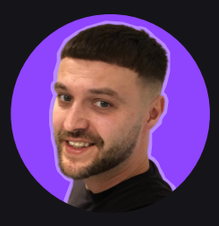
Learn CSS By Use Cases

ebook by Joe Harrison

Syntax

Selectors



Joe Harrison
@frontendjoe

Selectors

Syntax

Intro

In CSS, selectors are patterns used to target the element we need to style.

Simple Selectors

Simple selectors target things that are directly linked to HTML. Classes can be added to HTML elements, tag/element selectors refer to HTML element tag names and the universal selector targets all HTML elements.

Combinator Selectors

Combinator selectors allow us to combine many selectors into one selector pattern. They can be complex but are very useful in CSS.

Selectors

Syntax

Special Power

Selectors are what drives CSS to do what it needs to do - target and style specific elements. When building advanced components like dropdown menus, combinator selectors are a crucial part of the functionality.

Tips

If you’re not a massive fan of long winded selector patterns, then in most cases - classes will work better for you. I’d personally avoid using ID selectors in your everyday game and only use the universal selector to reset certain properties. Combinators offer a great way to create impressive animations and transitions, you’ll find examples in my source code.

Selectors

Simple Selectors

Simple Selectors

Selectors

Class

```
.form-control {
  /* property pairings */
}
```

Class selectors always begin with a . character

Class selectors allows us to create reusable style blocks, then apply them in our HTML. An element can inherit from multiple classes.

ID

```
#root {
  /* property pairings */
}
```

In a React app the whole website is wrapped in a div with an ID of #root

An ID selector will target an HTML element that has a specific ID attribute applied. ID is usually reserved for styling framework elements.

Simple Selectors

Selectors

Tag/Element

```
ul {
  /* property pairings */
}
```

Tag selectors will match the name of a speficified HTML element

Tag selectors will target any HTML elements of a given tag or type. Their primary use case is to reset the default styling applied by browsers

Universal

```
* {
  /* property pairings */
}
```

The universal selector will target all HTML elements. It’s only real use case is to reset certain properties on all elements.

Simple Selectors

Selectors

Knowledge Gained

- 🏆 Selectors are patterns used to target the element we need to style
- 🏆 There are four simple selectors and many combinator selectors (patterns)
- 🏆 Classes make your code reusable
- 🏆 The ID selector is more relevant when working with JavaScript frameworks
- 🏆 Tag/element selectors are most useful when resetting default CSS

Joe Harrison
@frontendjoe

Selectors

Combinator Selectors

Combinator Selectors

Selectors

Chained

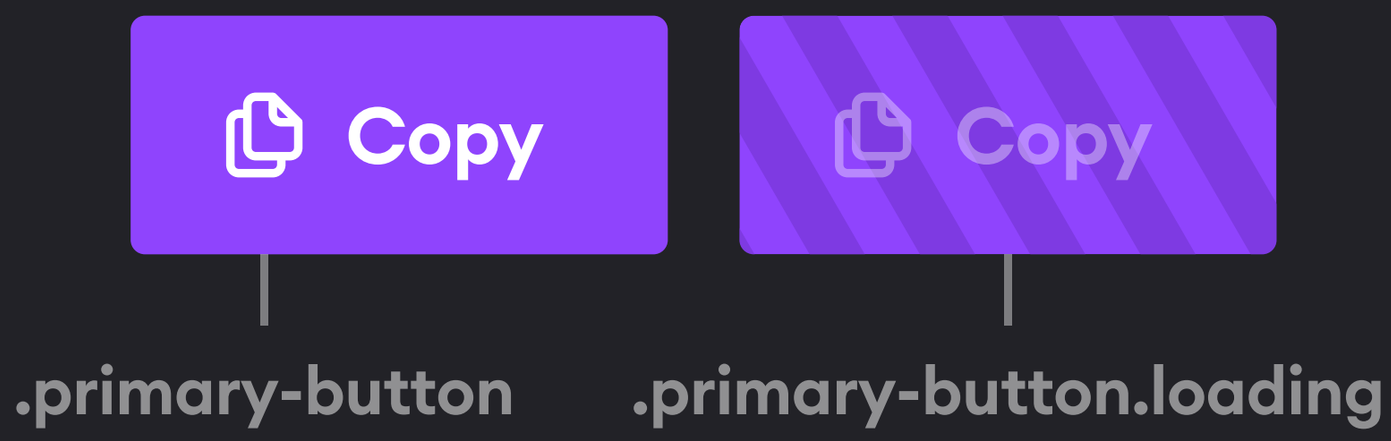
If we need to target a HTML element with multiple selectors then we can use the chained combinator syntax.

If we use this example HTML:

```
<button class="primary-button loading">
  <!-- button content -->
</button>
```

To style the combination of both classes we will use chained syntax. This means that the loading class will only be applied when added to a primary-button. With colored buttons the loading class will often be specific to that color.

Chained



class

```
.primary-button.loading {
  /* property pairings */
}
```

class

In CSS, placing multiple selectors next to each other is known as chaining

Combinator Selectors

Selectors

Child

To target any child selectors of a given element, we can add extra selectors to our pattern separated by a space character.

If we use this example HTML:

```
<div class="avatar online">
  <span></span> <!-- badge -->
</div>
```

By using child combinator syntax we only need to add the “online” class to the parent (less HTML). Then we can add styles easily to both the parent and any child elements, when the parent has the “online” class name.

Child

selector space

```
.avatar.online span {
  /* "online" badge styles */
}
```

selector

We can style any child elements if the parent **.avatar** has a class of **.online**

Combinator Selectors

Selectors

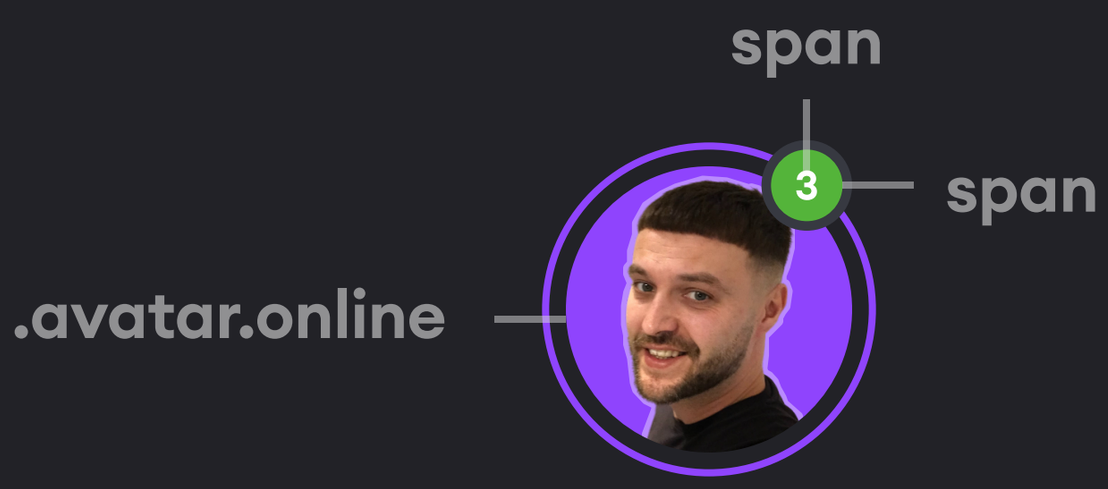
Direct Child

The direct child combinator is similar to child, however it will only target elements that are placed directly inside the selector.

```
<div class="avatar">
  <span> <!-- badge -->
    <span> <!-- badge-text -->
      3
    </span>
  </span>
</div>
```

Here we want to set a green background, but only on the first span as the HTML has changed. Our previous child example would apply this to both but we can fix this by targeting only direct children.

Direct Child



Greater than character

```
.avatar.online > span {
  /* "online" badge styles */
}
```

Target the badge text element

```
avatar.online > span > span {
  /* "online" badge text styles */
}
```

Combinator Selectors

Selectors

Adjacent Sibling

The adjacent sibling combinator separates two selectors with a + character. It essentially selects the element that appears immediately after, in the HTML.

```
<div class="wrapper">
  <input class="textbox" />
  <span class="icon">
</div>
```

Here we want to set the .icon color to our app's primary color when .textbox is focused. The following example will be fine if we only need to target one element. There is however a similar combinator that will target many adjacent siblings as you'll learn next. For this reason I tend to never use this adjacent sibling syntax.

Adjacent Sibling

Target only one sibling

Left side

Right side

Separator

Pseudo Class **:focus** which you'll learn later

```
.textbox:focus + .icon {
  /* property pairings */
}
```

Any selector on the **right side** will only be applied if it targets an element that is **immediately after** the selector on the **left side**

Combinator Selectors

Selectors

General Siblings

The general siblings combinator separates two selectors by a ~ (tilde) character. It essentially selects every element that appears after and at the same level, in the HTML.

```
<div class="wrapper">
  <input class="textbox" />
  <label class="label">
    Username
  </label>
  <span class="icon">
</div>
```

Here we want to set a color for both .label and .icon when the textbox is focused. The general siblings combinator will only target elements that appear after, but we can use the position absolute trick to make them appear before.

General Siblings

Target many siblings

You'll learn more about taking elements out of the **flow** later

Left side

Right side

Separator

```
.textbox:focus ~ .label {
  /* property pairings */
}

.textbox:focus ~ .icon {
  /* property pairings */
}
```

Combinator Selectors

Selectors

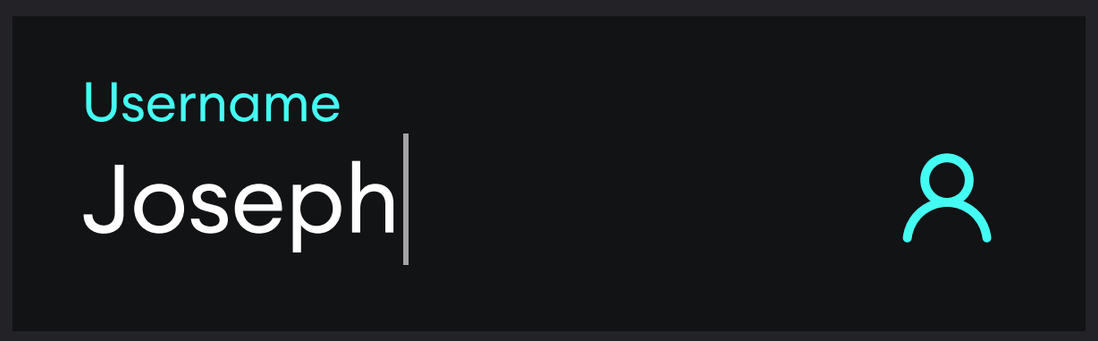
Stacked

To create style blocks for multiple selector patterns we can use the stacked selector syntax. It essentially consists of multiple selector patterns separated by a comma. I personally like to stack them up for better readability.

```
<div class="wrapper">
  <input class="textbox" />
  <label class="label">
    Username
  </label>
  <span class="icon">
</div>
```

In the previous use case we declare two separate blocks to target the general siblings, let’s clean that up with some stacked syntax.

Stacked



Avoid duplicating style blocks by stacking up selector patterns

```
.textbox:focus ~ .label,
.textbox:focus ~ .icon {
  /* property pairings */
}
```

Multiple patterns separated by commas

Selectors

Syntax

Knowledge Gained

- 🏆 Combinators combine two or more selectors into the same pattern
- 🏆 Chaining is useful when styling for multiple rules
- 🏆 Child styling results in less HTML
- 🏆 The general siblings combinator is more useful than adjacent sibling
- 🏆 Stacked selector syntax means we can define style blocks for multiple patterns

Joe Harrison
@frontendjoe