UNIT-1
HTML & CSS

by
Ranjani.B,
AP/CSE, EGSPEC

Contents

XHTML Evolution of HTML and XHTML- Standard XHTML Document Structure- Basic Text Markup- Images-Hypertext Links-Lists- Tables- Forms- Frames. Cascading Style Sheets Introduction to CSS – Levels of Style Sheets- Style Specification Formats- Selector Forms- Property Value Forms – Font Properties- List Properties – Color- Alignment of Text – Background Images- Span and Div Tags.

What are Markup Languages?

These markup languages are designed specifically to work for **web development**. Today, it is a foundation of the web and everything you see is the combination of markup text, CSS and front end scripts (interactivity) are based on mark up languages. It is what creates the final web presence. It sets the architecture of the website and gives it structure. These mark up languages structures data. Unlike other languages like Python or **PHP** or Ruby that guide the behavior of the data and the databases.

INTRODUCTION TO HTML

- **With HTML you can create your own Web site.**
- HTML stands for **Hyper Text Markup Language.**
- HTML is derived from a language **SGML (Standard Graphics Markup Language).**
- The future of HTML is **XML (eXtended Markup Language).**
- HTML is not a programming language, it is a **Markup Language.**
- A markup language is a set of **markup tags.**
- HTML uses **markup tags to describe web pages.**
- HTML is **not case sensitive language.**
- HTML documents **contain HTML tags and plain text.**

- HTML – Hyper Text Markup Language
- HTML documents describe web pages (Static Web Page)
- HTML tags are keywords surrounded by angle brackets like <html>
- HTML tags normally come in pairs like <b> and </b>
- The first tag in a pair is the start tag (opening tags),
- the second tag is the end tag(closing tags)

Origins and Evolution of HTML

- Hypertext Markup Language
- Developed for the delivery of hypertext on the WWW
- Built using SGML
- ASCII “Markup Language”

Recent Versions

- Tim Berners Lee – person who defined HTML in 1990 at CERN.
- HTML 2.0 – 1994
  - launched World Wide Web Consortium(W3C) – standards for html
- HTML 4.0 – 1997
  - Introduced many new features and deprecated many older features
- HTML 4.01 - 1999 - A cleanup of 4.0
- XHTML 1.0 – 2000
  - Just 4.01 defined using XML, instead of SGML
- XHTML 1.1 – 2001
  - Modularized 1.0, and drops frames
- XHTML 2.0 – development abandoned
- HTML 5.0
  - Working Draft (not a W3C Recommendation yet)
  - HTML and XHTML syntax

What Is XHTML?

- XHTML stands for
  E**X**tensible **H**yper**T**ext **M**arkup **L**anguage
- XHTML is almost identical to HTML
- XHTML is stricter than HTML
- XHTML is HTML defined as an XML application
- XHTML is supported by all major browsers
- XHTML was developed by combining the strengths of HTML and XML.
- XHTML is HTML redesigned as XML.

Most Important Differences from HTML

- Document Structure
  - XHTML DOCTYPE is **mandatory**
  - The xmlns attribute in <html> is **mandatory**
  - <html>, <head>, <title>, and <body> are **mandatory**
- XHTML Elements
  - XHTML elements must be **properly nested**
  - XHTML elements must always be **closed**
  - XHTML elements must be in **lowercase**
  - XHTML documents must have **one root element**
- XHTML Attributes
  - Attribute names must be in **lower case**
  - Attribute values must be **quoted**
  - Attribute minimization is **forbidden**

How to Convert from HTML to XHTML

1. Add an XHTML <!DOCTYPE> to the first line of every page
2. Add an xmlns attribute to the html element of every page
3. Change all element names to lowercase
4. Close all empty elements
5. Change all attribute names to lowercase
6. Quote all attribute values

<!DOCTYPE ....> Is Mandatory

- An XHTML document must have an XHTML DOCTYPE declaration.
- The <html>, <head>, <title>, and <body> elements must also be present, and the xmlns attribute in <html> must specify the xml namespace for the document.

EXAMPLE

- This example shows an XHTML document with a minimum of required tags:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

   <html xmlns="http://www.w3.org/1999/xhtml">

   <head>
     <title>Title of document</title>
   </head>

   <body>
     some content
   </body>

   </html>
```

- XHTML Elements Must Be Properly Nested
  - This is wrong
    - <b><i>This text is bold and italic</b></i>
  - This is correct
    - <b><i>This text is bold and italic</i></b>
- XHTML Elements Must Always Be Closed
  - This is wrong
    - <p>This is a paragraph
    - <p>This is another paragraph
  - This is correct
    - <p>This is a paragraph</p>
    - <p>This is another paragraph</p>

- Empty Elements Must Also Be Closed
  - This is wrong
    - A break: <br>
    - A horizontal rule: <hr>
    - An image: <img src="happy.gif" alt="Happy face">
  - This is correct
    - A break: <br />
    - A horizontal rule: <hr />
    - An image: <img src="happy.gif" alt="Happy face" />

- XHTML Elements Must Be In Lower Case
  - This is wrong
    - <BODY>
      <P>This is a paragraph</P>
      </BODY>
  - This is correct
    - <body>
      <p>This is a paragraph</p>
      </body>

- XHTML Attribute Names Must Be In Lower Case
  - This is wrong
    - <table WIDTH="100%">
  - This is correct
    - <table width="100%">
- Attribute Values Must Be Quoted
  - This is wrong
    - <table width=100%>
  - This is correct
    - <table width="100%">

- Attribute Minimization Is Forbidden
  - Wrong:
    - <input type="checkbox" name="vehicle" value="car" checked />
  - Correct:
    - <input type="checkbox" name="vehicle" value="car" checked="checked" />
  - Wrong:
    - <input type="text" name="lastname" disabled />
  - Correct:
    - <input type="text" name="lastname" disabled="disabled" />

HTML vs XHTML

HTML and **XHTML** are both languages in which web pages are written. HTML is SGML-based while XHTML is XML-based. They are like two sides of the same coin. XHTML was derived from HTML to conform to XML standards. Hence XHTML is strict when compared to HTML and does not allow user to get away with lapses in coding and structure.

HTML vs XHTML

S.No	HTML	XHTML
1	HTML or HyperText Markup Language is the main markup language for creating web pages and other information that can be displayed in a web browser.	XHTML (Extensible HyperText Markup Language) is a family of XML markup languages that mirror or extend versions of the widely used Hypertext Markup Language (HTML), the language in which web pages are written.
2	.html, .htm	.xhtml, .xht, .xml, .html, .htm
3	HTML is based on SGML.	XHTML is based on XML.
4	HTML tags are case insensitive.	XHTML tags are case sensitive.

Sr. No.	HTML	XHTML
1.	The HTML tags are case **insensitive**. Hence <body> or <BODY> or <Body> are treated as one and the same.	The XHTML is **case sensitive** and all the tags in XHTML document must be written in lower case.
2.	We can **omit** the **closing tags** sometimes in HTML document.	For every tag there **must be a closing tag**. Some browsers get confused if the closing tag is not given. There are two ways by which we can mention the closing tags <a href ="TajMahal.html"> </a> or <a href="Tajmahal.html"/>
3.	In HTML the **attribute values** it not always necessary to **quote** the attribute values. In fact numeric attribute values are rarely quoted in HTML. Only if some special characters or white spaces are present in the attribute values then only it is essential to put quotes around them in HTML.	In every XHTML document the attribute values must be quoted.
4.	In HTML there are some implicit attribute values.	In every XHTML the attribute values must be specified explicitly.

5.	In HTML even if we do not follow the nesting rules strictly it does not cause much difference.	In XHTML document the nesting rules must be strictly followed. These nesting rules are- A form element can not contain another form element. An anchor element does not contain another form element. List element can not be nested in the list element's. If there are two nested elements then the inner element must be enclosed first before closing the outer element. Text elements can not be directly nested in form elements.

6.2.1 Features of HTML 5.0

1. The XHTML doctype is just impossible to memorize (!DOCTYPE html PUBLIC "...). Hence a new HTML doctype is <!DOCTYPE html>
2. There are new graphic elements such as <canvas> and <svg> in HTML5.0
3. The support for multimedia elements such as <audio> and <video>
4. It has support for <header>, <footer>, <article>, and <section>
5. It has support for new form controls such as number, date, time, calendar, and range.
6. It has a rich set of Application Programming Interface(API) for geolocations, HTML Drag and Drop, Local Storage, Application cache and so on.

Sr.No.	HTML	HTML5
1.	The DOCTYPE declaration is much longer such as <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml11.dtd">	The DOCTYPE declaration is simple <!DOCTYPE html>
2.	<audio> and <video> tags are not supported.	<audio> and <video> tags are supported for playing audio and video files.
3.	Finding out geographic location is impossible using HTML	The HTML5 supports the API for identifying the geographic location.
4.	It supports the tag such as <applet>, <big>, <center>, <font>, <frame>, <strike>	The tags that are removed from HTML5 are <applet>, <big>, <center>, <font>, <frame>, <strike>
5.	Does not allow JavaScript to run in browser. JavaScript runs in same thread as browser interface.	It allows JavaScript to run in background.
6.	There is no support for <canvas>	The <canvas> tag is for 2D drawing.
7.	It needs external plugins such as flash.	The need for external plugin is reduced.

HTML Parts

- An HTML document consists of 3 main parts
  - DOCTYPE declaration
  - <head> section
  - <body> section

```
<!DOCTYPE html>
<html>
<head>
 <title> …. </title>
</head>
<body>
….
….
…
</body>
</html>
```

HTML Document Structure

- An HTML document is composed of 3 parts:
  - 1. a line containing HTML version information, e.g.:
    - <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN“ "http://www.w3.org/TR/html4/strict.dtd">

      Note: <!DOCTYPE html> for HTML5
  - 2. a declarative header section
    - Delimited with the <head> tag
    - The <title>tag is used to give the document a title normally displayed in the browser’s window title bar
  - 3. a body containing the document's actual content
    - Delimited with the <body> tag
    - After document type declaration, the remainder of an HTML document is contained by the html element

Document Type Definitions (DTD)

* A DTD specifies the syntax of a web page in SGML
* DTDs are used by SGML applications, such as HTML, to specify rules for documents of a particular type, including a set of elements and entity declarations
* An XHTML DTD describes in precise, computer-readable language, the allowed syntax of XHTML markup

There are three XHTML DTDs:

* STRICT
* TRANSITIONAL
* FRAMESET

Three Flavours of XHTML

There are three types of XHTML DTDs and those along with their uses are as given below -

1. **XHTML 1.0 Strict :** When we want a clean markup code then this type of dtd is used.
2. **XHTML 1.0 Transitional ;** When we want to use some html features in the existing XHTML document.
3. **XHTML 1.0 Frameset :** When want to make use of frames in the XHTML document.

This is a simple (minimal) XHTML document:

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-
   transitional.dtd">
<html>
<head>
<title>simple document</title>
</head>
<body>
<p>a simple paragraph</p>
</body>
</html>
```

e1

- XHTML 1.0 Strict
- <!DOCTYPE html
- PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
- "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
- Use the strict DOCTYPE when you want really clean markup, free of presentational clutter. Use together with CSS.

- XHTML 1.0 Transitional
- <!DOCTYPE html
- PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
- "http://www.w3.org/TR/xhtml1/DTD /xhtml1-transitional.dtd">
- Use the transitional DOCTYPE when you want to still use HTML's presentational features.

- XHTML 1.0 Frameset
- <!DOCTYPE html
- PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
- "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
- Use the frameset DOCTYPE when you want to use HTML Frames to split the web page into two or more frames.

HTML Elements and Tags

- A tag is always enclosed in angle bracket <>like <HTML>
- HTML tags normally **come in pairs like** <HTML> and </HTML>
- i.e. **Start tag = <HTML> & End tag =</HTML>**
- Start and end tags are also called **opening tags and closing tags.**

HOW TO START

- Write html code in notepad.
- Save the file with (.Html)/(.Htm) extension.
- View the page in any web browser viz. INTERNET EXPLORER, NETSCAPE NAVIGATOR etc.
- The purpose of a web browser (like internet explorer or firefox) is to read html documents and display them as web pages.

HTML syntax

1.The DOCTYPE

2.Elements

3.Attributes

4.Comments

DOCTYPE

< !DOCTYPE html> - Required Header

Elements

<element> …………… </element>

Start tag Content End tag

Attributes

<element attribute=”value”>

Name Value

Comments

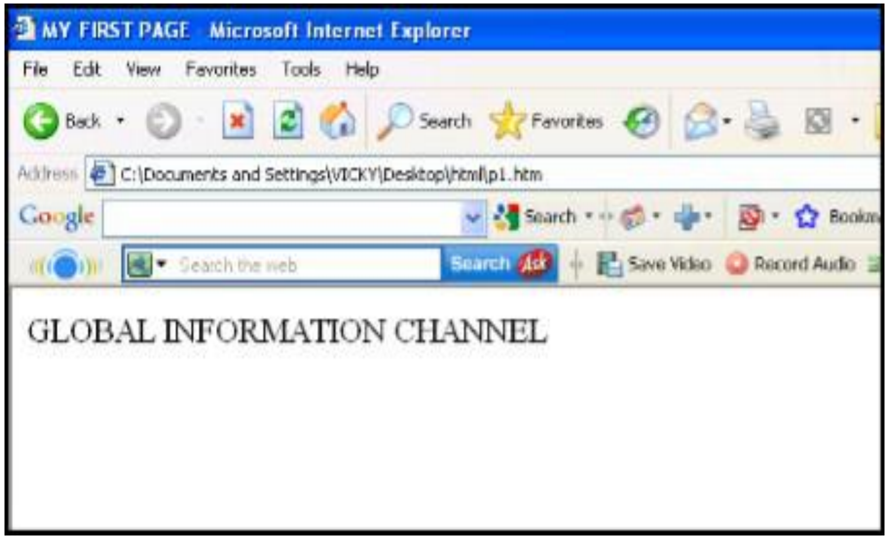
<!-- comment -->

Start Comment End

- **<HTML> - Describe HTML web page that is** to be viewed by a web browser.
- **<HEAD> - This defines the header section** of the page.
- **<TITLE> - This shows a caption in the title** bar of the page.
- **<BODY> - This tag show contents of the** web page will be displayed.

Code With HTML

```
<HTML>
<HEAD>
<TITLE> MY FIRST PAGE    </TITLE>
</HEAD>
<BODY>
GLOBAL INFORMATION CHANNEL
</BODY>
</HTML>
```


MY FIRST PAGE   Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\Documents and Settings\VICKY\Desktop\html\p1.htm
Google Search Bookm
Search the web Search Save Video Record Audio

GLOBAL INFORMATION CHANNEL

Types of HTML Tags

- There are two different types of tags:->

Container Element:->

Container Tags contains **start tag & end tag**

i.e. <HTML>… </HTML>

Empty Element:->

Empty Tags contains **start tag**

i.e. <BR> - break

Text Formatting Tags

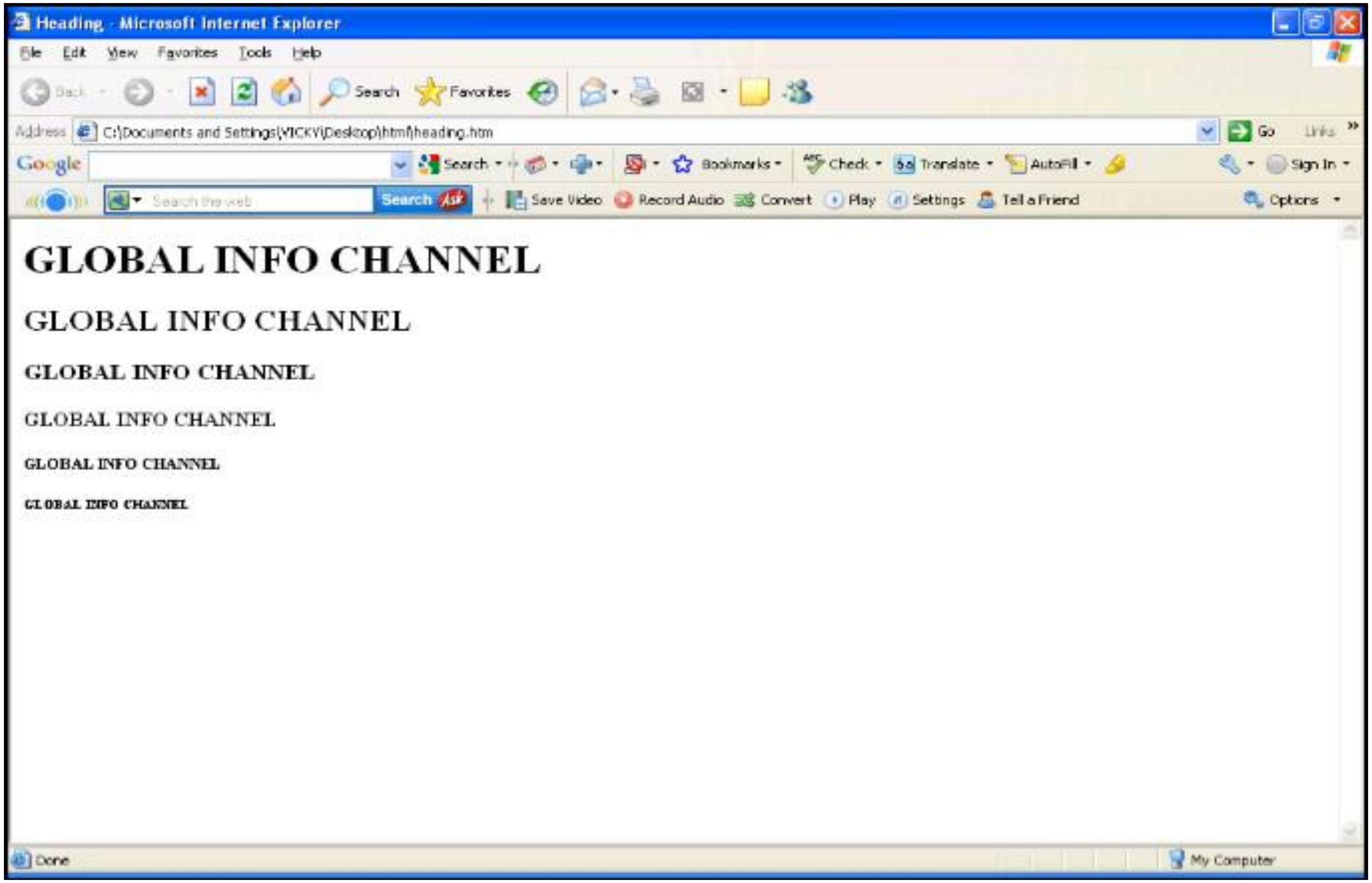
Heading Element:->

- There are six heading elements (<H1>,<H2>,<H3>,<H4>, <H5>,<H6>).
- All the six heading elements are container tag and requires a closing tag.
- **<h1>** will print the **largest heading**
- **<h6>** will print the **smallest heading**

Heading Tag Code

```
<html>
<head>
   <title>heading</title>
</head>
<body>
  <h1> GLOBAL INFO CHANNEL</h1>
  <h2> GLOBAL INFO CHANNEL</h2>
  <h3> GLOBAL INFO CHANNEL</h3>
  <h4> GLOBAL INFO CHANNEL</h4>
  <h5> GLOBAL INFO CHANNEL</h5>
  <h6> GLOBAL INFO CHANNEL</h6>
</body>
</html>
```

Result of Heading Code


Heading - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\Documents and Settings\VICKY\Desktop\html\heading.htm
Go Links
Done
My Computer

GLOBAL INFO CHANNEL

GLOBAL INFO CHANNEL

GLOBAL INFO CHANNEL

GLOBAL INFO CHANNEL

GLOBAL INFO CHANNEL

GLOBAL INFO CHANNEL

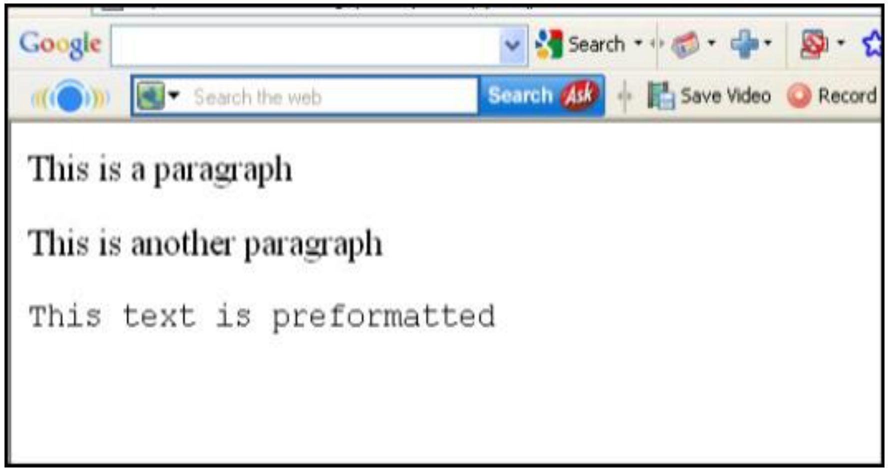
HTML Paragraph Tag

- HTML documents are divided into paragraphs.
- Paragraphs are defined with the **<p> tag**

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

  <p>This is another paragraph</p>

  <pre>This text is preformatted</pre>



This is a paragraph

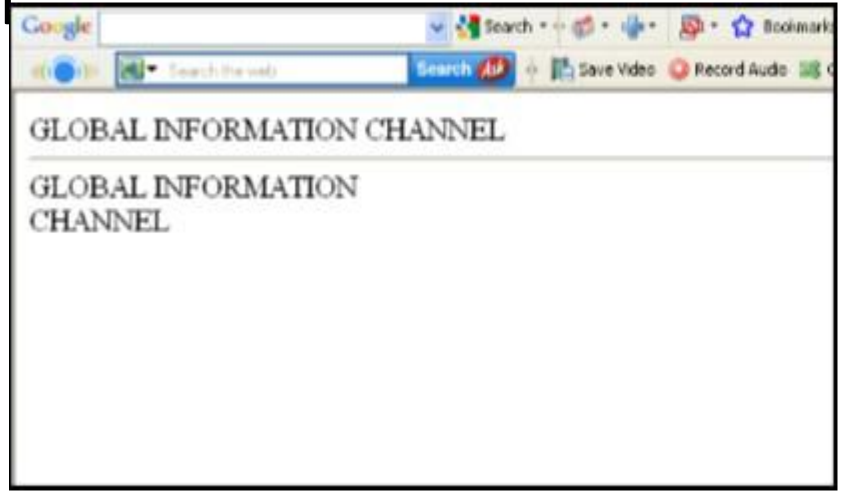
This is another paragraph

```
This text is preformatted
```

Line Break & Horizontal Line Tag

- If you want a line break or a new line without starting a new paragraph Use the <br> tag.
- Defines a horizontal line use <hr>tag.
- <br> <hr> element are empty HTML element

  i.e. Global Information Channel <hr> Global Information <br> Channel



```
GLOBAL INFORMATION CHANNEL
--------------------------
GLOBAL INFORMATION
CHANNEL
```

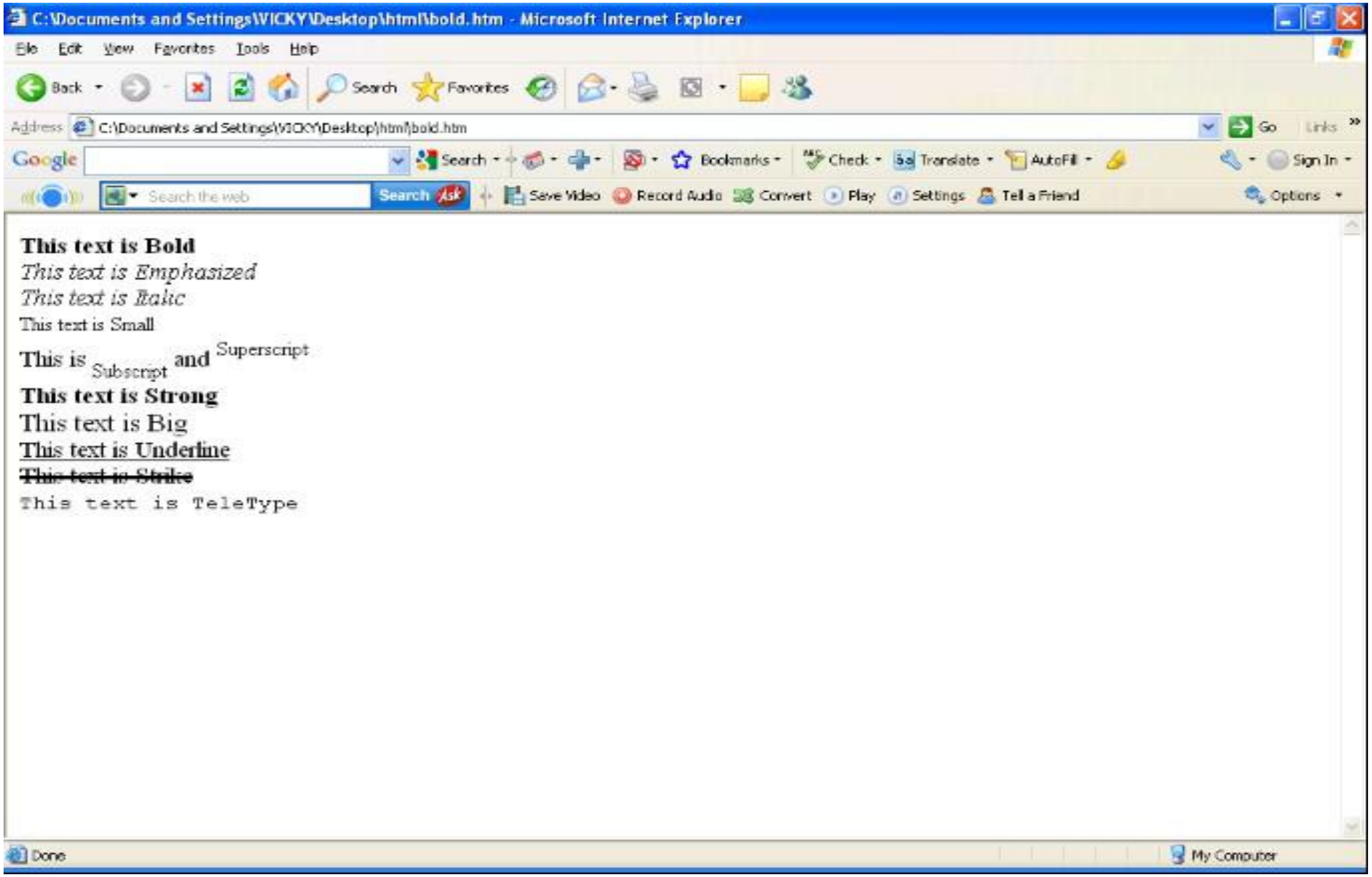
Text Formatting Tags

<b>	Defines bold text
<big>	Defines big text
<em>	Defines emphasized text
<i>	Defines italic text
<small>	Defines small text
<strong>	Defines strong text
<sub>	Defines subscripted text
<sup>	Defines superscripted text
<ins>	Defines inserted text
<del>	Defines deleted text
<tt>	Defines teletype text
<u>	Defines underline text
<strike>	Defines strike text

Text Formatting Code

```
<html>
<head></head>
<body>
<b>This text is Bold</b>
<br><em>This text is Emphasized</em>
<br><i>This text is Italic</i>
<br><small>This text is Small</small>
<br>This is<sub> Subscript</sub> and
<sup>Superscript</sup>
<br><strong>This text is Strong</strong>
<br><big>This text is Big</big>
<br><u>This text is Underline</u>
<br><strike>This text is Strike</strike>
<br><tt>This text is Teletype</tt>
</body>
</html>
```

Result of Text Formatting Code


C:\Documents and Settings\VICKY\Desktop\html\bold.htm - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address C:\Documents and Settings\VICKY\Desktop\html\bold.htm
Go
Links
Done
My Computer

This text is Bold
This text is Emphasized
This text is Italic
This text is Small
This is $_{\text{Subscript}}$ and $^{\text{Superscript}}$
This text is Strong
This text is Big
This text is Underline
~~This text is Strike~~
`This text is TeleType`

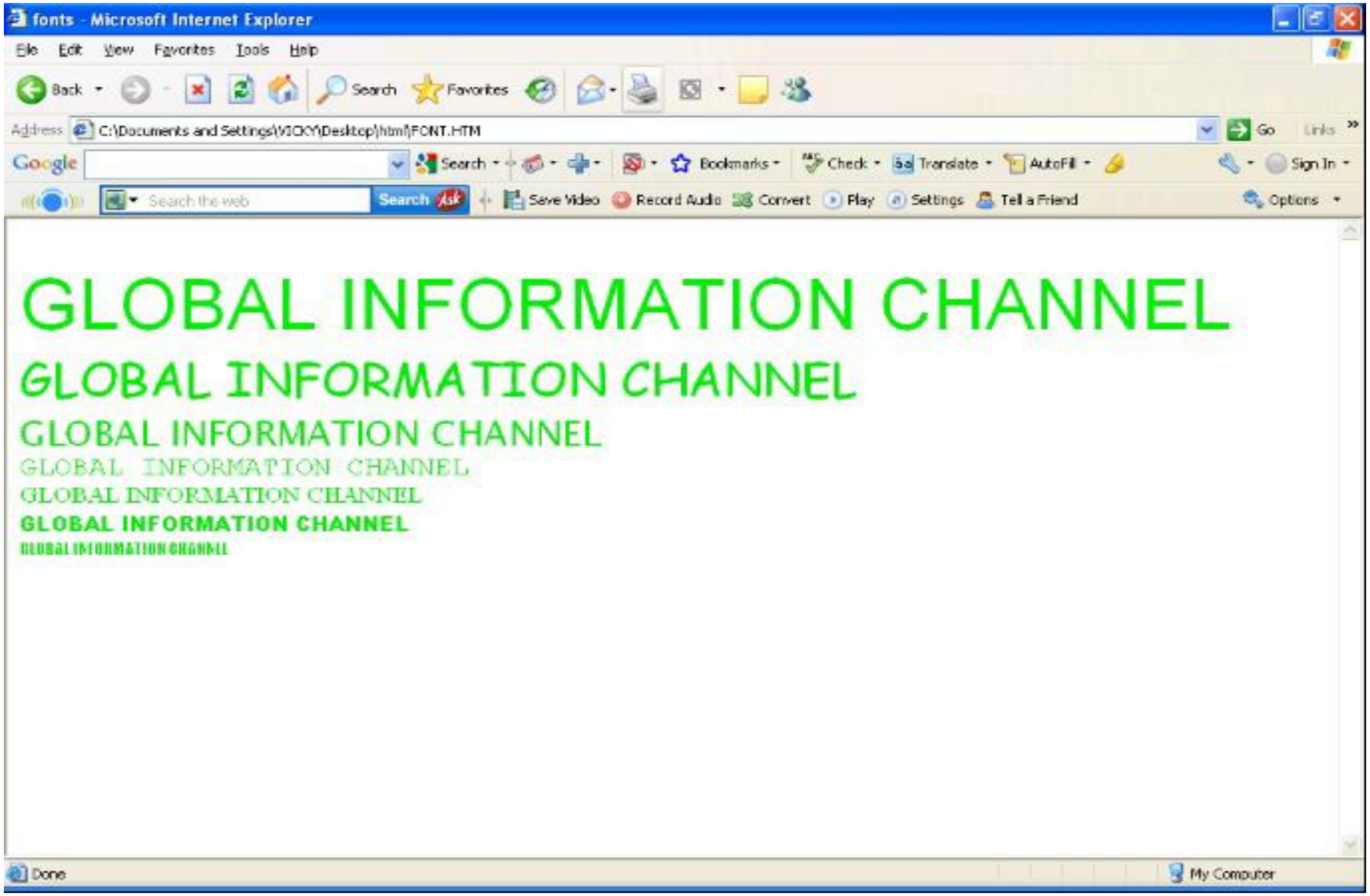
Font Tag

- This element is used to format the **size, typeface and color of the enclosed text.**
- The commonly used fonts for web pages are Arial, Comic Sans MS , Lucida Sans
- Unicode, Arial Black, Courier New, Times, New Roman, Arial Narrow, Impact,Verdana.
- The size attribute in font tag takes values from **1 to 7**.

Font Tag Code

```
<html>
<head><title> fonts</title></head>
<body>
<br><font color=“green" size="7" face="Arial"> GLOBAL INFORMATION CHANNEL
    </font>
<br><font color=“green" size="6" face="Comic Sans MS "> GLOBAL INFORMATION
    CHANNEL </font>
<br><font color=“green" size="5" face="Lucida Sans Unicode"> GLOBAL INFORMATION
    CHANNEL </font>
<br><font color=“green" size="4" face="Courier New"> GLOBAL INFORMATION
    CHANNEL </font>
<br><font color=“green" size="3" face="Times New Roman"> GLOBAL INFORMATION
    CHANNEL </font>
<br><font color=“green" size="2" face="Arial Black"> GLOBAL INFORMATION CHANNEL
    </font>
<br><font color=“green" size="1" face="Impact"> GLOBAL INFORMATION CHANNEL
    </font>
</body>
</html>
```

Result of Font Code


fonts - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\Documents and Settings\VICKY\Desktop\html\FONT.HTM
Go Links
Done
My Computer

GLOBAL INFORMATION CHANNEL
GLOBAL INFORMATION CHANNEL
GLOBAL INFORMATION CHANNEL
GLOBAL INFORMATION CHANNEL
GLOBAL INFORMATION CHANNEL
GLOBAL INFORMATION CHANNEL
GLOBAL INFORMATION CHANNEL

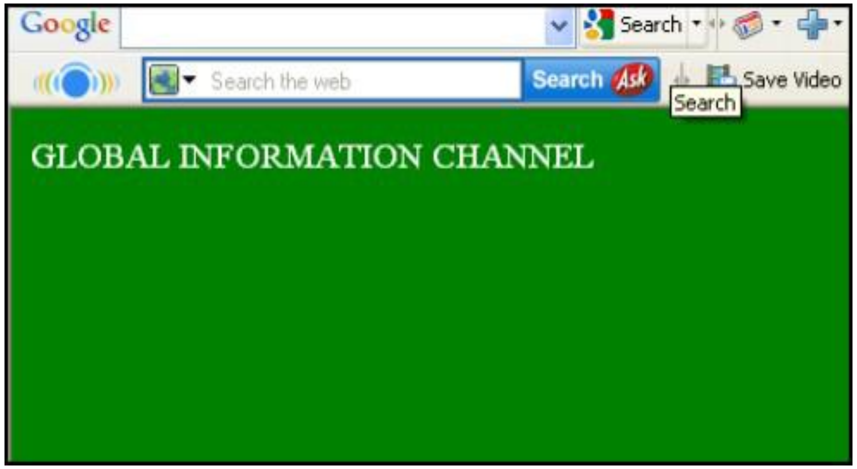
Background & Text Color Tag

- The attribute bgcolor is used for changing the back ground color of the page.

  `<body bgcolor="Green" >`

- Text is use to change the color of the enclosed text.

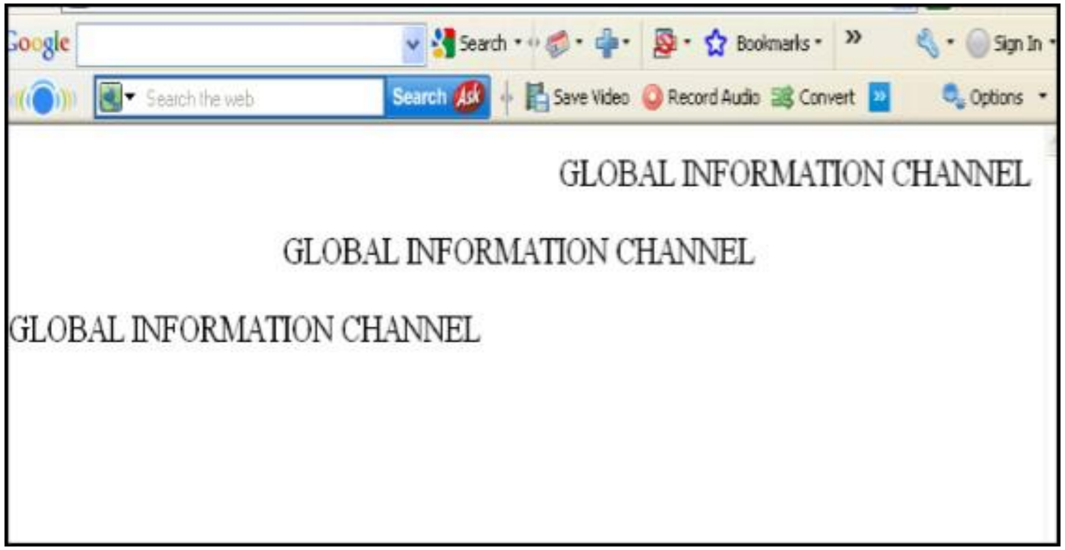
  `<body text="White">`



GLOBAL INFORMATION CHANNEL

Text Alignment Tag

- It is use to alignment of the text.
  - –Left alignment <align="left">
  - –Right alignment <align="right">
  - –Center alignment <align="center">



GLOBAL INFORMATION CHANNEL

GLOBAL INFORMATION CHANNEL

GLOBAL INFORMATION CHANNEL

Hyperlink Tag

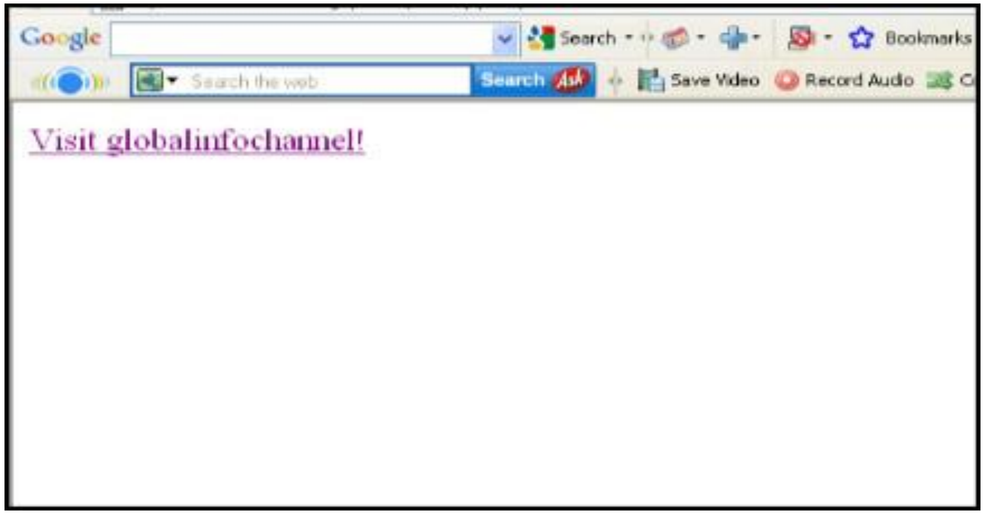
- A hyperlink is a reference (an address) to a resource on the web.
- Hyperlinks can point to any resource on the web: an HTML page, an image, a sound file, a movie, etc.
- The HTML anchor element <a>, is used to define both hyperlinks and anchors.

```
<a href="url">Link text</a>
```

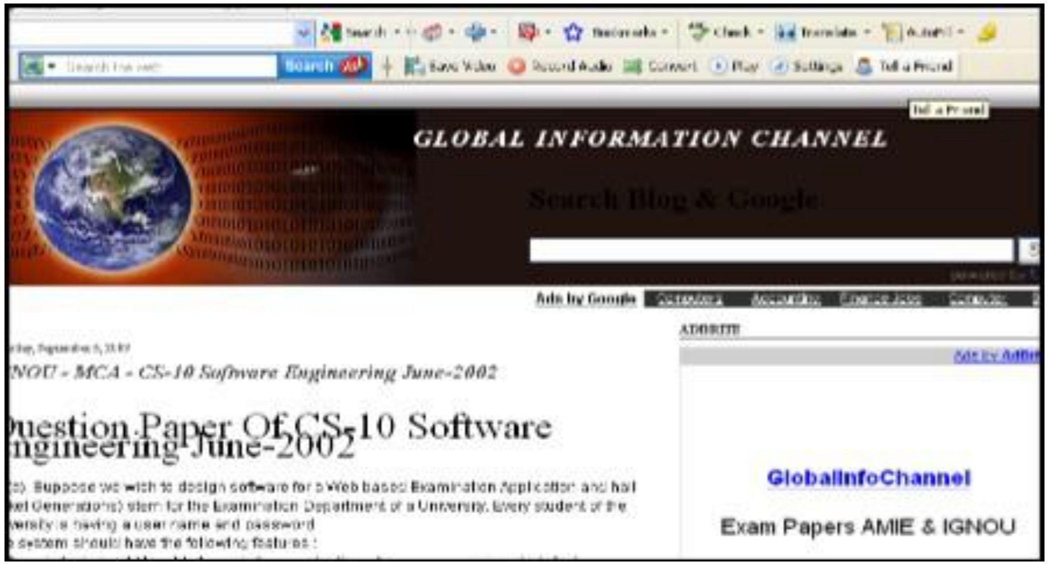
- The **href attribute defines the link address.**

```
<a href="http://www.globalinfochannel/">Visit
globalinfochannel!</a>
```

Result of Hyperlink Code



Visit globalinfochannel!



NOU - MCA - CS-10 Software Engineering June-2002

Question Paper Of CS-10 Software Engineering June-2002

Image Tag

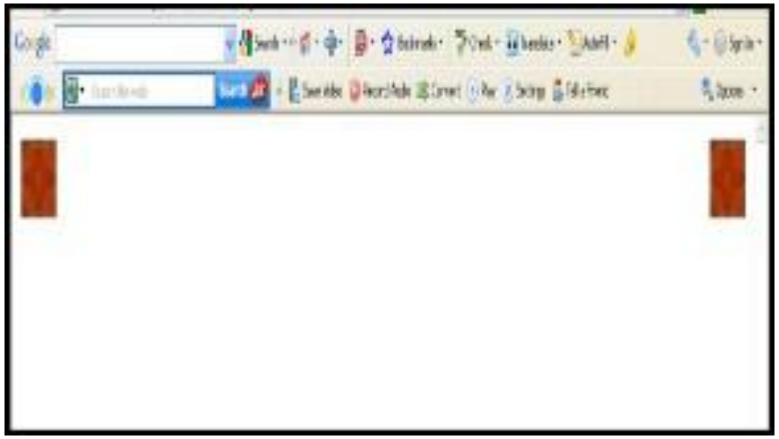
- To display an image on a page, you need to use the src attribute.
- src stands for "source". The value of the src attribute is the URL of the image you want to display on your page.
- It is a empty tag.

```
        <IMG SRC ="url">
        <IMG SRC="picture.gif“>
        <IMG SRC="picture.gif“ HEIGHT="30“
WIDTH="50">
```

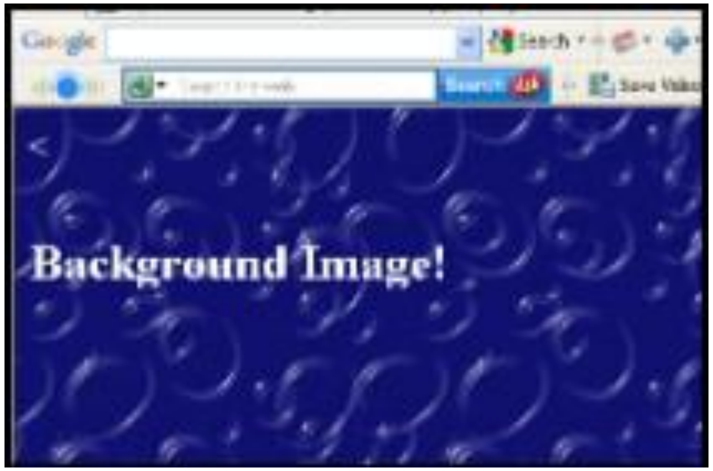
Image attributes - <img> tag

<img>	Defines an image
<Src>	display an image on a page, Src stands for "source"
<Alt>	Define "alternate text" for an image
<Width>	Defines the width of the image
<Height>	Defines the height of the image
<Border>	Defines border of the image
<Hspace>	Horizontal space of the image
<Vspace>	Vertical space of the image
<Align>	Align an image within the text
<background>	Add a background image to an HTML page

Code & Result of the Image

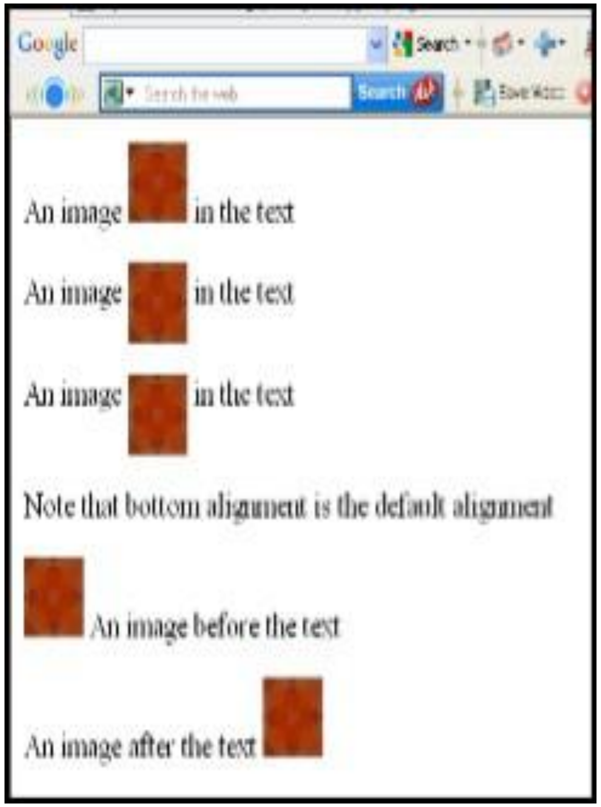


```
<html><body>
<p><img
src="file:///C:/WINDOWS/Zapotec.bmp"
align="left" width="48" height="48"> </p>
<p><img src
="file:///C:/WINDOWS/Zapotec.bmp"
align="right" width="48" height="48"></p>
</body></html>
```



```
<HTML>
<<body
background="file:///C:/WINDOWS/Soap
%20Bubbles.bmp" text="white">
<br><br><br>
<h2> Background Image!</h2>
</BODY></HTML>
```

Code & Result of the Image



An image in the text

An image in the text

An image in the text

Note that bottom alignment is the default alignment

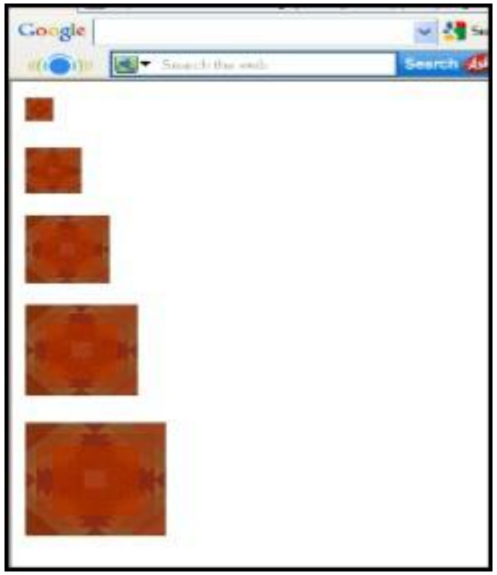
An image before the text

An image after the text

```
<html><body>
<p>An image
<img src="file:///C:/WINDOWS/Zapotec.bmp"
align="bottom" width="48" height="48"> in the text</p>
<p>An image
<img src ="file:///C:/WINDOWS/Zapotec.bmp"
align="middle" width="48" height="48"> in the text</p>
<p>An image
<img src ="file:///C:/WINDOWS/Zapotec.bmp"
align="top" width="48" height="48"> in the text</p>
<p>Note that bottom alignment is the default
alignment</p>
<p><img src ="file:///C:/WINDOWS/Zapotec.bmp"
width="48" height="48">
An image before the text</p>
<p>An image after the text
<img src ="file:///C:/WINDOWS/Zapotec.bmp"
width="48" height="48"> </p>
</body></html>
```

Code & Result of the Image

```
<html><body>
<p><img src="file:///C:/WINDOWS/Zapotec.bmp"
align="bottom" width="20" height="20"> </p>
<p><img src ="file:///C:/WINDOWS/Zapotec.bmp"
align="middle" width="40" height="40"></p>
<p><img src ="file:///C:/WINDOWS/Zapotec.bmp"
align="top" width="60" height="60"></p>
<p><img src ="file:///C:/WINDOWS/Zapotec.bmp"
width="80" height="80"> </p>
<p><img src ="file:///C:/WINDOWS/Zapotec.bmp"
width="100" height="100"> </p>
</body></html>
```



HTML Table Tag

<table>	used to create table
<tr>	table is divided into rows
<td>	each row is divided into data cells
<th>	Headings in a table
<Caption>	caption to the table
<colgroup>	Defines groups of table columns
<col>	Defines the attribute values for one or more columns in a table
<thead>	Defines a table head
<tbody>	Defines a table body
<tfoot>	Defines a table footer
<Cellspacing>	amount of space between table cells.
<Cellpadding>	space around the edges of each cell
<Colspan>	No of column working with will span
<rowspan>	No of rows working with will span attribute takes a number

<table> Tag

- The <table> tag defines an HTML table.
- An HTML table consists of the <table> element
- <table> closing tag is </table>
- An HTML table has two kinds of cells:
  - Header cells - contains header information (created with the <th> element)
  - Standard cells - contains data (created with the <td> element)

<th> Tag

- The <th> tag defines a header cell in an HTML table.
- Closing tag is </th>
- The text in <th> elements are bold and centered by default.

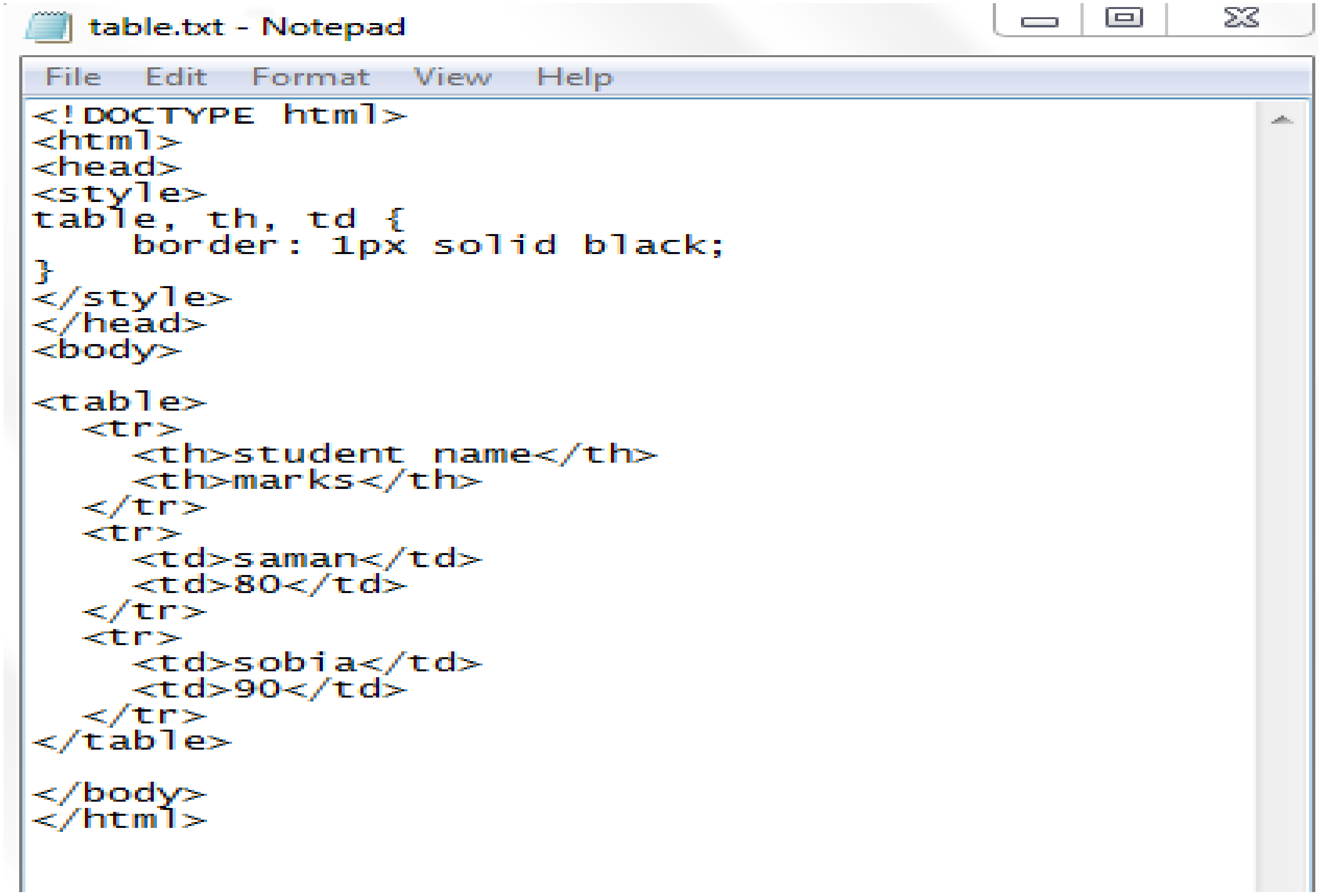
<td> Tag

- The <td> tag defines a standard cell in an HTML table.
- The text in <td> elements are regular and left-aligned by default
- Closing tag is </td>

<tr> Tag

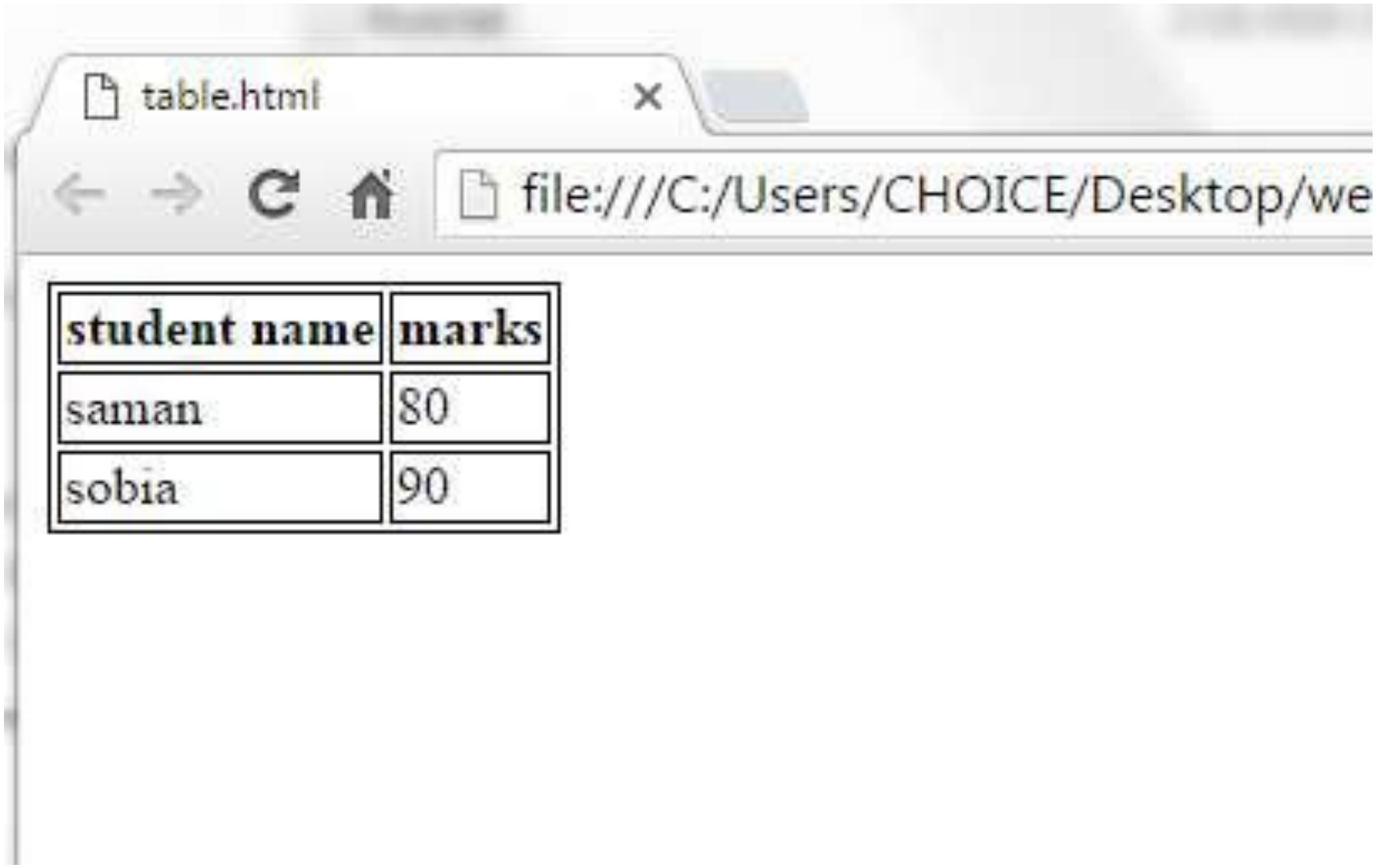
- The <tr> tag defines a row in an HTML table.
- A <tr> element contains one or more <th> or <td> elements.
- Closing tag is </tr>

<table> input



```
<!DOCTYPE html>
<html>
<head>
<style>
table, th, td {
    border: 1px solid black;
}
</style>
</head>
<body>

<table>
  <tr>
    <th>student name</th>
    <th>marks</th>
  </tr>
  <tr>
    <td>saman</td>
    <td>80</td>
  </tr>
  <tr>
    <td>sobia</td>
    <td>90</td>
  </tr>
</table>

</body>
</html>
```

<table>output



student name	marks
saman	80
sobia	90

Code & Result of the Table

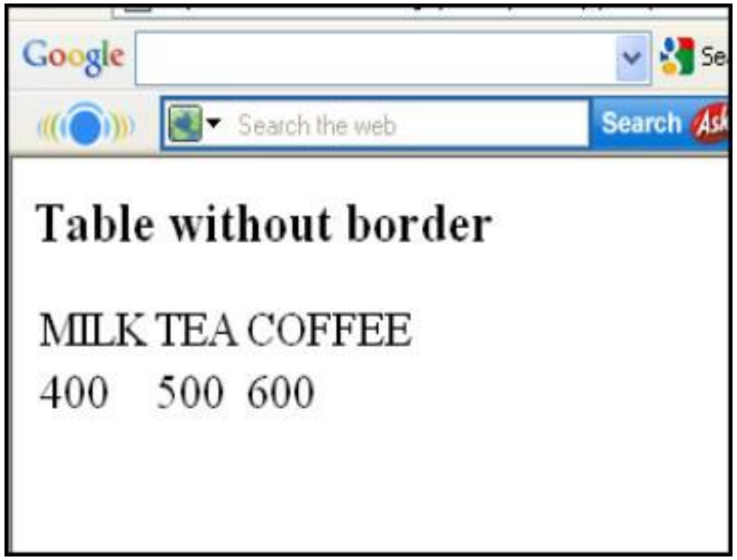


Table without border

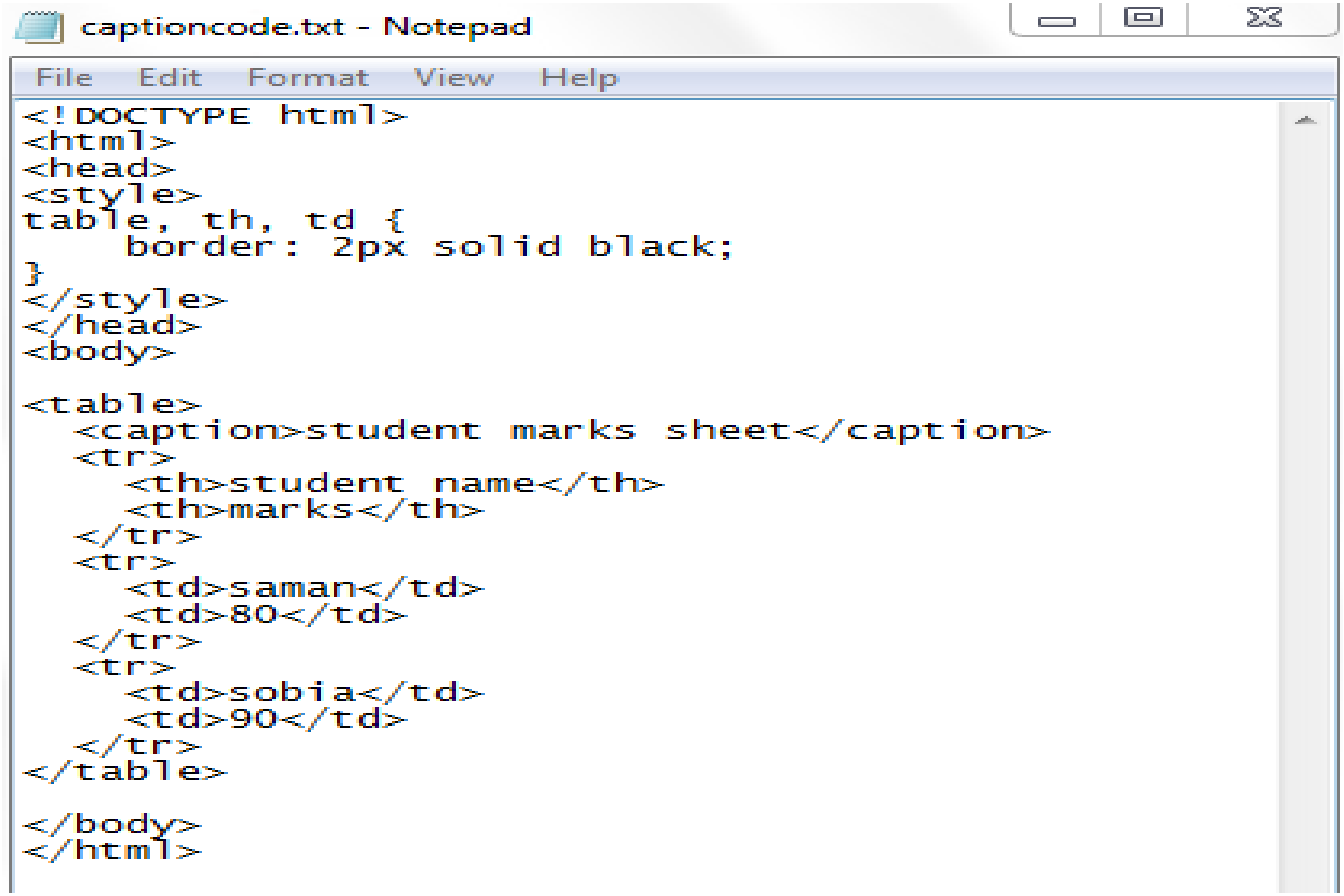
MILK TEA COFFEE
400 500 600

```
<html>
<body>
<h3>Table without border</h3>
<table>
    <tr> <td>MILK</td>
    <td>TEA</td>
    <td>COFFEE</td> </tr>
    <tr> <td>400</td>
    <td>500</td>
    <td>600</td> </tr>
</table>
</body>
</html>
```

<caption> Tag

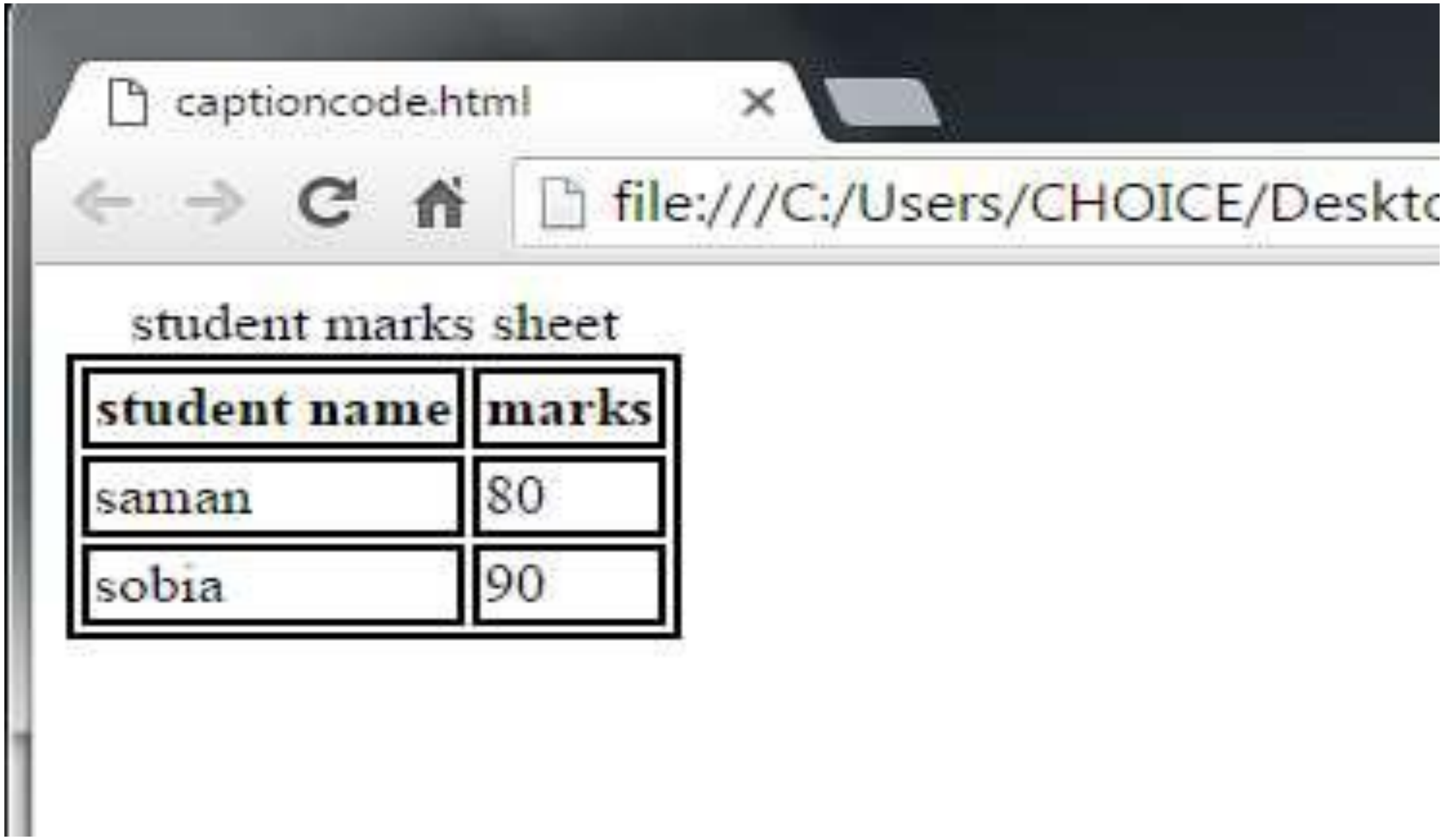
- The <caption> tag defines a table caption.
- The <caption> tag must be inserted immediately after the <table> tag.
- we can specify only one caption per table.
- By default, a table caption will be center aligned above a table.
- Closing tag is </caption>

<caption> input



```
<!DOCTYPE html>
<html>
<head>
<style>
table, th, td {
    border: 2px solid black;
}
</style>
</head>
<body>

<table>
  <caption>student marks sheet</caption>
  <tr>
    <th>student name</th>
    <th>marks</th>
  </tr>
  <tr>
    <td>saman</td>
    <td>80</td>
  </tr>
  <tr>
    <td>sobia</td>
    <td>90</td>
  </tr>
</table>

</body>
</html>
```

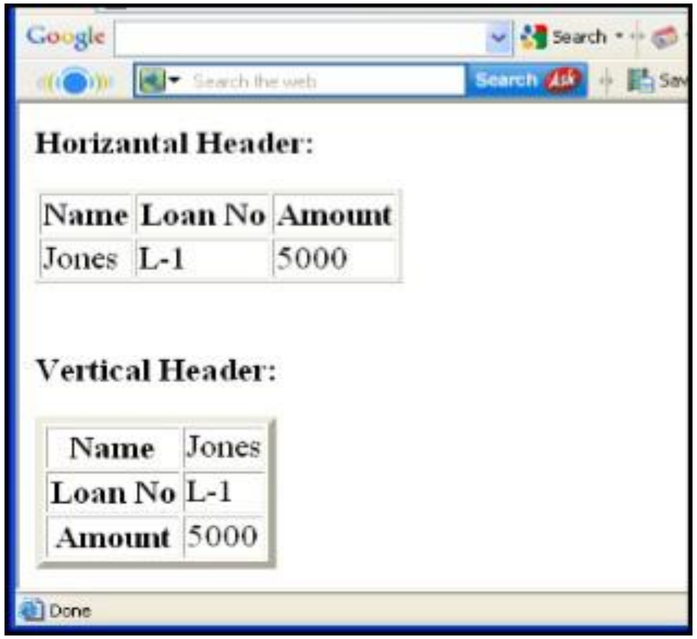
<caption> output



student marks sheet

student name	marks
saman	80
sobia	90

Table Code with Border & Header



Horizantal Header:

Name	Loan No	Amount
Jones	L-1	5000

Vertical Header:

Name	Jones
Loan No	L-1
Amount	5000

```
<html><body>
<h4>Horizontal Header:</h4>
<table border="1">
<tr> <th>Name</th> <th>Loan No</th>
<th>Amount</th> </tr>
<tr> <td>Jones</td> <td>L-1</td>
<td>5000</td></tr> </table><br><br>
<h4>Vertical Header:</h4>
<table border="5">
<tr> <th>Name</th> <td>Jones</td> </tr>
<tr> <th>Loan No</th>
<td>L-1</td> </tr>
<tr> <th>Amount</th> <td>5000</td></tr>
</table>
</body></html>
```

<thead> Tag

- The <thead> tag is used to group header content in an HTML table.
- The <thead> element is used in conjunction with the <tbody> and <tfoot> elements to specify each part of a table (header, body, footer).
- The <thead> tag must be used in the following context: As a child of a <table> element, after any <caption>, and <colgroup> elements, and before any <tbody>, <tfoot>, and <tr> elements.
- The <thead> element must have one or more <tr> tags inside.

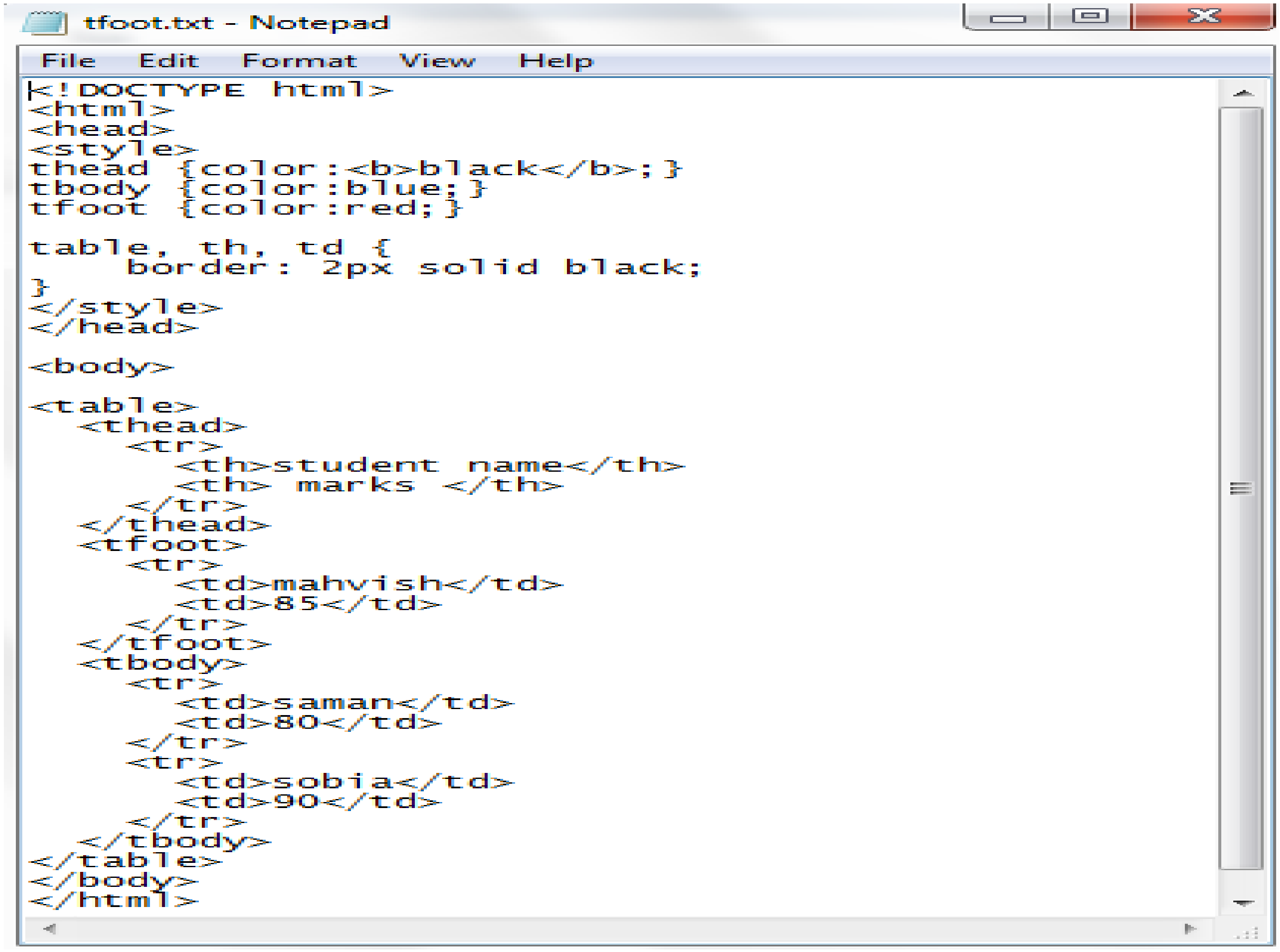
<tbody> Tag

- The <tbody> tag is used to group the body content in an HTML table.
- The <tbody> element is used in conjunction with the <thead> and <tfoot> elements to specify each part of a table (body, header, footer).
- The <tbody> tag must be used in the following context: As a child of a <table> element, after any <caption>, <colgroup>, and <thead> elements.
- The <tbody> element must have one or more <tr> tags inside.

<tfoot> Tag

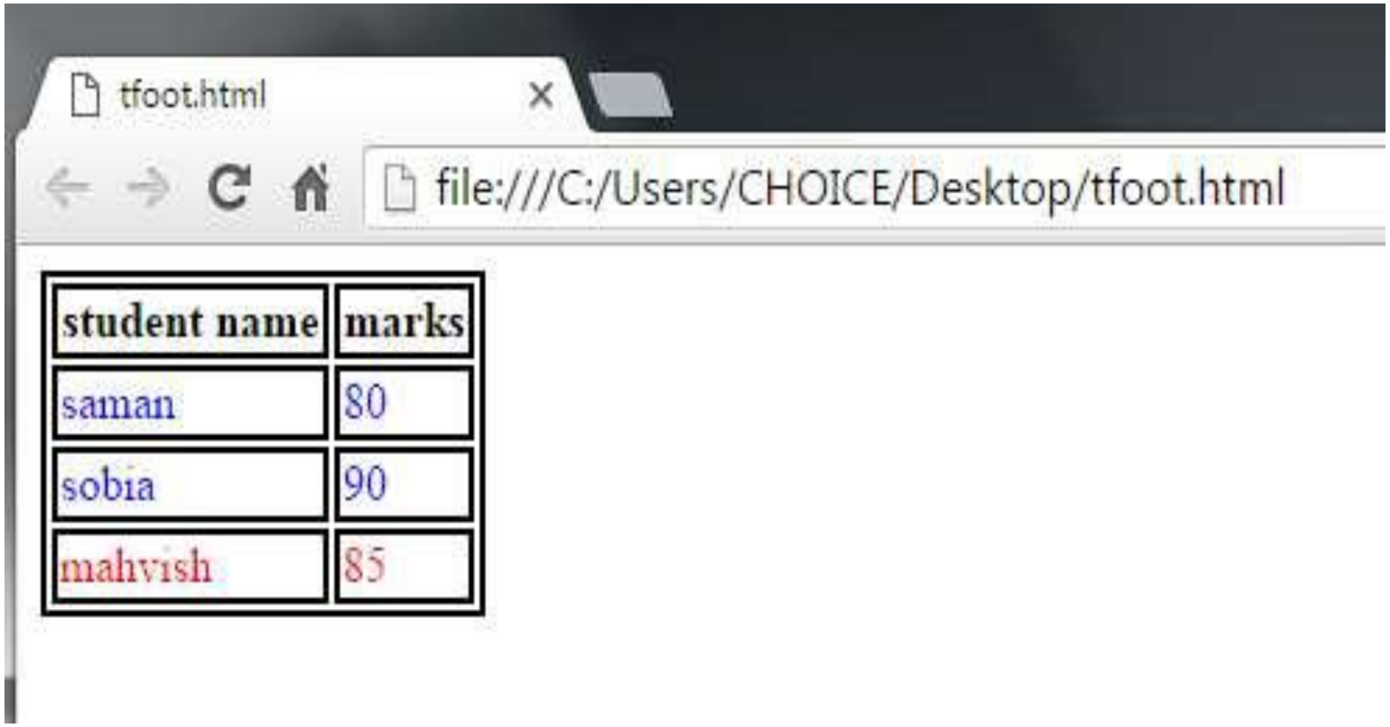
- The <tfoot> tag is used to group footer content in an HTML table.
- The <tfoot> element is used in conjunction with the <thead> and <tbody> elements to specify each part of a table (footer, header, body).
- The <tfoot> tag must be used in the following context: As a child of a <table> element, after any <caption>, <colgroup>, and <thead> elements and before any <tbody> and <tr> elements.
- The <tfoot> element must have one or more <tr> tags inside.

Input



```
<!DOCTYPE html>
<html>
<head>
<style>
thead {color:<b>black</b>;}
tbody {color:blue;}
tfoot {color:red;}

table, th, td {
    border: 2px solid black;
}
</style>
</head>

<body>

<table>
  <thead>
    <tr>
      <th>student name</th>
      <th> marks </th>
    </tr>
  </thead>
  <tfoot>
    <tr>
      <td>mahvish</td>
      <td>85</td>
    </tr>
  </tfoot>
  <tbody>
    <tr>
      <td>saman</td>
      <td>80</td>
    </tr>
    <tr>
      <td>sobia</td>
      <td>90</td>
    </tr>
  </tbody>
</table>
</body>
</html>
```

output



student name	marks
saman	80
sobia	90
mahvish	85

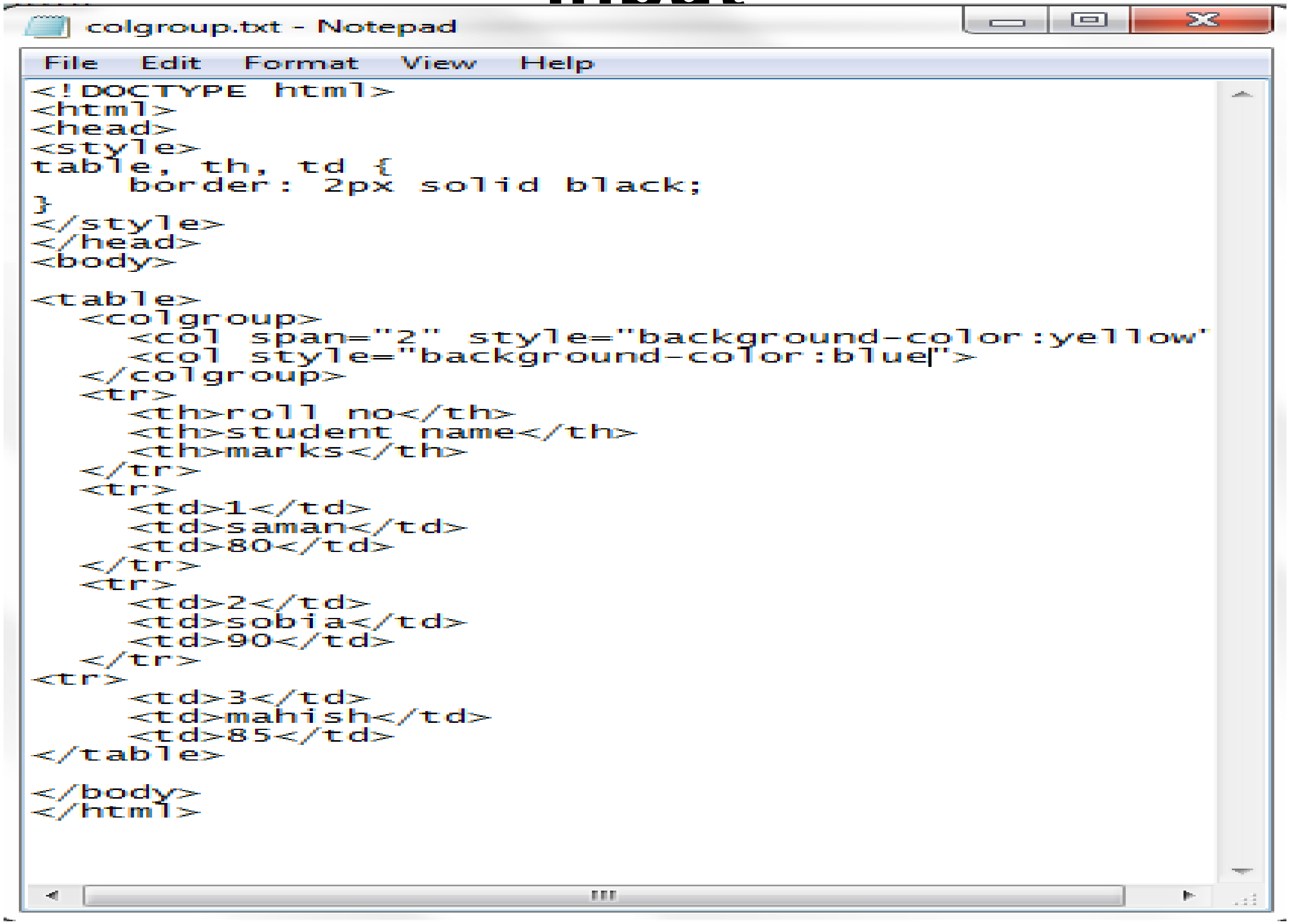
<col> Tag

- The <col> tag specifies column properties for each column within a <colgroup> element.
- The <col> tag is useful for applying styles to entire columns, instead of repeating the styles for each cell, for each row.

<colgroup> Tag

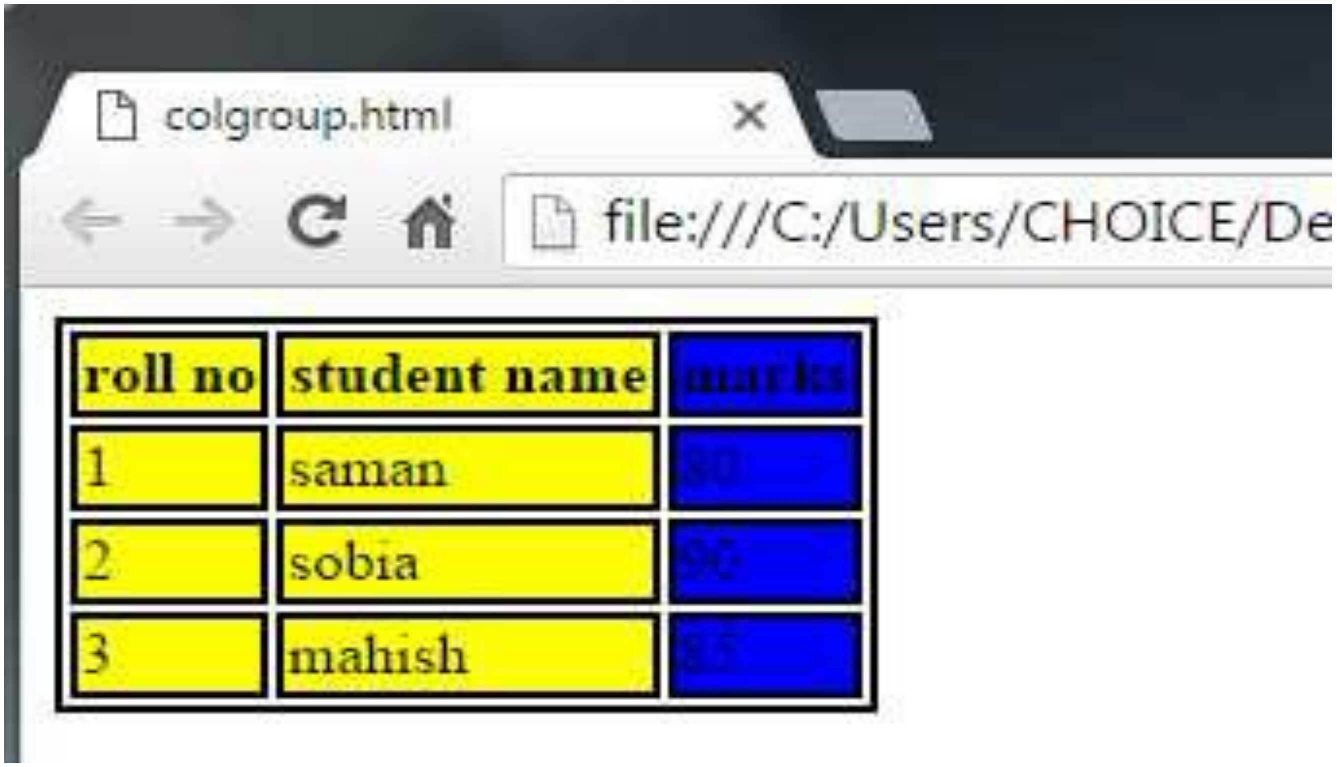
- The <colgroup> tag specifies a group of one or more columns in a table for formatting.
- The <colgroup> tag is useful for applying styles to entire columns, instead of repeating the styles for each cell, for each row.
- The <colgroup> tag must be a child of a <table> element, after any <caption> elements and before any <thead>, <tbody>, <tfoot>, and <tr> elements.
- To define different properties to a column within a <colgroup>, use the <col> tag within the <colgroup> tag.

input



```
<!DOCTYPE html>
<html>
<head>
<style>
table, th, td {
    border: 2px solid black;
}
</style>
</head>
<body>

<table>
  <colgroup>
    <col span="2" style="background-color:yellow"
    <col style="background-color:blue">
  </colgroup>
  <tr>
    <th>roll no</th>
    <th>student name</th>
    <th>marks</th>
  </tr>
  <tr>
    <td>1</td>
    <td>saman</td>
    <td>80</td>
  </tr>
  <tr>
    <td>2</td>
    <td>sobia</td>
    <td>90</td>
  </tr>
<tr>
    <td>3</td>
    <td>mahish</td>
    <td>85</td>
</table>

</body>
</html>
```

output

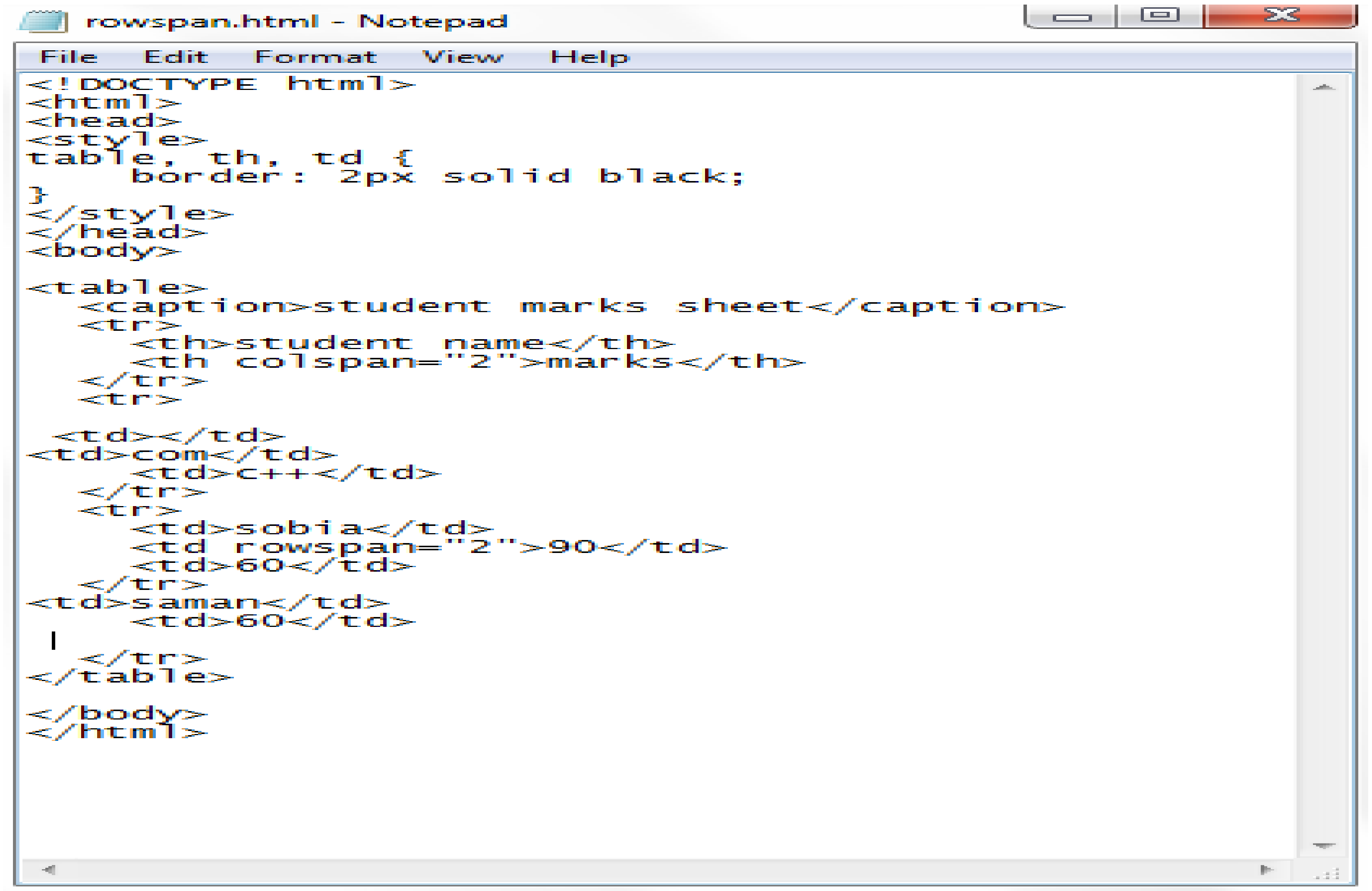


colgroup.html

file:///C:/Users/CHOICE/De

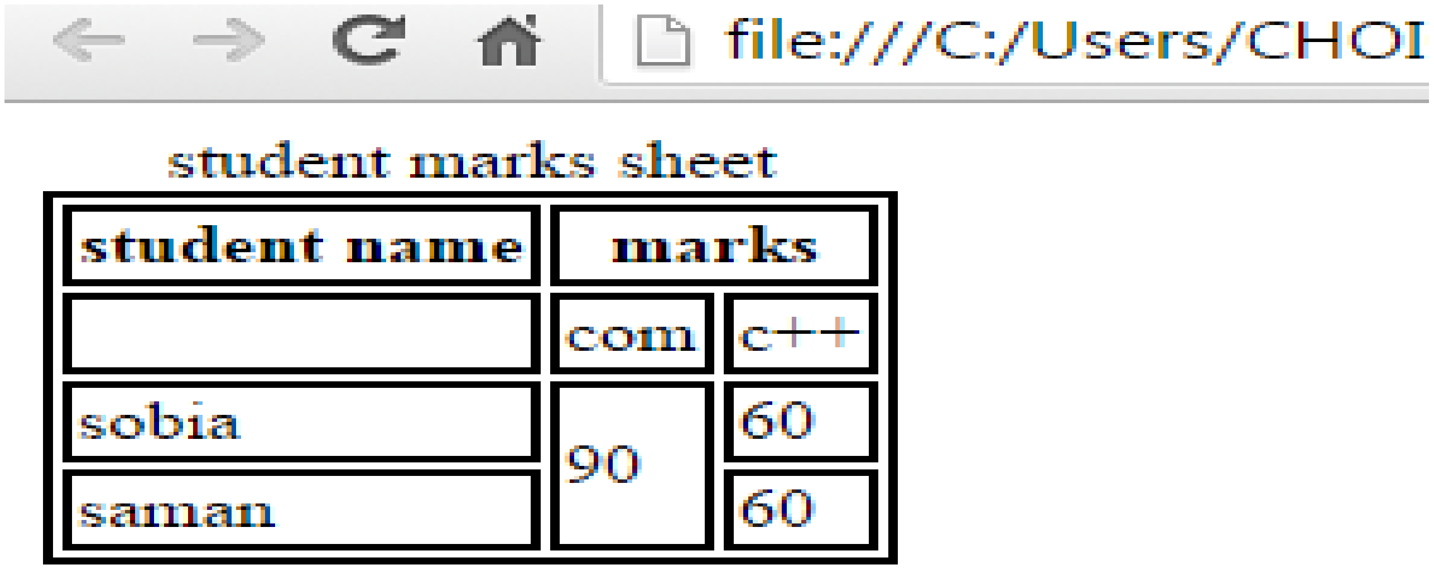
roll no	student name	marks
1	saman	80
2	sobia	90
3	mahish	85

Row span input



```
<!DOCTYPE html>
<html>
<head>
<style>
table, th, td {
        border: 2px solid black;
}
</style>
</head>
<body>

<table>
    <caption>student marks sheet</caption>
    <tr>
        <th>student name</th>
        <th colspan="2">marks</th>
    </tr>
    <tr>

  <td></td>
<td>com</td>
        <td>c++</td>
    </tr>
    <tr>
        <td>sobia</td>
        <td rowspan="2">90</td>
        <td>60</td>
    </tr>
<td>saman</td>
        <td>60</td>

    </tr>
</table>

</body>
</html>
```

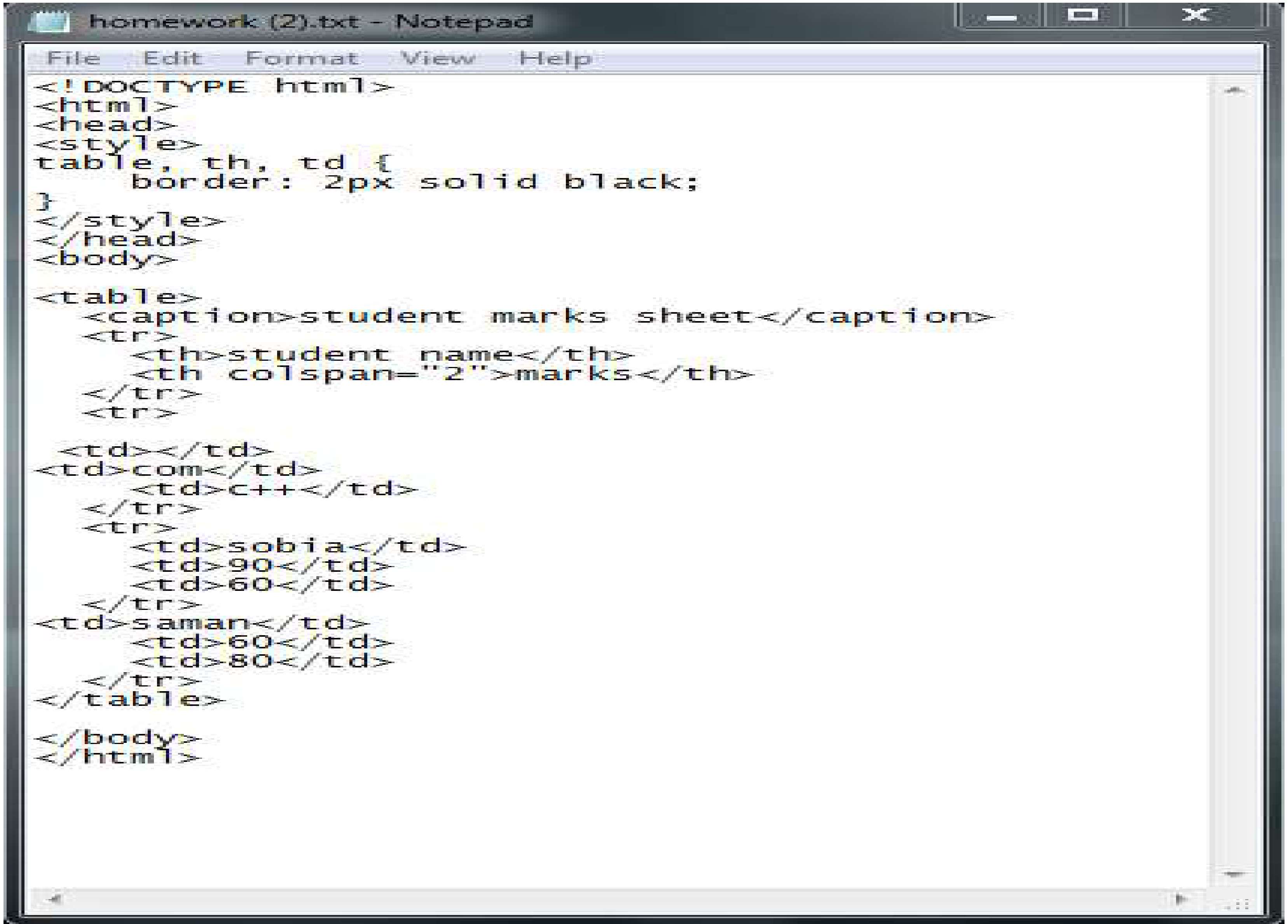
Rowspan output


file:///C:/Users/CHOI

student marks sheet

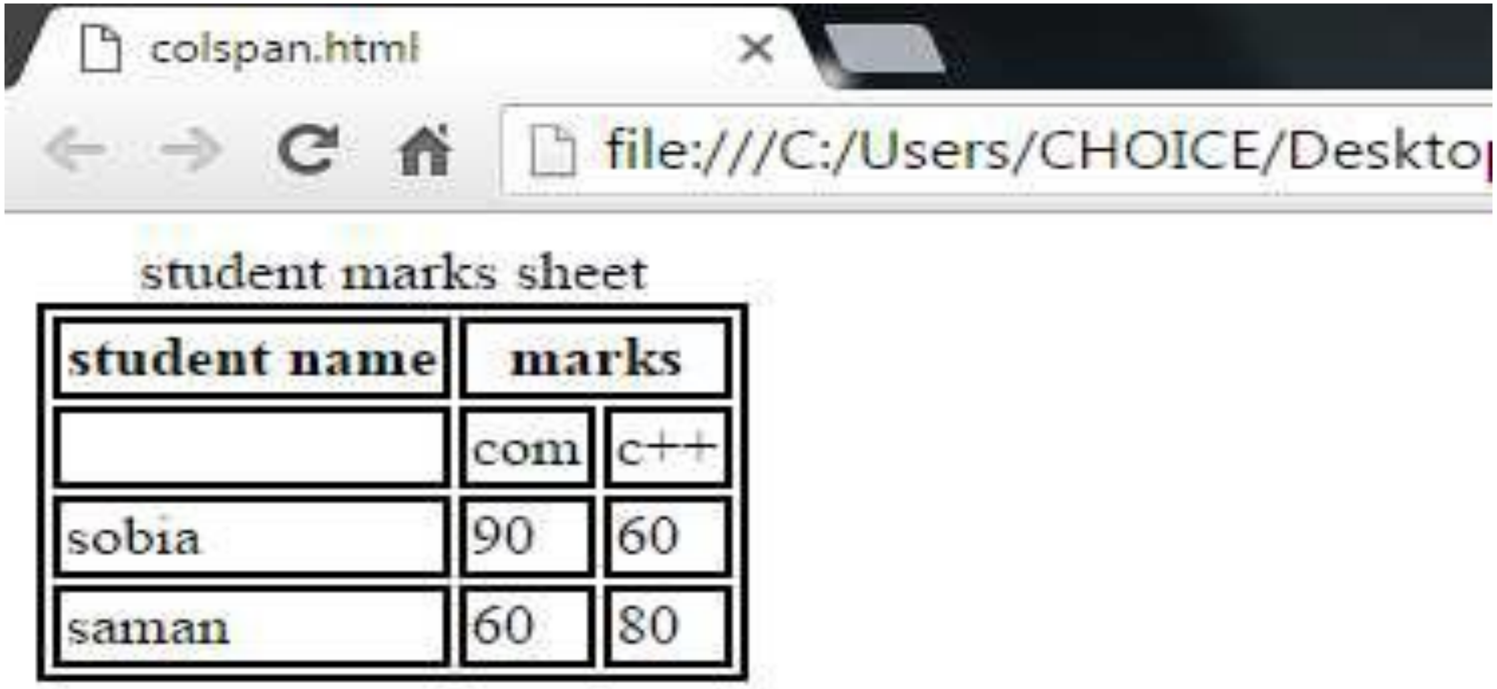
student name	marks	
	com	c++
sobia	90	60
saman		60

Colspan input



```
<!DOCTYPE html>
<html>
<head>
<style>
table, th, td {
    border: 2px solid black;
}
</style>
</head>
<body>

<table>
  <caption>student marks sheet</caption>
  <tr>
    <th>student name</th>
    <th colspan="2">marks</th>
  </tr>
  <tr>

 <td></td>
<td>com</td>
    <td>c++</td>
  </tr>
  <tr>
    <td>sobia</td>
    <td>90</td>
    <td>60</td>
  </tr>
<td>saman</td>
    <td>60</td>
    <td>80</td>
  </tr>
</table>

</body>
</html>
```

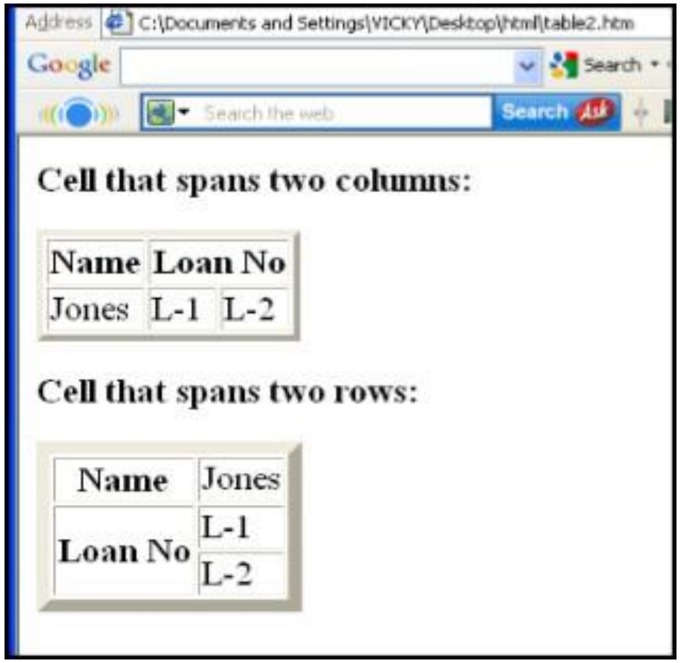
Colspan output



student marks sheet

student name	marks	
	com	c++
sobia	90	60
saman	60	80

Table Code with Colspan & Rowspan


Address C:\Documents and Settings\VICKY\Desktop\html\table2.htm
Google
Search
Search the web
Search

Cell that spans two columns:

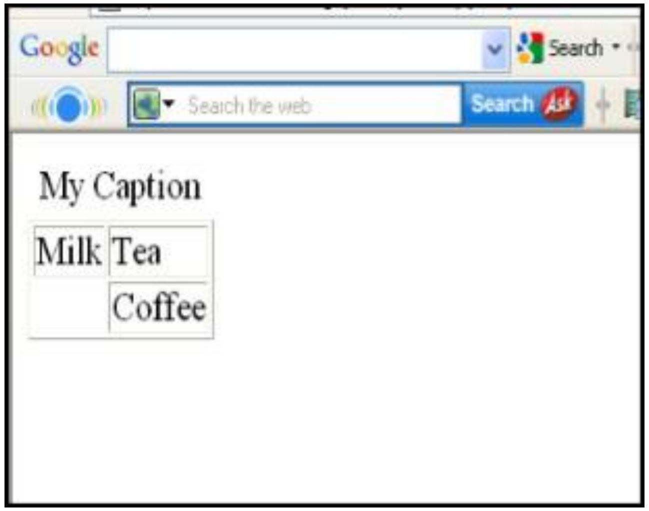
Name	Loan No	
Jones	L-1	L-2

Cell that spans two rows:

Name	Jones
Loan No	L-1
	L-2

```
<html><body>
<h4>Cell that spans two columns:</h4>
<table border="4">
<tr> <th>Name</th>
<th colspan="2">Loan No</th> </tr>
<tr> <td>Jones</td> <td>L-1</td>
<td>L-2</td> </tr> </table>
<h4>Cell that spans two rows:</h4>
<table border="8">
<tr> <th>Name</th>
<td>Jones</td></tr>
<tr> <th rowspan="2">Loan No</th>
<td>L-1</td></tr>
<tr> <td>L-2</td></tr> </table>
</body></html>
```

Table Code with Caption & ColSpacing

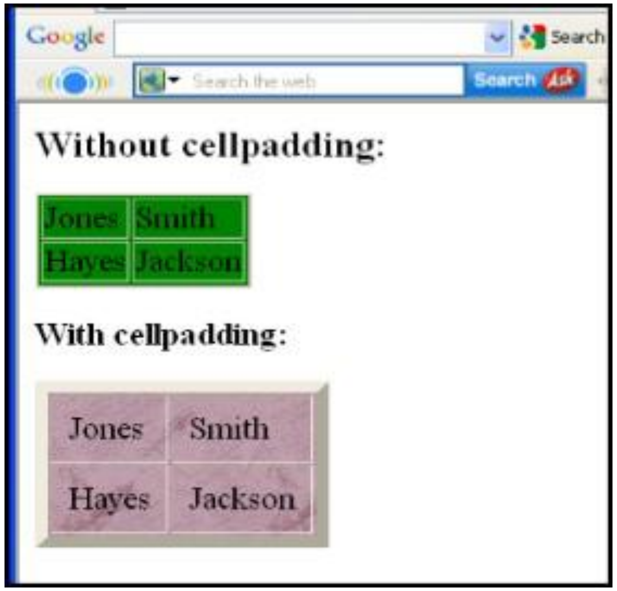


My Caption

Milk	Tea
	Coffee

```
<html>
<body>
<table border="1">
<caption>My Caption</caption>
<tr>
<td>Milk</td>
<td>Tea</td>
</tr>
<tr>
<td></td>
<td>Coffee</td>
</tr>
</table>
</body>
</html>
```

Cellpadding, Image & Backcolor Code



Without cellpadding:

Jones	Smith
Hayes	Jackson

With cellpadding:

Jones	Smith
Hayes	Jackson

```
<html><body>
<h3>Without cellpadding:</h3>
<table border="2" bgcolor="green">
<tr> <td>Jones</td>
<td>Smith</td></tr>
<tr> <td>Hayes</td>
<td>Jackson</td></tr></table>

<h4>With cellpadding:</h4>
<table border="8“ cellpadding="10"
    background="file:///C:/WINDOWS/
    FeatherTexture.bmp">
<tr> <td>Jones</td>
<td>Smith</td></tr>
<tr> <td>Hayes</td>
<td>Jackson</td></tr></table>
</body></html>
```

HTML List Tag

- Lists provide methods to show item or element sequences in document content.
- There are three main types of lists:->
  - **Unordered lists:-unordered lists are bulleted**
  - **Ordered lists:- Ordered lists are numbered**
  - **Definition lists:- Used to create a definition list**

List Tags

<LI> <LI> is an empty tag. it is used for representing the list items

<OL> Ordered list

<UL> Unordered list

<DL> Defines a definition list

<DT> Defines a term (an item) in a definition list

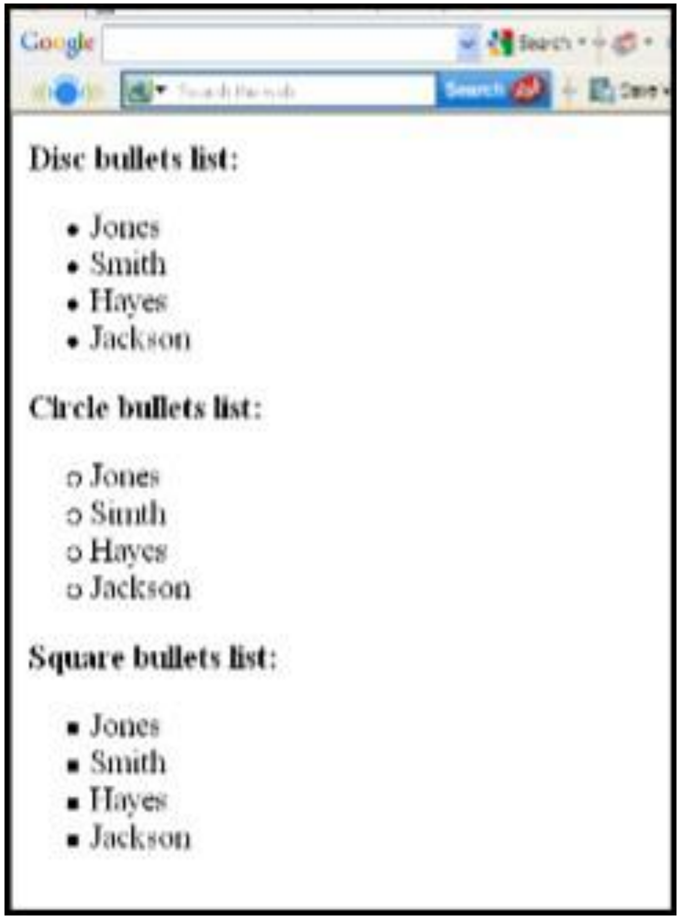
<DD> Defines a description of a term in a definition list

Unordered List

- TYPE attribute to the <UL> tag to show different bullets like:-
  - Disc
  - Circle
  - Square

    <ul Type =“disc”>…..</ul>
- The attribute TYPE can also be used with <LI> element.

Code & Result of the Unordered List



Disc bullets list:

- Jones
- Smith
- Hayes
- Jackson

Circle bullets list:

- Jones
- Smith
- Hayes
- Jackson

Square bullets list:

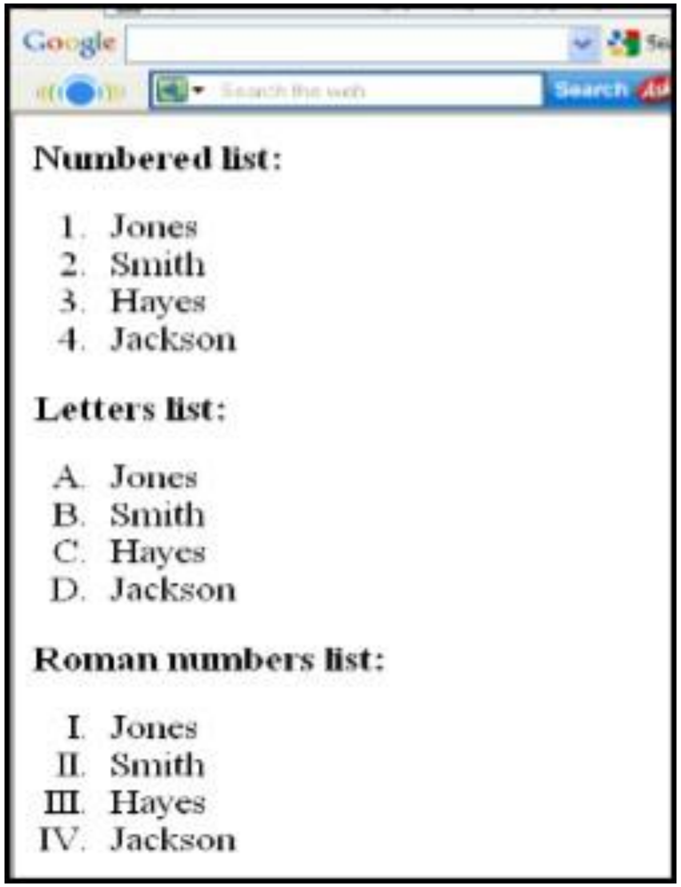
- Jones
- Smith
- Hayes
- Jackson

```
<html><body>
<h4>Disc bullets list:</h4>
<ul type="disc"> <li>Jones</li>
<li>Smith</li>
<li>Hayes</li>
<li>Jackson</li></ul>
<h4>Circle bullets list:</h4>
<ul type="circle"> <li>Jones</li>
<li>Simth</li>
<li>Hayes</li>
<li>Jackson</li></ul>
<h4>Square bullets list:</h4>
<ul type="square"> <li>Jones</li>
<li>Smith</li>
<li>Hayes</li>
<li>Jackson</li></ul>
</body></html>
```

Ordered List

- The TYPE attribute has the following value like:-
  - TYPE = "1" (Arabic numbers)
  - TYPE = "a" (Lowercase alphanumeric)
  - TYPE = "A" (Uppercase alphanumeric)
  - TYPE = "i" (Lowercase Roman numbers)
  - TYPE = "I" (Uppercase Roman numbers)
- By default **Arabic numbers are used**

Code & Result of the Ordered List


Numbered list:

1. Jones
2. Smith
3. Hayes
4. Jackson

Letters list:

A. Jones
B. Smith
C. Hayes
D. Jackson

Roman numbers list:

I. Jones
II. Smith
III. Hayes
IV. Jackson

```
<html><body>
<h4>Numbered list:</h4>
<ol> <li>Jones</li>
<li>Smith</li>
<li>Hayes</li>
<li>Jackson</li></ol>
<h4>Letters list:</h4>
<ol type="A"> <li>Jones</li>
<li>Smith</li>
<li>Hayes</li>
<li>Jackson</li></ol>
<h4>Roman numbers list:</h4>
<ol type="I"> <li>Jones</li>
<li>Smith</li>
<li>Hayes</li>
<li>Jackson</li></ol>
</body></html>
```

The start Attribute

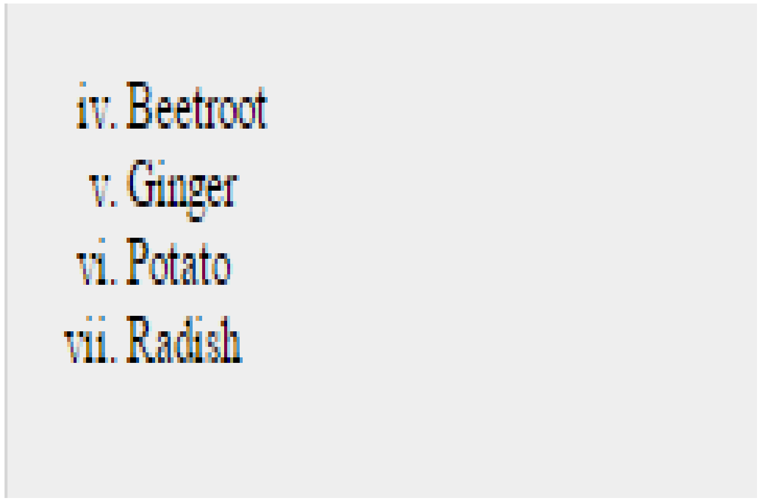
- You can use **start** attribute for <ol> tag to specify the starting point of numbering you need. Following are the possible options −

- <ol type = "1" start = "4"> - Numerals starts with 4.
  <ol type = "I" start = "4"> - Numerals starts with IV.
  <ol type = "i" start = "4"> - Numerals starts with iv.
  <ol type = "a" start = "4"> - Letters starts with d.
  <ol type = "A" start = "4"> - Letters starts with D.

Example

Following is an example where we used <ol type = "i" start = "4" >

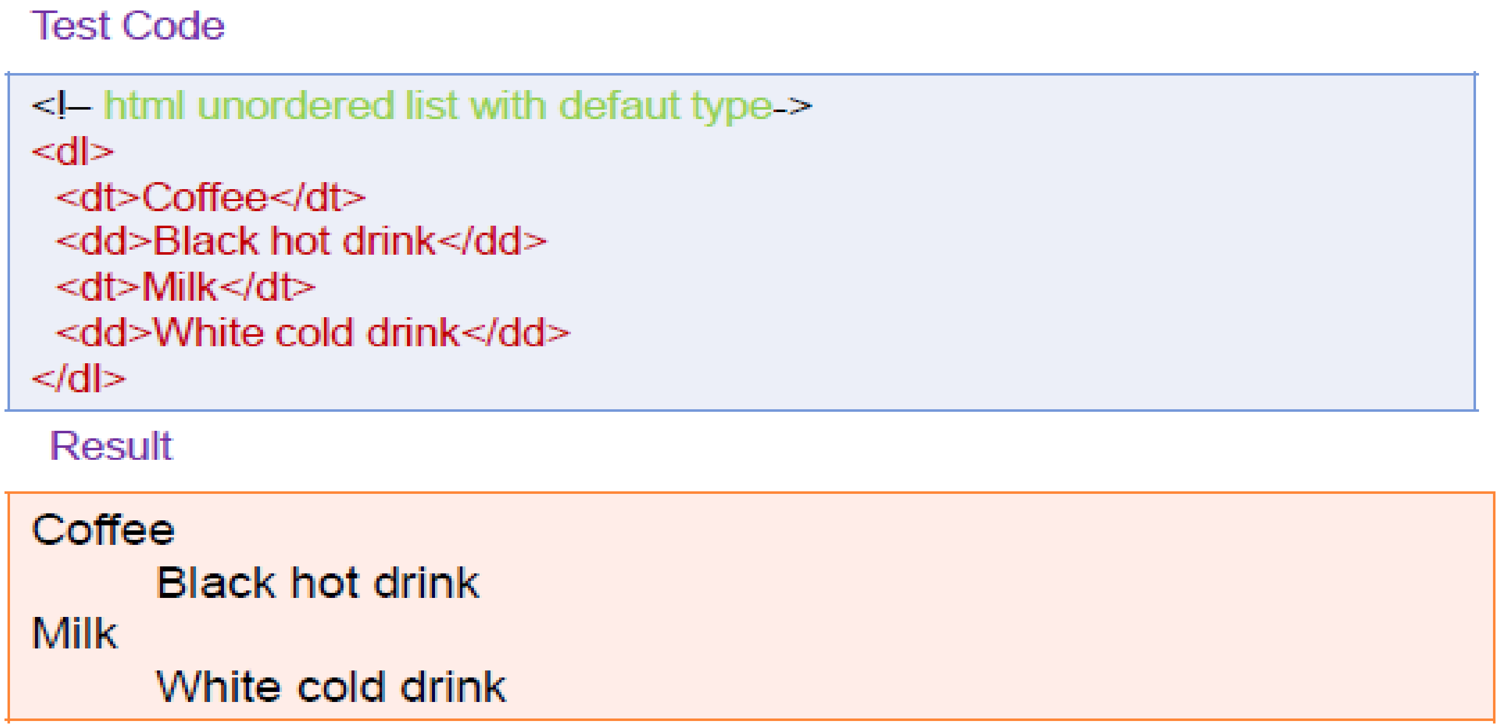
```
<!DOCTYPE html>
 <html>
    <head>
    <title>HTML Ordered List</title>
    </head>
    <body>
            <ol type = "i" start = "4">
                        <li>Beetroot</li>
                        <li>Ginger</li>
                        <li>Potato</li>
                         <li>Radish</li>
             </ol>
    </body>
</html>
```



```
 iv. Beetroot
  v. Ginger
 vi. Potato
vii. Radish
```

DEFINTION LIST

- Used for definitions. Like google define (ex: define: php).
- Starts with <dl> tag. Each term starts with <dt> tag.
- Description of term starts with <dd> tag.



Test Code

```
<!– html unordered list with defaut type–>
<dl>
  <dt>Coffee</dt>
  <dd>Black hot drink</dd>
  <dt>Milk</dt>
  <dd>White cold drink</dd>
</dl>
```

Result

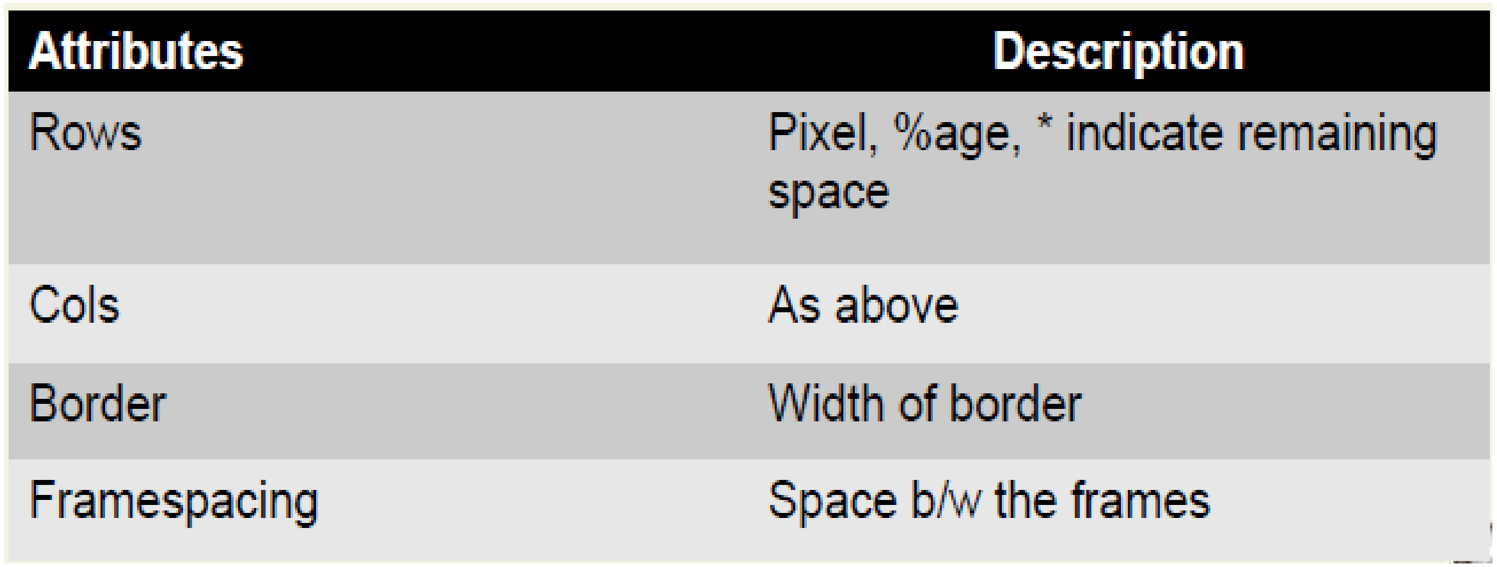
```
Coffee
        Black hot drink
Milk
        White cold drink
```

HTML FRAMES

- Frame is a container window that can display a web page; Many web pages can display; Break the page into different sections
  - Title
  - Navigational
  - Contents
  - Footer

FRAME Tags

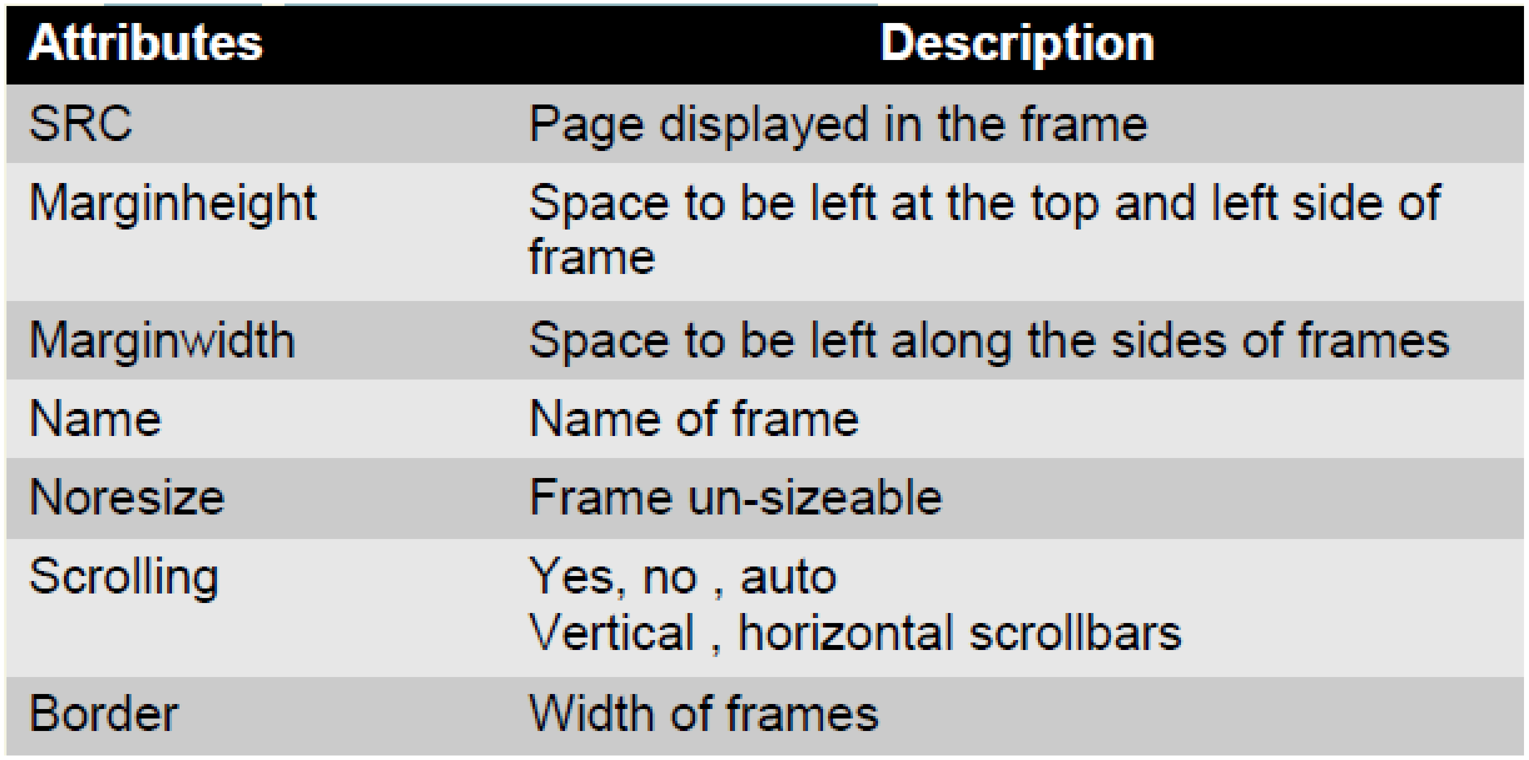
- <frameset>
- Define the style and no. of frames
- Nested <frameset> also allowed



Attributes	Description
Rows	Pixel, %age, * indicate remaining space
Cols	As above
Border	Width of border
Framespacing	Space b/w the frames

Tags contd…

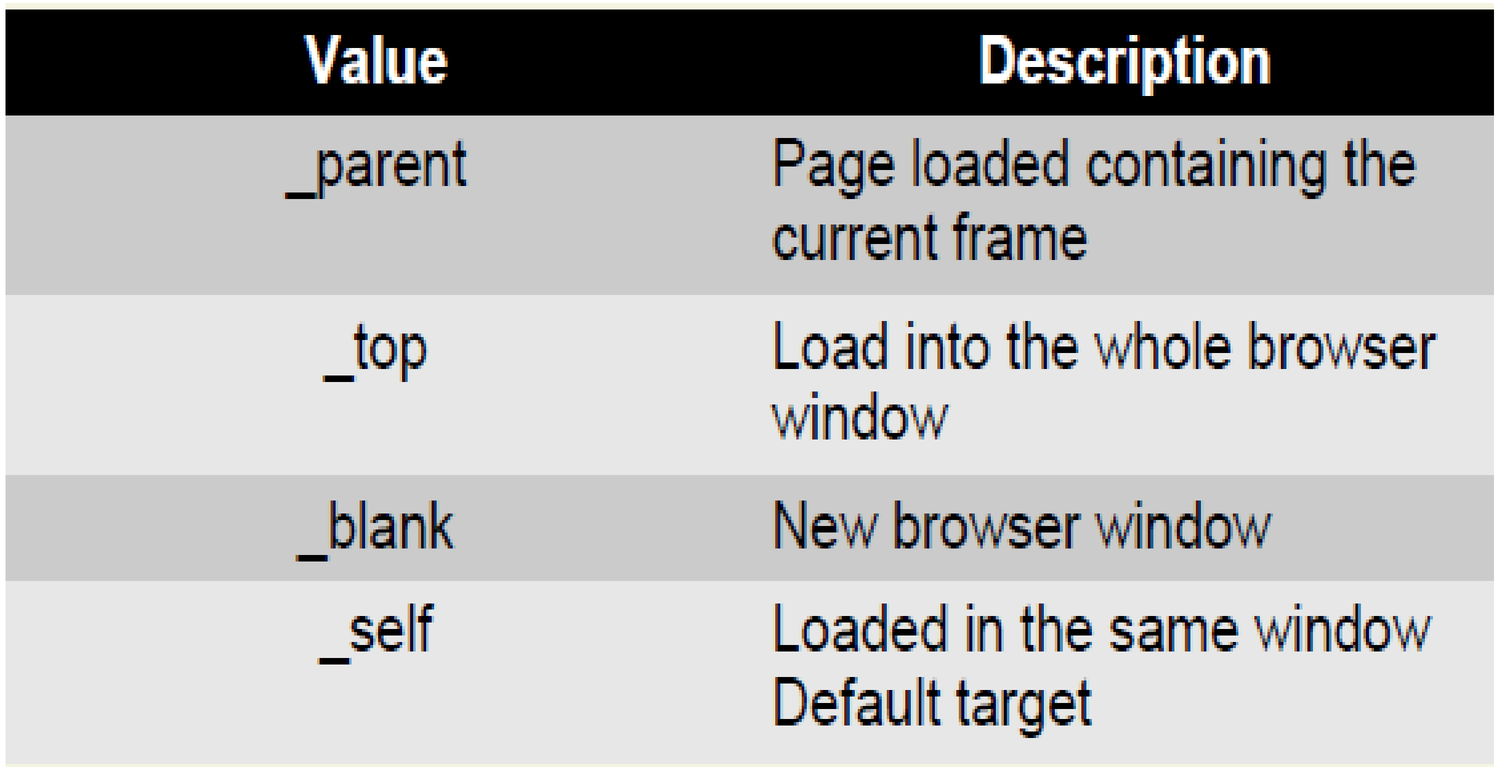
- <frame>
- Nested in frameset tag



Attributes	Description
SRC	Page displayed in the frame
Marginheight	Space to be left at the top and left side of frame
Marginwidth	Space to be left along the sides of frames
Name	Name of frame
Noresize	Frame un-sizeable
Scrolling	Yes, no , auto Vertical , horizontal scrollbars
Border	Width of frames

Tags contd….

- Target Frame



Value	Description
_parent	Page loaded containing the current frame
_top	Load into the whole browser window
_blank	New browser window
_self	Loaded in the same window Default target

Creating Frames - <frameset> Element

- The <frameset> tag replaces the <body> element in frameset documents.
- The <frameset> tag defines how to divide the window into frames.
- Each frameset defines a set of rows **or columns. If you** define frames by using rows then horizontal frames are created. If you define frames by using columns then vertical frames are created.
- The values of the rows/columns indicate the amount of screen area each row/column will occupy.
- Each frame is indicated by <frame> tag and it defines what HTML document to put into the frame.

The <frameset> Element Attributes

- **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 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.

The <frameset> Element Attributes

- **rows:** attribute works just like the cols attribute and can take 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.
- **border:** attribute specifies the width of the border of each frame in pixels. For example border="5". A value of zero specifies that no border should be there.
- **frameborder:** specifies whether a three-dimensional border should be displayed between frames. This attrubute takes value either 1 (yes) or 0 (no). For example frameborder="0" specifies no border.
- **framespacing:** 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.

Loading Content - <frame> Element

- The <frame> element indicates what goes in each frame of the frameset.
- The <frame> element is always an empty element, and therefore should not have any content, although each <frame> element should always carry one attribute, src, to indicate the page that should represent that frame.

The <frame> Element Attributes

- **src:** indicates the file that should be used in the frame. Its value can be any URL. For example, src="/html/top_frame.htm" will load an HTML file avaible in html directory.
- **name:** 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 a second frame, in which case the second frame needs a name to identify itself as the target of the link.
- **frameborder:** attribute specifies whether or not the borders of that frame are shown; it overrides the value given in the frameborder attribute on the <frameset> element if one is given, and the possible values are the same. This can take values either 1 (yes) or 0 (no).
- **marginwidth:** 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".

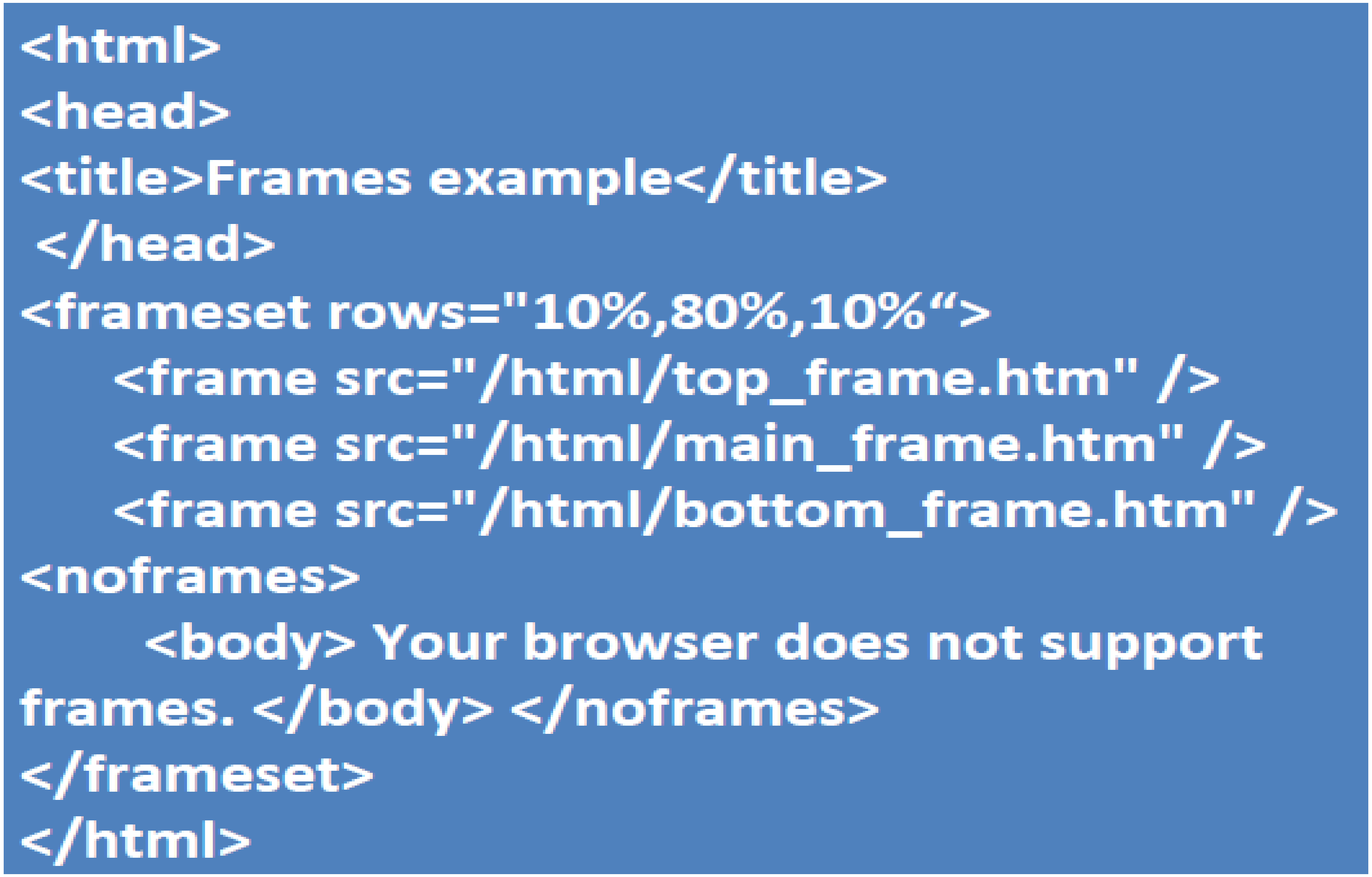
The <frame> Element Attributes

- **marginheight:** 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".
- **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".
- **scrolling:** 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.
- **longdesc:** allows you to provide a link to another page containing a long description of the contents of the frame. For example longdesc="framedescription.htm"

Browser Support - <noframes> Element

- If a user is using any old browser or any browser which does not support frames then <noframes> element should be displayed to the user.
- In XHTML you must place a <body> element inside the <noframes> element because the <frameset> element is supposed to replace the <body> element, but if a browser does not understand the <frameset> element it should understand what is inside the <body> element contained in the <noframes> element.
- You can put some nice message for your user having old browsers. For example *Sorry!! your browser does not support frames.*

Example-1



```
<html>
<head>
<title>Frames example</title>
 </head>
<frameset rows="10%,80%,10%“>
     <frame src="/html/top_frame.htm" />
     <frame src="/html/main_frame.htm" />
     <frame src="/html/bottom_frame.htm" />
<noframes>
       <body> Your browser does not support
frames. </body> </noframes>
</frameset>
</html>
```

Example-2

```
<frameset rows="20%, 70%,*">
        <frame src="header.html" name="header">

        <FRAMESET COLS="25%,*">

        <FRAME SRC="NAVIGATION.HTML"
NAME="NAVIGATION">
        <frame src="content.html" name="content">
        </FRAMESET>

        <FRAME SRC="FOOTER.HTML" NAME="footer">
</frameset>
```

“frame.html”

```
<html>
        <body>
<BR/><BR/><BR/>
                <a href="content.html" target="content"> Content1</a>
                <a href="content2.html" target="content">
Content2</a>
                <a href="content3.html" target="content">
Content3</a>
                <a href="content4.html" target="content">
Content4</a>
                <a href="content5.html" target="content">
Content5</a>
        </body>
</html>
```

"header.html"

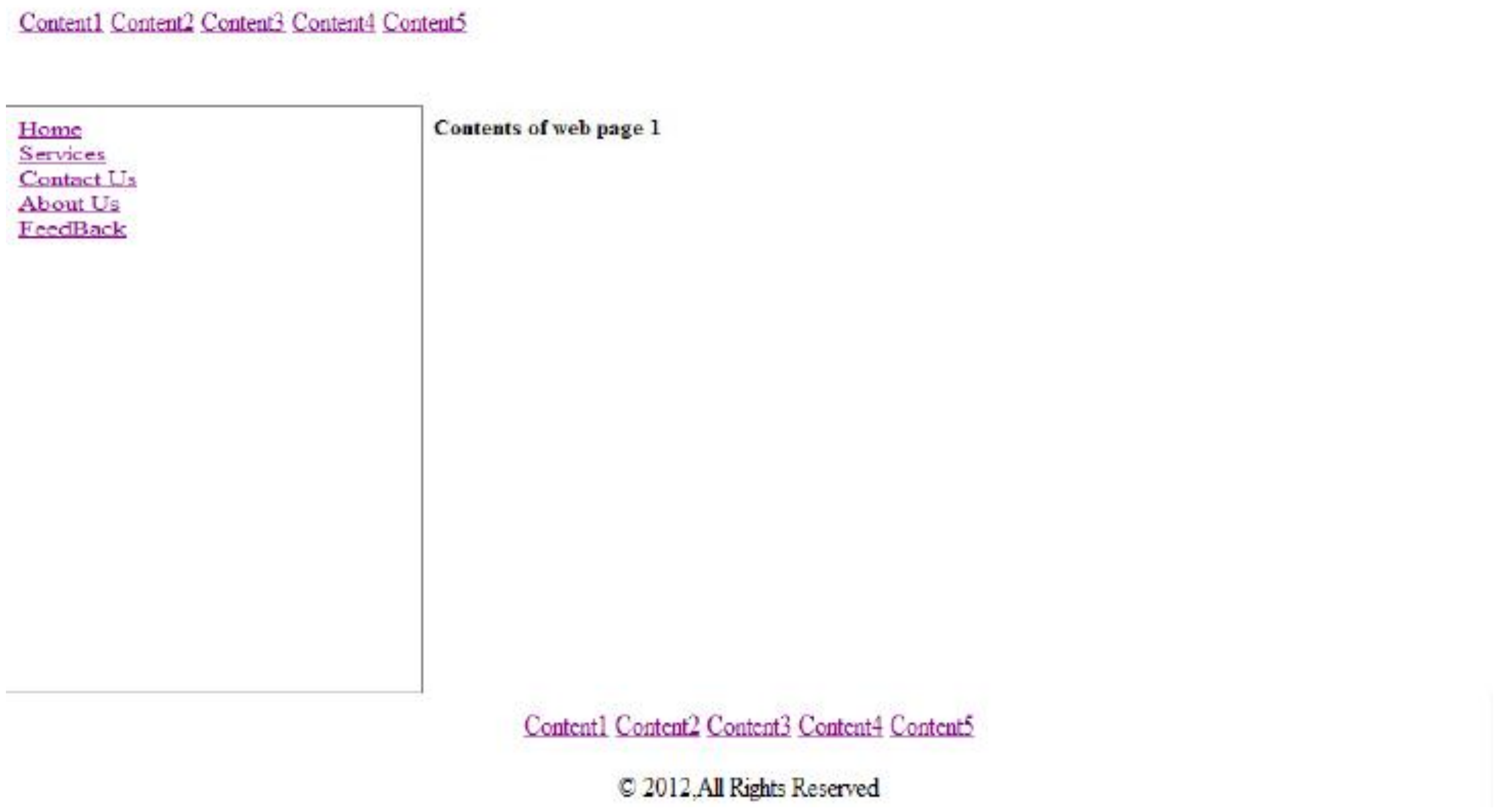
“navigation.html”

```
<html>
        <body>
                <a href="content.html" target="content"> Home</a><br/>
                <a href="content2.html" target="content"> Services</a><br/>
                <a href="content3.html" target="content"> Contact
Us</a><br/>
                <a href="content4.html" target="content"> About Us</a><br/>
                <a href="content5.html" target="content">
FeedBack</a><br/>
        </body>
</html>
```

“footer.html”

```
<html>
            <body>
<center>
                        <a href="content.html" target="content"> Content1</a>
                        <a href="content2.html" target="content"> Content2</a>
                        <a href="content3.html" target="content"> Content3</a>
                        <a href="content4.html" target="content"> Content4</a>
                        <a href="content5.html" target="content"> Content5</a>
                        <p>&copy 2012,All Rights Reserved</p>
</center>
            </body>
</html>
```

Output


Content1 Content2 Content3 Content4 Content5

Home
Services
Contact Us
About Us
FeedBack

Contents of web page 1

Content1 Content2 Content3 Content4 Content5

© 2012,All Rights Reserved

HTML Form

- A form is an area that can contain form elements.
- Form elements are elements that allow the user to enter information in a form like text fields, text area fields, drop-down menus, radio buttons and checkboxes etc
- A form is defined with the **<form> tag.**
- The syntax:-

<form>

.

input elements

.

</form>

Form Tags

<form> defines a form for user input

<input>used to create an input field

<text>Creates a single line text entry field

<textarea>Defines a text-area (a multi-line text input control)

<password>Creates a single line text entry field. And the characters entered are shown as asterisks (*)

<label>Defines a label to a control

<option>Creates a Radio Button.

<select>Defines a selectable list (a drop-down box)

<button>Defines a push button

<value>attribute of the option element.

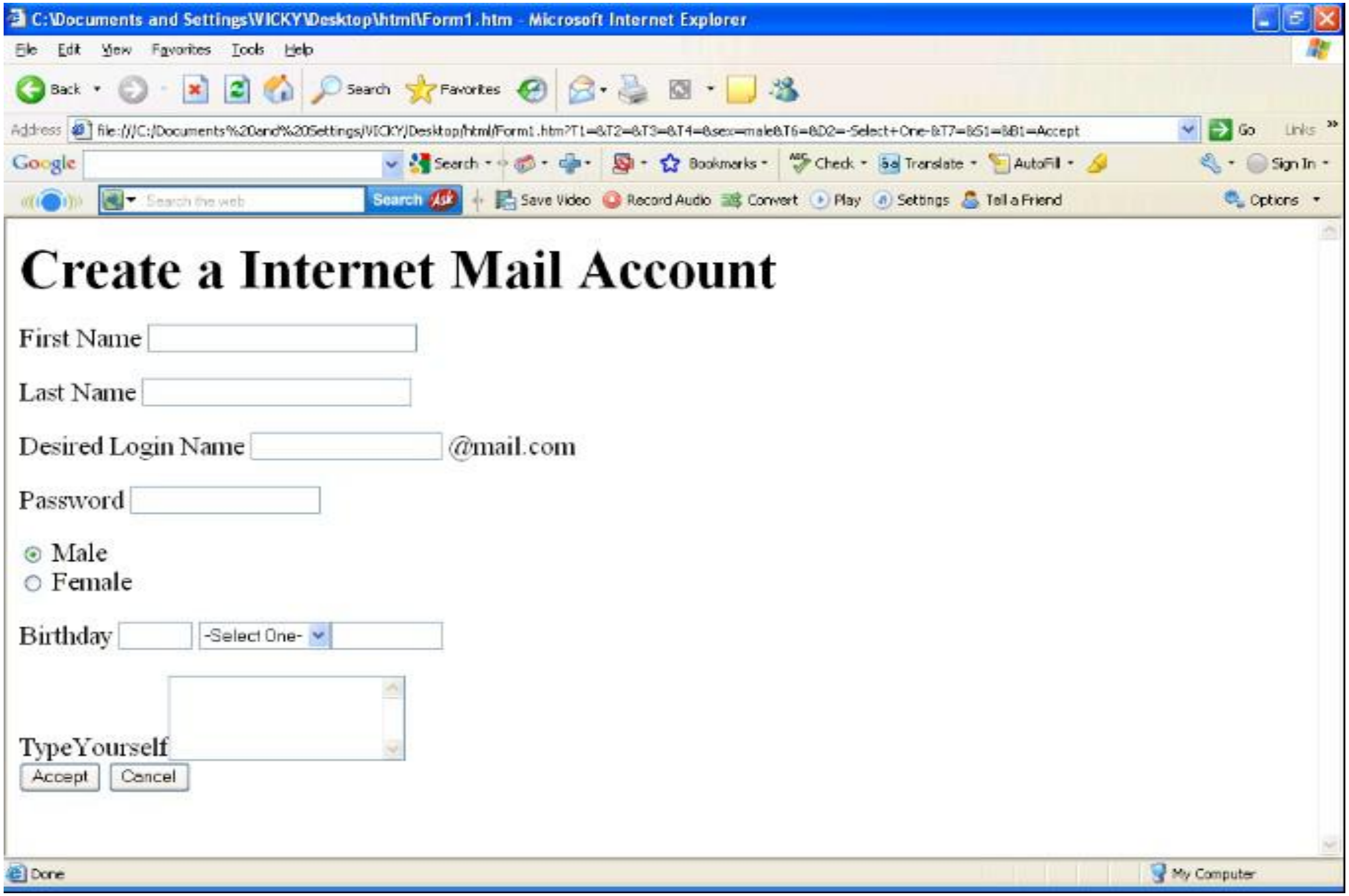
<checkbox>select or unselect a checkbox

<dropdown box>A drop-down box is a selectable list

Code of the HTML Form

```
<html><body><form>
<h1>Create a Internet Mail Account</h1>
<p>First Name <input type="text" name="T1" size="30"></p>
<p>Last Name <input type="text" name="T2" size="30"></p>
<p>Desired Login Name <input type="text" name="T3" size="20"> @mail.com</p>
<p>Password <input type="password" name="T4" size="20"></p>
<input type="radio" checked="checked" name="sex" value="male" /> Male</br>
<input type="radio" name="sex" value="female" /> Female
<p>Birthday <input type="text" name="T6" size="05">
<select size="1" name="D2">
<option>-Select One-</option>
<option>January</option>
<option>February</option>
<option>March</option> </select>
<input type="text" name="T7" size="10"></p>
TypeYourself<textarea rows="4" name="S1" cols="20"></textarea>
<br><input type="submit" value="Accept" name="B1"> <input
type="reset“ value="Cancel" name="B2"></br> </form></body></html>
```

Result of the Form Code


C:\Documents and Settings\VICKY\Desktop\html\Form1.htm - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address
Go Links
Google Search Bookmarks Check Translate AutoFill Sign In
Search the web Search Save Video Record Audio Convert Play Settings Tell a Friend Options
Done
My Computer

file:///C:/Documents%20and%20Settings/VICKY/Desktop/html/Form1.htm?T1=&T2=&T3=&T4=&sex=male&T6=&D2=-Select+One-&T7=&S1=&B1=Accept

Create a Internet Mail Account

First Name

Last Name

Desired Login Name @mail.com

Password

[✓ Male]
Female

Birthday -Select One-

TypeYourself

Accept Cancel

CSS-What is CSS?

- CSS stands for ***Cascading Style Sheet.*** Styles define **how to display HTML** elements
  - Typical CSS file is a text file with an extention**.css** and contains a series of commands or rules.
  - These rules tell the HTML how to display.

* *To create a style sheet, create a file using Notepad (PC) or Text Edit (Mac), save it as a .css document and start writing the CSS code (see right).*

Style.css

```
/* Styles for sitename.com*/
body {
font-family:Arial;
background: #000;
}
#container {
text-align:left;
width:1020px;
}
#header {
height:232px;
}
#footer {
width: 100%;
padding: 0 10px;
margin-bottom: 10px;
}
```

And so on….

CSS Versions

CSS has various levels and profiles. Each level of CSS builds upon the last, typically adding new features and typically denoted as CSS1, CSS2, and CSS3.

- **The first CSS** specification to become an official W3C Recommendation is CSS level 1, published in December 1996.
- **CSS level 2** was developed by the W3C and published as a Recommendation in May 1998. A superset of CSS1, CSS2 includes a number of new capabilities like absolute, relative, and fixed positioning of elements and z-index, the concept of media types etc.
- **CSS 3**

  CSS lev el 3 is currently under development. The W3C maintains a CSS3 progress report.

Why to Learn CSS?

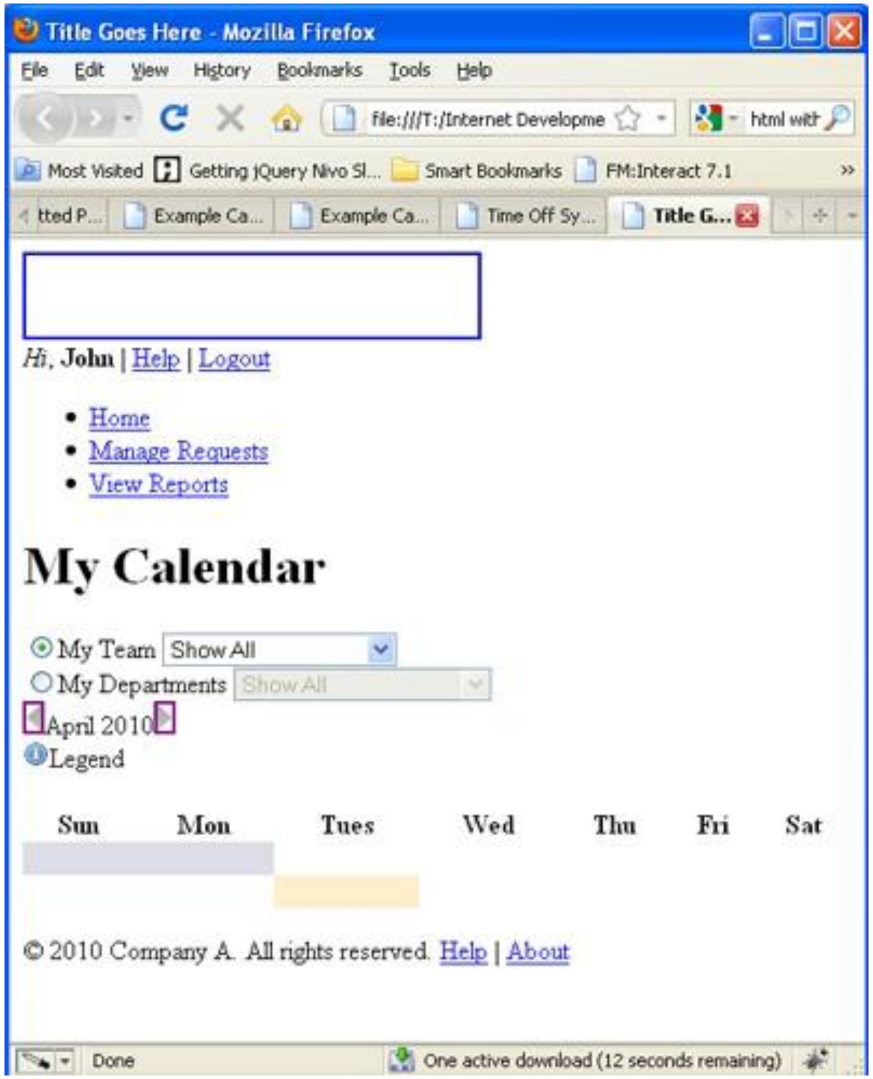
- **Cascading Style Sheets**, fondly referred to as **CSS**, is a simple design language intended to simplify the process of making web pages presentable.

- **CSS** is a MUST for students and working professionals to become a great Software Engineer specially when they are working in Web Development Domain. I will list down some of the key advantages of learning CSS:
  - **Create Stunning Web site** - CSS handles the look and feel part of a web page. Using CSS, you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colors are used, layout designs, variations in display for different devices and screen sizes as well as a variety of other effects.
  - **Become a web designer** - If you want to start a carrer as a professional web designer, HTML and CSS designing is a must skill.
  - **Control web** - CSS is easy to learn and understand but it provides powerful control over the presentation of an HTML document. Most commonly, CSS is combined with the markup languages HTML or XHTML.
  - **Learn other languages** - Once you understands the basic of HTML and CSS then other related technologies like javascript, php, or angular are become easier to understand.

Applications of CSS

- **CSS saves time** - You can write CSS once and then reuse same sheet in multiple HTML pages. You can define a style for each HTML element and apply it to as many Web pages as you want.
- **Pages load faster** - If you are using CSS, you do not need to write HTML tag attributes every time. Just write one CSS rule of a tag and apply it to all the occurrences of that tag. So less code means faster download times.
- **Easy maintenance** - To make a global change, simply change the style, and all elements in all the web pages will be updated automatically.
- **Superior styles to HTML** - CSS has a much wider array of attributes than HTML, so you can give a far better look to your HTML page in comparison to HTML attributes.
- **Multiple Device Compatibility** - Style sheets allow content to be optimized for more than one type of device. By using the same HTML document, different versions of a website can be presented for handheld devices such as PDAs and cell phones or for printing.
- **Global web standards** - Now HTML attributes are being deprecated and it is being recommended to use CSS. So its a good idea to start using CSS in all the HTML pages to make them compatible to future browsers.

HTML Without CSS


Title Goes Here - Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///T:/Internet Developme
html with
Most Visited
Getting jQuery Nivo Sl...
Smart Bookmarks
FM:Interact 7.1
tted P...
Example Ca...
Example Ca...
Time Off Sy...
Title G...
Done
One active download (12 seconds remaining)

Hi, John | Help | Logout

- Home
- Manage Requests
- View Reports

My Calendar

My Team Show All
My Departments Show All
April 2010
Legend

Sun	Mon	Tues	Wed	Thu	Fri	Sat

© 2010 Company A. All rights reserved. Help | About

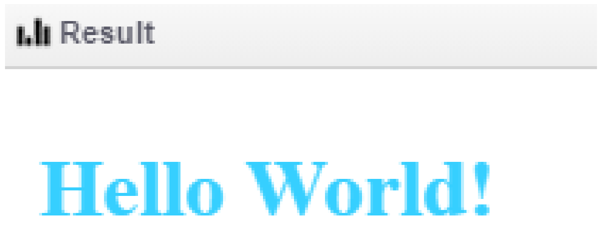
- *"HTML without CSS is like a piece of candy without a pretty wrapper."*
- **Without CSS, HTML elements typically flow from top to bottom of the page and position themselves to the left by default.**
- **With CSS help, we can create containers or DIVs to better organize content and make a Web page visually appealing.**

HTML & CSS

- HTML and CSS work together to produce beautiful and functional Web sites
- HTML = structure
- CSS = *style*

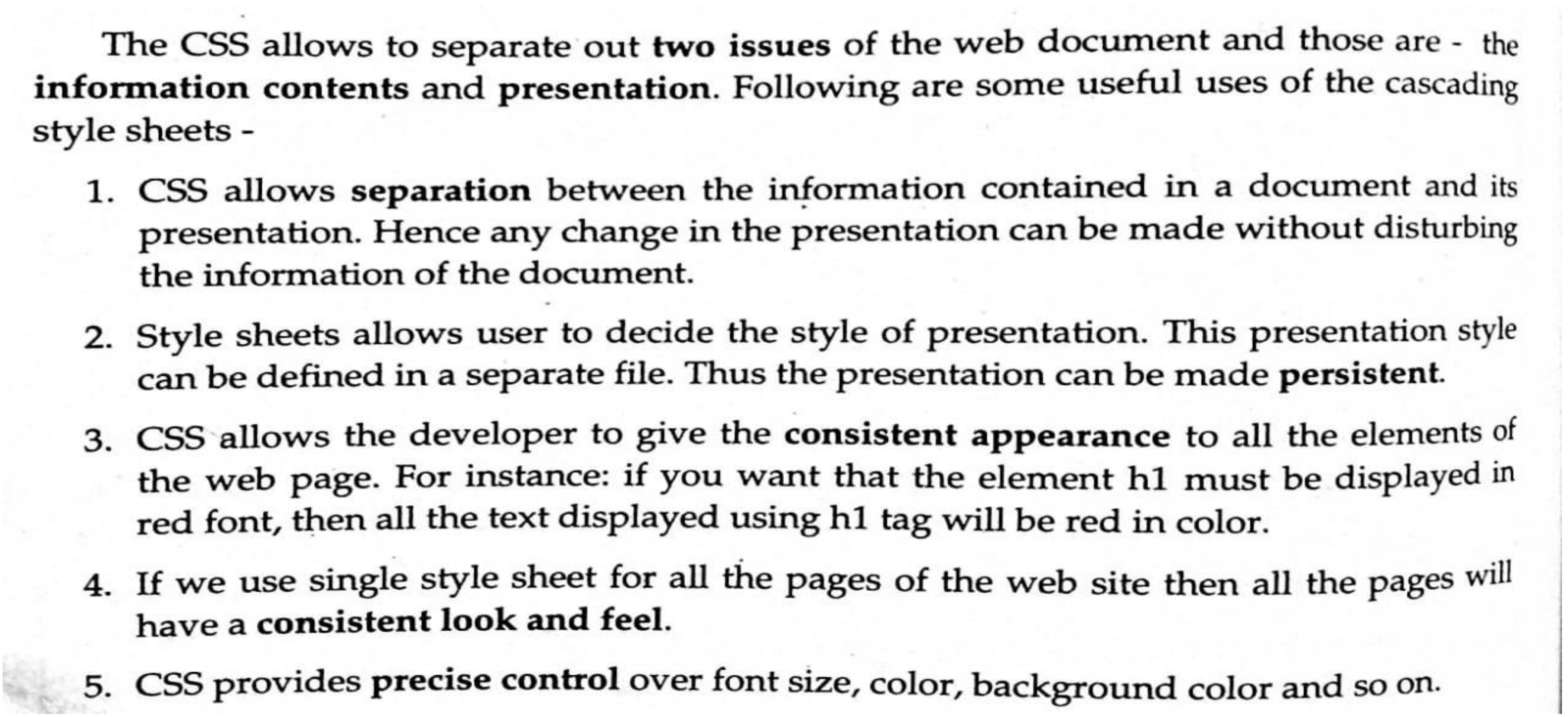
Hello World using CSS.

```
<!DOCTYPE html>
<html>
    <head>
          <title>This is document title</title>
          <style>
          h1  {
               color: #36CFFF;
               }
          </style>
    </head>
    <body>
          <h1>Hello World!</h1>
    </body>
</html>
```



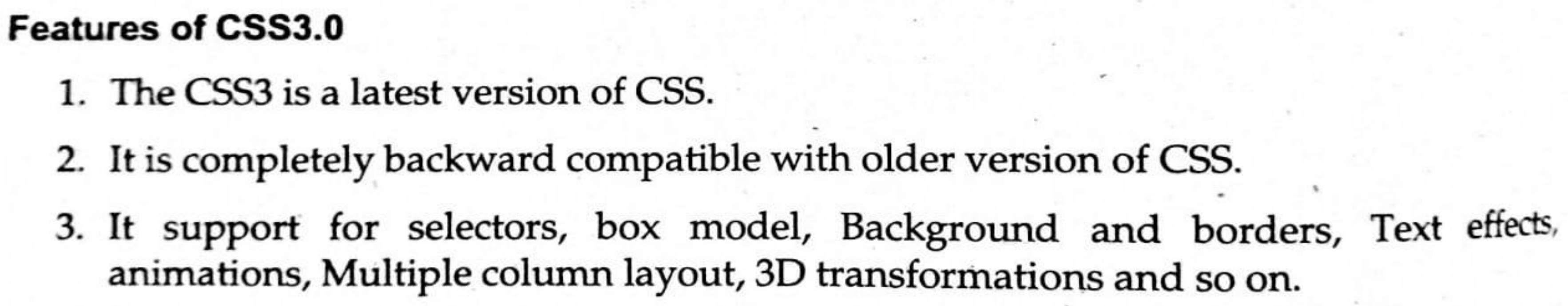
Hello World!

Features of CSS



The CSS allows to separate out **two issues** of the web document and those are - the **information contents** and **presentation**. Following are some useful uses of the cascading style sheets -

1. CSS allows **separation** between the information contained in a document and its presentation. Hence any change in the presentation can be made without disturbing the information of the document.
2. Style sheets allows user to decide the style of presentation. This presentation style can be defined in a separate file. Thus the presentation can be made **persistent**.
3. CSS allows the developer to give the **consistent appearance** to all the elements of the web page. For instance: if you want that the element h1 must be displayed in red font, then all the text displayed using h1 tag will be red in color.
4. If we use single style sheet for all the pages of the web site then all the pages will have a **consistent look and feel**.
5. CSS provides **precise control** over font size, color, background color and so on.

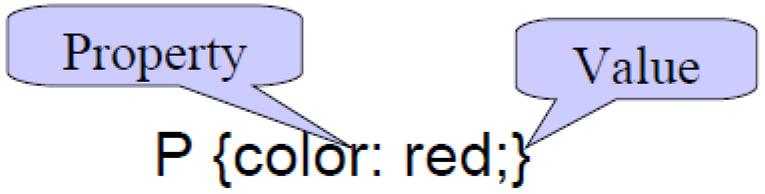


Features of CSS3.0

1. The CSS3 is a latest version of CSS.
2. It is completely backward compatible with older version of CSS.
3. It support for selectors, box model, Background and borders, Text effects, animations, Multiple column layout, 3D transformations and so on.

CSS - Syntax

- A CSS comprises of style rules that are interpreted by the browser and then applied to the corresponding elements in your document. A style rule is made of three parts –
  - **Selector** − A selector is an HTML tag at which a style will be applied. This could be any tag like <h1> or <table> etc.
  - **Property** − A property is a type of attribute of HTML tag. Put simply, all the HTML attributes are converted into CSS properties. They could be *color*, *border* etc.
  - **Value** − Values are assigned to properties. For example, *color* property can have value either *red* or *#F1F1F1* etc.

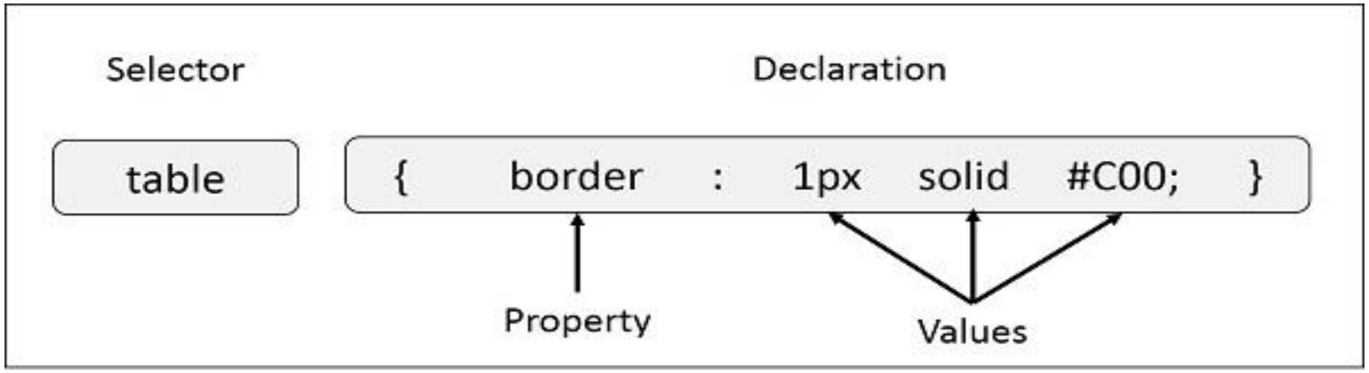


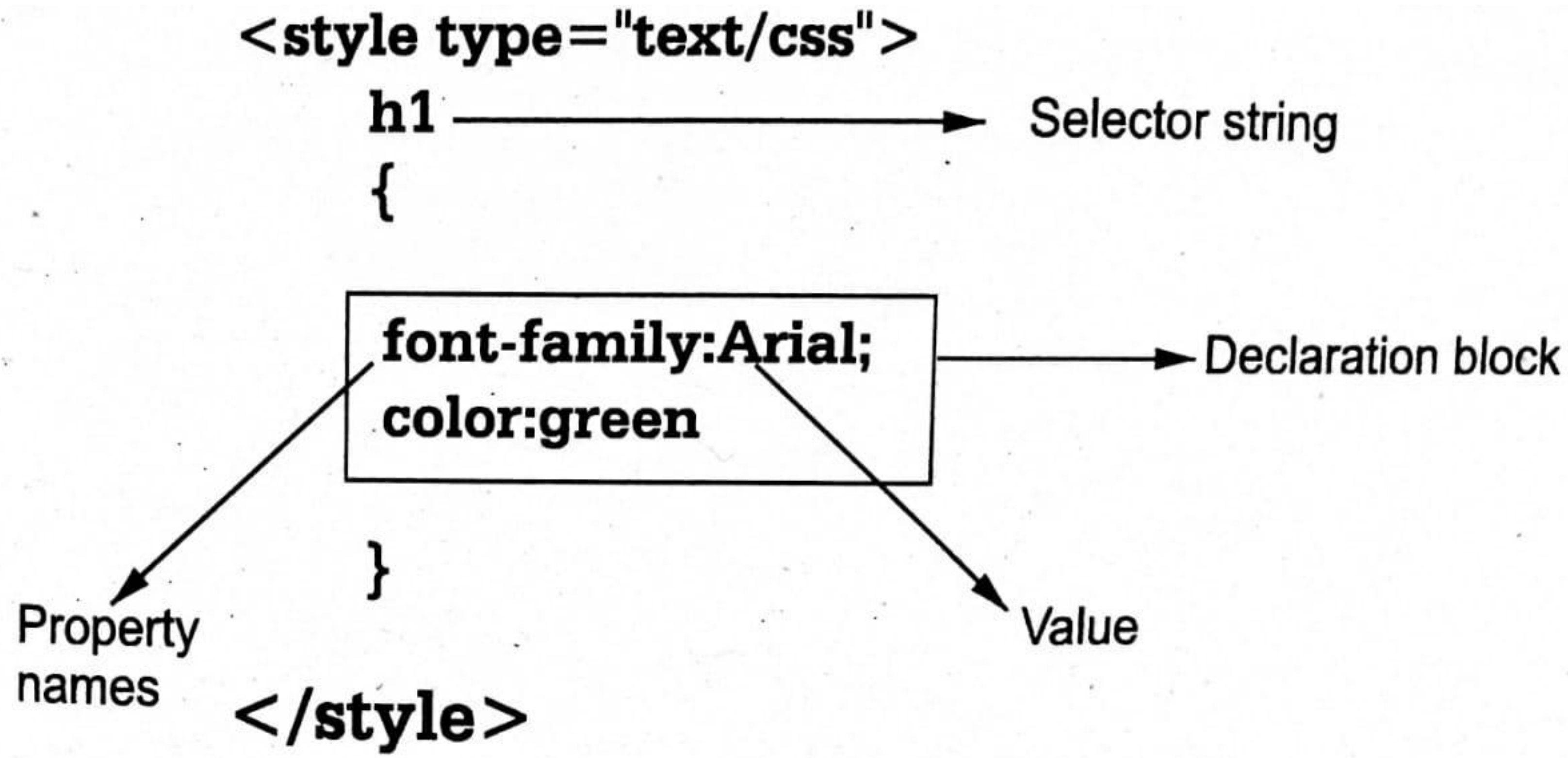
Understanding Style Rules

- A Style Rule is composed of two parts: **a selector string** and **a declaration block**.
- CSS Style Rule Syntax as follow:

```
selector { property: value;}
```

- The Selector indicates the element to which the rule is applied.
- The Declaration determines the property values of a selector.


Selector
Declaration
table
{ border : 1px solid #C00; }
Property
Values


<style type="text/css">
h1
Selector string
{
font-family:Arial;
color:green
Declaration block
}
Property names
Value
</style>

CSS declarations always ends with a semicolon, and declaration groups are surrounded by curly brackets:

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

To make the CSS more readable, you can put one declaration on each line, like this:

Example

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

CSS Comments

A CSS comment begins with "/*", and ends with "*/", like this:

```
/*This is a comment*/
p{
   text-align:center;
    /*This is another comment*/
   color:black;
   font-family:arial;
  }
```

CSS - Measurement Units

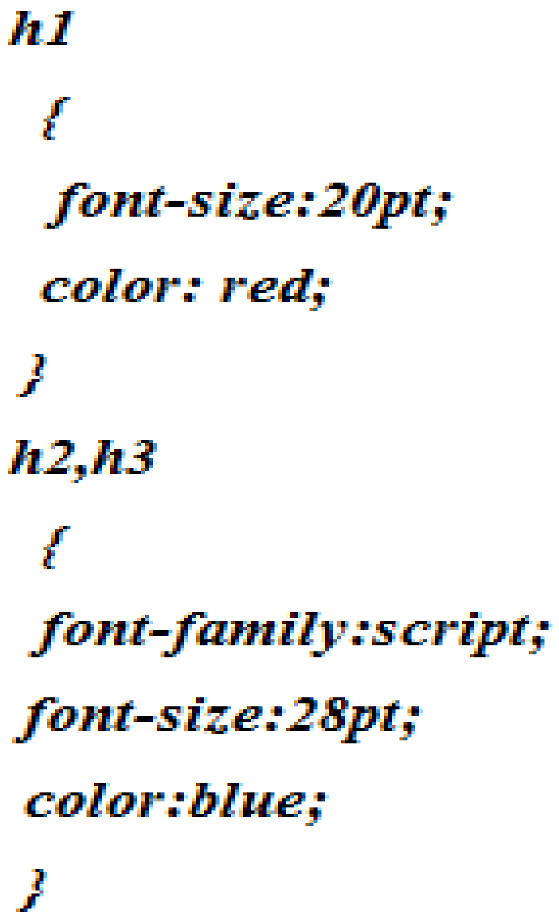
Unit	Description	Example
%	Defines a measurement as a percentage relative to another value, typically an enclosing element.	p {font-size: 16pt; line-height: 125%;}
cm	Defines a measurement in centimeters.	div {margin-bottom: 2cm;}
em	A relative measurement for the height of a font in em spaces. Because an em unit is equivalent to the size of a given font, if you assign a font to 12pt, each "em" unit would be 12pt; thus, 2em would be 24pt.	p {letter-spacing: 7em;}
ex	This value defines a measurement relative to a font's x-height. The x-height is determined by the height of the font's lowercase letter x.	p {font-size: 24pt; line-height: 3ex;}
in	Defines a measurement in inches.	p {word-spacing: .15in;}
mm	Defines a measurement in millimeters.	p {word-spacing: 15mm;}
pc	Defines a measurement in picas. A pica is equivalent to 12 points; thus, there are 6 picas per inch.	p {font-size: 20pc;}
pt	Defines a measurement in points. A point is defined as 1/72nd of an inch.	body {font-size: 18pt;}
px	Defines a measurement in screen pixels.	p {padding: 25px;}

CSS Selectors

- There are 5 varieties of CSS Selectors available for us. We will be looking at the following important CSS Selectors:
  1. CSS Universal Selector.
  2. CSS Element Selector.
  3. CSS Id Selector.
  4. CSS Class Selector.
  5. CSS Attribute Selector.

Simple Selector

- Simple selector form is a single element to which the property and value is applied.



```
h1
 {
  font-size:20pt;
  color: red;
}
h2,h3
 {
  font-family:script;
 font-size:28pt;
 color:blue;
 }
```

Universal Selector

In an HTML page, the content depends on HTML tags. A pair of tags defines a specific webpage element. The CSS universal selector selects all the elements on a webpage.

```
* {
color: blue;
font-size: 21px;
}
```

These two lines of code surrounded by the curly braces will affect all the elements present on the HTML page. We declare a universal selector with the help of an asterisk at the beginning of the curly brace. Universal Selector can be used along with the other selectors in combination.

CSS Element Selector

CSS Element Selector is also known as a Type selector. Element Selector in CSS tries to match one or more HTML elements having the same name. Therefore, a selector of <ul> matches all the <ul> elements i.e. all the unordered lists in that HTML page.

```
ul
{ list-style: none;
border: solid 1px #ccc;
}
```

```
<ul>
<li>Fish</li>
<li>Apples</li>
<li>Cheese</li>
</ul>
<div class="example">
<p>Example paragraph
text.</p>
</div>
<ul>
<li>Water</li>
<li>Juice</li>
<li>Maple Syrup</li>
</ul>
```

There are three main elements making up this part of the page:
Two `<ul>` elements and a `<div>`.
The CSS will apply only to the two `<ul>` elements, and not to the `<div>`.
Were we to change the element type selector to use `<div>` instead of `<ul>`, then the styles would apply to the `<div>` and not to the two `<ul>` elements.
Also note that the styles will not apply to the elements inside the `<ul>` or `<div>` elements.

ID Selector

An ID selector is declared using a hash, or pound symbol (#) preceding a string of characters. The string of characters is defined by the developer. This selector matches any HTML element that has an ID attribute with the same value as that of the selector, but minus the hash symbol.

This CSS uses an ID selector to match an HTML element such as:

```
#container {
width: 960px;
margin: 0 auto;
}
```

```
<div id="container"></div>
```

In this case, the fact that this is a `<div>` element doesn’t matter— it could be any kind of HTML element. As long as it has an ID attribute with a value of `container`, the styles will apply. An ID element on a web page should be unique. That is, there should only be a single element on any given page with an ID of `container`. This makes the ID selector quite inflexible, because the styles used in the ID selector rule set can be used only once per page. If there happens to be more than one element on the page with the same ID, the styles will still apply, but the HTML on such a page would be invalid from a technical standpoint, so you’ll want to avoid doing this.

CSS Class Selector

The CSS Class selector is one of the most helpful selectors of all the selectors. It is declared by using a dot followed by the name of the class. This class name is defined by the coder, as is the case with the ID selector. The class selector searches for every element having an attribute value with the same name as the class name, without the dot.

```
.square {
margin: 20px;
width: 20px;
}
```

This CSS code can be used to match the element having the class ‘square, like in the following sentence.

```
<div class="square"></div>
```

This style also applies to all the other HTML elements having an attribute value for the class as ‘square’.
An element having the same class attribute value helps us in reusing the styles and avoids unnecessary repetition.
Additionally, the Class Selector is beneficial because it permits us to add several classes to a particular element.
We can add more than one class to the attribute by separating each class with space.
Example:

```
<div class=”square bold shape”></div>
```

Here square, bold and shape are three different types of classes.

CSS Attribute Selector

The CSS Attribute selector styles content according to the attribute and the attribute value mentioned in the square brackets. No spaces can be present ahead of the opening square bracket.

```
input[type="text"] {
background-color:
#fff;
width: 100px;
}
```

This CSS code would be a match for the following HTML element.

```
        <input type="text">
```

Similarly, if the value of the attribute ‘type’ changes, the Attribute selector would not match it.
For example, the selector would not match the attribute if the value of ‘type’ changed from ‘text’ to ‘submit’.
If the attribute selector is declared with only the attribute, and no attribute value, then it will match to all the
HTML elements with the attribute ‘type’, regardless of whether the value is ‘text’ or ‘submit’.

Example:

```
    input[type] {
    background-color: #fff;
    width: 100px;
    }
```

We can also make use of attribute selectors with no specification of a value outside the square brackets.

This will help us to target the attribute only, regardless of the element. In our example, it will target based on the attribute ‘type’, regardless of the element ‘input’.

CSS Selectors help us to simplify our code and enable us to utilize and reuse the same CSS code for various HTML elements.

They help us in styling specific segments and portions of our webpage. They provide us with the option of uniformly styling similar elements in our web page.

CSS selectors are thus, an important part of the learning curve of CSS.

The advantage to this method is that the <input type = "submit" /> element is unaffected, and the color applied only to the desired text fields.

There are following rules applied to attribute selector.
p[lang] − Selects all paragraph elements with a *lang* attribute.
p[lang="fr"] − Selects all paragraph elements whose *lang* attribute has a value of exactly "fr".
p[lang~="fr"] − Selects all paragraph elements whose *lang* attribute contains the word "fr".
p[lang|="en"] − Selects all paragraph elements whose *lang* attribute contains values that are exactly "en", or begin with "en-".

Multiple Style Rules

- You may need to define multiple style rules for a single element. You can define these rules to combine multiple properties and corresponding values into a single block as defined in the following example –

```
H1 {
color: #36C;
font-weight: normal; letter-spacing:  .4em;
margin-bottom: 1em;
text-transform: lowercase;
}
```

- Here all the property and value pairs are separated by a **semicolon (;)**. You can keep them in a single line or multiple lines. For better readability, we keep them in separate lines.

Pseudo-class

- Using Pseudo classes we can give special effects on the selectors. Most commonly used pseudo-classes are
  - Focus->select links that are the current focus of the keyboard
  - Hover ->When the mouse cursor rolls over a link, that link is in it's hover state and this will select it.
  - hyperlink
  - Active->Selects the link while it is being activated
  - Visited->Selects links that have already been visited by the current browser.

```
a:hover { color: red; }
```

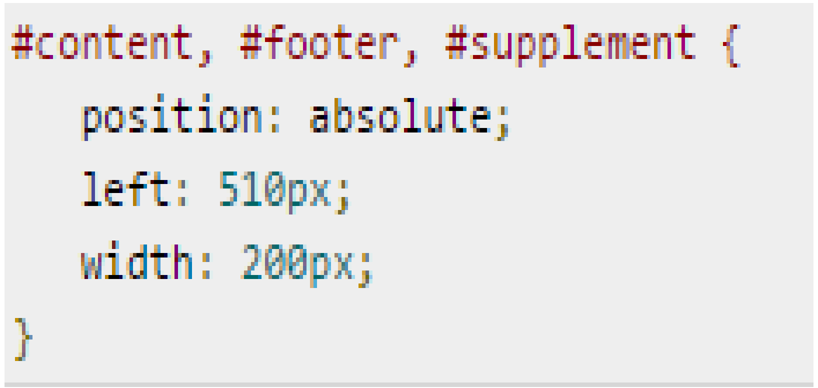
Grouping Selectors

- You can apply a style to many selectors if you like. Just separate the selectors with a comma, as given in the following example –

```
h1, h2, h3 {
color: #36C;
font-weight: normal;
letter-spacing: .4em;
margin-bottom: 1em;
text-transform: lowercase;
}
```

- This define style rule will be applicable to h1, h2 and h3 element as well. The order of the list is irrelevant. All the elements in the selector will have the corresponding declarations applied to them.

You can combine the various *id* selectors together as shown below –



```
#content, #footer, #supplement {
   position: absolute;
   left: 510px;
   width: 200px;
}
```

Different types of Stylesheets

The information contents of any web document can be represented with the help of cascading style sheets. We can define the stylesheets in 3 ways:

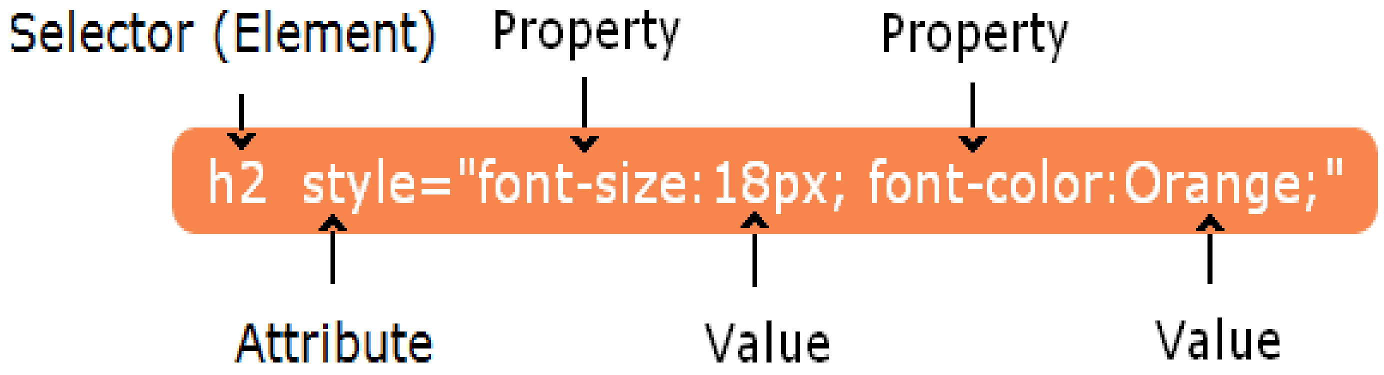
- **Internal** - Placed right on the page whose interface it will affect.
- **External** - Placed in a separate file.
- **Inline** - Placed inside a tag it will affect.

Inline Stylesheet

- Use inline stylesheets when you want to apply a style to a single occurence of an element.
- Inline stylesheets are declared within individual tags and affect those tags only (defined inside HTML element).
- An inline stylesheet has the highest priority.
- Inline stylesheets are declared with the ***style*** attribute in the corresponding tag.
- In-line CSS style consists set of rules in 4 part:
  1. Selector (Element)
  2. Style (Attribute)
  3. Property and
  4. Value

How to write In-line CSS Style

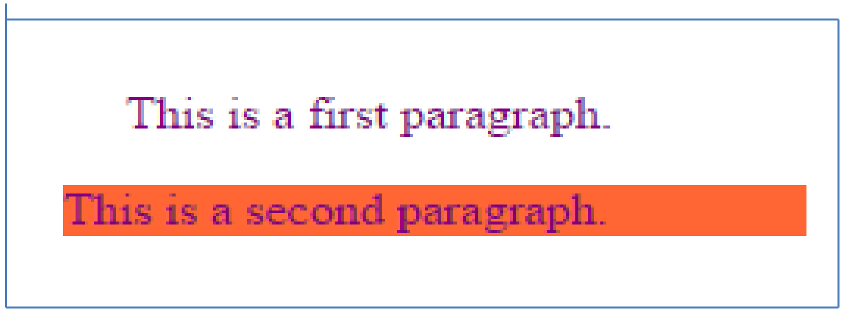
- Selector is normally HTML element that element you want to assign CSS style. And style is attribute to assign CSS property and set suitable value.



Example

```
<p style="color:purple; margin-left:20px">This is a first paragraph.</p>
<div style="color:purple; font-size:16px; background-color:#FF6633;">This is a second paragraph.</div>
```

Output:



This is a first paragraph.

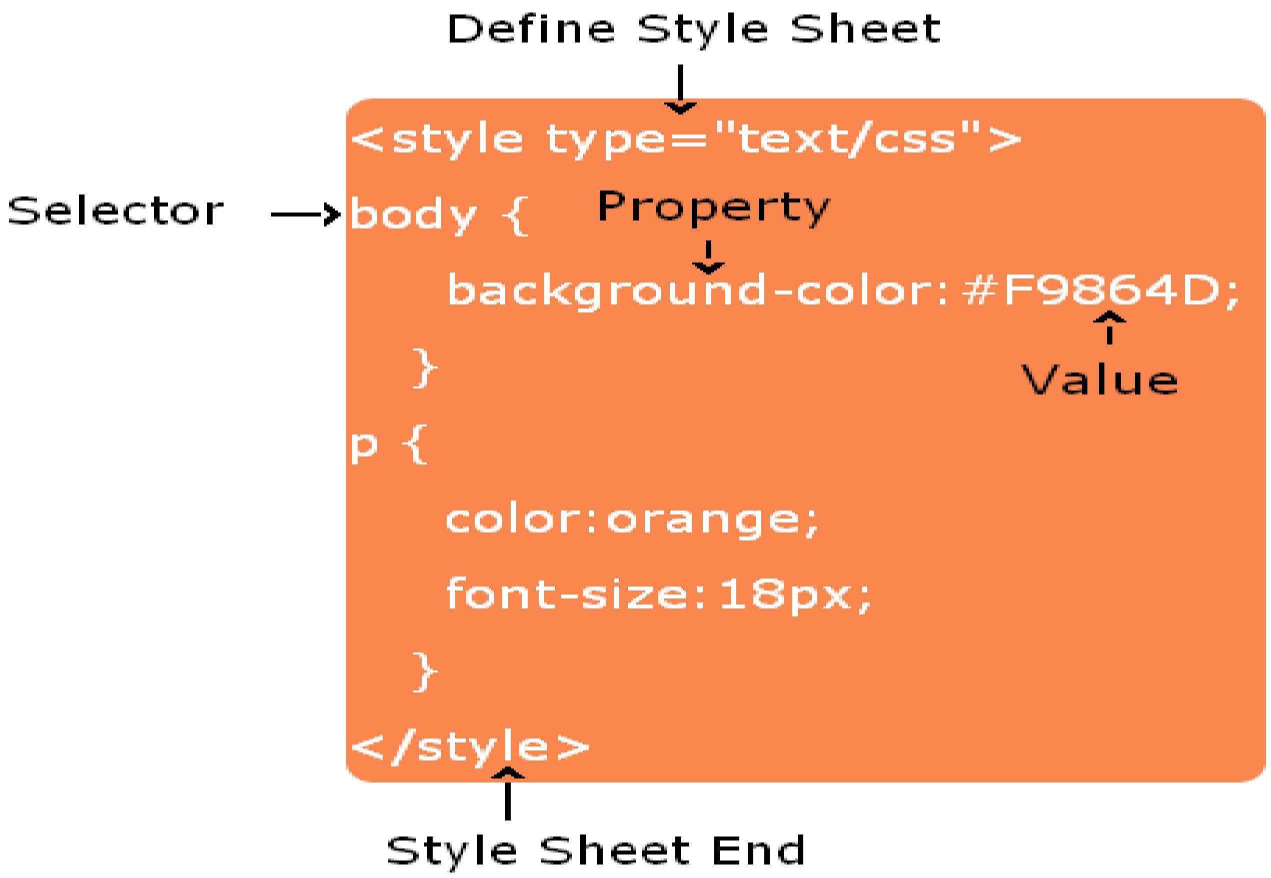
This is a second paragraph.

Internal Stylesheet

- Internal CSS Style includes within web page using *<style type="text/css">.....</style>* element and between this element CSS style properties are listed. Internal CSS style normally written within *<head>.....</head>* element of the web page.
- An internal stylesheet has the second highest priority.
- Internal CSS style consists set of rules in 3 part:
  1. Selector (element, class, id)
  2. Property and
  3. Value

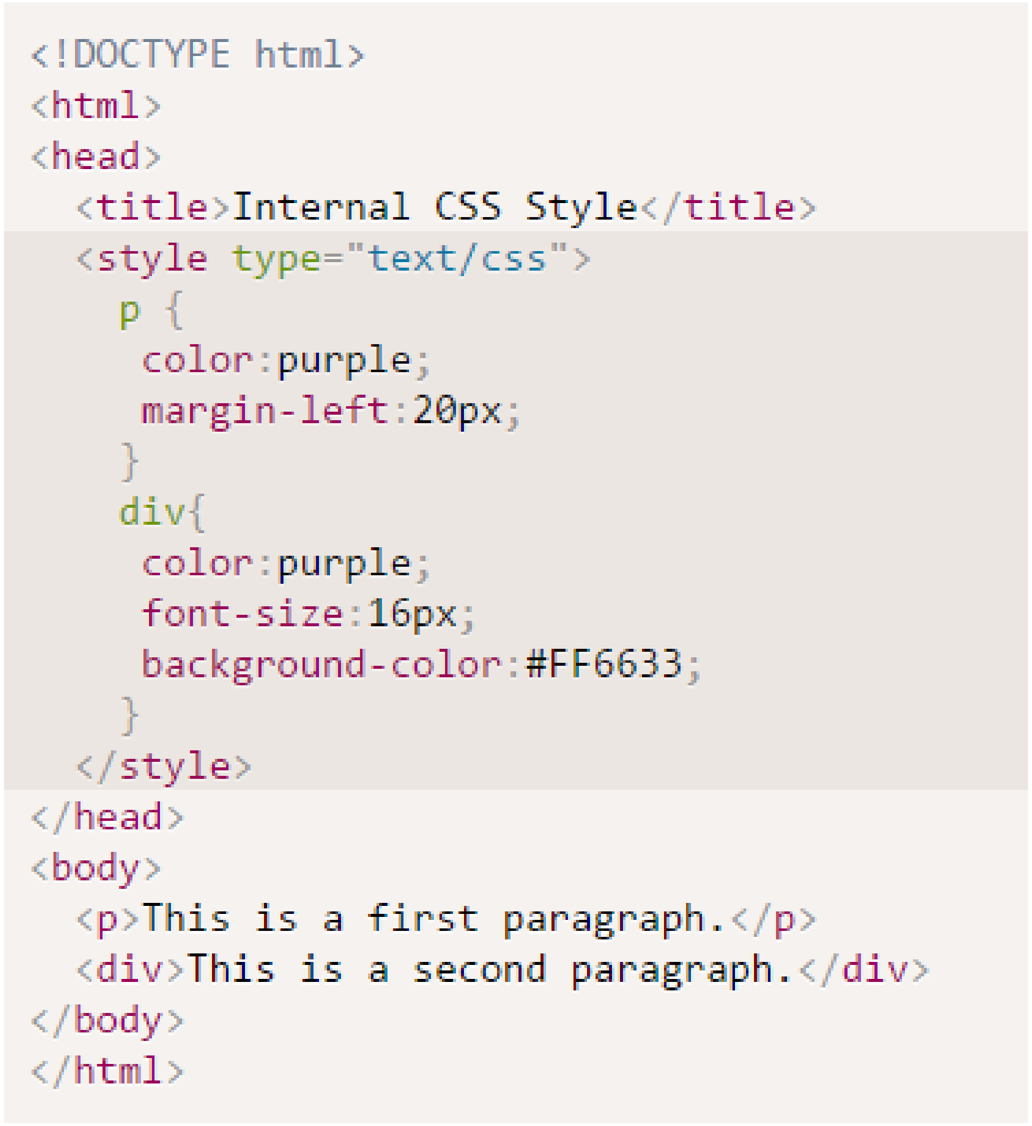
How to write Internal CSS Style

- Selector is normally HTML element that element you want to assign CSS style. All elements within web page that elements assign CSS properties and set suitable values.



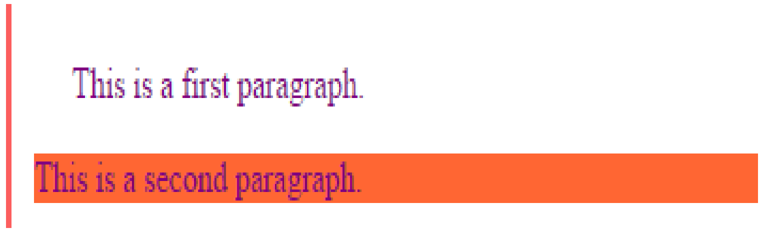
```
<style type="text/css">
body {
    background-color:#F9864D;
  }
p {
    color:orange;
    font-size:18px;
  }
</style>
```

Example



```
<!DOCTYPE html>
<html>
<head>
  <title>Internal CSS Style</title>
  <style type="text/css">
    p {
     color:purple;
     margin-left:20px;
    }
    div{
     color:purple;
     font-size:16px;
     background-color:#FF6633;
    }
  </style>
</head>
<body>
  <p>This is a first paragraph.</p>
  <div>This is a second paragraph.</div>
</body>
</html>
```

Output:



This is a first paragraph.

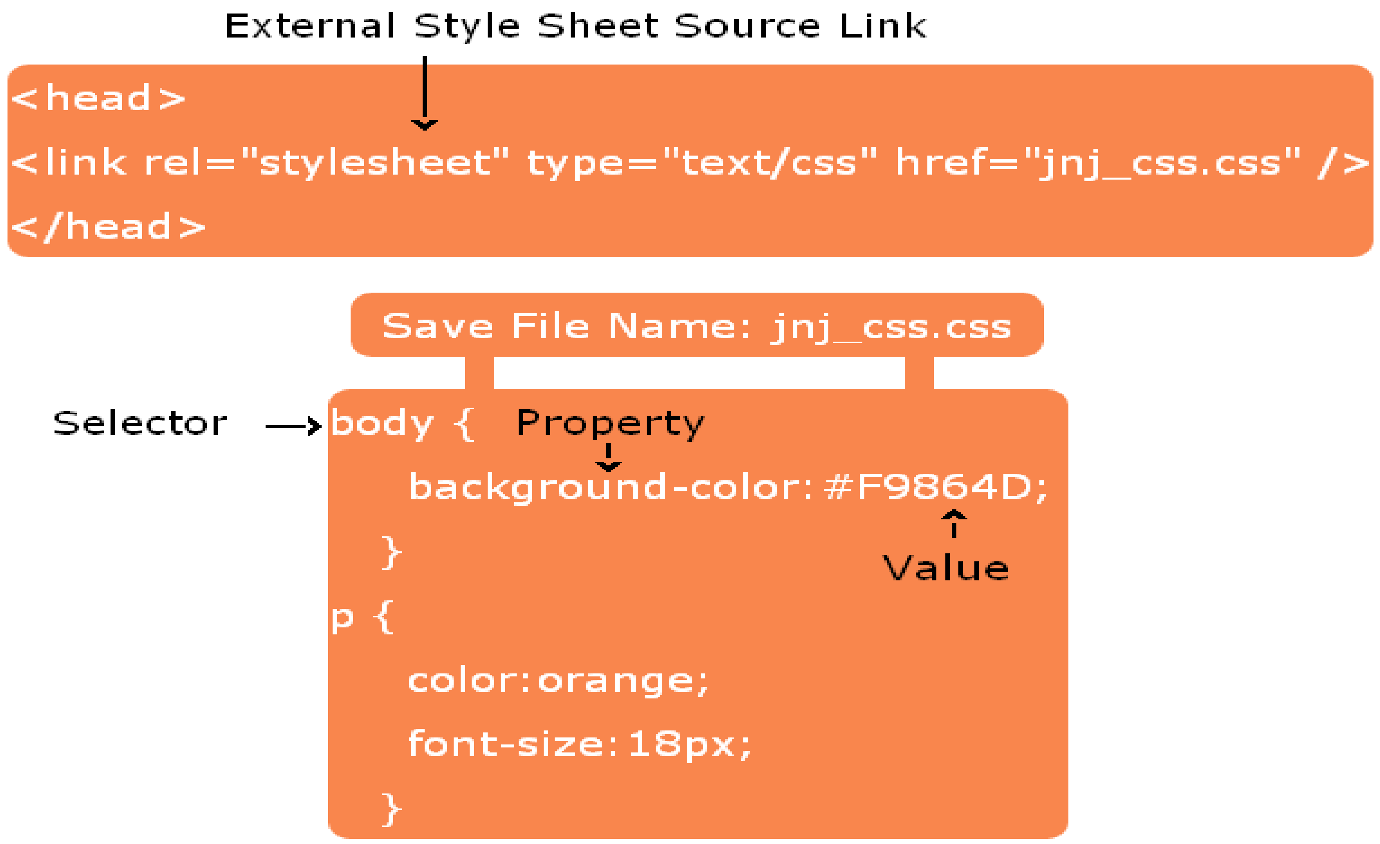
This is a second paragraph.

External Stylesheet

- Use an external stylesheet when you want to apply one style to many pages. If you make one change in an external stylesheet, the change is universal on all the pages where the stylesheet is used.
- An external stylesheet is declared in an external file with a .css extension. It is called by pages whose interface it will affect. External stylesheets are called using the <link> tag which should be placed in the head section of an HTML document. This tag takes three attributes.
- An external stylesheet has the third highest priority.
- External style sheet consists set of rules in 4 part:
    1. External Source link
    2. Selector (element, class, id)
    3. Property and
    4. Value

How to write External Stylesheet

- External stylesheet linked to a web page. Selector is normally HTML element (or class, id) to assign CSS properties and set suitable values.



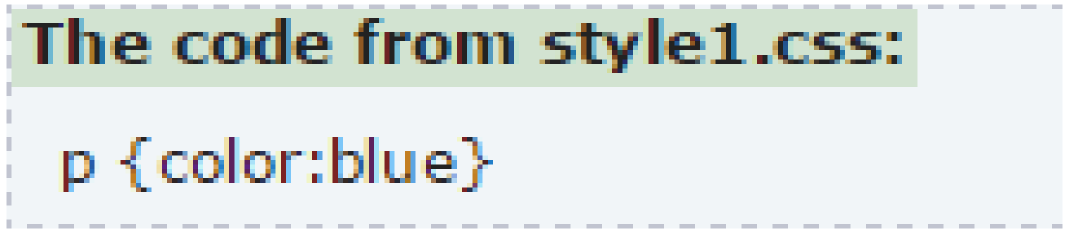
```
<head>
<link rel="stylesheet" type="text/css" href="jnj_css.css" />
</head>
```

```
body {
    background-color: #F9864D;
  }
p {
    color:orange;
    font-size: 18px;
  }
```

<link> Tag

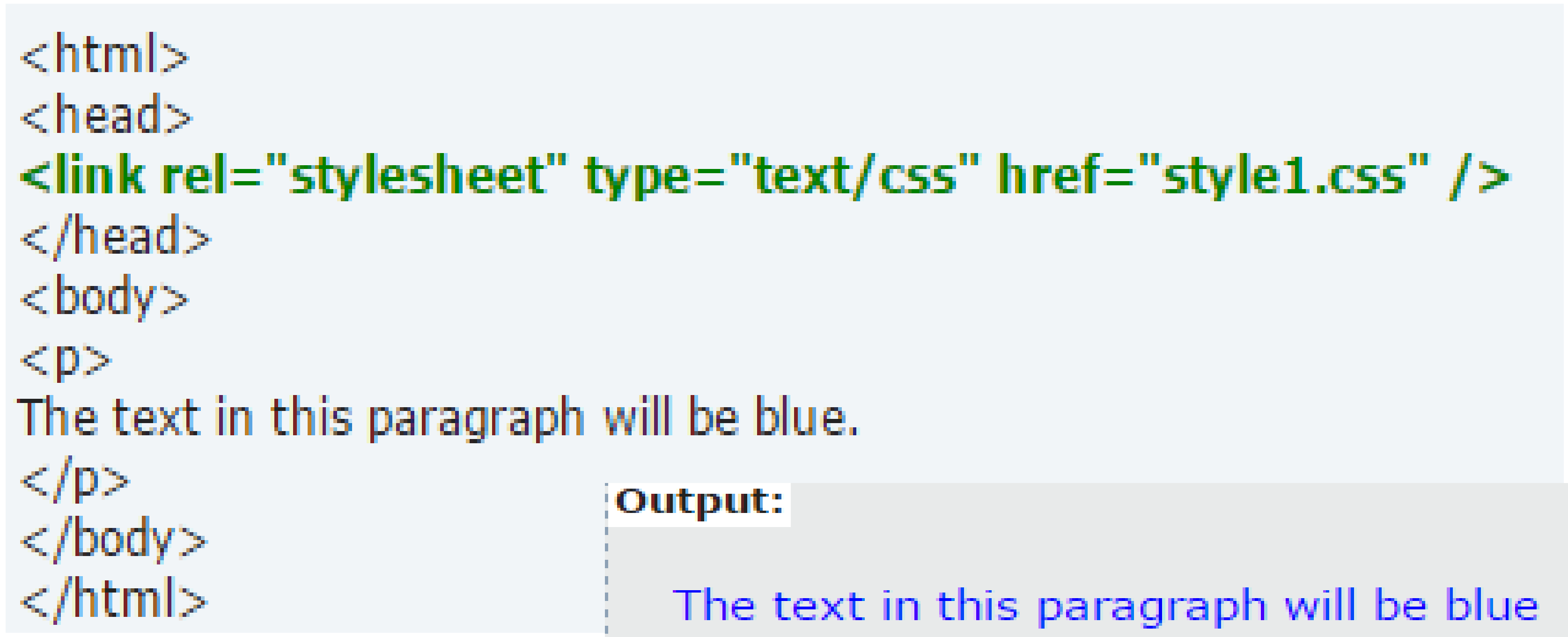
- **<link>** tells browser some file must be linked to the page.
- When we want to link the external style sheet then we have to use <link> tag which is to be written in the head section.
- **Attributes of the <link> tag:**
  - **rel = stylesheet** tells the browser that this linked thing is a stylesheet.
  - **type** = "text/css“ tells browser that what it is reading is text which is affected by the CSS.
  - **href** = “ ” denotes the name and location (pathname) of the external stylesheet to be used.

Example

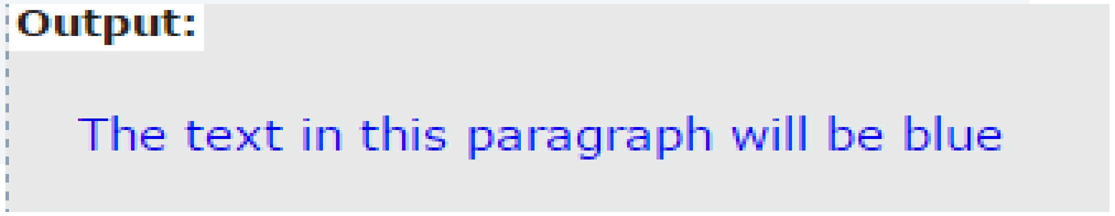


The code from style1.css:

```
p {color:blue}
```



```
<html>
<head>
<link rel="stylesheet" type="text/css" href="style1.css" />
</head>
<body>
<p>
The text in this paragraph will be blue.
</p>
</body>
</html>
```



Output:

```
The text in this paragraph will be blue
```

NOTE: The <style> tag is NOT used in an external stylesheet, and neither are HTML comments.

@import Style Sheet

- @import CSS Style is another way to loading a CSS file.
- @import CSS Style define within

  <style type="text/css">.....</style>

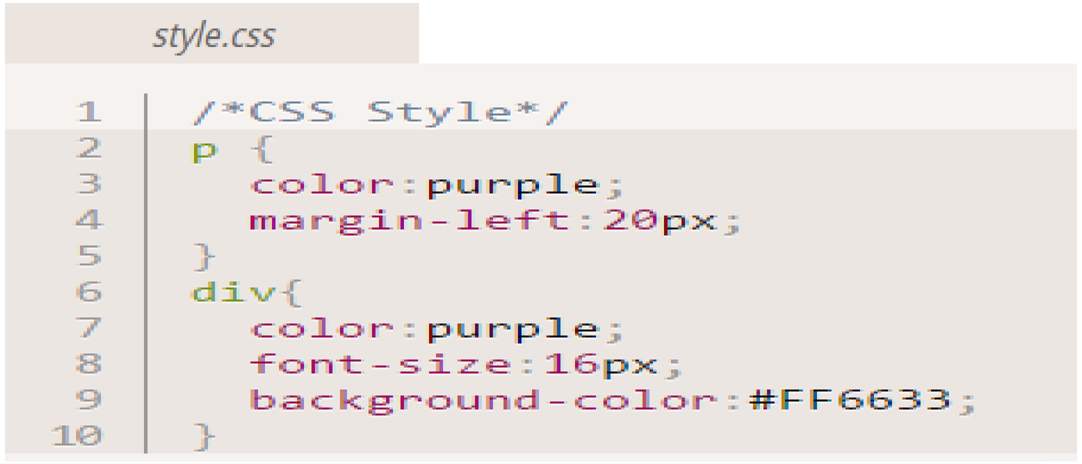
  element in your

  <head>.....</head> of your web page.
- @import CSS style consists set of rules in 3 part:
  1. @import (keyword)
  2. url()
  3. CSS file path

How to write @import CSS Style

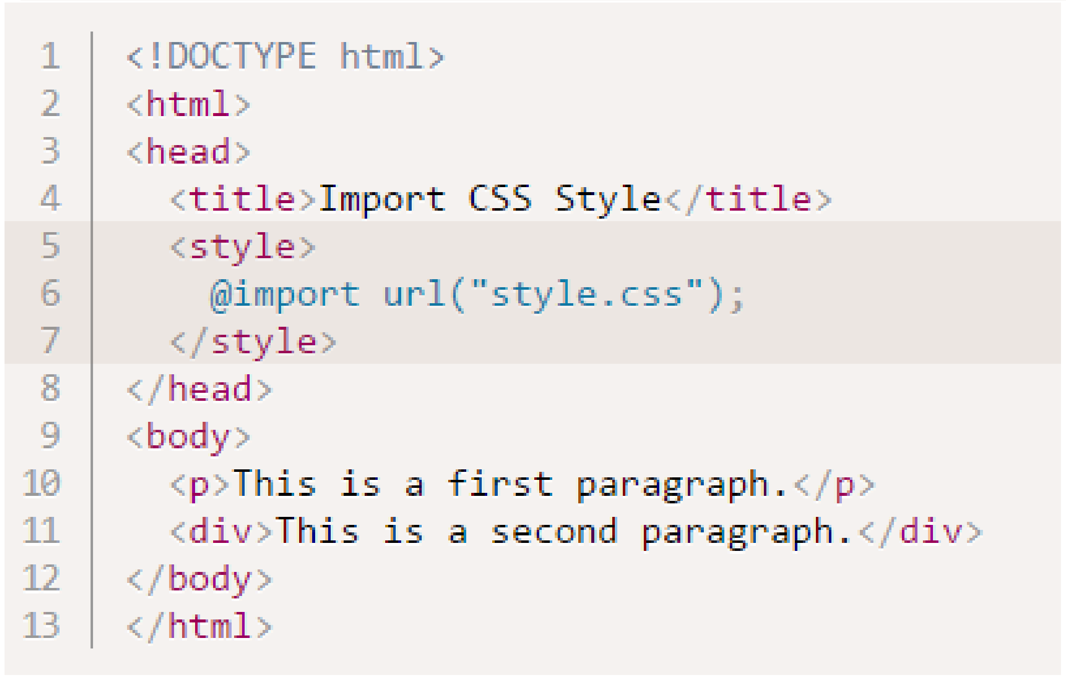
@import url('style.css');

Example



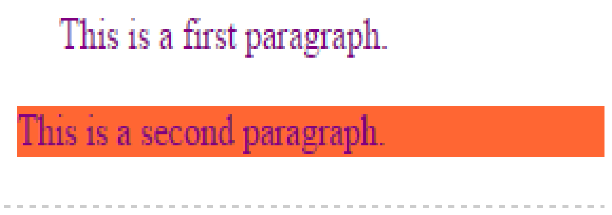
style.css

```css
/*CSS Style*/
p {
   color:purple;
   margin-left:20px;
}
div{
   color:purple;
   font-size:16px;
   background-color:#FF6633;
}
```



```html
<!DOCTYPE html>
<html>
<head>
  <title>Import CSS Style</title>
  <style>
    @import url("style.css");
  </style>
</head>
<body>
  <p>This is a first paragraph.</p>
  <div>This is a second paragraph.</div>
</body>
</html>
```

Output:



```
   This is a first paragraph.
This is a second paragraph.
```

CSS Text Properties

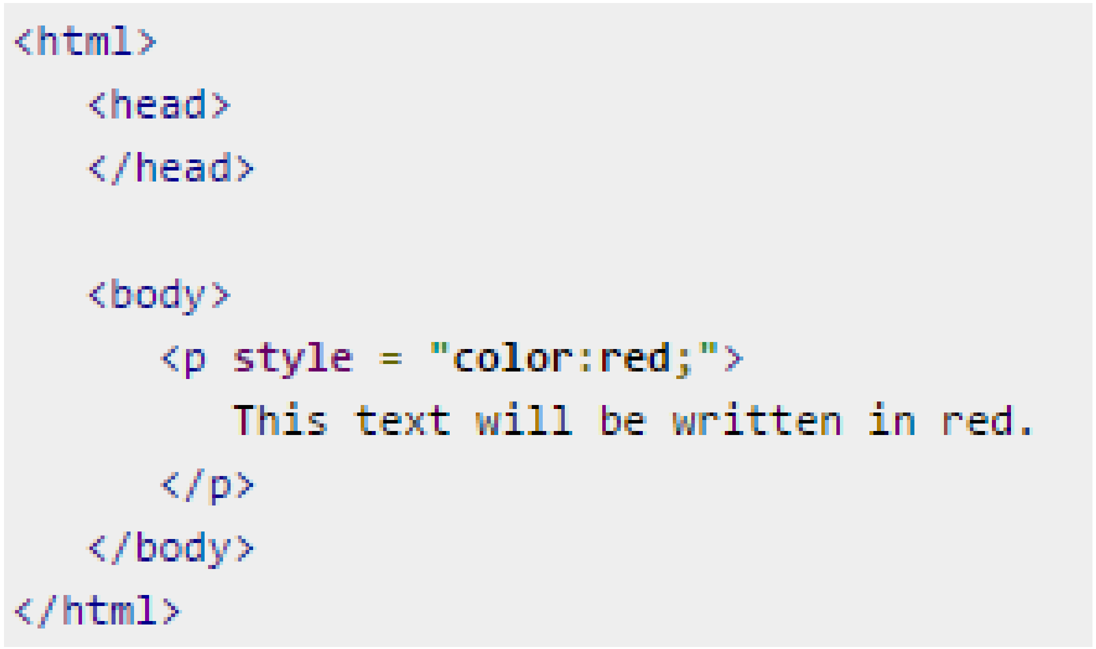
Using this properties you can change the text formatting style. Following are some CSS text properties listed:

- CSS Color
- CSS text-direction
- CSS text-align
- CSS text-indent
- CSS text-decoration
- CSS text-transformation
- CSS letter-spacing
- CSS word-spacing
- CSS white-space
- CSS text-shadow

CSS Color

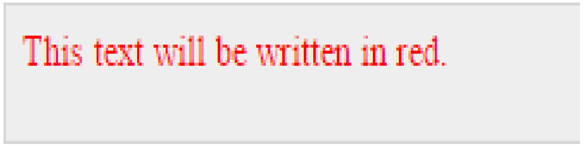
- CSS color property use to set the Text color. The color value can be specified following three types:
  1. Color Name: Orange
  2. Color Hexadecimal Code: #FFA500
  3. Color RGB: rgb(255, 165, 0)

  Example



```
<html>
   <head>
   </head>

   <body>
      <p style = "color:red;">
         This text will be written in red.
      </p>
   </body>
</html>
```

Output:

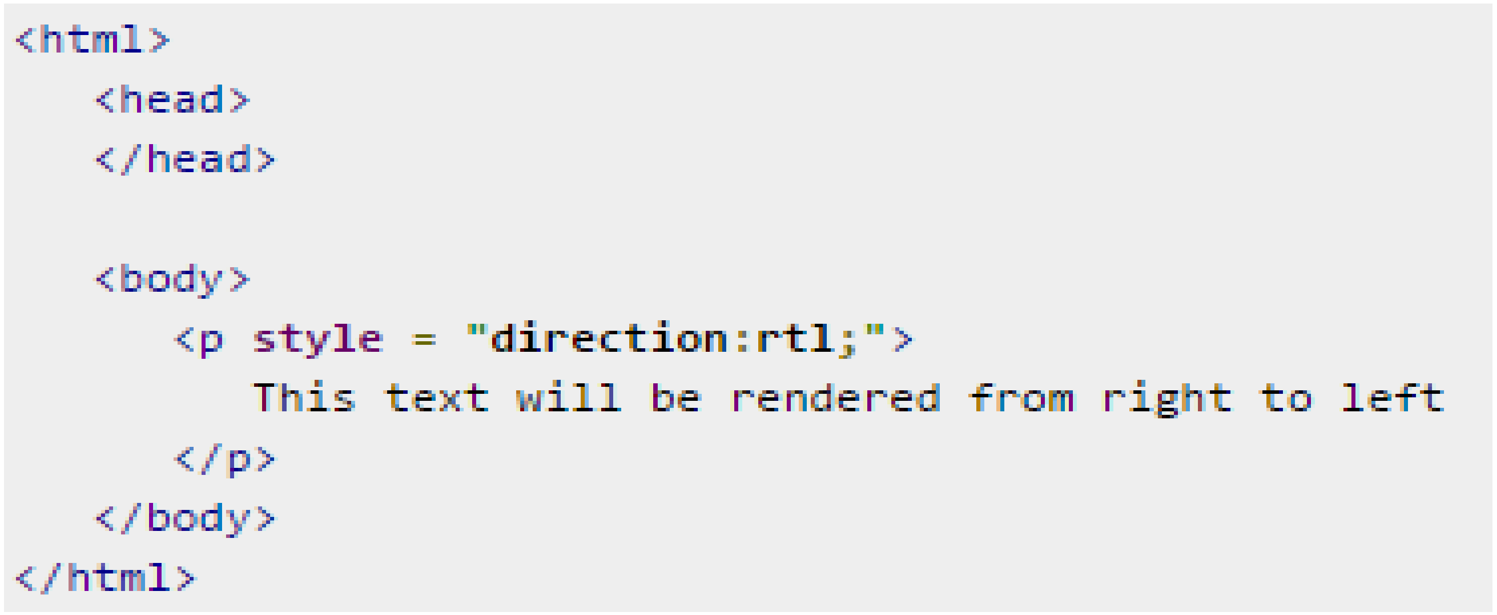


This text will be written in red.

CSS Text-Direction

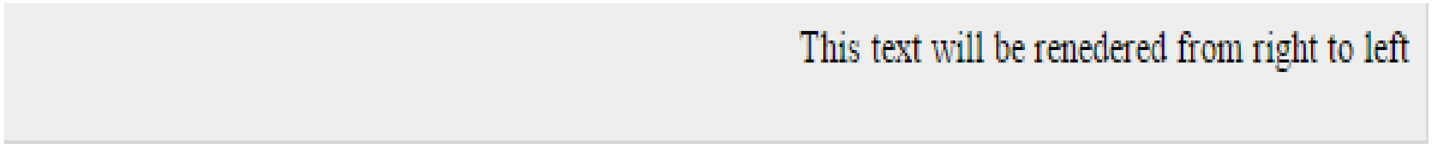
- CSS Text-Direction property is used to set the text direction. Possible values are *ltr or rtl*.

Example:



```
<html>
   <head>
   </head>

   <body>
      <p style = "direction:rtl;">
         This text will be rendered from right to left
      </p>
   </body>
</html>
```

Output:



This text will be renedered from right to left

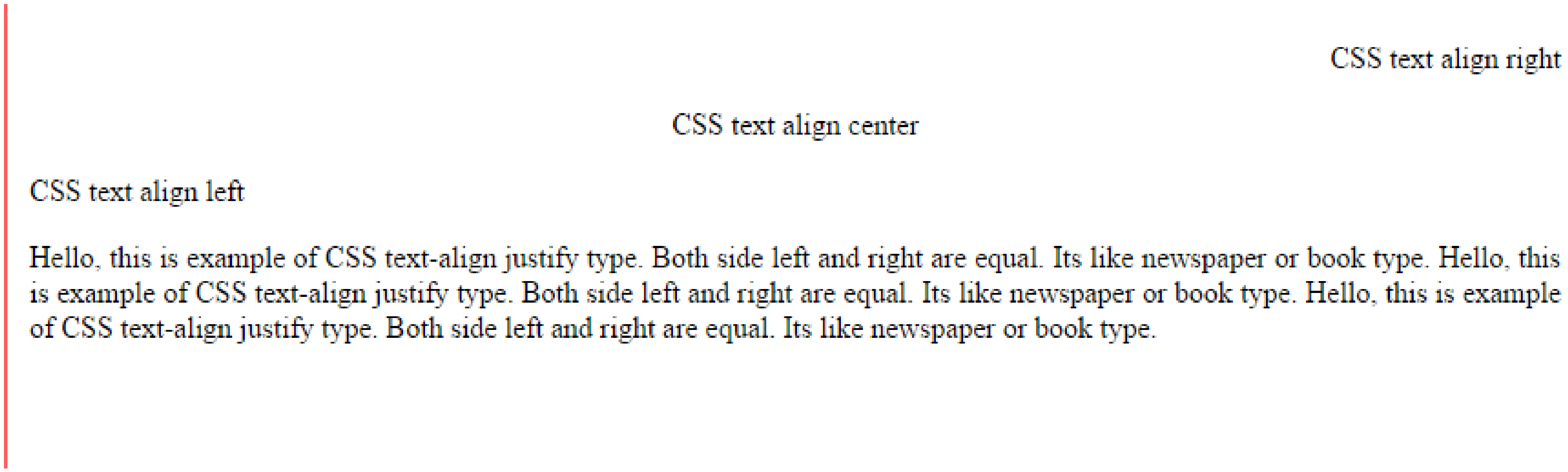
CSS Text-Align

CSS text-align property use to set the horizontal alignment of text. Text-align possible value center, left, right, or justify. When you set text-align property value justify than the effect is both width (left or right) equal like newspaper or books type.

Example:

```
<!DOCTYPE html>
<html>
   <head>
      <title>CSS Text Align</title>
    </head>
    <body>
      <p style="text-align: right;">CSS text align right</p>
      <p style="text-align: center;">CSS text align center</p>
      <p style="text-align: left;">CSS text align left</p>
      <p  style="text-align:  justify;">Hello,  this  is  example  of   CSS  text-align  justify  type.  Both
              side left  and right  are equal. Its like newspaper or book type. Hello,  this is example of
              CSS text-align justify type. Both   side left and right are equal.   Its   like   newspaper   or
              book type. Hello, this is example of CSS text-align justify type. Both side left and right are
              equal. Its   like newspaper or book type.</p>
      </body>
 </html>
```

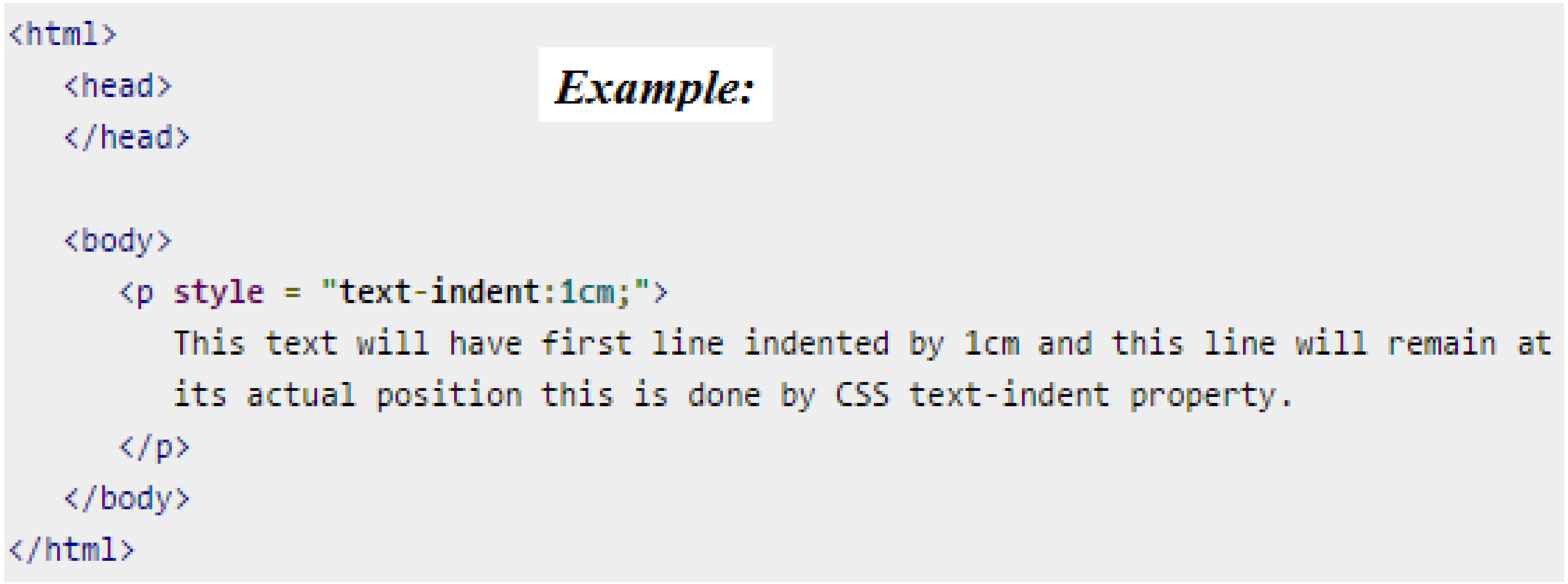
Output:



```
                                                              CSS text align right

                              CSS text align center

CSS text align left

Hello, this is example of CSS text-align justify type. Both side left and right are equal. Its like newspaper or book type. Hello, this
is example of CSS text-align justify type. Both side left and right are equal. Its like newspaper or book type. Hello, this is example
of CSS text-align justify type. Both side left and right are equal. Its like newspaper or book type.
```

CSS Text-Indent

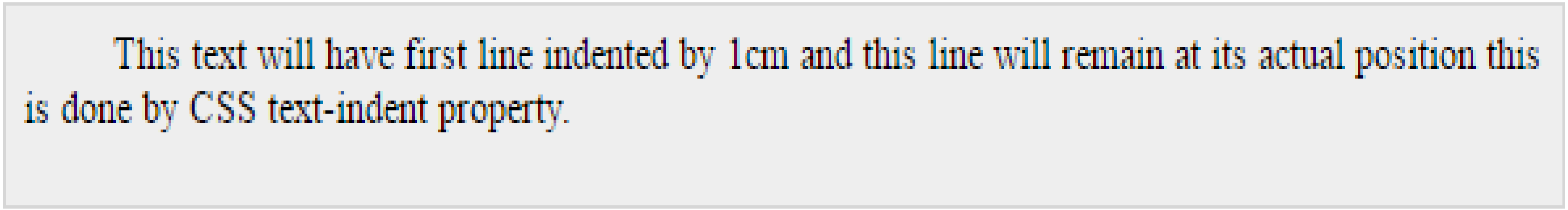
CSS text-indent property is used to set the paragraph first line left side leave the blank space. Possible values are *% or a number specifying indent space*.



Example:

```
<html>
   <head>
   </head>

   <body>
      <p style = "text-indent:1cm;">
         This text will have first line indented by 1cm and this line will remain at
         its actual position this is done by CSS text-indent property.
      </p>
   </body>
</html>
```

Output:



```
      This text will have first line indented by 1cm and this line will remain at its actual position this
is done by CSS text-indent property.
```

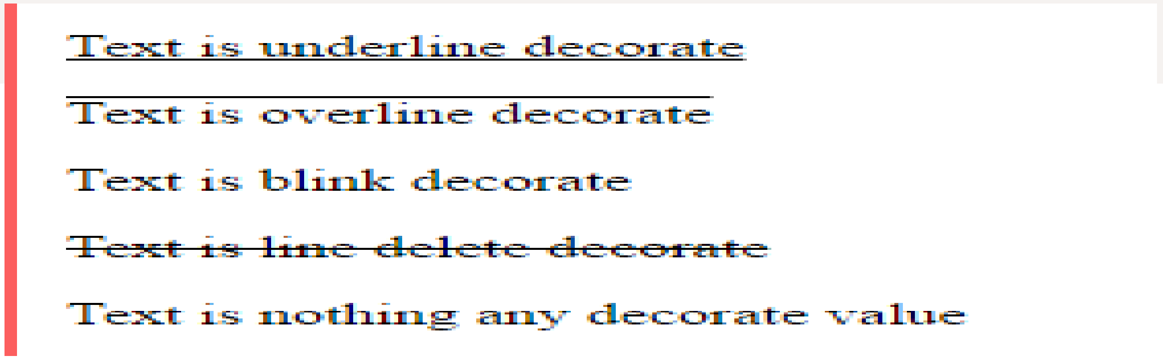
CSS Text-Decoration

CSS text-decoration property use to decorate the text. Possible values are none, underline, overline, blink, line-through etc.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <title>CSS text-decoration</title>
</head>
<body>
  <p style="text-decoration: underline;">Text is underline decorate</p>
  <p style="text-decoration: overline;">Text is overline decorate</p>
  <p style="text-decoration: blink;">Text is blink decorate</p>
  <p style="text-decoration: line-through">Text is line delete decorate</p>
  <p style="text-decoration: none;">Text is nothing any decorate value</p>
</body>
</html>
```

Output:



Text is underline decorate

Text is overline decorate

Text is blink decorate

~~Text is line delete decorate~~

Text is nothing any decorate value

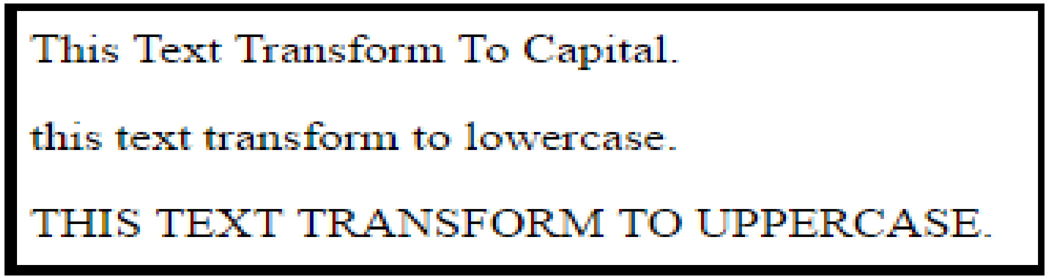
CSS Text-Transformation

CSS text-transformation property use to text transform. CSS text-transformation possible value none, capitalize, lowercase and uppercase in a text. CSS text-transformation property value capitalize, it means first letter capital for all word.

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>CSS text-transform</title>
</head>
<body>
<p style="text-transform: capitalize">This text transform to capital.</p>
<p style="text-transform: lowercase">This text transform to lowercase.</p>
<p style="text-transform: uppercase">This text transform to uppercase.</p>
</body>
</html>
```

Output:



```
This Text Transform To Capital.

this text transform to lowercase.

THIS TEXT TRANSFORM TO UPPERCASE.
```

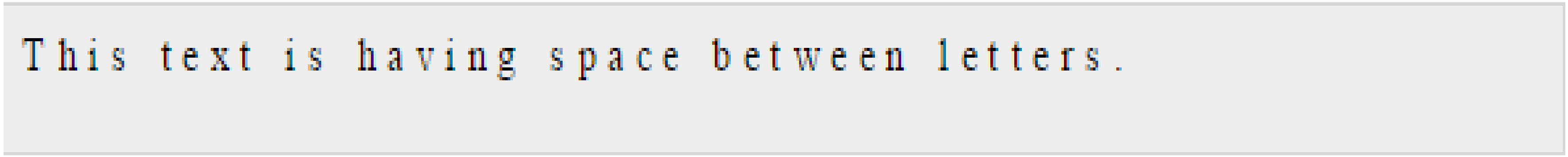
CSS Letter-Spacing

CSS letter-spacing property set blank space between each letters. Possible values are *normal or a number specifying space*..

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>CSS letter-spacing</title>
</head>
<body>
<p style="letter-spacing: 5px;">This text is having space between letters.</p>
</body>
</html>
```

Output:



```
T h i s   t e x t   i s   h a v i n g   s p a c e   b e t w e e n   l e t t e r s .
```

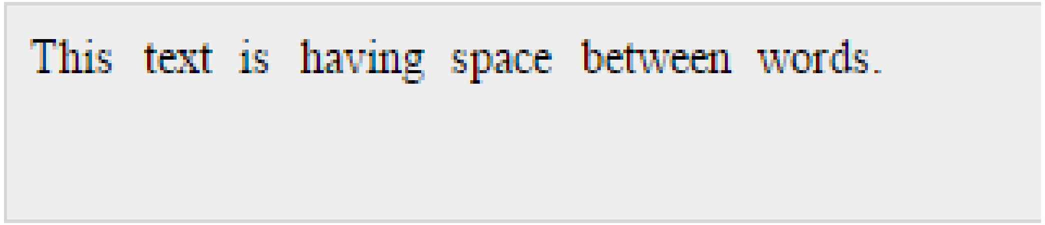
CSS Word-Spacing

CSS word-spacing property set blank space between each words. Possible values are *normal or a number specifying space*.

Example:

```
<!DOCTYPE html> <html>
<head>
<title>CSS word-spacing</title>
</head>
<body>
<p style="word-spacing: 25px;">This text is having space between words.</p>
</body>
</html>
```

Output:



This   text   is   having   space   between   words.

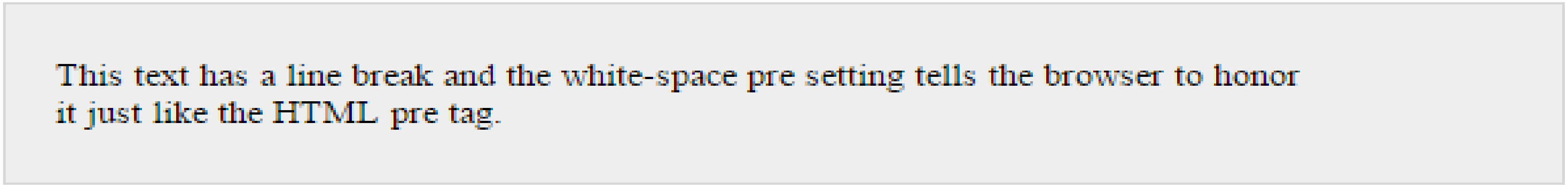
CSS White-Space

CSS white-space property use to set a predefine task. CSS white-space possible value is 'normal', 'pre‘, *‘nowrap’*.

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>CSS white-space</title>
</head>
<body>
<p style="white-space: pre;">
        This text has a line break and the white-space pre setting
        tells the browser to honor it just like the HTML pre tag.
</p>
</body>
</html>
```

Output:



```
This text has a line break and the white-space pre setting tells the browser to honor
it just like the HTML pre tag.
```

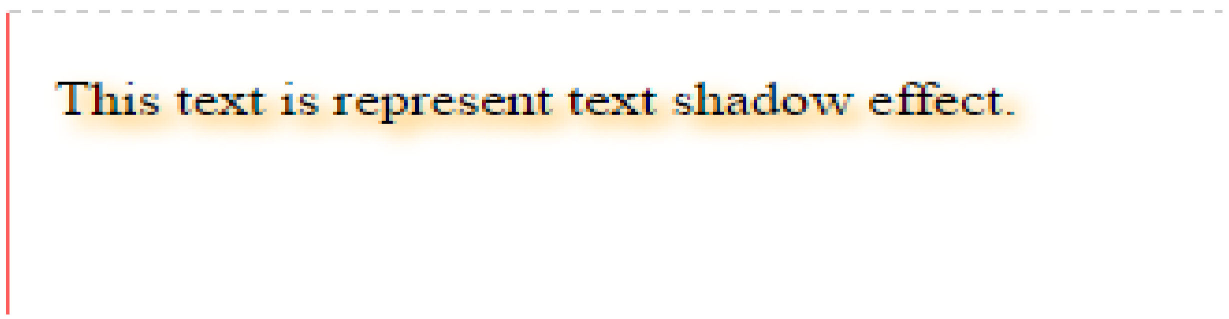
CSS Text-Shadow

CSS text-shadow property is use to decorate text and apply shadow effect style. This may not be supported by all the browsers.

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>CSS text-shadow</title>
</head>
<body>
<p style="text-shadow: 4px 4px 8px orange;">
        This text is represent text shadow effect.</p>
</body>
</html>
```

Output:



This text is represent text shadow effect.

CSS Font Properties

Using CSS font properties you can change the font formatting style. Here are some CSS font properties listed.

- –Font-Families
- –Font-Style
- –Font-Variant
- –Font-Weight
- –Font-Size
- –Font-Shorthands

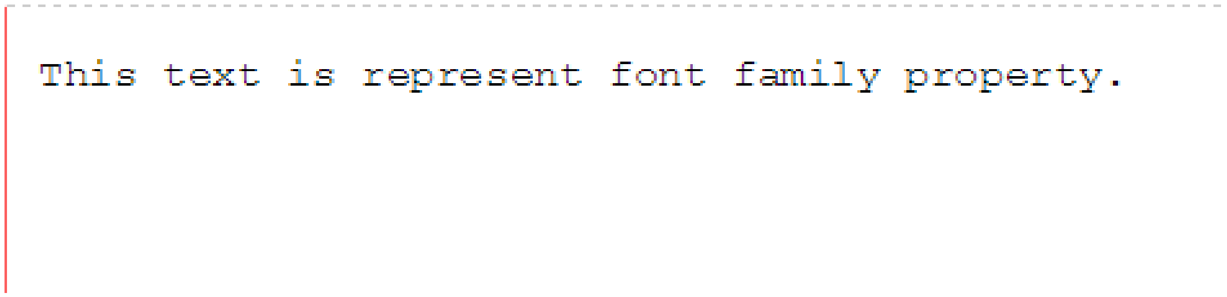
CSS Font-Family

CSS font-family property to define font face name with one or more font. Single Quote (' ') is first priority to apply first. If that font not available then left to right optional apply (eg. Courier).

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>CSS Font Family</title>
</head>
<body>
<p style="font-family: 'Courier New', Courier, monospace;">
        This text is represent font family property.</p>
</body>
</html>
```

Output:



```
This text is represent font family property.
```

Text Property-Font-Families

- The font-family property specifies the font for an element.
- There are two types of font family names:
  1. **family-name** - The name of a font-family, like "times", "courier", "arial", etc.
  2. **generic-family** - The name of a generic-family, like "serif", "sans-serif", "cursive", "fantasy", "monospace".
- **Syntax**

  font-family: *family-name*|*generic-family*|initial|inherit;

Value	**Description**
family-name ***generic-family***	**A prioritized list of font family names and/or generic family names**
initial	**Sets this property to its default value.**
inherit	**Inherits this property from its parent element.**

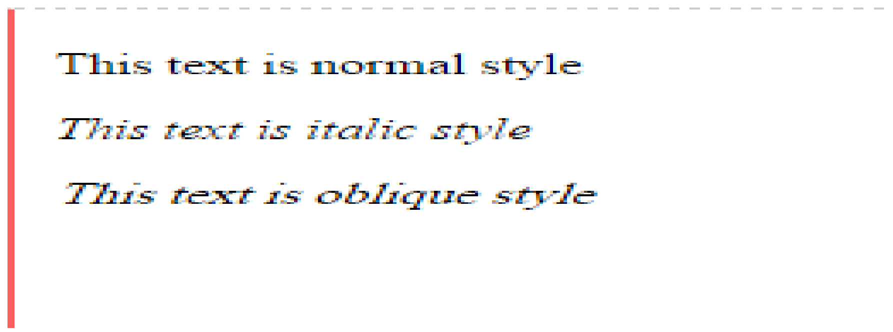
CSS Font-Style

CSS font-style property use to set font style like normal, italic, oblique.

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>CSS Font Style</title>
</head>
<body>
<p style="font-style: normal;">This text is normal style</p>
<p style="font-style: italic;">This text is italic style</p>
<p style="font-style: oblique;">This text is oblique style</p>
</body>
</html>
```

Output:



```
This text is normal style
This text is italic style
This text is oblique style
```

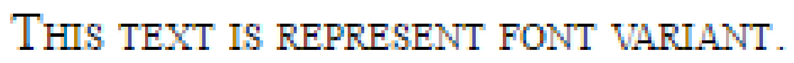
CSS Font-Variant

CSS font-variant property set word first capital letter can be display in variant style. Possible values are *normal and small-caps*.

Example:

```html
<!DOCTYPE html>
<html>
<head>
<title>CSS Font Variant</title>
</head>
<body>
<p style="font-variant: small-caps">This text is represent font variant.</p>
</body>
</html>
```

Output:

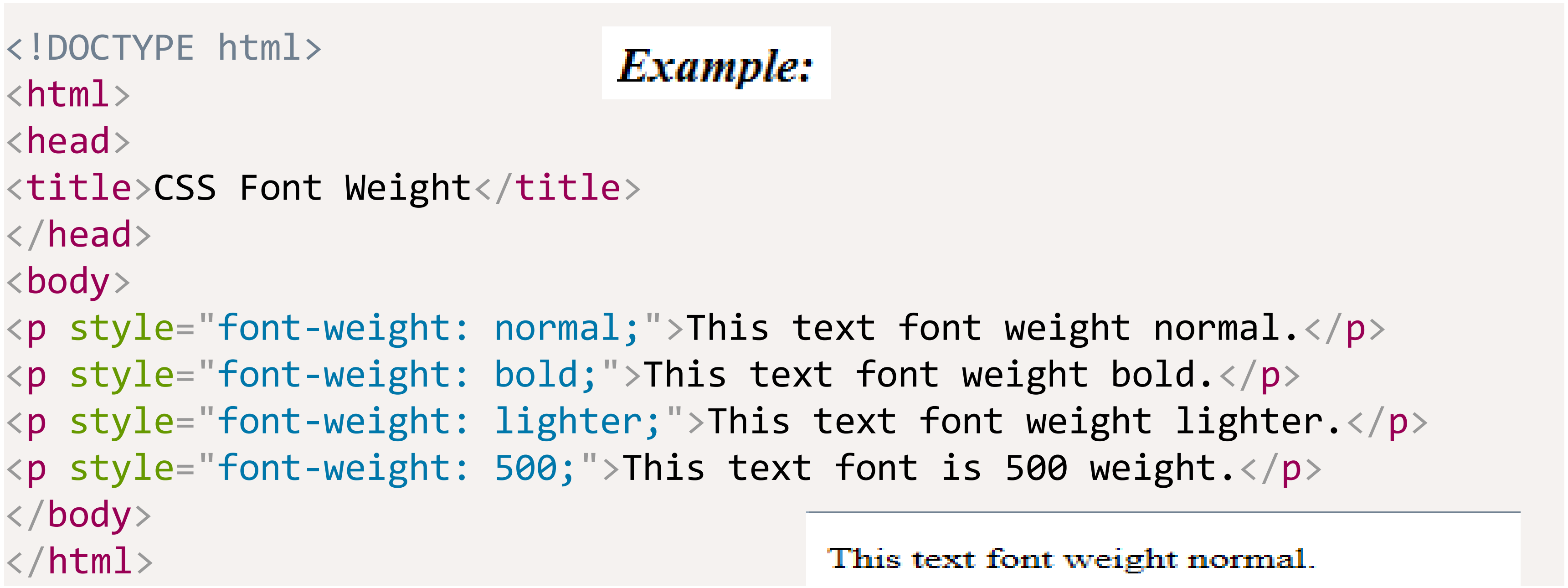


THIS TEXT IS REPRESENT FONT VARIANT.

CSS Font-Weight

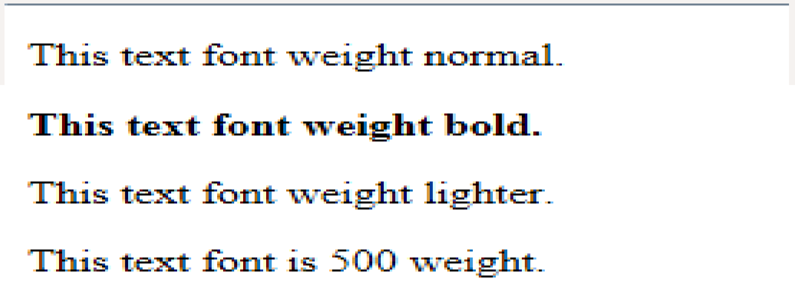
CSS font-weight property use to set font weight like bold, bolder, lighter. The font-weight property provides the functionality to specify how bold a font is. Possible values could be *normal, bold, bolder, lighter, 100, 200, 300, 400, 500, 600, 700, 800, 900*.

Example:



```
<!DOCTYPE html>
<html>
<head>
<title>CSS Font Weight</title>
</head>
<body>
<p style="font-weight: normal;">This text font weight normal.</p>
<p style="font-weight: bold;">This text font weight bold.</p>
<p style="font-weight: lighter;">This text font weight lighter.</p>
<p style="font-weight: 500;">This text font is 500 weight.</p>
</body>
</html>
```

Output:



This text font weight normal.

This text font weight bold.

This text font weight lighter.

This text font is 500 weight.

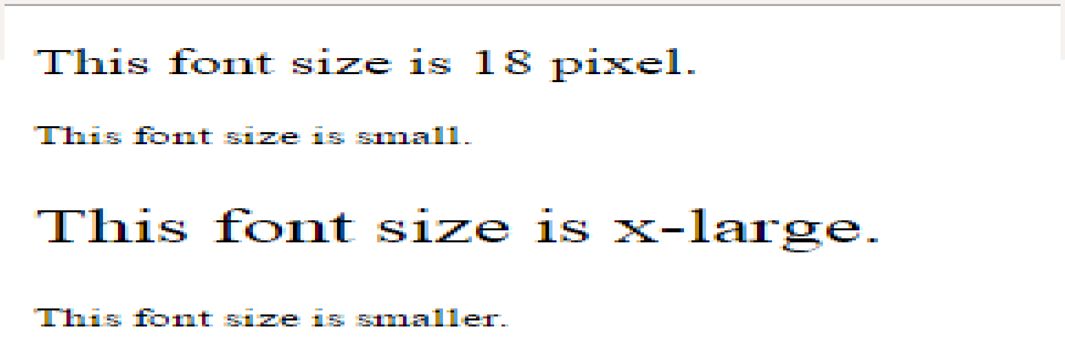
CSS Font-Size

CSS font-size property set the font size. Possible values could be *xx-small, x-small, small, medium, large, x-large, xx-large, smaller, larger, size in pixels or in %*.

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>CSS Font Size</title>
</head>
<body>
<p style="font-size: 18px;">This font size is 18 pixel.</p>
<p style="font-size: small;">This font size is small.</p>
<p style="font-size: x-large;">This font size is x-large.</p>
<p style="font-size: smaller;">This font size is smaller.</p>
</body>
</html>
```

Output:



```
This font size is 18 pixel.

This font size is small.

This font size is x-large.

This font size is smaller.
```

CSS Font-Size Adjust

CSS Font-Size Adjust property set the font size adjust of an element. This property enables you to adjust the x-height to make fonts more legible. Possible value could be any number.

```
<p style="font-size-adjust:0.61;">This text is using font style</p>
```

CSS Font-Stretch

CSS Font-Stretch property set the font stretch of an element. This property relies on the user's computer to have an expanded or condensed version of the font being used. Possible values could be *normal, wider, narrower, ultra-condensed, extra-condensed, condensed, semi-condensed, semi-expanded, expanded, extra-expanded, ultra-expanded*.

```
<p style="font-stretch:ultra-expanded;">This text is using font style</p>
```

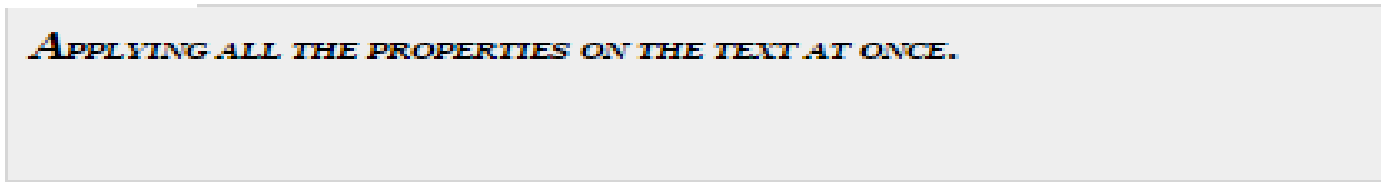
CSS Font-Shorthands

CSS Font-Shorthands property set more than one (or) all the font properties at once.

Example:

```
<html>
<head>
</head>
<body>
 <p style="font : italic small-caps bold 15px georgia;">
        Applying all the properties on the text at once.
 </p>
</body>
</html>
```

Output:



APPLYING ALL THE PROPERTIES ON THE TEXT AT ONCE.

CSS Images

1. **Image Border Property:**

   The *border* property of an image is used to set the width of an image border. This property can have a value in length or in %. A width of zero pixels means no border.

```
<body>
<img style = "border:0px;" src = "/css/images/logo.png" />
<br />
<img style = "border:3px dashed red;" src = "/css/images/logo.png" />
</body>
```

2. **Image Height Property :**

   The *height* property of an image is used to set the height of an image. This property can have a value in length or in %.

```
<img style = "border:1px solid red; height:100px;" src = "/css/images/logo.png" />
```

3. Image Width Property:

The *width* property of an image is used to set the width of an image. This property can have a value in length or in %.

<img style = "border:1px solid red; width:150px;" src = "/css/images/logo.png" />

4. -moz-opacity Property:

The -moz-opacity property of an image is used to set the opacity of an image. This property is used to create a transparent image in Mozilla. IE uses ***filter:alpha(opacity=x)*** *to create transparent images.*

- *In Mozilla (-moz-opacity:x) x can be a value from 0.0 - 1.0. A lower value makes the element more transparent (The same things goes for the CSS3-valid syntax opacity:x).*
- *In IE (filter:alpha(opacity=x)) x can be a value from 0 - 100. A lower value makes the element more transparent.*

<img style = "border:1px solid red; -moz-opacity:0.4; filter:alpha(opacity=40);" src = "/css/images/logo.png" />

CSS Background Properties

- CSS Background properties like background-color, background-image, background-position, background-repeat, background-attachment properties and many more.

CSS background-color=set background color.

```
<p style="background-color: orange;">This text background color orange.</p>
```

CSS background-image

```
<p style="background-image:url(../../images/img_nat.png); color:#FFFFFF;
height:130px; width:200px; font-size:20px;">This text element set background
image.</p>
```

CSS background-repeat

CSS background-repeat property repeat image both side horizontally or vertically. CSS background-repeat property possible value is repeat, no-repeat, repeat-x (vertically repeat), and repeat-y (horizontally repeat).

<p style="background-image:url(../../images/img_nat.png); height:120px; background-repeat: no-repeat;“>This element represent background-repeat property.</p>

CSS background-attachment

<p style="background-image: url(../../images/img_nat.png); height: 130px; width: 200px; overflow: scroll; background-attachment:fixed; color: white;“>This element represent background-attachment property. It means image does not moved only fixed attached a image. background attachment possible value fixed or scroll. you can use this value and display results.This example is Background attachment image is fixed means image does not moved only fixed attached a image. background attachment possible value fixed or scroll. you can use this value and display results.</p>

CSS background-position

CSS background-position property use to set background image in different type position like left, right, center, top, bottom.

<p style="background-image:url(../../images/img_nat.png); height:120px; background-repeat: no-repeat; background-position: 150px;">This element represent background-position property. </p>

CSS - Links

CSS Links set different properties of a hyper link using CSS. You can set following properties of a hyper link −

- The **:link** signifies unvisited hyperlinks.
- The **:visited** signifies visited hyperlinks.
- The **:hover** signifies an element that currently has the user's mouse pointer hovering over it.
- The **:active** signifies an element on which the user is currently clicking.

• Here some rules apply when you set the style for hyperlink.

- a:hover always come after a:link or a:visited
- a:active always come after a:hover

```
<style type = "text/css">
   a:link {color: #000000}
   a:visited {color: #006600}
   a:hover {color: #FFCC00}
   a:active {color: #FF00CC}  </style>
```

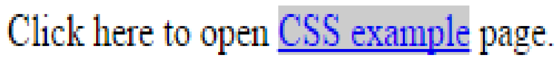
Set the Color of Links

```
<html>   <head>
 <style type = "text/css">
          a:link {color:#000000}
          a:visited {color: #006600} </style>
   </head>
  <body>
  <a href = "">Link</a>
  </body> </html>
```

Link

CSS Links Background Color

```
<head>
<style type="text/css"> a:link {background-color:#CCCCCC;} </style>
</head>
```


Click here to open CSS example page.

CSS Links Text Decoration

```
<head>
<style type="text/css"> a:link { text-decoration: none; }
a:hover { text-decoration: underline; } </style>
</head>
```

CSS Lists

- Lists are very helpful in conveying a set of either numbered or bullet points. We have the following five CSS properties, which can be used to control lists −
  - The **list-style-type** allows you to control the shape or appearance of the marker.
  - The **list-style-position** specifies whether a long point that wraps to a second line should align with the first line or start underneath the start of the marker.
  - The **list-style-image** specifies an image for the marker rather than a bullet point or number.
  - The **list-style** serves as shorthand for the preceding properties.
  - The **marker-offset** specifies the distance between a marker and the text in the list.

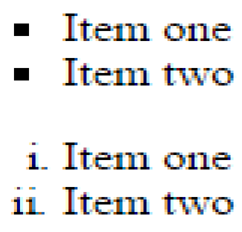
CSS list-style-type

- CSS list-style-type property use for display list item either Ordered or Unordered list.
- Ordered list possible value roman, alpha, numeric and manymore. Unordered list possible value circle, square, disk and none.
- List Style Type possible values see the list-style-type:

Unordered list value

Value	Description
disk	disk type list item display, this is default value
circle	Circle type list item display
square	Square type list item display.
none	Nothing any style apply in list item

```
<body>
<ul style="list-style-type: square;">
         <li>Item one</li>
         <li>Item two</li>
</ul>
<ul style="list-style-type: lower-roman;">
         <li>Item one</li>
         <li>Item two</li>
</ul> </body>
```



- ▪ Item one
- ▪ Item two

i. Item one
ii. Item two

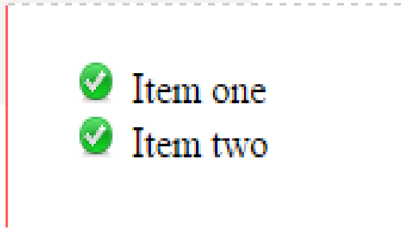
Ordered list value

Value	Description
decimal	decimal type numeric list style (eg. 1, 2, 3 and so on.) this is default value.
upper-alpha	Uppercase alphabetically list style (eg. A, B, C ans so on.)
lower-alpha	Lowercase alphabetically list style (eg. a, b, c and so on.)
upper-roman	Uppercase roman numerals list (eg. I, II, III and so on.)
lower-roman	Lowercase roman numerals list (eg. i, ii, iii and so on.)

CSS list-style-image

CSS list-style-image property set list style URL specified image.

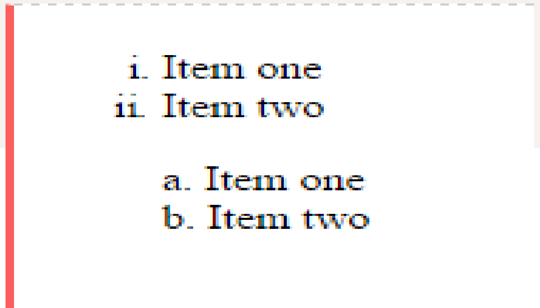
```
<body>
<ul style="list-style-image: url(../../images/new.png);">
<li>Item one</li>
<li>Item two</li>
</ul>
```


✓ Item one
✓ Item two

CSS list-style-position

CSS list-style-position set list style position either inside or outside.

```
<ul style="list-style-position: outside; list-style-type: lower-roman;">
<li>Item one</li>
<li>Item two</li>
</ul>
<ul style="list-style-position: inside; list-style-type: lower-alpha;">
<li>Item one</li>
<li>Item two</li>
</ul>
```

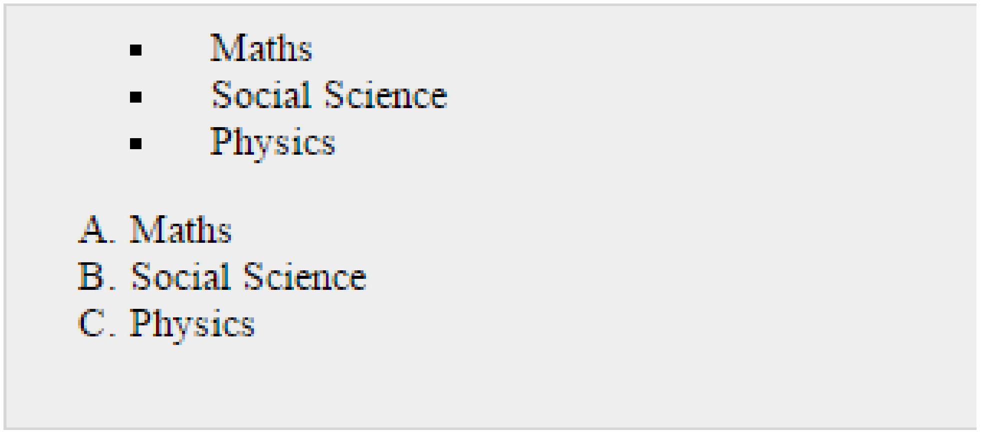

i. Item one
ii. Item two

a. Item one
b. Item two

The marker-offset Property

The *marker-offset* property allows you to specify the distance between the marker and the text relating to that marker. Its value should be a length as shown in the following example −

```
<body>
  <ul style = "list-style: inside square; marker-offset:2em;">
            <li>Maths</li>
             <li>Social Science</li>
            <li>Physics</li>
  </ul>
  <ol style = "list-style: outside upper-alpha; marker-offset:2cm;">
            <li>Maths</li>
            <li>Social Science</li>
            <li>Physics</li>
  </ol>
</body>
```

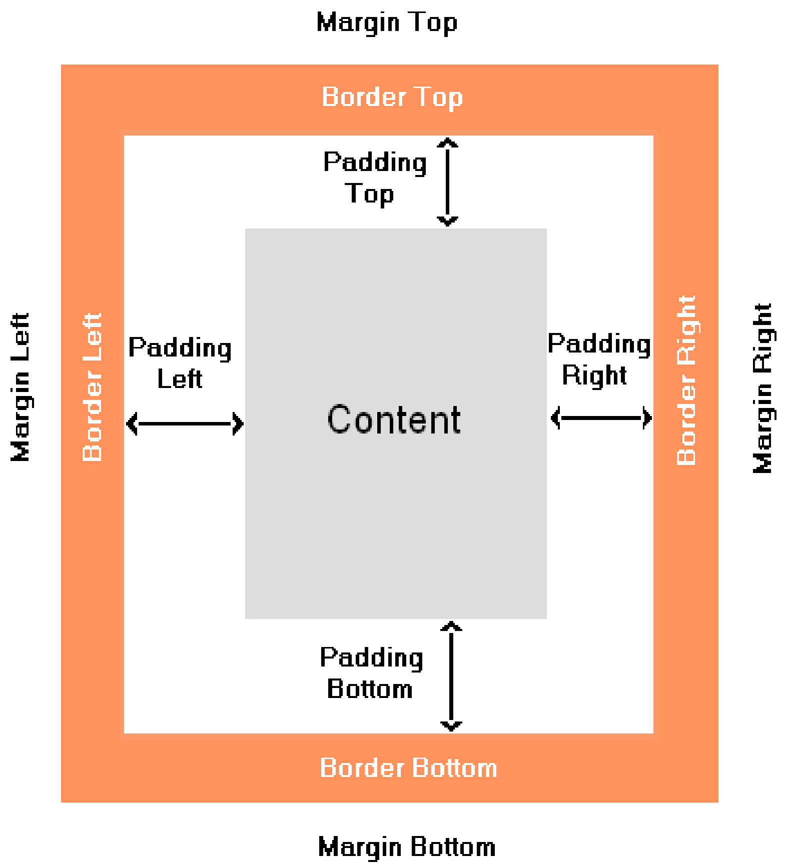


```
    ▪     Maths
    ▪     Social Science
    ▪     Physics

A. Maths
B. Social Science
C. Physics
```

CSS Layout Style

CSS Layout Box Model-CSS layout box model give you layout knowledge of element content and how to set padding, margin or border. CSS layout allow you to set border around padding and set margin around content to better manage.

Margin
Margin are border outside area. So margin is not support any special CSS properties exclude margin group properties It is use to leave a blank space around element content.
Border
Border are define the border around element content. It is use to display border outside content(like text, images and so forth).
Padding
Padding define the content or border in this two part middle part are call padding. It is use to leave blank space between content or border.
Width or Height of Elements
You can specify width or hight either absolute or relative value.
<u>Total width count by this way,</u>
Total width = width + left padding + right padding + left border + right border + left margin + right margin
<u>Total Height count by this way,</u>
Total Height = height + top padding + bottom padding + top border + bottom border + top margin + bottom margin



CSS Border

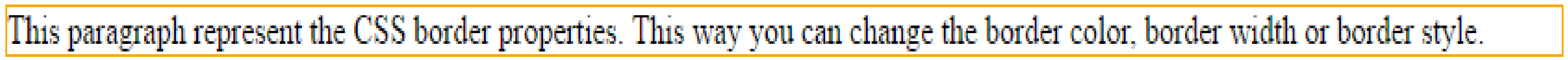
CSS Border properties give you control to set border style.

Border	**Border Properties**
border-top	border-top-color border-top-style border-top-width
border-bottom	border-bottom-color border-bottom-style border-bottom-width
border-left	border-left-color border-left-style border-left-width
border-right	border-right-color border-right-style border-right-width

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>CSS border property</title>
</head>
<body>
<p style="border-style: solid; border-width:1px; border-color: orange;">
        This paragraph represent the CSS border properties. This way you can
        change the border color, border width or border style.</p>
</body>
</html>
```

Output:

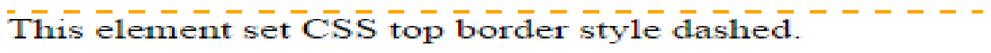


This paragraph represent the CSS border properties. This way you can change the border color, border width or border style.

CSS border shorthand property

```
<body>
<p style="border-top:2px dashed orange;">
         This element set CSS top border style dashed.</p>
</body>
```

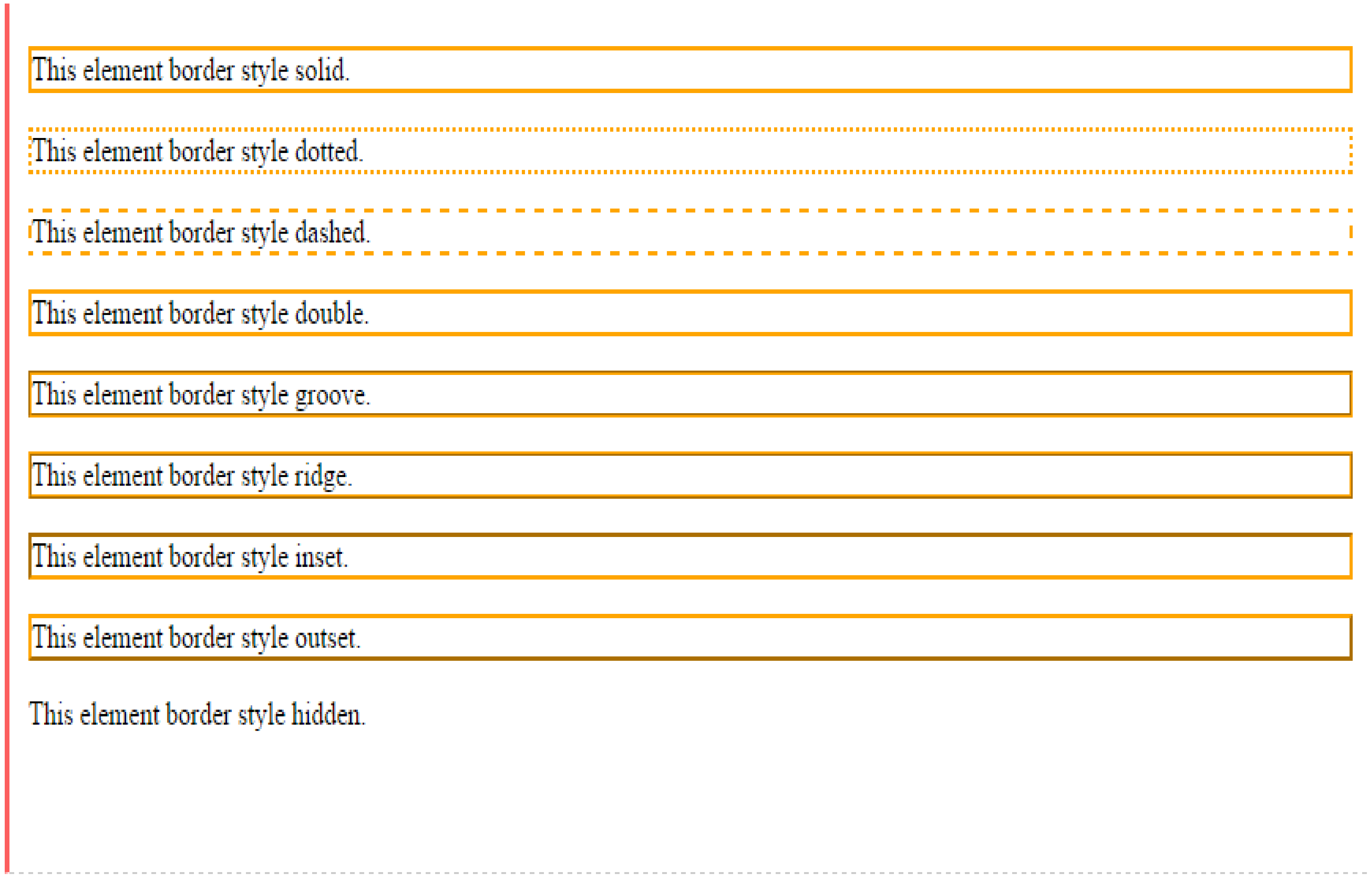
Output:


This element set CSS top border style dashed.

CSS border-style property

```
<!DOCTYPE html> <html>
<head>
<title>CSS border property</title>
</head>
<body>
<p style="border: 2px solid orange;">This element border style solid.</p>
<p style="border: 2px dotted orange;">This element border style dotted.</p>
<p style="border: 2px dashed orange;">This element border style dashed.</p>
<p style="border: 2px double orange;">This element border style double.</p>
<p style="border: 2px groove orange;">This element border style groove.</p>
<p style="border: 2px ridge orange;">This element border style ridge.</p>
<p style="border: 2px inset orange;">This element border style inset.</p>
<p style="border: 2px outset orange;">This element border style outset.</p>
<p style="border: 2px hidden orange;">This element border style hidden.</p>
</body> </html>
```

Output:



This element border style solid.

This element border style dotted.

This element border style dashed.

This element border style double.

This element border style groove.

This element border style ridge.

This element border style inset.

This element border style outset.

This element border style hidden.

CSS border-radius Property

The CSS `border-radius` property defines the radius of an element's corners.

1. Rounded corners for an element with a specified background color:
2. Rounded corners for an element with a border:
3. Rounded corners for an element with a background image:

```
<!DOCTYPE html> <html> <head>
<style>
#rcorners1 {   border-radius: 25px; background: #73AD21; padding: 20px; width: 200px;
height: 150px; }
#rcorners2 {    border-radius: 25px; border: 2px solid #73AD21; padding: 20px; width:
200px;   height: 150px;  }
#rcorners3 {   border-radius: 25px; background: url(paper.gif); background-position: left
top; background-repeat: repeat; padding: 20px; width: 200px; height: 150px;  }
</style> </head>
<body> <h1>The border-radius Property</h1>
<p>Rounded corners for an element with a specified background color:</p>
<p id="rcorners1">Rounded corners!</p>
<p>Rounded corners for an element with a border:</p>
<p id="rcorners2">Rounded corners!</p>
<p>Rounded corners for an element with a background image:</p>
<p id="rcorners3">Rounded corners!</p>
</body> </html>
```

```
<!DOCTYPE html> <html> <head>
<style>
#rcorners1 { border-radius: 15px 50px 30px 5px; background: #73AD21; padding: 20px;
  width: 200px; height: 150px; }
#rcorners2 { border-radius: 15px 50px 30px; background: #73AD21; padding: 20px;
  width: 200px; height: 150px; }
#rcorners3 { border-radius: 15px 50px; background: #73AD21; padding: 20px; width: 200px;
  height: 150px; }
#rcorners4 { border-radius: 15px; background: #73AD21; padding: 20px; width: 200px;
  height: 150px; }
</style> </head>
<body>
<h1>The border-radius Property</h1>
<p>Four values - border-radius: 15px 50px 30px 5px:</p>
<p id="rcorners1"></p>
<p>Three values - border-radius: 15px 50px 30px:</p>
<p id="rcorners2"></p>
<p>Two values - border-radius: 15px 50px:</p>
<p id="rcorners3"></p>
<p>One value - border-radius: 15px:</p>
<p id="rcorners4"></p>
</body>
</html>
```

CSS box-shadow Property

The `box-shadow` property attaches one or more shadows to an element.

CSS Syntax

box-shadow: none|*h-offset v-offset blur spread color* |inset|initial|inherit;

Property Values

Value	Description
none	Default value. No shadow is displayed
h-offset	Required. The horizontal offset of the shadow. A positive value puts the shadow on the right side of the box, a negative value puts the shadow on the left side of the box
v-offset	Required. The vertical offset of the shadow. A positive value puts the shadow below the box, a negative value puts the shadow above the box
blur	Optional. The blur radius. The higher the number, the more blurred the shadow will be
spread	Optional. The spread radius. A positive value increases the size of the shadow, a negative value decreases the size of the shadow
color	Optional. The color of the shadow. The default value is the text color.
inset	Optional. Changes the shadow from an outer shadow (outset) to an inner shadow
initial	Sets this property to its default value.
inherit	Inherits this property from its parent element.

Example-1
Add a blur effect to the shadow:

```
<!DOCTYPE html> <html> <head>
<style>
#example1 {  border: 1px solid;  padding: 10px;  box-shadow: 5px 10px 8px #888888; }
#example2 {  border: 1px solid; padding: 10px;  box-shadow: 5px 10px 18px #888888; }
#example3 {  border: 1px solid; padding: 10px;  box-shadow: 5px 10px 18px red; }
</style> </head>
<body>
<h2>box-shadow: 5px 10px 8px #888888:</h2>
<div id="example1">
  <p>The optional third value adds a blur effect to the shadow.</p>
</div>
<h2>box-shadow: 5px 10px 18px #888888:</h2>
<div id="example2">
  <p>More blurred.</p>
</div>
<h2>box-shadow: 5px 10px 18px red:</h2>
<div id="example3">
  <p>More blurred and red.</p>
</div>
</body> </html>
```

Example-2

Define the spread radius of the shadow:

```
<!DOCTYPE html>
<html>
<head>
<style>
#example1 {
  border: 1px solid;
  padding: 10px;
  box-shadow: 5px 10px 8px 10px #888888;
}
</style>
</head>
<body>

<h2>box-shadow: 5px 10px 8px 10px #888888:</h2>
<div id="example1">
  <p>The optional fourth value defines the spread of the shadow.</p>
</div>

</body>
</html>
```

Example-3
Define multiple shadows:

```
<!DOCTYPE html> <html> <head> <style>
#example1 {
  border: 1px solid;  padding: 10px;
  box-shadow: 5px 5px blue, 10px 10px red, 15px 15px        green; }
#example2 {
  border: 1px solid;   padding: 10px;
  box-shadow: 5px 5px 8px blue, 10px 10px 8px red, 15px 15px 8px green;
}
</style> </head>
<body>
<h2>box-shadow: 5px 5px blue, 10px 10px red, 15px 15px green:</h2>
<div id="example1">
  <p>Define multiple shadows.</p>
</div>
<h2>box-shadow: 5px 5px 8px blue, 10px 10px 8px red, 15px 15px 8px green:</h2>
<div id="example2">
  <p>Define multiple shadows with blur effect.</p>
</div>
</body></html>
```

Example-4
Add the inset keyword:

```
<!DOCTYPE html>
<html>
<head>
<style>
#example1 {  border: 1px solid;  padding: 10px;  box-shadow: 5px 10px inset; }
#example2 {  border: 1px solid;  padding: 10px;  box-shadow: 5px 10px 20px red inset;}
</style>
</head>
<body>
<h2>box-shadow: 5px 10px inset:</h2>
<div id="example1">
  <p>The inset keyword changes the shadow to one inside the frame.</p>
</div>
<h2>box-shadow: 5px 10px 20px red inset:</h2>
<div id="example2">
  <p>Inset, red and blur.</p>
</div>
</body>
</html>
```

Example-4

Images thrown on the table. This example demonstrates how to create "polaroid" pictures and rotate the pictures:

```
<!DOCTYPE html> <html> <head>
<style>
body {   margin: 30px;  background-color: #E9E9E9;}
div.polaroid {   width: 284px; padding: 10px 10px 20px 10px;  border: 1px solid #BFBFBF;
  background-color: white;   box-shadow: 10px 10px 5px #aaaaaa;
}
div.rotate_right {  float: left;
-ms-transform: rotate(7deg); /* IE 9 */
-webkit-transform: rotate(7deg); /* Safari */
transform: rotate(7deg);
}
div.rotate_left {   float: left;
-ms-transform: rotate(-8deg); /* IE 9 */
-webkit-transform: rotate(-8deg); /* Safari */
transform: rotate(-8deg);
}
</style></head>
<body>
<div class="polaroid rotate_right">
  <img src="pulpitrock.jpg" alt="Pulpit rock" width="284" height="213">
  <p class="caption">The pulpit rock in Lysefjorden, Norway.</p>
</div>
<div class="polaroid rotate_left">
  <img src="cinqueterre.jpg" alt="Monterosso al Mare" width="284" height="213">
  <p class="caption">Monterosso al Mare. One of the five villages in Cinque Terre, Italy.</p>
</div>
</body></html>
```

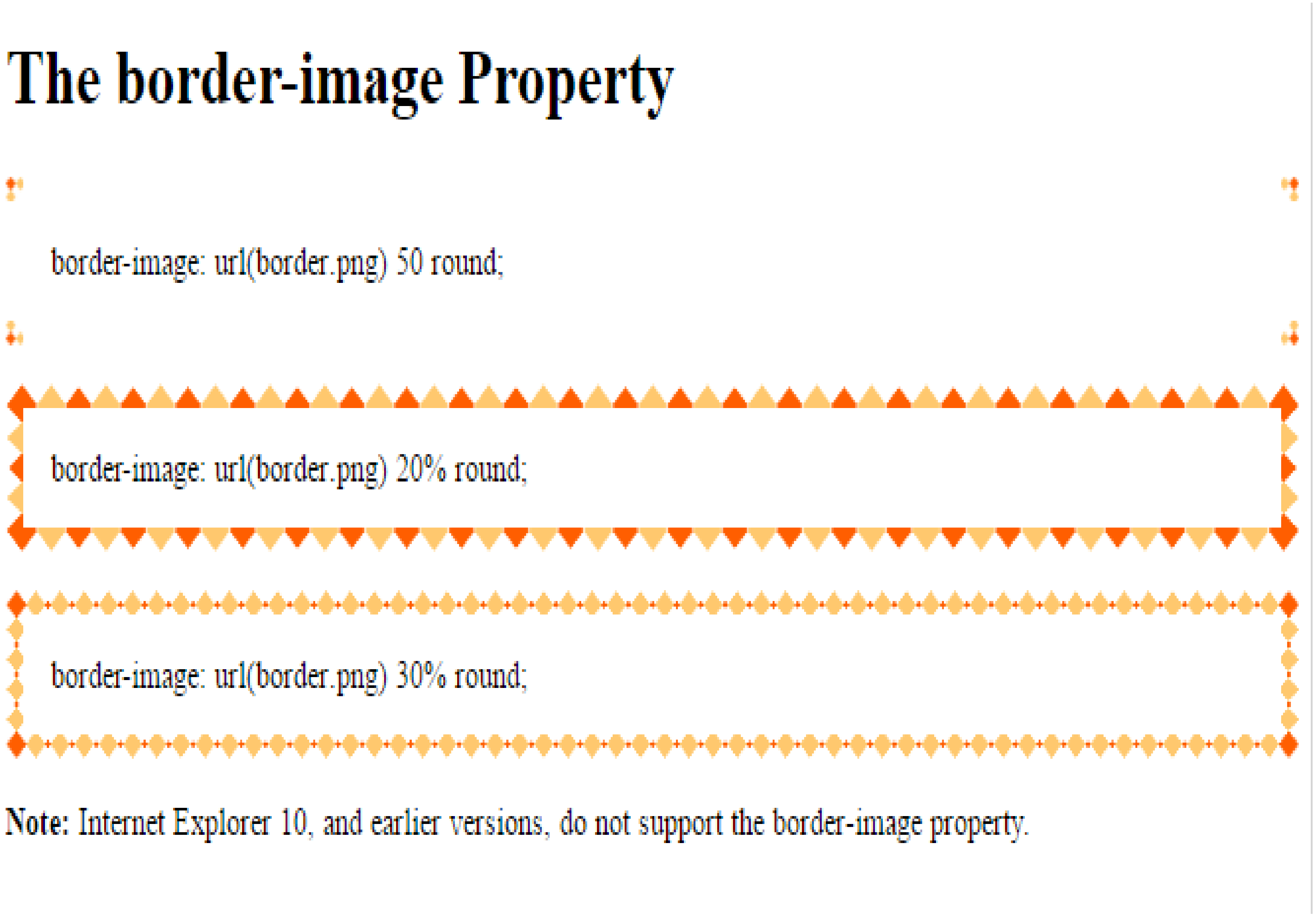
CSS border-image Property

The CSS `border-image` property allows you to specify an image to be used instead of the normal border around an element.

The property has three parts:

1. The image to use as the border
2. Where to slice the image
3. Define whether the middle sections should be repeated or stretched

```
<!DOCTYPE html><html><head>
<style>
#borderimg1 { border: 10px solid transparent; padding: 15px; border-image: url(border.png) 50
round;}
#borderimg2 { border: 10px solid transparent; padding: 15px; border-image: url(border.png) 20%
round;}
#borderimg3 { border: 10px solid transparent; padding: 15px; border-image: url(border.png) 30%
round;}
</style> </head>
 <body>
<h1>The border-image Property</h1>
<p id="borderimg1">border-image: url(border.png) 50 round;</p>
<p id="borderimg2">border-image: url(border.png) 20% round;</p>
<p id="borderimg3">border-image: url(border.png) 30% round;</p>
<p><strong>Note:</strong> Internet Explorer 10, and earlier versions, do not support the border-
image property.</p> </body>
</html>
```



The border-image Property

border-image: url(border.png) 50 round;

border-image: url(border.png) 20% round;

border-image: url(border.png) 30% round;

Note: Internet Explorer 10, and earlier versions, do not support the border-image property.

CSS Margin

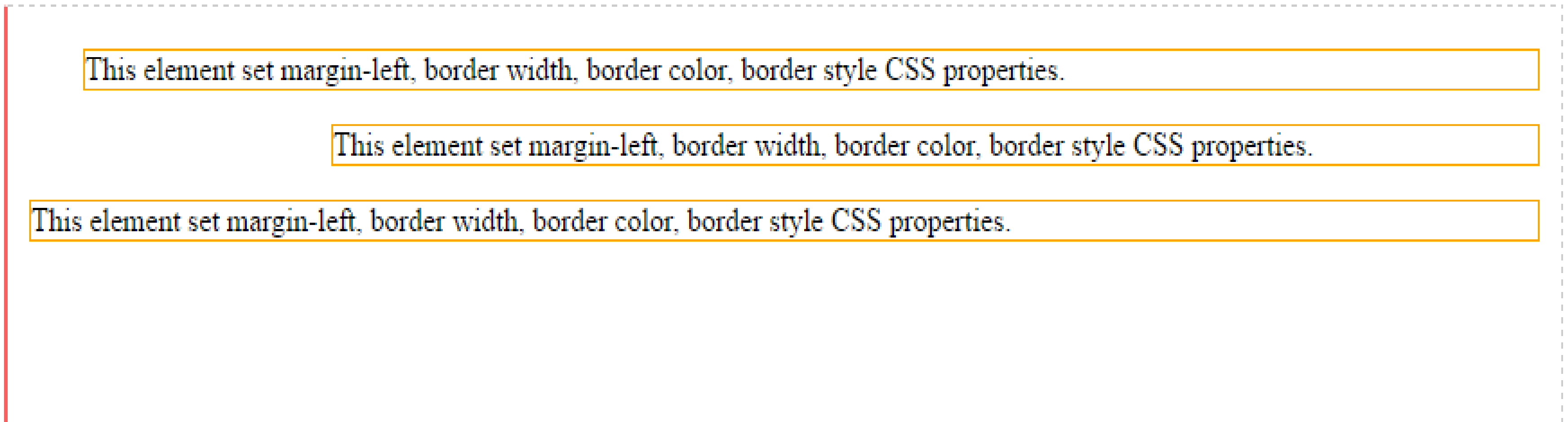
CSS Margin property leave blank space around the content elements (outside of border). CSS Margin property support pixel, percentage or auto measurement value.

Properties	Value	Description
margin	px % auto	User define pixel value. User define percentage value. Set automatic.
margin-left	px % auto	margin left side set define pixel value. margin left side set User define percentage value. margin left side Set automatic.
margin-right	px % auto	margin right side set User define pixel value. margin right side set User define percentage value. margin right side set Set automatic.
margin-top	px % auto	margin top set user define pixel value. margin top set user define percentage value. margin top set automatic.
margin-bottom	px % auto	margin bottom set user define pixel value. margin bottom set user define percentage value. margin bottom set automatic.

Example:

```html
<!DOCTYPE html>
<html>
   <head>
      <title>CSS margin property</title>
      <style type="text/css">
         p.first { border: 1px solid orange; margin-left: 30px; }
         p.second { border: 1px solid orange; margin-left: 20%; }
         p.third { border: 1px solid orange; margin-left: auto; }
      </style>
   </head>
 <body>
     <p class="first">This element set margin-left, border width, border color, border
         style CSS properties.</p>
     <p class="second">This element set margin-left, border width, border color, border
         style CSS properties.</p>
     <p class="third">This element set margin-left, border width, border color, border
         style CSS properties.</p>
 </body>
</html>
```

Output:



```
This element set margin-left, border width, border color, border style CSS properties.

This element set margin-left, border width, border color, border style CSS properties.

This element set margin-left, border width, border color, border style CSS properties.
```

CSS Margin shorthand property

CSS margin property write in shorthand way including following margin properties:

1. margin-top
2. margin-right
3. margin-bottom

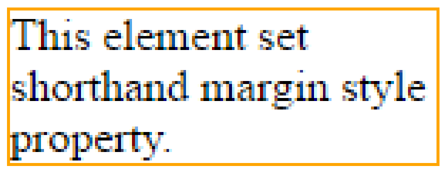
Margin value how to set

Value	Description
margin: 12px;	all 4 side margin 12 pixel.
margin: 10px 20px;	top and bottom margin 10 pixel right side and left side margin 20 pixel.
margin: 10px 20px 30px;	top margin 10 pixel left side and right side margin 20 pixel bottom margin is 30 pixel.
margin: 10px 20px 30px 40px;	top margin is 10 pixel right side margin is 20 pixel bottom margin is 30 pixel left side margin is 40 pixel.

Example:

```
<!DOCTYPE html>
  <html>
     <head>
        <title>CSS margin property</title>
           <style type="text/css">
                p {
         border: 1px solid orange; margin: 25px 25px 5px 50px; width: 150px;
                   }
           </style>
     </head>
  <body>
    <p class="first">
        This element set shorthand margin style property.
    </p>
  </body>
  </html>
```

Output:



This element set shorthand margin style property.

CSS Padding

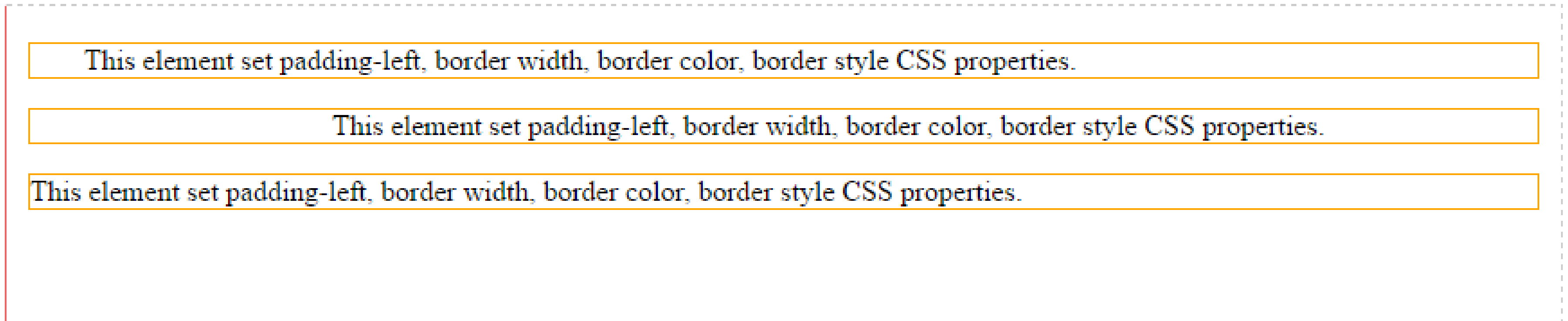
CSS Padding property leave blank space around the element content (inside of border). CSS Padding property support pixel, percentage or auto value.

Properties	**Value**	**Description**
padding	px % auto	User define pixel value. User define percentage value. Set automatic.
padding-left	px % auto	padding left side set define pixel value. padding left side set User define percentage value. padding left side Set automatic.
padding-right	px % auto	padding right side set User define pixel value. padding right side set User define percentage value. padding right side set Set automatic.
padding-top	px % auto	padding top set user define pixel value. padding top set user define percentage value. padding top set automatic.
padding-bottom	px % auto	padding bottom set user define pixel value. padding bottom set user define percentage value. padding bottom set automatic.

Example:

```
<!DOCTYPE html>
  <html>
     <head>
        <title>CSS padding property</title>
           <style type="text/css">
              p.first { border: 1px solid orange; padding-left: 30px; }
              p.second { border: 1px solid orange; padding-left: 20%; }
              p.third { border: 1px solid orange; padding-left: auto; }
           </style>
     </head>
     <body>
        <p class="first">This element set padding-left, border width, border
        color, border style CSS properties.</p>
        <p class="second">This element set padding-left, border width, border
        color, border style CSS properties.</p>
        <p class="third">This element set padding-left, border width, border
        color, border style CSS properties.</p>
     </body>
    </html>
```

Output:



```
      This element set padding-left, border width, border color, border style CSS properties.
                        This element set padding-left, border width, border color, border style CSS properties.
This element set padding-left, border width, border color, border style CSS properties.
```

CSS Padding shorthand property

CSS padding property write in shorthand way including following padding properties:

1. padding-top
2. padding-right
3. padding-bottom
4. padding-left

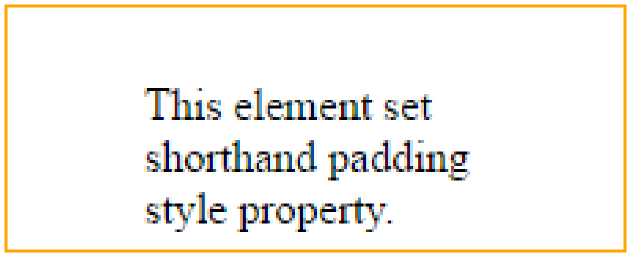
Margin value how to set

Value	Description
padding: 12px;	all 4 side padding 12 pixel.
padding: 10px 20px;	top and bottom padding 10 pixel right side and left side padding 20 pixel.
padding: 10px 20px 30px;	top padding 10 pixel left side and right side padding 20 pixel bottom padding is 30 pixel.
padding: 10px 20px 30px 40px;	top padding is 10 pixel right side padding is 20 pixel bottom padding is 30 pixel left side padding is 40 pixel.

Example:

```
<!DOCTYPE html>
  <html>
    <head>
       <title>CSS padding property</title>
         <style type="text/css">
            p {
               border: 1px solid orange; padding: 25px 25px 50px
50px;              width: 150px; }
         </style>
     </head>
     <body>
       <p class="first">This element set shorthand padding style
                          property.</p>
     </body>
   </html>
```

Output:



```
This element set
shorthand padding
style property.
```

CSS Display

CSS display property use for how to display list of item. Possible value is inline, block, inline-block and so on.

CSS display Property

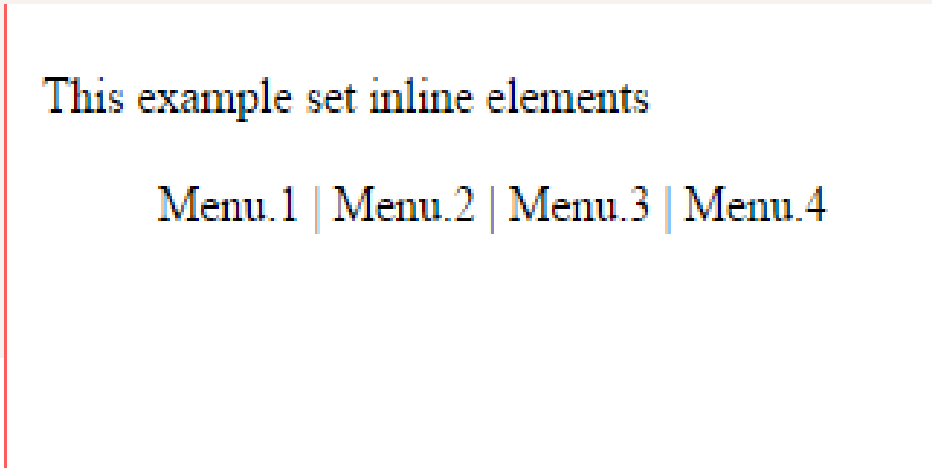
Syntax	Value	Description
display	inline block inline-block none	value specify how to list display inline, block, inline-block and none of display.

CSS display:inline style

CSS display inline means elements displayed inline in current block of line.

Example

```
<!DOCTYPE html>
   <html>
     <head>
        <title>CSS display property</title>
           <style type="text/css">
               li { display: inline; }
           </style>
     </head>
     <body>
       <p>This example set inline elements</p>
          <ul>
             <li>Menu.1</li> |
             <li>Menu.2</li> |
             <li>Menu.3</li> |
             <li>Menu.4</li>
          </ul>
     </body>
   </html>
```



This example set inline elements

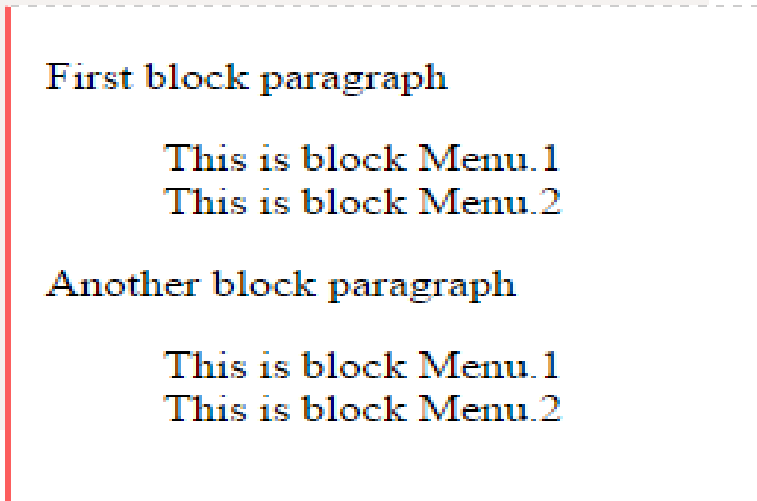
Menu.1 | Menu.2 | Menu.3 | Menu.4

CSS display:block style

CSS display block means elements displayed as a block. Header and Paragraphs are always in block style.

Example

```
  <!DOCTYPE html>
     <html>
       <head>
          <title>CSS display property</title>
          <style type="text/css">
             li { display: block; }
          </style>
       </head>
       <body>
          <p>First block paragraph</p>
          <ul>
            <li>This is block Menu.1</li>
            <li>This is block Menu.2</li>
          </ul>
          <p>Another block paragraph</p>
          <ul>
             <li>This is block Menu.1</li>
             <li>This is block Menu.2</li>
          </ul>
        </body>
     </html>
```



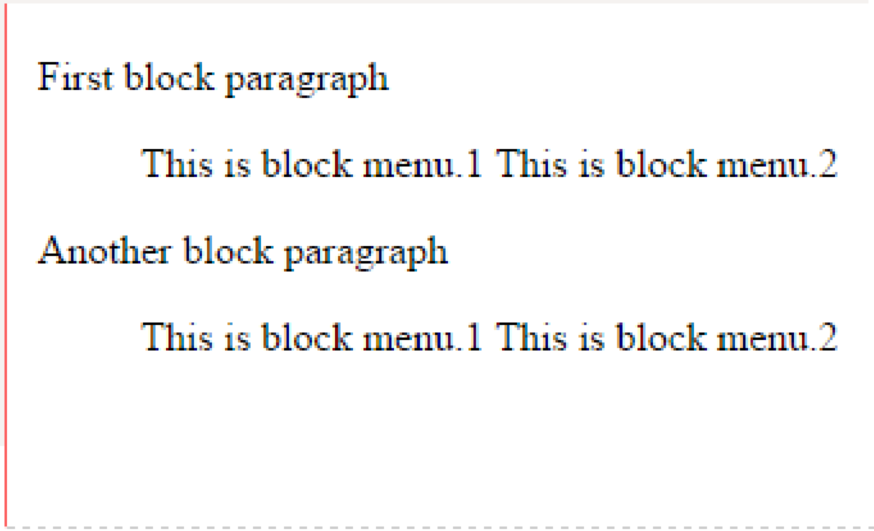
```
First block paragraph

     This is block Menu.1
     This is block Menu.2

Another block paragraph

     This is block Menu.1
     This is block Menu.2
```

CSS display:inline-block style

CSS display inline-block means elements is display as a inline but it's behaves is like block type.

Example

```
<!DOCTYPE html>
<html>
  <head>
     <title>CSS display property</title>
     <style type="text/css">
         li { display:inline-block; }
     </style>
  </head>
  <body>
     <p>First block paragraph</p>
     <ul>
        <li>This is block menu.1</li>
        <li>This is block menu.2</li>
     </ul>
     <p>Another block paragraph</p>
     <ul>
        <li>This is block menu.1</li>
        <li>This is block menu.2</li>
     </ul>
   </body>
</html>
```



First block paragraph

    This is block menu.1 This is block menu.2

Another block paragraph

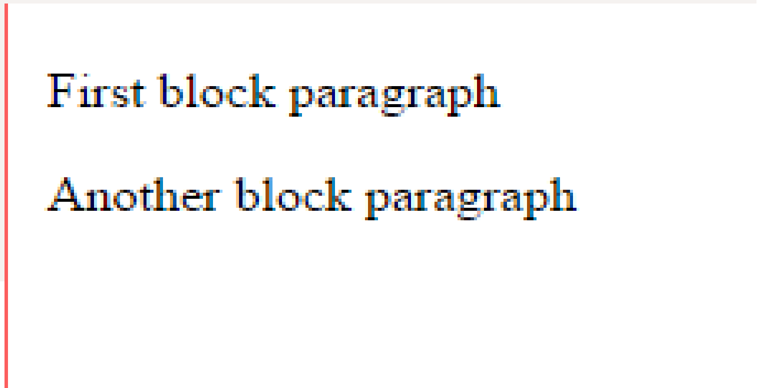
    This is block menu.1 This is block menu.2

CSS display:none style

CSS display type none means element is not display, element is no longer display.

Example

```
<!DOCTYPE html>
<html>
  <head>
     <title>CSS display property</title>
     <style type="text/css">
        li { display:none; }
     </style>
  </head>
  <body>
     <p>First block paragraph</p>
        <ul>
          <li>This is block menu.1</li>
          <li>This is block menu.2</li>
        </ul>
     <p>Another block paragraph</p>
        <ul>
          <li>This is block menu.1</li>
          <li>This is block menu.2</li>
        </ul>
   </body>
</html>
```



First block paragraph

Another block paragraph

CSS Position

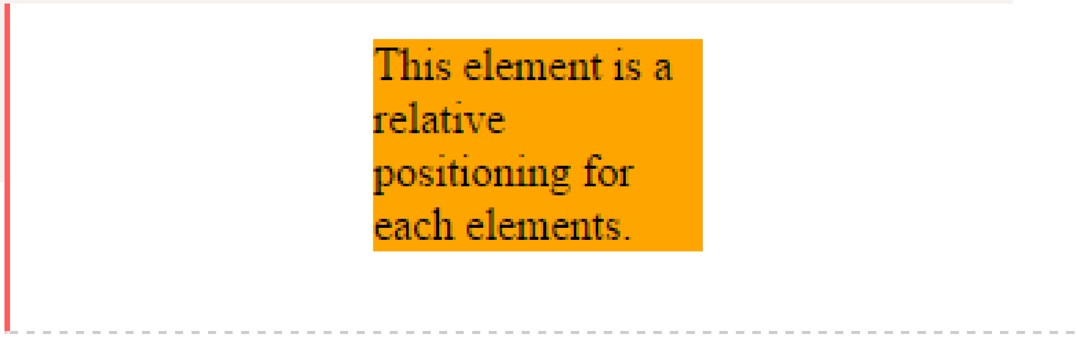
CSS position property set an element positioning to display in web page. CSS position property possible value relative, absolute and fixed.

CSS *position:relative* property

CSS position:relative property set element relatively followed by the relative offset from top, right, bottom or left. Relative position is related to each and every around element properties (like margin, background-color and so forth).

Example

```
<!DOCTYPE html>
 <html>
  <head>
    <title>CSS position property</title>
  </head>
  <body>
    <div style="position: relative; left: 120px; background-color: orange;
        width: 120px;"> This element is a relative positioning for each
                          elements.
    </div>
   </body>
 </html>
```

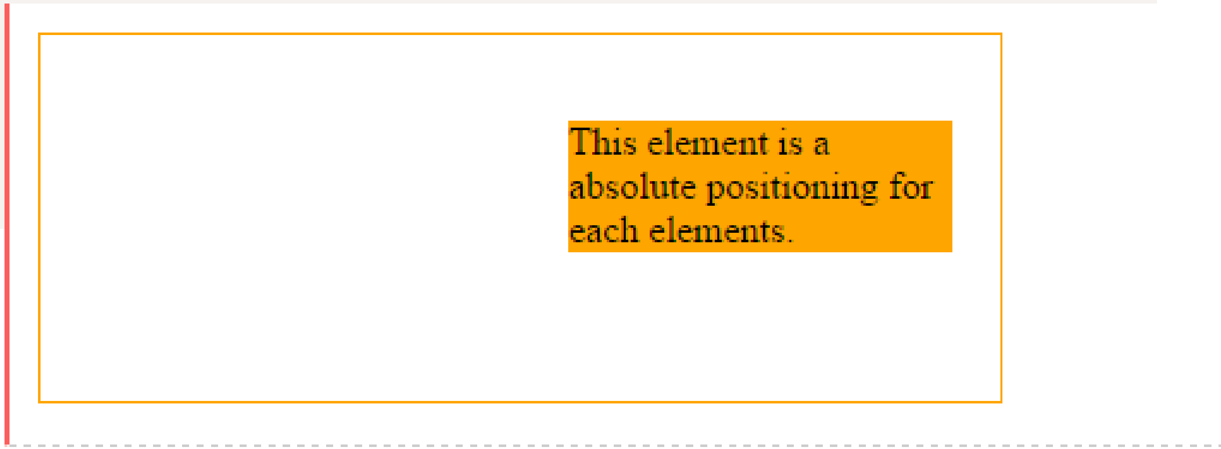

This element is a relative positioning for each elements.

CSS *position:absolute* property

CSS position:absolute property set element absolutely followed by the absolute offset from top, right, bottom or left. Relative positioning element inside you can set absolute position element.

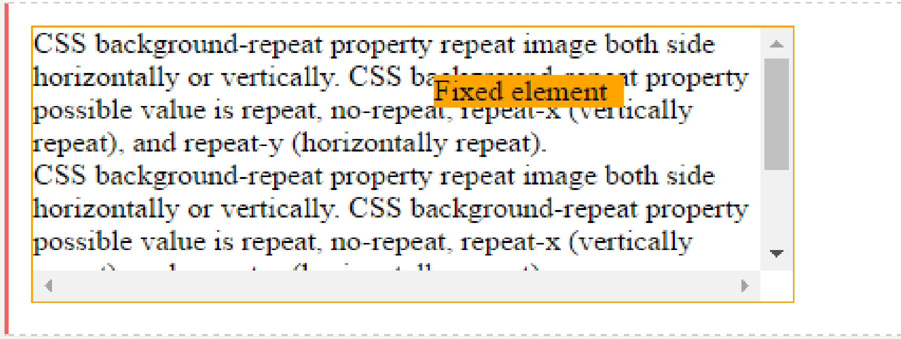
Example

```
<!DOCTYPE html>
 <html>
   <head>
      <title>CSS position property</title>
   </head>
   <body>
    <div style="position: relative; width: 400px; height: 150px;
       border: 1px solid orange;">
      <div style="position: absolute; top: 35px; left: 220px; width:
       160px; background-color: orange;">This element is a absolute
              positioning for each elements.
      </div>
    </div>
   </body>
 </html>
```


This element is a absolute positioning for each elements.

CSS *position:fixed* property

CSS position:fixed property set element in fixed, even if window scroll vertically or horizontally element is fixed place. CSS position fixed followed by the window offset top, right, bottom, left.


Fixed element
CSS background-repeat property repeat image both side horizontally or vertically. CSS background-repeat property possible value is repeat, no-repeat, repeat-x (vertically repeat), and repeat-y (horizontally repeat).
CSS background-repeat property repeat image both side horizontally or vertically. CSS background-repeat property possible value is repeat, no-repeat, repeat-x (vertically

Example

```
<!DOCTYPE html>
  <html>
    <head>
        <title>CSS position property</title>
    </head>
    <body>
        <div style="width:400px; height:150px; border: 1px solid orange; overflow:
                 scroll;">
         <div style="position: fixed; top: 35px; left: 220px; width: 100px; background-
                 color: orange;">Fixed element</div>
                 CSS background-repeat property repeat image both side horizontally
                 or vertically. CSS background-repeat property possible value is
                 repeat, no-repeat, repeat-x (vertically repeat), and repeat-y
                 (horizontally repeat).<br />
                 CSS background-repeat property repeat image both side horizontally
                 or vertically.CSS background-repeat property possible value is
                 repeat, no-repeat, repeat-x (vertically repeat), and repeat-y
                 (horizontally repeat).<br />
                 CSS background-repeat property repeat image both side horizontally
                 or vertically. CSS background-repeat property possible value is
                 repeat, no-repeat, repeat-x (vertically repeat), and repeat-y
                 (horizontally repeat). </div>
     </body>
    </html>
```

CSS - Cursors

The *cursor* property of CSS allows you to specify the type of cursor that should be displayed to the user.

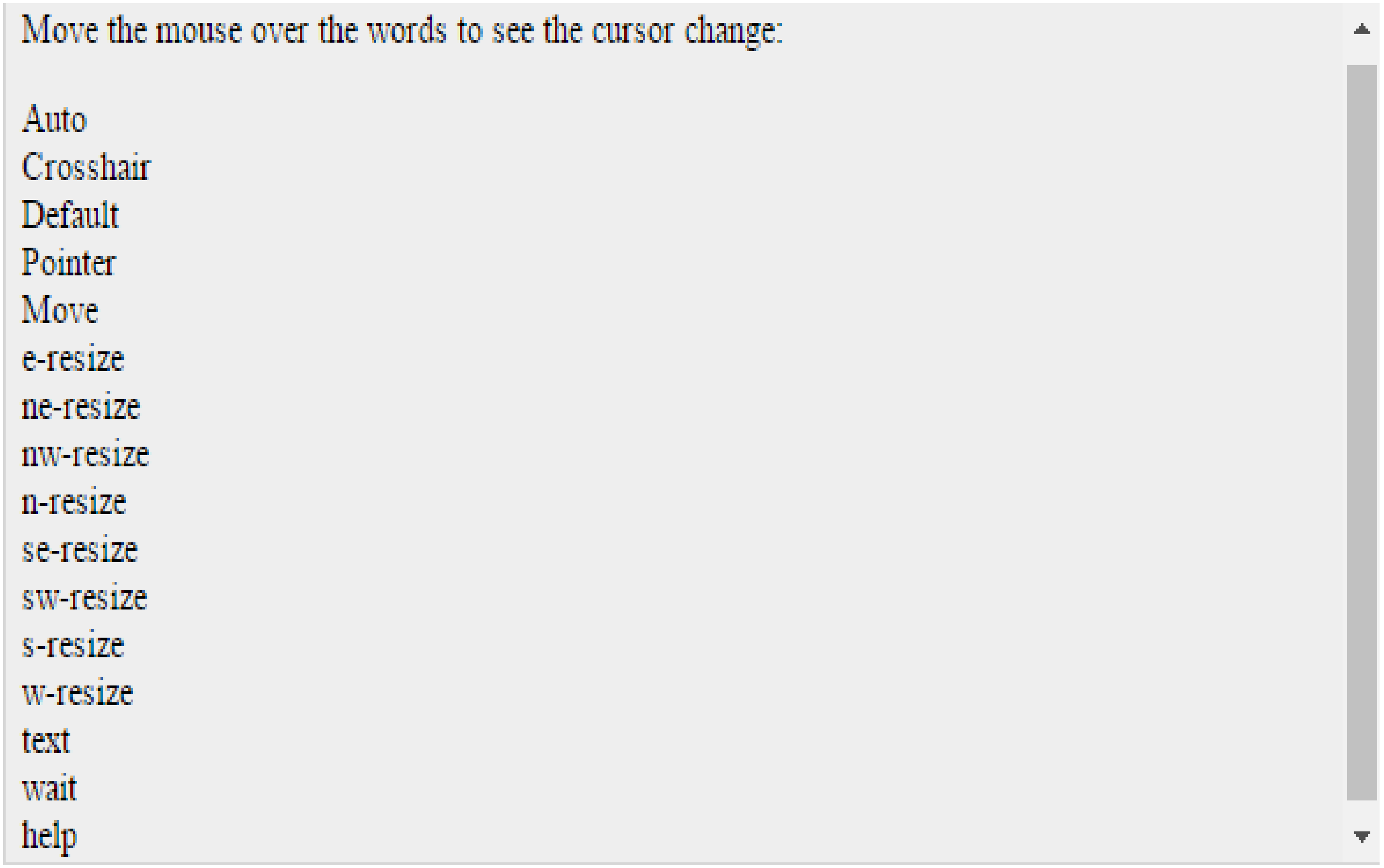
Sr.No.	Value & Description
1	**auto** Shape of the cursor depends on the context area it is over. For example an I over text, a hand over a link, and so on...
2	**crosshair** A crosshair or plus sign
3	**default** An arrow
4	**pointer** A pointing hand (in IE 4 this value is hand)
5	**move** The I bar
6	**e-resize** The cursor indicates that an edge of a box is to be moved right (east)
7	**ne-resize** The cursor indicates that an edge of a box is to be moved up and right (north/east)
8	**nw-resize** The cursor indicates that an edge of a box is to be moved up and left (north/west)

Sr.No.	Value & Description
9	**n-resize** The cursor indicates that an edge of a box is to be moved up (north)
10	**se-resize** The cursor indicates that an edge of a box is to be moved down and right (south/east)
11	**sw-resize** The cursor indicates that an edge of a box is to be moved down and left (south/west)
12	**s-resize** The cursor indicates that an edge of a box is to be moved down (south)
13	**w-resize** The cursor indicates that an edge of a box is to be moved left (west)
14	**text** The I bar
15	**wait** An hour glass
16	**help** A question mark or balloon, ideal for use over help buttons
17	**<url>** The source of a cursor image file

Example:

```
<html>
   <head>
   </head>
   <body>
      <p>Move the mouse over the words to see the cursor change:</p>
             <div style="cursor:auto">Auto</div>
              <div style="cursor:crosshair">Crosshair</div>
             <div style="cursor:default">Default</div>
              <div style="cursor:pointer">Pointer</div>
              <div style="cursor:move">Move</div>
              <div style="cursor:e-resize">e-resize</div>
              <div style="cursor:ne-resize">ne-resize</div>
              <div style="cursor:nw-resize">nw-resize</div>
              <div style="cursor:n-resize">n-resize</div>
              <div style="cursor:se-resize">se-resize</div>
              <div style="cursor:sw-resize">sw-resize</div>
              <div style="cursor:s-resize">s-resize</div>
              <div style="cursor:w-resize">w-resize</div>
              <div style="cursor:text">text</div>
              <div style="cursor:wait">wait</div>
              <div style="cursor:help">help</div>
     </body>
</html>
```

Output:



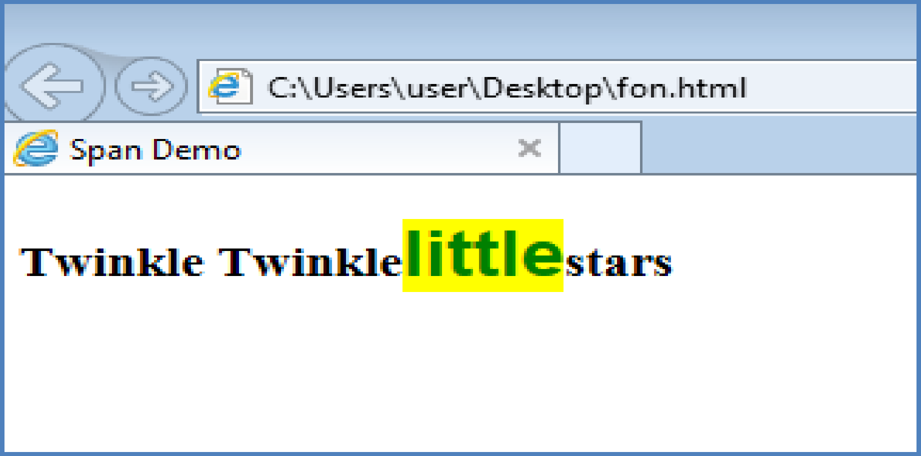
```
Move the mouse over the words to see the cursor change:

Auto
Crosshair
Default
Pointer
Move
e-resize
ne-resize
nw-resize
n-resize
se-resize
sw-resize
s-resize
w-resize
text
wait
help
```

<span> and <div> Tags

<span> Tag:

It is used to group elements for styling purposes so that some text which are grouped will appear differently in a paragraph. That means we can differentiate particular text using font-size, color, font-family, font-style and so on.

```
<!DOCTYPE html> <html>
<head>
<title> Span Demo </title>
<style type = "text/css">
.myspan
{ font-size:28px; font-family:Arial;
  color:green; background-color:yellow;
}
</style>
</head>
<body>
<h3>Twinkle Twinkle<span class="myspan">little</span>stars</h3>
</body>
</html>
```

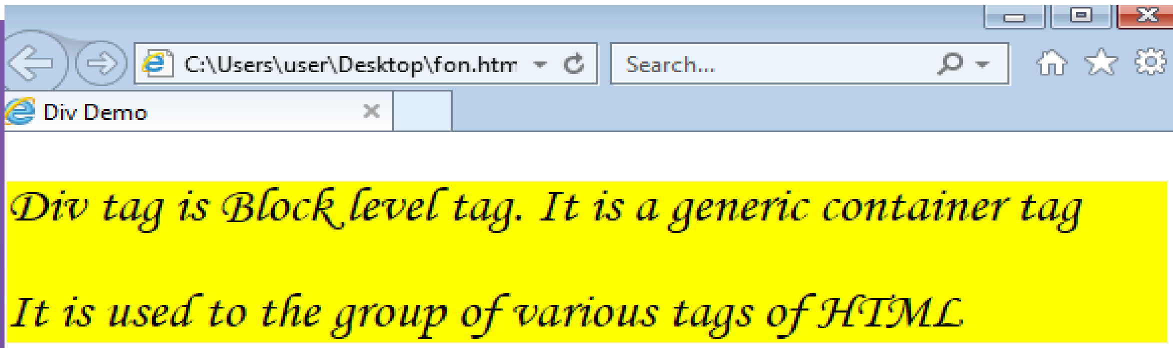


Twinkle Twinklelittlestars

<div> Tag:

Div tag is similar to span tag where div is a **block-level** tag but span is an **inline tag**. The div tag is known as Division tag. The div tag is used to make divisions of content in the web page like (text, images, header, footer, navigation bar, etc). The Div is the most usable tag in web development because it helps us to separate out data in the web page and we can create a particular section for particular data or function in the web pages.

```
<!DOCTYPE html> <html> <head>
<title> Div Demo </title>
<style type = "text/css">
.mysection
{ font-size:28px;
font-family:'monotype corsiva';
  background-color:yellow;}
</style>  </head>
<body>
 <div class="mysection">
<p> Div tag is Block level tag. It is a generic container tag</p>
<p>It is used to the group of various tags of HTML .</p>
</body>
</html>
```



Div tag is Block level tag. It is a generic container tag

It is used to the group of various tags of HTML

References

- https://way2tutorial.com/html/tutorial.php
- https://coursesweb.net/html
- https://www.geeksforgeeks.org/html-tutorials/
- https://www.tutorialride.com/html/html-tutorial.htm
- https://www.w3schools.com/html/
- https://www.tutorialspoint.com/html/index.htm
- https://tutorialehtml.com/en/html-tutorial-complete-html-guide/

Thank You !!

UNIT-2

XML

Contents

Introduction to SGML – features of XML - XML as a subset of SGML – XML Vs HTML – Views of an XML document - Syntax of XML- XML Document Structure – Namespaces- XML Schemas- simple XML documents – Different forms of markup that can occur in XML documents - Document Type declarations – Creating XML DTDs – Displaying XML Data in HTML browser – Converting XML to HTML with XSL minimalist XSL style sheets – XML applications

SGML(ISO 8879)

- **S**tandard **G**eneralized **M**arkup **L**anguage is a meta-markup language which means it allows us to create our own language or our own tags.
- The international standard for defining descriptions of structure and content in text documents.
- Interchangeable: device-independent, system-independent, tags are not predefined.
- Using DTD to validate the structure of the document
- Large, powerful, and very complex
- Heavily used in industrial and commercial usages for over a decade

XML Versions

- 1986 – SGML approved as International Standard Organisation (ISO) Standard.
- 1990 – SGML used as basis for development of HTML.
- 1998 – XML 1.0_First version of XML published in February.

What is XML?

- XML stands for eXtensible Markup Language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.
- XML is a markup language much like HTML
- XML was designed to store and transport data
- XML was designed to be self-descriptive
- XML is designed to carry data, not to display data.
- XML tags are not predefined. You must define your own tags.
- XML is platform independent and language independent.

Features of XML - 1

- **Allows XML validation**
  - A XML document can be validated using DTD or XML schema. This ensures that the XML document is syntactically correct and avoids any issues that may arise due to the incorrect XML.
- **XML focuses on data rather than how it looks**
  - XML is popular because it focuses on data rather than data presentation. his separates the data and its presentation part and gives us the freedom to present the data the way we want.
- **Easy and efficient data sharing**
  - Since XML is **software and hardware independent**, it is easier to share data between different systems with different hardware and software configuration. Any system with any programming language can read and process a XML document.
- **Compatibility with other markup language HTML**
  - It is so much easier to read the data from XML and display it on an GUI(graphical user interface) using HTML markup language.

Features of XML - 2

- **Supports platform transition**
  - XML simplifies the data is transportation on new upgraded systems without any data loss.
- **Adapts technology advancements**
  - it can adapt to the new technologies because of its platform-independent nature.
- **XML supports Unicode**
  - XML supports Unicode that allows it to communicate almost any information in any written human language.
- **XML can be used to create new internet languages**
  - A lot of new Internet languages are created with XML

HTML vs XML

No.	**HTML**	**XML**
1)	HTML is used **to display data** and focuses on how data looks.	XML is a software and hardware independent tool used **to transport and store data**. It focuses on what data is.
2)	HTML is a **markup language** itself.	XML provides a **framework to define markup languages**.
3)	HTML is **not case sensitive**.	XML is **case sensitive**.
4)	HTML is a presentation language.	XML is neither a presentation language nor a programming language.
5)	HTML **has its own predefined tags**.	You **can define tags according to your need**.
6)	In HTML, it is **not necessary to use a closing tag**.	XML **makes it mandatory to use a closing tag**.
7)	HTML is **static** because it is used to display data.	XML is **dynamic** because it is used to transport data.
8)	HTML **does not preserve whitespaces**.	XML **preserve whitespaces**.

XML Components

The most basic components of an XML document are :

- Elements
- Control Information
- Entities

Elements:

- Elements are used to mark up the sections of an XML document. Elements are the basic units used to identify and describe the data in XML. They are the building blocks of an XML document. Elements are represented using tags. An XML element has the following form:

  <ElementName>Content</ElementName>

- The **content** is contained within the XML tags. Content refers to the information represented by the elements of an XML document.

- **Empty Elements:**Although XML tags usually enclose content, you can also have elements that have no content, called *empty elements*. In XML, an empty element can be represented as follows:

  <ElementName/>

- **Nested Elements:**Elements can be nested. For example, if you wanted to group all the patient information under a single *Patient* element, you might want to rewrite the patient record example as follows:

```
<Patient>
   <PatientName>John Smith</PatientName>
   <PatientAge>108</PatientAge>
   <PatientWeight>155</PatientWeight>
 </Patient>
```

- Thus XML elements can contain other elements. However, the elements must be strictly nested: each start tag must have a corresponding end tag.

- **Attributes:** Attributes provide additional information about the elements for which they are declared. An attribute consists of a name-value pair. Consider the following example:

  <Student_name S_ID = “101”>shanshak </ Student_name >

Control Information:

This gives control to the XML document.

- **Comments:** Comments are statements used to explain the XML code. They are used to provide documentation information about the XML file or the application to which the file belongs. The parser ignores comments entries during code execution.

  `<!--Comment here-->`

- **Processing Instructions:** An XML Documents usually begins with the XML declaration statement called the Processing Instructions .This statement provides information on how the XML file should be processed. e.g. <?xml version =”1.0” encoding=”UTF-8”?> The Processing Instruction statement uses the encoding property to specify the encoding scheme used to create the XML file

- **DTD:**DTD stands for Document Type Definition. It defines the legal building blocks of an XML document. It is used to define document structure with a list of legal elements and attributes. Each XML has associated DTD held in a separate file so that it can be used with many document. DTD file holds the rules of grammar for a particular XML datastructure. Rules are used by validating parsers to check.

  `<!DOCTYPE employee SYSTEM "employee.dtd">`

- **CDATA:** (Unparsed Character data): CDATA contains the text which is not parsed further in an XML document. Tags inside the CDATA text are not treated as markup and entities will not be expanded.

```
<?xml version="1.0"?>
<!DOCTYPE employee SYSTEM "employee.dtd">
<employee>
<![CDATA[
  <firstname>vimal</firstname>
  <lastname>jaiswal</lastname>
  <email>vimal@javatpoint.com</email>
]]>
</employee>
```

- **PCDATA:** (Parsed Character Data): XML parsers are used to parse all the text in an XML document. PCDATA stands for Parsed Character data. PCDATA is the text that will be parsed by a parser. Tags inside the PCDATA will be treated as markup and entities will be expanded.

  employee.xml

```
<?xml version="1.0"?>
<!DOCTYPE employee SYSTEM "employee.dtd">
<employee>
  <firstname>vimal</firstname>
  <lastname>jaiswal</lastname>
  <email>vimal@javatpoint.com</email>
</employee>
```

  employee.dtd

```
<!ELEMENT employee (firstname,lastname,email)>
<!ELEMENT firstname (#PCDATA)>
<!ELEMENT lastname (#PCDATA)>
<!ELEMENT email (#PCDATA)>
```

Entities:

An entity is a name that is associated with a block of data, such as chunk of text or a reference to an external file that contains textual or binary information. It is a set of information that can be specifying a single name.

XML Document Structure

XML document is a tree structure which contain exactly one root element which is the start tag of the XML document and it contains all other elements. All elements in an XML document can contain sub elements, text and attributes. The tree represented by an XML document starts at the root element and branches to the lowest level of elements.

```
<root>
 <section>
  <sub-section></sub-section>
  <sub-section></sub-section>
 </section>

 <section>
  <sub-section></sub-section>
  <sub-section></sub-section>
 </section>
<root>
```

XML Declaration

- XML documents must begin with a prolog that appears before the root element. It has the metadata about the XML document such as character encoding, document structure an stylesheets.

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

XML Syntax Rules -1

- **All XML documents must have a root element.**
  - A root element is simply a set of tags that contains your XML content.

```
<root>
  <author>Ernest Hemingway</author>
  <author>John Steinbeck</author>
  <author>James Joyce</author>
</root>
```

- **All XML elements must have a closing tag.**
  - When a tag is declared (opened), it must also be closed. Any unclosed tags will break the code. Even tags that don’t need to be closed in HTML must be closed in XML or XHTML. To open a tag, type the name of the element between less-than (<) and greater-than (>) characters, like this opening tag:<author>.To close a tag, repeat the opening tag exactly, but insert a slash in front of the tag name, like this closing tag:</author>.Even empty tags, such as <hr> and <br>, must be closed.

<p>This is another paragraph</p>

XML Syntax Rules -2

- **Tag names have strict limits.**
  - Tag names can't start with the letters *xml*, a number, or punctuation, except for the underscore character (_).
  - The letters *XML* are used in various commands and can't start your tag name. Numbers and punctuation also aren't allowed in the beginning of the tag name.

    <author> or <_author>
- **XML tags are case sensitive.**
  - Uppercase and lowercase matter in XML. Opening and closing tags must match exactly. For example, <ROOT>, <Root>, and <root> are three different tags.

    <author>Hemingway</author> or

    <AUTHOR>Hemingway</AUTHOR>
- **Tag Names cannot contain spaces**
  - Spaces in tag names can cause all sorts of problems with data-intensive applications, so they're prohibited in XML.
-

XML Syntax Rules -3

- Attribute values must always be quoted.
  - Attribute values modify a tag or help identify the type of information being tagged. If you’re a web designer, you may be used to the flexibility of HTML, in which some attributes don’t require quotes. In XML, all attribute values must appear within quotes. For example

```
<chapter number="1">
<artist title="author" nationality="USA">
```

- White Space is preserved
  - XML does not truncate multiple white-spaces (HTML truncates multiple white-spaces to one single white-space)

XML Namespace

- Sometimes we need to create two different elements by the same name. The XML document allows us to create different elements which are having the common name. This technique is known as namespace. XML **Namespace** is used *to avoid element name conflict* in XML document.

XML Namespace Declaration:

- An XML namespace is declared using the reserved XML attribute. This attribute name must be started with "xmlns".

```
<element xmlns:name = "URL">
```

- Here, namespace starts with keyword **"xmlns"**. The word **name** is a namespace prefix. The **URL** is a namespace identifier.
- To avoid these types of confliction we use XML Namespaces. We can say that XML Namespaces provide a method to avoid element name conflict.
- Generally these conflict occurs when we try to mix XML documents from different XML application.

Let's take an example with two tables:

Table1:

```
<table>
  <tr>
    <td>Aries</td>
    <td>Bingo</td>
  </tr>
</table>
```

Table2: This table carries information about a computer table.

```
<table>
  <name>Computer table</name>
  <width>80</width>
  <length>120</length>
</table>
```

If you add these both XML fragments together, there would be a name conflict because both have <table> element. Although they have different name and meaning.

How to get rid of name conflict?

1) By Using a Prefix

You can easily avoid the XML namespace by using a name prefix.

```
<h:table>
  <h:tr>
    <h:td>Aries</h:td>
    <h:td>Bingo</h:td>
  </h:tr>
</h:table>
<f:table>
  <f:name>Computer table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
```

In this example, you will get no conflict because both the tables have specific names.

2) By Using xmlns Attribute

You can use xmlns attribute to define namespace with the following syntax:

```
<element xmlns:name = "URL">
```

Let's see the example:

```
<root>
<h:table xmlns:h="http://www.abc.com/TR/html4/">
  <h:tr>
    <h:td>Aries</h:td>
<h:td>Bingo</h:td>
  </h:tr>
</h:table>

<f:table xmlns:f="http://www.xyz.com/furniture">
  <f:name>Computer table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
</root>
```

In the above example, the <table> element defines a namespace and when a namespace is defined for an element, the child elements with the same prefixes are associated with the same namespace.

```
<root xmlns:h="http://www.abc.com/TR/html4/"
xmlns:f="http://www.xyz.com/furniture">
<h:table>
  <h:tr>
    <h:td>Aries</h:td>
    <h:td>Bingo</h:td>
  </h:tr>
</h:table>
<f:table>
  <f:name>Computer table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
</root>
```

The Namespace URI used in the above example is not necessary at all. It is not used by parser to look up information. It is only used to provide a unique name to the Namespace identifier.

Uniform Resource Identifier (URI):

- ✓ Uniform Resource Identifier is used to identify the internet resource. It is a string of characters.
- ✓ The most common URI is URL (Uniform Resource Locator) which identifies an internet domain address.
- ✓ There is also an URI name URN (Universal Resource Name) but it is not so common.

The Default Namespace:

- The default namespace is used in the XML document to save you from using prefixes in all the child elements. The only difference between default namespace and a simple namespace is that: There is no need to use a prefix in default namespace. You can also use multiple namespaces within the same document just define a namespace against a child node.

  <u>Example of Default Namespace:</u>

  ```
  <tutorials xmlns="http://www.javatpoint.com/java-tutorial">
    <tutorial>
      <title>Java-tutorial</title>
      <author>Sonoo Jaiswal</author>
    </tutorial>
    ...
  </tutorials>
  ```

- You can see that prefix is not used in this example, so it is a default namespace. If you define a namespace without a prefix, all descendant elements are considered to belong to that namespace.

DTD

- A DTD is a Document Type Definition.
- A DTD defines the structure and the legal elements and attributes of an XML document. DTDs check the validity of structure and vocabulary of an XML document against the grammatical rules of the appropriate XML language.
- With a DTD, independent groups of people can agree on a standard DTD for interchanging data.
- An application can use a DTD to verify that XML data is valid.

Types of DTD:

DTD can be classified on its declaration basis in the XML document, such as –

- Internal DTD
- External DTD

An Internal DTD Declaration:

- If the DTD is declared inside the XML file, it must be wrapped inside the <!DOCTYPE> definition:

```
<?xml version="1.0"?>
<!DOCTYPE note [
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend</body>
</note>
```

- The DTD above is interpreted like this:
  - **!DOCTYPE note** defines that the root element of this document is note
  - **!ELEMENT note** defines that the note element must contain four elements: "to,from,heading,body"
  - **!ELEMENT to** defines the to element to be of type "#PCDATA"
  - **!ELEMENT from** defines the from element to be of type "#PCDATA"
  - **!ELEMENT heading** defines the heading element to be of type "#PCDATA"
  - **!ELEMENT body** defines the body element to be of type "#PCDATA"

An External DTD Declaration:

- If the DTD is declared in an external file, the <!DOCTYPE> definition must contain a reference to the DTD file:

```
<?xml version="1.0"?>
<!DOCTYPE note SYSTEM "note.dtd">
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

- And here is the file "note.dtd", which contains the DTD:

```
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

Merits of DTD:

1.DTDs are used to define the structural components of XML document.

2. These are relatively simple and compact.

3. DTDs can be defined inline and hence can be embedded directly in the XML document.

4. Documentation − You can define your own format for the XML files. Looking at this document a user/developer can understand the structure of the data.

5.Validation − It gives a way to check the validity of XML files by checking whether the elements appear in the right order, mandatory elements and attributes are in place, the elements and attributes have not been inserted in an incorrect way, and so on.

Demerits of DTD:

1.The DTDs are very basic and hence cannot be much specific for complex document.

2.The language that DTD uses is not an XML document. Hence various frameworks used by XML cannot be supported by the DTDs.

3.The DTD cannot define the type of data contained within the XML document. Hence we cannot specify whether the element is numeric or string data type.

4.There are some XML processor which do not understand DTDs.

5.The DTDs are not aware of namespace concept. It does not support the namespaces

XML Schemas

- **XML Schema** is commonly known as **XML Schema Definition** (**XSD**) which is a language ffor XML Schema.
- It is used to describe and validate the structure and the content of **XML** data.
- **XML schema** defines the elements, attributes and data types.
- **Schema** element supports Namespaces.
- It is similar to a database **schema** that describes the data in a database.
- It also allows the developer to use data types.
- This can be used as an alternative to XML DTD.

DTD vs XSD

No.	DTD	XSD
1)	DTD stands for**Document Type Definition**.	XSD stands for XML Schema Definition.
2)	DTDs are derived from**SGML** syntax.	XSDs are written in XML.
3)	DTD **doesn't support datatypes**.	XSD **supports datatypes** for elements and attributes.
4)	DTD **doesn't support namespace**.	XSD **supports namespace**.
5)	DTD **doesn't define order** for child elements.	XSD **defines order** for child elements.
6)	DTD is **not extensible**.	XSD is **extensible**.
7)	DTD is **not simple to learn**.	XSD is **simple to learn** because you don't need to learn new language.

Advantages of XML schemas over DTDs

i.**XML schema use basic XML syntax** XML schemas are created by using XML syntax whereas DTD’s use separate syntax.

ii.**XML schema support namespace** XML schemas support namespace functionality, but DTDs doesn’t support this functionality completely. They also allow the usage of multiple namespaces in XML documents with less rigidity. For example, while designing an XML schema the prefixes of namespace are not required since the end-user to decide it. But, in DTDs the namespace prefixes are essential to be specified.

iii.**XML schema allow the validation of text elements based on datatypes** XML schemas specify the type of textual data that can be used within attributes and elements. This is done by the simple type declarations . Hence XML schemas can control the documents more rigidly. The most important feature of XML schemas is that , it include the commonly used simple types

iv.**XML schema allows the creation of complex and reusable content models easily** In a DTD , content model can be reused only when the utilization of parameter entities is allowed . But , this may lead to some situations where the parts of DTD are difficult to be reusable. XML schemas provide a wide variety of mechanisms to reuse the content models and also model some complex programming concepts easily.

Description of XML Schema

- **<xs:element name="employee">** : It defines the element name employee.
- **<xs:complexType>** : It defines that the element 'employee' is complex type.
- **<xs:sequence>** : It defines that the complex type is a sequence of elements.
- **<xs:element name="firstname" type="xs:string"/>** : It defines that the element 'firstname' is of string/text type.
- **<xs:element name="lastname" type="xs:string"/>** : It defines that the element 'lastname' is of string/text type.
- **<xs:element name="email" type="xs:string"/>** : It defines that the element 'email' is of string/text type

XML Schema Data types

- There are two types of data types in XML schema.
  - simpleType
  - complexType
- SimpleType
  - The simpleType allows you to have text-based elements. It contains less attributes, child elements, and cannot be left empty.
    - <xs:element name="Customer_dob"   type="xs:date" />
    - <xs:element name="Customer_address" type="xs:string" />
    - <xs:element name="Supplier_phone"   type="xs:integer" />
    - <xs:element name="Supplier_address" type="xs:string" />
- ComplexType
  - The complexType allows you to hold multiple attributes and elements. It can contain additional sub elements and can be left empty.
    - <xs:element name = "name" type = "xs:string" />
    - <xs:element name = "company" type = "xs:string" />

XML Schema Example

employee.xsd

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.javatpoint.com"
xmlns="http://www.javatpoint.com"
elementFormDefault="qualified">

<xs:element name="employee">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
      <xs:element name="email" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

employee.xml

```
<?xml version="1.0"?>
<employee
xmlns="http://www.javatpoint.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.javatpoint.com employee.xsd">

  <firstname>vimal</firstname>
  <lastname>jaiswal</lastname>
  <email>vimal@javatpoint.com</email>
</employee>
```

Global Types

- With the global type, you can define a single type in your document, which can be used by all other references. For example, suppose you want to generalize the *person* and *company* for different addresses of the company. In such case, you can define a general type as follows −

```
– <xs:element name = "AddressType">
–    <xs:complexType>
–       <xs:sequence>
–          <xs:element name = "name" type = "xs:string" />
–          <xs:element name = "company" type = "xs:string" />
–       </xs:sequence>
–    </xs:complexType>
– </xs:element>
```

- Now let us use this type in our example as follows −

```
– <xs:element name = "Address1">
–    <xs:complexType>
–       <xs:sequence>
–          <xs:element name = "address" type = "AddressType" />
–          <xs:element name = "phone1" type = "xs:int" />
–       </xs:sequence>
–    </xs:complexType>
– </xs:element>

– <xs:element name = "Address2">
–    <xs:complexType>
–       <xs:sequence>
–          <xs:element name = "address" type = "AddressType" />
–          <xs:element name = "phone2" type = "xs:int" />
–       </xs:sequence>
–    </xs:complexType>
– </xs:element>
```

- Instead of having to define the name and the company twice (once for *Address1* and once for *Address2*), we now have a single definition. This makes maintenance simpler, i.e., if you decide to add "Postcode" elements to the address, you need to add them at just one place.

Attributes

- Attributes in XSD provide extra information within an element. Attributes
  have *name* and *type* property as shown below –
- Syntax:

```
<xs:attribute name = "Name_of_attribute" type = "data_type"/>
```

- Example:

```
<xs:attribute name = "designation" type = "xs:string"/>
```

XSL

- XSL is a language for expressing style sheets. An XSL style sheet is, like with CSS, a file that describes how to display an XML document of a given type. XSL shares the functionality and is compatible with CSS2 (although it uses a different syntax). It also adds:
- a. A transformation language for XML documents: **XSLT**.
- b. A navigation language for XML documents: XPath.
- c. A formatting language for XML documents: XSL-FO

XSLT

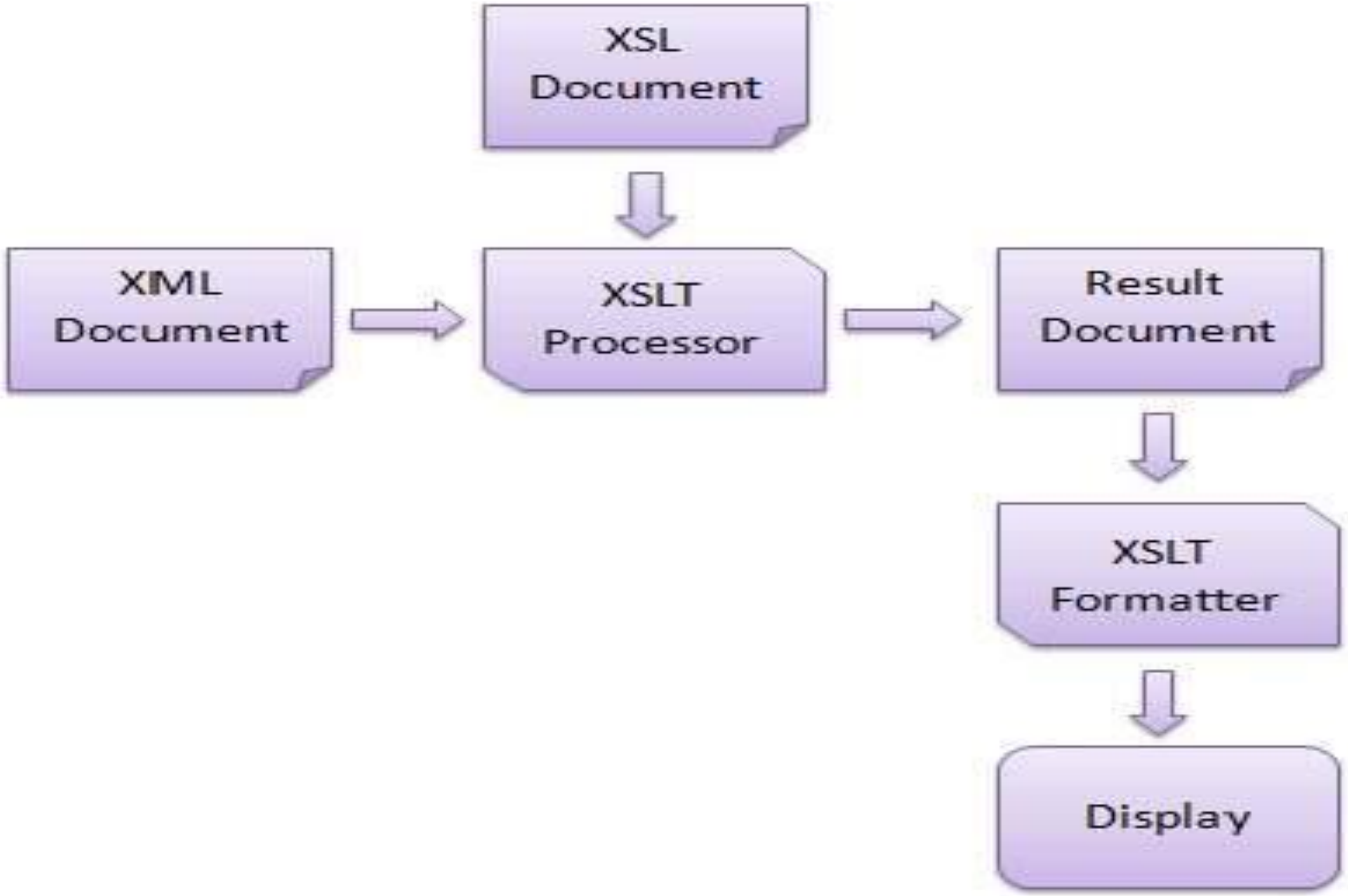
XSLT, Extensible Stylesheet Language Transformations, provides the ability to transform XML data from one format to another automatically.

How XSLT Works

An XSLT stylesheet is used to define the transformation rules to be applied on the target XML document. XSLT stylesheet is written in XML format. XSLT Processor takes the XSLT stylesheet and applies the transformation rules on the target XML document and then it generates a formatted document in the form of XML, HTML, or text format. This formatted document is then utilized by XSLT formatter to generate the actual output which is to be displayed to the end-user.

Advantages

- Independent of programming. Transformations are written in a separate xsl file which is again an XML document.
- Output can be altered by simply modifying the transformations in xsl file. No need to change any code. So Web designers can edit the stylesheet and can see the change in the output quickly.


XSL Document
XIML Document
XSLT Processor
Result Document
XSLT Formatter
Display

XSLT Elements

XSLT <xsl:element>

- The <xsl:element> element is used to create and name an element node in the output document.
- <u>Syntax</u>

```
<xsl:element
name="element_name"
namespace="URI"
use-attribute-sets="namelist">
  <!-- Content:template -->
</xsl:element>
```

XSLT <template> Element

- A template contains rules to apply when a specified node is matched.
- Following is the syntax declaration of <xsl:template> element:

```
<xsl:template name= Qname match = Pattern priority = number mode
= QName>
…
</xsl:template>
```

XSLT Elements

XSLT <xsl:apply-template> Element

- The XSLT <xsl:apply-template> element is used to tell XSLT processor to find the appropriate template to apply according to the type and context of each selected node.

```
<xsl:apply-template
  select = Expression
  mode = QName>
</xsl:apply-template>
```

XSLT <xsl:value-of> Element

- The XSLT <xsl:value-of> element is used to extract the value of selected node. It puts the value of selected node as per XPath expression, as text.

```
<xsl:value-of
  select = Expression
  disable-output-escaping = "yes" | "no">
</xsl:value-of>
```

XSLT Elements

- **XSLT <xsl:for-each> Element**
- The XSLT <xsl:for-each> element is used to apply a template repeatedly for each node.

```
<xsl:for-each
  select = Expression>
</xsl:for-each>
```

Example:

```
<xsl:for-each select="class/employee">
       <tr>
         <td>
           <!-- value-of processing instruction
           process the value of the element matching the XPath expression
           -->
           <xsl:value-of select = "@id"/>
         </td>
         <td><xsl:value-of select = "firstname"/></td>
         <td><xsl:value-of select = "lastname"/></td>
         <td><xsl:value-of select = "nickname"/></td>
         <td><xsl:value-of select = "salary"/></td>
       </tr>
     </xsl:for-each>
```

XSLT Elements

XSLT <xsl:sort> Element

- The XSLT <xsl:sort> element is used to specify a sort criteria on the nodes. It displays the output in sorted form.
- The <xml:sort> element is added inside the <xsl:for-each> element in the XSL file, to sort the output.

```
<xsl:sort
  select = string-expression
  lang = { nmtoken }
  data-type = { "text" | "number" | QName }
  order = { "ascending" | "descending" }
  case-order = { "upper-first" | "lower-first" } >
</xsl:sort>
```

XSLT <xsl:if> Element

- The XSLT <xsl:if> element is used to specify a conditional test against the content of the XML file.

```
<xsl:if test="expression">
  ...some output if the expression is true...
</xsl:if>
```

XSLT Elements

XSLT <xsl:choose> Element

- The XSLT <xsl:choose> element is used to specify a multiple conditional test against the content of nodes with the <xsl:otherwise> and <xsl:when> elements.

```
<xsl:choose>
 <xsl:when test="expression">
  ... some output ...
 </xsl:when>
 <xsl:otherwise>
  ... some output ....
 </xsl:otherwise>
</xsl:choose>
```

Example:

```
<xsl:choose>
        <xsl:when test = "salary > 50000">
        High
        </xsl:when>

        <xsl:when test = "salary > 40000">
        Medium
        </xsl:when>

        <xsl:otherwise>
        Low
        </xsl:otherwise>
        </xsl:choose>
```

XSLT Elements

- **XSLT <xsl:import> Element**
- The XSLT <xsl:import> element is used to import the content of one stylesheet to another stylesheet. The importing stylesheet has higher precedence over imported stylesheet.

  <xsl:import href = "uri">

  </xsl:import>

References

- https://www.w3schools.com/xml/
- https://www.tutorialspoint.com/xml/index.htm
- https://beginnersbook.com/2018/10/xml-tutorial-learn-xml/
- https://www.javatpoint.com/xml-tutorial
- https://www.geeksforgeeks.org/xml-basics/
- https://www.quackit.com/xml/tutorial/

UNIT - 3

Contents

Origin and Use of Perl- Scalars and their Operations – Assignment Statements and Simple Input and Output – Control Statements- Fundamentals of Arrays – Hashes References- Functions- Pattern Matching – File Input and Output – Simple programs in Perl -Using Perl for CGI Programming.

Perl Introduction

- Perl is a **programming language** which was originally developed for script manipulation. But now Perl is used for a variety of purpose including web development, GUI development, system administration and many more. It is a stable, cross platform programming language.
- For web development, **Perl CGI** is used. CGI is the gateway which interacts with the web browser and Perl in a system.
- Its typical use is extracting information from a text file and printing out report for converting a text file into another form. This is because it got its name after the expression, "**P**ractical **E**xtraction and **R**eport **L**anguage".
- Programs written in Perl are called **Perl scripts**, whereas system programs to execute Perl scripts are called **Perl program**.
- Perl is **an interpreted language**. When a Perl program run, it is first compiled into a byte code, then it is converted into machine instructions. So writing something in Perl instead of C saves your time.

Website to install PERL : “**http://padre.perlide.org/**”

Perl History

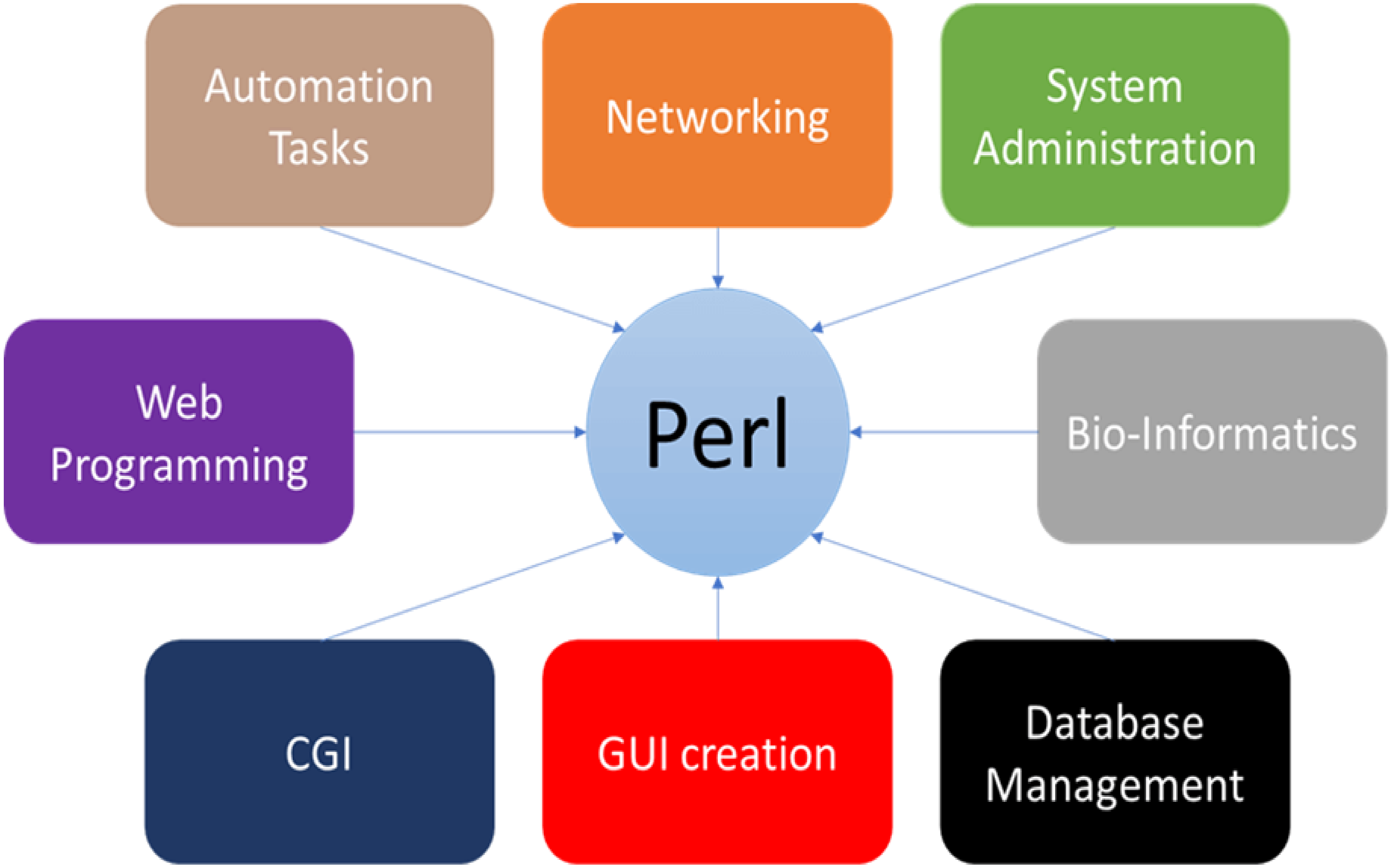
- Perl was developed by **Larry Wall in 1987** as a scripting language to make report processing easier.
- It was first released with version 1.0 on December 18, 1987.
- Perl 2, released in 1988 adding a much better regular expression engine.
- Perl 3, released in 1989 adding support for binary data streams.
- Perl 4, released in 1991 with a better documentation than earlier.
- Perl 5, released on October 17, 1994. It added many new features to its last version like objects, variables, references and modules.
- The latest version 5.24 is released on May 9, 2016.

Perl Features

- Perl supports both **procedural and object-oriented** programming.
- It is easily **extendible** as it supports 25,000 open source modules.
- It supports **Unicode**.
- It includes powerful tools to process text to make it compatible with mark-up languages like HTML, XML.
- Its **database integration** interface , DBI supports third-party databases including Oracle, Sybase, Postgres, MySQL and others.
- It is **embeddable in other systems** such as web servers and database servers.
- It is **open source** software licensed under GNU.
- Many frameworks are written in Perl.
- It can **handle encrypted web data** including e-commerce transactions.
- It is a **cross platform language.**

Where is PERL used

- The power of Perl can be implemented in many fields. The most popular use of Perl is in Web development.
- Perl is also used to automate many tasks in the Web servers and other administration jobs, it can automatically generate emails and clean up systems.
- Perl is still used for its original purpose i.e. extracting data and generating reports.
- Perl has become a popular language used in web development, networking and bioinformatics too.
- Apart from all this perl can also be used for CGI programming.
- Perl can also be utilized for image creation & manipulation.
- Perl is also known for implementation of OOP(object oriented programming) practices and supports all forms of inheritance (simple, multiple & diamond), polymorphism and encapsulation.
- Perl is flexible enough to support Procedural as well as OOP practices simultaneously.


Automation Tasks
Networking
System Administration
Web Programming
Perl
Bio-Informatics
CGI
GUI creation
Database Management

Use of Perl

- **Perl is very easy to learn**, especially if you have a background in computer programming. Perl was designed to be easy for humans to write and understand rather than making it easy for processing by computers. It uses regular expressions.

- **Perl is extremely portable.** It can run on any operating system that has Perl interpreter installed, so it is platform independent.

- **Small specific tasks in Perl become very easy and quick**.

- Perl is very good at **text processing, File handling and output reporting.**

- It is **free** to use.

Pros & Cons in PERL

PROS

- Compared to other Programming languages Perl is most powerful for text handling and Parsing.
- This is an interpreted language with fast execution time as there is no need to compile a Perl script.
- Simple and easy to program and understand.
- It is object oriented.
- Used in Web development for mostly Payment Gateways.
- Used in Automation and to test most of the Network and Storage related stuff.

CONS

- There is minimal GUI support as compared to other Programming languages.
- You need to refer to complex library modules which are not so easy to understand.
- Understanding complex Patterns requires experience.

Perl Example 1

Example:

```
!/usr/bin/perl
# Print a message
print "Hello World! with Perl\n";
```

- #!/usr/local/bin/perl, tells where to find the Perl compiler on your system.
- Perl statements end with semicolon (;)
- The (\n) is used to denote a new line.
- As it is a string, it will be enclosed in double quotes ("").
- And finally 'print' will display it on the screen.
- Perl is case-sensitive.

Saving File:

- Save the file with (.pl) extension.

Perl print() and say()

- The **say()** is not supported by the older perl versions.
- It acts like **print()** with only difference that it automatically adds a new line at the end without mentioning (\n).

Note: you need to mention the version in your script to use the say() function.

Comments in Perl

- Comments in any programming language are friends of developers.
- Comments can be used to make program user friendly and they are simply skipped by the interpreter without impacting the code functionality.
- For example:

```
# This is a single line comment
print "Hello, world\n“;
```

- Multi-line comments start with = and with the =cut statement.

```
=begin comment.This is another comment. And it spans
   multiple lines! =cut
```

Single and Double Quotes in Perl

```
#!/usr/bin/perl $a = 10;
print "Value of a = $a\n";
print 'Value of a = $a\n';
```

This will produce the following result –

```
Value of a = 10
Value of a = $a\n$
```

- Only double quotes **interpolate** variables and special characters such as newlines \n, whereas single quote does not interpolate any variable or special character.

"Here" Documents

- You can store or print multiline text with a great comfort. Even you can make use of variables inside the "here" document. Below is a simple syntax, check carefully there must be no space between the << and the identifier.
- An identifier may be either a bare word or some quoted text like we used EOF below. If identifier is quoted, the type of quote you use determines the treatment of the text inside the here document, just as in regular quoting. An unquoted identifier works like double quotes.

```
#!/usr/bin/perl

$a = 10;
$var = <<"EOF";
This is the syntax for here document and it will continue
until it encounters a EOF in the first line.
This is case of double quote so variable value will be
interpolated. For example value of a = $a
EOF
print "$var\n";

$var = <<'EOF';
This is case of single quote so variable value will be
interpolated. For example value of a = $a
EOF
print "$var\n";
```

```
This is the syntax for here document and it will continue
until it encounters a EOF in the first line.
This is case of double quote so variable value will be
interpolated. For example value of a = 10

This is case of single quote so variable value will be
interpolated. For example value of a = $a
```

Scalars and Their Operations

- Scalar variables start with $
- Scalar variables hold strings or numbers, and they are interchangeable
- When you first use (declare) a variable use the my keyword (not necessary) to indicate the variable's scope.
- Name must begin with a letter; any number of letters, digits, or underscore characters can follow
- Names are case sensitive
- Examples:
  - my $priority = 9;
  - my $priority = "A";

Scalars and Their Operations (continued)

- ***Numeric Operators***

  - Like those of C, Java, etc.

Operator	Associativity
++, --	nonassociative
unary -	right
**	right
*, /, %	left
binary +, -	left

Scalars and Their Operations (continued)

- *String Operators*

  - Concatenation - denoted by a period

    - e.g., If the value of $dessert is "apple", the value of
      $dessert . " pie" is "apple pie"

  - *Repetition - denoted by x*

    - e.g., If the value of $greeting is "hello ", the value of
      $greeting x 3 is "hello hello hello “

- String Functions

  - Functions and operators are closely related in Perl

    - e.g., if cube is a predefined function, it can be called with either cube(x) or cube x

Scalars and Their Operations (continued)

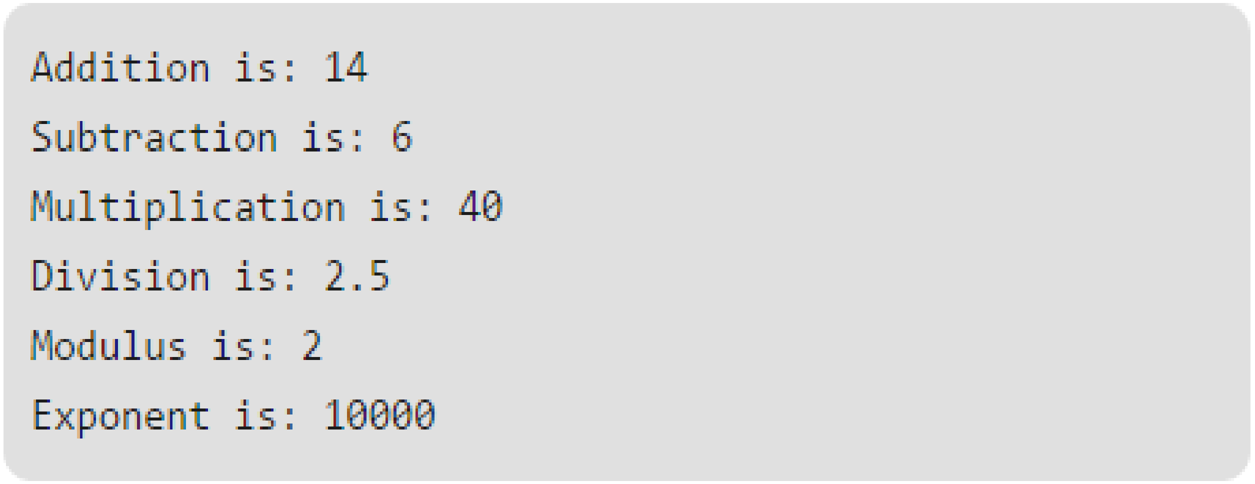
Name	Parameters	Result
`chomp`	a string	the string w/terminating newline characters removed
`length`	a string	the number of characters in the string
`lc`	a string	the string with uppercase letters converted to lower
`uc`	a string	the string with lowercase letters converted to upper
`hex`	a string	the decimal value of the hexadecimal number in the string
`join`	a character and a list of strings	the strings catenated together with the character inserted between them

Arithmetic in Perl

```
$a = 1 + 2;      # Add 1 and 2 and store in $a
$a = 3 - 4;      # Subtract 4 from 3 and store in $a
$a = 5 * 6;      # Multiply 5 and 6
$a = 7 / 8;      # Divide 7 by 8 to give 0.875
$a = 9 ** 10;    # Nine to the power of 10, that is, 9^10
$a = 5 % 2;      # Remainder of 5 divided by 2
++$a;            # Increment $a and then return it
$a++;            # Return $a and then increment it
--$a;            # Decrement $a and then return it
$a--;            # Return $a and then decrement it
```

```
# Perl Program to illustrate the Arithmetic Operators
# Operands
$a = 10;
$b = 4;
# using arithmetic operators
print "Addition is: ", $a + $b, "\n";
print "Subtraction is: ", $a - $b, "\n" ;
print "Multiplication is: ", $a * $b, "\n";
print "Division is: ", $a / $b, "\n";
print "Modulus is: ", $a % $b, "\n";
print "Exponent is: ", $a ** $b, "\n";
```

Output:



```
Addition is: 14
Subtraction is: 6
Multiplication is: 40
Division is: 2.5
Modulus is: 2
Exponent is: 10000
```

String and assignment operators

```
$a = $b . $c;   # Concatenate $b and $c
$a = $b x $c;   # $b repeated $c times

$a = $b;        # Assign $b to $a
$a += $b;       # Add $b to $a
$a -= $b;       # Subtract $b from $a
$a .= $b;       # Append $b onto $a
```

- You sometimes may need to group terms
  - Use parentheses ()
  - (5-6)*2 is not 5-(6*2)

Assignment Operators

- Assignment operators are used to assigning a value to a variable. The left side operand of the assignment operator is a variable and right side operand of the assignment operator is a value.

- Different types of assignment operators are shown below:
    - **“=”(Simple Assignment)** : This is the simplest assignment operator. This operator is used to assign the value on the right to the variable on the left. Example :$a = 10; $b = 20;

    - **“+=”(Add Assignment)** : This operator is combination of ‘+’ and ‘=’ operators. This operator first adds the current value of the variable on left to the value on the right and then assigns the result to the variable on the left.
      Example :($a += $b) can be written as ($a = $a + $b) If initially value stored in a is 5. Then ($a += 6) = 11.

Assignment Operators (continued)

- **“-=”(Subtract Assignment)** : This operator is combination of ‘-‘ and ‘=’ operators. This operator first subtracts the current value of the variable on left from the value on the right and then assigns the result to the variable on the left.
  Example :($a -= $b) can be written as ($a = $a - $b) If initially value stored in a is 8. Then ($a -= 6) = 2.

- **“*=”(Multiply Assignment)** : This operator is combination of ‘*’ and ‘=’ operators. This operator first multiplies the current value of the variable on left to the value on the right and then assigns the result to the variable on the left.
  Example :
  ($a *= $b) can be written as ($a = $a * $b) If initially value stored in a is 5. Then ($a *= 6) = 30.

Assignment Operators (continued)

- **“/=”(Division Assignment)** : This operator is combination of ‘/’ and ‘=’ operators. This operator first divides the current value of the variable on left by the value on the right and then assigns the result to the variable on the left.
  Example :($a /= $b) can be written as ($a = $a / $b) If initially value stored in a is 6. Then ($a /= 2) = 3.

- **“%=”(Modulus Assignment)** : This operator is combination of ‘%’ and ‘=’ operators. This operator first modulo the current value of the variable on left by the value on the right and then assigns the result to the variable on the left.
  Example :($a %= $b) can be written as ($a = $a % $b) If initially value stored in a is 6. Then ($a %= 2) = 0.

- **“**=”(Exponent Assignment)** : This operator is combination of ‘**’ and ‘=’ operators. This operator first exponent the current value of the variable on left by the value on the right and then assigns the result to the variable on the left.
  Example :($a **= $b) can be written as ($a = $a ** $b) If initially value stored in a is 6. Then ($a **= 2) = 36.

```perl
# Perl program to demonstrate the working
# of Assignment Operators
#!/usr/local/bin/perl

# taking two variables & using
# simple assignments operation
$a = 8;
$b = 5;

# using Assignment Operators
print "Addition Assignment Operator: ", $a += $b, "\n";

$a = 8;
$b = 4;
print "Subtractation Assignment Operator: ", $a -= $b, "\n" ;

$a = 8;
$b = 4;
print "Multiplication Assignment Operator: ", $a*=$b, "\n";

$a = 8;
$b = 4;
print "Division Assignment Operator: ",$a/=$b, "\n";

$a = 8;
$b = 5;
print "Modulo Assignment Operator: ", $a%=$b,"\n";

$a = 8;
$b = 4;
print "Exponent Assignment Operator: ", $a**=$b, "\n";
```

Output:

```
Addition Assignment Operator: 13
Subtractation Assignment Operator: 4
Multiplication Assignment Operator: 32
Division Assignment Operator: 2
Modulo Assignment Operator: 3
Exponent Assignment Operator: 4096
```

Control Statements

- Perl is an ***iterative language*** in which control flows from the first statement in the program to the last statement unless something interrupts. Some of the things that can interrupt this linear flow are conditional branches and loop structures.

statement block

- Statement blocks provide a mechanism for grouping statements that are to be executed as a result some expression being evaluated. They are used in all of the control structures discussed below. Statement blocks are designated by pairs of curly braces.

Syntax:

```
{
    stmt_1;
    stmt_2;
    stmt_3;
}
```

Control Statements (Continued)

if statement

Form: if (EXPR) BLOCK

Syntax:

```
if (expression)
{
   true_stmt_1;
   true_stmt_2;
   true_stmt_3;
}
```

if/else statement

Form: if (EXPR) BLOCK else BLOCK

Syntax:

```
if (expression)
{
true_stmt_1;
true_stmt_2;
true_stmt_3;
}
else
{
false_stmt_1;
false_stmt_2;
false_stmt_3;
}
```

if/elseif/else statement

Form:

if (EXPR) BLOCK elseif (EXPR) BLOCK . . .
else BLOCK

Syntax:

```
if (expression_A)
{
A_true_stmt_1;
A_true_stmt_2;
}
 elseif (expression_B)
{
B_true_stmt_1;
B_true_stmt_2;
}
else
{
false_stmt_1;
false_stmt_2;
}
```

Control Statements (Continued)

while statement **-** Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.

Form: LABEL: while (EXPR) BLOCK

The LABEL in this and the following control structures is optional. In addition to description, it also provides function in the quasi-goto statements: last, next, and redo. Perl conventional practice calls for labels to be expressed in uppercase to avoid confusion with variables or key words.

Syntax:

```
ALABEL: while (expression)
          {
               stmt_1;
               stmt_2;
               stmt_3;
           }
```

last operator - Terminates the loop statement and transfers execution to the statement immediately following the loop.

The last operator, as well as the next and redo operators that follow, apply only to loop control structures. They cause execution to jump from where they occur to some other position, defined with respect to the block structure of the encompassing control structure. Thus, they function as limited forms of *goto* statements.
Last causes control to jump from where it occurs to the first statement following the enclosing block.

Syntax:

```
ALABEL: while (expression)
        {
           stmt_1;
           stmt_2;
           last;
           stmt_3;
         } # last jumps to here
```

If last occurs within nested control structures, the jump can be made to the end of an outer loop by adding a label to that loop and specifying the label in the last statement.

Syntax:

```
    ALABEL: while (expression)
        {
        stmt_1;
        stmt_2;
        BLABEL: while (expression)
                {
                stmt_a;
                stmt_b;
                last ALABEL;
                stmt_c;
                }
        stmt_3;
        } # last jumps to here
```

***next operator* -** Causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.

The next operator is similar to last except that execution jumps to the end of the block, but remains *inside* the block, rather than exiting the block. Thus, iteration continues normally.

Syntax:

```
ALABEL: while (expression)
 {
     stmt_1;
     stmt_2;
     next;
     stmt_3;
     # next jumps to here
 }
```

As with last, next can be used with a label to jump to an outer designated loop.

redo operator **-** The redo command restarts the loop block without evaluating the conditional again. The continue block, if any, is not executed.

The redo operator is similar to next except that execution jumps to the top of the block without re-evaluating the control expression.

Syntax:

```
ALABEL: while (expression)
      {
            # redo jumps to here
            stmt_1;
            stmt_2;
            redo;
            stmt_3;
      }
```

As with last, next can be used with a label to jump to an outer designated loop.

Control Statements (Continued)

until statement **-** Repeats a statement or group of statements until a given condition becomes true. It tests the condition before executing the loop body.

Form: LABEL: until (EXPR) BLOCK

Syntax:

```
ALABEL: until (expression)
{  # while not
stmt_1;
stmt_2;
}
```

for statement **-** Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable

Form: LABEL: for (EXPR; EXPR; EXPR) BLOCK

Syntax:

```
ALABEL: for (initial exp; test exp; increment exp) # e.g., ($i=1; $i<5; $i++)
                {
                stmt_1;
                stmt_2;
                }
```

foreach statement - The foreach loop iterates over a normal list value and sets the variable VAR to be each element of the list in turn.

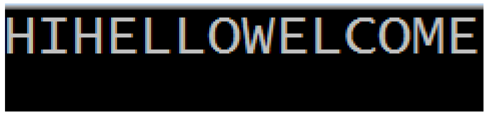
Form: LABEL: foreach VAR (EXPR) BLOCK

Syntax:

```
ALABEL: foreach $i (@aList)
{
stmt_1;
stmt_2;
stmt_3;
}
```

Example:

```
# Perl program to illustrate
# the foreach loop

# Array
@data = (‘HI’, ‘HELLO’, ‘WELCOME’);

# foreach loop
foreach $word (@data)
{
    print $word
}
```



```
HIHELLOWELCOME
```

Switch Statement

It is used to execute the code from multiple conditions. There is no case or switch statement in perl. Instead we use 'when' in place of case and 'given' in place of switch.

Syntax:

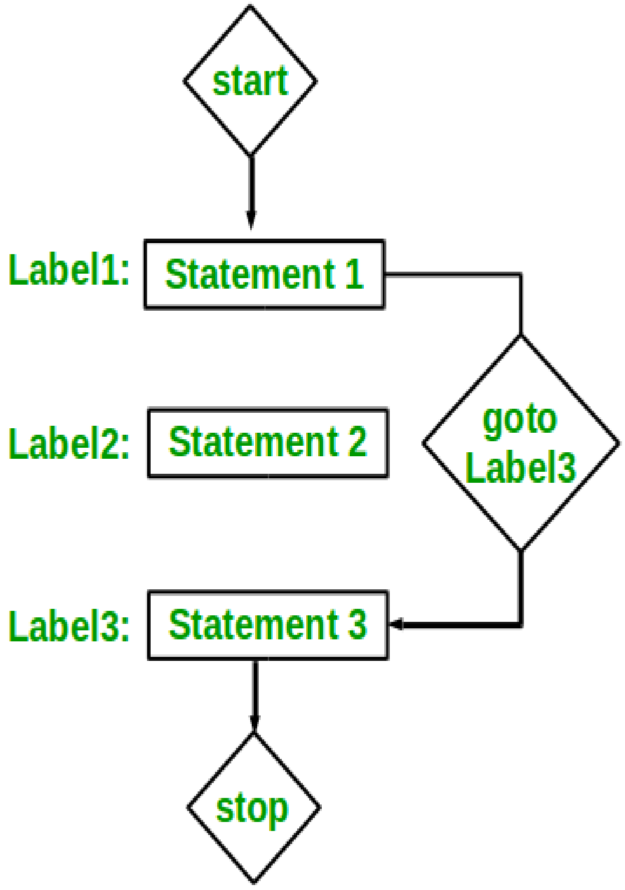
```
given (expression)
{
 when (condition)
        {
        statement(s);
        }
 when (condition)
        {
        statement(s);
        }
        . . .
 default
        {
        statement(s);
        }
}
```

goto statement

The goto statement in perl is a jump statement which is sometimes also referred to as unconditional jump statement. The `goto` statement can be used to jump from anywhere to anywhere within a function.

Syntax:

```
LABEL:
Statement 1;
Statement 2;
. . . . .
Statement n;
goto LABEL;
```



goto statement in Perl is of three forms- Label, Expression, and Subroutine.

1. **Label:** It will simply jump to the statement marked with the LABEL, and will continue the execution from that statement.
2. **Expression:** In this form, there will be an expression that will return a Label name after evaluation and goto will make it jump to the labeled statement.
3. **Subroutine:** goto will transfer the compiler to the subroutine of the given name from the currently running subroutine.

Example for goto statement:

```
# Perl program to print numbers
# from 1 to 10 using goto statement

# function to print numbers from 1 to
10
sub printNumbers()
{
    my $n = 1;
label:
    print "$n ";
    $n++;
    if ($n <= 10)
    {
        goto label;
    }
}

# Driver Code
printNumbers();
```

Output:

```
1 2 3 4 5 6 7 8 9 10
```

Example for goto statement using Expression:

An expression can also be used to give a call to a specific label and pass the execution control to that label. This expression when passed to the **goto** statement, evaluates to generate a label name, and further execution is continued from that statement defined by that label name.

Example:

```
# Defining two strings which contain label name in parts
$a = "lab";
$b = "el";
# function to print numbers from 1 to 10
sub printNumbers()
{
    my $n = 1;
label:
    print "$n ";
    $n++;
    if ($n <= 10)
    {
        # Passing Expression to label name
        goto $a.$b;
    }
}
# Driver Code
printNumbers();
```

Example for goto statement using subroutine:

A subroutine can also be called with the use of the **goto** statement. This subroutine is called from within another subroutine or individually based on its use. It holds the work that is to be performed next to the calling statement. This method can be used to call a function recursively to print a series or a range of characters.

Example:

```
# Perl program to print numbers from 1 to 10 using goto statement
 # function to print numbers from 1 to 10
sub label
{
    print "$n ";
    $n++;

    if($n <= 10)
    {
        goto &label;
    }
}

# Driver Code
my $n = 1;
label();
```

Continue Statement

A **continue** BLOCK, is always executed just before the conditional is about to be evaluated again. A continue statement can be used with *while* and *foreach* loops. A continue statement can also be used alone along with a BLOCK of code in which case it will be assumed as a flow control statement rather than a function.

Syntax

```
continue
{
statement(s);
}
```

```
foreach $a (@listA)
{
statement(s);
}
continue
{
statement(s);
}
```

```
while(condition)
{
statement(s);
}
continue
{
statement(s);
}
```

Perl Array

A Perl array variable stores an ordered list of scalar values.
Array variables are preceeded by an "at" (@) sign.
To refer a single element of Perl array, variable name will be preceded with dollar ($) sign followed by index of element in the square bracket.

Syntax: @arrayName = (element1, element2, element3..);

Perl Simple Array Example

```
#!/usr/bin/perl
 @num = (2015, 2016, 2017);
@string = ("One", "Two", "Three");
print "$num[0]\n";
print "$num[1]\n";
print "$num[2]\n";
print "$string[0]\n";
print "$string[1]\n";
print "$string[2]\n";
```

Output:

```
2015
2016
2017
One
Two
Three
```

In the above example, we have defined two arrays, one with number element and other with string element. Both arrays are printed with their index elements.

Perl Simple Array Example-2

```perl
#!/usr/bin/perl
@ages = (25, 30, 40);
@names = ("John Paul", "Lisa", "Kumar");
print "\$ages[0] = $ages[0]\n";
print "\$ages[1] = $ages[1]\n";
print "\$ages[2] = $ages[2]\n";
print "\$names[0] = $names[0]\n";
print "\$names[1] = $names[1]\n";
print "\$names[2] = $names[2]\n";
```

Here we have used the escape sign (\) before the $ sign just to print it. Other Perl will understand it as a variable and will print its value.

Output:

```
$ages[0] = 25
$ages[1] = 30
$ages[2] = 40
$names[0] = John Paul
$names[1] = Lisa
$names[2] = Kumar
```

Array Creation

Array variables are prefixed with the @ sign and are populated using either parentheses or the qw operator. For Example,

```
@array = (1, 2, 'Hello');
@array = qw/This is an array/;
```

The second line uses the qw// operator, which returns a list of strings, separating the delimited string by white space. In this example, this leads to a four-element array; the first element is 'this' and last (fourth) is 'array'. This means that you can use different lines as follows –

```
@days = qw/Monday
Tuesday
...
Sunday/;
```

You can also populate an array by assigning each value individually as follows –

```
$array[0] = 'Monday';
...
$array[6] = 'Sunday';
```

Accessing Array Elements

When accessing individual elements from an array, you must prefix the variable with a dollar sign ($) and then append the element index within the square brackets after the name of the variable. For example −

```perl
#!/usr/bin/perl
@days = qw/Mon Tue Wed Thu Fri Sat Sun/;
print "$days[0]\n";
print "$days[1]\n";
print "$days[2]\n";
print "$days[6]\n";
print "$days[-1]\n";
print "$days[-7]\n";
```

This will produce the following result −

```
Mon
Tue
Wed
Sun
Sun
Mon
```

Array indices start from zero, so to access the first element you need to give 0 as indices. You can also give a negative index, in which case you select the element from the end, rather than the beginning, of the array. This means the following −

```perl
print $days[-1]; # outputs Sun
print $days[-7]; # outputs Mon
```

Sequential Number Arrays

Perl offers a shortcut for sequential numbers and letters. Rather than typing out each element when counting to 100 for example, we can do something like as follows –

```
#!/usr/bin/perl
@var_10 = (1..10);
@var_20 = (10..20);
@var_abc = (a..z);
print "@var_10\n"; # Prints number from 1 to 10
print "@var_20\n"; # Prints number from 10 to 20
print "@var_abc\n"; # Prints number from a to z
```

Here double dot (..) is called **range operator**. This will produce the following result –

```
1 2 3 4 5 6 7 8 9 10
10 11 12 13 14 15 16 17 18 19 20
a b c d e f g h i j k l m n o p q r s t u v w x y z
```

Perl Array Size or Length

The size of an array is determined with scalar context on the array. The returned value will be always one greater than the largest index. In short the size of an array will be ($#array + 1). Here, $#array is the maximum index of the array.

```
@array = (you, me, us);
$array[5] = 4;
$size = @array;
$index_max = $#array;
print "Size:  $size\n";
print "Maximum Index: $index_max\n";
```

Output:

```
Size: 6
Maximum Index: 5
```

In the output, there are only three elements containing information, but the give array has total 5 elements.

Perl Array Functions

You can add or remove an element from an array using some array functions. We'll discuss following array Perl functions:

- Push
- Pop
- Shift
- Unshift

1) Push on Array

The push array function appends a new element at the end of the array.

Example:

```
@array = ("pink", "red", "blue");
push @array, "orange";
print "@array\n";
```

Output:

```
pink red blue orange
```

In the above program, "orange" element is added at the end of the array.

2) Pop on Array

The pop array function removes the last element from the array.

Example:

```
@array = ("pink", "red", "blue");
pop @array;
print "@array\n";
```

Output:

```
pink red
```

In the above program, "blue" element is removed from the end of the array.

3) Shift on Array

The shift array function removes the left most element of array and thus shorten the array by 1.

Example:

```
@array = ("pink", "red", "blue");
shift @array;
print "@array\n";
```

Output:

```
red blue
```

In the above program, "pink" is removed from the array.

4) Unshift on Array

The unshift array function adds a new element at the start of the array.

Example:

```
@array = ("pink", "red", "blue");
unshift @array, "orange";
print "@array\n";
```

Output:

```
orange pink red blue
```

In the above program, "orange" is added at the start of the array.

Slicing Array Elements

You can also extract a "slice" from an array - that is, you can select more than one item from an array in order to produce another array.

```
#!/usr/bin/perl
@days = qw/Mon Tue Wed Thu Fri Sat Sun/;
@weekdays = @days[3,4,5];
print "@weekdays\n";
```

This will produce the following result −

```
Thu Fri Sat
```

The specification for a slice must have a list of valid indices, either positive or negative, each separated by a comma.

For speed, you can also use the **..** range operator –

```
#!/usr/bin/perl
@days = qw/Mon Tue Wed Thu Fri Sat Sun/;
@weekdays = @days[3..5];
print "@weekdays\n";
```

This will produce the following result –

```
Thu Fri Sat
```

Replacing Array Elements

Now we are going to introduce one more function called **splice()**, which has the following syntax –

```
splice @ARRAY, OFFSET [ , LENGTH [ , LIST ] ]
```

This function will remove the elements of @ARRAY designated by OFFSET and LENGTH, and replaces them with LIST, if specified. Finally, it returns the elements removed from the array.

Following is the example –

```
#!/usr/bin/perl
@nums = (1..20);
print "Before - @nums\n";
splice(@nums, 5, 5, 21..25);
print "After - @nums\n";
```

This will produce the following result –

```
Before - 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
After - 1 2 3 4 5 21 22 23 24 25 11 12 13 14 15 16 17 18 19 20
```

Here, the actual replacement begins with the 6th number after that five elements are then replaced from 6 to 10 with the numbers 21, 22, 23, 24 and 25.

Perl Strings to Arrays, split()

With the help of split() function, we can split a string into array of strings and returns it.

Syntax :

```
split [ PATTERN [ , EXPR [ , LIMIT ] ] ]
```

```perl
#!/usr/bin/perl
# define Strings
$var_string = "Rain-Drops-On-Roses-And-Whiskers-On-Kittens";
$var_names = "Larry,David,Roger,Ken,Michael,Tom";

# transform above strings into arrays.
@string = split('-', $var_string);
@names = split(',', $var_names);
print "$string[3]\n"; # This will print Roses
print "$names[4]\n"; # This will print Michael
```

This will produce the following result –

```
Roses
Michael
```

Perl Arrays to Strings, join()

The join() function is used to combine arrays to make a string. It combines the separate arrays into one string and returns it.

```perl
#!/usr/bin/perl
# define Strings
$var_string = "Rain-Drops-On-Roses-And-Whiskers-On-Kittens";
$var_names = "Larry,David,Roger,Ken,Michael,Tom";
# transform above strings into arrays.
@string = split('-', $var_string);
@names = split(',', $var_names);
$string1 = join( '-', @string );
$string2 = join( ',', @names );
print "$string1\n";
print "$string2\n";
```

This will produce the following result –

```
Rain-Drops-On-Roses-And-Whiskers-On-Kittens
Larry,David,Roger,Ken,Michael,Tom
```

Perl Merging Two Arrays, merged()

Two arrays can be merged together using merged() function as a single string removing all the commas in between them.

Example:

```
#!/usr/bin/perl
@odd = (1,3,5);
@even = (2, 4, 6);
@numbers = (@odd, @even);
print "numbers = @numbers\n";
```

Output:

```
numbers = 1 3 5 2 4 6
```

In the above program, array1 and array2 are merged into one single string and then printed.

OR

```
#!/usr/bin/perl
@numbers = (1,3,(4,5,6));
print "numbers = @numbers\n";
```

```
numbers = 1 3 4 5 6
```

Perl Sorting Arrays, sort()

To sort an array, sort() array function is used. The sort() function sorts all the elements of an array according to the ASCII standard.

Example:

```
# defining array
@days = ("sun", "mon", "tue", "wed", "thu", "fri", "sat");
print "Original array: @days\n";
# sorting array
@days = sort(@days);
print "Sorted array: @days\n";
```

Output:

```
Original array: sun mon tue wed thu fri sat
Sorted array: fri mon sat sun thu tue wed
```

In the above program, we have printed both original and sorted array. This array is sorted in the alphabetical order.

Selecting Elements from Lists

The list notation is identical to that for arrays. You can extract an element from an array by appending square brackets to the list and giving one or more indices –

```
#!/usr/bin/perl
$var = (5,4,3,2,1)[4];
print "value of var = $var\n"
```

This will produce the following result –

```
value of var = 1
```

Similarly, we can extract slices, although without the requirement for a leading @ character –

```
#!/usr/bin/perl
@list = (5,4,3,2,1)[1..3];
print "Value of list = @list\n";
```

This will produce the following result –

```
Value of list = 4 3 2
```

Perl Array with Loops

Perl array elements can be accessed within a loop. Diferent types of loops can be used.

We will show array accessing with following loops:

- foreach loop
- for loop
- while loop
- until loop

Perl Array with foreach Loop

In foreach loop, the control variable is set over the elements of an array. Here, we have specified $i as the control variable and print it.

Example:

```
@num = qw(10 20 30 40 50);
foreach $i (@num) {
  print "$i\n";
}
```

Output:

```
10
20
30
40
50
```

Perl Array with for Loop

A control variable will be passed in for loop as the index of the given array.

Example:

```
@num = qw(10 20 30 40 50);
for($i = 0; $i < 5; $i++){
  print "@num[$i]\n";
}
```

Output:

```
10
20
30
40
50
```

Perl Array with until Loop

The until loop works like while loop, but they are opposite of each other. A while loop runs as long as a condition is true whereas an until loop runs as long as condition is false. Once the condition is false until loop terminates. The until loop can be written on the right hand side of the equation as an expression modifier.

Example:

```
@your_name = "John";
print "@your_name\n" until $i++ > 4;
```

Output:

```
John
John
John
John
John
```

In the above program, once $i is greater than 4 according to the condition, loop iteration stops.

Perl Multidimensional Array

Perl multidimensional arrays are arrays with more than one dimension. The multi dimensional array is represented in the form of rows and columns, also called Matrix. They can not hold arrays or hashes, they can only hold scalar values. They can contain references to another arrays or hashes.

Perl Multidimensional Array Matrix Example

Here, we are printing a 3 dimensional matrix by combining three different arrays **arr1**, **arr2** and **arr3**. These three arrays are merged to make a matrix array **final**.

Two for loops are used with two control variables **$i** and **$j**.

```
## Declaring arrays
my @arr1 = qw(0 10 0);
my @arr2 = qw(0 0 20);
my@arr3 = qw(30 0 0);
## Merging all the single dimensional arrays
my @final = (\@arr1, \@arr2, \@arr3);
print "Print Using Array Index\n";
for(my $i = 0; $i <= $#final; $i++){
  # $#final gives highest index from the array
  for(my $j = 0; $j <= $#final ; $j++){
     print "$final[$i][$j] ";
  }
 print "\n";
}
```

Output:

```
Print Using Array Index
0 10 0
0 0 20
30 0 0
```

Perl Multidimensional Array Initialization and Declaration Example

In this example we are initializing and declaring a three dimensional Perl array .

```
@array = (
     [1, 2, 3],
     [4, 5, 6],
     [7, 8, 9]
    );
    for($i = 0; $i < 3; $i++)
     {
      for($j = 0; $j < 3; $j++)
       {
          print "$array[$i][$j] ";
       }
      print "\n";
     }
```

Output:

```
1 2 3
4 5 6
7 8 9
```

Perl Hashes

- The hashes is the most essential and influential part of the perl language.
- A hash is a group of key-value pairs. The keys are unique strings and values are scalar values.
- Hashes are declared using my keyword. The variable name starts with a (%) sign.
- Hashes are like arrays but there are two differences between them.
  - First arrays are ordered but hashes are unordered.
  - Second, hash elements are accessed using its value while array elements are accessed using its index value.
- No repeating keys are allowed in hashes which makes the key values unique inside a hash. Every key has its single value.

Syntax for Perl hashes:

```
my %hashName = (
    "key" => "value";
    )
```

Perl Hash Accessing

To access single element of hash, ($) sign is used before the variable name. And then key element is written inside {} braces.

Example

```
my %capitals = (
    "India"  => "New Delhi",
    "South Korea" => "Seoul",
    "USA"  => "Washington, D.C.",
    "Australia"  => "Canberra"
);
print"$capitals{'India'}\n";
print"$capitals{'South Korea'}\n";
print"$capitals{'USA'}\n";
print"$capitals{'Australia'}\n";
```

Output:

```
New Delhi
Seoul
Washington,D.C.
Canberra
```

Perl Hash Indexing

Hashes are indexed using $key and $value variables. All the hash values will be printed using a while loop. As the while loop runs, values of each of these variables will be printed.

Example:

```
my %capitals = (
   "India"  => "New Delhi",
   "South Korea" => "Seoul",
   "USA"  => "Washington, D.C.",
   "Australia"  => "Canberra"
);
# LOOP THROUGH IT
while (($key, $value) = each(%capitals)){
    print $key.", ".$value."\n";
}
```

Output:

```
Australia, Canberra

India, New Delhi

USA, Washington, D.C.

South Korea, Seoul
```

Perl sorting Hash by key

You can sort a hash using either its key element or value element. Perl provides a sort() function for this. In this example, we'll sort the hash by its key elements

Example:

```
my %capitals = (
   "India"  => "New Delhi",
   "South Korea" => "Seoul",
   "USA"  => "Washington, D.C.",
   "Australia"  => "Canberra"
);
# Foreach loop
foreach $key (sort keys %capitals) {
    print "$key: $capitals{$key}\n";
}
```

Output:

```
Australia: Canberra
India: New Delhi
South Korea: Seoul
USA: Washington: D.C.
```

Look at the output, all the key elements are sorted alphabetically.

Perl sorting Hash by its value

Here we'll sort hash by its value elements.

Example:

```
my %capitals = (
   "India"  => "New Delhi",
   "South Korea" => "Seoul",
   "USA"  => "Washington, D.C.",
   "UK"  => "London"
);
# Foreach loop
foreach $value (sort {$capitals{$a} cmp $c
apitals{$b} }
         keys %capitals)
{
    print "$value $capitals{$value}\n";
}
```

Output:

```
UK London
India New Delhi
South Korea Seoul
USA Washington D.C.
```

Look at the output, all the value elements are sorted alphabetically.

Perl Hash key Existence

Accessing a key-value pair from hash which doesn't exist will return error or warnings. To prevent from this, you can check whether a key exist or not in a hash with **exists()** function. It returns true if the key exists.

Example:

```
my %capitals = (
    "India"  => "New Delhi",
    "South Korea" => "Seoul",
    "USA"  => "Washington, D.C.",
    "Australia"  => "Canberra"
);
if (exists($capitals{'India'}))
{
  print "found the key\n";
}
```

Output:

```
found the key
```

The above output shows that the 'India' key exists in the 'capitals' hash.

Perl Hash Slices
If you want only some values from a hash, you can extract them and display as a list of values.
For this, you have to store them in an array variable with @ prefix as they will return a list of values and then print them.

Example:

```
my %capitals = (
   "India"  => "New Delhi",
   "South Korea" => "Seoul",
   "USA"  => "Washington, D.C.",
   "Australia"  => "Canberra"
);
@array = @capitals{India, USA, Australia};

print "@array\n";
```

Output:

```
New Delhi
Washington, D.C.
Canberra
```

Perl Hash creating Empty hash

An empty hash will always have size 0.

In this example, first we have created a hash with size 3. Then we have created an empty hash with size 0.

Example:

```
my %first = ('john'=>9853147320, 'jose'=>7823654028, 'janie',=>'8850279610');
print 'hash size: ', scalar keys %first;
print "\n";
#creating emptyempty hash
my %emptyempty=();
print 'hash size: ', scalar keys %emptyempty;
```

Output:

```
hash size: 3
hash size: 0
```

Perl Adding Hash Elements

New key-value pair can be added in a hash by declaring them as single element in the hash variable. Here, we are adding two key-value pair, [Germany - Berlin] and [UK - London].

Example:

```
my %capitals = (
   "India"  => "New Delhi",
   "South Korea" => "Seoul",
   "USA"  => "Washington, D.C.",
   "Australia"  => "Canberra"
);
while (($key, $value) = each(%capitals)){
    print $key.", ".$value."\n";
}
#adding new element
$capitals{Germany} = Berlin;
$capitals{UK} = London;
# Printing new hash
print "\n";
while (($key, $value) = each(%capitals)){
    print $key.", ".$value."\n";
}
```

Output:

```
UK, London
Australia, Canberra
Germany, Berlin
India, New Delhi
USA, Washington D.C.
South Korea, Seoul
```

Perl Removing Hash Elements

To remove a hash element, use **delete()** function. Here, we have removed both the key-value pairs which were added in the last example.

Example:

```
my %capitals = (  "India"  => "New Delhi",
   "South Korea" => "Seoul",
   "USA"  => "Washington, D.C.",
   "Australia"  => "Canberra"
   "Germany "  => " Berlin"
   " UK "  => "London"  );

while (($key, $value) = each(%apitals)){
    print $key.", ".$value."\n";
}
#removing element
delete($capitals{Germany});
delete($capitals{UK});
# Printing new hash
print "\n";
while (($key, $value) = each(%capitals)){
    print $key.", ".$value."\n";
}
```

Output:

```
Australia, Canberra
India, New Delhi
USA, Washington D.C.
South Korea, Seoul
```

Perl deleting Vs Undefining Hash Elements

deleting: In deleting, key-value pair will be deleted from the hash.
Syntax:
delete($hash{$key});

undef: In undef, the value will be undefined but key will remain in the hash.
Syntax:
Undef $hash{$key};

In the following example, we have created a hash 'rank'.
One by one we'll undefine and remove all the key values from the hash.
On undefining a key only its value will be shown, on deleting a key it will be completely deleted from the hash along with its value.

Perl Functions and Subroutines

Perl functions and subroutines are used to reuse a code in a program. You can use a function at several places in your application with different parameters.

There is only one difference in function and subroutine, subroutine is created with **sub** keyword and it returns a value. You can divide your code into separate subroutines. Logically each function in each division should perform a specific task.

Syntax of subroutine:

```
sub subName
{
body
}
```

Perl define and call subroutine function

The syntax for Perl define a subroutine function is given below:

```
sub subName
{
body
}
        OR
subName(list of arguments);
&subName(list of arguments);
```

In the following example, we are defining a subroutine function 'myOffice' and call it.

```
#defining function
sub myOffice
{
   print "javaTpoint!\n";
}
# calling Function
myOffice();
```

Output:

```
javaTpoint!
```

Perl subroutine Function with Arguments

You can pass any number of arguments inside a subroutine. Parameters are passed as a list in the special @_ list array variables. Hence, the first argument to the function will be $_[0], second will be $_[1] and so on. In this example, we are calculating perimeter of a square by passing a single parameter.

```
$squarePerimeter = perimeter(25);
print("This is the perimter of a square with dimension 25: $square
          Perimeter\n");
sub perimeter {
$dimension = $_[0];
return(4 * $dimension);
 }
```

Output:

```
100
```

Perl subroutine with List

Here the @_ variable is an array, hence it is used to supply list to a subroutine. We have declared an array 'a' with list and call it.

```
sub myList{
   my @list = @_;
   print "Here is the list @list\n";
}
@a = ("Orange", "Pineapple", "Mango", "Grapes", "Guava");
# calling function with list
myList(@a);
```

Output:

```
Here is the list Orange Pineapple Mango Grapes Guava
```

Perl subroutine with Hashes

When a hash is passed to a subroutine, then hash is automatically translated into its key-value pair.

```
sub myHash{
   my (%hash) = @_;
   foreach my $key ( keys %hash ){
      my $value = $hash{$key};
      print "$key : $value\n";
   }
}
%hash = ('Carla' => 'Mother', 'Ray' => 'Father', 'Ana' => 'Daught
                      er', 'Jose' => 'Son');
# Function call with hash parameter
myHash(%hash);
```

Output:

```
Ray : Father
Jose : Son
Carla : Mother
Ana : Daughter
```

Perl subroutine Local and Global Variables

By default, all the variables are global variables inside Perl. But you can create local or private variables inside a function with 'my' keyword. The 'my' keyword restricts the variable to a particular region of code in which it can be used and accessed. Outside this region, this variable can not be used.

In the following example, we have shown both local and global variable. First, $str is called locally (AAABBBCCCDDD) and then it is called globally (AEIOU).

```
$str = "AEIOU";
sub abc{
   # Defining local variable
   my $str;
   $str = "AAABBBCCCDDD";
   print "Inside the function local variable is called $str\n";
}
# Function call
abc();
print "Outside the function global variable is called $str\n";
```

Output:

```
Inside the function local variable is called AAABBBCCCDDD
Outside the function global variable is called AEIOU
```

Perl File Handling

File handling is the most important part in any programming language. A filehandle is an internal Perl structure that associates with a file name.
Perl File handling is important as it is helpful in accessing file such as text files, log files or configuration files.
Perl filehandles are capable of creating, reading, opening and closing a file.

Perl Create File
We are creating a file, **file1.txt** with the help of open() function.
The $fh (file handle) is a scalar variable and we can define it inside or before the open() function. Here we have define it inside the function. The '>' sign means we are opening this file for writing. The **$filename** denotes the path or file location.
Once file is open, use $fh in print statement. The print() function will print the above text in the file.
Now we are closing $fh. Well, closing the file is not required in perl. Your file will be automatically closed when variable goes out of scope.

EXAMPLE:

```
my $filename = 'file1.txt';
open(my $fh, '>', $filename) or die "Could
not open file '$filename' $!";
print $fh "Hello!! We have created this file
 as an example\n";
close $fh;
print "done\n";
```

Output:

```
done.
```

A file file1.txt will be created in our system.

Perl Open File

We can open a file in following ways:

(<) Syntax

The < sign is used to open an already existing file. It opens the file in read mode.

```
          open FILE, "<", "fileName.txt" or die $!
```

(>) Syntax

The > sign is used to open and create the file if it doesn't exists. It opens the file in write mode.

```
          open FILE, ">", "fileName.txt" or die $!
```

The “>" sign will empty the file before opening it. It will clear all your data of that file. To prevent this use (+) sign before ">" or "<" characters.

(+>+<) Syntax

```
          open FILE, "+<", "fileName.txt" or die $!
          open FILE, "+>", "fileName.txt" or die $!
```

(>>) Syntax

The >> sign is used to read and append the file content. It places the file pointer at the end of the file where you can append the information. Here also, to read from this file, you need to put (+) sign before ">>" sign.

```
          open FILE, "<", "fileName.txt" or die $!
```

Perl Read File

You can read a complete file at once or you can read it one line at a time. We'll show an example for both. Opening a file to read is similar to open a file to write. With only one difference that ">" is used to write and "<" is used to read the file.

We have created a file **file1.txt** with the following content:

```
This is the First Line.
This is the Second Line.
This is the Third Line.
This is the Fourth Line.
```

To read Single line at a time

First line of file1.txt will be displayed. Content of $row will be printed with "done" to make it clear that we reached at the end of our program.

Example:

```
my $filename = 'file1.txt';
open(my $fh, '<:encoding(UTF-8)', $filename)
  or die "Could not open file '$filename' $!";
my $row = <$fh>;
print "$row\n";
print "done\n";
```

Output:

```
This is the First Line.
Done.
```

To read Multi lines at a time

Now we know to read single line from a file. To read multiple lines put $row in a while loop.
Every time, when while loop will reach its condition, it will execute **my $row = <$fh>**. It will read the next line from the file. At the last line, $fh will return undef which is false and loop will terminate.

Example:

```
my $filename = 'file1.txt';
open(my $fh, '<:encoding(UTF-8)', $filename)
  or die "Could not open file '$filename' $!";
while (my $row = <$fh>) {
  chomp $row;
  print "$row\n";
}
print "done\n";
```

Output:

```
This is the First Line.
This is the Second Line.
This is the Third Line.
This is the Fourth Line.
Done.
```

Perl Write File

Through file writing, we'll append lines in the file1.txt. As already stated, new lines will be added at the last of the file.

Example:

```
open (FILE, ">> file1.txt") || die "problem opening $file1.txt\n";
 print FILE "This line is added in the file1.txt\n";
# FILE array of lines is written here
print FILE @lines1;
# Another FILE array of lines is written here
print FILE "A complete new file is created";
# write a second array of lines to the file
print FILE @lines2;
```

Output:

```
This line is added in the file1.txt
A complete new file is created
```

Perl Close File

Perl close file is used to close a file handle using close() function. File closing is not compulsory in perl. Perl automatically closes file once the variable is out of scope.

```
open FILE1, "file1.txt" or die $!;
...
close FILE1;
```

Perl File Handle Operator

File handle operator is the main method to read information from a file. It is used to get input from user. In scalar context, it returns a single line from the filehandle and in line context, it returns a list of lines from the filehandle.

```
print "What is your age?\n";
$age = <STDIN>;
if($age >= 18)
{
  print "You are eligible to vote.\n";
} else {
  print "You are not eligible to vote.\n";
   }
```

Perl Copying a File

We can copy content of one file into another file as it is. First open file1 then open file2. Copy the content of file 1 to file2 by reading its line through a while loop.

```
# Opening file1 to read
open(File1Data, "<file1.txt");
# Opening new file to copy content of file1
open(File2Data, ">file2.txt");
# Copying data from file1 to file2.
while(<File1Data>)
{
   print File2Data $_;
}
close( File1Data );
close( File2Data );
```

Output:

```
done
A new file file2.pl will be created in the location
where file1.pl exists.
```

Perl Regular Expressions

- A regular expression is a string of characters that defines the pattern or patterns you are viewing.
- The basic method for applying a regular expression is to use the pattern binding operators =~ and !~. The first operator is a test and assignment operator.
- There are three regular expression operators within Perl.
  - Match Regular Expression - m//
  - Substitute Regular Expression - s///
  - Transliterate Regular Expression - tr///
- The forward slashes in each case act as delimiters for the regular expression (regex) that you are specifying. If you are comfortable with any other delimiter, then you can use in place of forward slash.

Perl matching operator :

- ‘**m**' operator in Perl is used to match a pattern within the given text. The string passed to m operator can be enclosed within any character which will be used as a delimiter to regular expressions.
- To print this matched pattern and the remaining string, m operator provides various operators which include $, which contains whatever the last grouping match matched.

$& – contains the entire matched string

$` – contains everything before the matched string

$’ – contains everything after the matched string

```perl
#!/usr/bin/perl
$bar = "This is foo and again foo";
    if ($bar =~ /foo/)
   {
      print "First time is matching\n";
   }
   else
   {
      print "First time is not matching\n";
    }
$bar = "foo";
    if ($bar =~ /foo/)
   {
      print "Second time is matching\n";
    }
   else
   {
      print "Second time is not matching\n";
    }
```

When this program is executed, it produces the following result −

```
First time is matching
Second time is matching
```

```perl
#!/usr/bin/perl -w

# Text String
$string = "Geeks for geeks is the best";

# Let us use m operator to search
# "or g"
$string =~ m/or g/;

# Printing the String
print "Before: $`\n";
print "Matched: $&\n";
print "After: $'\n";
```

Output:

```
Before: Geeks f
Matched: or g
After: eeks is the best
```

Perl substitution operator :

- The substitution operator, s///, is really just an extension of the match operator that allows you to replace the text matched with some new text.
- The basic form of the operator is –

```
s/PATTERN/REPLACEMENT/;
```

- The PATTERN is the regular expression for the text that we are looking for.
- The REPLACEMENT is a specification for the text or regular expression that we want to use to replace the found text with.

For example, we can replace all occurrences of **dog** with **cat** using the following regular expression –

```
#/user/bin/perl
$string = "The cat sat on the mat";
$string =~ s/cat/dog/;
print "$string\n";
```

When above program is executed, it produces the following result –

```
The dog sat on the mat
```

Perl translation operator :

- Translation is similar, but not identical, to the principles of substitution, but unlike substitution, translation (or transliteration) does not use regular expressions for its search on replacement values. The translation operators are –

  ```
  tr/SEARCHLIST/REPLACEMENTLIST/cds
  y/SEARCHLIST/REPLACEMENTLIST/cds
  ```

- The translation replaces all occurrences of the characters in SEARCHLIST with the corresponding characters in REPLACEMENTLIST.

For example, using the "The cat sat on the mat." string we have been using in this chapter –

```
#/user/bin/perl
$string = 'The cat sat on the mat';
$string =~ tr/a/o/;
print "$string\n";
```

When above program is executed, it produces the following result –

```
The cot sot on the mot.
```

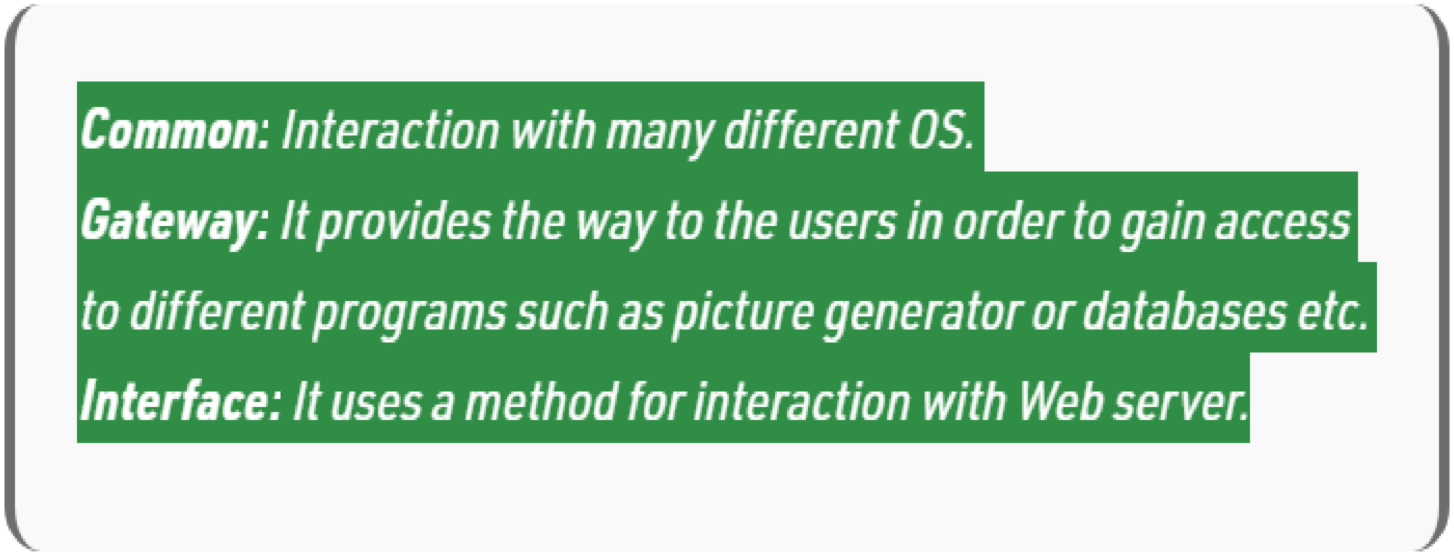
Perl CGI

In Perl, CGI(Common Gateway Interface) is a protocol for executing scripts via web requests. It is a set of rules and standards that define how the information is exchanged between the web server and custom scripts. Earlier, scripting languages like Perl were used for writing the CGI applications. And, CGI code called by HTTP server was referred to as the CGI script. Later, the growth of the web caused the increase in need of dynamic content through which CGI applications which were written in other languages instead of Perl became more popular and in demand and were referred as scripts only. The specifics of how the script is executed by the server are determined by the server. CGI applications can perform nearly any task. For example, you can access databases, hold telnet sessions, create Web pages on-the-fly and generate graphics, etc. CGI is having a very simple concept, but creating a CGI application requires real programming skills.

What is CGI?

Common Gateway Interface (CGI) is a protocol which defines the interaction of web servers with some executable programs in order to produce dynamic web pages. Basically, it shows how the web server sends information to the program and the program sends the information back to the web server which in turn can be sent back to the browser. Between web servers and external programs, it is considered as the standard programming interface.

CGI stands for:

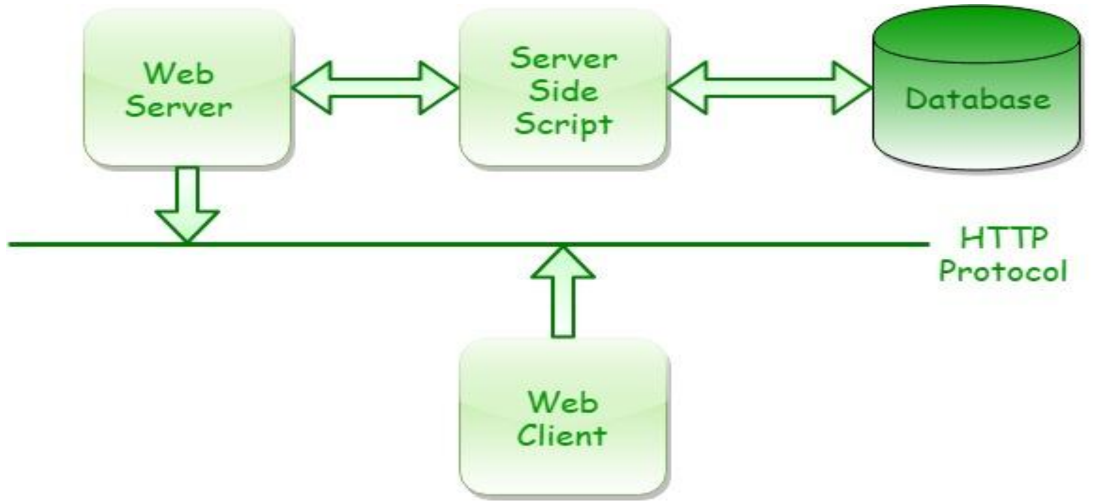


Common: *Interaction with many different OS.*

Gateway: *It provides the way to the users in order to gain access to different programs such as picture generator or databases etc.*

Interface: *It uses a method for interaction with Web server.*

CGI programs can send many types of data or media, like documents, images, audio clips, etc. Most Web sites use CGI with fields for input and deals of dynamic content on the Web is done mainly using CGI. It is a method through which a web server can get/send the data from/to databases, documents, and other programs respectively, and then present that data to viewers through the web.



In the above figure, with the help of HTTP (Hyper Text Transfer Protocol) web browser which is running on client machine exchanges information with the web server. Since, CGI program and web server normally run on the same system on which the web server resides, depending on the request from the browser, web server either provides the document from its own document directory or executes a CGI program.

CGI Programming in Perl

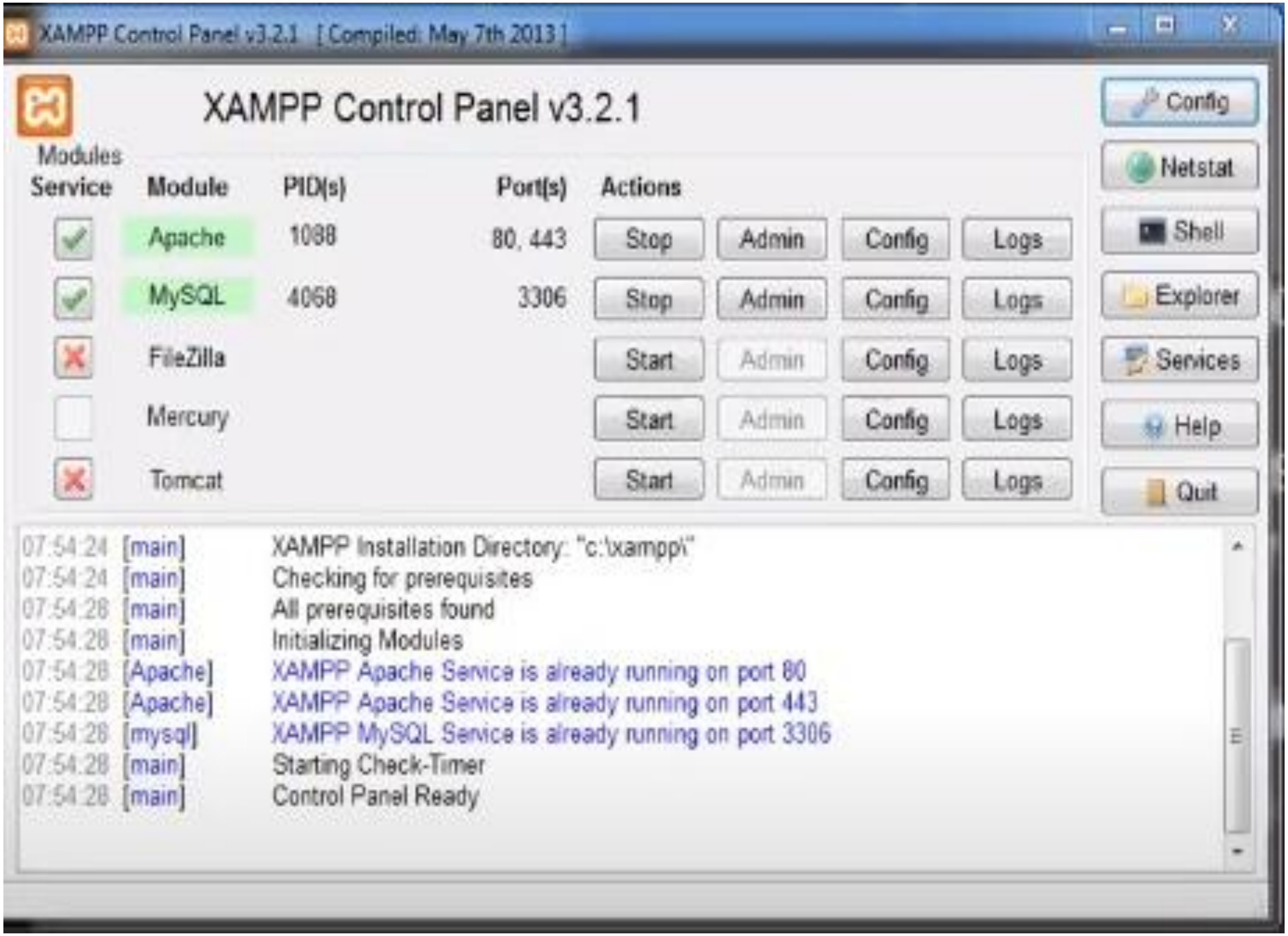
1. Download “XAMPP” server.
2. After installation look for a folder naming “XAMPP” in C:
3. Now create your “.cgi” file in editor and give the path to perl (#!"c:\xampp\perl\bin\perl.exe“ in shebang line of your program)
4. Save your file with “.cgi” extension in C:\xampp\cgi-bin.
5. Create your html file in editor and save the file with “.html” extension in C:\xampp\htdocs.
6. Now open your web browser . Type local host.
7. If apache is running on system then it shows “It’s working!”.
8. If it’s not then open XAMPP server. This will start all services associated with xampp.
9. If your apache is in running state then again open browser and type localhost.

CGI File

```
#!"c:\xampp\perl\bin\perl.exe"

use strict;
use CGI ':standard';

my $name = param('name');
my $gender = param('gender');
my $profession = param('profession');
my @sports = param('sport');
 my $list;
 if (@sports)
{
    $list = join ', ', @sports;
}
else
{
    $list = 'Null';
}
 print header,
start_html(-title=>$name), h1("Hello, $name"), h3 p('You have Submitted the following Data:'), h4
table(Tr(td('Name:'), h4 td($name)), h4 Tr(td('Gender:'),
h4 td($gender)), h4 Tr(td('Profession:'), h4 td($profession)), h4 Tr(td('Sports:'),
h4 td($list))),
end_html;
```

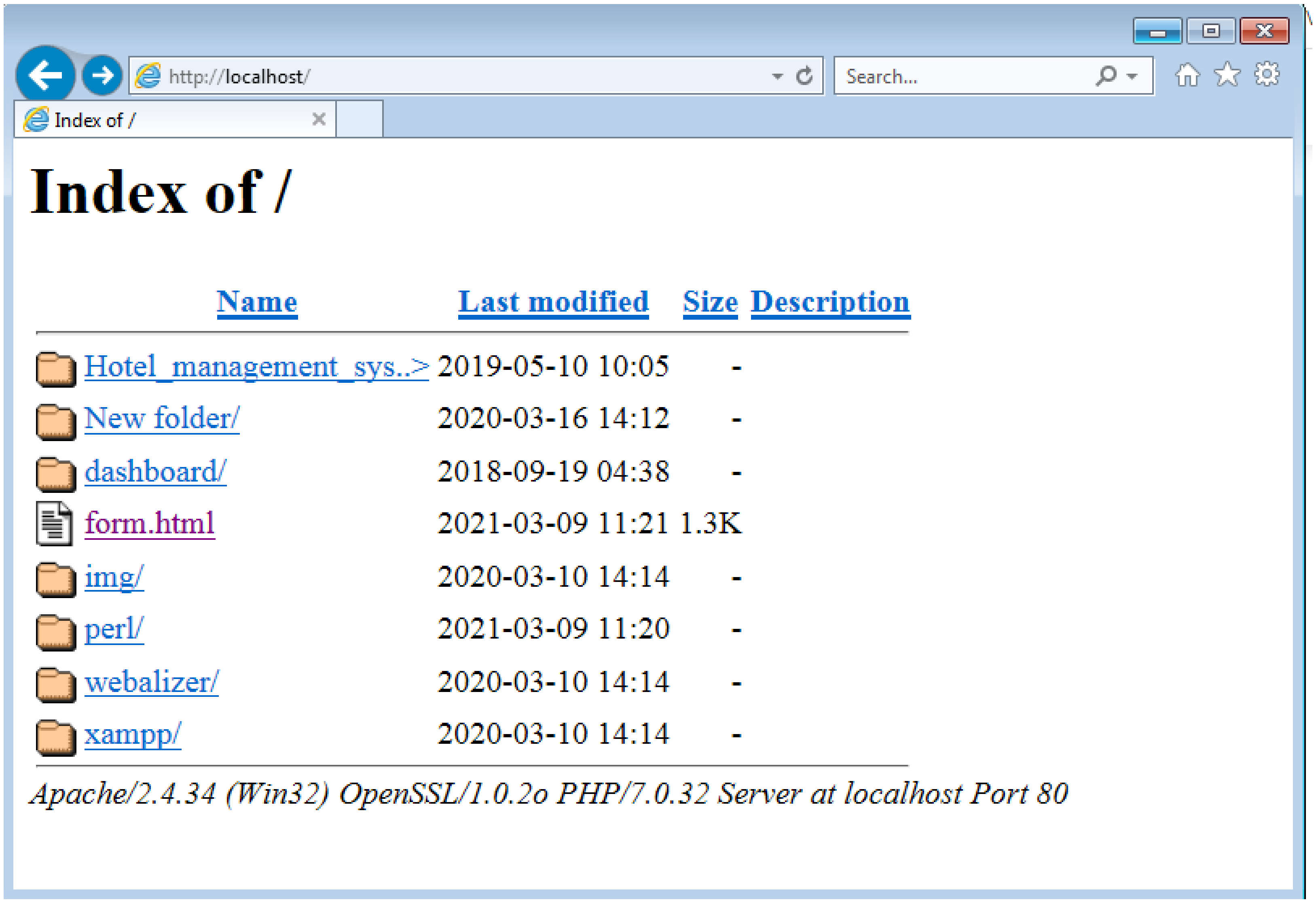
HTML FILE

```
<html>  <head>     <title>GfG Test Example Form</title>  </head>
<body>
   <h1>CGI-Example Form</h1>      <h3><p>Information Required.</p></h3>
   <form action="/cgi-bin/form.cgi" method="Post">
   <table>
      <tr>         <td>Name:</td>         <td><input type="text" name="name"><td>         </tr>
      <tr>
      <td>Gender:</td>
      <td><select name="gender" size="1">
         <option>Female</option>
         <option>Male</option>
         <option>Transgender</option>
      </select></td>
      </tr>
      <tr>  <td>Profession:</td>   <td><input type="text" name="profession"><td>   </tr>
      <tr>
      <td>Sports:</td>
         <td><input type="checkbox" name="sport“ value="Cricket">Cricket
            <input type="checkbox" name="sport“ value="Hockey">Hockey
            <input type="checkbox" name="sport“ value="TableTennis">TableTennis
            <input type="checkbox" name="sport“ value="Football">Football</td>
      </tr>
      <tr>
      <td colspan="2"><input type="submit"></td>
      </tr>
   </table>
   </form> </body>  </html>
```


XAMPP Control Panel v3.2.1 [Compiled: May 7th 2013]
XAMPP Control Panel v3.2.1
Config
Netstat
Shell
Explorer
Services
Help
Quit

Modules Service	Module	PID(s)	Port(s)	Actions			
✓	Apache	1088	80, 443	Stop	Admin	Config	Logs
✓	MySQL	4068	3306	Stop	Admin	Config	Logs
✗	FileZilla			Start	Admin	Config	Logs
	Mercury			Start	Admin	Config	Logs
✗	Tomcat			Start	Admin	Config	Logs

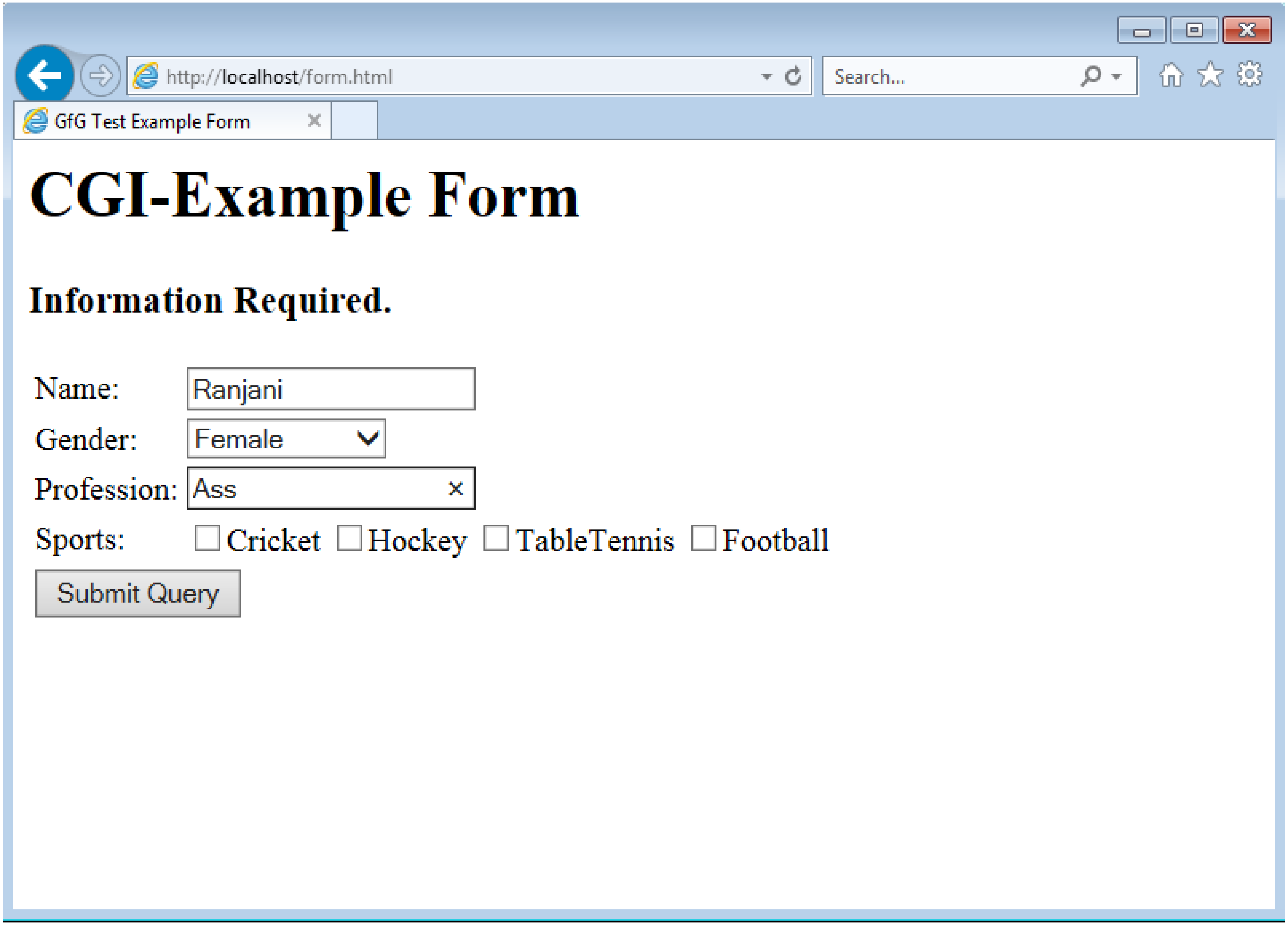
```
07:54:24  [main]     XAMPP Installation Directory: "c:\xampp\"
07:54:24  [main]     Checking for prerequisites
07:54:28  [main]     All prerequisites found
07:54:28  [main]     Initializing Modules
07:54:28  [Apache]   XAMPP Apache Service is already running on port 80
07:54:28  [Apache]   XAMPP Apache Service is already running on port 443
07:54:28  [mysql]    XAMPP MySQL Service is already running on port 3306
07:54:28  [main]     Starting Check-Timer
07:54:28  [main]     Control Panel Ready
```



Index of /

```
      Name                    Last modified    Size  Description
Hotel_management_sys..>  2019-05-10 10:05    -
New folder/              2020-03-16 14:12    -
dashboard/               2018-09-19 04:38    -
form.html                2021-03-09 11:21  1.3K
img/                     2020-03-10 14:14    -
perl/                    2021-03-09 11:20    -
webalizer/               2020-03-10 14:14    -
xampp/                   2020-03-10 14:14    -
```

Apache/2.4.34 (Win32) OpenSSL/1.0.2o PHP/7.0.32 Server at localhost Port 80

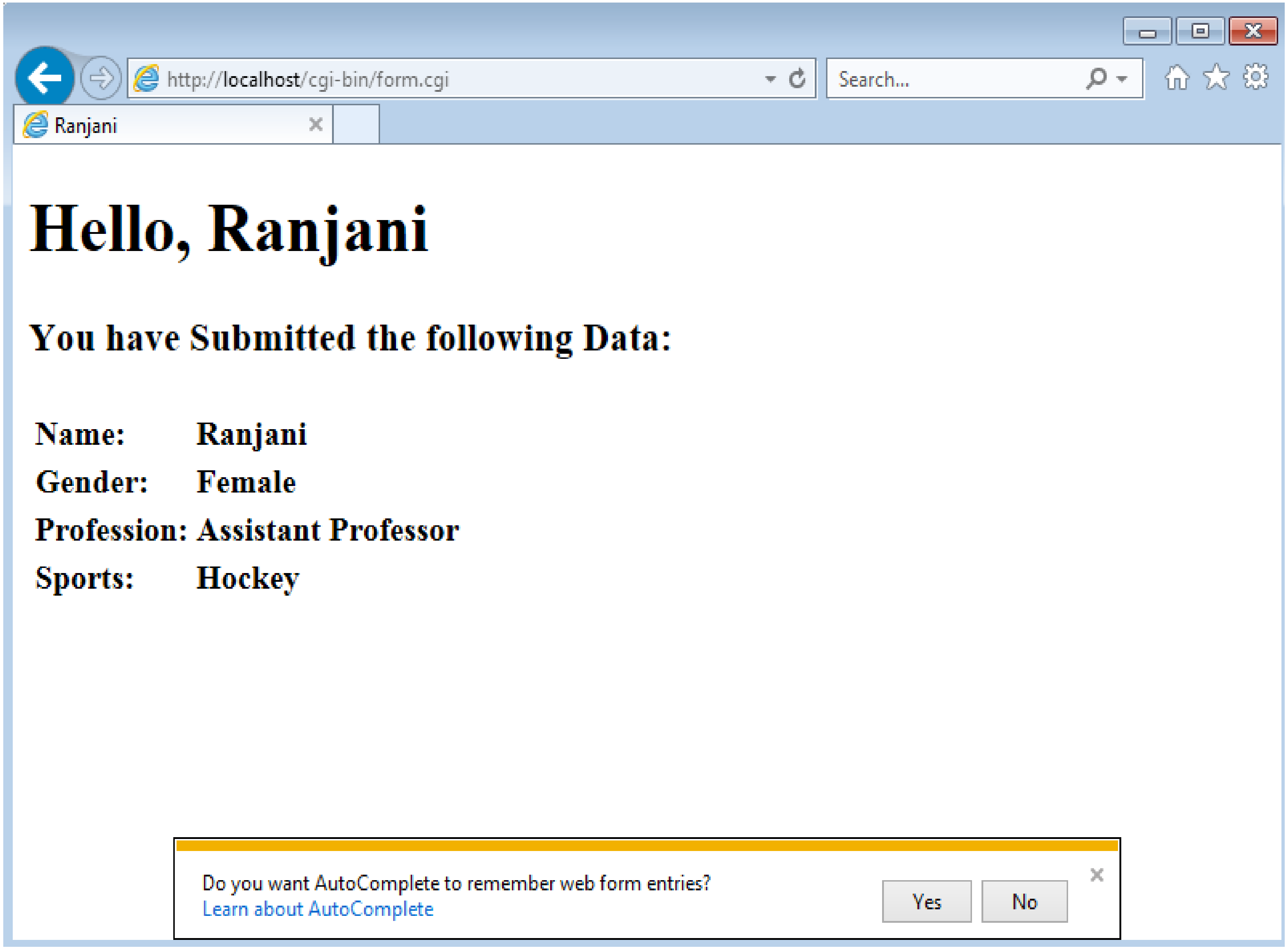


CGI-Example Form

Information Required.

Name:	Ranjani
Gender:	Female
Profession:	Ass
Sports:	☐ Cricket ☐ Hockey ☐ TableTennis ☐ Football

Submit Query



Hello, Ranjani

You have Submitted the following Data:

Name:	Ranjani
Gender:	Female
Profession:	Assistant Professor
Sports:	Hockey

Do you want AutoComplete to remember web form entries?
Learn about AutoComplete

References

- https://www.tutorialspoint.com/perl/
- https://www.javatpoint.com/perl-tutorial
- https://www.perltutorial.org/
- http://www.tizag.com/perlT/
- https://www.geeksforgeeks.org/introduction-to-perl/
- https://www.codesdope.com/perl-file-io/
- http://sandbox.mc.edu/~bennet/perl/leccode/

UNIT 4
PHP & MySQL

CONTENTS

Origin and Use of PHP- Overview of PHP- General Syntactic Characteristics Operations and Expressions- Control Statements- Arrays- Functions-Pattern Matching- Form Handling- Files-Cookies-Session Tracking - Database Connectivity, Simple programs in PHP and MySQL.

Origin of PHP

- The **Hypertext Preprocessor (PHP)** is a programming language that allows web developers to create dynamic content that interacts with databases.
- PHP is basically used for developing web based software applications.
- PHP is an open-source, interpreted, and object-oriented scripting language that can be executed at the server-side.
- PHP was created by **Rasmus Lerdorf in 1994** but appeared in the market in 1995.
- **PHP 7.4.0** is the latest version of PHP, which was released on **28 November**.

What is a PHP File?

- PHP files can contain text, HTML, CSS, JavaScript, and PHP code
- PHP code is executed on the server, and the result is returned to the browser as plain HTML
- PHP files have extension ".php"

Uses of PHP

- It handles dynamic content, database as well as session tracking for the website.
- You can create sessions in PHP.
- It can access cookies variable and also set cookies.
- It helps to encrypt the data and apply validation.
- PHP supports several protocols such as HTTP, POP3, SNMP, LDAP, IMAP, and many more.
- Using PHP language, you can control the user to access some pages of your website.
- PHP can handle the forms, such as - collect the data from users using forms, save it into the database, and return useful information to the user. **For example** - Registration form.

Overview of PHP

- PHP stands for Hypertext Preprocessor.
- PHP is an interpreted language, i.e., there is no need for compilation.
- PHP is faster than other scripting languages, for example, ASP and JSP.
- PHP is a server-side scripting language, which is used to manage the dynamic content of the website.
- PHP can be embedded into HTML.
- PHP is an object-oriented language.
- PHP is an open-source scripting language.
- PHP is simple and easy to learn language.

Characteristics of PHP

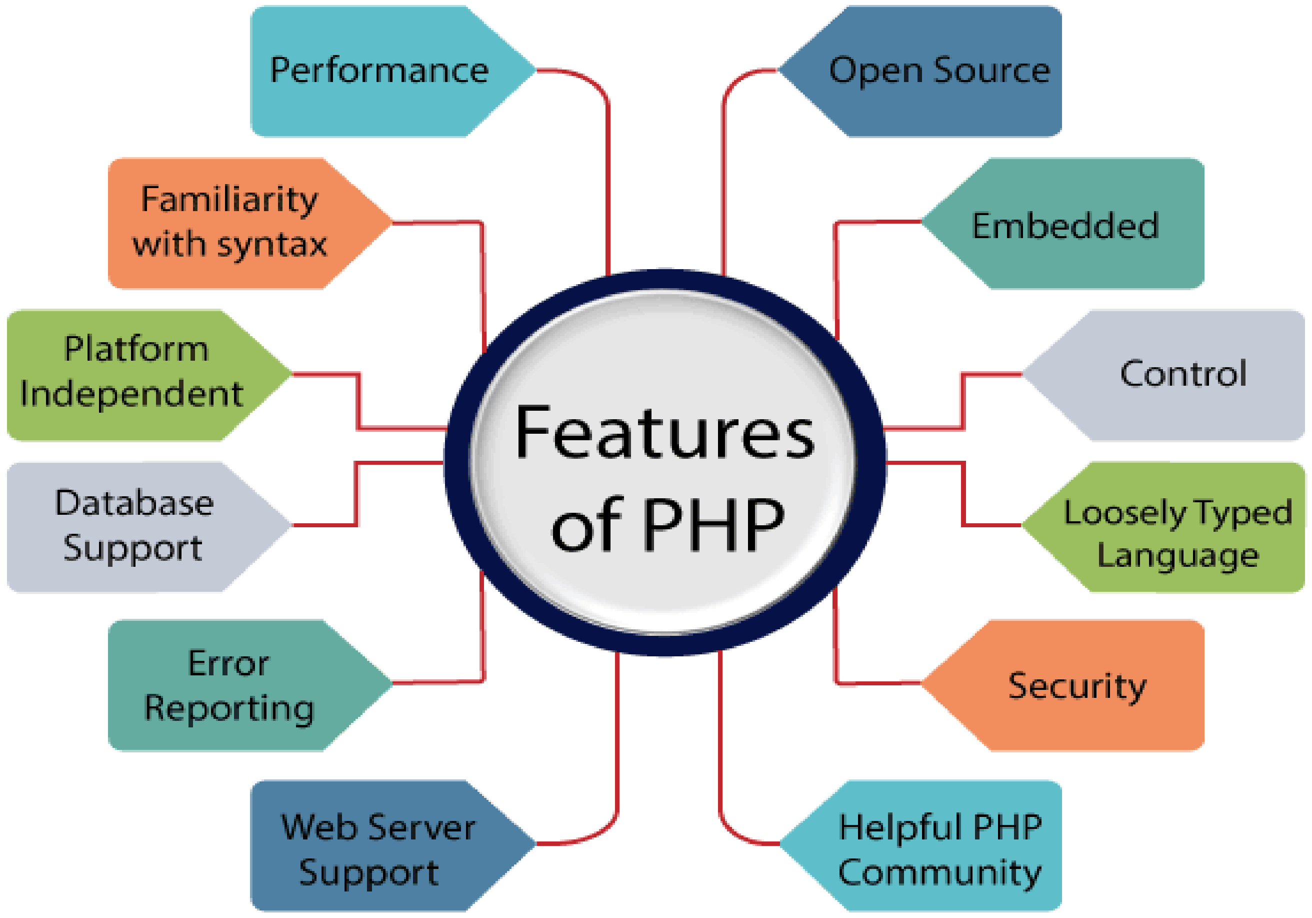
- Simplicity
- Efficiency
- Security
- Flexibility
- Familiarity

Applications of PHP

- PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.
- PHP can handle forms, i.e. gather data from files, save data to a file, through email you can send data, return data to the user.
- You add, delete, modify elements within your database through PHP.
- Access cookies variables and set cookies.
- Using PHP, you can restrict users to access some pages of your website.
- It can encrypt data.

PHP Features

- PHP is very popular language because of its simplicity and open source. There are some important features of PHP given below:



PHP Features (Cont..)

Performance:
PHP script is executed much faster than those scripts which are written in other languages such as JSP and ASP. PHP uses its own memory, so the server workload and loading time is automatically reduced, which results in faster processing speed and better performance.
Open Source:
PHP source code and software are freely available on the web. You can develop all the versions of PHP according to your requirement without paying any cost. All its components are free to download and use.
Familiarity with syntax:
PHP has easily understandable syntax. Programmers are comfortable coding with it.
Embedded:
PHP code can be easily embedded within HTML tags and script.
Platform Independent:
PHP is available for WINDOWS, MAC, LINUX & UNIX operating system. A PHP application developed in one OS can be easily executed in other OS also.
Database Support:
PHP supports all the leading databases such as MySQL, SQLite, ODBC, etc.

PHP Features (Cont..)

Error Reporting -
PHP has predefined error reporting constants to generate an error notice or warning at runtime. E.g., E_ERROR, E_WARNING, E_STRICT, E_PARSE.
Loosely Typed Language:
PHP allows us to use a variable without declaring its datatype. It will be taken automatically at the time of execution based on the type of data it contains on its value.
Web servers Support:
PHP is compatible with almost all local servers used today like Apache, Netscape, Microsoft IIS, etc.
Security:
PHP is a secure language to develop the website. It consists of multiple layers of security to prevent threads and malicious attacks.
Control:
Different programming languages require long script or code, whereas PHP can do the same work in a few lines of code. It has maximum control over the websites like you can make changes easily whenever you want.

PHP Installation

- To install PHP, we will suggest you to install AMP (Apache, MySQL, PHP) software stack. It is available for all operating systems. There are many AMP options available in the market that are given below:
  - **WAMP** for Windows
  - **LAMP** for Linux
  - **MAMP** for Mac
  - **SAMP** for Solaris
  - **FAMP** for FreeBSD
- **XAMPP** (Cross, Apache, MySQL, PHP, Perl) for Cross Platform: It includes some other components too such as FileZilla, OpenSSL, Webalizer, Mercury Mail, etc.
- If you are on Windows and don't want Perl and other features of XAMPP, you should go for WAMP. In a similar way, you may use LAMP for Linux and MAMP for Macintosh.

Basic PHP Syntax

- A PHP script can be placed anywhere in the document.
- A PHP script starts with **<?php** and ends with **?>**

```
<?php
// PHP code goes here
?>
```

- The default file extension for PHP files is ".php".
- A PHP file normally contains HTML tags, and some PHP scripting code

Example

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

<h1>My first PHP page</h1>

<?php
echo "Hello World!";
?>

</body>
</html>
```

Note: PHP statements end with a semicolon (`;`).

PHP Case Sensitivity

- In PHP, NO keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are case-sensitive.
- Example:

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

<?php
ECHO "Hello World!<br>";
echo "Hello World!<br>";
EcHo "Hello World!<br>";
?>

</body>
</html>
```

```
Hello World!
Hello World!
Hello World!
```

Note: However; all variable names are case-sensitive!

- Look at the example below; only the first statement will display the value of the $color variable! This is because$color, $COLOR, and $coLOR are treated as three different variables:
- Example:

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

<?php
$color = "red";
echo "My car is " . $color . "<br>";
echo "My house is " . $COLOR . "<br>";
echo "My boat is " . $coLOR . "<br>";
?>

</body>
</html>
```

```
My car is red
My house is
My boat is
```

PHP Comments

- A comment in PHP code is a line that is not executed as a part of the program. Its only purpose is to be read by someone who is looking at the code.

SINGLE LINE COMMENT

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

<?php
// This is a single-line comment
# This is also a single-line comment
?>

</body>
</html>
```

MULTI LINE COMMENT

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

<?php
/*
This is a multiple-lines comment
block that spans over multiple
lines
*/
?>

</body>
</html>
```

PHP Variables

- Variables are "containers" for storing information.

Creating (Declaring) PHP Variables

- In PHP, a variable starts with the $ sign, followed by the name of the variable:

```
<?php
$txt = "Hello world!";
$x = 5;
$y = 10.5;
?>
```

- After the execution of the statements above, the variable $txt will hold the value Hello world!, the variable $x will hold the value 5, and the variable $y will hold the value 10.5.

Note: When you assign a text value to a variable, put quotes around the value.

PHP Variables (Cont..)

- A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).
- Rules for PHP variables:
- A variable starts with the $ sign, followed by the name of the variable
- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive ($age and $AGE are two different variables)

Examples

```php
<?php
$txt = "W3Schools.com";
echo "I love $txt!";
?>
```

```
I love W3Schools.com!
```

or

```php
<?php
$txt = "W3Schools.com";
echo "I love " . $txt . "!";
?>
```

```
I love W3Schools.com!
```

```php
<?php
$x = 5;
$y = 4;
echo $x + $y;
?>
```

```
9
```

PHP Variables Scope

- In PHP, variables can be declared anywhere in the script.
- The scope of a variable is the part of the script where the variable can be referenced/used.
- PHP has <u>three</u> different variable scopes:
  - ➢**Local**
  - ➢**Global**
  - ➢**Static**
- **Local Scope**
  - – A variable declared **within** a function has a LOCAL SCOPE and can only be accessed within that function.
- **Global Scope**
  - – A variable declared **outside** a function has a GLOBAL SCOPE and can only be accessed outside a function

Examples

Variable with local scope:

```php
<?php
function myTest() {
    $x = 5; // local scope
    echo "<p>Variable x inside function is: $x</p>";
}
myTest();

// using x outside the function will generate an error
echo "<p>Variable x outside function is: $x</p>";
?>
```

```
Variable x inside function is: 5

Variable x outside function is:
```

Examples

Variable with global scope:

```
<?php
$x = 5; // global scope

function myTest() {
   // using x inside this function will generate an error
   echo "<p>Variable x inside function is: $x</p>";
}
myTest();

echo "<p>Variable x outside function is: $x</p>";
?>
```

```
Variable x inside function is:

Variable x outside function is: 5
```

PHP The global Keyword

The global keyword is used to access a global variable from within a function.

Example

```
<?php
$x = 5;
$y = 10;

function myTest() {
    global $x, $y;
    $y = $x + $y;
}

myTest();
echo $y; // outputs 15
?>
```

PHP also stores all global variables in an array called $GLOBALS[*index*]. The *index* holds the name of the variable. This array is also accessible from within functions and can be used to update global variables directly.

Example

```
<?php
$x = 5;
$y = 10;

function myTest() {
    $GLOBALS['y'] = $GLOBALS['x'] + $GLOBALS['y'];
}

myTest();
echo $y; // outputs 15
?>
```

PHP The static Keyword

Normally, when a function is completed/executed, all of its variables are deleted. However, sometimes we want a local variable NOT to be deleted. We need it for a further job. To do this, use the static keyword when you first declare the variable:

Example

```
<?php
function myTest() {
   static $x = 0;
   echo $x;
   $x++;
}

myTest();
myTest();
myTest();
?>
```

```
0
1
2
```

Then, each time the function is called, that variable will still have the information it contained from the last time the function was called.

Note: The variable is still local to the function.

PHP echo and print Statements

- With PHP, echo and print are more or less the same. They are both used to output data to the screen.
- The differences are small:
  - echo has no return value while print has a return value of 1 so it can be used in expressions.
  - echo can take multiple parameters (although such usage is rare) while print can take one argument.
  - echo is marginally faster than print.

PHP Data Types

- Variables can store data of different types, and different data types can do different things.
- PHP supports the following data types:
  - String
  - Integer
  - Float (floating point numbers - also called double)
  - Boolean
  - Array
  - Object
  - NULL
  - Resource

PHP String

A string is a sequence of characters, like "Hello world!".
A string can be any text inside quotes.
You can use single or double quotes:

Example

```
<?php
$x = "Hello world!";
$y = 'Hello world!';

echo $x;
echo "<br>";
echo $y;
?>
```

```
Hello world!
Hello world!
```

PHP Integer

An integer data type is a non-decimal number between -2,147,483,648 and 2,147,483,647.

Rules for integers:

- ✓An integer must have at least one digit
- ✓An integer must not have a decimal point
- ✓An integer can be either positive or negative
- ✓Integers can be specified in: decimal (base 10), hexadecimal (base 16), octal (base 8), or binary (base 2) notation

In the following example $x is an integer. The PHP var_dump() function returns the data type and value:

Example

```
<?php
$x = 5985;
var_dump($x);
?>
```

```
int(5985)
```

PHP Float

A float (floating point number) is a number with a decimal point or a number in exponential form.
In the following example $x is a float. The PHP var_dump() function returns the data type and value:

Example

```
<?php
$x = 10.365;
var_dump($x);
?>
```

```
float(10.365)
```

PHP Array

An array stores multiple values in one single variable.
In the following example $cars is an array. The PHP var_dump() function returns the data type and value:

Example

```
<?php
$cars = array("Volvo","BMW","Toyota");
var_dump($cars);
?>
```

```
array(3) { [0]=> string(5) "Volvo" [1]=> string(3) "BMW" [2]=> string(6) "Toyota" }
```

PHP Object

An object is a data type which stores data and information on how to process that data. In PHP, an object must be explicitly declared.
First we must declare a class of object. For this, we use the class keyword. A class is a structure that can contain properties and methods:

Example

```
<?php
class Car {
  function Car() {
    $this->model = "VW";
  }
}

// create an object
$herbie = new Car();

// show object properties
echo $herbie->model;
?>
```

```
VW
```

PHP NULL Value

Null is a special data type which can have only one value: NULL.
A variable of data type NULL is a variable that has no value assigned to it.

Tip: If a variable is created without a value, it is automatically assigned a value of NULL. Variables can also be emptied by setting the value to NULL:

Example

```
<?php
$x = "Hello world!";
$x = null;
var_dump($x);
?>
```

```
NULL
```

PHP String Functions

strlen()

The PHP strlen() function returns the length of a string.

Example

Return the length of the string "Hello world!":

```
<?php
echo strlen("Hello world!"); // outputs 12
?>
```

str_word_count()

The PHP str_word_count() function counts the number of words in a string.

Example

Count the number of word in the string "Hello world!":

```
<?php
echo str_word_count("Hello world!"); // outputs 2
?>
```

strrev()

The PHP strrev() function reverses a string.

Example

Reverse the string "Hello world!":

```
<?php
echo strrev("Hello world!"); // outputs !dlrow olleH
?>
```

strpos()

The PHP strpos() function searches for a specific text within a string. If a match is found, the function returns the character position of the first match. If no match is found, it will return FALSE.

Example

Search for the text "world" in the string "Hello world!":

```
<?php
echo strpos("Hello world!", "world"); // outputs 6
?>
```

Tip: The first character position in a string is 0 (not 1).

str_replace()

The PHP str_replace() function replaces some characters with some other characters in a string.

Example

Replace the text "world" with "Dolly":

```
<?php
echo str_replace("world", "Dolly", "Hello world!"); // outputs Hello Dolly!
?>
```

PHP Numbers

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

<?php
// Check if the type of a variable is integer
$x = 5985;
var_dump(is_int($x));

echo "<br>";
// Check again...
$x = 59.85;
var_dump(is_int($x));
?>
</body>
</html>
```

```
bool(true)
bool(false)
```

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

<?php
// Check if the type of a variable is
float
$x = 10.365;
var_dump(is_float($x));
?>

</body>
</html>
```

```
bool(true)
```

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

<?php
// Check if a numeric value is finite or
infinite
$x = 1.9e411;
var_dump($x);
?>

</body>
</html>
```

```
float(INF)
```

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

<?php
// Invalid calculation will return a
NaN value
$x = acos(8);
var_dump($x);
?>

</body>
</html>
```

```
float(NAN)
```

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

<?php
// Check if the variable is numeric
$x = 5985;
var_dump(is_numeric($x));
echo "<br>";
$x = "5985";
var_dump(is_numeric($x));
echo "<br>";
$x = "59.85" + 100;
var_dump(is_numeric($x));
echo "<br>";
$x = "Hello";
var_dump(is_numeric($x));
?>

</body>
</html>
```

```
bool(true)
bool(true)
bool(true)
bool(false)
```

PHP Constants

- Constants are like variables except that once they are defined they cannot be changed or undefined.
- A constant is an identifier (name) for a simple value. The value cannot be changed during the script.
- A valid constant name starts with a letter or underscore (no $ sign before the constant name).

Create a PHP Constant

To create a constant, use the define() function.

Syntax

define(*name*, *value*, *case-insensitive*)

Parameters:

- *name*: Specifies the name of the constant
- *value*: Specifies the value of the constant
- *case-insensitive*: Specifies whether the constant name should be case-insensitive. Default is false

Example: **Create a constant with a case-sensitive name:**

```
<?php
// case-sensitive constant name
define("GREETING", "Welcome to W3Schools.com!");
echo GREETING;
?>
```

```
Welcome to W3Schools.com!
```

Create a constant with a case-insensitive name:

```
<?php
define("GREETING", "Welcome to W3Schools.com!", true);
echo greeting;
?>
```

```
Welcome to W3Schools.com!
```

PHP Constant Arrays

```
<?php
define("cars", [
    "Alfa Romeo",
    "BMW",
    "Toyota"
]);
echo cars[0];
?>
```

```
Alfa Romeo
```

Constants are Global

```
<?php
define("GREETING", "Welcome to
W3Schools.com!");

function myTest() {
    echo GREETING;
}

myTest();
?>
```

```
Welcome to W3Schools.com!
```

PHP Operators

- Operators are used to perform operations on variables and values.
- PHP divides the operators in the following groups:
  - Arithmetic operators
  - Assignment operators
  - Comparison operators
  - Increment/Decrement operators
  - Logical operators
  - String operators
  - Array operators
  - Conditional assignment operators

PHP Arithmetic Operators

The PHP arithmetic operators are used with numeric values to perform common arithmetical operations, such as addition, subtraction, multiplication etc.

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

<?php
$x = 10;
$y = 6;

echo $x + $y, "<br>";
echo $x - $y, "<br>";
echo $x * $y, "<br>";
echo $x / $y, "<br>";
echo $x % $y, "<br>";
echo $x ** $y, "<br>";
?>

</body>
</html>
```

```
16
4
60
1.6666666666667
4
1000000
```

PHP Assignment Operators

The PHP assignment operators are used with numeric values to write a value to a variable. The basic assignment operator in PHP is "=". It means that the left operand gets set to the value of the assignment expression on the right

```
<!DOCTYPE html>
<html>
<body>
        <?php
        $x = 20;
        echo $x,"<br>";
        $x += 100;
        echo $x,"<br>";
        $x -= 30;
        echo $x,"<br>";
        $x = $x * 10;
        echo $x,"<br>";
        $x = $x / 10;
        echo $x,"<br>";
        ?>
</body>
</html>
```

```
20
120
90
900
90
```

PHP Comparison Operators

The PHP comparison operators are used to compare two values (number or string):

Operator	Name	Example	Result
==	Equal	$x == $y	Returns true if $x is equal to $y
===	Identical	$x === $y	Returns true if $x is equal to $y, and they are of the same type
!=	Not equal	$x != $y	Returns true if $x is not equal to $y
<>	Not equal	$x <> $y	Returns true if $x is not equal to $y
!==	Not identical	$x !== $y	Returns true if $x is not equal to $y, or they are not of the same type
>	Greater than	$x > $y	Returns true if $x is greater than $y
<	Less than	$x < $y	Returns true if $x is less than $y
>=	Greater than or equal to	$x >= $y	Returns true if $x is greater than or equal to $y
<=	Less than or equal to	$x <= $y	Returns true if $x is less than or equal to $y
<=>	Spaceship	$x <=> $y	Returns an integer less than, equal to, or greater than zero, depending on if $x is less than, equal to, or greater than $y. Introduced in PHP 7.

Example

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

<?php
$x = 5;
$y = 10;
echo ($x <=> $y); // returns -1 because $x is less than $y
echo "<br>";
$x = 10;
$y = 10;
echo ($x <=> $y); // returns 0 because values are equal
echo "<br>";
$x = 15;
$y = 10;
echo ($x <=> $y); // returns +1 because $x is greater than $y
?>

</body>
</html>
```

```
-1
0
1
```

PHP Increment / Decrement Operators

The PHP increment operators are used to increment a variable's value. The PHP decrement operators are used to decrement a variable's value.

Operator	Name	Description
++$x	Pre-increment	Increments $x by one, then returns $x
$x++	Post-increment	Returns $x, then increments $x by one
--$x	Pre-decrement	Decrements $x by one, then returns $x
$x--	Post-decrement	Returns $x, then decrements $x by one

Example

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

<?php
$x = 10;
echo ++$x; //outputs 11
?>

</body>
</html>
```

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

<?php
$x = 10;
echo $x++;  //outputs 10
?>

</body>
</html>
```

PHP Logical Operators

The PHP logical operators are used to combine conditional statements.

Operator	Name	Example	Result
and	And	$x and $y	True if both $x and $y are true
or	Or	$x or $y	True if either $x or $y is true
xor	Xor	$x xor $y	True if either $x or $y is true, but not both
&&	And	$x && $y	True if both $x and $y are true
||	Or	$x || $y	True if either $x or $y is true
!	Not	!$x	True if $x is not true

Example

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

<?php
$x = 100;
$y = 50;

if ($x == 100 && $y == 50) {
   echo "Hello world!";  // outputs Hello world!
}
?>

</body>
</html>
```

PHP String Operators

PHP has two operators that are specially designed for strings.

Operator	Name	Example	Result
.	Concatenation	$txt1 . $txt2	Concatenation of $txt1 and $txt2
.=	Concatenation assignment	$txt1 .= $txt2	Appends $txt2 to $txt1

```
<?php
$txt1 = "Hello";
$txt2 = " world!";
echo $txt1 . $txt2;
?>
```

OR

```
<?php
$txt1 = "Hello";
$txt2 = " world!";
$txt1 .= $txt2;
echo $txt1;
?>
```

```
Hello world!
```

PHP Array Operators

The PHP array operators are used to compare arrays.

Operator	Name	Example	Result
+	Union	$x + $y	Union of $x and $y
==	Equality	$x == $y	Returns true if $x and $y have the same key/value pairs
===	Identity	$x === $y	Returns true if $x and $y have the same key/value pairs in the same order and of the same types
!=	Inequality	$x != $y	Returns true if $x is not equal to $y
<>	Inequality	$x <> $y	Returns true if $x is not equal to $y
!==	Non-identity	$x !== $y	Returns true if $x is not identical to $y

Example

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

<?php
$x = array("a" => "red", "b" => "green");
$y = array("c" => "blue", "d" => "yellow");

var_dump($x == $y);  // outputs bool(false)
?>

</body>
</html>
```

PHP Control Statements

- PHP supports a number of different control structures:
    - if
    - else
    - elseif
    - switch
    - while
    - do-while
    - for
    - foreach
    - and more

If

The if construct allows you to execute a piece of code if tahe expression provided along with it evaluates to true.

```
<?php
$age = 50;

if ($age > 30)
{
  echo "Your age is greater than 30!";
}
?>
```

Else

The if construct allows you to execute a piece of code if the expression evaluates to true. On the other hand, if the expression evaluates to false, it won't do anything.

```
<?php
$age = 50;

if ($age < 30)
{
  echo "Your age is less than 30!";
}
else
{
  echo "Your age is greater than or
equal 30!";
}
?>
```

If-elseif-else

```
<?php
$age = 50;

if ($age < 30)
{
  echo "Your age is less than 30!";
}
elseif ($age > 30 && $age < 40)
{
  echo "Your age is between 30 and 40!";
}
elseif ($age > 40 && $age < 50)
{
  echo "Your age is between 40 and 50!";
}
else
{
  echo "Your age is greater than 50!";
}
?>
```

Switch statement

```
<?php
$favourite_site = 'Code';
 switch ($favourite_site) {
  case 'Business':
    echo "My favourite site is business.tutsplus.com!";
    break;
  case 'Code':
    echo "My favourite site is code.tutsplus.com!";
    break;
  case 'Web Design':
    echo "My favourite site is webdesign.tutsplus.com!";
    break;
  case 'Music':
    echo "My favourite site is music.tutsplus.com!";
    break;
  case 'Photography':
    echo "My favourite site is photography.tutsplus.com!";
    break;
  default:
    echo "I like everything at tutsplus.com!";
}
?>
```

While Loop

The while loop is used when you want to execute a piece of code repeatedly until the while condition evaluates to false.

```
<?php
$max = 0;
echo $i = 0;
echo ",";
echo $j = 1;
echo ",";
$result=0;

while ($max < 10 )
{
    $result = $i + $j;

    $i = $j;
    $j = $result;

    $max = $max + 1;
    echo $result;
    echo ",";
}
?>
```

Do-While Loop

The do-while loop is very similar to the while loop, with the only difference being that the while condition is checked at the end of the first iteration. Thus, we can guarantee that the loop code is executed at least once, irrespective of the result of the while expression.

```
<?php
$handle = fopen("file.txt", "r");
if ($handle)
{
    do
    {
        $line = fgets($handle);

        // process the line content

    } while($line !== false);
}
fclose($handle);
?>
```

For Loop

The for loop is used to execute a piece of code for a specific number of times.

```
<?php
for ($i=1; $i<=10; ++$i)
{
  echo sprintf("The square of %d is %d.</br>", $i, $i*$i);
}
?>
```

For Each

The foreach loop is used to iterate over array variables.

```
<?php
$fruits = array('apple', 'banana', 'orange', 'grapes');
foreach ($fruits as $fruit)
{   echo $fruit;    echo "<br/>";  }
 $employee = array('name' => 'John Smith', 'age' => 30, 'profession' => 'Software
Engineer');
foreach ($employee as $key => $value)
{    echo sprintf("%s: %s</br>", $key, $value);    echo "<br/>"; }
?>
```

PHP Functions

- A function is a block of statements that can be used repeatedly in a program.
- A function will not execute automatically when a page loads.
- A function will be executed by a call to the function.
- Syntax

```
function functionName() {
    code to be executed;
  }
```

Note: A function name must start with a letter or an   underscore. Function names are NOT case-sensitive.

EXAMPLE:

```
<?php
function writeMsg() {
   echo "Hello world!";
}
writeMsg(); // call the function
?>
```

PHP Function Arguments

Information can be passed to functions through arguments. An argument is just like a variable. Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.

```
<?php
function familyName($fname, $year) {
   echo "$fname Refsnes. Born in $year <br>";
}

familyName("Hege", "1975");
familyName("Stale", "1978");
familyName("Kai Jim", "1983");
?>
```

```
Hege Refsnes. Born in 1975
Stale Refsnes. Born in 1978
Kai Jim Refsnes. Born in 1983
```

PHP Arrays

- An array stores multiple values in one single variable.
- Example

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
```

```
I like Volvo, BMW and Toyota.
```

Get The Length of an Array - The count() Function

The `count()` function is used to return the length (the number of elements) of an array:

Example

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo count($cars);
?>
```

In PHP, there are three types of arrays:
Indexed arrays - Arrays with a numeric index
Associative arrays - Arrays with named keys
Multidimensional arrays - Arrays containing one or more arrays

PHP Indexed Arrays
There are two ways to create indexed arrays:
The index can be assigned automatically (index always starts at 0), like this:

```
$cars = array("Volvo", "BMW", "Toyota");
```

or the index can be assigned manually:

```
$cars[0] = "Volvo";
$cars[1] = "BMW";
$cars[2] = "Toyota";
```

The following example creates an indexed array named $cars, assigns three elements to it, and then prints a text containing the array values:
Example

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
```

Loop Through an Indexed Array

To loop through and print all the values of an indexed array, you could use a `for` loop, like this:

Example

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
$arrlength = count($cars);

for($x = 0; $x < $arrlength; $x++) {
    echo $cars[$x];
    echo "<br>";
}
?>
```

PHP Associative Arrays

Associative arrays are arrays that use named keys that you assign to them.

There are two ways to create an associative array:

```
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
```

or

```
$age['Peter'] = "35";
$age['Ben'] = "37";
$age['Joe'] = "43";
```

The named keys can then be used in a script:

Example

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
echo "Peter is " . $age['Peter'] . " years old.";
?>
```

Loop Through an Associative Array

To loop through and print all the values of an associative array, you could use a foreach loop, like this:

Example

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

foreach($age as $x => $x_value) {
   echo "Key=" . $x . ", Value=" . $x_value;
   echo "<br>";
}
?>
```

PHP - Multidimensional Arrays

A multidimensional array is an array containing one or more arrays.
PHP supports multidimensional arrays that are two, three, four, five, or more levels deep. However, arrays more than three levels deep are hard to manage for most people.

Example

```
<?php
echo $cars[0][0].": In stock: ".$cars[0][1].", sold: ".$cars[0][2].".<br>";
echo $cars[1][0].": In stock: ".$cars[1][1].", sold: ".$cars[1][2].".<br>";
echo $cars[2][0].": In stock: ".$cars[2][1].", sold: ".$cars[2][2].".<br>";
echo $cars[3][0].": In stock: ".$cars[3][1].", sold: ".$cars[3][2].".<br>";
?>
```

We can also put a `for` loop inside another `for` loop to get the elements of the $cars array (we still have to point to the two indices):

Example

```
<?php
for ($row = 0; $row < 4; $row++) {
  echo "<p><b>Row number $row</b></p>";
  echo "<ul>";
  for ($col = 0; $col < 3; $col++) {
    echo "<li>".$cars[$row][$col]."</li>";
  }
  echo "</ul>";
}
?>
```

PHP Sorting Arrays

In this chapter, we will go through the following PHP array sort functions:

- sort() - sort arrays in ascending order
- rsort() - sort arrays in descending order
- asort() - sort associative arrays in ascending order, according to the value
- ksort() - sort associative arrays in ascending order, according to the key
- arsort() - sort associative arrays in descending order, according to the value
- krsort() - sort associative arrays in descending order, according to the key

Sort Array in Ascending Order - sort()

The following example sorts the elements of the $cars array in ascending alphabetical order:

Example

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
sort($cars);
?>
```

```
BMW
Toyota
Volvo
```

The following example sorts the elements of the $numbers array in ascending numerical order:

Example

```
<?php
$numbers = array(4, 6, 2, 22, 11);
sort($numbers);
?>
```

```
2
4
6
11
22
```

Sort Array in Descending Order - rsort()
The following example sorts the elements of the $cars array in descending alphabetical order:
Example
```
<?php
$cars = array("Volvo", "BMW", "Toyota");
rsort($cars);
?>
```
```
Volvo
Toyota
BMW
```

Sort Array (Ascending Order), According to Value - asort()
The following example sorts an associative array in ascending order, according to the value:
Example
```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
asort($age);
?>
```
```
Key=Peter, Value=35
Key=Ben, Value=37
Key=Joe, Value=43
```

Sort Array (Ascending Order), According to Key - ksort()

The following example sorts an associative array in ascending order, according to the key:

Example

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
ksort($age);
?>
```

```
Key=Ben, Value=37
Key=Joe, Value=43
Key=Peter, Value=35
```

Sort Array (Descending Order), According to Value - arsort()

The following example sorts an associative array in descending order, according to the value:

Example

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
arsort($age);
?>
```

```
Key=Joe, Value=43
Key=Ben, Value=37
Key=Peter, Value=35
```

Sort Array (Descending Order), According to Key - krsort()

The following example sorts an associative array in descending order, according to the key:

Example

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
krsort($age);
?>
```

```
Key=Peter, Value=35
Key=Joe, Value=43
Key=Ben, Value=37
```

PHP Regular Expressions

- A regular expression is a sequence of characters that forms a search pattern. When you search for data in a text, you can use this search pattern to describe what you are searching for.
- A regular expression can be a single character, or a more complicated pattern.
- Regular expressions can be used to perform all types of text search and text replace operations.

Syntax

- In PHP, regular expressions are strings composed of delimiters, a pattern and optional modifiers.

$exp = "/w3schools/i";

- In the example above, / is the **delimiter**, *w3schools* is the **pattern** that is being searched for, and i is a **modifier** that makes the search case-insensitive.
- The delimiter can be any character that is not a letter, number, backslash or space. The most common delimiter is the forward slash (/), but when your pattern contains forward slashes it is convenient to choose other delimiters such as # or ~.

Regular Expression Functions

PHP provides a variety of functions that allow you to use regular expressions.
The preg_match(), preg_match_all() and preg_replace() functions are some of the most commonly used ones:

Using preg_match()

The preg_match() function will tell you whether a string contains matches of a pattern.

```
<?php
$str = "Visit W3Schools";
$pattern = "/w3schools/i";
echo preg_match($pattern, $str); // Outputs 1
?>
```

Using preg_match_all()

The preg_match_all() function will tell you how many matches were found for a pattern in a string.

```
<?php
$str = "The rain in SPAIN falls mainly on the plains.";
$pattern = "/ain/i";
echo preg_match_all($pattern, $str); // Outputs 4
?>
```

Using preg_replace()

The preg_replace() function will replace all of the matches of the pattern in a string with another string.

```
<?php
$str = "Visit Microsoft!";
$pattern = "/microsoft/i";
echo preg_replace($pattern, "W3Schools", $str); // Outputs "Visit W3Schools!"
?>
```

Regular Expression Modifiers

Modifiers can change how a search is performed.

Modifier	**Description**
i	Performs a case-insensitive search
m	Performs a multiline search (patterns that search for the beginning or end of a string will match the beginning or end of each line)
u	Enables correct matching of UTF-8 encoded patterns

Regular Expression Patterns

Brackets are used to find a range of characters:

Expression	Description
[abc]	Find one character from the options between the brackets
[^abc]	Find any character NOT between the brackets
[0-9]	Find one character from the range 0 to 9

Metacharacters

Metacharacters are characters with a special meaning:

Metacharacter	Description
|	Find a match for any one of the patterns separated by | as in: cat|dog|fish
.	Find just one instance of any character
^	Finds a match as the beginning of a string as in: ^Hello
$	Finds a match at the end of the string as in: World$
\d	Find a digit
\s	Find a whitespace character
\b	Find a match at the beginning of a word like this: \bWORD, or at the end of a word like this: WORD\b
\uxxxx	Find the Unicode character specified by the hexadecimal number xxxx

Quantifiers

Quantifiers define quantities:

Quantifier	**Description**
n+	Matches any string that contains at least one *n*
n*	Matches any string that contains zero or more occurrences of *n*
n?	Matches any string that contains zero or one occurrences of *n*
n{x}	Matches any string that contains a sequence of *X* *n*'s
n{x,y}	Matches any string that contains a sequence of X to Y *n*'s
n{x,}	Matches any string that contains a sequence of at least X *n*'s

Grouping

You can use parentheses ( ) to apply quantifiers to entire patterns. They also can be used to select parts of the pattern to be used as a match.

Example

Use grouping to search for the word "banana" by looking for *ba* followed by two instances of *na*:

```
<?php
$str = "Apples and bananas.";
$pattern = "/ba(na){2}/i";
echo preg_match($pattern, $str); // Outputs 1
?>
```

PHP Form Handling

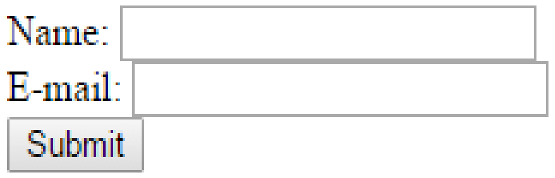
The PHP superglobals $_GET and $_POST are used to collect form-data.

PHP - A Simple HTML Form

The example below displays a simple HTML form with two input fields and a submit button:

Example

```
<html>
<body>

<form action="welcome.php" method="post">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>

</body>
</html>
```



When the user fills out the form above and clicks the submit button, the form data is sent for processing to a PHP file named "welcome.php". The form data is sent with the HTTP POST method.
To display the submitted data you could simply echo all the variables. The **"welcome.php"** looks like this:

```
<html>
<body>

Welcome <?php echo $_POST["name"]; ?><br>
Your email address is: <?php echo $_POST["email"]; ?>

</body>
</html>
```

The **output** could be something like this:

```
Welcome John
Your email address is john.doe@example.com
```

The same result could also be achieved using the HTTP GET method:

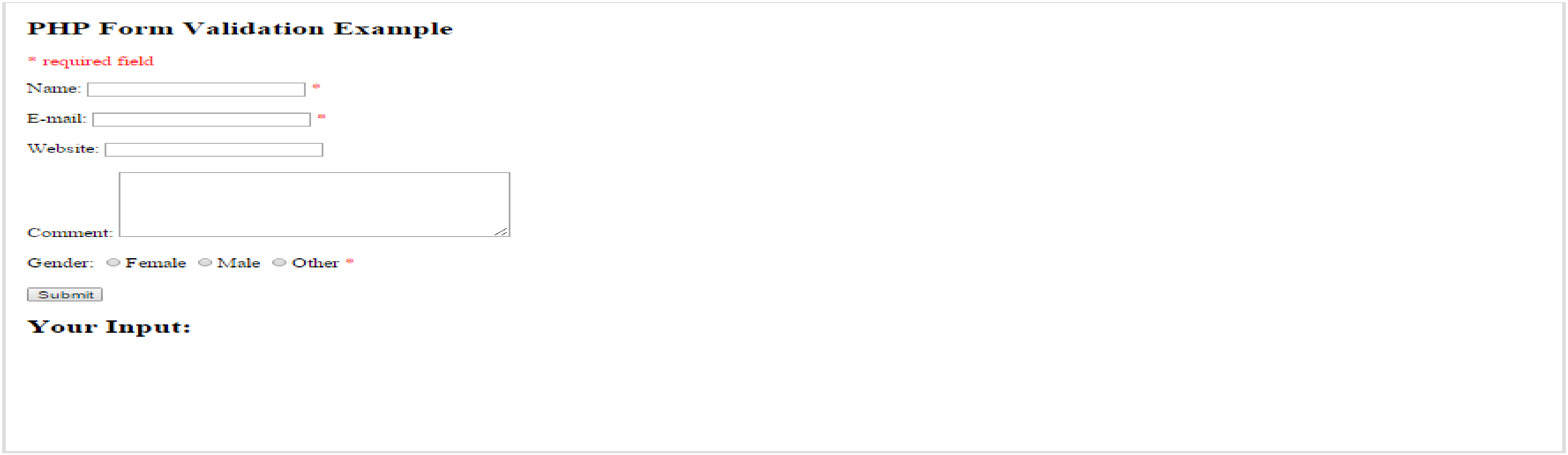
Example

```
<html>
<body>
<form action="welcome_get.php" method="get">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>
</body>
</html>
```

and **"welcome_get.php"** looks like this:

```
<html>
<body>
Welcome <?php echo $_GET["name"]; ?><br>
Your email address is: <?php echo $_GET["email"]; ?>
</body>
</html>
```

PHP Form Validation



PHP Form Validation Example

* required field

Name: *

E-mail: *

Website:

Comment:

Gender: ○ Female ○ Male ○ Other *

Your Input:

The validation rules for the form above are as follows:

Field	Validation Rules
Name	Required. + Must only contain letters and whitespace
E-mail	Required. + Must contain a valid email address (with @ and .)
Website	Optional. If present, it must contain a valid URL
Comment	Optional. Multi-line input field (textarea)
Gender	Required. Must select one

Text Fields

The name, email, and website fields are text input elements, and the comment field is a textarea. The HTML code looks like this:

```
Name: <input type="text" name="name">
E-mail: <input type="text" name="email">
Website: <input type="text" name="website">
Comment: <textarea name="comment" rows="5" cols="40"></textarea>
```

Radio Buttons

The gender fields are radio buttons and the HTML code looks like this:

```
Gender:
<input type="radio" name="gender" value="female">Female
<input type="radio" name="gender" value="male">Male
<input type="radio" name="gender" value="other">Other
```

The Form Element

The HTML code of the form looks like this:

```
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>"
>
```

When the form is submitted, the form data is sent with method="post".

What is the $_SERVER["PHP_SELF"] variable?

The $_SERVER["PHP_SELF"] is a super global variable that returns the filename of the currently executing script.

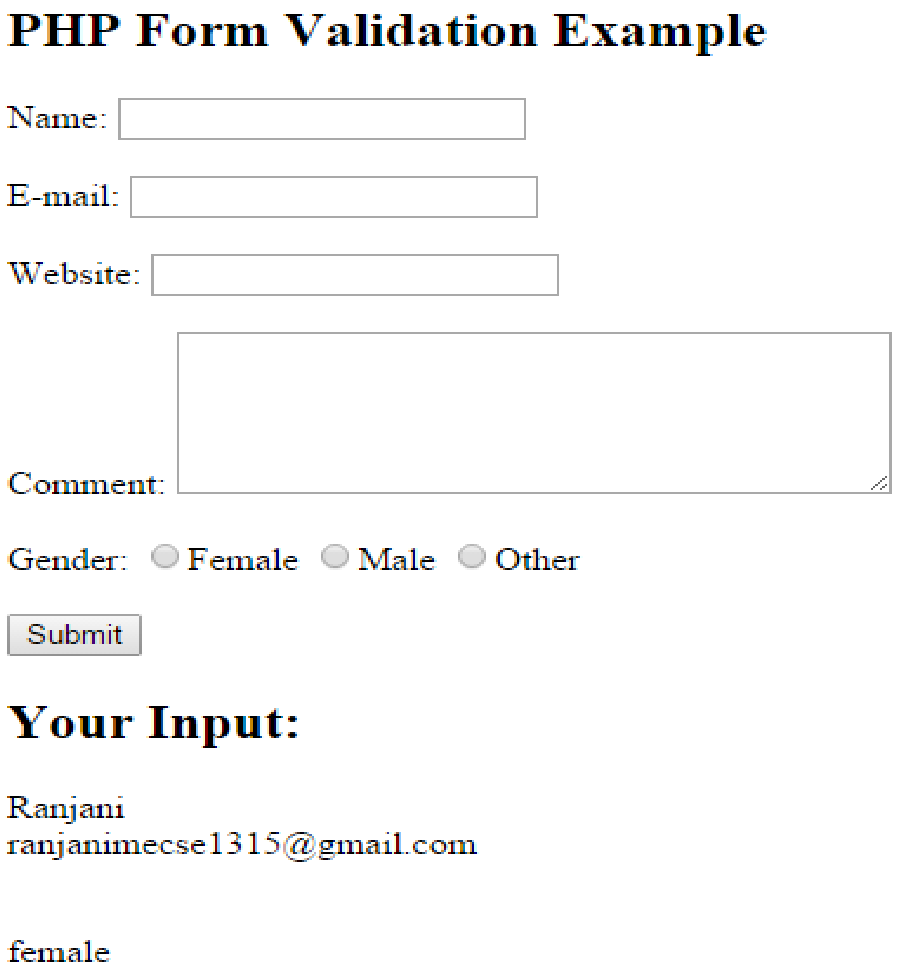
What is the htmlspecialchars() function?

The htmlspecialchars() function converts special characters to HTML entities. This means that it will replace HTML characters like < and > with &lt; and &gt;. This prevents attackers from exploiting the code by injecting HTML or Javascript code (Cross-site Scripting attacks) in forms.

```
<!DOCTYPE HTML>
<html>
<head>
</head>
<body>
<?php
// define variables and set to empty values
$name = $email = $gender = $comment = $website = "";
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  $name = test_input($_POST["name"]);
  $email = test_input($_POST["email"]);
  $website = test_input($_POST["website"]);
  $comment = test_input($_POST["comment"]);
  $gender = test_input($_POST["gender"]);
}

function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
?>
```

```
<h2>PHP Form Validation Example</h2>
<form method="post" action="<?php echohtmlspecialchars($_SERVER["PHP_SELF"]);?>">
  Name: <input type="text" name="name">
  <br><br>
  E-mail: <input type="text" name="email">
  <br><br>
  Website: <input type="text" name="website">
  <br><br>
  Comment: <textarea name="comment" rows="5" cols="40"></textarea>
  <br><br>
  Gender:
  <input type="radio" name="gender" value="female">Female
  <input type="radio" name="gender" value="male">Male
  <input type="radio" name="gender" value="other">Other
  <br><br>
  <input type="submit" name="submit" value="Submit">
</form>
```

```
<?php
echo "<h2>Your Input:</h2>";
echo $name;
echo "<br>";
echo $email;
echo "<br>";
echo $website;
echo "<br>";
echo $comment;
echo "<br>";
echo $gender;
?>

</body>
</html>
```



PHP Form Validation Example

Name:

E-mail:

Website:

Comment:

Gender: ○ Female ○ Male ○ Other

Submit

Your Input:

```
Ranjani
ranjanimecse1315@gmail.com


female
```

PHP File Handling

PHP File Create/Write

PHP Create File - fopen()

The fopen() function is also used to create a file. Maybe a little confusing, but in PHP, a file is created using the same function used to open files.
If you use fopen() on a file that does not exist, it will create it, given that the file is opened for writing (w) or appending (a).

Example

```
$myfile = fopen("testfile.txt", "w")
```

PHP File Permissions

If you are having errors when trying to get this code to run, check that you have granted your PHP file access to write information to the hard drive.

PHP Write to File - fwrite()

The fwrite() function is used to write to a file.
The first parameter of fwrite() contains the name of the file to write to and the second parameter is the string to be written.
The example below writes a couple of names into a new file called "newfile.txt":

Example

```
<?php
$myfile = fopen("newfile.txt", "w") or die("Unable to open file!");
$txt = "John Doe\n";
fwrite($myfile, $txt);
$txt = "Jane Doe\n";
fwrite($myfile, $txt);
fclose($myfile);
?>
```

Notice that we wrote to the file "newfile.txt" twice. Each time we wrote to the file we sent the string $txt that first contained "John Doe" and second contained "Jane Doe". After we finished writing, we closed the file using the fclose() function.

If we open the "newfile.txt" file it would look like this:

```
John Doe
Jane Doe
```

PHP Overwriting

Now that "newfile.txt" contains some data we can show what happens when we open an existing file for writing. All the existing data will be ERASED and we start with an empty file.

In the example below we open our existing file "newfile.txt", and write some new data into it:

Example

```
<?php
$myfile = fopen("newfile.txt", "w") or die("Unable to open file!");
$txt = "Mickey Mouse\n";
fwrite($myfile, $txt);
$txt = "Minnie Mouse\n";
fwrite($myfile, $txt);
fclose($myfile);
?>
```

If we now open the "newfile.txt" file, both John and Jane have vanished, and only the data we just wrote is present:

```
Mickey Mouse
Minnie Mouse
```

PHP Cookies

What is a Cookie?
A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.

Create Cookies With PHP
A cookie is created with the setcookie() function.
Syntax

setcookie(*name, value, expire, path, domain, secure, httponly*);

Only the *name* parameter is required. All other parameters are optional.

PHP Create/Retrieve a Cookie

The following example creates a cookie named "user" with the value "John Doe". The cookie will expire after 30 days (86400 * 30). The "/" means that the cookie is available in entire website (otherwise, select the directory you prefer).

We then retrieve the value of the cookie "user" (using the global variable $_COOKIE). We also use the `isset()` function to find out if the cookie is set:

Example

```
<?php
$cookie_name = "user";
$cookie_value = "John Doe";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/"); // 86400 =
1 day
?>
<html>
<body>
<?php
if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
    echo "Cookie '" . $cookie_name . "' is set!<br>";
    echo "Value is: " . $_COOKIE[$cookie_name];
}
?>
</body>
</html>
```

```
Cookie 'user' is set!
Value is: Alex Porter
```

Note: You might have to reload the page to see the value of the cookie.

Modify a Cookie Value

To modify a cookie, just set (again) the cookie using the `setcookie()` function:

Example

```
<?php
$cookie_name = "user";
$cookie_value = "Alex Porter";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");
?>
<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
    echo "Cookie '" . $cookie_name . "' is set!<br>";
    echo "Value is: " . $_COOKIE[$cookie_name];
}
?>

</body>
</html>
```

```
Cookie 'user' is set!
Value is: John Doe
```

Note: You might have to reload the page to see the new value of the cookie.

Delete a Cookie

To delete a cookie, use the `setcookie()` function with an expiration date in the past:

Example

```
<?php
// set the expiration date to one hour ago
setcookie("user", "", time() - 3600);
?>
<html>
<body>

<?php
echo "Cookie 'user' is deleted.";
?>

</body>
</html>
```

```
Cookie 'user' is deleted.
```

Check if Cookies are Enabled
The following example creates a small script that checks whether cookies are enabled. First, try to create a test cookie with the `setcookie()` function, then count the $_COOKIE array variable:

Example

```
<?php
setcookie("test_cookie", "test", time() + 3600, '/');
?>
<html>
<body>

<?php
if(count($_COOKIE) > 0) {
    echo "Cookies are enabled.";
} else {
    echo "Cookies are disabled.";
}
?>

</body>
</html>
```

```
Cookies are enabled.
```

PHP Sessions

A session is a way to store information (in variables) to be used across multiple pages. When you work with an application, you open it, do some changes, and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are or what you do, because the HTTP address doesn't maintain state.

Start a PHP Session

A session is started with the session_start() function.
Session variables are set with the PHP global variable: $_SESSION.
Now, let's create a new page called "demo_session1.php". In this page, we start a new PHP session and set some session variables:

Example

```
<?php
// Start the session
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// Set session variables
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat";
echo "Session variables are set.";
?>

</body>
</html>
```

```
Session variables are set.
```

Note: The session_start() function must be the very first thing in your document. Before any HTML tags.

Get PHP Session Variable Values

Next, we create another page called "demo_session2.php". From this page, we will access the session information we set on the first page ("demo_session1.php").

Notice that session variables are not passed individually to each new page, instead they are retrieved from the session we open at the beginning of each page (session_start()).

Also notice that all session variable values are stored in the global $_SESSION variable:

Example

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>
<?php
// Echo session variables that were set on
previous page
echo "Favorite color is
" . $_SESSION["favcolor"] . ".<br>";
echo "Favorite animal is
" . $_SESSION["favanimal"] . ".";
?>
</body>
</html>
```

```
Favorite color is green.
Favorite animal is cat.
```

Modify a PHP Session Variable

To change a session variable, just overwrite it:

Example

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// to change a session variable, just overwrite
it
$_SESSION["favcolor"] = "yellow";
print_r($_SESSION);
?>

</body>
</html>
```

```
Array ( [favcolor] => yellow [favanimal] => cat )
```

Destroy a PHP Session

To remove all global session variables and destroy the session, use session_unset() and session_destroy():

Example

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// remove all session variables
session_unset();

// destroy the session
session_destroy();
?>

</body>
</html>
```

All session variables are now removed, and the session is destroyed.

Database Connectivity

PHP MySQL Database

- With PHP, you can connect to and manipulate databases.
- MySQL is the most popular database system used with PHP.

What is MySQL?

- MySQL is a database system used on the web
- MySQL is a database system that runs on a server
- MySQL is ideal for both small and large applications
- MySQL is very fast, reliable, and easy to use
- MySQL uses standard SQL
- MySQL compiles on a number of platforms
- MySQL is free to download and use
- MySQL is developed, distributed, and supported by Oracle Corporation

The data in a MySQL database are stored in tables. A table is a collection of related data, and it consists of columns and rows.Databases are useful for storing information categorically.

PHP + MySQL Database System
PHP combined with MySQL are cross-platform (you can develop in Windows and serve on a Unix platform)

Database Queries
A query is a question or a request.
We can query a database for specific information and have a recordset returned.
Look at the following query (using standard SQL):

```
SELECT LastName FROM Employees
```

The query above selects all the data in the "LastName" column from the "Employees" table.

Download PHP Server with MySQL

Open a Connection to MySQL

Before we can access data in the MySQL database, we need to be able to connect to the server:

Example (MySQLi Object-Oriented)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
// Create connection
$conn = new mysqli($servername, $username, $password);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";
?>
```

Close the Connection

The connection will be closed automatically when the script ends. To close the connection before, use the following:

```
$conn->close();
```

Create a MySQL Database

The CREATE DATABASE statement is used to create a database in MySQL.

The following example create a database named "myDB":

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
// Create connection
$conn = new mysqli($servername, $username, $password);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}
// Create database
$sql = "CREATE DATABASE myDB";
if ($conn->query($sql) === TRUE) {
  echo "Database created successfully";
} else {
  echo "Error creating database: " . $conn->error;
}
$conn->close();
?>
```

How to Connect Mysql with PHP

When many developers refer to a database, they are usually referring to MySQL, a very popular DBMS that powers projects of all sizes. The USP of MySQL is its ability to handle huge volumes of data without breaking a sweat.

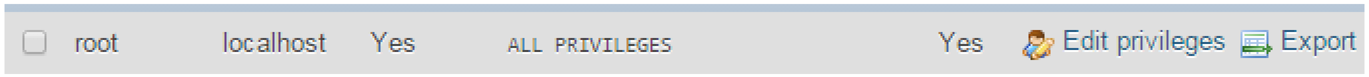
In this article I will discuss how to connect MySQL Database on different servers and also give an overview of connecting Database using PDO.

- ➢ Connect MySQL using Localhost Server
- ➢ Connect MySQL using Cloudways Server
- ➢ Connect MySQL using PDO
- ➢ Connect MySQL using Remote MySQL

Create MySQL Database at the Localhost

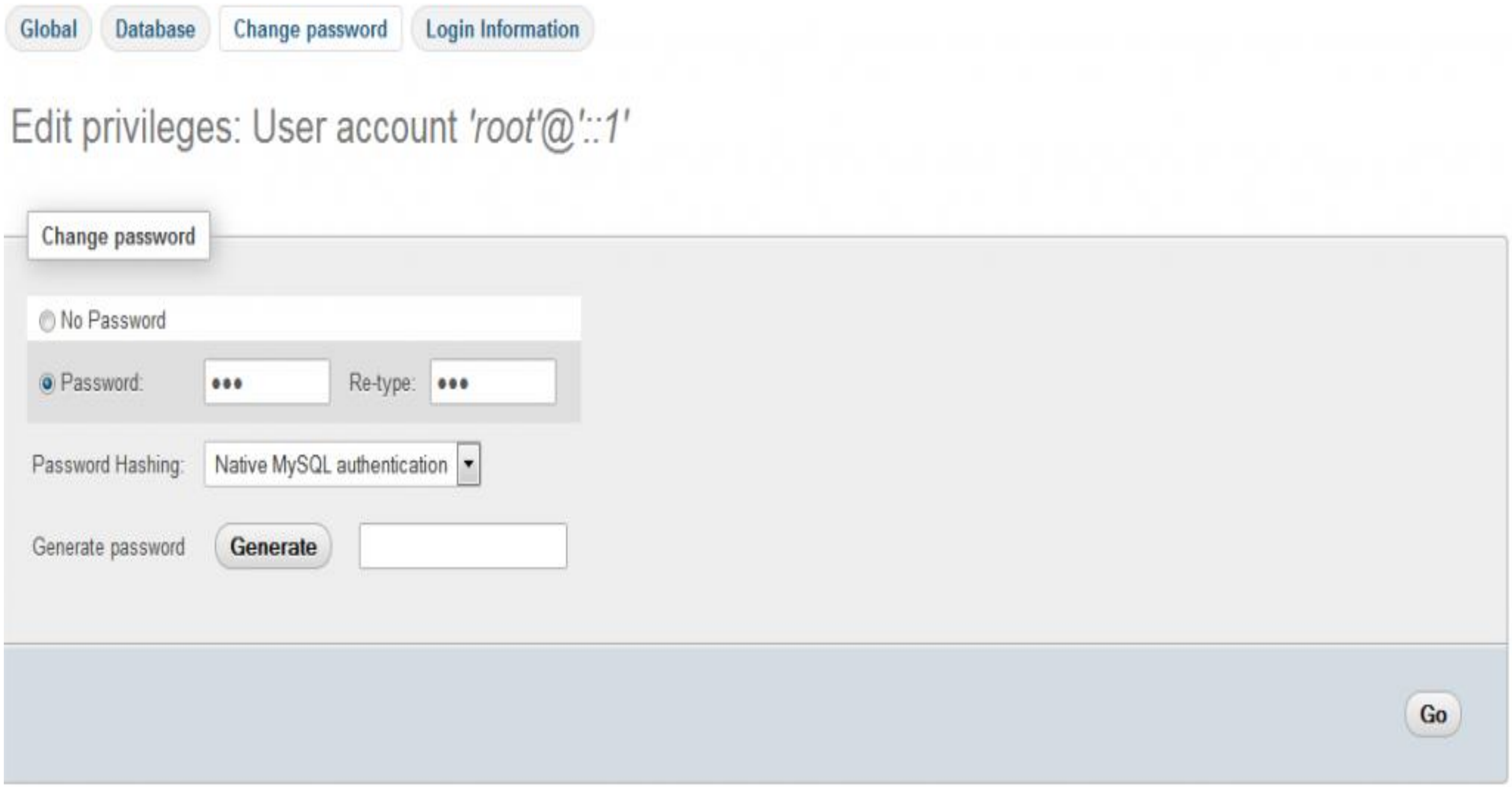
First, let me tell you what PHPMyAdmin is. It is a control panel from where you can manage your database that you have created. Open your browser and go to localhost/PHPMyAdmin or click “Admin” in XAMPP UI.

When you first installed XAMPP, it only created the username for it to be accessed, you now have to add a password to it by yourself. For this, you have to go to User account where the user is same as the one shown in this picture:



☐	root	localhost	Yes	ALL PRIVILEGES	Yes	Edit privileges Export

Now click *Edit privileges* and go to Change Admin password, type your password there and save it. Remember this password as it will be use to connect to your Database.



Edit privileges: User account *'root'@'::1'*

Change password

No Password

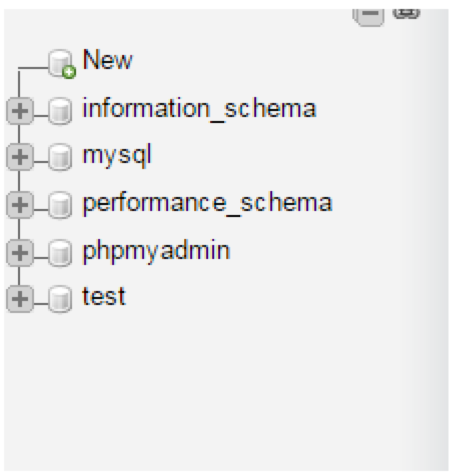
[✓ Password:] ••• Re-type: •••

Password Hashing: Native MySQL authentication

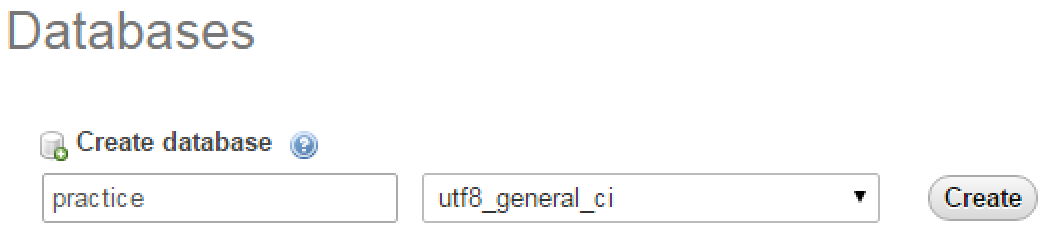
Note: It is not necessary to change password to access databases on local host. It is a good practice and that is why we have used a password.

Create Database

Now return to the homepage of phpmyadmin. Click *New* button to create a new database.



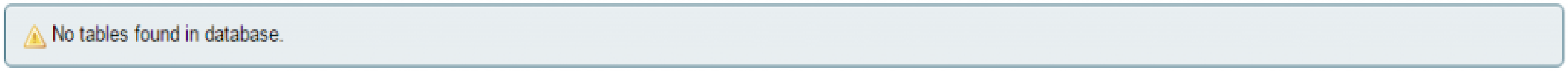
In the new window, name your database as per your need, I am naming it "**practice**". Now select Collation as utf8_general_ci, as we are using it for learning purposes and it will handle all of our queries and data that will be covered in this tutorial series. Now click on *Create* and your database will be created.



practice

utf8_general_ci

The newly created database will be empty now, as there are no tables in it. I will be covering that in the upcoming series where we will learn how to create tables and insert data in it. In this tutorial, we are going to connect this database to a localhost using PHP.



No tables found in database.

Create a Folder in htdocs

Now, locate the folder where you installed XAMPP and open htdocs folder (usually c:/xampp).
Create a new folder inside c:/xampp/htdocs/ and name it “practice” we will place web files in this folder.
Why we have created folder in htdocs?
XAMPP uses folders in htdocs to execute and run your PHP sites.

Note: If your using WAMP, then add your practice folder in c:/wamp/www folder.

Create Database Connection File In PHP

Create a new php file and name it *db_connnection.php* and save it. Why am I creating a separate database connection file? Because if you have created multiple files in which you want to insert data or select data from the databases, you don’t need to write the code for database connection every time. You just have to include it by using PHP custom function ***include*** (include ‘connection.php’) on the top of your code and call its function and use it. It also helps when you are moving your project location from one PC to another and you have to change the values on the single file and all the changes will be applied to all the other files automatically. Write the following code in your db_connection file.

```
<?php
function OpenCon()
{
$dbhost = “localhost”;
$dbuser = “root”;
$dbpass = “1234”;
$db = “example”;
$conn = new mysqli($dbhost, $dbuser, $dbpass,$db) or die(“Connect failed: %s\n”. $conn
-> error);
return $conn;
}
```

```
function CloseCon($conn)
{
$conn -> close();
}
?>
```

Here is the explanation of the variable that we have used in our db_connection file:
$dbhost will be the host where your server is running it is usually localhost.
$dbuser will be the username i.e. **root** and $dbpass will be the password which is the same that you used to access your phpmyadmin.
$dbname will be the name of your database which we have created in this tutorial.

Create new php file to check your database connection

Create a new php file to connect to your database. Name it index.php and add this code in this file.

```
<?php
include ‘db_connection.php’;
$conn = OpenCon();
echo “Connected Successfully”;
CloseCon($conn);
?>
```

Run it!
Now open your browser and goto localhost/practice/index.php and you should see this screen:

```
connected successfully
```

Confirmation Message

Congratulations! You’ve successfully connected your database with your localhost! If you are not able to see this screen, then check if you have done everything right in your db_connection.php file.

References

- https://www.w3schools.com/PHP/DEfaULT.asP
- https://www.tutorialspoint.com/php/index.htm
- https://www.tutorialrepublic.com/php-tutorial/
- https://www.javatpoint.com/php-tutorial
- https://www.phptpoint.com/php-tutorial/

UNIT 5
RAILS & AJAX

CONTENTS

RAILS - Overview of Rails- Document Requests- Processing Forms- Rails Application with Databases – Layouts

AJAX - Ajax Overview of Ajax – Basics of Ajax – Rails with Ajax.

Rails - Overview

- Rails is a web application development framework written in the Ruby programming language. It is designed to make programming web applications easier by making assumptions about what every developer needs to get started. It allows you to write less code while accomplishing more than many other languages and frameworks.
- Rails combines the Ruby programming language with HTML, CSS, and JavaScript to create a web application that runs on a web server. Because it runs on a web server, Rails is considered a server-side, or "back end," web application development platform (the web browser is the "front end").
- The Rails philosophy includes two major guiding principles:
  - **Don't Repeat Yourself:** DRY is a principle of software development which states that "Every piece of knowledge must have a single, unambiguous, authoritative representation within a system". By not writing the same information over and over again, our code is more maintainable, more extensible and less buggy.
  - **Convention Over Configuration:** Rails has opinions about the best way to do many things in a web application, and defaults to this set of conventions, rather than require that you specify minutiae through endless configuration files.

What is Rails?

- an application framework
    - › full stack: web server, actions, database
- a programming environment
    - › eg, rake (like make), unit testing
- an open-source community
    - › many plugins

History of Rails

- genesis in Basecamp
    - › project management tool by 37signals
- release
    - › open source in 2004
    - › shipped with OS X 10.5 in 2007
    - › Rails 3.1 in 2011, merging with Merb

Ruby on Rails creation

Ruby on Rails was created by David Heinemeier Hansson (DHH). He was working at 37signals (now Basecamp) company to create a project management application in Ruby. To help speed along the process, he created a custom web framework Ruby on Rails. It is also called Rails.

Ruby on Rails Features

- ➢ Rails 5 was launched on 18th September 2015 by David Heinemeier Hansson in Atlanta. Some new features were implemented in Rails 5 version.
- ➢ Some features are listed below:
  - ✓ Symbol garbage collector
  - ✓ Module #prepend
  - ✓ Keyword arguments
  - ✓ Action Mailer
  - ✓ Action view
  - ✓ Turbolinks
  - ✓ Action cable
  - ✓ Actionpack Assertions
  - ✓ Rails API
  - ✓ Render from anywhere
  - ✓ Rake command
  - ✓ Customized library
  - ✓ AJAX library

Symbol Garbage Collector
Passing symbols opens the possibility of several attacks in your system. The symbol garbage collector collects the symbols which prevents your system from several attacks.

Module #prepend
It allows you to insert a module in front of the class it was prepended.

Keyword Arguments
It supports keyword arguments which helps to reduce memory consumption by Rails application.

Action Mailer
New methods deliver_now or deliver_later are used instead of #deliver and #deliver!.

Action View
Helper methods like content_tag_for and div_for were removed from the core and moved out to a separate gem.

Turbolinks
Sometimes web pages reloads very slow because it loads full page from the server. Turbolinks 3 reloads only the content of the body, it doesn't reloads the whole page.

Action Cable
It is a framework which is used to extend Rails via Websockets to add some functionality. It very smoothly integrates Websockets with the rest of the Rails application. It allows you to easily add some real time features to your app.

ActionPack Assertions
The assertions assert_template and assigns() are deprecated and moved into its own gem.

Rails API
It allows you to generate API and cleans all the middleware which is not necessary for an application.

Render From Anywhere
Earlier we used gem render_anywhere to render views outside controller. In Rails 5, you can render your views from anywhere.

Rake Command
Rails 5 provides you a feature which allows you to restart all your apps with the **rake restart** command.

Customized URL
Search engine friendly URLs can be developed in Rails.

AJAX Library
Rails provide you an extensive library of AJAX functions. The associated java scripting required for AJAX gets generated automatically.

Rails Scripts

- Rails provides us some excellent tools that are used to develop Rails application. These tools are packaged as scripts from command line.
- Following are the most useful Rails scripts used in Rails application:
  - Rails Console
  - WEBrick Web Server
  - Generators
  - Migrations

Rails Console

The Rails console is as command line utility which runs Rails application from the command line. The Rails console is an extension of Ruby irb. It provides all the features of irb along with the ability to auto-load Rails application environment, including all its classes and components. It helps you to walk through your application step-by-step.

WEBrick Web server

Rails is configured to automatically use WEBrick server. This server is written in pure Ruby and supports almost all platforms like Windows, Mac or Unix. Alternatively, if you have Mongrel or light tpd server installed in your system, Rails uses either of those servers.

All the three Rails servers feature automatic reloading of code. It means, when you change your source code, you do not need to restart the server.

Generators
The Rails include code generation scripts, which are used to automatically generate model and controller classes for an application. Code generation increases your productivity when developing Web applications. By running generator command, skeleton files for all your model and controller classes will be generated. It also generates, database migration files for each model it generates.

Migrations
Migrations bring Rails DRY feature to life. It is a pure Ruby code that define the structure of a database. You don't have to use SQL to write your code while using migration.
The changes you make to your database schema is isolated in a separate migration file, which has a method to implement or reverse the change.

Ruby on Rails Directory Structure

- On creating a Rails application, the entire Rails directory structure is created. We will explain Rails 5 directory structure here.
- The **jtp** directory (shown below) has a number of auto-generated files and folders that comprises the structure of Rails application.

Ruby on Rails - Framework

- A framework is a program, set of programs, and/or code library that writes most of your application for you. When you use a framework, your job is to write the parts of the application that make it do the specific things you want.
- When you set out to write a Rails application, leaving aside the configuration and other housekeeping chores, you have to perform three primary tasks −
- **Describe and model your application's domain** − The domain is the universe of your application. The domain may be a music store, a university, a dating service, an address book, or a hardware inventory. So here you have to figure out what's in it, what entities exist in this universe and how the items in it relate to each other. This is equivalent to modeling a database structure to keep the entities and their relationship.
- **Specify what can happen in this domain** − The domain model is static; you have to make it dynamic. Addresses can be added to an address book. Musical scores can be purchased from music stores. Users can log in to a dating service. Students can register for classes at a university. You need to identify all the possible scenarios or actions that the elements of your domain can participate in.

- **Choose and design the publicly available views of the domain** − At this point, you can start thinking in Web-browser terms. Once you've decided that your domain has students, and that they can register for classes, you can envision a welcome page, a registration page, and a confirmation page, etc. Each of these pages, or views, shows the user how things stand at a certain point.

Based on the above three tasks, Ruby on Rails deals with a Model/View/Controller (MVC) framework.

Ruby on Rails MVC Framework

The **M**odel **V**iew **C**ontroller principle divides the work of an application into three separate but closely cooperative subsystems.

Model (ActiveRecord)

– It maintains the relationship between the objects and the database and handles validation, association, transactions, and more.

– This subsystem is implemented in Active Record library, which provides an interface and binding between the tables in a relational database and the Ruby program code that manipulates database records. Ruby method names are automatically generated from the field names of database tables.

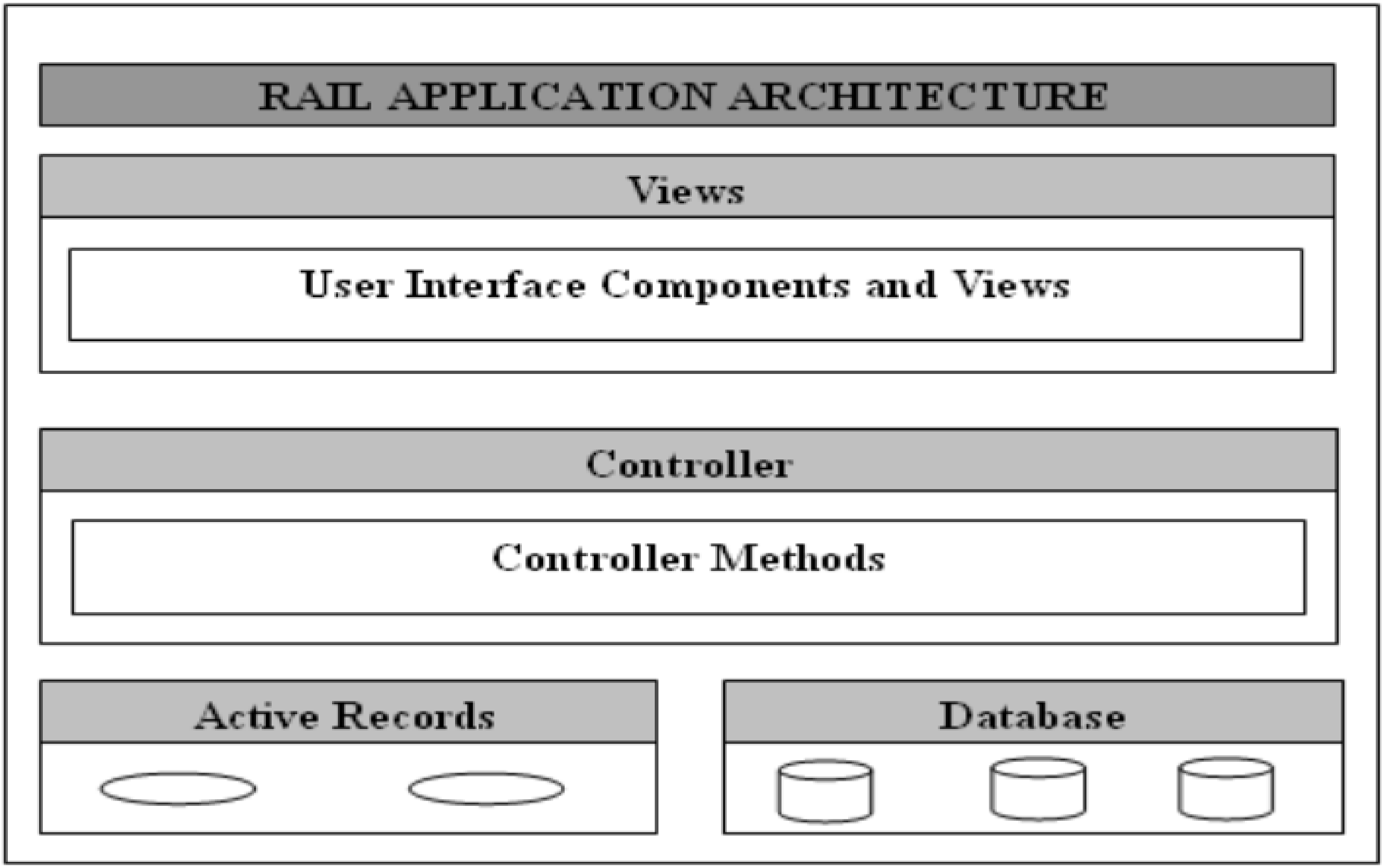
View (ActionView)

- It is a presentation of data in a particular format, triggered by a controller's decision to present the data. They are script-based template systems like JSP, ASP, PHP, and very easy to integrate with AJAX technology.
- This subsystem is implemented in Action View library, which is an Embedded Ruby (ERb) based system for defining presentation templates for data presentation. Every Web connection to a Rails application results in the displaying of a view.

Controller (ActionController)

- The facility within the application that directs traffic, on the one hand, querying the models for specific data, and on the other hand, organizing that data (searching, sorting, messaging it) into a form that fits the needs of a given view.
- This subsystem is implemented in ActionController, which is a data broker sitting between ActiveRecord (the database interface) and ActionView (the presentation engine).

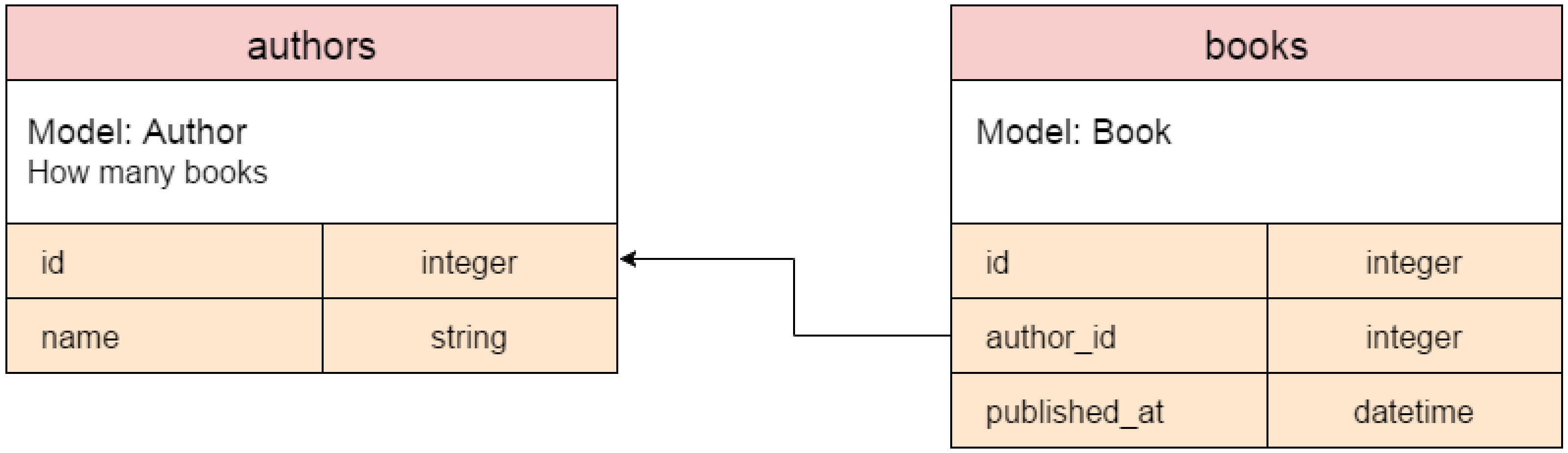
Pictorial Representation of MVC Framework


RAIL APPLICATION ARCHITECTURE
Views
User Interface Components and Views
Controller
Controller Methods
Active Records
Database

- When you use the Rails helper script to create your application, it creates the entire directory structure for the application. Rails knows where to find things it needs within this structure, so you don't have to provide any input.
- Here is a top-level view of a directory tree created by the helper script at the time of application creation. Except for minor changes between releases, every Rails project will have the same structure, with the same naming conventions. This consistency gives you a tremendous advantage; you can quickly move between Rails projects without relearning the project's organization.

- Active Record
  - Data structures are represented by a hierarchy of classes. Data is mostly stored in relational database tables. There is an essential mismatch between your program's object view and database's relational data view. To remove this mismatch, many attempts have been tried. One way to resolve this mismatch was through the use of Object-relational-mapping (ORM) tools. ORM is the mapping of relational database tables to object-oriented classes.
  - A perfect ORM hides the details of a database's relational data behind the object hierarchy. In Rails, ORM is implemented by Active Record which is one of the most important components of the Rails library.

- An ORM provides a mapping layer between how a database handles its data and how an object-oriented application handles its data. It maps database tables to classes, database table rows to objects, and database tables columns to object attributes. Active Record mainly carries out the mapping process for you. While using Active Record, you have to no longer deal with database constructs like tables, rows or columns. Your application only deals with classes, attributes and objects.



```
class Author < ActiveRecord: ;Base
has_many ;books
end
```

- Active Record is based on a design pattern created by Martin Fowler. From this design pattern only, the Active Record got its name. Its code works very well even with less number of lines. It is quite easy to use. Active Record Rails application does not need any configuration at all, if proper naming schemes is followed in your database and classes.
- There is one more feature of Active Record that makes your work easy, the implementation of a domain-specific-language (DSL). A DSL is a programming language intended to use in a specific problem domain. It allows you to use dynamically generated methods, like to retrieve a record, method find_by_first_name is used.

Active Record Basics

- Some of the basics of Active Record are classes, objects and naming conventions.
- **Active Record Classes and Objects**
  - Each table in a database is generally represented by a class that extends Active Record base class. By extending Active Record base classes, model objects inherit a lot of functionalities.
  - While using Active Records, you don't have to set up any database connections. It manages all the database connections for an appication. It adds attributes to your class for each of the columns in the database.
- **Active Record naming conventions**
  - Active Record uses the CoC (convention over configuration) principle. On following naming convention, you can take advantage of many dynamic features of Active Record without any configuration.

Class and Database

- Database table should be named in the plural form and in lowercase of your model classes. For example, if a model class name is **Student**, then corresponding table name will be **students**. With the help of this convention, Rails will automatically find the corresponding table to your model class without any configuration code. It even supports plural nouns such as 'people' as the plural of 'person'.
- Rails provides a facility where you can add plurals for a model. To define your own pluralization, add code to the config/environment.rb using **Inflector.**
- In some case, if you don't want to name your database in the plural form, Rails can be configured with singular-named database tables by adding following line to the config/environment.rb :

ActiveRecord::Base.pluralize_table_names = **false**

Ruby on Rails Migrations

- Migrations are a way to alter database schema over time in a consistent and organized manner. They use a Ruby DSL through which there is no need to write SQL by hand.
- SQL fragments can be edited by hand but then you have to tell other developers about the changes you made and then run them. You need to keep track of changes that need to be run against production machines next time you deploy.
- Each migration is a new version of the database. Each migration modifies database by adding or removing tables, columnns or entries. Active record will update your db/schema.rb file to match up-to-date structure of your database.

Purpose of Migrations

- It is important to know the purpose of migration before using it. Database is used in all web applications.
- Generally, a SQL statement is used to run database queries to create, modify, read or delete columns of a database.
- Migration file contains a specific set of instructions for how a database should be created. When this file is run, Rails will make changes in the database automatically. Gradually, the migration file will act as a versioned history of how database has changed. It implies that you will be able to recreate the database from the set of instructions file.

Creating Migration file

Syntax to create a migration file:

```
application_dir> rails generate migration table_name
```

- This will create a file with the name db/migrate/001_table_name.rb. A migration file contains basic data structure of a database table.
- It is advisable that before running the migration generator, clean the existing migrations generated by model generators.

Example:

- Let us create a migration called **java** in the application **tutorials.**

```
rails generate migration java
```

Editing Code

- Go to db/migrate directory in the tutorials application. Write the following code in the present file 001_java.rb,

```
class Java < ActiveRecord::Migration

  def self.up
    create_table :java do |t|
      t.column :title, :string, :limit => 32, :null => false
      t.column :fee, :float
      t.column :duration, :integer
      t.column :index, :string
      t.column :created_at, :timestamp
    end
  end

  def self.down
    drop_table :java
  end
end
```

- The method **self.up** is used during migrating to a new version and **self.down** is used to roll back any changes if needed.

Run Migration

- After creating all the required migration files you need to execute them. To execute migration file against database, run the following code:

```
rake db:migrate
```

- It will create a "schema_info" table if doesn't exist. It tracks the current version of the database.
- If new migration will be created then that will be a new version for the database.

Rails Layout

- In Rails, layouts are pieces that fit together (for example header, footer, menus, etc) to make a complete view. An application may have as many layouts as you want. Rails use convention over configuration to automatically pair up layouts with respective controllers having same name.
- Rails layouts basically work on Don't Repeat Yourself principle (DRY).
- In Rails, layouts are enabled by default. Whenever you generate a new Rails application, a layout is automatically generated for you in app/views/layouts.
- First we need to define a layout template and then define the path for controller to know that layout exists.

Creating Responses

- There are three ways to create an HTTP response from the controller's point of view:
- Call render to create a full response to send back to the browser
- Call redirect_to to send an HTTP redirect status code to the browser
- Call head to create a response to end back to the browser
- Importance of yield statement
- The yield statement in Rails decides where to render the content for the action in layout. If there is no yield statement in the layout, the layout file itself will be rendered but additional content into the action templates will not be correctly placed within the layout.
- Hence, a yield statement is necessary to add in a layout file.

  <%= **yield** %>

Relation between Rails Layouts and Templates

- When a request is made in an application, following process occur:
- First of all, Rails find a template for corresponding action to render method in your controllers action.
- Then finds correct layout to use.
- It uses action template to generate a content specific to the action.
- Finally it looks for the layout's yield statement and insert action's template here.

Finding correct layout

- Rails searches for the layout with same name in the app/layouts directory as the controllers name.
- For example, if you have a controller called **GioController**, then rails will search for **layouts/gio.html.erb** layout. It no layout with the same name is present, then it will use the default layout **app/views/layouts/appplication.html.erb**

Rails Filters

- Rails filters are methods that run before or after a controller's action method is executed. They are helpful when you want to ensure that a given block of code runs with whatever action method is called.

- Rails support three types of filter methods:
  - Before filters
  - After filters
  - Around filters

Before Filters

- Rails before filters are executed before the code in action controller is executed. The before filters are defined at the top of a controller class that calls them. To set it up, you need to call before_filter method.

```
class UserController < ApplicationController
before_filter :verify_password
def verify_password
...
end
end
```

In this example, method verify_password is applied as a before filter. Before any action method will be called, verify_password method is called.

After Filters

- Rails after filters are executed after the code in action controller is executed. Just like before filters, after filters are also defined at the top of a controller class that calls them. To set it up, you need to call after_filter method.

```
class PhotoController < ApplicationController
after_filter :resize_photo
def resize_photo
...
end
end
```

In this example, method resize_photo is applied as an after filter.

Around Filters

- Rails around filters contain codes that are executed both before and after controller's code is executed. They are generally used when you need both before and after filter. Its implementation is little bit different and more complex than other two filters. It is generally defined by a common class which contains both before and after methods.

```
class ActionLogger
def before(controller)
@start_time = Time.new
end
def after(controller)
@end_time = Time.now
@elapsed_time = @end_time.to_f -
 @start_time.to_f
@action = controller.action_name
# next save this logging detail to a file or datab
ase
table
end
end
```

In the ActionLogger class, before method captures time an action is started and after method captures time an action completes, the elapsed time.

Let us see how ActionLogger class works as an around filter. In your controller class, simply add around_filter method and pass an instance of ActionLogger as a parameter.

```
class PhotoController < ApplicationController
around_filter ActionLogger.new
end
```

Protecting Filter Methods

- Any method in your controller class can be routed from a browser. It is done through its ability to protect methods within a class. All Ruby methods have one of these protection levels.
- **Public:** These methods are accessible from any external class or method that uses the same class in which they are defined.
- **Protected:** These methods are accessible only within the class in which they are defined and in the classes that inherit from the class in which they are defined.
- **Private:** These methods are only accessible within the class in which they are defined.
- By default, methods are always public. Means any external class or method can access them. To define protection level, you can declare methods by putting a protected or private keyword before the methods that you want to protect.

```
class User
def new_user
...
end
protected
def sign_in
...
end
private
def user_identity
...
end
def assign_sidekick
...
end
end
```

- In the above example, there is one protected method and two private methods. In the User class, the user_identity method is made a private method. Only other methods within User class can call it. The sign_in method can be called only by methods within User class or classes that are inherited from User class.

Testing in Rails

- Rails test is very simple to write and run for your application. As Rails script generates models and controllers, in the same way test files are also generated. Rails also uses a separate database for testing. Test database in an application is rebuilt each time the application's test run, and hence you always have a consistent database when your tests are run.
- Rails uses Ruby Test::Unit testing library. Rails application test is usually run using Rake utility.

What is AJAX?

- AJAX is an acronym for **A**synchronous **J**avaScript **A**nd **X**ML.
- AJAX is not a programming language. but simply a development technique for creating interactive web applications.
- The technology uses JavaScript to send and receive data between a web browser and a web server.
- The AJAX technique makes web pages more responsive by exchanging data with a server behind the scenes, instead of reloading an entire web page each time a user makes a change.
- With AJAX, web applications can be faster, more interactive, and more user friendly.
- AJAX uses an XMLHttpRequest object to send data to a web server, and XML is commonly used as the format for receiving server data, although any format including and plain text can be used.
- AJAX allows web pages to be updated asynchronously by exchanging data with a web server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

Introduction to AJAX

- Conventional HTML + JavaScript
  - Functions of JavaScript:
    - (Form) checking
    - Event handling
    - **Dynamic presentation and/or content creation**
      - Where is the data on which dynamic creation is based
        - » Values of variables that are part of JavaScript code.
        - » Values of HTML/form elements
- AJAX: HTML + JavaScript + XML
  - **Dynamic presentation and/or content creation**
    - Where is the data on which dynamic creation is based
      - Values of variables that are part of JavaScript code.
      - Values of HTML/form elements
      - **Get from the server by the JavaScript code.**
        - » **Using standard HTTP Get/Post Request/Response protocol**
      - **The returned data in XML**
        - » **Data only without presentation –** JavaScript code has built in presentation
        - » **Data + presentation with (inline CSS, XSLT, HTML …)**
  - **Bottom line**
    - **No data sent to the client browser more than once from the server**
    - **One page with multiple server accesses** (compared to one page one access)
    - Extreme case: One AJAX page for interactions of the entire web application

How to AJAX

- Create XMLHttpRequest object
  - IE

    ```
    xmlHttp=new ActiveXObject("Microsoft.XMLHTTP" )
    ```
  - Mozilla

    ```
    xmlHttp=new XMLHttpRequest()
    ```
- Define response handler function

```
function responseHandler() {
        if (xmlHttp.readyState==4 || xmlHttp.readyState=="complete") {
                textResponse =xmlHttp.responseText ;
                xmlResponse = xmlHttp.responseXML }
}
```

- Binding XMLHttpRequest handler function to XMLHttpRequest object
  - IE

    ```
    xmlHttp.onreadystatechange=responseHandler
    ```
  - Mozilla

    ```
    xmlHttp.onload=responseHandler
    ```
- Connect to the server

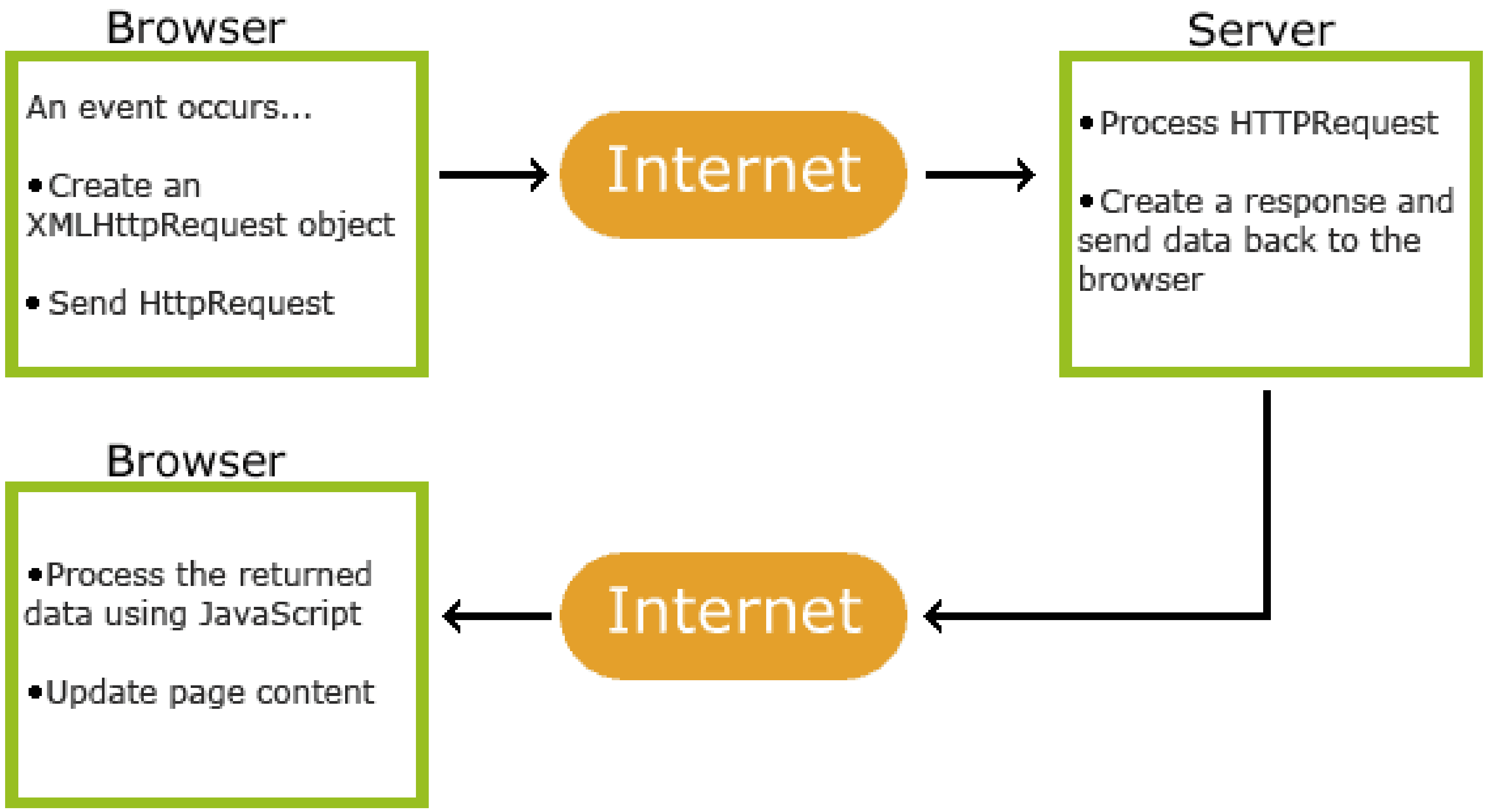
  ```
  xmlHttp.open(method, url, asyncFlag, userid, password)
  ```
- Send request to the server

  ```
  xmlHttp.send(requestMessageBody)
  ```

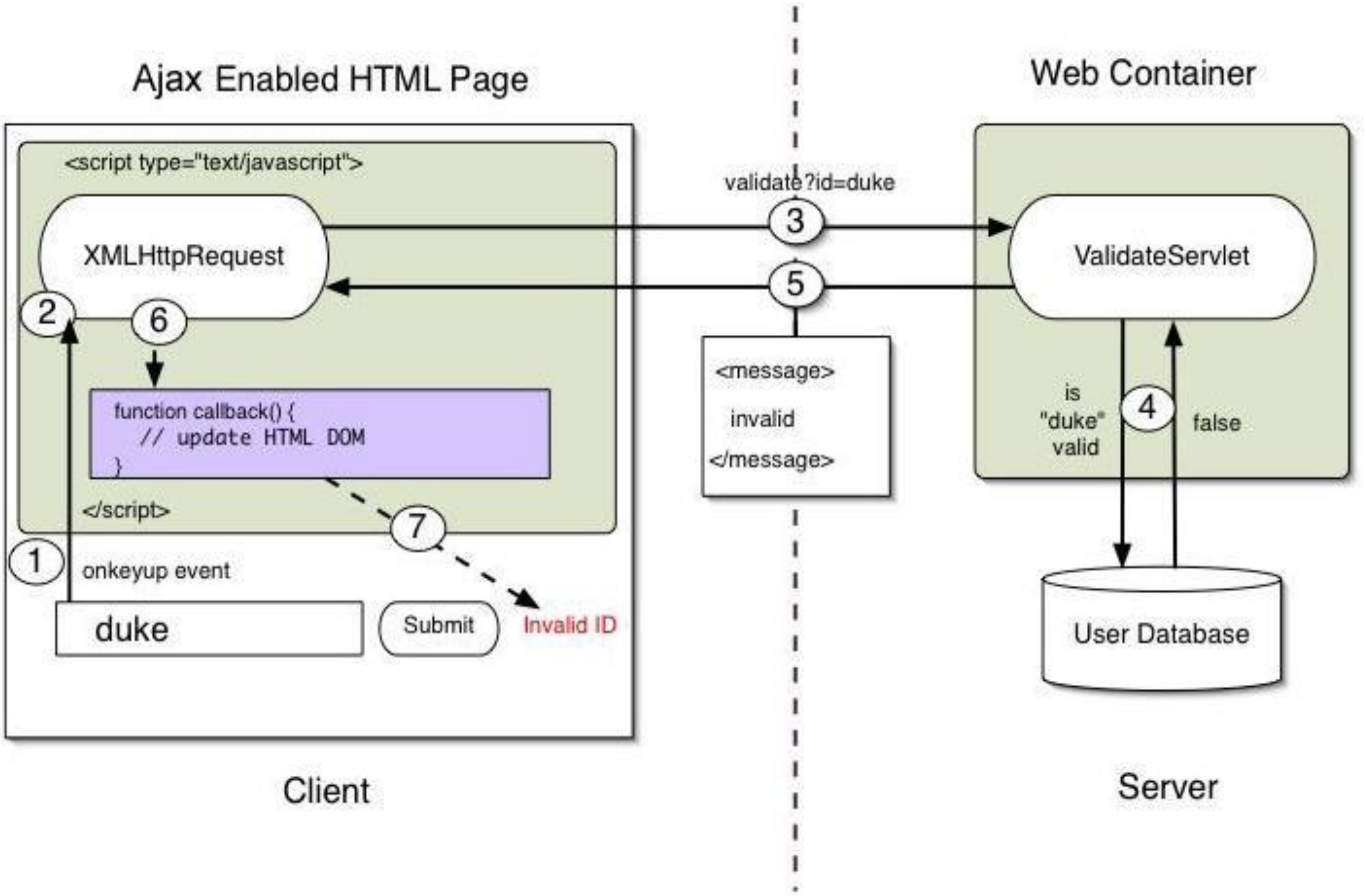
AJAX Application Example

- Online test
  - Many multiple choice questions
  - All questions are displayed on one page
  - After the user answers one question, the correct answer and explanation about why the user answer is wrong is shown on the page
  - For all already-answered questions, their correct answers and explanations are always shown on the page
- Pure sever-side solution using conventional web application
  - For each question answer submission, the whole page with most of repeated data sent to the browser
- Pure client-side solution using conventional JavaScript
  - The user can read JavaScript source code to view what is correct answer
  - Large amount of explanation data will be carried by the JavaScript code
- AJAX solution
  - After the user answers a question, use XmlHttpRequest to ask the server to send the correct answer and explanation.
  - Display the correct answer and explanation received from the server

How AJAX Works


Browser
An event occurs...
•Create an XMLHttpRequest object
•Send HttpRequest
Internet
Server
•Process HTTPRequest
•Create a response and send data back to the browser
Browser
•Process the returned data using JavaScript
•Update page content
Internet

1. An event occurs in a web page (the page is loaded, a button is clicked)
2. An XMLHttpRequest object is created by JavaScript
3. The XMLHttpRequest object sends a request to a web server
4. The server processes the request
5. The server sends a response back to the web page
6. The response is read by JavaScript
7. Proper action (like page update) is performed by JavaScript



```
function callback() {
  // update HTML DOM
}
```

Figure 1: An Ajax Interaction Provides Validation Logic

Based on Internet Standards

- XHTML/HTML and CSS
  - To display the data
- JavaScript (XMLHttpRequest calls)
  - To exchange data asynchronously with the server
- XML
  - To tranfer the data
- DOM (document object model)
  - To navigate the hierarchy of X/HTML elements

X/HTML and CSS

- Elements are identified in the X/HTML code that will be worked with by the JavaScript
- These provide the visual layout and structure for how the XML data will be displayed (performed on the client machine)
- CSS provides the styling

JavaScript/XMLHttpRequest

- Provides connection between the X/HTML element(s) and how they behave
- Sends XMLHttpRequests on demand when the user creates events
- Handles events both from the user and the replies from the server
- Can parse XML data and map it to data objects needed in the JavaScript
- Updates the X/HTML elements as needed

XML

- Provides format for data transfer between the JavaScript and the server

DOM

- Two DOMs involved
  - One for the elements in the X/HTML
  - One for the elements in the XML file
- Defines the logical structure of the documents
- Can be used by any programming language
- Used for navigating around the tree structure
- Provides quick access for changing/modifying elements

XMLHttpRequest

- Object used for fetching/returning data
- Can be synchronous or asynchronous—AJAX uses it asynchronously
- Allows the web pages to get more data from the server incrementally and asynchronously while the user is doing other things
- Examples are Gmail, which continuously asks the server for new mail and Google Maps, which update only the new parts of a map when the user mouses or clicks on a new point

Advantages

- Interactivity
  - Asynchronous transmission of data back and forth
- Bandwidth usage
  - Smaller payload
- Encourages modularization
  - Function, data sources, structure and style
- Allows non-related technologies to work together (server-side languages, databases, client-side languages, etc.)

Disadvantages

- Difficult to debug because it is asynchronous
- Search engines can’t index/optimize
- Browsers handle XHRs differently—Internet Explorer didn’t have a native one till version 7 (presently on version 8)
- Back button and bookmarks may not work as expected
- May experience response time/latency problems if there are many frequent updates

Uses for AJAX

- Real-time form data validation when server-side information is required
- Autocompletion (again when server-side info from a database, for example, is needed)
- Sophisticated user interface controls and effects such as progress bars
- Getting current data without reloading a full page

AJAX - The XMLHttpRequest Object

The keystone of AJAX is the XMLHttpRequest object.

The XMLHttpRequest Object

- All modern browsers support the XMLHttpRequest object.
- The XMLHttpRequest object can be used to exchange data with a web server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

Create an XMLHttpRequest Object

- All modern browsers (Chrome, Firefox, IE7+, Edge, Safari, Opera) have a built-in XMLHttpRequest object.
- Syntax for creating an XMLHttpRequest object:

  variable = new XMLHttpRequest();

Example

```
if (window.XMLHttpRequest) {
  // code for modern browsers
  xmlhttp = new XMLHttpRequest();
 } else {
  // code for old IE browsers
  xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
}
```

XMLHttpRequest Object Methods

Method	**Description**
new XMLHttpRequest()	Creates a new XMLHttpRequest object
abort()	Cancels the current request
getAllResponseHeaders()	Returns header information
getResponseHeader()	Returns specific header information
open(*method, url, async, user, psw*)	Specifies the request *method*: the request type GET or POST *url*: the file location *async*: true (asynchronous) or false (synchronous) *user*: optional user name *psw*: optional password
send()	Sends the request to the server Used for GET requests
send(*string*)	Sends the request to the server. Used for POST requests
setRequestHeader()	Adds a label/value pair to the header to be sent

XMLHttpRequest Object Properties

Property	Description
onreadystatechange	Defines a function to be called when the readyState property changes
readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
responseText	Returns the response data as a string
responseXML	Returns the response data as XML data
status	Returns the status-number of a request 200: "OK" 403: "Forbidden" 404: "Not Found" For a complete list go to the Http Messages Reference
statusText	Returns the status-text (e.g. "OK" or "Not Found")

AJAX - Send a Request To a Server

The XMLHttpRequest object is used to exchange data with a server.

<u>Send a Request To a Server</u>

- To send a request to a server, we use the open() and send() methods of the XMLHttpRequest object:

```
xhttp.open("GET", "ajax_info.txt", true);
xhttp.send();
```

AJAX - Server Response

- The onreadystatechange Property
- The readyState property holds the status of the XMLHttpRequest.
- The onreadystatechange property defines a function to be executed when the readyState changes.
- The status property and the statusText property holds the status of the XMLHttpRequest object.

AJAX XML Example

AJAX can be used for interactive communication with an XML file.

Example Explained

When a user clicks on the "Get CD info" button above, the loadDoc() function is executed.
The loadDoc() function creates an XMLHttpRequest object, adds the function to be executed when the server response is ready, and sends the request off to the server.
When the server response is ready, an HTML table is built, nodes (elements) are extracted from the XML file, and it finally updates the element "demo" with the HTML table filled with XML data:

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

```
function myFunction(xml) {
  var i;
  var xmlDoc = xml.responseXML;
  var table="<tr><th>Artist</th><th>Title</th></tr>";
  var x = xmlDoc.getElementsByTagName("CD");
  for (i = 0; i <x.length; i++) {
    table += "<tr><td>" +
    x[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue
 +
    "</td><td>" +
    x[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue
+
    "</td></tr>";
  }
  document.getElementById("demo").innerHTML = table;
}
```

AJAX PHP Example

```
<html> <body>
<p><b>Start typing a name in the input field below:</b></p>
<p>Suggestions: <span id="txtHint"></span></p>
        <form>
        First name: <input type="text" onkeyup="showHint(this.value)">
        </form>
                <script>
                function showHint(str) {
                if (str.length == 0) {
                document.getElementById("txtHint").innerHTML = "";
                return;  } else {
                var xmlhttp = new XMLHttpRequest();
                xmlhttp.onreadystatechange = function() {
                if (this.readyState == 4 && this.status == 200) {

        document.getElementById("txtHint").innerHTML = this.responseText;
                }   };
                xmlhttp.open("GET", "gethint.php?q=" + str, true);
                xmlhttp.send();
                 } }
                </script>
</body> </html>
```

AJAX Database Example

AJAX can be used for interactive communication with a database.

Example Explained - The showCustomer() Function
When a user selects a customer in the dropdown list above, a function called showCustomer() is executed. The function is triggered by the onchange event:
showCustomer

```
function showCustomer(str) {
 var xhttp;
 if (str == "") {
  document.getElementById("txtHint").innerHTML = "";
  return;
 }
 xhttp = new XMLHttpRequest();
 xhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
  document.getElementById("txtHint").innerHTML = this.responseText;
  }
 };
 xhttp.open("GET", "getcustomer.php?q="+str, true);
 xhttp.send();
}
```

The showCustomer() function does the following:
•Check if a customer is selected   Create an XMLHttpRequest object   Create the function to be executed when the server response is ready   Send the request off to a file on the server   Notice that a parameter (q) is added to the URL (with the content of the dropdown list)

Rails on Ajax

- AJAX stands for Asynchronous Javascript and XML. It is a mixture of several technologies and is an important part of Rails application. It allows client side changes without reloading the page.
- Let us see the working of a normal web server. On typing a web address and clicking on search, the browser makes a request to the server. To assemble the searched page, it fetches all associated assets like JavaScript files, images and stylesheets. On clicking a link, same process is followed. This is called 'request response cycle'.
- JavaScript makes request to the server, and parse the response. It can update information on the page. On combining these two abilities, a web page can be made with JavaScript that can update just a part of itself, without loading full page from the server. This technique is called AJAX.

By default Rails ships with CoffeeScript.
Let us see an example code to make Ajax request using the jQuery library

```
$.ajax(url: "/test").done (html) ->
$("#results").append html
```

The above code fetches data from "/test", then appends the result to the div with an id of results.

Unobtrusive JavaScript

To handle attached JavaScript to the DOM, Rails uses "Unobtrusive JavaScript" technique. This is considered as the best technique within the frontend community.

This is called 'Unobtrusive' JavaScript because we do not mix JavaScript code into HTML. With this, we can easily add behavior to any link by adding data attribute. Lot of benefits add up like the entire JavaScript is served on every page, it means it'll get downloaded on the first page load and then can be cached on every page after that.

Example

Let us see an example of performing Ajax on delete action.

Step 1 Create an application named **item.**

```
rails new item
```

Step 2 Write the following command.

```
rails generate scaffold Itemm state:string country:string
```

Step 3 Write migrate command.

```
rake db:migrate
```

Step 4 Update your destroy action in aap/views/itemms/index.html.erb file by writing the following code:

```
:remote => true, :class => 'delete_itemm'


<tbody>
  <% @itemms.each do |itemm| %>
   <tr>
    <td><%= itemm.state %></td>
    <td><%= itemm.capital %></td>
    <td><%= link_to 'Show', itemm %></td>
    <td><%= link_to 'Edit', edit_itemm_path(itemm) %></td>
    <td>
<%= link_to 'Destroy', itemm, method: :delete, data: { confirm: 'Are you sure?' }, :remote => true, :class => 'dele
</td>
   </tr>
  <% end %>
 </tbody>
```

Step 5 Create app/views/itemms/destroy.js.erb file.

```
$('.delete_itemm').bind('ajax:success', function() {
  $(this).closest('tr').fadeOut();
});
```

Step 6 Go to controller file at app/controllers/itemms_controller.rb and write the following code.

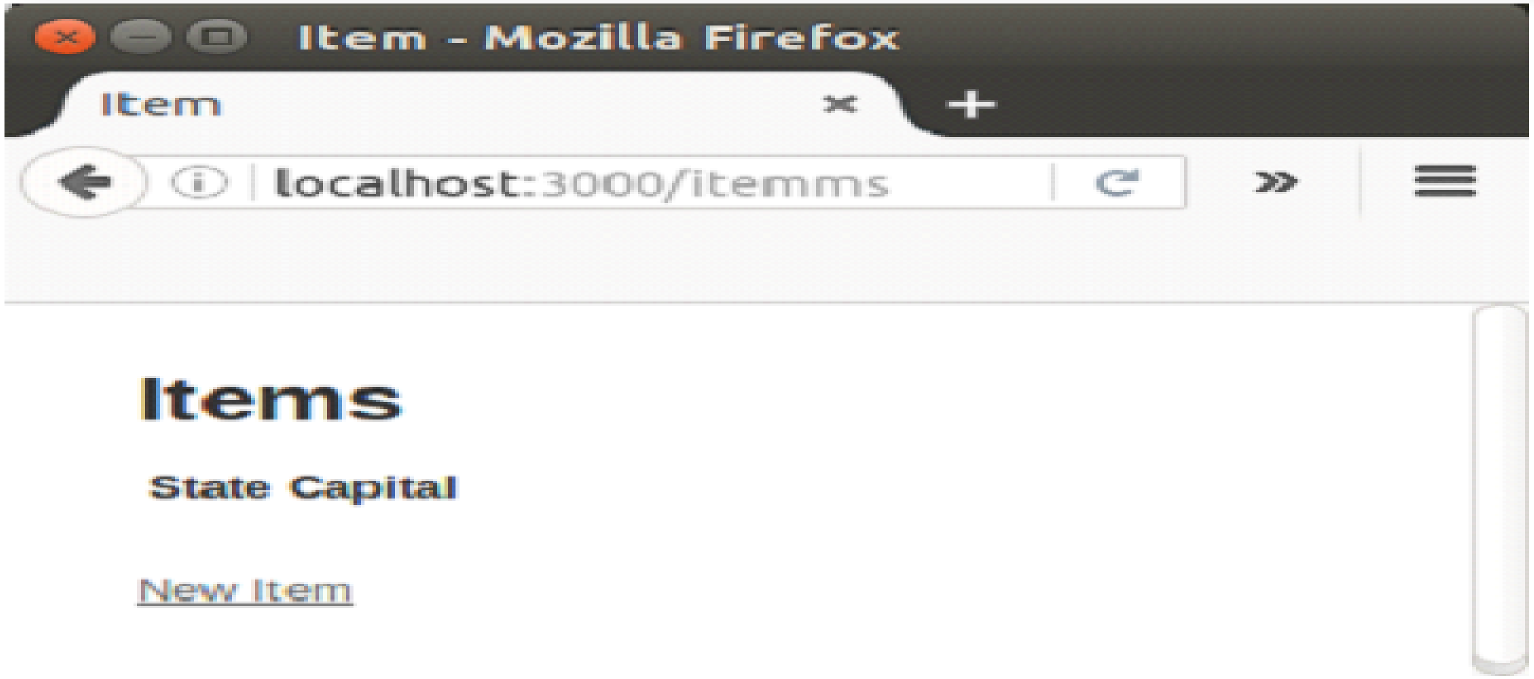
```
 def destroy
   @itemm = Itemm.find(params[:id])
   @itemm.destroy

   respond_to do |format|
     format.html { redirect_to item_url }
     format.json { head :no_content }
     format.js   { render :layout => false }
   end

end
```

Step 7 Start the Rails server.

```
rails s
```

Step 8 Run it on localhost.

```
http://localhost:3000/itemms
```

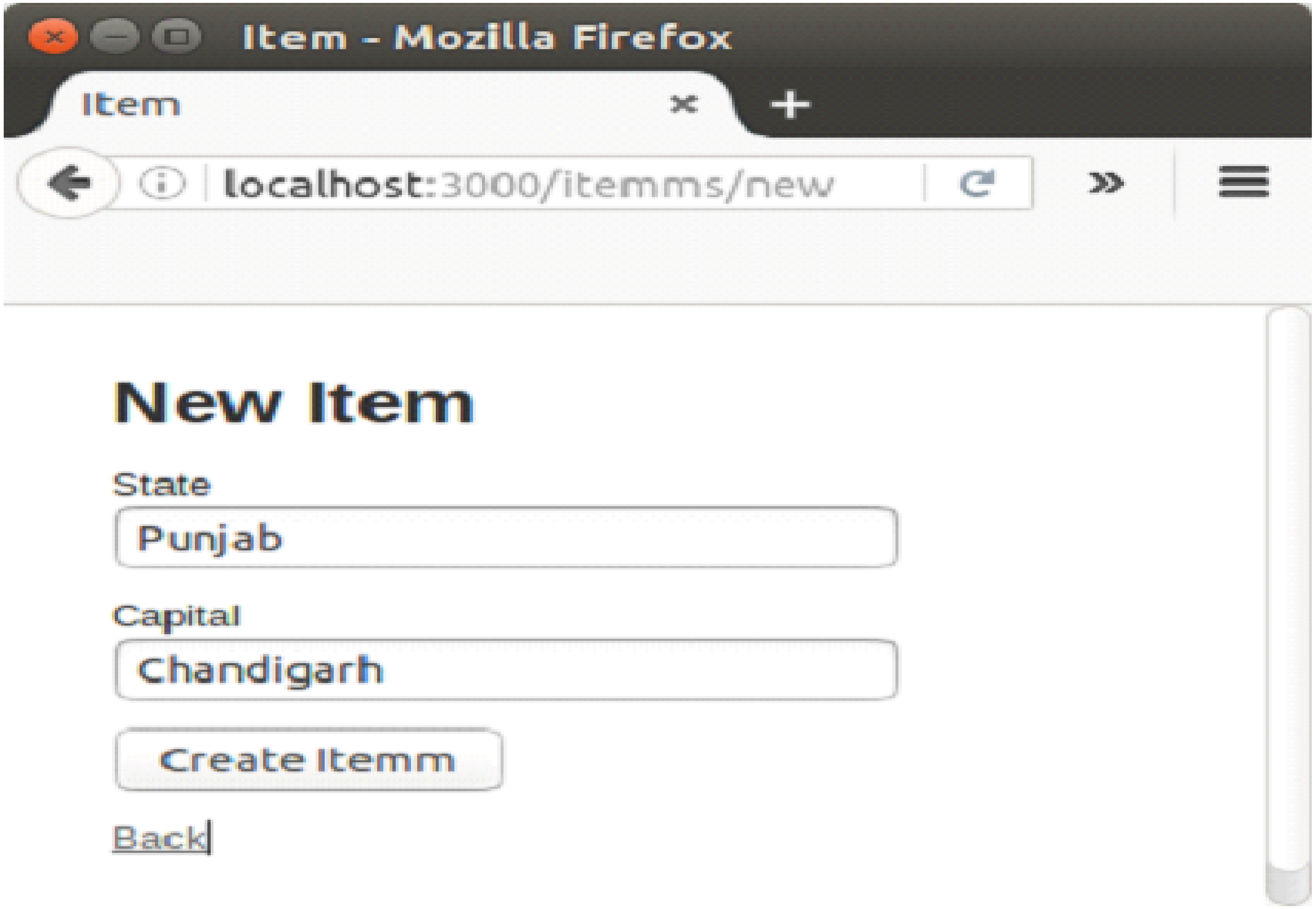


Items

State Capital

New Item

Create an item as shown in the following snapshot.


Item - Mozilla Firefox

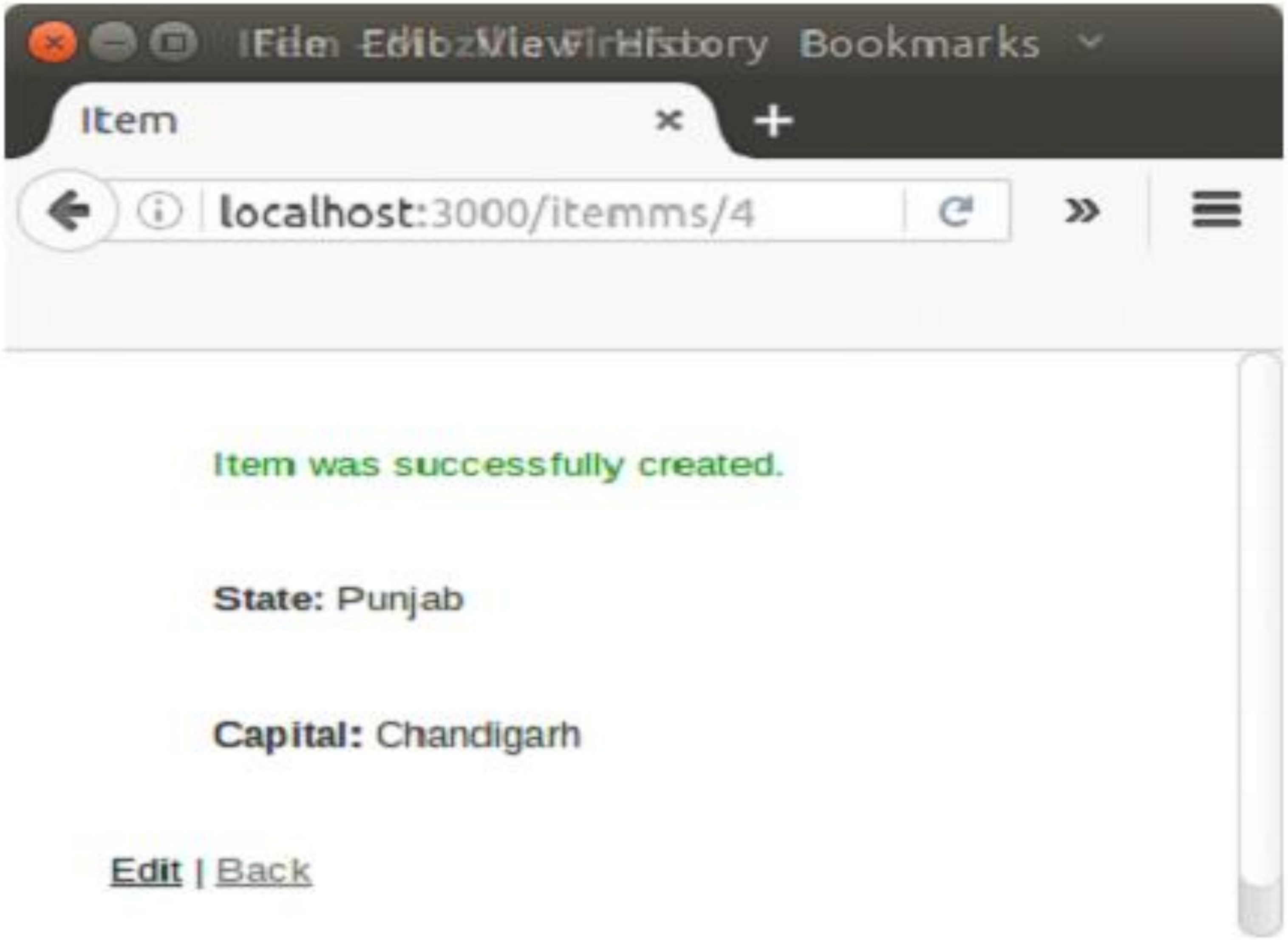
localhost:3000/itemms/new

New Item

State: Punjab

Capital: Chandigarh

It will create the item as shown below.



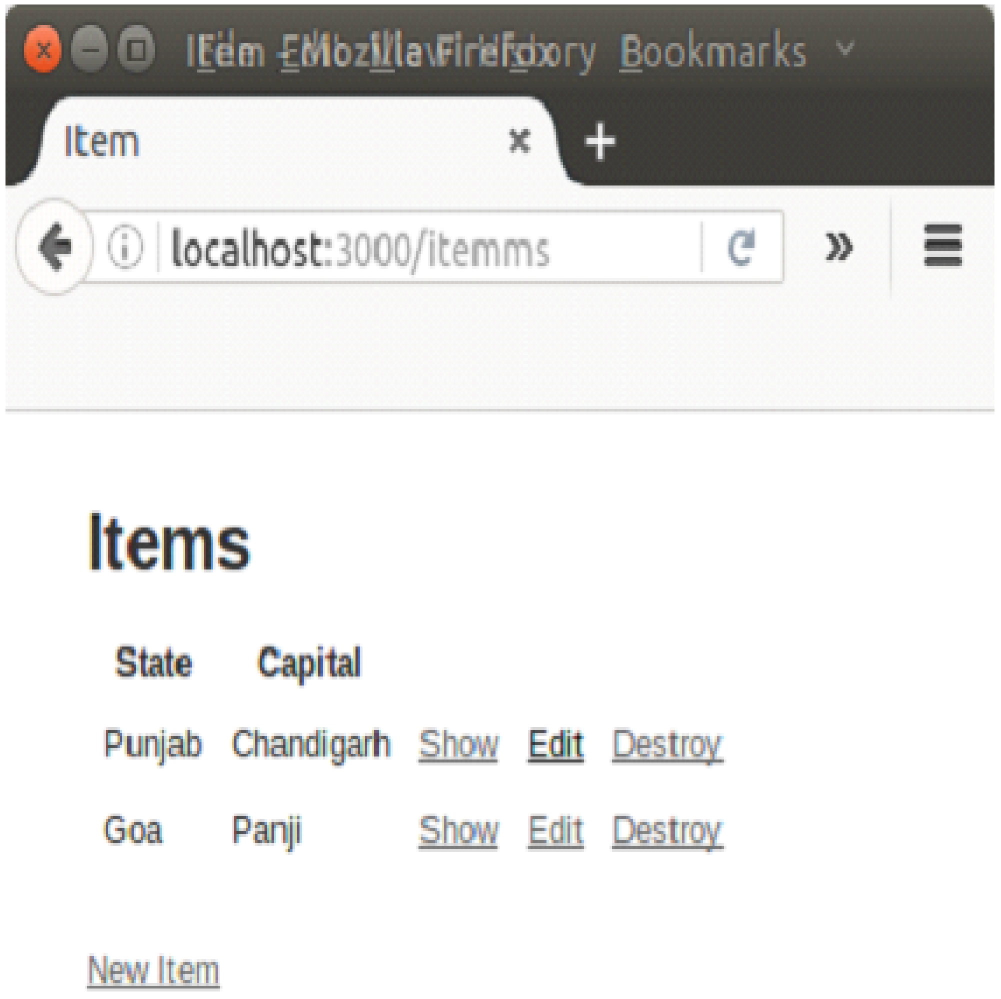
Item was successfully created.

State: Punjab

Capital: Chandigarh

Edit | Back

Click on Back button.

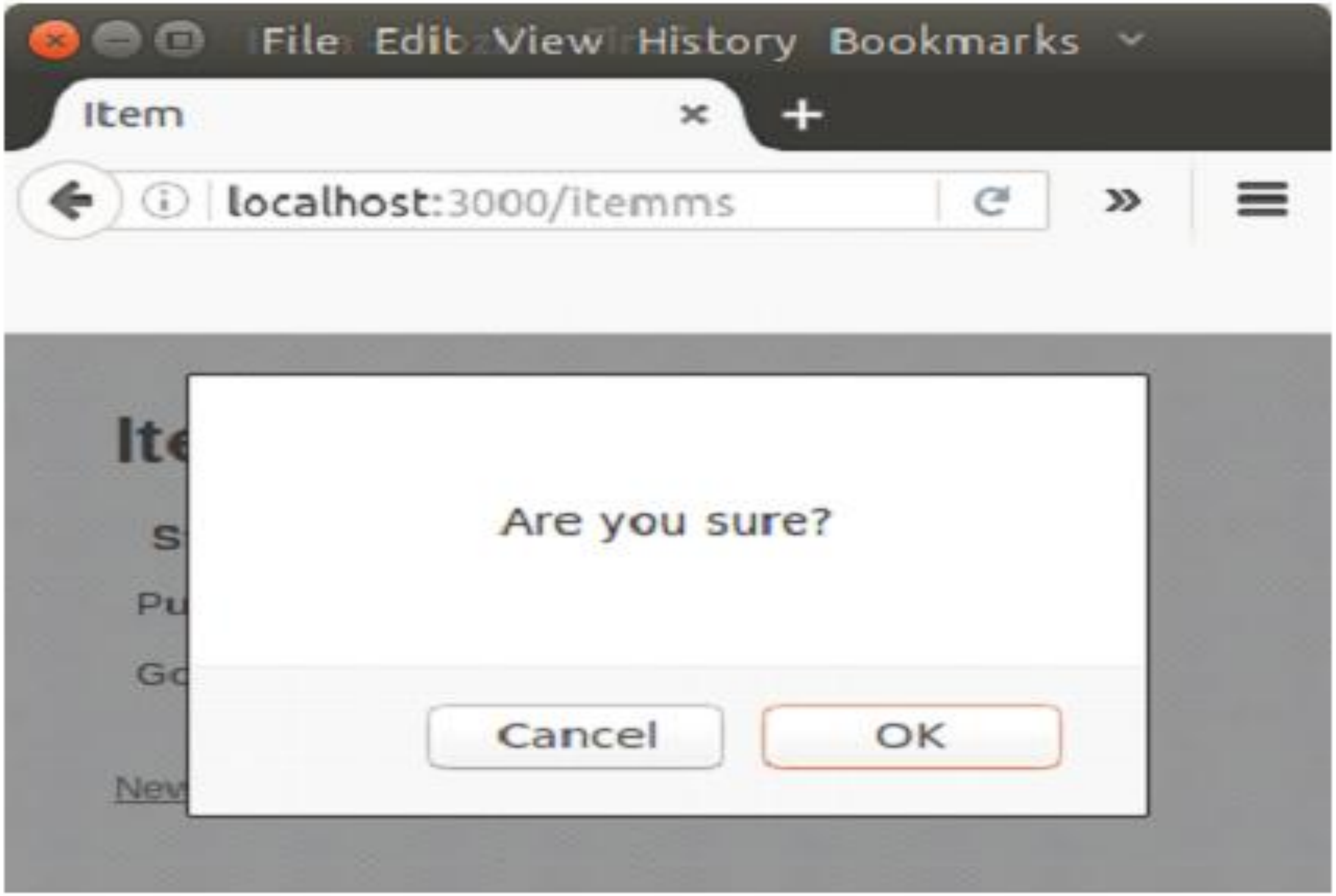


Items

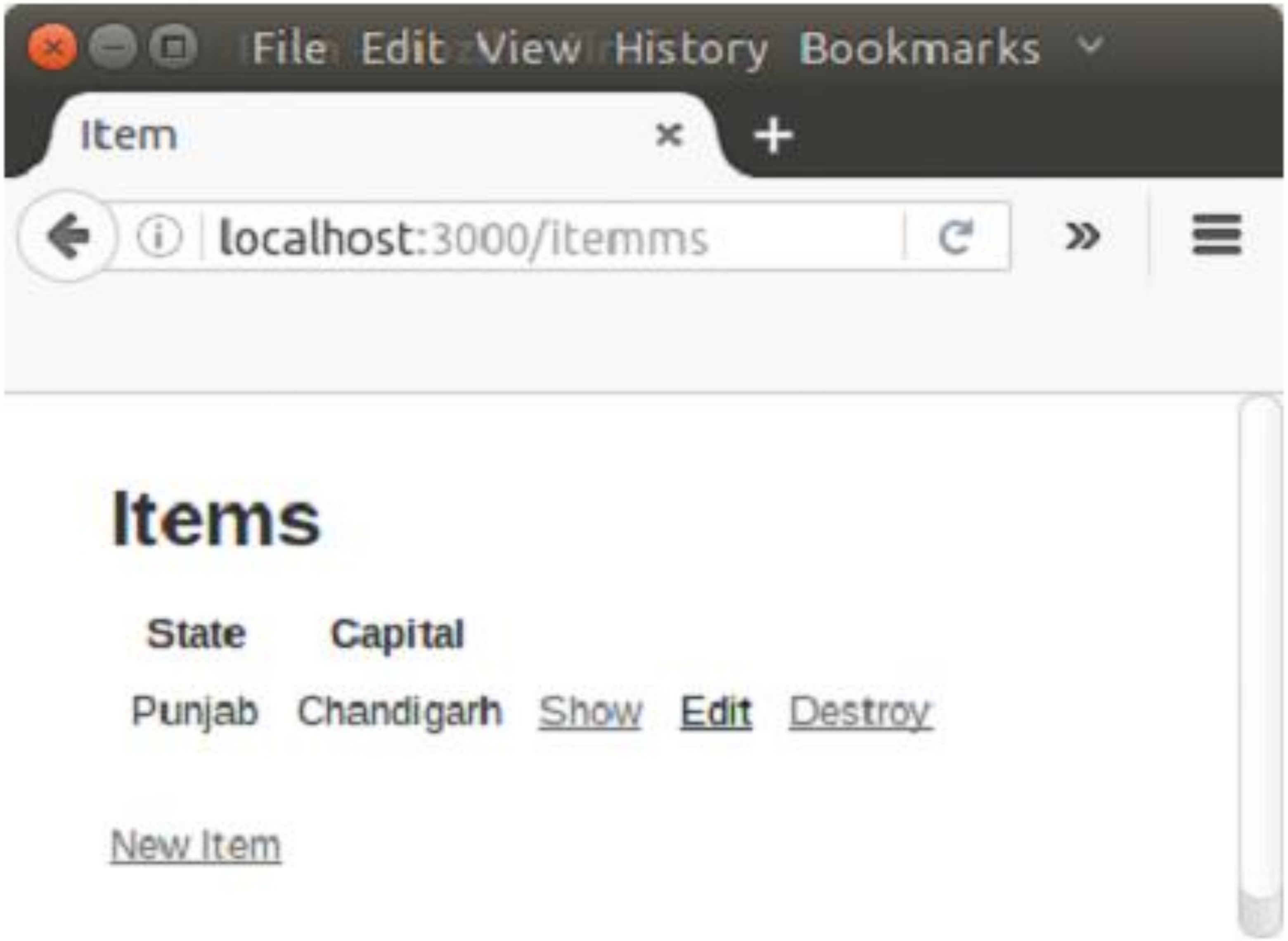
State	Capital			
Punjab	Chandigarh	Show	Edit	Destroy
Goa	Panji	Show	Edit	Destroy

New Item

If you will click on Destroy link, a popup will be shown through AJAX. It will destroy this item from the list.



Click OK to delete the item finally.



Items

State	Capital			
Punjab	Chandigarh	Show	Edit	Destroy

New Item