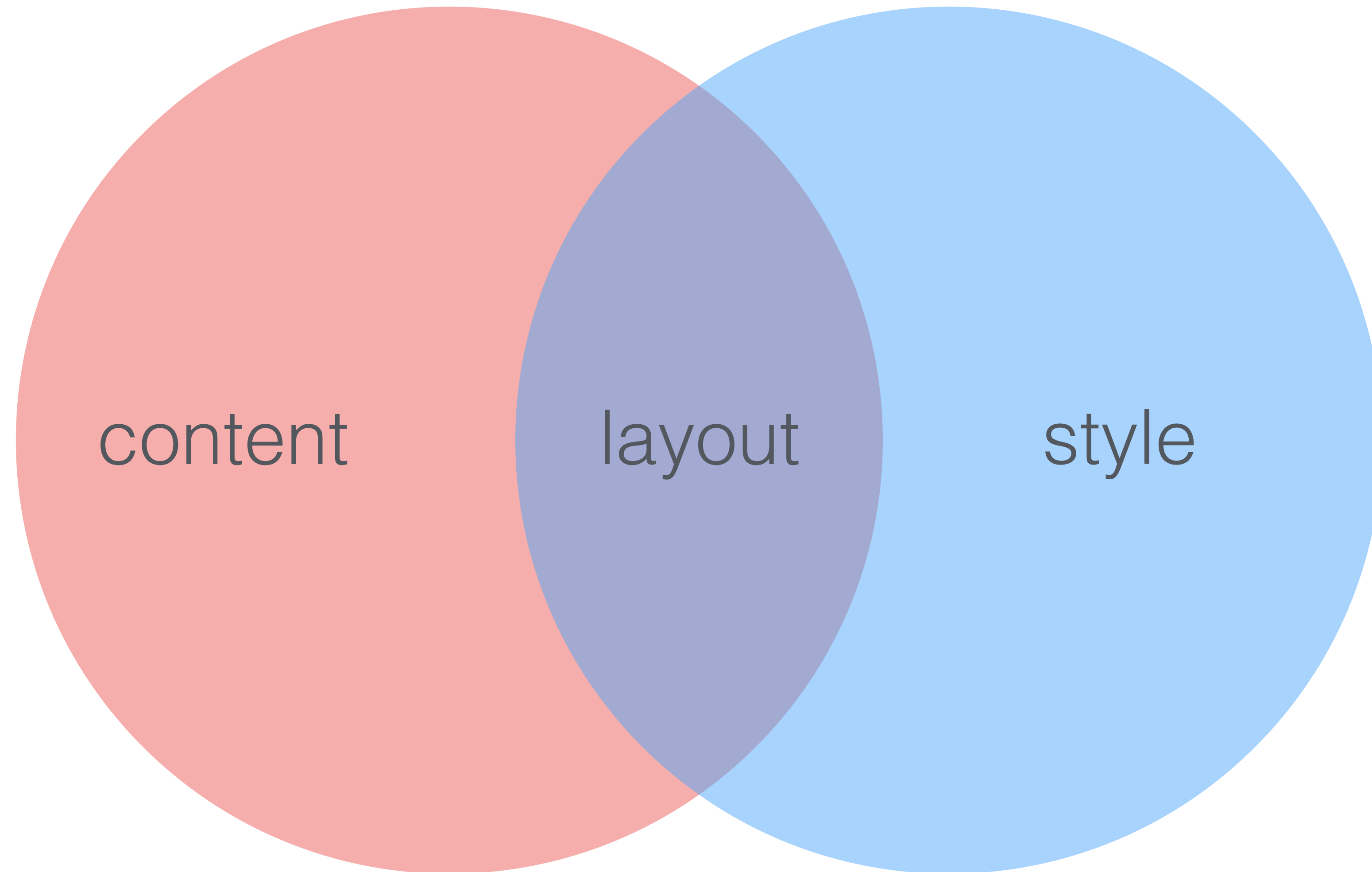


HTML & CSS

Layout laid out

HTML

CSS





I Am Devloper
@iamdevloper

Following



CSS is easy. It's like riding a bike, which is on fire and the ground is on fire and everything is on fire because it is hell.



I Am Developer
@iamdeveloper

Following



"Learn CSS in 24 Hours"

After 3 hours, you've learnt all of CSS.

Hours 3 to eternity are spent trying to vertically center stuff.

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

Workshop

Shoestring

Overview

Edit

Comments

Tracking

Pairs

Topics

Materials

Videos

Overview & Objectives

Many companies use a CSS framework for development speed & convenience. Popular frameworks are carefully designed, compatible across many browsers, and rich in features. However, there is a contingent of developers who believe that frameworks like Bootstrap are too aggressive or opinionated in what they provide, and that it's better to either build your own framework or write ad-hoc styles for each project.

In this workshop, we're going to try to recreate the look of a certain [Bootstrap Template](#) without actually using Bootstrap. To accomplish this, we'll have to create our own CSS framework — a subset of Bootstrap which we'll affectionately call "Shoestring". Shoestring will have three key components:

- Typography
- Grids
- Forms
- You are also encouraged to implement another major Bootstrap component, the navbar.

Along the way we'll learn about building modern semantic CSS using tools like Sass (a high-quality CSS extension language).

< Next >

1. Introduction

Pre-reading

Overview & Objectives

2. Setup

Get the Repo

Start the App

3. Sass and Grids

Intro to Sass

Using Sass in Our App

Grid Systems

Build It Already

4. Responsive Setup

What Is Responsive Layout?

Feature Branches

Update the View

5. Responsive Exercise

Write the Responsive Branch

Pull Requests

WITHOUT CSS

Workshop

Shoestring

Overview

Edit

Comments

Tracking

Pairs

Topics

Materials

Videos

Overview & Objectives

Many companies use a CSS framework for development speed & convenience. Popular frameworks are carefully designed, compatible across many browsers, and rich in features. However, there is a contingent of developers who believe that frameworks like Bootstrap are too aggressive or opinionated in what they provide, and that it's better to either build your own framework or write ad-hoc styles for each project.

In this workshop, we're going to try to recreate the look of a certain [Bootstrap Template](#) without actually using Bootstrap. To accomplish this, we'll have to create our own CSS framework — a subset of Bootstrap which we'll affectionately call "Shoestring". Shoestring will have three key components:

- Typography
- Grids
- Forms
- You are also encouraged to implement another major Bootstrap component, the navbar.

Along the way we'll learn about building modern semantic CSS using tools like Sass (a high-quality CSS extension language).

[Edit](#)

Select Content

- 1510FE
- 1511
- 1511JS
- 1511JS-MTD
- 1601FE
- 1601
- 1601F
- 1601GH

☐ Next

- 1. Introduction
 - [Pre-reading](#)
 - [Overview & Objectives](#)

TERMS



RULE EXAMPLE

apply **these** styles → 

```
article li > a:hover {  
  border: 1px solid red;  
  font-style: italic;  
}
```

to any elements matching **this** selector

even for any future changes ***declarative!***

SELECTORS

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

COMBINATORS

descendent	(whitespace)	div li
child	>	div > li
next sibling	+	img + span
later sibling	~	img ~ span

BEWARE!

- `tag.class` element with BOTH `tag` AND `.class`
- `tag .class` element that is a descendent of `tag` and has `.class`
- `tag, .class` element with EITHER `tag` OR `.class`

CASCADING STYLE SHEETS

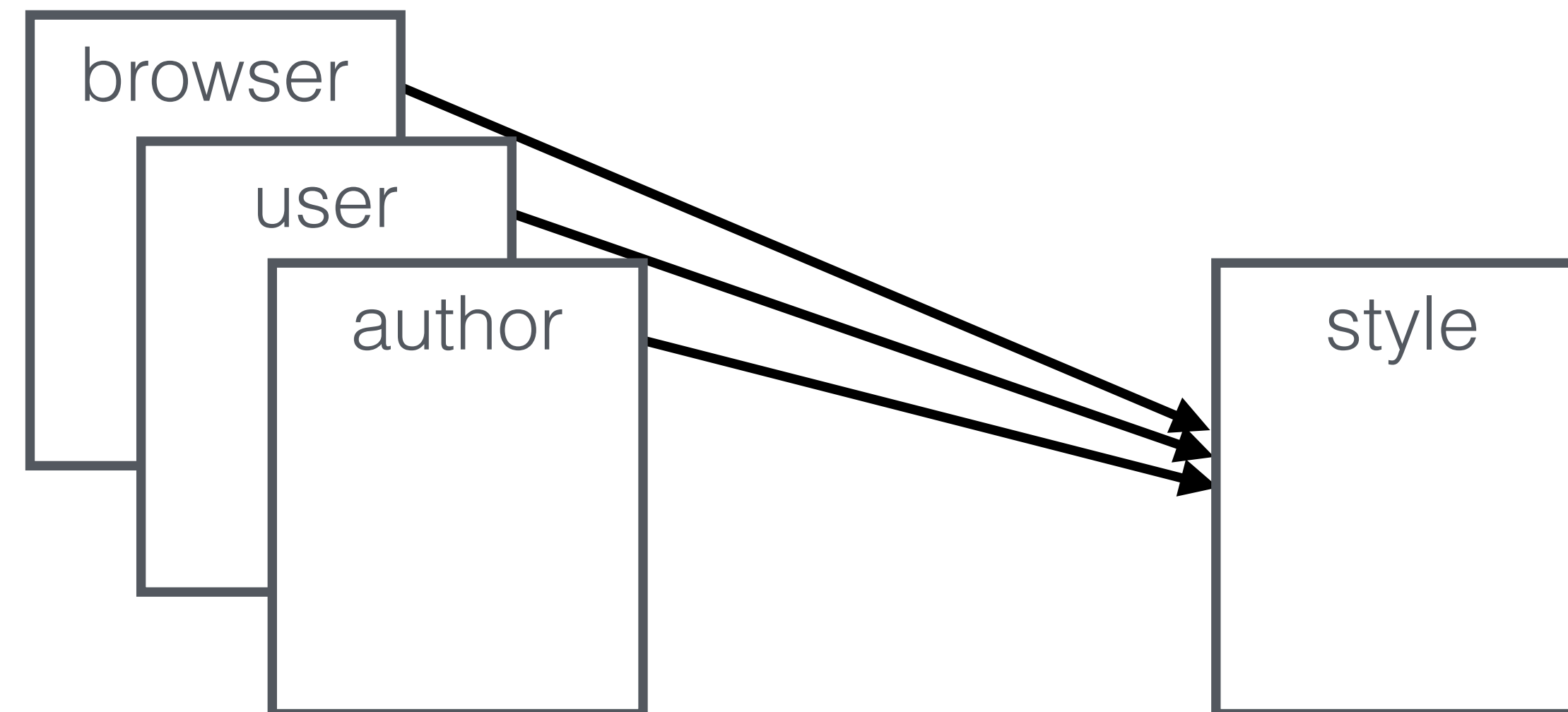
CASCADING

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 {
  color: red;
} styles-A.css:1
li {
  font-size: 40px;
} styles-B.css:1
li {
  display: list-item;
  text-align: -webkit-match-parent;
} user agent stylesheet
```

view



What happens when declarations conflict?



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

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



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




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

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

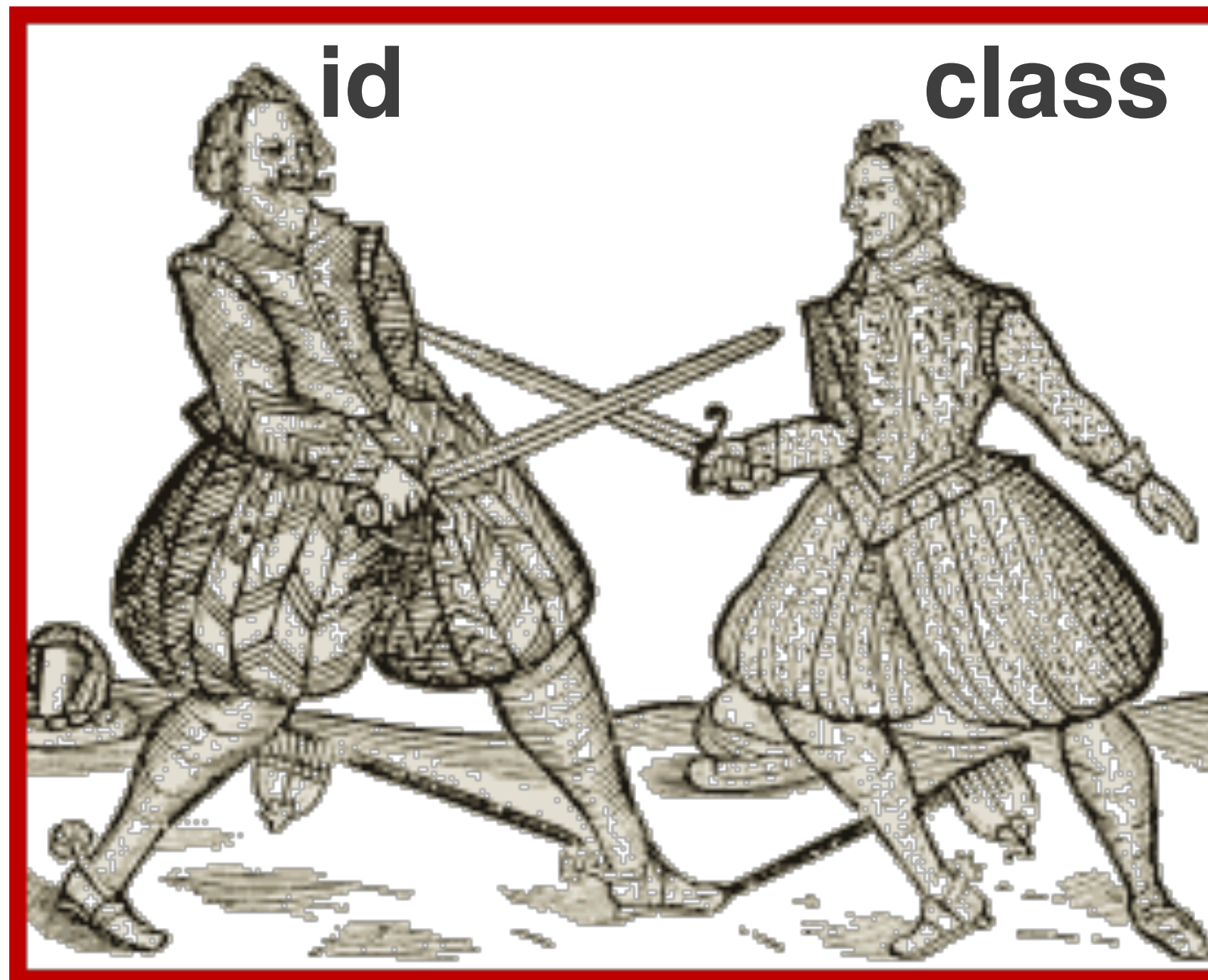


```
.foo {  
  background: green;  
}
```



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

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

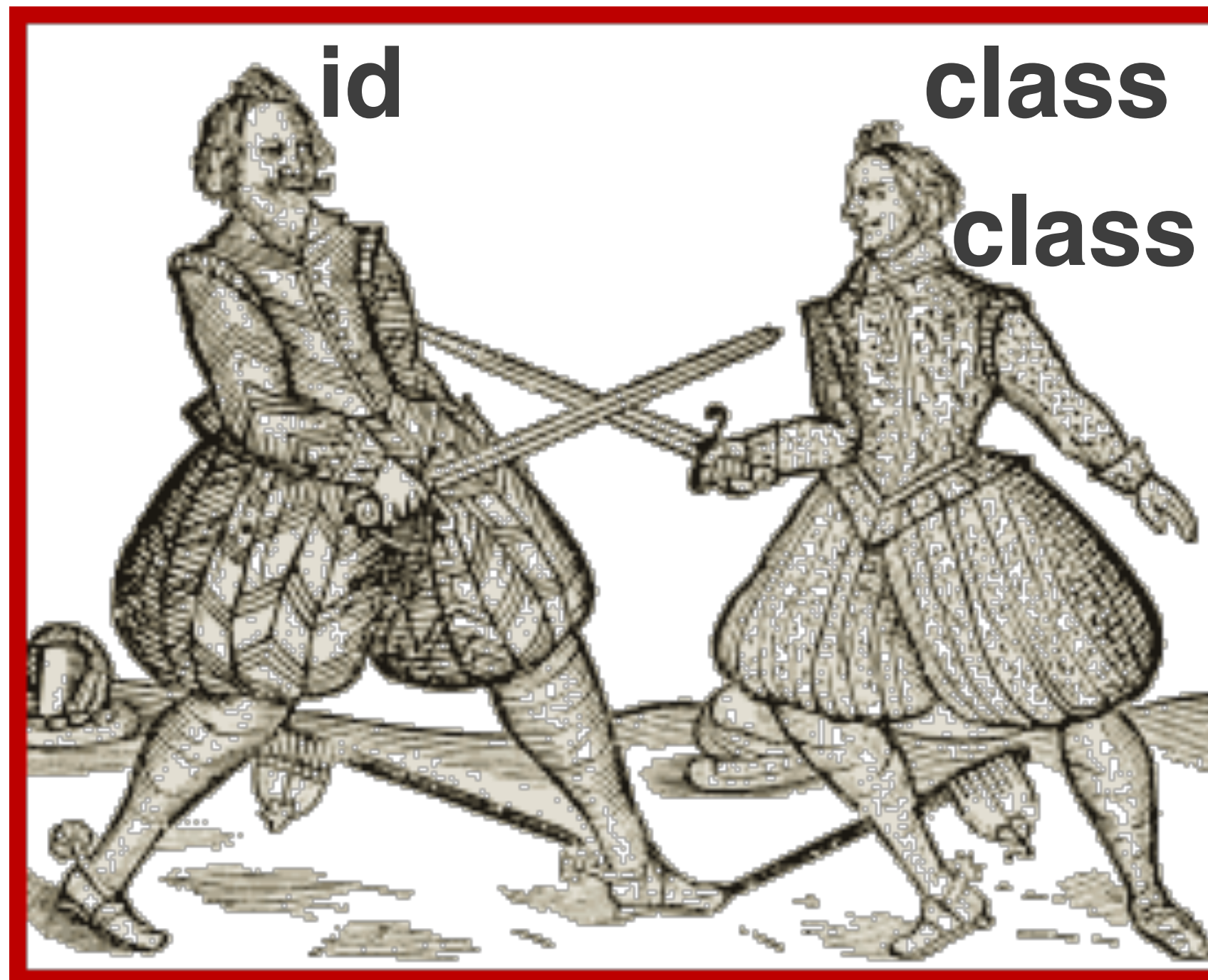


```
.foo {  
  background: green;  
}
```




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

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

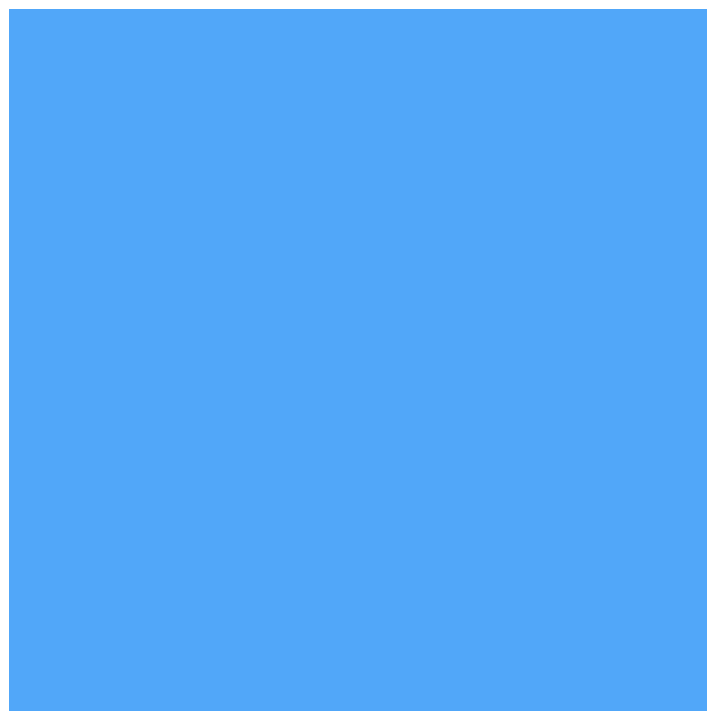


```
.foo.bar {  
  background: green;  
}
```



```
<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;  
}
```



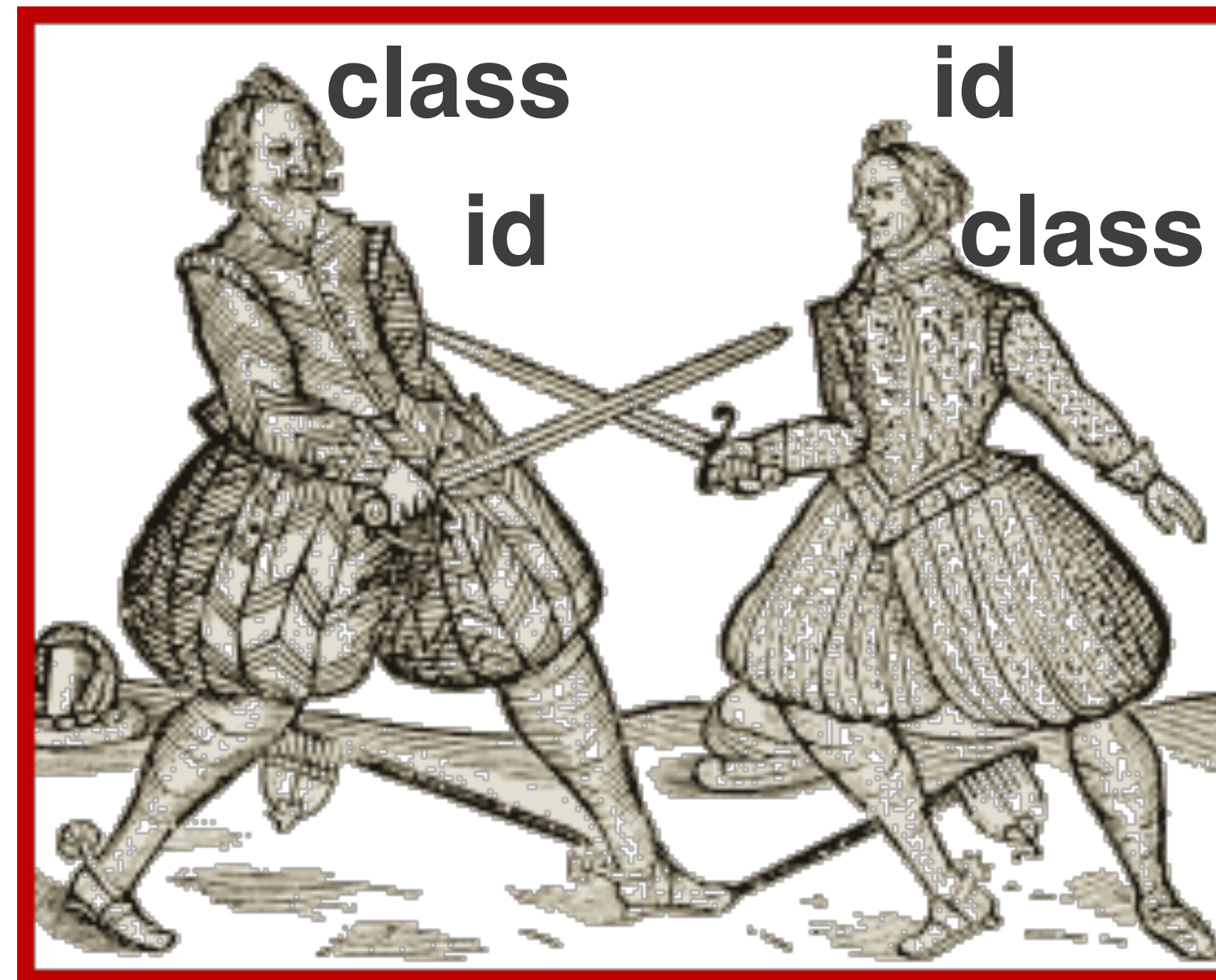
```
.outer .foo {  
  background: green;  
}
```



depends on which rule gets defined *last*

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

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

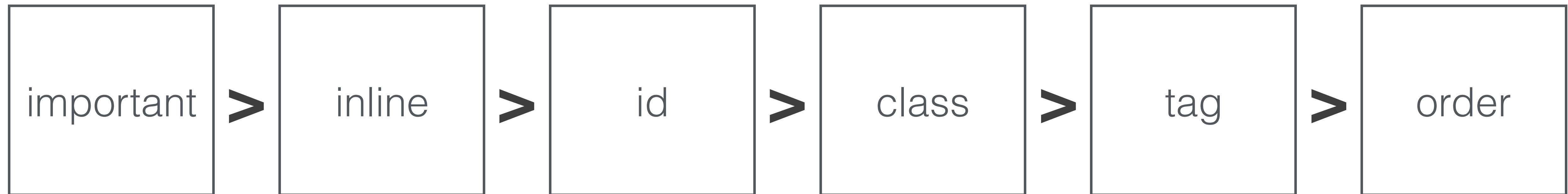


```
#profile .foo {  
  background: brown;  
}
```



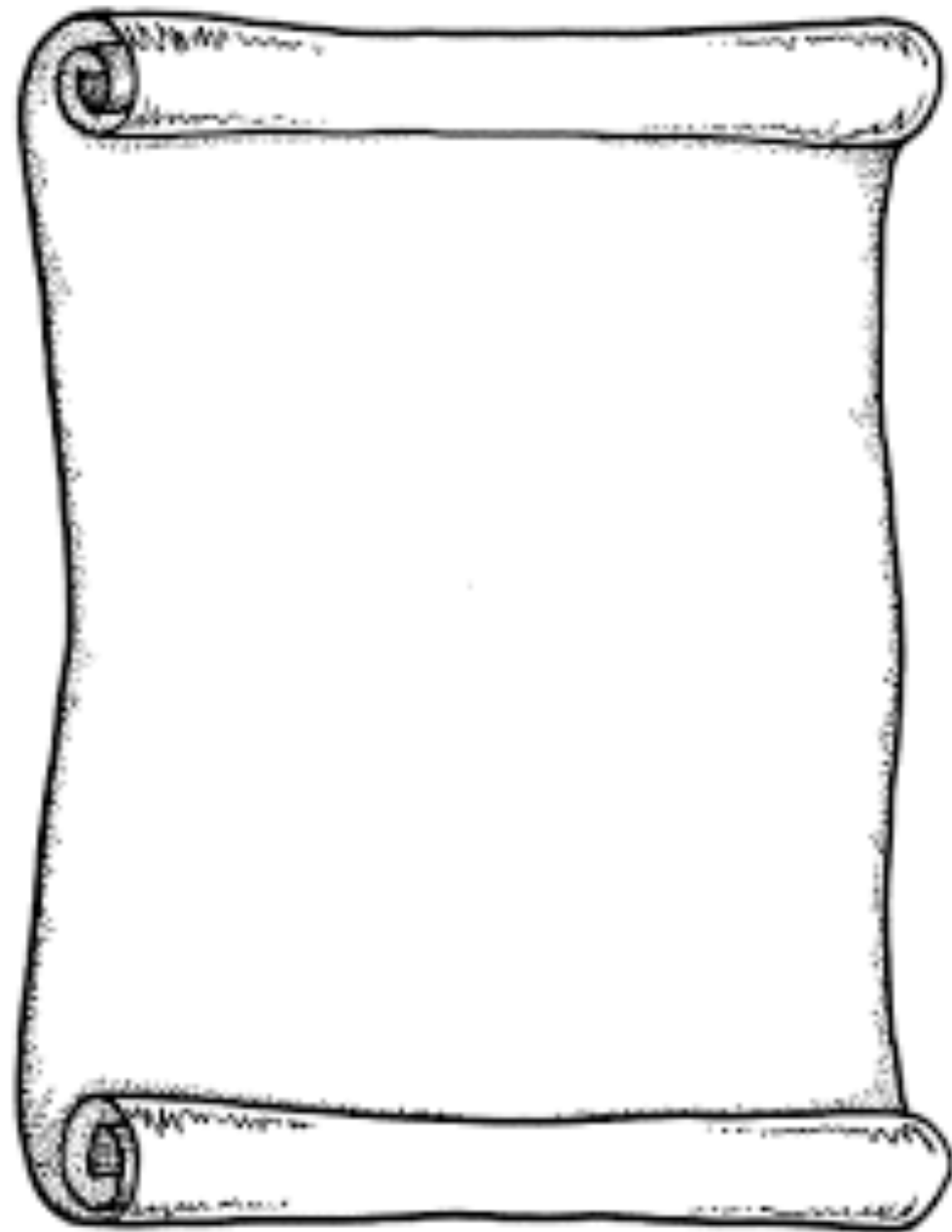
THAT WAS CSS SPECIFICITY

DECLARATION SPECIFICITY

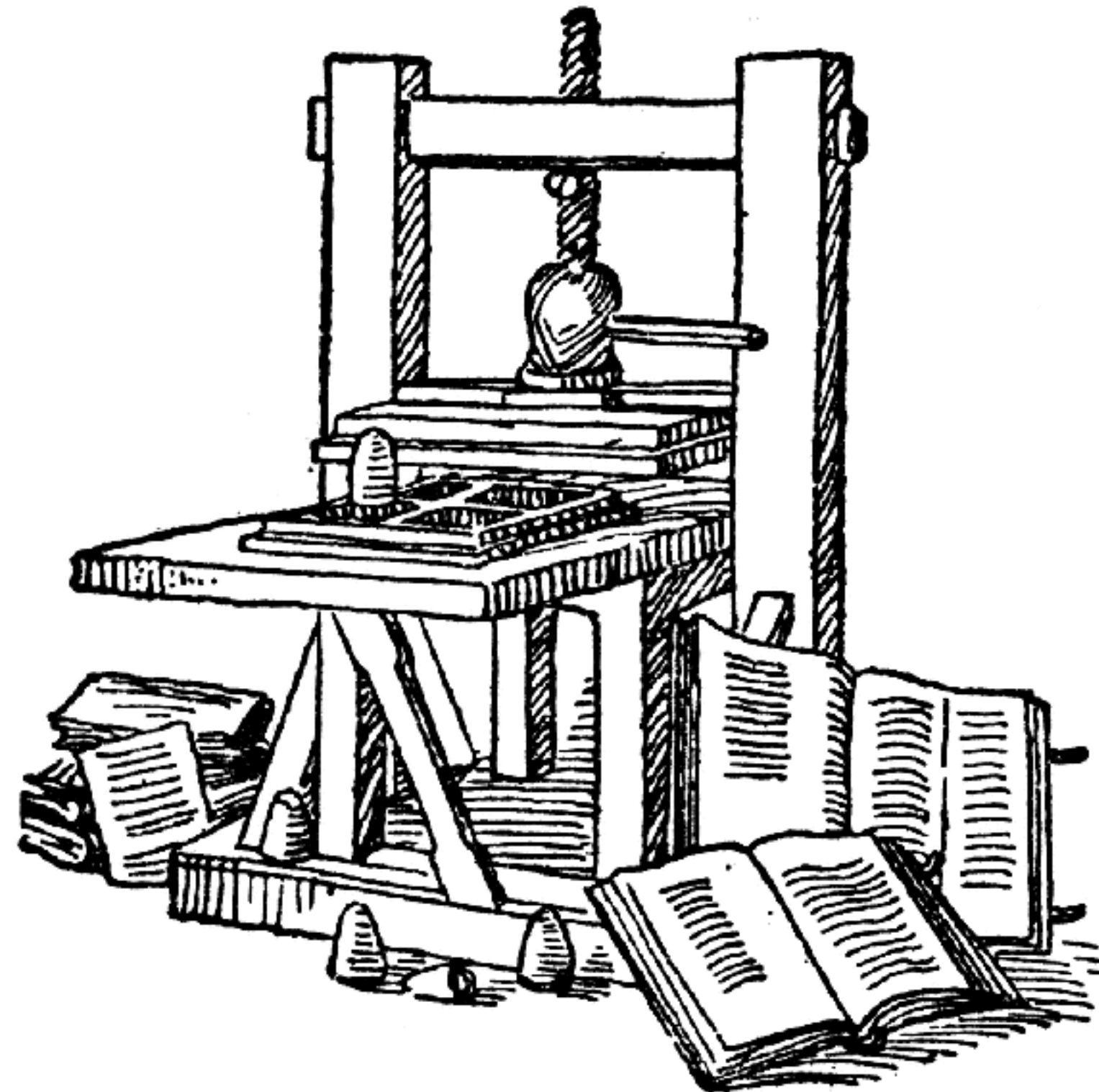


Combinators don't affect specificity!

LAYOUT



[HTTP://WWW.CLIPARTBEST.COM/CLIPART-9I4O55P6T](http://www.clipartbest.com/clipart-9I4O55P6T)



[HTTP://ETC.USF.EDU/CLIPART/44800/44880/44880_GUTEN_PRESS_LG.GIF](http://etc.usf.edu/clipart/44800/44880/44880_GUTEN_PRESS_LG.GIF)



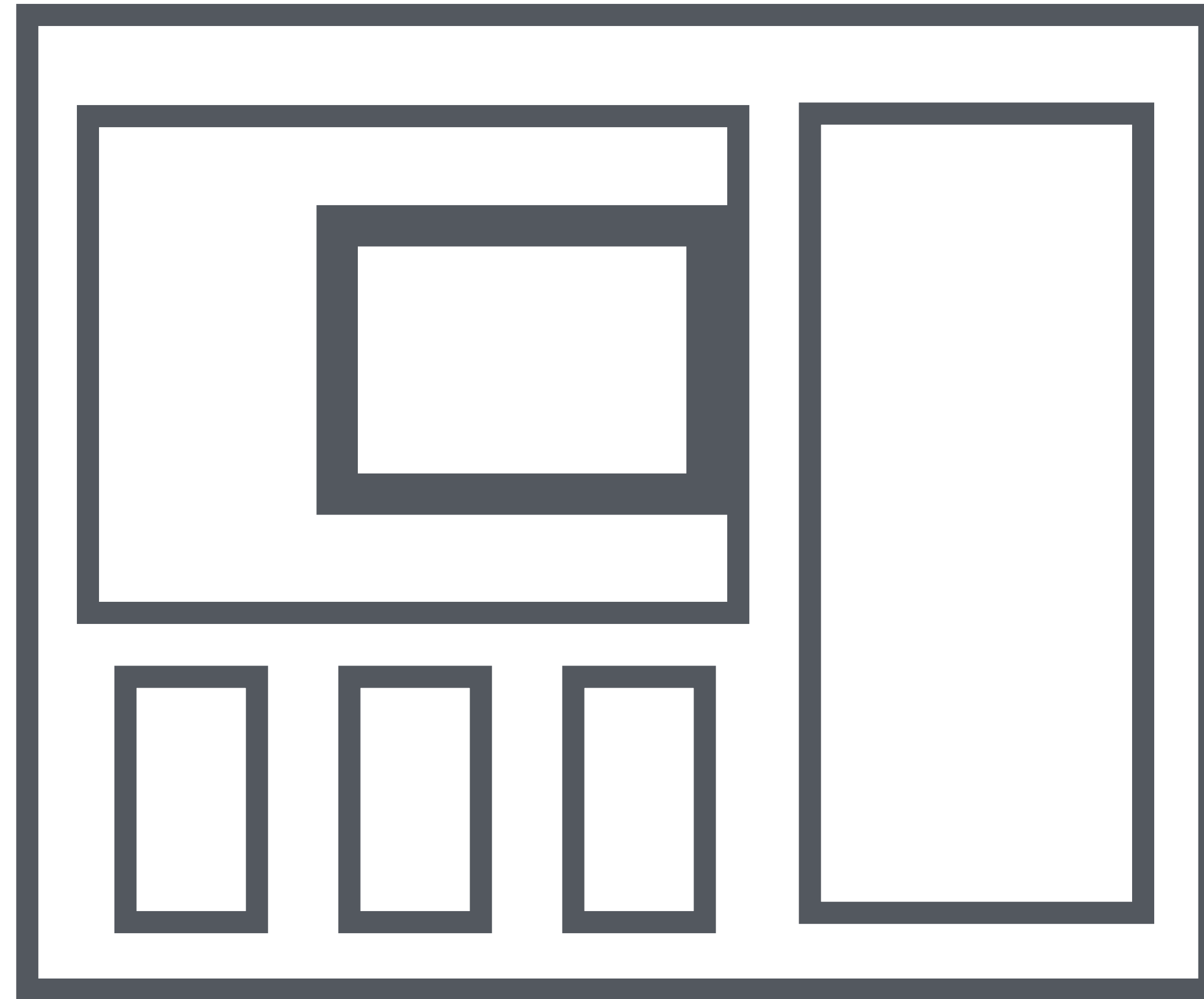
[HTTP://ASSETS.UXBOOTH.COM/UPLOADS/2009/11/MYSPACE.PNG](http://assets.uxbooth.com/uploads/2009/11/myspace.png)

THE BOX MODEL

“Everything on the DOM is a rectangle”

- JOHN AND TOM

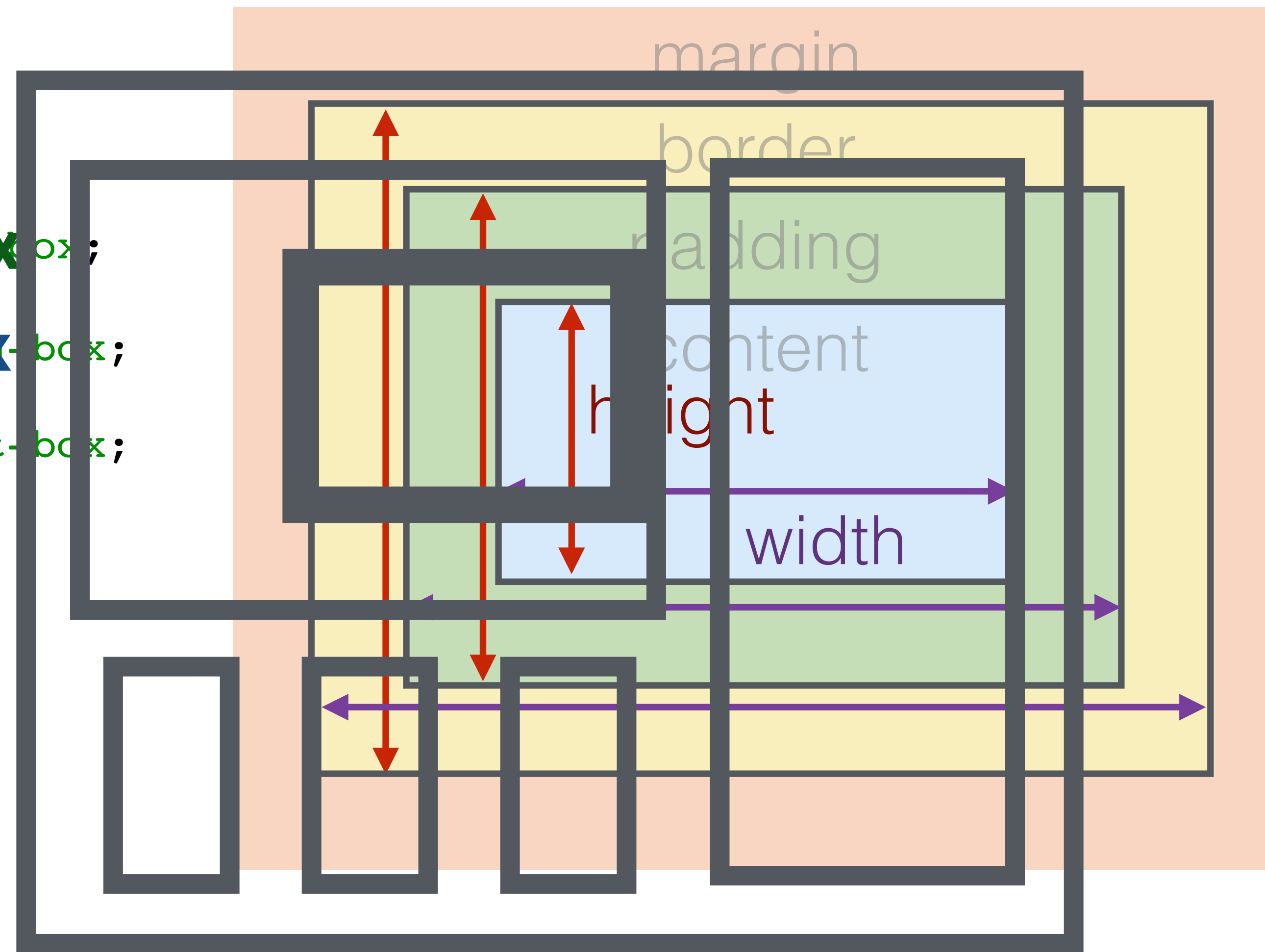
BOX MODEL



BOX MODEL

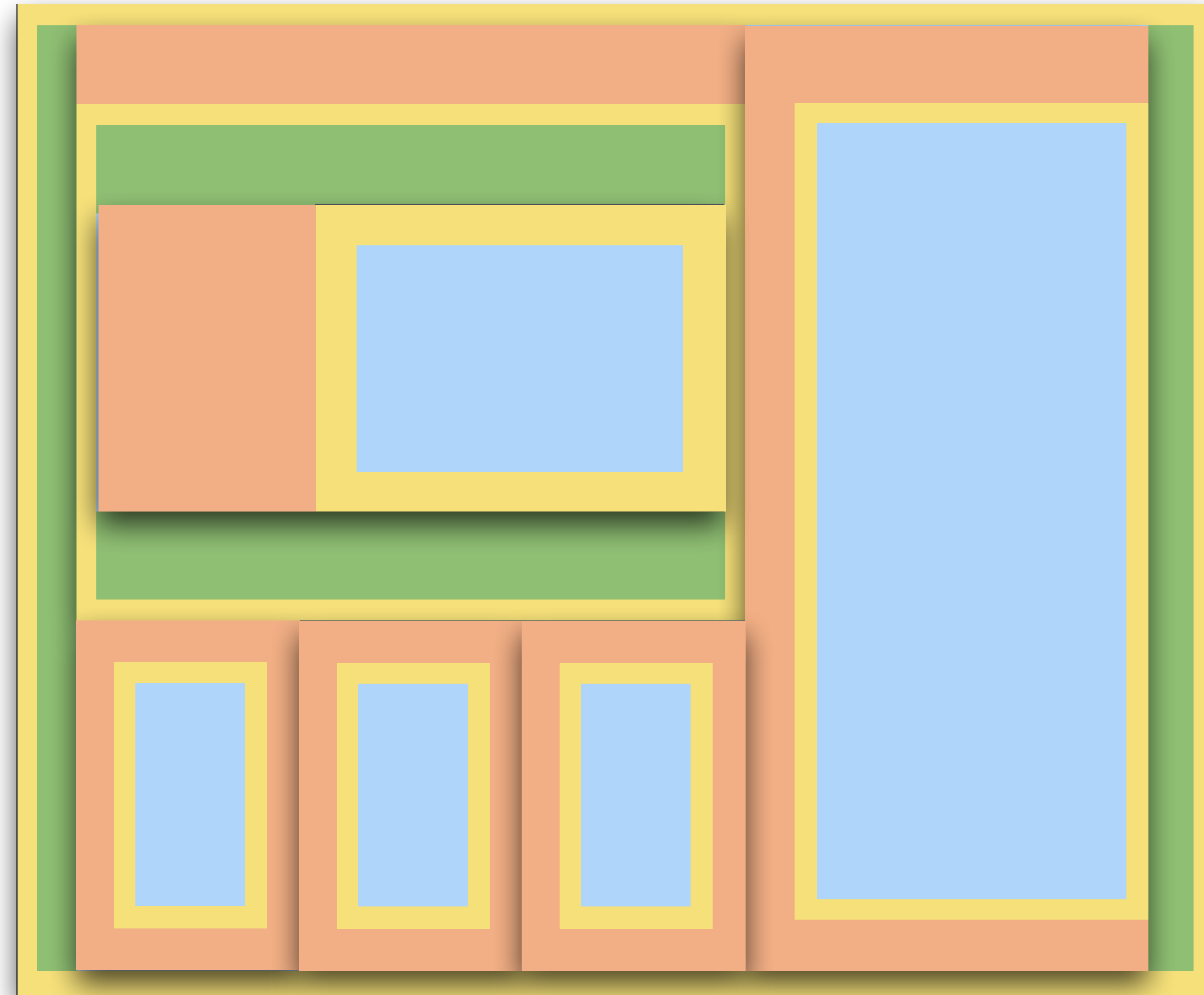
border box
padding box
content box

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



BOX MODEL

fractal!



BLOCK

vs

INLINE

vs

INLINE-BLOCK

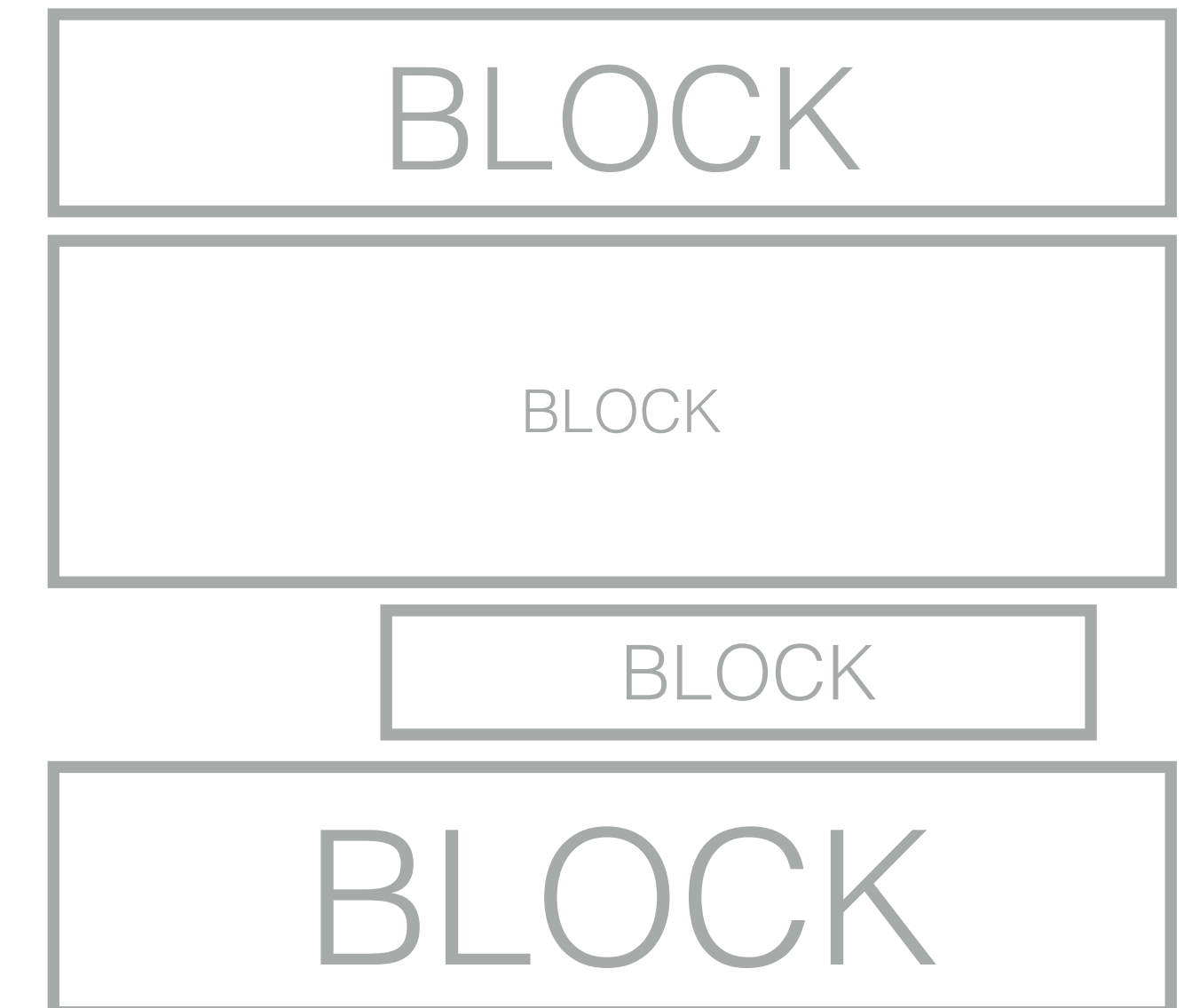
- ◉ `<div>`
- ◉ `<h1>`, `<h2>`, etc.
- ◉ `<p>`
- ◉ `<form>`
- ◉ `<header>`, `<footer>`,
`<main>`, `<section>`, `<nav>`

- ◉ `<a>`
- ◉ ``
- ◉ ``, ``

- ◉ ``
- ◉ `<input>`
- ◉ `<textarea>`

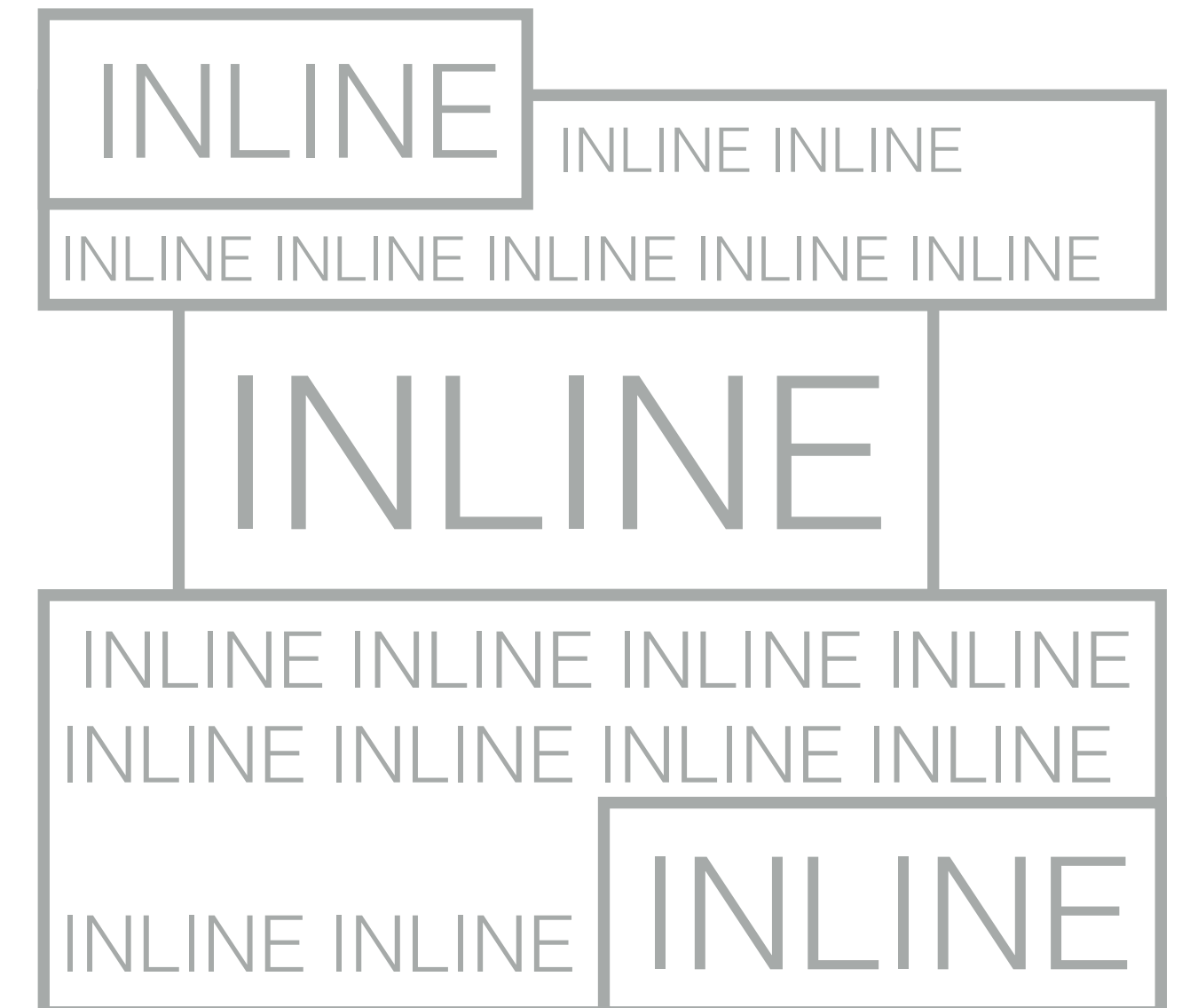
BLOCK-LEVEL

- 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



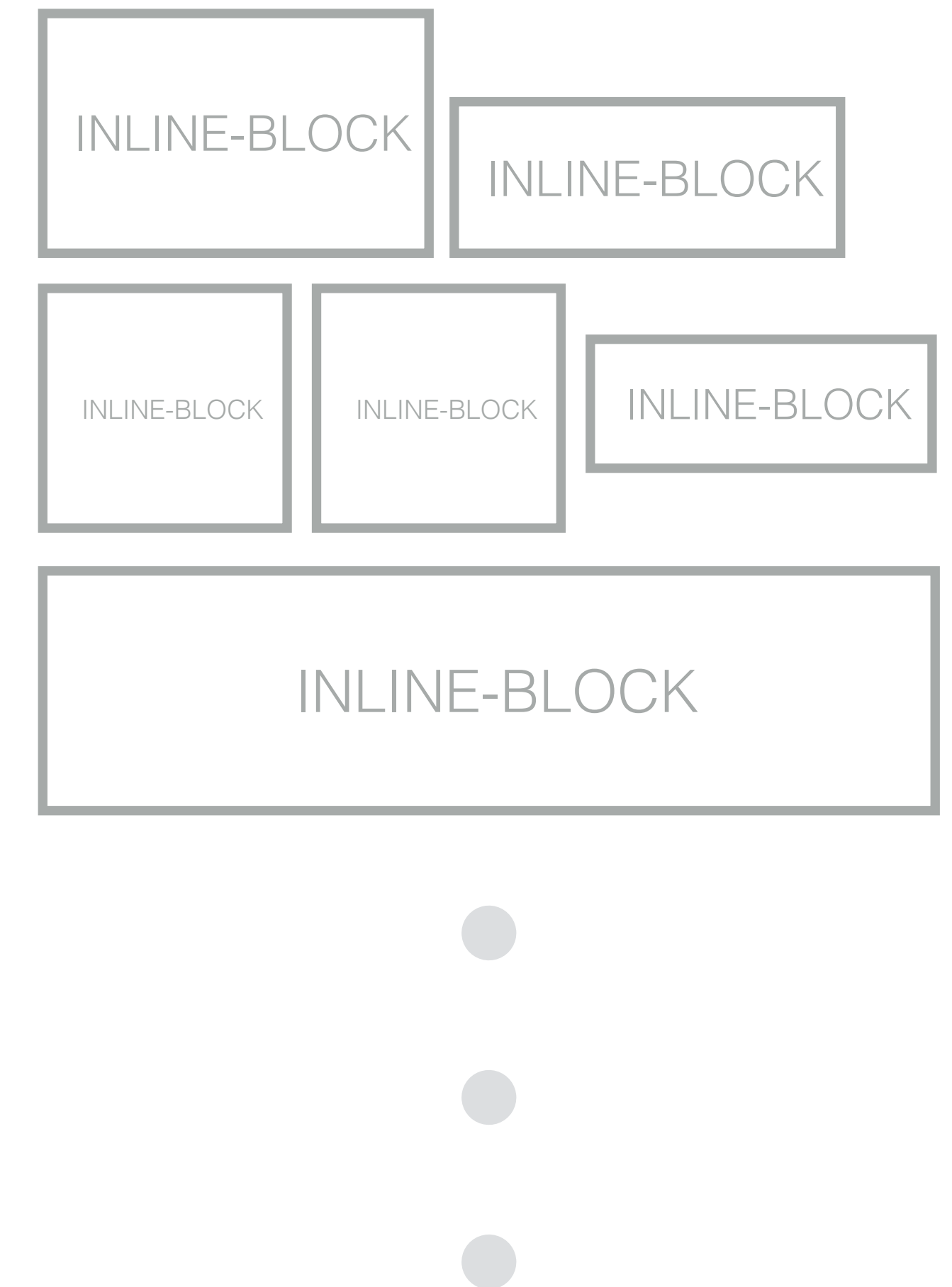
INLINE-LEVEL

- ◉ 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



FLOATS

“I understand floats.”

-Liar McPantsonfire

```

<body>
  <div>
    <div style="width:50px;">...</div>
    <div style="width:50px;">...</div>
    <div style="width:50px;">...</div>
  </div>
</body>

```



```

<body>
  <div>
    <div style="width:50px; float:right;">...</div>
    <div style="width:50px; float:right;">...</div>
    <div style="width:50px; float:right;">...</div>
  </div>
</body>

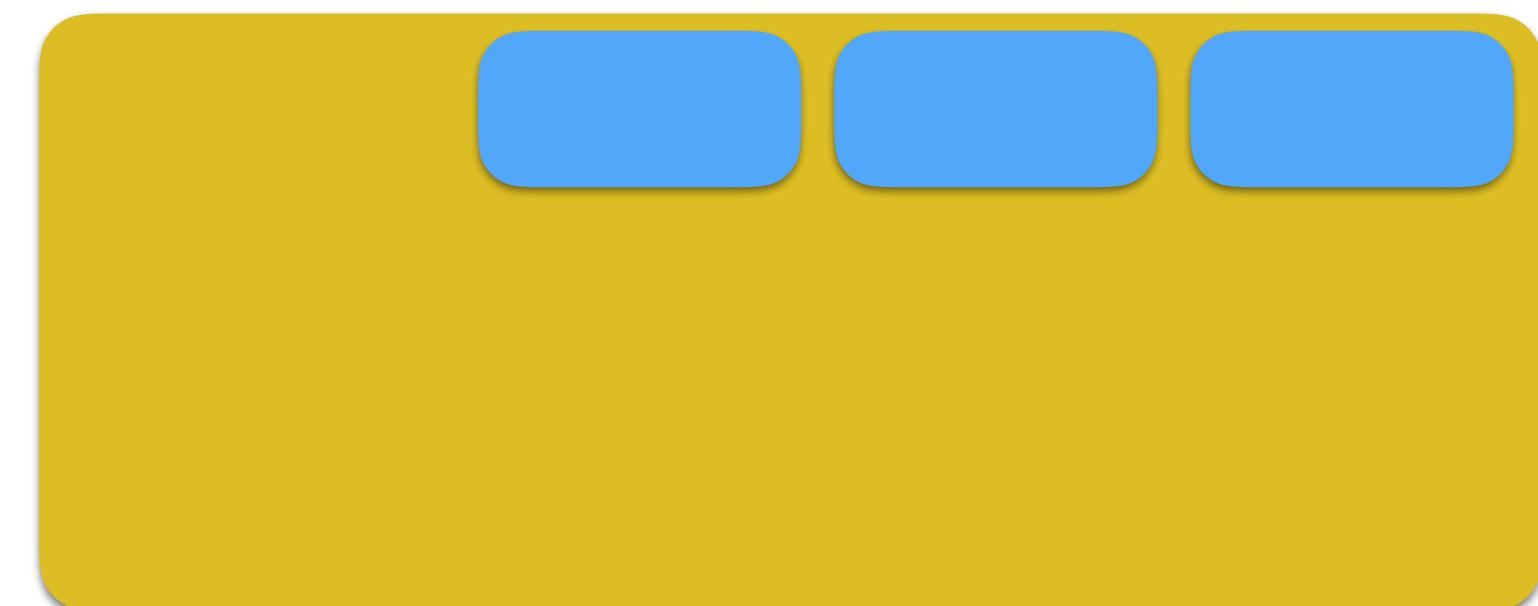
```



```

<body>
  <div style=overflow:auto;>
    <div style="width:50px; float:right;">...</div>
    <div style="width:50px; float:right;">...</div>
    <div style="width:50px; float:right;">...</div>
  </div>
</body>

```



'CLEARFIX' HACK?

- ◉ putting *overflow: auto* on the parent will work a lot of the time
- ◉ But it can have unpredictable effects, so we use something called the 'clearfix hack'
- ◉ give the parent element a class of *clearfix*
- ◉ In your CSS, write the following clearfix rule:

```
.clearfix:after {  
  content: "";  
  display: table;  
  clear: both;  
}
```



S Y N T A C T I C A L L Y
A W E S O M E
S T Y E E
S H E E T S
Y A E T
D S
I
N
G

SCSS

- **nesting**
- **variables**
- **loops**
- **functions**
- **modules**

HOW DOES SCSS WORK?

SCSS COMPILES TO CSS

NESTING

SCSS

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

CSS

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

VARIABLES

SCSS

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

CSS

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

LOOPS

SCSS

```
@for $i from 1 through 3 {  
  size#{$i} {  
    width: $i * 10px;  
  }  
}
```

CSS

```
size1 {  
  width: 10px;  
}  
size2 {  
  width: 20px;  
}  
size3 {  
  width: 30px;  
}
```

MIXINS

SCSS

```
@mixin border-radius ($r) {  
    -webkit-border-radius: $r;  
    -moz-border-radius: $r;  
    border-radius: $r;  
}  
  
.thing {  
    @include border-radius(10px);  
}
```

CSS

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

MODULES

SCSS

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

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; }  
/* ... */
```