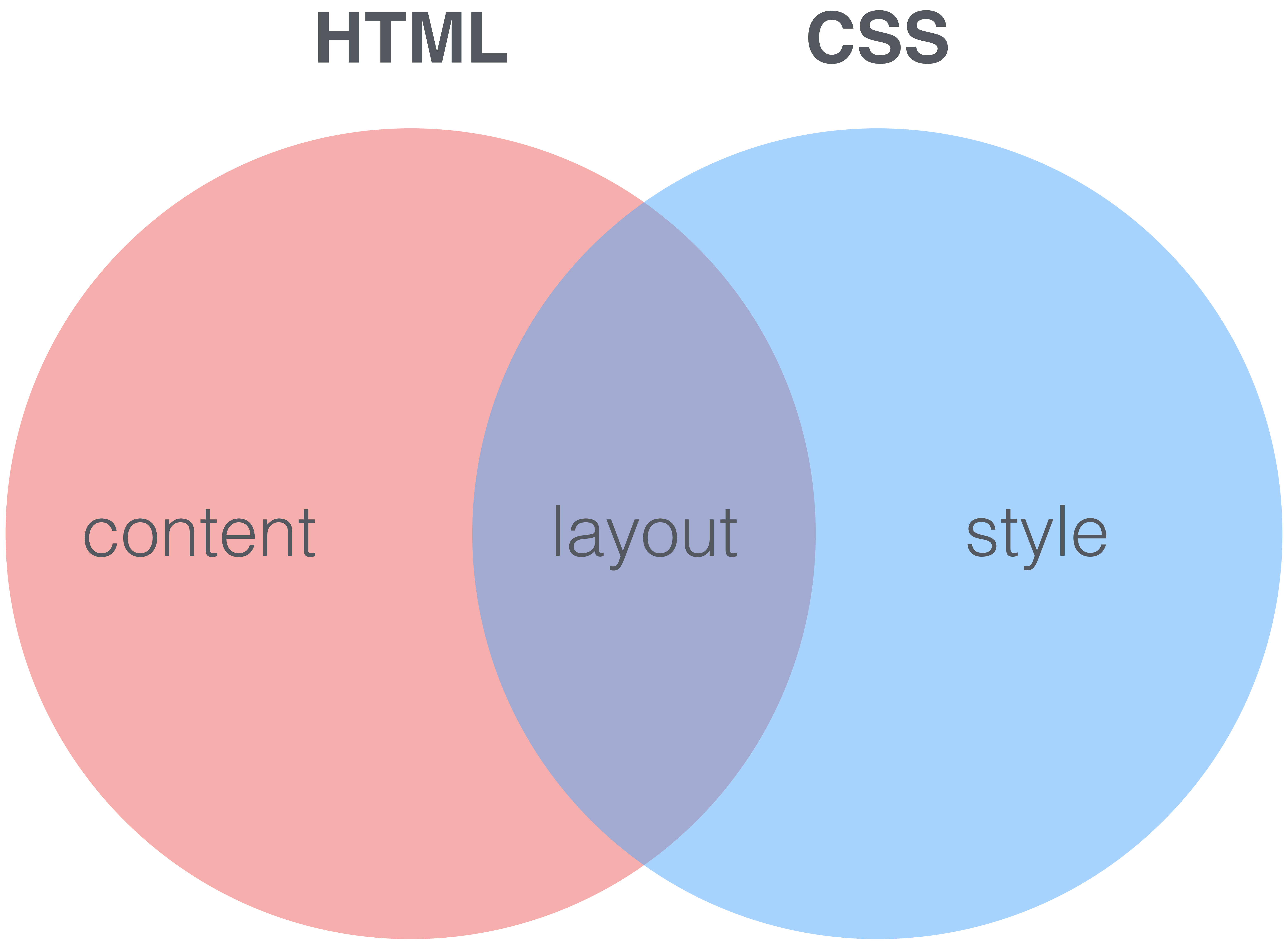
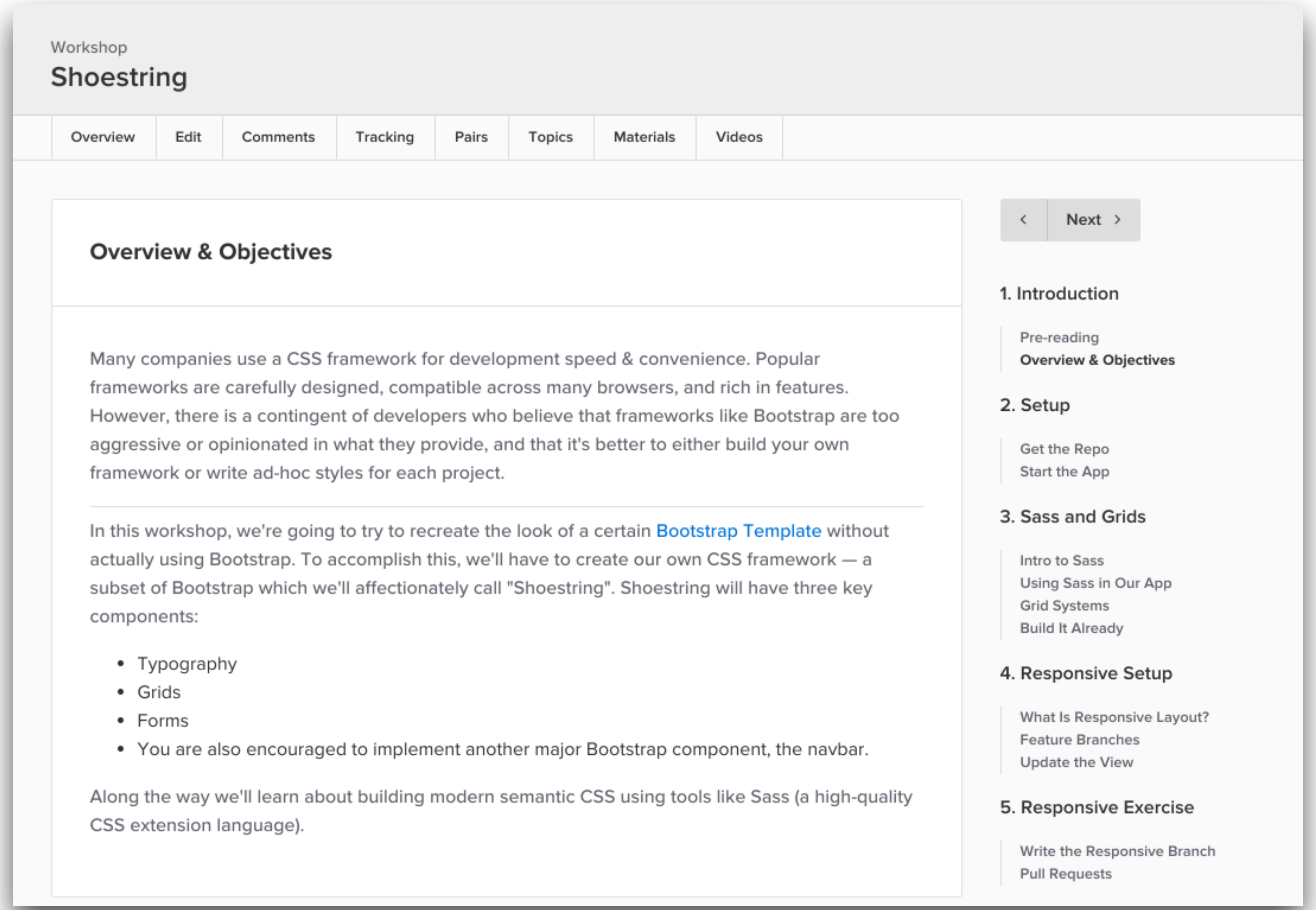
HTML & CSS

Layout laid out


HTML
CSS
content
layout
style

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:

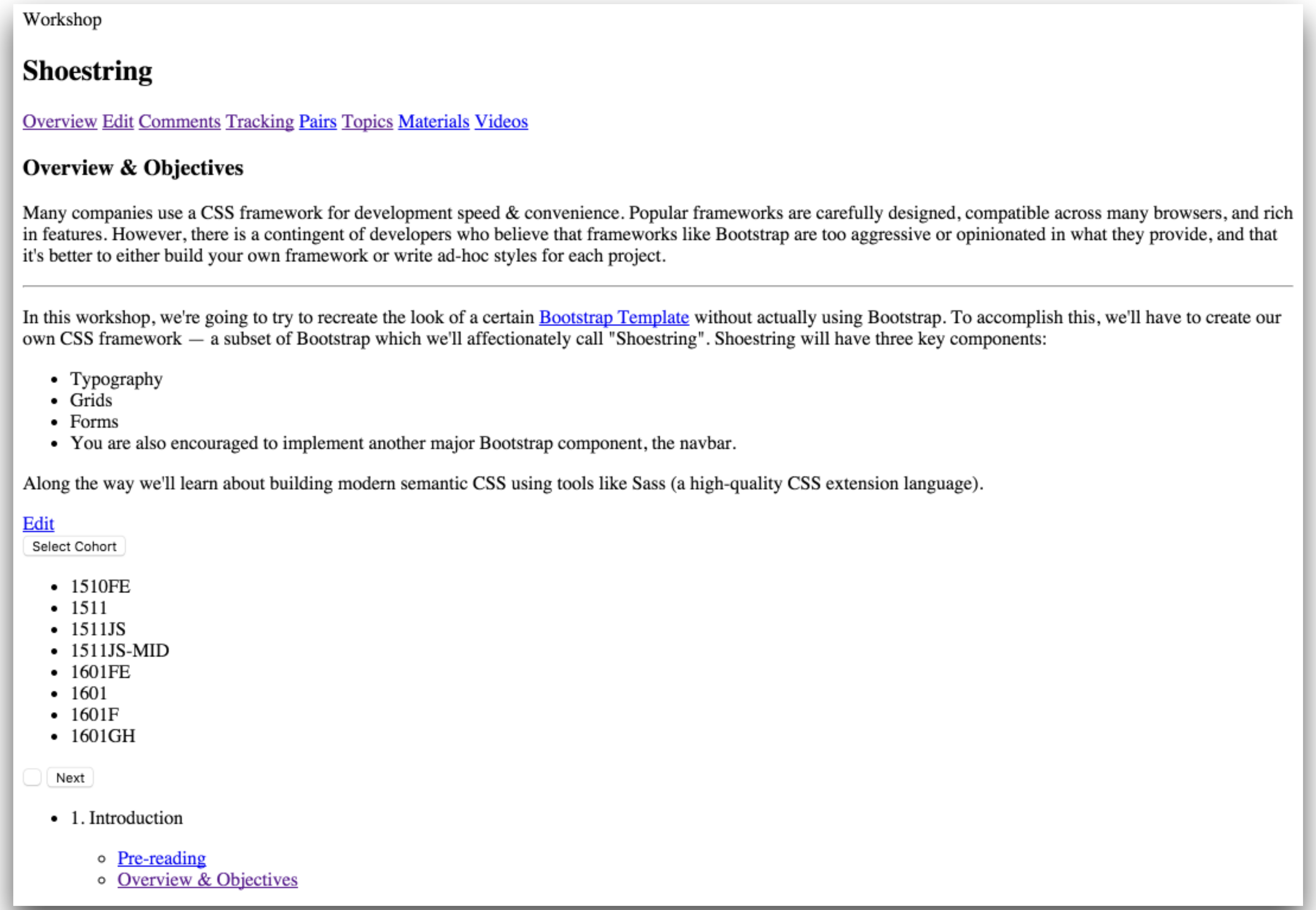
- Typography
- Grids
- Forms
- You are also encouraged to implement another major Bootstrap component, the navbar.

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

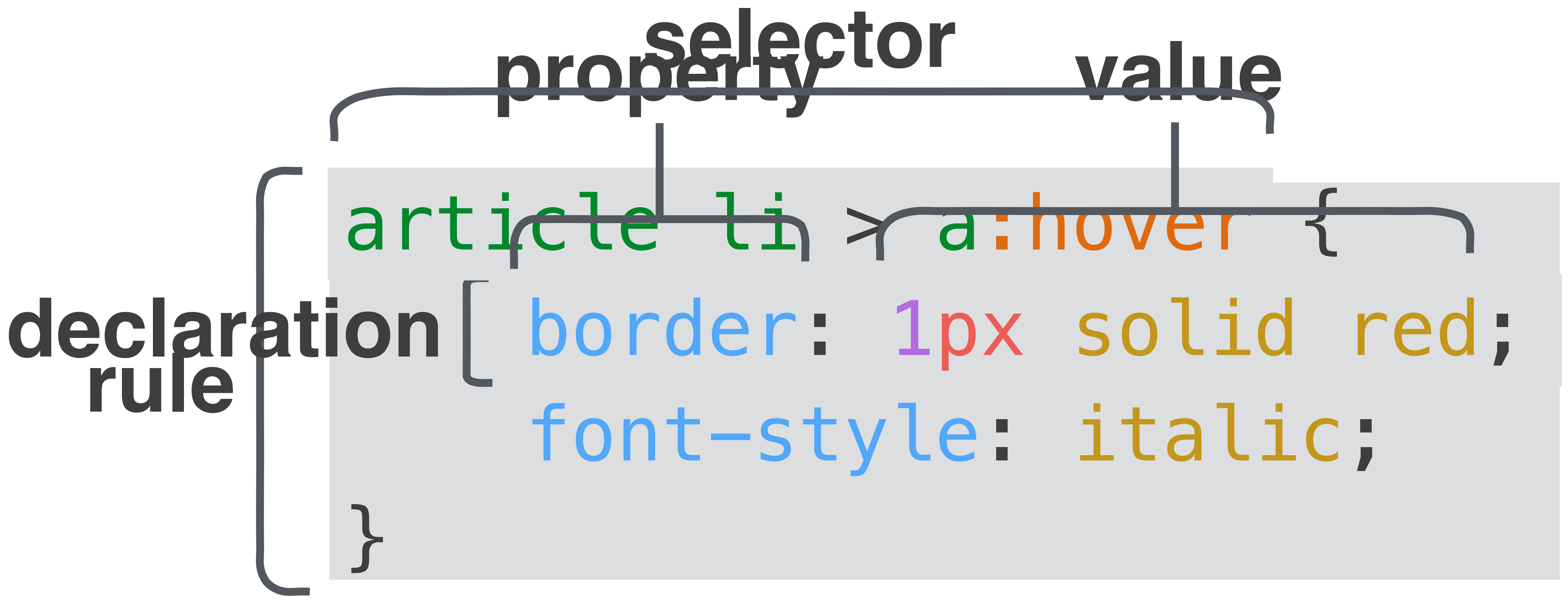
Select Cohort

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

Next

- 1. Introduction
  - Pre-reading
  - Overview & Objectives

TERMS



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

RULE EXAMPLE

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

apply **these** styles

to any elements matching **this** selector

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

SELECTORS

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

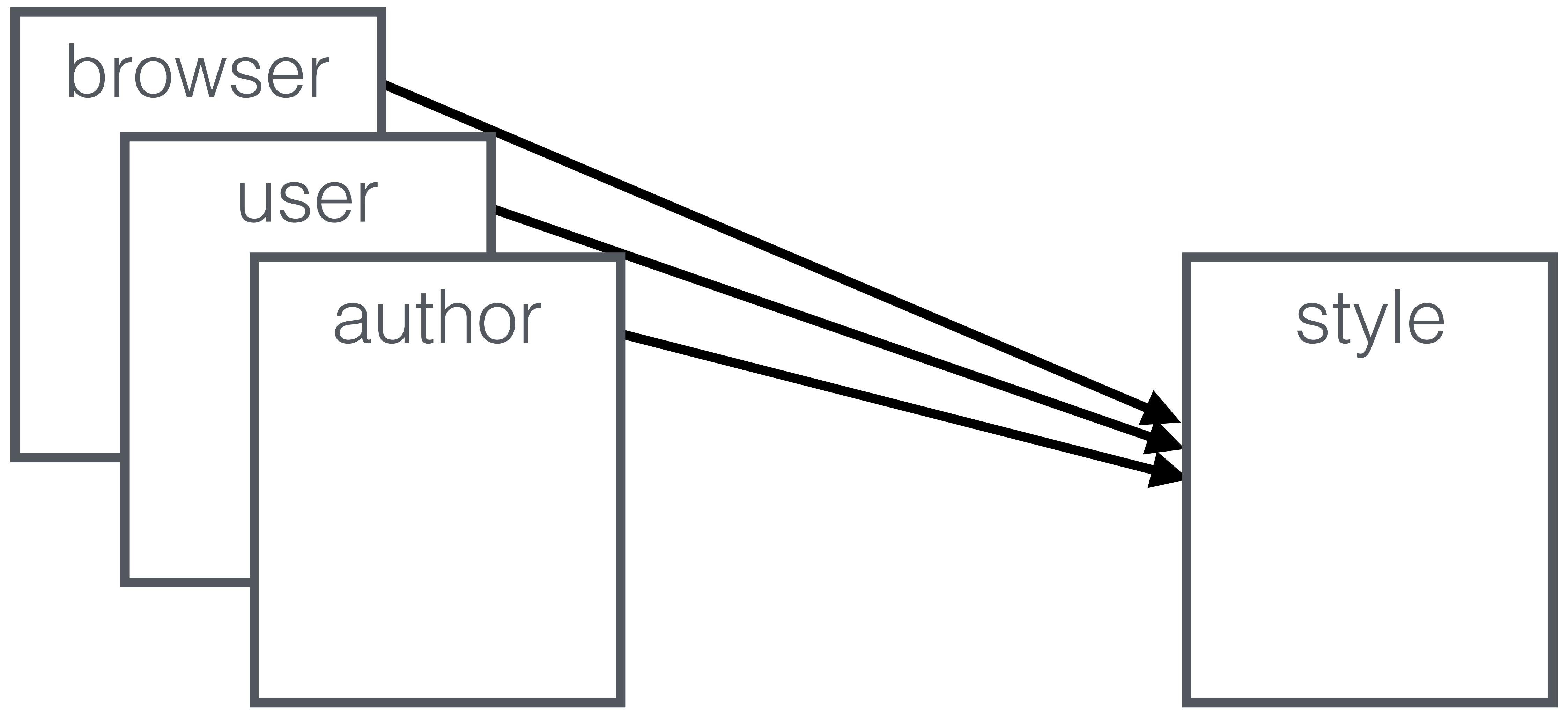
BEWARE!

`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`

CASCADING STYLE SHEETS

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>
```

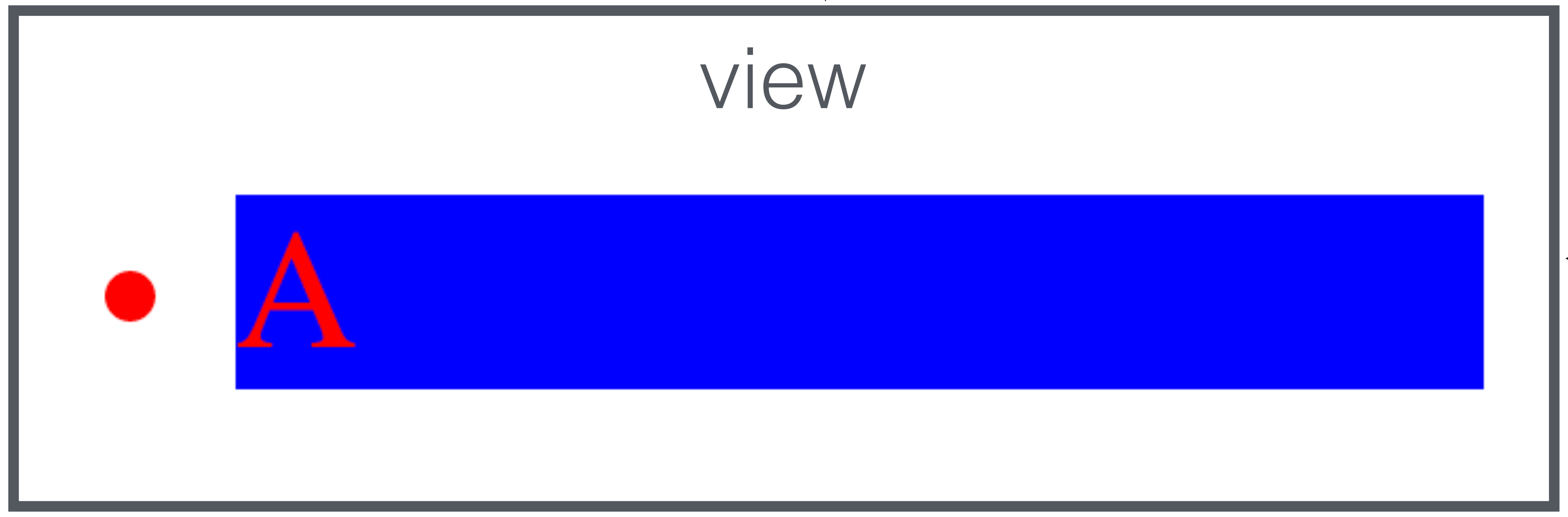
styles-A.css

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

styles-B.css

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

view



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?

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

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

🟥

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

🟦

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

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

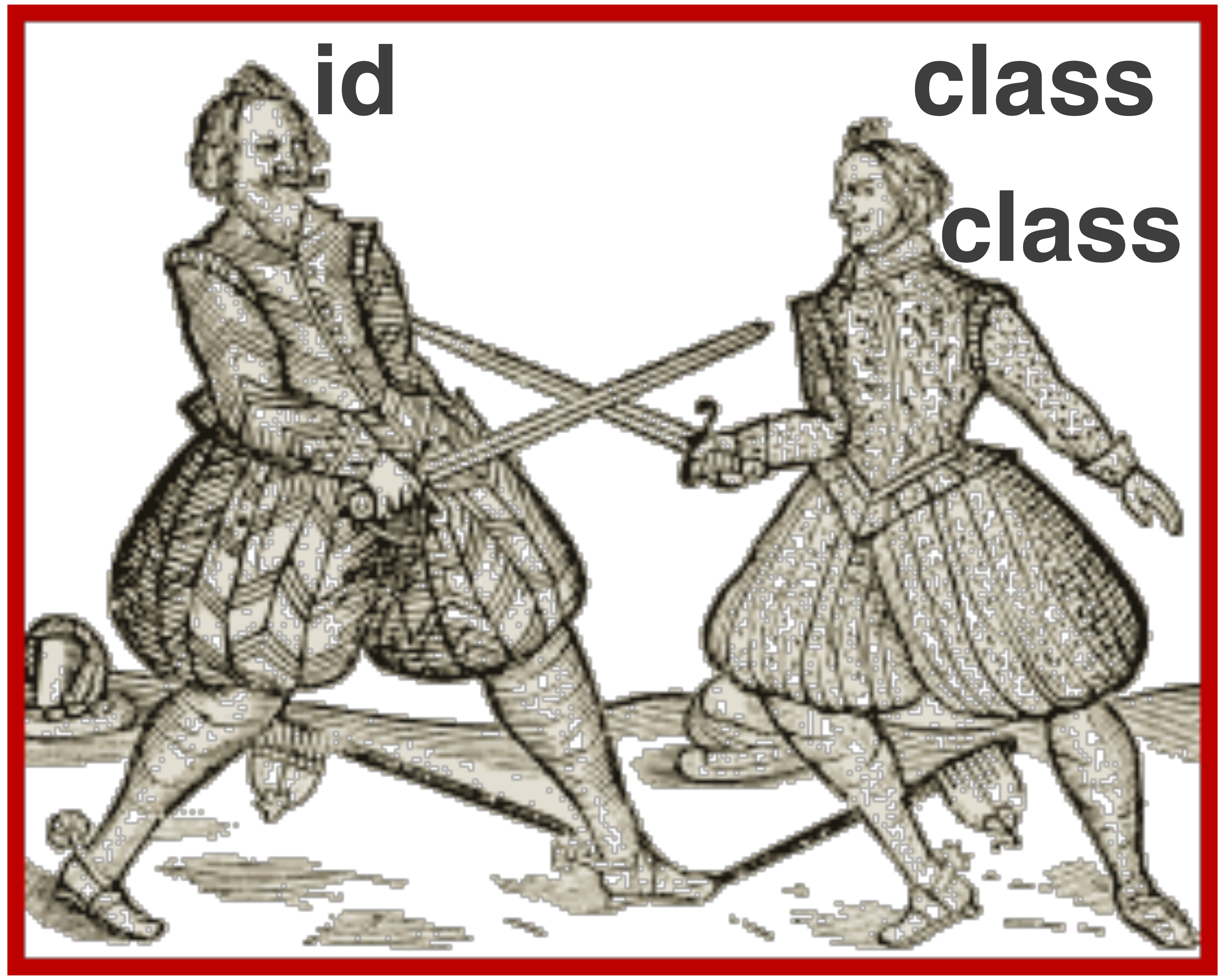
🟥

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

🟩

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

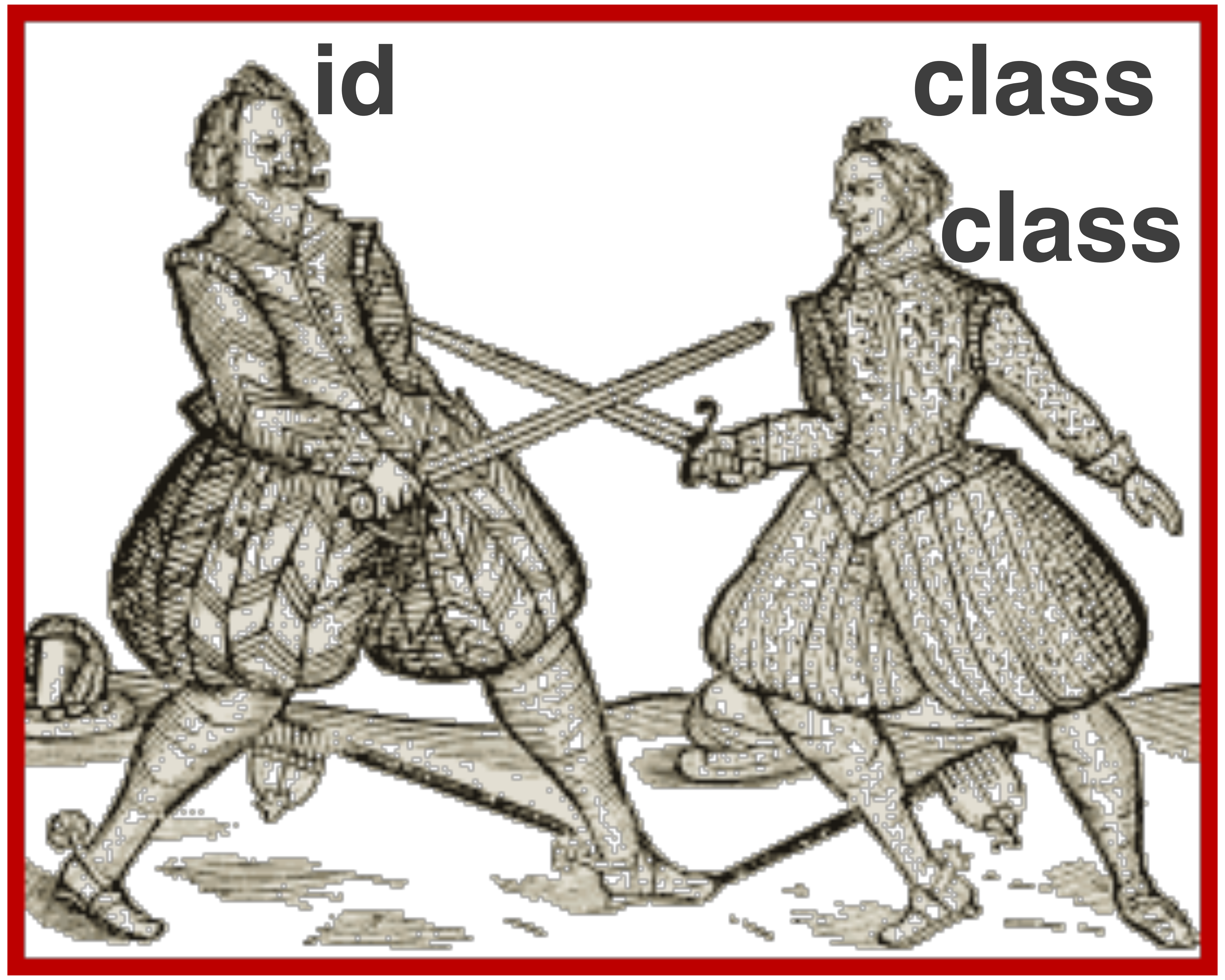
```css
#thing {
  background: blue;
}
```



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

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

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



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

CSS
Global HD