JavaScript

(Events)

Events

- Mouse
- Document/Window Events
- Form
- keyboard

Mouse Events

- **click:** The click event fires after you click and release the mouse button
- **dblclick:** When you press and release the mouse button twice, a double-click event fires.
- **mousedown:** The mousedown event is the first half of a click—the moment when you click the button before releasing it.
- **mouseup:** The mouseup event is the second half of a click—the moment when you release the button.
- **mouseover:** When you move your mouse over an element on a page, a mouseover event fires.
- **mouseout:** Moving a mouse off an element triggers the mouseout event.
- **mousemove:** Logically enough, the mousemove event fires when the mouse moves which means this event fires all of the time.

Document/Window Events

- **load**: The load event fires when the Web browser finishes downloading all of a Web page's files the HTML file itself, plus any linked images, Flash movies, and external CSS and JavaScript files.
- **resize**: When you resize your browser window by clicking the maximize button, or dragging the browser's resize handle, the browser triggers a resize event.
- **scroll**: The scroll event is triggered whenever you drag the scroll bar, or use the keyboard (up/down/home/end and so on keys) or mouse scroll wheel to scroll a Web page.
- **unload**: When you click a link to go to another page, close a browser tab, or close a browser window, a Web browser fires an unload event.

Form Events

- **submit**: Whenever a visitor submits a form, the submit event fires.
- **reset**. Although not as common as they used to be, a Reset button lets you undo any changes you've made to a form.
- **change**. Many form fields fire a change event when their status changes
- **focus**. When you tab or click into a text field, it gives the field focus
- **blur:** The blur event is the opposite of focus. It's triggered when you exit a currently focused field, by either tabbing or clicking outside the field.

Keyboard Events

- **keypress**: The moment you press a key, the keypress event fires.
- **keydown**: The keydown event is like the keypress, it's fired right before the keypress event
- **keyup**: Finally, the keyup event is triggered when you release a key.

Event Handlers

Event Handlers	Triggered when
onChange	The value of the text field, text area, or a drop down list is modified
onClick	A link, an image or a form element is clicked once
onDblClick	The element is double-clicked
onMouseDown	The user presses the mouse button
onLoad	A document or an image is loaded
onSubmit	A user submits a form
onReset	The form is reset
onUnLoad	The user closes a document or a frame
onResize	A form is resized by the user

Events Examples

- A link**
- <body onload="startSlideShow( )">

- function message( ) {
  alert("Welcome...");
  }
  window.onload=message;

`onLoad` Event Handler Example

```
<html><head>
<title>onLoad and onUnload Event Handler Example</title>
</head>
<body
  onLoad="alert('Welcome to this page')"
  onUnload="alert('Thanks for visiting this page')"
>
Load and UnLoad event test.
</body>
</html>
```

`onMouseOver` & `onMouseOut` Event Handler

```
<html>
<head>
<title>onMouseOver / onMouseOut Event Handler Demo</title>
</head>
<body>
<a href="http://www.jiit.ac.in"
  onMouseOver="window.status='JIIT Home'; return true;"
  onMouseOut="status=""
>JIIT</a>
</body>
</html>
```

- When the mouse cursor is over the link, the browser displays the text “JIIT Home" instead of the URL.
- The "return true;" of onMouseOver forces browser not to display the URL.

`onClick` Event Handler Example

```
<html>
<head>
<title>onClick Event Handler Example</title>
<script type="text/javascript">
function warnUser() {
        return confirm("Are you a student?”);
}
</script>
</head>
<body>
<a href="ref.html" onClick="return warnUser()">
<!--
  If onClick event handler returns false, the link
  is not followed.
-->
Students access only</a>
</body>
</html>
```

`onSubmit` Event Handler Example

```
<html><head>
<title>onSubmit Event Handler Example</title>
<script type="text/javascript">
  function validate() {
    // If everything is ok, return true
    // Otherwise return false
  }
</script>
</head>
<body>
<form action="MessageBoard" method="POST"
 onSubmit="return validate();"
>
…
</form></body></html>
```

- If onSubmit event handler returns false, data is not submitted.
- If onReset event handler returns false, form is not reset

Build-In JavaScript Objects

Object	Description
Array	Creates new array objects
Boolean	Creates new Boolean objects
Date	Retrieves and manipulates dates and times
Error	Returns run-time error information
Function	Creates new function objects
Math	Contains methods and properties for performing mathematical calculations
Number	Contains methods and properties for manipulating numbers.
String	Contains methods and properties for manipulating text strings

- See online references for complete list of available methods in these objects: http://javascript-reference.info/

String Object (Some useful methods)

- length
  - –A string property that tells the number of character in the string
- charAt(idx)
  - –Returns the character at location "idx"
- toUpperCase(), toLowerCase()
  - –Returns the same string with all uppercase/lowercase letters
- substring(beginIdx)
  - –Returns a substring started at location "beginIdx"
- substring(beginIdx, endIdx)
  - –Returns a substring started at "beginIdx" until "endIdx" (but not including "endIdx"
- indexOf(str)
  - –Returns the position where "str" first occurs in the string

Error and Exception Handling in JavaScript

- Javascript makes no distinction between Error and Exception (Unlike Java)
- Handling Exceptions
  - The onError event handler
    - A method associated with the window object.
    - It is called whenever an exception occurs
  - The try … catch … finally block
    - Similar to Java try … catch … finally block
    - For handling exceptions in a code segment
  - Use throw statement to throw an exception
  - The Error object
    - Default object for representing an exception
    - Each Error object has a name and message properties

try … catch … finally

```
try {
  // Contains normal codes that might throw an exception.

  // If an exception is thrown, immediately go to
  //   catch block.

} catch ( errorVariable ) {
  // Codes here get executed if an exception is thrown
  //   in the try block.

  // The errorVariable is an Error object.

} finally {
  // Executed after the catch or try block finish

  // Codes in finally block are always executed
}
// One or both of catch and finally blocks must accompany the try block.
```

try … catch … finally example

```
<script type="text/javascript">
try{
  document.write("Try block begins<br>");
  // create a syntax error
  eval ("10 + * 5");

} catch( errVar ) {// errVar is an Error object
  document.write("Exception caught<br>");
  // errVar is an Error object
  // All Error objects have a name and message properties
  document.write("Error name: " + errVar.name + "<br>");
  document.write("Error message: " + errVar.message +  "<br>");
} finally {
  document.write("Finally block reached!");
}
</script>
```

Throwing Exception

```
<script type="text/javascript">
try{
  var num = prompt("Enter a number (1-2):", "1");
  // You can throw exception of any type
  if (num == "1")
    throw "Some Error Message";
  else
  if (num == "2")
    throw 123;
  else
    throw new Error ("Invalid input");
} catch( err ) {
  alert(typeof(errMsg) + "\n" + err);
  // instanceof operator checks if err is an Error object
  if (err instanceof Error)
    alert("Error Message: " + err.message);
}
</script>
```

Characters used for matching multiple occurrences of the same character or pattern

Character	Matches
?	Zero or one occurrences of the previous item, meaning the previous item is optional, but if it does appear, it can only appear once. For example the regex *colou?r* will match both "color" and "colour," but not "colouur."
+	One or more occurrences of the previous item. The previous item must appear at least once.
*	Zero or more occurrences of the previous item. The previous item is optional and may appear any number of times. For example, .* matches any number of characters.
{n}	An exact number of occurrences of the previous item. For example \d{3} only matches three numbers in a row.
{n, }	The previous item *n* or more times. For example, a{2,} will match the letter "a" two or more times: that would match "aa" in the word "aardvark" and "aaaa" in the word "aaaahhhh."
{n,m}	The previous item at least *n* times but no more than *m* times. So \d{3,4} will match three or four numbers in a row (but not two numbers in a row, nor five numbers in a row).

Finding Patterns in Strings

Character	Matches
.	Any one character—will match a letter, number, space, or other symbol
\w	Any word character including a–z, A–Z, the numbers 0–9, and the underscore character: _.
\W	Any character that's not a word character. It's the exact opposite of \w.
\d	Any digit 0–9.
\D	Any character except a digit. The opposite of \d.
\s	A space, tab, carriage return, or new line.
\S	Anything but a space, tab, carriage return, or new line.
^	The beginning of a string. This is useful for making sure no other characters come before whatever you're trying to match.
$	The end of a string. Use $ to make sure the characters you wish to match are at the end of a string. For example, /com$/ matches the string "com", but only when it's the last three letters of the string. In other words, /com$/ would match "com" in the string "Infocom," but not 'com' in 'communication'.
\b	A space, beginning of the string, end of string, or any nonletter or number character such as +, =, or '. Use \b to match the beginning or ending of a word, even if that word is at the beginning or ending of a string.

[ ]	Any one character between the brackets. For example, [aeiou] matches any one of those letters in a string. For a range of characters, use a hyphen: [a-z] matches any one lower case letter; [0-9] matches any one number (the same as \d.)
[^]	Any character except one in brackets. For example, [^aeiouAEIOU] will match any character that isn't a vowel. [^0-9] matches any character that's not a number (the same as \D).
|	Either the characters before or after the | character. For example, a|b will match either *a* or *b*, but not both.
\	Used to escape any special regex symbol (*,.,\,/, for instance) to search for a literal example of the symbol in a string. For example, . in regex-speak means "any character," but if you want to really match a period in a string you need to

escape it, like this: \. .

Examples

- to match five numbers:
  - *\d{5}.*
- **Find .gif**
  - *\.gif*
- **Find any number of characters before .gif.**
  - matching more than just a file name.
    - *.*\.gif*
  - Matching only file logo.gif
    - *\S character* matches any nonspace character:
    - *\S*\.gif*

- **Make the search case-insensitive**
- /\S*\.gif/i
  - *i goes outside of the pattern and to the right of the / that defines* the end of the regular expression pattern

Example:

- var testString = 'The file is logo.gif'; // the string to test
- var gifRegex = /\S*\.gif/i; // the regular expression
- var results = testString.match(gifRegex);
- var file = results[0]; // logo.gif

Matching Apr

Possible words: Apr or April or Apricot or Aprimecorp

```
var sentence = 'April is a warm month.';
var aprMatch = /Apr(il)?\b/;
if (sentence.search(aprMatch) != -1) {
// found Apr or April
} else {
//not found
}
```

- Here—/Apr(il)?\b/—makes the “Apr” required, but
- the subpattern—(il)—optional (that ? character means zero or one time).
- Finally, the \b matches the end of a word

U.S. Zip code

- Either five numbers, or five numbers followed by a hyphen and four numbers: 97213 or 97213-3333.
- \d{5}(-\d{4})?
  - \d{5} matches five digits, as in 97213
  - ( ) creates a subpattern
  - -\d{4} matches the hyphen followed by four digits, like this: -1234.
  - ? matches zero or one instance of the preceding pattern.

U.S. phone number

- U.S. phone numbers have a three-digit area code followed by seven more digits
  - 503-555-1212,
  - (503) 555-1212,
  - 503.555.1212, or
  - 503 555 1212.

503-555-1212, (503) 555-1212, 503.555.1212, or just 503 555 1212.

\(?

\(matches a literal opening parenthesis character

? indicates that the (character is optional

\(?(\d{3})

(\d{3}) is a sub-pattern that matches any three digits

\(?(\d{3})\)?

\)? matches an optional closing parenthesis

\(?(\d{3})\)?[-.]

-.] will match either a space, hyphen, or period

\(?(\d{3})\)?[-.](\d{3})[-.](\d{4})

(\d{4}) is a sub-pattern that matches any four digits

Email address

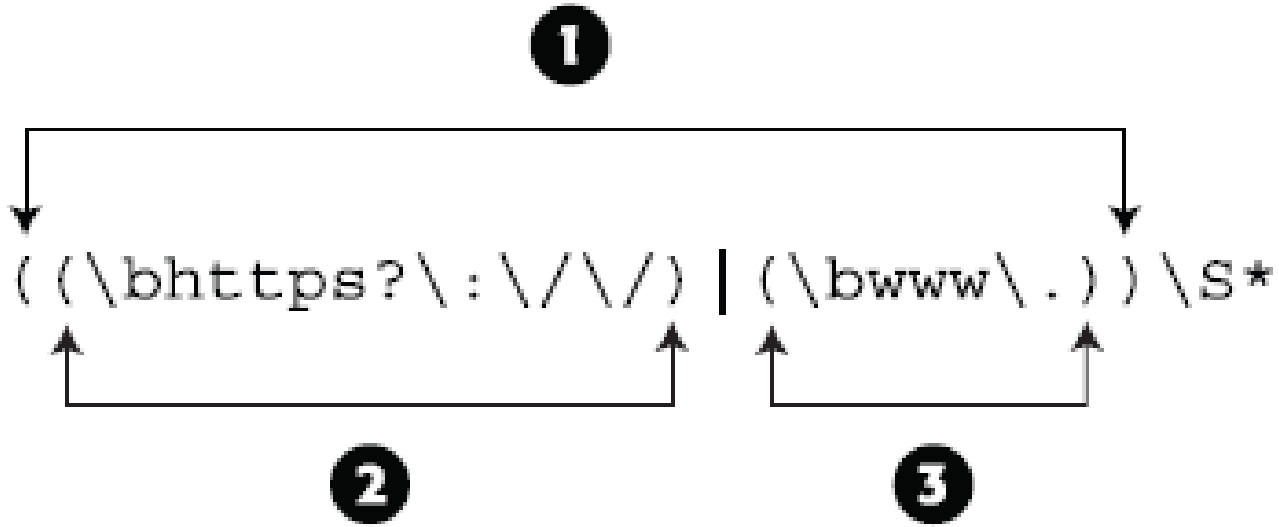
[-\w.]+@([A-z0-9][-A-z0-9]+\.)+[A-z]{2,4}

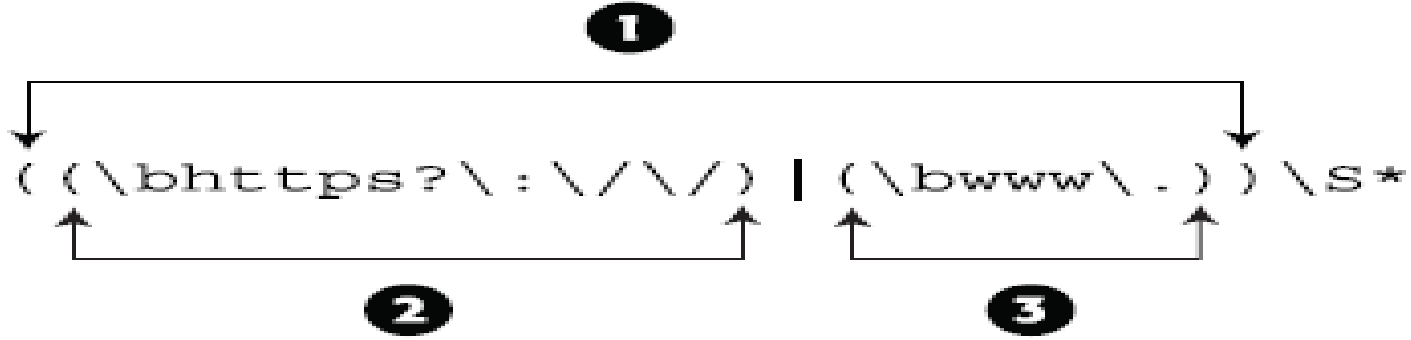
- [-\w.]+ matches a hyphen, any word character, or a period one or more times. So it will match “bg,” “bg.gupta,” or “bg-gupta.”
- @ is the @ sign you find in an email address: abc@gmail.com.
- [A-z0-9] matches one letter or number.
- [-A-z0-9]+ matches one or more instances of a letter, number, or hyphen.
- \. is a period character so it would match the period in gmail.com.
- + matches one or more instances of the pattern that includes the above three matches. This character allows for subdomain names like bob@mail.yahoo.com.
- [A-z]{2,4} is any letter 2, 3, or 4 times. This matches the com in .com, or .uk or .in.

Web Address

Web address

- **((\bhttps?:\/\/)|(\bwww\.))\S***





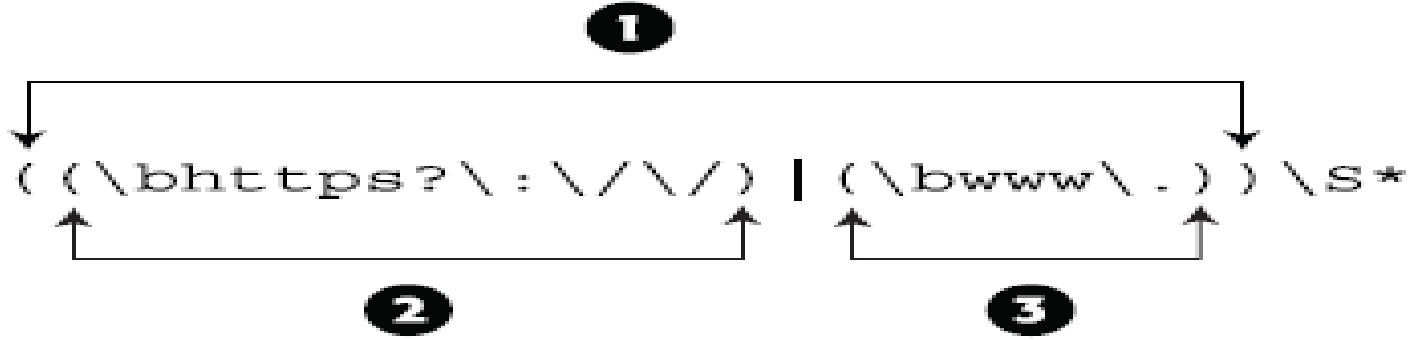
- *(is the start of the outer group (1 in Figure)*
- *(is the start of inner group (2 in Figure).*
- *\b matches the beginning of a word.*
- *http matches the beginning of a complete Web address that begins with http.*
- *s? is an optional s.*
- *:\/\/ matches ://. Since the forward slash has meaning in regular expressions,*

you need to precede it by a backslash to match the forward slash character.

- *) is the end of the inner group (2 in Figure). Taken all together, this group*

will match either *http:// or https://.*

- *| matches either one or the other group (2 or 3 in Figure).*



- *(is the start of second inner group (3 in Figure).*
- ***\b matches the beginning of a word.***
- *www\. matches www.*
- *) is the end of the second inner group (3 in Figure). This group will capture* a URL **that is missing the *http://*** *but begins with www.*
- *) is the end of the outer group (1 in Figure). At this point, the regular* expression will match text that begins with *http://, https://, or www.*
- **\S* matches zero or more nonspace characters.**

Date

- US date format: 09/28/2008, 9-28-2007,09 28 2007, or even 09.28.2007
- **([01]?\d)[-\/ .]([0123]?\d)[-\/ .](\d{4})**
- *[01]? matches either 0 or 1 and the ? makes this optional*
- *\d matches any number*
- *[-\/ .] will match a hyphen, a forward slash, a period, or a space character.*
- *[0123]? matches either 0, 1, 2, or 3 zero or more times*

URL using *match( )*

```
// create a variable containing a string with a URL
var text='my web site is www.missingmanuals.com';
// create a regular expression
var urlRegex = /((\bhttps?:\/\/)|(\bwww\.))\S*/
// find a match for the regular expression in the string
var url = text.match(urlRegex);
alert(url[0]);          // www.missingmanuals.com
```

global search for a URL

```
// create a variable containing a string with a URL
var text='there are a lot of great web sites like
    www.missingmanuals.com and http://www.oreilly.com';
// create a regular expression with global search
var urlRegex = /((\bhttps?:\/\/)|(\bwww\.))\S*/g
// find a match for the regular expression in the string
var url = text.match(urlRegex);
alert(url[0]);   // www.missingmanuals.com
alert(url[1]); // http://www.oreilly.com
```

10.28.2008➔ 10/28/2008

string.replace(regex,'replace symbol');

```
1 var date='10.28.2008';   // a string
2 var replaceRegex = /\./g          // a regular expression
3 var date = date.replace(replaceRegex, '/');   // replace . with /
4 alert(date);                      // 10/28/2008
```