|  |  |  |  |
| --- | --- | --- | --- |
| Feature | Inline css | Internal css | External css |
| Location | It is used within HTML tag using the style attribute. | It is used within <head> section of HTML document. | It is used in a separate .css file. |
| Selector Scope | Affects a single element or a group of elements. | Affects multiple elements within the same HTML element. | Affects multiple HTML documents or an entire website. |
| Reusability | Not reusable. Styles need to be repeated for each element. | Can be reused on multiple elements within the same HTML document. | Can be reused on multiple HTML documents or an entire website. |
| Maintainability | Not easy to maintain. Changes need to be made manually to each element. | Relatively easy to maintain. Changes need to be made in one place in the <head> section. | Easiest to maintain. Changes need to be made in one place in the external .css file. |
| Example | <html>  <body>  <h2 style="color: green; font-size: 18px;">Welcome To GFG  </h2>  <p style="color: red; font-size: 14px;">This is some text. style by inline CSS</p>  </body>  </html> | <html>  <head>  <style>  h1 {  color: blue;  font-size: 24px;  font-weight: bold;  }  p {  color: green;  font-size: 16px;  }  </style>  </head>  <body> <h1>GeeksForGeeks</h1>  <p>GeeksForGeeks</p>  </body>  </html> | **Index.html:**  <html>  <head>  <link rel="stylesheet"  type="css" href="style.css">  </head>  <body>  <h1>GeeksForGeeks</h1>  <p>GeeksForGeeks</p>  </body>  </html>  **Style.css**  h1 {  color: blue;  font-size: 24px;  font-weight: bold;  }  p {  color: green;  font-size: 16px;  } |

1. **Explain the difference between inline,internal and external CSS.**
2. **Describe CSS selectors and list the types of selector.**

* CSS Selectors are used to target [HTML elements](https://www.geeksforgeeks.org/html-elements/) on your pages, allowing you to apply styles based on their ID, class, [type attributes](https://www.geeksforgeeks.org/html-type-attribute/), and more. There are mainly 5 types of selectors.
* Basic CSS Selectors: These are used to target elements by tag, .class, or # ID for fundamental styling needs.
* Combinators: Ideal for styling elements based on their DOM relationships (e.g., parent-child or sibling relationships).
* Group Selectors: These are used to apply the same styles to multiple unrelated elements simultaneously.
* Attribute Selectors: Perfect for styling elements based on specific attributes or values, such as form inputs or links with certain prefixes or states.
* Pseudo-Classes: Best for styling elements dynamically or interactively, like: hover for user interaction, or:nth-child() for structural styling.
* Types of CSS Selectors
* Basic Selectors
* Basic selectors in [CSS](https://www.geeksforgeeks.org/css-tutorial/) are simple tools used to target specific HTML elements for styling. These include selecting by element name (e.g., h1), class (.class Name), ID (#idName), or universally (\* for all elements).

1. Universal Selector (\*): Selects all elements on the page and applies the same style universally. For example, setting the font color for every element.

Example:

\* {

color: red;

}

1. [Element Selector](https://www.geeksforgeeks.org/css-element-selector/): Targets all elements of a specific type, such as paragraphs or headers. For example, setting a common font size for all paragraphs

Example:

<style>

p {

font-size: 16px;

}

</style>

1. [Class Selector (.)](https://www.geeksforgeeks.org/css-class-selector/): Applies styles to elements with a specific[class attribute](https://www.geeksforgeeks.org/html-class-attribute/). For instance, making all buttons have a blue background.

Example:

<style>

.button {

background-color: blue;

color: white;

}

</style>

1. [ID Selector (#)](https://www.geeksforgeeks.org/css-id-selector/): Styles a single element identified by its[unique id](https://www.geeksforgeeks.org/how-to-generate-unique-id-with-node-js/). For example, changing the background color of a header.

Example:

<style>

#header {

background-color: gray;

}

</style>

* Combinator Selectors
* ond in CSS are used to define relationships between selectors, allowing you to style elements based on their hierarchy or [positioning](https://www.geeksforgeeks.org/css-positioning-elements/) in the document. Common combinators include descendant ( ), child (>), adjacent sibling (+), and general sibling (~).

1. [Descendant Selectors](https://www.geeksforgeeks.org/what-is-descendant-selector-in-css/): Targets an element inside another, such as paragraphs inside div .For example, styling paragraphs inside a div.

Example:

<style>

div p {

color: red;

}

</style>

1. [Child Selector (>)](https://www.geeksforgeeks.org/css-first-child-selector/): They only affects the [direct child elements](https://www.geeksforgeeks.org/how-to-select-only-direct-children-from-element-with-sass/) of a parent. For example, styling direct children paragraphs of a div.

Example:

<style>

div>p {

margin-left: 20px;

}

</style>

1. [Adjacent Sibling Selector (+)](https://www.geeksforgeeks.org/css-elementelement-selector-2/): Styles an element immediately following another .For example, making the first[paragraph](https://www.geeksforgeeks.org/html-paragraph/) bold after an h1.

Example:

<style>

h1+p {

font-weight: bold;

}

</style>

1. [General Sibling Selector (~)](https://www.geeksforgeeks.org/what-is-the-general-sibling-combinator/): Styles all siblings that follow a specific element. For example, italicizing all paragraphs following an h1.

Example:

<style>

h1~p {

font-style: italic;

}

</style>

* Attribute Selectors
* [Attribute selectors](https://www.geeksforgeeks.org/css-selectors/) in CSS target elements based on the presence or value of their attributes. Examples include [attr] (selects elements with the attribute), [attr="value"] (matches specific values), and [attr^="val"] (matches values starting with "val").

1. [Presence Selector](https://www.geeksforgeeks.org/css-attribute-selector/): It selects elements that contain a specific attribute. For example, styling all inputs with a [type attribute](https://www.geeksforgeeks.org/html-type-attribute/).

Example:

<style>

input[type] {

border: 2px solid black;

}

</style>

2. [Attribute Value Selector](https://www.geeksforgeeks.org/css-attribute-selector/): It targets elements with a particular attribute value. For example, styling text inputs.

Example:

<style>

input[type="text"] {

background-color: yellow;

}

</style>

3. [Substring Matching(^=)](https://www.geeksforgeeks.org/css-attribute-selector/): It matches elements where the attribute contains a substring. For example, styling links with https in their href.

Example:

<style>

a[href^="https"] {

color: green;

}

</style>

4. [Wildcard Selector (\*=)](https://www.geeksforgeeks.org/wildcard-selectors-and-in-css-for-classes/): Matches elements where the attribute value contains a specific string. For example, underlining links with example in the URL.

Example:

​ <style>

a[href\*="example"] {

text-decoration: underline;

}

</style>

Pseudo-Classes

[Pseudo-classes](https://www.geeksforgeeks.org/css-pseudo-classes/) in CSS define the special states of elements for styling. Examples include :hover (applies when an element is hovered), :first-child (targets the first child of a parent), and :nth-child(2) (targets the second child).

1. [:hover](https://www.geeksforgeeks.org/difference-between-hover-and-mouseover-methods-in-jquery/): Styles elements when the user hovers over them. For example, changing the color of a link when hovered.

Example:

<style>

a:hover {

color: red;

}

</style>

2. [:focus](https://www.geeksforgeeks.org/set-the-focus-to-html-form-element-using-javascript/): Styles the elements when the user focus on any particular element.

Example:

<style>

input:focus {

outline: 3px solid red;

}

</style>

3.[:first-child](https://www.geeksforgeeks.org/how-to-use-a-notfirst-child-selector-in-css/): Styles the element which is the first child of it's parent.

Example:

<style>

p:first-child {

color: brown;

}

</style>

4.[:last-child](https://www.geeksforgeeks.org/how-to-select-last-child-with-a-specific-class-using-css/): Style's the element which is the last child of it's parent.

Example:

<style>

p:last-child {

color:green;

}

</style>

5. [:not](https://www.geeksforgeeks.org/css-not-selector/): Helps to remove a particular element from the styling index or styling context.

Example:

<style>

p:not(.one) {

color: blue;

}

</style>

5. Pseudo-Element's

The [Pseudo Element](https://www.geeksforgeeks.org/css-pseudo-elements/)help's to access and control a specific part of an element for inserting content before an element or inserting content after an element. Targeting any specific part of a word or a sentence. It is usually used to beautify the internal content of an element.

1. [::before](https://www.geeksforgeeks.org/how-to-use-before-and-after-elements-in-tailwind-css/): It helps to insert some content before an element.

Example:

<style>

h1::before {

content: "★ "

}

</style>

2.[::after](https://www.geeksforgeeks.org/how-to-use-before-and-after-elements-in-tailwind-css/): Ithelp's to insert some content after an element.

Example:

<style>

h1:active::before {

content: " ";

color: orangered;

}

</style>

3.[::first-line](https://www.geeksforgeeks.org/css-first-line-selector/): Styles the first line of text within a [block element](https://www.geeksforgeeks.org/html-block-and-inline-elements/). Line breaks mark the beginning of a new line.

Example:

<style>

p::first-line {

color: red;

}

</style>

4. [::first-letter](https://www.geeksforgeeks.org/css-first-letter-selector/): It Styles the first-letter of a word or a sentence.

Example:

<style>

p::first-letter {

color: red;

font-size: 23px;

}

</style>

5.[::placeholder](https://www.geeksforgeeks.org/change-an-html5-input-placeholder-color-with-css/): Styles the placeholder of a specific input field.

<style>

input::placeholder {

font-size: 20x;

font-family: sans-serif;

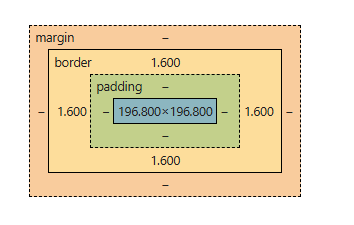
font-weight: 900;

}

</style>

1. **Discuss the CSS box model and its components.**

* The CSS Box Model defines how elements are sized, positioned, and rendered on a webpage. When a browser loads an HTML document, it creates a [DOM tree](https://www.geeksforgeeks.org/javascript/dom-document-object-model/)and assigns a box to each element. This box calculates the [element's dimensions](https://www.geeksforgeeks.org/html/how-to-get-the-elements-actual-width-and-height-in-javascript/) and [position relative](https://www.geeksforgeeks.org/css/difference-between-relative-and-absolute-position-in-css/) to its parent or the root <html> element, ensuring accurate layout and spacing.
* Box Model Component Layout:
* [Content](https://www.geeksforgeeks.org/css/how-is-border-box-different-from-content-box/): The area where text or other content is displayed.
* [Padding](https://www.geeksforgeeks.org/css/css-padding/): Space between the content and the element's border.
* [Border](https://www.geeksforgeeks.org/css/css-borders/): A frame that wraps around the padding and content.
* [Margin](https://www.geeksforgeeks.org/css/css-margins-padding/): Space between the element's border and neighbouring elements.



* Key components of the box model:
* 1. Content Area
* The content area is the central part of the CSS box model, containing the [main content](https://www.geeksforgeeks.org/html/html-main-tag/) (e.g., text, images, videos, or elements like <p> or <span>).
* It can be styled with [CSS properties](https://www.geeksforgeeks.org/css/css-properties-complete-reference/)like height and width.
* The content edge refers to the four edges of the content area
* Left content edge
* Right content edge
* Top content edge
* Bottom content edge
* 2. Padding Area
* The padding area is the space between the content and the border of an element.
* It includes the areas highlighted in light green and skin color in the example.
* The distance between the [content edge](https://www.geeksforgeeks.org/html/what-does-meta-http-equivx-ua-compatible-contentieedge-do/)and the border is the padding.
* The border marks the end of the padding area.
* The padding area contributes to the element's total dimensions.
* Padding can be adjusted using [CSS properties](https://www.geeksforgeeks.org/css/css-properties-complete-reference/).
* It works similarly with box-sizing: content-box and box-sizing: border-box, but with slight calculation differences.
* 3. Border Area
* The area that marks the end of an element is called as the[border](https://www.geeksforgeeks.org/css/css-borders/)it is the outer fencing for the element.
* The default border properties are provided in CSS to control the thickness of this outer fencing.
* The border area also add 's up to the complete[height and width](https://www.geeksforgeeks.org/html/how-to-set-the-width-and-height-of-an-image-using-html/)of the element.
* The more the border [width](https://www.geeksforgeeks.org/css/css-width-property/) the more will be the height or width of the element.
* In the above image the area marked with skin color is called the border area.
* 4. Margin Area
* The area outside the border of an element is called the [margin area](https://www.geeksforgeeks.org/css/css-box-model/).
* Basically this area depends on the parent of the element.
* The distance between the border of the parent element and the border of the child element is called as the margin.
* CSS has provides certain [margin properties](https://www.geeksforgeeks.org/css/css-margins-padding/)to get control over this scenario.
* Box Sizing Property in CSS:
* There are two type's of box-sizing properties in CSS
  1. Content-Box(default property)
* When the user set's the value of the [box-sizing property](https://www.geeksforgeeks.org/css/css-box-sizing-property/)for an element as[content-box](https://www.geeksforgeeks.org/css/how-is-border-box-different-from-content-box/)or even if user do not set's it ,it remains by default as content-box and in the actual height and width of the element the dimensions of the content area as well as the padding area is added to constitute the final dimensions of the element.
* Example

<style>

div {

height: 200px;

width: 200px;

box-sizing: content-box;

padding-left: 20px;

padding-right: 20px;

border-left: 2px solid red;

border-right: 2px solid red;

}

</style>

<!--Driver Code Starts-->

</head>

<body>

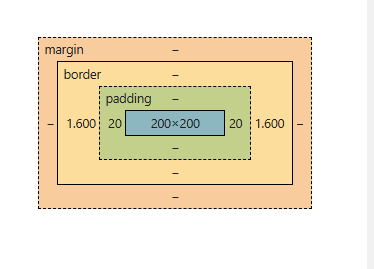
<div>Hello GFG</div>

</body>

</html>

<!--Driver Code Ends-->

* This code will create a box model with a[border line](https://www.geeksforgeeks.org/css/css-font-border/)width of 0.4px always and border-area of 1.6px and padding area as 20px width on both sides of the content area.



* **Content Area (Width**) :The width of the content area is fixed at 200px.
* **Padding:**
* Padding adds extra space inside the element, around the content.
* Padding Left: 20px
* Padding Right: 20px
* Total padding width: 20px + 20px = 40px
* **Border:**
* The border, being solid, has a width, but it is calculated differently from the padding.
* Line Width of Border: 0.4px (the width of the line itself)
* Area of Border: 1.6px (the actual space the border occupies visually)
* Border width for both sides: 1.6px (left) + 1.6px (right) = 3.2px
* **Total Width :**
* Total width of the element can be calculated by adding the padding and border areas to the content area width.
* Formula for Total Width = (Padding-Left + Padding-Right + Border-Area-Left + Border-Area-Right) + Content Area Width
* Total Width = (20px + 20px + 1.6px + 1.6px) + 200px = 243.2px
* The total width of the element becomes 243.2px.
* The reason the total width is increased unexpectedly is because[box-sizing: content-box](https://www.geeksforgeeks.org/css/css-box-sizing-property/)applies the width to the content area only .The padding and border are added outside the content area, leading to an increase in the overall [width and height](https://www.geeksforgeeks.org/css/html-width-height-attribute-vs-css-width-height-property/) of the element.
* **2. Border-Box :**
* When the box-sizing property is set as [border-box](https://www.geeksforgeeks.org/css/what-is-the-use-of-box-sizing-property-in-css/) the actual dimensions of the element's remains same as that of the actual dimensions set by the user. The difference it makes is just that the size of the content area get's altered in a manner so that it could accommodate the[padding area](https://www.geeksforgeeks.org/css/css-padding/) and the[border area](https://www.geeksforgeeks.org/css/css-border-property/)so the resultant could be equal to the actual dimensions entered by the user.
* Example:

<style>

div {

height: 200px;

width: 200px;

box-sizing:border-box;

padding-left: 20px;

padding-right: 20px;

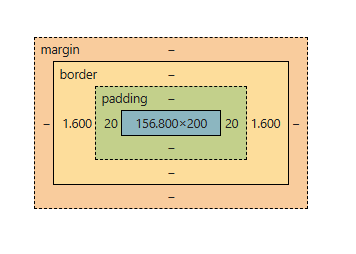
border-left: 2px solid red;

border-right: 2px solid red;

}

</style>​

* This code will create a box model by altering the dimensions specifically the width of the content area to accomodate the padding and the border area with the border line-width.



* **Width of Border and Padding Border width**: 0.4px (line width) and 1.6px + 1.6px = 3.2px (total border area).
* **Padding width**: 20px + 20px = 40px.
* **User-Entered Width** : The width entered by the user is 200px, which applies to the content area only when box-sizing: content-box is used.
* **Box-Sizing Behavior**:The box-sizing: content-box property adds the padding and border outside the content area, causing the total width to increase.
* **Adjusting Content Area Width**: To ensure the total width remains 200px, the extra width from padding and borders (40px + 3.2px = 43.2px) is subtracted from the total width.
* **New content area width** : 200px - 43.2px = 156.8px.
* **Final Width Calculation** : The final total width is: 156.8px (content area) + 40px (padding) + 3.2px (border) = 200px, ensuring the user’s entered width remains unchanged.
* **Use Case's of CSS Box Model**
* **1. Default box-sizing: content-box**
* Default behavior where padding and borders are added outside the content area, leading to an increased overall width/height.
* ​Example:

<style>

div {

width: 200px;

padding: 20px;

border: 5px solid black;

box-sizing: content-box;

background-color: lightgreen;

}

</style>

* The total width of the element will be 200px + 20px (left) + 20px (right) + 5px (left border) + 5px (right border) = 250px.