CSS

1. What is CSS ?

CSS (Cascading Style Sheets) is the code that styles web content.
- **Cascading** refers to the way CSS applies one style on top of another.
- **Style Sheets** control the look and feel of web documents.

Like HTML, CSS is not a programming language. It's not a markup language either.

CSS and HTML work hand in hand:
- HTML sorts out the page structure.
- CSS defines how HTML elements are displayed.

2. Why Use CSS ?

CSS helps you to keep the informational content of a document separate from the details of how to display it. The details of how to display the document are known as its style. You keep the style separate from the content so that you can:

- Avoid duplication
- Make maintenance easier
- Use the same content with different styles for different purposes

Your web site might have thousands of pages that look similar. Using CSS, you store the style information in common files that all the pages share. When a user displays a web page, the user’s browser loads the style information along with the content of the page. When a user prints a web page, you might provide different style information that makes the printed page easy to read.

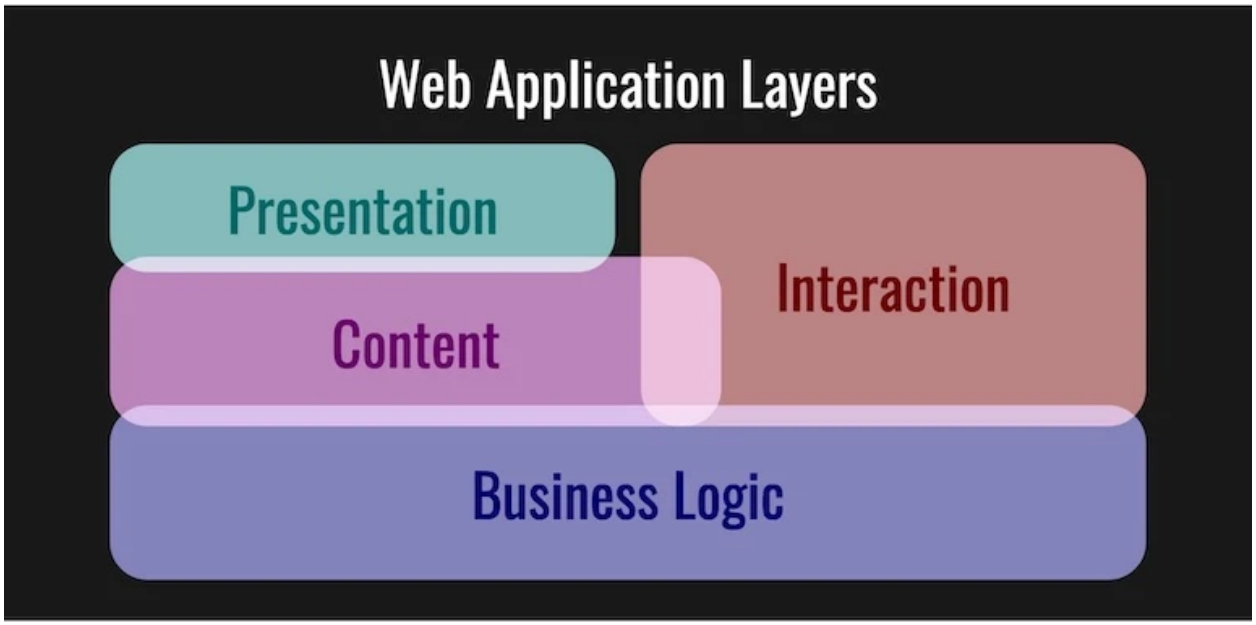
In general, you use HTML to describe the content of the document, not its style; you use CSS to specify its style, not its content. There are exceptions to this rule, of course, and HTML also provides some ways to specify style. For example, in HTML you can use a <b> tag to make text bold, and you can specify the background colour of a page in its <body> tag. When you use CSS, you normally avoid using these HTML style features so that all your document’s style information is in one place.

3. How CSS plays its role in Front End Development ?

A web application is comprised of four parts:

1. Business logic
2. Content (HTML)
3. Interaction (JavaScript)
4. Presentation (CSS)

Business logic is the most ambiguous, so for this article, we'll assume that it's server-side code, or code that interacts with a server.



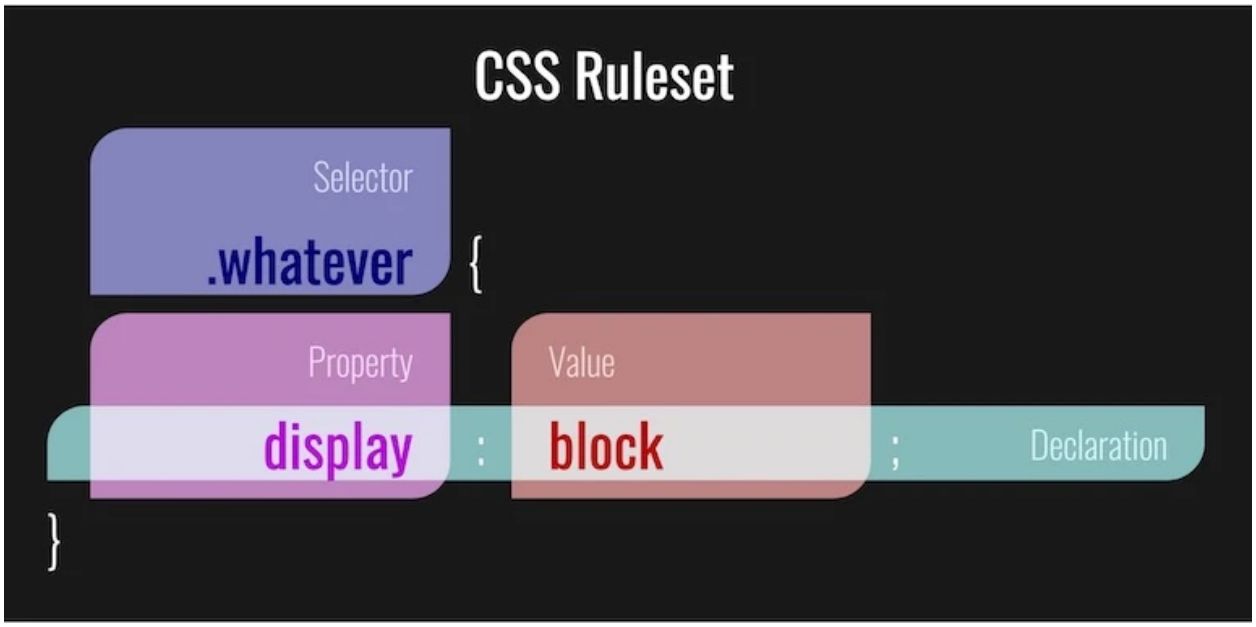
Each time a web page is requested by a browser, the business logic sends down HTML, which we'll call content. This content is unstyled black text on a white background at this point in time.

As the HTML is loaded, the browser will load more assets, like images and videos, but more importantly, JavaScript and CSS.

CSS uses the structure of the HTML to style the webpage. When the CSS code is read, the content transforms from black text on a white background to whatever creation has been described by the collection of styles.

4. CSS Rules

The foundation of writing scalable CSS for our web application is a ruleset. A ruleset contains one or more selector(s) and a declaration block with a collection of declarations. Declarations contain properties and values in a key/value pair syntax.



Selectors are the interface that CSS uses to match our rulesets to elements in our HTML document. Multiple selectors and rulesets might apply to the same element, and this is where the 'cascade' part of CSS comes into play.

The cascade is the referee that determines which properties are applied to a given element, with considerations to the specificity of the selector and possible inherited properties. We need a referee to determine which styles take precedence in the event of a rule conflict.
As a rule of thumb, the cascade determines which properties apply in this order:

1. Level of selector specificity
2. Source order of rulesets
3. Inherited values

For an in-depth approach to the cascade, check out this MDN article on the cascade and inheritance, or to go even deeper, read the latest spec on cascading and inheritance.
For an introduction on just the topic of specificity, CSS Tricks has an excellent article called the Specifics on CSS Specificity.

5. Basics of Styling

Before getting into the meatier concepts of CSS, let's cover a few fundamentals when it comes to styling and rendering content.

A. The concept of rectangles

When styling content, there are a few things to keep in mind to help build a solid mental model of how our content will be rendered:

1. Everything is a rectangle. Even if it doesn't look like a rectangle, it's bounded by one.
2. Every rectangle relates to another rectangle in some way or another.
3. Nested (child) rectangles are above their containing (parent) rectangles by default.

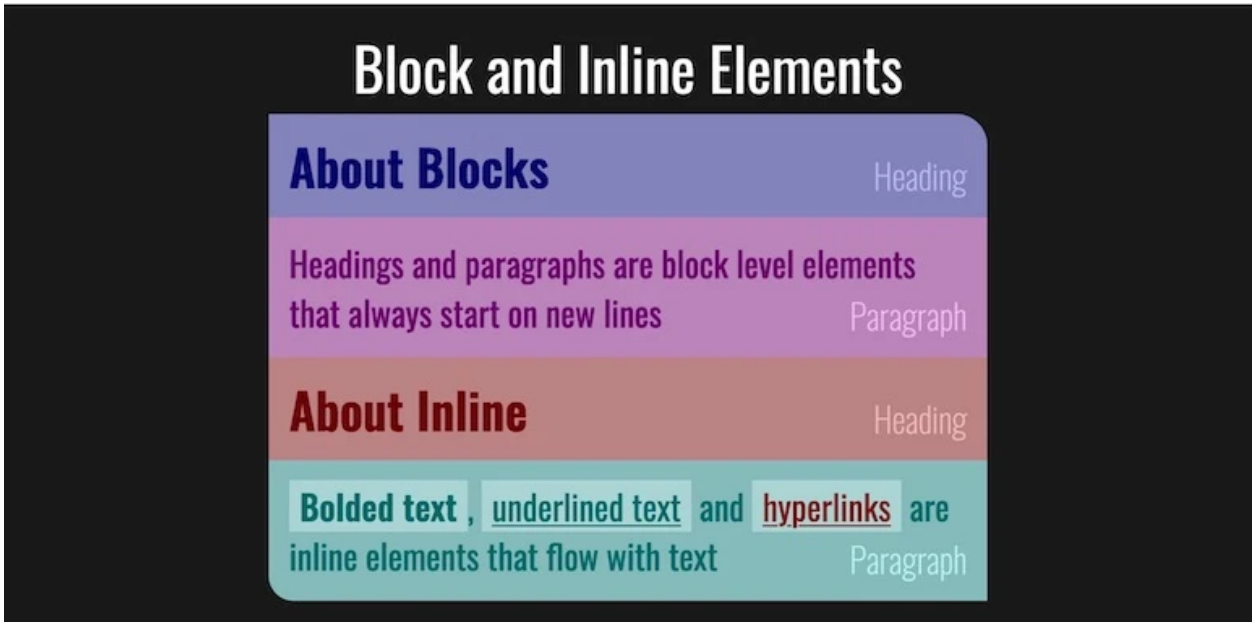
Setting these guidelines will help you when you are laying out web pages.

B. The **display** property

The first property we should learn about in this intro to CSS is the display property. There are a handful of different values for display, and we'll cover some of them below, but the two we should be aware of right now are block and inline. All HTML elements by default will have one of these two values.

Block-level elements will be rendered from the top to bottom of their parent rectangle, with each block-level sibling starting below the previous element.

Think about a heading, followed by a paragraph, another heading, and then a final paragraph. We view these elements from top to bottom.



Inline elements will be rendered from left to right (or right to left for languages natively read this way), with their contents wrapping to the next line and their next inline sibling starting after the previous content.

Think about some bold text, followed by underlined text, a link, and then some normal text. We view these elements in a reading direction and then top to bottom.

C. Document flow

The basics of document flow are how block-level and inline elements are rendered. The important thing to know is that rectangles are placed from top to bottom or in a reading direction, for each display type respectively.

However, there are times when we don't want our content to follow this straightforward approach to document flow, so CSS provides different ways to manipulate it. We'll cover those specifics later on but for now, just being aware of this possibility is good enough.

D. Typography

The foundation of the web is content, and the simplest form of content is text. When working with typography on the web, there are several things you need to keep in mind to make your designs practical and appealing.

For readability and accessibility, we need to consider the contrast of text color to the background color and the size of the font depending on the font family. For pixel perfection, it's important to know that line-height controls line spacing while font-size has less influence.

Related properties

Here are a list of related properties to be familiar with:

- font-family: A prioritized list of fonts, where the first font available will be used
- color: The color of the text
- text-align: The alignment of the text in the element
- font-size: The height of the text itself
- line-height: The height of the line of text, where the invisible space is based on this and font-size
- font-weight: The weight of the text, such as bold
- font-style: The style of the text, such as italic
- text-decoration: The decoration of the text, such as underlined
- letter-spacing: The space in between letters
- word-spacing: The space in between words
- text-shadow: The shadow effects around the text

Note: All of these typography-related properties will inherit, meaning that if we set them on a parent ruleset, they will apply to all their descendants.

E. Comments

Comments are used to explain your code, and may help you when you edit the source code later. Comments are ignored by browsers.

A CSS comment look like this:

`/* Comment goes here */`

Example:

```
p {
 color: green;
 /* This is a comment */
 font-size: 150%;
}
```

Types Of Coloring :

- `rgb(red, green, blue) /*eg: rgb(255, 99,12) */`
- `Color Names /*eg: red/yellow/blue */`
- `Hex Values /*eg: ff0023 */`

Background

1. Background-color :

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

2. Background-image:

```
        body {
     background-image: url("paper.gif");}
```

Borders

1. **Border Style :**

```
p.dotted{border-style: dotted;}

p.solid {border-style: solid;}

p.double {border-style: double;}
```

2. Border Color :

Note: If `border-color` is not set, it inherits the color of the element.

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

Margin

```
p {
       margin-top: 100px;
       margin-bottom: 100px;
       margin-right: 150px;
      margin-left: 80px;}
```

Padding

```
div {
      padding-top: 50px;
      padding-right: 30px;
```

```
padding-bottom: 50px;

padding-left: 80px;

}
```

CSS Height And Width

The height and width properties may have the following values:

- auto - This is default. The browser calculates the height and width
- length - Defines the height/width in px, cm etc.
- % - Defines the height/width in percent of the containing block
- initial - Sets the height/width to its default value
- inherit - The height/width will be inherited from its parent value

```
div {

height: 200px;

width: 50%;

background-color: powderblue;}
```

CSS Selectors

- CSS Element Selector

```
p {

  text-align: center;

  color: red;
```

- CSS “id” Selector

```
#para1 {
  text-align: center;
  color: red;
}
```

- CSS “class” Selector

```
.center {
  text-align: center;
  color: red;
}
```

Text Stylings

I Am Giving Some Examples Here Related To What We Learn Today

```
h1 {
  color: green; /*text color*/
  text-align: center; /*text alignment */
  text-decoration: none;
}
```

Font Family

```
serif {
```

```
        font-family: "Times New Roman", Times, serif;
        }

    .sansserif {
        font-family: Arial, Helvetica, sans-serif;
        }

    .monospace {
        font-family: "Lucida Console", Courier, monospace;
        }
```

Font Styles

```
        p.normal {
  font-style: normal;
}

p.italic {
  font-style: italic;
}
```

```
p.oblique {
  font-style: oblique;
}
```

6. An HTML Page With or Without CSS

NOTE : In this exercise, we have just scratched the surface of CSS.