**What is a Selector in CSS?**

CSS selectors are the combination of character and special symbols that instructs the browser to apply specific CSS properties to the desired HTML element. CSS selectors use **id name (begins with #), class name, type of the HTML tag (such as heading, div, etc.), attribute, etc of the HTML element to select them.** The HTML element(s) that is/are selected by the selector is/are called the subject of the selector.

**Types of Selectors**

1. **Basic CSS Selectors**
2. **Universal Selector**
3. **Type/Element Selector**
4. **Class Selector**
5. **ID Selector**
6. **CSS Grouping Selectors**
7. **Descendant selector (space)**
8. **General Sibling selector (~)**
9. **Adjacent Sibling selector (+)**
10. **Child selector (>)**
11. **Pseudo Class Selectors**
12. **Pseudo Element Selectors**
13. **Attribute Selector Selectors**

**Basic CSS Selectors**

**Universal Selector** :It selects all the HTML elements present on the HTML page. To use the universal selector, we need to use **asterisks (\*)** and then define the CSS property and after that, all the elements will be selected. HTML elements inside other elements will also be selected, by using the CSS universal selector. (for eg.- If there is a paragraph element inside a div element, then by using the CSS universal selector, the para element which is inside the div element will also be selected)

**Syntax:**

\* {

//css properties

}

**Type Selector:**

A type selector is also known as an element selector because it selects HTML tag(s)/element(s) in your document. The element selector selects HTML elements based on their name, a few of them are p (paragraph tag), h1 (heading 1), div (division tag), span, etc. To use a type selector, you just need to give the name of the HTML tag and then in the curly braces define the CSS properties.

**Syntax:**

element\_name{

//css properties

}

**Example:**

<!DOCTYPE html>

<html>

<head>

<style>

h1 {

text-align: center;

color: red;

background-color:pink;

}

</style>

</head>

<body>

<h1>This element is selected because it is h1</h1>

</body>

</html>

**Class Selector:**

We generally use class to group HTML elements and to apply unique styling to them. The class selector in CSS selects elements that belong to a particular class attribute. To select elements with a specific class, write a period or full stop (.) character, and then the name of the class.

**Syntax:**

.class\_name{

//css properties;

}

**Example:**

In this example, we will use two different classes "marked" and "unmarked" and both will contain a different set of CSS properties. Different HTML elements will be used and then they will be grouped into classes. Now, for selecting the classes we will be using the CSS class selector. As the first element(para inside a div) belongs to the marked class, the defined CSS properties under the marked class will be applied to it. And similarly, some elements belong to either one of the classes or none. As HTML elements can also refer to more than one class the last element has both classes so that we can understand how to merge two different classes.

<!DOCTYPE html>

<html>

<head>

<style>

.marked {

background-color: pink;

color:red;

text-align:center;

font-variant:small-caps;

}

.unmarked{

color:blue;

text-align:right;

}

</style>

</head>

<body>

<div class="marked">

<p>This element belongs to class "marked"</p>

</div>

<h1>This element doesn't belong to any class</h1>

<p class="unmarked">This element belongs to class "unmarked"</p>

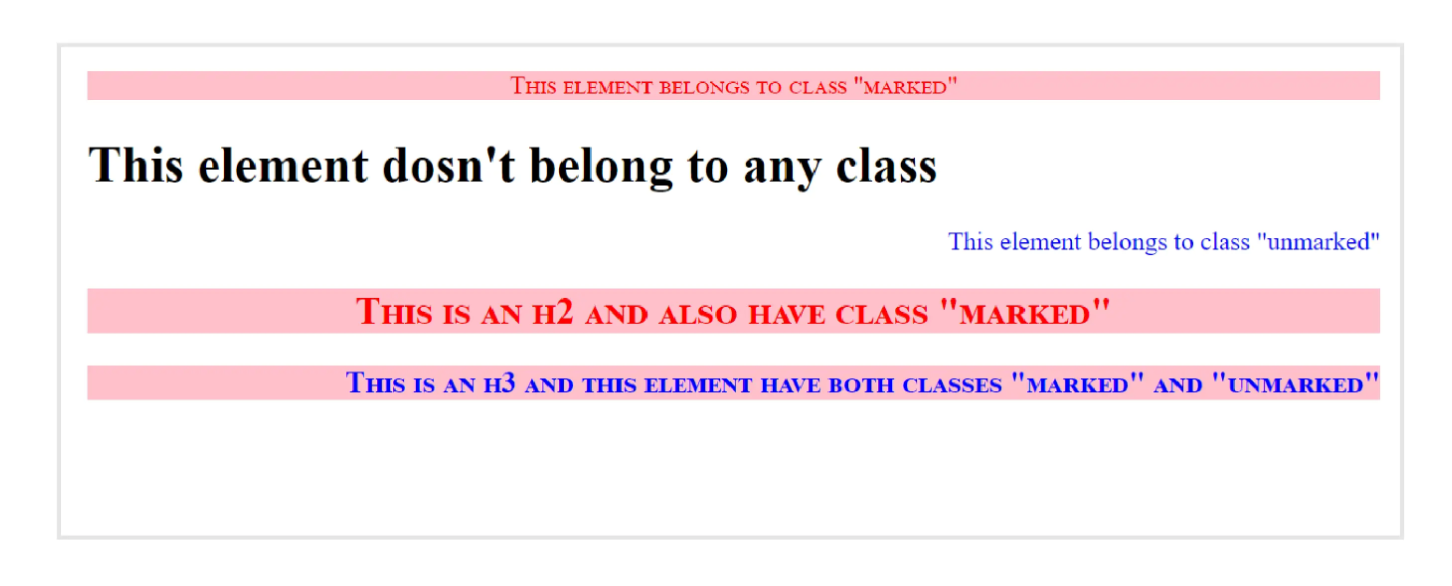
<h2 class="marked"> This is an h2 and also have class "marked"</h2>

<h3 class="marked unmarked"> This is an h3 and this element has both classes "marked" and "unmarked"</h3>

</body>

</html>

**Output:**

****

Now, suppose you have a set of CSS properties selected under a class name, and you only want to apply a specific CSS property to a specific HTML element under the class. For that specify the element name, then give the full stop/period(.) character, followed by the class name.

**Example**: This example is the same as the previous example, the only difference that makes the output look different is the <p> tag, here we only selected the <p> element which will have class "marked" or "unmarked"

<!DOCTYPE html>

<html>

<head>

<style>

p.marked {

background-color: pink;

color:red;

text-align:center;

font-variant:small-caps;

}

p.unmarked{

color:blue;

text-align:right;

}

</style>

</head>

<body>

<div class="marked">

<p class="marked">This element belongs to class "marked"</p>

</div>

<h1>This element doesn't belong to any class</h1>

<p class="unmarked">This element belongs to class "unmarked"</p>

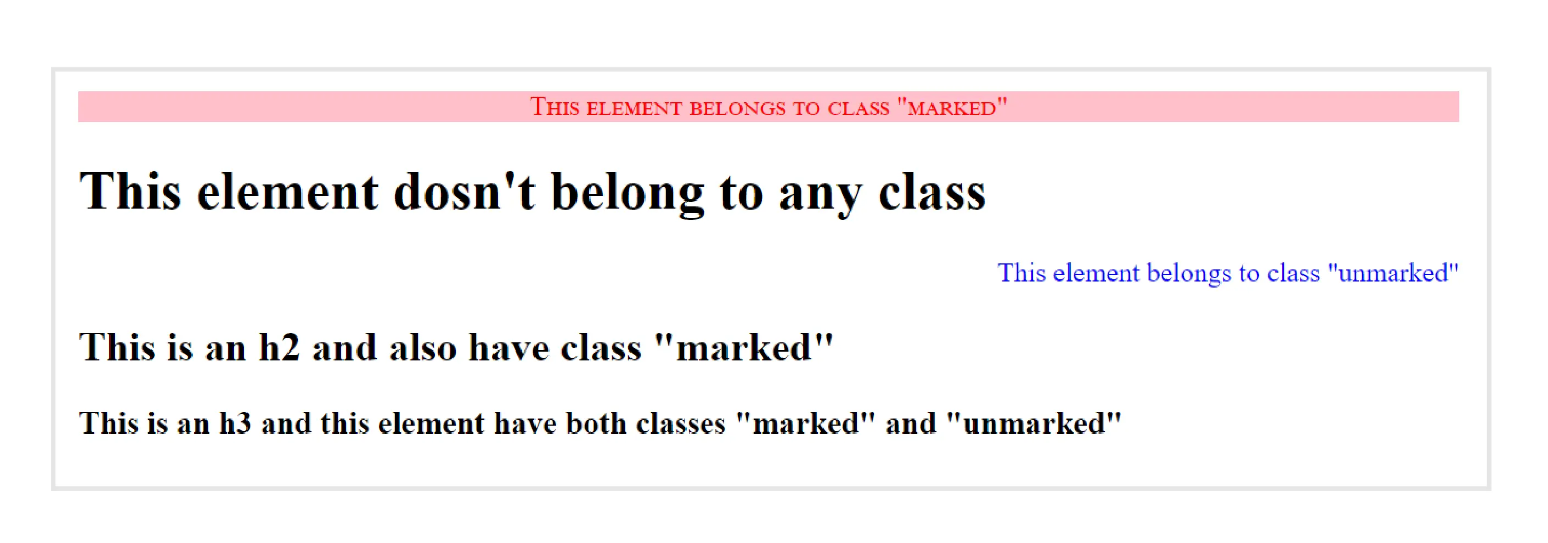
<h2 class="marked"> This is an h2 and also have class "marked"</h2>

<h3 class="marked unmarked"> This is an h3 and this element has both classes "marked" and "unmarked"</h3>

</body>

</html>

**Output:**

****

**ID Selector:** The CSS ID selector uses the HTML element’s ID attribute to select elements on a page and apply CSS properties on that. The id attribute is the unique identifier in an HTML document To use an ID selector in CSS, use a hashtag (#) followed by the name of the ID of the HTML element. The HTML element will only be selected if its id attribute matches exactly with the value given in the selector.

**Syntax:**

#id{

//css properties.

}

ID attribute must contain at least one character and cannot begin with a number.

**CSS Grouping Selector:**

The CSS grouping selector allows you to select multiple HTML elements having the same CSS properties and group them so that the style can be applied at once to all the elements. This reduces the code redundancy and makes code simpler, readable, re-usable.

**Syntax:**

element\_1,element\_2,…{

css properties;

}

**Example**: In this example, we will select <h1>,<h3> and will apply a set of CSS properties

h1, h3{

text-align: center;

color: red;

}

**CSS Combinators:**

CSS selectors are used to selecting HTML elements, but when we use complex selectors(more than one selector being used to select the HTML elements) then the use of combinators explains the relationship between the two selectors. The Combinators combine the selectors in a specific manner to provide them with a useful relationship and the position of content in the document.

There are four types of combinators in CSS:

* Descendant selector (space)
* General Sibling selector (~)
* Adjecant Sibling selector (+)
* Child selector (>)

**1.** **Descendant selector (space)**: The descendant selector in CSS selects all HTML elements that are descendants of a specified ancestor element. The descendant combinator is represented by a single space (" ") character. It combines two selectors in which the first selector represents an ancestor (parent, parent's parent, or ancestor of more than five levels), and the second selector represents descendants. It combines two selectors such that elements matched by the second selector are selected only if they have an ancestor element matching the first selector. The CSS selector that uses a descendant combinator is called descendant selectors.

**Syntax:**

Parent\_element Child\_element {

css properties;

}

**Example**: In this example, we will make <div> as the parnet element and then we will make <p> as the child element, So, if anywhere in the code <p> comes under <div>, then it will be selected.

<!DOCTYPE html>

<html>

<head>

<style>

div p {

background-color: yellow;

color:blue;

font-style: italic;

}

</style>

</head>

<body>

<h2 style="color:red">Descendant Selector</h2>

<p>The descendant selector in CSS selects all HTML elements that are descendants of a specified ancestor element.</p>

<p>This paragraph is not inside div.</p>

<div>

<p>This is paragraph in the div.</p>

<section><p>This is a paragraph inside the section which is in the div.</p></section>

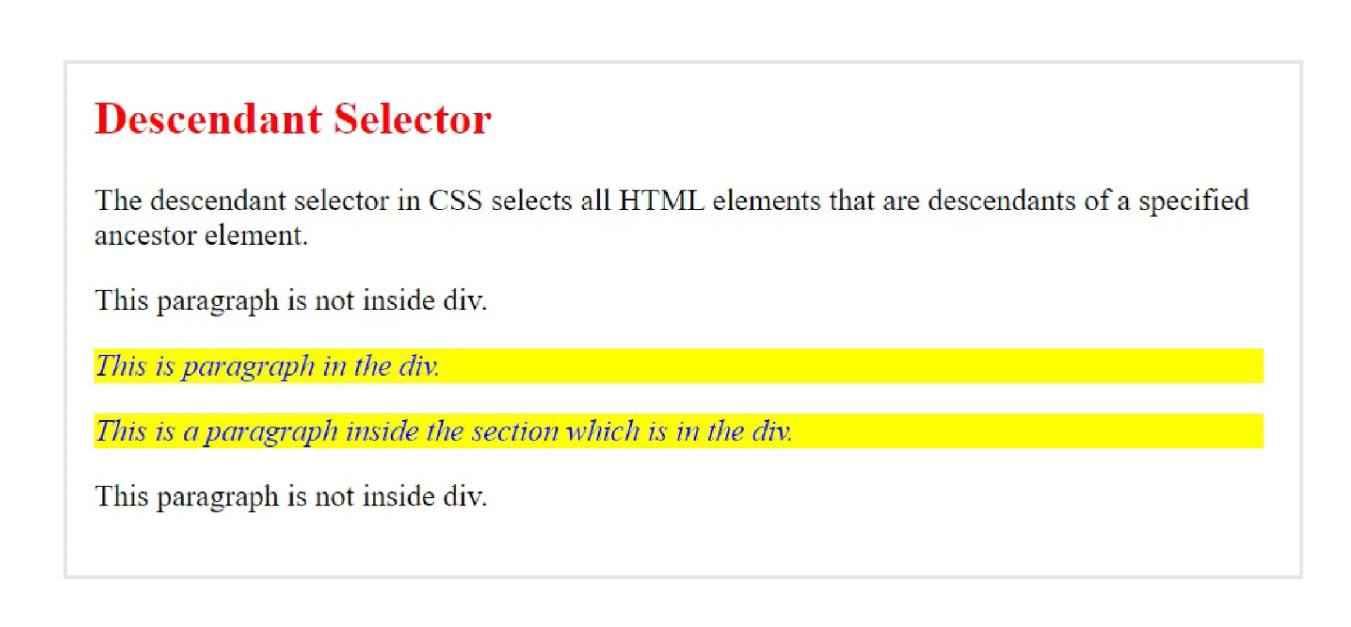
</div>

<p>This paragraph is not inside div.</p>

</body>

</html>

**Output:**

****

**2. General Sibling Selector:**

The general sibling selector selects all HTML elements that are siblings of a specified element i.e. the elements that have to have the common parent will be selected. The second element is specified will be the element that is to be selected, and the first element specified will be the element whose sibling should be the second element, and then only it will be selected. It uses the tilde (~) sign to separate the elements. It can be used for selecting the group of elements that share the common parent element.

**Syntax:**

Element\_1 ~ Element\_2 {

css properties;

}

**Example:** In this example, The elements we will use will be <div> and <p> , To be selected they must be in the same level and the <p> must come after <div>. If in the code these two elements becomes siblings, then the <p> will be selected.

<!DOCTYPE html>

<html>

<head>

<style>

div ~ p {

background-color: yellow;

}

</style>

</head>

<body>

<h2 style="color:red">General Sibling Selector</h2>

<p>The general sibling selector selects all HTML elements that are siblings of a specified element i.e. the elements have having the common parent will be selected.</p>

<p>This is a paragraph.</p>

<div>

<p>This is a paragraph inside div.</p>

</div>

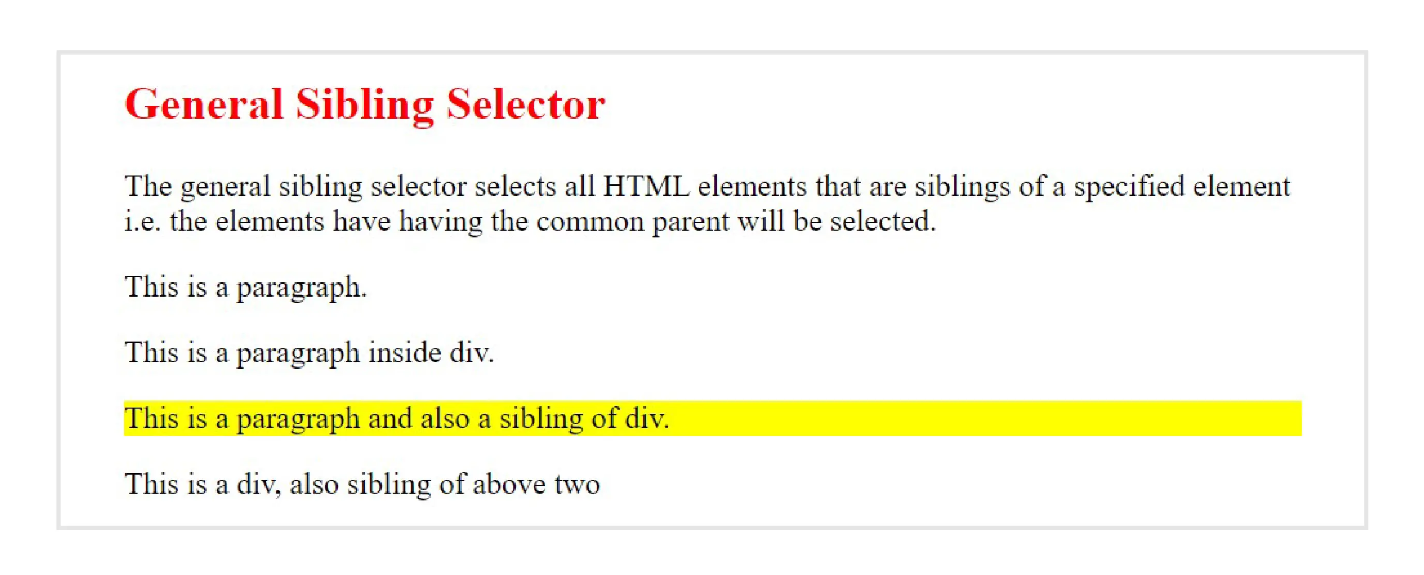
<p>This is a paragraph and also a sibling of div.</p>

<div>This is a div, also sibling of above two</div>

</body>

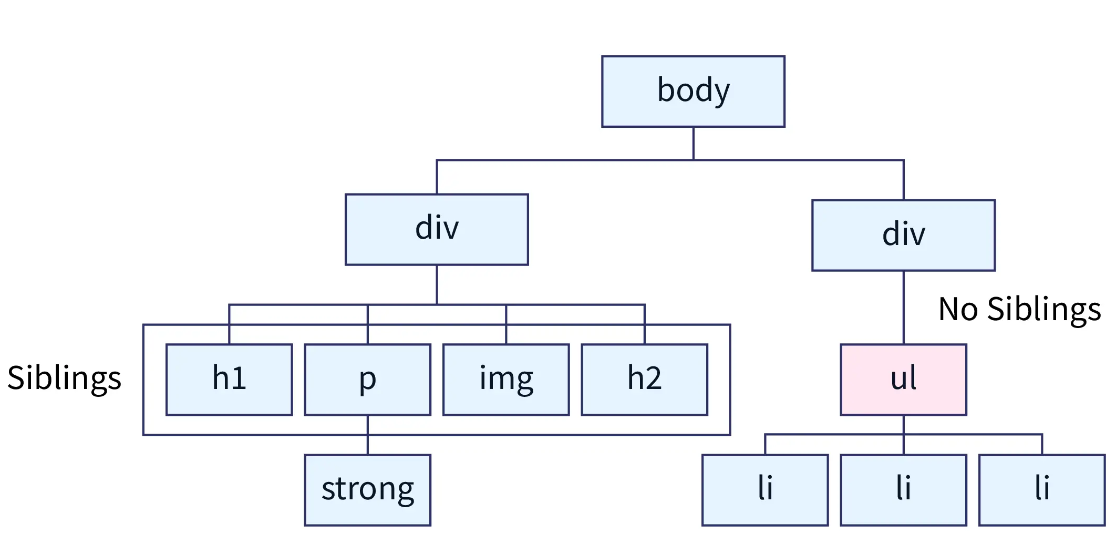
</html>

**Output:**

****

**3. Adjacent Sibling Selector (+):**

The adjacent sibling selector is used to select an element that is adjacent to the first specified element and also the sibling of the first element.



It uses the plus (+) sign to combine the two elements. The second element is selected only when the element specified in the second place is immediately after the first element and the first and the second elements are siblings. This sibling selector selects the adjacent element, or we can say that the element is next to the specified tag.

**Syntax:**

Element\_1 + Element\_2 {

css property;

}

**Example:** In this example we used <h2> and <p> , so <p> will only be selected if it comes immediately after <h2>.

<!DOCTYPE html>

<html>

<head>

<style>

h2 + p {

background-color: yellow;

color:red;

font-size:30px;

}

</style>

</head>

<body>

<h2 style="color:red">Adjacent Sibling Selector</h2>

<p>The adjacent sibling selector is used to select an element that is adjacent to the first specified element and also the sibling of the first element.</p>

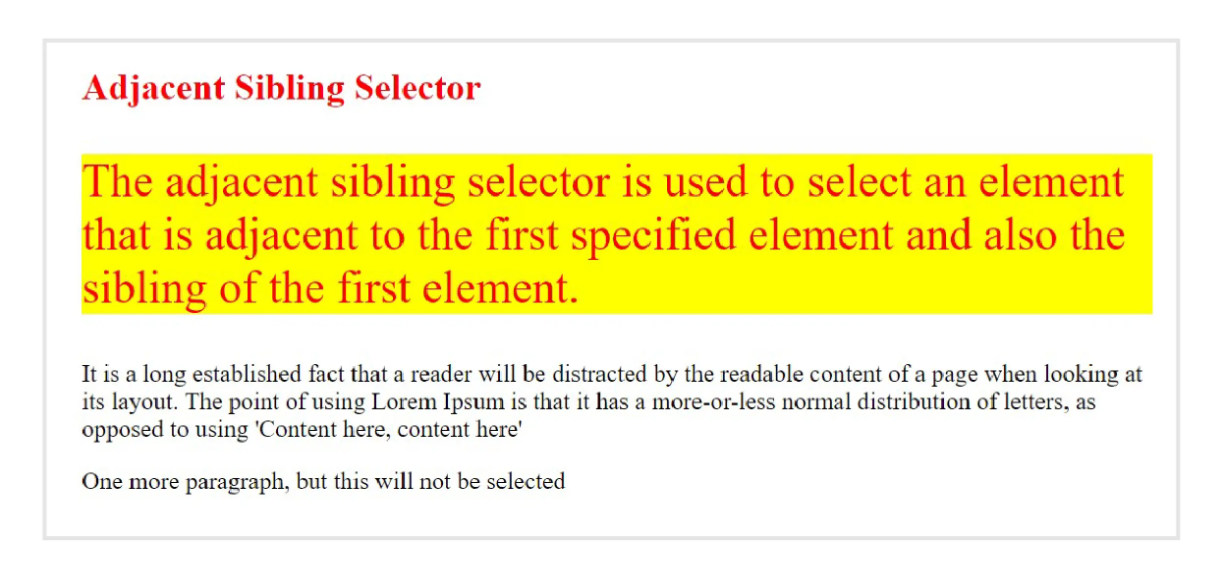
<p>It is a long established fact that a reader will be distracted by the readable content of a page when looking at its layout. The point of using Lorem Ipsum is that it has a more-or-less normal distribution of letters, as opposed to using 'Content here, content here'</p>

<p>One more paragraph, but this will not be selected</p>

</body>

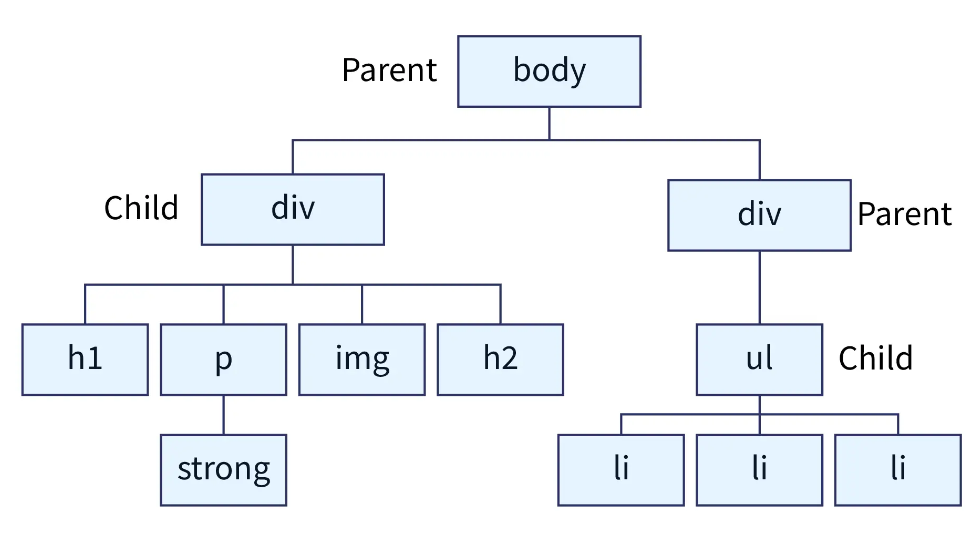
</html>

**Output:**



|  |  |
| --- | --- |
| **Adjacent Sibling Selector** | **General Sibling Selector** |
| The adjacent sibling selector targets the element immediately next to a given HTML element. | The general sibling selector in CSS enables you to select any sibling element that meets the specified criteria. It comes in handy when applying styles to multiple elements that follow a particular HTML element, regardless of their immediate order. |
| A plus sign + is used to denote adjacent sibling selection in CSS | A tilde sign ~ denotes general sibling selection logic in CSS. |

**4. Child selector (>):** The child selector selects all elements that are the immediate children of a specified element.



It uses the greater than (>) sign as the separator between the elements. The parent element must always be placed at the left of the ">". It only selects the elements that are the immediate child of the specified parent element. If we remove the greater than (>) symbol that designates this as a child combinator, then it will become the descendant selector.

**Syntax:**

Parent\_element > child\_element {

css property;

}

**Example**: In the below-discussed example, we had used <h1> as parent element and <h1> as the child element. So, if later in the code <h1> comes immediately after <h1> element, then the

<!DOCTYPE html>

<html>

<head>

<style>

div > p {

background-color: grey;

color: white;

font-size:large;

}

</style>

</head>

<body>

<div>

<p>This paragraph will be selected because it is inside a div</p>

<p>This paragraph will also be selected because it is inside a div</p>

<div>

<p>This paragraph will be selected because it is inside a div (inside another div element).</p>

</div>

<section>

<p>This paragraph will not be selected because it is inside a div but also inside a section element.</p>

</section>

<p>This paragraph will be selected because it is inside a div.</p>

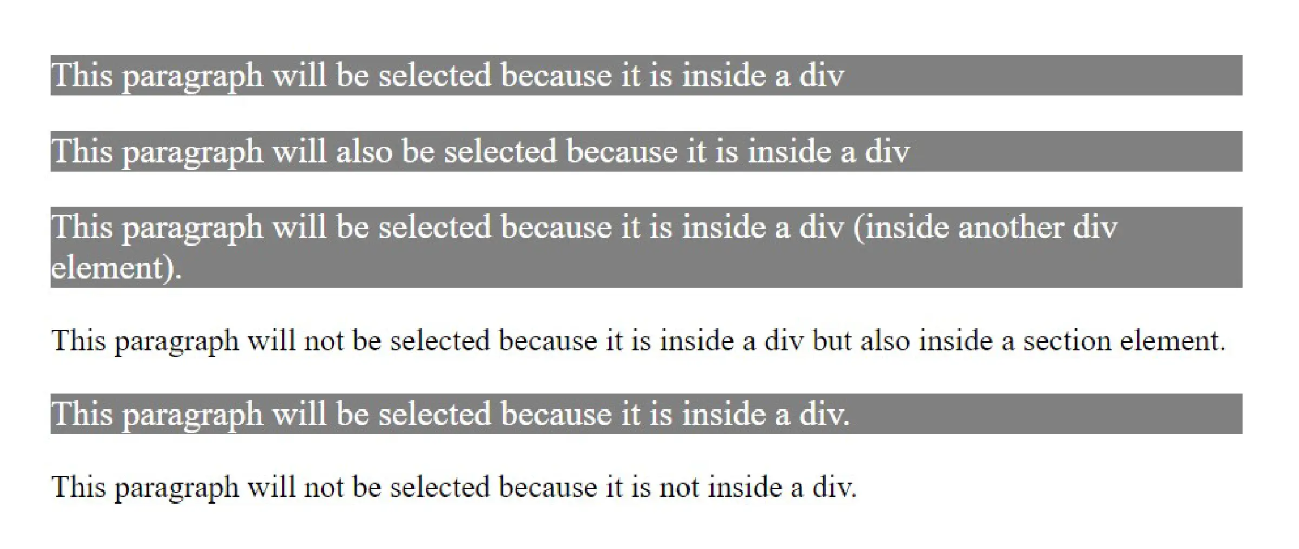
</div>

<p>This paragraph will not be selected because it is not inside a div.</p>

</body>

</html>

**Output:**

****

**CSS Pseudo Classes:**

Ever wondered how the elements in some webpage change their property when you hover your mouse cursor over them?? Or how an element changes its property when you click them?? These are all possible by the use of CSS pseudo-class selectors**.**

**Pseudo-Class is a keyword added to a selector that gives a special state to the selected element(s) when they meet the desired condition**.

**CSS pseudo-classes selector helps to add special effects to the HTML element based on the special state/condition of the element. A pseudo-class can also be understood as an assist that is combined with the simple selectors to change/highlight its property in the special condition**.

**For example,** ": hover" can be used to highlight/ add effect to an element when the mouse cursor hovers over them.

**Syntax** To use pseudo-class, we must use a colon (:) after the CSS element on which we want to apply effects on, and then use specify the pseudo-class.

element : pseudo-class {

css elements;

}

Various CSS pseudo-classes are used in a website. Let's discuss them one by one:

1. **hover:** This pseudo-class adds a special effect to the selected element when you point the mouse cursor over it, this can make the element interactive.

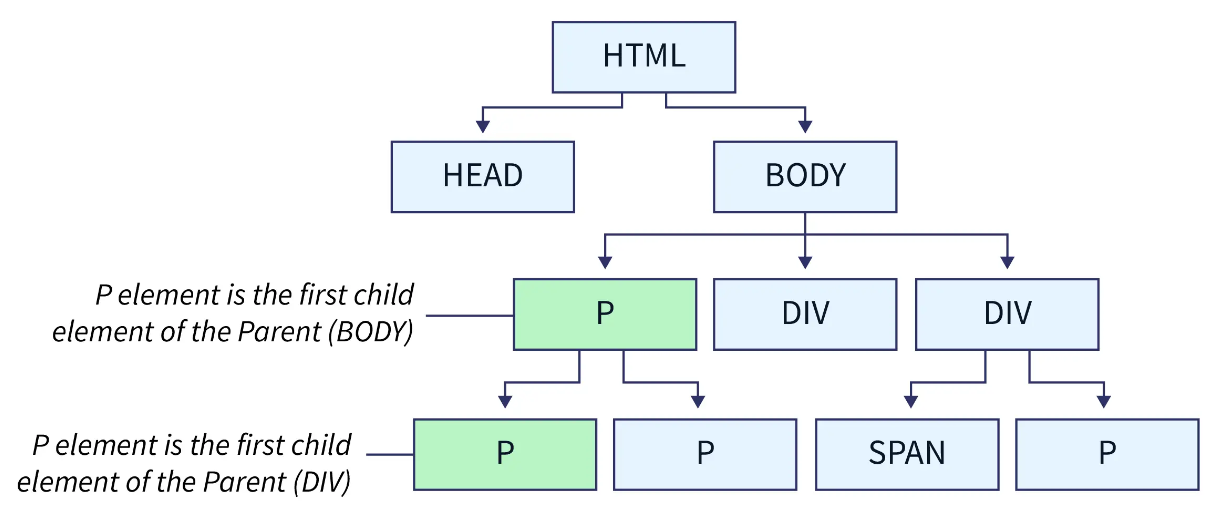
**Syntax:**

element:hover{

css properties;

}

2.  **first-child:** pseudo-class selector is used to target the specified first element(child) immediately inside any parent element.



**Syntax:**

element:first-child{

//css property;

}

**3. visited:** As you have seen in different browsers when you click over a link its color changes This can be achieved by using selectors.

Syntax

element:visited{

//css property;

}

**Different CSS pseudo-class selectors are:**

* **:dir** – It is used to select an element based on its directionality determined by the document language. dir stands for directionality
* **:lang**- It is used to select an element based on the specific value of the lang (language) attribute.
* **:link** – it is used to show the links that are not visited yet on the webpage.
* **:scope** – The CSS style which will be defined in the scope will be applied to every element inside the parent element.
* **:focus** – it is used to select the elements that take input when they are in focus.
* **:active** – it is used to represent an element that is being selected by the input device.
* **:playing** – it is a resource state pseudo-selector and is used to Represent a media element that is capable of playing when that element is playing.
* **:paused**- it is a resource state pseudo-selector and is used to Represent a media element that is capable of playing when that element is paused.
* **:checked** – when the radio button or check-box is enabled they are selected using this selector.
* **:enabled** – helps to show the elements that are enabled.
* **:disabled**- helps to show the elements that are disabled. Similarly, there are plenty of pseudo-class selectors.

**Functional Pseudo-Classes:**

Another variation of the pseudo-class type is the functional pseudo-class. These function calls take in a parameter of a [**selector-list**](https://developer.mozilla.org/en-US/docs/Web/CSS/Selector_list#selector_list)to match elements.

Unlike other types of pseudo-classes that target static state such as hover, these can dynamically target events and user interactions.

* :is()-The matches-any pseudo-class matches any element that matches any of the selectors in the list provided.

**Example:** When writing CSS, you can sometimes end up with long selector lists to target multiple elements with the same style rules. Suppose, if you want to color adjust any <b> tags found inside a heading element, you could write:

**h1 > b, h2 > b, h3 > b, h4 > b, h5 > b, h6 > b {  
 color: hotpink;  
}**

Instead, you could use :is() and improve legibility while avoiding a long selector:

**:is(h1,h2,h3,h4,h5,h6) > b {  
 color: hotpink;  
}**

* :not()-The negation, or matches-none, pseudo-class represents any element that is not represented by its argument.
* :where()- The specificity-adjustment pseudo-class matches any element that matches any of the selectors in the list provided without adding any specificity weight.
* :has()-The relational pseudo-class represents an element if any of the relative selectors match when anchored against the attached element.

All CSS Pseudo Classes:

|  |  |  |
| --- | --- | --- |
| **Selector** | **Example** | **Example description** |
| [:active](https://www.w3schools.com/cssref/sel_active.asp) | a:active | Selects the active link |
| [:checked](https://www.w3schools.com/cssref/sel_checked.asp) | input:checked | Selects every checked <input> element |
| [:disabled](https://www.w3schools.com/cssref/sel_disabled.asp) | input:disabled | Selects every disabled <input> element |
| [:empty](https://www.w3schools.com/cssref/sel_empty.asp) | p:empty | Selects every <p> element that has no children |
| [:enabled](https://www.w3schools.com/cssref/sel_enabled.asp) | input:enabled | Selects every enabled <input> element |
| [:first-child](https://www.w3schools.com/cssref/sel_firstchild.asp) | p:first-child | Selects every <p> elements that is the first child of its parent |
| [:first-of-type](https://www.w3schools.com/cssref/sel_first-of-type.asp) | p:first-of-type | Selects every <p> element that is the first <p> element of its parent |
| [:focus](https://www.w3schools.com/cssref/sel_focus.asp) | input:focus | Selects the <input> element that has focus |
| [:hover](https://www.w3schools.com/cssref/sel_hover.asp) | a:hover | Selects links on mouse over |
| [:in-range](https://www.w3schools.com/cssref/sel_in-range.asp) | input:in-range | Selects <input> elements with a value within a specified range |
| [:invalid](https://www.w3schools.com/cssref/sel_invalid.asp) | input:invalid | Selects all <input> elements with an invalid value |
| [:lang(*language*)](https://www.w3schools.com/cssref/sel_lang.asp) | p:lang(it) | Selects every <p> element with a lang attribute value starting with "it" |
| [:last-child](https://www.w3schools.com/cssref/sel_last-child.asp) | p:last-child | Selects every <p> elements that is the last child of its parent |
| [:last-of-type](https://www.w3schools.com/cssref/sel_last-of-type.asp) | p:last-of-type | Selects every <p> element that is the last <p> element of its parent |
| [:link](https://www.w3schools.com/cssref/sel_link.asp) | a:link | Selects all unvisited links |
| [:not(selector)](https://www.w3schools.com/cssref/sel_not.asp) | :not(p) | Selects every element that is not a <p> element |
| [:nth-child(n)](https://www.w3schools.com/cssref/sel_nth-child.asp) | p:nth-child(2) | Selects every <p> element that is the second child of its parent |
| [:nth-last-child(n)](https://www.w3schools.com/cssref/sel_nth-last-child.asp) | p:nth-last-child(2) | Selects every <p> element that is the second child of its parent, counting from the last child |
| [:nth-last-of-type(n)](https://www.w3schools.com/cssref/sel_nth-last-of-type.asp) | p:nth-last-of-type(2) | Selects every <p> element that is the second <p> element of its parent, counting from the last child |
| [:nth-of-type(n)](https://www.w3schools.com/cssref/sel_nth-of-type.asp) | p:nth-of-type(2) | Selects every <p> element that is the second <p> element of its parent |
| [:only-of-type](https://www.w3schools.com/cssref/sel_only-of-type.asp) | p:only-of-type | Selects every <p> element that is the only <p> element of its parent |
| [:only-child](https://www.w3schools.com/cssref/sel_only-child.asp) | p:only-child | Selects every <p> element that is the only child of its parent |
| [:optional](https://www.w3schools.com/cssref/sel_optional.asp) | input:optional | Selects <input> elements with no "required" attribute |
| [:out-of-range](https://www.w3schools.com/cssref/sel_out-of-range.asp) | input:out-of-range | Selects <input> elements with a value outside a specified range |
| [:read-only](https://www.w3schools.com/cssref/sel_read-only.asp) | input:read-only | Selects <input> elements with a "readonly" attribute specified |
| [:read-write](https://www.w3schools.com/cssref/sel_read-write.asp) | input:read-write | Selects <input> elements with no "readonly" attribute |
| [:required](https://www.w3schools.com/cssref/sel_required.asp) | input:required | Selects <input> elements with a "required" attribute specified |
| [:root](https://www.w3schools.com/cssref/sel_root.asp) | root | Selects the document's root element |
| [:target](https://www.w3schools.com/cssref/sel_target.asp) | #news:target | Selects the current active #news element (clicked on a URL containing that anchor name) |
| [:valid](https://www.w3schools.com/cssref/sel_valid.asp) | input:valid | Selects all <input> elements with a valid value |
| [:visited](https://www.w3schools.com/cssref/sel_visited.asp) | a:visited | Selects all visited links |

**Pseudo-elements Selectors:**

A CSS pseudo-element is combined with the selector and is used to style a specific part of the selected HTML elements. **The pseudo-class is used to style the element, whereas the pseudo-element will style the specific part of the element.** To segregate the pseudo-class from pseudo-elements, we can use a double colon(::) instead of a single colon.

**Syntax:**

selector :: pseudo-element {

//css properties;

}

The different pseudo-elements are:

**::first-line**- It is used to apply styles to the first line of the text. It can be applied only to block-level elements. The different properties that can be used with it are-

* font properties
* color properties
* background properties
* word-spacing
* letter-spacing
* text-decoration
* vertical-align
* text-transform
* line-height
* clear

**Syntax:**

element ::first-line {

//css properties;

}

**::first-letter:** The ::first-letter pseudo-element is used to add a special style to the first letter of a text.

**Note:** The ::first-letter pseudo-element can only be applied to block-level elements.

The following properties apply to the ::first-letter pseudo- element:

* font properties
* color properties
* background properties
* margin properties
* padding properties
* border properties
* text-decoration
* vertical-align (only if "float" is "none")
* text-transform
* line-height
* float
* clear

**::before:** It is used to add content before the contents specific element. It is generally used to add fancy content to an element. It uses the content property.

**Syntax:**

element:: before {

content : “ ” ;

//css properties;

}

**Example**: In the example given below, <h1> is selected as an element and then the contents will be inserted before the content of <h1>.

<!DOCTYPE html>

<html>

<head>

<style>

h1::before {

content: "elements inserted before";

color:red;

}

</style>

</head>

<body>

<h1>This is a heading h1</h1>

<h1>This is a heading h1</h1>

</body>

</html>

**Output:**



**::after**: It works similar to the ::before pseudo-class but it adds the content after the contents of the specified element. It also used content property.

**Syntax:**

element::after {

content : “ ”;

//css properties;

}

**Example:** In the example given below, <h1> is selected as an element and then the contents will be inserted after the content of <h1>.

<!DOCTYPE html>

<html>

<head>

<style>

h1::after {

content: "elements inserted after";

color:red;

}

</style>

</head>

<body>

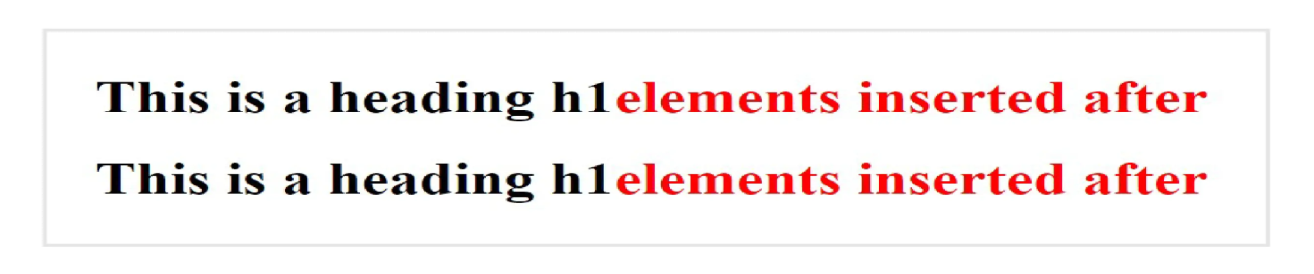
<h1>This is a heading h1</h1>

<h1>This is a heading h1</h1>

</body>

</html>

**Output:**



**::marker**- This is used to select the bullet points or the number of the list items.

**Syntax:**

::marker {

//css properties;

}

**Example:** In this example, we will apply marker pseudo element to the list items of ordered list and also to the unordered list.

<!DOCTYPE html>

<html>

<head>

<style>

::marker {

color: red;

font-size: 23px;

}

</style>

</head>

<body>

<ul>

<li>one</li>

<li>Two</li>

</ul>

<ol>

<li>First</li>

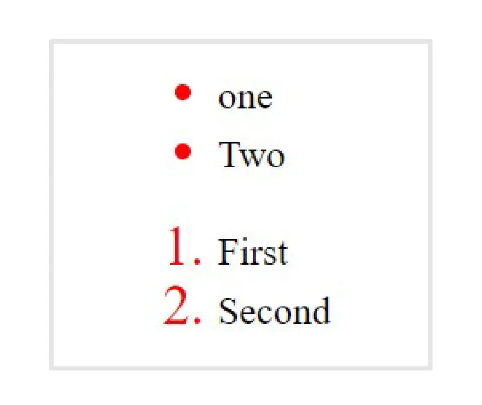
<li>Second</li>

</ol>

</body>

</html>

**Output:**

****

**::selection**: The ::selection pseudo-element selects(highlights) the elements when the user selects the element with the cursor.

**Syntax:**

::selection {

//css poperty;

}

**All CSS Pseudo elements:**

|  |  |  |
| --- | --- | --- |
| **Selector** | **Example** | **Example description** |
| [::after](https://www.w3schools.com/cssref/sel_after.asp) | p::after | Insert content after every <p> element |
| [::before](https://www.w3schools.com/cssref/sel_before.asp) | p::before | Insert content before every <p> element |
| [::first-letter](https://www.w3schools.com/cssref/sel_firstletter.asp) | p::first-letter | Selects the first letter of every <p> element |
| [::first-line](https://www.w3schools.com/cssref/sel_firstline.asp) | p::first-line | Selects the first line of every <p> element |
| [::marker](https://www.w3schools.com/cssref/sel_marker.asp) | ::marker | Selects the markers of list items |
| [::selection](https://www.w3schools.com/cssref/sel_selection.asp) | p::selection | Selects the portion of an element that is selected by a user |

**Attribute Selectors**:

 HTML elements can have one or more attributes, and any one of these attributes can be used to target and select HTML elements, they are selected if they match the condition being mentioned while selecting the attribute, and after selecting the attribute different CSS properties can be applied. The CSS Attribute Selector selects elements based on the existence of the given attribute or the value of the given attribute.

There are various methods of selecting an attribute:

**CSS [Attribute] selector** : In this type of attribute selector, the name of the element is used and then the name of the attribute is specified in the square brackets. The elements which match the name of the attribute specified in the square bracket are selected and the CSS properties are applied to them.

**Syntax:**

element [Attribute] {

css properties

}

**CSS [attribute = “value”] Selector:**

In this type of attribute selector, all the HTML elements whose attribute and attribute’s value (Attribute value is the string of the of characters that are assigned to the attribute of the HTML element. It mentioned inside the double quotes “ ”) matches exactly with the values specified in the selector are selected, and the CSS properties specified are applied.

**Syntax:**

element[attribute = “value”] {

css property;

}

**CSS [attr~=value] Selector:**

Using this type of attribute selection also, we select HTML elements based on attribute and their values, In this case, the element will be selected even if the value mentioned in the selector matches any value (as they can be a whitespace-separated list of words) in the element's attribute.

**Syntax:**

element [attribute ~= “value name”] {

css properties;

}

**CSS [attr|=value] Value:**

This attribute selector is used to select all the HTML elements whose attribute has a value of a hyphen-separated list of words, the element will be selected if the start of a hyphen-separated list of words in the attribute values matches the value specified in the selector. The value of the attribute can be a whole exact word either alone or should begin with the value immediately followed by a hyphen. (-)

**Syntax:**

[attribute|=value]{

Css properties;}

**[attribute^="value"] Selector:**

As in the previous type of attribute selector, where the value of the attribute is a hyphen-separated list of words, and then we apply [attr|=val] selector to select them. But In the [attr^=val] type of attribute selector, the value need not be separated by hyphen or space, the value of the attribute just need to begin with the value specified in the selector. This selector comes in the category of Substring matching selectors.

**Syntax:**

[attribute ^= “value”] {

Css properties;

}

**CSS [attribute$="value"] Selector:**

In this type of attribute selector, we will select HTML elements whose attribute’s value ends with the specified value. The value need not be the whole word.

**Syntax:**

[attribute $= “value”]{

Css properties;

}

**[attribute \*= "value"] selector:**

we can select elements whose attribute value matches the value specified in the selector and the value can be present anywhere (if there is a list of words in the value).

**Syntax:**

[attribute \*= “value”] {

Css property;

}

**What is CSS Specificity?**

CSS Specificity is the set of rules applied to the CSS selectors to determine which styles are applied to an element. Every selector has a specificity value. The more specific a CSS selector is, the larger it's the specificity value hence, it will be applied to an element.

CSS stands for Cascading Stylesheets, and in this context, cascading refers to the sequence in which CSS rules are applied to an element in order to style it. If two rules are applied to the same element, then, CSS specificity comes into picture to identify which rule should be applied to the element according to it's specificity value.

**Note:** When the specificity value of a selector between multiple rules is the same, the rule that is defined at the last is applied to that element.

**Specificity Rules include:**

* CSS style applied by referencing external stylesheet has lowest precedence and is overridden by Internal and inline CSS.
* Internal CSS is overridden by inline CSS.
* Inline CSS has highest priority and overrides all other selectors.

**Specificity Hierarchy :** Every element selector has a position in the Hierarchy.

1. **Inline style:** Inline style has highest priority.
2. **Identifiers(ID):** ID have the second highest priority.
3. **Classes, pseudo-classes and attributes:** Classes, pseudo-classes and attributes are come next.
4. **Elements and pseudo-elements:** Elements and pseudo-elements have lowest priority.

**Note:**

* When two or more selectors have equal specificity, the last(latest) one counts.
* Universal selectors like body and inherited selectors have least specificity.

# How are the points in CSS specificity calculated ?

The concept of CSS Specificity comes when more than one CSS rules points to a same element. At this point the selector with highest specificity is considered as highest priority and apply it to the HTML text.

**Calculating Specificity:**For each CSS selector there are different points that are considered while checking the priority. Given below is the table consisting of various CSS selectors and the points associated with them.

|  |  |
| --- | --- |
| **CSS Selector** | **Points** |
| Inline CSS | 1000 |
| ID selector | 100 |
| Class selector | 10 |
| Element Selector | 1 |

Let’s look into few examples on Specificity calculation.

|  |  |
| --- | --- |
| **Selector** | **Specificity Value** |
| b | 1 |
| b .classdemo | 1+10=11 |
| b #idDemo | 1+100=101 |
| Inline <b style=”color: green;”> | 1000 |

**CSS Inheritance:**

Inheritance is a process of receiving values of properties by a child element from its parent element.

**Example:**

<!DOCTYPE html>

<html>

<head>

    <style>

        #parentclass {

            color: black;

        }

        #child1 {

            color: green;

        }

        #childchild1 {

            color: red;

        }

    </style>

</head>

<body>

    <div id="parentClass">

        Parent

        <div id="child1">

            Child 1

            <div id="childchild1">

                Child Child 1

                <div id="childchildchild1">

                    Child Child Child

                </div>

            </div>

            <div id="childchild2">

                Child Child 2

            </div>

        </div>

        <div id="child2">

            Child 2

        </div>

    </div>

</body>

</html>

**Output:**



Here #parentclass has color:black, #child1 has color:green and #childchild1 has color:red. In the above code #child1 and #child2 are in #parentclass so both should get the color black inherited but only child 2 gets the color because we gave #child1 to color: green this is known as specificity.

We cannot inherit all the properties /rules of CSS. All font-\* properties are naturally inherited like  
• font-size  
• font-family  
• font-weight  
• font-style, etc.

The color property is also inherited.  
**CSS properties such as height, border, padding, margin, width, etc. are not inherited naturally. We can do inheritance on noninheritable CSS properties. We use inherit for doing so.**

**Example:**

<!DOCTYPE html>

<html>

<head>

    <style>

        #parentclass {

            padding: 30px;

            color: red;

        }

        #Child {

            padding: inherit;

        }

    </style>

</head>

<body>

    <div id="parentclass">

        Parent

        <div id="Child">Child</div>

    </div>

</body>

</html>

**Output:**



**Note:** Only the direct child element of a parent element can inherit it but the grandchild cannot.