

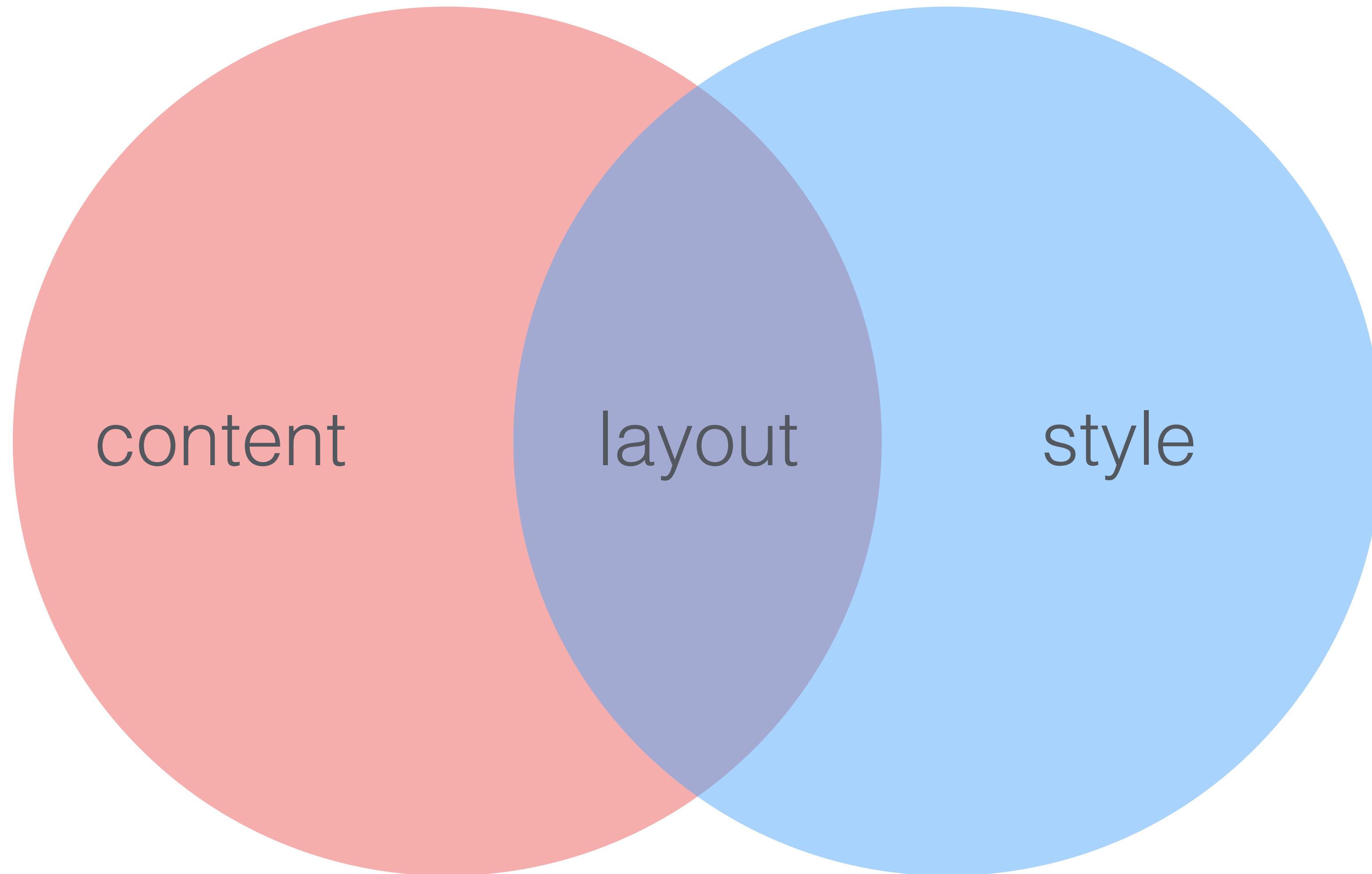
# HTML & CSS

---

*Layout laid out*

# HTML

# CSS



# 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](#)

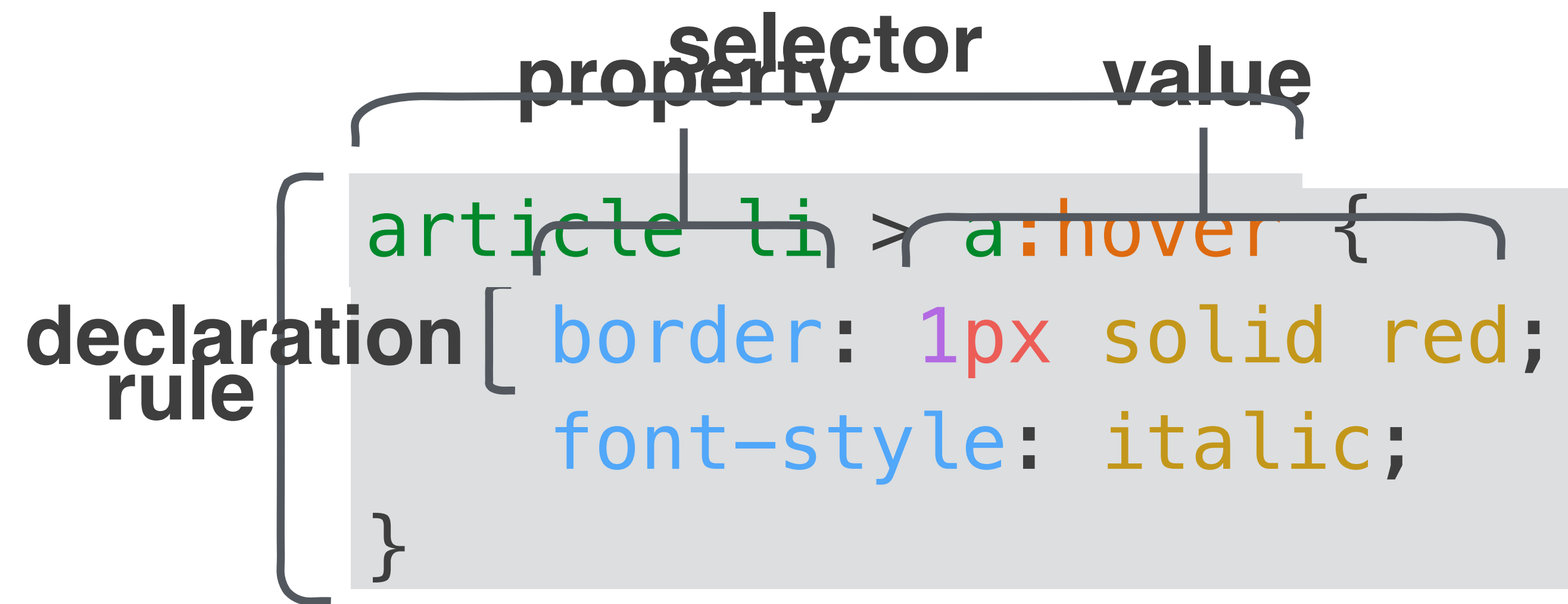
Select Cohort

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

☐ Next

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

# TERMS



# 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>
*	*

# 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



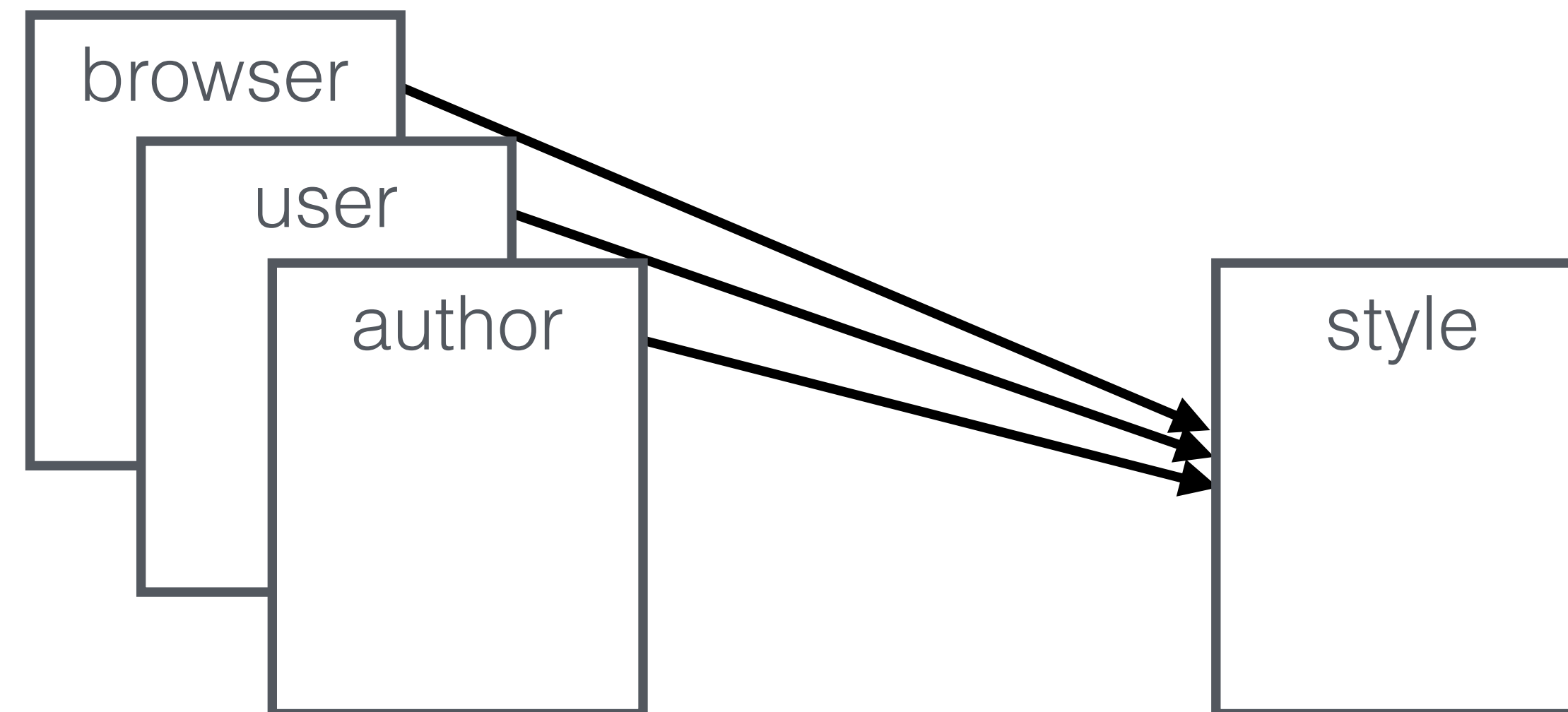
# CASCADING

**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;  
}
```



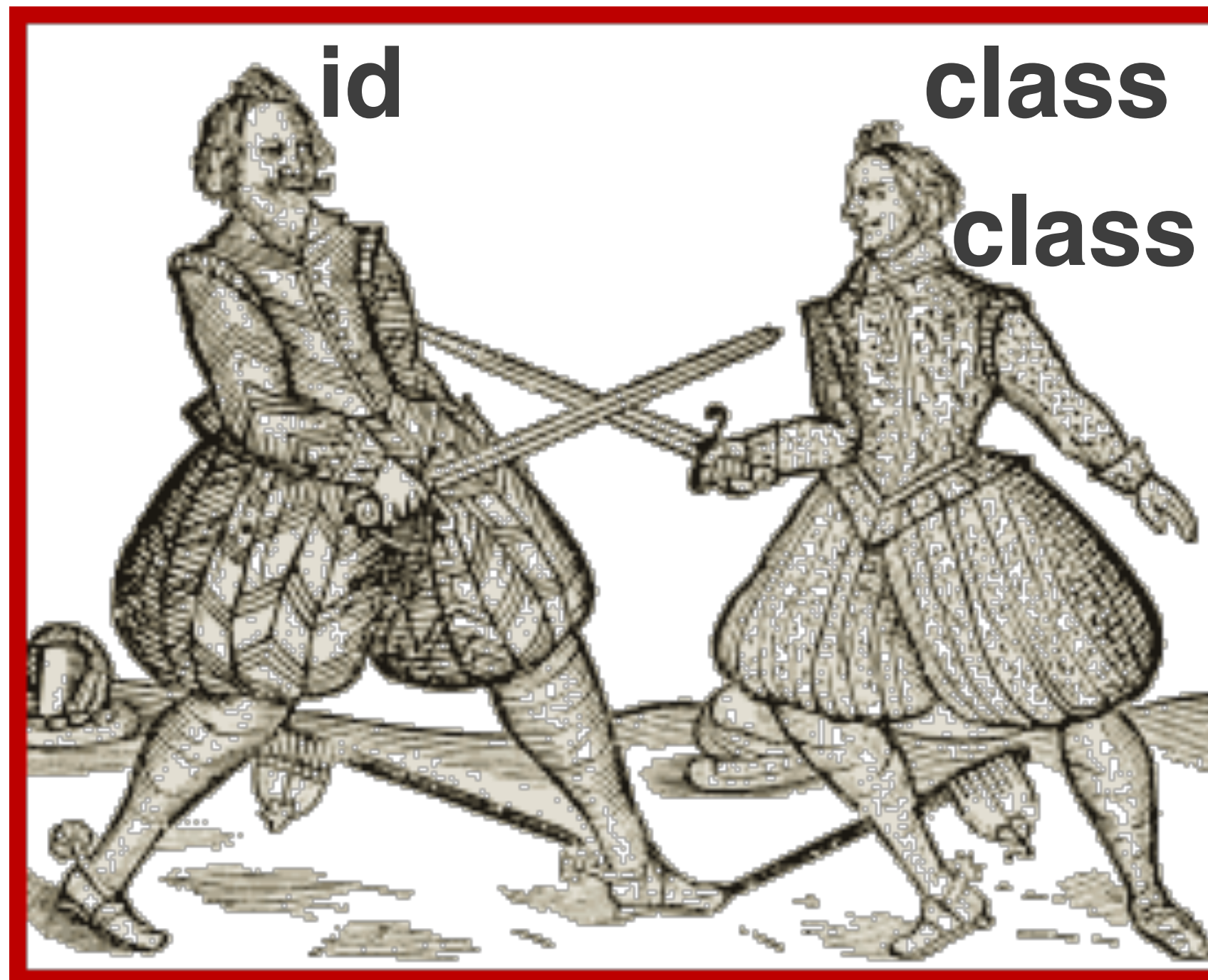
```
.foo {  
  background: green;  
}
```





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

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

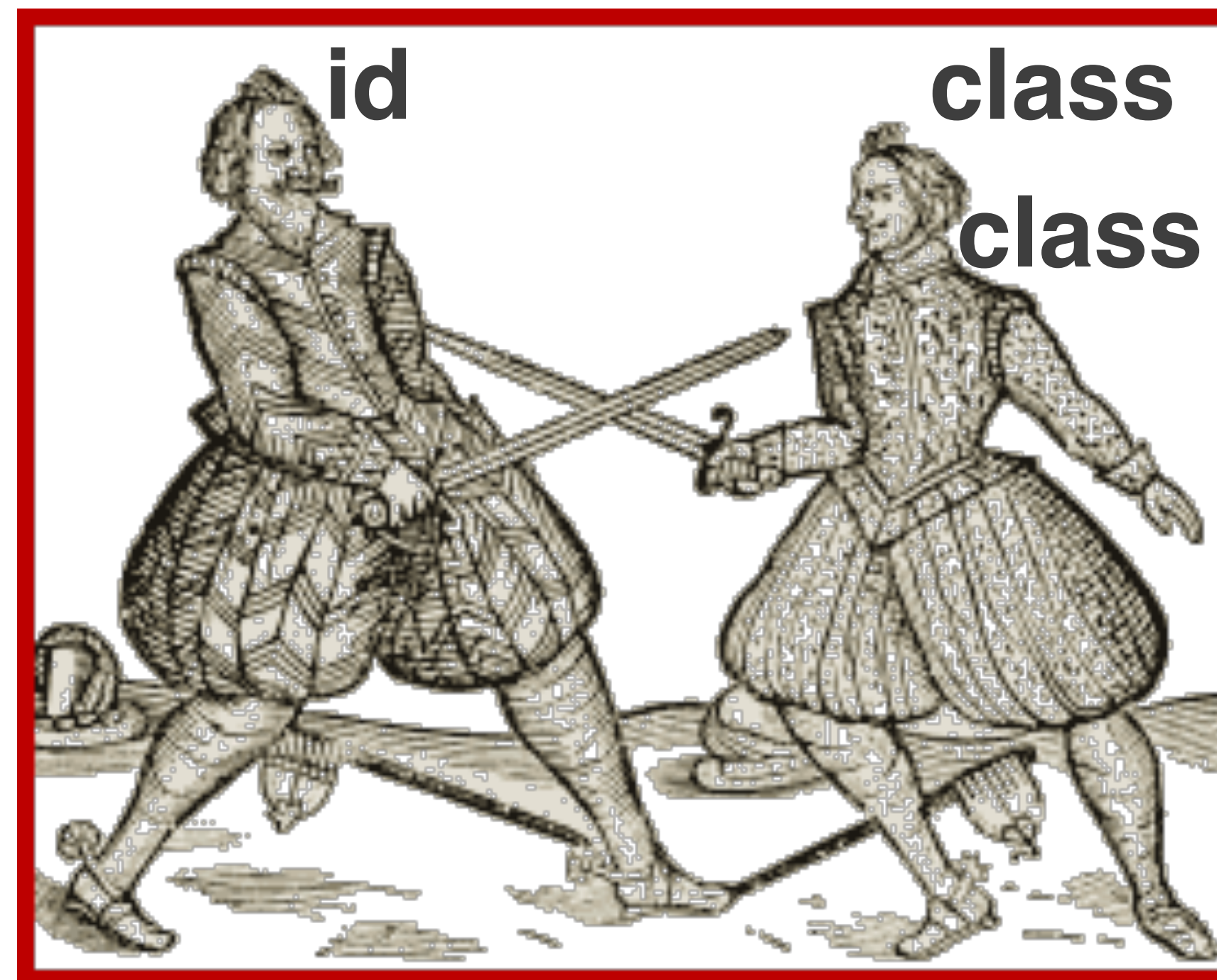
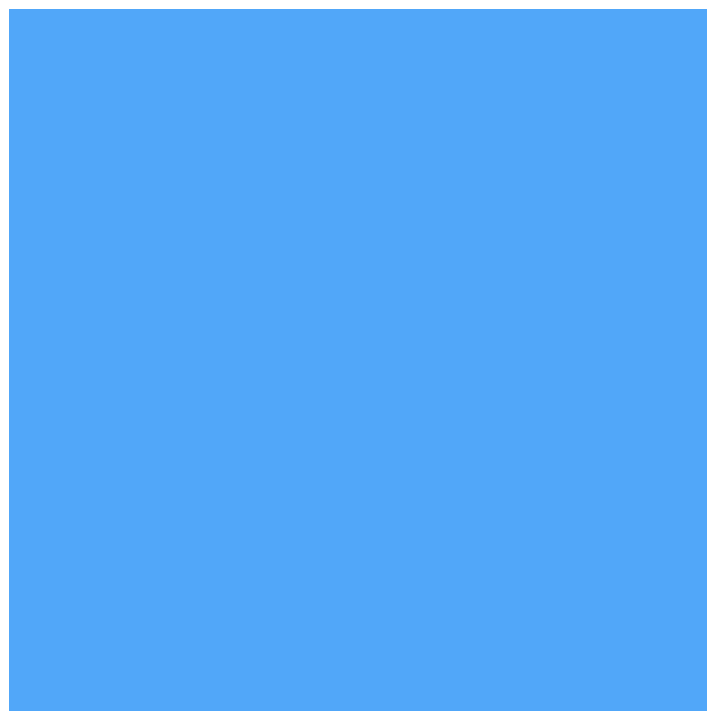


```
.foo.bar {  
  background: green;  
}
```



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

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



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









# FLEXBOX

# What is CSS Flexbox?

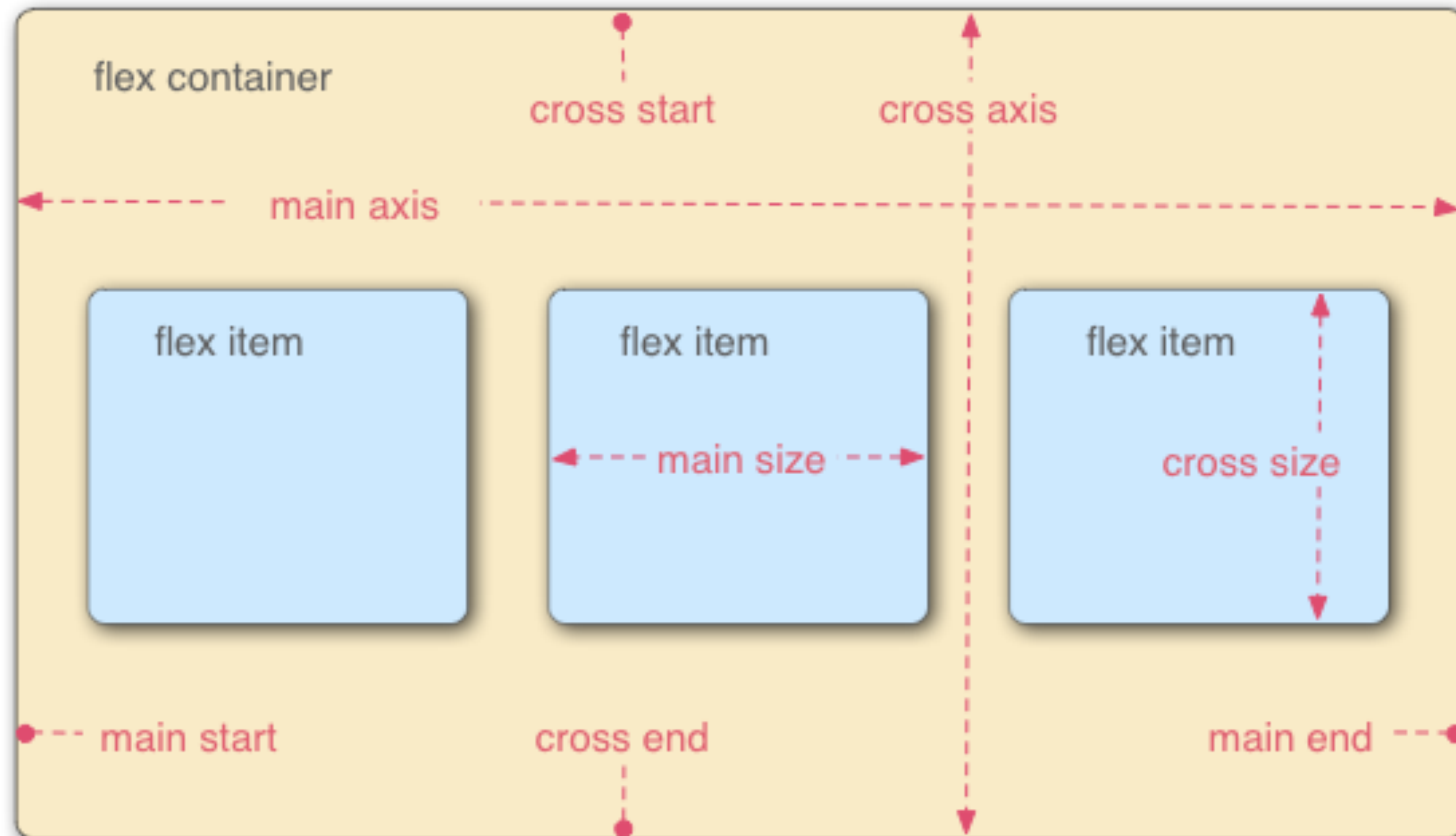
- **The Flexbox Layout module - more efficient way to**
  - lay out, align and distribute space among items in a container
  - even when the items' sizes are unknown and/or dynamic (thus the word "flex").

# Can I use it?

[caniuse.com](https://caniuse.com)

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
8			45					<sup>1</sup> 4.3	
9			46					4.4	
<sup>2 4</sup> 10		43	47			8.4		4.4.4	
<sup>4</sup> 11	13	44	48	9	34	9.2	8	47	47
	14	45	49	9.1	35	9.3			
		46	50		36				
		47	51						

# The Container



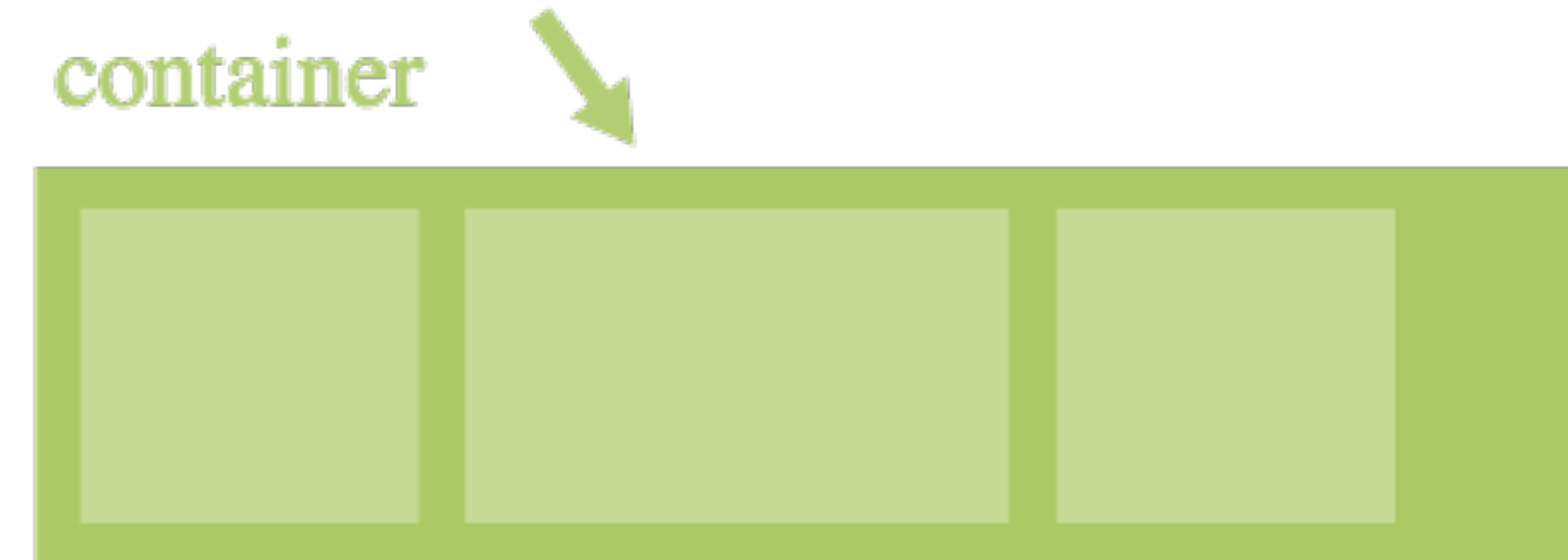
# Flexbox Container: display

- This defines a flex container; It enables a flex context for all its direct children.

```
.container {  
  display: flex; /* or inline-flex */  
}
```

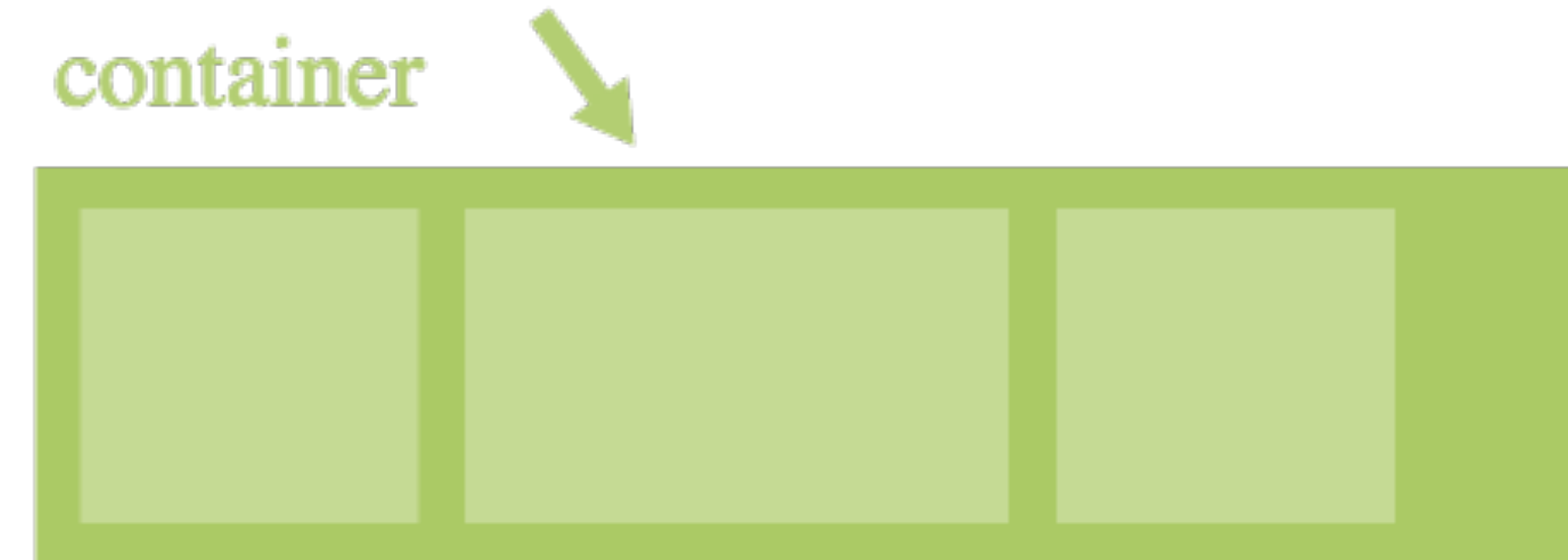
# Flexbox Container

- Flexbox gives the container the ability to alter its items dimensions (and order) to best fill the available space.



# Flexbox Container

- A flex container expands flexible items to fill free space, or shrinks them to prevent overflow.





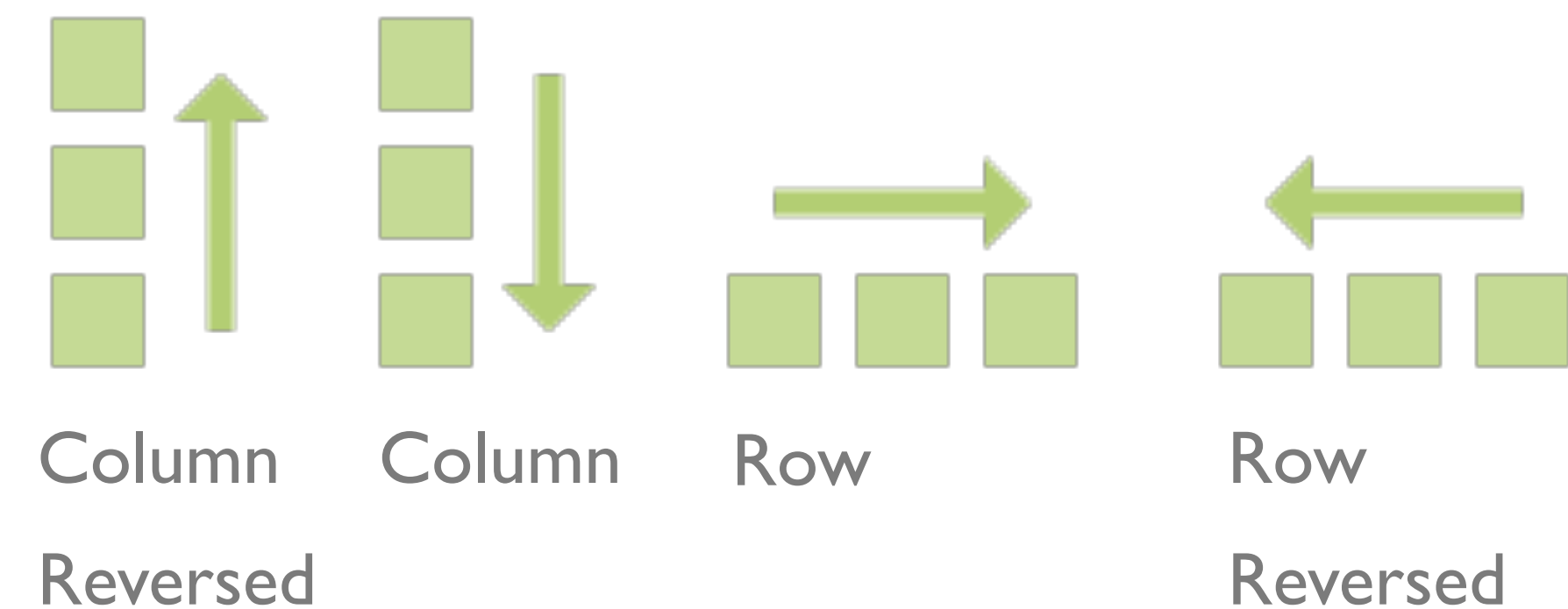
# Flexbox Container: flex-direction

- Flexbox is (aside from optional wrapping) a single-direction layout concept. Think of flex items as primarily laying out either in horizontal rows or vertical columns.
- “main axis” and “cross axis”



# Flexbox Container: Direction

- Think of flex items as primarily laying out either in horizontal rows or vertical columns.



```
.container {  
  flex-direction: row | row-reverse | column | column-reverse;  
}
```

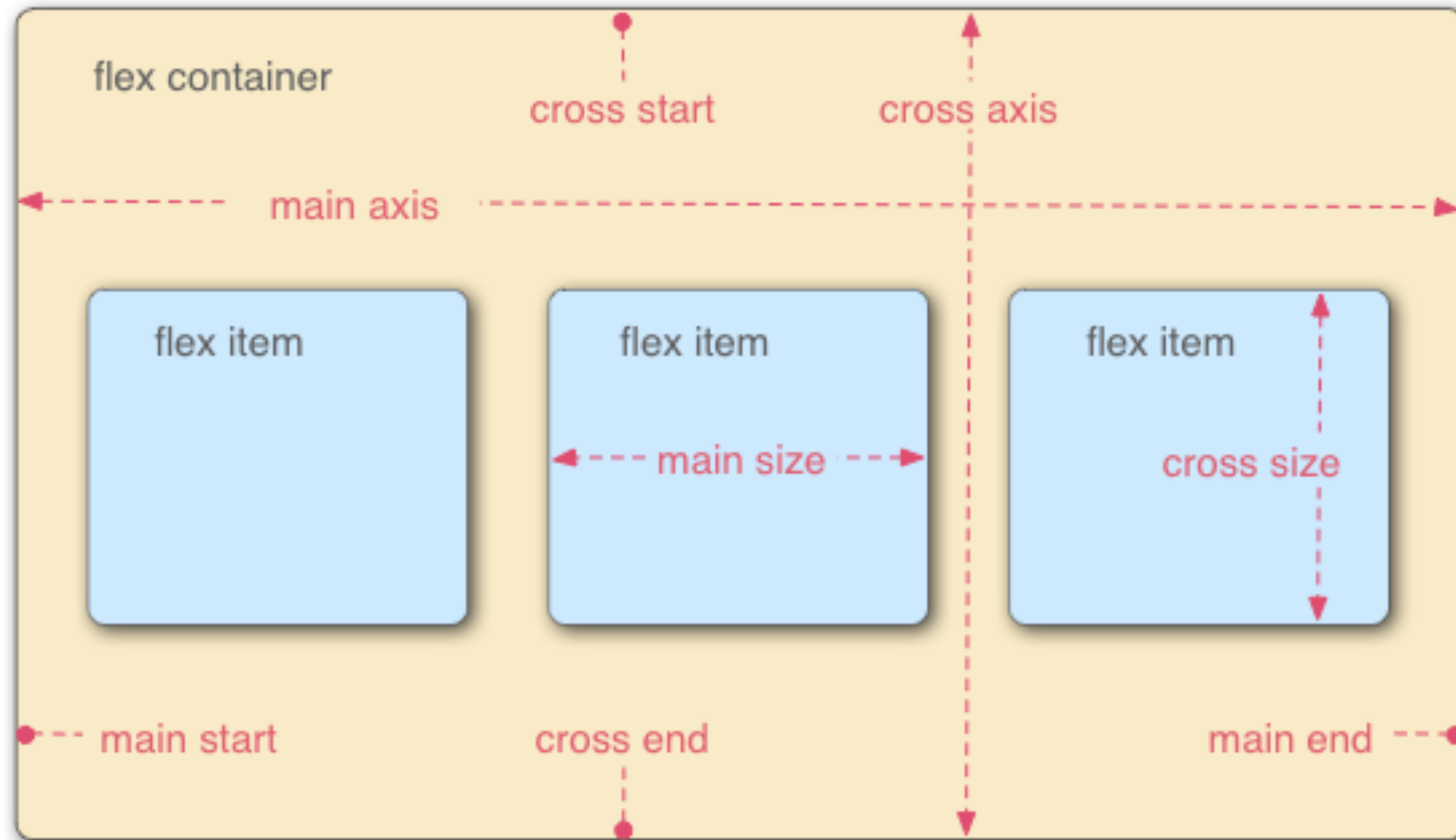
# Flexbox Container: wrap

- Items will all try to fit onto one line. Items can wrap as needed with this property. Direction also plays a role here, determining the direction new lines are stacked in.



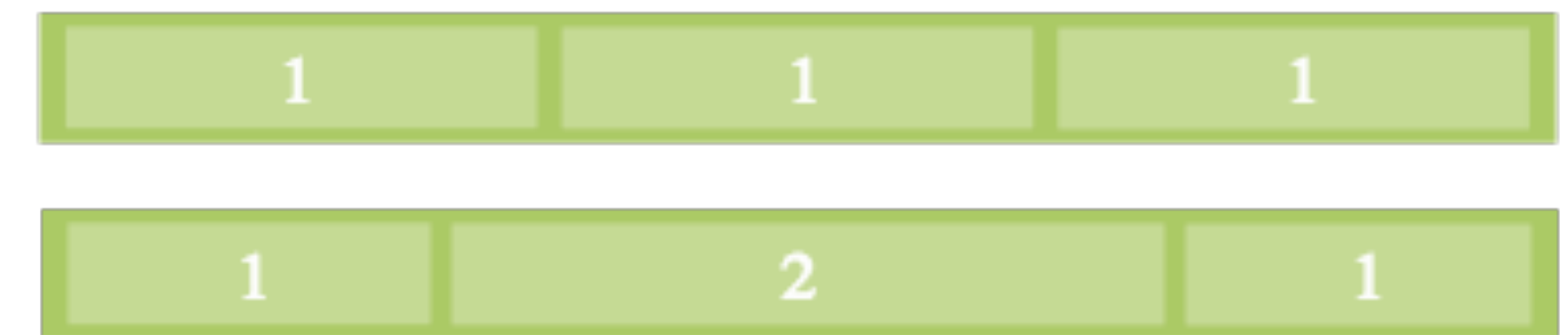
```
.container{  
  flex-wrap: nowrap | wrap | wrap-reverse;  
}
```

# The Items



# Flexbox Items: flex-grow

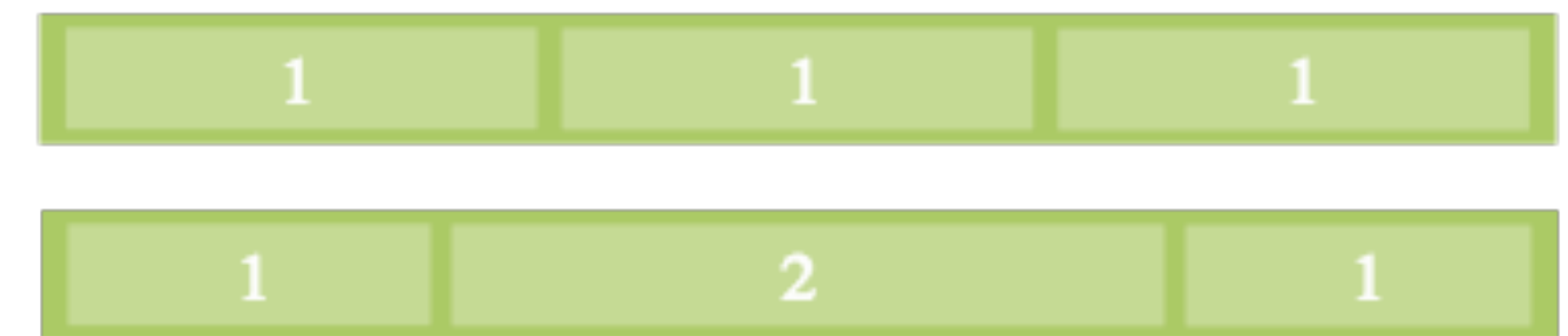
- accepts a unitless value that serves as a proportion
- Dictates the proportion of space inside the flex container the item should take up



```
.item {  
  flex-grow: <number>; /* default 0 */  
}
```

# Flexbox Items: flex-grow

- ◉ e.g. if all items have flex-grow set to 1, every child will set to an equal size inside the container.
- ◉ If one item is set to 2, that child would take up twice as much space as the others.



```
.item {  
  flex-grow: <number>; /* default 0 */  
}
```

# Flexbox Items: flex-shrink

- ⦿ This defines the ability for a flex item to shrink if necessary. Negative numbers are invalid.
- ⦿ Not that necessary

```
.item {  
  flex-shrink: <number>; /* default 1 */  
}
```

# Flexbox Items: flex-basis

- Like width property (or height, depending on flex-direction).
- If a relative value, indicates proportion of that item's width that should be applied.
- Default size of element before flex-grow or flex-shrink kick in

```
.item {  
  flex-basis: <length> | auto; /* default auto */  
}
```



# Flexbox Items: flex

- This is the shorthand for flex-grow, flex-shrink and flex-basis combined. The second and third parameters (flex-shrink and flex-basis) are optional. Default is 0 1 auto.

```
.item {  
  flex: none | [ <'flex-grow'> <'flex-shrink'>? || <'flex-basis'> ]  
}
```

**<liveCode />**



# RESPONSIVE





# RESPONSIVE





# 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





**<liveCode />**

# Chrome Developer Tools

The screenshot displays the Chrome Developer Tools interface. The top section shows the browser tabs and the address bar with the URL `https://github.com/FullstackAcademy/newlearn/issues?utf8=✓&q=is%3Aissue+is%3Aopen+comments`. The main content area shows the GitHub repository page for `FullstackAcademy / newlearn`, which is marked as `PRIVATE`. The repository has 24 Unwatch, 11 Star, and 5 Fork actions.

The Chrome Developer Tools are open at the bottom. The **Elements** panel on the left shows the DOM tree. The selected element is the `body` tag with the class `logged_in env-production macintosh vis-private`. The **Styles** panel on the right shows the default styles for the `body` element, including `display: block` and `margin: 8px`.

The **Elements** panel shows the following structure:

```
<!DOCTYPE html>
<html lang="en" class="is-copy-enabled">
  <head prefix="og: http://ogp.me/ns# fb: http://ogp.me/ns/fb# object: http://ogp.me/ns/object# article: http://ogp.me/ns/article# profile: http://ogp.me/ns/profile#">...</head>
  <body class="logged_in env-production macintosh vis-private">
    <a href="#start-of-content" tabindex="1" class="accessibility-aid js-skip-to-content">Skip to content</a>
    <div class="header header-logged-in true" role="banner">...</div>
    <div id="start-of-content" class="accessibility-aid">...</div>
    <div id="js-flash-container">...</div>
    <div role="main" class="main-content">...</div>
    <div class="container">...</div>
    <div id="ajax-error-message" class="flash flash-error">...</div>
    <script crossorigin="anonymous" integrity="sha256-Ln/D0mSiCOE4PehbgVN5vsz/VsH5d3FFFDTKx4I07z4=" src="https://assets-cdn.github.com/assets/frameworks-2e7fc3d264a208e1383de85b815379beccff56c1f977714515d4cac7820eef3e.js"></script>
    <script async="async" crossorigin="anonymous" integrity="sha256-0ZauYVQU9+hAsJTbTqtuSyWQK5rrZ0UHUHJlozDGwdA=" src="https://assets-cdn.github.com/assets/github-d196ae615414f7e840b094db4eab6e4b25902b9aeb674507521265a330c6c1d0.js"></script>
    <div class="js-stale-session-flash stale-session-flash flash flash-warn flash-banner hidden">...</div>
    <div class="facebox" id="facebox" style="display:none;">...</div>
    <script id="hiddenSubmitdiv" style="display: none;"></script>
    <script>...</script>
  </body>
</html>
```

The **Styles** panel shows the following styles:

- `element.style { }`
- `media="all" body { word-wrap: break-word; }`
- `media="all" body { font: 13px/1.4 Helvetica, arial, nimbussansl, liberation sans, freesans, clean, sans-serif, "Segoe UI Emoji", "Segoe UI Symbol"; color: #333; background-color: #fff; }`
- `media="all" body { margin: 0; }`
- `media="all" * { box-sizing: border-box; }`
- `body { display: block; margin: 8px; }`
- `Inherited from html.is-copy-enabled`
- `media="all" html { font-family: sans-serif; }`