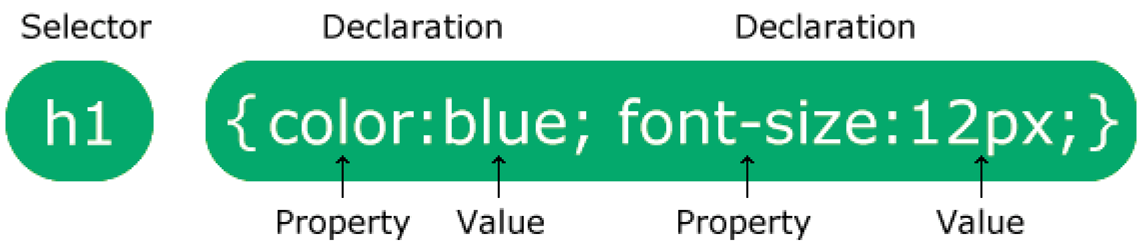
CSS is the language we use to style an HTML document. CSS describes how HTML elements should be displayed.

- CSS stands for Cascading Style Sheets
- CSS describes how HTML elements are to be displayed on screen, paper, or in other media
- CSS saves a lot of work. It can control the layout of multiple web pages all at once
- External stylesheets are stored in CSS files

CSS is used to define styles for your web pages, including the design, layout and variations in display for different devices and screen sizes.



CSS Selectors:

The selector points to the HTML element you want to 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.

Selector	**Example**	**Example description**
#id	#firstname	Selects the element with id="firstname"

.class	.intro	Selects all elements with class="intro"
element.class	p.intro	Selects only <p> elements with class="intro"
*	*	Selects all elements
element	p	Selects all <p> elements
element,element,..	div, p	Selects all <div> elements and all <p> elements

How To Add CSS:

There are three ways of inserting a style sheet:

- External CSS
- Internal CSS
- Inline CSS

1. **External CSS:**

   With an external style sheet, you can change the look of an entire website by changing just one file!

   Each HTML page must include a reference to the external style sheet file inside the <link> element, inside the head section.

   External styles are defined within the <link> element, inside the <head> section of an HTML page.

2. **Internal CSS:**

   An internal style sheet may be used if one single HTML page has a unique style.

The internal style is defined inside the <style> element, inside the head section.

3. **Inline CSS:**

   An inline style may be used to apply a unique style for a single element.

   To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.

   Inline styles are defined within the "style" attribute of the relevant element.

CSS Colors:

1. **Background Color**
2. **Text Color**
3. **Border Color**
4. **Color Values:**

   Colors can also be specified using RGB values, HEX values, HSL values.

   (1)RGB: rgb(255, 0, 0)
   (2)HEX: #ff0000
   (3)HSL: hsl(0, 100%, 50%)

CSS Background:

a) **Background Color:** The `background-color` property specifies the background color of an element.

b) **Opacity:** The `opacity` property specifies the opacity/transparency of an element. It can take a value from 0.0 - 1.0. The lower value, the more transparent.

c) **Background Image:** The `background-image` property specifies an image to use as the background of an element. By default, the image is repeated so it covers the entire element.

d) Background Repeat: If the image above is repeated only horizontally (background-repeat: repeat-x;). To repeat an image vertically, set (background-repeat: repeat-y;). If the background image used only once we used `background-repeat: no-repeat;`

CSS Borders:

CSS border properties allow you to specify the style, width, and color of an element's border.

a) Border Style: 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

b) **Border Width:** 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.

c) **Border Color:** The color can be set by:

- name - specify a color name, like "red"
- HEX - specify a HEX value, like "#ff0000"
- RGB - specify a RGB value, like "rgb(255,0,0)"
- HSL - specify a HSL value, like "hsl(0, 100%, 50%)"
- transparent

d) Border Radius: The `border-radius` property is used to add rounded borders to an element.

CSS Margins:

Margins are used to create space around elements, outside of any defined borders.

CSS has properties for specifying the margin for each side of an element.

- `margin-top`
- `margin-right`
- `margin-bottom`

- `margin-left`

CSS Padding:

Padding is used to create space around an element's content, inside of any defined borders.

CSS has properties for specifying the padding for each side of an element:

- `padding-top`
- `padding-right`
- `padding-bottom`
- `padding-left`

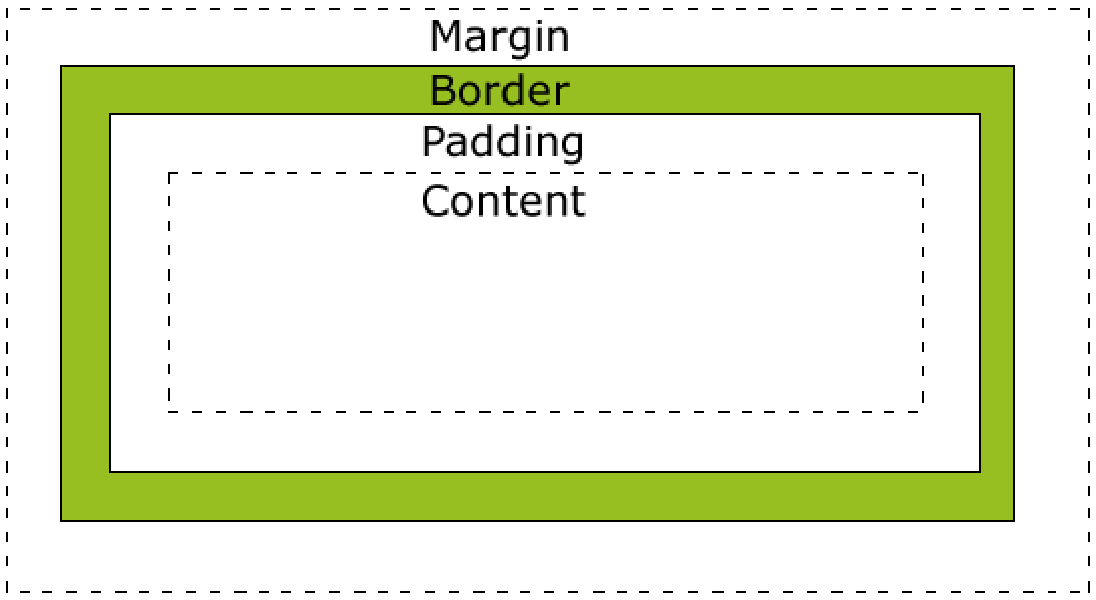
CSS Height, Width and Max-width:

The CSS `height` and `width` properties are used to set the height and width of an element. The CSS `max-width` property is used to set the maximum width of an element.

CSS Box Model:

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. The image below illustrates the box model.



- **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.

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`

CSS Outline Style:

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

CSS Outline Width:

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)

CSS Outline Color:

The `outline-color` property is used to set the color of the outline.

The color can be set by:

- name - specify a color name, like "red"
- HEX - specify a hex value, like "#ff0000"
- RGB - specify a RGB value, like "rgb(255,0,0)"
- HSL - specify a HSL value, like "hsl(0, 100%, 50%)"

CSS Outline Offset:

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;
}
```

CSS Text Color:

The `color` property is used to set the color of the text. The color is specified by:

- a color name - like "red"
- a HEX value - like "#ff0000"
- an RGB value - like "rgb(255,0,0)"

CSS Text Alignment:

- `text-align`
- `text-align-last`

Text Alignment:

The `text-align` property is used to set the horizontal alignment of a text.

A text can be left or right aligned, centered, or justified.

```
div {
  border: 1px solid black;
  padding: 10px;
  width: 200px;
  height: 200px;
  text-align: justify;
}
```

Text Align Last:

The `text-align-last` property specifies how to align the last line of a text.

It has right, left, justify.

CSS Text Decoration:

- `text-decoration-line`
- `text-decoration-color`
- `text-decoration-style`
- `text-decoration-thickness`
- `text-decoration`

Decoration Line to Text:

The `text-decoration-line` property is used to add a decoration line to text.

It has overline, line-through, underline, overline underline.

Color for the Decoration Line:

The `text-decoration-color` property is used to set the color of the decoration line. It is used with text decoration line.

Style for the Decoration Line:

The `text-decoration-style` property is used to set the style of the decoration line. It is used with text decoration line.

Thickness for the Decoration Line:

The `text-decoration-thickness` property is used to set the thickness of the decoration line. It is used with text decoration line.

Text Transformation:

The `text-transform` property is used to specify uppercase and lowercase letters in a text. It can be used to turn everything into uppercase or lowercase letters, or capitalize the first letter of each word.

CSS Text Spacing:

- `text-indent`
- `letter-spacing`
- `line-height`
- `word-spacing`
- `white-space`

Text Indentation:

The `text-indent` property is used to specify the indentation of the first line of a text.

Letter Spacing:

The `letter-spacing` property is used to specify the space between the characters in a text.

Line Height:

The `line-height` property is used to specify the space between lines.

Word Spacing:

The `word-spacing` property is used to specify the space between the words in a text.

White Space:

The `white-space` property specifies how white-space inside an element is handled.

```
p {
  white-space: nowrap;
}
```

Text Shadow:

The `text-shadow` property adds shadow to text. In its simplest use, you only specify the horizontal shadow (2px) and the vertical shadow (2px)

CSS Fonts:

Choosing the right font has a huge impact on how the readers experience a website. The right font can create a strong identity for your brand. Using a font that is easy to read is important. The font adds value to your text.

It is also important to choose the correct color and text size for the font.

1. **Serif** fonts have a small stroke at the edges of each letter. They create a sense of formality and elegance.
2. **Sans-serif** fonts have clean lines (no small strokes attached). They create a modern and minimalistic look.
3. **Monospace** fonts - here all the letters have the same fixed width. They create a mechanical look.
4. **Cursive** fonts imitate human handwriting.
5. **Fantasy** fonts are decorative/playful fonts.

CSS Web Safe Fonts:

Web safe fonts are fonts that are universally installed across all browsers and devices.

- Arial (sans-serif)
- Verdana (sans-serif)
- Helvetica (sans-serif)
- Tahoma (sans-serif)
- Trebuchet MS (sans-serif)
- Times New Roman (serif)
- Georgia (serif)
- Garamond (serif)
- Courier New (monospace)
- Brush Script MT (cursive)

Fallback Fonts:

However, there are no 100% completely web safe fonts. There is always a chance that a font is not found or is not installed properly.

Therefore, it is very important to always use fallback fonts.

This means that you should add a list of similar "backup fonts" in the `font-family` property. If the first font does not work, the browser will try the next one, and the next one, and so on. Always end the list with a generic font family name.

- **Serif**
- **Sans-serif**
- **Monospace**
- **Cursive**
- **Fantasy**

Font Style:

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)

Font Weight:

The `font-weight` property specifies the weight of a font. It has normal, bold.

Font Variant:

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

Font Size:

The `font-size` property sets the size of the text.

Font Size With Pixels:

Setting the text size with pixels gives you full control over the text size.

(16px = 1em = 100%)

Font Size With Em:

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*

Font Size with Percentage:

We also can use the font size with the help of percentage.

Google Fonts:

Google Fonts are free to use, and have more than 1000 fonts to choose from.

Just add a special style sheet link in the <head> section and then refer to the font in the CSS.

```
<head>
<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=So
fia">
<style>
body {
  font-family: Sofia;
}
</style>
</head>
```

Font Awesome Icons:

To use the Font Awesome icons, go to fontawesome.com, sign in, and get a code to add in the `<head>` section of your HTML page:

```
<script src="https://kit.fontawesome.com/0d639a0714.js"
crossorigin="anonymous"></script>
```

For ex: `<i class="fas fa-cloud"></i>`

Bootstrap Icons:

To use the Bootstrap glyphicons, add the following line inside the `<head>` section of your HTML page:

```
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

<i class="glyphicon glyphicon-cloud"></i>
```

Google Icons:

To use the Google icons, add the following line inside the `<head>` section of your HTML page:

```
<link rel="stylesheet"
href="https://fonts.googleapis.com/icon?family=Material+Icons">
```

```
<i class="material-icons">cloud</i>
```

CSS Links:

With CSS, links can be styled in many different ways. Links can be styled with any CSS property (e.g. `color`, `font-family`, `background`, etc.).

The links states are:

- `a:visited` - a link the user has visited
- `a:hover` - a link when the user mouses over it
- `a:active` - a link the moment it is clicked

CSS Display Property:

The `display` property is the most important CSS property for controlling layout.

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:

- <div>
- <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:

- <span>
- <a>
- <img>

Display: none;

`display: none;` is commonly used with JavaScript to hide and show elements without deleting and recreating them. The `<script>` element uses `display: none;` as default.

CSS Layout - width and max-width:

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 <div> element has a width of 500px, and margin set to auto.

Note: The problem with the `<div>` 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 <div> element has a max-width of 500px, and margin set to auto.

Tip: Resize the browser window to less than 500px wide, to see the difference between the two divs!

```
div.ex1 {
  width: 500px;
  margin: auto;
  border: 3px solid #73AD21;
}
```

```
div.ex2 {
  max-width: 500px;
  margin: auto;
  border: 3px solid #73AD21;
}
```

The position Property:

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

There are different position values:

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

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.

```
div.static {
  position: static;
  border: 3px solid #73AD21;
}
```

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.

```
div.relative {
  position: relative;
  left: 30px;
  border: 3px solid #73AD21;
}
```

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.

```css
div.fixed {
  position: fixed;
  bottom: 0;
  right: 0;
  width: 300px;
  border: 3px solid #73AD21;
}
```

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).

```css
div.sticky {
  position: sticky;
  top: 0;
  background-color: green;
  border: 2px solid #4CAF50;
}
```

CSS Overflow:

The `overflow` property specifies whether to clip the content or to add scrollbars when the content of an element is too big to fit in the specified area.

The `overflow` property has the following values:

- `visible` - Default. The overflow is not clipped. The content renders outside the element's box
- `hidden` - The overflow is clipped, and the rest of the content will be invisible

- `scroll` - The overflow is clipped, and a scrollbar is added to see the rest of the content
- `auto` - Similar to `scroll`, but it adds scrollbars only when necessary

overflow: visible

By default, the overflow is `visible`, meaning that it is not clipped and it renders outside the element's box.

```
div {
  width: 200px;
  height: 65px;
  background-color: coral;
  overflow: visible;
}
```

overflow: hidden

With the `hidden` value, the overflow is clipped, and the rest of the content is hidden.

overflow: scroll

Setting the value to `scroll`, the overflow is clipped and a scrollbar is added to scroll inside the box. Note that this will add a scrollbar both horizontally and vertically (even if you do not need it)

overflow: auto

The `auto` value is similar to `scroll`, but it adds scrollbars only when necessary

Float Property:

The `float` property is used for positioning and formatting content e.g. let an image float left to the text in a container.

The `float` property can have one of the following values:

- `left` - The element floats to the left of its container
- `right` - The element floats to the right of its container
- `none` - The element does not float (will be displayed just where it occurs in the text). This is default

```
img {
  float: right;
}
```

Center Align Elements:

To horizontally center a block element (like <div>), use `margin: auto;`

Setting the width of the element will prevent it from stretching out to the edges of its container.

The element will then take up the specified width, and the remaining space will be split equally between the two margins.

```
.center {
  margin: auto;
  width: 50%;
  border: 3px solid green;
  padding: 10px;
}
```

Center Align Text:

To just center the text inside an element, use `text-align: center;`

Center an Image:

To center an image, set left and right margin to `auto` and make it into a `block` element.

```
img {
  display: block;
  margin-left: auto;
  margin-right: auto;
  width: 40%;
}
```