UNIT II CSS AND CLIENT SIDE SCRIPTING

Style Sheets: CSS-Introduction to Cascading Style Sheets-Features-Core Syntax-Style Sheets and HTML-Style Rule Cascading and Inheritance-Text Properties-Box Model Normal Flow Box Layout- Beyond the Normal Flow-CSS3.0. Client-Side Programming: The JavaScript Language-History and Versions Introduction JavaScript in Perspective-Syntax-Variables & Data Types-Statements-Operators-Literals-Functions-Objects-Arrays-Built-in Objects-JavaScript Debuggers.

CSS

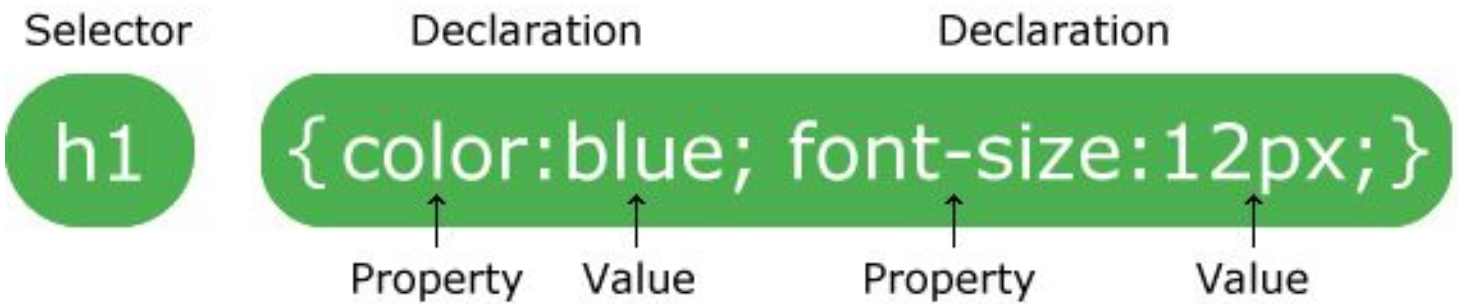
- **CSS** stands for **C**ascading **S**tyle **S**heets

- CSS describes how **HTML** elements are to be **displayed**

Introduction

- *Cascading Style Sheets* (CSS)
  - Specify the style of your page elements.
  - Spacing, margins, etc.
- Separate from the structure of your document.
  - Section headers, body text, links, etc.
- Separation of structure from content.
  - Greater manageability.
  - Easier to change style of document.

CSS Syntax



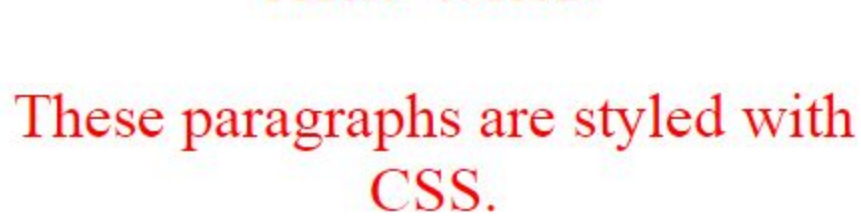
- The selector points to the HTML element to style (h1).
- The declaration block (in curly braces) contains one or more declarations separated by semicolons.
- Each declaration includes a CSS property name and a value, separated by a colon.

example

```
<style>
p {
    font-size:32px;
    color:red;
    text-align:center;
    }
</style>
```

Complete example

```
<html>
<style>
p {
    font-size: 32px;
    color: red;
    text-align: center;
}
</style>
<body>
<p>Hello World!</p>
<p>These paragraphs are styled with CSS.</p>
</body>
</html>
```



Types of CSS

- CSS can be added to HTML documents in 3 ways:
  - **Inline** - by using the style attribute inside HTML elements
  - **Internal** - by using a <style> element in the <head> section
  - **External** - by using a <link> element to link to an external CSS file

Inline CSS

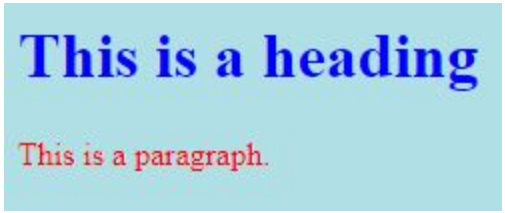
- An inline CSS is used to apply a unique style to a single HTML element.
- An inline CSS uses the style attribute of an HTML element.

```
<html>
<body>

<h1 style="color:blue;">A Blue Heading</h1>

<p style="color:red;">A red paragraph.</p>

</body>
</html>
```

Internal CSS

- An internal CSS is used to define a style for a single HTML page.
- An internal CSS is defined in the <head> section of an HTML page, within a <style> element.

```
<html>
<head>
<style>
body {background-color: powderblue;}
h1   {color: blue;}
p    {color: red;}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

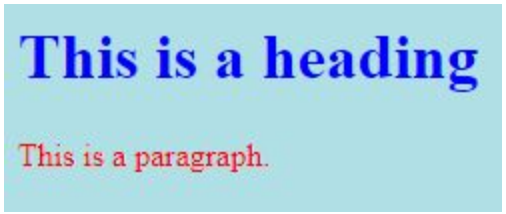


External CSS

- An external style sheet is used to define the style for many HTML pages.
- To use an external style sheet, add a link to it in the <head> section of each HTML page

Styles.css

```
body {background-color: powderblue;}
h1   {color: blue;}
p    {co
```



External.html

```
<html>
<head>
  <link rel="stylesheet" href="styles.css">
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

Cascading Order

Priority 1: Inline styles

Priority 2: internal style sheets

Priority 2: External style sheets

CSS properties

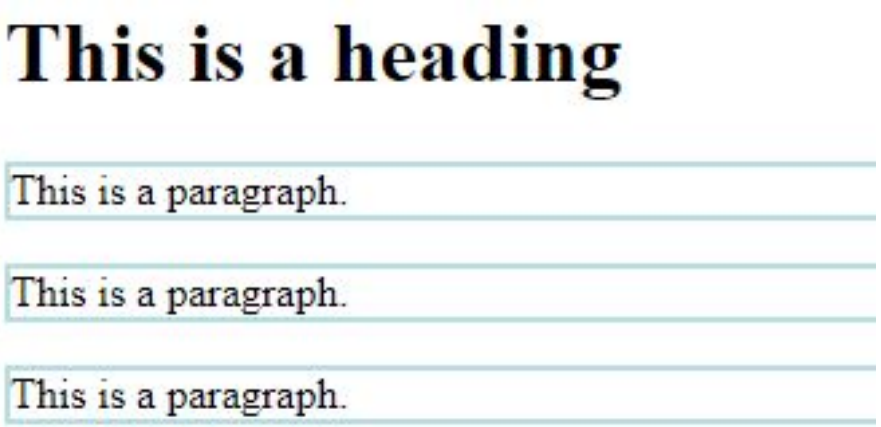
- The CSS color property defines the text color to be used.
- The CSS font-family property defines the font to be used.
- The CSS font-size property defines the text size to be used.

```
<html>
<head>
<style>
h1 {
  color: blue;
  font-family: Times New Roman;
  font-size: 300%;
}
p  {
  color: red;
  font-family: courier;
  font-size: 160%;
}
</style>
</head>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

CSS Border

- The CSS border property defines a border around an HTML element

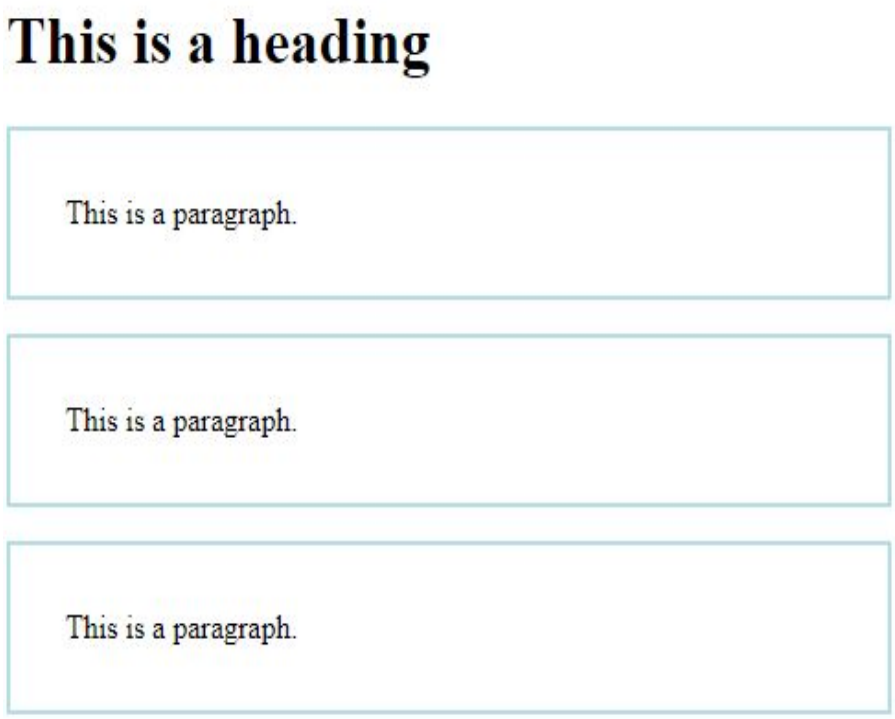
```
<html>
<head>
<style>
p {
  border: 2px solid powderblue;
}
</style>
</head>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
<p>This is a paragraph.</p>
<p>This is a paragraph.</p>
</body>
</html>
```



CSS Padding

The CSS padding property defines a padding (space) between the text and the border

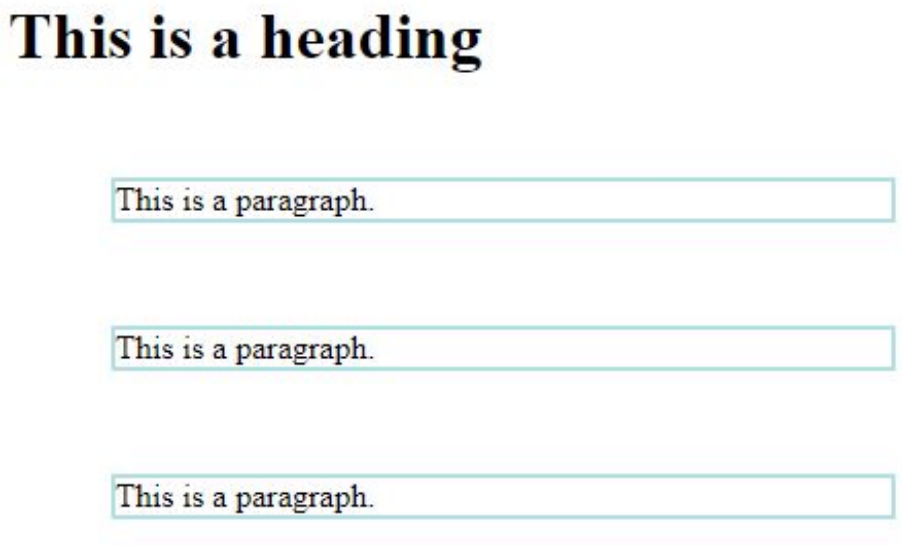
```
<html>
<head>
<style>
p {
  border: 2px solid powderblue;
  padding: 30px;
}
</style>
</head>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
<p>This is a paragraph.</p>
<p>This is a paragraph.</p>
</body>
</html>
```



CSS Margin

The CSS margin property defines a margin (space) outside the border.

```
<html>
<head>
<style>
p {
  border: 2px solid powderblue;
  margin: 50px;
}
</style>
</head>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
<p>This is a paragraph.</p>
<p>This is a paragraph.</p>
</body>
</html>
```



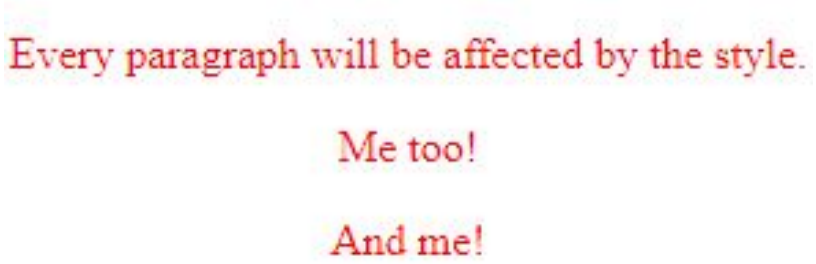
CSS Selectors

- CSS selectors are used to "find" (or select) the HTML elements you want to style.
  - The CSS element Selector
  - The CSS id Selector
  - The CSS class Selector
  - The CSS Universal Selector
  - The CSS Grouping Selector

The CSS element Selector

- The element selector selects HTML elements based on the element name.

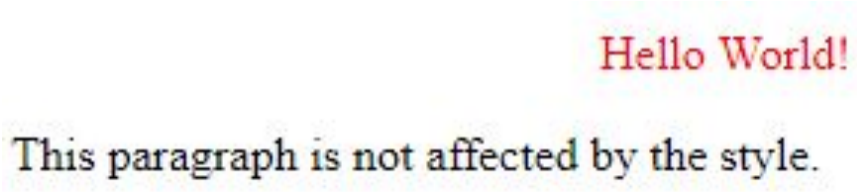
```
<html>
<head>
<style>
p {
  text-align: center;
  color: red;
}
</style>
</head>
<body>
<p>Every paragraph will be affected by
the style.</p>
<p >Me too!</p>
<p>And me!</p>
</body>
</html>
```



The CSS id Selector

- The id selector uses the id attribute of an HTML element to select a specific element.
- The id of an element is unique within a page, so the id selector is used to select one unique element!
- To select an element with a specific id, write a hash (#) character, followed by the id of the element

```
<html>
<head>
<style>
#para1 {
  text-align: center;
  color: red; }
</style>
</head>
<body>
<p id="para1">Hello World!</p>
<p>This paragraph is not affected by the
style.</p>
</body>
</html>
```



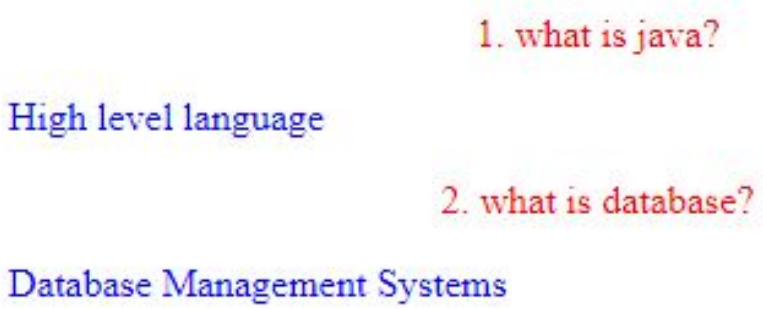
The CSS class Selector

- The class selector selects HTML elements with a specific class attribute.
- To select elements with a specific class, write a period (.) character, followed by the class name.

```
<html>
<head>
<style>
.center {
  text-align: center;
  color: red;
}
</style>
</head>
<body>
<h1 class="center">Red and
center-aligned heading</h1>
<p class="center">Red and
center-aligned paragraph.</p>
</body>
</html>
```

Example

```
<html>
<head>
<style>
p.center1 {
  text-align: center;
  color: red;
}
p.center2 {
  text-align: left;
  color: blue;
}
</style>
</head>
<body>
<p class="center1">1. what is java?</p>
<p class="center2">High level language</p>
<p class="center1">2. what is database?</p>
<p class="center2">Database Management Systems</p>
</body>
</html>
```



The CSS Universal Selector

- The universal selector (*) selects all HTML elements on the page.

```
<html>
<head>
<style>
* {
  text-align: center;
  color: blue;
}
</style>
</head>
<body>
<h1>Hello world!</h1>
<p>Every element on the page will be
affected by the style.</p>
<p id="para1">Me too!</p>
<p>And me!</p>
</body>
</html>
```

The CSS Grouping Selector

- The grouping selector selects all the HTML elements with the same style definitions.

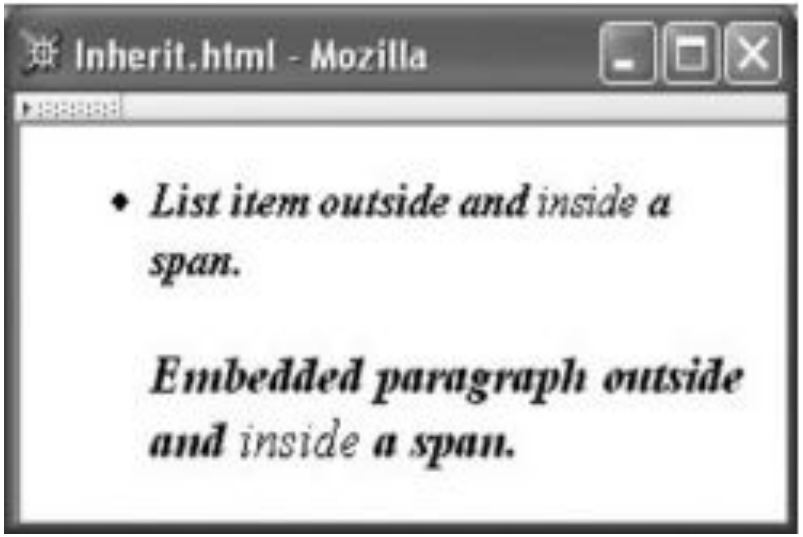
```
h1 {
  text-align: center;
  color: red;
}

h2 {
  text-align: center;
  color: red;
}

p {
  text-align: center;
  color: red;
}
```

```
<html>
<head>
<style>
h1, h2, p {
  text-align: center;
  color: red;
}
</style>
</head>
<body>
<h1>Hello World!</h1>
<h2>Smaller heading!</h2>
<p>This is a paragraph.</p>
</body>
</html>
```

CSS Inheritance

- cascading is based on the structure of style sheets
- inheritance is based on the tree structure of the document itself.
- Conceptually an element inherits a value for one of its properties by checking to see if its parent element in the document has a value for that property, and if so, inheriting the parent's value.
- The parent may in turn inherit its property value from its parent, and so on

```
<!DOCTYPE html
        PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>
      Inherit.html
    </title>
    <style type="text/css">
      body { font-weight:bold }
      li { font-style:italic }
      p { font-size:larger }
      span { font-weight:normal }
    </style>
  </head>
  <body>
    <ul>
      <li>
        List item outside and <span>inside</span> a span.
        <p>
          Embedded paragraph outside and <span>inside</span> a span.
        </p>
      </li>
    </ul>
  </body>
</html>
```



CSS Text properties

- CSS text formatting properties is used to format text and style text.
  CSS text formatting include following properties:
  1. Text-color
  2. Text-alignment
  3. Text-decoration
  4. Text-transformation
  5. Text-indentation
  6. Letter spacing
  7. Line height
  8. Text-direction
  9. Text-shadow
  10. Word spacing

1.TEXT COLOR

```
<html>
<head>
<style>
h1
{
color:red;
}
h2
{
color:green;
}
</style>
</head>
<body>
<h1>
GEEKS FOR GEEKS
</h1>
<h2>
TEXT FORMATTING
</h2>
</body>
</html>
```

2.TEXT ALIGNMENT

```
<html>
<head>
<style>
h1
{
color:red;
text-align:center;
}
h2
{
color:green;
text-align:left;
}
</style>
</head>
<body>
<h1>
GEEKS FOR GEEKS
</h1>
<h2>
TEXT FORMATTING
</h2>
</body>
</html>
```

3. TEXT DECORATION

```
<html>
<head>
<style>
h1
{
color:red;
text-decoration:underline;
}
</style>
</head>
<body>
<h1>
GEEKS FOR GEEKS
</h1>
<h2>
TEXT FORMATTING
</h2>
</body>
</html>
```

4. TEXT TRANSFORMATION

Text transformation property is used to change the case of text, uppercase or lowercase

```
<html>
<head>
<style>
h2
{
text-transform:lowercase;
}
</style>
</head>
<body>
<h1>
GEEKS FOR GEEKS
</h1>
<h2>
TEXT FORMATTING
</h2>
</body>
</html>
```

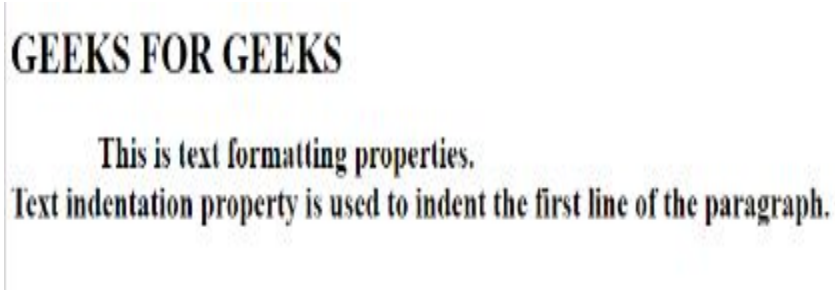
GEEKS FOR GEEKS

text formatting

5. TEXT INDENTATION

- Text indentation property is used to indent the first line of the paragraph.
- The size can be in px, cm, pt.

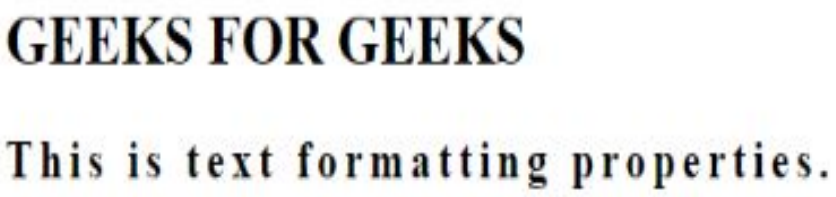
```
<!DOCTYPE html>
<html>
<head>
<style>
h2
{
text-indent:80px;
}
</style>
</head>
<body>
<h1>
GEEKS FOR GEEKS
</h1>
<h2>
This is text formatting properties.<br>
Text indentation property is used to indent the first line
of the paragraph.
</h2>
</body>
</html>
```



6.LETTER SPACING

- This property is used to specify the space between the characters of the text.
- The size can be given in px.

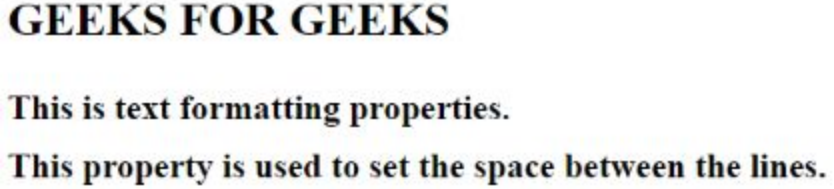
```
<html>
<head>
<style>
h2
{
letter-spacing:4px;
}
</style>
</head>
<body>
<h1>
GEEKS FOR GEEKS
</h1>
<h2>
This is text formatting properties.
</h2>
</body>
</html>
```



7.LINE HEIGHT

- This property is used to set the space between the lines.

```
<html>
<head>
<style>
h2
{
line-height:40px;
}
</style>
</head>
<body>
<h1>
GEEKS FOR GEEKS
</h1>
<h2>
This is text formatting properties.<br>
This property is used to set the space between the
lines.
</h2>
</body>
</html>
```



8.TEXT SHADOW

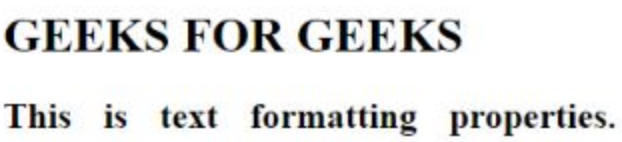
- Text shadow property is used to add shadow to the text.
- You can specify the horizontal size, vertical size and shadow color for the text.

```
<html>
<head>
<style>
h1
{
text-shadow:3px 1px blue;
}
</style>
</head>
<body>
<h1>
GEEKS FOR GEEKS
</h1>
<h2>
This is text formatting properties.
</h2>
</body>
</html>
```

9. WORD SPACING

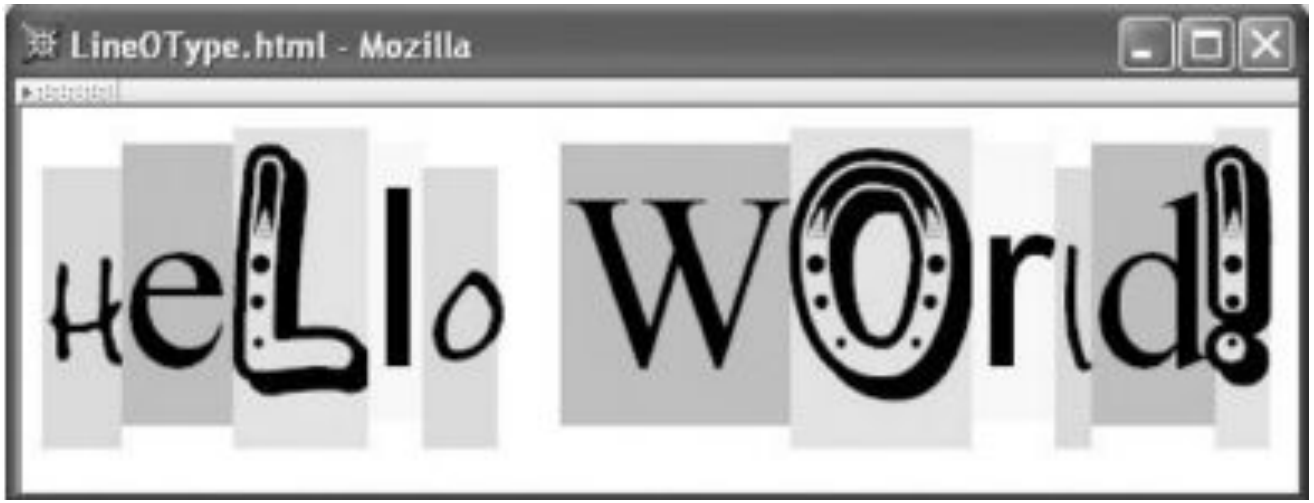
- Word spacing is used to specify the space between the words of the line.
- The size can be given in px.

```
<html>
<head>
<style>
h2
{
word-spacing:15px;
}
</style>
</head>
<body>
<h1>
GEEKS FOR GEEKS
</h1>
<h2>
This is text formatting properties.
</h2>
</body>
</html>
```



font families

- <p font-family: arial”> hai </p>



CSS Generic Font Families

Font Family	Description
`serif`	A *serif* is a short, decorative line at the end of a stroke of a letter. There are three serifs at the top of the W in Figure 3.12, for example. Most glyphs in a serif font family will have serifs, and such a family is typically *proportionately* spaced (different glyphs occupy different widths).
`sans-serif`	Usually proportionately spaced, but glyphs lack serifs, so they don't look as fancy as serif fonts.
`cursive`	Looks more like cursive handwriting than like printing.
`fantasy`	Glyphs are still recognizable as characters, but are nontraditional.
`monospace`	All glyphs have the same width. Since monospace fonts are often used in editors when programming, these font families are frequently used to display program code or other computer data.

Length Specifications in CSS

font-size:0.25in

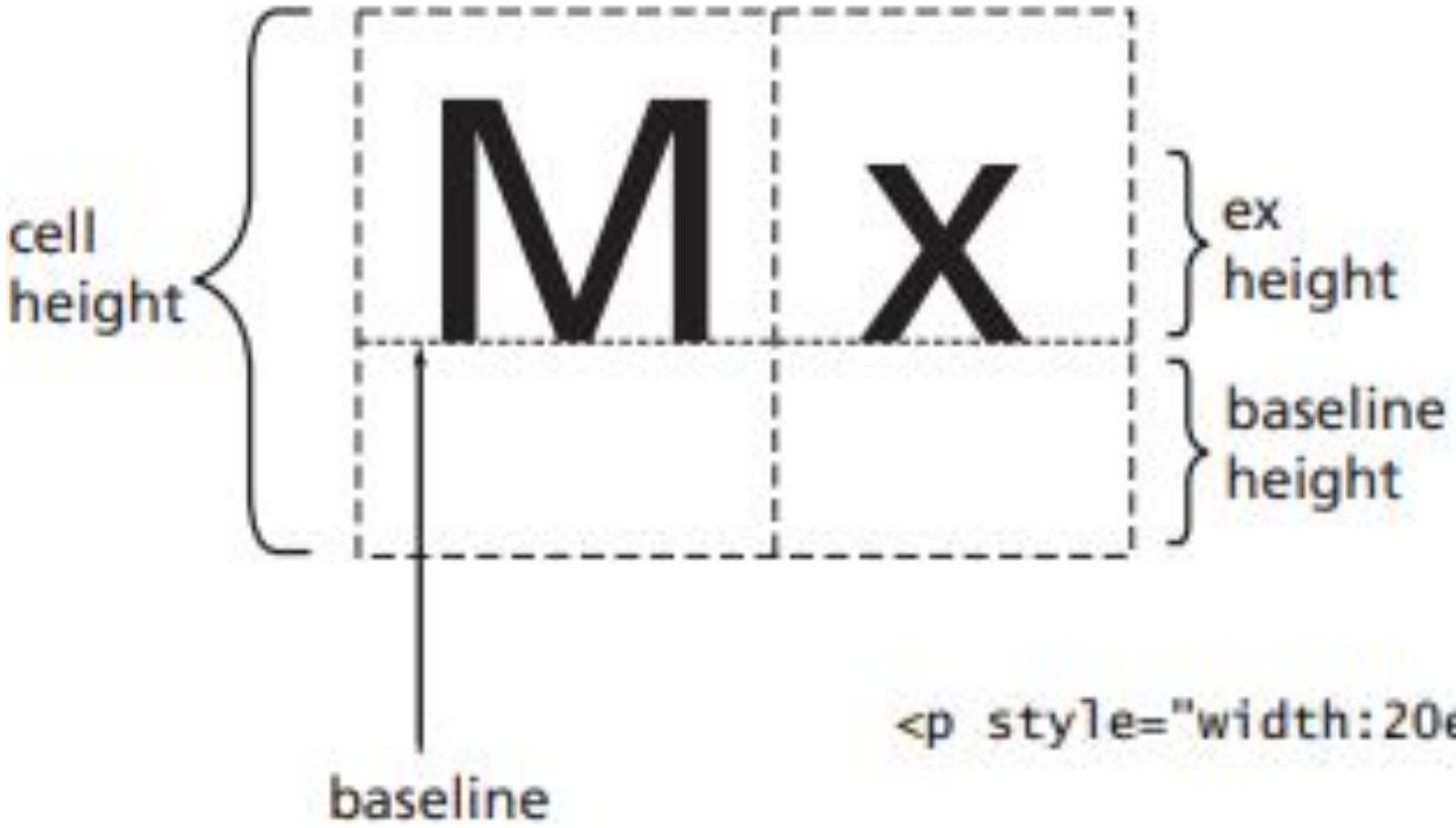
font-size:12pt

font-size:15px

CSS Length Unit Identifiers

Identifier	Meaning
in	Inch
cm	Centimeter
mm	Millimeter
pt	Point: 1/72 inch
pc	Pica: 12 points
px	Pixel: typically 1/96 inch (see text)
em	Em: reference font size (see text)
ex	Ex: roughly the height of the lowercase "x" character in the reference font (see text)

Some features and quantities defined by a font




```
<p style="width:20em">

<div id="d1" style="font-size:12pt">
  <div id="d2" style="font-size:2em">
```

Font Properties

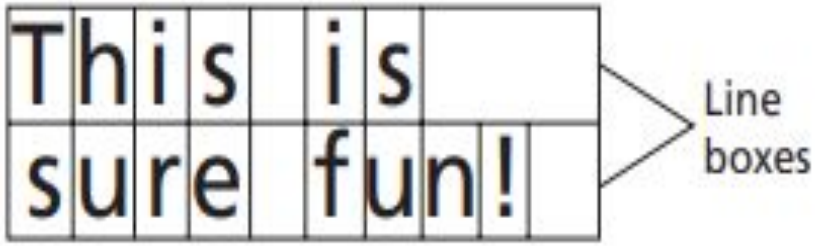
- The CSS font-size property, we now know, is used to specify the approximate height of character cells in the desired font within a font family

```
font-size:0.85em
font-size:85%
```

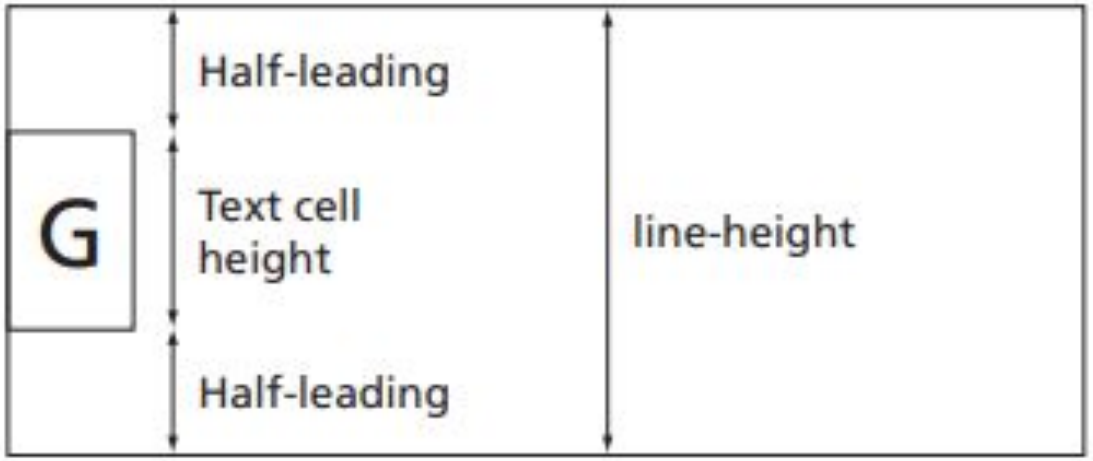
Additional Font Style Properties

Property	Possible Values
font-style	normal (initial value), italic (more cursive than normal), or oblique (more slanted than normal)
font-weight	bold or normal (initial value) are standard values, although other values can be used with font families having multiple gradations of boldness (see CSS2 [W3C-CSS-2.0] for details)
font-variant	small-caps, which displays lowercase characters using uppercase glyphs (small uppercase glyphs if possible), or normal (initial value)

Line Boxes



A box representing a p element that consists of two line boxes, each partially filled with character cells.




```
line-height:1.5em
line-height:150%
line-height:1.5
```

```
{ font: italic bold 12pt "Helvetica",sans-serif }
```

is equivalent to the the declaration block

```
{ font-style: italic;
  font-variant: normal;
  font-weight: bold;
  font-size: 12pt;
  line-height: normal;
  font-family: "Helvetica",sans-serif }
```

Primary CSS Text Properties

Property	Values
`text-decoration`	`none` (initial value), `underline`, `overline`, `line-through`, or space-separated list of values other than `none`.
`letter-spacing`	`normal` (initial value) or a length representing additional space to be included between adjacent letters in words. Negative value indicates space to be removed.
`word-spacing`	`normal` (initial value) or a length representing additional space to be included between adjacent words. Negative value indicates space to be removed.
`text-transform`	`none` (initial value), `capitalize` (capitalizes first letter of each word), `uppercase` (converts all text to uppercase), `lowercase` (converts all text to lowercase).
`text-indent`	Length (initial value 0) or percentage of box width, possibly negative. Specify for block elements and table cells to indent text within first line box.
`text-align`	`left` (initial value for left-to-right contexts), `right`, `center`, or `justified`. Specify for block elements and table cells.
`white-space`	`normal` (initial value), `pre`. Use to indicate whether or not white space should be retained.

Text Formatting and Color

Alternative Formats for Specifying Numeric Color Values

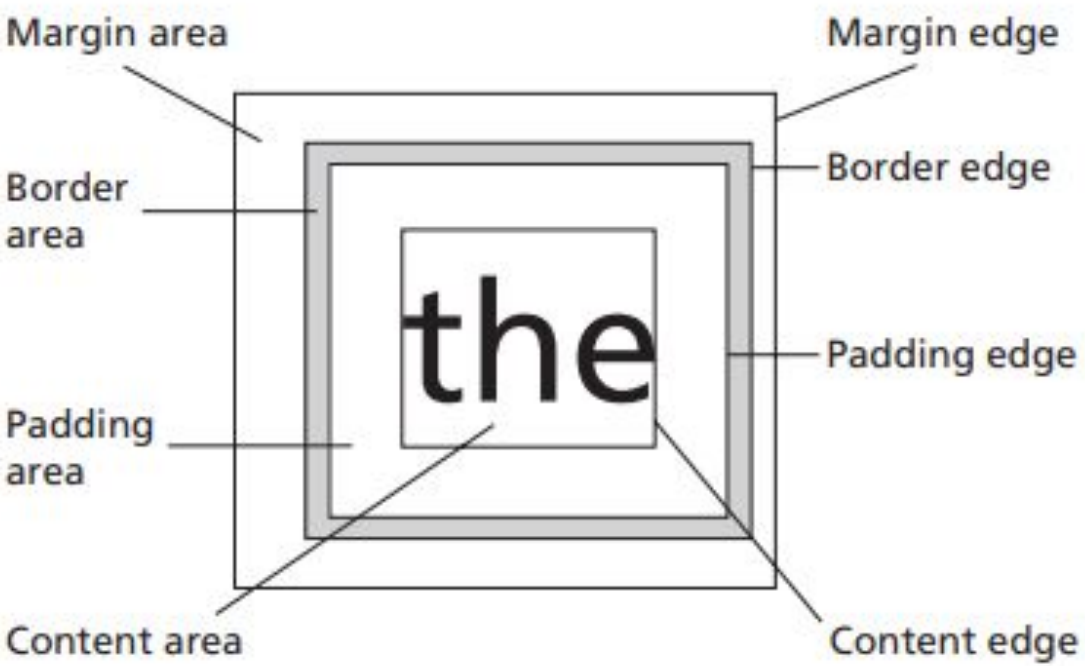
Format	Example	Meaning
Functional, integer arguments	`rgb(255,170,0)`	Use arguments as RGB values.
Functional, percentage arguments	`rgb(100%,66.7%,0%)`	Multiply arguments by 255 and round to obtain RGB values (at most one decimal place allowed in arguments).
Hexadecimal	`#ffaa00`	The first pair of hexadecimal digits represents the red intensity; the second and third represent green and blue, respectively.
Abbreviated hexadecimal	`#fa0`	Duplicate the first hexadecimal digit to obtain red intensity; duplicate the second and third to obtain green and blue, respectively.

CSS2 Color Names and RGB Values

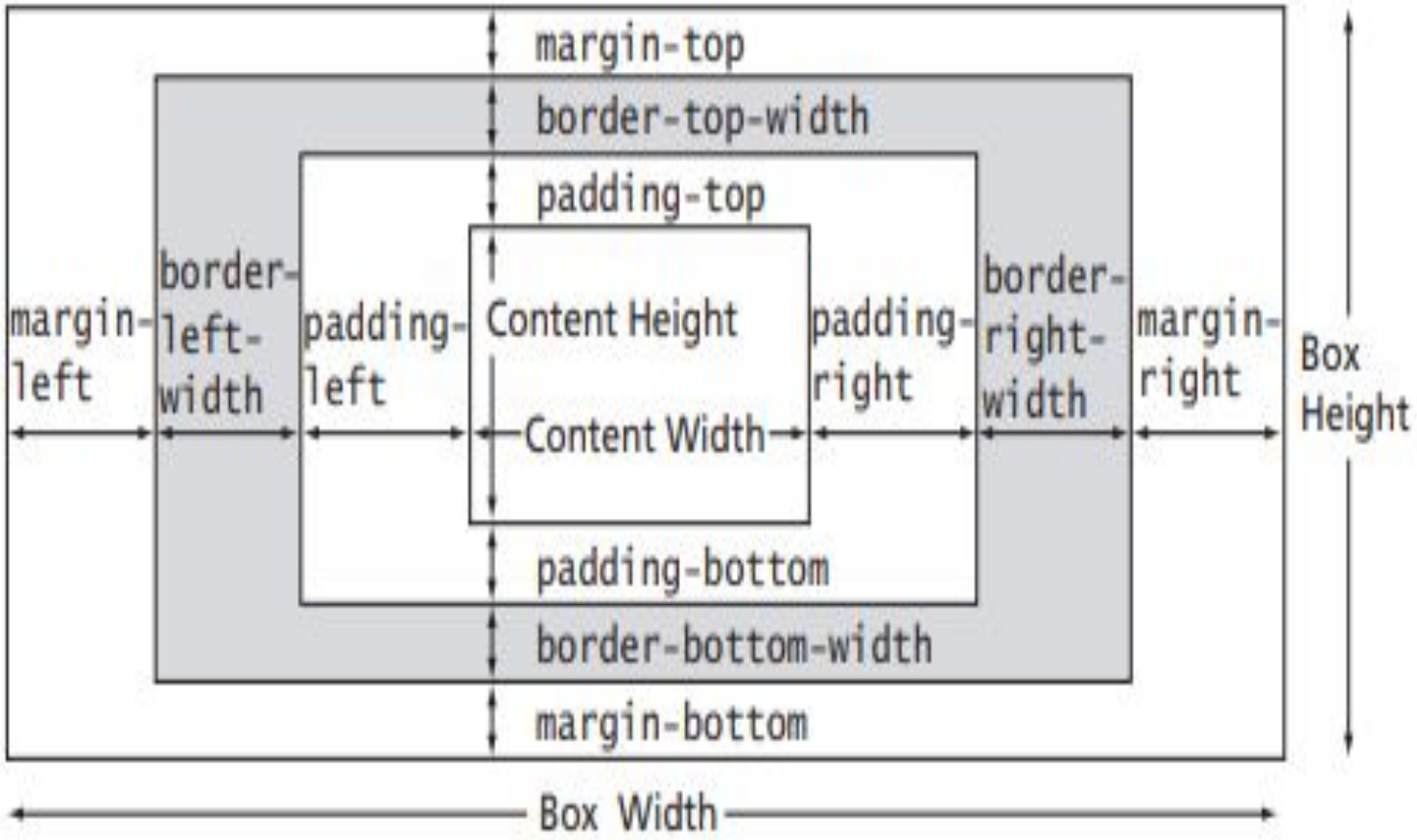
Color Name	RGB Value
black	#000000
gray	#808080
silver	#c0c0c0
white	#ffffff
red	#ff0000
lime	#00ff00
blue	#0000ff
yellow	#ffff00
aqua	#00ffff
fuchsia	#ff00ff
maroon	#800000
green	#008000
navy	#000080
olive	#808000
teal	#008080
purple	#800080

CSS Box Model

- In CSS, each element of an HTML or XML document, if it is rendered visually, occupies a rectangular area— **a box** —on the screen or other visual output medium



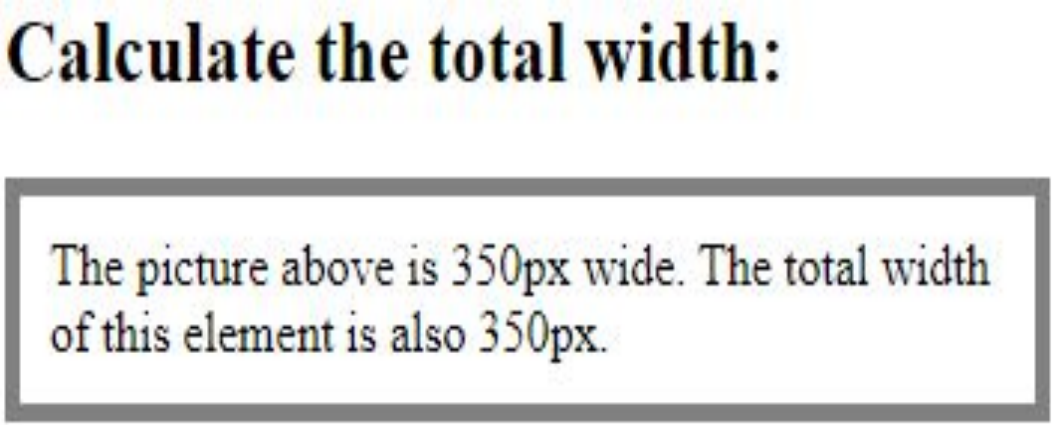
Definition of areas and edges in the CSS box model



Definition of various lengths in the CSS box model.

Width and Height of an Element

```
<html>
<head>
<style>
div {
  width: 320px;
  padding: 10px;
  border: 5px solid gray;
  margin: 0px;
}
</style>
</head>
<body>
<h2>Calculate the total width:</h2>
<div>The picture above is 350px wide. The total width of this element is also 350px.</div>
</body>
</html>
```



Box Model Shorthand Properties

```
padding: 30px;
```

is shorthand for four declarations:

```
padding-top: 30px;
padding-right: 30px;
padding-bottom: 30px;
padding-left: 30px;
```

Basic CSS Style Properties Associated with the Box Model

Property	Values
padding-{top,right,bottom,left}	CSS length (Section 3.6.2).
padding	One to four length values (see text).
border-{top,right,bottom,left}-width	thin, medium (initial value), thick, or a length.
border-width	One to four border-*-width values.
border-{top,right,bottom,left}-color	Color value. Initial value is value of element's color property.
border-color	transparent or one to four border-*-color values.
border-{top,right,bottom,left}-style	none (initial value), hidden, dotted, dashed, solid, double, groove, ridge, inset, outset.
border-style	One to four border-*-style values.
border-{top,right,bottom,left}	One to three values (in any order) for border-*-width, border-*-color, and border-*-style. Initial values are used for any unspecified values.
border	One to three values; equivalent to specifying given values for each of border-top, border-right, border-bottom, and border-left.
margin-{top,right,bottom,left}	auto (see text) or length.
margin	One to four margin-* values.

Meaning of Values for Certain Shorthand Properties that Take One to Four Values

Number of Values	Meaning
One	Assign this value to all four associated properties (`top`, `right`, `bottom`, and `left`).
Two	Assign first value to associated `top` and `bottom` properties, second value to associated `right` and `left` properties.
Three	Assign first value to associated `top` property, second value to `right` and `left`, and third value to `bottom`.
Four	Assign first value to associated `top` property, second to `right`, third to `bottom`, and fourth to `left`.

```
margin: 15px 45px 30px
```

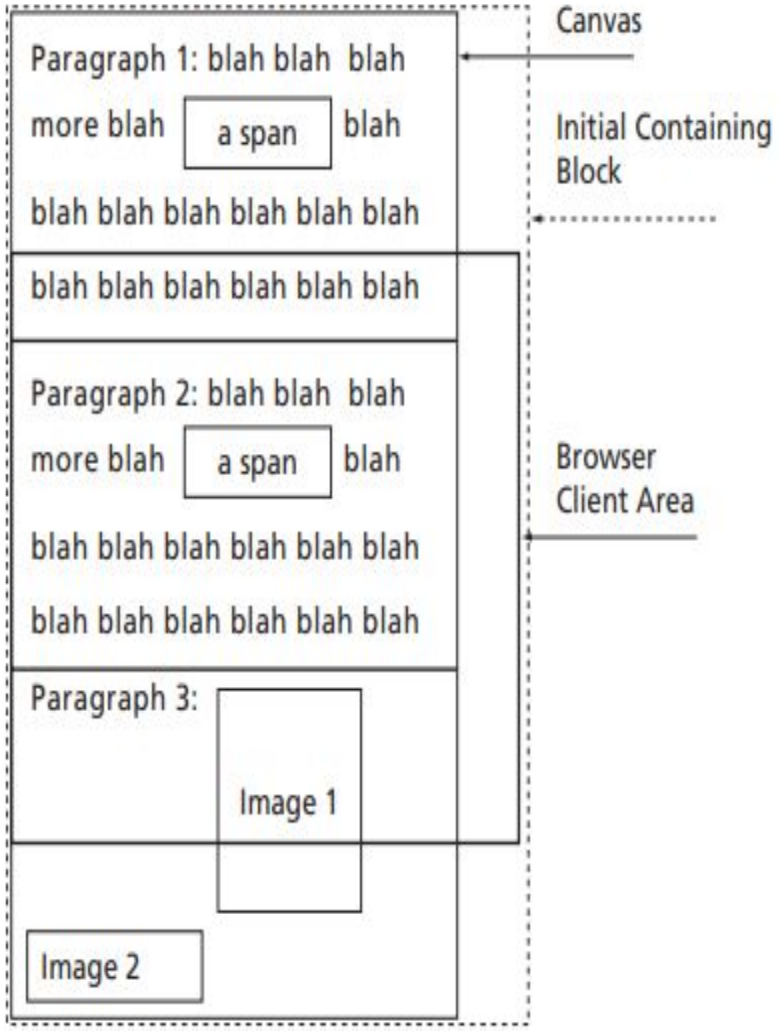
is equivalent to

```
margin-top: 15px
margin-right: 45px
margin-left: 45px
margin-bottom: 30px
```

Normal Flow Box Layout

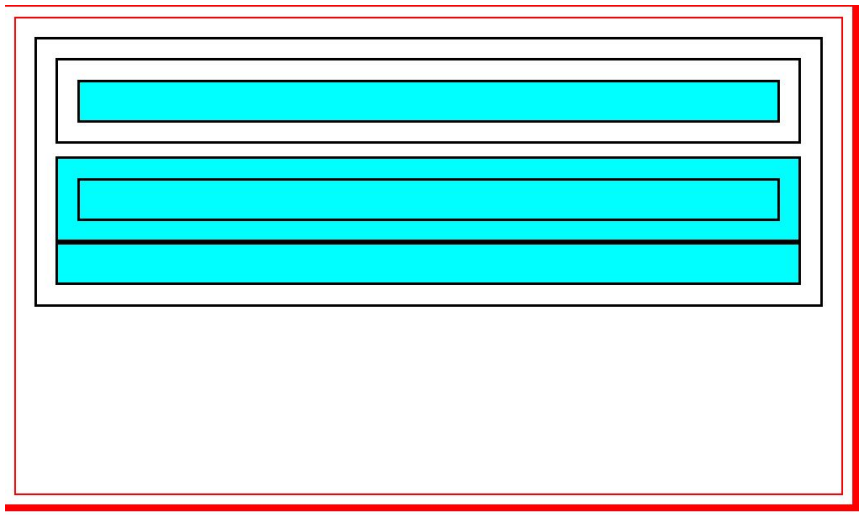
- In a browser's standard rendering model (which is called normal flow processing), every HTML element rendered by the browser has a corresponding rectangular box that contains the rendered content of the element.
- The primary question we address in this section is where these boxes will be placed within the client area of the browser.

Basic Box Layout

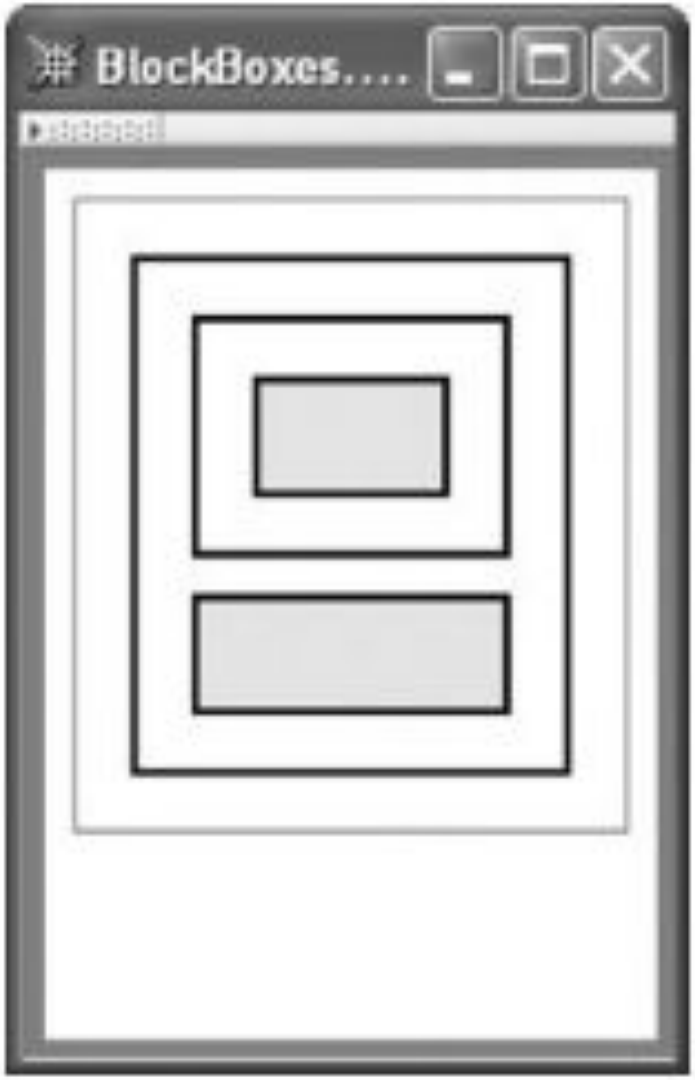


Initial containing block box when canvas is taller than client area but client area is wider than canvas.

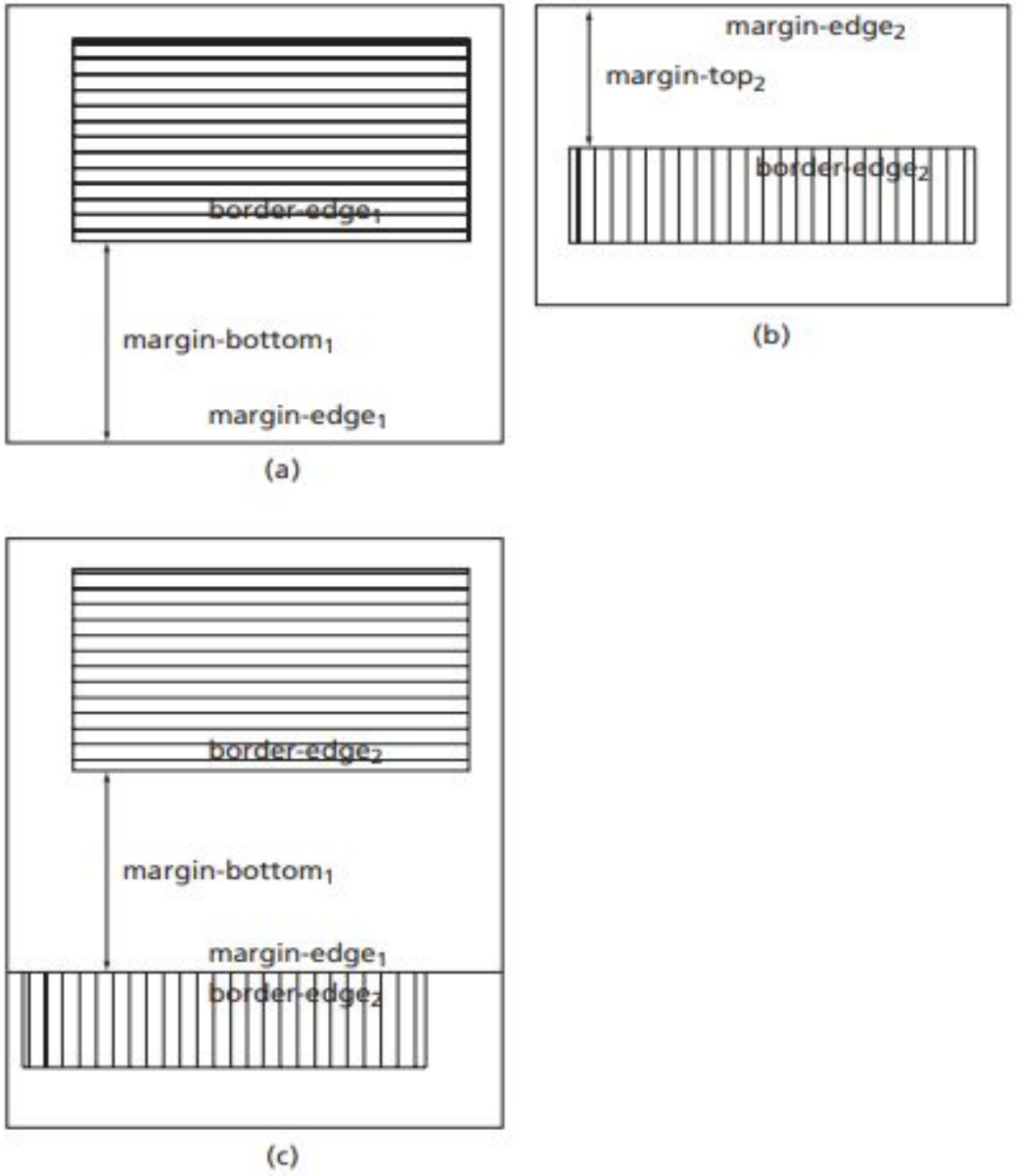
```
<!DOCTYPE html
       PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
       "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>
      BlockBoxes.html
    </title>
    <style type="text/css">
      html, body { border:solid red thin }
      html { border-width:thick }
      body { padding:15px }
      div { margin:0px; padding:15px; border:solid black 2px }
      .shade { background-color:aqua }
      .topMargin { margin-top:10px }
    </style>
  </head>
  <body>
    <div id="d1">
      <div id="d2">
        <div id="d3" class="shade"></div>
      </div>
      <div id="d4" class="shade topMargin"></div>
    </div>
  </body>
</html>
```



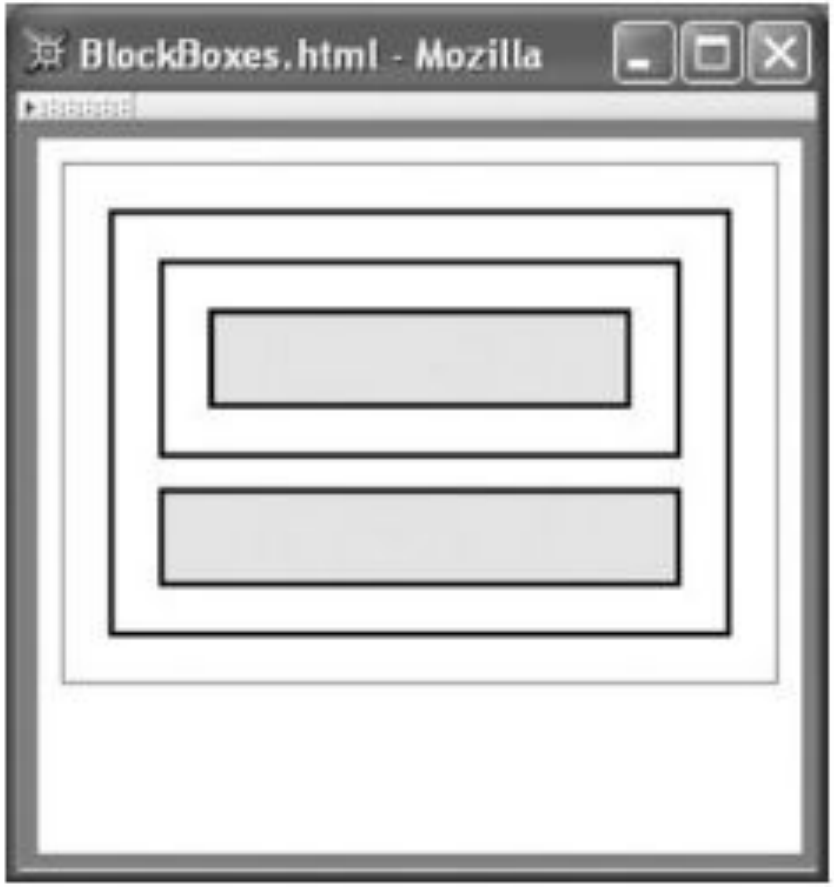
HTML document containing nested div elements.



Nested boxes.



(a) A block box (only margin and border edges are shown). (b) A second block box with `margin-top` smaller than `margin-bottom` of first box. (c) First and then second boxes rendered, illustrating margin collapse.

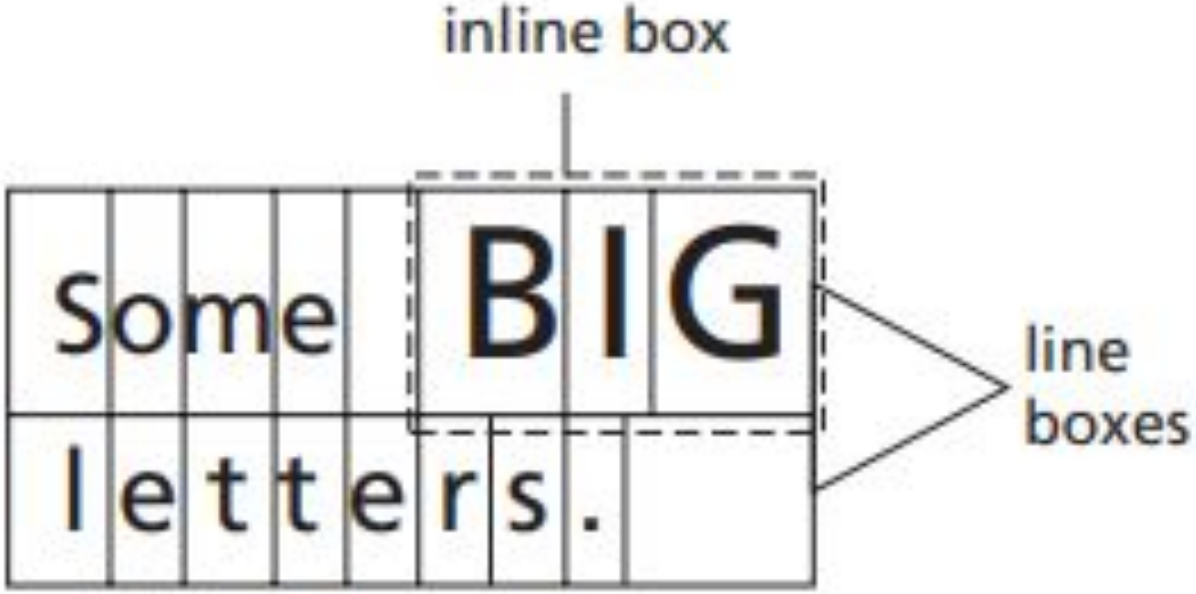


BlockBoxes.html displayed in a wider window.

#d3 { width:50% }

Simple inline boxes

```
<html>
<head>
<title>
BlockBoxes.html
</title>
<style type="text/css">
.shade { background-color:aqua }
</style>
</head>
<body>
<div id="d3" class="shade">
Here are
<span style="border:dotted silver
10px">some</span>
<span style="font-size:36pt">BIG</span>
lines of text.
</div>
</body>
</html>
```


inline box
Some BIG
letters.
line
boxes

Beyond the Normal Flow

- Properties for positioning
  - Relative positioning
  - Float positioning
  - Absolute positioning

Relative positioning

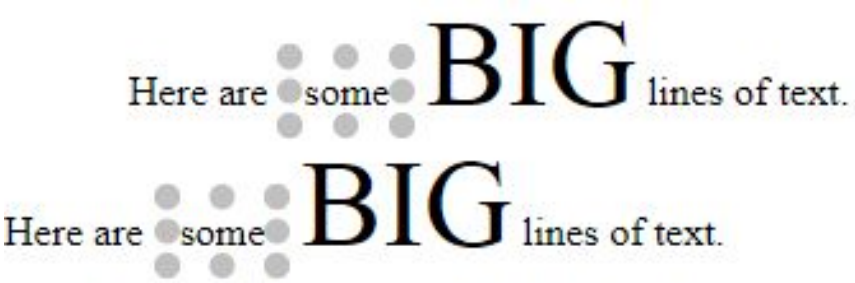
- Relative positioning is useful when you want to move a box a bit from the position where the browser would normally render it, and you want to do so without disturbing the placement of any other boxes.

```
.right { position:relative; right:0.25em }
```

```
.right { position:relative; left:-0.25em }
```

Example

```
<html>
<head>
<title>
BlockBoxes.html
</title>
<style type="text/css">
.right { position:relative; right:-2.55em }
</style>
</head>
<body>
<div id="d3" class="right">
Here are
<span style="border:dotted silver 10px">some</span>
<span style="font-size:36pt">BIG</span>
lines of text.
</div>
</body>
</html>
```



Float Positioning

```
<html>
<head>
<title>
BlockBoxes.html
</title>
<style type="text/css">

</style>
</head>
<body>
<p>
<img
src="a.jpg"
alt="Valid XHTML 1.0!" height="31" width="88"
style="float:left" />
HTML stands for Hyper Text Markup Language

HTML is the standard markup language for Web pages

HTML elements are the building blocks of HTML pages

HTML elements are represented by <> tags
</p>
</body>
</html>
```

- Float positioning is often used when embedding images within a paragraph.

Absolute Positioning

Absolute positioning offers total control over the placement of boxes on the canvas.

```
<html>
<head>
<title>
BlockBoxes.html
</title>
<style type="text/css">
.marginNote { position:absolute;
top:0; left:-10em; width:8em;
background-color:yellow }
</style>
</head>
<body class="marginNote">
<p >
HTML stands for Hyper Text Markup Language

HTML is the standard markup language for Web pages

HTML elements are the building blocks of HTML pages

HTML elements are represented by <> tags
</p>
</body>
</html>
```

Client-side Programming using java script

What is Client-side Scripting?

- The **client-side** environment used to run **scripts** is usually a browser.
- The processing takes place on the end users computer.
- The source code is transferred from the web server to the users computer over the internet and run directly in the browser.
- Examples of Client side scripting languages : **Javascript, VB script, etc.**

Server side scripting vs. Client side scripting

Server Side Validation

- In the Server Side Validation, the input submitted by the user is being sent to the server and validated using one of server side scripting languages such as ASP.Net, PHP etc.
- After the validation process on the Server Side, the feedback is sent back to the client by a new dynamically generated web page.
- It is better to validate user input on Server Side because you can protect against the malicious users, who can easily bypass your Client Side scripting language and submit dangerous input to the server.

Client Side Validation

- In the Client Side Validation you can provide a better user experience by responding quickly at the browser level.
- When you perform a Client Side Validation, all the user inputs validated in the user's browser itself.
- Client Side validation does not require a round trip to the server, so the network traffic which will help your server perform better.

client side scripting	server side scripting
The client is the system on which the Web browser is running.	The server is where the Web page and other content lives.
JavaScript is the main client-side scripting language for the Web.	JSP and Servlets is the server-side scripting language for the Web.
Client-side scripts are interpreted by the browser.	The server sends pages to the user/client on request.

What is JavaScript ?

- JavaScript is a lightweight, interpreted programming language
- Designed for creating network-centric applications
- integrated with Java
- integrated with HTML
- Open and cross-platform

uses for javascript:

- Browser Detection
  Detecting the browser used by a visitor at your page. Depending on the browser, another page specifically designed for that browser can then be loaded.
- Cookies
  Storing information on the visitor's computer, then retrieving this information automatically next time the user visits your page. This technique is called "cookies".
- Control Browsers
  Opening pages in customized windows, where you specify if the browser's buttons, menu line, status line or whatever should be present.
- Validate Forms
  Validating inputs to fields before submitting a form.

JavaScript Syntax:

- A JavaScript consists of JavaScript statements that are placed within the <script>... </script> HTML tags in a web page.
- You can place the <script> tag containing your JavaScript anywhere within you web page but it is preferred way to keep it within the <head> tags.
- The <script> tag alert the browser program to begin interpreting all the text between these tags as a script. So simple syntax of your JavaScript will be as follows

```
<script  language="javascript"  type=" text/javascript" >
JavaScript code
</script>
```

language: This attribute specifies what scripting language you are using. Typically, its value will be *javascript*.

type: This attribute is what is now recommended to indicate the scripting language in use and its value should be set to *"text/javascript"*.

First JavaScript Script:

```
<html>
<body>
 <script language="javascript" type="text/javascript">
document.write("Hello World!")
</script>
 </body>
</html>
```

Output:

Hello World!

Whitespace and Line Breaks:

JavaScript ignores spaces, tabs, and newlines that appear in JavaScript programs.

Semicolons are Optional:

Simple statements in JavaScript are generally followed by a semicolon character, just as they are in C, C++, and Java.

JavaScript, however, allows you to omit this semicolon if your statements are each placed on a separate line.

Case Sensitivity:

JavaScript is a case-sensitive language.

Comments in JavaScript:

- JavaScript supports both C-style and C++-style comments, Thus:
- Any text between a // and the end of a line is treated as a comment and is ignored by JavaScript.
- Any text between the characters /* and */ is treated as a comment. This may span multiple lines.
- JavaScript also recognizes the HTML comment opening sequence <!--. JavaScript treats this as a single-line comment, just as it does the // comment.
- The HTML comment closing sequence --> is not recognized by JavaScript so it should be written as //-->.

JavaScript Placement in HTML File:

- Script in <head>...</head> section.
- Script in <body>...</body> section.
- Script in and external file and then include in <head>...</head> section.

JavaScript DataTypes:

JavaScript allows you to work with three primitive data types:

- Numbers eg. 123, 120.50 etc.
- Strings of text e.g. "This text string" etc.
- Boolean e.g. true or false.
- JavaScript also defines another two data types *null* and *undefined*

JavaScript Variables

- Variables can be thought of as named containers.
- You can place data into these containers and then refer to the data simply by naming the container.
- Before you use a variable in a JavaScript program, you must declare it.
- Variables are declared with the **var** keyword.

```
var money;
var name;
```

JavaScript Variable Names:

- JavaScript variable names should not start with a numeral (0-9).
- They must begin with a letter or the underscore character. For example, *123test* is an invalid variable name but *_123test* is a valid one.
- You should not use any of the JavaScript reserved keyword as variable name. For example, *break* or *boolean* variable names are not valid.
- JavaScript variable names are case sensitive. For example, *Name* and *name* are two different variables.

JavaScript Reserved Words:

abstract	else	instanceof	switch
boolean	enum	int	synchronized
break	export	interface	this
byte	extends	long	throw
case	false	native	throws
catch	final	new	transient
char	finally	null	true
class	float	package	try
const	for	private	typeof
continue	function	protected	var
debugger	goto	public	void
default	if	return	volatile
delete	implements	short	while
do	import	static	with
double	in	super	

The Arithmatic Operators:

Assume variable A holds 10 and variable B holds 20 then:

Operator	Description	Example
+	Adds two operands	A + B will give 30
-	Subtracts second operand from the first	A - B will give -10
*	Multiply both operands	A * B will give 200
/	Divide numerator by denumerator	B / A will give 2
%	Modulus Operator and remainder of after an integer division	B % A will give 0
++	Increment operator, increases integer value by one	A++ will give 11
--	Decrement operator, decreases integer value by one	A-- will give 9

The Comparison Operators:

Operator	Description	Example
==	Checks if the value of two operands are equal or not, if yes then condition becomes true.	(A == B) is not true.
!=	Checks if the value of two operands are equal or not, if values are not equal then condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true.

The Logical Operators:

Operator	Description	Example
&&	Called Logical AND operator. If both the operands are non zero then then condition becomes true.	(A && B) is true.
||	Called Logical OR Operator. If any of the two operands are non zero then then condition becomes true.	(A || B) is true.
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!(A && B) is false.

The Bitwise Operators:

Operator	Description	Example
&	Called Bitwise AND operator. It performs a Boolean AND operation on each bit of its integer arguments.	(A & B) is 2 .
|	Called Bitwise OR Operator. It performs a Boolean OR operation on each bit of its integer arguments.	(A | B) is 3.
^	Called Bitwise XOR Operator. It performs a Boolean exclusive OR operation on each bit of its integer arguments. Exclusive OR means that either operand one is true or operand two is true, but not both.	(A ^ B) is 1.
~	Called Bitwise NOT Operator. It is a is a unary operator and operates by reversing all bits in the operand.	(~B) is -4 .

The Assignment Operators:

Operator	Description	Example
=	Simple assignment operator, Assigns values from right side operands to left side operand	C = A +
+=	It adds right operand to the left operand and assign the result to left operand	C += A is equivalent to C = C + A
-=	It subtracts right operand from the left operand and assign the result to left operand	C -= A is equivalent to C = C - A
*=	It multiplies right operand with the left operand and assign the result to left operand	C *= A is equivalent to C = C * A
/=	It divides left operand with the right operand and assign the result to left operand	C /= A is equivalent to C = C / A
%=	It takes modulus using two operands and assign the result to left operand	C %= A is equivalent to C = C % A

Miscellaneous Operator

The Conditional Operator (? :)

Operator	Description	Example
? :	Conditional Expression	if Condition is true ? Then value X : Otherwise value Y

Control statement

- if statement
- if...else statement
- if...else if... Statement
- switch statement
- The *while* Loop
- The *do...while* Loop
- The *for* Loop

Arrays

JavaScript arrays are used to store multiple values in a single variable.

var *array-name* = [*item1*, *item2*, ...];

Example:

```
var cars = [“Maruthi", “Honda", “Ford"];
(Or)
var cars = [
      “Maruthi",
      “Honda",
      “Ford"
   ];
```

Access the Elements of an Array

This statement access the value of the first element in myCars:

```
var name = cars[0];
```

This statement modifies the first element in cars:

```
cars[0] = “Tata ";
```

JavaScript Objects

- In JavaScript, almost "everything" is an object.
  - Booleans can be objects (or primitive data treated as objects)
  - Numbers can be objects (or primitive data treated as objects)
  - Strings can be objects (or primitive data treated as objects)
  - Dates are always objects
  - Maths are always objects
  - Regular expressions are always objects
  - Arrays are always objects
  - Functions are always objects
  - Objects are objects

- In JavaScript, all values, except primitive values, are objects.
- Primitive values are: strings ("John Doe"), numbers (3.14), true, false, null, and undefined.
- Objects are variables too. But objects can contain many values.

Example:

```
var person = {firstName:“Vijay",
   lastName:“Kumar", age:30 };
```

Function Definition:

The most common way to define a function in JavaScript is by using the function keyword, followed by a unique function name, a list of parameters (that might be empty), and a statement block surrounded by curly braces.

The basic syntax is shown here:

```
<script type="text/javascript">
<!-- function functionname(parameter-list)
{ statements }
//-->
</script>
```

Calling a Function:

To invoke a function somewhere later in the script, you would simple need to write the name of that function as follows:

```
<script type="text/javascript">
 <!-- sayHello(); //-->
</script>
```

Alert Dialog Box:

An alert dialog box is mostly used to give a warning message to the users.

```
<head>
 <script type="text/javascript">
 <!-- alert("Warning Message");
 //-->
</script>
</head>
```

Confirmation Dialog Box:

- A confirmation dialog box is mostly used to take user's consent on any option. It displays a dialog box with two buttons: **OK** and **Cancel**.

```
<head>
 <script type="text/javascript">
 <!-- var retVal = confirm("Do you want to continue ?");
 if( retVal == true )
{
 alert("User wants to continue!");
return true;
 }
else
{
 alert("User does not want to continue!");
  return false; } //-->
 </script>
</head>
```

Prompt Dialog Box:

- <head>
- <script type="text/javascript">
- <!-- var Val = prompt("Enter your name : ", "your name here");
- alert("You have entered : " + Val);
- //-->
- </script>
- </head>

Page Re-direction:

This is very simple to do a page redirect using JavaScript at client side.

```
<head>
 <script type="text/javascript">
 <!-- window.location="http://www.newlocation.com";
//-->
 </script>
 </head>
```

The Page Printing:

JavaScript helps you to implement this functionality using **print** function of *window* object.

```
<head>
    <script type="text/javascript">
    <!-- //-->
    </script>
</head>
<body>
    <form>
    <input type="button" value="Print" onclick="window.print()" />
    </form>
</body>
```

JavaScript Built-in Functions

- Number Methods
- Boolean Methods
- String Methods
- Array Methods
- Date Methods:
- Math Methods
- String HTML wrappers

Number Methods

Method	Description
constructor()	Returns the function that created this object's instance. By default this is the Number object.
toExponential()	Forces a number to display in exponential notation, even if the number is in the range in which JavaScript normally uses standard notation.
toFixed()	Formats a number with a specific number of digits to the right of the decimal.
toPrecision()	Defines how many total digits (including digits to the left and right of the decimal) to display of a number.
toString()	Returns the string representation of the number's value.
valueOf()	Returns the number's value.

Boolean Methods

Method	Description
toSource()	Returns a string containing the source of the Boolean object; you can use this string to create an equivalent object.
toString()	Returns a string of either "true" or "false" depending upon the value of the object.
valueOf()	Returns the primitive value of the Boolean object.

String Methods

Method	Description
charAt()	Returns the character at the specified index.
concat()	Combines the text of two strings and returns a new string.
indexOf()	Returns the index within the calling String object of the first occurrence of the specified value, or -1 if not found.
lastIndexOf()	Returns the index within the calling String object of the last occurrence of the specified value, or -1 if not found.
length()	Returns the length of the string.

replace()	**Used to find a match between a regular expression and a string, and to replace the matched substring with a new substring.**
search()	Executes the search for a match between a regular expression and a specified string.
substring()	Returns the characters in a string between two indexes into the string.
toLowerCase()	Returns the calling string value converted to lower case.
toString()	Returns a string representing the specified object.
toUpperCase()	Returns the calling string value converted to uppercase.
valueOf()	Returns the primitive value of the specified object.

Array Methods

Method	Description
concat()	Returns a new array comprised of this array joined with other array(s)
filter()	Creates a new array with all of the elements of this array for which the provided filtering function returns true.
forEach()	Calls a function for each element in the array.
indexOf()	Returns the first (least) index of an element within the array equal to the specified value, or -1 if none is found.
join()	Joins all elements of an array into a string.
pop()	Removes the last element from an array and returns that element.
push()	Adds one or more elements to the end of an array and returns the new length of the array.
reverse()	Reverses the order of the elements of an array – the first becomes the last, and the last becomes the first.
shift()	Removes the first element from an array and returns that element.
sort()	Sorts the elements of an array.
toString()	Returns a string representing the array and its elements.

Date Methods:

Method	Description
Date()	Returns today's date and time
getDate()	Returns the day of the month for the specified date according to local time.
getDay()	Returns the day of the week for the specified date according to local time.
getHours()	Returns the hour in the specified date according to local time.
getMilliseconds()	Returns the milliseconds in the specified date according to local time.
getMinutes()	Returns the minutes in the specified date according to local time.
getMonth()	Returns the month in the specified date according to local time.
getSeconds()	Returns the seconds in the specified date according to local time.
getTime()	Returns the numeric value of the specified date as the number of milliseconds since January 1, 1970, 00:00:00 UTC.

floor()	Returns the largest integer less than or equal to a number.
log()	Returns the natural logarithm (base E) of a number.
max()	Returns the largest of zero or more numbers.
min()	Returns the smallest of zero or more numbers.
pow()	Returns base to the exponent power, that is, base exponent.
random()	Returns a pseudo-random number between 0 and 1.
round()	Returns the value of a number rounded to the nearest integer.
sin()	Returns the sine of a number.
sqrt()	Returns the square root of a number.
tan()	Returns the tangent of a number.

Math Methods

Method	Description
abs()	Returns the absolute value of a number.
acos()	Returns the arccosine (in radians) of a number.
asin()	Returns the arcsine (in radians) of a number.
atan()	Returns the arctangent (in radians) of a number.
atan2()	Returns the arctangent of the quotient of its arguments.
ceil()	Returns the smallest integer greater than or equal to a number.
cos()	Returns the cosine of a number.

Regular Expression

- A regular expression is a certain way of representing a set of strings.
- Regular expressions are frequently used to test that a string entered in an HTML form has a certain format
- In the terminology of regular expressions, belongs to the set of strings that have the correct format.
- Simple example,
  - The set of strings that consist of exactly three
  - The set of valid area codes in a phone number

Pattern Matching

- Pattern matching based on methods of the ***String*** object are covered.
- *Meta characters* have special meaning in some contexts in patterns:
  - \ | ( ) [ ] { } ^ $ * + ? .

Pattern Matching

search(pattern)

- Returns position in the ***string*** object where the pattern matched

```
var str = "Hurricane";

var position = str.search(/can/);

/* postion is now 5 */
```

- If there is no match it returns-1
- The period matches any character except new line.

/snow . / *matches "snowy", "snowd", etc.*

Pattern Matching

- Placing desired characters in brackets allows class of characters
  - [abc] matches for 'a', 'b', or 'c'
  - [a-h] matches any lower case letter from 'a' to 'h‘
- Repeated character or character-class pattern
  - /xy{4}z/ matches for xyyyyz

Brackets 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]	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

Pattern Matching

- Predefined Character Classes
  - \d [0-9] A digit
  - \D [^0-9] Not a digit
  - \w [A-Za-z_0-9] An alphanumeric character
  - \W [^A-Za-z_0-9] Not an alphanumeric
  - \s[\r\t\n\f] A whitespace character
  - \S [^ \r\t\n\f] Not a whitespace character

Pattern Matching

- Symbolic quantifiers
  - * zero or more repetitions
  - + one or more repetitions
  - ? one or none repetitions
    - /x*y+z?/ any number of x, one or more y, may be a z
    - /\d+\.\d*/ a string of one or more digits followed by a decimal period and may be more digits.

Pattern Matching

Anchors

- Beginning of string (^):

  /^pearl/ matches patterns beginning with "pearl ..."

- Ending of string ($):

  /gold$/ matches patterns ending in " ... gold"

- Note anchor characters only have such meaning in these locations:

  /A^is$/ matches for /A^is$/ only

Metacharacters

Metacharacters are characters with a special meaning:

Metacharacter	Description
.	Find a single character, except newline or line terminator
\w	Find a word character
\W	Find a non-word character
\d	Find a digit
\D	Find a non-digit character
\s	Find a whitespace character
\S	Find a non-whitespace character
\b	Find a match at the beginning/end of a word, beginning like this: \bHI, end like this: HI\b
\B	Find a match, but not at the beginning/end of a word

\0	Find a NULL character
\n	Find a new line character
\f	Find a form feed character
\r	Find a carriage return character
\t	Find a tab character
\v	Find a vertical tab character
\xxx	Find the character specified by an octal number xxx
\xdd	Find the character specified by a hexadecimal number dd
\udddd	Find the Unicode character specified by a hexadecimal number dddd

Quantifiers

Quantifier	Description
n+	Matches any string that contains at least one *n*
n*	Matches any string that contains zero or more occurrences of *n*
n?	Matches any string that contains zero or one occurrences of *n*
n{X}	Matches any string that contains a sequence of *X* *n*'s
n{X,Y}	Matches any string that contains a sequence of X to Y *n*'s
n{X,}	Matches any string that contains a sequence of at least X *n*'s
n$	Matches any string with *n* at the end of it
^n	Matches any string with *n* at the beginning of it
?=n	Matches any string that is followed by a specific string *n*
?!n	Matches any string that is not followed by a specific string *n*

EXAMPLE

```
<html>
<head>
<script>
function myFunction() {
  var str = "Visit W3Schools";
  var patt = /w3schools/i;
  var result = str.match(patt);
  document.getElementById("demo").innerHTML = result;
}
</script>
</head>
<body>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
</body>
</html>
```