

Responsive Web Design

Design for web content

Overview

- A short history of web design for media
- “One Web” and “Mobile First”
- Adding flexibility to your layout
- Flexible images?
- Media queries
- Fixed/Fluid/Adaptive/Responsive
- “Content-out”
- How to design websites now

History and language

To fully understand the concept and principles of responsive web design, we need to understand the context from which it emerged.

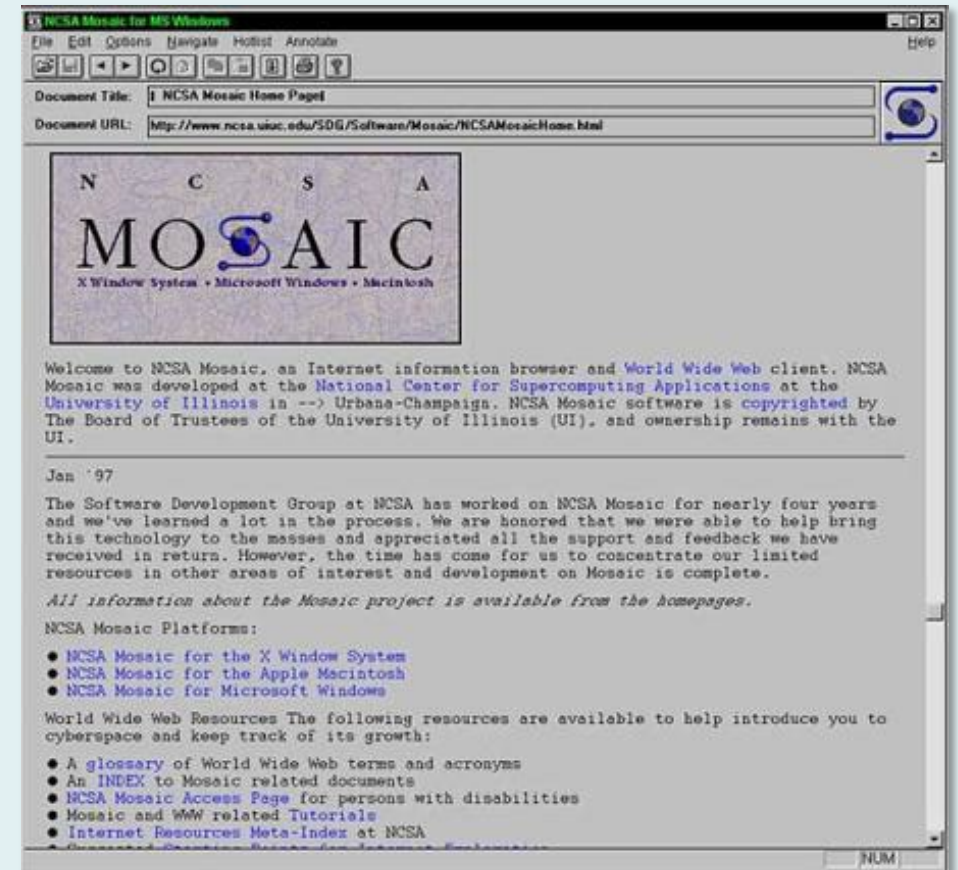
A short historical overview will help, along with a description of several other concepts that form a logical progression towards RWD.

Remember, responsive design is just 13 years old and for the first 20 or so years of the Web, other methods of page layout design were employed.

The evolution of web layouts

HTML4, Tables and Frames

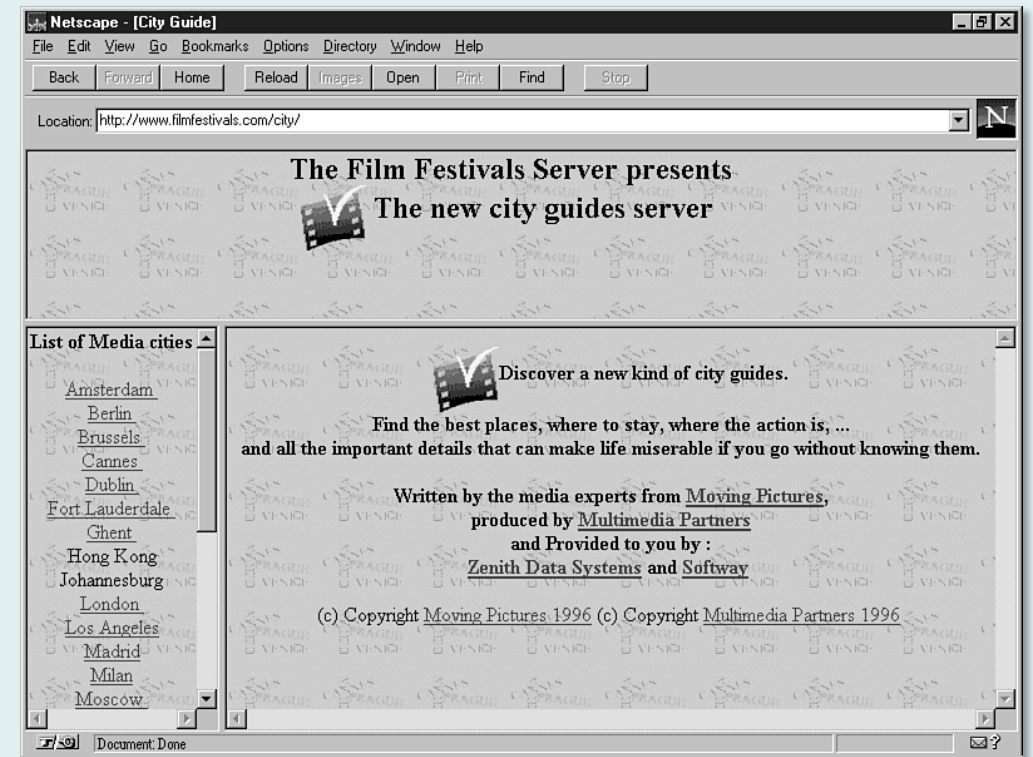
Years ago (mid 1990's) it was common practice to let content flow to the margins of the browser window. Most screens weren't high resolution, so lines of text rarely exceeded a comfortable length...



The Mosaic browser with free-flowing content 1993

The evolution of web layouts

Later (late 1990's), as interface design became more complex (navigation columns!), tables or frames were used to divide and arrange the components of a page. Tables were used because there wasn't much alternative, and they were simple to code. In March 1996, Netscape Navigator introduced frames, a system where different page elements could be marked up using different HTML files. This seemed like a good idea at the time. Frames were included as part of the HTML 3.0 specification by W3C.



Netscape Navigator with content frames 1996

[HTML Frames are obsolete](#)

Table-based layout

```
<html>
<head>
<title>WEB PAGE TITLE GOES HERE</title>
</head>

<body style="margin: 0px; padding: 0px; font-family: 'Trebuchet MS',verdana;">
<table width="100%" style="height: 100%;" cellpadding="10" cellspacing="0" border="0">
  <tr>
    <td colspan="2" style="height: 100px;" bgcolor="#777d6a"><h1>Website Logo</h1></td>
  </tr>
  <tr>
    <td width="20%" valign="top" bgcolor="#999f8e">
      <a href="#">Menu link</a><br>
      ...
      <a href="#">Menu link</a>
    </td>
    <td width="80%" valign="top" bgcolor="#d2d8c7">
      <h2>Page heading</h2>
      Here's a two column ...<br><br>
      The second table row ...
    </td>
  </tr>
</table>
</body>
</html>
```

For your amusement only, a typical table-based layout. Notice that all styling is done in the markup – there's no CSS (because it didn't exist at the time). However, this layout is "flexible", note the percentage values for width.

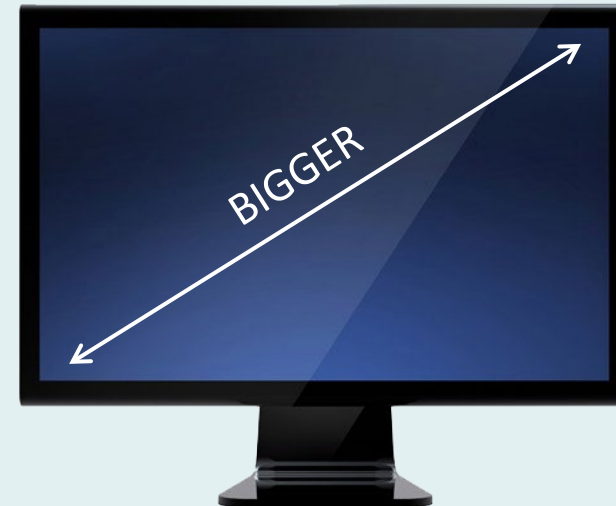
Website Logo < ~ Header ~ >	
Menu link Menu link Menu link Menu link Menu link	Page heading Here's a two column layout with a header section that spans the width of both columns. The first table row creates the header and contains a single table cell which uses the colspan="2" attribute-value pair. The website logo typically goes in the header section. The second table row contains two table cells which create the menu column (left) and the content column (right). The colspan attribute is not set in either so they default to colspan="1".
Menu column	Content column

XHTML: the new craze!

XHTML and CSS

In January 2000, XHTML became a W3C recommendation. Although CSS 1.0 was recommended back in December 1996, the first browser to fully support it (IE5 for Mac) wasn't released until March 2000, by which time, CSS2 had been recommended. So it wasn't until the early 21st Century that all the tools were in place to enable webpage layouts to be implemented in the way we do now.

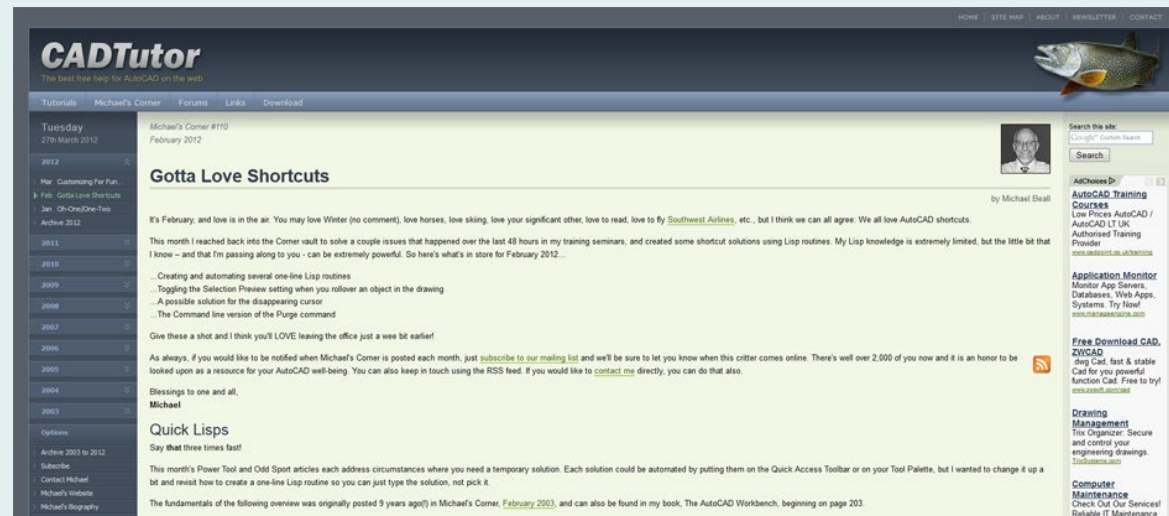
Through the 2000's, computer monitors changed from CRTs to LCDs, became bigger, changed aspect ratio, and delivered higher resolutions. Whereas in the past, browsers (and therefore websites) had been viewed full-screen, they were now viewed in resized windows.



The fixed/flexible debate

User choice or designer control?

A debate about how website layouts should be constructed followed. Some (mainly graphic designers) said that the designer should be in control of the layout because they knew what looked/worked best and that layouts should be fixed width. Others took a more “user-centred” approach and argued that the user should be able to choose what width they wanted the webpage to be and that layouts should be flexible (fluid or elastic). See [The Dao of Web Design](#).



A hybrid fluid/fixed layout became a common solution for many websites

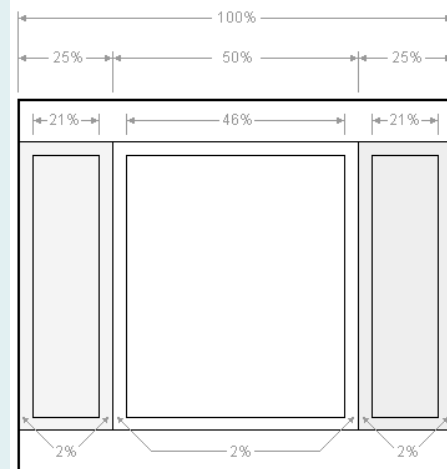
[Fixed vs. Fluid vs. Elastic Layout: What's The Right One For You?](#) Smashing Magazine (2009)

Complexity and divitis

How many nested divs!?

Although the philosophical debate was won by those who advocated flexible design and user choice, it soon became clear that the available tools (XHTML and CSS) were just not up to the job of creating simple, flexible designs in all the many combinations of columns required. In fact, the 3-column liquid layout became known as the “holy-grail” layout, because a good, simple solution was so difficult to achieve. During this same period, we saw the rise of the blog, with its distinctive multi-column layout. Many designers took a pragmatic view and resorted to fixed-width designs, making sure that they worked with most desktop/laptop monitors.

Percentage dimensions of the holy grail layout



The nested div structure

I've colour coded each div so it's easy to see:



Design for Web Content

MEDIA TYPES AND THE MOBILE WEB

The Mobile Web

You want to view websites on your phone!?

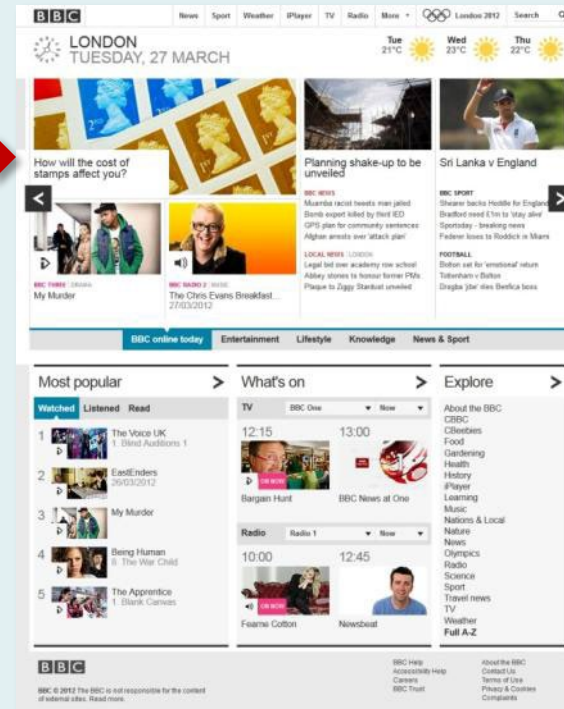
Although the Handheld media type could be used to serve a different stylesheet to handheld devices, few designers bothered because most mobile browsers weren't particularly capable (the *handheld* media type is now deprecated). As browsers improved, developers decided to simply scale the default stylesheet because most sites didn't have a handheld stylesheet. Many designers then decided it would be a good idea to provide an alternate mobile version of their websites – the mobile web was born.



bbc.co.uk/



bbc.co.uk/mobile/



One Web

W3C Mobile Web Best Practices

One Web means making, as far as is reasonable, the same information and services available to users irrespective of the device they are using. However, it does not mean that exactly the same information is available in exactly the same representation across all devices.

Opera said...

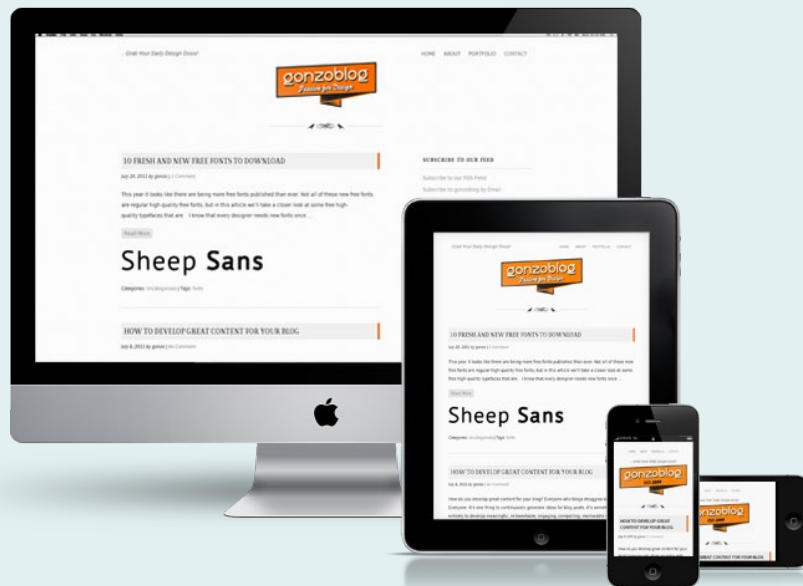
One Web is the full, complete web that people access every day from their computers, their mobile devices, their home appliances, in their homes, offices and on the go — anytime, anywhere. With fundamentals, like web standards, and technical innovations supporting widespread accessibility, people are able to access the same familiar web content seamlessly on any device.

The “One Web” debate is now over. Does it mean exactly the same content/services for all devices/contexts or does it mean modified versions? What is agreed is that it doesn’t mean one website for mobile users and another for desktop users.

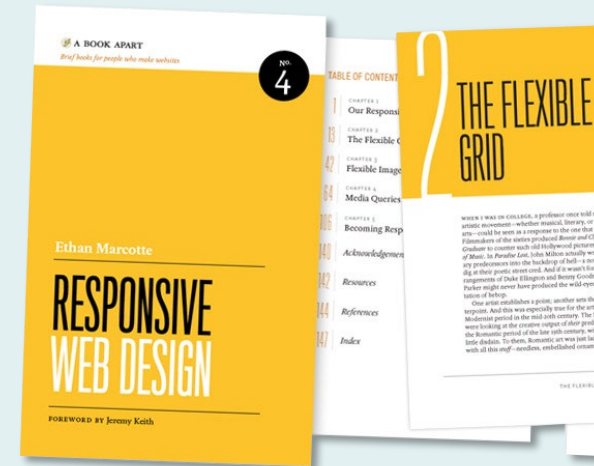
Responsive Web Design

Flexible, Adaptive, Responsive – what does it all mean?

Flexible web design means that a webpage can easily be viewed full-screen or in a resized browser window on a desktop monitor with widths being defined as percentages. *Adaptive* design means that a webpage can sense the screen width and configure itself accordingly, using fixed breakpoints*. *Responsive* design combines these 2 ideas, using flexibility between break points and providing a good user experience, irrespective of device/screen.



[Responsive Web Design](#) - .net Magazine



* [Other views](#) exist...

Mobile First

Mobile is where your design begins

It is now standard practice to begin the responsive design process by designing for mobile devices (narrow viewports) first. Once that phase of design is complete, the layout of a page can be tweaked so that the arrangement of content works well for wider viewports.



Design for Web Content

RESPONSIVE WEB DESIGN

Making webpages responsive



Ethan Marcotte – it's all his fault...

In May 2010, Ethan Marcotte wrote an article for A List Apart entitled [Responsive Web Design](#).

What are the ingredients?

Standing on the shoulders of giants...

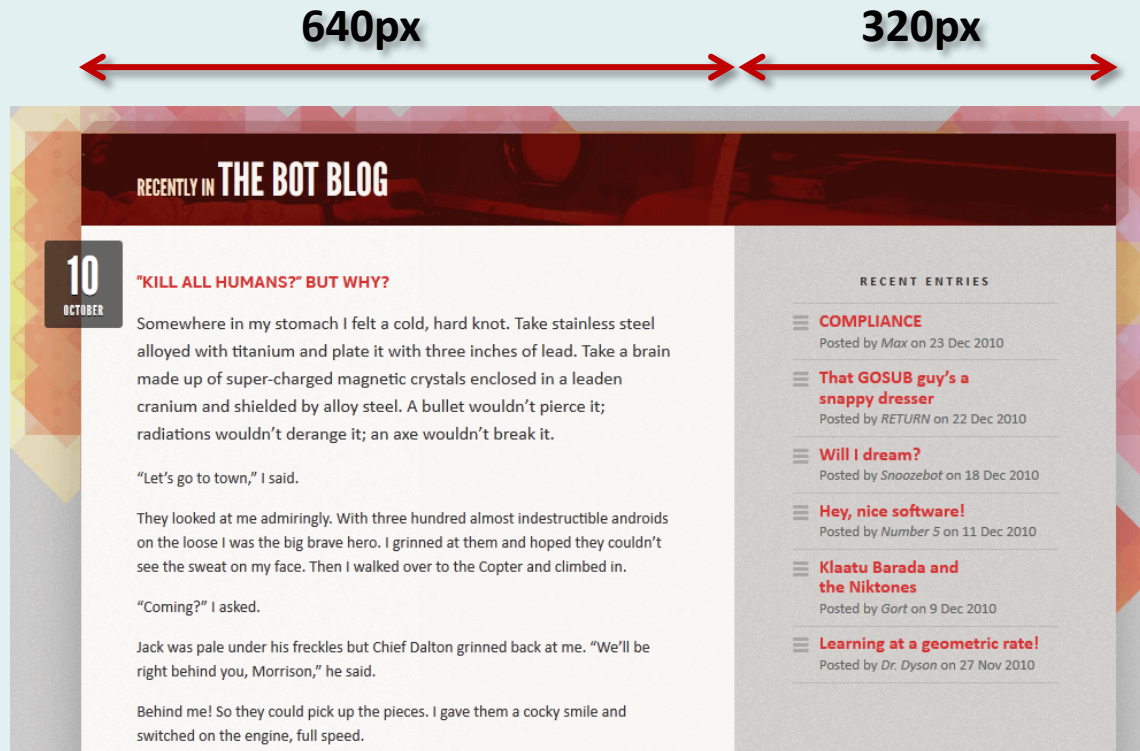
Marcotte was building on ideas first proposed in John Allsopp's seminal article [The Dao of Web Design](#) (also for A List Apart), published in April 2000. In it, Allsopp says:

The control which designers know in the print medium, and often desire in the web medium, is simply a function of the limitation of the printed page. We should embrace the fact that the web doesn't have the same constraints, and design for this flexibility. But first, we must "accept the ebb and flow of things."

In his article, Marcotte says that he believes that now is the time to put these ideas into practice and it is all made possible using CSS3 media queries, but that's not the only ingredient. Marcotte identifies 3...

- 1 A flexible grid-based layout that uses relative sizing.
- 2 Flexible images and media, through dynamic resizing or CSS.
- 3 Media queries and media query listeners.

A Typical Fixed Width Blog



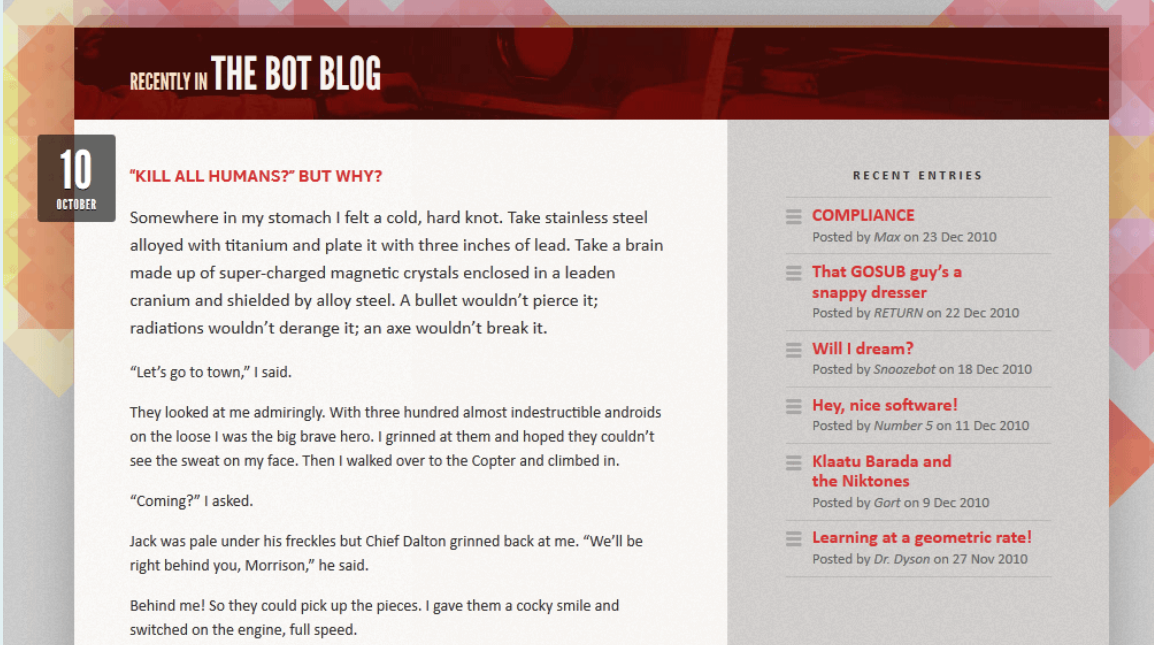
```
#content {  
    margin: 36px auto;  
    width: 960px;  
}  
main {  
    float: left;  
    width: 460px;  
}  
aside {  
    float: right;  
    width: 320px;  
}
```

When working with grids, the number of pixels in each column can be calculated and the fixed-width blog layout above could be described with the fixed pixel values given in the CSS on the right.

Pixels to Percentages

66.6666666%
33.3333333%

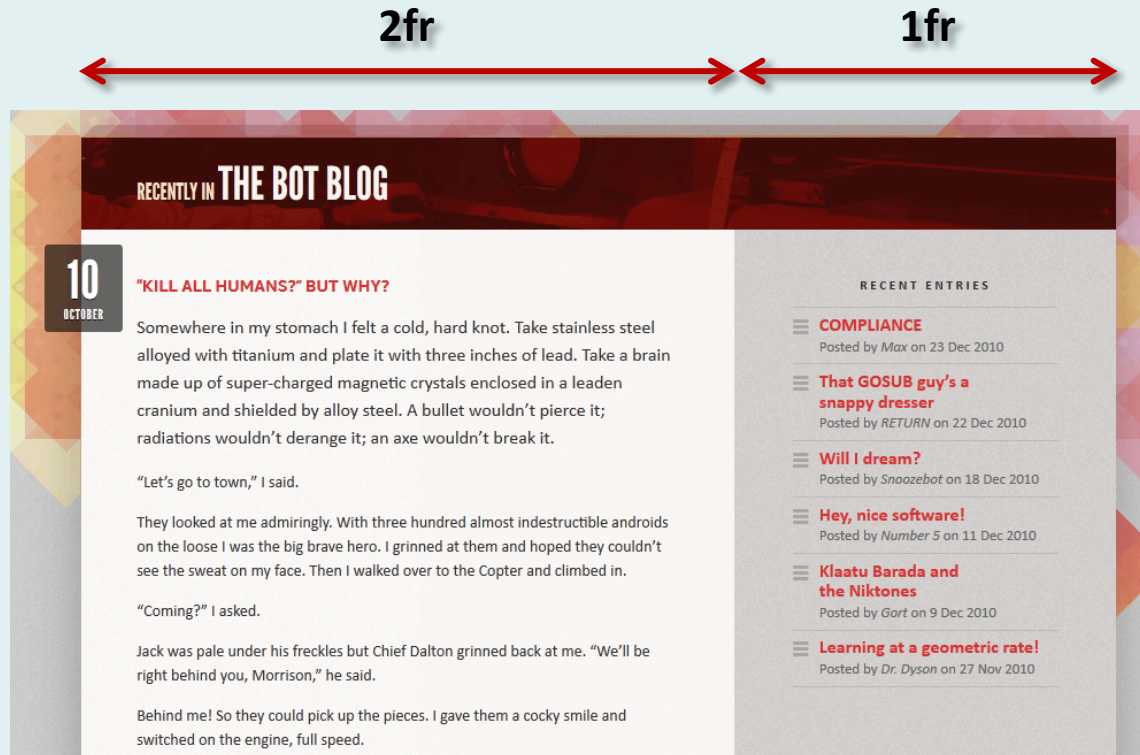
←
→



```
#content {
  margin: 36px auto;
  width: 90%;
}
main {
  float: left;
  width: 66.6666666%;
}
aside {
  float: right;
  width: 33.3333333%;
}
```

Although the calculated percentage values look a bit ungainly, they work perfectly well – don't be tempted to round the numbers up or down. Now, when the browser window is resized, the columns will also resize proportionally.

CSS Grid Layout



```
#content {  
  width: 90%;  
  display: grid;  
  grid-template-columns: 2fr 1fr;  
  margin: 36px auto;  
}
```

CSS Grid Layout was specifically designed to avoid all this percentage madness by introducing the more sensible fraction units. Now, a 2:1 ratio is easily expressed.

Flexible Images

You want to display images smaller than they really are?

To some extent, this technique goes against the grain. In order to conserve bandwidth, images should only be displayed at their actual size. However, responsive design requires that we resize images to maintain the integrity of our layouts.

```
<figure>  
    
  <figcaption>Lo, the robot walks</figcaption>  
</figure>
```

The markup, above will be used to describe our image and a caption. Note that we are using *width* or *height* attributes on the image element because it is still good practice to do so.

[Setting Height And Width On Images Is Important Again](#)

This markup uses the HTML5 `<figure>` and `<figcaption>` elements.



Flexible Images

The CSS

In order to create a “fluid” image, we must apply the fluid bit (the percentage) to the container. We must then use the *max-width* property with a value of 100% for the image so that it remains inside its container and is scaled with it as the container gets larger or smaller.

```
figure {  
    width: 33.3333333%;  
}  
img {  
    max-width: 100%;  
    height: auto;  
}
```

Of course, things are never quite that simple – older browsers may cause problems. See [Fluid Images](#) for more information.



Media Queries

The magic ingredient

CSS3 media queries are the magic ingredient that make responsive web design possible. A media query is a bit like a conditional statement. If the condition is true, something changes, if not, nothing changes. In the case of webpage layouts, we can make the condition the width of the screen in pixels and then apply a rule to change the default layout if it is true. For example, we could say, if the screen is more than 700px wide, the hamburger menu should be displayed as a horizontal navigation bar. In such an example, 700px would be the “breakpoint” where the design changes.

```
@media (min-width: 700px) {  
  main {  
    display: grid;  
    grid-template-columns: 1fr 1fr;  
  }  
}
```

This media query says: the media (in this case, our viewport) is 700px wide or more, set the main element as a grid container with two equal width columns. Elsewhere in our stylesheet there may be another media query with a min-width of (say) 1200px that sets three equal width columns.

Up or Down?

min-width or max-width?

Notice that in the previous example, we used *min-width* to control what happens when the screen gets wider than a certain number of pixels. However, we can also test for when the screen gets smaller by using *max-width*. For those scaling up from a Mobile First perspective, things will change when the *min-width* breakpoint is found, whereas if you are scaling down from a desktop design, you will want to test using *max-width*. *Mobile first* would suggest a preference for *min-width*.

Which breakpoints should I use?

Well that depends...

Different devices will have set screen sizes (iPhone, iPad, Laptop, Desktop see Chris Coyier's [Media Queries for Standard Devices](#)), you could target those devices but what happens when they change and resolutions increase (as they inevitably will)? Some designers believe that media queries should be content-focused rather than device specific. This debate is largely won – **breakpoints should be set for content, not devices.**

My Image Gallery

This is my image gallery and it's designed to be "responsive". That means the arrangement of images changes as the viewport gets wider or narrower. Try changing the width of the browser window or view it on different devices to see what happens.

Theme: Lake District Views



image1.jpg



image2.jpg



image3.jpg



image4.jpg

As you can see, the images are arranged in one, two or four columns dependent upon the viewport width. Typically, you'll see one column on a smartphone, two on a tablet and four on a laptop. This is achieved using a *mobile first* approach. The default style is for narrow screens (mobile) where the gallery displays as a single column. The design uses two breakpoints. If the viewport is between 700px and 1199px wide, the gallery displays in two columns and if the viewport is 1200px wide or wider, the gallery will display in four columns.

Mobile First: default layout

```
<div class="gallery">
  
  
  
  
</div>
```

For narrow viewport widths, the layout obeys the default CSS rules. No media query is needed for this (mobile) view.

Image width is set to 100% and they fill the entire width of the content column.

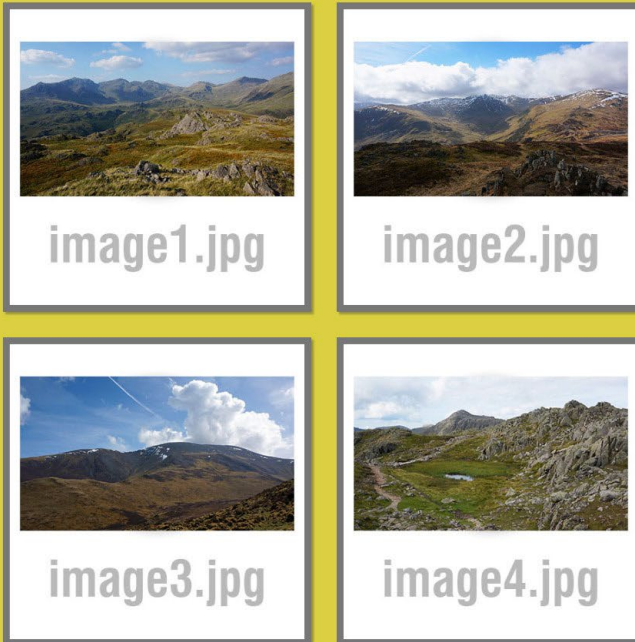
```
/* Gallery */
.gallery img {
  width: 100%; /* one image wide (mobile first default < 700px) */
  height: auto; /* to maintain aspect ratio */
  margin: 2% 0; /* no margin left or right */
  box-shadow: 3px 3px 3px rgba(50, 50, 50, 0.4);
}
```

Mobile First: breakpoint 1

My Image Gallery

This is my image gallery and it's designed to be "responsive". That means the arrangement of images changes as the viewport gets wider or narrower. Try changing the width of the browser window or view it on different devices to see what happens.

Theme: Lake District Views



As you can see, the images are arranged in one, two or four columns dependent upon the viewport width. Typically, you'll see one column on a smartphone, two on a tablet and four on a laptop. This is achieved using a *mobile first* approach. The default style is for narrow screens (mobile) where the gallery displays as a single column. The design uses two breakpoints. If the viewport is between 700px and 1199px wide, the gallery displays in two columns and if the viewport is 1200px wide or wider, the gallery will display in four columns.

As the viewport gets wider, there comes a point where the images get too big and it would make more sense for the layout to change and display two images across the width of the content column. We decide this should happen when the viewport gets to 700px, so we add an appropriate media query that changes the width and margin of the images. We also introduce floats.

```
@media (min-width: 700px) {  
  .gallery {  
    display: flow-root; /* prevents element collapsing */  
  }  
  
  .gallery img {  
    float: left;  
    width: 48%; /* two images wide: (48 x 2) + 4 = 100%; */  
  }  
  
  .gallery img:nth-child(even) { /* selecting even images */  
    margin-left: 4%;  
  }  
}
```

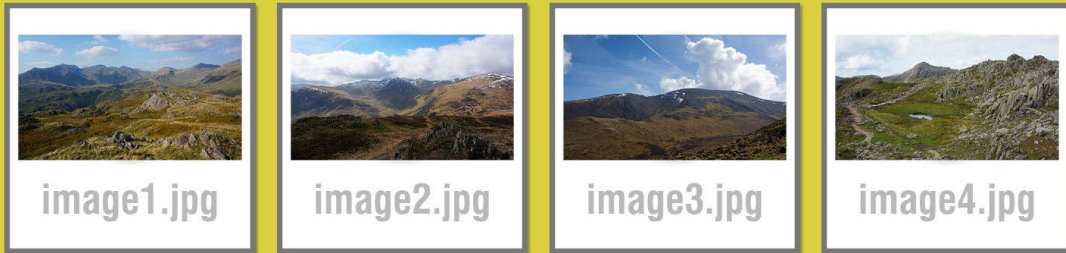
Media queries are used to change only those CSS properties that need to be changed.

Mobile First: breakpoint 2

My Image Gallery

This is my image gallery and it's designed to be "responsive". That means the arrangement of images changes as the viewport gets wider or narrower. Try changing the width of the browser window or view it on different devices to see what happens.

Theme: Lake District Views



As you can see, the images are arranged in one, two or four columns dependent upon the viewport width. Typically, you'll see one column on a smartphone, two on a tablet and four on a laptop. This is achieved using a *mobile first* approach. The default style is for narrow screens (mobile) where the gallery displays as a single column. The design uses two breakpoints. If the viewport is between 700px and 1199px wide, the gallery displays in two columns and if the viewport is 1200px wide or wider, the gallery will display in four columns.

© David Watson 2018

Eventually, even two images look too big, so we add another breakpoint that switches the layout to four images wide. We decide this should happen for viewport widths of 1200px or greater and we add another media query to deal with this situation. Again, we change only the width and margin. All the other default declarations remain unchanged, and this media query will inherit the changes made at 700px (e.g. floats) because we are using *min-width*.

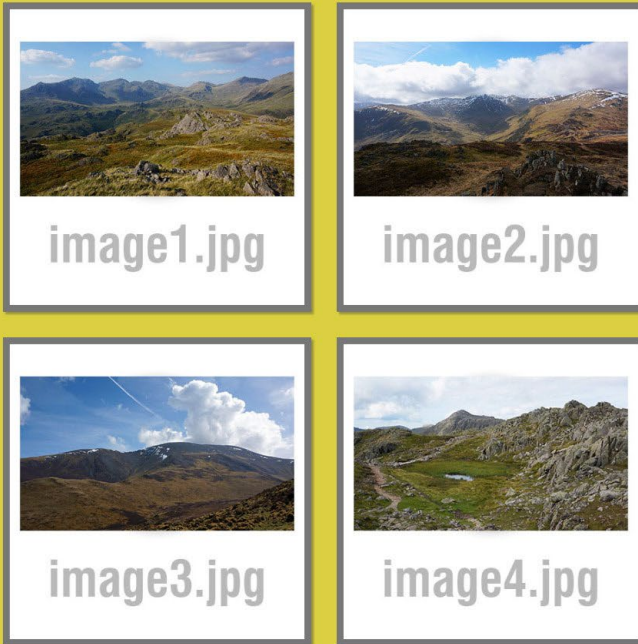
```
@media (min-width: 1200px) {  
  .gallery img {  
    width: 23.5%; /* four images wide: (23.5 x 4) + (2 x 3) = 100%; */  
  }  
  .gallery img:nth-child(n+2) { /* selecting all but the first image */  
    margin-left: 2%;  
  }  
}
```


Mobile First: Grid layout

My Image Gallery

This is my image gallery and it's designed to be "responsive". That means the arrangement of images changes as the viewport gets wider or narrower. Try changing the width of the browser window or view it on different devices to see what happens.

Theme: Lake District Views



As you can see, the images are arranged in one, two or four columns dependent upon the viewport width. Typically, you'll see one column on a smartphone, two on a tablet and four on a laptop. This is achieved using a *mobile first* approach. The default style is for narrow screens (mobile) where the gallery displays as a single column. The design uses two breakpoints. If the viewport is between 700px and 1199px wide, the gallery displays in two columns and if the viewport is 1200px wide or wider, the gallery will display in four columns.

CSS Grid was designed to be used for responsive layouts, so it's no surprise that the same responsive solution as implemented with floats on the previous slides, requires less CSS when using Grid. In this case, we don't even need to worry about the image sizes.

```
/* Media Queries for Gallery */
@media (min-width: 700px) {
  .gallery {
    display: grid;
    grid-template-columns: 1fr 1fr; /* two equal sized columns */
    grid-gap: 4%;
  }
}

@media (min-width: 1200px) {
  .gallery {
    grid-template-columns: 1fr 1fr 1fr 1fr; /* four equal sized columns */
    grid-gap: 2%;
  }
}
```

Breakpoints

How do we decide where the breakpoints should be?

Breakpoints are where the design changes as the viewport gets larger (or smaller) but should the design change for common viewport sizes (e.g. phone, tablet and desktop) or should it only change when appropriate for the content?

“While it’s tempting to choose breakpoints early in the design process, perhaps based on the dimensions of popular devices we know we need to support, the truth is that we shouldn’t choose breakpoints at all. Instead, we should find them, using our content as a guide.”

Scott Jehl – “Responsible Responsive Design”

Some commentators have a more straightforward way of expressing this principle...

“Start with the small screen first, then expand until it looks like shit. TIME FOR A BREAKPOINT!”

Stephen Hay

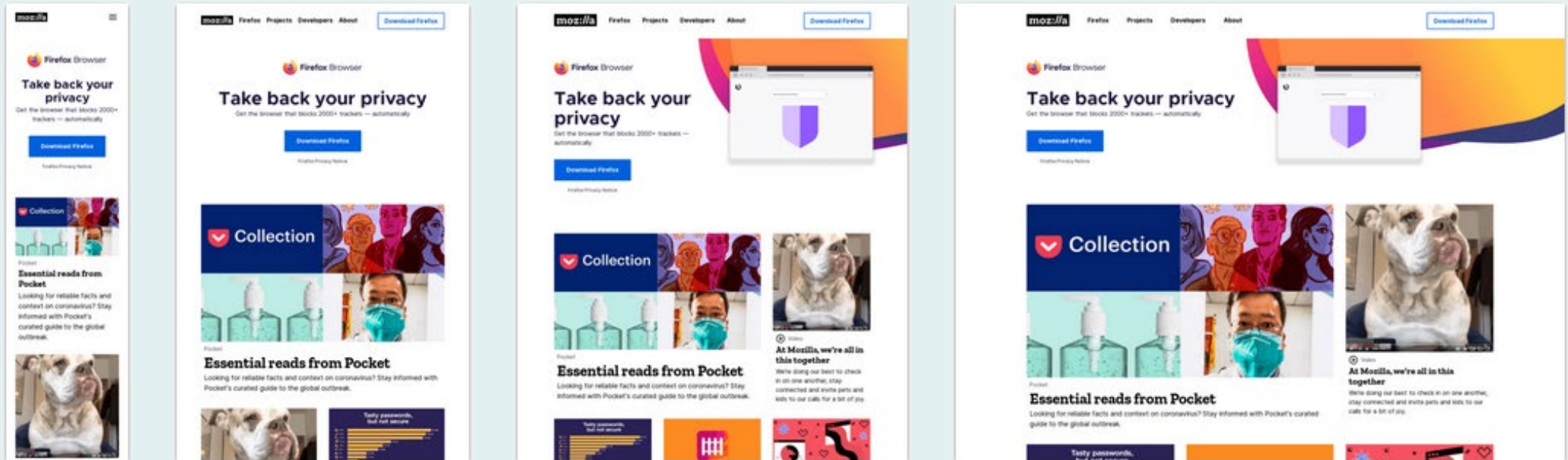
Scaling on mobile devices

Mobile device browsers assume that there is no responsive design

By default, mobile devices will scale all websites so that they fit on the device screen rather than obeying the media queries you have added to the CSS. We must tell mobile devices to display a webpage with a scale of 1 for them to render our responsive designs as we intended them to be seen. We do this by adding a viewport meta tag to the head of our webpage.

```
<head>  
  <meta name="viewport" content="width=device-width, initial-scale=1">  
  <title>...  
</head>
```

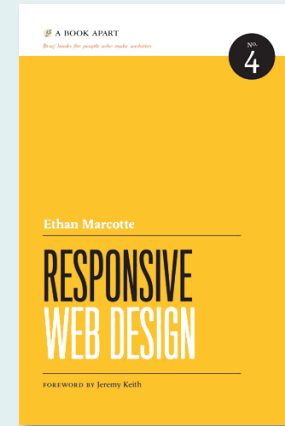

Media Queries in Action



Learning RWD

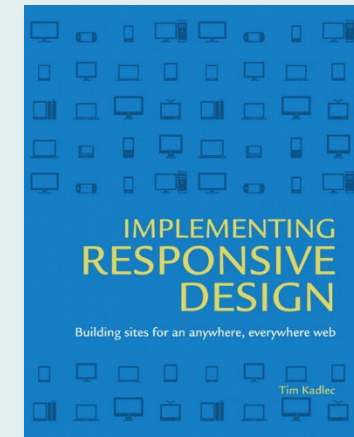
Step 1

Read *Responsive Web Design* by Ethan Marcotte. Just read it through, page by page, don't try any of it, just read and absorb the ideas.

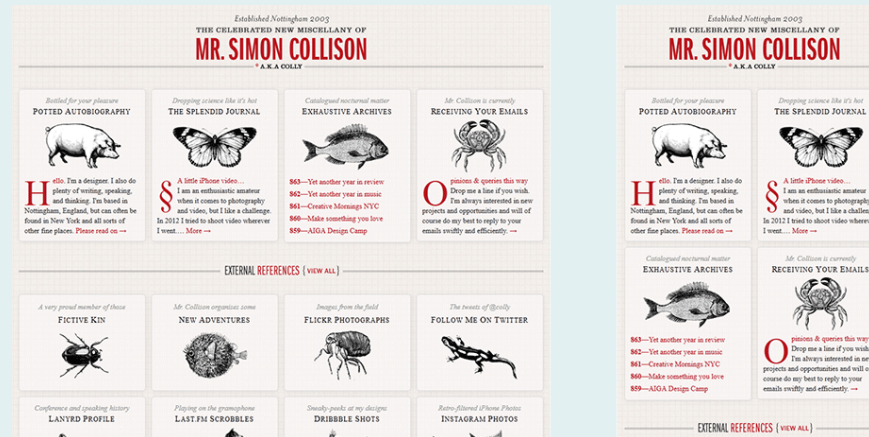


Step 2

Read and work through *Implementing Responsive Design* by Tim Kadlec. This will give you a practical understanding of the ideas.



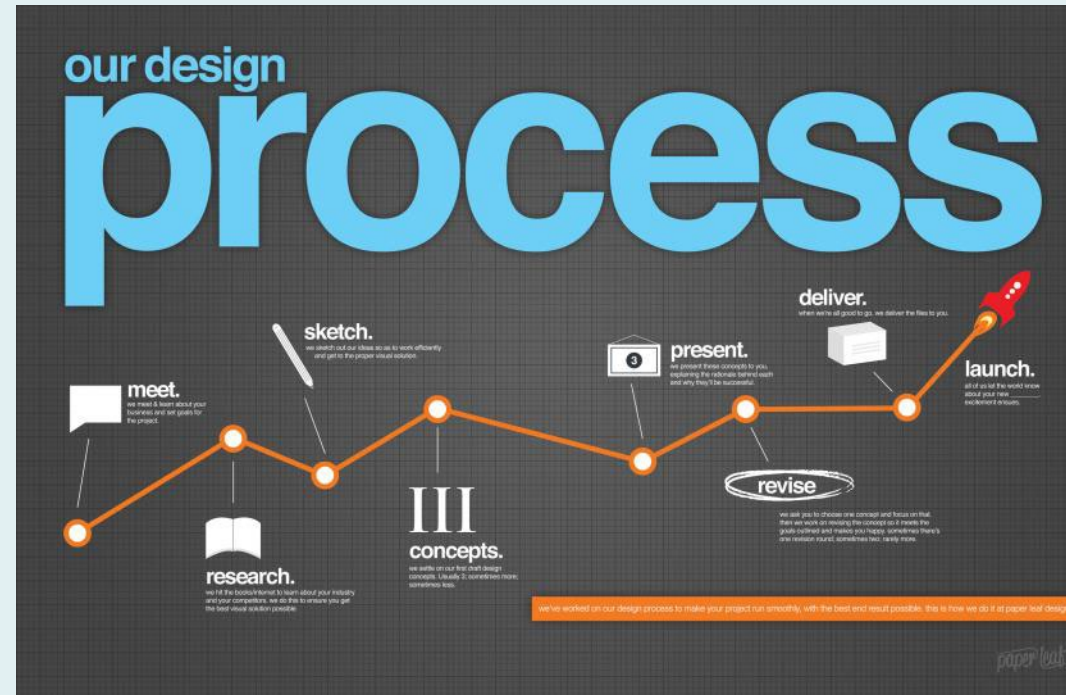
Content-out



OK, so now we look at the content first?

The “content-out” or “typography-first” approaches to web design are, in some ways, a reaction against the over-reliance that some designers place on grid systems and other non-content specific approaches to webpage layout (such as Bootstrap). The approach is logical because it focusses on the formation of blocks of content rather than the page. In adaptive/responsive pages, the page layout may change but blocks of content generally remain intact. It makes sense therefore to design content blocks first and then decide how they will be arranged at different page widths.

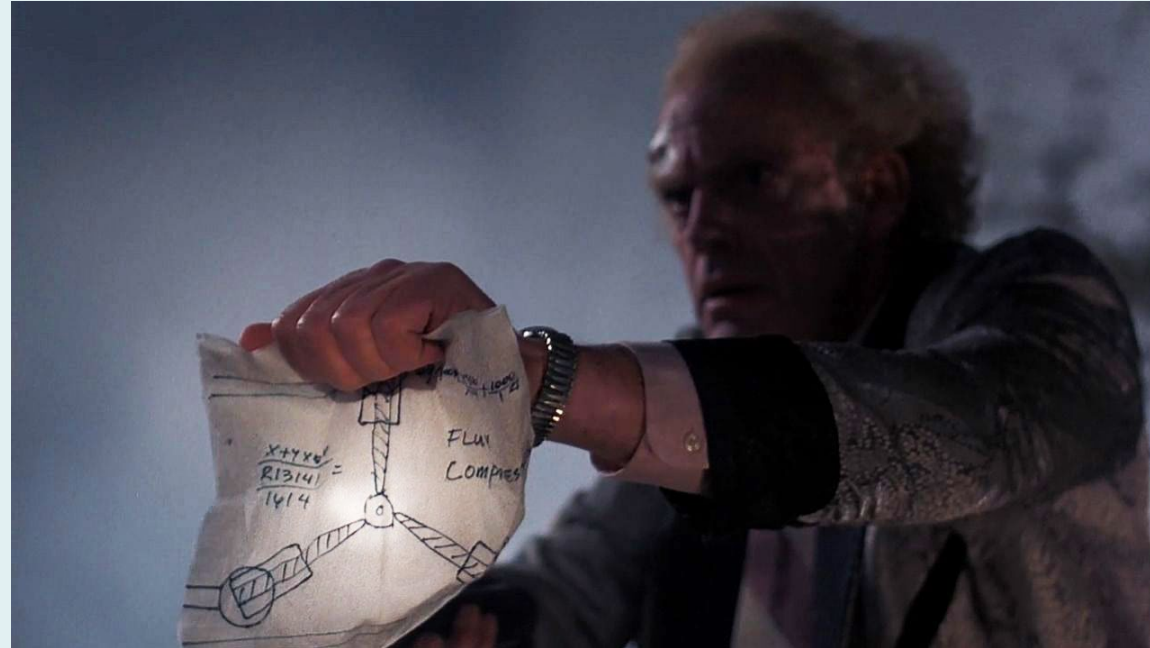
Web design – how?



I thought I knew how to design a website – now I'm not sure

Moodboards, grids, Figma, wireframes, photoshop comps and now moving content blocks and responsive pages – how the heck are we supposed to design a website, there don't appear to be any fixed points anymore? This is a mess. Should we be designing in the browser now?

Flux



The only thing we can depend on is that the web design process will be different next week, next month and next year

As designers, we need to experiment, find out what works best for us, share our ideas with others and listen to what others have to say about their own approaches to design...

As an industry, we are still learning



Jason Santa Maria – listen to what this man says

At the New Adventures in Web Design 2013 conference, Jason Santa Maria gave a talk entitled *The Nimble Process* in which he made a number of connected statements that hint at an approach to design...

[The Nimble Process](#) by Jason Santa Maria (video from the Smashing conference 2013)

*“Early in the design process,
details are your enemy.”*

Jason Santa Maria – “The Nimble Process”

*“When possible, start with text
– the web is about content.”*

Jason Santa Maria – “The Nimble Process”

“Use the right tool at the right time – designers need to be flexible.”

Jason Santa Maria – “The Nimble Process”

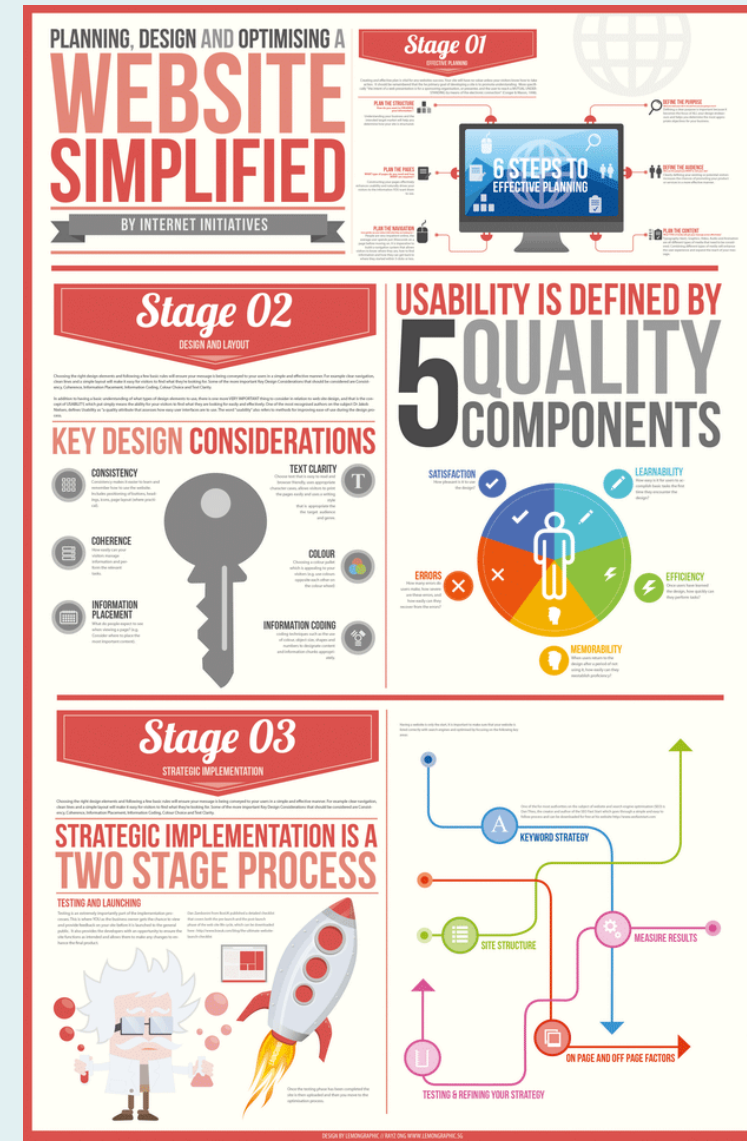
*“There is no one single path
but a pattern of possibilities;
trial and error is good.”*

Jason Santa Maria – “The Nimble Process”

Summary

So, just run that by me again – how do I design websites?

All you need to create competent web designs is a deep understanding of HTML and CSS, you must buy into the principles of *One Web*, *Content Out*, *Mobile First* and *Responsive Design* and get to grips with the use of media queries. Of course, there are many other considerations; content strategy, information architecture, search engine optimisation, accessibility and a little scripting in JavaScript and PHP will help. Oh, and familiarity with one or more content management systems is a good idea...



“I know it feels like there is an infinite amount of stuff to keep up with every single day, but you know what? It’s okay. It’s okay not to know everything.”

Jeremy Keith – The Mobile Book (forward)

THE END

of the beginning