In CSS, selectors are used to target and apply styles to specific HTML elements. There are several types of selectors that can be used, including:

1. Type Selectors:[**tag names**] These are the most basic selectors and target elements based on their tag names. For example, to target all the h1 elements on a page, you would use the h1 selector.
2. Class Selectors:[**class name (.)**] These selectors target elements based on the value of their class attribute. To use a class selector, you start with a period (.) followed by the name of the class. For example, to target all elements with the class "example", you would use the .example selector.
3. ID Selectors: [**id (#)**]These selectors target elements based on the value of their id attribute. To use an ID selector, you start with a hash symbol (#) followed by the name of the ID. For example, to target the element with the ID "header", you would use the #header selector.
4. Attribute Selectors: **tagName[attribute=’value’]** These selectors target elements based on the value of one of their attributes. To use an attribute selector, you enclose the attribute name in brackets ([]) and specify the value you're targeting. For example, to target all elements with a data-toggle attribute, you would use the [data-toggle] selector.
5. Descendant Selectors: These selectors target elements that are descendants of another element. To use a descendant selector, you specify the parent element followed by a space, then the descendant element. For example, to target all span elements that are inside a div element, you would use the div span selector.
6. Child Selectors: These selectors target elements that are direct children of another element. To use a child selector, you specify the parent element followed by a greater than sign (>), then the child element. For example, to target all p elements that are direct children of a div element, you would use the div > p selector.
7. Adjacent sibling selectors: These are used to select the next sibling element that immediately follows another element. The selector "h1 + p" selects the first paragraph that follows an h1 element.
8. General sibling selectors: These are used to select all sibling elements that follow another element. The selector "h1 ~ p" selects all paragraphs that follow an h1 element.
9. Pseudo-class Selectors: These selectors target elements based on their state or position in the document. To use a pseudo-class selector, you start with a colon (:) followed by the name of the pseudo-class. For example, to target all links that have been visited, you would use the :visited selector.
   1. :nth-child()
   2. :nth-last-child()
   3. :nth-last-of-type()
   4. :nth-of-type()
   5. :only-child
   6. :only-of-type
   7. :first-of-type
   8. :last-of-type
   9. :nth-of-type
10. Pseudo-element Selectors: These selectors target specific parts of an element, such as its first letter or its content. To use a pseudo-element selector, you start with two colons (::) followed by the name of the pseudo-element. For example, to target the first letter of a p element, you would use the ::first-letter selector.

We cannot identify elements using text in the CSS selector. To do that, you'll have to use an XPath.

| Selector | Example | Description |
| --- | --- | --- |
| # | #id1 | Used for selecting the element with id |
| . | .class1 | Used for selecting the element with class |
| [ ] | img[alt=’abc’] | Used for writing the attribute-based CSS selectors |
| > | div>button | Used for selecting the immediate child |
| + | div + button | Used for selecting the sibling element that is placed immediately after the first one (div) |
| ~ | div~nav | Used for selecting the general siblings |
| [attribute\*=value] | a[href\*=”tests”] | Selects elements which are containing the attribute value |
| [attribute$=value] | [href$=’.docx] | Selects an element that’s element value ends with the attribute value |
| [attribute=value] | [alt=’abc’] | Selects an element with the exact match of the attribute value |
| :nth-child(n) | div:nth-child(2) | Selects the nth child Just like the index in XPath |

| **Path Selector** | **CSS Selector** |
| --- | --- |
| Xpath is **bidirectional** you can traverse elements from parent to child or child to parent | CSS selector is **unidirectional** you can only traverse from parent to child |
| XPath is **slower** in terms of speed and performance | CSS selector is **comparatively faster** |
| XPath allows the construction of **text-based selectors** | CSS doesn’t allow to construct the text-based selectors |
| XPath should start with / or // followed by a tag name or wildcards like \* | CSS allows to directly direct some of the attribute-based selectors such as # for id and . for classes |
| XPath provides Axes to solve complicated selector problems | CSS doesn’t have any Axes methods |
| Xpath is less readable as it grows | CSS Selectors are more readable |