FUTURE VISION BIE

One Stop for All Study Materials

& Lab Programs

Future Vision

By K B Hemanth Raj

Scan the QR Code to Visit the Web Page

Or

Visit : https://hemanthrajhemu.github.io

Gain Access to All Study Materials according to VTU,
CSE – Computer Science Engineering,
ISE – Information Science Engineering,
ECE - Electronics and Communication Engineering
& MORE...

Join Telegram to get Instant Updates: https://bit.ly/VTU_TELEGRAM

Contact: MAIL: futurevisionbie@gmail.com

INSTAGRAM: www.instagram.com/hemanthraj_hemu/

INSTAGRAM: www.instagram.com/futurevisionbie/

WHATSAPP SHARE: https://bit.ly/FVBIESHARE

Module I
Introduction to HTML

What Is HTML and Where Did It Come from?

- HTML is a markup language
- First public specification of the HTML was published by Tim Berners-Lee in 1991
- HTML's codification by the World Wide Web Consortium (better known as the W3C) started in 1997.
- The Netscape Navigator and Microsoft Internet Explorer of the early and mid-1990s, a time when intrepid developers working for the two browser manufacturers ignored the W3C and brought forward a variety of essential new tags (such as, for instance, the <table> tag), and features such as CSS and JavaScript, all of which have been essential to the growth and popularization of the web.
- In 1998 the W3C froze the HTML specification at version 4.01.

XHTML

- In 1990s, the W3C developed a new specification called XHTML 1.0, which was a version of HTML that used stricter **XML** (extensible markup language) syntax rules.
- The goal of XHTML with strict rules was to make page rendering more predictable by forcing web authors to create web pages without **syntax errors.**

There are two versions of XHTML.
A) **XHTML 1.0 Transitional** and B) **XHTML 1.0 Strict**

- The **Transitional** version of HTML is the most common type of HTML. It has a flexible syntax, or grammar. Over the years, transitional HTML has been used without syntax restrictions. If tags are misspelled, the browsers do not correct web developer's errors, and they display the content anyway. Browsers do not report HTML errors, they simply display what they can.

- The **strict** version of HTML is meant to specify rules into HTML and make it more reliable. As a clean and error-free code helps to load pages faster, it uses the tag support described by the W3C XHTML 1.0 Strict specification. For example, the strict type requires closing all tags for all opened tags.

HTML versus XHTML

- There are some commonly heard arguments for using HTML rather than XHTML, especially XHTML 1.0 Strict. First, because of its less syntax rules, HTML is much **easier to write**, whereas XHTML requires a level of discipline many of us naturally resist.
- Because of the huge number of HTML documents available on the Web, browsers will continue to **support HTML** as far as one can see into the future. Indeed, some older browsers have problems with some parts of XHTML.
- There are strong reasons that one should use XHTML. One of the most compelling is

that quality and **consistency** in any endeavour

- HTML has few syntactic rules, and HTML processors (e.g., browsers) do not enforce the rules it does have. Therefore, HTML authors have a high degree of freedom to use their own syntactic preferences to create documents. Because of this freedom, HTML documents **lack consistency**, both in low-level syntax and in overall structure. XHTML has strict syntactic rules that impose a consistent structure on all XHTML documents.
- Another significant reason for using XHTML is that when you create an XHTML document, its syntactic correctness can be checked, either by an XML browser or by a **validation tool**.

XML

XML is a textual markup language, the formal rules for XML were set by the W3C. The XML-syntax rules are pretty easy to follow. The main rules are:

- There must be a single root element.
- Element names are composed of any of the valid characters (most punctuation symbols and spaces are not allowed) in XML.
- Element names can't start with a number.
- Element and attribute names are case sensitive.
- Attributes must always be within quotes.
- All elements must have a closing element (or be self-closing).

XML also provides a mechanism for validating its content : for instance, whether the text inside an element called <date> is actually a valid date, or the text within an element called <year> is a valid integer and falls between, say, the numbers 1950 and 2010.

HTML5

- A group of developers at Opera and Mozilla formed the **WHATWG** (Web Hypertext Application Technology Working Group) group within the W3C, and worked to enhance the features of HTML to higher versions.
- The WHATWG group was very small group led by Ian Hickson. By 2009, the W3C supported the WHATWG group and the work done by them and named it as HTML5.

There are three main aims of HTML5:

1. Specify unambiguously how browsers should deal with invalid markup.
2. Provide an open, nonproprietary programming framework (via JavaScript) for creating rich web applications.
3. Be backwards compatible with the existing web programs.

Structure of HTML Documents

A simple html program

```
<!DOCTYPE html>                          → Defines the type of document
<html>                                   → Root element – indicates that webpage is written in html
  <head>
      <title> A simple Program           → Contains title & information about this web page
      </title>
  </head>
  <body>
      <p> A simple program to            → Contents to be displayed on webpage
show the working of html
documents</p>
  </body>
</html>                                  → Closing of html document
```

The DOCTYPE (**Document Type Definition**) element informs the browser, the type of document it is about to process.
<!DOCTYPE html > specifies that the web page contains HTML code.

Head and Body

- HTML5 does not require the use of the <html>, <head>, and <body> elements. However, most web authors continue to use them.
- The <html> element is sometimes called the **root element** as it contains all the other HTML elements in the document.
- It has a **'lang'** attribute - This attribute tells the browser the natural language that is being used for textual content in the HTML document. Eg: <head lang = "en">
- HTML pages are divided into **two sections**: the **head** and the **body**, which correspond to the <head> and <body> elements.

The head contains descriptive elements *about* the document, such as its title, any style sheets or JavaScript files it uses, and other types of meta information used by search engines and other programs. The body contains content (both HTML elements and regular text) that will be displayed by the browser.

Eg:
```
<head>
    <meta charset = "utf-8" />
    <link rel = "stylesheet" href = "main.css" />
    <script src = "new.js"> </script>
    <title> Student activity </title>
<head>
```

- The <meta> element declares that the character encoding for the document is UTF- 8. Character encoding refers to which character set standard, is used to encode the characters in the document. **UTF-8** is a more complete variable-width encoding system that can encode all 110,000 characters in the Unicode character set.

- The <link> element specifies an external CSS style sheet file by name 'main.css' is used with this document. The style sheets is used to define the visual display of the HTML elements in the document. The styles can also be included within the document.

- The <script> element references an external JavaScript file. The JavaScript code can also be written directly within the HTML document.

- The <title> element is used to provide a broad description of the content. It is displayed by the browser in its window and/or tab.

Uses of <title> element –

1. To provide a broad description of content
2. Used by the browser for its bookmarks
3. Used by the browser to store the history list
4. Search engines use it as the linked text in their result pages.

HTML Syntax

Elements and Attributes

- HTML documents are composed of textual content and HTML elements.
- The term **HTML element** is often used interchangeably with the term **tag**.
- However, an HTML element consists of the element name within angle brackets (i.e., the tag) and the content within the tag.
- A tag consists of the element name within angle brackets.
- The element name appears in both the
- beginning tag and the closing tag.
- The closing tag contains a forward slash followed by the element name, all enclosed within angle brackets.

In the above example, <p> is the tag and "Hello I am a paragraph" is the content. HTML elements can also contain attributes.

- An **HTML attribute** is a 'name = value' pair that provides more information about the HTML element. In the above example, style is an attribute.

- An element which does not contain any text or image content is called an **empty element.** It is an instruction to the browser to do something.

Eg: <img> (image element) is an empty element.

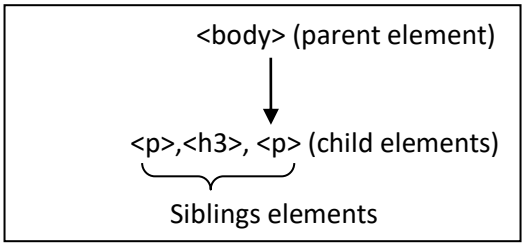
- The empty elements are terminated by a trailing slash.

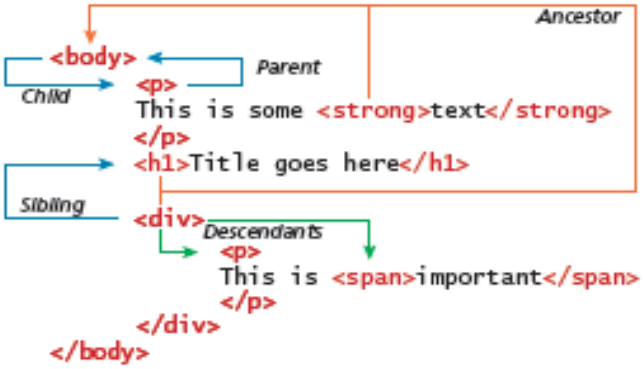
Eg: -------- empty element

Nesting HTML Elements

- An HTML element may contain other HTML elements. The container element is said to be a parent of the contained element (child).
- Any elements contained within the child are said to be **descendants** of the parent element.
- Any given child element, may have a variety of **ancestors**

```
<body>
    <p> This is some big <b> text </b>
     </p>
    <h3>A small title here </h3>
    <p> This is <span> important </span>
    </p>
</body>
```





- Each HTML element must be nested properly. That is, a child’s ending tag must occur before its parent’s ending tag. As shown in the above example, tag must be closed first and then the <p> tag is closed.

Semantic Markup

- HTML documents should focus on the structure of the document. Information about how the content should look when it is displayed in the browser is taken care by CSS(Cascading Style Sheets).
- HTML document should not describe how to visually present content, but only describe its content's structural semantics or meaning.
- Structure is a vital way of communicating information in paper and electronic documents. It makes it easier for the reader to quickly grasp the hierarchy of importance as well as broad meaning of information in the document.

Importance of writing HTML markup and its advantages:

- **Maintainability**. Semantic markup is easier to update and change than web pages that contain a great deal of presentation markup. More time is spent maintaining and modifying existing code than in writing the original code.
- **Faster**. Semantic web pages are typically quicker and faster to download.
- **Accessibility**. Not all web users are able to view the content on web pages. Users with sight disabilities experience the web using voice reading software. Visiting a web page using voice reading software can be a very frustrating experience if the site does not use semantic markup.
- **Search engine optimization**. The most important users of a website are the various search engine crawlers. These crawlers are automated programs that cross the web scanning sites for their content, which is then used for users' search queries. Semantic markup provides better instructions for these crawlers.

Some basic HTML Elements

HTML5 contains many structural and presentation elements, few elements are explained below -

Headings

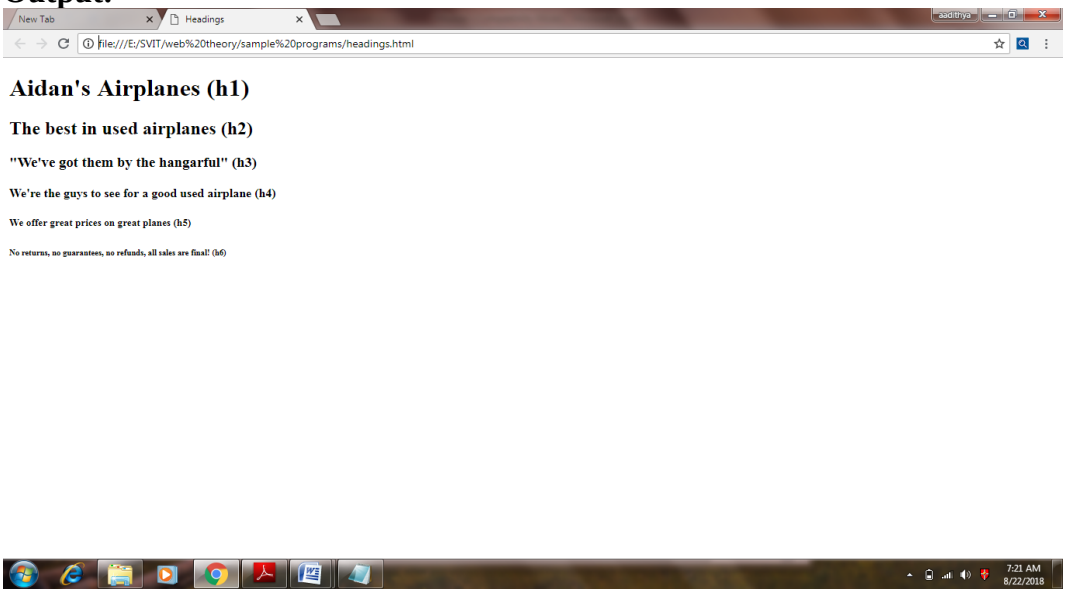
- Text is often separated into sections in documents by beginning each section with a heading.
- Larger sections sometimes have headings that appear more prominent than headings for sections nested inside them.
- In XHTML, there are six levels of headings, specified by the tags <h1>, <h2>, <h3>, <h4>, <h5>, and <h6>, where <h1> specifies the **highest-level heading**.
- Headings are usually displayed in a **boldface font** whose default size depends on the number in the heading tag.
- On most browsers, <h1>, <h2>, and <h3> use font sizes that are larger than that of the default size of text, <h4> uses the default size, and <h5> and <h6> use smaller sizes.

- The heading tags always break the current line, so their content always appears on a new line.

The following example illustrates the use of headings:

```
<!DOCTYPE html>
<!-- headings.html An example to illustrate headings-->
<head> <title> Headings </title>
</head>
<body>
        <h1> Aidan's Airplanes (h1) </h1>
        <h2> The best in used airplanes (h2) </h2>
        <h3> "We've got them by the hangarful" (h3) </h3>
        <h4> We're the guys to see for a good used airplane (h4) </h4>
        <h5> We offer great prices on great planes (h5) </h5>
        <h6> No returns, no guarantees, no refunds, all sales are final! (h6) </h6>
</body>
</html>
```

Output:

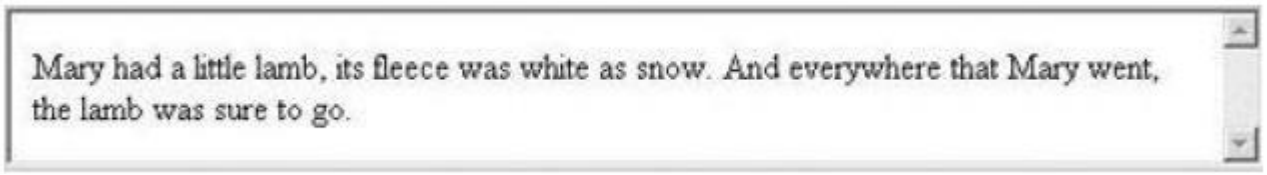


Paragraphs and Divisions

- Text is normally organized into paragraphs in the body of a document. T
- he HTML standard does not allow text to be placed directly in a document body. Instead, textual paragraphs appear as the content of a paragraph element, specified with the tag <p>.
- The <p> tag leaves a line before and after the tag. In displaying the content of a paragraph, the browser puts as many words as will fit on the lines in the browser window. The browser supplies a line break at the end of each line.

For example, the following paragraph might be displayed by a browser as shown below.

```
<p>
Mary had
        a
        little lamb, its fleece was white as
snow. And everywhere that
                Mary went, the lamb
was sure to go.
</p>
```



Notice that multiple spaces in the source paragraph element are replaced by single Spaces. If the content of a paragraph tag is displayed at a position other than the beginning of the line, the browser breaks the current line and inserts a blank line. For example, the following line would be displayed as shown below.

```
<p> Mary had a little lamb, </p> <p> its
        fleece was white as snow. </p>
```

The <p> tag is a container and can contain HTML and other **inline HTML elements** (the and <a> elements). Inline HTML elements are elements that do not cause a paragraph break but are part of the regular flow of the text.
The element is also a container element and is used to create a logical grouping of content

Links (anchor tag <a>)

Links are an essential feature of all web pages. Links are created using the <a> element (the "a" stands for anchor).
<a> is an inline tag. A link has two main parts: the destination and the label. The label of a link can be text or another HTML element such as an *image*.

destination label

 click here

<img src = “image1.jpg”>

- A link specifies the address of the destination. Such an address might be a file name, a directory path and a file name, or a complete URL.
- The document whose address is specified in a link is called the target of that link.
- The value assigned to href (hypertext reference) specifies the target of the link. If the target is in another document in the same directory, the target is just the document’s file name.

If the target file is in a subdirectory named ‘lab’, then the access to file is done by specifying the directory name. < a href = “lab/next.html”> clickhere.

```
<!DOCTYPE html >
<html>
<head>
    <title >sdfsfdfddddd</title>
</head>
<body>
    <p> Mary had a little
    <a href="lab/lamb.jpg"> lamb,</a>
    </p> <p> its fleece was white as snow. </p>
</body>
</html>
```

Output:

Mary had a little lamb,
Its fleece was white as snow.

Anchor element <a> can be used to create a wide range of links. These include:

- Links to external file (or to individual resources such as images or movies on an external site).
- Links to other pages or resources within the current file.
- Links to other places within the current page.

- Links to particular locations on another page (whether on the same site or on an external site).
- Links that are instructions to the browser to start the user's email program.
- Links that are instructions to the browser to execute a JavaScript function.
- Links that are instructions to the mobile browser to make a phone call.

Targets within Documents

If the target of a link is not at the beginning of a document, it must be some element within the document, in which case there must be some means of specifying it. The target element can include an id attribute, which can then be used to identify it in an href attribute. Consider the following example:

<h2 id = "avionics"> Avionics </h2>

Nearly all elements can include an id attribute. The value of an id attribute must be unique within the document. If the target is in the same document as the link, the target is specified in the href attribute value by preceding the id value with a pound sign (#), as in the following example:

* What about avionics? *

When the What about avionics? link is taken; the browser moves the display so that the h2 element whose id is *avionics* is at the top.

When the target is a part or fragment of another document, the name of the part is specified at the end of the URL, separated by a pound sign (#), as in this example:

* Avionics *

One common use of links to parts of the same document is to provide a table of contents in which each entry has a link. This technique provides a convenient way for the user to get to the various parts of the document simply and quickly

Link to external site

```
<a href="http://www.centralpark.com">Central Park</a>
```

Link to resource on external site

```
<a href="http://www.centralpark.com/logo.gif">Central Park</a>
```

Link to another page on same site as this page

```
<a href="index.html">Home</a>
```

Link to another place on the same page

```
<a href="#top">Go to Top of Document</a>
...
<a name="top">
```

Defines anchor for a link to another place on same page

Link to specific place on another page

```
<a href="productX.html#reviews">Reviews for product X</a>
```

Link to email

```
<a href="mailto:person@somewhere.com">Someone</a>
```

Link to JavaScript function

```
<a href="javascript:OpenAnnoyingPopup();">See This</a>
```

Link to telephone (automatically dials the number when user clicks on it using a smartphone browser)

```
<a href="tel:+18009220579">Call toll free (800) 922-0579</a>
```

URL Relative Referencing

- To construct links with the <a> element, reference images with the <img> element, or include external JavaScript or CSS files, the files should be successfully referred using **relative referencing** from the document.
- If the referred file is an external file then **absolute reference** is required. Absolute path contains the full path including the domain name, any paths, and then finally the file name of the desired resource.
- However, when referencing a resource that is on the same server as the HTML document, then **relative referencing** is done. Relative paths change depending upon where the links are present.

There are several rules to create a link using the relative path:

- links in the same directory as the current page have no path information listed **filename**
- sub-directories are listed without any preceding slashes **weekly/filename**
- links up one directory are listed as **../filename**

Sl. No.	Relative Link Type	Example
1	**Same Directory** – To link to a file within the same folder, simply use the file name.	<a href = "ex.html">
2	**Child Directory** – To link to a file within a subdirectory, use the name of subdirectory and a slash before the file name.	<a href = "images/ex.html">
3	**Grandchild/Descendant Directory** – To link to a file that is multiple subdirectories below the current one, construct the full path by including each subdirectory name before the file name.	<a href = "css/images/ex.html">
4	**Parent/Ancestor Directory** – use "../" to reference a folder above the current one. If trying to reference a file several levels above the current one, simply string together multiple "../".	<a href = "../ex.html"> <a href = "../../ex.html">
5	**Sibling Directory** – use "../" to move up to the appropriate level, and then use the same technique as for child or grandchild directories.	<a href = "../ex.html"> <a href = "../images/ex.html">
6	**Root Reference –** An alternative approach for ancestor and sibling references is to use the root reference approach. Ie. Begin the reference with the root reference ("/") and then use the same technique as for child or grandchild directories.	<a href = "/ex.html"> <a href = "/images/ex.html">
7	**Default Document** -Web servers allow references to directory names without file names. In such a case, the web server will serve the default document.	<a href = "members"> or <a href = "/members">

Inline Text Elements

Few HTML elements do not disrupt the flow of text (i.e., do not cause a line break), they are called inline elements. Eg: ,<u>,<i>,<b>, <a>, , <time>, <small> elements).

Element	Description
<a>	Anchor used for hyperlinks.
<abbr>	An abbreviation
 	Line break
<cite>	Citation (i.e., a reference to another work).
<code>	Used for displaying code, such as markup or programming code.
	Emphasis
<mark>	For displaying highlighted text
<small>	For displaying the fine-print, i.e., "non-vital" text, such as copyright or legal notices.
	The inline equivalent of the element. It is generally used to mark text that will receive special formatting using CSS.
	For content that is strongly important.
<time>	For displaying time and date data

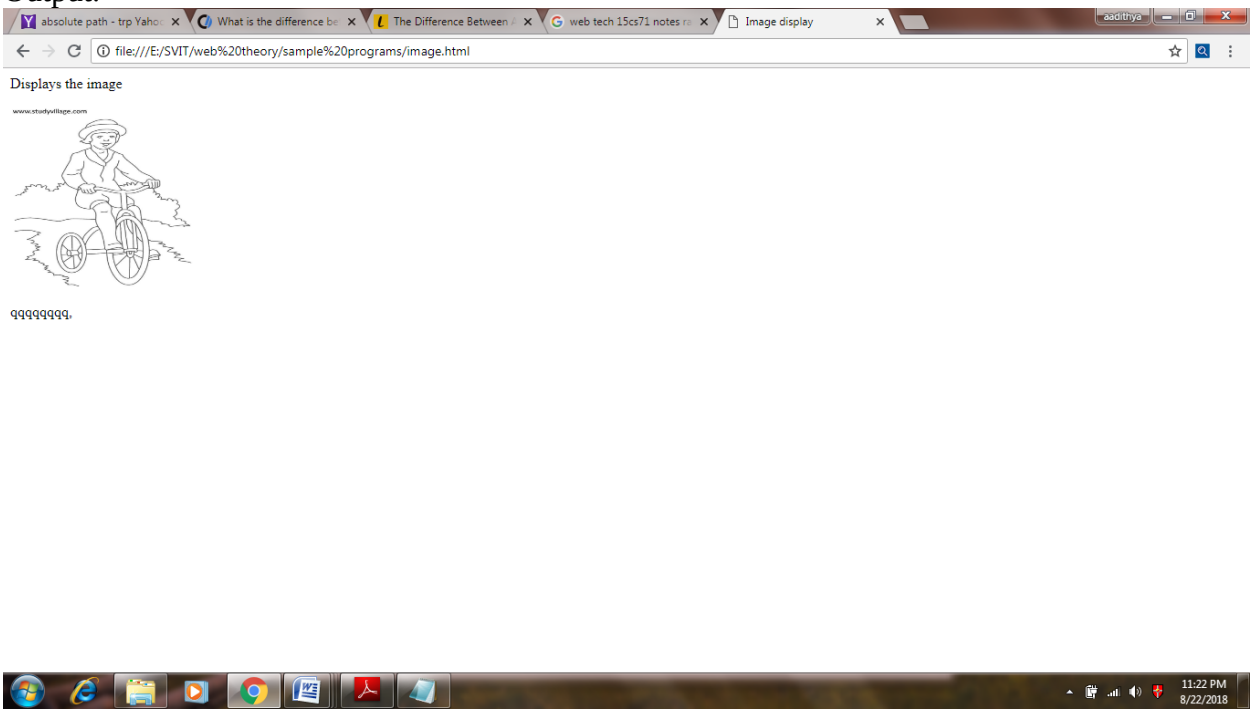
Images
The <img> tag is the oldest method for displaying an image.

src attribute - specifies the file containing the image;
alt - specifies text to be displayed when it is not possible to display the image.
title – specifies the title to be displayed in pop-up tool tip when user moves mouse over image.
width and height - can be included to specify (in pixels) the size of the rectangle for the image

Eg:

```
<html>
<head>
<head> <title>Image display</title>
</head>
<body>
<p >Displays the image</p>
<img  src="cycle-riding.jpg" width="200" height="200">
<p >qqqqqqqq,</p>
</body>
</html>
```

Output:



Character entities

- These are special characters for symbols for which there is either no easy way to type them via a
- keyboard (such as the copyright symbol or accented characters) or which have a reserved meaning in HTML (for instance the "<" or ">" symbols).
- There are many HTML character entities. They can be used in an HTML document by using the entity name or the entity number.

Entity Name	Entity Number	Description
		Nonbreakable space. The browser ignores multiple spaces in the source HTML file. If you need to display multiple spaces, you can do so using the nonbreakable space entity.
&lt;	&#60;	Less than symbol ("<").
&gt;	&#62;	Greater than symbol (">").
&copy;	&#169;	The © copyright symbol
&euro;	&#8364;	The € euro symbol.
&trade;	&#8482;	The ™ trademark symbol.
&uuml;	&#252;	The û—i.e., small u with umlaut mark.

&amp;	Ampersand
&quot;	Double quote
&apos;	Single quote (apostrophe)
&deg;	Degree

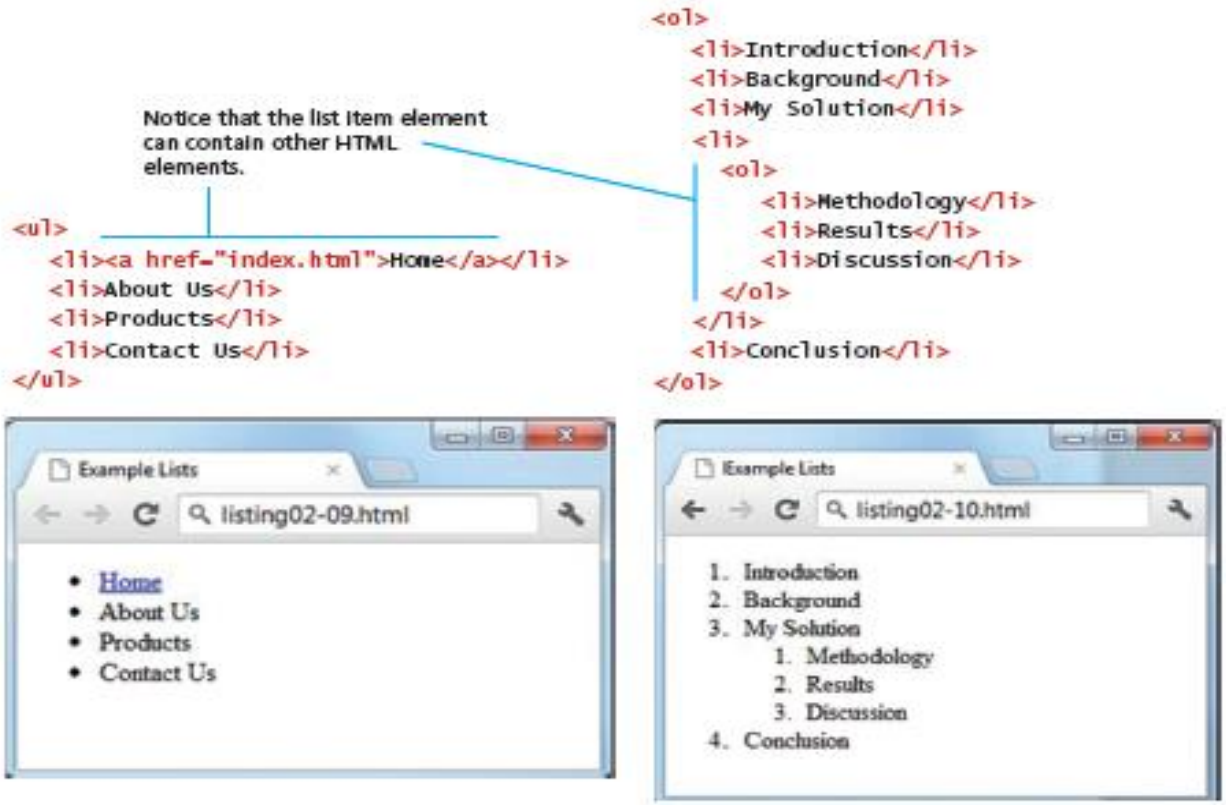
Lists

HTML provides simple and effective ways to specify lists in documents.
There are three types of lists:

■ **Unordered lists**. Collections of items in no particular order; these are by default rendered by the browser as a bulleted list. However, it is common in CSS to style unordered lists without the bullets. Unordered lists have become the conventional way to markup navigational menus.

■ **Ordered lists**. Collections of items that have a set order; these are by default rendered by the browser as a numbered list.

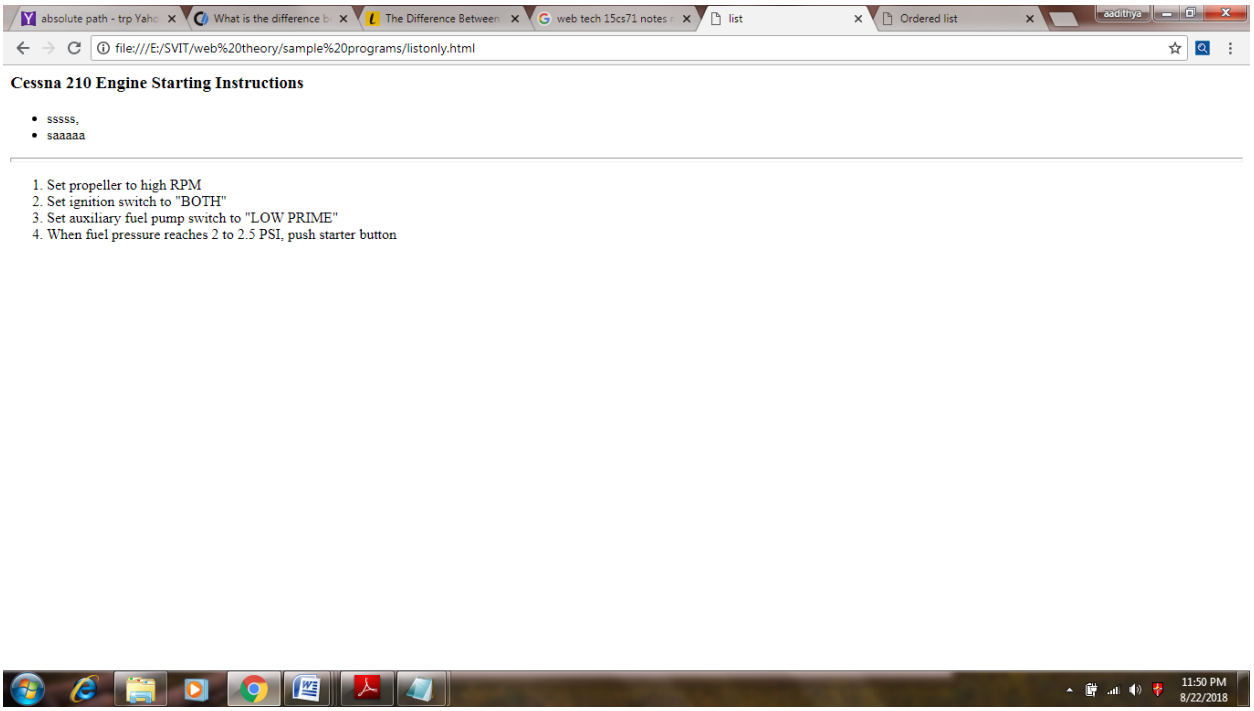
■ **Definition lists**. Collection of name and definition pairs. These tend to be used infrequently. Perhaps the most common example would be a FAQ list.




```
<html >
<head> <title> list </title>
</head>
<body>
    <h3> Cessna 210 Engine Starting Instructions </h3>
    <ul >
        <li>sssss,</li>
```

```
        <li>saaaaa</li>
    </ul>
    <hr size="5"  />
    <ol >
        <li> Set propeller to high RPM </li>
        <li> Set ignition switch to "BOTH" </li>
        <li> Set auxiliary fuel pump switch to "LOW PRIME" </li>
        <li> When fuel pressure reaches 2 to 2.5 PSI, push starter button </li>
    </ol>
</body>
</html>
```

Output:



HTML5 Semantic Structure Elements

- The different sections of a website are usually divided by using tag.
- Most complex websites are packed solid with elements. Most of these are marked with different id or class attributes, that are styled by using various CSS.
- But many elements make the markup confusing and hard to modify.
- Developers typically give suitable names to id or class of , so as to provide some clue to understand what that part of code means. The new elements help the bots to view sites and the codes more like people.
- The idea behind using new semantic block structuring elements in HTML5 is to make it easier to understand, what a particular part of document does.

- Some of tag is replaced with self-explanatory HTML5 elements. There is no predefined presentation for these new tags. The new semantic elements in HTML5 are listed below -
  1. Headers and Footer
  2. Heading Groups
  3. Navigation
  4. Articles and Sections
  5. Figure and Figure Captions
  6. Aside

Header and Footer

Most website pages have a recognizable header and footer section. Typically the header contains the site logo and title (and perhaps additional subtitles or taglines), horizontal navigation links, and perhaps one or two horizontal banners. The typical footer contains less important material, such as smaller text versions of the navigation, copyright notices, information about the site's privacy policy, and perhaps twitter feeds or links to other social sites.

Both the HTML5 <header> and <footer> element can be used not only for *page* headers and footers, but also for header and footer elements within other HTML5 containers, such as <article> or <section>.

```
<header>
<img src="logo.gif" alt="logo" />
<h1>Fundamentals of Web Development</h1>
...
</header>
<article>
   <header>
      <h2>HTML5 Semantic Structure Elements</h2>
      <p>By <em>Randy Connolly</em></p>
      <p><time>September 30, 2015</time></p>
   </header>
   ...
</article>
```

Heading Groups

A header may contain multiple headings <hgroup> element is usually used in such cases. The <hgroup> element can be used in contexts other than a header. For instance, one could also use an <hgroup> within an <article> or a <section> element. The <hgroup> element can *only* contain <h1>, <h2>, etc., elements.

```
<header>
   <hgroup>
   <h1>Chapter Two: HTML 1</h1>
   <h2>An Introduction</h2>
   </hgroup>
</header>
<article>
   <hgroup>
   <h2>HTML5 Semantic Structure Elements</h2>
   <h3>Overview</h3>
   </hgroup>
</article>
```

Navigation

The <nav> element represents a section of a page that contains links to other pages or to other parts within the same page. Like the other new HTML5 semantic elements, the browser does not apply any special presentation to the <nav> element. The <nav> element was intended to be used for major navigation blocks. However, like all the new HTML5 semantic elements, from the browser's perspective, there is no definite right or wrong way to use the <nav> element. Its sole purpose is to make the document easier to understand.

```
<header>
  <img src="logo.gif" alt="logo" />
  <h1>Fundamentals of Web Development</h1>
  <nav role="navigation">
    <ul>
       <li><a href="index.html">Home</a></li>
       <li><a href="about.html">About Us</a></li>
       <li><a href="browse.html">Browse</a></li>
    </ul>
  </nav>
</header>
```

Articles and Sections

The new HTML5 semantic elements <section> and <article> are used to group the tags. Suppose a book is divided into smaller blocks of content called chapters, which makes the book easier to read. If each chapter is further divided into sections (and these sections into even smaller subsections), this makes the content of the book easier to manage for both the reader and the authors.

The article element represents a section of content that forms an independent part of a document or site. The section element represents a section of a document, typically with a title or heading. It is a broader element.

Figure and Figure Captions
Prior to HTML5, web authors typically wrapped images and their related captions within a nonsemantic element. In HTML5 we can instead use the <figure> and <figcaption> elements. *The figure element represents some flow content, optionally with a caption, that is self-contained and is typically referenced as a single unit from the main flow of the document.*

```
<p>This photo was taken on October 22, 2011 with a Canon EOS 30D camera.</p>
<figure>
    <img src="images/central-park.jpg" alt="Central Park" /><br/>
    <figcaption>Conservatory Pond in Central Park</figcaption>
</figure>
</p>
```

The above tags illustrates a sample usage of the <figure> and <figcaption> element.

Aside
The <aside> element is similar to the <figure> element, the <aside> element "represents a section of a page that consists of content that is indirectly related to the content around the aside element". The <aside> element is be used for sidebars, pull quotes, groups of advertising images, or any other grouping of non-essential elements.

Cascading Style Sheets (CSS)

- CSS is a W3C standard for describing the appearance of HTML elements. It is used to define the **presentation** of HTML documents.
- Using CSS with HTML elements, we can assign font properties, colors, sizes, borders, background images, and even position elements on the page.
- In early 1990s, a variety of different style sheet standards were proposed, including JavaScript style sheets, which was proposed by Netscape in 1996.
- Netscape's proposal was one that required the use of JavaScript programming to perform style changes. Due to many nonprogrammers everywhere, the W3C decided to adopt CSS, and by the end of 1996 the CSS Level 1 Recommendation was published. The latest version is CSS Level 4.
- CSS can be added directly to any HTML element (via the style attribute), or within the <head> element, or in a separate text file that contains only CSS.

Benefits of CSS
■ **Improved control over formatting -** CSS gives website developers fine-grained control over the appearance of their web content. The contents are better formatted using CSS.

■ **Improved site maintainability -** Websites is easily maintainable, as all formatting can be put in a single CSS file.

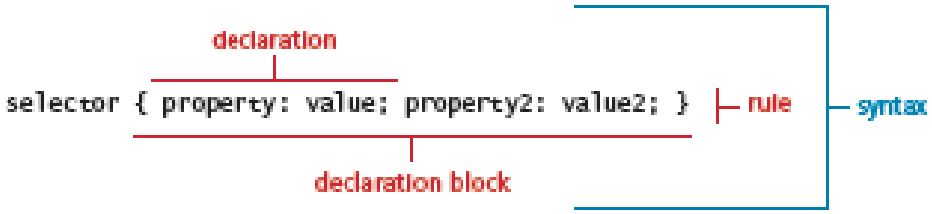
■ **Improved accessibility -** CSS-driven sites are more accessible.

■ **Improved page download speed** - A web site that uses a single CSS files will be quicker to download, because each individual HTML file will contain less style information and markup, and thus be smaller.

■ **Improved output flexibility** - CSS can be used to adopt a page for different output devices with varying sizes. This approach to CSS page design is often referred to as **responsive design**.

CSS Syntax

- A CSS rule consists of a **selector** that identifies the HTML elements that will be affected, followed by a series of **property:value pairs** (each pair is also called a **declaration**), as shown in Figure below.
- The series of declarations is called the **declaration block**. A declaration block can be together on a single line, or spread across multiple lines.
- The browser ignores white space (i.e., spaces, tabs, and returns) between your CSS rules so you can format the CSS however you want. Each declaration is terminated with a semicolon. The semicolon for the last declaration in a block is in fact optional.



Eg:
p{color: red; font-weight:bold;} =>here p is the selector; the declarations are implied to the p tag.

h2,p { font-size:40; color:green;}=> the declarations are implied to both h2 & p tags.

Selectors
Every CSS rule begins with a **selector**. The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule. They are a pattern that is used by the browser to select the HTML elements that will receive the style.

Properties
Each individual CSS declaration must contain a property. These property names are predefined by the CSS standard. The CSS recommendation defines over a hundred different property names. Eg: font-family, font-size, font-style, font-weight, text-align, text-decoration, background-color, background-image, border-color, border-width, border-style, border-top, etc.

Values
Each CSS declaration also contains a value for a property. The unit of any given value is dependent upon the property. Most of the property values are from a predefined list of keywords. Eg: px, em, %, In, cm, mm etc. are some of the units for measuring values.

Location of Styles (Levels of Stylesheets)

CSS style rules can be located in three different locations, in order from lowest level to highest level, are **inline, document level, and external**.

Inline Styles

Inline styles are style rules placed within an HTML element using the style attribute, as shown below. An inline style only affects the element it is defined within and overrides any other style definitions. Selector is not necessary with inline styles and that semicolons are only required for separating multiple rules.

Disadvantages of using inline style-
Style is applied to an element only
Maintaining the inline style is difficult

The advantage of using inline style is that it can be quickly tested for a style change.

Eg: <h2 style = “font-size:24pt;”> Description</h2>
<h2 style = “font-size:24pt; font-weight:bold;”> Reviews </h2>

Embedded Style Sheet (Document Level/Internal)

Embedded style sheets (also called **internal styles or document level styles**) are style rules placed within the <style> element (inside the <head> element of an HTML document) and apply to the whole body of the document.

The disadvantage of using embedded styles is that it is difficult to consistently style multiple documents when using embedded styles. But it is helpful when quickly testing out a style that is used in multiple places within a single HTML document. Spaces are ignored in <style> element.

```
<head>
<title>Student Data</title>
<style>
h1 { font-size: 24pt; }
h2 {
font-size: 18pt;
font-weight: bold;
}
</style>
</head>
<body>
<h1>Student count</h1>
<h2>CSE/ISE Department</h2>
……..
</body>
```

External Style Sheet

External style sheets are style rules placed within an external text file with the .**css** extension. This style provides the best maintainability. When you make a change to an external style sheet, all HTML documents that reference that style sheet will automatically use the updated version.

To reference an external style sheet, you must use a <link> element (within the <head> element). Several style sheets can be linked at a same time. Each linked style sheet will require its own <link> element.

```
<head >
<title>Share Your Travels -- New York - Central Park</title>
<link rel="stylesheet" href="styles.css" />
</head>
```

Selectors

The selector identifies which element or elements in the HTML document will be affected by the declarations in the rule. They are a pattern that is used by the browser to select the HTML elements that will receive the style.

- Element Selectors
- Class Selectors
- Id Selectors
- Attribute Selectors
- Pseudo-Element and Pseudo-Class Selectors
- Contextual Selectors

Element Selectors

Element selectors select an element or group of elements of the HTML document, and the properties are applied on it.

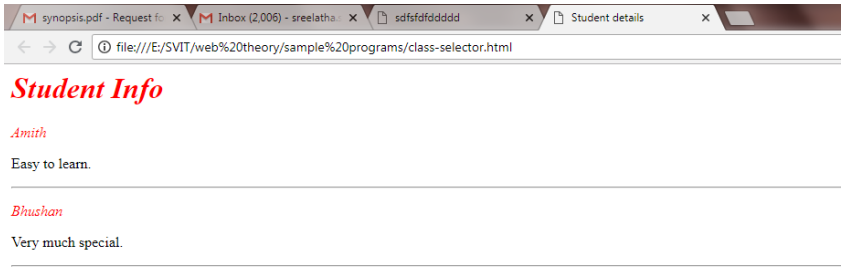
The group of elements are separated using commas is called **grouped selector**.

Universal element selector - All elements of the document can be selected by using the * (asterisk) character.

Eg of element selector -

```
<head>
<title>Student details </title>
<style>
*{ color:blue;}
h1 {
```

Output:



```
color: red;
}
</style>
</head>
<body>
<h1 >Student Info</h1>
<div>
<p >Amith</p>
<p>Easy to learn.</p>
</div>
<hr/>
<div>
<p >Bhushan</p>
<p>Very much special.</p>
</div>
<hr/>
</body>
```

Eg of grouped selector

```
p, div, h3 {
margin: 0;
padding: 0;
}
```

Eg of universal element selector

```
*{
    color : red;
  }
```

Class Selectors

A **class selector** allows to simultaneously target different HTML elements. The HTML elements with the same class attribute value, can be styled by using a class selector.
Syntax: period (.)classname{ styles}

Eg:

```
<head>
    <title>Student details </title>
    <style>
    .first {
    font-style: italic;
    color: red;
    }
    </style>
</head>
<body>
     <h1 class="first">Student Info</h1>
     <div>
     <p class="first">Amith</p>
```

Output:

```
    <p>Easy to learn.</p>
    </div>
    <hr/>
    <div>
    <p class="first">Bhushan</p>
    <p>Very much special.</p>
  </div>
  <hr/>
</body>
```

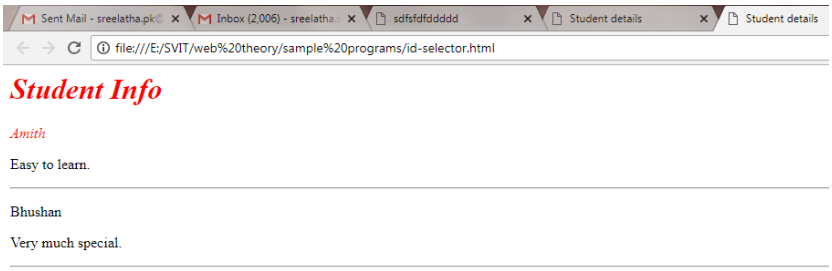
Id Selectors

An **id selector** allows to assign style to a specific element by its id attribute.
Syntax: hash (#)id name

Eg:

```
<head>
    <title>Student details </title>
    <style>
    #first {
    font-style: italic;
    color: red;
    }
    </style>
</head>
<body>
    <h1 id="first">Student Info</h1>
    <div>
    <p id="first">Amith</p>
    <p>Easy to learn.</p>
    </div>
    <hr/>
    <div>
    <p >Bhushan</p>
    <p>Very much special.</p>
    </div>
    <hr/>
</body>
```

Output:



Attribute Selectors

An **attribute selector** provides a way to select HTML elements either by the presence of an element attribute or by the value of an attribute.
Eg: [src], [src$=".jpg"] , a[href*="gala"] etc.

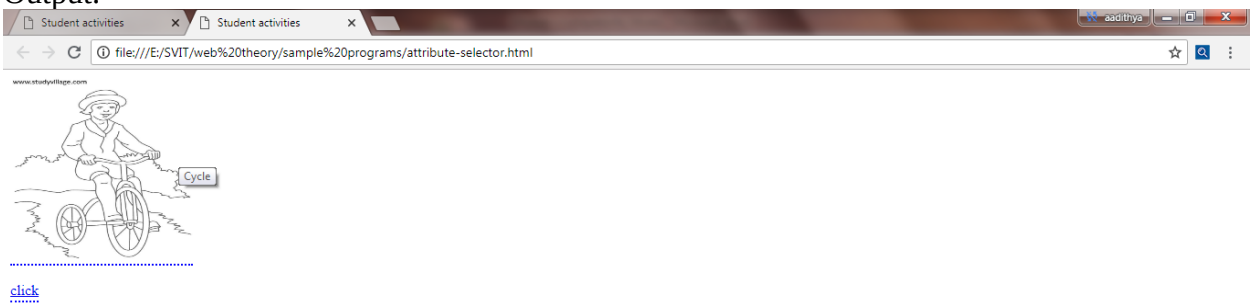
[src] – selects all the elements which have 'src' as an attribute
[src$=".jpg"] – selects all the elements with 'src' value ending with .jpg
a[href*="gala"] – selects <a> tag with 'href' value having text 'gala'.

Attribute selectors is very helpful technique in the styling of hyperlinks and images. Suppose, we want special attention of user when a pop-up tooltip is available for a link or image. This can be done by using the following attribute selector:
[title] { … }

Eg:

```
<head >
    <title>Student activities</title>
    <style>
    [title] {
    cursor: help;
    padding-bottom: 3px;
    border-bottom: 2px dotted blue;
    }
    </style>
</head>
<body>
<div>
<img src="cycling.jpg" title="Cycle" />
<a href = “s1.jpg” title= “link to photo”> click </a>
</div>
</body>
```

Output:



Selector	Matches	Example
[]	A specific attribute.	`[title]` Matches any element with a title attribute
[=]	A specific attribute with a specific value.	`a[title="posts from this country"]` Matches any <a> element whose title attribute is exactly `"posts from this country"`
[~=]	A specific attribute whose value matches at least one of the words in a space-delimited list of words.	`[title~="Countries"]` Matches any `title` attribute that contains the word `"Countries"`
[^=]	A specific attribute whose value begins with a specified value.	`a[href^="mailto"]` Matches any <a> element whose `href` attribute begins with `"mailto"`
[*=]	A specific attribute whose value contains a substring.	`img[src*="flag"]` Matches any <img> element whose `src` attribute contains somewhere within it the text `"flag"`
[$=]	A specific attribute whose value ends with a specified value.	`a[href$=".pdf"]` Matches any <a> element whose `href` attribute ends with the text ".pdf"

Pseudo-Element and Pseudo-Class Selectors

A **pseudo-element selector** is a way to select something that does not exist explicitly as an element in the HTML document but which is still a recognizable selectable object.
For instance, first line or first letter of any HTML element.

Selector	Type	Description
`a:link`	pseudo-class	Selects links that have not been visited
`a:visited`	pseudo-class	Selects links that have been visited
`:focus`	pseudo-class	Selects elements (such as text boxes or list boxes) that have the input focus.
`:hover`	pseudo-class	Selects elements that the mouse pointer is currently above.
`:active`	pseudo-class	Selects an element that is being activated by the user. A typical example is a link that is being clicked.
`:checked`	pseudo-class	Selects a form element that is currently checked. A typical example might be a radio button or a check box.
`:first-child`	pseudo-class	Selects an element that is the first child of its parent. A common use is to provide different styling to the first element in a list.
`:first-letter`	pseudo-element	Selects the first letter of an element. Useful for adding drop-caps to a paragraph.
`:first-line`	pseudo-element	Selects the first line of an element.

A **pseudo-class selector** does apply to an HTML element, but targets a particular state.

The most common use of this type of selectors is for targeting link states. By default, the browser displays link text blue and visited text links purple.

Contextual Selectors

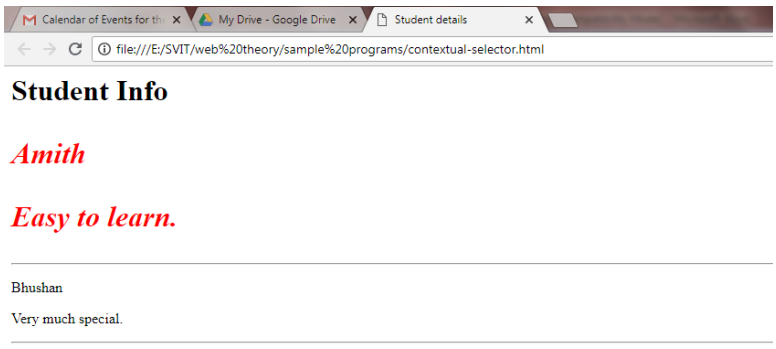
A **contextual selector** (in CSS3 also called **combinators**) allows to select elements based on their *ancestors*, *descendants*, or *siblings*. It selects elements based on their context or relation to other elements in the document tree.

As shown in below table, **descendant selector** matches all elements that are contained within another element. The character used to indicate descendant selection is the space character.

Selector	Matches	Example
Descendant	A specified element that is contained somewhere within another specified element.	`div p` Selects a <p> element that is contained somewhere within a element. That is, the <p> can be any descendant, not just a child.
Child	A specified element that is a direct child of the specified element.	`div>h2` Selects an <h2> element that is a child of a element.
Adjacent sibling	A specified element that is the next sibling (i.e., comes directly after) of the specified element.	`h3+p` Selects the first <p> after any <h3>.
General sibling	A specified element that shares the same parent as the specified element.	`h3~p` Selects all the <p> elements that share the same parent as the <h3>.

```
<head>
    <title>Student details </title>
    <style>
    #first  p{
    font-style: italic;
    color: red;
    }
    </style>
</head>
<body>
    <h1 id="first">Student Info
    <div>
    <p >Amith</p>
    <p>Easy to learn.</p>
    </div>
    </h1>
    <hr/>
    <div>
    <p >Bhushan</p>
```

Output:



```
    <p>Very much special.</p>
    </div>
    <hr/>
</body>
```

The Cascade: How Styles Interact

Multiple CSS rules can be defined for the same HTML element, at different locations – inline, embedded or external. The browser determines the style to be applied on an element, depending on the location and hierarchy of the html element.
The "Cascade" in CSS refers to how conflicting rules are handled. CSS uses the following cascade principles to help it deal with conflicts: inheritance, specificity, and location.

Inheritance

Inheritance is the first of these cascading principles. Many (but not all) CSS properties affect not only themselves but their descendants as well. Font, color, list, and text properties are inheritable; layout, sizing, border, background, and spacing properties are not inheritable.

If suppose, this is a document,

```
<head>
<style>
    body {
    font-family: Arial;
    color: red;
    border: 8pt solid green;
    margin: 100px;
    }

    div {
    font-weight: bold;
    }
</style>
</head>
<body>
<div>Will be displayed in red, with arial font and bold</div>
</body>
```

The font settings are inherited from the parent tag, border and margin are not inheritable.
However it is possible to tell elements to inherit properties that are normally not inheritable, by explicitly specifying as 'inherit'.

```
    div {
    font-weight: bold;
    border: inherit;
    margin: inherit;
    }
```

Specificity

Specificity is how the browser determines which style rule takes precedence when more than one style rule could be applied to the same element. In CSS, the more specific the selector, the more it takes precedence (i.e., overrides the previous definition)

```
<head>
<style>
    body {
    font-family: Arial;
    color: red;
    border: 8pt solid green;
    margin: 100px;
    }

    div {
    font-weight: bold;
    color: blue;
    }
</style>
</head>
<body>
<div>Will be displayed in blue, with arial font and bold</div>
</body>
```

The content of is displayed in blue, as the red color setting of <body> tag is overridden in the specification of tag.

Location

The principle of location is that when rules have the same specificity, then the latest are given more weight. Ie., an inline style will override one defined in an embedded style sheet and embedded style will override the external style sheet.

Styles defined in external style sheet X will override styles in external style sheet Y if X's <link> element is after Y's in the HTML document.

<link rel= "stylesheet" href= "Y">
<link rel= "stylesheet" href= "X">

When the same style property is defined multiple times within a single declaration block, the last one will take precedence.

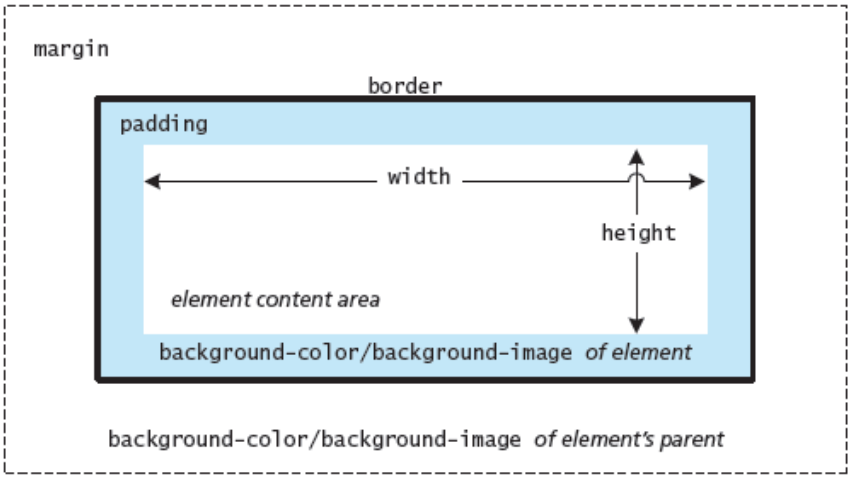
Specificity algorithm:

■ First count 1 if the declaration is from a "style" attribute in the HTML, 0 otherwise (let that value = a).
■ Count the number of ID attributes in the selector (let that value = b).
■ Count the number of class selectors, attribute selectors, and pseudo-classes in the selector (let that value = c).
■ Count the number of element names and pseudo-elements in the selector (let that value = d).
■ Finally, concatenate the four numbers a+b+c+d together to calculate the selector's specificity.

The following sample selectors are given along with their specificity value.
<tag style="color: red"> 1000
body .example 0011
body .example strong 0012
div#first 0101
div#first .error 0111
#footer .twitter a 0111
#footer .twitter a:hover 0121
body aside#left div#cart strong.price 0214

The Box Model

In CSS, all HTML elements exist within a rectangular **element box** shown in Figure.



The background color or image of an element fills an element within its border.
Some of the common background properties are –

Property	Description
background	A combined shorthand property that allows you to set multiple background property.
background-attachment	Specifies whether the background image scrolls with the document (default) or remains fixed. values are: *fixed, scroll*
background-color	Sets the background color of the element.
background-image	Specifies the background image
background-position	Specifies where background image will be placed. Possible values include: *bottom, center, left, and right*. Pixel or percentage numeric position value may also be specified.
background-repeat	Determines whether the background image will be repeated – for a tiled background.

	Possible values are: repeat, repeat-x, repeat-y, and no-repeat
background-size	To modify the size of the background image.

Borders

Borders are used to visually separate elements. Borders are put around all four sides of an element, or just one, two, or three of the sides. Various border properties are –

Property	Description
border	A shorthand property that allows to set the style, width, and color of a border in one property. The order is important and must be: border-style border-width border-color
border-style	Specifies the line type of the border. Possible values are: *solid, dotted, dashed, double, groove, ridge, inset, and outset*.
border-width	The width of the border in a unit(usually in px). A variety of keywords (thin, medium, thick etc.) are also supported.
border-color	The color of the border in a color unit.
border-radius	The radius of a rounded corner.
border-image	The URL of an image to use as a border

Eg: border-style: dotted;
Border-width: 15px;
Border-color:red;

Note: It is possible to set the properties for one or more sides of the element box in a single property, or to set them individually (all sides separate settings) using separate properties.
Eg:
margin-top:30px;
margin-left:10px;
padding-top:10px;
border-top-width:5px;
border-buttom-width:20px;
border-top-style:dashed; etc.

For instance, we can set the side properties individually:

border-top-color: red; / sets just the top side */*
border-right-color: green; / sets just the right side */*

border-bottom-color: yellow; / sets just the bottom side */*
border-left-color: blue; / sets just the left side */*

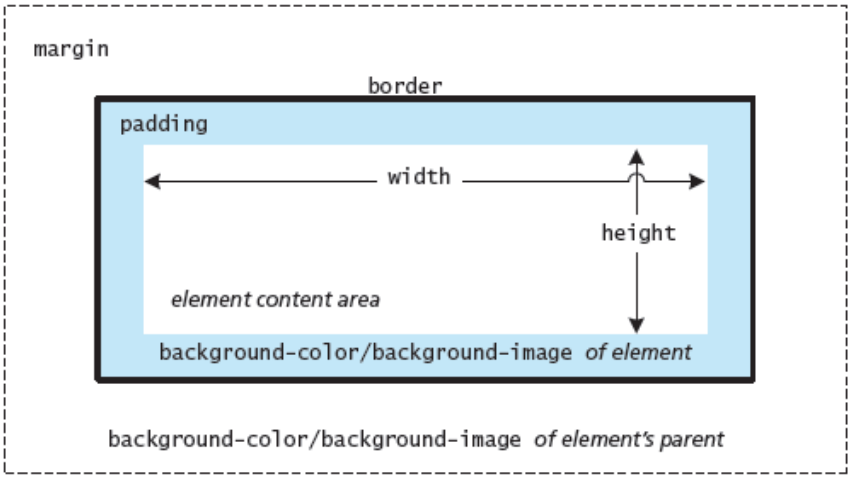
```
border-color: red green orange blue;
```

Alternately, we can set all four sides to a single value via:
border-color: red; / sets all four sides to red */*
Or
border-color: red green orange blue; (mnemonic **TRouBLe)**
When using this multiple values shortcut, they are applied in clockwise order starting at the top. Thus the order is: **top right bottom left .**

Another shortcut is to use just two values; in this case the first value sets top and bottom, while the second sets the right and left.

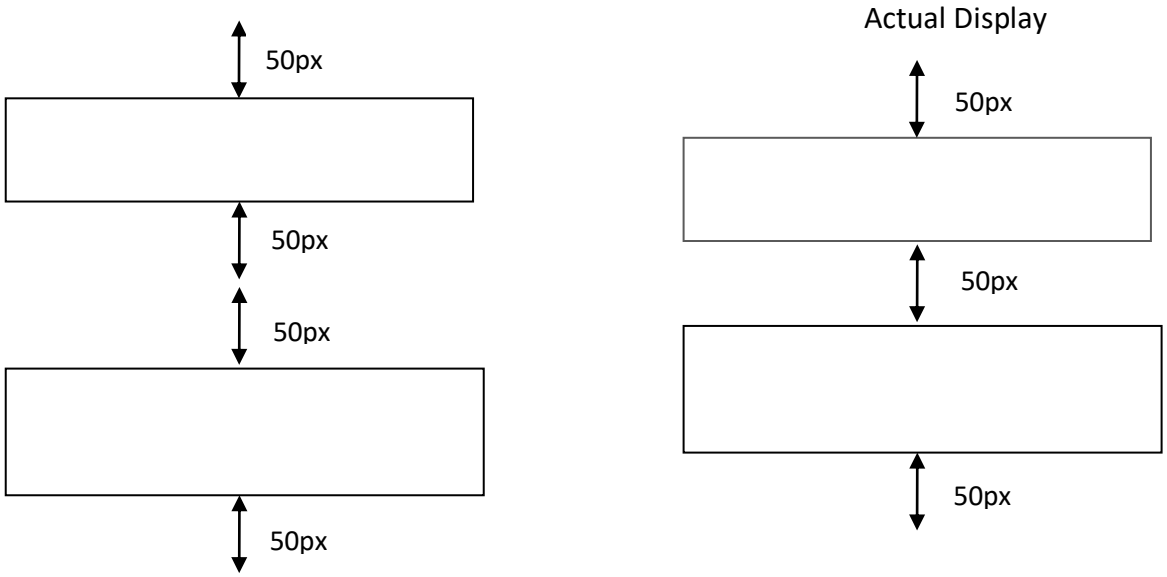
border-color: red yellow; /* *top+bottom=red, right+left=yellow* */

Margins and Padding
Margins and padding are essential properties for adding space in between two elements, which can help differentiate one element from another. Margins add spacing around an element's content, and padding adds spacing within elements. Borders divide the margin area from the padding area.



The adjoining vertical margins collapse with each other, and the margin value of the highest element is chosen between the two elements. Horizontal margins **never** collapse with each other.

If suppose, the margin of a <p> tag is set to 50px. Then the space between 2 paragraphs should be (50px +50px) 100 px. But the vertical margins are collapsed and the space of 50px is displayed.



CSS Text Styling

CSS provides two types of properties that affect text – the font properties and the paragraph properties.

Font Family

- The first of these problems involves specifying the font family. A word processor on a desktop machine can make use of any font that is installed on the computer; browsers are no different. However, just because a given font is available on the web developer's computer, it does not mean that that same font will be available for all users who view the site. For this reason, it is conventional to supply a so-called **web**

- **font stack**, that is, a series of alternate fonts to use in case the original font choice is not on the user's computer.

- One common approach is to make your font stack contain, in this order, the following: *ideal*, *alternative*, *common*, and then *generic*. Take for instance, the following font stack:

font-family { "Hoefler Text", Cambria, "Times New Roman", serif; }

Font Sizes

Another potential problem with web fonts is font sizes. In a print-based program such as a word processor, specifying a font size is unproblematic. Making some text 12 pt will mean that the font's bounding box (which in turn is roughly the size of its characters) will be 1/6 of an inch tall when printed, while making it 72 pt will make it roughly one inch tall when printed.

Property	Description
`font`	A combined shorthand property that allows you to set the family, style, size, variant, and weight in one property. While you do not have to specify each property, you must include at a minimum the font size and font family. In addtion, the order is important and must be: `style weight variant size font-family`
`font-family`	Specifies the typeface/font (or generic font family) to use. More than one can be specified.
`font-size`	The size of the font in one of the measurement units.
`font-style`	Specifies whether `italic`, `oblique` (i.e., skewed by the browser rather than a true italic), or `normal`.
`font-variant`	Specifies either `small-caps` text or none (i.e., regular text).
`font-weight`	Specifies either `normal`, `bold`, `bolder`, `lighter`, or a value between 100 and 900 in multiples of 100, where larger number represents weightier (i.e., bolder) text.

Paragraph Properties

Property	Description
letter-spacing	Adjusts the space between letters. Can be the value normal or a length unit.
line-height	Specifies the space between baselines (equivalent to leading in a desktop publishing program). The default value is normal, but can be set to any length unit. Can also be set via the shorthand font property.
list-style-image	Specifies the URL of an image to use as the marker for unordered lists.
list-style-type	Selects the marker type to use for ordered and unordered lists. Often set to none to remove markers when the list is a navigational menu or a input form.
text-align	Aligns the text horizontally in a container element in a similar way as a word processor. Possible values are left, right, center, and justify.
text-decoration	Specifies whether the text will have lines below, through, or over it. Possible values are: none, underline, overline, line-through, and blink. Hyperlinks by default have this property set to underline.
text-direction	Specifies the direction of the text, left-to-right (ltr) or right-to-left (rtl).
text-indent	Indents the first line of a paragraph by a specific amount.
text-shadow	A new CSS3 property that can be used to add a drop shadow to a text. Not yet supported in IE9.
text-transform	Changes the capitalization of text. Possible values are none, capitalize, lowercase, and uppercase.
vertical-align	Aligns the text vertically in a container element. Most common values are: top, bottom, and middle.
word-spacing	Adjusts the space between words. Can be the value normal or a length unit.