

Podcast Episode #126: We chat GitHub Actions, fake boyfriends apps, and the dangers of legacy code. [Listen now.](#)

# Is there a CSS parent selector?

Asked 10 years, 4 months ago   Active 3 months ago   Viewed 1.8m times

▲ How do I select the `<li>` element that is a direct parent of the anchor element?

2958 As an example, my CSS would be something like this:

▼  
★  
432

```
li < a.active {  
  property: value;  
}
```

Obviously there are ways of doing this with JavaScript, but I'm hoping that there is some sort of workaround that exists native to CSS Level 2.

The menu that I am trying to style is being spewed out by a CMS, so I can't move the active element to the `<li>` element... (unless I theme the menu creation module which I'd rather not do).

Any ideas?

css

css-selectors

edited Jul 24 at 22:14



Peter Mortensen

14.5k 19 89 118

asked Jun 18 '09 at 19:59



jcuenod

25k 10 54 91

31 Answers

1

2

next

▲ There is currently no way to select the parent of an element in CSS.

○○○○ If there was a way to do it, it would be in either of the current CSS selectors specs:

2369



- [Selectors Level 3 Spec](#)
- [CSS 2.1 Selectors Spec](#)



In the meantime, you'll have to resort to JavaScript if you need to select a parent element.

The [Selectors Level 4 Working Draft](#) includes a `:has()` pseudo-class that works the same as the [jQuery implementation](#). As of 2019, [this is still not supported by any browser](#).

Using `:has()` the original question could be solved with this:

```
li:has(> a.active) { /* styles to apply to the li tag */ }
```

edited Jun 14 at 14:49



[railgun](#)

559 1 7 20

answered Jun 18 '09 at 20:16



[Dan Herbert](#)

71.4k 41 166 212

61 It would seem that it has already been suggested and rejected: [stackoverflow.com/questions/45004/...](#) – [RobM](#) Oct 27 '10 at 12:22

13 Looks like the [subject selector](#) has been revisited, except by using a `!` now: The subject of the selector can be explicitly identified by appending an exclamation mark (!) to one of the compound selectors in a selector. – [animuson](#) ♦ Jan 29 '12 at 21:30

13 The prepended `$` looked better for me... the appended `!` can be overlooked more easily. – [Christoph](#) Feb 13 '12 at 11:25

49 Major plot twist! The Selectors 4 WD was just updated [today](#) to exclude the subject indicator from the fast profile, which is to be used by CSS implementations. If this change remains, it means you won't be able to use the subject indicator in stylesheets anymore (unless you use add some sort of polyfill like the one described in [another answer](#)) - you can only use it with the Selectors API and anywhere else that the complete profile may be implemented. – [BoltClock](#) ♦ May 2 '13 at 15:19 ✎

53 Another major update: it looks like they're considering doing away with the subject selector syntax altogether and replacing it with the `:has()` pseudo everybody has come to know and love from jQuery. The latest ED has removed all references to the subject indicator, and replaced it with the `:has()` pseudo. I don't know the exact reasons, but the CSSWG held a poll some time ago and the results must have influenced this decision. It's most likely for compatibility with jQuery (so it can leverage qSA without modifications), and because the subject indicator syntax proved too confusing. – [BoltClock](#) ♦ May 4 '14 at 14:28 ✎



I don't think you can select the parent in CSS only.

146

But as you already seem to have an `.active` class, it would be easier to move that class to the `li` (instead of the `a`). That way you can access both the `li` and the `a` via CSS only.

can access both the `li` and the `a` via CSS only.

edited Jul 24 at 22:21



Peter Mortensen

14.5k 19 89 118

answered Jun 18 '09 at 20:08



jeroen

84.2k 18 102 126

4 You can't shift the pseudo selector to the list item, as it is not a focusable element. He's using the `:active` selector, so when the anchor is active he wants the list item to be affected. List items will never be in the active state. As an aside, it's unfortunate that such a selector doesn't exist. Getting a pure CSS menu to be fully keyboard accessible seems to be impossible without it (using sibling selectors you can make submenus created using nested lists to appear, but once the list gains focus it becomes hidden again). If there are any CSS-only solutions to this particular conun – user914183 Aug 26 '11 at 13:46

11 @Dominic Aquilina Take a look at the question, the OP is using a class, not a pseudo selector. – jeroen Aug 26 '11 at 14:34

You can use [this script](#):

119

```
*! > input[type=text] { background: #000; }
```

This will select any parent of a text input. But wait, there's still much more. If you want, you can select a specified parent:

```
.input-wrap! > input[type=text] { background: #000; }
```

Or select it when it's active:

```
.input-wrap! > input[type=text]:focus { background: #000; }
```

Check out this HTML:

```
<div class="input-wrap">
  <input type="text" class="Name"/>
  <span class="help hide">Your name sir</span>
</div>
```

You can select that `span.help` when the `input` is active and show it:

```
.input-wrap! .help > input[type=text]:focus { display: block; }
```

There are many more capabilities; just check out the documentation of the plugin.  
BTW, it works in Internet Explorer.

edited Jul 24 at 22:24



Peter Mortensen

14.5k 19 89 118

answered Aug 13 '11 at 10:13



Idered

1,577 1 14 18

- 5 suppose using jquery `parent()` would be faster. This need testing, however – Dan Sep 22 '11 at 19:30
- 2 @Idered It fails when you have CSS declaration of a Selector Subject with no child selector ( `#a!` alone throws an error, `#a! p` works), and so the others will not work either because of `Uncaught TypeError: Cannot call method 'split' of undefined` : see [jsfiddle.net/HerrSerker/VkVPs](http://jsfiddle.net/HerrSerker/VkVPs) – yunzen Apr 16 '13 at 15:33
- 3 @HerrSerker I think `#a!` is an invalid selector, what should it select? – Idered Apr 17 '13 at 13:41
- 2 @Idered I don't know. The spec doesn't say it is illegal. `#a!` should select itself. At least there should be no error in the JavaScript – yunzen Apr 17 '13 at 15:31
- 5 Per my comment on the accepted answer, it looks like the polyfill may be required even in the near future after all, because the subject indicator may never be implemented by browsers in CSS. – BoltClock ♦ May 2 '13 at 15:35

As mentioned by a couple of others, there isn't a way to style an element's parent/s using just CSS but the following works with [jQuery](#):

100

```
$("#a.active").parents('li').css("property", "value");
```

edited May 2 '13 at 5:11



Alastair

5,923 2 31 29

answered Dec 14 '09 at 14:51



zcrar70

2,744 2 19 16

- 21 The `<` selector does not exist (verified using jQuery 1.7.1). – Rob W Feb 23 '12 at 17:53
- 6 Perhaps that `<` -syntax worked in 2009 but I've updated it (for 2013). – Alastair May 2 '13 at 5:10
- 5 Even better, use jQuery's built-in [:has\(\) selector](#): `$("#li:has(a.active)").css("property", "value");` . It reads similarly to CSS 4's proposed `! selector`. See also: [:parent selector](#), [.parents\(\) method](#), [.parent\(\) method](#). – Rory O'Kane May 8 '13 at 22:12 ✎
- 7 And rather than using `.css("property", "value")` to style the selected elements, you should usually `.addClass("someClass")` and have in your CSS `.someClass { property: value }` ([via](#)). That way, you can notate the style with the full power of CSS and any preprocessors you are using. – Rory O'Kane May 8 '13 at 22:20 ✎



63



There is no parent selector; just the way there is no previous sibling selector. One good reason for not having these selectors is because the browser has to traverse through all children of an element to determine whether or not a class should be applied. For example, if you wrote:

```
body:contains-selector(a.active) { background: red; }
```

Then the browser will have to wait until it has loaded and parsed everything until the `</body>` to determine if the page should be red or not.

The article [Why we don't have a parent selector](#) explains it in detail.

edited Jul 24 at 22:29



Peter Mortensen

14.5k 19 89 118

answered Jan 21 '14 at 8:43



Salman A

198k 69 360 458

11 so make the browser faster. the internet itself faster. this selector is definitely needed, and the reason for not implementing it is, sadly, because we live in a slow, primitive world. – [vsync](#) Feb 19 '14 at 21:33

10 actually, the selector would make a very fast browser look slow. – [Salman A](#) Feb 20 '14 at 4:09

3 I trust that browser developers would come up with an implementation which is (at least) as fast as the javascript version, which is the one people end up using anyway. – [Marc.2377](#) Jun 9 '18 at 0:15



53



There isn't a way to do this in CSS 2. You could add the class to the `li` and reference the `a` :

```
li.active > a {  
  property: value;  
}
```

edited Jul 24 at 22:19



Peter Mortensen

14.5k 19 89 118

answered Jun 18 '09 at 20:06



Josh

5,352 1 30 54

3 by making the `a` element `display: block` you can style it to fit the whole of the `li` area. if you can explain what style you are looking for perhaps I could help

by making the `a` element `display: block` you can style it to fit the whole of the `li` area. If you can explain what style you are looking for perhaps I could help with a solution. – [Josh](#) Jun 18 '09 at 21:53

Yes: [:has\(\)](#)

53

Browser support: [none](#)

edited Jul 24 at 22:48



[Peter Mortensen](#)

14.5k 19 89 118

answered Jan 20 '18 at 20:22



[Yukulélé](#)

6,924 7 41 63

Try to switch `a` to `block` display, and then use any style you want. The `a` element will fill the `li` element, and you will be able to modify its look as you want. Don't forget to set `li` padding to 0.

34

```
li {  
  padding: 0;  
  overflow: hidden;  
}  
a {  
  display: block;  
  width: 100%;  
  color: ..., background: ..., border-radius: ..., etc...  
}  
a.active {  
  color: ..., background: ...  
}
```

edited Jul 24 at 22:25



[Peter Mortensen](#)

14.5k 19 89 118

answered Nov 23 '11 at 9:03



[Raseko](#)

349 3 3

The CSS selector “[General Sibling Combinator](#)” could maybe used for what you want:

32

```
E ~ F {  
  property: value;  
}
```

This matches any `F` element that is preceded by an `E` element.

edited Aug 23 '17 at 12:27



Cody Gray ♦

202k 38 412 490

answered Jun 30 '11 at 14:00



cobaasta

561 4 2

15 This is just sibling selector, it's not child, parent... not really answering the question mate – Alireza Jun 16 '17 at 18:08



26



Not in CSS 2 as far as I'm aware. CSS 3 has more robust selectors but is not consistently implemented across all browsers. Even with the improved selectors, I don't believe it will accomplish exactly what you've specified in your example.

answered Jun 18 '09 at 20:09



Mark Hurd

12k 2 22 28



25



I know the OP was looking for a CSS solution but it is simple to achieve using jQuery. In my case I needed to find the `<ul>` parent tag for a `<span>` tag contained in the child `<li>`. jQuery has the `:has` selector so it's possible to identify a parent by the children it contains:

```
$("ul:has(#someId)")
```

will select the `ul` element that has a child element with id *someId*. Or to answer the original question, something like the following should do the trick (untested):

```
$("li:has(.active)")
```

answered May 16 '13 at 22:04



David Clarke

9,671 7 74 97

3 Or use `$yourSpan.closest("ul")` and you'll get the parent UL of your span element. Nevertheless your answer is completely offtopic imho. We can answer all of the CSS-tagged questions with a jQuery solution. – Robin van Baalen Jul 18 '13 at 17:40

1 As identified in other answers, there is no way to do this using CSS 2. Given that constraint, the answer I provided is one I know is effective and is the simplest way to answer the question using javascript imho. – David Clarke Dec 19 '15 at 19:27

Simplest way to answer the question using javascript info. – [David Clarke](#) Dec 19 '15 at 19:27

- 1 That approach would make SO almost completely useless. I search SO for how to solve problems, not to find the "only answer" doesn't address the issue. That is why SO is so awesome, because there is always more than one way to skin a cat. – [David Clarke](#) Dec 20 '15 at 18:35

▲ This is the most discussed aspect of the **Selectors Level 4** specification. With this, a selector will be able to style an element according to its child by using an exclamation mark after the given selector (!).

21

▼ For example:

```
body! a:hover{  
  background: red;  
}
```

will set a red background-color if the user hovers over any anchor.

But we have to wait for browsers' implementation :(

edited Jul 24 at 22:33



[Peter Mortensen](#)

14.5k 19 89 118

answered May 5 '15 at 18:24



[Marco Allori](#)

2,542 22 21

- 7 Sadly, but now this feature is out of specification as I know... – [Qwertiy](#) Nov 20 '15 at 12:06

▲ You might try to use hyperlink as the parent, and then change the inner elements on hover. Like this:

20

```
a.active h1 {color:red;}  
a.active:hover h1 {color:green;}  
a.active h2 {color:blue;}  
a.active:hover h1 {color:yellow;}
```

▼ This way you can change the style in multiple inner tags, based on the rollover of the parent element.

edited Feb 21 '12 at 20:07

answered Feb 13 '12 at 11:22



BoltClock ♦

riverstorm

556k 134 1216  
1238

482 3 7

- 
- 1 That is correct, but limits the markup code within the `a` tag to certain elements only, if you want to conform to XHTML standards. For instance, you cannot use a `div` within a `a`, without getting a warning of violating the schema. – [Ivaylo Slavov](#) Jul 24 '12 at 18:22 ✎
- 
- 2 Totally right Ivaylo! "a" is a non-block element, so can't use block elements inside it. – [riverstorm](#) Dec 12 '12 at 22:34
- 
- 1 In HTML5 it is perfectly fine to put block elements inside links. – [Matthew James Taylor](#) Apr 6 '14 at 7:47
- 
- 2 ... if it was semantically wrong, they wouldn't have allowed it in HTML5 where it wasn't before. – [BoltClock](#) ♦ Dec 28 '15 at 15:54
- 



The pseudo element `:focus-within` allows a parent to be selected if a descendent has focus.

19

An element can be focused if it has a `tabindex` attribute.



[Browser support for focus-within](#)

[Tabindex](#)

### Example

```
.click {  
  cursor: pointer;  
}  
  
.color:focus-within .change {  
  color: red;  
}  
  
.color:focus-within p {  
  outline: 0;  
}  
  
<div class="color">  
  <p class="change" tabindex="0">  
    I will change color  
  </p>  
</div>
```

```
<p class="click" tabindex="1">
  Click me

</p>
</div>
```

Run code snippet

[Expand snippet](#)

edited Mar 20 '18 at 13:36



roberrrt-s

6,192 30 46

answered Sep 25 '17 at 13:55



sol

16.2k 4 21 35

- 2 This works fine in my case, thanks! Just a not, the example would be more illustrative if you change the color to the parent, not to the sibling, this is replace `.color:focus-within .change` with `.color:focus-within`. In my case i'm using bootstrap nav-bar that add the class to the children when active and I want to add a class to the parent `.nav-bar`. I think this is a pretty common scenario where I own the markup but the component `css+js` is bootstrap (or other) so I cannot change the behavior. Although I can add `tabindex` and use this css. Thanks! – [cancerbero](#) Jun 29 '18 at 15:13



14



Currently there is no parent selector & it is not even being discussed in any of the talks of W3C. You need to understand how CSS is evaluated by the browser to actually understand if we need it or not.

There is a lot of technical explanation here.

[Jonathan Snook explains how CSS is evaluated.](#)

[Chris Coyier on the talks of Parent selector.](#)

[Harry Roberts again on writing efficient CSS selectors.](#)

But [Nicole Sullivan has some interesting facts on positive trends.](#)

These people are all top class in the field of front end development.

edited Nov 2 '17 at 0:38



Nathan Tuggy

2,215 9 26 35

answered Mar 21 '14 at 12:58



Suraj Naik

503 4 7

- 11 The need is defined by web developers' requirements, whether to have it in the spec is decided by other factors. – [nicodemus13](#) May 15 '14 at 9:04

Just an idea for horizontal menu...

14

## Part of HTML

```
<div class='list'>
  <div class='item'>
    <a>Link</a>
  </div>
  <div class='parent-background'></div>
  <!-- submenu takes this place -->
</div>
```

## Part of CSS

```
/* Hide parent backgrounds... */
.parent-background {
  display: none; }

/* ... and show it when hover on children */
.item:hover + .parent-background {
  display: block;
  position: absolute;
  z-index: 10;
  top: 0;
  width: 100%; }
```

[Updated demo and the rest of code](#)

[Another example](#) how to use it with text-inputs - select parent fieldset

edited Jul 24 at 22:37



Peter Mortensen

14.5k 19 89 118


answered Oct 27 '15 at 17:55



Ilya B.

872 7 13

- 3 It's not at all clear to me how you mean to generalize this to some/many/most of the parent selector use cases, or even exactly which parts of this CSS are doing what. Can you add a thorough explanation? – [Nathan Tuggy](#) Oct 28 '15 at 7:57
- 2 I did not try to apply this to real world scenarios, that is why I say "Not for production". But I think It can be applied to 2-level menu only with fixed item width. "which parts of this CSS are doing what" - .test-sibling here is actually background of parent item (the last line of CSS). – [Ilya B.](#) Oct 28 '15 at 8:08

- 2 Added explanation (css section of jsfiddle, starting from "MAIN PART")... And I was mistaken - there may be any number of sublevels. – [Ilya B.](#) Oct 28 '15 at 12:59 

There's a plugin that extends CSS to include some non-standard features that can really help when designing websites. It's called [EQCSS](#).

12

One of the things EQCSS adds is a parent selector. It works in all browsers, Internet Explorer 8 and up. Here's the format:

```
@element 'a.active' {  
  $parent {  
    background: red;  
  }  
}
```

So here we've opened an element query on every element `a.active`, and for the styles inside that query, things like `$parent` make sense, because there's a reference point. The browser can find the parent, because it's very similar to `parentNode` in JavaScript.

[Here's a demo of `\$parent`](#) and [another `\$parent` demo that works in Internet Explorer 8](#), as well as [a screenshot in case you don't have Internet Explorer 8 around to test with](#).

EQCSS also includes [meta-selectors](#): `$prev` for the element before a selected element and `$this` for only those elements that match an element query, and more.

edited Jul 24 at 22:43



[Peter Mortensen](#)

14.5k 19 89 118

answered Apr 7 '16 at 17:52



[innovati](#)

456 6 11

Technically there is no direct way to do this. However, you can sort that out with either jQuery or JavaScript.

11

However, you can do something like this as well.

```
a.active h1 {color: blue;}  
a.active p {color: green;}
```

**jQuery**

```
$("#a.active").parents('li').css("property", "value");
```

If you want to achieve this using jQuery here is the reference for the [jQuery parent selector](#).

edited Jul 24 at 22:36



Peter Mortensen

14.5k 19 89 118

answered Oct 14 '15 at 10:08



Prabhakar Undurthi

4,717 33 39

9

The W3C excluded such a selector because of the huge performance impact it would have on a browser.

answered Aug 16 '11 at 4:43



rgb

658 1 7 12

26 false. because the DOM is a tree, they have to go to the parent before getting to the child, so the simply just go back one node. o.o – [NullVoxPopuli](#) Nov 10 '11 at 16:56

9 CSS selectors are a [queue](#) so [selector order](#) is evaluated rather than the document XPath or DOM hierarchy. – [Paul Sweatte](#) Aug 18 '12 at 16:14 ✎

3 @rgb At least that's what they told us. – [yunzen](#) Apr 16 '13 at 15:35

8

The short answer is **NO**; we don't have a `parent selector` at this stage in CSS, but if you don't have to swap the elements or classes anyway, the second option is using JavaScript. Something like this:

```
var activeATag = Array.prototype.slice.call(document.querySelectorAll('a.active'));

activeATag.map(function(x) {
  if(x.parentNode.tagName === 'LI') {
    x.parentNode.style.color = 'red'; // Your property: value;
  }
});
```

Or a shorter way if you use **jQuery** in your application:

```
$('#a.active').parents('li').css('color', 'red'); // Your property: value;
```

answered Jun 16 '17 at 18:06



8

14/25

```

    }

    </style>

  </head>
  <body>
    <div class="parent">
      <div class="selector"></div>
    </div>
  </body>
</html>

```

Run code snippet

[Expand snippet](#)

edited Jul 26 at 1:59

answered Jul 31 '18 at 22:47



Jan Kyu Peblík

638 7 11

It's now 2019, and the [latest draft of the CSS Nesting Module](#) actually has something like this. Introducing `@nest` at-rules.

7

### 3.2. The Nesting At-Rule: `@nest`

While direct nesting looks nice, it is somewhat fragile. Some valid nesting selectors, like `.foo &`, are disallowed, and editing the selector in certain ways can make the rule invalid unexpectedly. As well, some people find the nesting challenging to distinguish visually from the surrounding declarations.

To aid in all these issues, this specification defines the `@nest` rule, which imposes fewer restrictions on how to validly nest style rules. Its syntax is:

```
@nest = @nest <selector> { <declaration-list> }
```

The `@nest` rule functions identically to a style rule: it starts with a selector, and contains declarations that apply to the elements the selector matches. The only difference is that the selector used in a `@nest` rule must be nest-containing, which means it contains a nesting selector in it somewhere. A list of selectors is nest-containing if all of its individual complex selectors are nest-containing.

(Copy and pasted from the URL above).

Example of valid selectors under this specification:

```
foo {
```

```

.foo {
  color: red;
  @nest & > .bar {
    color: blue;
  }
}
/* Equivalent to:
.foo { color: red; }
.foo > .bar { color: blue; }
*/

.foo {
  color: red;
  @nest .parent & {
    color: blue;
  }
}
/* Equivalent to:
.foo { color: red; }
.parent .foo { color: blue; }
*/

.foo {
  color: red;
  @nest :not(&) {
    color: blue;
  }
}
/* Equivalent to:
.foo { color: red; }
:not(.foo) { color: blue; }
*/

```

edited Jul 24 at 22:54



Peter Mortensen

14.5k 19 89 118

answered Mar 19 at 14:14



roberrrt-s

6,192 30 46



Although there is no parent selector in standard CSS at present, I am working on a (personal) project called **axe** (ie. *Augmented CSS Selector Syntax / ACSSSS*) which, among its 7 new selectors, includes both:

5



1. an *immediate parent* selector `<` (which enables the opposite selection to `>` )
2. an *any ancestor selector* `^` (which enables the opposite selection to `[SPACE]` )

**axe** is presently in a relatively early BETA stage of development.



See a demo here:

<http://rounin.co.uk/projects/axe/axe2.html>

(compare the two lists on the left styled with standard selectors and the two lists on the right styled with axe selectors)

edited Feb 18 '17 at 17:19

answered Feb 18 '17 at 17:06



Rounin

14.1k 3 30 50

- 3 The project is in an early Beta stage at present. You can (at least currently) activate axe selectors in your CSS styles by including `<script src="https://rouninmedia.github.io/axe/axe.js"></script>` at the very end of your html document, just before `</body>` . – Rounin Jul 12 '17 at 16:40

At least up to and including CSS 3 you cannot select like that. But it can be done pretty easily nowadays in JavaScript, you just need to add a bit of vanilla JavaScript, notice that the code is pretty short.

4

```
cells = document.querySelectorAll('div');
[].forEach.call(cells, function (el) {
    //console.log(el.nodeName)
    if (el.hasChildNodes() && el.firstChild.nodeName=="A") {
        console.log(el)
    }
});
```

```
<div>Peter</div>
<div><a href="#">Jackson link</a></div>
<div>Philip</div>
<div><a href="#">Pullman link</a></div>
```

Run code snippet

[Expand snippet](#)

edited Jul 24 at 22:50



Peter Mortensen

14.5k 19 89 118

answered Feb 1 '18 at 14:18



Eduard Florinescu

6,256 23 86 146

▲ No, you cannot select the parent in CSS only.

3 But as you already seem to have an `.active` class, it would be easier to move that class to the `li` (instead of the `a`). That way you can access both the `li` and the `a` via CSS only.

edited Jul 24 at 22:40



Peter Mortensen

14.5k 19 89 118

answered Nov 27 '15 at 21:16



Gaurav Aggarwal

7,588 3 22 53

2 The problem is that a CMS is generating the markup – [jcuenod](#) Nov 30 '15 at 20:15

▲ Any ideas?

3

▼ CSS 4 will be fancy if it adds some *hooks* into *walking backwards*. Till then it is possible (though **not** advisable) to use `checkbox` and/or `radio` `input` s to *break* the usual way that things are connected, and through that also allow CSS to operate outside of its normal scope...

```
/* Hide things that may be latter shown */
.menu__checkbox_selection,
.menu__checkbox_style,
.menu__hidden {
  display: none;
  visibility: hidden;
  opacity: 0;
  filter: alpha(opacity=0); /* Old Microsoft opacity */
}

/* Base style for content and style menu */
.main__content {
  background-color: lightgray;
  color: black;
}

.menu__hidden {
  background-color: black;
```

```
color: lightgray;
/* Make list look not so _listy_ */

list-style: none;
padding-left: 5px;
}

.menu__option {
  box-sizing: content-box;
  display: block;
  position: static;
  z-index: auto;
}

/* &#9660; - \u2630 - Three Bars */
/*
.menu__trigger__selection::before {
  content: '\u2630';
  display: inline-block;
}
*/

/* &#9660; - Down Arrow */
.menu__trigger__selection::after {
  content: "\u25BC";
  display: inline-block;
  transform: rotate(90deg);
}

/* Customize to look more `select` like if you like */
.menu__trigger__style:hover,
.menu__trigger__style:active {
  cursor: pointer;
  background-color: darkgray;
  color: white;
}

/**
 * Things to do when checkboxes/radios are checked
 */

.menu__checkbox__selection:checked + .menu__trigger__selection::after,
.menu__checkbox__selection:checked + .menu__trigger__selection::after {
  transform: rotate(0deg);
}

/* This bit is something that you may see elsewhere */
.menu__checkbox__selection:checked ~ .menu__hidden,
```

```
.menu__checkbox__selection[checked] ~ .menu__hidden {
  display: block;
  visibility: visible;
  opacity: 1;
  filter: alpha(opacity=100); /* Microsoft!?! */
}

/**
 * Hacky CSS only changes based off non-inline checkboxes
 * ... AKA the stuff you cannot unsee after this...
 */
.menu__checkbox__style[id="style-default"]:checked ~ .main__content {
  background-color: lightgray;
  color: black;
}

.menu__checkbox__style[id="style-default"]:checked ~ .main__content
.menu__trigger__style[for="style-default"] {
  color: darkorange;
}

.menu__checkbox__style[id="style-one"]:checked ~ .main__content {
  background-color: black;
  color: lightgray;
}

.menu__checkbox__style[id="style-one"]:checked ~ .main__content
.menu__trigger__style[for="style-one"] {
  color: darkorange;
}

.menu__checkbox__style[id="style-two"]:checked ~ .main__content {
  background-color: darkgreen;
  color: red;
}

.menu__checkbox__style[id="style-two"]:checked ~ .main__content
.menu__trigger__style[for="style-two"] {
  color: darkorange;
}

<!--
  This bit works, but will one day cause troubles,
  but truth is you can stick checkbox/radio inputs
  just about anywhere and then call them by id with
  a `for` label. Keep scrolling to see what I mean
-->
```

```
-->
<input type="radio"

    name="colorize"
    class="menu__checkbox__style"
    id="style-default">
<input type="radio"
    name="colorize"
    class="menu__checkbox__style"
    id="style-one">
<input type="radio"
    name="colorize"
    class="menu__checkbox__style"
    id="style-two">

<div class="main__content">

    <p class="paragraph__split">
        Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod
        tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
        quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
    </p>

    <input type="checkbox"
        class="menu__checkbox__selection"
        id="trigger-style-menu">
    <label for="trigger-style-menu"
        class="menu__trigger__selection"> Theme</label>

    <ul class="menu__hidden">
        <li class="menu__option">
            <label for="style-default"
                class="menu__trigger__style">Default Style</label>
        </li>

        <li class="menu__option">
            <label for="style-one"
                class="menu__trigger__style">First Alternative Style</label>
        </li>

        <li class="menu__option">
            <label for="style-two"
                class="menu__trigger__style">Second Alternative Style</label>
        </li>
    </ul>

    <p class="paragraph__split">
        consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse
        cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non
```

```

    proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
  </p>

</div>

```

[Run code snippet](#)
[Expand snippet](#)

... pretty *gross*, but with just CSS and HTML it is possible to touch and re-touch anything but the `body` and `:root` from just about anywhere by linking the `id` and `for` properties of `radio / checkbox input`s and `label triggers`; likely someone'll show how to re-touch those at some point.

One additional caveat is that only **one** `input` of a specific `id` maybe used, first `checkbox / radio` *wins* a toggled state in other words... **But** multiple labels can all point to the same `input`, though that would make both the HTML and CSS look even grosser.

... I'm hoping that there is some sort of workaround that exists native to CSS Level 2...

I am not sure about the other `:` selectors, but I `:checked` for pre-CSS 3. If I remember correctly, it was something like `[checked]` which is why you may find it in the above code, for example,

```

.menu__checkbox__selection:checked ~ .menu__hidden,
.menu__checkbox__selection[checked] ~ .menu__hidden {
  /* rules: and-stuff; */
}

```

... but for things like `::after` and `:hover`, I'm not at all certain in which CSS version those first appeared.

That all stated, please don't ever use this in production, not even in anger. As a joke sure, or in other words just because something *can* be done does not always mean it *should*.

edited Jul 24 at 23:02



Peter Mortensen

14.5k 19 89 118

answered Jun 8 at 6:31



S0AndS0

510 1 4 9

It's possible with ampersand in [Sass](#):

1

```
h3
  font-size: 20px
  margin-bottom: 10px
.some-parent-selector &
  font-size: 24px
  margin-bottom: 20px
```

CSS output:

```
h3 {
  font-size: 20px;
  margin-bottom: 10px;
}
.some-parent-selector h3 {
  font-size: 24px;
  margin-bottom: 20px;
}
```

edited Jul 24 at 22:47



Peter Mortensen

14.5k 19 89 118

answered Aug 3 '17 at 18:19



Rodolfo Jorge Nemer  
Nogueira

3,554 2 26 25

2 it's true, but only if you know the selector in advance. – [Shahar](#) Aug 9 '17 at 8:02

8 This does **not** select `h3` 's parent, which was the goal. This would select `h3` s which are descendants of `.some-parent-selector` . – [Jacob Ford](#) Mar 25 at 12:41

Changing parent element based on child element can currently only happen when we have an `<input>` element inside the parent element. When an input gets focus, its corresponding parent element can get affected using CSS.

0

Following example will help you understand using `:focus-within` in CSS.

```
.outer-div {
  width: 400px;
  height: 400px;
  ...
}
```

```
padding: 50px;
float: left;
}

.outer-div:focus-within {
  background: red;
}

.inner-div {
  width: 200px;
  height: 200px;
  float: left;
  background: yellow;
  padding: 50px;
}
```

```
<div class="outer-div">
  <div class="inner-div">
```

I want to change outer-div(Background color) class based on inner-div. Is it possible?

```
<input type="text" placeholder="Name" />
</div>
</div>
```

[Run code snippet](#)
[Expand snippet](#)

edited Jul 7 at 0:56


**VFDan**  
 630 3 19

answered Jul 5 at 6:34


**Sethuraman**  
 421 2 9


0



I'd hire some JavaScript code to do that. For example, in React when you iterate over an array, add another class to the parent component, which indicates it contains your children:

```
<div className={`parent ${hasKiddos ? 'has-kiddos' : ''}`}>
  {hasKiddos && kiddos.map(kid => (
    <div key={kid.id} className="kid">kid</div>
  ))}
</div>
```



And then simply:

```
.parent {  
  color: #000;  
}  
  
.parent.has-kiddos {  
  color: red;  
}
```

edited Jul 24 at 23:04



Peter Mortensen

14.5k 19 89 118

answered Jun 13 at 8:25



Damiano

755 2 9 22



In CSS, we can cascade to the properties down the hierarchy but not in the opposite direction. To modify the parent style on child event, probably use jQuery.

-1



```
$el.closest('.parent').css('prop', 'value');
```

answered Sep 15 '16 at 4:06



Sukhminder Parmar

127 1 4

1

2

next

protected by [zzzzBov](#) Sep 5 '13 at 0:39

Thank you for your interest in this question. Because it has attracted low-quality or spam answers that had to be removed, posting an answer now requires 10 [reputation](#) on this site (the [association bonus does not count](#)).

Would you like to answer one of these [unanswered questions](#) instead?