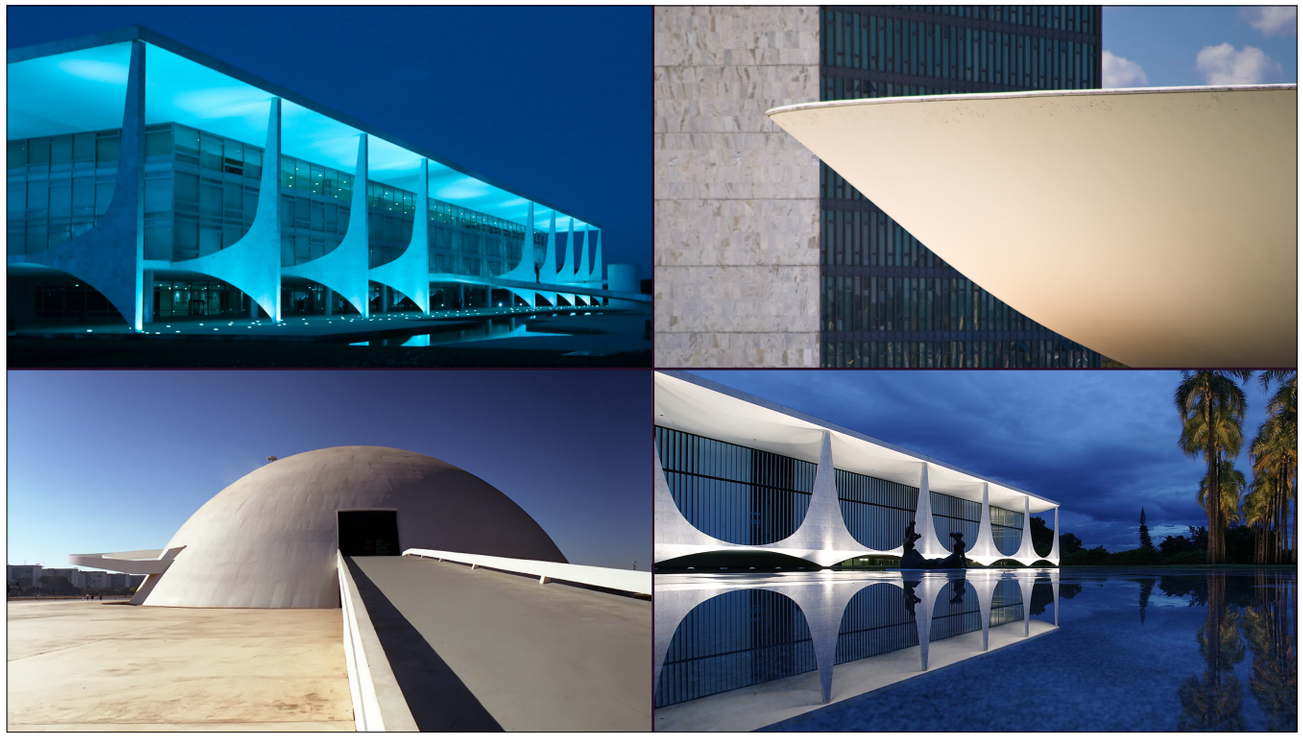
Designing Systems

Theory, Practice, and the Unfortunate In-between

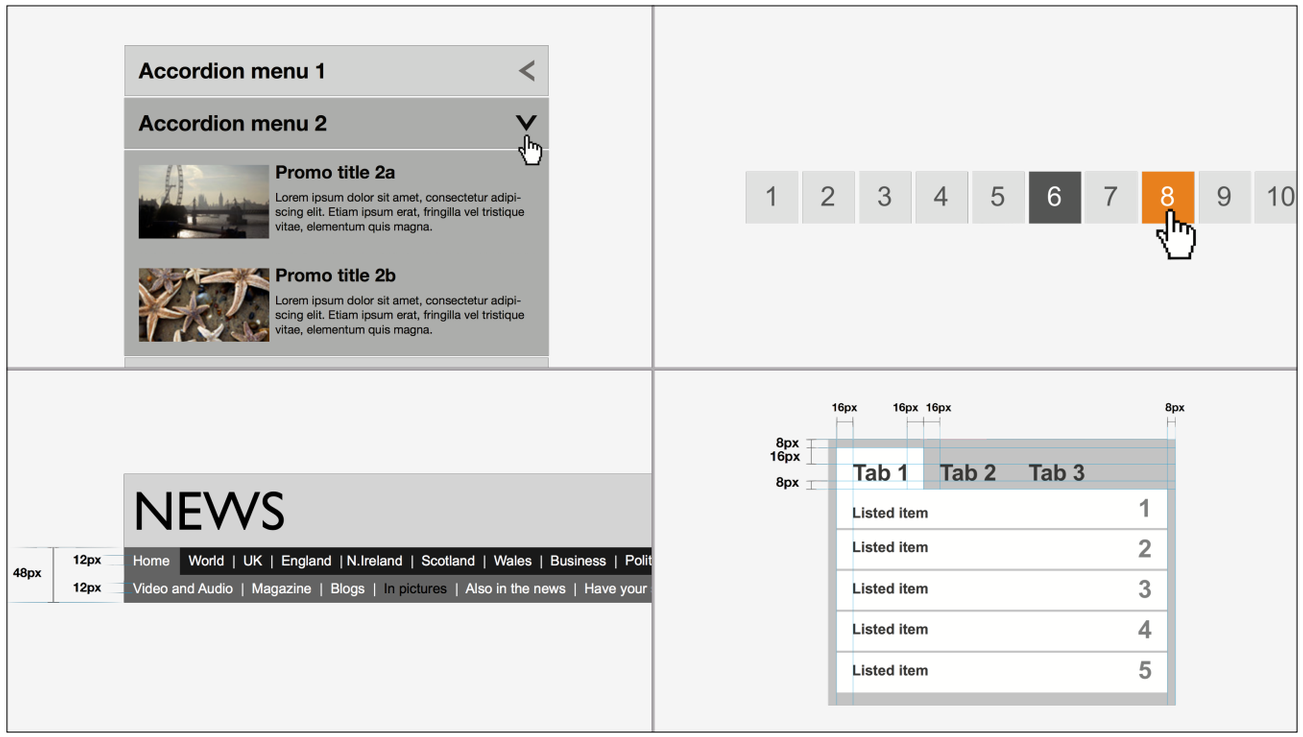
@paulrobertlloyd

Design Exchange Nottingham / 10 August 2016

- Does anyone recognise this city?
- It’s Brasilìa, capital of Brazil (not Rio or São Paulo: good pub quiz knowledge!)

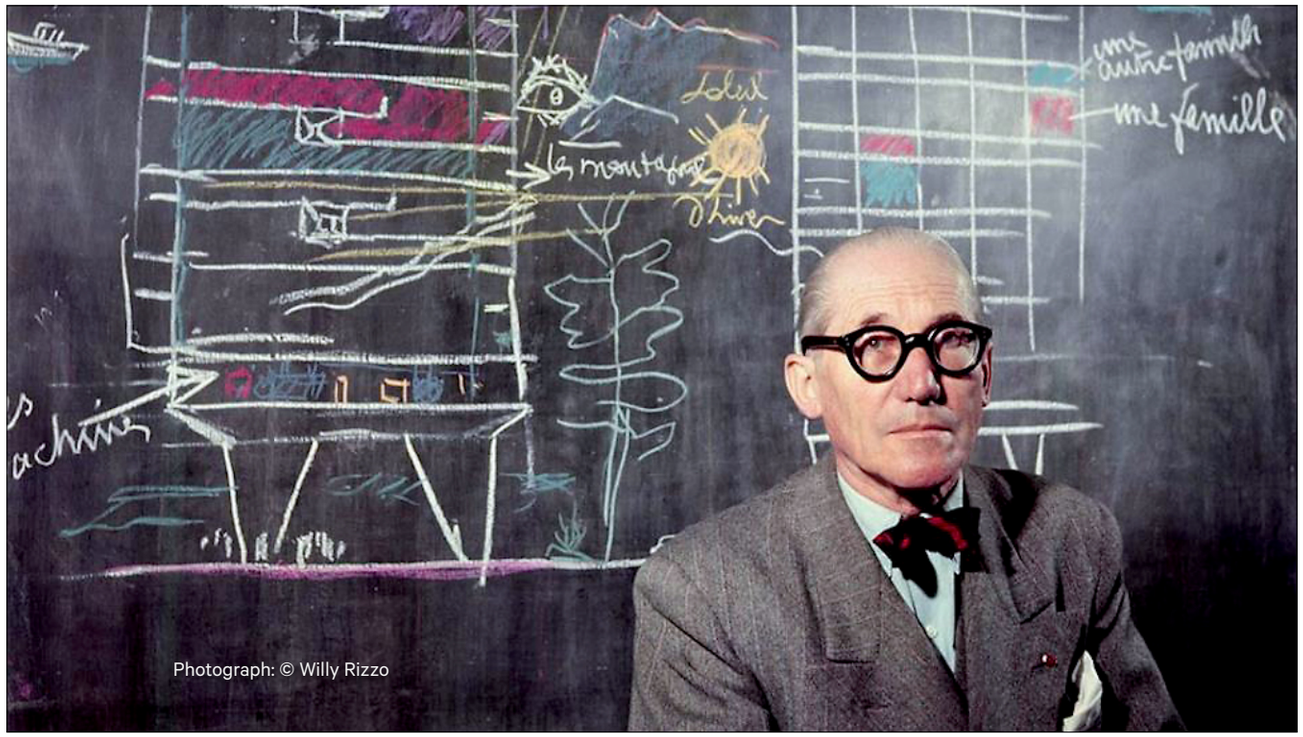


- I've long been fascinated by the city, got to see it first hand in 2011.
- Saw a number of **repeating motifs** in Oscar Niemeyer signature buildings:
    - Elongated cuboids sat alongside soft curves
    - Buildings appear suspended in air, or floating on water
    - Dramatic entrances, with long ramps that ascend or descend into reception spaces
- What analogies could be drawn between what I saw there and the patterns we use on the web?

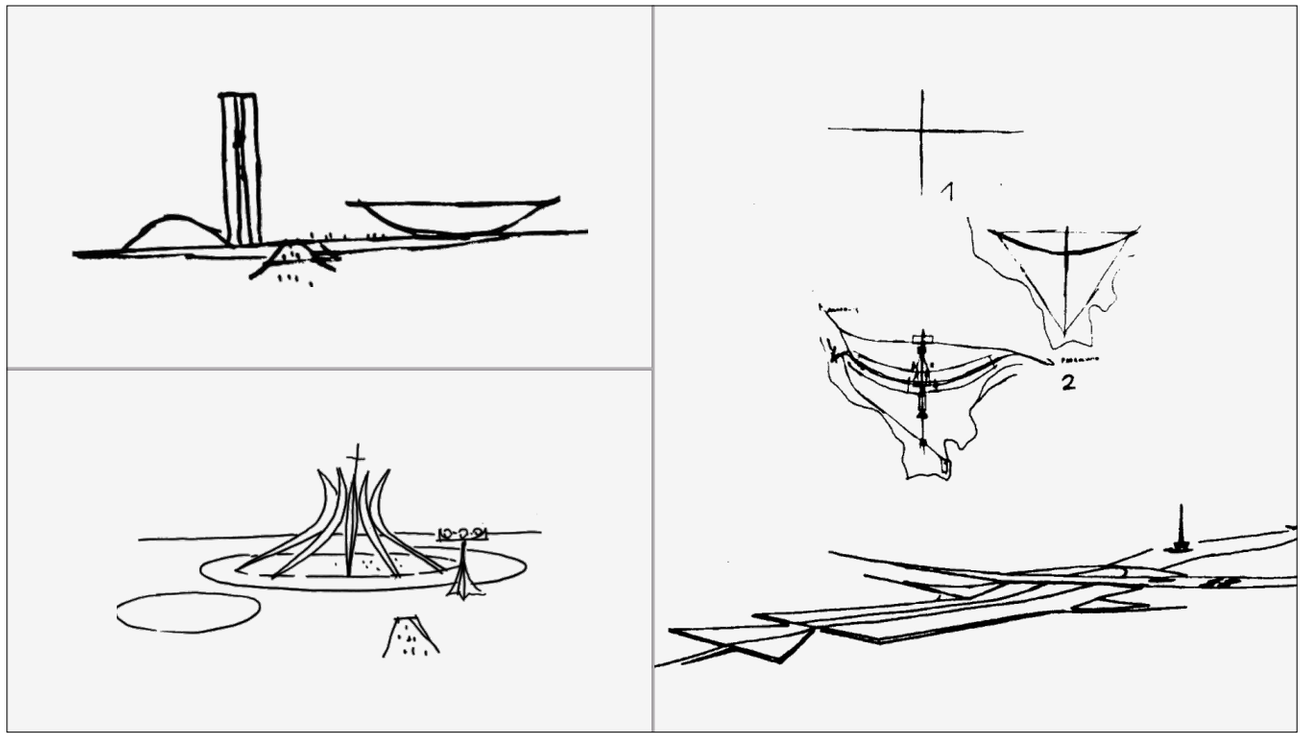


- When I spoke at Oxford Geek Night a few months later, I compared the design of Brasilìa with the BBC's GEL internal design language.
- Both use **identifiable and reusable patterns**, which are combined to produce a **cohesive whole**.
- However, I've started to wonder if a more pertinent lesson was to be found?

- Brazil's constitution had long stated that the capital should be moved from Rio de Janeiro to a more central location. The election of President Juscelino Kubitschek (shown on the left here) in 1956 saw this realised.
- Campaigned on a promise of "fifty years of prosperity in five", and building Brasilìa was a key pledge.
- Lúcio Costa (shown right) won the contest for its layout, with Oscar Niemeyer tasked with designing its buildings.



- Both disciples of Le Corbusier, the Swiss-French architect and pioneer of modern urbanism.
- Le Corbusier believed good architecture would lead to good society.
- Cities should be rigidly zoned, with standardised “machines for living in”, although typically more consideration was given to the car than the individual.
- Costa and Niemeyer had **a blank canvas** upon which they could turn these ideas into practice.

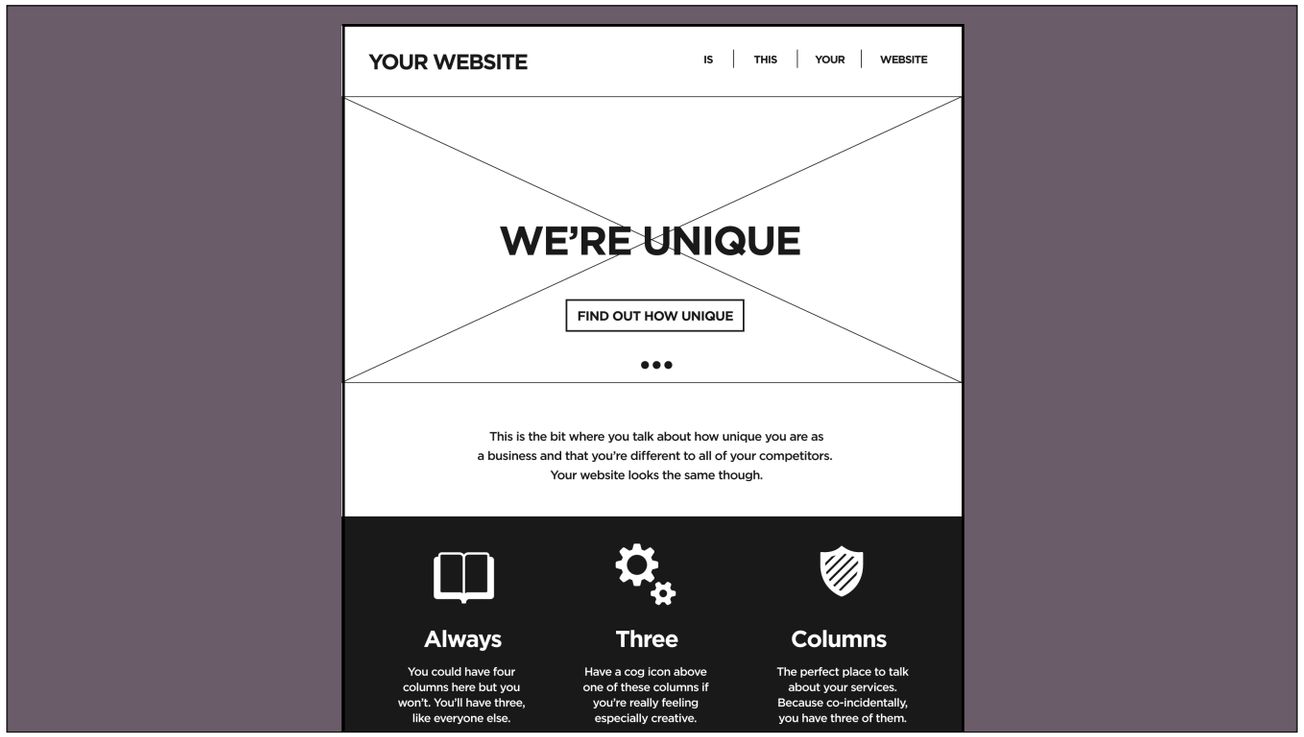


- They believed Brasilìa could transform Brazil’s heavily stratified society into a more egalitarian one.
- In their model utopia, governors and ambassadors would live next door to janitors and labourers, and everyone would use the same entrances and share the same spaces.
- City completed in just 41 months, yet these ideals would never be realised.

- As people from across the country came to the city looking for work, **city grew** rapidly, yet Costa's **plan limited the amount of residential space** available.
- Workers who built the city were banished from it; only after a long fight were they allowed to remain in distant satellite towns.
- Segregated from middle-class Brazilian society, they would spend their meagre salaries on long commutes into the capital.
- Servants who did live in the city ended in spaces worse than those they would have done otherwise.
- The real Brazil spoiled this utopian vision, and its class structures remained.
- **So, what went wrong?**

The Shock of the New: Trouble in Utopia
BBC Television, 1980

- Robert Hughes: “Nothing dates faster than people’s fantasies about the future
- Hughes was typically abrasive in his comments, and a lot has changed since this was filmed his documentary *The Shock of the New* in 1980.
- What the city achieves in elegant rationality, it suffers in sterile inhumanity.
- Built around the car, not people. Streets make way for highways.
- Memory of my visit was walking along unpaved verges, or attempting to cross streets or enter buildings.



- So, how does this relate to web design precisely?
- Well-meaning developers use frameworks like Bootstrap/Foundation, much like Costa/Niemeyer imported Le Corbusier's ideas.
- Kubitschek: "fifty years of prosperity in five". Frameworks: "fifty days of development in five".
- Conceived to solve problems within the organisations that created them.
- Their problems may not be your problems.
- Much like, Brasilìa, these frameworks offer an example of designing without sufficient context.
- Offering patterns designed to appeal to the broadest possible use case, is it ant wonder that all websites look the same.

- Lúcio Costa and Oscar Niemeyer had a black canvas, but the opposite was true for Frank Pick.
- Pick joined Underground Electric Railways of London (UERL) – or Underground Group – in 1906.
- London was home to over 6m people.
- Transport infrastructure run by a number of separate private companies.
- Group had constructed three new tube lines, but these were not attracting enough numbers.
- Pick was put in charge of publicity in 1909.

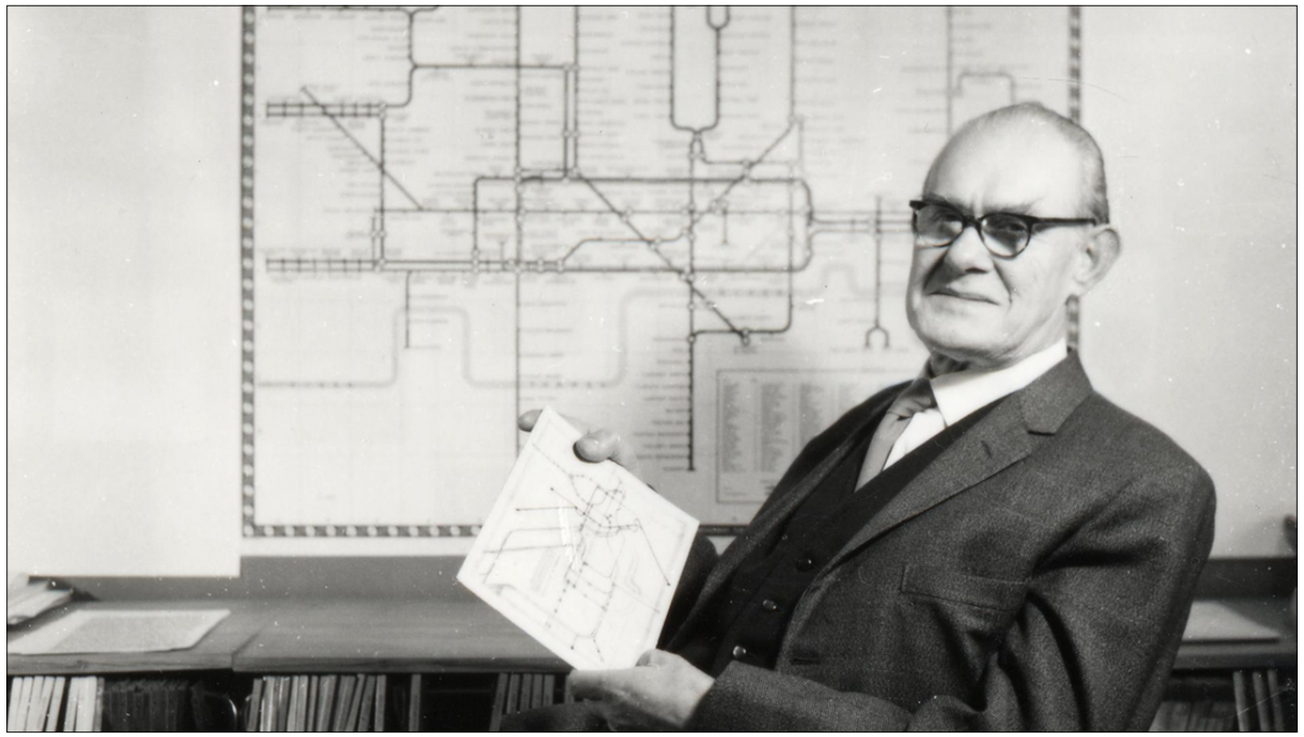
- Pick wasn’t a designer or artist, so judged what worked through trial and error.
- First initiative was to improve the appearance of stations, whose chaotic design hardly inspired confidence.
- Standardised poster sizes, limited their number, and arranged them on a grid.
- To ensure the stations names would stand out, he placed large red circle behind name board.
- This would later inspire the design of the now familiar Roundel.

- Pick wanted a font to distinguish the groups announcements from advertising.
- Approached Edward Johnston to design a proprietary typeface that should have “the bold simplicity of the authentic lettering of the finest periods” and belong “unmistakably to the twentieth century”.
- Johnston’s typeface was inspired by simple, hand drawn letters he’d seen on a tradesmen’s wagon.
- Some of Johnston’s contemporaries derided his font, calling it a betrayal of well-designed lettering.
- But it solved the problem it was created to address.

Magpie Studio

- The Underground is currently celebrating the centenary of it’s introduction.
- Of the posters it commissioned, this my favourite.

- As Pick rose up the ranks, he was no longer just commissioning posters and identity programmes, but buildings too.
- In 1927 commissioned architect Charles Holden to design a new head office, 55 Broadway, and a series of iconic stations on the extended Piccadilly line.
- By 1933 Pick had become the chief executive of a newly combined public body, the London Passenger Transport Board.



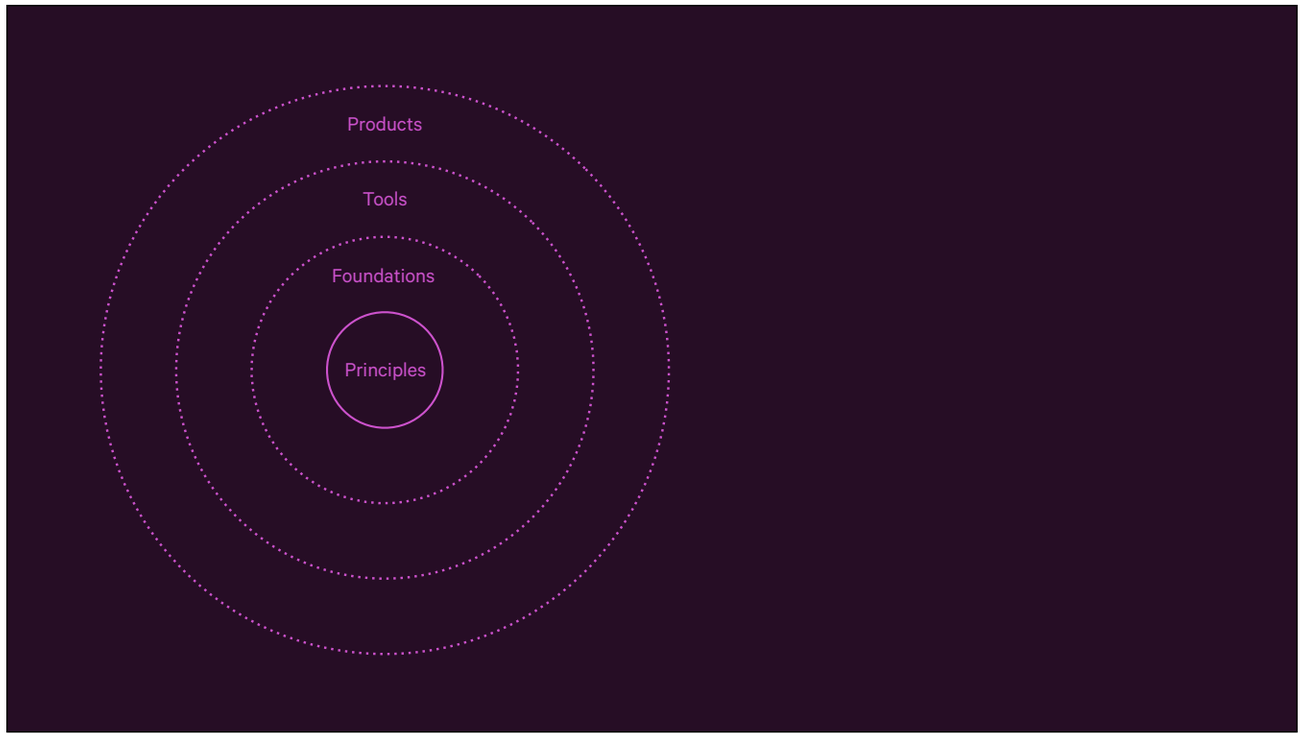
- Most notable piece of design created during his tenure wasn’t instigated by Pick.
- Underground map was conceived by Henry Beck, a young engineering draftsman.
- Beck did away with the geographical accuracy, drew lines on vertical, horizontal and diagonal axes.

> The test of the goodness of a thing is its **fitness for use**. If it fails on this first test, no amount of ornamentation will make it any better; it will only make it more expensive, more foolish.
>
> — Frank Pick, 1916

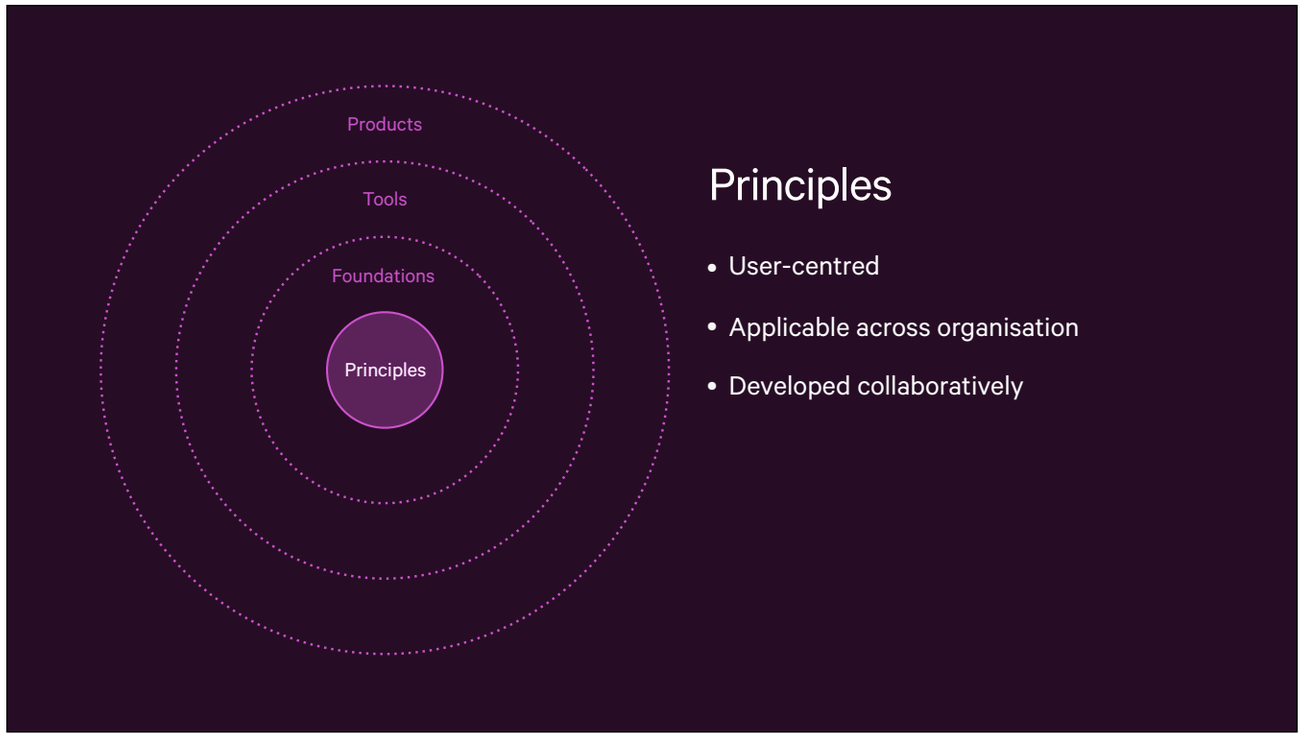
- Both Johnston’s typeface and Beck’s map satisfied Pick’s test of fitness for use.
- Pick saw the underground not simply as a machine for moving people, but something that could contribute positively to the urban environment and the communities it served.
- Became almost a patron for the arts, which gave the system humanity.
- Pick’s also said “art should come down off its pedestal and earn its living”, with the the tube referred to as the longest art gallery in the country.
- 75 years after his death, Pick’s ethos remains central to design of London’s transport network.
- How have the high standards Pick set been maintained over that time?
- The answer: a design system.

What is a design system?

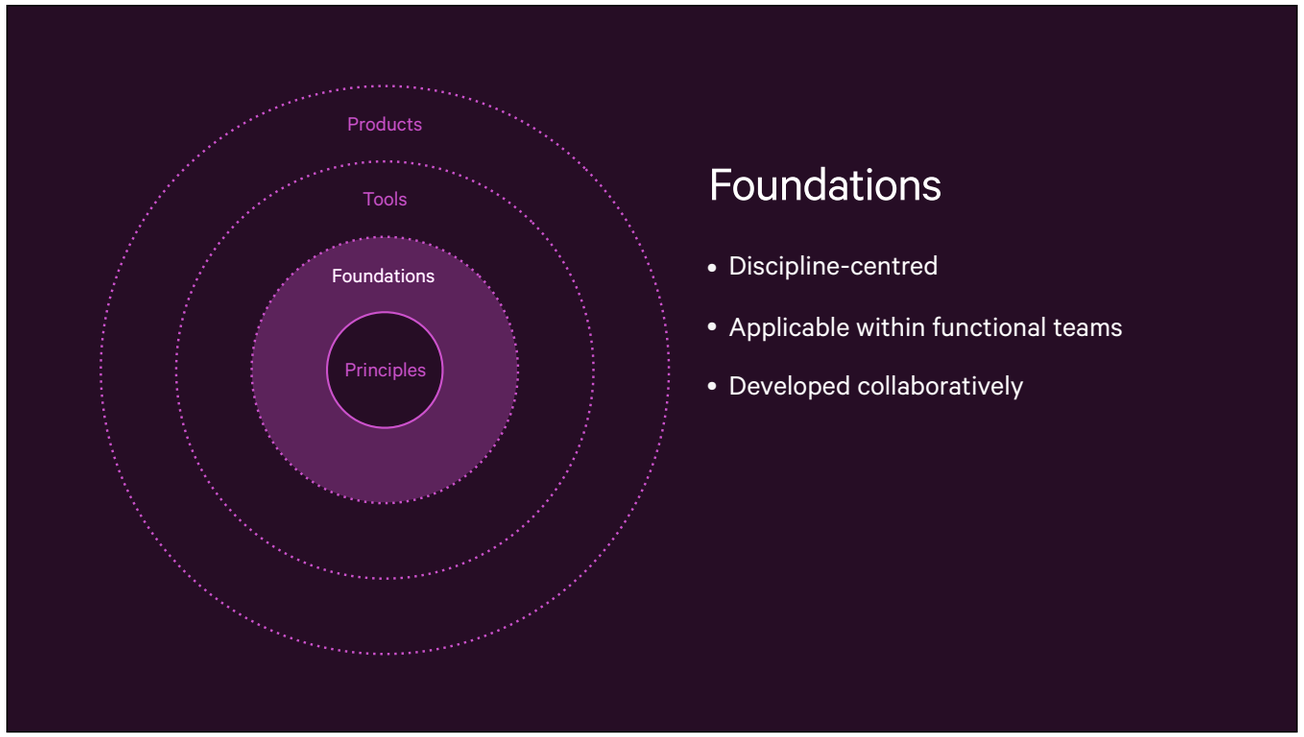
- But what do we actually mean when we say ‘design system’?
- In discussing this with colleagues, no one was able to give a definitive answer.
- Here’s my take.



- Easy to think a design system is simply a collection of style guides and colour palettes.
- I believe we should consider a design system in broader terms



- **Start with a core ethos**, supported by a set of design principles.
- Kate Rutter calls these: short, insightful phrases that act as guiding lights and support the development of great product experiences.
- Should be:
    - Focused primarily on user needs.
    - Unique to organisation; what makes product different from competitors?
    - Reviewed periodically, but otherwise remain fairly static.
- Consider them the constitution for your design system.



- On top of these principles a set of **practitioner-centred guidelines**.
- Focused on how different disciplines should embed these principles in their work.
- Other disciplines should be consulted when creating these.
    - i.e. any colour guidance should acknowledge the systems used by different professions
- Examples of foundational guidelines...

Designers: brand identity guidelines

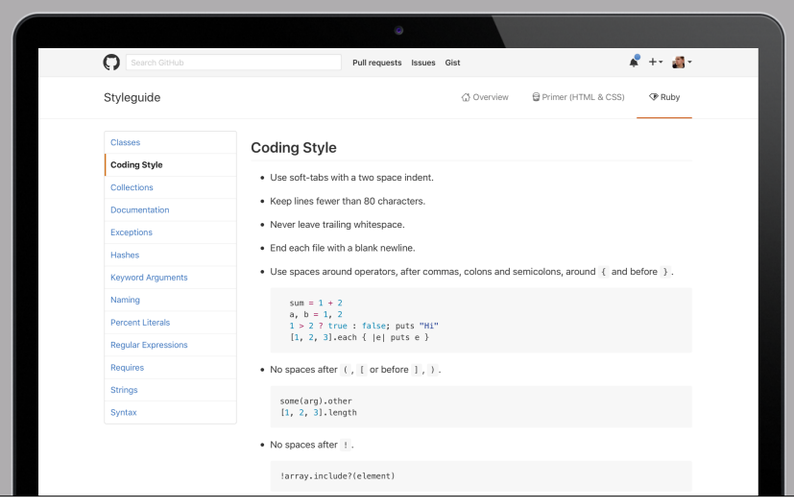
- Brand identity manuals used to be paper-bound, multi-page tomes.
- They document logo usage (including variants, spacing and placement), fonts, colour palettes, textures, patterns, iconography, photography etc.

Writers: editorial style guides

- The tone of voice will significantly **contribute to the distinctiveness** of your product or service.
- Editorial and content guidelines are the most important aspect of a design language.
- Style guides cover preferred usage of particular words and grammatical choices.

Developers: Code styles/conventions

editorconfig.org



- Code may not be seen by users, but guidance ensures it's consistent, predictable and maintainable.
- The correct answer to question "tabs or spaces" isn't your personal preference, but established conventions.
    - Like to include an `.editorconfig` file in your project
    - Add relevant linting tasks to build process

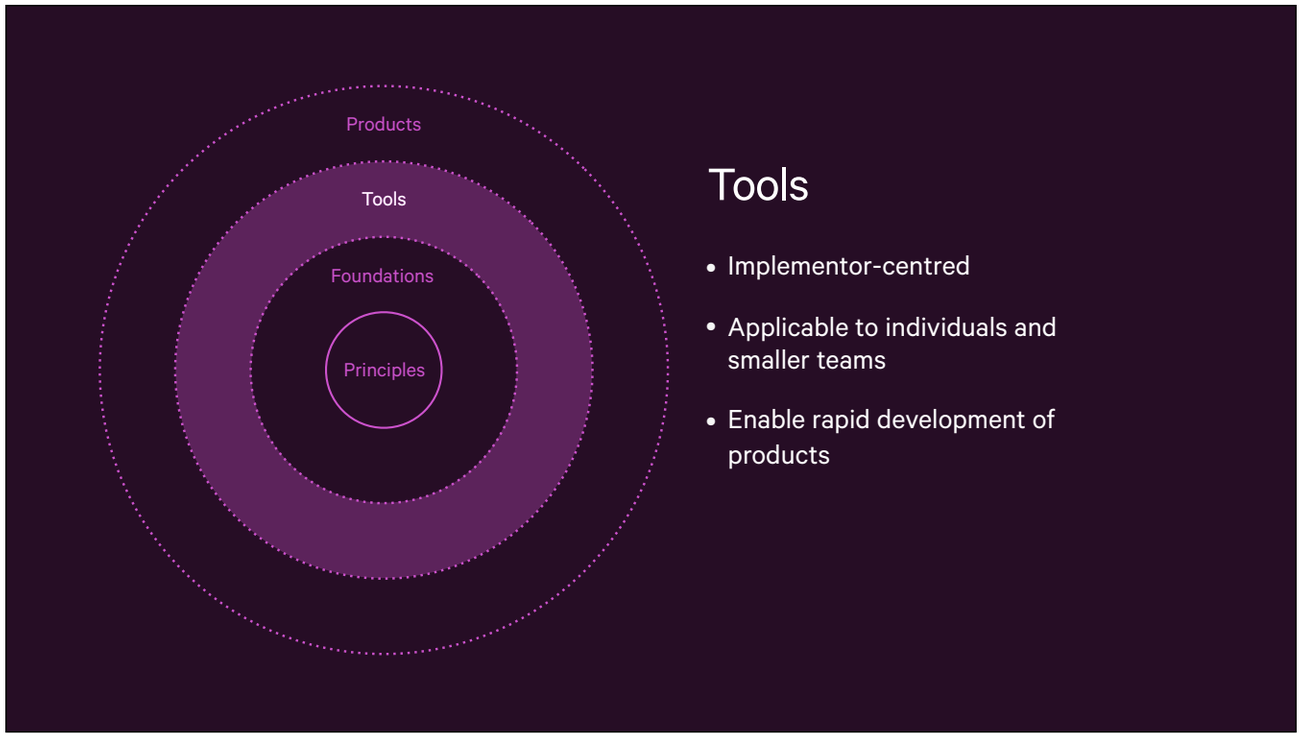
Shared vocabularies

There are only two hard things in computer science: cache invalidation and **naming things**.

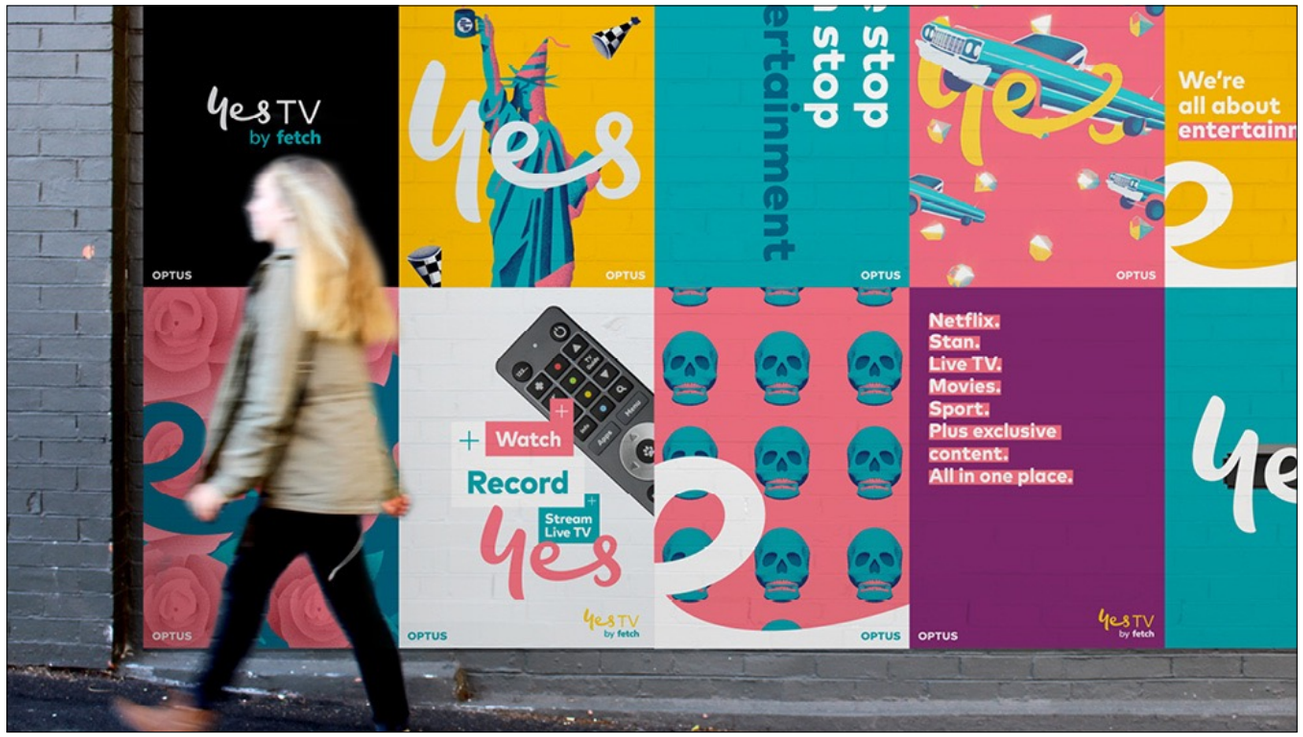
— Phil Karlton

- The single concept that ties these documents together is that they help create shared vocabularies.
- This means defining names, which is hard.
- Organisations may have a number of shared vocabularies already.
- Guardian example: *pixies* and *super-pixies*, *bentos* and *mini-bentos*, names which gave no hint.
- Names should be obvious where possible, and certainly documented.

- When starting a project, work collaboratively to come up with names.
- My colleague Charlotte Jackson describes a collaborative workshop exercise:
    1. Cut up an interface into its component parts
    2. Group similar components
    3. Everyone comes up with a name for each group
    4. Once everyone has thought of a name, reveal them!
    5. Discuss



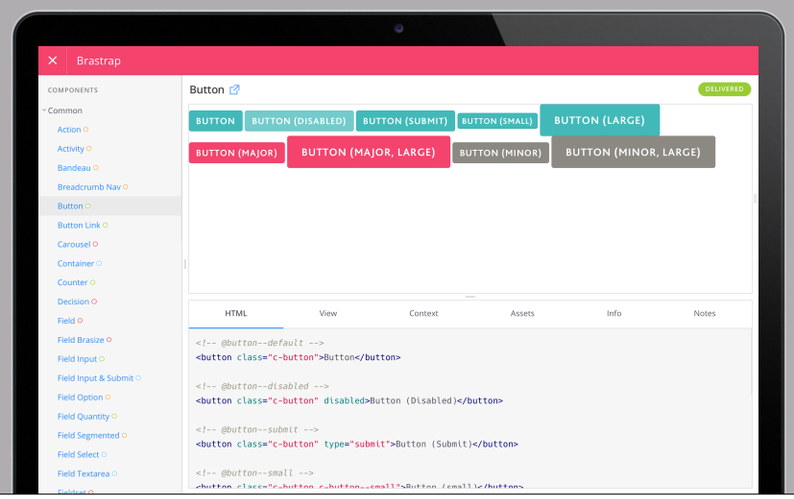
- Sandwiched somewhere between products and these design foundations are a set of tools.
- **Focused on helping the individual** get things done.
- Exist as a bridge to help in the design and development of different products.



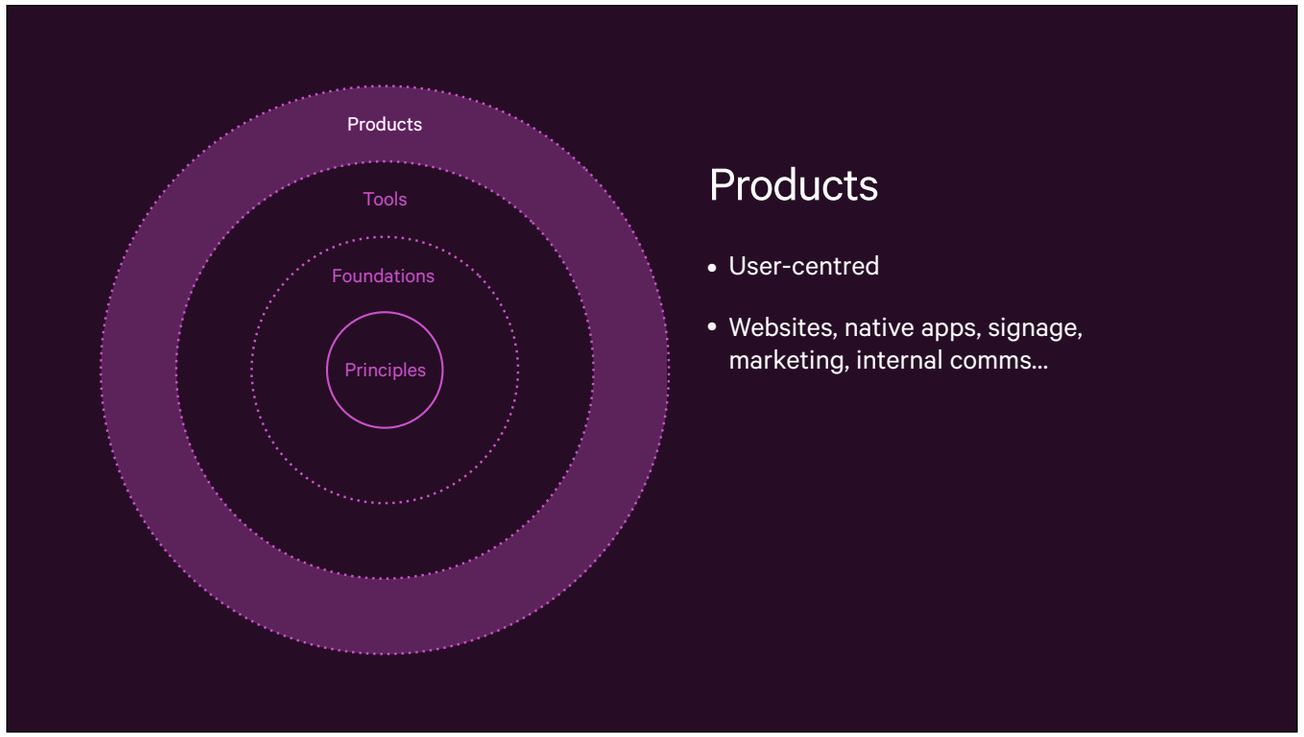
- Visual design language:
  - Similar to brand identity guidelines
  - Concentrate on the **application** of brand identity.
  - New Optus brand is pretty restrained, but enlivened by its accompanying design language.
  - Can evolve to keep brand fresh, current and appease changing tastes, while identity remains unchanged.

Component libraries

http://fractal.build

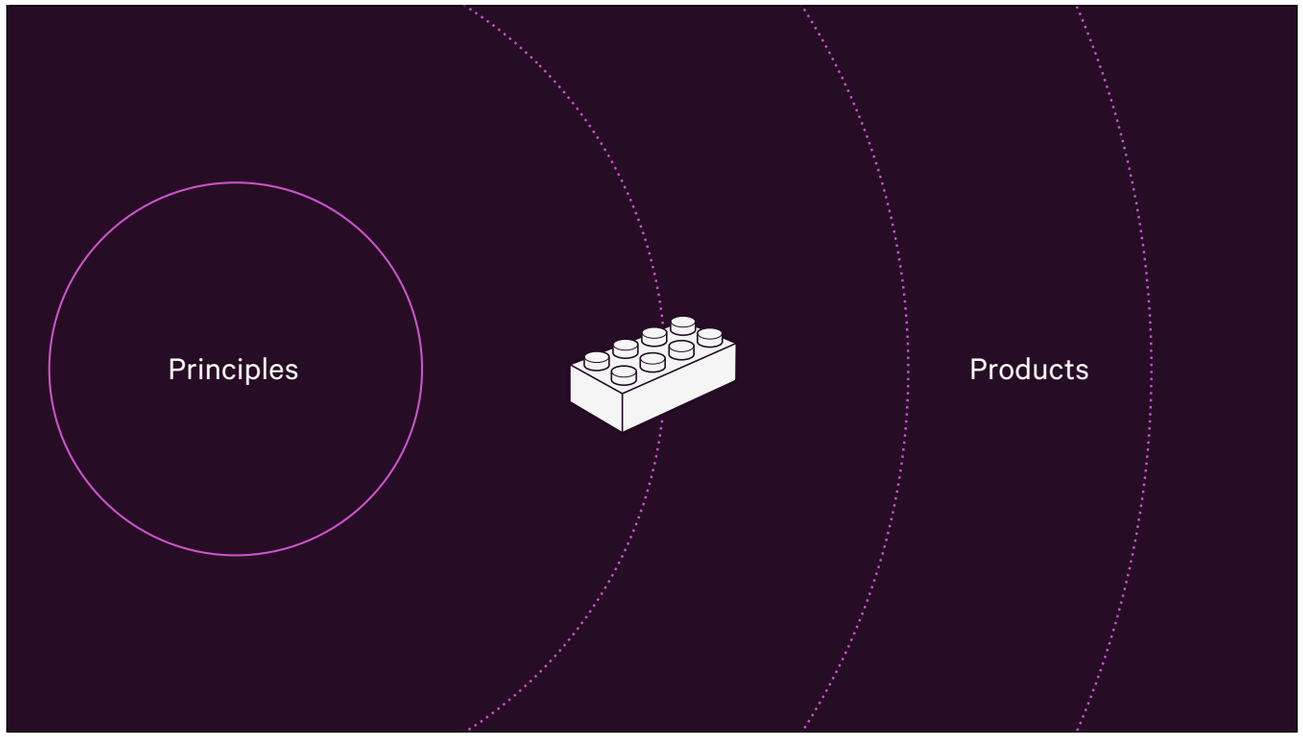


- Component libraries:
  - Sometimes called a pattern library or front-end style guide.
  - Words *pattern* and *style guide* have well defined meanings, so I use term *component library*.
  - Fractal is pretty damn awesome.

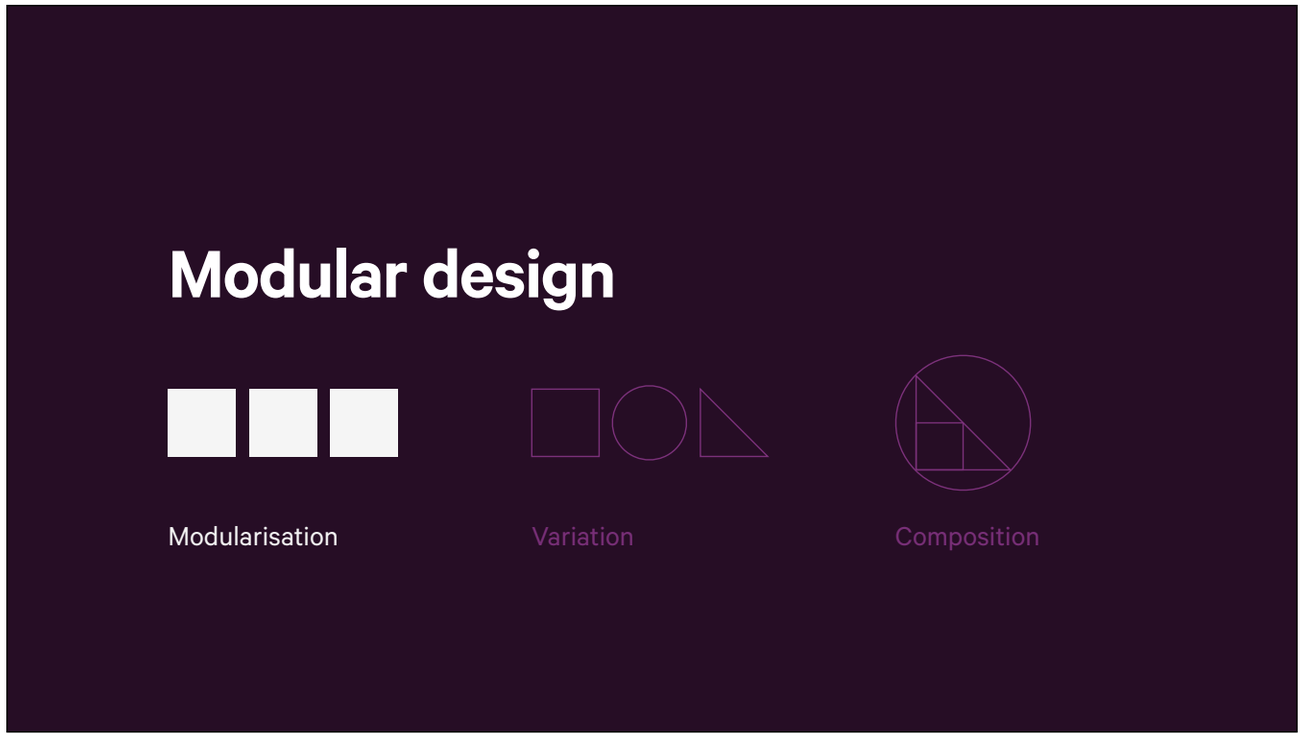


- Finally, we have the resulting implementations, be they websites, apps or other products.
- If we've done this right, **user-needs will be embedded across the entire system**.
- A fully considered design system like this will ensure:
  - Consistent design and development practices, leading to...
  - Predictable product for customers, engendering trust in your organisation.

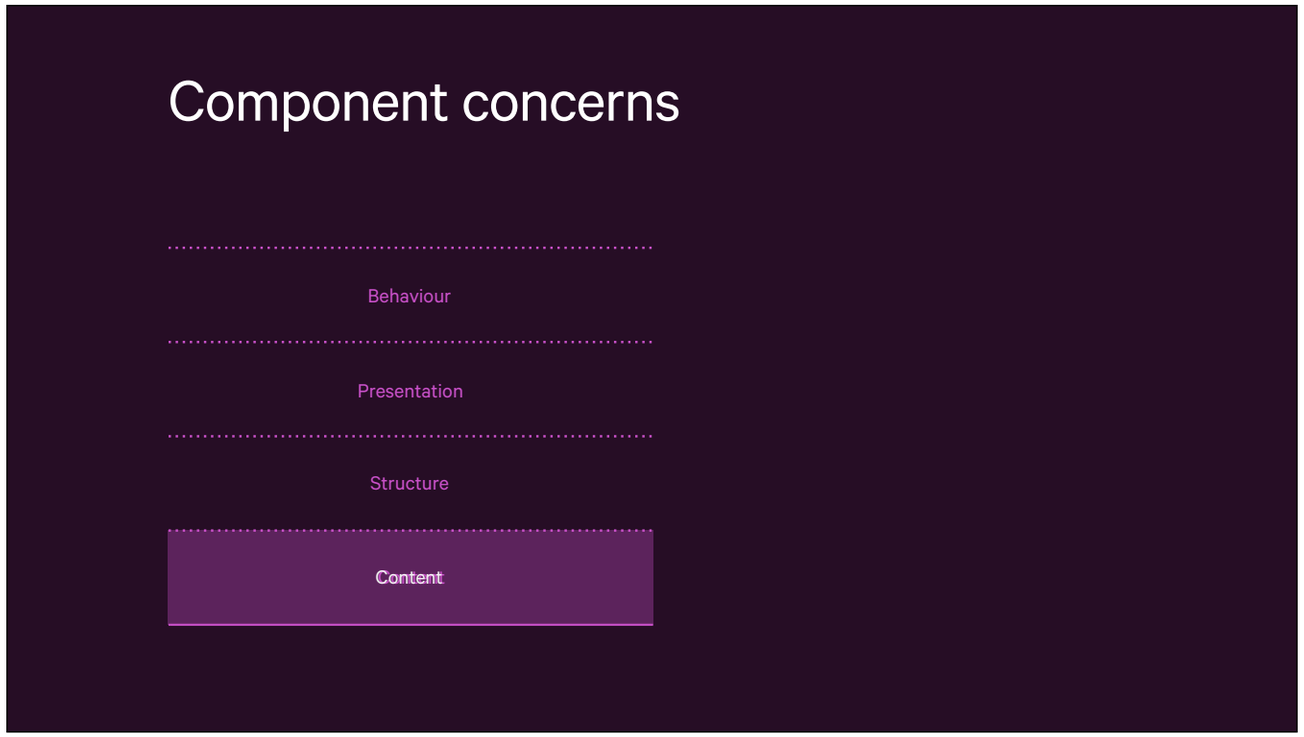
- Worth noting that in this model, the surface layers are more susceptible to change:
  - Principles should be reviewed periodically, but remain largely static.
  - Changes to products can change much more often, perhaps even daily.
  - If you're familiar with the concept of pace-layers, this will be familiar to you.



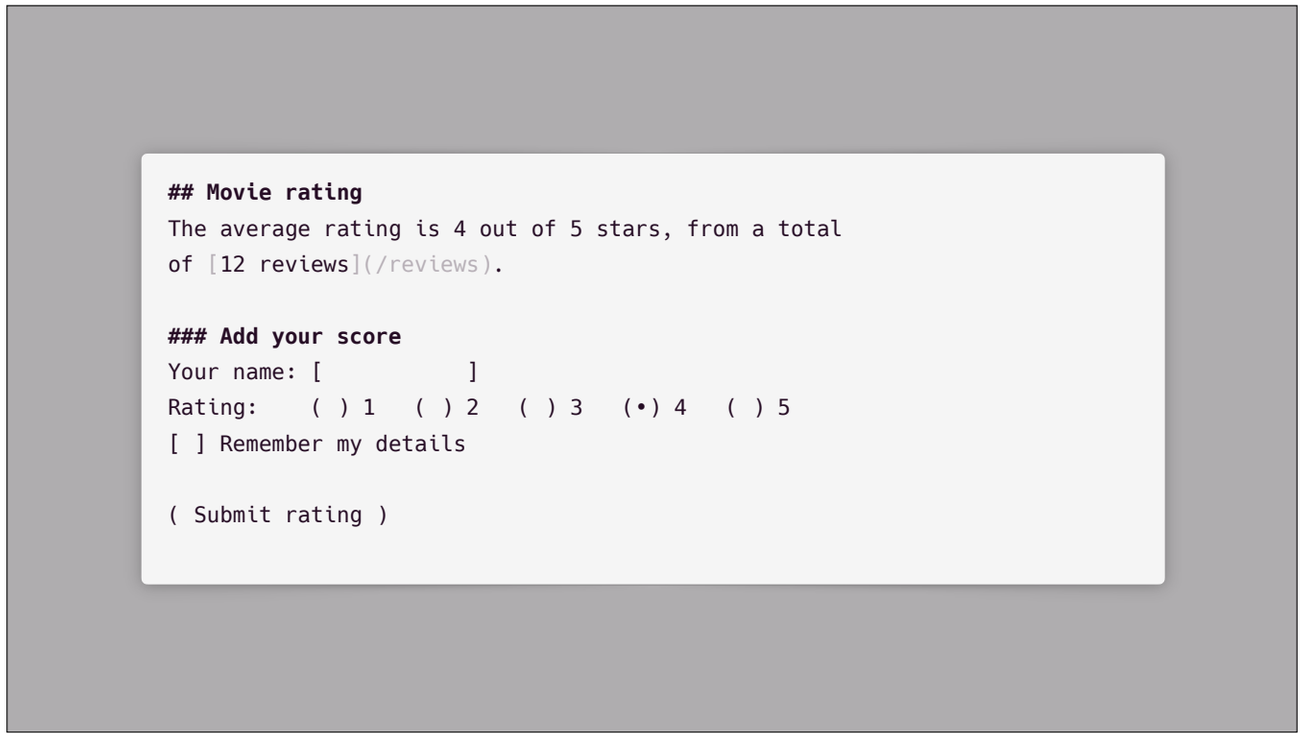
- How do we move from principles to products?
- Here lies complexity: a number of requirements, and any number of outputs.
- By breaking these down, become easier to manage.
- Enter: Modular design



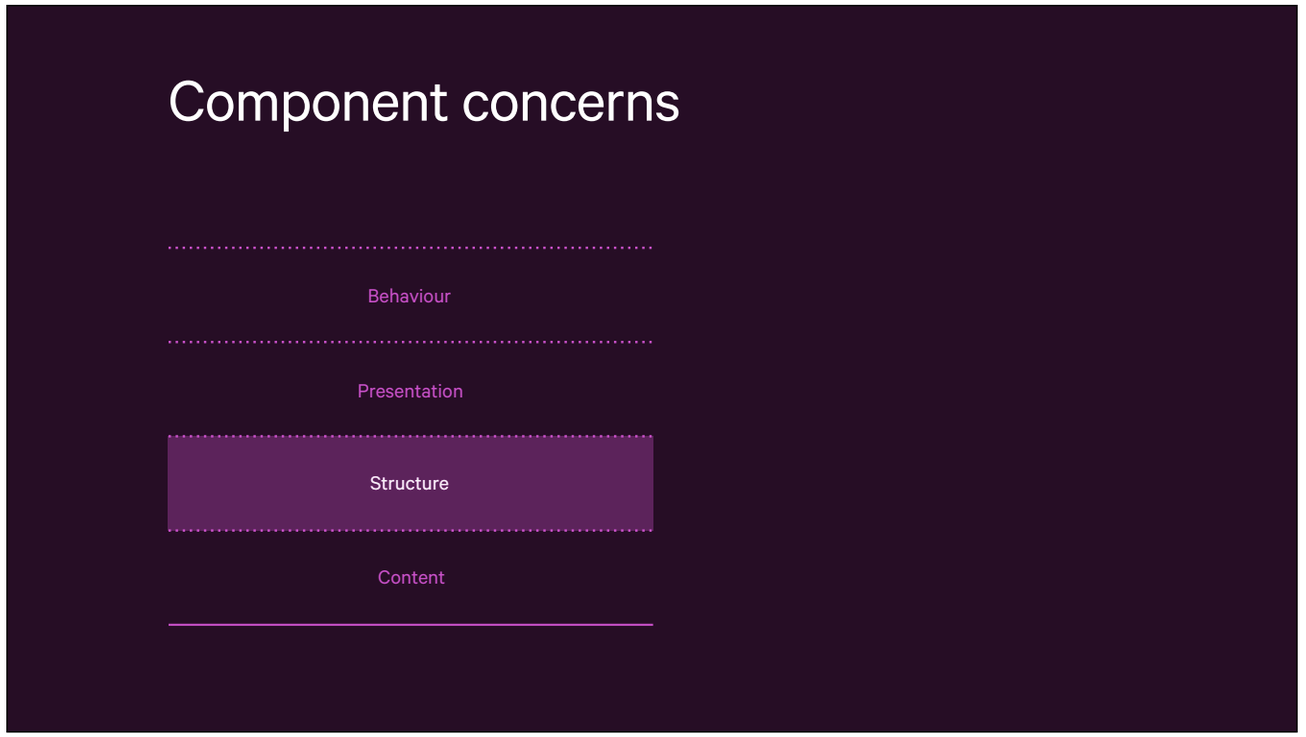
- What follows is a laymen’s description of modular design.
- Thoughts about creating a front-end design system.



- Like a design system, a webpage is comprised of layers
- Each responsible for a separate concern.
- First layer:
    - Reduce any webpage to its barest essentials, and you’re left with text.
    - Even the complex task-based application rely on labels and messaging.



- I like to start building a component by writing plain text.
- Markdown, gives me a limited set of structural markers.
- For forms, buttons and inputs, I can also use an ASCII-art like syntax.
- Constraints keep me focused on the content.



- Next step is to introduce structure and meaning.
- We'll **introduce a number of shared vocabularies** to help us do this.
- One has already been defined for us: HTML.

```
<meter value="4" min="0" max="5">4 out of 5 stars</meter>
```

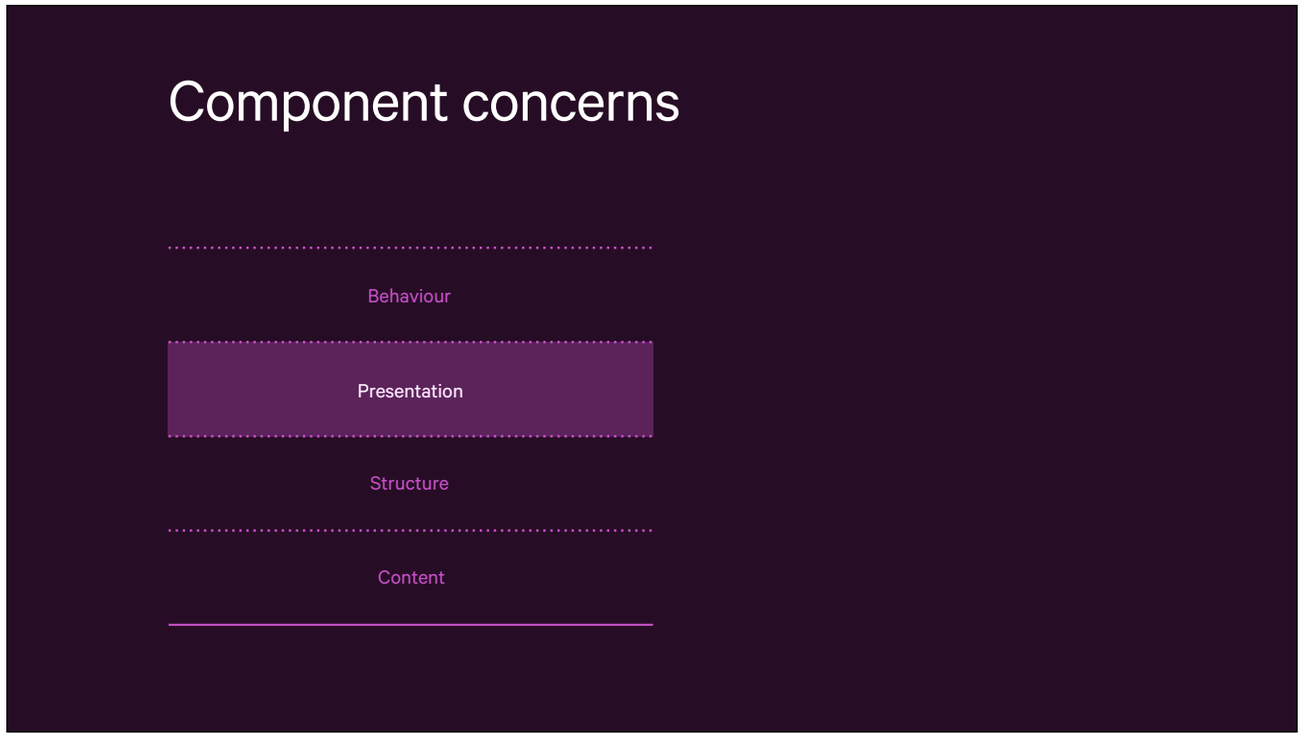
- Using HTML's relatively limited palette of markup again forces us to focus.
- Worth reviewing the options available, especially given the range of elements added in HTML5.
- For example, on a recent project I used the meter element to describe a rating.
- But HTML alone can only get us so far.
- To describe components in relation to internal vocabularies, we use the class attribute.

```html
<section class="c-rating">
    <h1 class="c-rating__title">Movie rating</h1>
    <span class="u-hidden">The average rating is</span>
    <meter class="c-rating__value" value="4" min="0" max="5">
        4 out of 5 stars
    </meter>
    <span class="u-hidden">from a total of</span>
    <a class="c-rating__reviews" href="/reviews/">12 reviews</a>
</section>
```

- With the class attribute, we can express our internal naming conventions.
- Notes:
    - Using BEM naming methodology.
    - Single letter prefixes indicate rule type (as per CSS namespacing, described by Harry Roberts)

```
.c-[module-name] {…}
    .c-[module-name]__body {…}

        .c-[module-name]__header {…}
            .c-[module-name]__title {…}

        .c-[module-name]__main {…}
            .c-[module-name]__items {…}
                .c-[module-name]__item {…}

            .c-[module-name]__content {…}
                .s-prose {…}

        .c-[module-name]__footer {…}
```

- On the topic of naming, I also like to use naming meta patterns
- These are repeated yet unrelated approaches for naming groups of elements.
- Here's a common meta pattern I've been using recently.
- Referred to as 'the workhorse with no name', and 'every module ever scaffold'.
- Repeating **patterns make our code more predictable**, easier to understand.



- Once I reach the presentational layer, it’s here that I start to translate the visual design language into code.

```scss
// Colour palette
$color-brand--crimson: #c00;
$color-brand--mustard: #fc0;

$color-neutral--darkest: #222;
$color-neutral--darker: #444;
$color-neutral--dark: #666;
$color-neutral--mid: #888;
$color-neutral--light: #bbb;
$color-neutral--lighter: #ddd;
$color-neutral--lightest: #eee;
```

- Start by abstracting common design specifications into a set of variables.
- Here using colours as an example, but concepts can apply to other values.
- Saving values as variables keeps our code DRY, but on their own lack meaning.
- Variables give us options, but they don’t give us decisions.

```scss
// Colour assignments
$color-text: $color-neutral--darkest;
$color-background--light: $color-neutral--darkest;
$color-background--dark: $color-neutral—darkest;

$color-link: $color-brand--crimson;
$color-link--hover: darken($color-brand--crimson, 10%);
$color-link--active: lighten($color-brand--crimson, 10%);
```

- Assigning variables to a second set gives them context.
- If we change values later, we can (hopefully) change them in one place.
- But also makes our code easier to reason with.

```scss
// Colour map
$colors: (
  crimson: #c00,
  mustard: #fc0,
  neutral: (
    darkest: #222;
    darker: #444;
    dark: #666;
    base: #888;
    light: #bbb;
    lighter: #ddd;
    lightest: #eee;
  )
);
```

```scss
// Helper
@function color($name, $var: null) {
  @if ($var != null) {
    @return map-get(map-get($colors, $name), $var);
  } @else {
    @return map-get($colors, $name);
  }
}

// Usage example
.c-module__title {
  color: color(neutral, darkest);
}
```

- Sometimes I like give my variables an implied structure, too.
- This helps me better describe the relationship between different values.
- In Sass, I can create a JSON-like data structure…
- Then, by creating a small Sass function…
- I can reference my colour palette like this.

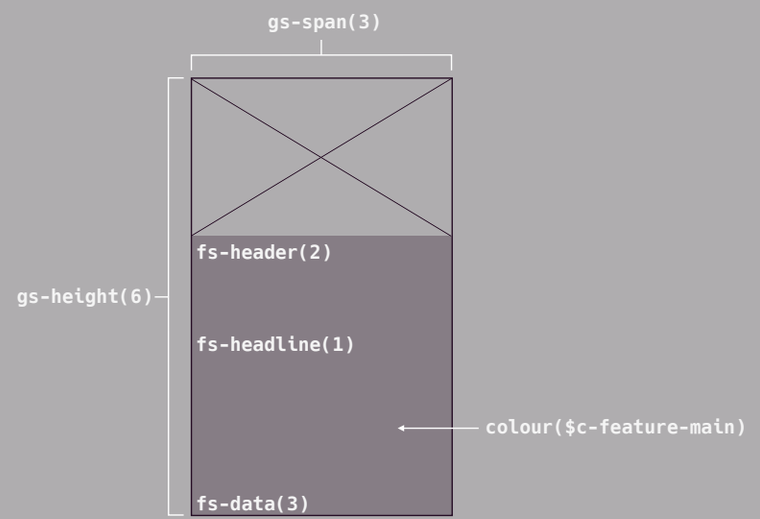
```scss
.c-module__title {
  font: bold 2em/1 Georgia, serif;
  margin-bottom: 1.5em;
  padding-top: 0.75em;
  letter-spacing: 0.0025em;
  text-transform: uppercase;
  color: #222;
}
```

```scss
.c-module__title {
  @include typeset(title, 2);
  margin-bottom: ($baseline * 4);
  padding-top: ($baseline * 2);
  color: $color-text;
}
```

- I’m using Sass, but it’s the concepts I want you to take away from this talk.
- Idea is to reach a point where we are **no longer writing intangible rules**…
- …but **a design specification**, from which we can infer intent.

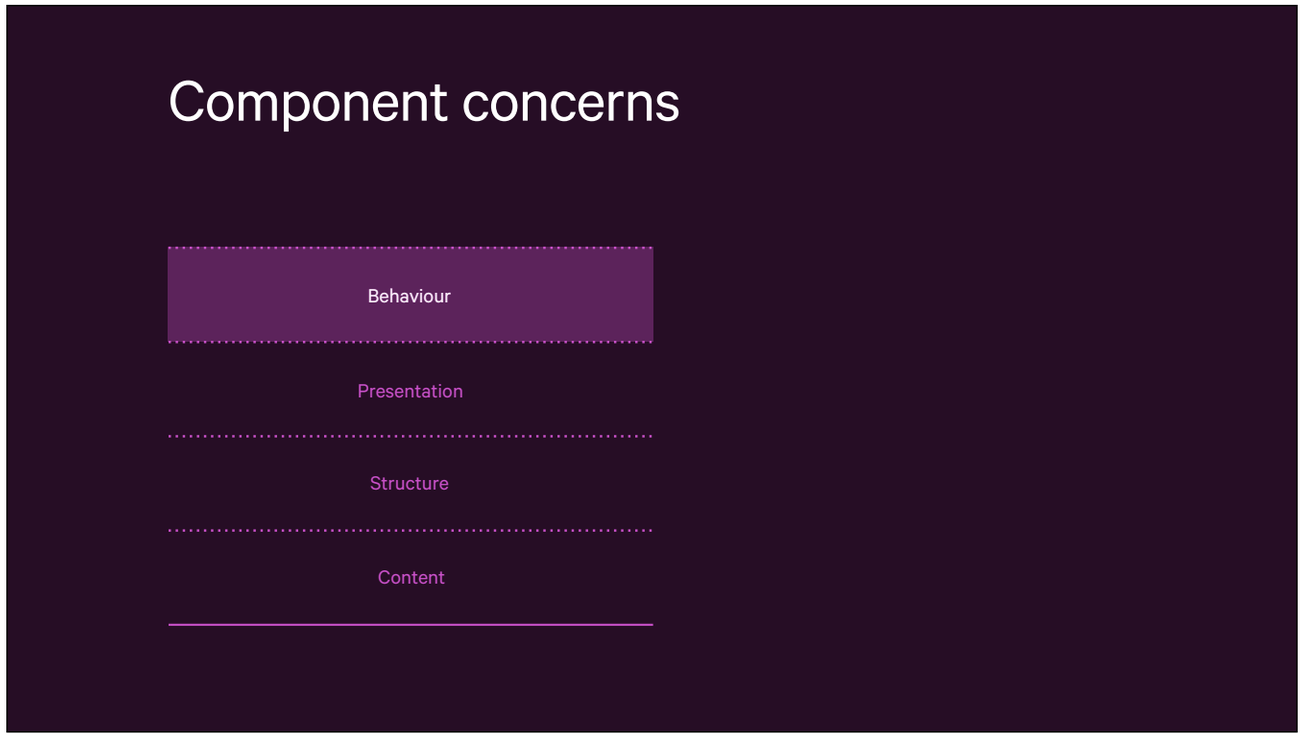
Guardian Style Sheets (GUSS)

github.com/guardian/guss

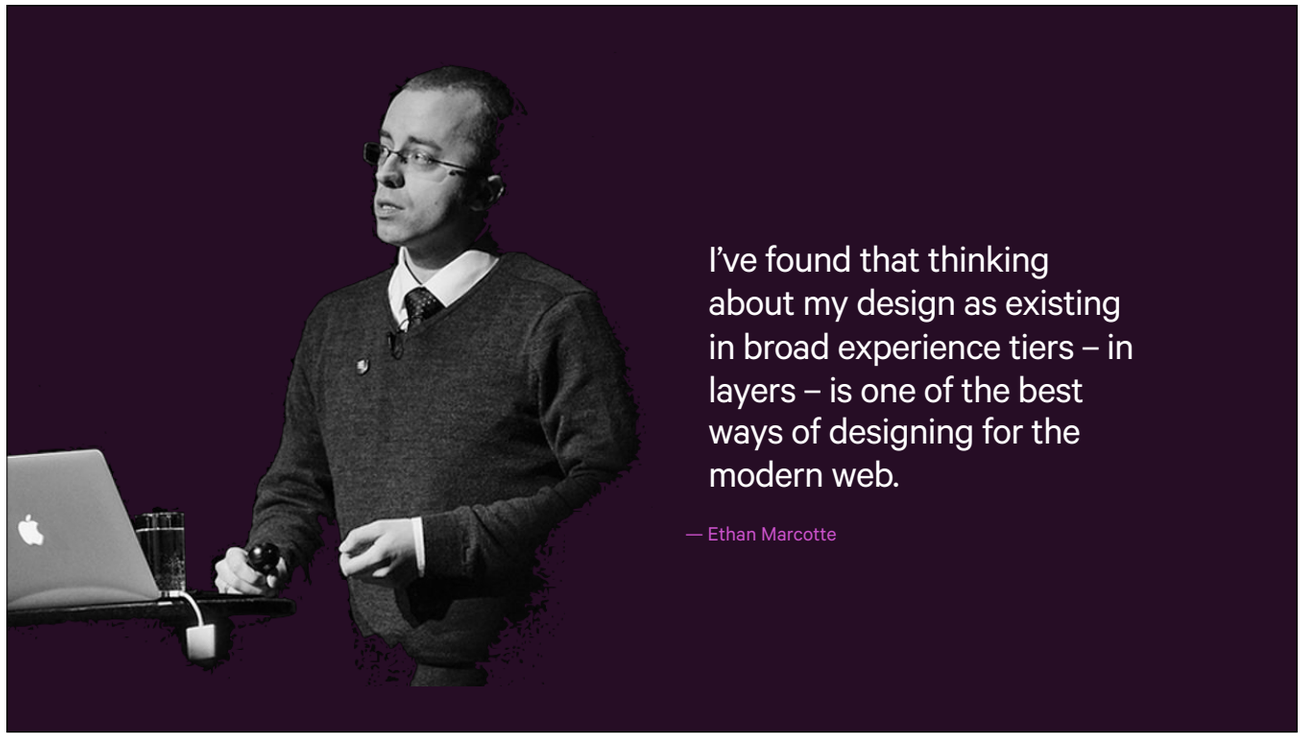


- This is similar to the system used at the Guardian.
- GUSS is a set of Sass components. These use internal names for different elements of our brand.
- Means components could be described using a **language familiar to everyone** on the team.

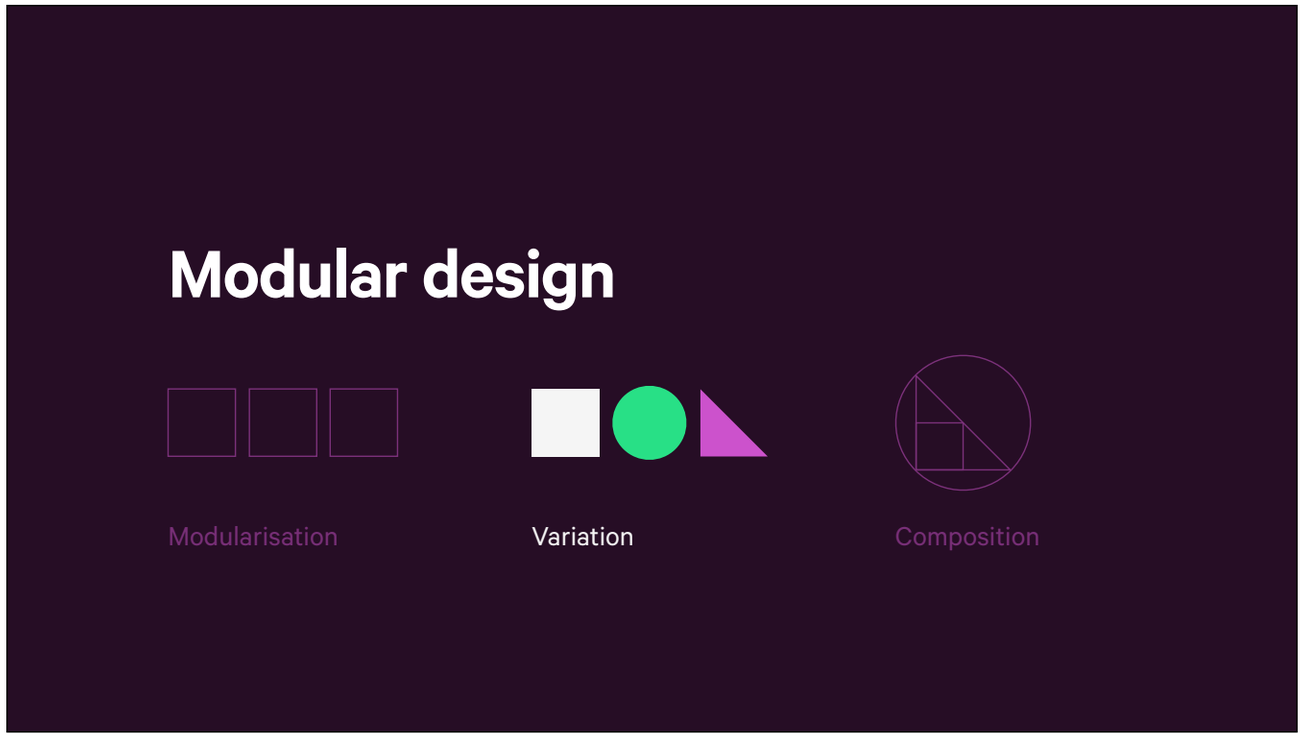
Component concerns



- As we reach the top of the stack, we can start to look at how our components behave.
- JavaScript it not my forte, but even here, we can introduce our naming conventions.
- Similar concepts of starting simple and building up can apply to: pseudo code.
- Should pay attention to our design principles, and who we are designing for.



- This means of creating components is in tune with the thoughts of Ethan Marcotte.
- While it’s tempting to describe a component as series of separate layers, certain aspects will transcend them.
- e.g. accessibility of a component, may influence:
    - choice of language,
    - legibility of text, colour contrast,
    - affordances provided for different states.
- Internationalisation and compatibility are other concerns that transcend layers.



- Beyond the components themselves, the next thing to consider is how we might vary them.

Extensibility

1. Create a new component

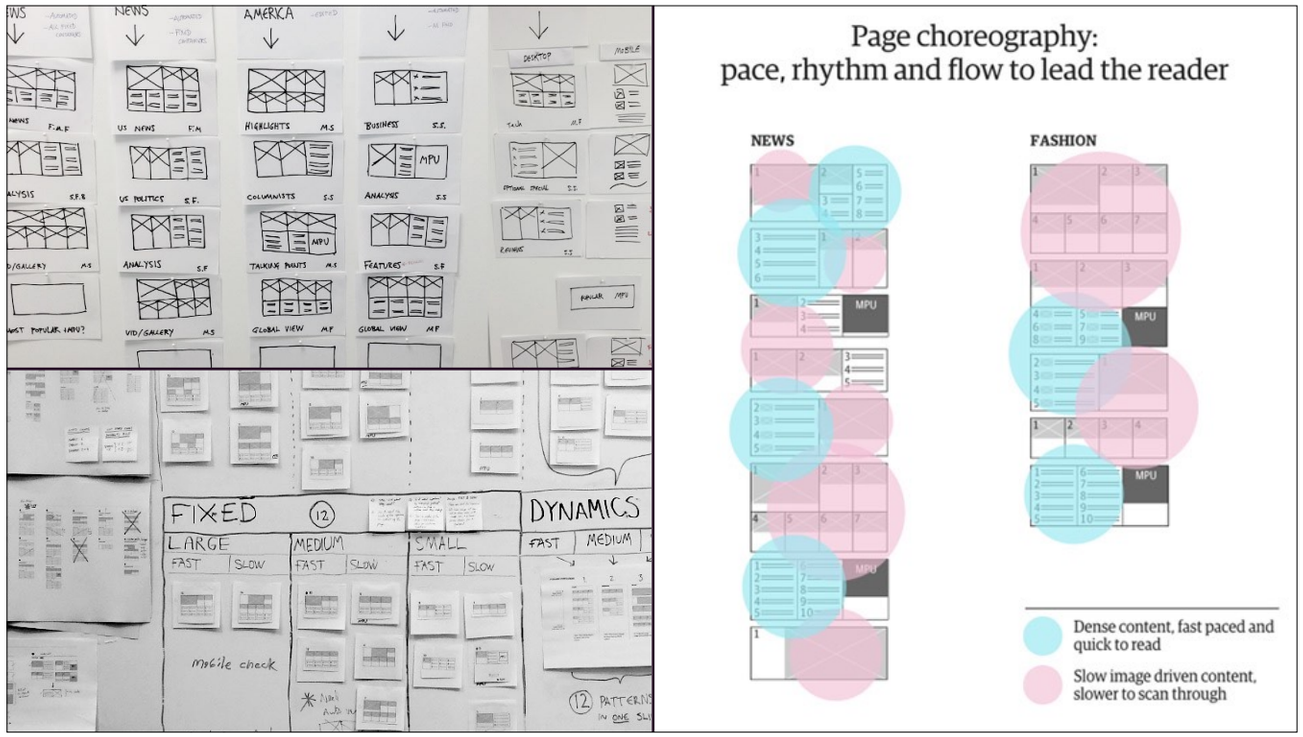
2. Modify an existing component

- This speaks to the benefit of a systematic approach to design: extensibility.
- There are essentially two options for how you can extend a system:
    1. Create a new component
    2. Modify an existing component
- A new component may add redundancy if it closely matches an existing one.
- A component with too many modifications will be difficult to manage, maintain and understand.
- This section will be less about the mechanics, more about understanding.

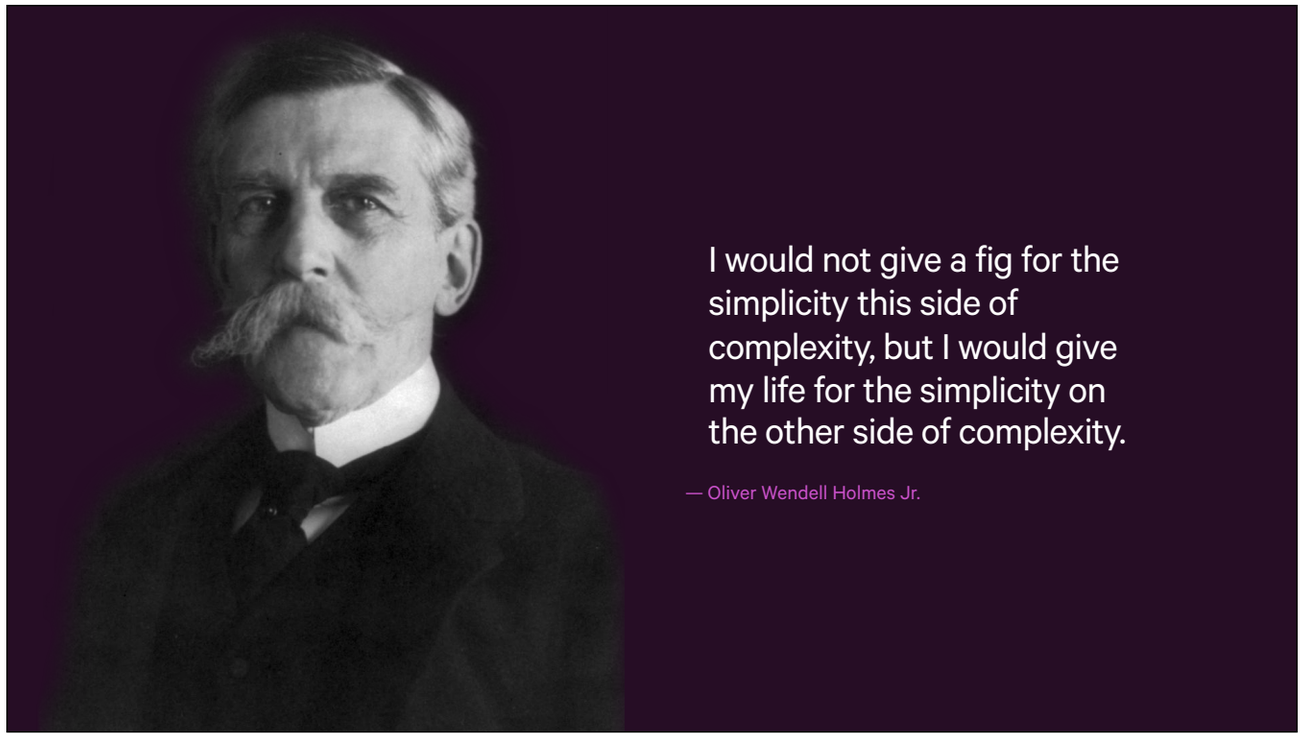
The container model

next.theguardian.com/blog/container-model-blended-content

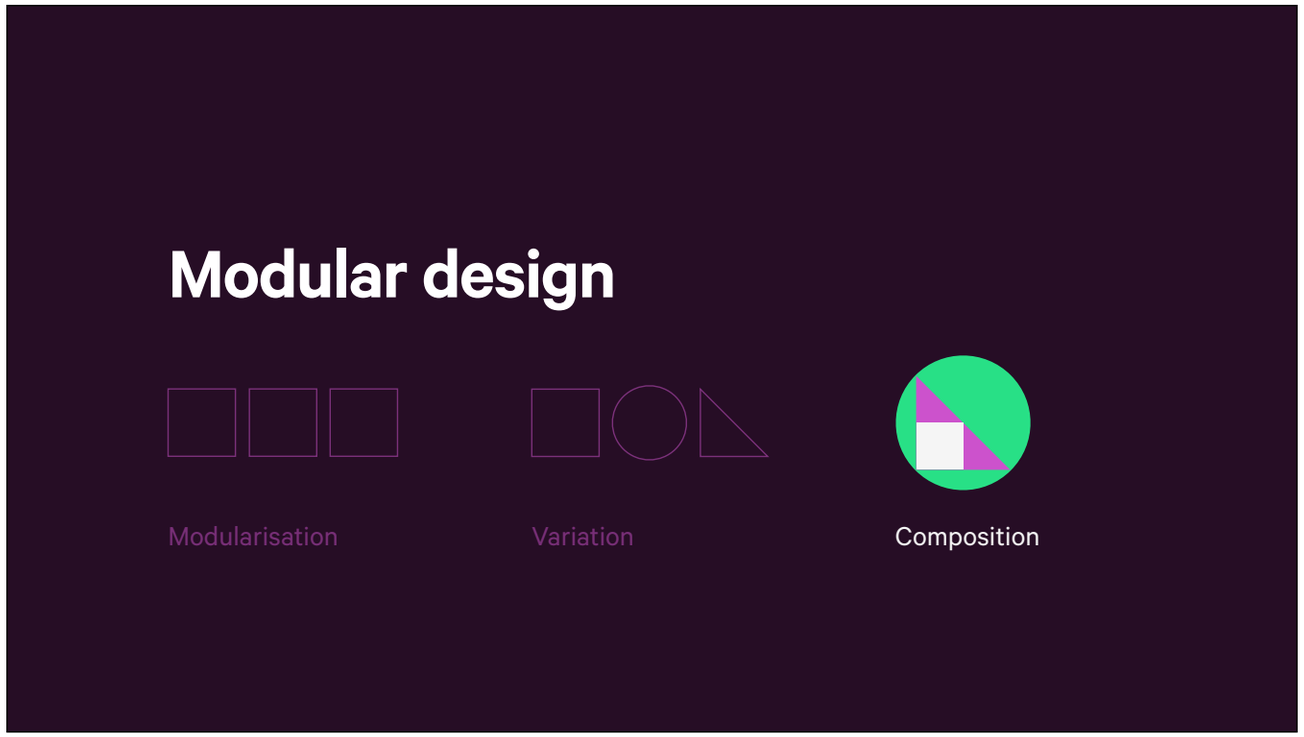
- The Guardian website uses something called the container model.
- Reduces complexity, brings a systematic approach to page composition.
- How it works:
    - A page is split up into different horizontal containers.
    - Each container contains multiple smaller horizontal slices
    - Inside these slices exist a number of items, which can take a number of different formats.
- Now, in retrospect this seems pretty straight forward. And it took only 6 weeks to design an build.
- But...



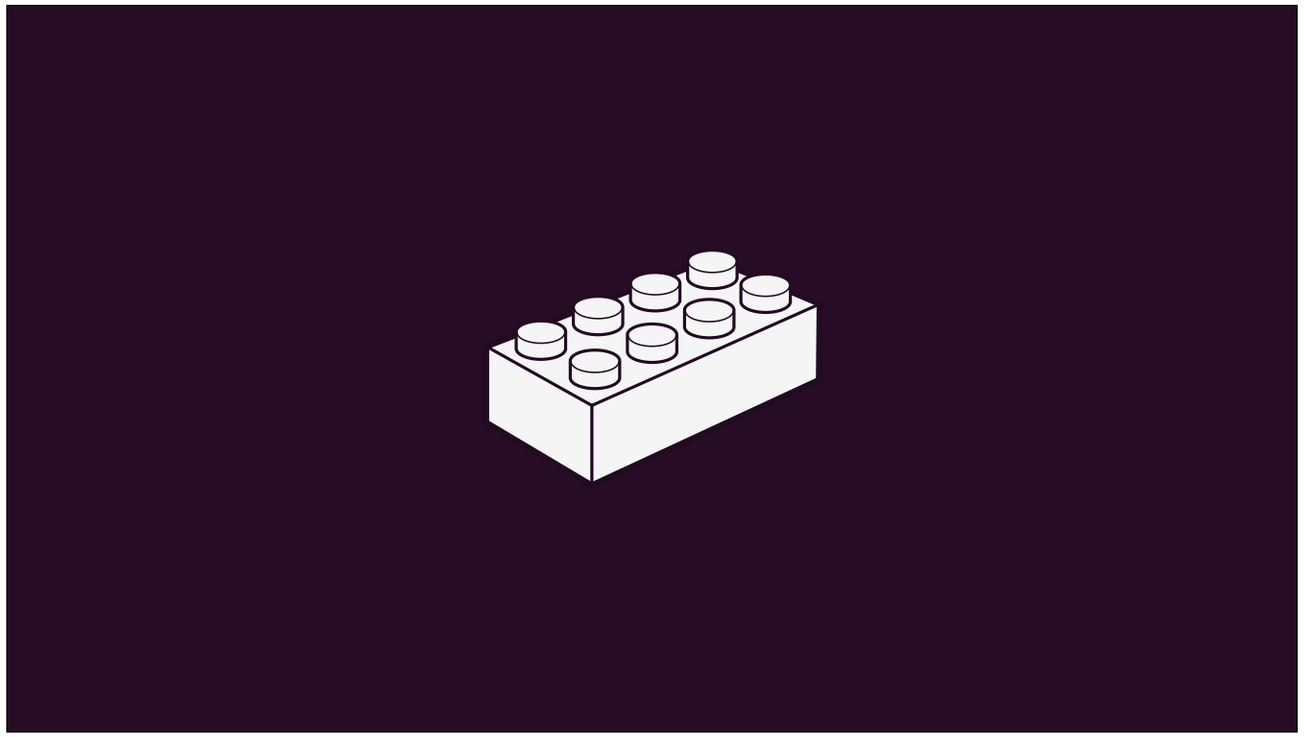
- Getting to the point where we actually understood the system we needed to build took much, much longer: over a year!
- I didn't work on this much, but my colleagues were tearing their hair out!
- Why? Needed to consider the following variations:
    - The type of story could affect it's **presentation**.
    - The newsworthiness of a story could affect its **position**.
    - Curated or dynamically collated items could affect **layout**.
    - The overall pacing of a page could affect the **number** of stories.
    - ...and any solution needing to work across a **range of viewports**.



- A friend told me of this quote the other day: I think it sums up designing a system perfectly.
- Easy think you can simply build a set of components, and the pages will follow.
- I’ve found the opposite to be true.
- Needing to **constantly zoom up to pages and back down to components** to ensure the system is workable and appropriate.



- Which brings us neatly on to the final consideration within modular design: composition.



- No discussion about modular design is complete with mention of Lego.
- This is the real world equivalent of object-orientated programming
- Separate blocks that can be pieced together in any number of combinations.
- But look closely, and you'll find shortcomings in this comparison.

The wrong analogy?

- Composite
- Static
- Encapsulated
- Clear affordances

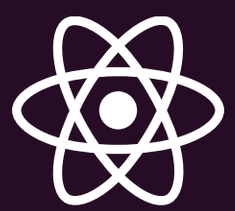
- Separate concerns
- Dynamic
- Leaky
- Difficult to reason with

- A Lego brick is…
  - a **composite**, made of a single material, ABS plastic
  - **static**, its state doesn't change under normal operating conditions
  - **encapsulated**; its colour/shape won't alter depending on the bricks attached to it
  - has clear **affordances**, you can understand its properties by looking at it.
- A front-end component is…
  - composed of **different concerns**, (as previously discussed)
  - **dynamic**, maybe intentionally through interaction, or affected by external factors
  - **leaky**: the very first initial of CSS describes the opposite approach, rules able to *cascade*
  - **difficult to reason with**, especially if code is in separate files, with little/no documentation.

Bringing encapsulation to the web

BEM

en.bem.info



React

facebook.github.io/react

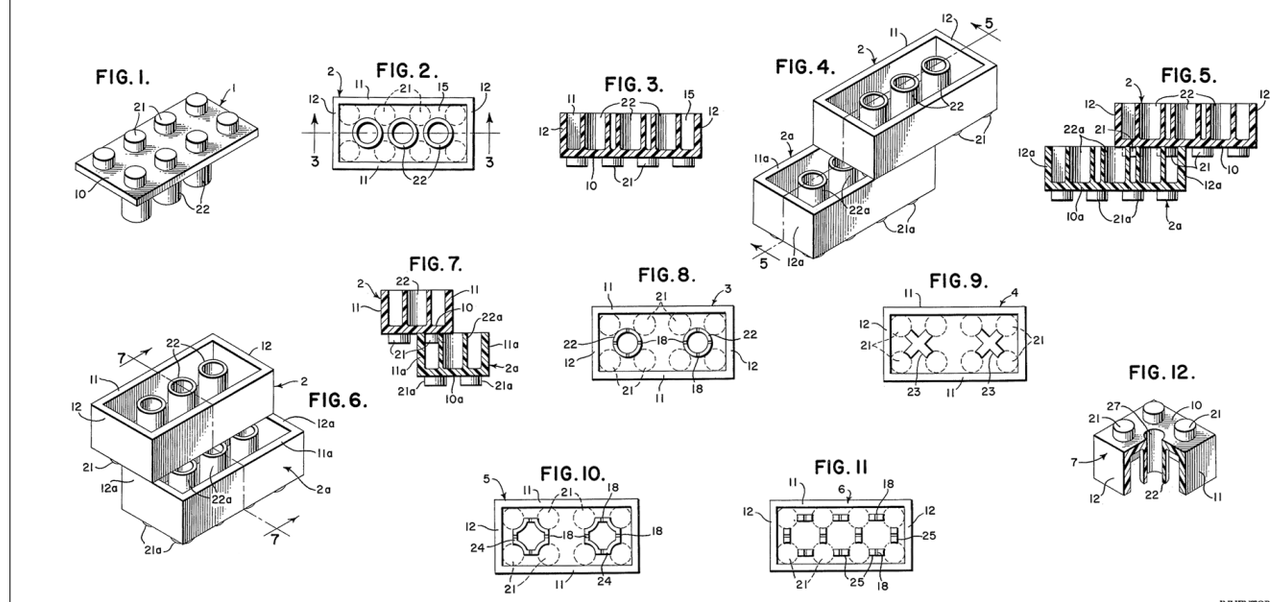
Web Components

webcomponents.org

- In the quest to build modular systems with greater predictability, libraries and **tools seek to address these differences**.
- Popular approaches to CSS seek to undermine the cascade, be that by name-spacing rules or applying styles directly.
- React components combine HTML and JavaScript into a single JSX file
- These approaches are entirely understandable, and not necessarily wrong.
- ...but, it feels like we're constantly **working against the grain** of the of the web.

> Always design a thing by considering it in its next larger context – a chair in a room, a room in a house, a house in an environment, an environment in a city plan.
>
> — Eliel Saarinen

- We look to the hardware for inspiration, yet we’re building software.
- Without connections, a system is merely a sum of its parts.
- It’s not so much about the components, but how we connect them, how they relate to each other.
- In looking at Lego, have we been concentrating on the wrong thing?



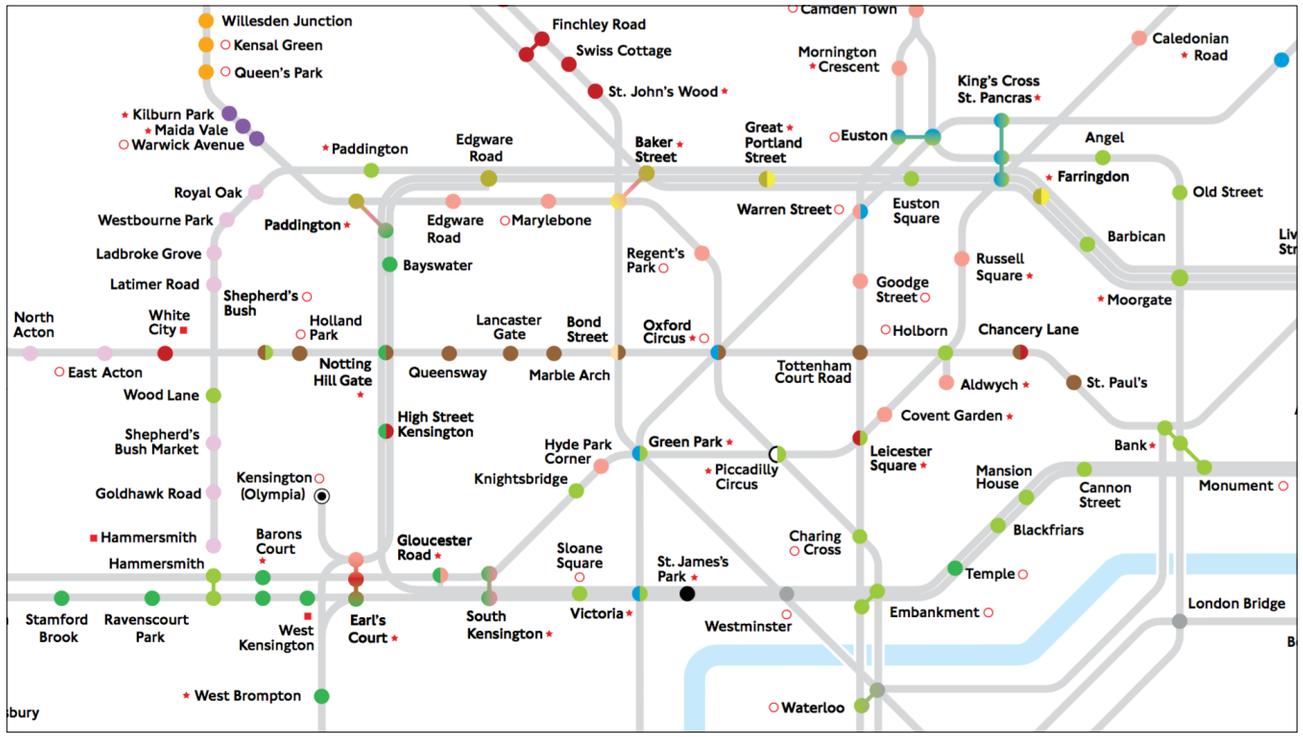
- The most important aspect of Lego is not the bricks, but the system of tubes and stubs that holds them to together.
- New brick shapes have been added to the system over the years, yet a brick manufactured today will connect with one of the first built in 1958.
- What are the equivalent concepts within the realm of front-end development?
- **I don’t have the answer, but its an intriguing question.**
- By focusing on the relationships rather than the components themselves, perhaps the cascade could become a powerful ally, rather than pernicious foe?

No system is or should be perfect

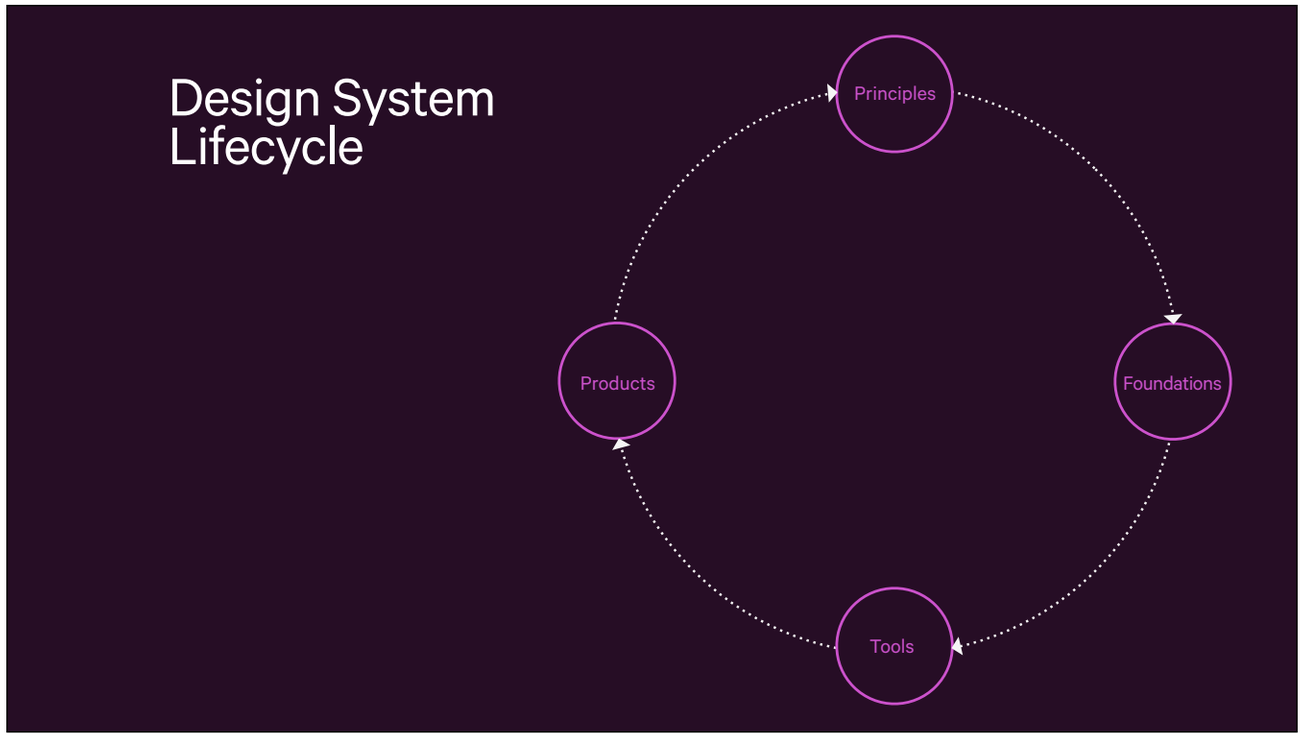
- So, once we've created our design system, we're done, right?
- No system is ever truly finished.
- It's constantly subject to change, to reassessment and adaption.
- Nor can it be self-sustaining. It needs constant stewardship and maintenance.

Photograph: © Kirk Bauer

- While today London’s public transport is held us an example of a well-designed, best-in-class public service, this wasn’t always the case.
- In the three decades following Pick’s departure, successive managers prioritised price over quality.
- This is a photo of the underground taken in 1997 – very different from how it looks today.
- Pick’s ethos of fitness for use was disregarded.
- Only over the last 15 years have London’s politicians given the network the attention it deserves.
- The roundel is now a mark of quality, only applied to services that meet a certain standard.



- And **new tools have been created to serve this aim**.
- Forming part of the ‘Station Design Idiom’, it gives contractors a station-by-station guide to the different architectural styles used on throughout the underground network.
- When new features or modifications are made, they must suit the context of the existing station.



- Going back to our earlier model, perhaps we should look on it less as a series of layers, but more of a feedback loop.

- Our principles inform the foundational choices we make, the tools we build to create products.
- But their suitability can only be judged upon use.
- Success or failure requires us to reassess our assumptions, and course correct where necessary.

- Pick was more successful in his attempt to remodel London because he had to work with what was already there, which could inform his work.
- Yet Lúcio Costa talked of Brasilìa coming to him in a dream, his ego unbounded.
- Our context is purpose, and context is a useful constraint.

Thank-you!

@paulrobertlloyd / paulrobertlloyd.com