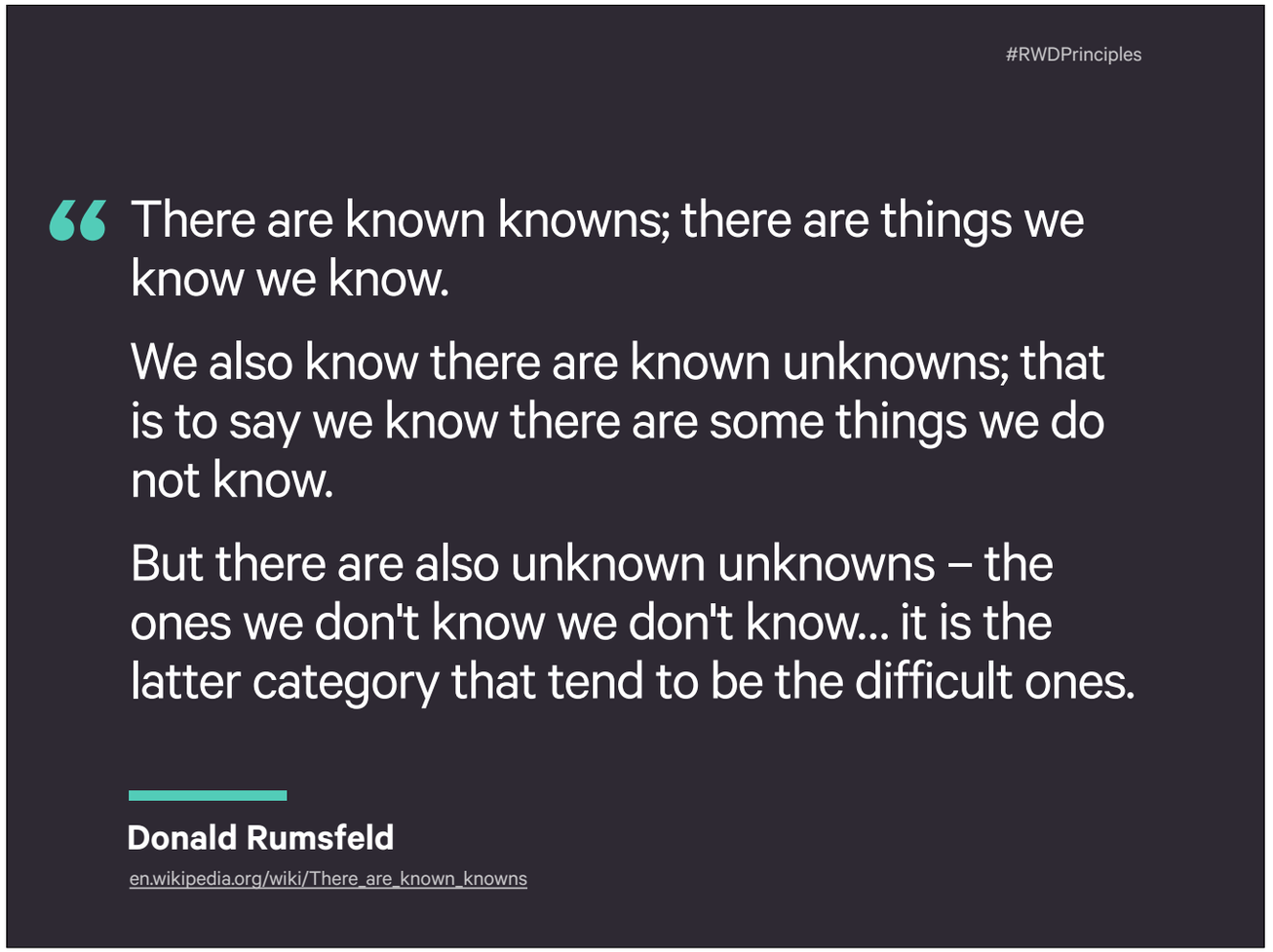
#RWDPrinciples

Responsive **Principles**

@paulrobertlloyd

Front-end London / 28 May 2015

I shall start my talk, by calling upon our good friend, Donald Rumsfeld.

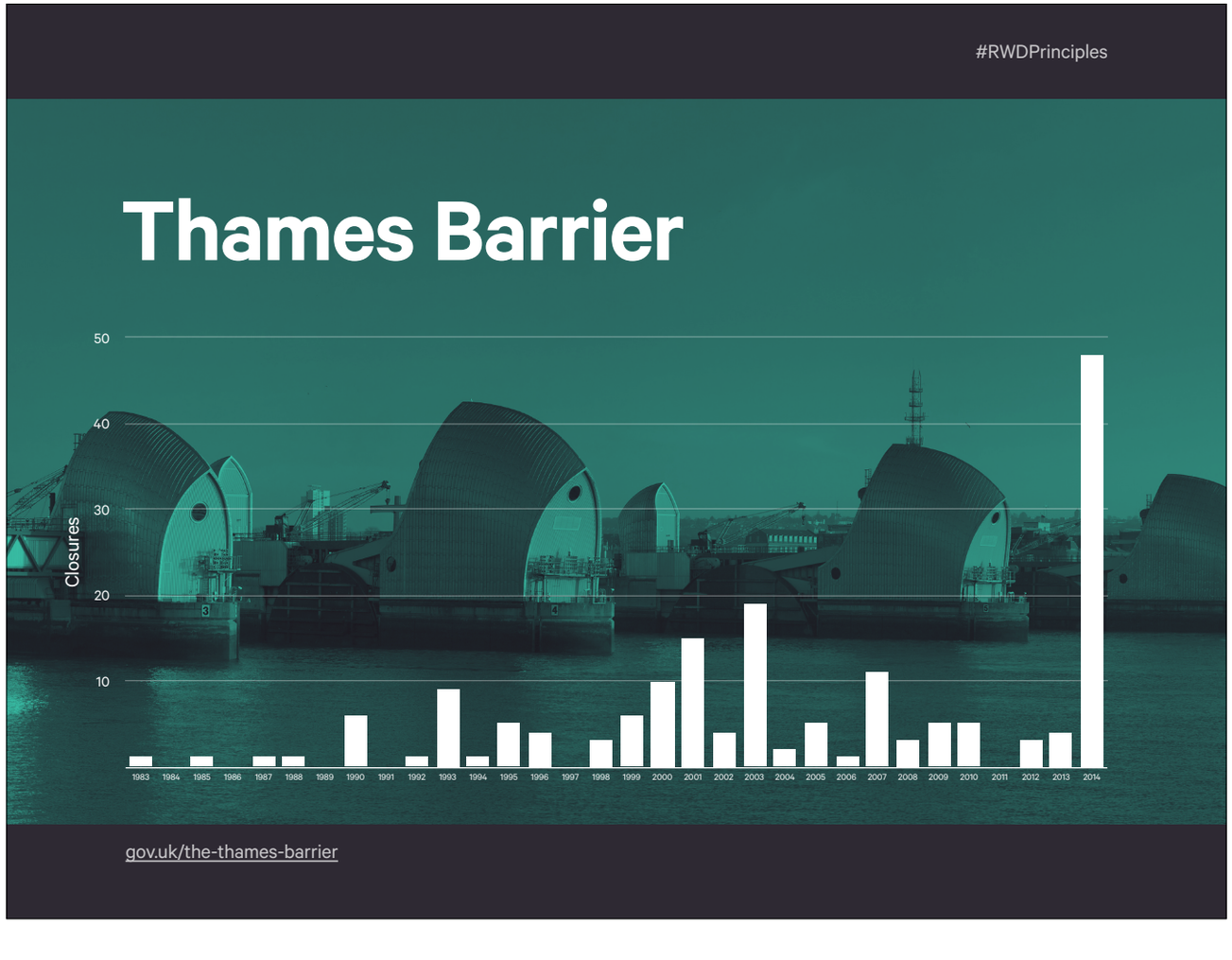


There are known knowns; there are things we know we know. We also know there are known unknowns; that is to say we know there are some things we do not know. But there are also unknown unknowns – the ones we don't know we don't know... it is the latter category that tend to be the difficult ones.

Our industry is one of constant change, which can be frustrating at times.

We are constantly dealing with known unknowns. Does a browser support a certain feature? What this interface make sense to users? How fast might the network me at any given time?

The future is a series of unknown unknowns, but we can look at trends, and make predictions.



This is the Thames Barrier

When it was opened in 1982, the barrier was closed once.

Last year it was closed a record 48 times.

This is just one of many indicators that can help us make a predictions about the future, and about the health of the planet.

And it doesn’t look good.

Smart phones increasingly take up our field of view.

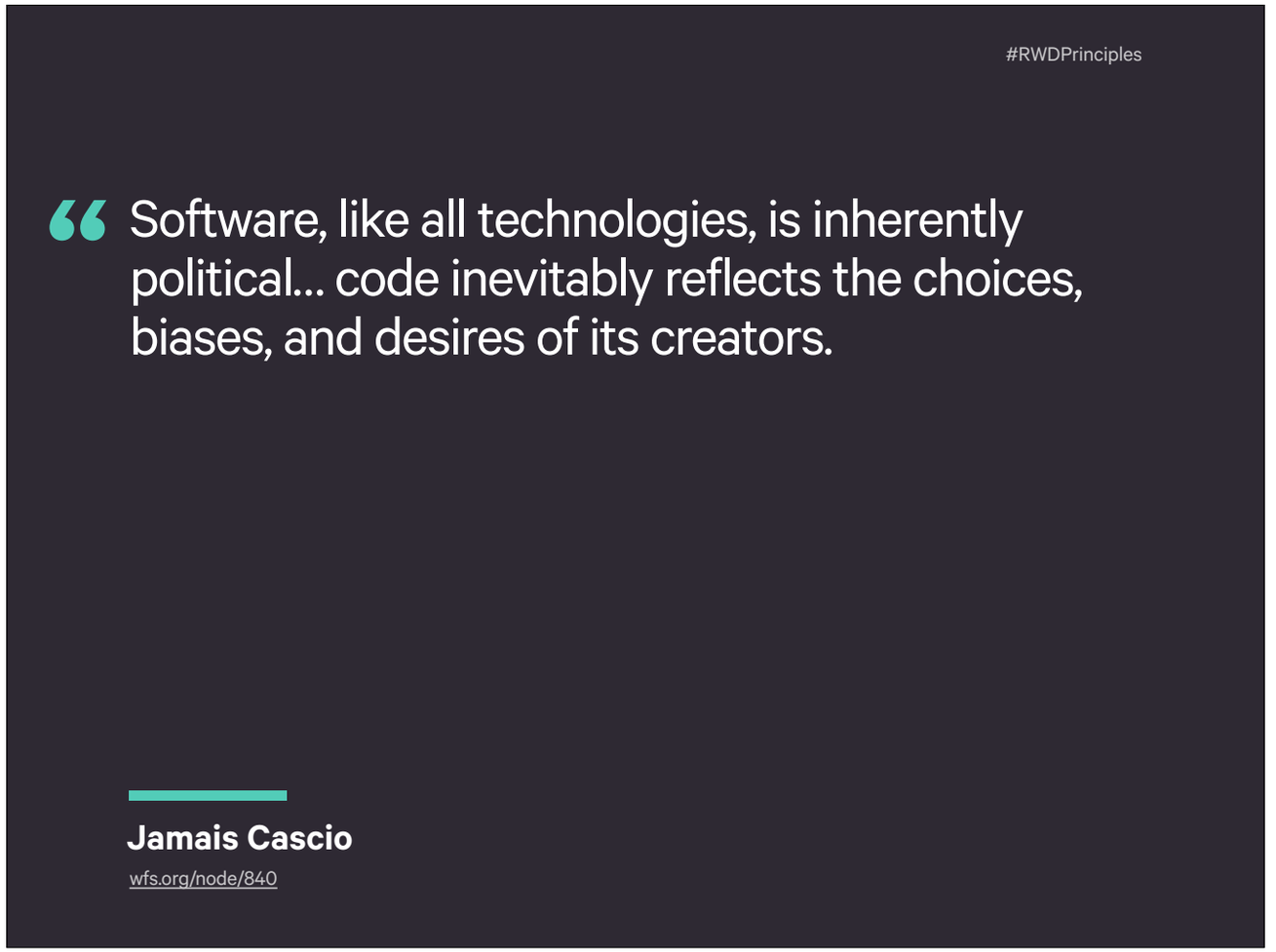
Thankfully, smart watches will help us loosen that tether, or so we are told.

Do smart watches solve a pressing need? Who are they designed for?

In his article, The Web's Grain, Frank Chimero wrote:

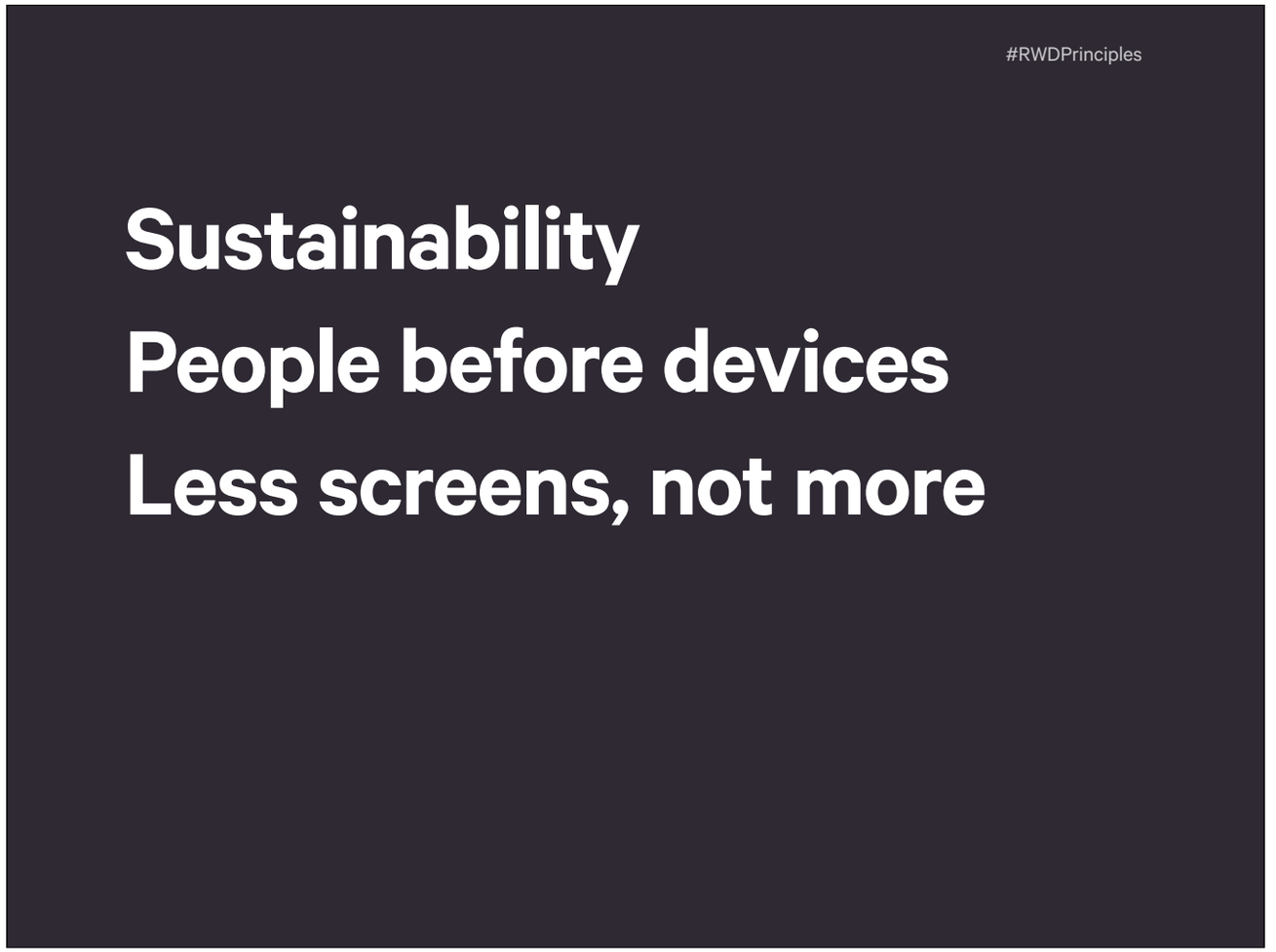
Technology only adds more—it is never this or that; it is always this and that.

Is this fetishisation of technology distracting us from what really matters?



This is one of my favourite quotes about technology:

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



So, here are some of my own choices and biases:

- I favour building products that last, and living in a sustainable manner.
- I prefer to think about the needs of users. If their device happens to be part of the consideration, so be it, but it's not my primary concern.
  I prefer to think of an internet of people, not an internet of things.
- I'm generally of the option we should be creating a world that has less screens, not more. (The world is getting a little too much like 'Children of Men', don't you think!?)

Thankfully, my own choices align well with those of the web.

This is for everyone

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

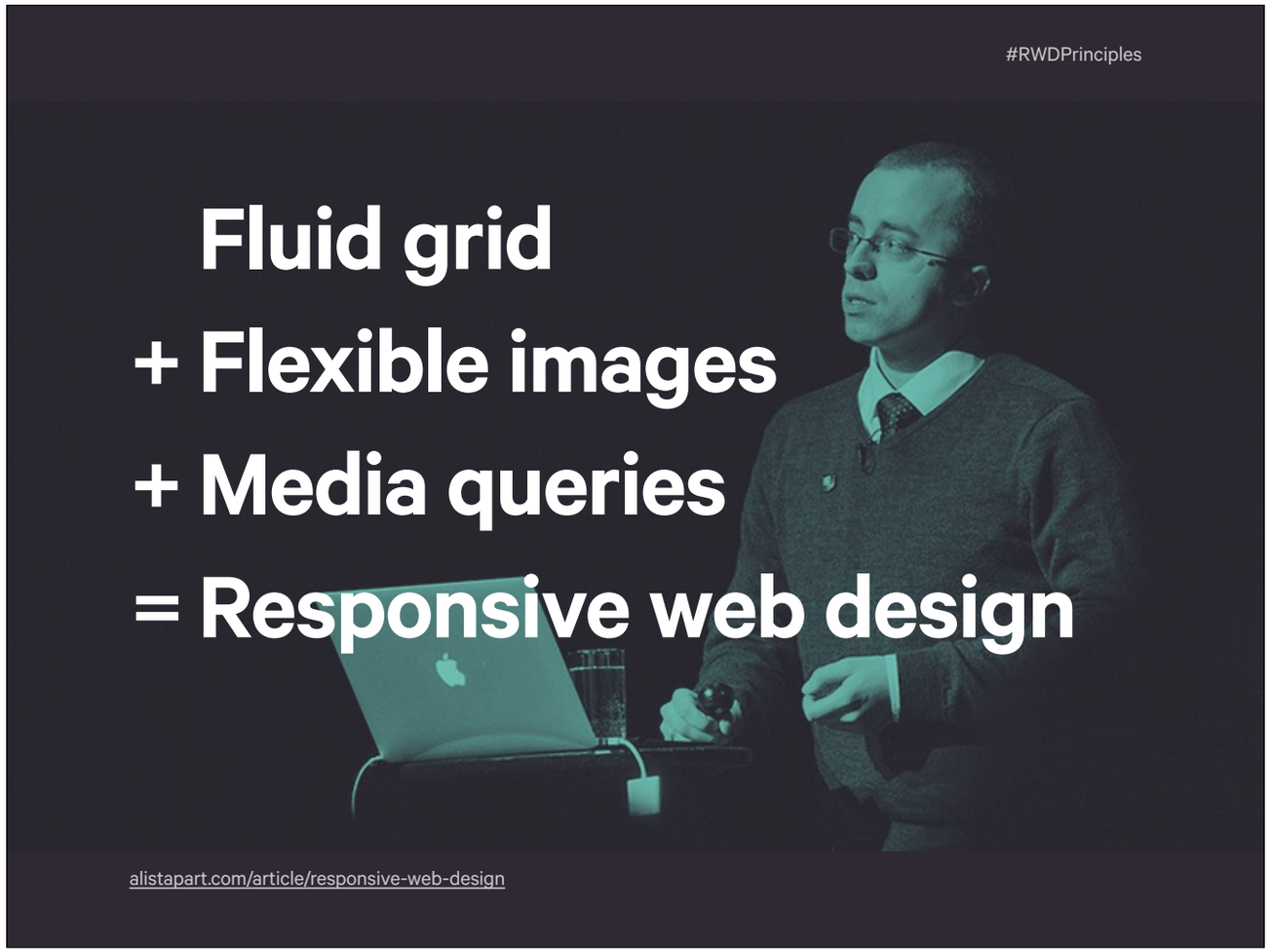
Tim Berners-Lee: Long Live the Web

scientificamerican.com/article/long-live-the-web/

Tim later expanded on this point in an article for Scientific American:

The web's primary design principle 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 – or universal access – is the web's killer feature.

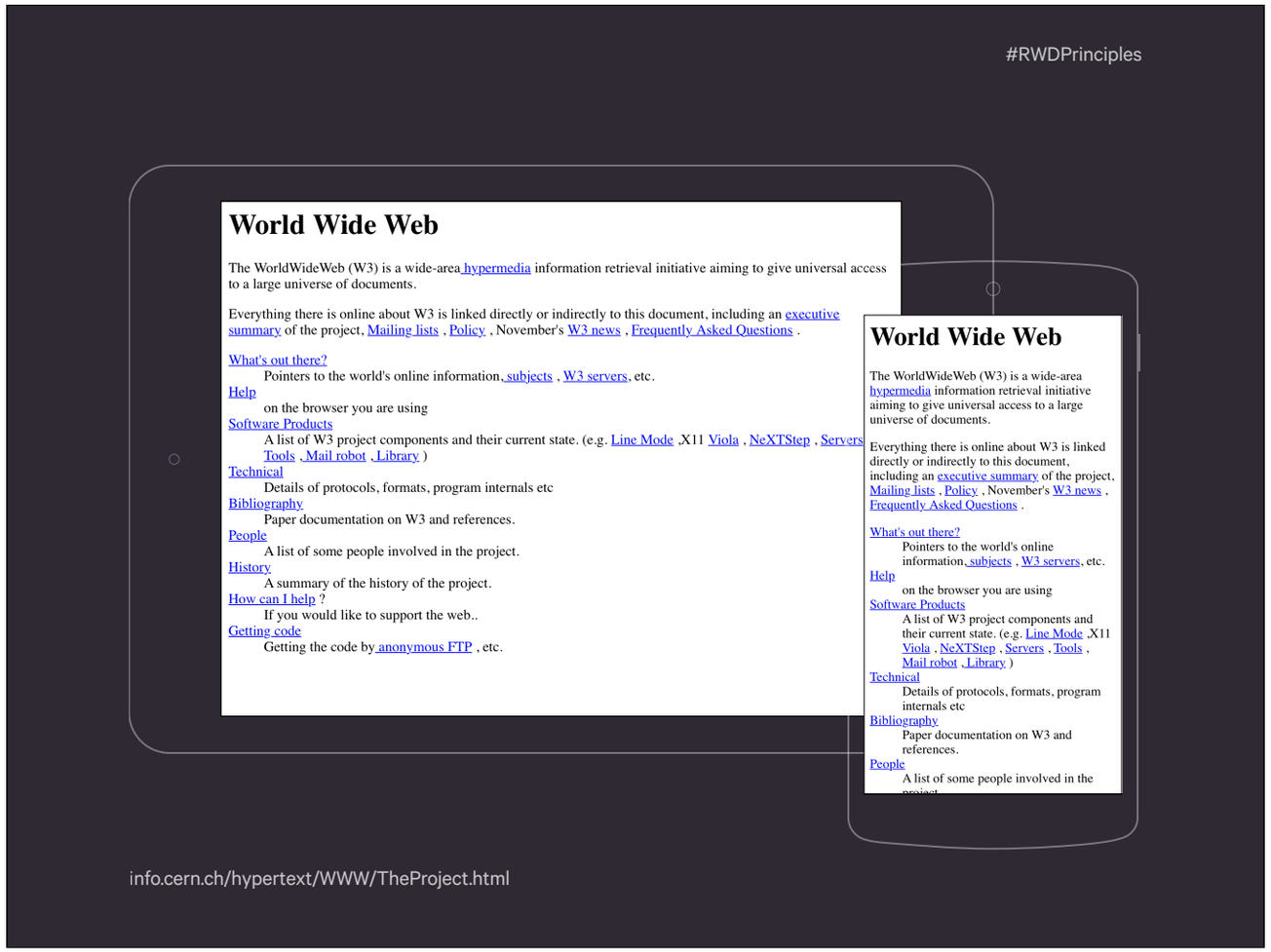


Responsive Web Design gets a little closer to this vision

This approach consists of three techniques that, when combined, help us produce websites more suited to a web accessed by different devices.

Not since the move away from table-based layouts or the adoption of web standards, has our industry undergone such radical realignment of both thought and practice.

Responsive web design is now an accepted best practice — but it is just that; a single tool within an suite of tools that we can use to build websites.



Now, in the past, we have looked at the first website, and told ourselves, hey that's responsive — we were right in the first place!

While this first humble exercise in HTML does have responsive characteristics — its layout responds to the width of the viewport — it doesn't use media queries, include images or have a grid. By definition it's not Responsive (capital R).

But there is something about this website that I find magical. It has endured.

That this site remains relevant and accessible to our work today as it did when it was created, speaks to that sustainable quality I spoke about before.

Frameworks

Plenty of frameworks and tools purport to help us build responsive sites, but often these abstractions only serve to distance ourselves from the realities of the medium and difficulties we face designing for it.

Creating layouts that respond to the dimensions of the viewport is progress, sure, but our work doesn't stop there.

If a responsive website can only be accessed if a user agent meets certain criteria, is it any better than a fixed width design that without those requirements?

Last year Ethan spoke about the need for a framework that would help us answer such questions. Instead, Ethan proposed:

A framework not about execution, but about philosophy and quality

This framework could help us frame ongoing discussions about our practice and act as a reference point for measuring the appropriateness of our work.

I love this idea. So, perhaps foolishly, I am going to try and do this.

> 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

adaptivepath.com/ideas/newsletter/archives/120209/

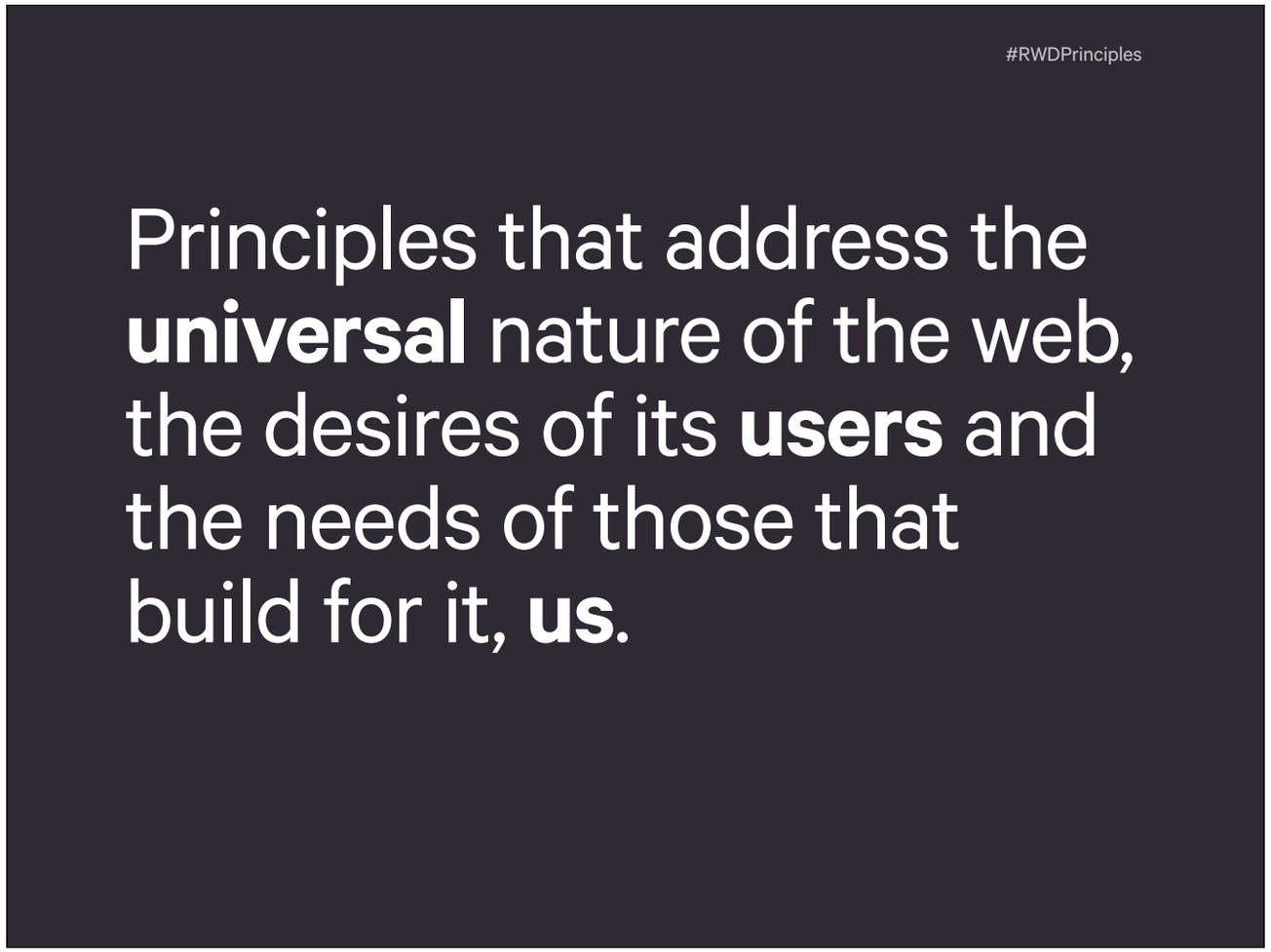
We can start this framework by creating a set of design principles.

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

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.

Our long term strategy is universal access, **this is for everyone**.



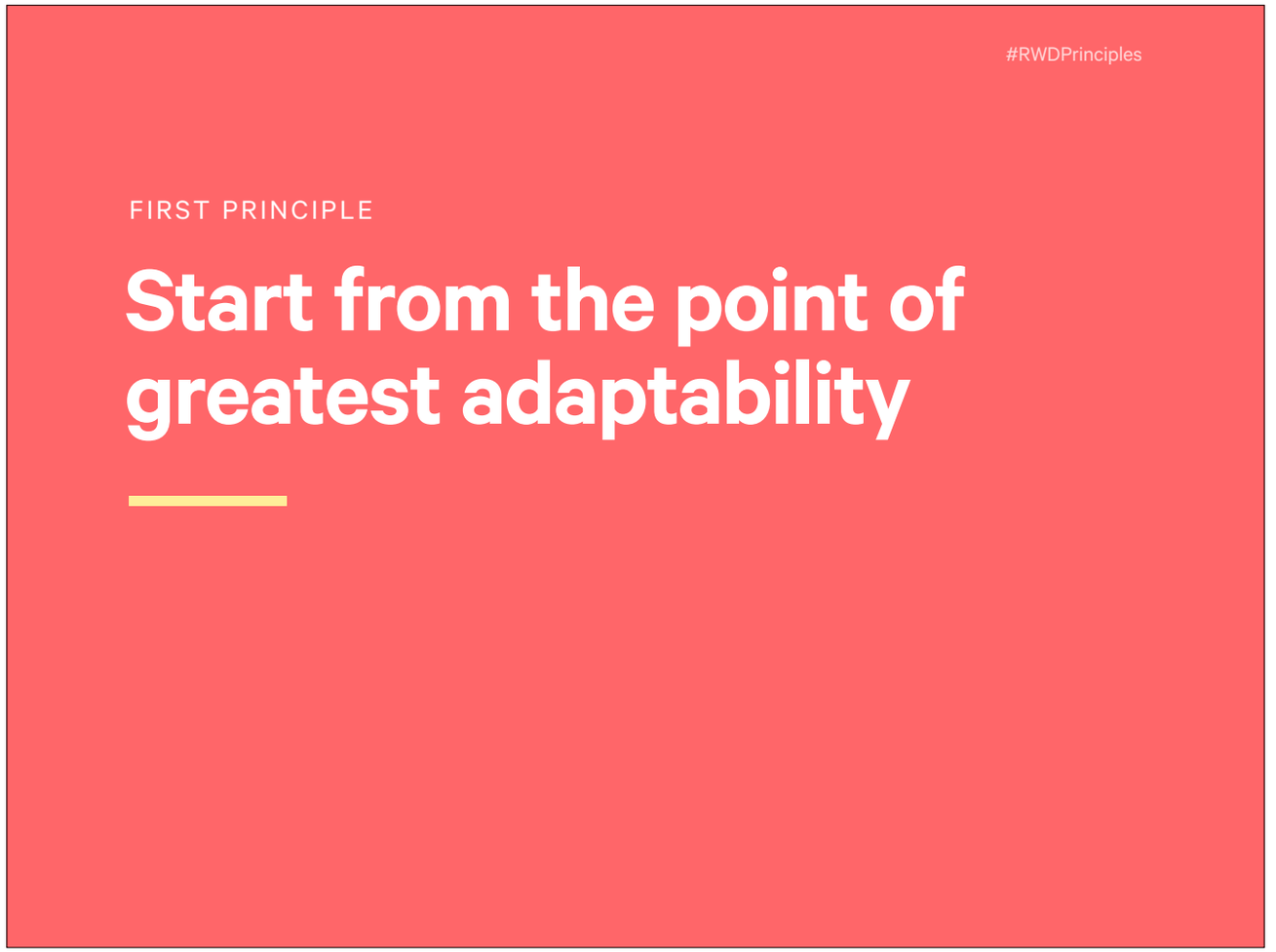
Any principles we create need to be in service of this grander vision.

Tonight, I will present three principles that address the following:

- The nature of the medium
- The desires of those who use it
- And finally, the needs of those that build it, us.

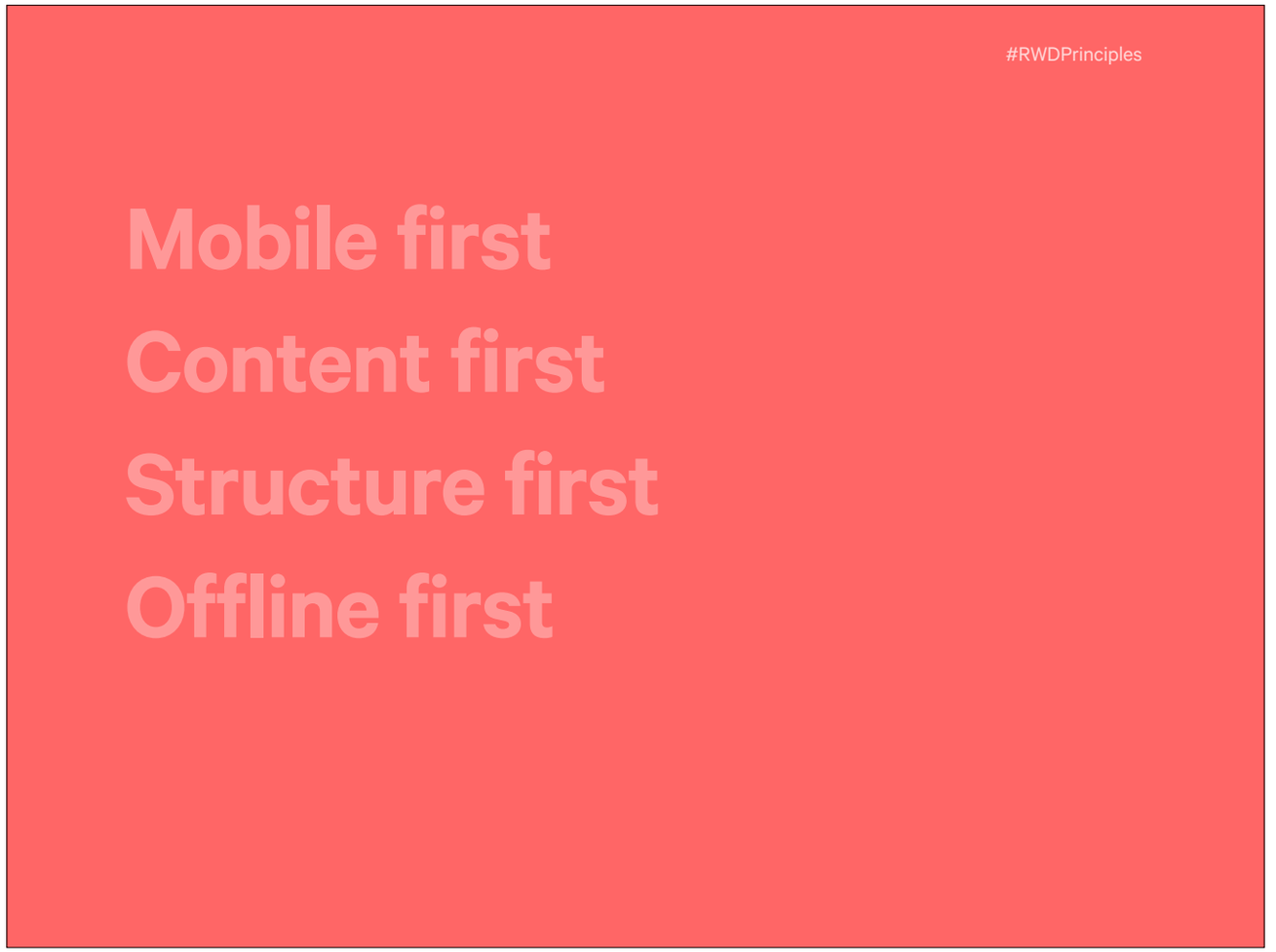
To judge their suitability, I will reflect on the discussions that have taken place over the last five years, and support each principle with an example or two from my own experience designing responsive sites.

So, let's begin.



Our first principle:

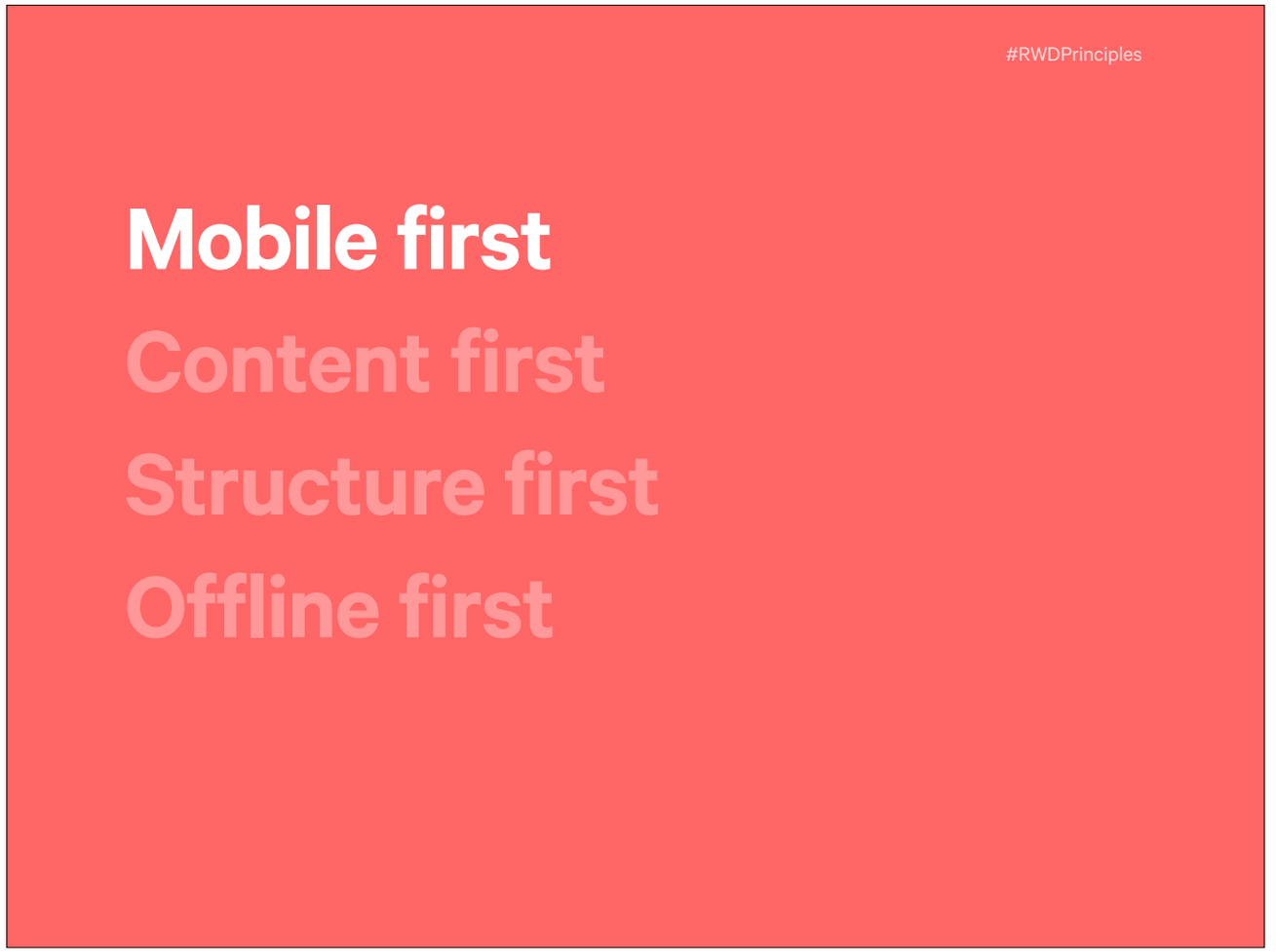
Start from the point of greatest adaptability



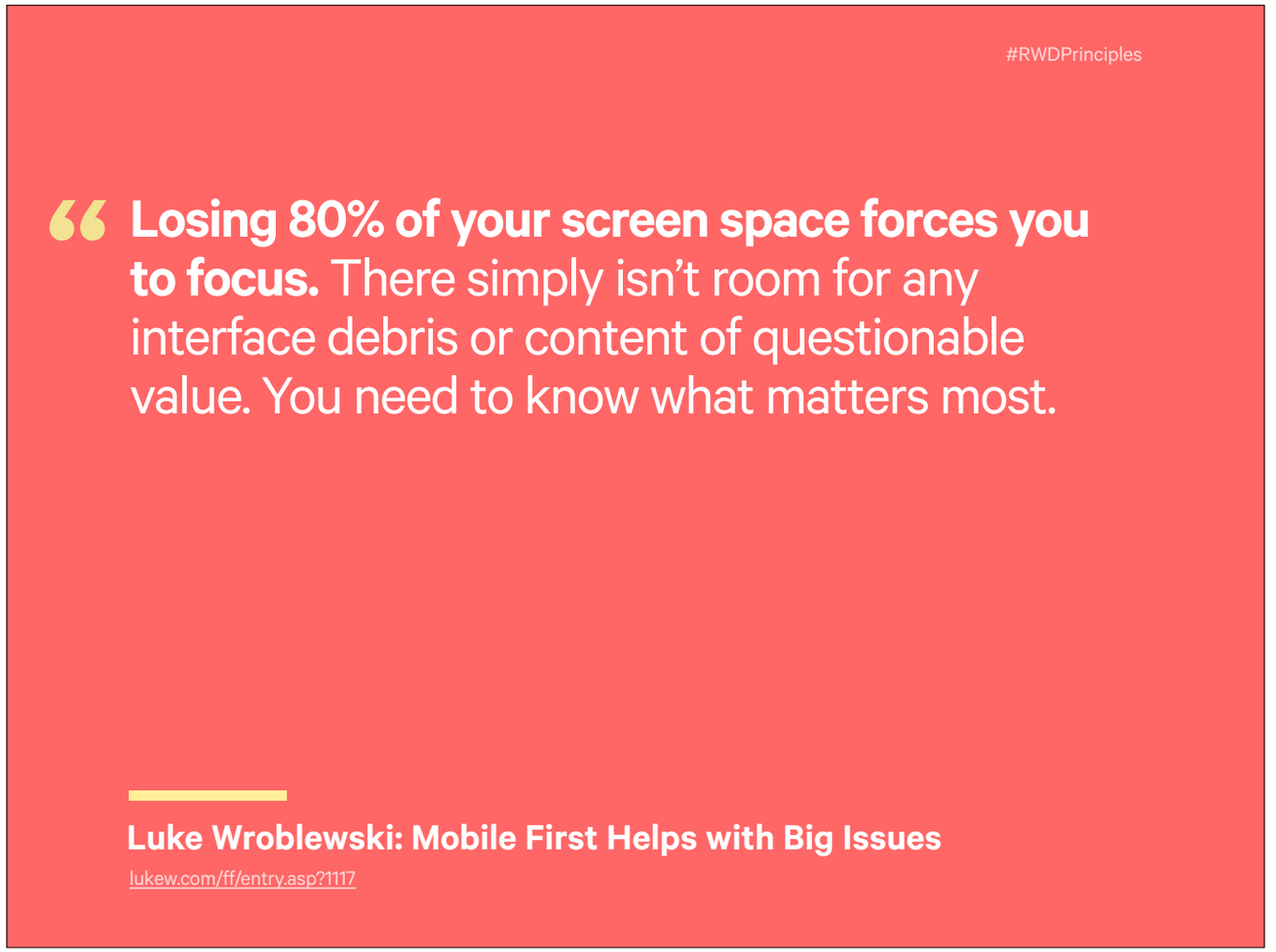
Multiple avenues can be explored when embarking on a product that can be widely accessed:

- Context of use
- Suitability of content
- Device capabilities
- …

It’s no surprise where we even begin to start, has confounded opinion.



In 2009, Luke Wroblewski put forward the case for designing websites mobile first; that is, considering the capabilities and features of mobile phones, before considering how an experience may manifest itself within a desktop browser.

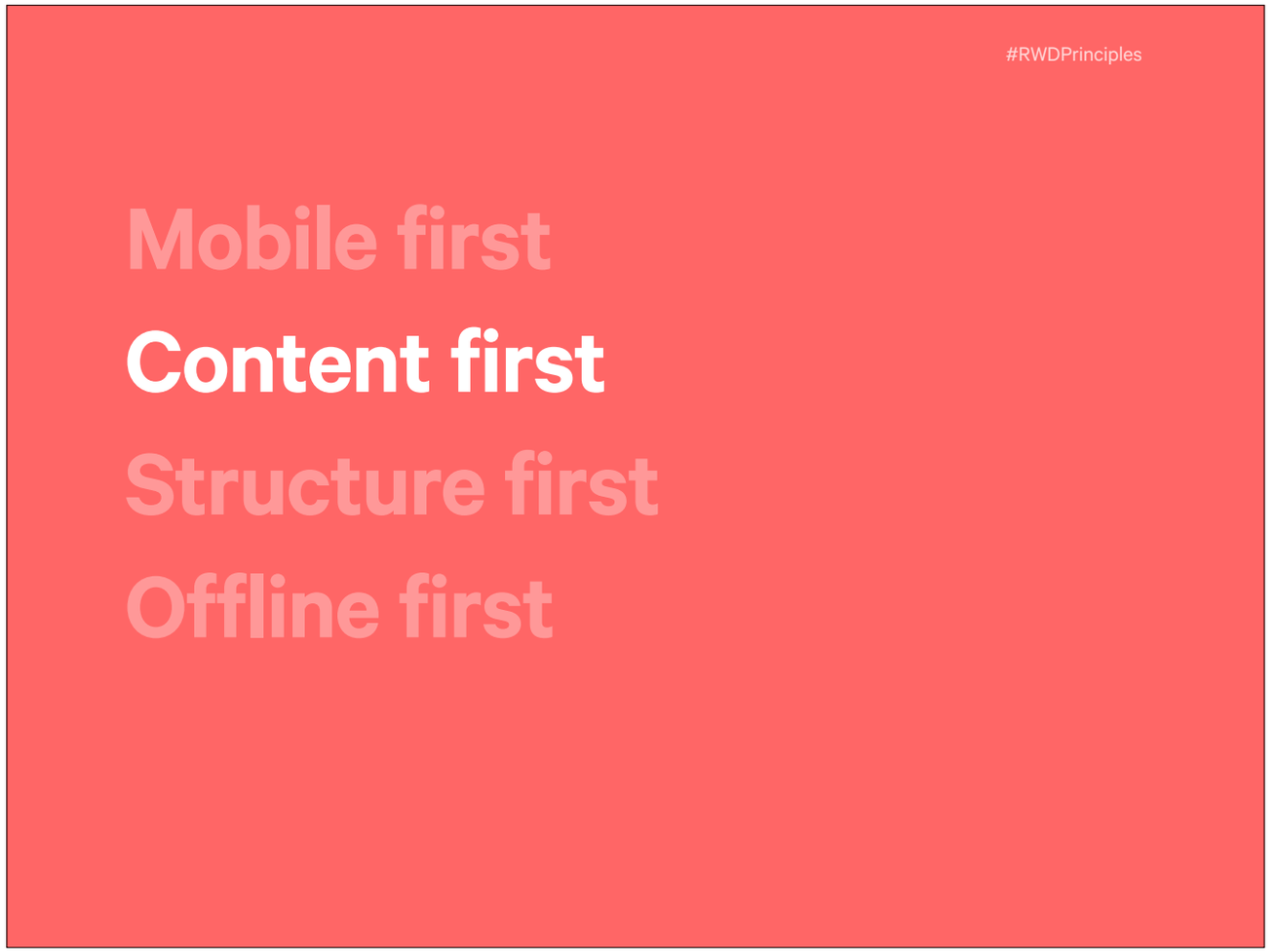


This approach was partly driven by the rapid growth of mobile, but it also had a side benefit. Luke stated that:

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.

A mobile first approach often requires prioritisation exercises and making decisions about which parts of an interface are critical and which are not.

These non-critical parts can be loaded later, perhaps conditionally — or not at all.

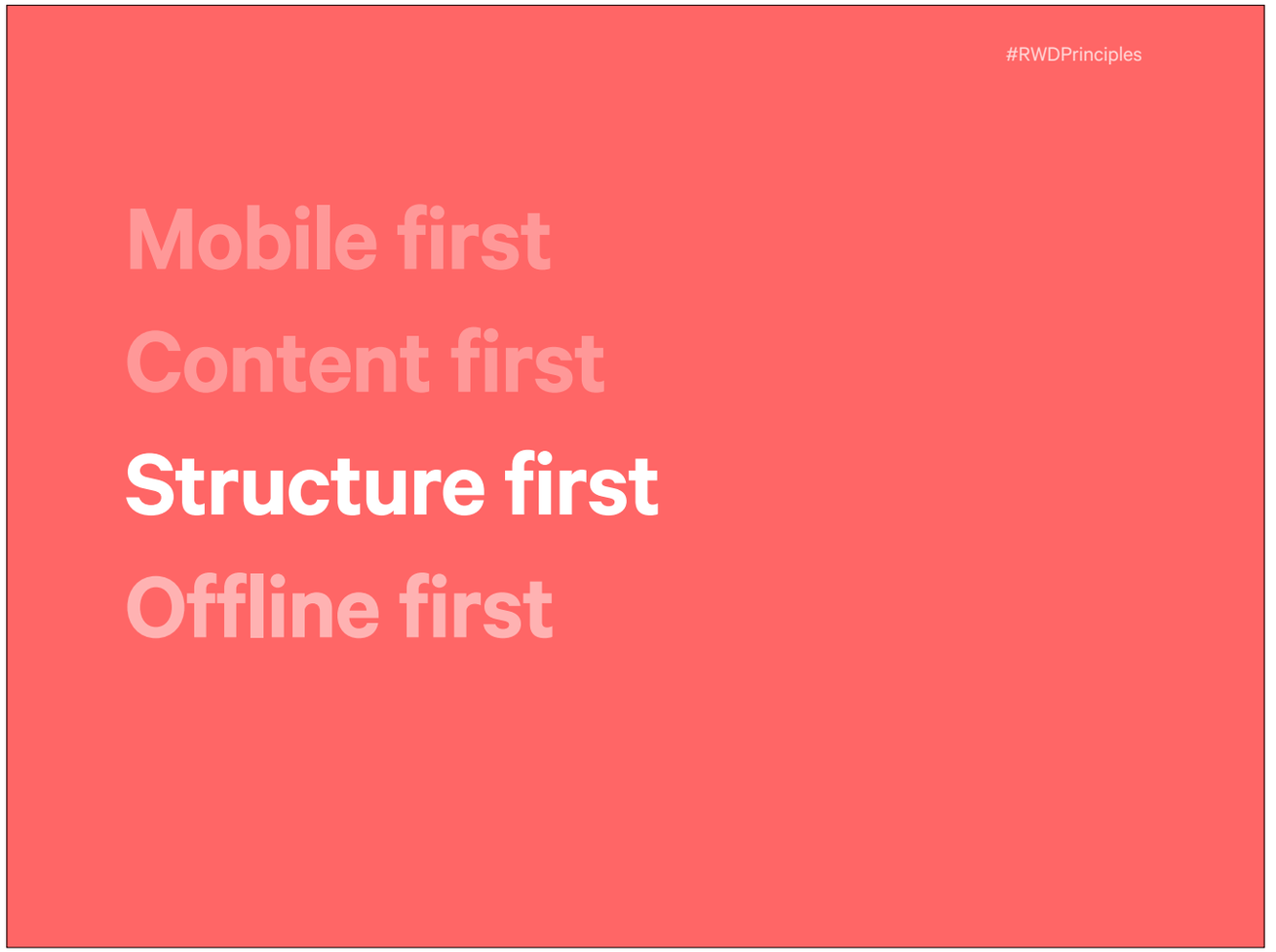


By dissecting an interface in this way, we often see content being prioritised above everything else.

For years we repeated the mantra ‘content is king’, before drawing rectangles and pouring in placeholder text.

With much of the interface governed by available screen real estate, we’ve become increasingly reliant upon content, given its relative consistency.

Content is no longer a commodity, but a prerequisite.

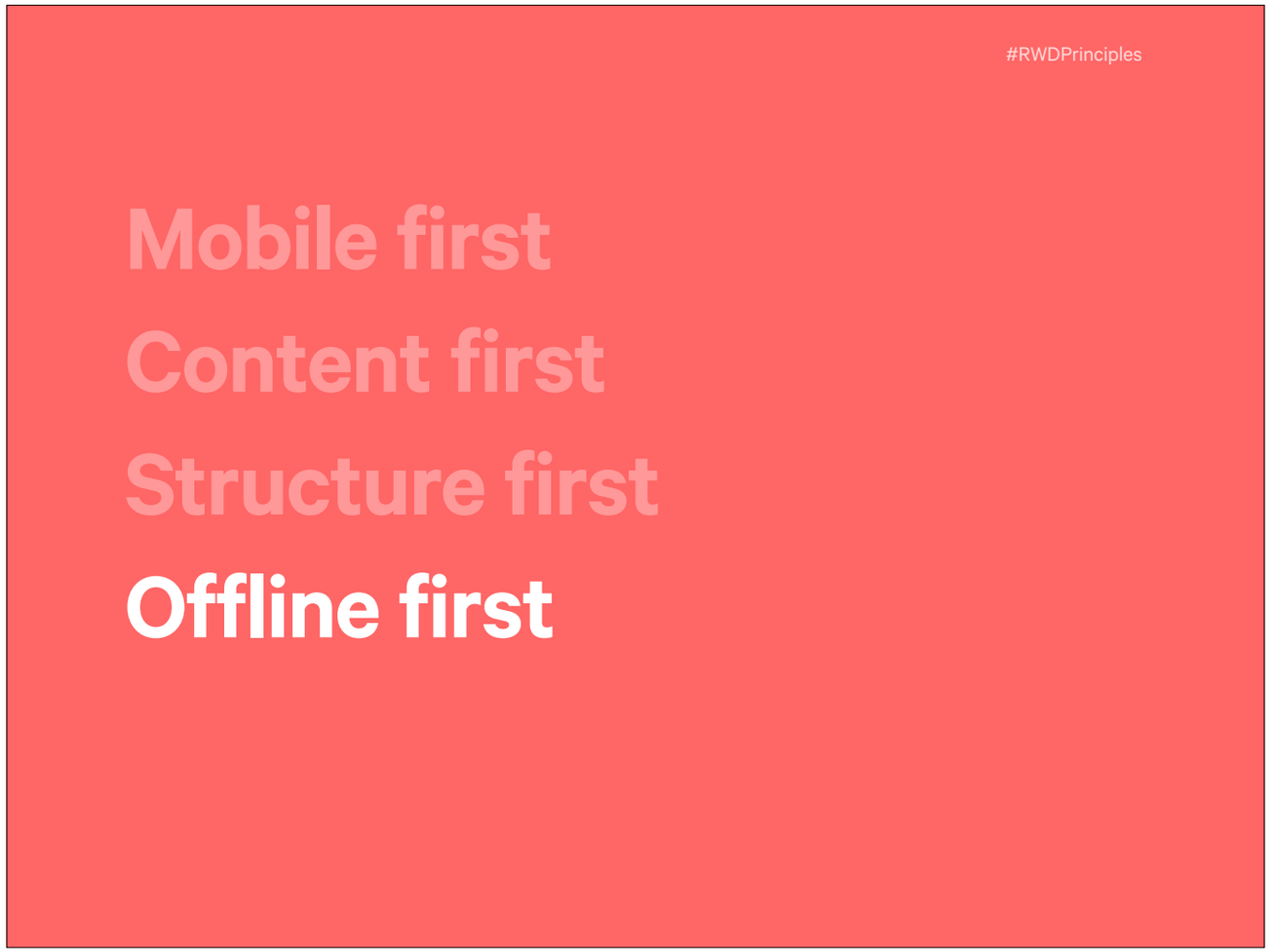


Mark Boulton suggested we “design from the content out, rather than the canvas in”.

Having content in place before we design a page, is unlikely. Mark later clarified his position, suggesting we consider “structure first, content always”.

By considering the type of content we need – tasks it needs to perform, information that should be conveyed – we can start by using content representative of the final form.

At the same time, we can help clients think about their more detailed requirements.



More recently there has been a move to think about offline first, taking into account the fact that a connection to the Internet is not always present (or reliable).

Again, this is a perspective which considers the nature of the medium and tries to design for the resulting scenarios.

What does ‘mobile’ mean?

While a mobile-first approach is useful, the definition of mobile has grown increasingly vague.

Luke’s article was introduced when phone displays were small and the market still developing.

Today, manufacturers compete by offering bigger and higher-density displays, while the speed of the processors that power them increases.

Another tenant of mobile-first was thinking about the capabilities mobile devices have; location, multi-touch, cameras and other sensors.

As a proxy for constraint, mobile has been spoiling us.

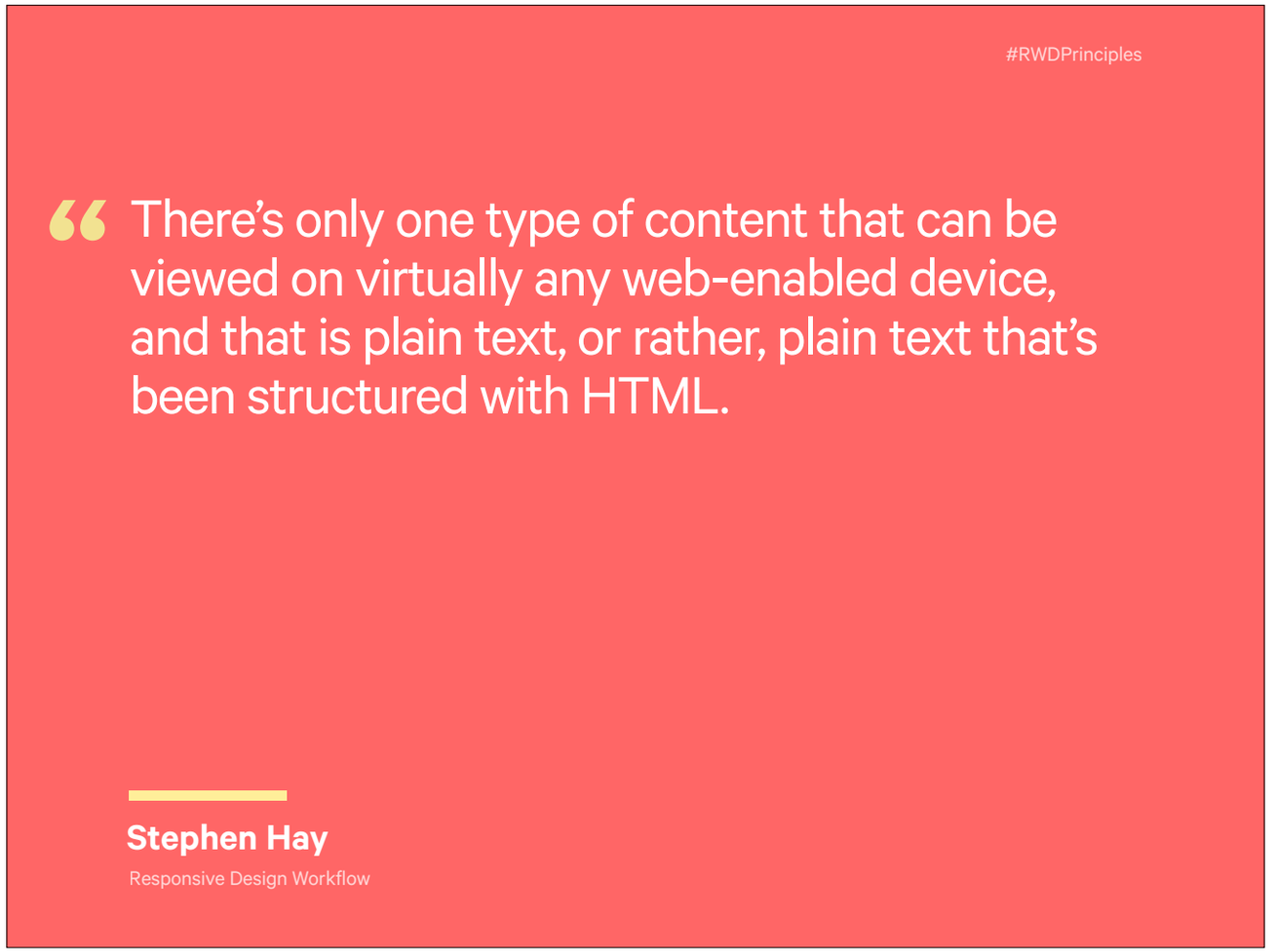
Watch first?

It might be tempting to suggest that we should now think ‘watch first’. In terms of screen size at least, this would reset that constraint.

Thinking about screens makes the assumption that the devices consuming our will always include a display, or that visual affordances alone will suffice.

That’s not always be true; it certainly isn’t how search engine spiders experience the web, let alone accessibility software.

To build truly universal and accessible websites, we need to look beyond visual design.



I often design an interface by thinking about the structure of a page in plain text first. This is an approach advocated Stephen Hay:

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.

When you consider other media types we use on the web can also be represented by text — images can include descriptive alt text, videos can provide transcripts — thinking this way can help us build products that are accessible from the very start.

Example

Conversation first

I've started to take this approach a bit further, by thinking about how an interface might be read out as part of a conversation.

What questions might a user be asking? How might I structure a response?

Let's imagine an article reviewing a film.

A key piece of information here 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 the 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>
```

We could mark this up as follows —this markup is actually taken from the website of a UK newspaper.

This markup is based entirely on the visual representation, making an assumption about the capabilities of the device consuming this content.

Unsurprisingly, it’s not very accessible.

“How is this movie rated?”

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

What if we were to think of the page as a conversation instead?

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

Our marked up answer would then be as follows.

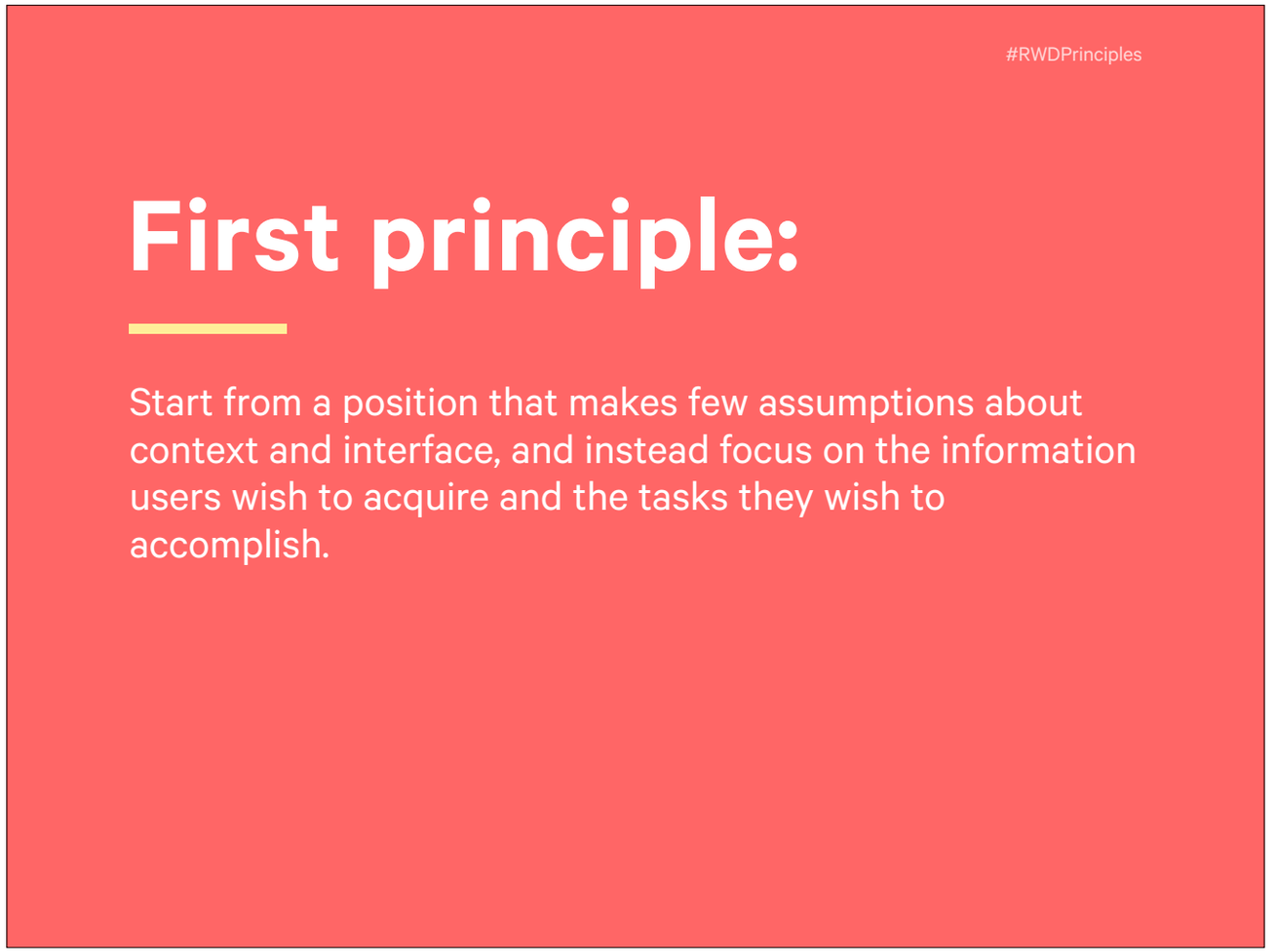
“How is this movie rated?”

```
<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 allows us to add some visual embellishment.

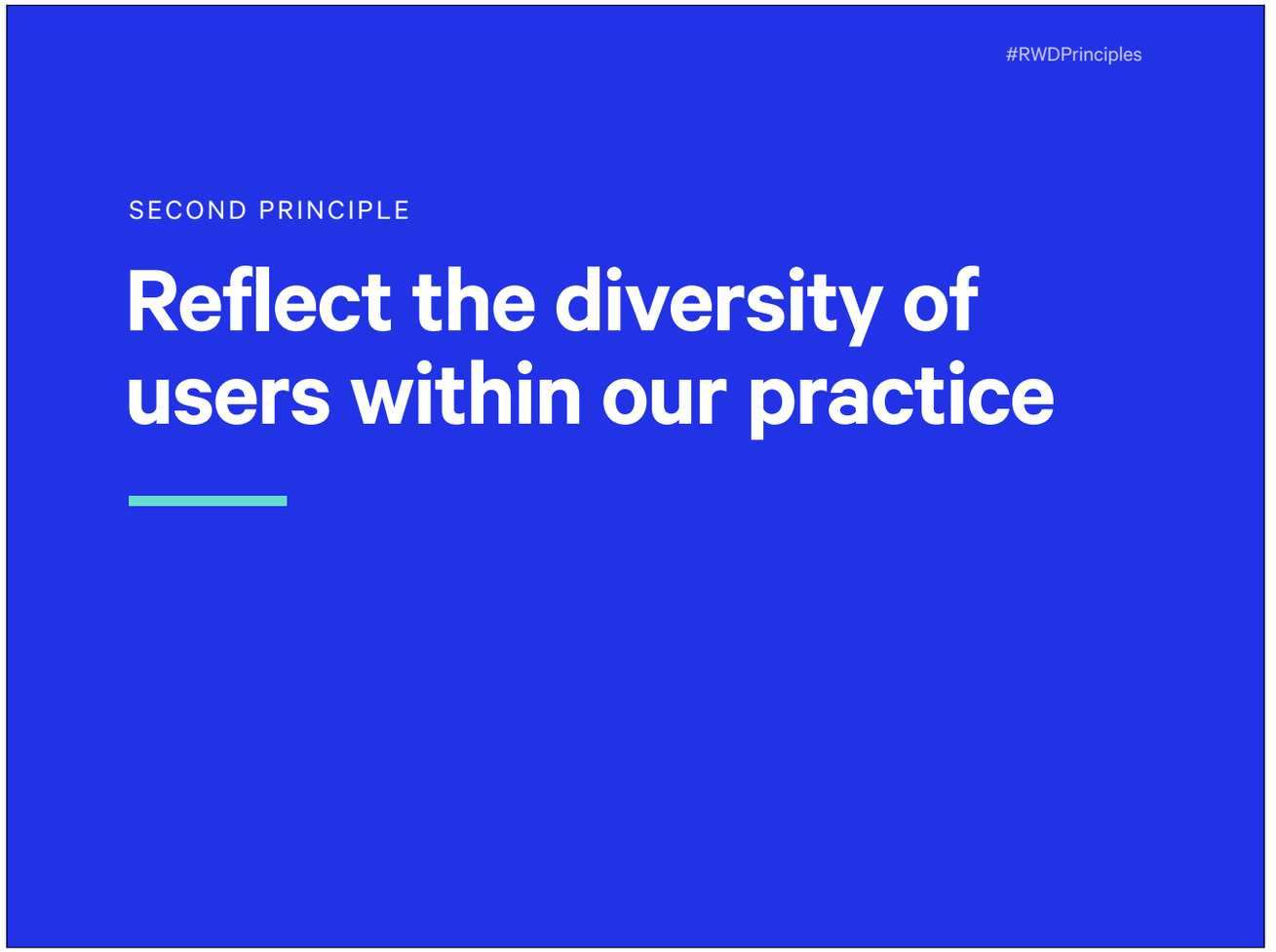
Here, empty spans are used for styling hooks (which we might inject later with JavaScript), with ARIA attributes to ensure these are not parsed by accessibility software.

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



Our first principle can be summed up as follows:

Start from a position that makes few assumptions about context and interface, and instead focus on the information users wish to acquire and the tasks they wish to accomplish.



It’s worth remembering that device fragmentation is a reflection of human diversity, consumers exercising their right to choose. And this diversity is the concern of our second responsive principle:

Reflect the diversity of users within our practice

Many of us are lucky to sit at large, high-resolution displays with new laptops and a broadband connection. This is hardly representative, and introduces an empathy gap between ourselves those using the products we're building.

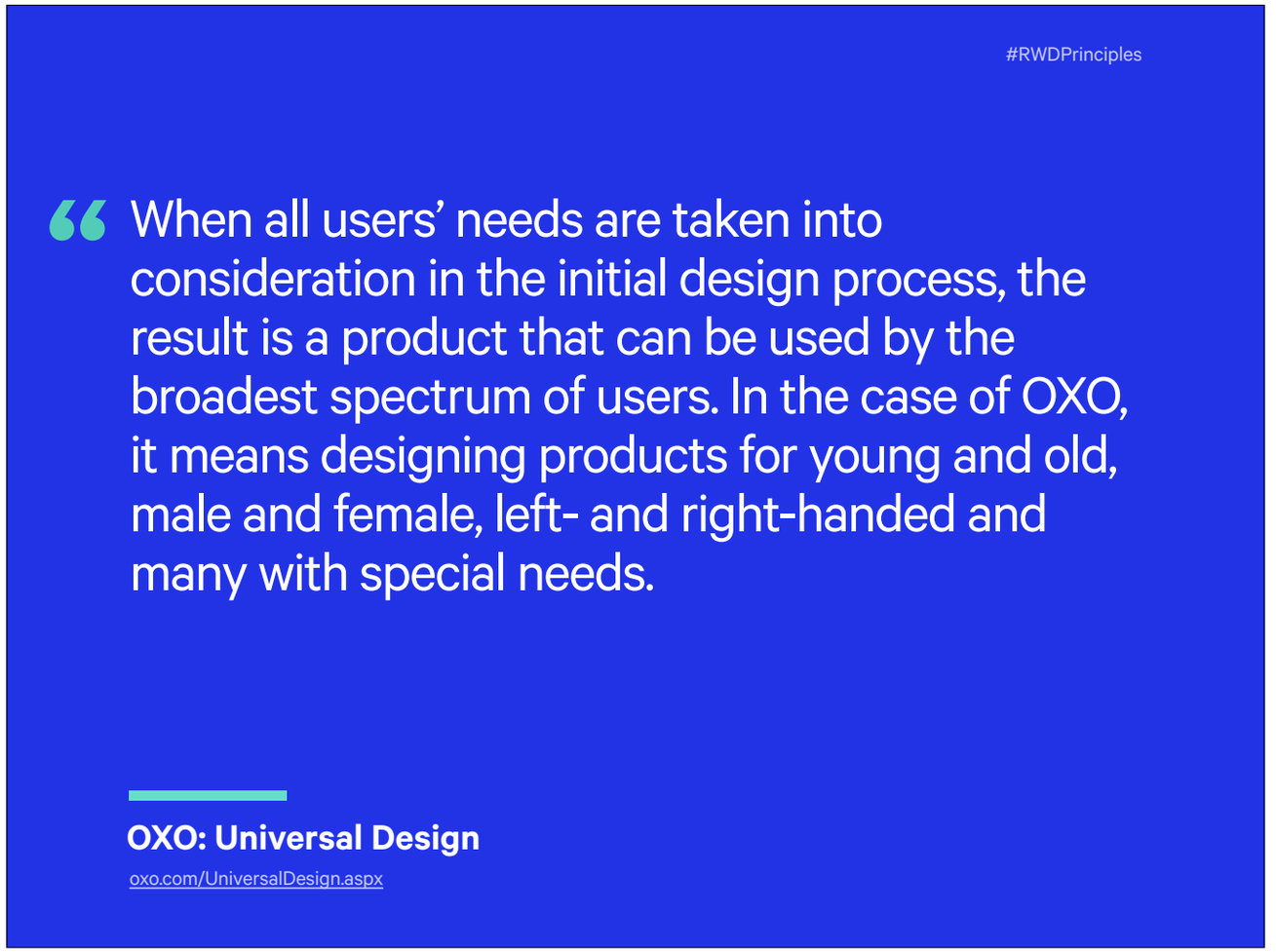
If we accept that the goal of the web is universal access, it follows that we should be thinking about universal design.

This is a practice with greater recognition within architecture and product design, but I think it can apply to products built with pixels as well.

This is a vegetable peeler, designed by Sam Farber.

A veteran of the kitchenware business, he spotted an opportunity when he saw that his wife, Betsey, an arthritis sufferer, struggled to use a conventional metal vegetable peeler.

So he created one with a more comfortable grip. Turns out, those without arthritis enjoyed using this peeler as well.



He went on to found OXO, a consumer goods company founded on the principle of universal design:

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. In the case of OXO, it means designing products for young and old, male and female, left- and right-handed and many with special needs.

When you look at the principles of universal design, they include recommendations like "eliminate unnecessary complexity", "minimise repetitive actions" and "arrange information consistent with its importance".

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

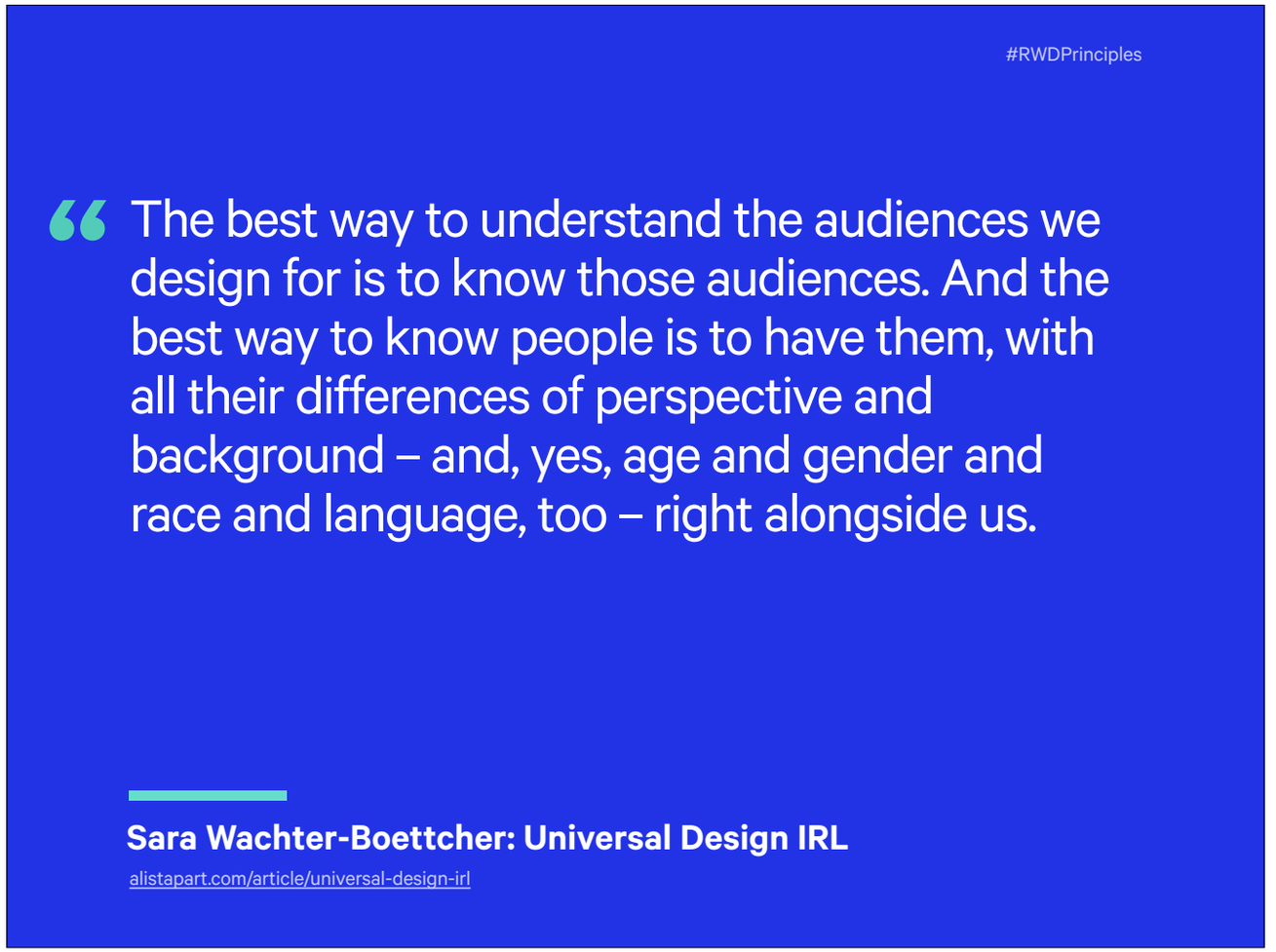
Collaborative practice

By running user centred design activities like user research and usability testing, we can learn how users interact with our products, and better understand their needs.

Working closely with clients, we can not only learn about their requirements, but also uncover any insight they've gleaned about their customers, too.

Including different roles and disciplines throughout the design process helps us build a shared understanding of the problems being solved.

Having collective responsibility for aspects like responsiveness, performance and accessibility, means the overall quality of the product is likely to be improved, too.

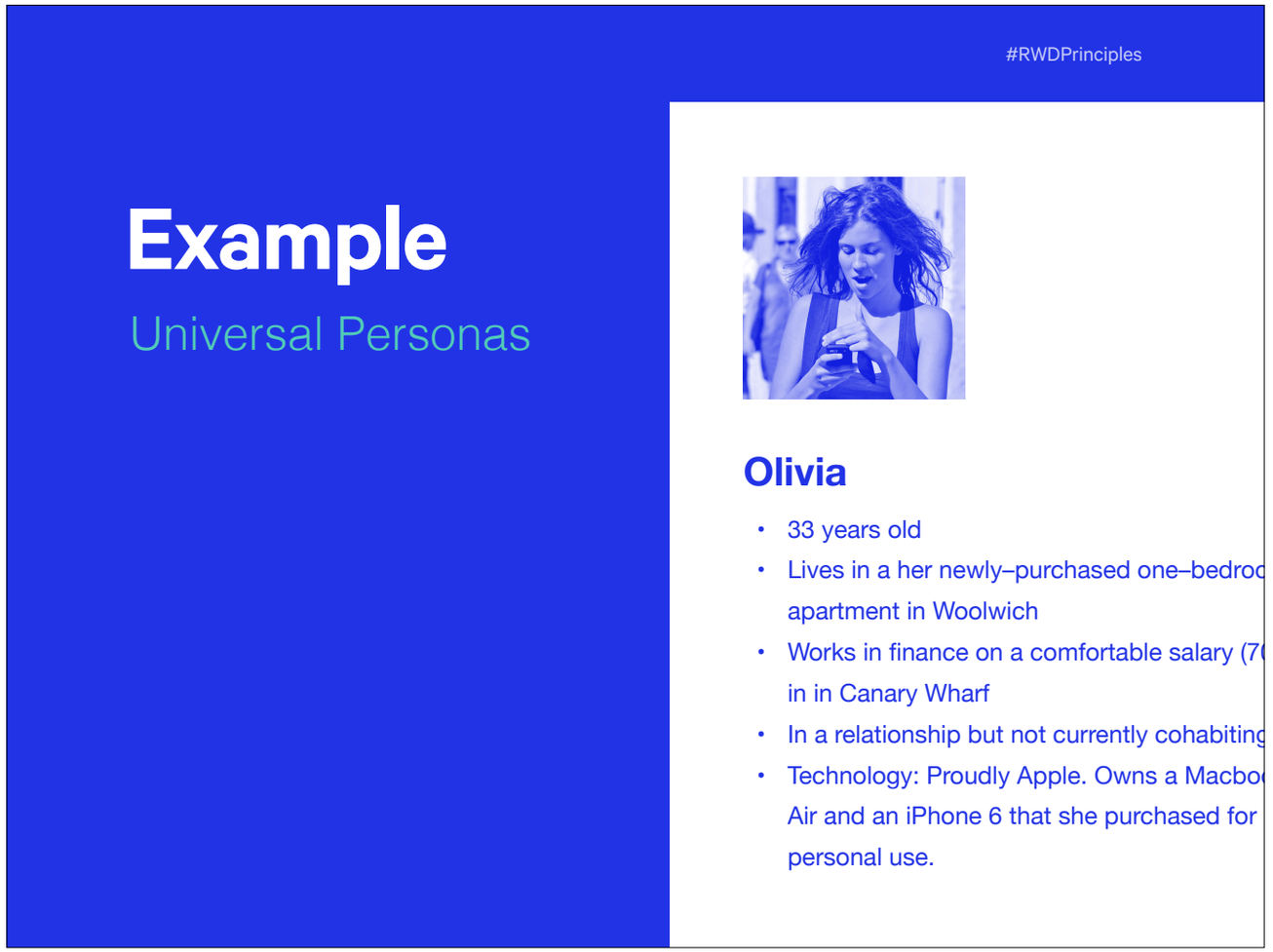


There’s another aspect effective, empathetic teams that is often overlooked. In her article Universal Design IRL, Sara Wachter-Boettcher states:

The 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.

Our community has become increasingly concerned about the diversity; the people we see, hear and work with. Once your eyes have been opened to its monoculture, our white, male dominated industry becomes hard to ignore.

Fighting to change this is a worthy and necessary goal.



How often do we create personas that include accessibility aspects?

I was reminded of this on a recent project. Even though we acknowledged early on that a large number of middle aged users with poor eyesight used the website, we made no consideration for that in our personas.

This raised a discussion about how we could factor accessibility issues into this tool, which formed the basis of the design exercises that followed.

It might be tempting to create a separate persona, one that includes a number of disabled attributes, but that wouldn't be realistic, and I suspect easy to disregard; personas need to be grounded in reality, after all.

#RWDPrinciples

Peter

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

Reduced motor ability

Olivia

...often uses her phone when walking to work

Poor eyesight

Sanjita

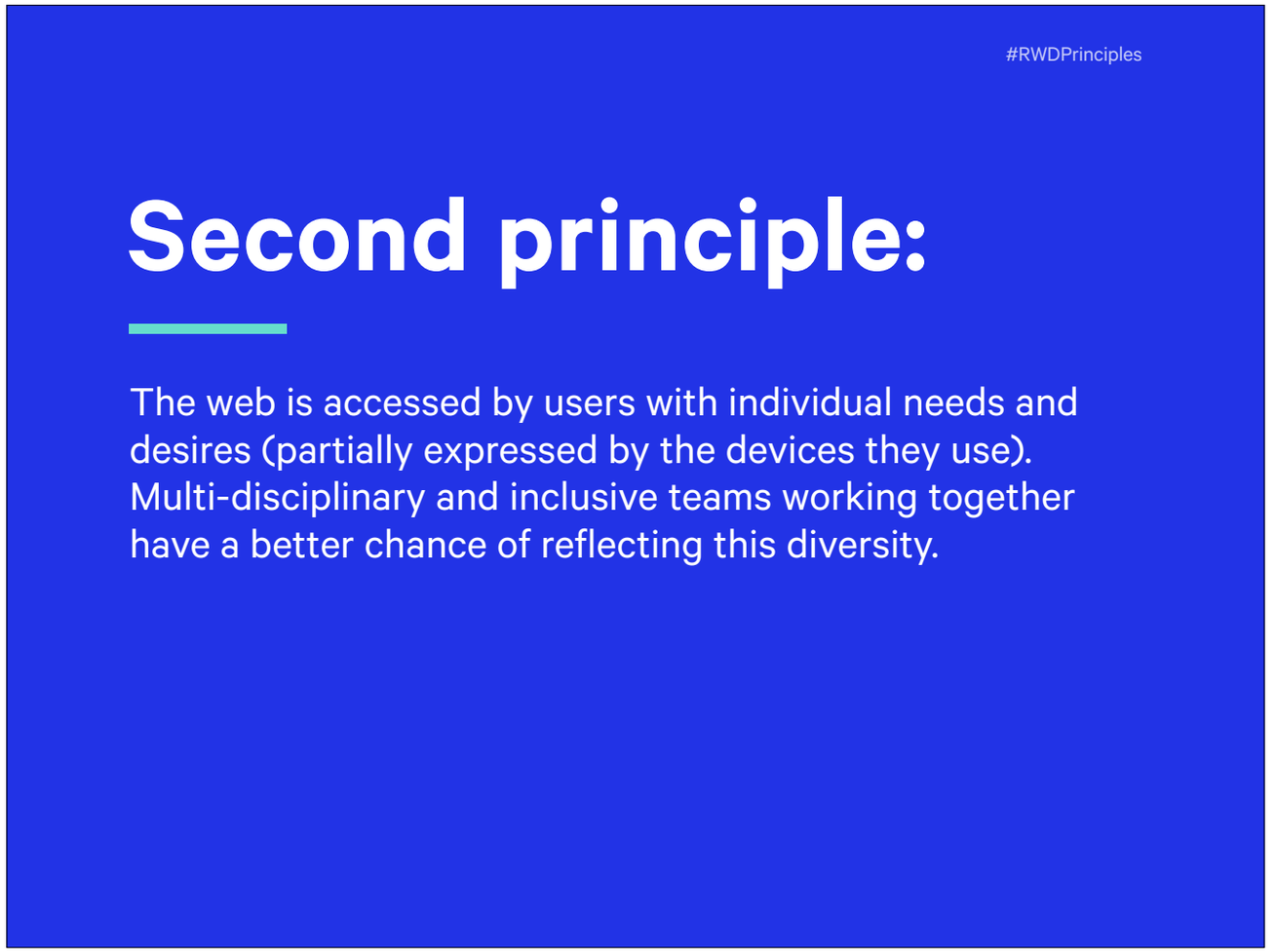
...has two children who play musical instruments

Hearing loss

flickr.com/photos/mikelo/3139837006 flickr.com/photos/ktoine/6263683606 actiononhearingloss.org.uk

Personas needn't be deliberately disabled; remember anyone can exhibit the characteristics of a disabled person from time to time.

- A persona could have recently broken their wrist in a skiing accident, meaning they have temporary restriction in that area.
- Another could often use her phone in bright sunlight, and thus display the characteristics of poor eyesight.
- Another could have hearing loss, but not because of a medical condition, but because they live in a loud household.



Our second principle therefore is as follows:

The web is accessed by users with individual needs and desires (partially expressed by the devices they use). Multi-disciplinary and inclusive teams working together have a better chance of reflecting this diversity.

#RWDPrinciples

THIRD PRINCIPLE

Build using systems that can be reasoned with

Our third principle concerns itself with building systems that can be reasoned with. Why is this important?

Automation

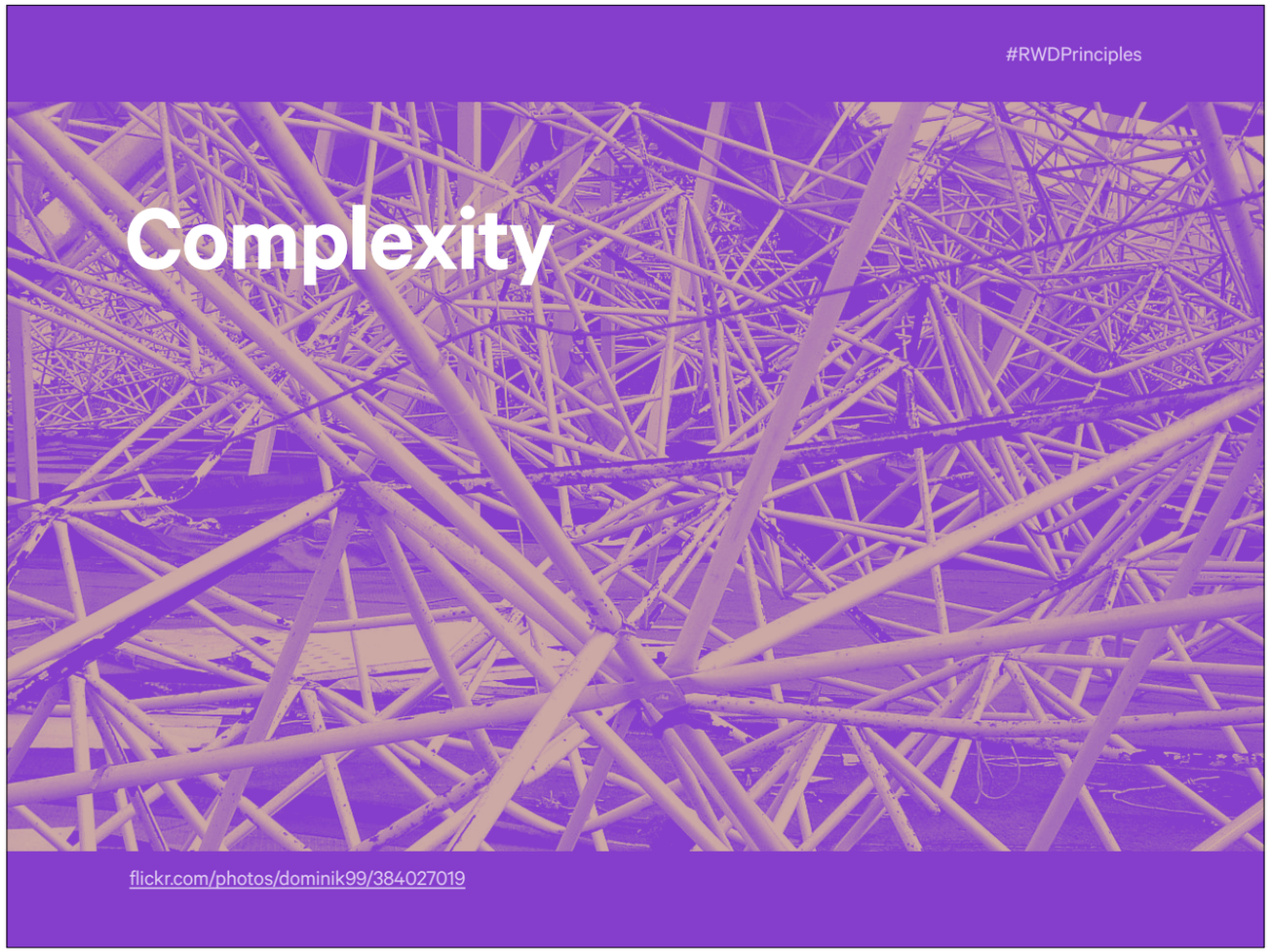
flickr.com/photos/spenceyc/7481166880/

Once upon a time, all you needed to build a website was a text editor and a working knowledge of HTML and CSS.

Now you need to understand a multitude of other tools and concepts.

We have seen a growing use of automation within the field of front-end development, with pre-processors, task runners, build tools and testing frameworks now integral to much of our work.

Some view this as maturity, others as madness.



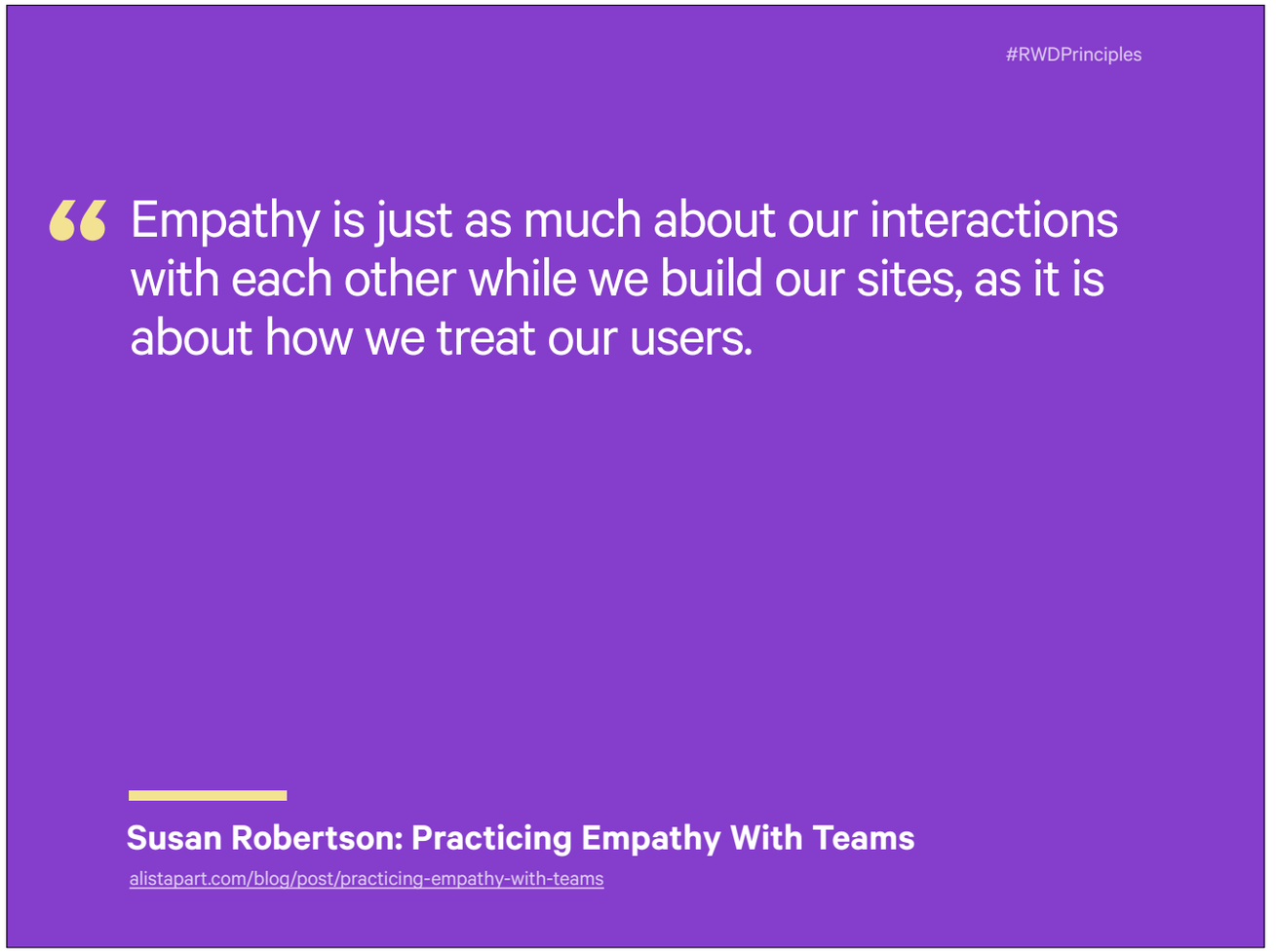
I’m broadly comfortable with this trend towards engineering and away from previous notions of craft.

Whether we’ve become too reliant on these tools is open to debate.

Right now we’re exploring the opportunities these tools bring:

- Pre-processors allows us to modularise our CSS and use features like constants.
- Testing code for errors helps us improve the robustness of products.
- Optimising assets makes these products faster.

These are all worthy goals, but they have a cost: complexity.



Susan Robertson recently wrote about working with other people in her team, noting that:

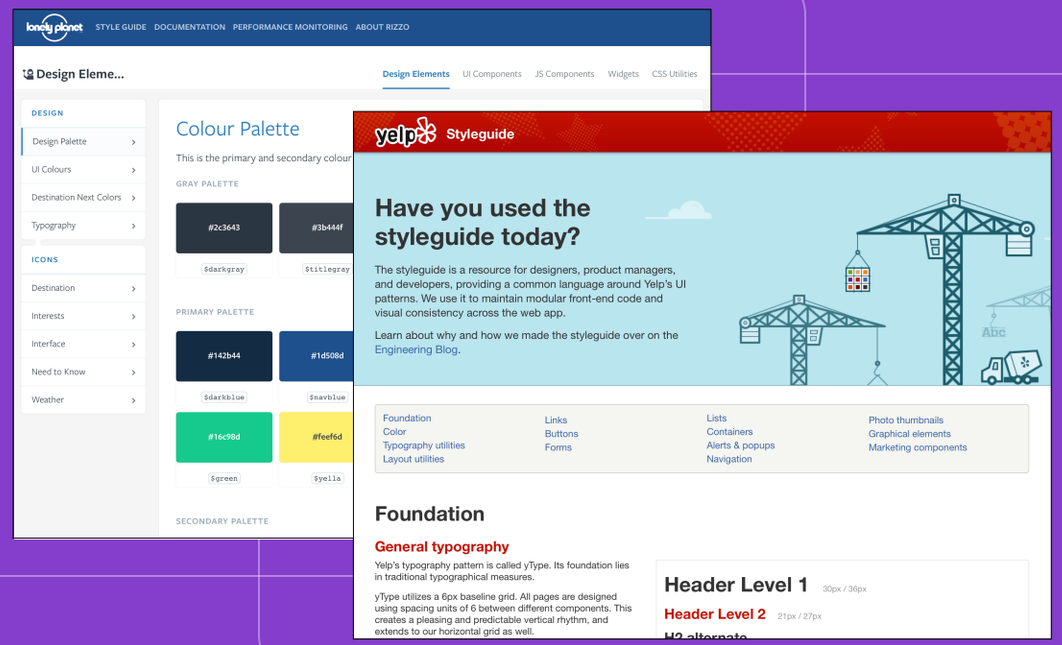
Empathy is just as much about our interactions with each other while we build our sites, as it is about how we treat our users.

Our last principle concerned itself with practicing empathy with users.

This principle follows on from this, accept in this case, the user is another member of our team.

Automated workflows are useful, but we should be careful not to leave our colleagues behind.

#RWDPrinciples



Styleguides for Lonely Planet and Yelp

styleguides.io/examples.html

Designers need to make their work more accessible to developers too.

A practice gaining momentum is the creation of visual design languages documented with style guides.

I find this development exciting, as it reflects the maturing relationship between designer and developer, working as one to build systems that each can understand.

It's also an acceptance, that designing a static pages is no longer an effective or helpful practice. Instead we should design components that work as part of a broader system.

Object-Oriented CSS

flickr.com/photos/13403905@N03/2080281038

Thinking in components is a widely excepted software engineering practice.

Tim Berners Lee considered “Modular Design” and “Being part of a Modular Design” as necessary principles of good design.

Object-oriented design principles have been advocated for many years, especially by people like Nicole Sullivan and Harry Roberts.

We can debate the pros and cons of an object-oriented approach to CSS (indeed I’ve found myself on both sides of the argument).

I’ve grown to like it because it encourages modularity, reuse and systems thinking.

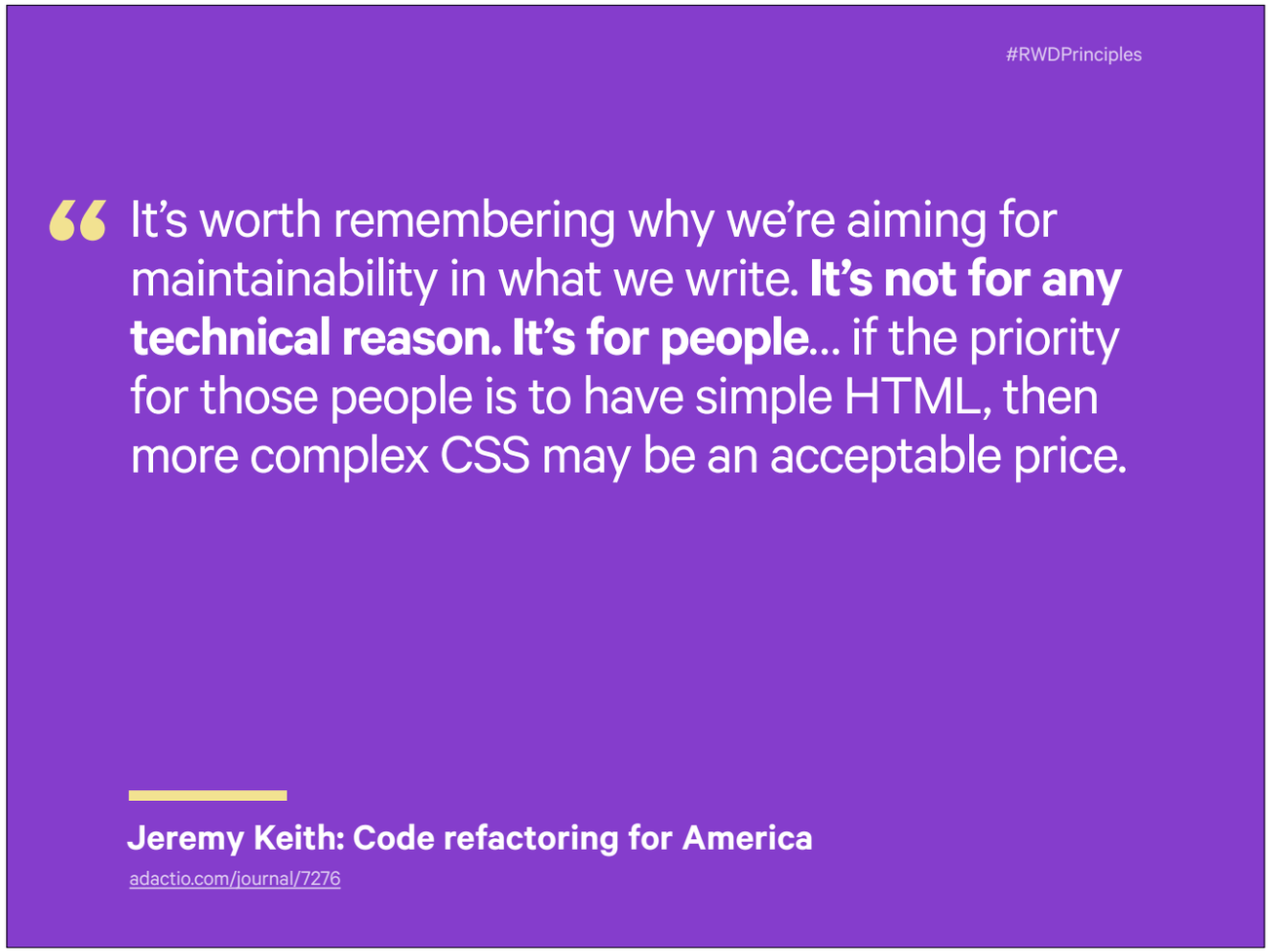
Pragmatic, not dogmatic

It’s easy to get dogmatic about this things, but it needn’t be so black and white.

A lot of people I’ve spoken to don’t like BEM because they get caught up on double hyphens and underscores that some flavours of BEM use.

Understand the approach before you get caught up in by a particular implementation; after all, you can always adapt it to your own tastes.

As always, try to favour pragmatism over any theoretical purity.



Last year, Jeremy Keith talked about refactoring the code he and Anna Debenham had developed for Code for America.

Originally used an object orientated approach, with HTML that made heavy use of classes. These components were being used by Code for America volunteers who had a limited understanding of HTML and CSS. In many cases, adding complexity to the HTML was hindering their ability to create pages.

Jeremy and Anna therefore decided to refactor these components; while this meant their CSS was more complex, it allowed for simpler HTML. Jeremy wrote:

It’s worth remembering why we’re aiming for maintainability in what we write. It’s not for any technical reason. It’s for people... if the priority for those people is to have simple HTML, then more complex CSS may be an acceptable price.

It depends

I believe thinking in more modular terms actually allows us to invoke ‘it depends’ more liberally than we would otherwise.

We needn't get into arguments about whether OOCSS is the right approach or not. Or BEM versus embracing the cascading nature of CSS.

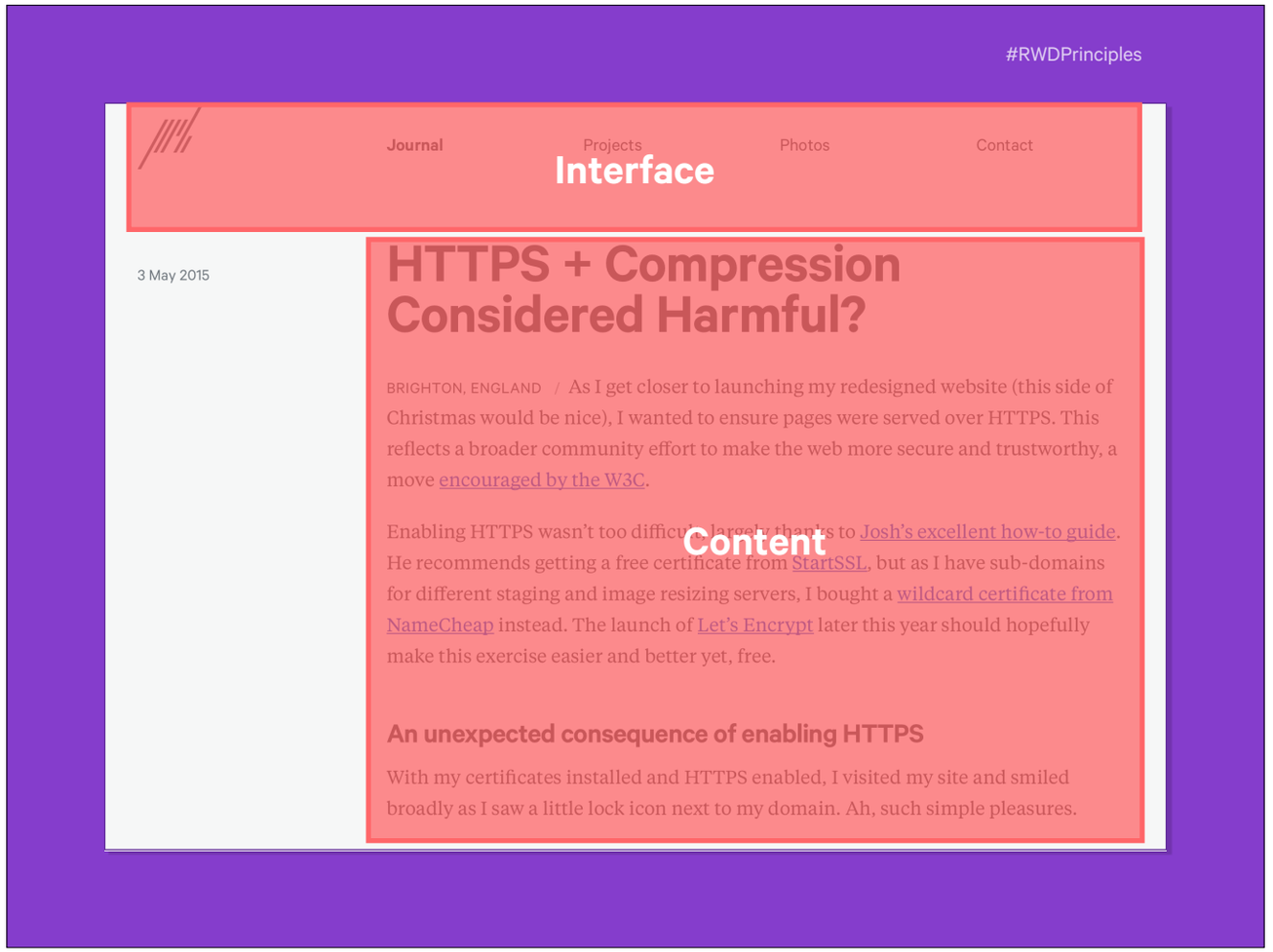
If each component in a system is designed to be truly self-contained, surely that frees us up to use whatever approach is most appropriate for a given component?

Example

Interface or content?

We can look at each component individually, and think about its role, its purpose.

Does it convey information (it has more document-like attributes). Is it used to perform a task (it has more application-like attributes)?



Interface elements are likely to remain relatively unchanged, their design and content determined by those who build the product.

Content on the other hand, may be input via a CMS, come from a third-party, and generally be more fluid and unpredictable.

Again, this is a very simple example, but it should give you an idea.

```
<nav class="nav  clearfix" role="navigation">
    <ul class="nav-list list">
        <li class="nav-list__item  list__item">
            <a class="nav-list__label" href="#">Journal</a>
        </li>
        <li class="nav-list__item  list__item">
            <a class="nav-list__label" href="#">Projects</a>
        </li>
        <li class="nav-list__item  list__item">
            <a class="nav-list__label" href="#">Photos</a>
        </li>
        <li class="nav-list__item  list__item">
            <a class="nav-list__label" href="#">Contact</a>
        </li>
    </ul>
</nav>
```

The navigation may exist as part of a broader system of UI components, so we can use a more object orientated approach with a BEM naming method.

```
<div class="prose">
    <h1>HTTPS + Compression Considered Harmful?</h1>
    <p>As I get closer to launching my redesigned website (this side…</p>
    <p>Enabling HTTPS wasn’t too difficult, largely thanks to Josh’s…</p>
    <h2>An unexpected consequence of enabling HTTPS</h2>
    <p>More precisely, a site is vulnerable to this attack when pages:</p>
    <ul>
        <li>Are served from a server that uses HTTP-level compression</li>
        <li>Reflect user-input in HTTP response bodies</li>
        <li>Reflect a secret in HTTP response bodies</li>
    </ul>
</div>
```

However, if a component contains content, perhaps output from a CMS or API, using the cascade may be more appropriate – not least because we can't add any classes to the marked-up content.

```
/* Clear floats */
.clearfix { }

/* Interface lists */
.list { }
.list__item { }

/* Navigation component */
.nav { }

/* Navigation list component */
.nav-list { }
.nav-list__item { }
.nav-list__label { }
```

```
/* HTML within prose */
.prose h1 { }
.prose h2 { }
.prose p + h2 { }
.prose ul > li { }
```

This navigation component has it's own styling, but also uses common utilities like `clearfix`, as well as more abstracted component styles, like `list` and `list__item` which declare the characteristics of interface lists.

Inside the content component, we can go crazy, and use all kinds of crazy selectors, a defensive approach often needed when dealing with the unpredictable nature of content.

How do you know which approach to use, when?

For a system to be maintainable, robust and error-free, anyone contributing to it should understand the consequences of changing an aspect of it.

Code is read more often than it is changed, so it should be documented, or better still, self-documented, when ever possible.

So while different CSS methodologies within the same system affords us greater flexibility, it also introduces a degree of fragility.

How do you know which approach is being used, when?

> A namespace will tell us exactly how classes behave in a more global sense. A namespace tells us exactly what a class (or suite of classes) does in non-relative terms.

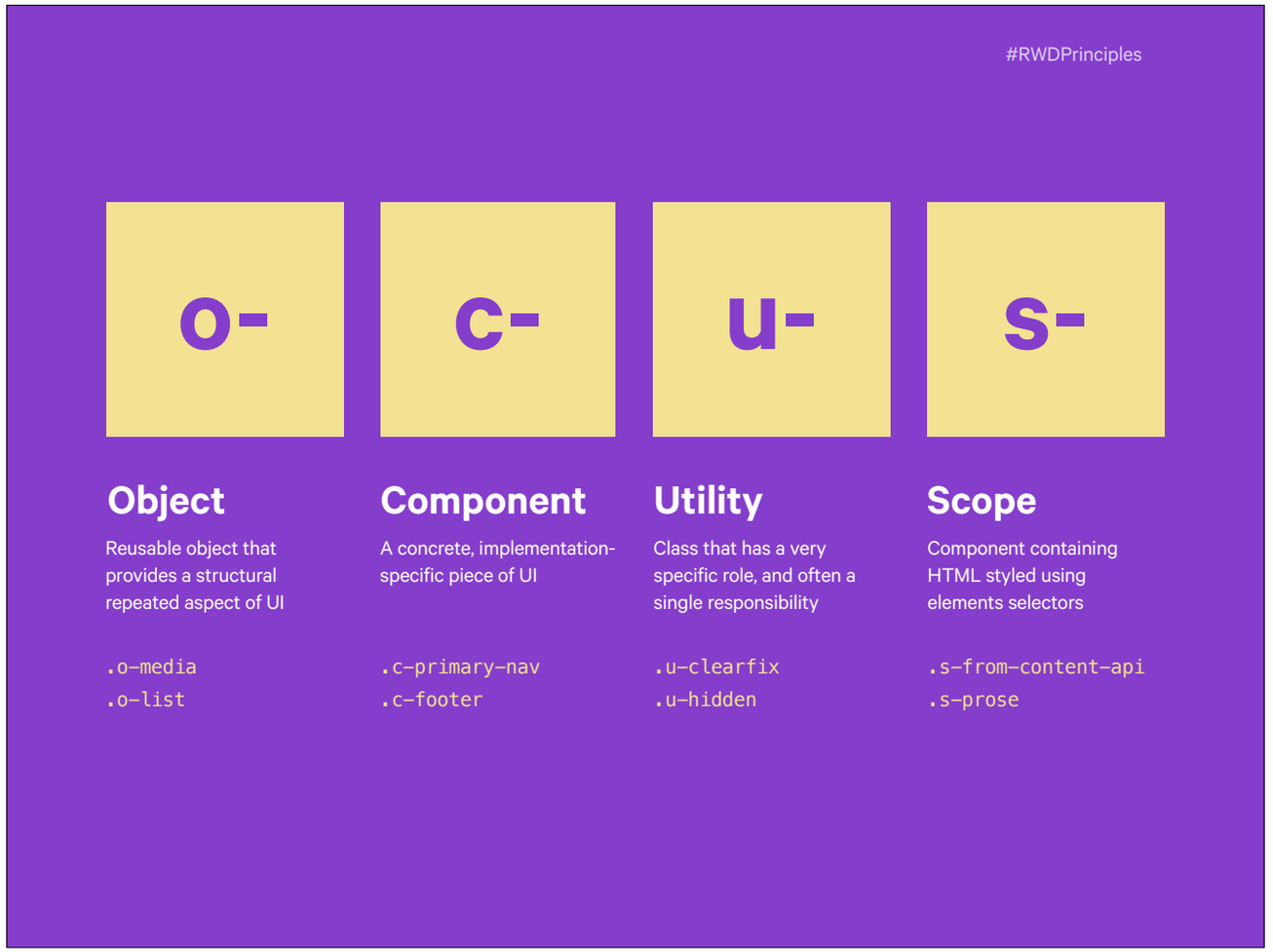
Harry Roberts: More Transparent UI Code with Namespaces

csswizardry.com/2015/03/more-transparent-ui-code-with-namespaces/

One way we can self-document code, is to use namespaces. I've recently starting using an approach advocated by Harry Roberts.

A namespace will tell us exactly how classes behave in a more global sense. A namespace tells us exactly what a class (or suite of classes) does in non-relative terms.

His post goes into more detail than I am able to here. But what I like about this approach, is that recognises the multiple ways CSS may be used in a project and allows us to identify them.



He provides eight examples, so I will only touch on four here, but each ties in well with a particular software design principle:

Object: Don’t repeat yourself

Component: Modular design

Utility: Single Responsibility

Scope: Be liberal in what you receive (conservative in what you send)

```
<nav class="c-nav u-clearfix" role="navigation">
    <ul class="c-nav-list o-list">
        <li class="c-nav-list__item  o-list__item">
            <a class="c-nav-list__label" href="#">Journal</a>
        </li>
        <li class="c-nav-list__item  o-list__item">
            <a class="c-nav-list__label" href="#">Projects</a>
        </li>
        <li class="c-nav-list__item  o-list__item">
            <a class="c-nav-list__label" href="#">Photos</a>
        </li>
        <li class="c-nav-list__item  o-list__item">
            <a class="c-nav-list__label" href="#">Contact</a>
        </li>
    </ul>
</nav>
```

Returning to our earlier example, we can now update it as follows.

```
<div class="s-prose">
    <h1>HTTPS + Compression Considered Harmful?</h1>
    <p>As I get closer to launching my redesigned website (this side…</p>
    <p>Enabling HTTPS wasn't too difficult, largely thanks to Josh's…</p>
    <h2>An unexpected consequence of enabling HTTPS</h2>
    <p>More precisely, a site is vulnerable to this attack when pages:</p>
    <ul>
        <li>Are served from a server that uses HTTP-level compression</li>
        <li>Reflect user-input in HTTP response bodies</li>
        <li>Reflect a secret in HTTP response bodies</li>
    </ul>
</div>
```

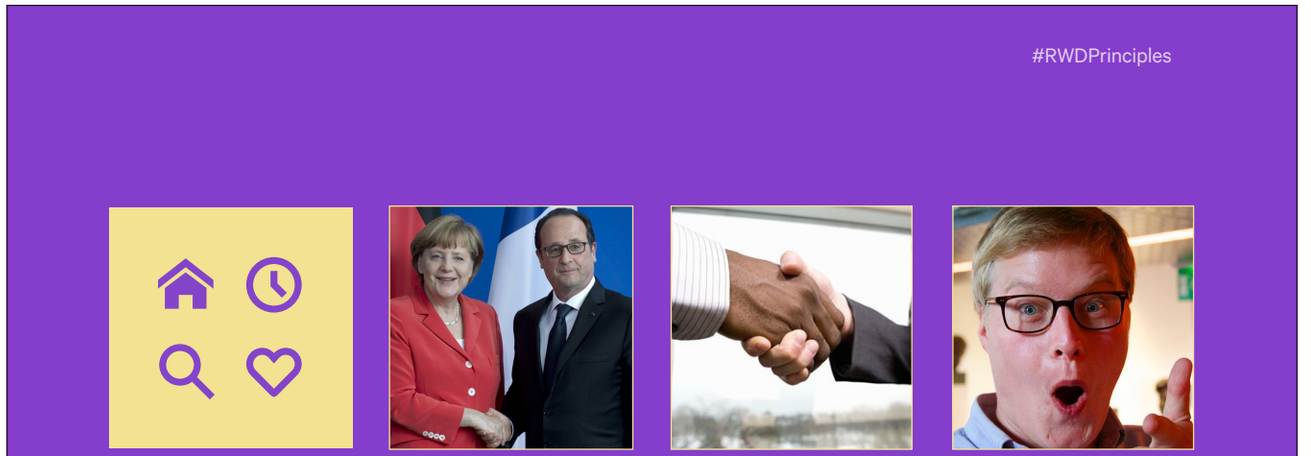
```
/* Utility: Clear floats */
.u-clearfix { }

/* Object: Interface lists */
.o-list { }
.o-list__item { }

/* 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 { }
```

Example

Image types

Again, thinking in a more modular way, we can invoke ‘it depends’, and use different implementation techniques for our images, depending on their purpose and what information they provide.



After all, not all images are created equal. Some convey information, others add visual affordances to interface elements, others are more decorative.

There is actually a fourth option that is often overlooked: we just don’t include an image at all.

#RWDPrinciples

Decoration

UK unemployment total falls to 2.5m

UK unemployment fell in the three months to December, while the number in work also jumped to a new record, official figures show.

The jobless total fell by 14,000 between October and December to 2.5 million, **the Office for National Statistics (ONS) said**.

The number in work rose by 154,000 to 29.7 million. More than 580,000 more people are employed than a year ago.

The number of people claiming Jobseeker's Allowance in January fell by 12,500 to 1.54 million.

Overall, the ONS said there were 29.73 million people in

Content

The Jobcentre Plus brand

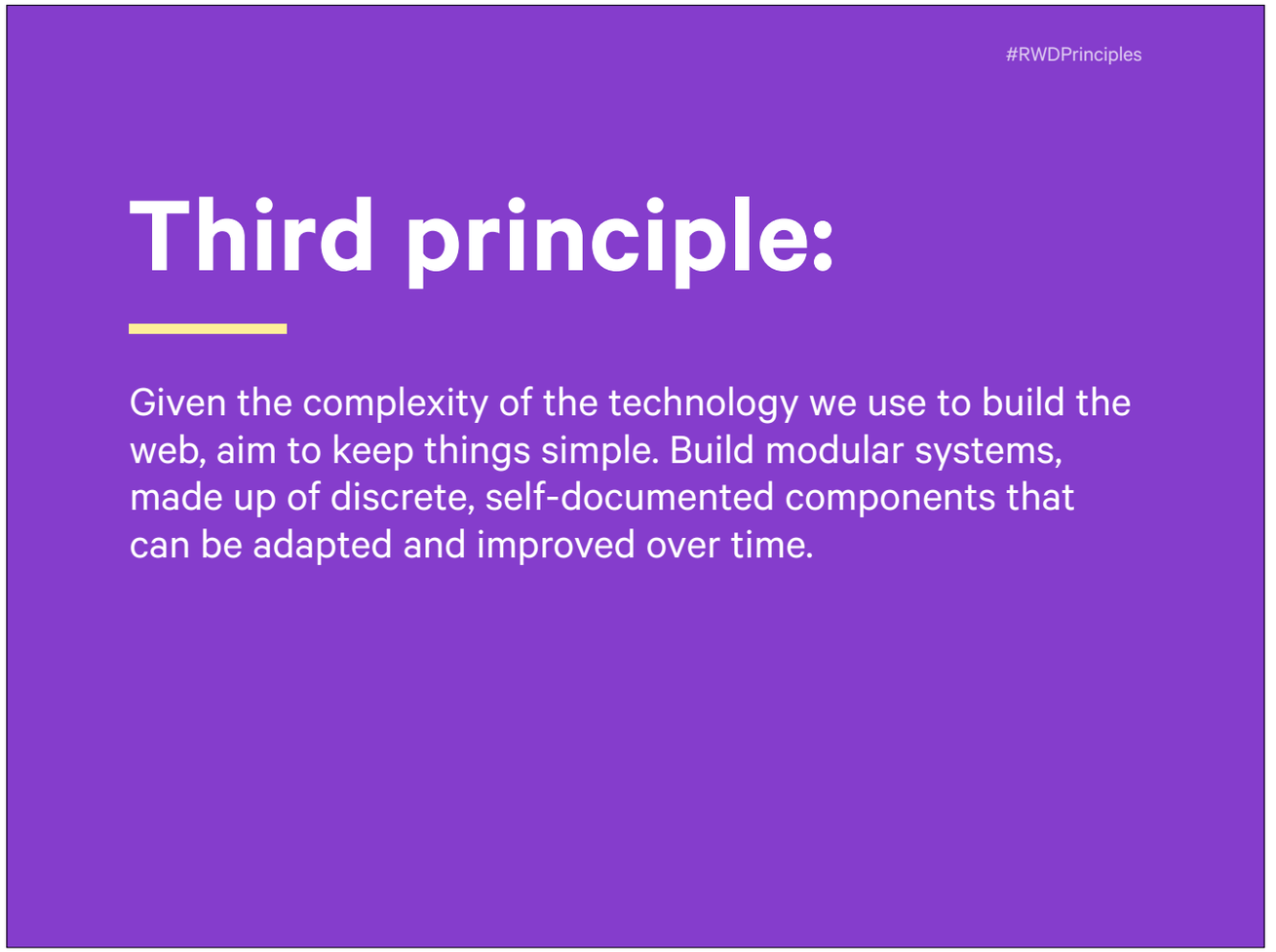
Our brand encompasses the nature of our relationship with customers. It is expressed throughout our communications, the attitude of our staff, and is symbolised by our identity.

Within the offices, the expression of the brand identity is delivered through branded signs, graphic communications, and a range of component elements that are synonymous with Jobcentre Plus. These are set against a backdrop of colours, finishes and materials of the brand environment. The consistent application of the Jobcentre Plus identity and the look and feel of the

It’s worth recognising that the implementation we choose depends just as much on the surrounding context, as it does the image.

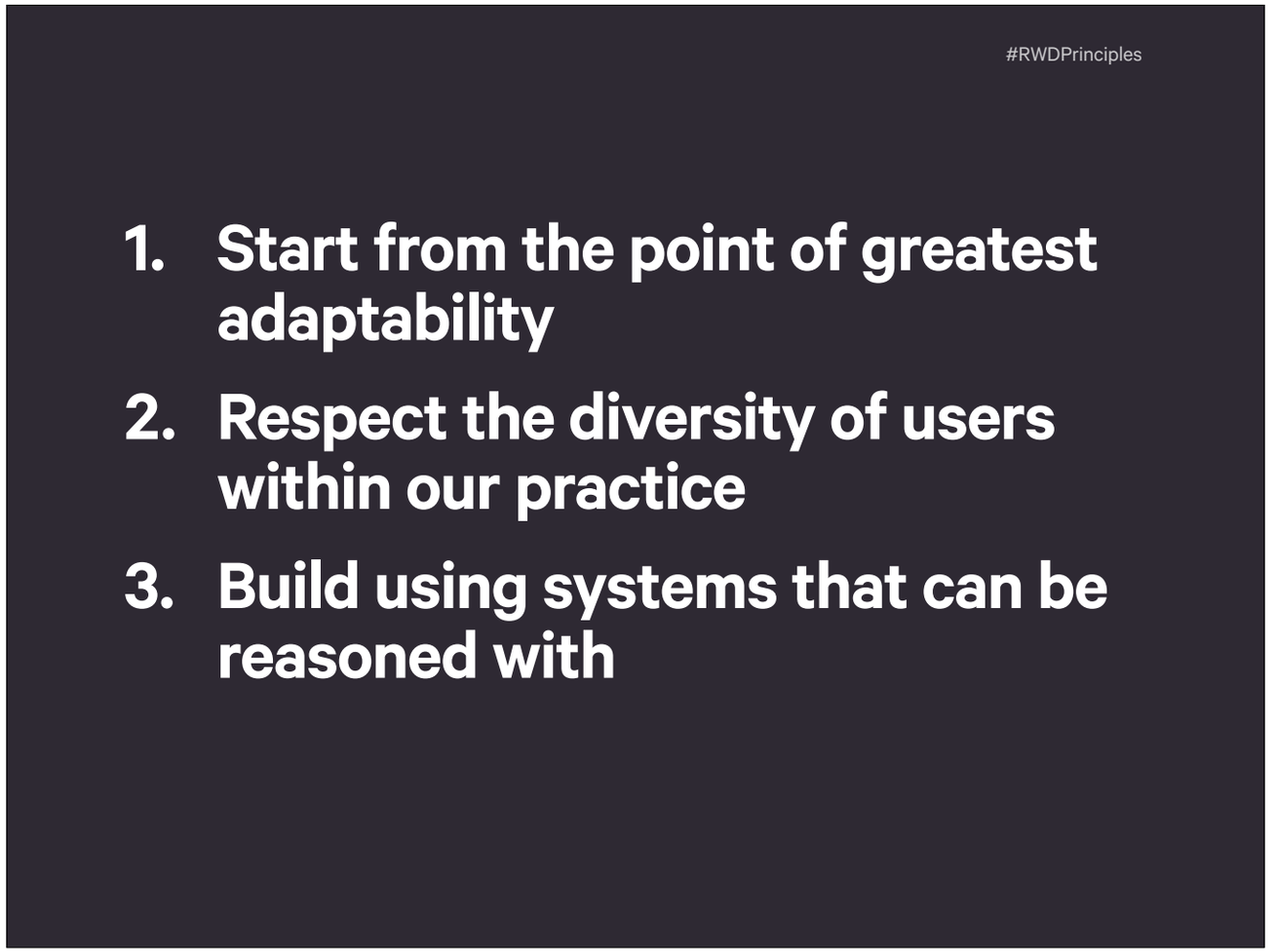
Take for example this image of a Job Centre Plus sign. Used on a news story talking about the latest employment figures, we might consider it decoration; it conveys no information, beyond perhaps reinforcing what the story is about.

Were that same image to appear on a page about the Job Centre Plus brand and how it should be implemented, this image provides essential information.



These are just a few examples, but there are many more.

Given the complexity of the technology we use to build the web, aim to keep things simple. We can achieve this by building modular systems, made up of discrete, self-documented components that can be adapted and improved over time.



So, to recap:

- We may not know how reliable the network will be, or what features a user agent will support, but by starting from a foundation that makes few assumptions, we can deal with this landscape.
- Through research and user centred design, we can can get to know our users. Better still, we should reflect their diversity within our own teams.
- By working closely with our colleagues we can build empathy within teams, and that will help us build better products.

By following these principles that deal with known unknowns, we will be better placed to deal with the larger, scarier unknown unknowns.

Somebody asked me the other day, why this talk is not called responsive web design principles, but responsive principles.

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, and you can use the following hashtag to continue the discussion.

I look forward to hearing your suggestions!

#RWDPrinciples

Thank-you

paulrobertlloyd.com / @paulrobertlloyd