

Selectors and Combinators

Cascading Style Sheets (CSS3)

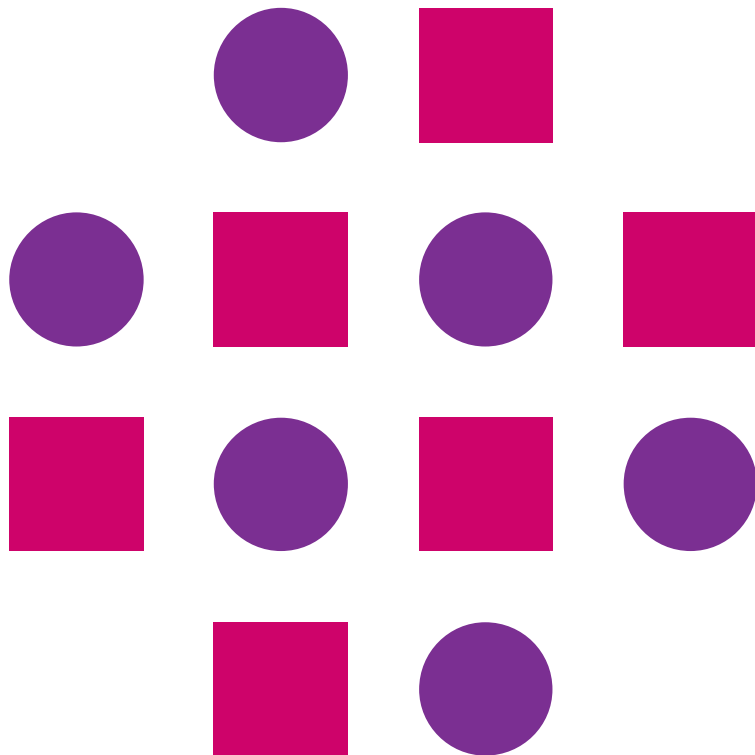


Table of Content

- Selectors
- Combinators



Selectors

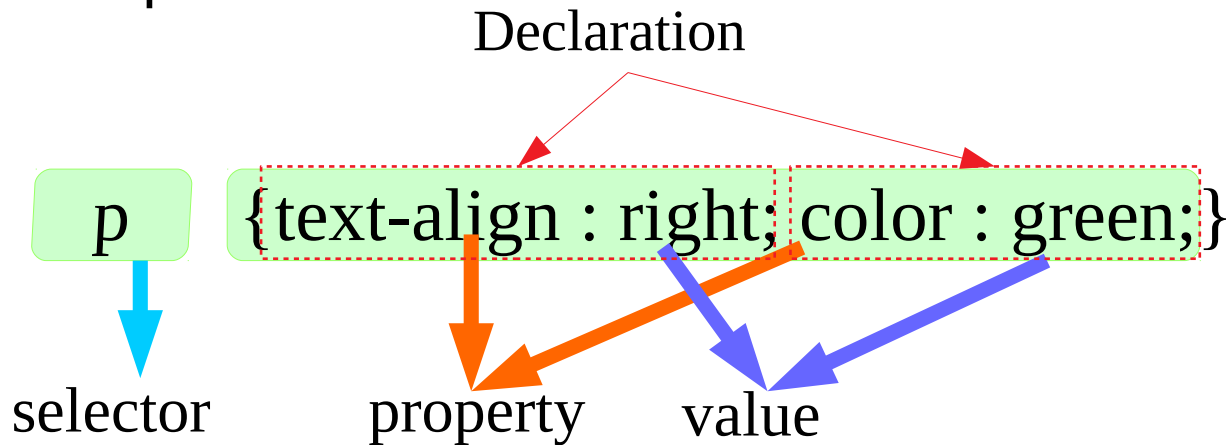
(Cascading Style Sheets 3)

The CSS selector

- The CSS selector points to the HTML element you want to style
- Selectors are the part of CSS rule-set
- CSS selectors select HTML elements according to its id, class, type, attribute etc
- The rule-set has 3 parts
 - Selector
 - Property
 - Value

Rule-set Syntax

- Selector { property : value }
- Example



Selectors

- Selectors tell the browser where to apply the rule
 - Element type Selector
 - ID Selector
 - Class Selector
 - Attributes Selector
 - Universal Selectors

Element Type Selector

- The element type selector selects element based on element name

Syntax :

Element-type-selector { property : value }

Element Type Selector

Example :

```
h1 {  
    font-family : "Times New Roman";  
    color : green;  
}
```


ID Selectors

- ID selector must be unique for entire page
- ID selector is used to specify single element
- The Selector is created by prefixing ID with hash or pound sign (#)

Syntax :

```
#id-selector { property : value; }
```

ID Selectors

Example :

```
#p { font-size : 12px;  
      color : blue;  
}
```

Class Selectors

- Class selector is used to specify group of elements
- Classes are not unique
- We can use same class on multiple elements
- We can use multiple class on the same element
- Class selector is defined with dot (.)

Class Selectors

Syntax :

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

Example :

```
.my-class {  
    font-size : 12px;  
    color : blue;  
}
```

Attribute Selectors

- An attribute selector matches any element that has a particular attribute
- We can create an attribute selector by putting the attribute, optionally with a value, in a pair of square brackets
- Element type selector can also be placed before attribute selector

Note : some browsers might not support attribute selectors

Attribute Selectors

Syntax :

```
[attribute] {  
    name : val;  
}  
Element-type-selector [attribute] { /* optional element selector */  
    name : val;  
}  
Element-type-selector [attribute operator "value" i] {  
    name : val;  
}
```

Attribute Selectors

- [attr] - Represents an element with an attribute name of attr
- [attr=value] - Represents an element with an attribute name of attr whose value is exactly value
- [attr operator value i] - Adding an i (or I) before the closing bracket causes the value to be compared case-insensitively (for characters within the ASCII range)
 - Operator : *=, ^=, ~=, |=, =, \$=

Attribute Selectors

- Setting deep pink color to all the elements where title attribute is used

Example :

```
[title] { /* title as attribute selector */  
    color : deeppink;  
}
```


Attribute Selectors

- Setting deep pink color to all the paragraph elements where title attribute is used

Example :

```
p[title] { /* title as attribute selector */  
    color : deeppink;  
}
```

Attribute Selectors

- Setting yellow background-color and italic font-style to all input elements whose type attribute is set to email

Example :

```
input[type="email"] {  /* no space between input and square bracket */  
    background-color : yellow;  
    font-style : italic;  
}
```

Attribute Selectors

- [attr~=value] - Represents an element with an attribute name of attr whose value is a whitespace-separated list of words, one of which is exactly value

```
/* All division in US English are blue */  
div[lang~="en-US"] {  
    color: blue;  
}
```

Attribute Selectors

- `[attr|=value]` - Represents an element with an attribute name of `attr` whose value can be exactly `value` or can begin with `value` immediately followed by a hyphen (-)
 - It is often used for language sub-code matches

```
/* Matches the elements with lang attribute that has the values en, en-US,  
en-GB etc */
```

```
[lang|=“en”] {  
    color: white;  
    background: blue;  
}
```

Attribute Selectors

- `[attr^=value]` - Represents an element with an attribute name of attr whose value is prefixed (preceded) by value

```
/* Internal links, beginning with "#" */  
a[href^="#"] {  
    background-color: gold;  
}
```

Attribute Selectors

- [attr\$=value] - Represents an element with an attribute name of attr whose value is suffixed (followed) by value

```
/* Anchor elements with an href ending with ".org" */  
a[href$=".org"] {  
    font-style: italic;  
}
```

Attribute Selectors

- [attr*=value] - Represents an element with an attribute name of attr whose value contains at least one occurrence of value within the string

```
/* Anchor elements with an href containing "example" */  
a[href*="example"] {  
    font-size: 2em;  
}
```

Attribute Selectors

- [attr operator value i] - Adding an i (or I) before the closing bracket causes the value to be compared case-insensitively (for characters within the ASCII range)

```
/* Links with "big" anywhere in the URL,  
   regardless of capitalization */  
a[href*="big" i] {  
  color: cyan;  
}
```


Universal Selector

- The CSS universal selector (*) matches elements of any type
- In CSS3 and later, the asterisk may be used in combination with namespaces
 - ns|* - matches all elements in namespace ns
 - *|* - matches all elements
 - |* - matches all elements without any declared namespace

Universal Selector

Syntax :

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

Example :

```
* {  
    color : blue;  
}
```

Universal Selector

- The asterisk is optional with simple selectors
- Example: *.example-class and .example-class are equivalent

Grouping Selectors

- Elements with same style sheet can be grouped together to avoid repetition
- Element selectors are grouped by separating with comma (,)

Example :

```
h1, h2 {  
    font-size : 12px;  
    color : blue;  
}
```

Combinators

(Cascading Style Sheets 3)

Child Combinator

- The `>` combinator selects nodes that are direct children of the first element

Syntax : `E1 > E2 { . . . }`

Example :

```
div > ul {  
    color : green;  
}
```

Adjacent Sibling Combinator

- The + combinator selects adjacent siblings
- This means, second element immediately follow the first, and both share the same parent

Syntax : E1 + E2 { . . . }

Example :

```
div + p {  
    color : blue;  
}
```

General Sibling Combinators

- The ~ combinator selects siblings
- This means that the second element follows the first (though **not necessarily immediately**), and both share the same parent

Syntax : E1 ~ E2 { . . . }

Example :

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


Descendent Combinator

- The space combinator selects nodes that are descendants of the first element

Syntax : E1 E2 { . . . }

Example :

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

Web Stack Academy (P) Ltd

#83, Farah Towers,
1st floor, MG Road,
Bangalore - 560001

M: +91-80-4128 9576

T: +91-98862 69112

E: info@www.webstackacademy.com

*Thank
you*