**1.Different between css 3.0 & css 2.0.**

Here are the key differences between CSS 2.0 and CSS 3.0:

1. Modular Design:

* CSS 2.0: CSS 2.0 is a single, large specification that defines all the styles and properties together.
* CSS 3.0: CSS 3.0 is modular, meaning it is divided into modules, each covering a specific aspect, like selectors, colors, text effects, animations, backgrounds, etc. This modular approach allows browsers to implement features gradually.

2. New Features:

* CSS 2.0: It contains basic styling properties such as fonts, colors, layout (margins, padding), and positioning (relative, absolute, fixed).
* CSS 3.0: Introduces many new features like:
  + Animations and Transitions: Smooth state changes (e.g., @keyframes, transition).
  + Rounded Corners: Property like border-radius.
  + Gradients: Linear and radial gradients (e.g., background: linear-gradient()).
  + Shadows: Text and box shadows (e.g., box-shadow, text-shadow).
  + Flexible Box Layout (Flexbox): A new layout model for responsive design.

3. Media Queries:

* CSS 2.0: Limited support for media types, such as screen, print, and speech.
* CSS 3.0: Introduces media queries, allowing styles to adapt based on screen size, resolution, and device orientation, supporting responsive web design.

4. Selectors:

* CSS 2.0: Basic selectors like class (.class), ID (#id), and pseudo-classes (e.g., :hover, :active).
* CSS 3.0: Introduces advanced selectors such as:
  + Attribute selectors ([type="text"])
  + Child combinators (>)
  + General sibling combinators (~)
  + Structural pseudo-classes (:nth-child(), :nth-of-type())

5. Text and Fonts:

* CSS 2.0: Basic font properties like font-family, font-size, and text alignment.
* CSS 3.0: Advanced text styling, including:
  + Web Fonts: The @font-face rule, allowing custom fonts.
  + Text Effects: Text shadows, custom letter spacing, and transformations (e.g., text rotation)

**2.Documentation on simple selectors.**

1. Universal Selector (\*)

* Targets all elements on the page.
* Example:

\* {

margin: 0;

padding: 0;

}

* This rule removes margin and padding from all elements

2. Type Selector (Element Selector)

* Targets all elements of a specific type (e.g., p, div, h1).
* Example:

p {

font-size: 16px;

}

* This rule applies a font size of 16px to all <p> elements.

3. Class Selector (.classname)

* Targets elements with a specific class attribute.
* Example:
* .btn {

background-color: blue;

color: white;

}

* This rule styles all elements with the class btn.

4. ID Selector (#idname)

* Targets an element with a specific ID attribute (IDs are unique).
* Example:

#header {

background-color: grey;

}

* This rule applies the background color to the element with the ID header.
* 5. Attribute Selector
* Targets elements based on their attribute values.
* Examples:

input[type="text"] {

border: 1px solid black;

}

Targets all <input> elements with type="text".

a[href^="https"] {

color: green;

}

Targets all <a> elements with an href attribute starting with "https".

6. Grouping Selector

* Allows multiple selectors to share the same style by separating them with a comma.
* Example:

h1, h2, h3 {

color: red;

}

* This rule applies the same color to all <h1>, <h2>, and <h3> elements.

7. Pseudo-Class Selector

* Targets elements based on their state or position.
* Example:

a:hover {

color: red;

}

This rule changes the color of links when they are hovered over.

**3.What is properties & values.**

**CSS Properties:**

* A **property** is a predefined attribute in CSS that you want to modify or style for an HTML element.
* Properties specify which aspect of an element you are going to change (e.g., color, font size, margin, border, etc.).

**Examples of common CSS properties**:

* color: Defines the text color.
* background-color: Defines the background color of an element.
* font-size: Specifies the size of the text.
* margin: Defines the space around an element.
* border: Sets the border around an element.

**2. CSS Values:**

* A **value** is the setting you assign to a property. It defines how the selected property will behave or be displayed.
* Values can be keywords, lengths, colors, or numbers, depending on the property they are applied to.

**Examples of common CSS values**:

* For color: Values can be keywords (red, blue), hex codes (#ff0000), RGB (rgb(255, 0, 0)), or HSL (hsl(0, 100%, 50%)).
* For font-size: Values can be units like px, em, rem, % (e.g., 16px, 1.5em).
* For margin: Values can be lengths (10px, 2em), percentages (10%), or auto (auto).

**Properties and Values Together:**

In CSS, properties and values are combined using the following syntax:

css

Copy code

property: value;

* The **property** comes first, followed by a colon (:), and then the **value**. The statement ends with a semicolon (;).

**Example:**

css

Copy code

color: blue;

font-size: 16px;

margin: 10px;

* color is the property, and blue is the value. This changes the text color to blue.
* font-size is the property, and 16px is the value. This sets the font size to 16 pixels.
* margin is the property, and 10px is the value. This adds 10 pixels of space around the element.

**1.What is advantage of inline style over  internal styles &external styles.**

**Advantages of Inline Styles:**

**a. Specificity and Priority:**

* Inline styles have the **highest priority** in CSS, meaning they will override any conflicting styles from internal or external stylesheets. This is useful when you want to ensure a particular element has a unique style that shouldn’t be affected by other styles.
* Example:

<p style="color: red;">This text will be red, no matter what.</p>

* Even if an external or internal stylesheet specifies a different color for paragraphs, the inline style (color: red;) will take precedence.

**b. Quick Styling for Specific Elements:**

* Inline styles are useful for **quick adjustments** or small, one-off changes to individual elements. You can apply styles directly within the HTML element without the need to edit or create additional CSS files.
* This is helpful for testing or making urgent tweaks when speed is essential, such as debugging a single element’s appearance.

**c. No Need for Separate CSS File:**

* Since the CSS is directly within the HTML element, there is **no need to load an external CSS file**. This can slightly reduce the complexity of your project for very small webpages or simple elements where external files would be overkill.

**d. Great for Email Templates:**

* Many email clients do not support external stylesheets or complex internal styles. In email development, inline styles are often the **only reliable way** to ensure styles are applied consistently across different email platforms.

**2.What is advantage of internal styles over inline & external.**

**1. Advantages of Internal Styles:**

**a. Easy to Maintain for Single Pages:**

* Internal styles are defined within the <style> tag in the <head> section of an HTML document. This makes it easy to maintain and manage styles for **single-page** websites or documents where all the styles are self-contained.
* Example:

<head>

<style>

p {

color: blue;

}

</style>

</head>

* This is especially useful when working on standalone pages, where all CSS can be kept in one place without needing a separate file.

**b. Less Clutter in HTML Elements (Compared to Inline Styles):**

* Unlike inline styles, internal styles are defined separately from the HTML content, making the HTML **cleaner and more readable**.
* Example:

<!-- Cleaner HTML with internal styles -->

<p>This is a paragraph with an internal style.</p>

* In contrast, inline styles would mix content and presentation, leading to cluttered and hard-to-maintain HTML:

<!-- Inline style -->

<p style="color: blue;">This is a paragraph with an inline style.</p>

**c. Global Styling for a Single Page:**

* Internal styles allow you to **apply styles globally** to multiple elements on a page without repeating them for each element (as you would with inline styles).
* With internal styles, you can create reusable styles that affect multiple elements at once, reducing redundancy and improving efficiency.
* Example:

<style>

h1, p {

color: green;

}

</style>

<h1>This is a heading</h1>

<p>This is a paragraph</p>

**d. No Additional File Requests (Compared to External Styles):**

* Unlike external stylesheets, internal styles do not require the browser to make **additional HTTP requests** to load a separate CSS file. This can slightly improve loading performance for simple or small pages since the CSS is embedded directly within the HTML file.
* In scenarios where performance is a concern and the page is small, using internal styles can be advantageous.

**e. Faster Development for Single-Page Projects:**

* Internal styles are useful when you're working on **small projects or one-off webpages**, as you don’t need to manage or maintain separate CSS files.
* This makes it easier to make quick styling adjustments without jumping between files.

**f. More Structured than Inline Styles:**

* Internal styles provide a more **structured approach** compared to inline styles. While both styles are located in the HTML file, internal styles are centralized in the <head>, making it easier to modify and update styles for the whole page.
* This contrasts with inline styles, which would need to be individually updated for every single element they are applied to.

**3.What is advantage of external styles over inline & internal.**

.Advantages of External Styles:

a. Separation of Content and Presentation:

* External stylesheets promote separation of concerns, where the HTML handles the content and structure, while the CSS file is responsible for the styling and presentation. This makes your code more organized and easier to maintain.
* The CSS file can be reused across multiple HTML pages, allowing for a clean division between content (HTML) and presentation (CSS).

b. Reusability Across Multiple Pages:

* External stylesheets are perfect for multi-page websites because the same CSS file can be linked to multiple HTML documents. This means you can define styles once and apply them across all pages, ensuring consistency across the entire website.
* If you need to make a global style change, you only need to update the external CSS file, and the change will be reflected across all linked pages.

Example:

html

Copy code

<link rel="stylesheet" href="styles.css">

* This line in the HTML file links to the styles.css file, allowing multiple HTML pages to use the same styles.

c. Easier to Maintain and Update:

* With external styles, updating or making changes to the design of your website is much easier. Since all styles are in a single CSS file (or multiple files if necessary), you don’t need to modify individual HTML documents.
* For large projects, maintaining a single CSS file (or a few CSS files) is more efficient than modifying internal or inline styles across multiple documents.

d. Faster Page Load Times (with Caching):

* External stylesheets can be cached by the browser. Once the browser downloads the CSS file for the first time, it stores it locally, so subsequent pages using the same stylesheet load faster since they don’t need to download the CSS again.
* This is especially beneficial for large websites with multiple pages that share the same styles. Caching reduces the bandwidth needed for reloading styles on each page visit.

e. Cleaner HTML Code:

* External stylesheets help keep your HTML clean and uncluttered since the styling rules are stored in a separate file. This makes the HTML easier to read and maintain.
* Unlike inline styles, which can make the HTML file messy and hard to navigate, external styles leave the HTML focused purely on structure and content.

Example of clean HTML using external styles:

html

Copy code

<head>

<link rel="stylesheet" href="styles.css">

</head>

<body>

<h1>Welcome to My Website</h1>

<p>This is a paragraph with styles applied externally.</p>

</body>

* The styles are kept separate, making the HTML easier to understand and manage.

f. Supports Modular Design:

* With external stylesheets, you can create modular CSS, breaking your styles into multiple CSS files. This makes it easier to manage large projects by dividing styles into components or sections (e.g., header styles, footer styles, etc.), leading to a more organized and maintainable project structure.
* Example:
  + header.css for header styles.
  + footer.css for footer styles.
  + main.css for general layout.

g. Improves Collaboration:

* External stylesheets make it easier for multiple developers to collaborate on a project. Designers or front-end developers can work on the CSS file, while other developers focus on HTML or JavaScript files. This separation of work reduces conflicts and increases efficiency in team projects.

h. Media Query Support for Responsive Design:

* External stylesheets are ideal for implementing responsive design using media queries. You can easily manage different style rules for different devices or screen sizes by using a single external stylesheet.
* Example:

css

Copy code

@media screen and (max-width: 600px) {

body {

background-color: lightblue;

}

}

* This way, the external stylesheet can handle multiple device layouts and adjustments without bloating the HTML file.