
KodeGo

KodeGo

Overview

CSS Advance Selector

SIMPLE CSS SELECTORS

1. Universal selector
2. Type selector
3. Class selector
4. ID selector

UNIVERSAL SELECTORS

- selects everything - every single element in the document.
- also known as a wildcard
- To use the universal selector, use the asterisk character (*).

```
* {
    property: value;
}
```

TYPE SELECTORS

- aka TAG BASED SELECTORS
- selects all HTML elements of the specified type.
- To use it, mention the name of the HTML element.
- For example, if you wanted to apply a style to every single paragraph in the HTML document, you would specify the p element:

```
p {
    property: value;
}
```

CLASS SELECTORS

- matches and selects HTML elements based on the value of their given class.
- selects every single element in the document with that specific class name.
- With the class selector, you can select multiple elements at once and style them the same way without copying and pasting the same styles for each one separately.
- To select elements with the class selector, use the dot character (.), followed by the name of the class.

```
.my_class {
    property: value;
}
```

ID SELECTOR

- selects an HTML element based on the value of its ID attribute.
- ID of an element should be unique in a document, meaning there should only be one HTML element with that given ID value.
- To select an element with a specific ID, use the hash character (#), followed by the name of the ID value:

```
#my_id {
    property: value;
}
```

ATTRIBUTE SELECTORS

1. The [attribute] selector
2. The [attribute="value"] selector
3. The [attribute^="value"] selector
4. The [attribute$="value"] selector
5. The [attribute*="value"] selector
6. The [attribute~="value"] selector

[attribute] SELECTOR

- To use the attribute selector, use a pair of square brackets [ ], to select the attribute you want.

General Syntax:

```
element[attribute]
```

- It selects an element if the given attribute exists.
- In the following example, elements that have the attribute attr present are selected, regardless of the specific value of attr:

```
a[attr] {
    property: value;
}
```

[attribute="value"] SELECTOR

Syntax:

```
element[attribute="value"]
```

- If you want to style a elements with an attr attribute that has an exact value of 1, you would do the following:

```
a[attr="1"] {
    property: value;
}
```

[attribute^="value"] SELECTOR

Syntax:

```
element[attribute^="value"]
```

- For example, if you wanted to select and style any a elements that have an attr attribute with a value that starts with www, you would do the following:

```
a[attr^="www"] {
    property: value;
}
```

[attribute$="value"] SELECTOR

Syntax:

```
element[attribute$="value"]
```

- For example, if you wanted to select a elements that have an attr attribute name with a value that ends with .com, you would do the following:

```
a[attr$=".com"] {
    property: value;
}
```

[attribute*="value"] SELECTOR

Syntax:

```
element[attribute*="value"]
```

- In this case, the string value needs to be present in the attribute's value followed by any number of other characters - value doesn't need to be a whole word.
- For example, if you wanted to select a elements that have an attr attribute with a value that contains the string free, you would do the following:

```
a[attr*="free"] {
    property:value;
}
```

[attribute~="value"] SELECTOR

Syntax:

```
element[attribute~= "value"]
```

- In this case, the string value needs to be a whole word.
- For example, if you wanted to select a elements that have an attr attribute name with a value that contains the word free, you would do the following:

```
a[attr~= "free"] {
    property: value;
}
```

GROUPING CSS SELECTOR

- With the grouping selector, you can target and style more than one element at once.
- To use the grouping selector, use a comma (,), to group and separate the different elements you want to select.
- For example, here is how you would target multiple elements such as divs, ps, and spans all at once and apply the same styles to each of them:

```
div, p, span {
    property: value;
}
```

CSS COMBINATORS

1. Descendant combinator
2. Direct child combinator
3. General sibling combinator
4. Adjacent sibling combinator

- Combinators allow you to combine two elements based on the relationship between the elements and their location in the document.

DESCENDANT COMBINATOR

- selects only the descendants of the specified element.
- Essentially, you first mention the parent element, leave a space, and then mention the descendant of the first element, which is the child element of the parent.
- The child element is an element inside the parent element.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="main.css">
  <title>Document</title>
</head>
<body>
  <div>
    <h2>I am level 2 heading</h2>
    <p>I am a paragraph inside a div</p>
    <span>I am a span</span>
    <p>I am a paragraph inside a div</p>
  </div>
  <p>I am a paragraph outside a div</p>
</body>
</html>
```

```
div p {
  color: red;
}
```

KodeGo

DIRECT CHILD COMBINATOR

- aka direct descendant
- selects only the direct children of the parent.
- To use the direct child combinator, specify the parent element, then add the > character followed by the direct children of the parent element you want to select.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="main.css">
  <title>Document</title>
</head>
<body>
  <div>
    <a href="#">I am a link</a>
    <a href="#">I am a link</a>
    <p><a href="#">I am a link inside a paragraph</a></p>
  </div>
</body>
</html>
```

```
div > a {
  color: red;
}
```

GENERAL SIBLING COMBINATOR

- selects siblings
- You can specify the first element and a second one that comes after the first one.
- The second element doesn't need to come right after the first one.
- To use the general sibling combinator, specify the first element, then use the ~ character followed by the second element that needs to follow the first one.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="main.css">
  <title>Document</title>
</head>
<body>
  <div>
    <p>I am paragraph inside a div</p>
  </div>
  <p>I am a paragraph outside a div</p>
  <h3>I am a level three heading</h3>
  <p>I am a paragraph outside a div</p>
</body>
</html>
```

```
div ~ p {
  color: red;
}
```

KodeGo

ADJACENT SIBLING COMBINATOR

- more specific than the general sibling combinator
- This selector matches only the immediate siblings.
- Immediate siblings are the siblings that come right after the first element.
- To use the adjacent sibling combinator, specify the first element, then add the + character followed by the element you want to select that immediately follows the first element.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="main.css">
  <title>Document</title>
</head>
<body>
  <div>
    <p>I am paragraph inside a div</p>
  </div>
  <p>I am a paragraph outside a div</p>
  <h3>I am a level three heading</h3>
  <p>I am a paragraph outside a div</p>
</body>
</html>
```

```
div + a {
  color: red;
}
```

KodeGo

KodeGo

PSEUDO-CLASS SELECTORS

1. Pseudo-class selectors for links
2. Pseudo-class selectors for inputs
3. Pseudo-class selectors for position

Pseudo-class selectors select elements that are in a specific state.
Pseudo-class selectors start with a colon,:, followed by a keyword that reflects the state of the specified element.

SYNTAX:

```
element:pseudo-class-name {
    property: value;
}
```

PSEUDO-CLASS SELECTORS FOR LINKS

Selector	Description	Syntax
:link	applies styling when the element has not been visited before	`a:link { property: value; }`
:visited	applies when the element has been visited before in the current browser	`a:visited { property: value; }`
:hover	applies when the mouse pointer hovers over an element	`a:hover { property: value; }`
:focus	applies when a user has tabbed onto an element	`a:focus { property:value; }`
:active	applies when the element is selected after being clicked on and after holding down a mouse button	`a:active { property: value; }`

PSEUDO-CLASS SELECTORS FOR INPUTS

Selector	Description	Syntax
:focus	used for inputs as well	`input:focus { property: value; }`
:required	selects inputs that are required. Inputs that are required have the required attribute	`input:required { property: value; }`
:checked	selects checkboxes or radio buttons that have been checked	`input:checked { property:value; }`
:disabled	selects inputs that are disabled. Disabled inputs have the disabled attribute.	`input:disabled { property:value; }`

PSEUDO-CLASS SELECTORS FOR POSITION

Selector	Description	Syntax
:first-child	selects the first element, which will be the first child inside the parent container.	`a:first-child { property: value; }`
:last-child	selects the last element, which will be the last child inside the parent container.	`a:last-child { property: value; }`
:nth-child()	selects a child element inside a container based on its position in a group of siblings.	`a:nth-child(n) { property: value; }`
:first-of-type	selects elements that are the first of that specific type in the parent container.	`p:first-of-type { property: value; }`
:last-of-type	selects elements that are the last of that specific type in the parent container.	`p:last-of-type { property: value; }`

KodeGo

PSEUDO-ELEMENT SELECTORS

1. The ::before pseudo-element
2. The ::after pseudo-element
3. The ::first-letter pseudo-element
4. The ::first-line pseudo-element

- used for styling a specific part of an element - you can use them to insert new content or change the look of a specific section of the content.
- The :: character is followed by a keyword that allows you to style a specific part of the selected element.

SYNTAX:

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

::before PSEUDO-ELEMENT

- use to insert content before an element

```
p::before {
    property: value;
}
```

::after PSEUDO-ELEMENT

- use to insert content at the end of an element

```
p::after {
    property: value;
}
```

::first-letter PSEUDO-ELEMENT

- use to select the first letter of a paragraph, which is helpful when you want to style the first letter in a certain way

```
p::first-letter {
    property: value;
}
```

::first-line PSEUDO-ELEMENT

- use to select the first line of a paragraph

```
p::first-line {
    property: value;
}
```