

Web Typography

*A handbook for designing beautiful and
effective responsive typography*



Richard Rutter

Web Typography

*A handbook for designing beautiful and
effective responsive typography*

by Richard Rutter

Ampersand Type
Brighton, UK
2017

Web Typography
by Richard Rutter

Published in 2017 by Ampersand Type, Brighton, UK
<http://book.webtypography.net>
Please send errors to errata@webtypography.net

Editor: Owen Gregory
Design: Richard Rutter & David Rutter
Compositor: David Rutter

Headlines set in Ingeborg by Michael Hochleitner
Body text set in Premiéra by Thomas Gabriel
Captions set in Skolar Sans by David Březina

Printed by Generation Press, Poynings, UK
on Munken Polar Smooth.

Copyright © 2017 Richard Rutter. All rights reserved.
ISBN: 978-0-9956642-0-3
A catalogue record of this book is available from the British Library.

Foreword

by Mark Boulton

What can I possibly say about typography that hasn't been said before? I could tell you that there is probably no design element more important. Or that mastery of typographic design should be first and foremost in a designer's endeavours. I could tell you that controlling how language is displayed controls how it is read and perceived. Typography is really important.

But all that has been said before. The difference, of course, is that was mostly said about typography in printed form, not of web typography. No, web typography requires a slightly different outlook. Sure, typography on the web requires the same attention to detail, and we employ a similar arsenal of guidelines and principles to our print-based cousins. But we do it on a different foundation: the web. An ever-changing, fluid, dynamic system where users consume words on a plethora of devices and web browsers. We must not just be comfortable with this chaos and lack of control – we must embrace it. As you will read in this book, typography on the web is familiar but subtly different. Our typography can never be as exacting and precise as print, but using the teachings in this book, we can create just as delightful reading experiences on the web.

In 2007, Richard and I stood on a stage at South by Southwest Interactive in Austin. A few months earlier, we had proposed a talk provocatively titled 'Web Typography Sucks' with the hope of maybe being accepted to speak to a few dozen typography nerds in one of the small rooms in the enormous convention centre. To our great surprise, not only were we to present in the largest hall at sxsw that year, but it was also packed. Typography, it seemed, was as cool as Web 2.0. Now, a decade later, we have almost as much typographic freedom and control on the web as we do in other media. So much has changed. But does web typography still suck? I'll leave you to answer that, once you've read this book.

How to use this book

This book was written as a practical guide and companion reference for designers, developers and anyone else involved in the process of creating a website.

Typography touches, and can be affected by, all people involved in the chain of events that leads to a website being conceived, designed, written, built and published. This means that if you're an information architect, user experience designer, author, editor, project manager or client, this book is for you too.

To get the most from this book you will benefit from some familiarity with the basics of `HTML` and `CSS`, but you will need no advanced knowledge of type and typography, so if you're new to the field you'll find this book accessible. A wise person once said that the best experts are keen on sharpening old saws and learning new tricks. This book provides enough in-depth detail that a seasonal professional will also find it useful.

You can read this book from cover to cover as a complete guide to best practices in designing and implementing the typography of a website. You can also use this book as a reference to be dipped into for technical and aesthetic guidance, as and when the need arises.

The book is divided into three parts: '[Setting Type to be Read](#)', '[Typographic Detail](#)' and '[Choosing and Using Fonts](#)'. Each section is written as a series of guidelines. Take these not as *rules* but as guidance for making good decisions. Some lucky people have immaculate technique in their veins. The rest of us need instructional books like this to provide best practices for us to follow as we hone our craft and develop our own instincts.

The section on choosing typefaces is deliberately the final part of the book. Choosing a typeface is not typography. Before the advent of web fonts, typographic discipline online was scant. The source of this neglect was largely due to a resentment of the paucity of fonts available.

This frustration was both unwarranted and misdirected. A typographer's focus should be to make the very best of what is available. There will be times when, for whatever reason, the typefaces you have at your disposal are not what you would ideally wish to use. Do not be disheartened by this – treat it as a call to arms, a challenge to rise above what the typeface offers. This book should show you that even very ordinary faces can be crafted into an engaging, absorbing typographic experience for your reader.

During your process of designing a website you can follow the guidelines in this book in order – it would certainly be a suitable way to learn the craft. This would leave the act of choosing a typeface until after you have designed all the essentials of a website. It is certainly possible to do that, but as with all design you should accept that a process of iteration will be involved. The typeface you eventually choose will subtly (and occasionally greatly) influence many other parts of the design and the precise nature of the typography. That said, it is worth repeating: *choosing a typeface is not typography*; but if you have the choice, it's an important aspect.

A few caveats

This book only explores the requirements of European written languages, with a particular emphasis on the Latin alphabet and quite possibly an unintentional bias towards English. Complex scripts, such as Arabic or Devanagari, as well as logographic writing such as Hanzi or Kanji, are not covered.

As you might rightly guess for a book on web typography, this guide covers a lot of HTML and CSS. One of the great aspects of CSS, and working on the web as whole, is that technologies are continually evolving. To ensure that this remains relevant and accurate for as long as possible into the future, all the CSS techniques mentioned in this book can be found in W3C specifications and have at least some browser support (unless otherwise stated). The progressive nature of the web means that there will be more support in browsers now than when the text was written. Consequently there is only occasional mention of which browsers do or don't support any particular CSS, and very few workarounds are detailed. The open web is a far better place than a book to find up-to-date information on browser support¹ and shims or hacks.

¹ For up to date information on browser support try [Can I Use](http://caniuse.com) (<http://wbtyp.net/61>).

Typefaces versus fonts

In general parlance the terms ‘typeface’ and ‘font’ can be used interchangeably and anyone who admonishes you otherwise should be deemed a pedant. In almost all situations it is perfectly fine and comprehensible to say, ‘Gill Sans is a font’.

That said, throughout this book the two terms are used to mean slightly different things. *Typeface* is always used to mean a font family – a collection of fonts based on the same design (as in Gill Sans). The term *font* is mostly used to imply an individual style such as Gill Sans Bold Condensed. There may well be occasions when ‘font’ has been used to avoid repetition of ‘typeface’ – hopefully your reading experience will be improved by that small indulgence. *Web font* is used to mean a font file downloaded for the rendering of text in web pages.

Contents

We are all Typographers

Why web typography really does matter	3
Embracing the medium	6
Preparing the ground	8

1 Setting Type to be Read

How we read	11
The amazing em (and friends)	14
Designing paragraphs: line length	21
Designing paragraphs: text size	26
Designing paragraphs: line spacing	34
Alignment, justification and hyphenation	43
Responsive paragraphs	57

2 Typographic Detail

Symbols, signs and accents	67
Ligatures and abbreviations	78
Hierarchy and scale	87
Meaning and semantics	96
Numerals and tables	119
Tracking and kerning	145
Headlines and impact	152
Applying vertical rhythm	171
Arrangement and composition	177

3 Choosing and Using Fonts

How fonts render on screens	195
Practical and pragmatic considerations	203
Knowing and browsing type	209
Selecting typefaces for body text	237
Choosing typefaces for display	246
Choosing typefaces for functional text	256
Combining typefaces	262
Using web fonts	269
Building a library	285

Final Thoughts

Communicating your design	291
If you read nothing else, read this	294

Appendices

Glossary	303
Bibliography and further reading	314
CSS index	325
Thank you	327

§ We are all Typographers

- Why web typography really does matter
- Embracing the medium
- Preparing the ground

 *CSS was born of typography.
If you design websites or use CSS then
you are making typographic decisions
whether you know it or not.*

Why web typography really does matter

Typography on the web matters because typography matters. Typography is what comes between the author and the reader. If a text has anything at all significant to say, it needs a typographer's care, which will in turn be repaid by the reader's attention.

The majority of content on the web consists of words. If our web of words is to be read, and for that reading experience to be good – to be elevated from poor, past satisfactory, to great – then it's the details that will count. That's what typography itself is all about: the little details adding up to something greater than the sum of the parts.

In 2006, the designer Oliver Reichenstein presented web designers with a call to arms. He notoriously wrote that 'web design is 95% typography'¹. He based this on the supposition that 95% of the information on the web is written language, leading to his assertion that every web designer should get good training in the main discipline of shaping written information: that is, typography.

Whether or not 95% is a true figure, it is indisputable that much of the web is intended for reading: customer reviews, charity campaigns, industry reports, social network updates, blog posts, newspapers, magazines, books, wikis, email, and far more besides. It's also clear that much of the written web is designed with the same uninspiring, uninviting, unreadable uniformity. This needn't be so, and among the purposes of this book are to show how, and provide some of the training called for by Reichenstein.

¹ 'Web Design is 95% Typography' (<http://wbtyp.net/28>) by Oliver Reichenstein on *iA.net* (2006).

The job of typography

The revered type designer Hermann Zapf said that, for designers, ‘typographic design is sometimes misconstrued as a form of private self-expression’². Typography is not art – it is craft and design with purpose. It is there to perform a service for the reader. Quoting another great designer, Emil Ruder said in 1969³:

Typography has one plain duty before it and that is to convey information in writing. No argument or consideration can absolve typography from this duty. A printed work which cannot be read becomes a product without purpose.

A website which cannot be read is a product without purpose. Typography’s chief role is to ensure legibility and readability, but that is not its only job. Typography should invite the reader into the text. It should honour and enhance the tone and message, and clarify the structure and relationships with other elements. Typography should create the ideal conditions for consuming the text, be it immersive reading, scanning for information, or somewhere in between.

Good typography induces a good mood

Typography has a direct, visceral influence on the reader. As well as the practical service of enabling readers to understand and absorb the text, it demonstrably affects their emotional and physical state in ways that few other forms of communication can.

In 2005, a report⁴ entitled ‘The Aesthetics of Reading’ was published by Dr Rosalind W. Picard of MIT, and Dr Kevin Larson, a cognitive psychologist on staff at Microsoft. This publication detailed research which revealed that while typographic quality has little measurable effect on reading speed and comprehension, ‘good quality typography is responsible for greater engagement during reading’.

The research showed that while the quality of typography didn’t necessarily have an impact on the *actual* reading speed, it did have a measurable effect on the *perceived* reading speed. With good typography, participants significantly underestimated the time taken to read a passage. There was

2 Zapf quoted on the cover of Robert Bringhurst’s *The Elements of Typographic Style*.

3 *Typography: A Manual of Design* by Emil Ruder (1967).

4 ‘The Aesthetics of Reading’ PDF (<http://wbtyp.net/22>) by Kevin Larson & Rosalind W. Picard (2005).

also ‘reduced activation in the corrugator muscle’ with better typography, which in plain English means readers frowned less. These combined factors led scientists to conclude that ‘good typography induces a good mood’.

Until recently, the only evidence in favour of spending time and energy on good typography was our own empirical sense of design and aesthetic, combined with the weight of history and inherited typographic wisdom. Now there is a whole new wave of psychologists and typographers doing serious peer-reviewed experimental research on the effects of good typography and type design. For the most part, it is evident the old masters were right all along.

Embracing the medium

The web is often described as a medium, but it is no more a medium than ink. To be a medium, ink must be combined with paper and a process of imprinting letters, otherwise ink is merely a means of transporting pigment. Similarly, the web is merely means of transporting data and structuring ideas. It is only when these meet a combination of software and hardware that we have a medium for design. But unlike all written material before it, that medium is under the control and influence of the reader.

When the World Wide Web was launched over twenty years ago, web pages had no design component. Web sites consisted of linked documents loosely structured by the hypertext markup language (HTML) inherent and unique to the web. This, as now, defined parts of the document as headings, paragraphs, lists, citations, quotations, emphasis, code samples, and so forth. It was left up to the reader's browser software to determine how these elements would look on their screen or other hardware.

The web is arguably the best medium for text there has ever been. Readers can adjust the medium to their advantage or when affected by the external environment. From the beginnings of the web, the reader has been able to tweak the display of text to their own satisfaction, be it colour, size or font choice (albeit initially limited to those typefaces installed on their computer).

Five years after the launch of the web, cascading style sheets (css) were introduced as a way of applying a presentation layer to web pages, and it's where all a web typographer's typesetting is applied. css is a layer that can be peeled away, in part or in whole; and it's a layer, not of definitive design directions, but of guidelines, hints and nudges.

Learn to relinquish control

Designing websites is a strange and wonderful thing. Since the invention of the printing press, designers and printers have decided on the stock, the dimensions and layout of the page, the choice of typeface and the size of font. As the scale of print runs increased, these decisions may have become based on commercial as well as aesthetic concerns, but aside from very wealthy and demanding individuals, the reader will have had no direct influence on the medium or the typographic design.

With the web, the opposite is true. The designer has no direct influence on the reader's medium. Not only can the designer not change their reader's hardware or software, the reader can override completely any directions the designer might send along with the content. This is as it should be. It's what makes the web special.

Where a print designer is all-powerful, dictating every aspect of the reading experience, the web designer must know to relinquish control into the hands of the reader and not fool themselves into believing the situation is otherwise. In one of the most important writings¹ on web design, John Allsopp used an extract from Tao Te Ching to make this point beautifully:

*"The sage ... accepts the ebb and flow of things,
Nurtures them, but does not own them"*

The wonderful thing about the web is that it takes many forms and those forms can be shaped by the reader to their benefit. That is a strength not a weakness, a feature not a bug. The control which print designers have – and so often desire when they transition to the web – is a limitation of the printed medium. The output dictated by print designers, often with great skill and the reader's best interests at heart, is inherently a one-size-fits-all solution. It may not suit the short-sighted reader resorting to a magnifying glass, or work well for the commuter struggling on a busy train with a broadsheet. Web designers, then, must be flexible. Their designs must adapt to the environment of the reader, and the reader must be allowed to adapt the design to best fit their needs.

¹ 'A Dao of Web Design' (<http://wbtyp.net/6>) by John Allsop in *A List Apart* (2000).

Preparing the ground

Before beginning any typographic design, get to know the content, the subject matter and the tone.

If you are to do justice to the text and serve your reader well, do your research. Talk to your client, the writers, design researchers, user experience designers, information architects, content strategists and anyone else who can give you a full picture of what you are about to design.

Know what you are designing and for whom

If it is at all practical, read the text in full. Failing that, invest a good deal of time in reading representative content. If that's not possible, you must still get to grips with what the text might be before designing it. Understanding the tenor is as important as addressing the structure and topic.

Find out who your readers are likely to be. Ask about their demographics, devices and context – when and where might they be reading?

Honour the text, don't dominate it

The reason you should know your readership and understand the text you are setting is so you can design specifically for them. Avoid design for design's sake and, worse still, avoid designing to fashion. Trends are harder to bypass as they creep up on us, and in hindsight can define the life and time in which we worked. Be wary of succumbing to their lure and falling into the trap of being on-trend.

Good typography should do its job and disappear. Make your setting fit and enhance the meaning of the text and the words will shine through. Don't design the page beautiful and attempt to fit the text into it. You are a slave to the text, not its master.

1 Setting Type to be Read

- How we read
- The amazing em (and friends)
- Designing paragraphs: line length
- Designing paragraphs: text size
- Designing paragraphs: line spacing
- Alignment, justification and hyphenation
- Responsive paragraphs

 *The vast majority of the web is text, and a website which cannot be read is a product without purpose. You should strive to make the reading experience as good as possible.*

How we read

Typography is not about designing an environment for optimum reading speed, retention or even comprehension. Typography is about designing an inviting, comfortable reading experience. To do this, we first need to know how people actually read.

Broadly speaking, a piece of text will serve one of three purposes. The first is *display* text, used to grab attention and set a mood. It is intended initially for *looking at*. The second type is *reference* text, used for consulting and providing discrete pieces of information. Examples include glossaries, listings, data tables and search results. The third kind is *linear* text, intended for continuous reading, such as books, news articles and product descriptions. We will cover the first two types of text in detail later in the book, but for now we will focus our attention on linear text as this is where the reading experience must be at its most comfortable and inviting.

Readers approach linear text on the web in a number of different ways. People *skim* read, casting their eye over text, reading words and sentences here and there to get a sense of the content. People also *scan with purpose*, jumping from section to section, looking for a particular piece of information. In doing so they might skim headings and glance at chunks of sentences. Finally, people read in an *engaged* manner once they find a passage or entire page they are interested in. Readers will slow down and consume the whole text, becoming *immersed* in the words.

The immersive *flow* state is the ultimate condition for your reader to reach. Aim for them to be absorbed by the text, unaware of their surroundings, with time flying by unnoticed as the words consume them.

Only engaging writing will result in this state of immersion, but good typography can help your reader achieve it. More importantly, bad typography can prevent your reader reaching flow. Our job is to respect both reader and writer, by providing the typographic conditions for immersive flow to occur. Designing for immersion is what we will concentrate on in the first section of this book.

We read in hops, skips and jumps... and pauses

When we read linear text, we don't do so in a smooth linear manner. The lenses in our eyes can only focus with total acuity on a tiny section in our field of vision. This is known as *foveal vision*. Everything outside the centre of our vision gets more out of focus the nearer it is to the periphery. Because only a small part of our vision is sharply in focus, our eyes can only register a few letters at a time.

We can only see four to five letters at a time with complete acuity

We can only see four to five letters at a time with complete acuity

We can only see four to five letters at a time with complete acuity; however, the brain assumes all the text is in focus.

Foveal vision makes us read by skipping our eyes along lines of text in quick movements called *saccades*. Between each saccade we pause long enough to see the characters on the page clearly, and as we do so, we *fixate* very briefly on the letters within words, and occasionally the spaces between them.

Foveal vision makes us read by skipping our eyes along lines of text in quick movements called saccades.

Circles show where we fixate, arrows show where our eyes jump. Note our eyes sometimes jump back to verify.

As soon as our brain recognises a group of letters, our eyes move on; meanwhile, our brain puts the letters together and searches for word matches. The brain tries to guess ahead but sometimes sends the eyes back to confirm it got the word right. When we get to the end of a line, our eyes break

off and jump in one big saccade, rejoining the text at the beginning of the next line.

When we learn to read, our initial attention is on processing individual words, translating graphemes (spelling patterns) out loud into their corresponding phonemes (spoken sounds). As we become competent at reading out loud, we're encouraged to read silently. As we become more experienced at reading to ourselves, we begin to develop the ability to follow text in saccades and fixations rather than word to word.

For competent readers, these mechanisms happen extremely quickly and subconsciously. Just as we rarely notice when we blink, nor see the individual frames of a film, so it is we don't notice the blurred letters, saccades or fixations. It's a smooth, seamless process. The duration of the fixations and the size of the saccades can vary, though. These depend on our proficiency as readers and our familiarity with the subject matter. They can also be hugely affected by the setting of the text, from the choice of typeface to the spacing between the letters, words and lines.

When designing text to keep your reader immersed, the key is to help your reader's eyes move along the current line with as little back-pedalling as possible, and – most importantly – make an accurate jump to the beginning of the following line. You need to gently guide their gaze across the screen and deftly point them onwards and downwards.

This highlights the difference between readability and legibility. *Readability*, which we are currently concerning ourselves with, is the level of comprehension and visual comfort when reading typeset material. *Legibility* refers to the clarity of individual characters and how easily they are deciphered. Legibility belongs to type designers more than web designers, but can be affected by typesetting.

Arranging type well on whatever constitutes a page is the main undertaking of the typographer designing for readability. However, it is far from the only concern of the web designer. The printed page works by light reflecting off the paper and into our eyes. So long as there is enough light to reflect, paper will work just as well wherever the reader is. For someone reading the web, their page is far more likely to be *emitting* light from little dots through a slab of glass, and that can be problematic. Bright lights, darkened rooms, sunlight, reflections; low resolutions, tiny screens, giant screens, distant screens; vibrations, low bandwidths and slow processors; all of these conspire against your reader. To make type truly readable and bring about a state of repose, we must make our typography perform as well as it can in all situations, and this will be our fundamental consideration throughout the book.

The amazing em (and friends)

The web has always been a type-based medium. Cascading style sheets joined HTML as a part of the web in 1996, and did so with typographic foundations from their inception. The principal underpinning those foundations is the em unit.

The beauty of the em in css is that it provides a unit of length which relates directly to font size. Ems are indispensable to modern web design. As you will see throughout this book, ems enable us to design truly scalable web pages, which is why we are introducing them before going any further.

Ems have a long-standing tradition in typography, where they have typically been used for horizontal measurements. Ems are named after the letter *M* (pronounced as you would the letter), but are not directly related to the character itself. It is more likely the em was derived from the width of the metal *sort* which held a capital *M* in moveable type printing presses.



Some metal type sorts (mirrored for clarity). © Barbara Hauser

In CSS, the em is a general unit of length related directly to font size. We use it to define widths, heights, padding, margins and other measurements in both horizontal *and* vertical directions – something typographers with a traditional background may find surprising. The em unit can also be used to define the font size itself.

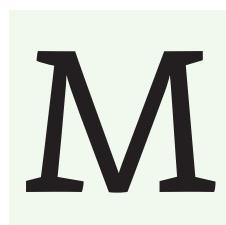
The relationship between ems and font size is the same on the web as it is in traditional printed typography. It's very simple: one em is a distance equal to the font size. If the font size of an element (`<body>`, `<div>`, `<h1>`, `<p>`, ``, and so on) is 16 pixels then one em is equivalent to 16 pixels; in an element with 36-pixel type, one em equals 36 pixels; and for 120-pixel type one em is 120 pixels.

Just as in traditional typography, a crucial aspect of the relationship between ems and font size is that it is independent of the font's design. One em equals the font size whether the typeface is an elaborate calligraphic script, Japanese kanji, or a plain sans serif. To illustrate the relationship between length and font size, consider these styles:

```
#box1 {  
    font-size: 36px;  
    width: 1em;  
    height: 1em;  
  
#box2 {  
    font-size: 120px;  
    width: 1em;  
    height: 1em;  
}
```

These styles will render like this:

M and



Both boxes have a height and width of 1em, but because they have different font sizes, one box is bigger than the other. Box 1 has a font-size of 36px so its width and height is also 36px; the text of box 2 is set to 120px and so its width and height are 120px.

The fundamental advantage of using ems is that distances, lengths and spaces will all scale proportionately with text size. If your reader adjusts their default text size to better suit their requirements, everything sized in ems adjusts itself accordingly. This effect is particularly useful when relationships between elements on the page are tied to type size; margins and padding, for example.

As we saw above, lengths set in ems are relative to the font size of the selected element. However, when setting `font-size` in ems, we do so relative to the *inherited* font size; that is, the font size of the element's parent. Consider the following markup:

```
<div id="d1">  
  <p>...</p>  
</div>  
<div id="d2">  
  <p>...</p>  
</div>
```

with the following styles applied:

```
#d1 { font-size: 16px; }  
#d2 { font-size: 32px; }  
p { font-size: 1em; }
```

These will render as:

#d1 This paragraph has `font-size` set to 1em and inherits a font-size of 16px from its parent `<div>`.

#d2 This paragraph has `font-size` set to 1em and inherits a font-size of 32px from its parent `<div>`.

Even though both paragraphs have `font-size:1em` they display different text sizes because their parent elements have text sized differently.

Rem units

In web pages, the `<html>` element is known as the *root*. Which is what the *r* in rem stands for.

The rem unit was introduced to CSS3 in 2005. It is similar to the em in that it lets you set lengths relative to font size. The key difference is that rem units are not relative to a selected element's font size; they are always relative to the root `<html>` element. By extension, this means that rem units are always directly related to the browser's default text size, which can be adjusted by the reader. This gives us additional precision and ease of use, while still ceding ultimate control to our reader.

If a browser's default text size is `16px` then `1rem` is always `16px`, regardless of where a selected element might sit in the page or what its context might be. If your reader changes their default text size to `21px`, then `1rem` will always be `21px`. Let's take our previous example and set the paragraph font size in rem instead of ems:

```
#d1 { font-size: 16px; }
#d2 { font-size: 32px; }
p { font-size: 1rem; }
```

These will render as:

- #d1 This paragraph has `font-size` set to `1rem` so it does not inherit the font size from its parent `<div>`.
- #d2 This paragraph has `font-size` set to `1rem` so it does not inherit the font size from its parent `<div>`.

Use rem units for global sizing; use em units for local sizing

In ems and rem units we have two extremely useful and versatile units which enable us to relate lengths, distances and dimensions to font size. Unfortunately, it is not always obvious when it's better to use one rather than the other. We'll be using both rem and ems throughout this book so the advantages of each unit will become clear, but as a guideline, use rem units to scale something with the page (global sizing) and use ems to scale within a component (local sizing).

Take the example of a pull-quote containing two short paragraphs. The spacing between the paragraphs will depend on the size of the paragraph text, so you could feasibly set that spacing in either rem or ems.

Should you decide during your design process that the pull-quote should be set a bit larger, then the spacing between the paragraphs must also be increased. Therefore, the paragraph spacing is *directly related to the paragraph text size* and should be considered as local sizing within a component. Hence, you should use ems in this case.

As for the paragraph text itself, that does not have any direct relation to another part of the page. It is only *related directly to the default text size*. Therefore, the paragraph text size can be considered to be global sizing and thus should be set in rem units rather than ems. We will examine text sizing in much more detail later on.

Ch units

The ch unit was introduced in CSS3 in 2006. It is equivalent to the width of a character, hence ch. More specifically, 1ch equals the width of the zero (0) character in the current font. This means that unlike ems and rem units, which do not change with different type designs, a ch changes as the font family changes. In cases where it is impossible or impractical for browsers to determine the width of a 0 (perhaps because the font doesn't include such a glyph), browsers will set 1ch to be equal to 0.5em.

Typography Should Be Invisible

Typography demands a humility of mind, for the lack of which many of the fine arts are even now floundering in self-conscious and maudlin experiments. There is nothing simple or dull in achieving the transparent page.

Typography Should Be Invisible

Typography demands a humility of mind, for the lack of which many of the fine arts are even now floundering in self-conscious and maudlin experiments. There is nothing simple or dull in achieving the transparent page.

Two text blocks are set to 34ch wide, but the use of a condensed font (*top*) and an expanded font (*bottom*) makes the rendered width narrower and wider respectively.

The ch unit can be useful if you want to set lengths or sizes that relate directly to the width of the font. For example, you might want to set the width of a block of text to be wider for an expanded font, and narrower for a condensed font. Using ch could achieve this automatically for you. In the preceding example, both text blocks have a width of 34 ch, but because they use fonts of different widths, the blocks are different sizes.

css includes many more units of length: you may have noticed we've omitted to mention pixels at this point, and there are plenty of others we haven't yet touched on: 1ex, for example, is a length equal to the current font's x-height.

Nevertheless, em, rem and, to a lesser extent, their cousin ch provide us with the primary tools for designing and developing websites that bend, squash, stretch and – most importantly – *adapt* to the requirements of our readers and their web-enabled devices.

A pixel is not a pixel

It used to be the case that a pixel was the smallest unit in web design and considered indivisible. This was because a pixel was the tiniest dot any computer screen was capable of displaying. We now know these more specifically as *hardware pixels*.

For many years, screen resolutions remained reasonably static. One pixel rendered on a screen could fairly reliably be measured as being 1/96 of an inch across (96 ppi). Pixels were hardware pixels and this served us well, even when LCD screens became more commonplace and subpixels were able to be used for text smoothing and graphics rendering.

The situation all changed with the advent of very high-resolution smartphone screens, popularised by Apple's double-density Retina displays. The first iPhones had 3½" screens with a resolution of 320×480. The resulting 163 ppi was close enough to the more usual 96 ppi for pixels to still be hardware pixels. In other words, 16 px text would be rendered 16 physical pixels tall just like on any other screen.

When the iPhone 4 was introduced, however, it too had a 3½" screen but with a resolution of 640×960, resulting in 326 ppi. This meant if 16 px text was rendered using 16 physical pixels it would be half the size of the same text on an earlier iPhone. Clearly this would have been an untenable situation for both reader and designer, so the definition of a pixel was changed in css2. Where pixels used to be hardware pixels they are now *reference pixels*.

Subpixels are the red, green and blue dots which comprise a hardware pixel on an LCD screen.

The official, somewhat impenetrable definition in the specification¹ is:

The reference pixel is the visual angle of one pixel on a device with a pixel density of 96 dpi and a distance from the reader of an arm's length. For a nominal arm's length of 28 inches, the visual angle is therefore about 0.0213 degrees. For reading at arm's length, 1px thus corresponds to about 0.26 mm (1/96 inch).

At its simplest, this means that pixels in CSS are no longer individual physical picture elements – the smallest dots a screen can produce – but a measure of distance equal to 1/96 of an inch. Old displays, therefore, adhere to the new specification (by definition, each pixel on a low-tech 96 ppi screen will be 1/96" across), and higher-resolution screens should try to make CSS pixels that same size.

The upshot is that an iPhone with a Retina display renders 16px text at exactly the same size as older iPhones with low-resolution displays. By extension, an iPhone 4 also *reports* its screen resolution to be 320×480, even though there are 640×960 actual pixels built into the display. We call this the *logical resolution*. The same applies to newer iPhones and other very high-resolution screens.

There is, however, a complication in the definition of a reference pixel which leads to inconsistency in the way it is interpreted. A CSS pixel is defined as being 1/96" *on a screen held at arm's length*. This implies that reference pixels on a screen held closer to the face should be smaller than 1/96", and pixels on a screen further away than arm's length should be bigger than that. This makes sense but adds subjectivity in the way devices render CSS pixels and how they report their logical resolutions².

CSS pixels rendered on double-density screens vary between manufacturers and models. On most mobile devices, reference pixels are almost always smaller than 1/96". Phones are held closer than arm's length so this fits the specification – but the range in rendered size is somewhere between 1/110" and 1/150". The variation is even greater across tablets and mini-tablets. While you'll have a vague idea, then, you'll never quite know exactly how big a pixel actually will be, even with software rather than hardware determining its size.

1 ‘Absolute Lengths’ in [CSS Values and Units Module Level 3](http://wbtyp.net/103) (<http://wbtyp.net/103>).

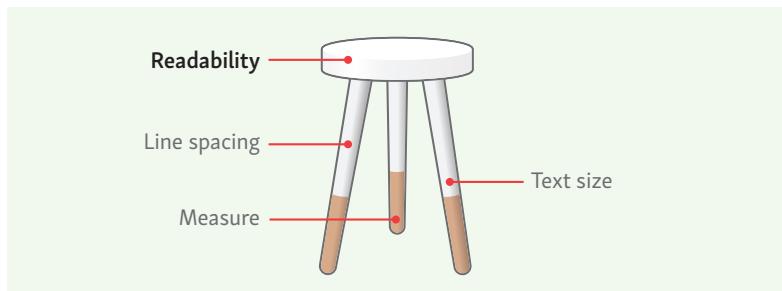
2 For more detail on logical resolutions see ‘[The Ultimate Guide To iPhone Resolutions](http://wbtyp.net/26)’ (<http://wbtyp.net/26>) from PaintCode.

Designing paragraphs: line length

A paragraph represents an individual unit of thought. From a typographer's point of view, a paragraph is the shortest passage of text which we should design for immersive reading. Your reader should be able to absorb the writer's thoughts without difficulty or interruption.

Immersive reading is the most intense and involving form of reading. The paragraph is an ideal place to start when designing the typography of your website. Making a paragraph readable is about helping your reader's eyes stay on the line they are currently following, pass along it with as little back-and-forth as possible, and when they reach the end of the line, reattach effortlessly to the next one.

Three factors combine to make a paragraph readable: the triumvirate of line spacing, measure and text size. Think of them as legs on a three-legged stool – you'll upset the balance if you change one leg without adjusting the others.



Readability balances on a stool with legs of line spacing, measure and text size.

Of the three legs, measure is the best place to start as it has the greatest influence on your reader's ability to go from line to line.

Choose a comfortable measure

Measure is a typographer's term for the length of a line of text. Web designers might describe this as the width of a paragraph. If left unchecked, measure is the factor which can vary most in a web context and thus have the greatest effect on readability.

If you set your lines too long, your reader will have difficulty in following from one line to the next; they may mistakenly skip lines or reread the same line. Set your measure too short and your reader's eyes will have to change direction too frequently, resulting in tired eyes and a reading experience that feels like hard work, however engaging the text.

Lean on six centuries of typesetting experience

The often quoted ideal length for a line is 66 characters, with a range of 45–75 characters (including spaces) generally considered satisfactory. Rather than representing a predefined guideline, these figures reflect the de facto habits of master printers and typesetters since the invention of the moveable type printing press nearly six hundred years ago. In fact, as Geoffrey Dowding put it in *Finer Points in the Spacing and Arrangement of Type* in 1966:

The first printers inherited what was, and had been for centuries, the accepted usage of the scribe: and the best of our contemporary printers adhere to this for the eminently practical reason that its observance still produces the most readable and craftsmanlike work.

The implication is that any book, newspaper or other reading material printed in the last few centuries is likely to be set with lines containing between 45 and 75 characters.

That said, we established in '[How we read](#)' that reading screens is different from reading from paper. The very limited amount of research¹ that has been undertaken on reading from screens indicates that longer line lengths (up to 100 characters) do not adversely affect readability, and can in fact increase the speed of reading. The temptation, therefore, might be to use

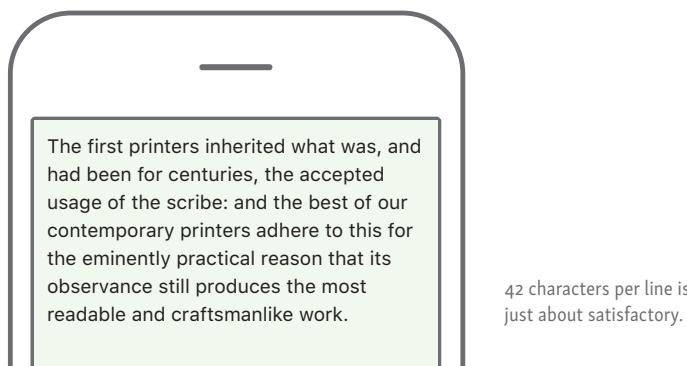
¹ '[How physical text layout affects reading from screen](#)' (<http://wbtyp.net/14>) by M. C. Dyson in *Behaviour and Information Technology* 23/6 (2004).

longer lines on screens. However, the same studies revealed that readers stated *a preference for shorter lines*. This is an important finding. Readability is as much about setting text that *wants to be read*, as it is about setting text that *can be read quickly and easily*.

It leads us back to our history and the six centuries of print typography that have come before us: aim for setting paragraphs 45–75 characters wide, just as was being done on vellum all those years ago. Remember, though, that this range is an observation, not a mandate, so a little either side – for good reason – won’t do any harm.

Set text for mobile first

All modern web designers are taught to think *mobile first*. That is to say, your starting point of any design consideration should be for small screens and less capable devices. This applies just as much to the typography as it does to any other aspect of web design, so let’s take our paragraph and put it in the context of a smartphone.

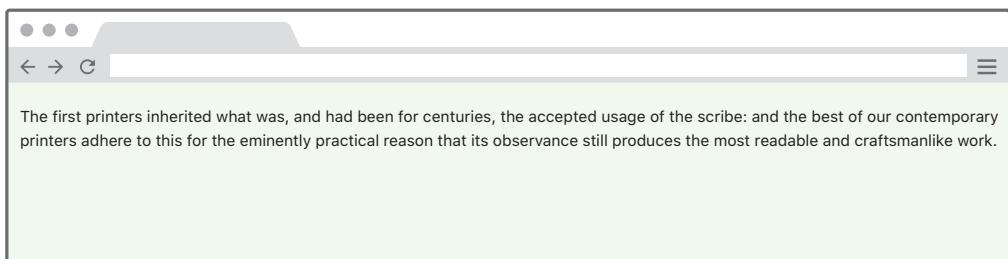


As you can see, on a small screen, such as a typical smartphone, you will usually fit around 40 characters per line, assuming the default text size and typeface. This is just below the lower end of the range for a satisfactory line length, so you certainly wouldn’t want it shorter.

You could fit more characters on a line by reducing the size of the text, but the standard text size of any device will normally have been set by the manufacturer to be what they consider comfortable for the majority of their customers, so it’s best leaving the default as the starting point for your paragraphs. We’ll discuss potential refinements to this guideline in ‘[Designing paragraphs: text size](#)’.

Use a liquid setting

On larger screens, lines of text will be longer unless you do something to stop that happening. Here, for example, you can see the lines are around 135 characters long:



135 characters per line is too long for comfortable reading.

The lines are clearly far too wide for a good reading experience. We need to shorten them to keep them within our ideal range of 45–75 characters.

In CSS we don't have characters as a consistent unit of length. The ch unit we introduced in the chapter '[The amazing em \(and friends\)](#)' is equal to the width of a character. However, this varies from font to font so it's not usually suitable for layout purposes as it doesn't offer the consistency we'd need. We can, though, use the em unit instead. One em is roughly equal to two characters, so to achieve a width of 45–75 characters, our paragraphs should be set to the equivalent range of 23–38ems.

Thinking in terms of a range rather than an absolute figure such as 33ems is extremely helpful when considering designs that can adapt to different screen sizes. It means your design can be far less opinionated about the size of the browser window, which is good for your reader. Your design will mould itself to the size and shape of your reader's chosen device, as opposed to their device attempting to crowbar your design into its screen.

This line has sixty six characters, counting both letters and spaces. It is about 32 ems long and suitable for long-term reading with text set in paragraphs. On average, 1 em accounts for two characters, so a satisfactory line length (also known as the measure) is 22 to 38 ems.

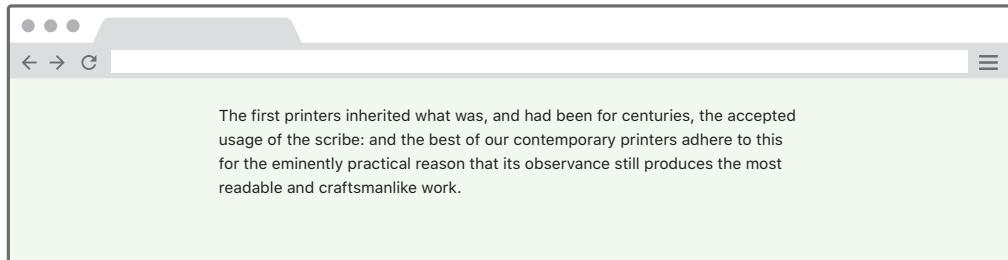


66 characters is about 32ems.

The easiest way to restrict the width of your paragraphs to the top end of the readable range is to set a `max-width` on the paragraph:

```
p { max-width: 38em; }
```

This rule means paragraphs will still fill the whole width of smaller screens, but in larger windows lines will never be longer than 38ems – the upper limit of our range. As an additional nicety, you can centre the column of text by adding `margin:auto`.



Lines centred and limited to 38ems long.

The important thing to note here is that we are *not setting a width* for the paragraph. We are letting the lines of text vary in length according to the `viewport` (the screen size or browser window), and stopping the lines becoming too long where necessary, based on the number of characters in the line. This technique is known as liquid or fluid layout, and ensures your typesetting is independent of – and adapts to – your reader’s device. Fluid layouts are the very essence of the web typographer’s craft. Next, we’ll learn how to bring text size into the equation.

Designing paragraphs: text size

Text size is the second leg of the readability stool. Start with a size suitable for comfortable reading, which could vary with your choice of typeface and the distance of the screen from your reader's eyes.

Use the default font size for paragraphs

Your starting point for paragraph text should be whatever size has been set as the default in the browser. Device manufacturers will have determined an appropriate initial size, usually 16 px. Be aware that your reader may have adjusted the default size to better suit their needs. You should respect your reader's wishes – if they've gone to the effort of adjusting the font size (especially the browser default) it will be for a good reason.



Some people set their default text very large indeed.

Many designers, especially those new to the web, are surprised at how large the default text size is on screen compared with printed material. But this comparison only holds up if you compare the two media side by side. We tend to read screens from further away than the distance at which we hold books, newspapers or magazines, so the perceived screen and print text sizes are actually similar.

Your inner designer may err towards smaller text, but from your reader's perspective it is safer to make the text a little too big than too small. It is true that your reader can adjust the size of text within their browsers, but that's no excuse for setting type too small in the first place. Why make them go through the effort of correcting the type size? You're the typographer – it's your job to set the type correctly, and the default is the size at which browsers were intended to display text.

Take small print into account

You should consider whether the text you are designing is likely to include long passages set smaller than your regular body text. Examples of this include small print, annotations and comments.

You may not yet have decided whether your small print should actually be set in a smaller size, but now's the time to start thinking about the possibility. Your small print will need to be differentiated from your regular text, but if dropping down a text size means your small print becomes illegible, you'll need to bump up its size a bit and increase the body text to accommodate it.

Reversed-out text

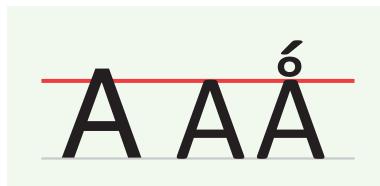
Be aware that light text set on a dark background can often do with being set a little bigger, particularly when viewed on lower-resolution screens. Experiment to see what works for you.

Adjust the font size if the typeface requires it

The browser default is always the best place to *start* for text size. However, typefaces can vary significantly – even among ones suitable for continuous reading – and your choice of typeface may require you to adjust your paragraph text size up or down from the default.

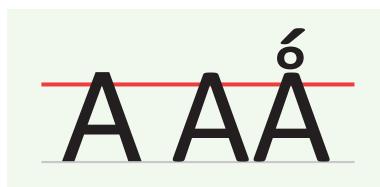
Some fonts are simply smaller owing to the way they were built. Often these are fonts which were designed from the start with heavily accented characters or the need to accommodate non-Latin scripts such as Arabic.

A good example of this is Calibri, which contains the character Å, a rarely used double-accented Danish glyph. It is an extremely tall character and, to fit the diacritics into the font's vertical metrics, Calibri is built with smaller letters compared to similar typefaces.



Helvetica's (left) is drawn bigger than Calibri (right). Here the glyphs are set at the same font size, despite appearances to the contrary.

Calibri's capital letters are about 10% shorter than the version of Helvetica included in macOS, so to achieve parity with Helvetica at a typical default of 16 px, you should set Calibri at 18 px.



Helvetica set at 16 px compared with Calibri at 18 px (*magnified*).

Fonts smaller (or bigger) by design

A given typeface may appear smaller or bigger compared with another, despite the two fonts being constructed with similar vertical metrics. It is the design of the typefaces that makes them look different sizes, rather than the way they have been developed. The main difference in design is usually the x-height. X-height refers to the size of the lowercase letters relative to the size of capital letters. A shorter x-height leads to a typeface which looks smaller.

Helvetica has a more generous x-height than many other fonts, but as it's commonly used as a default typeface we can employ it as our benchmark. Compared with Helvetica, Altis has a tall x-height, so to match the apparent type size of Helvetica you would need to set Altis at a smaller size. Conversely, Lato has a slightly shorter x-height than Helvetica, so you might consider setting Lato slightly bigger than the default. Futura has an even smaller x-height, so you would need to set Futura significantly bigger to match the perceived type size of Helvetica.



Clockwise (from top-left): Altis, Helvetica, Lato and Futura showing a variety of x-heights.

When deciding precisely how much to adjust your typefaces there are two approaches. First, you could do so by eye and experiment. Set a paragraph in Helvetica alongside the same paragraph set in your choice of typeface. Using your judgement, tweak the type size of your font one pixel at a time until the lowercase letters look about the same size as Helvetica. It's not an exact science but any nudge in the right direction will make a difference to your reader.

Helvetica 16px

One more attribute the modern typographer must have: the capacity for taking great pains with seemingly unimportant detail. To them, one typographical point must be as important as one inch, and they must harden their heart against the accusation of being too fussy.

Futura 20px

One more attribute the modern typographer must have: the capacity for taking great pains with seemingly unimportant detail. To them, one typographical point must be as important as one inch, and they must harden their heart against the accusation of being too fussy.

Futura must be set at 20px to match the perceived size of Helvetica at 16px.

Calculating type size adjustments

An experienced typographer will be able to tweak type sizes by eye. It is also possible to use the internal font metrics to calculate a size adjustment to make one font appear the same size as another. The theory is to adjust the font size so that the x-heights are the same. The relative height of lowercase

letters to their uppercase counterparts is commonly referred to as the *aspect value*; it is this value we will use in our calculations. Precisely defined, the aspect value is equal to the x-height of a font divided by the font size.

A typical aspect value is about 0.5, meaning the x-height is half that of the type size. The beauty of type is that there's no such thing as an average typeface, and it's the differences we are making adjustments for. If you have access to font editing software, you can determine the aspect value from the font metrics used in constructing the fonts. A quicker and easier way is to use an online aspect value calculator¹. Using such tools you'll find that the fonts we've been looking at have the following aspect values:

Typeface	Aspect Value
Helvetica	0.521
Lato	0.507
Futura	0.417
Altis	0.542

Given this information, we can calculate a text size for our alternative fonts that will make them *look* about the same size as Helvetica. Here's how. Assuming that Helvetica is set at 16 px, then the equivalent size for Lato is:

$$16 \times 0.521 \div 0.507 = 16.5\text{px}$$

Using the same formula with Futura:

$$16 \times 0.521 \div 0.417 = 20.1\text{px}$$

And for Altis:

$$16 \times 0.521 \div 0.542 = 15.2\text{px}$$

¹ Online aspect value calculators by [Bruno Fassino](http://wbtyp.net/59) (<http://wbtyp.net/59>) and [Richard Rutter](http://wbtyp.net/58) (<http://wbtyp.net/58>).

Altis 15.2px

One more attribute the modern typographer must have: the capacity for taking great pains with seemingly unimportant detail. To them, one typographical point must be as important as one inch, and they must harden their heart against the accusation of being too fussy.

Lato 16.5px

One more attribute the modern typographer must have: the capacity for taking great pains with seemingly unimportant detail. To them, one typographical point must be as important as one inch, and they must harden their heart against the accusation of being too fussy.

Helvetica 16px

One more attribute the modern typographer must have: the capacity for taking great pains with seemingly unimportant detail. To them, one typographical point must be as important as one inch, and they must harden their heart against the accusation of being too fussy.

Futura 20px

One more attribute the modern typographer must have: the capacity for taking great pains with seemingly unimportant detail. To them, one typographical point must be as important as one inch, and they must harden their heart against the accusation of being too fussy.

Four typefaces with actual size adjusted to match the perceived size of Helvetica.

The calculations we just went through resulted in font sizes at fractions of a pixel. In reality, it's better not to set your fonts in anything other than whole pixels. Nudging one whole pixel at a time will prevent unevenness in rendering on lower-resolution screens, and also make layout and type-setting calculations easier further down the line.

Use rem units to size your text

Up to this point we've talked about font sizes in terms of pixels. This was just for the convenience of explanation. When it comes to CSS you should use `rem` units to size your text. In the chapter '[The amazing em \(and friends\)](#)' we learned that `rem`s are relative units which enable you to specify sizes in relation to the root `<html>` element's font size. If the default text size of the browser is `16px` then text sized at `2rem` would always be displayed at `32px`, regardless of its context.

In contrast, text sized in ems is always relative to the parent element and is thereby prone to inheritance complications. A common example of this is the shrinking list. Consider the following nested list:

```
<ul>
  <li>Item 1
    <ul>
      <li>Item 2
        <ul>
          <li>Item 3</li>
        </ul>
      </li>
    </ul>
  </li>
</ul>
```

If you size the text using something like `font-size:0.9em`, the list items will get progressively smaller. If you size the text using rem instead of ems, `font-size:0.9rem`, the list items will all stay the same size.

- Item 1
 - Item 2
 - Item 3

- Item 1
 - Item 2
 - Item 3

Left: List items sized in ems. *Right:* List items sized in rem.

So why not just use pixels? In CSS, pixels are absolute units anchored to a physical measurement. A pixel – short for *picture element* – used to refer to a physical pixel on a device, but with the advent of high-resolution screens the definition has changed. As explained in ‘[The amazing em \(and friends\)](#)’, CSS3 defines a pixel to be 1/96th of 1inch (0.26 mm). The idea is that a CSS pixel is equivalent to the size of a pixel if the screen had a resolution of 96 dpi.

In theory, this means that because pixels refer to a physical length, text sized in pixels should stay the same regardless of any settings changed by the reader. In reality, users of most devices can change the text size even when it is sized in pixels; this wasn’t always the case, however.

Microsoft's Internet Explorer 6 (and earlier) wouldn't resize text set in pixels, causing a major accessibility problem for any reader needing to adjust the text size to better suit their eyesight. The same happened more recently with an update to Chrome 52. By the letter of the css specification, these implementations could not really be said to be wrong – and that logic still stands. For the sake of future-proofing, pixels are best avoided for sizing text. Once bitten, twice shy.

Having said that, when web designers think of text size we do tend to think in pixels. To think in rems we need to do a simple calculation: for the size in rem, divide the required size in pixels by 16 (the typical size of the root element). For example, to change your body text to 18 px by setting it in rem, you would calculate the following:

$$18 \div 16 = 1.125$$

with the resulting css being:

```
body { font-size: 1.125rem; }
```

Rems provide the same non-inheriting convenience as pixels, but because they are a relative unit not tied to any physical dimensions, text sized in rem can be resized at will.

Next we will attach the third leg of the readability stool, and consider how line spacing adds stability to the situation.

Designing paragraphs: line spacing

Line spacing is the final leg of the readability stool and plays an important role in reading accuracy and comfort. Line spacing is determined by considering the line length, the typeface, its fit and size, and the typographic context.

Line height should suit text size and measure

Setting a suitable space between lines of text will ensure your reader's eye can rejoin the next line accurately and comfortably. If you set lines too tightly, your reader's eyes will struggle to rejoin the text at the correct line. Reading speed will also be impaired because excessively tight text causes the eye to take in two lines at the same time, leaving your reader struggling to focus and tiring quickly.

Printing demands a humility of mind, for the lack of which many of the fine arts are even now floundering in self-conscious and maudlin experiments. There is nothing simple or dull in achieving the transparent page. Vulgar ostentation is twice as easy as discipline. When you realise that ugly typography never effaces itself; you will be able to capture beauty as the wise men capture happiness by aiming at something else.

Lines set very tightly.

Reading speed also falls for lines too widely spaced. Space lines too loosely and you end up with inefficiently used screen estate, and tired eyes. Like walking to avoid cracks in the pavement, your reader will need to alter their stride unnaturally and concentrate hard just to move forward. Your reader will become fatigued as they hop, skip and jump down the page, with their eyes still struggling to join the correct line and make sense of the paragraph.

Printing demands a humility of mind, for the lack of which many of the fine arts are even now floundering in self-conscious and maudlin experiments. There is nothing simple or dull in achieving the transparent page. Vulgar ostentation is twice as easy as discipline. When you realise that ugly typography never effaces itself; you will be able to capture beauty as the wise men capture happiness by aiming at something else.

Lines set very loosely.

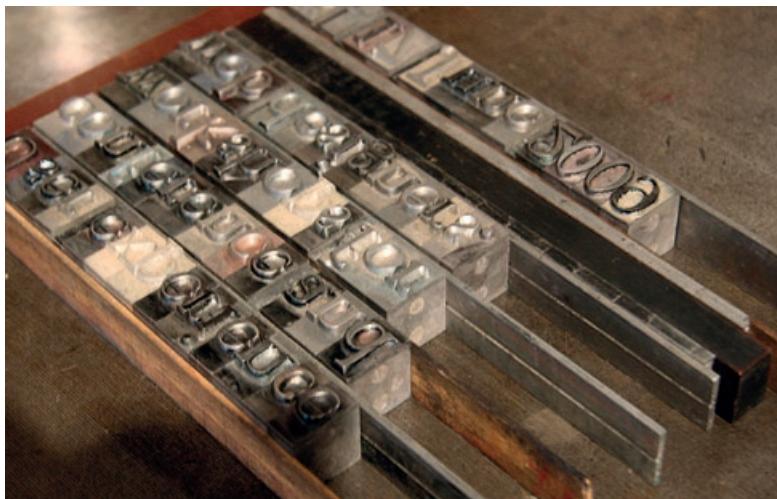
Line spacing set just right for comfortable reading strikes a happy balance between compactness and separation. Your reader is afforded an easy efficiency, with lines of text spaced just far enough to ensure accuracy, but tight enough to reduce fatigue and scrolling.

Printing demands a humility of mind, for the lack of which many of the fine arts are even now floundering in self-conscious and maudlin experiments. There is nothing simple or dull in achieving the transparent page. Vulgar ostentation is twice as easy as discipline. When you realise that ugly typography never effaces itself; you will be able to capture beauty as the wise men capture happiness by aiming at something else.

Line spacing set sympathetically.

Introducing leading

In the predigital age of letterpress, when type was set in rows of metal and wooden letters, strips of lead were used to separate lines. This is the source of the term *leading* (pronounced ‘ledding’) to mean the spacing between lines of text.



Strips of lead used to separate lines of text.

Leading is described in terms of the distance from one baseline to the next, as opposed to the size of a strip of lead. If you set lines of 12 pt type, each separated by 6 pt thick strips of lead, you would say you have 18 pt leading. In the traditional world of metal type, compositors often express leading in combination with the type size. For example, 12 pt text with 18 pt leading would be written using this notation:

12pt/18pt

In digital typesetting, the convention is to set leading as a multiple or percentage. This means 12 pt with 18 pt leading is expressed as:

12pt/150%

or:

12pt/1.5

Use unitless values to set line-height

In CSS the spacing between lines is created using the `line-height` property. This property is surprisingly flexible and allows you to take any of the aforementioned approaches to specifying line spacing. The example at the beginning of the chapter, where we saw line spacing set sympathetically, had 16px text with 24px leading. We could achieve that using `line-height` in any of the following ways:

```
line-height: 24px; /* absolute length */  
  
line-height: 1.5em; /* relative length */  
  
line-height: 150%; /* percentage */  
  
line-height: 1.5; /* unitless multiplier */
```

There is an important difference between how the unitless multiplier and values with units work. The distinction lies in how line heights are inherited by child elements (for example, how a heading would inherit the `line-height` you set on its parent `<div>`). The inheritance rule can catch you out at first as it's not entirely intuitive. With unitless values the *multiplier* is inherited. For all values with units, including px, em and %, the *calculated value* on the parent is inherited. Consider the following markup with three simple `<div>`s, each containing a heading:

```
<div id="unitless">  
  <h1>1 Avoid unexpected results by using unitless  
  line-height</h1>  
</div>  
<div id="units-pc">  
  <h1>2 Avoid unexpected results by using unitless  
  line-height</h1>  
</div>  
<div id="units-em">  
  <h1>3 Avoid unexpected results by using unitless  
  line-height</h1>  
</div>
```

Now apply the following CSS. The headings all have a text size three times the size of their containers, and we apply line-height to the `<div>` in three different ways: without a unit, with a percentage, and in ems.

```
div {  
  font-size: 15px;  
}  
h1 {  
  font-size: 45px;  
}  
#unitless {  
  line-height: 1.1;  
}  
#units-pc {  
  line-height: 110%;  
}  
#units-em {  
  line-height: 1.1em;  
}
```

This results in different line spacing on the heading due to the inheritance rules of unitless and unit-specific line heights:

#unitless

1 Avoid unexpected results by using unitless line-height

#unit-pc

2 Avoid unexpected results by using unitless line-height

#unit-em

3 Avoid unexpected results by using unitless line-height

Line-height inheritance can lead to unexpected results.

In the example above, the heading inside the `<div>` with unitless line height has a leading of 49.5 px because it has inherited the unitless multiplier (45×1.1). The headings inside the `<div>` with line heights set in % and em have a leading of 16.5 px because they have inherited the `line-height` as calculated on the parent `<div>` (15×1.1). In most cases, using unitless values is the safest way for you to set `line-height` without getting unexpected results.

Increase line spacing from the browser default

If you don't specify `line-height` anywhere in your style sheet, browsers will use the setting associated with the `normal` keyword. This is defined in CSS as 'between 1.0 to 1.2' and is left to browsers to decide. For setting type on the web, a `line-height` of 1.2 is usually too small, as comfortable reading on screens tends to benefit from quite generous line spacing. A good starting point is to set a `line-height` of 1.4, but as you'll see later, you may need to adjust that depending on the font.

Adapt the line spacing to suit the typeface

Our examples of text set with comfortable line spacing show an evenness of *colour*. This doesn't mean black (or grey or red or blue) but colour in a centuries-old metaphorical sense. When typographers talk about colour we are referring to the density of the texture of a body of text – the overall blackness of the letterforms en masse against the background. You can think of colour as the amount of ink that would be used to print a given area on a page. The more ink required, the denser the typographic colour.

You can't adjust the *typographic colour* by changing the type's actual colour. Typographic colour is affected by four factors: the design of the type, the spacing between the letters, the spacing between the words, and the spacing between the lines. Changing line spacing changes the typographic colour: the denser the line spacing, the darker the colour. Colour isn't a matter of accuracy or precise measurement, but something determined by eye. It's about perception. When you typeset paragraphs, aim for an evenness of colour as opposed to blotches or stripes. It is the stripes of white between the lines that can be most distracting to readers.

Printing demands a humility of mind, for the lack of which many of the fine arts are even now floundering in self-conscious and maudlin experiments. There is nothing simple or dull in achieving the transparent page. Vulgar ostentation is twice as easy as discipline. When you realise that ugly typography never effaces itself; you will be able to capture beauty as the wise men capture happiness by aiming at something else.

Printing demands a humility of mind, for the lack of which many of the fine arts are even now floundering in self-conscious and maudlin experiments. There is nothing simple or dull in achieving the transparent page. Vulgar ostentation is twice as easy as discipline. When you realise that ugly typography never effaces itself; you will be able to capture beauty as the wise men capture happiness by aiming at something else.

Typographic colour: aim for evenness not stripes.

You need to balance your quest for evenness with a desire for a comfortable amount of blackness to the colour. Too light or too black and the text will become tiring or hard work to read. Colour is affected significantly by the weight of the font, but it's far from being the only factor: colour is just as much affected by the design of the typeface itself. Fonts have more colour if their letterforms have heavy strokes or high contrast between thicks and thins. Wide fonts or those with smaller x-heights have less colour.

Printing demands a humility of mind, for the lack of which many of the fine arts are even now floundering in self-conscious and maudlin experiments. There is nothing simple or dull in achieving the transparent page. Vulgar ostentation is twice as easy as discipline. When you realise that ugly typography never effaces itself; you will be able to capture beauty as the wise men capture happiness by aiming at something else.

Printing demands a humility of mind, for the lack of which many of the fine arts are even now floundering in self-conscious and maudlin experiments. There is nothing simple or dull in achieving the transparent page. Vulgar ostentation is twice as easy as discipline. When you realise that ugly typography never effaces itself; you will be able to capture beauty as the wise men capture happiness by aiming at something else.

The design of a typeface affects the colour. Baskerville (*left*) is much lighter in colour than Adelle.

Fonts with more colour – those which are bolder in weight and shorter in stature – need their line spacing increased slightly to reduce their heaviness and lighten the overall colour. The opposite applies to fonts with less colour. Tighten the line spacing for fonts which are lighter in weight or have a smaller x-height, thereby giving them more presence on the screen and darkening their overall appearance.

Another way to think about it is that the ascenders and descenders of a font effectively push the letters apart. A font with a small x-height has proportionally longer ascenders, pushing the letterforms further apart, so you need to reduce line spacing to bring them closer together. Conversely, you should nudge apart lines of text which are set in a font with a large x-height.

Printing demands a humility of mind, for the lack of which many of the fine arts are even now floundering in self-conscious and maudlin experiments. There is nothing simple or dull in achieving the transparent page. Vulgar ostentation is twice as easy as discipline. When you realise that ugly typography never effaces itself; you will be able to capture beauty as the wise men capture happiness by aiming at something else.

Printing demands a humility of mind, for the lack of which many of the fine arts are even now floundering in self-conscious and maudlin experiments. There is nothing simple or dull in achieving the transparent page. Vulgar ostentation is twice as easy as discipline. When you realise that ugly typography never effaces itself; you will be able to capture beauty as the wise men capture happiness by aiming at something else.

Tightening Baskerville's line spacing to 1.25 compensates for its light colour.

A shorthand for font settings

Earlier we learned that typographers often express type size and leading together. The same can be done in CSS using the `font` shorthand property. For example, this property:

```
font: 15px/1.5 sans-serif;
```

is equivalent to:

```
font-size: 15px;  
line-height: 1.5;  
font-family: sans-serif;
```

You can combine almost all the font-related properties into a single value. Consider this more comprehensive example:

```
font: bold italic 2rem/1.2 "Hoefler Text", "Times", serif;
```

which is equivalent to:

```
font-weight: bold;  
font-style: italic;  
font-size: 2rem;  
line-height: 1.2;  
font-family: "Hoefler Text", "Times", serif;
```

The `font` shorthand makes for a handy shortcut, although there are quite a few gotchas. You must always include the `font-size` and `font-family` portions. If you omit either of these values, browsers will deem your `font` property invalid and ignore it. If you don't include values for `font-weight` and `font-style` the browser will set these to `normal` and they will not be inherited. The order of the properties in the `font` shorthand is also important. You must include the `font-weight` and `font-style` values first, and you must include the `font-family` value last.

Alignment, justification and hyphenation

Paragraphs can be set in three ways: justified, so that both sides have a straight edge; left- or right-aligned, so that they have an uneven edge; or they can be centre-aligned.

Justify well or not at all; if in doubt, align left

Type has been justified since at least 1439, when Johannes Gutenberg introduced the printing press and moveable type to Europe in order to diligently mimic and reproduce scribes' handwritten bibles (see opposite).

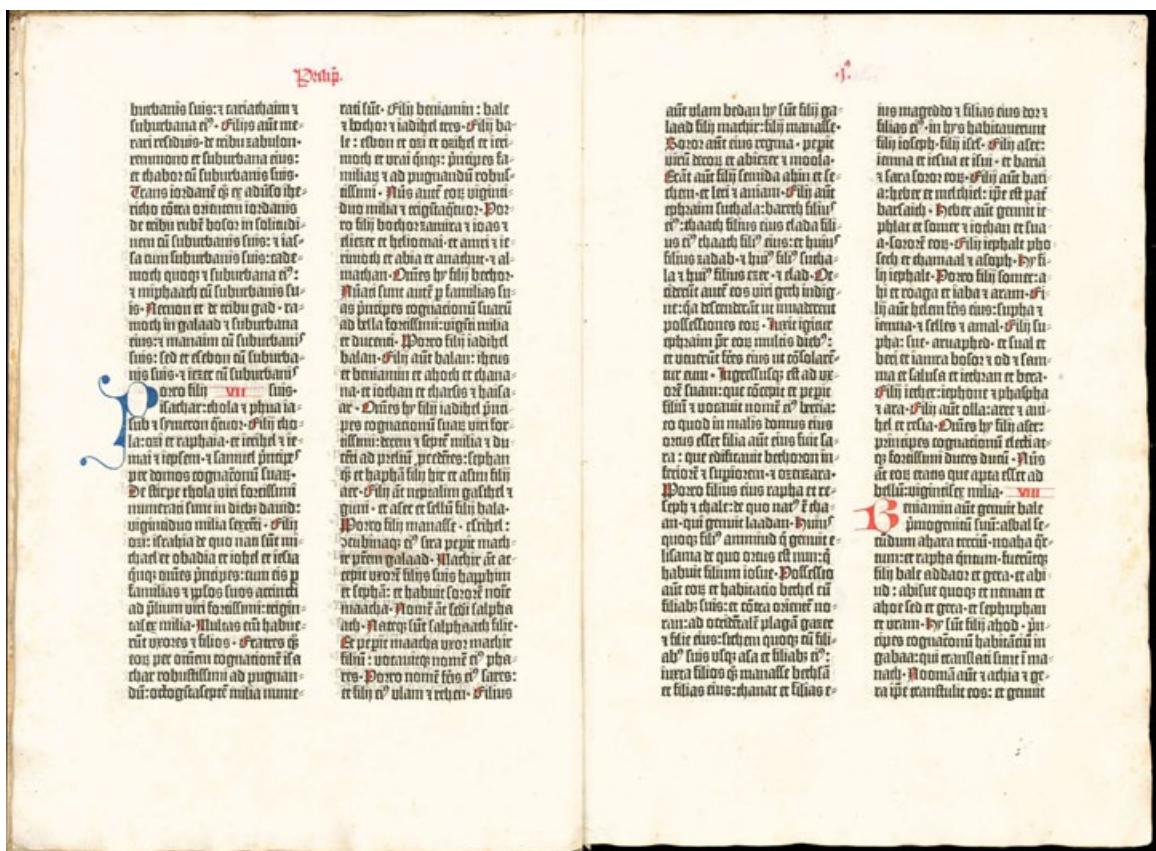
Justified text is an aesthetic design choice which creates neat, straight-edged text blocks and columns on a page. Justified paragraphs can look tidier from a distance but that doesn't mean they are easier to read. Justified text has always precipitated a trade-off between evenness of spacing and neatness of layout – setting lines *without* justifying them is the only way to achieve the optimum space between letters and words.

By default, browsers will align text to the left; or as typographers like to say: *ragged right*. In CSS, text alignment is set with the aptly named `text-align` property, like this:

```
p { text-align: left; }
```

To justify paragraphs of text, use the value `justify`:

```
p { text-align: justify; }
```



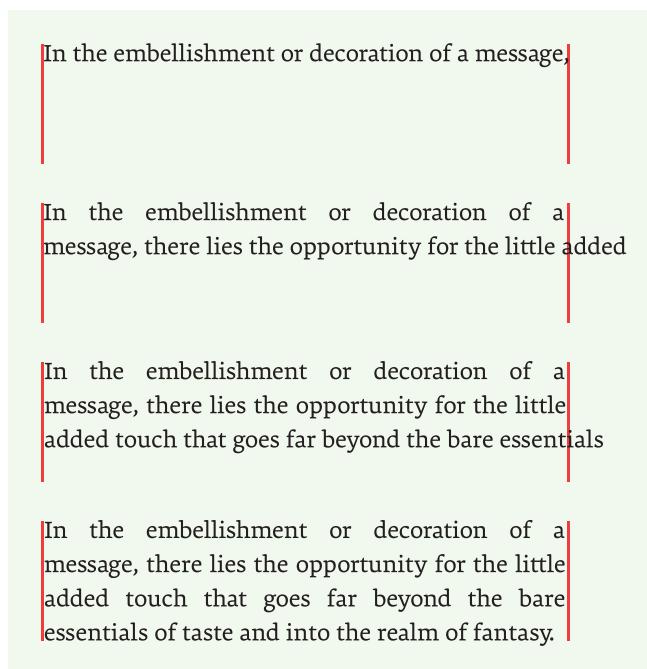
A Gutenberg bible with a four-column spread of justified text.

We could just leave it at that – the CSS could not be simpler. However, the decision to justify text or not on the web has long been contentious. Ideally it should be a pure design choice, playing layout against the needs of the text; unfortunately, our hand is somewhat forced by limitations in the way browsers justify text.

Broadly speaking, when justifying text the spacing along a line is adjusted so that all lines are the same length. The spacing between words can be expanded or reduced accordingly; the spacing between letters can also be adjusted, as can the spacing within letters themselves (for example, all letter Os on a line could be compressed slightly to fit an extra word in). There are different procedures for adjusting the spacing to achieve justification. CSS3 does not specify a complete justification algorithm: it leaves browsers to decide the most appropriate way for themselves.

Type designers would rather neither the letterforms nor the letter spacing were adjusted at all, as letter spacing and the enclosed shapes of characters are intrinsic to the art of type design.

Some procedures for justifying text are better than others. The simplest method is called the *greedy* algorithm. This algorithm adjusts only word spacing, and is called greedy because it is limited to *adding* space between words. Greedy justification attempts to fit each word on a line until it runs out of space and the last word in drops to start the line below. The word spacing in the line is then increased so that the text is spread out across the whole line. The process repeats until the whole text is set.



The justification process of the greedy algorithm.

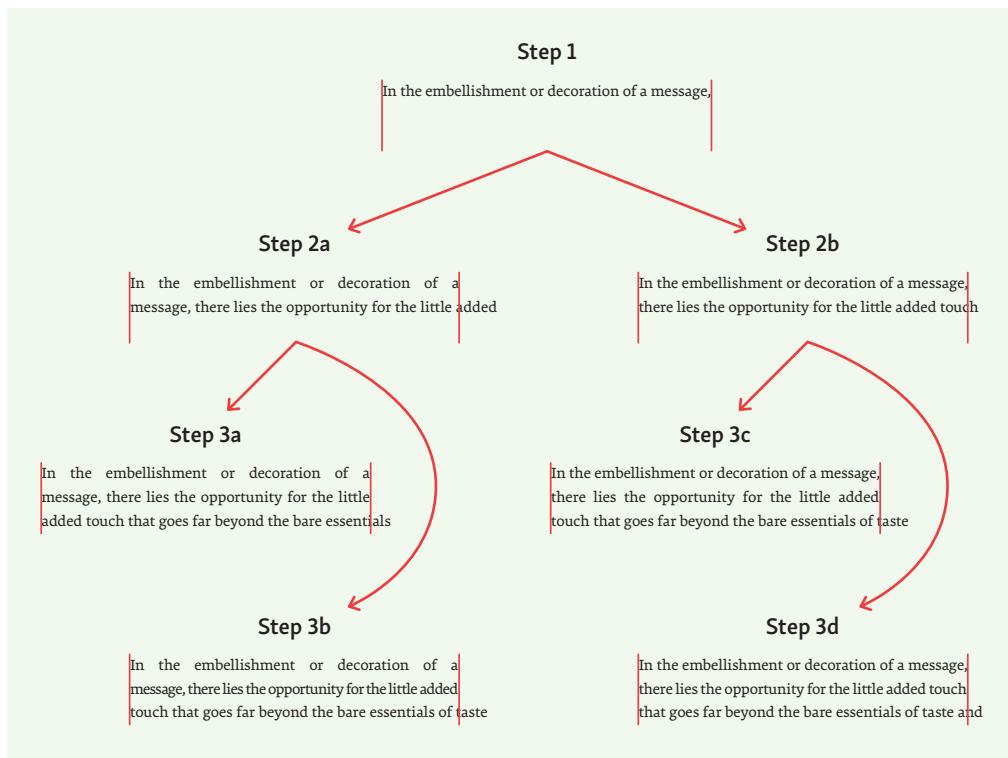
The greedy algorithm is quick and easy for computers to follow, and takes relatively little processing power. It is probably for this reason that this method is favoured by web browsers, which tend to prefer speed over complexity. The greedy algorithm is, however, extremely crude. It will often result in unsightly gaps between words, with *rivers* cascading through paragraphs where spaces between words line up. These disruptions to the tempo of the text interrupt and detract from the reading experience. This is exaggerated on lower-resolution screens where the adjustments to white space may be dealt out in whole pixel intervals, thus widening the word spacing even further.

little added touch that goes far beyond the bare essentials of taste and into the realm of fantasy. For this reason the ornamentation of printing is at once the most charming and the most dangerous diversion | the typographer | can | find: dangerous | because | all | matters | of decoration | call | upon | the | utmost discretion and sense of fitness for their effective use.

Illustration of rivers in greedily justified text.

A more sophisticated justification algorithm is the *Knuth–Plass* method, developed for the TeX typesetting system, and adapted by software including Adobe InDesign and Illustrator. Knuth–Plass differs from the greedy method in two important ways. First, as well as adding space between words, it will *reduce* space, resulting in fewer large gaps between words. Second, it assesses the effect of justification on the paragraph as a whole and will adjust the line breaks to find the set of breakpoints that will give the most pleasing result for the entire paragraph. The algorithm may compromise the spacing of one line to minimise situations where a very loose line is immediately followed by a very tightly spaced line, thus benefiting the paragraph as a whole.

Knuth–Plass is a far more complex algorithm than the greedy method and thus takes more processing power. Additionally the memory requirements increase quadratically with the paragraph length, so browser makers are wary of implementing Knuth–Plass despite the obvious benefits to the reader. That said, there should still be scope to use a more sophisticated version of the line-by-line greedy method in order to reduce, as well as increase, spacing between words and letters.



The Knuth–Plass method:

The first line is inserted word by word until it no longer fits (1). The algorithm now has a choice: it could move the word to the next line (2a); or decrease the word spacing and fit the word on the first line (2b). This process continues with each new line producing more alternatives (3a–d). The algorithm eventually chooses the best outcome for the paragraph.

Don't justify the final line of a paragraph

The final line of a justified paragraph should always be aligned left. If not, you will almost always end up with horrendous gaps between the words. There may be occasions when you want to break this rule, such as setting a display paragraph in a perfect rectangle. By default, CSS will left-justify the final line of an element. CSS3 provides `text-align-last`, enabling you to fully justify the final line if you need to:

```
p { text-align-last: justify; }
```

Use knowingly and sparingly, and not for paragraphs within a block of continuous text.

Don't justify without hyphenating

Never set justified text without using hyphenation. You can, however, hyphenate without justifying text. Using hyphenation with left-aligned, right-aligned and centred text can improve readability and overall appearance by greatly reducing the raggedness of text.

Hyphenation has been around as long as justification. Look closer at Gutenberg's 600-year-old bible designs and you will see hyphenation is an intrinsic part of the justification process, just as it was for the scribes he was emulating.



Gutenberg's justified text was heavily hyphenated.

Hyphens in text read from a screen can sometimes seem incongruous, especially if the general design is dissimilar to books, newspapers or other print work where hyphens are commonplace. If hyphens jar with you as a designer, or – more importantly – if you find evidence that they jar with your readers, simply avoid justifying text and set your paragraphs ragged right, thereby reducing the need to hyphenate. Remember, though, that to justify text *without* hyphenating is to wilfully harm the reading experience.

Agglutinative languages will always need hyphenating. Consider arbeidsongeschikheidsverzekering (which means disability insurance in Dutch).

Manual hyphenation

In HTML, there are two types of hyphen: the ordinary hyphen and the *soft hyphen* (also known as a discretionary hyphen). Ordinary hyphens are always displayed by browsers, so you use them mostly to indicate composite words such as *fly-by-wire*. You can also use them for purposes of clarity; for example, the phrase *twenty four hour shifts* is ambiguous. Use a hyphen to clarify whether you mean *twenty four-hour shifts* or *twenty-four hour shifts*.

The soft hyphen enables you to tell the browser where a word could be broken across two lines. It will not be read out by screen reading software and will only be displayed when a word is actually being broken. Soft hyphens are therefore what you would use to hyphenate manually.

In **HTML** the soft hyphen's named character entity reference is `­` and its Unicode number is U+00AD. Given that automatic hyphenation is now possible in **CSS** (as we shall see), the main use for soft hyphens is for providing hyphenation hints within unusual or fictional words. For example you could set hyphenation break points for the made-up word *contrafibularity* as follows:

```
contra&shy;fibu&shy;lar&shy;ity
```

The various JavaScript-based hyphenation routines available online work by inserting soft hyphens into your text.

Word breaks without hyphens

It's worth noting that **HTML** provides an element that enables you to indicate where a word could be broken across lines, even when line-breaking rules would not normally apply at that point in the word. This element is `<wbr>` or *word break opportunity*. By definition, it does *not* introduce a hyphen at the line break point, but does still allow the word to render across multiple lines. It is particularly useful for splitting long URLs which would not normally break; for example:

```
<p>http://this<wbr>.is<wbr>.a<wbr>.really<wbr>.long<wbr>.  
example<wbr>.com/With<wbr>/deeper<wbr>/level<wbr>/  
pages<wbr>/deeper<wbr>/level<wbr>/pages<wbr>/deeper<wbr>/  
level<wbr>/pages<wbr>/deeper<wbr>/level<wbr>/pages<wbr>/  
deeper<wbr>/level<wbr>/pages</p>
```

Which could render as:

http://this.is.a.really.long.example.com/With/deeper/level/pages/deeper/level/pages/deeper/level/pages/deeper/level/pages/deeper/level/pages

Automatic hyphenation

Hyphenation is a complex subject. Hyphenation points are mainly based on syllables using a combination of etymology and phonology, but house styles also have differing rules on dividing words.

There are two steps required to turn on automatic hyphens. The first is to set the language of the text. This will tell the browser which *hyphenation dictionary* to use. Each language has different rules – even for the same words – and correct automatic hyphenation requires a hyphenation dictionary appropriate to the language of the text. In English we hyphenate *cab·ri·o·let*, but in French the hyphenation is *ca·brio·let*. If the browser does not know

the language of the text, the CSS guidelines say it is not required to hyphenate automatically – even if you turn on hyphenation in a style sheet.

The language of a webpage should ideally be sent as an HTTP header by the web server (speak to your friendly sysadmin about this). You can also set it using the HTML lang attribute:

```
<html lang="en">
```

Setting the language is best practice for all web pages regardless of whether you are hyphenating or not. Knowing the language of the text will help automatic translating tools, screen readers and other assistive software.

The lang="en" attribute uses an ISO language tag¹ to tell the browser that the text is in English. In this case the browser will choose its default English hyphenation dictionary, which will often mean hyphenating for American English. This will usually be fine, but there are some differences in spelling and pronunciation between American and British English (Canadian, Australian, and others also have their differences). For example, in Britain, *advertisement* is pronounced ad-VERT-is-ment, whereas in America it is typically pronounced ad-ver-TIZE-ment. The differently stressed syllables change where the ideal word break should be. Similarly, North American English has the word *aluminum*. Outside North America, the word is *aluminium*, which may not be present in an American English hyphenation dictionary, and thus might not be hyphenated as or when required. The solution is to add a region to the language:

```
<html lang="en-GB">
```

Indicate the language even for single foreign words or brief quotations. That way hyphenation should be applied appropriately (and the words interpreted correctly by other language-sensitive tools). Do this by adding lang attributes to containing elements in your markup:

```
<p>You'd say that in Spanish as <span lang="es">Cuatro  
cervezas, por favor</span>.</p>
```

Now you've set your language, you are ready to turn on automatic hyphenation in CSS. This couldn't be much easier:

```
hyphens: auto
```

¹ W3C advice on language tags in HTML (<http://w3c.org/82>).

The `hyphens` property has two other values. `hyphens:manual` means only hyphenate a word across a line break at manually inserted normal or soft hyphens. The third value `hyphens:none` means that words must not be broken across lines even if they contain normal or soft hyphens. Put more simply, `hyphens:manual` turns off automatic hyphenation, whereas `hyphens:none` turns off all hyphenation.

There are some widely accepted rules around which words should or shouldn't be hyphenated. Don't hyphenate words of one syllable, even though several are quite long: *strengths, wrought, grasped, screeched, stretched*. As a general rule don't hyphenate proper nouns (names of people, places and organisations) or fully capitalised abbreviations. That said, you can get away with hyphenating frequently occurring proper nouns after their first appearance, or proper nouns which appear as frequently as common nouns. Consider the sentence:

Irina Bokova is the Director-General of UNESCO.

You can mark this up as follows:

```
<p><span class="pn">Irina Bokova</span> is the <span class="pn">  
Director-General</span> of <abbr>UNESCO</abbr>.</p>
```

and apply the following CSS rule:

```
.pn, abbr { hyphens: manual; }
```

This will prevent 'Irina', 'Bokova' and 'UNESCO' from being hyphenated. It will allow 'Director-General' to be broken at the hyphen, but will not hyphenate the individual words.

Hyphenate judiciously and with care

The Knuth–Plass justification algorithm described earlier has an advanced hyphenation routine built in. This takes into account – and limits – the amount of hyphenation over an entire paragraph.

Hyphenation routines used in browsers are generally not as sophisticated as Knuth–Plass, but the CSS Text Module Level 4 has introduced the same kind of hyphenation controls provided in advanced layout software. These enable you to tweak hyphenation routines to your satisfaction.

Limit the number of consecutive hyphenated lines

For aesthetic reasons, limit the number of lines in a row that are hyphenated. Consecutively hyphenated lines, particularly three or more, are pejoratively called a *ladder*. The general rule of thumb for English is that two consecutive lines is the ideal maximum (in contrast, readers of German may well be faced with many ladders). By default, CSS sets no limit to the number of consecutive hyphens, but you can use the `hyphenate-limit-lines` property to specify a maximum:

```
hyphenate-limit-lines: 2
```

message, there lies the opportunity for the little added touch that goes far beyond the bare essentials of taste and into the realm of fantasy. For this reason the ornamentation of printing is at once the most charming and the most dangerous diversion the typographer can find: dangerous because all matters of decoration call upon the utmost discretion and sense of fitness for their effective use.

message, there lies the opportunity for the little added touch that goes far beyond the bare essentials of taste and into the realm of fantasy. For this reason the ornamentation of printing is at once the most charming and the most dangerous diversion the typographer can find: dangerous because all matters of decoration call upon the utmost discretion and sense of fitness for their effective use.

hyphenate-limit-lines applied to prevent a ladder.

You can remove the limit using the `no-limit` value:

```
hyphenate-limit-lines: no-limit
```

Limit the word length and the number of characters before and after a hyphen

If you hyphenate short words they can be harder to read. Likewise, you don't want too few characters left on a line before the hyphen, or pushed to the next line after the hyphen. A common rule of thumb is to only allow words at least six-letters long to be hyphenated, leaving at least three characters before the word break, and taking a minimum of two to the next line².

² The *Oxford Style Manual* recommends that three is the minimum number of letters after a hyphen at a line break, though exceptions can be made in very short measures for words ending -ad, -al, -an, -en, -fy, -ic and -or. Before the break, two is permissible in very short measures, depending on the syllabic nature of the word. Other style manuals have their own guidance.

You can set these limits with the `hyphenate-limit-chars` property. It takes three space-separated values. The first is the minimum character limit for a word to be hyphenated; the second is the minimum number of characters before the hyphenation break; and the last is the minimum characters after the hyphenation break. To set the aforementioned rule of thumb, with a six-character word limit, three characters before the hyphenation break and two after, use:

```
hyphenate-limit-chars: 6 3 2
```

grapher can find: dangerous because all matters of decoration call upon the utmost discretion and sense of fitness for their effective use.

grapher can find: dangerous because all matters of decoration call upon the utmost discretion and sense of fitness for their effective use.

hyphenate-limit-chars in action.

If you omit the last value, it is set to be the same as the second, so:

```
hyphenate-limit-chars: 6 3
```

is the same as:

```
hyphenate-limit-chars: 6 3 3
```

If you only include one value, the last two are set automatically by the browser, so:

```
hyphenate-limit-chars: 6
```

is the same as:

```
hyphenate-limit-chars: 6 auto auto
```

The default value for `hyphenate-limit-chars` is `auto` for all three settings. This means that the browser should pick the best settings based on the current language and layout. The CSS Text Module Level 4 suggests that browsers use `5 2 2` as their starting point, but browsers are free to vary that as their hyphenation algorithms see fit.

Legacy support for hyphenation limits

Hyphenation character limits were originally specified in CSS3 using two separate properties: `hyphenate-limit-before` and `hyphenate-limit-after`. In April 2011 these were merged into the `hyphenate-limit-chars` property described above. However, at the time of writing, Safari 10 only supports hyphenation character limits using the legacy properties. For a more backwards-compatible set of instructions, you could support hyphenation character limits in Safari 10 using:

```
p {
    -webkit-hyphenate-limit-before: 3;
    -webkit-hyphenate-limit-after: 2;
    hyphenate-limit-chars: 6 3 2;
}
```

Reduce hyphenation by setting a hyphenation zone

By default, hyphenation will occur as often as the browser can split a word across two lines (within the `hyphenate-limit-chars` and `hyphenate-limit-lines` values you set). This means an entire paragraph could be set with three hyphenated lines, followed by an unbroken word, followed by three hyphenated lines, followed by an unbroken word, followed by three hyphenated lines, and so on. The `hyphenate-limit-lines: 3` rule hasn't been broken, but the paragraph will still look a mess. Fortunately you can influence this by setting a *hyphenation zone*.

Consider a left-aligned paragraph. The right edge is ragged, which hyphenation can reduce. By default you will get the maximum amount of hyphenation and thus the maximum reduction to the rag. If you are prepared to tolerate a little more unevenness to the edge of the paragraph, you can reduce the amount of hyphenation. In effect you are allowing there to be a bit more space between the last word in the line and the edge of the paragraph.

By setting the maximum amount of space you'll allow between the last word of a line and the edge of the paragraph, you are setting the hyphenation zone. To do this you use the `hyphenate-limit-zone` property, which takes a percentage value (in terms of the width of the text box) or a length; for example:

```
hyphenate-limit-zone: 2em
```

The default value for `hyphenate-limit-zone` is zero; that is, minimum rag, maximum hyphenation.

In the embellishment or decoration of a message, there lies the opportunity for the little added touch that goes far beyond the bare essentials of taste and into the realm of fantasy. For this reason the ornamentation of printing is at once the most charming and the most dangerous diversion the typographer can find: dangerous because all makers of decoration call upon the utmost discretion and sense of fitness for their effective use.

In the embellishment or decoration of a message, there lies the opportunity for the little added touch that goes far beyond the bare essentials of taste and into the realm of fantasy. For this reason the ornamentation of printing is at once the most charming and the most dangerous diversion the typographer can find: dangerous because all makers of decoration call upon the utmost discretion and sense of fitness for their effective use.

In the embellishment or decoration of a message, there lies the opportunity for the little added touch that goes far beyond the bare essentials of taste and into the realm of fantasy. For this reason the ornamentation of printing is at once the most charming and the most dangerous diversion the typographer can find: dangerous because all makers of decoration call upon the utmost discretion and sense of fitness for their effective use.

In the embellishment or decoration of a message, there lies the opportunity for the little added touch that goes far beyond the bare essentials of taste and into the realm of fantasy. For this reason the ornamentation of printing is at once the most charming and the most dangerous diversion the typographer can find: dangerous because all makers of decoration call upon the utmost discretion and sense of fitness for their effective use.

Left: Regular hyphenation with no (zero) hyphenation zone. *Right:* Hyphenation with 2em hyphenation zone. Arrows indicate lines where hyphenation is allowed.

In the context of the flexible paragraphs we discussed in earlier chapters, it makes most sense to set your hyphenation zone as a percentage. Based on typical defaults in page layout software, 8% is a good start:

`hyphenate-limit-zone: 8%`

In terms of absolute dimensions, using a percentage means you would get a smaller hyphenation zone on smaller screens, leading to more hyphenation and less rag. Conversely on wider screens you would get a broader hyphenation zone, hence less hyphenation and more rag, which a wider measure would be better able to accommodate.

You can set a hyphenation zone regardless of whether you are aligning or justifying. In both circumstances, you reduce the number of hyphens by making the hyphenation zone wider; or reduce the raggedness of the text by making the hyphenation zone narrower.

Avoid leaving the stub of a hyphenated word as the last line of a paragraph

Unless you tell it otherwise, a browser will happily hyphenate the very last word of a paragraph such that the end of the broken word sits alone on the final line, a lonely orphan of an orphan. It is preferable to have a large gap at the end of the penultimate line than having half a word on the final line. You can achieve this by activating the `hyphenate-limit-last` property with a value of `always`.

`hyphenate-limit-last: always`

It's worth noting that while you should ensure only whole words form the final line of a paragraph, it's better still to have at least two words on the final line. Unfortunately, there is no way in CSS to automate that.

Don't centre long passages of text

Centred headings are fine, as are short introductory paragraphs. Stylistically, centred text can provide a classical feel: nice to look at, affording harmony, elegance and even authority. But long passages of centred text are tiring and more difficult to read, so avoid centring your main body of text.

Responsive paragraphs

With more and more web-enabled gadgets entering the hands of readers every day, web designers must aspire to make their designs flexible and adaptable to screens and devices of all shapes and sizes. This is what we call *responsive design*. At its heart, responsive design is not about finding the perfect design, but finding the best compromise.

Adapt your design to the reading context

Your reader could be using a small phone, a large phone, an old phone, a tablet, a square screen, a wide screen, a mini-laptop, a massive laptop, a modest desktop, a huge monitor, a television, a game console, an aircraft seat, or a cinematic projector. Your design will need to work regardless.

For your design to respond and adapt to the dimensions and capabilities of your reader's device, you need to know something about that device and have a mechanism by which you can tweak your design accordingly. It is precisely for this reason that *media queries* were developed in css3.

Introducing media queries

A media query enables you to not only target certain types of rendering (primarily either screen or print), but also to inspect the physical characteristics of a device. Media queries are rules used by browsers to determine whether a style sheet should be used or not. By way of example we can incorporate a media query into a linked style sheet's `media` attribute like this:

```
<link rel="stylesheet" media="screen and (min-width: 960px)"  
      href="wide.css" />
```

The media query contains two components: a media *type* of `screen`, and an *expression* enclosed within parentheses. The media query expression contains a media *feature* (`min-width`) followed by the target value (`960px`). In plain English, the media query is asking the device if it is a screen and, if so, whether its horizontal resolution is at least 960 px. If it is, then the browser will load `wide.css`; otherwise the link is ignored altogether, the style sheet won't be loaded and the styles within it won't be applied.

Many devices with small screens will initially try to render a web page by shrinking it down. In effect, they pretend to have bigger screens than they actually do, and then squeeze a large page into their small screen. This works adequately for web pages which are not responsive, but if you are using media queries you will need to tell the device that it should tell the truth about its screen size and initially not scale the page. To make these two requests of a device, add the following `<meta>` tag to the `<head>` of your page:

```
<meta name="viewport" content="width=device-width,  
initial-scale=1.0">
```

Media queries can also be used to selectively apply rules *within* a style sheet. To do this, use `@media` rules in your style sheet as follows:

```
p { font-size: 1rem; }  
  
@media screen and (min-width: 60em) {  
    p { font-size: 1.2rem; }  
}
```

In this example, paragraphs are sized at `1rem` unless the device's screen is `60em` or wider, in which case the media query overrides the preceding rule and paragraphs are sized at `1.2rem`. When you use a minimum width like this, it is usually called a *breakpoint*. Think of it as the point at which the design would break unless you applied additional styles.

Use ems rather than pixels to determine the screen width

Use ems to ensure your layout adaptations are based on typographic reasoning rather than being tied to specific devices. What the preceding media query is *not* saying is: on an Apple tablet do this, on a Samsung smartphone do that, on an Amazon e-reader do the other. Importantly, using ems takes into account situations when your reader adjusts the default font size of

their device. Defining screen sizes in ems means your rules are set not solely by the size of a screen, but also based on the effect a screen has on the type.

Adjust type size according to reading distance

The ideal size of your text doesn't just depend on your reader's default preference, but also on reading distance: the farther away the text, the smaller it becomes perceptually. To compensate for longer reading distances, you therefore need to make the text size bigger.

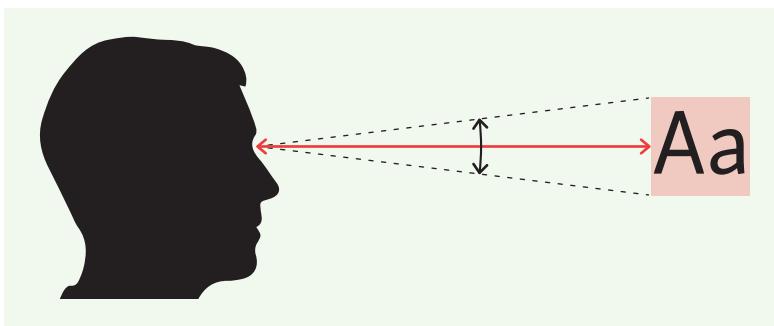


'These are small, but the ones out there are far away.' © Hat Trick Productions Ltd with permission.

Computers (that is, desktop screens and laptops) are generally read from farther away than books, newspapers and other printed matter. Typically, we hold books with bent arms such that the text is about 35 cm from our eyes. In contrast, a laptop or tablet screen – whether on a table or a knee – might well be 45 cm away, and a desktop is likely to be even farther, up to 60 cm. As each device is read from a different distance, the perceived text size differs, getting smaller the farther away the screen. You need to correct for this.

Calculate text sizes for different devices

A printed book, typically held 35 cm away from the eyes and with type set at 10 pt, gives a *perceived text size* of 30 arcminutes. An *arcminute* is a measure of the size of a perceived object, used by ergonomists. It is actually an angle (one sixtieth of a degree), which means text size expressed in arcminutes can take into account both type size and distance from the eye.



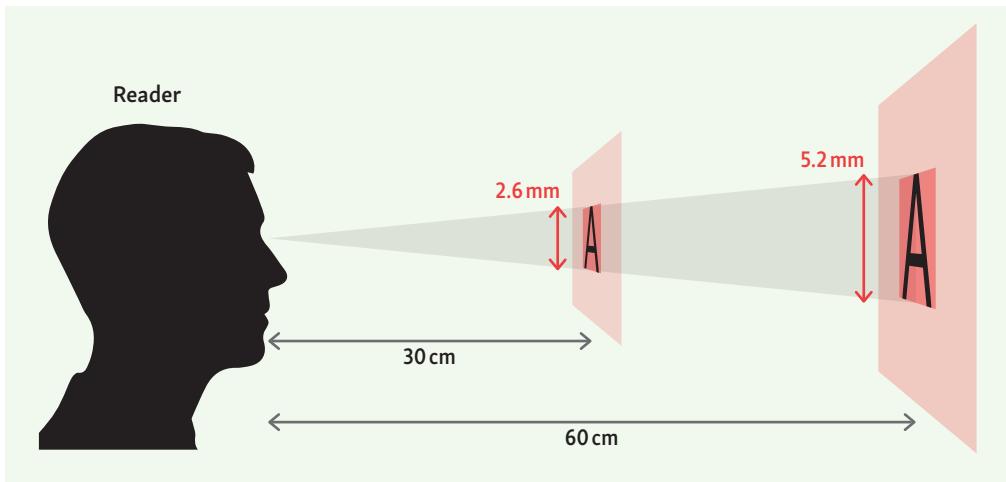
Text size expressed in arcminutes reveals the angle formed by the height of the text and the distance from the eye. *Image courtesy [SizeCalc.com](#)*

Studies show that when reading we hold smartphones slightly nearer than we do books, often at 30 cm¹ or closer. To achieve a perceived text size of 30 arcmins – the same comfortable size as a book – we can calculate² that text on a phone screen should be about 2.6 mm tall. On a smartphone this approximates to text set at 16 px, which nicely justifies the assertion we made in ‘[Designing paragraphs: text size](#)’ that you should use the default text size for continuous reading.

On that same basis, let’s consider screens viewed from 45 cm away. To achieve 30 arcmins at that distance, you would need text to be about 3.9 mm tall, which is equivalent to 18–19 px on a modern laptop or tablet screen. For desktop screens positioned 60 cm away from your reader, the text would need to be 5.2 mm tall, which is equivalent to 22 px on a large display.

¹ ‘[Font Size and Viewing Distance of Handheld Smart Phones](#)’ (<http://wbtyp.net/8>) by Yuliya Bababekova, Mark Rosenfield, Jennifer E. Hue, Rae R Huang in *Optometry & Vision Science* Vol 88, Issue 7, (2011).

² Using [Size Calc](#) (<http://wbtyp.net/74>).



Showing that text must become larger if the viewing distance increases.

All this implies a situation where we'd ideally want to increase the text size depending on how far away the screen is from our reader – yet we have no way of determining precisely how far away their screen actually is³. We must, therefore, turn to some approximations. First of all, we start with our default text size:

```
p { font-size: 1rem; }
```

We established that this is fine for smartphones, but for laptops and tablets on knees or desks we need to bump the text size up to 18px.

$$18\text{px} \text{ in rem} = 18 \div 16 = 1.125\text{rem}$$

Typical laptop and tablet widths start from 960 px, which is equivalent to 60 ems ($960 \div 16$). So we can compile our first media query:

```
@media screen and (min-width: 60em) {
  p { font-size: 1.125rem; }
}
```

³ Apart from some clever hacks such as Marko Dugonjić's [face detection experiment](http://wbtyp.net/72) (<http://wbtyp.net/72>).

That rule will match *any* screen at least 60em wide. To target large desktop screens we need an additional media query to increase the text size to 22px. If we posit that desktop screens start at 1,920px wide, equivalent to 120ems ($1,920 \div 16$), we can add this second media query:

```
@media screen and (min-width: 120em) {
  p { font-size: 1.1375rem; }
}
```

Of course, we've made many assumptions along the way here, not least of which is guessing how far away readers are from their screens. We've also had to use broad approximations to correlate screen size with device type and reading distance. This is the limited set of tools we web typographers are given, so we must make the most of it. We can at least rest easy in the knowledge that if our assumptions are wrong, our readers can adjust the text size to their satisfaction, which is more than you can say for our cousins designing for the printed word.

Keep your big text under control

Text that's too big also poses a problem. When type gets huge, your reader's eyes try to follow their usual pattern. But a font size that's too large takes up more horizontal space, and it interferes with your reader's horizontal flow of saccades and fixations (see '[How we read](#)'). Overly large text forces the reader to slow down and adjust how far they skip ahead as they read. Reading horizontally becomes cumbersome, and your reader will start to skip vertically down the left edge of the text, missing out on much of the content carefully crafted by the writer.

Text which ends up really big will have your reader physically turning their head as they scan the screen. Either that or they end up scooting their chair backwards into whatever (or whoever) is behind them.

Beware of the tablet problem

When it comes to reading, tablets are extremely versatile devices. Your readers can hold them in a landscape or portrait orientation, and they can be used in numerous contexts and environments. According to designer Craig Mod⁴, these typically break down into three reading distances: bed, knee and breakfast:

⁴ '[A Simpler Page](#)' (<http://wbtyp.net/24>) by Craig Mod in *A List Apart* Issue 321 (2011).

- Bed (close to face): your reader is lying on their front with their tablet propped up on a pillow.
- Knee (medium distance from face): your reader is sitting on their sofa, or perhaps a train or bus, with a tablet on their knee.
- Breakfast (far from face): Their tablet is propped up on a breakfast table behind their tea and toast (or coffee and croissant), reading hands-free.

There's no way to detect these different reading distances and dispositions, so there's nothing you can do about it, other than being aware of the possibilities, and testing with them.

Test your design in a responsive prototype

Typography for the web is really hard to design anywhere other than a browser. By testing and iterating your design in a browser-based prototype, you can solve the fundamentals before (or even instead of) jumping to other design tools, such as Photoshop, Illustrator or Sketch.

Improvise, calculate,
test, iterate and
improvise some more.

Your typography prototype should be a single web page that preferably consists of some real content. Gradually apply styles to the text to build up the design, just as we have done – and will continue to do – through the chapters in this book. As you add more styles, test them in different contexts: different devices, different screens, different reading distances, and so on, tweaking the design as you go. Use your typography prototype as the foundation for the rest of your design by starting with the fundamental ingredients – the content and the typography – and adding hierarchy, layout, colour and flourishes as you go.

Stepping back and squinting are no substitute for reading

A prototype gives you the perfect opportunity to properly test your design. Many designers test their work by positioning themselves away from the monitor to consider their work from afar, like an oil painter stepping back from a canvas to survey the bigger picture. Others too will squint their eyes, blurring their vision to get a sense of the overall colour, hierarchy and organisation of the page. There's no harm in this, and both techniques can help you quickly assess how well your design holds together. However, they are of no use whatsoever when it comes to determining the readability of your text. There's only one way to test that, and that's to actually read what you're setting.

This where real content comes into its own. If real content is not readily available from your client, customer or boss, do your best to demand, beg

or borrow it, emphasising its importance to your task. Ask for early drafts, sample content from other sources, or, failing that, simply find some text with which you are unfamiliar (a random news story, for example). Change your text each time you test a new iteration of your design – you'll never be able to judge readability with a passage of text you know back to front.

2 Typographic Detail

- Symbols, signs and accents
- Ligatures and abbreviations
- Hierarchy and scale
- Meaning and semantics
- Numerals and tables
- Tracking and kerning
- Headlines and impact
- Vertical rhythm
- Arrangement and composition

¶ *Typography is all about
the little details adding up to a
reading experience greater than
the sum of its parts.*

Symbols, signs and accents

Typography is about sweating the tiniest details. In isolation these are subtle, almost invisible and seemingly trivial, but taken as a whole the impression they give is greater than the sum of their parts. Give full typographic attention even to incidental details, and use the right character for the job.

There is more than one space

The space between characters which you insert using your space bar is a glyph unto itself: Unicode encodes it at U+0020. A space usually separates words by about 0.25em, but there is no standard size. Type designers will specify the width of a space in a given font just as they would design any other glyph.

Non-breaking spaces

A *non-breaking space* (also known as a hard space) keeps the two words it separates together on the same line. You can insert a non-breaking space into HTML using the entity . Use a non-breaking space in phrases such as ‘Page 2’ (Page 2) to prevent the 2 dropping to a subsequent line should the text wrap. The same applies to initials in names such as D. H. Lawrence, where you will want to keep the initials alongside the surname (note also there are little to no spaces between the initials themselves). Likewise, dates such as 14 March should be kept together with a non-breaking space.

Hairs and thins

Several other spaces exist. These are generally used when a regular space is too big but you still need to separate characters. A *thin space* is usually 1/6em wide and you can insert them in HTML using the entity  .

One particular use for thin spaces is separating nested quotation marks; for example: *Looking up, he said, ‘She mouthed “I love you” and then returned to his book* has the final quotation marks separated by a thin space.

Hair spaces – as the name implies – are very narrow, usually 1/24 em wide. They are used when you simply want to prevent adjacent characters touching, rather than to provide any significant separation. You can insert a hair space in HTML with   (and with   since HTML5).

For the aforementioned D. H. Lawrence, you can separate the initials using a thin space, although if you do so you might run the risk of the initials wrapping across lines. To prevent that, you should use a *narrow no-break space*, which is  ; in HTML. Also use narrow no-break spaces between numbers and their units.

D. H. Lawrence was a writer.

A hair space is 1/24 em wide.

If a word has an accent, use it

As web typographers, we often have very little control or even knowledge of the actual words, letters and symbols we are typesetting. Content is sucked out of a database and funnelled through our design filter onto the screens of our readers. Wherever possible we should still care, worry and take ownership of the finer details of that content. In particular, we should be aware of the scourge of the keyboard.

Computer keyboards betray their ancestry as teleprinters: electro-mechanical typewriters developed as communication tools. In the 1960s, teleprinters began to use a system known as ASCII (American Standard Code for Information Interchange), which was developed from the binary telegraphic codes of the time. ASCII includes 95 printable characters covering 26 upper- and lowercase letters, digits 0–9, and various punctuation marks including a space. These 95 characters are still what make up a modern computer keyboard in English-speaking nations. Each of the 26 letters of the alphabet are accessed by a single keypress with or without Shift held down. Any non-ASCII character is far more difficult to locate and type, which might not be so bad were it not for the fact that ASCII is not sufficient for English, let alone other languages.

Consider these common words: aïoli, crème fraîche, doppelgänger, El Niño, façade, fiancée, flambé, jalapeño, maître d’, Māori, pâté. Too often we see words like these without their requisite accents and diacritics. While accents in English usually disappear once a foreign word has become

thoroughly anglicised, it can lead to confusion. For example, we may wonder whether the balding man who likes chutney on his pate is referring to a lunchtime habit or a novel cure for his receding hairline. Which accents are used can be a matter of house style, but it is nonetheless our job to care about diacritics and other fine details should they leave a writer's keyboard and sneak past an editor.

Pay particular attention to accented names

Some names in common use include José, Chloë, Ciarán, Sîan, Beyoncé and Brontë. People deserve to have their names spelled correctly – it's a common courtesy, and a missing accent amounts to an incorrect spelling. Unfortunately, keyboard-induced typographic ethnocentrism doesn't leave much hope for our friends Dugonjić and Mrđenović.

Accented names are commonplace, even in countries where English is the first language. For example, here is a fictional football formation consisting of players from the English Premier League whose names contain all the accents and diacritics that have featured over the past twenty years:



Footballers featuring all the different accents since the formation of the English Premier League.

Since the Premier League began two decades ago, there have been over 200 players with an accent in their surname. Any publication covering the English Premier League should ensure its writers know and include the correct accents in every player's name. By implication, the fonts chosen to display the text should also include these accented characters – see 'Practical and pragmatic considerations' for more detail.

Use the appropriate punctuation marks

Orthotyography is the aspect of typography that defines the meaning and accepted usage of typographic signs and symbols, in particular punctuation. Treat punctuation as notation, not expression. In other words, don't draw attention to the symbols; they are there to provide additional meaning to the text, not to decorate it. (Unless you are setting text for attention-grabbing display purposes, in which case all such guidelines can be wilfully ignored.)

Don't use a hyphen instead of a dash

A hyphen (-) is directly accessible with one stroke of your keyboard. In typography it has four uses: joining words to indicate they have a combined meaning, such as *pick-me-up*, *government-mandated*, *non-negotiable*, *Berners-Lee*; indicating missing words shared by a series of compounds, as in *the short- and long-term effects*; indicating stuttering speech, like 'p-p-please'; and splitting words when breaking them across lines. For any other use you require a different symbol, usually a dash.

There are two common kinds of dash: the *en dash* and the *em dash*. Both dashes are longer than a hyphen, can sometimes be slightly thicker, and occasionally will sit lower. The em dash (—) is 1em long and the en dash (–) is 1en long, precisely half the length of an em dash.

En dashes

You can insert an en dash into HTML with the – entity. Use en dashes in phrases with numerical ranges such as: 4–5 minutes; 28 March–3 April; copyright 2005–2016; and 87–135 Brompton Road. There's usually no need to insert a space either side of the dash in these situations, although you could slip in a hair space if the dash appears to be touching the characters either side.

In British English, spaced en dashes – rather than em dashes – are generally used to set off phrases and parenthetical statements. In other words, if you'd rather not use parentheses (like this) then you could use

dashes – like this – instead. Using dashes tends to add emphasis to the text, whereas parentheses remove it, marking something tangential to the main point. Separate en dashes from phrases with a full space. When you would choose dashes over parentheses is entirely down to writing style and context.

Em dashes

In American English, em dashes—rather than en dashes—are generally used to set off phrases. Separate em dashes from phrases with hair spaces. You can insert an em dash into HTML using —.

Normal space with em dash	this and that — that and this
Thin space with em dash	this and that— that and this
Hair space with em dash	this and that— that and this
No space with em dash	this and this— that and this

Comparison of spaces around an em dash.

Other uses of em dashes might be mandated by house style. You might choose to use an em dash followed by a full space to indicate attribution after a quote:

One more attribute the modern typographer must have: the capacity for taking great pains with seemingly unimportant detail. To him, one typographical point must be as important as one inch, and he must harden his heart against the accusation of being too fussy.

— Hans P. Schmoller

You can also use em dashes to indicate lines in dialogue and remove the need for quotation marks:

— Aren't you a little short for a Stormtrooper?
— I'm Luke Skywalker. I'm here to rescue you!

The required presence of an em dash is often indicated by a double hyphen -- a relic from manual typewriters when only hyphens were available. Replace double hyphens with the appropriate dash.

In running text use a proper minus

Hyphens are very commonly used to mean minus. For programmers writing code, a hyphen has a special encoding which does indeed imply minus. However, for anyone needing a minus symbol in running text, a hyphen is the wrong character to use. Use a proper minus glyph; for example: $10-7=3$ is simple subtraction. The main advantage is that a proper minus symbol is designed to go between digits and will therefore sit at the appropriate height in the line. Minus symbols are also usually identical in length and weight to one of the strokes in an equals sign and thus look more consistent in an equation; or, rather, a hyphen can look wrong when placed in proximity to an equals: $10-7=3$. A hyphen is often shorter and sits higher in the line. In HTML you can use the − entity.

Use a division symbol rather than a slash

Like neutral quotation marks, use of a / slash symbol (otherwise known as a solidus or virgule) to denote *divided by* is another survivor from the typewriter keyboard. If you are not quoting code then use the proper division sign, an *obelus* ÷. It is neater, more likely to be understood by people not familiar with programming conventions, and less likely to be confused with alternative meanings such as ‘or’ should the context not be clear. A division symbol can be inserted with the ÷ entity.

Use proper and appropriate quotation marks and apostrophes

Other relics of typewriters are the neutral quotation marks (‘ ’) inherited from a time when only one key was available to indicate a quote. Always use proper quotation marks (“ ”) in running text. These are otherwise known as smart quotes, curly quotes, or typographers’ quotes (although they belong to everyone, not just designers).

Proper quotation marks give clearer indications of the beginning and end of a quote. Being directional they usher along the flow of the text, whereas neutral primes appear as tiny hindering wedges through a passage.

When nesting quotes, the predominant British style is to have single quotation marks surrounding double quotation marks: ‘The “stunt typographer” learns the fickleness of rich men who hate to read.’ American style prefers double quotes first, with single quotes nested: “The ‘stunt typographer’ learns the fickleness of rich men who hate to read.” In HTML you can use “ and ” entities for “”. The equivalent entities for ‘’ are ‘ and ’.

Across Europe there are even greater differences in quotation mark notation¹, with quotes commonly indicated with guillemets (« » and « ») and baseline commas („ „ and , ,) in varying combinations. Use the marks appropriate to the text.

Don't use italic brackets

Italic parentheses and brackets, even when properly redrawn, are sloped versions of their upright selves. This gives them an awkward unbalanced stance which can unsettle the text they enclose. Use upright brackets to surround italicised passages and within italicised text. You will need to mark up text accurately and, in some cases, with additional tags:

Their adopted language [*<i>lingua franca</i>*] is French.

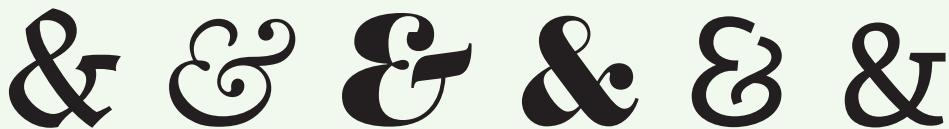
Their adopted language [*lingua franca*] is French.

```
<cite>Nobody <span class="paren">(</span>save the other  
craftsmen<span class="paren">)</span> will appreciate half  
your skill.</cite>
```

Nobody (save the other craftsmen) will appreciate half your skill.

Don't overuse or over-elaborate ampersands

An ampersand is a contraction of two letters: an *e* and a *t*, spelling *et*, Latin for *and*. Over the centuries, ampersands have taken many different forms, the more familiar design being more akin to a treble clef than an *et* ligature: &. Type designers often like to let their hair down a bit when it comes to designing ampersands. Some fonts retain the historical origin of the contraction, others have evolved significantly:



Ampersands come in many forms.

¹ International variation in quotation marks (<http://wbtyp.net/66>).

Use ampersands sparingly. Always use *and* in running text unless you particularly want to effect a historical setting. Today's readers are far less used to ampersands, and their presence may impede and interrupt progress through the text.

Some style guides say you should only ever use an ampersand when one appears in a name, such as Selfridges & Co. Other styles give leeway to use ampersands in headings and titles provided they are used consistently. In *The Elements of Typographic Style*, Robert Bringhurst writes:

In heads and titles, use the best available ampersand.

This advice seems to have been taken to heart by many designers and blown out of all proportion. Bringhurst is not saying to replace plain ampersands with elaborate ones. He is not asking us to use the *fanciest* ampersand available. Unless your goal is decoration or embellishment, there is nothing wrong with a nice plain ampersand in a title. There is certainly no excuse for sticking a Baskerville italic ampersand in the middle of a Helvetica subheading.

also probably no coincidence that among first uses of symbols we have on record appear to be either records of (or how-to instructions on) hunting.

Recognition & Reading

What has all this to do with the life of modern man, and especially with reading? Well, most of us may have left the woods to live in towns and cities, but...

An italic Baskerville ampersand does not belong in a Helvetica heading.

Bringhurst's point is that where you are using an ampersand, particularly in a heading, be aware that some ampersands are a bit ugly or might not be in keeping with the spirit of your text. Your first port of call in this instance is to turn to the italic ampersand as an alternative, as these are often more attractive than in the roman (regular upright) font.

Forgive & Forget

Forgive & Forget

You may prefer Palatino's flowing italic ampersand to its chiselled roman counterpart.

Ampersands are extremely useful where space is limited. Use them in tables, charts, captions, or anywhere character count is at a premium. You must use the entity & in HTML – you cannot insert ampersands on their own as they have a reserved meaning (for forming character entities).

Other punctuation

Use a proper ellipsis (...) instead of three full stops. You can insert this in HTML using a … entity.

Use a proper multiplication sign instead of an x when specifying dimensions. An x looks particularly wrong when using a serif typeface. Thus: Country folk drive 4×4 vehicles; A4 paper is 210×297 mm. Use × to insert a multiplication sign into HTML.

Primes look very much like neutral quotes, but are different characters with specific meanings. Primes are often slightly sloped, whereas neutral quotes are necessarily straight. Always use primes where the text signifies feet and inches, and minutes and seconds. Thus: He was 6' 4" and full of muscle; Brighton is located at 50°50'35"N 0°07'53"W. Always use a proper degree symbol (°) rather than a raised or superscript o. You can insert a single prime into HTML using ′, a double prime using ″ (note the capitalisation), and degrees through °.

Use UTF-8 in preference to entities

Before HTML 4.01 was standardised in 1999, web pages were only allowed to be written using characters from a set defined by ISO-8859-1. This character set allowed for 191 printable characters, comprising ASCII plus the so-called *Latin-1 supplement* of western European accents and some additional symbols. Any glyphs outside that range – including almost all the quotation marks, dashes and other signs mentioned so far – needed an entity such as – to display the glyph. I have included the entity

codes for most of the symbols I have mentioned so far; however, they are no longer strictly necessary. With HTML 4.01 came the ability to use a character encoding technology called UTF-8. Enable this by including a `<meta>` tag:

```
<meta charset="utf-8">
```

This means you can use any character defined by Unicode (of which there are over one hundred thousand) directly in a web page without the need for typing out an entity code. There are four exceptions, which must all use entities as they have a special meaning in HTML:

<	<
>	>
"	"
&	&

It is far from obvious how to input, for example, an em dash directly into your HTML. Rather like a secret move in a computer game, you can input a special character through a sequence of keystrokes. These vary wildly between operating systems, as shown on the following page.

Automate character substitution whenever possible

We established earlier that as a web typographer you are unlikely to be typesetting the actual words of a given page, the implication being that you will rarely have the opportunity to get your hands dirty and correct hyphens with dashes, swap in proper quotation marks, and so on. However, all is not lost. Dozens of pieces of software are available to plug in to content management systems, which will automatically do the conversions for you.

Symbol	Windows	Mac	iOS	HTML
Hair space	alt + 200A	none	none	 
Thin space	alt + 2009	none	none	 
- En dash	alt + 0150	↖ + -	123 → hold -	–
— Em dash	alt + 0151	⇧ + ↖ + -	123 → hold -	—
- Minus	none	none	☺ → !?#	−
× Multiply	alt + 0215	none	☺ → !?#	×
÷ Divide	alt + 0247	↖ + /	☺ → !?#	÷
‘ Left single quote	alt + 0145	↖ +]	123 → hold '	‘
’ Right single quote	alt + 0146	⇧ + ↖ +]	123 → hold '	‘
“ Left double quote	alt + 0147	↖ + [123 → hold "	“
” Right double quote	alt + 0148	⇧ + ↖ + [123 → hold "	”
& Ampersand	⇧ + 7	⇧ + 7	123 → &	&
… Ellipsis	alt + 0133	↖ + ;	123 → hold .	…
‘ Single prime	none	none	none	′
” Double prime	none	none	none	″
° Degree	alt + 0176	⇧ + ↖ + 8	123 → hold 0	°
• Middle dot	alt + 0183	⇧ + ↖ + 9	123 → hold -	·
• Bullet	alt + 0149	↖ + 8	123 → hold -	•
© Copyright	alt + 0169	↖ + g	☺ → !?#	©
® Registered trademark	alt + 0174	↖ + r	☺ → !?#	®
™ Trademark	alt + 0153	↖ + 2	☺ → !?#	™

Ligatures and abbreviations

Just as there are different forms of punctuation, so there are alternative versions of letters and character combinations designed for readability and usability, and aesthetic and historical purposes.

Use standard ligatures for improved legibility

A ligature is a single glyph formed by combining two or more characters. Some ligatures occur as accepted (if increasingly outdated) forms of spelling, such as the *æ* in *encyclopædia* and the *œ* in *œuvre*. More frequently, ligatures are used to prevent unsightly and awkward clashes between adjacent letters. For example, the *tittle* (dot) on a lowercase *i* can sometimes clash with the hood of a preceding *f*. You can prevent this collision by replacing the *f* and the *i* with an *fi* ligature which harmoniously joins the two letter-forms together.



Plantin's *fi* ligature prevents a collision.

The clash of letters is generally more unsightly than adversely affecting legibility, but replacing the letters with a ligature improves the visual flow of the text and contributes to a smoother experience for your reader. The *fi* glyph is an example of a *standard* or *common ligature*. In letterpress days, this glyph would have been cast as a single sort.



Letterpress ligatures (mirrored for clarity). © Emma Charleston

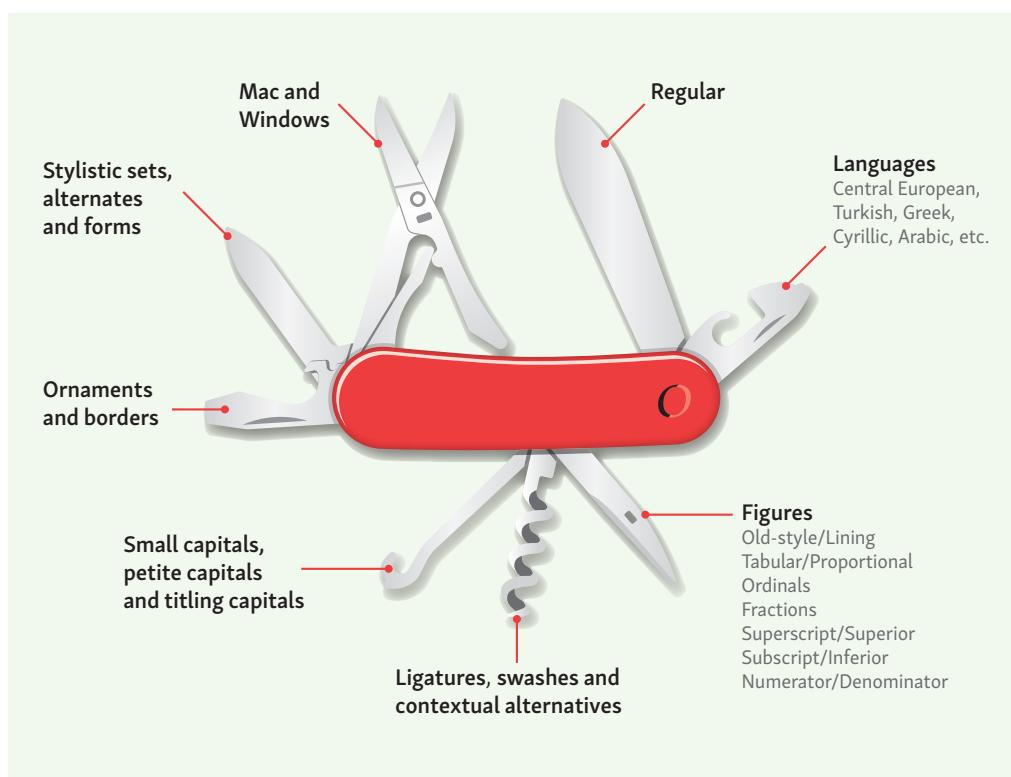
There is no particular specification to denote which ligatures are ‘common’. It depends on which character pairs need replacing in any given typeface design. Typically, ligatures are required where an *f* is involved, and this set is most often included:

fight office flight piffle fjord

Introducing OpenType

In earlier days of digital typesetting, typographers would have swapped out the individual letters one by one and replaced them with a ligature glyph. Early font formats were severely limited in the number of characters they could contain in a single file: PostScript Type 1, for example, is limited to 256 characters per file. This meant that glyphs such as ligatures would often need to be accessed from a separate font file containing only alternate characters, and rarely in any way that was standardised from one font to the next. This complicated the designer’s job considerably. That all changed in 1996 with the introduction of *OpenType* technology.

OpenType is a cross-platform font file format developed jointly by Adobe and Microsoft. Unlike earlier file formats, OpenType fonts can contain many thousands of characters – as many as 65,535. If a legacy font file format is a single paring knife, with its simple set of upper- and lowercase letters, numbers and basic punctuation, then OpenType is a Swiss Army knife of multiple blades and widgets, providing not just the basic characters, but those for multiple languages, and assortments of alternative glyphs too – all in one file.



OpenType is the Swiss Army knife of font formats.

OpenType is more than just a way to store lots of glyphs in one file; it is smart too. OpenType includes programmable automatic substitutions, a simple one being ligature substitution. This means that an *f* followed by an *i* can automatically be replaced by an *fi* glyph without any manual intervention (beyond turning on that setting in OpenType-savvy type-setting software).

By coincidence, OpenType was introduced in the same year as css (1996), but it took until a 2011 draft of css3 for OpenType features to be made available to web designers. It took a further five years for OpenType to gain universal support across modern web browsers. It is now a welcome and accepted tool in our box.

OpenType as implemented in css is an excellent example of *progressive enhancement*. Because OpenType replaces regular characters with specially designed ones, if OpenType is not supported then the regular characters are displayed. Readability and aesthetics may be impaired, but the content remains.

How to use OpenType ligatures in CSS

In all typesetting software, including web browsers, common ligatures are usually turned on by default. Most fonts – particularly free or cheap ones – don't contain ligatures, but any professional font that needs them will have ligatures included. You can control ligatures in css through the `font-variant-ligatures` property. To ensure that common ligatures are turned on through all your text, use:

```
body { font-variant-ligatures: common-ligatures; }
```

To turn off all common ligatures:

```
body { font-variant-ligatures: no-common-ligatures; }
```

Turning off common ligatures is not recommended – css3 requires that they should be enabled by default. However, OpenType features do cause a small performance hit so this could be important under circumstances where processor speeds are severely limited or at an absolute premium. If you need to turn off common ligatures, use a typeface which does not require ligatures in the first place. Good examples include most sans serif fonts and many typefaces designed especially for lower-resolution screens such as Georgia.

Hide legacy OpenType support from modern browsers

The correct way to toggle ligatures is with the `font-variant-ligatures` property. But when browsers first started supporting OpenType, they only supported the low-level property `font-feature-settings`, which was exclusively intended as a way of selecting OpenType features that are not otherwise accessible in css. Browser makers used this property as a

shortcut while they were perfecting OpenType support. Unfortunately, some took many years to replace this temporary solution. (At the time of writing, Microsoft still hasn't rectified this.)

To let modern browsers use the correct `font-variant-` properties, while ensuring only legacy browsers use `font-feature-settings` instead, you can use an `@supports` rule to check if the browser supports `font-variant-position`. `@supports` is a really simple, powerful way for you to provide styles to browsers that either do or don't support a given set of properties or values.

```
body {
  font-feature-settings: "liga" 1;
}

@supports (font-variant-ligatures: common-ligatures) {
  body {
    font-feature-settings: normal;
    font-variant-ligatures: common-ligatures;
  }
}
```

The code snippet shown above turns on common ligatures using `font-feature-settings` with "liga" 1. Then the `@supports` rule asks if the browser supports `font-variant-ligatures:common-ligatures` (you must always specify a property *and* a value). If that's the case, common ligatures are turned on with the `font-variant-ligatures` property and `font-feature-settings` is reset to normal.

If you're thinking the "liga" 1 value doesn't look very CSS-like, you'd be right. The seemingly arcane nature of the syntax reflects the property's intended use as only a low-level control. The 1 part is easy: it turns the feature on; use 0 to turn it off. The `liga` is a case-sensitive OpenType *feature tag*¹. A feature tag is one of the blades on the metaphorical OpenType Swiss Army knife. The feature provides instructions to the browser about how to use certain glyphs in a font; in this case, how to replace regular characters with standard ligatures. The `font-feature-settings` property takes a comma-separated list of feature tags.

Always use `font-variant-` and its related subproperties (more on those later). Because `font-feature-settings` is a low-level property it will

¹ List of OpenType features (<http://wbtyp.net/71>).

override itself. For example, the following code sets numbers throughout the document to be set as old-style numerals, with elements having a class of `dlig` given discretionary ligatures:

```
body { font-feature-settings: "onum" 1; }

.dlig { font-feature-settings: "dlig" 1; }
```

However, all inherited `font-feature-settings` values are overridden when the property is reapplied. In the example above, you might expect elements with a class of `dlig` to also have old-style numerals, but because `onum` is omitted from the second declaration, old-style numerals are turned off again. To fix this you could repeat the `onum` value in the class rule:

```
.dlig { font-feature-settings: "onum" 1, "dlig" 1; }
```

Use alternative ligatures judiciously

There are many other ligatures besides standard ones. They exist for historical and stylistic purposes, the latter known as *discretionary ligatures*. A typical discretionary ligature might join a single-storey letter with an adjacent ascender using a dainty loop, harking back to handwriting, and giving the impression of small child holding hands with a parent.



ch ck ct fb ffy fh fr gi ih
im ip sh si sky sp st LA OO

Discretionary ligatures in Esta Pro.

The clue to applying discretionary ligatures is in the name: use them with *discretion*. A Victorian reader would have found most of these ligatures quite normal, but for today's reader they are unusual and serve only to interrupt the reading experience when used in running text. Treat discretionary ligatures as purely decorative: avoid using them unless the text you are designing requires a specific historical setting or deliberate sense of pretentiousness (or grandeur, at a push).

That said, historical and discretionary ligatures can look attractive in display text or large heading settings. To turn on both discretionary and historical ligatures, use spaces to separate the property values:

```
font-variant-ligatures: discretionary-ligatures  
historical-ligatures
```

For legacy browsers requiring `font-feature-settings`, use the `dlig` and `hist` OpenType feature tags:

```
font-feature-settings: "dlig" 1, "hist" 1
```

There is no standard identifying which ligatures are deemed common, which are defined as historical, and which are discretionary. It is down to the type designer to make these distinctions should they choose to design ligatures for their typeface at all.

Use real small caps

English and other Latin-based alphabets, as well as Greek and Cyrillic, use a *bicameral* script: the letters have both majuscule and minuscule forms. Put another way, we have both uppercase and lowercase letters. The term *uppercase* comes from the common arrangement of type cases used to hold the movable type for letterpress printing; traditionally, capital letters are kept in a separate case above the minuscule sorts.



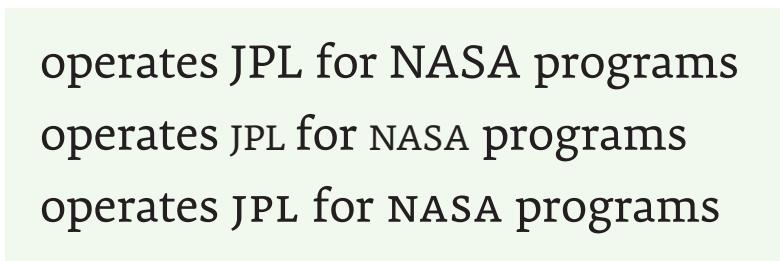
Upper and lower type cases containing moveable type. © Carole Mullen

While we use a bicameral script, there are more than two forms of each letter available to typographers. In addition to the familiar lower- and uppercase letters, we also have *small caps*. These are, as the name implies, small capital letters, about the same height as the x-height of the lowercase letters.

Small caps are most often used for abbreviations (chiefly initialisms and acronyms), subheadings or run-in text at the beginning of chapters and sections. You can also use them as an alternative to bold or italics in some situations, such as providing **LIGHT EMPHASIS** without changing weight or slope.

Online etiquette has it that writing IN CAPITALS is akin to SHOUTING and is thus undesirable – unless shouting is your aim. The same convention can be applied to all text. If you want to shout, by all means SHOUT, but if you want to write strings of capitals like NASA and UNESCO without calling unwarranted attention to their presence, you can turn to SMALL CAPS.

True small caps are not simply capital letters made smaller; they are drawn differently by type designers. Small caps are usually squatter and bolder than equivalently resized uppercase letterforms, to ensure they sit well next to lowercase letters. If you were to simply shrink down a regular capital letter, the result would look skinny, spindly and generally out of place.



operates JPL for NASA programs
operates JPL for NASA programs
operates JPL for NASA programs

Top: Regular uppercase. *Middle:* Uppercase reduced. *Bottom:* True small caps.

A font with well-designed small caps will not just include a full alphabet along with accents, but also specially designed numerals and punctuation to sit perfectly together. Compare regular caps and numerals with properly designed small caps characters: THIRTY (30)! with THIRTY (30)! To turn on small caps, use the `font-variant-caps` property:

```
abbr.smallcaps { font-variant-caps: small-caps; }
```

Following convention in digital typography, the `small-caps` value will only replace lowercase letters with small caps. It's more likely, and a better progressive enhancement, that you will want to replace uppercase letters with small caps. To do this, use the `all-small-caps` value, which will turn both lowercase and uppercase letters into small caps:

```
abbr.smallcaps { font-variant-caps: all-small-caps; }
```

The equivalent for legacy browsers is to reference the `small-caps` (`smcp`) and `caps-to-small-caps` (`c2sc`) OpenType feature tags:

```
font-feature-settings: "smcp" 1, "c2sc" 1
```

Sometimes fonts only have a `smcp` feature specified. This means that you won't be able to turn full caps into proper small caps without transforming them into lowercase letters first:

```
text-transform: lowercase
```

A word of warning: when you use `font-variant-caps` and you specify `small-caps` or `all-small-caps`, but small caps glyphs are not available for a given font, browsers will simulate a small-caps font by transforming the text into regular capitals and reducing the font size, thus creating the ugly effect illustrated earlier.

You don't always have to use small caps for acronyms and abbreviations. It is a matter of style, and you could quite happily use full caps if you prefer. As much as it can be important that abbreviations don't shout and distract, it is equally important that small caps seem natural and fitting within the text. In particular, avoid using small caps within Title Case Headings: they'll stand out like a sore thumb – use full caps instead.

Hierarchy and scale

The structure of a website is communicated through a typographic system. A good system makes documents scannable and easy to grasp.

Establish the structure up front

First, you'll need to understand the hierarchy and structure for yourself. What are the aspects of the content? How many levels of headings are required? Are there notes and captions? Is there any small print?

Don't rely on size alone for differentiation

The primary role of your typographic system is to provide differentiation to denote hierarchy. Major headings should immediately look different from subheadings; emphasised text should stand out from surrounding prose; quotations should be distinct from original copy; notes and small print should be visually subordinate to the main text.

There are many long-established methods of implying hierarchy. Text size, certainly, but it's not the only one. Consider also: weight, style (*italics* and **CAPITALISATION**), typeface, colour, spacing and proximity. The most important things don't always have to be the largest; they just need to be further distinguished from other elements. In other words, they need more contrast. Whatever you choose, just change one attribute at a time. Too much variation between elements of the content will lead to a messy, unintelligible system and a distracting experience overall.

Always size type using a scale

The simplest expression of a type scale is a single size used throughout. This was the approach used by the very earliest typographers and printers, who set books in a single font – one typeface at one size – and used ornamentation and spacing to indicate chapters and other structures. Their approach would have been mandated by the expense required to cut and found metal

type at that time. But you're likely to need more than one font size to effectively convey hierarchy and semantics. Consistency is the key in making meaning apparent. To achieve consistency, use a scale and stick to it.

A type scale provides a predefined set of type sizes. By design, this limits your choices and forces rigour and consistency into your typesetting. In these days of digital type, we're very used to having an infinite choice of type size, right down to fractions of a pixel. In the days of moveable type you would have had no option but to use a scale, as offered by the sizes of metal (and wooden) sorts in your type drawers.

During the sixteenth century, European typographers began to standardise the set of sizes at which those sorts were available. Over four hundred years later, that scale is still with us in the form of presets in the word-processing and layout software we use today. It remains a bedrock of modern typography, providing a balanced and harmonious progression of font sizes.



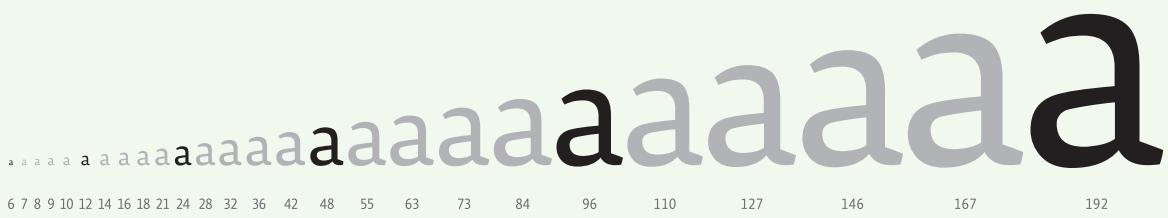
The classic typographic scale is often compared to a musical scale, and with good reason. The sizes 6, 12, 24 and 48 are all present in the scale; there is a precise doubling in magnitude at various points as the text size is increased. Consequently, a recurring *ratio* within the classic typographic scale is 2:1. That same ratio can be found in the harmonics manifested in music. Each C note on a piano, for example, vibrates at twice the frequency of the C one octave lower.

A musical scale is defined by a sequence of notes between the octaves. The mathematical relationships between the notes of a scale are fundamentally what gives us musical harmony (and discord), and just as these relationships are pleasing (or not) to our ears, so the harmonics of visual relationships can be pleasing or jarring to our eyes.

Music occurs not just in space but in time, and it is often the progression from one harmony to another that we find pleasing, rather than one particular harmonic effect in itself. Typography also occurs in time: text is primarily read rather than looked at. As in music, progressions in typography – where one text size leads to another – can be agreeable or disconcerting.

In music, the distance between any two notes (minor third, perfect fifth, octave, major ninth, and so on) is known as an interval. In typography, the distance between type sizes is also described as an interval, but in this case *interval* is defined as the number of type sizes between one size and its double. For example, in the classic typographic scale we can see that the interval is five. Between 12 and 24 there are five sizes: 12, 14, 16, 18 and 21. We can confirm this by looking at the next interval down, which also has five sizes: 6, 7, 8, 9 and 10. You may have noticed this interval also contains a size of 11, which is an anomaly, as we shall see.

Knowing the ratio and interval provides us with enough information to mathematically describe the classic typographic scale, enabling us to recalculate and extend the scale accordingly:

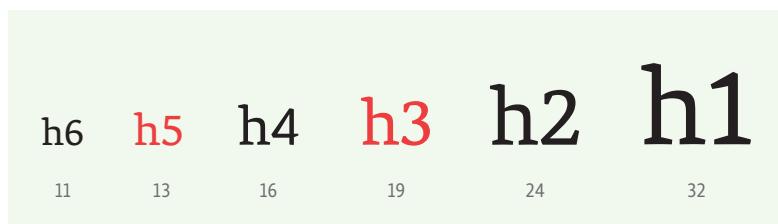


Comparing our calculated sizes to the values arrived at 400 years ago, it's clear the intuition of the Renaissance typographers was extremely close, with just a few deviations:



First there is the additional 11pt, then there are four lost ‘notes’ – sizes which would be perfectly harmonic within the scale. Furthermore, two of the notes are flat: the 60pt and 72pt sizes are off slightly. These two font sizes appear to be rounded down to the nearest multiple of the pica (1 pica equals 12pt), so this may have been a practicality required for the tooling and manufacture of early printing processes.

The scale all web browsers use as a default for headings and other elements closely matches the classic typographic scale. There are two slight deviations from both the classic and calculated scale. Given a base text size of 16px, `<h5>` and `<small>` elements are both rendered at 13px, and `<h3>` elements are sized at 19px. Browsers size HTML elements relatively, so the discrepancies may be down to a lack of precision in their internal style sheets.



Unlike the limitations of physical type, modern methods of digital typesetting – and CSS itself – enable type sizes to fall outside the notes of the classic scale. This freedom has resulted in a typographic free-for-all, allowing designers to pick sizes which may or may not be related to one another. This is akin to composing a discordant piece of music: clashing notes, clashing type. If typographic dissonance is what you’re after, that’s fine. If, however, the text you are setting will benefit from harmony and concord – as is almost always the case – stick to the notes in a scale. Even then, limit the number of different sizes you use – three is a good starting point to keep things simple for your reader.

Designs using a great many font sizes, however consistently, may convey the semantics of the text comprehensively, but will nonetheless appear confusing and complicated. To be readily made sense of, your typesetting should strike a single chord, not play out an entire melody.

Choose your smallest size first

Broadly speaking there are three kinds of text, each of which can take on a different magnitude:

1. **Reference (small).** Reference text includes small print, notes and specialist purposes such as bibliographies, dictionaries and data presentation. Reference text does not have to be set small, but in print it often is for economic reasons. This kind of text is not read in the immersive manner of running copy, so even where printing costs are not an issue, smaller text can be accommodated.
2. **Reading (medium).** Text set for immersive reading. This usually includes headings which may need to be sized differently to set them off from the body text.
3. **Display (large).** Display text is there to make an impact, evoke an emotional response and set a tone. It is intended to be *looked at* before it is read, and is thus set larger.

When using a type scale, identify your smallest size first and then go bigger. This will help you ensure the smallest text is still comfortably legible.

It is possible to start with your body text size, but always identify the smallest text next. Fail to do so and you may end up inside a trap many designers have fallen into: you choose undersized body text, then uncover a requirement for subordinate text which, when set small enough to be distinguished from the body text, ends up illegible.

Use alternative modular scales to suit the content

The classic typographic scale has been used for centuries. If set correctly, designs using only sizes found in this scale will appear pleasing to the eye. It could be the only scale you ever need. Other scales, though, are both possible and potentially useful.

The classic typographic sequence is a *modular scale*; a prearranged set of harmonious proportions which have a meaningful foundation. Any single modular scale can be described using the following equation, where f_i is the text size at the i^{th} point in the scale:

$$f_i = f_0 r^{\frac{n}{i}}$$

f_i = type size, f_0 = base size, r = ratio, i = interval, n = point on scale

Applying the modular scale equation to the classic typographic sequence, the meaningful foundation – or *base size* – is 12, reflecting the 12 pt body text often used in print, and the 12 pt found in a pica. As mentioned earlier, the interval i is 5, and the ratio r settled on 400 years ago is 2:1, or 2 in decimal terms. The mathematical sequence we derived previously was calculated by plugging these numbers into the equation as follows:

$$f_i = 12 \times 2^{\frac{n}{5}}$$

f_i = type size, n = point on scale

A pica is an imperial unit of length.
6 picas = 1 inch.

Equations like this, while relatively simple, are not expected to be used directly (although they can be). Instead, they serve to demonstrate that typography is a craft rather than an art, and that there are readily repeatable patterns we can rely on to support and enable creative solutions to design problems. In the case of a modular scale, you can create a scale tailored to your needs, and stick to the scale to ensure consistency in your design.

By changing the base size, interval and, in particular, the ratio you can create different modular scales customised to the needs of the text and the setting. Online calculators such as Tim Brown's [Modular Scale](http://wbtyp.net/69) (<http://wbtyp.net/69>) and this author's [Simple Scale](http://wbtyp.net/73) (<http://wbtyp.net/73>) can do the maths for you and make it quick to experiment.

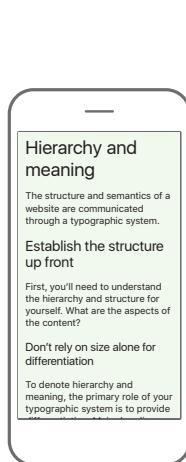
To define your own modular scale, start by specifying the base size: simply use the size of your body text; for example, 16. For the ratio you can pick a culturally or historically relevant number to provide you with a visual harmony not found in layouts that use arbitrary or easily divisible numbers. Many people might immediately reach for the *golden ratio* φ (1.618 as a decimal) – that so-called divine number supposedly appearing repeatedly in nature and all manner of classical art and architecture. The golden ratio would indeed give you a pleasant harmonious type scale, although it is unlikely to have any relationship to the content or proportions of your reader's screen.

In the world around us and online, musical ratios are at least as commonplace as the golden ratio. In particular, they can be found in the aspect ratios of many screens, and they appear in the dimensions of much digital imagery. For example, the aspect ratio of most digital SLR cameras is 3:2, a perfect fifth (1.5 as a decimal). A perfect fourth is 4:3 (1.333), and is the aspect ratio of lots of other digital photography, particularly compact

cameras and many phones. If your content includes significant amounts of photography, you can tie your text to the images by using a related scale.

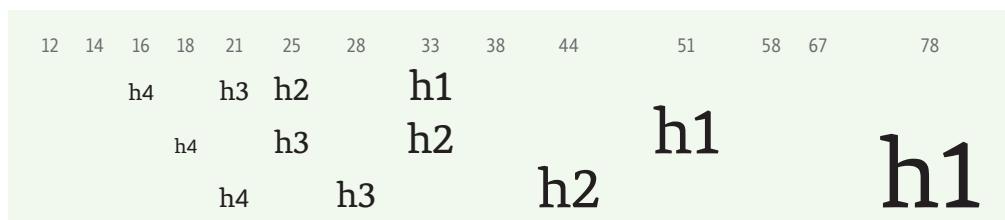
Adjust the text size and scale for different screens

Choose text sizes of similar magnitudes to the browser defaults and you will have a set of proportions eminently suitable for small screens. Apply those same sizes to larger screens, where bigger text can be accommodated, and you will diminish the contrast necessary for your reader to determine the hierarchy at a glance.



Default text sizes work well on smartphones but lack contrast and impact on large screens.

It's not strictly necessary to choose a different *scale* for larger screens, particularly if you are relating your choice of text scale to the aspect ratio of other content elements. Simply choose larger sizes from within the scale. For example, using a single scale based on a perfect fourth 4:3 ratio, you could pick different, incrementally bigger sizes for small, medium and larger screens:



Using media queries to target larger screens with bigger text, your CSS would be:

```

h1 { font-size: 2.0625rem; /* 33px */ }
h2 { font-size: 1.5625rem; /* 25px */ }
h3 { font-size: 1.3125rem; /* 21px */ }
h4 { font-size: 1rem; /* 16px */ }
p { font-size: 1rem; /* 16px */ }

@media screen and (min-width: 60em) {
  h1 { font-size: 3.1875rem; /* 51px */ }
  h2 { font-size: 2.0625rem; /* 33px */ }
  h3 { font-size: 1.5625rem; /* 25px */ }
  h4 { font-size: 1.125rem; /* 18px */ }
  p { font-size: 1.125rem /* 18px */ }
}

@media screen and (min-width: 120em) {
  h1 { font-size: 4.875rem; /* 78px */ }
  h2 { font-size: 2.75rem; /* 44px */ }
  h3 { font-size: 1.75rem; /* 28px */ }
  h4 { font-size: 1.3125rem; /* 21px */ }
  p { font-size: 1.3125rem; /* 21px */ }
}

```

Hierarchy and meaning

The structure and semantics of a website are communicated through a typographic system. Your reader should be able to pick up this undocumented visual language intuitively. A good system makes documents scannable, implies meaning and provides consistency.

Establish the structure up front

First, you'll need to understand the hierarchy and structure for yourself. What are the aspects of the content? How many levels of headings are required? Are there notes and captions? Does the text contain tables, reference material or asides? How much variation in emphasis is there? Are there quotes and extracts?

Don't rely on size alone for differentiation

To denote hierarchy and meaning, the primary role of your typographic system is to provide differentiation. Major headings should immediately look different from subheadings; emphasised text should stand out from surrounding prose; quotations should be distinct from original copy; notes and small print should be visually subordinate to the main text.

There are many long-established methods of implying hierarchy and meaning. Text size, certainly, but it's not the only one. Consider also: weight, style (italics and CAPITALISATION), typeface, colour, spacing and proximity. The most important things don't always have to be the largest; they just need to be further distinguished from other elements. In other words, they need more contrast. Whatever you choose, just change one attribute at a time. Too much variation between elements of the content will lead to a messy, unintelligible.

Exaggerating the type sizes for larger screen sizes gives a more effective sense of proportion.

Cinemascopic displays are particularly wide and shallow, with an aspect ratio of 21:9.

Take screen height into account

Exponentially increasing text size based on viewport width will work well in most situations, but it relies on one key assumption: that viewport height increases proportionally with the width. A very wide viewport might, on the face of it, demand significantly large headings. However, if the window height is very shallow, perhaps due to a configuration by your reader, or an uncommon device shape, then the text could be disproportionately large or even cropped.

In the spirit of responsive design, account for this possibility and ensure that larger text is only displayed in viewports which are both wider *and* taller. Include a height-based feature in your media queries; for example:

```
@media screen and (min-width: 60em) and (min-height: 30em)  
{ ... }
```

```
@media screen and (min-width: 120em) and (min-height: 60em)  
{ ... }
```

Next we'll learn how to add semantics to your system and use your scales to imply meaning.

Meaning and semantics

Your typographic system provides an undocumented visual language which your reader should be able to pick up intuitively. A good system confers meaning, and provides differentiation and consistency.

Establish the semantics up front

If you have control over the underlying `HTML` then you will be in the enviable position of conveying the meaning and structural semantics under the hood of the website, as well as the visual interpretation of that on top. If the `HTML` is coded well it will give you a good guide to the elements which you'll need to provide styles for. Either way, the onus is on you to comprehend the text in advance; or if the text doesn't yet exist, to get as good an understanding as possible of what textual components are likely to come your way. Does the text contain tables, reference material or asides? How much variation in emphasis is there? Are there quotes and extracts? Without that knowledge up front you are designing completely in the dark.

Headings

Headlines and headings draw your reader into the page and to sections within the text. They help your reader perceive the structure of the text, navigate through it and focus in on passages of interest.

A heading must look like a heading at first glance, and you can achieve this with more than size alone. Spacing, alignment and visual changes including small caps, italics, bolds and condensed styles provide a diverse palette for your type. Consider side headings and inline headings as possible placements. You can also use a different, contrasting typeface;

for example, balancing serifed headings against sans serif body text (see ‘[Combining typefaces](#)’). Change one parameter at a time – your role is to set headings in a form that contributes to the style of the whole.

In this digital age of limitless font sizes, it is a revealing and challenging assignment to set an entire page of content at one size using a single typeface. It’s an exercise worth undertaking to fully appreciate the possibilities available to you as a designer. First, try to get the page to make sense simply through use of spacing. Then add further differences such as a style and weight only where additional clarification is required.

HIERARCHY AND MEANING

The structure and semantics of a website are communicated through a typographic system. Your reader should be able to pick up this undocumented visual language intuitively. A good system makes documents scannable,

Establish the structure and semantics up front

First, you’ll need to understand the hierarchy and structure for yourself. What are the aspects of the content? How many levels of headings are required? Are there notes and captions? Does the text contain tables, reference material or

Don’t rely on size alone for differentiation

To denote hierarchy and meaning, the primary role of your typographic system is to provide differentiation. Major headings should immediately look different from subheadings; emphasised text should stand out from

CHOOSE YOUR SMALLEST SIZE FIRST

Broadly speaking there are three kinds of text, each of which can take on a different magnitude, small, medium and large.

Reference. Reference text includes small print, notes and specialist purposes such as bibliographies, dictionaries and data presentation.

It is possible to convey structure and meaning using just a single font size.

When writing your CSS it is good practice to style all [HTML](#) heading levels (1–6) in order to cover future occurrences. If you expect there to only ever be four levels of heading, simply style your h5 and h6 headings the same as your h4. You won’t get differentiation this way, but you will at least achieve consistency should those lesser headings unexpectedly make their way into the content.

Tighten up leading in headings

Larger text requires relatively less leading. A line height of 1.5 may be fine for body text but large headings will be too widely spaced, to the point that a two-line heading may look more like two separate headings. When typesetting large text, always test your design using enough text to make the line wrap to properly check the spacing between lines.

**Massive Supernova Visible
Millions of Light Years from Earth**

**Massive Supernova Visible
Millions of Light Years from Earth**

Heading with line height of 1.5 versus 1.

Avoid clash (unless it's a deliberate choice)

You may find you need to set your line-height to a value of 1 (called *setting solid*) for your headings to be spaced correctly. In doing so you run the risk of *clash*. Clash happens when descenders touch or overlap ascenders, as though a *g* has gone fishing and caught an *f*. Clash causes difficulty or tension for your reader, and can adversely affect legibility – not something you'd want with text as important as a heading. Always check what happens when a tightly set long heading wraps.

Mark the opening of each passage of text

Draw your reader into the prose by indicating each opening and resumption of the text. If the very beginning of the material is preceded by site furniture and metadata such as navigation, advertising, title, byline, date and tags, then the opening text will need to be particularly attention-grabbing.

Your first tactic should be to turn to white space: separate the opening text from what precedes it by at least a couple of lines. To further accentuate entry into the text, you could set the first line of the opening paragraph in bold or, more commonly and gracefully, in small caps. This is called a *run-in* and can be achieved simply using the `::first-line` pseudo-element. To bold the first line:

```
p::first-line { font-weight: bold; }
```

You can be more specific in your code and target the opening paragraph of an article using the `:first-of-type` pseudo-class. To set the first line of the first paragraph of an article in small caps:

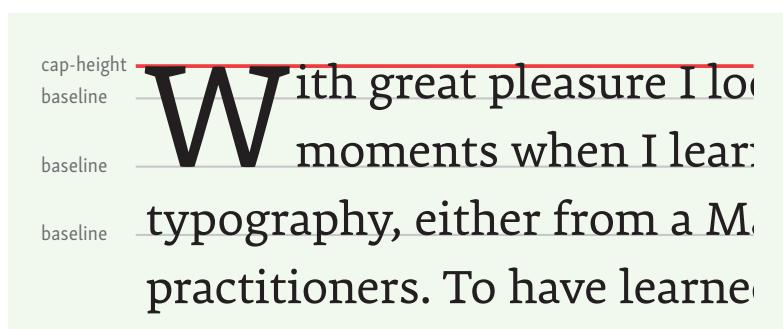
```
article p:first-of-type::first-line {
    font-feature-settings: "smcp" 1; /* for legacy browsers */
}
@supports (font-variant-caps: small-caps) {
    article p:first-of-type::first-line {
        font-feature-settings: normal;
        font-variant-caps: small-caps;
    }
}
```

Implement drop caps accurately or not at all

A classical approach to drawing a reader's eye to the beginning of the text is the decorative initial, or *drop capital*: a larger than usual letter at the start of a paragraph. When implemented well, drop caps are effective, dignified devices which go largely unnoticed by the reader yet still serve their purpose. Poorly implemented drop caps – incorrectly sized and poorly aligned – are a distracting foible, as if an orator were stumbling over their opening words.

If you're using a web font purely for drop caps, it makes sense to subset the font to include capital letters only.

Accurately executed drop caps share a baseline with the adjoining text and are tall enough to align with the cap height of the first line of text. Drop caps do not have to be set in the same typeface as the adjoining text, and frequently aren't – think of the beautiful illuminated manuscripts in which this device originated. The main requirement, for aesthetic and readability reasons, is that they adhere to the top-and-tail alignment.



A two-line drop cap correctly aligning to baselines and cap-heights.

Drop caps have two siblings: *sunken caps* and *raised caps*. Raised caps are large initial letters which align to the baseline of the first line. Sunken caps sit between raised and drop caps, in that they are large letters which are taller than the first line but are located on a lower baseline.

W

ith great pleasure I look back to all the mc
when I learned something new in typography, either :

Three-line raised cap.

W

ith great pleasure I look back to all the mc
when I learned something new in typogra

from a Master or from fellow practitioners. To have le

Three-line sunken cap.

W

ith great pleasure I look back to all the mc
when I learned something new in typogra

from a Master or from fellow practitioner.

learned about disciplined design from my Swiss fello

learned about the white space from my American fell

Three-line drop cap.

For a long time, accurately positioned drop caps have been extremely difficult to achieve on the web, particularly once considerations of responsiveness are brought into the mix. They have tended to be individually art-directed, making them expensive to create and difficult to maintain, leaving a typographical feature that has been used for over 400 years a relatively rare sight on the web. That is no longer the case. The advent of the css Inline Layout Module Level 3 brought with it a new property, `initial-letter`, which puts all the onus of alignment on browsers – indeed the alignment in the prior examples is fine, but the kerning of the W still leaves a lot to be desired.

The `initial-letter` property takes a value made up of two numbers. The first is the height of the drop cap, expressed in lines. The second is the number of lines down on which the initial cap sits. A three-line drop cap is achieved by selecting the first letter and setting `initial-letter` to 3 (you can omit the second number for drop caps). Adapting our earlier example:

```
article p:first-of-type::first-letter {  
    initial-letter: 3;  
}
```

A sunken initial three lines high, and sticking up one line can be specified thus:

```
article p:first-of-type::first-letter {  
    initial-letter: 3 2;  
}
```

And a three-line raised cap is achieved with:

```
article p:first-of-type::first-letter {  
    initial-letter: 3 1;  
}
```

♪ A variation on drop caps is to open the passage with a typographic symbol such as a pilcrow ¶, manicule ↗ or fleuron ♀. You can do this by adding the symbol to the opening paragraph as generated content and applying `initial-letter` as described above. For example, you can open a passage with a dropped fleuron as follows:

```
p:first-of-type::before {  
    content: "♀ ";  
}  
p:first-of-type::first-letter {  
    initial-letter: 2;  
}
```

A convention of drop caps is that any opening punctuation, such as a quotation mark, is treated as if it is part of the opening letter, meaning that the opening punctuation is also enlarged and dropped. This behaviour is also mandated by CSS and will almost always be the desirable situation.

The exception is when you only want to drop the punctuation but leave the first letter intact. The fleuron in the previous example is not considered punctuation; however, a pilcrow is, which means the pilcrow *and* the opening letter are both dropped even if they are separated by a space. To prevent this happening, you can persuade a browser that the pilcrow is not part of the first word by inserting a zero-width space (encoded in CSS as \200B).

```
p:first-of-type::before {
  content: "¶ \200B";
}
```

If you have multiple sections or passages of text within a page, you can use a fleuron to separate them. This can provide a gentle, stylish and elegant solution, which works in settings calling for a classical touch.

Use an `<hr>` between your sections and replace its standard styling with a background image:

```
hr {
  height: 1em;
  border: 0;
  background: url(fleuron.svg) center no-repeat;
  background-size: contain;
}
```

In this code we have made the `<hr>` 1em tall, set the background image as an SVG, and set it to cover the element. This way the fleuron will scale nicely when your reader changes their text settings.



A standfirst should stand first

A *standfirst* is a short opening paragraph often used in journalism to set the scene or summarise the subsequent article. When prose opens with a standfirst, make sure it is distinguishable from the rest of the text. A useful technique, especially if the page contains many distractions and much supplementary matter, is to set the standfirst paragraph in a larger size.

If you do this, make sure the standfirst is differentiated from any sub-headings of the same size – consider italicising the standfirst or choosing a different weight for the headings.

Demarcate paragraphs appropriately

Choosing the means to delineate paragraph breaks is both an aesthetic and a functional decision. We use paragraphs to mark individual units of thought, and you help your readers by giving them a pause between those thoughts. The boundary between one paragraph and the next should therefore be clearly and simply marked.

There is no right or wrong way to indicate a new paragraph; it's a choice largely based on personal preference or house style. As with all things typographic, smooth the way for your reader by sticking to established and well-recognised patterns. Demarcate sequential paragraphs with an indent, or insert a space between them. But never both.

A single line space is the most common way to delimit paragraphs on the web today, and it's the default style in all browsers. However, the indent is still the most prevalent paragraph marker in printed books and publications. Before the 19th century, the insertion of a full line of white space between paragraphs would have been considered decadent. Printing was an expensive process, and paper was particularly costly (not to mention even more pricey substrates such as vellum). ‘Printing’ on a screen effectively removes cost as a factor, so the *reading experience* is the only currency by which web typography is measured.

Marking paragraphs with indents benefits readers of essays, books, news articles and other longer-form writing. They keep the eye scanning down the text without interruption, and the brain engaged with the words (assuming the words themselves are engaging). Inserting white space between paragraphs impedes reading and induces your reader to skip to the next paragraph.

For the same reason, the delineation provided by a line break can be advantageous to readers of reference material, where typically each paragraph is more self-contained. The line breaks can thereby help separate ideas and enable purposeful scanning of the text.

Compared with fiction and newspapers, writing on the web tends to be more *chunked*, with each paragraph holding a complete thought. This style of composition has the skimming reader in mind and thus lends itself more to line breaks between paragraphs.

Indent paragraphs at least 1em

Use at least 1em for the paragraph indent – any less lacks clarity. Better still, use an indent equivalent to the line height. This will create a neat square of white space between paragraphs.

tion, so that it be a deep shimmering crimson.
 You have two goblets before you. One is ornate, in the most exquisite patterns. The other is a simple, plain chalice.

A paragraph indent equivalent to the line height, forming a square.

The following CSS indents the first line of any paragraph that directly follows another by a distance equivalent to the line height:

```
p {
  font-size: 1rem;
  line-height: 1.5;
  margin-bottom: 0;
}

p + p {
  text-indent: 1.5em;
  margin-top: 0;
}
```

Never indent the first paragraph after a title, heading, table, list, image or other interruption.

Separate paragraphs by no more than 1em

If you use a blank line to separate paragraphs, make it no bigger than the type size. The purpose is not to tire the eyes of your reader, merely to aid scanning. Half the line height should be sufficient in most circumstances and will be a suitable compromise should the text reward linear reading.

Don't use stylised paragraph breaks without good reason

Paragraphs can also run inline using a typographic symbol such as a pilcrow ¶ to demarcate the boundaries. These are unconventional in modern typesetting and thus prone to inducing confusion and fatigue in your reader.

Nonetheless, this setting can be attractive and even historically appropriate in short runs of text and under the right circumstances.

Imagine that you have before you a flagon of wine. You may choose your own favourite vintage for this imaginary demonstration, so that it be a deep shimmering crimson in colour. ¶ You have two goblets before you. One is of solid gold, wrought in the most exquisite patterns. The other is of crystal-clear glass, thin as a bubble, and as transparent. ¶ Pour and drink; and according to your choice of goblet, I shall know whether or not you are a connoisseur of wine. ¶ For if you have no feelings about wine one way or the other, you will want the sensation of drinking the stuff out of a vessel that may have cost thousands of pounds; but if you are a member of that vanishing tribe, the amateurs of fine vintages, you will choose the crystal, because everything about it is calculated to reveal rather than hide the beautiful thing which it was meant to contain. ¶ Bear with me in this long-winded and fragrant metaphor; for you will find that almost all the virtues of the perfect wine-glass have a parallel in typography. ¶ There is the long, thin stem that obviates fingerprints on the bowl. Why? Because no cloud must come between your eyes and the fiery heart of the liquid. Are not the margins on book pages similarly meant to obviate the necessity of fingering the type-page? ¶ Again: the glass is colourless or at the most only faintly tinged in the bowl, because the connoisseur judges wine partly by its colour and is impatient of anything that alters it. ¶ There are a thousand mannerisms in typography that are as impudent and arbitrary as putting port in tumblers of red or green glass!

Running paragraphs separated by pilcrows can be pretty, if tiring and unconventional.

The effect can be achieved through generated content and the `display:run-in` property (although `display:inline` works reasonably well too, for older browsers).

```
p {
  display: inline;
  display: run-in;
  margin: 0;
}

p + p:before {
  content: "¶ ";
}
```

Not all block quotes need to look like pull-quotes

A *pull-quote* is a typographic device used to *pull a quote* out of the text to serve a specific purpose: that of an eye-catching snippet to add visual interest and scannability. Pull-quotes tend to be set much larger than, and in juxtaposition to, the body text. They are considered in more detail in '[Headlines and impact](#)'.

A block quote is different: it is simply a passage of cited text. In fiction this might be a letter quoted within the story; for journalists this could be an extract from a legal document. In other non-fiction, such as a tutorial or blog post, it might be an excerpt from a specification or related blog post. Whichever the case, the block quote will not warrant the attention-seeking presentation of a pull-quote.

Set block quotes apart from the rest of the text, but don't give them undue emphasis. Set them no bigger than the body text and inset from the left margin – as a guide, use the same indent as your paragraphs. There is no need to inset from the right margin. Consider one additional variation to set the block quote apart. Typically you would either italicise or set slightly smaller, but not both. If you are also crediting the source of the quote, do so by ranging the citation right, set it off using an em dash and consider setting it smaller than the body text. For example:

There is nothing simple or dull in achieving the transparent page. Vulgar ostentation is twice as easy as discipline. When you realise that ugly typography never effaces itself; you will be able to capture beauty as the wise men capture happiness by aiming at something else. The 'stunt typographer' learns the fickleness of rich men who hate to read. Not for them are long breaths held over serif and kern. Nobody (save the other craftsmen) will appreciate half your skill. But you may spend endless years of happy experiment in devising that crystalline goblet which is worthy to hold the vintage of the human mind.

— from *The Crystal Goblet* by Beatrice Warde, 1932

Hang punctuation

When quotation marks appear at the beginning of a line they nudge the start of the text inwards, causing an unsightly gap and unevenness within the paragraph, which can make reading a slightly jerky experience. If you open a block quote with a quotation mark, align the first *letter* in the quote with the first letter of all the other lines. To do so, *hang* the opening quote

mark by moving it into the margin or indent. The closing punctuation can simply follow on from the text without special positioning.

“One more attribute the modern typographer must have: the capacity for taking great pains with seemingly unimportant detail. To him, one typographical point must be as important as one inch, and he must harden his heart against the accusation of being too fussy.”

— Hans P. Schmoller, in ‘Book Design Today’, *Printing Review*, Spring 1951

Hang punctuation by giving the opening quote mark a negative left margin to move it from its original position, pulling the following text with it. Rather than inserting the quote mark directly into the text, implying you’d need an additional `` to make it selectable, insert the marks using generated content, keeping your HTML clean and simple:

```
<blockquote class="quoted">
  <p>One more attribute the modern typographer must have:
    the capacity for taking great pains with seemingly
    unimportant detail. To him, one typographical point
    must be as important as one inch, and he must harden his
    heart against the accusation of being too fussy.</p>
  <footer>— Hans P. Schmoller, in ‘Book Design Today’,
    <cite>Printing Review</cite>, Spring 1951</footer>
</blockquote>
```

Use the `css quotes` property to specify which quotation marks you would like to use, and in which order. Then combine the `::before` and `::after` pseudo-elements with `content:open-quote` and `content:close-quote` properties to prepend and append quotation marks to the text.

Positioning the opening quote mark using a negative margin means you retain the correct spacing between the punctuation and the letters. A drawback is that the required distance of manoeuvre will vary slightly from font to font. Specifying the margin using `ch` rather than `em` will at least tie the deviation to the width of the font rather than the independent `em` square. In the following example, the negative margin shift of `-0.83ch` was determined by experiment for double quotes set in the font used in the prior block quote. There’s no fixed rule for how far to indent the quotes, so just use your eye (it can be a good idea to bump the text right up while you’re doing so).

```
.quoted p {  
    quotes: "“””””;  
}  
.quoted p::before {  
    content: open-quote;  
    margin-left: -0.83ch;  
}  
.quoted p::after {  
    content: close-quote;  
}
```

Treat lists as text to be read

Lists should be treated with the same reverence you would show any other text. If a list is inserted within a passage of text, treat it as a continuation and integral part of that text. Offset the list from the preceding paragraph or heading with a modest space – half a line would normally suffice.

- If the list items are invariably short, needing only one or two lines, remove the spacing between each item. The bullet or numbering will provide sufficient demarcation.
 - For longer list items, apply additional separation.
 - If separating list items, use half a line or the same spacing by which you delineate paragraphs. Any more and your reader will need to hop, skip and jump from one list item to another, rather than continuing to read smoothly.

For bulleted lists, indent each list item the equivalent distance of a line height. This will allow the bullet to sit neatly in a white square. Numbered lists which reach into double figures and beyond will require more space. Allocate this space in multiples of the line height.

- 1 You can save further space by removing the punctuation from numbering.
 - 2 This is particularly true if you hang the numbering in the margin, providing the optimum expression of continual reading.

Removing the punctuation from a list means overriding the browser's inbuilt styles. This is possible but not trivial, and gives us the opportunity to get to know *CSS counters*. First, remove the browser's defaults and create a counter by using `counter-reset`, and giving the counter a name (a variable which can be used to continue counting in further lists):

```
ol {  
    padding-left: 0; margin-left: 0; list-style: none;  
    counter-reset: mylist;  
}  
}
```

Then use a `:before` pseudo-element and generated content to increment and place a counter (that is, an incrementing number) before the list item, and move it into the margin.

```
ol li::before {  
    counter-increment: mylist;  
    content: counter(mylist);  
    margin-left: -2em; margin-right: 1em;  
}  
}
```

In list-led text, such as bibliographies, you can use outdents instead. These can be achieved by combining a negative `text-indent` with a positive left margin.

```
li {  
    margin-left: 1.5em;  
    text-indent: -1.5em;  
    list-style: none;  
}  
}
```

Dyson, M.C. (2004) “How physical text layout affects reading from screen”,

Behaviour and Information Technology, 23(6).

Bababekova, Yuliya (2011) “Font Size and Viewing Distance of Handheld Smart Phones”,

Optometry & Vision Science, Vol 88, Issue 7.

Use outdents in a bibliography to help your reader scan the references by author.

Emphasis and other semantics

Avoid using underlines for emphasis

As with neutral quotes and two spaces after a full stop, underlining is a relic of the typewriter age. With a typewriter, the single weight, monospaced font and mechanical operation meant typing over with underscores was the only realistic method of highlighting. This is doubly problematic on

the web because, conventionally, anything underlined is initially deemed to be a link. Your readers may therefore be confused when they click on some underlined text and nothing happens.

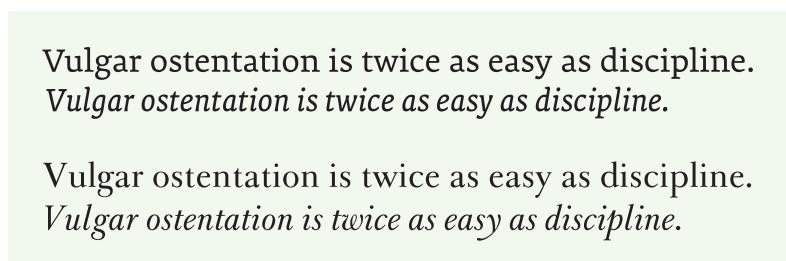
Avoid faux bolds and italics

No type family of professional quality will incorporate bold or italic styles which are simply computer-generated versions of their normal siblings. A true bold or italic, as opposed to a synthesised *faux* bold or italic, will have been either hand-drawn or at least skilfully modified to optically correct the distortion and imbalance that results from mathematical fakery. The following example compares a faux italic with its properly drawn counterpart. The differences are particularly noticeable on curved letters such as the O.



Clockwise (from top-left): Proxima Nova normal; faux italic; real italic; real italic superimposed on faux.

The Proxima Nova italic illustrated is not strictly a true italic. It should be described as a *sloped roman* or *oblique*. Obliques are slanted, albeit optically corrected versions of their corresponding *roman* (the normal upright font). Italic fonts are significantly different to their roman styles, and are generally more compact and cursive in nature – they are derived from handwriting rather than stone carving.



Vulgar ostentation is twice as easy as discipline.
Vulgar ostentation is twice as easy as discipline.

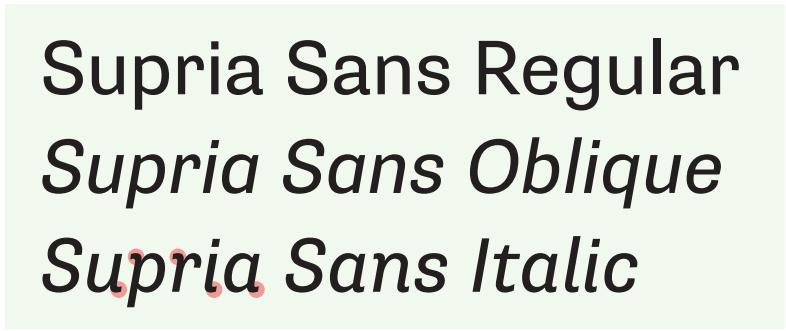
Vulgar ostentation is twice as easy as discipline.
Vulgar ostentation is twice as easy as discipline.

Italic forms are far more common in serifed typefaces than sans serifs. When italics appear in a sans serif family, they tend to be obliques with a few subtly different letterforms; in particular, a double-storey *a* or *g* in the roman may have a single storey in the italic.

Use the CSS `font-style` property to select one of three styles: `italic`, `oblique` and `normal`. Use `normal` (the default) to select a face that is neither italic nor oblique. Using `italic` will select a font that is internally labelled as an italic face, or, if no italic is available, it will select an oblique face if there is one. Similarly, `oblique` selects a font that is labelled as an oblique face, or an italic face if one is available and an oblique is not. In both cases, if neither an italic nor an oblique is available, a faux italic will be synthesised by the browser by artificially sloping the glyphs of the regular face.

The fallback behaviour of `font-style` means that `italic` and `oblique` can usually be used interchangeably. This is particularly true as most fonts have their obliques labelled as italic. However, there is a small handful of fonts which have both italics and obliques. One such example is Supria Sans. In that case, selecting `oblique` and `italic` will result in a different font being rendered.

```
p {  
    font-family: "Supria Sans", sans-serif;  
    font-style: normal;  
}  
  
p.italic { font-style: italic; }  
  
p.oblique { font-style: oblique; }
```



Supria Sans Regular
Supria Sans Oblique
Supria Sans Italic

Supria Sans normal, oblique and italic.

Never use faux bolds and italics, even as a means of saving bandwidth. Explained in more detail in ‘[Using web fonts](#)’, you could show a faux bold first so long as it is replaced by the real thing once it has loaded. The same can be applied to obliques, but be aware that italics can be problematic in this approach as they tend to take far less horizontal space, resulting in a potentially jarring reflow of the page as the italic font loads.

You can control whether a browser synthesises fonts using the `font-synthesis` property. By default its value is `weight style` meaning that both bold and oblique/italic fonts are synthesised. To turn all synthesis off, set `font-synthesis` to `none`. The following rule will turn off oblique and italic synthesis in top-level headings, while allowing faux bolds:

```
h1 { font-synthesis: style; }
```

As objectionable as synthesised bolds and italics are, they are a useful fallback should the real fonts fail to load or render. If you turn synthesis off and the italic or bold font does fail to load, your reader will not be able to see where that styling was being applied and may therefore miss out on important structure or meaning in the text. You should only turn off synthesis if not having synthesis is vital, or the styling is purely decorative.

Weights

Use the CSS `font-weight` property to select bold fonts. At its simplest, use `font-weight:bold` to select a bold, and `font-weight:normal` to deselect it. As with italics and obliques, if no true bold font is available, browsers will synthesise one, with all its attendant imperfections.



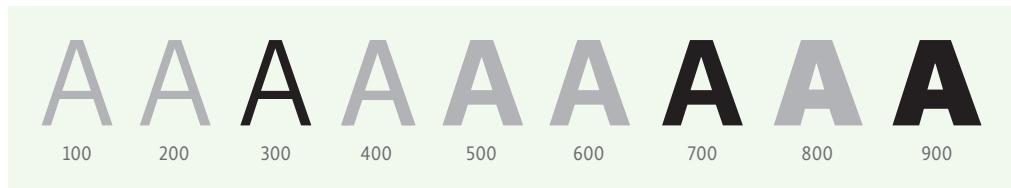
The drawn Premiéra bold is thicker and more balanced than the synthesised faux.

There is much more to font weights than simple regular and bold. Some type families will include an entire spectrum of weights from ultra light through to extra black, and many more will have multiple fonts at key intervals along that spectrum. You select a specific weight using `font-weight`'s numeric scale which has nine values running from 100 to 900 in steps of 100. By definition, regular weights are 400 on this scale, and bold weights are 700. Other than those definitions, the scale is somewhat arbitrary. The 100 to 900 values roughly correspond to common weight names, but these are approximations since font weights are rarely numbered and don't fit into precise pigeonholes.

	Weight	Common names
A	100	ultra light, extra light, thin, hairline
A	200	light
A	300	book
A	400	regular, normal
A	500	medium
A	600	semibold, demibold
A	700	bold
A	800	extra bold, black
A	900	ultra bold, heavy, poster

One designer's *medium* is another designer's *semibold* (and that's just for de facto font names in English). Unless you are specifically setting `font-weight` in a `@font-face` rule, you may need to experiment with the number scale to select your intended font.

It's very rare for a font family to contain nine different weights. If you specify a weight for which no font exists, the nearest weight is used. Bold weights (600 upwards) will map to the next heaviest font. Light weights (500 downwards) will select the next lightest font. Synthesis of weights in browsers is limited to making a bold from a lighter weight. Browsers do not increase the weight further or try to lighten the font.



Weight mappings for a family with 300, 700 and 900 weights.

The numeric weight system, which has existed in the css specification since it was first published in 1996, was intended to support a dynamic stylistic range from the get-go¹.

“It makes sense to recognize integers between 100 and 900. One of the reasons we chose to use three-digit numbers was to support intermediate values in the future.”

— Håkon Wium Lie, co-inventor of css

Integers are likely to be introduced in css Fonts Level 4 with the advent of variable fonts (see ‘[Using web fonts](#)’), but as of css3 only intervals of 100 are allowed. If you use any other number, browsers will not round to the nearest hundred; instead, they will treat the style rule as invalid and ignore it. This is the correct behaviour, but will make any future use of a more fine-grained scale problematic until browser support catches up.

Make links clear but unobtrusive

Hyperlinks – actionable connections joining one online resource to another – are unique to digital media. Clickable links put the ‘web’ into World Wide Web, and are fundamental to its adoption, growth and incorporation into modern society.

¹ Reaffirmed by Håkon Wium Lie in a post to the [css Working Group](#) (<http://wbttyp.net/145>), January 2015.

The perfect link should be visible but unobtrusive, allowing people to realise what's clickable, but without drawing too much attention to itself. Often, context is enough to indicate the presence of a link: site navigation, lists of further reading, search results should all be self-evident. Inline links – hyperlinked words and phrases within running text – are more problematic. Inline links pose a compromise between allowing your reader to flow through the text with minimal distraction, while still making it clear that a link is present. The default treatment of links – bright blue underlined text – draws way too much attention. The links pop out like molehills on an ornamental lawn.

In the embellishment or decoration of a message, there lies the opportunity for the little added touch that goes far beyond the bare essentials of taste and into the realm of fantasy. For this reason the ornamentation of printing is at once the most charming and the most dangerous diversion the typographer can find: dangerous because all matters of decoration call upon the utmost discretion and sense of fitness for their effective use.

The default styling of links causes undue distraction.

Removing the underlining and reducing the colour intensity of the links would be two ways to lessen the distracting nature of the default styling. The downsides of this approach are that underlining links on the web is a well-recognised convention, and, just as importantly, the underlining helps colour-blind readers and those using monochrome displays to pick out the links. The [Web Content Accessibility Guidelines 2.0](http://wbtyp.net/83) (<http://wbtyp.net/83>) (WCAG2) specify that you can rely on colour alone to indicate links provided you are ‘using a contrast ratio of 3:1 with surrounding text and providing additional visual cues on focus’. (There are [online tools](http://wbtyp.net/85) (<http://wbtyp.net/85>) available for measuring colour contrast.) But in essence, if you remove the underline from links and wish them to remain accessible, you will need to render your inline links in a highly distinctive colour, or provide an alternative visual cue.

Links are not just the preserve of the web. Directions to other resources have long been provided in printed matter; techniques used include footnotes indicated with a superscript, a fist  pointing to the resource, or a simple arrow →. You could also use a background colour to highlight the link, or any combination thereof.

In the embellishment[°] or decoration of a message, there lies the opportunity[°] for the little added touch that goes far beyond the bare essentials of taste and into the realm of fantasy[°]. For this reason the ornamentation[°] of printing is at once the most charming and the most dangerous diversion the typographer can find: dangerous because all matters of decoration[°] call upon the utmost discretion and sense of fitness[°] for their effective use.

Links appended with [°] and background colours on hover in the style of Matthew Butterick.

Underlining remains the web's idiomatic way to indicate a link and for that reason it should always be your first consideration. Like any mark which has the potential to detract from the reading experience, handle underlines with great care. The ideal underline should be positioned at just the right distance from the text, sitting comfortably behind it when descenders want to occupy the same space. It should be as thick or thin as the serifs, or as thick as the thinnest part of sans serif type.

parapsychologists

Subtle underlining avoiding descenders.

To set the best possible underline, we would need to be able to:

- adjust the thickness of the line
- change the distance of the line from the text
- change the colour
- skip descenders
- separately style hovered, focused and visited links
- have the underlining wrap with the text

The standard way to toggle an underline is with the `text-decoration` property. To remove underlines from links except when they are hovered over or have focus:

```
a {  
    text-decoration: none;  
}  
  
a:focus, a:hover {  
    text-decoration: underline;  
}
```

Using `text-decoration` alone affords us no control. Browsers implement underlines in many subtle – and not so subtle – differing ways: they position the underline in different places; they usually use the same colour as the link text, but not always; some increase the thickness of the underline as the font size increases, some set the thickness according to screen resolution, others always stick to 1px. In all cases, the presentation of the underline is rarely good enough.

The `text-decoration` property has been hugely expanded in CSS3 and provides us with some of the control we need:

- `text-decoration-color` changes the colour
- `text-decoration-skip` skips descenders and other content
- `text-decoration-style` selects solid, dotted or wavy lines

However, the thickness and position are still controlled by the browser, and (at the time of writing) support for CSS3 text decorating is patchy at best. To wrest full control of underlines, therefore, requires some hacks and workarounds. Turning off the text decoration and applying a bottom border instead can often work well:

```
a {  
    text-decoration: none;  
    border-bottom: 1px solid #ccc;  
}  
  
@supports (text-decoration-skip: ink) {  
    a {  
        text-decoration: underline 1px solid #ccc;  
        text-decoration-skip: ink;  
        border-bottom: 0;  
    }  
}
```

The downside of this approach is that you still don't have control over the positioning. This can be a problem: depending on the font metrics and your line spacing, the border can sometimes be placed too low. A further workaround is to use a background-image gradient for the underline and a white text shadow to clear the descenders, both of which can affect browsing performance on slow devices. See experiments by Gary Hepting² and an explanation by Marcin Wichary of Medium's underlines³ for more details on these clever hacks.

2 Gary Hepting's experiments (<http://wbtyp.net/18>) on Codepen.

3 'Crafting link underlines on Medium' (<http://wbtyp.net/36>) by Marcin Wichary in *Designing Medium* (2014).

Numerals and tables

Take just as much care over numerical and tabular content as you would do over text. Use the correct form of numeral for the situation, and design tables to be read, not just looked at.

Use old-style numerals in running text

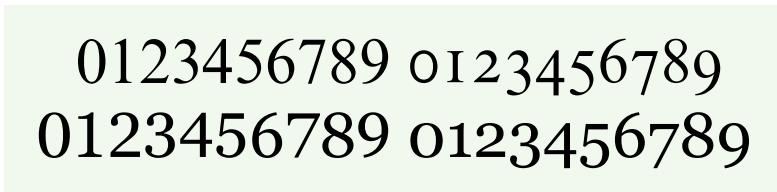
In ‘[Ligatures and abbreviations](#)’ we established that writing systems based on the Latin alphabet, in addition to Greek and Cyrillic, use a bicameral script, with each letter represented by two different forms – uppercase and lower (or majuscule and minuscule, to use more formal terms). The same is true of numerals. We have at our disposal ‘uppercase’ numbers 0123456789 called *lining* or *titling* numerals, and ‘lowercase’ numerals 0123456789 called *old-style* or *text* numerals.

Unlike capital and lowercase letters, different forms of numbers do not convey different meanings; they are, however, an essential component of the typographer’s palette. Just as a string of capital letters in the middle of a sentence SHOUTS at your reader, so numbers set in lining numerals call undue attention to themselves. Are pounds, dollars, dates and quotas really more important than the words and ideas which give them context and meaning?

Treat numbers as you would letters, making sure they don’t stand out unnecessarily. Do this by using old-style numerals in all your running text. Most modern, professional fonts will include both old-style and lining numerals as OpenType features. One or other of these styles will be used for the default numbers. More often it will be the old-style numerals, but there is no strict rule or consistency, and the choice of default is down to the type designer.

It’s also the case that the vast majority of fonts are neither modern nor professional, if modern means OpenType-enabled and professional means designed with both sets of numerals. Take Georgia, for example. Designed

by Matthew Carter in 1993 as a screen font for Microsoft, it is extremely well put together, elegant and appealing, and one of the most popular and widely distributed fonts in the world. But it is not packaged as an OpenType font and so only contains one set of numbers, in this case old-style numerals. Times New Roman, which is similarly widespread but, again, not as an OpenType font, is packaged only with lining numerals. Georgia and Times New Roman are so widely distributed because they are bundled free of charge with Windows and Mac operating systems. However, both these fonts – like many others – are available to buy in professional versions, which do come as OpenType fonts complete with both sets of numerals, small caps and many other features.



0123456789 0123456789
0123456789 0123456789

Top: Numerals in Times New Roman Pro. *Bottom:* Numerals in Georgia Pro.

To specify old-style numerals, set the `font-variant-numeric` property with a value of `oldstyle-nums`. If most of what you’re designing on a page is running text, then your best approach is to set old-style numerals so that they are inherited from the `<body>`.

```
body { font-variant-numeric: oldstyle-nums; }
```

For legacy browsers requiring `font-feature-settings`, use the `onum` OpenType feature tag. As explained in ‘[Ligatures and abbreviations](#)’, you can add an `@supports` rule to cater for legacy browsers that only support `font-feature-settings`:

```
body { font-feature-settings: "onum" 1; }

@supports (font-variant-numeric: oldstyle-nums) {
  body {
    font-feature-settings: normal;
    font-variant-numeric: oldstyle-nums;
  }
}
```

Many sans serif fonts of the early to mid-twentieth century, including Helvetica, were never designed with anything other than lining numerals. This is one of the reasons why Helvetica is rarely your best choice for body text. That said, the lining numerals are less of a problem in Helvetica than they are in some other typefaces.

As we saw in ‘[Designing paragraphs: line spacing](#)’, Helvetica has a large x-height. A consequence of this is that its lowercase letters are closer in stature to its lining numerals when compared to other sans serif fonts such as Futura and Avenir, which have far smaller x-heights.

For nearly 50 years the Swiss type designer Adrian Frutiger, born in 1928, was a hugely influential figure in typography.

For nearly 50 years the Swiss type designer Adrian Frutiger, born in 1928, was a hugely influential figure in typography.

Compared with old-style numerals, lining numerals shout loudly in Avenir.

Clearly Paul Renner and Adrian Frutiger, the designers of Futura and Avenir respectively, recognised the need for old-style numerals in their fonts as both these typefaces were designed with them from the start. Sadly, the versions of Futura and Avenir widely distributed with Apple devices have lining numerals as the default, and do not include old-style numerals as OpenType features (the macOS version of Avenir Next, however, does include them).

Use lining numerals in headings

Old-style numerals are your go-to glyphs for making numbers sit well in running text. For the same reason they are at ease with lowercase letters, so old-style numerals feel uncomfortable in close proximity to capital letters. If you set headings in anything other than sentence case, in particular ALL CAPS, or Title Case, then don’t use old-style numerals. Lining numerals will sit far more naturally in the company of uppercase letterforms.

1912 Ditchling Sussex

Edward Johnston moves to the village, where Eric Gill already lived and an artistic community was forming.

1912 DITCHLING SUSSEX

Edward Johnston moves to the village, where Eric Gill already lived and an artistic community was forming.

1912 Ditchling Sussex

Edward Johnston moves to the village, where Eric Gill already lived and an artistic community was forming.

1912 DITCHLING SUSSEX

Edward Johnston moves to the village, where Eric Gill already lived and an artistic community was forming.

Lining numerals sit more naturally in headings than old-style numerals.

On those occasions when numbers are the star attraction, lining numerals are the way to give them the attention they crave. Old-style numerals have a wavy rhythm to them, with some numbers reaching upwards towards the capitals, some squatting at the x-height, and others ducking down below the baseline: 1234567890. This is why they work so well in continuous reading – they replicate the patterns of words in running text. However, if your reader is scanning a sequence of numbers, looking for patterns, making comparisons, or hunting for data in a list, table or other setting, they will find it far easier to do so with the familiarity and evenness of lining numerals. To specify lining numerals, set the `font-variant-numeric` property with a value of `lining-nums`:

```
h1 { font-variant-numeric: lining-nums; }
```

For legacy browsers requiring `font-feature-settings`, use the `lnum` OpenType feature tag.

Use proper subscripts and superscripts

Subscripts and superscripts are tiny numerals which are lowered or raised. They are used in chemical and mathematical formulas, as footnote indicators, and other specialist situations. For example: ‘Caffeine¹ is C₈H₁₀N₄O₂.’ Mark this up meaningfully in HTML using the `<sup>` and `<sub>` elements:

```
Caffeine<sup>1</sup> is C<sup>8</sup>H<sup>10</sup>N<sup>4</sup>O<sup>2</sup>.
```

Browsers' default styling for `<sup>` and `<sub>` is to take a regular numeral, make it a bit smaller, and raise or lower it accordingly:

Caffeine¹ is C₈H₁₀N₄O₂

This works fine up to a point, but the numerals are still a little too big aesthetically and they affect the line spacing, causing unevenness in the rhythm:

~~Caffeine is a central nervous system stimulant of the methylxanthine class. Its chemical formula is C₈H₁₀N₄O₂. It is a bitter, white crystalline purine, a methylxanthine alkaloid, and is chemically related to the adenine and...~~

Most professional fonts contain properly designed subscripts and superscripts built in as OpenType features. These numerals are smaller and squatter than regular numbers, and because their position relative to other characters is part of their design, the line spacing is unaffected:

Caffeine¹ is C₈H₁₀N₄O₂

To use proper subscripts and superscripts, use the `font-variant-position` property, like this:

```
sub { font-variant-position: sub; }
sup { font-variant-position: super; }
```

Unfortunately, this still leaves us with a problem: the browser's default styling is still applied. Our special subscript character is being made smaller and it's getting moved downwards, affecting the line spacing:

Caffeine is a central nervous system stimulant of the methylxanthine class. Its chemical formula is $\text{C}_8\text{H}_{10}\text{N}_4\text{O}_2$. It is a bitter, white crystalline purine, a methylxanthine alkaloid, and is chemically related to the adenine and...

Caffeine is a central nervous system stimulant of the methylxanthine class. Its chemical formula is $\text{C}_8\text{H}_{10}\text{N}_4\text{O}_2$. It is a bitter, white crystalline purine, a methylxanthine alkaloid, and is chemically related to the adenine and...

Caffeine is a central nervous system stimulant of the methylxanthine class. Its chemical formula is $\text{C}_8\text{H}_{10}\text{N}_4\text{O}_2$. It is a bitter, white crystalline purine, a methylxanthine alkaloid, and is chemically related to the adenine and...

Top: Default `<sub>` styling. *Middle:* Proper subscripts with browser default styling.
Bottom: Proper subscripts only.

The styles the browser applies to our subscript characters are these:

```
vertical-align: sub; font-size: smaller;
```

We need to remove those styles to get the desired effect, so our rule now becomes:

```
sub { vertical-align: inherit;
      font-size: inherit;
      font-variant-position: sub; }
```

That will work fine for browsers that support OpenType. But browsers that don't will get C8H10N4O2, a degraded rendering compared with the browser defaults. To address this we can use an `@supports` rule to check if the browser supports `font-variant-position` and only override the browser's default `<sub>` styling if that's the case:

```
sub { font-variant-position: sub; }

@supports (font-variant-position: sub) {
  sub { vertical-align: inherit;
        font-size: inherit; }
}
```

For legacy browsers requiring `font-feature-settings`, use the `sups` OpenType feature tag for superscripts, and `subs` for subscripts. If we factor these in, we get comprehensive, backwards-compatible style rules, but where two simple properties should have done the job, we now have a far more verbose proposition:

Subscripts

```
sub { font-feature-settings: "subs" 1; }

@supports (font-variant-position: sub) {
    sub { font-feature-settings: normal;
          font-variant-position: sub; }
}

@supports ((font-variant-position: sub) or (font-feature-
settings: "subs" 1)) {
    sub { vertical-align: inherit;
          font-size: inherit; }
}
```

Superscripts

```
sup { font-feature-settings: "sups" 1; }

@supports (font-variant-position: super) {
    sup { font-feature-settings: normal;
          font-variant-position: super; }
}

@supports ((font-variant-position: super) or (font-
feature-settings: "sups" 1)) {
    sup { vertical-align: inherit;
          font-size: inherit; }
}
```

Reference notes with superscripts

One particular use of superscripts is for footnotes. When you reference notes using numbers, use true superscripts in the text but full-size numbers in the notes themselves.

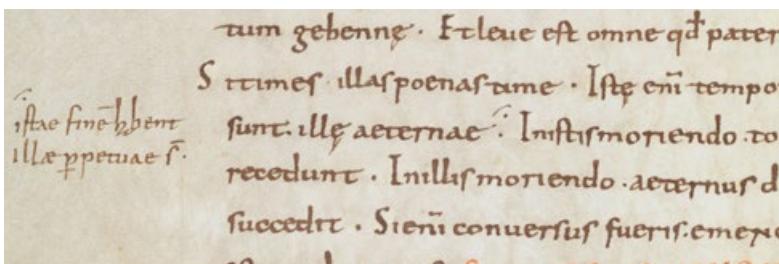
Show footnotes in context

While we're on the subject of footnotes, it's worth making a brief diversion into how the web improves their usability compared with the limitations of paper.

Many forms of writing, including academic papers, historical novels, detailed journalism and non-fiction books such as this one, contain additional citations, explanations and thoughts referred to within the text itself. A symbol is used to connect the note to the relevant location in the text. The symbols employed as references to annotations are either superscripted numbers or an esoteric series of devices starting with asterisks* and processing through daggers† to double daggers‡ and beyond.

Since the advent of mass printing in the Victorian era, the notes themselves have typically been positioned either at the bottom of the referring printed page (*footnotes*), or at the end of a chapter or the entire work (*end-notes*). However, this approach means the notes are located away from their position within the body of text. This can disturb the reader who wishes to refer to the annotation as they proceed through the text. The connected point in the text may well be halfway through a sentence in the middle of a paragraph at some point higher up the page, or on a different preceding page altogether, and attempting to return to it disrupts the reader's rhythm.

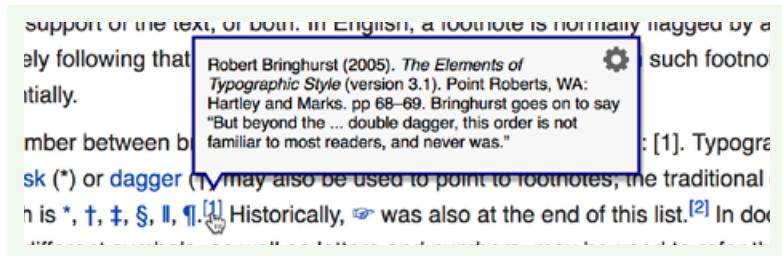
An earlier approach by medieval scribes and Renaissance printers placed notes in the margins (*side notes*) rather than at the bottom of the page. By including notes as marginalia, the annotations are present where needed and can be read with little more than a glance away from the main text.



A side note in a 9th-century manuscript. Source: Einsiedeln, Stiftsbibliothek, Codex 172(1128), p.20.

Although side notes are an improvement on footnotes, both solutions are designed within the confines of the two-dimensional printed page. The web is an interactive medium and provides us with at least three dimensions in which to work, implying you can use the z-axis to place the note *on top of* the main text.

Enable your reader to reveal the note on demand in the very place they are reading. Put simply, link to the footnote using a conventional symbol, but have it pop up in the vicinity of the link, thus providing a thoroughly modern solution impossible within the limitations of a printed page.



Clicking a superscript could pop up a footnote in situ.

Set tables as text to be read

Tables come in many forms. Some contain simple numbers, others are complex with mixtures of numeric data and textual information. Some require reading row by row, others are scanned vertically. The potential use for tables is as varied as the written word. They can be financial statements, bus timetables, multilanguage dictionaries, tables of contents, numerical conversions, pricing options, feature comparisons, technical specifications, and so on.

Despite the huge variation in table size, complexity, contents and purpose, every table shares two simple design principles: they should be readable and support a sense of the data held within. They should not be prettied up to satisfy a sense of aesthetic when simply looked at. That said, a well-designed table can still be a thing of beauty but with the form following the function. Tables are not pictures of data: they are catalogues of data to be perused, parsed, referenced and interrogated. A well-designed table will enable the information to be read and understood, and will reveal the patterns and correlations inherent in the data. As Jan Tschichold, the virtuoso of typography design, put it in *Asymmetric Typography*¹:

Tabular matter need no longer be a rather unpleasant job to design: on the contrary, it can become a really charming and artistic exercise, in no way less interesting than any other area.

¹ *Asymmetric Typography* by Jan Tschichold (1967, after 1935 original).

Wherever possible plan the readability of every table in advance. Your design process should be an investigation into making the data undemanding to read, simple to follow and easy to extract.

Just as you wouldn't design body text with the aim of fitting as many words as possible on the screen, so you shouldn't treat designing a table as an exercise in cramming as much data as possible into one space. You might be tempted to reduce the text size of your table – and if the data is entirely numeric you might be able to get away with it. Your reader should still be able to be comfortably read and interpret the table from their normal position, without needing to lean in.

Don't stretch tables

Many designers will instinctively apply a width to their tables – just as they might an image – stretching them to fill the text column or page. And that is the appeal of setting tables full-width: you can make them look somewhat image-like when viewed from afar. However, while a table spread across the screen might look preferable from a distance, on closer inspection it will be harder to read as the data will be unnecessarily separated. To add insult to injury, tables set full-width are often replete with background colours and borders to give the table further the texture of an image, when what your reader really requires is the texture of text. For the sake of your readers, avoid these temptations.

You might consider making all the columns an even width. This too does nothing for the readability of the contents. Some table cells will be too wide, leaving the data lost and detached from its neighbours. Other table cells will be too narrow, cramping the data uncomfortably. Table columns should be sized according to the data they contain. Columns of small numbers should be narrow, and columns of paragraphs should be relatively wide. This sounds like a lot of effort, and for a print designer it would be, as they would have to size each column manually in their layout software. Fortunately, web browsers are very clever when it comes to laying out tables and will do all that hard work for you. Browsers have been laying out tables automatically according to complex algorithms since long before CSS came along – just let them do their thing.

Keep table furniture and fills to a minimum

The statistician and information designer Edward Tufte introduced the concept of *data-ink* in his 1983 classic, *The Visual Display of Quantitative Information*. He defines data-ink as 'the non-erasable core of the graphic', whereas non-data-ink is the ink used in the graphic, not to directly represent data

but for scales, labels, fills and edges. Tufte goes on to define the *data-ink ratio* as the proportion of ink that is used to present actual data compared to the total amount of ink used in the entire graphic. The goal is to design a graphic with the highest possible data-ink ratio (tending towards 1.0) without eliminating what is necessary for effective communication.

Where Tufte talks about graphics he includes charts, diagrams and tables, and where he uses ‘ink’ we can think of pixels. In terms of tables, he’s saying that we should remove almost everything in the design which is not data or white space. Minimise furniture, maximise information. This is an ideal first principle to bear in mind when considering the *typographic* design of a table.

As a starting point, avoid any border or frame surrounding the table. This is a Victorian embellishment which is entirely unnecessary as text alignment will shape the table just fine.

Try to achieve a readable table using just alignment, spacing and grouping. Avoid zebra striping, tints and fills, and any other backgrounds. These can be superficially pretty but are usually a distraction. They serve to distort the meaning of the data by highlighting every other row to the detriment of neighbouring rows. Only use tints as a subtle means of guiding your reader’s eyes, and then only if you cannot arrange the data to that end. If you choose to tint, do so only in the primary direction of reading: down if lists, across otherwise.

When it comes to lines and borders between rows and columns – typographically referred to as *rules* – the same applies: use them judiciously and preferably not at all. In *Asymmetric Typography* Jan Tschichold sums this up wonderfully:

*Tables should not be set to look like nets with every number enclosed.
Try to do without rules altogether. They should be used only when
they are absolutely necessary. Vertical rules are needed only when the
space between columns is so narrow that mistakes will occur in reading
without rules. Tables without vertical rules look better. Thin rules are
better than thick ones.*

Avoid using row or column borders unless the data alignment, spacing and grouping are not sufficient to guide your reader’s eye. If you do need to use rules for this purpose, use them in one direction only and employ a lighter colour to reduce the impact of the lines: you are making a distinction, not constructing a barricade.

Left-align text, right-align numbers, and align headings with data

In the spirit of treating tables as artefacts to be read, don't centre-align text within tables. Align table text as you would anywhere else; that is, aligned left. As text in tables tends to end up in narrow columns, don't justify the text either – leave it ragged-right – or you will end up with rivers flowing down the tables, potentially causing confusion and certainly harming readability. You can hyphenate, however, particularly if the table columns would otherwise have a pronounced rag.

Right-align numbers to help your reader make easier comparisons of magnitude when scanning down columns. To aid scanning in this manner you will need consistent precision of your numeric data; that is, use the same number of decimal places.

For consistency and ease of understanding, match the alignment of headings to the alignment of the data. Right-align headings of numeric data and left-align headings of columns with text, for example:

Country	Area	Population	GDP	Capital
Austria	83,858	8,169,929	339	Vienna
Belgium	30,528	11,007,000	410	Brussels
Denmark	43,094	5,564,219	271	Copenhagen
France	547,030	66,104,000	2,181	Paris
Germany	357,021	80,716,000	3,032	Berlin
Greece	131,957	11,123,034	176	Athens
Ireland	70,280	4,234,925	255	Dublin
Italy	301,230	60,655,464	1,642	Rome
Luxembourg	2,586	448,569	51	Luxembourg
Netherlands	41,526	16,902,103	676	Amsterdam
Portugal	91,568	10,409,995	179	Lisbon
Spain	504,851	47,059,533	1,075	Madrid
Sweden	449,964	9,090,113	447	Stockholm
UK	244,820	65,110,000	2,727	London

Align to the decimal point

You may find yourself not having control of numerical precision, or perhaps the data you're working with is rounded to the same *significant number* rather than adhering to the same precision. In this case, simply right-aligning a column of numbers will not help your reader scan down the column: small, high-precision numbers will look at first glance like a large number. Instead, align numbers to the decimal point. This will enable your reader to more readily compare magnitudes among a wider variety of data:

Call charge
\$1.30
\$2.50
\$10.80
\$111.01
\$85
N/A
\$.05
\$.06

Aligning to the decimal point was theoretically possible by using the `HTML4 char` attribute on a `<td>` tag, but in reality it was never supported. The modern way to align numbers to a decimal point (or to any character, in fact) is through a new value of the `text-align` property, although at the time of writing this is languishing in the `css Text Level 4 Module2` and support is patchy at best.

The syntax of the new value is simple. You include the alignment character (usually a full stop or comma) in quotes, followed by a space and your desired alignment keyword, which defaults to `right` if you omit it. For example, the following will centre the data and align to a decimal point as in our prior example:

```
td { text-align: "." center; }
```

By specifying different alignment characters you can lay out more complex tables in a useful way; in this example, aligning digits to ‘.’ and ‘?’.

² [Character-based Alignment in a Table Column](http://wbtyp.net/103) (<http://wbtyp.net/103>) in the `css Text Module Level 4`.

Video standard	Resolution	Pixels	Aspect ratio
QQVGA	160 × 120	19k	4:3
HQVGA	240 × 160	38k	3:2
QVGA	320 × 240	76k	4:3
WQVGA	480 × 272	130k	16:9
VGA	640 × 480	307k	4:3
SVGA	800 × 600	480k	4:3
XGA	1024 × 768	786k	4:3
HD	1260 × 768	967k	16:9
WXGA	1280 × 800	1,024k	16:10
SXGA	1280 × 1024	1,310k	5:4
UXGA	1600 × 1200	1,920k	4:3
FHD	1920 × 1080	2,073k	16:9
DCI 2K	2048 × 1080	2,211k	19:10
WQXGA	2560 × 1600	4,096k	16:10
4K UHD	3840 × 2160	8,294k	16:9
8K UHD	7680 × 4320	33,177k	16:9

Selected display standards

Use tabular lining numerals in tables of numbers

Many tables, such as financial statements or timetables, are made up mostly of numbers. Generally speaking, their purpose is to provide the reader with numeric data, presented in either columns or rows, and sometimes in a matrix of the two. Your reader may use the table by scanning down the columns, either searching for a data point or by making comparisons between numbers. Your reader may also make sense of the data by simply glancing at the column or row. It is far easier to compare numbers if the ones, tens and hundreds are all lined up vertically; that is, all the digits should occupy exactly the same width.

Digits of the same width can inherently be found in monospaced fonts, and there is nothing wrong with choosing a suitable monospaced font to present a table of data (see ‘Combining typefaces’). However, many proportionally spaced fonts (those where a *1* is narrower than an *8*, and a *W* is wider than an *I*) also come with additional sets of figures which are monospaced.

These are called *tabular numerals*. As well as being of equal width, tabular numerals will be subtly designed differently from the standard proportional numerals. For example, a 1 will normally have a bar for its base, and a 0 (zero) may be designed slightly narrower to better fit the chosen number width. Tabular numerals are usually available in old-style and lining variations. Use tabular lining numerals to provide your reader with the most effective way to reference vertically and horizontally in tables of data.

	Lining	Old-style
Proportional	409,280 367,112 155,068 171,792	409,280 367,112 155,068 171,792
Tabular	409,280 367,112 155,068 171,792	409,280 367,112 155,068 171,792

Different numeral styles compared.

To specify tabular lining numerals, set the `font-variant-numeric` property with a value of `lining-nums` and `tabular-nums`:

```
table { font-variant-numeric: lining-nums tabular-nums; }
```

The equivalent properties for legacy browsers requiring `font-feature-settings`, use the `lnum` and `tnum` OpenType feature tags.

Proportional numerals

If you need to specify proportional numerals, set the `font-variant-numeric` property with a value of `proportional-nums`. For legacy browsers requiring `font-feature-settings`, use the `pnum` OpenType feature tag.

Put white space to work to group and separate

Having eliminated rules and fills (borders and backgrounds) from your table, you will need to apply white space to your table so your reader can make sense of it. It is at this point that you should remove from your mind's eye all visions of spreadsheets and other such uniform grids, and think instead in terms of typography and simple gestalt grouping principles.

You will primarily need to separate the data so that each element can be individually identified and read as separate from the others. To have more control over the spacing, first collapse the spacing between borders:

```
table { border-collapse: collapse; }
```

In traditional HTML tables, adjacent cells each have their own distinct borders which are separated from each other, with the separation still present even if the borders are not. In the collapsed border model, adjacent table cells share borders. As we are removing (almost) all cell borders, and any we retain will be single key lines, the collapsed border model is the most appropriate.

Now apply padding to the table cells to separate the data. You'll find that adding a smaller amount of padding to the top of the cell is a useful way to provide a visually balanced separation from the rows above and below. To ensure everything lines up nicely, apply the same padding to heading cells as to data cells. Because line lengths are often very short in tables, you can reduce the line height right down. In the following example, we've removed all additional line spacing, but you may need more depending on your choice of font and the amount of text in the table cells.

```
td, th {  
  padding: 0.125em 0.5em 0.25em 0.5em;  
  line-height: 1;  
}
```

The gestalt grouping principles most useful in tables are those of proximity and similarity. Move related data closer together to be distinct from other data; in other words, space apart groups of rows or columns. A by-product of grouping rows is that the data becomes much easier to scan and refer to than if the table consisted of a succession of undifferentiated rows. Ensure data of a similar content or meaning look similar at a glance, which you can do through alignment, colour and font style.

Table captions

We will attend to the typographic specifics of table captions in '[Choosing typefaces for functional text](#)' but it's worth noting now how to mark up captions for tables. If you are choosing to place your table inside a `<figure>` element, which is a perfectly reasonable thing to do, then use a `<figcaption>` element before or after the table. If your table is not inside

a `<figure>` element, or you have multiple items in the figure, use the aptly named `<caption>` element, which HTML provides specifically for tables. Always write the `<caption>` tag immediately after the opening `<table>` tag and before any table data, like this:

```
<table>
  <caption>
    Imperial to metric conversion factors
    <p><i>Values are given to three significant figures</i></p>
  </caption>
  <thead> ... </thead>
  <tbody> ... </tbody>
</table>
```

You can position the caption either above or below the table using the `caption-side` property and a corresponding value of either `top` or `bottom`.

```
caption { caption-side: bottom; }
```

The following table shows a caption and demonstrates gestalt grouping principles by separating the data into related rows:

	<i>To convert</i>	<i>into</i>	<i>multiply by</i>
Length	inches	millimetres (mm)	25.4
	feet	centimetres (cm)	30.48
	yards	metres (m)	0.914
	miles	kilometres (km)	1.61
Area	square inches	sq. millimetres (mm^2)	645
	square feet	square metres (m^2)	0.093
	square yards	square metres (m^2)	0.836
	acres	hectares	2.47
Volume	cubic inches	millilitres (ml)	16.4
	cubic feet	litres	28.3
	imperial gallons	litres	4.55
	US barrels	cubic metres (m^3)	0.159

Imperial to metric conversion factors
Values are given to three significant figures unless exact

Note that in this example, the numbers do not align to the decimal point. This is because the purpose of the table is for the reader to easily identify and extract a multiplication factor. In this instance there is no obvious use case for comparing the relative magnitudes of the factors, which is when decimal alignment would be useful.

Do not over-stylise tables

The French writer-aviator Antoine de Saint-Exupéry wrote³ ‘perfection is attained not when there is nothing more to add, but when there is no longer anything to take away.’ Quoting de Saint-Exupéry may have become a cliché, but his idiom is entirely apt when applied to table design.

There is no need to make a table look like a spreadsheet. A spreadsheet is a tool unto itself; a table is for presenting data and information that can be read. Spreadsheet software offers a multitude of options for table styles, which add text formatting, borders, background fills and all manner of ornament. They may make pretty pictures but do nothing for table readability, so do not try to emulate them. Tables can be beautiful but they are not works of art. Instead of painting and decorating them, design tables for your reader.

Director	Film	Released	Budget	Gross	Rating
John Ford	The Informer	May 1935	\$243,000	\$950,000	8.3/10
	The Grapes of Wrath	Jan 1940	\$800,000	\$2,500,000	9/10
	How Green Was My Valley	Oct 1941	\$800,000	\$2,800,000	7.9/10
	The Quiet Man	Jul 1952	\$1,750,000	\$3,200,000	8/10
William Wyler	Mrs. Miniver	Jun 1942	\$1,340,000	\$8,878,000	7.8/10
	The Best Years of Our Lives	Nov 1946	\$2,100,000	\$23,700,000	8.6/10
	Ben-Hur	Nov 1959	\$15,200,000	\$146,900,000	7.9/10
Frank Capra	It Happened One Night	Feb 1934	\$325,000	\$2,500,000	8.9/10
	Mr. Deeds Goes to Town	Apr 1936	\$845,710	\$1,000,000	8.4/10
	You Can't Take It With You	Aug 1938	\$1,644,736	\$5,295,526	7.6/10
Billy Wilder	The Lost Weekend	Nov 1945	\$1,250,000	\$11,000,000	8.2/10
	The Apartment	Jun 1960	\$3,000,000	\$24,600,000	8.5/10
David Lean	The Bridge on the River Kwai	Oct 1957	\$2,800,000	\$30,600,000	9.2/10
	Lawrence of Arabia	Dec 1962	\$15,000,000	\$70,000,000	9/10
Steven Spielberg	Schindler's List	Nov 1993	\$22,000,000	\$321,200,000	9/10
	Saving Private Ryan	Jul 1998	\$70,000,000	\$481,800,000	8.6/10
Fred Zinnemann	From Here to Eternity	Aug 1953	\$2,500,000	\$30,500,000	8.2/10
	A Man for All Seasons	Dec 1966	\$2,000,000	\$28,400,000	7.8/10

A typical spreadsheet-styled table set full-width with borders, fills and centre-alignment.

³ *Terre des Hommes* (translated into English as *Wind, Sand and Stars*) by Antoine de Saint-Exupéry (1939).

Director	Film	Released	Budget	Gross	Rating
John Ford	The Informer	May 1935	\$243,000	\$950,000	8.3/10
	The Grapes of Wrath	Jan 1940	\$800,000	\$2,500,000	9/10
	How Green Was My Valley	Oct 1941	\$800,000	\$2,800,000	7.9/10
	The Quiet Man	Jul 1952	\$1,750,000	\$3,200,000	8/10
William Wyler	Mrs. Miniver	Jun 1942	\$1,340,000	\$8,878,000	7.8/10
	The Best Years of Our Lives	Nov 1946	\$2,100,000	\$23,700,000	8.6/10
	Ben-Hur	Nov 1959	\$15,200,000	\$146,900,000	7.9/10
Frank Capra	It Happened One Night	Feb 1934	\$325,000	\$2,500,000	8.9/10
	Mr. Deeds Goes to Town	Apr 1936	\$845,710	\$1,000,000	8.4/10
	You Can't Take It With You	Aug 1938	\$1,644,736	\$5,295,526	7.6/10
Billy Wilder	The Lost Weekend	Nov 1945	\$1,250,000	\$11,000,000	8.2/10
	The Apartment	Jun 1960	\$3,000,000	\$24,600,000	8.5/10
David Lean	The Bridge on the River Kwai	Oct 1957	\$2,800,000	\$30,600,000	9.2/10
	Lawrence of Arabia	Dec 1962	\$15,000,000	\$70,000,000	9/10
	Schindler's List	Nov 1993	\$22,000,000	\$321,200,000	9/10
Steven Spielberg	Saving Private Ryan	Jul 1998	\$70,000,000	\$481,800,000	8.6/10
	From Here to Eternity	Aug 1953	\$2,500,000	\$30,500,000	8.2/10
Fred Zinnemann	A Man for All Seasons	Dec 1966	\$2,000,000	\$28,400,000	7.8/10

1. Remove stretch and size columns to data.

Director	Film	Released	Budget	Gross	Rating
John Ford	The Informer	May 1935	\$243,000	\$950,000	8.3/10
	The Grapes of Wrath	Jan 1940	\$800,000	\$2,500,000	9/10
	How Green Was My Valley	Oct 1941	\$800,000	\$2,800,000	7.9/10
	The Quiet Man	Jul 1952	\$1,750,000	\$3,200,000	8/10
William Wyler	Mrs. Miniver	Jun 1942	\$1,340,000	\$8,878,000	7.8/10
	The Best Years of Our Lives	Nov 1946	\$2,100,000	\$23,700,000	8.6/10
	Ben-Hur	Nov 1959	\$15,200,000	\$146,900,000	7.9/10
Frank Capra	It Happened One Night	Feb 1934	\$325,000	\$2,500,000	8.9/10
	Mr. Deeds Goes to Town	Apr 1936	\$845,710	\$1,000,000	8.4/10
	You Can't Take It With You	Aug 1938	\$1,644,736	\$5,295,526	7.6/10
Billy Wilder	The Lost Weekend	Nov 1945	\$1,250,000	\$11,000,000	8.2/10
	The Apartment	Jun 1960	\$3,000,000	\$24,600,000	8.5/10
David Lean	The Bridge on the River Kwai	Oct 1957	\$2,800,000	\$30,600,000	9.2/10
	Lawrence of Arabia	Dec 1962	\$15,000,000	\$70,000,000	9/10
	Schindler's List	Nov 1993	\$22,000,000	\$321,200,000	9/10
Steven Spielberg	Saving Private Ryan	Jul 1998	\$70,000,000	\$481,800,000	8.6/10
	From Here to Eternity	Aug 1953	\$2,500,000	\$30,500,000	8.2/10
Fred Zinnemann	A Man for All Seasons	Dec 1966	\$2,000,000	\$28,400,000	7.8/10

2. Remove fills, gridlines, border and bolding.

Director	Film	Released	Budget	Gross	Rating
John Ford	The Informer	May 1935	\$243,000	\$950,000	8.3/10
	The Grapes of Wrath	Jan 1940	\$800,000	\$2,500,000	9/10
	How Green Was My Valley	Oct 1941	\$800,000	\$2,800,000	7.9/10
	The Quiet Man	Jul 1952	\$1,750,000	\$3,200,000	8/10
William Wyler	Mrs. Miniver	Jun 1942	\$1,340,000	\$8,878,000	7.8/10
	The Best Years of Our Lives	Nov 1946	\$2,100,000	\$23,700,000	8.6/10
	Ben-Hur	Nov 1959	\$15,200,000	\$146,900,000	7.9/10
Frank Capra	It Happened One Night	Feb 1934	\$325,000	\$2,500,000	8.9/10
	Mr. Deeds Goes to Town	Apr 1936	\$845,710	\$1,000,000	8.4/10
	You Can't Take It With You	Aug 1938	\$1,644,736	\$5,295,526	7.6/10
Billy Wilder	The Lost Weekend	Nov 1945	\$1,250,000	\$11,000,000	8.2/10
	The Apartment	Jun 1960	\$3,000,000	\$24,600,000	8.5/10
David Lean	The Bridge on the River Kwai	Oct 1957	\$2,800,000	\$30,600,000	9.2/10
	Lawrence of Arabia	Dec 1962	\$15,000,000	\$70,000,000	9/10
Steven Spielberg	Schindler's List	Nov 1993	\$22,000,000	\$321,200,000	9/10
	Saving Private Ryan	Jul 1998	\$70,000,000	\$481,800,000	8.6/10
Fred Zinnemann	From Here to Eternity	Aug 1953	\$2,500,000	\$30,500,000	8.2/10
	A Man for All Seasons	Dec 1966	\$2,000,000	\$28,400,000	7.8/10

3. Left-align text, right-align numbers and align headings with data.

Director	Film	Released	Budget	Gross	Rating
John Ford	The Informer	May 1935	\$243,000	\$950,000	8.3/10
	The Grapes of Wrath	Jan 1940	\$800,000	\$2,500,000	9/10
	How Green Was My Valley	Oct 1941	\$800,000	\$2,800,000	7.9/10
	The Quiet Man	Jul 1952	\$1,750,000	\$3,200,000	8/10
William Wyler	Mrs. Miniver	Jun 1942	\$1,340,000	\$8,878,000	7.8/10
	The Best Years of Our Lives	Nov 1946	\$2,100,000	\$23,700,000	8.6/10
	Ben-Hur	Nov 1959	\$15,200,000	\$146,900,000	7.9/10
Frank Capra	It Happened One Night	Feb 1934	\$325,000	\$2,500,000	8.9/10
	Mr. Deeds Goes to Town	Apr 1936	\$845,710	\$1,000,000	8.4/10
	You Can't Take It With You	Aug 1938	\$1,644,736	\$5,295,526	7.6/10
Billy Wilder	The Lost Weekend	Nov 1945	\$1,250,000	\$11,000,000	8.2/10
	The Apartment	Jun 1960	\$3,000,000	\$24,600,000	8.5/10
David Lean	The Bridge on the River Kwai	Oct 1957	\$2,800,000	\$30,600,000	9.2/10
	Lawrence of Arabia	Dec 1962	\$15,000,000	\$70,000,000	9/10
Steven Spielberg	Schindler's List	Nov 1993	\$22,000,000	\$321,200,000	9/10
	Saving Private Ryan	Jul 1998	\$70,000,000	\$481,800,000	8.6/10
Fred Zinnemann	From Here to Eternity	Aug 1953	\$2,500,000	\$30,500,000	8.2/10
	A Man for All Seasons	Dec 1966	\$2,000,000	\$28,400,000	7.8/10

4. Put white space to work to group and separate.

Director	Film	Released	Budget (\$)	Gross (\$)	Rating
John Ford	The Informer	May 1935	243,000	950,000	8.3
	The Grapes of Wrath	Jan 1940	800,000	2,500,000	9.0
	How Green Was My Valley	Oct 1941	800,000	2,800,000	7.9
	The Quiet Man	Jul 1952	1,750,000	3,200,000	8.0
William Wyler	Mrs. Miniver	Jun 1942	1,340,000	8,878,000	7.8
	The Best Years of Our Lives	Nov 1946	2,100,000	23,700,000	8.6
	Ben-Hur	Nov 1959	15,200,000	146,900,000	7.9
Frank Capra	It Happened One Night	Feb 1934	325,000	2,500,000	8.9
	Mr. Deeds Goes to Town	Apr 1936	845,710	1,000,000	8.4
	You Can't Take It With You	Aug 1938	1,644,736	5,295,526	7.6
Billy Wilder	The Lost Weekend	Nov 1945	1,250,000	11,000,000	8.2
	The Apartment	Jun 1960	3,000,000	24,600,000	8.5
David Lean	The Bridge on the River Kwai	Oct 1957	2,800,000	30,600,000	9.2
	Lawrence of Arabia	Dec 1962	15,000,000	70,000,000	9.0
Steven Spielberg	Schindler's List	Nov 1993	22,000,000	321,200,000	9.0
	Saving Private Ryan	Jul 1998	70,000,000	481,800,000	8.6
Fred Zinnemann	From Here to Eternity	Aug 1953	2,500,000	30,500,000	8.2
	A Man for All Seasons	Dec 1966	2,000,000	28,400,000	7.8

5. Use tabular lining numerals, consistent precision, and remove repetition.

Adapt tables to small screens

Tables regularly require a fair bit of horizontal space to display the information they contain. Even when judiciously designed and edited, a typical table may need to be wider than the 45–75 characters we normally allow for paragraphs of text. For small screens, such as phones, designing readable tables which work under such cramped conditions presents us with a serious challenge. The best approaches always deal with each table on case-by-case basis, but that's not always possible if we need to generically style whatever comes out of a CMS database.

One immediate approach is to use either a condensed font or a slightly smaller size (but not both smaller and condensed). In both cases, readability must remain paramount and other options should also be explored.

Consider setting oblique headings to save space

One way to save horizontal space, particularly when you have short pieces of data but long headings, is to set the headings at an oblique angle.

Celsius (°C)	Fahrenheit (°F)
0	32
10	50
20	68
30	86
40	104
50	122
60	140
70	158
80	194
90	194
100	212

Using oblique headings to save space.

You can use a simple css translation to achieve the effect. You will also need to absolutely position the headings so the original width of the columns isn't retained and they shrink to wrap the data instead.

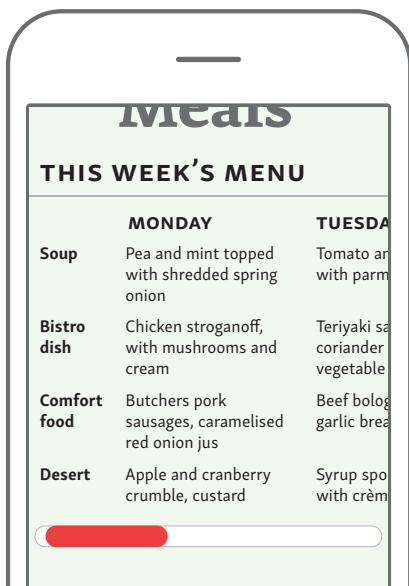
```
th {
    transform-origin: bottom left;
    position: absolute;
}

th.degC {
    transform: translate(2.58em, -2em) rotate(-60deg);
}

th.degF {
    transform: translate(5.14em, -2em) rotate(-60deg);
}
```

Let the browser handle tables with horizontal scrolling

The simplest solution to help tables of any size and complexity is to let the browser lay out the table as best it can and render part of the table off-screen as necessary. Provided you then enable your reader to scroll the table sideways independently of the rest of the text, the table can be relatively easily brought into view.



Using a crawl bar to scroll a table into view.

To do this, first wrap your table in a `<figure>` element:

```
<figure class="fig-table">
  <table> ... </table>
</figure>
```

Then apply the following simple rules to hide the portion of the table off-screen and enable your reader to scroll the table without affecting the rest of the text:

```
.fig-table {
  max-width: 100%;
  overflow-x: scroll;
}
```

It is important not to set a width on your table; the browser can then compress the table as far as it can before overflowing off the screen. To preserve readability, make good use of non-breaking spaces and `white-space: nowrap` to limit the amount the data wraps in the cells. It's better to have a readable table that requires scrolling than an unreadable one which doesn't.

Linearise simple tables into lists

You can safely linearise simple data tables when space is limited. The tables most suitable for this treatment are lists of structured data; for example, an employee directory:

Name	Email	Title	Phone
Jones, Claire	claire.jones@domain.com	Managing Director	01234 567890
Smith, Darren	darren.smith@domain.com	Head of Sales	01234 567891

When there is not enough room for the table to render comfortably, we can set it with a completely different layout. This is less compact overall, and takes more space vertically, but it succeeds in fitting the table into a much narrower viewport:

Name:	Jones, Claire
Email:	claire.jones@domain.com
Title:	Managing Director
Phone:	01234 567890
Name:	Smith, Darren
Email:	darren.smith@domain.com
Title:	Head of Sales
Phone:	01234 567891

The two renderings of our employee directory table use exactly the same markup, comprising the conventional HTML elements you would expect in any table. The one addition is a `data-title` attribute on each cell enabling us to repeat the label in the list view, should we need to.

```
<th data-title="Name">Jones, Claire</th>
<th data-title="Email">claire.jones@domain.com</th>
<th data-title="Title">Managing Director</th>
<th data-title="Phone">01234&nbsp;567890</th>
```

There are four simple steps to turning the table into a list, using a media query and css (no JavaScript is required).

1. Identify the viewport width at which the table starts to render poorly.
2. Apply `display: block` to all table-related elements so they align vertically instead of as a table.
3. Hide the header row and any empty cells.
4. Display labels for each data item (optional).

You will need to apply some additional styling for aesthetics and readability, but the responsiveness described can be accomplished in these few lines of CSS:

```
@media (max-width: 25em) {  
    table, caption, tbody, tr, th, td {  
        display: block;  
        text-align: left;  
    }  
    thead, th:empty, td:empty {  
        display: none;  
        visibility: hidden;  
    }  
    th[data-title]:before, td[data-title]:before {  
        content: attr(data-title) ":";  
        display: inline-block;  
        width: 3.5em;  
    }  
}
```

This technique was first popularised by Aaron Gustafson⁴.

Make tables responsive according to their purpose

There are many different techniques⁵ available for making data tables responsive. Some are simple CSS-only methods (we've covered two already); others are complex, enhanced by JavaScript. When considering which technique to use, ask yourself how your reader will use the table. In particular, consider if your reader is likely to compare either rows or columns – these kinds of tables need extra attention owing to the way they are used.

When being able to compare columns is important, one method is to hide non-essential fields and provide an option to turn them back on.

⁴ 'Responsive Tables' (<http://wbtyp.net/16>) by Aaron Gustafson on *Easy Designs blog* (2013).

⁵ See CSS-Tricks' 'Responsive Tables' (<http://wbtyp.net/148>) for the latest options.

This technique was popularised by Filament Group⁶ using a stocks table as an example:

Company	Last Trade	Trade Time	Change	Prev Close	Open	Bid	Ask	1y Target Est
GOOG Google Inc.	597.74	12:12PM	14.81 (2.54%)	582.93	597.95	597.73 x 100	597.91 x 300	731.10
AAPL Apple Inc.	378.94	12:22PM	5.74 (1.54%)	373.20	381.02	378.92 x 300	378.99 x 100	505.94
AMZN Amazon.com Inc.	191.55	12:23PM	3.16 (1.68%)	188.39	194.99	191.52 x 300	191.58 x 100	240.32
ORCL Oracle Corporation	31.15	12:44PM	1.41 (4.72%)	29.74	30.67	31.14 x 6500	31.15 x 3200	36.11
MSFT Microsoft Corporation	25.50	12:27PM	0.66 (2.67%)	24.84	25.37	25.50 x 71100	25.51 x 17800	31.50
CSCO Cisco Systems, Inc.	18.65	12:45PM	0.97 (5.49%)	17.68	18.23	18.65 x 10300	18.66 x 24000	21.12
YHOO Yahoo! Inc.	15.81	12:25PM	0.11 (0.67%)	15.70	15.94	15.79 x 6100	15.80 x 17000	18.16

A data-rich table rendered on a large screen.

Company	GOOG Google Inc.	18.65 0.97 (5.49%)
	AAPL Apple Inc.	
	AMZN Amazon.com Inc.	
	ORCL Oracle Corporation	
	MSFT Microsoft Corporation	
CSCO		

The same table with hidden columns and options to toggle.

Tables are a frequently overlooked aspect of reading, sometimes overstyled, sometimes poorly thought out. Responsiveness is a particularly thorny issue as the best solutions depend very much on the utility of the table. Tables can be packed with data, rich in content and meaning. Give them the attention they deserve.

⁶ Filament Group's 'Tablesaw' (<http://wbttyp.net/15>) responsive table plug-ins.

Tracking and kerning

Kerning and tracking are techniques for adjusting the spacing between letters. They are an important aspect of improving the legibility, readability and overall look of your text.

Kern your text to improve readability

Kerning is a local adjustment between two characters. Kern text to eliminate awkward spacing between two adjacent letters, reducing unsightliness and the potential for confusion. Although far easier with digital typography, kerning has been around since the days of moveable type.



Kerning in metal type (*reversed*). © Barbara Hauser

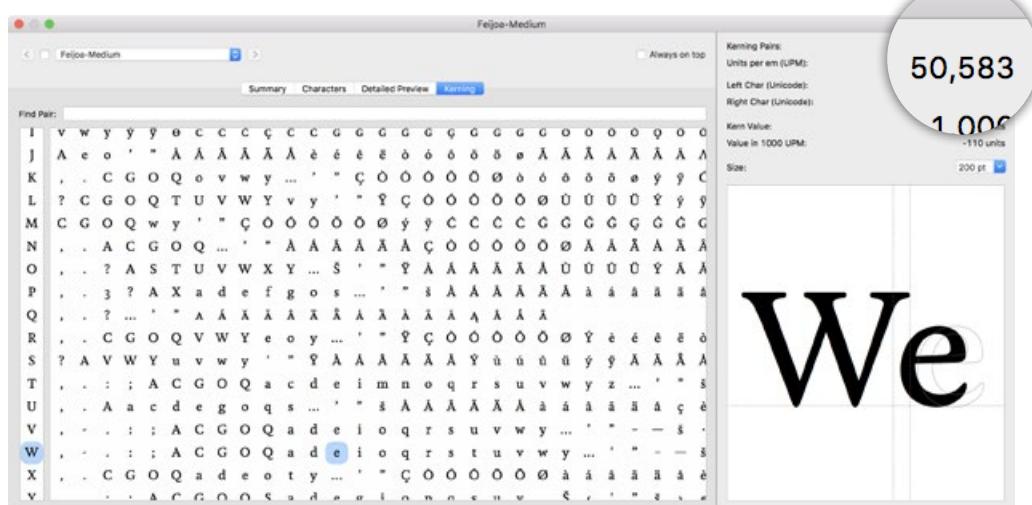
Kerning can mean the addition or reduction of space between two characters. Depending on the typeface design, different character pairs need different amounts of kerning. There are some critical character combinations which almost always need kerning; for example, capital W followed by a lowercase letter:

World Wide Web

World Wide Web

Without kerning, words like ‘World’ and ‘Web’ can be split asunder.

Inside font files, kerning instructions are stored in tables. These are large two-dimensional matrices which, for each character pair, show whether they should be drawn closer or pushed apart, and by how much. A modest typeface with around 250 characters might have 1,500 character pairs requiring kerning. A more fully featured font with 1,000 glyphs might have over 50,000 kerning pairs.



Kerning table showing Feijoa Medium with 50,583 kerning pairs.

Under the hood of a font, kerning tables are stored in one or two places. The glyph positioning (GPOS) table, which defines all manner of glyph adjustments, may include kerning instructions. Alternatively, the OpenType kern table may be used. The choice of where to store the information is down to the type designer or foundry and usually won't affect us, with one exception.

In the CSS3 fonts specification, kerning is turned on by default (and this is respected by all modern browsers). Should you want to explicitly

disable or re-enable kerning, use the `font-kerning` property as this works with both types of kerning data. (Browsers will use the data in the `GPOS` table in preference to the `kern` table.)

```
h1 { font-kerning: normal; } /* turn on kerning */  
  
footer { font-kerning: none; }
```

Be aware that if you use the `font` shorthand property then `font-kerning` is reset to its initial value – but because you can't set `font-kerning` with the `font` shorthand, you may have to restate the property.

Legacy browsers that don't support `font-kerning` can access kerning if there is an OpenType kern table present. This is a feature tag like any other, so you can use the `font-feature-settings` property to turn it on. The downside is that if you apply letter spacing to the text it will remove the kerning, whereas if you use `font-kerning` then the kerning will stay applied, as it should. Therefore, use a negated `@supports` rule so only legacy browsers get the problematic `font-feature-settings` property.

```
h1 { font-feature-settings: "kern" 1 }  
  
@supports (kerning: normal) {  
  h1 { font-feature-settings: normal; kerning: normal; }  
}
```

Not all fonts have kerning tables, including many free or low-quality fonts. In some cases, web fonts may have had kerning data removed to save on file size. If you're trying to turn on kerning and it doesn't seem to be working, that could be why.

Latterspace your text rarely and carefully

Tracking is a global change affecting many characters simultaneously to the same extent. Tracking, or letterspacing, is the tightening or loosening of spacing between the letters in a block of text. Unlike kerning, which affects only designated pairs of letters, tracking affects all the characters equally.

Adjusting tracking in the right circumstances can improve the readability in segments of text, just like a little sugar is sometimes necessary to enhance a sauce. And just as too much sugar will ruin a dish, so too much tracking adjustment can leave text illegible.

Don't letterspace lowercase without good cause

When there is sufficient white space around characters, the individual letters emerge and can be recognised by your reader more quickly. In a professionally designed font, you can expect correct spacing between characters to be built into the font and not need any further adjustment. If you have chosen a font for your body text and think it needs extra spacing (or possibly tightening), first of all question your choice of typeface. Perhaps it's not of the quality you were expecting – if the tracking is off, what else might be wrong? Ensure you are making a decision about *readability* and not at-a-glance aesthetics. As always, test your decision on multiple devices.

Latterspaced text, stretched beyond a relaxed body composition, looks affected and overstyled. Any font with a calligraphic link between letters, such as a Garamond or other old-style serif face, will be destroyed by tracking adjustments. As the type designer Frederic Goudy supposedly said¹:

Anyone who would letterspace
lowercase would steal sheep.

Anyone who would letterspace
lowercase would steal sheep.

The elegant flow of Calluna, a contemporary old-style font, is ruined by letterspacing.

Worse still, additional letterspacing may isolate individual letters and make it harder for your reader to recognise groups of letters or perceive whole words.

Latterspace all strings of capitals and all long strings of digits

Latterspace strings of capitals including WHOLE WORDS in the middle of sentences, but don't letterspace strings of initials. In instances such as Doogie Howser, M. D. and J K Rowling, slightly separate the initials using narrow no-break spaces as described in '[Symbols, signs and accents](#)'. It's not usually necessary to add extra spacing to small caps as the spacing is built in to most fonts.

¹ Most sources claim that Goudy referred to blackletter rather than lowercase, and that being stolen was the least of the sheep's worries!

Add tracking where you have three or more digits in a row, such as phone numbers and serial numbers. In both cases, add around 5% spacing using the `letter-spacing` property, combined with a positive value set in ems:

```
.caps { letter-spacing: 0.05em; } /* add 5% spacing */
```

For headings set in all capitals, try adding even more spacing, particularly if the heading is set in a lightweight font. If the spaces between letters are large enough to fit more letters, you've gone overboard.

FINE NORFOLK HAY

This is very fine, soft red fescue hay. It is harvested late to reduce the level of sugar and

FINE NORFOLK HAY

This is very fine, soft red fescue hay. It is harvested late to reduce the level of sugar and

Top: Kerned but not letterspaced; *Bottom:* Letterspaced caps.

Eliminate erroneous spaces after last letters

The CSS Text Module Level 3 specification says that ‘letter-spacing must not be applied at the beginning or at the end of a line’. At the time of writing, no web browser follows this – they all add letterspacing after each letter, including the last character of the selected text. For small fonts or left-aligned text this is not a problem and your reader will not notice the erroneous space. However, on large, centred text its presence can throw off the alignment of text. In these situations, you can offset the effect of the errant white space on the last letter by applying a negative right margin the same size as the letterspacing:

```
h1 {
  text-align: center;
  text-transform: uppercase;
  letter-spacing: 0.5em;
  margin-right: -0.5em;
}
```

Note that if you are justifying text, the letterspacing is added before the justification algorithm is applied.

Gently tighten big, bold and wide fonts

The bigger, bolder and wider the font, the tighter the tracking can be. By reducing the letterspacing a little, a more even colour can be achieved while keeping the letterforms distinct. In this example, the heading set in Value has been tightened by 3%:

```
h1 { letter-spacing: -0.03em; }
```



Fine Norfolk Hay
Fine Norfolk Hay

Tracking tightened by 3% improves colour.

Some sans serif text faces can also benefit from a little tightening. These are the exception rather than the rule, but the digital fonts of Univers, for example, come quite loosely tracked, as were their original metal forms. Univers can therefore accommodate some modest tightening of the letter spaces, even at body sizes.

```
p { letter-spacing: -0.01em; }
```

One more attribute the modern typographer must have: the capacity for taking great pains with seemingly unimportant detail. To them, one typographical point must be as important as one inch, and they must harden their heart against the accusation of being too fussy.

One more attribute the modern typographer must have: the capacity for taking great pains with seemingly unimportant detail. To them, one typographical point must be as important as one inch, and they must harden their heart against the accusation of being too fussy.

Tightening Univers by 1% gives a more contemporary feel.

Be very careful when tightening tracking. Do so judiciously and test extensively to ensure that you keep adequate spacing between letters, so that you don't make any unfortunate joins. For example, *cl* can become a *d*; *LI* can join to form a *U*; and *Fi* can end up as an *A*. These can cause embarrassment with words like *click*, *FLICKER* and *Final*.

Turn off ligatures if increasing letterspacing

According to the CSS3 Text module, browsers shouldn't apply ligatures when tracking text. At the time of writing, however, all browsers replace character pairs with ligatures regardless of letterspacing. Since a ligature is a single character, its internal spacing won't change when you tighten or open up the tracking, resulting in this kind of unevenness:



wharfingers
wharfingers

Ligatures are not required when letterspacing.

If you find it's necessary to turn off ligatures when letterspacing, apply `font-variant-ligatures` and its `font-feature-settings` fallback:

```
h1 {  
    letter-spacing: 0.1em;  
    font-feature-settings: "liga" 0;  
}  
@supports (font-variant-ligatures: no-common-ligatures) {  
    h1 {  
        font-feature-settings: normal;  
        font-variant-ligatures: no-common-ligatures;  
    }  
}
```

Alternatively, don't letterspace lowercase text any more than the tracking already built into the font. Remember what Goudy would think of you.

Headlines and impact

Typography can broadly be divided into two roles: novels and billboards¹. That is, immersion and interruption; reading and impact. Billboards are settings of *display* type used to grab the reader's attention. Provided it looks good, they can break many of the normal typographic rules but must still be handled with care and attention.

Headings and display text are your attention-grabbers. They set the scene and draw people in. Readers will reward the time you spent designing the typography with their attention.

Set text at display sizes, even on small screens

In ‘[Why web typography really does matter](#)’ we established how scientific studies have shown how good typography can improve your reader’s mood and overall experience without them even knowing it. If you set your display text at a large size, it stands to reason that you can have a bigger effect on your reader. When you set text big, by which we mean at least three times the body size, your visitors will *see* it before they *read* it. The initial impression of your text setting and choice of font will elicit a subconscious emotional reaction in your reader (see ‘[Choosing typefaces for display](#)’ for why and how), enabling you to set a mood for the content to follow. Consider large text the same way you might a photograph: a picture of type that creates an atmosphere and anchors your layout.

Movie posters, book covers and, in particular, magazine spreads all use this approach to great effect. When opened up, magazines are roughly the same size as a large display. Take a lead from their conventions and don’t

¹ A description coined by British designer [Jon Tan](#) (<http://wbtyp.net/146>).

be afraid to be generous with your text size. Don't forgo the opportunity to make the same initial impact on a screen that designers do in print.



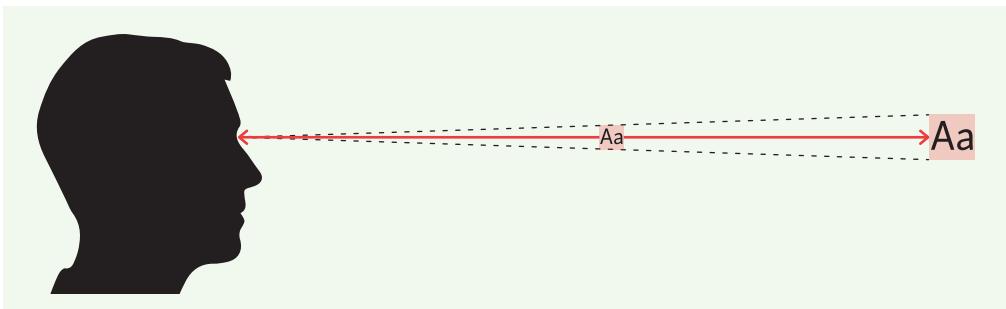
A magazine spread is similar in size and shape to a large screen.

You can clearly make an impression when you have a canvas as large as a 27" display, but can you really have that kind of impact on a screen so small it fits in your pocket? Yes, arguably so.

The magazine illustrated² has headline text set at 72 pt. To achieve the same physical size (roughly 25 mm) on a large screen, you would set your headline text to about 98 px. Given a typical viewing distance of 60 cm, we can calculate the perceived size of that type to be 145 arcmins (as introduced in '[Responsive paragraphs](#)').

Smartphones are read from much closer than desktop displays, typically 30 cm. Text viewed from that distance, having the same perceived size of 145 arcmins, would need to be around 13 mm on the screen. This equates to around 48 px, which is easily accommodated on a small screen.

² Issue 3 of [The Great Discontent](#) (<http://wbtyp.net/147>).



Same perceived text size at different viewing distances. *Image courtesy SizeCalc.com*

Because smartphones are held close to the face, filling your reader's field of vision, you can indeed set a mood and induce a reaction if you consider setting type as display text even on such a small device.

Resize display text as you would an image

As we discussed in '[Hierarchy and scale](#)', we choose text and heading sizes appropriate to the viewport dimensions from a predefined scale. We set those sizes using rem, locking the text sizes together so they all scale according to the page default and your reader's preferences. You can take the same approach with display text by choosing larger sizes from the same scale.

Display text, however, as defined by its purpose and relative size, is text to be *seen* first and *read* second; in other words, a *picture* of text. When it comes to pictures, you are likely to scale all scene-setting imagery – cover photos, hero images, and so on – relative to the viewport. Take the same approach with display text: lock the size and shape of the text to the screen or browser window.

Introducing viewport units

With CSS3 came a new set of units which are locked to the viewport. You can use these *viewport units* wherever you might otherwise use any other unit of length, such as pixels, ems or percentages. There are four viewport units, and in each case a value of 1 is equal to 1% of either the viewport width or height as reported in reference pixels:

- vw – viewport width
- vh – viewport height
- vmin – viewport height or width, whichever is smaller
- vmax – viewport height or width, whichever is larger

A reference pixel is based on the logical resolution of a device which takes into account double-density screens such as Retina displays.

In one fell swoop you can set the size of a display heading to be proportional to the screen or browser width, rather than choosing from a scale in a series of media queries. The following makes the heading's font size 13% of the viewport width:

```
h1 {  
    font-size: 13vw;  
}
```

For a selection of widths, the rendered font size would be:

Viewport width	13 vw
320	42
768	100
1,024	133
1,280	166
1,920	250

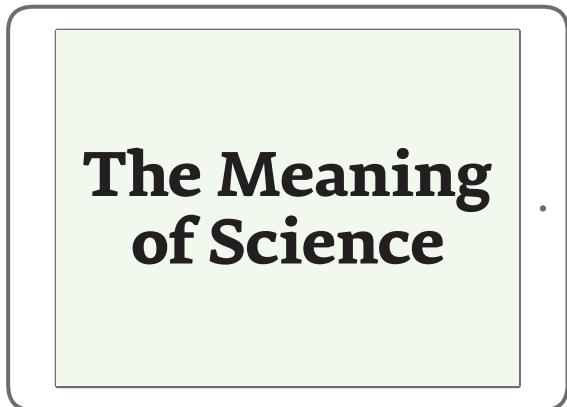
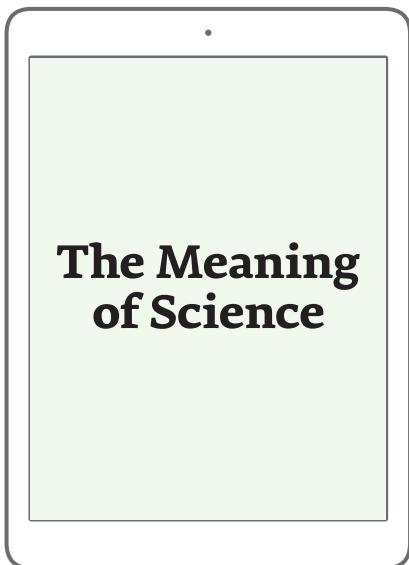
Rendered font size (px).

A problem with using `vw` in this manner is the difference in text block proportions between portrait and landscape devices. Because the font size is based on the viewport width, the text on a landscape display is far bigger than when rendered on the same device held in a portrait orientation.

The proportions of the display text relative to the screen are so dissimilar that each orientation has its own different character, losing the consistency and considered design you need when endeavouring to make an impression.

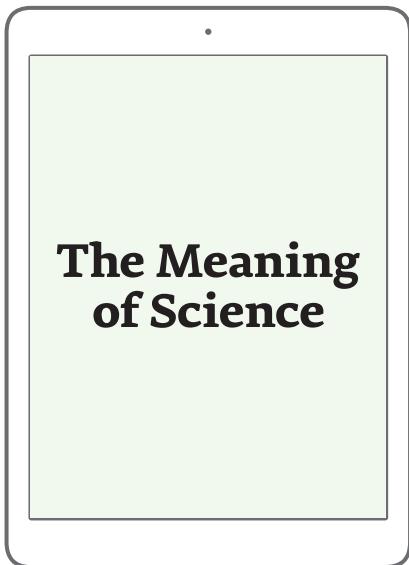
If the text was the same size in both orientations, the visual effect would be much more consistent. This is where `vmin` comes into its own. Set the font size using `vmin` and the size is now set as a proportion of the smallest side of the viewport, giving you a far more consistent rendering.

```
h1 {  
    font-size: 13vmin;  
}
```



vw units

Landscape text is much bigger than portrait text when using vw units.



vmin units

Landscape text is consistent with portrait text when using vmin units.

Comparing `vw` and `vmin` renderings for various common screen dimensions, you can see how using `vmin` keeps the text size down to a usable magnitude:

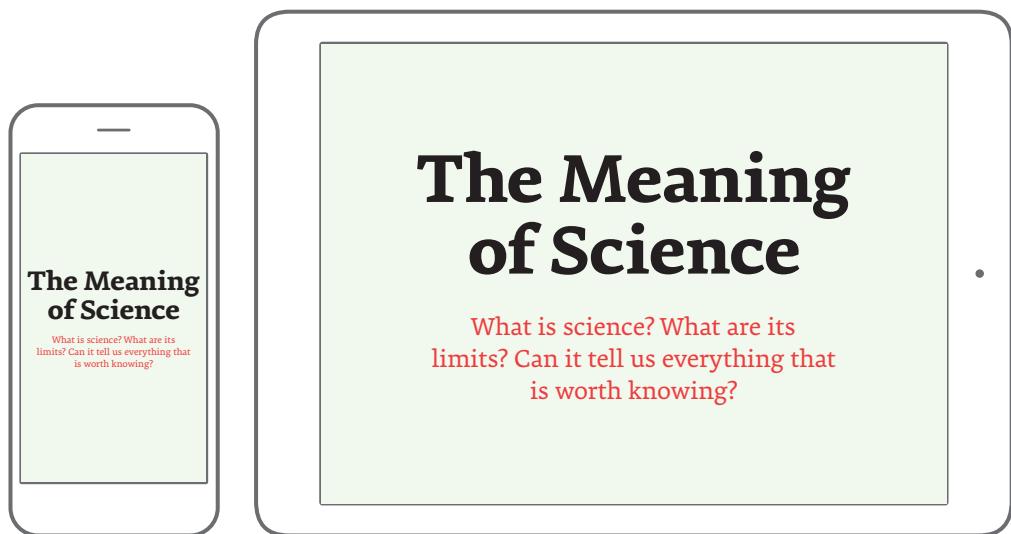
Viewport	13 vw	13 vmin
320 × 480	42	42
414 × 736	54	54
768 × 1024	100	100
1024 × 768	133	100
1280 × 720	166	94
1366 × 768	178	100
1440 × 900	187	117
1680 × 1050	218	137
1920 × 1080	250	140
2560 × 1440	333	187

Rendered font size (px).

Hybrid font sizing

Using vertical media queries to set text in direct proportion to screen dimensions works well when sizing display text. It can be less desirable when sizing supporting text such as subheadings, which you may not want to scale upwards at the same rate as the display text. For example, we can size a subheading using `vmin` so that it starts at 16px on smaller screens and scales up in the same way as the main heading:

```
h1 {  
  font-size: 13vmin;  
}  
  
h2 {  
  font-size: 5vmin;  
}
```



Using vmin alone for supporting text can scale it too quickly.

The balance of display text to supporting text on the phone works well, but the subheading text on the tablet, even though it has been increased in line with the main heading, is starting to feel disproportionately large and a little clumsy. This problem is only exacerbated on bigger screens.

A solution to this is to use a hybrid method of sizing text. We can use the CSS calc() function to calculate a font size based simultaneously on rem and viewport units. For example:

```
h2 {
  font-size: calc(0.5rem + 2.5vmin);
}
```

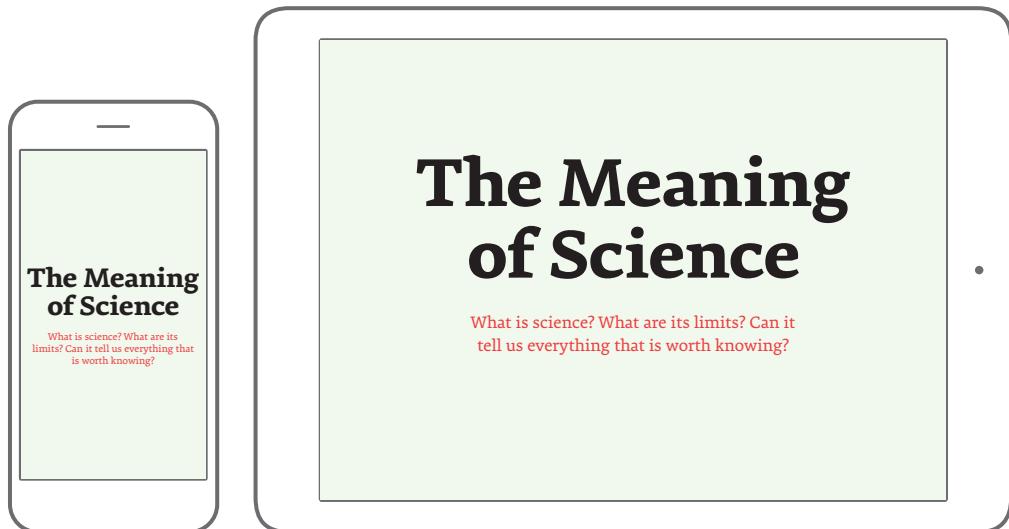
For a 320 px wide screen, the font size will be 16px, calculated as follows:

$$(0.5 \times 16) + (320 \times 0.025) = 8 + 8 = 16\text{px}$$

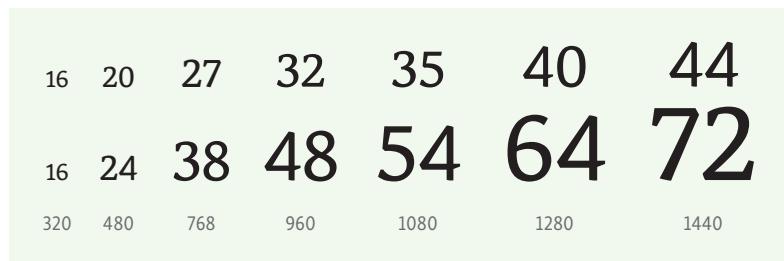
For a 768 px wide screen, the font size will be 27px:

$$(0.5 \times 16) + (768 \times 0.025) = 8 + 19 = 27\text{px}$$

This results in a more balanced subheading that doesn't take emphasis away from the main heading:



To give you an idea of the effect of using a hybrid approach, here's a side-by-side comparison of hybrid and viewport text sizing:



Top: calc() hybrid method; Bottom: vmin only

Experiment with the proportion of `rem` and `vmin` in your hybrid calculation to see what feels best for your particular setting.

Attend to wide, shallow screens

Sizing display text using `vmin` will always take into account both the width and the height of a viewport, and base the text size on the smallest dimension. If you are using a different method to size your display text – perhaps

you are picking sizes from a scale – then you must also account for the screen height. This applies not just to the text size, but to the white space which surrounds it.

For example, on large screens you may wish to set some substantial display text in generous white space, using something like:

```
@media screen and (min-width: 90em) {  
  h1 {  
    font-size: 6rem;  
    margin-top: 2em;  
    margin-bottom: 4em;  
  }  
}
```

But if that viewport is particularly shallow – it might be an unusually wide screen, or your reader has resized their window in such a way – then the heading may feel disproportionately large, with the added white space pushing the content too far down. To address this situation, add a vertical media expression, so that the large text and spacing is only rendered on screens which are both wide *and* tall enough. Given that most widescreen displays have aspect ratios of around 16:9, as a rule of thumb for your vertical media expression, specify the minimum height to be half the minimum width.

```
@media screen and (min-width: 90em) and (min-height: 45em) { ... }
```

You can additionally specify your white space using viewport units. This is a particularly beneficial tactic if your white space is substantial, as you can tie it to the viewport dimensions both stylistically and practically. For example, you can use rules like this to keep your vertical space relative to the viewport at all sizes:

```
h1 {  
  font-size: 2rem;  
  margin-top: 25vh;  
  margin-bottom: 1em;  
}  
  
@media screen and (min-width: 90em) and (min-height: 45em) {  
  h1 { font-size: 6rem; }  
}
```

Typesetting display text

Earlier we saw how display text works in the first instance as a picture setting an emotional tone, and is read after that first impression has been made. The picture-painting role of display text leaves you free to break rules, so long as you ensure the text is ultimately legible. That said, display text brings with it an additional set of considerations.

Pay attention to the shape of the text

True responsive design allows for fluid layouts, where lines of text inherently vary. Computers acting this way don't have a sense of what's good or bad – they can't make informed typographic decisions. Normally the rules you can provide in CSS will enable you to get body text set comfortably. Short passages of text provided for impact will still need a human hand.

Responsive design is not
about finding the
perfect design, it's about
finding the best
compromise

Default browser settings can be ugly.

If you are in the (fortunate) position to know the precise words you are setting, then consider where you break lines and the aesthetic shape of your arrangement, particularly when centring text, but also when aligning left or right. A simple guideline is to alternate line lengths as this produces a more considered and balanced setting.

Responsive design
is not about finding the
perfect design,
it's about finding the
best compromise

Responsive design is
not about finding
the perfect design, it's
about finding the
best compromise

Alternating text settings. *Left:* short, long. *Right:* long, short.

In the example above, both settings alternate shorter and longer lines. You'll consider one to be better than the other (I prefer short, long in this instance). The important thing is to control the setting if you can – don't leave such subtleties to the vagaries of computer algorithms. If you find yourself judiciously inserting `
`s to set the text, then check your fluid layout doesn't fall apart with unintended line wraps.

A useful tactic for width-sensitive text blocks is to set the width using character units, like this:

```
.example { width: 21ch; }
```

As explained in ‘[The amazing em \(and friends\)](#)’, one ch is equivalent to the width of the 0 (zero) character in the rendered font. This means the width of a text block will vary with the design of the font in such a way as to keep the line breaks consistent. This is particularly useful for handling fallback system fonts when you're using display text with very narrow or wide web fonts (see ‘[Using web fonts](#)’).

Visually centre text or hang punctuation

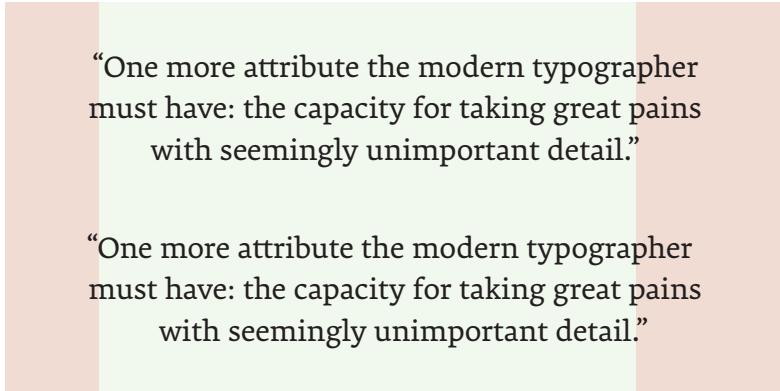
In ‘[Meaning and semantics](#)’ we saw that quoted passages of text with left or right alignment should have the punctuation marks in the margin. This ensures the start of each new line is aligned with the previous one, making the text block more pleasing to look at and to read. This requirement becomes even more apparent with text set at display sizes:

“One more attribute the modern typographer must have: the capacity for taking great pains with seemingly unimportant detail.”

“One more attribute the modern typographer must have: the capacity for taking great pains with seemingly unimportant detail.”

Top: Quote set as is. *Bottom:* Quote set with hanging punctuation.

If you are centre-aligning a text block with punctuation, the same thinking applies. Instead of hanging the quotes, visually align the text by setting it as if there were no punctuation marks, then add the punctuation back in.



“One more attribute the modern typographer must have: the capacity for taking great pains with seemingly unimportant detail.”

“One more attribute the modern typographer must have: the capacity for taking great pains with seemingly unimportant detail.”

Top: Quote set as is. *Bottom:* Quote set visually centred.

Whether centring text or hanging punctuation, you can handle the visual alignment by setting negative margins on the punctuation. For example:

```
blockquote {  
    text-align: center;  
    position: relative;  
}  
  
blockquote::before {  
    content: '“';  
    margin-left: -0.6ch;  
}  
  
blockquote::after {  
    content: ‘.”’;  
    margin-right: -1.1ch;  
}
```

It's worth restating that you should be extra careful to punctuate correctly when setting text at display sizes. Erroneous use of hyphens or neutral quotes will show up even more with big text.

Prevent widows in headings

A widow is a single word which drops onto the final line of a heading (or any other block of text). Unless you have specifically designed it to be there, widows tend to look unsightly. Use a non-breaking space between the final two words of a heading to prevent widows dropping down.

Adjust line spacing for evenness of colour

In ‘[Tracking and kerning](#)’ we learned that bold fonts require tighter tracking. The same is true of their line spacing, particularly at large sizes. Simpler geometric sans serif typefaces also benefit from tighter line spacing. Setting solid using `line-height:1;` is a good starting point. Lightweight, thin fonts need a touch more spacing, just as more ornate serifed fonts need room to breathe and strut their stuff.

If you are setting in all caps, then you will probably need to set the line height to less than 1 (try 0.75 to begin with, and experiment). Be wary, though, as capital letters have many vertical strokes. Set lines of caps too close and the verticals start to resemble a page of prison bars.

**Give me twenty six
soldiers of lead and I
will conquer the world**

Give me twenty six
soldiers of lead and I
will conquer the world

Calypso E Heavy and Light require tighter and looser line spacing respectively.

When you tighten line height to 1 or less, you increase the chance of clash, as introduced in ‘[Meaning and semantics](#)’. A simple rule is that ascenders and descenders should not touch, but that rule can be suspended if aesthetics dictate. Sometimes you may wish to set very tight line spacing for stylistic reasons. Perhaps the letters nicely interweave and create a harmonious, readable effect, or perhaps the text calls for deliberately obscured reading.

All rules are there to be broken if – and only if – the text calls for it and the picture you are painting benefits.

**Intimate
touching is
allowed
if it looks
better. It
increases
visual
strength &
firmness.**

Erik Spiekermann on line spacing in *Rhyme and Reason*, set in Spike Sondike's Blenny.

Make the most of a font's hidden gems

Many typefaces include features enabling you to be even more creative and expressive in your typesetting. These hidden gems are accessible through OpenType technology (introduced in '[Ligatures and abbreviations](#)') and include specially designed optional letterforms such as swashes, ligatures and alternates. These can bring your display text to life in unique, unusual and beautiful ways.

Swashes

Use swashes to add an air of sophistication to your text. They provide a typographic flourish, by way of flamboyant additions to a character, such as an exaggerated serif, tail or entry stroke.



Trilogy Fatface is made even more flamboyant with contextual swashes.

Turn on swashes using the `font-variant-alternates` property with a value of `swash(1)`.

```
.swsh { font-variant-alternates: swash(1); }
```

Some fonts have more than one swash design for certain characters. These are stored in numbered tables which you access using `swash(2)`, for example. To provide support for legacy browsers, use the `swsh` OpenType tag with the `font-feature-settings` property.

```
.swsh {
  font-feature-settings: "swsh" 1;
  @supports (font-variant-alternates: swash(1)) {
    .swsh {
      font-feature-settings: normal;
      font-variant-alternates: swash(1);
    }
  }
}
```

Trilogy Fatface (featured above) makes extensive use of *contextual* swashes. Its designer, Jeremy Tankard, programmed the font to swap in swashes which are appropriate for the start, middle or end of words, thus avoiding awkward clashes or gaps. Typefaces with this attention to detail are unusual.

Using OpenType features often requires a certain amount of manual intervention. You will usually need to mark up each of the characters you swap out to target precisely the changes you need. If you don't, and the font you're using is loaded with alternates – Bookmania in the following example has swashes for all the letters in *Margarita* – you would get an unappetising mess instead of a delicious concoction.

```
<h1><span class="swsh">M</span>ar<span class="swsh">g
</span>arit<span class="swsh">a</span></h1>
```



Margarita
Margarita

Just a few of the multitude of swashes in Bookmania.

Discretionary and historical ligatures

We touched on discretionary and historical alternates in ‘[Ligatures and abbreviations](#)’ and were warned against using them in running text. In display text, however, they can capture the attention effectively, lending a glamorous, bygone or unique feel.



Lucky giant feasts
BLACK SYMBOLS

Using discretionary ligatures in Salomé and Bague Sans to lend antiquity and glamour respectively.

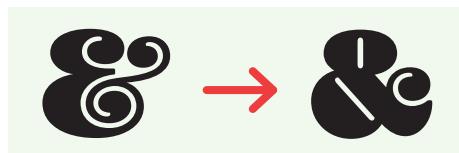
You can turn on discretionary ligatures and historical ligatures by using the `font-variant-ligatures` property:

```
.dlig { font-variant-ligatures: discretionary-ligatures; }
.hist { font-variant-ligatures: historical-ligatures; }
```

For legacy browsers requiring `font-feature-settings`, use the `dlig` and `hist` OpenType feature tags.

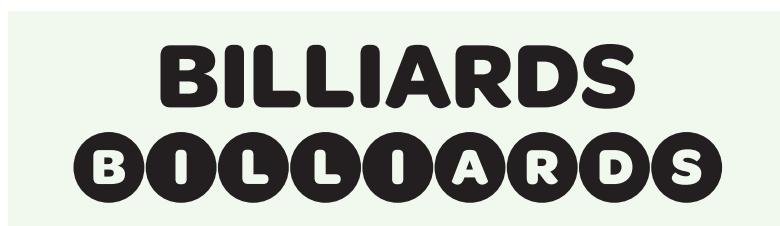
Stylistic alternates and style sets

Many fonts contain alternative glyphs that don't fit into a clear category like swash or historical. You can select these alternates on an individual basis using `font-variant-alternates:stylistic(1)` where 1 is the feature number. It is equivalent to the `salt(1)` OpenType tag.



Blenny has an alternative ampersand.

Some fonts also contain entire sets of stylistic variants, such as multiple variants for all or some of the lowercase letters. These stylistic sets are often designed to harmonise visually or interact in particular ways. These sets are also numbered, and accessible using `font-variant-alternates:styleset(1)`, equivalent to the `ss01` OpenType tag.



Enclosed caps as style set 6 in Omnes Bold.

You can turn on multiple style sets by space-separating the values:

```
.ss03_04 { font-variant-alternates: styleset(3) styleset(4); }
```

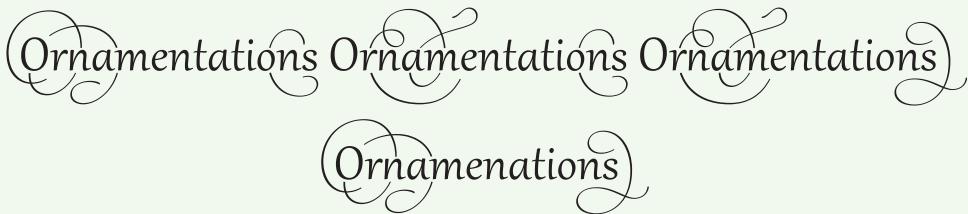


Alternative *l*, *g*, *y* and *a* turned on with style sets 3 and 4 in Omnes.

Gabriola, designed by John Hudson for Microsoft, has contextual awareness built into the alternative styles, ensuring that calligraphic features are applied intelligently. Gabriola has seven style sets defined. Turning on style set 7 with support for legacy browsers:

```
.ss07 {  
    font-feature-settings: "ss07" 1;  
    @supports (font-variant-alternates: styleset(7)) {  
        .ss07 {  
            font-feature-settings: normal;  
            font-variant-alternates: styleset(7);  
        }  
    }  
}
```

Ornamentations



Turning on style set 7 in Gabriola displays contextual calligraphic ornaments. These automatically adapt to the word's position in the line.

Seek out OpenType features

OpenType features are buried within font files and eminently undiscoverable. At the time of writing, no software adequately exposes the features available within a given font file. Some font management and design software will tell you whether a feature, such as swash alternates, is available in a font, but won't tell you which characters will be swapped out, let alone what the alternates actually look like.

Use this to your advantage. Visit the font designer's website and find the samples and specimens any good foundry will have created for its fonts. These should show you all the alternative characters available and how to access them. Use what you find to create a truly unique design with features that most designers simply will not have put in the effort to find.

Accessing OpenType alternates in CSS

We've covered the more common kinds of aesthetic alternative characters in this section. As you delve deeper into fonts you may come across other kinds. The full list of features accessible through `font-variant-` as defined in CSS3 is:

Feature	CSS property	OpenType tag
common ligatures	<code>font-variant-ligatures: common-ligatures</code>	<code>liga</code>
discretionary ligatures	<code>font-variant-ligatures: discretionary-ligatures</code>	<code>dlig</code>
historical ligatures	<code>font-variant-ligatures: historical-ligatures</code>	<code>hist</code>
contextual alternates	<code>font-variant-ligatures: contextual</code>	<code>calt</code>
historical forms	<code>font-variant-alternates: historical-forms</code>	<code>hist</code>
stylistic alternates	<code>font-variant-alternates: stylistic(n)</code>	<code>salt n</code>
style sets	<code>font-variant-alternates: styleset(1-99)</code>	<code>ss01 - ss99</code>
character variants	<code>font-variant-alternates: character-variant(1-99)</code>	<code>cv01 - cv99</code>
swash glyphs	<code>font-variant-alternates: swash(n)</code>	<code>swsh n, cswh n</code>
ornaments	<code>font-variant-alternates: ornaments(n)</code>	<code>ornm n</code>
annotation forms	<code>font-variant-alternates: annotation(n)</code>	<code>nalt n</code>

We opened the chapter by describing how display type is more akin to a billboard than a novel. On the web, good design is firstly about making people want to read, then about telling stories. We use display type, set well, with perhaps a unique twist, to grab our reader's attention, take them by the hand and lead them into the stories we are telling.

Applying vertical rhythm

Just as regular use of time provides rhythm in music, so regular use of space provides rhythm in typography. Without rhythm, the listener or reader becomes disorientated and lost.

Apply vertical space in measured intervals

The vertical rhythm of a design is formed by the arrangement and spacing of text encountered by your reader as they descend the page. If spacing between lines and blocks is inconsistent, the rhythm will be inconsistent, leading at best to an untidy feel to the composition, and at worst a jarring reading experience.

Composing to a vertical rhythm helps engage and guide the reader down the page, but it takes typographic discipline to do so. It may seem like a lot of fiddly maths is involved (a few divisions and multiplications never hurt anyone) but good typesetting is all about numbers, and it is this attention to detail which is the key to success.

You set the vertical rhythm for a page when you decide on the line height for your body text. If you have 16px body text with a line height of 21px then your basic unit of vertical space is also 21px. To establish a vertical rhythm on the page, ensure that all vertical spacing, including margins and the line heights of other text elements, is a factor of 21. A solid, dependable vertical rhythm will engage and guide your reader down the page and through the text.

Rhythm is commonly lost in the spacing between paragraphs. By default, browsers insert a top- and bottom-margin of 1em. In our case, this would add 16px between paragraphs and throw the text out of the beat of the page. To maintain rhythm, space the paragraphs apart by an amount related to the basic line height. Simply set the bottom margin equal to the line height.

```

body {
  font-size: 1rem;      /* 16px */
  line-height: 1.3125; /* 21px */
}

p {
  margin-top: 0;
  margin-bottom: 1.3125em; /* 21px */
}

```

If you prefer your paragraph spacing tighter, try making it equivalent to half the line height instead.

Use a basic reset to ensure consistency

Browsers set their own margins on all block-level elements including headings, lists, block quotes and tables. Apply a basic reset to all block-level elements, removing their margins and padding, and providing a solid foundation on which you can explicitly declare your design intentions.

The following will cover most bases:

```

html, body, section, nav, article, aside, h1, h2, h3,
h4, h5, h6, hgroup, header, footer, address, p, hr, pre,
blockquote, ol, ul, li, dl, dt, dd, figure, figcaption,
div, table, caption, form, fieldset {
  vertical-align: baseline;
  margin: 0;
  padding: 0;
}

```

When there is a change in text size, perhaps with a heading or caption, the differing text should also take up a multiple of the basic leading. This means that every diversion from the basic text size should take up multiples of 21px. You can accomplish this by adjusting the line height and margins accordingly.

Let's say you require your subheadings to be 18px. In order that the height of each line is 21px, set your line-height to:

$$21 \div 18 = 1.167$$

Using the same line height with the bigger text of a heading means the type will be set relatively tighter than the body text if the heading wraps. However, a bold heading wrapped over two lines will benefit rather than suffer from this.

To keep the rhythm going you should adjust the margins above and below the heading to fit. Set the margins to 1.167em, the same as the line height, so that they equal the full 21px vertical rhythm unit. *Don't be tempted to use line-height to add significant white space around headings – if your heading wraps you will end up with what looks like two headings.*

```
h2 {  
    font-size: 1.125rem;      /* 18px */  
    line-height: 1.167;        /* 21px */  
    margin-top: 1.167em;       /* 21px */  
    margin-bottom: 1.167em;     /* 21px */  
}
```

You can also set asymmetrical margins for headings, provided the margins combine to be multiples of the basic line height. For instance, you could widen the space between a heading and the preceding content, and tighten the separation from the following text. In this example, the top and bottom margins are one and a half times and half the vertical rhythm respectively:

```
h2 {  
    font-size: 1.125rem;      /* 18px */  
    line-height: 1.167;        /* 21px */  
    margin-top: 1.75em;        /* 31.5px */  
    margin-bottom: 0.583em;     /* 10.5px */  
}
```

Side notes and other supplementary material such as captions are often set at a smaller size to the basic text. To keep small text at the correct rhythm, use a calculation similar to that for headings. Let's say your side notes are set at 14px. Their line height must equal 21px, so set your line height to 1.5:

```
aside.note {  
    font-size: 0.875rem; /* 14px */  
    line-height: 1.5;      /* 21px */  
}
```

Aboard Minerva off the Coast of New England

Daniel is roused by a rooster on the forecastledeck[†] that is growing certain it's not just imagining that light in the eastern sky. Unfortunately the eastern sky is off to port this morning. Yesterday it was starboard. Minerva has been sailing up and down the New England coast for the better part of a fortnight, trying to catch a wind that will decisively take her out into deep water, or "off soundings," as they say. They are probably not more than fifty miles away from Boston.

[†] The forecastle deck is the short deck that, towards the ship's bow, is built above the upperdeck.

Contrary Winds

Daniel goes back and sits by one of the windows – these are undershot so that he can look straight down and see Minerva's wake being born in a foamy collision around the rudder. He opens a small hatch below a window and drops out a Fahrenheit thermometer on a string. It is the very latest in temperature measurement from Europe – Enoch presented it to him as a sort of party favour. He lets it bounce through the surf for a few minutes, then hauls it in and takes a reading.

He's been trying to perform this ritual every four hours – the objective being to see if there's any rumour that the North Atlantic is striped with currents of warm water.

Vertical rhythm maintained with headings and side notes.

Your side matter might actually be better off with a different line height from the body text. Such a narrow measure warrants lines set closer together; this way you will also minimise the risk of your body text and marginalia being read as if they are a continuous line of text. If you reduce the line height of your aside, you can still do so with respect to the overall page rhythm. Decreasing the line height to 18px, for example, will mean the side matter rejoins the rhythm every seven lines. That will rarely make a difference to your reader, unless it's a particularly long side note, but it does keep your typographic system thoughtful and intact.

Allow embedded media to break the rhythm

Illustrations, videos, tables, forms and other interruptions create syncopations and variations against the base rhythm of the page. These variations can and should add life to your composition, but there is no particular reason why the main text should return precisely in phase.

For that, you would need to crop images just so they fit the rhythm. Any attempt at responsive design would render this approach extremely difficult, if not impossible – variable-width columns tend to lead to images with variable widths and heights. But, more to the point, the tail would wag the dog. Images which are *content* rather than *decoration* can be cropped, but only to convey the right message, not so they fit into a mathematical arrangement.

Vertical rhythm is most apparent in running text. Interrupting that text with embedded media means you can simply restart the beat when the text recommences.

Don't slavishly follow a baseline grid

The baseline is an imaginary line on which the letters in a font appear to rest. In the world of print design, the basic unit of vertical rhythm is measured from baseline to baseline, and blocks of text are arranged and aligned to baselines.

The web is slightly different. In browsers, text is centred within a text block. You can adjust the text blocks precisely and accurately, but the browser knows very little about the baseline¹, unlike software such as InDesign, which offers configurable settings, allowing us to force text back onto a baseline grid if it's thrown off by an inserted element.

To have all text sit on a common baseline in a browser takes an awful lot of jiggery-pokery², but having little concept of a baseline does not preclude a page from having a vertical rhythm and the benefits one can provide. The way web pages are laid out by browsers is like upside-down Tetris. Blocks stack downwards from the top of the page, and text fills those blocks. They are not built from the text outwards, which is why we don't have an innate baseline grid to work with. Forcing all type to a baseline grid is to pretend the web is something it isn't: that is, print. Vertical rhythm is not precisely the same as a baseline grid, but provides all the consistency and regular downward motion required for your reader.

One exception is when you want to position blocks of differently sized text next to each other and have the baselines align. The alignment is not strictly necessary but can be more pleasing to the eye. Because text sits in

¹ CSS has an increasing number of ways to align baselines. All it had initially was `vertical-align:baseline` for sitting inline images on the baseline. More recently [Flexbox and Grid Layout modules](http://wbtyp.net/7) (<http://wbtyp.net/7>) have introduced ways to align adjacent blocks to baselines.

² Tom Bredin-Grey had much to consider and calculate when writing [MegaType](http://wbtyp.net/68) (<http://wbtyp.net/68>) as a baseline alignment project.

the middle of a text block rather than on the baseline, you will normally need to nudge one of the blocks up or down to get the alignment you crave.

Baseline grids are, ultimately, most useful in print, and their traits show up best with opposing pages, and when light shines through paper as you are reading; the paper's translucency reveals a shadow of the type on the other side of the page. Aligning the text on both sides of the paper prevents misaligned baselines distracting the reading experience. We don't have the same problem with screens.

Arrangement and composition

Layout is intrinsically connected to typography. Not everything about layout has to do with the text, but as a typographer it is your task to organise and arrange the textual matter in such a way that your reader will understand how the different elements on a page relate to one another, so they will have a good chance of finding what is of interest to them.

The field of typography concerned with layout is *macrotypography*. This incorporates many aspects of what is frequently called information design, as part of an overall user experience design. There are many fine books¹ written specifically about layout, most of which are far bigger than the one you are currently reading about type. This lone chapter cannot go into great detail about all aspects of layout, though it does cover important strategies for your overall page structure and composition.

Design layouts from the content out

In 1928, the German typographer Jan Tschichold, later to become chief designer at Penguin Books, published his seminal manifesto: *Die neue Typographie*². In it he wrote:

Though largely forgotten today, methods and rules upon which it is impossible to improve have been developed for centuries. To produce perfect books these rules have to be brought to life.

¹ In particular, *Grid Systems in Graphic Design* by Josef Müller-Brockmann, and *The Typographic Grid* by Hans Rudolf Bosshard.

² *The New Typography* by Jan Tschichold (1995, after 1928 original).

What Tschichold refers to are finely honed geometric systems that create beautiful text for books and other printed publications. Binding content to the physical form is what good book designers do; but with all paper-based design, they start with paper. Paper that has edges, ratios that can be repeated. A physical canvas. And that's where Tschichold's assertion falls down. The 'rules and methods upon which it is impossible to improve' no longer apply. We've entered an era that must challenge the design and layout theory that has existed for over 1,000 years. We have no predefined canvas on which to paint our beautiful geometry – the outside-in approach to layout no longer applies. Instead, we have content that we need to apply to whatever canvas – that is, whatever screen – we are presented with by our reader.

As web designers we don't have the opportunity to choose or know in advance the canvas size or stock. Often we don't even know the text or imagery we are designing for. Somehow we have to set rules to lay out and typeset unknown content in an unknown medium. For all the precision that computers can provide, we have to make estimates, guesses, accommodations and compromises.

The rules and methods Tschichold referred to are not entirely defunct, but we need to re-evaluate them and apply what we can in our modern fluid context where the medium is in the hands of the user (often, but not always, literally). By applying rules of contrast, repetition, alignment and proximity we can design from the content outwards, and use proportions to have layouts that adapt themselves to their surroundings.

Adopt a mobile-first philosophy

Designing for small screens does not leave you much scope for layout. You have only one dimension in which to set text: downwards. A single, narrow column, which your reader perceives bit by bit as if peering through a key-hole, is a challenging environment to design for, which is why it should be considered up front. Because your reader cannot usually see much of the page on a mobile device, and because the content is stacked rather than arranged, success relies heavily on prioritisation and hierarchy.

Thinking mobile first is a design *philosophy*, and a very useful one at that. Being forced to consider a linear priority has the fortunate side effect of benefitting all potential layouts. Your client can no longer get away with saying one block is of equal importance to another – something has to come first – and that's exactly the kind of information you need to know as a designer.

Think mobile first by having small screens at the forefront of your mind when designing, and consider the likes of desktop monitors as an enhancement. You or your colleagues may have a development process where you build mobile first, but *thinking* mobile first doesn't have to mean *designing* mobile first. The two extremes need to coexist, so neither phone nor desktop (nor anything in between) should be an afterthought. All screens should be an intrinsic part of your design thinking and process.

Avoid designing different layouts for specific devices: this is a Sisyphean task and will lull you into a false sense of security. Allow your designs to take on the shape of water; they should continuously adjust to the vessel into which they are poured. When layouts of elements break (hence *break-point*), adapt the design accordingly. This is true responsive design and provides the most flexible (in both senses) and long-lasting approach to design. For convenience and maintainability it is fine to group your adaptations into batches at predictable sizes (small, medium, large, giant), but think about those from a modular point of view. You are enhancing layouts to better fit a vessel; you are not designing for a given device.

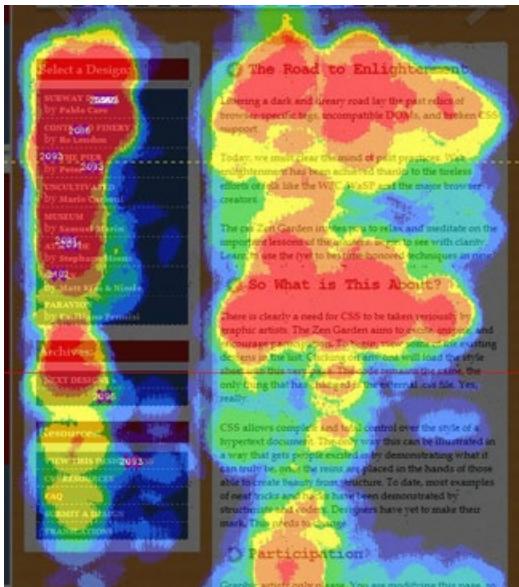
Stage the main text block for scanning

In '[How we read](#)' we learned that people *scan with purpose*, jumping from section to section, looking for a particular piece of information. They *skim read*, casting their eye over text, reading words and sentences here and there to get a sense of the content. Ultimately (we hope), people read linearly in an *engaged* manner once they find a passage or entire page they are interested in.

Everything you've read so far in this book has been about the up-close detail of text and attention to the linear reading experience. When you sit back and consider a page as a designer, you move away from the lean-in typographic perspective and take on a graphic designer's-eye view. The graphic designer's role is to arrange text blocks, imagery and other elements of the page in ways which make immediate sense to the beholder. Your job is to convert the visitor from a skimmer into a reader, but to do that you must accept that all potential readers will scan first, and design the page for that scenario.

The way people scan is well documented and informative. On larger screens, people tend to scan with fixations forming an F-pattern³.

³ '[F-Shaped Pattern For Reading Web Content](#)' (<http://wbtyp.net/25>) by Jakob Nielsen on *Nielsen-Norman Group* (2006).



Eye-tracking heatmap showing F-pattern reading.

At first they read in a horizontal movement; this would normally be the main heading, standfirst or lede. This initial pass forms the top bar of the F.

If the opening hasn't grabbed their attention straight away, visitors then move down the page and read in a second horizontal movement, typically shorter than the initial movement. This forms the F's middle bar.

Finally, your visitors will work their way down the left side of the content in a vertical movement forming the stem of the F. Sometimes this can be fairly slow and meticulous; other times users move faster, skipping from chunk to chunk. This F-shaped behaviour leaves us with three conclusions to take into our design:

1. The overall reading experience must be as inviting as possible because it competes with the momentum of scanning.
2. Subheadings must be clearly demarcated and easy to latch on to.
3. There must be a clear left-hand edge, giving your visitors' eyes a visual bannister to hold on to as they descend the page.

Bring the design into the margins

Margins frame the text block and provide a guide rail for your reader, whether they are immersed in the text or scanning the page. When the margins are sufficiently capacious you can use them to house side notes,

captions or pull-quotes, and as a means of placing images and tables which are wider than the measure of the body text.

For example, you might have an article containing some paragraphs and an aside marked up simply like this:

```
<article>
  <p>Text...</p>
  <aside>Note...</aside>
  <p>More text...</p>
</article>
```

You would start off with the text block centred nicely on the page, restricted in width to a comfortable measure:

```
article {
  max-width: 38em;
  margin-left: auto;
  margin-right: auto;
}

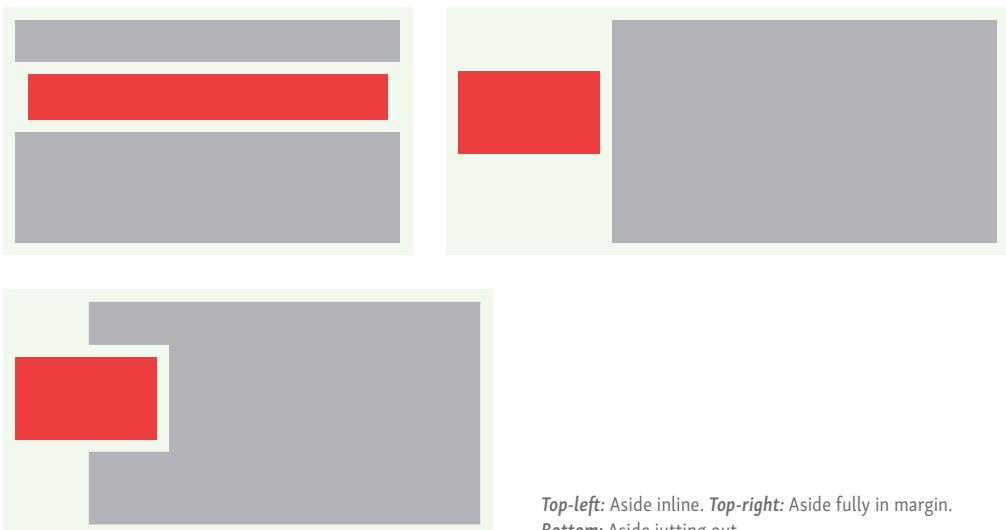
aside {
  margin: 1em;
}
```

When there is insufficient room in the margins, the aside runs inline with the column. When the page is wide enough, you can float the aside into the margin. In this example we have a side note sized at 10 em wide, with 1em gutter between the aside and the text block, meaning the page needs to have an 11 em margin either side of the central column, leaving us with a minimum width of 60 em ($38+11+11=60$):

```
@media only screen and (min-width: 60em) {
  aside {
    width: 10em;
    float: left;
    margin-left: -11em;
    margin-top: 0;
  }
}
```

For screens somewhere between the two extremes, you can situate the aside halfway into the margin:

```
@media only screen and (min-width: 49em) {
  aside {
    width: 10em;
    float: left;
    margin-left: -5em;
    margin-top: 0;
  }
}
```



Top-left: Aside inline. *Top-right:* Aside fully in margin.
Bottom: Aside jutting out.

Use active white space

White space is the space between elements in a composition. White space frames the content and provides structure to the page. White space is an indispensable factor not only in giving the eye room to read, but also in creating a welcoming environment. Without white space your page will feel too cluttered (because it is).

The practical value of white space towers over its value as a design element. A page with generous amounts of white space might look nicer, but it's not about taste; it's a welcome side effect: the logical consequence of functional design.

White space doesn't have to be white, which is why it's also known as negative space.

The spacing between words and lines – the result of competent typesetting – is known as *passive* white space. When you use white space to frame text blocks, highlight elements and lead your reader from one aspect to another, it's called *active* white space.

Use active white space to surround your text with enough air to breathe. This reduces stress levels when there are other elements on the page, as it makes it much easier for your reader to focus on the core content.

The more white space you surround an element with, the greater the emphasis you give it. In the previous example we indented the aside. The spacing we gave it – the active white space – reduced its impact on the text block as a downward reading experience, but also distinctly separated the note from the surrounding prose.

Adjust the spacing within the text block using typographic increments, but rely on free proportions to adjust the empty space. In other words, use ems to set margins in running text, but use percentages or viewport units to tie the surrounding white space and overall layout to the dimensions and proportions of the browser window (see ‘[Headlines and impact](#)’ for more details on viewport units).

Show relationships and draw attention

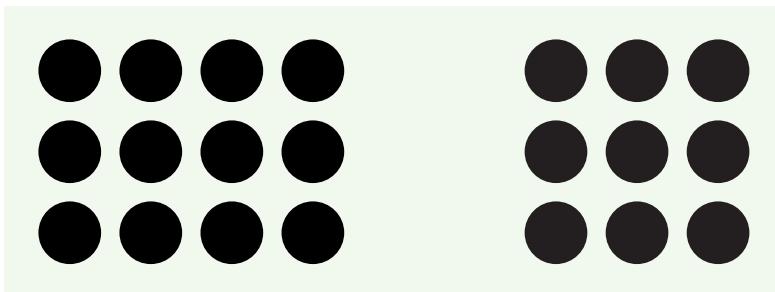
The sparsest of content can be simply accommodated in a block in the middle of the screen, or perhaps positioned to the left, comfortably offset from the edge of the browser window. Pages with more forms of content will need additional consideration. On a small screen your main text block will be preceded, interrupted or followed by ancillary accompaniments such as asides, imagery, data visualisation, notes, related links and advertising. For larger screens these are blocks of content floating around waiting for a home.

Any subsidiary and supporting elements will need to be placed in context, grabbing just the right amount of attention, and associating themselves with the right neighbours. All this can be achieved through the application of some *gestalt* principles.

The German word *Gestalt* can be translated as shape or form, and the term refers to how visual input is perceived by human beings. Our view of one thing can be altered by the things around it. Once we see the whole, our perception of the individual parts changes. Conversely, once we can see all the parts, our understanding of the whole is altered. This is what gestalt theorists call field theory. The field is the holistic view of the world, or, in this case, the full design. In gestalt theory, you get to the best outcomes

by adjusting all the attributes in the field and not worrying about an individual one. The whole is not greater than the sum of the parts, it is *other* than the sum of the parts.

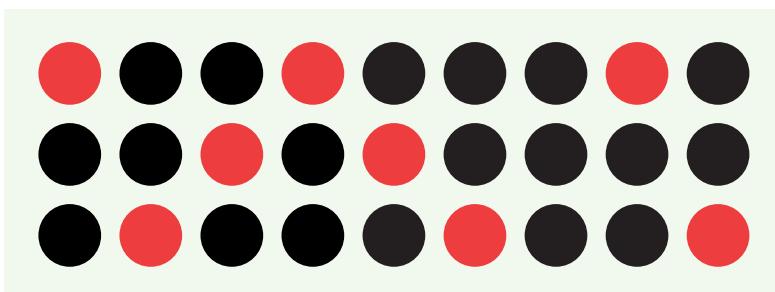
In active white space we have already come across one of the gestalt principles: proximity.



Proximity.

When you position elements close to one another, they are seen as part of a group rather than as individual elements. This is an obvious way to associate one thing with another; for example, placing a side note next to the text it supplements.

The opposite effect is useful too. By increasing the spacing around an element you decrease the proximity to its neighbours, setting it apart contextually and emphatically.

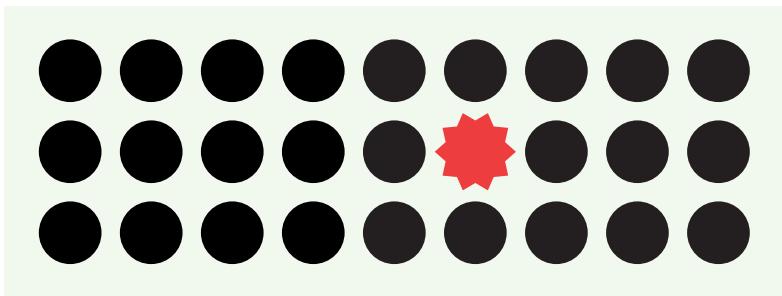


Similarity.

Another principle is similarity. Elements sharing similar characteristics are perceived as more related than elements that don't share those characteristics.

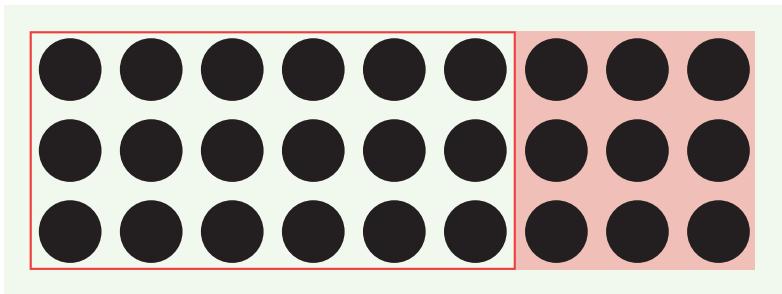
Any number of characteristics can be similar: color, shape, size, texture, and so on. When your reader sees these similar characteristics, they

perceive the elements as being related. Links the same colour will have the same meaning (unvisited, visited). Headings the same size will be at the same level in the hierarchy. All small text in a sans serif will be deemed related; and small text in a serif will be assumed to be related but different to a sans.



Focal points.

A variation on similarity is focal points, as they can't be seen without the presence of similarity among nearby elements. Elements with a point of interest, emphasis or difference will capture your reader's attention. A classic focal point would be a dropped cap. Other examples include a thumbnail image, icon, or symbol, such as a manicule ↗, section §, ornamental quote « or fleuron ♀. Any small device which draws attention is a focal point, bearing in mind that many such devices on one page compete with one another and revert to similarity rather than focus.



Common regions.

Elements are perceived as part of a group if they are located within the same closed region. Drawing a line around things groups them together. Placing the elements on a different background colour will also work.

Like all of the gestalt principles, this seems obvious, but by thinking about them consciously and with reference to the whole page you can come up with variations and options which offer more effective solutions when considered together. For example, you may have a white page and group together some related links on a darker background. The colour may clash with or confuse some imagery on the page, in which case a solution might be to keep the related links white but make the rest of the page darker. Alternatively, changing the colour and typeface of the links would be a different way to group them together if background colour proved problematic.

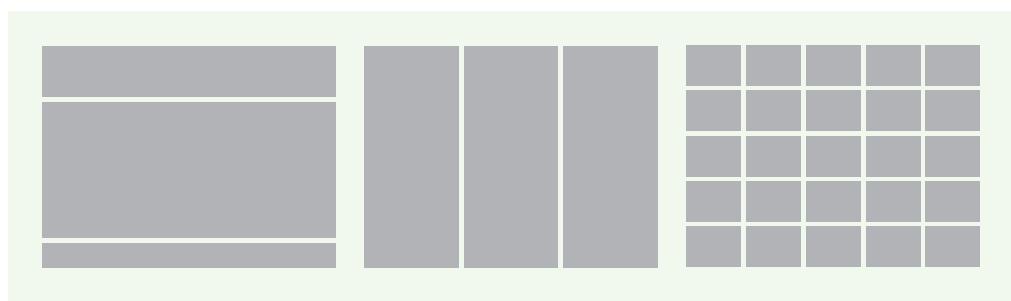
Grids

Hierarchy lets you outline your typography; gestalt principles enable you to group and relate elements. Grids help you organise and arrange your content. They give you a structure on which to hang your content, and a consistent way to expose the information system to your reader.

One thing grids are not is rigid. Overlaid on a design they can look and feel like prison bars, but grids are not jail cells waiting for content, with a framework regime telling content where to go. A grid is a collection of well-organised possibilities.

Types of grid

There are essentially three types of grid: hierarchical, columnar and modular.

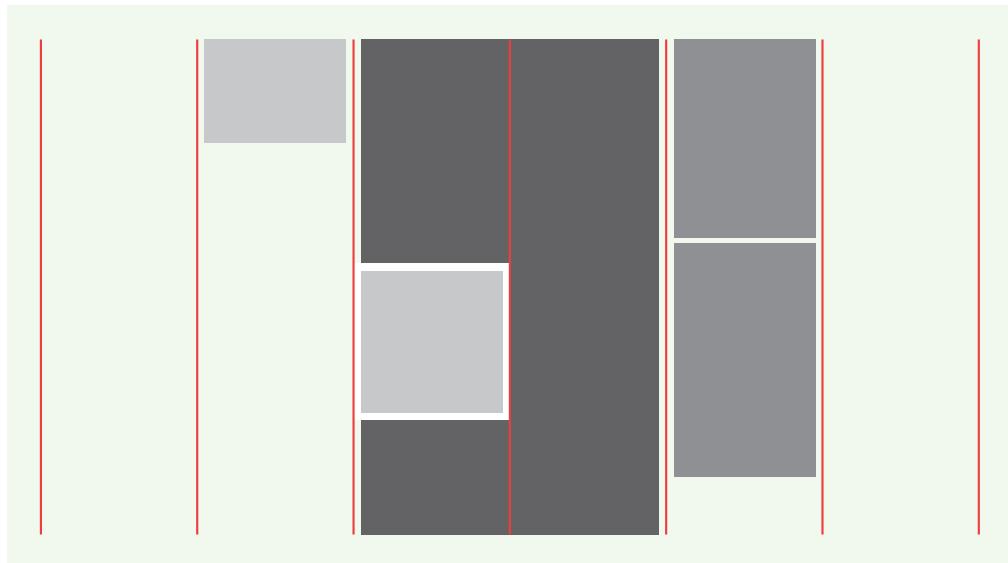


Left to right: Hierarchical, columnar and modular grids.

Hierarchical grids organise disparate content by grouping it in horizontal modules stacked vertically one after the other. They are simple and extremely flexible as they have no limitations on the length (height) of each element. Hierarchical grids require very little horizontal space

besides a comfortable measure, so they are ideally suited to small screens. The downside is that relationships between the groupings are limited to a linear hierarchy.

Columnar grids consist of vertical columns spaced predictably across a page. You can use them to situate different types of content next to each other, such as related links, imagery or advertising alongside an article. You can also design your text block to span multiple columns, and use the column grid to place content within the text block, or alongside it at strategic points.



Columnar grid with sidebars and inset block.

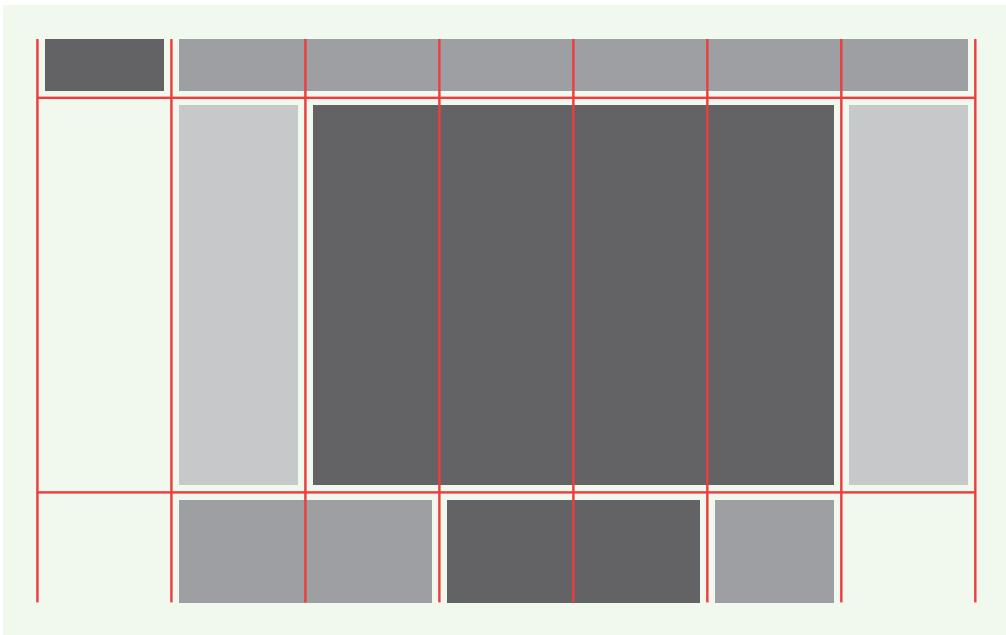
Modular grids are similar to columnar grids, with the addition of rows placed at regular intervals to create a matrix of modules across and down the page. Modules can contain individual units of content, or can be combined in multiple ways, from a few simple rectangles to layouts resembling a game of Scrabble. Modular grids provide both precision and flexibility, enabling you to be simultaneously creative and systematic. Their drawback is that they rely very much on an outside-in, canvas-first approach.

Complex modular grids are rarely applicable to the web, as their effectiveness lies in the reader being able to perceive the entire grid and make associations between elements based on their dimensions and juxtaposition. True modular grids require knowledge and control of text size, screen

dimensions, and the content itself so that the modules are rendered and populated as intended. Web pages are filled with content from the top downwards; modular grids require that content is placed precisely on its two-dimensional lattice. All of these factors compete with the requirements of responsive design.

Use compound grids for maximum flexibility

An ideal compromise which addresses the downsides of modular grids is the *compound grid*. Compound grids are created by combining hierarchical grids and columnar grids. You can do this in two ways: first, start with a columnar grid and give each column its own hierarchical grid, grouping and dividing the content as you go down the column; or second, start with a hierarchical grid and give each element of the hierarchy a columnar grid.

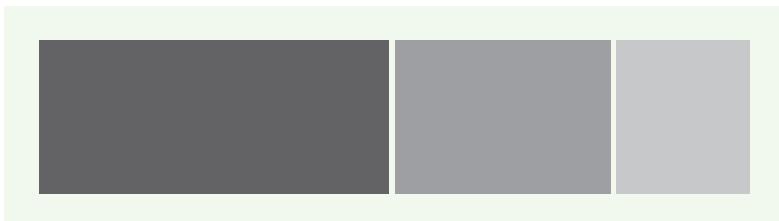


Compound hierarchical/columnar grid.

The latter approach is particularly flexible as you can use a different columnar grid for each element of the hierarchy, choosing whatever is most appropriate to the content. Both approaches will adapt themselves well to your reader's preferences, screen dimensions and unknown content poured out of a database.

Choose odd over even grid systems

Many grids on the web, particularly those used by CSS frameworks, are based on even numbers of equally sized columns. Grids do not require uniformity – their job is to provide consistency, not monotonous regularity. You could create a simple asymmetrical grid based on the golden ratio, with each column 1.618 times smaller than the previous one. Or you could instead have decreasing column sizes based on a ratio of $1:\sqrt{2}$, with each column a factor of 1.414 narrower than its neighbour.



Asymmetrical grid based on golden ratio.

You could also choose to have a column sized specifically for advertising, with adjacent columns flexibly holding content with a readable line-length. A grid system can make your work as a designer easier, but always choose your grid by what works best for the content and your reader, not your own personal convenience. The even number of columns dished up by most grid frameworks on the web is a consequence of mathematics, not design. That the grids are divisible by 12 (and 6, 4, 3, 2) is espoused as a virtue, but it has nothing to do with the needs of your content, and everything to do with the desire to provide a generic system.

If you were to research the kinds of grids in use across all media, you would find that an odd number of columns is the norm. Newspapers in particular tend to use either five or seven columns. An even number of columns is stable, but an odd number of columns provides a useful tension which you can use to readily imply hierarchy and lead your reader's eye around a layout.

The CSS Grid Layout⁴ module will go some way to addressing some of the limitations of modular grids, adapting its advantages to a responsive design context, but the canvas-based nature of modular grids with fixed columns and rows will always be problematic on the web.

⁴ For a great introduction, read *Get Ready for CSS Grid Layout* (<http://wbtyp.net/149>) by Rachel Andrew.

Restrict column length to the viewport height

You can use CSS columns to span short passages of text across a page. This can be a useful way to present introductory or supplementary text in a compact manner while restricting the measure to a readable width.

You can set columns on any block-level element such as a paragraph. You can also use them for multiple elements by targeting their parent, such as a `<div>` or `<article>`. Declare columns by specifying the minimum column width using `column-width`:

```
article {  
    column-width: 10em;  
}
```

In this instance, browsers will render the `<article>` element with as many columns as it can fit, provided the columns are no narrower than 10em. If your article element is 15em wide, the text will be rendered in a single column with a width of 15em. If the article is 22em wide, there will be two columns, each 11em wide.

You can also turn on column layout by specifying the number of columns using `column-count`:

```
article {  
    column-count: 3;  
}
```

This will lay out text in three columns regardless of how wide or narrow they need to be to fit the available width. This gives you much less control over the readability of text. If you combine `column-count` with `column-width` then `column-count` becomes the maximum number of columns, and `column-width` is the minimum width for those columns, giving you a little more control over your column layout.

You can also specify the gutter between columns using `column-gap`, and insert a border between columns using `column-rule`:

```
article {  
    column-count: 3;  
    column-width: 10em;  
    column-gap: 1.5em;  
    column-rule: 1px dotted #ddd;  
}
```

In the embellishment or decoration of a message, there lies the opportunity for the little added touch that goes far beyond the bare essentials of taste and

into the realm of fantasy. For this reason the ornamentation of printing is at once the most charming and the most dangerous diversion the

typographer can find: dangerous because all matters of decoration call upon the utmost discretion and sense of fitness for their effective use.

In the embellishment or decoration of a message, there lies the opportunity for the little added touch that goes far beyond the bare essentials of taste and into the realm of fantasy. For this reason the ornamentation of printing

is at once the most charming and the most dangerous diversion the typographer can find: dangerous because all matters of decoration call upon the utmost discretion and sense of fitness for their effective use.

In the embellishment or decoration of a message, there lies the opportunity for the little added touch that goes far beyond the bare essentials of taste and into the realm of fantasy. For this reason the ornamentation of printing is at once the most charming and the most dangerous diversion the typographer can find: dangerous because all matters of decoration call upon the utmost discretion and sense of fitness for their effective use.

Columns automatically adjust to viewport width.

css columns set with `column-width` are inherently responsive from a viewport width perspective. However, if your column extends beyond the viewport then your reader will need to scroll back up to follow the columns, interrupting the flow with an awkward manual intervention. If you know how much text you are setting in columns, work around this limitation using a vertical media query which turns on columns only when there is enough room to show the whole column.

```
@media (min-height: 25em) {
  article {
    columns: 3 10em;
    column-gap: 1.5em;
    column-rule: 1px dotted #ddd;
  }
}
```

You will need to experiment to see what your minimum height requirement is.

3 Choosing and Using Fonts

- How fonts render on screens
- Practical and pragmatic considerations
- Knowing and browsing type
- Selecting typefaces for body text
- Selecting typefaces for display text
- Selecting typefaces for functional text
- Combining typefaces
- Using web fonts
- Building a library

¶ *The typeface shouldn't be the design;
it should serve the typesetting.
Your choice of typeface is decided
only when you know the precise
requirements of the job.*

How fonts render on screens

Screens are crude, hostile places for words to live and breathe. Their emitted light¹ is tiring compared to the reflected light of paper. Resolutions are often dismally low compared with print. Device dimensions are disparate and unpredictable, and the appearance of text varies significantly from system to system.

These factors are why it is important for us to pay as much attention as possible to typography on the web. When text is crudely rendered, it is not an inherently comfortable place for our readers to be. They naturally scan the page, sweeping across it as an astronomer might observe the sky, pausing occasionally to study small details. We need to do everything in our power to overcome the obstacles to reading, and that means understanding how letterforms end up represented on the screen.

Accept the limits of screen resolution

The majority of the text we see around us, whether in print or on screens, is comprised of dots of various colours, shapes and sizes. Digital printers squirt or stick tiny drops of ink. Screens radiate the tiny on-or-off points of light we know as pixels.

When we talk about resolution, we're describing how many dots are present in any given area. When it comes to screens, the dots in question are pixels so we more correctly describe screen resolutions in terms of *pixels per inch* (ppi), as opposed to dots per inch (dpi), which should be reserved for print. Many screens, especially laptops and desktops, have a resolution of around 140 ppi. High-resolution devices such as Apple's Retina screens have less than 400 ppi. Compare this with the 600 dpi

¹ Except for some e-ink devices such as Kindles.

resolution of a budget laser printer, or professional digital typesetting at 2,438 dpi.

One square inch of a 140 ppi screen contains 19,600 pixels. One square inch of ink from a cheap laser printer will comprise 360,000 dots – eighteen times as many as the screen, and four times as many as a Retina display. The more dots or pixels available, the less they are individually perceptible, and the greater the detail, sharpness and tonal range. Resolution matters, but there's nothing we can do about it.

Understand how fonts render on screens

Digital fonts are designed and stored as vectors: mathematical formulas describing perfect lines and curves. Screens, however, are made up of pixels; at some point, the font file's vectors need to be turned into dots to be drawn – or rendered – on a screen. This is a process called *rasterisation*.

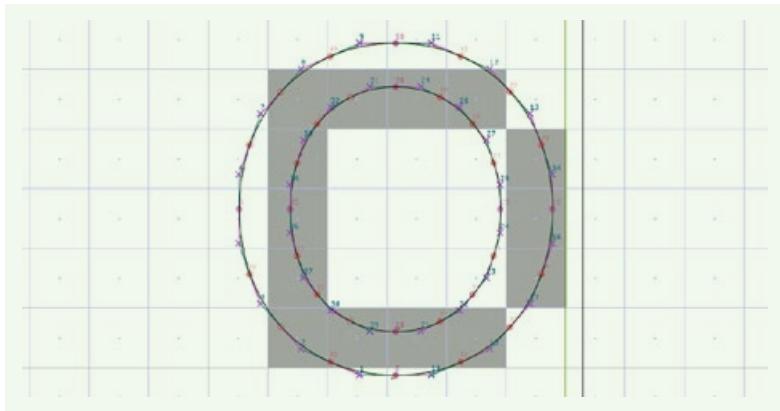
TrueType and PostScript fonts use quadratic and cubic Bézier curves respectively.

Rasterisation is inherently lossy. Consequently, letterforms can be distorted or even made illegible when converted into groups of pixels. It is also subjective. Different settings on different text rendering engines will rasterise the font files with varying degrees of success and significantly different matrices of pixels for the same letterforms.

Each operating system contains a text rendering engine (sometimes more than one) and each web browser controls which rendering engine is used. This means that on the same operating system, two browsers can produce text with very different appearances. Choice of text rendering engine and its settings can also vary between versions of an OS and between browser releases. On top of that, users can tweak some of the rasterisation settings in browsers and operating systems.

Introducing anti-aliasing

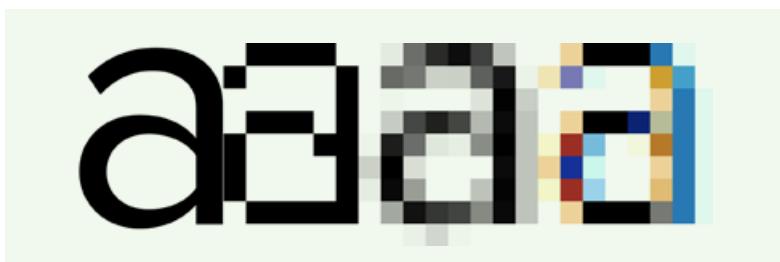
Assuming we require black text on a white background, the simplest way to rasterise a letter is to make a pixel black whenever the centre of the pixel falls inside the outline. Aligning letterforms to the pixel grid in this manner gives you *aliased* text. The downside of this simplistic approach shows itself when part of the letter accidentally catches too many pixels: the characters can become blobby and ugly. Worse still, if the shape catches too few pixels the letterform can become illegible.



Aliasing can result in lopsided characters like this 11px o.

A way to mitigate the unevenness of aliased text is greyscale smoothing, a form of *anti-aliasing*. With this method, whenever *part* of a pixel falls inside the outline it is turned a shade of grey. The more of the pixel that is inside the outline, the darker the shade of grey. Pixels entirely inside the outline are turned black.

The pixels in liquid-crystal (LCD) and organic light-emitting diode (OLED) displays are made up of red, green and blue stripes known as subpixels. By altering the brightness of each colour within the pixel, *subpixel anti-aliasing* can be achieved, increasing the apparent resolution of the screen and sharpening the text.



L-R: Vector, aliased, greyscale smoothing, subpixel anti-aliasing.

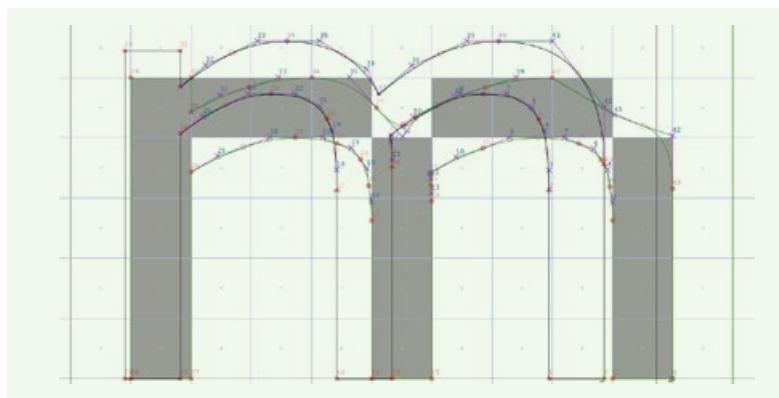
On close inspection, greyscale smoothing is always blurrier than subpixel anti-aliased text. This is not a matter of opinion – it's how the rendering works. Subpixel anti-aliasing uses each of the three individual subpixels in every pixel of the screen to smooth out and sharpen the edges of the font. Greyscale smoothing relies on whole pixels only, which effectively reduces

its smoothing sharpness by three times. That said, subpixel anti-aliasing comes with downsides. On lower-resolution screens there can be visible colour fringes and a slight emboldening of letterforms, but these are a worthwhile compromise for sharper text.

Enter hinting

As we've seen, mapping vectors onto pixels can be problematic. This is especially true for black and white rasterisation, but also an issue for greyscale and subpixel anti-aliasing. This is where *hinting* steps in. TrueType hinting provides additional instructions which bend the contours of the letterform into a slightly different shape, so that the pixelated letterform, once rasterised, represents a more legible character.

Steps involved in programming hinting instructions include snapping the outline to the pixel grid, ensuring the stem weights (thickness of the vertical lines) are consistent within a letterform, across the alphabet, and representative of the original design.



Hinting instructions adjust the design to fit the pixel grid.

Hinting is done on a character-by-character basis for many different text sizes. If this sounds like a rather tedious, time-consuming activity, that's because it is. It can take an experienced hinter around 80 hours to fully hint a single font with the basic 256-character set.

Most computer users think of Verdana, Georgia, Arial and other core web fonts as free, but in reality Verdana is probably the most expensive, labour-intensive font ever produced. It includes characters used to write an extremely wide range of languages, and each of these characters had to be adjusted to be readable at every point size between 9 px and 60 px.

In other words, each of more than 890 characters was effectively redesigned dozens of times for black and white aliasing, and again for greyscale and subpixel anti-aliasing.

Autohinting to the rescue

It is possible to hint fonts using an automated process. Most web fonts will be provided autohinted because full manual hinting is so expensive. Autohinting is far more successful for anti-aliased fonts than for black and white rendering, and is much better than no hinting at all. In the words of type designer and engineer Tim Ahrens²:

In hinting, like with many things in the world, initial improvements can be achieved quickly and without too much effort but as you get closer to perfection the effort that is necessary to achieve improvements increases. [...] Spending a few hours setting manual hints fixes most remaining problems [after autohinting] but if you want really perfect rendering you can spend much more time, virtually without any upper limit. Thinking of Verdana as an extreme example, where the hinting probably took weeks or months.

Accept rendering differences between platforms

We've seen there is more than one way to turn a font from vectors in a file to pixels on a screen, and within those methods there are settings, tweaks and tuning that can be applied by both font engineer and your reader. There are also philosophical differences of opinion between platforms and operating systems, in particular between Microsoft and the rest of the world. These are:

- Only Windows uses hinting. No other operating system (including iOS, macOS, Android and Linux) makes use of hints.
- Handheld devices (including iOS and Android devices) use greyscale smoothing rather than subpixel anti-aliasing. This is because physical subpixels are aligned in a row, so the antialiasing doesn't work well when rotated. Windows-based mobile devices combine greyscale smoothing with hinting.
- LCD and OLED desktops, laptops and other screens whose orientation can be predicted use subpixel anti-aliasing by default. Windows machines combine subpixel anti-aliasing with hinting.

² In a reply to a comment on '[A closer look at TrueType hinting](http://wbttyp.net/4)' (<http://wbttyp.net/4>) on the *Typekit blog*.

Microsoft's stance is to opt for clarity over accuracy. Using hinting, Microsoft's DirectWrite rasteriser tries to align characters to the whole pixel grid, with the result that regular weights can look lighter, bold weights can look heavier, and subtle details of design can be lost at small type sizes. Apple and others prefer to preserve the design of the typeface as much as possible. Apple's Core Text rasteriser tries to present text as close to the original vectors as possible, sometimes at the cost of image clarity. The result on lower-resolution screens (<200 ppi) is that Windows users get sharper, less blurry text and everyone else gets accurate but fuzzier renditions. This can be frustrating for those of us used to one system or another, but in the words³ of type designer Christian Schwartz, 'Our goal should be to make everything look good but not the same. After all, a typeface looks different on coated and uncoated paper.'

We are entering a period of digital divide: the haves and have nots of font performance on screen. Lower-resolution screens provide fewer pixels with which to render text, poorly engineered fonts are more problematic, and the subjectivities of rasterisation are more apparent.

Conversely, with high-resolution screens, text rendering converges. More pixels means clarity catches up with accuracy, rasterisation opinion becomes a moot point, and consistency wins out.

Typography is the art and technique of arranging and selecting **typefaces**, **point sizes**, **line lengths**, **line**

Typography is the art and technique of arranging and selecting **typefaces**, **point sizes**, **line lengths**, **line**

Text rendering on hi-res screens differs little. *Top:* Windows. *Bottom:* macOS.

It is clear that one day font hinting will become obsolete. It is not used on mobile devices, which now account for over half of all internet usage, and is far less necessary on high-resolution screens. As of now, however, hinting is still important and will remain so for as long as Microsoft embraces hinting and a significant number of people use Windows machines with lower-resolution displays; that may be the case for at least another decade.

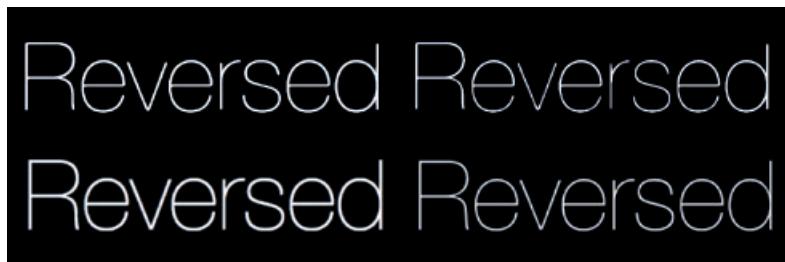
³ At Ampersand NYC conference (<http://wbtyp.net/150>) on 2 November 2013.

Nonetheless, type designers are agreed⁴ that situations calling for extensive manual hinting are fewer. In most situations they would prefer to autohint, then manually tune the exceptions that slipped through the autohinter's net.

Don't disable subpixel rendering except when text is reversed out

Windows machines, mobiles and high-res screens suffer less from this.

Subpixel anti-aliasing has one major drawback on lower-resolution macOS screens. When text is reversed out (that is, light text on a dark background), it often renders far more heavily than dark on light. The issue is exacerbated with large text, which can sometimes seem an entire weight bolder.



Left/right: Subpixel/greyscale rendering. *Top/bottom:* High-res/low-res screen.

You can force browsers in these circumstances to use greyscale smoothing instead of subpixel anti-aliasing by using a couple of proprietary properties:

```
h1 {
  background: #000;
  color: #fff;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}
```

Use these properties sparingly. Subpixel anti-aliasing gives a superior reading experience to greyscale smoothing, so only turn it off when you are reversing out text, and preferably only with larger text. For body text, allow the differences to exist.

⁴ See discussion 'Is type hinting for screens still relevant?' (<http://wbtyp.net/33>) on *TypeDrawers*.

Check your type across platforms

As the majority of screens gradually move to high resolutions, we'll stop talking about quality of rendering and go back to talking about the type design itself. In the meantime, your duty to your readers is to check the rendering of your chosen fonts across platforms and browsers. Some fonts rasterise far better than others, and it's not just a case of aesthetics. Poorly designed, unhinted and badly hinted fonts can make the difference between legible and unusable text.

Test early and often. If you live and work on a Mac, invest in a Windows machine for testing. A cheap laptop will be ideal, especially one with a lower-resolution screen to better expose the frailties in fonts.

Print designers who gauge their work on the screen, and web designers who gauge their work exclusively on their own machines, are arrogant in their disregard for the reader.

— Dean Allen in ‘Reading Design’ on *A List Apart*

As with all web design, don't expect rendering to be the same across devices. Assess whether something is just different or degraded based on a choice you have made. Even if your readers use multiple devices – a PC at work, an iPad at home, an Android smartphone – they won't be comparing them side by side and will be used to the vagaries of their machines. Ask yourself: is the reader experience harmed or will your reader think something is broken? From a typographical perspective, can they still read comfortably and does the type do what is required of it, emotionally and structurally? If the answer is still no, then there's very little you can do other than choose a different font.

Practical and pragmatic considerations

For many designers the highlight of their process is choosing typefaces. It's an absorbing, enlightening, intellectual and joyful undertaking, and one which you should tackle as a designer, not as an artist or a scholar. Your role is to choose fonts which fit the requirements and constraints of your brief. Before you embark on the process of choosing a typeface to subjectively match the voice, message and feel required by the text, ensure you address the objective needs of the project.

It is acceptable to choose a typeface that you know inside out, provided it supports all the practical considerations of the text. It is also fine to choose a brand new typeface you've been itching to try out, again provided it meets all the pragmatic criteria. Your primary obligation as a typographer choosing typefaces is to ensure your final choices are *not wrong* for the job at hand. Do that, and what you're left with is subjective opinion of *how right* the ultimate decision is. If the objective needs have been met, and your client is happy with the final choice, then you can be content with a job well done.

Don't jump straight into choosing fonts

Avoid making typeface choices at the beginning of the project. Spend time setting the typographic direction of the design first. You need this to give you a clearer idea of the contexts and situations in which the typefaces will be used.

Your final choice of fonts will depend on specific requirements, context, constraints, confidence and, finally, taste. The first stage in reaching a decision is to create a list of viable candidates that you then whittle down. All the fonts in your list should do what you need them to do from a practical point of view. There's no point shortlisting a typeface which will not do the job required of it. Consider the pragmatic requirements before you start making any aesthetic or emotional judgements.

Understand the purpose and requirements

Ensure you have a complete overview of what the fonts will need to do. A great landscape gardener will know the contours they are working with. They will know the shapes they are forming, the shadows they wish to cast, the colours and textures they want to create. Only once these criteria are known can they then decide which plants to situate and where.

Understand the nature of the website you are designing. Think about its future. Might the small project you are working on end up much bigger? You could need more complex functionality and features in the future. Be clear about the general purpose of the fonts you are choosing. Are they required for immersive reading or merely scanning? Do you need to style many interface elements, such as navigation, labels and form controls? Is there much microcopy such as notes, notifications, alerts and captions? What about small print or tabular data? Are you setting display text to be looked at rather than read?

Use a typeface with the necessary characters

Once you have understood the general purpose for the font, your first criterion is to work out which characters – letters, accents and punctuation – are required. It stands to reason that your chosen font should contain the characters needed to form the words and sentences the writer composes. This sounds obvious, and yet so often we see this, recreated from a British national newspaper website:



Cesc Fàbregas

Mixing typefaces due to missing glyphs.

In this case, Cesc Fàbregas is the name of an internationally famous Spanish footballer, at the time of writing playing in the English Premier League. The Catalan à in his name has been rendered in a different typeface, causing it to stand out like an intrepid traveller among locals in a foreign bar. This, despite Fàbregas appearing in the newspaper's football coverage on a weekly basis.

It's worth understanding what is happening here. When a browser renders text it looks at each character individually. It then iterates through the list of typefaces in the `font-family` stack until it finds an available font that contains a glyph for the character to be rendered. In this case, the first choice font did not contain an à, but a different font further down the chain did. This is the web's way to allow for differences in available fonts across platforms and for differences in the range of characters supported by individual fonts.

Specify typefaces suitable for all the languages you are setting. Be aware that some topics (such as football or chemistry) may include glyphs that are not in the primary language of the text. Perhaps you need to set some official German writing and require an eszett as in Haßfurt, or you might need a Turkish dotless i, as in Fırat Üniversitesi.

Free fonts will often lack important accents and other characters such as currency symbols and less common punctuation marks. Test with an example text. Any good font service or foundry will let you try before you buy.

Make sure your typeface has the requisite styles

Your design scheme may require a number of different styles of your chosen typeface: a regular weight for reading; an italic for quotations and emphasis; a bold for headings. Maybe your hierarchy system requires a semibold for some or all of its headings. You may want your display text to be rendered in a huge ultralight setting.

You might have requirements for specific text to fit in a given region, or have limited room for captions. In both cases, a condensed version of the typeface might be required.

It's not unusual for font families to come with only one width and four styles (bold and italic). Some typefaces have even fewer than four. John Baskerville didn't cut bold fonts for his eponymous typeface, so faithfully revived Baskervilles come with an italic, but no bold.

Some other fonts have regular, bold and italic but no bold italic, it being seen as an unnecessary evil of the digital age. Premiéra, designed in 2009 by Thomas Gabriel, and used throughout this book, is one of those without a bold italic.

Choose fonts with the features and special effects you require

Study the OpenType features available in the fonts you are considering. Do they have any at all? Many free fonts don't. If you're designing text that asks for subtle typography with a hint of tradition you might look for the presence of small caps, historical ligatures and old-style numerals.

Will the content contain many tables of data? If so, you'll probably need tabular lining numerals on hand. What about scientific notation? Do you require proper superscripts, subscripts and mathematical symbols?

Are you going for a friendly, inviting, maybe child-friendly appearance? Perhaps a font's standard double-storey *a* and *g* are available as single-storey alternates.

What about fancy swashes, discretionary ligatures and other ornaments to lift a font from ordinary to remarkable, and lend your display text a unique and striking air of confidence?

Consider your fonts' performance

You need to ask yourself how you want the fonts to perform from a technical perspective. Are there particular devices or operating systems used by your readers? Is their internet connection particularly slow or unreliable?

Rendering

In '[How fonts render on screens](#)' we discussed how fonts render differently on different machines and platforms. You'll need to test your short-listed fonts on targeted devices to ensure the text rendering is of a suitable standard.

File size

Download speed is important to everyone, even in this age of supposedly fast connections and ubiquitous broadband coverage (something far from the truth in Europe and North America, let alone the rest of the world). But performance matters more in some projects than others. Be aware of the limitations and expectations your readers might have before you bombard them with half a dozen or more 100 Kb font files chock full of massive character sets, hinting, kerning and OpenType features.

Users of web apps might be better served with judicious use of web fonts, whereas readers of magazines and immersive experiences may have a little more patience.

Don't let the drive for performance lead you down the road of faux bolds for the sake of not downloading a bold font file. If performance is that crucial, think of a design solution that doesn't require a bold. Your readers may not know that they are faced with a faux bold but they'll feel it inside. See '[Using web fonts](#)' for more details on addressing web font file size.

Work within your means, but be wary of free

A key consideration for any font shortlist is budget. Has the project or client set aside money for investment in web fonts? If not, can you persuade them or absorb it in your day rate?

With few notable exceptions, good type costs money. Quality typefaces take a significant amount of time to design, finesse and engineer, regardless of whether they are subsequently given away or bundled with operating systems and software.

Some free fonts are very good, and come fully equipped with sizeable character sets and features. Source Serif, Open Sans, Lato, Clear Sans, Merriweather, Fira, Overpass and Alegreya all come to mind. Most free fonts are limited so it's important to find out what they are missing and not to push beyond their limitations. Free or cheap typefaces will require *more* effort to bend them to your requirements.

Licensing

There are different models for licensing fonts. Put simply, they are either free, available to buy for a one-off self-hosting fee, or rented through a service. Prices often depend on usage measured by page views or bandwidth.

If your company or client has a subscription to a web font service you may need to limit your search to that service, although most do have thousands of typefaces available to choose from. If you're not tied to a service then you can self-host fonts; your choice can be wider as most foundries have self-hosting options available. Many also offer their fonts through a web font service. But some still choose one option or the other, confusing the landscape further.

Not all typefaces are yet available as web fonts. Buying a desktop font and converting it to a web font is easy, but the chances are you would be breaking the licence agreement. In these situations, contact the font foundry to see if a solution can be found.

Be pragmatic in the face of branding requirements

You might work for a company or client which has a brand font they would ideally like to incorporate into your design. You will need to ask yourself whether the brand font is suitable for all the practical purposes we've discussed so far. If not, can you use an alternative typeface that is perhaps consistent with or complementary to the brand? If the specified typeface is not suitable as a text font, can you use the brand font just for headings and pair it with something else more appropriate?

Sometimes a typeface was bought at some point in the distant past and is still in common use. You may be free to use something completely different, but you could always take on the challenge of using the incumbent fonts and use them in a different way. Remember typography is less about the typeface and more about what you do with it – it's still possible to cook a fine dish with basic ingredients.

Knowing and browsing type

Learning to scrutinise and describe a typeface's nuances means you will be better equipped to discuss your choices with colleagues and clients. If you are able to get a client to articulate the feel or requirements of a font, you can apply your understanding of type to narrow down the selection effectively.

Why might you want to use a typeface like Avenir? What makes Avenir different from other sans serifs like Helvetica or Gill Sans? If you want a typeface similar to Avenir, how do you track one down? To answer questions like these you need to know type, and that means understanding the varieties, features and history of fonts. There are tens of thousands of professional typefaces to choose from, but only twenty-six letters in the alphabet. It stands to reason that, like so much in typography, the differences are usually subtle, but nonetheless visible if you know what to look for. Being able to describe and distinguish type will help you make informed decisions and navigate the different ways font vendors organise typefaces.

Typeface anatomy

I've tried to avoid excessive typographic jargon in this book, but when describing the fine details of any discipline, a certain amount of specific terminology becomes necessary. Features are illustrated on page 211.

Strokes and shapes

All characters, letters, numbers and punctuation are formed from a combination of straight and curved lines known as strokes, a term derived directly from *the stroke of a pen*. The way a stroke is formed is fundamental to the nature of a typeface. Knowing the terminology of different stroke forms is essential to effectively understand and describe a type design.

Axis and stress. If you imagine that a glyph was drawn with a broad-nibbed pen, then the *axis* of a letterform is an imaginary line drawn from top to bottom giving an indication of the angle at which the pen was held. This angle is known as the *stress*. The stress is often – but not always – the same through all the characters in a typeface, and will usually be vertical or back-slanted to some degree.

The easiest way to determine the stress axis is to look closely at the letter *O*. Identify whether the bottom-left is thicker than the top-left, and the top-right is thicker than the bottom-right: if a difference exists, then the letter has diagonal stress. If the top and bottom of the *O* are a mirror image of each other, but with the sides thicker, then the letter has vertical stress. If the top and bottom of the *O* are the same thickness as the sides, there is neither contrast nor stress.

Contrast and modulation. *Contrast* is the degree of difference between the thick and thin strokes in a glyph. High contrast means there is a big difference between thick and thin strokes. No contrast at all, so all strokes are the same width, is often called *monoline*.

Modulation is the way the contrast changes within a glyph. A *modulated stroke* usually implies that the thickness varies predictably and smoothly with the direction the imaginary pen is moving. *Abrupt modulation* means that the stroke changes from thick to thin quickly.

Baseline. An imaginary line on which the letters in a font appear to rest.

Cap height. The distance from the baseline to the top of the uppercase letters (not including accents or diacritics). This is often slightly lower than the height of the tallest lowercase letters.

X-height. The height of the lowercase letters. While x-height does not specifically refer to the height of a letter *x*, the two are usually equal. X-height is a key distinguishing feature of a font and, as detailed in the chapter ‘[Designing paragraphs: text size](#)’ in part 1 of this book, the relationship of x-height to cap height defines the perceived type size. A typeface with a large x-height looks much bigger than a typeface with a small x-height when set at the same size.

Glyph features

The form of a glyph's strokes, and the basic shapes they make, give us the torso of the letter. The parts attached to the torso give us the anatomical features of a glyph, which lend a typeface its distinguishing characteristics. The nomenclature used to label features of a glyph is as follows.

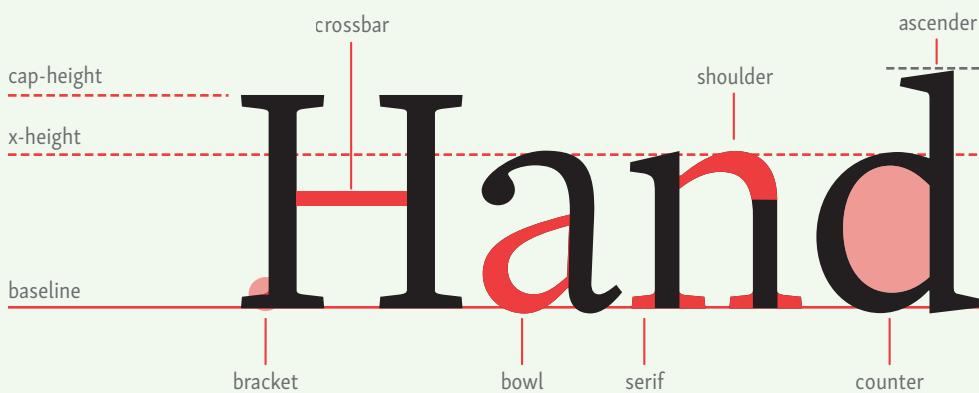
Arm. A stroke which joins a stem somewhere in the middle. A letter *k* has two such arms.

Ascender. A stem on some lowercase letters, such as *h* and *b*, that extends above the x-height.

Aperture. The opening of a partially enclosed part of a letter such as *n*, *C*, *S*, the lower part of *e*, and the upper part of a double-storey *a*.

Bowl. The curved line which fully encloses part of a letter such as *d*, *b*, *o*, *D* and *B*. Compare with *counter*, which is the interior space created when a bowl is drawn.

Anatomical features of glyphs: strokes, shapes and positions



Bracket. A curved or wedge-like connection between a serif and the main stroke or stem.

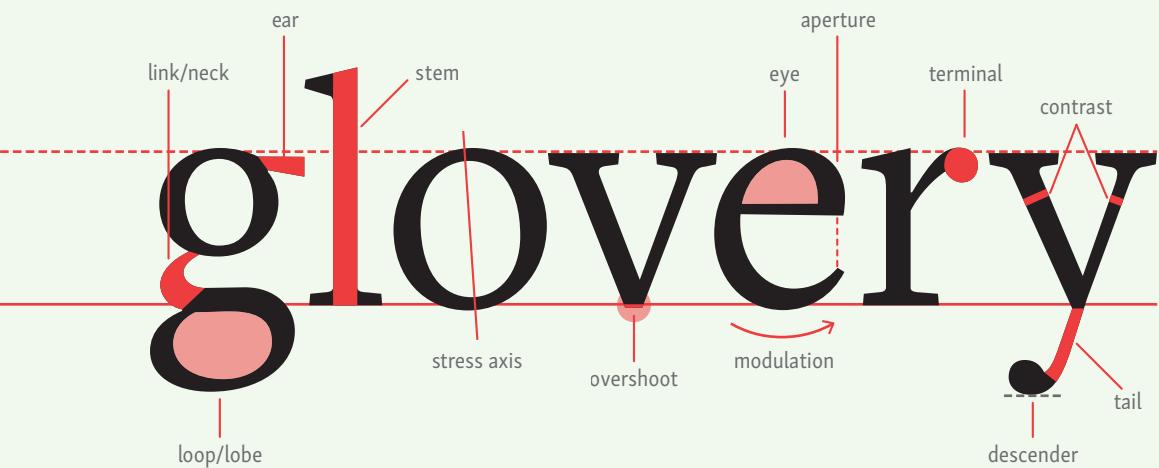
Counter. The interior shape, enclosed or partially enclosed, in letters such as *d*, *o*, *c* and *s*. Compare with *bowl*, which is the line drawn to form a fully enclosed counter.

Crossbar. The usually horizontal stroke across the middle of letters such as *A*, *H* and *e*.

Descender. Any part of a lowercase letter that extends below the baseline, found in lowercase letters including *g*, *j*, *p*, *q* and *y*, and occasionally in uppercase letters such as *J* and *Q*.

Ear. A decorative flourish usually on the upper right side of a bowl, most often found on a lowercase *g*.

Eye. The counter, or enclosed space, of a lowercase *e*.



Link/neck. The join between two parts of a letter, particularly between the bowl and loop of a double-storey *g*.

Loop/lobe. An enclosed or partially enclosed counter below the baseline, most often found in a double-storey *g*, but also in cursive (handwritten) styles of letters including *p*, *b* and *l*.

Negative space. White space formed within the shape and extremities of a letterform.

Overshoot. Many letters, such as *v*, overshoot the baseline slightly. This gives them optical balance and stops the letters looking like they are about to topple over.

Serif. A small mark or foot at the end of a stroke. For example, when the number 1 or the letter *i* are drawn with a bar across the bottom, the two halves of the bar are serifs. If the serif is joined to the stem by a slight flaring out, rather than an abrupt deviation, it is said to be bracketed (see *bracket* above).

Shoulder. The join of a stroke on to the main stem of a glyph, such as where the arm of an *r* joins the vertical stem.

Stem. The main, usually vertical, stroke in a glyph.

Tail. A stroke on letters such as *Q*, *K* and *R* which descends below the baseline. When an *a* has a stroke rising from the stem towards the next letter, as you might in handwriting, this is also known as a tail.

Terminal. The end of any stroke that doesn't include a serif. Some terminals expand into shapes; for example, ball terminals and lachrymal (tear-drop-shaped) terminals.

Learn how to describe typefaces

This approach to describing type is based on the work of Prof. Indra Kupferschmidt.

The terms we've just seen give us a vocabulary with which to describe typefaces. Descriptions, more so than arcane classification systems (more of which later) provide practical ways to identify a typeface, understand the feeling it might impart, what purpose it could be put to, and what other typefaces might be successfully combined with it. You can best describe a typeface using three layers: skeleton, flesh and skin.

Skeleton

Most typefaces can be differentiated according to three basic forms described primarily by stress and modulation. Think of these as being determined by three different writing tools.

1. *Dynamic* forms have a contrast and structure derived from writing with a broad-nibbed pen held at a consistent angle. The result is a modulated stroke, fairly high contrast, angled stress and open apertures. This gives the letters a friendly, natural feel.

Ancient Geography

Dynamic form, set in Bembo.

2. *Rational* forms are derived from writing with a pointed pen. The thickness of the stroke is related to the pressure put on the nib during a down-stroke, while other strokes remain thin. The axis is vertical with high but less modulated contrast and more closed apertures. This gives the letters a stiff and formal impression.

Ancient Geography

Rational form, set in Abril.

3. *Geometric* forms are modelled as if produced by a ballpoint pen. The letters appear more drawn than written, with consistent, mathematical shapes, such as an optically circular o. The axis is vertical with contrast low to non-existent. The letters appear mechanical and precise.

Ancient Geography

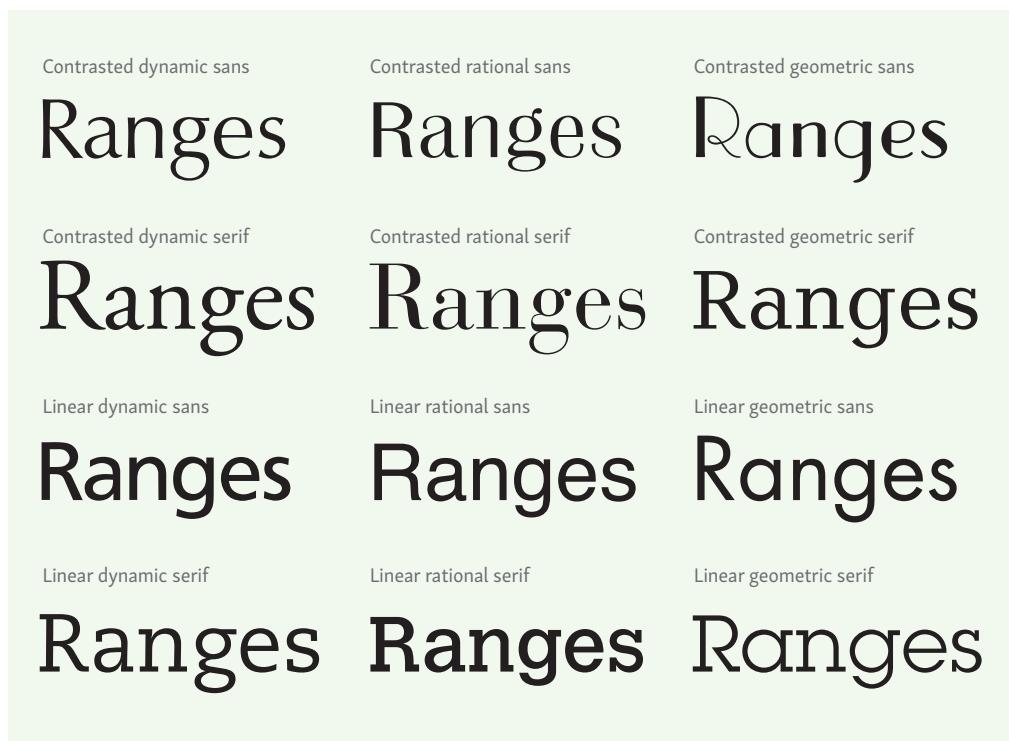
Geometric form, set in FF Mark.

Flesh

The second layer, the flesh, forms the body around the skeleton. This comprises two components:

1. Stroke contrast and linearity. Typefaces can range from being extremely high in contrast to appearing monolinear.
2. Serifs. For the flesh layer, consider typefaces as being either seriffed (with serifs that taper in some form), slab serifs (with thick blocky serifs) or sans serif with no serifs at all.

Taking the skeleton and flesh layers together give us typeface torsos sufficiently distinct to be categorised into broad groups. Each torso can be placed on a two-dimensional matrix: from dynamic through rational to geometric along the horizontal axis; and from contrasted to linear down the vertical axis.



Typefaces ranging from dynamic to geometric (*left-right*) and contrasting to linear (*top-bottom*).

As well as providing a useful structure for identifying stylistically related typefaces, this matrix can help you easily combine typefaces. All fonts that stand in one column will combine harmoniously, whereas mixing the horizontal neighbours is trickier. You can also successfully combine typefaces by going diagonally to give you a more contrasting combination. Put simply, stay in one form model or go for lots of difference; birds of a feather flock together, and opposites attract.

Skin

The third and final layer, the skin, is where you add in the finer details which differentiate typefaces within the main groups. Start off by assessing the x-height. Next, go on to describe features such as the precise nature of serifs, the shapes of the terminals, the forms of individual letters, such as a single-storey *a*. The skin layer should give you enough detail to categorise a typeface into a particular subgenre or classification, and if you add enough unique characteristics, to identify the typeface itself.

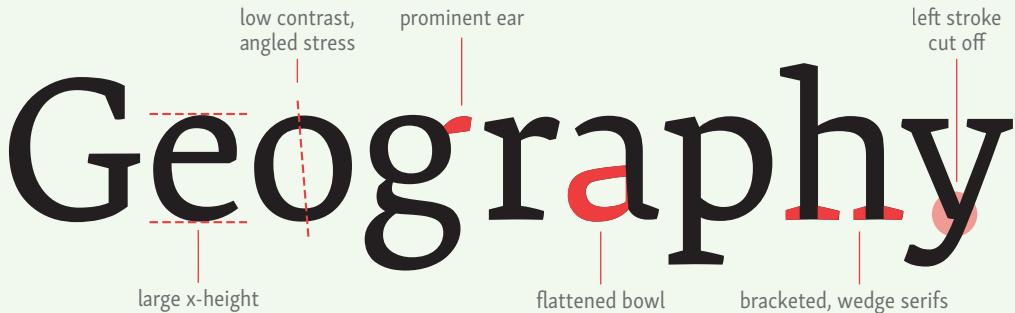
Putting it all together

Let's put the system to the test and use the three-layer model and anatomical terminology to describe the typefaces used in this book.



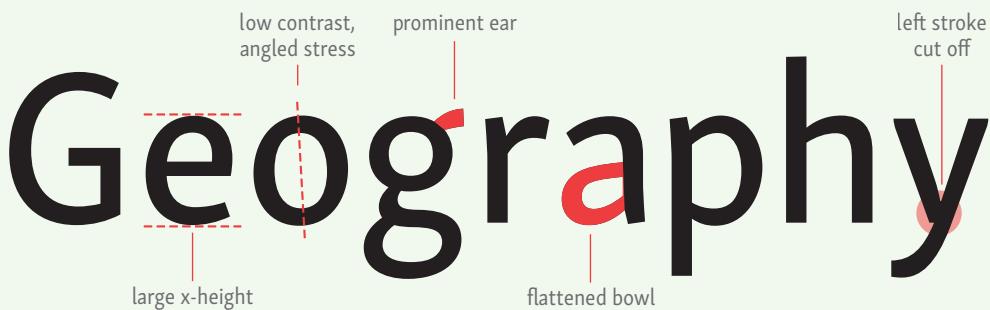
Ingeborg features.

Ingeborg, used for the headings, is a highly **contrasted rational serif** with a tall x-height and long, slightly curved and mostly unbracketed serifs. Finer details include ball terminals, a pronounced ear on the *g*, and a flat top to the bowl of the *a*. The overall effect is one of sturdiness with a glamorous twist.



Premiéra features.

Premiéra is used for the body copy and is a somewhat **linear dynamic serif** with a tall x-height and wedge serifs. Further details include a small and flat bowl on the *a*, and a pronounced ear on the *g*. The dynamic form makes it an engaging read while the generous x-height and simple serifs mean it renders well on screens.



Skolar Sans features.

Skolar Sans, used for the captions, is a **linear dynamic sans serif** with a tall x-height. It too has a small and fairly flat bowl on the *a*, and a pronounced ear on the *g*.

Armed with our descriptions we can now investigate why these typefaces might work well together. Premiéra and Skolar Sans sit well together because they are both linear dynamic forms – each is formed from the same skeleton. The negative space in both typefaces shows many similarities and they share some of the same fine details, such as the ear on the *g*.

Ingeborg works well with Premiéra not because they are similar but because they are opposites, enabling headlines to contrast robustly with body copy. The two typefaces do, however, share some features: the pronounced ear on the *g*, and the flat bowl on the *a* to name two. A third thing in common is that they were released by the same foundry, namely Typejockeys, based in Vienna.

Type genres are more useful than classifications

We've seen how and why you might describe a typeface as being a linear dynamic sans serif. If you want a robust, comfortable to read, informal typeface, this is the kind of design you might well want to go for. Unfortunately, you will be hard-pressed to find any font vendor grouping typefaces together in a category named 'linear dynamic sans serif', or any of the other descriptive terms we've been using.

When browsing for certain kinds of typefaces, one of the issues we face is that there is no universal classification system for typefaces. This is understandable. Just as pigeonholing musicians into predefined genres is difficult and contentious, so typeface designs are subjective and genre-spanning. For around a hundred years many people have tried to create order from the chaos, often resulting in academic and arcane solutions far more closely tied to historical references than to the use and impression you want to achieve. Taken as a whole, these schemes provide us with little more than contradictory and confusing nomenclature. Historically, some foundries used Egyptian or Antique to mean slab serif. The French call sans serif faces Antique, the Germans call them Grotesk, and the Americans use Gothic. But Gothic is sometimes the term for blackletter in European countries.

The classification scheme closest to a de facto standard is arguably the most problematic and unsuitable for use in the modern day. The Vox-ATypI classification started out as proposal by French typographer Francis Thibaudeau in 1921. It is solely based on the form of the serifs, which are used to distinguish between different style periods over the centuries.

The Thibaudeau system was developed further by Maximilien Vox, who published his version in 1954. Continuing with the same main groups as Thibaudeau, Vox came up with names for the groups derived from the names of iconic printers (*garalde* is a portmanteau of Garamond and Aldus; *didone* of Didot and Bodoni). He also expanded the system to accommodate some sans serif designs, but considering the likes of Helvetica had yet to be released at this point, even his additional classifications were soon insufficient.

In 2010, ATypI set up a new working group to revise typeface classification.

The Vox system was taken over by the ATypI (Association Typographique Internationale) in 1960 with some slight modifications. It was subsequently adopted as a international standard, including the German DIN in 1964 and the British Standards Institute in 1967, now notably withdrawn. The Vox-ATypI system has not been significantly updated since that time.

The design of many contemporary typefaces renders Vox-ATypI inadequate. Aside from the impenetrable naming scheme, Vox-ATypI puts too much emphasis on the historical sequence and distinguishing the subtle differences between early serifed typefaces. It generalises greatly when it comes to sans and slab serifs, which is particularly problematic given how widely developed and popular these styles have become in the years since. Many modern typefaces draw from history but cannot be put into individual historical categories because their designers borrowed from different eras and styles, then mashed them up to suit today's tastes and needs.

Despite this, it is worth being familiar with the Vox-ATypI terms and how they match, overlap and compare with more contemporary and descriptive categorisations. Before we do that, a short history lesson is in order.

Know your history

The Vox-ATypI classification scheme is tied up in history. While that is one of its downfalls, knowing when and where things have come from can be illuminating, and often goes some way to explaining why things look like they do. Knowing the history, heritage and geography of typefaces can be fascinating and useful when you're in the process of choosing fonts (it can also help you avoid embarrassing mistakes that, admittedly, would only be picked up by fellow type nerds).

Western typefaces first appeared around 600 years ago, when Johannes Gutenberg introduced mechanical movable type printing to Europe. The typeface Gutenberg introduced was a *textura* blackletter based on the lettering of monks and scribes who up until that time were solely responsible for the production of books. The so-called roman letterforms we are more used to nowadays were also around in hand-lettering at the time (known as Carolingian script), and it was only a few decades after Gutenberg introduced the press that roman fonts began to appear.

Since then, many new genres of typeface have appeared, but these never supplant earlier designs – they are added to the canon of possibilities. Typefaces designed in the fifteenth century are still in use today, albeit primarily in digital incarnations. The dates in the following timeline indicate when the style first began to appear, and the example typefaces given are taken from that time where possible.

15th century: Old-style serif

The original form of roman typefaces are humanist serifs. They are *dynamic* in form with a diagonal stress and a gradually modulated, low to *moderate contrast*. They can be further divided into the Venetian style, which has an angled crossbar on the *e*; and the French or Dutch style (appearing in the 17th century), in which the *e* usually has a smaller eye. Both variants tend to have bracketed, sometimes asymmetrical serifs. Examples include Jenson, Centaur, Garamond and Bembo.



Early 18th century: Transitional serif

Transitional – or realist – serifs saw a gradual move away from letterforms resembling handwriting. They are more *rational* in nature, with a mostly vertical stress, an *increased contrast* and more regular proportions. They have a larger x-height than old-style serifs, with apertures more closed, bracketed serifs and often lachrymal terminals. Examples include Caslon, Baskerville, Bell, Bulmer, Scotch Modern and Bookman.



Late 18th century: Rational serif

So-called modern serifs take the realist form to an extreme, with upright *rational* forms and *high contrast* between thick vertical stems and fine horizontal hairlines. The serifs are horizontal, thin and abrupt. The letterforms are consistently structured and often feature ball terminals. In the 19th century, the style was exaggerated into font designs for display purposes. Examples include Didot, Bodoni and Walbaum.



19th century: Realist sans serif

The first sans serifs were literally labelled ‘grotesque’. The advent of the Industrial Revolution had led to cheaper paper and printing costs, so setting type in relatively huge sizes became economically viable. At this time, display fonts were made from wood, which is likely to have influenced the simplified shapes of these designs.



Realist sans serif typefaces have upright, *rational* forms analogous to the transitional serifs, with relatively *low-contrast linear* strokes and closed apertures. Early forms have a double-storey *g* and a diagonal leg on the *R*. Later, mid-twentieth century neo-grotesques are more open with larger x-heights. Examples include Franklin Gothic, Akzidenz Grotesk, Helvetica and Univers.



aen
aen

19th century: Rational slab serif

At the same time as the sans serifs, slab serifs were appearing, again initially for display use. Stylistically, they began with block-like rectangular serifs added to the emerging realist sans serifs. The sturdy *rational low-contrast* designs held up well under adverse printing conditions, and text faces were developed with ball terminals and bracketed serifs (familiarly known as Clarendons). Examples include *Serifa*, *Clarendon*, *Century Schoolbook*.



aen
aen

20th century: Geometric sans and slabs

The geometric sans serif developed in 1920s Germany in the attempt to further rationalise the vernacular grotesques. The letters are constructed with optically corrected *geometric* forms based on circles or rectangles, and very little contrast. The *a* and *g* are usually single-storey. Slab serifs were added to the geometric forms, often literally – Lubalin Graph is a modification of Avant Garde. Examples include *Futura*, *Avenir*, *Century Gothic*, *Lubalin Graph*, *Memphis*.



aen
aen

20th century: Humanist sans

As a counterpoint to the industrialisation of the early 20th century, humanist sans serifs appeared, bringing *dynamic* forms reminiscent of the old-style serifs of the Renaissance. These were more calligraphic in nature and were usually drawn with true italics. Examples include *Gill Sans*, *Frutiger*, *Syntax*, *Optima*.



aen
aen

Late 20th century: Contemporary serif

Various forms with *low contrast*, large x-heights, wide apertures and chunky serifs. Pragmatic, highly functional designs with details simplified to work on poor paper, low-resolution printing and screens. Examples include *Fedra Serif*, *Premiéra*, *Skolar*, *Georgia*.

Late 20th century: Revivals and synthesis

From the later 20th century, the landscape of typeface design changed hugely. There was an exponential increase in available typefaces, made possible by digital fonts supplanting metal and wood, and further expanded by the distribution channels of the internet. In terms of design, contemporary typefaces are either revivals or syntheses. As the saying goes, originality is for people with short memories.

Revivals are remakes of past typefaces, digital incarnations of iconic designs from earlier centuries for use in today's typographic settings. Some of these attempt to faithfully reproduce the feel of the original metal type,

correcting poorly made versions created in the early days of digital fonts (Lars Bergquist's Baskerville 1757 is one such example). Otherwise classic designs are modified for contemporary tastes and expanded with character sets and features possible in modern technology. Examples include Mark Simonson's Bookmania and Kris Sowersby's Founders Grotesk (derived from early 20th century sans serifs).

The most interesting typefaces of modern times do not fit easily into historical categories. These designs are a synthesis of different styles, merging different forms through cumulative experience. By way of example, Jos Buivenga's Calluna has the solidity of a transitional serif combined with the flow and dynamic form of an old-style. Hoefler & Co's Sentinel is a Clarendon-like rational slab serif with added contrast and reduced serifs for improved readability; it also has an accompanying true italic missing from the original 19th century faces. The multifaceted typeface Abril, by Veronika Burian and José Scaglione at TypeTogether, was designed from the ground up with two different complimentary forms. The display styles are based on the high-contrast rational serif form, while the text style is based on the lower-contrast transitional form; but because the two were designed at the same time they are entirely compatible, with consistent shapes and features.

You can't copyright a typeface design

Revivals and derived designs are perfectly allowable as it is not possible to copyright a typeface design, the thinking being that to do so would be akin to copyrighting the alphabet. The only protection you are afforded as a designer is a trademark for the typeface's name. Over the centuries this has led to lots of very similar or even identical typefaces with different names. However, digital fonts are deemed software, and software is covered under copyright. This means you can't just copy some digital files and rename the font. You could, though, print out each character of a font (in each weight and style) and draw a brand new set of vectors from scratch for every single glyph, correct the spacing around the characters, kern the letter pairs, test it at multiple sizes on multiple paper stocks, hint the font and test on multiple screens. And think of a new name.

Understand type categories

Typographers, graphic designers and type vendors mix different terminologies and classifications when grouping type. Some categories have a multitude of synonyms, many of which you will come across at some point when browsing typefaces. Descriptive rather than historical classifications

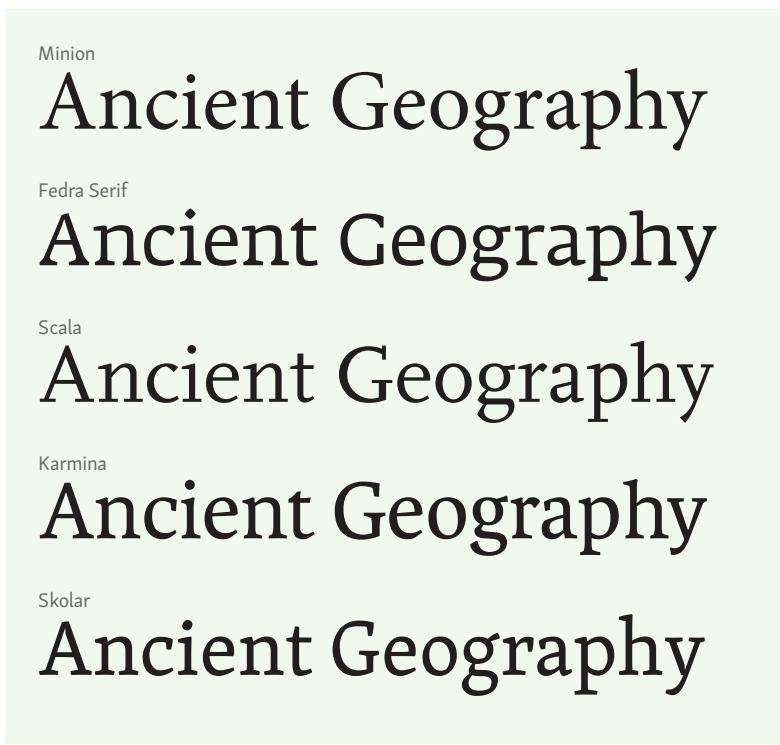
are becoming more common, and so the following categories map stylistic genres and subgenres to alternative names, including the Vox-ATypI system (*italicised*). The categories detailed are divided into serif, sans serif and slab serif styles, which cover all typefaces suitable for body text as well as many display faces. Script fonts are much less used on the web, and decorative display faces rarely fit into any consistent categories, so they are not covered here.

1. Humanist serif

Dynamic form, moderate contrast. Very calligraphic with consistent stress angle and bracketed, often asymmetrical serifs.

ALSO KNOWN AS: *classical, old-style, dynamic, Renaissance, humane, antiqua, medieval.*

USE AND IMPRESSION: Long or short text. Friendly, open, rich, flowery, importance, reverence, historical, natural world.



1a. Garalde

Small counter and horizontal crossbar on the *e*.

ALSO KNOWN AS: *garalde*, Aldine, French old-style

Adobe Garamond

Ancient Geography

Bembo

Ancient Geography

Goudy Old Style

Ancient Geography

Sabon

Ancient Geography

Magneta

Ancient Geography

Humanist garalde serifs.

1b. Venetian

Diagonal crossbar on the *e*.

ALSO KNOWN AS: *humanist*

Calluna

Ancient Geography

Adobe Jenson

Ancient Geography

Centaur

Ancient Geography

Feijoa

Ancient Geography

Humanist Venetian serifs.

1c. Dutch-taste

Denser with more contrast.

ALSO KNOWN AS: Dutch old-style

Janson

Ancient Geography

Adobe Caslon

Ancient Geography

Humanist Dutch-taste serifs.

2. Transitional serif

Rational form, higher contrast, larger x-height. Slightly calligraphic with variable stress angle, bracketed serifs and bulbous terminals.

ALSO KNOWN AS: *transitional, réale, Baroque.*

USE AND IMPRESSION: Long text, newspapers, magazines, serious, expedience, authority, magazines, forward-thinking, charm, swagger, exuberance.

Perpetua

Ancient Geography

Plantin

Ancient Geography

Arnhem

Ancient Geography

Bookmania

Ancient Geography

Libre Baskerville

Ancient Geography

Mrs Eaves

Ancient Geography

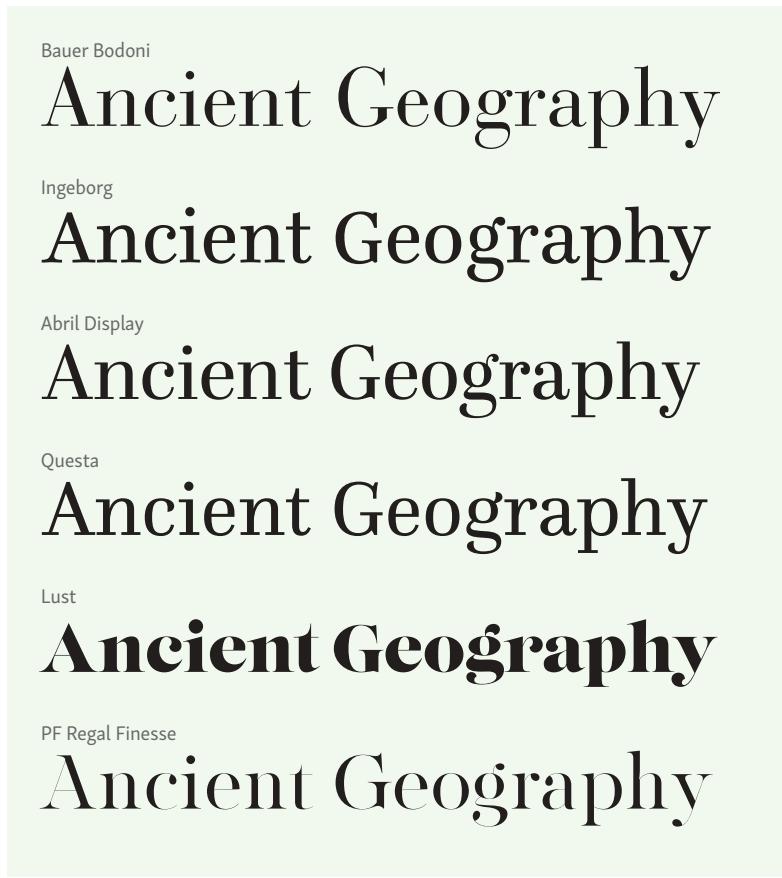
Transitional serifs.

3. Rational serif

Rational form, strongly modulated high contrast. Regularised structure with vertical stress. Often thin, unbracketed serifs and ball terminals.

ALSO KNOWN AS: *modern, didone, neoclassical, classicist, static, romantic.*

USE AND IMPRESSION: Headlines, elegance, stylish, fashion, wine, romantic, expensive, artisanal, reminiscing.



Rational serifs.

3a. Scotch

Much lower contrast with bracketed serifs and more robust features.

USE AND IMPRESSION: Long form text, newspapers

Abril

Ancient Geography

Georgia

Ancient Geography

Miller

Ancient Geography

Benton Modern

Ancient Geography

Bressay

Ancient Geography

Rational scotch serifs.

4. Grotesque sans

Rational form, modulated low contrast. Somewhat irregular, narrow proportions based on ovals rather than circles. Double-storey *a* and *g*.

ALSO KNOWN AS: *lineal, grotesque, grotesk, gothic, lineale, realist, industrial.*

USE AND IMPRESSION: Headlines, short text, lively, energetic, industrial.

Franklin Gothic

Ancient Geography

Aperçu

Ancient Geography

Questa Sans

Ancient Geography

Adelle Sans

Ancient Geography

Tablet Gothic

Ancient Geography

Marr Sans

Ancient Geography

Grotesque sans serifs.

5. Neo-grotesque sans

Rational form, low contrast. Large x-height, closed apertures, wider and more regular and circular shapes, usually horizontal terminals.

ALSO KNOWN AS: *lineal, neo-grotesque.*

USE AND IMPRESSION: All text and signage, neutral, grids, subtly informal.

Univers

Ancient Geography

Helvetica

Ancient Geography

National

Ancient Geography

New Transport

Ancient Geography

Activ Grotesque

Ancient Geography

LFT Etica

Ancient Geography

JAF Facit

Ancient Geography

Neogrotesque sans serifs.

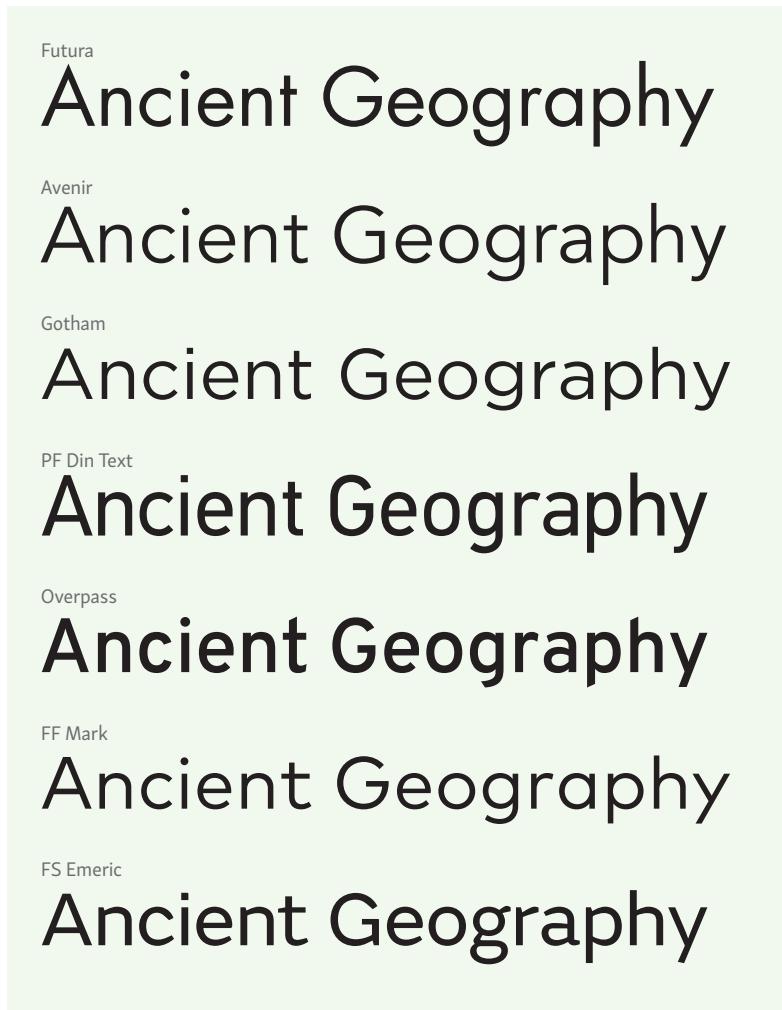
6. Geometric sans

Geometric linear form. Contrast is sufficient to optically correct but little more.

Shapes are based on circles or squares, with moderate apertures.

ALSO KNOWN AS: *lineal, geometric, static.*

USE AND IMPRESSION: Headings, longer text, modern, technical, mechanical, infographics, vintage, authority.



Geometric sans serifs.

7. Humanist sans

Dynamic form, moderate contrast. Calligraphic forms with open apertures.

Contemporary forms have lower contrast with very open apertures.

ALSO KNOWN AS: *lineal*.

USE AND IMPRESSION: Long text, fresh, warmth, charm, approachable , readability, energetic, cheerful, organic, active.

Gill Sans

Ancient Geography

Bliss

Ancient Geography

TheSans

Ancient Geography

Frutiger

Ancient Geography

Syntax

Ancient Geography

FS Albert

Ancient Geography

Akagi

Ancient Geography

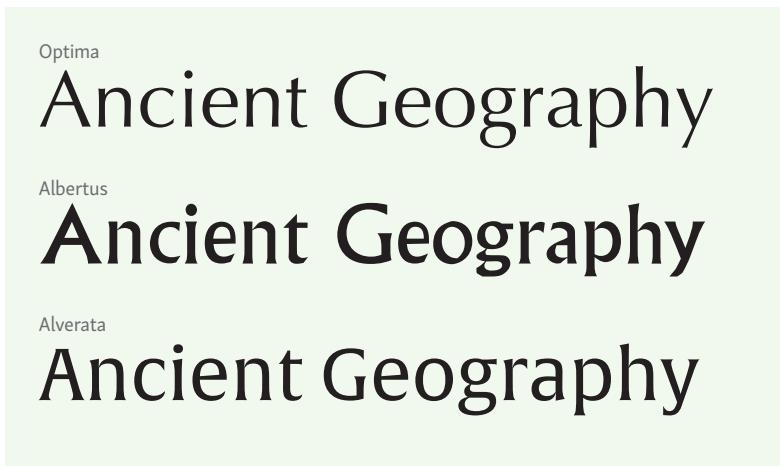
Humanist sans serifs.

7a. Glyptic

Low contrast with forms derived more from chiselling and inscribing than writing. Usually feature narrowed stems and flared terminals which almost become serifs.

ALSO KNOWN AS: inscribed, engraved, incised, flared, stressed.

USE AND IMPRESSION: Signage, headlines, official, ceremonial, establishment, warmth, wellness, cosmetics.



Glyptic sans serifs.

8. Grotesque slab

Rational form, low contrast. Similar to grotesque sans serifs, with the addition of heavy rectangular slabs, closed apertures and often ball terminals.

ALSO KNOWN AS: *mechanistic*, mechane, Clarendon, Egizian.

USE AND IMPRESSION: Headlines, short text, vintage, ephemera, powerful, industrial, smile.

Clarendon

Ancient Geography

Serifa

Ancient Geography

Century Modern

Ancient Geography

Sentinel

Ancient Geography

Grotesque slab serifs.

9. Geometric slab

Geometric linear form. Minimal contrast, with unbracketed rectangular serifs the same width as the stems.

ALSO KNOWN AS: *mechanistic*, mechane, egyptian, egyptienne, antique.

USE AND IMPRESSION: Headlines, short text, industrial, friendly, cookbooks, architectural, handmade.

Memphis

Ancient Geography

ITC Lubalin Graph

Ancient Geography

Rockwell

Ancient Geography

Archer

Ancient Geography

Bague Slab

Ancient Geography

Museo Slab

Ancient Geography

Geometric slab serifs.

10. Humanist slab

Dynamic forms with low contrast. Similar proportions to humanist sans serifs, with the additional of unbracketed rectangular or wedge serifs.

ALSO KNOWN AS: *mechanistic*.

USE AND IMPRESSION: All text, professional, business, approachable, informal.

PMN Caecilia

Ancient Geography

Chapparal

Ancient Geography

Adelle

Ancient Geography

FF Tisa

Ancient Geography

Guardian Egyptian

Ancient Geography

Humanist slab serifs.

Selecting typefaces for body text

The job of the typeface for the majority of text in a website is to get out of the way and let the words speak. It should be unobtrusive, so that eye-catching headings can stand out, and steady so that your reader can skim and flow as required. The choice of typeface also sets the tone for the text.

The features of the typeface may not be noticed or even discernible to the untrained eye, but they do register as they change the feel and texture of the text. Even if your reader is not consciously aware of the type they're reading, they'll certainly be affected by it.

Remove the friction between reader and text

While you can set the tone for a piece of text with your choice of typeface, the primary decision should be about removing as much friction as possible between your reader and the text. A smooth-reading typeface is one with instantly determinable letterforms and an inviting flow to usher your reader through the text.

Your reader shouldn't notice the type. They shouldn't squint, stumble, halt, or find cause to examine the type. If they do, their reading experience will have been interrupted and even cut short. Your prime directive is to ensure that you are not making choices which are difficult or uncomfortable to read.

Choose robust faces for better readability on screens

Typefaces designed for body text are intended to be read easily and smoothly at small sizes. To achieve this, their glyph shapes are made intentionally uncomplicated and sturdy. Screens are hostile environments for type to live, as described in '[How fonts render on screens](#)'. Type survives on screens

as it does in other challenging environments, like newspaper print and signage, with high x-heights, low contrast and few frills.

Small x-heights are problematic because they draw too much attention to capitals and extenders. Subtle and delicate letterforms stand little chance and are lost in the coarseness of rendering environments. Distinctive glyphs end up exaggerated and distracting.

Although modern screens have a pixel density capable of rendering intricate glyphs, the nature of emitted rather than reflected light eats into those forms. Robust shapes stand up to this bullying, leaving high resolutions to render any subtleties, thereby rewarding you and your reader in tempering the ruggedness of the type.

The categories newspapers turn to for robustness are Scotch transitional serifs and humanist slab serifs. Signage tends to use humanist sans serifs, particularly those which are more on the geometric side. These categories tend to produce low-contrast type, with open counters and sturdy terminals, all of which are ideal characteristics for maintaining readability on screens.

Even the most carefully planned design will fall short of perfection unless unremitting attention is paid to these details, these minor canons which have governed both the written and printed manifestations of the Latin script from the earliest times.

Even the most carefully planned design will fall short of perfection unless unremitting attention is paid to these details, these minor canons which have governed both the written and printed manifestations of the Latin script from the earliest times.

Top: Oxtail is a charming typeface but it's a bad choice for body text. Its high contrast and quirky serif styles make it feel distracting for running text.

Bottom: Abril Text has a relatively high contrast and is full of personality, but its slab serifs and consistent forms subdue any potential distractions.

Choose active texture and even colour for a smoother read

Your reader may be spending a great deal of time with the type, and if you have done your job well they could lose track of that time altogether. Such

a pleasurable reading experience is provided by a typeface with a smooth, active texture. If your reader was instead a *listener* settling in for a lengthy recitation, they wouldn't want to listen to a dull, monotonous, somnolent delivery. Neither would they want to endure a melodramatic rendition with histrionics and overacted enunciation. They would want a clear, engaging delivery with enough vocal variation to keep their interest, but not so much as to become tiresome and irritating.

An orator's vocal variation is equivalent to a typeface's texture. For tone, rate and volume, think contrast, flow and size. Texture is the apparent visual activity of a text block. Good body text typefaces have an active texture that is not too lively and not too dull. An active texture will have glyphs with a certain amount of contrast and flow from one character to the next. Textures which are too lively, caused by high contrast and complex shapes, make reading strenuous because they demand too much attention. Dull textures with no contrast or flow make distinguishing letterforms harder and require more effort to skip along a line of text.

Even the most carefully planned design will fall short of perfection unless unremitting attention is paid to these details, these minor canons which have governed both the written and printed manifestations of the Latin script from the earliest times.

Even the most carefully planned design will fall short of perfection unless unremitting attention is paid to these details, these minor canons which have governed both the written and printed manifestations of the Latin script from the earliest times.

Top: Museo Sans's shapes are sturdy and its colour is even, but its texture is fairly dull. As a multipurpose font it does a good job, but does little to energise the reader.

Bottom: FS Emeric has subtly chiselled terminals, an openness to its characters, and micro-modulations in weight, all of which invite the reader to skip along the lines.

Body text typefaces are also designed for an evenness of typographic colour. Typefaces with patches of dense blackness or whiteness, resulting from certain character shapes and combinations, add friction to the reading experience. Type with spotty colour along a line is often more upright and narrow,

as in low-contrast grotesque sans serifs. High-contrast rational serifs can suffer from patchiness within words if too many thin or thick strokes line up next to one another.

Texture and colour are inevitably influenced by aspects of the typesetting, particularly line height and word spacing. We can identify fonts with sturdy shapes and even colour, but a suitably active texture can only be achieved by setting the text block well, as described in '[Designing paragraphs: line spacing](#)'.

If the text you are setting is shorter – intended more for scanning and snacking than immersive reading – you might pique your reader's interest with the initial swagger of a more idiosyncratic typeface. Where such lively type might tire your reader over a long period, it might be just what is required to attract the attention for a few minutes.

Choose faces in keeping with the text

In 1955, Beatrice Warde, an American communicator on typography, published a series of essays entitled *The Crystal Goblet*, in which she wrote: 'People who love ideas must have a love of words. They will take a vivid interest in the clothes that words wear.' And with that proposition Warde introduced the idea that just as we judge someone based on the clothes they wear, so we make judgements about text based on the typefaces in which it is set. Typefaces are the clothes words wear, and just as the choice of clothing says something to us about the wearer, so the choice of typeface says something about the text and its author.

Your job is to choose typefaces that will honour and elucidate the character of the text. Think about the topic of the text but – more importantly – get to grips with the nature of the writing. Think of words that describe the way the writer is coming across; these are the feelings and moods you need to impart with your design. The text might be about a serious disease, but maybe the words provide an optimistic, self-effacing view of caring for someone; this would warrant type with a lighter touch, such as a humanist sans. Alternatively, the piece might be reporting academic research, in which case it would deserve a more serious, authoritative face, such as a transitional serif.

Perhaps the text you are designing is for the pharmaceutical industry. You could take many different approaches, each of which might be valid depending on the topic and nature of text, and whether it is in keeping with the wider brand and context.

- Chemical plants and engineering would lead you towards a face that is more industrial in feel, such as a grotesque slab serif.
- You could evoke the feeling of a chemistry lab through a simple and scientific face that you might find among the geometric sans serifs.
- Research findings would require something more grounded and academic, as you might find in a modern take on a humanist serif.
- To give the right feel for health and wellbeing, you would turn to a humanist sans serif.

Aspirin is a common medicine that has a number of uses, from relieving pain to reducing the risk of serious problems such as heart attacks and strokes. It comes in many forms, including pills, tablets that are dissolved in water, powders and oral gels. Some types can be bought

Aspirin is a common medicine that has a number of uses, from relieving pain to reducing the risk of serious problems such as heart attacks and strokes. It comes in many forms, including pills, tablets that are dissolved in water, powders and oral gels. Some types can be bought

Aspirin is a common medicine that has a number of uses, from relieving pain to reducing the risk of serious problems such as heart attacks and strokes. It comes in many forms, including pills, tablets that are dissolved in water, powders and oral gels. Some types can be bought

Aspirin is a common medicine that has a number of uses, from relieving pain to reducing the risk of serious problems such as heart attacks and strokes. It comes in many forms, including pills, tablets that are dissolved in water, powders and oral gels. Some types can be bought

Clockwise from top-left: grotesque slab serif (Serifa); geometric sans serif (FF Mark); humanist serif (Bembo); humanist sans serif (Bliss).

Returning to Warde's analogy, imagine an auditorium filled with the target audience of the web page. Now consider the text being delivered as a speech. What would the orator(s) be wearing and why? Referring to '[Knowing and browsing type](#)', how would that attire translate to typefaces?

In your final choice, remember that suits come cheap and shabby as well as finely tailored. The fit and the detail in the design make the difference. If your speaker wears jeans and a T-shirt to be casual and approachable, they needn't be scruffy and unkempt. Jeans and T-shirts can still be uncreased, well fitted, contemporary and sharp.

There's no such thing as a neutral typeface

You cannot make a choice that calls for a neutral typeface. This would imply your author is expressing no opinion, has nothing to say and their writing is deliberately soulless. You cannot wear neutral clothes. Boring, predictable, unadventurous and inoffensive, perhaps; but never neutral. Your decision to play it safe sartorially is nonetheless a decision and, by definition, not neutral.

Should you require something neutral, you probably mean the text calls for what some might describe as a 'boring' typeface – one which gets out of the way and lets the ideas shine through. Any successful boring typeface will have been carefully designed to be highly readable in a way that doesn't bring attention to itself. It will be quiet, reliable and helpful, with just enough texture to guide the reader without being noticed. That is not neutral and bereft of character; that is subtle and deliberately so.

Choose a face whose history and culture resonates with the text

Typefaces carry baggage from the time, place or reason they were designed, and the way they have been used in the past. They tap into a complex library of associations we've all been building up throughout our lives. You make more associations every time you see a font in a certain context. Typefaces can give a sense of place, evoke a mood or even refer to a specific period in history. Ensure your typeface's history and culture are harmonious with the subject matter of the text. These associations need to complement what you're trying to convey if the font is to work *with* the text and your design, not against.

Your approach can be quite simple. For overtly modern subjects, choose a recent typeface. For historical subjects, choose an older face (or one derived from an older style). You could even find a font that was created the same year the client's organisation was founded, although the chances of it being appropriate are slim.

If you are setting text written by a woman, you could choose a typeface designed by a woman¹; for example, Carol Twombly, Veronika Burian, Nicole Dotin, Freda Sack, Zuzana Licko and Nina Stössinger. That's not to say either the type or the text would be – or need to be – at all feminine, but there's an association which might help you whittle down choices.

¹ For more about – and from – women in type, see [Alphabettes](http://wbtyp.net/38) (<http://wbtyp.net/38>).

For text about the Netherlands you could choose a typeface from a Dutch designer. In this case you'd be spoilt for choice as anecdotally² there are more type designers per capita there than in any other country. Contemporaries include Erik van Blokland, Jos Buivenga, Luc(as) de Groot, Martin Majoor, Fred Smeijers and Gerard Unger, to name but six. You could also reach back four centuries to the original days of Dutch-taste humanist serifs and the work of Anton Janson. Clearly, not every font by these designers will be suitable for your purposes, or even Dutch in any recognisable way, but if you've got a Dutch-designed typeface on your shortlist it could become a real contender in this scenario.

The right font must primarily match all your practical and subjective criteria. Just as you wouldn't set a text about ancient Egypt entirely in Papyrus, if you are designing a text about early 19th-century Italy, instead of reaching for the nearest Bodoni, and losing all its high-contrast glory on a low-resolution screen, you might be better off with a more robust descendant, such as Dalton Maag's Bressay.

Do some detective work

The best way to find out if a typeface is appropriate or, more likely, inappropriate is to track down information about the typeface itself. Head to either the designer's or foundry's website, or their printed specimens (often beautiful, collectable and free). A typeface's background information reflects the tremendous amount of thought and research poured into the work. Designers will often outline why they created a typeface, from where they drew their inspiration, and what they intended the type to be used for.

Don't neglect to research how the fonts have been used in the past. Check books, magazines and useful websites like [Fonts In Use](http://wbtyp.net/52) (<http://wbtyp.net/52>) and [Typewolf](http://wbtyp.net/55) (<http://wbtyp.net/55>). There maybe associations with incompatible topics and styles of which you may be unaware. Try not to choose a typeface that transmits something that the text doesn't say. It shouldn't communicate something you don't want to communicate.

Make a specimen to reduce your shortlist

In 'Responsive paragraphs' we talked about testing your typographic design in a responsive prototype. Create a sampler to compare your candidate typefaces side by side.

² According to Dutch designer Gerard Unger in 'Dutch Type Design' (<http://wbtyp.net/10>) by Peter Bil'ak in *Typotheque articles* (2004).

One more attribute the modern typographer must have: the capacity for taking great pains with seemingly unimportant detail. To them, one typographical point must be as important as one inch, and they must harden their heart against the accusation of being too fussy.

One more attribute the modern typographer must have: the capacity for taking great pains with seemingly unimportant detail. To them, one typographical point must be as important as one inch, and they must harden their heart against the accusation of being too fussy.

One more attribute the modern typographer must have: the capacity for taking great pains with seemingly unimportant detail. To them, one typographical point must be as important as one inch, and they must harden their heart against the accusation of being too fussy.

One more attribute the modern typographer must have: the capacity for taking great pains with seemingly unimportant detail. To them, one typographical point must be as important as one inch, and they must harden their heart against the accusation of being too fussy.

One more attribute the modern typographer must have: the capacity for taking great pains with seemingly unimportant detail. To them, one typographical point must be as important as one inch, and they must harden their heart against the accusation of being too fussy.

One more attribute the modern typographer must have: the capacity for taking great pains with seemingly unimportant detail. To them, one typographical point must be as important as one inch, and they must harden their heart against the accusation of being too fussy.

One more attribute the modern typographer must have: the capacity for taking great pains with seemingly unimportant detail. To them, one typographical point must be as important as one inch, and they must harden their heart against the accusation of being too fussy.

One more attribute the modern typographer must have: the capacity for taking great pains with seemingly unimportant detail. To them, one typographical point must be as important as one inch, and they must harden their heart against the accusation of being too fussy.

One more attribute the modern typographer must have: the capacity for taking great pains with seemingly unimportant detail. To them, one typographical point must be as important as one inch, and they must harden their heart against the accusation of being too fussy.

One more attribute the modern typographer must have: the capacity for taking great pains with seemingly unimportant detail. To them, one typographical point must be as important as one inch, and they must harden their heart against the accusation of being too fussy.

One more attribute the modern typographer must have: the capacity for taking great pains with seemingly unimportant detail. To them, one typographical point must be as important as one inch, and they must harden their heart against the accusation of being too fussy.

One more attribute the modern typographer must have: the capacity for taking great pains with seemingly unimportant detail. To them, one typographical point must be as important as one inch, and they must harden their heart against the accusation of being too fussy.

Sampler with twelve candidate typefaces.

Narrow down your choices and experiment with them in more detailed browser-based specimens. Make a page containing different kinds of likely content as well as paragraphs: headings and subheadings, a list, a table, some captions. Also include some bold, italic and any other styles (such as small caps) which you might want to use.

The samplers and specimens you compile are to help you evaluate typefaces for body text. The example content you use should, therefore, be as real as possible. You should want to read a specimen's content. With well-considered, representative content in place, you'll be able to better focus on whether the typefaces properly represent the personality you're trying to evoke. At some point make sure you include the client or company name in your sampler. You don't want that to unexpectedly look weird.

You'll probably evaluate many different typefaces. In times gone by this could have been very expensive, but now most web font services and type vendors allow you to test with their entire library for very little cost (paying the full licence once you launch with your final decision).

Judge typefaces in context

Make your decision about typeface robustness, texture and colour in every context you care about. You need to ensure all your readers benefit from the effort and consideration you've put into type selection, no matter how they view your design. Load up your specimens on different devices.

Imagine how your audience will encounter the content. Your reader might be perusing their phone on the way to work, immersed in the text during a lengthy commute. If the content is interesting and comfortable to read, they might pick up the piece when they get to the office, this time on a desktop or laptop, and again at home on a tablet. All of these contexts and other likely scenarios should figure in your evaluation process.

Trust your personal preference

Your choice of typeface shouldn't *be* the design. It should serve the text and the typesetting, and be decided only when you know the nature of the job, as described in '[Practical and pragmatic considerations](#)'. Your final selection is a subjective choice made from a shortlist of worthy and appropriate contenders.

Proxima Nova is a fine typeface and a very common choice. There are, however, other possibilities.

Your process of elimination should never leave you with Proxima Nova on a shortlist of one. You may, however, come to the objective conclusion that a geometric sans serif with neo-grotesque attributes is fit for the job. If you then pick Proxima Nova from the pool of pragmatic possibilities, know that the final decision came down to your taste and preference. How you come to prefer one possibility over another is where the fun lies. A fascination with type means your interest can be provoked at any time. You might want to try a typeface for any number of reasons beyond the pragmatic ones laid out over the past two chapters. The name, its designer, because it reminds you of something special, because it's beautiful, because it's a challenge, because you bought it two years ago and haven't had a chance to use it in anger yet (see '[Building a library](#)'). These are all perfectly good reasons to choose a typeface on the proviso that it's not wrong for the job at hand.

Choosing typefaces for display

Display text is all about seduction. Its purpose is to draw your reader into the content. Effective typographic seduction is expressive and strong, and it takes determination and skill to make a firm aesthetic decision.

Don't be reverential, dogmatic or ordinary

Don't abdicate responsibility and opt for inoffensive blandness or perceived safety. Be appropriate, but allow your display type to have its own vernacular and effect.

Influence the way people feel through type

We established in '[Headlines and impact](#)' that visitors to your webpage will *see* display text before they *read* it. This is your chance to choose a typeface that immediately expresses what the text, and the entire website, stands for. Through font choice and typesetting you will create an ambience and subtly change the emotional state of your reader.

The way you set your type – the size, spacing, colour and context – will go a long way to forming a negative or positive experience. The quality of your typesetting can have a notable impact on how your reader feels, but the most significant contributor will be your choice of typeface.

When we choose typefaces for body text, we're driven mostly by practical decisions, aiming first for readability. The message the typeface sends is important but considered later, and will only ever be subtle. With display text the effect is magnified. Striking typography can make a big impression on your reader and can influence the way they actually feel.

Our conscious mind perceives type and interprets what it sees to understand what the words say. But type also communicates with our subconscious. Like all visual stimuli, type taps into the amygdala, an ancient part

of the brain which is hardwired into our systems of protection and survival. Buried deep inside each hemisphere, our amygdalae perform a primary role in the processing of memory, decision-making and emotional reactions. One of the amygdala's main functions in these systems is the formation of associations between our senses and our biological needs. When we experience something round and soft as sweet and good to eat, or something hard and jagged as bitter or dangerous, the amygdala holds on to that association. Each time the same experience is repeated, the association is reinforced and, through the amygdala, becomes instinctive.



Sarah Hyndman at [TEDxBedford](#). © TED Conferences, LLC.

Sarah Hyndman, an expert in multisensory typography, has investigated how typefaces communicate with our subconscious. Demonstrating with conference audiences, Hyndman showed that different fonts have an effect on how food tastes. A rounded font placed near a bowl of jellybeans would make them taste sweeter, and a jagged angular font would make them taste more sour. Although not scientifically rigorous, Hyndman has repeated the same experiment¹ on numerous occasions with the same result.

Your choice of typeface can, it would seem, directly affect the senses. If you can affect the senses, you can alter the way people feel. You can change their emotional state through type alone. This is a huge opportunity. Think how you might want your readers to feel, because you can make them

¹ See [Type Tasting](#) (<http://wbtyp.net/51>) for more details of Sarah Hyndman's work.

more conducive to feeling a particular emotion. Consider how you'd like to influence their mood: for a stop-smoking campaign you might choose a typeface to inspire them; for long-form journalism you could want them to feel intrigued; for a tropical getaway you'd want to invoke a sense of repose and reassurance.

Typography does not just convey information or impart feeling, emotion and sentiment. It also arouses preconceived ideas of trust, tone and content.

I hate obtuse theoretical mathematics

Font choice can set up expectations which are at odds with the content.

Your initial expectation of the above example, set in a cheeky rounded face, is more likely to be 'I love little fluffy kittens' than a complaint against a difficult discipline. Similarly, the following example may not be all it is expected to be:

This is a story about someone who lived in Happyland. His name was Mr Happy and he was fat and round, and happy. One day Mr Happy went for a walk in the woods.

Blackletter is not associated with children's books.

Your display text should connect people with the spirit and personality of the website, and establish a relationship with the story they are about to read. That might be a news story, the story of your product, or the history behind a charity.

Working with different kinds of display faces

When it comes to working with display typefaces, they can be divided into two broad categories: workhorses and personalities.

Workhorses

Workhorse typefaces can be bent to your will. They can be shaped in many different ways to perform in a wide range of identities and contexts; consequently, they are somewhat ubiquitous. Helvetica, Futura and Proxima Nova are good examples.

Workhorses are versatile but take a lot of practice and a deft hand to work with in a unique and striking manner. Think of them as you might a salmon fillet. It's a simple and flexible ingredient which you could roast, poach, bake, or fry in all manner of ways. But you will need a lot of skill and experience to turn it into a truly exceptional meal.

Many workhorse typefaces are members of larger font families, so-called superfamilies. Display faces within in superfamilies are the extreme weights and widths. If you have chosen a body text font which is part of a superfamily, you could look straight away to the display styles within that family, knowing that the two fonts will sit together well.

Reflects **Period** SINGLE **United** Category
 Surface **WHICH** Popular **Clearly** STATING
 FORCED **Earlier** Heavier **INSIDE** Designer
 Human Curved STATES Instead Variation
 Angles HEIGHT Display Though **BRINGS**
BELOW Shapes **Capital** BEGAN **Regards**
Elbow Ancient **BATHS** Existing **Graphic**
About LIGHTER **Lining** Warmth **ENTITY**

Acumin is a typical workhorse superfamily with a variety of widths and weights.

An advantage of picking display faces from within a superfamily is that it gives you the opportunity to choose a different style for a different size screen. As described in ‘[Headlines and impact](#)’, for larger screens you might want to proportionally increase the size of the your display text compared with the body text. Choosing a more condensed style under these circumstances means you can increase the text size without making it uncomfortably wide. The following example has sized the text as 10vmin and switches the style from condensed to extra-condensed if the screen is wider than 80em:

```
h1 {  
  font-family: "Acumin Condensed", sans-serif;  
  font-size: 10vmin;  
}  
  
@media screen and (min-width: 80em) {  
  h1 {  
    font-family: "Acumin Extra-Condensed", sans-serif;  
  }  
}
```

Personalities

The second kind of typeface is one that does most of the work for you. These are typefaces with much more inherent character and flavour. Like finely tailored clothing, the detail in their design adds interest to the innate functionality. Personalities do more for you straight out of the box, but are less malleable. It’s harder to adapt them to different identities or contexts.

We’re not talking readymades like scary halloween fonts or party fonts made from balloons. *Personalities* are quality fonts providing you with a more esoteric ingredient. Consider them as a locally caught whole turbot compared with a pack of farmed salmon fillets (but without the price difference). The salmon is a tasty ingredient that can be integrated into most forms of cuisine, whereas the turbot would be the centrepiece around which you would build a meal. A turbot will go a long way towards you creating something exceptional, but is suited to fewer cooking methods and accompaniments.

The south wall glowed in a brilliant luminescence

Marr Sans (*above*) is a slightly eccentric sans serif that lends character straight away and provides editorial design with a sharp and distinctive taste.

The five myths about contemporary music

FS Clerkenwell (*above*), based on influences from an area of London, is a slab serif which provides a quirky, contemporary twist.

Why we should give free money to everyone

Bree (*above*) lends headlines character that would simply be lost should they be set in Helvetica.

Favour personalities over flexibility

Both kinds of display face have their place - workhorses for flexibility, personalities for individuality. Be aware of which one you're using and you can treat it accordingly.

The best guides to specific treatments benefitting personality fonts can be found in foundry specimens.

For the sake of readers and consumers of content the world over, it would be preferable to see more personality typefaces being used. Like most media for design, the web breeds homogeneity. Personalities bring their own idioms and sprinkle the online world with individuality and charisma. They make it easier to set yourself apart from the rest. Better still, with a little respect and understanding you can just let the typeface do most of the job.

Use display styles for display text

More sophisticated type families will have specific display styles which are more than just extreme widths or weights. Display styles are variations on the body text, specially adjusted for setting at large sizes. The variations can be quite extreme. For example, Joshua Darden's Freight family has five optical sizes. Comparing Freight Macro (designed for use on screen at small sizes) with Freight Big (designed for use at large sizes) reveals that the display style has a greatly increased contrast and more delicate detailing, which would be lost at small sizes. Conversely, the large x-height and crude detailing of the text face would look clumsy and uninviting at large sizes.

Freight Big
Freight Text
Freight Macro

Freight

A typeface family
designed with five
different variations for
use at different sizes.

By Joshua Darden

Three of Freight's five optical variations.

The most expressive display styles exaggerate design features of body text. Questa Grande, designed by Martin Majoor and Jos Buivenga, beautifully enhances the fine curves, flicks and curls hinted at in the text styles.

aq → aq

Questa Grande exaggerates Questa text styles.

Abril, designed by Veronika Burian and José Scaglione, takes this approach even further. Its text and display styles are derived from different styles but designed to work together in harmony. Abril Text takes its inspiration from both 19th-century slab serifs and Scotch Roman typefaces – it is essentially a rugged newspaper font. Abril Display applies the proportions of the text face to a seemingly different style, that of a rational serif, enabling it to hold the reader’s attention with measured tension in its curves and high contrast.

The curious gardener

The curious gardener

Abril Display (*top*) and Abril Text.

Choose display faces that reinforce the structure of the text face

Display styles are an exaggeration of the text face. Sometimes this involves amplifying a design feature; other times it might be exaggerating a width or weight within a superfamily. But if no specific display style exists, you can look for other typefaces which magnify aspects of your text face.

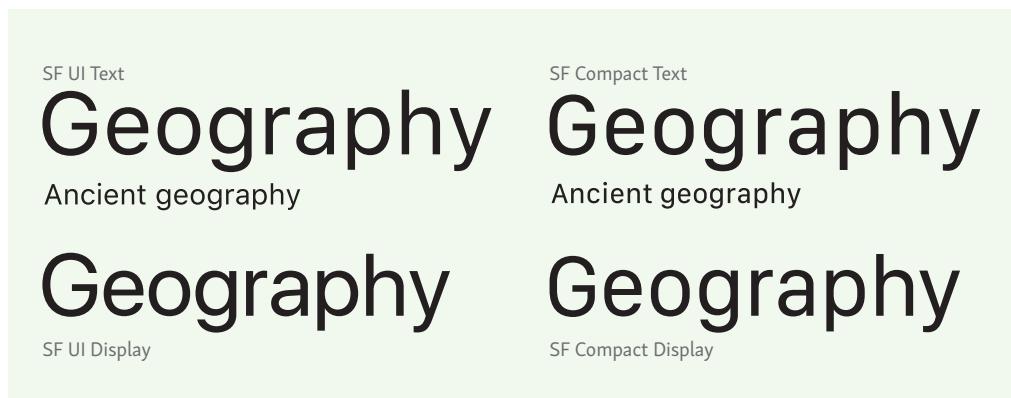
Look for similar shapes and inner structure by choosing fonts from the same category. As illustrated in ‘[Knowing and browsing type](#)’, Bodoni and Baskerville are both rational serifs, but with its high contrast, Bodoni is an exaggeration of Baskerville. Fira Sans doesn’t have a display style, but both it and Bree are dynamic sans serifs, meaning Bree could be a sophisticated companion to the more workaday Fira Sans text face.

Optical sizes

The display styles we’ve looked at so far are significantly and deliberately different to their corresponding text styles, taking advantage of bigger sizes to offer more delicate and complex forms. Even without any changes, the same font will look different at different sizes. In the days of metal type, subtle adjustments were made to the glyphs at each font size to ensure style and legibility of the typeface were retained throughout the various sizes. As the letters got smaller, spacing was increased and detail was simplified to maximise legibility once printed small.

In the digital age, we can make type any size we want, but this does not take into account how a font will look at that size. A digitised font uses the same outlines regardless of the rendered size. Type designers are starting to solve for this again by creating multiple versions of typefaces which optically correct for type used at different sizes.

Apple's San Francisco typeface was introduced in 2015 as its operating system font. San Francisco has two optical sizes, Text and Display. The operating systems automatically switch between these two versions at 20 pt (at 144 dpi). The Text version has increased letterspacing to aid legibility at small sizes, and there are subtle adjustments made to the outlines of each glyph so they look best at their intended sizes. With San Francisco on Apple's operating systems, the letterspacing is size-specific: each font size is intended to be used with a specific tracking value, which is automatically applied by the operating system. You can specify San Francisco for use on a web page (taking advantage of the optical features) but to do so is not straightforward, as explained in detail in '[Choosing typefaces for functional text](#)'.



San Francisco's optical adjustments.

Microsoft commissioned Matthew Carter to create a typeface for reading on screen, specifically as part of the 'reading view' mode in Internet Explorer 11. The Sitka family has four styles, each available in six optical sizes: Small, Text, Subheading, Heading, Display and Banner.

Each optical style is specifically optimised for a size range and use case. For image captions, reading view uses Sitka Small, which is designed with thicker strokes, larger x-height, and looser letterspacing. Sitka Banner, designed with thinner strokes and tighter letterspacing, is used for article titles.

Banner

Tunguska

Display

Enormously

Heading

powerful airburst

Subheading

explosion in Siberian
wilderness, June 1908

Text

At around 07:17 local time, Evenks natives and Russian settlers in the hills northwest of Lake Baikal observed a column of bluish light, nearly as bright as the Sun, moving across the sky.

Small

The sounds were accompanied by a shock wave that knocked people off their feet and broke windows hundreds of kilometres away. The majority of witnesses reported only the sounds and the tremors, not the sighting of the explosion. Eyewitness accounts differ as to the sequence of events and their overall duration. The explosion registered on seismic stations across Eurasia. In some places the shock wave would have been equivalent to an earthquake of 5.0 on the Richter scale.

Optical sizing in Sitka (based on a sampler by Stephen Coles).

Keep inspired, and play

Choosing a typeface for display purposes is very subjective. Because you are setting the text relatively large, there are fewer practicalities to adhere to. It will help you enormously if you know the precise wording. The subject matter and, more importantly, the tone will give you a sense of direction for the feeling you need to set up.

When you are browsing and searching for fonts, think about the kinds of emotion the text should be conveying and provoking. Should it give a sense of childlike enthusiasm? Businesslike authority or edgy discomfort? Should it be cutting-edge? Hipster, futuristic, sensible and secure? How should the reader feel? Excited, comforted, reassured, tense?

Once you've found some candidates for a good direction, make a sampler. Plug them into a prototype and try variations to fine-tune your choice. Look at your display faces alongside the body copy. Check the rendering at different sizes and on different devices to ensure the large display fonts scale down well. Experiment and play – as in many walks of life, it's the best way to learn.

Choosing typefaces for functional text

Functional text is a catch-all term for any type which has a specific purpose other than being display or body text. This includes user interface elements, navigation and error messages; labels, such as captions and footers; and specialised content including asides and data visualisation.

Clearly distinguish functional text from body text

Functional text has different, usually discrete purposes. Using context, positioning and colour will distinguish functional text from body text effectively, especially when combined with the instinctive tactic of setting it smaller than body copy. Functional text is subordinate to primary content, and so reducing its size makes that relationship clear.

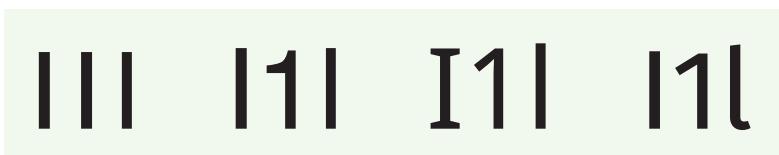
In print, captions and labels can function at very small sizes. Setting captions at 6–8 points – half the size of the body text – is not uncommon. On screens, even at high resolutions, type that small would likely be illegible to many people. As a rule of thumb, set functional text no smaller than 75% of the body text.

Functional text is usually short. It may be just a single word, or as much as a couple of sentences. Small amounts of text are not as sensitive to their setting as long-form reading. They can be scanned and interpreted quickly, hence normal guidelines of paragraph setting can be loosened somewhat. You can set short passages with tighter line spacing and very short lines. You can also set them very wide if necessary, as it's far less of a problem for your reader's eyes to reattach if there are only two lines of text.

Choose open, distinct letterforms for functional text

Functional text needs to be glanceable and effective when set small. It should be set in a typeface which doesn't scream for attention, but goes unnoticed. It should be compatible with the rest of the type on the page, but stay out of your reader's way. It should honour the content in a way that doesn't add to your reader's cognitive load.

Your reader will have fewer contextual clues when processing functional text, so it is vital the typeface has readily distinguishable characters. Your reader shouldn't have to stop to discern if a capital *I* is a lowercase *l* or number 1. This is less of problem in running text where your reader has more context to help distinguish words. It becomes far more relevant when your reader encounters individual words, or mixed numbers and letters, in labels, serial numbers or data.



L-R: Gill Sans and Helvetica have poor distinctions, unlike Droid Sans and Fira Sans.

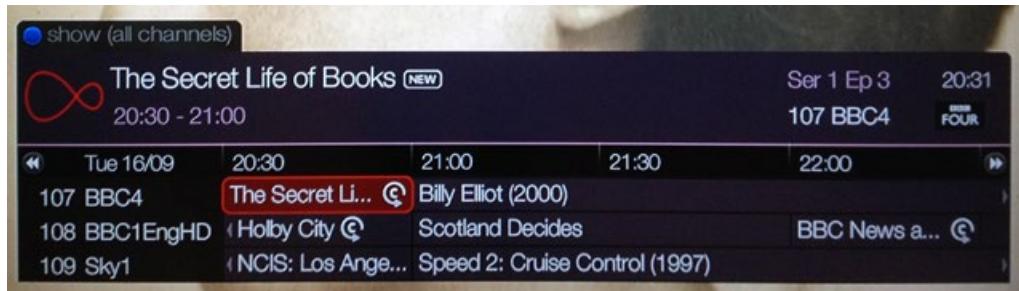
Small text, especially on screens, requires a typeface which is simple and unfussy. This does not specifically exclude serifed fonts, although slab serifs have a more usable and robust nature that holds its own at smaller sizes. Functional type needs to be flexible and work well in different sizes and a multitude of devices. Sans serifs tend to fare the best in this regard.

Typefaces with a large x-height are generally easier to read, and render better in small sizes. You still need to look for distinct ascenders, though, as too large an x-height will make the letters *n* and *h* difficult to distinguish.

For type to work well at small sizes, generous letterspacing is required. Letters that are too close to one another can be hard to tell apart and read. A good functional typeface should have enough breathing room between letters, but resist applying letter-spacing; instead, choose a correctly designed font.

Don't choose light weights for functional text. Interfaces should look elegant, but however much you might prefer the aesthetics of a light font, usability must always win out. Neue Helvetica Light, for example, is a poor choice for a user interface such as an electronic programme guide (EPG); the

letterspacing is tight, the thin strokes either blur unpredictably or dissolve into the background, and it has too many indistinguishable character forms.



'Billy Elliot' set in Neue Helvetica Light is almost unreadable in this EPG.

Your safest approach is to look towards typefaces which were designed specifically for signage or user interfaces. Examples include:

Frutiger

Choose open, distinct letterforms for info text.

Parisine

Choose open, distinct letterforms for info text.

Fira

Choose open, distinct letterforms for info text.

Lucida Grande

Choose open, distinct letterforms for info text.

New Transport

Choose open, distinct letterforms for info text.

Wayfinding Sans

Choose open, distinct letterforms for info text.

DIN

Choose open, distinct letterforms for info text.

Overpass

Choose open, distinct letterforms for info text.

San Francisco

Choose open, distinct letterforms for info text.

New Rail Alphabet

Choose open, distinct letterforms for info text.

Typefaces designed for signage and interfaces.

All of these faces have their own character, but share the same practical characteristics of sturdy simplicity and easily distinguishable, well-spaced characters.

Some typeface families have special caption styles. Designed for setting small, these are often wider with a taller x-height than the text face. One such example is PT Sans Caption:

PT Sans Caption

This is a caption.

PT Sans Text

This is a caption.

PT Sans Narrow

This is a caption.

PT Sans Caption, Text and Narrow compared.

Captions often have to be set to a narrow measure, such as in this book. In these instances a condensed font may be more practical than a wider caption-specific face, as you'll be able to get more words per line. Most faces designed for signage have been drawn with additional narrow and condensed styles; for example, Skolar Sans:

Skolar Sans Extended

This is a caption.

Skolar Sans Text

This is a caption.

Skolar Sans Condensed

This is a caption.

Skolar Sans Extended, Text and Condensed compared.

System UI fonts

Functional text can often be served well by the system's inbuilt user interface fonts. These tend to be very well designed, specifically for use at relatively small sizes, and modern operating systems have increasingly sophisticated type rendering mechanisms; for example, Apple's automatic tracking adjustments of San Francisco as described in '[Choosing typefaces for display](#)'.

One way to specify a system's UI font in your CSS is to list as many different operating system font names as you can in a `font-family` rule:

```
caption { font-family: -apple-system, BlinkMacSystemFont,
  "Segoe UI", "Roboto", "Oxygen", "Ubuntu", "Cantarell",
  "Fira Sans", "Droid Sans", "Helvetica Neue", sans-serif;
}
```

At the time of writing, this will work, but it's not at all future-proof as you'll need to update it every time a target operating system changes its system font.

The first two font names in the list show a promising direction, in that they are proxies for whatever the system font is on a Mac. This approach has been available since CSS 2.1 by making system fonts selectable through the `font` shorthand. The values are single keywords which set the font family, size, style and weight at the same time; for example, `font:caption`. The intended use was (and is) for designers to be able to reliably replicate the current system's UI components, including the user's preferences. The allowed keywords are:

- `caption`: the font used for captioned controls (e.g. buttons, dropdowns, etc.).
- `icon`: the font used to label icons.
- `menu`: the font used in menus (e.g. dropdown menus and menu lists).
- `message-box`: the font used in dialog boxes.
- `small-caption`: the font used for labelling small controls.
- `status-bar`: the font used in window status bars.

For the most common operating systems, the font and size used is the same across all keywords, with the exceptions of `small-caption` and `status-bar` being smaller. For general use, the `message-box` keyword may be the most appropriate as it is defined for dialog box text rather than a more specific purpose, such as an icon label.

The font shorthand can only be used with the system font keyword on its own. If you add other values, such as font size, it will assume that message-box is the name of a font rather than a special keyword. To control the size or style of the system font in your web page, set it afterwards.

```
caption {  
    font: message-box;  
    font-size: 1.2rem;  
}
```

One substantial thorn in the side of this technique (at the time of writing) is Apple's iOS. Mobile Safari does not recognise CSS 2.1 system font keywords. Instead, it uses its own keywords¹. If you repeat the font property (the order is important) you can get system fonts to work on iOS and all other modern browsers:

```
caption {  
    font: message-box;  
    font: -apple-system-body;  
    font-size: 1.2rem;  
}
```

¹ For full details see '[Using the System Font in Web Content](http://wbtyp.net/23)' (<http://wbtyp.net/23>) by Myles Maxfield on the *Webkit blog* (2015).

Combining typefaces

Combining different typefaces gives our designs graphical and emotional variety, and aids the communication of hierarchy and meaning. But it can be fraught with danger. Pair the wrong type together and you'll end up with awkward juxtapositions and a visual cacophony, confusing the message and muddying the text. Get a pairing right, and you clarify and enliven the entire reading experience.

Combine typefaces for a reason

The more typefaces you use, the more voices you have vying for attention. Sometimes simplicity can be the best approach, with the virtuoso solo performance shining through; at other times you need the harmony and dynamics of a full choir. A general rule is to use as few typefaces as possible in a design – two or three at most. But if you have good reason, choose wisely and are relentlessly consistent in using each typeface for a specific purpose, you can use a far bigger palette.

Use different typefaces to group similar types of text under a common visual system. You might choose different faces for display, body and functional text, the different purposes having different demands on the type. You might also choose an alternative typeface to give a unique voice to quotes and introductions. Think about how you are building up the different sections of a choir, the colours on a palette, or the ingredients of a dish. All the typefaces you use must contribute harmoniously and meaningfully to the whole.

The preceding chapters on selecting typefaces for **body text**, **display text** and **functional text** explain how to choose for these purposes, and should leave you with a shortlist for each of the different kinds of text in your composition. This chapter will help you make your final decision by considering which of your choices will sit best together.

Start your combination with an anchor typeface

Start with one typeface to help set a uniting voice and act as a reference point for subsequent choices. This could be a unique display face, a predetermined logotype or masthead font, or – preferably – the body type.

Your body text represents the majority of the content, and your choice of typeface for this will ultimately be based on an emotional decision to match the tone and message of the text. Use that emotion to guide you as you narrow down your shortlists.

Anchoring your decisions to a single typeface means you can look really closely at that choice in detail. The geometry, the contrast, the proportions and particular features of the glyphs can all lead you towards good matches.

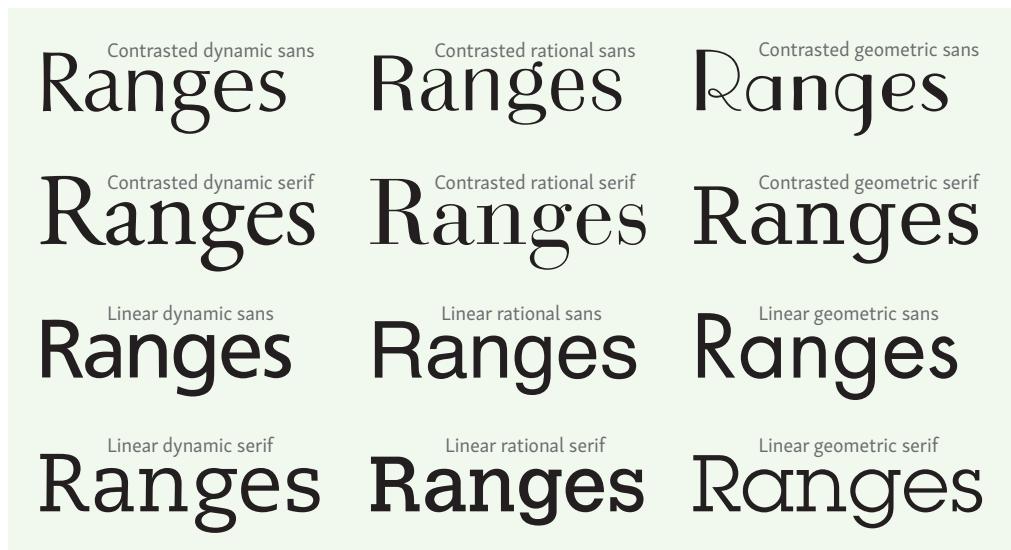
Find typefaces that complement the anchor

Smart combinations are the key to typographic interest. Musical notes which are adjacent clash and cause dissonance. Notes which are further apart allow harmonies to emerge, particularly if taken from the same scale. A choir of typefaces needs both distinction and polyphony. Choose typefaces that don't compete too much with each other, but aren't so similar as to be barely indistinguishable.

Combine typefaces with similar structure and form

Harmonious typefaces have innate relationships in their underlying structure, and complementary forms which strengthen connections within your design. Pair typefaces with similarities to create a visual consistency that will not distract your reader from the writing itself. To help do this, refer to the matrix of geometry and contrast we introduced in '[Knowing and browsing type](#)' (repeated opposite). Typefaces in each column share the same structure and will go together well. Successfully pairing faces along rows is much more difficult.

Try to find typefaces with common structures first, then look for other aspects of the letterforms. Enlarge the type and look at specific letters from each of the typefaces you might combine. Look for similar x-heights, cap heights, ascenders and descenders. Compare the counter in an *e* or *a*, the bar on a *T* or the loop of a *g*, the terminals of an *f* and the middle drop of an *M*. Not all features will be the same, but because you've picked the same underlying form, the inherent compatibility will still be there.



Typefaces ranging from dynamic to geometric (*left-right*) and contrasting to linear (*top-bottom*).

Remember that compatibility and similarity are not the same thing. For example, a square and a circle are very different shapes, but both are strict geometric forms that have more in common with each other than either has with a more calligraphic shape.

Let's say you are designing an Italian restaurant menu and you decide to consider Italian typefaces. If you were to look at Bodoni and Eurostile, you would have two completely different typefaces, separated by hundreds of years and seemingly connected by country of origin and nothing more. And yet they work together. There's a reason and it's down to geometry. Bodoni was designed with a strict vertical stress giving it many symmetrical shapes that are echoed in the linear geometric form of Eurostile.

Eurostile was designed in 1962 by Aldo Novaresi. It works pleasingly well at text sizes.

antipasti

TOP ITALIAN BREAD SELECTION £3.25

Our own focaccia with extra virgin olive oil and the finest aged balsamic. Baked twice daily. Served with a selection of wonderful breads from Jamie's artisan bakery including 16-hour sourdough and crisp music bread with rosemary and lemon gremolata.

Bodoni and Eurostile working in harmony.

Find complements through contrast

When you pair faces from extremes in the same column of the matrix, you are choosing faces with the same skeleton but opposing high and low contrast. You can choose other scales, particularly weight and width. Pairing fonts diagonally on the matrix provides additional contrast through form. Combine contrasting typefaces to reduce monotony, increase scannability and aid distinction.

Another effective form of contrast is to pair a serif with a sans serif. To ensure harmony, look for pairings which share the same skeleton. A rational serif such as Bressay will sit happily with a rational sans serif like LFT Etica.

The curious gardener

In 1634, anyone who spent a day at John Tradescant's house and garden in south Lambeth could discover more curiosities than a man might see in a lifetime of travel. Tradescant's 'Ark', a must-see attraction for half a century, was Britain's first museum.

LFT Etica and Bressay: sans and serif with the same skeleton.

Colour and proportion

When combining fonts it can be important they be close in colour, especially if mixing them up in context; for example, inserted code samples within sentences or using serif subheadings with sans paragraphs.

When you combine two different typefaces, consider adjusting their sizes. Even though two typefaces have the same body, seldom will they have the same size. For example, pay attention if inserting bold sans serifs of one typeface into serifed running text set in another typeface.

Stick with a single type family

Superfamilies provide you with typefaces consisting of a ready-made variety of contrasting styles, widths and weights, all guaranteed to work together. They are the easiest and safest place to look for pairings. Combining a regular text font with, for example, a condensed black weight will introduce radical but complementary contrast. Choosing a family with lots of styles gives you the flexibility to choose styles from just one source, knowing that they all have design traits in common – design traits you can get to know well. It keeps things simple while still giving you the opportunity to add hierarchy and character.



Univers was one of the first superfamilies and is available in a multitude of styles.

Some superfamilies include sans and serif siblings designed to share a set of consistent traits and play together nicely. One great example is Luc(as) de Groot's ground-breaking Thesis superfamily, which consists of sans, semi-serif, serif, slab serif and Arabic faces in a huge variety of weights, widths and styles.

The curious gardener

In 1634, anyone who spent a day at John Tradescant's house and garden in south Lambeth could discover more curiosities than a man might see in a lifetime of travel. Tradescant's 'Ark', a must-see attraction for half a century, was Britain's first museum.

TheSans and TheSerif are harmonious siblings.

Questa was designed as a multifaceted font family from the start and now has four variations. We've already seen how the display style, Questa Grande, was derived from the serif text face. Questa Sans sits perfectly with the serif text face too, as it was the logical outcome of a process of cutting away the hairline serifs, adjusting the contrast, and optically correcting its shapes. Questa Slab has slab serifs added to the sans to complete the set.

Change
Change

ang

Questa Sans was derived by cutting away the serifs and optically correcting.

Consider pairing typefaces from the same designer

Many type designers leave personal stylistic thumbprints on their typefaces, regardless of the style, purpose or historical influence on the design. A canonical example are Eric Gill's Joanna and Gill Sans, which share a very similar skeleton but were not specifically intended as a pairing.

The curious gardener

In 1634, anyone who spent a day at John Tradescant's house and garden in south Lambeth could discover more curiosities than a man might see in a lifetime of travel. Tradescant's 'Ark', a must-see attraction for half a century, was Britain's first museum.

Gill Sans and Joanna.

Another fine example is Neil Summerour's Akagi and Magneta. These both have humanist skeletons influenced by Garamonds. Having the same x-height and similar letterforms with right-leaning vertical stress, they sit well together. It's certainly not guaranteed that one designer's typefaces will combine effectively, but it's a good starting point.

The curious gardener

In 1634, anyone who spent a day at John Tradescant's house and garden in south Lambeth could discover more curiosities than a man might see in a lifetime of travel. Tradescant's 'Ark', a must-see attraction for half a century, was Britain's first museum.

Magneta and Akagi.

Pair typefaces from the same foundry

Often typefaces from the same foundry go well together. In some cases, one typeface might have been designed with another one in mind; in other cases, a foundry might identify suitable pairings in its catalogue. For example, TypeTogether has combination recommendations for all of its typefaces, saying of Abril Text and Tablet Gothic that “both fonts share the same vertical stress, large-x-height and mechanic feel.”

The curious gardener

In 1634, anyone who spent a day at John Tradescant’s house and garden in south Lambeth could discover more curiosities than a man might see in a lifetime of travel. Tradescant’s ‘Ark’, a must-see attraction for half a century, was Britain’s first museum.

Tablet Gothic paired with Abril Text.

Deciding on type combinations comes down to experience and your eye. You might instinctively think two fonts look wrong together without being able to identify precisely why. That’s fine: cross off that pairing and try the next one. Don’t forget to revisit your work. Fresh eyes and renewed energy almost always result in better judgement.

Using web fonts

Using web fonts can be a simple matter, but their potential impact on page load means there are two particular aspects you should optimise when using them: payload and rendering timing.

When we render a web page using a font on a server rather than one which is installed on our reader's device, that is known as a web font. There's no fundamental difference between it and any other font, other than the way the font file is accessed. However, because they are delivered from a server, web fonts can increase the time it takes for a page to download and render, hence the scope for optimisation.

Font-linking basics

Linking to font files to render text in web pages has been technically possible since 1998, when the specification was added to CSS2 and Microsoft introduced the original web font format, Embedded OpenType (EOT), for Internet Explorer. For the next eleven years no other browser supported the standard way to embed fonts, in part because manufacturers didn't want to support the DRM-laden EOT format.

Come 2009, Apple changed that. A new version of Safari was launched which supported regular TTF and OTF files as web fonts. This was soon followed by Firefox and Opera. The same year also saw the prompt launch of Typekit's and Fontdeck's web font services as a way to reconcile web designers' intense desire for web fonts and type foundries' deep concern for the protection of their intellectual property. The age of web fonts had begun.

In 2010, the Web Open Font Format (WOFF) was introduced as the standard web font file format, and was soon supported by all browsers. WOFF is a wrapper for TTF and OTF files. It enables those file formats to be compressed while not being made available for installation directly onto a desktop. In 2014, WOFF2 was introduced as a new standard format with improved compression.

To implement web font linking, use the `@font-face` rule, specifying the font family name (which can be anything you like), the font file URL

and format, and optionally the weight and style of the font. For example, to link to the Light Italic and Regular styles of a font you're calling 'Nicefont', write this:

```
@font-face {  
    font-family: 'Nicefont';  
    src: url('nicefont_light-italic.woff2') format('woff2');  
    font-weight: 200;  
    font-style: italic;  
}  
  
@font-face {  
    font-family: 'Nicefont';  
    src: url('nicefont_regular.woff2') format('woff2');  
    font-weight: normal;  
    font-style: normal;  
}
```

Specify the styles of Nicefont for subheadings and paragraphs as you would any other typeface.

```
h2 {  
    font-family: 'Nicefont', sans-serif;  
    font-weight: 200;  
    font-style: italic;  
}  
  
p {  
    font-family: 'Nicefont', sans-serif;  
    font-weight: normal;  
    font-style: normal;  
}
```

Fonts are only downloaded as and when they are needed. In the example above, if there is no `<h2>` on the page the Nicefont Light Italic won't be downloaded.

Because WOFF2 is a relatively new standard, it does not have universal support, so you need to add support for WOFF files. The `src` property of the `@font-face` rule accepts a comma-separated list of files. The order is important as a browser will traverse the list, skipping formats it doesn't

support and stopping when it finds a file it can download successfully. To provide a WOFF back-up to WOFF2, add separate `url` and `format` values to the `src` property:

```
@font-face {  
    font-family: 'Nicefont';  
    src: url('nicefont_regular.woff2') format('woff2'),  
        url('nicefont_regular.woff') format('woff');  
    font-weight: normal;  
    font-style: normal;  
}
```

There is very little need to support older formats, including EOT. WOFF is the standard and has been supported by all the latest desktop browsers since 2011, and by all the latest mobile browsers since 2013.

Don't specify local fonts without due diligence

If the `font-family` name you give your web font is the same as a font family available in your reader's environment, the existence of that local font is ignored. This permits you as a designer to freely choose `font-family` names without worrying about conflicts with installed fonts.

Should you want to specify a locally available copy of a given font, and download it should it be unavailable, add `local()` to your `src` list:

```
@font-face {  
    font-family: MyGentium;  
    src: local(Gentium),  
        url(Gentium.woff2) format('woff2'),  
        url(Gentium.woff) format('woff');  
}
```

The font name that you use with `local()` is a format-specific name that uniquely identifies a single font within a larger family. For OpenType and TrueType fonts, this string is used to match either the PostScript name or the full font name. Which type of name is used varies by platform and font, so you will need to include both of these names to assure proper matching across platforms.

```
@font-face {  
    font-family: MyGentium;  
    src: local(Gentium Bold), /* full font name */  
        local(Gentium-Bold), /* PostScript name */  
        url(GentiumBold.woff2) format('woff2'),  
        url(GentiumBold.woff) format('woff');  
    font-weight: bold;  
}
```

Don't specify a local font if you cannot be sure of its quality. Fonts end up on people's machines through all sorts of routes, so there is no guarantee that the copy of Helvetica sitting on your reader's PC will render at all well – it might be completely bereft of hinting, for example – in which case it will be disastrous for you and your reader. You are probably on safe ground if you're in a corporate environment with fonts controlled by a central server, or if you are specifying a bespoke typeface distributed to staff which you know to be suitable for your purposes.

Reduce your payload

Limit the number of fonts you use

Web fonts add to the weight of a page. A typical WOFF could be anywhere between 20 Kb and 100 Kb, for a single style and weight (Asian fonts can be an order of magnitude bigger). Complex designs increase file size, as do OpenType features, kerning tables and hinting instructions. An individual font is probably smaller in file size than any photo on a page, almost certainly a lot smaller than an advertising tracker, and significantly lighter than a JavaScript framework. But in combination, font files can add up, so limit the number of fonts you actually need. For example, you may have chosen a different font for your navigation, but ask yourself if that was really necessary, and whether the body text could be a suitable compromise (the answer might well be no).

Always provide a WOFF2 option

As mentioned earlier, WOFF2 files offer improved compression over WOFF, while still allowing fast decompression, even on mobile devices. The average saving over WOFF is 30%.

Subset as much as you can, but no more

When you subset a font you remove some of the glyphs with which it was originally designed. Subsetting can be a highly effective way to reduce load (often cutting files in half). Candidate glyphs for removal include unwanted OpenType features and characters not required for the language in which the text is written. Subset too aggressively, though, and you'll end up with the same problems we discussed in '[Practical and pragmatic considerations](#)', such as this abomination:

**Strongman Hafþór Júlíus Björnsson
is launching his own brand of vodka.**

Mixing typefaces due to missing glyphs.

Most type vendors and web font services now provide subsetting as part of their offerings.

Optimise page render timing

Avoid the flash of invisible text

Font linking is pretty simple and straightforward to implement, especially now the necessary format options are reduced to WOFF and WOFF2. However, it does lead to one particular quandary for web browsers – what to do with the web page text while the web font file is downloading and recompiling. This boils down to three choices:

1. Don't render the web page at all until the web fonts have loaded.
2. Render the page but hide the text until the requisite web font has loaded.
3. Render the page and use a fallback font for the text, swapping it out when the requisite web font has loaded.

The good news is that browsers have rejected option 1. The bad news is that they have settled on option 2, thus hiding textual content until it can be rendered in the desired font. This transition period is known as FOIT – the dreaded *flash of invisible text*.

The alternative option 3 is not great either. Your reader can begin as soon as the content is loaded, but when the web fonts are rendered, the page

layout will reflow to accommodate the new typeface, sometimes resulting in a disconcerting if not confusing experience as words and lines jump around. This is called **FOUT**, the *flash of unstyled text*, and is more often the lesser of the two evils.

A general goal for web designers and developers is to deliver a usable page within one second. Lots of factors aside from bandwidth problems can get in the way of this – we've already mentioned onerous advertising and frameworks – but fonts needn't be one of those. As problematic as FOUT is, it would be more of a feature than a bug to see the content as soon as it has loaded.

Use **font-display** to tailor browsers' behaviour

You can tell browsers to change their behaviour from FOIT to FOUT using a nascent CSS property under development for CSS Fonts Level 4. Use **font-display** in each of your **@font-face** rules to control how browsers should treat web fonts on an individual basis:

```
@font-face {  
    font-family: 'Nicefont';  
    src: url('nicefont_regular.woff2') format('woff2'),  
        url('nicefont_regular.woff') format('woff');  
    font-display: fallback;  
}
```

The **font-display** property takes five values. In most cases you will want to choose **fallback**.

auto The font display strategy is defined by the browser (most often similar to **block**).

block The FOIT option. The browser renders invisible text at first and swaps in the web font as soon as it loads. This value must only be used when rendering text in a particular font is required for the page to be usable. It must only be used for small pieces of text.

swap A FOUT option. The browser renders the text immediately with a fallback if the web font isn't loaded, and swaps the web font in as soon as it loads. This value should only be used when rendering text in a particular font is very important for the page, but rendering in any font will still get a correct message across. It should only be used for small pieces of text.

fallback The preferred FOUT option. The browser waits for at most a fraction of a second before rendering the text, in a fallback font if necessary, swapping in the web font as soon as it loads. However, if too much time passes, the fallback will be used for the rest of the page's lifetime instead. This value should be used for body text, or any other text where the use of the chosen font is useful and desired, but it's important the reader isn't disturbed by the text suddenly shifting as a new font is swapped in.

optional The web font is used if it has already downloaded and is available, but otherwise a fallback is used for the rest of the page's lifetime. The web font might download in the background and be available to future page loads, but if the browser detects very limited bandwidth, it might choose to simply never download and use the font.

At the time of writing, `font-display` is only supported behind experimental flags in browsers, but is perfectly safe to include in your `@font-face` rules ready for when support is more widespread.

Preload the critical font

Preload is a relatively new web standard aimed at improving performance and providing more granular loading control to web developers. You can use preload with web fonts to reduce the amount of FOIT or FOUT visitors will see when they visit your site. Here's all you need to add to the `<head>` of your page to start preloading a web font:

```
<link rel="preload" href="nicefont.woff2" as="font"
      type="font/woff2" crossorigin>
```

The improvement you get from preloading the font occurs because you're telling the browser to start downloading the font straight away. Normally, the browser would have to download all the `HTML` and any `css` files, and parse the `HTML` and `css` on the page, matching selectors to `@font-face` rules which then initiate the font downloading. In the time it takes for those things to happen, a font file flagged for preloading may already have downloaded.

The downside is that you pay for the preload by sacrificing initial render time. The page won't start rendering on the screen until the font file has downloaded, so don't preload too much or your reader will have to wait too long to see anything. Try to only preload a single web font – pick the most important one, which is most likely to be for the body text.

Any bold and italic required will initially be synthesised, but swapped out as soon as the pertinent font downloads.

Fine-tune your web font strategy with font events

The Font Loading API¹ is a way for the browser to say which web fonts are currently loading, which have successfully loaded, and which have failed to load. JavaScript can be used to access events announced by the Font Loading API, and write various CSS classes to the web page's root `<html>` element, which in turn enable you to tweak the text presentation depending on the download status of one or all of the web fonts.

All web font services provide this functionality as standard, and there are many online resources explaining how to use open source libraries that do the same. Filament Group's '[How We Load Web Fonts Progressively](http://wbtyp.net/151)' (<http://wbtyp.net/151>) is a good place to start.

Notably, most of the JavaScript functions which make use of font events can provide a workaround for browsers that do not support `font-display` options. The precise nature of each program varies, but almost all set classes on the root to track the overall web font progress:

wf-loading Class applied while some web fonts are loading.

wf-active Class swapped in once all web fonts have downloaded, or some have downloaded and all others failed.

wf-inactive Class swapped in if all web fonts failed to load.

Use these classes to force a browser that doesn't support `font-display` to show fallback fonts until web fonts are deemed loaded. First, set a `font-family` list without the web font, and then duplicate the selector with the addition of a `wf-active` class, adding the web font to the front of the `font-family` list:

```
h1 { font-family: "Calibri", "PT Sans", "Roboto", sans-serif; }
.wf-active h1 {
  font-family: "Questa Sans", "Calibri", "PT Sans",
  "Roboto", sans-serif;
}
```

¹ [CSS Font Loading Module Level 3](http://wbtyp.net/93) (<http://wbtyp.net/93>).

In this example, the browser will set headings in a system font until the web fonts have loaded. At that point, JavaScript swaps in the `wf-active` class, the second rule overrides the first, and the Questa Sans web font is used.

You will need to apply this to every selector where you have a `font-family` property calling for a web font. This makes it a far higher maintenance proposition than `font-display`, which is only used in `@font-face` rules (but lacks universal support).

Load web fonts asynchronously

One important advantage of JavaScript font events enhancements is that for the programs to function as required, web fonts begin loading before any separate CSS files. Rather than waiting for the CSS file to load, parsing the HTML and CSS, and then downloading the web fonts, they are loaded *asynchronously*. This is different from the preload web font, which kicks in as soon as the preload `<link>` is encountered as the HTML downloads, but blocks rendering of the page.

Choose the best fallbacks

In the prior example, you will have seen three system fonts listed as fallbacks for Questa Sans. When you take a FOUT approach to web font rendering, a system font renders the text before the web font loads, so you need to ensure you are choosing the best possible fallback.

Your first priority is to ensure the fallback text is as legible and readable as the web font. This means choosing a system font which has a similar x-height and renders at about the same size. Next, you need to reduce reflow when the web font is swapped in. For this, horizontal metrics come into play. If your web font is fairly narrow, your chosen system font should also be economical. Finally, you need to get the best stylistic match you can. Look in particular for similarities in the *G*, *W*, *a*, *y* and *g*.

You need to pick a fallback from each of the operating systems you care about. These will probably include various vintages of Android, Windows, macOS and iOS. Start building up your `font-family` list by selecting the generic font style; for example, `sans-serif`. Then look to Android as it has the next easiest decision to make: in terms of the Latin alphabet, you can choose sans or serif versions of either Droid or Roboto. Now turn to iOS. You have a few more to choose from here: start with the core web fonts (*Georgia*, *Verdana*, *Arial*, *Times New Roman*, *Trebuchet*, and so on); then there are some other niceties, including *Avenir*, *Baskerville*, *Futura*, *Gill Sans*, *Hoefler Text*, *Optima*, *Palatino*, *Helvetica* and the *San Francisco UI*

font (which you'll need to invoke as `-apple-system`). Install the free Font Locator app² to see a list of the fonts currently installed on your iOS device.

Next up: Windows. Nowadays, mobile and desktop Windows share the same fonts (mobile used to have a small subset of the desktop faces), so your choice is pretty wide. You should find something suitable for body text among the core web fonts, the ClearType fonts (Calibri, Cambria, Candara, Constantia, Corbel), Segoe UI or Sitka. If none of those comes up trumps, you'll also find Gill Sans, Lucida Sans, Franklin Gothic Medium and Century Gothic, among others.

Finally, look at macOS. The choice you made for iOS might do the job nicely. If not, there are plenty of other really good typefaces to consider. For starters, try Athelas, Charter, Lucida Grande, Marion, PT Sans and PT Serif, and the little known but excellent Seravek.

Set up a web page on the devices and operating systems you are targeting and compare your choices alongside the web font. As a general rule, put your final `font-family` rule in order of best match. In our example, Calibri is a better match for Questa Sans than PT Sans, so it comes earlier in the font stack. Calibri is installed with Microsoft Office, so this means any Mac user with Office on their machine will get a slightly improved experience, with text rendered in Calibri rather than PT Sans.

Ancient geography.	Questa Sans
Ancient geography.	Calibri
Ancient geography.	PT Sans
Ancient geography.	Roboto

Questa Sans, Calibri, PT Sans, Roboto.

Adjust fallback font size to match x-height

We said that your first priority for a fallback font should be matching x-height to ensure readability at the desired text size. If your fallback fonts are, for the most part, consistently different to your web font, then you could use font event classes to make adjustments to the fallback fonts. For instance, if you are using a body font such as Altis, with its particularly

² [Font Locator](http://wbtyp.net/64) (<http://wbtyp.net/64>) iOS app.

large x-height, you could make the fallback fonts slightly bigger so they appear to match. You will need to reduce the line spacing by the same proportion you increased the text size, to keep the line height the same when the fonts swap over.

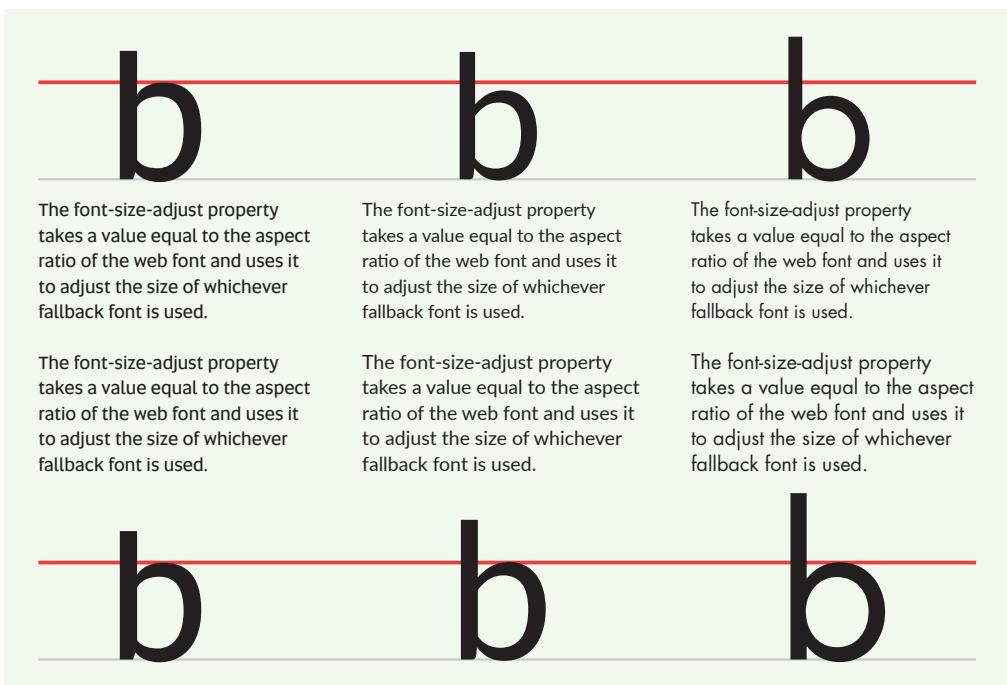
```
body {  
    font-family: "Arial", "Droid Sans", sans-serif;  
    font-size: 1.0625rem; /* 17px */  
    line-height: 1.41;    /* 24px */  
}  
.wf-active body {  
    font-family: "Altis", "Arial", "Droid Sans", sans-serif;  
    font-size: 1rem;   /* 16px */  
    line-height: 1.5; /* 24px */  
}
```

If you choose consistent fallback fonts, you can apply this methodology to more extreme differences in display faces. If you have a heading set in a large narrow web font, you could choose a regular width fallback and make it smaller to get closer to the horizontal metrics of the web font.

In terms of matching x-height alone, CSS has a property designed to do just that. The `font-size-adjust` property takes a value equal to the aspect ratio of the web font (see ‘[Designing paragraphs: text size](#)’ for more detail on aspect ratios) and uses it to adjust the size of whichever fallback font is used. You can use an online tool³ to calculate the web font’s aspect ratio. Because `font-size-adjust` only affects the *rendered* size of the text, not the reported or *calculated* size, the line height remains unaffected and your code can be simplified:

```
body {  
    font-family: "Arial", "Droid Sans", sans-serif;  
}  
.wf-active body {  
    font-family: "Altis", "Arial", "Droid Sans", sans-serif;  
    font-size-adjust: 0.545;  
}
```

³ Online aspect value calculators by [Bruno Fassino](http://wbtyp.net/59) (<http://wbtyp.net/59>) and [Richard Rutter](http://wbtyp.net/58) (<http://wbtyp.net/58>).



Before and after applying font-size-adjust.

Services versus self-hosting

You have two ways to license and serve web fonts. You can either buy fonts to host on your own servers, known as *self-hosting*, or you can rent them from a web font service.

Web font services

Web font services provide a number of advantages over self-hosting. For a start, you can be sure the licensing will be correct and you won't be breaking any end user license agreements (EULAs) by inadvertently copying web fonts to a server you didn't have a licence for. You will often get access to an entire library of fonts, all licensed for you to use.

Web font services host the fonts for you and most charge an annual or monthly fee. The cost includes hosting, bandwidth and delivery of the fonts through a content delivery network (CDN). This serves files from locations all around the world, automatically choosing the closest to reduce response and delivery times. You may not be able to match the latency and speed of a global CDN with your own setup.

Web font services also keep the fonts updated. This might include adding in missing glyphs, improved hinting, additional OpenType features or better compression. For some services the updates go out automatically; with others you are notified and the update requires some action from you (just in case the update unexpectedly affects the layout of your web pages).

You can use most web font services to tune the web fonts to your requirements; for example, specifying which OpenType features you do or don't need, and limiting the character sets to the languages you need to support.

Most services optimise which font files are delivered to which browser. For example, older Internet Explorers (8 and below) don't support kerning, so there's no point wasting bandwidth with a kerning table in the files delivered to IE8 and earlier. Conversely, only Windows browsers use hinting, so there's no need to send files to other platforms with hinting included.

Self-hosting

The main advantage that self-hosting gives you is flexibility. First of all, there is a one-off fee, which is easier to pass on if necessary, and can sometimes work out cheaper in the long run than a recurring fee with a web font service.

You also have flexibility in how you call in the web fonts to your pages. Some web font services require you to use their JavaScript. By self-hosting you can choose whichever method suits you best. The flip side is that you are entirely responsible for how you serve web fonts. If you want any optimisation of file size, you will have to subset it yourself, or through a foundry or vendor's tool if there is one (doing so yourself may be ethically justifiable, but could break your EULA). If you want to serve specific files to specific platforms, again you'll have to work out how to do that yourself, and the same caveat applies when it comes to modifying supplied web fonts.

If your site is served from a shared server in a single geographic location, you should consider putting your fonts on a CDN, such as Amazon S3, which has worldwide coverage. This will particularly benefit those readers who are geographically distant from your server. Australia, for example, has a particular problem with latency when requesting files from the US or Europe.

If you bought a web font for self-hosting, keep track of when (or if) updates are available – these should be available free of charge, but it will usually be up to you to find out if an update has been made.

Variable fonts

In October 2016, version 1.8 of OpenType was released⁴, and with it an extensive new technology: OpenType Font Variations. More commonly known as *variable fonts*, the technology enables a single font file to behave like multiple fonts. This is done by defining variations within the font, which are interpolated along one or more axes. Two of these axes might be width and weight, but the type designer can define many others too.



Continuous variation along width and weight axes.

The preceding image shows a variable font rendered in 36 different styles, all from one file. If you were to pick four styles and serve them as normal fonts, a variable font file capable of providing the same styles would be significantly smaller than the four separate files, with the added speed advantage of requiring just one call to the server.

The illustration varies width and weight. Those two axes alone mean that, according to the OpenType Font Variations specification, theoretically $1,000 \times 1,000$ (one million) variations are possible within the one file with no extra data. A third axis could increase the possibilities to one billion.

⁴ ‘Introducing OpenType Variable Fonts’ (<http://wbtyp.net/19>) by John Hudson (2016).

At the time of writing, the technology is in its infancy, but it potentially opens up tremendous opportunities for new kinds of responsive typography. The file size savings and fine precision mean that many small adjustments could be made to the rendered font, potentially responding dynamically to the reader's device and environment, as well to the text.

Within the design space created by the axes of variation in a font, the type designer can define specific positions as *named instances*. Each named instance could appear to users of design software as if it were a separate font; for example, 'regular', 'light condensed' or 'extra bold extended'.

In the OpenType specification, five common axes of variation have been predefined as four-character tags: weight (`wght`), width (`wdth`), italic (`ital`), slant (`slnr`) and optical size (`opsz`). These font variations can be enabled by the `font-weight`, `font-stretch` and `font-style` properties, with CSS Fonts Level 4⁵ adding new values for the properties to work with font variations:

- `font-weight` takes any integer from 1–999 (not limited to multiples of 100 as in CSS3).
- `font-stretch` takes a percentage number where 100% is predefined as normal, 50% as ultra-condensed and 200% as ultra-expanded.
- `font-style` takes an oblique angle value from `oblique -90deg` to `oblique 90deg`.
- `font-optical-sizing` is a new property taking a value of `auto` or `none`, which turns on optical sizing if it's available as an axis in the variable font.



An An An An An

Continuous variation along an optical sizing axis.

Font designers can also define custom axes with their own four-character tags. This enables designers to vary almost any imaginable aspect of a typeface, such as contrast, x-height, serif shape, grunginess, and even parts of individual glyphs, such as the length of the tail on a Q. Using a syntax similar to `font-feature-settings`, custom axes (as well as the predefined ones) are available through the low-level `font-variation-settings` property.

⁵ CSS Fonts Module Level 4 (<http://wbtyp.net/95>).

For example, the following would render text with a variation that is very wide, light in weight and optically sized for 48 pt:

```
h2 { font-variation-settings: "wdth" 600, "wght" 200, "opsz" 48; }
```

Visit Laurence Penney's [Axis-Praxis](http://wbtyp.net/60) (<http://wbtyp.net/60>) to play with variations and design instances of some variable fonts.

As with regular OpenType fonts, variable fonts can be used as web fonts as is, or preferably wrapped up as a WOFF. To use a variable font as a web font, your `@font-face` rule should set the `format` to `woff-variations` or `ttf-variations`. If you wish to provide regular font fallbacks for browsers that don't support variable fonts, you can use multiple `@font-face` rules where necessary, repeating the variable font each time.

```
@font-face {
    font-family: 'Nicefont';
    src: url('nicefont_var.woff2') format('woff-variations');
    src: url('nicefont_regular.woff2') format('woff2');
    font-weight: normal;
    font-style: normal;
}

@font-face {
    font-family: 'Nicefont';
    src: url('nicefont_var.woff2') format('woff-variations');
    src: url('nicefont_black.woff2') format('woff2');
    font-weight: 800;
    font-style: normal;
}
```

At the time of writing, there is support for `font-variation-settings` in development versions of all the major browsers, but none support `font-weight` or other such properties with variable fonts. Additionally, the web font `format` needs to be `woff` or `ttf`.

Variable fonts were jointly developed by Adobe, Apple, Google and Microsoft. This means support in new versions of browsers should arrive across the board as soon as the precise implementations and CSS specifications are agreed. Current estimates⁶ have variable fonts being a viable option on the web by early 2018.

⁶ According to Typekit's Tim Brown and Bram Stein, interviewed in '[Variable Fonts](#)' (<http://wbtyp.net/31>) on the *Responsive Web Design* podcast (2017).

Building a library

Compiling your own library of typefaces is a lifelong endeavour. Assemble your library of faces slowly and well. Learn how they work, their virtues and limitations. You will naturally develop favourites and gain an instinctive feel for when one typeface will do a job better than another.

Develop your own taste and preferences

When we talk about building up a library, it's not so much about buying an expensive collection of fonts over time, but getting to know, use and deeply understand your own particular selection.

When it comes to selecting type for a job, some designers spend days luxuriating in research: downloading type specimens, noodling with online typesetting apps, comparing character sets, weights and alternates. This is time well spent. Doing so will broaden and deepen your knowledge, opening up opportunities to expand your typographic palette.

Other designers will reach straight for their familiar go-to repertoire, experienced and confident in the applicability of their choices. Either one of these approaches is appropriate for a given job.

Italian designer Massimo Vignelli, responsible for the New York City Subway map, among other iconic designs, railed against desktop publishing and the freedom to butcher any available typeface as '*a disaster of mega proportions [and] a cultural pollution of incomparable dimension*'. He maintained that twelve typefaces should be all someone needs in their canon¹.

To draw attention to the issue, he made an exhibition showing work that Vignelli Associates had done over many years consisting of only four typefaces: Garamond, Bodoni, Century Expanded and Helvetica. The aim of the exhibition was to show that a large variety of work could be achieved with only a few typefaces. In other words, it's not the type but what you do with it that counts, and if you know the type well enough you can work it to your needs.

¹ 'The Vignelli Canon' PDF (<http://wbtyp.net/34>) by Massimo Vignelli (2009).

Vignelli's point is well made, but flawed. Your typographic palette should be small enough that you know intimately how each face works under all manner of conditions, to the point you can instinctively reach for a suitable option for any job. But to stop when you reach twelve typefaces would be very sad indeed. It's true that font design follows Sturgeon's law² and we might be better off if many typefaces had never seen the light of day. But to say that twelve will do, and that those are – and always will be – the pinnacle of type design is nonsense. Of course, you don't need to use a different typeface for every new job, particularly when it comes to body text, but to limit yourself to an arbitrarily small number is a wasted opportunity. It is an insult to your reader, and to those designers, standing on the shoulders of giants, pushing forward the technical capability, aesthetics, readability and durability of modern typefaces. To say 'I have enough' is to deprive yourself of the joy of discovering, learning and using a new typeface.

Steal inspiration, don't copy

Pablo Picasso once said 'When there's anything to steal, I steal.' Explaining this, he said of his contemporary Henri Matisse:

You have got to be able to picture side by side everything Matisse and I were doing at that time. No one has ever looked at Matisse's painting more carefully than I; and no one has looked at mine more carefully than he.

The difference between copying and stealing in this case is not one of intellectual property rights. It's not about taking someone else's idea and pretending it's yours. If you copy someone else's type choice or technique without thinking, you'll be using something that worked for them in their circumstances. By *stealing* an approach, you are making it your own. You'll have to ascertain why it worked for the originator, and how it can work for you, adapting and moulding the approach as necessary, completely changing the context if necessary.

Get your inspiration from where fonts are actually being used. Consider it an investment to spend part of your week looking closely at other people's work. Screenshot nicely designed websites, tear out or scan pages of magazines, visit design review sites and read typography blogs (see the bibliography for suggestions). Identify graphic and web designers you admire

² The adage commonly cited as '90% of everything is crap' is derived from quotations by Theodore Sturgeon, an American science fiction author and critic.

and try out their preferred typefaces. Look at the font choices they made and the way they use them. Finding designers you like and adopting their type palettes is an effective way to shape your own library. Subscribe to font services and make full use of the favouriting functionality provided by type services.

It pays to be a design sponge in the wider world. Magazines are currently undergoing something of a revolution. Look to them for inspiration, but not just the hipster designer ones. Go outside your realm - there are many things to steal from the mainstream as well as the underground. Go back in time to classic posters or advertising from the 1970s (or even the 1980s, if you must). Keep searching for typography that works for you. Identify fonts³ and spend time investigating them. If you feel an affinity with a particular typeface, research the history and work of the designer. You may find their other typefaces work in a way you like too.

Spend time on type foundry websites. They are often great sources of beautiful and expressive typography, as their intention is to show off the typefaces in ways for which they were designed.

Get what you pay for, and pay for what you get

Building up your typographic knowledge and repertoire means a lot of playing, experimenting and working with type. In the past (before digital fonts or the internet) this would have been prohibitively expensive. With the advent of the internet, access to a vast array of fonts became possible, but still either costly or frequently illegal. More recently, retail models have changed for the better. Almost all professional fonts are available through online services and libraries designed to give you cheap or free access to typefaces for trial and experimentation purposes. Some foundries, such as Dalton Maag, offer downloads of fonts on a free-to-try, pay-to-use basis.

Designing and developing a professional typeface takes many, many months of meticulous work⁴. While many of the font distribution and vending services are run by large, publicly listed companies, the majority of type designers work as individuals or small independents. It's vital for their livelihoods and the cultural vibrancy and longevity of graphic design that you honour their work correctly. Ensure you, your agency, company or client pays for the right licence when it is required – you have a moral, as well as legal, obligation.

3 Plenty of automated tools exist to identify fonts. For example: [Identifont](http://wbtyp.net/65) (<http://wbtyp.net/65>), [WhatTheFont!](http://wbtyp.net/87) (<http://wbtyp.net/87>) and [Matcherator](http://wbtyp.net/67) (<http://wbtyp.net/67>).

4 Nina Stössinger on [designing FF Ernestine](http://wbtyp.net/32) (<http://wbtyp.net/32>).

Be careful with free. While developing a professional typeface is time-consuming, completing the basic design can be a matter of weeks. Often that's the point at which a free font is released, bereft of useful character sets, different weights, proper italic, accurate spacing, kerning, ligatures or hinting. Even some free fonts available from large, otherwise professional distributors, will be rather unfinished. *Caveat emptor*: you'll need to pay for free with additional testing.

Typomania is incurable but not lethal

Typophilia is love of type. If you are hoping – or have already started – to build your own type library, then you are already affected by typophilia. This will inevitably lead to *typomania*, a word affectionately coined by Erik Spiekermann. This is a state of being where you see type everywhere, when you mentally try to label every typeface you see and are inadvertently lost in its use and misuse. Typomania is a harmless but incurable disease of the mind, shared by many sufferers who enjoy nothing more than to geek about kerning, terminals, counters and cuts. You have been warned. *Welcome to the club!*

§ Final Thoughts

- Communicating your design
- If you read nothing else, read this

¶ *It is only with practice that you can become truly accomplished as a typographer, but a good set of guidelines can help you along the way.*

Communicating your design

Web typography is complex, intricate, detailed and time-consuming. If the fine details you have agonised over in your design are not communicated effectively from designer to developer, then all that effort will be for nothing.

If you have followed the advice in this book, you will have created a responsive prototype. This both illustrates and demonstrates the typography in a browser, providing you with an invaluable communication tool. Under the hood of your prototype will be css, which – if coded and commented well – can potentially be the ultimate typography specification. If your prototype is a comprehensive collection of all your typographic decisions, then its css will contain all the necessary dimensions, values and proportions of your typesetting, including how it should respond to different viewports. In theory, a developer could refer to your css for everything they need to know about the typographic system you have designed.

The aforementioned developer may, of course, be you (as designer-developer), but not all designers can or need to code. Designing and developing can be a partnership, a symbiotic relationship that frees the designer from shortcomings in their coding ability, and the developer from any limits to their design sensibilities. If lucky enough to be in a partnership situation, you will probably need the modern equivalent of what Erik Spiekermann termed a *typogram*¹.

A typogram comprehensively lists all the details of any part of your typographic scheme. The figure shows an entire page of notes describing the precise setting of body text in a book, including typeface, size, line spacing, letterspacing, margins, justification and hyphenation settings.

¹ In *Rhyme & Reason: A Typographic Novel* by Erik Spiekermann (1987).

Job	Rhyme & Reason		
<hr/>			
Typeface	Walbaum Standard with Small Caps		
Walbaum Standard Italic			
Type size	Line space	Word space	Letterspace
9 p (H = 2,4 mm)	4.25 mm	13 units	0 (except where marked)
8 p (H = 2,12 mm)	3.50 mm	12 units	0 (except where marked)
Setting mode & measure	justified to 81 mm		
Rules	see mark-up	Trimmed page size	
		185 × 123 mm	
Remarks	<ul style="list-style-type: none"> → Emphasis in text : set in 9 p italic, with 12 unit word space - otherwise as text : → set with Aesthetics programme : optical adjustment of left & right margin → Never more than two consecutive hyphens ! 		
<hr/>			
Output as	Film - wrong - reading positive		
Contact	Erik	Delivery	Immediately (27.07.87)

A typogram as featured in *Rhyme & Reason: A Typographic Novel* by Erik Spiekermann.

If you are not writing CSS yourself to fully demonstrate and document your design, you will need to list every single detail of the type and its setting. Software can help you with this. For example, there's an open source plug-in for Sketch called Sketch Measure² which enables your design to be interactively inspected element by element.

² Sketch Measure (<http://wbtyp.net/75>) plug-in for Sketch.

Your best bet, though, is a living style guide³, a continually updated website containing component patterns, type specifications, page layouts, samples and, often, front-end code. Website style guide resources, including dozens of live examples, can be found at [Website Style Guide Resources](http://wbtyp.net/88) (<http://wbtyp.net/88>).

Example Default typographic scale		
Heading, typeface, line height and margins	font size to use above 768px viewport	font size to use below 768px viewport
H Mega - Demi Bold 64px	72px 60px	36px
H1 - Demi Bold 32px	64px 46px	30px
H2 - Demi Bold 8px	40px 26px	22px
H3 - Demi Bold Box top and bottom	32px 22px	20px
H4 - Demi Bold Box top and bottom	32px 22px	20px
H5 - Demi Bold Box top and bottom	32px 22px	20px

A small extract from the *Co-op Design Manual* showing type sizes, line heights and margins.

It is your job to communicate your design in the most detailed and usable way you can for the situation you find yourself in. This aspect is as important as any other, because without it your work up until now will be missed, misinterpreted or mislaid. Your job is to honour the text and put your reader at ease. Communicating your solution to that is the final step of the journey.

³ For example: ‘[Typography](#)’ (<http://wbtyp.net/13>), *Co-op Design Manual*.

If you read nothing else, read this

Wherever there is type, there is typography; and where there is typography, it's the little things that count.

The way type is set – whether good or bad, with care or little thought – always says something about the text. The legibility of the words, the ease with which they can be read, and the scene set by the choice of typeface, all combine to create your reader's experience.

Typography is just that – the design of the reading experience. It is a craft which, perhaps more than any other discipline, encompasses the notion of the whole being greater than the sum of the parts. The details we strive to get right are frequently mundane and seemingly trivial, and yet, when taken together, can elevate an experience from tiresome to delightful.

The very best typographers share a number of traits. Good taste and a sense of the aesthetic are chief among them, although these must be tempered by humility in their work. An eye for detail and aptitude for rigour will ensure the work is done, and done right. But a typographer must be willing to learn. No one is born with an innate mastery of typography. There are rules and conventions whose causes and effects we only get to understand and appreciate over time. Books like this one attempt to collate some of that understanding, but it is only with practice that a typographer can become truly accomplished. On that note, here is one final piece of advice.

Don't trust computers

Or rather, don't trust browser defaults – use your own judgement. If you know what good work looks like (and hopefully you will after reading this book), you'll do good work. In an interview in 1998, design writer Steven Heller expressed this splendidly:

If you have never done design or typography any other way than by computer, your entire thinking can be structured by what the computer will do so easily. It's bad in typography because many people believe that if the computer spaced it, it must be right. A typographer can't live with that.

If you find yourself using a browser's default value for text size, or line spacing, or width, or margin, or colour, or any other value affecting your design, then question it. It may be fine. It may be just off, or it may be entirely inappropriate.

Guidelines

This book is written as a series of guidelines. As you grow more experienced, you may find yourself ignoring and contravening this guidance. That's fine. These are not rules and you should not take them as such. But if you do work around these guidelines, do so knowingly and with good reason.

We are all Typographers

- Good typography induces a good mood 4
- Learn to relinquish control 7
- Know what you are designing and for whom 8
- Honour the text, don't dominate it 8

Setting Type to be Read

- We read in hops, skips and jumps... and pauses 12
- Use rem for global sizing; use em for local sizing 17
- Choose a comfortable measure 22
- Lean on six centuries of typesetting experience 22
- Set text for mobile first 23
- Use a liquid setting 24
- Use the default font size for paragraphs 26
- Take small print into account 27
- Adjust the font size if the typeface requires it 27
- Use rem to size your text 31
- Line height should suit text size and measure 34
- Use unitless values to set line-height 37
- Increase line spacing from the browser default 39
- Adapt the line spacing to suit the typeface 39
- Justify well or not at all; if in doubt, align left 43
- Don't justify the final line of a paragraph 47

- Don't justify without hyphenating 48
- Hyphenate judiciously and with care 51
- Limit the number of consecutive hyphenated lines 52
- Limit the word length and the number of characters before and after a hyphen 52
- Reduce hyphenation by setting a hyphenation zone 54
- Avoid leaving the stub of a hyphenated word as the last line of a paragraph 56
- Don't centre long passages of text 56
- Adapt your design to the reading context 57
- Use ems rather than pixels to determine the screen width 58
- Adjust type size according to reading distance 59
- Calculate text sizes for different devices 60
- Keep your big text under control 62
- Beware of the tablet problem 62
- Test your design in a responsive prototype 63
- Stepping back and squinting are no substitute for reading 63

Typographic Detail

- There is more than one space 67
- If a word has an accent, use it 68
- Pay particular attention to accented names 69
- Use the appropriate punctuation marks 70
- Don't use a hyphen instead of a dash 70
- In running text use a proper minus 72
- Use a division symbol rather than a slash 72
- Use proper and appropriate quotation marks and apostrophes 72
- Don't use italic brackets 73
- Don't overuse or over-elaborate ampersands 73
- Use UTF-8 in preference to entities 75
- Automate character substitution whenever possible 76
- Use standard ligatures for improved legibility 78
- Hide legacy OpenType support from modern browsers 81
- Use alternative ligatures judiciously 83
- Use real small caps 84
- Establish the structure up front 87
- Don't rely on size alone for differentiation 87
- Always size type using a scale 87
- Choose your smallest size first 91
- Use alternative modular scales to suit the content 91

- Adjust the text size and scale for different screens 93
- Take screen height into account 95
- Establish the semantics up front 96
- Tighten up leading in headings 98
- Avoid clash (unless it's a deliberate choice) 98
- Mark the opening of each passage of text 98
- Implement drop caps accurately or not at all 99
- A standfirst should stand first 102
- Demarcate paragraphs appropriately 103
- Indent paragraphs at least 1em 104
- Separate paragraphs by no more than 1em 104
- Don't use stylised paragraph breaks without good reason 104
- Not all block quotes need to look like pull-quotes 106
- Hang punctuation 106
- Treat lists as text to be read 108
- Avoid using underlines for emphasis 109
- Avoid faux bolds and italics 110
- Make links clear but unobtrusive 114
- Use old-style numerals in running text 119
- Use lining numerals in headings 121
- Use proper subscripts and superscripts 122
- Reference notes with superscripts 125
- Show footnotes in context 126
- Set tables as text to be read 127
- Don't stretch tables 128
- Keep table furniture and fills to a minimum 128
- Left-align text, right-align numbers, and align headings with data 130
- Align to the decimal point 131
- Use tabular lining numerals in tables of numbers 132
- Put white space to work to group and separate 133
- Do not over-stylise tables 136
- Adapt tables to small screens 139
- Consider setting oblique headings to save space 139
- Let the browser handle tables with horizontal scrolling 140
- Linearise simple tables into lists 142
- Make tables responsive according to their purpose 143
- Kern your text to improve readability 145
- Letterspace your text rarely and carefully 147
- Don't letterspace lowercase without good cause 148
- Letterspace all strings of capitals and all long strings of digits 148

- Eliminate erroneous spaces after last letters 149
- Gently tighten big, bold and wide fonts 150
- Turn off ligatures if increasing letterspacing 151
- Set text at display sizes, even on small screens 152
- Resize display text as you would an image 154
- Attend to wide, shallow screens 159
- Pay attention to the shape of the text 161
- Visually centre text or hang punctuation 162
- Prevent widows in headings 164
- Adjust line spacing for evenness of colour 164
- Make the most of a font's hidden gems 165
- Seek out OpenType features 169
- Apply vertical space in measured intervals 171
- Use a basic reset to ensure consistency 172
- Allow embedded media to break the rhythm 174
- Don't slavishly follow a baseline grid 175
- Design layouts from the content out 177
- Adopt a mobile-first philosophy 178
- Stage the main text block for scanning 179
- Bring the design into the margins 180
- Use active white space 182
- Show relationships and draw attention 183
- Use compound grids for maximum flexibility 188
- Choose odd over even grid systems 189
- Restrict column length to the viewport height 190

Choosing and Using Fonts

- Accept the limits of screen resolution 195
- Understand how fonts render on screens 196
- Accept rendering differences between platforms 199
- Don't disable subpixel rendering except when text is reversed out 201
- Check your type across platforms 202
- Don't jump straight into choosing fonts 203
- Understand the purpose and requirements 204
- Use a typeface with the necessary characters 204
- Make sure your typeface has the requisite styles 205
- Choose fonts with the features and special effects you require 206
- Consider your fonts' performance 206
- Work within your means, but be wary of free 207
- Be pragmatic in the face of branding requirements 208

- Learn how to describe typefaces 213
- Type genres are more useful than classifications 218
- Know your history 219
- Understand type categories 222
- Remove the friction between reader and text 237
- Choose robust faces for better readability on screens 237
- Choose active texture and even colour for a smoother read 238
- Choose faces in keeping with the text 240
- There's no such thing as a neutral typeface 242
- Choose a face whose history and culture resonates with the text 242
- Do some detective work 243
- Make a specimen to reduce your shortlist 243
- Judge typefaces in context 245
- Trust your personal preference 245
- Don't be reverential, dogmatic or ordinary 246
- Influence the way people feel through type 246
- Favour personalities over flexibility 251
- Use display styles for display text 252
- Choose display faces that reinforce the structure of the text face 253
- Keep inspired, and play 255
- Clearly distinguish functional text from body text 256
- Choose open, distinct letterforms for functional text 257
- Combine typefaces for a reason 262
- Start your combination with an anchor typeface 263
- Find typefaces that complement the anchor 263
- Combine typefaces with similar structure and form 263
- Find complements through contrast 265
- Stick with a single type family 265
- Consider pairing typefaces from the same designer 267
- Pair typefaces from the same foundry 268
- Don't specify local fonts without due diligence 271
- Reduce your payload 272
- Limit the number of fonts you use 272
- Always provide a WOFF2 option 272
- Subset as much as you can, but no more 273
- Optimise page render timing 273
- Avoid the flash of invisible text 273
- Use font-display to tailor browsers' behaviour 274
- Preload the critical font 275
- Fine-tune your web font strategy with font events 276

- Load web fonts asynchronously 277
- Choose the best fallbacks 277
- Adjust fallback font size to match x-height 278
- Develop your own taste and preferences 285
- Steal inspiration, don't copy 286
- Get what you pay for, and pay for what you get 287
- Typomania is incurable but not lethal 288

§ Appendices

- Glossary
- Bibliography and further reading
- CSS index
- Thank you

Glossary

Active white space. Use of white space to frame text blocks, highlight elements and lead your reader from one aspect to another.

Aliased text. Aligning pixels precisely to a pixel grid so that pixels are turned on or off. See *rasterisation* and compare with *anti-aliased text*.

Anti-aliased text. Font smoothing to improve accuracy of rasterisation by partly dimming pixels whenever part of a pixel falls inside a font vector outline. See *rasterisation* and compare with *aliased text*.

Aperture. The opening of a partially enclosed part of a letter such as *n*, *C*, *S*, the lower part of *e*, and the upper part of a double-storey *a*.

Arcminute. A measure of an angle, equal to one sixtieth of a degree.

Arm. A stroke which joins a stem somewhere in the middle. A letter *k* has two such arms.

Ascender. A stem on some lowercase letters, such as *h* and *b*, that extends above the x-height.

Aspect value. The relative height of lowercase letters to their uppercase counterparts; more precisely, the x-height of a font divided by the font size.

ATypI. Association Typographique International. An international non-profit organisation representing the worldwide type and typography community and industry. Holds a conference in a different global city each year.

Axis. An imaginary line drawn from top to bottom of a glyph giving an indication of the angle at which a nibbed pen would be held to draw the letterform.

Baseline. An imaginary line on which the letters in a font appear to rest.

Bicameral. A script whose letters have both majuscule and minuscule forms; that is, uppercase and lowercase letters.

Body text. The main text of a page.

Bowl. The curved line which fully encloses part of a letter such as *d*, *b*, *o*, *D* and *B*. Compare with *counter*, which is the interior space created when a bowl is drawn.

Bracket. A curved or wedge-like connection between a serif and the main stroke or stem.

Breakpoint. A screen width set in a media query used to trigger a different style sheet at the point the design would otherwise break.

Cap height. The distance from the baseline to the top of the uppercase letters (not including accents or diacritics).

Chunked text. A style of composition with the skimming reader in mind, featuring short paragraphs and frequent subheadings.

Clash. What happens when descenders touch or overlap ascenders.

Colour. Typographic colour refers to the density of the texture of a body of text – the overall blackness of the letterforms en masse against the background.

Columnar grid. Consists of vertical columns spaced predictably across a page.

Common ligature. Ligature turned on by default in OpenType and css. Examples include *ff fi fl ffi* and *ffl*.

Composer. Someone who lays out and typesets a page, particularly on a letterpress.

Compound grid. Created by combining hierarchical grids and columnar grids.

Contextual swash. Swashes which are appropriate for the start, middle or end of words, thus avoiding awkward clashes or gaps. See *swash*.

Contrast. The degree of difference between the thick and thin strokes in a glyph. High contrast means there is a big difference between thick and thin strokes.

Counter. The interior shape, enclosed or partially enclosed, in letters such as *d*, *o*, *c* and *s*. Compare with *bowl*, which is the line drawn to form a fully enclosed counter.

Crossbar. The usually horizontal stroke across the middle of letters such as *A*, *H* and *e*.

Data-ink. Defined by Edward Tufte as ‘the non-erasable core of the graphic’, whereas non-data-ink is the ink used in the graphic, not to directly represent data but for scales, labels, fills and edges.

Data-ink ratio. Defined by Edward Tufte as the proportion of ink that is used to present actual data compared to the total amount of ink used in the entire graphic.

Descender. Any part of a lowercase letter that extends below the baseline, found in lowercase letters including *g, j, p, q* and *y*, and occasionally in uppercase letters such as *J* and *Q*.

Diacritic. A mark, such as an accent, added to a letter, often indicating a difference in pronunciation.

Discretionary ligature. Optional ligature used for stylistic purposes.

Display text. Usually set large, used to grab attention and set a mood.

DPI. Dots per inch, a way to quantify a printer’s resolution. Compare with *PPI*.

Drop cap. Dropped capital – a larger than usual letter at the start of a paragraph spanning multiple rows downwards. See also *raised cap* and *sunken cap*.

Dynamic form. Letterforms with a contrast and structure derived from writing with a broad-nibbed pen held at a consistent angle.

Ear. A decorative flourish usually on the upper right side of a bowl, most often found on a lowercase *g*.

Ellipsis. ... symbol, used to indicate omitted text or to imply continuation.

Em. A distance equal to the font size.

Em dash. — symbol, 1em long, typically used to separate phrases, indicate dialogue and attribution.

Em space. A space 1em wide.

En. A distance equal to half an em; that is, half the font size.

En dash. – symbol, 1en long, typically used to separate phrases and ranges.

En space. A space 1en wide.

Endnote. Notes accompanying the main text positioned either at the end of a chapter or the entire work.

Eye. The counter, or enclosed space, of a lowercase *e*.

Faux bold. Bold text synthesised from the regular weight.

Faux italic. Italic text synthesised by slanting the roman.

Feature tag. OpenType table providing instructions about how to use certain glyphs in a font.

Fist. See *manicule*.

Fixations. Slight pauses in reading eye movement made to see the letters within words, and occasionally the spaces between them.

Fleuron. Flower-like symbol used for decoration, such as ♀.

Flow. A mental state in which your reader is absorbed by the text, unaware of their surroundings, with time flying by unnoticed.

FOIT. Flash of invisible text. The period during which a browser hides textual content until it can be rendered in the desired web font. Compare with *FOUT*.

Font. An individual style such as Gill Sans Bold Condensed. Compare with *typeface*.

Footnote. Notes accompanying the main text positioned at the bottom of the referring printed page.

FOUT. Flash of unstyled text. The period during which a browser renders textual content in a fallback font until the desired web font is rendered. Compare with *FOIT*.

Foveal vision. When lenses in eyes can only focus with total acuity on a tiny section of the field of vision.

Geometric form. Letterform modelled as if produced by a ballpoint pen, often with consistent geometric shapes.

Gestalt principles. Design techniques derived from how visual input is perceived by human beings. In particular, how once we see the whole, our perception of the individual parts changes.

Glyph. The specific shape, design or representation of a character.

Golden ratio. φ (≈ 1.618) So-called divine number supposedly appearing repeatedly in nature and all manner of classical art and architecture.

Greedy algorithm. A text justification method which only adds space between words.

Hair space. A very narrow space, usually 1/24 em wide, used to prevent adjacent characters touching, rather than to provide any significant separation.

Hard space. Synonymous with *non-breaking space*.

Hardware pixel. The tiniest full dot a computer screen is capable of displaying.

Hierarchical grid. Organises disparate content by grouping it in horizontal modules stacked vertically one after the other.

Hinting. Instructions in a font file which bend the contours of letterforms so that the pixelated letterform represents a more legible character.

Historical ligature. Optional ligature used for historical or stylistic purposes.

Hung quote. Punctuation situated in a margin or indent so the text lines up.

Hyphenation. Splitting of a word across two lines, indicated by a hyphen -.

Hyphenation dictionary. Language-specific rules for how words can be hyphenated.

Hyphenation zone. The maximum amount of unfilled space (before justification) that may be left at the end of a line before hyphenation is triggered.

Justification. Spacing of text adjusted so that both sides of a text block have an approximately straight edge.

Kerning. An adjustment of the spacing between two characters to reduce unsightliness and the potential for confusion.

Knuth–Plass. A sophisticated line breaking, justification and hyphenation algorithm developed for the TeX typesetting system by Donald Knuth and Michael Plass in 1981.

Lachrymal. Teardrop-shaped.

Ladder. Pejorative term for consecutively hyphenated lines, particularly when there are three or more.

Leading. The spacing between lines of text (pronounced ‘ledding’).

Legibility. The clarity of individual characters and how easily they are deciphered. Compare with *readability*.

Letterspacing. See *tracking*.

Ligature. A single glyph formed by combining two or more characters.

Linear text. Intended for continuous reading, such as books, news articles and product descriptions

Lining numeral. Full height ‘uppercase’ numbers 0123456789 used in headings and tabular data.

Link. See *neck*.

Lobe. An enclosed or partially enclosed counter below the baseline, most often found in a double-storey *g*, but also in cursive (handwritten) styles of letters including *p*, *b* and *l*.

Logical resolution. The screen resolution as reported by a device. For example, an iPhone 4 tells browsers its screen resolution to be 320×480, even though there are 640×960 hardware pixels built into the display.

Loop. See *lobe*.

Macrotypography. The field of typography concerned with layout and the bigger picture on the page.

Manicule. ↗ symbol used to bring attention to something.

Measure. The length of a line of text.

Media query. Rules used by browsers to determine whether a style sheet should be used or not, based on the type of rendering (primarily either screen or print) and the physical characteristics of a device.

Mobile first. A philosophy of design where the starting point of any design consideration is for small screens and less capable devices.

Modular grid. A matrix of grid modules (cells) across and down the page, which can be combined into larger containers.

Modular scale. A prearranged set of harmonious proportions which have a meaningful foundation.

Modulation. Modulation is the rate and manner the contrast changes within a glyph.

Monoline. Letterforms drawn with no discernible contrast, meaning all strokes are the same width.

Narrow no-break space. A narrow non-breaking space, for example used to keep initials thinly spaced but always on the same line.

Neck. The join between two parts of a letter, particularly between the bowl and loop of a double-storey *g*.

Negative space. White space formed within the shape and extremities of a letterform.

Non-breaking space. A space which keeps on the same line the two words it separates.

Obelus. ÷ division symbol.

Oblique. Slanted, optically corrected version of the corresponding roman.

Old-style numeral. ‘Lowercase’ numbers 0123456789 used in running text.

OpenType. A cross-platform font file format developed jointly by Adobe and Microsoft in 1996. Can contain many thousands of characters, languages and glyph variations.

Orphan. A single line left at the bottom of a page or column. Compare with *widow*.

Orthotypography. The aspect of typography that defines the meaning and accepted usage of typographic signs and symbols, in particular punctuation.

Overshoot. Many letters, such as *v*, overshoot the baseline slightly. This gives them optical balance and stops the letters looking like they are about to topple over.

Passive white space. The inherent spacing between words and lines.

Perceived text size. Measure as an angle to take into account both rendered type size and distance from the eye.

Pilcrow. ¶ symbol historically used to indicate a new paragraph.

Pixel. See *hardware pixel* and *reference pixel*.

PPI. Pixels per inch, a way to quantify a screen's resolution. Compare with *DPI*.

Primes. ' and " symbols used to indicate feet and inches, minutes and seconds, and in scientific notation.

Progressive enhancement. A strategy for web design that emphasises core web page content and functionality first, adding more nuanced and technically rigorous layers of presentation and features on top of the content as the user's browser/internet connection allow.

Pull-quote. A typographic device used to pull a quote out of the text by way of an eye-catching snippet.

Ragged left. Right-aligned text.

Ragged right. Left-aligned text.

Raised cap. Large initial letters which align to the baseline of the first line. See also *drop cap* and *sunken cap*.

Rasterisation. The process by which a font file's vectors are turned into pixels rendered on a screen or dots from a printer.

Rational form. Letterforms derived from writing with a pointed pen.

Readability. The level of comprehension and visual comfort when reading typeset material. Compare with *legibility*.

Reference pixel. A measure of distance on a screen equal to 1/96 of an inch.

Reference text. Used for consulting and providing discrete pieces of information, such as glossaries, listings and data tables.

Responsive design. Making designs flexible and adaptable to screens and devices of all shapes and sizes.

Rivers. Unsightly gaps between words cascading through paragraphs where spaces between words line up.

Roman. The normal upright form of a typeface.

Root. In web pages, the <html> element.

Rule. Line or border drawn between rows and columns.

Run-in. Changing the style of the first line of an opening paragraph (to bold or small caps, for example).

Saccades. The jumps made by our eyes as they read along a line of text in quick movements.

Serif. A small mark or foot at the end of a stroke. For example, when the number 1 or the letter *i* are drawn with a bar across the bottom, the two halves of the bar are serifs. If the serif is joined to the stem by a slight flaring out, rather than an abrupt deviation, it is said to be *bracketed*.

Setting solid. Text set with no line spacing (line-height of 1).

Shoulder. The join of a stroke on to the main stem of a glyph, such as where the arm of an *r* joins the vertical stem.

Sidenote. Notes accompanying the main text positioned in the margin.

Sloped roman. See *oblique*.

Small caps. Specially designed small capital letters, about the same height as the x-height of the lowercase letters.

Soft hyphen. Enables you to tell the browser where a word could be broken across two lines. It will not be read out by screen reading software and will only be displayed when a word is actually being broken.

Solid. See *setting solid*.

Sort. An individual metal or wood character used in moveable type printing presses

Standard ligature. Synonymous with *common ligature*.

Standfirst. A short opening paragraph often used in journalism to set the scene or summarise the subsequent article.

Stem. The main, usually vertical, stroke in a glyph.

Stress. See *axis*.

Stroke. The lines used to form a glyph, particularly when thought of as how the letterform would be written or drawn by hand.

Subpixel anti-aliasing. A form of *anti-aliased text* which takes advantage of the red, green and blue stripes which make up pixels in liquid-crystal (LCD) and organic light-emitting diode (OLED) displays.

Subset. The removal of unrequired glyphs from a font to save on file size.

Sunken cap. Large initial letters which are taller than the first line but are located on a lower baseline. See also *raised cap* and *drop cap*.

Swash. Flamboyant additions to a character, such as an exaggerated serif, tail or entry stroke

Tabular numerals. Monospaced (equal width) numbers, typically used in tables and columns of data.

Tail. A stroke on letters such as *Q*, *K* and *R* which descends below the baseline. When an *a* has a stroke rising from the stem towards the next letter, as you might in handwriting, this is also known as a tail.

Terminal. The end of any stroke that doesn't include a serif. Some terminals expand into shapes; for example, ball terminals and lachrymal (teardrop-shaped) terminals.

Text numeral. Synonymous with *old-style numeral*.

Thin space. A small space character, usually 1/6 em wide.

Titling numeral. Synonymous with *lining numeral*.

Tittle. Dot on a lowercase *i* and *j*.

Tracking. The tightening or loosening of spacing between all the glyphs in a block of text.

Typeface. A font family – a collection of fonts based on the same design (as in Gill Sans). Compare with *font*.

Typogram. Term invented by Erik Spiekermann. A comprehensive list of all the details of any part of your typographic scheme.

Typomania. Coined by Erik Spiekermann, a state of being where you see type everywhere, when you mentally try to label every typeface you see, and are inadvertently lost in its use and misuse.

Typophilia. A love of type.

Variable fonts. Technology introduced in OpenType 1.8 which enables a single font file to behave like multiple fonts. This is done by defining variations within the font, which are interpolated along one or more axes.

Viewport. The browser window (sometimes equivalent to the screen).

Viewport units. CSS unit of length where a value of 1 is equal to 1% of either the viewport width or height as reported in reference pixels.

Vox-ATypI. Typeface classification system developed by Maximilien Vox, modified and adopted by ATypI in 1960.

Web font. A font file downloaded for the rendering of text in web pages.

White space. The space between elements in a composition.

Widow. A single word which drops onto the final line of a heading (or any other block of text), or a single line carried over to a new page or column. Compare with *orphan*.

X-height. The height of the lowercase letters. While x-height does not specifically refer to the height of a letter x, the two are usually equal.

Bibliography and further reading

Books

References in bold were particularly well thumbed in the creation of this book – consider them essential reading.

- *Get Ready for CSS Grid Layout* by Rachel Andrew (2016).
- *Explorations in Typography: Mastering the Art of Fine Typesetting* by Carolina de Bartolo with Erik Spiekermann (2011).
- *The Typographic Grid* by Hans Rudolf Bosshard (1999).
- **The Elements of Typographic Style** by Robert Bringhurst (2008).
- *Combining Typefaces* by Tim Brown (2013).
- **The Geometry of Type: the Anatomy of 100 Essential Typefaces** by Stephen Coles (2013).
- *An Initiation in Typography* by Anne Denastas and Camille Gallet (2006).
- *Finer Points in the Spacing & Arrangement of Type* by Geoffrey Dowding (1995, after 1954 original).
- **Inside Paragraphs: Typographic Fundamentals** by Cyrus Highsmith (2013).
- *Detail in Typography* by Jost Hochuli (2008).
- *Shady Characters* by Keith Houston (2013).
- **Shaping Text** by Jan Middendorp (2012).
- *Grid Systems in Graphic Design* by Josef Müller-Brockmann (1996, after 1981 original).
- *Responsive Typography* by Jason Pamental (2014).
- *Typeface: Classic Typography for Contemporary Design* by Tamye Riggs.
- *The Typographic Desk Reference* by Theodore Rosendorf (2009).
- *Typography: A Manual of Design* by Emil Ruder (1967).
- *On Web Typography* by Jason Santa Maria (2014).
- *Rhyme & Reason: A Typographic Novel* by Erik Spiekermann (1987).
- *Getting it Right with Type* by Victoria Squire (2006).

- *Stop Stealing Sheep & Find Out How Type Works* by Erik Spiekermann & E. M. Ginger (2000).
- *The New Typography* by Jan Tschichold (1995, after 1928 original).
- *Asymmetric Typography* by Jan Tschichold (1967, after 1935 original).
- *Type Matters!* by Jim Williams (2012).

Printed periodicals

- *Baseline* magazine.
- *Circular*, annual publication of the Typographic Circle.
- *Eye* magazine.
- *Grafik* magazine.
- *The Recorder*, published by Monotype.
- *Ultrabold*, the journal of St Bride Library.

Articles and papers

- ‘A closer look at TrueType hinting’ (<http://wbtyp.net/4>) by Tim Ahrens on the *Typekit blog* (2010).
- ‘Reading Design’ (<http://wbtyp.net/5>) by Dean Allen in *A List Apart* (2001).
- ‘A Dao of Web Design’ (<http://wbtyp.net/6>) by John Allsop in *A List Apart* (2000).
- ‘css Grid, Flexbox And Box Alignment: Our New System For Web Layout’ (<http://wbtyp.net/7>) by Rachel Andrew in *Smashing Magazine* (2016).
- ‘Font Size and Viewing Distance of Handheld Smart Phones’ (<http://wbtyp.net/8>) by Yuliya Bababekova, Mark Rosenfield, Jennifer E. Hue, Rae R. Huang in *Optometry & Vision Science* Vol 88, Issue 7 (2011).
- ‘Thirteen Ways of Looking at a Typeface’ (<http://wbtyp.net/9>) by Michael Bierut in *Design Observer* (2007).
- ‘Dutch Type Design’ (<http://wbtyp.net/10>) by Peter Biľák in *Typotheque articles* (2004).
- ‘Best Practices of Combining Typefaces’ (<http://wbtyp.net/11>) by Douglas Bonneville in *Smashing Magazine* (2010).
- ‘Line lengths when reading from a screen’ (<http://wbtyp.net/12>) by Joe Clark (2005).
- ‘Typography specification’ (<http://wbtyp.net/13>) in the *Co-op Design Manual* (2017).
- ‘How physical text layout affects reading from screen’ (<http://wbtyp.net/14>) by M.C. Dyson in *Behaviour and Information Technology* 23/6 (2004).

- ‘[How We Load Web Fonts Progressively](http://wbtyp.net/151)’ (<http://wbtyp.net/151>) by Filament Group.
- ‘[Tablesaw](http://wbtyp.net/15)’ (<http://wbtyp.net/15>) responsive table plug-ins by Filament Group.
- ‘[Responsive Tables](http://wbtyp.net/16)’ (<http://wbtyp.net/16>) by Aaron Gustafson on *Easy Designs blog* (2013).
- ‘[Multi-typeface design](http://wbtyp.net/17)’ (<http://wbtyp.net/17>) by Bethany Heck on *Prototypr.io* (2016).
- ‘[Experimenting with customizable, retina-pixel underlining](http://wbtyp.net/18)’ (<http://wbtyp.net/18>) by Gary Hepting on *Codepen*.
- ‘[Introducing OpenType Variable Fonts](http://wbtyp.net/19)’ (<http://wbtyp.net/19>) by John Hudson (2016).
- ‘[Type classifications are useful, but the commons ones are not](http://wbtyp.net/20)’ (<http://wbtyp.net/20>) by Indra Kupferschmidt (2012).
- ‘[Some type genres explained](http://wbtyp.net/21)’ (<http://wbtyp.net/21>) by Indra Kupferschmidt (2016).
- ‘[The Aesthetics of Reading](http://wbtyp.net/22)’ PDF (<http://wbtyp.net/22>) by Kevin Larson & Rosalind W. Picard (2005).
- ‘[Using the System Font in Web Content](http://wbtyp.net/23)’ (<http://wbtyp.net/23>) by Myles Maxfield on the *Webkit blog* (2015).
- ‘[A Simpler Page](http://wbtyp.net/24)’ (<http://wbtyp.net/24>) by Craig Mod in *A List Apart* Issue 321 (2011).
- ‘[F-Shaped Pattern For Reading Web Content](http://wbtyp.net/25)’ (<http://wbtyp.net/25>) by Jakob Nielsen on *Nielsen Norman Group* (2006).
- ‘[The Ultimate Guide To iPhone Resolutions](http://wbtyp.net/26)’ (<http://wbtyp.net/26>) from PaintCode.
- ‘[Responsive Typography: The Basics](http://wbtyp.net/27)’ (<http://wbtyp.net/27>) by Oliver Reichenstein on *iA.net* (2012).
- ‘[Web Design is 95% Typography](http://wbtyp.net/28)’ (<http://wbtyp.net/28>) by Oliver Reichenstein on *iA.net* (2006).
- ‘[The Futures of Typography](http://wbtyp.net/29)’ (<http://wbtyp.net/29>) by Robin Rendle (2017).
- ‘[The New Web Typography](http://wbtyp.net/30)’ (<http://wbtyp.net/30>) by Robin Rendle (2016).
- ‘[Variable Fonts](http://wbtyp.net/31)’ (<http://wbtyp.net/31>) on the *Responsive Web Design* podcast (2017).
- ‘[The Importance of Being Ernestine](http://wbtyp.net/32)’ (<http://wbtyp.net/32>) by Nina Stössinger at *TYPO Berlin* (2012).
- ‘[Is type hinting for screens still relevant?](http://wbtyp.net/33)’ (<http://wbtyp.net/33>) thread on *TypeDrawers*.

- ‘[The Vignelli Canon](http://wbtyp.net/34)’ PDF (<http://wbtyp.net/34>) by Massimo Vignelli (2009).
- ‘[The Crystal Goblet, or Printing Should Be Invisible](http://wbtyp.net/35)’ (<http://wbtyp.net/35>) by Beatrice Warde (1930).
- ‘[Crafting link underlines on Medium](http://wbtyp.net/36)’ (<http://wbtyp.net/36>) by Marcin Wichary in *Designing Medium* (2014).

Websites

- [A List Apart typography](http://wbtyp.net/37) (<http://wbtyp.net/37>).
- [Alphabettes](http://wbtyp.net/38) (<http://wbtyp.net/38>) – work, commentary and research by women in type.
- [Clagnut typography](http://wbtyp.net/39) (<http://wbtyp.net/39>) – Richard Rutter’s blog posts on typography.
- [The Elements of Typography Style Applied to the Web](http://wbtyp.net/40) (<http://wbtyp.net/40>).
- [I Love Typography](http://wbtyp.net/41) (<http://wbtyp.net/41>).
- [International Society of Typographic Designers](http://wbtyp.net/42) (<http://wbtyp.net/42>) (ISTD).
- [Practical Typography](http://wbtyp.net/43) (<http://wbtyp.net/43>) by Matthew Butterick.
- [Professional Web Typography](http://wbtyp.net/44) (<http://wbtyp.net/44>) by Donny Truong.
- [Shady Characters](http://wbtyp.net/45) (<http://wbtyp.net/45>) by Keith Houston.
- [Typedia](http://wbtyp.net/46) (<http://wbtyp.net/46>).
- [Typedrawers](http://wbtyp.net/154) (<http://wbtyp.net/154>) community.
- [Typekit blog](http://wbtyp.net/47) (<http://wbtyp.net/47>).
- [Typekit Practice](http://wbtyp.net/48) (<http://wbtyp.net/48>).
- [Type Tasting](http://wbtyp.net/51) (<http://wbtyp.net/51>) with Sarah Hyndman.
- [Typographica](http://wbtyp.net/49) (<http://wbtyp.net/49>) – type reviews, books and commentary.
- [Typography.Guru](http://wbtyp.net/153) (<http://wbtyp.net/153>) community.
- [Typography on the Web](http://wbtyp.net/50) (<http://wbtyp.net/50>) – resources and references.

Inspiration

- [Fonts in Use](http://wbtyp.net/52) (<http://wbtyp.net/52>)
- [Typeroom](http://wbtyp.net/53) (<http://wbtyp.net/53>)
- [TypeToken](http://wbtyp.net/54) (<http://wbtyp.net/54>)
- [Typewolf](http://wbtyp.net/55) (<http://wbtyp.net/55>)
- [Typ.io](http://wbtyp.net/56) (<http://wbtyp.net/56>)

Tools and reference

- [Aspect value calculator](http://wbtyp.net/57) (http://wbtyp.net/57) using font-size-adjust.
- [Aspect value calculator](http://wbtyp.net/58) (http://wbtyp.net/58) using ex units.
- [Axis-Praxis](http://wbtyp.net/60) (http://wbtyp.net/60) – variable font playground.
- [Can I Use](http://wbtyp.net/61) (http://wbtyp.net/61) – browser support for HTML, CSS, etc.
- [Compute aspect value](http://wbtyp.net/59) (http://wbtyp.net/59) displaying installed fonts.
- [CSS Tricks](http://wbtyp.net/62) (http://wbtyp.net/62).
- [Explorations in Typography DIY](http://wbtyp.net/63) (http://wbtyp.net/63).
- [Font Locator](http://wbtyp.net/64) (http://wbtyp.net/64) iOS app.
- [Identifont](http://wbtyp.net/65) (http://wbtyp.net/65).
- [Matcherator](http://wbtyp.net/67) (http://wbtyp.net/67).
- [MegaType](http://wbtyp.net/68) (http://wbtyp.net/68) – baseline alignment project.
- [Modular Scale](http://wbtyp.net/69) (http://wbtyp.net/69).
- [OpenType features](http://wbtyp.net/71) (http://wbtyp.net/71).
- [Quotation marks](http://wbtyp.net/66) (http://wbtyp.net/66) on Wikipedia.
- [Quotes & Accents](http://wbtyp.net/70) (http://wbtyp.net/70).
- [Responsive typography experiments](http://wbtyp.net/72) (http://wbtyp.net/72) using face detection.
- [Simple Scale](http://wbtyp.net/73) (http://wbtyp.net/73) – modular scale generator.
- [Size Calc](http://wbtyp.net/74) (http://wbtyp.net/74).
- [Sketch Measure](http://wbtyp.net/75) (http://wbtyp.net/75).
- [State of Webtype](http://wbtyp.net/76) (http://wbtyp.net/76) – support for typographic features on the web.
- [Typecast](http://wbtyp.net/77) (http://wbtyp.net/77) – browser-based typesetting and design tool.
- [Type Sample](http://wbtyp.net/78) (http://wbtyp.net/78) – tool for identifying and sampling web fonts.
- [TypeTester](http://wbtyp.net/79) (http://wbtyp.net/79) – online typography editor.
- [Typography Supply](http://wbtyp.net/80) (http://wbtyp.net/80) – an inventory of typographic tools.
- [University of Oxford Style Guide](http://wbtyp.net/81) (http://wbtyp.net/81).
- [w3c advice on language tags in HTML](http://wbtyp.net/82) (http://wbtyp.net/82).
- [wcAG 2.0 Technique G183 colour contrast resources](http://wbtyp.net/85) (http://wbtyp.net/85).
- [Web Content Accessibility Guidelines 2.0](http://wbtyp.net/83) (http://wbtyp.net/83).
- [Web Font Specimen](http://wbtyp.net/84) (http://wbtyp.net/84).
- [Website Style Guide Resources](http://wbtyp.net/88) (http://wbtyp.net/88).
- [WhatFont](http://wbtyp.net/86) (http://wbtyp.net/86) – extension and bookmarklet.
- [WhatTheFont!](http://wbtyp.net/87) (http://wbtyp.net/87).

CSS Modules

- [Backgrounds and Borders Level 3](http://wbtyp.net/90) (<http://wbtyp.net/90>)
- [css Level 2 Revision 1 \(css 2.1\) Specification](http://wbtyp.net/89) (<http://wbtyp.net/89>)
- [Display Level 3](http://wbtyp.net/91) (<http://wbtyp.net/91>)
- [Flexible Box Layout Level 1](http://wbtyp.net/92) (<http://wbtyp.net/92>)
- [Font Loading Level 3](http://wbtyp.net/93) (<http://wbtyp.net/93>)
- [Fonts Level 3](http://wbtyp.net/94) (<http://wbtyp.net/94>)
- [Fonts Level 4](http://wbtyp.net/95) (<http://wbtyp.net/95>)
- [Grid Layout Level 1](http://wbtyp.net/96) (<http://wbtyp.net/96>)
- [Inline Layout Level 3](http://wbtyp.net/97) (<http://wbtyp.net/97>)
- [Media Queries](http://wbtyp.net/98) (<http://wbtyp.net/98>)
- [Multi-column Layout](http://wbtyp.net/99) (<http://wbtyp.net/99>)
- [Selectors Level 3](http://wbtyp.net/100) (<http://wbtyp.net/100>)
- [Text Decoration Level 3](http://wbtyp.net/101) (<http://wbtyp.net/101>)
- [Text Level 3](http://wbtyp.net/102) (<http://wbtyp.net/102>)
- [Text Level 4](http://wbtyp.net/103) (<http://wbtyp.net/103>)
- [Transforms Level 1](http://wbtyp.net/104) (<http://wbtyp.net/104>)
- [Values and Units Level 3](http://wbtyp.net/105) (<http://wbtyp.net/105>)

Foundries

- [A2-Type](http://wbtyp.net/106) (<http://wbtyp.net/106>)
- [Colophon](http://wbtyp.net/107) (<http://wbtyp.net/107>)
- [Commercial Type](http://wbtyp.net/108) (<http://wbtyp.net/108>)
- [Dalton Maag](http://wbtyp.net/109) (<http://wbtyp.net/109>)
- [Darden Studio](http://wbtyp.net/110) (<http://wbtyp.net/110>)
- [DSType](http://wbtyp.net/111) (<http://wbtyp.net/111>)
- [Emigre](http://wbtyp.net/112) (<http://wbtyp.net/112>)
- [Exljbris](http://wbtyp.net/113) (<http://wbtyp.net/113>)
- [Font Bureau](http://wbtyp.net/114) (<http://wbtyp.net/114>)
- [FontFont](http://wbtyp.net/115) (<http://wbtyp.net/115>)
- [Fontsmith](http://wbtyp.net/116) (<http://wbtyp.net/116>)
- [The Foundry](http://wbtyp.net/137) (<http://wbtyp.net/137>)
- [Frere-Jones Type](http://wbtyp.net/117) (<http://wbtyp.net/117>)
- [G-Type](http://wbtyp.net/118) (<http://wbtyp.net/118>)
- [Hoefler & Co.](http://wbtyp.net/119) (<http://wbtyp.net/119>)
- [House Industries](http://wbtyp.net/120) (<http://wbtyp.net/120>)
- [HvD Fonts](http://wbtyp.net/121) (<http://wbtyp.net/121>)
- [Jeremy Tankard Typography](http://wbtyp.net/122) (<http://wbtyp.net/122>)
- [Just Another Foundry](http://wbtyp.net/123) (<http://wbtyp.net/123>)

- [Klim Type Foundry](http://wbtyp.net/124) (<http://wbtyp.net/124>)
- [Letters from Sweden](http://wbtyp.net/125) (<http://wbtyp.net/125>)
- [LucasFonts](http://wbtyp.net/126) (<http://wbtyp.net/126>)
- [Mark Simonson Studio](http://wbtyp.net/127) (<http://wbtyp.net/127>)
- [Milieu Grotesque](http://wbtyp.net/128) (<http://wbtyp.net/128>)
- [Mostardesign](http://wbtyp.net/129) (<http://wbtyp.net/129>)
- [OurType](http://wbtyp.net/130) (<http://wbtyp.net/130>)
- [Parachute](http://wbtyp.net/131) (<http://wbtyp.net/131>)
- [Playtype](http://wbtyp.net/132) (<http://wbtyp.net/132>)
- [Positype](http://wbtyp.net/133) (<http://wbtyp.net/133>)
- [PSTL \(ps.type\)](http://wbtyp.net/134) (<http://wbtyp.net/134>)
- [Rosetta](http://wbtyp.net/135) (<http://wbtyp.net/135>)
- [Sudtipos](http://wbtyp.net/136) (<http://wbtyp.net/136>)
- [Typejockeys](http://wbtyp.net/139) (<http://wbtyp.net/139>)
- [TypeTogether](http://wbtyp.net/138) (<http://wbtyp.net/138>)
- [Typofonderie](http://wbtyp.net/140) (<http://wbtyp.net/140>)
- [Typolar](http://wbtyp.net/141) (<http://wbtyp.net/141>)
- [Typotheque](http://wbtyp.net/142) (<http://wbtyp.net/142>)
- [Underware](http://wbtyp.net/143) (<http://wbtyp.net/143>)
- [VirusFonts](http://wbtyp.net/144) (<http://wbtyp.net/144>)

Links references

All the websites linked to in this book use a custom URL shortener of the form <http://wbtyp.net/n>. The full length URLs are listed here in numerical sequence.

1. <http://book.webtypography.net/>
2. <http://clagnut.com/>
3. <http://clearleft.com/>
4. <https://blog.typekit.com/2010/12/14/a-closer-look-at-truetype-hinting/>
5. <http://alistapart.com/article/readingdesign>
6. <https://alistapart.com/article/dao>
7. <https://www.smashingmagazine.com/2016/11/css-grids-flexbox-and-box-alignment-our-new-system-for-web-layout/>
8. http://journals.lww.com/optvissci/Fulltext/2011/07000/Font_Size_and_Viewing_Distance_of_Handheld_Smart.5.aspx
9. <http://designobserver.com/feature/thirteen-ways-of-looking-at-a-typeface/5497>

10. https://www.typotheque.com/articles/dutch_type_design
11. <http://www.smashingmagazine.com/2010/11/04/best-practices-of-combining-typefaces/>
12. <http://blog.fawny.org/2005/09/21/measures/>
13. <https://coop-design-manual.herokuapp.com/styles/typography.html>
14. <http://scholar.google.com/scholar?q=%22How+physical+text+layout+affects+reading+from+screen%22>
15. <https://github.com/filamentgroup/tablessaw>
16. <http://blog.easy-designs.net/archives/responsive-tables/>
17. <https://blog.prototypio.io/the-value-of-multi-typeface-design-cccd67227boee>
18. <http://codepen.io/ghepting/pen/tLnHK/>
19. <https://medium.com/@tiro/12ba6cd2369>
20. <http://kupferschrift.de/cms/2012/03/on-classifications/>
21. <http://kupferschrift.de/cms/2016/01/type-classification-texts/>
22. <http://affect.media.mit.edu/pdfs/05.larson-picard.pdf>
23. <https://webkit.org/blog/3709/using-the-system-font-in-web-content/>
24. <http://alistapart.com/article/a-simpler-page>
25. <https://www.nngroup.com/articles/f-shaped-pattern-reading-web-content/>
26. <https://www.paintcodeapp.com/news/ultimate-guide-to-iphone-resolutions>
27. <http://ia.net/blog/responsive-typography-the-basics/>
28. <https://ia.net/topics/the-web-is-all-about-typography-period/>
29. <https://robinrendle.com/essays/futures-of-typography/>
30. <https://robinrendle.com/essays/new-web-of-typography/>
31. <http://responsivewebdesign.com/podcast/variable-fonts>
32. <http://www.typtalks.com/videos/the-importance-of-being-ernestine/>
33. http://typedrawers.com/discussion/comment/24051/#Comment_24051
34. <http://www.vignelli.com/canon.pdf>
35. <http://gmunch.home.pipeline.com/typo-L/misc/ward.htm>
36. <https://medium.design/7c03a9274f9>
37. <http://alistapart.com/topic/typography-web-fonts>
38. <http://alphabettes.com/>
39. <http://clagnut.com/archive/typography/>
40. <http://webtypography.net/>
41. <http://ilovetypography.com/>
42. <http://www.istd.org.uk/>
43. <http://practicaltypography.com/>
44. <https://prowebtype.com/>

45. <http://www.shadycharacters.co.uk/>
46. <http://typedia.com/>
47. <http://blog.typekit.com/>
48. <http://practice.typekit.com/>
49. <http://typographica.org/>
50. <https://typographyontheweb.com/>
51. <http://typetasting.com/>
52. <http://fontsinuse.com/>
53. <http://www.typeroom.eu/>
54. <http://www.typetoken.net/>
55. <http://typewolf.com/>
56. <http://typ.io/>
57. <http://clagnut.com/sandbox/font-size-adjust.html>
58. <http://clagnut.com/sandbox/font-size-adjust-ex.html>
59. <http://brunildo.org/test/fontlist3.html>
60. <http://www.axis-praxis.org/>
61. <http://caniuse.com/>
62. <https://css-tricks.com/>
63. <http://explorationsintypography.com/diy/>
64. <https://itunes.apple.com/gb/app/font-locator/id582240320?mt=8>
65. <http://www.identifont.com/>
66. https://en.wikipedia.org/wiki/Quotation_mark
67. <https://www.fontspring.com/matcherator>
68. <http://www.studiothick.com/essays/web-typography-is-broken>
69. <http://modularscale.com/>
70. <http://quotesandaccents.com/>
71. https://en.wikipedia.org/wiki/List_of_typographic_features#OpenType_t typographic_features
72. <http://webdesign.maratz.com/lab/responsivetypography/>
73. <http://simplescale.online/>
74. <https://sizecalc.com/>
75. <http://utom.design/measure/>
76. <http://stateofwebtype.com/>
77. <http://typecast.com/>
78. <http://www.typesample.com/>
79. <https://typetester.org/>
80. <http://typography.supply/>
81. <https://www.ox.ac.uk/public-affairs/style-guide>
82. <https://www.w3.org/International/articles/language-tags/>
83. <https://www.w3.org/TR/WCAG2o/>

84. <http://webfontspecimen.com/>
85. <https://www.w3.org/tr/WCAG2o-TECHS/G183.html>
#G183-resources
86. <http://www.chengyinliu.com/whatfont.html>
87. <https://www.myfonts.com/WhatTheFont/>
88. <http://styleguides.io/>
89. <https://www.w3.org/tr/css2/>
90. <https://www.w3.org/tr/css3-background/>
91. <https://www.w3.org/tr/css-display-3/>
92. <https://www.w3.org/tr/css-flexbox-1/>
93. <https://www.w3.org/tr/css-font-loading-3/>
94. <https://www.w3.org/tr/css-fonts-3/>
95. <https://drafts.csswg.org/css-fonts-4/>
96. <https://www.w3.org/tr/css-grid-1/>
97. <https://www.w3.org/tr/css-inline-3/>
98. <https://www.w3.org/tr/css3-mediaqueries/>
99. <https://www.w3.org/tr/css3-multicol/>
100. <https://www.w3.org/tr/selectors/>
101. <https://www.w3.org/tr/css-text-decor-3/>
102. <https://www.w3.org/tr/css-text-3/>
103. <https://drafts.csswg.org/css-text-4/>
104. <https://www.w3.org/tr/css-transforms-1/>
105. <https://www.w3.org/tr/css3-values/>
106. <https://www.a2-type.co.uk>
107. <https://www.colophon-foundry.org>
108. <https://commercialtype.com>
109. <https://www.daltonmaag.com/>
110. <https://www.dardenstudio.com/>
111. <http://www.dstype.com/>
112. <http://emigre.com/>
113. <http://www.exljbris.com/>
114. <http://www.fontbureau.com/>
115. <https://www.fontfont.com>
116. <http://www.fontsmith.com/>
117. <https://frerejones.com/>
118. <http://www.g-type.com/>
119. <https://www.typography.com/>
120. <https://houseind.com/>
121. <http://hvdfonts.com/>
122. <http://typography.net/>

-
- 123. <http://justanotherfoundry.com/>
 - 124. <https://klim.co.nz/>
 - 125. <http://lettersfromsweden.se/>
 - 126. <http://www.lucasfonts.com/>
 - 127. <http://www.marksimonson.com/>
 - 128. <http://www.milieugrotesque.com>
 - 129. <http://motyfo.com>
 - 130. <https://ourtype.com>
 - 131. <http://www.parachutefonts.com/>
 - 132. <https://playtype.com/>
 - 133. <https://positype.com/>
 - 134. <https://pstypelab.com>
 - 135. <https://www.rosettatype.com/>
 - 136. <http://sudtipos.com/>
 - 137. <http://www.foundrytypes.co.uk/>
 - 138. <http://type-together.com/>
 - 139. <http://www.typejockeys.com/>
 - 140. <https://typofonderie.com/>
 - 141. <http://www.typolar.com/>
 - 142. <https://www.typotheque.com/>
 - 143. <http://www.underware.nl/>
 - 144. <http://virusfonts.com/>
 - 145. <https://lists.w3.org/Archives/Public/www-style/2015Jan/0078.html>
 - 146. <http://jontangerine.com/>
 - 147. <https://thegreatdiscontent.com/interview/alison-sudol>
 - 148. <https://css-tricks.com/tag/responsive-tables/>
 - 149. <https://abookapart.com/products/get-ready-for-css-grid-layout>
 - 150. <http://nyc.ampersandconf.com/>
 - 151. <https://www.filamentgroup.com/lab/font-events.html>
 - 152. <https://www.kickstarter.com/projects/clagnut/web-typography-a-handbook>
 - 153. <https://typography.guru/>
 - 154. <http://typedrawers.com>

CSS index

Index of featured css properties and other syntax.

`::after` 107, 163
`::before` 107, 109, 163
`::first-line` 98
`:first-of-type` 99, 101
`@font-face` 269, 284
`@media` 58, 61, 94, 95, 143, 160, 181, 191, 250
`-moz-osx-font-smoothing` 201
`@supports` 82, 99, 117, 120, 124, 125, 147, 151, 166, 169
`-webkit-font-smoothing` 201
`background` 102
`background-image` 118
`background-size` 102
`border-collapse` 134
`calc()` 158
`caption-side` 135
`column-count` 190
`column-gap` 190, 191
`column-rule` 190, 191
`columns` 191
`column-width` 190, 191
`content` 101, 102, 105, 107, 108, 109, 143, 163
`counter-increment` 109
`counter-reset` 108, 109
`display` 105, 143
`font` 41, 261
`font-display` 274, 277
`font-family` 41, 42, 205, 260, 270, 271, 274, 276, 277, 278, 279, 284
`font-feature-settings` 81, 82, 83, 84, 120, 122, 125, 133, 147, 151, 167, 283
`font-kerning` 147
`font-optical-sizing` 283
`font-size` 16, 32, 38, 41, 42, 94, 124, 125, 155, 157, 250, 261, 279

font-size-adjust 279
font-smoothing 201
font-stretch 283
font-style 42, 111, 270, 271, 283, 284
font-synthesis 112
font-variant-alternates 166, 168, 169
font-variant-caps 85, 86, 99
font-variant-ligatures 81, 82, 84, 151
font-variant-numeric 120, 122, 133
font-variant-position 123, 124, 125
font-variation-settings 283, 284
font-weight 42, 98, 112, 113, 270, 271, 272, 283, 284
hyphenate-limit-after 54
hyphenate-limit-before 54
hyphenate-limit-chars 53, 54
hyphenate-limit-last 56
hyphenate-limit-lines 52, 54
hyphenation-limit-zone 54
hyphens 51
initial-letter 100, 101
letter-spacing 149, 150, 151, 257
line-height 37, 38, 39, 41, 42, 98, 104, 134, 164, 172, 173, 279, 311
list-style 109
max-width 25, 141, 143, 181
min-height 95, 160, 191
min-width 57, 58, 61, 62, 94, 95, 160, 181, 182, 250
overflow-x 141
quotes 107, 108
text-align 43, 143, 149, 163
text-align-last 47
text-decoration 116, 117
text-decoration-color 117
text-decoration-skip 117
text-decoration-style 117
text-indent 104, 109
text-transform 86, 149
transform 140
transform-origin 140
vertical-align 124, 125, 172, 175

Thank you

This book was over a decade in the making. Over the course of that time I have built up a significant debt of gratitude, some of which I will now attempt to recognise.

In 2005, as a frustrated web designer, I published [The Elements of Typographic Style Applied to the Web](http://wbtyp.net/40) (<http://wbtyp.net/40>) as my attempt at applying Robert Bringhurst's guidelines to the web. My intention was to put to bed some of the myths surrounding web typography, and show that with a little effort, good typography was perfectly possible on the web.



Mark Boulton and Richard Rutter at SXSW 2007. *Mark Norman Francis*

Two years later, that site led to the SXSW talk Mark eloquently describes in the [Foreword](#). Realising we were not alone in believing the web could and should be a better place for typography, a book seemed a logical outcome.

The next ten years saw a series of setbacks and competing commitments which meant our plan to write the book together never saw fruition. In that time, Mark set up his own publishing company and we seemed set to go, but Five Simple Steps shut its doors with Mark's departure to Monotype. I decided the book still needed to see the light of day and made a commitment to write it myself and do the best job I possibly could.

Knowing the upfront costs required for professional editing, proofreading, typesetting and – most of all – printing, I turned to [Kickstarter](http://wbtyp.net/152) (<http://wbtyp.net/152>) to get funding. Evidently, I was successful; in fact, I reached my funding target after only 48 hours. For that, I owe my biggest thanks to the trust, generosity and patience of all 699 people who backed me and this book on Kickstarter. You are listed below.

Without whom

The person responsible for getting me hooked on typography is a talented designer named Ian Lewis; thank you, Ian, for lending me your copies of Bringhurst and Dowding. Further thanks go to Mark Boulton and Jon Tan for helping shape the book. My brother Dave did a fantastic job with the printed book and many of the illustrations. Owen Gregory was stellar in his painstaking editing, proofreading and words of support (any errors or aberrations are entirely my fault).

Thank you to all my colleagues at Clearleft for your feedback and encouragement throughout, and to the wider web and type community for your help and sharing of knowledge – you are too numerous to name here (which is wonderful in itself), but hopefully you know who you are. And finally, thanks to my friend Jason for these words of encouragement from a fellow typophile and author:

- me: “*What kind of an idiot would try to write a book on web typography when things keep changing all the time?*”
- jasonsantamaria: “*It’s the stupidest thing to do. Except for all the other things.*”

Backers

Once again, thanks for your trust, generosity, encouragement and patience. It is a statement of fact that this book would not have been possible without you.

Kristof Bernaert • Jon Hicks • Jay Kaufmann

Stuart Miller • Marc Thiele

Matt Alcock • Chris Armstrong • Christos Asteriou • Nic Barajas

Rogier Barendregt • Natanael Bjorkgard • Joshua Bock • Pablo Boerr

Scott Boms • Emma Boulton • François Bourgaux • Matthew Brown

Rade Brujic • Jaime Caballero • Chris Cannam • Bea Canut

Pedro Carrasquilla • Alistair Chisholm • Andy Clarke • Radrad Co

John F Croston III • Niurka Rodriguez de Fernandez • Jennifer de la Fuente

Theresa Desfosses • Stuart Fano • Oliver Farrell • Jason Garber

Mathias Gmeiner • Simon Goumaz • Michael Gravel • David W Gray

Duncan Gray • Aegir Hallmundur • Tim Harper • Mike Hartley

Kevin Healy • Luke Hill • Andy Hume • Grant Hutchinson • Elisabeth Irgens

Dennis J. Jackson II • Marc Jenkins • Serge K. Keller • Jonathan Kempf

Kerning Conference • Taroh Kogure • Zhifeng Koh • Charlie Langridge

Simon Law • Yoli Lawrence • Mark Lednor • Chris Lilley • Jeff Long

Andres Lopez Josenge • Woody MacDuffie • Dan Mall • Chris Marshall

Niqui Merret • Vitorio Miliano • Ant Miller • David Miller • Fleur Mongan

David Moulton • Marko Mrdjenović • Keith Mutch • Marcel Maurice Naef

Jamie Neely • Sebastian Nikolaus • John Norris • Anya Ohmai

John Oxton-King • Nick Page • Jason Pamental • Grant Pan • Jackie Parkin

Doc Parsons • Cole Peters • David Randall • Marco Reisner • Dan Richmond

Stu Robson • David Roessli • Derek A. Rosenstrauch • Craig Saily

Wade Sakundiak • Jason Santa Maria • Joel Santiago • Jorren Schauwaert

Christopher Schmitt • Alex Sexton • Peter Simpson • Dennis Skelton

Matthias Slovig • Vincent Smedinga • Matthew Smith • Andy Spencer

Bevan Stephens • Terin Stock • Greg Storey • Rachel Thompson

Fabio Venni • Dylan Walsh • Trent Walton • Tiffany Wardle

Samantha Warren • Ludwig Wendzich • Jeremy Whitbred

Michelle Whitman • Yassir Yahya • Jeremy J. Zahner

Marius Aabel • Alexandra Aas • Matthijs Abeelen • Jeremy Abrams
Philipp Acsany • Filip S. Adamsen • Georg S. Adamsen • Ire Aderinokun
Charles Adler • Cédric Aellen • Steven Ametjan • Artem Aminov • Anarore
Rachel Andrew • Josh Apostolopoulos • Fred Aragao • Alberto Arias
Marta Armada • Ben Armstrong • Ari Arsyadi • Andrew Austin • James Aylett
Pete Aylward • Derik Badman • Stephen Baggs • D. Bailey • Jasper Bakker
André Baldinger • Joel Bardsley • Matthew Barger • David Barker • Michael Barlow
Mike Barmonde • Steve Barnett • Jim Barraud • Sander Baumann • Matthew Beech
Sami Beese • Andy Bell • Cory Bennett • Julian Benton • Jason Birnie
Ashley Bischoff • Stefan Boehm • Timo Bontenbal • Jonas Boserup • Lorenzo Bovini
Matthew Boyd • Myles Braithwaite • Nick Bramwell • Ralph Brandi
Charlie Briggs • Florian Brinkmann • Jim Brittain • Brad Brooks • Tim Brown
Sarah Bryant and David Allen • Matthew Buchanan • Nat Buckley
Michael Bundscherer • James Burt • Doru Calangea • Nick Caldwell • Jonas Calvo
Ian Cameron • Eric Campos • Graham Carlyle • Ann Carrier • Timothy Carroll
Christopher Cashdollar • Katrina Cass • Jeremy Champion • Roinson Chan
Alex Chang • Kevin Chang • Justin Charles • Sylvain Charpiot • John R Chennells
Ready Chi • Frank Chimero • Hannah Cho • Shaun Choh • Stephanie A. Chuah
Adam Churchill • Nikki Clark • Michael Clayton • Stephen Collins
Timothy Conroy • Katy Coope • Charles Cooper • Joe Corcoran • Katherine Cory
Jérôme Coupé • Andy Cowan • Simon Cox • Joshua Crane • Alexandra Crasqi
Andy Croll • Andrew Crookston • Chip Cullen • Pedro Custódio • Simon Cutajar
Azlan Cuttilan • Martin Dady • David Darke • Ian Dash • Nac Datta • Andy Davies
Adam Dawkins • Johan de Koning • Davy De Pauw • Mark De Solla Price
Randall De Weerd • Hans de Wolf • Pablo Defendini • Jed Dela Cruz
Marissa Demers • Andy Dennis • Remko Dijksma • Covington Doan
Graham Dobie • Jordan Robert Dobson • Ben Doherty • Benjamin Dong
Luke Dorny • Paul Downey • Matias Doyle • Joel Drapper • Ryan Duffy
Patrick Dugan • Seb Duggan • Alistair Duggin • Marko Dugonjic • Ben Duguid
Tom Dunn • Mark Durbin • Ali Edin • Alex Edwards • Sylvia Egger • Eric Eggert
Johannes Ekman • Jacob Ela • John Ellis • Emanuel Eriksson • Miguel Espi
Adriano Esteves • Guillermo Esteves • Eric Evans • Nigel Ewan • Ana Ferreira
Marc Fini • David Finnegan • Sean Fitzpatrick • Levi Flair • Travis Fleck
Christian Flindt • Casey Forbes • Nathan Ford • Travis Forden • Goncalo Forte
Aurélien Foutoyet • Mark Norman Francis • Dan Freeman • Martin Freer
Jasmine Rae Friedrich • Ville Frisk • Michael Fruehmann • Jitachi Garcia
Andy Gardner • Mike Garrett • Joel Gerhold • Rae Gibbs • Sebastian Gilits
Kyle Gillen • Jean-Marc Giorgi • Mark Glassford • Jon Gold • Robert Golias
Mart Gordon • Mike Gowen • Graeme • Lee Graham • Steve Graham • Vojta Grec

John Griffiths • Scott Gruber • Yannick Guti • Gilbert Guttmann • Celeste H Maximilian Haack • Ken Haase • Brooke Hamilton • Matt Hamm • Edd Hannay Andy Hansen • Peter Bo Hansen • Kiel Hanson • Tim Harbour • Chad Harris Simon Harriyott • Marc Hartwig • Jake Harvey • Brendan Hastings Michael Hastrich • Andy Hawkes • Stephen Hay • Tom Hazledine • Val Head Krystyn Heide • Erwin Heiser • Rafael Hernandez • Ralf Herrmann Mike Hickman • Matt Hill • Kim Hjortholm • Sven Hoehn • Peter Hofmann Joan Højberg • Helen V. Holmes • Jonathan Holst • George B Hopkins Matt Hornsby • Evey Huang • David Hughes • Chris Hurst • Peter Imhoff Piotr Ingling • Zach Inglis • Borna Izad • DJ Jahn • Jaffry Jalal • Matthew James Rob James • Rock Jethwa • Zachary Jewell • Are Johannessen • Sean Johnson Mark Joint • Elias Jones • David Joseph • Michael Jovel • Giovanni Jubert Nina K • Peter Kaizer • Kuku Kaka • Kake • Laura Kalbag • Boril Karaivanov Noah Kardos-Fein • Joshua Katigbak • Karl Kaufmann • Éric Kavanagh Micky Kelleher • Patrick Kelley • Scott Kellum • Collin Kelly • Peter Kennaugh John Keyes • Chu Kin Shing • Grant Kindrick • Galen King • Howard Kistler Chris Korhonen • Joni Korpi • John Kosinski II • Jan-Paul Koudstaal Dave Kraayeveld • Akarawut Kraikhajornkiti • Jason Kunesh • Mark Kunzmann Lloyd Kupchanko • Indra Kupferschmid • David Kusiak • Ben Kutil • Giulia Laco Madeleine Lamou • Frode Lamøy • V Lau • Christian Lawrence • Dan Layton Joey Leech • Phil Leitch • Andy LeMay • David Lemon • Gerry Leonidas • Tim Lewis Stanley Lim • Jon Linklater-Johnson • Caren Litherland • Tom Livingston Paul Lloyd • Tracy Locke Castro • Dan London • Chris Lowe • Martin Lundberg Erin Lynch • Kirsten M • Matt Machell • Dannell Macilwraith • Jeanie Mackinder James Madson • Ricardo Magalhães • Alex Magill • Dominic Magnifico Barry Mahaffey • Diego Maldonado • Amer Mall • Chris Manvell • Yiran Mao Carina Marano • Michael Marcialis • Dominic Marcinkus • Fabien Marry JD Marsters • Scott Martin • Eric Martz • Frederic Marx • Simon Maslaveckas A.J. May • Nick McFarlane • Christina McGuire • Lenore McLean • Drew McLellan Mark McLoughlin • Matteo Menapace • Ryan Merrill • Monica Messaggi Kaya Woodie Milano • Mark Mitchell • Marc Mitrani • Stuart Moffatt • William Mon Alexander Moore • Chris Moore • Kevin Moran • Oskar Rough Mosumgaard Sarah Munt • Raul Murciano • Christopher Murphy • Cheryl Murray Julien Musluk • Stig Morten Myre • Andreas Nebiker • Nick Ng • Linh Nguyen Jörgen Nilsson • Lauri Niva • Ji Noh • Mary Nolan • Andreas Nymark Tibor チビカ Obert • Ivan Odbratt • Knight of Words • Dominik Ogilvie Hans Olav Elsebo • António Ornela • Tracy Osborn • Jack Osborne Johan Oskarsson • Oscar Palmér • andy parker • Benjamin Parry • Serina Patterson James Pellizzi • Christian Petersen • Luc Pettett • Pinnemouche Studio SPRL Lauren Plews • Geoff Pollard • Anders Pollas • Cobey Potter • David Powers

Ryan Price • Shawn Price • Volker Probst • Adam Procter • Martin Pulec
Rowdy Rabouw • Markus Rademacher • Apiram Raengpradub • Alan Ralph
David Rapson • Sara Regan • Ariëlla F. Reinders • Matthew Reynolds • Micah Rich
Kaj Rietberg • Dan Ritz • Susan Robertson • Olivia Rohan • François Romain
Alexa Roman • Jennifer Rourke • Steve Rowling • Martin Ruston • Daniel Ryan
Leanda Ryan • Ben S • Daniel Sabino de Souza • Gurur Sarbanoglu • Ben Sauer
Richard Saunders • Rob Sawkins • Yannick Schall • Michael Schlierbach
Prisca Schmarsow • Ivy Scholz • Rik Schreurs • Vivian Schung • Pete Schuster
Auyok Sean • Eric Semling • Ben Seymour • Kerem Shefik • Benna Shelanski
Michael Shields • Jonathan Sick • Jason Sidoryn • N Silas Munro • Josh Silverman
Yvonne Sim • Dean Simons • Jason Skaare • David Sky • Justin Slack • Matt Slack
Philip Slesarev Levin • David Smith • David Smith • Steve Smith • Jonathan Snook
Manuel Muñoz Solera • Jessica Spengler • Andy Staple • Bram Stein
Carsten Stemmler • Dom Stevens • Ron Stewart • Sergey Storchay • Dudley Storey
Jonas Strandell • Nathan Strange • Max Strebler • Yancey Strickler • Jason Stuart
Jeremy Stucki • Brian Suda • Matt Sugihara • Neil Summerour • Graeme Sutherland
Takeru Suzuki • Ólafur Sværrir Kjartansson • Craig Swanson • Ian Tadashi Moore
Peter Tait • Kevin Tamura • Tuija Tarkiainen • Mike Taylor • Robin Taylor
Max Temkin • Hasso Tepper • George Terezakis • Yorick Terweijden
Lee Theobald • Todd Thille • Gareth Thomas • Scott Thomas • Adrian Thompson
David Thompson • Rick Thurston • Martin Tiefenthaler • Matthew Tobiasz
Brian Townsend • Andrew Travers • Donny Truong • Eric Tsai • Simo Tuominen
David Turner • Luděk Uiberlay • Gerrit van Aaken • Dieter van Baarle
Rex Dylan van Coller • Ted van der Togt • Gertjan van Heertum • Steven Van Loy
Tom VanAntwerp • Angie Vandenberghe • Thomas Vander Wal • Johan Vandewalle
Hans Verschooten • Jitendra Vyas • Bjorn W • Paul Wagner • Trevin Wagner
Clive Walker • Nathan Walker • Thomas Wallace • Chengpeng Wang • Tony Wareck
Peter Warman • Brian Warren • David Watson • Luke Watts • Tristan Watts-Willis
Denise Weatherburn • Brad Weaver • Harry Weihe • Sebastian Werner • Jon West
Rob Weychert • Craig Whyte • Marc Wiesenthal • Richard Wiggins • Joe Wilde
Spencer Williams • Steffan Williams • Darrell Wilson • Richard Winchell
Tom Windsor • Aaron Wolf • Wong Ken Ming • Heetaek Woo • K. David Woolley
Gail Wooten-Votaw • Yan Yan • Hermit Yang • Hsin-You Yeh • Emily Young
Melina Zerbib • Zero Agency • Yu Zhou

Dedication



This book is dedicated to my wife Wendy and
my children Gracie and Laurence.

I'll be forever grateful for their unstinting support
and understanding while I sacrificed quality family
time to indulge myself in the two year undertaking
that this book became.

TYPGRAPHY is what comes between the author and the reader. This is as true on the web as it is in any other medium. If a text has anything at all significant to say, it needs a typographer's care, which will in turn be repaid by the reader's attention. If you design websites or use CSS then you are a typographer whether you know it or not.

This book is a practical guide and companion reference to all aspects of typography on the web. It deftly combines implementation details with typographic theory, and is ideal for designers, developers and anyone else involved in the process of creating a website.

“It’s THE book to buy. Stunning. An instant classic.”

— Andy Clarke, author of Hardboiled CSS

“Very useful.”

— Erik Spiekermann

*“Looks great, reads well, full of insight, deep knowledge and expertise.
If you design the web, you’ll need it.”*

“a pleasure to read, insightful and informative”

“a joy to read, so much nice detail”

“a great resource”

— Reviews across the web



Ampersand Type, Brighton, UK, 2017
ISBN: 978-0-9956642-0-3

