



is() pseudo class

There are times when we want the same element across the page to do the same.

The following block styles the h2 elements inside the header, footer and main elements.

```
header h2,  
main h2,  
footer h2 {  
  color: red;  
}
```

It works but it's error prone, repetitive and may cause all rules to be ignored if there is a mistake.

Published in [Selectors Level 4](#), the `:is()` [pseudo-class](#) takes a selector list, and selects any element that matches any of the selectors in that list. Using `:is()` we can write the previous example like this

```
:is(header, main, footer) h2 {  
  color: red;  
}
```

Older browsers support this functionality as `:matches()`, or through an older, prefixed pseudo-class — `:any()`, including older versions of Chrome, Firefox, and Safari. `:any()` works in exactly the same way as `:matches()/:is()`, except that it requires vendor prefixes and doesn't support complex selectors.

The specificity of the `:is()` pseudo class is the highest specificity of the `:is()` arguments plus the specificity of the remaining arguments in the selector

For example the specificity for all h2 elements that match the following CSS rule:

```
:is(#header main footer) h2 {
```

```
font-weight: 700;
}
```

is **1-0-1**. They all take the highest specificity of the attributes for the `:is()` pseudo class plus any remaining attributes for the selector. The highest specificity for the `:is` attributes is the one for `#header` (shown as 1-0-0) and the specificity for `h2` is 0-0-1). This behavior is different than older selectors like `:any()` or `:matches()` where the specificity was calculated differently

For a basic primer on specificity see Estelle Weyl's [CSS Specificity](#) and the [CSS Specificity Chart](#)

Because `:is` is not implemented in all browsers, some browsers require vendor prefixes and some browsers use earlier version of the pseudo class like `:matches`.

```
/* basic syntax without pseudo classes */
header h2,
main h2,
footer h2 {
  font-weight: 700;
  color: rebeccapurple;
}

/* older webkit */
:-webkit-any(header, main, footer) h2 {
  font-weight: 700;
  color: rebeccapurple;
}

/* older Firefox syntax */
:-moz-any(header, main, footer) h2 {
  font-weight: 700;
  color: rebeccapurple;
}

/* Older syntax using :matches() */
:matches(header, main, footer) h2 {
```

```
font-weight: 700;  
color: rebeccapurple;  
}
```

:is() and :where() accept a [forgiving selector list](#).

In CSS when using a selector list, if any of the selectors are invalid then the whole list is deemed invalid. When using :is() or :where() instead of the whole list of selectors being deemed invalid if one fails to parse, the incorrect or unsupported selector will be ignored and the others used.

```
:is(#header, :unsupported) h2{  
font-weight: 700;  
color: rebeccapurple;  
}
```

Will still parse correctly and match supported values, even in browsers which don't support :unsupported, whereas:

```
#header h2,  
:unsupported h2{  
font-weight: 700;  
color: rebeccapurple;  
}
```

Will be completely ignored in browsers that don't support :unsupported even if they support the other selectors.