CSS:
THE SPECIFICITY WARS

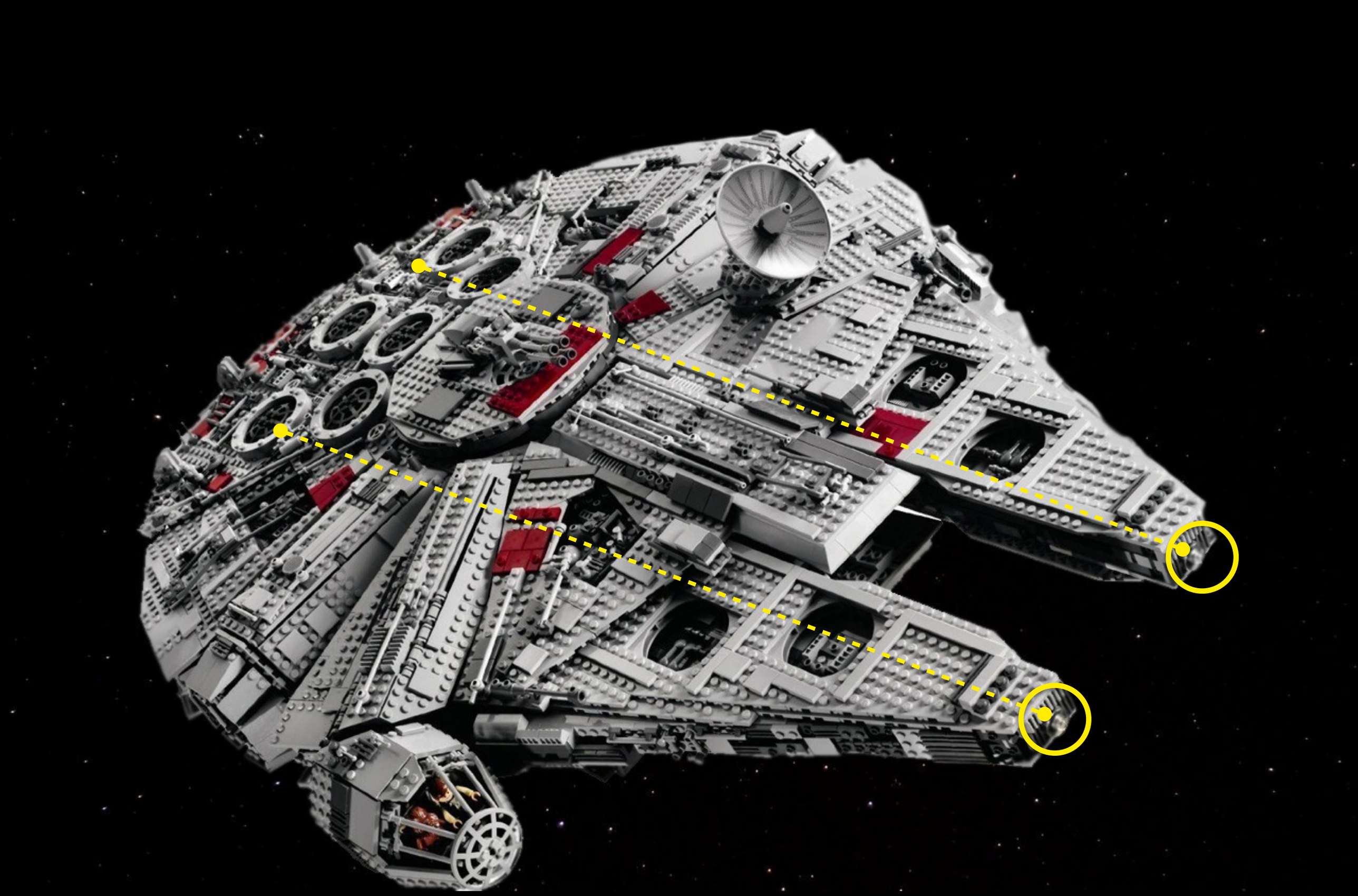
Benjamin Hong
Front End Developer

@bencodezen

“Two CSS properties walk into a bar.

A bar stool in a completely different bar falls over.”

- Thomas Fuchs



CSS

CRAZY SPECIFIC STYLES

“It doesn’t matter how well-considered your source order is; how well you’re utilizing the cascade; what naming conventions you use; **specificity can undo everything**.

- Harry Roberts (@csswizardry)

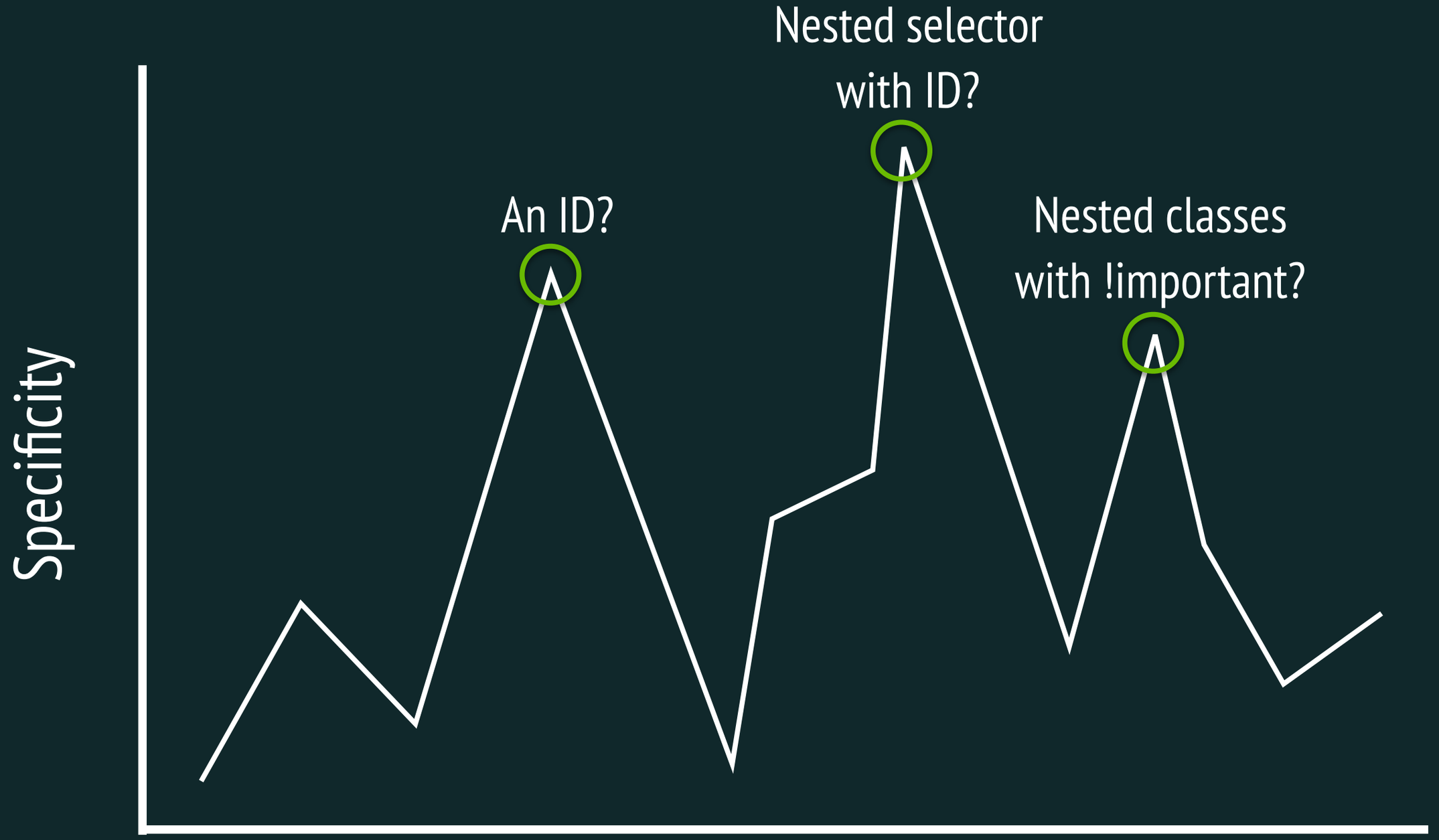
WHAT'S SPECIFICITY?

main.css

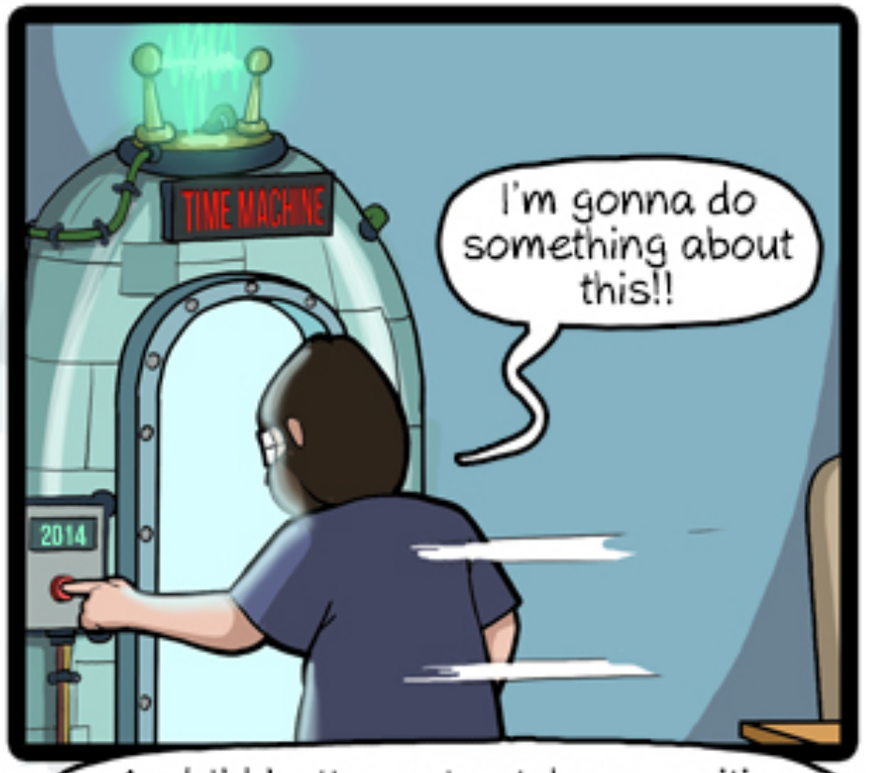
3,000 - 5,000

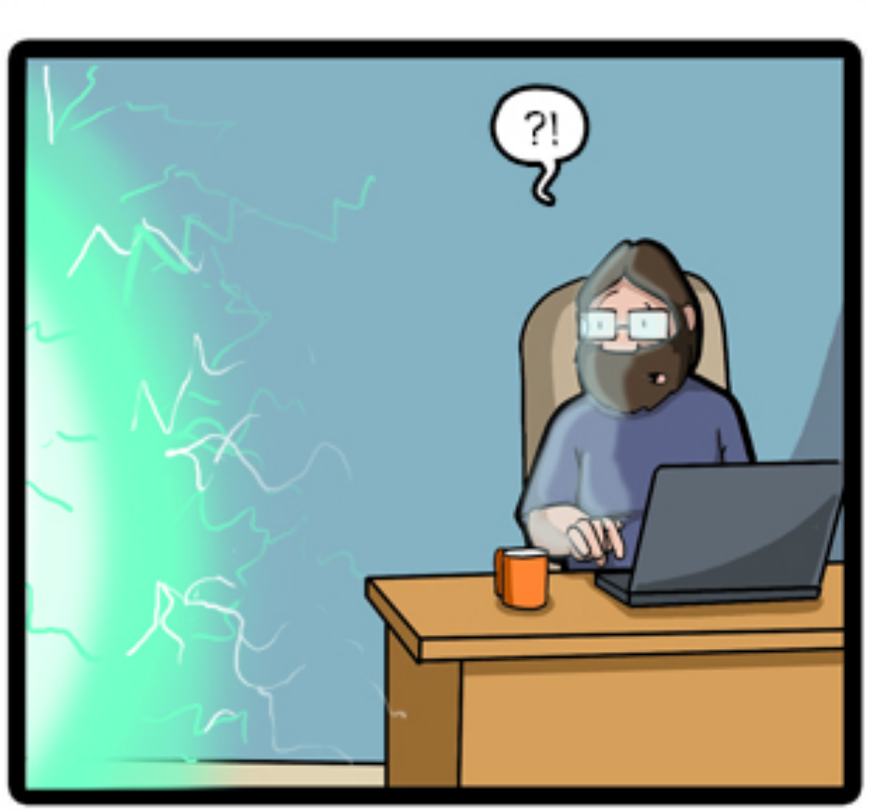
lines of code

Credit to @csswizardry



Credit to @csswizardry





CommitStrip.com

Instead of building beautiful products, you're wasting time and energy fixing someone's poorly written code.

WHY NOW?

Why is this so important all of a sudden?

- Everything is “agile”
- Websites have evolved
- Explosion of interest in coding

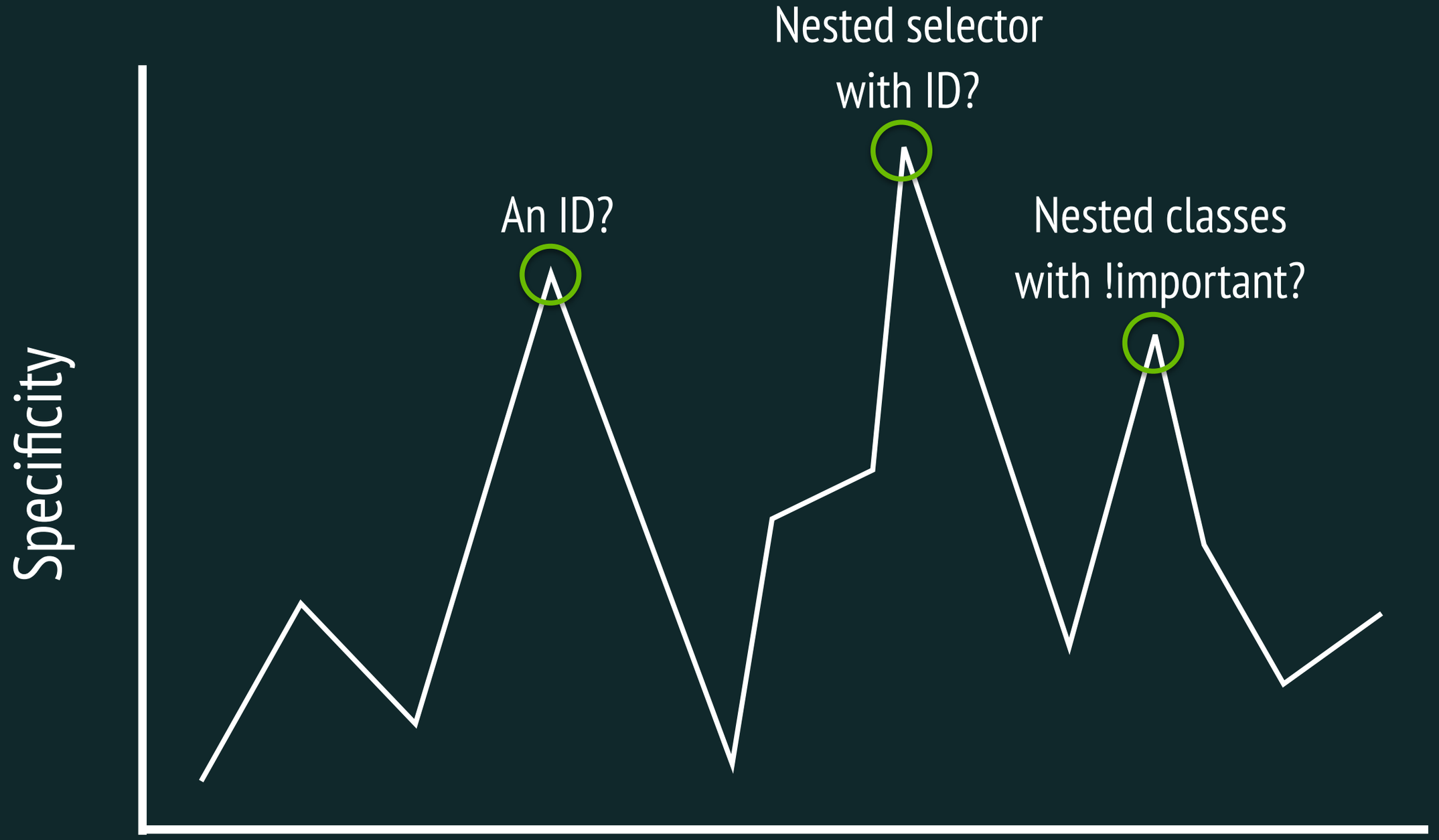
ALL CSS IS NOT CREATED EQUALLY

```css
p {
  color: red;
  font-size: 1em;
  line-height: 1.5;
}
```

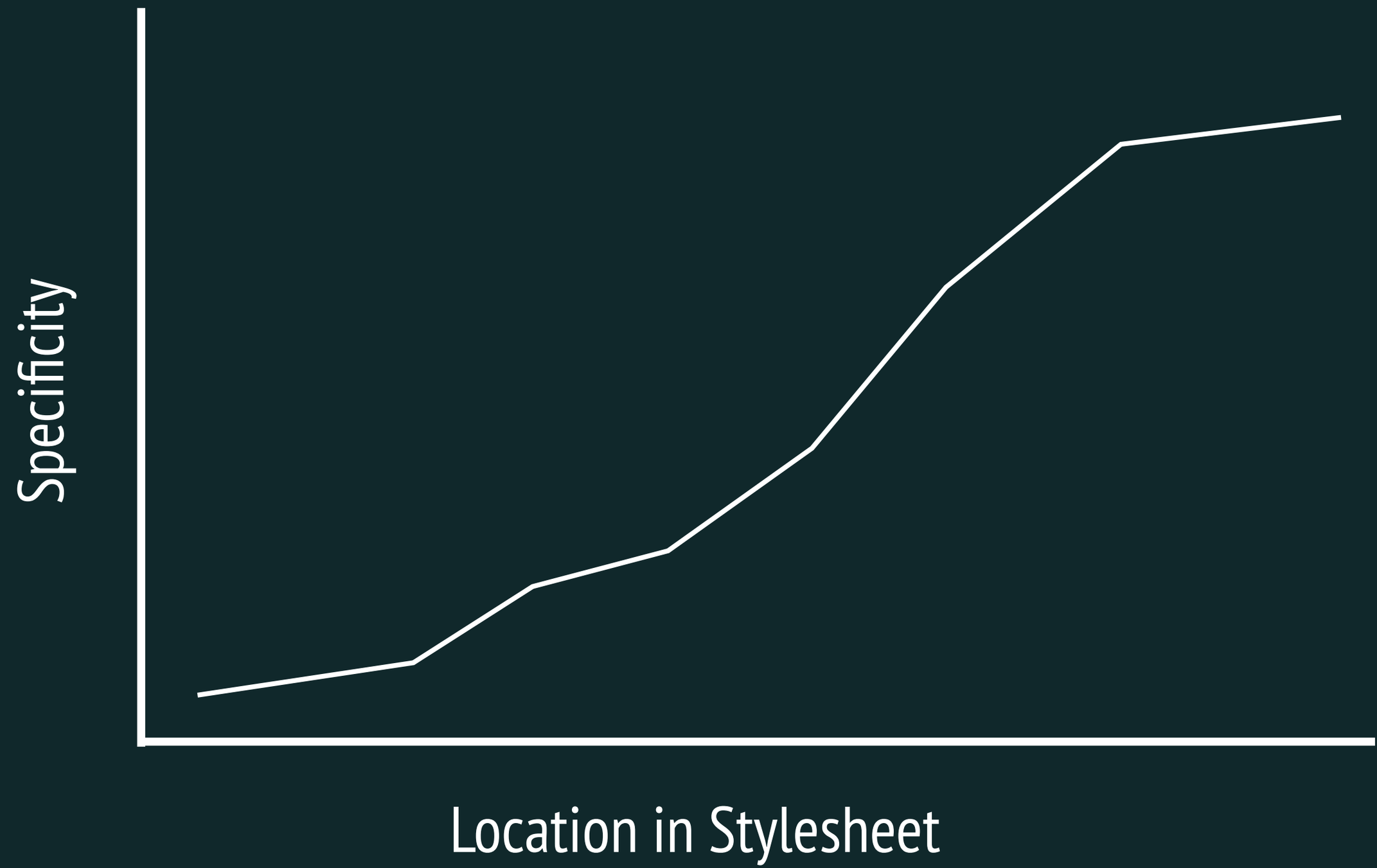
```css
html body p {
  color: red;
  font-size: 1em;
  line-height: 1.5;
}
```

COMMON MISTAKES

- Defaulting to ID for selecting elements
  - `#hero` instead of `.hero`
- Selectors that are more than 3 layers deep
  - `section #hero ul li a.current-page { … }`
- `!important`
- `a:link`



Credit to @csswizardry



Credit to @csswizardry

MODULAR ARCHITECTURE

ARCHITECTUR
E

CSS MODULAR
ARCHITECTURE

PRE-
PROCESSOR

NAMING CONVENTION

MY WORKFLOW

PRE-PROCESSOR	ARCHITECTURE	NAMING CONVENTION
Sass	ITCSS	BEM
	SMACSS	ATOMIC DESIGN

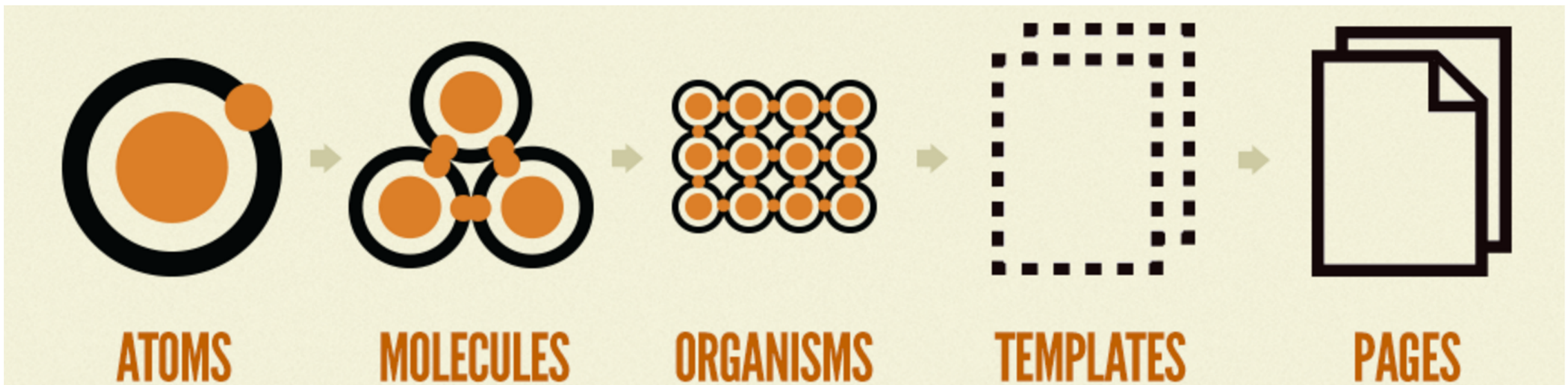
MY ARCHITECTURE

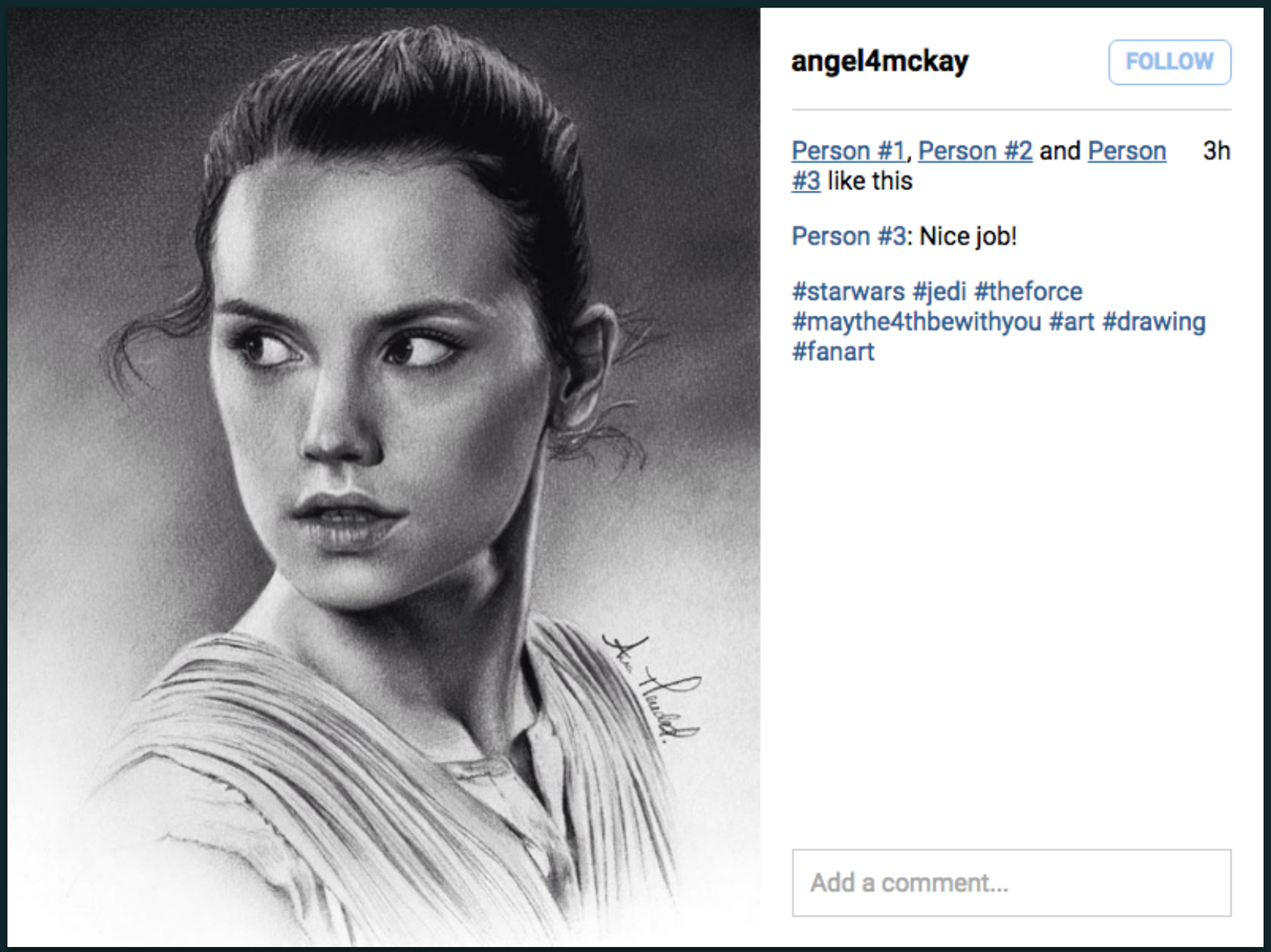
- **main.css**
    1. **settings** - configuration (i.e., colors, fonts, mixins, etc.)
    2. **vendor** - third party styles (i.e., normalize, reset, etc.)
    3. **base** - unclassed HTML elements (i.e., global, html, body, etc.)
    4. **layout** - generic page layout styles (i.e., grid, spacing, etc.)
    5. **components** - UI modules of the site (i.e., buttons, carousels, etc.)
    6. **pages** - custom page layouts (i.e., short marketing campaign)
    7. **overrides** - takes priority over everything else (i.e., 508, etc.)

NAMING CONVENTION

BLOCK ELEMENT MODIFIER

```
.hero__button--primary
```





Problem: http://bit.ly/1TKP5FC

Solution: http://bit.ly/1OeWOaP

FINAL THOUGHTS

GENERAL RULES

1. Always choose classes over IDs for selectors*

2. Avoid using general element selectors (i.e., h1, h2, p, span, etc.)*

3. If you have more than 3 selectors in a declaration, you're probably doing something wrong*

4. If your BEM class name is more than one level deep, you need to break it up into something small.

5. In the event you have to use `!important` or a very specific selector, add a quick comment so that future developers can make an informed decision.

THINGS TO REMEMBER

1. You won't get it right the first time.

2. "Simpler is better" when writing good CSS

3. Discipline to write beautiful CSS is one of the hardest parts

4. Remember to pick and choose what works best for your team and projects.

5. This philosophy is just as much about improving code quality as it is for keeping your mental sanity

TOOLS INVENTORY

- **Pre-Processors**
  - Sass (SCSS) - http://sass-lang.com/
  - Stylus - http://stylus-lang.com/
- **Architecture**
  - ITCSS - http://itcss.io/
  - SMACSS - https://smacss.com/
- **Naming Conventions**
  - BEM - http://getbem.com/
  - Atomic Design - http://atomicdesign.bradfrost.com/
  - OOCSS - https://www.smashingmagazine.com/2011/12/an-introduction-to-object-oriented-css-oocss/

ADDITIONAL RESOURCES

- **Managing CSS Projects with ITCSS** by Harry Roberts
  https://speakerdeck.com/dafed/managing-css-projects-with-itcss

- **Atomic OOBEMITSCSS** by Una Kravetz
  http://www.sitepoint.com/atomic-oobemitscss/

- **An Introduction to BEM**
  http://getbem.com/introduction/

- **Installing Sass**
  http://sass-lang.com/install

Q&A

THANK YOU!

If you have any additional questions, feel free to reach out to me @bencodezen!