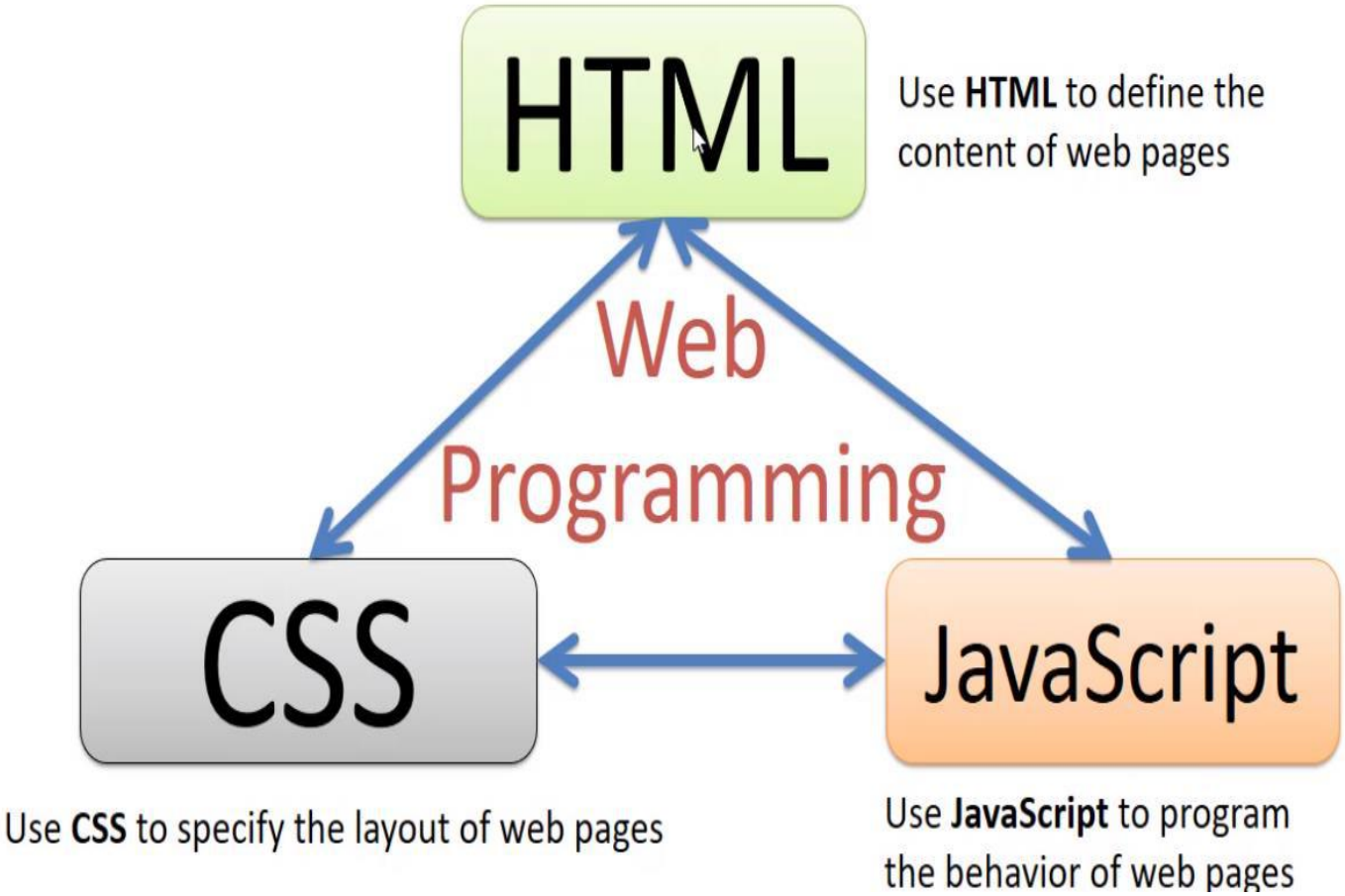
WEB PROGRAMMING
(19APC0523)

UNIT- I
Introduction

- Web Technology is essential today because Internet has become the number one source to information, and many of the traditional software applications have become Web Applications.
- Web Applications have become more powerful and can fully in most of the situations.
- Web Programming including HTML, CSS and JavaScript.
- Web Frameworks like PHP, ASP.NET, etc.
- In the Year (1960s) all are started with Internet and the World Wide Web - WWW (1991).

- The first Web Browser, Netscape, came in 1994. This was the beginning of a new era, where everything is connected on internet, the so called Internet of Things (IoT).



- What is Web?
- World Wide Web, which is also known as a Web
- It is a collection of websites or web pages stored in web servers and connected to local computers through the internet.
- These websites contain text pages, digital images, audios, videos, etc.
- Users can access the content of these sites from any part of the world over the internet using their devices such as computers, laptops, cell phones, etc.

Fundamentals of HTML

- What is HTML?
- HTML stands for Hyper Text Markup Language
- HTML is the standard markup language for creating Web pages
- HTML is used to create web pages and web applications
- HTML consists of a series of elements
- HTML elements tell the browser how to display the content
- HTML contains Tags and Attributes that are used to design the web pages
- HTML is a markup language used by the browser to manipulate text, images, and other content, in order to display it in the required format
- HTML was created by Tim Berners-Lee in 1991

- **Hyper Text:** Hyper Text means "Text within Text."
- A text has a link within it, is a hypertext. Whenever you click on a link which brings you to a new webpage, you have clicked on a hypertext.
- **Markup language:** A markup language is a computer language that is used to apply layout and formatting conventions to a text document.
- Markup language makes text more interactive and dynamic.
- **Web Page:** A web page is a document which is commonly written in HTML and translated by a web browser.
- A web page can be identified by entering an URL.
- A Web page can be of the static or dynamic type.
- With the help of HTML only, we can create static web pages.

HTML Page Structure

```
<html>
<head>
<title>Page title</title>
</head>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
</body>
</html>
```

HTML Documents

- All HTML documents must start with a document type declaration: <!DOCTYPE html>.
- The HTML document itself begins with <html> and ends with </html>.
- The visible part of the HTML document is between <body> and </body>.

The <!DOCTYPE> Declaration

- All HTML documents must start with a <!DOCTYPE> declaration.
- The declaration is not an HTML tag. It is an "information" to the browser about what document type to expect.
- The <!DOCTYPE> declaration represents the document type and helps browsers to display web pages correctly.
- It must only appear once, at the top of the page (before any HTML tags).
- The <!DOCTYPE> declaration is not case sensitive.
- The <!DOCTYPE> declaration for HTML5 is:

```
<!DOCTYPE html>
```

Browser Support

	Chrome		Firefox	Safari	Opera
Element					
<!DOCTYPE>	Yes	Yes	Yes	Yes	Yes

Older HTML Documents

- In older documents (HTML 4 or XHTML), the declaration is more complicated because the declaration must refer to a DTD (Document Type Definition)

- HTML 4.01:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

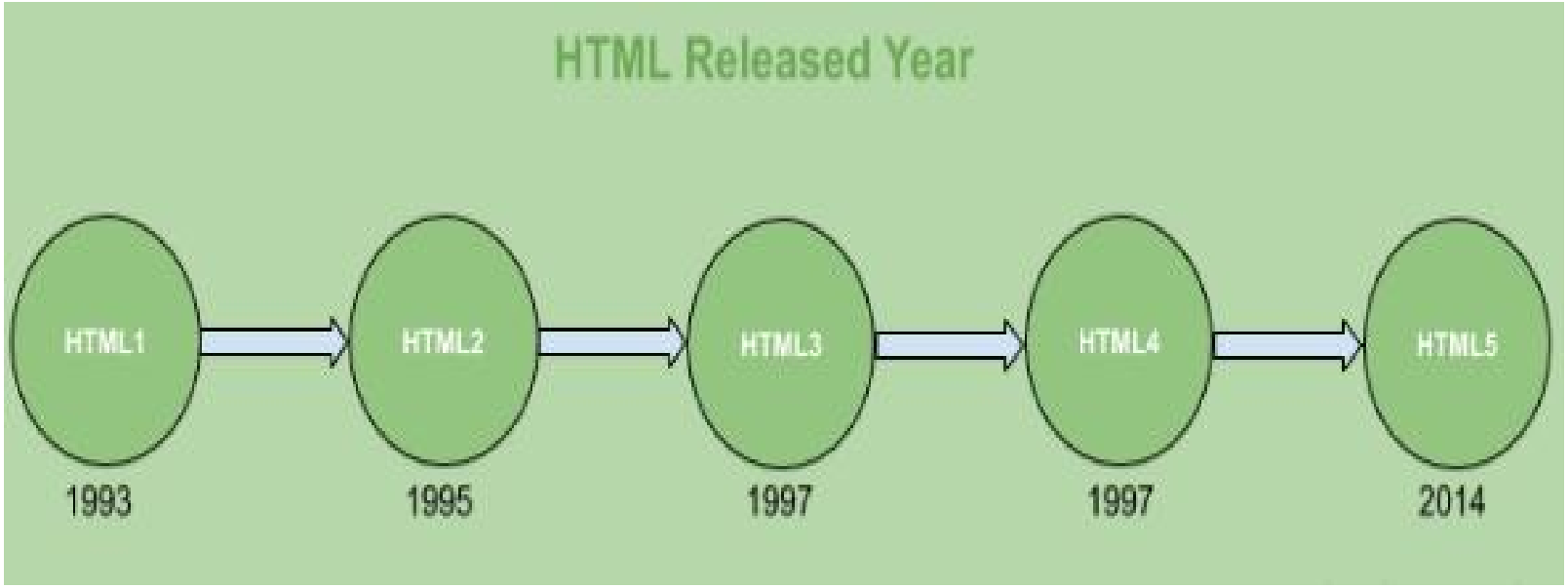
- XHTML 1.1:

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

A Simple HTML Document

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

- The <!DOCTYPE html> declaration defines that this document is an HTML5 document
- The <html> element is the root element of an HTML page
- The <head> element contains meta information about the HTML page
- The <title> element specifies a title for the HTML page (which is shown in the browser's title bar or in the page's tab)
- The <body> element defines the document's body, and it is a container for all the visible contents, such as headings, paragraphs, images, hyperlinks, tables, lists, etc.
- The <h1> element defines a large heading
- The <p> element defines a paragraph


HTML Released Year
HTML1
HTML2
HTML3
HTML4
HTML5
1993
1995
1997
1997
2014

HTML Comments

- HTML comments are not displayed in the browser, but they can help document your HTML source code.
- **Syntax:-**

```
<!-- Write your comments here -->  (Single Line)

<!-- This is
        multi-line
                comment -->  (Multi-line)
```

- **Hide Inline Content :**
- Comments can be used to hide parts in the middle of the HTML code
- **EX:-**

```
<!DOCTYPE html>
<html>
<body>

<p>This <!-- great text --> is a paragraph.</p>

</body>
</html>
```

HTML Headings

- HTML headings are defined with the <h1> to <h6> tags.
- <h1> defines the most important heading.
- <h6> defines the least important heading.

```
<!DOCTYPE html>
<html>
<body>

<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<h3>This is heading 3</h3>
<h4>This is heading 4</h4>
<h5>This is heading 5</h5>
<h6>This is heading 6</h6>

</body>
</html>
```

HTML Paragraphs

- HTML paragraphs are defined with the <p> tag

```
<!DOCTYPE html>
<html>
<body>

<p>This is a paragraph.</p>
<p>This is another paragraph.</p>

</body>
</html>
```

HTML Links

- HTML links are defined with the <a> tag.
- (Hypertext REFerence):The HTML code used to create a link to another page.
- The HREF is an attribute of the anchor tag

```
<!DOCTYPE html>
<html>
<body>

<h2>HTML Links</h2>
<p>HTML links are defined with the a tag:</p>

<a href="https://www.w3schools.com">This is a link</a>

</body>
</html>
```

HTML Images

- How to add the images in HTML
- Images can improve the design and the appearance of a web page.
- **Syntax:-**

  <img src="url" alt="some_text" width="" height="">
- **Ex:-**

<img src="pic_trulli.jpg" alt="No Image" width="500" height="333">

- **src:** It is used to specify the path to the image.
- **alt:** It is used to specify an alternate text for the image. It is useful as it informs the user about what the image means and also due to any network issue if the image cannot be displayed then this alternate text will be displayed.
- **height:** It is used to specify the height of the image.
- **width:** It is used to specify the width of the image.

What is an HTML Element?

- An HTML element is defined by a start tag, some content, and an end tag:
- <tagname> Content goes here... </tagname>
- The HTML **element** is everything from the start tag to the end tag:
- <h1>My First Heading</h1>
- My first paragraph.

HTML Styles

- The HTML style attribute is used to add styles to an element, such as color, font, size, and more.

The HTML Style Attribute

- Setting the style of an HTML element, can be done with the style attribute.
- The HTML style attribute has the following syntax:

<tagname style="property:value;">

```
<!DOCTYPE html>
<html>
<body>

<p>I am normal</p>
<p style="color:red;">I am red</p>
<p style="color:blue;">I am blue</p>
<p style="font-size:50px;">I am big</p>

</body>
</html>
```

HTML header Tag

- The <header> element represents a container for introductory content or a set of navigational links
- A <header> element contains:
  - one or more heading elements (<h1> - <h6>)
  - logo or icon
  - authorship information

```
<!DOCTYPE html>
<html>
<body>

<header>
  <h1>Main page heading here</h1>
  <p>Posted by John Doe</p>
</header>

</body>
</html>
```

HTML footer Tag

- A footer section in a document
- A <footer> element contains:
  - authorship information
  - copyright information
  - contact information
  - sitemap
  - back to top links
  - related documents

HTML footer Tag

- <!DOCTYPE html>
- <html>
- <body>

- <h1>The footer element</h1>

- <footer>
- Author: Hege Refsnes
- hege@example.com
- </footer>

- </body>
- </html>

Features of HTML

- It is easy to learn and easy to use.
- It is platform-independent.
- Images, videos, and audio can be added to a web page.
- Hypertext can be added to the text.
- It is a markup language.

Advantages

- HTML is used to build websites.
- It is supported by all browsers.
- It can be integrated with other languages like CSS, JavaScript, etc.

Disadvantages

- A large amount of code has to be written to create a simple web page

Working with text

- **HTML Formatting** is a process of formatting text for better look and feel.
- HTML provides us ability to format text without using CSS.
- There are many formatting tags in HTML.
- These tags are used to make text bold, italicized, or underlined.
- There are almost 14 options available that how text appears in HTML
- In HTML the formatting tags are divided into two categories:
  - Physical tag: These tags are used to provide the visual appearance to the text.
  - Logical tag: These tags are used to add some logical or semantic value to the text.

Bold Text

- HTML and formatting elements
- The HTML element is a physical tag which display text in bold font, without any logical importance.
- If you write anything within ..... element, is shown in bold letters.

```
<!DOCTYPE>
<html>
<body>
<p> <b>Write Your First Paragraph in bold text.</b></p>
</body>
</html>
```

- The HTML tag is a logical tag, which displays the content in bold font and informs the browser about its logical importance.
- If you write anything between ???????. , is shown important text.
- **Ex:-**
- This is an important content, and this is normal content

```
<!DOCTYPE html>
<html>
<head>
        <title>formatting elements</title>
</head>
<body>
<h1>Explanation of formatting element</h1>
<p><strong>This is an important content</strong>, and this is normal content</p>
</body>
</html>
```

Italic Text

- **HTML and formatting elements**
- The HTML element is physical element, which display the enclosed content in italic font.
- If you write anything within ..... element, is shown in italic letters.

```
<!DOCTYPE>
<html>
<body>
<p> <i>Write Your First Paragraph in italic text.</i></p>
</body>
</html>
```

•The HTML tag is a logical element,
• which will display the enclosed content in italic font, with added semantics importance.
•The tag is used to define emphasized text. The content inside is typically displayed in italic.
• **EX:-**
• ** **This is an important content****, which displayed in italic font.** **

```
<!DOCTYPE html>
<html>
<head>
   <title>formatting elements</title>
</head>
<body>
<h1>Explanation of italic formatting element</h1>
<p><em>This is an important content</em>, which displayed in italic font.</p>
</body>
</html>
```

HTML Marked formatting

- If you want to mark or highlight a text, you should write the content within <mark>.....</mark>.
- **EX:-** **<h2>** I want to put a **<mark>** Mark**</mark>** on your face**</h2>**

- <!DOCTYPE>
- <html>
- <body>
- <h2> I want to put a <mark> Mark</mark> on your face</h2>
- </body>
- </html>

Underlined Text

- If you write anything within <u>.....</u> element, is shown in underlined text.
- Ex:- ** <u>**Write Your First Paragraph in underlined text.**</u> **
- <!DOCTYPE>
- <html>
- <body>
- <u>Write Your First Paragraph in underlined text.</u>
- </body>
- </html>

Strike Text

- Anything written within <strike>.....</strike> element is displayed with strikethrough.
- It is a thin line which cross the statement.
- **Ex:-** <strike>Write Your First Paragraph with strikethrough </strike>.
- <!DOCTYPE>
- <html>
- <body>
- <strike>Write Your First Paragraph with strikethrough</strike>.
- </body>
- </html>

Monospaced Font

- If you want that each letter has the same width then you should write the content within <tt>.....</tt> element.
- <tt> means-The **Teletype Text element**
- Note:
- We know that most of the fonts are known as variable-width fonts because different letters have different width. (for example: 'w' is wider than 'i').
- Monospaced Font provides similar space among every letter.
- <!DOCTYPE>
- <html>
- <body>
- Hello <tt>Write Your First Paragraph in monospaced font.</tt>
- </body>
- </html>

Superscript Text

- If you put the content within ^{.....} element, is shown in superscript; means it is displayed half a character's height above the other characters.
- <!DOCTYPE>
- <html>
- <body>
- Hello ^{Write Your First Paragraph in superscript.}
- </body>
- </html>

Subscript Text

- If you put the content within _{.....} element, is shown in subscript ; means it is displayed half a character's height below the other characters.
- <!DOCTYPE>
- <html>
- <body>
- Hello _{Write Your First Paragraph in subscript.}
- </body>
- </html>

Deleted Text

- Anything that puts within ~~.....~~ is displayed as deleted text.
- <!DOCTYPE>
- <html>
- <body>
- Hello ~~Delete your first paragraph.~~
- </body>
- </html>

Inserted Text

- Anything that puts within <ins>.....</ins> is displayed as inserted text.
- <!DOCTYPE>
- <html>
- <body>
- ~~Delete your first paragraph.~~<ins>Write another paragraph.</ins>
- </body>
- </html>

Larger Text

- If you want to put your font size larger than the rest of the text then put the content within <big>.....</big>.
- It increase one font size larger than the previous one.
- <!DOCTYPE>
- <html>
- <body>
- Hello <big>Write the paragraph in larger font.</big>
- </body>
- </html>

Smaller Text

- If you want to put your font size smaller than the rest of the text then put the content within <small>.....</small>tag.
- It reduces one font size than the previous one.
- <!DOCTYPE>
- <html>
- <body>
- Hello <small>Write the paragraph in smaller font.</small>
- </body>
- </html>

HTML Styles - CSS

Styling HTML with CSS

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

CSS describes **how HTML elements are to be displayed on screen, paper, or in other media**.

CSS **saves a lot of work**. It can control the layout of multiple web pages all at once.

CSS can be added to HTML elements in 3 ways:

- **Inline** - by using the style attribute in HTML elements
- **Internal** - by using a <style> element in the <head> section
- **External** - by using an external CSS file

The most common way to add CSS, is to keep the styles in separate CSS files. However, here we will use inline and internal styling, because this is easier to demonstrate, and easier for you to try it yourself.

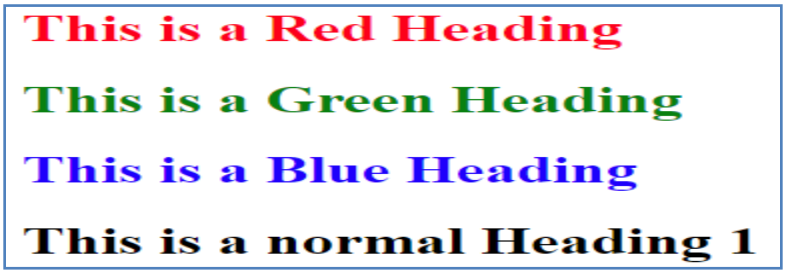
Inline CSS

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.

This example sets the text color of the <h1> element to blue:

```
<!DOCTYPE html>
<html>
<body>
<h1 style="color:red;">This is a Red Heading</h1>
<h1 style="color:green;">This is a Green Heading</h1>
<h1 style="color:blue;">This is a Blue Heading</h1>
<h1>This is a normal Heading 1</h1>
</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:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {background-color: black;}
h1   {color: red;}
h2       {color: green;}
h3   {color: blue;}
h4   {color: yellow;}
p    {color: magenta;}
</style>
</head>
<body>
<h1>This is a Red Heading 1</h1>
```

```
<h2>This is a Green Heading 2</h2>
<h3>This is a Blue Heading 3</h3>
<h4>This is a Yellow Heading 4</h4>
<p>This is a paragraph in Magenta color.</p>
</body>
</html>
```

External CSS

An external style sheet is used to define the style for many HTML pages.

With an external style sheet, you can change the look of an entire web site, by changing one file!

To use an external style sheet, add a link to it in the <head> section of the HTML page:

Externalcss.html

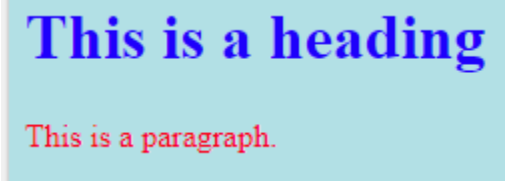
```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

An external style sheet can be written in any text editor. The file must not contain any HTML code, and must be saved with a .css extension.

Here is how the "styles.css" looks:

styles.css

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

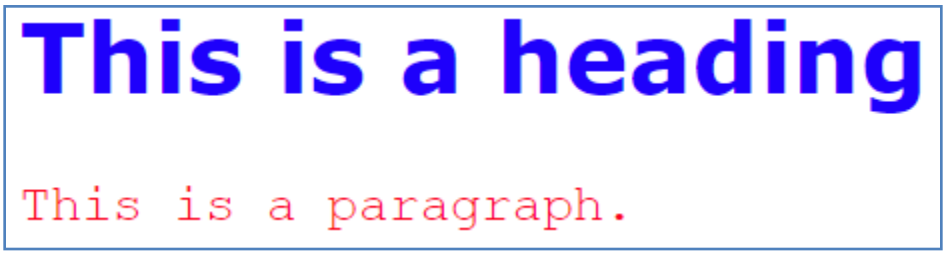


- Use the HTML style attribute for inline styling
- Use the HTML <style> element to define internal CSS
- Use the HTML <link> element to refer to an external CSS file
- Use the HTML <head> element to store <style> and <link> elements
- Use the CSS color property for text colors
- Use the CSS font-family property for text fonts
- Use the CSS font-size property for text sizes
- Use the CSS border property for borders
- Use the CSS padding property for space inside the border
- Use the CSS margin property for space outside the border

CSS Fonts

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.

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
  color: blue;
  font-family: verdana;
  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>
```



External References

External style sheets can be referenced with a full URL or with a path relative to the current web page.

This example uses a full URL to link to a style sheet:

```
<link rel="stylesheet" href="https://www.w3schools.com/html/styles.css">
```

This example links to a style sheet located in the html folder on the current web site:

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

This example links to a style sheet located in the same folder as the current page:

```
<link rel="stylesheet" href="styles.css">
```

HTML

What is HTML?

HTML is the standard markup language for creating Web pages.

- HTML stands for Hyper Text Markup Language
- HTML describes the structure of Web pages using markup
- HTML elements are the building blocks of HTML pages
- HTML elements are represented by tags
- HTML tags label pieces of content such as "heading", "paragraph", "table", and so on
- Browsers do not display the HTML tags, but use them to render the content of the page

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>
<h1>My First Heading</h1>
<p>My first paragraph.</p>
</body>
</html>
```

Output:

Explanation

- The <!DOCTYPE html> declaration defines this document to be HTML5
- The <html> element is the root element of an HTML page
- The <head> element contains meta information about the document
- The <title> element specifies a title for the document
- The <body> element contains the visible page content
- The <h1> element defines a large heading
- The element defines a paragraph

HTML Tags

HTML tags are element names surrounded by angle brackets:
<tagname>content goes here...</tagname>

- HTML tags normally come **in pairs** like and
- The first tag in a pair is the **start tag,** the second tag is the **end tag**
- The end tag is written like the start tag, but with a **forward slash** inserted before the tag name

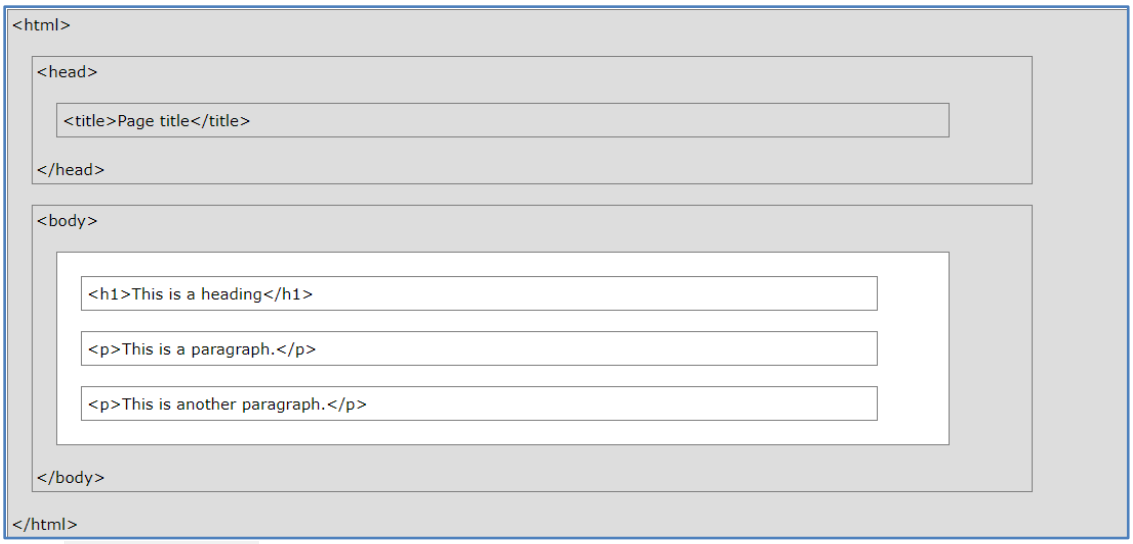
Web Browsers

The purpose of a web browser (Chrome, IE, Firefox, Safari) is to read HTML documents and display them.

The browser does not display the HTML tags, but uses them to determine how to display the document

HTML Page Structure

Below is a visualization of an HTML page structure:



The <!DOCTYPE> declaration represents the document type, and helps browsers to display web pages correctly.
It must only appear once, at the top of the page (before any HTML tags).
The <!DOCTYPE> declaration is not case sensitive.

HTML Versions

Since the early days of the web, there have been many versions of HTML:

Version	Year
HTML	1991
HTML 2.0	1995
HTML 3.2	1997
HTML 4.01	1999
XHTML	2000
HTML5	2014

HTML History

Since the early days of the World Wide Web, there have been many versions of HTML:

Year	Version
1989	Tim Berners-Lee invented www
1991	Tim Berners-Lee invented HTML
1993	Dave Raggett drafted HTML+
1995	HTML Working Group defined HTML 2.0
1997	W3C Recommendation: HTML 3.2
1999	W3C Recommendation: HTML 4.01
2000	W3C Recommendation: XHTML 1.0
2008	WHATWG HTML5 First Public Draft
2012	WHATWG HTML5 Living Standard
2014	W3C Recommendation: HTML5

2016	W3C Candidate Recommendation: HTML 5.1
2017	W3C Recommendation: HTML5.1 2nd Edition
2017	W3C Recommendation: HTML5.2

Write HTML Using Notepad or TextEdit

Web pages can be created and modified by using professional HTML editors.
However, for learning HTML we recommend a simple text editor like Notepad (PC) or TextEdit (Mac).

Steps to create a web page

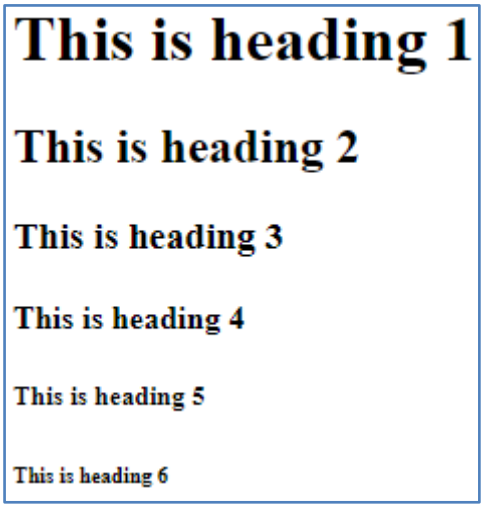
1. Open Notepad
2. Write or copy some HTML into Notepad.
3. Save the file on your computer. Select **File > Save as** in the Notepad menu.
4. Name the file **"index.htm"** and set the encoding to **UTF-8** (which is the preferred encoding for HTML files).
5. Open the saved HTML file in your favorite browser (double click on the file, or right-click - and choose "Open with")

HTML Headings

HTML headings are defined with the <h1> to <h6> tags.
<h1> defines the most important heading. <h6> defines the least important heading:

```
<!DOCTYPE html>
<html>
<body>
<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<h3>This is heading 3</h3>
<h4>This is heading 4</h4>
<h5>This is heading 5</h5>
<h6>This is heading 6</h6>
</body>
</html>
```

Output



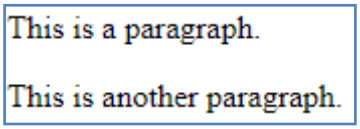
HTML Paragraphs

HTML paragraphs are defined with the <p> tag:

```
<!DOCTYPE html>
<html>
<body>
```

```
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
</body>
</html>
```

Output



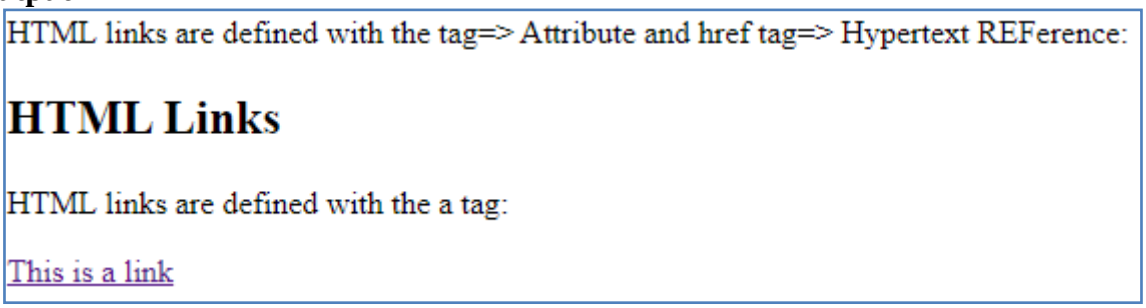
This is a paragraph.
This is another paragraph.

HTML Links

HTML links are defined with the <a> tag=> Attribute and href tag=> Hypertext REFerence:

```
<!DOCTYPE html>
<html>
<body>
<h2>HTML Links</h2>
<p>HTML links are defined with the a tag:</p>
<a href="https://www.w3schools.com">This is a link</a>
</body>
</html>
```

Output



Explanation

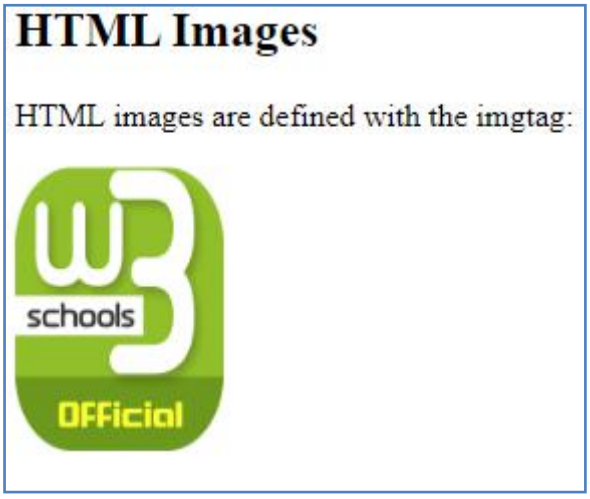
The link's destination is specified in the href attribute.
Attributes are used to provide additional information about HTML elements.
You will learn more about attributes in a later chapter.

HTML Images

HTML images are defined with the <img> tag.
The source file (src), alternative text (alt), width, and height are provided as attributes:

```
<!DOCTYPE html>
<html>
<body>
<h2>HTML Images</h2>
<p>HTML images are defined with the imgtag:</p>
<img src="w3schoolslogo.png" alt="W3Schools.com" width="104" height="142">
</body>
</html>
```

Output



HTML Attributes

Attributes provide additional information about HTML elements.

- All HTML elements can have **attributes**
- Attributes provide **additional information** about an element
- Attributes are always specified in **the start tag**
- Attributes usually come in name/value pairs like: **name="value"**

The src Attribute

HTML images are defined with the <img> tag.

The filename of the image source is specified in the src attribute:

The width and height Attributes

Images in HTML have a set of **size** attributes, which specifies the width and height of the image

```
<!DOCTYPE html>
<html>
<body>
<h2>HTML Images</h2>
<p>HTML images are defined with the imgtag:</p>
<img src="" alt="W3Schools.com" width="104" height="142">
</body>
</html>
```

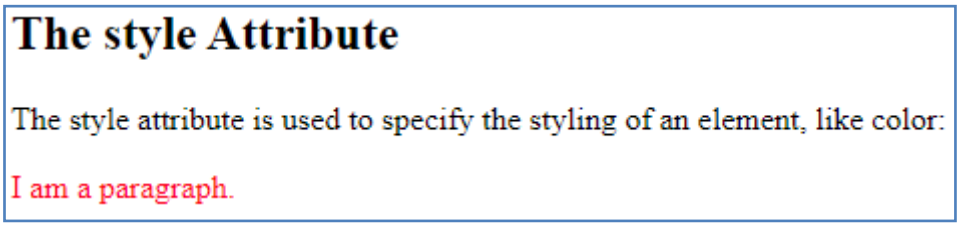
Output

The style Attribute

The style attribute is used to specify the styling of an element, like color, font, size etc.

```
<!DOCTYPE html>
<html>
<body>
<h2>The style Attribute</h2>
<p>The style attribute is used to specify the styling of an element, like color:</p>
<p style="color:red">I am a paragraph.</p>
</body>
</html>
```

Output

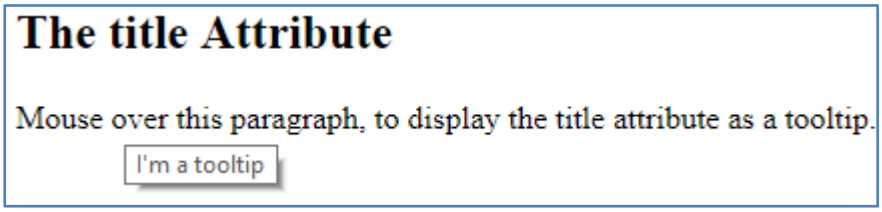


The title Attribute

Here, a title attribute is added to the <p> element. The value of the title attribute will be displayed as a tooltip when you mouse over the paragraph:

```
<!DOCTYPE html>
<html>
<body>
<h2 title="I'm a header in size h2">The title Attribute</h2>
<p title="I'm a tooltip">
Mouse over this paragraph, to display the title attribute as a tooltip.
</p>
</body>
</html>
```

Output



HTML Horizontal Rules

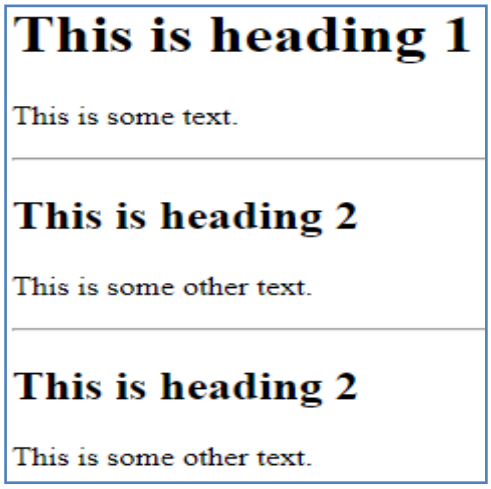
The <hr> tag defines a thematic break in an HTML page, and is most often displayed as a horizontal rule.

The <hr> element is used to separate content (or define a change) in an HTML page:

```
<!DOCTYPE html>
<html>
<body>
<h1>This is heading 1</h1>
<p>This is some text.</p>
<hr>
<h2>This is heading 2</h2>
<p>This is some other text.</p>
<hr>
<h2>This is heading 2</h2>
<p>This is some other text.</p>
</body>
```

</html>

Output



The HTML <head> Element

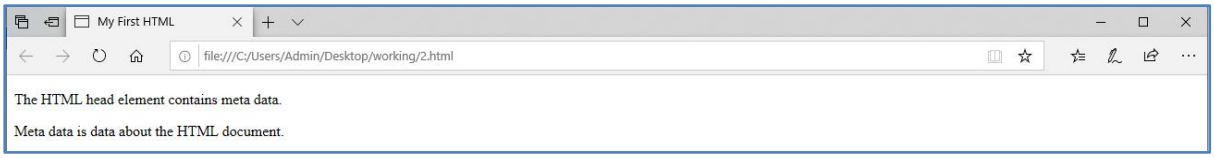
The HTML <head> element has nothing to do with HTML headings.

The <head> element is a container for metadata. HTML metadata is data about the HTML document. Metadata is not displayed.

The <head> element is placed between the <html> tag and the <body> tag:

```
<!DOCTYPE html>
<html>
<head>
<title>My First HTML</title>
<meta charset="UTF-8">
</head>
<body>
<p>The HTML head element contains meta data.</p>
<p>Meta data is data about the HTML document.</p>
</body>
</html>
```

Output



How to View HTML Source?

Have you ever seen a Web page and wondered "Hey! How did they do that?"

View HTML Source Code:

Right-click in an HTML page and select "View Page Source" (in Chrome) or "View Source" (in IE), or similar in other browsers. This will open a window containing the HTML source code of the page.

Inspect an HTML Element:

Right-click on an element (or a blank area), and choose "Inspect" or "Inspect Element" to see what elements are made up of (you will see both the HTML and the CSS). You can also edit the HTML or CSS on-the-fly in the Elements or Styles panel that opens.

HTML Background Color

The background-color property defines the background color for an HTML element.

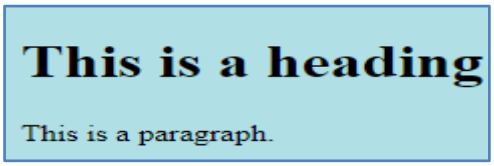
This example sets the background color for a page to powderblue:

(<body bgcolor="red;">)

```
<!DOCTYPE html>
<html>
```

```
<body style="background-color:powderblue;">
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

Output



HTML Text Color

The color property defines the text color for an HTML element:

```
<!DOCTYPE html>
<html>
<body>
<h1 style="color:blue;">This is a heading</h1>
<p style="color:red;">This is a paragraph.</p>
</body>
</html>
```

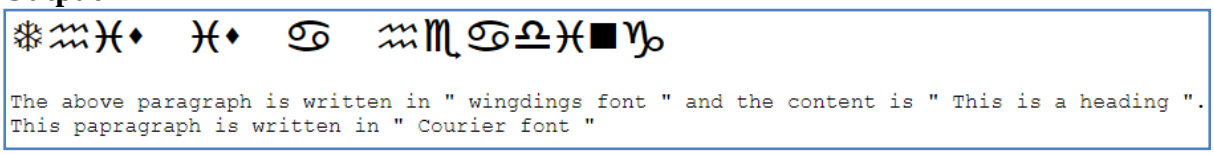
Output

HTML Fonts

The font-family property defines the font to be used for an HTML element:

```
<!DOCTYPE html>
<html>
<body>
<h1 style="font-family:wingdings;">This is a heading</h1>
<p style="font-family:courier;">The above paragraph is written in &amp;quot; wingdings font
&amp;quot; and the content is &amp;quot; This is a heading &amp;quot;.<br>
This papragraph is written in &amp;quot; Courier font &amp;quot;</p>
</body>
</html>
```

Output



HTML Text Alignment

The text-align property defines the horizontal text alignment for an HTML element:

```
<!DOCTYPE html>
<html>
<body>
<h1 style="text-align:Left;">left Heading</h1>
<h1 style="text-align:center;">Center Heading</h1>
<h1 style="text-align:Right;">Right Heading</h1>
</body>
```

```
</html>
```

Output

HTML Formatting Elements

In the previous chapter, you learned about the HTML **style attribute**.
HTML also defines special **elements** for defining text with a special **meaning**.
HTML uses elements like <b> and <i> for formatting output, like **bold** or *italic* text.
Formatting elements were designed to display special types of text:

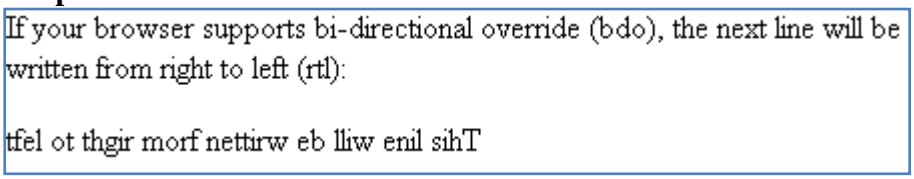
- <b> - Bold text
- - Important text
- <i> - Italic text
- - Emphasized text
- <mark> - Marked text
- <small> - Small text
- <del> - Deleted text
- <ins> - Inserted text
- <sub> - Subscript text
- <sup> - Superscript text

HTML <bdo> for Bi-Directional Override

The HTML <bdo> element defines bi-directional override.
The <bdo> element is used to override the current text direction:

```
<!DOCTYPE html>
<html>
<body>
<p>If your browser supports bi-directional override (bdo), the next line will be written from right to left (rtl):</p>
<bdo dir="rtl">This line will be written from right to left</bdo>
</body>
</html>
```

Output



HTML Comment Tags

You can add comments to your HTML source by using the following syntax:

```
<!-- Write your comments here -->
```

Notice that there is an exclamation point (!) in the opening tag, but not in the closing tag.
Note: Comments are not displayed by the browser, but they can help document your HTML source code.
With comments you can place notifications and reminders in your HTML:

```
<!DOCTYPE html>
<html>
<body>
<!-- This is a comment -->
<p>This is a testing for comment.</p>
<!-- Comments are not displayed in the browser -->
```

</body>
</html>

Output

This is a testing for comment.

HTML Colors

HTML colors are specified using predefined color names, or RGB, HEX, HSL, RGBA, HSLA values.

Named Colors Sorted by HEX Value

Colorname	HEX	RGB
Black	0	0,0,0
Blue	0000FF	0,0,255
Brown	A52A2A	165,42,42
DeepPink	FF1493	2,55,20,147
Gold	FFD700	255,215,0
Green	8000	0,128,0
Magenta	FF00FF	255,0,255
Maroon	800000	128,0,0
Orange	FFA500	255,165,0
Red	FF0000	255,0,0
White	FFFFFF	25,52,55,255

Background Color or Background

You can set the background color for HTML elements:

(<body bgcolor="red;">)

```
<!DOCTYPE html>
<html>
<body>
<h1 style="background-color:DodgerBlue;">Hello World</h1>
<p style="background-color:Tomato;">
• HTML stands for Hyper Text Markup Language<br>
• HTML describes the structure of Web pages using markup<br>
• HTML elements are the building blocks of HTML pages<br>
• HTML elements are represented by tags<br>
• HTML tags label pieces of content such as "heading", "paragraph", "table", and so on<br>
• Browsers do not display the HTML tags, but use them to render the content of the page
</p>
</body>
</html>
```

Output

Hello World

• HTML stands for Hyper Text Markup Language
• HTML describes the structure of Web pages using markup
• HTML elements are the building blocks of HTML pages
• HTML elements are represented by tags
• HTML tags label pieces of content such as "heading", "paragraph", "table", and so on
• Browsers do not display the HTML tags, but use them to render the content of the page

Text Color

You can set the color of text:

```
<!DOCTYPE html>
```

```
<html>
<body>
<h3 style="color:Tomato;">Hello World</h3>
<p style="color:DodgerBlue;">Hello World</p>
<p style="color:MediumSeaGreen;">Hello World</p>
</body>
</html>
```

Output

Border Color

You can set the color of borders:

```
<!DOCTYPE html>
<html>
<body>
<h1 style="border: 2px solid Tomato;">Hello World</h1>
<h1 style="border: 2px solid DodgerBlue;">Hello World</h1>
<h1 style="border: 2px solid Violet;">Hello World</h1>
</body>
</html>
```

Output

Color Values

In HTML, colors can also be specified using RGB values, HEX values, HSL values, RGBA values, and HSLA values:

Same as color name "Tomato":

RGB Value

In HTML, a color can be specified as an RGB value, using this formula:

rgb(*red, green, blue*)

Each parameter (red, green, and blue) defines the intensity of the color between 0 and 255.

For example, rgb(255, 0, 0) is displayed as red, because red is set to its highest value (255) and the others are set to 0.

To display the color black, all color parameters must be set to 0, like this: rgb(0, 0, 0).

To display the color white, all color parameters must be set to 255, like this: rgb(255, 255, 255).

HEX Value

In HTML, a color can be specified using a hexadecimal value in the form:

#rrggbb

Where rr (red), gg (green) and bb (blue) are hexadecimal values between 00 and ff (same as decimal 0-255).

For example, #ff0000 is displayed as red, because red is set to its highest value (ff) and the others are set to the lowest value (00).

HSL Value

In HTML, a color can be specified using hue, saturation, and lightness (HSL) in the form:

hsl(*hue, saturation, lightness*)
Hue is a degree on the color wheel from 0 to 360. 0 is red, 120 is green, and 240 is blue.
Saturation is a percentage value, 0% means a shade of gray, and 100% is the full color.
Lightness is also a percentage, 0% is black, 50% is neither light or dark, 100% is white
Saturation
Saturation can be described as the intensity of a color.
100% is pure color, no shades of gray
50% is 50% gray, but you can still see the color.
0% is completely gray, you can no longer see the color.
Lightness
The lightness of a color can be described as how much light you want to give the color, where 0% means no light (black), 50% means 50% light (neither dark nor light) 100% means full lightness (white).

RGBA Value

RGBA color values are an extension of RGB color values with an alpha channel - which specifies the opacity for a color.
An RGBA color value is specified with:
rgba(*red, green, blue, alpha*)
The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (not transparent at all):
HSLA Value
HSLA color values are an extension of HSL color values with an alpha channel - which specifies the opacity for a color.
An HSLA color value is specified with:
hsla(*hue, saturation, lightness, alpha*)
The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (not transparent at all):

HTML Lists

Unordered HTML List

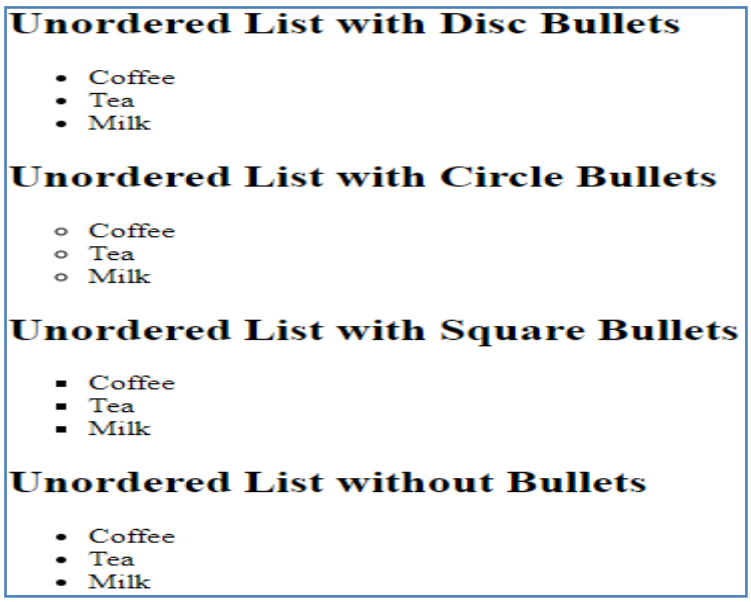
An unordered list starts with the <ul> tag. Each list item starts with the <li> tag.
The list items will be marked with bullets (small black circles) by default:
Unordered HTML List - Choose List Item Marker
The CSS list-style-type property is used to define the style of the list item marker:

Value	Description
disc	Sets the list item marker to a bullet (default)
circle	Sets the list item marker to a circle
square	Sets the list item marker to a square
none	The list items will not be marked

```
<!DOCTYPE html>
<html>
<body>
<h2>Unordered List with Disc Bullets</h2>
<ul style="list-style-type:disc">
<li>Coffee</li>
<li>Tea</li>
<li>Milk</li>
</ul>

<h2>Unordered List with Circle Bullets</h2>
<ul style="list-style-type:circle">
<li>Coffee</li>
```

```
<li>Tea</li>
<li>Milk</li>
</ul>

<h2>Unordered List with Square Bullets</h2>
<ul style="list-style-type:square">
<li>Coffee</li>
<li>Tea</li>
<li>Milk</li>
</ul>

<h2>Unordered List without Bullets</h2>
<ul>
<li>Coffee</li>
<li>Tea</li>
<li>Milk</li>
</ul>
</body>
</html>
```

Output

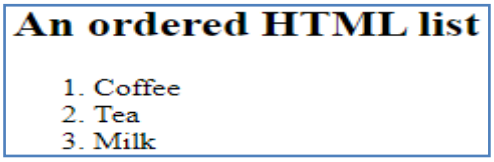


Ordered HTML List

An ordered list starts with the <ol> tag. Each list item starts with the <li> tag.
The list items will be marked with numbers by default:

```
<!DOCTYPE html>
<html>
<body>
<h2>An ordered HTML list</h2>
<ol>
<li>Coffee</li>
<li>Tea</li>
<li>Milk</li>
</ol>
</body>
</html>
```

Output

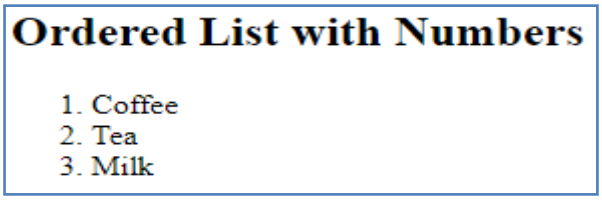


Ordered HTML List - The Type Attribute

The type attribute of the <ol> tag, defines the type of the list item marker:

```
<!DOCTYPE html>
<html>
<body>
<h2>Ordered List with Numbers</h2>
<ol type="1">
<li>Coffee</li>
<li>Tea</li>
<li>Milk</li>
</ol>
</body>
</html>
```

Output



Type	Description
type="1"	The list items will be numbered with numbers (default)
type="A"	The list items will be numbered with uppercase letters
type="a"	The list items will be numbered with lowercase letters
type="I"	The list items will be numbered with uppercase roman numbers
type="i"	The list items will be numbered with lowercase roman numbers
type="3"	The list items will be numbered with number 3
type="A" start="4"	The list items will be numbered with uppercase letters from D
type="a" start="4"	The list items will be numbered with lowercase letters from d
type="I" start="4"	The list items will be numbered with uppercase roman numbers from IV
type="i" start="4"	The list items will be numbered with lowercase roman numbers from iv

Ordered List with Numbers

1. Coffee
2. Tea
3. Milk

Ordered List with Letters

A. Coffee
B. Tea
C. Milk

Ordered List with Lowercase Letters

a. Coffee
b. Tea
c. Milk

Ordered List with Roman Numbers

I. Coffee
II. Tea
III. Milk

Ordered List with Numbers starting from 4

4. Coffee
5. Tea
6. Milk

Ordered List with Uppercase letters starting from 4

D. Coffee
E. Tea
F. Milk

Ordered List with Lowercase letters starting from 4

d. Coffee
e. Tea
f. Milk

Ordered List with Roman letters starting from 4

IV. Coffee
V. Tea
VI. Milk

Ordered List with Roman letters starting from 4

iv. Coffee
v. Tea
vi. Milk

HTML Description Lists

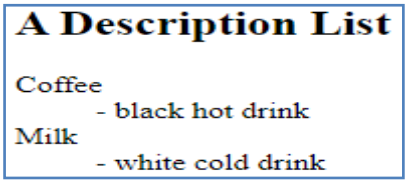
HTML also supports description lists.

A description list is a list of terms, with a description of each term.

The <dl> tag defines the description list, the <dt> tag defines the term (name), and the <dd> tag describes each term:

```
<!DOCTYPE html>
<html>
<body>
<h2>A Description List</h2>
<dl>
<dt>Coffee</dt>
<dd>- black hot drink</dd>
<dt>Milk</dt>
<dd>- white cold drink</dd>
</dl>
</body>
</html>
```

Output

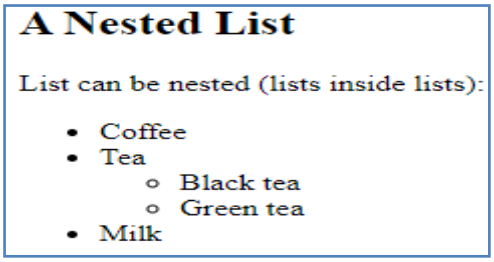


Nested HTML Lists

List can be nested (lists inside lists):

```
<!DOCTYPE html>
<html>
<body>
<h2>A Nested List</h2>
<p>List can be nested (lists inside lists):</p>
<ul>
<li>Coffee</li>
<li>Tea
        <ul>
        <li>Black tea</li>
        <li>Green tea</li>
        </ul>
</li>
<li>Milk</li>
</ul>
</body>
</html>
```

Output



Control List Counting

By default, an ordered list will start counting from 1. If you want to start counting from a specified number, you can use the start attribute:

```
<!DOCTYPE html>
<html>
<body>
<h2>The start attribute</h2>
<p>By default, an ordered list will start counting from 1. Use the start attribute to start counting from a specified number:</p>
<ol start="50">
<li>Coffee</li>
<li>Tea</li>
<li>Milk</li>
</ol>
<ol type="I" start="50">
<li>Coffee</li>
<li>Tea</li>
<li>Milk</li>
</ol>
</body>
</html>
```

Output

The start attribute

By default, an ordered list will start counting from 1. Use the start attribute to start counting from a specified number:

50. Coffee
51. Tea
52. Milk

L. Coffee
LI. Tea
LII. Milk

Defining an HTML Table

An HTML table is defined with the <table> tag.
Each table row is defined with the <tr> tag. A table header is defined with the <th> tag. By default, table headings are bold and centered. A table data/cell is defined with the <td> tag.

```
<!DOCTYPE html>
<html>
<table border = "1">
<tr>
<th>Name</th>
<th>Oops</th>
<th>dbms</th>
</tr>
<tr>
<td>Jill</td>
<td>80</td>
<td>50</td>
</tr>
<tr>
<td>Eve</td>
<td>50</td>
<td>94</td>
</tr>
<tr>
<td>John</td>
<td>90</td>
<td>80</td>
</tr>
</table>
</html>
```

Output

Name	Oops	dbms
Jill	80	50
Eve	50	94
John	90	80

Name	Oops	dbms
Jill	80	50
Eve	50	94
John	90	80

HTML Table - Cells that Span Many Columns& Rows, Caption

To make a cell span more than one column, use the colspan attribute:

To make a cell span more than one row, use the rowspan attribute:
To add a caption to a table, use the <caption> tag:

```
<!DOCTYPE html>
<html>

<table border = "1">
<caption>Marks Statement</caption>
<tr>
<!--Rowspan merges n number of rows
                colspan merges n number of columns
        -->
<th rowspan=2>Name</th><th colspan=3>Marks</th>
</tr>
        <tr>
        <th>Oops</th><th>Dbms</th><th>Java</th>
        </tr>
<tr>
<td>Jill</td><td>80</td><td>50</td><td>50</td>
</tr>
<tr>
<td>Eve</td><td>50</td><td>94</td><td>50</td>
</tr>
<tr>
<td>John</td><td>90</td><td>80</td><td>50</td>
</tr>
</table>
</html>
```

Output

Marks Statement

Name	Marks		
	Oops	Dbms	Java
Jill	80	50	50
Eve	50	94	50
John	90	80	50

Use the HTML <table> element to define a table
Use the HTML <tr> element to define a table row
Use the HTML <td> element to define a table data
Use the HTML <th> element to define a table heading
Use the HTML <caption> element to define a table caption
Use the CSS border property to define a border
Use the CSS border-collapse property to collapse cell borders
Use the CSS padding property to add padding to cells
Use the CSS text-align property to align cell text
Use the CSS border-spacing property to set the spacing between cells
Use the colspan attribute to make a cell span many columns

Use the rowspan attribute to make a cell span many rows
Use the id attribute to uniquely define one table

Attributes of table tag

Define Table: <TABLE></TABLE>
Columns to Span: <TH COLSPAN=?>
Rows to Span: <TH ROWSPAN=?>
Desired Width: <TH WIDTH=?> – *(in pixels)*
Width Percent: <TH WIDTH="%"> – *(percentage of table)*
Cell Color: <TH BGCOLOR="#$$$$$$">
Table Caption: <CAPTION></CAPTION>
Alignment: <CAPTION ALIGN=TOP|BOTTOM> – *(above/below table)*
Table Border: <TABLE BORDER=?></TABLE>
Cell Spacing: <TABLE CELLSPACING=?>
Cell Padding: <TABLE CELLPADDING=?>
Desired Width: <TABLE WIDTH=?> – *(in pixels)*
Width Percent: <TABLE WIDTH="%"> – *(percentage of page)*
Table Row: <TR></TR>
Alignment: <TR ALIGN=LEFT|RIGHT| CENTER|MIDDLE|BOTTOM
VALIGN=TOP|BOTTOM|MIDDLE>
Table Cell: <TD></TD> – *(must appear within table rows)*
Alignment: <TD ALIGN=LEFT|RIGHT| CENTER|MIDDLE|BOTTOM
VALIGN=TOP|BOTTOM|MIDDLE>
No linebreaks: <TD NOWRAP>
Columns to Span: <TD COLSPAN=?>
Rows to Span: <TD ROWSPAN=?>
Desired Width: <TD WIDTH=?>
Width Percent: <TD WIDTH="%"> – *(percentage of table)*
Cell Color: <TD BGCOLOR="#$$$$$$">
Table Header: <TH></TH> – *(same as data, except bold centered)*
Alignment: <TH ALIGN=LEFT|RIGHT| CENTER|MIDDLE|BOTTOM
VALIGN=TOP|BOTTOM|MIDDLE>
No Linebreaks: <TH NOWRAP>

Attributes of <tr> tag

Attribute name	**Notes**

<tr align="">	Sets the horizontal alignment for the contents of each <td> element in a table row.
<trvalign="">	Sets the vertical alignment of all content in a table row.
<trbgcolor="">	Sets the background color for a single table row in an HTML table.
<tr background="">	Identifies the URL of a file to be used as a background image for a table row.
<trbordercolor="">	Sets the border color for all inside borders of a table row.

Attributes of <td > tag

Attribute name	Notes
<td nowrap>	NOWRAP indicates that text should not wrap in the cell.
<td bgcolor="">	Sets the background color of a single cell in a table.
<td bordercolor="">	Sets the color of the entire border around a cell.
<td background="">	Specifies the URL of an image file to be used as the <td> element background image.
<td colspan="">	Indicates how many columns a cell should take up.
<td align="">	Was used to specify the alignment of the contents of a single table data cell. This attribute has been deprecated. Use CSS to control alignment of the contents of a table data cell.
<td width="">	Was used to set the width of a table data cell to a value that would override the default width. This attribute has been deprecated. Use CSS to control layout of data cells in HTML tables.

HTML - Frames

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

Disadvantages of Frames

There are few drawbacks with using frames, so it's never recommended to use frames in your webpages −

- Some smaller devices cannot cope with frames often because their screen is not big enough to be divided up.
- Sometimes your page will be displayed differently on different computers due to different screen resolution.
- The browser's *back* button might not work as the user hopes.
- There are still few browsers that do not support frame technology.

Creating Frames

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

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

Red.html

```
<!DOCTYPE html>
<html>
<body style="background-color:red;">
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
```

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

Green.html

```
<!DOCTYPE html>
<html>
<body style="background-color:green;">
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

Blue.html

```
<!DOCTYPE html>
<html>
<body style="background-color:blue;">
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

Horzframes.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML Frames</title>
  </head>
  <frameset rows = "10%,80%,10%">
    <frame name = "top" src = "red.html" />
    <frame name = "main" src = "blue.html" />
    <frame name = "bottom" src = "green.html" />
      <noframes>
      <body>Your browser does not support frames.</body>
    </noframes>
    </frameset>
  </html>
```

output

Vertframes.html

```
<!DOCTYPE html>
<html>
    <head>
    <title>HTML Frames</title>
  </head>
    <frameset cols = "25%,50%,25%">
    <frame name = "left" src = "red.html" />
```

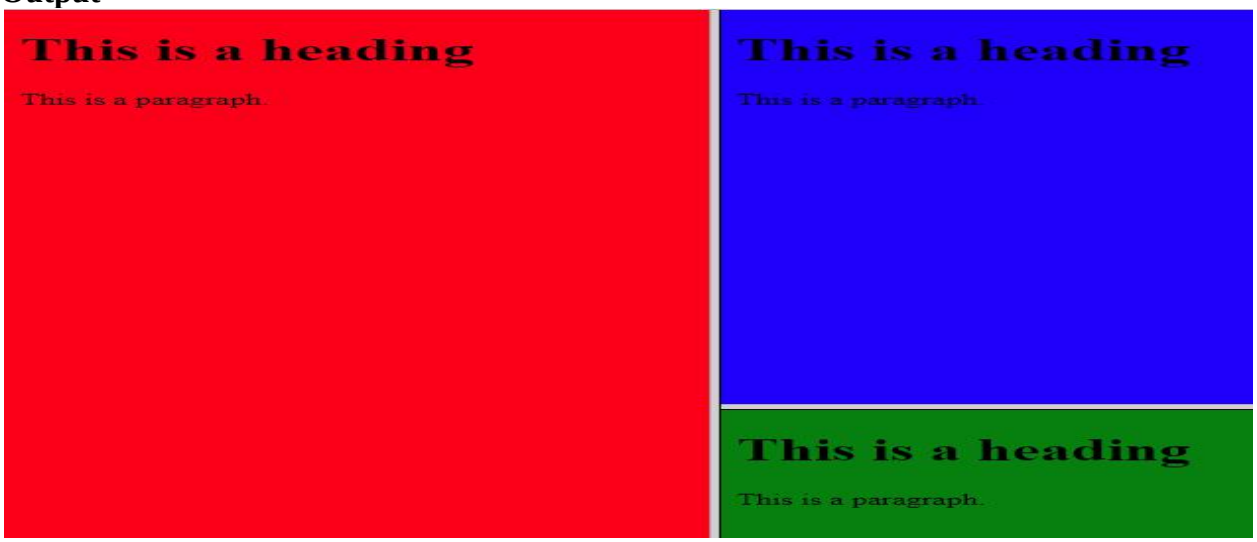
```
    <frame name = "center" src = "blue.html" />
    <frame name = "right" src = "green.html" />
        <noframes>
      <body>Your browser does not support frames.</body>
    </noframes>
  </frameset>
  </html>
```

output

Mixedframes.html

```
<html>
<frameset cols="25%,*" scrolling="no" noresize>
<frame name = "top" src = "red.html" />
<frameset rows="50%,*" scrolling="no" noresize>
    <frame name = "main" src = "blue.html" />
    <frame name = "bottom" src = "green.html" />
</frameset>
</html>
```

Output



The <frameset> Tag Attributes

Following are important attributes of the <frameset> tag −

Sr.No	Attribute & Description
1	**cols** Specifies how many columns are contained in the frameset and the size of each column. You can specify the width of each column in one of the four ways − Absolute values in pixels. For example, to create three vertical frames, use *cols = "100, 500, 100"*. A percentage of the browser window. For example, to create three vertical frames,

	use *cols = "10%, 80%, 10%"*. Using a wildcard symbol. For example, to create three vertical frames, use *cols = "10%, *, 10%"*. In this case wildcard takes remainder of the window. As relative widths of the browser window. For example, to create three vertical frames, use *cols = "3*, 2*, 1*"*. This is an alternative to percentages. You can use relative widths of the browser window. Here the window is divided into sixths: the first column takes up half of the window, the second takes one third, and the third takes one sixth.
2	**rows** This attribute works just like the cols attribute and takes the same values, but it is used to specify the rows in the frameset. For example, to create two horizontal frames, use *rows = "10%, 90%"*. You can specify the height of each row in the same way as explained above for columns.
3	**border** This attribute specifies the width of the border of each frame in pixels. For example, border = "5". A value of zero means no border.
4	**frameborder** This attribute specifies whether a three-dimensional border should be displayed between frames. This attribute takes value either 1 (yes) or 0 (no). For example frameborder = "0" specifies no border.
5	**framespacing** This attribute specifies the amount of space between frames in a frameset. This can take any integer value. For example framespacing = "10" means there should be 10 pixels spacing between each frames.

The <frame> Tag Attributes

Following are the important attributes of <frame> tag −

Sr.No	Attribute & Description
1	**src** This attribute is used to give the file name that should be loaded in the frame. Its value can be any URL. For example, src = "/html/top_frame.htm" will load an HTML file available in html directory.
2	**name**

	This attribute allows you to give a name to a frame. It is used to indicate which frame a document should be loaded into. This is especially important when you want to create links in one frame that load pages into an another frame, in which case the second frame needs a name to identify itself as the target of the link.
3	**frameborder** This attribute specifies whether or not the borders of that frame are shown; it overrides the value given in the frameborder attribute on the <frameset> tag if one is given, and this can take values either 1 (yes) or 0 (no).
4	**marginwidth** This attribute allows you to specify the width of the space between the left and right of the frame's borders and the frame's content. The value is given in pixels. For example marginwidth = "10".
5	**marginheight** This attribute allows you to specify the height of the space between the top and bottom of the frame's borders and its contents. The value is given in pixels. For example marginheight = "10".
6	**noresize** By default, you can resize any frame by clicking and dragging on the borders of a frame. The noresize attribute prevents a user from being able to resize the frame. For example noresize = "noresize".
7	**scrolling** This attribute controls the appearance of the scrollbars that appear on the frame. This takes values either "yes", "no" or "auto". For example scrolling = "no" means it should not have scroll bars.
8	**longdesc** This attribute allows you to provide a link to another page containing a long description of the contents of the frame. For example longdesc = "framedescription.htm"

HTML Forms

The <form> Element

The HTML <form> element defines a form that is used to collect user input:

```
<form>
.
form elements
.
</form>
```

An HTML form contains **form elements**.

Form elements are different types of input elements, like text fields, checkboxes, radio buttons, submit buttons, and more.

The <input> Element

The <input> element is the most important form element.

The <input> element can be displayed in several ways, depending on the **type**attribute.

Here are some examples:

HTML Input Types

Here are the different input types you can use in HTML:

```
<input type="button">
<input type="checkbox">
<input type="color">
<input type="date">
<input type="datetime-local">
<input type="email">
<input type="file">
<input type="hidden">
<input type="image">
<input type="month">
<input type="number">
<input type="password">
<input type="radio">
<input type="range">
<input type="reset">
<input type="search">
<input type="submit">
<input type="tel">
<input type="text">
<input type="time">
<input type="url">
<input type="week">
```

Type	Description
<input type="text">	Defines a one-line text input field
<input type="radio">	Defines a radio button (for selecting one of many choices)
<input type="submit">	Defines a submit button (for submitting the form)

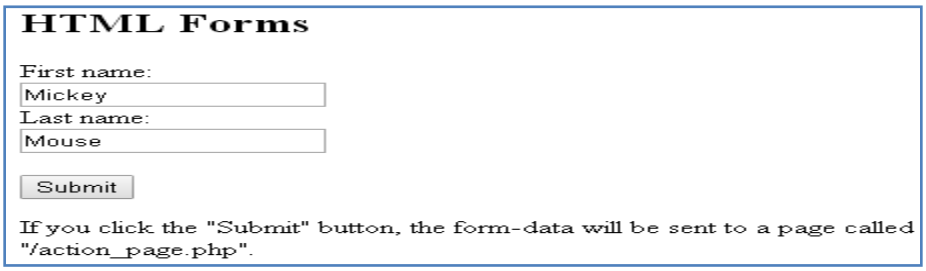
```
<!DOCTYPE html>
```

```
<html>
<body>

<h2>HTML Forms</h2>

<form action="welcome1.php">
  First name:<br>
  <input type="text" name="firstname" value="Mickey">
  <br>
  Last name:<br>
  <input type="text" name="lastname" value="Mouse">
  <br><br>
  <input type="submit" value="Submit">
</form>
<p>If you click the "Submit" button, the form-data will be sent to a page called
"welcome1.php".</p>
```

welcome1.php

```
<html>
<body>
Welcome  your first name is: <?php echo $_POST["firstname "]; ?><br>
Your Last name is:  <?php echo $_POST["lastname "]; ?>
</body>
</html>
```

Output



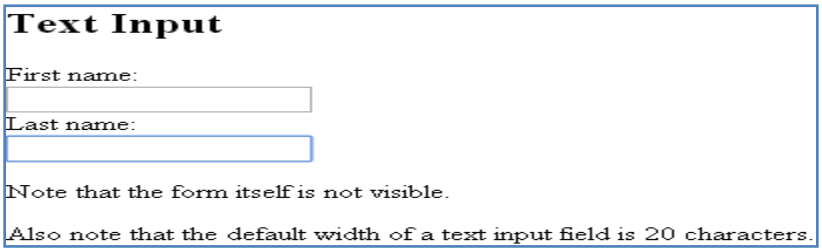
Text Input

<input type="text"> defines a one-line input field for **text input**:

Example

```
<form>
  First name:<br>
  <input type="text" name="firstname"><br>
  Last name:<br>
  <input type="text" name="lastname">
</form>
```

output



Radio Button Input

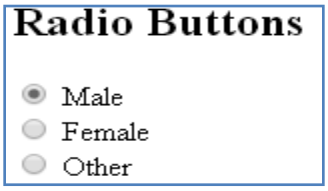
<input type="radio"> defines a **radio button**.

Radio buttons let a user select ONE of a limited number of choices:

Example

```
<form>
  <input type="radio" name="gender" value="male" checked>Male<br>
  <input type="radio" name="gender" value="female"> Female<br>
  <input type="radio" name="gender" value="other"> Other
</form>
```

Output



The Submit Button

<input type="submit"> defines a button for **submitting** the form data to a**form-handler**.

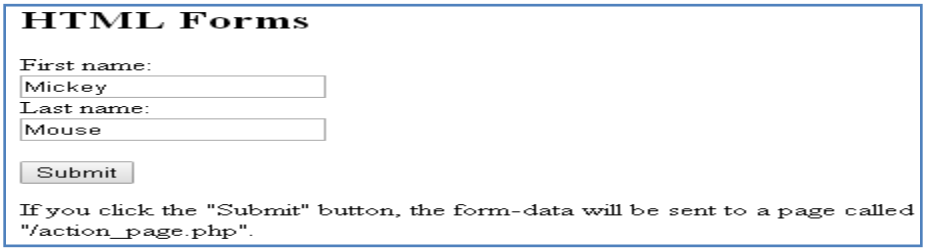
The form-handler is typically a server page with a script for processing input data.

The form-handler is specified in the form's **action** attribute:

Example

```
<form action="/action_page.php">
  First name:<br>
  <input type="text" name="firstname" value="Mickey"><br>
  Last name:<br>
  <input type="text" name="lastname" value="Mouse"><br><br>
  <input type="submit" value="Submit">
</form>
```

Output



The Action Attribute

The action attribute defines the action to be performed when the form is submitted.

Normally, the form data is sent to a web page on the server when the user clicks on the submit button.

In the example above, the form data is sent to a page on the server called "/action_page.php". This page contains a server-side script that handles the form data:

```
<form action="/action_page.php">
```

If the action attribute is omitted, the action is set to the current page.

The Target Attribute

The target attribute specifies if the submitted result will open in a new browser tab, a frame, or in the current window.

The default value is "_self" which means the form will be submitted in the current window.

To make the form result open in a new browser tab, use the value "_blank":

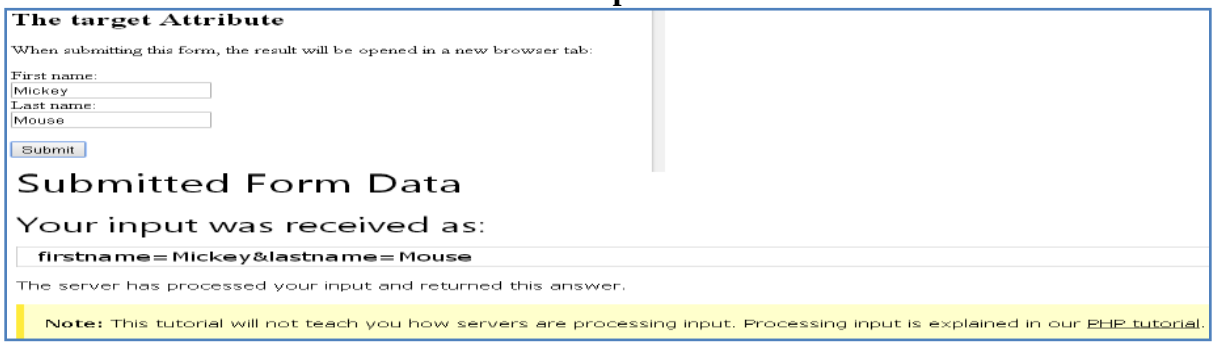
Other legal values are "_parent", "_top", or a name representing the name of an iframe.

Example

```
<form action="/action_page.php" target="_blank">
```

```
<!DOCTYPE html>
<html>
<body>
<h2>The target Attribute</h2>
<p>When submitting this form, the result will be opened in a new browser tab:</p>
<form action="/action_page.php" target="_blank">
  First name:<br>
  <input type="text" name="firstname" value="Mickey">
  <br>
  Last name:<br>
  <input type="text" name="lastname" value="Mouse">
  <br><br>
  <input type="submit" value="Submit">
</form>
</body>
</html>
```

Output



The Method Attribute

The method attribute specifies the HTTP method (**GET** or **POST**) to be used when submitting the form data:

Example

```
<form action="/action_page.php" method="get">
<!DOCTYPE html>
<html>
<body>
<h2>The method Attribute</h2>
<p>This form will be submitted using the GET method:</p>
<form action="/action_page.php" target="_blank" method="GET">
  First name:<br>
  <input type="text" name="firstname" value="Mickey">
  <br>
  Last name:<br>
  <input type="text" name="lastname" value="Mouse">
  <br><br>
  <input type="submit" value="Submit">
</form>
<p>After you submit, notice that the form values is visible in the address bar of the new browser tab.</p>
</body>
</html>
```

Example

```
<form action="/action_page.php" method="post">
<!DOCTYPE html>
<html>
<body>
<h2>The method Attribute</h2>
<p>This form will be submitted using the POST method:</p>
<form action="/action_page.php" target="_blank" method="POST">
  First name:<br>
  <input type="text" name="firstname" value="Mickey">
  <br>
  Last name:<br>
  <input type="text" name="lastname" value="Mouse">
  <br><br>
  <input type="submit" value="Submit">
</form>
<p>After you submit, notice that, unlike the GET method, the form values is NOT visible in the address bar of the new browser tab.</p>
</body>
</html>
```

Output

When to Use GET?

The default method when submitting form data is GET.

However, when GET is used, the submitted form data will be **visible in the page address field**: /action_page.php?firstname=Mickey&lastname=Mouse

Notes on GET:

- Appends form-data into the URL in name/value pairs
- The length of a URL is limited (about 3000 characters)
- Never use GET to send sensitive data! (will be visible in the URL)
- Useful for form submissions where a user wants to bookmark the result

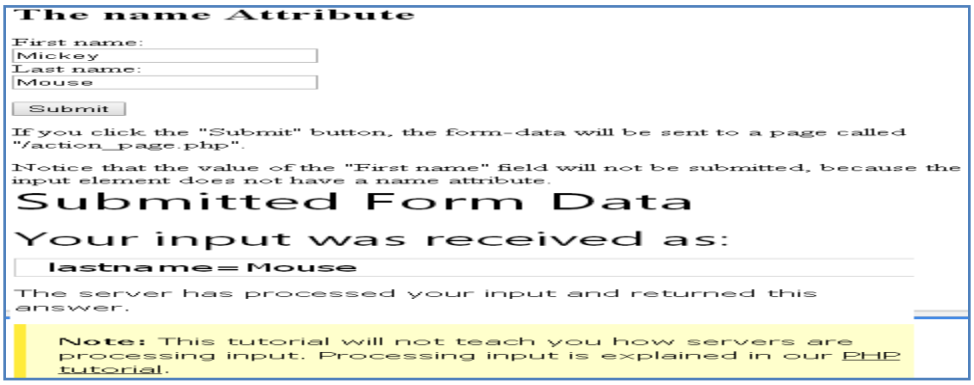
- GET is better for non-secure data, like query strings in Google

When to Use POST?
Always use POST if the form data contains sensitive or personal information. The POST method does not display the submitted form data in the page address field.
Notes on POST:

- POST has no size limitations, and can be used to send large amounts of data.
- Form submissions with POST cannot be bookmarked

The Name Attribute
Each input field must have a name attribute to be submitted.
If the name attribute is omitted, the data of that input field will not be sent at all.
This example will only submit the "Last name" input field:
Example

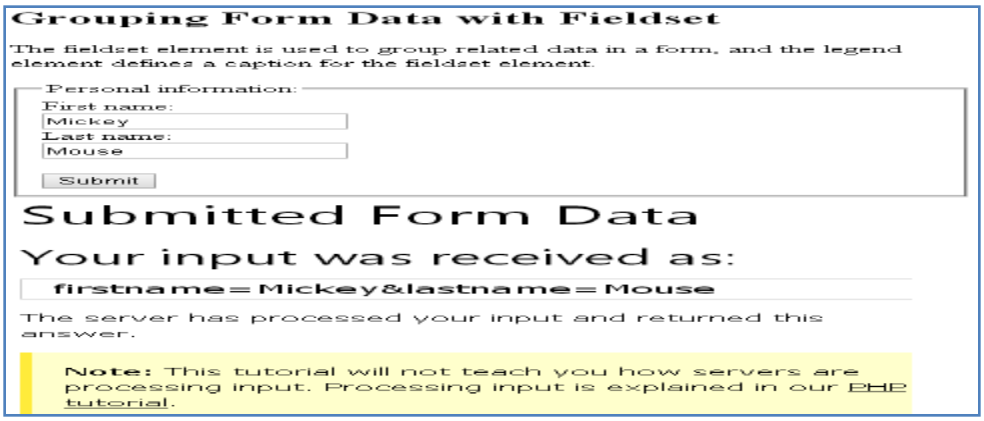
```
<form action="/action_page.php">
  First name:<br>
  <input type="text" value="Mickey"><br>
  Last name:<br>
  <input type="text" name="lastname" value="Mouse"><br><br>
  <input type="submit" value="Submit">
</form>
```



Grouping Form Data with <fieldset>
The <fieldset> element is used to group related data in a form.
The <legend> element defines a caption for the <fieldset> element.
Example

```
<form action="/action_page.php">
  <fieldset>
    <legend>Personal information:</legend>
    First name:<br>
    <input type="text" name="firstname" value="Mickey"><br>
    Last name:<br>
    <input type="text" name="lastname" value="Mouse"><br><br>
    <input type="submit" value="Submit">
  </fieldset>
</form>
```

Output



HTML Form Elements

The <input> Element

The most important form element is the <input> element.

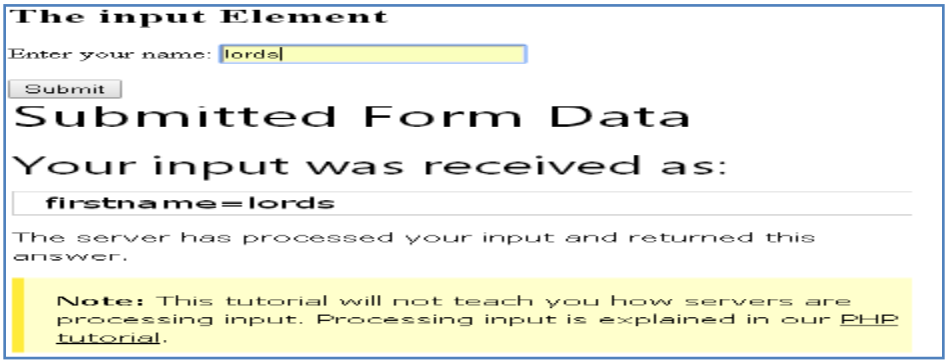
The <input> element can be displayed in several ways, depending on the typeattribute.

If the type attribute is omitted, the input field gets the default type: "text".

Example

```
<input name="firstname" type="text">
<!DOCTYPE html>
<html>
<body>
<h2>The input Element</h2>

<form action="/action_page.php">
  Enter your name:
  <input name="firstname" type="text">
  <br><br>
  <input type="submit">
</form>
</body>
</html>
```

Output



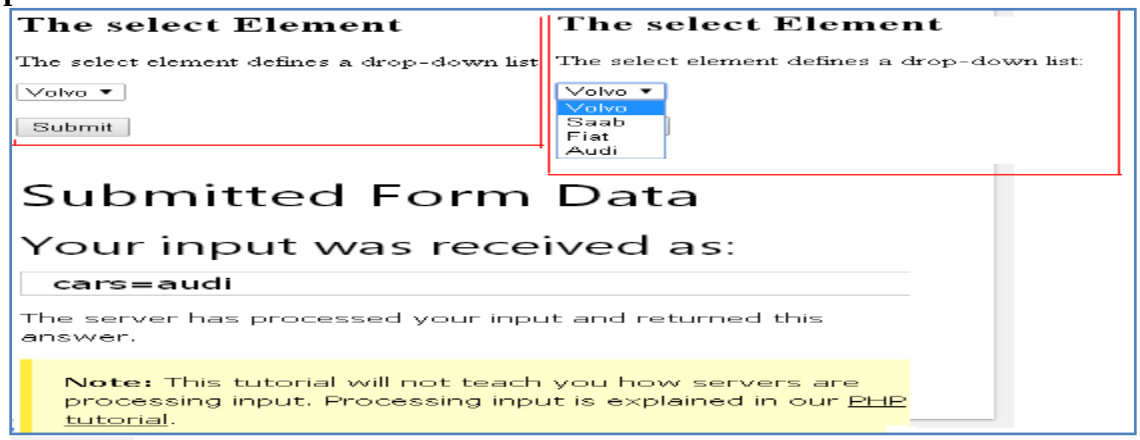
The <select> Element

The <select> element defines a **drop-down list**:

```
<!DOCTYPE html>
<html>
<body>
<h2>The select Element</h2>
<p>The select element defines a drop-down list:</p>
<form action="/action_page.php">
```

```
  <select name="cars">
    <option value="volvo">Volvo</option>
    <option value="saab">Saab</option>
    <option value="fiat">Fiat</option>
    <option value="audi">Audi</option>
  </select>
  <br><br>
  <input type="submit">
</form>
</body>
</html>
```

Output



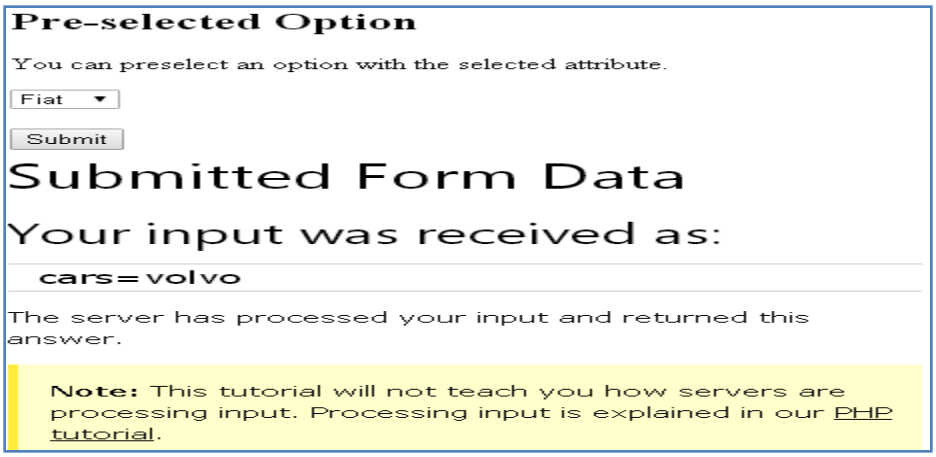
The <option> elements defines an option that can be selected.

By default, the first item in the drop-down list is selected.

To define a pre-selected option, add the selected attribute to the option:

```
<!DOCTYPE html>
<html>
<body>
<h2>Pre-selected Option</h2>
<p>You can preselect an option with the selected attribute.</p>
<form action="/action_page.php">
  <select name="cars">
    <option value="volvo">Volvo</option>
    <option value="saab">Saab</option>
    <option value="fiat" selected>Fiat</option>
    <option value="audi">Audi</option>
  </select>
  <br><br>
  <input type="submit">
</form>
</body>
</html>
```

Output

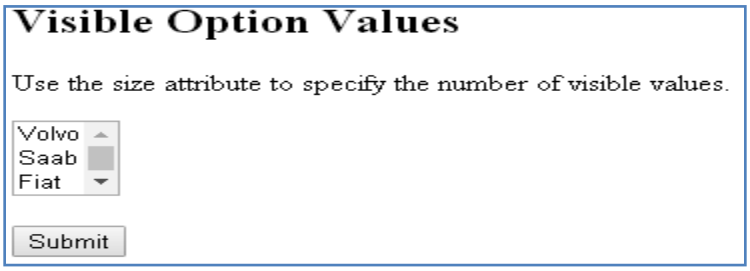


Visible Values:

Use the size attribute to specify the number of visible values:

```
<!DOCTYPE html>
<html>
<body>

<h2>Visible Option Values</h2>
<p>Use the size attribute to specify the number of visible values.</p>
<form action="/action_page.php">
  <select name="cars" size="3">
    <option value="volvo">Volvo</option>
    <option value="saab">Saab</option>
    <option value="fiat">Fiat</option>
    <option value="audi">Audi</option>
  </select>
  <br><br>
  <input type="submit">
</form>
</body>
</html>
```

Output



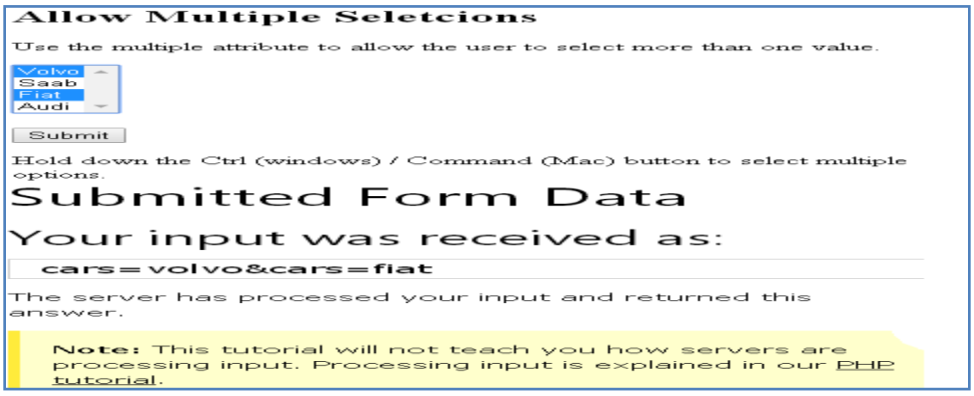
Allow Multiple Selections:

Use the multiple attribute to allow the user to select more than one value:

```
<!DOCTYPE html>
<html>
<body>
<h2>Allow Multiple Seletcions</h2>
<p>Use the multiple attribute to allow the user to select more than one value.</p>
<form action="/action_page.php">
  <select name="cars" size="4" multiple>
    <option value="volvo">Volvo</option>
```

```
    <option value="saab">Saab</option>
    <option value="fiat">Fiat</option>
    <option value="audi">Audi</option>
  </select>
  <br><br>
  <input type="submit">
</form>
<p>Hold down the Ctrl (windows) / Command (Mac) button to select multiple options.</p>
</body>
</html>
```

Output

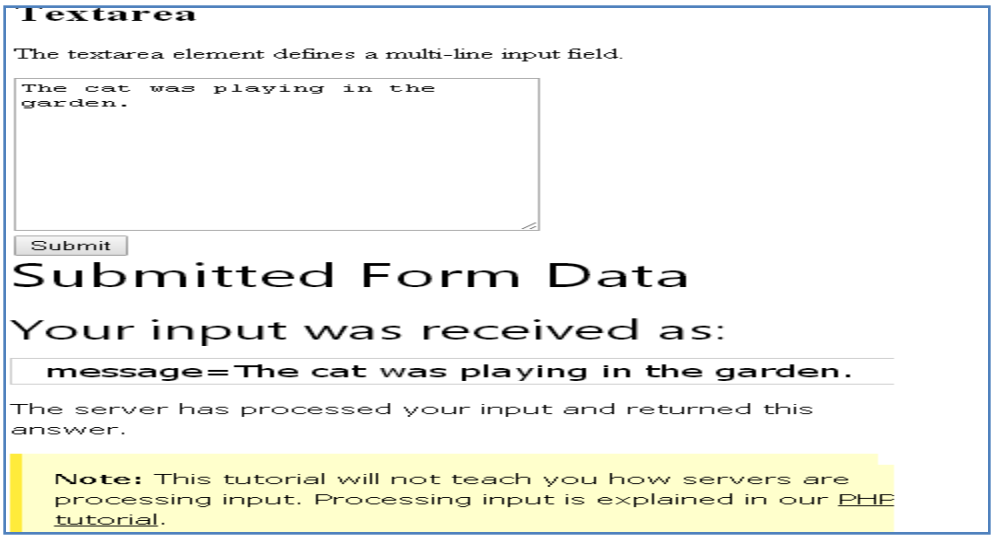


The <textarea> Element

The <textarea> element defines a multi-line input field (**a text area**):
The rows attribute specifies the visible number of lines in a text area.
The cols attribute specifies the visible width of a text area.

```
<!DOCTYPE html>
<html>
<body>
<h2>Textarea</h2>
<p>The textarea element defines a multi-line input field.</p>
<form action="/action_page.php">
  <textarea name="message" rows="10" cols="30">The cat was playing in the
garden.</textarea>
  <br>
  <input type="submit">
</form>
</body>
</html>
```

Output



The <button> Element

The <button> element defines a clickable **button**:

```
<!DOCTYPE html>
<html>
<body>
<h2>The button Element</h2>
<button type="button" onclick="alert('Hello World!')">Click Me!</button>
</body>
</html>
```

Output

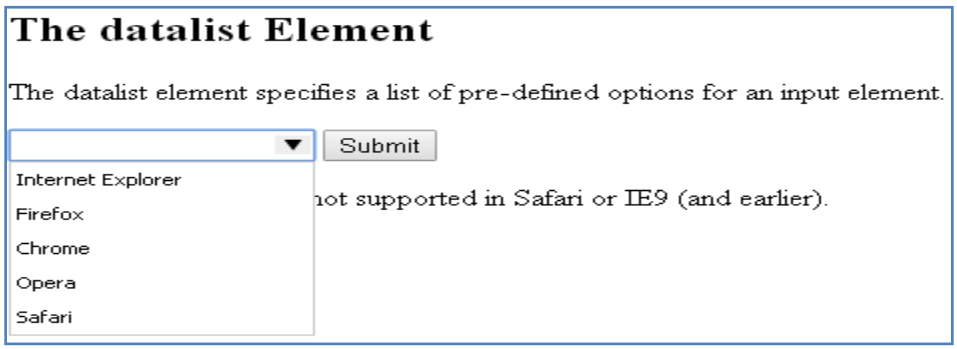
HTML5 <datalist> Element

The <datalist> element specifies a list of pre-defined options for an <input>element.
Users will see a drop-down list of the pre-defined options as they input data.
The list attribute of the <input> element, must refer to the id attribute of the<datalist> element.

```
<!DOCTYPE html>
<html>
<body>

<h2>The datalist Element</h2>
<p>The datalist element specifies a list of pre-defined options for an input element.</p>
<form action="/action_page.php">
  <input list="browsers" name="browser">
  <datalist id="browsers">
    <option value="Internet Explorer">
    <option value="Firefox">
    <option value="Chrome">
    <option value="Opera">
    <option value="Safari">
```

```
  </datalist>
  <input type="submit">
</form>
<p><b>Note:</b> The datalist tag is not supported in Safari or IE9 (and earlier).</p>
</body>
</html>
```

Output

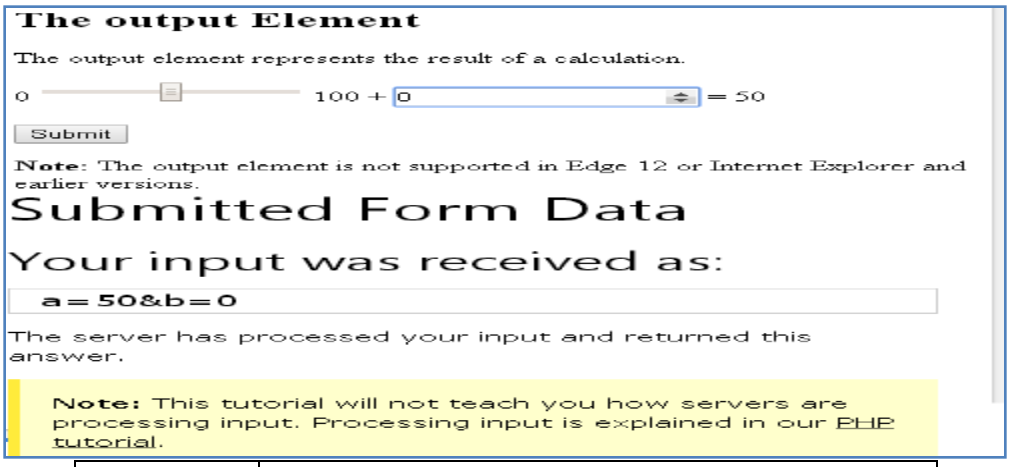


HTML5 <output> Element

The <output> element represents the result of a calculation (like one performed by a script).

```
<!DOCTYPE html>
<html>
<body>
<h2>The output Element</h2>
<p>The output element represents the result of a calculation.</p>
<form action="/action_page.php"
oninput="x.value=parseInt(a.value)+parseInt(b.value)">
  0
  <input type="range" id="a" name="a" value="50">
  100 +
  <input type="number" id="b" name="b" value="50">
  =
  <output name="x" for="a b"></output>
  <br><br>
  <input type="submit">
</form>
<p><strong>Note:</strong> The output element is not supported in Edge 12 or Internet Explorer
and earlier versions.</p>
</body>
</html>
```

Output



Tag	Description
<form>	Defines an HTML form for user input
<input>	Defines an input control
<textarea>	Defines a multiline input control (text area)
<label>	Defines a label for an <input> element
<fieldset>	Groups related elements in a form
<legend>	Defines a caption for a <fieldset> element
<select>	Defines a drop-down list
<optgroup>	Defines a group of related options in a drop-down list
<option>	Defines an option in a drop-down list
<button>	Defines a clickable button
<datalist>	Specifies a list of pre-defined options for input controls
<output>	Defines the result of a calculation

HTML Input Types

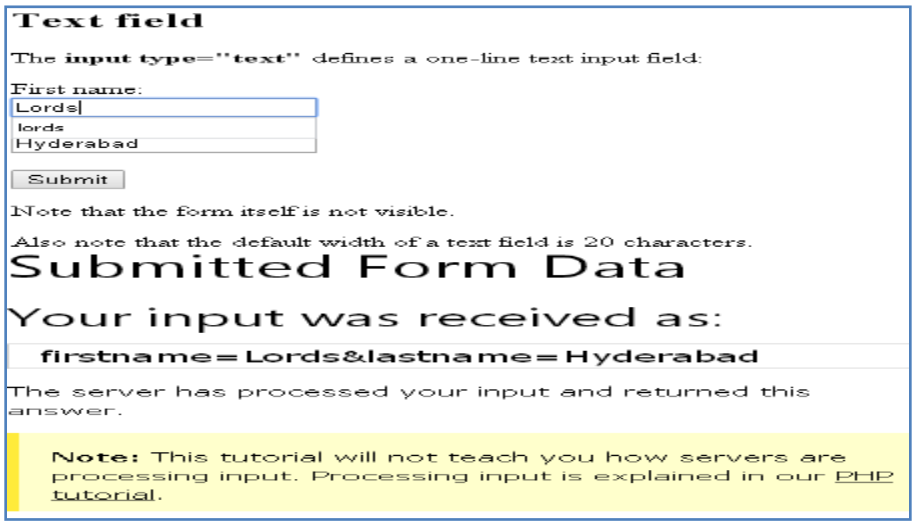
Input Type Text

<input type="text"> defines a **one-line text input field**:

```
<!DOCTYPE html>
<html>
<body>
<h2>Text field</h2>
<p>The <strong>input type="text"</strong> defines a one-line text input field:</p>
```

```
<form action="/action_page.php">
First name:<br>
<input type="text" name="firstname">
<br>
Last name:<br>
<input type="text" name="lastname">
<br><br>
<input type="submit">
</form>
<p>Note that the form itself is not visible.</p>
<p>Also note that the default width of a text field is 20 characters.</p>
</body>
</html>
```

Output



Input Type Password

<input type="password"> defines a **password field**:

```
<!DOCTYPE html>
<html>
<body>
<h2>Password field</h2>
<p>The <strong>input type="password"</strong> defines a password field:</p>
<form action="">
User name:<br>
<input type="text" name="userid">
<br>
User password:<br>
<input type="password" name="psw">
</form>
<p>The characters in a password field are masked (shown as asterisks or circles).</p>
</body>
</html>
```

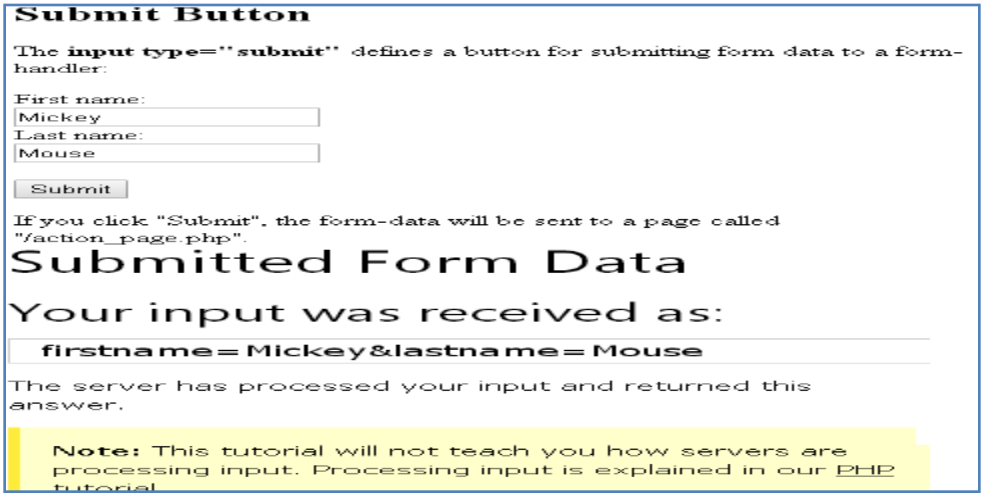
Output

Input Type Submit

<input type="submit"> defines a button for **submitting** form data to a **form-handler**.
The form-handler is typically a server page with a script for processing input data.
The form-handler is specified in the form's action attribute:

```
<!DOCTYPE html>
<html>
<body>
<h2>Submit Button</h2>
<p>The <strong>input type="submit"</strong> defines a button for submitting form data to a form-handler:</p>
<form action="/action_page.php">
First name:<br>
<input type="text" name="firstname" value="Mickey">
<br>
Last name:<br>
<input type="text" name="lastname" value="Mouse">
<br><br>
<input type="submit" value="Submit">
</form>
<p>If you click "Submit", the form-data will be sent to a page called "/action_page.php".</p>
</body>
</html>
```

Output



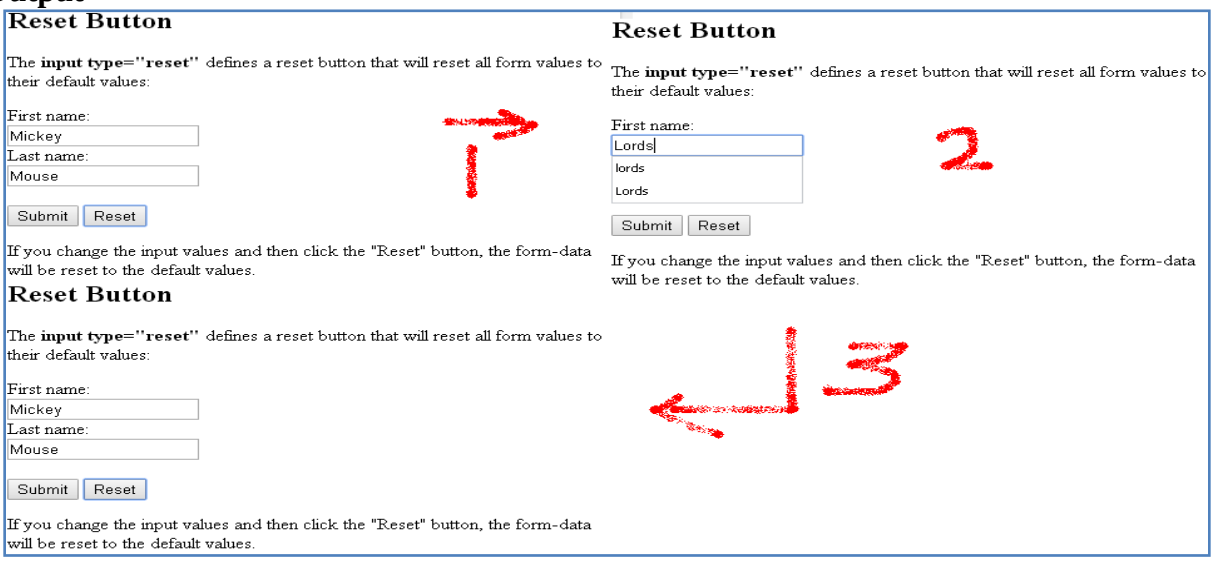
Input Type Reset

<input type="reset"> defines a **reset button** that will reset all form values to their default values:

```
<!DOCTYPE html>
<html>
```

```
<body>
<h2>Reset Button</h2>
<p>The <strong>input type="reset"</strong> defines a reset button that will reset all form values to their default values:</p>
<form action="/action_page.php">
First name:<br>
<input type="text" name="firstname" value="Mickey">
<br>
Last name:<br>
<input type="text" name="lastname" value="Mouse">
<br><br>
<input type="submit" value="Submit">
<input type="reset">
</form>
<p>If you change the input values and then click the "Reset" button, the form-data will be reset to the default values.</p>
</body>
</html>
```

Output



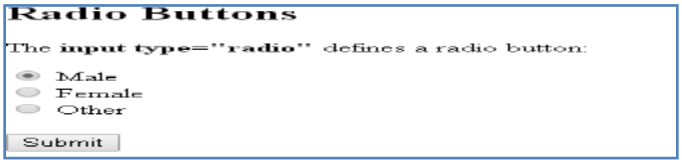
Input Type Radio

<input type="radio"> defines a **radio button**.

Radio buttons let a user select ONLY ONE of a limited number of choices:

```
<!DOCTYPE html>
<html>
<body>
<h2>Radio Buttons</h2>
<p>The <strong>input type="radio"</strong> defines a radio button:</p>
<form action="/action_page.php">
  <input type="radio" name="gender" value="male" checked> Male<br>
  <input type="radio" name="gender" value="female"> Female<br>
  <input type="radio" name="gender" value="other"> Other<br><br>
  <input type="submit">
</form>
</body>
</html>
```

Output



Input Type Checkbox

<input type="checkbox"> defines a **checkbox**.

Checkboxes let a user select ZERO or MORE options of a limited number of choices.

```
<!DOCTYPE html>
<html>
<body>
<h2>Checkboxes</h2>
<p>The <strong>input type="checkbox"</strong> defines a checkbox:</p>
<form action="/action_page.php">
<input type="checkbox" name="vehicle1" value="Bike">I have a bike
<br>
<input type="checkbox" name="vehicle2" value="Car">I have a car
<br><br>
<input type="submit">
</form>
</body>
</html>
```

Output

Input Type Button

<input type="button"> defines a **button**:

```
<!DOCTYPE html>
<html>
<body>
<h2>Input Button</h2>
<input type="button" onclick="alert('Hello World!')" value="Click Me!">
</body>
</html>
```

Output

https://www.youtube.com/watch?v=OjwV5X1Ugvs

https://www.tutorialspoint.com/xml/index.htm

XML Tutorial

XML stands for **Ex**tensible **M**arkup **L**anguage and is a text-based markup language derived from Standard Generalized Markup Language (SGML). The tutorial is divided into sections such as XML Basics, Advanced XML, and XML tools.

Prerequisites

Before proceeding with this tutorial, you should have basic knowledge of HTML and JavaScript.

XML - Overview

XML stands for **E**xtensible **M**arkup **L**anguage. It is a text-based markup language derived from Standard Generalized Markup Language (SGML).

XML tags identify the data and are used to store and organize the data, rather than specifying how to display it like HTML tags, which are used to display the data. XML is not going to replace HTML in the near future, but it introduces new possibilities by adopting many successful features of HTML.

There are three important characteristics of XML that make it useful in a variety of systems and solutions −

- **XML is extensible** − XML allows you to create your own self-descriptive tags, or language, that suits your application.
- **XML carries the data, does not present it** − XML allows you to store the data irrespective of how it will be presented.
- **XML is a public standard** − XML was developed by an organization called the World Wide Web Consortium (W3C) and is available as an open standard.

XML Usage

A short list of XML usage says it all −

- XML can work behind the scene to simplify the creation of HTML documents for large web sites.
- XML can be used to exchange the information between organizations and systems.
- XML can be used for offloading and reloading of databases.
- XML can be used to store and arrange the data, which can customize your data handling needs.
- XML can easily be merged with style sheets to create almost any desired output.
- Virtually, any type of data can be expressed as an XML document.

What is Markup?

XML is a markup language that defines set of rules for encoding documents in a format that is both human-readable and machine-readable. So what exactly is a markup language? Markup is information added to a document that enhances its meaning in certain ways, in that it identifies the parts and how they relate to each other. More specifically, a markup language is a set of symbols that can be placed in the text of a document to demarcate and label the parts of that document.

Following example shows how XML markup looks, when embedded in a piece of text −

```
<message>
  <text>Hello, world!</text>
```

```
</message>
```

This snippet includes the markup symbols, or the tags such as <message>...</message> and <text>... </text>. The tags <message> and </message> mark the start and the end of the XML code fragment. The tags <text> and </text> surround the text Hello, world!.

Is XML a Programming Language?

A programming language consists of grammar rules and its own vocabulary which is used to create computer programs. These programs instruct the computer to perform specific tasks. XML does not qualify to be a programming language as it does not perform any computation or algorithms. It is usually stored in a simple text file and is processed by special software that is capable of interpreting XML.

XML - Syntax

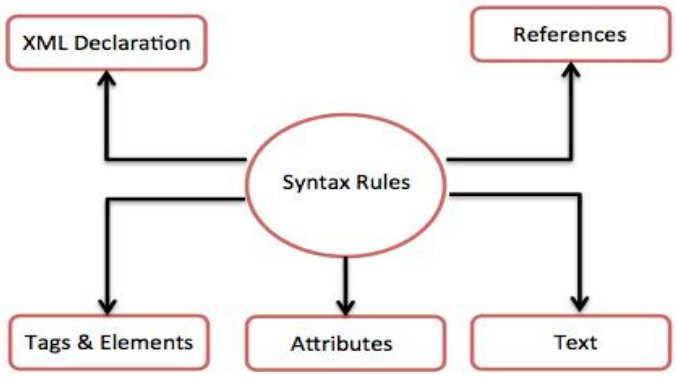
In this chapter, we will discuss the simple syntax rules to write an XML document. Following is a complete XML document −

```
<?xml version = "1.0"?>
<contact-info>
   <name>Tanmay Patil</name>
   <company>TutorialsPoint</company>
   <phone>(011) 123-4567</phone>
</contact-info>
```

You can notice there are two kinds of information in the above example −

- Markup, like <contact-info>
- The text, or the character data, Tutorials Point and (040) 123-4567.

The following diagram depicts the syntax rules to write different types of markup and text in an XML document.



Let us see each component of the above diagram in detail.

XML Declaration

The XML document can optionally have an XML declaration. It is written as follows −

```
<?xml version = "1.0" encoding = "UTF-8"?>
```

Where version is the XML version and encoding specifies the character encoding used in the document.

Syntax Rules for XML Declaration

- The XML declaration is case sensitive and must begin with "**<?xml>**" where "**xml**" is written in lower-case.

- If document contains XML declaration, then it strictly needs to be the first statement of the XML document.
- The XML declaration strictly needs be the first statement in the XML document.
- An HTTP protocol can override the value of encoding that you put in the XML declaration.

Tags and Elements

An XML file is structured by several XML-elements, also called XML-nodes or XML-tags. The names of XML-elements are enclosed in triangular brackets < > as shown below −

```
<element>
```

Syntax Rules for Tags and Elements

Element Syntax − Each XML-element needs to be closed either with start or with end elements as shown below −

```
<element>....</element>
```

or in simple-cases, just this way −

```
<element/>
```

Nesting of Elements − An XML-element can contain multiple XML-elements as its children, but the children elements must not overlap. i.e., an end tag of an element must have the same name as that of the most recent unmatched start tag.

The Following example shows incorrect nested tags −

```
<?xml version = "1.0"?>
<contact-info>
<company>TutorialsPoint
<contact-info>
</company>
```

The Following example shows correct nested tags −

```
<?xml version = "1.0"?>
<contact-info>
   <company>TutorialsPoint</company>
<contact-info>
```

Root Element − An XML document can have only one root element. For example, following is not a correct XML document, because both the **x** and **y** elements occur at the top level without a root element −

```
<x>...</x>
<y>...</y>
```

The Following example shows a correctly formed XML document −

```
<root>
   <x>...</x>
   <y>...</y>
</root>
```

Case Sensitivity − The names of XML-elements are case-sensitive. That means the name of the start and the end elements need to be exactly in the same case.

For example, **<contact-info>** is different from **<Contact-Info>**

XML Attributes

An **attribute** specifies a single property for the element, using a name/value pair. An XML-element can have one or more attributes. For example −

```
<a href = "http://www.tutorialspoint.com/">Tutorialspoint!</a>
```

Here **href** is the attribute name and**http://www.tutorialspoint.com/** is attribute value.

Syntax Rules for XML Attributes

- Attribute names in XML (unlike HTML) are case sensitive. That is, HREF and href are considered two different XML attributes.
- Same attribute cannot have two values in a syntax. The following example shows incorrect syntax because the attribute b is specified twice
  –

```
<a b = "x" c = "y" b = "z">....</a>
```

- Attribute names are defined without quotation marks, whereas attribute values must always appear in quotation marks. Following example demonstrates incorrect xml syntax
  –

```
<a b = x>....</a>
```

In the above syntax, the attribute value is not defined in quotation marks.

XML References

References usually allow you to add or include additional text or markup in an XML document. References always begin with the symbol **"&"** which is a reserved character and end with the symbol **";".** XML has two types of references −

- **Entity References** − An entity reference contains a name between the start and the end delimiters. For example**&** where amp is name. The name refers to a predefined string of text and/or markup.
- **Character References** − These contain references, such as **A**, contains a hash mark (“#”) followed by a number. The number always refers to the Unicode code of a character. In this case, 65 refers to alphabet "A".

XML Text

The names of XML-elements and XML-attributes are case-sensitive, which means the name of start and end elements need to be written in the same case. To avoid character encoding problems, all XML files should be saved as Unicode UTF-8 or UTF-16 files.

Whitespace characters like blanks, tabs and line-breaks between XML-elements and between the XML-attributes will be ignored.

Some characters are reserved by the XML syntax itself. Hence, they cannot be used directly. To use them, some replacement-entities are used, which are listed below −

Not Allowed Character	Replacement Entity	Character Description
<	&lt;	less than
>	&gt;	greater than
&	&amp;	ampersand
'	&apos;	apostrophe

"	&quot;	quotation mark

XML - Documents

An XML document is a basic unit of XML information composed of elements and other markup in an orderly package. An XMLdocument can contains wide variety of data. For example, database of numbers, numbers representing molecular structure or a mathematical equation.

XML Document Example

A simple document is shown in the following example −

```
<?xml version = "1.0"?>
<contact-info>
   <name>Tanmay Patil</name>
   <company>TutorialsPoint</company>
   <phone>(011) 123-4567</phone>
</contact-info>
```

The following image depicts the parts of XML document.

Document Prolog Section

Document Prolog comes at the top of the document, before the root element. This section contains −

- XML declaration
- Document type declaration

You can learn more about XML declaration in this chapter − XML Declaration

Document Elements Section

Document Elements are the building blocks of XML. These divide the document into a hierarchy of sections, each serving a specific purpose. You can separate a document into multiple sections so that they can be rendered differently, or used by a search engine. The elements can be containers, with a combination of text and other elements.

XML - Declaration

This chapter covers XML declaration in detail. **XML declaration**contains details that prepare an XML processor to parse the XML document. It is optional, but when used, it must appear in the first line of the XML document.

Syntax

Following syntax shows XML declaration −

```
<?xml
   version = "version_number"
   encoding = "encoding_declaration"
   standalone = "standalone_status"
?>
```

Each parameter consists of a parameter name, an equals sign (=), and parameter value inside a quote. Following table shows the above syntax in detail −

Parameter	Parameter_value	Parameter_description
Version	1.0	Specifies the version of the XML standard used.
Encoding	UTF-8, UTF-16, ISO-10646-UCS-2, ISO-10646-UCS-4, ISO-8859-1 to ISO-8859-9, ISO-2022-JP, Shift_JIS, EUC-JP	It defines the character encoding used in the document. UTF-8 is the default encoding used.
Standalone	yes or no	It informs the parser whether the document relies on the information from an external source, such as external document type definition (DTD), for its content. The default value is set to no. Setting it to yestells the processor there are no external declarations required for parsing the document.

Rules

An XML declaration should abide with the following rules −

- If the XML declaration is present in the XML, it must be placed as the first line in the XML document.
- If the XML declaration is included, it must contain version number attribute.
- The Parameter names and values are case-sensitive.
- The names are always in lower case.
- The order of placing the parameters is important. The correct order is: version, encoding and standalone.
- Either single or double quotes may be used.
- The XML declaration has no closing tag i.e. **</?xml>**

XML Declaration Examples

Following are few examples of XML declarations −

XML declaration with no parameters −

```
<?xml >
```

XML declaration with version definition −

```
<?xml version = "1.0">
```

XML declaration with all parameters defined −

```
<?xml version = "1.0" encoding = "UTF-8" standalone = "no" ?>
```

XML declaration with all parameters defined in single quotes −

```
<?xml version = '1.0' encoding = 'iso-8859-1' standalone = 'no' ?>
```

XML - Tags

Let us learn about one of the most important part of XML, the XML tags. **XML tags** form the foundation of XML. They define the scope of an element in XML. They can also be used to

insert comments, declare settings required for parsing the environment, and to insert special instructions.

We can broadly categorize XML tags as follows −

Start Tag

The beginning of every non-empty XML element is marked by a start-tag. Following is an example of start-tag −

```
<address>
```

End Tag

Every element that has a start tag should end with an end-tag. Following is an example of end-tag −

```
</address>
```

Note, that the end tags include a solidus ("/") before the name of an element.

Empty Tag

The text that appears between start-tag and end-tag is called content. An element which has no content is termed as empty. An empty element can be represented in two ways as follows −

A start-tag immediately followed by an end-tag as shown below −

```
<hr></hr>
```

A complete empty-element tag is as shown below −

```
<hr />
```

Empty-element tags may be used for any element which has no content.

XML Tags Rules

Following are the rules that need to be followed to use XML tags −

Rule 1

XML tags are case-sensitive. Following line of code is an example of wrong syntax </Address>, because of the case difference in two tags, which is treated as erroneous syntax in XML.

```
<address>This is wrong syntax</Address>
```

Following code shows a correct way, where we use the same case to name the start and the end tag.

```
<address>This is correct syntax</address>
```

Rule 2

XML tags must be closed in an appropriate order, i.e., an XML tag opened inside another element must be closed before the outer element is closed. For example −

```
<outer_element>
  <internal_element>
    This tag is closed before the outer_element
  </internal_element>
</outer_element>
```

XML - Elements

XML elements can be defined as building blocks of an XML. Elements can behave as containers to hold text, elements, attributes, media objects or all of these.

Each XML document contains one or more elements, the scope of which are either delimited by start and end tags, or for empty elements, by an empty-element tag.

Syntax

Following is the syntax to write an XML element −

```
<element-name attribute1 attribute2>
....content
</element-name>
```

where,

- **element-name** is the name of the element. The name its case in the start and end tags must match.
- **attribute1, attribute2** are attributes of the element separated by white spaces. An attribute defines a property of the element. It associates a name with a value, which is a string of characters. An attribute is written as −

```
name = "value"
```

name is followed by an = sign and a string value inside double(" ") or single(' ') quotes.

Empty Element

An empty element (element with no content) has following syntax −

```
<name attribute1 attribute2.../>
```

Following is an example of an XML document using various XML element −

```
<?xml version = "1.0"?>
<contact-info>
   <address category = "residence">
      <name>Tanmay Patil</name>
      <company>TutorialsPoint</company>
      <phone>(011) 123-4567</phone>
   </address>
</contact-info>
```

XML Elements Rules

Following rules are required to be followed for XML elements −

- An element name can contain any alphanumeric characters. The only punctuation mark allowed in names are the hyphen (-), under-score (_) and period (.).
- Names are case sensitive. For example, Address, address, and ADDRESS are different names.
- Start and end tags of an element must be identical.
- An element, which is a container, can contain text or elements as seen in the above example.

XML - Attributes

his chapter describes the **XML attributes**. Attributes are part of XML elements. An element can have multiple unique attributes. Attribute gives more information about XML elements. To be more precise, they define properties of elements. An XML attribute is always a name-value pair.

Syntax

An XML attribute has the following syntax −

```
<element-name attribute1 attribute2 >
....content..
< /element-name>
```

where attribute1 and attribute2 has the following form −

```
name = "value"
```

value has to be in double (" ") or single (' ') quotes. Here,attribute1 and attribute2 are unique attribute labels.

Attributes are used to add a unique label to an element, place the label in a category, add a Boolean flag, or otherwise associate it with some string of data. Following example demonstrates the use of attributes −

```
<?xml version = "1.0" encoding = "UTF-8"?>
<!DOCTYPE garden [
  <!ELEMENT garden (plants)*>
  <!ELEMENT plants (#PCDATA)>
  <!ATTLIST plants category CDATA #REQUIRED>
]>

<garden>
  <plants category = "flowers" />
  <plants category = "shrubs">
  </plants>
</garden>
```

Attributes are used to distinguish among elements of the same name, when you do not want to create a new element for every situation. Hence, the use of an attribute can add a little more detail in differentiating two or more similar elements.

In the above example, we have categorized the plants by including attribute category and assigning different values to each of the elements. Hence, we have two categories of plants, one flowers and other color. Thus, we have two plant elements with different attributes.

You can also observe that we have declared this attribute at the beginning of XML.

Attribute Types

Following table lists the type of attributes −

Attribute Type	Description
StringType	It takes any literal string as a value. CDATA is a StringType. CDATA is character data. This means, any string of non-markup characters is a legal part of the attribute.
TokenizedType	This is a more constrained type. The validity constraints noted in the grammar are applied after the attribute value is normalized. The TokenizedType attributes are given as − • **ID** − It is used to specify the element as unique. • **IDREF** − It is used to reference an ID that has been named for another element. • **IDREFS** − It is used to reference all IDs of an element. • **ENTITY** − It indicates that the attribute will represent an external entity in the document. • **ENTITIES** − It indicates that the attribute will represent external entities in the document.

	• **NMTOKEN** − It is similar to CDATA with restrictions on what data can be part of the attribute. • **NMTOKENS** − It is similar to CDATA with restrictions on what data can be part of the attribute.
EnumeratedType	This has a list of predefined values in its declaration. out of which, it must assign one value. There are two types of enumerated attribute − • **NotationType** − It declares that an element will be referenced to a NOTATION declared somewhere else in the XML document. • **Enumeration** − Enumeration allows you to define a specific list of values that the attribute value must match.

Element Attribute Rules

Following are the rules that need to be followed for attributes −

- An attribute name must not appear more than once in the same start-tag or empty-element tag.
- An attribute must be declared in the Document Type Definition (DTD) using an Attribute-List Declaration.
- Attribute values must not contain direct or indirect entity references to external entities.
- The replacement text of any entity referred to directly or indirectly in an attribute value must not contain a less than sign (<)

XML - Comments

This chapter explains how comments work in XML documents.**XML comments** are similar to HTML comments. The comments are added as notes or lines for understanding the purpose of an XML code.

Comments can be used to include related links, information, and terms. They are visible only in the source code; not in the XML code. Comments may appear anywhere in XML code.

Syntax

XML comment has the following syntax −

```
<!--Your comment-->
```

A comment starts with **<!--** and ends with **-->**. You can add textual notes as comments between the characters. You must not nest one comment inside the other.

Example

Following example demonstrates the use of comments in XML document −

```
<?xml version = "1.0" encoding = "UTF-8" ?>
<!--Students grades are uploaded by months-->
<class_list>
  <student>
    <name>Tanmay</name>
    <grade>A</grade>
  </student>
</class_list>
```

Any text between **<!--** and **-->** characters is considered as a comment.

XML Comments Rules

Following rules should be followed for XML comments −

- Comments cannot appear before XML declaration.

- Comments may appear anywhere in a document.
- Comments must not appear within attribute values.
- Comments cannot be nested inside the other comments.

XML - Character Entities

This chapter describes the XML **Character Entities**. Before we understand the Character Entities, let us first understand what an XML entity is.

As put by W3 Consortium the definition of an entity is as follows −

"The document entity serves as the root of the entity tree and a starting-point for an XML processor".

This means, entities are the placeholders in XML. These can be declared in the document prolog or in a DTD. There are different types of entities and in this chapter we will discuss Character Entity.

Both, HTML and XML, have some symbols reserved for their use, which cannot be used as content in XML code. For example, <and > signs are used for opening and closing XML tags. To display these special characters, the character entities are used.

There are few special characters or symbols which are not available to be typed directly from the keyboard. Character Entities can also be used to display those symbols/special characters.

Types of Character Entities

There are three types of character entities −

- Predefined Character Entities
- Numbered Character Entities
- Named Character Entities

Predefined Character Entities

They are introduced to avoid the ambiguity while using some symbols. For example, an ambiguity is observed when less than (<) or greater than (>) symbol is used with the angle tag (<>). Character entities are basically used to delimit tags in XML. Following is a list of pre-defined character entities from XML specification. These can be used to express characters without ambiguity.

- Ampersand − **&**
- Single quote − **'**
- Greater than − **>**
- Less than − **<**
- Double quote − **"**

Numeric Character Entities

The numeric reference is used to refer to a character entity. Numeric reference can either be in decimal or hexadecimal format. As there are thousands of numeric references available, these are a bit hard to remember. Numeric reference refers to the character by its number in the **Unicode character set.**

General syntax for decimal numeric reference is −

```
&# decimal number ;
```

General syntax for hexadecimal numeric reference is −

```
&#x Hexadecimal number ;
```

The following table lists some predefined character entities with their numeric values −

Entity name	Character	Decimal reference	Hexadecimal reference

Entity name	Character	Decimal reference	Hexadecimal reference
quot	"	&#34;	&#x22;
amp	&	&#38;	&#x26;
apos	'	&#39;	&#x27;
lt	<	&#60;	&#x3C;
gt	>	&#62;	&#x3E;

Named Character Entity

As it is hard to remember the numeric characters, the most preferred type of character entity is the named character entity. Here, each entity is identified with a name.

For example −

- 'Aacute' represents capital character with acute accent.
- 'ugrave' represents the small with grave accent.

XML - CDATA Sections

In this chapter, we will discuss **XML CDATA section**. The term CDATA means, Character Data. CDATA is defined as blocks of text that are not parsed by the parser, but are otherwise recognized as markup.

The predefined entities such as **<, >,** and **&** require typing and are generally difficult to read in the markup. In such cases, CDATA section can be used. By using CDATA section, you are commanding the parser that the particular section of the document contains no markup and should be treated as regular text.

Syntax

Following is the syntax for CDATA section −

```
<![CDATA[
  characters with markup
]]>
```

The above syntax is composed of three sections −

- **CDATA Start section** − CDATA begins with the nine-character delimiter **<![CDATA[**
- **CDATA End section** − CDATA section ends with **]]>**delimiter.
- **CData section** − Characters between these two enclosures are interpreted as characters, and not as markup. This section may contain markup characters (<, >, and &), but they are ignored by the XML processor.

Example

The following markup code shows an example of CDATA. Here, each character written inside the CDATA section is ignored by the parser.

```
<script>
  <![CDATA[
    <message> Welcome to TutorialsPoint </message>
  ]] >
```

```
</script >
```

In the above syntax, everything between <message> and </message> is treated as character data and not as markup.

CDATA Rules

The given rules are required to be followed for XML CDATA −

- CDATA cannot contain the string "]]>" anywhere in the XML document.
- Nesting is not allowed in CDATA section.

XML – White Spaces

In this chapter, we will discuss **whitespace** handling in XML documents. Whitespace is a collection of spaces, tabs, and newlines. They are generally used to make a document more readable.

XML document contains two types of whitespaces - Significant Whitespace and Insignificant Whitespace. Both are explained below with examples.

Significant Whitespace

A significant Whitespace occurs within the element which contains text and markup present together. For example −

```
<name>TanmayPatil</name>
```

and

```
<name>Tanmay Patil</name>
```

The above two elements are different because of the space between **Tanmay** and **Patil**. Any program reading this element in an XML file is obliged to maintain the distinction.

Insignificant Whitespace

Insignificant whitespace means the space where only element content is allowed. For example −

```
<address.category = "residence">
```

or

```
<address....category = "..residence">
```

The above examples are same. Here, the space is represented by dots (.). In the above example, the space between address andcategory is insignificant.

A special attribute named **xml:space** may be attached to an element. This indicates that whitespace should not be removed for that element by the application. You can set this attribute to**default** or **preserve** as shown in the following example −

```
<!ATTLIST address  xml:space (default|preserve) 'preserve'>
```

Where,

- The value **default** signals that the default whitespace processing modes of an application are acceptable for this element.
- The value **preserve** indicates the application to preserve all the whitespaces.

XML - Processing

This chapter describes the **Processing Instructions (PIs)**. As defined by the XML 1.0 Recommendation,

"Processing instructions (PIs) allow documents to contain instructions for applications. PIs are not part of the character data of the document, but MUST be passed through to the application.

Processing instructions (PIs) can be used to pass information to applications. PIs can appear anywhere in the document outside the markup. They can appear in the prolog, including the document type definition (DTD), in textual content, or after the document.

Syntax

Following is the syntax of PI −

```
<?target instructions?>
```

Where

- **target** − Identifies the application to which the instruction is directed.
- **instruction** − A character that describes the information for the application to process.

A PI starts with a special tag **<?** and ends with **?>**. Processing of the contents ends immediately after the string **?>** is encountered.

Example

PIs are rarely used. They are mostly used to link XML document to a style sheet. Following is an example −

```
<?xml-stylesheet href = "tutorialspointstyle.css" type = "text/css"?>
```

Here, the target is xml-stylesheet. href="tutorialspointstyle.css"and type="text/css" are data or instructions the target application will use at the time of processing the given XML document.

In this case, a browser recognizes the target by indicating that the XML should be transformed before being shown; the first attribute states that the type of the transform is XSL and the second attribute points to its location.

Processing Instructions Rules

A PI can contain any data except the combination **?>**, which is interpreted as the closing delimiter. Here are two examples of valid PIs −

```
<?welcome to pg = 10 of tutorials point?>

<?welcome?>
```

XML - Encoding

Encoding is the process of converting unicode characters into their equivalent binary representation. When the XML processor reads an XML document, it encodes the document depending on the type of encoding. Hence, we need to specify the type of encoding in the XML declaration.

Encoding Types

There are mainly two types of encoding −

- UTF-8
- UTF-16

UTF stands for UCS Transformation Format, and UCS itself meansUniversal Character Set. The number 8 or 16 refers to the number of bits used to represent a character. They are either 8(one byte) or 16(two bytes). For the documents without encoding information, UTF-8 is set by default.

Syntax

Encoding type is included in the prolog section of the XML document. The syntax for UTF-8 encoding is as follows −

```
<?xml version = "1.0" encoding = "UTF-8" standalone = "no" ?>
```

The syntax for UTF-16 encoding is as follows −

```
<?xml version = "1.0" encoding = "UTF-16" standalone = "no" ?>
```

Example

Following example shows the declaration of encoding −

```
<?xml version = "1.0" encoding = "UTF-8" standalone = "no" ?>
<contact-info>
   <name>Tanmay Patil</name>
   <company>TutorialsPoint</company>
   <phone>(011) 123-4567</phone>
</contact-info>
```

In the above example **encoding="UTF-8"**, specifies that 8-bits are used to represent the characters. To represent 16-bit characters, **UTF-16** encoding can be used.

The XML files encoded with UTF-8 tend to be smaller in size than those encoded with UTF-16 format.

XML - Validation

Validation is a process by which an XML document is validated. An XML document is said to be valid if its contents match with the elements, attributes and associated document type declaration(DTD), and if the document complies with the constraints expressed in it. Validation is dealt in two ways by the XML parser. They are −

- Well-formed XML document
- Valid XML document

Well-formed XML Document

An XML document is said to be **well-formed** if it adheres to the following rules −

- Non DTD XML files must use the predefined character entities for **amp(&)**, **apos(single quote)**, **gt(>)**, **lt(<)**,**quot(double quote)**.
- It must follow the ordering of the tag. i.e., the inner tag must be closed before closing the outer tag.
- Each of its opening tags must have a closing tag or it must be a self ending tag.(<title>....</title> or <title/>).
- It must have only one attribute in a start tag, which needs to be quoted.
- **amp(&)**, **apos(single quote)**, **gt(>)**, **lt(<)**,**quot(double quote)** entities other than these must be declared.

Example

Following is an example of a well-formed XML document −

```
<?xml version = "1.0" encoding = "UTF-8" standalone = "yes" ?>
<!DOCTYPE address
[
   <!ELEMENT address (name,company,phone)>
   <!ELEMENT name (#PCDATA)>
   <!ELEMENT company (#PCDATA)>
   <!ELEMENT phone (#PCDATA)>
]>

<address>
   <name>Tanmay Patil</name>
```

```
  <company>TutorialsPoint</company>
  <phone>(011) 123-4567</phone>
</address>
```

The above example is said to be well-formed as −

- It defines the type of document. Here, the document type is **element** type.
- It includes a root element named as **address**.
- Each of the child elements among name, company and phone is enclosed in its self explanatory tag.
- Order of the tags is maintained.

Valid XML Document

If an XML document is well-formed and has an associated Document Type Declaration (DTD), then it is said to be a valid XML document.

https://www.tutorialride.com/xml/xml-document-type-definition.htm

Introduction to XML DTD

- DTD stands for Document Type Definition.
- It is used to validate the XML documents.
- XML provides facility to create your own DTDs for XML document.
- DTD specifies the structure of the XML document.
- DTD is part of the file or separate document file.

Types of DTD

There are two types of DTDs:

a) Internal DTD.

b) External DTD.

Internal DTD vs External DTD.

Internal DTD	**External DTD**
Internal DTD is part of the document.	External DTD is a separate file.
The scope is limited to the document which is created.	It is accessible across the multiple documents.
Syntax: <!DOCTYPE rootelement[element and attribute declarations]>	**Syntax:** <!DOCTYPE rootelement|SYSTEM "path of .dtd file">

Components of DTD

We have to associate the .xml file with the DTD and run the same .xml file in the browser.

Syntax for declaring elements in DTD:

<!ELEMENT elementname (content-type or content-model)>

Where elementname specifies the name of the element present in the xml document and content-type or content-model specifies whether the element contains textual data or other elements.

The three types of elements are as follows:

Element	**Description**	**DTD Declaration**

type		
Empty	Contains attributes, but can't contain text or any other element.	<!Element elementname EMPTY>
Unrestricted	Contains text content or any other element.	<!ELEMENT elementname ANY>
Container	Contain another elements.	<!ELEMENT elementname (elementname, elementname [elementname])>

Declaring Attributes in DTD

Syntax:

<!!ATTLIST elementname atributename valuetype [atributetype] [“default”]>

Each attribute declaration must include at least the attribute name in a DTD as follows:

Value Type	Description
PCDATA	Represents the plain text values.
ID	Assigns unique value to each element in the document.
(enumerated)	Assigns a specific range of values which is specified within parenthesis.

Example : Declaring Attributes in DTD

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
    <!DOCTYPE Employee
[
    <!ELEMENT address (name,company,phone)>
    <!ELEMENT name (#PCDATA)>
    <!ELEMENT company (#PCDATA)>
    <!ELEMENT phone (#PCDATA)>
]>
<address>
    <name>Mayuri S</name>
    <company>TutorialRide</company>
    <phone>91-9800000000</phone>
</address>
```

What is XML schema?

- XML schema defines the structure of an XML document.
- A schema defines, list of elements, attributes used in XML documents and data type of these elements.
- It is known as XSD.

XSD vs DTD

XSD	DTD
Defines list, order, data types of elements and attributes.	Defines list, order of elements and attributes.

Provides control over the elements and attributes used in XML documents.	It does not provide control over elements and attributes.
XSD allows to create customized datatype.	DTD does not allow to create customized datatype.
Syntax of XSD is similar to XML document.	Syntax of DTD is different from XML document.
XSD allows to define restrictions on data. **For example:** Define the content in a document by using only integer data type.	DTD does not allows to define restrictions on data.

Declaring Elements in XSD

Syntax:

<xsd:element name =“elementname” type=“datatype” minOccurs = “notNegativeInteger” maxOccurs=“nonNegativeInteger | unbounded”/>

Where,

name: Defines the element name

type: Defines data type.

minOccurs: If its value is zero, the use of element is optional and if its value is greater than zero, the use of element is compulsory, and should occur at least for specified number of times.

maxOccurs: If value is set as unbounded, the use of element can appear any number of times in the XML document without any limitation.

Example : XSD for simple type elements using predefined datatypes

```
<xsd:element name=“CARNAME” TYPE=“xsd:string”/>
<xsd:element name=“COMPANYNAME” TYPE=“xsd:string”/>
<xsd:element name=“PRICE” TYPE=“xsd:positiveInteger”/>
```

XML document with simple type elements and associated XSD using creation of custom data type.

```
<EMPLOYEENAME>Surabhi</EMPLOYEENAME>
<EMPLOYEEPHONE>9800000000</EMPLOYEEPHONE>
<xsd:element name=“EMPLOYEENAME” TYPE=“xsd:string”/>
<xsd:element name=“EMPLOYEEPHONE” TYPE=“xsd:string”/>
    <xsd:simpleType name=“phoneno”>
    <xsd:restriction base=“xsd:string”>
        <xsd:length value=“15”/>
        <xsd:pattern value=“\d{3}-\d{6}”/>
    </xsd:restriction>
    </xsd:simpleType>
```

In above example, the XSD defines the custom data type **“phoneno”.**

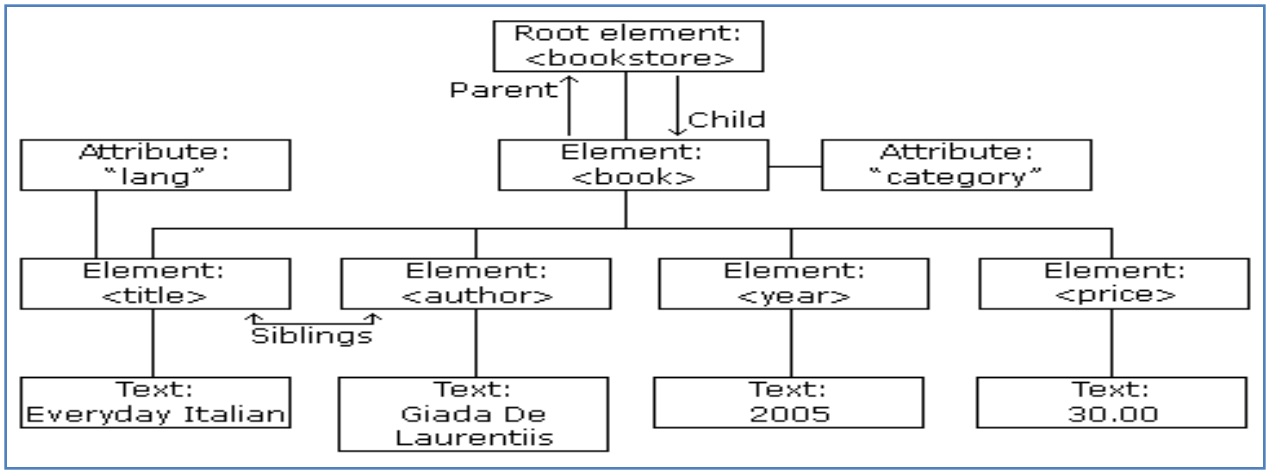
The data type “phoneno” has restrictions as, it can hold predefined data type string and it should be 12 characters long and \d presents 'digits'.

Inbuilt characters classes commonly used in XSD.

Class	Method	Meaning
[0-9]	\ d	Any digit.
[\ f\ r\ t \ n \ v]	\ s	Any white space.
[A-Za-z0-9]	\ w	Any word character.
[^0-9]	\ D	Not a digit.
[^ \ f \ r \ t \ n\ v]	\ S	Not a White space.
[^A-Za-z0-9]	\ W	Not a word character.

https://www.w3schools.com/xml/dom_intro.asp

XML DOM



What is the DOM?

The DOM defines a standard for accessing and manipulating documents:

"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."

The HTML DOM defines a standard way for accessing and manipulating HTML documents. It presents an HTML document as a tree-structure.

The XML DOM defines a standard way for accessing and manipulating XML documents. It presents an XML document as a tree-structure.

Understanding the DOM is a must for anyone working with HTML or XML.

The HTML DOM

All HTML elements can be accessed through the HTML DOM.

This example changes the value of an HTML element with id="demo":

```
<!DOCTYPE html>
<html>
```

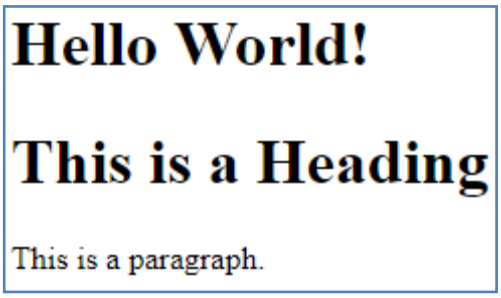
```
<body>

<h1 id="demo">This is a Heading</h1>

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

<script>
document.getElementById("demo").innerHTML = "Hello World!";
</script>

</body>
</html>
```

Output

This example changes the value of the first <h1> element in an HTML document:

```
<!DOCTYPE html>
<html>
<body>

<h1>This is a Heading</h1>

<h1>This is a Heading</h1>

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

<script>
document.getElementsByTagName("h1")[0].innerHTML = "Hello World!";
</script>

</body>
</html>
```

Output



Note: Even if the HTML document contains only ONE <h1> element you still have to specify the array index [0], because the getElementsByTagName() method always returns an array.

The XML DOM

All XML elements can be accessed through the XML DOM.

The XML DOM is:

- A standard object model for XML
- A standard programming interface for XML
- Platform- and language-independent
- A W3C standard

In other words: **The XML DOM is a standard for how to get, change, add, or delete XML elements.**

Get the Value of an XML Element

This code retrieves the text value of the first <title> element in an XML document:

```
txt = xmlDoc.getElementsByTagName("title")[0].childNodes[0].nodeValue;
```

Loading an XML File

The XML file used in the examples below is books.xml.

This example reads "books.xml" into xmlDoc and retrieves the text value of the first <title> element in books.xml:

```
<!DOCTYPE html>
<html>
<body>

<p id="demo"></p>

<script>
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
   if (this.readyState == 4 && this.status == 200) {
      myFunction(this);
   }
};
xhttp.open("GET", "books.xml", true);
xhttp.send();

function myFunction(xml) {
   var xmlDoc = xml.responseXML;
   document.getElementById("demo").innerHTML =
   xmlDoc.getElementsByTagName("title")[0].childNodes[0].nodeValue;
}
</script>

</body>
</html>
```

Output

Everyday Italian

Example Explained

- **xmlDoc** - the XML DOM object created by the parser.
- **getElementsByTagName("title")[0]** - get the first <title> element
- **childNodes[0]** - the first child of the <title> element (the text node)
- **nodeValue** - the value of the node (the text itself)

Loading an XML String

This example loads a text string into an XML DOM object, and extracts the info from it with JavaScript:

Programming Interface

The DOM models XML as a set of node objects. The nodes can be accessed with JavaScript or other programming languages. In this tutorial we use JavaScript.

The programming interface to the DOM is defined by a set standard properties and methods.

Properties are often referred to as something that is (i.e. nodename is "book").

Methods are often referred to as something that is done (i.e. delete "book").

XML DOM Properties

These are some typical DOM properties:

- x.nodeName - the name of x
- x.nodeValue - the value of x
- x.parentNode - the parent node of x
- x.childNodes - the child nodes of x
- x.attributes - the attributes nodes of x

Note: In the list above, x is a node object.

XML DOM Methods

- x.getElementsByTagName(name) - get all elements with a specified tag name
- x.appendChild(node) - insert a child node to x
- x.removeChild(node) - remove a child node from x

Note: In the list above, x is a node object.

http://presentingxml.sourceforge.net/

Presenting XML

The core **Presenting XML** mark-up language is currently being further developed in a companion project, Serving XML. Anyone interested primarily in the command line examples should use the **Serving XML** software instead.

Presenting XML is a Java web application framework for presenting HTML, PDF, WML etc. in a device independent manner. It aims to achieve a complete separation of content and presentation. It supports various kinds of content including XML files, dynamic content, SQL result sets and flat files. It provides a declarative way for applying filters and XSLT transforms to a stream of XML content in a pipeline. It allows user defined filters and serializers written as Java plug-ins. It is component based and extendible.

Presenting XML may be used as a command line tool or as a framework for a servlet-based web application. Simple examples with XML, XSLT, and Java are included in the distribution. There are command line examples for converting flat files to XML (and vice versa), for converting SQL results to XML, for applying a sequence of XSLT transforms and SAX filters in a pipeline, and for setting up streaming filters.

There is a sample web application that illustrates how to build a simple shopping cart.

The last **Presenting XML** file distribution may be downloaded from SourceForge. It includes Java source code and ant build files as well as binary jar files, a sample web application, and run scripts. It includes api documentation and a user guide. It also includes a number of third party jar files including Saxon (also Xalan), Apache Xerces, Sun's Multi-Schema XML Validator, and the Apache fop pdf driver. It includes everything for running the sample web application except the Java JDK and a web server.

Presenting XML requires Java JDK 1.3 or later (it will not work with earlier versions.) It also requires a web server. The examples are configured to work with Jakarta Tomcat 4.0, which may be downloaded and installed separately. The WML sample pages require a WAP browser. A free WAP browser emulator may be obtained from OPENWAVE.

https://www.tutorialride.com/xml/xml-parsers.htm

What is XML Parsers?

XML parsers are libraries used for checking and validating XML document schema.

DOM Parser

- DOM (Document Object Model) provides a programming interface for manipulating XML documents.
- DOM includes a set of objects and interfaces, which represent the content and structure of an XML document and enables a program to traverse XML tree structure.
- DOM allows to create new XML document with programming interfaces, for example, we can create XML document through ASP.net.

SAX Parser

- SAX stands for Simple API for XML.
- API (Application Programming interface) allows programmer to read and write XML data.
- SAX parser is based on events.
- SAX Parser follows a push model which allows sequential access.

SAX Parser vs DOM Parser.

SAX Parser	DOM Parser
Instead of creating internal structure in the document, it takes the occurrences of components as events.	DOM parser creates a tree structure in memory from an input document and waits for client request.
SAX Parser allows to extract the information in the document according to the users interest.	User can not extract the information of his interest.
SAX parser is space efficient.	DOM Parser is space inefficient.
Users have to create their own data structures.	DOM parser allows user to access any part of the document and modify the DOM tree.

When should you use SAX and DOM Parser?

Example:

Assume that University has an XML document in which all the information of MCA students

with their marks is there. It wants to assign final grades to the student and produce a list with Exam No and grades.

Case1: If University decides to give 'A' to those who earned class average and, 'B' to others.
Case2: If University decides to give A to those students who got 65% marks or more and B to others.

- We can use DOM Parser for Case1. For this, the university has to look out for the marks and calculate the average. They can then assign the grades by comparing the marks of all students.
- We can use SAX Parser in Case2. For this, there is no need to go through the whole document. A grade is assigned to a student once the SAX parser reads the marks of the students.

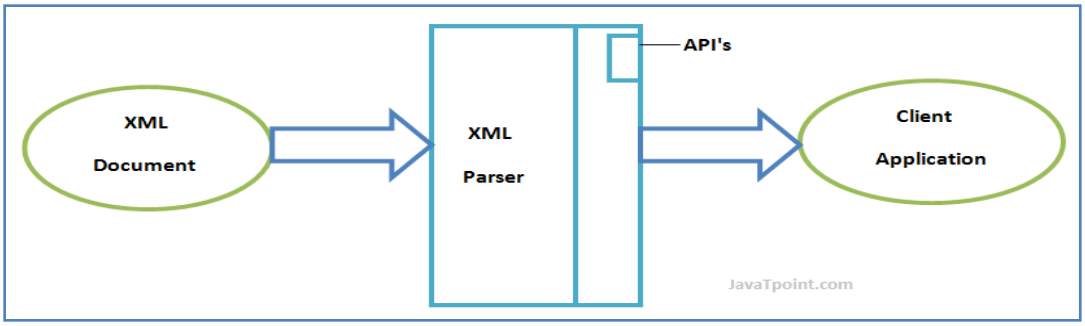
https://www.javatpoint.com/xml-parsers

XML Parsers

An XML parser is a software library or package that provides interfaces for client applications to work with an XML document. The XML Parser is designed to read the XML and create a way for programs to use XML.

XML parser validates the document and check that the document is well formatted.

Let's understand the working of XML parser by the figure given below:



Types of XML Parsers

These are the two main types of XML Parsers:

1. DOM
2. SAX

DOM (Document Object Model)

A DOM document is an object which contains all the information of an XML document. It is composed like a tree structure. The DOM Parser implements a DOM API. This API is very simple to use.

Features of DOM Parser

A DOM Parser creates an internal structure in memory which is a DOM document object and the client applications get information of the original XML document by invoking methods on this document object.

DOM Parser has a tree based structure.

Advantages

1) It supports both read and write operations and the API is very simple to use.
2) It is preferred when random access to widely separated parts of a document is required.

Disadvantages

1) It is memory inefficient. (consumes more memory because the whole XML document needs to loaded into memory).
2) It is comparatively slower than other parsers.

SAX (Simple API for XML)
A SAX Parser implements SAX API. This API is an event based API and less intuitive.

Features of SAX Parser

It does not create any internal structure.
Clients does not know what methods to call, they just overrides the methods of the API and place his own code inside method.
It is an event based parser, it works like an event handler in Java.

Advantages

1) It is simple and memory efficient.
2) It is very fast and works for huge documents.

Disadvantages

1) It is event-based so its API is less intuitive.
2) Clients never know the full information because the data is broken into pieces.

https://www.w3schools.com/xml/xml_validator.asp

Well Formed XML Documents

An XML document with correct syntax is called "Well Formed".
The syntax rules were described in the previous chapters:

- XML documents must have a root element
- XML elements must have a closing tag
- XML tags are case sensitive
- XML elements must be properly nested
- XML attribute values must be quoted

XML Errors Will Stop You

Errors in XML documents will stop your XML applications.
The W3C XML specification states that a program should stop processing an XML document if it finds an error. The reason is that XML software should be small, fast, and compatible.
HTML browsers are allowed to display HTML documents with errors (like missing end tags).
With XML, errors are not allowed.

https://www.w3schools.com/xml/xsl_intro.asp

XSL (eXtensible Stylesheet Language) is a styling language for XML.
XSLT stands for XSL Transformations.
This tutorial will teach you how to use XSLT to transform XML documents into other formats (like transforming XML into HTML).

XML Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
```

```
</cd>
<cd>
  <title>Hide your heart</title>
  <artist>Bonnie Tyler</artist>
  <country>UK</country>
  <company>CBS Records</company>
  <price>9.90</price>
  <year>1988</year>
</cd>
<cd>
  <title>Greatest Hits</title>
  <artist>Dolly Parton</artist>
  <country>USA</country>
  <company>RCA</company>
  <price>9.90</price>
  <year>1982</year>
</cd>
<cd>
  <title>Still got the blues</title>
  <artist>Gary Moore</artist>
  <country>UK</country>
  <company>Virgin records</company>
  <price>10.20</price>
  <year>1990</year>
</cd>
<cd>
  <title>Eros</title>
  <artist>Eros Ramazzotti</artist>
  <country>EU</country>
  <company>BMG</company>
  <price>9.90</price>
  <year>1997</year>
</cd>
<cd>
  <title>One night only</title>
  <artist>Bee Gees</artist>
  <country>UK</country>
  <company>Polydor</company>
  <price>10.90</price>
  <year>1998</year>
</cd>
<cd>
  <title>Sylvias Mother</title>
  <artist>Dr.Hook</artist>
  <country>UK</country>
  <company>CBS</company>
```

```
  <price>8.10</price>
  <year>1973</year>
</cd>
<cd>
  <title>Maggie May</title>
  <artist>Rod Stewart</artist>
  <country>UK</country>
  <company>Pickwick</company>
  <price>8.50</price>
  <year>1990</year>
</cd>
<cd>
  <title>Romanza</title>
  <artist>Andrea Bocelli</artist>
  <country>EU</country>
  <company>Polydor</company>
  <price>10.80</price>
  <year>1996</year>
</cd>
<cd>
  <title>When a man loves a woman</title>
  <artist>Percy Sledge</artist>
  <country>USA</country>
  <company>Atlantic</company>
  <price>8.70</price>
  <year>1987</year>
</cd>
<cd>
  <title>Black angel</title>
  <artist>Savage Rose</artist>
  <country>EU</country>
  <company>Mega</company>
  <price>10.90</price>
  <year>1995</year>
</cd>
<cd>
  <title>1999 Grammy Nominees</title>
  <artist>Many</artist>
  <country>USA</country>
  <company>Grammy</company>
  <price>10.20</price>
  <year>1999</year>
</cd>
<cd>
  <title>For the good times</title>
  <artist>Kenny Rogers</artist>
```

```
  <country>UK</country>
  <company>Mucik Master</company>
  <price>8.70</price>
  <year>1995</year>
</cd>
<cd>
  <title>Big Willie style</title>
  <artist>Will Smith</artist>
  <country>USA</country>
  <company>Columbia</company>
  <price>9.90</price>
  <year>1997</year>
</cd>
<cd>
  <title>Tupelo Honey</title>
  <artist>Van Morrison</artist>
  <country>UK</country>
  <company>Polydor</company>
  <price>8.20</price>
  <year>1971</year>
</cd>
<cd>
  <title>Soulsville</title>
  <artist>Jorn Hoel</artist>
  <country>Norway</country>
  <company>WEA</company>
  <price>7.90</price>
  <year>1996</year>
</cd>
<cd>
  <title>The very best of</title>
  <artist>Cat Stevens</artist>
  <country>UK</country>
  <company>Island</company>
  <price>8.90</price>
  <year>1990</year>
</cd>
<cd>
  <title>Stop</title>
  <artist>Sam Brown</artist>
  <country>UK</country>
  <company>A and M</company>
  <price>8.90</price>
  <year>1988</year>
</cd>
<cd>
```

```
  <title>Bridge of Spies</title>
  <artist>T`Pau</artist>
  <country>UK</country>
  <company>Siren</company>
  <price>7.90</price>
  <year>1987</year>
</cd>
<cd>
  <title>Private Dancer</title>
  <artist>Tina Turner</artist>
  <country>UK</country>
  <company>Capitol</company>
  <price>8.90</price>
  <year>1983</year>
</cd>
<cd>
  <title>Midt om natten</title>
  <artist>Kim Larsen</artist>
  <country>EU</country>
  <company>Medley</company>
  <price>7.80</price>
  <year>1983</year>
</cd>
<cd>
  <title>Pavarotti Gala Concert</title>
  <artist>Luciano Pavarotti</artist>
  <country>UK</country>
  <company>DECCA</company>
  <price>9.90</price>
  <year>1991</year>
</cd>
<cd>
  <title>The dock of the bay</title>
  <artist>Otis Redding</artist>
  <country>USA</country>
  <company>Stax Records</company>
  <price>7.90</price>
  <year>1968</year>
</cd>
<cd>
  <title>Picture book</title>
  <artist>Simply Red</artist>
  <country>EU</country>
  <company>Elektra</company>
  <price>7.20</price>
  <year>1985</year>
```

```
  </cd>
  <cd>
    <title>Red</title>
    <artist>The Communards</artist>
    <country>UK</country>
    <company>London</company>
    <price>7.80</price>
    <year>1987</year>
  </cd>
  <cd>
    <title>Unchain my heart</title>
    <artist>Joe Cocker</artist>
    <country>USA</country>
    <company>EMI</company>
    <price>8.20</price>
    <year>1987</year>
  </cd>
</catalog>
```

XSLT Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
  <h2>My CD Collection</h2>
  <table border="1">
    <tr bgcolor="#9acd32">
      <th style="text-align:left">Title</th>
      <th style="text-align:left">Artist</th>
    </tr>
    <xsl:for-each select="catalog/cd">
    <tr>
      <td><xsl:value-of select="title"/></td>
      <td><xsl:value-of select="artist"/></td>
    </tr>
    </xsl:for-each>
  </table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Output

My CD Collection

Title	Artist
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler
Greatest Hits	Dolly Parton
Still got the blues	Gary Moore
Eros	Eros Ramazzotti
One night only	Bee Gees
Sylvias Mother	Dr.Hook
Maggie May	Rod Stewart
Romanza	Andrea Bocelli
When a man loves a woman	Percy Sledge
Black angel	Savage Rose
1999 Grammy Nominees	Many
For the good times	Kenny Rogers
Big Willie style	Will Smith
Tupelo Honey	Van Morrison
Soulsville	Jorn Hoel
The very best of	Cat Stevens
Stop	Sam Brown
Bridge of Spies	T`Pau
Private Dancer	Tina Turner
Midt om natten	Kim Larsen
Pavarotti Gala Concert	Luciano Pavarotti
The dock of the bay	Otis Redding
Picture book	Simply Red
Red	The Communards
Unchain my heart	Joe Cocker

https://www.w3.org/2000/Talks/www9-xsl/

XSL: Transformations and Style

Vincent Quint

World Wide Web Consortium

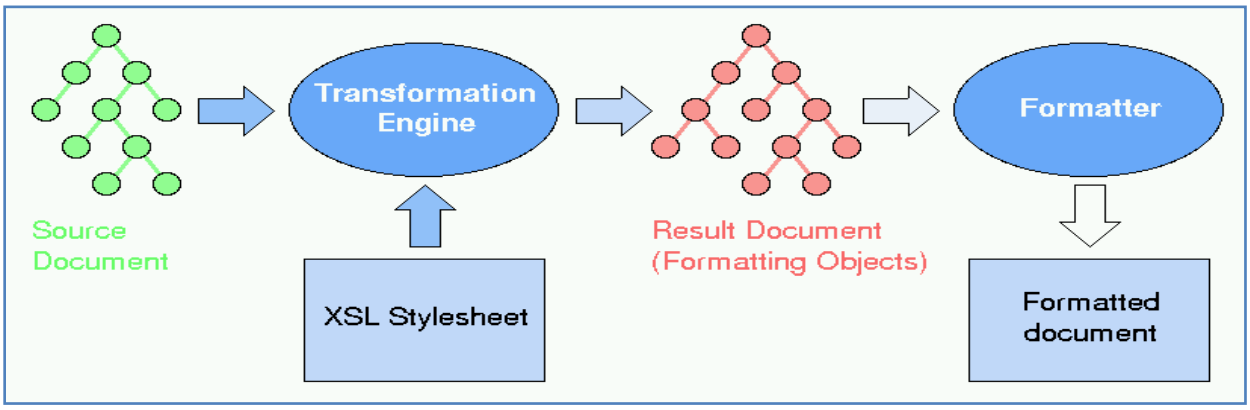
WWW9 - Amsterdam - 18 May 2000

Extensible Stylesheet Language (XSL)

XSL is a language for expressing stylesheets

- support for browsing, printing, and aural rendering
- formatting highly structured documents (XML)
- performing complex publishing tasks: tables of contents, indexes, reports,...
- addressing accessibility and internationalization issues
- written in XML

XSL Architecture



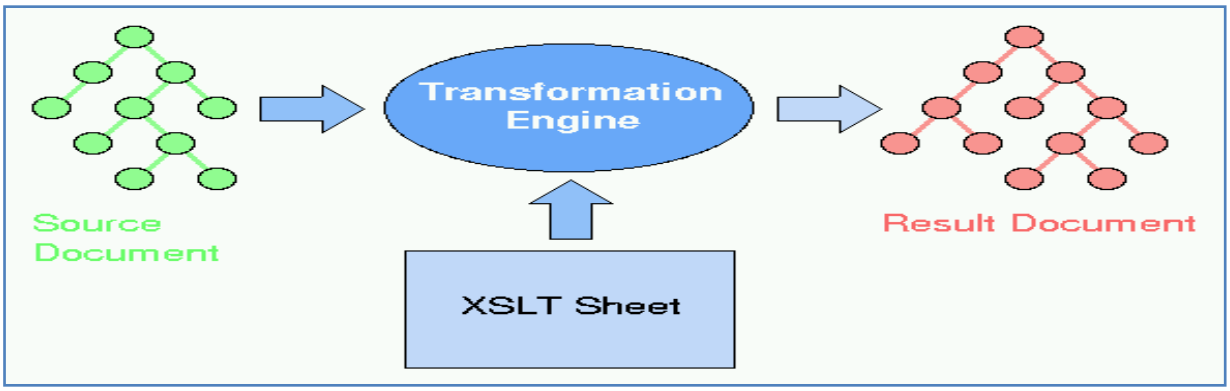
XSL Components

XSL is constituted of three main components:

- XSLT: a transformation language
- XPath: an expression language for addressing parts of XML documents
- FO: a vocabulary of formatting objects with their associated formatting properties

XSL uses XSLT which uses XPath

XSL Transformations



XSLT - Basic Principle

Patterns and **Templates**

- A style sheets describes transformation rules
- A transformation rule: a pattern + a template
- Pattern: a configuration in the source tree
- Template: a structure to be instantiated in the result tree
- When a pattern is matched in the source tree, the corresponding pattern is generated in the result tree

An Example: Transformation

```
<xsl:template match="Title">
  <H1>
    <xsl:apply-templates/>
  </H1>
</xsl:template>
```

Input : <Title>Introduction</Title>

Output : <H1>Introduction</H1>

An Example: Formatting

```
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:fo="http://www.w3.org/1999/XSL/Format"
    result-ns="fo">
 <xsl:template match="/">
  <fo:page-sequence font-family="serif">
    <xsl:apply-templates/>
  </fo:page-sequence>
 </xsl:template>
```

```
  <xsl:template match="para">
    <fo:block font-size="10pt" space-before="12pt">
      <xsl:apply-templates/>
    </fo:block>
  </xsl:template>
</xsl:stylesheet>
```

XPath: XML Path Language

An elementary XPath expression contains

- an axis, which specifies the tree relationship: child, descendants, ancestors, siblings, attributes,...
- a node test, which specifies the node type
- predicates, to further refine the set of nodes selected

Example: all para children that have a type attribute with value warning

```
child::para[attribute::type="warning"]
```

XSL Usage

- Format XML documents by generating FOs
- Generate HTML or XHTML pages from XML data/documents
- Transform XML documents into other XML documents
- Generate some textual representation of an XML document
- ...and more

XSL may be used server-side or client-side,but is not intended to send FOs over the wire

Implementations

XSL software include:

- XSLT transformation engines: 4XSLT, IE5, iXSLT, LotusXSL, Transformiix, Resin, Sablotron, Saxon, Xalan, XML Parser (Oracle), XT
- FO formatters: FOP, FO2PDF, InDelv browser, Passive TeX, REXP
- XSL stylesheet editors
- Style sheets and transformation sheets

XSL and CSS

XSL is not intended to replace CSS, but provides functionality beyond that of CSS

XSL/CSS Common features:

- Formatting model
- Properties and values

History

XSL is inspired by both CSS and DSSSL (ISO/IEC 10179:1996)

- **Sep 1997**: "A Proposal for XSL" submitted to W3C by Inso, Microsoft, ArborText, U. of Edinburgh
- **Jan 1998**: The W3C Working Group for XSL is formed
- **Feb 1998**: The XSL list starts up
- **Aug 1998**: The first XSL working draft is published
- **Apr 1999**: The third XSL working draft is released. A separate document specifies the transformation language (XSLT)
- **Jul 1999**: A new working draft of XSLT is released. The expression language is specified in a separate working draft, XPath (with XML Linking WG)
- **Aug 1999**: XSLT, XPath enter Last Call
- **Oct 1999**: XSLT, XPath move to Proposed Recommendation status

- **Nov 1999**: XSLT, XPath become W3C Recommendations.
- **Jan 2000**: New XSL (FO) working draft published
- **Mar 2000**: XSL enters Last Call

Current Situation

- XPath 1.0 is a W3C Recommendation
- XSLT 1.0 is a W3C Recommendation
- XSL (FO) is in Last Call

What the Future Holds

Next step: move XSL 1.0 to W3C Recommendation status

New WG proposed to work on new versions of all 3 specifications: XSL, XSLT, XPath

- **FO**s: additional support of internationalized formatting objects, general regions
- **XSLT**, **XPath**: expanded extension mechanism, requirements from appendix G of XSLT version 1.0

XPath is used in XPointer and XLink, considered by XML Query WG

More information

W3C XSL page: http://www.w3.org/Style/XSL

XSL, XSLT, XPath specifications: http://www.w3.org/TR

Public mailing list: http://www.mulberrytech.com/xsl/xsl-list

https://www.w3schools.com/xml/xsl_languages.asp

XSL(T) Languages

XSLT is a language for transforming XML documents.

XPath is a language for navigating in XML documents.

XQuery is a language for querying XML documents.

It Started with XSL

XSL stands for E**X**tensible **S**tylesheet **L**anguage.

The World Wide Web Consortium (W3C) started to develop XSL because there was a need for an XML-based Stylesheet Language.

CSS = Style Sheets for HTML

HTML uses predefined tags. The meaning of, and how to display each tag is well understood.

CSS is used to add styles to HTML elements.

XSL = Style Sheets for XML

XML does not use predefined tags, and therefore the meaning of each tag is not well understood.

A <table> element could indicate an HTML table, a piece of furniture, or something else - and browsers do not know how to display it!

So, XSL describes how the XML elements should be displayed.

XSL - More Than a Style Sheet Language

XSL consists of four parts:

- XSLT - a language for transforming XML documents
- XPath - a language for navigating in XML documents
- XSL-FO - a language for formatting XML documents (discontinued in 2013)
- XQuery - a language for querying XML documents

What is XSLT?

- XSLT stands for XSL Transformations
- XSLT is the most important part of XSL
- XSLT transforms an XML document into another XML document
- XSLT uses XPath to navigate in XML documents
- XSLT is a W3C Recommendation

XSLT = XSL Transformations
XSLT is the most important part of XSL.
XSLT is used to transform an XML document into another XML document, or another type of document that is recognized by a browser, like HTML and XHTML. Normally XSLT does this by transforming each XML element into an (X)HTML element.
With XSLT you can add/remove elements and attributes to or from the output file. You can also rearrange and sort elements, perform tests and make decisions about which elements to hide and display, and a lot more.
A common way to describe the transformation process is to say that **XSLT transforms an XML source-tree into an XML result-tree**.

XSLT Uses XPath
XSLT uses XPath to find information in an XML document. XPath is used to navigate through elements and attributes in XML documents.
If you want to study XPath first, please read our XPath Tutorial.

How Does it Work?
In the transformation process, XSLT uses XPath to define parts of the source document that should match one or more predefined templates. When a match is found, XSLT will transform the matching part of the source document into the result document.

XSLT Browser Support
All major browsers support XSLT and XPath.

https://www.w3schools.com/xml/xsl_transformation.asp
XSLT - Transformation
Correct Style Sheet Declaration
The root element that declares the document to be an XSL style sheet is <xsl:stylesheet> or <xsl:transform>.
Note: <xsl:stylesheet> and <xsl:transform> are completely synonymous and either can be used!
The correct way to declare an XSL style sheet according to the W3C XSLT Recommendation is:

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
(Or)
<xsl:transform version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

To get access to the XSLT elements, attributes and features we must declare the XSLT namespace at the top of the document.
The xmlns:xsl="http://www.w3.org/1999/XSL/Transform" points to the official W3C XSLT namespace. If you use this namespace, you must also include the attribute version="1.0".

Start with a Raw XML Document

We want to **transform** the following XML document ("cdcatalog.xml") into XHTML:

```
<?xml version="1.0" encoding="UTF-8"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
.
.
</catalog>
```

Viewing XML Files in IE, Chrome, Firefox, Safari, and Opera: Open the XML file (click on the link below) - The XML document will be displayed with color-coded root and child elements (except in Safari). Often, there is a plus (+) or minus sign (-) to the left of the elements that can be clicked to expand or collapse the element structure. **Tip: To view the raw XML source, right-click in XML file and select "View Source"!**

http://fplanque.net/Blog/itTrends/2004/01/10/rss_rdf_and_atom_in_a_nutshell

Syndication, RSS, RDF and Atom in a Nutshell

Once upon a time, there was a company called Netscape who was investigating a new market: portal sites and content syndication. The idea was simple: a variety of websites produce relevant content in a (nearly) continuous flow. Portals would be designed to aggregate news and content from those sites and present it to the user all in one page…

Thus, Netscape invented a format called RSS, which stood for "Remote Site Syndication". This spec allows content producers to publish their news/content in an "RSS feed" (an XML based document) and content consumers to periodically check those feeds for updates.

When Netscape lost interest in developing portals, they abandonned their original (and complex) RSS 0.9 spec as well as their efforts in creating a more appropriate and simpler version. At that time, UserLand picked up the simpler 0.91 spec and applied it to its blog tools. RSS had become "Really Simple Syndication".

Today, the blogging community still uses RSS extensively: every (serious) blogger publishes an RSS feed of his posts and readers aggregate all their favorites blogs' feeds in an aggregator. This, of course, makes it more convenient to check for new posts daily on all your favorite blogs…

b2evolution is an example of a blog tool used to publish RSS feeds. SharpReader is an example of a program used to aggregate RSS feeds.

Then, the story gets more and more complicated…

UserLand started improving on RSS 0.91 which actually was a little bit too simple. That led to RSS 0.92, 0.93, 0.94…

But another group had started working on a successor to RSS 0.9 when Netscape dropped it. This group was more concerned with completeness than with simplicity. The result of their work has been released as RSS 1.0. This format is based on RDF and is so complex that UserLand decided

to go on with its own branch and finally released RSS 2.0… which is a successor to RSS 0.94, not RSS 1.0 !
This is why we have so many different RSS formats in use today. :-/
In an effort to make everybody happy and focus on a single syndication format, a new group has been created to work on a new format called Atom. This new format, currently at version 0.3, is full of promises… but it is too early to tell if it will finally be the one and only syndication spec, accepted by all… Let's hope so. ;)
2007 Update
Some time has passed. RSS 2.0 now seems to be the de facto standard used for blogs, forums, news sites, podcasts and event torrent feeds! (Atom is probably technically superior but it seems that history likes to repeat the VHS vs. Betamax scenario ;))
The most popular RSS readers now seem to be online services such as Google Reader, Bloglines or even Netvibes.
It has become very easy to (freely) subscribe to RSS feeds, as all you need to do now is to click

on the RSS icon in the address bar of Firefox 2 or Internet Explorer 7. After that, you just need to choose your favorite reader — the most popular ones being pre-configured.
If you want to generate RSS feeds for people to subscribe to, you can do so with a variety of tools like b2evolution. It is blog/news/podcast software that you install on your own hosting account (hosting can be shared web hosting, a VPS, a dedicated server or even just a cheap hosting plan depending on how many users you will have.)
Blog software like b2evolution lets you type posts which get published to your site automatically and concurrently generates a matching RSS feed of all new posts.

http://www.webreference.fr/defintions/rss-atom-xml

What is RSS? (and Atom?)

RSS stands for both Rich Site Summary and Really Simple Syndication but it always refers to the same technology.

It is a mean of transmitting and updating news in an automated way.

Most news sites (including virtually all blogs) will publish what is called an RSS feed which is regularly updated with the latest available headlines and/or articles.
The RSS feed is not human readable. It is an XML format which is designed to be read by machines rather than humans.

Reading RSS

To take advantage of an RSS feed you would use a piece of software called an RSS aggregator. Most of them are very similar to email client programs, but instead of incoming emails, they display news from various sources (from all the feeds you have registered with, or "subscribed to" as is commonly said but it has nothing to do with money). Unread news typically appear in bold, just as unread emails do.
An RSS aggregator makes it very convenient to follow up on news from a large number of sources in a single place. SharpReader is an example of an RSS Reader.
Just like there is webmail, there are also are web-RSS-aggregators. Bloglines is such an online aggregator. Il allows you to track all your news from a single place you can access with a regular web browser.
Also, most modern web browsers will also handle RSS feeds, but in a limited manner. They will use an RSS feed as a dynamic bookmark folder with automatic bookmarks to all the news in the feed. Unlike aggregators, browsers will not save the news if you don't check on them every day.

Finally, on a more professional level, some websites will aggregate news from different sources onto a single site. Hence the "syndication" in the name.

Producing RSS

While you could theoretically write an RSS file by hand and update it regularly, writing XML manually is a tedious task.

Most RSS feeds are produced automatically by the same content management software which also generates the web pages dynamically. All blog tools for example can generate RSS feeds on the fly. b2evolution is an example of such a blog tool.

Distributing RSS

Whether you generate your RSS by hand or have a tool generate it automatically, you need to make it available on the web for users (and their "client" software) to consume.

Just like with regular web content, you need to host it on a website with a web hosting provider. Note that until you get hundreds of thousands of users consuming your feeds every day, cheap web hosting is perfectly suited for distributing RSS/Atom feeds.

Different flavours

There are different versions of RSS in use. RSS 2.0 is the most common. It is used for news/blog feeds as well as for Podcasting.

A newer format, called Atom, is a more standardized way of providing XML content updates. However, it has not gotten wide acceptance yet outside of the blog communities. (Again, almost all blog tools can generate an Atom feed on the fly.)

https://www.practicalecommerce.com/How-Is-An-Atom-Feed-Different-From-An-RSS-Feed

How Is An Atom Feed Different From An RSS Feed?

An Atom feed is very similar to an RSS feed in that it is a lightweight XML format allowing for easy syndication of web content. In fact, most RSS readers and news aggregators will be able to read Atom feeds just fine, as it is becoming a widely-used alternative to RSS feeds.

What's a "feed"?

For those who are unfamiliar with syndication feeds (such as an Atom feed, an RSS feed or an RDF feed), they are small text files that provide information about content on websites. When content is updated, the feed text file is also updated, either manually or programatically. Applications called "readers" or "aggregators" can then check these small text files and notify someone when new content is available.

If you already have an RSS feed, creating an Atom feed is extremely easy. The primary difference between the two formats (from a developer's standpoint) is the XML tags that are used. You can refer to Atomenabled.org for more information about the Atom specification and for information about required tags as well as available tags.

https://www.saksoft.com/rss-vs-atom/

RSS vs Atom

RSS (Really Simple Syndication or Rich Site Summary) is a familiar name in the internet world, but Atom (Atom Syndication Format) may not ring a bell for many.

Content Syndication is a mechanism using which the contents of a website including articles, news, blogs and forums are published partially or fully to other websites in a specific format. RSS and Atom are the two main standards of web syndication.

Atom was developed to avoid the limitations and flaws of RSS. Though Atom is more robust than RSS, the latter still remains a widely used standard. Just as how Xerox has become

synonymous to photocopying, RSS has generally become a common term to refer to all feeds including Atom.
Some websites provide feeds in both these formats, while the other in one of these formats. For examples, IBM developerWorks website has both Atom and RSS feeds, Google supports only Atom and CNN has only RSS.
Some differences between RSS and Atom are listed below.

Criteria	**RSS**	**Atom**
File extension	.rss or .xml	.atom or .xml
Media type identification	application/rss+xml	application/atom+xml
Extension to namespaces	Not Supported	Supported
Payload content support	Plain text and escaped HTML	Plain text, escaped HTML, XHTML, XML, Base64-encoded binary and references to external binary content
Content type identification	Not available; cannot identify if the content is plain text or escaped HTML	Explicit identification of content is supported
Date formats	Feed creation or last updated date	Website last updated date
Internationalization	Language context at the feed level	Language context at every individual element level
Modularity	Vocabularies not usable in other XML vocabularies	Vocabularies reusable outside the context of the feed
Robustness	Easy	Rigid
Ease of aggregation	Complex	Easy
Popularity trend	Steady	Increasing

Both the hosting website and the feeding website tend to gain from syndication. In simple terms, the hosting website gains exposure of its content across numerous websites and the sites consuming the feeds gains more depth in the information displayed in its pages.
Both RSS and Atom have their own respective backing in the industry and both these will likely to continue their sustenance. Selecting to implement either of the standards for a website is need and convenience based.

https://en.wikipedia.org/wiki/RSS#RSS_compared_with_Atom

RSS compared with Atom

Both RSS and Atom are widely supported and are compatible with all major consumer feed readers. RSS gained wider use because of early feed reader support. Technically, Atom has several advantages: less restrictive licensing, IANA-registered MIME type, XML namespace, URI support, Relax NG support.[30]
The following table shows RSS elements alongside Atom elements where they are equivalent.
Note: the asterisk character (*) indicates that an element must be provided (Atom elements "author" and "link" are only required under certain conditions).

RSS 2.0	**Atom 1.0**

author	author *
category	category
channel	feed
copyright	rights
—	subtitle
description *	summary and/or content
generator	generator
guid	id *
image	logo
item	entry
lastBuildDate (in channel)	updated *
link *	link *
managingEditor	author or contributor
pubDate	published (subelement of entry)
title *	title *
ttl	—

Current usage

Several major sites such as Facebook and Twitter previously offered RSS feeds but have reduced or removed support. Additionally, widely used readers such as Shiira, FeedDemon, and Google Reader have been discontinued having cited declining popularity in RSS.[31] RSS support was removed in OS X Mountain Lion's versions of Mail and Safari, although the features were partially restored in Safari 8.[32][33] Mozilla removed RSS support from Mozilla Firefox version 64.0, joining Google Chrome and Microsoft Edge which do not include RSS support, thus leaving Internet Explorer as the last major browser to include RSS support by default.