

```
1  HTML(Hyper Text Markup Language)
2  =====
3  > HTML (Hypertext Markup Language) IS describes the structure of Web
   pages.
4
5  > With HTML you can create your own Web site.
6
7  > HTML is not a programming language.
8
9  > Markup languages are designed for the processing,definition and
   presentation of text.
10
11 Note:
12 > HTML is the only language that Browser understands.
13 > HTML is error free language.
14
15 Structure of html
16 =====
17
18 <!DOCTYPE html>
19 <html>
20     <head>
21
22         <title>Title of the document</title>
23     </head>
24
25     <body>
26         Content of the document.....
27     </body>
28 </html>
29
30
31 Note: anything which is written inside <> we call as tag.
32
33 # <!DOCTYPE html>
34     declaration defines this document to be HTML5.
35
36 # <html>
37     it is the root tag of an HTML page.
38
39 # <head>
40     tag contains meta information about the document.
41
42 # <title>
43     tag specifies a title for the document.
44
45 # <body>
46     tag contains the visible page content.
47
48
49 The main parts of our element are:
50 =====
51 > The opening tag:
52 =====
53 This consists of the name of the element wrapped in opening and
   closing angle brackets(<>).
54
55 > The closing tag:
56 =====
```

```

57 This is the same as the opening tag, except that it includes a
58 forward slash before
59 the element name.
60 > The content:
61 =====
62 This is the content of the element, which in this case is just text.
63
64 > The element:
65 =====
66 The opening tag plus the closing tag plus the
67 content equals the element.
68
69
70 HTML Basic Tags
71 =====
72 # <h1> to <h6>: Defines HTML headings
73 # <p>: Defines a paragraph
74     <p>this is paragraph</p>
75 # <br>: Inserts a single line break
76 # <hr>: to draw horizontal line
77 # <!--This is a comment-->: Defines a comment
78 # <b>: Defines bold text
79 # <strong>: Defines important text
80 # <u>: underlines the content
81 # <i>: defines in italics.
82 # <em>: Defines emphasized text
83 # <pre>: Defines preformatted text.
84 # <sub>: Defines subscripted text
85 # <sup>: Defines superscripted text
86 # <mark>:Marked text
87 # <del>:Deleted text
88 # <small> - Small text
89 # <marquee>: it is used for scrolling piece of text.
90 # <bdo> element defines bi-directional override.
91
92 character Entities
93 =====
94     non-breaking space           &nbsp;
95 < less than                     &lt;
96 > greater than                 &gt;
97 & ampersand                    &amp;
98 " double quotation mark       &quot;
99 ' single quotation mark       &apos;
100 ¢ cent                         &cent;
101 £ pound                       &pound;
102 ¥ yen                        &yen;
103 € euro                     &euro;
104 © copyright                  &copy;
105 ® registered trademark       &reg;
106 ♥ black heart                &hearts;
107 ♦ black diamond              &diams;
108 ♣ black clubs                &clubs;
109 ♠ black spade                &spades;
110
111 =====
112 ∇ &forall; FOR ALL
113 ∂ &part; PARTIAL DIFFERENTIAL
114 ∇ &nabla; NABLA

```

115	\prod	∏	N-ARY PRODUCT
116	\sum	∑	N-ARY SUMMATION
117	TM	™	TRADEMARK
118	←	←	LEFTWARDS ARROW
119	↑	↑	UPWARDS ARROW
120	→	→	RIGHTWARDS ARROW
121	↓	↓	DOWNWARDS ARROW
122	A	Α	GREEK CAPITAL LETTER ALPHA
123	B	Β	GREEK CAPITAL LETTER BETA
124	Γ	Γ	GREEK CAPITAL LETTER GAMMA
125	Δ	Δ	GREEK CAPITAL LETTER DELTA
126	E	Ε	GREEK CAPITAL LETTER EPSILON
127	Z	Ζ	GREEK CAPITAL LETTER ZETA
128	=====		

```

129
130 Attributes:
131 =====
132 contain extra information about the element.
133
134 An attribute should always have:
135 =====
136 A space between it and the element name.
137 The attribute name, followed by an equals sign.
138 Opening and closing quote marks wrapped around the attribute value.
139
140 EX:<a href="filepath"> HELLO</a>
141 ====
142 Lists
143 =====
144 >There are two types of list
145     -Ordered Lists
146     -Unordered Lists
147
148 Ordered list (<ol>)
149 =====
150 # <ol> element represents an ordered list of items.
151 # also called as Numbered Lists.
152
153 type:-
154 Indicates the numbering type:
155     'a' indicates lowercase letters,
156     'A' indicates uppercase letters,
157     'i' indicates lowercase Roman numerals,
158     'I' indicates uppercase Roman numerals,
159     and '1' indicates numbers (default).
160
161 example:
162 =====
163 <ol>
164     <li>first item</li>
165     <li>second item</li>
166     <li>third item</li>
167 </ol>
168 Unordered Lists(<ul>)
169 =====
170 > <ul> element represents an unordered list of items,
171 > also called as bulleted list.
172
173 type:-

```

```
174 Indicates the bulleting type:
175     -circle,
176     -disc,
177     -square.
178 example:
179 =====
180 <ul>
181     <li>first item</li>
182     <li>second item</li>
183     <li>third item</li>
184 </ul>
185
186 Image
187 =====
188 ->You can insert any image in your web page by using "<img>" tag.
189 ->it can contain only list of attributes and it has no closing tag.
190 syntax
191 =====
192 <img src = "Image URL" ... attributes-list/>
193 Example
194 =====
195 
197 Marquee
198 =====
199 An HTML marquee is a scrolling piece of text displayed either
200 horizontally across or vertically down your webpage depending
201 on the settings.
202
203 Note
204 =====
205 The <marquee> tag deprecated in HTML5.
206
207 Syntax
208 =====
209 <marquee>text message or image</marquee>
210
211 Attribute & Description
212 =====
213 direction="value"
214     This value can be:-up, down, left or right.
215
216 width="value"
217     This value can be:-10 or 20%.
218
219 Links
220 =====
221 ->links that take you directly to other pages .
222 ->These links are known as hyperlinks.
223 -> We can create hyperlinks using text or images available
224 ->When you move the mouse over a link, the mouse arrow will
225     turn into a little hand.
226 ->A link is specified using HTML tag "<a>"
227 ->This tag is called "anchor tag "
228 ->anything between the opening <a> tag and the closing </a>
229 tag becomes part of the link
230 syntax
231 =====
232 <a href = "give URL">Text to Click</a>
```

```

233 Image Links
234 =====
235 <a href = "Give Url">
236     <img src = "image">
237 </a>
238
239
240 Tables
241 =====
242 -> tables are created using the "<table>" tag.
243 ->"<th>" tag is used to create table heading.
244 -> "<tr>" tag is used to create table rows.
245 ->"<td>" tag is used to create data cells.
246 <table border = "1">
247     <tr>
248         <th>Items</th>
249         <th>Price</th>
250     </tr>
251     <tr>
252         <td>1stRow  1stcolumn</td>
253         <td>1stRow, 2ndColumn</td>
254     </tr>
255     <tr>
256         <td>2ndRow, 1stColumn</td>
257         <td>2ndRow , 2ndColumn </td>
258     </tr>
259 </table>
260 Cellpadding
261 =====
262 ->The cellpadding attribute specifies the space, in pixels,
263 between the cell wall and the cell content.
264
265 Eample
266 -----
267 <table cellpadding="2">
268 Cellspacing
269 =====
270 ->The cellspacing attribute specifies the space, in pixels,
271 between cells.
272 Note: Do not confuse this with the cellpadding attribute,
273 which specifies the space between the cell wall and the cell
274 content.
275 Example
276 =====
277 <table cellspacing="3">
278 <table border = "1" cellpadding = "5" cellspacing = "5">
279     <tr>
280         <th>Name</th>
281         <th>Salary</th>
282     </tr>
283 </table>
284 Colspan and Rowspan
285 =====
286 ->colspan attribute use to merge two or more columns into a
287 single column.
288 ->rowspan attribute use to merge two or more rows.
289 Example
290 =====
291 <table border = "1" bordercolor = "red" bgcolor = "blue">

```

```

292         <tr>
293             <th>1stColumn</th>
294             <th>2ndColumn</th>
295             <th>3Column</th>
296         </tr>
297         <tr>
298             <td rowspan = "2">1stRow 1stCell </td>
299             <td>1stRow 2ndCell</td>
300             <td>1stRow 3rdCell</td>
301         </tr>
302         <tr>
303             <td colspan = "3">Row 3 Cell 1</td>
304         </tr>
305     </table>
306 Table Height and Width
307 =====
308 Ex= <table border = "1" width = "100" height = "120">
309     Caption
310     =====
311     ->The caption tag will serve as a title of table
312     ->This tag is deprecated in newer version of HTML/XHTML.
313 Example
314 =====
315     <table border = "1" width = "100%">
316         <caption>Time table</caption>
317
318
319 Forms
320 =====
321 HTML forms are a very powerful tool for interacting with users;
322 "<form>" element is used to collect user input:
323 <form>
324 form elements
325 </form>
326 form Element
327 =====
328 "<input>" element is the most important form element.
329
330 ->we can displayed in several ways, depending on the "type" attribute.
331
332 Example
333 =====
334 Type                Description
335 =====
336 <input type="text">    Defines a one-line text input field.
337 <input type="radio">    Allowing a single value to be selected out
338 of multiple choices.
339 <input type="checkbox">    Defines a checkbox.select ZERO or MORE
340 options.
341 <input type="submit">    Defines a submit button.
342
343 "<select>" Element is used to drop down list.
344 =====
345 <option>" elements defines an option that can be selected.
346 ->To define a pre-selected option, add the "selected"attribute to
347 the option:
348 ->Use the "size"attribute to specify the number of visible values.
349 -> to select more than one value use "multiple" attribute.

```

```
348 "<u>textarea</u>" Element represents a multi-line plain-text editing
control.
349 =====
350 ->"rows" attribute specifies the visible number of lines in a text
area.
351 ->"cols" attribute specifies the visible width of a text area.
352


---


353 "<u>fieldset</u>" element is used to group related data in a form.
354
355 "<legend>" element defines a caption for the <u>fieldset</u> element.
356


---


357 "<button>" element defines a clickable button:
358
359
360 Example
361 =====
362 <body>
363 <form>
364     <u>fieldset</u>
365         <legend>Personal information:</legend>
366         First name:<br>
367         <input type="text" name="firstname" value="Fname"><br>
368         Last name:<br>
369         <input type="text" name="lastname" value="Lname"><br><br>
370     </u>fieldset>
371
372     <input type="radio" name="institute" value="jspider" checked>
jspider<br>
373     <input type="radio" name="institute" value="Qspiders"> Qspiders<br>
374     <input type="checkbox" name="Sub1" value="java"> I done java<br>
375     <input type="checkbox" name="Sub2" value="j2ee"> I done j2ee
376
377     <select name="text" size="2" multiple>
378     <option value="value1">Value 1</option>
379     <option value="value2" selected>Value 2</option>
380     <option value="value3">Value 3</option>
381     Hold down the Ctrl (windows) button to select multiple options.
382 </select>
383
384 <u>textarea</u> name="textarea" rows="10" cols="30">
385 Write something here.
386 </u>textarea>
387
388 <button type="button" onclick="alert('Hello Shekar!')">Click
Me!</button>
389
390 <input type="submit" value="Submit">
391 <input type="reset" value="Reset">
392 </form>
393 </body>
394 =====
395 =====
396
397
398
```

```
399 File Upload Box
400 =====
401 ->If you want to allow a user to upload a file to your web site.
402 ->This is also created using the "<input>" element but
403 type attribute is set to file.
404 Example
405 =====
406 <input type = "file" name = "fileupload" accept = "image/*" />
407 name:->Used to give a name to the control which is sent to the
408 server to be recognized and get the value.
409 accept:->Specifies the types of files that the server accepts.
410


---


411 <input type="color"> is used for input fields that should
412 contain a color.
413
414 Note: type="color" is not supported in Internet Explorer 11
415 and earlier versions or Safari 9.1 and earlier versions.
416
417 Example
418 =====
419 <form>
420     Select your favorite color:
421     <input type="color" name="favcolor">
422 </form>
423


---


424
425 "<input type="date"> "is used for input fields that should
426 contain a date.
427 Note: type="date" is not supported in Internet Explorer 11
428 and earlier versions.
429 Example
430 =====
431 <form>
432 Birthday:
433 <input type="date" name="bday">
434 </form>
435


---


436 "<input type="datetime-local"> "specifies a date and time input field,
437
438 Note: type="datetime-local" is not supported in Firefox,
439 or Internet Explorer 12 and earlier versions.
440 Example
441 =====
442 Joiningday (date and time):
443     <input type="datetime-local" name="jointime">
444


---


445 <input type="email">is used for input fields that should
446 contain an e-mail address.
447 Example
448 =====
449 E-mail:
450 <input type="email" name="email">
451


---


452 <input type="month"> allows the user to select a month and year.
```



```
453
454 Note: type="month" is not supported in Firefox, or Internet
455 Explorer 11 and earlier versions.
456 Example
457 =====
458 Joiningday(month and year):
459 <input type="month" name="jdaymonth">
460


---


461 <input type="number"> defines a numeric input field.
462
463 We can set restrictions on what numbers are accepted.
464 Example
465 =====
466 Quantity (between 1 and 10):
467 <input type="number" name="quantity" min="1" max="10">
468 Quantity:
469 <input type="number" name="points" min="0" max="100" step="10"
470 value="30">
471


---


472 <input type="range"> defines a control for entering a number
473 whose exact value is not important.
474 Example
475 =====
476 <input type="range" name="points" min="0" max="10">
477
478


---


479 <input type="search"> is used for search fields.
480 ->a search field behaves like a regular text field
481 Example
482 =====
483 Search Google:
484 <input type="search" name="googlesearch">
485


---


486 <input type="tel"> is used for input fields that should
487 contain a telephone number.
488
489 Note: type="tel" is only supported in Safari 8 and newer
490 versions.
491 Example
492 =====
493 Telephone:
494 <input type="tel" name="usrtel">
495


---


496 <input type="time"> allows the user to select a time
497 (no time zone).
498 Example
499 =====
500 Select a time:
501 <input type="time" name="usr_time">
502


---


503 <input type="url"> is used for input fields that should contain
504 a URL address.
505 Example
```

```
506 =====
507 Add your homepage:
508 <input type="url" name="homepage">
509


---


510 <input type="week"> allows the user to select a week and year.
511 Note: type="week" is not supported in Firefox, or Internet
512 Explorer 11 and earlier versions.
513 Example
514 =====
515 Select a week:
516 <input type="week" name="year_week">
517


---


518 "size" attribute specifies the size for the input
519 field(in characters).
520 First name:<br>
521 Ex:<input type="text" name="firstname" value="shekar Gowda"
522 size="40">
523


---


524 "maxlength" attribute specifies the maximum length for the
525 input field:
526 Ex:<input type="text" name="firstname" maxlength="15">
527


---


528 "height" and "width" attributes specify the height and width of
529 an <input type="image"> element.
530 Ex:<input type="image" src="a.gpg" alt="Submit" width="48"
531 height="48">
532


---


533 "placeholder" attribute specifies a hint that describes the
534 expected value of an input field
535 ->The hint is displayed in the input field before the user
536 enters a value.
537 -> its works with the following input types: text, search,
538 url, tel, email, and password.
539 Ex:<input type="text" name="fname" placeholder="First name">
540


---


541 "required" attribute specifies that an input field must be
542 filled out before submitting the form.
543 -> its works with the following input types: text, search,
544 url, tel, email, password, date pickers, number, checkbox,
545 radio, and file.
546 Ex: Username: <input type="text" name="username" required>
547


---


548 "step" attribute specifies the legal number intervals for an
549 <input> element
550 -> its works with the following input types: number, range,
551 date, datetime-local, month, time and week
552
553
554
555 Attribute Options Function
556 =====
557 align
```

```

558  =====
559  right, left, center Horizontally aligns tags
560
561  valign
562  =====
563  top, middle, bottom Vertically aligns tags within
564  an HTML element.
565  bgcolor
566  =====
567  numeric, hexidecimal, RGB values Places a background
568  color behind an element
569  background:-URL Places a background image behind an element
570  id
571  =====
572  User Defined      Names an element for use with
573  Cascading Style Sheets.
574  class
575  =====
576  User Defined Classifies an element for use
577  with Cascading Style Sheets.
578  width
579  =====
580  Numeric Value     Specifies the width of tables,
581  images, or table cells.
582  height
583  =====
584  Numeric Value     Specifies the height of tables,
585  images, or table cells.
586  title:- User Defined"Pop-up" title of the elements.
587
588
589  =====
590  <audio> files could be played in a browser
591  Example
592  =====
593  <audio src="path /good_enough.mp3" controls>
594  </audio>
595  OR
596  <audio controls>
597  <source src="aa.mp3" type="audio/mpeg">
598  </audio>
599  -> "controls" attribute adds audio controls, like play, pause, and
600  volume.
601  "source"Defines multiple media resources for media elements, such as
602  <video> and <audio>.
603  video
604  =====
605  <video src="path pass-countdown.mp4" width="170" height="85" controls>
606  </video>
607  OR
608  <video width="320" height="240" controls>
609  <source src="movie.mp4" type="video/mp4">
610  <source src="movie.ogg" type="video/ogg">
611  </video>
612  iframe
613  =====
614  <iframe> tag is used to specify an inline frame
615  Example

```

```

614  =====
615  <iframe src="/html_iframe_tag_example.html" name="iframe_1"
    width="150" height="150"></iframe>
616
617  "src" Location of the frame contents OR URL (web address) of the
    inline frame page.
618
619  "name" Assigns a name to a frame. This is useful for loading contents
    into one frame from
620  another.
621
622  "width" "height" Specifies the width and height of the inline frame.
623
624  =====
625  CSS
626  =====
627  CSS stands for Cascading Style Sheets
628  =====
629  >CSS describes how HTML elements are to be displayed on screen.
630  >CSS is used to define styles for your web pages, including the
    design, layout.
631
632  CSS Syntax
633  =====
634  >CSS rule-set consists of a "selector" and a "declaration" block:
635  >p
636  {
637      color: red;
638      text-align: center;
639  }
640
641  > selector points to the HTML element you want to style.
642  >declaration block contains one or more declarations separated by
    semicolons.
643  >declaration blocks are surrounded by curly braces . declaration
    always ends with a semicolon.
644  Example
645  =====
646  h1
647  {
648      color: blue;
649      background-color: yellow;
650      border: 1px solid black;
651  }
652  p
653  {
654      color: red;
655  }
656
657  Selectors
658  =====
659  >selectors are used to select HTML elements based on their element
    name, id, class,
660  attributes.
661  >selector selects elements based on the "element name".
662  Example
663  =====
664  p
665  {

```

```
666     color: red;
667 }
668


---


669 "id selector"uses the id attribute of an HTML element to select a
670 specific element.
671 >id of an element should be unique within a page, so the id selector
672 is used to select one
673 unique element.
674 >To select an element with a specific id, write a hash (#) character,
675 followed by the id of the element.
676 Note: An id name cannot start with a number.
677 Example
678 =====
679 #fid1
680 {
681     text-align: center;
682     color: red;
683 }
684
685 class Selector
686 =====
687 >class selector selects elements with a specific class attribute.
688 >To select elements with a specific class, write a period (.)
689 character
690 followed by the name of the class
691 Note: A class name cannot start with a number!
692 Example
693 =====
694 .center
695 {
696     text-align: center;
697     color: red;
698 }
699
700 You can also specify that only specific HTML elements should be
701 affected by a class.
702 p.center
703 {
704     text-align: center;
705     color: red;
706 }
707
708 Grouping Selectors
709 =====
710 >If you have elements with the same style definitions
711 >It will be better to group the selectors, to minimize the code.
712
713 how to Insert CSS to HTML ?
714 =====
715 3 different ways to apply CSS to an HTML.
716
717 1)Internal style sheet.
718
719     2)Inline style sheet.
720
721 3)External style sheet.
722


---


723 1)Internal style sheet.
724 =====
```

719 >An internal style sheet may be used if one single page
720 >Internal styles are defined within the <style> element,
721 inside the <head> section of an HTML page:

```
722
723 Example
724 =====
725 <html>
726 <head>
727   <style>
728     h1
729     {
730       color: maroon;
731       margin-left: 40px;
732     }
733   </style>
734 </head>
735 <body>
736
737   <h1>This is a heading</h1>
738   <p>This is a paragraph.</p>
739
740 </body>
741 </html>
```

743 Note: Do not add a space between the property value and unit
744 ->Like margin-left: 20 px; (its not works).
745 ->The correct way is: margin-left: 20px;

747 2) Inline style sheet.

```
748 =====
749 >An inline style may be used to apply a unique style for a single
750 element
751 >add the style attribute to the relevant element.
752 >The style attribute can contain any CSS property.
753 Example
754 =====
755 <h1 style="color:blue;margin-left:30px;">This is a heading</h1>
```

756 3) External Style Sheet

```
757 =====
758 >With an external style sheet, you can change the look of an entire
759 website by changing
760 just one file!
761 >Each page must include a reference to the external style sheet file
762 inside the
763 <link> element.
764 The <link> element goes inside the <head> section:
765 >style sheet file must be saved with a .css extension.
766 >The file should not contain any html tags.
```

```
767 Example
768 =====
769 <head>
770   <link rel="stylesheet" type="text/css" href="first.css">
771 </head>
772
773 cssFile.css
```

```

772  =====
773  body
774  {
775      background-color: lightblue;
776  }
777
778  h1
779  {
780      color: navy;
781      margin-left: 20px;
782  }
783
784
785  colors
786  =====
787  Colors are specified using predefined
788  color names, OR
789  RGB, OR
790  HEX, OR
791  HSL, OR
792  RGBA, OR
793  HSLA values.
794  using names
795  =====
796  Example
797  =====
798  <h1 style="color:Tomato;">Hello World</h1>
799  <h1 style="background-color:DodgerBlue;">Hello HTML</h1>
800  <h1 style="border:2px solid Violet;">Hello CSS</h1>
801


---


802  Using RGB values
803  =====
804  rgb(red, green, blue).
805  parameter defines the intensity of the
806  color between 0 and 255.
807  Example
808  =====
809  <h1 style="background-color:rgb(255, 99, 71);">Shekar</h1>
810


---


811  using a hexadecimal value(HEX)
812  =====
813  #rrggbb;
814  Where rr (red), gg (green), bb (blue) are hexadecimal values
815  between 00 and ff (same as decimal 0-255).
816
817  Example
818  =====
819  <h1 style="background-color:#ff0000;">my Name is Red</h1>
820


---


821  HSL Value
822  =====
823  hue, saturation, and lightness (HSL)
824  hue:->Green, orange, yellow, and blue – each of these is a hue,
825
826  Saturation:-> is also a percentage. 0% means a shade of gray,
827  and 100% is the full color.

```

828 >Saturation can be describe as the intensity of a color.
829 100% is pure color, no shades of gray
830 50% is 50% gray, but you can still see the color.
831 0% is completely gray, you can no longer see the color.
832
833 Lightness:-> is also a percentage.0% is black, 50% is neither light
or dark,
834 100% is white
835 >lightness of a color can be described as how
836 much light you want to give the color,
837
838 Example
839 =====
840 <h1 style="background-color:hsl(147, 50%, 47%);"> my HSL color</h1>
841
842

843 RGBA Value
844 =====
845 >RGBA color values are an extension of RGB color values with an
alpha channel
846 >An RGBA color value is specified with:
847
848 rgba(red, green, blue, alpha)
849 > alpha parameter is a number between 0.0 (fully transparent)
850 and 1.0 (not transparent at all):
851 Example
852 =====
853 <h1 style="background-color:rgba(255, 99, 71, 0.4);"> my color is
RGBA</h1>
854

855 HSLA Value
856 =====
857 >HSLA color values are an extension of HSL color values with
858 an alpha channel - which specifies the opacity for a color.
859 HSLA color value is specified with:
860
861 hsla(hue, saturation, lightness, alpha)
862 Example
863 =====
864 <h1 style="background-color:hsla(9, 100%, 64%, 0.8);">my color is
hsla</h1>
865

866
867 Backgrounds
868 =====
869 CSS background properties:
870
871 background-color
872 background-image
873 background-repeat
874 background-attachment
875 background-position
876

877 background-color: specifies the background color of an element.
878 =====


```
879 Example
880 =====
881 body
882 {
883     background-color: lightblue;
884 }
885
```

886 Background Image

887 =====

888 >background-image property specifies an image to use as the
background of an element.
889 >By default, the image is repeated so it covers the entire element.
890 >By default, the background-image property repeats an image both
horizontally and vertically.

891
892 Example
893 =====

```
894 body
895 {
896     background-image: url("imgpath.jpg");
897 }
898  
899 >TO repeat an image horizontally.
900 >set background-repeat: repeat-x;
901  
902 >To repeat an image vertically,
903 >set background-repeat: repeat-y;
904  
905 >Showing the background image only once;
906 >background-repeat: no-repeat;
907 attachment
908
```

909
910 >position of the image is specified by the background-position
property:

911 Example
912 =====

```
913 background-position: right top;
914
```

915 Background Image - Fixed position

916 =====

917 Example
918 =====

```
919 background-attachment: fixed;
920
```

921
922 Shorthand property
923 =====

```
924 body {
925     background: #ffffff url("img.png") no-repeat right top;
926 }
927
```

927 Multiple Backgrounds

928 =====

929 #example1

```
930 {
931     background-image: url(img_flwr.gif), url(paper.gif);
932     background-position: right bottom, left top;
933 }
```

```

933     background-repeat: no-repeat, repeat;
934 }
935
936 shorthand property:
937 =====
938 Example
939 =====
940 #example1
941 {
942     background: url(img_flwr.gif) right bottom no-repeat,
943                url(paper.gif) left top repeat;
944 }
945 Background Size
946 =====
947 to specify the size of background images.
948 The two other possible values for background-size are contain and
949 cover.
950
951 contain keyword scales the background image to be as large as
952 possible
953 cover keyword scales the background image so that the content area
954 is completely
955 covered by the background image.
956 #div1
957 {
958     background-size: contain;
959 }
960 #div2
961 {
962     background-size: cover;
963 }
964 Borders
965 =====
966 >border properties allow you to specify the
967 style, width, and color
968 Border Style
969 =====
970 Example
971 =====
972 <style>
973 .solid {border-style: solid;}
974 .dotted {border-style: dotted;}
975 .dashed {border-style: dashed;}
976 .inset {border-style: inset;}
977 .outset {border-style: outset;}
978 .double {border-style: double;}
979 .groove {border-style: groove;}
980 .ridge {border-style: ridge;}
981 .none {border-style: none;}
982 .hidden {border-style: hidden;}
983 .mix {border-style: dotted dashed solid double;}
984 </style>

```

```

983 border-width
984 =====
985 border-width property can have from one to four values,

```

986 (top border, right border, bottom border,left border).

987 Example

988 =====

```
989 p{
990     border-style: solid;
991     border-width: 3px 4px 5px 150px;
992 }
993
```

994 Border Color

995 =====

996 border-color property can have from one to four values ,
997 (top border, right border, bottom border, left border).

998 Example

999 =====

```
1000 p{
1001     border-style: solid;
1002     border-color: red green blue yellow;
1003 }
1004
```

1005 border sides

1006 =====

1007 there are also properties for specifying each of the borders
1008 (top, right, bottom, and left):

1009 Example

1010 =====

```
1011 p {
1012     border-top-style: dotted;
1013     border-right-style: solid;
1014     border-bottom-style: dotted;
1015     border-left-style: solid;
1016 }
1017
```

1018 Shorthand Property

1019 =====

1020 shorthand property for the following individual border properties:

1021 >border-width

1022 >border-style (required)

1023 >border-color

1024 Example

1025 =====

```
1026 p{
1027     border: 5px solid red;
1028 }
1029
```

1030 we can specify all the individual border properties for just one
1031 side:

1032 Example

1033 =====

```
1034 p
1035 {
1036     border-left: 6px solid red;
1037     background-color: lightgrey;
1038 }
```

```
1039 border-radius property is used to add rounded borders to an element:
1040 Example
1041 =====
1042 p
1043 {
1044     border: 2px solid red;
1045     border-radius: 5px;
1046 }
1047
```

```
1048
1049 Margins
1050 =====
1051 margin properties are used to create space around elements,
1052 outside of any defined borders.
```

Property	Description
margin-bottom	Sets the bottom margin of an element
margin-left	Sets the left margin of an element
margin-right	Sets the right margin of an element
margin-top	Sets the top margin of an element

1059 Example

1060 =====

```
1061 p
1062 {
1063     margin-top: 100px;
1064     margin-bottom: 100px;
1065     margin-right: 150px;
1066     margin-left: 80px;
1067 }
1068 Shorthand Property
1069 =====
1070 p
1071 {
1072     margin: 25px 50px 75px 100px;
1073 }
1074
```

1075 padding

1076 =====

1077 padding properties are used to generate space around an element's
1078 content,
1079 inside of any defined borders.

Property	Description
padding-top	Sets the top padding of an element.
padding-right	Sets the right padding of an element.
padding-bottom	Sets the bottom padding of an element.
padding-left	Sets the left padding of an element.

1085 Example

1086 =====

```
1087 div
1088 {
1089     padding-top: 50px;
1090     padding-right: 30px;
1091     padding-bottom: 50px;
1092     padding-left: 80px;
1093 }
```

1094 Shorthand Property


```
1150 {
1151     text-decoration: underline;
1152 }
1153
1154 h1
1155 {
1156     text-transform:uppercase;
1157 }
1158
1159 h1
1160 {
1161     text-transform: capitalize;
1162 }
1163 p
1164 {
1165     text-indent: 50px;
1166 }
1167 h1
1168 {
1169     letter-spacing: 3px;
1170 }
1171 p
1172 {
1173     direction: rtl;
1174 }
1175 h1
1176 {
1177     word-spacing: 10px;
1178 }
1179 h1
1180 {
1181     text-shadow: 3px 2px red;
1182 }
1183
1184
```

```
1185 links
1186 =====
1187 Example
1188
1189 a{
1190 color:red;
1191 }
1192 styled differently depending on what state they are in
1193 =====
1194 a:link - a normal, unvisited link
1195 a:visited - a link the user has visited
1196 a:hover - a link when the user mouses over it
1197 a:active - a link the moment it is clicked
1198
1199 unvisited link
1200 =====
1201 a:link
1202 {
1203     color: red;
1204 }
1205
1206 visited link
1207 =====
```

```
1208 a:visited {
1209     color: green;
1210 }
1211
1212 mouse over link
1213 =====
1214 a:hover {
1215     color: hotpink;
1216 }
1217
1218 selected link
1219 =====
1220 a:active
1221 {
1222     color: blue;
1223 }
1224
1225 list in css
1226 =====
1227 list-style-type property specifies the type of list item marker.
1228
```

```
1229 Example
1230 =====
1231 <head>
1232 <style>
1233 .au
1234 {
1235     list-style-type: circle;
1236 }
1237
1238 .bu
1239 {
1240     list-style-type: square;
1241 }
1242
1243 .c
1244 {
1245     list-style-type: upper-roman;
1246 }
1247
1248 .ao
1249 {
1250     list-style-type: lower-alpha;
1251 }
1252 </style>
1253 </head>
1254 <body>
1255 <p>unordered lists:</p>
1256 <ul class="au">
1257     <li>pani puri</li>
1258     <li>masala puri</li>
1259     <li>kali puri</li>
1260 </ul>
1261
1262
1263 <p> ordered lists:</p>
1264 <ol class="ao">
1265     <li>java</li>
```

```
1266     <li>j2ee</li>
1267     <li>Web</li>
1268 </ol>
1269
1270 </body>
1271 </html>
1272
1273
1274 Tables
1275 =====
1276 >border-collapse property sets table borders should be collapsed
1277 into a single border.
1278 Example
1279 =====
1280 table
1281 {
1282     border-collapse: collapse;
1283 }
1284 Horizontal Alignment
1285 =====
1286 text-align property sets the horizontal alignment
1287 th
1288 {
1289     text-align: left;
1290 }
1291 Vertical Alignment
1292 =====
1293 vertical-align property sets the vertical alignment
1294 Example
1295 =====
1296 td
1297 {
1298     height: 50px;
1299     vertical-align: top;
1300 }
1301


---


1301 border-bottom property to <th> and <td> for horizontal dividers.
1302 Example
1303 =====
1304 th, td
1305 {
1306     border-bottom: 1px solid red;
1307 }
1308
1309 Hoverable Table
1310 =====
1311 >hover selector on <tr> to highlight table rows on mouse over.
1312 tr:hover
1313 {
1314     background-color: green;
1315 }
1316


---


1317 use the nth-child() selector and add a background-color to all even
1318 or odd table rows.
1319 tr:nth-child(even)
1320 {
1321     background-color: yellow;
```



```
1321 }
1322
1323
1324
1325 display Property
1326 =====
1327 >display property is the most important CSS property.
1328 >It specifies how the element is displayed.
1329 >The default display value for most elements is block or inline.
1330 Block-level Elements
1331 =====
1332 block-level element always starts on a new line.takes up the full
width available.
1333 Example
1334 =====
1335 <div>
1336 <h1>____<h6>
1337 <p>
1338 <form>
1339 <header>
1340 <footer>
1341 Inline Elements
1342 =====
1343 inline element does not start on a new line and only takes up as
much width as necessary.
1344 <span>
1345 <a>
1346 <img>
1347 Example
1348 =====
1349 li
1350 {
1351     display: inline;
1352 }
1353


---


1354 Hiding an element can be done by setting the display property to none.
1355 Example
1356 =====
1357 .hidden
1358 {
1359     display: none;
1360 }
1361


---


1362 visibility:hidden;
1363 >also hides an element.
1364
1365 >The element will still take up the same space as before.
1366 >The element will be hidden, but still affect the layout.
1367 Example
1368 =====
1369 .hidden
1370 {
1371     visibility: hidden;
1372 }
1373


---


1374 position
```

```
1375  =====
1376  position property specifies the type of positioning method used for
      an element.
1377
1378  We have five different position values.
1379  =====
1380  ->static
1381  ->relative
1382  ->fixed
1383  ->absolute
1384  ->sticky
1385


---


1386  >Static positioned elements are not affected by the top, bottom,
      left, right properties.
1387  Example
1388  =====
1389  div
1390  {
1391      position: static;
1392      border: 3px solid #73AD21;
1393  }
1394
1395
1396  Combinators
1397  =====
1398  combinator is something that explains the relationship between the
      selectors.
1399  There are four different combinators in CS
1400  =====
1401  1)descendant selector (space)
1402  2)child selector (>)
1403  3)adjacent sibling selector (+)
1404  4)general sibling selector (~)
1405


---


1406  descendant selector
1407  =====
1408  matches all elements that are descendants of a specified element.
1409  div p
1410  {
1411      background-color: yellow;
1412  }
1413


---


1414  child selector
1415  selects all elements that are the immediate children of a specified
      element.
1416  div > p
1417  {
1418      background-color: yellow;
1419  }
1420
1421


---


1422  adjacent sibling selector selects all elements that are the adjacent
      siblings of a
1423  specified element.
1424  and "adjacent" means "immediately following".
```

```
1425 Sibling elements must have the same parent element.
1426 div + p
1427 {
1428     background-color: yellow;
1429 }
1430
```

```
1431 general sibling selector selects all elements that are siblings of a
1432 specified element.
1433 div ~ p
1434 {
1435     background-color: yellow;
1436 }
1437
1438
1439
1440
1441
```

```
1442
1443 JavaScript
1444 =====
1445 >JavaScript is an object-based scripting language that is
1446 lightweight.
1447 >JavaScript to program the behavior of web pages
1448 >JavaScript Can Change HTML Content
1449 Why JavaScript is used
1450 =====
1451 JavaScript is used to create interactive websites.
1452 Example
1453 =====
1454 *Displaying clocks.
1455 *Client-side validation.etc
1456
1457 JavaScript Example
1458 =====
1459 <script>
1460
1461     document.write("Hello JavaScript")
1462
1463 </script>
1464 >script tag specifies that we are using JavaScript.
1465 >document.write() function is used to display dynamic content
1466 through JavaScript.
1467
1468 3 Places to put JavaScript code
1469 =====
1470 Between the body tag of html
1471 Between the head tag of html
1472 In .js file (external javaScript)
1473
1474 Variable
1475 =====
1476 local variable and global variable.
1477 >local variable is declared inside block or function.
1478 It is accessible within the function or block only.
1479
1480 function xyz()
```

```

1480 {
1481 var x=10;//local variable
1482 }
1483
1484 >global variable is declared outside the function
1485 it can be declared inside any function and can be accessed from any
    function.
1486
1487 Data Types
1488 =====
1489 >JavaScript is a dynamic type language, means we don't need to
    specify type of the variable.
1490 >use var here to specify the data type.
1491 >It can hold any type of values such as numbers, strings etc.
1492 Example
1493 =====
1494 var a=420;//holding number
1495 var b="Raju";//holding string
1496
1497 In javaScript there is no default values.
1498 O/P=undefined.
1499
1500
1501
1502 Operators
1503 =====
1504 same operators are present in javaScript as same as a java
1505 But only difference in Relational operator.
1506 == only compares values
1507
1508 === compares values + type
1509
1510 0 == false // true
1511 0 === false // false, because they are of a different type
1512 1 == "1" // true, automatic type conversion for value only
1513 1 === "1" // false, because they are of a different type
1514 null == undefined // true
1515 null === undefined // false
1516 '0' == false // true
1517 '0' === false // false
1518
1519 typeof:->is operator which is used to indicate
1520 the what data type we have used for a variable.
1521 this is present in javaScript.not in java.
1522 Example
1523 =====
1524 document.writeln(typeof(a));
1525 document.writeln(typeof(b));
1526 =====|
1527 Control Statement: ALL ARE SIMILAR TO JAVA.|
1528 =====
1529 Functions
1530 =====
1531 Advantage
1532 -----
1533 Code reusability.
1534 Less coding.
1535 Example
1536 =====

```

```
1537 <script>
1538 function msg()
1539 {
1540 alert("hello! this is message");
1541 }
1542 </script>
1543 <input type="button" onclick=msg() value="call function"/>
1544
1545
1546 Function Return Value.
1547 =====
1548 <script>
1549 function getValue()
1550 {
1551 return "hello shekar How r u?";
1552 }
1553 </script>
1554
1555 JavaScript Objects
1556 =====
1557 >JavaScript is an object-based language.
1558 >Everything is an object in JavaScript.
1559
1560 3 ways to create objects.
1561 =====
1562 1>By object literal
1563
1564 property and value is separated by :(colon).
1565
1566 emp={id:102,name:"shekar",salary:40000};
1567
1568 document.write(emp.id+" "+emp.name+" "+emp.salary);
1569
1570 2>By creating instance of Object directly(using new keyword).
1571
1572 <script>
1573     var emp=new Object();
1574     emp.id=143;
1575     emp.name="Shekar";
1576     emp.salary=50000;
1577     document.write(emp.id+" "+emp.name+" "+emp.salary);
1578 </script>
1579
1580 3>By using an object constructor (using new keyword)
1581
1582 >create function with arguments.
1583 >Each argument value can be assigned in the current object by using
this keyword.
1584
1585 <script>
1586
1587 function emp(id,name,salary)
1588 {
1589     this.id=id;
1590     this.name=name;
1591     this.salary=salary;
1592 }
1593 e=new emp(103,"Shekar Gowda",30000);
1594
```

```

1595     document.write(e.id+" "+e.name+" "+e.salary);
1596
1597 </script>
1598
1599
1600 JavaScript Array
1601 =====
1602 3 ways to construct array in JavaScript.
1603 1>By array literal.
1604 <script>
1605     var emp=["Shekar", "Raju", "Kirshna"];
1606     for (i=0;i<emp.length;i++)
1607     {
1608         document.write(emp[i] + "<br/>");
1609     }
1610 </script>
1611 2>By creating instance of Array directly (using new keyword)
1612 <script>
1613     var i;
1614     var emp = new Array();
1615     emp[0] = "Arun";
1616     emp[1] = "Harshit";
1617     emp[2] = "Akash";
1618
1619     for (i=0;i<emp.length;i++)
1620     {
1621         document.write(emp[i] + "<br>");
1622     }
1623 </script>
1624 3>By using an Array constructor (using new keyword)
1625 <script>
1626     var emp=new Array("Priya", "shweta", "Sumera");
1627     for (i=0;i<emp.length;i++)
1628     {
1629         document.write(emp[i] + "<br>");
1630     }
1631 </script>
1632
1633
1634 String
1635 =====
1636 2 ways to create string in JavaScript
1637 1>By string literal
1638 var str="This is javaScript";
1639
1640 2>By string object (using new keyword)
1641
1642 var stringname=new String("hello javascript string");
1643 document.write(stringname);
1644
1645 String Methods
1646 =====
1647 1>charAt(index)
1648 var str="javascript";
1649 document.write(str.charAt(2));
1650
1651 2>concat(str)
1652 var s1="javascript ";
1653 var s2="example";

```

```
1654 var s3=s1.concat(s2);
1655 document.write(s3);
1656
1657 3>indexOf(str)
1658 JavaScript String indexOf(str) method returns
1659 the index position of the given string.
1660
1661 var s1="javascript from jspider";
1662 var n=s1.indexOf("from");
1663 document.write(n);
1664 O/P=11
1665
1666 4>lastIndexOf(str)
1667 The JavaScript String lastIndexOf(str) method returns
1668 the last index position of the given string.
1669
1670
1671 var s1="javascript from java indexof";
1672 var n=s1.lastIndexOf("java");
1673 document.write(n);
1674 O/P=16;
1675
1676 5>toLowerCase()
1677 var s1="JavaScript toLowerCase";
1678 var s2=s1.toLowerCase();
1679 document.write(s2);
1680
1681 6>toUpperCase()
1682 var s1="JavaScript toUpperCase";
1683 var s2=s1.toUpperCase();
1684 document.write(s2);
1685
1686 7>slice(beginIndex, endIndex)
1687 In slice() method, beginIndex is inclusive and endIndex is exclusive.
1688 var s1="abcdefgh";
1689 var s2=s1.slice(2,5);
1690 document.write(s2);
1691 O/P=cde
1692
1693 8>trim()
1694 trim() method removes leading and trailing whitespaces from the
1695 string.
1696 var s1="    I am javascript trim    ";
1697 var s2=s1.trim();
1698 document.write(s2);
1699 o/p=I am javascript trim
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709 Date
1710 =====
1711 Date objects are created with the new Date() constructor.
```

```
1712
1713 4 ways of initiating a date:
1714
1715 1>new Date()
1716 creates a new date object with the current date and time:
1717
1718 Examples
1719 =====
1720 <script>
1721
1722 var d = new Date();
1723 document.getElementById("demo").innerHTML = d;
1724
1725 </script>
1726
1727 2>new Date(milliseconds)
1728 <script>
1729 var d = new Date(1000000000000);
1730 document.getElementById("demo").innerHTML = d;
1731 </script>
1732
1733 3>new Date(dateString)
1734 <script>
1735 var d = new Date("October 13, 2014 11:13:00");
1736 document.getElementById("demo").innerHTML = d;
1737
1738 </script>
1739 4>new Date(year, month, day, hours, minutes, seconds, milliseconds)
1740 <script>
1741 var d = new Date(99, 5, 24, 11, 33, 30, 0);
1742 document.getElementById("demo").innerHTML = d;
1743 </script>
1744
1745
1746
1747
1748
1749
1750
1751
1752 <button
1753   onclick="document.getElementById('demo').innerHTML=Date()">The time
1754   is?</button>
1755
1756
1757 <button onclick="this.innerHTML=Date()">The time is?</button>
1758
1759
1760 Random number
1761 =====
1762 Math.random()
1763 returns a random number between 0 (inclusive), and 1 (exclusive).
1764 >Math.random() always returns a number lower than 1.
1765
1766 Math.floor(Math.random() * 10);
1767 returns a number between 0 and 9
1768
```



```
1769 Math.floor(Math.random() * 11);
1770 returns a number between 0 and 10
1771
1772 Math.floor(Math.random() * 100);
1773 returns a number between 0 and 99
1774
1775 Math.floor(Math.random() * 101);
1776 returns a number between 0 and 100
1777
1778 Math.floor(Math.random() * 10) + 1;
1779 returns a number between 1 and 10
1780
1781 Math.floor(Math.random() * 100) + 1;
1782 returns a number between 1 and 100
1783
1784 Math.round(4.7);
1785 // returns 5
1786 Math.round(4.4);
1787 // returns 4
1788 Math.pow(2, 2);
1789 // returns 64
1790
1791
1792
1793 Example ahdhwefwefrger
1794 =====
1795 <script>
1796
1797
1798 document.write("charAt(2)<br/>");
1799 var str="javascript";
1800 document.write(str.charAt(2));
1801 document.write("<br/>");
1802
1803
1804 document.write("<concat<br/>");
1805 var s1="javascript ";
1806 var s2="example";
1807 var s3=s1.concat(s2);
1808 document.write(s3);
1809 document.write("<br/>");
1810
1811 document.write("IndexOf(str)<br/>");
1812 var s4="javascript from jspider";
1813 var n=s4.indexOf("from");
1814 document.write(n);
1815 document.write("<br/>");
1816
1817
1818 document.write("lastIndexOf(str)<br/>");
1819 var s5="javascript from java indexof";
1820 var n=s5.lastIndexOf("java");
1821 document.write(n);
1822 document.write("<br/>");
1823
1824 document.write("toLowerCase()<br/>");
1825 var s6="JavaScript toLowerCase";
1826 var s7=s6.toLowerCase();
1827 document.write(s7);
```

```

1828     document.write("<br/>");
1829
1830     document.write("toUpperCase()<br/>");
1831     var s8="JavaScript toUpperCase";
1832     var s9=s8.toUpperCase();
1833     document.write(s9);
1834     document.write("<br/>");
1835
1836     document.write("slice()<br/>");
1837     var s10="abcdefgh";
1838     var s11=s10.slice(2,5);
1839     document.write(s11);
1840     document.write("<br/>");
1841
1842     document.write("trim()<br/>");
1843
1844     var s12="    I am javascript trim    ";
1845     var s13=s12.trim();
1846     document.write(s13);
1847     document.write("<br/>");
1848     </script>
1849
1850
1851
1852     Date
1853     =====
1854     <script>
1855     var d = new Date();
1856     document.getElementById("demo").innerHTML = d;
1857
1858
1859
1860     document.write("2>new Date(milliseconds)<br/>");
1861
1862     var d1= new Date(1000000000000);
1863     document.getElementById("demo1").innerHTML = d1;
1864
1865
1866     document.write("3>new Date(dateString)<br/>");
1867
1868     var d2 = new Date("October 13, 2014 11:13:00");
1869     document.getElementById("demo2").innerHTML = d2;
1870
1871     document.write("4>new Date(year, month, day, hours, minutes,
seconds, milliseconds)");
1872
1873     var d3 = new Date(99, 5, 24, 11, 33, 30, 0);
1874     document.getElementById("demo3").innerHTML = d3;
1875     </script>
1876
1877
1878     Browser Object Model (BOM)
1879     =====
1880     is used to interact with the browser.
1881     The default object of browser is window.
1882     window Object is created automatically by the browser.
1883
1884     Methods of window object
1885     =====

```

```
1886
1887 alert()      displays the alert box containing message with ok button.
1888 function msg()
1889 {
1890     alert("Hello Alert Box");
1891 }
1892


---


1893 confirm()      displays the confirm dialog box containing message with
1894 ok and cancel button.
1895 function msg()
1896 {
1897     var v= confirm("Are u sure?");
1898     if(v==true)
1899     {
1900         alert("ok");
1901     }
1902     else
1903     {
1904         alert("cancel");
1905     }
1906 }
1907


---


1908 prompt()       displays a dialog box to get input from the user.
1909 function msg()
1910 {
1911     var v= prompt("What is you age?");
1912     alert("your age is "+v);
1913 }
1914


---


1915 open()         opens the new window.
1916 function msg()
1917 {
1918     open("http://www.jspiders.com");
1919 }
1920


---


1921 setTimeout()   performs action after specified time like calling
1922 function,
1923 evaluating expressions etc.
1924
1925 function msg()
1926 {
1927     setTimeout(
1928     function()
1929     {
1930         alert("Welcome Shekar After 2 seconds")
1931     },2000);
1932 }
1933
1934


---


1935 screen object holds information of browser screen.
1936 It can be used to display screen width, height, colorDepth,
pixelDepth etc.
```

```

1937
1938 Document Object Model(DOM)
1939 =====
1940 >document object represents the whole html document.
1941 Methods of document object
1942 =====
1943 We can access and change the contents of document by its methods.
1944
1945
1946 write("string")           writes the given string on the doucment.
1947 writeln("string")       writes the given string on the doucment
with newline character at
1948                             the end.
1949 getElementById()         returns the element having the given id
value.
1950 getElementsByName()      returns all the elements having the given
name value.
1951 getElementsByClassName() returns all the elements having the given
class name.
1952 getElementsByTagName()   returns all the elements having the given
tag name.
1953
1954 Accessing field value by document object
1955 =====
1956 <script>
1957 function printname()
1958 {
1959 var name=document.form1.name.value;
1960 alert("Welcome: "+name);
1961 }
1962 </script>
1963
1964 <form name="form1">
1965 Enter Name:<input type="text" name="name"/>
1966 <button onclick="printname()">GetValue</button>
1967 </form>
1968
1969
1970 document.getElementById() method
1971 =====
1972 <script>
1973 function cube()
1974 {
1975 var number=document.getElementById("number").value;
1976 alert(number*number*number);
1977 }
1978 </script>
1979 <form>
1980 Enter No:<input type="text" id="number" name="number"/><br/>
1981 <button onclick="cube()">Cube</button>
1982 </form>
1983 document.getElementsByName()
1984 =====
1985 method returns a collection of all elements in the document with the
specified name
1986 as a NodeList object.
1987
1988 The NodeList object represents a collection of nodes.
1989 The nodes can be accessed by index numbers. The index starts at 0.

```

```

1990 function totalelements()
1991 {
1992 var allgenders=document.getElementsByName("gender");
1993 alert("Total Genders:"+allgenders.length);
1994 }
1995 </script>
1996 <form>
1997 Male:<input type="radio" name="gender" value="male">
1998 Female:<input type="radio" name="gender" value="female">
1999
2000 <button onclick="totalelements()">NoOfGenders<button/>
2001
2002 document.getElementsByTagName()
2003 =====
2004 method returns all the element of specified tag name.
2005 <script>
2006 function counth1tag()
2007 {
2008 var totalh1=document.getElementsByTagName("h1");
2009 alert("total h1 tags are: "+totalh1.length);
2010 }
2011 </script>
2012 <h1>This is a h1</h1>
2013 <h1>Here getElementByTagName() method.</h1>
2014 <h1>Let's see the simple example</h1>
2015 <button onclick="counth1tag()">count h1 tag</button>
2016


---


2017 innerHTML property can be used to write the dynamic html on the html
document.
2018 <p id="demo"></p>
2019
2020 <script>
2021 document.getElementById("demo").innerHTML = "Hello World!";
2022 </script>
2023 getElementById is a method, while innerHTML is a property.
2024


---


2025
2026 Events
2027 =====
2028 onblur
2029 =====
2030 When you leave the input field, a function is triggered which
transforms
2031 the input text color to red.
2032 <script>
2033 function myFunction()
2034 {
2035     var x = document.getElementById("fname");
2036     x.style.color = "red";
2037 }
2038 </script>
2039
2040 Enter your name: <input type="text" id="fname" onblur="myFunction()">
2041


---


2042 onfocus
2043 =====

```

```
2044 When the input field gets focus, a function is triggered which
2045 changes the
2046 background-color.
2047 <script>
2048 function myFunction(x)
2049 {
2050     x.style.background = "yellow";
2051 }
2052 </script>
2053 Enter your name: <input type="text" onfocus="myFunction(this)">
2054
2055
2056 <script>
2057 function myFunction()
2058 {
2059     document.getElementById("demo").innerHTML = "You selected some
2060 text";
2061 }
2062 </script>
2063 <body>
2064 Some text: <input type="text" value="Hello world!"
2065 onselect="myFunction()">
2066
2067 <p id="demo"></p>
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
```

```
onsubmit
=====
function confirmInput()
{
    fname = document.forms[0].fname.value;
    alert("Hello " + fname + "! You will now be redirected to
    www.w3Schools.com");
}
</script>
<form onsubmit="confirmInput()" action="https://www.jspiders.com/">
Enter your name: <input id="fname" type="text" size="20">
<input type="submit">

```

```
onkeydown
=====
A function is triggered when the user is pressing a key in the input
field.
function myFunction()
{
    alert("You pressed a key inside the input field");
}
</script>
<input type="text" onkeydown="myFunction()">

```

```
onkeyup
=====
A function is triggered when the user releases a key in the input
field.
```

```
2093 The function transforms the character to upper case.
2094 function myFunction()
2095 {
2096     var x = document.getElementById("fname");
2097     x.value = x.value.toUpperCase();
2098 }n m
2099 </script>
2100
2101 Enter your name: <input type="text" id="fname" onkeyup="myFunction()">
2102
2103


---


2104 function color(color)
2105 {
2106     document.forms[0].myInput.style.background = color;
2107 }
2108 </script>
2109
2110 <form>
2111 Write a message:<br>
2112 <input
2113 type="text"
2114 onkeydown="color('yellow')"
2115 onkeyup="color('white')"
2116 name="myInput">
2117 </form>
2118


---


2119
2120


---


2121 Mouse Events
2122 =====
2123 onmouseover and onmouseout
2124 =====
2125 <h1 onmouseover="style.color='red'"
2126 onmouseout="style.color='black'">Mouse over this text</h1>
2127


---


2128 onmousedown and onmouseup
2129 =====
2130 <script>
2131 function myFunction(element, color)
2132 {
2133     element.style.color = color;
2134 }
2135 </script>
2136 </head>
2137 <body>
2138 <p onmousedown="myFunction(this,'red')"
2139 onmouseup="myFunction(this,'green')">
2140 Click the text to change the color.
2141 </body>
2142


---


2143 <script>
2144 function bigImg(x)
2145 {
```

```

2145         x.style.height = "64px";
2146         x.style.width = "64px";
2147     }
2148     function normalImg(x)
2149     {
2150         x.style.height = "32px";
2151         x.style.width = "32px";
2152     }
2153 </script>
2154 <body>
2155
2156 
2158 </body>
2159


---


2160 Form Validation
2161 =====
2162 validate the name and password. The name can't be empty and
2163 password can't be less than 6 characters long.
2164
2165 function validateNameandPassform()
2166 {
2167     var name=document.myform.name.value;
2168     var password=document.myform.password.value;
2169
2170     if (name==null || name=="")
2171     {
2172         alert("Name can't be blank");
2173         return false;
2174     }else if(password.length<6)
2175     {
2176         alert("Password must be at least 6 characters long.");
2177         return false;
2178     }
2179 }
2180 </script>
2181 <body>
2182 <form name="myform" onsubmit="return validateNameandPassform()" >
2183 Name: <input type="text" name="name"><br/>
2184 Password: <input type="password" name="password"><br/>
2185 <input type="submit" value="register">
2186 </form>
2187 Password Validation
2188 =====
2189 <script>
2190 function matchpass()
2191 {
2192     var firstpassword=document.f1.password.value;
2193     var secondpassword=document.f1.password2.value;
2194
2195     if(firstpassword==secondpassword)
2196     {
2197         return true;
2198     }
2199     else
2200     {
2201         alert("password must be same!");

```



```
2202     return false;
2203 }
2204 }
2205 </script>
2206
2207 <form name="f1" onsubmit="return matchpass()">
2208 Password:<input type="password" name="password" /><br/>
2209 Re-enter Password:<input type="password" name="password2"/><br/>
2210 <input type="submit">
2211 </form>
2212


---


2213 <script>
2214 function validate()
2215 {
2216     var msg;
2217     if(document.myForm.userPass.value.length>6)
2218     {
2219         msg="good";
2220     }
2221     else
2222     {
2223         msg="poor";
2224     }
2225     document.getElementById('mylocation').innerText=msg;
2226 }
2227
2228 </script>
2229 <form name="myForm">
2230 <input type="password" value="" name="userPass"
2231 onkeyup="validate()">
2232 Strength:<span id="mylocation">no strength</span>
2233 </form>
2234


---


2235
2236 Number Validation
2237 =====
2238 <script>
2239 function validate()
2240 {
2241     var num=document.myform.num.value;
2242     if (isNaN(num)) {
2243         document.getElementById("num").innerHTML="Enter Numeric value
2244         only";
2245         return false;
2246     }
2247     else
2248     {
2249         return true;
2250     }
2251 }
2252 </script>
2253 <form name="myform" onsubmit="return validate()" >
2254 Number: <input type="text" name="num"><h1 id="num"></h1><br/>
2255 <input type="submit" value="submit">
2256 </form>
```

```

2257 What Is a Regular Expression?
2258 =====
2259 A regular expression is a sequence of characters that forms a search
    pattern.
2260 Syntax
2261 =====
2262 var reg= /pattern/modifier;
2263
2264 where "pattern" is the regular expression.
2265 and the "modifiers" is optional
2266 Modifiers
2267 =====
2268 i   Perform case-insensitive matching
2269 g   Perform a global match (find all matches rather than stopping
    after the first match)
2270 m   Perform multiline matching
2271
2272 Brackets are used to find a range of characters:
2273 =====
2274 Expression  Description
2275 [a-z]       Find any character between the brackets
2276
2277 [0-9]       Find any character between the brackets (any digit)
2278
2279 (x|y)       Find any of the alternatives specified
2280
2281 Metacharacters are characters with a special meaning:
2282 =====
2283 \d Find a digit 0-9
2284 \w Find a all words and A-z a-z 0-9
2285 \s Find a whitespace character
2286 \b Find a match at the beginning or at the end of a word
2287


---


2288 Quantifiers define quantities:
2289 =====
2290 n+ Matches any string that contains at least one n(1 or more)
2291 n* Matches any string that contains zero or more occurrences of n(0
    or more)
2292 n? Matches any string that contains zero or one occurrences of n(0
    or 1)
2293


---


2294 var regex = /^\\d{2}$/;
2295 The pattern portion above starts with an ^ indicating the beginning
    of a string.
2296 The \\d indicates a digit followed by {2} meaning 2 consecutive digits.
2297 The $ indicates end of a string.
2298 So, this pattern will attempt to find
2299 exactly 2 consecutive digits from the beginning to the end of a
    string.
2300


---


2301
2302 HTML form contains only letters.
2303 =====
2304 var letters = /^[A-Za-z]+$/;
2305


---



```

```

2306 To get a string contains only numbers (0-9)
2307 =====
2308 var dig=/^[0-9]+$/
2309 which allows only numbers.
2310


---


2311 validate a phone number of 10 digits with no comma,
2312 =====
2313 var phoneno = /^[d{10}]$/
2314 permit only phone numbers with 10 digits.
2315


---


2316 contains letters and numbers only
2317 =====
2318 var letterNumber = /^[0-9a-zA-Z]+$/
2319


---


2320 whether an input string is a valid email
2321 =====
2322 /^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$/
2323 OR
2324 (/^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/
2325
2326
2327 Uppercase (A-Z) and lowercase (a-z) English letters.
2328 Digits (0-9).
2329 Characters ! # $ % & ' * + - / = ? ^ _ ` { | } ~
2330 Character . ( period, dot or fullstop) provided that it is not the
2331 first or last
2332 character and it will not come one after the other.
2333 The domain name [for example com, org, net, in, us, info]
2334 part contains letters, digits, hyphens, and dots.
2335
2336 Example of valid email id
2337 =====
2338 mysite@ourearth.com
2339 my.ownsite@ourearth.org
2340 mysite@you.me.net
2341
2342 Example of invalid email id
2343 =====
2344 mysite.ourearth.com [@ is not present]
2345 mysite@.com.my [ tld (Top Level domain) can not start with dot "." ]
2346 @you.me.net [ No character before @ ]
2347 mysite123@gmail.b [ ".b" is not a valid tld ]
2348 mysite@.org.org [ tld can not start with dot "." ]
2349 .mysite@mysite.org [ an email should not be start with "." ]
2350 mysite()@gmail.com [ here the regular expression only allows
2351 character, digit,
2352 underscore, and dash ]
2353 mysite..1234@yahoo.com [double dots are not allowed]
2354
2355 Character Description
2356 =====
2357 / .. / All regular expressions start and end with forward slashes.
2358


---


2358 ^ Matches the beginning of the string or line.

```

2359	\w+ Matches one or more word characters including the underscore.
2360	Equivalent to [A-Za-z0-9_].
2361	
2362	[\\. -] \ Indicates that the next character is special and not to be interpreted literally.
2363	.- matches character . or -.
2364	
2365	? Matches the previous character 0 or 1 time. Here previous character is [.-].
2366	
2367	\w+ Matches 1 or more word characters including the underscore. Equivalent to [A-Za-z0-9_].
2368	
2369	* Matches the previous character 0 or more times.
2370	
2371	([.-]? \w+)* Matches 0 or more occurrences of [.-]? \w+.
2372	
2373	\w+([.-]? \w+)* The sub-expression \w+([.-]? \w+)* is used to match the <u>username</u> in the email.
2374	
	_____ It begins with at least one or more word characters including the underscore, equivalent to [A-Za-z0-9_]. , followed by . or - and . or - must follow by a word character (A-Za-z0-9_).
2375	@ It matches only @ character.
2376	
2377	\w+([.-]? \w+)* It matches the domain name with the same pattern of user name described above.
2378	
2379	\.\w{2,3} It matches a . followed by two or three word characters, e.g., . <u>edu</u> , . <u>org</u> , .com,
2380	. <u>uk</u> , .us, .co etc.
2381	
2382	+ The + sign specifies that the above sub-expression shall occur one or more times,
2383	e.g., .com, .co.us, . <u>edu.uk</u> etc.
2384	
2385	\$ Matches the end of the string or line.
2386	
2387	Note: If you want to work on 4 digit domain,
2388	for example, .info then you must change w{2,3} to w{2,4}.
2389	
2390	n{X} Matches any string that contains a sequence of X n's
2391	n{X,Y} Matches any string that contains a sequence of X to Y n's
2392	n{X,} Matches any string that contains a sequence of at least X n's
2393	n\$ Matches any string with n at the end of it
2394	^n Matches any string with n at the beginning of it

2395	?=n	Matches any string that is followed by a specific string n
2396	?!n	Matches any string that is not followed by a specific string n