
BEST OF
9 SMASHING
YEARS
REDESIGN
THE WEB
SMASHING
MAGAZINE

Imprint

ISBN (PDF): 978-3-945749-34-0
Cover Design: Ricardo Gimenes
eBook Strategy and Editing: Vitaly Friedman
Technical Editing: Cosima Mielke
Planning and Quality Control: Vitaly Friedman, Iris Lješnjanin
Tools: Elja Friedman
Syntax Highlighting: Prism by Lea Verou
Idea & Concept: Smashing Magazine GmbH

September 8th, 2015

Birthdays are an occasion to look back at what you have achieved and forward to what the future might bring. It's not any different here at Smashing Magazine. When we hit the publish button for the first time back in 2006, there was no master plan, only the desire to create something that we found useful and that we hoped others would find useful, too. Today, nine years, countless articles, a library full of eBooks and nine printed books later, we want to take this anniversary as an opportunity to give something back. We dived deep into our books on the look out for timeless treasures and neatly bundled them up in this free eBook. A humble gift to you, our dear readers, you who made it all possible and with whom we've grown and evolved over the years.

The choice of what to include in this best-of wasn't easy, to be honest; so much good stuff has gone through our editorial process in all these years. So to make the eBook as diverse and valuable as possible, we decided on excerpts from all our printed books for this special occasion: the Smashing Books 1–5, the Mobile Book, Digital Adaptation and the Mobile Web Handbook — a panorama of Smashing's past and current publishing history.

This eBook is going to be a journey through nine smashing years, a journey through typography, psychology, responsive web design, performance optimization and more. Will you join us?

Thank you for being part of the Smashing family!

— The Smashing Team

TABLE OF CONTENTS

Typography: Rules, Guidelines And Common Mistakes

BY ALESSANDRO CATTANEO

The following is an excerpt from the Smashing Book #1 chapter "Typography: Rules, Guidelines And Common Mistakes."

Typography is the soul of design; it lies at the heart of visual literacy. This chapter explores typography for the web and describes the methods and techniques of composition that bring documents to life and facilitate understanding.

Choosing The "Right" Type

With literally more than a hundred thousand typefaces to choose from, finding the right one for a specific purpose may seem like a daunting task. The common mistake is to choose a beautiful typeface, one that looks attractive, thus favoring form over function. This is putting the cart before the horse. However strange this sounds, the "look" of the typeface should be your least concern. But if looks are not so important, what do we base our decision on? The most important criterion is the context in which the typeface will be used and the purpose it will serve.

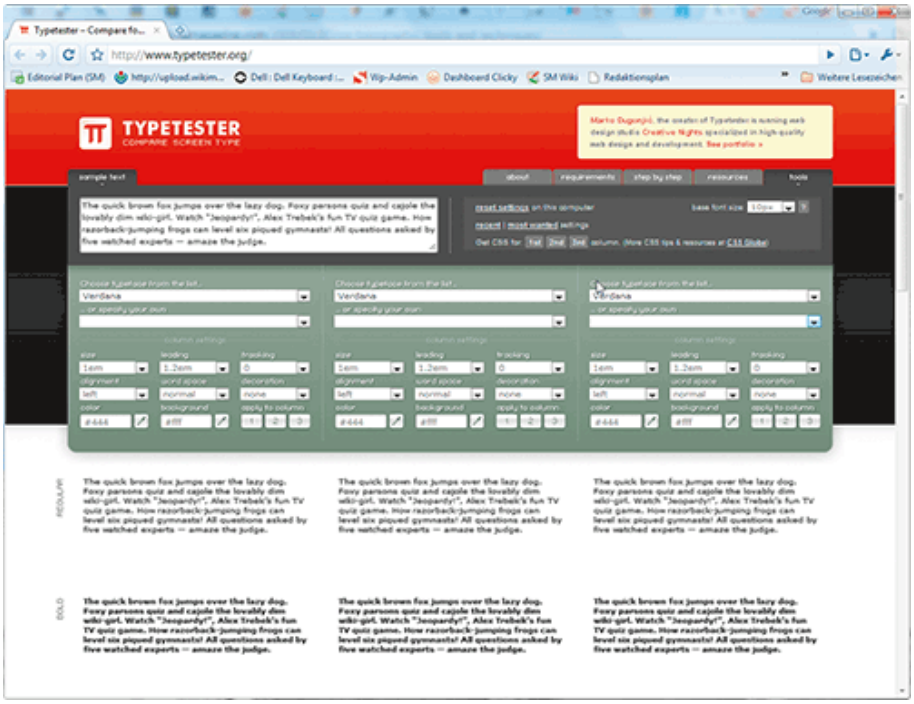
CONSIDER CONTEXT AND NARRATIVE

Before getting into the nitty-gritty of choosing a typeface and layout, always read the text first if possible. This may seem obvious, but the typeface and its presentation should be determined by the meaning of the text itself. There's an obvious problem, though. In the era of the Web, we often deal with two very different kinds of narratives:

- **Enacted narratives**: the ones we know and love. The meaning and substance of the text is known by the designer beforehand. The audience starts at the beginning and reads to the end, becoming enlightened along the way.
- **Emergent narratives**: this is content that will be generated after our design is complete. Think of social networks, in which everyone contributes text, or websites with a CMS or blog engine in which new copy is constantly generated. The audience might skim this content, jumping around the page and website in leaps and bounds, gathering nuggets of meaning along the way, pausing only when something grabs their attention.

For enacted narratives, we can read the text beforehand, understand its meaning, tone and nuances, and we can choose the type accordingly. But what do we do for emergent narratives? Well, we do the best we can. Look at the brand and environment. Imagine the context in which the text will be read. Choose a type that enriches

the meaning of the text but fades into the background rather than clamors for attention.



Marko Dugonjić's Typetester is a popular web application that allows you to test different typefaces, leading and measure on the screen and choose the best one.

NEVER CLIMB A MOUNTAIN WEARING DANCING SHOES

Virtually every designer of typefaces creates a font with well-defined criteria and for specific functions. It is up to the website designer to recognize that function before choosing the typeface. At one end of the spectrum are the workhorse typefaces designed to perform well in small sizes and unfavorable conditions (the hiking boots), and at the other end are the display typefaces (the dancing shoes).

Although they may look similar, text and display typefaces reveal many differences when scaled up.

Workhorse typefaces have sturdy features: conventional and easily recognizable letterforms, generous letter spacing, solid serifs, clear open counters, a tall x-height and ink traps (cut-out areas that prevent corners from clogging up from spreading ink). Ink spread is a non-issue on the Web, but the other factors apply. Display typefaces are much more delicate and have considerably more leeway in their shapes. There is a world between these two extremes, and many of the typefaces at either end were designed for very specific applications. In the middle are the very important day-to-day typefaces that are fit for normal text but suitable also for headlines.

Workhorse typefaces are more flexible than display typefaces. To some extent, one can wear hiking boots to a posh reception. If you combine them with casual jeans and a stylish shirt and jacket, you might get away

with it, just as workhorse typefaces might look acceptable in headlines if you adapt their spacing and use them in combination with the right design elements. But you wouldn't climb a mountain in dancing shoes. Sure, you *could*, but don't come crying after. Likewise, never use display typefaces for body text.

SIZE DOES MATTER

Of course, a lot of the above has to do with type size. With the advent of scalable digital type, any font can be used at any size, and the time-honored mastering process is abandoned. As a result, many designs suffer in this age of convenience: many display fonts are too ornate or fragile to be displayed at a small size, and text fonts are clunky and dull when set large.

Optical size mastering isn't important only for print. When designing for the screen, similar problems arise. The stroke thickness to pixel size ratio is crucial. Delicate shapes break up and hairline serifs disappear when type is so small that fine lines shrink to less than a pixel wide. Fonts with a short x-height become hard to read if the body of the type is not rendered with enough pixels, as do unconventional or intricate shapes, closed counters and tightly spaced letters.

WHAT ARE YOU AFTER: A BASIC MEAL OR GOURMET DINING?

A typeface is more than what you see on the keyboard. In addition to the alphabet, numbers and assorted punctuation, the average font also includes a series of

“hidden” characters. They can be accessed via the shift and/or option key or through special menus. The approximately 250 glyphs found in standard typefaces are sufficient for display and basic text use. However, just as you can’t get away with serving a burger and fries at a fine restaurant, good typography requires quite a bit more. If you are considering a certain typeface for professional typesetting, make sure the following extra features are available.

For instance, small capitals (also called small caps) are a little taller than the x-height. They can be used for acronyms or to avoid putting words in all capitals, which stands out too much in running text. They can also be used for emphasis or for the first words or line of a chapter or paragraph. Petite caps are even smaller: exactly as high as the x-height. Some designers use them for typographic experiments like mixed-case setting (of lowercase and uppercase forms), also called unicase setting.

Standard typefaces mostly come with only one set of numerals, yet different styles are needed for professional typesetting. Proportional hanging figures blend in perfectly with running text; proportional lining figures match text in all caps; tabular figures are needed for setting tables and number crunching; and superscript and subscript figures are for setting fractions and scientific text. This last type of text requires not only superscript and subscript characters but special mathematical characters and several types of brackets as well.

Professional typesetting requires the presence of small caps, several sets of numerals, extended ligature sets, assorted expert glyphs, and it never hurts to have alternates and swash characters available. On the image: Capsa expert characters.

When choosing an appropriate typeface, make sure to check that the font family is complete and includes all glyphs you may need in your design. For instance, some cheap font families may not include German umlauts (ä, ö, ü) or Eszett (ß).

In fact, ligatures are needed when an overhanging part of a character bumps into the next character. Most digital fonts used to include only a limited number of "f" ligatures, which was insufficient. An extended ligature set guarantees that any unusual letter combination will have an aesthetically pleasing glyph. Some designs include quite unusual ligatures, often to add some swoosh to the type. Swoosh can also be added with

swash characters, be they initial capitals or initial and end characters.

Take a careful look at a typeface before purchasing it. To analyze a type family, carefully consider how the glyphs look at a big font size (100 to 150 points). Also look at how various members of the family will look in the size you will typically use them in. Are the italics readable? Are the small-caps clear enough? How do the bold and italics work together? Is it easy to read a word like “rococo”? Can you distinguish between the 0 (zero) and o (small o) in the word “lolo”? What about ligatures in the words “floria” and “Eigenschaft”?

SANS OR SERIF? ... OR MONOSPACE OR CURSIVE?

The ongoing debate about serif versus sans serif on the web has to do with legibility. Vociferous advocates are on both sides of the argument. As screen sizes increase and resolutions or pixel density increase, the argument that less fussy sans serif forms aid legibility loses some merit. As Jost Hochuli demonstrates in *Detail in typography*, serif fonts may be easier to read, but many people’s familiarity with and attachment to commonly used fonts on the web may make these typefaces more legible anyway. However, even typefaces designed specifically for the screen can appear very different across platforms. What is highly legible on one screen may not be on another.

Here are some key factors in choosing typefaces for legibility on the screen:

Familiar letterforms

Choose a face with customary shapes. Avoid “quirky” typefaces that break convention. Never set body copy in all capitals or all small-caps, and be careful with italics and obliques and other variations of the normal shape. For example, ascenders and descenders are important for legibility; choose a font whose ascenders for letters such as “b” and “d” rise above the x-height, and vice-versa for “p” and “q”.

Choose fonts that were designed for the screen

Most were not, either not being designed for the screen or not even optimized for it by the foundry. If a font is not hinted properly, screen performance will suffer. For example, the bar in the capital “A” can disappear at small sizes, as do serifs and other delicate glyph components.

True fonts, not synthetics

Consider the available fonts of the typeface you wish to use. Many common typefaces have only regular, bold, italic and bold italic fonts available. Whatever stylistic variants are available will have been specifically designed.

Browsers are able to synthesize fonts. A common example is synthetic italics or, more properly, obliques (regular type set at an angle without changing the letterform, to mimic a true oblique). The ubiquitous typeface of Apple, Lucida Grande by Charles Bigelow and

Kris Holmes has no italic or oblique font. That doesn't stop people from instantiating one with CSS, though; the browser synthesizes it, and to our eyes it looks awful. Small caps is another example; none of the core web fonts have a small-caps version. However, we use synthetic small-caps sometimes. The trick is to avoid synthetic fonts, but use your discretion and know when to do it and why.

Generous x-height

Again, this is the distance between the baseline and (typically) the top of a lowercase "x" (hence the name) compared to the overall body height. A generous x-height is critical for legibility, especially on the screen. Georgia and Verdana both were designed specifically for the screen; both have a generous x-height.

Comfortable letter spacing

It is possible to adjust letter spacing with CSS using the letter-spacing property; a font that already has good letter spacing is a great starting point.

Comfortable word spacing

This is the gap between words. It can also be adjusted with CSS but should be easy on the eye without adjustment.

A BEAUTIFUL FACE IS THE ONE THAT SERVES ITS PURPOSE

When all other requirements have been met, you can finally pick the typeface whose "looks" you like the most, the one you find most beautiful. But as you now know, this is actually the last step in the selection process. Understanding the importance of the preceding steps is vital. A typeface that truly serves its purpose will get you farther than one that is merely beautiful. Unless the font performs well on your operating system, comes in all the required languages, has a complete character set, is part of a suitably large family and has a design that evokes the right atmosphere and cultural connotations, your message could be misunderstood.

COMPLEMENTS WILL GET YOU EVERYWHERE

Combining fonts is an art. Unless there's a very good reason to use more, fewer is always better. Many designers use the same typefaces time after time, relying on the styles within each font for variety. Even more rely on just a handful, no matter how many they download, because those few are tried and trusted friends. So it is with the Web.

We may have hundreds of thousands of typefaces at our disposal in the years to come, but with a few well-loved and well-understood families, most designers can never go wrong.

Combining serif and sans serif is a well-worn path that can work wonders. When combining them for body text, be sure to match the x-height.

Just as opposite colors on the color wheel are complements, so it is with type. However, tread carefully. The contrast between certain fonts can be just as harsh as the contrast between blue and yellow.

Keep in mind that computer displays have much greater black/white contrast than the typical printed page. Therefore, many designers prefer off-black to pure black on white backgrounds"[1]. Likewise, it is usually more elegant to use very light gray instead of pure white on black backgrounds. To achieve quality typography with CSS, aim for the smallest effective difference and "make all visual distinctions as subtle as possible, but still clear and effective[2]."

Try alternate styles of the same face as a starting point. Use a bold as a display face for headings, small caps or caps for sub-headings, an italic for further sub-headings and a regular for body copy. Experiment with style to find the right hierarchy of elements on the page.

If you do get a little experimental with your font stacks, beware of baseline variations between fonts on different platforms. You may think Helvetica Neue and Arial would have similar baselines but they don't. The differences in viewing a page in Windows with Arial and on OS X with Helvetica can nudge the grid alignment askew.

1. Jeff Croft, "Elegant Web Typography."
2. Edward Tufte, *Visual Explanations*.

Pay Attention To Details

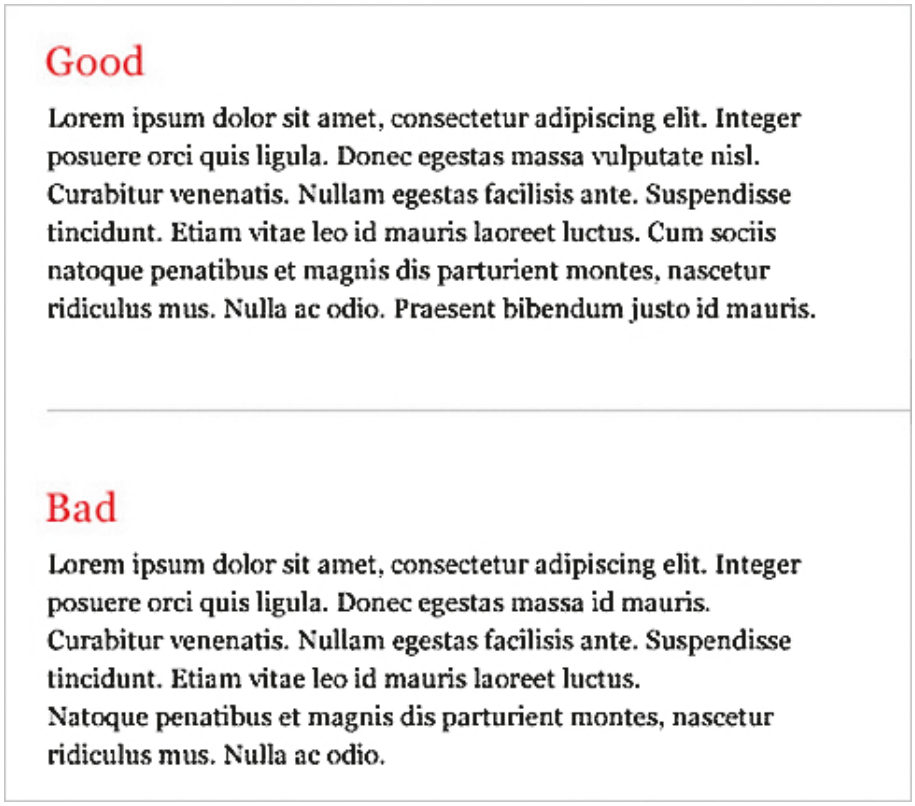
Now that we've considered best practices and practical recommendations, let's look at typography from a different perspective. To achieve beautiful, well-rounded and effective typography, one needs to create properly formatted and carefully written copy. That is, if you want to leave a great impression, you need to polish your writing and pay close attention to the smallest typographic details.

WIDOWS AND ORPHANS

A *widow* is a short line or single word at the end of a paragraph. An *orphan* is a word or short line at the beginning or end of a column that is separated from the rest of the paragraph. Widows and orphans create awkward rags, interrupt the reader's flow and impair readability. They can be avoided by adjusting the type size, leading, measure, word spacing and letter spacing or by entering manual line breaks.

CLEAN RAGS AND HYPHENATION

When setting a block of text that is not justified, be sure to keep the rag (the uneven side) balanced without any sudden "holes" or awkward shapes. A bad rag can be unsettling to the eye and distract the reader. A good rag has a "soft" unevenness, with no lines that are too long or too short.



Be sure to keep the rag (the uneven side) balanced, without any sudden "holes" or awkward shapes.

On the Web, justified text is usually more difficult to read and scan than left-aligned text. Keep in mind that justified text usually doesn't work as well with sans serif typefaces as with serif typefaces and it works best with a narrower measure.

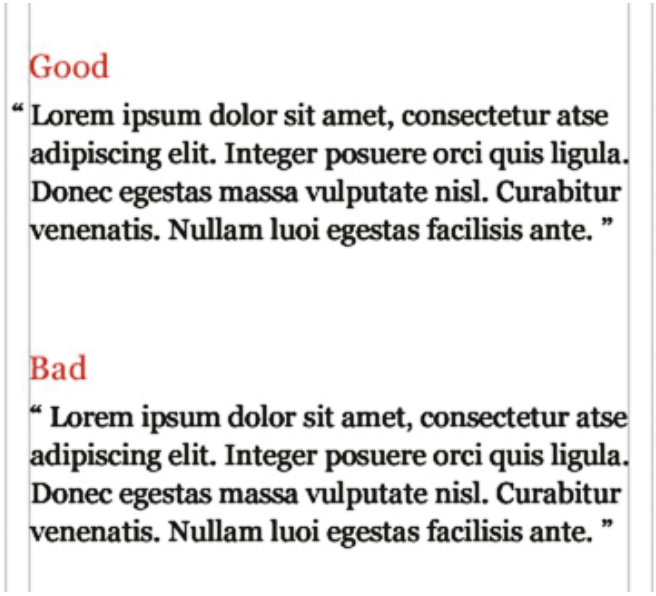
EMPHASIS

Emphasizing words without interrupting the reader is important. Italics are widely considered to be the best form of emphasis. Other common forms of emphasis are bold, caps, small caps, type size, color and different typeface. No matter which you choose, try to limit

yourself to one. Combining, say, caps, bold and italics would be disruptive and look clumsy.

HANGING PUNCTUATION

Good typographic practice is to put pull quotes, bullets and numbers outside the flow of text. Indenting bullets can disrupt the flow of text. With hanging punctuation, text and quotes are highlighted and appear more sophisticated and legible. In your CSS, make sure to hang the punctuation outside of the margin of the body text. For pull quotes, give the CSS `text-indent` property a negative value (which will depend, of course, on the font size).



An example of good and bad quote presentation. Hang the quotation marks outside of the margin of the body text so that the visual rhythm is not broken.

For the same effect with ordered or unordered lists, use the `list-style-position` property to push bullets outside the left rung.

```
ul, ol {
    list-style-position: outside;
}
```

Indented lists can also serve as calls to action as people skim the page. If you do use an indent, make sure you do it deliberately, for a specific purpose.

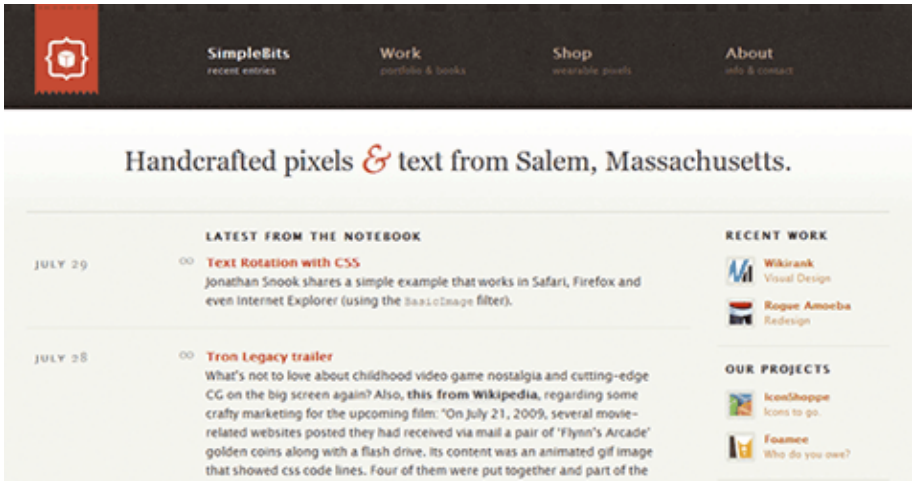
AMPERSAND WITH STYLE

The ampersand is essentially a series of curves and demonstrates nice variation from font to font. With CSS, you may want to choose a special font for it. The Simplebits article "Use the Best Available Ampersand[3]" offers an interesting and effective approach to choosing the best ampersand by setting up your font family of choice.

```
<p>pixels <span class="amp>&amp;</span> text</p>

span {
    font-family: Baskerville, Palatino, "Book
    Antiqua", serif;
    font-style: italic;
}
```

3. http://simplebits.com/notebook/2008/08/14/ampersands.html



Simplebits has a beautiful and elegant ampersand in its slogan.

DO NOT USE A HYPHEN FOR THE EM DASH

If you need to interrupt yourself, do it with an em dash (`&#8212;`) instead of a pair of hyphens (`--`). This is a top pet peeve of countless editors.

DO NOT USE DUMB QUOTES

Quote “this way” (with quotation marks that look like 66 and 99) and not "this way". Open and closed quotes are not the same. Please notice that the choice of quotes varies depending on the language in which it is used. For American English, quotes are normally surrounded by double quotation marks, while nested quotes use single quotation marks. For British English, it can be either way: doubles then singles, or singles then doubles.

In CSS, you can style the appearance of quotes using the `:lang` `pseudo`-class:

```
:lang(en-us)>q {
    quotes: "\201c" "\201d" "\2018" "\2019";
```

```
}
:lang(en-gb)>q {
    quotes: "\2018" "\2019" "\201c" "\201d";
}
```

It is highly recommendable to consider regional differences when using smart quotes, and avoid the so-called "dumb" quotes. For instance, in Arabic, Dutch and German language „this way" (99 and 66) is correct and in Russia, France and Italy guillemets – «this way» – are more common.

USE ACCENT CHARACTERS WHEN NECESSARY

Although accent characters can be difficult to type in or copy in HTML, paying attention to these non-standard characters is an important courtesy and sign of respect. Many tables listing character entities, including the one on Wikipedia[4], are available for your convenience.

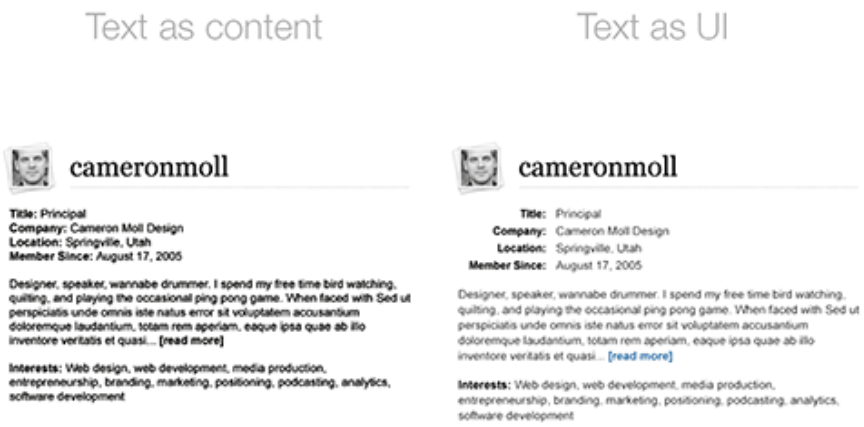
TREAT TEXT AS A USER INTERFACE

Word choice in interfaces is extremely important and can make or break the functionality of a website. The presentation of those words is equally important. Unstyled letterforms give no indication as to what users should interact with.

The image below compares text as content and text as user interface. On the left is unformatted text, and

4. https://en.wikipedia.org/wiki/List_of_XML_and_HTML_character_entity_references

on the right the text functions as a user interface. Notice the different colors and weights of the text, the ample spacing between paragraphs and lines and the links that stand out and are easy to identify.



In his presentation entitled "Nine skills that separate good from great designers[5]*," Cameron Moll discusses the importance of treating text as user interface.*

If you want your content to convey a function as well as a meaning and to help your users understand how they are supposed to achieve their goal with your text, you'll have to include subtle visual cues like these.

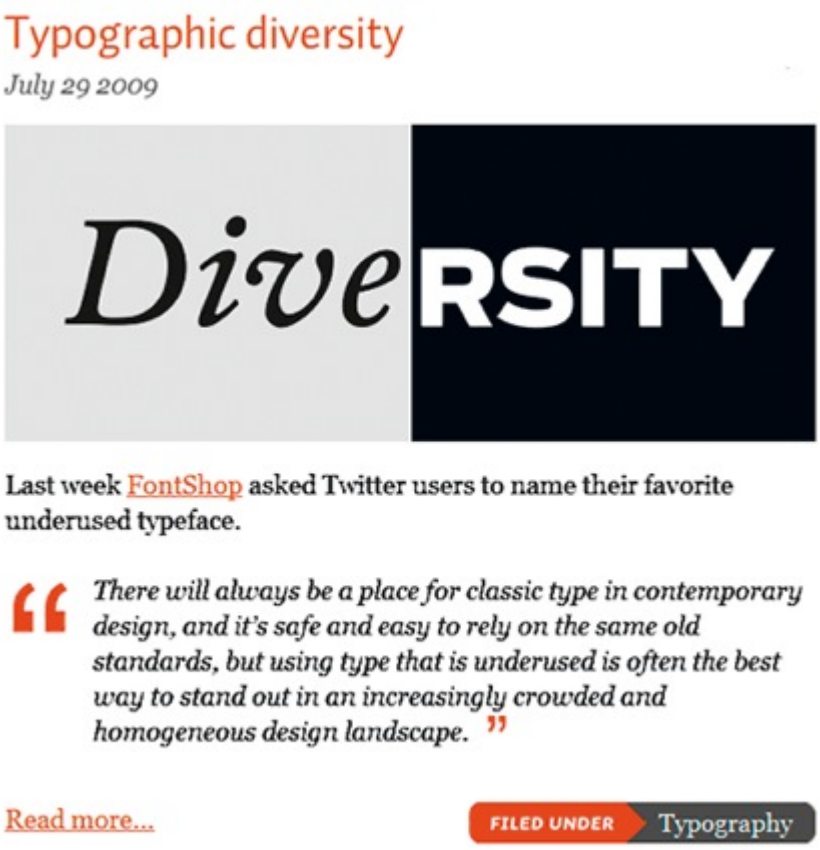
QUOTES

Quotes highlight excerpted text. They are always used for testimonials and sometimes for blog comments, but most importantly they are used in paragraphs of text. Not all quotes are the same, though. Pull quotes are

5. http://www.cameronmoll.com/archives/001266.html

short excerpts of text taken straight from the page itself. They pull a bit of text out of the flow of content and repeat it in a prominent location on the page to grab user's attention.

Like pull quotes, block quotes (or, more precisely, block quotations) are set off from the main text as distinct paragraphs or blocks. Unlike pull quotes, they contain a passage from an external source that does not appear elsewhere on the page. Block quotations are usually set within the main flow of text. "Normal" quotes cite content from other sources and are incorporated in the main flow of content rather than set apart.



A vivid block quote element on Idsgn.org.

According to HTML specifications, three elements are available to semantically mark up quotations: `<blockquote>`, `<q>` and `<cite>`. While all mark up quotes, each is meant for a different use.

- A `<blockquote>` is used for relatively long quotations; it is a large section of text set apart from the main body. This text usually comes from an external source, but can also refer to your previous article.
- `<q>` is used for short inline quotations:

```
<p>Andrew answered <q>fonts</q>, but Steven said
<q>typefaces</q></p>
```

  Although it is rarely used, it has some useful properties. For instance, you can modify the appearance of quotations inside the `<q>` element using CSS. This is helpful because languages have different quotation marks (see the section “Do not use dumb quotes” above).

- Finally, `<cite>` defines an in-line citation or reference to another source:

```
<p>But then <cite>Andrew</cite> said <q>No, I think
these fonts work better</q>.</p>
```

  Quotes, braces, lines, indentation, quote visuals, dialogue boxes, balloons – there are a number of paths a designer can take to create a beautiful and memorable quote. Design solutions vary in colors, forms and sizes. Different techniques produce different results; howev-

er, it is important that it is clear to the visitors that the quote is actually a quote, otherwise it becomes hard to keep track on the content. Also, use quotes sparingly; they should play a secondary role, supporting the main content, not the other way around. They should be a pleasant diversion, not a central design feature.

As pretty as they are, pull quotes have inherent problems in the way they are placed in the middle of HTML content. To people viewing the page in CSS-enabled browsers, all may be hunky-dory. But for those who have disabled CSS or are using screen readers, pull quotes appear slap bang in the middle of the main content. A quote suddenly appearing between two paragraphs but not connected to them will most certainly break the flow and confuse readers.

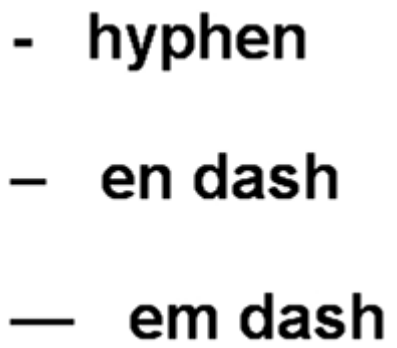
If your content has pull quotes, providing a little extra information for users who stumble on them is probably wise. In HTML, you could include a message, hidden from view with CSS, that says “Start of pull quote” before the quote and “end quote” after it. You could even have something like the “Skip navigation” link that lets users skip the pull quote and continue with the main content.

HYPHENS AND SPECIAL CHARACTERS

Finally, here we cover some of the most common typographic symbols, their uses and good practice when dealing with advanced typography and special kinds of text.

The hyphen (-) is one of the most used typographic symbols. This punctuation mark divides and joins words. The hyphen is not a minus sign (−) or a dash (–, –, —). The soft hyphen (`&#173;`) appears at the end of a line of text to indicate that a word continues on the next line. “Soft” refers to the fact that the hyphen should disappear if the entire word ends up being written on one line. The hard hyphen wraps text and joins words. “Hard” refers to the fact that the hyphen must always appear, even when the hyphenated term appears on one line.

En dash and em dash: the former is longer than a hyphen and half an em dash. It indicates a range, such as for dates, numbers, game scores and pages (2:00–3:00 p.m.), or groups compound adjectives (Meyer–Stevenson Lexicon). The HTML entity for the en dash is `&8211;`.



The hyphen, en dash and em dash.

An em dash is 1em wide. It indicates an interruption in speech (I was debugging the style sheet — wait a second; what was he saying about IE6?). The HTML entity

is `&8212;`. Note that some dashes benefit from having spaces around them; for instance, the em dash can be surrounded by a thin (` `) or hair (` `) space[6].

Curly single and double opening quotes are not the correct symbols for feet and inches. A *single prime* should be used to represent feet and minutes (`&#8242;`), and a *double prime* for inches or seconds (`&#8243;`).

To indicate missing words in a quotation or a thought that trails off, use an ellipsis (`&#8230;`) instead of three consequent periods. Three periods are not an ellipsis. This is a very common mistake because they look so alike, but semantically they are different.

The Last Word

In its essence, typography is a powerful medium that allows for precise, effective communication. On the Web, typography can be used to enhance content, turning lifeless chunks of data into vivid, elegant conversations.

But one has to thoroughly consider numerous typographic details, not only typeface and the context in which it will appear, but also measure, leading, tracking, contrast and font size. Composition aids such as grids and vertical rhythm provide a powerful frame-

6. See also Jon Tan's "Typographic Spaces Test Suite," a series of examples of different types of typographic spaces, using the core web fonts for user agent and operating system comparison.

work for creating harmonious layouts in which typography can breathe and serve its purpose.

Proper paragraph formatting, judicious use of white space and typographic hierarchy and scale can improve the structure of text, making it easier to scan and read. Besides, close attention to the quality of body copy, including punctuation marks, empty spaces and special characters, contributes to a better reading experience and has a major impact on the usability of the overall design. Because web typography is all about communication with users, you better make sure that your conversation is rich and meaningful: your readers will appreciate it. ❧

When They Click: Psychology Of Web Design

BY SUSAN WEINSCHENK

This chapter was first published in Smashing Book #2.

You might have heard this story about an elephant:

A king takes his most trusted advisors and puts a blindfold on each one. He then brings them into a dark room. They cannot see anything. The king says to them, "I have been to a far away land, and I have brought back with me something that is unlike anything you have ever known. It is called an elephant." "What is an elephant?" the advisors ask. The king says, "Feel the elephant and describe it to me." The first advisor feels a leg and says, "The elephant is a pillar." Another advisor feels the tail and says, "The elephant is a rope." The third advisor feels the belly and declares, "The elephant is a wall." And the last advisor feels the tusk and announces, "The elephant is a solid pipe." "You are all wrong and all correct at the same time," says the king, "for you are each feeling just a part of the elephant."

This story reminds me of the different views of design that are held by people of different backgrounds, education and experience. A visual designer approaches design from one point of view, the interaction designer from another and the programmer from yet another. Then there is the business owner, the information architect and on and on. Each person's own experience,

education and background influence their perception and determine which part of the elephant they focus on. Sometimes, experiencing a different part of the elephant is helpful.

I'm a psychologist by training and education. This means that I consider design in the context of the mental model of the user. Whether the design is of software, a website, a medical device, online instructions or product packaging, I can't help but see it from a psychologist's point of view. I enjoy applying what psychology research tells us about how people think, learn, play and work to design challenges. I take research and insight into the brain, the visual system, memory and motivation and extrapolate design principles from them. This chapter offers a snapshot of the psychologist's view of design.

People Are Social Animals

We seem to always be surprised when someone comes up with a new way for people to use technology or computers to interact with each other.

MySpace and Facebook were a surprise when they came out, as was Twitter. We asked ourselves, "What is this for? Why would people want to use this?" These technologies and applications were new and surprised us when they first appeared, but adoption of them was fast and broad. They become well used and a part of millions of people's daily lives in an amazingly short span of time.

We underestimate how important it is for people to be social. We forget that humans are social animals. People will use whatever is around them to be social, including technology. If you can think up a new way for people to be social using technology, then put that idea into action, because it could very well turn out to be the next big thing and earn you millions of dollars.

ALL INTERACTIONS ONLINE ARE SOCIAL INTERACTIONS

When people interact with each other, they follow rules and guidelines for social interaction. We have expectations of how social interactions will go, and if anyone violates those expectations, we get uncomfortable. The same is true of online interaction. When we go to a website or use an application, we have assumptions about how it will respond to us and what the interaction will be like. Many of these expectations mirror the expectations we have of interpersonal interaction.

If the website is not responsive or takes too long to load, it is like the person we are speaking to is not looking at us or is ignoring us. If the website asks for personal information too soon in the flow of the interaction, it's like someone getting too personal. If the website does not save our information from session to session, it's like the person does not remember that we know each other.

When you design an interface, think about the interactions you are building. Think about how they mirror

interpersonal interaction, and be sure not to violate any rules of social interaction.

Most Decisions Are Made Unconsciously

We like to think that we are rational beings and that we make decisions (such as which website to visit or which product to buy) based on a logical thought process. But the latest research shows that we make decisions largely unconsciously. Over the last 10 to 15 years, a lot of research has been done on unconscious mental processing. Functional magnetic resonance imaging (fMRI) even enables us to look at parts of the brain that are active when a person makes a decision.

Humans have evolved an efficient way of dealing with the millions of pieces of sensory information that we are bombarded with every second. Our eyes, ears, nose, sense of touch and taste buds take in huge amounts of data – the latest estimate is 40 million sensory inputs each second. But we can consciously process only about 40 of those at any one time. Most of the information we take in is processed efficiently and quickly – unconsciously. Our brain processes the data and makes decisions. Is this good or bad? Dangerous or harmless? Something I can eat? Something I can have sex with? Is this a familiar face? Is this person threatening me? Is that thing over there a snake? We are not even aware of this processing, but it affects our decision to click a "Buy Now" button, follow a link, read a paragraph or abandon a transaction.

Godiva chocolate[7] *company is lucky to have content that naturally lends itself to luscious photography.*

If you want to create a website that is engaging, that grabs the visitor's attention, encourages them to continue interacting and persuades them to take a particular action, then you have to communicate with more than just the logical part of the brain (the pre-frontal cortex). You have to also engage the midbrain (or emotional brain), where emotions are processed, and the old (or reptilian) brain, where danger, sex and food messages are processed.

Here is a summary of what to consider:

- The old brain is constantly scanning the environment for food, sex and danger, so any images of food, attrac-

7. http://www.godiva.com

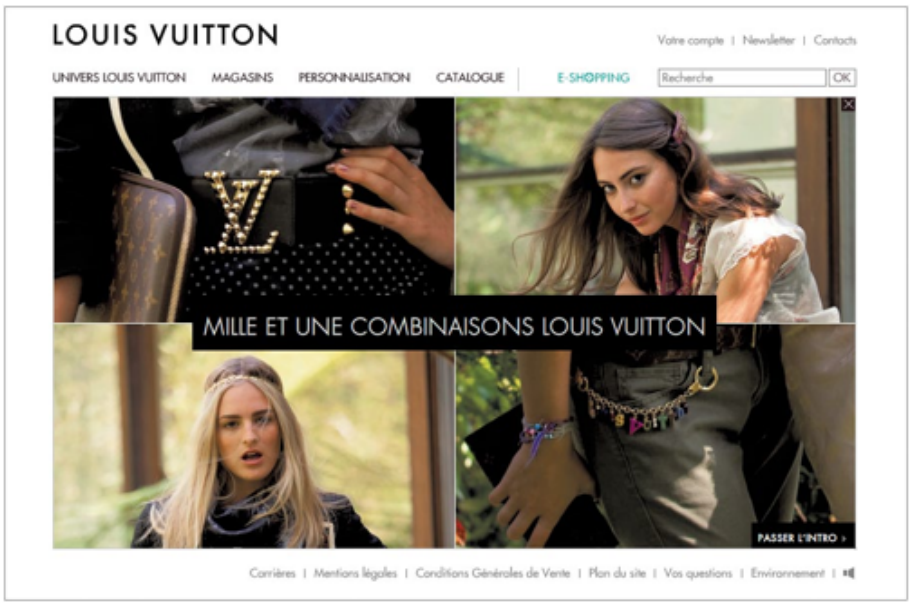
tive people or scary scenes will grab their attention, as will animation or movement.

- To hold attention for more than a few seconds, you need to engage the midbrain. You can do this with stories and pictures of people's faces, especially ones that show some kind of emotion.

The FIFA.com[8] site shows the emotions of the team winning a game.

- People process information best through stories, so use as many as you can, even short ones.
- People are naturally drawn to pictures of other people, especially people who are attractive or appear similar in dress, age and other physical characteristics.

8. http://www.fifa.com



The Louis Vuitton site[9] *shows attractive people looking right into the camera.*

- The best pictures to use are of people looking right into the camera. The picture should be large enough so that you can really see the faces.
- Although people make decisions unconsciously, they still need rational reasons to explain to themselves and others why they made the decisions they made. These rational reasons are probably not their real reasons, but having them is important, so make sure to provide compelling arguments for why they should choose your service, tool or resource.

9. http://www.louisvuitton.com

People Look To Others When Uncertain What To Decide

The tendency to look to others to decide what to do is called "social validation." Many studies have been done on the phenomenon. Latane and Darley conducted a series of studies[10] in which they set up ambiguous situations to see whether people were affected by what others around them did.

For example, they would bring someone into a room supposedly to fill out a survey on creativity. One or more actors would be in the room pretending they were also participants. Sometimes one actor would be in the room, sometimes two or more. As everyone was filling out the survey, smoke would seep into the room through an air vent. The researchers were interested in seeing whether the participant would leave the room, tell someone about the smoke or just ignore it. It wasn't clear what the smoke was or whether it was dangerous, so the situation was ambiguous.

The participant's response depended on the behavior and number of other people in the room. The more people in the room who ignored the smoke, the more the participant was likely to also. If the participant was alone, then he or she would leave the room to notify someone. If others were in the room but not reacting, then the participant would not take action either.

---

10. Latane, Bibb and Darley, John M. *The Unresponsive Bystander*. Upper Saddle River, NJ. Prentice Hall: 1970.

Online, social validation is most evident with ratings and reviews. When we are unsure what to do, we look to testimonials, ratings and reviews. The most powerful ratings and reviews share a couple of key characteristics:

- They provide more information than just a score. The more information that is available, the more people will trust and believe the review.
- They tell a story about the product or service. Stories are powerful because they speak to our midbrain.
- If you want to persuade people to take action, then include ratings, reviews and testimonials.

When a visitor comes to your website, they make decisions about what the website is about, whether they like it, whether they will stay and whether they trust the information. These decisions are made very quickly, in some cases in less than a second. The decisions are also made largely unconsciously. B.J. Fogg's theory[11] on trust and websites posits that decisions about whether to trust a website are based heavily on superficial aspects of the user interface. Here are some factors that lead to mistrust:

11. Fogg, B.J. "Prominence-Interpretation Theory: Explaining How People Assess Credibility Online." Conference on Human Factors in Computing Systems. CHI: 2003.

- Text is not aligned or orderly. Text is centered rather than left-aligned, and the screen has many distinct margins.
- Too many fonts and font sizes are used, and too much text is colored.
- The page has too much information and looks cluttered.
- The page has too many animations, especially dated amateurish ones.
- There are too many areas of color on the screen, and they are not coordinated into one color scheme.

Bottom line: if you want to appear trustworthy, then pay attention to basic principles of interface design, and avoid inconsistencies that would make users feel awkward or not take you seriously. You don't have many opportunities to gain a user's trust, so you'd better get it right the first time.

People Are Motivated By Mastery

In his book *Drive*, Daniel Pink talks about how motivating it is to have a sense of mastery.[12] People want to feel that they are growing and acquiring knowledge in a field or gaining skill in a task. In order for people to

12. Pink, Daniel. *Drive*. Riverhead: 2009.

gain a sense of mastery, they need to feel that they are making progress.

Ran Kivetz, Oleg Urminsky and Yuhuang Zheng[13] have conducted experiments on human motivation, including research on what motivates people on websites. In one of their studies involving a website for rating songs, they found that people would visit the website more frequently and rate more songs on each visit as they got closer to the reward goal on the website. This is called a "goal-gradient" effect.

The goal-gradient effect was first studied by Hull in 1934 using rats. He found that rats that were trained to run a maze to find a food reward would run faster as they got closer to the end of the maze.

The goal-gradient effect states that you will accelerate your activity as you move closer to your goal. Here are some things to keep in mind about this behavioral pattern:

- The shorter the distance to the goal, the more motivated people will be to reach it.
- You can trigger this boost of motivation with even the illusion of progress.

13. Ran Kivetz, Oleg Urminsky and Yuhuang Zheng. "The Goal-Gradient Hypothesis Resurrected: Purchase Acceleration, Illusionary Goal Progress, and Customer Retention" in Journal of Marketing Research, 39 Vol. XLIII (Feb. 2006): pp. 39–58.

- Motivation plummets right after the goal is reached. You are most at risk of losing a customer immediately after the reward is attained.

Mastery is one reason why gaming is so powerful. People become addicted to trying to master a level. While your website might not be a game, look for ways to build the concept of mastery into it. Enabling visitors to set goals and chart their progress keeps them engaged and motivated to use your website.

People Are Easily Distracted

Even when performing an important task, we tend to easily get distracted and end up clicking to various websites, instead of following the linear path to our objective. In general, we get easily distracted on websites for three reasons:

The old brain. Because our old brain is constantly scanning the environment for food, sex and danger, we can get distracted when any of those things show up.

Peripheral vision. Basically, we have two types of vision: central and peripheral. Central vision is the one we use when we look directly at something and see the details. Peripheral vision is the rest of the visual field – what we do not directly look at. The evolutionary theory goes that thousands of years ago, the people who were able to see the lion coming as they were sharpening their flint or gazing at the clouds survived to pass

on their genes. So, peripheral vision has always been important.

Multi-tasking is a myth. Many people believe that they're able to multi-task, but the research is clear that we actually can't. Many years of psychology research show that people can attend to only one task at a time. We can alternately talk, read, type and listen, and we're pretty quick at switching between them, so we think we are multi-tasking, but in reality we are not.

A study by Ira Hyman and others[14] shows that people who talk on their cell phones while walking, run into people more often and notice less of what is around them. The researchers planted an actor in a clown suit riding a unicycle. The people talking on their cell phones were much less likely to notice or remember the clown. The fact that we are so easily distracted can be both good and bad. If you want to draw attention to a certain part of a web page, that is easy to do, since it is easy to distract people.

But getting people to concentrate on one part of the page can be hard. Also, drawing people's attention to your website is easy, but having them focus on one section of the page and keeping them from getting distracted while they're there is hard. Keep these points in mind:

14. Ira E. Hyman Jr., S. Matthew Boss, Breanne M. Wise, Kira E. McKenzie, Jenna M. Caggiano "Did You See the Unicycling Clown? Inattentional Blindness While Walking and Talking on a Cell Phone." in Applied Cognitive Psychology. December 2009.

- Anything that moves will grab attention, including video, animations and scrolling banners – not that you should use scrolling banners much (and starting them automatically is especially bad practice), but it's worth knowing nevertheless.
- Photos of people, food, sex and dangerous scenes are distracting.
- Anything that blinks is annoying because it constantly catches the peripheral vision.

Use distractions to grab attention, but eliminate them where you want people to stay focused.

People Are Inherently Lazy

People do the least amount of work possible to get a task done. Over eons of evolution, humans have learned that they survive longer and live better if they conserve energy. We've learned that we have to spend some energy to acquire certain resources, but beyond that, we know that we waste energy if we spend too much time running around trying to get stuff.

SATISFY + SUFFICE = SATISFICE

For most activities most of the time, humans operate on a principle called "satisficing." Herbert Simon coined the term. It was originally used to describe a decision-making strategy in which a person picks the option that is adequate rather than optimal. The rationale is that

not only is the cost of completely analyzing all options not worth the investment, but such a thorough analysis might be impossible.

According to Simon, we often don't possess the cognitive faculties necessary to weigh all options. So, deciding based on what's good enough is better than trying to find the optimal solution. What are the implications for those of us who design websites, software, products or even design surveys? Satisficing leads us to some interesting guidelines:

Design websites for scanning, not reading. In his book *Don't Make Me Think*, Steve Krug applies the idea of satisficing to the behavior of a website's visitors.[15] The owner hopes they read the whole page but knows that "what they actually do most of the time (if we're lucky) is glance at each new page, scan some of the text, and click on the first link that catches their interest or vaguely resembles the thing they're looking for. There are usually large parts of the page that they don't even look at."

Assume that people look for shortcuts. People look for ways to do things faster and in fewer steps, especially repetitive tasks. But if a shortcut is too hard to figure out, then they will keep doing it the old way. As odd as this seems, it comes down to perceived effort. If finding a shortcut seems like too much work, then people will stick with their old habits (being satisficed with their satisficing).

---

15. Krug, Steve. *Don't Make Me Think*. New Riders: 2005.

Provide defaults. Defaults reduce the amount of effort required of visitors. Automatically filling in the name and address fields on a web form, for example, means less work for the user. The risk is that people often don't notice defaults and so might unwittingly confirm a wrong one. Again, it comes down to the effort required. If changing a default would take considerable effort, then you might want to reconsider providing it.

People are wired to get things done with as little work as possible, and yet they don't want to be bored either. Accounting for these two impulses is the key to creating an engaging website.

People Have Mental Models About How Things Work

Imagine that you've never seen a Kindle, and I've just handed you one and told you that you can read books on it. Before you turn it on to use it, you have a model in your head of what reading a book on the Kindle is like.

You have assumptions about what the book will look like on the screen, what things you will be able to do and how you will do them – things like turning pages or adding a bookmark. You have a mental model of reading a book on the Kindle, even if you've never done it before.

That mental model is determined by a lot of things. Someone who has used a Kindle before, someone else who hasn't and yet another who has never even heard

of it will all have different mental models of the experience.

If you've been reading books on the iPad, then your mental model will be different from that of someone who has never read an eBook. And once you do get a Kindle and have read a couple of books on it, your mental model will adjust to the experience.

MENTAL MODELS ARE NOT NEW

Mental models have been defined in many ways for at least 25 years. In her 1986 study, Susan Carey[16] defines mental models as such:

> *"A mental model represents a person's thought process for how something works (i.e. a person's understanding of the surrounding world). Mental models are based on incomplete facts, past experiences, and even intuitive perceptions. They help shape actions and behavior, influence what people pay attention to in complicated situations, and define how people approach and solve problems."*

In the context of user interface design, a mental model is a representation that a user has in their mind of a product. Here's what we know about mental models:

16. Carey, Susan. "Cognitive Science and Science Education" in American Psychologist. v41/ n10. 1986. pp. 1123–30.

- Users create mental models very quickly, often before even using the product.
- Mental models are subject to change.
- Users base their mental models on prior experience with similar products, their assumptions, other people's observations and direct experience with the product.

People use mental models to predict what a product will do or what they should do with it.

MENTAL MODELS VS. CONCEPTUAL MODELS

To understand why mental models are so important to designing user interfaces, we have to also understand conceptual models. A conceptual model is the actual model of a user interface itself.

Going back to the iPad, you have a mental model of what reading a book on it will be like, how it will work and what you can do with it. But when you sit down with the iPad, the device presents the conceptual model of the book app. There will be screens and buttons, and things will move around. The conceptual model is the interface itself, and you are interacting with it by using the product.

So, why should we bother distinguishing between mental models and conceptual models? Here's why: every choice you make with the user experience constitutes either a match or mismatch between the user's mental model and the product's conceptual model. Here are some examples:

- If the product's conceptual model doesn't match the user's mental model, then the user will find the product hard to learn and use.
- If the designer hasn't taken the user's mental model into account, then the product will likely be hard to learn and use.
- If a product has multiple user groups (e.g. people who have used a Kindle before, people who have never read an eBook, etc.) and the conceptual model is designed to match just one mental model, then the other groups will find the device hard to learn and use.
- If the conceptual model is not deliberate but merely reflects the underlying hardware or software or database, then it will not match the users' mental model very well, and users will find the device hard to learn and use.

WHAT TO DO IF YOU NEED TO CHANGE THE MENTAL MODEL

Sometimes you don't want to match the user's mental model. You might be designing something really different, a breakthrough product, and decide that you want the user to alter their mental model to fit your new conceptual model, rather than the other way around. You know, for example, that people who have read only paper books will not have an accurate mental model of reading books on the Kindle.

In this case, you still need to understand the user's mental model so that you know how it diverges, and then you need to train the user on the model itself (not just on the product). Training is crucial because it sets or resets the user's mental model before they start using the product. It doesn't have to be a laborious multi-day training experience. You can alter a mental model in a few minutes. For example, you could present a short video before the Kindle arrives at their door. The goal is to alter the user's mental model to fit the conceptual model of the product.

IT'S ALL ABOUT MENTAL AND CONCEPTUAL MODELS

Mental models and conceptual models are powerful for the work that interface designers do. You could even go so far as to say that almost everything we do in a user-centered design process has to do with one of the following:

1. Understanding the user's mental model (through task analysis, observation, interviews, etc.)
2. Designing a conceptual model to fit the user's mental model (through interface design, iteration, validation testing, etc.)

The secret to designing an intuitive user experience is to match the conceptual model of your product as closely as possible to the mental model of your users. If you

get that right, then you will have created a positive and useful user experience.

People Make Mistakes

People are wrong a lot, and they don't like being wrong. How a website handles error determines whether people stay and come back. You need to recognize that people make errors and then anticipate what those errors might be and how best to deal with them. Avoid giving error messages that make no sense.

Creating error messages probably requires the least time and energy of all sections of a website, and maybe that is appropriate. After all, the best error message is no error message (meaning that the system is designed so that no one can take a wrong step). But when something goes wrong, it is important that people know what to do.

The reality is that something always goes wrong. Everyone makes mistakes: users make mistakes when typing, companies make mistakes by releasing buggy software, and designers create unusable websites because they don't understand what users need. But you can mitigate error by following these tips:

- Think ahead about what mistakes will likely occur. Figure out as much as possible what kinds of mistakes people will make, and then adjust your design accordingly before releasing it.

- Create a prototype, and then get real people to use it so that you see what errors they make. In other words, perform usability testing.
- Write error messages in plain language. They should communicate the following:

    1. That an error has been made.
    2. What the error is.
    3. How the user can correct it.
    4. Where to go for more help.

- Use the active voice, and be direct. Weak: “Before the invoice can be paid, it is necessary that the payment date be set earlier than the invoice creation date.” Better: “Enter an invoice payment date that falls before the invoice creation date.”

    Creating a system that guarantees that no errors will be made is difficult… in fact, impossible. Look at Three Mile Island, Chernobyl and British Petroleum. The more costly the potential errors, the more you need to do to avoid them. The more you need to do to avoid them, the more expensive will be the system. If it is critical that people not make mistakes (for example, if you’re dealing with a nuclear power plant, oil rig or medical device), then be prepared: you will have to test two to three times more than usual, and training will have to be two to three times more intensive. Trying to

design a fail-safe system is expensive. And you'll never fully succeed. It's just the way humans are built. We make mistakes.

Looking Is Not Necessarily Seeing

When we design a website, we pay a lot of attention to what people will be looking at. The web is, ultimately, a visual medium. We assume that visitors will spend time looking at various elements of the website, and we make a lot of assumptions about what that entails. One common assumption turns out to be unfounded: that someone actually "sees" what they're looking at.

Two interesting lines of research point out the fallacy in assuming that people see what they look at. One compares peripheral and central vision, and the other addresses "inattentional blindness."

CENTRAL VS. PERIPHERAL VISION

We've already gone over our two types of vision, central and peripheral. Research from Kansas State University shows that we rely on peripheral vision to understand the world more than we previously thought.[17] We seem to get the "gist" of the scene before us from our peripheral vision. The researchers showed people photographs of common scenes; for example, a kitchen or

17. Adam M. Larson and Lester C. Loschky, "The Contributions of Central Versus Peripheral Vision to Scene Gist Recognition," Journal of Vision, 2009: 9 (10).

living room. In some of the photographs, the periphery was obscured; in others, the center was obscured. The images were shown rapidly, and then the participants were asked to describe what they saw. They found that if the center of the photo was obscured, people could still identify the scene. But when the periphery was obscured, they couldn't tell whether it was a living room or kitchen.

INATTENTIONAL BLINDNESS

In their book, *The Invisible Gorilla*, Chabris and Simons present their research on what is called inattentional blindness. Through many studies, they show that people often don't see what is right in front of them. We are so bombarded with visual stimuli that our brains have developed the capacity to automatically and unconsciously decide what we need to pay attention to. The eye takes in much more than the brain brings to the level of awareness.

If peripheral vision is in some ways more important than central vision, and if we are able to look at objects without really registering them, what are the implications for website design? Here are some:

- Just because someone has looked at a page element, doesn't mean they paid attention to it.
- Just because it is on the page, doesn't mean everyone sees it.

- Although eye-tracking technology provides interesting data, it tells you primarily where people looked with their central vision. Because it does not track peripheral vision, and because looking is not really seeing, don't make crucial design decisions based on eye tracking data alone.

Summary

Whatever type of website you are designing, someone unlike you will eventually use it. You have your own perception of the elephant. If you want the website to be useful, interesting and relevant to other people, then broaden your view of the elephant and incorporate psychology into your decisions. Here are the 10 principles about people to remember:

- People are social animals.
- Most decisions are made unconsciously.
- People look to others when they are uncertain what to do.
- People decide in a split second whether they trust your website.
- People are motivated by a desire for mastery.
- People are easily distracted.
- People are inherently lazy.
- Everyone has a mental model of how something works.

- People make mistakes.
- Looking is not the same as seeing. ❧

Redesigning With Personality

BY AARRON WALTER

The following is an excerpt from the Smashing Book #3 chapter "Redesigning With Personality."

Redesigning a website can be the seven-layer taco dip of hell. You've searched for inspiration on dozens of websites, captured screenshots, jotted down notes, consulted friends and colleagues, maybe even interviewed users. But despite your due diligence, your vision for the new website remains unclear.

I feel your pain, my friend. I have been there many times. A redesign brings with it the pressure to innovate, to reimagine, to make a better version of the website so that it lasts for years to come. It can be paralyzing.

Whether the website is for a client or for yourself, if you're struggling to find your way, it's probably because you are starting from the wrong place. The inspiration you seek is not where you think it is. It's not in a blog post entitled "25 Amazingly Beautiful Websites." It's not in your Twitter stream, nor on Facebook. It's not even on the web. It's right there on your seat. It's you.

Just for a moment, stop thinking about HTML semantics, CSS magic and jQuery tricks. Instead, ask yourself, "Who am I, and what do I want to say?" What do you stand for, what's important to you, and who are

you speaking to? Let's make the answers to these questions the trailhead of your redesign journey.

We web designers have many tantalizing tools at our fingertips, and because the web is a large community centered on sharing, new ideas and fancy techniques enter our field of vision daily. But in this chapter, I would like to turn your gaze from those shiny objects and focus it on what we're really trying to do with our medium. Our true aim is to communicate clearly and to create human connections.

We achieve that goal not by collecting bells and whistles for our next project, but by discovering who we are and what our message is. The interfaces we design are not walls upon which our users click and tap. They are windows through which we show the world who we really are. As we will see in the principles and examples to come, sharing our personality can help us create lasting relationships with the people who use our websites, and it can improve the bottom line of our business.

Personality will set your brand apart from competitors and help you connect with a passionate audience. Making personality central to the ethos of your redesign might sound scary, especially if you're working with a big corporation accustomed to speaking like the Borg. But even the biggest corporations can communicate with a human voice.

Products Are People, Too

Personality determines how we express emotions and the degree to which we do it. It's the framework within which we share jokes, select our circle of friends and even find a mate. It is at the heart of all human interaction.

Steven Pinker, Johnstone Family Professor in the Department of Psychology at Harvard University, points out in his bestselling book *How the Mind Works*[18], "Much of the variation in personality — about fifty percent — has genetic causes." That's right: the moment you enter this world, half of your personality is already predetermined.

The other half is primarily shaped by social and cultural influences. Only about 5% of your personality is influenced by your parents' nurturing. As a dad, I find this painfully depressing. The fact that so much of our personality is genetic indicates the extent to which it shapes our lives and ensures our survival.

So what if the websites, products and services we design could be imbued with personality? Personality is a natural interface that is familiar to humans. We already know how to respond to people we meet based on cues from their tone of voice, language, appearance and posture. We process this information subconsciously and behave accordingly. If we detect courteousness and trustworthiness in a person, then we might share more

18. Steven Pinker. *How the Mind Works*, W. W. Norton & Company, Reissue edition, 2009.

of ourselves and linger in conversation. If a person is rude or suspect, we are likely to make excuses for a sudden departure.

The cues we naturally take from people's personalities can also come through in design. Color, type, imagery, copy and interaction patterns can all serve as channels for a personality. Just as our personality influences the behavior of the people around us, personality in design can shape the behavior of visitors to our websites.

There are four key benefits to expressing your personality in design:

1. In a crowded market, personality helps distinguish you from competitors.
2. Personality elicits an emotional response from the audience that encourages long-term memory of your brand.
3. Personality attracts those who get you and deters those who don't.
4. Personality impassions users, who will become your most powerful marketing channel.

Let's look at each of these in detail.

ONE OF THESE WEBSITES IS NOT LIKE THE OTHERS

No matter what kind of website you're publishing, dozens of others like it are on the web. Put yourself in

your audience's position. How will they distinguish your website from all of the others? What makes yours different? The travel-booking company Hipmunk[19] carefully considered these questions before launching into a very competitive marketplace. Travelocity, Orbitz, Expedia and several others have had a tight grip on online travel booking for some time, but Hipmunk has managed to stand out from the crowd.

Visit a few well-known travel websites and you'll see common traits among them that express something about their personalities. Advertisements for last-minute travel deals and money-back guarantees litter their home pages, each distracting users from the central goal of the page, which is to get them to book a flight. Each of these elements is asking something of users. "Gimme, gimme!" they scream.

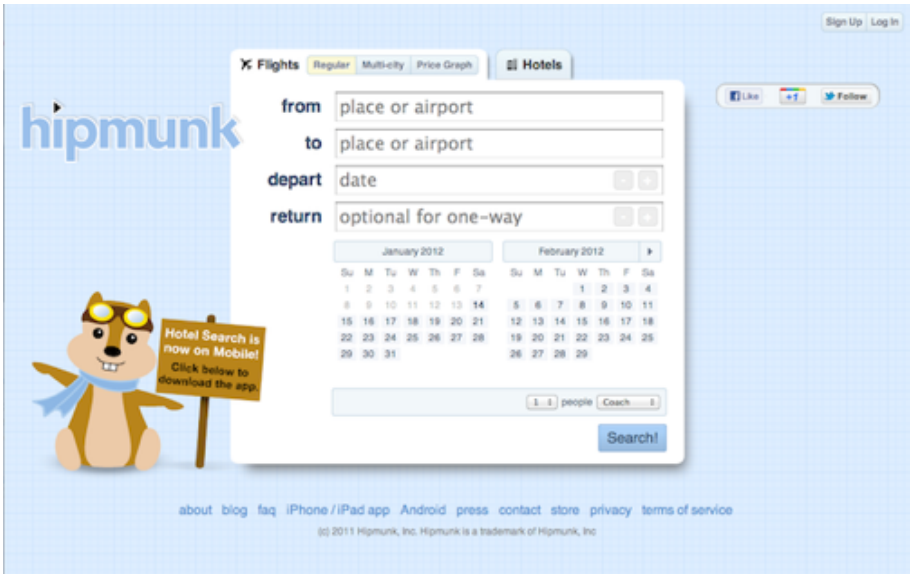
I hate making travel plans because it's such a stressful experience. Coordinating schedules, figuring out time constraints, paying high fees for basic services, and fearing the airline will once again screw something up — all leaving me frazzled and gun-shy.

This emotional state is not helped by the attention-assaulting design of most travel websites. A calm, focused personality is the best medicine for a stressful situation. That is exactly what Hipmunk offers on its website. Its home page is squarely focused on one thing: choosing a flight. A cheery chipmunk adds much-

19. http://smashed.by/hipmunk

needed levity to an otherwise stressful interaction. No discount offers distract users or add to their stress.

Hipmunk's search results are equally as focused, showing an empathy for users that is unseen in any competitor. Instead of simply listing flights by time and airline, it presents an "Agony" index, showing users which flights will be the most painful. Flights with early departures, late arrivals or long layovers rank higher in the Agony index. One could argue that this feature is simply a clever design pattern, but something more is going on. The Agony index literally communicates to users that Hipmunk is sympathetic: it feels your pain.



Hipmunk's cheery personality and focus on one task are the perfect remedy for the stressed-out emotional state of so many users of travel-booking services.

Very few websites possess these traits. Just as an act of kindness on the street says something about the individual who extends it, sympathetic interaction design

can convey a designer's compassion for their audience. It's the sort of thing users will not soon forget.

I Remember

Emotional experiences make a profound imprint on our long-term memory. Both the generation of emotion and the recording of memories happen in the limbic system, a collection of glands and structures under all of the folded gray matter of the brain. There is a good reason why the limbic system unites these essential functions. The brain couples emotion and long-term memory because, otherwise, humans would be doomed to repeat negative experiences and would not be able to consciously repeat positive experiences. As John Medina explains in his book *Brain Rules: 12 Principles of Surviving and Thriving at Work, Home and School*[20], our brains take note of emotionally charged events:

> *"The amygdala is chock-full of the neurotransmitter dopamine, and it uses dopamine the way an office assistant uses Post-It notes. When the brain detects an emotionally charged event, the amygdala releases dopamine into the system. Because dopamine greatly aids memory and information processing, you could say the Post-It note reads 'Remember this!' Getting the brain to put a chemical Post-It note on a given piece of information means that information is going to be*

20. John Medina. *Brain Rules: 12 Principles for Surviving and Thriving at Work, Home, and School*, Pear Press, Reprint edition, 2009.

more robustly processed. It is what every teacher, parent, and ad executive wants."

Experiences that lack an emotional charge tend to fade from memory. Thus, conservative, familiar designs are likely to be forgotten. Medina eloquently sums up the main reason why we designers should employ personality and emotion in design: "The brain doesn't pay attention to boring things." When we design with personality, we are building a framework through which emotional experiences will remain in the memories of our users.

I'll be honest: there is some risk in designing with personality. Not everyone will like the result. But if you design to please everyone, you will please no one. As we will see in the next section, an interesting dichotomy of positive and negative emotions is elicited by the expression of personality.

I LOVE YOU, I HATE YOU

It's OK if some people hate your redesign and do not connect with the personality you're sharing. That's a sign that you are indeed engaging with your audience on an emotional level. Disdain is always better than apathy.

Showing personality in your design always carries some risk. Some people will feel very connected to it, while others will be turned off. When you share a bit of yourself with the world, someone is not going to like you. But that's fine. The people who are turned off by

your personality are not the people you want to court. They are the ones who would cause the most problems in your product support queue or who would constantly insist that you change your product into something it isn't.

If you're a freelancer or agency on the hunt for new projects, you can ward off clients from hell by expressing your personality on your website. The people who are turned off by your website will not want to work with you — and I can tell you from experience, that's not a bad thing!

Even the design process we typically follow for a project reminds us that we are not designing for everyone. The reason we do user research is to find out who we're actually designing for. If the goal was to design for everyone, research would be unnecessary.

In our personal lives, we have all encountered individuals with whom we just don't see eye to eye. Although painful to accept, some people not liking our personality is perfectly fine. It is a fool's game to try to cater to the desires of every person we encounter. The best we can do is be ourselves and trust that some people on this planet will accept us as we are. (OMG! Did we just sneak a life lesson into a web design book?)

This lesson holds true with design. Personality will help you filter the audience down to those with whom you share common values, interests and goals. These folks are your passionate users. They are the ones to cater to, and they will express their love for your brand openly.

As the user experience lead at MailChimp, I've seen this first hand. The humor, bonmots and good times I experience every day with my colleagues is visible in the copy, illustrations and interaction design of the stuff we make. The quips of Freddie von Chimpenheimer IV, our chimp mascot, that sit atop each page of the app are collected from people in the company, providing a snapshot of our sense of humor. When you interact with MailChimp, you interact with the people who make it.

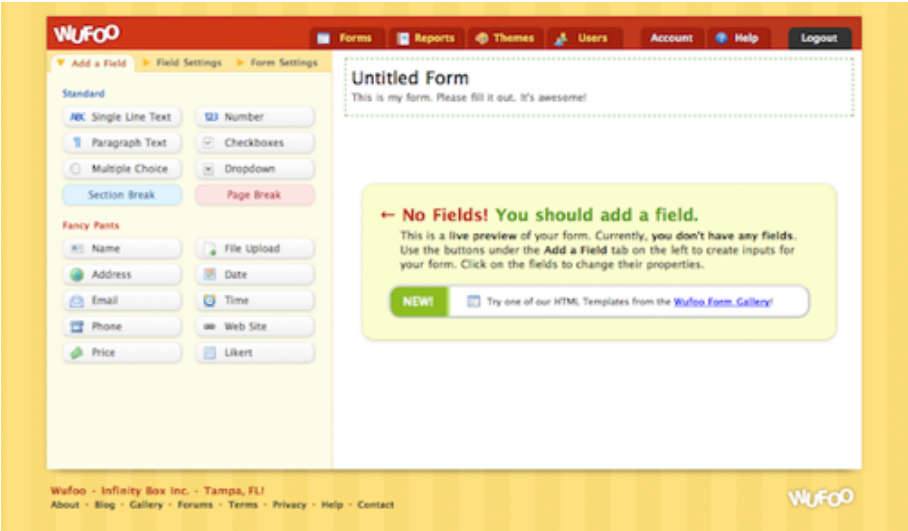
We have heard from a few people that the humor woven into the interface is distracting or annoying. When you put your personality out there, that kind of feedback can sting.

We were concerned that these perceptions were widely held by our users, so we did some research. We created an option called Party Pooper mode, which disables Freddie's jokes and buttons up the informal copy throughout the app. After tracking its usage, we discovered that only 0.007% of our customers actually turned off our personality. The lesson learned was that a bit of criticism is no reason to stop being yourself.

We have found that expressing our personality in the things we make has overwhelmingly greater benefit than risk. It distinguishes our brand from competitors; it helps our customers remember their experiences with us; and it makes many of our customers fall in love with our products. Oh, and it doesn't exactly hurt your marketing budget either, as we will see in the next section.

YOUR USERS HAVE A WAY WITH WORDS

Personality in design will help spread the word about your website, no ad campaign required. Wufoo[21], creator of a web app that makes it easy to build web forms and to manage the data they collect, eliminated its marketing budget entirely when it discovered that users market the app for it.



Wufoo's unique approach to designing a business-focused web app has elicited admiration and joy in customers, who are anxious to tell the world about their positive experiences.

Tweets, blog posts and word of mouth brought in new users more effectively than banner ads. Its youthful design, witty copy and occasional rogue dinosaur set Wufoo in stark contrast to its competitors, which are much more reluctant to express personality. When using Wu-

21. http://smashed.by/wufoo

foo, we can easily see the personalities of the people who made it.

Having a product that is beloved by users and that grows constantly by word of mouth made Wufoo an attractive acquisition for SurveyMonkey[22], which purchased it in 2011 for an estimated $35 million.

PERSONALITY: NOT JUST FOR DESIGNERS ANYMORE

Respected user experience designers such as Stephen Anderson and Andy Budd have been thinking and talking about the connections between personality, emotion and design for a few years now. The topic has gained traction with designers. But we are not the only ones who recognize the power of personality in a crowded marketplace. The people who fund startups are getting it, too.

On his blog, Fred Wilson, venture capitalist and principle of Union Square Ventures, advocates for clarity of voice and personality in product design, suggesting that the two are "critical to building a successful product."[23]

Dave McClure, angel investor and founder of the business incubator 500 Startups, is of the same mind. He sees personality and emotion as important factors in the success of a product. In his talk at the Warm Gun conference in 2011, McClure told designers and entre-

22. http://smashed.by/monkey
23. Wilson, Fred. "Minimum Viable Personality."

preneurs, "Be yourself, your super-self. What can you be authentic about and comfortable with? Find an essence that you think is you and amplify it. Find a feeling or emotion that you can sustain."[24]

Business news outlets are exploring the role of personality in design, too. Forbes contributor Anthony Wing Kosner attests that personality can make a website more memorable:

> *"Why are some companies' websites more memorable than others? On the surface, it might seem to have to do with originality, visual impact and branding. But what if I were to tell you that the most important factor is how a site makes a visitor feel?"*[25]

Wilson, McClure and Kosner are all describing something we saw earlier in this chapter. GE made its giant corporation feel more human by sharing the individual personalities of its workforce. It showed us the passion and pride it has for its craft, and we felt it.

The informal personality of Hipmunk's website allays the stress and negativity that so many travellers feel when booking a flight. The Agony index literally ranks results based on emotional response, which not only helps customers make a decision, but makes them grateful for Hipmunk's empathy.

And Wufoo eliminated its entire marketing budget after realizing that passionate users were spreading the

24. Wroblewski, Luke. "Warm Gun: Designing for Emotion."
25. Forbes, "Why Does Emotional Design Work on the Web."

word for it. The personality in its app brightens the day of thousands of workers who carry out the data-collection orders of their corporate overlords while festering in gray cubicles. The levity, color and personality of Wufoo are to users as water is to a desert wanderer.

Once upon a time, personality and emotion in design were a novelty of small-scale websites powered by one or two creative individuals. Today, we are seeing personality being carefully infused into the websites of many brands, small, gigantic and everything in between. Designers are not the only ones who see the value in making the user experience more human; investors now recognize that personality is a key factor in the success of a product.

However, starting a redesign by thinking about personality can be nebulous. Clearly defining the traits of your personality before jumping into Photoshop or Illustrator would be more helpful. That is exactly what a design persona is for.

DEFINING YOUR PERSONALITY WITH A DESIGN PERSONA

A user persona is like a dossier of your archetypal user. It answers the question, "Who are they?" A design persona flips that question on its head, asking, "Who are we?" Both user and design personas set the parameters for the design process. They help us overcome blank-canvas syndrome.

With so many possibilities, where do we start? By understanding both your users and yourself, the options are no longer vast, and the direction is clearer.

Think about it: if your website were a person, who would it be? Would the person be a serious, buttoned-up, all-business type, yet trustworthy and capable? Or a wisecracking buddy who makes mundane tasks fun? A design persona is a document that outlines the key traits of the personality you wish to convey in a design. We'll look at a real-world example of a design persona momentarily, but let's first look at the structure of the document.

A design persona has nine parts:

1. **Brand name**
   The name of your company or product.

2. **Overview**
   A short overview of your brand's personality. What makes it unique?

3. **Personality image**
   This is an actual picture of a person who embodies the traits you wish to convey. This makes the personality less abstract. Pick a famous person or someone with whom your team is familiar. If your brand already has a mascot or representative that does this, use it. Describe the attributes of the mascot that communicate the brand's personality.

4. **Brand traits**
   List five to seven traits that best describe your brand, along with one trait you want to avoid. This will help those who design and write the website to construct a consistent personality, while avoiding traits that would take your brand in the wrong direction.

5. **Personality map**
   We can map this personality on a graph. The x axis ranges from unfriendly to friendly, and the y axis ranges from submissive to dominant.[26]

6. **Voice**
   If your brand could talk, how would it speak? What would it say? Would it speak in a folksy vernacular or a refined style? Describe the particular aspects of your brand's voice and how it might change in various situations. People change their language and tone to fit the situation, and so should your brand.

7. **Copy examples**
   Provide examples of copy that might be used in different scenarios on your website. This will help the writers understand how the design should communicate.

8. **Visual lexicon**
   If you are creating this document for yourself as the designer or for a design team, develop a visual lexicon

---

26. To learn more about personality mapping and the research behind it, see "Emotional Design With A.C.T.: Part 1" by Trevor van Gorp on Boxes and Arrow.

that summarizes the colors, typography and visual style of your brand's personality. You can be general in concept or include a mood board.

9. **Engagement methods**
   Describe the methods by which you might emotionally engage users in order to create a memorable experience. Stephen Anderson's Mental Notes card deck[27] is a handy collection of such methods.

CREATING YOUR DESIGN PERSONA

The process of creating a design persona is as valuable as the document itself. When you stop to consider the traits you want in the design, you gain clarity of what you wouldn't have had, had you jumped straight into your favorite design app. Defining a personality through a team brainstorming session will help the writers, designers, information architects and developers think about your website as a person and recognize the boundaries they are working within.

To get started on your design persona, download the template and sample files[28] first. If you're working on a team, gather everyone together with some snacks and a whiteboard to work through the initial ideas of the persona. It'll be fun — and a cooler full of adult beverages wouldn't hurt either.

27. http://smashed.by/notes
28. http://smashed.by/mailchimp

Start the discussion by asking, “What seven words best describe who we are?” Be honest. You might hear some traits mentioned that you’re not proud of. Now tweak the question: “What seven words best describe who we hope to become with this redesign?” Redesigns are inherently aspirational, and being specific about your aspirations early on is helpful.

Now look for overlap between the two lists. Discuss how you might go about transforming the traits you don’t like into some of the aspirational traits. If the remaining list contains more than seven traits, continue whittling it down to seven or fewer, because a personality can become diluted and insincere if you try to be everything to everyone.

With the final list of traits scribbled on the whiteboard, reflect on the boundaries of each trait. What don’t you want your personality to become? For instance, if you have listed “fun” as a trait, that can mean a lot of things. Is it fun like Tickle Me Elmo, or fun like driving a Ferrari California along the winding Blue Ridge Parkway? Boundaries add clarity to a design persona and will help you see when a line has been crossed during the project.

To get a better sense of how traits and boundaries work, let’s look at the ones we defined for MailChimp’s persona:

1. Fun but not childish,
2. Funny but not goofy,

3. Powerful but not complicated,
4. Hip but not alienating,
5. Easy but not simplistic,
6. Trustworthy but not stodgy,
7. Informal but not sloppy.

A little list like this one is a value system. It tells you who you are, guides your voice and helps shape the audience's perception of your brand.

Now that you know what personality traits you want and don't want, can you think of a person — either a celebrity or someone you know — who could serve as a common reference point for you and your team? Putting a face to these traits will make it even easier during the design process to answer the question, "What would my persona do in this situation?"

Write an overview of this personality to flesh it out further. Describe the voice of the personality, and write out examples of copy to illustrate. How would the personality manifest itself in color, typography and visual style? At the end of this exercise, you will have learned a lot about your starting point for the design.

When creating a personality for your website, keep one important rule in mind. Make the personality an honest reflection of the company you're working for or yourself. You will have a hard time staying true to the design persona if the personality is a stretch. In all of the examples we saw earlier in this chapter, the person-

alities of the people who created the websites were evident. They were not idealistic concoctions.

We can tell in an instant when a person is pretending to be someone they aren't, and the same holds true in design. The whole point of conveying personality in design is to share more of ourselves so that we can forge meaningful connections with other people.

Faking it won't work; not only is it difficult to do, but you'll miss out on connecting with your ideal audience.

When we created our design persona at MailChimp, we were simply documenting the collective personality of the people on our teams. exercise wasn't difficult because we had hired like-minded people with similar senses of humor and complementary personalities. Although our primary goal in putting together a team was to find people who would work well together, the process turned out to be helpful for defining the brand's personality, too. Here's the design persona we created for ourselves (you can download this example from http://smashed.by/mailchimp):

Brand name

MailChimp

Overview

Freddie von Chimpenheimer IV is the face of MailChimp and the embodiment of the brand's personality. Freddie's stout frame communicates the power of

the application, and his on-the-go pose lets people know that this brand means business.

Freddie always has a kind smile that welcomes users and makes them feel at home. The cartoon style communicates that the brand offers a fun and informal experience. Yes, he's a cartoon ape, but Freddie is still cool. He likes to crack witty jokes, but when the situation is serious, the funny business stops.

MailChimp often surprises users with a fun Easter egg or a link to a gut-busting YouTube video. Fun is around every corner, but never gets in the way of the workflow.

Brand traits

Fun but not childish. Funny but not goofy. Powerful but not complicated. Hip but not alienating. Easy but not simplistic. Trustworthy but not stodgy. Informal but not sloppy.

Voice

MailChimp's voice is familiar, friendly and — above all — human. The personalities of the people behind the brand shine through honestly. MailChimp cracks jokes (ones you can share with your mom), tells stories and communicates in the folksy voice you might use with an old friend.

MailChimp uses contractions like "don't" instead of "do not" because that's how real humans speak to one another. MailChimp uses casual interjections, like

"Hmm" when he's thinking hard, and "Blech! That's awful" to communicate empathy.

The design persona that was created to guide the personality of MailChimp.

Copy examples

- **Success message**:
  "High fives! Your list has been imported."
- **Error message**:
  "Oops! Looks like you forgot to enter an email address."
- **Critical failure**:
  "One of our servers is temporarily down. Our engineers are already on the case and will have it back online shortly. Thanks for your patience."

Visual lexicon

- **Color**
  MailChimp's bright yet slightly desaturated palette

conveys fun — although not to the point of being Romper Roomy. In line with the brand's traits, the colors convey humor and yet are powerful and refined.

- **Typography**
  MailChimp is easygoing, efficient and simple to use, and its typography reflects this. Simple sans-serif headings and body copy vary appropriately in scale, weight and color to communicate information hierarchy, making MailChimp feel like a familiar, comfortable cardigan that is both functional and beloved.

- **General style**
  Interface elements are flat and simple, keeping everything easy to understand and unintimidating. Soft, subtle textures may appear in spots to warm up the space and make it feel human. Freddie should be used sparingly, and only to inject a bit of humor. Freddie never gives application-related feedback or statistics, nor does he help with tasks.

Engagement methods

- **Surprise and delight**
  Themed log-in screens commemorate holidays, cultural events and beloved figures in history. Easter eggs create unexpected moments of humor, sometimes conveying nostalgia or referencing kitschy pop culture.

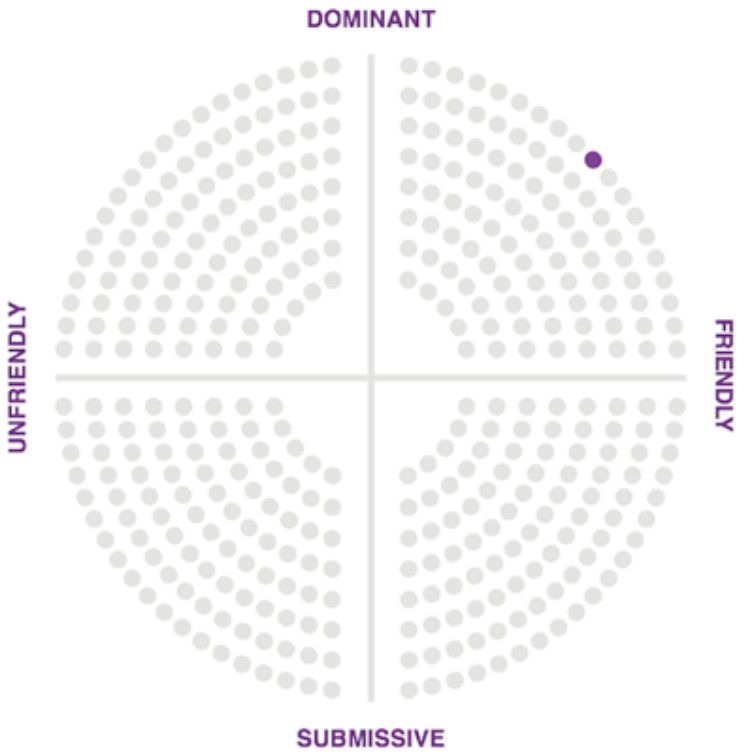
- **Anticipation**
  Freddie's random funny greetings at the top of each

main page create anticipation for the next page. These greetings do not provide information or feedback; they are a fun layer that never interferes with functionality or usability. We created this design persona primarily to guide the design of the application. It helped us dial in the voice we wanted for important elements such as success and error messages, and it helped us reconsider small design elements that were starting to deviate from our personality.

Freddie von Chimpenheimer IV, MailChimp's mascot.

The design persona is not a style guide. It's not intended to dictate how a logo can or cannot be used. It contains no color or typography specifications. It is simply a summary of the spirit of a design. When creating a design persona, you discover so much about yourself

and your colleagues, which you can then share with your audience. It's not a document to be policed, but rather a compass to keep you pointed in the right direction.



MailChimp's personality map.

With personality on our minds, we started to recognize its presence in everything we were creating, especially our writing. Blog posts, product copy, knowledge-base articles and support guides all convey our personality. We didn't always do a great job of it, though. We were injecting humor at inappropriate times. When a user is stressed out because they can't figure out a workflow

that they desperately need to complete in order to meet a deadline, a joke or informal tone is not welcome.

While the design persona helped to establish our voice, we found that further clarity was needed. Our content curator, Kate Kiefer-Lee, discovered that although a brand's voice must remain consistent in design and writing, the tone should change to match the emotional context of users. Determining the relationship between voice and tone is tricky, but Kate found an interesting solution.

DEFINING VOICE AND ADAPTING TONE

Have you ever tried to write a style guide that sets the design or writing standards for a company? Style guides identify a myriad of scenarios in which design or copy might get mangled, and they provide a framework within which everyone on a team can stay on the same path.

Many redesigns start with a well-intentioned style guide, but more often than not it's abandoned because having to constantly refer to it is impractical, and it makes people feel like the creativity police are peering over their shoulder. Kate Kiefer-Lee penned the content style guide for us to help current and incoming writers, and she had some reservations about how colleagues would receive it. Would people actually use it, or would it be seen as a burden? Having guidance on grammar and punctuation was handy, but the essence of what she wanted to communicate to the team was the relationship between voice and tone.

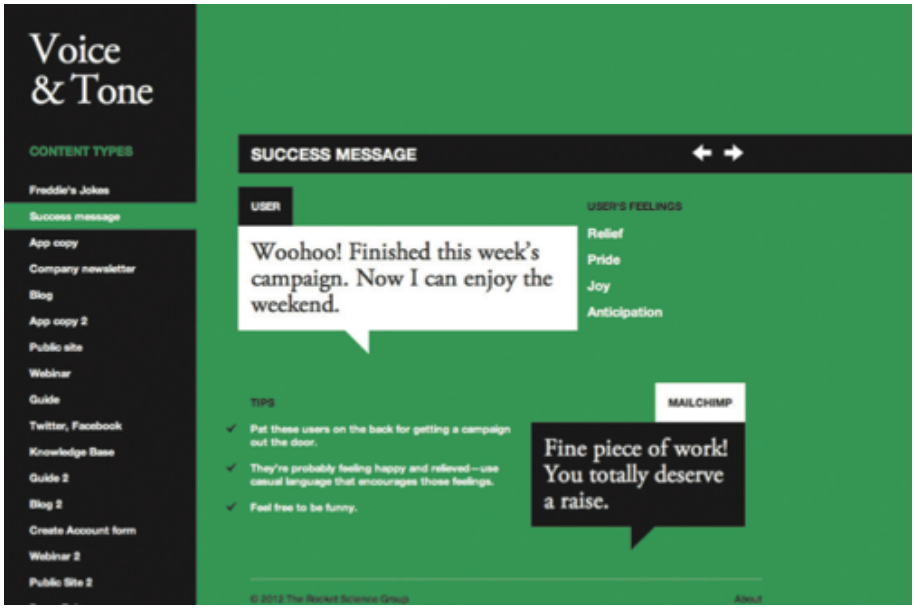
As Kate combed through blog posts, the knowledge base, microcopy, tutorials and guides to take stock of the company's voice, she noticed areas where MailChimp's voice was clear but the tone was off.

Would you tell a joke while comforting a widow at a funeral? Would you use informal language when meeting the Queen of England? I hope not! You'd certainly want to be yourself, but you would adjust your tone to suit the occasion.

In the design persona laid out above, a few examples of copy were included to show the language style and, to some degree, the variance in tone for different situations. Error messages in an application are the wrong place for a joke because the user is likely confused or stressed at the time. But a success message is a great place to employ humor or an informal tone because the user has just had a positive experience.

Rather than write a formal style guide to govern communication, Kate decided to take a different approach, one that would make understanding the voice of the brand and the variations in tone easy and fun. With the help of some colleagues, Kate created Voice & Tone[29], an interactive guide that ties tone of voice to the emotional state of readers.

29. http://smashed.by/voit



Voice & Tone helps anyone who writes for MailChimp maintain the voice and personality of the brand while adapting the tone to match the reader's emotional state.

This approach to defining personality and writing style has been well received; it's easy for people to understand, and it's fun for people to use. It's not full of rules and regulations. No one wants to be governed by the personality police. This guide gives people the freedom to write using the MailChimp voice while guiding tone in the right direction.

Kate sees a close connection between voice and personality: Voice is closely connected to personality. It's who we are. It's our perspective and it's what we bring to every piece of content our customers are reading. Voice is from our end. It's all about us. And tone is something we adapt to match our readers' feelings.

The examples of copy on Voice & Tone not only identify the feelings of readers in particular situations,

but are accompanied by matching colors on the page's background. A red background indicates anger or frustration in the reader, while green signifies delight or joy. Like any style guide, Voice & Tone helps authors establish and maintain consistency, an often-preached brand trait that fosters customer trust and strengthens relationships. But it also suggests that adaptability is important; rather than being dogmatic about style, it simply illustrates the spectrum of communication styles that writers can use in various situations.

Conclusion

So often in our industry, the motivation for starting a project is to use a new technology or technique. While expanding our skill set is lovely, technique is not the reason we fell in love with this medium. The real reward of our work is the human connections we build on the web with our fellow designers and the people we design for. Sharing more of ourselves on our websites amplifies those connections.

We've seen many examples of personality and many methods of cultivating it in this chapter. Hipmunk expresses empathy for its users with its Agony index. Wufoo managed to eliminate its marketing budget by sharing its personality in its interface and letting customers spread the word. MailChimp shaped its personality through design personas and an interactive writing guide.

These examples are shared here not to suggest that you do the same in your next project, but to spark new ideas in your work and to help you see the power of personality in design. A design that conveys a clear personality stands out in a crowded market. It elicits an emotional response from the audience that fosters long-term memory of your brand. It attracts the people you want and deters those who will be burdensome. And it impassions those users who will be your most powerful marketing channel.

Make the starting point of your next redesign this simple question, "Who am I, and what do I want to say?" Once you've answered that, then design elements and a writing style will emerge that bring out the personality at the heart of your work. ❧

Responsive Design Patterns

BY BRAD FROST

The following is an excerpt from the Mobile Book chapter "Responsive Design Patterns."

As responsive design continues to evolve, we're confronted with difficult problems about how to create adaptive interfaces that look and function beautifully across many screen sizes and environments. How do we handle navigation that's four levels deep? What about tables with a ton of columns? How do we deal with carousels?

The exciting news is that the community is discovering ways to create responsive layouts, adapt existing design conventions to a multi-screen world, and generate entirely new interaction methods. It's a great and challenging time to be a web designer as we dive head first into this multi-device future.

MODULARITY AND PATTERNS AND STYLE GUIDES, OH MY!

It's literally impossible to articulate in Photoshop all of the different design outcomes that arise from different device environments, screen sizes and user settings and a host of other variables. So we must address this by moving away from designing web pages and toward designing systems.

Just as all matter consists of small molecules, which consist of even smaller atoms, web interfaces are made

up of many small components. This means we can break the entire design system down into more modular, more manageable chunks. In doing so, we define the elements of our interfaces, figure out how each element works in a responsive environment, and then build the final system by assembling these units.

This type of atomic thinking is certainly different than the way a lot of us are used to crafting web experiences. That's where front-end style guides can be a tremendous help. A front-end style guide is a collection of your project's interface components. In her article "Front-End Style Guides[30]," Anna Debenham explains the many benefits of a front-end style guide: it makes testing easier, establishes a better workflow, offers a shared vocabulary among team members, and serves as a useful reference to keep coming back to. In my experience, the guide becomes a sort of glue for a project's team members, communicating to everyone involved how the design system works.

With the rise of responsive web design, style guides are becoming more important than ever. What better way to show how an element will adapt than to show it in its final environment? Thankfully for us, a lot of great work has been done recently in the area of responsive style guides. Starbucks graciously published its responsive pattern library[31], which it used to guide its responsive redesign. Samantha Warren created Style

30. http://smashed.by/front-end-guides
31. http://smashed.by/starbucks-styleguide

Tiles[32], a design deliverable that captures a project's fonts, colors and interface elements. Dan Cederholm created Pears[33], an open-source framework for developing one's own pattern library. And, as it happens, I've created an open-source library of responsive patterns[34].

Picking Things Apart

We need to look at our designs in a more modular way. So, let's get our hands dirty, pick apart web interfaces and look at some emerging responsive patterns! We'll look at some common ways to handle layout, navigation, interactive elements and more. By examining these emerging patterns, we can see what options are already in the wild and use them as a starting point to evolve and innovate.

Remember, these techniques are all relatively new, so don't treat them as gospel, but rather as great starting points for your particular project. There are plenty of opportunities to improve existing patterns and perhaps even create entirely new techniques. These are exciting times, indeed!

Page Layout

A logical place to start is with layout, because it is certainly the most affected by the core tenets of responsive

32. http://smashed.by/style-tiles
33. http://smashed.by/pears
34. http://smashed.by/rwd-patterns

design. Historically, we've created multi-column designs without batting an eye, but now layouts need to work harder than ever to ensure that users have a legible, functional experience, no matter how they access the web.

So, what do we do with all of these columns and modules? How should they reflow in a responsive environment? Luke Wroblewski answered these questions when he wrote about the various flavors of responsive layout patterns[35] in the wild, after having combed through Mediaqueri.es[36], a responsive gallery website. Let's take a close look at the patterns he discovered in his research.

MOSTLY FLUID

In this pattern, the layout relies mostly on a fluid grid for most of the adaptation; it stays roughly the same on medium and large screens and only dramatically shifts on small screens. This mostly fluid pattern is popular for many responsive websites because it requires maintaining only two major design views, instead of more complex patterns such as the column drop or layout shifter described next.

COLUMN DROP

The column drop is a popular technique for three-column designs. If the viewport is too small to display

35. http://smashed.by/md-patterns
36. http://smashed.by/mq-showcase

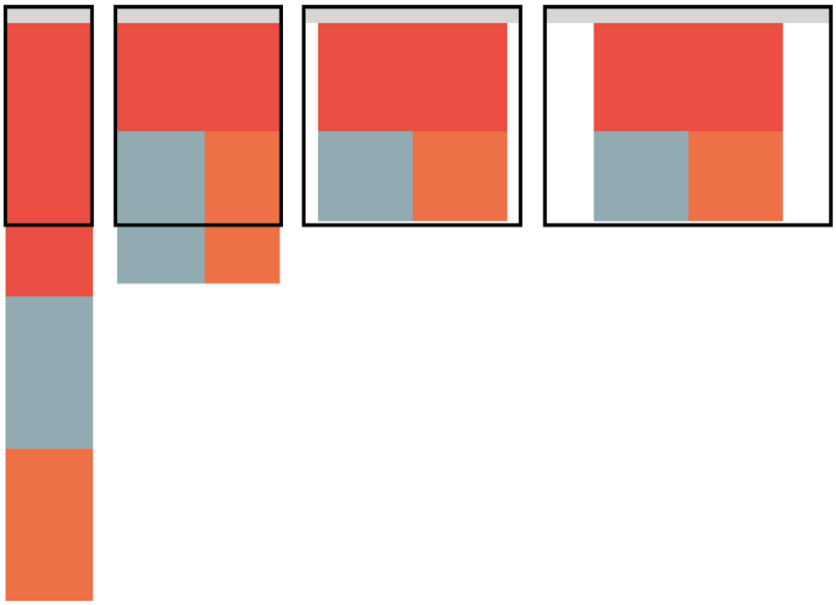
three columns side by side, one of the sidebar columns drops beneath the other two columns. The columns eventually all stack above one another on small screens. This pattern can be effective if the content contained in the three columns isn't related, but content hierarchy can become difficult to manage if the content chunks in the columns depend on one another.

LAYOUT SHIFTER

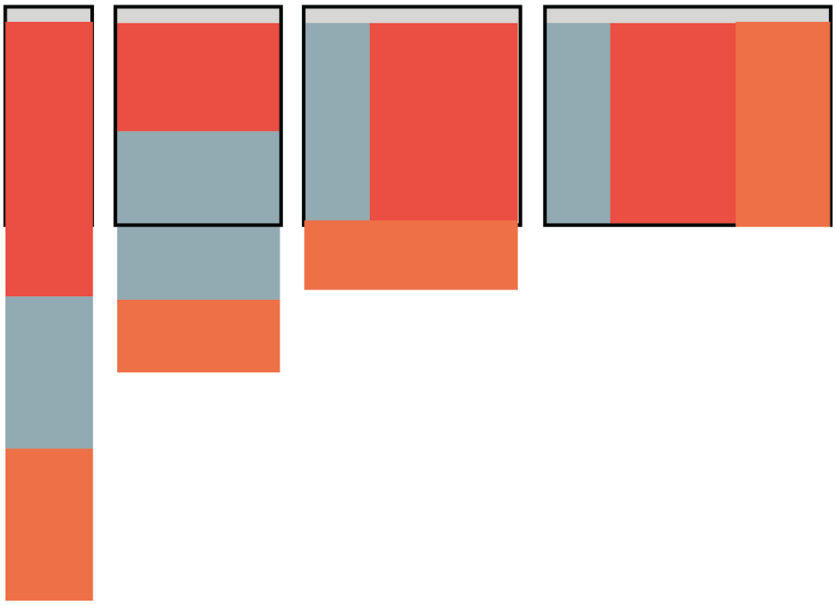
This relatively complex pattern involves creating distinct layouts at each breakpoint. As Luke states in his article:

> *"Because this inherently requires more work, it seems to be less popular than the previous two patterns I outlined."*

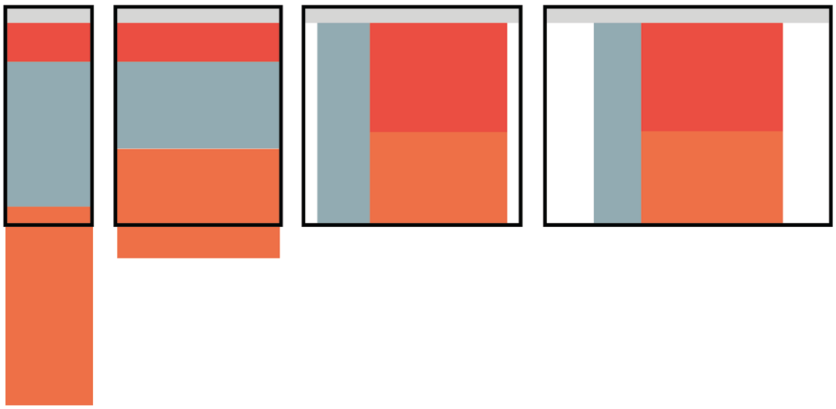
Media queries are certainly a lot of fun, but take care to avoid overcomplicating layouts just for the sake of declaring "Look, ma! It's responsive!" Once you introduce big changes at every major responsive breakpoint, you are basically creating different states in your layout that might need to be addressed separately and that, thus, could increase complexity. Remember that the goal of responsive design is to create legible, beautiful and functional experiences across as many environments as possible. Sometimes that involves 200 media queries, while other times it involves just 2.



The mostly fluid layout remains consistent until it appears on a small screen. See a demo at smashed.by/mostlyfluid.



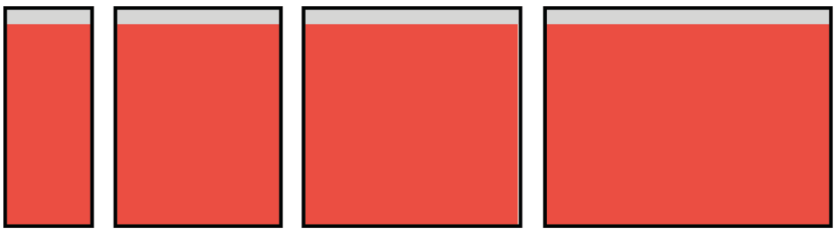
A demo for the column drop layout pattern is available at smashed.by/col-drop.



The layout shifter pattern creates a distinct layout at each breakpoint. See the demo at smashed.by/layout-shifter.

TINY TWEAKS

Speaking of minimalist approaches, the tiny tweaks pattern is as simple as it sounds. Especially suitable for single-column layouts, tiny tweaks maintains its basic structure while subtly adjusting type size, images and other components where appropriate. This pattern is useful for pages that contain only a single piece of content, such as articles, and for one-page linear websites.

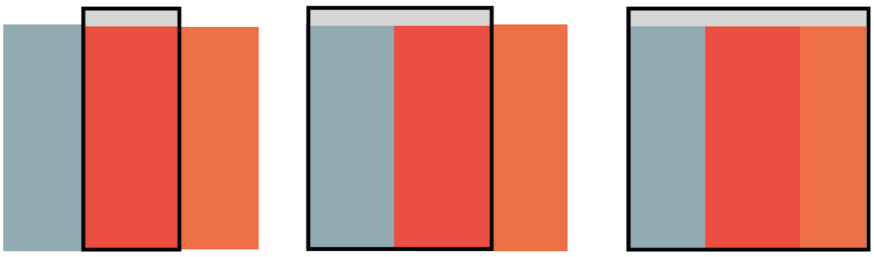


The Tiny Tweaks pattern subtly adjusts type, images and other components in an otherwise consistent layout: smashed.by/tiny-tweaks.

OFF-CANVAS

Luke lays out one of the fundamental problems with the previous patterns:

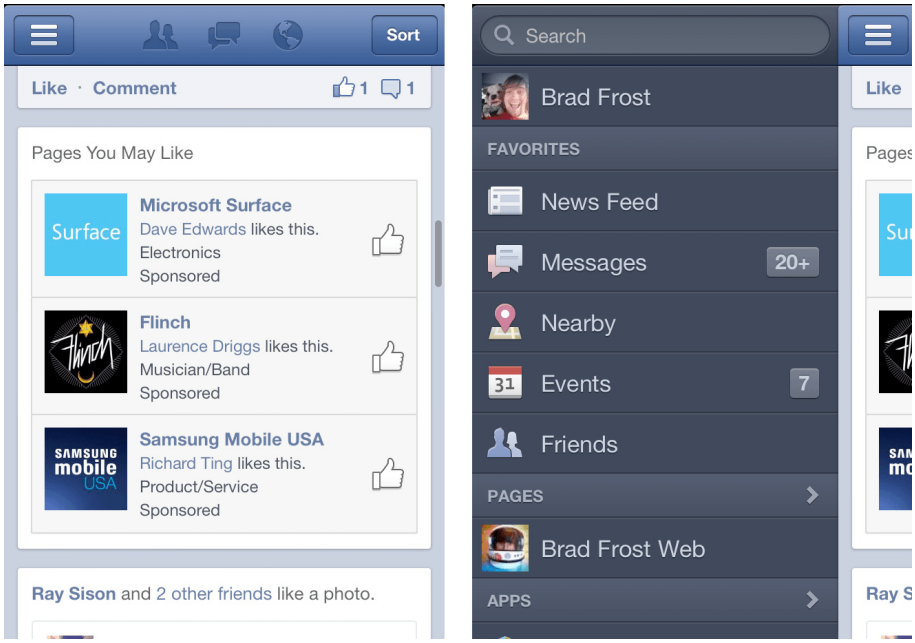
> *"While there's a lot of variety in the responsive design layout patterns listed above, there's also some common characteristics. They all tend to stack everything vertically on small screens resulting in long pages full of diverse components."*



Off-canvas layouts move supplementary content off screen, from where it can be brought in as requested: smashed.by/off-canvas.

Stacking content on top of each other isn't necessarily a bad thing, but mobile users are typically used to scrolling through a singular content type, such as a social feed, email feed or group of contacts, or diving deep into an article. Introducing multiple content types further down the page requires users to go on a scavenger hunt to find them, and it could lead to frustration and/or confusion when users stumble upon this new, potentially unrelated content.

What are the alternatives? The off-canvas pattern[37] is a clever solution that moves a page component just off screen and, when requested, exposes the content using a sliding-door effect.



Facebook's mobile navigation popularized the off-canvas technique.

This technique was popularized by Facebook's mobile fly-out navigation[38]. The off-canvas approach reserves the visible screen area for core content, while keeping additional content off to the side for easy access.

37. http://smashed.by/off-canvas-layouts
38. http://smashed.by/nav-patterns

LAYOUT CONSIDERATIONS

Whatever approach you decide to take for your responsive layout, focus on delivering a legible, logical experience to your users. Some things to think about:

- **Ensure that content flows logically.**
  As mentioned, users don't want to sift through a bunch of disparate content to find what they're looking for.

- **Treat layout as an enhancement.**
  In his talk "Rethinking the Mobile Web[39]," Bryan Rieger eloquently stated that, "The absence of support for @media queries is in fact the first @media query." By constructing layouts mobile-first, we're able to serve optimized layouts to more mobile platforms, which are less likely to support media queries. This approach also embraces the cascading nature of CSS and results in smaller, more legible, more maintainable CSS.

- **Let the content, not device dimensions, determine breakpoints.**
  Way too many devices and screen sizes are out there, and new ones pop up every day. Instead of chasing down the screen sizes du jour, establish breakpoints where the design necessitates them, such as when line length becomes uncomfortably long, images become unruly or interface elements looked stretched and unnatural. Follow Stephan Hay's advice: "Start with a fluid

39. http://smashed.by/mobile-web

layout, start small and expand until the layout looks like shit. Time to insert a breakpoint!"

- **Don't go overboard.**
  While dramatically changing a layout across screen sizes is certainly fun, don't feel obligated to insert breakpoints and radical changes just for the sake of being responsive. Instead, do whatever's required in order to create a legible layout that's a joy to interact with.

Navigation

Navigation is sometimes the most important interaction in a web experience. I've always said that navigation should be like a good friend: there when you need it, just like a friend who's willing to help you move into a new home. But it should also be considerate enough to give you your space, just as a good friend doesn't demand your constant attention.

The community has been creating some clever solutions for dealing with responsive navigation.[40] Let's look at some of the more popular techniques.

TOP NAVIGATION, OR "DO NOTHING"

For navigation that contains only a few links, keeping it at the top of the page might make sense. It's certainly easy to implement, but it's not particularly scalable and could potentially take up a lot of valuable real estate.

40. I've been researching and collecting them in my very own blog post on "Responsive Navigation Patterns."

Remember that we want navigation to be accessible but not take up a lot of space, in order to make room for the core content.

Top navigation simply displays links across the top of the page. Demo: smashed.by/topnav.

THE FOOTER ANCHOR

This clever solution places the navigation in the footer, leaving a simple anchor link in the header that lets users jump to the footer.

The footer anchor lets users jump to the navigation in the footer. Demo: smashed.by/footeranchor.

This clears up room for the core content while still providing quick access to the navigation. It is relatively easy to implement and provides great support. Just keep in mind that a jarring jump to the website's footer could disorient some users.

THE SELECT MENU

Another clever approach is to transform navigation links into a select form element for small screens. This approach saves a lot of real estate; but the select menu pattern could also be confusing because you'd be using a form element outside of its natural environment. This approach can also be tricky if your navigation includes submenus (more on complex navigation later).

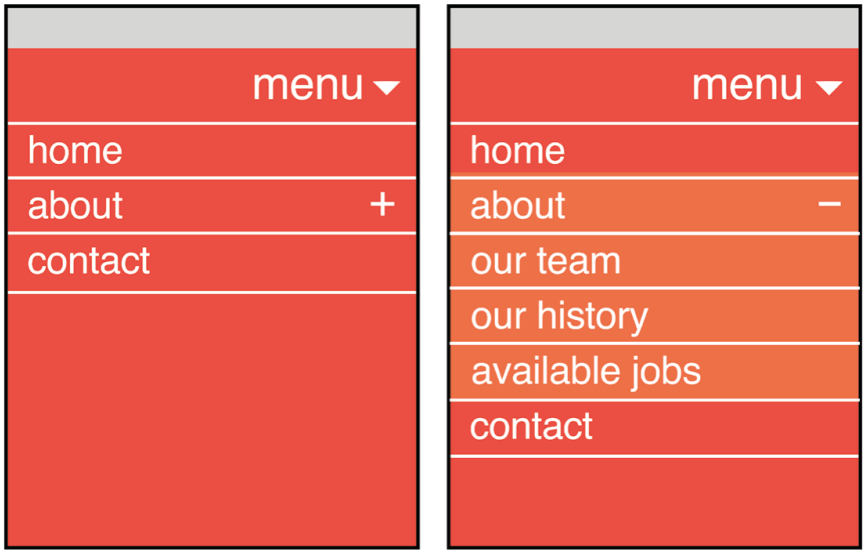
The select menu converts navigation links into a drop-down menu: smashed.by/dropdown-menu.

THE TOGGLE

The toggle is one of the more elegant patterns out there. On a small screen, the navigation is collapsed, with only a button (typically denoted by three horizontal lines[41] or a "Menu" anchor) that, when tapped, toggles the navi-

gation's visibility. The result is navigation that saves space but is still easily accessible.

We need to make sure that the navigation remains accessible to users with poor JavaScript support. But we can bypass JavaScript altogether with this pattern by using the technique that Aaron Gustafson describes in "Build a Smart Mobile Navigation Without Hacks[42]," employing the `:target` CSS pseudo-selector to toggle the navigation.



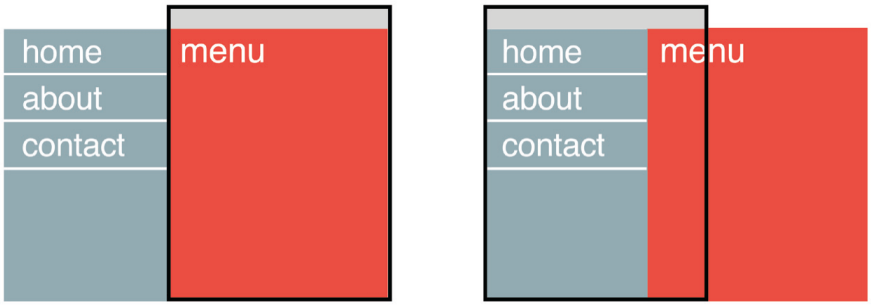
The toggle collapses navigation items, revealing them conditionally: smashed.by/togglenav.

NAVIGATION FLY-OUT

The navigation fly-out is a good implementation of the off-canvas pattern explained in the layout section earli-

41. http://smashed.by/design-navicon
42. http://smashed.by/nav-no-hacks

er. Instead of appearing above or below the core content, the navigation exists just to the side, off screen, and slides in when requested. This technique is certainly elegant, but because there are plenty of (literal) moving parts, make sure to test in as many environments as possible to avoid rendering the navigation non-functional.

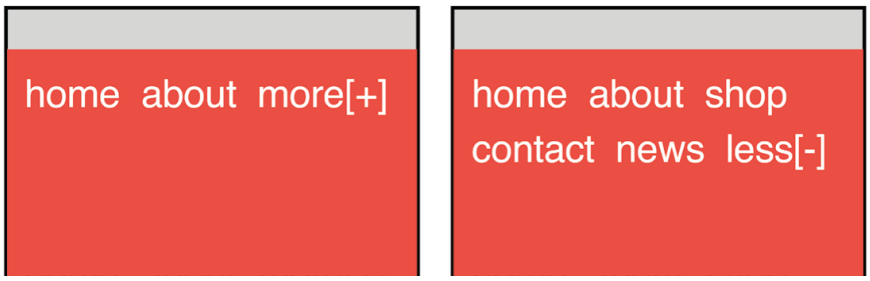


The navigation fly-out pattern tucks the navigation off screen and animates in when requested: smashed.by/navflyout.

PRIORITY+

The name "priority+" was coined by Michael Scharnagl to describe the technique of exposing what are deemed to be the most important navigation items and tucking away less important ones behind a "More" link.

This pattern can be effective for navigation that contains a lot of links at the same hierarchy level. However, deciding which navigation items to prioritize requires making assumptions about the user's intentions; while we hope those assumptions are correct, the user may be looking for something else.



Priority+ exposes the most important navigation icons and hides the rest behind a "More" link: smashed.by/priority-nav.

THE HIDE 'N' CRY

This pattern (or should I say anti-pattern?) hides navigation that is deemed unimportant for small-screen users. While this certainly saves real estate, it deprives the increasingly mobile audience of access to those parts of the experience. Remember that small-screen users want to do anything and everything that large-screen users do, provided that it's presented in a usable way. Don't penalize users for the device they happen to be browsing with.

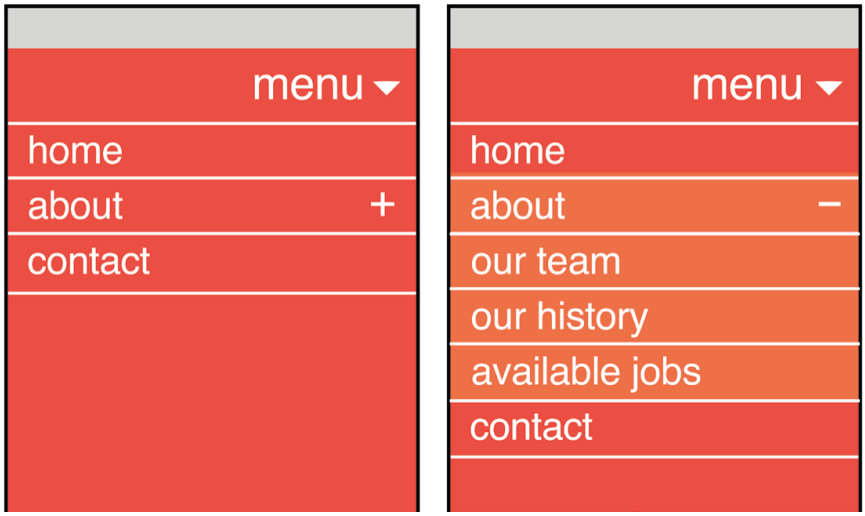
Complex Navigation

Desktop screens have plenty of room for navigation. In addition, the mouse enables users to hover over parent categories to reveal dropdown menus containing child links. However, the mobile web environment, with its lack of screen real estate and lack of hovering, poses a significant challenge for implementing complex navigation.

Sometimes whittling thousands of pages of content into three little links that fit tidily on a phone's screen is not realistic. Major organizations, massive e-commerce retailers, universities and other owners of large websites often need complex multi-level navigation. So, once again, the community is rising to the challenge and creating some great solutions for complex navigation for responsive designs[43]. Let's dive in, shall we?

THE MULTI-TOGGLE

The multi-toggle is basically just a series of nested accordions. The user taps on a parent category to reveal child categories below. At a large enough screen size, the navigation converts to the usual multi-level drop-down menus we're used to seeing.



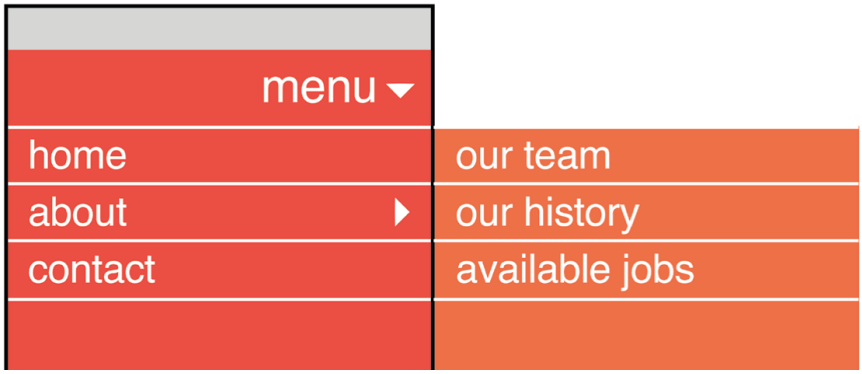
The multi-toggle is basically a series of nested accordions: smashed.by/multi-toggle.

43. http://smashed.by/complex-nav

This pattern allows the user to easily scan parent categories, while providing quick access to child elements. It's also a scalable solution for navigation that is many levels deep.

THE OL' RIGHT TO LEFT

While the multi-toggle reveals child categories beneath the parent category, the ol' right to left takes a cue from the off-canvas pattern and slides in sub-navigation from off screen to the right.



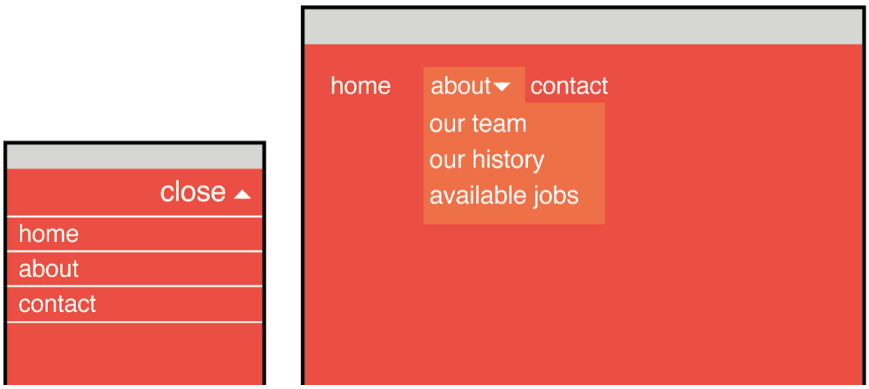
The ol' right to left pattern animates submenus in from the right when requested: smashed.by/right-left.

This is definitely one of the more elegant solutions for complex navigation; it is scalable to multiple levels, and it follows a common mobile convention of enabling the user to drill down into an experience with right-to-left animation. Because this approach is relatively intricate, test it in as many environments as possible to ensure the navigation remains accessible. Also, keep in mind

that animation (especially on mobile browsers) can vary extremely in performance (or be absent altogether).

SKIP THE SUB-NAVIGATION

Sub-navigation typically contains items that are also included on the parent category's landing page. Because that content is accessible on the landing page, it's perfectly viable to simply take small-screen users straight to the landing page and let them make their next move from there.



This pattern takes small-screen users to a new page for submenu items: smashed.by/skip-subnav.

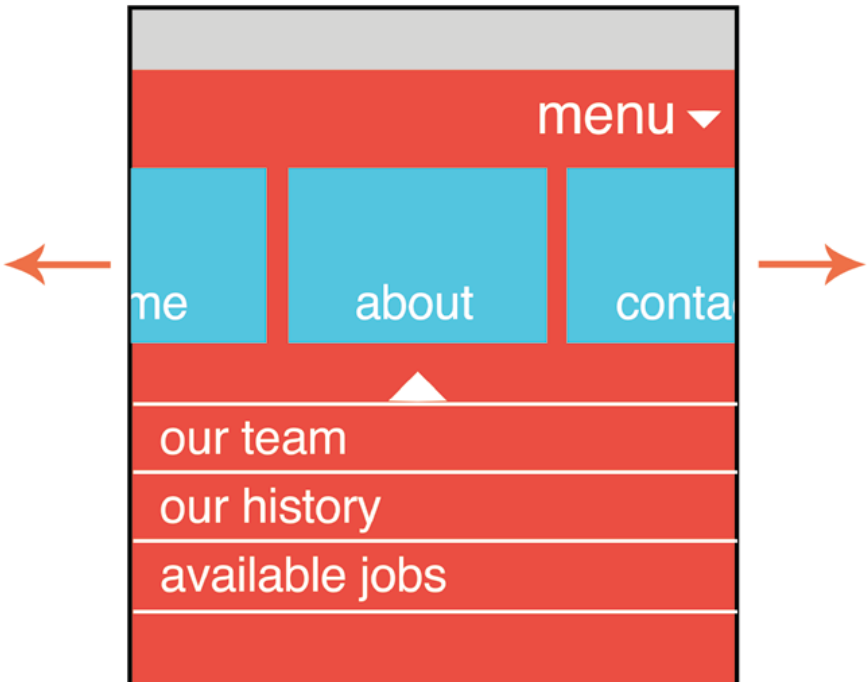
This approach frees the user from having to deal with sub-navigation altogether; however, requiring a full page refresh just to get to the next level of navigation isn't very efficient.

CAROUSEL+

In this pattern, parent categories are displayed as a carousel, with sub-menus displayed individually as a

list below. The user can swipe horizontally or use right and left arrows to move through the carousel.

This pattern is unique; however, while swiping through a carousel can work well on touch devices, it can feel tedious on non-touch ones. Also, by not exposing all parent categories at once, the user is forced to interact with the menu just to see what other categories are available to them.



Carousel+ is carousel navigation that exposes sub-navigation options below the chosen option: smashed.by/carousel-plus.

NAVIGATION CONSIDERATIONS

Whatever navigation method you choose, keep these things in mind:

- Find the balance between navigation accessibility and unobtrusiveness. Like a good friend, it should be there when you need it, but not always in your face, demanding attention.
- Because navigating is such an important interaction, test in as many environments as possible. See how the navigation works on a variety of devices and mobile browsers, including proxy browsers.
- If you're collapsing the navigation for small screens, be explicit with the navigation icon. An icon consisting of three horizontal bars is becoming the de facto standard[44]. Just make sure that whatever icon or wording you use screams "Navigation!" to the user.

Interactive Components

Some website elements contain more moving parts than simple text and images, and putting relatively complex interactive components into a responsive environment can be tricky. Let's take a look at how the community is handling common interactive components like carousels and accordions.

CAROUSELS

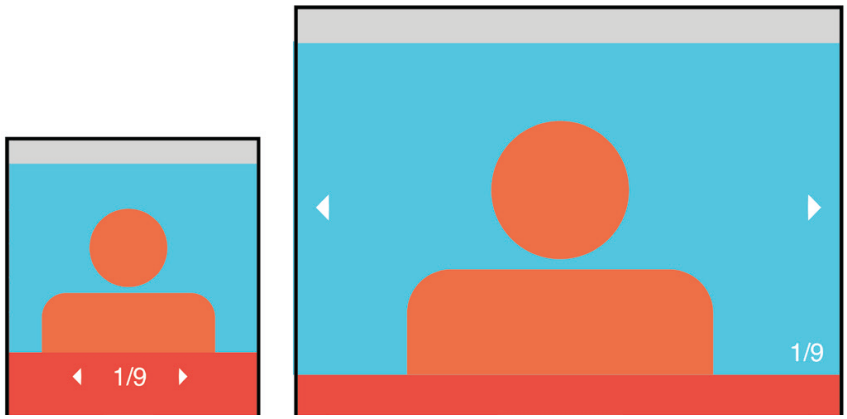
We see carousels everywhere on the web: home page promotional areas, featured stories, product images, related content and so on. They're a convenient way to

44. http://smashed.by/navicon

save real estate; they establish hierarchy within a group of content elements; and they introduce a sense of motion and interactivity that can really enhance the experience. Let's look at some ways to implement carousels in a responsive environment.

Fully Fluid

The fully fluid carousel matches the visible panel's width to the screen's size, similar to a fluid image. While the result looks similar to a static image, more is going on under the hood because the carousel's logic needs to be adjusted via JavaScript to ensure that the panels advance according to the screen's size.



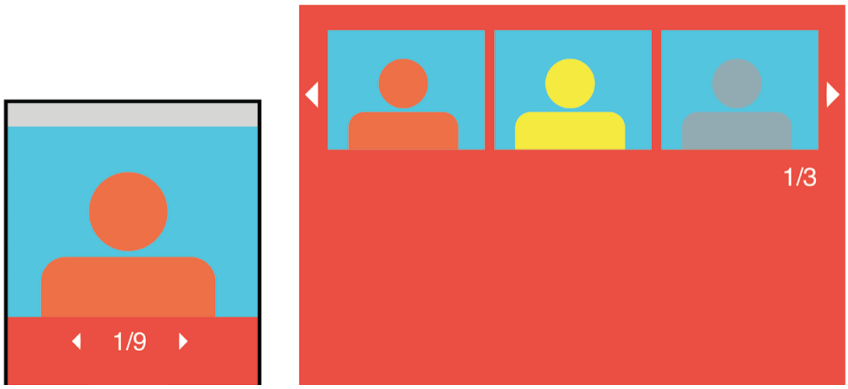
A fully fluid carousel scales each panel to match the container's width: smashed.by/fluid-carousel.

Another consideration is placement of type. A common desktop convention is to overlay text on top of a panel's images. But small screens mean small images, so setting text over an image could obstruct the image's content. To avoid this, we can simply stack text underneath

the image on small screens to ensure that both the type and the image remain legible.

The Reveal

The reveal pattern exposes more carousel panels when space is available. This technique is used on many fluid desktop websites, such as Amazon's and Vimeo's, and on responsive websites, such as Disney's.



The reveal exposes more panels in the carousel for screens that can accommodate them: smashed.by/reveal.

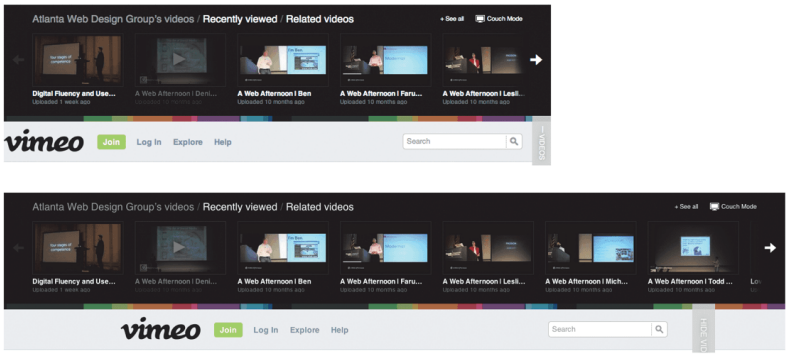
The reveal keeps content compact on small screens, while capitalizing on more space by exposing more content. A common complaint against mobile-first responsive design is that large-screen versions of mobile-first designs look vacant. The reveal solves this by introducing more content to fill the space.

Carousel Considerations

- **Make sure you actually need one.**
  Many carousels are used to sweep content under the rug. A carousel should not be a lazy substitute for making important content strategy decisions.

- **Cycle through like items.**
  Carousels should be predictable.The user should have a rough idea of what the next panel will be. Instead of cycling through disparate items ("Check out our summer sale!," "Like us on Facebook!," "Read our blog!"), give panels a running thread (product images or related news stories or related products, etc.).

- **Load only what you need.**
  If a carousel has 20 panels, please don't load all 20 by default! You don't know whether the user will even make it to the second panel. Don't load more than you have to, to keep the experience nice and fast.



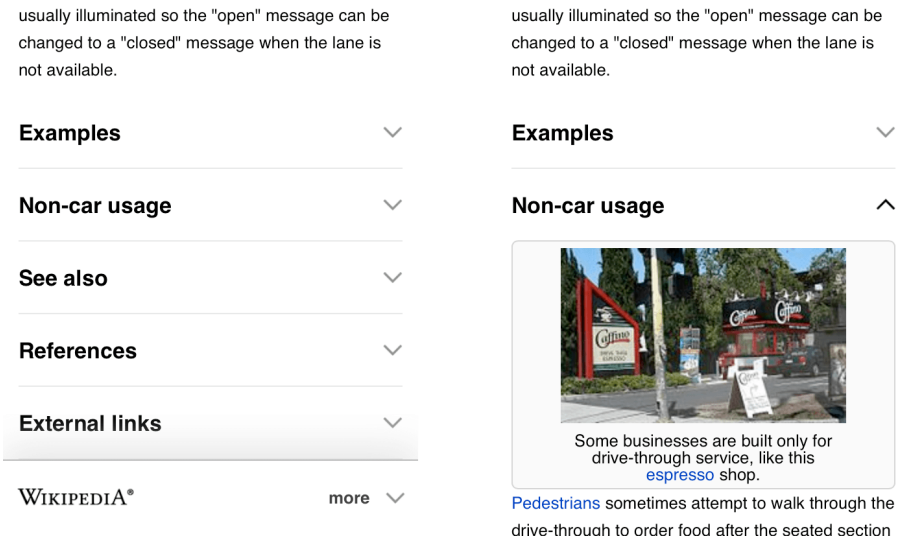
Vimeo uses the reveal pattern to show more related content for screens that can accommodate it.

- **Be explicit.**
  Tiny little dots sitting by themselves below a carousel aren't explicit enough to tell the user that this is a carousel. Add previous and next arrows, and indicate the user's current position in the carousel.

- **Treat touch as an enhancement.**
  Enabling touch gestures to swipe through a carousel is a great way to add some pizazz to the user experience. Remember, though, that not every device or browser supports touch events. Provide additional means of navigation for optimal support.

ACCORDIONS

Progressive disclosure is an extremely useful technique to help deal with super long page lengths on small screens.

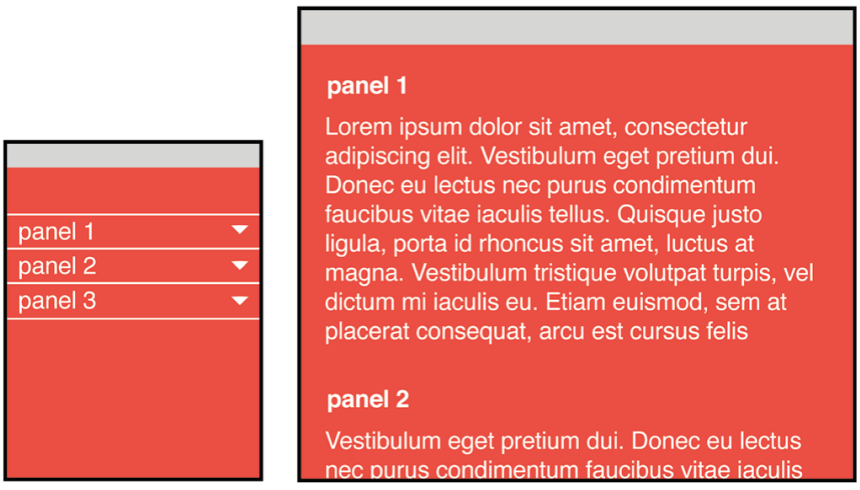


Wikipedia's mobile website uses progressive disclosure to enable users to quickly scan each section.

Progressive disclosure hints at more content by providing a title or summary of a content chunk that's hidden or not loaded. Once the user taps the desired section, the section is expanded and the associated content chunk is revealed.

Accordion to Full

Accordions have long been a common interface convention, but they makes even sense for small screens where space is at a premium. Collapsing sections of content makes for more scannable pages on small mobile screens. However, once enough screen space is available to accommodate more content, it might make sense to remove the accordion altogether and simply expose the content in full.



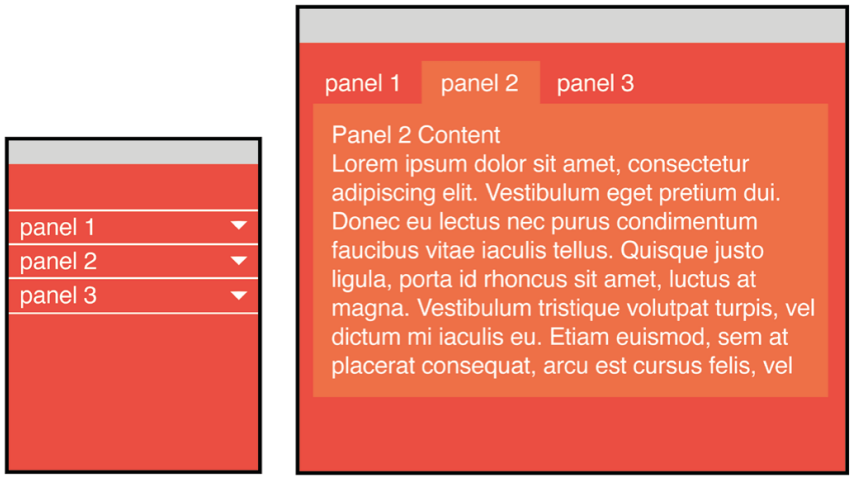
The accordion-to-full pattern collapses content sections on small screens and exposes them on large screens: smashed.by/accordion.

What I've just described is a basic accordion. Not terribly exciting on its own, but in a responsive environment, an accordion makes sense when space is at a premium. And if the screen is large enough, simply remove the accordion altogether and expose all of the content.

Accordion to Tabs

Tabs are another component that work really well for conditionally exposing chunks of content. However, tabs don't always fit right on small screens.

Thankfully, accordions work consistently well on small screens. We can combine these two popular patterns, so that an accordion is shown on small screens for a group of content and then converts into a series of tabs for large screens. While doing this certainly requires a bit of hoop jumping, the solution ensures that users can efficiently navigate those chunks of content.



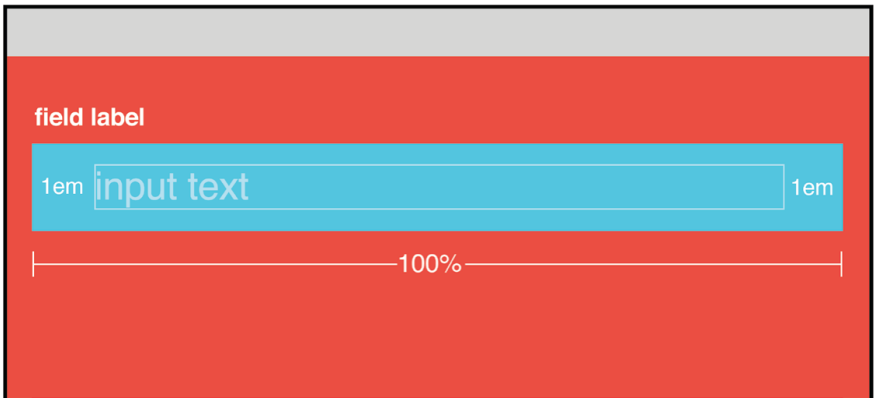
The accordion-to-tabs pattern presents an accordion on small screens and a set of tabs on large screens: smashed.by/acc-to-tabs.

FORMS

Forms add a vital element of interactivity to the web experience. Because forms are so crucial, they must be as easy to use as possible, no matter how a user accesses the web.

Fully Fluid

A common challenge on small screens is creating fully fluid form fields while still maintaining a fixed amount of padding. The traditional box model makes this surprisingly difficult. Thankfully, the CSS box-sizing property gives us a way forward. By setting box-sizing to border-box[45], we can make form fields fully fluid while keeping a fixed amount of padding.

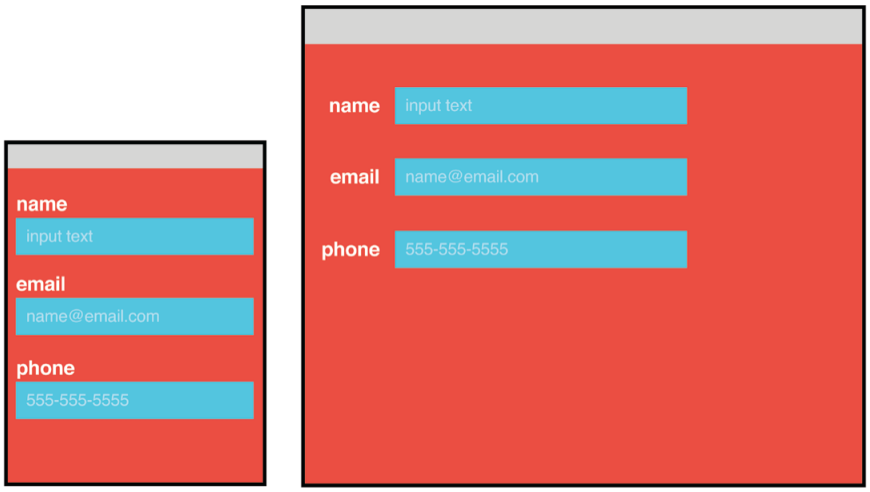


The CSS rule `box-sizing: border-box` *makes elements such as form fields fully fluid, while retaining fixed padding and borders: smashed.by/fluid-form.*

45. http://smashed.by/box-sizing

Top-to-Left Labels

Aligning labels to the left of form fields is common on large screens. But the pattern doesn't always translate well to constrained screens. A simple solution is to stack labels on top of form fields on handsets.



We could stack form labels vertically on small screens and float them left on large screens: smashed.by/top-left.

Considerations

- **Look for alternatives to forms.**
  Look for opportunities to reduce friction and perhaps remove a form altogether. Social log-in buttons can reduce the pain of logging in on small screens, and geolocation can be used in lieu of address, city and ZIP forms to serve up location-specific information.
- **Reduce the number of fields in a form.**
  Completing forms is a chore, especially if the user is

pecking on a mobile keyboard. So, in adapting your big honking form to small screens, look for opportunities to shed extraneous fields.

- **Use the appropriate input types.**
  The proper HTML5 input type will pull up a contextualized keyboard in many mobile browsers. For example, specifying `type="email"` will pull up a virtual keyboard containing dedicated .com and @ buttons, which makes completing the form just a little less painful.

- **Consider different input methods.**
  Keep in mind that devices with a myriad of input types are used to access the web: touch, D-pads, scroll wheels and many others. Employing things like the tabindex attribute can make forms more accessible and enjoyable, regardless of input method.

DATA TABLES

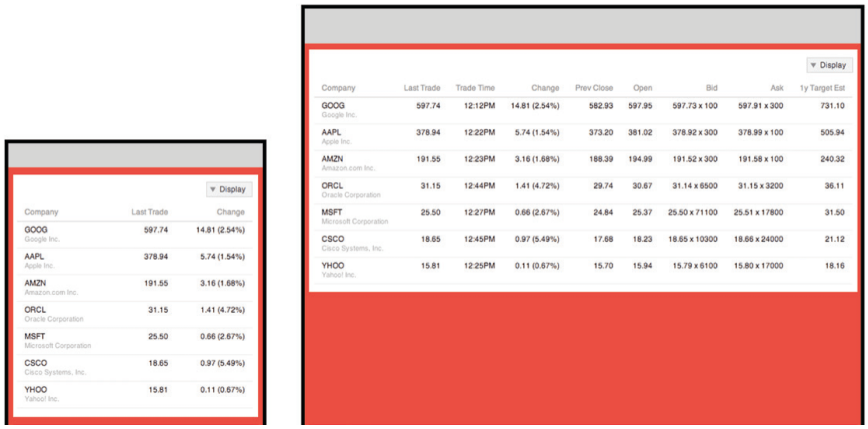
Responsive data tables are tricky buggers. A table is one interface component whose semantic meaning is closely tied to its presentation, making it a challenge to translate across many screen sizes.

Chris Coyier has rounded up[46] some of the more popular ways our community is dealing with responsive data tables. Let's look at some of the techniques.

46. http://smashed.by/resp-tables

Priority

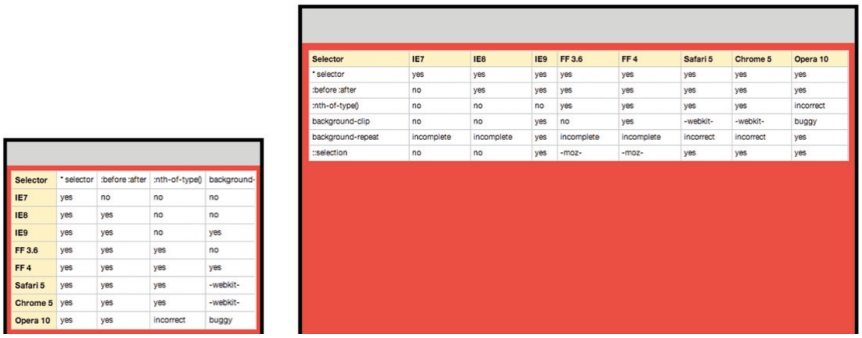
Similar to the priority+ navigation pattern, this one displays columns that are deemed to be most important for small screens, while still giving users a filtering option to see the remaining content. The wider the screen, the more columns are conditionally displayed. The result is a relatively elegant solution that gives users control over what data to view.



Priority tables expose the most important columns on small screens, making the remaining ones available on request: smashed.by/display-data.

Horizontal Overflow

Another approach taken by data-rich mobile websites such as Wikipedia is to use the `overflow-x` CSS property, which lets mobile users scroll horizontally to view more columns. Some techniques even dock the left-most column for easy comparison and analysis.



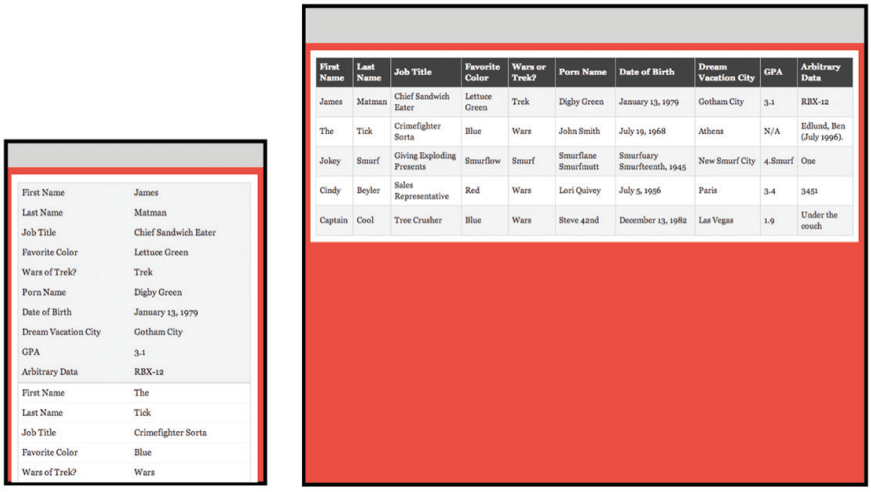
Selector	* selector	:before :after	:nth-of-type()	background:
IE7	yes	no	no	no
IE8	yes	yes	no	no
IE9	yes	yes	no	yes
FF 3.6	yes	yes	yes	no
FF 4	yes	yes	yes	yes
Safari 5	yes	yes	yes	-webkit-
Chrome 5	yes	yes	yes	-webkit-
Opera 10	yes	yes	incorrect	buggy

Selector	IE7	IE8	IE9	FF 3.6	FF 4	Safari 5	Chrome 5	Opera 10
* selector	yes	yes	yes	yes	yes	yes	yes	yes
:before :after	no	yes	yes	yes	yes	yes	yes	yes
:nth-of-type()	no	no	no	yes	yes	yes	yes	incorrect
background-clip	no	no	yes	no	yes	-webkit-	-webkit-	buggy
background-repeat	incomplete	incomplete	yes	incomplete	incomplete	incorrect	incorrect	yes
::selection	no	no	yes	-moz-	-moz-	yes	yes	yes

The horizontal overflow pattern lets users scroll horizontally to see more table columns: smashed.by/overflow-tables.

Definition List

One way to deal with tabular data on small screens is to avoid using a table altogether, swapping it with a definition list, which is more scannable on small screens. But prioritize semantics, and use whatever is most appropriate.



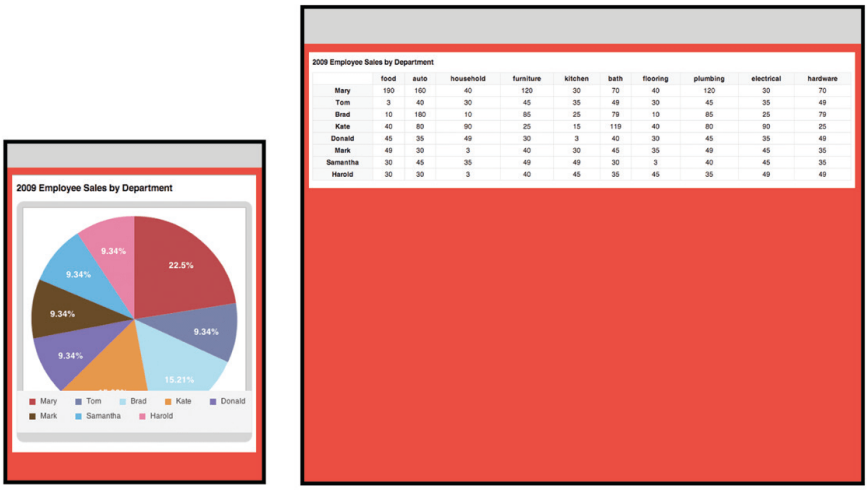
First Name	James
Last Name	Matman
Job Title	Chief Sandwich Eater
Favorite Color	Lettuce Green
Wars of Trek?	Trek
Porn Name	Digby Green
Date of Birth	January 13, 1979
Dream Vacation City	Gotham City
GPA	3.1
Arbitrary Data	RBX-12
First Name	The
Last Name	Tick
Job Title	Crimefighter Sorta
Favorite Color	Blue
Wars of Trek?	Wars

First Name	Last Name	Job Title	Favorite Color	Wars or Trek?	Porn Name	Date of Birth	Dream Vacation City	GPA	Arbitrary Data
James	Matman	Chief Sandwich Eater	Lettuce Green	Trek	Digby Green	January 13, 1979	Gotham City	3.1	RBX-12
The	Tick	Crimefighter Sorta	Blue	Wars	John Smith	July 19, 1968	Athens	N/A	Edlund, Ben (July 1996).
Jokey	Smurf	Giving Exploding Presents	Smurflow	Smurf	Smurflane Smurfmutt	Smurfuary Smurfteenth, 1945	New Smurf City	4.Smurf	One
Cindy	Beyler	Sales Representative	Red	Wars	Lori Quivey	July 5, 1956	Paris	3.4	3451
Captain	Cool	Tree Crusher	Blue	Wars	Steve 42nd	December 13, 1982	Las Vegas	1.9	Under the couch

The definition list pattern converts a table into a definition list, which is friendlier to small screens: smashed.by/table-def-list.

Table to Chart

One of the wilder solutions out there is to convert data tables into pie charts on small screens. While not appropriate for every data table, the solution is pretty interesting and visually appealing for some use cases.



Some tables can be converted into graphical charts on small screens: smashed.by/table-to-charts.

Into the Great Wide Open

We as web designers are tasked with crafting experiences that look and function beautifully in a host of contexts, even ones that haven't been invented yet. That's certainly a tall order. But by breaking the overall interface into smaller puzzle pieces, we can make these admittedly tough problems a little more bearable.

We can always count on change in this ever-shifting web landscape. As responsive design continues to evolve, we'll see some techniques and patterns fall away, and the ones that stick around will surely evolve.

And that's OK. While implementations will surely change as time marches on, the need remains to serve the web in all its glory to more people in more environments than ever. That's certainly a challenge for us, but also an absolutely amazing opportunity.

Here's to the future! ❧

Robust, Responsible, Responsive Web Design

BY MAT MARQUIS

The following is an excerpt from the Smashing Book #4 chapter "Robust, Responsible, Responsive Web Design."

Progressive Enhancement

Responsive web design is, in a sense, a natural extension of progressive enhancement. Responsive web design and progressive enhancement both aim to ensure a usable experience for all users, regardless of their browsing context. RWD is concerned more with providing users with a natural-feeling layout and presentation for accessing content, and progressive enhancement aims to ensure access to that content regardless of a browser's features and capabilities. We often see the two concepts conflated — a rare bit of web development confusion that ends up working in everyone's favor. To many, responsive web design extends beyond simply a flexible layout tailored by media queries, and includes such concerns as the ever-fluctuating landscape of mobile browser features, offline experiences, and network performance. These concerns are the realm of progressive enhancement for sure, but it's a blurring of lines that I'm glad to see — the kind that only serves to highlight potential issues that might inhibit users' access to

the sites we build, and keep those issues at the front of our minds.

The foundation of progressive enhancement is to think in terms of an element's inherent meaning: what is the best way to express this widget in markup alone? How can we represent the intention of an element in a way that works for everyone, and then build up from there? At the core of every jQuery Mobile widget is markup that's meaningful regardless of JavaScript or CSS. The range slider[47], for example, is powered by a `number` input, which degrades to a text input in browsers unfamiliar with the relatively new `number` input. In JavaScript-capable environments, that input is enhanced to a sliding control. Whether a user receives the enhanced or the basic experience, they'll be able to input their data. When you apply this principle to an entire site, you build a site that makes sense — one that's usable — before layering your JavaScript enhancements over that.

It's not hard to find figures about the number of users that have thrown the sparking knife-switch buried in their browser preferences and disabled JavaScript completely. As you might have guessed, it's fairly rare — and those figures are usually accompanied by a call to abandon progressive enhancement and cater only to the highest common denominator. But support on the web is rarely a binary thing — even if we were to assume that a very low percentage of our users will

47. http://smashed.by/jquerysliders

have JavaScript disabled wholesale, relying on JavaScript for the core functionality of a site introduces a tremendous point of failure.

I don't have to tell you guys how atrocious the code is in most of those third-party share widgets or advertisements. What happens when your user clicks on something while waiting for a megabyte of the "Like" button to download? What happens when one of your ad tags is so catastrophically broken that the browser gives up on JavaScript altogether? Will your site function at all?

Saying that everyone has JavaScript enabled isn't much more than an excuse for developer convenience — it doesn't improve the users' experiences, just ours. Even if we could assume that a site that completely relies on JavaScript will work for every user, we're still not doing them any favors. We're still introducing the potential for a user to find themselves staring at an indefinite loading GIF or an empty page — frustrating things we've all seen plenty of times, even here, with fast and reliable internet connections. I've personally seen a number of otherwise brilliantly constructed sites time out or display a blank page as a result of an ad-blocking browser plugin: a single point of failure that halts the execution of every other script on the page, rendering the entire site unusable. Granted, this is little more than an inconvenience for you and me. We can safely pinpoint where a problem like that comes from and work around it. For less tech-savvy users, what

might only be an inconvenience for us could mean a complete loss of access to a page or an entire site.

Access Is Vital

The Boston Marathon bombing took place on April 15, 2013, and like many people, I turned to the *Boston Globe* site throughout the day to find out what was going on around me — the reasons behind the constant sirens — and for reassurance that the blasts I heard all day were controlled detonations carried out by the police. Suffice to say, reliable access to BostonGlobe.com became very, very important for a huge number of people.

Given the tremendous initial surge and the sustained traffic throughout the day, and despite the tremendous efforts of the *Globe's* dev and ops teams, it's easy to understand how their servers started experiencing some issues. At peak traffic their content delivery network gave out completely, resulting in a site with no external assets: no images, no CSS, and no JavaScript.

I'm certain that more than a few of you just winced, but it wasn't the disaster you might be envisioning. The site wasn't as pretty as it could have been, true, but it worked just fine. Had the *Globe* relied entirely on JavaScript for rendering or navigation, or if the scripts and styles weren't built on a foundation of progressive enhancement, tens of thousands of understandably uneasy users would have been left at the mercy of their search engines for news.

We'll be talking about progressive enhancement as it relates to responsive web design for the remainder of this chapter, but not strictly in the ways you might have read about it in the past. For us, it won't be a matter of simply adding functionality in unobtrusive ways, but applying that same philosophy to how we load a site's assets (JavaScript, CSS, images, and even our markup) in a way that's best suited to a user's context. We'll work towards building a responsive website that not only looks tailor-made for any display, but delivers device- and context-appropriate assets to match.

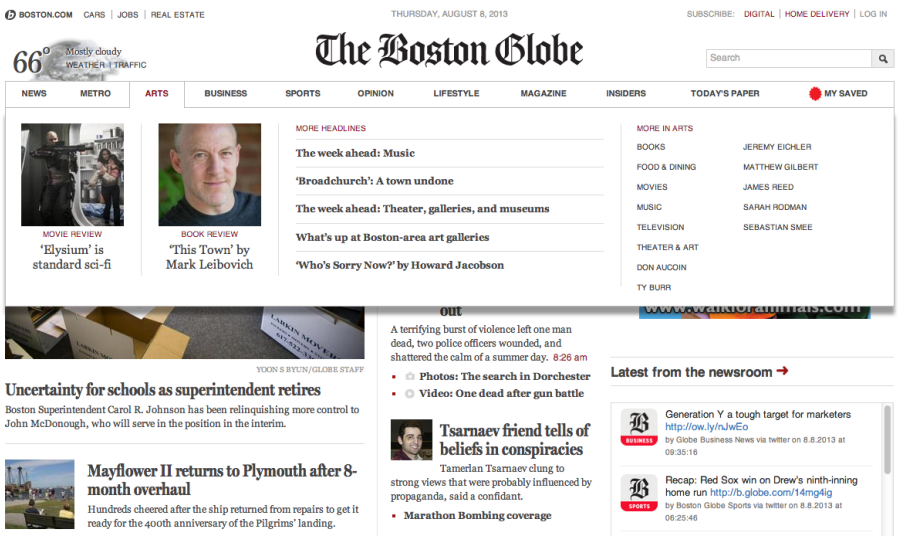
Markup

In terms of our markup alone — the foundation for everything else on our site — there's a great deal of room for optimization. How do we deliver rich experiences while ensuring that the most important part of a page, the content itself, is available as soon as possible?

The simplest (though far from ideal) way to handle showing and hiding this content would be to use `display:none;` and reveal the drop-down menus when the navigation links are hovered over. But we can safely assume that a user looking to find the scores of yesterday's Bruins game wouldn't be hovering over each link in the navigation on each page of the site. Most of this content would never be visible to the user, who would still be left requesting all this markup — including the images, which would still be loaded in the vast majority

of browsers, even if they were never displayed — on every single page of the site.

While we didn't want to include all that content just in case, we also didn't want to limit the user at certain breakpoints, or make assumptions based on the users' context. The drop-downs are a useful addition to the site, and worth keeping.



On hovering over the links in the primary navigation, BostonGlobe.com shows a drop-down containing featured articles, the most recent articles in that section, and related subsections.

We needed a baseline, strictly markup means of allowing a user to opt into content on the site, which was simpler than it might sound. As one would expect, all the links in the primary navigation take the user to a landing page containing all the same content: featured articles, recent articles, and navigation for each subsection. The drop-down menus are a nice convenience, but not essential for navigating the site. It made sense to

treat these drop-down menus as enhancements, and load them via JavaScript only as they were needed.

To lazily load other pieces of inessential content on the *Globe* site without introducing any barriers to access, and inspired by the approach we took to the drop-down menus, we developed a simple markup-driven JavaScript pattern named AjaxInclude[48]. AjaxInclude enhances links by using them to fetch a fragment of the linked content. The anchor tag itself serves as the fallback, ensuring that users will be able to access the underlying content whether or not JavaScript is available, but also provides all the information our script would need to fetch the related content and apply it to the page: the location of the document fragment to be used in place of the link; and how that content should be injected into the current page relative to the position of the link.

To replace the link with the injected markup:

```
<a href="/sports" data-replace="articles/sports/
fragment">Sports</a>
```

To insert the content before the link's position in the source:

```
<a href="/sports" data-before="articles/sports/
fragment">Sports</a>
```

48. http://smashed.by/github-ajax

To insert the content after to the link's position in the source:

```
<a href="/sports" data-after="articles/sports/
fragment">Sports</a>
```

From a responsive design standpoint, this same script can help us avoid serving all of our content and selectively hiding it, instead allowing us to include the content only when it suits our layouts. AjaxInclude also allows us to use a second attribute so we can qualify the content's inclusion above or below a certain breakpoint.

Here, we would include the referenced fragment only at viewport widths of 30em and up:

```
<a href="/sports" data-append="articles/sports/
fragment"
data-media="(min-width: 30em)">Sports</a>
```

Starting with our site's core content as our foundation allowed us to make larger decisions about how and when we could load additional assets without running the risk of leaving any users out in the cold. If all else should fail, the site's intent would still remain intact: no user would be left unable to use the site. It establishes a baseline that works for everyone, while affording us the freedom to deliver context-appropriate scripts and styles in the same way: a functional website for all, with conditional enhancements when appropriate.

This same reasoning extended a recent project, a site that featured an image carousel on every page, showing

apartment listings. This would frequently mean including dozens of high-resolution images per page. But where these carousels appeared alongside a number of other vital pieces of information — potentially the users' primary concerns, certainly more so than the photographs of the listing — there was no guarantee that the user would click through the galleries upon landing on each page. Had we included all of the associated `img` tags in our markup and simply hidden them with CSS, it would do nothing to prevent those images from being requested. We would potentially cost our users megabytes at a time, simply to read a few snippets of text.

In this instance we applied the same "as needed" philosophy as we did with our AJAX navigation pattern: the initial photo is included with the first payload of the page, and the gallery script is initialized. Although using AJAX to fetch each individual slide while the user navigated through the gallery would end up adding a troublesome delay and harming the overall experience, we wanted to ensure that we were still responding to the user's intent.

In this instance, when the user triggered the next link, we would fetch and append all the other slides at once. This still resulted in a slight delay; the user was presented with a loading indicator upon hitting the second item in the slideshow while the other items were loaded. After a few iterations, we ended up loading the first and second photos up front. When the user clicks to the next item in the gallery — already in the markup

— the third item onward is quietly loaded behind the scenes and added to the slideshow. By the time the user interacts with the next trigger again just a moment later, the third slide is ready and waiting for them. The result is massively reduced bandwidth cost for each user on the site, but a completely seamless experience at the same time.

These kinds of optimizations are matters of finesse and refinement of simple concepts — seemingly small steps with enormous potential benefits for our users.

CSS

Once we have meaningful markup as our foundation and we know that we're ensuring a usable experience for everyone, we can make some key decisions about how we deliver those experiences without leaving anyone out.

While working on the jQuery Mobile project, we quickly realized that even serving our core CSS indiscriminately could introduce the potential for a broken experience. Anyone using a feature phone or early smartphone would be poorly equipped to deal with advanced styles, even if we were to assume that the styles themselves would degrade gracefully. Simply attempting to parse the style sheet itself could introduce issues on platforms as common as Blackberry devices that are only a few years old.

We settled on the idea of serving two different levels of enhancement — sort of an experience breakpoint —

after determining whether a browser is capable of handling complex layouts and enhanced functionality, or whether it's somewhere closer to a feature phone. As daunting as that may sound, this process turned out to be simpler than you might think. The first step is to split up our style sheet.

The initial style sheet is a feature phone-caliber set of styles: font sizes, block versus inline elements, a couple of solid-color backgrounds. This style sheet gets delivered to every user. Since it's served to everyone, we've recently started using the Normalize CSS reset[49] as the basis for our initial style sheet. Rather than acting as a reset that zeroes out browser default styles, it provides a sensible normalized set of defaults: useful to bundle up our enhanced styles for qualified users, while serving as a reasonable baseline for our basic styles.

The enhanced style sheet is pretty much everything else — advanced layouts, animations, web fonts, and so on — minified and gzipped, of course.

What we end up with are two very different *looking* experiences, granted, but not different core experiences. We're not simply hiding anything, even from basic users. No one is left out in the cold. A user on an older or underpowered device will be provided with a far more usable view of the site. For that matter, they likely aren't expecting much in the way of bells and whistles

49. http://necolas.github.io/normalize.css

in the first place: the age-old question "Does my website have to look the same in every browser?" writ large.

In the original version of Filament Group's Enhance.js[50] and early versions of jQuery Mobile, we used a series of largely unrelated feature tests to determine whether a device qualified for the basic or enhanced experience. We would set the result of these tests in a cookie, and use that variable to deliver assets throughout the user's time on the site.

Eventually we realized that the test lined up closely with support for media queries, which is the key component in our complex layouts and a far more relevant test. In addition, it gave us a native method for conditionally delivering these style sheets, removing the dependency on JavaScript.

```
<link rel="stylesheet" href="/css/basic.css">
<link rel="stylesheet" href="/css/enhanced.css"
media="only all">
```

Here, the basic style sheet is linked as usual. Everyone gets that. The `media="only all"` attribute on the enhanced style sheet ensures that the style sheet is only applied by browsers that understand media queries. Of course, this means excluding older versions of IE — but we've given ourselves some options there. We can still deliver our enhanced style sheets to Internet Explorer through conditional comments, while ensuring that

50. https://github.com/filamentgroup/EnhanceJS

versions of IE prior to the minimum version we've specified get a perfectly usable basic experience. Rather than choosing a minimum version of IE to support and leaving the site broken in earlier versions, we simply provide earlier versions of IE with the basic experience.

```
<link rel="stylesheet" href="basic.css" id="basic">
<!--[ if ( gte IE 6 ) & ( lte IE8 ) ]>
    <link rel="stylesheet" href="enhanced.css">
<![endif]-->
<!--[ if ( !IE ) | ( gte IE 9 ) ]><!-->
    <link rel="stylesheet" href="enhanced.css"
    media="only all">
<!--<![endif]-->
```

We serve up our enhanced CSS the old-fashioned way for IE8 and above, while other browsers will still use the `media` qualified link. True, IE8 still doesn't know what to do with a media query, so we might include Filament's Respond.js[51], a lightweight script that parses and translates `min-width` and `max-width` media query support for IE 6–8.

Now, we've already gone to the trouble of ensuring that our site remains useful, if not ideal, in browsers that don't receive our enhanced styles. In this way, we've bought ourselves some breathing room: should we choose not to shim media query support for vastly outdated versions of Internet Explorer, we can change

51. https://github.com/scottjehl/Respond

our conditional comments to only deliver the enhanced experience to IE8 and above, or serve them a separate, static-width style sheet. Support for older versions of Internet Explorer is no longer a black-and-white issue; we're no longer painting ourselves into a corner. We support older versions of IE, for certain, just differently. It just so happens we do so in the most painless possible way for us, and in a way that ensures users at the mercy of archaic browsers aren't presented with a Jackson Pollock painting of our site. It's hard to argue with that, from any angle.

There is one other catch aside from old IE support, and unfortunately one that affects all other major browsers: when we qualify our style sheets with `media` attributes, they aren't applied, but they will be *requested*. This makes sense since we never know if an environment is going to change. If an external monitor is plugged in or a window is resized, we wouldn't want to wait for those new style sheets to be requested. As you can see from the figure below, browsers do tend to be a bit excessive about it.

These are, unfortunately, blocking requests. Even if the style sheet could never possibly apply to a user's context, the page will still be prevented from loading while the style sheet is requested, downloaded and then ignored by the browser.

We did some experimenting with asynchronously loading applicable style sheets in a project named eCSSential[52], but found that using JavaScript to request our style sheets meant sidestepping a number of

browser-level optimizations. For the most part, eCSSential roughly broke even with loading CSS the old-fashioned way — better in some browsers and worse in others — but it did introduce a dependency on JavaScript.

	iOS6, Android 4.0, Chrome 24, Firefox, IE 9, Opera 12	Opera 11
only all	Downloaded	Downloaded
(min-width: 9999px)	Downloaded	Downloaded
(min-device-width: 9999px)	Downloaded	Downloaded
(min-device-pixel-ration: 7)	Downloaded	Downloaded
tv	Downloaded	Downloaded
handheld	Downloaded	Downloaded
dinosaur	Downloaded	No request

Requested `media` attribute table: Interestingly, somewhere between version 11 and 12, Opera decided that it should account for me plugging my laptop into an external brachiosaurus.

While eCSSential didn't give us the definitive way forward we'd been hoping for, it did lead to a number of bugs filed against browsers, and conversations are taking place right now about how browsers might asyn-

52. https://github.com/scottjehl/eCSSential

chronously download inapplicable style sheets in an optimized, non-blocking way.

JavaScript

The same approach we took with our CSS applies to our custom JavaScript: we start with a basic set of scripts for everyone, and use the same test to determine whether they get the enhanced JavaScript experience: media query support.

The initial JavaScript payload includes: Filament's Enhance.js[53] to conditionally request scripts and style sheets; Modernizr[54] as our feature testing framework; and Respond.js if we've chosen to give IE8 the enhanced experience. These scripts are loaded in the head of the page, since they're either time sensitive (Respond.js) or things we need to be ready right away in case we reference them in our other scripts (Modernizr).

```
(function(win, undefined){
    var mqSupport = "matchMedia" in win &&
    win.matchMedia( "only all" ).matches;
    if( !mqSupport &&
    !respond.mediaQueriesSupported ){
        return;
    }
})( this );
```

53. https://github.com/filamentgroup/enhance
54. http://modernizr.com

This script checks whether the user's browser supports the `matchMedia` method (JavaScript's native method of parsing media queries) and then, just for good measure, it ensures that the same `only all` test that we're using in our CSS passes. If the native method doesn't exist, it checks against Respond.js's shimmed media query support. If you're targeting a specific minimum version of IE for the enhanced experience, this Respond.js test could be swapped out in favor of checking for the existence of an IE conditional class.

```
(function(win, undefined){
    /* This script assumes a conditional comment
    scheme along the lines of the following:
    <!--[if (lt IE 8) ]> <html class="old-ie">
    <![endif]-->
    <!--[if (IE 8) ]> <html class="ie8">
    <![endif]-->
    <!--[if (gt IE 8)|!(IE)]><!--> <html>
    <!--<![endif]-->
    */
    var mqSupport = "matchMedia" in win &&
    win.matchMedia( "only all" ).matches,
        htmlClass = document.getElementsByTagName(
        "html" )[ 0 ].getAttribute( "class" ),
        ie8 = htmlClass && htmlClass.indexOf( "ie8"
        ) > -1;
    if( !enhanced && !ie8 ){
        return;
```

```
    }
})( this );
```

The trouble is, between Enhance.js, Respond.js, our Modernizr build, and our new enhancement test, we've just added four blocking requests to the head of the page. We could chuck all of these into a single file, but that's likely to cause us headaches when it comes time to update any of these libraries.

To avoid inconveniencing ourselves or burdening users with additional requests, we've recently introduced the Grunt task-running framework[55] to our development process. Grunt can be set to watch a directory and concatenate your files whenever anything changes, meaning you can keep all your libraries and custom JavaScript in separate files and work on them as usual, but link your templates to a single automatically generated "dist" file that's ready for production. It will do the same for your style sheets, allowing you to split up your enhanced CSS and organize your development environment however you'd like, but output a single concatenated file. Further, Grunt will minify all your JavaScript and CSS, lint your code for errors, run your unit tests, or run any custom task you could imagine — all done automatically, via the command line. Grunt has very quickly become an indispensable part of our development process.

55. http://gruntjs.com

With our concatenated initial JavaScript file in place, we now have a framework for conditionally loading files as needed, based on the user's context. We can asynchronously load larger JavaScript libraries, plugins and custom scripts that apply site-wide without delaying the page load. If we have custom scripts to add swipe interaction on touch devices, we can feature-detect for touch events and include those scripts, and their corresponding styles, only if they're needed. If there are any unique parts of the site with highly specific CSS or JS, we add a class to the `body` tag and load those page-specific scripts and style sheets only when those pages are loaded.

Remember that there will be a slight delay when loading a style sheet using this method. Be sure to limit this approach to styles for specific components in a layout rather than entire pages, or you risk presenting the user with a flash of unstyled content.

A Little Help From The Server

These conditional requests can add up quickly on a large project: a JavaScript framework, a few plugin libraries, scripts to add device-specific enhancements like offline storage or touch support. While they're no longer running the risk of delaying a page's core content from loading, enough asynchronous requests will still have the potential to make a site feel sluggish, and prevent our enhancements from being available to the user as quickly as they might expect.

To get around this, we're using a server-side concatenation pattern called QuickConcat[56], built to work with Enhance.js, to bundle up all our conditional scripts and style sheets into a single request.

QuickConcat is a small PHP file that intercepts and parses requests for comma-separated sets of scripts or style sheets, and assembles them into a single file for delivery to the user. For example:

```
<script src="quickconcat.php?files=js/file1.js, js/
file2.js, js/file3.js"></script>
```

Or:

```
<link href="quickconcat.php?files=css/file1.css,
css/file2.css, css/file3.css" rel="stylesheet">
```

With a bit of clean-up via an .htaccess file (or the equivalent for your server environment):

```
<script src="js/file1.js, js/file2.js, js/
file3.js"></script>

<link href="css/file1.css, css/file2.css, css/
file3.css" rel="stylesheet">
```

We'll still use Grunt to combine our initial JavaScript files and our global CSS, since they'll never vary on the client side — where QuickConcat shines is our asynchronous requests. Rather than writing multiple

56. https://github.com/filamentgroup/quickconcat

`script` and `link` tags into the page and sending out a handful of requests, we can use Enhance.js to prepare a list of scripts and style sheets that apply to the user's context and request them all at once:

```
(function(win, undefined){
    var mqSupport = "matchMedia" in win &&
    win.matchMedia( "only all").matches;
    if( !mqSupport &&
    !respond.mediaQueriesSupported ){
        return;
    }

    ejs.addFile.jsToLoad( "js/lib/jQuery.js" );
    ejs.addFile.jsToLoad( "js/lib/konami-code.js" );

    // Load custom fonts > 600px
    if( window.screen.width > 600 ){
        ejs.addFile.cssToLoad( "css/fonts.css" );
    }

    if( Modernizr.touch ) {
        ejs.addFile.jsToLoad( "js/swipe.js" );
        ejs.addFile.cssToLoad( "css/swipe.css" );
    }

    ejs.enhance();
})( this );
```

When Enhance.js is invoked, all the files queued up with `ejs.addFile.cssToLoad` and `ejs.addFile.jsToLoad` are sent off as a single request, through QuickConcat.

I usually refer to QuickConcat as a pattern because it's rarely something that drops into a production environment as is. It's usually something that we'll hand off to clients for implementation in their back-end language of choice.

Thanks to QuickConcat, even if we're loading a handful of scripts and styles after the page is loaded, we're only adding two requests: one for all of our additional scripts, and one for our additional styles.

Down On SouthStreet

Filament Group has rolled all the lessons we've learned about optimizing delivery of HTML, CSS, JavaScript and images into a project we're calling SouthStreet[57], named for the location of FG's office.

SouthStreet provides you with a set of tools you can use to ensure that devices get the most efficient amount of code possible, while still maintaining broad accessibility and support. We're continuing to refine SouthStreet as new approaches and techniques come to light, and all of the projects mentioned in this chapter are completely open source: feedback, suggestions and new ideas are always welcomed.

---

57. https://github.com/filamentgroup/southstreet

The day BostonGlobe.com launched, we opened the site up on a number of devices which we never tested and definitely didn't plan for in advance: a first-generation Amazon Kindle, a Nintendo DS, the Playstation 3's built-in browser, and even an Apple Newton. At no point were we presented with a blank screen: no matter the context, we could use the website to the best the device could allow.

Every time a new device shows up for the jQuery Mobile test lab, we have a look at the Globe site — and so far, we're batting a thousand. No panicked emails about updating UA strings; no worrying about new mismatched features, or unplanned display sizes. There are new enhancements we could make as new browser features and APIs roll out, of course, but our foundation is solid and we're never limited by it. It *works* everywhere, for everyone. By following a few principles for serving assets responsibly, you can do the same.

Building websites is a complicated business, and it isn't an easy one. That's the nature of the game, not the fault of any of the tools or techniques we use. The most challenging part of developing for the web is simplifying: stripping away the inessentials. Responsive web design can ensure that we don't hinder the inherent flexibility of the web; progressive enhancement can ensure that we don't hinder the inherent usability of the Web. The very first page of the Web, to this day, works for users of any browsing context.

We can't expect to always get everything right; we're still going to bend a nail or two here and there.

Our tools aren't perfect either. But we can do better — we can always do better — and we can use the tools we've got to build amazing things. We might be a little clumsy right now, but we're just getting started. ❧

Performance Optimization Roadmap

BY VITALY FRIEDMAN

The following is an excerpt from the Smashing Book 5 chapter "Performance Optimization Roadmap."

Improvement is a matter of steady, ongoing iteration. When we redesigned Smashing Magazine back in 2012, our main goal was to establish trustworthy branding to reflect the ambitious editorial direction of the magazine. We did that primarily by focusing on crafting a delightful reading experience. Over the years, our focus hasn't changed a bit; however, the very asset that helped to establish our branding turned into a major performance bottleneck.

Deferring Web Fonts

Despite the fact that the proportion of Smashing Magazine's readers on mobile has always been quite modest (around 15%, mainly owing to the length of articles), we never considered mobile as an afterthought — but we never pushed user experience on mobile either. And when we talked about user experience on mobile, we mostly talked about speed, since typography was pretty much well designed from day one.

We had conversations during the 2012 redesign about how to deal with fonts, but we couldn't find a so-

lution that made everybody happy. The visual appearance of content was important, and because the new Smashing Magazine was all about beautiful, rich typography, not loading web fonts at all on mobile wasn't really an option.

With the redesign back then, we switched to Skolar for headings and Proxima Nova for body copy, delivered by Fontdeck. Overall, we had three fonts for each typeface (regular, italic and bold) totalling six font files to be delivered over the network. Even after our dear friends at Fontdeck subsetted and optimized the fonts, the assets were quite heavy with over 300KB in total. Because we wanted to avoid the flash of unstyled text, we had them loaded in the `<head>` of every page. Initially, we thought the fonts would reliably be cached in HTTP cache, so they wouldn't be retrieved with every single page load. Yet it turned out that HTTP cache was quite unreliable: the fonts showed up in the waterfall loading chart every now and again for no apparent reason, both on desktop and mobile.

The biggest problem, of course, was that the fonts were blocking rendering[58]. Even if the HTML, CSS and JavaScript had already loaded completely, the content wouldn't appear until the fonts had loaded and rendered. A visitor had this beautiful experience of seeing link underlines first, then a few keywords in bold here and there, then subheadings in the middle of the page and then, finally, the rest of the page. In some cases,

58. http://ianfeather.co.uk/web-fonts-and-the-critical-path

when Fontdeck had server issues, the content didn't appear at all, even though it was already sitting in the DOM, waiting to be displayed.

In his article, "Web Fonts and the Critical Path," Ian Feather provides a very detailed overview of the FOUT issues and font loading solutions. We tested them all.

We experimented with a few solutions before settling on what turned out to be perhaps the most difficult one. At first, we looked into using Typekit and Google's Web Font Loader[59], an asynchronous script which gives you more granular control of what appears on the page while fonts are being loaded. The script adds a few classes to the <body> element, which allows you to specify the styling of content in CSS during loading and after the fonts have loaded. You can be very precise about how the content is displayed in fallback fonts

59. https://github.com/typekit/webfontloader

first, before users see the switch from fallback fonts to web fonts.

We added fallback font declarations and ended up with pretty verbose CSS font stacks, using iOS fonts, Android fonts, Windows Phone fonts, and good ol' web-safe fonts as fallbacks — we still use these font stacks today. For example, we used this cascade for the main headings (it reflects the order of popularity of mobile operating systems in our analytics):

```
h2 {
   font-family: "Skolar Bold",
   AvenirNext-Bold, "Avenir Bold",
   "Roboto Slab", "Droid Serif",
   "Segoe UI Bold",
   Georgia, "Times New Roman", Times, serif;
}
```

Readers would see a mobile OS font (or any other fallback font first), and it would probably be a font that they were quite familiar with on their device. Once the fonts had loaded, they would see a switch, triggered by Web Font Loader. However, we discovered that after switching to Web Font Loader, we started seeing FOUT far too often, with HTTP cache becoming quite unreliable again; the permanent switch from a fallback font to the web font was quite annoying, ruining the reading experience.

So we looked for alternatives. One solution was to include the `@font-face` directive only on larger screens by wrapping it in a media query, thereby avoiding load-

ing web fonts on mobile devices and in legacy browsers altogether. (In fact, if you declare web fonts in a media query, they will be loaded only when the media query matches the screen size — no performance hit there.) Obviously, it helped us improve performance on mobile devices in no time, but we didn't feel right about having a simplified reading experience on mobile devices. So it was a no-go, too.

What else could we do? The only other option was to improve font caching. We couldn't do much with HTTP cache, but there was one possibility we hadn't looked into: storing fonts in AppCache or localStorage. Jake Archibald's article on the beautiful complexity of AppCache[60] led us away from AppCache to experiment with localStorage, a technique[61] that the Guardian's team was using at the time.

Now, offline caching comes with one major requirement: you need to have the actual font files to be able to cache them locally in the client's browser. And you can't cache a lot because localStorage space is very limited[62], sometimes with just 5MB available per domain. Luckily, the Fontdeck guys were very helpful and forthcoming with our undertaking; despite the fact that font delivery services usually require you to load files and have a synchronous or asynchronous callback to count the number of impressions, Fontdeck has been perfectly

60. http://alistapart.com/article/application-cache-is-a-douchebag
61. https://github.com/ahume/webfontjson
62. http://www.html5rocks.com/en/tutorials/offline/quota-research

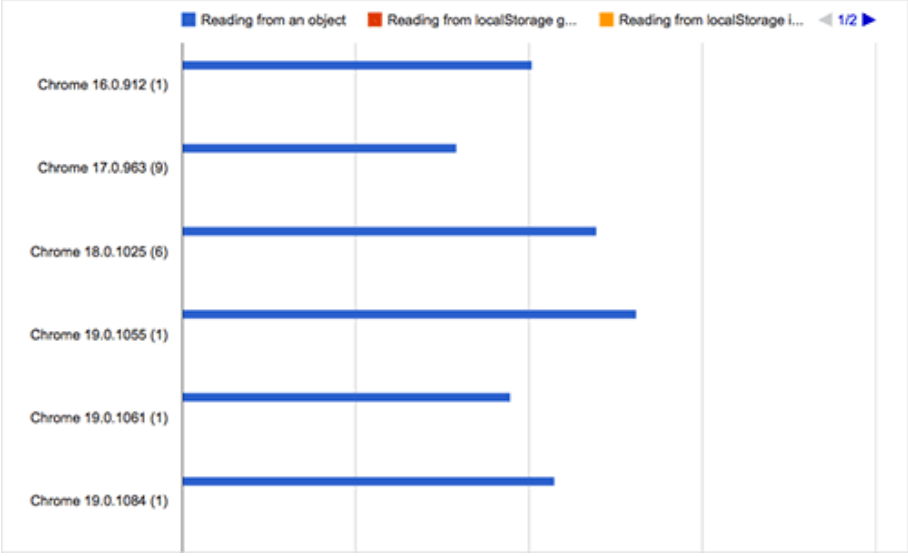
fine with us grabbing WOFF files from Google Chrome's cache and setting up a flat pricing structure based on the number of page impressions in recent history.

We grabbed the WOFF files and embedded them, base64-encoded, in a single CSS file, moving from six external HTTP requests with about 50KB each, to at most one HTTP request on the first load and 400KB of CSS. Obviously, we didn't want this file to be loaded on every visit, so if localStorage is available on the user's machine, we store the entire CSS file in localStorage, set a cookie, and switch from the fallback font to the web font. This switch usually happens once at most because for subsequent visits we check whether the cookie has been set and, if so, retrieve the fonts from localStorage (causing about 50ms in latency) and display the content in the web font right away. Just before you ask: yes, read/write to localStorage is much slower than retrieving files from HTTP cache[63], but it proved to be a bit more reliable in our case.

If the browser doesn't support localStorage, we include fonts with `<link href="">` and, well, frankly just hope for the best — that the fonts will be properly cached and persist in the user's browser cache. For browsers that don't support WOFF[64] (IE8, Opera Mini, Android <= 4.3), we provide external URLs to fonts with older font mime types, hosted on Fontdeck.

---

63. https://github.com/addyosmani/basket.js/issues/24
64. http://caniuse.com/#search=woff



Yes, localStorage is much slower than HTTP cache, but it's more reliable. Storing fonts in localStorage isn't the perfect solution, but it helped us improve performance dramatically.

Now, if localStorage is available, we still don't want it to block the rendering of the content. And we don't want to see FOUT every single time a user loads the page. That's why we have a little JavaScript snippet in the header before the `<body>` element: it checks whether a cookie has been set and, if not, we load web fonts asynchronously after the page has started rendering. Of course, we could have avoided the switch by just storing the fonts in localStorage on the first visit and have no switch during the first visit, but we decided that one switch is acceptable, because our typography is important to our identity.

The script was written, tested and documented by our good friend Horia Dragomir. Of course, it's available as a gist on GitHub[65]:

```
<script type="text/javascript">
    (function () {
      "use strict";
      // once cached, the css file is stored on the
      // client forever unless the URL below is
      // changed. Any change will invalidate the
      // cache
      var css_href = './web-fonts.css';
      // a simple event handler wrapper
      function on(el, ev, callback) {
        if (el.addEventListener) {
          el.addEventListener(ev, callback, false);
        } else if (el.attachEvent) {
          el.attachEvent("on" + ev, callback);
        }
      }

      // if we have the fonts in localStorage or if
      // we've cached them using the native browser
      // cache
      if ((window.localStorage &&
      localStorage.font_css_cache) ||
      document.cookie.indexOf('font_css_cache')
      > -1){
        // just use the cached version
        injectFontsStylesheet();
      } else {
```

65. https://gist.github.com/hdragomir/8f00ce2581795fd7b1b7

```
 // otherwise, don't block the loading of the
 // page; wait until it's done.
  on(window, "load", injectFontsStylesheet);
}

// quick way to determine whether a css file
// has been cached locally
function fileIsCached(href) {
  return window.localStorage &&
  localStorage.font_css_cache &&
  (localStorage.font_css_cache_file === href);
}

// time to get the actual css file
function injectFontsStylesheet() {
 // if this is an older browser
  if (!window.localStorage ||
  !window.XMLHttpRequest) {
    var stylesheet =
    document.createElement('link');
    stylesheet.href = css_href;
    stylesheet.rel = 'stylesheet';
    stylesheet.type = 'text/css';

document.getElementsByTagName('head')[0].appendChild(stylesheet);
    // just use the native browser cache
    // this requires a good expires header on
    // the server
    document.cookie = "font_css_cache";
```

```
        // if this isn't an old browser
        } else {
           // use the cached version if we already
           // have it
          if (fileIsCached(css_href)) {

injectRawStyle(localStorage.font_css_cache);
          // otherwise, load it with ajax
          } else {
            var xhr = new XMLHttpRequest();
            xhr.open("GET", css_href, true);
            on(xhr, 'load', function () {
              if (xhr.readyState === 4) {
                // once we have the content,
                // quickly inject the css rules
                injectRawStyle(xhr.responseText);
                // and cache the text content for
                // further use
                // notice that this overwrites
                // anything that might have already
                // been previously cached
                localStorage.font_css_cache =
                xhr.responseText;
                localStorage.font_css_cache_file =
                css_href;
              }
            });
            xhr.send();
```

```
        }
      }
    }

    // this is the simple utitily that injects
    // the cached or loaded css text
    function injectRawStyle(text) {
      var style = document.createElement('style');
      style.innerHTML = text;

document.getElementsByTagName('head')[0].appendChild(style);
    }

  }());
</script>
```

During testing of this technique, we discovered a few surprising problems. Because the cache's persistence varies in WebViews, sometimes fonts do load asynchronously in applications such as TweetDeck and Facebook; yet they don't remain in the cache once the application is closed (although they remain in the cache while the application is running!). localStorage isn't shared between the apps and the Safari browser. In other words, in the worst case, with every WebViews visit — let's say from a Facebook app or a Twitter app — the fonts will be redownloaded and rerendered. Some old BlackBerry devices seem to clear cookies and delete the cache when the battery is running out. And depend-

ing on the configuration of the device, sometimes fonts might not persist in localStorage either.

Still, once the snippet was in place, articles started rendering much faster. By deferring the loading of web fonts and storing them in localStorage, we've prevented around 700ms delay, and thus shortened the critical path significantly by avoiding the latency for retrieving all the fonts. The result was quite impressive for the first load of an uncached page, and it was even more impressive for concurrent visits since we were able to reduce the latency caused by web fonts to just 40–50ms. In fact, if we had to highlight just one improvement to performance on the website, deferring web fonts is by far the most effective.

This isn't a definitive solution though. In many ways, the technique feels like a heavy workaround. For example, it would be smarter to detect whether the browser supports the new WOFF2 format[66] (currently supported by Chrome and Opera), WOFF (most modern browsers, IE9+), or TrueType (Android 4.1–4.3 support) and store corresponding base64-encoded font files in the browser's cache, rather than serving and caching primarily WOFF and loading other font formats on demand. In fact, that's what Filament Group have been doing recently[67]: using a style sheet loader, a WOFF2/TrueType feature test and a cookie utility to load, cache

66. https://gist.github.com/sergejmueller/cf6b4f2133bcb3e2f64a
67. http://www.filamentgroup.com/lab/font-loading.html

and later access only supported font files from HTTP cache.

This is also an approach we are considering at the moment. In our case we would provide six font variants in three file formats, generating eighteen CSS files with base64-encoded web fonts, that would then be stored either in localStorage or in HTTP cache with a cookie set. A move to WOFF2 might be very much worth it, since the file format promises a better compression for font files and it has already shown remarkable results. In fact, the Guardian was able to cut down on 200ms latency and 50KB of the file weight[68] by switching to WOFF2.

Of course, grabbing WOFFs might not always be an option for you, but it wouldn't hurt just to talk to type foundries to see where you stand, or to work out a deal to host fonts locally. Otherwise, tweaking Web Font Loader for Typekit and Fontdeck is definitely worth considering. These techniques work well today, and they can significantly boost the performance of your website. In the long run, the Font Load Events API will (hopefully) replace them: think of it as the Web Font Loader library, but implemented natively in browsers. You can construct a `FontFace` object directly in JavaScript, specify how and when the font will be rendered, trigger an immediate fetch of the font files, and avoid blocking on CSSOM and DOM entirely. The API is already supported in Chrome and Firefox, and it's a

68. https://twitter.com/patrickhamann/status/497767778703933442

solution that is very likely to solve most web font performance issues for good.

Inlining Critical CSS

That still wasn't good enough, though. Performance had improved dramatically; but even with all of these optimizations in place, we didn't hit that magical speed index value of below 1,000. In light of the ongoing discussion about inline CSS and above-the-fold CSS, as recommended by Google[69], we looked into more radical ways to deliver content quickly. To avoid an HTTP request when loading CSS, we measured how fast the website would be if we were to load critical CSS inline and then load the rest of the CSS once the page had rendered.

But what exactly is critical CSS? And how do you extract it from a potentially complex code base? As Scott Jehl points out[70], critical CSS is the subset of CSS that is needed to render the top portion of the page across all breakpoints. What does that mean? Well, you would decide on a certain height that you would consider to be "above the fold" content — it could be 600, 800 or 1,200 pixels or anything else — and you would collect into their own style sheet all of the styles that specify how to render content within that height across all screen widths.

69. https://developers.google.com/web/fundamentals/performance/critical-rendering-path/page-speed-rules-and-recommendations
70. http://www.filamentgroup.com/lab/performance-rwd.html

Then you inline those styles in the <head>, and thus give browsers everything they need to start rendering that visible portion of the page — within one single HTTP request. Everything else is deferred after the initial rendering. You avoid an HTTP request, and you load the full CSS asynchronously, so once the user starts scrolling, the full CSS will (hopefully) already have loaded.

Content will appear to render more quickly, but there will also be more reflowing and jumping on the page. If a user has followed a link to a particular comment below the initially loaded screen, they will see a few reflows as the website is constructed; the page is rendered with critical CSS first (there is only so much we can fit within 14KB!) and adjusted later with the complete CSS. Of course, inline CSS isn't cached. If you have critical CSS and load the complete CSS on rendering, it's useful to set a cookie so that inline styles aren't inlined with every single load. The drawback is that you might have duplicate CSS because styles would be defined both inline and in the full CSS, unless you're able to strictly separate them.

Because we had just refactored our CSS code base, identifying critical CSS wasn't very difficult. There are smart tools that analyze the markup and CSS, identify critical CSS styles and export them into a separate file during the build process; but we were able to do it manually. Again, you have to keep in mind that 14KB is your budget for HTML and CSS, so in the end we had to re-

name a few classes here and there, and compress CSS as well.

We analyzed the first 800px, checking the inspector for the CSS that was needed and separating our style sheet into two files – and actually, that was pretty much it. One of those files, *above-the-fold.css*, is minified and compressed, and its content is placed inline in the `<head>` of our document as early as possible, so as to not block rendering. The other file, our full CSS file, is then loaded by JavaScript after the content has loaded, and if JavaScript isn't available for some reason, or the user has a legacy browser, we've put a full CSS file inside `<noscript>` tags at the end of the `<head>`, so the user doesn't get an unstyled HTML page.

Finally, we gradually tweaked the amount of inline CSS to avoid drastic reflows as the page is constructed. We do load some CSS for styling the comments area, in case the displayed page is a permalink to one of the comments. We keep refining and prioritizing what does and what doesn't have to be inlined; with every refactoring session, the amount of CSS in the `<head>` gets smaller — which obviously is a pretty good sign.

Conclusion

Finding the right strategy to make our website fast took a lot of experimentation, blood, sweat and cursing. Our discussions kept circling around next steps and critical and not-so-critical components, and sometimes we had to take three steps back in order to pivot in a different

direction. But we learned a lot along the way, and we have a pretty clear idea of where we are heading now and, most importantly, how to get there. ❧

Becoming A Mobile Web Developer

BY PETER-PAUL KOCH

The following is an excerpt from the Mobile Web Handbook chapter "Becoming A Mobile Web Developer."

What exactly is a mobile web developer? By now most web developers will have looked at their sites on their own mobile phone, and even solved an iOS or Android bug or two, but that's not enough. To me, a mobile web developer is someone who spends a lot of time on mobile browsers, and for whom Android WebKit compatibility is as important as IE compatibility.

The most important advice I can give you is to start doing mobile testing right now. You probably have your own iPhone or Android device: use it to look at your current project. Now. Also, download Opera Mini and test in that browser, too. If you have an Android device, download a few more browsers and test in them, too. I advise Chrome, Firefox, and UC. All of them are available in Google Play — though, as we saw in the Browsers chapter, they don't exist for any of the other platforms. (Remember: Chrome on iOS actually uses the Apple WebView, and not Blink.)

True, your single device is not representative of the market as a whole, but any mobile test is better than no mobile test. Besides, even one single browser on one de-

vice allows you to get acquainted with the small screen and responsive web design.

The Ideal Device Lab

Your first job will be to create a device test lab. Here's the ideal lab as of summer 2014:

1. At least half your device lab will be **Android**s. We'll get back to them below.
2. **iOS**: at least one iPhone and one iPad; possibly also an older iPhone (with less capable hardware) or an iPad Mini (with a smaller screen). It's useful to have one device on an older version of iOS; a few users won't upgrade, either because they don't want to or because their device is too old. In fact, it's useful to have an old device yourself, so that you're sure your site will also run in adverse conditions. Also, make sure you have one Retina device for those resolution and responsive images tests.
3. **BlackBerry**: especially in the UK, where BlackBerry still has a browser market share between 5 and 10%, it's important to have one or more of them for testing purposes. I'd say one BB10 device (Z10 or Q10 or even newer), and an older one with OS6 or 7. If you can get your hands on a non-touchscreen one that would also be useful.
4. **Nokia** is complicated because it currently supports three different platforms with varying levels of global

market share. What you will need in any case is a Windows Phone, preferably a new one. Windows Phone has a fairly low market share but it's slowly rising and you do want to test in IE Mobile. If you have enough money, buy a Windows Phone 8 (with IE10 Mobile) and a Windows Phone 8.1 (with IE11 Mobile); increase both by one version number if IE12 Mobile is available by the time you read this.
As for the other Nokia platforms, that depends on whether you will do a lot of work for Africa, Asia, or Latin America. If you do, it's good to have an Asha (S40) with Nokia Xpress handy, since they are still being used a lot. Symbian is a dying platform. If you happen to have a Symbian phone or can get one cheaply, add it to your line-up. If you don't, don't bother. (By the way, even at its zenith Symbian was not important in the US.)

5. **Windows 8**: at least one Windows 8 tablet: either a Microsoft Surface or one from the other vendors. Windows 8 tablets are different from phones, both in support (no meta viewport, for instance), and in the fact that they support touch, mouse, and keyboard and you can use the three interaction modes interchangeably.

6. Then, the **minor OS**s: Firefox OS, Tizen, Amazon Kindle, and Sailfish by Jolla. They don't have market share to speak of at the time of writing, but that could change. Keep an eye on them and add them to your lab when necessary — and if you can get one on the cheap, do it.

UPDATES

Be careful with browser or OS updates, since they can disturb your test setup. I never update anything while going through a test, but since I run compatibility tests and you will likely run website tests, your mileage may vary. Set a rule at the start: either update everything straight away, or postpone all updates until your current test run is finished.

Android has a complicated update timetable. The other OSs are usually less hassle. iOS users, in particular, tend to update their devices quickly, although a small minority will not be able to update due to their device being too old.

I alternate updates between my iPhone and iPad. At first my iPhone ran iOS5 and my iPad iOS6, and when iOS7 was released I updated my iPad, but not my iPhone. Thus I always have the latest two versions available. You should establish a similar rule for your iOS devices.

As for downloadable browsers, they will update far more frequently than device OSs. Install every update, just as you would on desktop.

ANDROID

The main characteristic of Android is its differentiation. Whenever you buy an Android device, make sure it comes from a different vendor and has a different screen size and Android version than the devices you already own. You could make an exception for the Sam-

sung Galaxy range, or any other model that has a very high market share at the time you read this.

Once you've bought the device, test the default browser carefully to establish its identity and make a note of it. Remember: any browser that has `Chrome` in its user-agent string (`navigator.userAgent`) is Chrome, though not necessarily Google's Chrome, while any default browser that hasn't is Android WebKit.

One of your purposes is to have at least one of each default browser available, and two or three Android WebKit 4s.

So a good Android lab consists of the latest high-end Samsung Galaxy, maybe an earlier high-end Galaxy, an HTC, a Sony, and an LG. Grab a Chinese Android (preferably a Lenovo, Huawei, or Xiaomi) if they are available in your market, and add Motorola to the list if you're in the US. Make sure at least one of these devices is a mid-range one (say €100–150), and at least one runs Android 2.

Owning a Google device is not a top priority. Despite their popularity among web developers, normal consumers don't buy all that many of them, and the Google Chrome that runs on these devices is not representative of Android default browsers of other vendors. I advise you to postpone buying a Nexus until you have three or four non-Google devices.

Once you have these devices, install the other browsers on them: Chrome, Firefox, Opera Mobile, Opera Mini, and UC are the important ones, but as long

as you're at it you should add UC Mini (proxy browser; very popular in China), QQ (also called One; Chinese), Puffin (Korea), and anything else you can lay your hands on. The purpose is not so much making sure your site runs perfectly (though that is a definite bonus), but getting acquainted with the odd things mobile browsers may do to your site. And all these browsers are free.

Spread out these browsers across your Android devices — don't install them all on one device. Once you get to the actual testing you want to be able to use several devices side by side.

What To Test

OK, so you have the perfect device lab. Now, what will you test? Obviously, you start with basic website behavior, just as on desktop. Do the CSS and JavaScript work? If not, how do you solve the problem? Do your responsive design breakpoints need adjustment? Maybe the two-column layout should kick in at a slightly larger viewport width? You can figure this out for yourself.

There are a few things that are different from desktop, though, so a really thorough mobile testing procedure includes the following:

- In addition to Wi-Fi, test your site over a data network — ideally 3G or better, but also on 2G if many people in your target region can't get anything else. If you're really thorough, you test on several networks. A real net-

work connection can lead to unexpected situations such as a blazing fast connection directly followed by no connection at all. (If you're really thorough, test in a moving train where circumstances change all the time.) What happens when the connection suddenly fails?

- Test in both portrait and landscape orientation. The ideal viewport will change with the orientation in most browsers. That's not all, though. How do fixed layers behave? Maybe there are problems with repositioning or recalculating a specific element, especially modal windows and complex items such as image carousels. Open a modal window or use a carousel, switch orientation, and see how it behaves.

- Test your interactions, especially when they involve custom gestures. Do all gestures work properly on all devices?

- Form elements merit special attention, since usually they're linked to critical transactions such as paying for something. They have to work flawlessly in all browsers, both orientations, and when the user zooms in. Make sure to actually fill out the form and try to guess what the user will do once the software keyboard appears. Pay special attention to heavily styled form elements or components such as calendar widgets. Do they work in all browsers and orientations?

How To Test

You'll quickly find that testing your sites on mobile devices is much more time-consuming than on desktop, and not only because there are more mobile browsers. Below is some advice for mobile browser testing based on my own experience. Please don't get too hung up on the details; it's perfectly fine if you structure your testing differently. Novice mobile testers will find some useful hints here, though.

TIME

Testing something on mobile devices takes more time than you think, even if you start out by assuming it'll take more time than you think. There is no such thing as a quick mobile test. Allow them to go way over time if necessary. Do not start a quick test 15 minutes before you're supposed to go home. You won't go home on time.

PREPARING THE DEVICES

> *For an overview of mobile testing see http://smashed.by/mwhb15, where Addy Osmani talks about several tools.*

There are certain preparations you have to make before starting the actual tests. The most important one is some sort of syncing solution. You want to be able to click through your site on your desktop computer, with

the phones following along. This greatly cuts down testing time, since you don't have to perform every click on every phone. At the time of writing there are two major tools for syncing:

- Ghostlab, which requires you to add a script to your page. See http://smashed.by/mwhb16 for more information. Don't forget to remove the script once your tests are complete.
- Adobe Edge Inspect syncs iOS and Android devices to your desktop Chrome. You can find more information at http://smashed.by/mwhb17.

On every device, set the screen timeout to its maximum value. This timeout is for switching off the screen after a period of inactivity, and switching the phone on before reloading your page becomes annoying after a while. The timeout can generally be found in the display settings of the device.

Add an icon to the home screen for every browser. You'll usually do this, but forgetting it even once can lead to problems down the line. If you can change the icon text, note the browser version.

Make sure all devices are charged. This sounds like a no-brainer, but I found that the only way to actually make sure is to be pretty disciplined about it. Plug in all phones in the next batch before starting on your current batch (see below for batches): this ensures that the phones are ready when you are. Another no-brainer: make sure you have multiple power sockets available.

You don't want to charge just one phone, but up to eight or so.

Nowadays, nearly every device has Wi-Fi capability. Nonetheless, a few devices (notably Windows Phones) can be fairly slow in setting up a connection even to a known access point. Make sure to switch these phones on a minute or so before the actual testing starts. In case you test on non-Wi-Fi phones, make sure to insert a SIM card and start them up a few minutes before you need them.

TESTING IN BATCHES

Once you get beyond five or six browsers to test in, it becomes useful to divide them up into batches and test one batch at a time. Count every instance of Android WebKit as a separate browser, as well as every version of any browser. The purpose of batch testing is not to be overwhelmed by bugs and oddities, but to solve them one or two at a time.

Make an ordered list of devices and browsers you want to test. I encourage you to write down this list, including device names and exact version numbers of OSs and browsers. This will become an invaluable reference in the later stages of testing, when your enthusiasm is flagging and your true interest lies in throwing mobile devices around the room. By then you won't be able to remember what you should test next, and looking at your list will save you a lot of headaches.

Once you have that list, divide it into batches. The first batch should be a mixed one, while the rest should

have a common theme (Android, Opera, and so on). The purpose of the mixed batch is to get a quick overview of whether your code is going to work across different browsers or not. If you find a lot of problems in the first batch, you should probably choose another approach. If you don't find many problems you're on the right track and you can continue with detailed tests in the other batches.

Each batch should contain between three to eight browsers. Make sure that in every batch each browser runs on its own device. You do not want to switch to different browsers on the same device since that takes way too much time and will become confusing after a while.

When you initially create a page you should test constantly in the mixed batch of about four to five browsers. This batch should contain one Safari, one Android WebKit (not Chrome!), one Opera Mini, one browser that's neither iOS nor Android, and maybe one or two other browsers that are important for your client's target audience.

Once your page works in this first batch, the time comes to test it in all browsers on your list, batch by batch. The problem here is that if you notice a problem and change the page, you have to go back to the beginning and test the new version in all previous batches. Therefore, it's best to start with the most problematic browsers that will likely need many adjustments: Android WebKit, IE, and the proxy browsers.

So a possible batch list could look like this; adjust to taste and device lab:

1. The mixed batch of one Safari, one Samsung Android WebKit 4 (not Chrome!), one Windows Phone, one Opera Mini, and one Chrome.
2. A batch of Android WebKits: all Android devices you have available. If you have more than one Android 2 device, do Android 2 first, then Android 4. Make sure these are all Android WebKits, and not Chromes.
3. A batch of IEs and proxy browsers: say IE10, IE11, Opera Mini on two or three devices, Nokia Xpress, UC Mini. By the time you finish this batch you'll have found many problems and will likely have started again a few times with the Android WebKit batch.
4. If you have them, a batch of unusual browsers like UC, QQ One, Tizen, Puffin, and game consoles. You may want to ignore a few bugs for these rare birds. Don't tell anyone.
5. Finally, the easy browsers: Safari on all iOS devices you own, BlackBerry WebKit, Chrome, Firefox, and Opera Mobile. These browsers usually behave decently and shouldn't cause too many problems, which is why they should go last.

Don't get too hung up on these exact batches, but I hope you understand the principles. Take a few hours to design the batch list; this overhead time will pay itself

back many times over when you're in the thick of mobile testing.

TESTING PROCESS

Once you get to the actual testing, the following tips and tricks will help you:

1. Use simple URLs. You don't want to type in `192.168.17.49:8080/testsite/default/Default.aspx` twenty times on software keyboards. I use `quirksmode.org/m` as a standard page and add links to whatever I want to test.

2. Make sure you're testing in the right browser. This sounds a bit silly, but if you have three or four browsers on one device, you may accidentally start up the wrong one and think you're testing in Opera Mobile while you actually have Android WebKit open. This has happened to me several times.

3. Be very finicky and precise with testing gesture-based interactions. Make sure that in each browser your gesture is exactly the same. If it's not, you might find differences due not to the browsers but to slight differences in your gestures. In general, it's best to predefine gestures, and make sure their start and end points are visible on the page; for instance, "swipe from the bottom-left corner of element X to the top of the screen."

4. When you're testing responsive designs it may be useful to see the viewport width and height onscreen. Print out `document.documentElement.clientWidth/Height`. Make sure to do it again `onresize` and `onorientationchange`; these events usually fire when the viewport dimensions change.

Overcoming Outdated Reflexes

There are some reflexes from traditional desktop web development that we have to let go of. The most important ones are our distrust of browser detection and our overuse of JavaScript libraries.

BROWSER DETECTION

Traditionally, browser detection is a no-go for web developers. If you distinguish between IE and Chrome through their `navigator.userAgent` strings you can expect some pointed questions. Instead, we've learned you should detect the feature you want to use, and make decisions based on the result of that check.

I have been preaching feature detection since 1998 and played my part in convincing web developers of the perils of browser detection, so it took me a while to acknowledge that the situation on mobile is sometimes different. As soon as I started talking to people with years of experience in mobile, they all told me that some sort of browser detection is a necessary evil. The more experienced a mobile web developer is, the more they rely on server-side browser detection because fea-

ture detection doesn't help in certain cases. Consider:

- Suppose you need "Back" functionality on your website. Certain OSs, such as Android and BlackBerry, have a native "Back" button, and inserting your own button would only confuse users — or, worse, confuse the OS's "Back" functionality.
- Some Android devices claim to support `input type="date"` and such, but don't actually have the native components to fill in a date. BlackBerry 6's default browser supports touch events and tells you so — even if it is running on a device without a touchscreen.
- A browser might support animations and transitions but have a poor GPU (or none at all) so that everything slows to a crawl, and the user would be better off without them.

In all these cases, the problem is not with the browser's support for CSS and JavaScript but with physical device characteristics or specific OS functionalities. Detecting the presence of a "Back" button is impossible, and in the other examples the feature detection would return a false positive, since the browser only indicates it supports the feature, but not how bad that support is.

If you encounter use cases like this, it might be time to start detecting browsers. This is something you need a bit of experience with. The point here is not that you should use browser detection for everything, but that there are certain device features that are undetectable

in any other way. I'm not saying you should ditch feature detection, but you may encounter situations where browser detection is a necessary addition to your regular feature detection.

If you decide you need a browser or device detection script, don't write your own. Knowledgeable people have already done the work for you. There's a whole ecosystem of device detection services, of which WURFL[71] and DeviceAtlas[72] are the best known. Hand it the UA string of a mobile browser, and it will tell you something about the browser's and the device's capabilities. If you're looking for a pure browser detect, without device information, try WhichBrowser[73].

JAVASCRIPT LIBRARIES

The second outdated reflex is to use a JavaScript library for absolutely everything. This is the sad result of the overreliance on libraries that we developed in the 2006–2011 era, to the extent that some web developers can't even write JavaScript anymore.

I've always had reservations about JavaScript libraries, and they were reinforced by a research paper[74] from April 2012. The researchers measured the battery use of an Android phone while loading several websites, including Wikipedia, and experimented with redesign-

71. http://smashed.by/wurfl
72. http://smashed.by/atlas
73. http://whichbrowser.net
74. http://smashed.by/wwwcon

ing one function of that website. Wikipedia's accordion script, which uses jQuery, was replaced by a custom-made function, and the measured energy consumption for downloading and rendering the page fell by one-third.

The size of the download isn't even the issue: library vendors are well aware of that problem and have taken steps to make their files as light as possible. The problem is that the entire library has to be executed, draining the battery as feature after feature is initialized — and your page might not even use most of the features.

The solution to this problem is not to relegate JavaScript libraries to the ash-heap of history, but to ask yourself whether you really need one. If you're building a complex interface with lots of functionalities, the answer will likely remain yes. If you just need one basic effect such as a show/hide toggle or simple form validation, it's time to write the entire script by hand. Not only will that sharpen your JavaScript skills, but it will also make your site perform better. Don't worry about browsers: all of them support simple effects well; it's only when you need complex ones that they start to behave erratically and a library becomes a useful addition to your site.

The Mobile Network

Mobile networks were set up to accommodate devices that have to be reachable only on occasion — when a device makes or receives a call, and when it sends or re-

ceives an SMS. Setting up a mobile connection takes some time, and if the mobile connection remains idle for a while, it is closed down in order to save battery life.

> *Steve Souders did the fundamental research on mobile connections. Read his conclusions at http://smashed.by/mwhb18.*

This principle also goes for data connections: when the browser requests assets, it takes roughly two seconds to set up a mobile connection, which then closes down after five to twelve seconds of inactivity. Two seconds might not sound like much, but compounded with the normal latency of a mobile network and web server, it makes for an annoying wait.

Again, there's little you can do about this, except for one thing: if the user needs data anyway, take the opportunity to load as much as you can. This sometimes means making an educated guess as to what data the user is going to need next, and you may occasionally guess wrong, but that's still better than reopening a mobile connection every time the user needs a tiny bit of data. You could decide to store some data, or even things like web fonts, in the browser's localStorage.

CONNECTION SPEED

The most serious problem you can encounter on mobile is a slow connection. While desktop connections are generally reliable in the sense that they don't change a lot, mobile connections may vary immensely if a user is on the move. Besides, it's almost impossible to find out anything that's not instantly obsolete about your users' connection speeds. It's essentially an impossible problem to solve.

Instinctively, web developers assume that a 3G or even a 4G connection is slower than a Wi-Fi connection. This may be true most of the time, but not always. Sometimes a user is in a public space with Wi-Fi, but it's slow or unreliable Wi-Fi used by many people simultaneously. It could also be that in one country the 4G network is overused and slow, while another country has a brand-new 4G network that doesn't yet have all that many users — and thus blindingly fast connections.

In such cases, the user's mobile connection might actually be faster. Do not fall into the trap of assuming that a user on 3G has a slow connection speed, or that a user on Wi-Fi has a fast one. Connection type is not a proxy for connection speed.

Although measuring connection speed is in fact not all that hard, the problem is that the result is worthless. The connection speed may be decent at the moment you measure it, but what if the user is on the move and goes from cell tower to cell tower — or from Wi-Fi to a mobile connection? Or maybe reception is perfect right

now, but the user's train is about to enter a connectionless tunnel. Or the user may reach their roaming limit and the connection may suddenly disappear. Although you can detect all that, you can't define a general download policy in such a changing environment, so I advise you not to try. ❧

Grassroots Change

BY PAUL BOAG

The following is an excerpt from the Digital Adaptation chapter "Grassroots Change."

Reaching Colleagues

I looked out over the audience of hundreds of web designers and took a deep breath. I am not normally somebody who gets nervous speaking, but this audience had proved vocal when they didn't like something. I knew they would hate my next sentence. Stealing myself for the inevitable backlash on Twitter, I said: "I believe that as web designers we should never say *no* to our clients." Sure enough there was a sharp intake of breath and the rapid taps of indignant tweets being composed.

Despite the negativity I get every time I echo this statement, I still stand by it. Whether working with internal stakeholders or paying clients, we shouldn't simply say *no* when they request something, no matter how inappropriate it is. For me, saying *no* fails on two counts. It fails to educate and it fails to build bridges. Both of these are fundamental requirements if we want to have anything but a superficial impact on our organization.

Let's start with the subject of building bridges.

Saying "no" to requests damages your reputation internally and drives people toward using external contractors.

BUILDING BRIDGES

When somebody in our organization comes to us and we say no, we are closing a door. *No* is a dead-end statement. It leaves our colleague with one of two options: walk away disappointed; or argue. Neither of these are positive outcomes.

Have you ever wondered why so many companies pay external consultants like me, when they already have a perfectly competent in-house team? Frequently it is because the internal team is seen as an obstacle that has to be worked around. All too often, on being hired by a company, I am quietly taken aside and told what a problem the web team has been.

If you want to change your company, this perception has to change first. You have to build bridges and find allies. You must listen and discuss matters with your colleagues, rather than being a blockage they have to work around.

Start by working with, rather than against, colleagues. When they come to you with an idea that you are unsure about, sit down and discuss alternatives. Maybe bring in other people and form a working group to explore the best approach. If you can't do it for practical reasons, explain those and try to come to a compromise.

Start engaging with colleagues even before they come to you. Ask them for advice and look for chances to collaborate. The more you demonstrate you value their opinion and expertise, the more likely they are to take you seriously.

Be careful not to just favor those who already like you. Reach out to people you consider difficult too. I know this will run counter to everything you want to do, but it is right. The more you help them, the harder you make it for them to remain an obstacle. They begin to look unreasonable and even downright obnoxious.

This may sound like office politics, that in some way you are manipulating people or building alliances. Nobody likes the idea of office politics or wants to play that game. But that is not what we are talking about. We are talking about building relationships, helping colleagues, and educating people about your role. Ultimately, if you are there for your colleagues across the organization then they are much more likely to be there for you. If that sounds like office politics, then so be it.

Offer advice and support. Try to identify ways that digital can help fulfill their objectives and show them how you can help. Not only will this provide you with

an opportunity to build bridges, it will also give you an opportunity to educate.

EDUCATING COLLEAGUES

Saying no to an idea not only alienates people, it fails to educate them. It does nothing to increase the understanding of digital, or move you closer to being a digital company.

Within traditional organizations at the early stages of digital adoption, the number one priority of the digital team should be to educate others in the company. This means taking the time to communicate clearly why their ideas might not be appropriate and to do so in language they can understand. However, it goes beyond even that.

As digital professionals we need to instigate a concerted campaign of education within our organizations. In particular, you should have four objectives in mind.

1. Highlight best practice

Show colleagues examples of organizations which are doing a good job with digital, and explain why. This helps them better understand why you want to pursue the direction you do and provides evidence that the direction works. This is particularly effective when those examples are taken from the competition.

2. Destroy preconceptions

Many of your colleagues will have formed incorrect preconceptions about what makes good digital practice. Typical examples of this might include that users hate scrolling, or that to appear high in Google search results you must stuff a site with keywords. You need to gently correct this thinking with evidence to the contrary.

3. Promote your successes

By drawing attention to your successes and explaining why they worked, you not only improve the perception of your team, you also educate colleagues about what works and why.

4. Explain failures

We tend to cover up our failures. However, if you want to build a culture that embraces failures, you must learn to share them as widely as possible. Doing so not only helps shape your culture, it also can be a learning opportunity if you explain why your failures happened and discuss how they could be avoided in future.

There are lots of ways you can educate people about digital best practice and you need to pick the right ones for your organization. Over the years, I have seen people launch internal newsletters and blogs, offer one-to-one training or group workshops, and run open usability testing sessions. This last one is particularly effective as nothing is more educational than seeing real users

encountering real problems with your website or other digital asset.

That said, the educational approach that most impressed me was an internal conference at a higher education institution. The conference was organised by Nicola, who has since become a role model of mine because of her impressive people skills. Nicola was an accomplished project manager who had been given the task of getting the entire institution moving in the same direction over digital. It was a formidable task in an organization with little central control. Despite her impressive political and organizationals skills, she could not force people to change — she had to persuade them. Instead of running multiple workshops, she bravely decided to hold a one-day conference with well over one hundred internal stakeholders.

She set a date and invited a massive cross-section of the institution, including most of the senior management team. She arranged guest speakers from both inside the institution and outside, and hoped people would turn up. The response was impressive with not only lower-level people attending, but a large number of the senior management team. This was all the more impressive as she made no effort to accommodate their diaries, instead setting a date that was most convenient. When I asked about this, she said that the more you try to accommodate the management team, the less likely they would be to attend. As I said, Nicola was very astute.

What made the conference work so well was the buzz it created. The inspirational talks, lovely venue, and large crowd of people created real excitement that succeeded in giving the project momentum. Not only were people enthused about the potential and direction of the project, they were educated about best practice as well. What made me smile the most was that in true project manager style Nicola proudly told me that the conference was considerably cheaper than arranging a series of smaller workshops, despite the lavish venue.

I am not suggesting you should run out and arrange a conference. What you should do is start engaging and educating colleagues about digital best practice. This is a fundamental part of our job that we often fail to recognize. These programs of education should not just be aimed at your colleagues; they should also (as demonstrated by Nicola's conference) be targeted at management. Because sooner or later, if you want to instigate real change, you will have to get management on board.

Convincing Management

I would like to tell you that if you follow certain steps, success with senior management is guaranteed. Unfortunately, there are no such guarantees, no such steps.

Working with senior management to help them understand the potential of digital and to agree a direction is often a painful, time-consuming, and frustrating process. It will include many setbacks and require substantial patience on your part, but it is important to un-

derstand that they are not being intentionally difficult — they simply see the world in a different way.

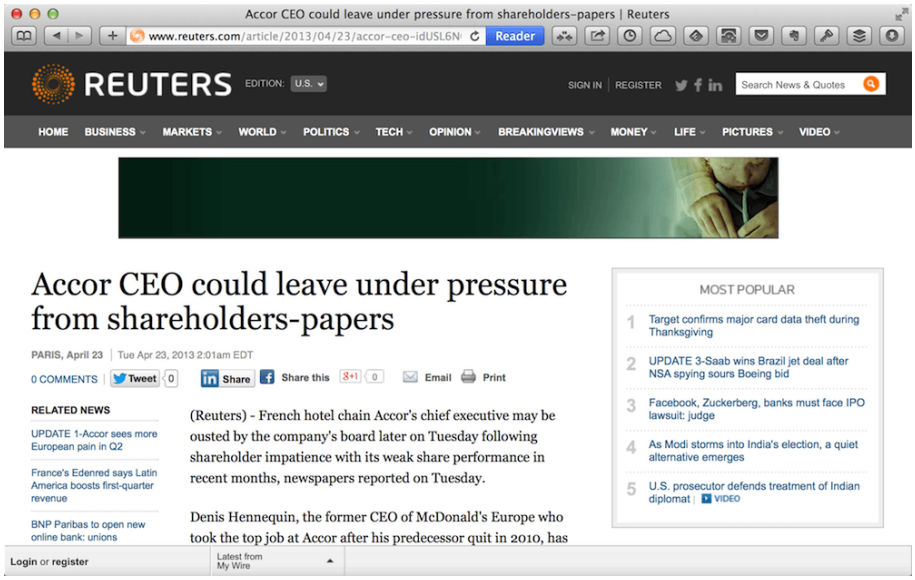
If you want any hope of moving your management team into the digital age, you will need to shelve the frustration and take the time to really understand them. Getting frustrated with them will not help, but learning more about them will.

UNDERSTANDING MANAGEMENT

Those of us who work on the web like to pride ourselves on our ability to empathize with users. We obsess about getting inside users' heads and understanding their motivation. Yet we rarely apply that skill to our colleagues or management. Taking the time to understand senior management will go a long way to influencing their digital thinking. It will allow us to see the reasons they are reluctant to be more aggressive in making changes.

For example, you would be forgiven for thinking that senior management staff are the top of the corporate tree and so free to take the company in whatever direction they wish. That is rarely the case. They are often subject to substantial pressures from investors to provide short-term revenue that improves dividend payments.

For public companies things are even worse. Independent analysts often make optimistic predictions of future revenue that when failed to be met by the company depresses share prices.



Understanding that senior management staff are under pressure to perform and meet certain targets is crucial in knowing how to communicate effectively with them.

Finally, senior management's salary and bonus packages are often tied to performance. This means they are sensitive to change that will impact negatively on achieving their targets. That said, if you can demonstrate a capability to help them reach or exceed those targets, you will find them much more amenable.

In essence, if you wish management to take digital seriously, you need to know their motivating factors and speak about digital in terms they understand.

SPEAKING THE LANGUAGE OF MANAGEMENT

As I have already said, I once worked with a major UK charity with an appallingly low conversion rate. When I worked with them, they were in the middle of a national marketing campaign designed to drive large amounts

of traffic to their website. The tactic of regular, large-scale marketing campaigns had been the bedrock of their marketing strategy for a number of years. However, as I looked at the site, I concluded that their focus should be on conversion and not driving even more traffic to the site. In my eyes they were failing to convert the traffic they had, so why drive even more?

This was a hard argument to make to management. They had invested heavily in forming the infrastructure to support advertising campaigns, and also had mentally committed to a program of future campaigns over the next eighteen months. How could I justify such a major change in direction and the costs associated with that? The answer was clear. I had to demonstrate that the change in strategy would generate a significantly higher return on investment to justify the pain and cost.

Analytics held the answer. Their conversion rate lay at less than half a percent. By looking at other charities we had worked with, I concluded that it would be reasonable to expect a conversion rate of 1.5% if we implemented a program of website improvements. After looking at the numbers with senior management it became apparent that this would equate to an extra £2 million in donations (a increase of 440%) over twelve months. To achieve the same figure on the existing site with advertising alone would mean driving an extra 33 million people to the site!

On top of this, improvements to the site would also address senior management's concerns over the 30% of

the traffic that came from mobile devices. Improving conversion rates would require a significant redesign of the website, and so we could use the opportunity to make the site responsive as well.

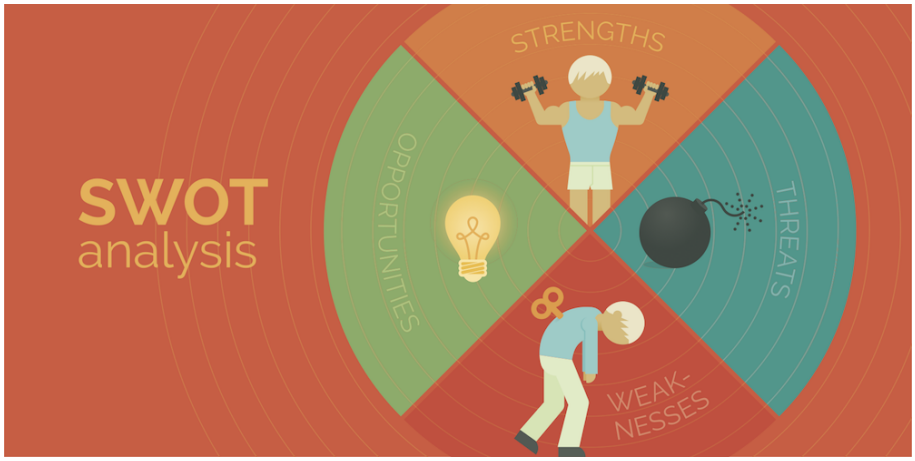
My proposal was approved for two reasons. First, it demonstrated a strong case for a significant return on investment. This is something that all management teams are looking for and a key component in any proposal for change. Second, it played to their areas of interest. The senior management team was concerned about mobile and I was able to address this concern as part of my proposal. To me this was a secondary issue (although an important one), but by mentioning it as part of my proposal it helped convince management. By speaking in terms that senior management could identify with (return on investment and topics they are interested in), I was able to push through a significant change in approach.

Another established technique for communicating with management is to produce a *SWOT analysis*. Like return on investment, this speaks a language that senior management will recognize and respond to.

HIGHLIGHTING THE THREATS

A SWOT analysis looks at an organization's strengths, weaknesses, opportunities, and threats. In our case, this is strengths, weaknesses, opportunities, and threats in the digital arena, but it can be applied to any part of the business.

Strengths and weaknesses refer to internal factors within the company, while opportunities and threats are external.



A SWOT analysis is a powerful tool for communicating with management.

To give you a more concrete example let's look at how a SWOT analysis would apply to Wiltshire Farm Foods. (*Editor's Note:* The e-commerce site was already mentioned as an example in a previous chapter of Digital Adaptation.)

Among Wiltshire Farm Foods greatest strengths are its drivers who deliver meals. Elderly people who ordered online were very nervous about who would turn up at their homes, but because all their drivers were police-checked this proved a strong selling point and was something we emphasized strongly online.

Its biggest weakness was the franchise business model that allowed individual pricing in different regions. This was a huge problem online because it pre-

vented us showing a single price on the website for a very long time.

Outside opportunities lay in individual carers who looked after several elderly people. These carers bought large quantities of meals and were long-term customers. We took advantage of this opportunity by building tools that made the ordering process for these users considerably easier than our competition.

Sadly, the biggest threat to the company was the age of the audience. Because customers were approaching the end of their lives, Wiltshire Farm Foods would have to constantly seek new customers, and things like national TV advertising proved expensive. Digital helped address this threat through cheaper online marketing campaigns and search engine optimization.

In my experience, the most powerful part of a SWOT analysis for encouraging a change in approach to digital are the weaknesses and threats.

Focusing on strengths and opportunities works well for smaller organizations hungry to get a competitive advantage. But larger, established organizations are more interested in maintaining their positions than grasping new opportunities. Fear is a powerful motivator, and one that only increases the more there is to lose. I find that senior management are particularly responsive to perceived threats to the sector or from competitors. It makes sense to present your digital strategy within that context.

For example, if you can demonstrate that digital has a fundamental impact on the way the sector works

(something that is increasingly common these days), you are much more likely to find management receptive to change.

Even more effective is demonstrating that the competition has a competitive lead in digital. What makes this such a powerful tool is that not only does it help address management's fears, but also our in-built tendency to follow the crowd. After all, if our competitors are investing in digital then surely it is worth investing in.

We like to moan when senior management fails to see the potential of digital or doesn't recognize the need for change. But moaning will achieve nothing and isn't justified. If senior management fails to see the need for change in digital, it is because we have failed to present a convincing argument. It is our job to present the argument in terms they will understand and respond to, by speaking about return on investment, focusing on their objectives, and demonstrating the threats they face.

You may say that it is hard to justify the particular changes you have in mind using those criteria. However, I would argue that if you cannot show a return on investment, demonstrate a threat, or establish how your ideas fit into senior management's broader objectives, then perhaps your vision for digital is wrong.

The journey of helping your organization adapt to the digital economy is not an easy one. Like me back at IBM, you may be tempted to walk away. But I want to end with some words of encouragement. I want to encourage you to stay, to fight, to be disruptive.

Be Disruptive

It's easy to become institutionalized when you work for a large organization with established ways of working, rigid hierarchies, and lots of bureaucracy. These can feel like constraints that stop you doing what needs to be done.

Such constraints only exist if you allow them to. You can choose to ignore the hierarchy, rules, and regulations if you truly feel they are holding the business back. You can choose to challenge and disrupt. There is nobody stopping you but yourself. As Jonathan Kahn wrote in his groundbreaking article on A List Apart[75]:

> *"It's about pointing out risks, shining a light on organizational denial, overcoming resistance, and facilitating constructive discussions about change."*

It is only our fear that holds us back. But what are we really afraid of? What is the worst that could happen? Yes, you could be fired, but is that so terrible? If your company is so resistant to embracing digital that they decided to fire you, is it really somewhere you want to work? There is no shortage of employers out there who are willing to change and are desperately looking for people like you.

Remember that leaders are not picked, they step up. If you want to change your organization's digital direc-

75. http://smashed.by/webgov

tion, you cannot wait until you are given permission. As Grace Hopper famously said:

> *"It is often easier to ask for forgiveness than to ask for permission."*

If you wish to change your organization's digital direction you need to be a maverick, willing to take risks and cause disruption. You need to make changes without waiting for permission, but be willing to ask for forgiveness if they turn out to be wrong. A great example of this is Microsoft's developer blog Channel 9. Microsoft had a poor relationship with its development community and its traditional marketing approach was not helping. Developers do not like being oversold to and generally are not receptive to slick marketing campaigns.

A group of nine developers within Microsoft decided they wanted to change things. They wanted to engage directly with the developer community and overcome the barriers that existed. Their answer was to create Channel 9, a resource featuring videos from people behind the scenes building products at Microsoft. They didn't go to the powers that be for permission, or ask how it needed to be integrated into the overall marketing strategy — they just launched it. The result was a shift in Microsoft's relationship with the developer community, something that seemed impossible before.

The question is: if you won't step up and change things, then who will? The truth is that only digital workers fully understand the problem and can see a so-

lution. Your company is not suddenly and miraculously going to become a digitally oriented organization.

Here is the truth, if you don't take action to change it, nobody else will. But if you do take action, there is a real opportunity to make your work more enjoyable and to have a real impact on your company. ❧

Beyond The Boring: The Hunt For The Web's Lost Soul

BY JOSHUA JOHNSON

Has web design lost its soul? And is responsive design to blame? These questions, posed by my friend and colleague Noah Stokes[76], are provocative to say the least. After all, the responsive web has made browsing on our ever increasing collection of Internet-connected screens not only possible, but enjoyable.

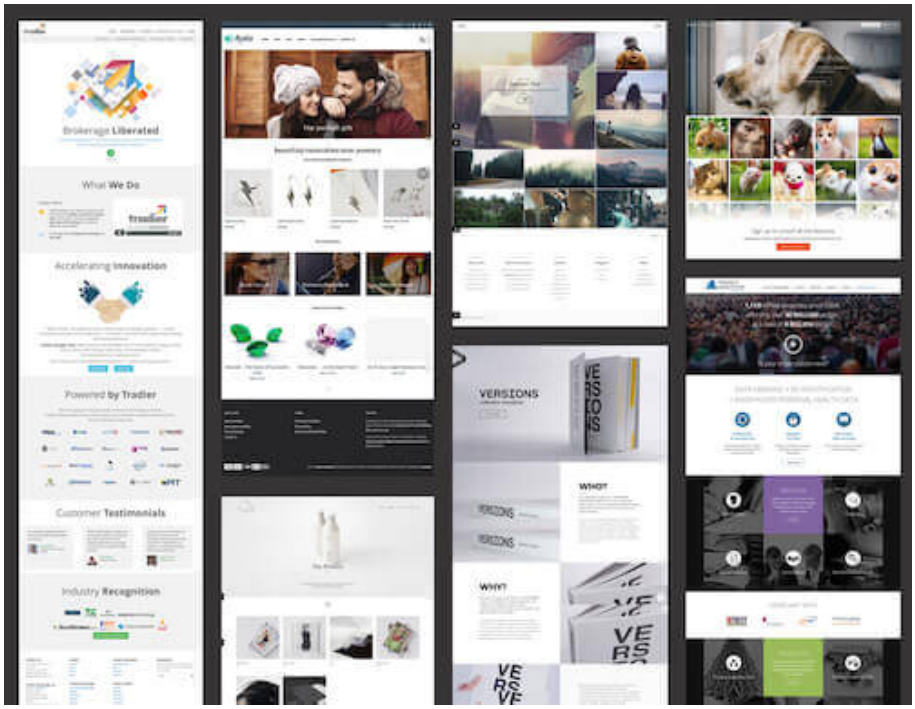
Our priority as designers must be to solve problems; perhaps more than anything else, this is what we do. Responsive web design is a fantastic solution to the problem of creating virtual experiences that adapt to different devices. There are other problems out there that we're called on to solve, though, not least of which is to make content of all kinds appear interesting and engaging. A page of plain text becomes a beautiful blog post, a mess of unconnected JPGs becomes a professional portfolio.

The question, then, becomes: can we succeed at solving both of these problems?

76. http://www.creativebloq.com/web-design/why-web-design-losing-its-soul-51514950

What We've Gained

There's no doubt about it, the web has become an aesthetically beautiful place. Simple, attractive sites that are built on solid grid layouts have become the standard. The wild west of the web has been tamed. Law and order in the form of frameworks and fluid grids have taken over and peace reigns throughout the land.



A stroll through a web design gallery like Awwwards[77] reveals hundreds of fantastic sites that fit the criteria of simple and attractive sites. (View large version[78])

77. http://www.awwwards.com/nominees
78. http://media.mediatemple.netdna-cdn.com/wp-content/uploads/2015/06/01-awwwards-opt.jpg

All of this is a good thing. The benefits of simple, attractive sites are both clear and plentiful:

1. Familiar and easy for users to navigate (when done correctly).
2. Prototyping is (relatively) fast and easy.
3. Heavy standardization of site-building techniques (assembly lines have replaced artisans).
4. Fast layout leads to big cost savings (which may or may not be passed down to the customer).
5. Strict grid layouts lend themselves to responsive design (the minimal layout trend is driven heavily by this need).

It can't be overstated that there is immense value in standardization and associated techniques. Countless individuals and small businesses benefit from having simple, attractive (and not at all unique) sites to share their brand with the world. However, that's just one side of the story.

What We've Lost

Taken in individual doses, the average professional website today looks great. Compare even a lowly designer's portfolio site today to the best design agency sites ten years ago, and you'll have to concede that we've gotten a *lot* better at this web design thing. How-

ever, as you look around, it's easy to come to the conclusion that everything is starting to look the same.

Have designers lost that pioneer spirit? Has creativity been sacrificed on the altar of convenience? Before answering these questions, let's take a look at what's causing the lack of variation in web design today.

Reasons Why Sites All Look The Same

What's the driving force behind the feeling of sameness that we get as we look around the web today? What's to blame? As it turns out, it's not as simple as pinning it all on one tool or method. RWD might contribute, but it's just one item in a long list. Here are a few of the likely suspects.

LAYOUT

Limited layout ideas are one of the most prominent and obvious reasons for a lack of variation on the web. Strip out colors, animations, parallax scrolling effects and the like, and you start to see that a few basic layouts rule the web. How many sites can you name that use only slight variations of the five layouts pictured on the next page?

This is what Noah meant when he lamented that all he saw were "boxes and grids everywhere." We seem to have reached a stagnation point where unique layouts are a lost art.

Five common web layouts. (View large version[79])

RESPONSIVE WEB DESIGN

Once upon a time, you could guarantee that everyone visiting your website would be doing so on a low-reso-

79. http://media.mediatemple.netdna-cdn.com/wp-content/uploads/2015/06/02-layout-opt.jpg

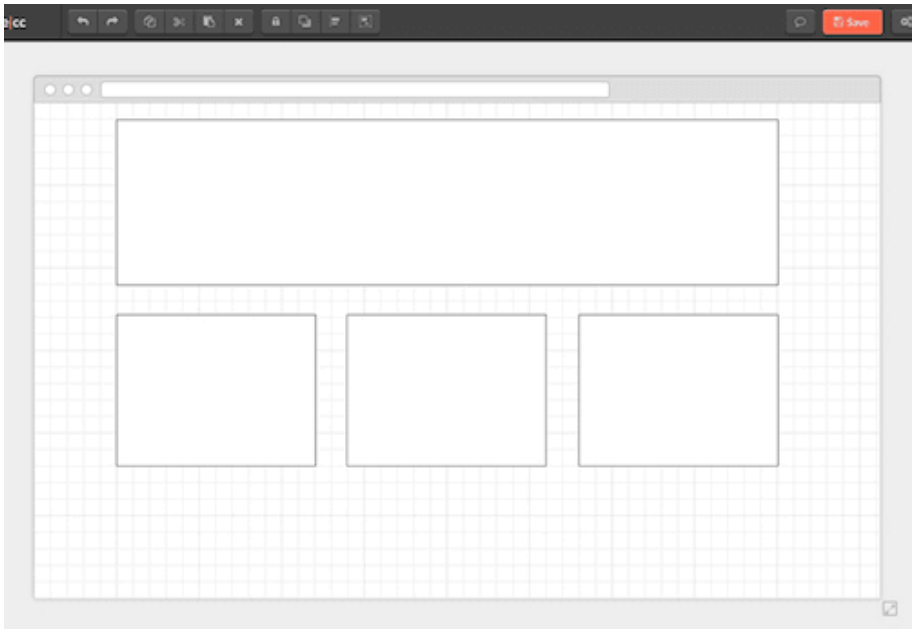
lution desktop computer screen. These days, websites that look great on almost any viewport at any resolution aren't a luxury, they're a necessity. Basic, fluid-width, collapsing grids make responsive web design a much simpler process than more organic layouts.

FRAMEWORKS

Web design frameworks have the potential to rapidly speed up both design and development workflows. For many, they're the safest, most straightforward route to a responsive, cross-browser website. As a bonus, they also take care of simple styling for all manner of common elements, from buttons to forms. The incredible popularity of tools like Bootstrap and Foundation leads to thousands of web designers using the exact same codebase, layouts, and even aesthetic style on every project they take on.

PROTOTYPING TOOLS AND PROCESSES

Even our blank-canvas design process has intentionally reduced creativity from the design process. Most prototyping tools encourage and perhaps even force you to use standardized, boxy elements that conform to strict grid layouts.



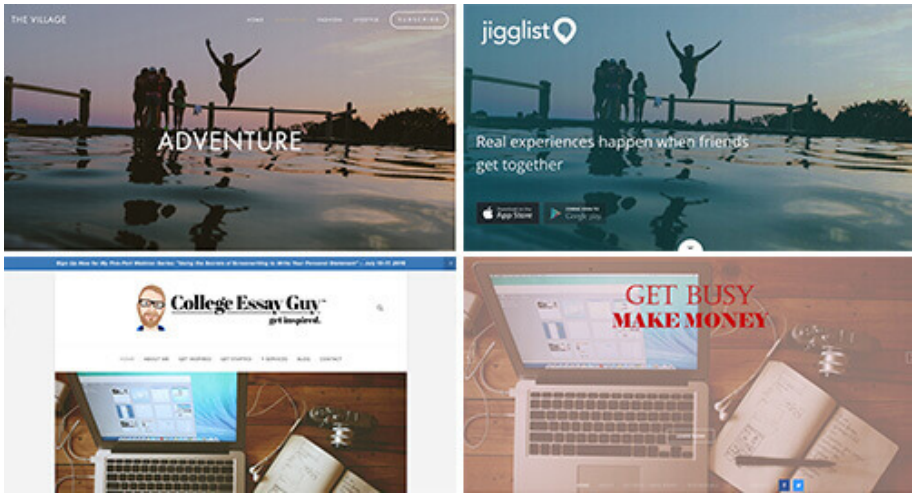
Many prototyping tools encourage common grid layout structures.

HIGH-QUALITY FREE PHOTOS AND GRAPHICS

Never before have beautiful, free design assets been so readily available. Again, convenience and affordability reign supreme, so we see designers flocking to free photo sites, pulling from the same small (but growing) pool of resources.

Two free stock photographs.

In the last two years, how many sites have you seen using one (or both) of the two photos above? If you browse the web regularly, I'd wager the number is in the dozens.



The Village Style[80] *and Jigglist*[81]*; College Essay Guy*[82] *and Get Busy Make Money*[83] *(View large version*[84]*)*

This goes way beyond photos, extending to icons, fonts, patterns, and so on. The upside here is that designers with zero budget can still make great looking sites; the downside is that every other designer is doing the same thing with the same resources.

---

80. http://thevillagestyle.com/adventure1
81. http://jigglist.com
82. http://www.collegeessayguy.com
83. http://www.getbusymakemoney.com
84. http://media.mediatemple.netdna-cdn.com/wp-content/uploads/2015/06/05-stockphoto-examples-opt.jpg

DESIGN TRENDS

To some designers, "trend" is a dirty word. It shouldn't be though. Pick a decade in the 20th century and examine its design trends. What you find is fascinating. It gives that time period its own distinct style and personality, and is often a reflection of the entire culture. Even if you can't see it, it's happening right now in your work. Everything you see and experience is affecting what you do, and the web amplifies this like never before. The result is a lot of designers gaining inspiration from the same sources and pursuing the same trends.

YOU AND ME

Every tool and resource listed above is incredibly valuable. These things make our jobs easier, open up web design and development to more people, and save clients money. I didn't choose them arbitrarily, I use them. If we're looking for someone to blame for a lack of variation in web design, I'll raise my hand. It's me. And while I don't think every project merits a unique design, I'd like to tip my hat to the designers out there who are trying to do something more interesting.

It's also important to note that you can use any or all of the above and still create a unique design. It's all in how you wield the tools that are available to you.

How To Challenge The Status Quo

At this point, you've already decided whether or not you care if your site designs are unique. Maybe that's not

your thing — that's completely acceptable. If you find yourself wanting to break out of your typical workflow, though, here are a few ideas to get you going and some sites that serve as great examples.

GET WEIRD WITH LAYOUT

Close your prototyping app, take out a pen and some paper, and think about how you can make an interesting, usable layout that's not something you've ever done before.

Phases Magazine

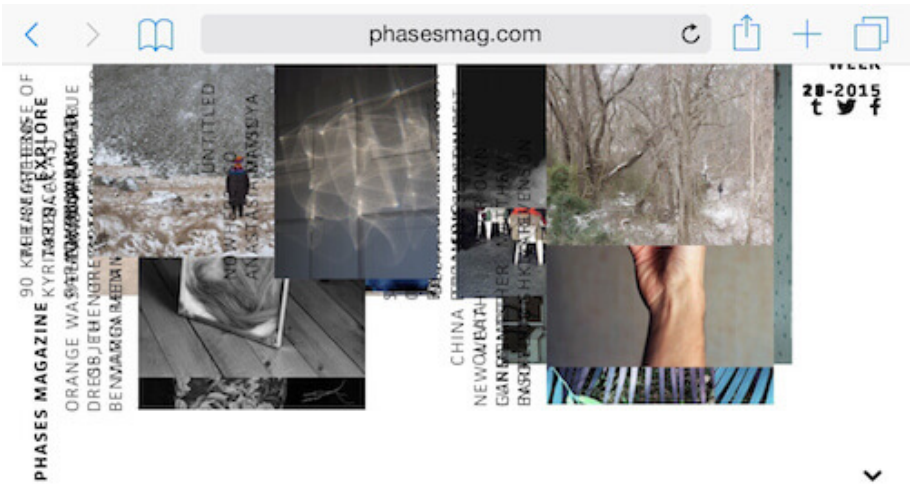
Phases Magazine[85] is a good example of a site doing something unexpected with layout. It's boxy — there is a grid at play here — and yet somehow they made it feel totally outside the typical web design experience (the screenshot doesn't really give you a good feel for it; be sure to visit the site). If you see this and think, "Whoa, that's weird," good! That's what they're going for. Some will love it, some will hate it, but I really like that they're trying something different.

Unfortunately, the site's overall experience is quite poor. There's some unnecessary scrolljacking, and the seemingly responsive layout breaks massively in certain viewports.

85. http://www.phasesmag.com

Phases Magazine uses an unconventional grid. (View large version[86])

86. http://media.mediatemple.netdna-cdn.com/wp-content/uploads/2015/06/06-phases-magazine-opt.jpg



Phases Magazine's layout breaks on smaller screens. (View large version[87])

Curious Space

A better implementation of a similar idea is Curious Space[88]. Here the grid still has a more organic feel, but scrolling functionality is normal and the breakpoints are perfectly functional.

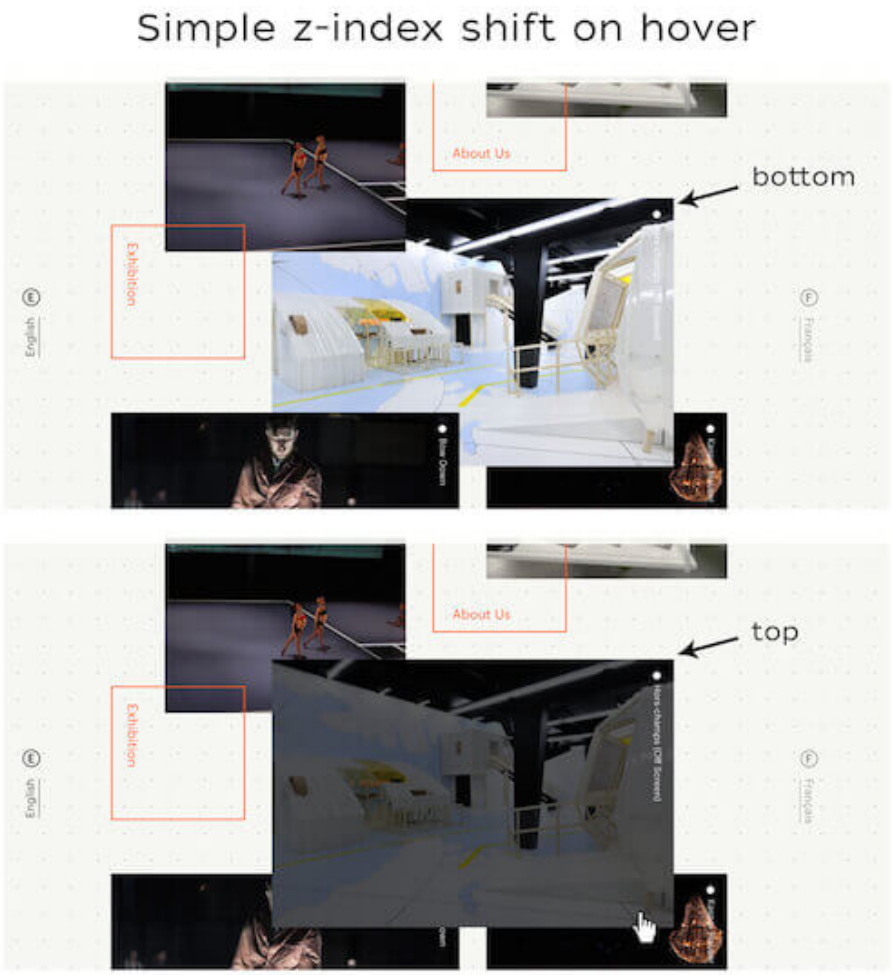
Curious Space adapts well to different devices. (View large version[89])

87. http://media.mediatemple.netdna-cdn.com/wp-content/uploads/2015/06/07-phases-magazine-opt.jpg
88. http://www.curiousspace.com
89. http://media.mediatemple.netdna-cdn.com/wp-content/uploads/2015/06/09-curious-space-opt.jpg

The layout of Curious Space. (View large version[90])

There are a lot of nice little design touches that you notice as you interact with the site. For instance, the image stack order changes on hover:

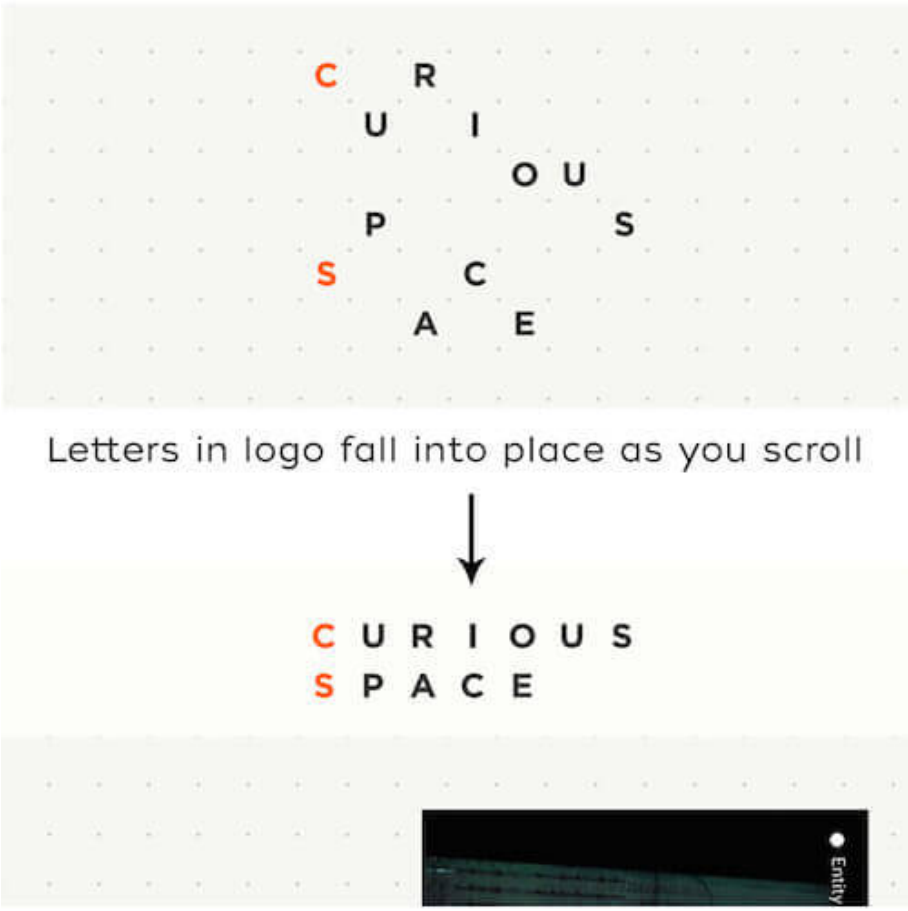
90. http://media.mediatemple.netdna-cdn.com/wp-content/uploads/2015/06/08-curious-space-opt.jpg



Hovering with a mouse brings content to the foreground.
(View large version[91])

Also, they have a bit of fun with scrolling, but it's not disrupting in the least, and it fits with the haphazard visual theme of the site. The logo starts off as a jumble of letters, but as you scroll, they fall into place and form "Curious Space" in the navigation bar.

91. http://media.mediatemple.netdna-cdn.com/wp-content/uploads/2015/06/10-curious-space-opt.jpg



The letters of the logo fall into place as you scroll. (View large version[92])

Le Temps D'un Trajet

Another interesting example is Le Temps d'un trajet[93]. Once again, we see a non-standard grid, but instead of being sporadic, the layout is more intentional and clustered.

92. http://media.mediatemple.netdna-cdn.com/wp-content/uploads/2015/06/11-curious-space-opt.jpg
93. http://letempsduntrajet.fr

The homepage layout of Le Temps d'un trajet. (View large version[94])

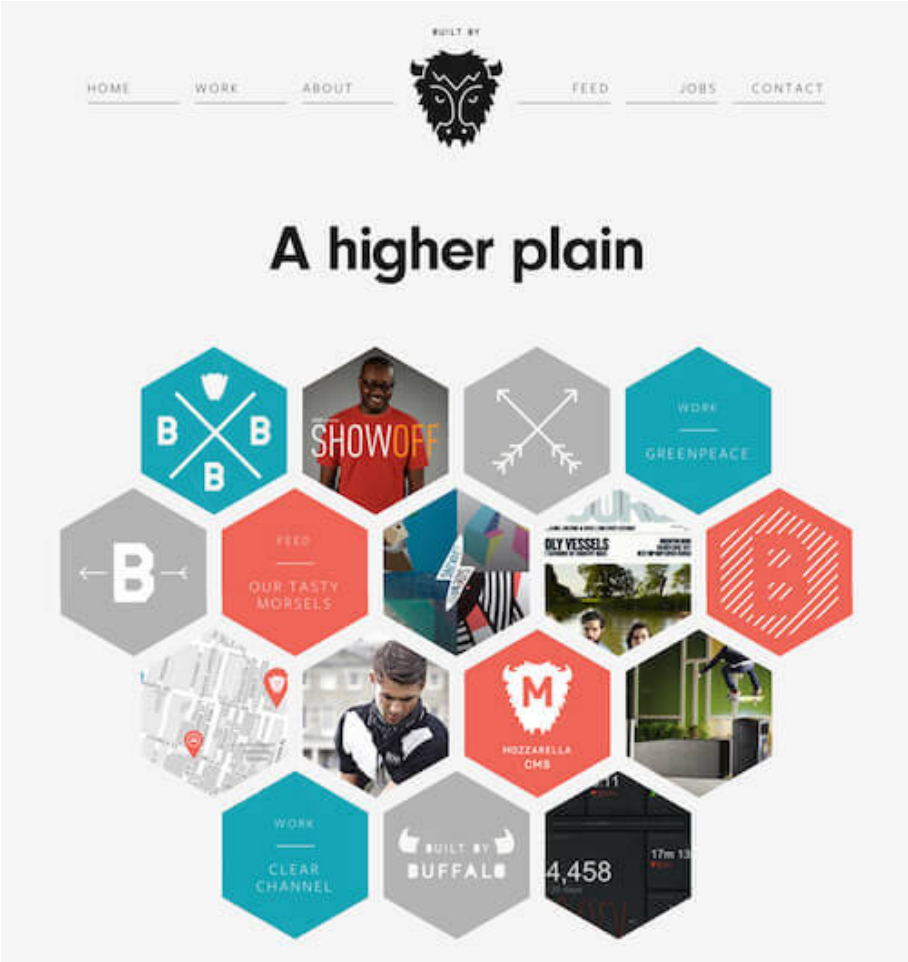
Each block in the arrangement is a static image that turns into a short video on hover. The coolest part, though, is how the grid shifts fluidly to accommodate focusing on different nodes.

DON'T BE A SQUARE

One easy way to give your site a different atmosphere is to think outside the box. As cringeworthy as that sounds, when you open yourself up to different geometry, things can get really interesting.

94. http://media.mediatemple.netdna-cdn.com/wp-content/uploads/2015/06/12-le-temps-dun-trajet-opt.jpg

Built By Buffalo

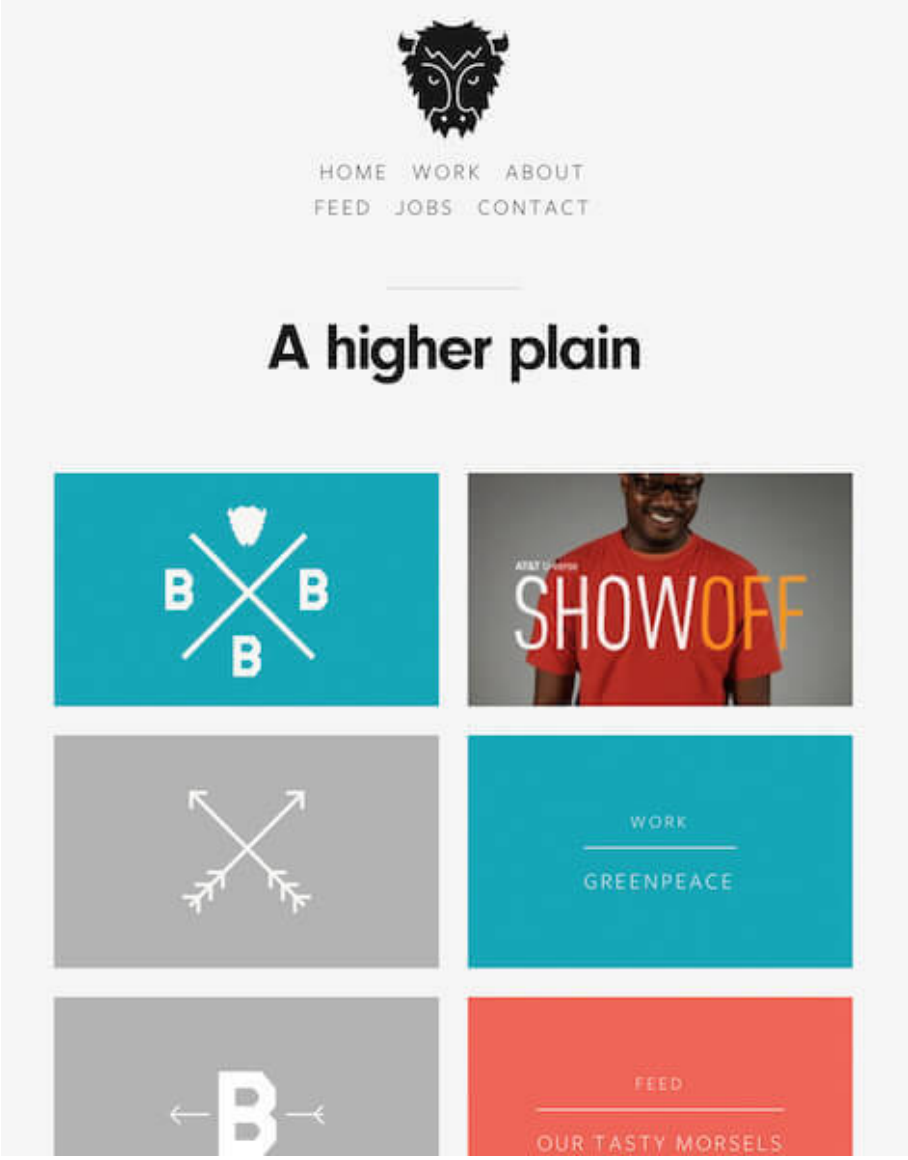


Built By Buffalo[95] *uses hexagons for layout. (View large version*[96]*)*

Check out the hexagon hive that Built By Buffalo has going on. This gallery design doesn't translate nicely to mobile, so they simply switch to rectangles at one of their breakpoints. This is a great example of doing something unique where appropriate, but realizing

95. http://builtbybuffalo.com
96. http://media.mediatemple.netdna-cdn.com/wp-content/uploads/2015/06/12b-build-by-buffalo-opt.jpg

where the boundary should be drawn to give your users the best possible experience.



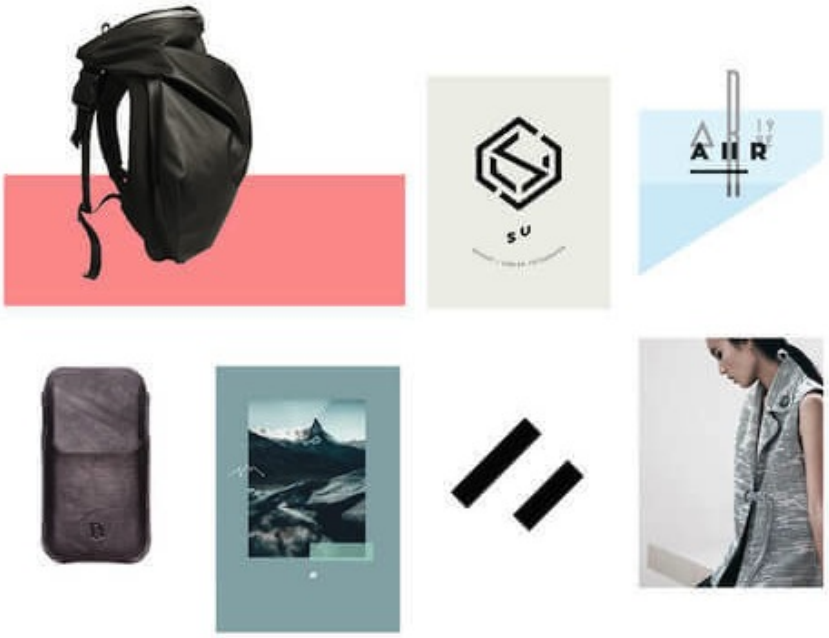
The layout switches to rectangles on narrower viewports.
(View large version[97])

97. http://media.mediatemple.netdna-cdn.com/wp-content/uploads/2015/06/13-build-by-bufallo-opt.jpg

Anakin Design Studio

One of my favorite sites in this vein that I've seen recently is Anakin Design Studio[98]. Not only is the layout beautiful and unpredictable, the shapes at play here are all far outside what you'd expect scrolling through today's websites.

As you can see, the huge, masked typography makes a bold impression. Beyond that though, if you move down the page you see a display of recent work. Most designers would put a simple rectangular thumbnail grid here and call it a day, but Anakin has played with the shapes to make it a lot more interesting. They're still rectangular images, but they've used white backgrounds to create the illusion of varying shapes.



Anakin Design Studio's portfolio. (View large version[99])

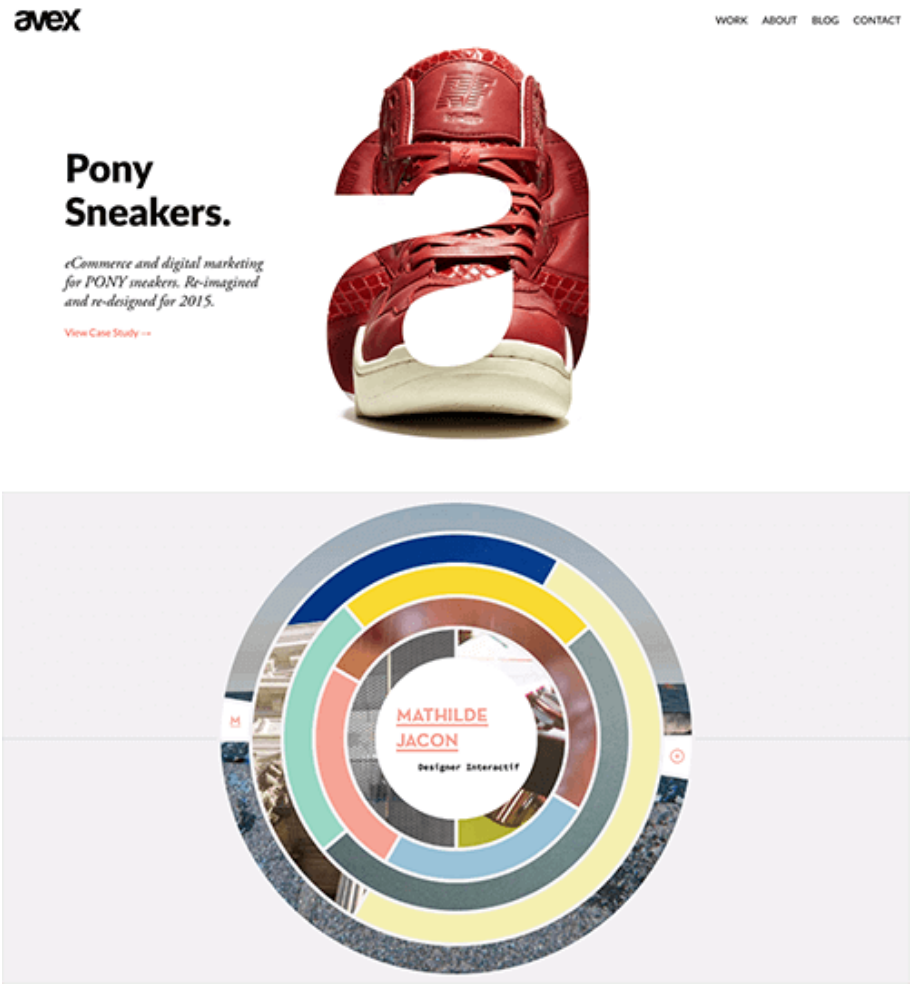
98. http://www.anakin.co/en

Anakin Design Studio's homepage. (View large version[100])

99. http://media.mediatemple.netdna-cdn.com/wp-content/uploads/2015/06/15-anakin-opt.jpg
100. http://media.mediatemple.netdna-cdn.com/wp-content/uploads/2015/06/14-anakin-opt.jpg

Avex Designs; Mathilde Jacon

Here are some other sites doing unexpected experiments with interesting shapes.



Avex Designs[101] *and Mathilde Jacon*[102] *use unconventional shapes to create unique experiences. (View large version*[103]*)*

101. http://avexdesigns.com
102. http://www.mathildejacon.com
103. http://media.mediatemple.netdna-cdn.com/wp-content/uploads/2015/06/16-interesting-shapes-opt.png

GO ORGANIC

Fixate; For Better Coffee

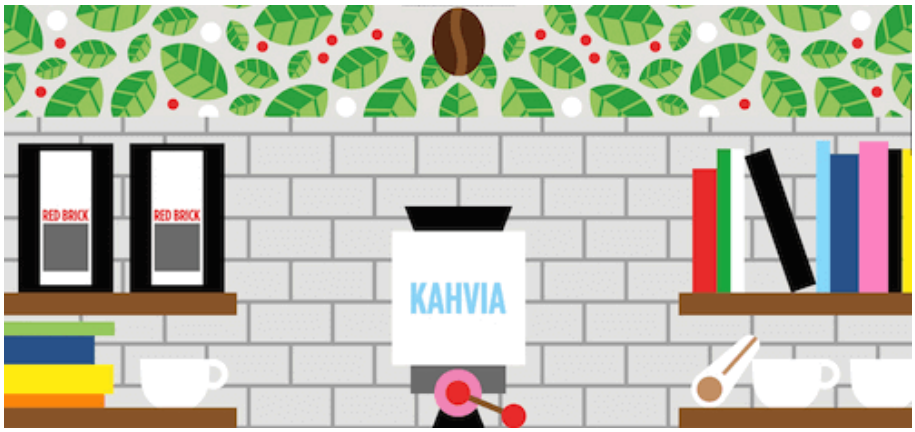
Sites like Fixate[104] and For Better Coffee[105] combine illustration with organic, crazy-busy layouts that make for powerful and memorable experiences.

The layouts below only seem complex because of the artwork; in reality, they can be pulled off fairly easily. Custom illustration work is a fantastic way to communicate a unique brand personality in a world lost in minimal thumbnail grids that all look the same.

104. http://fixate.it
105. http://forbetter.coffee

Fixate and For Better Coffee use highly illustrated layouts to help them stand out. (View large version[106])

For Better Coffee uses entertaining animations to track the coffee bean through the coffee creation process as you scroll. The experience is smooth, and it doesn't jump the page to predefined points, so scrolling becomes a story-telling feature.



A coffee bean falls toward a grinder as the page is scrolled. (View large version[107])

106. http://media.mediatemple.netdna-cdn.com/wp-content/uploads/2015/06/17-organic-sites-opt.png

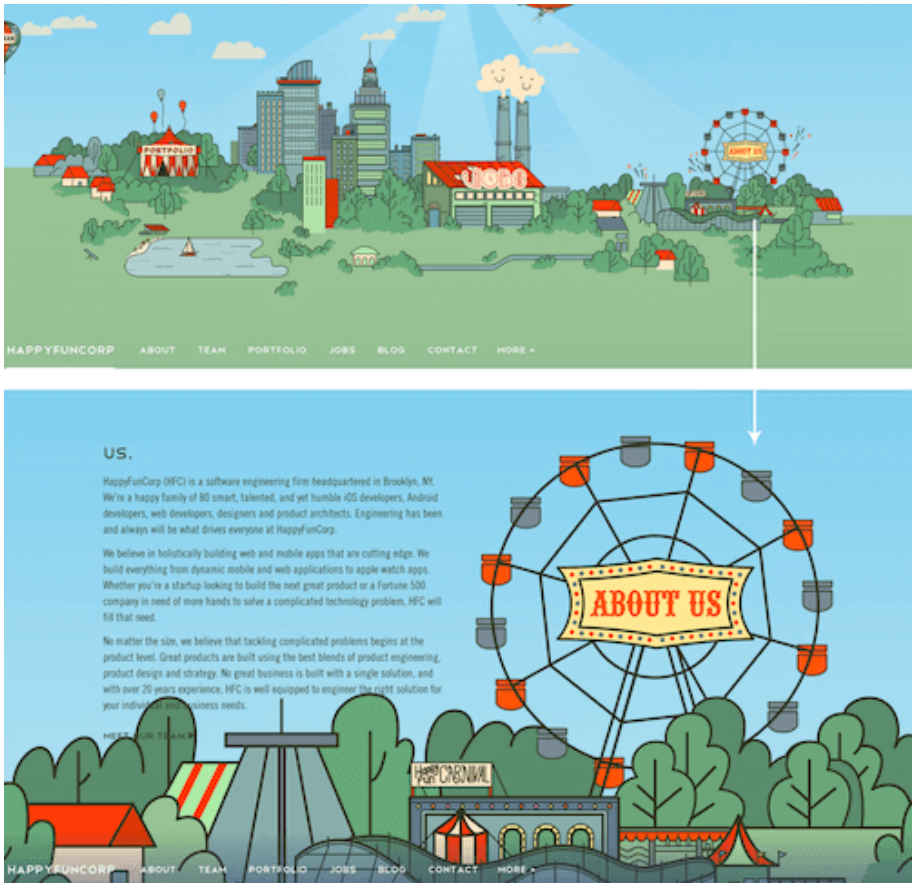
Further down the page, scrolling pours fresh coffee into a cup. (View large version[108])

HappyFunCorp

While we're on the topic of beautifully illustrated sites, you should check out HappyFunCorp[109].

At first, it seems like a fairly normal site with some fun little animations. The cool part happens when you start to navigate. The scene on the homepage serves as the basis for the rest of the site, so when you follow a link, instead of loading a brand-new page it zooms in to a detail of the whole scene.

107. http://media.mediatemple.netdna-cdn.com/wp-content/uploads/2015/06/18-for-better-coffee-opt.png
108. http://media.mediatemple.netdna-cdn.com/wp-content/uploads/2015/06/19-for-better-coffee-opt.png
109. http://happyfuncorp.com/#home



Following a link zooms in to a detail. (View large version[110])

It's quirky, but I love the original thinking that went into it. Also, because the navigation is still presented in a standard way means that there's no learning curve for users. The whole experience is delightfully unexpected, but in a way that doesn't contradict how you normally interact with a site.

110. http://media.mediatemple.netdna-cdn.com/wp-content/uploads/2015/06/20-happy-fun-corp-opt.png

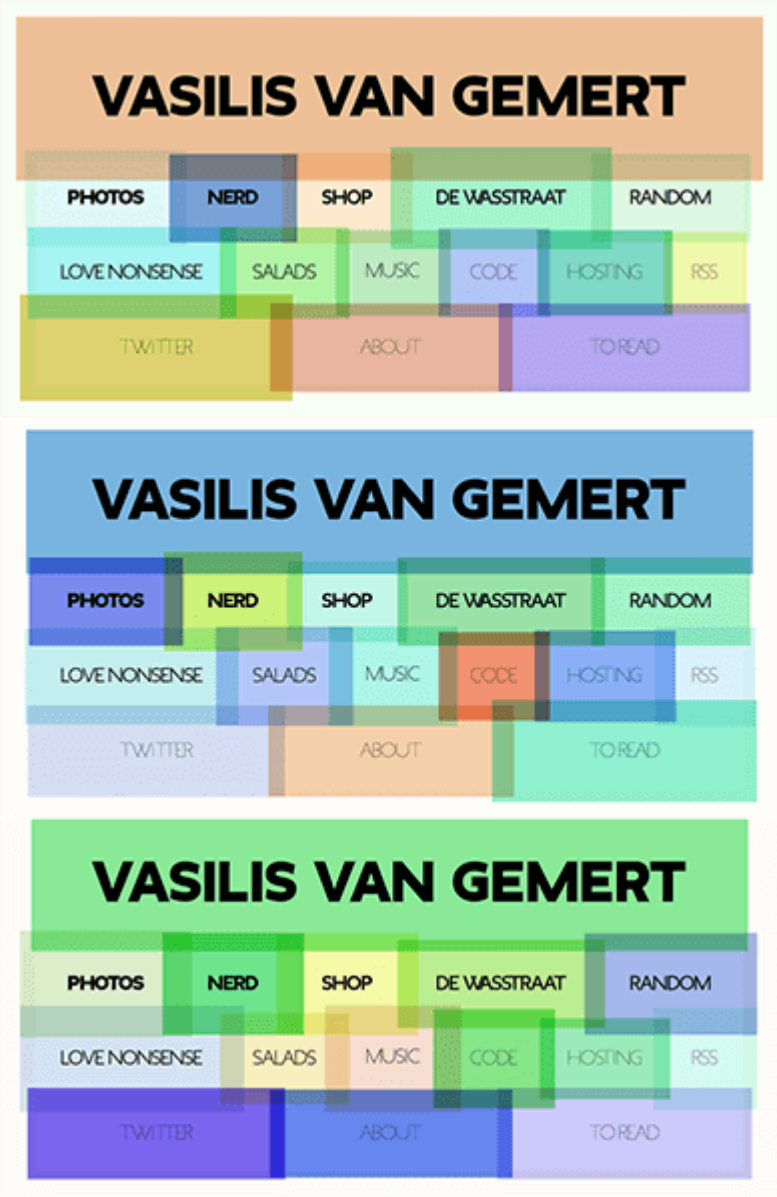
VARY THE EXPERIENCE

One basic expectation that users have is that everyone who lands on the same site will receive the same experience; a fun way to do something different is to toss that out the window. Vasilis van Gemert's site[111] not only uses a unique, overlapping box layout, it also changes its entire color scheme for every visit.

This creative feature extends beyond the homepage to all of the major subpages.

Subpages also change their color schemes. (View large version[112])

111. https://vasilis.nl
112. http://media.mediatemple.netdna-cdn.com/wp-content/uploads/2015/06/22-vasilis-van-gemert-opt.png



Three different color schemes from Vasilis van Gemert's site.
(View large version[113])

113. http://media.mediatemple.netdna-cdn.com/wp-content/uploads/2015/06/21-vasilis-van-gemert-opt.png

CREATE A UNIQUE VISUAL THEME

Another way to make your site design unique is to decide on an entertaining or interesting theme that you can use as the basis of all your design decisions. This provides a nice framework for everything you do and encourages you to explore beyond traditional UI.

Part of the retro-futuristic design of the dConstruct 2015 site.
(View large version[114])

114. http://media.mediatemple.netdna-cdn.com/wp-content/uploads/2015/06/23-dconstruct-opt.jpg

While not appropriate for all sites (a government website, for instance), for things like event announcements or small company pages it can be refreshing. The new site for dConstruct 2015[115] is a great example of this idea.

As you can see, they went for a retro-futuristic vibe, heavily reminiscent of The Jetsons[116]. The result is a site that's flat out fun to scroll through as you discover the how they present each new section.

A detail from the dConstruct design. (View large version[117])

115. http://2015.dconstruct.org
116. https://en.wikipedia.org/wiki/The_Jetsons
117. http://media.mediatemple.netdna-cdn.com/wp-content/uploads/2015/06/24-dconstruct-opt.jpg

There are a couple of really great things to note about this site. First of all, the headline treatments are wonderfully retro, using a combination of Lamplighter Script and Andes.

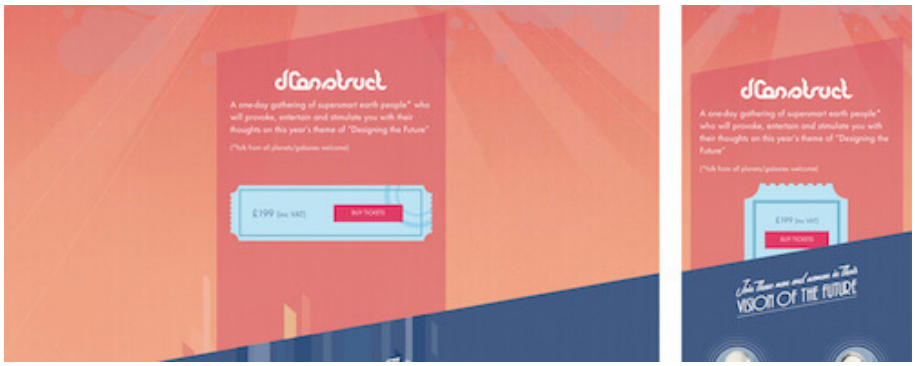
The best part, though, is that these are live web fonts with the diagonal direction implemented via a simple CSS skew. The repetition of diagonal lines throughout the site helps the design feel both consistent and creative.

With creative use of styled web fonts, all the text is selectable. (View large version[118])

This site is a great example of how responsive design doesn't have to be boring. The layout doesn't feel boxy or typical, and yet it manages to reflow nicely to any viewport size. In fact, I really love how creative they were with transforming elements for smaller screens.

118. http://media.mediatemple.netdna-cdn.com/wp-content/uploads/2015/06/25-dconstruct-opt.jpg

For instance, as your viewport shrinks, the ticket graphic shown below jumps into an animated transform that shifts from horizontal to vertical orientation. It's a tiny detail, but it's clever and shows how much thought was put into every aspect of the layout.



For narrow viewports, the ticket flips from horizontal to vertical. (View large version[119])

Hats off to Clearleft[120] for the great work on this one.

USE STOCK AS AN INPUT, NOT AN END RESULT

I work at Creative Market[121], so I'm absolutely in favor of using great stock graphics, fonts, photos, and more. Quality stock resources can be immensely helpful for every designer, but how you use them is an important consideration. Whether you're using some interesting vector artwork, an icon set, or a full-blown website theme, consider putting in some extra effort so that

119. http://media.mediatemple.netdna-cdn.com/wp-content/uploads/2015/06/26-dconstruct-opt.jpg
120. http://clearleft.com
121. https://creativemarket.com

your implementation doesn't look exactly like that of everyone else. The vast majority of people downloading that asset won't bother with much or any customization, so a little bit goes a long way.

The Trouble With Unique Sites

Finding sites that are truly surprising and unique is a tall order. I put in hours of site searching to prepare for this article and still feel like I have very little to show for it. One overwhelming trend I've found is that it often feels like the only designers really pursuing unique web design are producing sites with wonky user experiences.

Experimentation is great, but sites that confuse users with weird, unexpected and unpleasant UX often fall short of their goals. Many of these sites take a step backwards by presenting us with Flash-like experiences: long loading times, overly ornate animations, jumpy scrolling, and complex user flows. There is a middle ground. You can create beautiful, unique looking websites without trying to reinvent the interaction wheel.

Web Design Lives

Standardization and predictable design will always have their place on the web. In fact, they may be the best possible solution for presenting most types of content to most screens. That said, we should let our cre-

ative instincts fight it out with our analytical instincts from time to time.

Let's do our part to make sure the web remains what it has always been: a place for technology, art and design to overlap in new and interesting ways. Be a pioneer, try something you've never seen anyone do — and yes, make lots of mistakes along the way. It's great to create something weird every now and then, even if other people hate it. That's how this crazy thing called the web was built, and that's how we'll keep pushing it forward.

Ultimately, despite the fact that plenty of sites look pretty similar, I don't believe that web design has reached a point of stagnation. There are countless extremely talented designers creating unbelievable sites, constantly raising the bar for their peers. ❧

About The Authors

Aarron Walter

Aarron Walter[122] is the General Manager of New Products at MailChimp[123] and the author of *Designing for Emotion*[124] from A Book Apart[125]. Before joining MailChimp, Aarron taught design at colleges in the US and Europe for nearly a decade. He's a frequent speaker at conferences around the world, and his design guidance has helped the White House, the US Department of State, and dozens of startups and venture capitalists. He tweets about design under the moniker @aarron on Twitter[126].

Alessandro Cattaneo

Alessandro Cattaneo is an experienced digital marketing manager, with a thing for web design, digital advertising and online communication.

Brad Frost

Brad Frost[127] is a web designer, speaker, writer, and consultant located in beautiful Pittsburgh, PA. He's pas-

122. http://aarronwalter.com
123. http://mailchimp.com
124. http://www.abookapart.com/products/designing-for-emotion
125. http://abookapart.com
126. http://twitter.com/aarron
127. http://bradfrost.com

sionate about creating Web experiences that look and function beautifully on a never-ending stream of connected devices, and is constantly tweeting[128], writing[129] and speaking[130] about it. He's the author of Atomic Design[131], and has also helped create several tools and resources for web designers, including This Is Responsive[132], Pattern Lab[133], Styleguides.io[134], WTF Mobile Web[135], and Mobile Web Best Practices[136].

Joshua Johnson

Josh Johnson is the Product Manager for Creative Market[137]. He's also a writer, designer, and photographer, and currently lives in Phoenix. Twitter: @secondfret[138].

Mat Marquis

Mat "Wilto" Marquis makes websites[139] for a living at Bocoup[140] and crashes his motorcycle for free on the streets of North Cambridge. Mat is Chair of the Respon-

128. https://twitter.com/brad_frost
129. http://bradfrostweb.com/blog
130. http://bradfrostweb.com/speaking
131. http://atomicdesign.bradfrost.com
132. http://bradfrost.github.com/this-is-responsive
133. http://pattern-lab.info
134. http://styleguides.io
135. http://wtfmobileweb.com
136. http://mobilewebbestpractices.com
137. https://creativemarket.com
138. http://www.twitter.com/secondfret
139. https://the-pastry-box-project.net/mat-marquis/2013-july-27
140. http://bocoup.com/

sive Issues Community Group[141], technical editor at A List Apart[142], and a former member of the jQuery team[143]. Mat has finished Mega Man 2 — on difficult — without losing a single life. He's probably flipping out about something on Twitter[144] as we speak, as @wilto.

Paul Boag

Paul Boag is the author of *Digital Adaptation*. He is a leader in digital and user experience strategy with over 20 years experience. Through consultancy, speaking, writing, training and mentoring he passionately promotes digital best practice. Twitter: @boagworld[145].

Peter-Paul Koch

Peter-Paul Koch[146] (PPK) has been around for quite some time. Known for his browser compatibility tables on Quirksmode.org[147], he is a mobile platform strategist, browser researcher, consultant, and trainer in Amsterdam, the Netherlands. He specialises in the mobile web, and especially mobile browser research, advising mobile and desktop browser vendors on their implementation of web standards. Twitter: @ppk[148].

---

141. http://ricg.io/
142. http://alistapart.com
143. http://jquerymobile.com
144. http://twitter.com/wilto
145. http://twitter.com/boagworld
146. http://www.quirksmode.org/about
147. http://quirksmode.org
148. https://twitter.com/ppk

Susan Weinschenk

Susan Weinschenk has a Ph.D. in Psychology and over 30 years of experience as a behavioral scientist. She is a consultant to Fortune 1000 companies, start-ups, and government and non-profits. Her clients call her "The Brain Lady" because she applies research on brain science to predict, understand, and explain what motivates people and how they behave. Dr. Weinschenk is the author of several books, including *100 Things Every Designer Needs To Know About People* and *How To Get People To Do Stuff*.

Susan is a keynote speaker around the world, and these days spends a lot of time in her video studio recording her online training classes and she's also writing her next book. Her clients include Medtronic. Walmart, Disney, Amazon, The Mayo Clinic, and the European Union Commission. Dr. Weinschenk is also an Adjunct Professor at the University of Wisconsin, and writes two popular blogs — one at her own website[149] and "Brain Wise: Work better, work smarter" for Psychology Today.

Vitaly Friedman

Vitaly loves beautiful content and complex challenges, and does not give up easily. He co-founded Smashing Magazine back in September 2006 and since then spends pretty much every day trying to make it better,

149. http://www.blog.theteamw.com/

faster and useful. He runs responsive design training and workshops and loves solving complex UX, performance and front-end problems in large and small companies.

About Smashing Magazine

Smashing Magazine[150] is an online magazine dedicated to Web designers and developers worldwide. Its rigorous quality control and thorough editorial work has gathered a devoted community exceeding half a million subscribers, followers and fans. Each and every published article is carefully prepared, edited, reviewed and curated according to the high quality standards set in Smashing Magazine's own publishing policy[151].

Smashing Magazine publishes articles on a daily basis with topics ranging from business, visual design, typography, front-end as well as back-end development, all the way to usability and user experience design. The magazine is — and always has been — a professional and independent online publication neither controlled nor influenced by any third parties, delivering content in the best interest of its readers. These guidelines are continually revised and updated to assure that the quality of the published content is never compromised. Since its emergence back in 2006 Smashing Magazine has proven to be a trustworthy online source.

150. http://www.smashingmagazine.com
151. http://www.smashingmagazine.com/publishing-policy