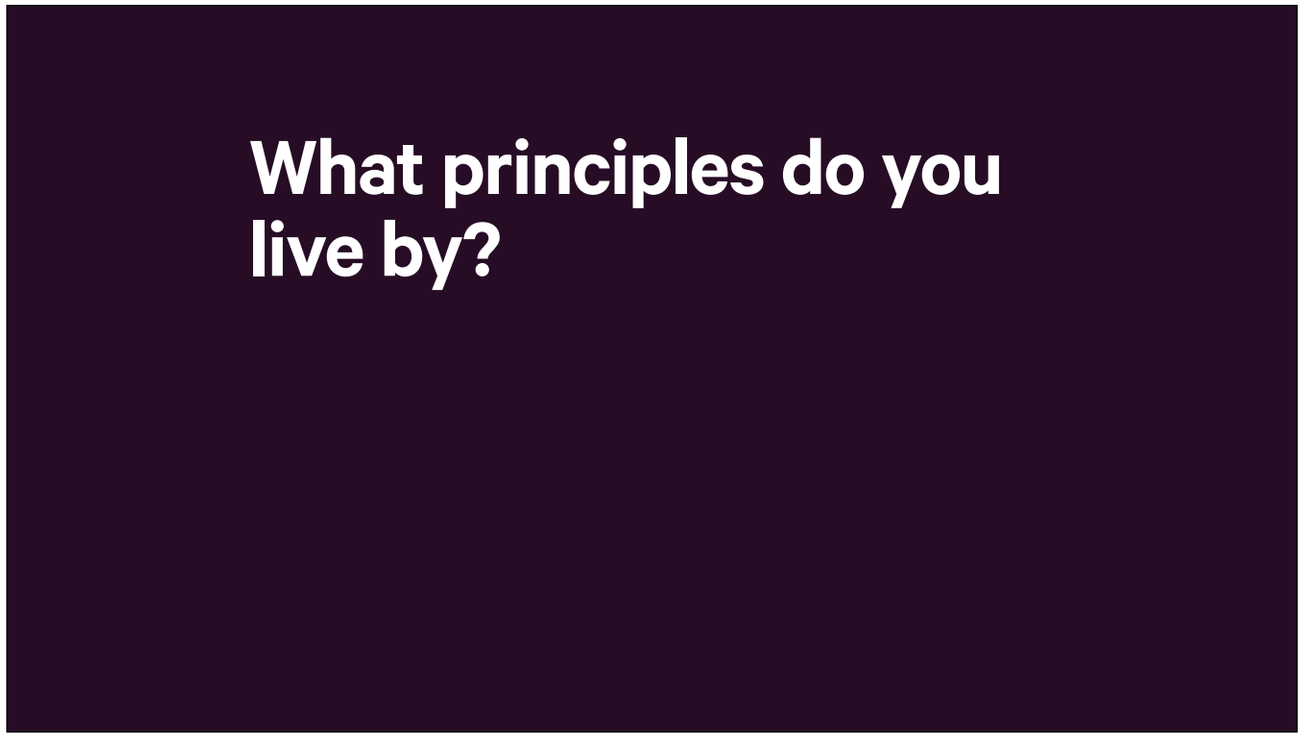
Responsive Principles

A Framework for Future Learning

@paulrobertlloyd

WebExpo / 19 September 2015



I'll start with a question:

> *What principles do you live by?*

You don't need to answer now... but maybe a this is a good conversation starter, perhaps during the breaks or at the after parties.

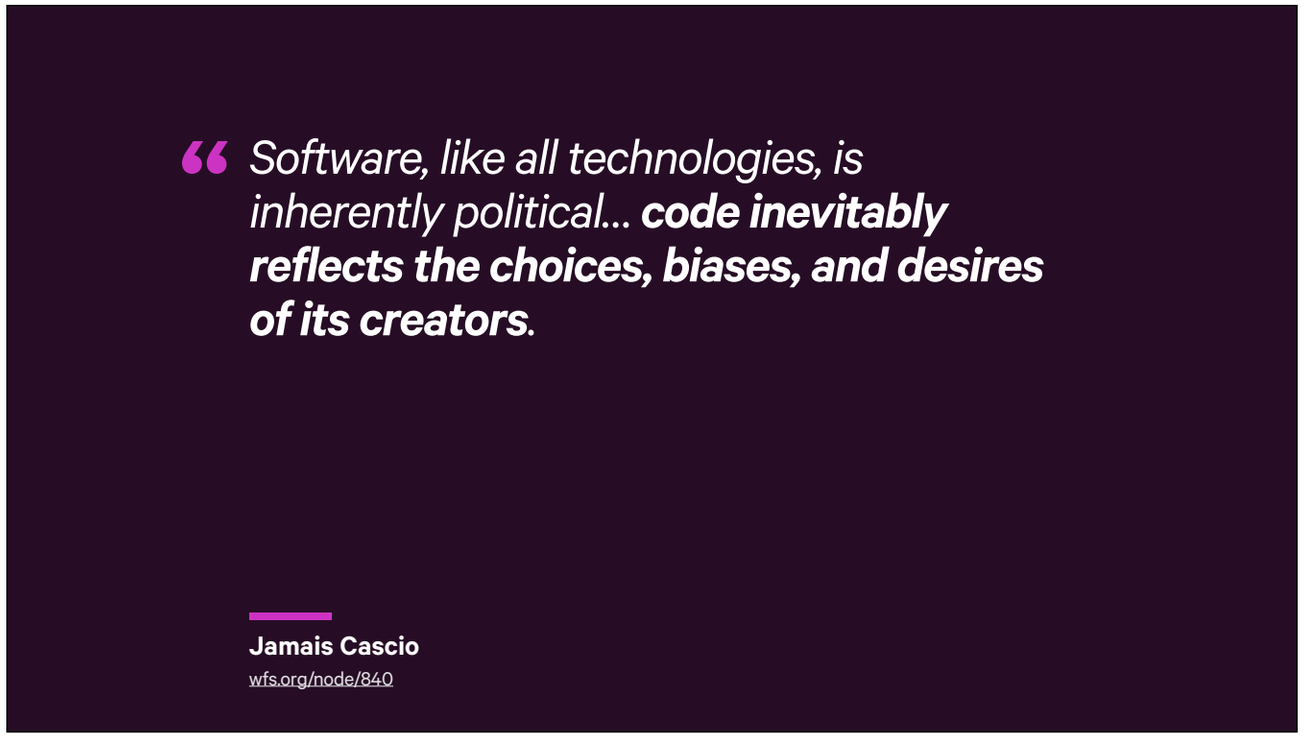
Living sustainably

I'll share one of mine...

By recognising the earth's dwindle natural resources, I seek opportunities where I can choose a more sustainable option.

It's for that reason why I **travelled to Prague from my home in Brighton** on the train; this is one of the four I travelled on to get here on Thursday.

Principles can **help us make choices**, and dictate our behaviour.



The same is true of software.

The futurist Jamais Cascio wrote that:

> *Software, like all technologies, is inherently political... code inevitably reflects the choices, biases, and desires of its creators.*

> *Technology is neither good nor bad;*
> ***nor is it neutral.***

Kranzberg's First Law of Technology
en.wikipedia.org/wiki/Melvin_Kranzberg

Kranzberg's First Law of Technology, put it another way:

> *Technology is neither good nor bad, nor is it neutral*

How we use the technologies we have at our disposal — as well as the new ones that will be invented — is **a product of our own belief**s, and have consequences.

The choices we make matter.

What better example of this, than the **World Wide Web**.

Tim Berners-Lee's creation is imbued with his own principles. He envisioned a web that enabled open access to information and services *and* also enabled anyone to publish content unimpeded, too.

Perhaps Tim expressed this best, when he spelt out his desire for the web at the start of 2012 Olympic Games in London:

> *"This is for everyone"*

Fluid grids
+ Flexible images
+ Media queries
= **Responsive web design**

Responsive web design brings us closer to his vision

By building device agnostic layouts, we're removing another potential barrier that could prevent easy access.

On the surface, approach is the combination of three simple front-end techniques.

Approach is greater than the sum of its parts.

Not since the adoption of web standards has our industry undergone such a **radical realignment of thought and practice**.

The next five years

In the **five years since** Ethan Marcotte described this approach, thousands of websites have launched with responsive layouts at their core.

We've experimented with **new ways of working**, and **refined our design and development practice** so that it's more suited to a fluid, messy medium.

So what do the next five years look like?

As we emerge from this period of enlightenment, we should take a moment to **consolidate our learning** and **consider how we build upon it**.

Last year, Ethan spoke many times about the need for a new framework.

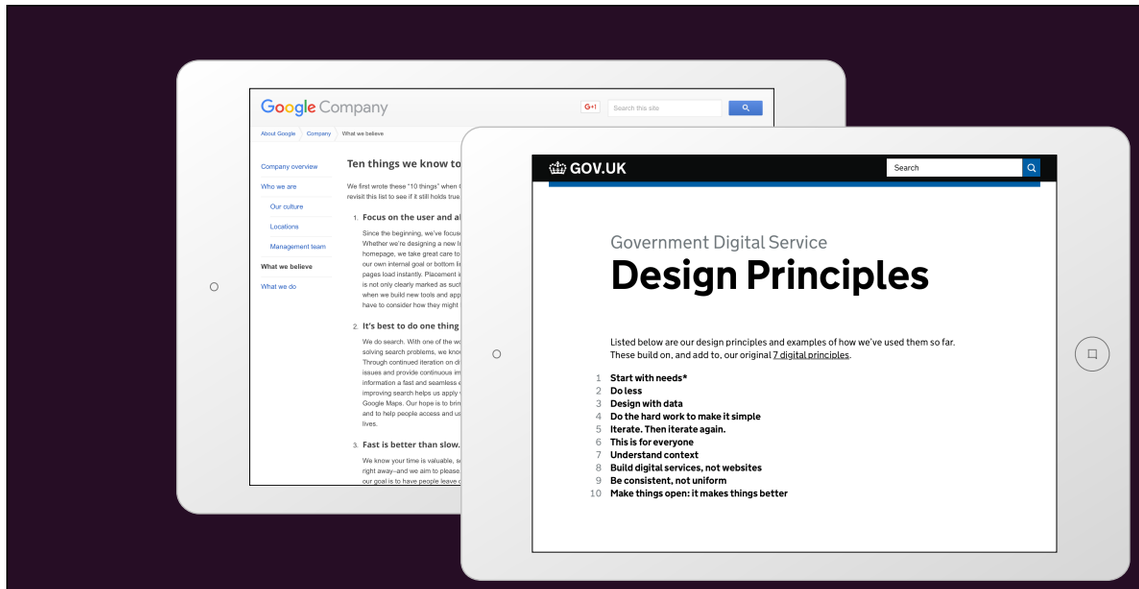
When we think of frameworks, **we often imagine software libraries and other abstractions concerned with execution and code**.

Bootstrap? Frameworks can put distance between us and the realities of the medium. Ethan spoke about a different framework, as he put it:

> *A framework not about execution, but about philosophy and quality.*

Such a framework could help us **model our discussion**, and **measure the quality and appropriateness** of our work.

How might we start building this framework?



Design Principles

google.com/about/company/philosophy/ gov.uk/design-principles

I believe we can conceive this framework by first agreeing on a set of underlying design principles.

You may be familiar with the concept. Organisations like Google and the British Government use them to define the characteristics of their products and organisations.

> *Design principles are short, insightful phrases that act as guiding lights and support the development of great product experiences.*
>
> *Design principles enable you to be true to your users and true to your* ***strategy over the long term****.*
>
> **Kate Rutter: Making Design Principles Stick**
> adaptivepath.org/ideas/essays/archives/001123.php

Kate Rutter describes design principles as:

> *...short, insightful phrases that act as guiding lights and support the development of great product experiences. Design principles enable you to be true to your users and true to your* ***strategy over the long term****.*

What is that long term strategy?

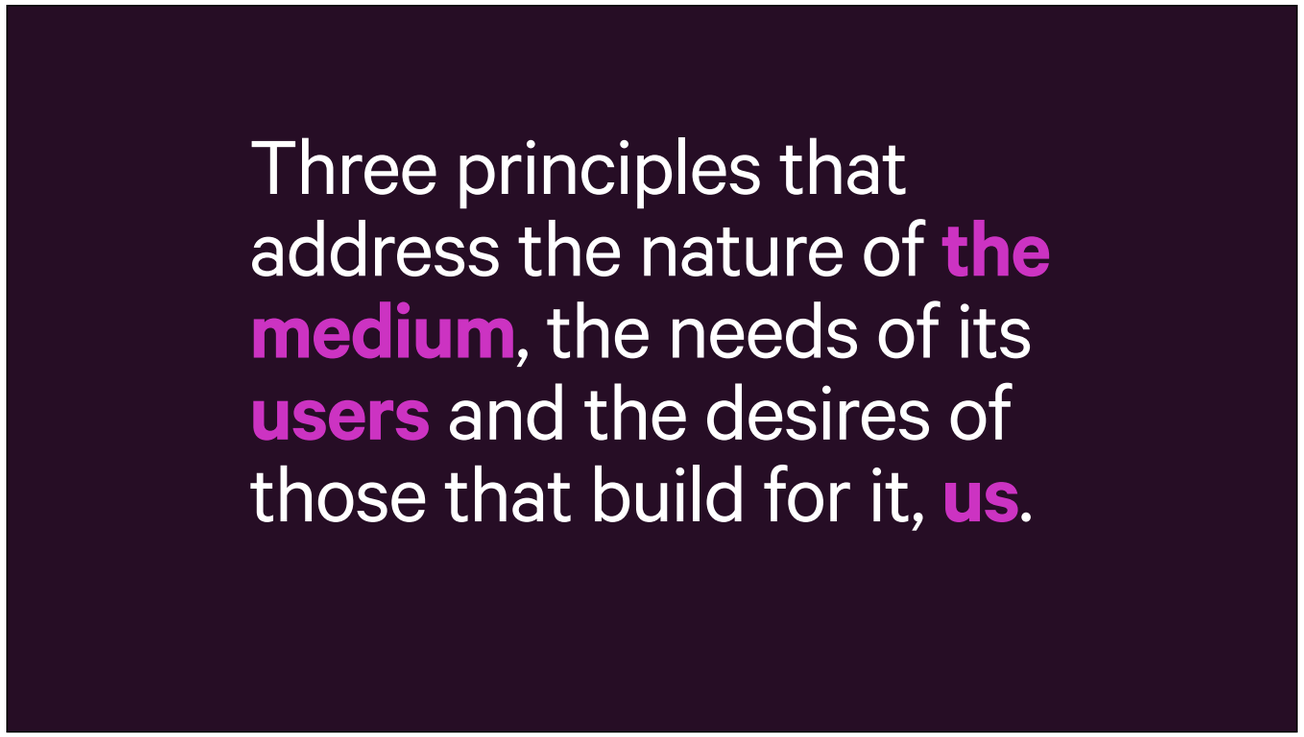
Universal access: this is for everyone.

Tim Berners-Lee described it in an article for Scientific American in 2010:

> *The primary design principle underlying the Web’s usefulness and growth is universality... it should be accessible from any kind of hardware that can connect to the Internet: stationary or mobile, small screen or large.*

Universality is **not only a noble goal**, but **fundamental to ensuring the web’s continued relevance**.

Native platforms can’t achieve the same level of reach. We should **embrace this aspect of it, whole heartedly**. Achieving it is not easy, but **having a strong set of design principles** in service of this vision, will help us.



I believe we can **answer this goal of universal access** by addressing three primary concerns:

- the medium
- those who use it,
- and those that build for it, ourselves.

So, let’s begin.

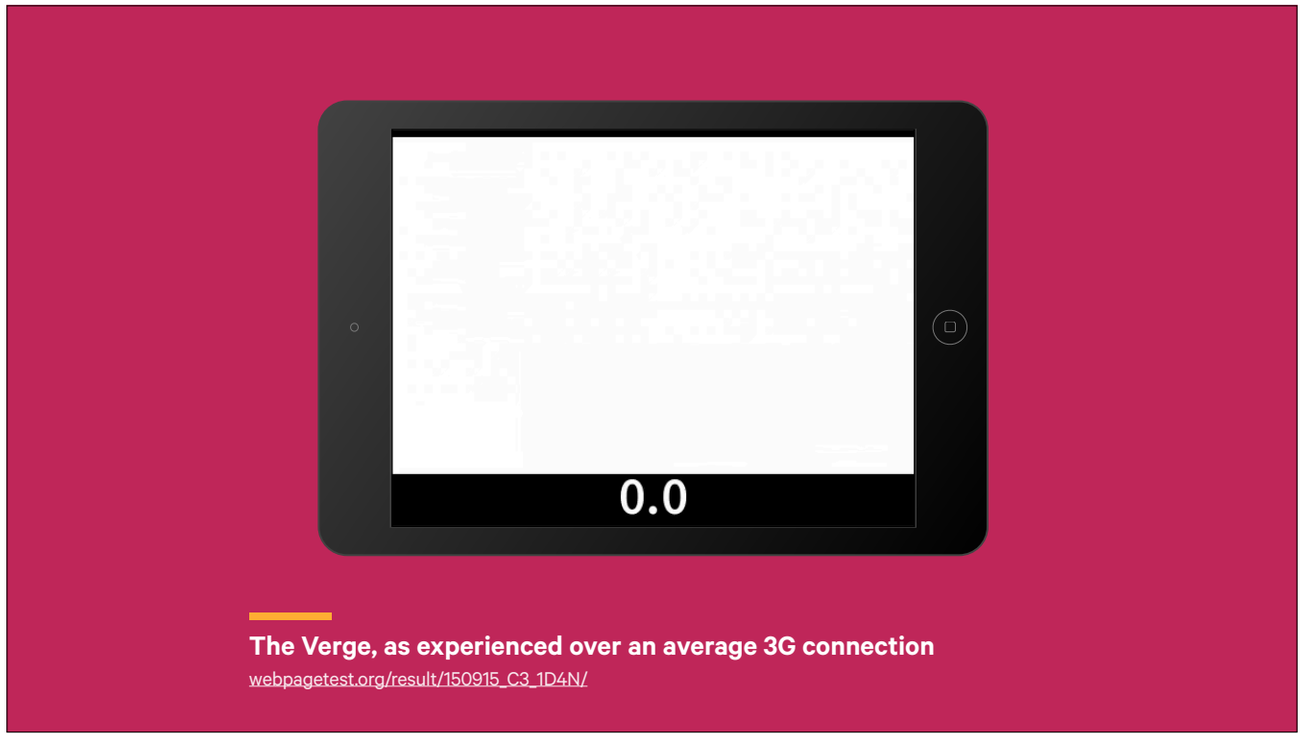
Users

The web is accessed by a broad spectrum of individuals, expressed by the devices they use.

Our first concern, appropriately, is users of the web.

We may have a tendency be overly concerned by devices, and grow frustrated by their growing variety of features and characteristics.

Yet **device fragmentation is merely a reflection of human diversity** and consumers exercising their right to choose.



The Verge, as experienced over an average 3G connection
webpagetest.org/result/150915_C3_1D4N/

Empathy for consumers is seen as **the domain of user experience designers** and researchers. Yet while a badly designed interface can adversely effect a site's usability, so can a poorly considered technology choice.

For example, consider the experience users may face using The Verge. The speed of this site is the result of decisions made not just by the developers, but **visual designers, product managers, content designers, sales teams**...

We **all have a responsibility** to consider how our work may affect the resulting experience.

In physical product design and architecture, the concept of universal design is well understood.

This is supported by seven principles, **each promoting the creation of products that are usable by anyone**, regardless of age, ability or status.

Guidelines are also included, for example:

- “Eliminate unnecessary complexity”
- “Minimise repetitive actions”
- “Arrange information consistent with its importance.”

Universal, accessible design is just good design, full stop.

Take for example, this vegetable peeler.

In 1989, Sam Farber, inspired by his wife's arthritis, **redesigned the conventional vegetable peeler**, replacing its metal handles with softer grips.

This meant that **anyone, regardless of strength or manual dexterity**, could use this tool.

Farber also **considered its aesthetic appeal also**: accessible products shouldn't mean ugly products.

He applied this approach to a range of products; Good Grips became an internationally recognised, award-winning brand, while Farber's company, OXO, went on to generate significant sales.

OXO remain advocates of designing inherently accessible products, and they note on their website that:

> *When all users’ needs are taken into consideration in the initial design process, the result is a product that can be used by the broadest spectrum of users.*

Equivalent textual alternative for an image resource:

```
<img src="charles-bridge.jpg" alt="Charles Bridge in Prague"/>
```

Fallback content for browsers that don’t support `<video>`:

```
<video controls poster=“prague-placeholder.jpg">
    <source src="/videos/prague.ogv" type="video/ogg"/>
    <source src="/videos/prague.mp4" type="video/mp4"/>
    <a href=“/videos/prague.mp4">Download this video</a>
</video>
```

In fact, you could say that many of the technologies and specifications we use today feature aspects of universal design.

Beyond specifications like **WAI-ARIA that increase the accessibility of dynamic interfaces**, HTML has long included **basic accessibility features**.

- The `alt` attribute allows authors to add textual alternatives to images
- The `<video>` element allows fallback content to be provided if a particular media plug-in or codec isn’t available.

You can find many examples of this sort of behaviour throughout HTML

> *In case of conflict, **consider users over authors** over implementors over specifiers over theoretical purity.*

HTML Design Principles: Priority of Constituencies
w3.org/TR/html-design-principles/

Examples can also be found within W3C and WHATWG working groups.

A **key principle used in the design of HTML5** concerns itself with how authors should assess additions or changes to the specification.

Called the priority of constituencies, it states that:

> *In case of conflict, consider users over authors over implementors over specifiers over theoretical purity.*

I think we can **use similar prioritisation guidelines** on our own projects.

Yet too often we see technology **choices made that prioritise developer convenience over user experience**. Widespread use of front-end MVC JavaScript frameworks = havoc for users who don't match the precise browsing requirements.

Choices like this are made because of **an empathy gap between users and ourselves**.

e.g. Attached to a high-resolution display/super-fast broadband connection, difficult to visualise resulting experience on a low-powered mobile device/ unreliable connection.

The wider the gap between those making a product and those using it, the greater the likelihood that they'll make wrong choice.

We need to get much closer to users.

User research and usability testing helps us see how users interact with our products.

Having **different disciplines** (developers, interface and content designers, product managers) **participate** can ensure this learning is widely shared.

Susan Robertson wrote about how spending a week answering support emails, **gave her new insights into how customers were using the application** she was building:

> *Rather than a faceless person typing away on a keyboard, users become people with names who want to use what you are helping to create.*

Example

Universal Personas

Olivia

- 33 years old
- Lives in a her newly–purchased one–bedroom apartment in Woolwich
- Works in finance on a comfortable salary (70k) in Canary Wharf
- In a relationship but not currently cohabiting
- Technology: Proudly Apple. Owns a Macbook Air and an iPhone 6 that she purchased for personal use.

How else can we involve universal design principles within our day-to-day practice, and make our products more inclusive?

On a recent project, although we **had earlier acknowledged** that visitors to a website included a large number of middle-aged users with poor eyesight, this wasn’t factored into the user personas we had created.

Realising this later on, we asked ourselves how we could better factor accessibility issues into this tool?

Personas need to be grounded in a degree of reality, and **backed up with research**, so such a persona could be easily dismissed.

Peter

...recently had a skiing accident, and broke his wrist

Reduced motor ability

flickr.com/photos/mikelo/3139837006

Olivia

...often uses her phone when walking to work

Poor eyesight

flickr.com/photos/ktoine/6263683606

Sanjita

...has two children who play musical instruments

Hearing loss

actiononhearingloss.org.uk

Instead, we thought about enhancing our existing personas; giving them short-term accessibility concerns would provide a proxy for long-term disabilities:

- A persona could have recently broken their wrist in a skiing accident, meaning they have temporary motor restriction.
- Acknowledging that users may access websites in bright sunlight, may be similar to the experience of users with poor eyesight.
- If a persona lives in a loud household, she may exhibits the same characteristics as someone with hearing loss.

By adding these stories to our personas, we ensure we account for these issues; plus, we can probably have greater appreciation for the difficulties they cause.

But! Why did it take so long for us to realise that our personas where not truly reflecting the audience of the product we were designing?

Maybe if our team was more inclusive, and featured members with disabilities, we would we have noticed earlier.

In her article “Universal Design IRL” Sara noted that:

> *[T]he best way to understand the audiences we design for is to know those audiences. And the best way to know people is to have them, with all their differences of perspective and background—and, yes, age and gender and race and language, too—right alongside us.*

Perhaps it’s no coincidence that as we learn more about the diversity of our users, **we’re beginning to acknowledge the lack of diversity within our own industry**.

1

Reflect the diversity of users in our practice

By building inclusive teams that listen to – and even work alongside – users, we can achieve wider reach.

Which gives us our first principle.

If we **strive to reflect the real world**, we can build more empathetic and effective teams, and in turn, **build better products**.

The medium

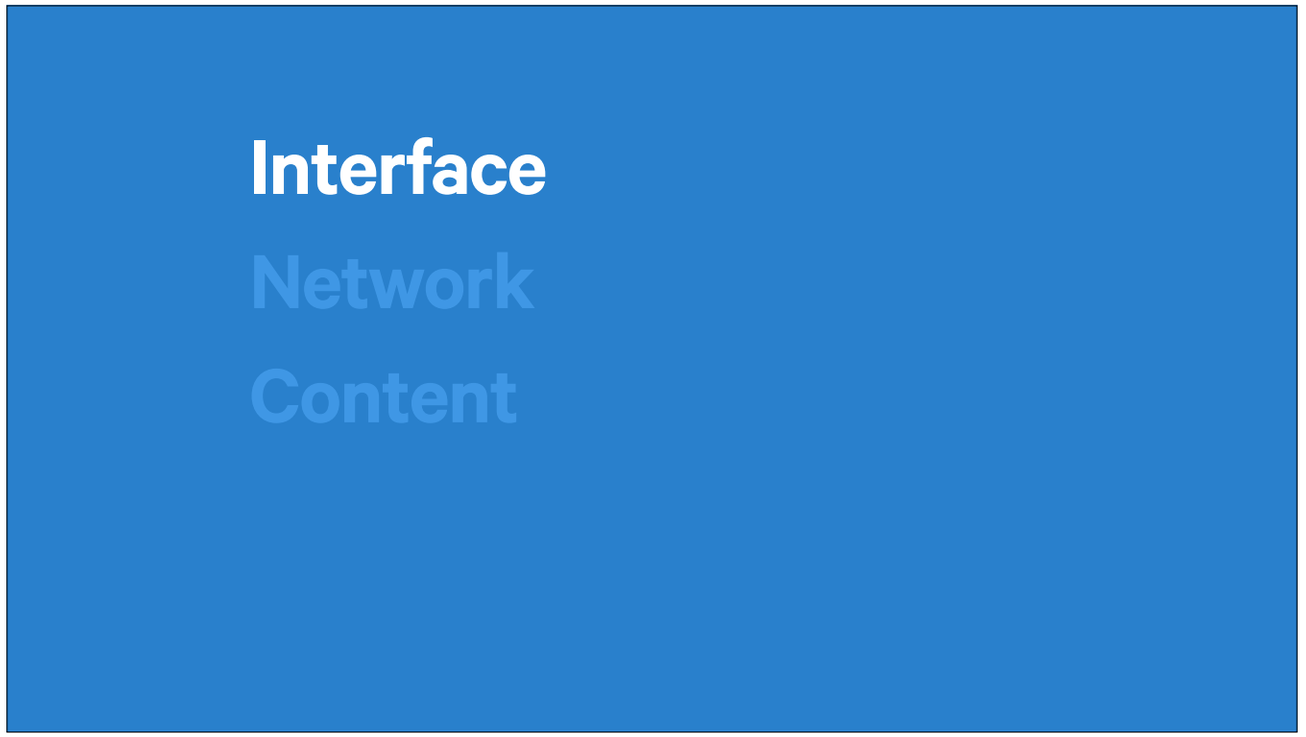
The web is a fluid and unpredictable medium.

Our second concern is the medium itself and **its unpredictable (and sometimes unreliable) nature**.

For our work to reach users, it needs to travel across unreliable networks before being consumed by different devices with unknown characteristics.

It’s hard to make decisions if you’re unable to predict the outcomes.

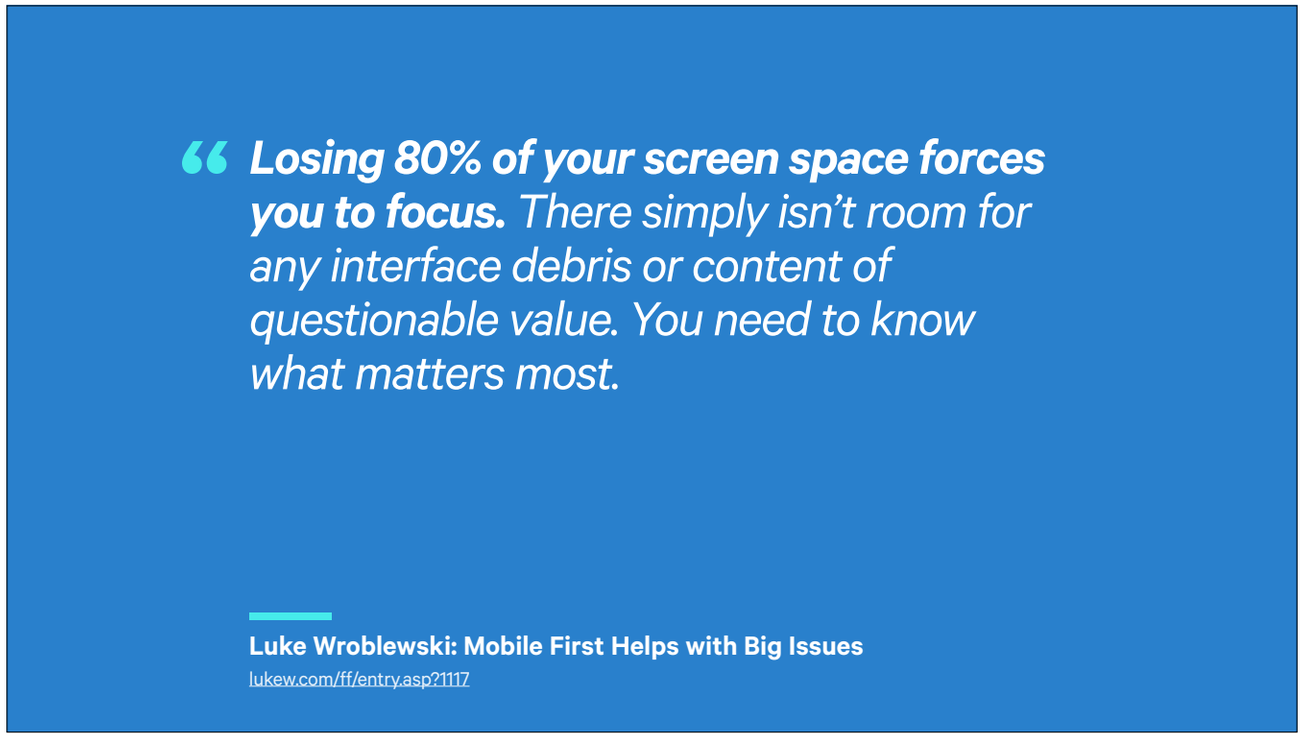
However, if we choose to view a website through different lenses, **we can uncover areas of constraint**, which in turn can highlight opportunities for enabling **greater reach and adaptability**.



Perhaps we consider our products from the perspective of a more constrained interface?

In 2009, Luke Wroblewski proposed *Mobile First*.

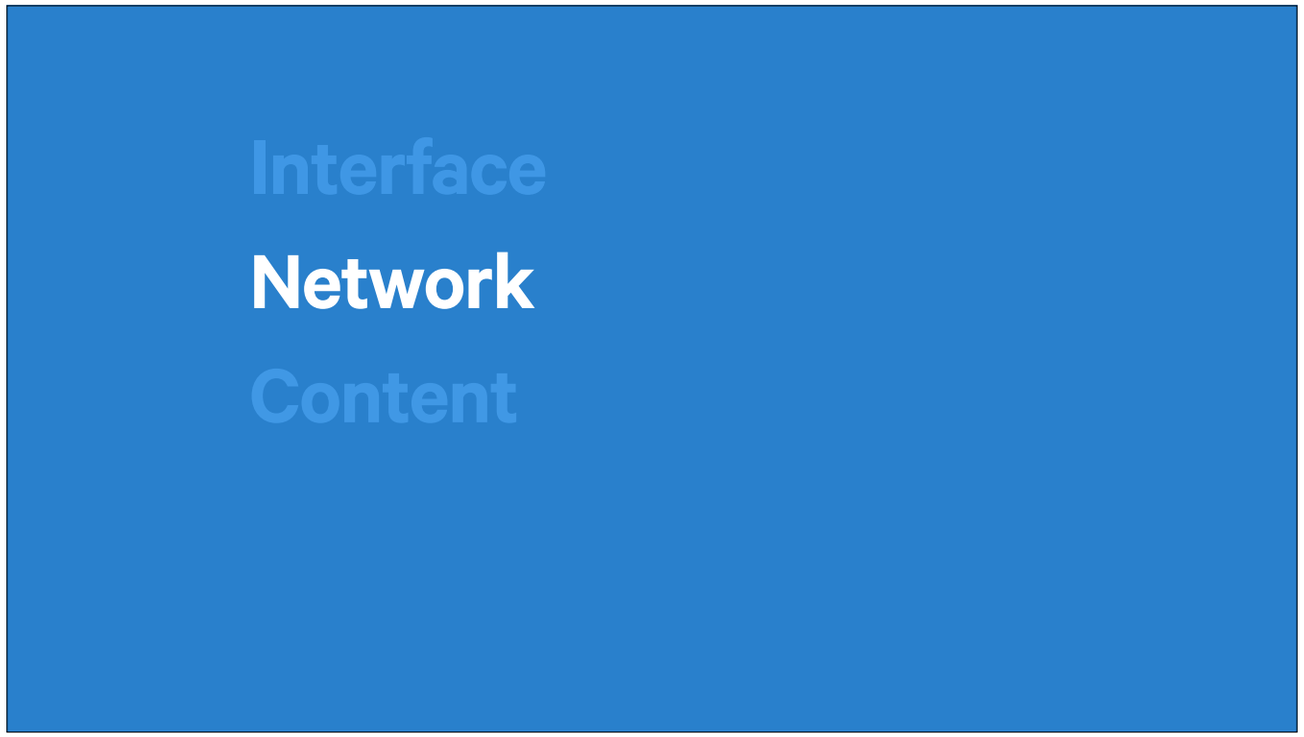
This approach asked us to **consider how interfaces could take advantage of mobile device capabilities** *before* thinking about their manifestation in desktop browsers.



This approach was partly driven by the rapid growth of mobile, but it also has a side benefit:

> *Losing 80% of your screen space forces you to focus. There simply isn’t room for any interface debris or content of questionable value. You need to know what matters most.*

By asking questions about **which parts of an interface are critical** and which are not, we can decide whether those non-critical parts are loaded conditionally or lazily—or perhaps not at all.



Addressing the **realities of its reliability and availability**, in 2013 the team at Huddle proposed considering the network, with **offline first**.

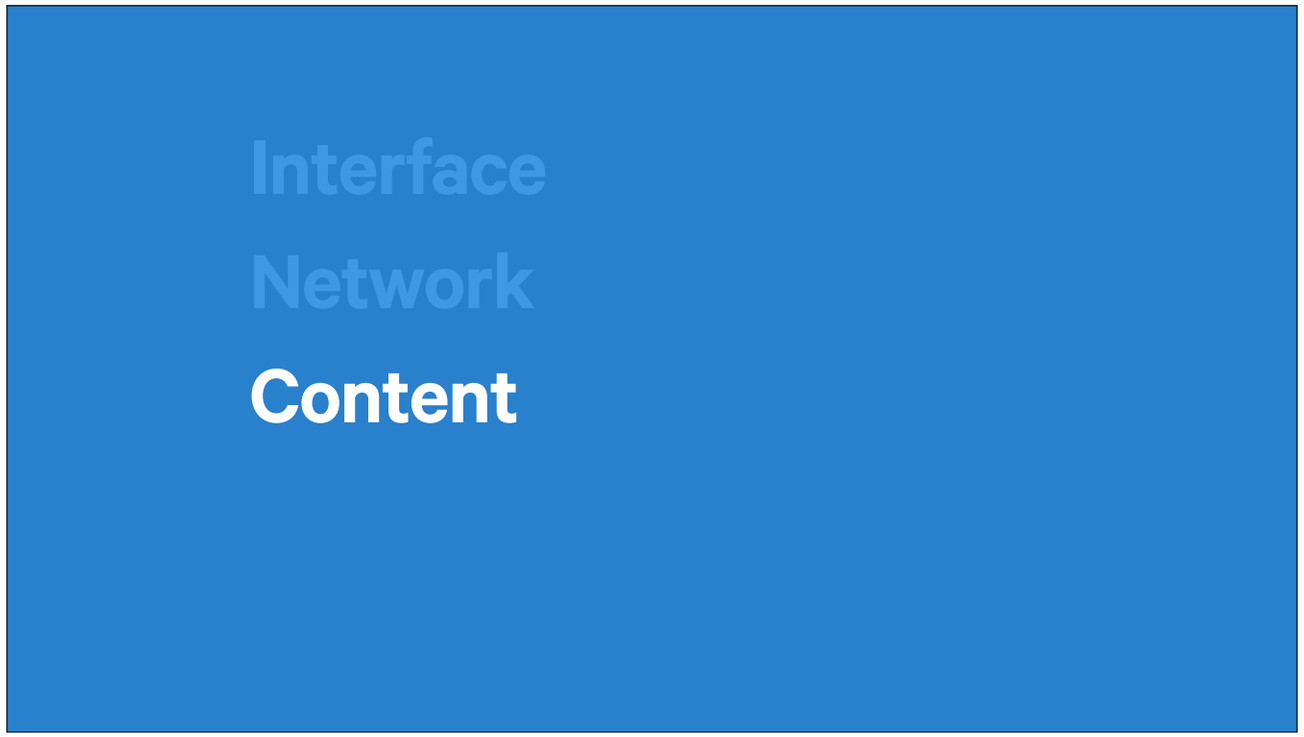
Rather than treat offline as an edge case, plan for it:

- Online: Preemptively downloading assets and synchronising data
- Offline: Use aggressive caching and client-side computation

We can create **experiences that work regardless of connectivity**.

Others have suggested starting with **URLs or an API first**, using these to think about where content lives within a system.

Each approach embraces the underlying fabric of the web to help us build more robust and resilient products.



What about content? Although the adage of ‘**content is king**’ was often heard, historically this meant pouring lorem ipsum into **empty rectangles**.

> *In order to embrace designing native layouts for the web – whatever the device – we need to shed the notion that we create layouts from a canvas in. We need to flip it on its head, and* ***create layouts from the content out****.*
>
> **Mark Boulton: A Richer Canvas**
> markboulton.co.uk/journal/a-richer-canvas

In 2011, reflecting on hundreds of years of graphic design practice in his New Canon, **Mark Boulton signalled a move away from our *canvas in* approach**, to one where layouts are designed from the *content out*.

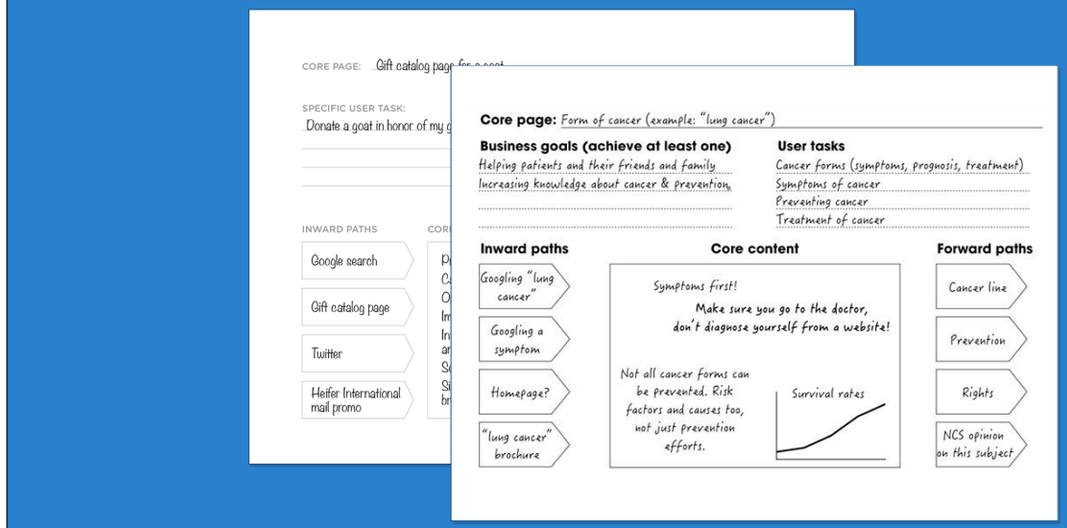
By defining visual **relationships based on page elements**, and using **ratios instead of fixed values**, we could **imbue connectedness within a page**, independent of its dimensions.

This might suggest that we start designing with the content first, but **content and design ideally should work hand-in-hand**.

Mark later clarified his position, and suggested we consider **structure first, content always**:

> *You can create good experiences without knowing the content. What you can’t do is create good experiences without knowing your content structure.*

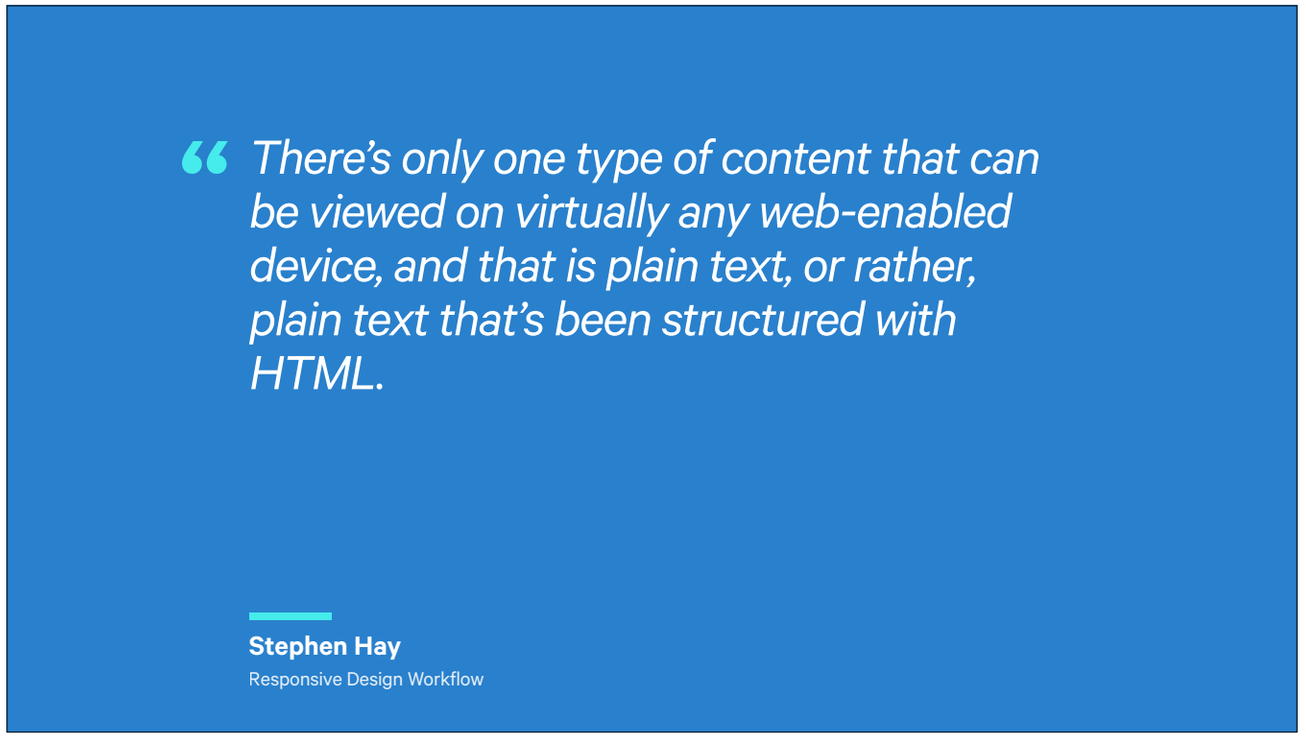
By thinking about the structure, we can use representative content as we design; we don’t have to wait for the final copy.



The Core Model: Designing Inside Out for Better Results
alistapart.com/article/the-core-model-designing-inside-out-for-better-results

This fits in well with the Core Model, an idea first introduced by Are Halland at the IA Summit in 2007.

By **asking questions about a site's core content**—what task it needs to perform, what information it should convey—we can help clients think more critically about their strategic objectives, business goals, and user needs.



Each of these approaches has a time and place, as provides different emphasis.

But if you were to ask me where I would start designing, I would suggest plain text. As Stephen Hay reminded us in his book Responsive Design Workflow:

> *There’s only one type of content that can be viewed on virtually any web-enabled device, and that is plain text, or rather, plain text that’s been structured with HTML*.

Let's imagine an article reviewing a film. A key piece of information is the rating the reviewer gave.

With our graphic design hat on, we might think about this purely in visual terms: the rating shown as a series of stars, with the filled in stars representing a rating.

```
<div class="rating">
    <img src="star_whole.png" alt="Star number 1"/>
    <img src="star_whole.png" alt="Star number 2"/>
    <img src="star_whole.png" alt="Star number 3”/>
    <img src="star_empty.png" alt="Star number 4"/>
    <img src="star_empty.png" alt="Star number 5"/>
</div>
```

And the markup may be as follows. This is **based entirely on the visual representation**, making an assumption about the capabilities of the device consuming this content.

Unsurprisingly, this is not very accessible, either.

What if we were to think about the interface as it might be read out as part of a phone conversation instead?

What questions might a user be asking, and how might I structure a response?

“How is this movie rated?”

```
<div class="rating">
    <p>This movie is rated 3 out of 5 stars.<p>
</div>
```

A user might ask: “*how is this movie rated?*”

Our marked up answer would then be as follows:

“This movie is rated 3 out of 5 stars”

“How is this movie rated?”

```
<div class="rating rating--3">
    <p>This movie is rated 3 out of 5 stars.<p>
    <span class="rating__star-whole" aria-hidden="true"/>
    <span class="rating__star-whole" aria-hidden="true"/>
    <span class="rating__star-whole" aria-hidden="true"/>
    <span class="rating__star-empty" aria-hidden="true"/>
    <span class="rating__star-empty" aria-hidden="true"/>
</div>
```

Additional markup can then allow us to add extra visual embellishment.

Here, empty spans are used as styling hooks, with ARIA attributes to ensure these are not parsed by accessibility software.

This simple piece of markup, demonstrates **a layered experience**; a core piece of content that allows for a degree of enhancement.

2
Start from a point of greatest adaptability

By making fewer assumptions about context and interface, focusing more on users' tasks and goals, we can create more adaptable products.

So our second principle is this.

If an interface works on a mobile device, it'll work on a larger display. If data can be interrogated when there's no network, an unreliable connection will be of little hinderance.

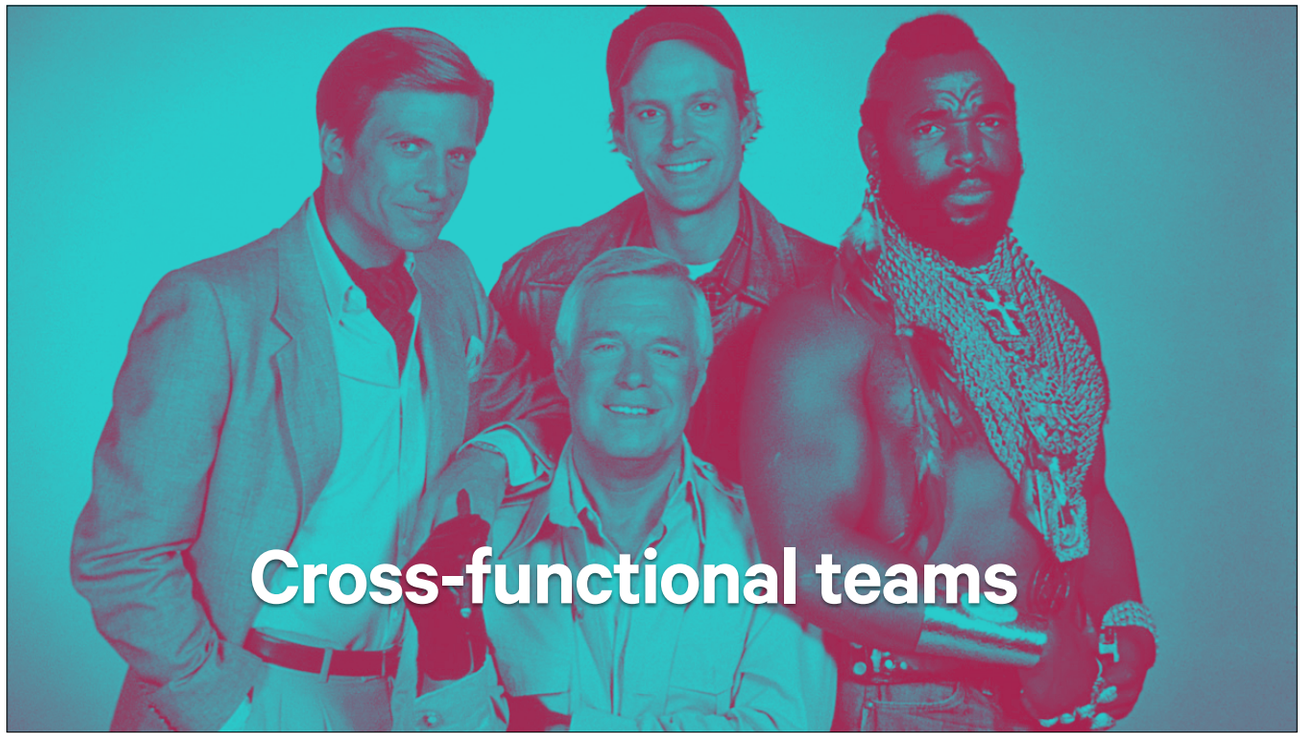
If content forms the basis of our design, a user can still attain information if only plain text can be served.

Optimisations based on more uncertain assumptions can be layered on later, safe in the knowledge we've provided fallbacks.

Ourselves

The web is built upon complex and interwoven technologies.

Our third concern is about how we, those that build for the web, might **navigate the elaborate and complicated landscape of tools and processes**.



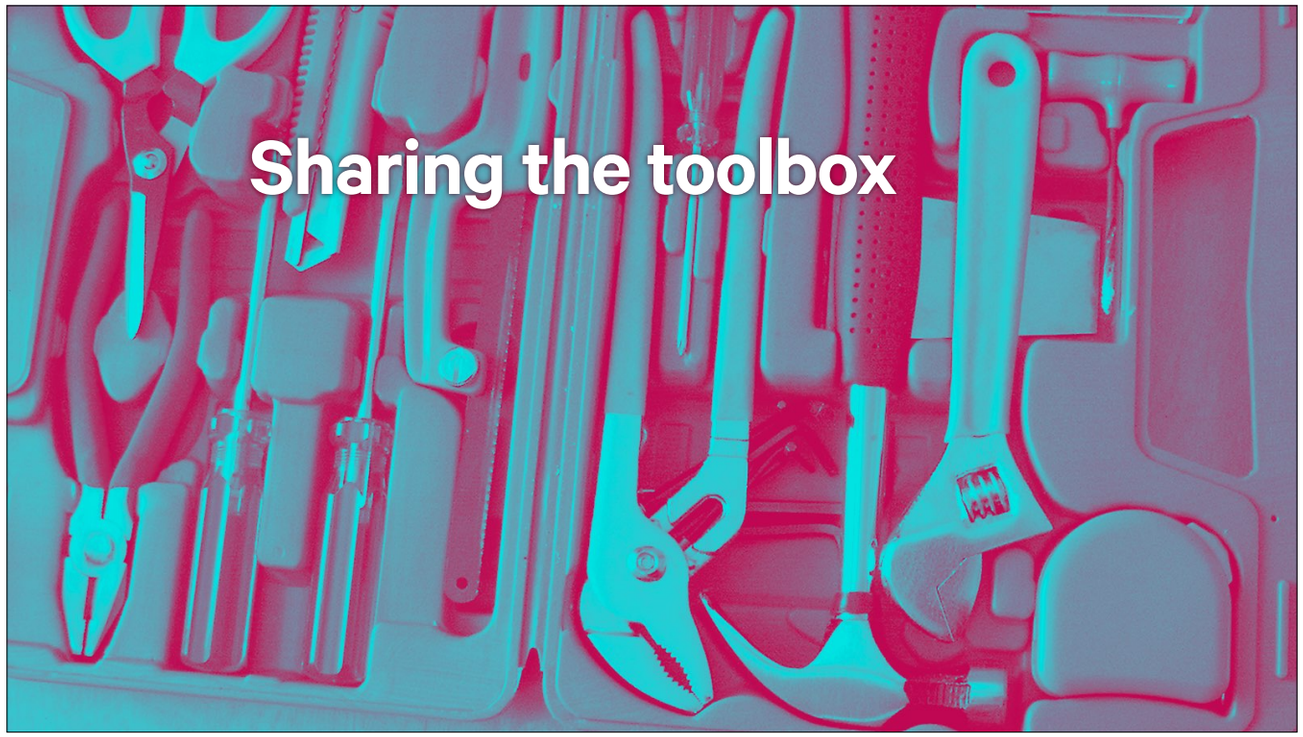
As responsive design has become embedded within organisations, teams are becoming **increasingly collaborative and cross-functional**.

Previously well-defined **roles are beginning to merge**.

Job titles starting to reflect this change: see ‘full-stack developer’ or ‘product designer’.

Tools once the preserve of specific disciplines are now **being co-opted**.

e.g. Prototyping an animation may require writing JavaScript, while building a modular component library may require understanding visual language and design theories.

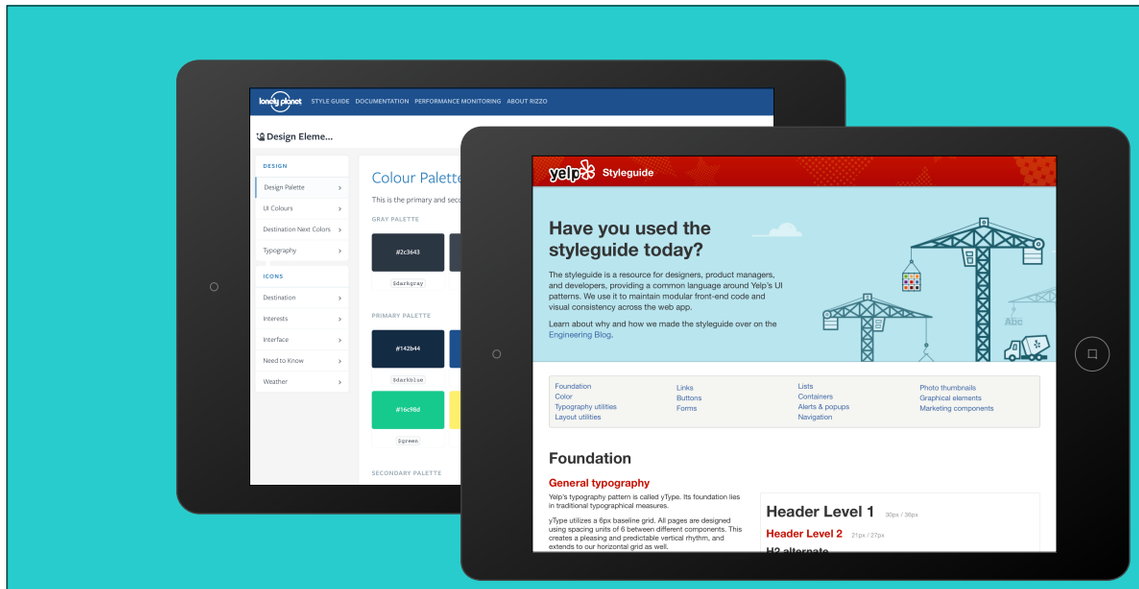


As we start to use this shared box of tools, we **need to ensure different people are able to pick them up**, understand how they work, and the effect they may have.

If the tools used are too opaque, and processes difficult to adopt, then **opportunities for collaboration will diminish**.

Make systems too complex, and on-boarding new members of a team will become difficult and time consuming.

We need to constantly **make sure our work is accessible to others**.



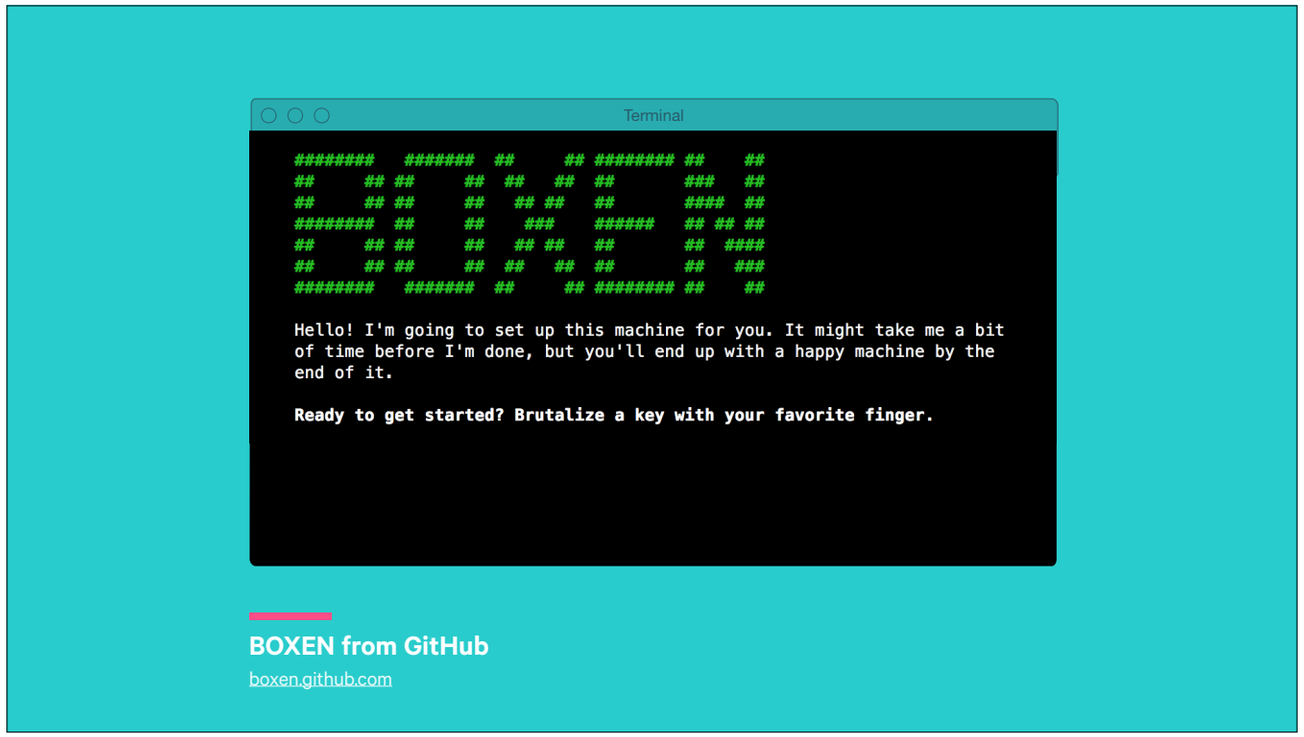
Style guides for Lonely Planet and Yelp

styleguides.io/examples.html

The growing use of front-end style guides is one example of this, and **demonstrates a maturing relationship between disciplines**.

Rather than producing static, bespoke layouts, designers are employing more systematic design approaches.

Front-end developers are taking these and building pattern libraries and modular components, a form of delivery that fits in better with back-end development approaches.



On the other side of the coin, I look at tools like Boxen.

A common complaint for those wishing to use software like preprocessors and task runners is that they require knowledge of the command line.

Tools tease new recruits with one-line install instructions, but the reality often involves hair-pulling and shaving of yaks.

With this in mind, GitHub created Boxen, **a tool that allows anyone in their company can run a local instance of GitHub.com** on their own computer by typing a single command.

A shared language

Closer collaboration between designers and developers can see the creation of a shared language, with **common names used across all aspects of a products creation**.

I saw an example of this while working at the Guardian, where we made heavy use of Sass.

At times, the inclusion of Sass meant aspects of the code became overly complicated, making it difficult for those new to the team (or unfamiliar with the latest Sass features) to understand the underlying system.

SCSS input

```
.element {
    @include fs-header(1);
}
```

Compiled output:

```
.element {
    font-size: 16px;
    line-height: 20px;
    font-family: "Guardian Egyptian Headline", Georgia, serif;
    font-weight: 900;
}
```

github.com/guardian/guss

However, where it worked best, was where it allowed for **the creation of a set of declarative functions and mixins**.

These meant the **design language could be embedded within code**, using naming conventions understood by the whole team.

This almost allowed us to extend the declarative nature of CSS, adding naming conventions specific to the design needs of the Guardian.

The Guardian: GUSS

github.com/guardian/guss

Many aspects of the site can be **described using real names and words**, without getting bogged down in the specifics.

Rather than say 20px header, with 24px line-height, instead everyone can say: ‘header font size’, and people will know what that means.

Common conventions

But we don’t always need to involve tools like Sass or Grunt or the many other libraries and frameworks we can look at.

In fact, trying to solve technology problems by employing more technology ignores the fact that **establishing conventions can be just as helpful**, and easier for others to adopt.

`.block`
Encapsulates a standalone entity that's meaningful on its own.

`.block__element`
Parts of a block and have no standalone meaning.

`.block--modifier`
Flags on blocks or elements, used to change appearance, behaviour or state.

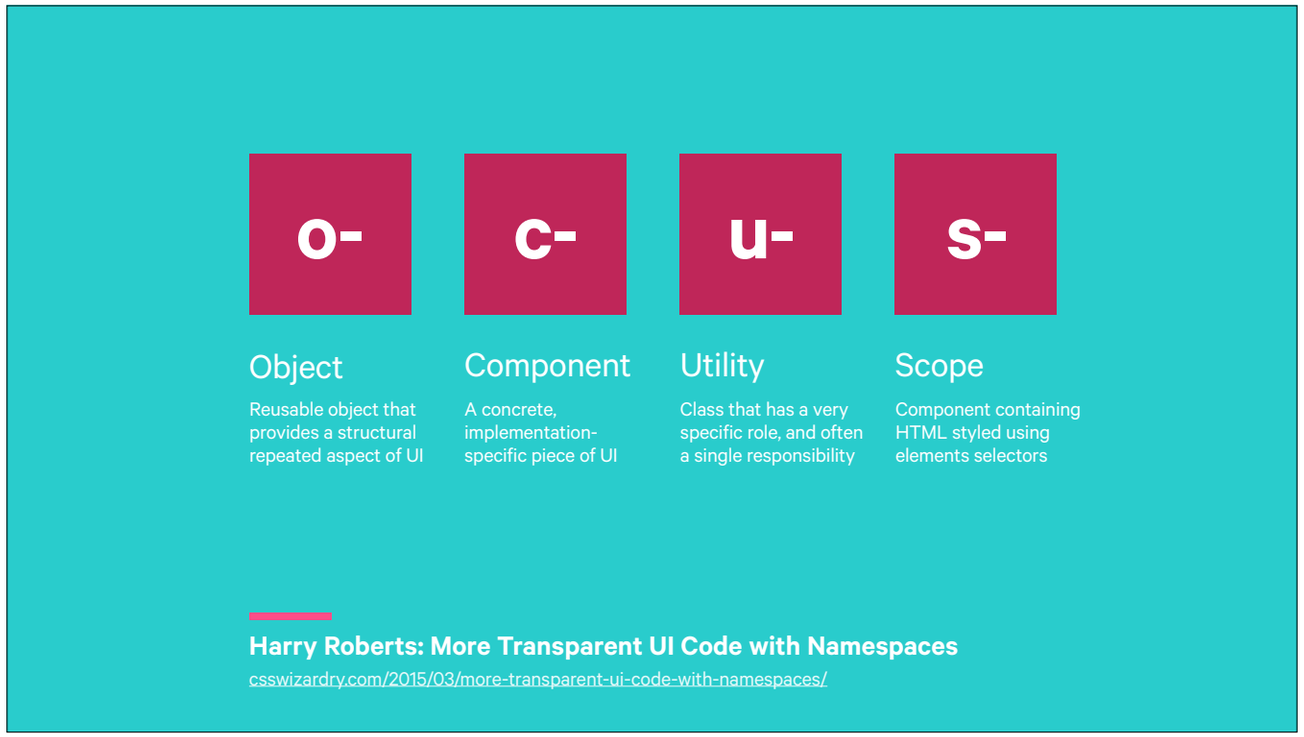
getbem.com/naming/

Question: Who's familiar with the BEM naming methodology?

This helps CSS styles remain scoped, encapsulated, and easier to maintain.

This approach has **no dependancy on a particular technology**; it is purely a set of documented conventions.

We could have been using BEM in 2005, should we have foreseen the need.



Another technique I've been using is CSS namespaces, as advocated by Harry Roberts.

His post goes into more detail than I am able to here, but I'll describe a few examples:

Object: *Reusable structural objects*

Component: *Implementation specific pieces of UI*

Utility: *Classes with a single specific role*

Scope: *Classless HTML*

```
/* Utility: Clear floats */
.u-clearfix { … }
.u-hidden { … }

/* Object: OCSS ‘Media’ */
.o-media { … }
.o-media__image { … }

/* Component: Navigation */
.c-nav { … }

/* Component: Navigation list */
.c-nav-list { … }
.c-nav-list__item { … }
.c-nav-list__label { … }
```

```
/* Scope: HTML within prose */
.s-prose h1 { … }
.s-prose h2 { … }
.s-prose p + h2 { … }
.s-prose ul > li { … }
```

Again, no reliance on technology, just agreement within a team to accept a set of conventions.

If these conventions are rigid enough, you may even be able to test to ensure they are being followed.

`make install`	Install project dependencies → *Project 1 (Ruby):* `bundle install` → *Project 2 (Node):* `npm install`
`make build`	Run build steps
`make test`	Run unit tests, linting and other tests
`make run`	Run the application locally
`make deploy`	Deploy the application to environment

financial-times.github.io/next/docs/developer-guide/#make

Another practice gaining momentum is ensuring different projects created within a company all provide a standard set of commands for installing, testing and running new projects.

Don’t need to know specific requirements **to get a project up and running**.

These are some examples for those commands used at the Financial Times.

3
Build systems that humans understand

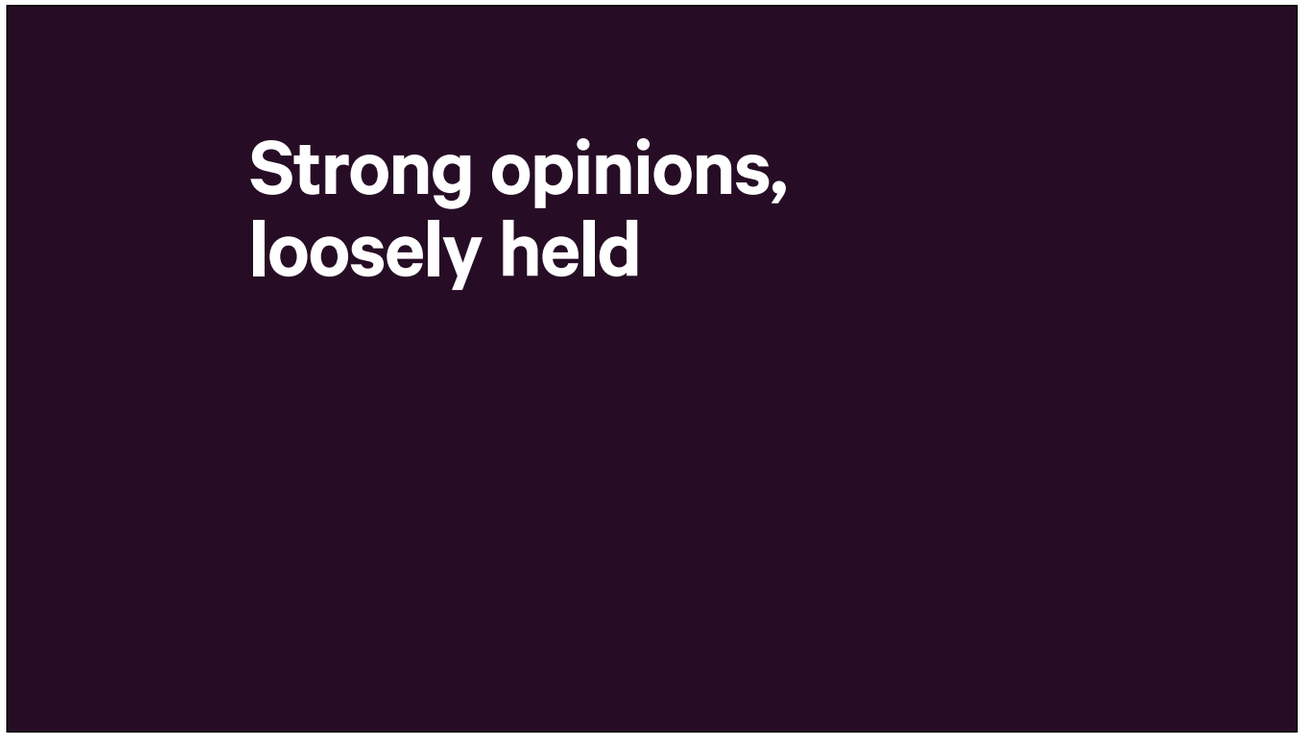
Choose tools that are approachable, simple to use and open to change so as to elicit greater collaboration.

So our third principle is this: **Build systems that humans understand**.

We need to choose and build tools that are **approachable**, **easy to use**, and **adaptable**.

Only this way can we elicit greater collaboration within our teams.

- We must listen more closely to the full spectrum of our audience, and **build greater empathy for them**.
- We must **embrace the web's unpredictable nature**, and see opportunities for optimisation where we previously saw barriers to entry.
- And we must **consider our fellow makers** in this process by building tools that will help us navigate these challenges together.



Another one of the principles I live by is this one:

> *Strong opinions, loosely held*

While these three principles are designed to endure, we should also recognise that our industry, and indeed the world we live in is rapidly changing.

That means, even these principles need to be responsive to change.

And that’s what I encourage you to do; change them.

These principles are available on Github, so please summit issues against them!

Or use the hashtag #RWDPrinciples to continue the discussion.

I look forward to hearing your suggestions!


Thank-you!
Děkuji!
paulrobertlloyd.com / @paulrobertlloyd
Attribution, Non-Commercial, Share Alike

Thank-you!