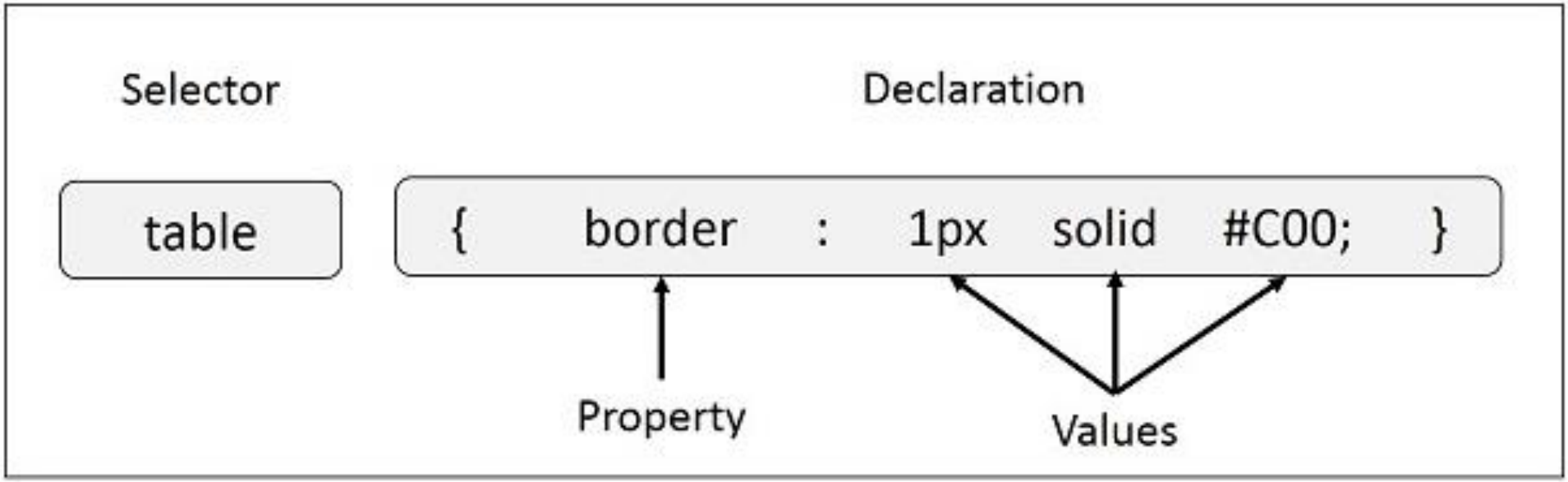
CSS

- CSS is used to control the style of a web document in a simple and easy way. CSS is the acronym for "Cascading Style Sheet".
- CSS handles the look and feel part of a web page.
- Using CSS, you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colors are used, layout designs, variations in display for different devices and screen sizes as well as a variety of other effects.

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

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

- table{ border :1px solid #C00; }



- **CSS Selectors**
- CSS selectors are used to "find" (or select) HTML elements based on their element name, id, class, attribute, and more.
- **The element Selector**
- The element selector selects elements based on the element name.
- You can select all <p> elements on a page like this (in this case, all <p> elements will be center-aligned, with a red text color):
- p {
  text-align: center;
  color: red;
  }

- **The id Selector**
- The id selector uses the id attribute of an HTML element to select a specific element.
- The id of an element should be unique within a page, so the id selector is used to select one unique element!
- To select an element with a specific id, write a hash (#) character, followed by the id of the element.
- The style rule below will be applied to the HTML element with id="para1":
- ```
  #para1 {
      text-align: center;
      color: red;
  }
  ```
  ```

- **The class Selector**
- The class selector selects elements with a specific class attribute.
- To select elements with a specific class, write a period (.) character, followed by the name of the class.
- In the example below, all HTML elements with class="center" will be red and center-aligned:
- ```
  .center {
      text-align: center;
      color: red;
  }
  ```
- In the example below, only <p> elements with class="center" will be center-aligned:
- ```
  p.center {
      text-align: center;
      color: red;
  }
  ```

- Grouping Selectors
- h1, h2, p {
  text-align: center;
  color: red;
  }
- CSS Comments
- p {
  color: red;
  /* This is a single-line comment */
  text-align: center;
  }

  /* This is
  a multi-line
  comment */

- There are three ways of inserting a style sheet:
- External style sheet
- Internal style sheet
- Inline style
- With an external style sheet, you can change the look of an entire website by changing just one file!
- Each page must include a reference to the external style sheet file inside the <link> element. The <link> element goes inside the <head> section:
- <head>
  <link rel="stylesheet" type="text/css" href="mystyle.css">
  </head>
- An external style sheet can be written in any text editor. The file should not contain any html tags. The style sheet file must be saved with a .css extension.

- ```
  body {
      background-color: lightblue;
  }

  h1 {
      color: navy;
      margin-left: 20px;
  }
  ```
- An internal style sheet may be used if one single page has a unique style.
- Internal styles are defined within the <style> element, inside the <head> section of an HTML page:
- ```
  <head>
  <style>
  body {
      background-color: linen;
  }

  h1 {
      color: maroon;
      margin-left: 40px;
  }
  </style>
  </head>
  ```

- 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.
- The example below shows how to change the color and the left margin of a <h1> element:
- <h1 style="color:blue;margin-left:30px;">This is a heading</h1>

- **CSS Backgrounds**
- The CSS background properties are used to define the background effects for elements.
- CSS background properties:
  - background-color
  - background-image
  - background-repeat
  - background-attachment
  - background-position
- ```
  body {
      background-color: lightblue;
  }
  ```
- ```
  body {
      background-image: url("paper.gif");
  }
  ```

- body {
  background-image: url("gradient_bg.png");
  background-repeat: repeat-x;
  }
- If the image is repeated only horizontally (background-repeat: repeat-x;).
- To repeat an image vertically, set background-repeat: repeat-y;
- body {
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
  background-position: right top;
  }
- body {
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
  background-position: right top;
  background-attachment: fixed;
  }
- body {
  background: #ffffff url("img_tree.png") no-repeat right top;
  margin-right: 200px;
  }

- The CSS border properties allow you to specify the style, width, and color of an element's border.
- The **border-style** property specifies what kind of border to display.
- The following values are allowed:
  - dotted - Defines a dotted border
  - dashed - Defines a dashed border
  - solid - Defines a solid border
  - double - Defines a double border
  - groove - Defines a 3D grooved border. The effect depends on the border-color value
  - ridge - Defines a 3D ridged border. The effect depends on the border-color value
  - inset - Defines a 3D inset border. The effect depends on the border-color value
  - outset - Defines a 3D outset border. The effect depends on the border-color value
  - none - Defines no border
  - hidden - Defines a hidden border
- The border-style property can have from one to four values (for the top border, right border, bottom border, and the left border).

- p.dotted {border-style: dotted;}
  p.dashed {border-style: dashed;}
  p.solid {border-style: solid;}
  p.double {border-style: double;}
  p.groove {border-style: groove;}
  p.ridge {border-style: ridge;}
  p.inset {border-style: inset;}
  p.outset {border-style: outset;}
  p.none {border-style: none;}
  p.hidden {border-style: hidden;}
  p.mix {border-style: dotted dashed solid double;}
- The border-width property specifies the width of the four borders.
- The width can be set as a specific size (in px, pt, cm, em, etc) or by using one of the three pre-defined values: thin, medium, or thick.
- The border-width property can have from one to four values (for the top border, right border, bottom border, and the left border).

```
p.one {
    border-style: solid;
    border-width: 5px;
}

p.two {
    border-style: solid;
    border-width: medium;
}

p.three {
    border-style: solid;
    border-width: 2px 10px 4px 20px;
}
```

```
p.one {
    border-style: solid;
    border-color: red;
}

p.two {
    border-style: solid;
    border-color: green;
}

p.three {
    border-style: solid;
    border-color: red green blue yellow;
}
```

- p {
  border-top-style: dotted;
  border-right-style: solid;
  border-bottom-style: dotted;
  border-left-style: solid;
  }
- p {
  border: 2px solid red;
  border-radius: 5px;
  }

• CSS Margins

- The CSS margin properties are used to create space around elements, outside of any defined borders.
- With CSS, full control over the margins. There are properties for setting the margin for each side of an element (top, right, bottom, and left).
- CSS has properties for specifying the margin for each side of an element:
  - margin-top
  - margin-right
  - margin-bottom
  - margin-left
- All the margin properties can have the following values:
  - auto - the browser calculates the margin
  - *length* - specifies a margin in px, pt, cm, etc.
  - *%* - specifies a margin in % of the width of the containing element
  - inherit - specifies that the margin should be inherited from the parent element

- If the margin property has four values:
- margin: 25px 50px 75px 100px;
  - top margin is 25px
  - right margin is 50px
  - bottom margin is 75px
  - left margin is 100px
- **margin: 25px 50px 75px;**
  - top margin is 25px
  - right and left margins are 50px
  - bottom margin is 75px
- **margin: 25px 50px;**
  - top and bottom margins are 25px
  - right and left margins are 50px

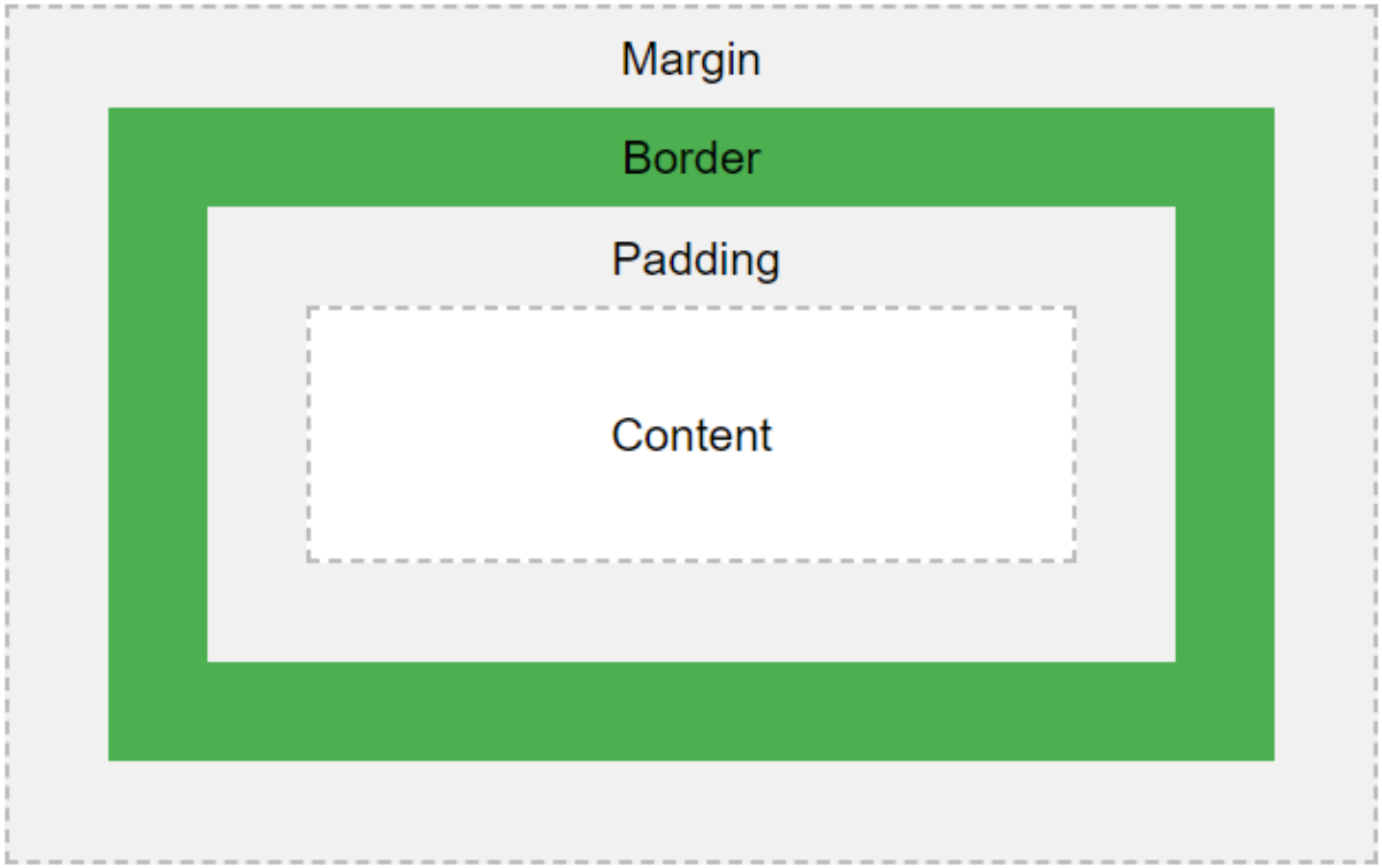
- The auto Value
- Can set the margin property to auto to horizontally center the element within its container.
- The element will then take up the specified width, and the remaining space will be split equally between the left and right margins:
  - div {
    width: 300px;
    margin: auto;
    border: 1px solid red;
    }
- The inherit Value
- This example lets the left margin of the <p class="ex1"> element be inherited from the parent element ( ):
  - div {
    border: 1px solid red;
    margin-left: 100px;
    }

    p.ex1 {
    margin-left: inherit;
    }

• The CSS Box Model

- All HTML elements can be considered as boxes. In CSS, the term "box model" is used when talking about design and layout.
- The CSS box model is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content.
- Explanation of the different parts:
  - **Content** - The content of the box, where text and images appear
  - **Padding** - Clears an area around the content. The padding is transparent
  - **Border** - A border that goes around the padding and content
  - **Margin** - Clears an area outside the border. The margin is transparent
- The box model allows us to add a border around elements, and to define space between elements.

- div {
    width: 300px;
    border: 25px solid green;
    padding: 25px;
    margin: 25px;
  }



- When you set the width and height properties of an element with CSS, just set the width and height of the **content area**. To calculate the full size of an element, you must also add padding, borders and margins.
- Assume we want to style a element to have a total width of 350px:
  - div {
    width: 320px;
    padding: 10px;
    border: 5px solid gray;
    margin: 0;
    }
  - 320px (width)
    + 20px (left + right padding)
    + 10px (left + right border)
    + 0px (left + right margin)
    = 350px

- Total element width = width + left padding + right padding + left border + right border + left margin + right margin
- Total element height = height + top padding + bottom padding + top border + bottom border + top margin + bottom margin

• CSS Outline

- An outline is a line that is drawn around elements, OUTSIDE the borders, to make the element "stand out“
- CSS has the following outline properties:
  - outline-style
  - outline-color
  - outline-width
  - outline-offset
  - outline

- The outline-style property specifies the style of the outline, and can have one of the following values:
  - dotted - Defines a dotted outline
  - dashed - Defines a dashed outline
  - solid - Defines a solid outline
  - double - Defines a double outline
  - groove - Defines a 3D grooved outline
  - ridge - Defines a 3D ridged outline
  - inset - Defines a 3D inset outline
  - outset - Defines a 3D outset outline
  - none - Defines no outline
  - hidden - Defines a hidden outline
- The outline-width property specifies the width of the outline, and can have one of the following values:
  - thin (typically 1px)
  - medium (typically 3px)
  - thick (typically 5px)
  - A specific size (in px, pt, cm, em, etc)

- The outline-offset property adds space between an outline and the edge/border of an element. The space between an element and its outline is transparent.
- p {
  margin: 30px;
  border: 1px solid black;
  outline: 1px solid red;
  outline-offset: 15px;
  }
- The text-decoration property is used to set or remove decorations from text.
- h1 {
  text-decoration: overline;
  }

  h2 {
  text-decoration: line-through;
  }

  h3 {
  text-decoration: underline;
  }

- The text-transform property is used to specify uppercase and lowercase letters in a text.
- p.uppercase {
  text-transform: uppercase;
  }

  p.lowercase {
  text-transform: lowercase;
  }

  p.capitalize {
  text-transform: capitalize;
  }
- The text-indent property is used to specify the indentation of the first line of a text:
- p {
  text-indent: 50px;
  }

- The letter-spacing property is used to specify the space between the characters in a text.
  - h1 {
    letter-spacing: 3px;
    }

    h2 {
    letter-spacing: -3px;
    }
- The line-height property is used to specify the space between lines:
  - p.small {
    line-height: 0.8;
    }

    p.big {
    line-height: 1.8;
    }
- The direction property is used to change the text direction of an element:
  - p {
    direction: rtl;
    }

- The word-spacing property is used to specify the space between the words in a text.
  - h1 {
    word-spacing: 10px;
    }

    h2 {
    word-spacing: -5px;
    }
- The text-shadow property adds shadow to text.
- The following example specifies the position of the horizontal shadow (3px), the position of the vertical shadow (2px) and the color of the shadow (red):
  - h1 {
    text-shadow: 3px 2px red;
    }

- CSS Font Families
- In CSS, there are two types of font family names:
  - **generic family** - a group of font families with a similar look (like "Serif" or "Monospace")
  - **font family** - a specific font family (like "Times New Roman" or "Arial")

Generic family	Font family	Description
Serif	Times New Roman Georgia	Serif fonts have small lines at the ends on some characters
Sans-serif	Arial Verdana	"Sans" means without - these fonts do not have the lines at the ends of characters
Monospace	Courier New Lucida Console	All monospace characters have the same width

- The font family of a text is set with the font-family property.
- The font-family property should hold several font names as a "fallback" system. If the browser does not support the first font, it tries the next font, and so on.
- Start with the font you want, and end with a generic family, to let the browser pick a similar font in the generic family, if no other fonts are available.
- p {
  font-family: "Times New Roman", Times, serif;
  }
- The font-style property is mostly used to specify italic text.
- This property has three values:
  - normal - The text is shown normally
  - italic - The text is shown in italics
  - oblique - The text is "leaning" (oblique is very similar to italic, but less supported)

- The font-size property sets the size of the text.
- Being able to manage the text size is important in web design. However, you should not use font size adjustments to make paragraphs look like headings, or headings look like paragraphs.
- Always use the proper HTML tags, like <h1> - <h6> for headings and <p> for paragraphs.
- The font-size value can be an absolute, or relative size.
- Absolute size:
  - Sets the text to a specified size
  - Does not allow a user to change the text size in all browsers (bad for accessibility reasons)
  - Absolute size is useful when the physical size of the output is known
- Relative size:
  - Sets the size relative to surrounding elements
  - Allows a user to change the text size in browsers

- Setting the text size with pixels gives you full control over the text size:
  - h1 {
    font-size: 40px;
    }

    h2 {
    font-size: 30px;
    }

    p {
    font-size: 14px;
    }
- To allow users to resize the text (in the browser menu), many developers use em instead of pixels.
- 1em is equal to the current font size. The default text size in browsers is 16px. So, the default size of 1em is 16px.
- The size can be calculated from pixels to em using this formula: pixels/16=em

- h1 {
  font-size: 2.5em; /* 40px/16=2.5em */
  }

  h2 {
  font-size: 1.875em; /* 30px/16=1.875em */
  }

  p {
  font-size: 0.875em; /* 14px/16=0.875em */
  }

- The font-weight property specifies the weight of a font:
  - p.normal {
  font-weight: normal;
  }

  p.thick {
  font-weight: bold;
  }

- Responsive Font Size
- The text size can be set with a vw unit, which means the "viewport width".
- <h1 style="**font-size:10vw**">Hello World</h1>
- The font-variant property specifies whether or not a text should be displayed in a small-caps font.
- In a small-caps font, all lowercase letters are converted to uppercase letters. However, the converted uppercase letters appears in a smaller font size than the original uppercase letters in the text.
- p.normal {
  font-variant: normal;
  }

  p.small {
  font-variant: small-caps;
  }

• Styling Links

- Links can be styled with any CSS property (e.g. color, font-family, background, etc.).
- In addition, links can be styled differently depending on what state they are in.
- The four links states are:
  - a:link - a normal, unvisited link
  - a:visited - a link the user has visited
  - a:hover - a link when the user mouse over it
  - a:active - a link the moment it is clicked
-
```
/* unvisited link */
a:link {
    color: red;
}

/* visited link */
a:visited {
    color: green;
}

/* mouse over link */
a:hover {
    color: hotpink;
}

/* selected link */
a:active {
    color: blue;
}
```

- When setting the style for several link states, there are some order rules:
  - a:hover MUST come after a:link and a:visited
  - a:active MUST come after a:hover
- a:link {
  text-decoration: none;
  }

  a:visited {
  text-decoration: none;
  }

  a:hover {
  text-decoration: underline;
  }

  a:active {
  text-decoration: underline;
  }

- a:link {
  background-color: yellow;
  }
  a:visited {
  background-color: cyan;
  }
  a:hover {
  background-color: lightgreen;
  }
  a:active {
  background-color: hotpink;
  }

- HTML Lists and CSS List Properties
- The CSS list properties allow you to:
  - Set different list item markers for ordered lists
  - Set different list item markers for unordered lists
  - Set an image as the list item marker
  - Add background colors to lists and list items
- The list-style-type property specifies the type of list item marker.
- ul.a {
  list-style-type: circle;
  }

  ul.b {
  list-style-type: square;
  }

  ol.c {
  list-style-type: upper-roman;
  }

  ol.d {
  list-style-type: lower-alpha;
  }

- The list-style-image property specifies an image as the list item marker:
- ul {
  list-style-image: url('sqpurple.gif');
  }
- The list-style-position property specifies the position of the list-item markers (bullet points).
- "list-style-position: outside;" means that the bullet points will be outside the list item. The start of each line of a list item will be aligned vertically. This is default.
- "list-style-position: inside;" means that the bullet points will be inside the list item. As it is part of the list item, it will be part of the text and push the text at the start.
- Example
- Example

- **CSS Table**

- table, th, td {
  border: 1px solid black;
  }
- table {
  border-collapse: collapse;
  }
  table, th, td {
  border: 1px solid black;
  }
- table {
  width: 100%;
  }

  th {
  height: 50px;
  }
- th {
  text-align: left;
  }
- td {
  height: 50px;
  vertical-align: bottom;
  }

- th, td {
  padding: 15px;
  text-align: left;
  }
- th, td {
  border-bottom: 1px solid #ddd;
  }
- Use the :hover selector on <tr> to highlight table rows on mouse over
- tr:hover {background-color: #f5f5f5;}
- For zebra-striped tables, use the nth-child() selector and add a background-color to all even (or odd) table rows
- tr:nth-child(even) {background-color: #f2f2f2;}

- A responsive table will display a horizontal scroll bar if the screen is too small to display the full content
- Add a container element (like) with overflow-x: auto around the <table> element to make it responsive
- Example

• The display Property

- The display property specifies if/how an element is displayed.
- Every HTML element has a default display value depending on what type of element it is. The default display value for most elements is block or inline.
- Block-level Elements
- A block-level element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can).
- Examples of block-level elements:
  -
  - <h1> - <h6>
  - <p>
  - <form>
  - <header>
  - <footer>
  - <section>

- Inline Elements
- An inline element does not start on a new line and only takes up as much width as necessary.
- Examples of inline elements:
  -
  - <a>
  - <img>
- Display: none;
- display: none; is commonly used with JavaScript to hide and show elements without deleting and recreating them. Take a look at our last example on this page if you want to know how this can be achieved.
- The <script> element uses display: none; as default.

- Override The Default Display Value
- As mentioned, every element has a default display value. However, you can override this.
- Changing an inline element to a block element, or vice versa, can be useful for making the page look a specific way, and still follow the web standards.
- A common example is making inline <li> elements for horizontal menus:
- li {
  display: inline;
  }

```
<!DOCTYPE html>
<html>
<head>
<style>
h1.hidden {
  display: none;
}
</style>
</head>
<body>
<h1>This is a visible heading</h1>
<h1 class="hidden">This is a hidden heading</h1>
<p>Notice that the h1 element with display: none; does not take up any space.</p>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
<style>
h1.hidden {
  visibility: hidden;
}
</style>
</head>
<body>
<h1>This is a visible heading</h1>
<h1 class="hidden">This is a hidden heading</h1>
<p>Notice that the hidden heading still takes up space.</p>
</body>
</html>
```

- CSS Layout - width and max-width
- A block-level element always takes up the full width available (stretches out to the left and right as far as it can).
- Setting the width of a block-level element will prevent it from stretching out to the edges of its container.
- Then, you can set the margins to auto, to horizontally center the element within its container.
- The element will take up the specified width, and the remaining space will be split equally between the two margins:
- This element has a width of 500px, and margin set to auto.
- The problem with the above occurs when the browser window is smaller than the width of the element. The browser then adds a horizontal scrollbar to the page.
- Using max-width instead, in this situation, will improve the browser's handling of small windows. This is important when making a site usable on small devices:
- This element has a max-width of 500px, and margin set to auto.

- Example
- CSS Layout - 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.

- position: static;
- HTML elements are positioned static by default.
- 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:
- Example
- 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.
- Example

- position: fixed;
- An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled. 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.
- Notice the fixed element in the lower-right corner of the page. Here is the CSS that is used:
- Example
- position: absolute;
- An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).
- However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.
- Example

- position: sticky;
- An element with position: sticky; is positioned based on the user's scroll position.
- A 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).
- Example

CSS Pseudo-classes

- A Pseudo class in CSS is used to define the special state of an element. It can be combined with a CSS selector to add an effect to existing elements based on their states.
- For Example, changing the style of an element when the user hovers over it, or when a link is visited. All of these can be done using Pseudo Classes in CSS.
- For example, it can be used to:
  - Style an element when a user mouses over it
  - Style visited and unvisited links differently
  - Style an element when it gets focus
- selector:pseudo-class {
  property:value;
  }

- **:hover Pseudo-class:** This pseudo-class is used to add special effect to an element when our mouse pointer is over it.
- The below example demonstrates that when your mouse enters the box area, its background color changes from yellow to orange.
- Example

- **:active Pseudo-class:** This pseudo-class is used to select an element which is activated when the user clicks on it.
- The following example demonstrates that when you click on the box, its background color changes for a moment.
- Example

- **:focus Pseudo-class:** This pseudo-class is used to select an element which is currently focused by the user.
- It works on user input elements used in forms and is triggered as soon as the user clicks on it.
- In the following example, the background color of the input field which is currently focused changes.
- Example

- **:visited Pseudo-class:** This pseudo-class is used to select the links which have been already visited by the user.
- In the following example, the color of the link changes once it is visited.
- Example