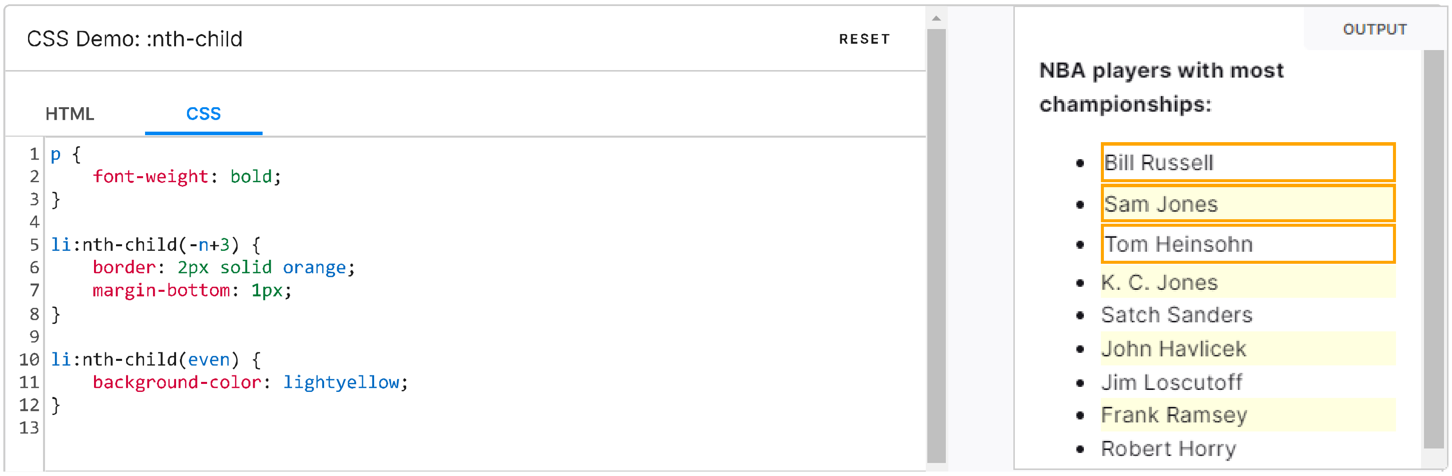
mdn web docs

:nth-child()

The `:nth-child()` CSS pseudo-class matches elements based on their position among a group of siblings.

Try it



```
p {
    font-weight: bold;
}

li:nth-child(-n+3) {
    border: 2px solid orange;
    margin-bottom: 1px;
}

li:nth-child(even) {
    background-color: lightyellow;
}

```

NBA players with most championships:

- Bill Russell
- Sam Jones
- Tom Heinsohn
- K. C. Jones
- Satch Sanders
- John Havlicek
- Jim Loscutoff
- Frank Ramsey
- Robert Horry

Note that, in the `element:nth-child()` syntax, the child count includes children of any element type; but it is considered a match only if the element *at that child position* is of the specified element type.

Syntax

`:nth-child()` takes a single argument that describes a pattern for matching element indices in a list of siblings. Element indices are 1-based.

```
:nth-child( <nth> [ of <complex-selector-list> ]? )
```

Keyword values

`odd`

Represents elements whose numeric position in a series of siblings is odd: 1, 3, 5, etc.

`even`

Represents elements whose numeric position in a series of siblings is even: 2, 4, 6, etc.

Functional notation

`<An+B>`

Represents elements in a list whose indices match those found in a custom pattern of numbers, defined by `An+B`, where:

- `A` is an integer step size,
- `B` is an integer offset,
- `n` is all nonnegative integers, starting from 0.

It can be read as the `An+B`-th element of a list.

Examples

Example selectors

`tr:nth-child(odd)` or `tr:nth-child(2n+1)`

Represents the odd rows of an HTML table: 1, 3, 5, etc.

`tr:nth-child(even)` or `tr:nth-child(2n)`

Represents the even rows of an HTML table: 2, 4, 6, etc.

`:nth-child(7)`

Represents the seventh element.

`:nth-child(5n)`

Represents elements **5** [=5×1], **10** [=5×2], **15** [=5×3], **etc.** The first one to be returned as a result of the formula is **0** [=5x0], resulting in a no-match, since the elements are indexed from 1, whereas `n` starts from 0. This may seem weird at first, but it makes more sense when the `B` part of the formula is `>0`, like in the next example.

`:nth-child(n+7)`

Represents the seventh and all following elements: **7** [=0+7], **8** [=1+7], **9** [=2+7], **etc.**

`:nth-child(3n+4)`

Represents elements **4** [=(3×0)+4], **7** [=(3×1)+4], **10** [=(3×2)+4], **13** [=(3×3)+4], **etc.**

`:nth-child(-n+3)`

Represents the first three elements. [=-0+3, -1+3, -2+3]

`p:nth-child(n)`

Represents every `<p>` element in a group of siblings. This selects the same elements as a simple `p` selector (although with a higher specificity).

`p:nth-child(1)` or `p:nth-child(0n+1)`

Represents every `<p>` that is the first element in a group of siblings. This is the same as the `:first-child` selector (and has the same specificity).

`p:nth-child(n+8):nth-child(-n+15)`

Represents the eighth through the fifteenth `<p>` elements of a group of siblings.

Detailed example

HTML

```html
<h3><code>span:nth-child(2n+1)</code>, WITHOUT an
   <code>&lt;em&gt;</code> among the child elements.</h3>
<p>Children 1, 3, 5, and 7 are selected.</p>
<div class="first">
  <span>Span 1!</span>
  <span>Span 2</span>
  <span>Span 3!</span>
  <span>Span 4</span>
  <span>Span 5!</span>
  <span>Span 6</span>
  <span>Span 7!</span>
</div>

<br>

<h3><code>span:nth-child(2n+1)</code>, WITH an
   <code>&lt;em&gt;</code> among the child elements.</h3>
<p>Children 1, 5, and 7 are selected.<br>
   3 is used in the counting because it is a child, but it isn't
   selected because it isn't a <code>&lt;span&gt;</code>.</p>
<div class="second">
  <span>Span!</span>
  <span>Span</span>
  <em>This is an `em`.</em>
  <span>Span</span>
  <span>Span!</span>
  <span>Span</span>
  <span>Span!</span>
  <span>Span</span>
</div>

<br>

<h3><code>span:nth-of-type(2n+1)</code>, WITH an
   <code>&lt;em&gt;</code> among the child elements.</h3>
<p>Children 1, 4, 6, and 8 are selected.<br>
   3 isn't used in the counting or selected because it is an <code>&lt;em&gt;</code>,
   not a <code>&lt;span&gt;</code>, and <code>nth-of-type</code> only selects
   children of that type. The <code>&lt;em&gt;</code> is completely skipped
   over and ignored.</p>
<div class="third">
  <span>Span!</span>
  <span>Span</span>
  <em>This is an `em`.</em>
  <span>Span!</span>
  <span>Span</span>
  <span>Span!</span>
  <span>Span</span>
  <span>Span!</span>
</div>
```

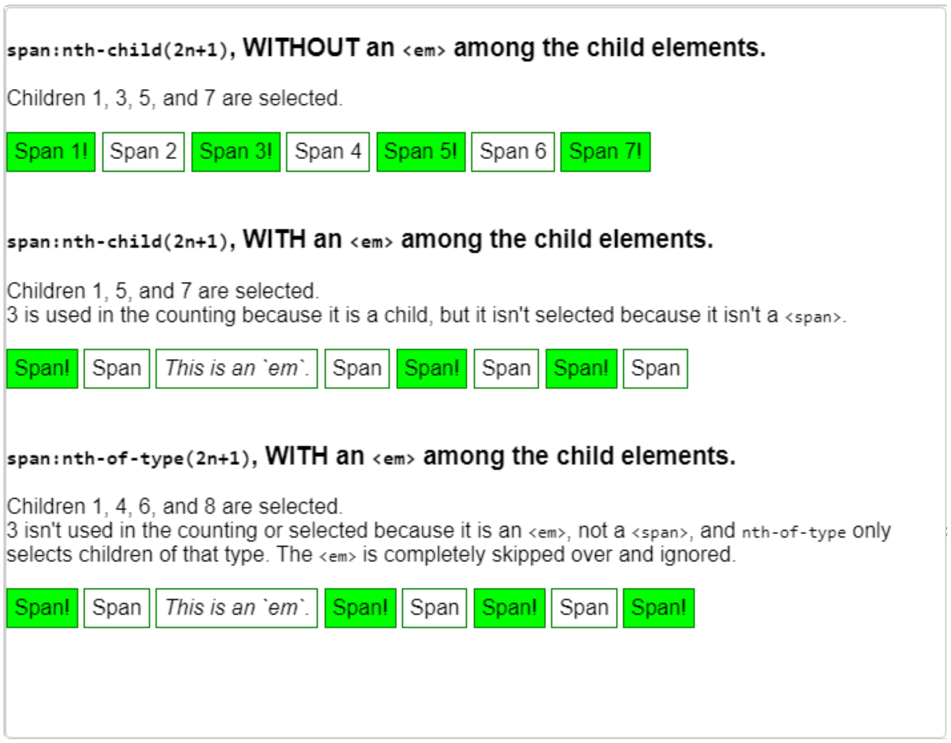
CSS

```
html {
  font-family: sans-serif;
}

span,
div em {
  padding: 5px;
  border: 1px solid green;
  display: inline-block;
  margin-bottom: 3px;
}

.first span:nth-child(2n+1),
.second span:nth-child(2n+1),
.third span:nth-of-type(2n+1) {
  background-color: lime;
}
```

Result



`span:nth-child(2n+1)`, WITHOUT an `<em>` among the child elements.

Children 1, 3, 5, and 7 are selected.

Span 1! Span 2 Span 3! Span 4 Span 5! Span 6 Span 7!

`span:nth-child(2n+1)`, WITH an `<em>` among the child elements.

Children 1, 5, and 7 are selected.
3 is used in the counting because it is a child, but it isn't selected because it isn't a `<span>`.

Span! Span *This is an `em`.* Span Span! Span Span! Span

`span:nth-of-type(2n+1)`, WITH an `<em>` among the child elements.

Children 1, 4, 6, and 8 are selected.
3 isn't used in the counting or selected because it is an `<em>`, not a `<span>`, and `nth-of-type` only selects children of that type. The `<em>` is completely skipped over and ignored.

Span! Span *This is an `em`.* Span! Span Span! Span Span!

Specifications

Specification
Selectors Level 4 # nth-child-pseudo

Browser compatibility

Report problems with this compatibility data on GitHub

	Chrome	Edge	Firefox	Opera	Safari	Chrome Android	Firefox for Android
`:nth-child()`	✓ Chrome 1	✓ Edge 12	✓ Firefox 3.5	✓ Opera 9.5 ✱	✓ Safari 3.1	✓ Chrome Android 18	✓ Firefox for Android 4
Matches elements with no parent	✓ Chrome 57	✓ Edge 79	✓ Firefox 52	✓ Opera 44	✗ Safari No	✓ Chrome Android 57	✓ Firefox for Android 52
`of <selector>` syntax	✗ Chrome No ✱	✗ Edge No ✱	✗ Firefox No ✱	✗ Opera No ✱	✓ Safari 9	✗ Chrome Android No ✱	✗ Firefox for Android No ✱

Tip: you can click/tap on a cell for more information.

✓ Full support ✗ No support ✱ See implementation notes.

See also

- `:nth-of-type`, `:nth-last-child`

Last modified: Sep 5, 2022, by MDN contributors