Advanced Selectors

Mohamed Emary

October 24, 2023

1 CSS Selectors

1. `*`: This selector targets all elements on the page. It selects everything.

2. `Element`: Selects all instances of a specific HTML element, such as `p`, `div`, or `h2`.

3. `Element OtherElement`: This selects `OtherElement` that is a descendant of `Element`. For example, `div p` selects all `p` elements that are inside a `div`.

4. `.class-name`: Selects all elements with a specific class. For example, `.class-name` targets all elements with the class "class-name".

5. `#id-name`: Selects a specific element with the given ID. For instance, `#id-name` selects the element with the ID "id-name".

6. `.parent .child`: Selects all elements with class "child" that are descendants of an element with class "parent".

7. `.class-one.class-two`: Targets elements with both class "class-one" and class "class-two".

8. `.class-name div, .class-name p`: Selects `div` and `p` elements that have the class "class-name".

9. `Element.class-name`: Targets elements of a specific type with a particular class. For example, `p.class-name` selects all `p` elements with the class "class-name".

10. `.parent > .child`: Selects all elements with class "child" that are a direct child of an element with class "parent".

11. `Element + Other Element => [div + p]`: This selector targets a `p` element that is immediately preceded by a `div` element.

12. `Element ~ Other Elements => [p ~ div]`: This selector targets all `div` elements that are siblings of a `p` element, meaning they share the same parent and are preceded by the `p` element.

13. `[Attribute]`: Selects all elements that have the specified attribute, regardless of its value. For example, `[data-attribute]` selects all elements with a "data-attribute" attribute.

14. `Element[Attribute]`: Targets elements of a specific type that have the specified attribute. For instance, `a[href]` selects all `a` (anchor) elements with an "href" attribute.

15. `[Attribute=Value]`: Selects elements with a specific attribute value. For example, `[class="example"]` targets all elements with a class attribute equal to "example".

16. `Element[Attribute=Value] => input[type="submit"]`: Targets `input` elements with the attribute "type" set to "submit".

17. `[Attribute~=Value] => Contains A Word`: Selects elements where the attribute contains a specific word. For instance, `[class~=important]` targets elements with the class attribute containing the word "important."

18. `[Attribute*=Value] => Contains A String`: Selects elements where the attribute contains a specific substring. For example, `[href*="example"]` targets elements with an "href" attribute containing the substring "example."

19. `[Attribute^=Value] => Start With A String`: Selects elements where the attribute value starts with a specific string. For instance, `[class^="prefix"]` targets elements

with a class attribute starting with "prefix."

20. `[Attribute$=Value] => End With A String`: Selects elements where the attribute value ends with a specific string. For example, `[class$="suffix"]` targets elements with a class attribute ending with "suffix."

21. `:first-child`: Selects an element that is the first child of its parent. For example, `p:first-child` selects the first `p` element among its siblings.

22. `:last-child`: Targets an element that is the last child of its parent. For instance, `li:last-child` selects the last `li` element among its siblings.

23. `:first-of-type`: Selects the first element of a specific type among its siblings. For example, `h2:first-of-type` selects the first `h2` element among its siblings.

24. `:last-of-type`: Targets the last element of a specific type among its siblings. For instance, `span:last-of-type` selects the last `span` element among its siblings.

25. `:only-child`: Selects an element that is the only child of its parent, meaning it has no siblings. For example, `div:only-child` selects a `div` element that is the only child of its parent.

26. `:not(Selectors)`: Selects elements that do not match the specified selectors. For example, `p:not(.special)` selects all `p` elements that do not have the class "special."

27. `:nth-child(n)`: Selects the nth child of its parent. The value of `n` can be a number, a keyword, or a formula. For instance, `li:nth-child(2)` selects the second `li` element among its siblings.

28. `:nth-last-child(n)`: Similar to `:nth-child(n)`, but counts the children from the end. For example, `div:nth-last-child(3)` selects the third-to-last `div` element among its siblings.

29. `:nth-of-type(n)`: Selects the nth child of its parent with a specific type. For example, `p:nth-of-type(odd)` selects all odd-numbered `p` elements among its siblings.

30. `:nth-last-of-type(n)`: Similar to `:nth-of-type(n)`, but counts the elements from the end. For instance, `span:nth-last-of-type(4)` selects the fourth-to-last `span` element among its siblings.

31. `:root`: Selects the highest-level parent element in the document, typically the `<html>` element. It is often used to define global styles or CSS variables.

32. `:checked`: Selects input elements that are checked, such as checkboxes or radio buttons. It is often used in conjunction with these input types for styling or interaction.

33. `:empty`: Selects elements that have no children, including text nodes and empty spaces. For example, `p:empty` selects all empty `<p>` elements.

34. `:disabled`: Targets form elements that are disabled. This can include input fields, buttons, or other form elements with the `disabled` attribute.

35. `:required`: Selects form elements that are marked as required using the HTML `required` attribute. It's often used to style or apply specific behavior to required form fields.

36. `:focus`: Selects an element that is currently in focus. This is often used to style elements when they are selected by the user.

37. `::selection`: Targets the portion of a document that is currently selected by the user. It is often used to style the background or text color of the selected text.

38. `::placeholder`: Targets the placeholder text in an input field. It is often used to style the appearance of the placeholder text.