UNIT- III

Introduction to CSS3:

What is CSS3? Differences between CSS3 and earlier CSS, Specifications How browsers are handling CSS3? CSS3 Selectors: Selectors Overview Explore specific selectors, Designing and Developing with CSS3: Background and color, Typography,CSS3 Box Model, Page layout, Media Queries, Implementing CSS3, Advantages and limitations of working with CSS.

Cascading Style Sheets (CSS) is a style sheet language used for describing the look and formatting of a document written in a markup language. CSS3 is a latest standard of css earlier versions (CSS2). The main difference between css2 and css3 is follows −

- Media Queries
- Namespaces
- Selectors Level 3
- Color

CSS3 modules

CSS3 is collaboration of CSS2 specifications and new specifications, we can called this collaboration is **module**. Some of the modules are shown below −

- Selectors
- Box Model
- Backgrounds
- Image Values and Replaced Content
- Text Effects
- 2D Transformations
- 3D Transformations
- Animations
- Multiple Column Layout
- User Interface

CSS works with HTML and provides a basic style and look to the website. CSS3 is the latest version of CSS. **CSS3** provides JavaScript and mobile development features with additional features such as **transitions, gradients**, and more.

Some of the essential differences between CSS and CSS3 are:

Features	CSS	CSS3
Compatibility	The primary purpose of CSS was to focus on formatting features. They have capabilities for positioning texts and objects. CSS is backward compatible with CSS3.	When CSS3 code is written in CSS, it is invalid. CSS3 makes the Web Pages more attractive. It takes less time to create a page.
Design	CSS does not support responsive design.	CSS3 is the latest version and supports the responsive design.
Modules	CSS is not divided into modules.	CSS3 could split into modules.
Animation	CSS cannot produce 3D animation and transformation.	In CSS3, the animation and 3D transformations are used. Elements are moved across the screen with the assistance of JavaScript and flash. By using the elements, also be ready to change the size and color. All kinds of **transformations** and **animations** are performed by using CSS3.
Rounded corners and gradient	When CSS3 was launched, the developers designed some images that looked like the rounded corners with the **structures** and **backgrounds**. Developers are designing a **border** and **uploading** the design to the server.	In CSS3 the developer writes the code like: round border {border-radius: 20px}. They have not been sent by any server and perform any other activities. Gradients will also be set by using the code which is given below: gradBG {Background:linear-gradient(red,black);}
Text Effects and Text Animations	In CSS, animations are written in **JavaScript** and **JQuery**. It has not to design layer features and page elements. It had no special effects such as **shadowing text, text animation,** etc.	In CSS3, the developer adds **text-shadows** to make it easy and effective. They add words for the visual effects of the break line and a comfortable fit inside the column. It changes the size and color of the text.
Capacity	CSS is slower.	CSS3 is faster than CSS.
Color	CSS provides unique color schemas and standard color.	CSS3 supports **HSL RGBA, HSLA** and the gradient colors**.**

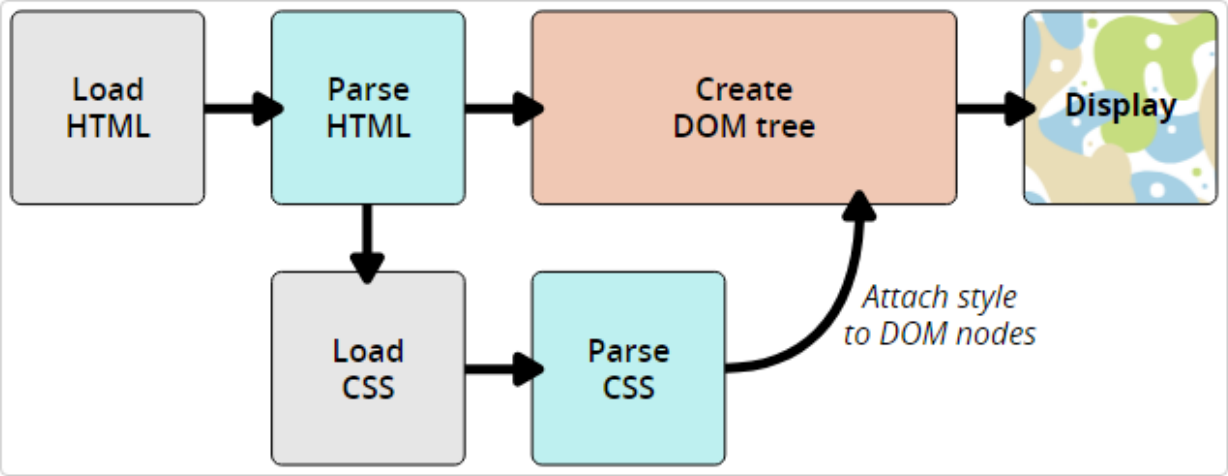
Blocks	Multi-column text blocks are defined in **CSS3**.	CSS supports single text blocks.
Lists	CSS allows a user to: 1. It set different lists for ordered lists 2. CSS set an image for a list item marker 3. CSS add background colors to the list and list items. Some list item markers are**: list-style-type**. These can be set **circle, square,** etc.	In CSS3, The list item has a counter and affected by counter increment and counter reset properties. In CSS if we use list in **CSS3** the '**display**' property must have a list defined in it. CSS3 cannot support a numbering system. The list style image property enables an image is set against the style type marker. It will set as inside the box or outside the box.

How does CSS actually work?

When a browser displays a document, it must combine the document's content with its style information. It processes the document in a number of stages, which we've listed below. Bear in mind that this is a very simplified version of what happens when a browser loads a webpage, and that different browsers will handle the process in different ways. But this is roughly what happens.

1. The browser loads the HTML (e.g. receives it from the network).
2. It converts the HTML into a DOM (*Document Object Model*). The DOM represents the document in the computer's memory. The DOM is explained in a bit more detail in the next section.
3. The browser then fetches most of the resources that are linked to by the HTML document, such as embedded images, videos, and even linked CSS! JavaScript is handled a bit later on in the process, and we won't talk about it here to keep things simpler.
4. The browser parses the fetched CSS, and sorts the different rules by their selector types into different "buckets", e.g. element, class, ID, and so on. Based on the selectors it finds, it works out which rules should be applied to which nodes in the DOM, and attaches style to them as required (this intermediate step is called a render tree).
5. The render tree is laid out in the structure it should appear in after the rules have been applied to it.
6. The visual display of the page is shown on the screen (this stage is called painting).

The following diagram also offers a simple view of the process.



About the DOM

A DOM has a tree-like structure. Each element, attribute, and piece of text in the markup language becomes a DOM node in the tree structure. The nodes are defined by their relationship to other DOM nodes. Some elements are parents of child nodes, and child nodes have siblings.

Understanding the DOM helps you design, debug and maintain your CSS because the DOM is where your CSS and the document's content meet up. When you start working with browser DevTools you will be navigating the DOM as you select items in order to see which rules apply.

A real DOM representation

Rather than a long, boring explanation, let's look at an example to see how a real HTML snippet is converted into a DOM.

Take the following HTML code:

HTMLPlayCopy to Clipboard

```
<p>
  Let's use:
  <span>Cascading</span>
  <span>Style</span>
  <span>Sheets</span>
</p>
```

In the DOM, the node corresponding to our <p> element is a parent. Its children are a text node and the three nodes corresponding to our elements. The SPAN nodes are also parents, with text nodes as their children:

```
P
├── "Let's use:"
├── SPAN
|   └── "Cascading"
├── SPAN
|   └── "Style"
└── SPAN
    └── "Sheets"
```

This is how a browser interprets the previous HTML snippet — it renders the above DOM tree and then outputs it in the browser.

What is a selector?

A CSS selector is the first part of a CSS Rule. It is a pattern of elements and other terms that tell the browser which HTML elements should be selected to have the CSS property values inside the rule applied to them. The element or elements which are selected by the selector are referred to as the *subject of the selector*.

Here are some commonly used CSS3 selectors:

1. Element Selector: Selects all elements of a particular type. For example, `p` selects all `<p>` elements.

```
p {
  /* CSS styles */
}
```

2. Class Selector: Selects elements with a specific class attribute. Use a dot (`.`) followed by the class name. For example, `.btn` selects all elements with `class="btn"`.

```
.btn {
    /* CSS styles */
}
```

3. ID Selector: Selects a specific element with a unique ID attribute. Use a hash (`#`) followed by the ID name. For example, `#header` selects the element with `id="header"`.

```
#header {
    /* CSS styles */
}
```

4. Descendant Selector: Selects an element that is a descendant of another element. For example, `ul li` selects all `<li>` elements that are descendants of a `<ul>` element.

```
ul li {
    /* CSS styles */
}
```

5. Child Selector: Selects an element that is a direct child of another element. Use the greater-than sign (`>`). For example, `ul > li` selects all `<li>` elements that are direct children of a `<ul>` element.

```
ul > li {
    /* CSS styles */
}
```

6. Adjacent Sibling Selector: Selects an element that is immediately preceded by another element. Use the plus sign (`+`). For example, `h2 + p` selects all `<p>` elements that are immediately preceded by an `<h2>` element.

```
h2 + p {
    /* CSS styles */
}
```

7. General Sibling Selector: Selects elements that are siblings of another element. Use the tilde sign (`~`). For example, `h2 ~ p` selects all `<p>` elements that are siblings of an `<h2>` element.

```
h2 ~ p {

    /* CSS styles */

}
```

8. Attribute Selector: Selects elements with a specific attribute value. For example, `[type="text"]` selects all elements with `type="text"`.

```
[type="text"] {

    /* CSS styles */

}
```

9. Pseudo-Classes: These selectors target elements based on their state or position, such as `:hover` for mouse-over effects, `:nth-child()` to select elements based on their position in a parent, and `:not()` to exclude specific elements.

```
a:hover {

    /* CSS styles for hover state */

}

li:nth-child(odd) {

    /* CSS styles for odd-numbered list items */

}

input:not([type="text"]) {

    /* CSS styles for input elements that are not text inputs */

}
```

10. Pseudo-Elements: These selectors target specific parts of an element, like `::before` and `::after`, which allow you to insert content before or after an element's content.

```
 p::before {
```

```
    content: "Before Text: ";
  }
  p::after {
    content: " After Text.";
  }
```

These are some of the CSS3 selectors you can use to apply styles to HTML elements. CSS3 has expanded selectors significantly, providing more power and flexibility for styling web pages.

Example:

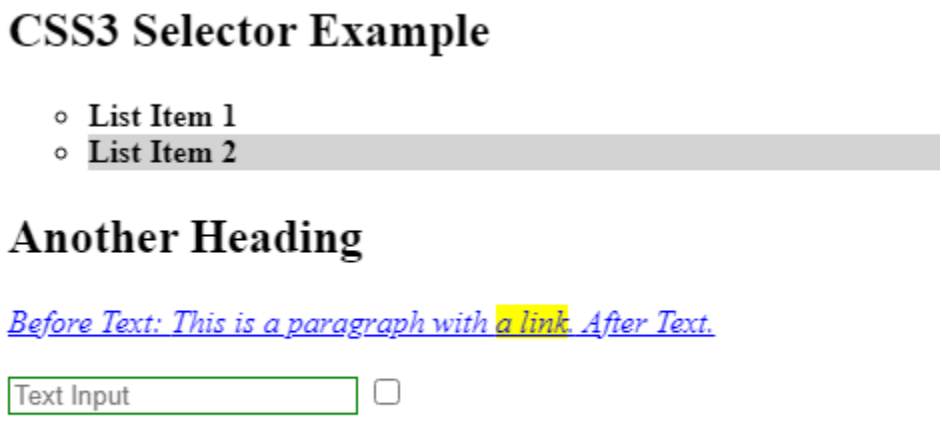
```
<!DOCTYPE html>
<html>
<head>
  <style>
    /* Element Selector */
    p {
      color: blue;
    }
    /* Class Selector */
    .btn {
      background-color: yellow;
    }
    /* ID Selector */
    #header {
      font-size: 24px;
    }
```

```
/* Descendant Selector */
ul li {
    list-style-type: circle;
}
/* Child Selector */
ul > li {
    font-weight: bold;
}
/* Adjacent Sibling Selector */
h2 + p {
    font-style: italic;
}
/* General Sibling Selector */
h2 ~ p {
    text-decoration: underline;
}
/* Attribute Selector */
[type="text"] {
    border: 1px solid green;
}
/* Pseudo-Classes */
a:hover {
    text-decoration: underline;  }
```

```
        li:nth-child(even) {
            background-color: lightgray;
        }
        input:not([type="text"]) {
            background-color: pink;
        }
        /* Pseudo-Elements */
        p::before {
            content: "Before Text: ";
        }
        p::after {
            content: " After Text.";
        }
    </style>
</head>
<body>
    <h2 id="header">CSS3 Selector Example</h2>
    <ul>
        <li>List Item 1</li>
        <li>List Item 2</li>
    </ul>
    <h2>Another Heading</h2>
    <p>This is a paragraph with <a href="#" class="btn">a link</a>.</p>
```

```
    <input type="text" placeholder="Text Input">
    <input type="checkbox">
</body>
</html>
```

Output:



3D Transformation

```
<!DOCTYPE html>
<html>
<head>
    <title>3D Transformation with Hover Effects</title>
    <style>
        .normal_div {
            width: 300px;
            height: 150px;
            color: white;
            font-size: 25px;
```

```
            background-color: green;
            border: 1px solid black;
            margin-bottom: 20px;
            transition: transform 0.3s ease, background-color 0.3s ease; /* Add transitions for transform and background-color */
        }
        #rotateX {
            width: 300px;
            height: 150px;
            color: white;
            font-size: 25px;
            background-color: green;
            border: 1px solid black;
            -webkit-transform: rotateX(180deg);
            transition: transform 0.3s ease; /* Add a transition effect for smooth rotation */
        }
        #rotateX:hover {
            -webkit-transform: rotateX(0deg); /* Rotate back to its original state on hover */
        }
        .normal_div:hover {
            background-color: blue; /* Change background color on hover */
        }
    </style>
</head>
```

```
<body>
    <center>
        <div class="normal_div">
            Normal Contents...
        </div>
        <div id="rotateX">
            180 degree rotated contents...
        </div>
    </center>
</body>
</html>
```

Output: