CSS selectors and combinators

CSS selectors are used to define a pattern of the elements that you want to select for applying a set of CSS rules on the selected elements. Combinators define the relationship between the selectors. Using various selectors and combinators, you can precisely select and style the desired elements based on their type, attributes, state, or relationship to other elements.

Types of selectors

There are over 80 selectors and combinators. CSS selectors can be grouped into the following categories based on the type of elements they can select

1- **Simple selectors and combinators**

   1- **Simple selectors**

   The type selector selects all elements that have the given node name. For example, div will select all <div> elements and input will match any <input> element. The universal selector, denoted with an asterisk (*), is a special type selector that selects all elements.

   2- **Combinators**

   A CSS selector can contain more than one simple selector. Between the simple selectors, we can include a combinator.

   There are four different combinators in CSS:

   - **descendant selector (space)**
     The descendant selector matches all elements that are descendants of a specified element.
     Example:
     ```
     div p {
       background-color: yellow;
     }
     ```

   - **child selector (>)**
     The child selector selects all elements that are the children of a specified element.
     Example:
     ```
     div > p {
       background-color: yellow;
     }
     ```

- **adjacent sibling selector (+)**
  The adjacent sibling selector is used to select an element that is directly after another specific element.
  Sibling elements must have the same parent element, and "adjacent" means "immediately following".
  Example:
  ```
  div + p {
    background-color: yellow;
  }
  ```

- **general sibling selector (~)**
  The general sibling selector selects all elements that are next siblings of a specified element.
  Example:
  ```
  div ~ p {
    background-color: yellow;
  }
  ```

2- Class and ID selectors

The **class selector** selects all elements that have the given class attribute denoted by the class name prefixed with a period (.). For example, .index will match any element that has class="index". The **ID selector** selects an element based on the value of its id attribute. The selector is the id prefixed with a "number sign" (U+0023, #). For example, #toc will match the element that has id="toc". Both **class** and **id** are global attributes. There should be only one element with a given id in a document; but if there is more than one, the ID selector will match all the elements with that id.

When combining a type or universal selector with a class or id selector to create a compound selector, the type or universal selector must precede the class or id.

Example:

HTML :-

<p class="myClass" id="myId">I match everything.</p>

<p>I match the universal and type selectors only.</p>

CSS :-

```
* {
  font-style: italic;
}
p {
  color: red;
}
.myClass {
  text-decoration: underline;
}
#myId {
  font-family: monospace;
}
p.myClass#myId {
  font-size: 1.5rem;
}
```

3- Attribute selectors

Attribute selector select all elements that, depending on how the selector is written, either have the given attribute or have the given attribute with a substring value match. For example, [type] will match all elements that have the type attribute set (to any value), and [type="submit"] will match <input type="submit"> and <button type="submit">, or any element with type="submit" set, even though this attribute-value pair is only supported on **<input>** and **<button>** elements. The match is case-insensitive.

The case sensitivity of the attribute depends on the language. Generally, in HTML, if an attribute is **enumerated**, the value in the selector is case-insensitive, even if the value is not one of the enumerated values or if the attribute is not a valid value for the element on which it is set. For non-enumerated attributes, like class, id, or any data-* attribute, or for non-HTML attributes, like role or aria-* attributes, the value match is case-sensitive; the match can be made case-insensitive with a case-insensitive modifier (i).

4- Pseudo-class selectors

A pseudo-class is used to define a special state of an element.

For example, it can be used to:

- Style an element when a user mouses over it
- Style visited and unvisited links differently
- Style an element when it gets focus

The syntax of pseudo-classes:

```
selector:pseudo-class {
  property: value;
}
```

All CSS Pseudo Classes

Selector	Example	Example description
:active	a:active	Selects the active link
:checked	input:checked	Selects every checked <input> element
:disabled	input:disabled	Selects every disabled <input> element
:empty	p:empty	Selects every <p> element that has no children
:enabled	input:enabled	Selects every enabled <input> element
:first-child	p:first-child	Selects every <p> elements that is the first child of its parent
:first-of-type	p:first-of-type	Selects every <p> element that is the first <p> element of its parent
:focus	input:focus	Selects the <input> element that has focus
:hover	a:hover	Selects links on mouse over
:in-range	input:in-range	Selects <input> elements with a value within a specified range
:invalid	input:invalid	Selects all <input> elements with an invalid value
:lang(language)	p:lang(it)	Selects every <p> element with a lang attribute value starting with"it"
:last-child	p:last-child	Selects every <p> elements that is the last child of its parent
:last-of-type	p:last-of-type	Selects every <p> element that is the last <p> element of its parent
:link	a:link	Selects all unvisited links
:not(selector)	:not(p)	Selects every element that is not a <p> element
:nth-child(n)	p:nth-child(2)	Selects every <p> element that is the second child of its parent
:nth-lastchild(n)	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)	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)	p:nth-of-type(2)	Selects every <p> element that is the second <p> element of its parent
:only-of-type	p:only-of-type	Selects every <p> element that is the only <p> element of its parent
:only-child	p:only-child	Selects every <p> element that is the only child of its parent
:optional	input:optional	Selects <input> elements with no "required" attribute
:out-of-range	input:out-of-range	Selects <input> elements with a value outside a specified range
:read-only	input:read-only	Selects <input> elements with a "readonly" attribute specified
:read-write	input:read-write	Selects <input> elements with no "readonly" attribute
:required	input:required	Selects <input> elements with a "required" attribute specified
:root	root	Selects the document's root element
:target	#news:target	Selects the current active #news element (clicked on a URL containing that anchor name)
:valid	input:valid	Selects all <input> elements with a valid value
:visited	a:visited	Selects all visited links

5- **Pseudo-element selectors**

Not all CSS selectors are defined in the CSS selectors module. CSS pseudo-element selectors are defined in the CSS pseudo-elements module.

CSS **pseudo-elements**, prefixed with two colons (::), represent entities that are not included in HTML. For example, the simple ::marker selector selects list item bullets, and the compound selector p::first-line matches the first line of all <p> elements.

All CSS Pseudo element

Selector	**Example**	**Example description**
::after	p::after	Insert content after every <p> element
::before	p::before	Insert content before every <p> element
::first-letter	p::first-letter	Selects the first letter of every <p> element
::first-line	p::first-line	Selects the first line of every <p> element
::marker	::marker	Selects the markers of list items
::selection	p::selection	Selects the portion of an element that is selected by a user