Lean Code and Advanced CSS selectors

Design for web content

Lean Code

What are the advantages of lean code?

- More manageable – easier to work with/develop
- More maintainable – updates are easier
- More considerate to others in your team – easier to understand
- Better user experience – faster loading pages
- More accessible – easier for assistive technologies to interpret
- More sustainable – uses less energy
- ~~It will make you more attractive~~

Checking and keeping your code in good order
2 key processes:

Validation

Checking against known standards

Optimisation

Self-checking for redundancy and format

```
<aside>
  <div class="sidebar">
    <nav>
      <div class="main-navigation">
        <ul class="navigation-list">
          <li><a href="#">Item 1</a></li>
          <li><a href="#">Item 2</a></li>
          <li><a href="#">Item 3</a></li>
        </ul>
      </div>
    </nav>
  </div>
</aside>
```

Note: This is valid code, but not optimal

```
<div class="navigation">
    <ul class="navigation-list">
        <li class="link"><a href="#">Item 1</a></li>
        <li class="link"><a href="#">Item 2</a></li>
        <li class="link"><a href="#">Item 3</a></li>
        <li class="link"><a href="#">Item 4</a></li>
        <li class="link"><a href="#">Item 5</a></li>
    </ul>
</div>
```

Note: This is valid code, but not optimal

```
<!-- start global navigation -->
<nav>
    <ul>
        <li><a href="#">Item 1</a></li>
        <li><a href="#">Item 2</a></li>
        <li><a href="#">Item 3</a></li>
        <li><a href="#">Item 4</a></li>
        <li><a href="#">Item 5</a></li>
    </ul>
</nav>
<!-- end global navigation -->
```

Note: This is valid code

There is no need for classes or ids because all elements can be selected with reference to semantic elements using descendent selectors (`nav ul li`).

```
<!-- start site navigation -->
<nav aria-label="primary">
    <ul>
        <li><a href="#">Item 1</a></li>
        <li><a href="#">Item 2</a></li>
        <li><a href="#">Item 3</a></li>
        <li><a href="#">Item 4</a></li>
        <li><a href="#">Item 5</a></li>
    </ul>
</nav>
<!-- end site navigation -->
```

If your nav element contains the main website navigation, add the appropriate ARIA label.

Selecting all elements

```
nav a {
  color: #c00;
}
```

All anchor elements with nav as an ancestor (parent, grandparent etc.)

https://developer.mozilla.org/en-US/docs/Web/CSS/Descendant_combinator

Selecting the second anchor

```
nav a:nth-of-type(2) {
  color: #c00;
}
```

The second anchor element with nav as an ancestor

https://developer.mozilla.org/en-US/docs/Web/CSS/Descendant_combinator

```
<!-- start site navigation -->
<nav aria-label="primary">
    <ul>
        <li><a href="liquorice.html">Liquorice</a></li>
        <li><a href="toffee.html">Toffee</a></li>
        <li><a href="fudge.html">Fudge</a></li>
        <li><a href="nougat.html">Nougat</a></li>
        <li><a href="marzipan.html">Marzipan</a></li>
    </ul>
</nav>
<!-- end site navigation -->
```

How might we select only the anchor that links to toffee.html? Surely we need a class?

Selecting anchor with attribute

```
a[href="toffee.html"] {
    color: #c00;
}
```

The anchor element with an *href* attribute **and** a value of *toffee.html*

https://developer.mozilla.org/en-US/docs/Web/CSS/Attribute_selectors

```
<!-- start site navigation -->
<nav aria-label="primary">
    <ul>
        <li><a href="liquorice.html">Liquorice</a></li>
        <li><a href="toffee.html">Toffee</a></li>
        <li><a href="fudge.html">Fudge</a></li>
        <li><a href="nougat.html">Nougat</a></li>
        <li><a href="marzipan.html">Marzipan</a></li>
    </ul>
</nav>
<!-- end site navigation -->
```

How might we select this nav element if there are several in the document? Do we need a class?

Selecting nav with attribute

```
nav[aria-label="primary"] {
   color: #c00;
}
```

For accessibility, it's good practice to add an aria label to nav elements to help non-sighted users interpret their function.

The anchor element with an *aria-label* attribute **and** a value of *primary*

https://developer.mozilla.org/en-US/docs/Web/CSS/Attribute_selectors

To help us keep our code lean there are many advanced CSS selector types that allow us to target elements without the need for classes. They fall into 2 main categories:

Combinator

Defined by relationship with sibling elements

Structural pseudo classes

Defined by their location within a parent element

Combinator Selectors

Descendent combinator

```
main p {
   color: #c00;
}
```

All paragraphs that are descendants (children, grandchildren…) of main

Child combinator

```
main > p {
  color: #c00;
}
```

All paragraphs that are *direct* children of main (i.e. not grandchildren)

https://developer.mozilla.org/en-US/docs/Web/CSS/Child_combinator

Next-sibling combinator

```
h2 + p {
  color: #c00;
}
```

All paragraphs that *immediately* follow an h2 element

https://developer.mozilla.org/en-US/docs/Web/CSS/Adjacent_sibling_combinator

Subsequent-sibling combinator

```
h2 ~ p {
  color: #c00;
}
```

All paragraphs that follow h2 within the same parent (not immediately)

https://developer.mozilla.org/en-US/docs/Web/CSS/General_sibling_combinator

Child and next-sibling combinators

```
main > h2 + p {
  color: #c00;
}
```

All paragraphs that are *direct* children of main *and* immediately follow h2

Structural Pseudo Classes

User action pseudo-classes

```
a:hover {
   color: #c00;
}
```

Some pseudo-classes will already be known to you

https://developer.mozilla.org/en-US/docs/Web/CSS/:hover

Tree-structural pseudo-classes

```
p:first-child {
   color: #c00;
}
```

All paragraphs that are first children

https://developer.mozilla.org/en-US/docs/Web/CSS/:first-child

Tree-structural pseudo-classes

```
p:last-child {
  color: #c00;
}
```

All paragraphs that are last children

https://developer.mozilla.org/en-US/docs/Web/CSS/:last-child

Tree-structural pseudo-classes

```
p:nth-child(3) {
  color: #c00;
}
```

All paragraphs that are third children

https://developer.mozilla.org/en-US/docs/Web/CSS/:nth-child

Tree-structural pseudo-classes

```
p:nth-child(odd) {
  color: #c00;
}
```

All paragraphs that are odd children (1, 3, 5, 7 etc.)

Tree-structural pseudo-classes

```
p:nth-child(even) {
  color: #c00;
}
```

All paragraphs that are even children (2, 4, 6, 8 etc.)

Tree-structural pseudo-classes

```
p:nth-of-type(3) {
  color: #c00;
}
```

The third paragraph (not necessarily the third child). This is often a more logical/useful selector for general use.

https://developer.mozilla.org/en-US/docs/Web/CSS/:nth-of-type

Tree-structural pseudo-classes

```
p:nth-of-type(odd) {
  color: #c00;
}
```

All odd numbered paragraphs within any container.
You can also use `nth-of-type(even)`.

Tree-structural pseudo-classes

```
p:first-of-type {
  color: #c00;
}
```

The first paragraph within any container.
You can also use `last-of-type`.

Tree-structural pseudo-classes

```
main p:last-of-type {
  color: #c00;
}
```

The last paragraph within `<main>`. In this case, we are using a descendent combinator in conjunction with a structural pseudo class.

Attribute Selectors

Attribute selectors

```
[href="index.html"] {
  color: #c00;
}
```

Any element having an *href* attribute **and** a value of *index.html*

https://developer.mozilla.org/en-US/docs/Web/CSS/Attribute_selectors

Attribute selectors

```
a[href^="https"] {
  color: #c00;
}
```

An <a> element where the *href* attribute value starts with the string “https”. This is a neat way of selecting all external links.

Negation pseudo-class selector

```
main :not(p) {
  color: #c00;
}
```

All elements in <main> except paragraphs who are direct children.

https://developer.mozilla.org/en-US/docs/Web/CSS/:not

Relational pseudo-class selector

```
:has(p) {
  color: #c00;
}
```

Not ready for production sites!

All elements containing paragraphs that are direct children.

https://developer.mozilla.org/en-US/docs/Web/CSS/:not

Relational pseudo-class selector

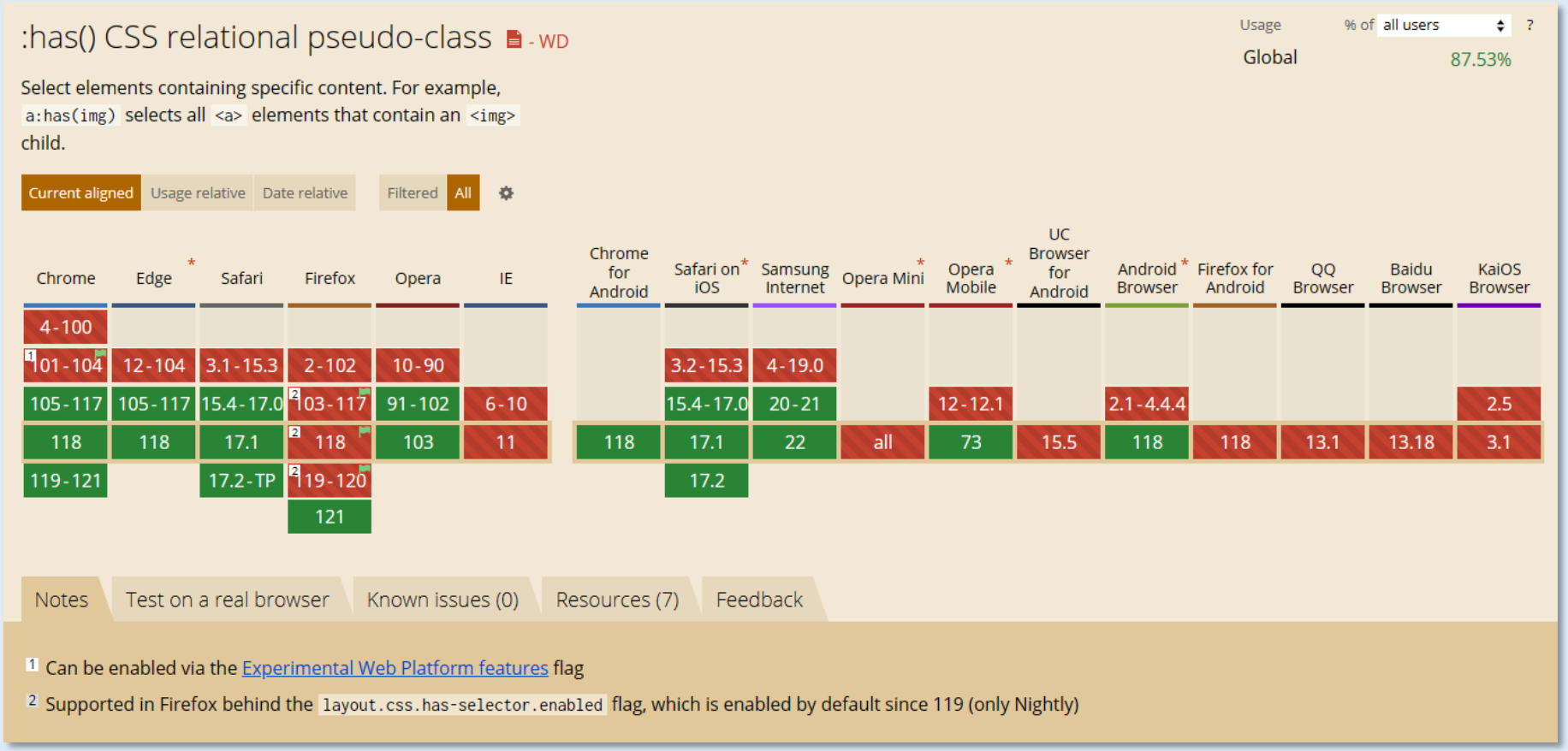
```
h1:has(+ h2) {
  color: #c00;
}
```

Not ready for production sites!

h1 elements that are immediately followed by h2.

https://developer.mozilla.org/en-US/docs/Web/CSS/:not

Browser support for :has



:has() CSS relational pseudo-class - WD

Select elements containing specific content. For example, `a:has(img)` selects all `<a>` elements that contain an `<img>` child.

Usage: % of all users — Global 87.53%

Current aligned | Usage relative | Date relative | Filtered | All

Chrome	Edge *	Safari	Firefox	Opera	IE
4-100					
101-104 [1]	12-104	3.1-15.3	2-102	10-90	
105-117	105-117	15.4-17.0	103-117 [2]	91-102	6-10
118	118	17.1	118 [2]	103	11
119-121		17.2-TP	119-120 [2]		
			121		

Chrome for Android	Safari on iOS *	Samsung Internet	Opera Mini *	Opera Mobile *	UC Browser for Android	Android Browser *	Firefox for Android	QQ Browser	Baidu Browser	KaiOS Browser
	3.2-15.3	4-19.0								
	15.4-17.0	20-21		12-12.1		2.1-4.4.4				2.5
118	17.1	22	all	73	15.5	118	118	13.1	13.18	3.1
	17.2									

Notes | Test on a real browser | Known issues (0) | Resources (7) | Feedback

[1] Can be enabled via the Experimental Web Platform features flag

[2] Supported in Firefox behind the `layout.css.has-selector.enabled` flag, which is enabled by default since 119 (only Nightly)

Chrome support = 30th August 2022. Firefox support pending.

https://caniuse.com/?search=%3Ahas

```
[class="end"] {
	color: #c00;
}
```

<slideshow class="end">