Lean Code and Advanced CSS selectors

Design for web content

Lean Code

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>Item 1</li>
          <li>Item 2</li>
          <li>Item 3</li>
        </ul>
      </div>
    </nav>
  </div>
</aside>
```

Note: This is valid code

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

Note: This is valid code

```
<!-- start global navigation -->
<nav>
    <ul>
        <li>Item 1</li>
        <li>Item 2</li>
        <li>Item 3</li>
        <li>Item 4</li>
        <li>Item 5</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`).

Descendent selector (combinator)

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

All list items with nav as a parent, grandparent etc.

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

General sibling combinator

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

All paragraphs that follow h2 within the same parent

Adjacent sibling combinator

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

All paragraphs that *immediately* follow h2

Descendent combinator

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

All paragraphs that are children of main

Child combinator

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

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

Child and adjacent combinators

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

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

Structural Pseudo Classes

Structural pseudo classes

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

All paragraphs that are first children

Structural pseudo classes

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

All paragraphs that are last children

Structural pseudo classes

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

All paragraphs that are third children

Structural pseudo classes

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

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

Structural pseudo classes

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

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

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.

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)`.

Structural pseudo classes

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

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

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*

Attribute selectors

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

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

<slideshow class="end" />