Planning and Building a Big Front-End

FOWD – London, 2014

Me

Harry Roberts

Consultant Front-end Architect

@csswizardry

csswizardry.com

CSS


sky

BSkyB

Sky Betting & Gaming

2012 revenue: £113m

(m.)skybet.com

Engineering – 100+

Aluminium

You?

The workshop

What are we going to do?

Please see README.md for a loose schedule

Why?

Sites and teams are getting bigger.

Front-ends are getting more complex.

More apps and products being built all the time.

Many browsers on many devices… it's not easy!

Front-end developers are in high demand.

Highest Quality **PSD to HTML** Conversion

send the design
send us your design in any common format

pay & place order
pay for the service you've chosen by various payment methods

delivery chopped
get your design chopped crossbrowser / w3c valid

start **Order Now**

Convert your PSD to XHTML

- Hand Coded XHTML Strict Markup
- Semantic Markup used for SEO
- **180 Days** Free support
- Light-weight Tableless CSS Layout
- Clean W3C Valid XHTML / CSS
- Non-Disclosure Agreement

100% satisfaction or MONEYBACK GUARANTEE

5500+ Happy clients

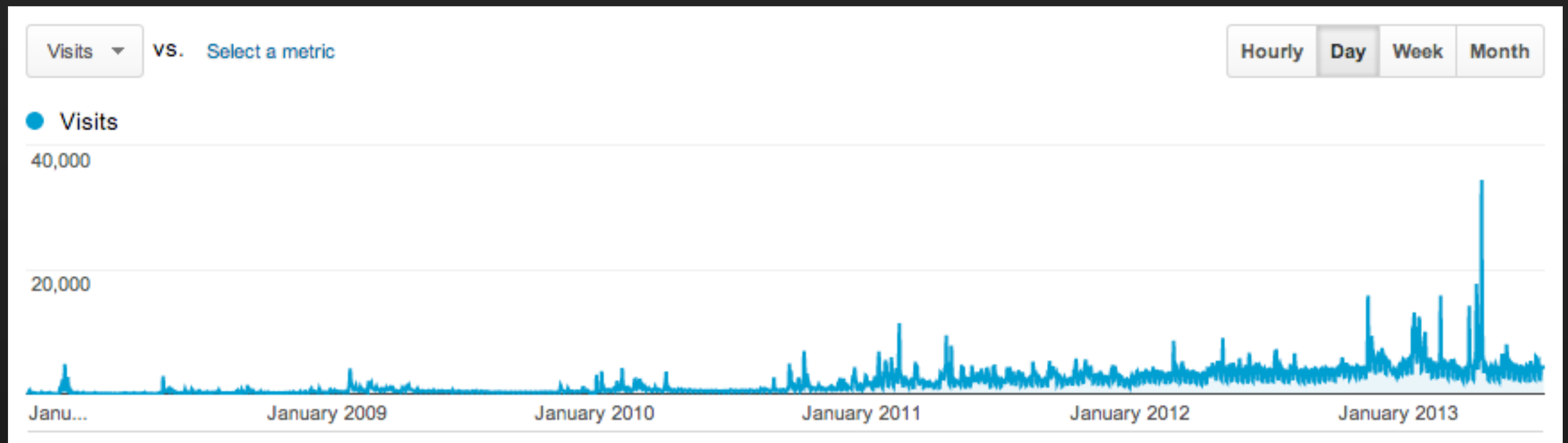
Highest Quality { Markup }

Ps Ai Fw

Starting from $45 Homepage $29 Innerpage

psd to XHTML	1 day turnaround	$45	Additional Page	(Inner) Pages	$29
psd to Email		$45	psd to Wordpress		$99
psd to Joomla		$149	psd to Responsive		$149
psd to HTML5		$85	psd to Magento		$299

php Hire DEDICATED PHP/Ajax Developers


Visits
vs.
Select a metric
Hourly
Day
Week
Month
Visits
40,000
20,000
Janu...
January 2009
January 2010
January 2011
January 2012
January 2013

What?

Work bigger, faster, smarter.

Keep codebases lean.

Write more scalable code.

Make sites easier to manage and maintain.

Anything else?

Setup

You (might) need

A text editor

Sass

Photoshop – optional

Git(Hub) – optional

tlk.io/fowdcss2014

Git(Hub)

```
$ git clone https://github.com/csswizardry/planning-and-building-a-big-front-end.git fowdcss2014
```

oilerplate — Edit

commit | 1 branch | 0 releases | 1 contributor

anch: master | fronteers /

it

dry authored an hour ago — latest commit b5c0f48626

E.md — Initial commit — an hour ago

ME.md

nteers

op boilerplate

Sass

```
$ cd fowdcss2014/css
$ ./watch
```

Sass

Includes

Variables

Task

Do anything at all with the codebase.

Let's look at a design…

CSS WORKSHOPS

They say every journey begins with a single step; the same is true with building big websites.

In this workshop we shall look at some simple and small steps we can take to lay the foundations on which to build maintainable, scalable and big front-ends.

More on the workshop »

“

Harry’s way of

CSS RESOURCES

New Perspectives On Coding »

Book | Smashing shop

Our PSD

/psd/page.p[sd|ng]

Designed by Naomi Atkinson (@naomisusi)

Intentional faults and inconsistency

Completely fictional

Task

Take a look over the design

Breaking designs down

It's all about components

No longer think about pages…

Think about components and modules…

Which are our building blocks that we…

Make pages from

Spotting components

Spot a component

Look at its underlying design patterns

Spot the same patterns elsewhere

Spot the differences

Group similarities

SOLD OUT

FIND OUR MORE

Task

Start spotting and grouping your components

Normalising and rationalising designs

Trimming things out

What can we get rid of?

LEARN FROM HOME
WORKSHOP

Can't make it to a workshop?
Learn from:

- Tutored screencasts
- Sample files
- Access to a Q&A session
- Test material

FIND OUR MORE

SOLD OUT

FIND OUR MORE

Task

Get rid of some stuff!

Let's talk about code!

A change in attitude

A change in attitude

OOCSS – engineering led front-end dev.

Semantics – not what we thought they were.

Pragmatism – we're not developers, we're problem solvers who use development.

Front-ends aren't as simple as they used to be – the internet has changed but our 'best practices' haven't.

The three stakeholders

Who we write our code for.

The three stakeholders

The client – wants a robust, well-built site

The user – wants a fast, consistent experience

The developer – wants a nice, maintainable codebase

Single Responsibility Principle

Every piece of code should do one job, one job only, and do that job very well.

LEARN FROM HOME
WORKSHOP

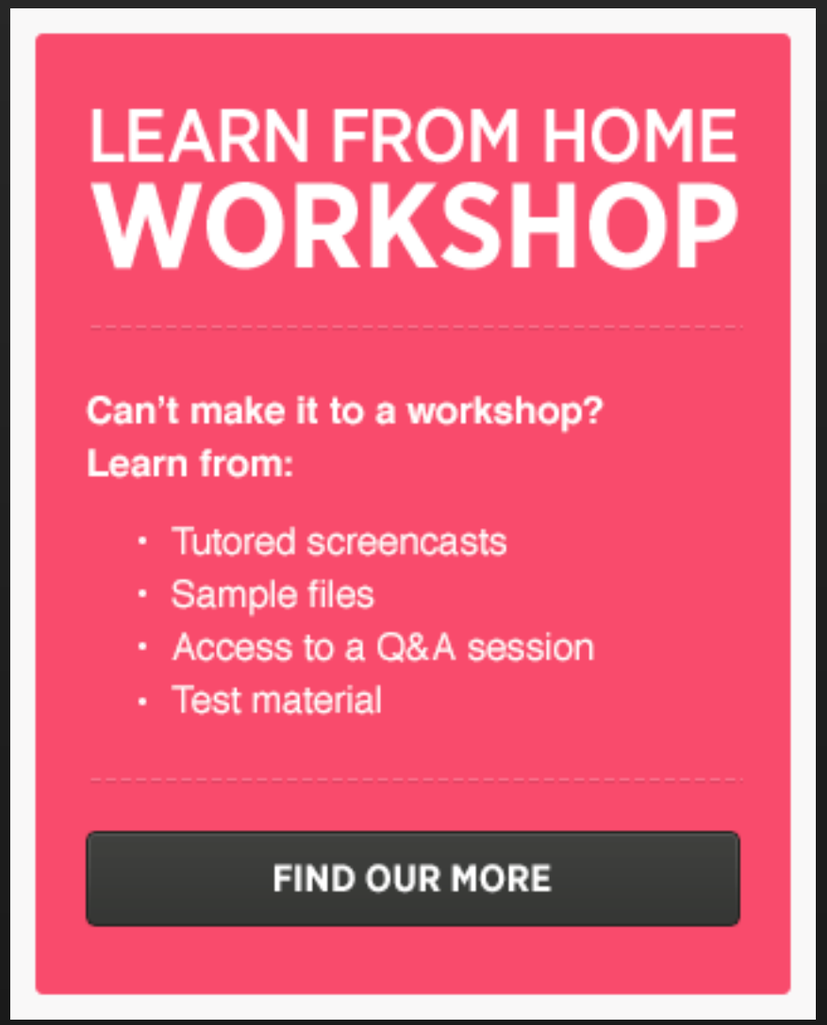
Can't make it to a workshop?
Learn from:

- Tutored screencasts
- Sample files
- Access to a Q&A session
- Test material

FIND OUR MORE

Promotional box

Round cornered box with a pink background?



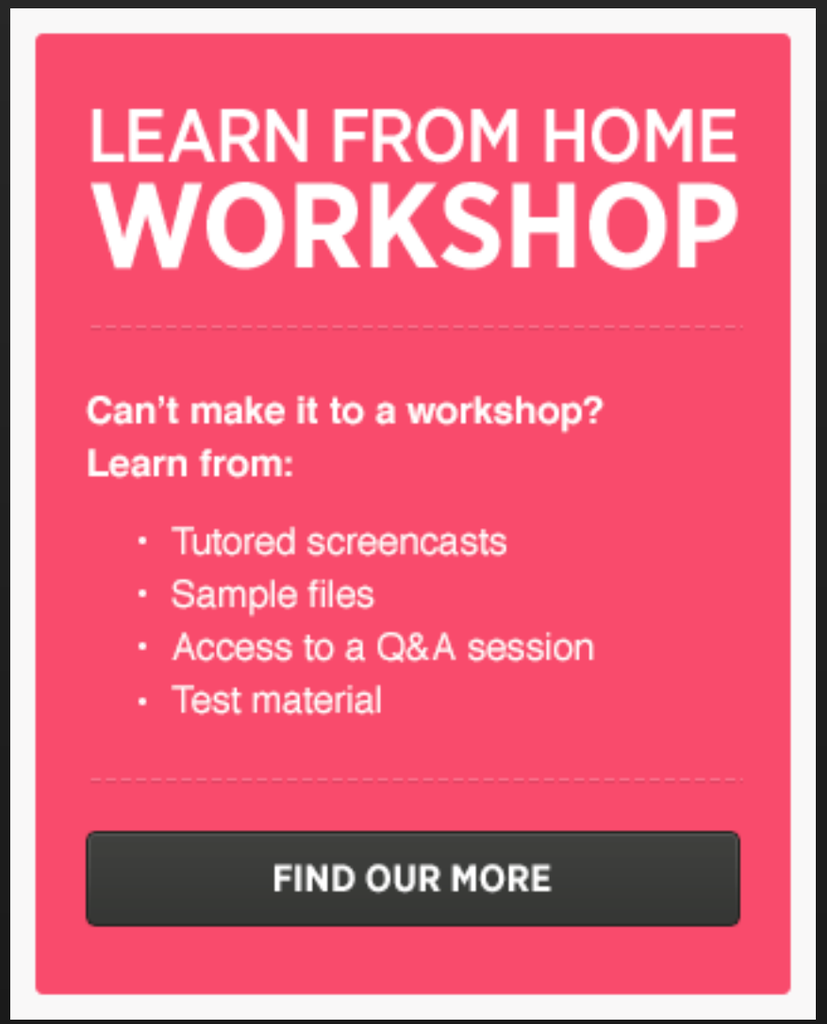

```
<div class="promo-box">
    ...
</div>
```

Promotional box

Box

Round corners

Pink background




```
<div class="box
            box--promo
            box--soft">
    ...
</div>
```

The Single Responsibility Principle

Makes code much more granular

Makes code more reusable and flexible

Allows for a greater number of combinations

> Consectetur dolor fermentum incepo mattis commodo

Naomi Atkinson | Whosit & Whatsit

Task

Pick some components and break them into responsibilities

CSS selectors

Requirements

Predictable – they need to do what we expect

Robust – we don't want things breaking easily

Portable – we need to move our components about

Reusable – we need to make sure we can recycle

Low specificity – all selectors are born equal

Specificity…

…and trying to avoid it.

Specificity

Evil

Hard to manage

Classes

The perfect selector

Selector intent

Being a CSS sniper

```
header ul {}
```

```
.site-nav {}
```

Task

Start scoping some selectors for your components

Semantics vs. sensibility

Semantics vs. sensibility

HTML elements are semantic – predefined and understood by machines.

Classes are not semantic – subjective and only understood by people.

```
<div class="hello">...</div>
<div class="bonjour">...</div>

<div class="hallo">...</div>
```

```
<article class="box  box--inverse  post">
    <h1 class="gamma  post__title">Hello</h1>
    <p class="post__excerpt">Lorem ipsum...</p>
</article>
```

```
<article class="box  box--inverse  post">
    <h1 class="gamma  post__title">Hello</h1>
    <p class="post__excerpt">Lorem ipsum...</p>
</article>
```

```
<article class="box  box--inverse  post">
    <h1 class="gamma  post__title">Hello</h1>
    <p class="post__excerpt">Lorem ipsum...</p>
</article>
```

Semantics vs. sensibility

Describing content is redundant – the content does that itself!

Classes' primary role is as a styling hook – use them as such.

```
<div class="bordered  red  small  left">
    ...
</div>
```

```
<div class="box  box--round">
    ...
</div>
```

```
<h1 class="red">...</h1>
```

```
<h1 class="brand-color">...</h1>
```

Naming things

There are only two hard things in Computer Science: cache invalidation and naming things.

— **Phil Karlton**

Naming things sucks

It's worth doing properly.

It's worth a lot of thought.

It's very difficult.

Objects, abstractions and helpers

Agnostic

Loosely named

Abstract naming

Can be applied to any type of content

LATEST POSTS

Hashed Classes in CSS »

For a long time now I have advised people not to use IDs in CSS. Use them in your JS, sure, and as fragment identifiers in HTML, but do not use them to style things in CSS.

‘Scope’ in CSS »

One thing you will no doubt be familiar with, as a web developer, is the idea of scope. Wikipedia’s introduction to the subject: “In computer programming, a scope is the context…

The flag object »

It all started with the media object. That one snippet of CSS, by Nicole, got me fully sold and hooked on OOCSS. This article will only really make sense if you, too, are familiar with the…

Visit the blog »

```
<ul class="ui-list">
    <li class="ui-list__item">...</li>
    <li class="ui-list__item">...</li>
    <li class="ui-list__item">...</li>
    <li class="ui-list__item">...</li>
</ul>
```

Task

Name some objects and abstractions!

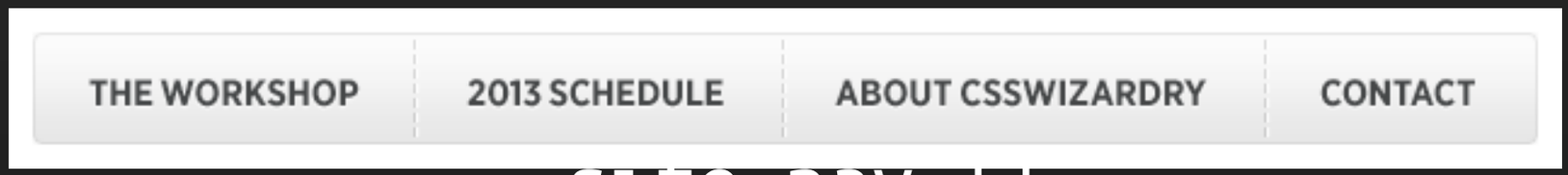
Modules and components

Much more explicitly named.

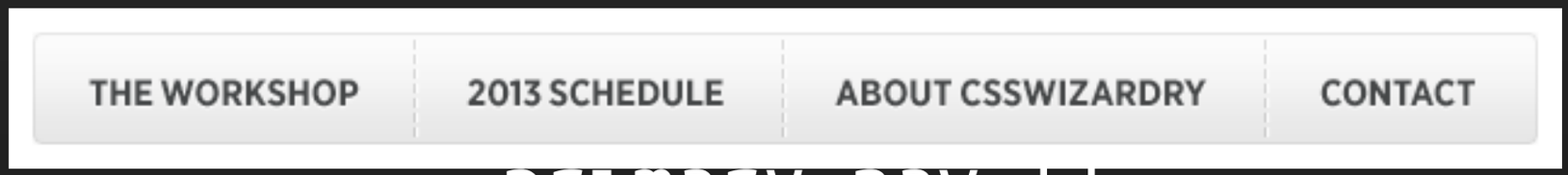
Align themselves to content.

```
<ul class="ui-list  products">
    <li class="ui-list__item  products__item">...</li>
    <li class="ui-list__item  products__item">...</li>
    <li class="ui-list__item  products__item">...</li>
    <li class="ui-list__item  products__item">...</li>
</ul>
```

```
<ul class="ui-list  users">
    <li class="ui-list__item  users__item">...</li>
    <li class="ui-list__item  users__item">...</li>
    <li class="ui-list__item  users__item">...</li>
    <li class="ui-list__item  users__item">...</li>
</ul>
```



.site-nav {}




```
.primary-nav {}
```

SOLD OUT

.btn--sign-up {}

FIND OUR MORE

.btn--learn-more {}

SOLD OUT

.btn--primary {}

FIND OUR MORE

.btn--secondary {}

Task

Name some components!

Naming conventions

What's in a name?

More than just what something is called.

Tells us about what it does…

How it might behave.

BEM

Thanks, Yandex!

Block

```
<div class="block  block--modifier">
    <span class="block__element">...</span>
</div>
```

Element

```
<div class="block  block--modifier">
    <span class="block__element">...</span>
</div>
```

Modifier

```
<div class="block  block--modifier">
    <span class="block__element">...</span>
</div>
```

BEM-style naming

Tells us what the markup is doing

Tell us how and where to use CSS

Tells us how chunks of markup are related (if at all)

‘Scopes’ our CSS with a namespace

Example

```
<article class="blog-post  blog-post--is-guest-post">
    <h1 class="blog-post__title">...</h1>
    <p class="blog-post__intro">...</p>

</article>
```

Example

```
<div class="box  profile  pro-user">
    <img class="avatar  image" />
    <p class="bio">...</p>
</div>
```

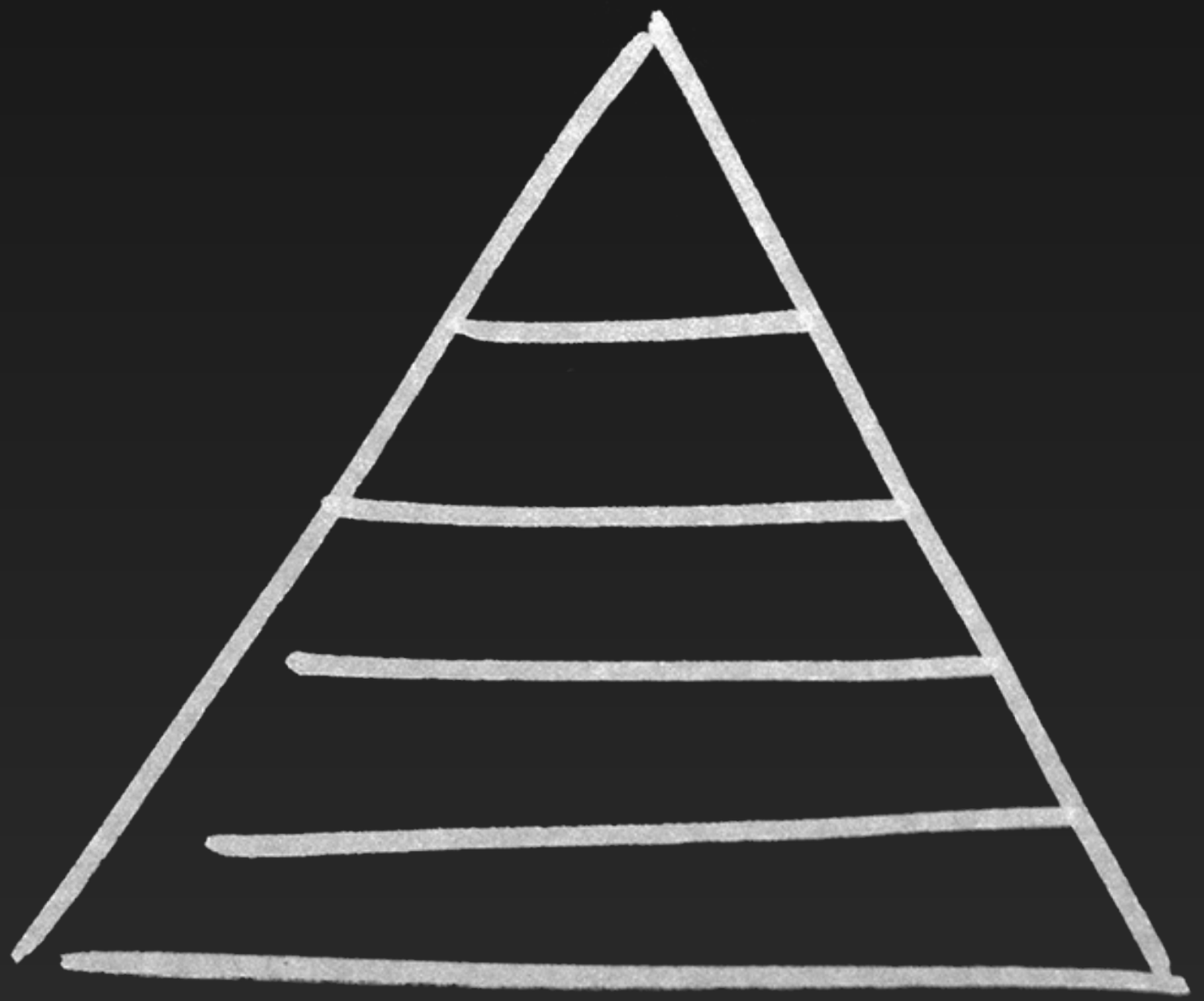
Example

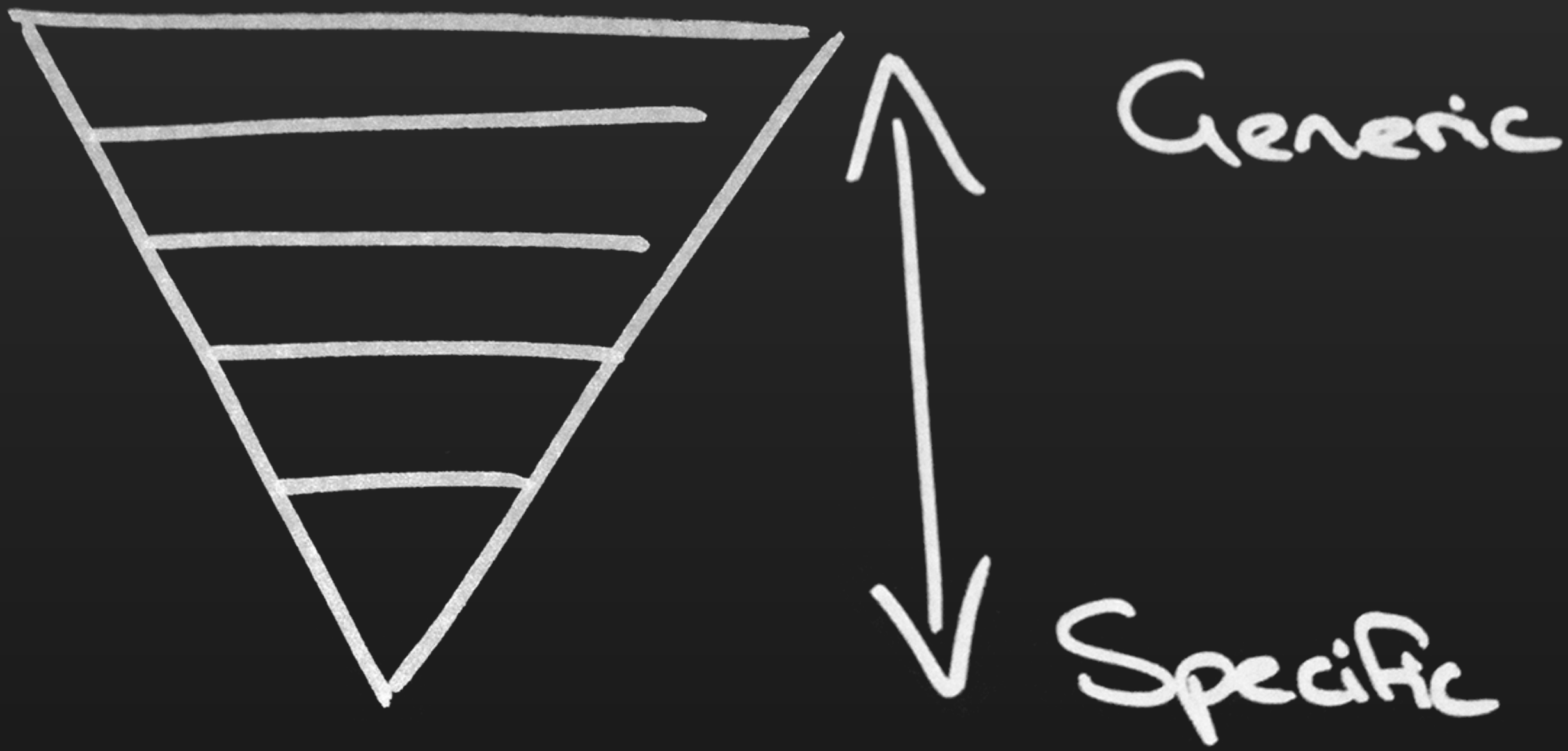
```
<div class="box  profile  profile--is-pro-user">
    <img class="avatar  profile__image" />
    <p class="profile__bio">...</p>

</div>
```

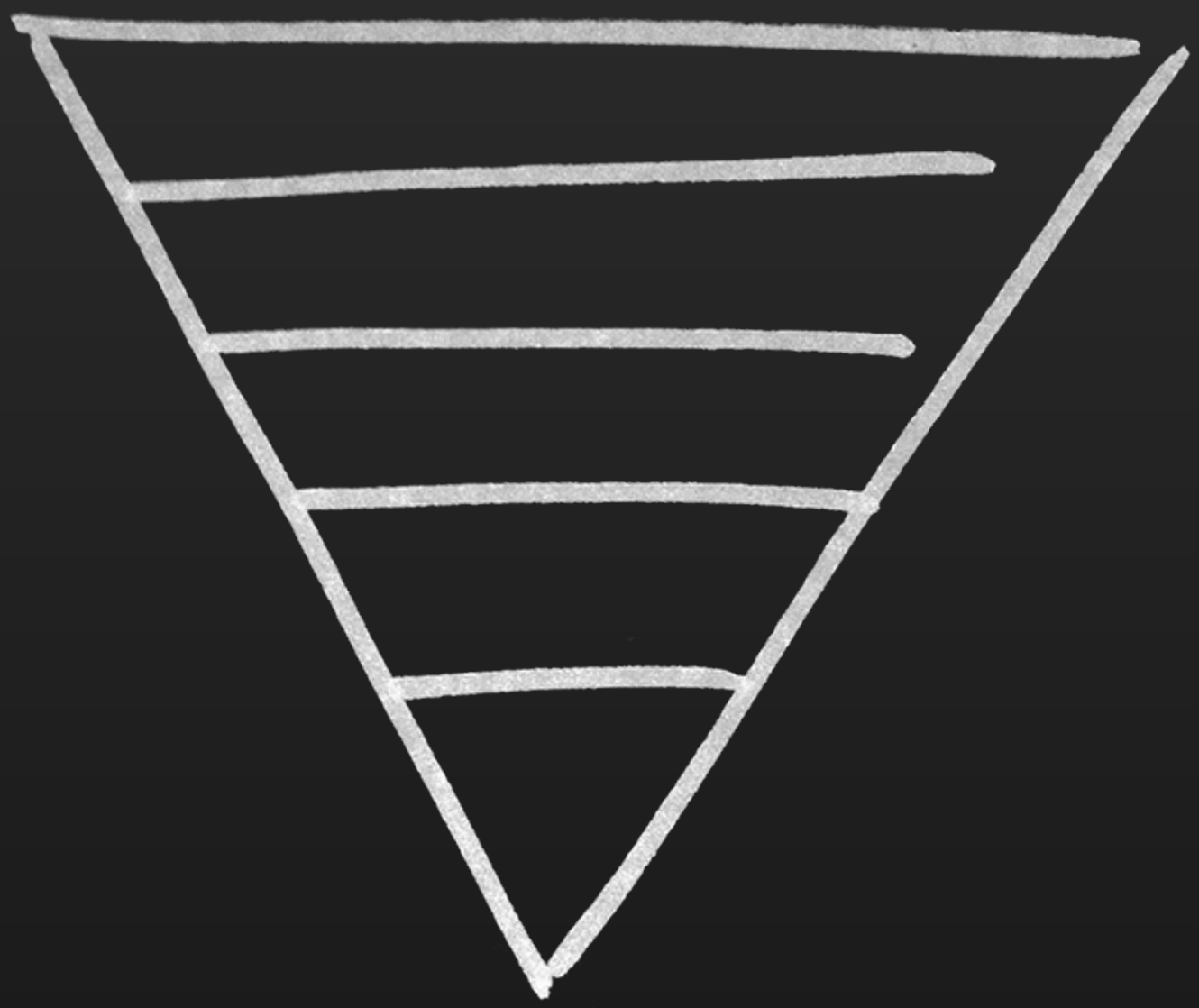
Task

Start giving some objects and components suitable names

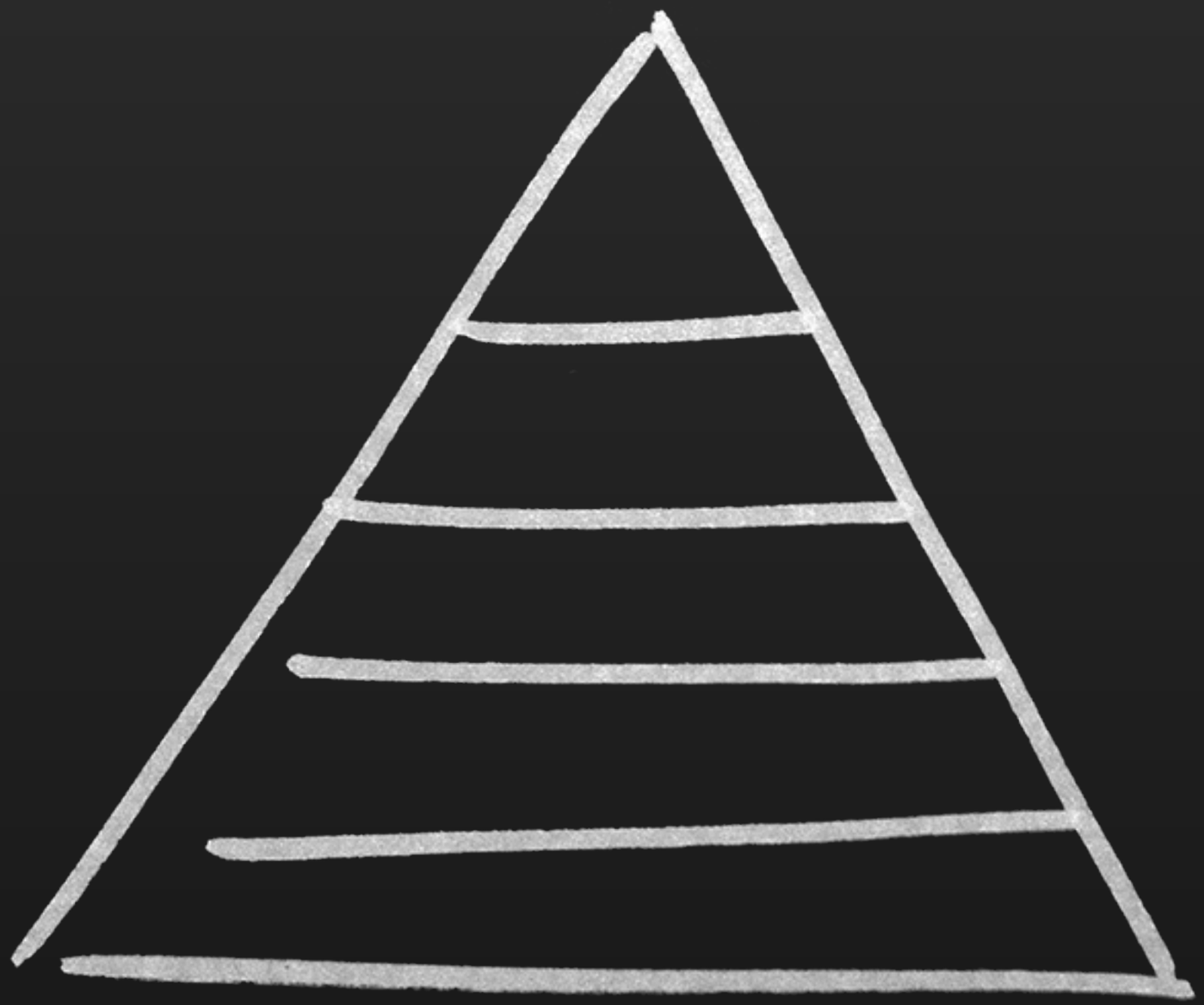
CSS architecture




Generic
Specific



```
*
h1, p, ul
.grid
.masthead

.is-error
```



Shearing Layers

Task – 5 min

How would you build a house?

Shearing layers

Foundations – reset, normalize.css, global box-sizing

Structure – page layout, design patterns, abstractions

Fixtures/fittings – components, modules

Decorations – design, ‘look and feel’

Ornaments – style trumps, themes, takeovers

"A classic and probably a work of genius" —JANE JACOBS, author of *The Death and Life of Great American Cities*

HOW BUILDINGS LEARN

What happens after they're built

New Orleans, 1857

The same two buildings, 1993

STEWART BRAND

creator of *THE WHOLE EARTH CATALOG*

Lorem ipsum

Dolor sit amet.

Lorem ipsum

Dolor sit amet.

Lorem ipsum

Dolor sit amet.

Lorem ipsum

Dolor sit amet.

Lorem ipsum

Dolor sit amet.

Task

Pick some components and break them into layers

Let's look at a our codebase…

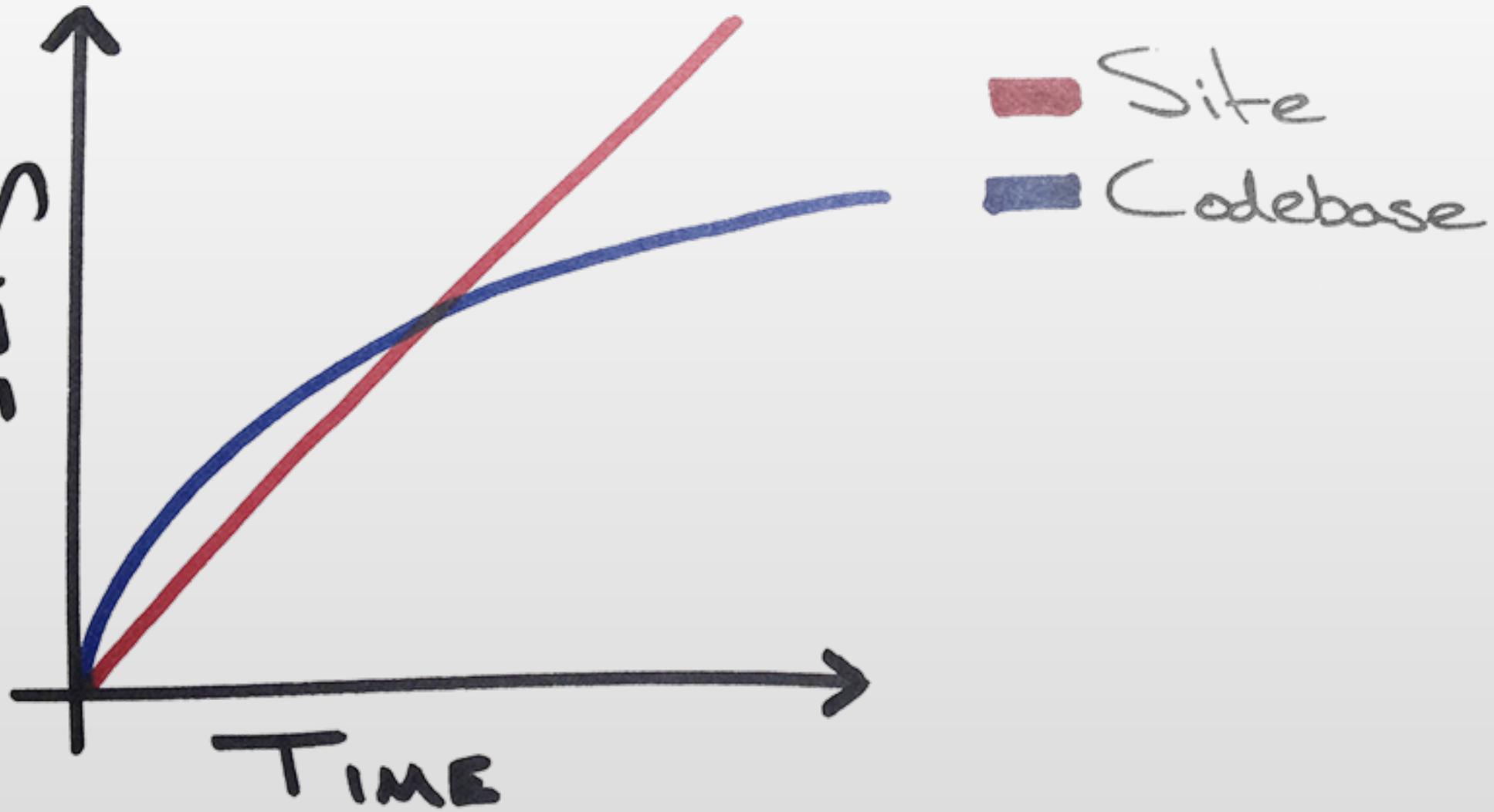
Our codebase

Aagh!

Looks pretty huge!

Very split out.

Lots and lots of files.


Size
Time
Site
Codebase

Task – 15 min

Take a look over the codebase

Task

Get building!

Managing layout

Layout as a component

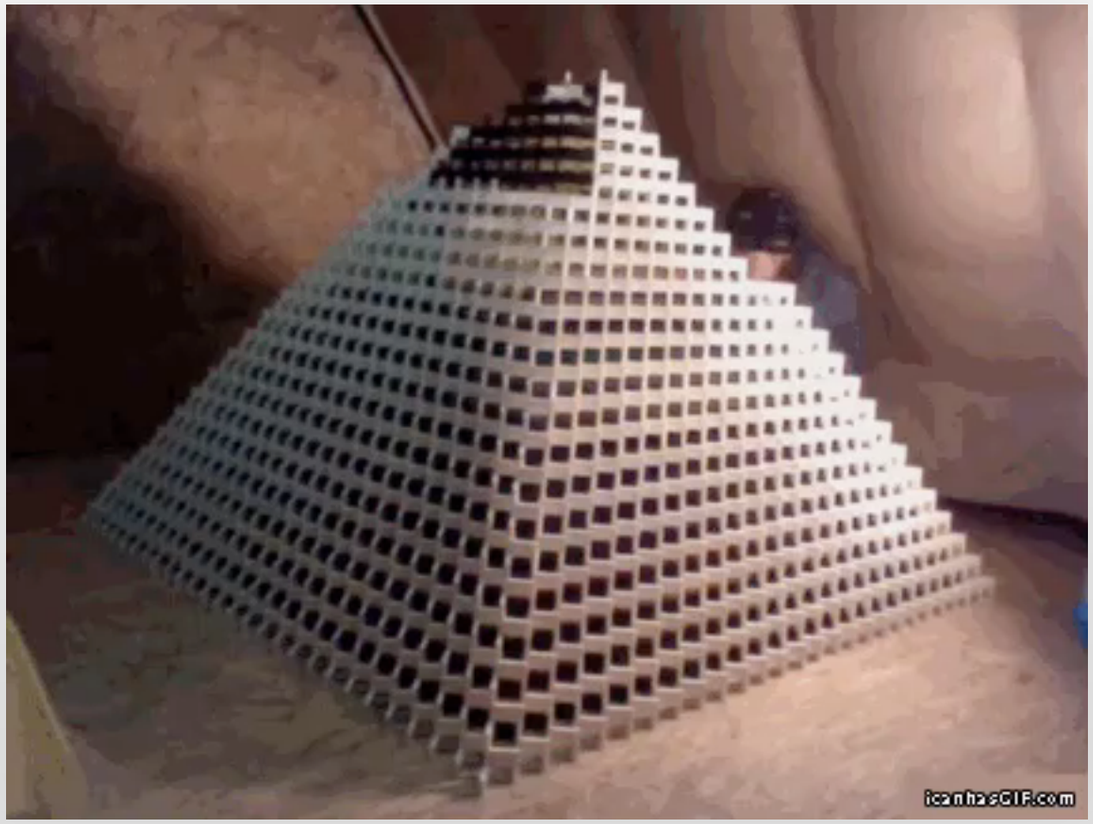
Use a layout system (not the same as a grid system).

Components are useless if you put widths on them.

Treat layout as its own separate entity.

Layout systems are like shelves.

Your UI without a layout system



Your UI with a layout system

Putting it all together

Putting it all together

Break everything apart.

Split it into layers.

Do not consider layout until the very end.

Recap

Keep selectors short, portable, reusable and low specificity (i.e. classes).

Write everything with reuse in mind.

Write code for the right people.

Break everything into shearing layers.

Manage layout as its own entity.