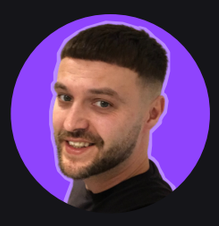
Learn CSS By Use Cases

ebook by Joe Harrison

Syntax

Pseudo Classes



Joe Harrison
@frontendjoe

Pseudo classes

Syntax

Intro

Pseudo classes offer a way of styling the different special states of each element. Things like whether the element is hovered or disabled are styled using pseudo classes. There are probably hundreds of these guys that I haven’t mentioned, but honestly - I just don’t use them.

Syntax

All psuedo classes are prefixed with a colon and accessed via chaining. Some pseudo classes are actually functions themselves, with an identical syntax to that of Property Value Type functions. A pattern can contain many psuedo classes.

Pseudo classes

Syntax

Special Power

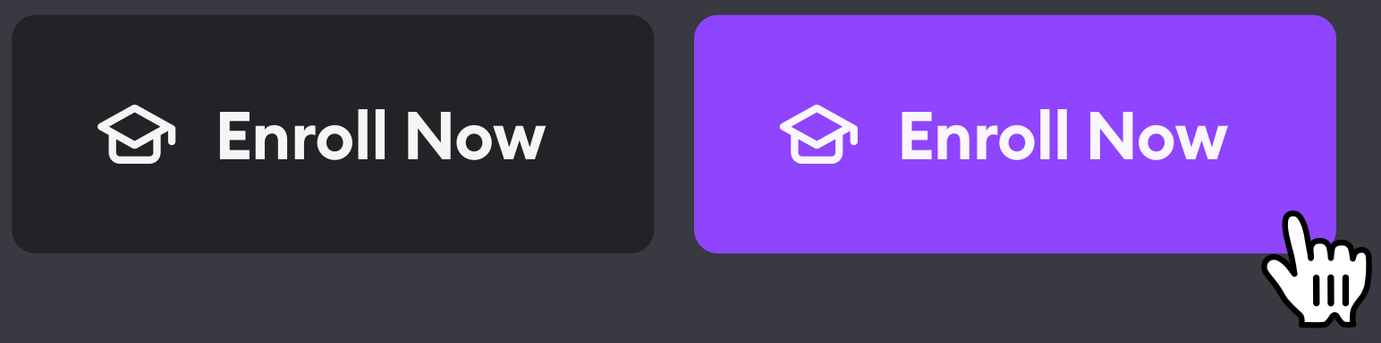
Catering for all of the special states of elements will improve your UI and UX massively. Knowing the in's and out's of pseudo classes and their styling powers is a great skill to have in your frontend toolbox.

Tips

There are many more pseudo selectors than I've shown use cases for here. My advice however, is to stick with these ones initially and master them, before moving on to more obscure ones. The newer experimental ones can be fun to play with, but please master my core set here first.

:hover

Element is about to be interacted with



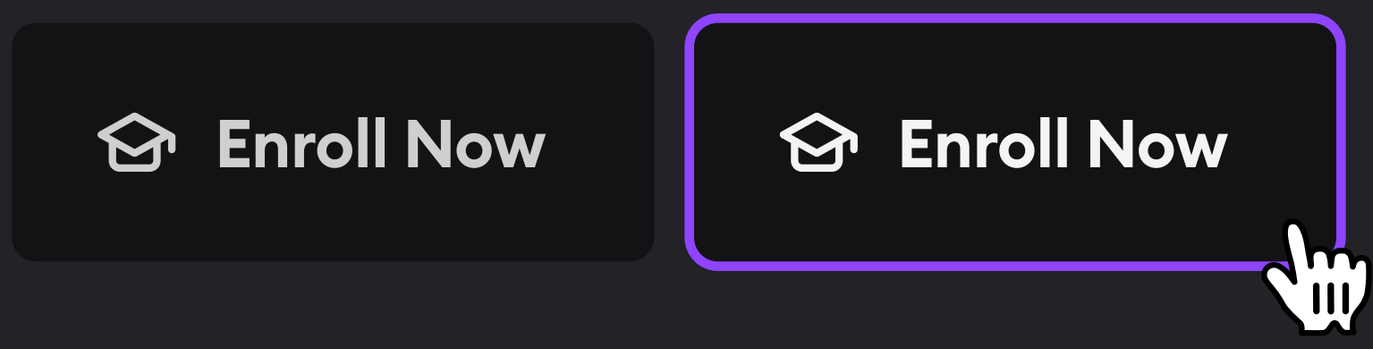
.button .button:hover

Chained selector

```
.button:hover {
  /* hover styles */
}
```

:focus

Element is focused



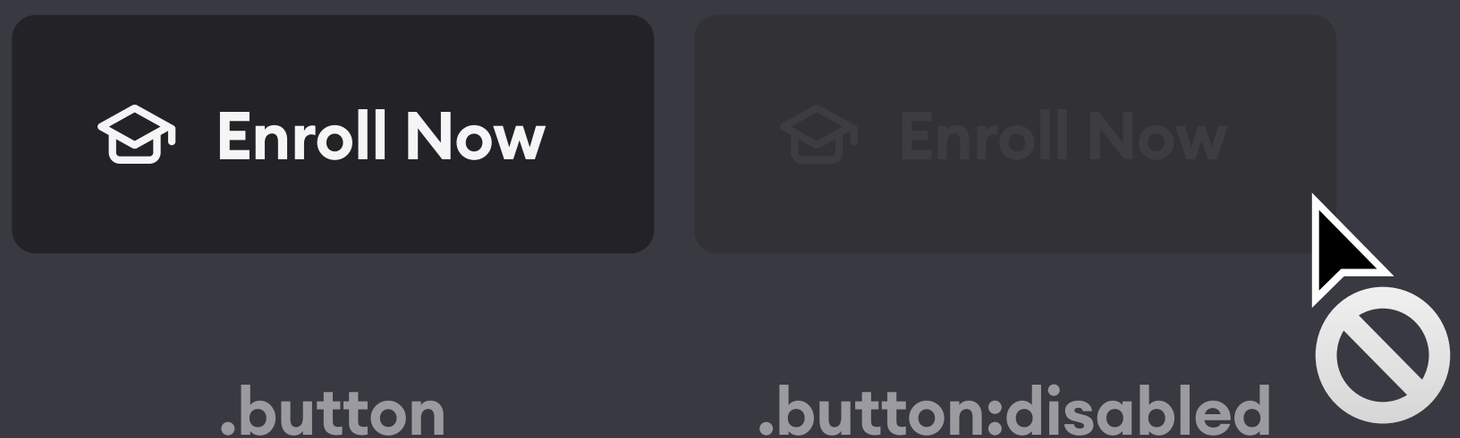
.button .button:focus

```
.button:focus {
  /* focus styles */
}
```

It’s good practice to add a **:focus** state to any interactive elements. This **improves accessibility** as it will be much easier to use your website - without a mouse for example.

:disabled

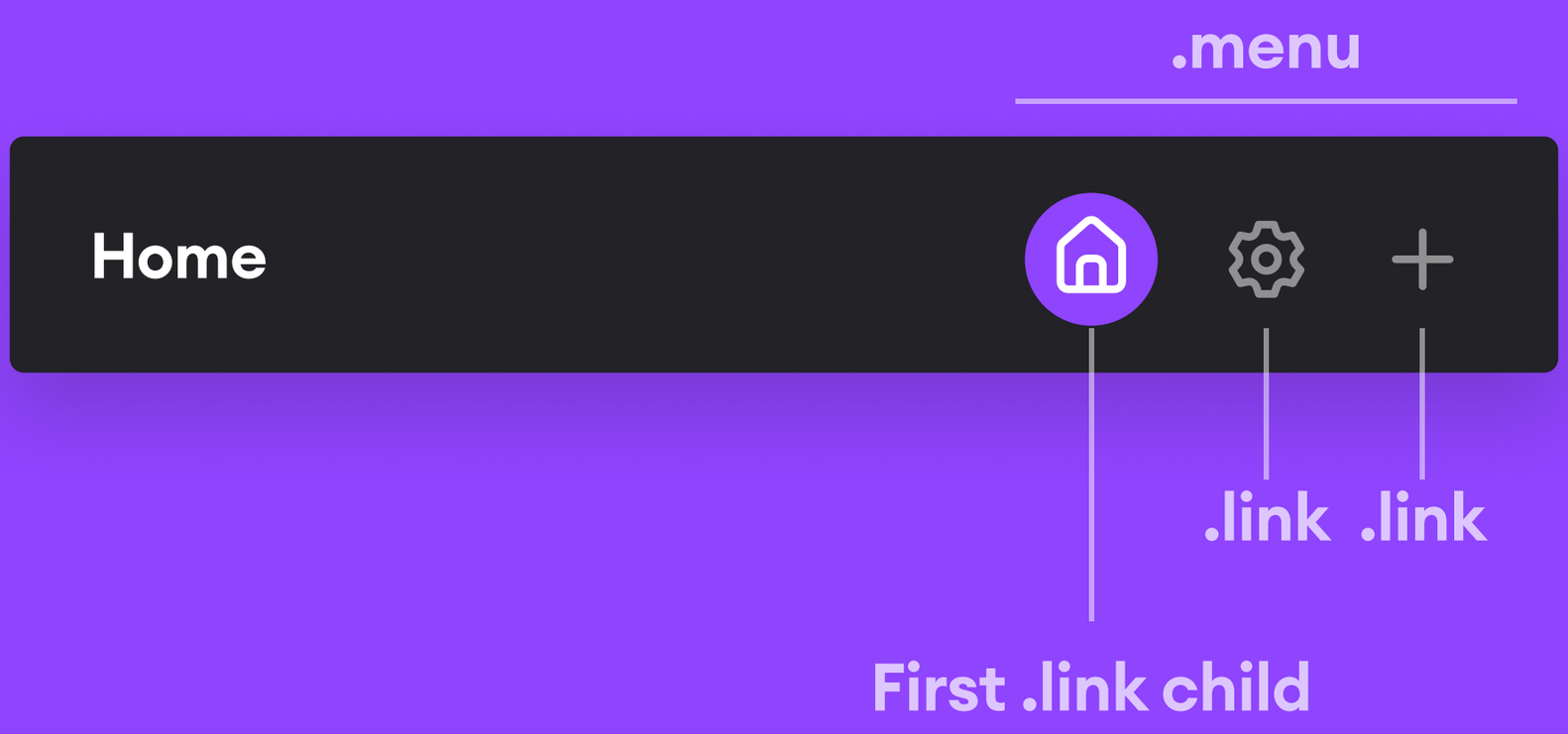
Element does not allow interactions




```
.button:disabled {
  /* disabled styles */
}
```

It’s good practice to clearly indicate to the user that an element does not allow interactions

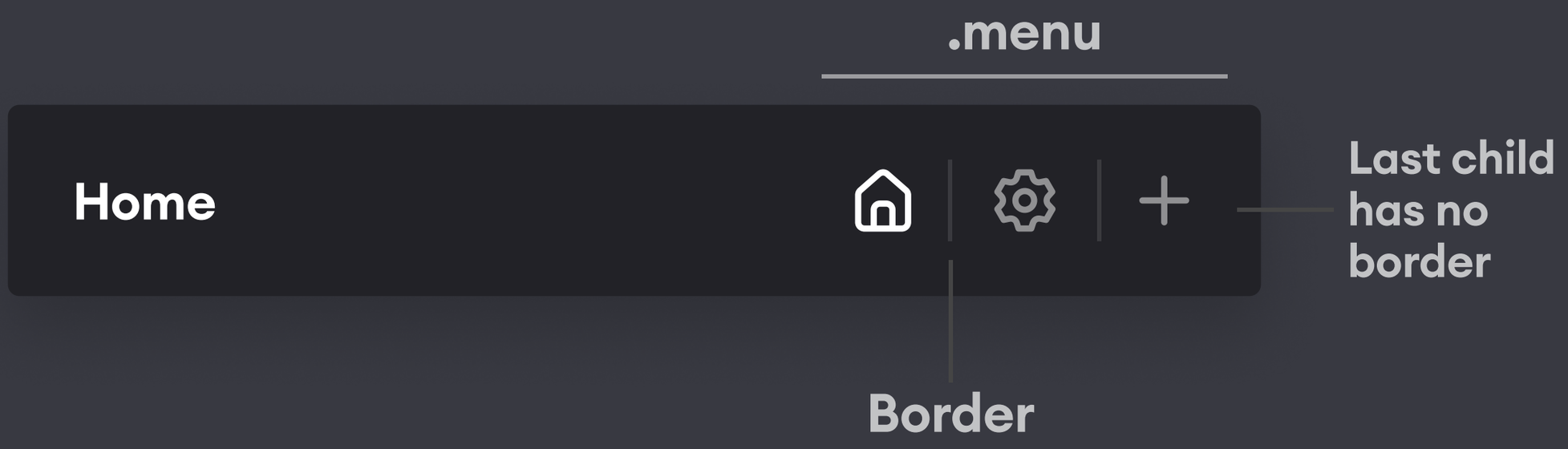
:first-child



Child selector

```
.menu .link:first-child {
  /* add purple background */
}
```

:last-child



It's common in CSS to add properties, then remove them via individual selectors

```
.menu .link { /* add border */ }

.menu .link:last-child {
  /* remove border */
}
```

As this selector pattern is more specific than the first, it's styles will take priority

You'll see on the next use case that there is now a cleaner way of achieving this

:not()

Elements that **don't** meet a condition

No need to set border styles and remove

```
.menu .link:not(:last-child) {
  /* add border */
}
```

We write a bit less code by using the **:not()** function and passing in our **:last-child** pseudo, pretty clean

:nth-child()

Elements of a set index

User	Age	Grade
Elvis Harris	24	87%
Claire Hatcliffe	31	83%
Kerry Keenan	43	80%

Pre-defined keyword

```
.table .row:nth-child(odd) {
  /* add grey background */
}
```

Another clean **function** that can target odd/even elements. You can also pass a **non zero index** if required

Pseudo Class

Syntax

Knowledge Gained

- 🏆 Pseudo classes target the special states of each element
- 🏆 They are prefixed with a colon character and accessed via a chained combinator
- 🏆 Multiple pseudo classes can appear in one selector pattern
- 🏆 Styling pseudo classes will improve UX
- 🏆 :not and :nth-child are useful functions that help us write clean code

Joe Harrison
@frontendjoe