Maintenance of Modern Web Apps' Front-end Code

CPEN400A (Fall '18 - Dr. Karthik Pattabiraman)

Guest lecture by Davood Mazinanian

http://dmazinaian.me/

Who am I?!

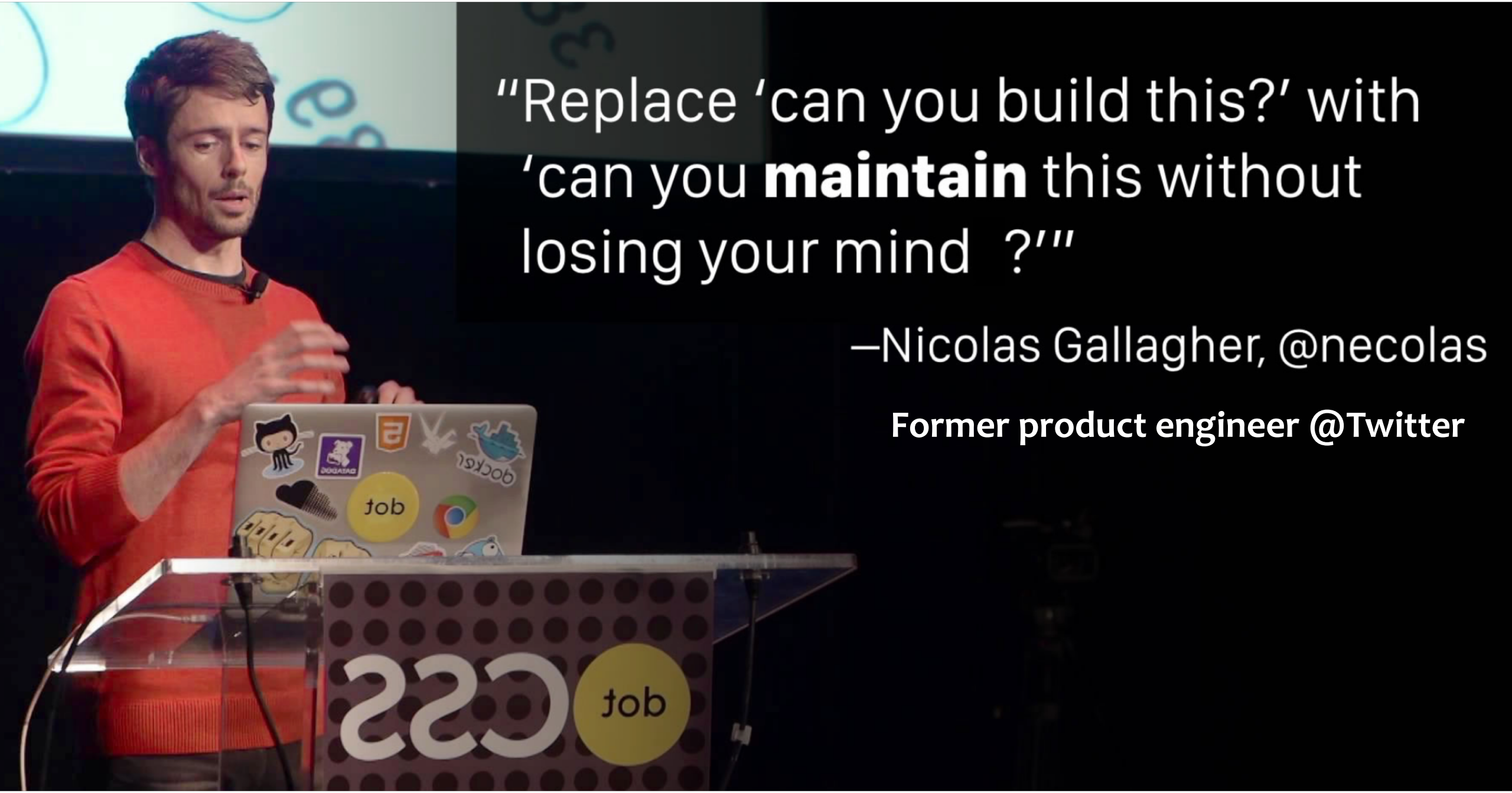
- PhD @ Concordia - 2017
  - Thesis: Duplicated code in CSS
  - Also: Refactoring duplicated code in Java, empirical studies, JS source code analysis
- Postdoc @ ECE SALT lab with Dr. Ali Mesbah
  - Refactoring HTML using visual cues
  - Driving crawling using CSS style cues
  - …

Software maintenance is important

- Maintenance is believed to consume as high as 80% of the total software life cycle costs
- Estimated at around $70 billion annually

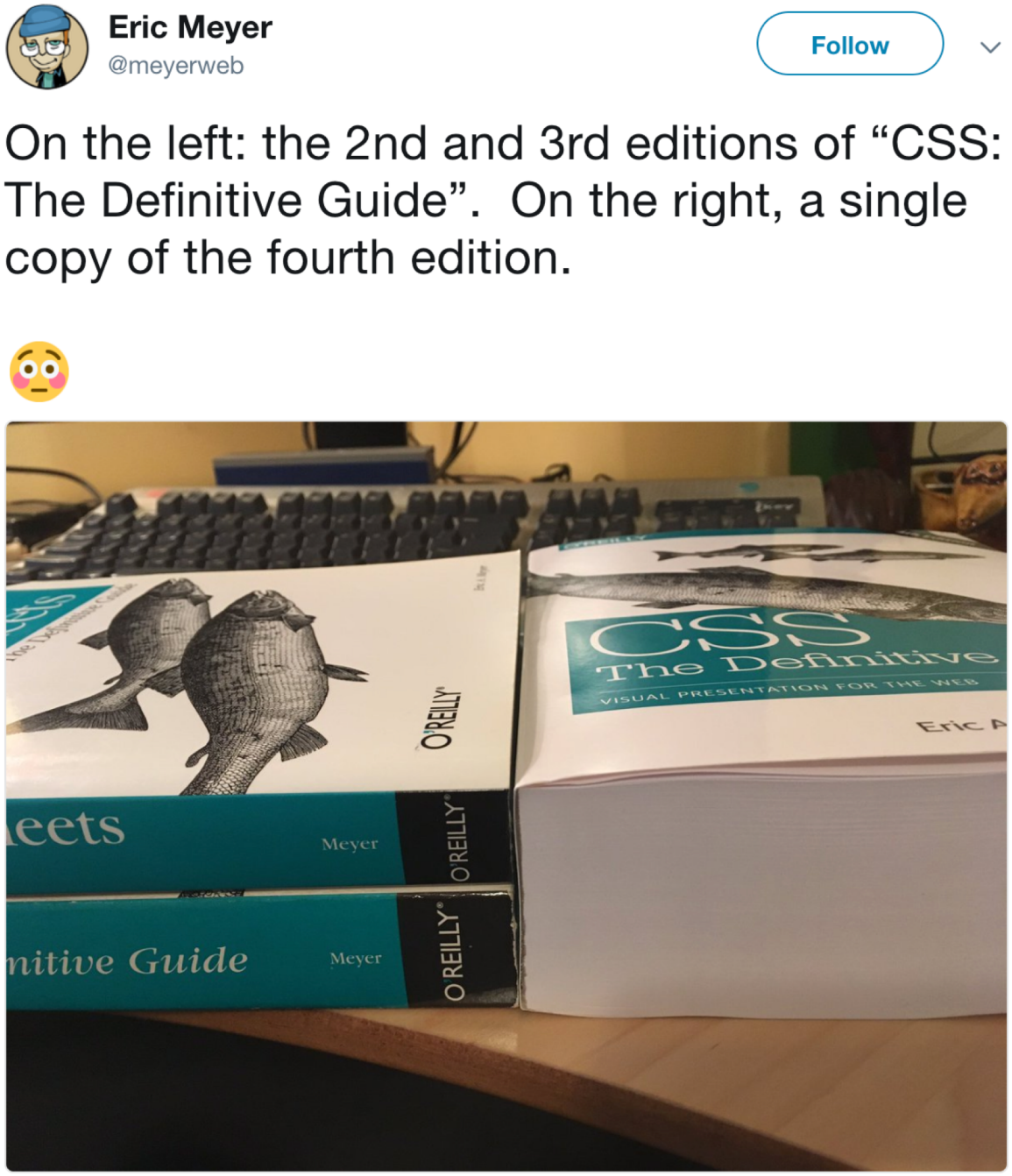
Maintenance is difficult in modern web apps

- Polyglot in nature, Interleaved code
- Dynamicity
- Immature tooling
- Platform and technology fragmentation
- Ad-hoc solutions
- …
- All these are especially true for the front-end code


"Replace 'can you build this?' with
'can you **maintain** this without
losing your mind ?'"
—Nicolas Gallagher, @necolas
Former product engineer @Twitter

This talk: focus on HTML and CSS

- You probably know (or have to know) a lot about JS ☺
- Everyone talks about JS
  - Lots of material, lots of focus
- There is still a lot of pain points in HTML/CSS that you should be aware of



Dropbox

200K CSS LOC!

Etsy

>400K CSS LOC

>2K CSS files!

How would you maintain this?!

What we will talk about

- (Just two) maintenance activities for the front-end
- Some tips to improve maintainability

Some maintenance activities for front-end

1. **Dead code** detection and removal
2. **Duplicated code** detection and removal

- There is A LOT MORE that we don't have the time to cover
  - Migrating front-end code, Impact analysis, reverse engineering, fault localization, etc.

1) Dead code

- A section in the source code of a program **which is executed** but whose **result is never used** in any other computation

1) Dead code (in HTML)

- Bloated HTML at GitHub in 2012
- Average diff page has ~9K lines
  - It took 15ms to load only the HTML (without CSS and JS)

Don't show manage emails button when not logged in

Browse files

master

dmazinanian committed 21 days ago 1 parent 178bc59 commit 932e99890d0947dd35755e9529bcce551b091dcf

Showing 2 changed files with **9 additions** and **4 deletions**.

Unified | Split

4 FrontEnd/src/app/jumbotron.component.html View

```
@@ -4,8 +4,8 @@ <h1 class="display-4">Merge Conflicts Study</h1>
```

```
4           <p class="lead">
5               This is a study of merge conflicts.
6           </p>
7   -       <hr class="my-4">
8   -       <a class="btn btn-primary btn-md" [routerLink]="['/emails']" routerLinkActive="active" role="button">
9               Manage Emails
10          </a>
11      </div>
```

```
4           <p class="lead">
5               This is a study of merge conflicts.
6           </p>
7   +       <hr class="my-4" *ngIf="backEndService.getUser()">
8   +       <a class="btn btn-primary btn-md" [routerLink]="['/emails']" routerLinkActive="active" role="button" *ngIf="backEndService.getUser()">
9               Manage Emails
10          </a>
11      </div>
```

1) Dead code (in HTML)

- Same logic with less markup (less s)

```
<pre><div>+ this is a line in the diff</div></pre>
```

dead

```
<pre>+ this is a line in the diff</pre>
```

There were 6,387 Unnecessary s
The so-called soup

Other changes to the GitHub's diff page

- Extra <a> tags (this is not dead code)
  - Controversial, since you will have worse accessibility instead

```
<td><a …>+ this is a line in the diff</a></td>
```

```
<td>+ this is a line in the diff</td>
```

Other changes to the GitHub's diff page

- Use simpler, shorter tags
  - Keep accessibility in mind again

```
<div class="add-bubble">
```

```
<b class="add-bubble">
```

1) Dead (and redundant) code (in HTML)

- Decreased diff page load by >40%

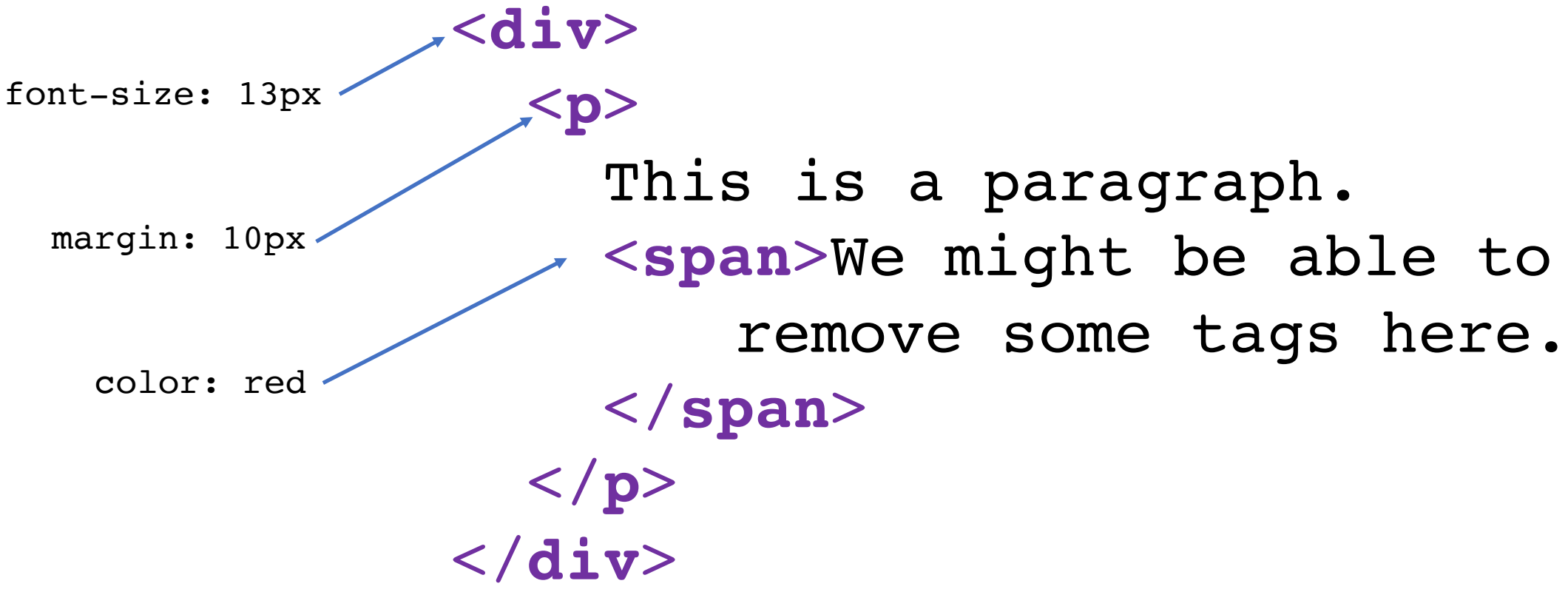
Activity

- Do we have dead code here?

```
<div>
  <p>
    This is a paragraph.
    <span>We might be able to
      remove some tags here.
    </span>
  </p>
</div>
```

Activity

- Do we have dead code here?



1) Dead (and redundant) code (in HTML)

- I'm not aware of a tool that can do this!
  - Could be a nice research idea ☺

1) Dead code (in CSS)

- Usually in the form of **Oxbow code**
  - Fragments of program code that were once needed but now are not used.

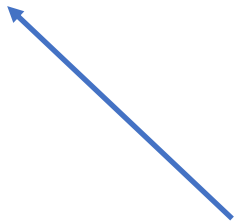
1) Dead code (in CSS)

```
<div>
  This content is
  <span class="underlined">NOT</span>
  useful.
</div>
```

```
.underlined {
  text-decoration: underline
}

.bold-faced {
  font-weight: bold
}
```

1) Dead code (in CSS)

```
<div>
  This content is
  <span class="underlined">NOT</span>
  useful.
</div>
```



```
.underlined {
  text-decoration: underline
}
```

Dead CSS

```
.bold-faced {
  font-weight: bold
}
```

Activity

- Is there dead CSS code here?

```
.underlined {
    text-decoration: underline
}

.im-dead {
    font-weight: bold
}
```

Activity

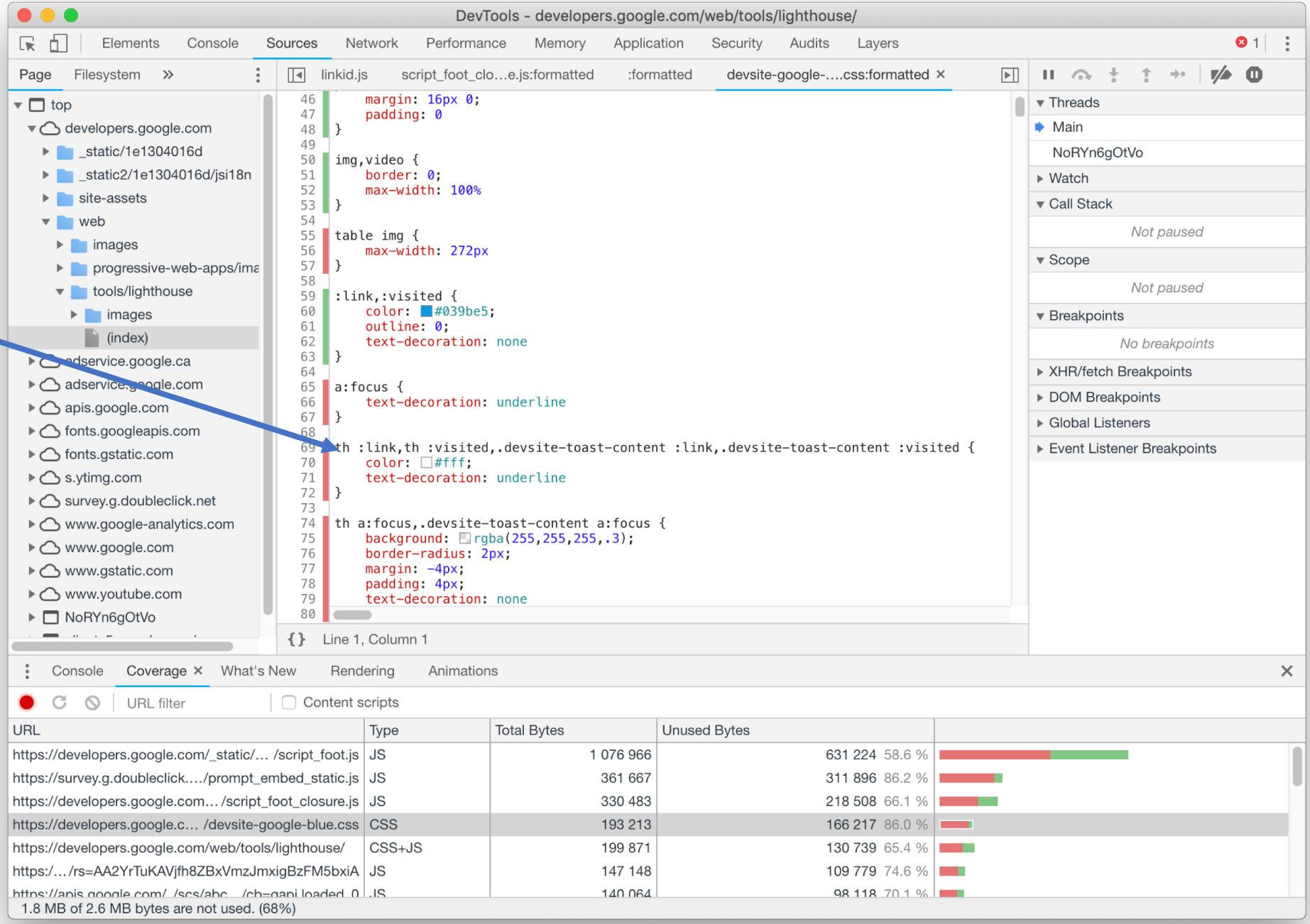
- Is there dead (or redundant) code here?

```
a {
  text-decoration: underline
}

a span {
  text-decoration: underline
}
```

1) Dead code (in CSS)

- Are there any tools to detect/remove dead code in CSS?
  - An active research problem!
    - Mesbah and Mirshokraie @UBC (Cilla - Dynamic analysis)
    - Hague et al. (Static analysis – tree rewriting)
    - Geneves et al. (Static analysis – reasoning)
    - …
  - In the industry:
    - Chrome DevTool's coverage tab
    - …

Can we remove these?



1) Dead code (in CSS)

Beware: no tools can detect CSS (and JS) dead code 100% correctly.
You don’t know about hidden states

YOU’LL NEED TESTS!

- A tiny change in CSS killed a revenue-generating page
  - The design engineers did not know it existed
- Dropbox could lose $$$ because of a few lines of CSS

Daniel Eden, Design Engineer at Dropbox

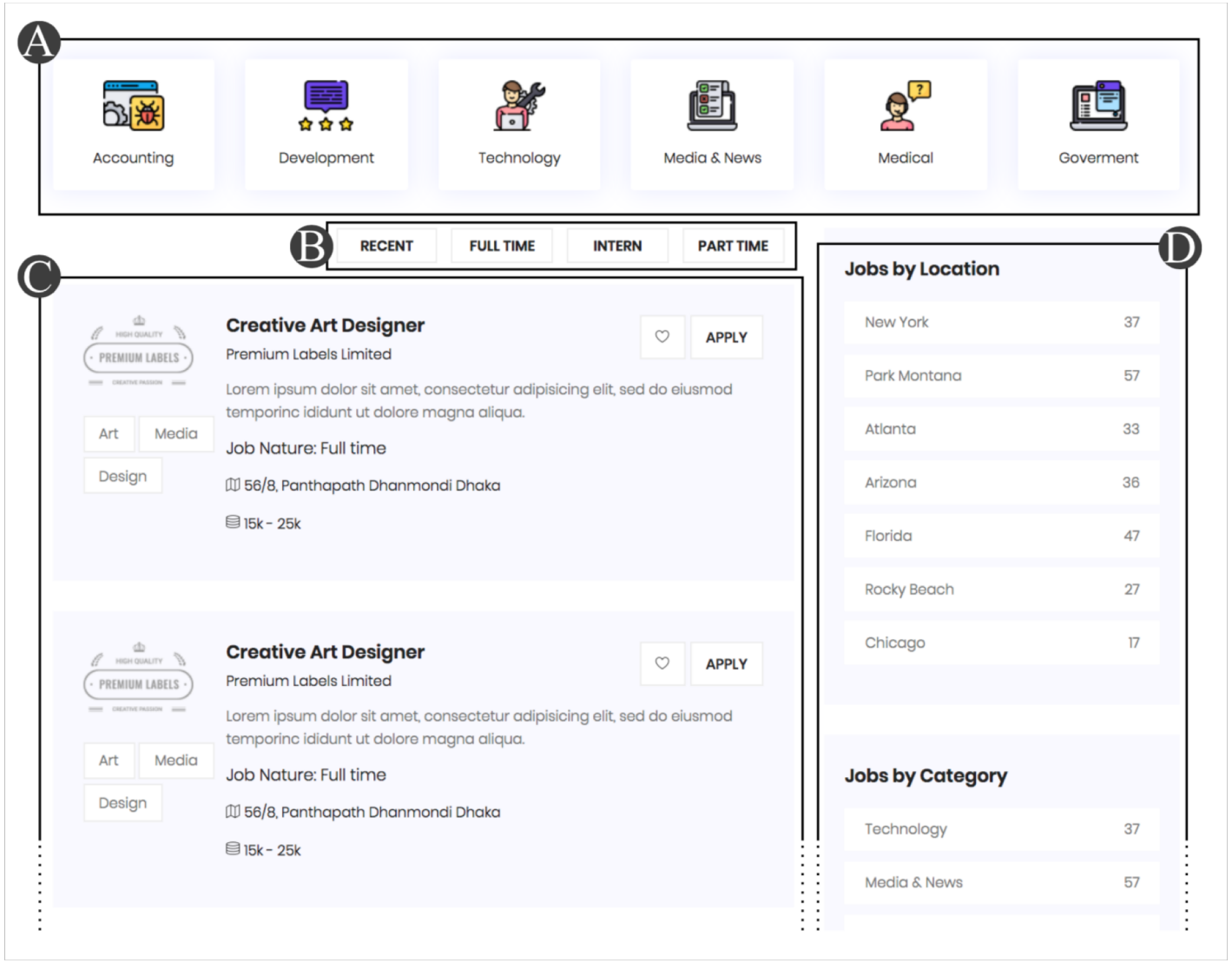
2) Duplicated code (AKA code clones)

- A sequence of source code that occurs more than once
- [Very] active line of research
  - Both detection and refactoring
- Research estimates up to 80% duplicated code in web applications

2) Duplicated code in HTML & CSS

- Scarcity of abstraction mechanisms => duplicated code
  - `for` loops, functions, variables, classes
    - Recently: variables in CSS (AKA custom properties)
  - That's why we generate HTML and CSS code
    - Server-side languages (PHP, …)
    - Preprocessors (Sass, Less, …),
    - Postprocessors (PostCSS, …)
    - JS Frameworks (Angular, React, Vue.js, …)

2) Duplicated code (in HTML)



2) Duplicated code (in HTML)

```
<div>
  <div>
    foo <img src="1.png"/>
  </div>

  <span>baz</span>

</div>
```

```
<div>
  <div>
    bar <img src="2.png"/>
  </div>
  <div>
    <span>text1</span> text2
  </div>
</div>
```

2) Duplicated code (in HTML)

- The “template” is being repeated

```
<div>
  <div>
    foo <img src="1.png"/>
  </div>

  <span>baz</span>

</div>
```

```
<div>
  <div>
    bar <img src="2.png"/>
  </div>
  <div>
    <span>text1</span> text2
  </div>
</div>
```

2) Duplicated code (in HTML)

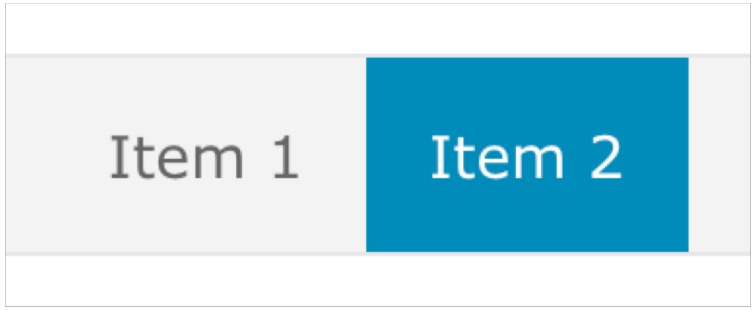
- How to detect this?
  - Can we use normal clone detectors?
    - NiCAD, CCFinder, etc
  - However, they leave out presentation altogether

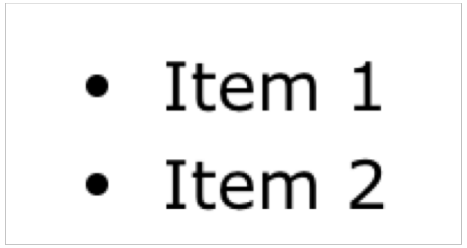
2) Duplicated code (in HTML)

- How to detect this?
  - Visual duplication => clones in DOM, but not necessarily the way around

```
<ul class="menu">
  <li>Item 1</li>
  <li>Item 2</li>
</ul>
```

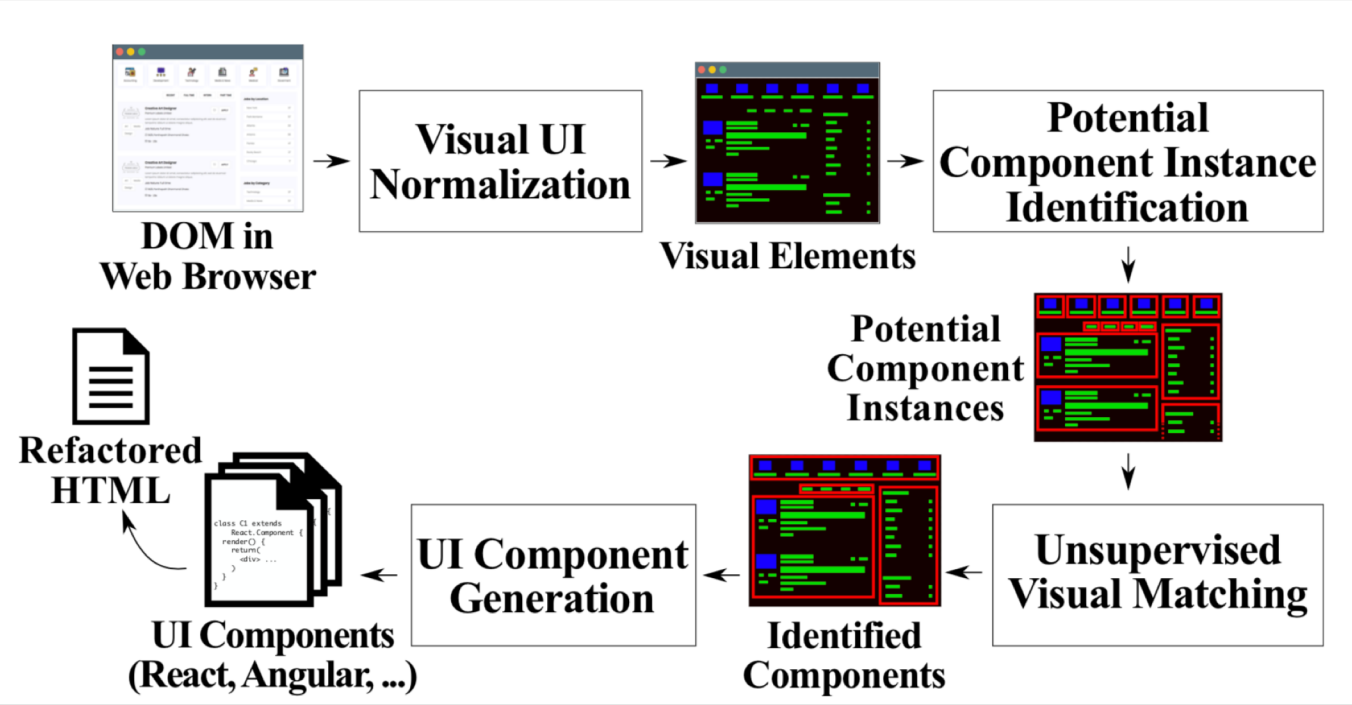
```
<ul class="text">
  <li>Item 2</li>
  <li>Item 1</li>
</ul>
```





2) Duplicated code (in HTML)

- How to detect this?
  - Research tool: VizMod {Bajammal, Mazinanian, Mesbah} @UBC



2) Duplicated code (in HTML)

- Refactoring

```
<div>
  <div>
    foo <img src="1.png"/>
  </div>
  <span>baz</span>
</div>

<div>
  <div>
    bar <img src="2.png"/>
  </div>
  <div>
    <span>text1</span> text2
  </div>
</div>
```

Refactor

```
<component>
  <div>
    <div>
      {var1} <img src="{var2}"/>
    </div>
    {var3}
  </div>
</component>
```

Component "template"

"calls"

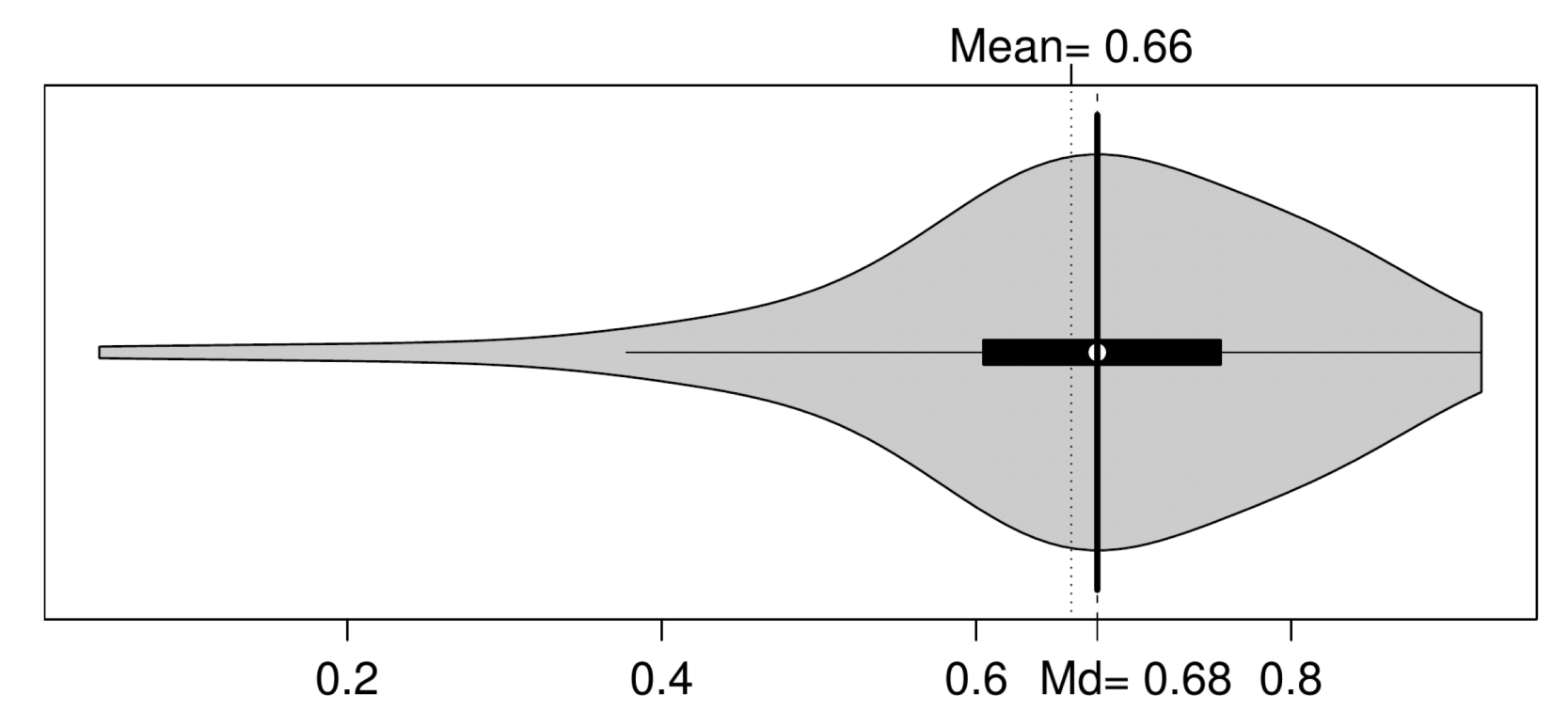
```
<component args=["foo", "1.png",
                 <span>baz</span>] />

<component args=["bar", "2.png",
     <div><span>text1</span>text2</div>] />
```

2) Duplicated code (in HTML)

- Use standard Web Components Suite
  - Custom Elements, Shadow DOM for scoping, etc.
  - Watch out browser compatibility
- Use a JS framework (Angular, React, Vue.js, etc.)

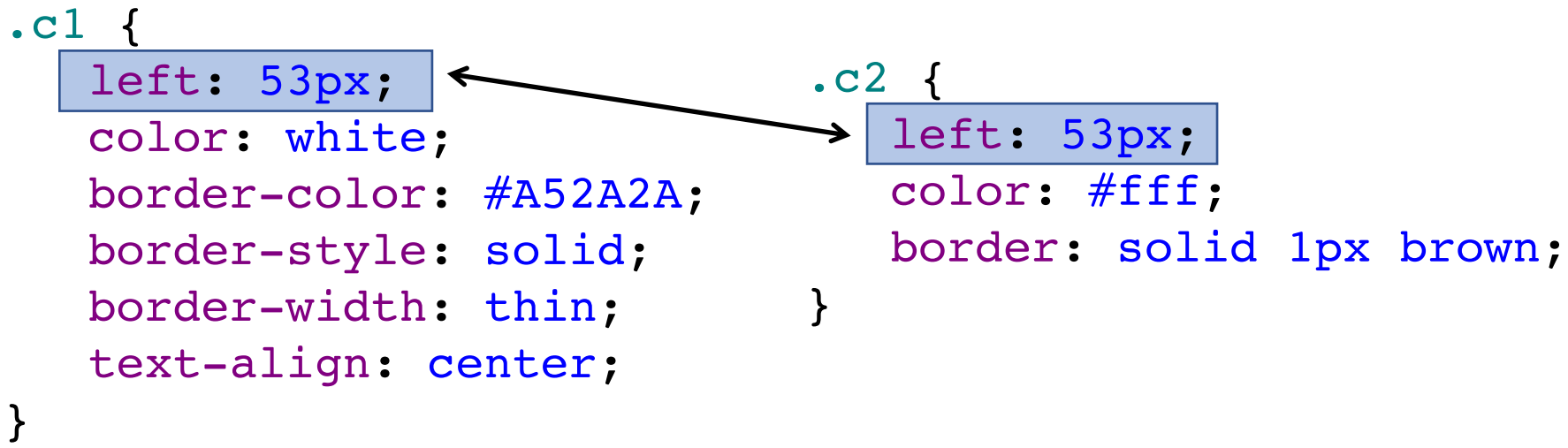
2) Duplicated code (in CSS)

- Again, occurs very frequently.
  - On median, 60% [Mazinanian et al, 2014]



2) Duplicated code (in CSS)

Type I (lexically identical values)



2) Duplicated code (in CSS)

Type II (equivalent values)

```
.c1 {
    left: 53px;
    color: white;
    border-color: #A52A2A;
    border-style: solid;
    border-width: thin;
    text-align: center;
}
```

```
.c2 {
    left: 53px;
    color: #fff;
    border: solid 1px brown;
}
```

2) Duplicated code (in CSS)

Type III (equivalent shorthand / individual values)

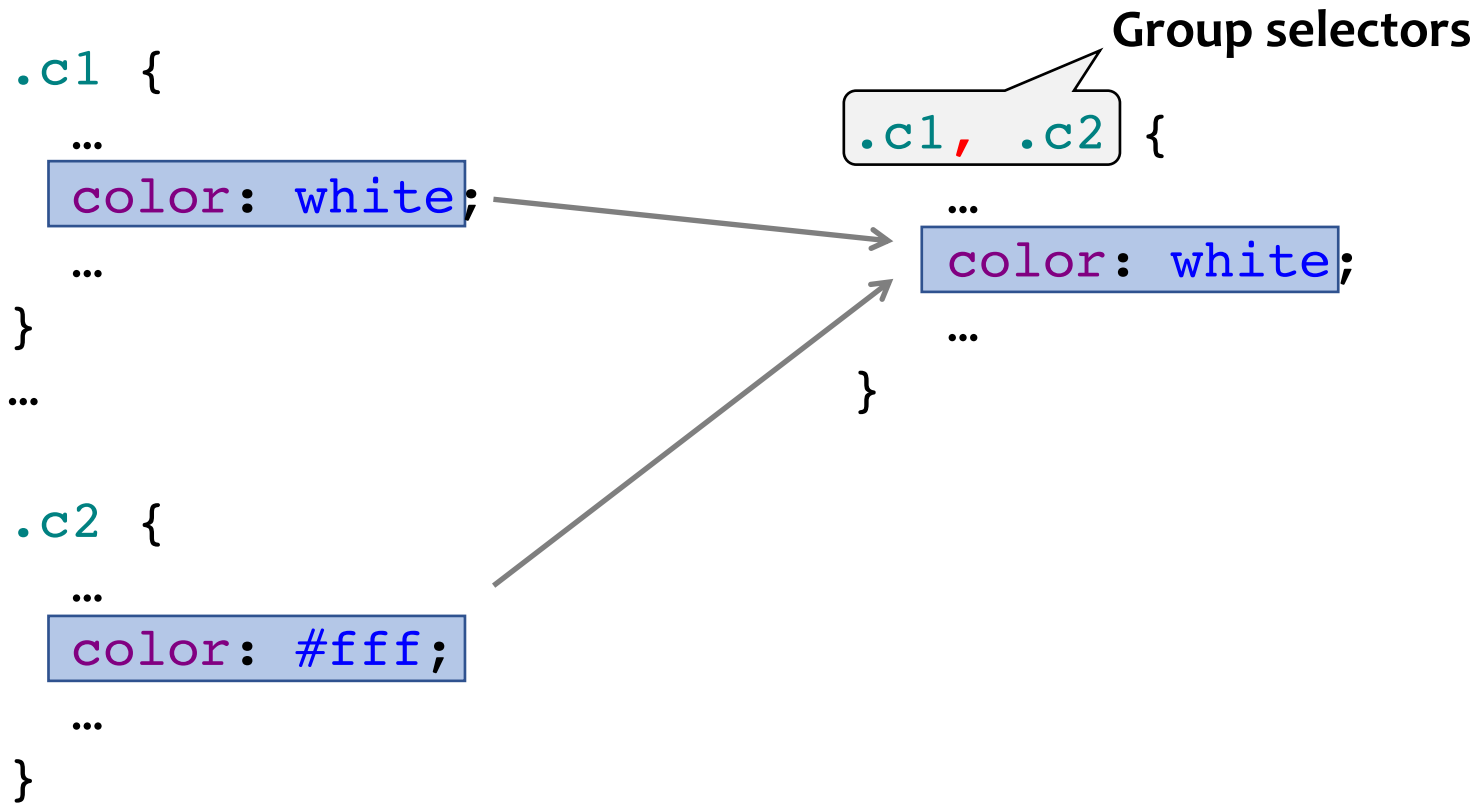
```
.c1 {
    left: 53px;
    color: white;
    border-color: #A52A2A;
    border-style: solid;
    border-width: thin;
    text-align: center;
}
```

↔

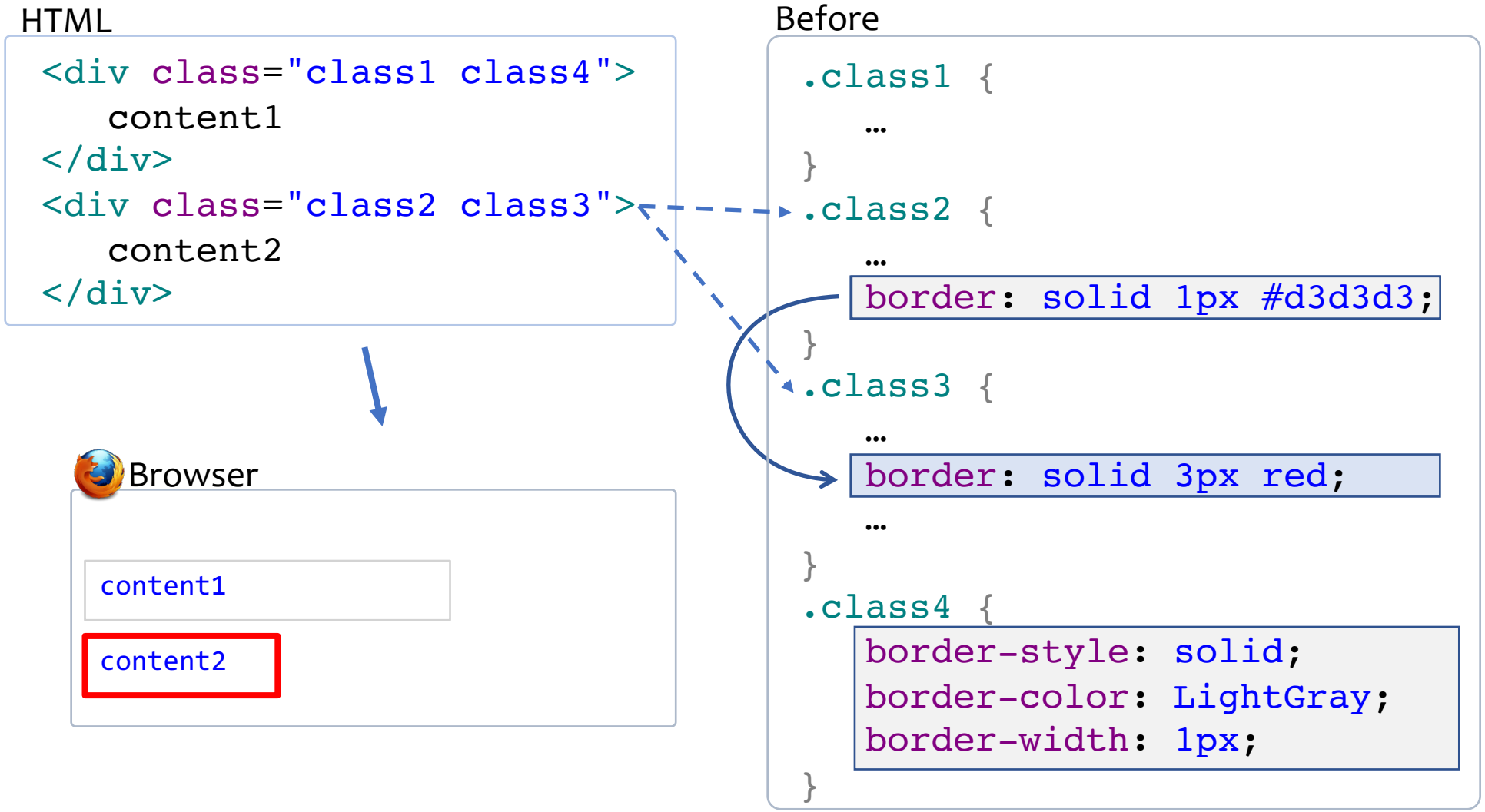
```
.c2 {
    left: 53px;
    color: #fff;
    border: solid 1px brown;
}
```

2) Duplicated code (in CSS)

- How to refactor?



2) Duplicated code (in CSS)



2) Duplicated code (in CSS)

HTML

```
<div class="class1 class4">
    content1
</div>
<div class="class2 class3">
    content2
</div>
```

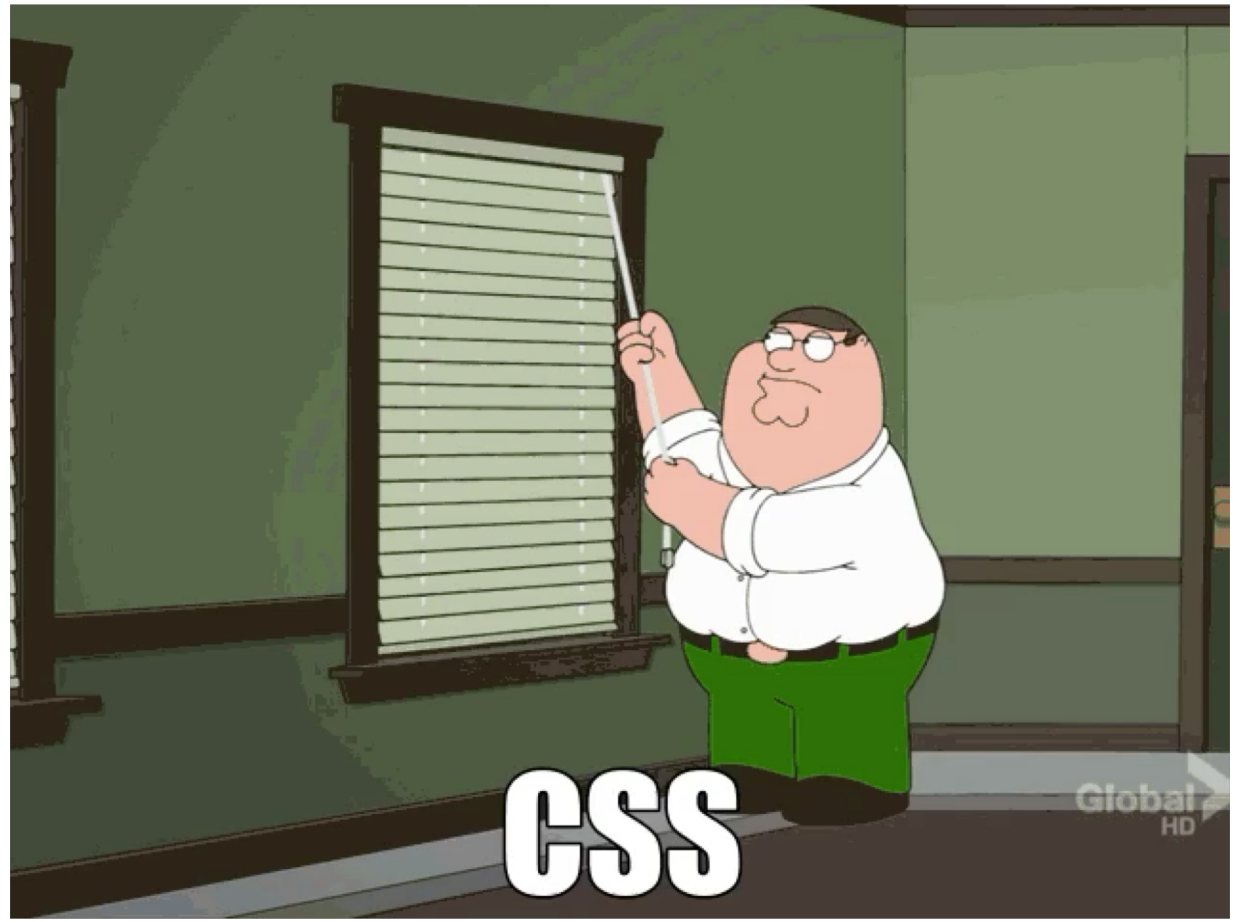
Browser

content1

content2

After

```
.class1 {
    …
}
.class2 {
    …
}
.class3 {
    …
    border: solid 3px red;
    …
}
.class2, .class4 {
    border: solid 1px #d3d3d3;
}
```


CSS
Global
HD

2) Duplicated code (in CSS)

Before

```
…
.class2 {
    color: #00F;
    font-weight: bold;
    border: solid 1px #d3d3d3;
}
.class3 {
    font-weight: 700;
    border: solid 3px red;
    float: left;
}
…
```

After

```
…
.class2 {
    color: #00F;
    font-weight: bold;
}
.class2, .class4 {
    border: solid 1px #d3d3d3;
}
.class3 {
    font-weight: 700;
    border: solid 3px red;
    float: left;
}
…
```

Reorder
New style rule

Activity

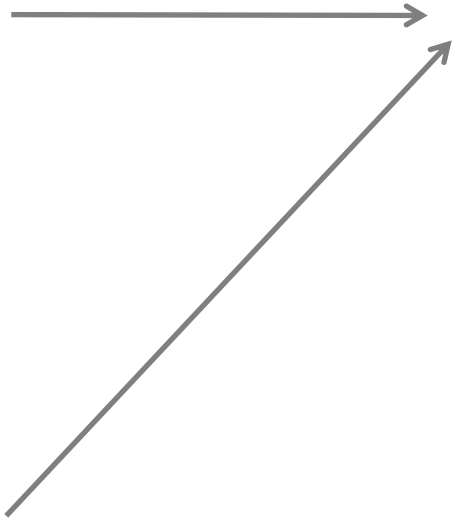
- Is this change safe?

Before

```
.c1 {
  color: white;
}

.c2 {
  color: blue;
}

.c3:hover {
  color: white;
}
```

After

```
.c1, .c3:hover {
   color: white;
}
.c2 {
  color: blue;
}
```



2) Duplicated code (in CSS)

What if the style values were different?

```
.c1 {
    left: 53px;
    color: white;
    border-color: #A52A2A;
    border-style: dotted;
    border-width: thin;
    text-align: center;
}
```

```
.c2 {
    left: 53px;
    color: #fff;
    border: solid 1px brown;
}
```

2) Duplicated code (in CSS)

- Currently not possible to avoid in pure CSS
- CSS preprocessors can be used
  - Less, Sass, Stylus, Closure Style Sheets, etc.

```
.c1 {
  .f(dotted; 3px);
   text-align: center;
}
```

```
.c2 {
   .f(solid; 1px);
}
```

Preprocessor Mixin →

```
.f(@style; @width) {
   left: 53px;
   color: white;
   border-color: brown;
   border-style: @style;
   border-width: @width;
}
```

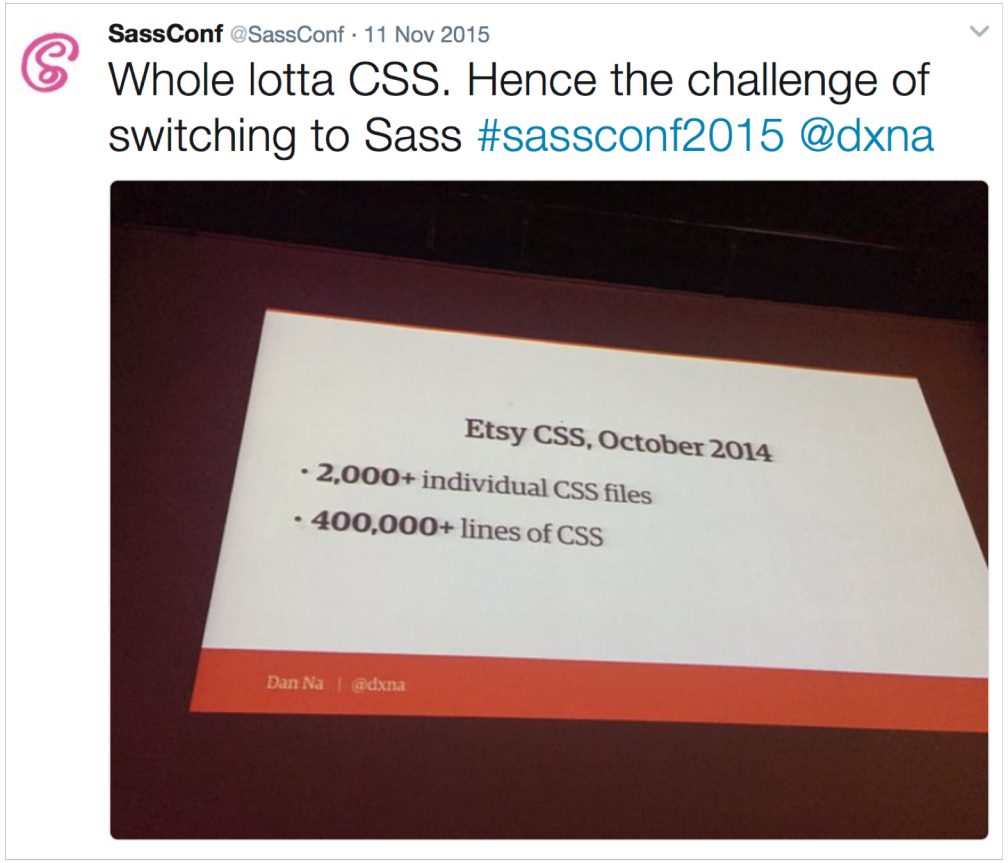
Transpilation

```
.c1 {
   left: 53px;
   color: white;
   border-color: #A52A2A;
   border-style: dotted;
   border-width: 3px;
   text-align: center;
}
```

```
.c2 {
   left: 53px;
   color: #fff;
   border: solid 1px brown;
}
```

2) Duplicated code (in CSS)

- Refactor existing CSS code to preprocessors to avoid duplication



2) Duplicated code (in CSS)

- Can we automate this?
  - I'm not aware of any industrial tool
  - Research tool: CSSDev [Mazinanian et al. 2017]
  - Refactor CSS by
    - Grouping selectors
    - Migrating to preprocessor

There is a lot more to talk about

Write maintainable front-end code

1) Learn fundamentals
   - Frameworks come and go, web platform fundamentals stay

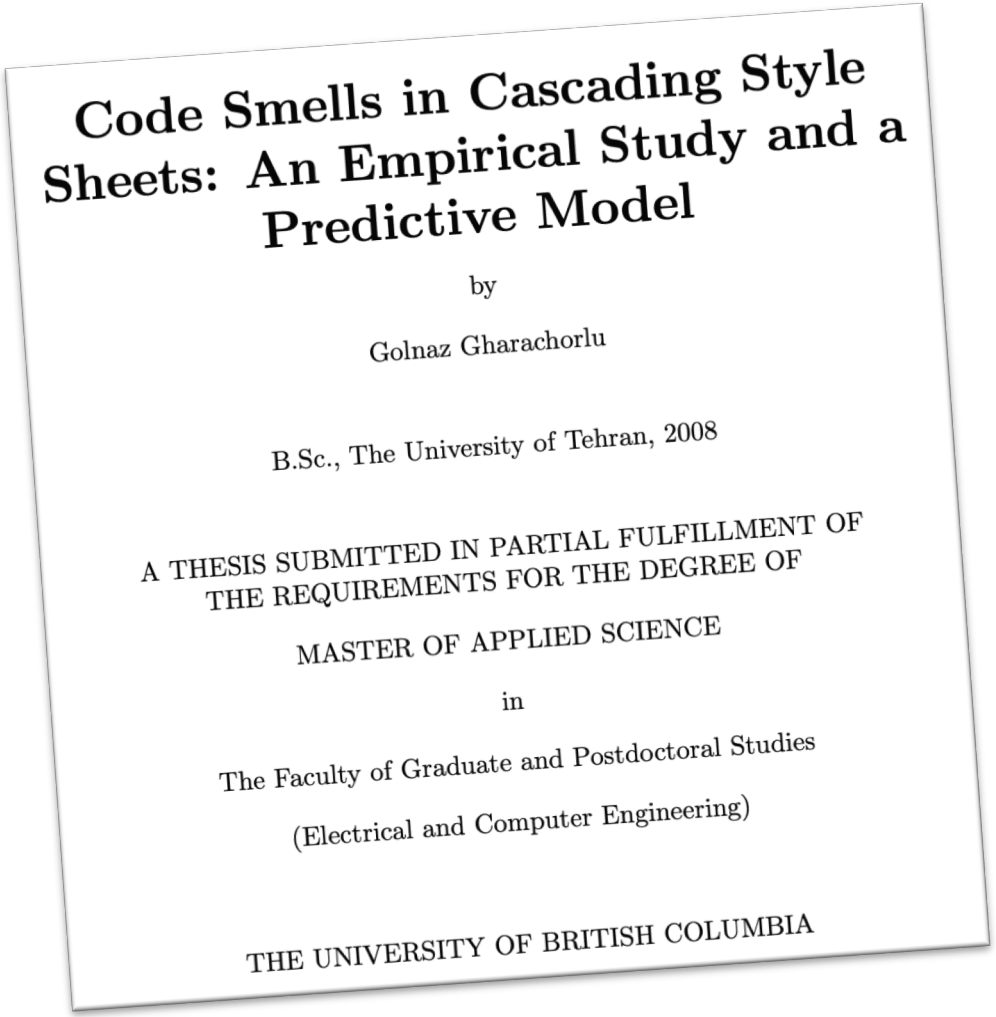
Write maintainable front-end code

2) Use frameworks when there are clear benefits

- Lots of big names are staying away from frameworks

Write maintainable front-end code

3) Learn best practices and avoid “code smells”



Code Smells in Cascading Style Sheets: An Empirical Study and a Predictive Model

by

Golnaz Gharachorlu

B.Sc., The University of Tehran, 2008

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

The Faculty of Graduate and Postdoctoral Studies

(Electrical and Computer Engineering)

THE UNIVERSITY OF BRITISH COLUMBIA

Write maintainable front-end code

4) Strive to write modular code

- Modularity is still a problem at client-side
- Not weird to see this kind of articles in 2018:

Write maintainable front-end code

5) Know your tools, but don’t become too attached to them

- Chrome DevTools, Lighthouse, etc. are amazing tools for front-end development, debugging, and maintenance

That’s all, folks.

(Never put this kind of slide at the end of your talks)