

Chapter 2:

Client Side Scripting: HTML and CSS

Introduction to HTML

- **HTML : Hyper Text Markup Language**
 - A browser understands and interpret the HTML tags, identifies the structure of the document (which part are which) and makes decision about presentation (how the parts look) of the document.
 - HTML also provides tags to make the document look attractive using graphics, font size and colors.
 - User can make a link to the other document by creating Hypertext Links also known as Hyperlinks.

Website Development Process

- A Web site is composed of individual pages that are linked together each of these relating to a different aspects of your site such as news , links and biography.
- It is a much like a SDLC (Software development Life Cycle).
 1. Requirements
 2. Design
 3. Write Code
 4. Test
 5. Upload
 6. Reiterate

Basic HTML

- A page can be loaded into a browser by two ways
 - By writing a URL in the address bar
 - By manually open a file
- The essential tags that are required to create a HTML document are:

<html>.....</html>

<head>.....</head>

<body>.....</body>

Basic HTML

- **HTML Tag <HTML>:**
 - The <HTML> tag encloses all other HTML tags and associated text within your document.

<HTML>

Your Title and Document (contains text with HTML tags) goes here

</HTML>

- The *slash mark* is always used in closing tags.

Basic HTML

- **HEAD Tag <HEAD>:**

- HEAD tag comes after the HTML start tag. It contains TITLE tag to give the document a title that displays on the browsers title bar at the top.

<HEAD>

<TITLE>

Your title goes here

</TITLE>

</HEAD>

Basic HTML

- **BODY Tag <BODY>:**

- The BODY tag contains all the text and graphics of the document with all the HTML tags that are used for control and formatting of the page.

<BODY>

Your Document goes here

</BODY>

- An HTML document, web page can be created using a **text editor, Notepad or WordPad**. All the HTML documents should have the extension .htm or .html

Basic HTML

- **Example:**

```
<!DOCTYPE html>
```

```
<HTML>
```

```
<HEAD>
```

```
<TITLE>
```

My first Page

```
</TITLE>
```

```
</HEAD>
```

```
<BODY>
```

WELCOME TO MY FIRST WEB PAGE

```
</BODY>
```

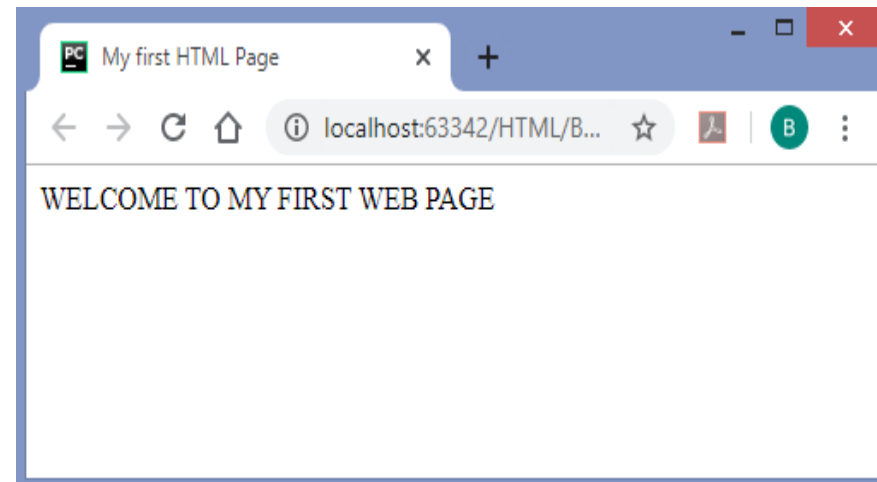
```
</HTML>
```



Basic HTML

- **Example:**

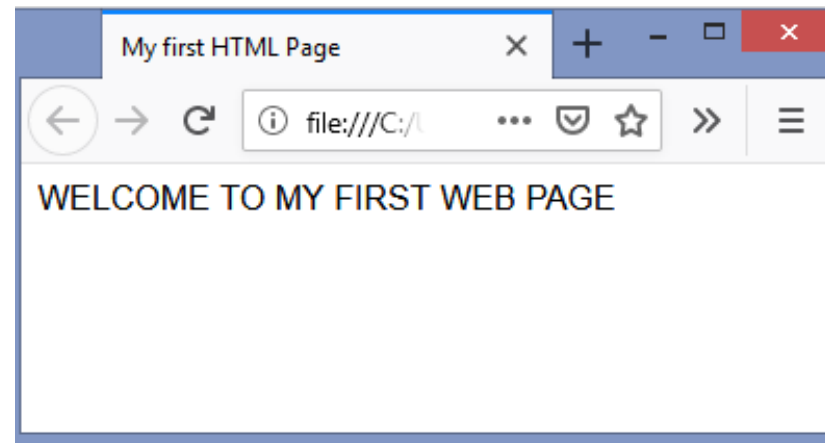
```
<!DOCTYPE html>
<HTML>
  <HEAD>
    <TITLE>
      My first Page
    </TITLE>
  </HEAD>
  <BODY>
    WELCOME TO MY FIRST
    WEB PAGE
  </BODY>
</HTML>
```



Basic HTML

- **Example:**

```
<!DOCTYPE html>
<HTML>
  <HEAD>
    <TITLE>
      My first Page
    </TITLE>
  </HEAD>
  <BODY>
    WELCOME TO MY FIRST
    WEB PAGE
  </BODY>
</HTML>
```



Basic HTML

- **Attributes used with <BODY> :**

- **BGCOLOR:** used to set the background color for the document

`<BODY BGCOLOR="yellow">`

Your document text goes here.

`</BODY>`

- **TEXT:** used to set the color of the text of the document

`<BODY TEXT="red">`

Document text changed to red color

`</BODY>`

- **BACKGROUND:** It is used to point to an image file (the files with an extension .gif, .jpeg) that will be used as the background of the document.

`<BODY BACKGROUND="filename.jpeg"> <`

Document background changed

`/BODY>`

Basic HTML

- **Example:**

```
<!DOCTYPE html>
```

```
<HTML>
```

```
<HEAD>
```

```
<TITLE>
```

My first Page

```
</TITLE>
```

```
</HEAD>
```

```
<BODY BGCOLOR="yellow", TEXT = "red">
```

Your document text goes here.

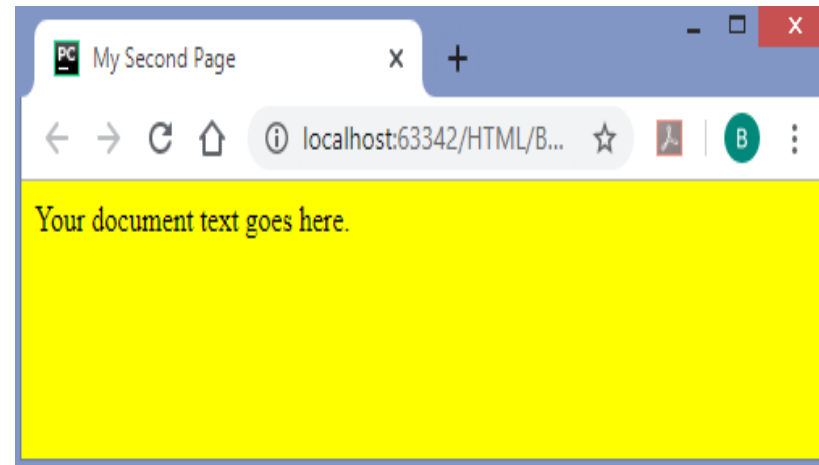
```
</BODY>
```

```
</HTML>
```

Basic HTML

- **Example:**

```
<!DOCTYPE html>
<HTML>
  <HEAD>
    <TITLE>
      My Second Page
    </TITLE>
  </HEAD>
  <BODY BGCOLOR="yellow">
    Your document
    text goes here.
  </BODY>
</HTML>
```



Basic HTML

- **Example:**

```
<!DOCTYPE html>
```

```
<HTML>
```

```
<HEAD>
```

```
<TITLE>
```

My Second Page

```
</TITLE>
```

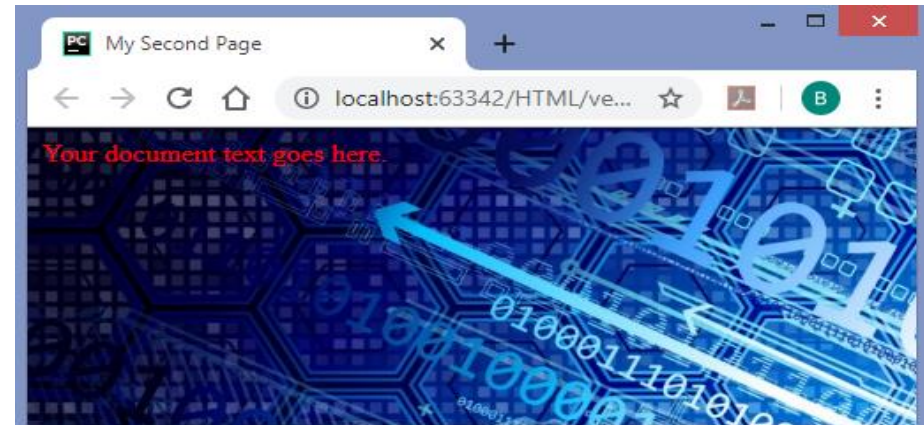
```
</HEAD>
```

```
<BODY TEXT="red", BACKGROUND="first.jpg">
```

Your document text goes here.

```
</BODY>
```

```
</HTML>
```



Basic HTML

- **Attributes used with <BODY> :**

- **MARGINS:** Set the left hand/right hand margin of the document

`<BODY LEFTMARGIN="60" TOPMARGIN="60" >`

This document is indented **60 pixels** from the left hand side and also from top side of the page.

`</BODY>`

- **ALINK:** Specifies the color of an active link in a document

`<BODY ALINK="red">`

Document active link color changed to red

`</BODY>`

- **LINK:** Specifies the color of an unvisited link in a document

`<BODY ALINK="blue">`

Document unvisited link color changed to blue

`</BODY>`

Basic HTML

- **Attributes used with <BODY> :**
 - **VLINK:** Specifies the color of an visited link in a document
 <BODY VLINK="yellow">
 Document visited link color changed to yellow
 </BODY>

Basic HTML

- `<!DOCTYPE html>`

`<html>`

`<head>`

`<title>My Second Page </title>`

`</head>`

`<BODY leftmargin="60", topmargin="60" alink="red", link="black",
vlink="green">`

`First`

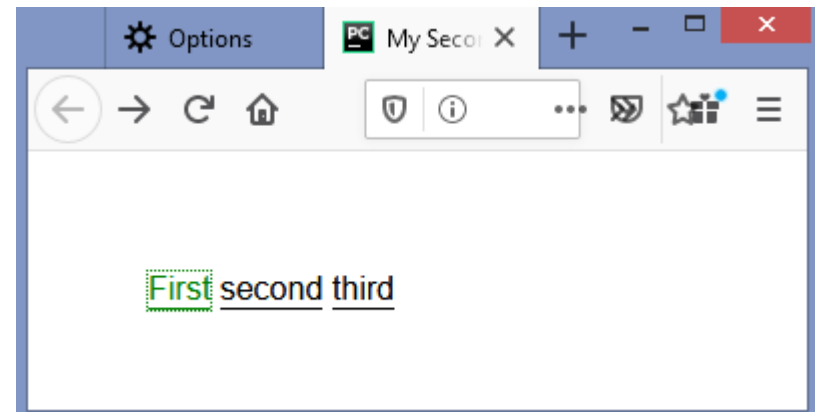
`second`

`third`

`</body>`

`</html>`

[Unit1_1.html](#)



Basic HTML

- **Container and Empty Tags :**
 - Container Tags : Tags which have both the opening and closing i.e. **<TAG> and </TAG>** are called container tags.
 - Empty Tags: Tags, which have only opening and no ending, are called empty tags. Line break **
 or
 and <HR />** tags are empty tags.

Basic HTML

- **Formatting web page :**

- HTML has six header tags **<H1>**, **<H2>**.....**<H6>** used to specify section headings. Text with header tags is displayed in larger and bolder fonts than the normal body text by a web browser.
- Every header leaves a blank line above and below it when displayed in browser.

```
<HTML>
```

```
  <HEAD>
```

```
    <TITLE>
```

```
      My first Page
```

```
    </TITLE>
```

```
  </HEAD>
```

```
  <BODY>
```

```
    <b1> Welcome to my first web page </b1>
```

```
  </BODY>
```

```
</HTML>
```

Basic HTML

- **Formatting web page :**

- Browsers ignore extra space within HTML document.
- For Example: You can have text “**Hello GCET**” in HTML document but in browser it display, “**Hello GCET**”.
- **Paragraph tag <P>:**
 - » This tag <P> indicates a paragraph, used to separate two paragraphs with a blank line.
`<P> Welcome to the world of HTML </P>`
- **Line Break Tag
:**
 - » The empty tag
 is used, where the text needs to start from a new line and not continue on the same line. To get every sentence on a new line, it is necessary to use a line break.

Example:

G. H. Patel College of Engineering and Technology

Near Bakrol Gate

Vallabh Vidhyanagar-388120

Basic HTML

- **Formatting web page :**

- **Preformatted Text Tag <PRE>:**

- » <PRE> tag can be used, where it requires total control over spacing and line breaks. Browser preserves your space and line break in the text written inside the tag. It uses courier font.

<PRE> Welcome to the world of HTML </PRE>

- **Horizontal Rule Tag <HR>:**

- » An empty tag <HR> basically used to draw lines and horizontal rules. It can be used to separate two sections of text. It accept SIZE, COLOR, ALIGN attribute.
 - » **Example:**

<BODY>

Your horizontal rule goes here. **<HR>**

The rest of the text goes here.

</BODY>

Basic HTML

- **Character Formatting Tags:**
 - The character formatting tags are used to specify how a particular text should be displayed on the screen to distinguish certain characters within the document.
 - Boldface ****
 - Italics **<I>**
 - Subscript **<SUB>**: displays text in Subscript
 - Superscript **<SUP>**: displays text in Superscript
 - Small **<SMALL>**: displays text in smaller font as compared to normal font
 - Big **<BIG>**: displays text in larger font as compared to normal font

Basic HTML

- ** Tag:**
 - Attributes of are:
 - » **COLOR:** Sets the color of the text that will appear on the screen. It can be set by giving the value as #rr0000 for red or by name.
 - » **SIZE:** Sets the size of the text, takes value between 1 and 7, default is 3.
 - » **FACE:** Sets the normal font type, provided it is installed on the user's machine. GCET

Basic HTML

Font Size	Heading	Point Size
7	-	36pt
6	<h1>	24pt
5	<h2>	18pt
-	<h3>	14pt
4	<h4>	12pt bold
3	Body text	12pt plain
-	<h5>	10pt
-	<h6>	7pt
2	-	9pt

Basic HTML

- **List in Web Page:**
 - HTML Supports several ways of arranging items in lists. The most commonly used are:
 - » Ordered List (Numbered List)
 - » Unordered List (Bulleted List)
 - **Ordered List **
 - » Ordered list also called as Numbered list, is used to present a numbered list of item in the order of importance or the item (paragraph) is marked with a number.
 - » An ordered list must begin with the followed by an list item tag.

Basic HTML

- **List in Web Page:**
 - **Ordered List attributes:**
 - » **START** : Used for lists that need to start at values other than 1.
 - » START always specified in default numbers, and is completed based on TYPE before display, For example, If START =5 it would display either an 'E', 'e', 'V', 'v', or '5' based on TYPE attribute.
 - » **TYPE** : allows marking list items with different types. By default the list item markers are set to numbers 1,2,3... so on.

<OL type="i">

i : Roman numerals i, ii, iii, iv.....

I : Roman Capitals I, II, III, IV.....

a : Lowercase letter a, ,b, c,

A : Uppercase letters A, B, C,

Basic HTML

- **List in Web Page:**
 - **Unordered List :**
 - » Unordered List also called as bulleted list, used to present list of items marked with bullets. An unordered list starts with in followed by (List Item) tag. Use of is very similar to (ordered list).
 - » **Example:**

```
<UL>  
  
    <LI> Brinjal  
    <LI> Cabbage  
    <LI> Tomato  
  
</UL>
```

Output:

- Brinjal
- Cabbage
- Tomato

Basic HTML

- **List in Web Page:**
 - **Unordered List attributes:**
 - » **TYPE:** allows marking list items with different types. By default the list item markers are set to bullets.
 - » Other values are circle and square.
 - » **Example:**

```
<UL type="circle">  
    <LI> GCET  
    <LI> ADIT  
    <LI> BVM  
</UL>
```

 - GCET
 - ADIT
 - BVM

Basic HTML

- **Graphics in Web Page :**
 - Images can be placed in a web page by using tag.

Syntax:

Where **SRC** – Source of the image file

image_URL – represents the image file with its location.

Example:

Basic HTML

- **Links in Web Page <A>:**

- Web pages are linked to one another through Hypertext Links.
- Section of text or image in the HTML document can be linked to an external document or to a specific place within the same document.

Example:

** Click here **

** Click here **

Basic HTML

- **Linking to a specific place within the same Document :**
 - It is required to jump different sections in the same document. For this it needs two steps, first; identify section with a name and or second; use jumps to the location using the name used.
 - ** link to another section of the same document **
 - » This link text jumps to the section named with HREF on click.
 - » The # symbol before the section name is must.
 - ** Beginning of the section **
 - » The NAME attribute is used to identify a section with a name. It is a unique identifier within the document for the anchor.

Basic HTML

- **Linking to a specific place within the same Document :**
 - Same folder, but different document:
 - ****
 - Different folder, different document:
 - ****
 - A different server:
 - ****

Basic HTML

- **Tables <TABLE>:**

- A table is divided into rows <tr> and each row is divided into data cells <td>.
- A data cell can contain text, images, lists, paragraphs, forms, horizontal rules, tables, etc.

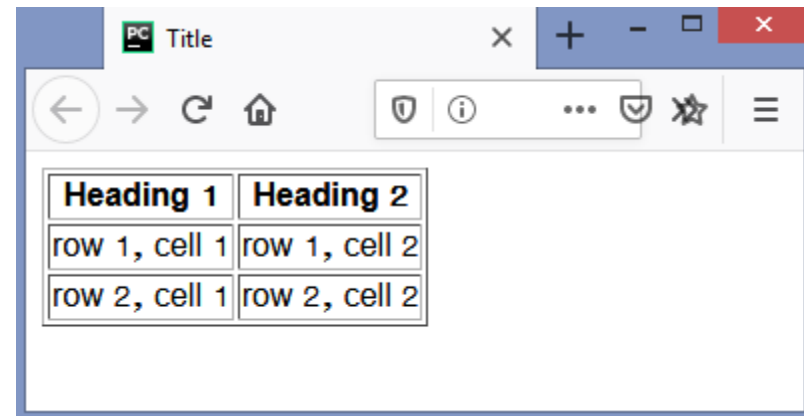
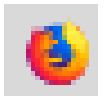
Example : <table border="1">

```
    <th>
        <td> Heading  1</td>
        <td> Heading  2</td>
    </th>
    <tr>
        <td>row 1, cell 1</td>
        <td>row 1, cell 2</td>
    </tr>
    <tr>
        <td>row 2, cell 1</td>
        <td>row 2, cell 2</td>
    </tr>
</table>
```

Basic HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
</head>
<body>
  <table border="1">
    <tr>
      <th>Heading 1</th>
      <th>Heading 2</th>
    </tr>
    <tr>
      <td>row 1, cell 1</td>
      <td>row 1, cell 2</td>
    </tr>
    <tr>
      <td>row 2, cell 1</td>
      <td>row 2, cell 2</td>
    </tr>
  </table>
</body>
</html>
```

[Unit1_table.html](#)



Basic HTML

- **Tables <TABLE> attributes:**
 - **Border:** Which sets the border width in pixels around the table.
 - **Width:** Which in this case is as a percentage of the screen
 - **Cellpadding:** to give the distance in pixels between the inner border and the text
 - **Cellspacing:** Which sets the spacing in pixels between the inner and outer borders.
 - **Rowspan:** The rowspan attribute specifies the number of rows a cell should span.
 - **Colspan:** The colspan attribute specifies the number of columns a cell should span.

Basic HTML

- **Forms in Web Page <FORM>:**
 - HTML forms provide a simple and reliable user interface to collect data from the user and transmit the data to a server-side program for processing.
 - The server simply reads all the HTTP data sent to it by the browser, then returns a Web page to the client.
 - Each of the controls typically has a name and a value, where the name is specified in the HTML and the value comes either from user input or from a default value in the HTML.

Basic HTML

- **Forms in Web Page <FORM>:**
 - **<form action="processform.php" name="f1" method="POST">**
 - The action attributes tells the HTML where to send the collected information.
 - The method attribute describes the way to send it.
 - **There are two types of Method:**
 - GET
 - POST

Basic HTML

- **Forms in Web Page <FORM>:**
 - **GET Method:**
 - » An HTTP GET request, appends the form data to the end of the specified URL after a question mark.
 - » GET is the default and is also the method that is used when the user types a URL into the address bar or clicks on a hypertext link.
 - Advantage of GET Method:**
 - » Save the results of a form submission: you can submit data and bookmark the resultant URL, send it to a other by email, or put it in a normal hypertext link. Google.com, yahoo.com etc.

Basic HTML

- **Forms in Web Page <FORM>:**

- **POST Method:**

- » HTTP POST, sends the data after the HTTP request headers and a blank line.

- Advantage of POST Method:**

- » When POST is used, the data is sent on a separate line.

- » POST is Secure than GET.

Basic HTML

- **<FORM> Elements:**

- **<INPUT>** tag is used to collect information from the user.

- **Different Types of Input:**

- » `<input type="text" name="fn" id="fn1" value="fname"/>`

- » `<input type="password"/>`

- » `<input type="radio"/>`

- » `<input type="checkbox"/>`

- » `<input type="file"/>`

- » `<input type="submit"/>`

Basic HTML

- **<FORM> Elements:**

- **Different Types of Input:**

- » `<input type="button"/>`

- » `<input type="image"/>`

- » `<input type="reset"/>`

- » `<input type="hidden"/>`: allows hidden data to be passed along with form. (not seen by user)

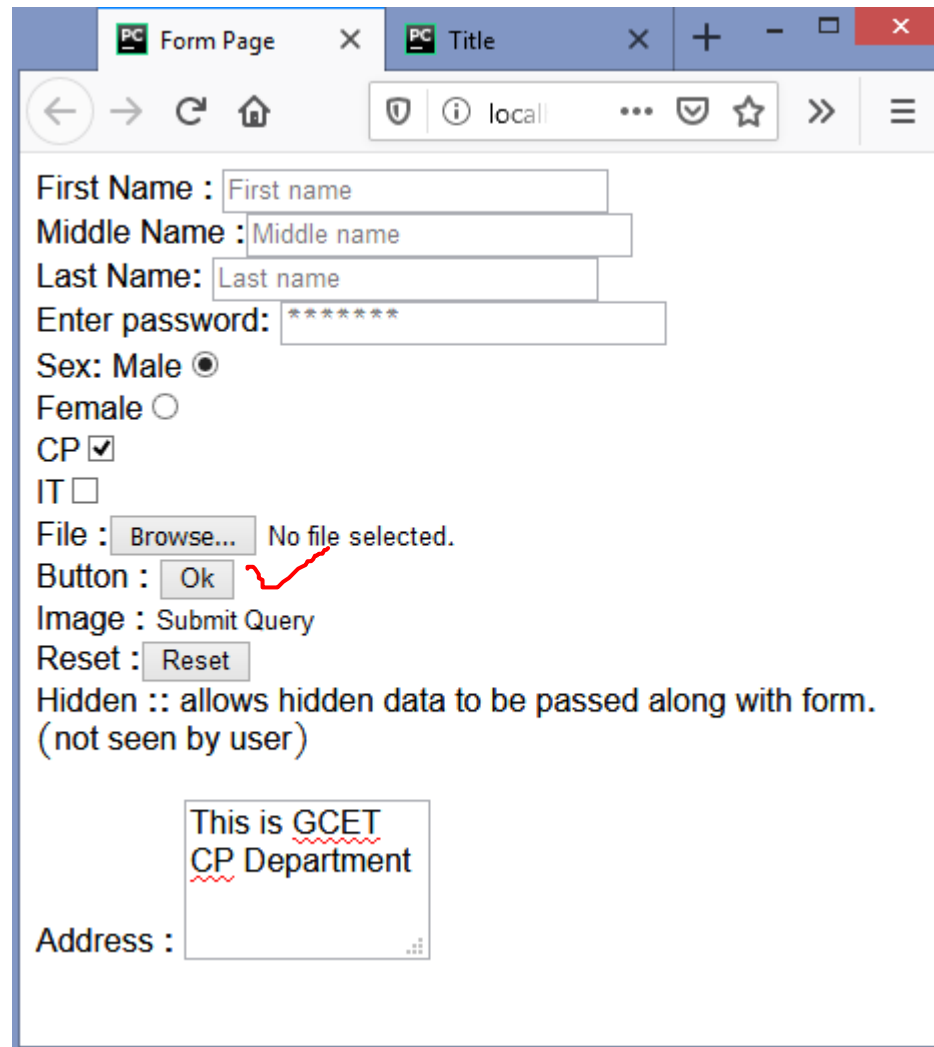
- » `<textarea name="add1" id="add11" rows="10" cols="10">`

Basic HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Form Page</title>
</head>
<body>
  <form method="get" action="#">
    First Name : <input type="text" name="fname" placeholder="First name" required><br>
    Middle Name :<input type="text" name="mname" placeholder="Middle name" ><br>
    Last Name: <input type="text" name="ln" placeholder="Last name" required><br>
    Enter password: <input type="password" name="pwd" placeholder="*****" required><br>
    Sex: Male<input type="radio" name="male" required checked><br>
    Female<input type="radio" name="male"><br>
    CP<input type="checkbox" as checked/><br>
    IT<input type="checkbox"/><br>
    File :<input type="file"/><br>
    Button : <input type="button" value="Ok"/><br>
    Image : <input type="image"/><br>
    Reset :<input type="reset"/><br>
    Hidden :<input type="hidden"/>: allows hidden data to be passed along with form. (not seen by user)
  <br><br>
  Address : <textarea name="add1" id="add11" rows="3" cols="8">This is GCET CP Department
</textarea> </form> </body> </html>
```

Basic HTML

[Unit1_form.html](#)



The screenshot shows a web browser window with two tabs: 'Form Page' and 'Title'. The address bar shows 'local'. The form contains the following elements:

- First Name :
- Middle Name :
- Last Name:
- Enter password:
- Sex: Male ☒ Female ☐
- CP ☒ IT ☐
- File : No file selected.
- Button : (marked with a red checkmark)
- Image :
- Reset :
- Hidden :: allows hidden data to be passed along with form.
(not seen by user)
- This is GCET
CP Department
- Address :

Basic HTML

- **<FORM> Elements:**

- **Drop Down Menu:**

- » A SELECT element presents a set of options to the user

<SELECT NAME="FavLang" >

<OPTION VALUE="C" Selected > C </OPTION>

<OPTION VALUE="C++" > C++ </OPTION>

<OPTION VALUE="JAVA"> Java </OPTION>

</SELECT>



Basic HTML

- **<FORM> Elements:**

- **Drop Down Menu:**

```
<SELECT NAME="FavLang" >
```

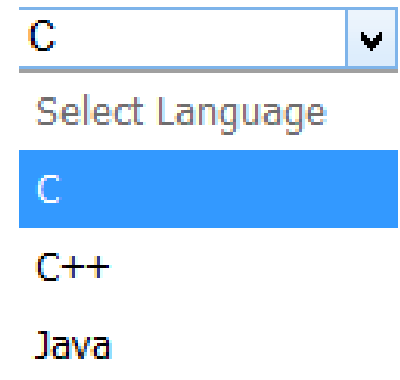
```
    <OPTION Selected disabled > Select Language </OPTION>
```

```
    <OPTION VALUE="C" > C </OPTION>
```

```
    <OPTION VALUE="C++" > C++ </OPTION>
```

```
    <OPTION VALUE="JAVA"> Java </OPTION>
```

```
</SELECT>
```



C

Select Language

C

C++

Java

Basic HTML

- **<FORM> Elements:**

- **List Box:**

- » List boxes are used when multiple selections are permitted, or a specific visible size has been specified.

- ```
<SELECT NAME="FavLang" multiple size="2">
```

- ```
    <OPTION VALUE="C" > C </OPTION>
```

- ```
 <OPTION VALUE="C++" > C++ </OPTION>
```

- ```
    <OPTION VALUE="JAVA"> Java </OPTION>
```

- ```
</SELECT>
```

# Basic HTML

- **Fieldset and Legend:**

- **Fieldset:** draws a rectangle around the area.
- **Legend:** gives heading to each input section.

- **Example:**

<fieldset>

<legend> Personal details</legend>

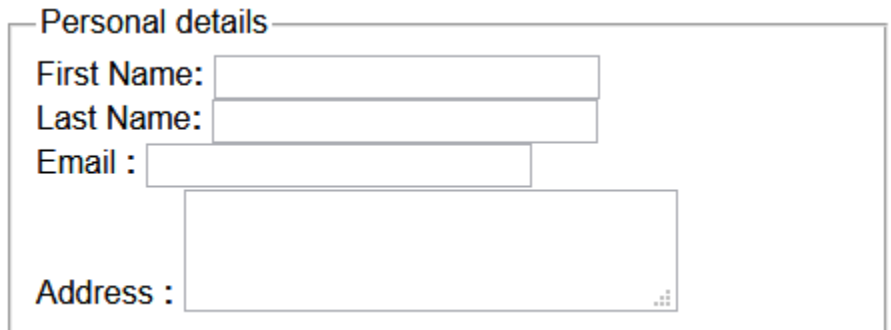
First Name: <input type=text />

Last Name: <input type=text />

Email : <input type=text />

Address : <textarea type=text />

</fieldset>



Personal details

First Name:

Last Name:

Email :

Address :

# Basic HTML

- **Web Site Structure :**
  - A Web site is collection of pages associated by hyperlinks , but it should be broken up into area for structure.
  - Your site is broken into several sub-sites.
  - A website has a root directory which is entered first , then several sub-directories that serve as the sub-sites.
- [www.mywebsite.co.in](http://www.mywebsite.co.in)
  - ⑩ index.html ( homepage)
  - ⑩ Department ( All department related files (pages))
  - ⑩ Images ( images and other files (pages))
  - ⑩ CSS (All CSS files)
  - ⑩ JS (All JavaScript files)



# Introduction to XHTML

- XHTML stands for Extensible Hypertext Markup Language
- XHTML is aimed to replace HTML
- XHTML is almost identical to HTML 4.01
- XHTML is stricter and cleaner version of HTML
- XML is a markup language designed for describing data
- XHTML is a Bridge between HTML and XML

# Introduction to XHTML

- 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**.
- Attribute names must be in **lower case**.
- Attribute values must be **quoted**.

# Introduction to XHTML

- **DOCTYPE** declaration of XHTML:

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

- This is what the **<html>** element looked like in XHTML:

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
 lang="en">
```

- Here is a typical XHTML **<head>** section:

```
<head>
 <meta http-equiv="Content-type" content="text/html;
 charset=UTF-8" />
 <title>My First XHTML Page</title>
 <link rel="stylesheet" type="text/css" href="style.css" />
</head>
```

# Introduction to XHTML

- XHTML elements must be properly nested

**`<b><i> bold and italic </b> </i>` is wrong**

**`<b><i> bold and italic </i> </b>` is correct**

- XHTML documents must be well-formed

`<html>`

`<head>.....</head>`

`<body>.....</body>`

`</html>`

- If an HTML tag is not a container, close it like this :

**`<br />`, `<hr />`, ``**

# Introduction to XHTML

- **META tag:**

- Metadata is information about data(information) in this context machine understandable information about web resources.
- It can be included in both HTML and XHTML to describe the actual documents rather than the document's content.
- `<meta>` tags always go inside the `<head>` element, and are typically used to specify character set, page description, keywords, author of the document, and viewport settings.
- Metadata will not be displayed on the page, but is machine parsable.
- Metadata is used by browsers (how to display content or reload page), search engines (keywords), and other web services.
- There is a method to let web designers take control over the viewport (the user's visible area of a web page), through the `<meta>` tag

# Introduction to XHTML

- **META tag:**

```
<html>
```

```
<head>
```

```
<title> My Web Page </title>
```

```
<meta name =“author” content=“Brijesh Patel”/>
```

```
<meta name =“description” content=“This page is
related to information of faculties of GCET”/>
```

```
</head>
```

```
</html>
```

# Introduction to XHTML

- **META tag:**

```
<html>
```

```
 <head>
```

```
 <title> My Web Page </title>
```

```
 <meta name =“expires” content=“Mon,20 Jul 2017
16:00:00 ”/>
```

```
 <meta http-equiv =“refresh” content=“50;abc1.html”/>
```

```
 </head>
```

```
</html>
```

# Introduction to XHTML

- **META tag:**

- The viewport is the user's visible area of a web page. It varies with the device - it will be smaller on a mobile phone than on a computer screen.

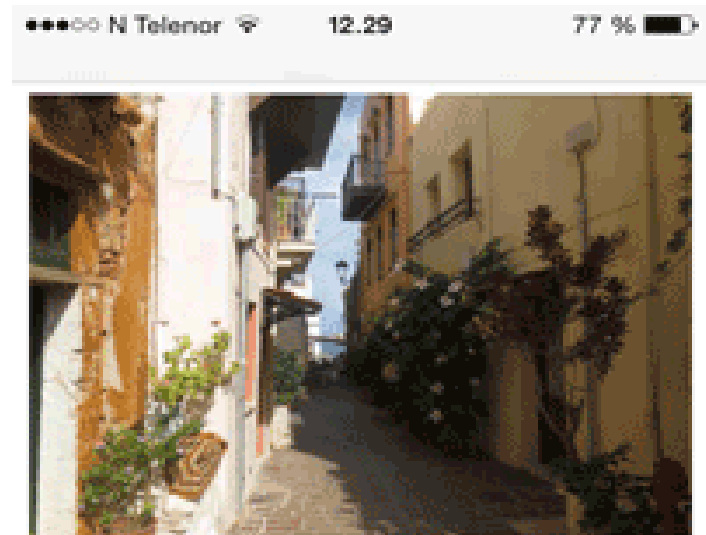
**<meta name="viewport" content="width=device-width, initial-scale=1.0">**

- This gives the browser **instructions** on how to **control** the page's **dimensions** and **scaling**.
- The **width=device-width** part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).
- The **initial-scale=1.0** part sets the initial **zoom** level when the page is first loaded by the browser.



# Introduction to XHTML

- META tag:**



Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue dui dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend orion congue nihil imperdiet domina

# Introduction to XHTML

- **META tag (Memory Cache):**
  - Web browsers can cache pages for quick reviewing without having to request them again and re-download the document.
  - Each page has a TTL ( Time To Leave) . Usually 30 days, when the browser cache has been cleared or the allotted memory is all used up.
  - Browser can be stopped from caching a page. If it supports the meta elements **http-equiv** attribute.
  - For this we have to set value **pragma** is assigned to **http-equiv** attribute and no-cache value to the content attribute.
  - **<meta http-equiv =“pragma” content=“no-cache”/>**

# Introduction to XHTML

- **Character Entities:**
  - &lt; , &gt; , &amp; , &divide; , &copy; , &reg; , &quot;.
- **Frameset:**
  - **<frame> and <noframes> tags:**
    - Dividing content on website when it is large, then use frames to break information up and partition it.
    - To Separate the different elements of your content like to divide the page into rows and to place a new page into frame.

# Introduction to XHTML

- **Frameset Example:**

```
<frameset frameborder="3" cols="25%, *, 25%" >
```

```
 <frame name="banner" src="banner.html" />
```

```
 <frame name="main" src="main.html" />
```

```
 <frame name="options" src="options.html" />
```

```
</frameset>
```

# Introduction to XHTML

## Frame A

**Note:** The frameset, frame, and noframes elements are not supported in HTML5.

## Frame B

## Frame C

# Introduction to XHTML

- **<noframe>:**
  - Only give a body section to such a program where you want to suppose browsers that don't have frame capability.

`<noframes>`

`<body>`

`<p> your browser does not support frame. </p>`

`</body>`

`</noframe>`

# Introduction to HTML5



- HTML5 is the newest version of HTML, only recently gaining partial support by the makers of web browsers.
- It incorporates all features from earlier versions of HTML, including the stricter XHTML.
- It is still a work in progress. No browsers have full HTML5 support. It will be many years – perhaps not until 2018 or later - before being fully defined and supported.
- New features are based on HTML, CSS, DOM, and JavaScript



# Goals of HTML5

- Support all existing web pages. With HTML5, there is no requirement to go back and revise older websites.
- Reduce the need for external plugins and scripts to show website content.
- Make the rendering of web content universal and independent of the device being used.
- Enhanced form controls and attributes
- Built-in audio and video support (without plugins)



# First Look at HTML5



- Remember the DOCTYPE declaration from XHTML?

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

- In HTML5, there is just one possible DOCTYPE declaration and it is simpler:

```
<!DOCTYPE html>
```

The DOCTYPE tells the browser which type and version of document to expect. This should be the last time the DOCTYPE is ever changed. From now on, all future versions of HTML will use this same simplified declaration.



# First Look at HTML5

- This is what the `<html>` element looked like in XHTML:

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
 lang="en">
```

- Again, HTML5 simplifies this line:

```
<html lang="en">
```

The `lang` attribute in the `<html>` element declares which language the page content is in. Though not strictly required, it should always be specified, as it can assist search engines.

Each of the world's major languages has a two-character code, e.g. Spanish = "es", French = "fr", German = "de", Chinese = "zh", Arabic = "ar".



# First Look at HTML5

- Here is a typical XHTML <head> section:

```
<head>
 <meta http-equiv="Content-type" content="text/html;
 charset=UTF-8" />
 <title>My First XHTML Page</title>
 <link rel="stylesheet" type="text/css" href="style.css" />
</head>
```

- And the HTML5 version:

```
<head>
 <meta charset="utf-8">
 <title>My First HTML5 Page</title>
 <link rel="stylesheet" href="style.css">
</head>
```

Notice the simplified character set declaration, the shorter CSS stylesheet link text, and the removal of the trailing slashes for these two lines.

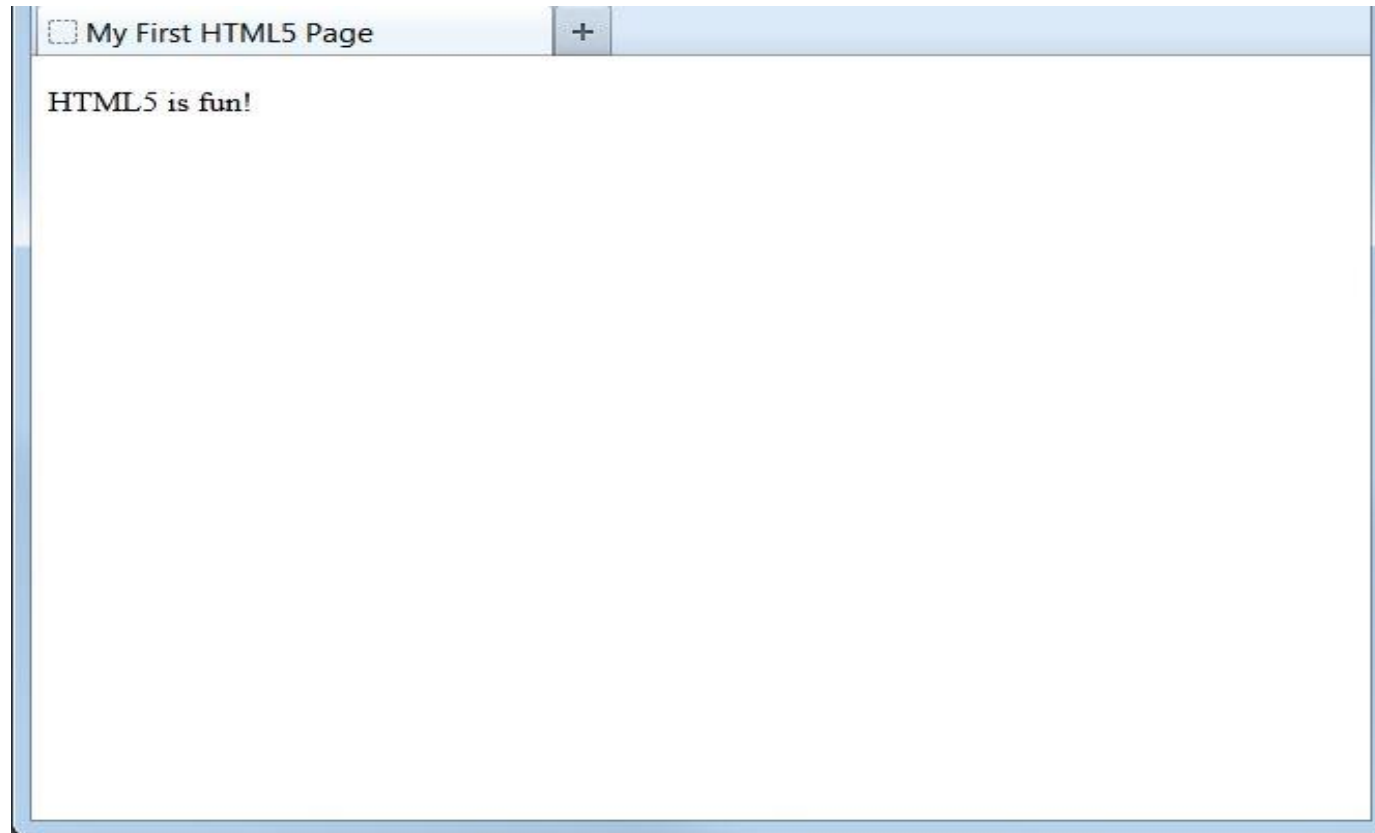


# First Look at HTML5

- Putting the prior sections together, and now adding the `<body>` section and closing tags, we have our first complete web page in HTML5:

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="utf-8">
 <title>My First HTML5 Page</title>
 <link rel="stylesheet" href="style.css">
</head>
<body>
 <p>HTML5 is fun!</p>
</body>
</html>
```

# First Look at HTML5



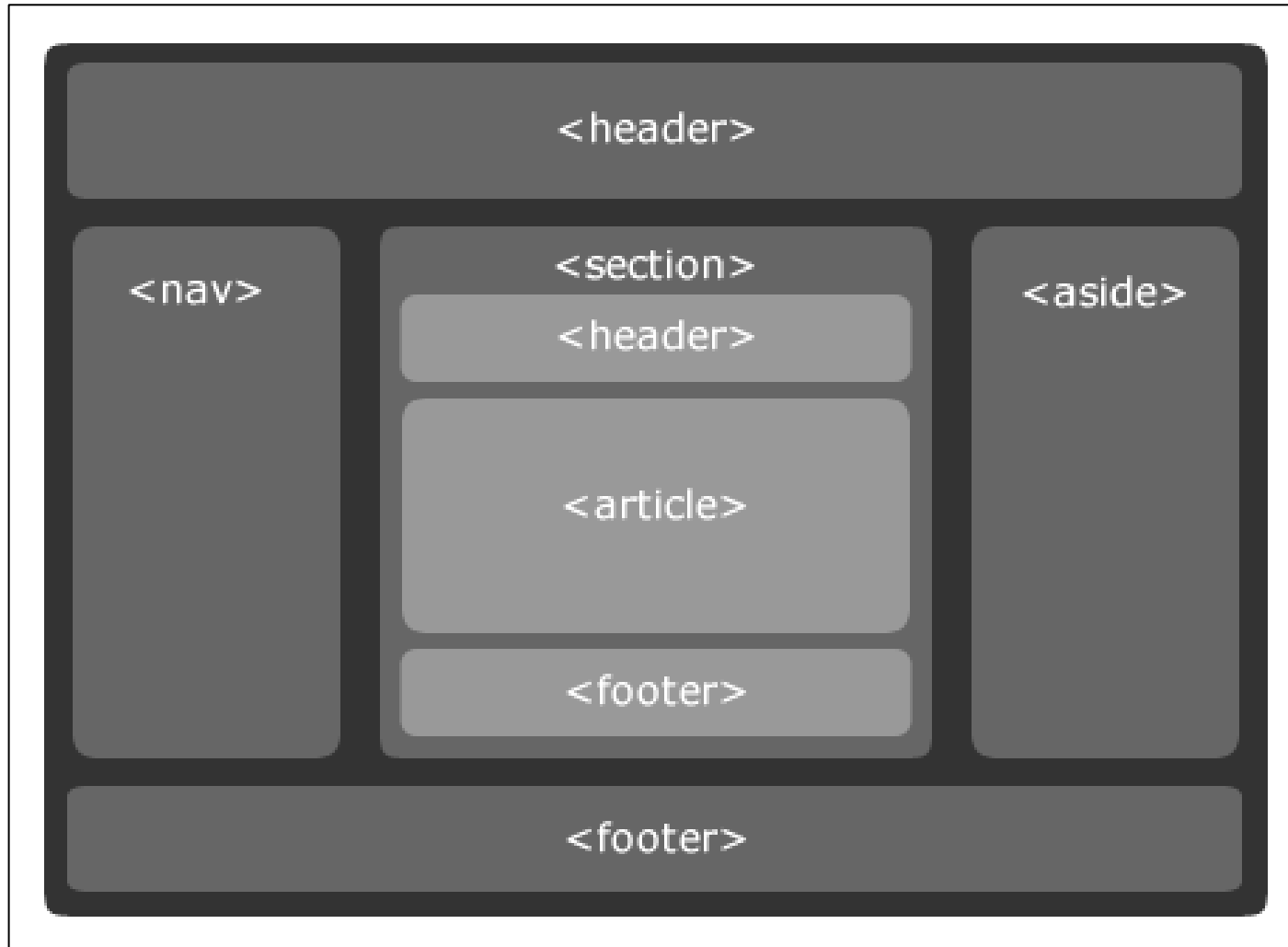
Even though we used HTML5, the page looks exactly the same in a web browser as it would in XHTML. Without looking at the source code, web visitors will not know which version of HTML the page was created with.



# HTML5 Basic Elements

- **<header>**: used to define a header for section or document
  - **<footer>**: used to define a footer for section or document
  - **<article>**: used to define a article
  - **<section>**: used to define a section within document
  - **<nav>**: used to define a navigation links
  - **<menuitem>**: used to define a list of menuitems
- 
- **header**, **nav** and **footer** are not doing fancy things like the other new HTML5 elements, but these elements are primarily designed to make the web structure more meaningful both for browsers and humans, just like how the World Wide Web inventor, **Tim Barners-Lee**, think of it.

# HTML5 Basic Elements





# HTML5 Elements: Video

```
<!DOCTYPE HTML>
```

```
<html>
```

```
<body>
```

```
<video src="movie.ogg" width="320" height="240"
controls="controls">
```

Your browser does not support the video tag.

```
</video>
```

```
</body>
```

```
</html>
```

The above example uses an **Ogg** file, and will work in Firefox, Opera and Chrome. To make the video work in Safari and future versions of Chrome, we must add an **MPEG4** and **WebM** file.



# HTML5 Elements: Video



```
<!DOCTYPE HTML>
```

```
<html>
```

```
<body>
```

```
<video width="320" height="240" controls="controls">
```

```
<source src="movie.ogg" type="video/ogg" />
```

```
<source src="movie.mp4" type="video/mp4" />
```

```
<source src="movie.webm" type="video/webm" />
```

Your browser does not support the video tag.

```
</video>
```

```
</body>
```

```
</html>
```

# HTML5 Elements: Video



Attribute	Value	Description
audio	muted	Defining the default state of the the audio. Currently, only "muted" is allowed
autoplay	autoplay	If present, then the video will start playing as soon as it is ready
controls	controls	If present, controls will be displayed, such as a play button
height	<i>pixels</i>	Sets the height of the video player
loop	loop	If present, the video will start over again, every time it is finished
poster	<i>url</i>	Specifies the URL of an image representing the video
preload	preload	If present, the video will be loaded at page load, and ready to run. Ignored if "autoplay" is present
src	<i>url</i>	The URL of the video to play
width	<i>pixels</i>	Sets the width of the video player

# HTML5 Elements: Audio



```
<!DOCTYPE HTML>
```

```
<html>
```

```
<body>
```

```
<audio controls="controls">
```

```
<source src="song.ogg" type="audio/ogg" />
```

```
<source src="song.mp3" type="audio/mpeg" />
```

Your browser does not support the audio element.

```
</audio>
```

```
</body>
```

```
</html>
```



# HTML5 Elements: Audio

Attribute	Value	Description
autoplay	autoplay	Specifies that the audio will start playing as soon as it is ready.
controls	controls	Specifies that controls will be displayed, such as a play button.
loop	loop	Specifies that the audio will start playing again (looping) when it reaches the end
preload	preload	Specifies that the audio will be loaded at page load, and ready to run. Ignored if autoplay is present.
src	<i>url</i>	Specifies the URL of the audio to play

# HTML5 Elements: Input Types



- HTML5 has several new input types for forms.
  - email
  - url
  - number
  - range
  - date pickers (date, month, week, time, datetime, datetime-local)
  - color

# HTML5 Elements: Input Attributes



- HTML5 has several new input attributes for forms.
  - required
  - autofocus
  - min, max, step
  - pattern
  - placeholder
  - readonly

# HTML Semantic Elements



- Semantic elements = elements with a meaning.

## What are Semantic Elements?

- A semantic element clearly describes its meaning to both the browser and the developer.
- Examples of **non-semantic elements**: `<div>` and `<span>` - Tells nothing about its content.
- Examples of **semantic elements**: `<form>`, `<table>`, and `<article>` - Clearly defines its content.

# HTML Semantic Elements



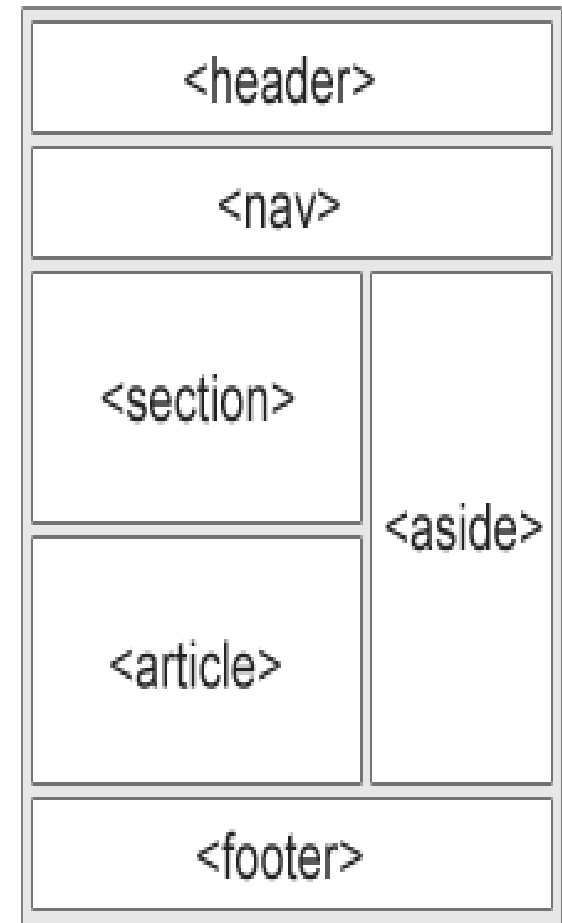
- Semantic Elements in HTML
- Many web sites contain HTML code like: `<div id="nav">` `<div class="header">` `<div id="footer">` to indicate navigation, header, and footer.
- In HTML there are some semantic elements that can be used to define different parts of a web page:



# HTML Semantic Elements



Tag	Description
<a href="#"><code>&lt;article&gt;</code></a>	Defines independent, self-contained content
<a href="#"><code>&lt;aside&gt;</code></a>	Defines content aside from the page content
<a href="#"><code>&lt;details&gt;</code></a>	Defines additional details that the user can view or hide
<a href="#"><code>&lt;figcaption&gt;</code></a>	Defines a caption for a <code>&lt;figure&gt;</code> element
<a href="#"><code>&lt;figure&gt;</code></a>	Specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.
<a href="#"><code>&lt;footer&gt;</code></a>	Defines a footer for a document or section
<a href="#"><code>&lt;header&gt;</code></a>	Specifies a header for a document or section
<a href="#"><code>&lt;main&gt;</code></a>	Specifies the main content of a document
<a href="#"><code>&lt;mark&gt;</code></a>	Defines marked/highlighted text
<a href="#"><code>&lt;nav&gt;</code></a>	Defines navigation links
<a href="#"><code>&lt;section&gt;</code></a>	Defines a section in a document
<a href="#"><code>&lt;summary&gt;</code></a>	Defines a visible heading for a <code>&lt;details&gt;</code> element
<a href="#"><code>&lt;time&gt;</code></a>	Defines a date/time



# Cascading Style Sheets (CSS)

# Introduction to CSS

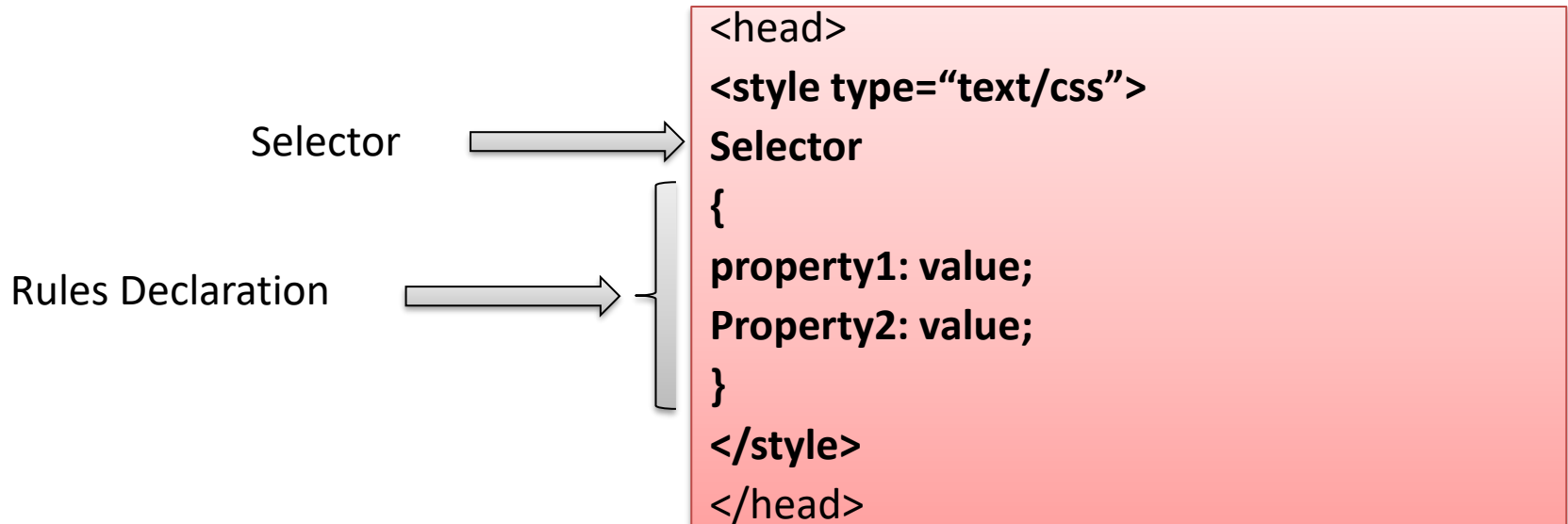
- CSS stands for Cascading Style Sheets and is a simple styling language which allows attaching style to HTML elements.
- Style Sheets are templates, very similar to templates in desktop publishing applications, containing a collection of rules declared to various selectors (elements).
- Style defines how to display HTML contents.

# Introduction to CSS

- Powerful and flexible way to specify the formatting of HTML elements.
  - Can define font, size, background color, background image ,margins etc..
- Share style Sheets across multiple documents or entire web site.
- Can specify a class definition for a style effectively defining new HTML elements.

# Understanding Style Rules

- The style characteristics for an HTML element are expressed by **Style Rules** .
- A set of style rules is called a **Style Sheet**.
- Style rules are contained in the **<STYLE>** element in the document's **<HEAD>** section.
- A Style Rule is composed of two parts: a **selector** and a **rules declaration**.



# Understanding Style Rules

- The **Selector** indicates the element to which the rule is applied.
- The **Declaration** determines the property values of a selector.
- The **Property** specifies a characteristic, such as color, font-family, position, and is followed by a colon (:).
- The **Value** expresses specification of a property, such as red for color, arial for font family, 12 pt for font-size, and is followed by a semicolon (;).
- The Style Sheet Property Names are **case-sensitive**.

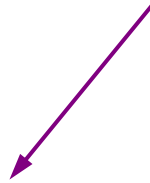
# Writing CSS: Example

Selector



body  
{

Declaration Block



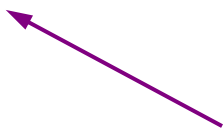
font-family: Tahoma, Arial, sans-serif;  
color: black;  
background: white;  
margin: 8px;

}

Property Name



Value



# CSS Selectors

- **There are three types of selectors in CSS:**
  - Element Selector
  - ID selector
  - CLASS selector



# CSS Selectors

- **Element Selector:**
  - The element selector selects all elements with the specified element name.
  - **Example:**

```
<style>
```

```
p
```

```
{
```

```
 background-color: yellow;
```

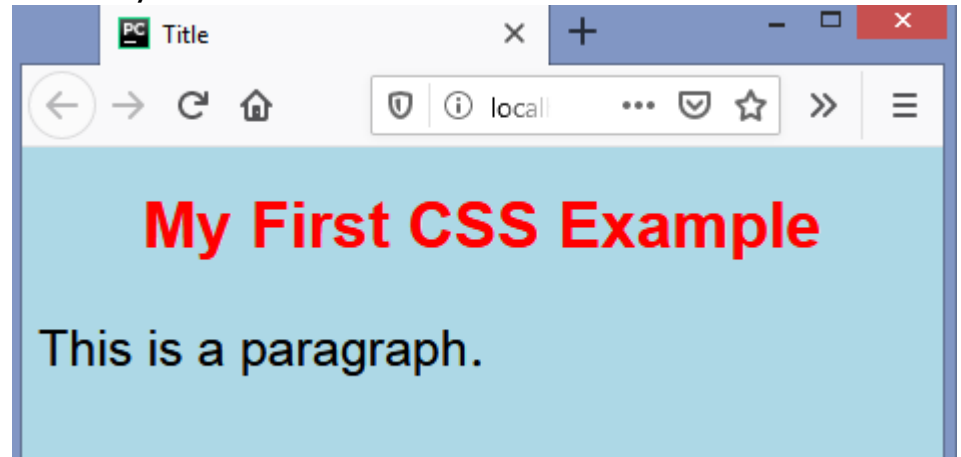
```
}
```

```
</style>
```

# CSS Selectors

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>Title</title>
 <style type="text/css">
body {
 background-color: lightblue;
}
h1 {
 color: red;
 text-align: center;
}
```

```
p {
 font-family: calibri ;
 font-size: 25px; }
</style>
</head>
<body>
<h1>My First CSS Example</h1>
<p>This is a paragraph.</p></body>
</html>
```



# CSS Selectors

- **ID Selector:**
  - The id selector is used to specify a style for a **single, unique element**.
  - The id selector uses the id attribute of the HTML element, and is defined with a "#".
  - **Each element can have only one ID.**
  - **Each page can have only one element with that ID.**
  - The style rule in example will be applied to the element with **id="para1"**.

# CSS Selectors

- **ID Selector Example:**

```
<html> <head>

<style type="text/css">

#para1

{

text-align: center;

color: blue;

}

</style></head>
```

```
<body>

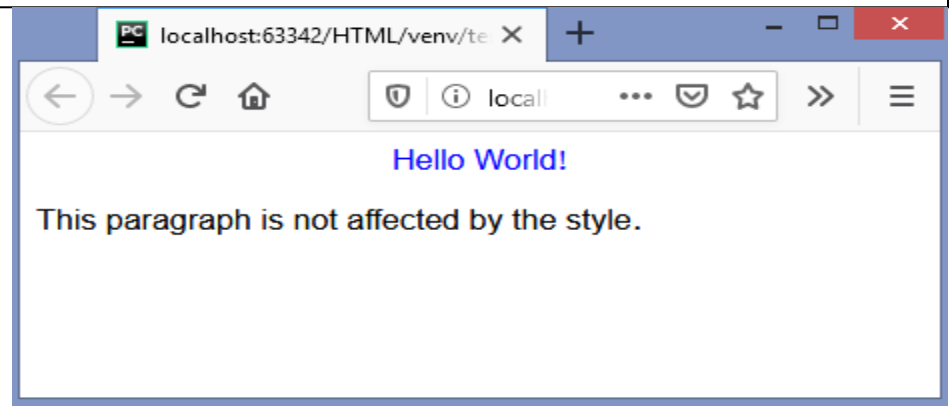
<p id="para1">Hello World!</p>

<p>This paragraph is not affected by the

style.</p>

</body>

</html>
```



# CSS Selectors

- **ID Selector problem:**
  - Your code will not pass validation if you use the same ID on more than one element.
  - Can not apply multiple ID to single element.

# CSS Selectors

- **CLASS Selector:**
  - The class selector is used to specify a style for a **group of elements**.
  - CLASS is an HTML attribute that assigns a class name to any HTML element on a Web page.
  - A class name is created by declaring a style rule and adding (.) flag character **indicating that the selector is a class selector**.
  - **You can use the same class on multiple elements.**
  - **You can use multiple classes on the same element.**

# CSS Selectors

- **CLASS Selector Example:**

```
<html> <head>

<style type="text/css">

 .center

 {

 text-align: center;

 }

</style>

</head>
```

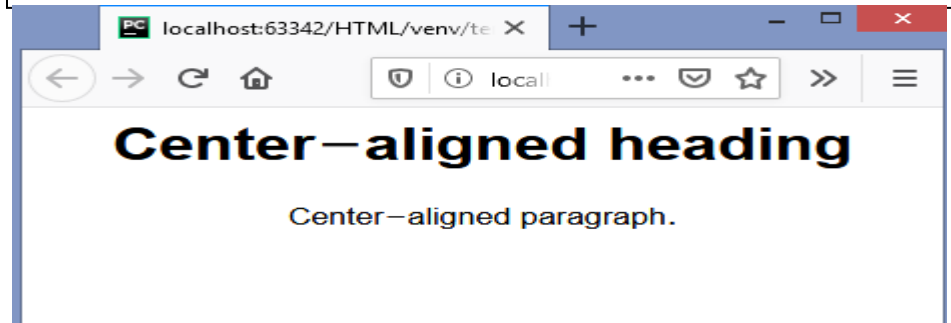
```
<body>

<h1 class="center"> Center-aligned
heading </h1>

<p class="center"> Center-aligned
paragraph. </p>

</body>

</html>
```



# CSS Selectors

- **CLASS Selector Example 2:**

```
<html> <head>

<style type="text/css">

p.center

{

text-align: center;

}

</style>

</head>
```

```
<body>

<h1 class="center"> This heading will not
be affected </h1>

<p class="center"> This paragraph will be
center aligned. </p>

</body>

</html>
```



# Types of CSS

- **Three ways to Specifying CSS:**

1. Inline Style Sheet
2. Internal Style Sheet
3. External Style Sheet

# Types of CSS

- **Inline Style Sheet:**

- An inline style may be used to apply a unique **style for a single element**.
- To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.
- Inline styles are the least flexible type of style to implement
- **For example:**

**`<p style="color: red;">Paragraph text goes in here</p>`**

- It suffers from major **drawback** - that if you wanted to change the style properties, you would have to edit each and every instance of the style on every single page of your website.

# Types of CSS

- **Internal Style Sheet:**

- An internal style sheet may be used if one single page has a unique style.
- With internal style sheets, a web page's styles are all specified at the top of the page code, within the <head> tag for the page.

- **For Example:**

```
<head>
```

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

```
 h1 { color: blue; font-weight: bold; }
```

```
 p { color: gray; }
```

```
</style>
```

```
</head>
```

# Types of CSS

- **External Style Sheet:**
  - It allow you to put all of your styling information into a completely separate CSS file.
  - We can then simply reference this file from within each web page, and the page's content will then be styled accordingly.
  - The obvious huge advantage of this method is that you need only change a style in your style sheet file (**.css file**), and the changes will cascade through the rest of your website.

# Types of CSS

- **External Style Sheet:**

- Each page must include a reference to the external style sheet file inside the **<link>** element. The **<link>** element goes inside the **<head>** section.

- **For Example:**

- **.html file**

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

- **mystyle.css file**

```
h1 { color:pink; }

p { margin-left:20px; }
```

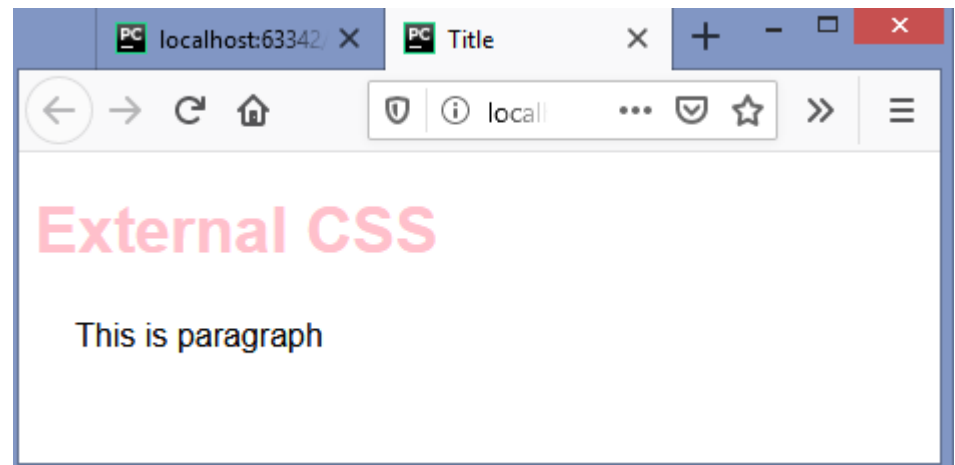
# Types of CSS

- HTML External CSS.html

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>Title</title>
 <link rel="stylesheet" type="text/css" href="first.css" />
</head>
<body>
<h1>External CSS</h1>
<p> This is paragraph</p>
</body>
</html>
```

- First.css

```
h1 { color:pink; }
p { margin-left:20px; }
```



# HTML page with CSS Example

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
 background-color: linen;
}
h1 {
 color: maroon;
 margin-left: 40px;
}
</style>
</head>
```

```
<body>

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

</body>
</html>
```

**This is a heading**

This is a paragraph.

# Using the Pseudo-Class Selector

- Classes are used to add special effects to some selectors.
- The syntax of pseudo-classes:

**selector:pseudo-class { property:value; }**

## Example: Anchor pseudo class

- **a:link** { color:#FF0000; }                      /\* unvisited link \*/
- **a:visited** { color:#00FF00; }                      /\* visited link \*/
- **a:hover** { color:#FF00FF; }                      /\* mouse over link \*/
- **a:active** { color:#0000FF; }                      /\* selected link \*/



# Using the Pseudo-Class Selector

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
 <style>
```

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

```
 a:visited {
 color: green; }
```

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

```
 a:active {
 color: blue; }
```

```
 </style>
```

```
</head>
```

```
</head>
```

```
<body>
```

```
<p><a href="http://www.gcet.ac.in"
target="_blank">This is a link</p>
```

```
<p>Note: a:hover MUST come after a:link
and a:visited in the CSS definition in order to be
effective.</p>
```

```
<p>Note: a:active MUST come after
a:hover in the CSS definition in order to be
effective.</p>
```

```
</body>
```

```
</html>
```

# Using the Pseudo-Class Selector

- **The :first-child Pseudo-class:**

- The **:first-child** pseudo-class matches a specified element that is the first child of another element. **For Example:**

```
<html>
 <head>
 <style type="text/css">
 p:first-child
 {
 color: blue;
 }
 </style></head>
 <body>
 <p> This is some text. </p>
 <p> This is some text. </p>
 </body>
</html>
```

This is some text.

This is some text.

# Using the Pseudo-Class Selector

- Match the first `<i>` element in all `<p>` elements:

```
<html>
```

```
<head>
```

```
<style>
```

```
 p i:first-child {
```

```
 color: blue;
```

```
 }
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<p>Welcome to <i>GCET</i>. Welcome to <i>GCET</i>.</p>
```

```
< p>Welcome to <i>GCET</i>. Welcome to <i>GCET</i>.</p>
```

```
</body>
```

```
</html>
```

Welcome to *GCET*. Welcome to *GCET*.

Welcome to *GCET*. Welcome to *GCET*.

# Using the Pseudo-Class Selector

- Match all `<i>` elements in all first child `<p>` elements:

```
<html>
```

```
<head>
```

```
<style>
```

```
 p:first-child i {
```

```
 color: blue;
```

```
 }
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<p>Welcome to <i>GCET</i>. Welcome to <i>GCET</i>.</p>
```

```
< p>Welcome to <i>GCET</i>. Welcome to <i>GCET</i>.</p>
```

```
</body>
```

```
</html>
```

Welcome to *GCET*. Welcome to *GCET*.

Welcome to *GCET*. Welcome to *GCET*.

# Using the Pseudo-Class Selector

- **Example:**

```
<head><style>
```

```
p {
```

```
 display: none;
```

```
 background-color: yellow;
```

```
 padding: 20px;
```

```
}
```

```
div:hover p {
```

```
 display: block;
```

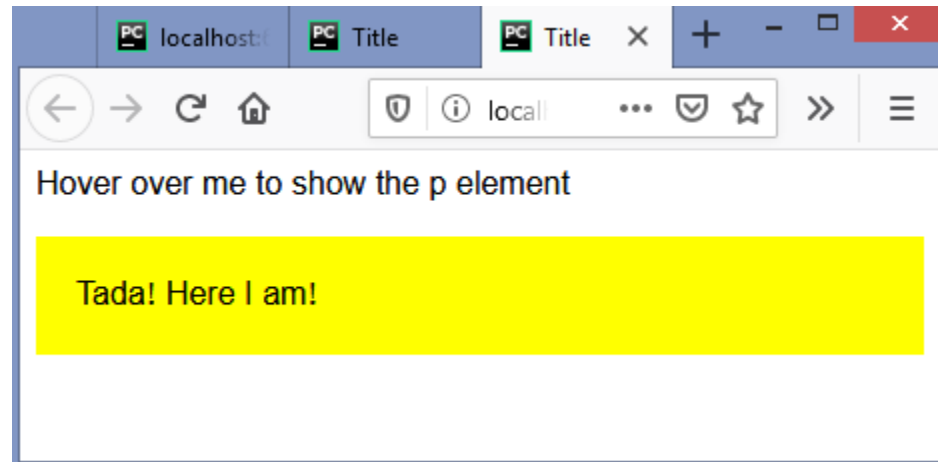
```
}
```

```
</style></head><body>
```

```
<div>Hover over me to show the p element
```

```
 <p>Tada! Here I am!</p>
```

```
</div></body>
```



# CSS Properties

- **Background Property:**

Property	Description	Values
background-color	Specifies the background color to be used.	blue, #ff00ff, rgb(255,0,0)
background-image	Specifies ONE or MORE background images to be used.	url('img1.jpg');
background-position	Specifies the position of the background images.	left top, left center, right top, right bottom, x% y%
background-size	Specifies the size of the background images.	100px 50px, cover, contain, 50px (auto)
background-repeat	Specifies how to repeat the background images.	repeat, repeat-x, repeat-y, no-repeat
background-attachment	Specifies whether the background images are fixed or scrolls with the rest of the page.	scroll, fixed

# CSS Properties

- **Background Property Example:**

```
<style>
 body {
 background-color: yellow;
 }
 h1 {
 background-color: #00ff00;
 }
 p {
 background-color: rgb(255,0,255);
 }
</style>
```

# CSS Properties

- **Background Shorthand Property:**

- The background shorthand property sets all the background properties in one declaration.
- The properties that can be set, are: **background-color, background-image, background-position, background-size, background-repeat, background-attachment.**

- **Example:**

```
body {
 background: #00ff00 url("smiley.gif") center no-repeat
 fixed;
}
```



# CSS Properties

- **Border Property:**

Property	Description	Values
<code>border-width</code>	Specifies the width of the border. Default value is "medium"	medium, thin, thick, length (default is medium=2px)
<code>border-style</code>	Specifies the style of the border. Default value is "none"	none, dotted, dashed, solid, double, grooved
<code>border-color</code>	Specifies the color of the border. Default value is the color of the element	blue, #ff00ff, rgb(255,0,0)

- **Border Shorthand Property:**

```
p {
 border: 5px solid red;
}
```

# CSS Properties

- **Margin Property:**

Property	Description	Values
<code>margin-top</code>	Set the top margin of an element.	Length in px, cm, %, pt
<code>margin-right</code>	Set the right margin of an element.	Length in px, cm, %, pt
<code>margin-bottom</code>	Set the bottom margin of an element.	Length in px, cm, %, pt
<code>margin-left</code>	Set the left margin of an element.	Length in px, cm, %, pt

- **Margin Shorthand Property:**

```
p {
 margin: 2cm 4cm 3cm 4cm; //top right bottom left
 margin: 2cm 4cm 3cm; // top (right left) bottom
 margin: 2cm 4cm; // (top bottom) (right left)
}
```

# CSS Properties

- **Padding Property:**

Property	Description	Values
padding-top	Set the top padding(space) of an element.	Length in px, cm, %, pt
padding-right	Set the right padding of an element.	Length in px, cm, %, pt
padding-bottom	Set the bottom padding of an element.	Length in px, cm, %, pt
padding-left	Set the left padding of an element.	Length in px, cm, %, pt

- **Margin Shorthand Property:**

```
p {
 padding: 2cm 4cm 3cm 4cm; //top right bottom left
 padding: 2cm 4cm 3cm; // top (right left) bottom
 padding: 2cm 4cm; // (top bottom) (right left)
}
```

# CSS Properties

- **Text Manipulation Property:**

Property	Description	Values
text-align	The horizontal alignment of text in an element.	left, right, center, justify
text-decoration	The decoration added to text.	underline, overline, line-through
text-indent	The indentation of the first line in a text-block.	Length in px, cm, pt, %
text-transform	Controls the capitalization of text.	Capitalize, uppercase, lowercase
text-shadow	Adds shadow to text.	h-shadow v-shadow color e.g. 5px 5px red

# CSS Properties

- **List Property:**

Property	Description	Values
<code>list-style-type</code>	Specifies the type of list-item marker in a list.	disc, circle, decimal, decimal-leading-zero, lower-alpha, lower-greek, lower-roman, square, upper-alpha
<code>list-style-image</code>	Replaces the list-item marker with an image.	<code>url('img1.gif');</code>
<code>list-style-position</code>	Markers should appear inside the content flow (results in an extra indentation)	Inside, outside

# Positioning using CSS

- The **position** property specifies the type of positioning method used for an element.

- CSS Position & Helper Properties

- There are 4 main values of the Position Property:

**position: static | relative | absolute | fixed**

- and additional properties for setting the coordinates of an element (“helper properties”):

**top | right | bottom | left AND the z-index**

- They also work differently depending on the position value.

# Positioning using CSS

- **position: static;**
  - position: static is the default value. Whether we declare it or not, elements are positioned in a normal order on the webpage. .
  - Static positioned elements are **not affected** by the top, bottom, left, and right properties.
  - An element with **position: static;** is not positioned in any special way; it is always positioned according to the **normal flow** of the page.

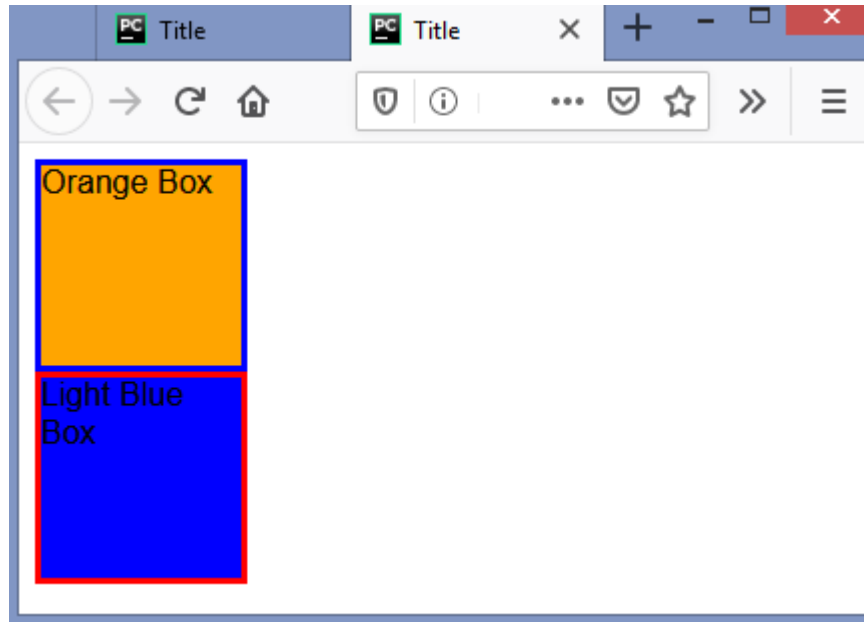
# Positioning using CSS

- **Position: Static**
- ```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
  <style>
    .box-orange {
      border: solid blue;
      background: orange;
      height: 100px;
      width: 100px;
    }
  </style>
</head>
<body>
  <div class="box-orange">Orange
  Box</div>
  <div class="box-blue">Light Blue
  Box</div>
</body>
</html>
```
- ```
.box-blue {
 border: solid red;
 background: blue;
 height: 100px;
 width: 100px;
 position: static;
}
```



# Positioning using CSS

- 



same result with & without **position: static**

As we can see in the picture, defining **position: static** or not doesn't make any difference. The boxes are positioned according to the **normal document flow**.

# Positioning using CSS

- **position: relative;**
  - An element with **position: relative;** is positioned relative to its normal position.
  - Setting the **top**, **right**, **bottom** and **left** properties of a relatively-positioned element will cause it to be adjusted away from its normal position.
  - Other content will not be adjusted to fit into any gap left by the element.
  - **Let's move the orange box next to the blue one.**

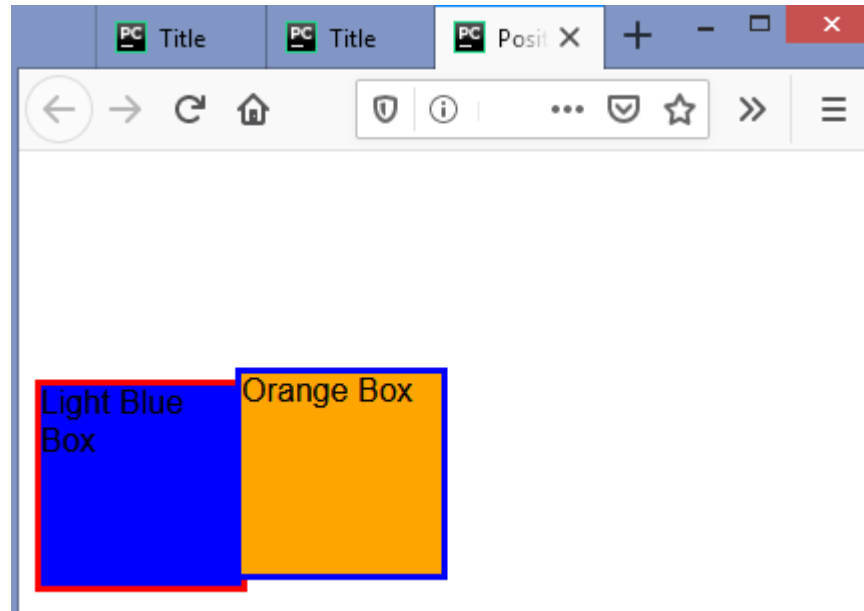
# Positioning using CSS

- **position: relative;** Example:

```
• <!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>Position Property</title>
 <style>
 .box-orange {
 position: relative;
 border: solid blue;
 background: orange;
 height: 100px;
 width: 100px;
 top: 100px;
 left: 100px;
 }
```

```
 .box-blue {
 border: solid red;
 background: blue;
 height: 100px;
 width: 100px;
 position: static;
 }
 </style>
</head>
<body>
 <div class="box-orange">Orange
 Box</div>
 <div class="box-blue">Light Blue
 Box</div>
</body>
</html>
```

# Positioning using CSS



Orange box is moved 100px to bottom & right, relative to its normal position

**NOTE:** Using **position: relative** for an element, doesn't affect other elements' positions.

# Positioning using CSS

- **position: absolute;**
  - In **position: relative**, the element is positioned relative to itself. However, an absolutely positioned element is relative to its parent.
  - An element with **position: absolute** is removed from the normal document flow. It is positioned automatically to the starting point (top-left corner) of its parent element. If it doesn't have any parent elements, then the initial document <html> will be its parent.
  - Since **position: absolute** removes the element from the document flow, other elements are affected and behave as the element is removed completely from the webpage.

# Positioning using CSS

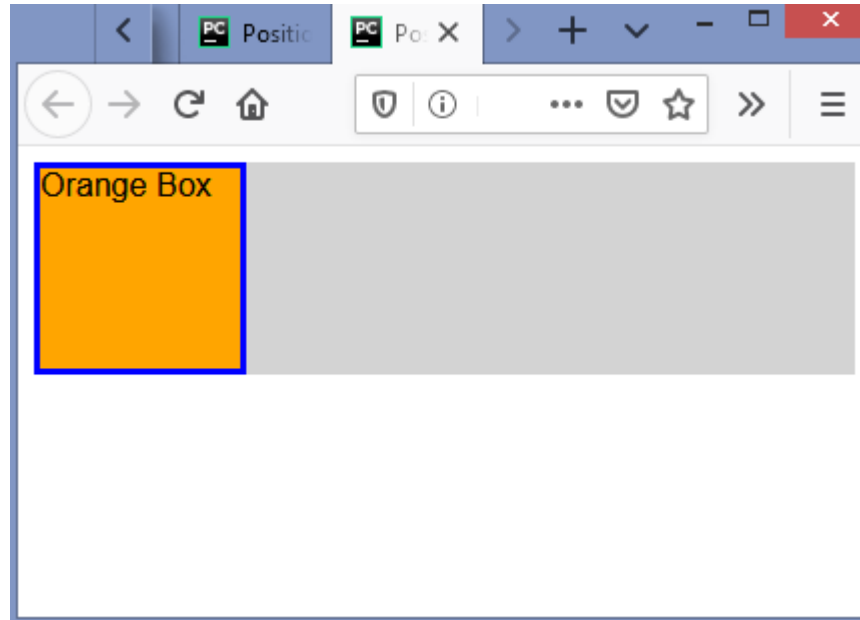
- ```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Position
Property</title>
  <style>
    .box-orange {
      position: absolute;
      border: solid blue;
      background: orange;
      height: 100px;
      width: 100px;

      //right: 5px;
    }
    .box-blue {
      border: solid red;
      background: blue;
```

```
      height: 100px;
      width: 100px;
      position: static;
    }
    .container {
      position: relative;
      background: lightgray;
    }

  </style>
</head>
<body>
  <div class="container">
    <div class="box-orange">Orange
Box</div>
    <div class="box-blue">Light Blue
Box</div>
  </div>
</body>
</html>
```

Positioning using CSS



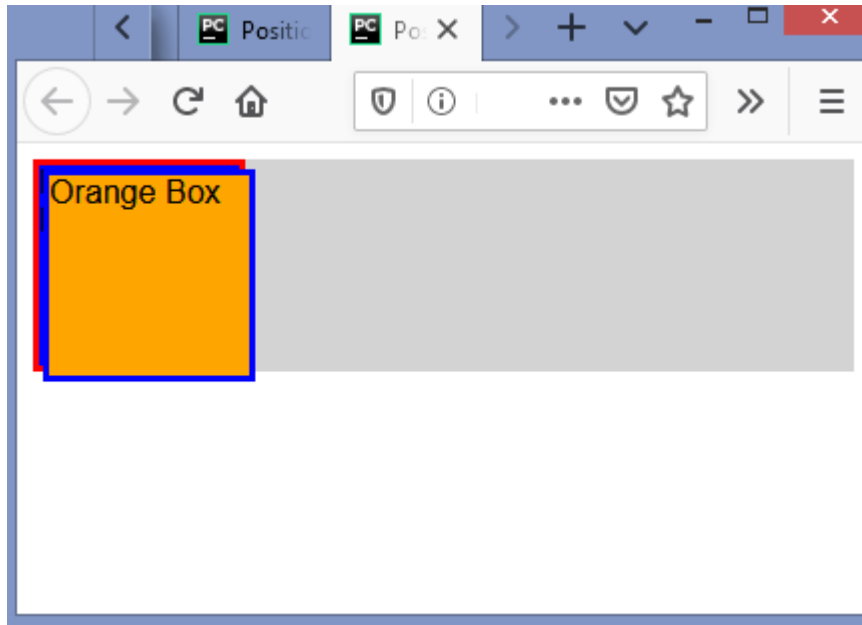
position: absolute takes the element to the **beginning** of its parent

Now it looks like the blue box has disappeared, but it hasn't. The blue box behaves like the orange box is removed, so it shifts up to the orange box's place.

Let's move the orange box 5 pixels:

Positioning using CSS

```
.box-orange {  
  position: absolute;  
  background: orange;  
  width: 100px; height:  
  100px; left: 5px; top:  
  5px;  
}
```

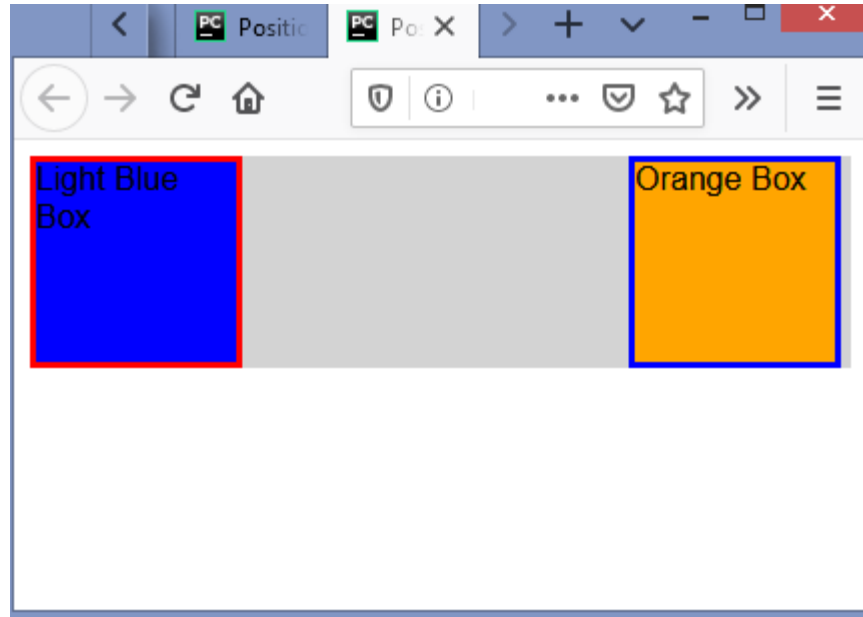


Now we can see the blue box

The coordinates of an **absolute** positioned element are **relative to its parent** if the parent also has a **non-static position**. Otherwise, helper properties position the element relative to the **initial <html>**.

Positioning using CSS

```
..box-orange {  
  position: absolute;  
  border: solid blue;  
  background: orange;  
  height: 100px;  
  width: 100px;  
  right: 5px;  
}
```



5px relative to the most-right of parent

Positioning using CSS

- **position: fixed;**
 - Like position: absolute, fixed positioned elements are also removed from the normal document flow.
 - The differences are:
 - They are only relative to the <html> document, not any other parents.
 - They are not affected by scrolling.
 - The top, right, bottom, and left properties are used to position the element.
 - A fixed element does not leave a gap in the page where it would normally have been located.

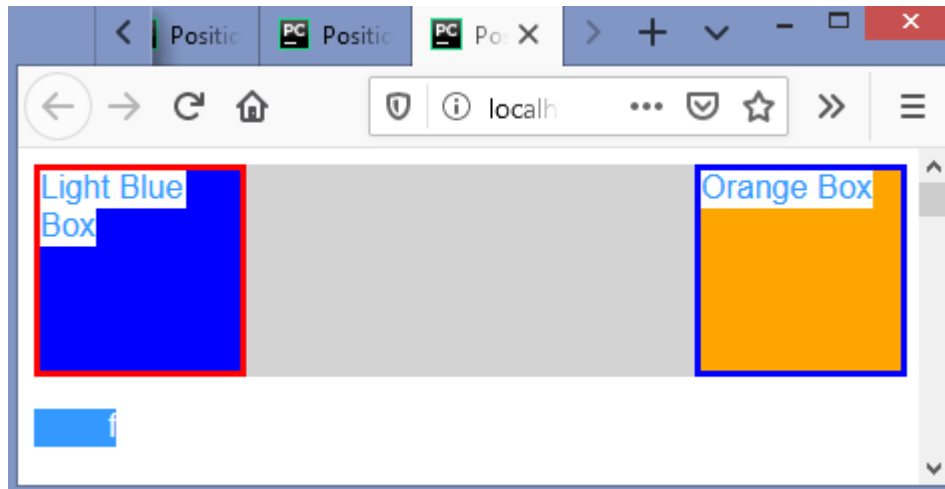
Positioning using CSS

- ```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>Position Property</title>
 <style>
 .box-orange {
 position: fixed;
 border: solid blue;
 background: orange;
 height: 100px;
 width: 100px;

 right: 5px;
 }

 .box-blue {
 border: solid red;
 background: blue;
 height: 100px;
 width: 100px;
 position: static;
 }
 </style>
</head>
<body>
 <div class="container">
 <div class="box-orange">Orange
 Box</div>
 <div class="box-blue">Light Blue
 Box</div>
 </div>
</body> </html>
```

# Positioning using CSS



# Positioning using CSS

- **Overlapping Elements:**

- What is this z-index?
- We have height and width (x, y) as 2 dimensions. Z is the 3rd dimension. An element in the webpage comes in front of other elements as its z-index value increases.
- Z-index doesn't work with position: static or without a declared position.
- An element can have a positive or negative stack order.



# Positioning using CSS

- **Overlapping Elements with z-index:**

```
<style>
```

```
img {
```

```
 position: absolute;
```

```
 left: 0px;
```

```
 top: 0px;
```

```
 z-index: -1;
```

```
}
```

```
</style></head><body>
```

```
<h1>This is a heading</h1>
```

```

```

```
<p>Because the image has a z-index of -1, it will be placed behind the
text.</p>
```

# Positioning using CSS

## Positioning Text In an Image

How to position text over an image:

### Example







# The CSS Box Model

- `<!DOCTYPE html>`

`<head>   <title>Title</title>`

`<style>`

```
.main {
 font-size:36px;
 font-weight:bold;
 Text-align:center; }
```

```
.outer {
 margin-left:60px;
 border:50px solid green;
 width:300px;
 height:200px;
```

`text-align:center;`

```
padding:50px; }
```

```
.inner {
 font-size:42px;
 font-weight:bold;
 color:blue;
 margin-top:60px;
 background-color:yellow; }
```

```
.nested {
 font-size:18px;
 font-weight:bold;
 background-color: pink; </style>
```

`</head> <body>`

`<div class = "main">CSS Box-Model Property</div>`

`<div class = "outer">`

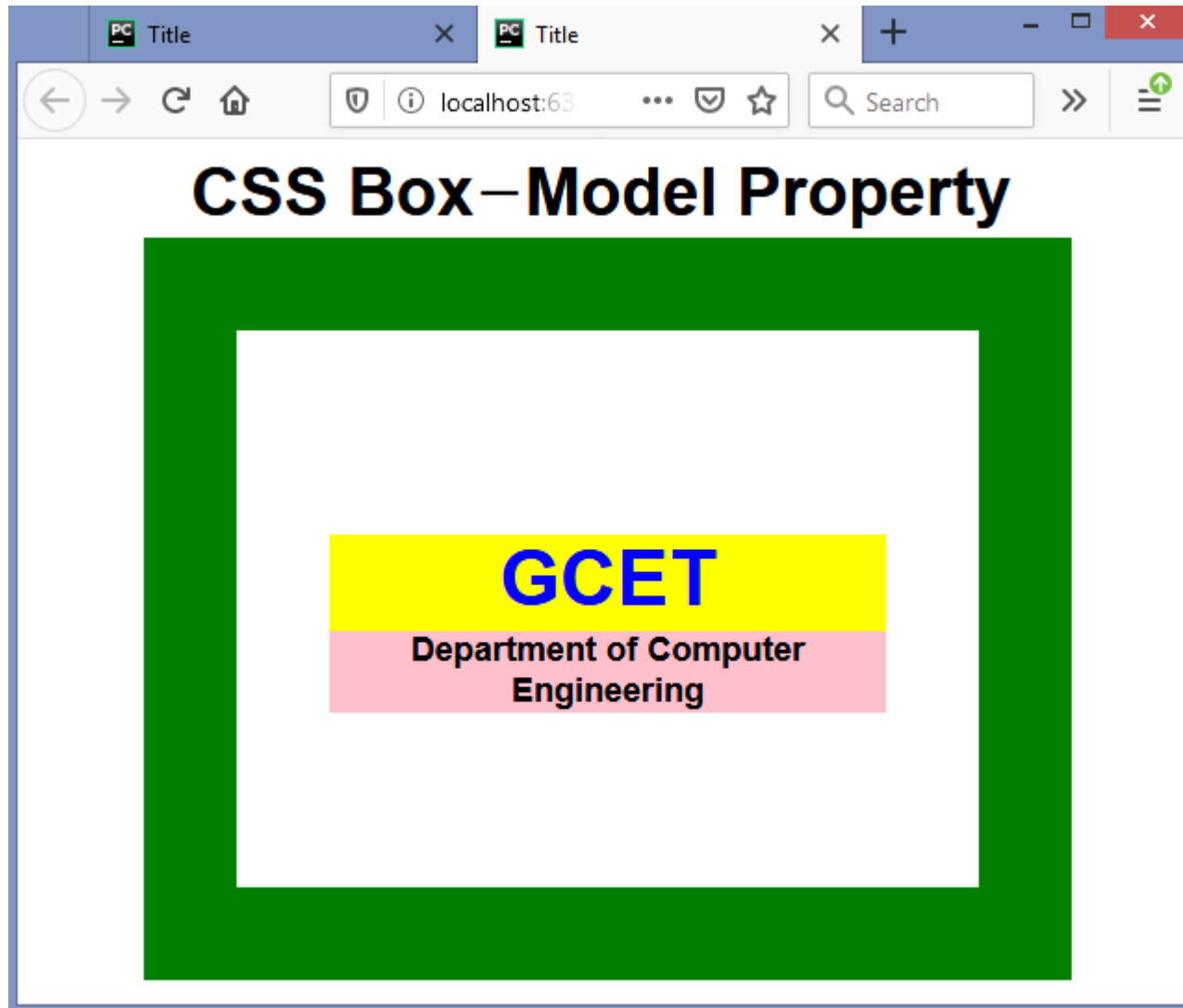
`<div class = "inner">GCET</div>`

`<div class = "nested">Department of`

`Computer Engineering </div>`

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

# The CSS Box Model



# CSS Layout - float and clear

- **float Property:** With CSS float, an element can be pushed to the left or right, allowing other elements to wrap around it.
- Elements are floated horizontally, this means that an element can only be floated left or right, not up or down.
- The elements after the floating element will flow around it.
- If an image is floated to the left, a following text flows around it, to the right.



Hello World!

W3Schools background image

The background image is

This is some text. This is some text. This is some text. This is some text. This is text. This is some text. This is some text. This is some text. This is some text. T some text. This is some text. This is some text. This is some text. This is some t

In the paragraph below, we have added an image with style **float**. The result is t

In the paragraph below, we have added an image with style **float**. The result is that the image v

# CSS Layout - float and clear

- **float Example:**

```
<head><style>
```

```
img
```

```
{
```

```
 float: right;
```

```
}
```

```
</style> </head><body>
```

```
<p>In the paragraph below, we have
 added an image with style
 float. The result is that
 the image will float to the right in
 the paragraph.</p>
```

```

```

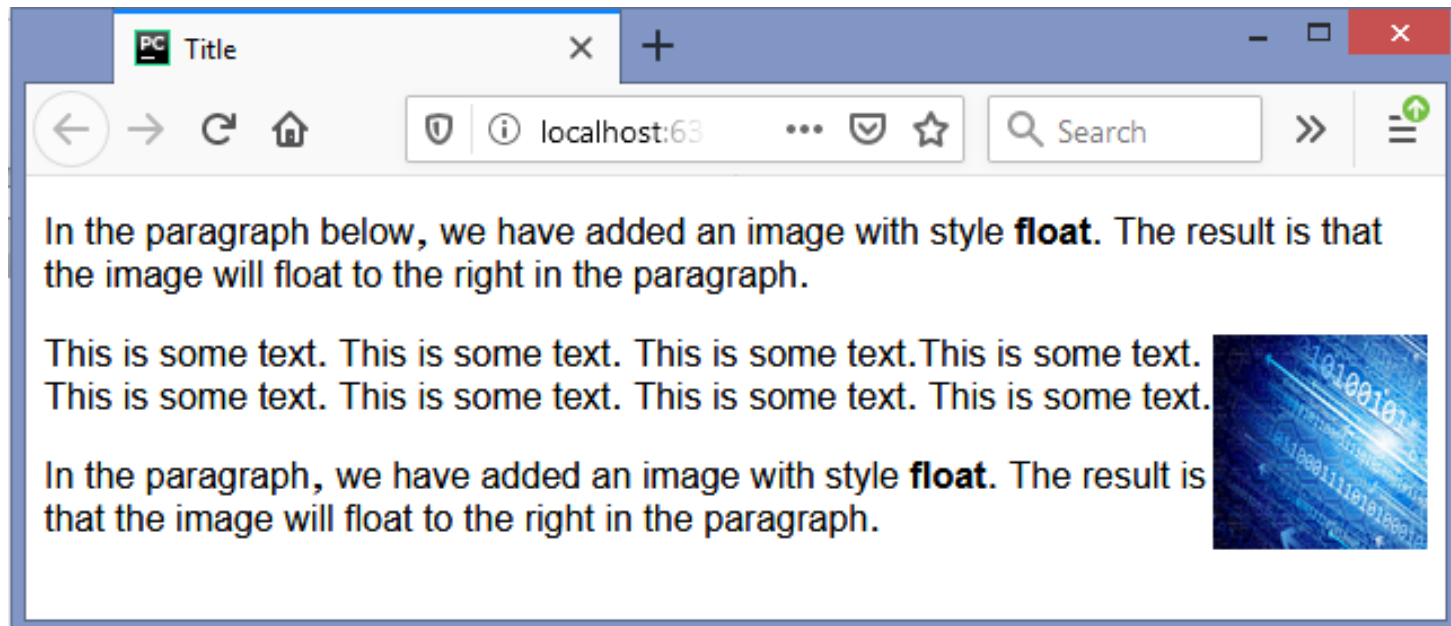
```
<p>This is some text. This is some text.
 This is some text.This is some text.
 This is some text. This is some text.
 This is some text. This is some text.
```

```
</p>
```

```
<p>In the paragraph, we have added
 an image with style float.
 The result is that the image will
 float to the right in the
 paragraph.</p>
```

```
</body>
```

# CSS Layout - float and clear



# CSS Layout - float and clear

- **float multiple Example:**

```
<head><style>
```

```
.thumbnail
```

```
{
```

```
 float:left;
```

```
 width:200px;
```

```
 height:150px;
```

```
 margin:5px;
```

```
}
```

```
</style> </head><body>
```

```
<h3>Image Gallery</h3>
```

```
<p>Try resizing the window to see what
happens when the images does not
have enough room.</p>
```

```

```

```

```

```

```

```

```

```

```

```

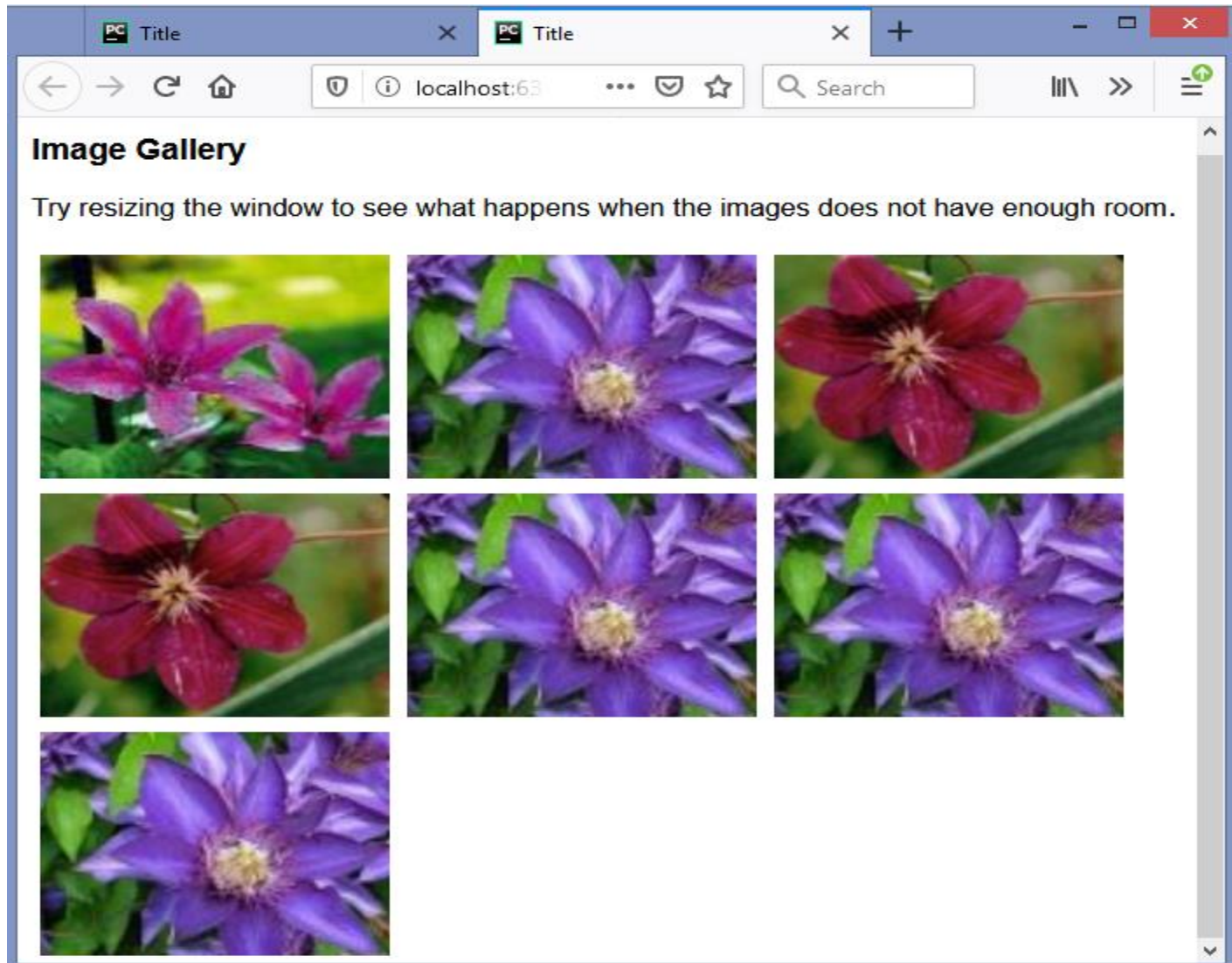
```

```

```

```
</body>
```

# CSS Layout - float and clear



# CSS Layout - float and clear

- **clear Property:**

- The clear property is used to control the behavior of floating elements.
- Elements after a floating element will flow around it. To avoid this, use the clear property.

## **Image Gallery**



## **Second row**





# CSS Layout - float and clear

- **clear Property Example:**

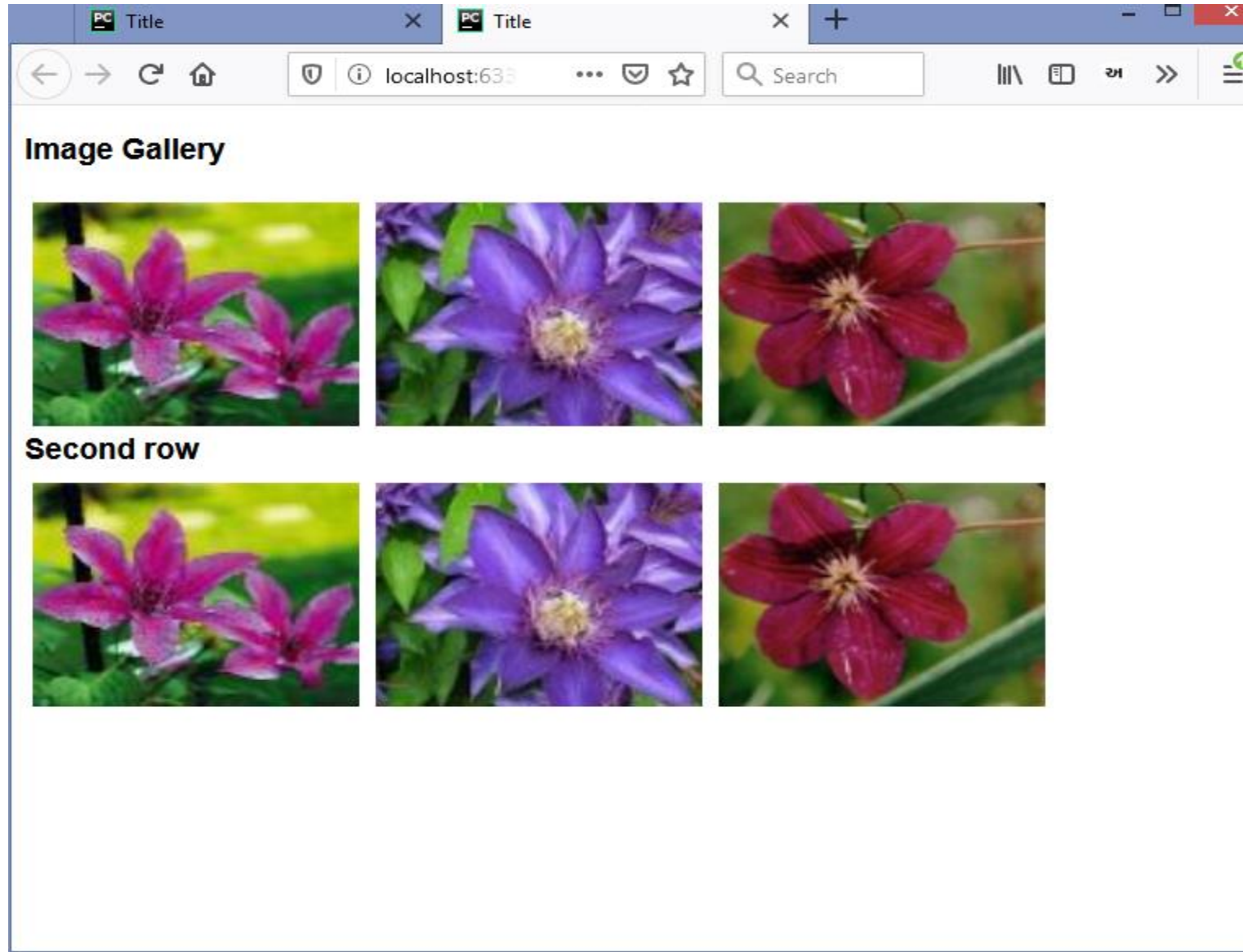
```
<head><style>
.thumbnail
{
 float:left;
 width:200px;
 height:150px;
 margin:5px;
}
.text_line
{
 clear:both;
 margin-bottom:5px;
}
</style> </head><body>
```

```
<h3>Image Gallery</h3>

<h3 class="text_line">Second row</h3>

</body>
```

# CSS Layout - float and clear



# CSS Layout - float and clear

- **clear Property Example:**

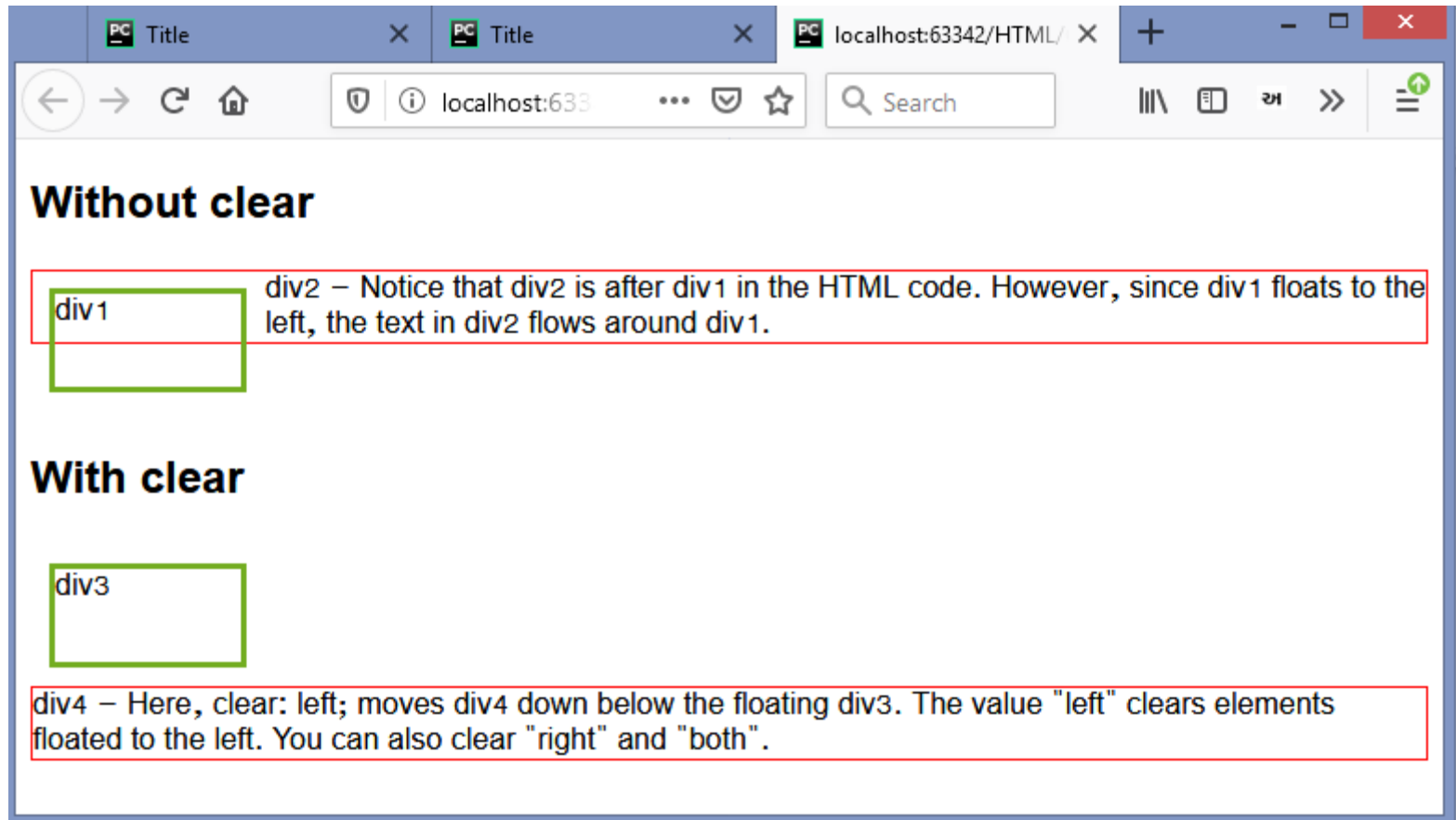
```
<!DOCTYPE html>
<html>
<head>
<style>

```

```
.div4 {
 border: 1px solid red;
 clear: left; }
</style> </head> <body>
<h2>Without clear</h2>
<div class="div1">div1</div>
<div class="div2">div2 - Notice that
div2 is after div1 in the HTML code.
However, since div1 floats to the left,
the text in div2 flows around
div1.</div>

<h2>With clear</h2>
<div class="div3">div3</div>
<div class="div4">div4 - Here, clear:
left; moves div4 down below the
floating div3. The value "left" clears
elements floated to the left. You can
also clear "right" and "both".</div>
</body> </html>
```

# CSS Layout - float and clear



The screenshot shows a web browser window with three tabs. The active tab is titled 'localhost:63342/HTML/'. The address bar shows 'localhost:633'. The page content is divided into two sections: 'Without clear' and 'With clear'.

**Without clear**

div1

div2 – Notice that div2 is after div1 in the HTML code. However, since div1 floats to the left, the text in div2 flows around div1.

**With clear**

div3

div4 – Here, clear: left; moves div4 down below the floating div3. The value "left" clears elements floated to the left. You can also clear "right" and "both".

# CSS Layout - display

- **display Property:**

- The display property is the most important CSS property for controlling layout.
- The display property specifies if/how an element is displayed.
- Mostly used for Horizontal or Vertical Navigation bar.
- Values are: **none**, **block** and **inline**.

```
<style>
```

```
li {
```

```
 display: inline;
```

```
//
```

```
 display: block;
```



[HTML](#)  
[CSS](#)  
[JavaScript](#)

```
}
```

```
</style>
```

```

```

```
 HTML
```

```
 CSS
```

```
 JavaScript
```

```

```

[HTML](#) [CSS](#) [JavaScript](#)

# CSS Layout - display

- **display Property Example:**

```
<head><style>
<style>
#d {
 width: 500px;
 height: 50px;
 background-color: lightblue;
}
</style>
<script>
function hide() {
 document.getElementById("d").style.display=
 "none";
}
```

```
function show() {
 document.getElementById("d").style.display=
 "block";
}
</script>
</style> </head><body>

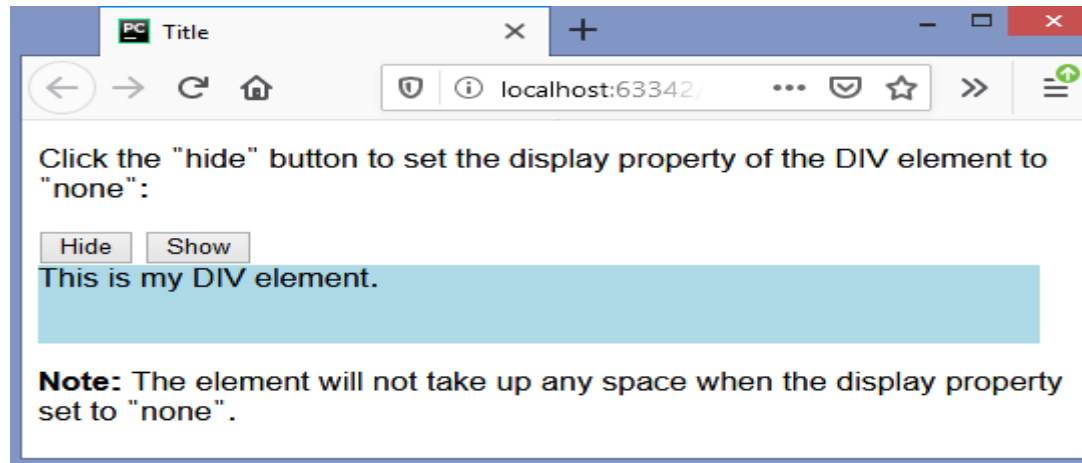
<p>Click the "hide" button to set the display
property of the DIV element to "none":</p>

<button onclick="hide()">Hide</button>
<button onclick="show()">Show</button>

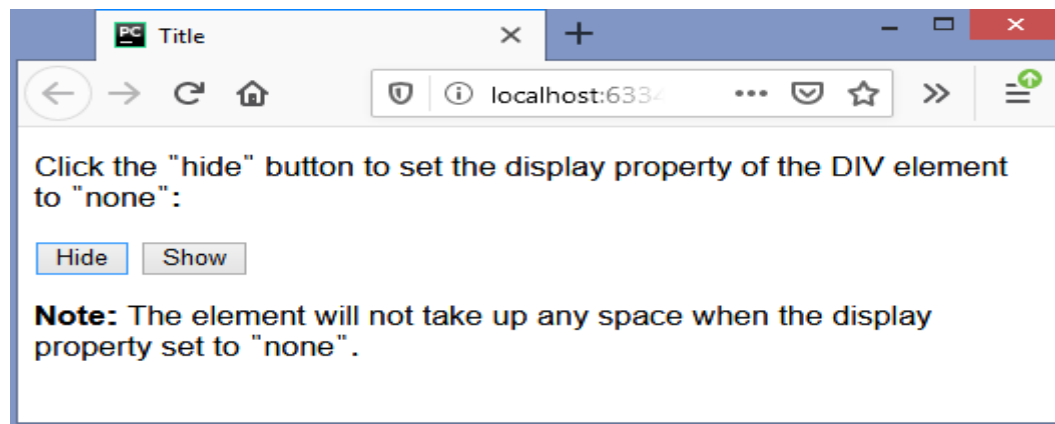
<div id="d">This is my DIV element.</div>
<p>Note: The element will not take up any
space when the display property set to
"none".</p>
</body>
```

# CSS Layout - display

## Initial View



## After click on Hide



# CSS3

- CSS3 is the latest standard for CSS.
- CSS3 is completely backwards-compatible with earlier versions of CSS.



# CSS3 – Rounded Borders

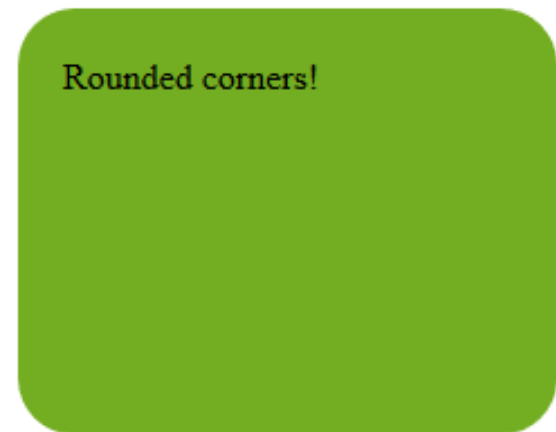
- With the CSS3 border-radius property, you can give any element "rounded corners".
- **Example:**

**.css**

```
#rcorners {
 border-radius: 25px; // 10 px 50px 10px 50px → t-l, t-r, b-r and b-l
 background: #73AD21;
 padding: 20px;
 width: 200px;
 height: 150px;
}
```

**.html**

```
<p id="rcorners">Rounded corners!</p>
```



# CSS3 – Rounded Borders

- CSS border-radius - Specify Each Corner
- The border-radius property can have from one to four values. Here are the rules:
- **Four values - border-radius: 15px 50px 30px 5px;** (first value applies to top-left corner, second value applies to top-right corner, third value applies to bottom-right corner, and fourth value applies to bottom-left corner):
- **Three values - border-radius: 15px 50px 30px;** (first value applies to top-left corner, second value applies to top-right and bottom-left corners, and third value applies to bottom-right corner):
- **Two values - border-radius: 15px 50px;** (first value applies to top-left and bottom-right corners, and the second value applies to top-right and bottom-left corners):
- **One value - border-radius: 15px;** (the value applies to all four corners, which are rounded equally):



# CSS3 –Borders Image

- With the CSS3 border-image property, you can set an image to be used as the border around an element.

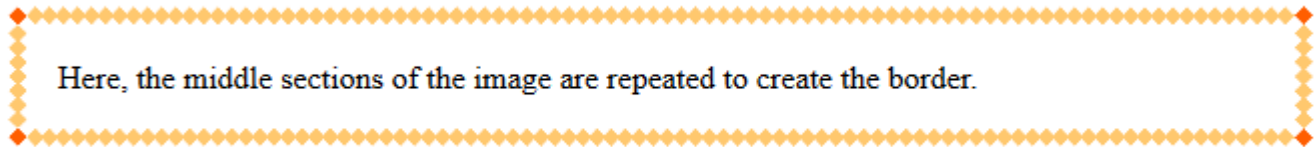
- **Example:**

**.css**

```
#borderimg {
 border: 10px solid transparent;
 padding: 15px;
 border-image: url('border.png') 30 repeat;
}
```

**.html**

```
<p id="borderimg">Here, the middle sections of the image are repeated to create the
border.</p>
```



# CSS3 –background-image

- CSS3 allows you to add multiple background images for an element, through the **background-image** property.
- The different background images are separated by commas, and the images are stacked on top of each other, where the first image is closest to the viewer.
- **Example:**

**.css**

#example1

```
{
 background-image: url(img_flwr.gif), url(paper.gif);
}
```

#example2

```
{
 background: url(img_flwr.gif) right bottom no-repeat, url(paper.gif) left
 top repeat;
}
```

# CSS3 –background-color

- **RGBA** color values are an extension of RGB color values with an alpha channel - which specifies the opacity for a color.
- An RGBA color value is specified with: **rgba(red, green, blue, alpha)**. The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (fully opaque).
- **Example:**

**.css**

```
#p1 {background-color:rgba(255,0,0,0.4);}
```

**.html**

```
<p id="p1">Red</p>
```

Red



# CSS3 –linear-gradient

- CSS3 gradients let you display smooth transitions between two or more specified colors.
- background: **linear-gradient(direction, color1, color2, ...);**
- **direction** values: **to right** and **to left**
- **Example:**

**.css**

**#grad1**

**{**

background: linear-gradient(red, yellow);

**}**

**.html**

<div id="grad1"></div>



# CSS3: text-shadow and box-shadow

- With CSS3 you can add shadow to text and to elements.
- **Example:**

**.css**

```
div {
 box-shadow: 10px 10px red;
}

P{
 text-shadow: 5px 7px blue;
}
```

# CSS Animation

- CSS **animation** is a proposed module for Cascading Style Sheets that allows designers and developers to add animations by **editing** the CSS code of their websites, instead of uploading GIF or flash images directly.
- We not only reuse the similar CSS animations on different websites easily by copying and pasting the CSS code but also make **lighter** websites with better **compatibility**. With animation tool will make websites/apps as **realistic** as possible.
- An animation lets an element **gradually** change from one style to another.
- You can change as many CSS properties you want, as many times you want.
- To use CSS animation, you must first specify some **keyframes** for the animation.
- Keyframes **hold** what styles the element will have at certain times.



# CSS Animation

## Properties

- **@keyframes** - the animation will gradually change from the current style to the new style at certain times.
- **animation-name** – name of the animation
- **animation-duration** – Time duration for the animation  
Example – [Unit2 CSS animation.html](#)
- Note: The animation-duration property defines how long time an animation should take to complete. If the animation-duration property is not specified, no animation will occur, because the default value is 0s (0 seconds).
- In the example above we have specified when the style will change by using the keywords "from" and "to" (which represents 0% (start) and 100% (complete)).
- It is also possible to use percent. By using percent, you can add as many style changes as you like.

# CSS Animation

## Properties

- **animation-delay** - delay for the start of an animation.
  - The previous example has a 5 seconds delay before starting the animation
  - Negative values are also allowed. If using negative values, the animation will start as if it had already been playing for N seconds.
- **animation-iteration-count** - specifies the number of times an animation should run
  - The previous example will run the animation 2 times before it stops
  - Infinite value also used for animation to continue for ever

# CSS Animation

## Properties

- **animation-direction** - specifies whether an animation should be played forwards, backwards or in alternate cycles

The animation-direction property can have the following values:

- normal - The animation is played as normal (forwards). This is default
- reverse - The animation is played in reverse direction (backwards)
- alternate - The animation is played forwards first, then backwards
- alternate-reverse - The animation is played backwards first, then forwards

The previous example will run the animation in reverse direction (backwards): and uses the value "alternate" to make the animation run forwards first, then backwards

# CSS Animation

## Properties

- **animation-timing-function** - specifies the speed curve of the animation

The animation-timing-function property can have the following values:

- ease - Specifies an animation with a slow start, then fast, then end slowly (this is default)
  - linear - Specifies an animation with the same speed from start to end
  - ease-in - Specifies an animation with a slow start
  - ease-out - Specifies an animation with a slow end
  - ease-in-out - Specifies an animation with a slow start and end
  - cubic-bezier(n,n,n,n) - Lets you define your own values in a cubic-bezier function
- The example - [Unit2\\_CSS\\_animation1.html](#) shows some of the different speed curves that can be used
  - For cubic-Bezier() - [Unit2\\_CSS\\_animation\\_cubic Bezier.html](#)

# CSS Animation

## Properties

- **animation-fill-mode** - CSS animations do not affect an element before the first keyframe is played or after the last keyframe is played. The animation-fill-mode property can override this behavior.

- The animation-fill-mode property specifies a style for the target element when the animation is not playing (before it starts, after it ends, or both).

**The animation-fill-mode property can have the following values:**

- **none** - Default value. Animation will not apply any styles to the element before or after it is executing
- **forwards** - The element will retain the style values that is set by the last keyframe (depends on animation-direction and animation-iteration-count)
- **backwards** - The element will get the style values that is set by the first keyframe (depends on animation-direction), and retain this during the animation-delay period
- **both** - The animation will follow the rules for both forwards and backwards, extending the animation properties in both directions

The Example - [Unit2 CSS animation2.html](#) lets the <div> element get the style values set by the first keyframe before the animation starts (during the animation-delay period)

# CSS Animation

## Properties

- **Animation Shorthand Property**
- The example below uses six of the animation properties

```
div {
 animation-name: example;
 animation-duration: 5s;
 animation-timing-function: linear;
 animation-delay: 2s;
 animation-iteration-count: infinite;
 animation-direction: alternate;
}
```

The same animation effect as above can be achieved by using the shorthand animation property

```
div { animation: example 5s linear 2s infinite alternate; }
```

# CSS Tooltip

- Create a tooltip that appears when the user moves the mouse over an element
- [Example](#) - Unit2\_CSS\_tooltip.html
- HTML: Use a container element (like <div>) and add the "tooltip" class to it. When the user mouse over this <div>, it will show the tooltip text.
- The tooltip text is placed inside an inline element (like <span>) with class="tooltiptext".
- CSS: The tooltip class use position:relative, which is needed to position the tooltip text (position:absolute). Note: See examples below on how to position the tooltip.
- The tooltiptext class holds the actual tooltip text. It is hidden by default, and will be visible on hover (see below). We have also added some basic styles to it: 120px width, black background color, white text color, centered text, and 5px top and bottom padding.
- The CSS border-radius property is used to add rounded corners to the tooltip text.
- The :hover selector is used to show the tooltip text when the user moves the mouse over the <div> with class="tooltip".

# CSS Style Images

## Rounded Images

- Use the border-radius property to create rounded images:
  - [Example](#) - Unit2\_CSS\_style\_image1.html

## Thumbnail Images

- Use the border property to create thumbnail images.
  - [Example](#) - Unit2\_CSS\_style\_image2.html

## Responsive Images

- Responsive images will automatically adjust to fit the size of the screen.
- Resize the browser window to see the effect - [Example](#) - Unit2\_CSS\_style\_image3.html



# CSS Style Images

## Center an Image

- To center an image, set left and right margin to auto and make it into a block element
  - [Example](#) - Unit2\_CSS\_style\_image3.html

## Polaroid Images / Cards

[Example](#) - Unit2\_CSS\_style\_image4.html

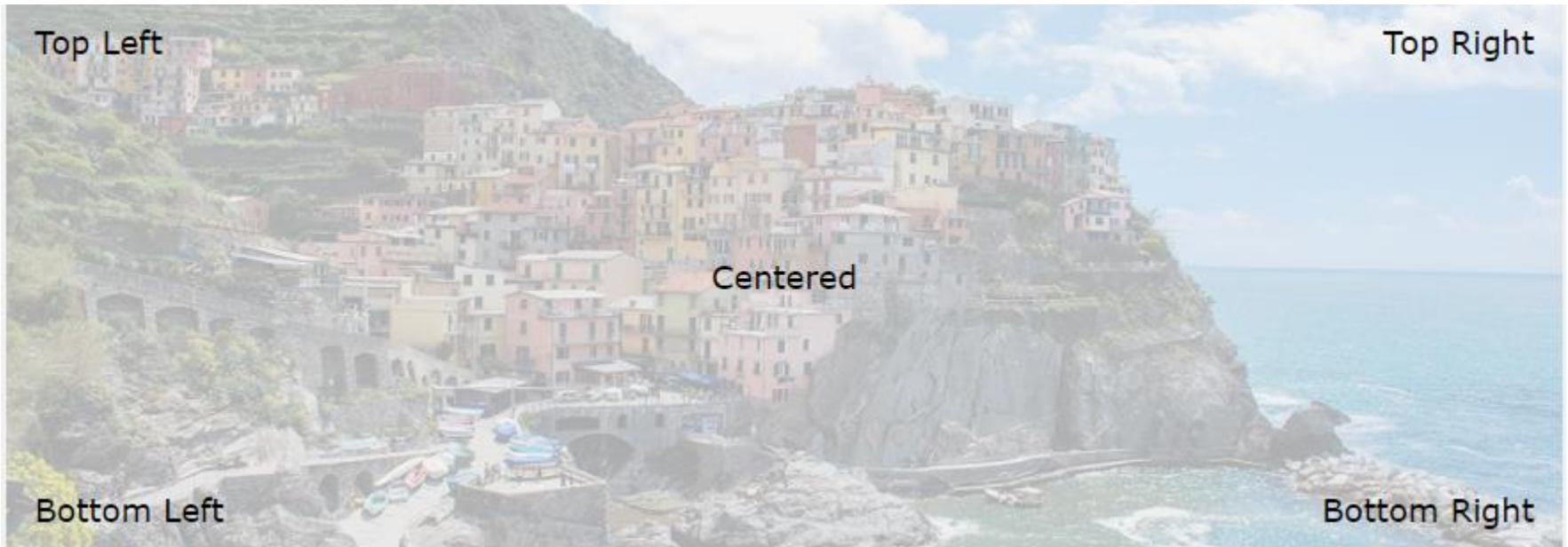
## Transparent Image

- The opacity property can take a value from 0.0 - 1.0. The lower value, the more transparent
  - [Example](#) - Unit2\_CSS\_style\_image5.html

# CSS Style Images

## Image Text

- How to position text in an image — [Example](#) -  
Unit2\_CSS\_style\_image6\_textonimage.html



# CSS Style Images

## Image Filters

- The CSS filter property adds visual effects (like blur and saturation) to an element. – [Example](#) - *Unit2\_CSS\_style\_image7\_filter.html*

## Image Hover Overlay

- Create an overlay effect on hover – [Example](#) - *Unit2\_CSS\_style\_image8\_hoverralay.html*

## Flip an Image

- Move your mouse over the image – [Example](#) - *Unit2\_CSS\_style\_image9\_flipnimage.html*

# CSS Variables

[Example](#) - *Unit2\_CSS\_withoutvar().html*

## The var() Function

- The var() function is used to insert the value of a CSS variable.
- CSS variables have access to the DOM, which means that you can create variables with local or global scope, change the variables with JavaScript, and change the variables based on media queries.
- A good way to use CSS variables is when it comes to the colors of your design. Instead of copy and paste the same colors over and over again, you can place them in variables.

# CSS Variables

## Syntax of the var() Function

- The var() function is used to insert the value of a CSS variable.

The syntax of the var() function is as follows

*var(name, value)*

<b>Value</b>	<b>Description</b>
--------------	--------------------

name	Required. The variable name (must start with two dashes)
------	----------------------------------------------------------

value	Optional. The fallback value (used if the variable is not found)
-------	------------------------------------------------------------------

Note: The variable name must begin with two dashes (--) and it is case sensitive!

# CSS Variables

## How var() Works

- First of all: CSS variables can have a global or local scope.
- Global variables can be accessed/used through the entire document, while local variables can be used only inside the selector where it is declared.
- To create a variable with global scope, declare it inside the :root selector. The :root selector matches the document's root element.
- To create a variable with local scope, declare it inside the selector that is going to use it.

# CSS Variables

## Example

- First, we declare two global variables (--blue and --white). Then, we use the var() function to insert the value of the variables later in the style sheet
  - [Example](#) - *Unit2\_CSS\_var().html*
- To change the blue and white color to a softer blue and white, you just need to change the two variable values
- Advantages of using var() are:
  - makes the code easier to read (more understandable)
  - makes it much easier to change the color values

# CSS Variables

## CSS Overriding Variables

- Overriding Global Variables With Local Variables
- From the previous example we have learned that global variables can be accessed/used through the entire document, while local variables can be used only inside the selector where it is declared.
- Sometimes we want the variables to change only in a specific section of the page.
- Assume we want a different color of bottom border in the Header2 . Then, we can re-declare the --blue variable inside the button selector. When we use var(--blue) inside this selector, it will use the local --blue variable value declared here.
- We see that the local --blue variable will override the global --blue variable for the button elements
- Add a New Local Variable - If a variable is to be used only one single place, we could also have declared a new local variable – [Example](#) - *Unit2\_CSS\_var()\_override.html*



# CSS Variables

## Change Variables With JavaScript

- CSS variables have access to the DOM, which means that you can change them with JavaScript.
- Here is an [example](#) - *Unit2\_CSS\_var()\_javascript.html* of how you can create a script to display and change the --blue variable from the previous example.

# CSS Media Queries

## CSS2 Introduced Media Types

- The `@media` rule, introduced in CSS2, made it possible to define different style rules for different media types.
- Examples: You could have one set of style rules for computer screens, one for printers, one for handheld devices, one for television-type devices, and so on.
- Unfortunately these media types never got a lot of support by devices, other than the print media type.

# CSS Media Queries

## CSS3 Introduced Media Queries

- Media queries in CSS3 extended the CSS2 media types idea: Instead of looking for a type of device, they look at the capability of the device.
- Media queries can be used to check many things, such as:
  - width and height of the viewport
  - width and height of the device
  - orientation (is the tablet/phone in landscape or portrait mode?)
  - resolution
  - Using media queries are a popular technique for delivering a tailored style sheet to desktops, laptops, tablets, and mobile phones (such as iPhone and Android phones).

# CSS Media Queries

## Media Query Syntax

- A media query consists of a media type and can contain one or more expressions, which resolve to either true or false.

```
@media not|only mediatype and (expressions) {
 CSS-Code;
}
```

- The result of the query is true if the specified media type matches the type of device the document is being displayed on and all expressions in the media query are true. When a media query is true, the corresponding style sheet or style rules are applied, following the normal cascading rules.
- Unless you use the not or only operators, the media type is optional, and the all type will be implied.
- You can also have different stylesheets for different media
  - `<link rel="stylesheet" media="mediatype and|not|only (expressions)" href="print.css">`

# CSS Media Queries

## CSS3 Media Types

Value	Description
all	Used for all media type devices
print	Used for printers
screen	Used for computer screens, tablets, smart-phones etc.
speech	Used for screenreaders that "reads" the page out loud

# CSS Media Queries

## Media Queries Simple Examples

- One way to use media queries is to have an alternate CSS section right inside your style sheet.
- The following example changes the background-color to lightgreen if the viewport is 480 pixels wide or wider (if the viewport is less than 480 pixels, the background-color will be pink) – [Example](#) - *Unit2\_CSS\_mediaqueries1.html*
- The following example shows a menu that will float to the left of the page if the viewport is 480 pixels wide or wider (if the viewport is less than 480 pixels, the menu will be on top of the content) – [Example](#) - *Unit2\_CSS\_mediaqueries2.html*
- [Example](#) - *Unit2\_CSS\_mediaqueries3.html*

# CSS - Wildcard Selectors (\*, ^ and \$)

- Now that we are aware of selectors and their importance, why don't we talk about a very specific selector called the wildcard selectors (\*, ^ and \$)?

## Definition

- The Wildcard Selector is used to select or choose various elements at the same time simultaneously. This selector would help in selecting similar types of class designation, name or attribute and make them CSS property usable. \* wildcard can also be referred to as containing a wildcard.

### 1) [attribute\*="str"] Selector

- The first and foremost in the list is [attribute\*="str"] selector. The behavior of this selector is very easy to understand and you can easily put this selector to use. The purpose of this selector is to select only those elements who comprise of specified substring str.
- Now for implementation, let us have a look at the syntax,  
[attribute\*="value"] {    } – [Example](#) - *Unit2\_CSS\_starselector.html*

# CSS - Wildcard Selectors (\*, ^ and \$)

## 2) [attribute^="str"] Selector

- The second selector that you need to learn about is the [attribute^="str"] selector. This selector is something similar to the [attribute\*=" str"]selector as this selector selects those elements whose attributes begin with very specified value str.

[attribute^="str"]{} – [Example](#)-Unit2\_CSS\_carretselector.html

- the elements are selected whose class name starts with clr.



# CSS - Wildcard Selectors (\*, ^ and \$)

## 3) [attribute\$="str"] Selector

- Similar to the first two selectors the purpose of [attribute\$="str"] Selector is to select those elements whose attributes end with very specified value str.

[attribute\$=" str"]{    } - Example

the elements are selected whose class name ends with clr

# CSS Gradients

- CSS gradients let you display smooth transitions between two or more specified colors.
- CSS defines two types of gradients:
  - Linear Gradients (goes down/up/left/right/diagonally)
  - Radial Gradients (defined by their center)
- **CSS Linear Gradients**
  - To create a linear gradient you must define at least two color stops. Color stops are the colors you want to render smooth transitions among. You can also set a starting point and a direction (or an angle) along with the gradient effect.

Syntax:

```
background-image: linear-gradient(direction, color-stop1, color-stop2, ...);
```

Direction - Top to Bottom (this is default) - [Example](#)

# CSS Gradients

- **CSS Radial Gradients**

- A radial gradient is defined by its center.
- To create a radial gradient you must also define at least two color stops.

## Syntax

`background-image: radial-gradient(shape size at position, start-color, ..., last-color);`

- By default, shape is ellipse, size is farthest-corner, and position is center.
- Radial Gradient - Evenly Spaced Color Stops (this is default) -

[Example](#)

# Basic of Framework - Bootstrap

- Twitter Bootstrap is the most popular front end framework in the recent time.
- It is sleek, intuitive, and powerful mobile first front-end framework for faster and easier web development.
- It uses HTML, CSS and Javascript.
- **Bootstrap** was developed by **Mark Otto** and **Jacob Thornton** at Twitter. It was released as an **open source** product in **August 2011** on GitHub.

## Why to Learn Bootstrap?

- **Mobile first approach** – Bootstrap 3, framework consists of Mobile first styles throughout the entire library instead them of in separate files.
- It is **supported** by all popular **browsers**.

# Basic of Framework - Bootstrap

- **Easy to get started** – With just the knowledge of HTML and CSS anyone can get started with Bootstrap. Also the Bootstrap official site has a good documentation.
- **Responsive design** – Bootstrap's responsive CSS adjusts to Desktops, Tablets and Mobiles.
- Provides a clean and uniform solution for building an interface for developers.
- It contains beautiful and functional built-in components which are easy to customize.
- It also provides web based customization.  
And best of all it is an open source.



# Basic of Framework - Bootstrap

## Applications of Bootstrap

- **Scaffolding** – Bootstrap provides a basic structure with Grid System, link styles, and background.
- **CSS** – Bootstrap comes with the feature of global CSS settings, fundamental HTML elements styled and enhanced with extensible classes, and an advanced grid system.
- **Components** – Bootstrap contains over a dozen reusable components built to provide iconography, dropdowns, navigation, alerts, pop-overs, and much more.
- **JavaScript Plugins** – Bootstrap contains over a dozen custom jQuery plugins. You can easily include them all, or one by one.
- **Customize** – You can customize Bootstrap's components, LESS variables, and jQuery plugins to get your very own version.

# Basic of Framework - Bootstrap

## A basic HTML template using Bootstrap

```
<!DOCTYPE html>
<html>
 <head>
 <title>Bootstrap 101 Template</title>
 <meta name = "viewport" content = "width = device-width, initial-scale = 1.0">
 <!-- Bootstrap -->
 <link href = "css/bootstrap.min.css" rel = "stylesheet">
 </head>

 <body>
 <h1>Hello, world!</h1>

 <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
 <script src = "https://code.jquery.com/jquery.js"></script>

 <!-- Include all compiled plugins (below), or include individual files as needed -->
 <script src = "js/bootstrap.min.js"></script>

 </body>
</html>
```

# Basic of Framework - Bootstrap

- Here you can see the **jquery.js**, **bootstrap.min.js** and **bootstrap.min.css** files that are included to make a normal HTML file to the Bootstrapped Template. Just make sure to include jQuery library before you include Bootstrap library



# Basic of Framework - Bootstrap

- **What is a Grid?**

- In graphic design, a grid is a structure (usually two-dimensional) made up of a series of intersecting straight (vertical, horizontal) lines used to structure the content. It is widely used to design layout and content structure in print design. In web design, it is a very effective method to create a consistent layout rapidly and effectively using HTML and CSS. (wikipedia)

- To put in simple words, grids in web design organise and structure content, makes the websites easy to scan and reduces the cognitive load on users.

- **What is Bootstrap Grid System?**

- Bootstrap includes a responsive, mobile first fluid grid system that appropriately scales up to 12 columns as the device or viewport size increases. It includes predefined classes for easy layout options, as well as powerful mixins for generating more semantic layouts.

# Basic of Framework - Bootstrap

- Bootstrap 3 is mobile first in the sense that the code for Bootstrap now starts by targeting smaller screens like mobile devices, tablets, and then “expands” components and grids for larger screens such as laptops, desktops.

## Mobile First Strategy

- Content
  - Determine what is most important.
- Layout
  - Design to smaller widths first.
  - Base CSS address mobile device first; media queries address for tablet, desktops.
- Progressive Enhancement
  - Add elements as screen size increases.

# Basic of Framework - Bootstrap

## Working of Bootstrap Grid System

- Grid systems are used for creating page layouts through a series of rows and columns that house your content.

## How the Bootstrap grid system works

- Rows must be placed within a **.container** class for proper alignment and padding.
- Use **rows** to create horizontal groups of columns.
- Content should be placed within the **columns**, and only columns may be the immediate children of rows.
- Predefined grid classes like **.row** and **.col-xs-4** are available for quickly making grid layouts. LESS mixins can also be used for more semantic layouts.
- Columns create **gutters** (gaps between column content) via padding. That padding is offset in rows for the first and the last column via negative margin on **.rows**.
- **Grid** columns are created by specifying the number of twelve available columns you wish to span. For example, three equal columns would use three **.col-xs-4**.

# Basic of Framework - Bootstrap

## Media Queries

Media query is a really fancy term for "conditional CSS rule". It simply applies some CSS, based on certain conditions set forth. If those conditions are met, the style is applied.

Media Queries in Bootstrap allow you to move, show and hide content based on the viewport size. Following media queries are used in LESS files to create the key breakpoints in the Bootstrap grid system.

```
/* Extra small devices (phones, less than 768px) */
/* No media query since this is the default in Bootstrap */
```

```
/* Small devices (tablets, 768px and up) */
@media (min-width: @screen-sm-min) { ... }
```

```
/* Medium devices (desktops, 992px and up) */
@media (min-width: @screen-md-min) { ... }
```

```
/* Large devices (large desktops, 1200px and up) */
@media (min-width: @screen-lg-min) { ... }
```

# Basic of Framework - Bootstrap

## Media Queries

Occasionally these are expanded to include a max-width to limit CSS to a narrower set of devices.

```
@media (max-width: @screen-xs-max) { ... }
```

```
@media (min-width: @screen-sm-min) and (max-width: @screen-sm-max) { ... }
```

```
@media (min-width: @screen-md-min) and (max-width: @screen-md-max) { ... }
```

```
@media (min-width: @screen-lg-min) { ... }
```

Media queries have two parts, a device specification and then a size rule.

In the above case, the following rule is set –

```
@media (min-width: @screen-sm-min) and (max-width: @screen-sm-max) { ... }
```

For all devices no matter what kind with min-width: @screen-sm-min if the width of the screen gets smaller than @screen-sm-max, then do something.

# Basic of Framework - Bootstrap

## Grid options

The following table summarizes aspects of how Bootstrap grid system works across multiple devices

	Extra small devices Phones (<768px)	Small devices Tablets (≥768px)	Medium devices Desktops (≥992px)	Large devices Desktops (≥1200px)
Grid behavior	Horizontal at all times	Collapsed to start, horizontal above breakpoints	Collapsed to start, horizontal above breakpoints	Collapsed to start, horizontal above breakpoints
Max container width	None (auto)	750px	970px	1170px
Class prefix	.col-xs-	.col-sm-	.col-md-	.col-lg-
# of columns	12	12	12	12
Max column width	Auto	60px	78px	95px
Gutter width	30px (15px on each side of a column)	30px (15px on each side of a column)	30px (15px on each side of a column)	30px (15px on each side of a column)
Nestable	Yes	Yes	Yes	Yes
Offsets	Yes	Yes	Yes	Yes
Column ordering	Yes	Yes	Yes	Yes

# Basic of Framework - Bootstrap

## Basic Grid Structure

Following is basic structure of Bootstrap grid –

```
<div class = "container">

 <div class = "row">
 <div class = "col-*-*"></div>
 <div class = "col-*-*"></div>
 </div>

 <div class = "row">...</div>

</div>

<div class = "container">

</div>
```

Let us see some simple grid examples –

- [Example – Stacked-to-horizontal](#)
- [Example – Medium and Large Device](#)
- [Example – Mobile, tablet, desktops](#)