UNIT - I: HTML & CSS

Introduction to HTML & HTML 5

HTML stands for Hyper Text Markup Language used to design the web pages. Hypertext defines the link between the web pages. Markup Language is a tag-based system that produces formatted, annotated, and human-readable results as shown in Fig.1.0. An HTML file can be created by using a simple text editor viz notepad, textpad, Eclipse IDE editor. HTML file must have an extension .htm or .html. The communication on the web happens through Hypertext Transfer Protocol (HTTP). HTML is interpreted by browser. It is a browser/platform independent. HTML is not a Case Sensitive Language because, during parsing, all HTML elements are converted to lowercase first. It acts as a medium for Graphical User Interface (GUI).

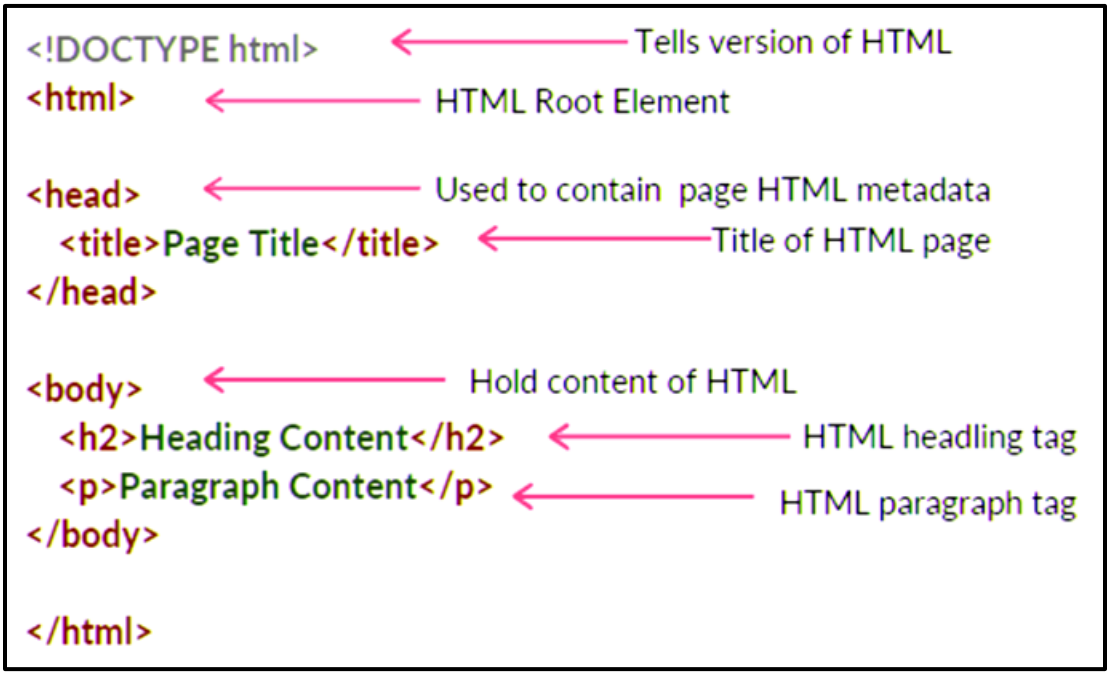


Figure 1.0: Structure of HTML

Tim Berners-Lee, a physicist at the CERN research institute in Switzerland invented HTML in 1991. This first version consisted of 18 HTML tags. Now, there are currently about 140 HTML tags, although not all of them are supported by modern browsers.

Table 1.0: Versions of HTML

S.No.	HTML Versions	Description
1	HTML 1.0	1991 - Tim Berners-Lee invents HTML 1.0. 1993 is released.
2	HTML 2.0	1995 is published it contains the features of HTML 1.0 plus new features.
3	HTML 3.0	1997 - Dave Raggett introduced a fresh draft on HTML, which improved new features of HTML and gave more powerful characteristics for webmasters in designing websites.
4	HTML 4.0	1999 - The widely-used comes out. It is very successful.
5	HTML 5.0	2014 is released and used worldwide. It is said to be the extended version of HTML 4.01 which was published in 2012.

HTML 5 is the fifth and current version of HTML enriched with new elements and attributes. It is a standard specification developed by Web Hypertext Application Technology Working Group (WHATWG) and World Wide Web Consortium (W3C). The main objectives are,

- To define robust error handling.
- To ensure backwards compatibility.
- To develop new features.
- Reduce the need of external plug-in in the web browsers.

HTML 5 has improved the markup available for documents and has introduced Application Programming Interfaces (API) and Document Object Model (DOM). The latest version of Apple Safari, Google Chrome, Mozilla Firefox and Opera supports HTML5 features. The mobile web browsers that come pre-installed on iPhones, iPads, and Android phones all have excellent support for HTML5. The earlier versions of HTML required third party plug-ins such as, Adobe Flash, Microsoft Silverlight and Google Gears to play video. Whereas HTML5 includes the features as,

- **New Semantic Elements** like <header>, <footer>, and <section>.
- **Attributes of Form 2.0** new attributes have been introduced for <input> tag like number, date, time, calendar and range.
- **Persistent Local Storage** is achieved without resorting to third-party plugins.

- **WebSocket** a next-generation bidirectional communication technology for web applications.
- **Server-Sent Events** starts flow from web server to the web browsers called as Server-Sent Events (SSE).
- **Canvas** a two-dimensional drawing surface is supported with JavaScript.
- **Audio & Video** to be embedded in the webpages without resorting to third-party plugins.
- **Geolocation** visitors can select to share their physical location using web application.
- **Microdata** helps to create the own vocabularies beyond HTML5 and extend the web pages with custom semantics.
- **Drag and drop** the items from one location to another location on the same webpage.

Table 1.1: HTML 4 versus HTML 5

Parameters	HTML 4	HTML 5
Structure	It uses common structure like header and footer	It uses new structure like drag and drop
DOCTYPE declaration	Is very lengthy and refers to an external resource	Is simple and in one line
Applets display	Applet tag used to display applets was removed	Object tag used to display applet type
Inaccurate syntax handling	Cannot handle	Can handle
Acronym tag	Removed	A new tag <abbr> was introduced
Compatibility	Compatible with almost all browsers	No compatible with all browsers
Multimedia supporting tags	Not present	Present
Storage	Browser cache can be used as temporary storage	Application cache, Web SQL DB and Web storage can be used as client storage.
Use of cookies	Yes	No. Local storage instead of cookies.

Drawing of shapes	Not possible	Possible
2D drawing	Flash is required	Flash not required

Advantages

- All browsers supported.
- More devices friendly.
- Easy to use and implement.
- HTML 5 in integration with CSS, JavaScript, etc. can help build beautiful websites.

Disadvantages

- Long codes have to be written which is time consuming.
- Only modern browsers support it.

Semantic Elements

Semantic HTML elements are those that clearly describe their meaning in a human - and machine-readable way. Elements such as <header>, <footer> and <article> are all considered semantic because they accurately describe the purpose of the element and the type of content that is inside them.

- **Examples of non-semantic elements:** and - Tells nothing about its content.
- **Examples of semantic elements:** <form>, <table>, and <article> - Clearly defines its content.

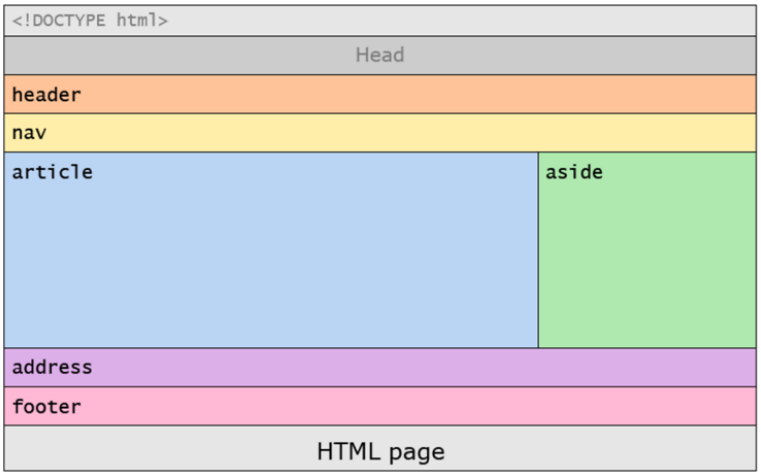


Figure 1.1: Structure of an HTML5 Web page

In HTML5 there are some semantic elements as shown in Fig.1.1. used to define different parts of a web page.

Page Structured Elements

- **<article> -** it specifies independent, self-contained content. An article should make sense on its own and it should be possible to distribute it independently from the rest of the web site. It might be used in Forum posts, Blog posts, User comments.
- **<aside> -** as a sidebar to the main content. It is intended for content that is not part of the flow of the text in which it appears.
- **<details> -** defines additional details that the user can view or hide.
- **<figure> -** it specifies self-contained content, like illustrations, diagrams, photos, code listings etc.
- **<figcaption> -** defines a caption for a <figure> element.
- **<footer> -** is generally found at the bottom of a document, a section, or an article. Just like the <header> the content is generally meta information, such as author details, legal information, and/or links to related information. It is also valid to include <section> elements within footer.
- **<header> -** It is found at the top of a document, a section, or an article and usually contains the main heading and some navigation and search tools.
- **<main> -** specifies the main content of a document.
- **<mark> -** defines the marked / highlighted text.
- **<nav> -** defines a set of navigation links.
- **<section> -** is a confined grouping of content, typically with a heading. Example: A web page could normally be split into sections for introduction, content, and contact information.
- **<summary> -** defines a visible heading for a <details> element.
- **<time> -** used to display the human-readable date / time.

Global attributes

HTML5 defines few attributes that are common to all HTML tags / elements. An attribute of any HTML tag is nothing but a part of it which can be used to add more information to any HTML tag. Example: In an anchor tag as shown in Fig.1.2 href is an attribute which is

used to provide the URL to which the text will be hyperlinked. Similarly, HTML tags can have global attributes to provide additional identification, information like CSS style classes, or some standard properties hidden (to hide HTML element), language, etc.

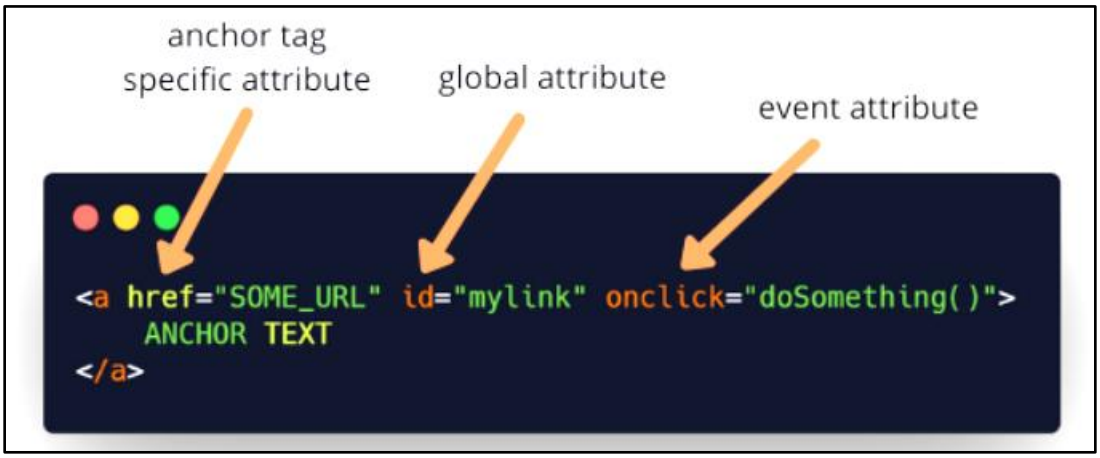


Figure 1.2: Identification of Specific, Global & Event attribute

- Any HTML tag can have multiple attributes.
- Global attributes can be used with all tags; though on some HTML tags they do not have any effect.
- There are some exceptions too where these attributes are not relevant, some of the exceptional tags that do not support global attributes are <base> tag, <script> tag, <title> tag, etc.
- Global attributes can even be used with the tags that are not specified as HTML standard tags; i.e. the non-standard tags also permit global attributes. Using these non-standard elements in a document indicates that a document is no longer HTML5 compliant.

Table 1.2: List of Global attributes

Attribute	Value	Description
accesskey	shortcut key	Specifies a keyboard shortcut to activate or focus the element.
class	Classname	Assigns a class name or space-separated list of class names to an element.

contenteditable	true false	Indicates whether the content of an element is editable by the user or not.
contextmenu	menu-id	Specifies a context menu for an element. A context menu is a menu that appears when the user clicks the right mouse button on the element.
data-*	Data	Specified on any HTML element, to store custom data specific to the page.
dir	ltr rtl	Specifies the base direction of directionality of the element's text.
draggable	true false	Specifies whether an element is draggable or not.
dropzone	copy move link	Specifies whether the dragged data is copied, moved, or linked, when dropped.
hidden	Hidden	Indicates that the element is not yet, or is no longer, relevant.
id	Name	Specifies a unique identifier (ID) for an element which must be unique in the whole document.
lang	language-code	Specifies the primary language for the element's text content.
spellcheck	true false	Specifies whether the element may be checked for spelling errors or not.
style	Style	Specifies inline style information for an element.
tabindex	Number	Specifies the tabbing order of an element.
title	Text	Provides advisory information related to the element. It would be appropriate for a tooltip.
translate	yes no	Specifies whether the text content of an element should be translated or not.
xml:lang	language-code	Specifies the primary language for the element's text content, in XHTML documents.

Basic Tags

Table 1.3: Basic Tags in HTML5

Tag	Description	Attributes
<!-- -->	**comment:** comments are displayed in code only. Tag contents are not rendered in the browser.	none
<!DOCTYPE>	**document type:** defines which specification the document follows.	None
<html>	**html document:** root of an HTML document.	manifest: a URL to the address of the document's application cache manifest
<head>	**head element:** contains information about the document.	none
<h1> to <h6>	**headers (1-6):** represent headings for their sections. elements have a rank given by the number in their name.	global attributes**
<body>	**body element:** main content of the document.	global attributes**
<p>	**paragraph:** creates a paragraph	global attributes**
<pre>	**preformatted text:** represents a block of preformatted text.	global attributes**
 	break: inserts a single line break.	global attributes**

Table 1.4: Link Tags in HTML5

Tag	Description	Attributes
<a>	**anchor:** used to provide a link to another web resource.	**href:** destination resource of the hyperlink. **hreflang:** gives the language of the linked resource. **media:** describes for which media the target document was designed. **ping:** gives the URLs of the resources that are interested in being notified if the user follows the hyperlink.

		rel: relationship between the document containing the hyperlink and the destination resource [alternate | archives | author | bookmark | contact | external | feed | first | help | icon | index | last | license | next | nofollow | noreferrer | pingback | prefetch | prev |search | stylesheet | sidebar | tag | up]. **target:** gives the name of the browsing context that will be used [_blank | _parent | _self | _top]. **type:** gives the MIME type of the linked resource.
<link>	**resource link:** allows authors to link their document to other resources.	**href:** destination resource of the hyperlink. **rel:** relationship between the document containing the hyperlink and the destination resource [alternate | archives | author | bookmark | contact | external | feed | first | help | icon | index | last | license | next | nofollow | noreferrer | pingback | prefetch | prev |search | stylesheet | sidebar | tag | up]. **media:** describes for which media the target document was designed. **hreflang:** gives the language of the linked resource type: gives the MIME type of the linked resource. **sizes:** gives the sizes of icons for visual media.
<nav>	**navigation element:** section of a page that links to other pages or to parts within the page: a section with navigation links.	global attributes**

Table 1.5: Image Tags in HTML5

Tag	Description	Attributes
<img>	image: represents an image	**alt:** text to display if the image can not **src:** a URL to the image file.

		usemap: name of the map to use for the **image ismap:** provides access to a server-side image map height: height of the image in pixels. **width:** width of the image in pixels.
<figure>	**figure element:** some flow content, optionally with a caption, that is self-contained and is typically referenced as a single unit from the main flow of the document.	global attributes**

Table 1.6: List Tags in HTML5

Tag	Description	Attributes
** **	**unordered list:** a list of items, where the order of the items is not important.	global attributes**
** **	**ordered list:** list of items, where the items have been intentionally ordered	**start:** the ordinal value of the first list item reversed: indicates that the list is a descending list [reversed].
** **	**list item:** represents a list item of an Ordered (OL) or Unordered list (UL).	**value:** used in an Ordered List (OL) to set the display value.
<dl>	**definition list:** an association list consisting of zero or more name-value groups (a description list). Each group must consist of one or more names (dt elements) followed by one or more values (dd elements).	global attributes**
<dt>	**definition term:** the term, or name, part of a term description group in a description list (dl element), and the talker, or speaker, part of a talker-discourse pair in a conversation.	global attributes**
<dd>	**definition description:** description, definition, or value, part of a term-description group in a description list (dl element), and the discourse, or quote, part in a conversation (dialog element).	global attributes**

Table 1.7: Formatting Tags in HTML5

Tag	Description	Attributes
<b>	**bold text:** creates text that will be made bold.	global attributes**
<i>	**italic text:** indicates the text is to be rendered with emphasis.	global attributes**
****	**emphasized text:** represents stress emphasis of its contents.	global attributes**
****	**strong:** represents strong importance for its contents.	global attributes**
<small>	**small text:** small print or other side comments.	global attributes**
<bdo>	**bdo element:** represents explicit text directionality formatting control for its children.	**dir:** direction override [ltr | rtl].
<del>	**deleted text:** represents a removal from the document.	**cite:** a URL used to specify the address of a document that explains the change. **datetime:** used to specify the time and date of the change.
<ins>	**inserted text:** an addition to the document	cite: a URL used to specify the address of a document that explains the change. **datetime:** used to specify the time and date of the change.
<mark>	**marked text:** a run of text in one document marked or high-lighted for reference purposes, due to its relevance in another context.	global attributes**
<sub>	**subscript:** subscript text.	global attributes**
<sup>	**superscript:** superscript text.	global attributes**
<meter>	**meter element:** scalar measurement within a known range, or a fractional value.	**high:** specifies the range that is considered to be the "high" part **low:** specifies the range that is considered to be the "low" part **min:** specifies the lower boundary **max:** specifies the upper boundary **optimum:** specifies the range that is considered to be the "optimum" part

		value: current location within the range
<progress>	**progress element:** represents the completion progress of a task.	**max:** specifies how much work the task requires in total. **value:** specifies how much of the task has been completed.

Table 1.8: Programming Tags in HTML 5

Tag	Description	Attributes
<script>	**script element:** allows authors to include dynamic script and data blocks in their documents.	**async:** the script will be executed asynchronously, as soon as it is available [async]. **type:** gives the MIME type of the script or format of the data. **defer:** the script is executed when the page has finished parsing [defer]. **src:** gives the address of the external script resource to use. **charset:** specifies the character encoding of the external script resource.
<noscript>	**noscript section:** represents nothing if scripting is enabled, and represents its children if scripting is disabled.	global attributes**

Table 1.9: Style & Semantics Tags in HTML 5

Tag	Description	Attributes
<style>	**style definition:** allows authors to embed style information in their documents	**media:** says which media the styles. **apply to type:** gives the MIME type (default: text/css) scoped: indicates that the styles are intended just for the subtree rooted at the style element's parent element [scoped].
** **	**document block:** creates a block level element with no special meaning.	global attributes**
****	**span:** used for an inline element.	global attributes**

<header>	**header element:** represents a group of introductory or navigational aids.	global attributes**
<footer>	**footer element:** represents a footer for the section it applies to.	global attributes**
<main>	specifies the main content of a document.	
<section>	**section element:** represents a generic document or application section.	**cite:** a URL of a page where the section was taken from.
<article>	**article element:** a section of a page that consists of a composition that forms an independent part of a document, page, or site.	global attributes**
<details>	**details element:** represents additional information or controls which the user can obtain on demand.	**open:** indicates whether the details are to be shown to the user [open].
<summary>	defines a visible heading for a <details> element.	

Table 1.10: Media: Audio & Video Tags in HTML 5

Tag	Description	Attributes
<audio>	**sound content:** represents a sound or audio stream	**autobuffer:** determines if the audio will be buffered [autobuffer]. **autoplay:** determine if the audio will automatically play [autoplay]. **controls:** indicates that the author has not provided a scripted controller and would like the user agent to provide its own set of controls [controls]. **loop:** sets whether the audio will start once the end is reached [loop]. **src:** URL of the audio to play.
<video>	**video element:** a video or movie.	**poster:** URL of an image file that the user agent can show while no video data is available. **autobuffer:** determines if the audio will

		be buffered [autobuffer]. **autoplay:** determine if the audio will automatically play [autoplay]. **controls:** indicates that the author has not provided a scripted controller and would like the user agent to provide its own set of controls [controls]. **loop:** sets whether the audio will start once the end is reached [loop]. **src:** URL of the audio to play. **width:** width of the video in pixels **height:** height of the video in pixels
<source>	**source element:** allows authors to specify multiple media resources for media elements.	**media:** gives the intended media type of the media resource. **src:** URL of the media resource. **type:** gives the MIME type of the source.

Table 1.11: Table Tags in HTML 5

Tag	Description	Attributes
<table>	**table element:** represents data with more than one dimension, in the form of a table.	global attributes**
<tbody>	**table body:** represents a block of rows that consist of a body of data for a table.	global attributes**
<td>	**table cell:** represents a data cell in a table.	**colspan:** sets how many columns a cell will span. **rowspan:** sets how many rows a cell will span. **headers:** space separated list of ids corresponding to the th ids and give header information for the cell.
<caption>	**table caption:** the title of the table that is its parent, if it has a parent and that is a table element.	global attributes**
<th>	**table header:** represents a header cell in a table.	**colspan:** determines how many columns a cell will span. **rowspan:** determines how many rows a

		cell will span. **headers:** space separated list of ids corresponding to the th ids and give header information for the cell. **scope:** determines where the cell provides its header information [col | colgroup | row | rowgroup].
<thead>	**table header:** the block of rows that consist of the column labels (headers) for a table.	global attributes**
<tr>	**table row:** a row of cells in a table	global attributes**
<tfoot>	**table footer:** the block of rows that consist of the column summaries (footers) for a table.	global attributes**
<col>	**column:** defines the attribute values for one or more columns in a table. Used inside of a table or colgroup.	**span:** number of columns the tag should span.
<colgroup>	**column group:** a group of one or more columns in the table that is its parent, if it has a parent and that is a table element	**span:** number of columns the tag should span.

Table 1.12: Form and Input Tags in HTML5

Tag	Description	Attributes
<form>	**form element:** represents a collection of form-associated elements, some of which can represent editable values that can be submitted to a server for processing.	**accept-charset:** gives the character encodings that are to be used for the submission. **action:** URL that specifies a form **processing agent autocomplete:** determines if form elements will have their autocomplete turned on or off by default [on | off]. **enctype:** specifies the content type used to submit the form to the server [application/x-www-form-urlencoded | multipart/form-data | text/plain]. **method:** which HTTP method will be used to submit the forms data [get | post |

		put | delete]. **name:** elements name. **novalidate:** indicate whether the form is to be validated during submission [novalidate]. **target:** gives the target when the form is submitted [_blank | _parent | _self | _top].
<label>	**label:** caption in a user interface	**for:** specified to indicate a form control with which the caption is to be associated
<input>	**input field:** a typed data field, usually with a form control to allow the user to edit the data	Attributes are dependant upon input type **accept:** specified to provide user agents with a hint of what file types the server will be able to accept alt: provides the textual label for the alternative button for users and user agents who cannot use the image. **autocomplete:** determines if the data is considered sensitive and if autocomplete will be used [on | off | default]. **autofocus:** determines if the input will get focus when a page loads [autofocus]. **checked:** determines if the input will be checked by default [checked]. **disabled:** prevents the input from being pressed [disabled]. **form:** used to explicitly associate the button element with its form owner. **formaction:** URL that specifies a form processing agent. **formenctype:** specifies the content type used to submit the form to the server [application/x-www-form-urlencoded | multipart/form-data | text/plain]. **formmethod:** which HTTP method will be used to submit the forms data [get | post | put | delete]. **formnovalidate:** indicate whether the form is to be validated during submission [formnovalidate]. **formtarget:** gives the target when the

		form is submitted [_blank | _parent | _self | _top]. **height:** height of the input in pixels. **list:** used to identify an element that lists predefined options suggested to the user max and max: indicate the allowed range of values for the element. **maxlength:** controls the maxlength of the input to a control. **multiple:** indicates whether the user is to be allowed to specify more than one value [multiple] name: elements name. **pattern:** specifies a regular expression against which the control's value is to be checked. **placeholder:** a short hint intended to aid the user with data entry. **readonly:** determines if the control is readonly [readonly]. **required:** determines if the input is required before the form submits [required]. **size:** gives the number of characters that, in a visual rendering, the user agent is to allow the user to see while editing. **src:** URL to an image (image button). **step:** indicates the granularity that is expected (and required) of the value. **type:** controls the data type (and associated con- trol) of the element [hidden | text | search | tel | url | email | password | datetime | date | month | week | time | datetime-local | number | range | color | checkbox | radio | file | submit | image | reset | button]. **value:** sets the element's value. **width:** width of the input in pixels.
<textarea>	**text area:** a multiline plain text edit control for the element's raw value	**autofocus:** determines if the textarea gets focus when the page loads [autofocus].

		cols: specifies the expected maximum number of characters per line. **disabled:** prevents entry of text [disabled]. **form:** form to associate the textarea. **readonly:** control whether the text can be edited by the user or not [readonly]. **required:** will be required to enter a value before submitting the form [required]. **rows:** specifies the number of lines to show. **maxlength:** controls the maximum amount of characters which can be entered. **placeholder:** a hint intended to aid the user with data entry. **wrap:** defines how text is wrapped [soft | hard].
<button>	**button:** a button page element	**autofocus:** indicate that a control is to be focused as soon as the page is loaded. **[autofocus] disabled:** prevents the button from being pressed [disabled]. **form:** used to explicitly associate the button element with its form owner. **formaction:** URL that specifies a form processing agent. **formenctype:** specifies the content type used to submit the form to the server [application/x-www-form-urlencoded | multipart/form-data | text/plain]. **formmethod:** which HTTP method will be used to submit the forms data [get | post | put | delete]. **formnovalidate:** indicate whether the form is to be validated during submission. **[formnovalidate]** **formtarget:** gives the target when the form is submitted [_blank | _parent | _self | _top].

		name: elements name. **type:** controls the behavior of the button when it is activated [submit | reset | button]. **value:** gives the element's value for the purposes of form submission.
<select>	**selectable list:** a control for selecting amongst a set of options.	**autofocus:** determines if the controls gets focus when the page loads [autofocus]. **disabled:** prevent the selection of an item [disabled]. **form:** form to associate the select with. **multiple:** allows the selection of multiple items [multiple]. **size:** gives the number of options to show to the user.
<option>	**option element:** an option in a select element or as part of a list of suggestions in a datalist element.	**disabled:** prevent any clicks on an option item [disabled]. **label:** provides a label for element. **selected:** determines if the option is selected by default [selected]. **value:** provides a value for element.
<optgroup>	**option group:** a group of option elements with a common label.	**disabled:** disables all options in the group [disabled]. **label:** gives the name of the group, as shown to the user.
<datagrid>	**datagrid element:** an interactive representation of tree, list.	**disabled:** defines whether the list is selectable [disabled].
<datalist>	**dropdown list:** a set of option elements that represent predefined options for other controls.	global attributes**
<fieldset>	**fieldset element:** a set of form controls grouped under a common name.	**disabled:** controls whether all the form control descendants are disabled. **form:** used to explicitly associate the fieldset element with its form owner. **name:** gives the name of the form control.
<legend>	**fieldset title:** sets the title of a fieldset element.	global attributes**

Cascading Style Sheet

Cascading Style Sheets (CSS) is used to format the layout of a webpage. It is used to add styling to the plain HTML i.e. Style refers how to display HTML elements. CSS 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. The syntax of CSS as shown in Fig.1.3.

The selector points to the HTML element want to be style. The declaration block contains one or more declarations separated by semicolons. Each declaration includes a CSS property name and a value, separated by a colon. Multiple CSS declarations are separated with semicolons, and declaration blocks are surrounded by curly braces.

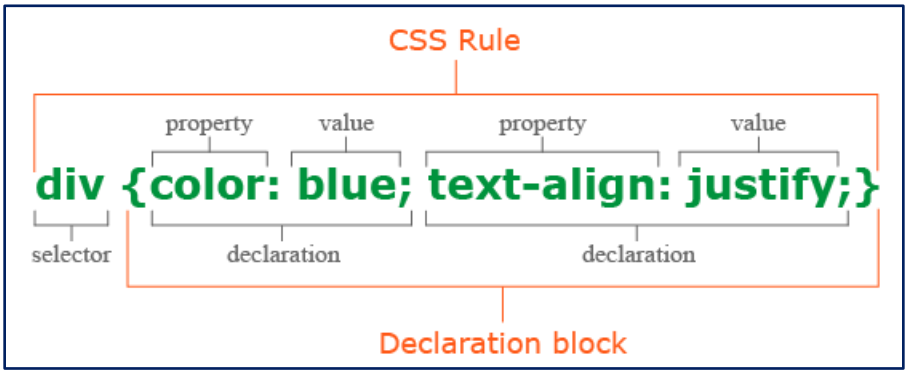


Figure 1.3: General form of CSS3

History of CSS

CSS came first into existence in 1994. CSS evolved all these years and W3C maintained the standards with CSS1, CSS2 and CSS3. They were built on top of each other with better adaptability and more features. Each one had issues that were solved by the next standard. CSS got major adaptability issues with multiple web browsers coming up all these years starting from Internet Explorer, Opera, Firefox, Chrome and Edge.

- **CSS 1** was adapted in 1996. It was difficult and less adapted by then recent browsers such as Internet Explorer 3, Internet Explorer 4 and Netscape 4.x. It had font properties and specification for typeface and emphasis. The text attributes were supported such as spacing of letters and line of text. Alignment of text, positioning and tables were also added. Margin, padding, border and positioning for elements were also implemented. This recommendation was not maintained by W3C.

- **CSS2** is the improvement of CSS1. It removed the not fully interoperable features. It also included the browser extensions.
- **CSS3** is the most recent and currently used. It has Extensible HyperText Markup Language (XHTML) specification. CSS3 has its major focus on modularization and separation of concerns. CSS3 has support for almost all recent web browsers. It has even included new selectors along with new combinator and new pseudo-elements. CSS3 has several new CSS properties. It supports animation which is not a part of earlier recommendations. There were various properties added such as transforms, gradients, animation and transition for animation effect in the website. Recent add-ons are like border-radius, box-shadow, flex-box and CSS grid. It had many new features such as absolute, relative and fixed positioning of elements. It supported different media types. It also included new font properties like shadow.

About CSS3:

CSS3 is divided into "modules". Previous specifications of CSS are divided into smaller pieces, and latest ones are also added. The most important modules of CSS3 are,

- Advance Selectors
- Box Model
- Backgrounds and Borders
- Some Text Effects
- 2D / 3D Transformations
- Animations
- Multiple Column Layout
- User Interface

Features of CSS3

1. **Selectors –** CSS3 allow the designer to select more precise levels of the web page. They are structural pseudo-classes that perform partial matches to help match attribute and attribute values. New selectors target a pseudo-class to style the elements targeted in the URL. Selectors also include a checked pseudo-class to style checked elements such as checkboxes and radio buttons.

2. **Text Effects and Layout -** With CSS3, we can change the justification of text, make whitespace adjustments to the document, and style the hyphenation of words.
3. **First-Letter and First-Line Pseudo-Classes -** CSS 3 includes properties that help with kerning (adjusting the spacing between characters to achieve a visually pleasing effect) and positioning drop-caps (large decorative capital letters at the start of a paragraph).
4. **Paged Media and Generated Content -** CSS 3 has additional choices in Paged Media, such as page numbers and running headers and footers. CSS offers additional properties for printing generated content, including properties that specifically address cross-references and footnotes.
5. **Multi-Column Layout -** This feature includes properties to allow designers to present their content in multiple columns with options like the column count, column gap, and column width.

Advantages of CSS3

- CSS3 provides a consistent and precise positioning of navigable elements.
- It is easy to customize a web page as it can be done by merely altering a modular file.
- Graphics are more accessible in CSS3, thus making it easy to make the site appealing.
- It permits online videos to be seen without using third-party plugins.
- CSS3 is economical and time-saving, and most browsers support it.

Types of CSS

CSS can be added to HTML documents in three 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

1. Inline CSS

Inline CSS is used to style a specific HTML element. For this CSS style, you'll only need to add the style attribute to each HTML tag, without using selectors. Example, need to apply style for a single element. The HTML for adding an inline CSS to the <p> and <h1> tag will looks like,

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

```
        <body style= “background-color:black;”>
            <h1 style=“color:white;padding:30px; ”>Hostinger Tutorials</h1>
            <p style=“color:white;”>Something usefull here.</p> </body>
</html>
```

Advantages of Inline CSS:

- Inserting CSS rules to an HTML page is simple and easy for testing or previewing the changes, and performing quick-fixes to the website.
- Do not need to create and upload a separate document as in the external style.

Disadvantages of Inline CSS:

- Adding CSS rules to every HTML element is time-consuming and it makes the HTML structure messy.
- Styling multiple elements can affect the webpage’s size and download time.

2. Internal CSS

Internal or embedded CSS requires adding <style> tag in the <head> section of the HTML document. This CSS style is an effective method of styling a single webpage. However, using this style for multiple pages is time-consuming i.e. need to include CSS rules on every webpage of the website. The steps to include internal CSS are,

1. Open your HTML page and locate <head> opening tag.
2. Include the following code right after the <head> tag
   <style type="text/css">
3. Add CSS rules on a new line. Example:

```
body {
        background-color: blue;
        }
        h1 {
                color: red;
                padding: 60px;
        }
```

4. Type the closing tag:

   </style>

The HTML document will look likes,

```
<!DOCTYPE html>
<html>
    <head>
        <style>
          body {
              background-color: blue;
           }
           h1 {
               color: red;
               padding: 60px;
            }
        </style>
    </head>
    <body>
        <h1>Tutorials to learn CSS3</h1>
        <p>This is our paragraph.</p>
    </body>
</html>
```

Advantages of Internal CSS:

Internal CSS use class and ID selectors in this style sheet. Since it is possible to add the code within the same HTML file,

Example:

```
.class {
  property1 : value1;
  property2 : value2;
  property3 : value3;
}
```

```
#id {
   property1 : value1;
   property2 : value2;
   property3 : value3;
}
```

Disadvantages of Internal CSS:

- Adding the code to the HTML file can increase the page's size and loading time.

External CSS

Link the web pages to an external .css file, which can be created by any text editor (e.g., Notepad++). This CSS type is a more efficient method, especially for styling a large website. By editing a single .css file, it is possible to changes of the entire website at once. The steps to include external CSS are,

1. Create a new .css file with the text editor, and add the style rules. For example:

```
.xleftcol {
  float: left;
  width: 33%;
  background:#809900;
}
.xmiddlecol {
  float: left;
  width: 34%;
  background:#eff2df;
}
```

2. In the <head> section of the HTML file, add a reference to link external .css file right after <title> tag: <link rel="stylesheet" type="text/css" href="style.css" />

Advantages of External CSS:

- Since the CSS code is in a separate document, the HTML files will have a cleaner structure and are smaller in size.
- Use the same .css file for multiple pages.

Disadvantages of External CSS:

- The webpages may not be rendered correctly until the external CSS is loaded.
- Uploading or linking to multiple CSS files can increase the website download time.

Positioning Elements

The positioning elements in CSS define the position of the HTML entity or the element. In CSS, there are five different types of positioning properties available are,

- Fixed
- Static
- Relative
- Absolute
- Sticky

1. Fixed

Elements with the position fixed always remain in the same position even we scroll the webpage. The properties are used to position the elements on the left, right, top, and bottom.

Example:

```
<html>
   <title> CSS Positioning Element</title>
      <style>
            div.fixed {
            position: fixed;
            bottom: 0;
            left: 0;
            width: 300px;
            border: 5px solid #046ad7;
           }
      </style>
 <body>
       <div class="fixed">
            It has Fix Position      </div>
```

```
        <p> An element with the position fixed always stays in the same place even if we scroll the page it still remains in the same place. To position the elements the left, right, top and bottom properties are used. </p>
    </body>
</html>
```

2. Static

The static method of positioning is set by default. Such elements are not affected by the properties of the left, right, top, and bottom. It is always positioned according to the normal flow of the webpage.

Example:

```
<html>
    <title> CSS Positioning Element</title>
        <style>
                div.static {
                 position: static;
                border: 5px solid #046ad7;
              }
        </style>
    <body>
        <div class=“ static”>
                It has static position
        </div>
        <p> The method of positioning is set by default. They are not affected by the left, right, top and bottom properties. It is always positioned according to the normal flow of the page. It is not positioned in any other special way. </p>
    </body>
</html>
```

3. Relative

An element with a relative position is positioned relative to its normal position. If we set its left, right, top, and bottom, other elements will not fill up the gap left by this element.

Example:

```
<html>
   <title> CSS Positioning Element</title>
      <style>
            div.relative {
            position: relative;
            left: 30px;
            border: 5px solid #046ad7;
          }
      </style>
    <body>
        <div class=“ relative”>
            It has relative position
        </div>
        <p> An element with relative position is positioned relative to its normal position. If we set up the left, right, top and bottom will not be able to fill the gap which is left by this element. </p>
     </body>
</html>
```

4. Absolute

An element with the absolute position is positioned with respect to its parent. It doesn’t depend on the elements under the same level or the siblings.

Example:

```
<html>
   <title> CSS Positioning Element</title>
      <style>
            div.relative {
            position: relative;
            width: 400px;
            height: 200px;
            border: 3px solid #046ad7;   }
```

```
            div.absolute {
            position: absolute;
            top: 80px;
            right: 0;
            width: 200px;
            height: 100px;
            border: 3px solid #046ad7;
        }
    </style>
<body>
    <div class="relative">It has relative position
        <div class="absolute">It has absolute position</div>
    </div>
    <p> An element with the absolute position is positioned with respect to its parent. It doesn't depend on the elements which are under the same level or on the siblings. </p>
</body>
</html>
```

5. Sticky

An element with the sticky position is positioned according to the position of the user's scroll. Depending upon the position of the scroll, the sticky element toggles between the fixed and relative.

Example:

```
<html>
    <title> CSS Positioning Element</title>
    <style>
            div.sticky {
            position: -webkit-sticky;
            position: sticky;
            top: 0;
            padding: 5px;
```

```
            background-color: #22dad1;
            border: 5px solid #007f9b;
        }
    </style>

<body>
    <div class="sticky">Sticky Element</div>
        <div style="padding-bottom:2000px">
        <p>In this example, the sticky element sticks to the top of the page (top: 0), when you reach its scroll position.</p>
        <p>Scroll back up to remove the stickyness.</p>
        <p>Some text to enable scrolling.. Lorem ipsum dolor sit amet, illum definitiones no quo, maluisset concludaturque et eum, altera fabulas ut quo. Atqui causae gloriatur ius te, id agam omnis evertitur eum. Affert laboramus repudiandae nec et. Inciderint efficiantur his ad. Eum no molestiae voluptatibus.</p>
        <p>Some text to enable scrolling.. Lorem ipsum dolor sit amet, illum definitiones no quo, maluisset concludaturque et eum, altera fabulas ut quo. Atqui causae gloriatur ius te, id agam omnis evertitur eum. Affert laboramus repudiandae nec et. Inciderint efficiantur his ad. Eum no molestiae voluptatibus.</p>
    </div>
</body>
</html>
```

CSS3 Backgrounds

CSS3 contains a few new background properties, which allow greater control of the background element. The CSS3 properties are shown in Table 1.17.

Table 1.13: CSS3 Background properties

Property	Description
background	For setting all the background properties in one declaration.

background-clip	Specifies the painting area of the background.
background-image	Specifies one or more background images for an element.
background-origin	Specifies where the background image(s) is / are positioned.
background-size	Specifies the size of the background image(s).

1. CSS3 Multiple Backgrounds

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: Two background images, the first image is a flower (aligned to the bottom and right) and the second image is a paper background (aligned to the top-left corner):

```
#example1 {
   background-image: url(img_flwr.gif), url(paper.gif);
   background-position: right bottom, left top;
   background-repeat: no-repeat, repeat;
}
```

Multiple background images can be specified using either the individual background properties (as mentioned above) or the background shorthand property. The following example uses the background shorthand property.

```
#example1 {
   background: url(img_flwr.gif) right bottom no-repeat, url(paper.gif) left top repeat;
}
```

2. CSS3 Background Size

The CSS3 background-size property specifies the size of background images. Before CSS3, the size of a background image was the actual size of the image. CSS3 allows us to re-use background images in different contexts. The size can be specified in lengths, percentages, or by using one of the two keywords as, contain or cover. The following example resizes a background image to much smaller than the actual image (using pixels):

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

```
  <style>
      #example1 {
         border: 1px solid black;
         background:url(img_flwr.gif);
         background-repeat: no-repeat;
         padding:15px;
      }
      #example2 {
         border: 1px solid black;
         background:url(img_flwr.gif);
         background-size: 100px 80px;
         background-repeat: no-repeat;
         padding:15px;
      }
   </style>
</head>
<body>
    <p>Original background-image:</p>
    <div id="example1">
    <h2>Lorem Ipsum Dolor</h2>
    <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.</p>
    <p>Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.</p>
    </div>
    <p>Resized background-image:</p>
    <div id="example2">
    <h2>Lorem Ipsum Dolor</h2>
    <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.</p>
```

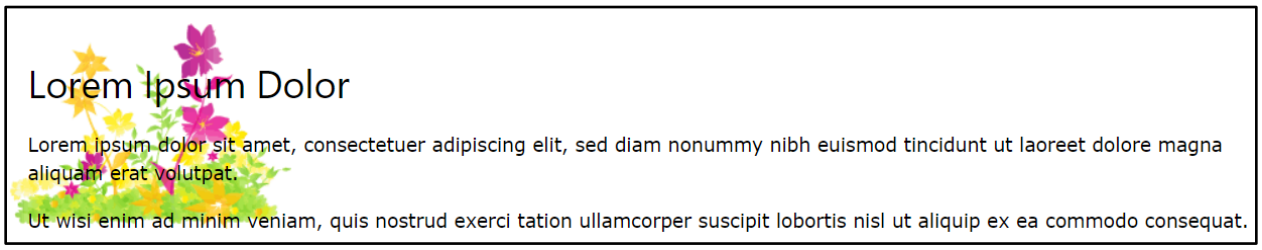
Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

</body>

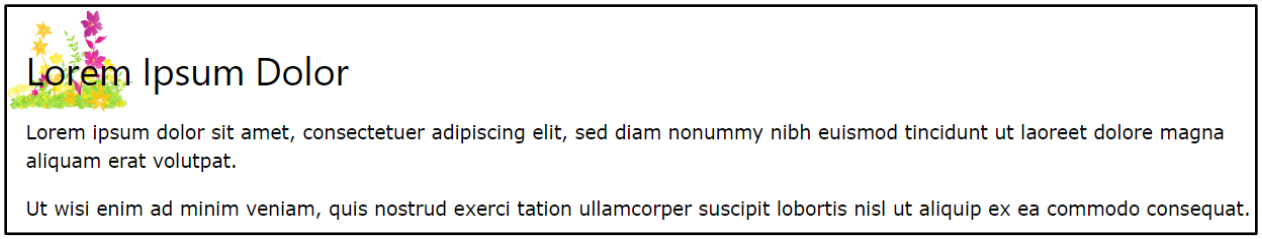
</html>

Output:

The actual background image layout will looks like,



The resized background image layout will looks like,



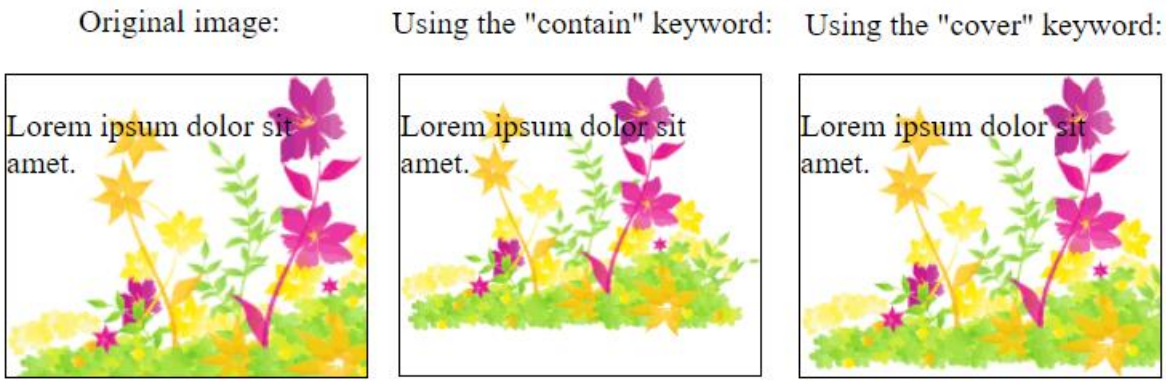
The contain keyword scales the background image to be as large as possible (but both its width and its height must fit inside the content area). The cover keyword scales the background image so that the content area is completely covered by the background image (both its width and height are equal to or exceed the content area).

```
<!DOCTYPE html>
<html>
    <head>
      <style>
          .div1 {
            border: 1px solid black;
            height:150px;
            width:180px;
            background:url(img_flwr.gif);
```

```
            background-repeat: no-repeat;
        }
        .div2 {
            border: 1px solid black;
            height:150px;
            width:180px;
            background:url(img_flwr.gif);
            background-repeat: no-repeat;
            background-size: contain;
        }
        .div3 {
            border: 1px solid black;
            height:150px;
            width:180px;
            background:url(img_flwr.gif);
            background-repeat: no-repeat;
            background-size: cover;
        }
    </style>
</head>
<body>
    <p>Original image:</p>
    <div class="div1">
    <p>Lorem ipsum dolor sit amet.</p>
    </div>
    <p>Using the "contain" keyword:</p>
    <div class="div2">
    <p>Lorem ipsum dolor sit amet.</p>
    </div>
    <p>Using the "cover" keyword:</p>
    <div class="div3">
```

```
        <p>Lorem ipsum dolor sit amet.</p>
        </div>
    </body>
</html>
```

Output:



3. Define Sizes of Multiple Background Images

The background-size property also accepts multiple values for background size (using a comma-separated list).

Example:

```
#example1 {
background: url(img_flwr.gif) left top no-repeat, url(img_flwr.gif) right bottom no-repeat,
url(paper.gif) left top repeat;
  background-size: 50px, 130px, auto;
}
```

4. Full Size Background Image

To have a background image on a website that covers the entire browser window at all times. The requirements are as follows:

- Fill the entire page with the image (no white space)
- Scale image as needed
- Center image on page
- Do not cause scrollbars

Example: Use the html element (the html element is always at least the height of the browser window). Then set a fixed and centered background on it. Then adjust its size with the background-size property:

```
html {
  background: url(img_flower.jpg) no-repeat center fixed;
  background-size: cover;
}
```

5. CSS3 background-origin Property

The CSS3 background-origin property specifies where the background image is positioned. The property takes three different values:

- **border-box:** the background image starts from the upper left corner of the border.
- **padding-box:** (default) the background image starts from the upper left corner of the padding edge.
- **content-box**: the background image starts from the upper left corner of the content.

Example:

```
#example1 {
  border: 10px solid black;
  padding: 35px;
  background: url(img_flwr.gif);
  background-repeat: no-repeat;
  background-origin: content-box;
}
```

6. CSS3 background-clip Property

The CSS3 background-clip property specifies the painting area of the background. The property takes three different values:

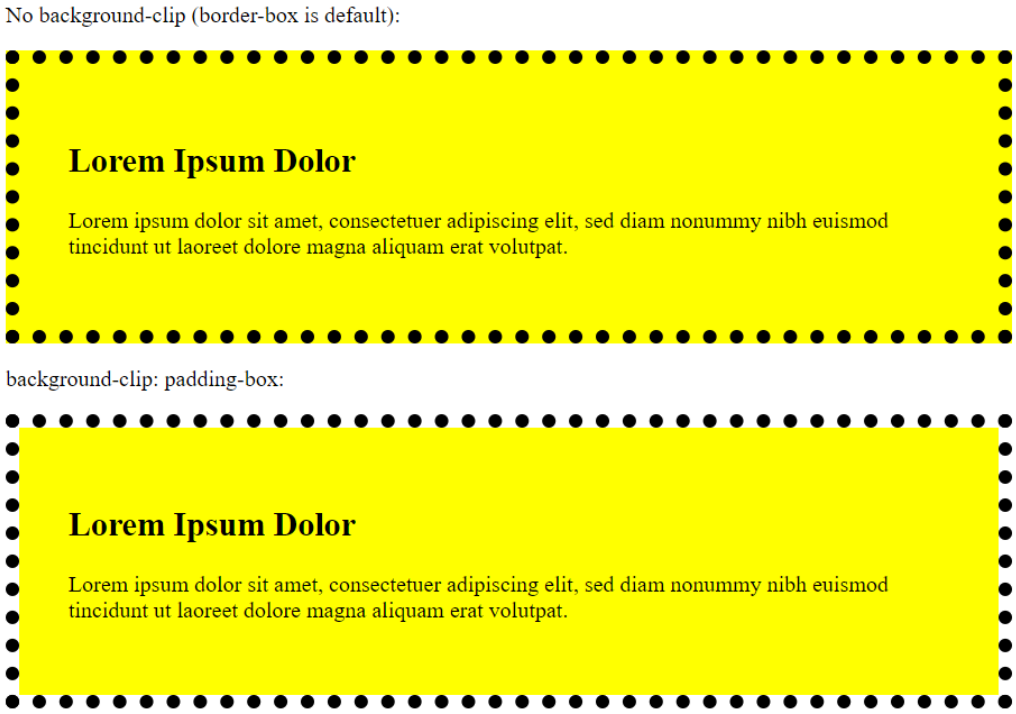
- **border-box:** (default) the background is painted to the outside edge of the border.
- **padding-box:** the background is painted to the outside edge of the padding.
- **content-box:** the background is painted within the content box.

Example:

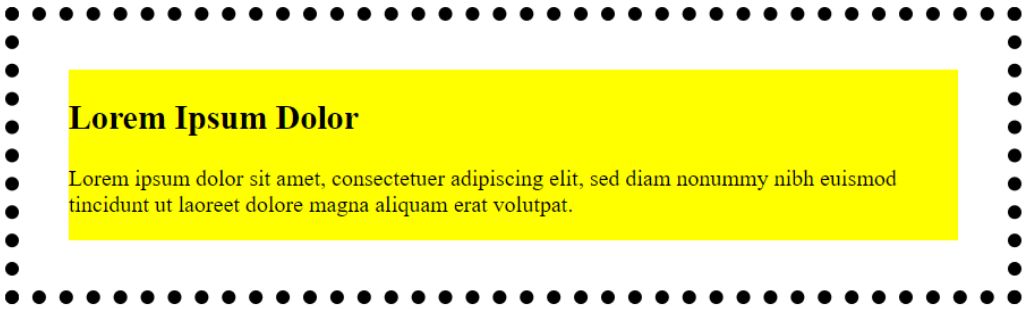
```
<!DOCTYPE html>
        <html>
                <head>
                   <style>
                      #example1 {
                         border: 10px dotted black;
                         padding:35px;
                         background: yellow;
                      }
                      #example2 {
                         border: 10px dotted black;
                         padding:35px;
                         background: yellow;
                         background-clip: padding-box;
                      }
                      #example3 {
                         border: 10px dotted black;
                         padding:35px;
                         background: yellow;
                         background-clip: content-box;
                      }
                </style>
        </head>
        <body>
           <p>No background-clip (border-box is default):</p>
           <div id="example1">
           <h2>Lorem Ipsum Dolor</h2>
           <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh
           euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.</p>
           </div>
```

```
        <p>background-clip: padding-box:</p>
        <div id="example2">
        <h2>Lorem Ipsum Dolor</h2>
        <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.</p>
        </div>
        <p>background-clip: content-box:</p>
        <div id="example3">
        <h2>Lorem Ipsum Dolor</h2>
        <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.</p>
        </div>
    </body>
</html>
```

Output:



background-clip: content-box:



CSS Box Model and Text Flow

The purpose of the CSS Box Model is to define all the elements on the webpage as a box. The browser uses the CSS box model to determine how an element should appear and how it should be positioned on a web page. It serves as a toolkit for positioning the elements, their content, and the elements surrounding them. The CSS Box Model, in a nutshell, is a layout guide for the elements on the webpage. The CSS Box Model as shown in Fig.1.4. describes all the HTML elements of the webpage as rectangular boxes. As shown in Figure the contents of the navigation bar or buttons is a box. A CSS basic box model consists of a content area, where any text, images, or other HTML elements are displayed. This is optionally surrounded by padding (space inside the element's border), a border (wraps the content area and padding), and a margin (space outside the element's border), on one or more sides. In the standard box model, the width and a height attribute defines the content area of the element. Any padding and border is then added to that width and height to get the total size of the box.

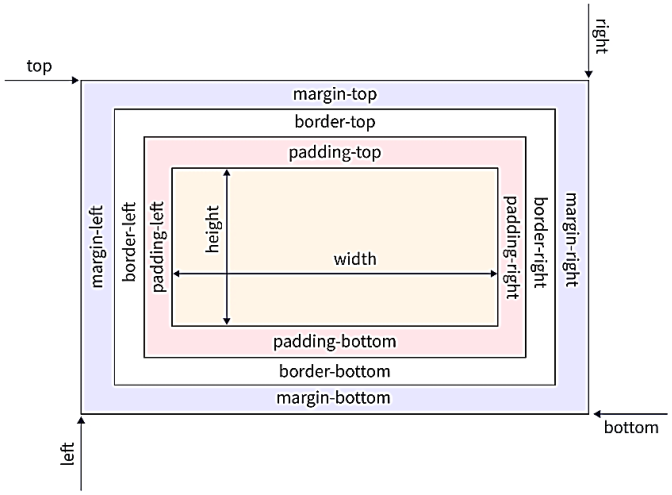


Figure1.4: CSS Box Model

The whole webpage layout is the arrangement of these HTML elements that are represented as boxes. Here, the content-area is the innermost layer of the box model that contains the main content of the element, like text or an image. The next layer is the padding area, which creates extra space around the content area and within the element's border. The border area is the third layer, which exists as a boundary between the padding and margin of the box. It wraps the content area and the padding and can be styled in many ways. The outermost layer is the margin, which creates extra space outside the element, generally used to separate the element from other elements on the web page.

Properties of the CSS Box Model:

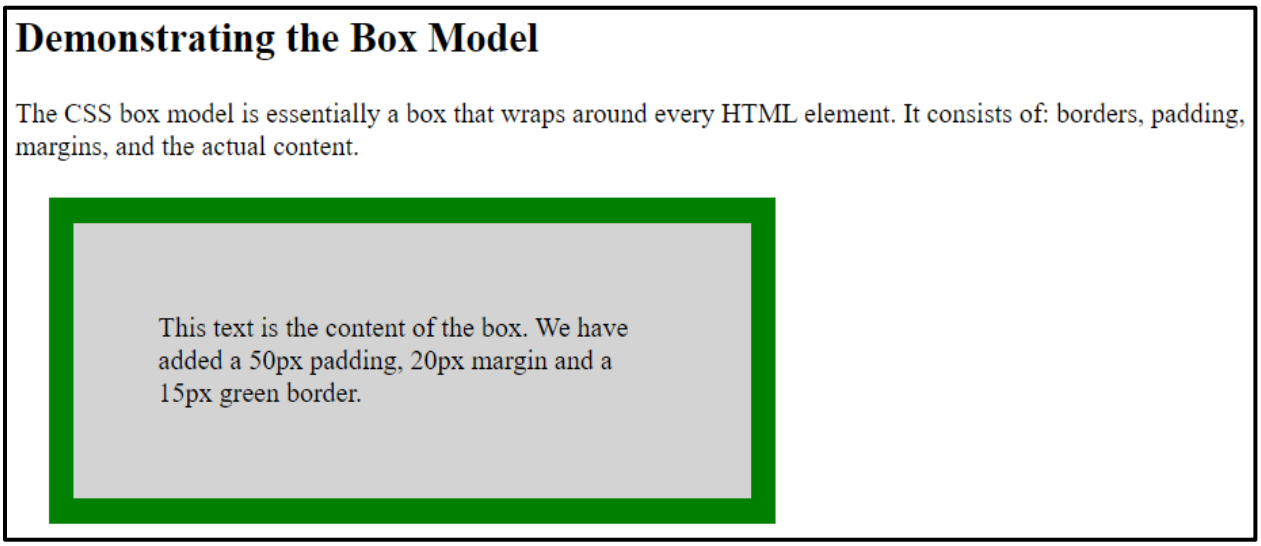
- **Content**: The content area contains the real content of the element, such as text, an image, or a video player. It is the area where the content gets displayed on the webpage. Its dimensions can be modified using properties like width and height.
- **Padding**: The padding area is the space around the content area and within the element's border. It creates extra space inside the element's border and uses the same background as the element itself. The dimensions of the padding is determined by the padding-top, padding-right, padding-bottom, padding-left, and shorthand padding properties.
- **Border**: The border area is the space around the padding area and within the margin. It includes the element's borders and wraps the content and any padding. Its size and style can be controlled using border and related properties. Example, it can be set to dotted, dashed, solid, double, none, or hidden. It can also have rounded corners using the border-radius property.
- **Margin:** The margin area is the transparent space outside the element's border and doesn't have any background color. The margin wraps the content, padding, and border and mostly used to separate the element from other HTML elements on the web page. The size of the margin area is specified using the margin-top, margin-right, margin-bottom, margin-left, and shorthand margin properties.

Example:

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

```
            div {
              background-color: lightgrey;
              width: 300px;
              border: 15px solid green;
              padding: 50px;
              margin: 20px;
            }
        </style>
    </head>
    <body>
        <h2>Demonstrating the Box Model</h2>
            <p>The CSS box model is essentially a box that wraps around every HTML element. It consists of: borders, padding, margins, and the actual content.</p>
            <div>This text is the content of the box. We have added a 50px padding, 20px margin and a 15px green border. </div>
    </body>
</html>
```

Output



Dropdown Menus

Dropdown menus are a type of design pattern that allows us to hide or reveal a dropdown box, making it easy for users to navigate websites and applications. With a simple click or hover, this type of menu will gracefully display predefined content or a list of items.

Example:

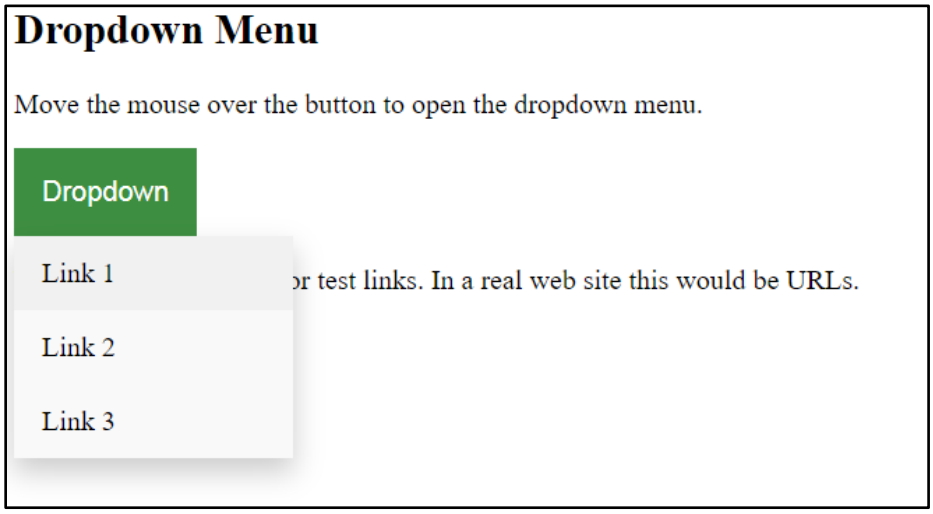
```
<!DOCTYPE html>
<html>
  <head>
    <style>
      .dropbtn {
        background-color: #4CAF50;
        color: white;
        padding: 16px;
        font-size: 16px;
        border: none;
        cursor: pointer;
      }
      .dropdown {
        position: relative;
        display: inline-block;
      }
      .dropdown-content {
        display: none;
        position: absolute;
        background-color: #f9f9f9;
        min-width: 160px;
        box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);
        z-index: 1;
      }
      .dropdown-content a {
        color: black;
```

```
            padding: 12px 16px;
            text-decoration: none;
            display: block;
          }

          .dropdown-content a:hover {background-color: #f1f1f1}

          .dropdown:hover .dropdown-content {
            display: block;
          }
          .dropdown:hover .dropbtn {
            background-color: #3e8e41;
          }
        </style>
    </head>
    <body>
        <h2>Dropdown Menu</h2>
        <p>Move the mouse over the button to open the dropdown menu.</p>
        <div class="dropdown">
          <button class="dropbtn">Dropdown</button>
            <div class="dropdown-content">
            <a href="#">Link 1</a>
            <a href="#">Link 2</a>
            <a href="#">Link 3</a>
            </div>
        </div>
          <p><strong>Note:</strong> We use href="#" for test links. In a real web site this
          would be URLs.</p>
    </body>
</html>
```

Output



Responsive Web Design

Introduction

Responsive Web Design (RWD) was introduced in the early 2000s and eventually adopted as a standard feature for many web pages around 2013. This was around the time when smartphones and other mobile devices became widespread, and it was necessary to have web pages that could be easily viewed on small screens. RWD is not a new concept for web developers by any means but it is no less important than when it initially surfaced. There are a multitude of different screen sizes as shown in Fig.1.5. From mobile to desktop, and everything in between, and proprietors want their webpage to look its best on all of them. Although static pages are easier and cheaper to create, they fall short when it comes to engaging with a wider audience. Since these pages cannot adjust their content to fit different resolutions, then viewers may struggle to view the content on different devices. Consequently, static pages require multiple different versions to be adapted to different device sizes. This can mean that different URLs are required between mobile and desktop web pages.

Responsive Web Design (RWD) describes an approach to web design that makes a web page render well on a variety of devices, regardless of the screen or browser window size. Web pages and applications made with RWD fluidly rescale themselves according to the size of the user's device or screen. Responsive web pages are adaptable, giving users the best possible viewing experience regardless of their device of choice.

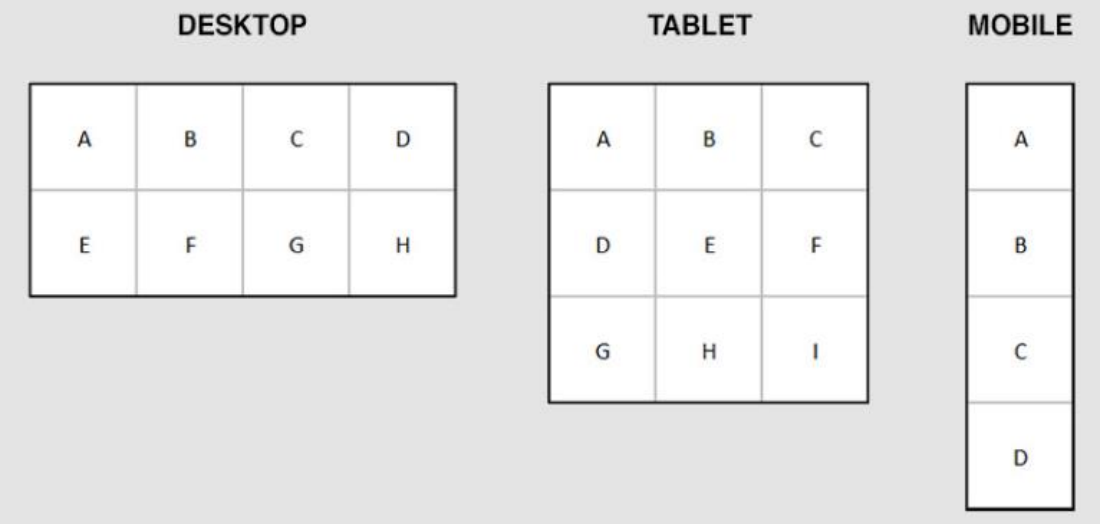


Figure1.5: Basics of Responsive Web Design

The key idea of RWD is setting different breakpoints, such as the width of a browser then changing the layout of a website dynamically based on the current breakpoint. Traditionally, the only responsive web pages were HTML and CSS, but now most contemporary responsive web pages are built using CSS3 or JavaScript's jQuery. The main reasons for RWD is crucial when it comes to web development are,

- **Increased mobile usage:** Recently, people using their smartphones more than ever before to browse the internet. In fact, mobile traffic now accounts for over 50% of all internet traffic. This means that if any company not designed their website for mobile will lose a lot of potential visitors.
- **Improved user experience:** Responsive design ensures that users can access the content no matter what device they are using. This is important as users are more likely to engage with and return to a website that is easy to use.
- **Faster development:** A single webpage for all devices is easier for the developer than making a half a dozen versions of a same webpage for target devices.
- **Easier to maintain:** A single website as compared to several is much more manageable over time.
- **Design is future-proof:** RWD works on future devices with different screen sizes with little or no change.

- **Better SEO:** Responsive web pages are easier for browsers to crawl and index, meaning that the content is more readily and accurately depicted in search results.

Design Aspects of RWD

- **Device capability media queries:** The media queries are based on the capabilities of the hardware used to load the webpage.
- **Responsive images:** Including media assets with fixed resolutions means that percentage-based scaling must be used to resize them in an accessible way.
- **Responsive data tables:** Since data tables can require a lot of screen real estate, it can be challenging to reformat them in a meaningful and visually appealing way. This may include charts, graphs, or re-oriented tables.
- **Responsive navigation menus:** This is most readily apparent on mobile; many navigation menus are converted to dropdown menus that are easy to scroll through on a mobile phone screen.
- **RWD frameworks:** There are plenty of frameworks for web development, but among the most popular are Pure CSS, Montage JS, and Bootstrap.

Examples:

- **Shopify:** Users can create responsive eCommerce sites with Shopify, an eCommerce platform designed to be responsive. Shopify offers a wide variety of responsive themes. Most traffic and orders come from mobile devices on Shopify stores, as over 80% of them do. It adapts its design to a single column by choosing images that crop instead of zooming and removing some of the extra elements. The hamburger menu appears on mobile and tablet websites as well.
- **YouTube:** More than 70 percent of YouTube's users are mobile, and the service has over 2 billion users. The users can watch YouTube on the web or as a responsive Progressive Web App (PWA), adjusted to a smaller grid size (fewer columns) for watching single videos on very small screens.
- **Google Maps:** The fact that Google Maps is the most popular navigation app on mobile devices should come as no surprise.

Advantages

- Attract a wider audience
- Easier to monitor analytics
- Easier to maintain
- Boost for SEO
- Consistency in design and brand
- Cost effective
- Increases the sales and conversion rate

Media queries

A Media query is a CSS3 feature that makes a webpage adapt its layout to different screen sizes and media types.

Syntax

```
@media media type and (condition: breakpoint) {
  // CSS rules
}
```

The developers can target different media types under a variety of conditions. If the condition and / or media types meet, then the rules inside the media query will be applied.

- **@ Media Rule -** Defining media queries with @media rule and later include CSS rules inside the curly braces. The @ media rule is also used to specify target media types.

```
@media () {
  // CSS rules
}
```

- **Parenthesis -** Inside the parenthesis, we set a condition. Example, apply a larger font size for mobile devices. To do that, we need to set a maximum width which checks the width of a device:

```
.text {
  font-size: 14px;
}
@media (max-width: 480px) {
  .text {
```

```
    font-size: 16px;
  }  }
```

- Normally, the text size will be 14px. However since we applied a media query, it will change to 16px when a device has a maximum width of 480px or less.
- **Note:** Always add the media queries at the end of the CSS file.

Media Types

However not applied a media type, the @ media rule selects all types of devices by default. Otherwise, Media types come right after the @ media rule. There are many kinds of devices grouped them into four categories as,

- **all-** for all media types
- **print -** for printers
- **screen -** for computer screens, tablets and, smart-phones
- **speech -** for screen readers that "read" the page out loud

Breakpoints

A breakpoint is a key to determine when to change the layout and adapt the new rules inside the media queries. Example

```
@media (max-width: 480px) {
  .text {
    font-size: 16px;
  }
}
```

Here, the breakpoint is 480px. Now the media query knows when to set or overwrite the new class. Basically, if the width of a device is smaller than 480px, the text class will be applied, otherwise, it won't. There is no standard resolution for devices in which it can be exactly defined; some of the common breakpoints for widths of devices are,

- **320px - 480px:** Mobile devices
- **481px - 768px:** iPads, Tablets
- **769px -1024px:** Small screens, laptops
- **1025px -1200px:** Desktops, large screens
- **1201px and more:** Extra-large screens, TV