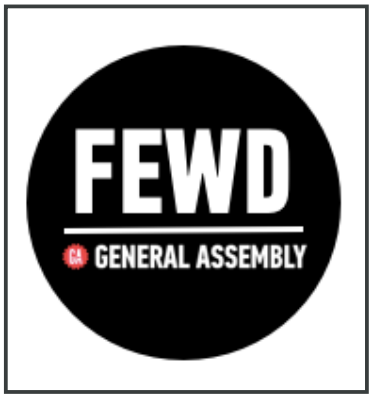


CSS LAYOUT

WITH GUY ROUTLEDGE

@GUYROUTLEDGE | #FEWD

AGENDA

- Review
- New CSS selectors
- Floats
- Lab Time

REVIEW

SELECTORS

So far we've seen the following type of CSS selectors. They match the HTML elements in your document but are quite limiting when dealing with complex projects.

```
p { } /* element selector */
h1, h2, h3  { } /* multiple elements */

header a { } /* descendent element selector */
```

SELECTORS

To have more control over the sections of the page to be styled, there are a whole load of other selectors available.

These can further be combined by comma separating or descendent selectors.

PSEUDO SELECTORS

Other selectors include *pseudo* selectors for styling state. A classic example is the **`hover`** state when mousing over links or the **`focus`** state when filling in a form.

```
a { color:red; }
a:hover { color:blue; }

input { background:#fff; }
input:focus { background:#eee; }
```

PSEUDO ELEMENTS

We also can style pseudo elements.

```
p:first-letter      { }
p:first-line        { }
p:first-child       { }
p:last-child        { }
p:nth-child(exp)    { }
p:first-of-type     { }
p:last-of-type      { }
p:nth-of-type(exp)  { }
```

`CLASS & ID`

By far the most targeted selectors to use are **`class`** or **`id`** selectors.

They allow us to target specific parts of a page regardless of the type of element.

CLASS & ID

class and **id** are **attributes** that are added to the HTML and then selected from the CSS to apply styling.

```
<div id="main-content" class="page-wrap">
    ...
</div>
```

IDS ARE UNIQUE

They can be used **ONCE** per page - best for JavaScript

CLASSES ARE NOT UNIQUE

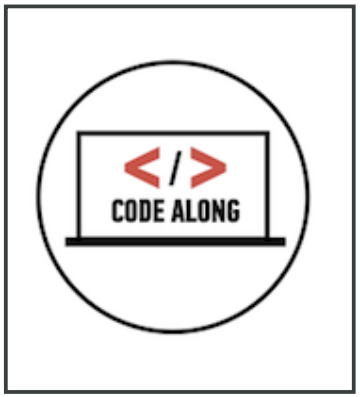
They can be reused **MANY** times per page - best for CSS

CLASS & ID

How to use them in CSS

```
.page-wrap {
    /* styles go here */
}

#main-content {
    /* styles go here */
}
```



CLASS & ID

http://codepen.io/guyroutledge/pen/zKqqKp

WHEN TO USE CLASS AND ID

Add `class` attributes to the HTML for targeting with CSS.

Add `id` attributes to the HTML for targeting with JS

Don't use `id` for adding styling to elements

SPECIFICITY

When writing CSS it's common for there to be conflicting properties applied to the same element.

SPECIFICITY

The styles that "win" and will be rendered in the browser are determined by 3 major criteria in this order:

- Importance
- Specificity
- Source Order

SPECIFICITY

Take the following example HTML:

```
<h1 id="main-title" class="title">some title</h1>
```

IMPORTANCE

CSS declarations with **`!important`** beat everything.

```
h1 { color: white !important; } /* wins */
#main-title { color: red; }
```

Avoid using this because it's a very heavy-handed.

SPECIFICITY

We calculate a selectors specificity by counting the number of inline styles, ids, classes and element selectors.

- `style` is more powerful than
- `id` which is more powerful than
- `class` which is more powerful than
- element selectors

This produces a 4 digit number called the specificity value.

SPECIFICITY

1 element and 1 id beats 3 elements

```
header #main-title { color: red; } /* 0101 wins */
header div h1 { color: white; } /* 0003 */
```

SOURCE ORDER

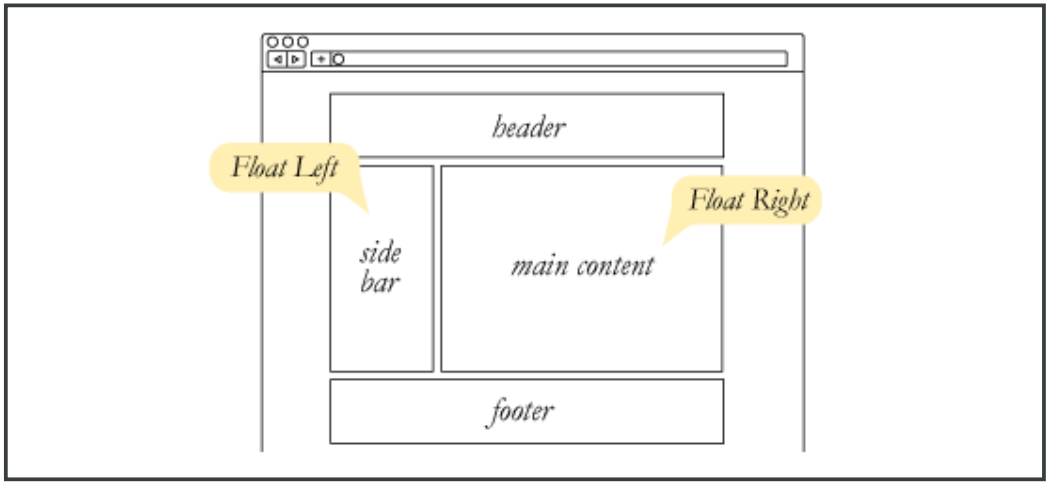
If two selectors have the same level of importance and specificity, the one that comes later will win.

```
header .title { color: black; }
header .title { color: red; } /* wins */
```

CSS LAYOUT

FLOATS

`float` is a property that can be used to have block elements "float" next to each other.



FLOATS

The `float` property was never really designed for layout but rather to allow text to wrap around an image.

This was a popular design style back in the day of print and web designers wanted to mimick the effect.

FLOATS

There are now a number of layout options available in CSS but `float` still remains a popular solution even though there are some weird side effects.

FLOATS

To enable two block elements to site next to each other, to create multi-column layouts, we need to float them.

This is because `block` elements normally stack vertically and `inline` elements don't respond to layout properties like `width` and `height`.

FLOATS

An element can be floated to the **`left`** or the **`right`** side of its container

```
.main-content { width: 500px; float:left; }
.sidebar { width: 300px; float:right; }
```

FLOATS

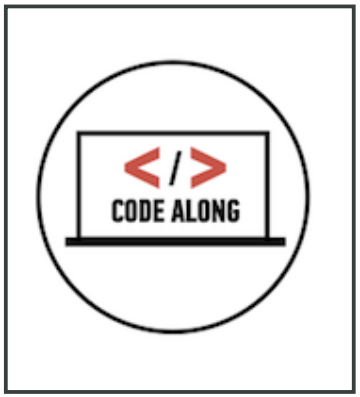
When elements float, surrounding elements try to flow around them which can cause some weird knock-on effects.

To get the layout back on track, we can **`clear`** the floats and bring everything back to normal.

```
.footer { clear:both }
```

FLOATS

We can clear to the `left`, `right`, or `both` sides. Which will clear the affect of elements floating to the left, right or both left and right.



FLOATING SECTIONS

FASHION BLOG 2