.
The +, *, ?, and $ flags all follow a character sequence.

Sr.No.	Expression & Description
1	**p+** It matches any string containing one or more p's.





2	**p*** It matches any string containing zero or more p's.
3	**p?** It matches any string containing at most one p.(zero or one occurrences)
4	**p{N}** It matches any string containing a sequence of **N** p's
5	**p{2,3}** It matches any string containing a sequence of two or three p's.
6	**p{2, }** It matches any string containing a sequence of at least two p's.
7	**p$** It matches any string with p at the end of it.
8	**^p** It matches any string with p at the beginning of it.

Code: The n+ quantifier matches any string that contains at least one n.

```
<html>
<body>
<p>Click the button to do a global search for at least one "o" in a string.</p>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
function myFunction()
 {
  var str = "Hellooo World! Hello Vidyalanakr School!";
```

```
  var patt1 = /o+/g;
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML = result;
}
</script>
</body>
</html>
```

Output:

Click the button to do a global search for an "l", followed by zero or more "o" characters.

Try it

ooo,o,o,oo

Code: The n* quantifier matches any string that contains zero or more occurrences of n.

```
<html>
<body>

<p>Click the button to do a global search for at least one "o" in a string.</p>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
function myFunction() {
  var str = "Hellooo World! Hello Vidyalanakr School!";
  var patt1 = /o+/g;
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML = result;
}
</script>
</body>
```

```
</html>
```

Output:

Click the button to do a global search for an "1", followed by zero or more "o" characters.

Try it

1,1ooo,1,1,1o,1,1

Code: The n? quantifier matches any string that contains zero or one occurrences of n.

```
<html>
<body>
<p>Click the button to do a global search for a "1", followed by zero or one "0" characters.</p>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>

<script>
function myFunction() {
  var str = "1, 100 or 1000?";
  var patt1 = /10?/g;
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML = result;
}
</script>
</body>
</html>
```

Output:

Click the button to do a global search for a "1", followed by zero or one "0" characters.

Try it

1,10,10

Code:The n{X,Y} quantifier matches any string that contains a sequence of X to Y n's.
X and Y must be a number.

```
<html>
<body>
<p>Click the button to global search for a substring that contains a sequence of three to four digits.</p>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
function myFunction()
 {
  var str = "100, 1000 or 10000?";
  var patt1 = /\d{3,4}/g;
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML = result;
}
</script>  </body> </html>
```

Output:

Click the button to global search for a substring that contains a sequence of three to four digits.

Try it

100,1000,1000

Metacharacters
A metacharacter is simply an alphabetical character preceded by a backslash that acts to give the combination a special meaning.

For instance, you can search for a large sum of money using the '\d' metacharacter: **/([\d]+)000/**, Here **\d** will search for any string of numerical character.

The following table lists a set of metacharacters which can be used in PERL Style Regular Expressions.

Sr.No.	Character & Description
1	**. (dot)** a single character
2	**\s** a whitespace character (space, tab, newline)
3	**\S** non-whitespace character
4	**\d** a digit (0-9)
5	**\D** a non-digit
6	**\w** a word character (a-z, A-Z, 0-9, _)
7	**\W** a non-word character
8	**[\b]** a literal backspace (special case).
9	**[aeiou]**

	matches a single character in the given set
10	**[^aeiou]** matches a single character outside the given set
11	**(foo|bar|baz)** matches any of the alternatives specified

5.1.4 Matching Digits and Nondigits

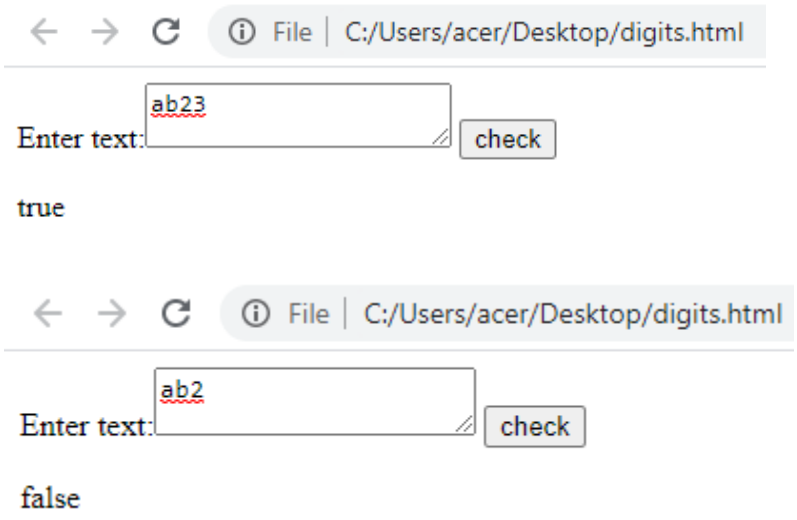
- You can have the browser check to see whether the text has digits or non-digits by writing a regular expression. The regular expression must contain either \d or \D, depending on whether you want the browser to search the text for digits (\d) or nondigits (\D).
- The \d symbol, tells the browser to determine whether the text contains digits. The browser returns a true if at least one digit appears in the text.
- You'd use this regular expression to determine whether a first name has any digits, The \D symbol is used to tell the browser to search for any nondigit in the text.
- The browser returns a true if a nondigit is found. This is the regular expression you would use to validate a telephone number, assuming the userwas asked to enter digits only. If the browser finds a nondigit, the telephone number is invalid and you can notify the user who entered the information into the form.

Example: Program to check for any two-consecutive digit in input.

```
<html>
<script>
function check()
{
var exp=/\d\d/;
var str=exp.test(document.getElementById("txt").value);
document.getElementById("demo1").innerHTML=str;
}
```

```
</script>
<body>
Enter text:<textarea id="txt"></textarea>
<input type="button" onclick="check()" value="check">
<p id="demo1"></p>
</body>
</html>
```

Output:



Following table shows the test cases on the above program.

Pattern	Code	Input	Output
/\d\d/	var exp=/\d\d/; var output=exp.test(input);	A23	True
		A2	False
		23	True

		23A	True
		2	False
		A1B3	False
/\d{4}	var exp=/\d{4} var output=exp.test(Input);	A1B2C3D4	False
		A1B2C3D	False
		A1b2c3456	True

Code: The \d metacharacter is used to find a digit from 0-9.

```
<html>
<body>
<p>Click the button to do a global search for digits in a string.</p>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
function myFunction()
 {
  var str = "Give 100%!";
  var patt1 = /\d/g;
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML = result;
}
</script>
</body>
</html>
```

Output:

Click the button to do a global search for digits in a string.

Try it

1,0,0

Code: The \D metacharacter is used to find a non-digit character.

```
<html> <body>
<p id="demo">Click the button to do a global search for non-digit characters in a string.</p>
<button onclick="myFunction()">Try it</button>
<script>
function myFunction()
{
var str = "Give 100%!";
var patt1 = /\D/g;
var result = str.match(patt1);
document.getElementById("demo").innerHTML=result;
}
</script> </body> </html>
```

Output:

Click the button to do a global search for non-digit characters in a string.

Try it

G,i,v,e, ,%,!

Try it

Code: The . metacharacter is used to find a single character, except newline or other line terminators.

```
<html>
<body>
<p>Click the button to do a global search for "h.t" in a string.</p>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
function myFunction()
{
  var str = "That's hot!";
  var patt1 = /h.t/g;
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML = result;
}
</script>
</body>
</html>
```

Output:

Click the button to do a global search for "h.t" in a string.

Try it

hat,hot

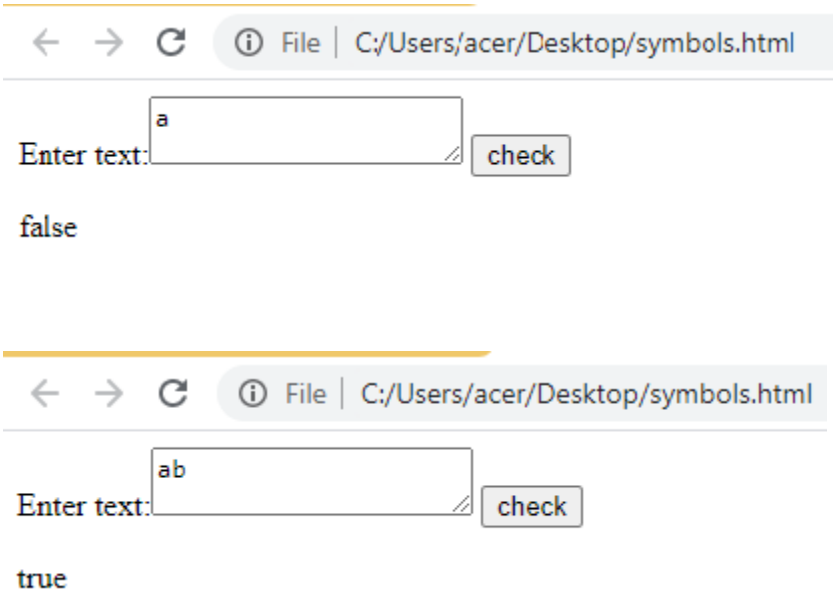
5.1.5 Matching Punctuation and Symbols

- You can have the browser determine whether text contains or doesn't contain letters, punctuation, or symbols, such as the @ sign in an e-mail address, by using the \w and \W special symbols in a regular expression.
- The \w special symbol tells the browser to determine whether the text contains a letter, number, or an underscore, and the \W special symbol reverses this request by telling the browser to determine whether the text contains a character other than a letter, number, or underscore.
- You can use the following regular expression to determine whether the product name that was entered into the form on your web page contains a symbol:
  /\W/
- Using \W is equivalent to using [a-zA-Z0-9].

Example:

```
<html>
<script>
function check()
{
var exp=/\w{2}/;
var str=document.getElementById("txt").value;
var res=exp.test(str);
document.getElementById("demo1").innerHTML=res;
}
</script>
<body>
Enter text:<textarea id="txt"></textarea>
<input type="button" onclick="check()" value="check">
<p id="demo1"></p>
</body>
</html>
```

Output:



Code: The \w metacharacter is used to find a word character.
A word character is a character from a-z, A-Z, 0-9, including the _ (underscore) character.

```
<html>
<body>
<p>Click the button to do a global search for word characters in a string.</p>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
function myFunction()
{
  var str = "Give 100%!";
  var patt1 = /\w/g;
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML = result;
}
```

```
</script>
</body>
</html>
```

Output:

Click the button to do a global search for word characters in a string.

Try it

G,i,v,e,1,0,0

Code: The \W metacharacter is used to find a non-word character.
A word character is a character from a-z, A-Z, 0-9, including the _ (underscore) character.

```
<html>
<body>
<p>Click the button to do a global search for non-word characters in a string.</p>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
function myFunction()
 {
  var str = "Give 100%!";
  var patt1 = /\W/g;
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML = result;
}
</script>
</body>
</html>
```

Output:

Click the button to do a global search for non-word characters in a string.

Try it

,%,!

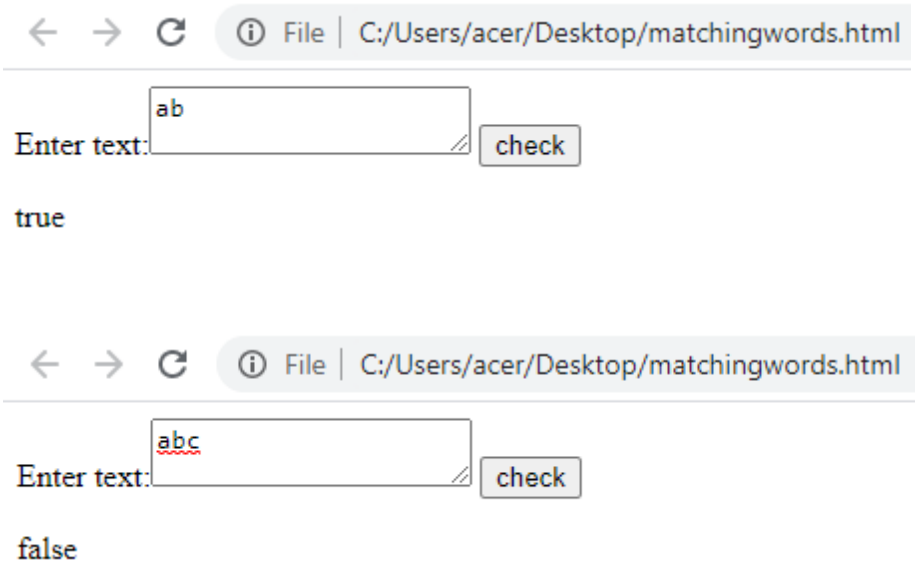
5.1.6 Matching Words

- You might want the browser to search for a particular word within the text. A word is defined by a word boundary—that is, the space between two words. You define a word boundary within a regular expression by using the \b special symbol.
- You need to use two \b special symbols in a regular expression if you want the browser to search for a word: the first \b represents the space at the beginning of the word and the second represents the space at the end of the word.

Example:

```
<html>
<script>
function check()
{
var exp=/\b\w{2}\b/;
var str=document.getElementById("txt").value;
var res=exp.test(str);
document.getElementById("demo1").innerHTML=res;
}
</script>
<body>
Enter text:<textarea id="txt"></textarea>
<input type="button" onclick="check()" value="check">
<p id="demo1"></p>
</body>
</html>
```

Output:

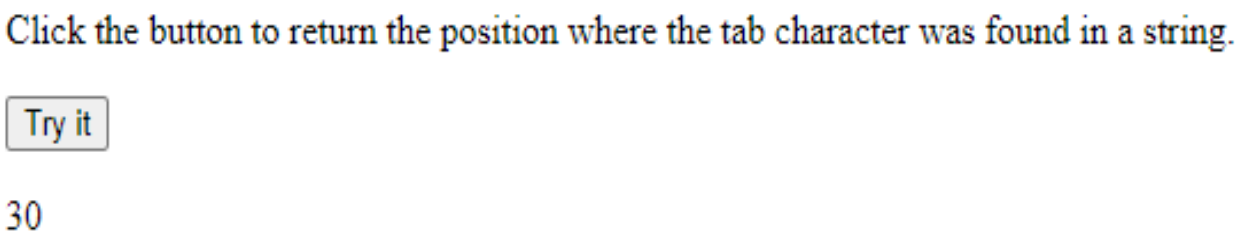


Code: The \t metacharacter is used to find a tab character.
\t returns the position where the tab character was found. If no match is found, it returns -1.

```
<html> <body>
<p>Click the button to return the position where the tab character was found in a string.</p>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
function myFunction()
{
  var str = "Visit Vidyalankar Polytechnic.\t To Learn JavaScript.";
  var patt1 = /\t/;
  var result = str.search(patt1);
  document.getElementById("demo").innerHTML = result;
```

```
}
</script> </body>
</html>
```

Output:



Modifiers/flag

Several modifiers are available that can simplify the way you work with **regexps,** like case sensitivity, searching in multiple lines, etc.

Sr.No.	Modifier & Description
1	**i** Perform case-insensitive matching.
2	**m** Specifies that if the string has newline or carriage return characters, the ^ and $ operators will now match against a newline boundary, instead of a string boundary
3	**g** Performs a global match that is, find all matches rather than stopping after the first match.

Code: The g modifier is used to perform a global match (find all matches rather than stopping after the first match).

Syntax:

```
new RegExp("regexp", "g")
or simply:
/regexp/g
```

Example: 1)

```
<html>
<body>
<p>Click the button to do a global search for "is" in a string.</p>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
function myFunction() {
  var str = "Is this all there is?";
  var patt1 = /is/g;
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML = result;
}
</script>
</body>
</html>
```

Output:

Click the button to do a global search for "is" in a string.

Try it

is,is

Example 2):

```
<html>
<body>
<p>Click the button to do a global search for the character-span [a-h] in a string.</p>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
```

```
<script>
function myFunction()
 {
  var str = "Is this all there is?";
  var patt1 = /[a-h]/g;
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML = result;
}
</script>
</body>
</html>
```

Output:

Click the button to do a global search for the character-span [a-h] in a string.

Try it

h,a,h,e,e

Code:
The i modifier is used to perform case-insensitive matching.

Syntax

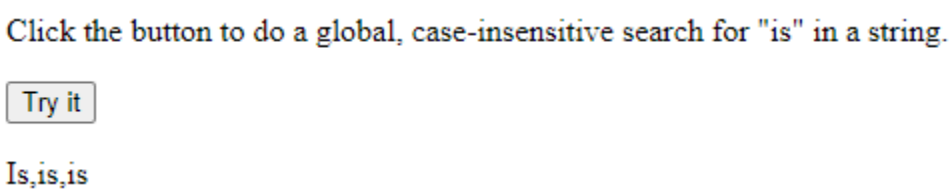
```
new RegExp("regexp", "i")
or simply:
/regexp/i
```

Example:

```
<html>
<body>
<p>Click the button to do a global, case-insensitive search for "is" in a string.
</p>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
```

```
<script>
function myFunction() {
  var str = "Is this all there is?";
  var patt1 = /is/gi;
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML = result;
}
</script>
</body>
</html>
```

Output:



Code: The m modifier is used to perform a multiline match.

The m modifier treat beginning (^) and end ($) characters to match the beginning or end of each line of a string (delimited by \n or \r), rather than just the beginning or end of the string.

Syntax:

```
new RegExp("regexp", "m")

or simply:

/regexp/m
```

Example:

```
<html>
<body>
```

```
<p>Click the button to do a global, case-insensitive, multiline search for "is" at the beginning of each line in a string.</p>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
function myFunction() {
  var str = "\nIs th\nis h\nis?";
  var patt1 = /^is/gmi;
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML = result;
}
</script>
</body>
</html>
```

Output:

Click the button to do a global, case-insensitive, multiline search for "is" at the beginning of each line in a string.

Try it

Is,is,is

RegExp Methods

Here is a list of the methods associated with RegExp along with their description.

Sr.No.	Method & Description
1	exec() Executes a search for a match in its string parameter.
2	test() Tests for a match in its string parameter.

3	toSource() Returns an object literal representing the specified object; you can use this value to create a new object.
4	toString() Returns a string representing the specified object.

5.1.7 Replace Text using a Regular Expression:

- you can also use a regular expression to replace portions of the text by using the replace() method.
- The replace() method requires two parameters: a regular expression and the
- replacement text.
- Here's how the replace() method works. First, you create a regular expression that identifies the portion of the text that you want replaced.
- Then you determine the replacement text. Pass both of these to the replace() method, and the browser follows the direction given in the regular expression to locate the text. If the text is found, the browser replaces it with the new text that you provided.

The replace() method searches a string for a specified value, or a *regular exp*

Syntax

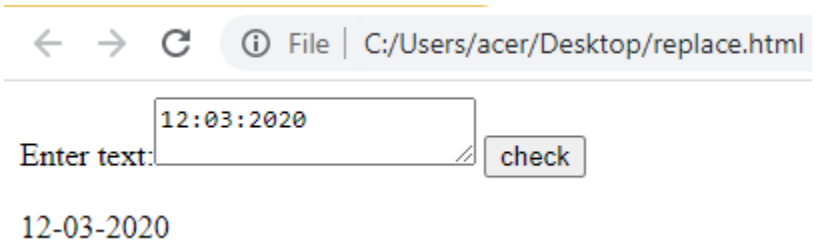
```
string.replace(searchvalue, newvalue)
```

Example:

```
<html>
<script>
function check()
{
var exp=/:/g;
var str=document.getElementById("txt").value;
var res=str.replace(exp,"-");
document.getElementById("demo1").innerHTML=res;
}
```

```
</script>
<body>
Enter text:<textarea id="txt"></textarea>
<input type="button" onclick="check()" value="check">
<p id="demo1"></p>
</body>
</html>
```

Output:



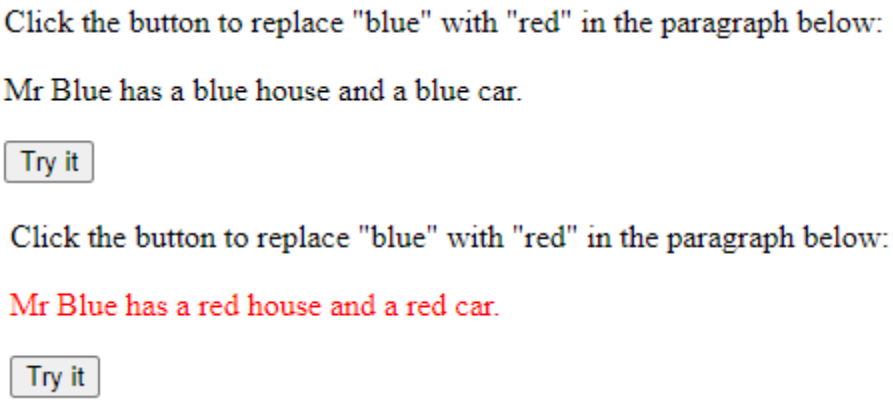
In the above example ":" symbol replace by "-"

Code:

```
<html>
<body>
<p>Click the button to replace "blue" with "red" in the paragraph below:</p>
<p id="demo">Mr Blue has a blue house and a blue car.</p>
<button onclick="myFunction()">Try it</button>
<script>
function myFunction()
{
  var str = document.getElementById("demo").innerHTML;
  var res = str.replace(/blue/g, "red");
  document.getElementById("demo").innerHTML = res;
   document.getElementById("demo").style.color = "red";
```

```
}
</script>
</body>
</html>
```

Output:



5.1.8 Returned the Match Character

- Sometimes your JavaScript application requires you to retrieve characters that match a regular expression rather than simply testing whether or not those characters exist in the text.
- You can have the browser return characters that match the pattern in your regular expression by calling the exec() method of the regular expression object.
- How to use the exec() method. First, create a regular expression that specifies the pattern that you want to match within the text. Characters that match this pattern will be returned to your JavaScript. Next, pass the exec() method the text for which you want to search. The exec() method returns an array. The first array element contains the text that matches your regular expression.

The **exec** method searches string for text that matches regexp. If it finds a match, it returns an array of results; otherwise, it returns null.

Syntax

```
RegExpObject.exec( string );
```

Code:

```
<html>
  <head>
    <title>JavaScript RegExp exec Method</title>
  </head>
    <body>
    <script type = "text/javascript">
      var str = "Javascript is an interesting scripting language";
      var re = new RegExp( "script", "g" );
      var result = re.exec(str);
      document.write("Test 1 - returned value : " +  result);
      re = new RegExp( "pushing", "g" );
      var result = re.exec(str);
      document.write("<br />Test 2 - returned value : " +  result);
    </script>
  </body>
</html>
```

Output:

Test 1 - returned value : script
Test 2 - returned value : null

The **test** method searches string for text that matches regexp. If it finds a match, it returns true; otherwise, it returns false.

Syntax

```
RegExpObject.test( string );
```

Code:

```
<html>
  <head>
```

```
    <title>JavaScript RegExp test Method</title>
  </head>
    <body>
    <script type = "text/javascript">
       var str = "Javascript is an interesting scripting language";
       var re = new RegExp( "script", "g" );

       var result = re.test(str);
       document.write("Test 1 - returned value : " +  result);

       re = new RegExp( "nothing", "g" );

       var result = re.test(str);
       document.write("<br />Test 2 - returned value : " +  result);
    </script>
  </body>
</html>
```

Output:

```
Test 1 - returned value : true
Test 2 - returned value : false
```

The search() method searches a string for a specified value, and returns the position of the match.

The search value can be string or a regular expression.

This method returns -1 if no match is found.

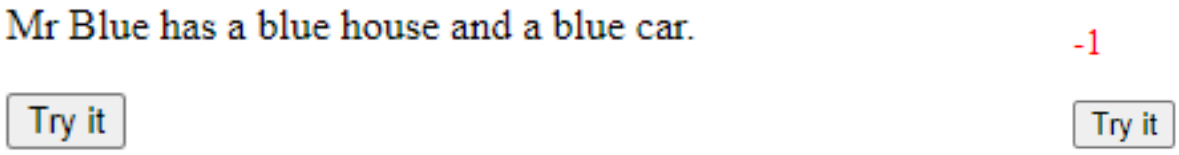
Syntax

```
string.search(searchvalue)
```

Code:

```
<html>
<body>

<p id="demo">Mr Blue has a blue house and a blue car.</p>
<button onclick="myFunction()">Try it</button>
<script>
function myFunction()
{
  var str = document.getElementById("demo").innerHTML;
  var res = str.search(/red/g);
  document.getElementById("demo").innerHTML = res;
   document.getElementById("demo").style.color = "red";
}
</script>
</body>
</html>
```

Output:



5.1.9 RegExp Properties

Here is a list of the properties associated with RegExp and their description.

Sr.No.	Property & Description
1	constructor Specifies the function that creates an object's prototype.
2	global

	Specifies if the "g" modifier is set.
3	ignoreCase Specifies if the "i" modifier is set.
4	lastIndex The index at which to start the next match.
5	multiline Specifies if the "m" modifier is set.
6	source The text of the pattern.

In the following sections, we will have a few examples to demonstrate the usage of RegExp properties.

ignoreCase is a read-only boolean property of RegExp objects. It specifies whether a particular regular expression performs case-insensitive matching, i.e., whether it was created with the "i" attribute.

Syntax:

```
RegExpObject.ignoreCase
```

Code:

```
<html>
  <head>
    <title>JavaScript RegExp ignoreCase Property</title>
  </head>
    <body>
    <script type = "text/javascript">
      var re = new RegExp( "string" );
```

```
      if ( re.ignoreCase )
    {
        document.write("Test1-ignoreCase property is set");
      } else
     {
        document.write("Test1-ignoreCase property is not set");
      }

      re = new RegExp( "string", "i" );

      if ( re.ignoreCase )
    {
        document.write("<br/>Test2-ignoreCase property is set");
      } else
     {
        document.write("<br/>Test2-ignoreCase property is not set");
      }
    </script>
  </body>
</html>
```

Output:

```
Test1-ignoreCase property is not set
Test2-ignoreCase property is set
```

multiline is a read-only boolean property of RegExp objects. It specifies whether a particular regular expression performs multiline matching, i.e., whether it was created with the "m" attribute.

Syntax

```
RegExpObject.multiline
```

Code:

```
<html>
  <head>
    <title>JavaScript RegExp multiline Property</title>
  </head>
    <body>
    <script type = "text/javascript">
       var re = new RegExp( "string" );
       if ( re.multiline )
      {
          document.write("Test1-multiline property is set");
       }
    else
      {
          document.write("Test1-multiline property is not set");
       }
       re = new RegExp( "string", "m" );

       if ( re.multiline )
    {
          document.write("<br/>Test2-multiline property is set");
       }
       Else
      {
          document.write("<br/>Test2-multiline property is not set");
       }
    </script>
  </body>
</html>
```

Output:

```
Test1 - multiline property is not set
Test2 - multiline property is set
```

5.2 Frames

HTML frames are used to divide your browser window into multiple sections where each section can load a separate HTML document. A collection of frames in the browser window is known as a frameset. The window is divided into frames in a similar way the tables are organized: into rows and columns.

5.2.1 Create a Frame

To use frames on a page we use <frameset> tag instead of <body> tag. The <frameset> tag defines, how to divide the window into frames. The rows attribute of <frameset> tag defines horizontal frames and cols attribute defines vertical frames. Each frame is indicated by <frame> tag and it defines which HTML document shall open into the frame.

Note – The <frame> tag deprecated in HTML5. Do not use this element.

Example

Following is the example to create three horizontal frames –

```
<html>
<head>
<title>Create a Frame</title>
</head>
<frameset rows="50%,30%,*">
<frame src="webpage1.html" name="topPage" />
<frame src="webpage2.html" name="bottomPage" />
<frame src="webpage3.html" name="bottomPage" />
</frameset>
</html>
```

Output:

Example

Let's put the above example as follows, here we replaced rows attribute by cols and changed their width. This will create all the three frames vertically –

```
<html>
<head>
<title>Create a Frame</title>
</head>
<frameset cols="50%,30%,*">
<frame src="webpage1.html" name="topPage" />
<frame src="webpage2.html" name="bottomPage" />
<frame src="webpage3.html" name="bottomPage" />
```

```
</frameset>
</html>
```

Output:

The <frameset> Tag Attributes

Following are important attributes of the <frameset> tag –

Sr.No	Attribute & Description
1	**cols** Specifies how many columns are contained in the frameset and the size of each column. You can specify the width of each column in one of the four ways – Absolute values in pixels. For example, to create three vertical frames, use *cols = "100, 500, 100"*. A percentage of the browser window. For example, to create three vertical frames, use *cols = "10%, 80%, 10%"*. Using a wildcard symbol. For example, to create three vertical frames, use *cols = "10%, *, 10%"*. In this case wildcard takes remainder of the window.
2	**rows**

	This attribute works just like the cols attribute and takes the same values, but it is used to specify the rows in the frameset. For example, to create two horizontal frames, use *rows = "10%, 90%"*. You can specify the height of each row in the same way as explained above for columns.
3	**border** This attribute specifies the width of the border of each frame in pixels. For example, border = "5". A value of zero means no border.
4	**frameborder** This attribute specifies whether a three-dimensional border should be displayed between frames. This attribute takes value either 1 (yes) or 0 (no). For example, frameborder = "0" specifies no border.
5	**framespacing** This attribute specifies the amount of space between frames in a frameset. This can take any integer value. For example: framespacing = "10" means there should be 10 pixels spacing between each frame.

The <frame> Tag Attributes

Following are the important attributes of <frame> tag –

Sr.No	Attribute & Description
1	**src**

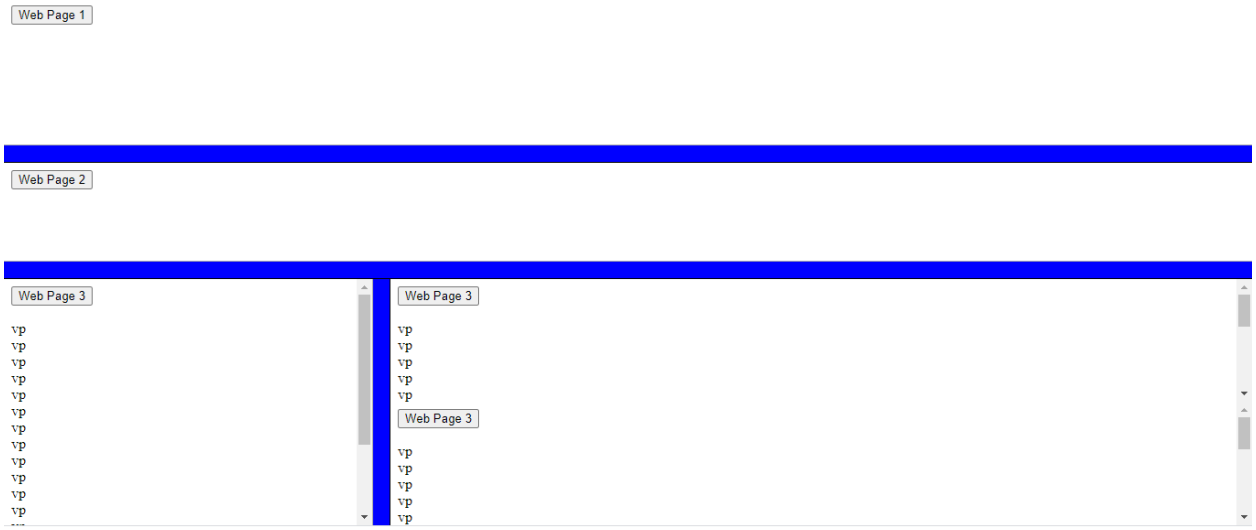
	This attribute is used to give the file name that should be loaded in the frame. Its value can be any URL. For example, src = "/html/top_frame.htm" will load an HTML file available in html directory.
2	**name** This attribute allows you to give a name to a frame. It is used to indicate which frame a document should be loaded into. This is especially important when you want to create links in one frame that load pages into an another frame, in which case the second frame needs a name to identify itself as the target of the link.
3	**frameborder** This attribute specifies whether or not the borders of that frame are shown; it overrides the value given in the frameborder attribute on the <frameset> tag if one is given, and this can take values either 1 (yes) or 0 (no).
4	**marginwidth** This attribute allows you to specify the width of the space between the left and right of the frame's borders and the frame's content. The value is given in pixels. For example, marginwidth = "10".
5	**marginheight** This attribute allows you to specify the height of the space between the top and bottom of the frame's borders and its contents. The value is given in pixels. For example, marginheight = "10".
6	**noresize** By default, you can resize any frame by clicking and dragging on the borders of a frame. The noresize attribute prevents a user from being able to resize the frame.

7	**scrolling** This attribute controls the appearance of the scrollbars that appear on the frame. This takes values either "yes", "no" or "auto". For example, scrolling = "no" means it should not have scroll bars.

Example:

```
<html>
<head>
<title>Create a Frame</title>
</head>
<frameset rows="30%,20%,*" frameborder="1" border="20" bordercolor="blue" scrolling="auto" noresize>
<frame src="webpage1.html" name="topPage" />
<frame src="webpage2.html" name="bottomPage" />
<frameset cols="30%,*">
<frame src="webpage3.html" name="bottomPage" />
<frameset rows="50%,*" frameborder="0" bordercolor="red">
<frame src="webpage3.html" name="bottomPage" />
<frame src="webpage3.html" name="bottomPage" />
</frameset>
</frameset></frameset>
</html>
```

Output:



5.2.2 Invisible border of Frame

User can hide border by using frameborder attribute.

frameborder

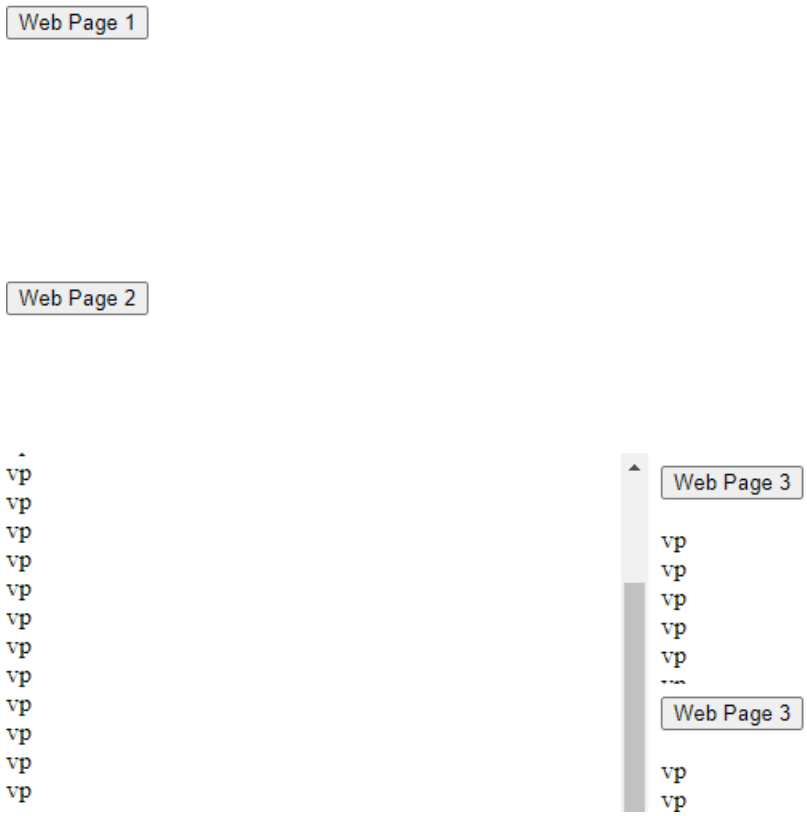
This attribute specifies whether or not the borders of that frame are shown; it overrides the value given in the frameborder attribute on the <frameset> tag if one is given, and this can take values either 1 (yes) or 0 (no).

Example:

```
<html>
<head>
<title>Create a Frame</title>
</head>
<frameset rows="30%,20%,*" frameborder="0">
<frame src="webpage1.html" name="topPage" />
<frame src="webpage2.html" name="bottomPage" />
<frameset cols="30%,*">
```

```
<frame src="webpage3.html" name="bottomPage" />
<frameset rows="50%,*" frameborder="0" bordercolor="red">
<frame src="webpage3.html" name="bottomPage" />
<frame src="webpage3.html" name="bottomPage" />
</frameset>
</frameset></frameset>
</html>
```

Output:



5.2.3 Calling a child window's Javascript function

Using frame in the frameset means creating child window inside the parent window. Here each frame represents a child window and frameset represents the parent window.

You can refer to another child window by referencing the frameset, which is the parent window, and then by referencing the name of the child window, followed by whatever element within the web page of the child window that you want to access.
Following example creates two child windows. First child window (name=topPage) consists of a button.
And Second child window (name=bottomPage) consist of a button.

From second child window you call the method of the first child window using the reference of parent hence when you click on button of second child window then you can call Javascript function of first child window via reference of parent.

Example:

Sample.html

```
<html >
<head>
<title>Create a Frame</title>
</head>
<frameset rows="50%,50%">
<frame src="webpage1.html" name="topPage" frameborder="0"  />
<frame src="webpage2.html" name="bottomPage" frameborder="0"  />
</frameset>
</html>
```

Webpage1.html

```
<html>
<head>
<title>Web Page 1</title>
<script>
function ChangeContent()
 {
alert("Function Called")
}
</script>
```

```
</head>
<body>
<input name="WebPage1" value="Web Page 1" type="button" />
</body>
</html>
```

Webpage2.html

```
<html >
<head>
<title>Web Page 2</title>
</head>
<body>
<p>
<input name="WebPage2" value="Web Page 2" type="button"
onclick="parent.topPage.ChangeContent()" />
</P>
</form>
</body>
</html>
```

5.2.4.1 Changing the content of Child Window:

You can change the content of a child window from a JavaScript function by modifying the source web page for the child window. To do this, you must assign the new source to the child window's href attribute.

In this example, you were able to get a reference to the parent frame's topPage element because they are both from the same domain. At that point, you have two options: if they're in the same domain, you reference it as illustrated previously, but you can also just change the frame src attribute in the frameset to point the frame to a new page.

In the following example we'll need to modify both the WebPage1.html and WebPage2.html fi les. In addition, we'll need to define a new web page called WebPage3.html.

Here is the new WebPage1.html fi le. WebPage1.html appears in the bottom child

window, and when the Web Page 1 button is clicked, the content of the top child window changes from WebPage2.html to WebPage3.html. You'll notice that the value of the button reflects the new content.

Sample.html

```
<html>
<head>
<title>Frame</title>
</head>
<frameset rows="50%,50%">
<frame src="webpage1.html" name="topPage" frameborder="0" border="0" />
<frame src="webpage2.html" name="bottomPage" frameborder="0" border="0" />
</frameset>
</html>
```

Webpage1.html

```
<html>
<head>
<title>web page1</title>
<script>
function ChangeContent()
{
parent.topPage.location.href="webpage3.html";
}
</script>
</head>
<body>
<input name="webpage1" value="WebPage1" type="button" /><br>
</body>
</html>
```

Webpage2.html

```
<html>
<head>
<title></title>
</head>
<body>
<input name="WebPage2" value="Web Page2" type="button" />
</body>
</html>
```

Webpage3.html

```
<html>
<input name="WebPage3" value="web Page3" type="button" />
</html>
```

5.2.4.2 Changing the Focus of a Child Window

The last child window that is created has the focus by default; however, you can

give any child window the focus by changing the focus after all the web pages have loaded in their corresponding child windows. You do this by calling the focus() method of the child window, as shown next, where the focus is being given to the web page that appears in the bottomPage child window. You can call the focus() method from a JavaScript function or directly in response to an event such as the onclick event. The reference to parent. bottomPage is needed to get past the security issues.

parent.bottomPage.focus();

5.2.5 Writing to a Child Window from a JavaScript

Typically, the content of a child window is a web page that exists on the web server. However, you can dynamically create the content when you defi ne the frameset by directly writing to the child window from a JavaScript. The JavaScript must be defi ned in the HTML fi le that defi nes the frameset and called when the frameset is loaded.

```
<html>
```

```
<head>
<title>Create a Frame</title>
<script>
function ChangeContent()
{
window.topPage.document.open()
window.topPage.document.writeln('<html >')
window.topPage.document.writeln('<head>')
window.topPage.document.writeln('<title>Web Page 3</title>')
window.topPage.document.writeln('</head>')
window.topPage.document.writeln('<body>')
window.topPage.document.writeln('<FORM action="" method="post">')
window.topPage.document.writeln('<P>')
window.topPage.document.writeln(
'<INPUT name="WebPage3" value="Web Page 3" type="button" />')
window.topPage.document.writeln('</P>')
window.topPage.document.writeln('</FORM>')
window.topPage.document.writeln('</body>')
window.topPage.document.writeln('</html>')
window.topPage.document.close()
}
</script>
<frameset rows="50%,50%" onload="ChangeContent()">
<frame src="webpage1.html" name="topPage" />
<frame src="webpage2.html" name="bottomPage" />
</frameset>
</html>
<frame src="webpage2.html" name="bottomPage" />
</frameset>
</html>
```

5.2.6 Accessing Elements of Another Child Window

You can access and change the value of elements of another child window by directly referencing the element from within your JavaScript. You must explicitly specify the full path to the element in the JavaScript statement that references the element, and it must be from the same domain as the web page; otherwise, a security violation occurs.

```
<html>
<head>
<title>Create a Frame</title>
<script >
function ChangeContent()
 {
window.topPage.document.open()
window.topPage.document.writeln('<html >')
window.topPage.document.writeln('<head>')
window.topPage.document.writeln('<title>Web Page 3</title>')
window.topPage.document.writeln('</head>')
window.topPage.document.writeln('<body>')
window.topPage.document.writeln(
'<FORM name="Form1" action="" method="post">')
window.topPage.document.writeln('<P>')
window.topPage.document.writeln('<input name="Text1" type="text"/>
window.topPage.document.writeln(
'<INPUT name="WebPage1" value="Web Page 1" type="button" />')
window.topPage.document.writeln('</P>')
window.topPage.document.writeln('</FORM>')
window.topPage.document.writeln('</body>')
window.topPage.document.writeln('</html>')
window.topPage.document.close()
}
```

```
</script>
</head>
<frameset rows="50%,50%" onload="ChangeContent()">
<frame src="webpage1.html" name="topPage" />
<frame src="webpage2.html" name="bottomPage" />
</frameset>
</html>
```

```
<html>
<head>
<title></title>
</head>
<script>
function accessElement()
{
parent.topPage.Form1.Text1.value="Manisha";
parent.topPage.Form1.WebPage1.value="Accessed";
}
</script>
<body>
<h1>Web Page 2</h1>
<input name="WebPage2" value="Web Page2" type="button" onclick="accessElement()" />
</body>
</html>
```

Note: This Frameset and javascript is not supported by browser.

IFRAMES

You can define an inline frame with HTML tag **<iframe>**. The <iframe> tag is not somehow related to <frameset> tag, instead, it can appear anywhere in your document. The <iframe> tag defines a rectangular region within the document in which the browser can display a

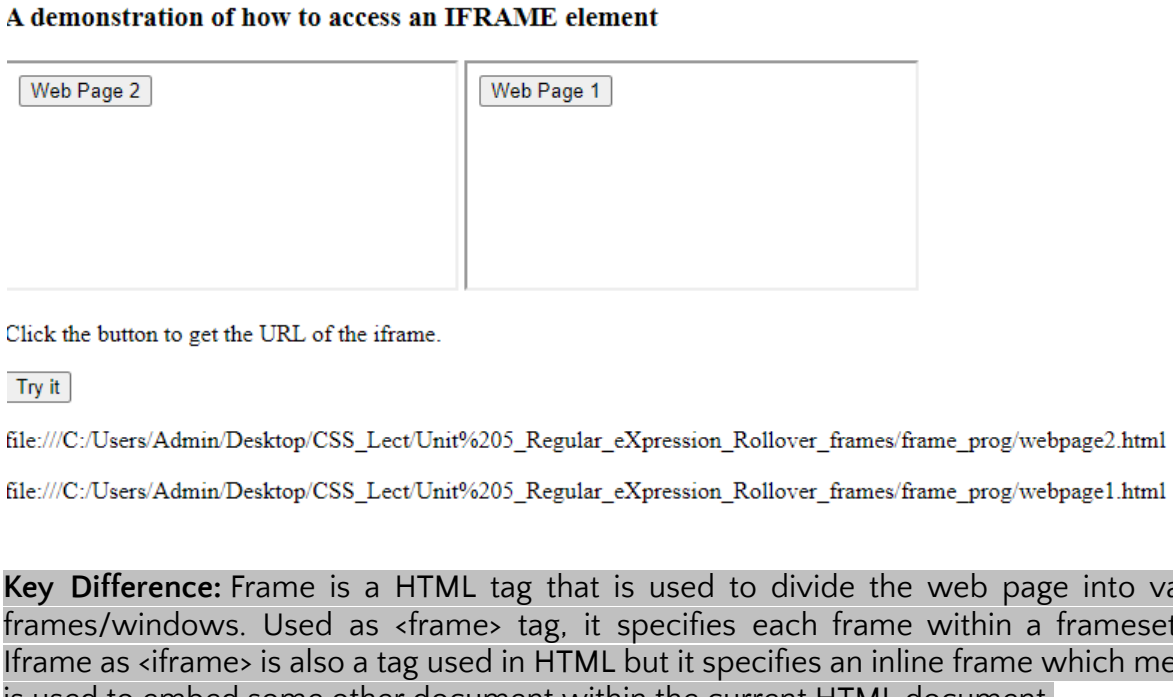
separate document, including scrollbars and borders. An inline frame is used to embed another document within the current HTML document.

The **src** attribute is used to specify the URL of the document that occupies the inline frame.

Example:

```
<html>
<body>
<h3>A demonstration of how to access an IFRAME element</h3>
<iframe id="myFrame" src="webpage2.html"></iframe>
<iframe id="myFrame1" src="webpage1.html"></iframe>
<p>Click the button to get the URL of the iframe.</p>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<p id="demo1"></p>
<script>
function myFunction()
 {
  var x = document.getElementById("myFrame").src;
  document.getElementById("demo").innerHTML = x;
var y = document.getElementById("myFrame1").src;
  document.getElementById("demo1").innerHTML = y;
}
</script>
</body>
</html>
```

Output:



Key Difference: Frame is a HTML tag that is used to divide the web page into various frames/windows. Used as <frame> tag, it specifies each frame within a frameset tag. Iframe as <iframe> is also a tag used in HTML but it specifies an inline frame which means it is used to embed some other document within the current HTML document.

We can access the content of child window by using <iframe>.
In following example, we have used window.addEventListener().

Code: Parent.html

```
<html>
<head>
   <script type="text/javascript">
      window.addEventListener('message', receiveMessage, false);
function receiveMessage(event)
     {
         alert("got message: " + event.data);
      }
   </script>
```

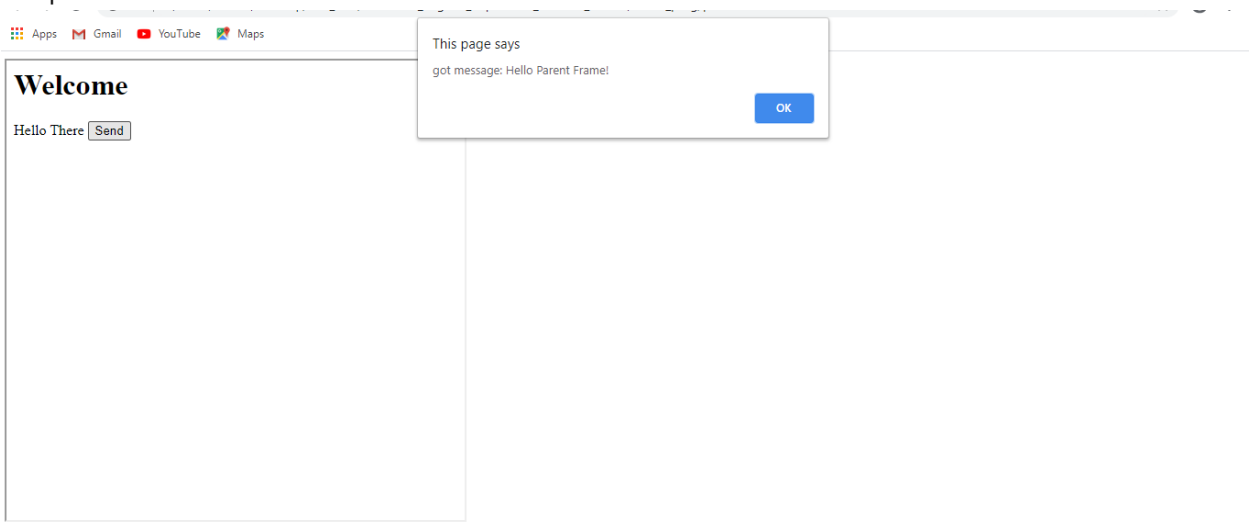
```
</head>
<title> Parent Page </title>
<body>
   <iframe src="iframe_p.html" width="500" height="500"></iframe>
</body>
</html>
```

iframe_p.html

The **window.postMessage()** method safely enables cross-origin communication between Window objects; *e.g.*, between a page and a pop-up that it produced, or between a page and an iframe embedded within it.

```
<html>
<head>
   <script>
      function send()
      {
         window.parent.postMessage('Hello Parent Frame!', '*');
      }
   </script>
</head>
<title> IFrame Test </title>
<body>
   <h1> Welcome </h1>
   <p> Hello There </body>
<button onclick="send()">Send</button>
</body>
</html>
```

Output:



5.3 Rollover

Rollover means a webpage changes when the user moves his or her mouse over an object on the page. It is often used in advertising. There are two ways to create rollover, using plain HTML or using a mixture of JavaScript and HTML. We will demonstrate the creation of rollovers using both methods.

The keyword that is used to create rollover is the <onmousover> event.

For example, we want to create a rollover text that appears in a text area. The text *"What is rollover?"* appears when the user place his or her mouse over the text area and the rollover text changes to *"Rollover means a webpage changes when the user moves his or her mouse over an object on the page"* when the user moves his or her mouse away from the text area.

The HTML script is shown in the following example:

```
<html>
<head></head>
```

```
<Body>
<textarea rows=”2″ cols=”50″ name=”rollovertext” onmouseover=”this.value='What is rollover?'”
onmouseout=”this.value='Rollover means a webpage changes when the user moves his or her mouse over an object on the page'”></textarea>
</body>
</html>
```

In following example, we create a rollover effect that can change the color of its text using the style attribute.

```
<p
onmouseover=”this.style.color='red'”
onmouseout=”this.style.color='blue'”>
Move the mouse over this text to change its color to red. Move the mouse away to change the text color to blue.
</p>
```

Using an iFrame as Link Target

An iframe can also be used as a target for the hyperlinks.

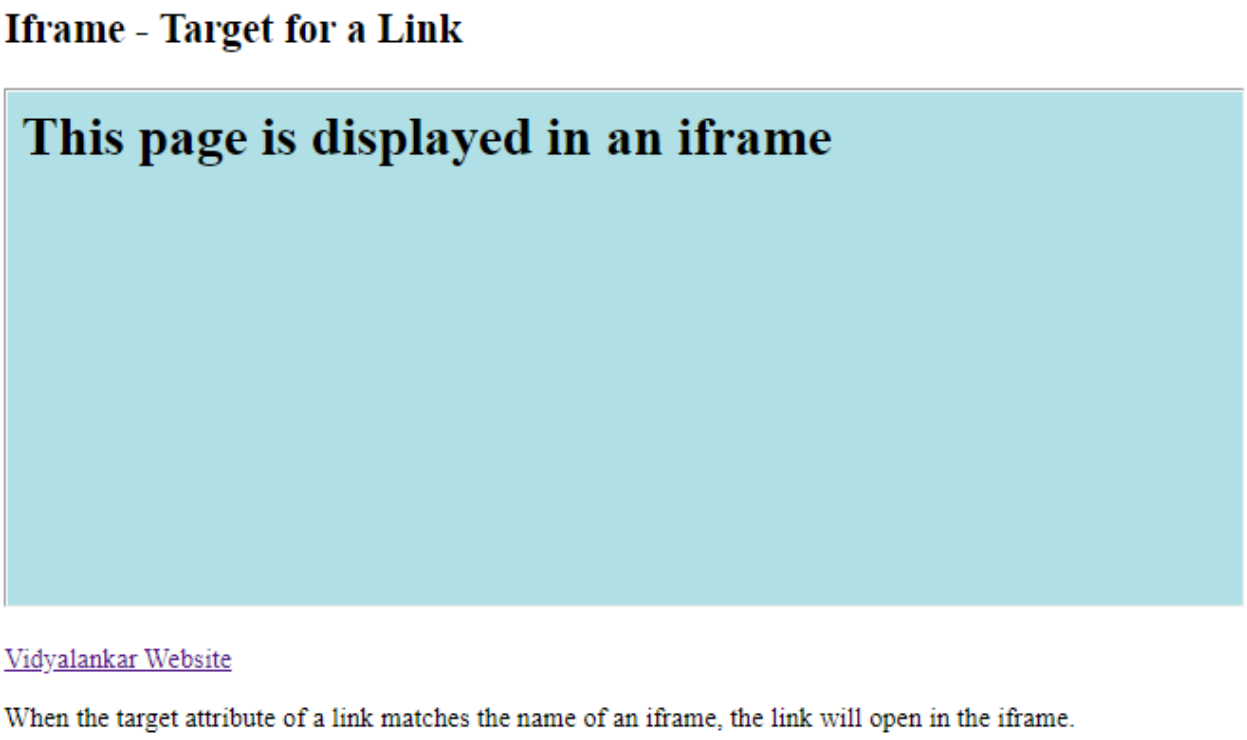
An iframe can be named using the name attribute. This implies that when a link with a target attribute with that name as value is clicked, the linked resource will open in that iframe.

Example:

```
<html>
<body>

<h2>Iframe – Target for a Link</h2>

<iframe src="demo_iframe.htm" name="myframe" height="300px" width="100%"
title="Iframe Example using Link"></iframe>
```

```
<p><a href="https://vpt.edu.in" target="myframe">Vidyalankar Website</a></p>

<p>When the target attribute of a link matches the name of an iframe, the link will open in the iframe.</p>

</body>
</html>
```

Output:



After clicking on Vidyalankar Website, vpt.edu.in will open

Iframe - Target for a Link

Vidyalankar Website

When the target attribute of a link matches the name of an iframe, the link will open in the iframe.

5.3.1 Creating Rollover

Though HTML can be used to create rollovers, it can only perform simple actions. If you wish to create more powerful rollovers, you need to use JavaScript. To create rollovers in JavaScript, we need to create a JavaScript function.

Code: Changing image using onmouseover and onmouseout (rollover and rollback)

```
<html>
<head>
<title>JavaScript Image Rollovers</title>
</head>
<body>
<img src="blue.png" name="image1" onmouseover="src='car.jpg' "
onmouseout="src='blue.png' ">
  </img>
</body>
</html>
```

5.3.2 Text Rollover

We can also create a rollover and rollback for text using the **onmouseover** and **onmouseout.**

Code: Changing image (rollover and rollback)

```
<html>
<head>
<title>
text rollovers</title>
</head>
<body>
<table border="1" width="100%">
<tbody>
<tr valign="top">
```

```
<td width="50%">
<a><img height="500" src="blue.png" width="900" name="clr"></a></td>
<td><a onmouseover="document.clr.src='blue.png' ">
<b><u> Blue Color</u></b></a>
<br>
<a onmouseover="document.clr.src='red.png' ">
<b><u> Red Color</u></b></a>
<br>
<a onmouseover="document.clr.src='green.png' ">
<b><u> Green Color</u></b></a>
</td>
</tr>
</tbody>
</table>
</body>
</html>
```

Output:

5.3.3 Multiple actions for a Rollover

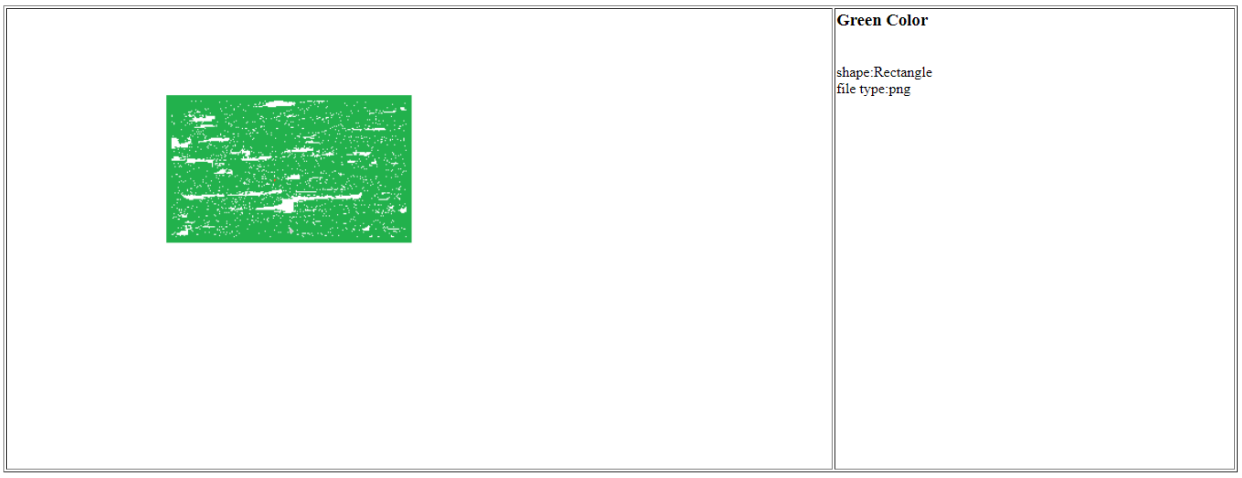
User want to change the image as well as its description on the web page whenever the user moves the mouse cursor over the image.

Code: change image and its description.

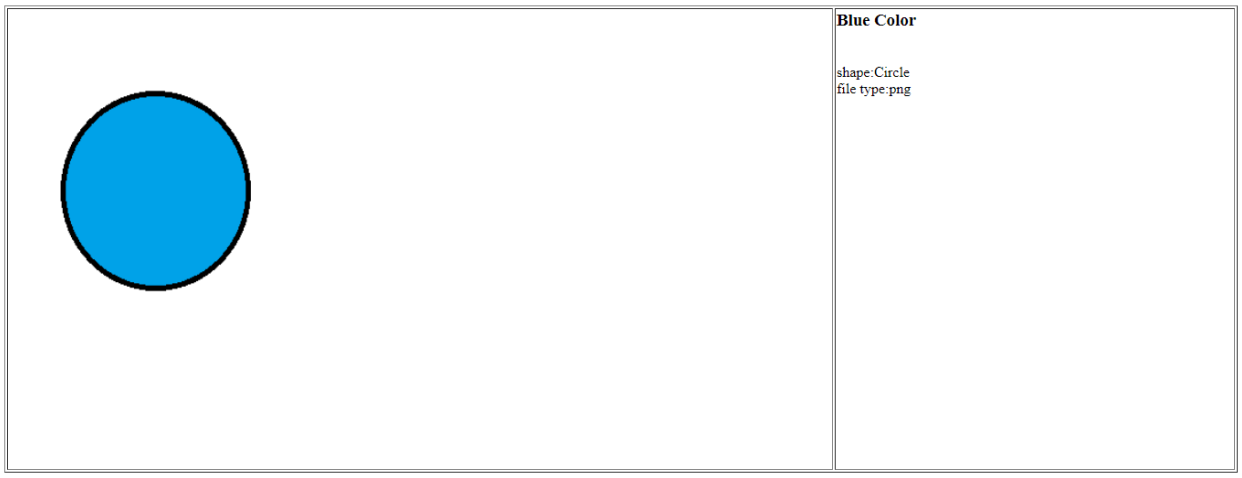
```
<html> <head>
<title>
text rollovers</title>
<script>
function update_details()
{
document.getElementById("i1").src="blue.png";
document.getElementById("detail").innerHTML="<h3> Blue Color </h3> <br> shape:Circle <br> file type:png";
}
</script>
</head>
<body>
<table id="t1" border="1" width="100%">
<tbody>
<tr valign="top">
<td width="50%">
<a><img id="i1" height="500" src="green.png" width="900" name="clr" onmouseover="update_details()"></a></td>
<td id="detail">
<h3> Green Color </h3>
<br> shape:Rectangle
<br> file type:png
</td>
</tr>
```

```
</tbody>
</table>
</body>
</html>
```

Output:



When user moves mouse over an image, description and image will chage.

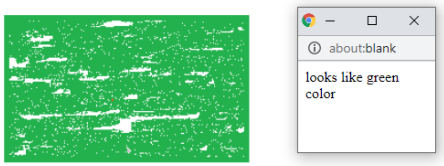


Code: Multiple rollover and rollback.

```
<html>
<head>
<title>
text rollovers</title>
<script>
function open_new_window(clrname)
{
if(clrname==1)
{
document.clr.src="red.png";
mwin=window.open('','myadwin','height=100,width=150,left=500,top=200');
mwin.document.write("looks like red color");
}
if(clrname==2)
{
document.clr.src="green.png";
mwin=window.open('','myadwin','height=100,width=150,left=500,top=200');
mwin.document.write("looks like green color");
}

if(clrname==3)
{
document.clr.src="blue.png";
mwin=window.open('','myadwin','height=100,width=150,left=500,top=200');
mwin.document.write("looks like blue color");
}
}
</script>
</head>
<body>
```

```
<table border="1" width="100%">
<tbody>
<tr valign="top">
<td width="50%">
<a><img height="500" src="red.png" width="900" name="clr"></a></td>
<td><H2>
<a onmouseover="open_new_window(1)" onmouseout="mwin.close()">
<b><u>RED</u></b></a>
<br><br>
<a onmouseover="open_new_window(2)" onmouseout="mwin.close()">
<b><u>GREEN</u></b></a>
<br><br>
<a onmouseover="open_new_window(3)" onmouseout="mwin.close()">
<b><u>BLUE</u></b></a>
</H2>
</td>
</tr>
</tbody>
</table>
</body>
</html>
```

Output:



RED

GREEN

BLUE

RED

GREEN

BLUE

5.3.4 More efficient Rollover

We can avoid the delay by loading all images once when they are referred first time on web page.

Code:

```
<html>
<head>
<title>
```

```
text rollovers</title>
<script>
b=new Image;
r=new Image;
g=new Image;
if(document.images)
{
b.src='blue.png';
r.src='red.png';
g.src='green.png';
}
else
{
b.src='';
r.src='';
g.src='';
document.clr='';
}

</script>
</head>
<body>
<table border="1" width="100%">
<tbody>
<tr valign="top">
<td width="50%">
<a><img height="500" src="blue.png" width="900" name="clr"></a></td>
<td><H2>
<a onmouseover="document.clr.src='blue.png'">
<b><u>BLUE</u></b></a>
```

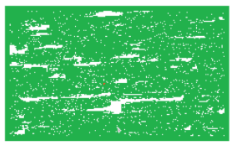
```
<br><br>
<a onmouseover="document.clr.src='red.png'">
<b><u>RED</u></b></a>
<br><br>
<a onmouseover="document.clr.src='green.png'">
<b><u>GREEN</u></b></a>
</H2>
</td>
</tr>
</tbody>
</table>
</body>
</html>
```

Output:

BLUE

RED

GREEN

	**BLUE** **RED** **GREEN**

```
<html>
<head>
<title>JavaScript Regular expression to valid an email address</title>
</head>
<body>
<script>
function valid_email(str)
{
```

```
var mailformat = /^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/;
if(mailformat.test(str))
{
alert("Valid email address!");
}
else
{
alert("You have entered an invalid email address!");
}
}

valid_email('yogita.jore@gmail.com');



</script>
</body>
</html>
```