CSS3

chapter 5 / 7

CSS3

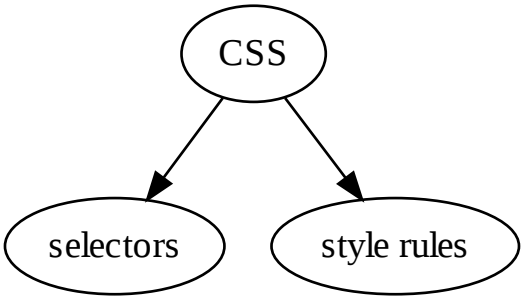
Sections in this chapter:

1. CSS basics
2. Selectors
3. Combinators

5-1. CSS basics

CSS, or **Cascading Style Sheets**, is a language we use to control **style** and **layout** of HTML content. *5-1-1*

It is expressed through **selectors** and **style rules** which define how a certain element should be rendered in an HTML-page. *5-1-2*



A rule consists of a **selector** and a **declaration**. *5-1-3*

```
selector {
    style-rule;
}
```

The **selector** specifies which elements to style and could for instance be an explicit HTML tag name, an ID or a class name. *5-1-4*

```
body {
    style-rule;
}
```

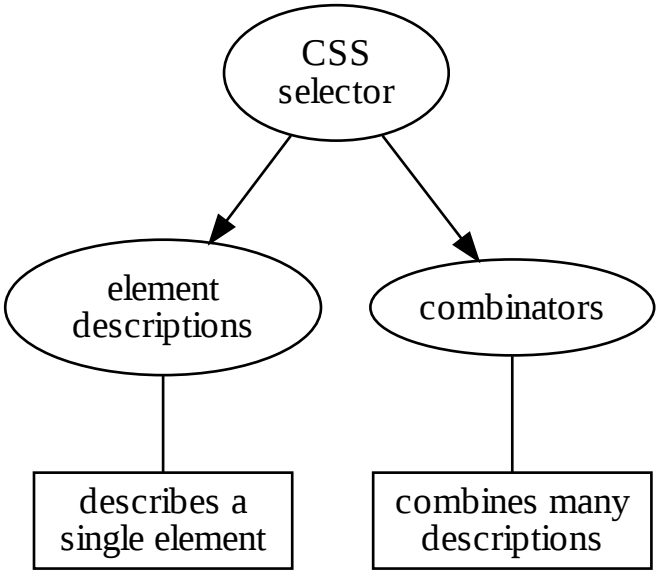
A **declaration** is written as a **property/value pair**, defining how to style the selected element. The property is an attribute, for instance `color`. The value depends on the attribute: *5-1-5*

```
body {
    color: red;
}
```

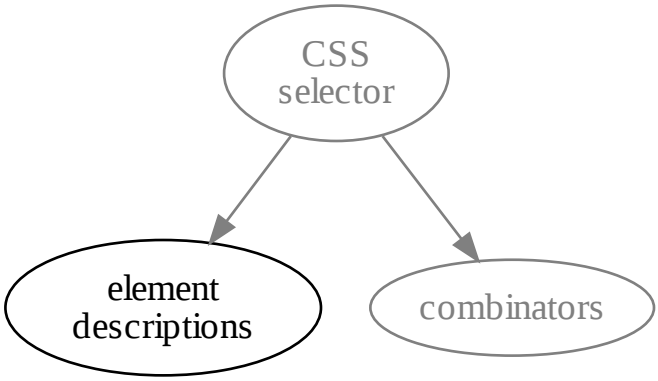
5-2. Selectors

To apply a CSS rule to an HTML element, we need to target the element using a selector. *5-2-1*

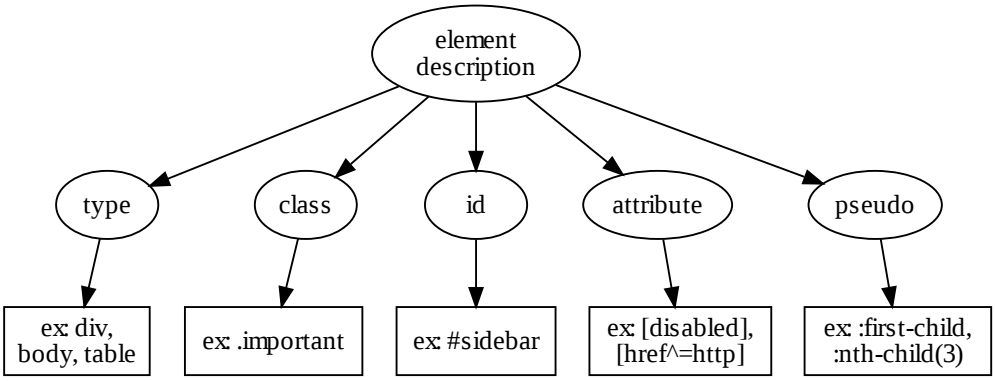
Just like CSS splits into **selectors** and **style rules**, so does **selectors** split into **descriptions** and **combinators**. *5-2-2*



The most important part of a CSS selector is to **describe elements** that should be selected. *5-2-3*



There are **five different aspects** that we can describe, each with its own syntax: *5-2-4*



These **can be combined** however you see fit. Here is an (exaggerated) example using all of them: *5-2-5*

```
button[disabled]#deletemsg.big:first-child
```

This would match all *5-2-6*

- nodes of type **button**
- that has a disabled **attribute**
- and **id** is `deletemsg`
- and class attribute contains `big`
- and it is the **first child** of its parent

As per usual, **the details can be found on MDN**: *5-2-7*
https://developer.mozilla.org/en-US/docs/Glossary/CSS_Selector

We will, however, look at a few simple examples.

Let's say we have the following HTML: *5-2-8*

```
<h2>Course objectives</h1>
<p id="first-paragraph">
    To endow participants with awesome powers
</p>
<h2>CSS syntax</h2>
<p class="warning">Is awesome</p>
<a href="https://css-tricks.com/css-is-awesome/">
    CSS-Tricks
</a>
```

To select by type description: *5-2-9*

```
h2 {
    color:red;
}
```

The above rule will apply to all h2 headers, since the rule is set to all h2 elements.

To select by id description: *5-2-10*

```
#first-paragraph {
    font-size: 12pt;
}
```

This will apply the above rule to the first paragraph, with the id `first-paragraph`. An **id** uniquely identifies an element on a page, and may only be used for one element, not several.

To select by class: *5-2-11*

```
.warning {
    color: orange;
}
```

The above rule will apply only to the paragraphs with the class `warning`, in our case the second paragraph. A **class** is any name inside an HTML `class` attribute. It may be applied to several elements.

To select by attribute: *5-2-12*

```
a[href^="https"] {
    color: green;
}
```

There is also a selector which targets values in attributes that end with a specific word or value. *5-2-13*

```
a[href$="shop"] {
    background: green;
}
```

What if I want to target an element whose value contains a specific word or value? For this there is a contains selector: *5-2-14*

```
a[href*="products"] {
    background: green;
}
```

There is a bunch of ways to match values, check out MDN Attribute Selectors for more examples! *5-2-15*

Q Ok, so ^= allows us to match against the **beginning of an attribute value**. But what would be the point of that first example we saw? *5-2-16*

A If all local links are relative, which is normally the case, then this would be an excellent way to **catch all external links**, to make them look or behave differently. *5-2-17*

Finally, some notes on the pseudo-classes: *5-2-18*

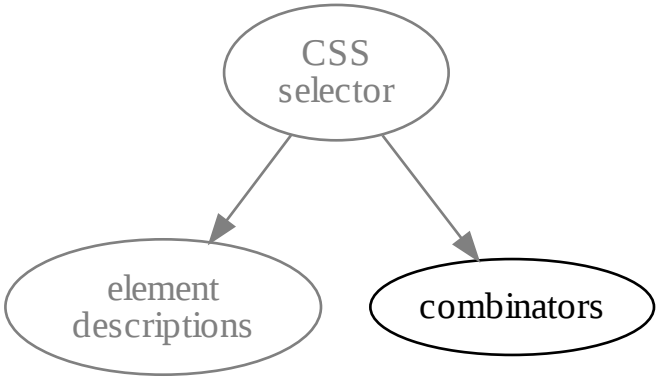
They are **prefixed with** :, and allow matching on **position or state**.

- `:first-child`
- `:link`
- `:hover`, `:active`, `:focus`

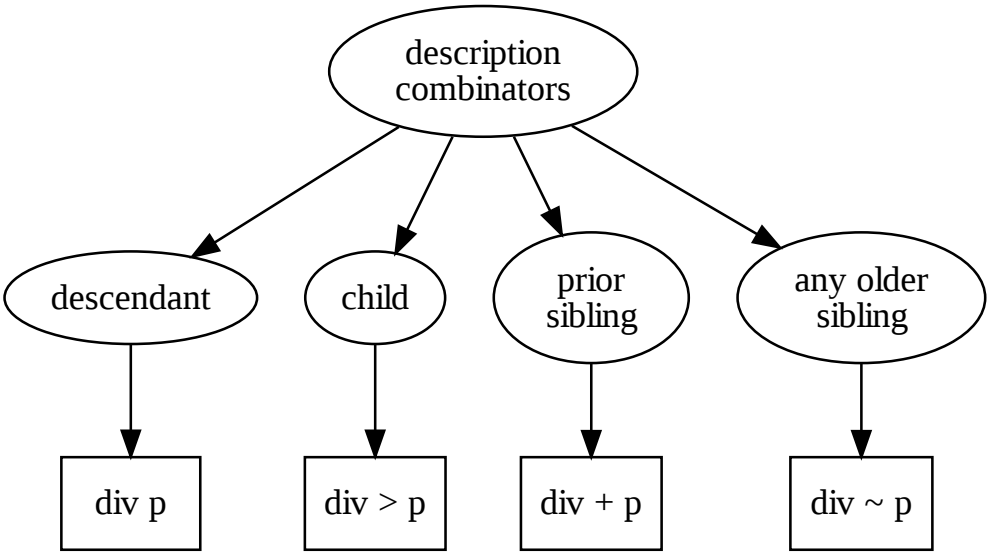
MDN has very good documentation about pseudo-classes.

5-3. Combinators

Let's now look at the other half of selectors, namely how we **combine** descriptions! *5-3-1*



There are **four different ways** that descriptions can be combined, which we'll look at one at a time: *5-3-2*



The perhaps most common one is the **descendant combinator**. By having **two descriptions with space between**: *5-3-3*

```
div p
```

we match all elements that *5-3-4*

- **match the last description**
- have an **ancestor matching the first description**. This can be any number of generations up the tree.

The **child combinator**: *5-3-5*

```
div > p
```

is very similar to descendant selector, but here the first selector must match the **parent** and not just any ancestor.

Thus the child combinator is **smaller in scope** than the descendant combinator. *5-3-6*

The **sibling combinator**: *5-3-7*

```
div ~ p
```

is similar to the descendant combinator, but **works horizontally** instead.

It matches elements that: *5-3-8*

- **match the last description**
- have an **older sibling that matches the first description**

Finally the **adjacent sibling combinator**: *5-3-9*

```
div + p
```

works in the exact same way, but requires the **neighbouring older sibling to match the first description**.

The two **sibling combinators are not often used**, but they are **good at what they do**.

Solving that problem with other means would require **brittle workarounds**, something you often see from web developers who don't know about them.