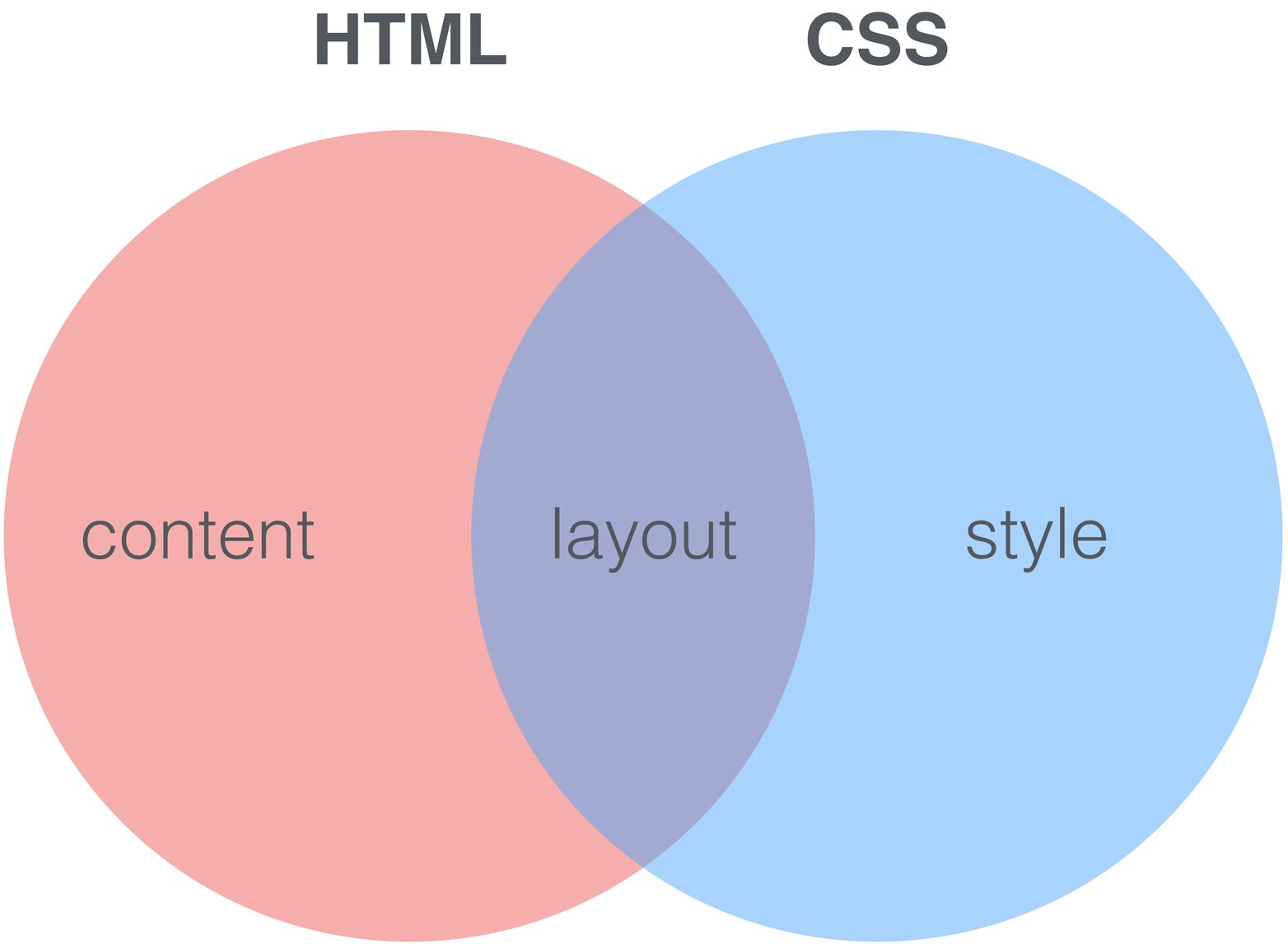
HTML & CSS

Layout laid out

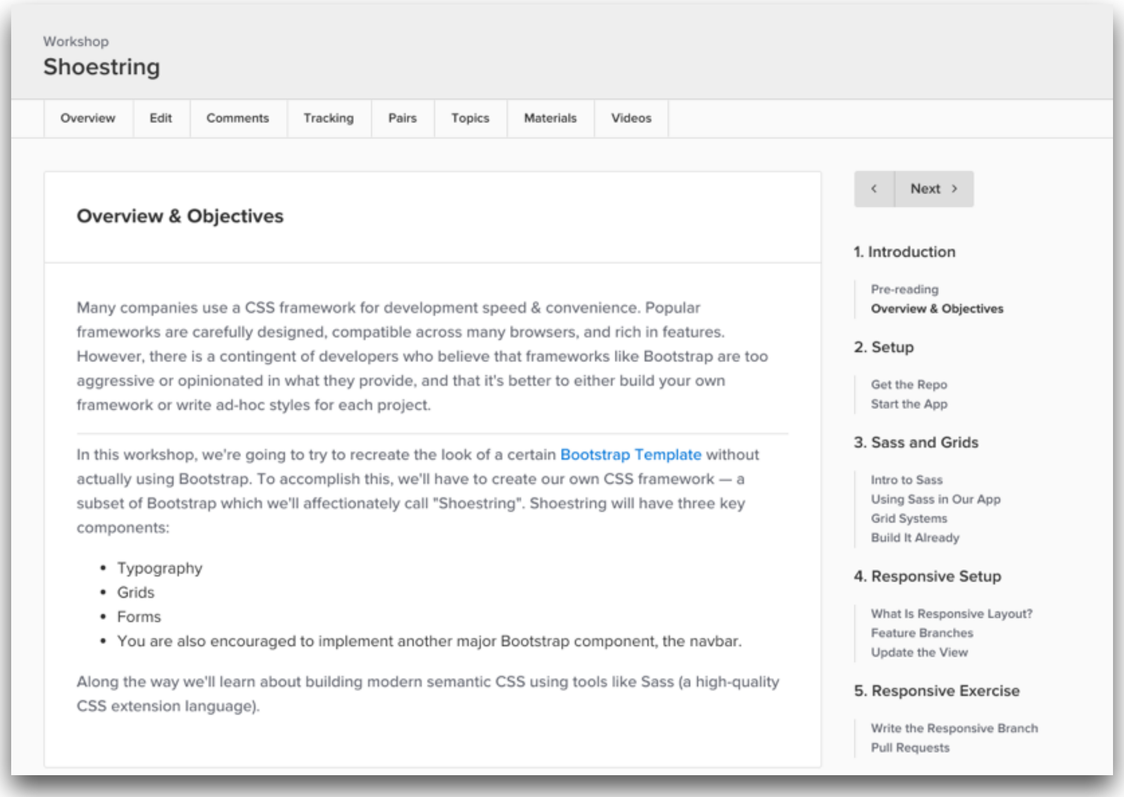

HTML
CSS
content
layout
style

WHY IS CSS IMPORTANT?

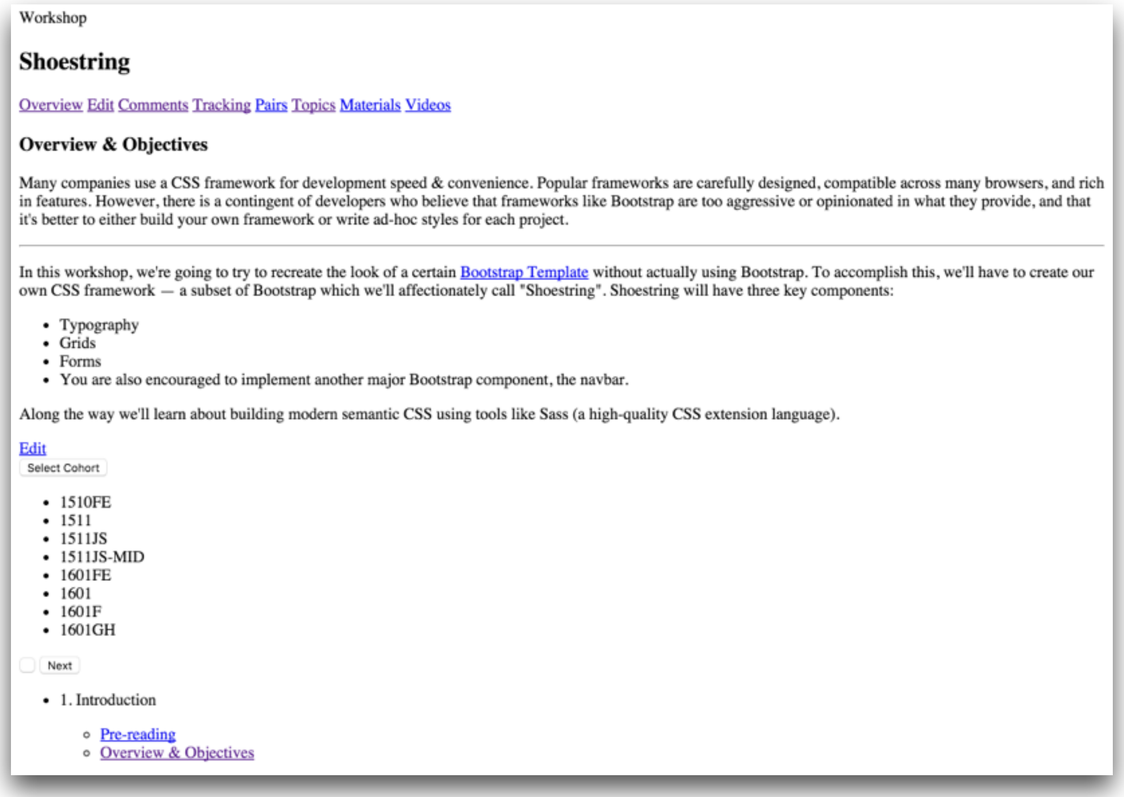
CSS IS IMPORTANT

- The web needs to look nice.
- It’s the only game in town.
- Be a triple threat.

WITH CSS



WITHOUT CSS



TERMS


selector
property
keyword
declaration
rule
article li > a:hover {
border: 1px solid red;
font-style: italic;
}

SELECTORS

tag	`input`
class	`.btn`
id	`#upload`
attribute	`[type="file"]`
pseudo-element	`::after`
pseudo-class	`:hover`
*	*

COMBINATORS

descendent	(whitespace)
child	>
next sibling	+
later sibling	~

BEWARE!

`tag.class`	element with BOTH `tag` AND `.class`
`tag .class`	element with `.class` whose ANCESTOR matches `tag`
`tag,.class`	element with EITHER `tag` OR `.class`
`tag+.class`	element with `.class` whose left SIBLING matches `tag`

CASCADING STYLE SHEETS

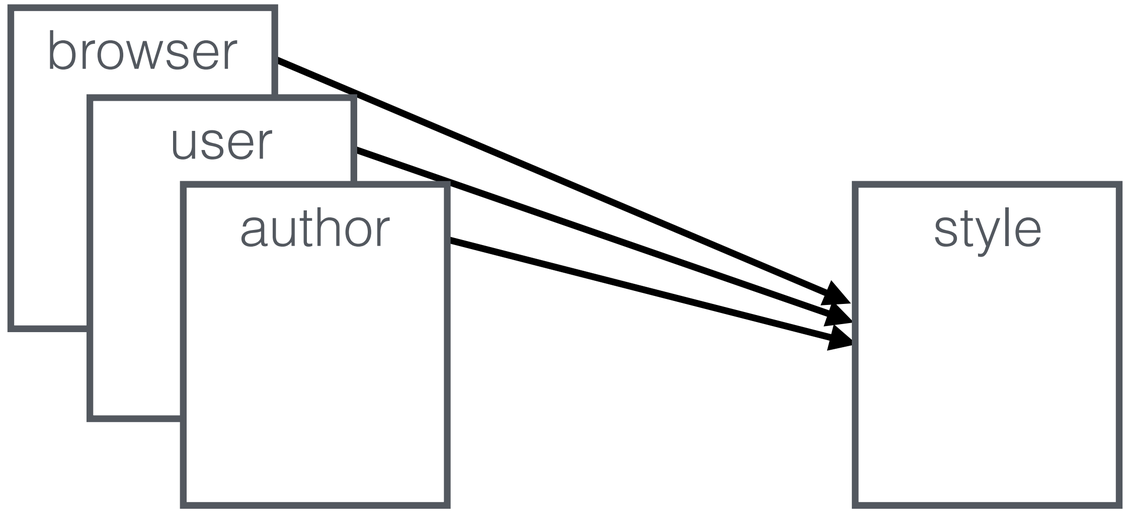
CASCADING

In ~1994... *CSS had one feature that distinguished it from all the [competing style languages]: it took into account that on the Web the style of a document couldn't be designed by either the author or the reader on their own, but that their wishes had to be combined, or "cascaded," in some way.*

CASCADING STYLE SHEETS, DESIGNING FOR THE WEB, BY HÅKON WIUM LIE AND BERT BOS (1999) - CHAPTER 20

CASCADING

An element's style is a merge of every rule whose selector matches



index.html

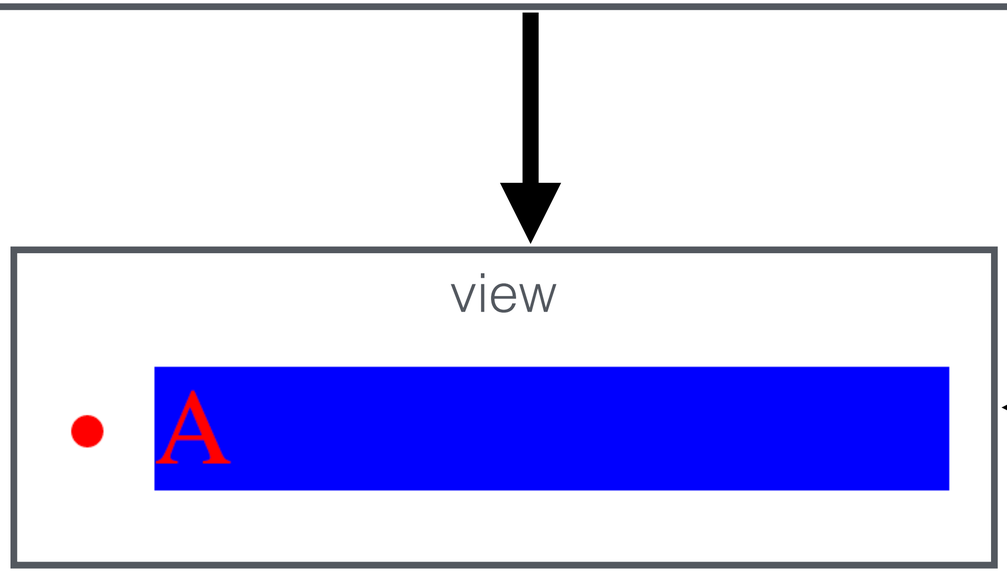
```
<head>
  <link rel="stylesheet" href="styles-B.css" />
  <link rel="stylesheet" href="styles-A.css" />
</head>
<body>
  <ul>
    <li style="background-color:blue;">A</li>
  </ul>
</body>
```

styles-A.css

```
li {
  color: red;
}
```

styles-B.css

```
li {
  font-size: 40px;
}
```



style

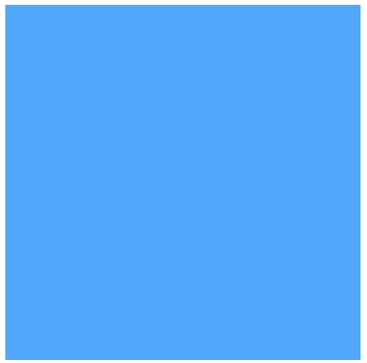
```
element.style {
  background-color: blue;
}

li {                                    styles-A.css:1
  color: red;
}

li {                                    styles-B.css:1
  font-size: 40px;
}

li {                                    user agent stylesheet
  display: list-item;
  text-align: -webkit-match-parent;
}
```

What happens when declarations conflict?

```
<div id="thing"></div>
```

```
div {
  background: red;
}
```


```
#thing {
  background: blue;
}
```



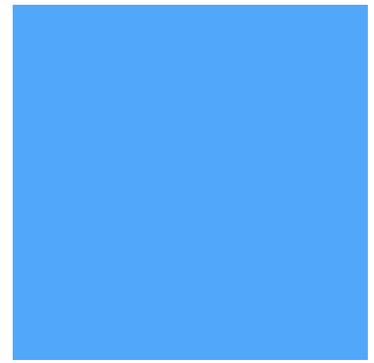
```
<div class="foo"></div>
```

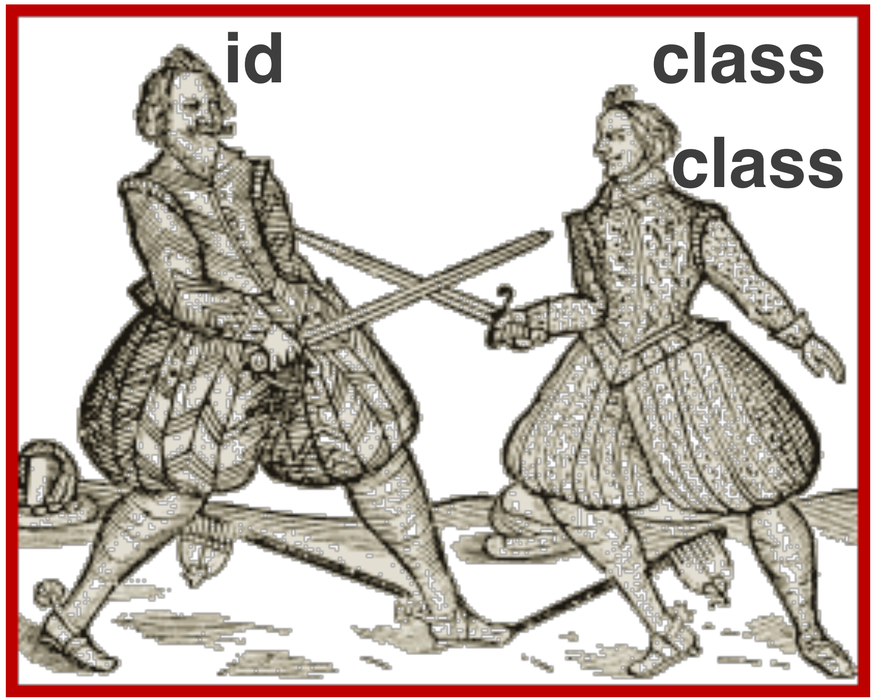
```
div {
  background: red;
}
```


```
.foo {
  background: green;
}
```

```
<div id="thing" class="foo bar"></div>
```

```
#thing {
  background: blue;
}
```

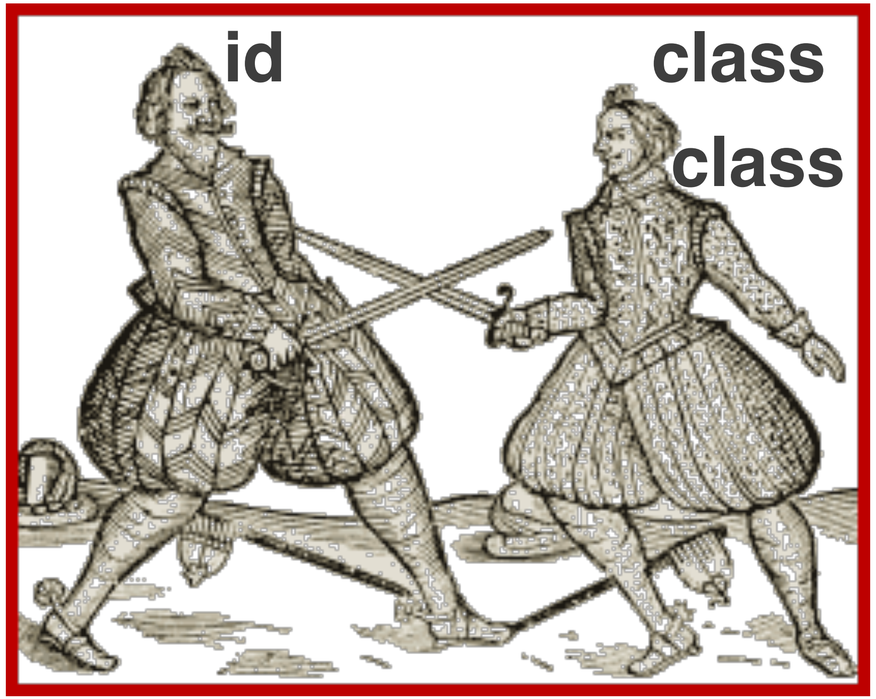





```
.foo.bar {
  background: green;
}
```

```
<div class="outer">
  <div id="thing" class="foo" style="background:orange;"></div>
</div>
```

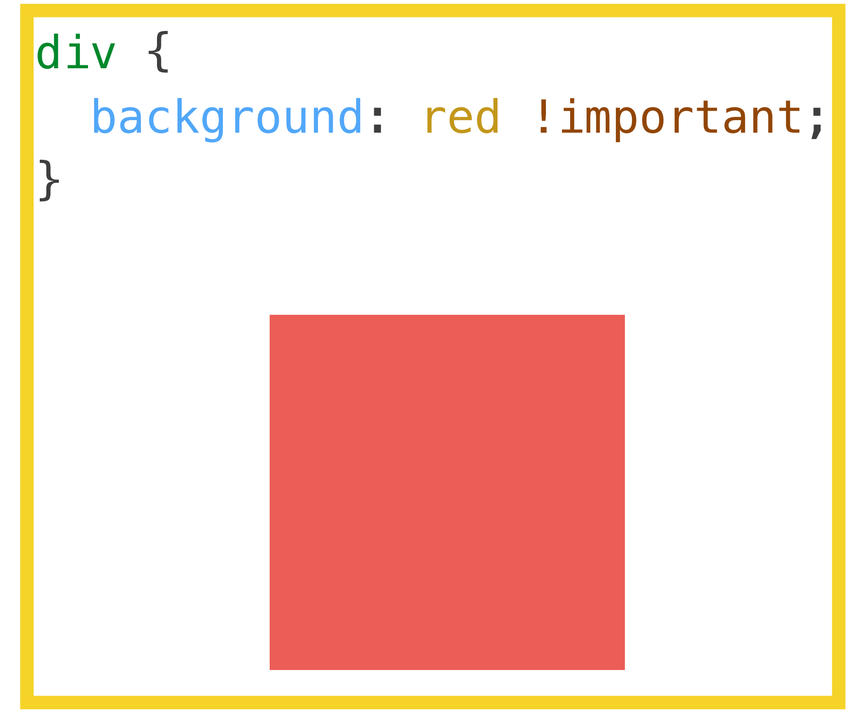
```
#thing {
  background: blue;
}
```

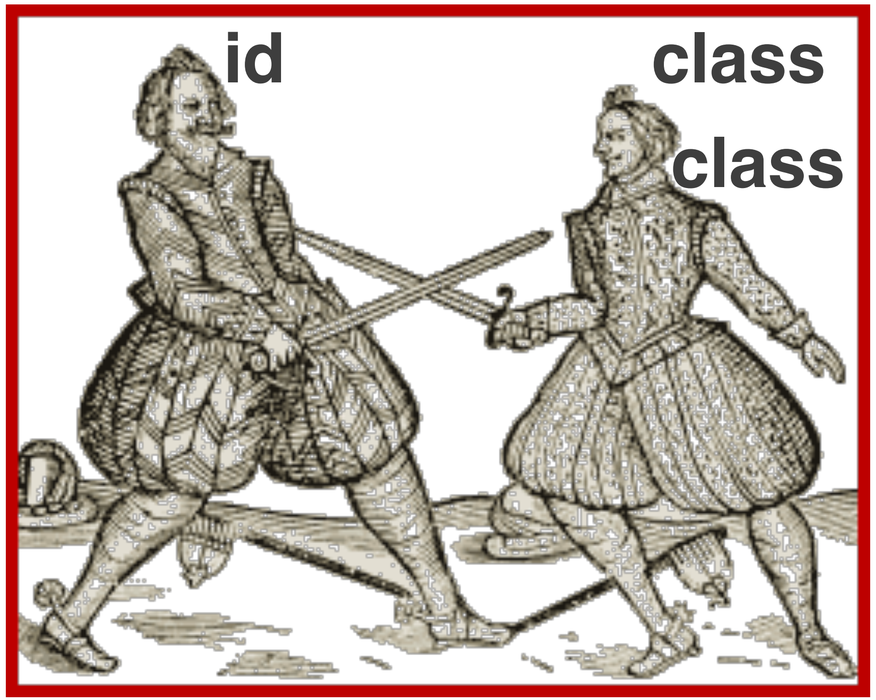



```
.outer .foo {
  background: green;
}
```

```
<div class="outer">
  <div id="thing" class="foo" style="background:orange;"></div>
</div>
```

```
div {
  background: red !important;
}
```



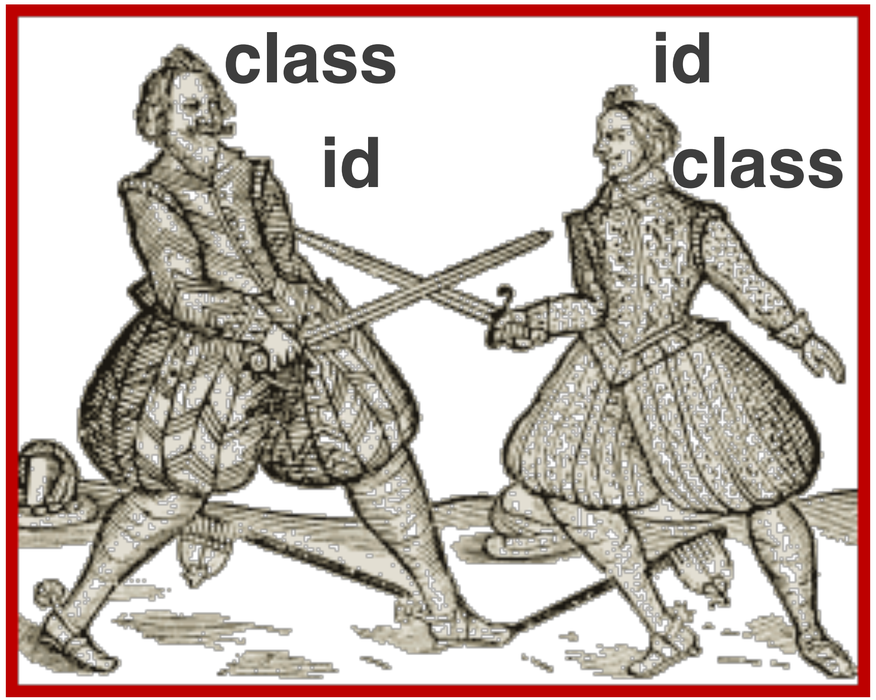



```
.outer .foo {
  background: green;
}
```

depends on which rule gets defined *last*

```
<div id="profile" class="outer">
  <div id="thing" class="foo"></div>
</div>
```

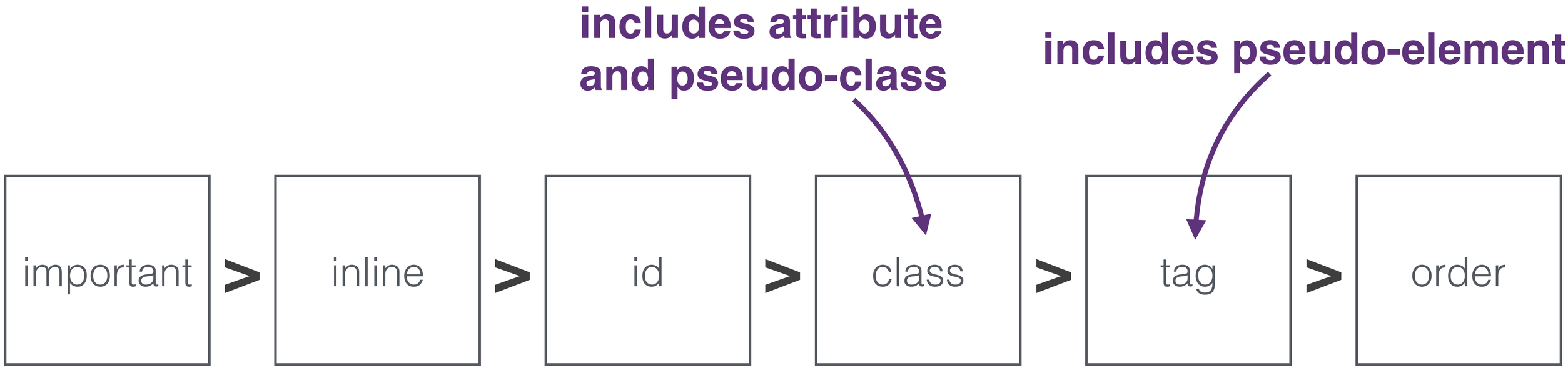
```
.outer #thing {
  background: purple;
}
```




```
#profile .foo {
  background: brown;
}
```

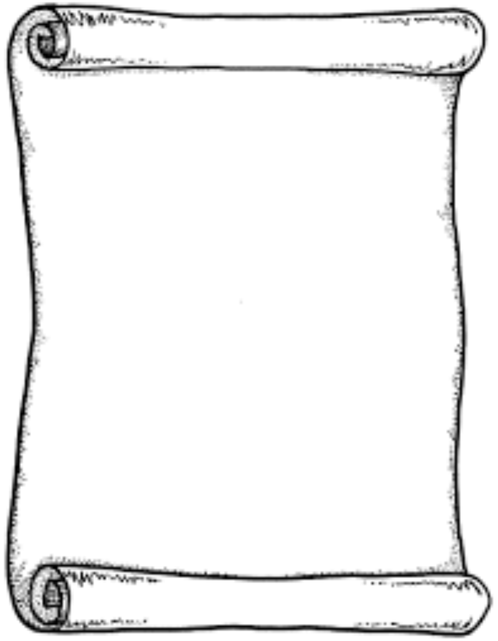
THAT WAS CSS SPECIFICITY

DECLARATION SPECIFICITY

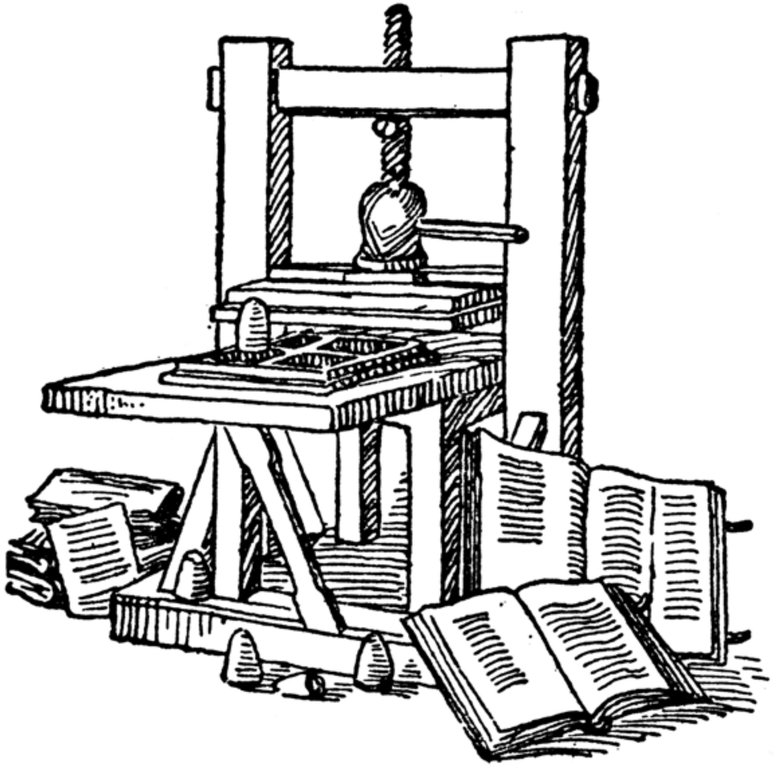


Combinators don’t affect specificity!

LAYOUT



HTTP://WWW.CLIPARTBEST.COM/CLIPART-9I4O55P6T



HTTP://ETC.USF.EDU/CLIPART/44800/44880/44880_GUTEN_PRESS_LG.GIF

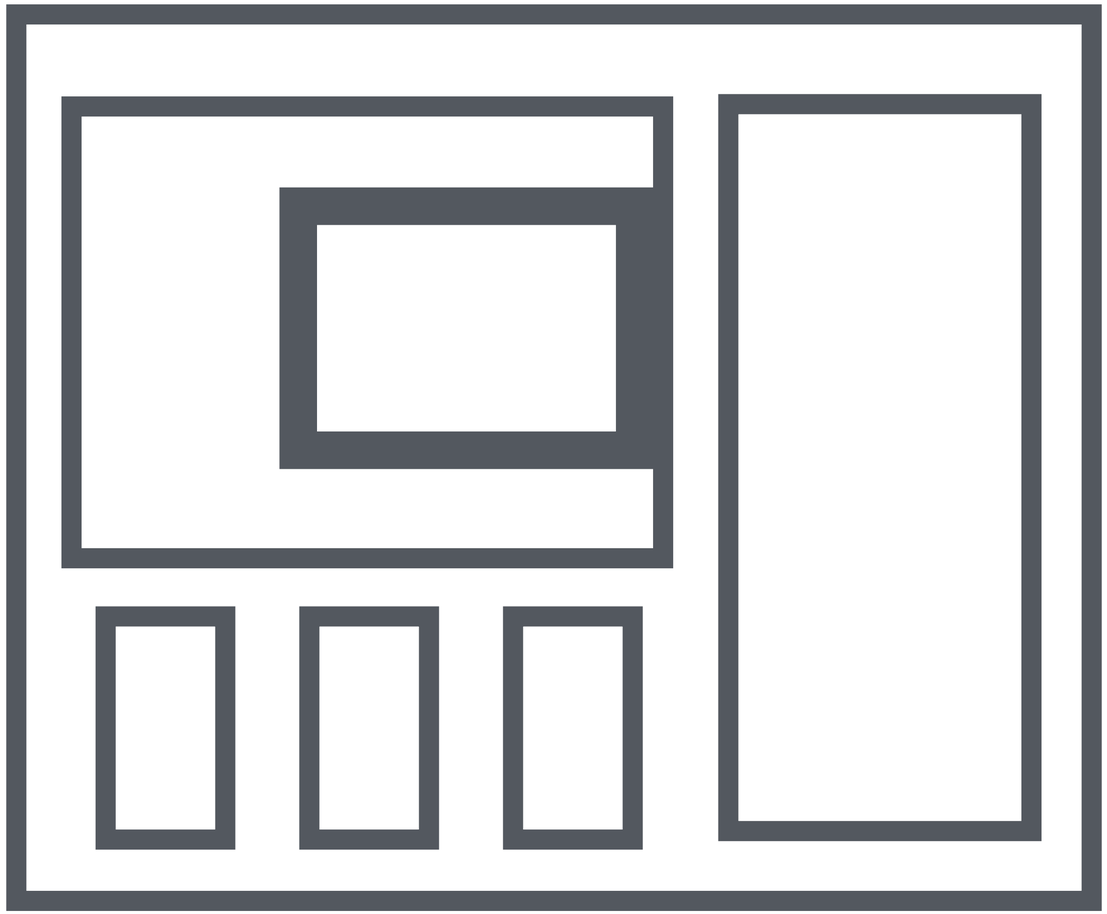
HTTP://ASSETS.UXBOOTH.COM/UPLOADS/2009/11/MYSPACE.PNG

THE BOX MODEL

"Everything on the DOM is a rectangle"

JOE ALVES

BOX MODEL



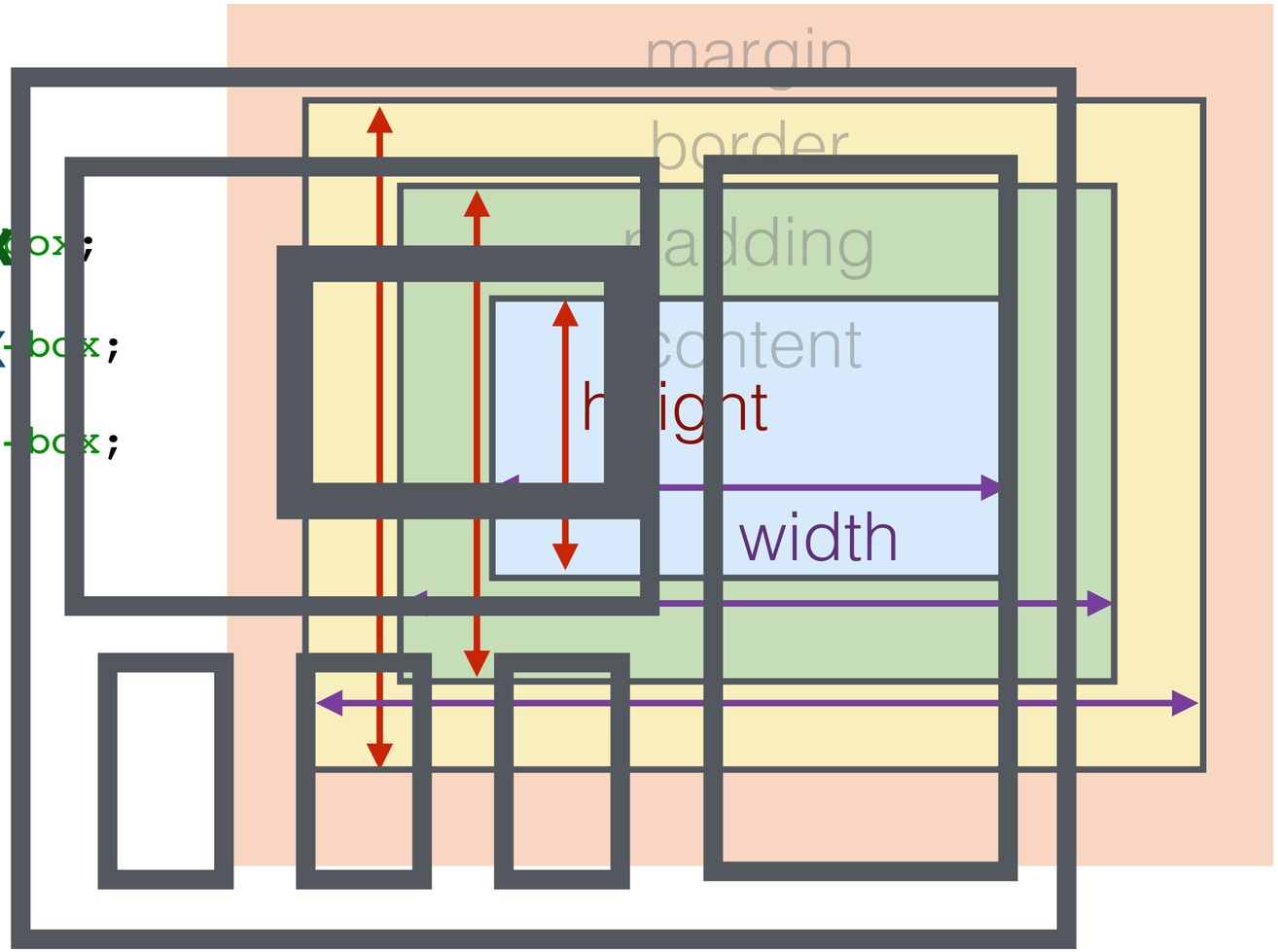
BOX MODEL

```
{
  box-sizing: border-box;
}
{
  box-sizing: padding-box;
}
{
  box-sizing: content-box;
}
```

border box

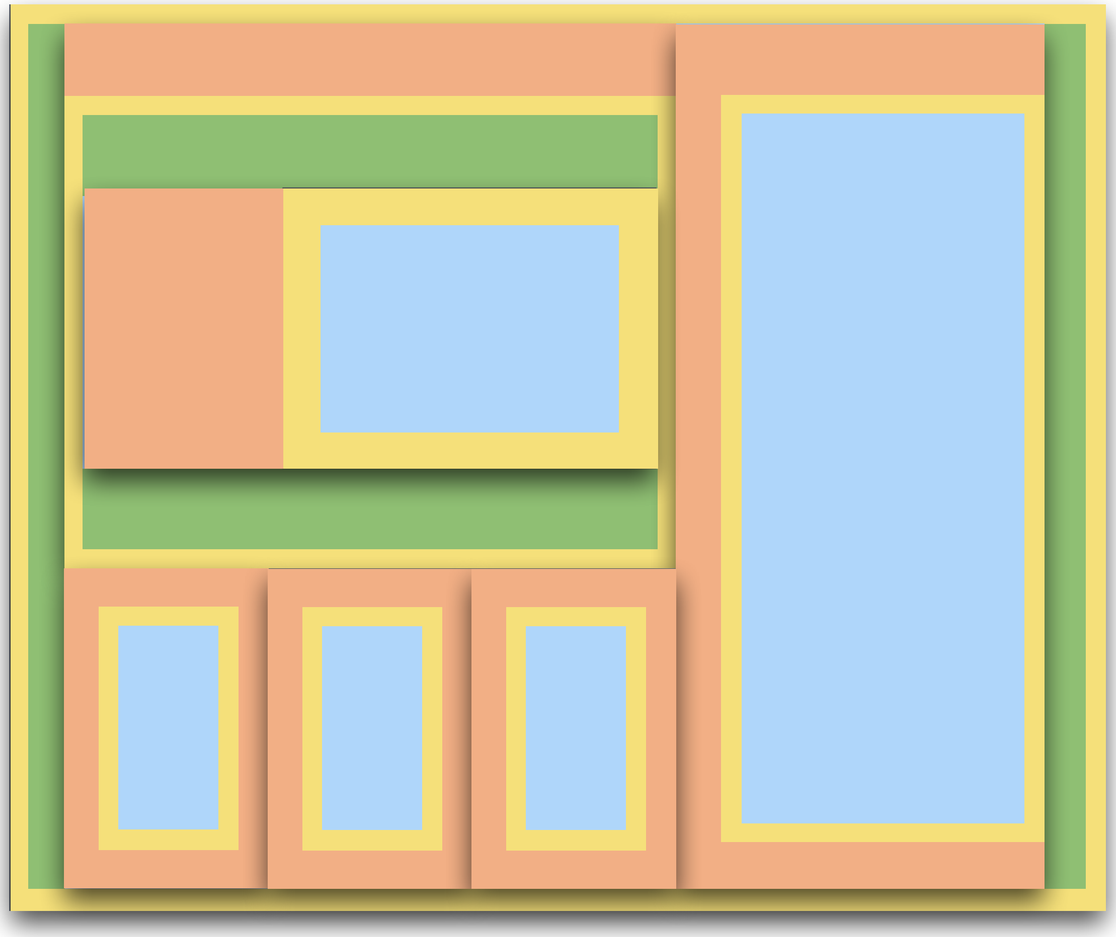
padding box

content box



BOX MODEL

fractal!



BLOCK vs INLINE vs INLINE-BLOCK

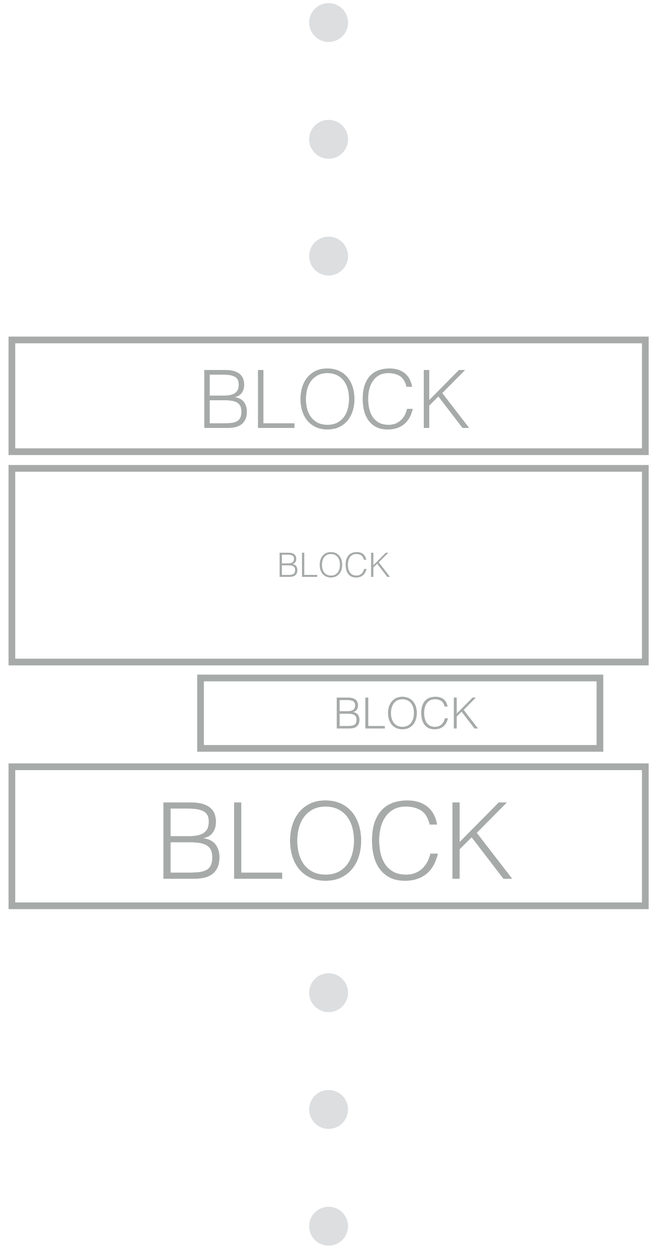
-
- <h1>, <h2>, etc.
- <p>
- <form>
- <header>, <footer>, <main>, <section>, <nav>

- <a>
-
- ,

- <img>
- <input>
- <textarea>

BLOCK-LEVEL

- By default will try to **clear** their own line
- Default width is 100% of parent
- Default height will expand to fit all children
- Can have margins on all sides
- Can set height and width

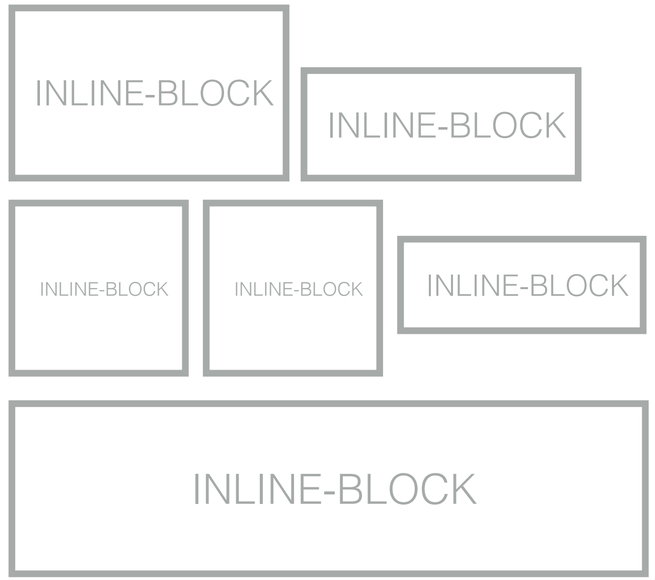


INLINE-LEVEL

- Flows with content (does **not** clear line)
- Ignores top and bottom margins
- Height and width fit content
- Cannot set height or width

INLINE-BLOCK

- Flows with content (does **not** clear line)
- Default width will expand to fit all children
- Default height will expand to fit all children
- Can have margins on all sides
- Can set height and width

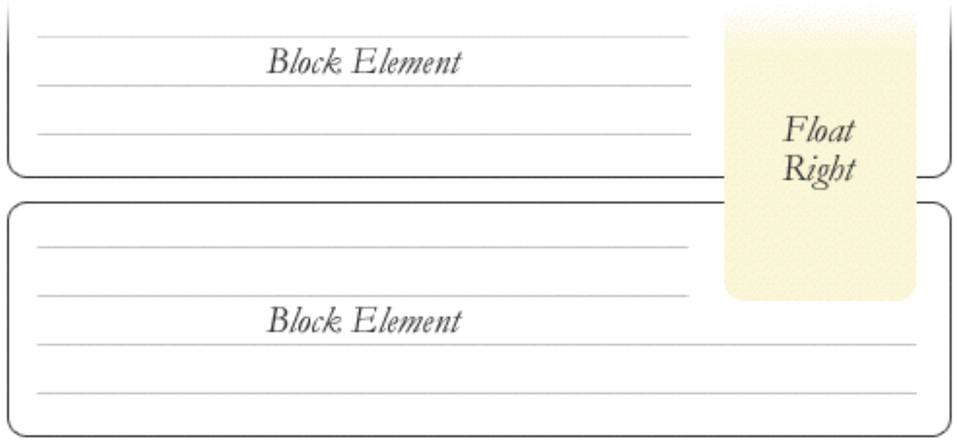


FLOAT

“I understand floats.”

–LIAR MCPANTSONFIRE

```
<body>
  <div>...</div>
  <div style="float:right;">...</div>
  <div>...</div>
</body>
```




```
<body>
  <div>...</div>
  <div style="float:right;">...</div>
  <div style="clear:right;">...</div>
</body>
```

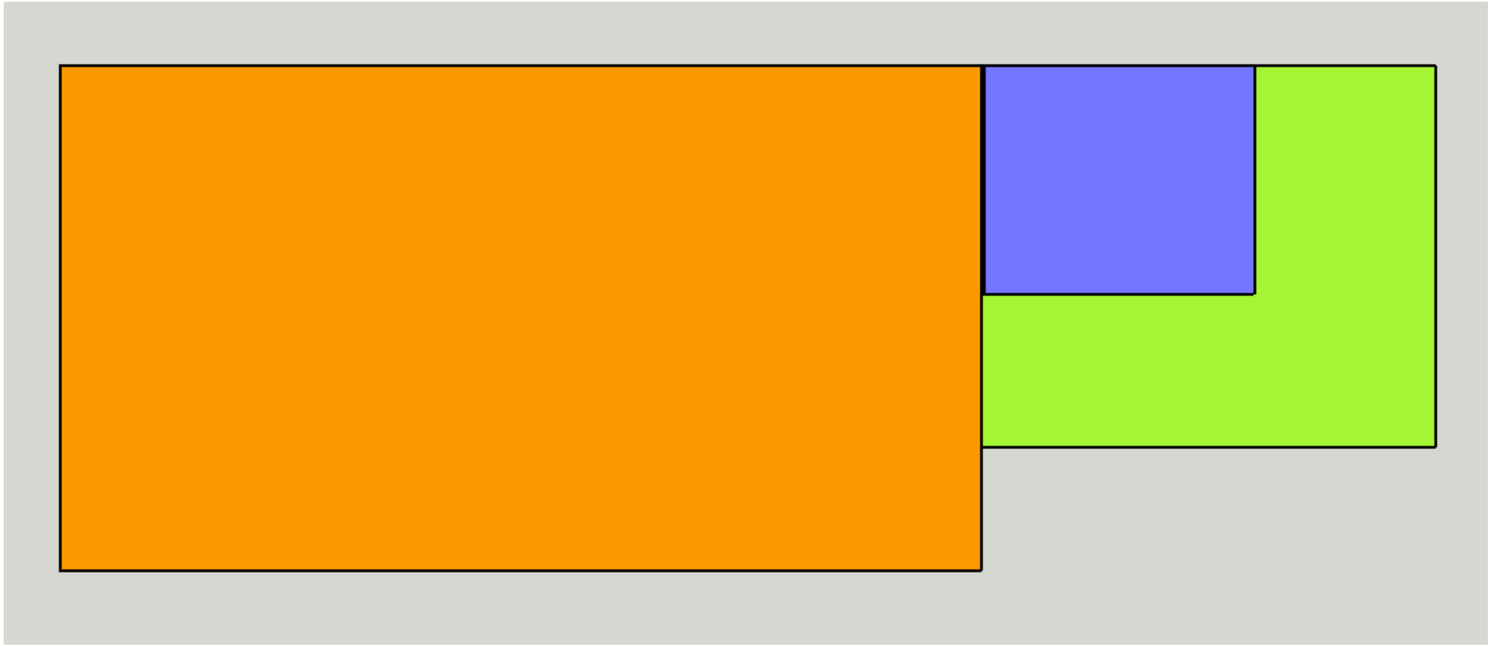
HTTPS://CSS-TRICKS.COM/ALL-ABOUT-FLOATS/

GRIDS VIA FLOAT

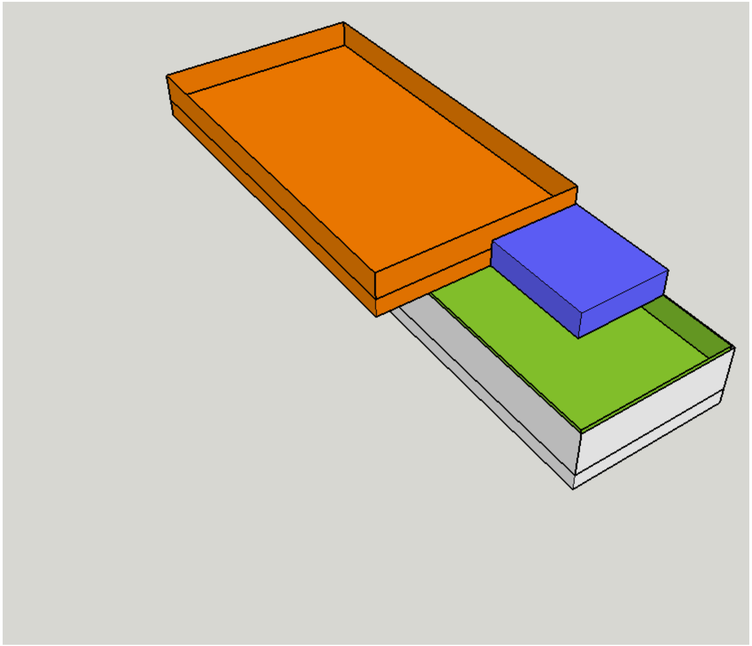
- Floats were originally intended for *wrapping text*
- Floats can be used to implement a grid/column layout
- ..but if so you must use clearfix to avoid parent collapse

```
<div>
  <div style="float:left; background:orange; width:70%;"></div>
  <div style="background:green;">
    <span style="background:blue;">X</span>
  </div>
</div>
```

renders as…

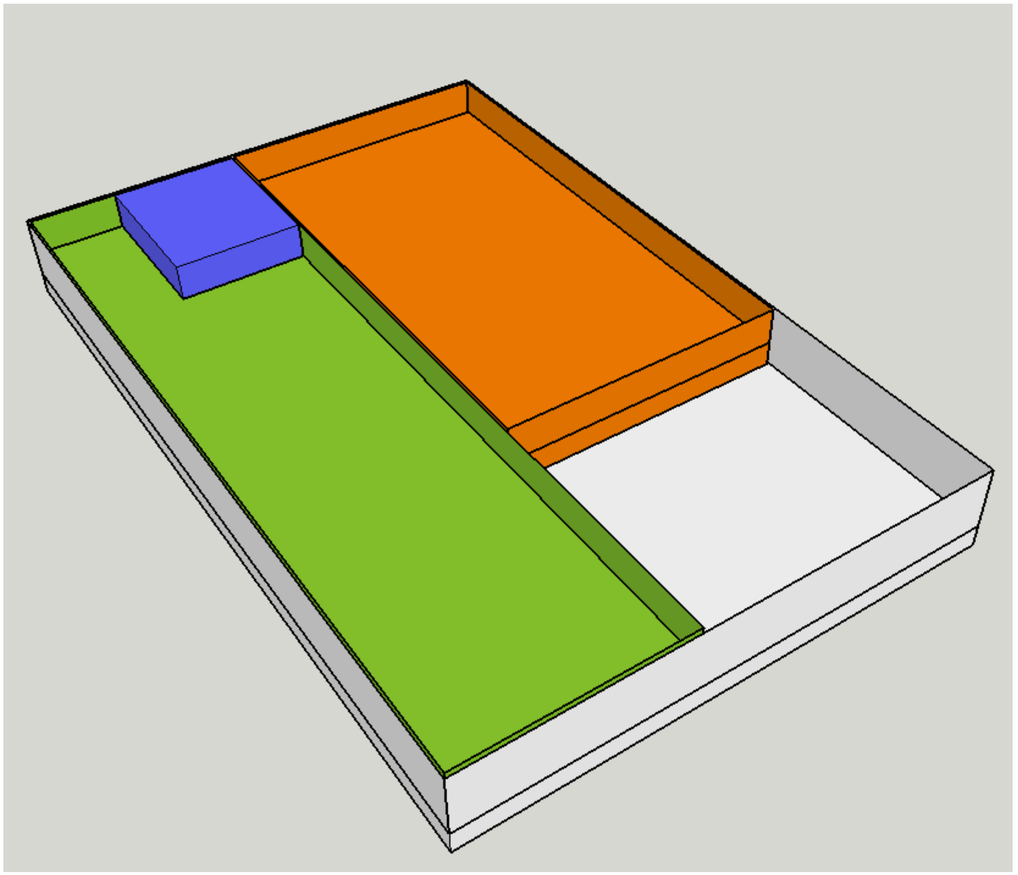


“different” perspective…



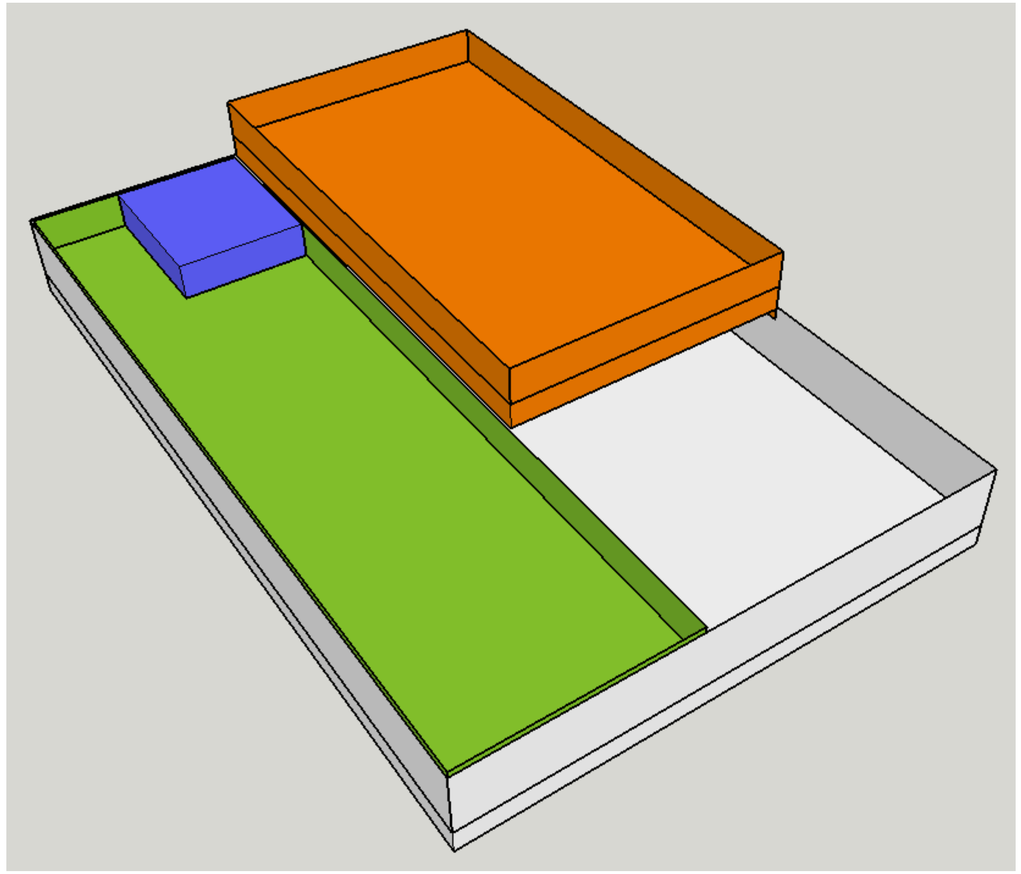
VISUAL METAPHOR
"STEP BY STEP"

```
<div>
  <div style="float:left; background:orange; width:70%;”></div>
  <div style="background:green;">
    <span style="background:blue;">x</span>
  </div>
</div>
```



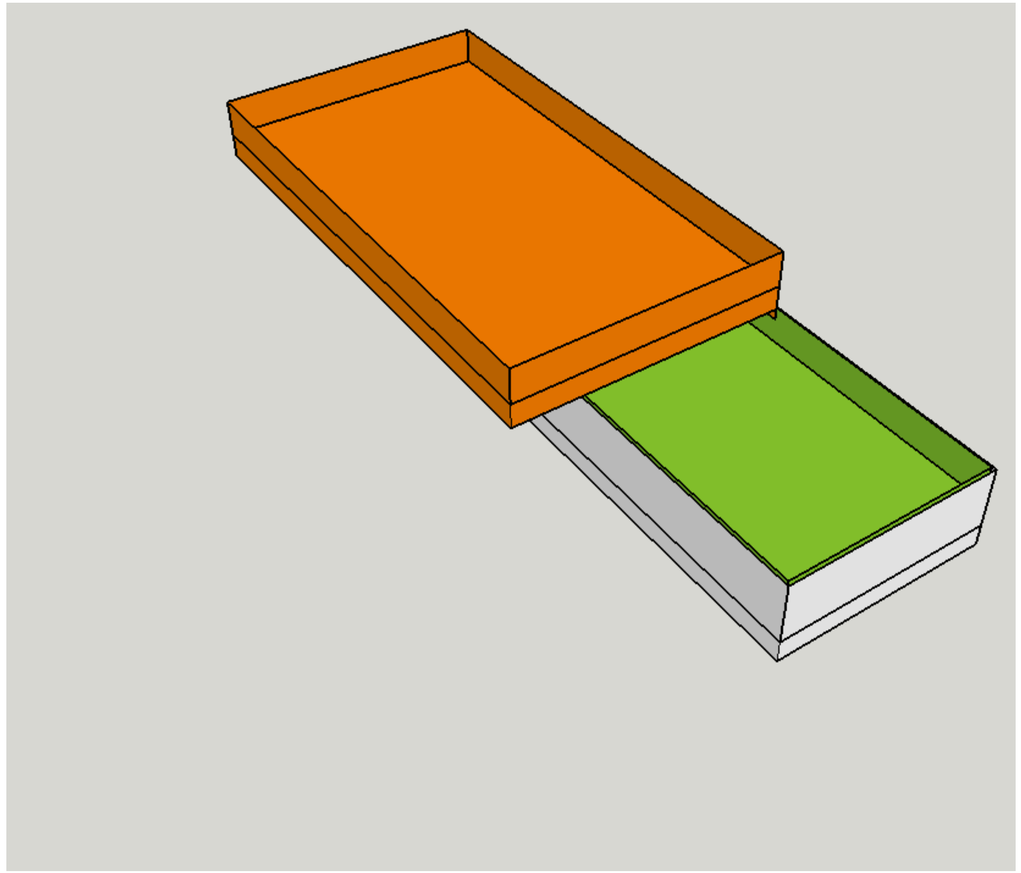
```
<div>
  <div style="float:left; background:orange; width:70%;”></div>
  <div style="background:green;">
    <span style="background:blue;">x</span>
  </div>
</div>
```

orange floats and moves to top left



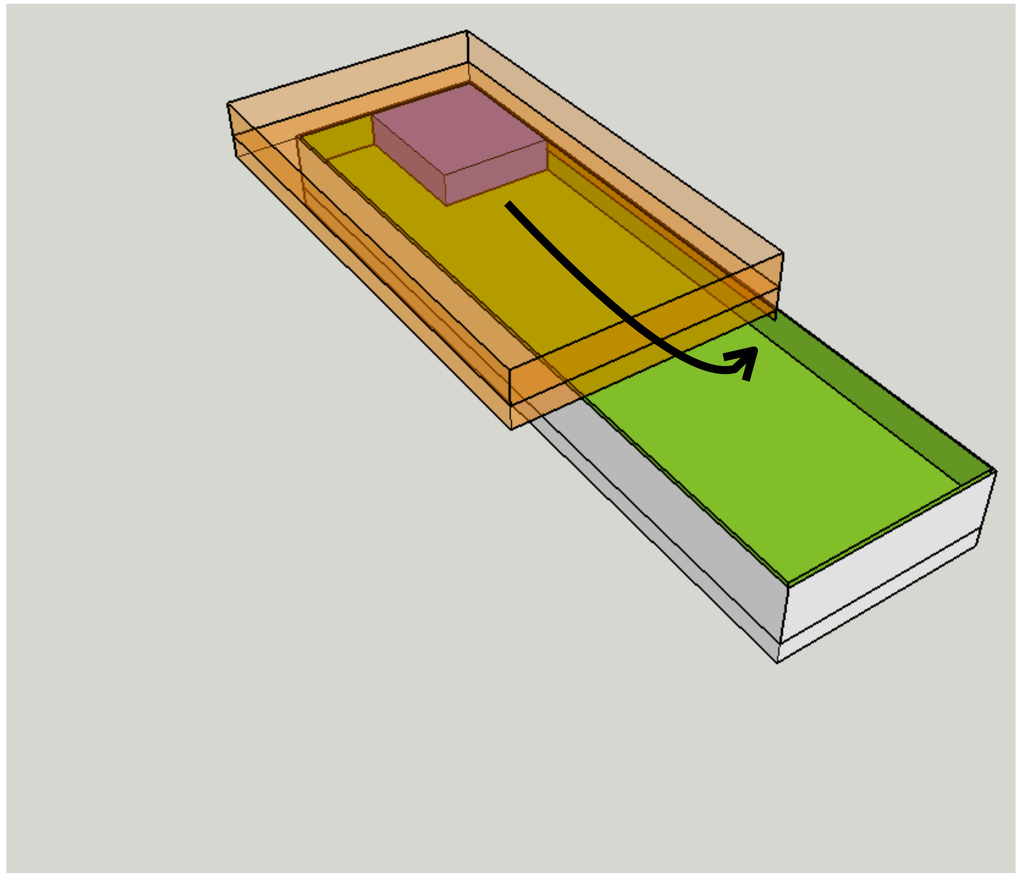
```
<div>
  <div style="float:left; background:orange; width:70%;”></div>
  <div style="background:green;">
    <span style="background:blue;">x</span>
  </div>
</div>
```

parent collapses



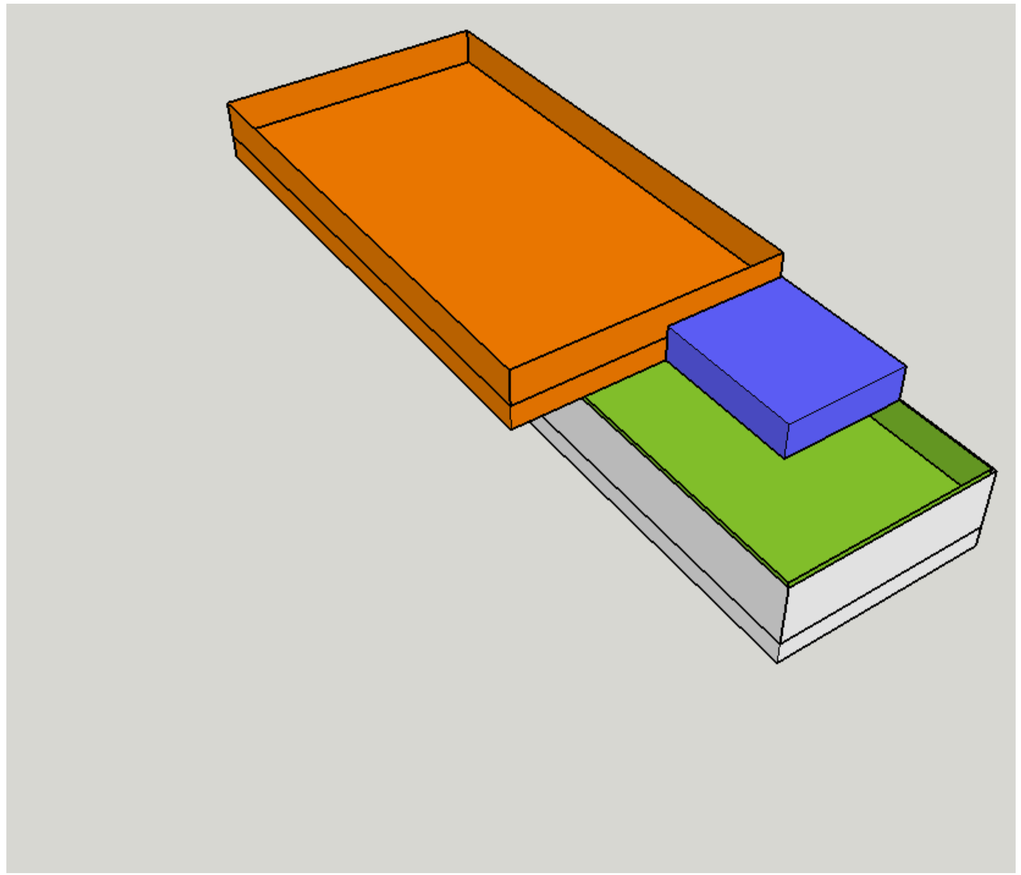
```
<div>
  <div style="float:left; background:orange; width:70%;”></div>
  <div style="background:green;">
    <span style="background:blue;">x</span>
  </div>
</div>
```

inline elements are “buoyant”



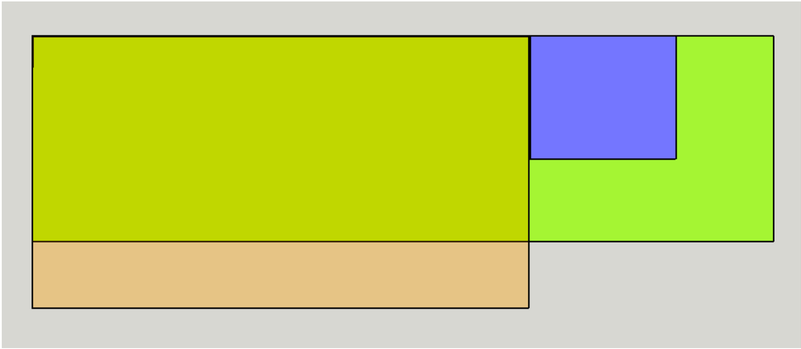
```
<div>
  <div style="float:left; background:orange; width:70%;”></div>
  <div style="background:green;">
    <span style="background:blue;">x</span>
  </div>
</div>
```

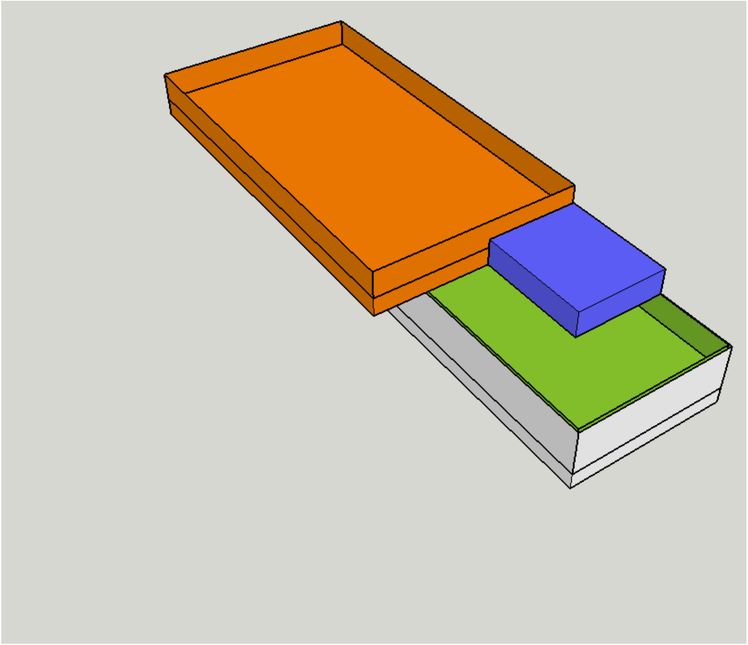
inline element wraps floated one

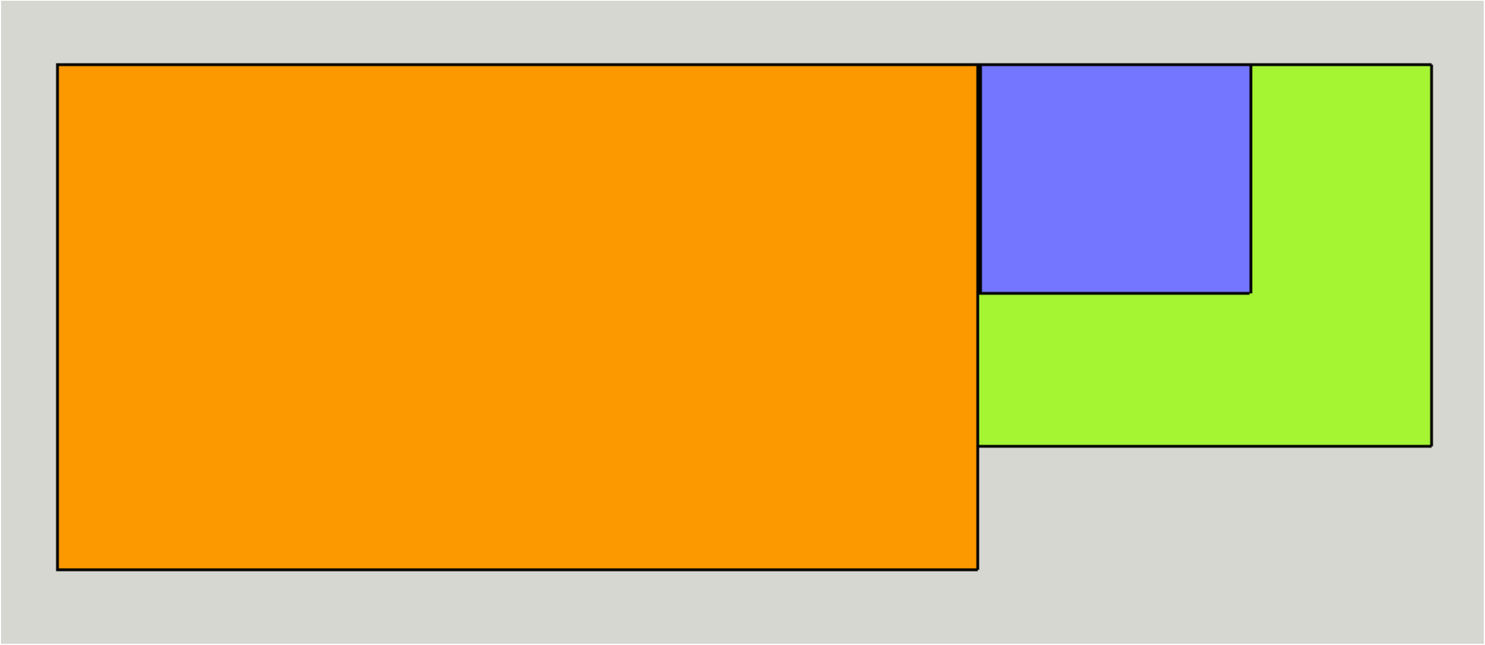


```
<div>
  <div style="float:left; background:orange; width:70%;”></div>
  <div style="background:green;">
    <span style="background:blue;">x</span>
  </div>
</div>
```

green div is behind orange div







GRIDS VIA FLOAT

- Floats were originally intended for *wrapping text*
- Floats can be used to implement a grid/column layout
- ..but if so you must use clearfix to avoid parent collapse

CLEARFIX

```
.clearfix::before, .clearfix::after {
    clear: both;
    display: block;
    content: “”;
}
```


CSS
Global
HD

SYNTACTICALLY AWESOME STYLE SHEETS

SASSY

CASCADING STYLE SHEETS

SCSS

- “nesting”
- “variables”
- “loops”
- “functions”
- “modules”

HOW DOES SCSS WORK?

SCSS COMPILES TO CSS

NESTING

SCSS

```scss
article {
    border: 1px solid red;
    li {
        background: gray;
    }
}
```

CSS

```css
article {
    border: 1px solid red;
}
article li {
    background: gray;
}
```

VARIABLES

SCSS

```scss
$deep-red: #990000;
a {
    color: $deep-red;
}
.warning {
    border-color: $deep-red;
}
```

CSS

```css
a {
    color: #990000;
}
.warning {
    border-color: #990000;
}
```

LOOPS

SCSS

```scss
@for $i from 1 through 3 {
    h#{$i} {
        font-size: $i * 10px;
    }
}
```

CSS

```css
h1 {
    font-size: 10px;
}
h2 {
    font-size: 20px;
}
h3 {
    font-size: 30px;
}
```

MIXINS

SCSS

```scss
@mixin border-radius ($r) {
    -webkit-border-radius: $r;
    -moz-border-radius: $r;
    border-radius: $r;
}

.thing {
    @include border-radius(10px);
}
```

CSS

```css
.thing {
    -webkit-border-radius: 10px;
    -moz-border-radius: 10px;
    border-radius: 10px;
}
```

MODULES

SCSS

```scss
/* pulls in normalize.scss */
@import 'normalize';
```

CSS

```css
/* ... */
/**
 * 1. Set default font family to sans-serif
 * 2. Prevent iOS text size adjust after orientation change, without disabling
 *    user zoom.
 */
html {
    font-family: sans-serif;
    /* 1 */
    -ms-text-size-adjust: 100%;
    /* 2 */
    -webkit-text-size-adjust: 100%;
    /* 2 */ }
/**
 * Remove default margin.
 */
body {
    margin: 0; }
/* ==========================================================================
   Links
   ========================================================================== */
/**
 * Remove the gray background color from active links in IE 10.
 */
a {
    background: transparent; }
/* ... */
```

RESPONSIVE

RESPONSIVE


I heard you want to be a web developer
Here are a few devices to test your site

RESPONSIVE DESIGN

- Website is fully functional for all screen sizes, resolutions and orientations
- Born out of necessity (see previous slide)
- Developers and designers should cater to the user's environment, not the other way around

MEDIA QUERIES

The CSS3 media queries let us set CSS rules that only apply at specific screen sizes

```
/* EXAMPLE */
/* for 980px or less */
@media screen and (max-width: 980px) {
    #pagewrap {
        width: 94%;
    }
    #content {
        width: 65%;
    }
    #sidebar {
        width: 30%;
    }
}
```

HOW TO USE MEDIA QUERIES

- Set default styles for your page OUTSIDE of the media queries
- Progressively target smaller page sizes and let styles cascade
- Make things flexible and fluid
- Customary to add your responsive media-query tags to a separate CSS file

EXAMPLE MEDIA QUERIES

MIN-WIDTH

```
/* This rule will apply if viewing area is >900px */
@media screen and (min-width: 900px) {
  .class {
    background: #666;
  }
}
```

MULTIPLE MEDIA QUERIES

```
/* The following code will apply if the viewing area is
between 600px and 900px */
@media screen and (min-width: 600px) and (max-width: 900px) {
  .class {
    background: #333;
  }
}
```

DEVICE WIDTH

max-device-width means the **actual resolution of the device**
max-width means the **viewing area resolution**

```
/* The following code will apply if the max-
device-width is 480px (eg. iPhone display). */
@media screen and (max-device-width: 480px) {
  .class {
    background: #000;
  }
}
```

SUMMARY

- CSS role
- terminology
- specificity
- layout & float
- box model
- SCSS
- responsive CSS

SO MUCH MORE...

- consistent centering (guide)
- fonts (intro)
- flexbox (guide, game)
- animations (intro)
- z-index (explanation)
- margin collapsing (docs)
- attribute selection operators (docs)
- patterns (intro)
- sprites (intro)