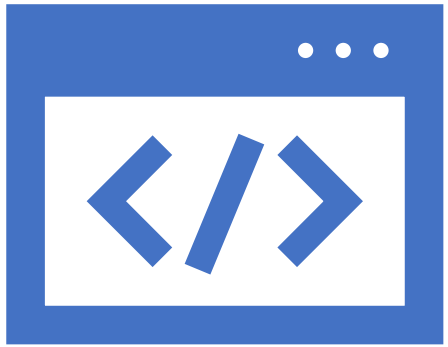
WEB DESIGNING



MODULE 2

Dynamic HTML

- **DHTML** stands for **Dynamic Hypertext Markup language** i.e, **Dynamic HTML**.

- Dynamic HTML is not a markup or programming language but it is a term that combines the features of

various web development technologies for creating the web pages dynamic and interactive.
The DHTML application was introduced by Microsoft with the release of the 4th version of IE (Internet Explorer) in 1997.

- The DHTML is based on the properties of the HTML, JavaScript, CSS, and DOM which

helps in making dynamic content.

- It is the combination of HTML, CSS, JS, and DOM. The DHTML make use of Dynamic

object model to make changes in settings and also in properties and methods.

- DHTML is used to create interactive and animated web pages that are generated in real-time,

also known as dynamic web pages so that when such a page is accessed, the code within the page is analyzed on the web server and the resulting HTML is sent to the client's web browser.

Features:

1. Tags and their properties can be changed using DHTML.
2. It is used for real-time positioning.
3. Dynamic fonts can be generated using DHTML.
4. It is also used for data binding.
5. It makes a webpage dynamic and be used to create animations, games, applications along
with providing new ways of navigating through websites.
6. The functionality of a webpage is enhanced due to the usage of low-bandwidth effect by
DHTML.
7. DHTML also facilitates the use of methods, events, properties, and codes.

Uses:

1. It is used for designing the animated and interactive web pages that are developed in realtime.
2. DHTML helps users by animating the text and images in their documents.
3. It allows the authors for adding the effects on their pages.
4. It also allows the page authors for including the drop-down menus or rollover buttons.
5. This term is also used to create various browser-based action games.
6. It is also used to add the ticker on various websites, which needs to refresh their content automatically.

Advantage:

1. Sizes of the files are compact in compared to other interactional media like Flash or Shockwave, and it downloads faster.
2. It is supported by big browser manufacturers like Microsoft and Netscape.
3. Highly flexible and easy to make changes.
4. Viewer requires no extra plug-ins for browsing through the webpage that uses DHTML, they do not need any extra requirements or special software to view it.
5. User time is saved by sending less number of requests to the server. As it is possible to modify and replace elements even after a page is loaded, it is not required to create separate pages for changing styles which in turn saves time in building pages and also reduces the number of requests that are sent to the server.
6. It has more advanced functionality than a static HTML. It is capable of holding more content on the web page at the same time.

Disadvantage:

1. It is not supported by all the browsers. It is supported only by recent browsers such as Netscape 6, IE 5.5, and Opera 5 like browsers.
2. Learning of DHTML requires a lot of pre-requisites languages such as HTML, CSS, JS, etc should be known to the designer before starting with DHTML which is a long and time-consuming in itself.
3. Implementations of different browsers are different. So if it worked in one browser, it might not necessarily work the same way in another browser.
4. Even after being great with functionality, DHTML requires a few tools and utilities that are some expensive. For example, the DHTML text editor, Dreamweaver. Along with it the improvement cost of transferring from HTML to DHTML makes cost rise much higher.

Difference between HTML and DHTML:

1. HTML is a markup language while DHTML is a collection of technologies.
2. HTML is used to create static webpages while DHTML is capable of creating dynamic webpages.
3. DHTML is used to create animations and dynamic menus but HTML not used.
4. HTML sites are slow upon client-side technologies whereas DHTML sites are comparatively faster.
5. Web pages created using HTML are rather simple and have no styling as it uses only one language whereas DHTML uses HTML, CSS, and JavaScript which results in a much better and way more presentable webpage.
6. HTML cannot be used as server side code but DHTML used as server side code.
7. DHTML needs database connectivity but not in case of HTML.
8. Files in HTML are stored using .htm or .html extension while DHTML uses .dhtm extension.
9. HTML requires no processing from the browser but DHTML does.

Components:

DHTML consists of the following four components or languages:

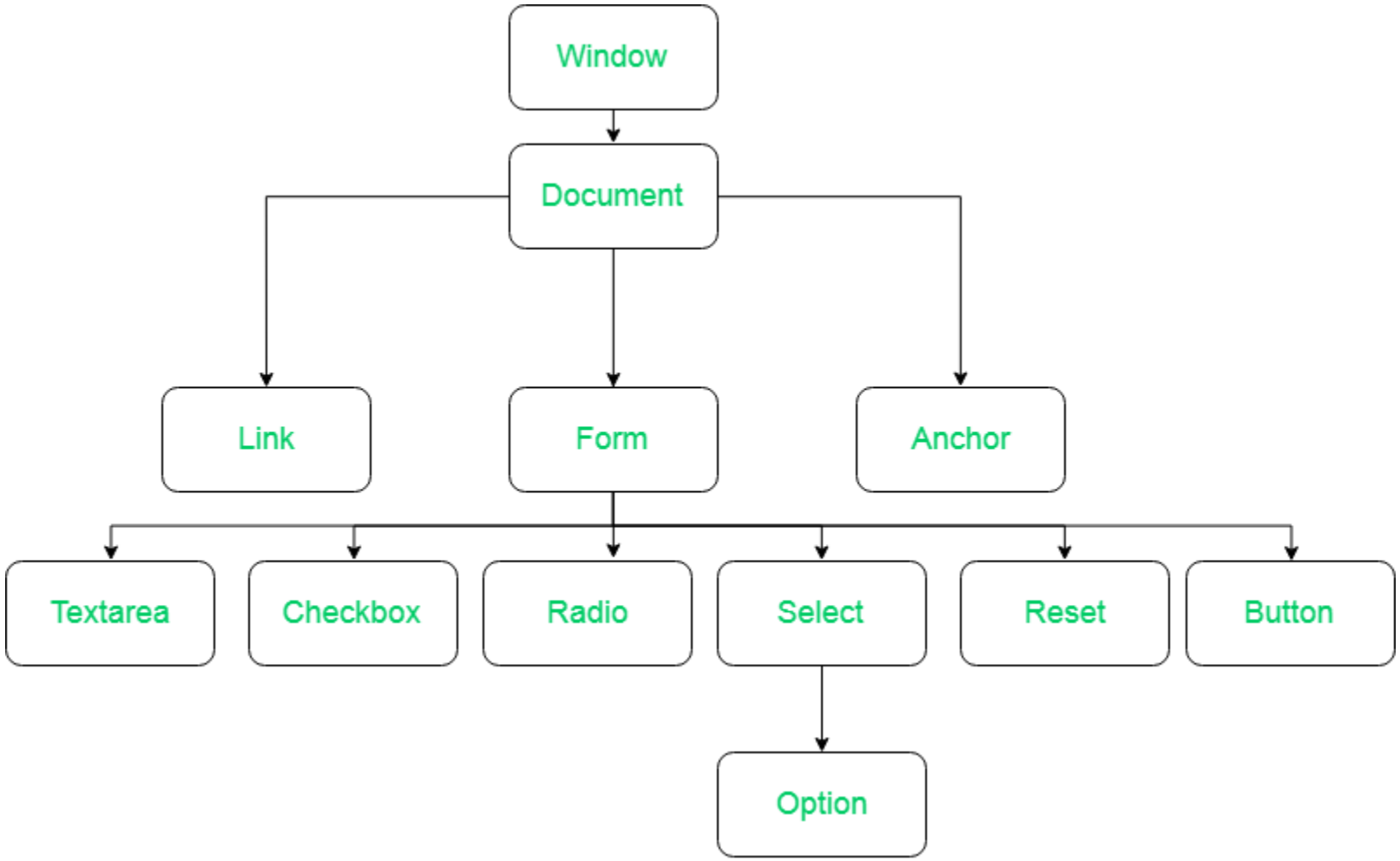
1. **HTML 4.0:**HTML is a client-side markup language, which is a core component of the DHTML. It defines the structure of a web page with various defined basic elements or tags.
2. **CSS:**CSS stands for Cascading Style Sheet, which allows the web users or developers for controlling the style and layout of the HTML elements on the web pages.
3. **JavaScript:** JavaScript is a scripting language which is done on a client-side. The various browser supports JavaScript technology. DHTML uses the JavaScript technology for accessing, controlling, and manipulating the HTML elements. The statements in JavaScript are the commands which tell the browser for performing an action.
4. **DOM:** DOM is the document object model. It is a w3c standard, which is a standard interface of programming for HTML. It is mainly used for defining the objects and properties of all elements in HTML.

The Document Object Model (DOM) is a ***programming interface*** for **HTML(HyperText Markup Language)** and **XML**(Extensible markup language) documents. It defines the **logical structure** of documents and the way a document is accessed and manipulated.

DOM is a way to represent the webpage in a structured hierarchical way so that it will become easier for programmers and users to glide through the document. With DOM, we can easily access and manipulate tags, IDs, classes, Attributes, or Elements of HTML using commands or methods provided by the Document object. Using DOM, the JavaScript gets access to HTML as well as CSS of the web page and can also add behavior to the HTML elements. so basically **Document Object Model is an API that represents and interacts with HTML or XML documents.**

The **document object** represents the whole html document.
When html document is loaded in the browser, it becomes a document object. It is the **root element** that represents the html document. It has properties and methods. By the help of document object, we can add dynamic content to our web page.
As mentioned earlier, it is the object of window.

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


Window
Document
Link
Form
Anchor
Textarea
Checkbox
Radio
Select
Reset
Button
Option

- **Window object**: Window Object is object of the browser which is always at top of the hierarchy. It is like an API that is used to set and access all the properties and methods of the browser. It is automatically created by the browser.
- **Document object:** When an HTML document is loaded into a window, it becomes a document object. The 'document' object has various properties that refer to other objects which allow access to and modification of the content of the web page. If there is a need to access any element in an HTML page, we always start with accessing the 'document' object. Document object is property of window object.
- **Form Object:** It is represented by ***form*** tags.
- **Link object:** It is represented by ***link*** tags.
- **Anchor object:** It is represented by ***a href*** tags.
- **Form Control Elements:**: Form can have many control elements such as text fields, buttons, radio buttons, checkboxes, etc.

The important methods of document object are as follows:

Method	Description
write("string")	writes the given string on the doucment.
writeln("string")	writes the given string on the doucment with newline character at the end.
getElementById()	returns the element having the given id value.
getElementsByName()	returns all the elements having the given name value.
getElementsByTagName()	returns all the elements having the given tag name.
getElementsByClassName()	returns all the elements having the given class name.

Accessing field value by document object

In this example, we are going to get the value of input text by user. Here, we are using **document.form1.name.value** to get the value of name field.

Here, **document** is the root element that represents the html document.

form1 is the name of the form.

name is the attribute name of the input text.

value is the property, that returns the value of the input text.

Let's see the simple example of document object that prints name with welcome message.

```
<script type="text/javascript">
function printvalue(){
var name=document.form1.name.value;
alert("Welcome: "+name);
}
</script>
<form name="form1">
Enter Name:<input type="text" name="name"/>
<input type="button" onclick="printvalue()" value="print name"/>
</form>
```

Enter Name: Print name

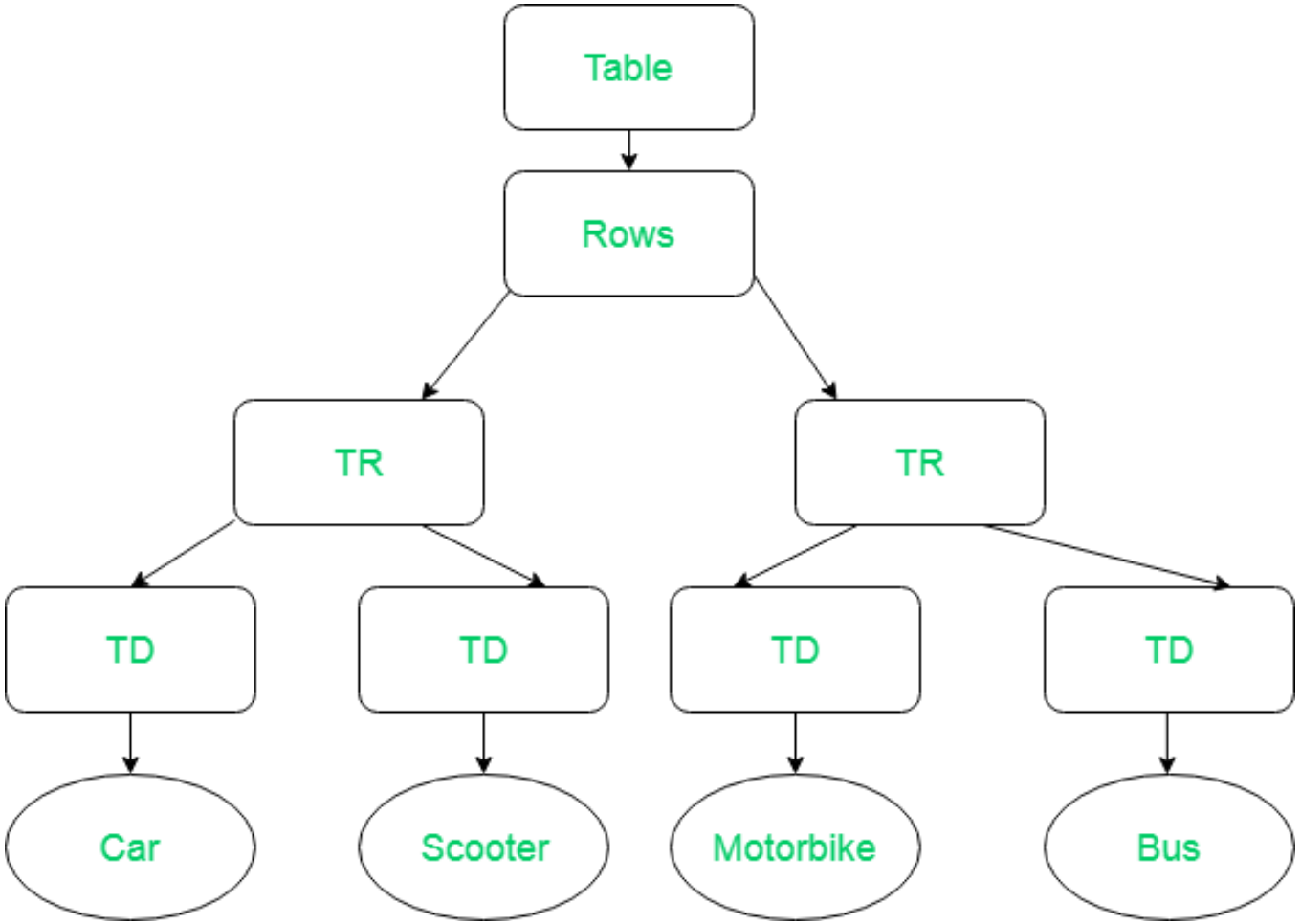
Why DOM is required?

HTML is used to **structure** the web pages and Javascript is used to add **behavior** to our web pages. When an HTML file is loaded into the browser, the javascript can not understand the HTML document directly. So, a corresponding document is created(DOM). **DOM is basically the representation of the same HTML document but in a different format with the use of objects**. Javascript interprets DOM easily i.e javascript can not understand the tags(<h1>H</h1>) in HTML document but can understand object h1 in DOM. Now, Javascript can access each of the objects (h1, p, etc) by using different functions.

Documents are modeled using objects, and the model includes not only the structure of a document but also the behavior of a document and the objects of which it is composed like tag elements with attributes in HTML.

Example for: The representation of HTML elements in the tree structure.

```
<table>
    <ROWS>
        <tr>
            <td>Car</td>
            <td>Scooter</td>
        </tr>
        <tr>
            <td>MotorBike</td>
            <td>Bus</td>
        </tr>
    </ROWS>
</table>
```



•DOM describes XML and HTML documents as objects.
•The Document Object Model is not represented by a **set of data structures**; it is an interface that specifies object representation.
•The Document Object Model does not show the **criticality of objects** in documents
• The window interface is home to a variety of functions, namespaces, objects, and constructors which are not necessarily directly associated with the concept of a user interface window. However, the window interface is a suitable place to include these items that need to be globally available.
•A window for a given document can be obtained using the **window.defaultView** property**.**

Global Event Attributes

HTML has the ability to let events trigger actions in a browser, like starting a JavaScript when a user clicks on an element. Below are the global event attributes that can be added to HTML elements to define event actions.

Window Event Attributes

Events triggered for the window object (applies to the <body> tag):

Attribute	Value	Description
onafterprint	*script*	Script to be run after the document is printed
onbeforeprint	*script*	Script to be run before the document is printed
onbeforeunload	*script*	Script to be run when the document is about to be unloaded
onerror	*script*	Script to be run when an error occurs
onhashchange	*script*	Script to be run when there has been changes to the anchor part of the a URL
onload	*script*	Fires after the page is finished loading
onmessage	*script*	Script to be run when the message is triggered
onoffline	*script*	Script to be run when the browser starts to work offline
ononline	*script*	Script to be run when the browser starts to work online
onpagehide	*script*	Script to be run when a user navigates away from a page
onpageshow	*script*	Script to be run when a user navigates to a page
onpopstate	*script*	Script to be run when the window's history changes
onresize	*script*	Fires when the browser window is resized
onstorage	*script*	Script to be run when a Web Storage area is updated
onunload	*script*	Fires once a page has unloaded (or the browser window has been closed)

Form Events

Events triggered by actions inside a HTML form (applies to almost all HTML elements, but is most used in form elements):

Attribute	Value	Description
onblur	*script*	Fires the moment that the element loses focus
onchange	*script*	Fires the moment when the value of the element is changed
oncontextmenu	*script*	Script to be run when a context menu is triggered
onfocus	*script*	Fires the moment when the element gets focus
oninput	*script*	Script to be run when an element gets user input
oninvalid	*script*	Script to be run when an element is invalid
onreset	*script*	Fires when the Reset button in a form is clicked
onsearch	*script*	Fires when the user writes something in a search field (for <input="search">)
onselect	*script*	Fires after some text has been selected in an element
onsubmit	*script*	Fires when a form is submitted

Keyboard Events

Attribute	Value	Description
onkeydown	*script*	Fires when a user is pressing a key
onkeypress	*script*	Fires when a user presses a key
onkeyup	*script*	Fires when a user releases a key

Mouse Events

Attribute	Value	Description
onclick	*script*	Fires on a mouse click on the element
ondblclick	*script*	Fires on a mouse double-click on the element
onmousedown	*script*	Fires when a mouse button is pressed down on an element
onmousemove	*script*	Fires when the mouse pointer is moving while it is over an element
onmouseout	*script*	Fires when the mouse pointer moves out of an element
onmouseover	*script*	Fires when the mouse pointer moves over an element
onmouseup	*script*	Fires when a mouse button is released over an element
onmousewheel	*script*	Deprecated. Use the onwheel attribute instead
onwheel	*script*	Fires when the mouse wheel rolls up or down over an element

STYLE SHEETS

- A style sheet is a file or form that is used in word processing and desktop publishing to define the layout style of a document. A style sheet contains the specifications of a document’s layout, such as the page size, margins, fonts and font sizes. In modern word processors such as Microsoft Word, a style sheet is known as a template. The most well-known form of style sheet is the Cascading Style Sheet (CSS), which is used for styling Web pages.

- A style sheet is a file that can accompany the main file of any web page. With the css extension, **it is responsible for establishing the different aspects related to the format and design of the web**, specifying the parameters that define the tags used in the source code.
- In this file where information of the style is stored (type of font to be used, the colors, its size, the spacing between paragraphs, the dimensions of the different titles ...). This data collection results in a **unified style across all sections and pages** of a website. In this way a homogeneous and characteristic image is achieved.

- CSS stands for Cascading Style Sheets.
- CSS saves a lot of work. It can control the layout of multiple web pages all at once.
- Cascading Style Sheets (CSS) is used to format the layout of a webpage.
- With CSS, you can control the color, font, the size of text, the spacing between elements, how elements are positioned and laid out, what background images or background colors are to be used, different displays for different devices and screen sizes, and much more!

Tip: The word **cascading** means that a style applied to a parent element will also apply to all children elements within the parent. So, if you set the color of the body text to "blue", all headings, paragraphs, and other text elements within the body will also get the same color (unless you specify something else)!

Why use CSS

1) Solves a big problem

Before CSS, tags like font, color, background style, element alignments, border and size had to be repeated on every web page. This was a very long process. For example: If you are developing a large website where fonts and color information are added on every single page, it will be become a long and expensive process. CSS was created to solve this problem. It was a W3C recommendation.

2) Saves a lot of time

CSS style definitions are saved in external CSS files so it is possible to change the entire website by changing just one file.

3) Provide more attributes

CSS provides more detailed attributes than plain HTML to define the look and feel of the website.

CSS can be added to HTML documents in 3 ways:

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

The most common way to add CSS, is to keep the styles in external CSS files.

Inline CSS

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

The following example sets the color of the <h1> element to blue , and the text color of the <p> element to red:

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

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

Internal CSS

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

The following example sets the text color of ALL the <h1> elements (on that page) to blue, and the text color of ALL the <p> elements to red. In addition, the page will be displayed with a “yellow" background color:

```
<!DOCTYPE html>
<html>
<head>
<style>
   body{background-color:yellow;}
   h1{color:blue;}
   p{color:red;}
</style>
</head>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

External CSS

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

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

The external style sheet can be written in any text editor. The file must not contain any HTML code, and must be saved with a .css extension.
Here is what the "styles.css" file looks like:

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

CSS Comments

- Comments are used to explain the code, and may help when you edit the source code later.
- Comments are ignored by browsers.
- A CSS comment is placed inside the <style> tag, and starts with /* and ends with */.

```
/* This is a single-line comment */
p {
  color: red;
}
```

- you learned that you can add comments to your HTML source by using the <!--…….-->

style sheet properties

1. CSS Background (color,image,position,size….)
2. CSS Text (color,alignment,spacing….)
3. CSS Font (style,family,weight…)
4. CSS Border (width,style,color…)
5. CSS Outline (style,color,width….)
6. CSS Margin (top,right,bottom,left)
7. CSS Padding (top,right,bottom,left)
8. CSS List (style-type,style-image,style-position…)
9. CSS Table (border,width,height,text-align…)

CSS background Property

The background property is a shorthand property for:
•background-color
•background-image
•background-position
•background-size
•background-repeat
•background-origin
•background-clip
•background-attachment

•Property Values

Value	Description
scroll	The background image will scroll with the page. This is default
fixed	The background image will not scroll with the page
local	The background image will scroll with the element's contents
initial	Sets this property to its default value. Read about *initial*
inherit	Inherits this property from its parent element. Read about *inherit*

The position property specifies the type of positioning method used for an element (static, relative, fixed, absolute or sticky).

The position Property

The `position` property specifies the type of positioning method used for an element.
There are five different position values:

- `static`
- `relative`
- `fixed`
- `absolute`
- `sticky`

Elements are then positioned using the top, bottom, left, and right properties. However, these properties will not work unless the `position` property is set first. They also work differently depending on the position value.

Value	Description	Demo
static	Default value. Elements render in order, as they appear in the document flow	Play it »
absolute	The element is positioned relative to its first positioned (not static) ancestor element	Play it »
fixed	The element is positioned relative to the browser window	Play it »
relative	The element is positioned relative to its normal position, so "left:20px" adds 20 pixels to the element's LEFT position	Play it »
sticky	The element is positioned based on the user's scroll positionA sticky element toggles between relative and fixed, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like position:fixed).	