BETTER WEB TYPOGRAPHY FOR A BETTER WEB

MATEJ LATIN

Editor: Nick Jones
Designer: Matej Latin
June 2017

ISBN 978-1-9998095-0-8

betterwebtype.com
matejlatin.co.uk

CONTENT

Preface

Whether you're a self-taught web designer looking to expand your skills, a graphic designer keen to refresh your typography skills or a web developer looking to finally go down the road of tackling the unknown and possibly feared topic of typography, I salute you. By the end of the book you'll be one of the Better Web Type advocates—people who aren't happy with the current state of web typography and proactively doing something about it. You'll be a beacon in the dark universe of typography, leading others by example. It takes infinite curiosity and endless courage to go down this road and it's definitely worth it. For typography is a fundamental part of visual communication in our world. So by reading this book you won't just become a master of web typography, you'll be a master of visual communication through the largest medium in the world—the web.

This book is based on an email course that I launched on 14 February 2017. It was such a memorable moment and I never expected the course to become as popular as it did. I was sitting in a cosy coffee shop in Greenwich, London, when I clicked the "Start sending" button. I had always been flirting with an idea of writing a book but I never really knew what it would be about. Well, that became clear when people finishing the 10-lesson course started asking for more. They started asking for a book. The course was featured in Net magazine as the main, cover feature, and it was featured in Web Designer magazine as well as on Codrops, Designer News, Sidebar and CSS Tricks. This helped it grow but the feedback remained immensely positive—everybody knows that as soon as the number of users grows, that's when the negative reviews start coming in but it wasn't the case with the Better Web Type course. At the time of writing this paragraph, the course has 702 reviews and the average rating is at 4.7 out of 5.

There was one thing that was quite hard about the course though: the part where I had to trim away the content but still keep the essence of the message for each of the lessons. Nobody likes to read extremely long emails so I set a limit of 1,000 words per lesson. That meant that I had to cut out a lot of things I wanted to say. I now get to remedy this by writing a fully fledged book. This is everything I know about web typography and it describes my approach to web design too (as you'll see later on, the two really can't and shouldn't be separated). The coolest thing about it is that I expanded the chapters that people have asked for. I sent a poll to all my

students asking them which chapter they enjoyed the most and wanted to learn more about it and used that feedback for the book. Chapters like Choosing typefaces, Rhythm in typography and Modular scale are now expanded by up to 400%. On top of that, there are three completely new chapters in the book: Page composition, Responsive web typography and Dropcaps. Everything is tied up through an example that gradually gets built with each chapter. This way we put what we learn into practice.

Why this book

The answer is simple. Web typography is rarely considered as a part of the web design and development process. The "development" part in that sentence is essential as web developers aren't really considered to have a role in web typography. That's one of the reasons for writing the course, at first, and now the book. The other one is that people who are normally involved in that process usually take typography for granted. No or very little attention is given to things like choosing typefaces, shaping pages and defining font sizes. Very little attention is paid to the content itself too. As you'll see at the beginning of the book, the content is the main thing that our process needs to revolve around. With this book, I want to challenge all these things. I want to make people realise how important typography is. And lastly, I want everyone involved in the web design process to be able to contribute to better web typography, but they can only do that if they learn about it. That's what it's all about—educating whoever is involved in a web design process.

Who is this book for?

Most of the course students were either web designers, developers or graphic designers. That's who the course was written for but there were others who found value in it—people who studied psychology and don't do web design at all, people who teach web design to high school students, college professors etc. Despite that, the book remains true to its origins—the course. It still is a resource for web designers and web developers. Now there are even more code examples and the book now has Sketch examples too (something I had to remove from the course completely). I hope that makes it even more valuable to those two groups but I'm sure others will find topics discussed in it interesting too.

How to use this book

I recommend reading through it at first, cover to cover. This way you'll get to know the process from start to end and you'll get to build the example website as you go along. That will help you understand each of the chapters better. After that, you should have it somewhere handy so you can always get back to it. Some code examples are included in the book

itself; some are live code examples on CodePen and come as URLs. Those are generally more complex but seeing them in action should help you to understand the topics discussed.

I hope you have a great time reading the book; it's a product of a few months of hard work. Any feedback is greatly appreciated and I'm always happy to discuss anything about web typography or design in general. You're welcome to get in touch at hello@matejlatin.co.uk.

Matej
2 July 2017
Camden, London.

Part I: Macro Typography

Introduction to web typography

“Typography is the art and technique of arranging type to make written language legible, readable, and appealing when displayed.”[1]

If you google “typography”, wanting to learn what it is, you’ll most probably stumble upon the Wikipedia page dedicated to this topic. There, typography is predominantly defined as an art. If you look it up in the *Oxford Learner’s Dictionary*[2] you’ll find “the art or work of preparing books, etc. for printing, especially of designing how text will appear when it is printed” as its definition. The second of the two definitions, associating typography only with print, is definitely outdated.

I want to challenge how typography is defined and ultimately how it is perceived. It’s not something that should be associated with print alone. Not any more. Not in the age where we consume more that 50% of content online, and spend more than six hours a day doing so.[3] I also want to challenge typography being considered solely an art—especially when it comes to the web.

“Perfect typography is more science than an art”[4] is a famous quote attributed to Jan Tschichold—the man who published his revolutionary book *The New Typography* in 1928. He, among other typographers from that era, had a profound impact in the field of typography.

“Typography is more art than engineering—though engineering is certainly a part of it”[5] is what Robert Bringhurst, author of *The Elements of Typographic Style* (recognised by many as the typography bible), believes in, on the other hand.

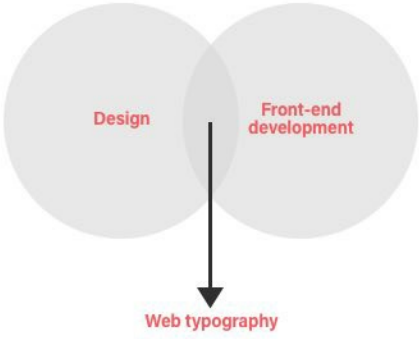
It might seem strange at first, that such influential men in typography could have such contradicting opinions when it comes to defining it. Tschichold believed it was more science/engineering, yet Bringhurst sees it more as an art. Both placed it on the line connecting the two but they placed it at opposite ends.

Different definitions of typography.

Tschichold was a part of the revolution in graphic design in the early 1920s. It was the time of industrialisation. A time when the engineer was seen as the saviour of the world because engineering was in front and centre of whatever was produced. Typesetting as a process was being swept into this new industrialised world. Whatever tools graphic designers used at that time, they needed engineering skills. This probably shaped Tschichold's opinion when it came to defining typography.

Bringhurst has already witnessed the magic of digital design. There are no longer physical parts that need to be assembled in the typesetting process. The age of information reduced that process to a matter of seconds instead of hours. Designers only need the right software and a printer these days. Both are tools they don't produce themselves. So there's very little, or no, engineering in a modern print design process. Bringhurst focused on the print typography in his book which is what may sway his definition of typography more to the side of art.

Web developers should have typography skills



Web typography, along with interactive design, web animation etc., belongs in the grey area between design and front-end development.

I believe that when it comes to the web, typography lies right in the middle. That's why I think both designers *and* developers need to learn the basics of typography. It's expected that designers should have these skills. It's expected that they know how to typeset and when to choose what typeface, how to shape a paragraph and how to use a scale, how to use grids to establish rhythm in their work. I've worked with many designers and found that's rarely the case. Either the designers are self-taught and haven't learned about the importance of typography yet or they studied graphic design and saw it just as a class they had to attend. Developers, don't expect your designers to be proficient in typography.

Whatever gets designed for the web needs to be transformed into code. There is no printer. No magic button that transforms designs into code. Not in a way that could actually get used (at least not at the time of writing this book). People that can do both parts well, design and code, are rare. It takes a lot of time and dedication to really master both sets of skills. This means that even the designers who have good typography skills usually don't have the skills to transform their work into what really counts—code. So it falls on web developers to cover a large part of this grey area: web developers who aren't expected to have any of those skills, web developers who try their best to transform those designs into code, but fail to see small details that make the difference. It's time we recognise the importance of their role in shaping web typography. It's time we stop excluding them from the design process. The design process isn't something designers should own themselves, it's something that other people can take part in and make their own contributions to. Don't simply follow what is drawn by a designer—an approximation of what the website should look like, made with a tool meant for graphic design work. Understand why it's done so. Correct your designer when it doesn't make sense. Work on web typography. Work on typographic details with your designer. Work together. It's your job to produce a typographic style that works—as a team.

That's where this book comes in. And that's why it's written for both web designers and web developers. Its goal is to put web typography right in the middle between art (design) and engineering (development). To cover most of that grey area and invite web developers into the design process. To educate the people who are involved (or the ones who should be) in a web design process to do a better job when it comes to web typography. To start producing a better web typography in order to be able to

contribute to a better web. A web that is readable. A web with content presented in an easily digestible way. A web that informs.

Why is typography important?

Jan Tschichold and Robert Bringhurst may not agree on how to define typography, but they certainly agreed on what its goal is. They just used different words to describe it. “The essence of new typography is clarity”[6] is set in bold in Jan Tschichold’s book. “The new typography is distinguished from the old by the fact that its first objective is to develop its visible form out of the functions of the text… form must be created out of function”[7] Tschichold elaborates.

Bringhurst was more specific: “Typography must invite the reader into text , reveal the tenor and meaning of the text, clarify its structure and order and link the text with other existing elements.”[8]

> *Most people think typography is about fonts. Most designers think typography is about fonts. Typography is more than that, it’s expressing language through type. Placement, composition, type choice.*
> —Mark Boulton

Typography is not about fonts alone. It’s not about type size, margins, spacing, scales or colour. Typography is about organising information in an objective way. Its main focus is information and how we present it to the reader. Size, colours and fonts are just tools that help us shape it. And as we will see, all these tools must work together as a whole. None is independent of the others. Web designers and web developers alike need to take a step back from merely choosing fonts and take a look at the bigger picture that typography actually is.

As a result of the digital revolution, typography is something everyone does (but only few do it well). When a communication tool like that comes into the hands of the masses, rules get forgotten and best practices abandoned. Web started off ignoring the centuries of typography’s evolution. But finally, in the last few years, there’s been an awakening desire for better typography on the web. It’s time to equally acknowledge the importance of typography and the importance of both design and development in shaping it for the web.

Macro and micro typography

Typography is split into macro and micro typography. Macro typography is mainly focused on readability—type size and colour, spacing, layout, hierarchy and rhythm. Micro typography focuses on legibility—things like letterforms themselves, symbols, letter spacing and kerning. This book starts off by exploring macro typography in the first part and continues to look at typography from the micro level in the second part. We'll cover some of the details in typography that can set your work apart from others. It can make a difference between good and great typography.

Font vs typeface

Let's clarify this before we continue. Using the word "font" in a company of typographers will earn you a few odd looks. Some of them might even get offensive and wouldn't hesitate to call you a newbie or even ignorant. "Font" is for a typeface like "mp3" is for a song. And I have never heard anyone say "I really like that mp3"!

I tend to use both words. It simply depends on who I talk to. "Typeface" might be unclear to people who aren't into typography so I'll use "font" in that case. But in this book I'll use the words according to their actual meaning. A "font" will mean "helvetica.ttf" and a typeface will mean "Helvetica" (a family of different weights and styles).

Key takeaways

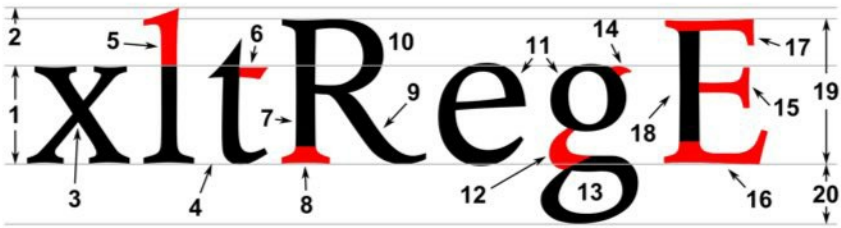
- Web typography lies in the grey area between design and front-end development. Therefore, both web designers and web developers should have typography skills.
- Typography is not merely about fonts. Fonts, or typefaces, are tools through which we put information into a shape, ready to be consumed.
- According to Robert Bringhurst, typography must invite a reader into text, reveal its meaning, clarify its structure and connect it with other surrounding elements.
- Don’t mix up the words font and typeface. Font is a file (for example helvetica.ttf), a typeface is a group of styles (usually separated into fonts: helvetica-italic.ttf, helvetica-bold.ttf etc.).

Anatomy of a typeface

Before we start, we need to get to know our main tool—the typeface. It is at the core of typography and most of our decisions will come back to the different properties of typefaces. Choosing typefaces, combining typefaces and shaping perfect paragraphs will be much easier once we learn a few basics of the anatomy of a typeface.

> *It is not enough to make typeface selections on the base of a name.*
> —Rob Carter

Typefaces come in all shapes and forms. Serif and sans-serif are the two main groups but each consists of numerous styles. Old style, transitional, neoclassical, slab, clarendon and glyphic for serifed typefaces; grotesque, square, humanistic and geometric for the sans serifs. Exploring these reminds me of space exploration. A vast amount of typefaces, each different from the other, each unique, each special. But just like stars and planets, despite their uniqueness, all typefaces have something in common —their anatomy. No matter how special or unique their style is, all typefaces share most of their main features. Looking at typefaces is just like gazing at the night sky and looking at the stars: nothing special if you don't know what you're looking at but very interesting if you know where to look and how to recognise unique features that define the typefaces (or stars).



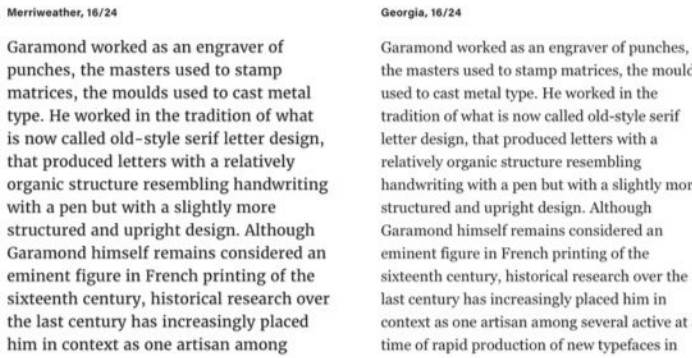
1) x-height; 2) ascender line; 3) apex; 4) baseline; 5) ascender; 6) crossbar; 7) stem; 8) serif; 9) leg; 10) bowl; 11) counter; 12) collar; 13)

loop; 14) ear; 15) tie; 16) horizontal bar; 17) arm; 18) vertical bar; 19) cap height; 20) descender line.

Let’s look at a few main features that will help you gaze at the sky a bit better. We don’t need to know all the details so we’ll cover the basic features. Ascenders and descenders are the most noticeable. They define the shapes of letters and they define the letters themselves. Ascenders are the parts of letters that lie above the x-height line and the descenders are the parts that lie below the baseline. Cap height marks the uppermost part of capital letters. Notice that the ascenders go beyond that line. Also notice how some letters go slightly beyond the baseline and the x-height line. Take a look at the “e” and “g” in the figure above. Type designers do that to make the letters optically balanced. A more rounded letter needs to be a bit larger to seem balanced with the other larger and less round letters.

Type colour and weight

In typography, by type colour we don’t mean actual colour. We mean how heavy a black typeface looks like on a white background. Let’s do a quick test. We have two typefaces: Merriweather and Georgia. Both are beautiful serifed typefaces. Let’s set them to a size of 16 pixels, apply a dark grey colour and put them on a white background. Let’s put them side by side so we can compare them easily.



Paragraphs set in Merriweather and Georgia at 16 pixels and with 24 pixels line height, side by side.

Spend a few moments observing the typefaces and the features we just

went through. If you spend enough time doing this, you'll notice the main difference between the two. Merriweather looks heavier. It looks darker. It even looks larger—we'll come to that soon, let's focus on the colour and weight for now. Let's take a step backwards and look at the whole paragraphs. Both are set in regular weight, at the same size and with the same line height. Yet, they look very different. Georgia has a higher contrast than Merriweather which makes it look lighter. This, combined with other features (height and size of letters), gives Georgia less presence on the page. That's what we mean when we say "type colour" in typography.

Contrast is also a major contributing factor to how "heavy" a typeface looks. Merriweather is a low-contrast typeface, while Baskerville in the picture above is a high-contrast one. Take a closer look at each letter strokes. The difference is obvious.

X-height

We noticed the difference in size of type in the last example. Even though both paragraphs are set at the same size (16 pixels), the two typefaces look very different. Let's take a closer look.

Merriweather seems heavier but also larger than the same-size Georgia.

Let's compare the attributes of both typefaces. Spend a few minutes observing and comparing them. Done? OK, let's see. Let's draw the lines: the baseline, the x-height line, the cap-height line and the line where

descenders end. Let’s do that for Merriweather only as it will be easier to spot the differences.

Merriweather’s x-height is larger than the one of Georgia.

The baseline lies right at the bottom edge of the lowercase x and the x-height line right at the top edge of it, hence the name x-height. I drew the lines long enough so they go all the way to the side where the same word set in Georgia lies. Have you noticed the difference yet? The letters of the same word, set at the same size but set in Georgia don’t touch the x-height line coming from the same word set in Merriweather. The x-height of Georgia is therefore smaller than the one of Merriweather. Take a look at the cap-height line and you’ll notice the same. The height of letters in Georgia, both upper and lowercase, is shorter than the height of the same letters in Merriweather. That’s what makes Merriweather seem larger than Georgia.

We set type sizes on the web by specific dimensions. Whether in pixels, EMs or REMs—we always end up with type set in a particular size in pixels. But these pixels don’t mean a lot in typography. It all depends on the x-height and actual size of a typeface.

Typefaces with a taller x-height are considered modern and easier to read. Merriweather, Roboto and Freight Text are great examples of modern serifed typefaces with a large x-height.

Typefaces with a smaller x-height are considered less modern and classical. Great examples include typefaces like Garamond, Baskerville and Bodoni.

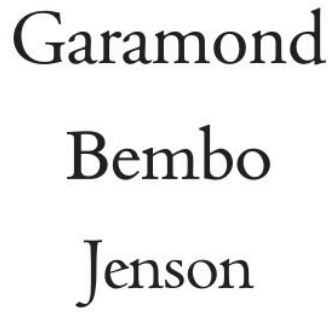
We’ll come back to x-height in the chapter about combining typefaces so keep it in mind.

Typeface classifications

Let’s take a closer look at the major styles of typefaces, starting with serif styles.

Serif styles

Old style
These are usually the first roman typefaces that were designed for print—usually between the 15th and 18th century. Roman, in this case, means that their designs were inspired by the Roman capital letters. First Roman typefaces were cut in 1465 in Italy. Caslon, Jenson, Bembo and Garamond are the best examples of typefaces of this style. Before that, printers would use blackletter typefaces. The Old style typefaces were new and modern in that age. The skills of punch-cutters (people who carved typefaces out of metal) improved with the invention of printing, and the typefaces became more refined. The contrast of strokes is low and the hairlines are heavier than what we would see in high-contrast typefaces. Serifs are bracketed, the axis of curved strokes shifts to the left and the x-height is relatively small.



Transitional
Very much defined by the work of John Baskerville, an English typographer in the 18th century. His work is a representation of the era between the Old style and Neoclassical, hence the name. The serifs are still bracketed but the contrast is slightly higher. The axis of curved strokes is vertical. Baskerville typefaces (there’s numerous interpretations, all based on original designs) are the most well-known transitional serif typefaces. Another commonly unnoticed but great example is Georgia.

Baskerville

Bulmer

Georgia

Neoclassical and didone
Defined by the work of Giambattista Bodoni, an Italian typographer who took typeface designs to another level. The contrast between thick and thin strokes is incredibly high, the axis of curved strokes vertical and no bracketing for serifs. All interpretations of the original Bodoni typeface are great examples of a neoclassical/didone typeface. Can you imagine cutting such thin shapes out of metal?

Bodoni

Marconi

Slab
Slab serifs emerged in the 19th century when they were mostly used in advertising. There's no bracketing on the serifs which are usually very heavy and there's no contrast between the thick and thin strokes. Rockwell and Egyptian Slate are two popular examples of slab serif typefaces.

Roboto Slab

Rockwell

Clarendon
A style of serifed typefaces that became popular in the mid 19th century. Serifs are usually very short and the contrast quite low.

Charter

Clarendon

Sans-serif styles

Grotesque
These were the first sans-serif styles that first surfaced in the late 19th century. There is slight contrast noticeable in these typefaces and there's a certain squareness associated with them. Even the round letters like 'e' and 'o' have obviously squared curves. The axis is vertical. Helvetica, Univers and Akzidenz are typical examples of this style.

Helvetica

Akzidenz

Univers

Geometric
Just like other sans-serif styles, these were popularised by the New Typography movement, defined by the aforementioned Jan Tschichold and other like-minded typographers from the early 20th century. These typefaces are based on the most basic geometric shapes: circle, triangle and square. There is no contrast as the strokes seem to be uniform. These typefaces tend to be less readable then the grotesques. Futura is the best known and (even now, despite being designed in the 1920s) most popular geometric sans-serif typeface. It was the most commonly used typeface on the web in 2015.[9]

Humanistic
Many typography experts claim that this style is the most legible and readable sans-serif style. These styles usually match the design characteristics of the serifed typefaces—the contrast and axis of the curved strokes are noticeable (unlike with other sans-serif styles). Frutiger and Gill Sans are two very common humanistic sans-serif typefaces.

Key takeaways

- By “type colour” we don’t mean actual colour like red, green or blue. In typography “type colour” means how dark a text set in a certain typeface looks like. This is affected by the contrast of a typeface but also by its x-height and style.
- X-height is one of the most important parts to remember about the anatomy of typefaces. It closely matches the bodies of lowercase letters. Typefaces with larger x-height are considered to be modern and are generally easier to read. Smaller x-height is a property usually noticed in classic typefaces like Garamond and Baskerville.
- The two main groups of typeface styles are serif and sans-serif. Serif styles are further divided into Old style, transitional, neo-classical, slab and didone. Sans-serif typeface styles are grotesque, geometric and humanistic.

Choosing typefaces

We are now armed with the information needed to better explore the universe of typefaces. We learned about x-height, the contrast and the colour and how they define a typeface. We learned about different styles of typefaces and we have a basic idea of the era that produced them. It's time we start using that knowledge—by the end of this chapter we'll start creating our example website.

I don't really believe in the mobile first/desktop first mantra. We've come to a point when that shouldn't even be a consideration any more. I think we should start creating websites by firstly looking at what the main purpose of a website is. Not just the particular website we're working on. The real, fundamental purpose. I'm talking about content. Think about it, what is a website without content? It's an empty shell. What is a website with poor content that looks beautiful? It's like a good-looking person without character. What about the other way around—a website with great content that looks poor? It's like an interesting person with a great character that nobody notices because of their "not up to standard" looks.

Content is what will define the website. Most of the time, the content won't be up to you (unless you're working on your own personal website). There's not much you can do about that and that's all right. But when it comes to choosing typefaces, your choice, in the end, should be based on the content. In the age when there's so much content online, it's hard to stand out. But if we present it in an appropriate way, it has a much better chance. Therefore, we'll start creating our example website with the content-first approach. But first, I want to reinforce the fact that typography is not merely about "choosing fonts" (yes I should use the word "typefaces" here but I want to illustrate how others usually understand typography).

Typography is not merely "choosing fonts"

> *In the new computer age the proliferation of typefaces and type manipulations represents a new level of visual pollution threatening our culture. Out of thousands of typefaces, all we need are a few basic ones, and trash the rest.*
> —Massimo Vignelli

One of the greatest visual communicators, Massimo Vignelli, made a

career by using only six typefaces. And he produced a lot of graphic design work in his life. For him, all the many typefaces presented noise and pollution. He saw them as distractions—meaningless trends that come and go. His view was quite radical but I think he had a point. There are so many typefaces out there that it’s hard to know how to separate the good from the bad. And believe me, there are a lot of poor typefaces out there.

Garamond, 1532
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
1234567890

Bodoni, 1788
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
1234567890

Century Expanded, 1900
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
1234567890

Futura, 1930
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
1234567890

Times Roman, 1931
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
1234567890

Helvetica, 1957
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
1234567890

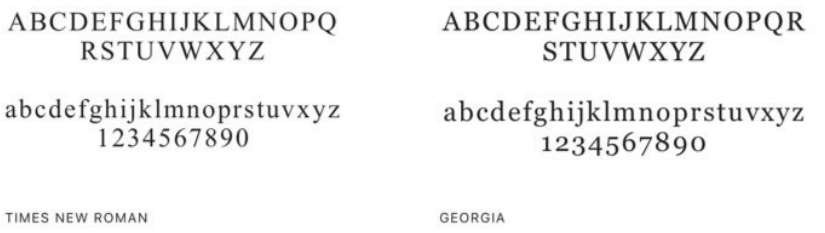
Massimo Vignelli's six typefaces[10]

Choosing a typeface for a website is more often than not a mindless decision based on personal preferences. What looks good to the designer is sadly more important than what the readers expect. There's no connection with the content and so no real relationship with the person reading either. *Typography must invite the reader into text.*[11] I quoted Robert Bringhurst before and I'm quoting him again because I think this is the most important goal of typography. It's not about how many typefaces we have to choose from, or how many times we have used a particular one. It's not about its popularity and it's not about the trends either. Choosing typefaces based on trends will only make sure our websites fade into faceless artefacts without identity.

Typography has many definitions. Depending on whom you ask, each will give you their own, slightly different idea of what typography is and what defines it. But none of the best minds in the typography field ever said that typography is about choosing typefaces alone. Typefaces are merely tools. With them, we give shape to the information we want the reader to consume. So the question isn't what typeface to use. The right question is what typeface to use to best match the content. Therefore, typefaces, or even typography, and information (content) are inseparable. One does not exist without the other.

Web-safe fonts don't suck

The times when we had just a few of typefaces to choose from are far behind us. I remember how I'd always check what fonts are web-safe when I first started doing professional web design work. There weren't many to choose from. Most of the time, I just wanted to get rid of that default look of a website that came with Times New Roman. Soon I started to connect it to whenever something didn't work, or I messed up something in CSS and the browser would fall back to the default. That negative connotation with it stopped me from realising that Times New Roman is a beautiful old style typeface. I don't think I ever used it at all.



Two beautiful typefaces that don't cost a thing (zero dollars, zero kilobytes).

But think about it. Times New Roman and Georgia. Both are nice serifed typefaces that don't affect your website's loading time. Georgia is a transitional serif typeface that was specifically designed for digital use. It was meant to be used on screens. It looks a bit narrower and taller than Times New Roman. A bit more modern. Have you ever used any of these typefaces? Why not? Why not give them a try on your next project?

A website set in Georgia only.

Web-safe fonts should remain an option. Especially when it comes to large-scale websites that need to work on various devices—not just iPhones and fancy MacBooks. What about devices with a poor connection to the internet? 200 kb of a fancy font might be the difference between a returning visitor and a bounce.

Things to consider when choosing typefaces

I tried sorting these by assigning a priority to each, but it was really hard to do. I think these considerations should be taken into account almost equally, but I still managed to sort them by some level of priority.

Goals
What is the goal of your text? Or even, of your website? Is it a place where people come to read medium to long articles? Or is it a website where most people are expected to scan the content, looking for something particular? Are they expected to perform a task and leave afterwards?

The text
This is the most important step everyone skips or even completely ignores. So many web designers design pretty pictures of websites in their perfect shapes. Most of them still use Lorem Ipsum as dummy text. Find out what the website is about instead. What type of content will be published? News? Blog posts? Poems? Essays? Get some content samples. This is important. How will your typeface choice complement the content and the information on the website if you don't read it? At this point you probably thinking: "read it??" Yes, read it. "But I don't have time, I have pages to design and lines of code to write. I have deadlines to hit." Excuses. You'll never get better in choosing typefaces if you ignore the content. What is the author trying to convey? Don't design without content. Don't use dummy text. At worst, use similar content—find something online, maybe even a competitor's website, and use that.

Typeface
What are you after? What is the content of the website about? What typeface would complement that content in a most objective way? What are the people that will read this used to? What do they expect? Will a modern geometric sans-serif seem off? Should we go for a serif that feels modern instead?

Body text or headings?
Are you choosing a typeface for body text or will it be for titles only? Will the two be different? Will one work well with the other? (We'll cover the topic of combining typefaces in Chapter 5.) This is an important question that needs to be addressed early in the process. If the typeface is for body text, you should pay particular attention to the readability of longer texts set in it.

Weights and styles
Does the typeface come with the basic weights and styles? Bold, italic and bold italic? Are there more weights? Some typefaces have numerous weights and styles and are commonly being referred to as workhorse typefaces. Even if they do, use only a few weights. Two or three should suffice.

Font weights

Font weight	Weight name
100	Thin (Hairline)
200	Extra light (Ultra light)
300	Light
400	Normal
500	Medium
600	Semi bold (Demi bold)
700	Bold
800	Extra bold (Ultra bold)
900	Black (Heavy)

Some may only come in one weight or style. I recommend staying away from those. They'll limit you too much and if you still use bold and italic font style with those, you'll end up with the so-called "faux font styles". In short, because there's no bold style for example, the browser simply renders the regular weight as bold. By doing so, it distorts the original typeface design. This should be avoided at all costs.

Font styles

Style	Description
Normal	Normal style.
Italic	Slanted version but with some characters having unique designs.
Oblique	Slanted version of the normal style.

Note that some fonts don't have distinct styles for oblique and italic. If this is the case, browsers are simulating the missing one by using the style that is present. Take a look at the live example at *betterwebtype.com/book/c3e1* for an oblique version of Baskerville and *betterwebtype.com/book/c3e2* for

a proper italic style.

Fonts size in kilobytes
This depends on all considerations mentioned so far. Some typefaces will be heavier than others. Serifs are generally heavier in kilobytes than the sans-serifs. Some bold styles are bolder than others which, again, affects the size in kilobytes. So does how many weights you want to use. I tend to avoid using the bold italic style so I can save a few kilobytes by excluding those.

OpenType features
I consider the OpenType features a bonus. If there are two typefaces that I'm considering and one supports the OpenType features and the other doesn't, I'll go with the one that does. Not all typefaces support these features and even the ones that do, don't always support all of them.

What is OpenType?
OpenType is font format that was introduced by Adobe and Microsoft in 1997 as a successor to the PostScript and TrueType formats (now known as "legacy" formats). The main benefits of this new format are:[12]

- 100% cross-platform compatibility (it works on Mac OS, Windows, Linux and is fully supported by the software industry)
- all font styles are grouped in a single .otf file
- larger character sets (65,536 characters per .otf instead of 210 characters per legacy font files)
- larger character sets leads to extensive language support and
- support for advanced typographic features.

What are the advanced typographic features?
We'll take a close look at most of these in the second part of the book but let's have a quick overview of what these are.

Ligatures
There are three types of ligatures: common, discretionary and contextual. The easiest way to describe them is to say they're more than one character in a single glyph. They serve as an aesthetic enhancement but can also help with legibility and rhythm in typography. We'll take a closer look at ligatures in Chapter 10.

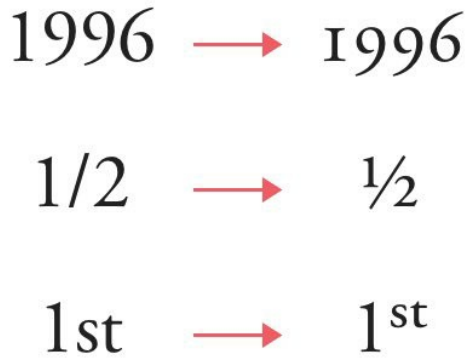
Small caps
Small caps are basically, as the name implies, “smaller capital letters”. They should be used for acronyms, especially the ones used in body text. Skilled typographers use these to enhance the aesthetics of their typographic styles. We’ll take a closer look at small caps in Chapter 11.

Stylistic alternatives and stylistic sets
Some typefaces come with so-called stylistic alternatives or stylistic sets. Both are alternative character designs. The most common are single-storey “a” and “g” characters which look very different from their two-storey counterparts. The “a” letters in the picture below are a great example to see the difference between a one- and two-storey letter style. This simple feature gives designers more options to work with.

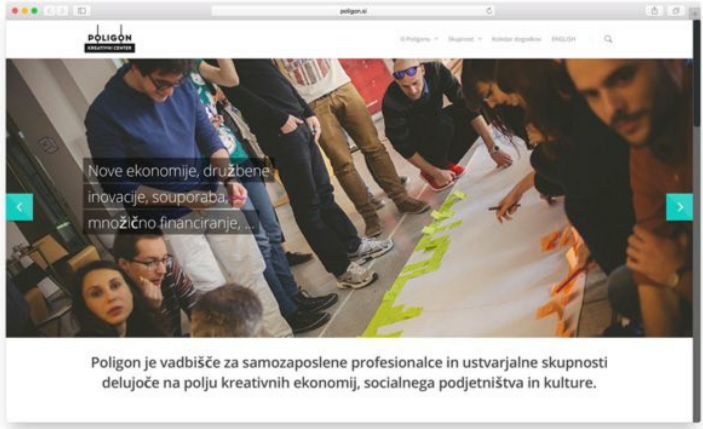
Two-storey “a” and “g” on the left, single-storey on the right.

Figures, fractions and ordinals
As we’ll learn later on in the second part of the book, there are four different styles of figures: lining, old style, proportional and tabular. Each

of them has a different purpose. OpenType features also include fractions and ordinals. Fractions replace the “1/2” and ordinals replace the “1st, 2nd…” with typographically correct glyphs. We’ll take a closer look at figure styles in Chapter 11.



Language support
What language will the website be in? English only? Multiple languages? The Latin language range should suffice for most western languages but here’s a trick. If your website is in a single language only, you can save a few kilobytes by excluding all others. My websites have been English only for a while now, so I always choose that set of characters alone. Sometimes it can save up to 20 kilobytes. If the language that your website is in requires special characters, make sure that the font you want to use has those.



Poligon.si, a Slovenian website, uses a font without the characters “č”, “š” and “ž”, commonly used in Slovenian language. The browser resorts to using web-safe font characters as a replacement, resulting in unappealing and poor typography.

Web fonts

There are two ways to load a web font so we can use it on our website. Let’s take a look.

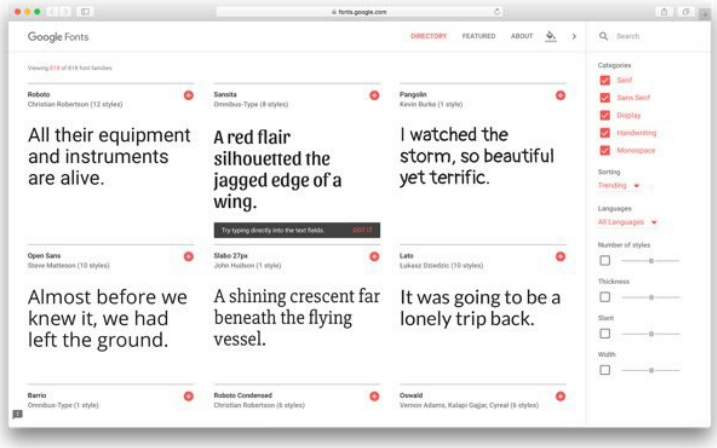
@font-face
This small piece of CSS revolutionised the web a few years ago. Until then, web designers had to rely on web-safe fonts. @font-face changed that completely. Finally, fonts could be uploaded to the server and used on the website through this simple CSS:

```
@font-face {
  font-family: 'Font-name';
  font-style: normal;
  font-weight: 400;
    src:
      url('font-name-regular.woff2') format('woff2'),
      url('font-name-regular.woff') format('woff'),
      url('font-name-regular.ttf') format('truetype');
}
```

Italic and bold styles had to be loaded separately so the code above would have to be repeated for each (with different font style and font weight, of course).

I’m using the past tense when writing about this approach because it’s rarely used these days. It’s now more common to use proper web fonts, from web font providers.

Web font providers
There are many web font providers out there. Google Fonts is probably the most widespread. Offering fonts for free probably helps but it comes with a setback. A lot of the fonts there wouldn’t fall into the “good” category. This doesn’t mean that you can’t still find good fonts to use. Searching the fonts directory is quite limited. The most helpful part is probably just excluding or including certain categories.

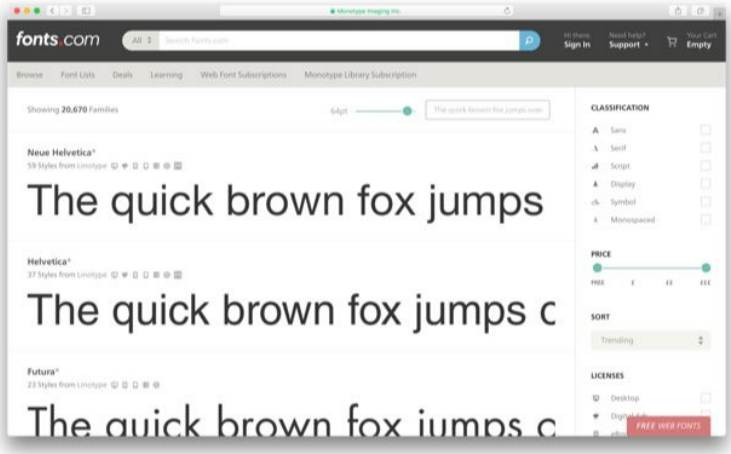


Google Fonts is a popular font directory but with limited filtering capabilities.

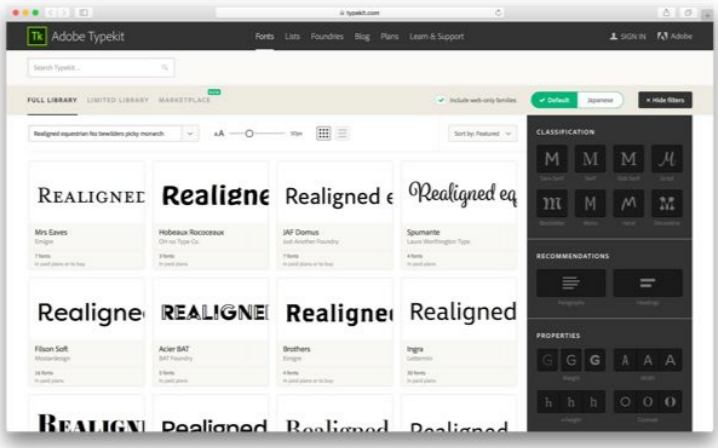
Loading fonts from Google Fonts is simple; it only takes one line of code. What I like about it is the fact that it only needs the following CSS:

```
@import url('https://fonts.googleapis.com/css?family=Robo
```

Fonts.com comes with slightly more sophisticated filters. It allows us to filter fonts by weight, width, x-height and figure styles. TypeKit comes with, in my opinion, the best options to filter fonts, especially for people who are just starting to explore this vast universe. It comes with filters for width, weight, contrast, x-height, figure styles, and a recommendation for either paragraphs or headings. All of them come with the ability to filter by languages supported which should be a deciding factor in the process.



Fonts.com comes with a vast collection of fonts and a better way to filter them than Google.



TypeKit from Adobe comes with a large range of fonts and a very good filtering system.

These web font providers usually depend on JavaScript to load fonts. It's not as clean a solution as the one from Google Fonts, but it comes with a unique advantage: it allows better control of what happens for each of the stages in the web font loading process. We'll explore these in the next section.

```
<script src="//use.typekit.net/XXXXXXX.js"></script>
<script>try{Typekit.load({ async: true });}catch(e){}</sc
```

Issues with loading web fonts

FOUT

There's another problem when it comes to loading custom web fonts. The so-called "flash of unstyled text" or FOUT. It's that second or two when the content of the website is loaded already but the custom web font isn't yet. It's a second or two on a desktop with a broadband connection but it may be a few seconds or more on a mobile device with a 3G connection. Keep this in mind.

To see an example go to betterwebtype.com/book/fout

What is the problem with FOUT? I assume most people just hated seeing the shifting between the fonts on their website. It does look a bit strange. But I think people are used to it nowadays. It's one of the things that happen when a website is being loaded. It must have been the typical designer's chase for perfection that started driving people towards having a problem with it.

FOIT

To avoid the FOUT, some people resorted to hiding the text until the font had loaded—FOIT[13] (flash of invisible text). I used to do this too (in fact, the first Better Web Type website did that), but there's a serious problem with that. Let's say that the font that you want to use doesn't load for some reason. Your website is waiting for it to load and has hidden the text in its anticipation. The font doesn't ever load and the text stays invisible. That sucks. OK, some people came up with a solution for that too. They set a timer on waiting for the font and showed the web-safe font after x seconds. That's just a nasty solution. So much work goes into something that the majority of internet users pay no attention to. So much work to counter the browser's default behaviour. My suggestion is, just let it be. Stick with the FOUT, but make sure you use a serif font as a fallback when you use a serif web font and a sans-serif fallback otherwise.

FOFT

But maybe you're working on a large-scale website that needs to work on all kinds of devices (even the slow ones) and you really want (or must) load a web font. There's a third way of loading fonts. It's called *Flash of Faux Text* (FOFT). Proposed by Zach Leatherman on his blog,[14] it's a way of loading web fonts in phases. The idea is simple: first, we load the font we use for the body text (because that's what dominates the page), which is usually a regular weight font. Then we use that regular font weight

that’s already loaded to produce the so-called faux bold and faux italic (explained in the following section). Once the bold and italic fonts are loaded, we replace the faux counterparts. I think this approach is definitely better than the FOIT one, but I’m still not convinced. Call me a purist, but I don’t like messing with the way browsers work. Once the default browser behaviour changes, the implementation that depended on it needs to be changed as well. For now, I’m sticking with the FOUT.

Faux font styles

I mentioned faux font styles earlier and just now when we were looking at the FOFT approach of loading fonts. There, Zach Leatherman used them as a temporary replacement for font styles that haven’t been loaded yet. Let’s take a closer look at what these are.

Let’s say we found a typeface we want to use for a website. It’s a high-traffic website (millions of views per day, coming from various devices) so the developers are cautious about affecting its load time just because designers want to use a “fancy font”. Because of that, we decide we’ll only use the regular font weight and force the other two through CSS alone (without loading the respective bold and italic fonts). Let’s take a look at what happens.

The example above is represented on the left side of the image and it’s compared to how it should be done (with bold and italic fonts loaded) on the right side. The developers may be happy with almost unaffected performance, but the designers are in the corner, weeping. And rightly so. We can’t allow the bold and italic styles to be improvised by the browser.

Either use the typeface as it was designed or don’t use it at all. Use a web-safe or use system fonts. But don’t allow people who don’t understand typography to butcher typefaces like that. And while you’re at it, use the opportunity to educate them about typography a bit. Send them this book if necessary.

Take a look at the live examples at *betterwebtype.com/book/c3e3* for faux font styles and *betterwebtype.com/book/c3e4* for real font styles.

Text vs display fonts

Back in the day, when the fonts were cut out of metal, punch-cutters had to cut all the characters in a font and repeat the process for each size. On top of that, fonts are split into two size groups: text and display. Text sizes are optimised to work best at smaller, body text sizes. That’s basically where the name comes from. Display fonts on the other hand were optimised for use at larger sizes. This means that a typeface’s proportions and consequently design actually changes depending on the size group. It’s not as simple as merely increasing the size.

The thin parts in serif typefaces are heavier at the text size than they are at the display size. X-height is usually larger in text sizes, the serifs are more obvious and the spacing inside characters is more open. All this helps make the typeface more readable at small sizes.[15]

Notice the difference between the two? Take a look at the letter “e” and the serifs. Bulmer Display looks a bit lighter in general. That’s because it was made for use at larger sizes where more details come to light.

As their names imply, use the different fonts accordingly. Don’t use a display font for body text; use it only for large headings. It’s easier to get away with using a text size for large type on websites but if you have a chance to use a display font and if you can afford it (kilobytes, remember?), definitely use it. It will add another touch of well-crafted

typography to your work. Take a look at the live example at *betterwebtype.com/book/c3e5*.

Print vs web fonts

We're not done dividing fonts into groups yet. There's one more distinction you should know about: print fonts aren't the same as web fonts. In the early days, when @font-face finally became widely supported and popular, I would simply find a print font and convert it to a web font with a tool like Font Squirrel, upload it to the server and load it with CSS. I highly recommend you don't do that any more. Print fonts are optimised, you guessed it, for printing, and web fonts are adapted versions optimised for use on the web.[16] The differences may be small but in typography small details make a big difference. Always try to use a font through a web font provider.

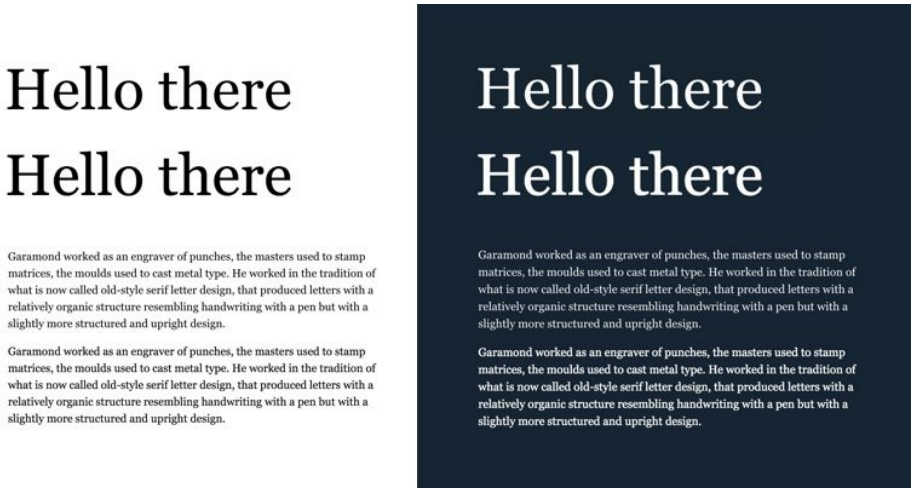
Notice how the web font looks heavier? Web fonts are optimised for better rendering on screens which have a lower resolution than printers.

Font smoothing

A lot of web designers figured out that if they apply font-smoothing: antialiased to their text it looks lighter and softer. They loved it. Now they could finally create something that looks nice in their minimalistic portfolios. But the thing is, it's not meant to be used for dark text on a light background. Dmitry Fadeyev explains it best:[17]

> *Subpixel rendering (default rendering) gives us optimal font clarity for your typical dark text on light background. However, on Mac OS X, when this is reversed and you set light text on a dark background, you get a fairly ugly effect where the text becomes overly bold, spilling out of its lines. Switching away from subpixel rendering to antialiasing for light text on dark backgrounds makes it look lighter, countering the bolding effect.*

Based on that, changing font-smoothing to anti-aliased makes most sense for light text on a dark background (check out the figure below). Don't misuse it just to make the text lighter and softer.



Notice how the top of the two "Hello there" and the top of the two paragraphs look lighter? (It may depend on the quality of your screen if you're able to see such details). The top left paragraph is too light and the bottom right one is too heavy. Font-smoothing makes the top right paragraph and title just right. Check the example at betterwebtype.com/book/c3e6.

Debunking serif vs sans-serif myths

Myth 1: sans-serif is better for the web

Back in the 90s, when the internet was starting to become really popular and widespread, we had lousy 14-inch CRT screens. Their density was 60 pixels per inch (PPI). Font rendering on those screens was extremely poor, and that's why simpler, sans-serif typefaces rendered better. The screens have got a lot better since then. I'm now typing this on a 220 PPI screen. This means that screens now fit almost 4 times more pixels inside an inch. Yes, not everyone has a fancy MacBook Retina screen, but even regular LCD screens have improved immensely.

Sans-serif typefaces may have rendered better on the 90s screens but that's simply not true any more. Recent research shows that there's no major difference between the two and that it's perfectly OK to use a serif typeface on websites.[18]

Myth 2: serif typefaces are easier to read
One more thing before we go on to choosing a typeface for our example website. There's a myth surrounding the serif vs sans-serif typefaces in terms of which one is easier to read in general. Most people believe that serif typefaces easily win this battle. It's a myth that still keeps spreading like wildfire. It also seems to be a myth that people like to believe to be true, so they gladly share their "knowledge" whenever they have an opportunity.

The origin of that myth is bogus research conducted by Cyril Burt in 1955. I say bogus because it was discredited after his death, just like most of his other work. Today, anyone who cites Burt is citing discredited nonsense, basically.[19] There's simply no proof that serif typefaces are easier to read than their sans-serif counterparts. It's a myth that people like to believe in and accept as fact. All it requires is a quick background check to prove it's untrue.

Let's choose a typeface!

All right. That was a lot of information in this chapter. A lot of theory that we just had to get through. Now, let's use everything we learned and finally choose a typeface for our imaginary example website.

Example website brief
We need to design a website for a magazine/blog with an expected traffic of around 100,000 visitors per month who look for up-to-date content that's easy to digest and simple/quick to read. The topics of the website are technology, web design/development and app design/development. Common types of content are articles, reviews, tutorials, interviews and shorter blog posts. The authors of the website believe in quality content and don't rely on excessive advertisement. Still, there will be one ad displayed on the main post page. Everything else should enhance the visitor's reading experience. Authors believe this should rather be done through typography rather than through graphic elements. We'll be building this website from scratch. The name of the website is Tech Geek.

OK, a brief like that is common for a website design project. It's short but already gives a good idea of what the website aims to be. With that, it also gives us a good starting point. Here are my key takeaways from the brief:

- the website needs to feel modern and up to date; its content needs to be easy to read to cater for the busy but curious technology

enthusiast
- the website must be content-focused, with very few but high-quality and well-targeted ads
- as any website with that kind of traffic and audience, it must work across different devices.

Let's go find a typeface for our project

Get a sample of content and read it
As mentioned earlier, content is what should guide your typeface choice. But the website doesn't exist yet in our case. What do we do? Use Lorem Ipsum? No. Find similar content elsewhere and use that. In this particular example, I'll use content from Tech Crunch. I feel it has a similar feel to the content and the way it's presented. Let's go and find an article that will work well as an example and read it (I might actually read a few, just to get a feel of the vibe). Bookmark that article. We'll need it later.

Write down what you're looking for
This will help you remain focused through the laborious process that is finding a typeface. We have our brief and we have an example of content. We should start to get an idea of what kind of typeface we're looking for. Here's what I wrote down for our example:

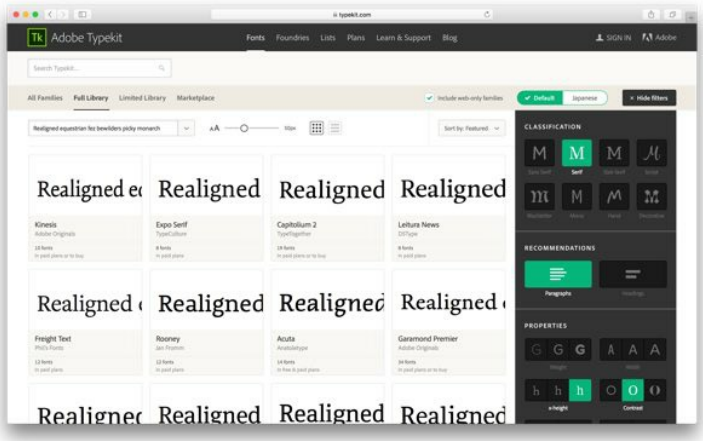
- preferably serif: people are more used to reading significant amount of text content in serif typefaces
- needs to feel modern which translates to vertical axis and larger x-height
- must be easy to read so needs a large x-height, low to regular contrast and needs to work well at the text size
- must have bold and italic styles
- OpenType features like small caps and old-style figures would be a welcome addition
- needs the English subset of characters only
- needs to render well on the common screens (both mobile and desktop).

Think about the big picture
I haven't told you a few things that I have in mind for this example project. I plan to use a sans-serif typeface for the headings. We'll find and combine that typeface with the one we end up choosing in this chapter. We'll look at that later on in Chapter 5. We haven't covered that yet, but I want you to know that it's important to have that big picture idea in mind.

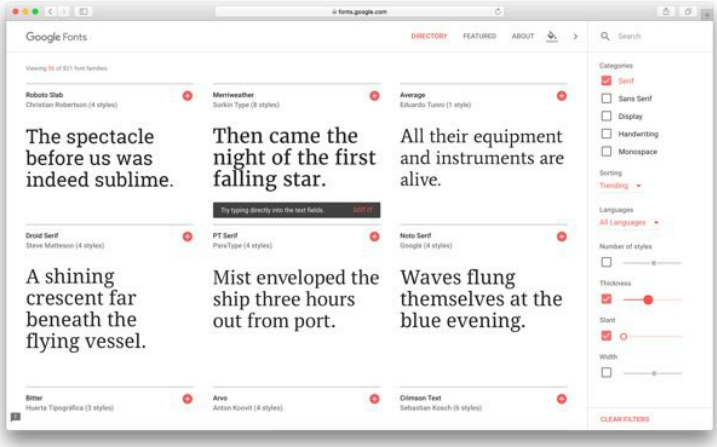
We’re just starting our project content-first—that’s why we’re choosing the typeface for body text as our first step.

Find potential candidates

We now have a good idea of what we’re looking for. Let’s go and try to find it. At this stage I usually go and look for fonts on TypeKit. There, I can filter the fonts down so it only shows me serif fonts with a large x-height, regular contrast and recommended for paragraphs. I do that and I end up with a list of 12 fonts. Out of those, I write the ones I like down. In this case these would be: Expo Serif, Capitolium 2, Rooney, FF Meta Serif, Cordale and FF More.



You could do a similar search on Google Fonts. To do that, exclude all categories but serif, check the thickness slider and change it to one level away from the middle. Then check the slant slider and reduce it to minimum.



Review shortlisted typefaces
This is the most time-consuming step in the process, but it's important we do it right. The key is to review the typeface in context. This means with the actual content and on a website. Website, in this case, meaning actual HTML and CSS in a browser, not an approximation in a graphic design tool.

Don't look at typefaces in Photoshop, Sketch or some other graphic design tool. Your content will not appear there (Photoshop is notorious for its strange font rendering). It will be displayed as part of a website in a web browser. Multiple web browsers, actually. Try your main typeface in different sizes, colours and different browsers. Each has a different way of rendering fonts (Firefox is known to render all fonts a bit heavier than others, so do Macs compared to Windows PCs). Try it on different OS systems as well. Don't just design a fancy-looking visual design of a website on your MacBook Retina and make decisions based on that. Try out what it looks like on Windows and on other screens.

This is important for heavy-traffic, professional websites like the one we're building. You can get away with designing for MacBooks if it's your own personal website. If not, get your example content, put a few paragraphs into an HTML and review it on a few devices (there are Open Device Labs everywhere these days).

What if I can't code?
I prepared a simple tool for the designers reading this book who don't code. It's essential that you preview your typeface in browsers so follow these simple instructions.

1. Open the tool I built in CodePen (*betterwebtype.com/book/font-previewer*),
2. Copy a paragraph from your content and paste it inside the paragraph HTML tags: `<p>Your content goes here</p>`,
3. Repeat step two for at least three paragraphs,
4. Import your fonts:
5. Paste your font import tags into HTML if you're using TypeKit or similar (anywhere in HTML will do),
6. paste the CSS import link if you're using Google fonts.
7. Use the typeface by changing the font-family name in CSS. Make sure you use the correct name as is stated by the font provider ("font-family: garamond-pro, …", not just "font-family: Garamond, …")

That's it! You should be able to see your typeface in a browser now.

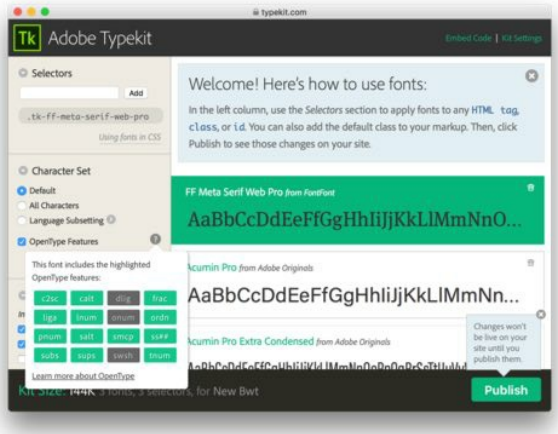
Note: an alternative to this tool is to use the Font Changer plugin[20] *(Chrome only). Find a website with decent typography (I recommend Web Font Specimen*[21]*) and change the font to the one you want to use. The plugin only supports fonts from Google Fonts for now, so you'll be quite limited. But it's still better than looking at fonts in Photoshop.*

Review typeface features

This step comes last because your typeface decision shouldn't be dictated by the support of particular features. Yet they should play a role in that process. Support of OpenType features like small caps and old-style figures (covered in the second part of the book) can have a major impact on typography of a website. A lot of fonts, provided by the web font providers, already have the basic OpenType features enabled. If you're undecided between two typeface choices, take a look at what features they support. Maybe that can help you make the decision.

How do I find the supported OpenType features?

It depends on the web font provider but it's usually not in plain sight. TypeKit only shows these once the font is added to a kit (project) where the features can be disabled or enabled.

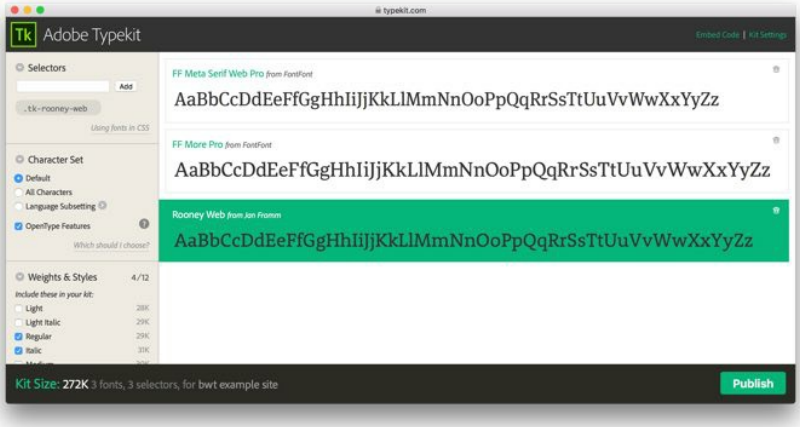


Fonts.com has these listed in the font details page, hidden under the ‘OpenType’ tab.

Choose your typeface
Simply going through this process should help you eliminate some of the typeface candidates. If not, try to narrow the list down to two or three. Then take a look at the things we wrote down in step two. How well does the typeface in question match those? Eliminate the ones that don’t match. Repeat until you’re left with one typeface.

Let’s get back to our example website. I was able to narrow down my list of potential candidates by looking at example content on the TypeKit website. I discarded most of them simply because they didn’t match what I was looking for. I’m now left with three candidates: FF More, FF Meta

and Rooney. I wanted to take a closer look so I added them to a project in TypeKit.



My three candidates for the body text.

I then loaded the fonts in that CodePen tool and switched between the fonts for a while. I added three lines of font-family properties and commented out to switch between them.

```
body {
  font-family: "ff-meta-serif-web-pro";
/*   font-family: "ff-more-web-pro",serif; */
/*   font-family: "rooney-web",serif; */
  font-size: 100%;
  padding: 5% 10%;
}
```

Note: I only recommend doing this for up to three typefaces. It'll take way too much of your time if you do it for every one of your possible candidates.

The next one that I crossed off the list was Rooney. It has a subtle roundness to it; even the serifs are quite soft and rounded, and that didn't match what I was looking for. I then took another close look at the two remaining options. I checked what OpenType features they support. FF More supports old-style figures but doesn't support small caps. If I use any of these two features, I like to use the other one as well. I'll explain why in the chapter about small caps. I wanted to find a typeface that supports old-style figures because I saw that there are lots of numbers used

in the example content. And as we'll learn later on, old-style figures should ideally be used in the body text.

So that was FF More. On the other hand, I had FF Meta. I preferred the overall appearance of FF Meta as I thought it matched the content of the website better, plus it supports old-style figures *and* small caps.

In the end, I decided to go with FF Meta. FF Meta is a large x-height, low-contrast serif typeface. It was designed by Erik Spiekermann in 2007 which reinforces it as a good choice for our modern, technology-related website.

Then I chose the English language subset and removed the bold italic style. I did that to save quite a few kilobytes—25 by limiting the characters subset and another 25 by removing the font style I don't believe should be used (do we really ever need double emphasis—bold *and* italic?). Instead of 114 kilobytes, the fonts now take only 64. That's 44%

less! Check out the example website so far at *betterwebtype.com/book/c3*.

Helpful tools

Here are a few tools that will help you with choosing fonts.

Font Face Ninja[20]
You learned a lot about typefaces already and you'll probably start noticing some of these things when you browse the web. At this time, I'd like to recommend a tool that will help you get better at recognising typefaces. It's called Font Face Ninja and it's a browser plugin. It works with Chrome, Safari and Firefox. Install it, have it in your toolbar and start it up whenever you notice a website with an interesting typeface and cool typography.

Web Font Specimen[21]
Web Font Specimen is a handy resource web designers and web developers can use to see how typefaces will look on the web. I recommend using it in this step to preview how the typeface that you're considering using will render on the web (and not in Photoshop!).

Key takeaways

- Web-safe fonts like Times New Roman and Georgia are

underappreciated and rarely used. They come at zero costs (both in dollars and kilobytes) and are beautifully designed typefaces. They should be considered as an option.

- Things to consider when choosing typefaces: goals (website, project…), the content, is it for body text or for headings, how many weights and styles do we need, font sizes in kilobytes, OpenType features.
- Using web font providers and using proper web fonts is recommended over manually importing fonts (which are possibly intended for print).
- Go with the flash of unstyled text when it comes to loading web fonts. It's something internet users have become used to.
- If available, use a display font for larger sizes, but keep in mind that it will add more kilobytes to your website.
- The myth of sans-serif fonts being more suitable or easier to read on the web is nothing more than that—a myth.

Equilateral triangle of a perfect paragraph

Long before I launched the Better Web Type course I had a talk at the Slovenian CSS Meetup. I didn't know it at the time, but that talk was the basis for the course later on. In it, I wanted to illustrate why developers also need to learn typography and a few basics on how to get better at it. I remember saying: "…if there's one key takeaway that I hope you guys remember from this talk, it's the following—it takes three things to get a perfect paragraph: font size, line height and line length. They need to be balanced." I vividly recall emphasising that a few times during that talk. I didn't have a simple concept to illustrate it at the time.

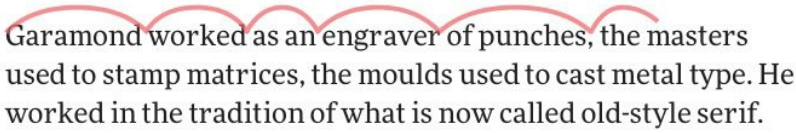
> *The reader should be able to read the message of a text easily and comfortably. This depends to a not inconsiderable extent on the size of the type, the length of the lines and the leading (line-height).*
> —Josef Mueller–Brockmann

While writing the lessons for the Better Web Type course, I was trying to think of a concept that would illustrate exactly that: font size, line height and length of lines need to be in perfect balance. They mustn't be considered and set in isolation. They're interconnected. I struggled for a while but then, out of nothing, it hit me—the equilateral triangle. All three sides in an equilateral triangle are equal and so are all the three angles: a great representation of three things in perfect balance. It turned out that it was easy to apply this idea to the three elements that shape a perfect paragraph. We'll come back to the equilateral triangle concept at the end of this chapter, and before that we'll take a look at each of the properties separately. But even before that, let's take a look at what shapes these guidelines.

How we read

We don't read letter by letter. We only do that when we're children and are still learning how to read. After that we read word by word. The words that we've become familiar with are stored in our brains. So reading is basically recognising shapes of words. Our eyes constantly move left and right in the process of reading, so quickly that we don't even notice. These eye movements are called saccades. They're broken up by fixations which usually come at the end of words. After each line of text our eyes get a longer break as we need time to switch focus to the next line in the text. Make these lines too long and the eyes will get tired quicker. Make the

spacing between the lines too tight and our eyes will get confused over which line to start reading.



Saccades—eye movements—illustrated.

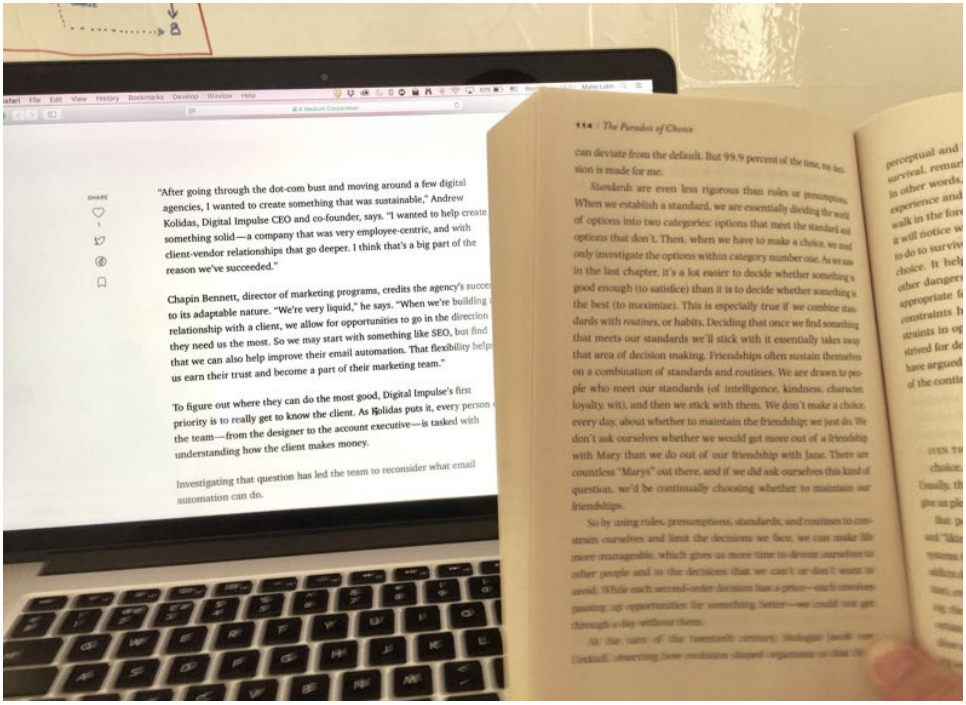
Typography can improve a text's readability by following the basic general principles that have been developed since the first printing machine back in the 15th century. These general principles are adapted to the way we read.

A well-designed and typeset text is easier to read because it takes these considerations into account. This is why type size (and colour), line length and line height are so important. With these we literally shape the reading experience. There are typographers that do this for books and graphic designers that do it for magazines and other printed media. On the web, this task falls somewhere between the web designer and web developer. In rare cases, this would be the same person.

Type size and colour

One of the mistakes that most websites make is that they set the main body text too small. Back in early 2000s, it was common practice to set the body text to a size of around 11 pixels. But at that time the screens were smaller with lower screen resolutions, which means that a font size of 11 pixels back then seemed larger than it does now.

A common rule for setting the body text is to set it to a size that would match the size of the text in a book at an arm's length distance.



Matching the font size with the text size from a book at arm's length.

The recommended size for today's screens is 16 pixels for mobile (usually held closer) and from 18 to 22 pixels for desktop. This also depends on the typeface. As we saw earlier in Chapter 2, some typefaces set at 16 pixels may seem larger than others.

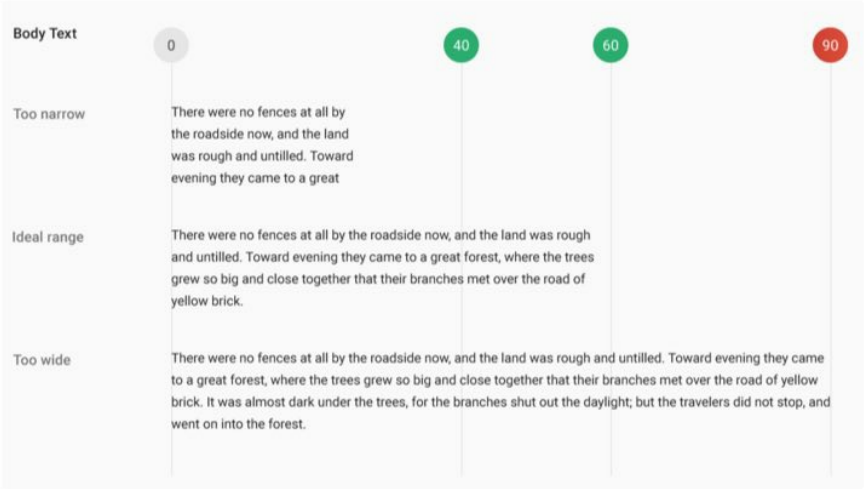
We already learned what type colour means in typography in Chapter 2: Anatomy of a typeface. We took a look at two paragraphs of the same text in different typefaces: Merriweather and Georgia. They were set at the same sizes and matching colour. Still, Merriweather looked larger, heavier and had more presence. That's type colour in typography.

Line length

Reading very long lines of text is tiring for our eyes. Our eyes need constant pauses which are provided by line breaks. In that instance of switching to the next line, our eyes get that short pause. Short, but long enough to keep them going for longer. It's like an engine that doesn't run on full power all the time, so it can keep going longer without overheating.

The ideal width of a line of text is from 45 to 75 characters—including

spaces. Anything that reaches far from that range becomes hard to read: too much line switching when lines are too short and too few breaks for the eyes when they’re too long.



Recommended line width in Google’s Material Design Guidelines[22]

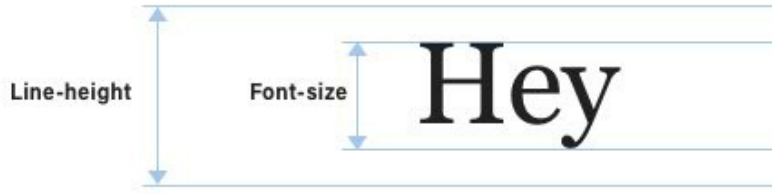
The language consideration

Something we often forget is that not everything out there is written in English. I worked in Luxembourg a few years ago. It’s a small country where three languages are spoken: French, Luxembourgish and German. The company I worked for had German origins so most of the content was in German. Something I noticed, while working with texts in German, was the vastly different amount of white space on the right of paragraphs when the text was aligned to the left. Germans are known to simply join words together to form new ones—producing lots of long words in the process. So I decided to do some research and found that the average German word length is 11.66 characters. Compare that to the average English word length of 8 characters.[23] That’s a difference of three and a half characters per word! Assuming we fit 10 words on a line, that could mean a difference of 35 characters.

Web browsers don’t support word hyphenation yet so all those long words get pushed onto a new line. That’s what’s causing more white space on the right: the paragraph is very ragged on the right. To counter that, I recommend longer lines of text so more words can fit. Try to stay in the recommended range of 45—75 characters when working with languages like that, but lean more towards the 75.

Line height

Line height, or leading as it’s usually called in print, is the main pillar of rhythm in typography. We’ll cover that later in the book. For now, let’s take a look at what affects line height.



Line height works differently on the web. It's evenly distributed below and above the line of text.

Too many designers or web developers that I've met think of line height as an isolated text feature. So they tend to set it based on what seems right. Consequently, line height is something that gets set without too much consideration. But line height is too important to be set so mindlessly. The line length affects the line height. The longer the lines, the more space between them is required. Type size affects the line height. The larger it is, the larger the line height should be as well. Type colour affects it too. Darker and heavier type requires more space between the lines. And in the end, the typeface itself may affect it as well. Some, mostly serifed, typefaces will seem heavier. Those will require more space between the lines.

Now that you know that line height is very important and that it should never be considered as an isolated feature, let's look at general best practices. For paragraphs, ideal line height is usually between 1.3 and 1.6 times the size of the text. So a body text set at 16 pixels should have a line height of 21 to 26 pixels. This will depend on the things mentioned earlier: typeface design, type size, weight, etc.

Georgia, #202020, 16/23

Garamond worked as an engraver of punches, the masters used to stamp matrices, the moulds used to cast metal type. He worked in the tradition of what is now called old-style serif letter design, that produced letters with a relatively organic structure resembling handwriting with a pen but with a slightly more structured and upright design. Although Garamond himself remains considered an eminent figure in French printing of the sixteenth century, historical research over the last century has increasingly placed him in context as one artisan among several active at a time of rapid production of new typefaces in sixteenth-century France, operating within a

Georgia, #646464, 16/21

Garamond worked as an engraver of punches, the masters used to stamp matrices, the moulds used to cast metal type. He worked in the tradition of what is now called old-style serif letter design, that produced letters with a relatively organic structure resembling handwriting with a pen but with a slightly more structured and upright design. Although Garamond himself remains considered an eminent figure in French printing of the sixteenth century, historical research over the last century has increasingly placed him in context as one artisan among several active at a time of rapid production of new typefaces in sixteenth-century France, operating within a pre-existing tradition defined by the work of

Same typeface, same font size, different colour. Darker text should have a larger line height.

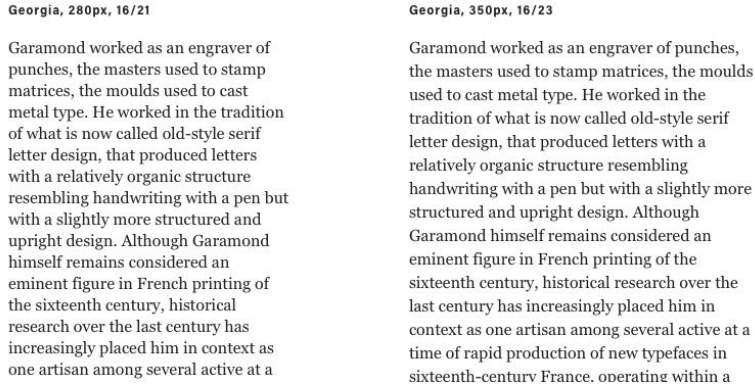
Georgia, 18/26

Garamond worked as an engraver of punches, the masters used to stamp matrices, the moulds used to cast metal type. He worked in the tradition of what is now called old-style serif letter design, that produced letters with a relatively organic structure resembling handwriting with a pen but with a slightly more structured and upright design. Although Garamond himself remains considered an eminent figure in French printing of the sixteenth century, historical research over the last century has increasingly placed

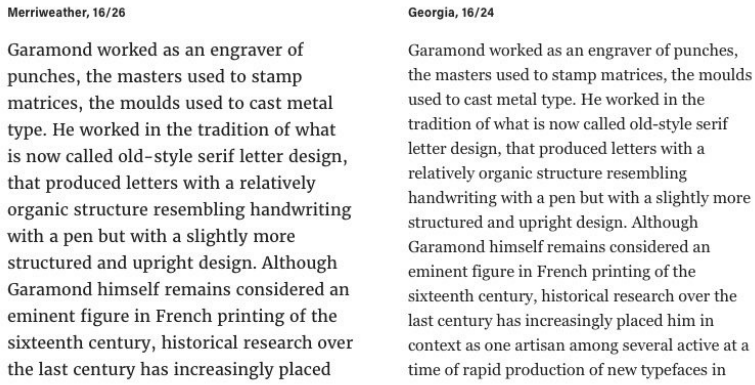
Georgia, 16/23

Garamond worked as an engraver of punches, the masters used to stamp matrices, the moulds used to cast metal type. He worked in the tradition of what is now called old-style serif letter design, that produced letters with a relatively organic structure resembling handwriting with a pen but with a slightly more structured and upright design. Although Garamond himself remains considered an eminent figure in French printing of the sixteenth century, historical research over the last century has increasingly placed him in context as one artisan among several active at a time of rapid production of new typefaces in sixteenth-century France, operating within a

Same typeface, same colour, different font size. Larger text should have a larger line height.



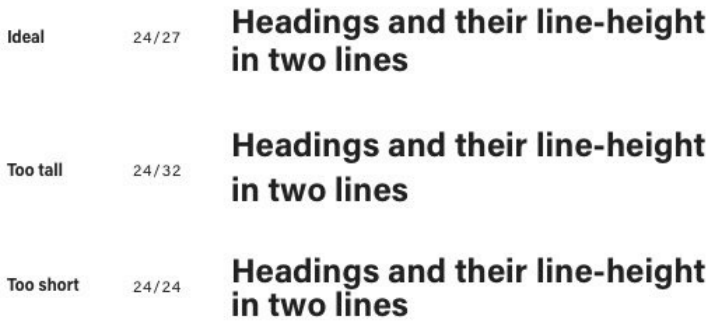
Same typeface, same font size, same colour, different line length. The longer the line of text, the larger the line height required.



Different typefaces, same font size, same colour. Typefaces that seem larger need more line height.

Headings need special attention when it comes to line height
Headings are usually much shorter so they need to have less space between the lines, so it doesn't look like the lines of text are drifting apart. The recommended line height for headings is usually 1 to 1.2 times the size of the text so a heading set at 24 pixels will need a line height of 24 to 29 pixels. This may seem a bit ridiculous at first but try it. Once you see the difference you can't unsee it. And once you get used to it, you'll start

noticing how most websites get this wrong.



Headings (especially larger ones) need a smaller line height than body text.

The language consideration
Similarly to line length, languages have a major role when it comes to setting line height. English only uses the basic Latin characters set. Slavic languages, for example, use characters like “č”, “ć”, “š”, “ž” and “ź” quite a lot. Those characters add more weight to the words and look larger vertically. Line height should be slightly larger to accommodate those characters better.

Some languages, like German for example, use capital letters more extensively. All nouns are capitalised in German. Again, line height should be slightly larger to better accommodate all those capital letters in the lines of text.

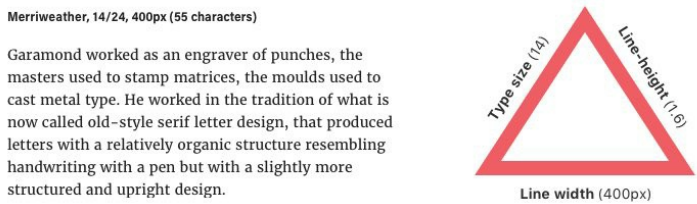
The equilateral triangle

There are some general best practices in typography but they’re never definite. As we saw with the line height, we’re commonly presented with ranges that we should use. What to choose from that range takes time to learn. The eye needs time to get sharper in noticing these differences. That’s why it may seem wrong at first when you set your heading to a smaller line height than the paragraphs, but in time you’ll see that it’s actually correct.

And so we come to the concept I call “the equilateral triangle of a perfect

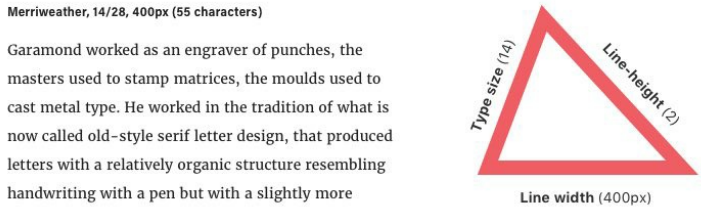
paragraph”. We looked at type size, measure and line height in isolation (as much as we could). By doing so, we already learned that these properties are interconnected. They can’t be considered in isolation and they never should be. That’s why an equilateral triangle is a perfect representation of a well-designed paragraph of text. For it, we need a good type size that matches the line length, which matches the line height. Get one of them wrong and your triangle will get skewed.

Unfortunately I can’t give you definite numbers for a perfect paragraph as there are millions of combinations out there. But I can give you a few examples that will help you train your eye. Try to pay attention to the details and compare the type size, the lengths of the lines and the spacing between them.



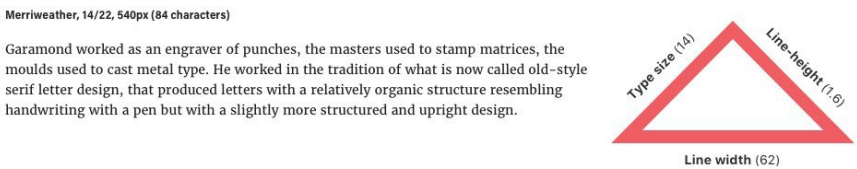
A perfectly balanced paragraph represented by an equilateral triangle.

In the example above, the text is set in Merriweather—an uncommonly large and heavy typeface. That’s why it’s set to 14 pixels. The line height is therefore closer to the upper edge of the recommended range of 1.3 to 1.7. The paragraph above neatly fits 55 characters per line (also in the recommended range).



Lines of text in this paragraph are too long. The triangle isn't equilateral. To fix this we'd need to either make the type size and line height larger or decrease the length of the line.

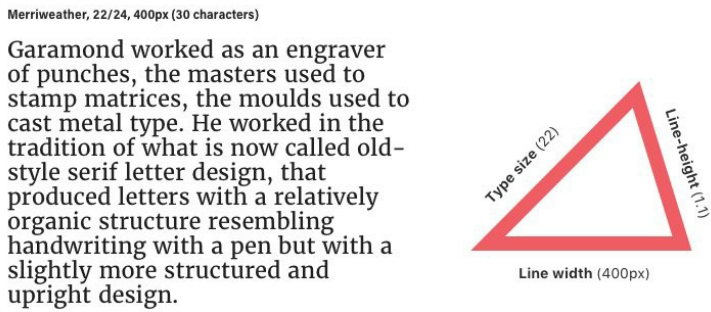
In the example above, the length of the lines is outside the recommended range of 55–75 characters. It's 84 on average, meaning that the bottom side of the triangle expands on both sides. The triangle is not equilateral any more.



The line height of this paragraph is too large. The lines of text are starting to drift apart. This example could be improved if we increased the length of the lines. To properly fix it we'd also need to increase the font size to balance it out. A simpler way to fix it would be to decrease the line height.

The line height in the example above is obviously too large. It's set to two

and very much outside the recommended range. The lines of text are too far apart. Unfortunately this happens too often on the web. It seems like there's a trend of setting type to a light grey colour and applying a large line height on top of it. It's an attempt to get that clean, minimalistic style but it's wrong. Texts like that, especially if long, are hard to read.



The font size is way too large in this paragraph. It forces the length of the lines to 30 characters and completely skews the triangle. The lines of text are too close together. To fix this we'd either have to decrease the size of the text or increase both line height and the length of the lines.

Fortunately, I don't encounter this on the web too often. I still wanted to include it, so we cover all the possible options that can come out of this concept. The font size in the example above is way too large for the given line height and length of the lines. Texts set like this are extremely hard to read.

The perfect paragraph learning game

Many of the students of the Better Web Type course have asked me to elaborate on this concept. And what's the best way to illustrate how this concept works in action? A learning game that will help you sharpen your eyes. Presenting "The equilateral triangle of a perfect paragraph—a web typography learning game".

The game consists of 10 levels, each asking you to set one of the three properties: font size, line height or the length of the lines. At the end, you get a score out of 100 which should give you an idea of how good you are at setting perfect paragraphs.

Head over to *betterwebtype.com/triangle* and give it a try. We'll apply this concept to our example website when you get back.

Let's set a perfect paragraph

We have our paragraphs of what we believe is similar content to what will be used on the website set in FF Meta from the previous chapter. Now let's shape those paragraphs until they're perfect.

I used a font size of 16 pixels for mobile and 18 for desktop so far. I also set the line height to 1.6 and the line length to 35em (line length limit only really applies to desktop). That's a good starting point, but let's see if we can improve it.

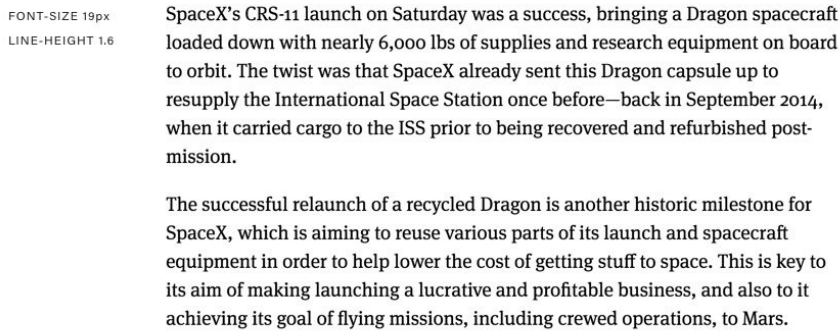
FONT-SIZE 18px
LINE-HEIGHT 1.6

SpaceX's CRS-11 launch on Saturday was a success, bringing a Dragon spacecraft loaded down with nearly 6,000 lbs of supplies and research equipment on board to orbit. The twist was that SpaceX already sent this Dragon capsule up to resupply the International Space Station once before—back in September 2014, when it carried cargo to the ISS prior to being recovered and refurbished post-mission.

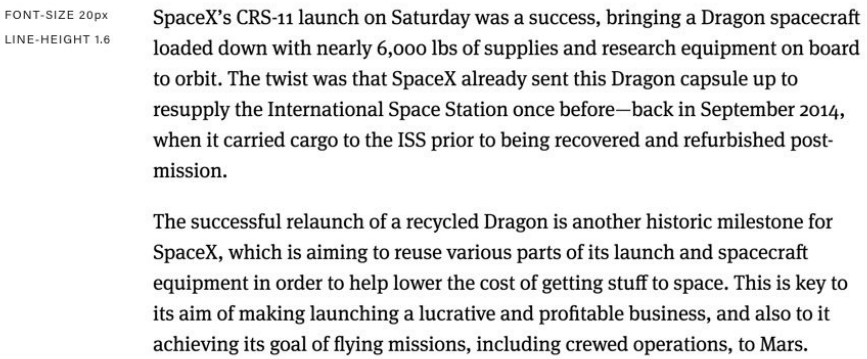
The successful relaunch of a recycled Dragon is another historic milestone for SpaceX, which is aiming to reuse various parts of its launch and spacecraft equipment in order to help lower the cost of getting stuff to space. This is key to its aim of making launching a lucrative and profitable business, and also to it achieving its goal of flying missions, including crewed operations, to Mars.

Font size

Let’s experiment with the font size first. 16 pixels is OK for mobile; let’s take a closer look at what we want to do for desktop. 18 pixels looks like a decent size but we know that the website will be content-focused. Let’s try 19 pixels.



OK, interesting. Let’s try 20, why not?



All right, 20 is clearly too large. It must be either 18 or 19 pixels. The more I compare the two, the more the 18 pixels looks right. Awesome, we now have our base font size. All the future font sizes (headings etc.) will be based on this size.

```
html {
  font-size: 112.5%; // = 18px
  color: #333;
}
```

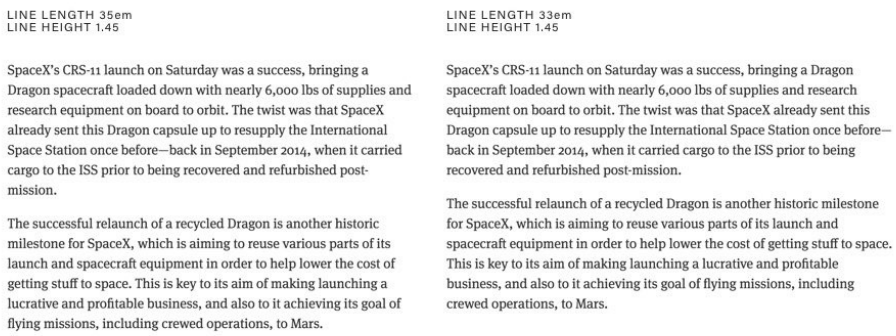
By setting the font size property of the `html` element we tell CSS that this is our base font size. If left unchanged, it would still be 16 pixels (agreed

default among the web browsers).

Note: Did you notice that I set the text colour to #333 in the example above? It's a dark grey colour but you may be wondering why I didn't simply use black (#000). Pure black on a white background has a very high contrast. Too high actually. If you take a look at text printed on paper you'll notice that it's not pure black. It never is because it can't be. It's recommended to use a dark grey instead of black to control that contrast a bit better. #333 is usually my starting point but this will also depend on typeface, its weight and its type colour.

Line height
I'm expecting quite a few acronyms to be used in the content of the website and I already noticed the heavy usage of numbers in the body text. FF Meta supports the old-style figures (lowercase numbers in short) so I need to keep that in mind. A line height of 1.6 seems too large so let's try 1.55. Still a bit too much. What about 1.5? Looks neat, let's go with it for now. This could still change, as I haven't experimented with the line length yet.

Line length
I'm starting off with the line length of 35em which equals to 630 pixels (35 × base font size). As it is, around 75 characters fit on one line of text. Some lines are a bit longer with up to 80 characters. That's still in the recommended range but very close to the edge. Let's try shorter lines.

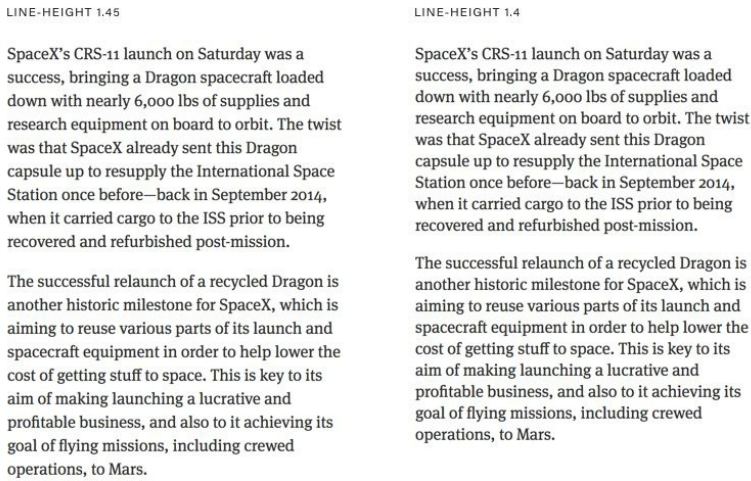


Let's try 30 and 33 ems. Both are good, 30 looks a bit too short, while 33 looks just about right. It limits the longest lines to slightly less than 75 characters which seats neatly in the recommended range. Let's go with 33 then. Ok we changed the line length which means we need to revisit the line-height.

Line height revisited

We changed the line length from 35 to 33 ems, 2 ems less, that's 36 pixels. The change to the line height, if we find out that it's needed, will probably be subtle. I think it should still be slightly smaller than what we started with. 1.45 is just a little smaller than 1.5 but it looks neater.

Because the line length is shorter on mobile screens (even if we use the full width of the screen the text column is still quite narrow) we need to change the line height to match that better. Let's give 1.4 a try.



And yes, it does look better. Most of the time it doesn't need to change that much; small increments do the magic. I've done this so many times that I can make a good guess on what will fit.

That's how the triangle works

As you can see, there's a lot of experimentation in this process. We needed to explore a few options to make sure we're choosing the right one. There's quite a lot of going back and forth as well. Once we changed the line length we needed to revisit the line height. The process isn't simple and it isn't linear. We started off with defaults that were quite close to the final solution, which made it slightly quicker. All the effort was definitely worth it as we now have a perfectly shaped paragraph set in a typeface that matches the content of the website. See the example website so far at *betterwebtype.com/book/c4*. Believe it or not, it gets even more fun after this. Let's find a typeface for our headings that combines well with our body text.

Key takeaways

- We don't read letter by letter, we only do that when we're learning to read. After that, we read word by word as our eyes make thousands of movements across the lines of text.
- Don't make your body text size too small. Use at least 16 pixels on mobile and 18 pixels on desktop screens but keep in mind that typefaces come in different sizes. Remember the example of Merriweather which is larger than usual.
- Recommended length of lines of text is between 45 and 75 characters. Keep in mind the average word length of the language you're using.
- Line height is too important to be set mindlessly. Type colour, typeface, length of lines and even the language should be considered when setting line height.
- Keep in mind the equilateral triangle and how these three things need to be considered together in order to shape a perfect paragraph: line height, line length and font size.

Combining typefaces

Do you really need more typefaces?

Bad typography on the web in the early years of the internet was often blamed on the poor range of typefaces available. But we already learned that typography is not merely about choosing typefaces. One of its goals (probably the main one) is presenting information in a way that is readable and easy to consume. Before we start looking for a typeface to add to our design and combine it with our body text typeface, we need to ask ourselves: do we really need more than one typeface? Don't just add one for the sake of it. There's no rule in typography that says the more typefaces used the better (the contrary is probably more true). There's no guarantee that adding another typeface to your website will improve its typography. I like to keep my typography simple and solid. Like a machine—the fewer movable parts, the less likely it is to break down.

A

SPECIMEN

By JOHN BASKERVILLE

of BIRMINGHAM, *Letter-Founder and Printer.*

To Cneius Plancius.

I Am indebted to you for two letters dated from Cosyra. You congratulate me in one of them on the account you have received, that I fill preserve my former authority in the commonwealth: and with me joy in the other of my late marriage. With respect to the firſt, if to mean well to the interest of my country and to approve that meaning to every friend of its liberties, may be conſidered as maintaining my authority; the account you have heard is certainly true. But if it conſiſts in rendering thoſe ſentiments effectual to the public welfare, or at leaſt in daring freely to ſuppoſe and inforce them; alas! my friend I have not the leaſt ſhadow of authority remaining. The truth of it is, it will be ſufficient honor if I can have ſo much authority over my ſelf as to bear with patience our preſent and impending calamities; a frame of mind not to be acquired without difficulty, when it is conſidered that the preſent war is ſuch, that if one part is ſucceſsful, it will be attended with an infinite effuſion of blood; and if the other, with a total extinction of liberty. It affords me ſome conſolation however under theſe dangers to reflect, that I clearly foreſaw them when I declared how greatly I dreaded our victory, as well as our defeat; as I was perfectly aware of the hazard to which our liberties would be expoſed, by referring our public diſputes to the deciſion of the ſword.

18th century typographic work in a single typeface—Baskerville—by John Baskerville himself.[24]

> *It is a safe rule not to mix different styles of letters on the same page, or different faces of type in the same book.*
> —Eric Gill

Eric Gill, quite a controversial man but a great typographer (among many other things he did), had a similar opinion. There's not much you can do better with more typefaces than you can with a single one. Try to challenge yourself and create a website using one typeface only. If only for experiment and fun, I think everyone should try that as there's lots to

learn from it.

Recently redesigned matejlatin.co.uk. Typography-focused but using a single typeface.

I recently redesigned my personal website. One of the things I wanted to stand out was the minimalism that is such a big part of my life. To best do that, I decided to use a single typeface. I also wanted the website to be typography-focused (literally 95% of it is typography) on top of that, which made the challenge even harder. I was looking for a clean, crisp and simple sans-serif typeface that would emphasise that even more. In the end I chose Gibson from Canada Type.[25] It has a modern, but slightly friendly feel to it. And it works great when set to huge sizes. It was a bit of a challenge to get the typography right for that website, but I managed to do that just using different sizes, weights and only two colours—dark and medium grey.

I included that example in the book, because it's great at showing how much can be done with a single typeface. Not only by typography masters like John Baskerville, but also by self-taught designers like me. Do you think you still need another typeface for your project? No worries, read on.

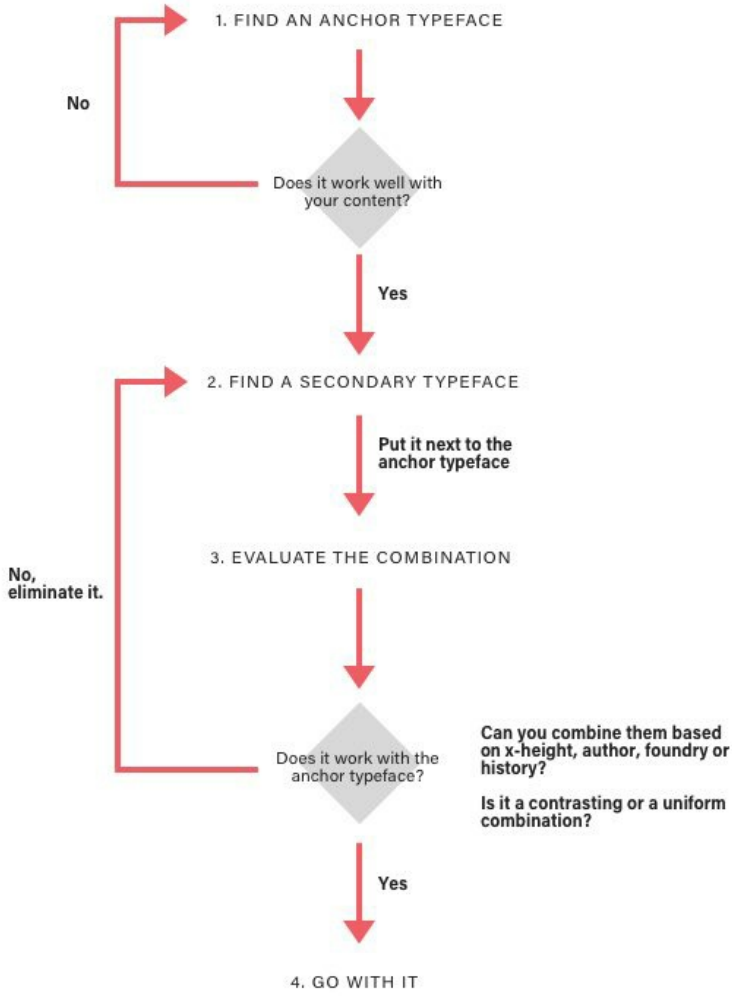
Exploring the typeface combinations

Combining typefaces is probably one of the trickiest parts of typography.

There are no definite rules to follow, which makes it even trickier. There are guidelines and best practices but you'll need to spend more time exploring typefaces and trying out different combinations. I like to believe that in typography, if it's done well, anything goes. As it is with most things when it comes to typography, it's only a matter of how much time you're willing to invest.

If you do believe that a second typeface is needed in your design, and if you're OK with spending some time exploring the universe of typefaces instead of drawing and coding websites, then here's a four-step process that will help you get better at it. It's very much based on Tim Brown's short book called Combining Typefaces, which you can get for free and you should definitely read.[26]

1. Find an anchor typeface for your main body text
2. Find a few secondary typefaces for possible combinations
3. Evaluate the combination
4. Eliminate/choose typeface combination.



1 Find an anchor

At this point you should have your main body typeface—the one you want to use for your body text. Just like we described in the previous chapter, you read the content example, you know what the website is about, you have a sample of the content, you tried previewing it in different web browsers and you decided that this is the typeface you want to go with. We did that at the end of Chapter 3, remember? Now let's find a secondary typeface that will complement it well.

2 Find secondary typefaces for possible combinations
This is probably the most time-consuming task in the process. Maybe even the most time-consuming task in typography. It's time to explore the universe of typefaces. When I start doing that, I'm completely disconnected from the world for a couple of hours. At least. But where do you start? At first you might just add a few typefaces that you like to the list of possible candidates. Maybe it's a typeface you used before. Maybe it's the one you've been saving to use for a project just because you like it that much. Give it a try, put it on the list.

Just like we did when we started looking for our main body typeface, go to your favourite web font provider and start searching. Filter the list to match what you're looking for and browse the results. Whenever you see something you like, something you think fits your grand design, add it to the list of potential candidates. Try not to spend too much time on this. Don't go too deep into exploring each typeface at this stage.

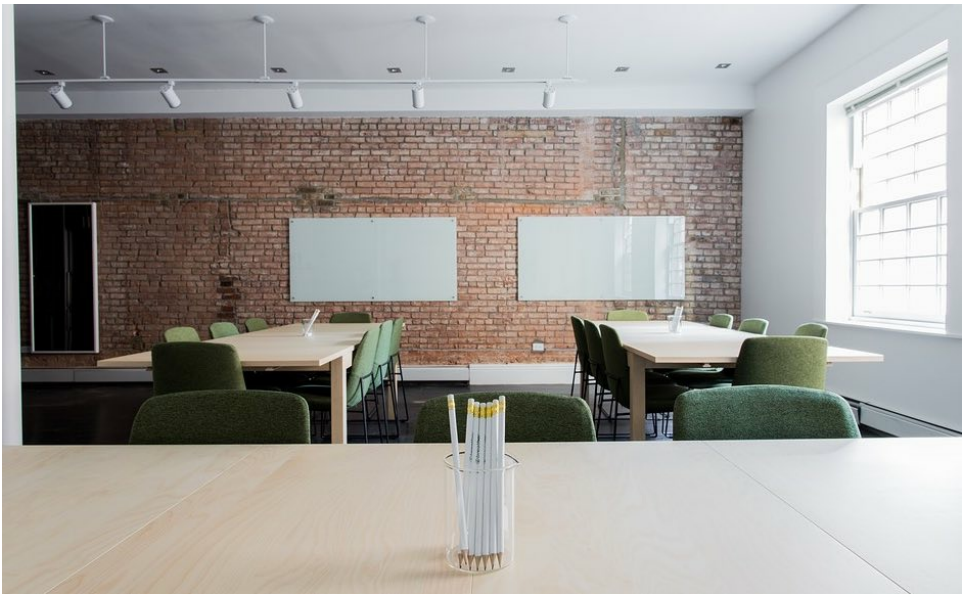
3 Evaluate the combination
Again, I recommend doing this in the browser, possibly in the right context. Do you have an early design of the website? Use that. Do you have a slight idea of what the layout will be like? Make a quick prototype. Don't know how to code? Pair up with your developer. Don't know much about the design and the project? Pair up with your designer. Remember that both roles are important in web typography. If your skills are one-sided, working together is key to producing better web typography.

> *Contrast is one of the most important principles in typographic design. Any shift in typeface, type size, or type weight should be emphasised.*
> —Rob Carter[27]

Contrast and uniformity are the two most basic principles when it comes to design, architecture and art. Both are crucial when it comes to typography as well. Some evaluations described in this process will only work as a contrasting combination, some only as a uniform one and some as both. Let's take a look at a real-life example.

Take a look at the photo of the office below. How would you describe it? It feels modern, right? But there's something else. It feels warm, too. Unusual for offices with modern, almost futuristic furniture. Yet, this one does. The brick wall brings this feeling of warmth and cosiness to a room that would look cold without it. This is a great example of contrast in interior design. Through contrast, the designer was able to achieve perfect

harmony between modern and warm.



An old brick wall and modern furniture in the same room. Harmony through contrast.

How can we apply this to typography? Let’s take the Baskerville typeface for our main body text. Baskerville was designed by John Baskerville in 1754. It was a modern typeface at the time as it was pushing the boundaries of what was possible in typography. It’s perceived as a classic these days though. It evokes feelings of trust and warmth. Based on an experiment, conducted by Errol Morris, it’s the most trusted typeface around.[28] Now let’s complement it with an uppercase-only Futura for headings. Futura was designed by Paul Renner in 1927. It’s a geometric sans-serif typeface. It was a modern geometric sans-serif typeface when it was first designed and it still feels modern to this day.

Harmony through contrast—Futura and Baskerville combined (content adapted from idsgn.org).

Our design looks rich. Luxurious. It has a story to tell. But most importantly—it works. It uses two typeface classics from different time periods. Each comes from the opposite of the scale between modern/cold and classic/warm. Together, they're like fire and ice that turn into flowing water. Harmony through contrast. This brings us to the first type of evaluating typeface combinations: based on history.

Based on history
Evaluating typefaces based on the time period works well with contrast or harmony. You could choose two typefaces that are very far apart or from the same time period. Both should work.

There's almost 200 years separating the creation of Baskerville and Futura.

I find that this approach works best with typefaces that were designed

before the digital revolution. There are just too many typefaces being designed at the moment. Pairing them solely on the fact that they're from the same time period might not be enough.

Based on x-height
Remember when I told you to remember x-height in the chapter about the anatomy of typefaces? This is where it comes handy. Another way to evaluate a typeface combination is to take a close look at their x-heights, to see if they match. They should be very close at worst. Typefaces with similar x-height work better together as they have a similar visual weight. Especially when it comes to vertical rhythm (something we'll cover in the next chapter).

Futura and Baskerville have a similar x-height which reinforces them as a good typeface combination.

This combination works with either contrast or uniformity. Either choose typefaces that have matching x-height or typefaces that have very different x-heights. Don't go with something in between.

Based on foundry
I noticed something through all the typography work I did so far. By choosing typefaces, I tend to keep coming back to a certain few type foundries that produced them. Germany-based FontFont is one of them—I'm a fan of Erik Spiekermann's work. London-based Dalton Maag is another one. There were occasions when I took time to simply explore all the typefaces coming from these two foundries and there were hardly any that I didn't like. It seems like their styles match what I most commonly look for. And that's not a coincidence.

You see, when a type designer starts to work on a new typeface, they rarely start from scratch. It's more common that they start designing a new typeface on top of an existing one. They use the same structure but still come up with a design that is original. But because of that shared underlying structure, the typefaces can usually work well together. Combining typefaces that came from the same foundry is therefore a good bet.

You should only try to match typefaces coming from particular foundries based on uniformity. Sometimes you can find out if a typeface was designed on top of an existing one by reading about its history/origin.

Based on author
Similar to combining typefaces based on the foundry that designed them is combining them based on their author. As mentioned, type foundries commonly use a base underlying structure to produce different typefaces. It's probably even more common for typeface authors to do that. Combining, for example, typefaces that Adrian Frutiger designed is almost a sure bet.

I still recommend putting the two typefaces next to each other and comparing them, just like we did with Apollo and Univers above. Try to establish whether a combination works or not, in a similar way to how we compared typefaces based on x-height, but focus more on the overall proportions. At this point, you need to really study the two typefaces. I told you this takes time, remember?

Again, this combination should only be used as a uniform, not a contrasting one.

Some typefaces are meant to be together
There are the so-called super typefaces. Typefaces that come in both sans-serif and serif styles. Merriweather and Merriweather Sans are great examples. So are Roboto and Roboto Slab. You simply can't go wrong by using the styles of one super family. This can be a great way to start developing your typeface-combining skills.

Some typefaces come in both serif and sans serif. You can't go wrong if you combine these.

On combining similar styles
It's OK to combine a serif typeface with another serif typeface. Or a sans-serif with another sans-serif one. The problem with doing that is, it's hard to do it well. Especially at the start. To do it properly, you have to either find two styles (either serif or sans-serif) that are very similar or two that are very different. The latter is the easier, but I recommend combining different styles when starting to experiment with typeface combinations.

4 Eliminate/choose a combination
Based on the previous step, does the typeface match any of the evaluating factors? If not, you should probably eliminate the potential candidate. Repeat steps 3 and 4 until you're left with only one typeface. Does it match your grand design idea? Do you feel comfortable with it? Does it feel right? Not sure? Come back the next day. I'm not kidding. At this point you probably spent a few hours looking at typefaces and combinations of typefaces. It's time to take a break. You'll get a fresh look on your work by doing so. My personal guideline is: come back tomorrow and if it feels right, go with it; if not, change it.

All of your potential candidates got eliminated? You'll have to start all over. There's still an option to go with a single typeface. It's your choice.

Let's find our typeface combination

Anchor typeface

OK, we learned a lot in this chapter. Let’s put that newly acquired knowledge to use. Following the process described above we must start with the anchor typeface. We have that, as we already decided what we wanted to use in Chapter 3 when we chose FF Meta. Let’s go straight to step 2.

SpaceX’s CRS-11 launch on Saturday was a success, bringing a Dragon spacecraft loaded down with nearly 6,000 lbs of supplies and research equipment on board to orbit. The twist was that SpaceX already sent this Dragon capsule up to resupply the International Space Station once before—back in September 2014, when it carried cargo to the ISS prior to being recovered and refurbished post-mission.

The successful relaunch of a recycled Dragon is another historic milestone for SpaceX, which is aiming to reuse various parts of its launch and spacecraft equipment in order to help lower the cost of getting stuff to space. This is key to its aim of making launching a lucrative and profitable business, and also to it achieving its goal of flying missions, including crewed operations, to Mars.

Finding candidates for possible combinations We’re looking for a typeface we can use for headings. Let’s review the key points we wrote down from the brief when we were looking for the body text typeface:

- the website needs to feel modern and up to date; its content needs to be easy to read to cater for the busy but curious technology enthusiast
- the website must be content-focused, with very few but high-quality and well-targeted ads
- as any website with that kind of traffic and audience, it must work across different devices.

Let’s write down a few key points that will help us find the heading typeface, just like we did for body text:

- definitely sans-serif (we need something that will be of high contrast compared to our body text typeface)
- needs to feel modern which translates to larger x-height
- needs to work well with larger font sizes
- the more weights and styles the better
- OpenType features not required (besides ligatures, I rarely use other

OpenType features with headings)
- needs the English subset of characters only
- needs to render well on the common screens (both mobile and desktop).

With that written down, we're ready to start searching. Let's fire up TypeKit and find a few candidates. Let's set the filters so we can see sans-serif only, recommended for headings, heavy weight and large x-height. The large x-height will match what we're looking for but also the x-height of FF Meta we chose for body text.

I go through the 40 results that came through and shortlist the following: Korolev, FF Enzo, Quatro, Camingodos and JAF Facit. Only five; I guess I have a good idea of what I'm looking for.

Evaluating combinations
Because there are only five we can dive straight in and start experimenting with them. Let's add them to our project in TypeKit and load them up in our example website. Then we copy some of the titles from our example content article and add them in.

```
h1, h2 {
  font-family: "korolev";
/*   font-family: "jaf-facitweb"; */
/*   font-family: "ff-enzo-web"; */
/*   font-family: "quatro"; */
/*   font-family: "camingodos-web"; */
}
```

This will make it easy to switch between the different candidates. At first, let’s evaluate the candidates at their default size and colour. Maybe we can eliminate some of them without digging too deep. Indeed, there’s something that doesn’t feel right about JAF Facit so we can eliminate it. Some characters, like “w” and “y”, in Quatro are quite uniquely designed and they stand out. In a bad way. Let’s cross it off the list. Sometimes it really is that simple. Taking a closer look at FF Enzo reveals it’s clearly humanist style (noticeable contrast and slightly sloped) which is not something we’re looking for. That’s the third one off our list.

Eliminating/choosing a combination
We were able to eliminate three combinations in the evaluating stage, so that leaves us with two possible candidates: Korolev and Camingodos. Let’s take a closer look at these two and experiment further.

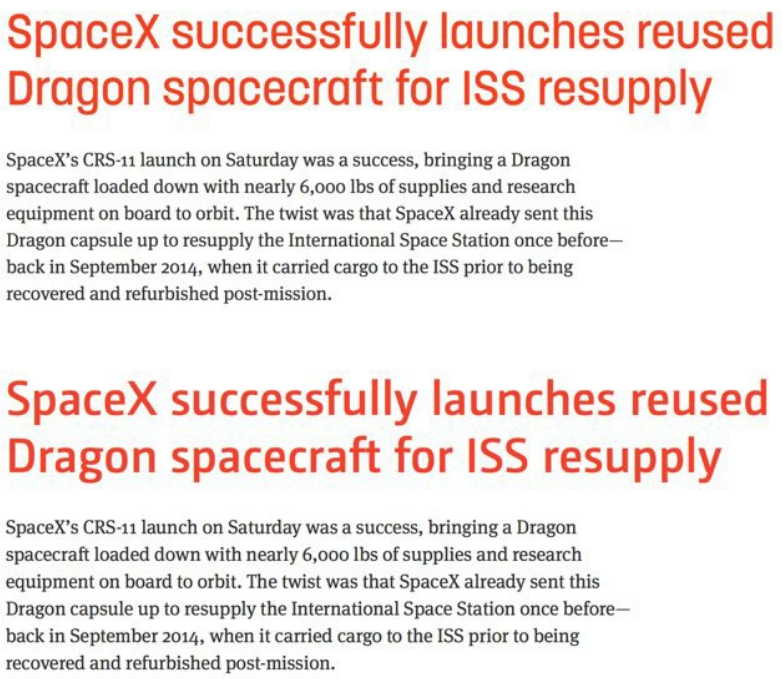

SpaceX successfully launches reused Dragon spacecraft for ISS resupply

SpaceX’s CRS-11 launch on Saturday was a success, bringing a Dragon spacecraft loaded down with nearly 6,000 lbs of supplies and research equipment on board to orbit. The twist was that SpaceX already sent this Dragon capsule up to resupply the International Space Station once before—back in September 2014, when it carried cargo to the ISS prior to being recovered and refurbished post-mission.

Camingodos also has a slight humanist feel to it but it’s not as clear as it was with FF Enzo. The characters are slightly narrower and x-height is large. There’s a certain square feel to it which I’m not sure about. It feels like it’s trying to be tech-y but not going all the way (because of that humanist side). Let’s take a closer look at Korolev.

Korolev feels unique. Very narrow characters, large x-height and it seems like it's slightly condensed. It also has a square feel but not as obvious as Camingodos. Let's try the two in the colour that they'll most likely be used—red.

Let’s try a slightly lighter weight as well—medium instead of bold.



Now that’s interesting. Korolev literally comes to life when slightly lighter. It is now in complete contrast with the body text typeface. Camingodos, on the other hand, looks strange. It lacks identity and with that doesn’t stand up well against the body text typeface. The main problem with it is that it’s on the fence—it’s too similar to the body text typeface to be truly contrasting and yet contrasting to some degree. The thing with contrast in typography is that it needs to be either clear or non-existent. This makes the Korolev a clear winner. Our final result looks like this:

SpaceX successfully launches reused Dragon spacecraft for ISS resupply

SpaceX's CRS-11 launch on Saturday was a success, bringing a Dragon spacecraft loaded down with nearly 6,000 lbs of supplies and research equipment on board to orbit. The twist was that SpaceX already sent this Dragon capsule up to resupply the International Space Station once before—back in September 2014, when it carried cargo to the ISS prior to being recovered and refurbished post-mission.

The successful relaunch of a recycled Dragon is another historic milestone for SpaceX, which is aiming to reuse various parts of its launch and spacecraft equipment in order to help lower the cost of getting stuff to space. This is key to its aim of making launching a lucrative and profitable business, and also to it achieving its goal of flying missions, including crewed operations, to Mars.

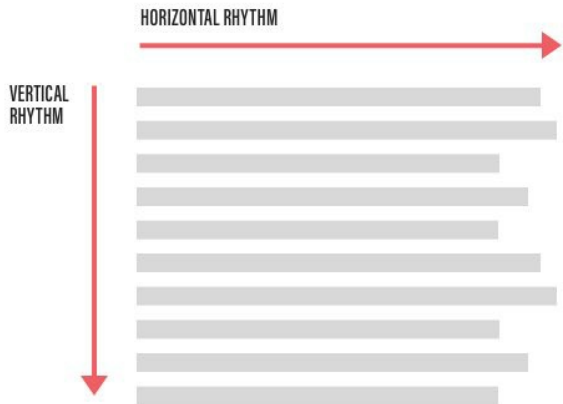
Check it out at *betterwebtype.com/book/c5*.

Key takeaways

- Do you really need another typeface or is one enough? Some of the best typographic work was done using a single typeface only.
- There's a four-step process to finding typeface combinations: find an anchor typeface (usually body text), find candidates for combinations, evaluate the combination, eliminate and choose the second typeface.
- Evaluating typeface combinations can be tricky but keep in mind that they should be combined based on contrast or uniformity.
- Typefaces can be combined based on history, x-height, foundry, author or you can simply use a super typeface—a typeface that comes in both serif and sans-serif style.
- When you first start combining typefaces try combining different styles first: a sans-serif with a serif. Combining typefaces of the same style is harder.

Rhythm in web typography

Rhythm in typography is just like rhythm in music. A text can either flow like a masterpiece symphony performed by an in-tune orchestra, or it can be a disjointed flimsy song by a one-man band wannabe. Just like in music, where order is more pleasurable to our ears than chaos, so is a well-designed text with an established rhythm easier to read and more enjoyable to consume by our eyes. Ears and eyes are just the sensory tools; it's our mind that processes the information. And our mind is a machine for pattern recognition. That's why a well-tuned, rhythmic and proportional text will always triumph over a scrappy one. But, unlike in music, there are two types of rhythm in typography: horizontal and vertical.



Vertical and horizontal rhythm in typography.

The music analogy works very well with typography, because your text will either be easy to read and the reader will get into flow—not focusing on reading but simply consuming the content —or struggle through the information before finally giving up. Horizontal rhythm mostly impacts the legibility, while vertical rhythm impacts the readability of the text and establishes a sense of visual hierarchy.

Horizontal rhythm

Letter spacing (tracking)

Letter spacing is more commonly known as tracking in print design. It can have a massive impact on the legibility of the words, so it should be used with caution. Letter spacing lower case text is not recommended. “A man who would letterspace lower case would steel sheep,” Frederic Goudy used to say.[29] Letter spacing impacts legibility because it makes the words harder to be deciphered by our brain. Reading slows down, even for fast readers. Unless you have a very good reason for doing so, don’t letter space the main body text.

There are two occasions when letter spacing can have a positive impact. Headings tend to be larger and heavier than the body text. Because of their larger size, the spacing between the letters also looks optically larger than at smaller sizes. In this case, it’s a good idea to slightly reduce the letter spacing. We’re talking about 3–5%, not more. This will make your heading a bit more compact, and a bit closer in appearance to the body type.

Applying negative letter spacing to headings makes them more compact and closer in appearance to the body type.

Another occasion where letter spacing can be useful is when applied to small caps or uppercase text only. Using uppercase for longer text is a really bad idea, so this might be best combined with headings again. Whenever we have a line of text set in all uppercase or small caps (we’ll cover small caps in the second part of the book), it’s a good idea to increase the spacing between the letters just a bit. Again, we’re talking about small increases, but just enough to make a difference. My recommendation is from 5 to 10%.

OKAY

ESSAY ON TYPOGRAPHY

No letter-spacing

BETTER

ESSAY ON TYPOGRAPHY

letter-spacing: 0.1em;

Applying letter spacing to uppercase or small caps helps with legibility.

By doing so, we make the uppercase letters and words easier to read and process because the letters are easier to recognise. Besides that, a bit more space between the letters will add a touch of sophistication to our design. Pay attention to well-designed products or brands that use all uppercase in their branding. You'll notice that most of them are letter spaced.

Letter spacing acronyms and long series of digits is also recommended, even in the body text.

Kerning

Spacing between different letters is far from equal. Each letter comes with a default set of spacing around it, no matter the neighbouring letter. That's why we get inconsistencies like this:

Bad kerning, also known as keming. In this particular case it's so bad that the word "SAVE" seems to be two words "SA" and "VE".

Kerning—altering the spaces between particular pairs of letters—can resolve these issues. The result is a much better proportioned word and optical perfection. Kerning, unlike letter spacing, changes the spacing around letters depending on their neighbouring letters.

Fixing the bad kerning.

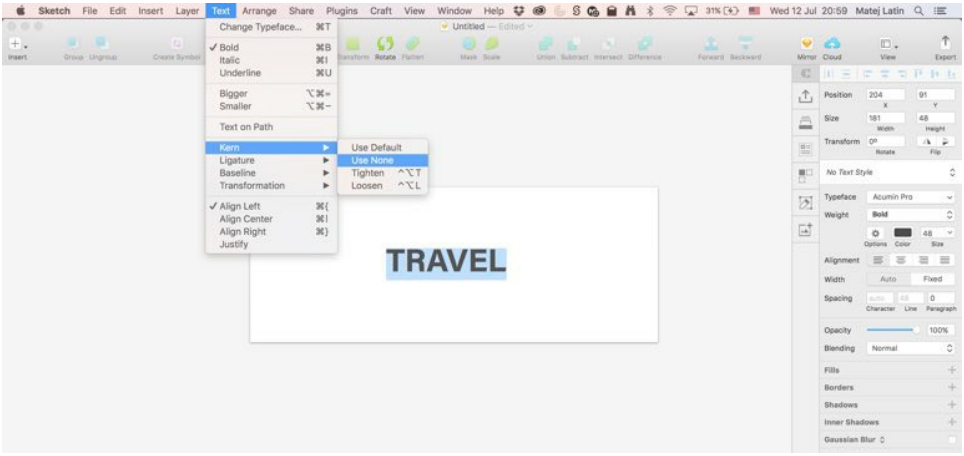
Most web browsers default kerning to auto. This means that the kerning will be enabled for larger type and disabled for smaller. Bad kerning is not as obvious on small type. If you wish, you can control it like this:

```
font-kerning: auto; // default
font-kerning: normal; // enables kerning
font-kerning: none; // disables kerning
```

That's about it when it comes to what we can do with the default browser support right now. This is probably not good enough for special occasions when we need to move a particular letter by x pixels to achieve that optical perfection. Thankfully, there are tools like Lettering.js.[30] With it, we can control the positioning (and also style) of each letter.

Kerning in Sketch
Sketch comes with default kerning enabled but you can change it or disable it completely if needed. To do that, you need to select the text (actual text, not the text box) and go to *Type* > *Kern* > *Tighten* if you want tighter kerning and *Type* > *Kern* > *Loosen* if you want looser kerning. Choose the default option to go back to normal or the disable option to

completely disable it.



Don’t justify on the web

A break in music has a meaning. It separates the sound from silence. Nothingness from a rich sound of a chord. It seemingly breaks the rhythm (even though breaks in music always match the rhythm). We get the same effect in typography. A combination of letters, words and empty spaces define the rhythm. For reading to flow, that rhythm needs to be consistent. And because (as we learned earlier) we read word by word, too much spacing between words breaks this rhythm. It breaks the flow of reading. It turns the easiest text to read into something that is hard to consume, no matter the language or words used. I still encounter this far too often on the web:

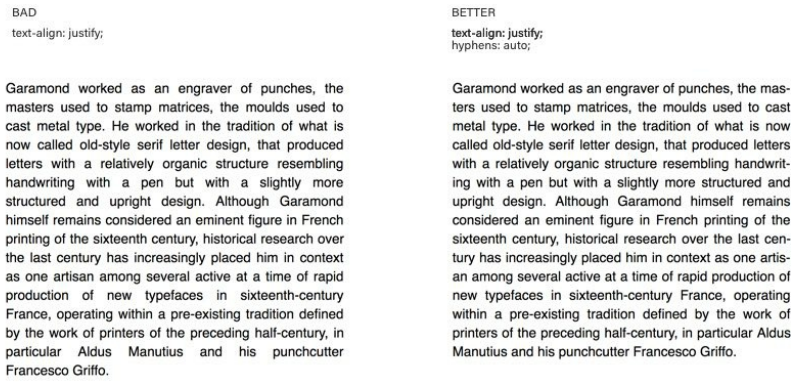
GOOD
text-align: left;

Garamond worked as an engraver of punches, the masters used to stamp matrices, the moulds used to cast metal type. He worked in the tradition of what is now called old-style serif letter design, that produced letters with a relatively organic structure resembling handwriting with a pen but with a slightly more structured and upright design. Although Garamond himself remains considered an eminent figure in French printing of the sixteenth century, historical research over the last century has increasingly placed him in context as one artisan among several active at a time of rapid production of new typefaces in sixteenth-century France, operating within a pre-existing tradition defined by the work of printers of the preceding half-century, in particular Aldus Manutius and his punchcutter Francesco Griffo.

BAD
text-align: justify;

Garamond worked as an engraver of punches, the masters used to stamp matrices, the moulds used to cast metal type. He worked in the tradition of what is now called old-style serif letter design, that produced letters with a relatively organic structure resembling handwriting with a pen but with a slightly more structured and upright design. Although Garamond himself remains considered an eminent figure in French printing of the sixteenth century, historical research over the last century has increasingly placed him in context as one artisan among several active at a time of rapid production of new typefaces in sixteenth-century France, operating within a pre-existing tradition defined by the work of printers of the preceding half-century, in particular Aldus Manutius and his punchcutter Francesco Griffo.

Comparing left-aligned and justified text on the web (no hyphenation).



Comparing two justified paragraphs: one hyphenated, one not.

Web browsers render justified text very poorly. They don't have the necessary algorithms to properly set the spacing between words and even individual letters (which is what proper text editors do). That's why justified texts come with patches of space between the words: rivers of white between the black of text. This makes it very hard to read so justifying text on the web should be avoided at all costs. Web browsers are getting better in supporting hyphenation, though. If you do use justified text, complement it with hyphenation. For now, I recommend not using justified alignment at all.

Justified alignment and accessibility
Benedicte, a student from the Better Web Type course sent me an email recently. He told me that he works for an organisation in Norway that specialises in books for readers with different special needs. He pointed me to the fact that people with dyslexia have a particular problem with reading justified aligned texts. It's not clear where a line of text ends which makes it very easy to switch to the wrong line. I did a bit of research and even Gov.uk (a UK public sector information website) recommends aligning text to the left.[31]

Paragraph indenting
A common way to visually separate paragraphs in books is to indent the first line. It's actually more common than putting an empty line between them. In terms of rhythm, we're changing the horizontal rhythm to separate paragraphs instead of the vertical one. The contrary is true on the web—paragraphs are more commonly spaced apart but indenting the first line is quite a simple thing to do. There are two rules that *must* be

followed:

1. *Don’t indent the first line of the first paragraph or of the paragraph that comes after a heading, an image or any other type of figure.* I know it sounds quite complicated but it’s actually very simple when it comes to CSS.

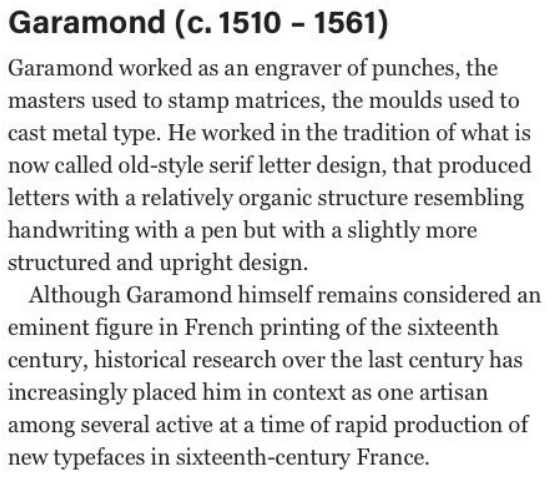
```
p + p {
  text-indent: 1em;
}
```

This works because the text-indent property will only be applied to paragraphs that are preceded by another paragraph. If a paragraph is preceded by an h1, for example, the text indent won’t be applied. That’s exactly what we want.

2. *Don’t put a bottom margin on your paragraphs.* They’re visually divided by the indented line and that’s enough. There’s no point in having space between them. Indented and spaced-apart paragraphs make skilled typographers cringe.

```
p {
  margin: 0 auto;
}
```

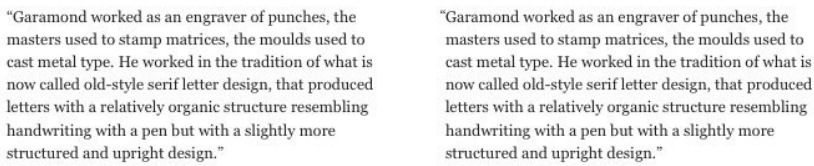
This will set the top and bottom margins of all paragraphs to 0, just to make sure that a top margin doesn’t put a blank space between the paragraphs.



I recommend doing this for texts that aren't broken down into different sections, divided by titles or a lot of images. It simply works best with longer texts divided into paragraphs. Long articles or web books are best-use cases for paragraph indentation.

How much indentation?
The recommended and most common paragraph text indent is 1em, just like we set in our example above. It's also common to set it to one unit of line height (more on line height being a unit in the upcoming vertical rhythm section). So if we had a paragraph with a font size of 18 pixels and a line height of 27 pixels, we would set the paragraph indent to 27 pixels, or 1.5em (equalling the line height of 1.5). Half an em is considered the minimum for paragraph indentation and 3em the maximum. My recommendation is either 1em or equal to one unit of line height.

Hanging punctuation
Hanging punctuation is something 9 out of 10 websites get wrong. I'm sure that once I tell you about it, you'll start noticing it everywhere. By its definition, hanging punctuation "is a way of typesetting punctuation marks and bullet points, most commonly quotation marks and hyphens, so that they do not disrupt the 'flow' of a body of text or 'break' the margin of alignment".[32] Let's take a look at a few examples.



Can you notice the difference in the image above? It's jarring to me and it really hurts to see the example on the left so often. The quotation marks at the very beginning of the paragraph must sit outside the main block of text so it doesn't disrupt the flow, or the rhythm, of it. It's a small detail, I know, but it can have a major impact on the overall look of your typography if done right. So how do we do it right? That's where it gets a bit complicated. There is a CSS property for hanging punctuation but it's only well supported by Safari at the time of writing this.

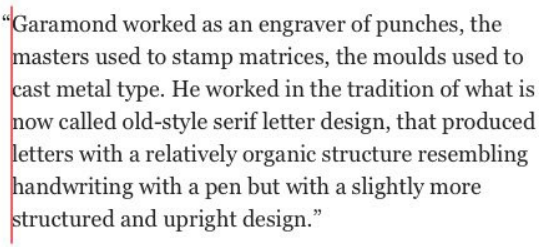
```
blockquote p {
  hanging-punctuation: first;
}
```

Setting the hanging-punctuation property to "first" means that "*An available character at the start of the first formatted line of an element hangs*".[33] In our case this means the quotation mark at the start of the first line of our paragraph hangs. The "available character" part simply means that the quotation mark is on the list of characters that CSS considers as the ones that can hang.

That's great, but because of the poor browser support, we can't use this at the time of writing this book. Too bad, that would make it so easy. Well, as it turns out, there's a workaround that's equally simple: negative text indent.

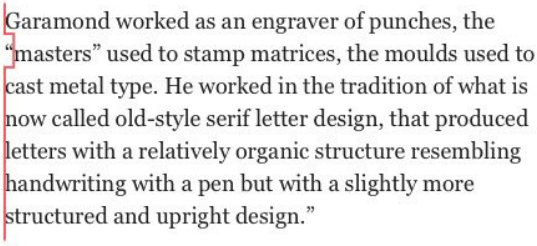
```
blockquote p {
  text-indent: -0.5em;
}
```

Here's what we get:



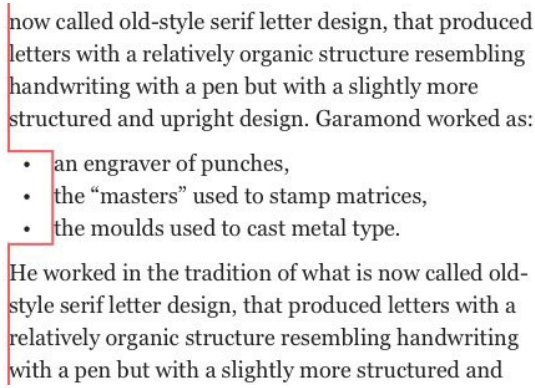
Just make sure you change its value so it works with the font you’re using. -0.5em should be pretty much there but make sure you change it if needed, so it’s spot on. Take a look at the live example at *betterwebtype.com/book/c6e1*.

There’s a problem with both of the two solutions I haven’t told you about yet. Well, actually, both solutions work considering the quotation marks only come in the first line. What if we were to quote someone in the middle of a sentence and that quotation happens to start on a new line?



None of the two solutions work in this example. The quotation mark is pushed into the main body of the text. It’s aesthetically unappealing but there’s not much we can do about it. At least not yet.

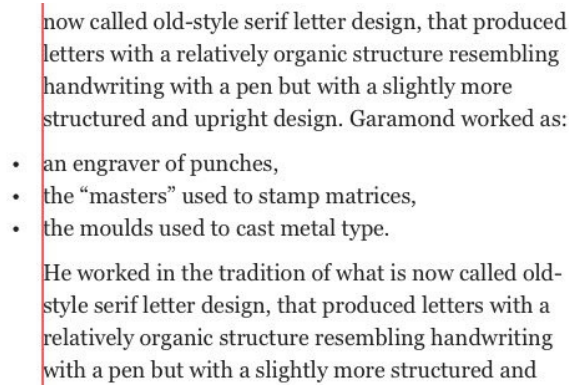
Hanging punctuation with other characters
Hanging punctuation should be applied to other characters as well. After the quotation marks, the next most important are bullets. Again, most websites get this wrong but, actually, browsers get this wrong by default as well. This is how most browsers will render ordered and unordered lists by default.



In fact, the bullets should be hanging so they don’t disrupt the flow of the text. To keep the horizontal rhythm undisrupted we need to change the padding of the unordered and ordered list elements.

```
ul, ol {
  padding-left: 0;
  list-style-position: outside;
}
```

Note: Also make sure that list-style-position is set to outside.

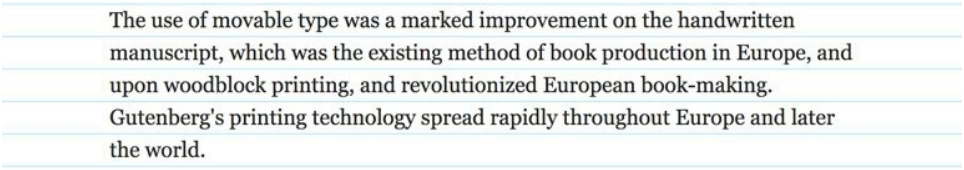


This will push the bullets outside and keep the horizontal rhythm unaffected. Having the bullet points inside the main block of text is not as big a crime as having the quotation marks inside. It is typographically correct but I know that it looks strange to some people, at least at first. They don’t mind it once they get used to it. We, as a web design community, have been making mistakes like this for so long that these things need to be re-hardwired. I, personally, see the hanging bullets as a recommendation, but hanging quotation marks as a rule. Take a look at a live example of the list at *betterwebtype.com/book/c6e2*.

Vertical rhythm

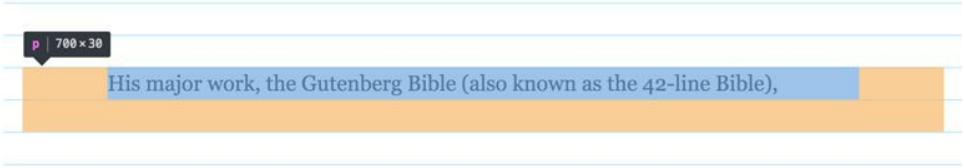
Let’s say that a website has the main body text set at 20 pixels and a line height of 30 pixels. The line length should be appropriate for this size and line height: 600 pixels looks just about right. We now have all we need to set up the vertical rhythm throughout the page. To do that, we need a basic rhythmical unit. In typography, that’s line height. We can see it by adding a 30-pixels-tall baseline grid to our website.

```
p {
  font-size: 20px;
  line-height: 30px;
  max-width: 600px;
}
```



Baseline grid indicates equal line height and vertical rhythm.

Note: Unlike in print and graphic design, the baseline grid lies right in the middle of the lines. Lots of people ask me if it shouldn’t lie right at the bottom of the bodies of letters. Not on the web. Take a look at how web browsers interpret a line of text:



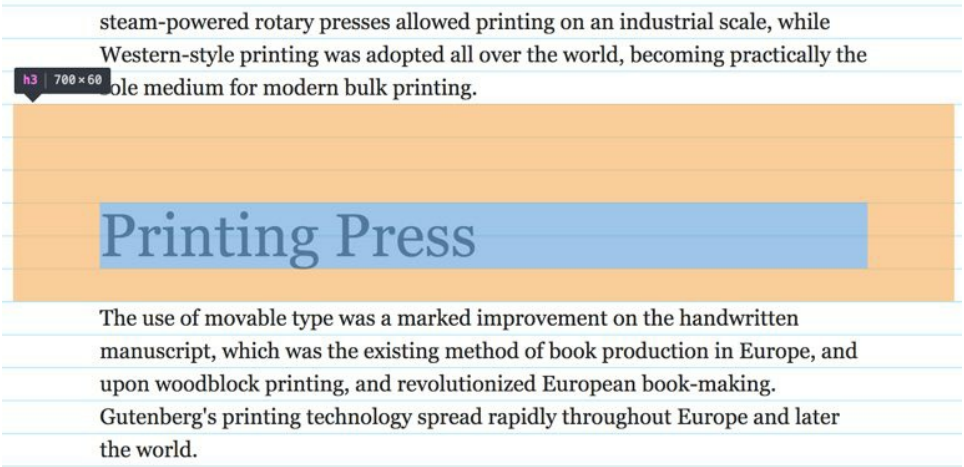
A baseline grid on the web falls right in the middle between the line, unlike in print where letter bodies lie directly on it.

We only have a paragraph of text for now, so everything looks right. To

keep this rhythm going, we need to use the line height as a base unit for every size, margin and padding on the site. Let's see. We want to add a title to our text. We assign a size of 55 pixels to it, to make it stand out. Its line height now needs to be an even multiple of the original (base) line height, our main rhythmic unit. This also applies to its margins—especially the top and bottom ones.

```
h3 {
  font-size: 55px;
  line-height: 60px; // = 2 × 30px (main body text line-h
  margin-top: 90px; // = 3 × 30px
  margin-bottom: 30px; // = 1 × 30px
}
```

Note: I'm using pixels for these examples but you should be using units like em, rem or just a decimal 1.5 or a percentage (150%) value for line height.



Heading 3's line height equals two lines, its margins equal three lines on top and one line at the bottom. Everything falls into place.

We assigned a line height of 60 pixels because it's the next multiple of our base line height (30 pixels) that comfortably accommodates the title set at 55 pixels. A guideline for heading margins that I like to stick with is that the bottom margin should be noticeably smaller than the top one. A lot of websites make the mistake of having equal top and bottom margins for headings, so they float right in the middle between the blocks of text. The

title needs to visually connect with the text beneath it, the text it's actually referring to. And that's exactly what we want to achieve with vertical rhythm and visual hierarchy. A reader can now understand the structure of the text by simply scanning it.

Because we'll need this in most cases it's best practice to assign default line height, margins and paddings to all elements and deviate from it only when necessary.

```
* {
  line-height: 30px;
  margin-top: 0;
  margin-bottom: 30px; // = 1 × 30px
}
```

So if you want your lists to have a specific bottom margin, you'd go for something like this:

```
ul, ol {
  margin-bottom: 60px; // = 2 × 30px
}
```

You may be questioning what happens when an image breaks the vertical rhythm. It's a good question. Images come in various sizes. It's impossible to expect that we'll be able to control their height on every occasion, especially for large-scale websites. What do we do then? My suggestion: let it be. Use the baseline grid as a guide, not as a restraint. Your text and your page proportions are still rhythmically correct. It still has an established visual hierarchy. Large solid blocks of content, which images certainly are, don't visually break the rhythm apart. It may break your grid, yes, but at the end of the day it doesn't matter. The grid is just a tool.

Vertical rhythm with Sass

Last year, I built a tool revolving around vertical rhythm and modular scales in typography. I called it Gutenberg—a meaningful web typography starter kit.[34] Sass was the one thing that made it so much easier. In fact, I don't think it could have been built without it. Its main goal is that a developer sets a few basic sizes (font size, line height and max width) and Gutenberg takes care of the proportions and vertical rhythm. I now use it for every website I build. Here's the main thing that Sass made so much easier:

```
$base-font-size: 112.5; // Is used as %
$line-height: 1.5;
```

```
$base: 16 * ($base-font-size / 100);
$leading: $base * $line-height;

@mixin margin-bottom($number){
  margin-bottom: #{$number * $leading + 'px'};
  margin-bottom: #{$number * $line-height + 'rem'};
}
```

This is the mixin I used a lot. Gutenberg has similar mixins for other margins and padding as well. Instead of manually setting any margins or paddings, I used mixins every time. That way, I'm sure the vertical rhythm is left intact. An example of using the mixin above is:

```
h3 {
  @include margin-bottom(2);
}
```

Which translates to:

```
h3 {
  margin-bottom: 60px; // 2 × line-height in pixels (30px
  margin-bottom: 3rem; // 2 × line-height in decimal (1.5
}
```

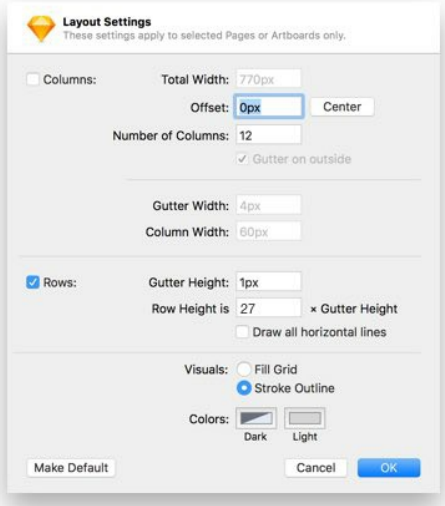
The Sass mixin sets the bottom margin in rems, and in pixels as a fallback.

Vertical rhythm in Sketch

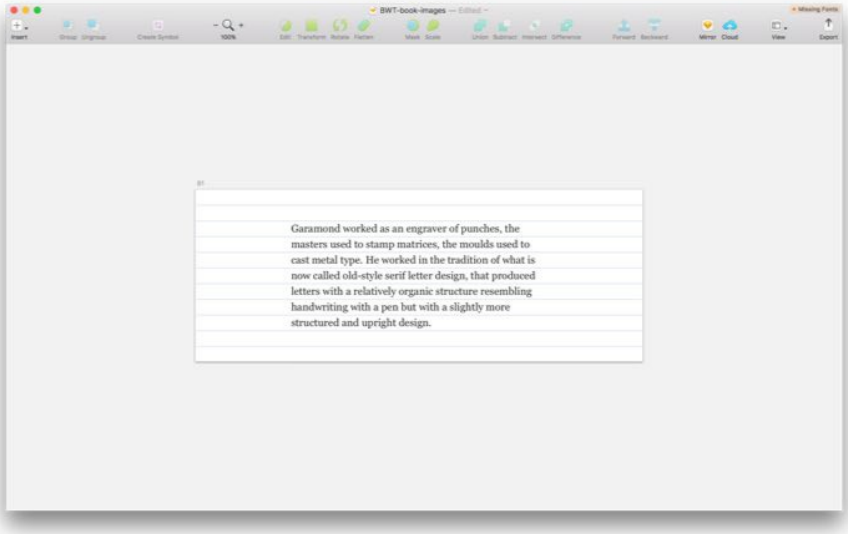
You don't need a plugin to set a baseline grid in Sketch. We can do that with the features Sketch offers out of the box with a simple workaround. Once in Sketch, add an artboard. Then, go to *View* > *Canvas* > *Layout Settings*. The layout settings window will open. Here's what you need to do:

1. Tick the "Columns" checkbox and change the value of "Total width" to match the width of your artboard. "Offset" must be set to 0. You can disable the "Columns" checkbox then, as we don't really need them.
2. Now tick the 'Rows' checkbox if it isn't already. Change the 'Gutter height' to '1px' and 'Row height' to what your line-height is in pixels + 1 pixel (to accommodate the 'Gutter height' we set in the previous step. In this case, that's 27 + 1 which translates to 28 pixels.
3. Then change the "Visuals" radio to "Stroke outline".
4. Change the colour of the lines to something subtle. You need to

change the “Dark” colour. I usually set it to a very light shade of blue. Click “OK”.



Bam! You now have a baseline grid in Sketch. Simple, right?



Note: your content will display behind the grid and there’s no way of

changing that. What you can do is change the opacity of the lines. You can do that in the colour selection window while selecting the colour in Step 4 described above.

Let's apply rhythm to our website

Well done for making it so far. Rhythm is one of the most important things in typography and it doesn't take too much effort to get it right. I know that some designers like to use what looks right to their eyes (they make the decision based on what looks right optically), and others prefer to back their decisions with maths. I believe both can and should work together. I like to start off with what's mathematically correct and make optical corrections when they're needed.

In the previous chapter, we decided which typeface to use for headings on our example website. The following is where we left off.

We have our body text set to 18 pixels and our line height is 1.45 (26 pixels). With that, we have everything we need to apply vertical rhythm to our website. I'll be using Sass throughout this example and I'll only be focusing on desktop, to make things easier to understand. In a real

example, the process would need to be repeated for the mobile text size and line height (if it's not the same as on desktop).

Let's start by setting the line height and the bottom margin for all elements.

```
// Variables
$base-font-size: 112.5; // Gets used as %
$line-height: 1.45;

// Vertical rhythm mixins
@mixin line-height($number) {
  line-height: #{ $number * $line-height + 'rem'};
}

@mixin margin-top($number) {
  margin-top: #{ $number * $line-height + 'rem'};
}

@mixin margin-bottom($number) {
  margin-bottom: #{ $number * $line-height + 'rem'};
}

html {
  font-size: #{$base-font-size + '%'}; // 112.5% = 18 pix
}

* {
  @include line-height(1);
  @include margin-bottom(1);
  @include margin-top(0);
}
```

We've just reset the line height and the margins of every element. They all fit the baseline grid now. With that, we created a problem at the same time. Headings are usually larger than body text size and won't fit into a single line height. We need to change the line height for all headings. Let's change their margins while we're at it as well.

The easiest way to do that is to create an object with a list of all headings and their values for their line height, top and bottom margins.

```
// Headings parameters [ h1: line-height: 2 × 30px, margi
$headings: (
  h1: (2, 3, 1),
  h2: (1.5, 2, 1),
```

```
  h3: (1.5, 1, 0),
  h4: (1, 1, 0),
  h5: (1, 1, 0),
  h6: (1, 1, 0)
);

// Set line-heights and margins
@each $heading, $properties in $headings {
  #{$heading} {
    @include line-height(nth($properties, 1));
    @include margin-top(nth($properties, 2));
    @include margin-bottom(nth($properties, 3));
  }
}
```

All right, now we're talking. Now even all the headlines fit the baseline grid. And that's pretty much it when it comes to vertical rhythm. We have a good foundation to work on. Let's make sure we set all our line heights and margins with the Sass mixins from now on and everything will be all right. Check out our example website so far at *betterwebtype.com/book/c6*.

Key takeaways

- Letter spacing and kerning adjust the spacing between individual letters, but letter spacing changes all spaces equally, while kerning adjusts the spaces depending on their neighbouring letters to improve legibility.
- Don't use justified alignment for text on the web. Web browsers still do a poor job at rendering it, resulting in "rivers of white space" between the words. That impacts the flow and horizontal rhythm, making the text harder to read.
- Paragraph indenting can be a neat way of indicating new paragraphs using horizontal rhythm. Indentation is most commonly set to either 1em or 1 line height.
- Hanging punctuation helps keep the flow of the text intact. Quotation marks should be placed outside of the main body text, as should the bullet points in unordered lists and numbers in ordered lists.

- Vertical rhythm makes a text easier to scan and read, and it establishes visual hierarchy on the page. The best way to remain consistent in defining vertical margins and sizes is using a tool like Sass and its mixins.

Modular scale and meaningful typography

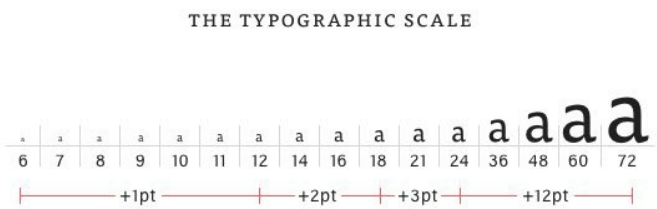
How do you assign text sizes when you work on a website? Do you just randomly pick numbers depending on what you think looks good? Do you usually end up with way too many different sizes and no idea where they came from? Does your website text start to look chaotic after a while? How about your CSS? Can you manage the upkeep without going crazy?

I remember how I used to set text sizes before I met the idea of modular scales. It was very similar to what I described above. No real reason for a particular size. No real meaning. It was all just numbers. Random numbers mostly. *Main body type..? 16 pixels should be fine. Heading one..? Yeah let's go for 40 pixels. Figure caption..? 15 pixels? Why not.* This was my process before.

A modular scale brings order, harmony and meaning to your text sizes, code, typography in general and in the end—what matters the most—your website.

The next time you need to choose a size for a particular text on a website don't choose from the infinite sizes incremented by 1 pixel: choose from a range of predefined and calculated sizes that are in some sort of harmonious relationship. This will help you keep your code modular and therefore clean and easier to understand. As this works well for you, it works equally well for the reader. Instead of figuring out what numerous different text sizes represent, they only have to figure out a few. Because they repeat more often, it's easier to decipher their role and the relationships between them. The visual hierarchy and order are properly established.

Most graphic design tools come with a default typographic scale. So do some front-end development frameworks.



Diatonic scale.

This is the so-called diatonic scale. You probably stumbled upon it before. I'm sure designers that work in either Photoshop or Sketch have. In the worst case you follow the sizes in this scale. "Worst case" here sounds worse than it should, as there's nothing wrong with using this scale. It's a scale like any other.

The origin of the typographic scale

Typographers have been using this scale for centuries. As it turns out, one of the main reasons for using a scale in printing was similar to ours for using modular scales in web design. The only major difference is that they used a single scale while now we have many to choose from.

Typesetting digitally is so much easier compared to how type was set before. Punch cutters used to cut metal moulds and each letterform was cast as a metal block. Producing lots of styles, weights and sizes for each typeface was an excruciatingly long process. Through the centuries of casting metal type, a standard began to develop: the diatonic scale, based on the same-named scale in music. With it, typographers still had a wide range of sizes to choose from but the process of producing new typefaces was simplified and faster. As a side effect, print work became harmonious and standardised.

The typographic scale simplified the process of typesetting. It was established from a practical need and as an aesthetic enhancement.[35] That's exactly what we're looking for in using modular scales on the web. It gives us a predefined range of sizes we can reuse, thus bringing order to what is otherwise chaos (in my opinion, anything that isn't modular can quickly turn into chaos).

Typographic scales in the digital age

With digital type being so easy to manipulate, we introduce scales to guide us towards more meaning in our typography. A web designer should never set type for a website without using a scale. The default typographic scale is a good place to start. But sometimes you may want to experiment with different ones. Throw the old out of the window and bring in something new. Something fresh. Tim Brown from Adobe came up with the idea of Modular scale[36]—a tool to create your own scales. The idea of using a modular scale is very simple:

Modular Scale by Tim Brown.

1 Choose your base size
Once you've decided what typeface to use for your main body you need to find out what size works best. As we've seen before, not all typeface sizes are equal. Some seem taller or bigger, some shorter and lighter. Put it in your browser and experiment with the size, line height and measure. Once you've decided what size works best, use it as your base size for the modular scale.

2 Choose a scale
There are quite a few to choose from. Explore and compare them. The golden ratio is probably the most popular. It's been popular in graphic design and architecture for ages. It's based on the Fibonacci numbers which produce a pattern that is supposedly present everywhere in nature.[37] The one that I found works best for me is the Perfect Fifth with a ratio of

2:3. I find it to be more flexible than the rest. The goal here is to find out what works best for you and come up with your own unique style.

3 Define all your sizes so they match the scale
Once you've decided which scale to use, you must define all your text sizes by choosing one from the scale.

Example
Let's say we decided to set our main body text at 18 pixels and we chose the Perfect Fifth scale. We now want to set the basic text sizes that we'll use: Headings 1 to 4, figure captions and the small print. These are the values that the scale offers to choose from:

136.688px	7.594em	8.543em @ 16
91.125px	5.063em	5.695em @ 16
60.75px	3.375em	3.797em @ 16
40.5px	2.25em	2.531em @ 16
27px	1.5em	1.688em @ 16
18px	1em	1.125em @ 16
12px	0.667em	0.75em @ 16
8px	0.444em	0.5em @ 16

As we only need Headings 1, 2, 3 and 4 we need four sizes larger than our base. We also need two sizes smaller than the base for the figure caption and small print text. So we may assign the sizes like so:

Element	**Size**
Heading 1	3.375rem
Heading 2	2.25rem
Heading 3	1.5rem
Heading 4	1rem
Figure caption	0.75rem
Small	0.5rem

I only use Headings 1 to 4 most of the time so I have a habit of setting the text size of Heading 4 to match the body text size. I then set it in either bold or italic to emphasise it.

Using modular scale with Sass

I'm sure the developers going through these can already see how this fits perfectly with the typical development mentality. It's modular, hence the name modular scale, and scalable. And what's best, this works great with the CSS preprocessors. Let's take a look at how we can set it up in Sass in three easy steps.

```
// 1. Base size and ratio

$base: 1.125; // = 18px
$ratio: 1.5; // Perfect Fifth

// 2. The formula for modular scale is (ratio^value) × ba

@function pow($number, $exponent) {
  $value: 1;

  @if $exponent > 0 {
    @for $i from 1 through $exponent {
      $value: $value * $number;
    }
  }

  @return $value;
}

// 3. Let's make it simpler to use by combining everythin

@function ms($value, $ms-ratio: $ratio, $ms-base: $base){
  $size: pow($ms-ratio, $value) * $ms-base;
  @return #{ $size + "rem" };
}

h1 {
  font-size: ms(3); // = 60.75px
}
```

All we need to use modular scale in a preprocessor like Sass is a couple of variables and functions. Each modular scale breaks down into a single number: 1.5 for the Perfect Fifth, for example. So we save that to the variable `$ratio`. The other variable we need is the base font size. This needs to match the size of your body font. It's 18 pixels in this example so I set it to 1.125em.

$$(ratio^{value}) \times base$$

$$(1.5^{3}) \times 1.125em = 60.75px$$

This is the formula to calculate any value in a given modular scale. That's why we need the power formula.

In the third and final part we just write that main formula in code and it's ready to use. We preset two out of three parameters in that formula, so all it needs is the value parameter. `ms(3)` means that we want the third size on the scale.

But what about the negative integer exponents? It's true that a modular scale comes with sizes larger but also smaller than the base size. To set text to one of those sizes, we need to allow negative exponents. It turns out that's quite easy to do. We just need another if statement that checks whether the exponent is less than 0 and do a different calculation if it is. The whole pow function in the example above would change to the following:

```
@function pow($number, $exponent) {
  $value: 1;

  @if $exponent > 0 {
    @for $i from 1 through $exponent {
      $value: $value * $number; //Multiply by $number if
    }
  } @else if $exponent < 0 {
    @for $i from 1 through -$exponent {
      $value: $value / $number; //Divide by $number if ex
    }
  }

  @return $value;
}
```

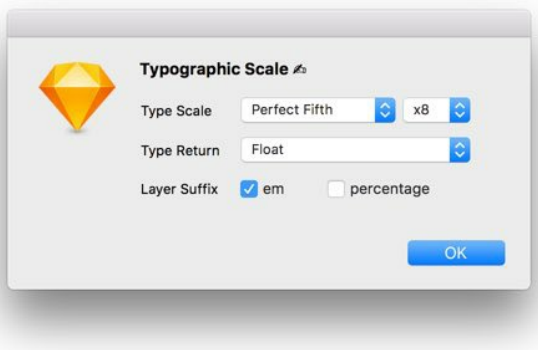
Now our modular scale function in Sass is bulletproof and we can do this as well:

```
.class {
  font-size: ms(-1); // = 13.5px
}
```

We're now all set to start using modular scale for our website. We have our body text size set at 18 pixels and a formula to calculate other sizes based on that. No more needless and meaningless deciding between 24 and 25 pixels. We use a predefined size from the scale. If we think it's too small, we use the one that comes after it. If too large, we use the one before it. It may feel limiting at first but go with it for a while. You'll get used to it and your work will benefit from it as well. Modular scale makes everyone happy: developers because of the modular solution for something that can become too complex and chaotic very easily, designers because they know their typography is meaningful and readers because the content is so much easier to understand and consume (though they probably won't notice it).

Using modular scale in Sketch

Using modular scale in Sketch is easy but you'll need a plugin called "Typographic scale".[38] Once installed, put some text into your artboard and set it to your preferred base size (most probably your body text size). Then go to *Plugins* > *Typographic scale*. The plugin window will pop up where you can choose which scale you want to use and how many sizes you want.



Confirm your choice and the script will add more text layers with sizes according to the modular scale you chose. Some sizes will be smaller than your base size, but most will be larger.

I recommend saving these sizes as text styles so you can easily reuse them later on.

What comes first, the grid or the scale?

At this stage, you're probably a bit confused. You're probably thinking: OK, I have my baseline grid which is x pixels, how does my modular scale work with the grid? Should I set a modular scale first and then apply a grid? The other way around? Does the line height of the body text need to be a value from the modular scale? What about when it comes to margins and paddings? Do they need to be a value from the scale? I had the exact same questions on my mind when I first started working with modular scales and grids.

First of all, both baseline grid and modular scale are tools to help you out. Things get complex quickly if we start thinking about them too much. Their sole purpose is to guide you, not limit you. But they do need to work well together. Let's answer those questions.

Should I set a modular scale first and then apply the grid or the other way around?
As I recommended at the start of the book: start with the content. This means finding out what typeface and at what size you need to set your main body text first. Then you perfect your paragraphs by getting the line height and line length right. At this point, it really doesn't matter whether

your next step is setting the baseline grid or using a modular scale to set other sizes. Your baseline grid is defined at the moment you decide what line height you'll use. This means that the modular scale sizes will always need to work with the baseline grid and not the other way around. You can't and shouldn't modify the baseline grid to make the text sizes fit.

Does the line height of the body text need to be a value from the modular scale?
The answer to this question is simple: no, the line height only needs to match the other two paragraph features from the triangle (line length and font size). Using a value from the modular scale would be too limiting and we don't want that. The same goes for line heights of other elements like headings. You want your line height to be based on the text you're setting it for. Not on some number from a scale that has no real connection to it.

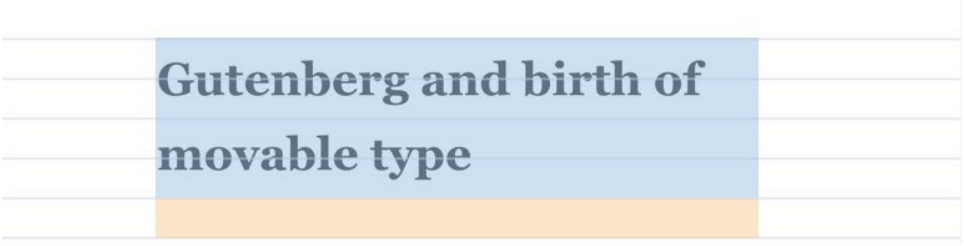
Do the margins and paddings need to match a value from the modular scale?
I would say no but you could do that, especially with the side margins and paddings. I prefer to use multiples of line height for all margins and paddings but if for some reason you don't want that, try using values from the scale. I definitely recommend using line height multiples for top and bottom margins and paddings.

What do I do when my elements don't align exactly to the baseline grid because of the decimals in sizes?
Sometimes you'll get a size like 22.4 pixels when you multiply your base font size with line height ratio (16 × 1.4). Rendering of sizes like that depends heavily on the browser as some round numbers up and some round them down. In such cases you'll notice that the elements seem to be off the grid by a pixel. I know it looks awful but you shouldn't worry about it. You vertical rhythm is intact and your sizes are ok.

This seems quite simple now, where's the tricky part?
In short, modular scale sizes should mostly be applied to text sizes. But you need to make those sizes that are based on values from the scale, and line heights that are based on the baseline grid, work well together. That's the tricky part. Let's take a look at a proper example.

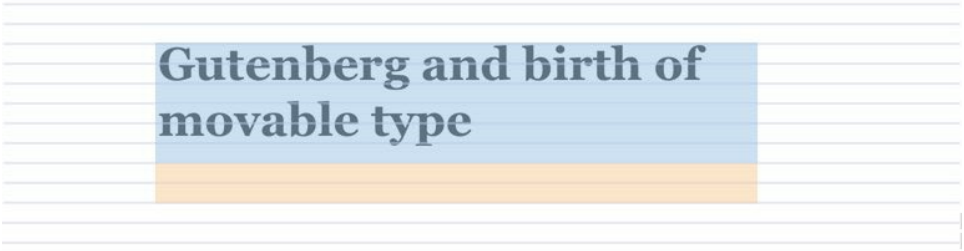
Let's say we have our body text set to 18 pixels with a line height of 30 pixels. We want to set the sizes for our headings so we establish a scale, based on the minor third which equals 1.2. We decide that the font size of Heading 1 should be 37 pixels which is one of the values from the scale. We set its line height to 60 pixels to accommodate its size.

Looks nice, right? Not so fast. Let’s see what it looks like when the heading stretches across two lines.



We know that headings look nicer with a smaller line height than body text, so this is not what we want. But what can we do? We have our sizes from the scale and we have our grid. It all fits but it doesn’t look right. There’s one rule in typography that will guide you in situations like this: *when torn between two options like that, go with what looks optically right, not with what’s mathematically correct*. Based on that rule, you could go and change the line height of the heading to anything that gets the optically better result. You can do that, there’s nothing wrong with it. I’m a bit of a purist and when I work with a grid, I like to stay within the grid. Do I use the mathematically correct solution instead of the optically right one? Nope, I bend the rules.

Instead of using the line height of 60 pixels, I use 45 pixels. Where does the 45 come from? It’s line height multiplied by 1.5. I like this solution because instead of breaking the rule of the grid, it bends it. Because it’s still a multiple of the line height it (kinda) remains true to the grid. A two-line heading in this case gets a total of three line heights.



And the best part? If you calculate the ratio of the heading's line height (45 ÷ 37) it's 1.22, which is in the recommended range for headings of 1.1 to 1.3. As I said earlier in the book, there are no particular rules in typography. It's up to you to shape your own and then bend or even break them, as Robert Bringhurst put it best[39]:

> *By all means break the rules, and break them beautifully, deliberately and well.*

Gutenberg, the meaningful web typography starter kit, uses a double baseline grid to make it more flexible when it comes to font sizes, margins and how they fit with the baseline grid.

Let's set our text sizes

OK, how does all this apply to our example website? Let's add some more content to our example and set some sizes by taking values from the

modular scale. For this example I'll use the Perfect Fifth scale, my favourite.

We wrote some code in Sass in the previous chapter which sets the margins and line heights for all our headings. Let's modify that a bit so it also sets their sizes.

```
$base: 1.125; //= 18px
$ratio: 1.5; // Perfect Fifth

// pow and ms functions (from previous chapter)
@function pow($number, $exponent) {
  $value: 1;

  @if $exponent > 0 {
    @for $i from 1 through $exponent {
      $value: $value * $number; //Multiply by $number if
    }
  } @else if $exponent < 0 {
    @for $i from 1 through -$exponent {
      $value: $value / $number; //Divide by $number if ex
    }
  }

  @return $value;
}

@function ms($value, $ms-ratio: $ratio, $ms-base: $base){
  $size: pow($ms-ratio, $value)*$ms-base;
  @return #{ $size + "rem" };
}

// [ h1: font-size: [x value from scale], line-height: 3
$headings: (
  h1: (3, 3, 2, 1),
  h2: (1, 1.5, 2, 1),
  h3: (0, 1, 1, 0),
  h4: (0, 1, 1, 0),
  h5: (0, 1, 1, 0),
  h6: (0, 1, 1, 0)
);

// Set font-size, line-heights and margins
@each $heading, $properties in $headings {
  #{$heading} {
    font-size: ms(nth($properties, 1));
    @include line-height(nth($properties, 2));
```

```
    @include margin-top(nth($properties, 3));
    @include margin-bottom(nth($properties, 4));
  }
}
```

This relatively short and simple code sets all our headings' font sizes, line heights and margins. And the best part is that it's very easy to maintain. Line height doesn't accommodate the font size? Change it and check the result. Need a bit smaller line-height for a heading to get closer to that recommended 1.1 to 1.3 range? Try reducing it by 0.5 and check the result. We don't need to do that in this particular example because our heading 1 size fits perfectly into a 3 × line-height space.

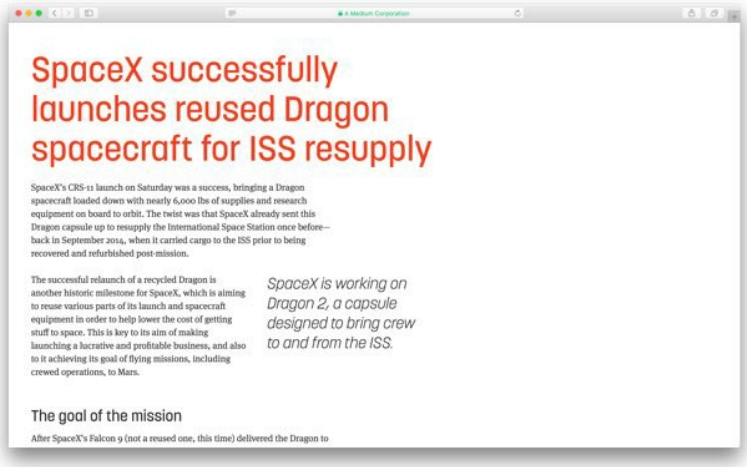
Now let's set some other font sizes.

```
small, figcaption {
  font-size: ms(-1);
}

.big-quote {
  font-family: $font-korolev;
  font-size: ms(1);
  float: right;
  max-width: 10em;
  @include line-height(1);
}

// etc.
```

After adding the `.big-quote` to our page I noticed that there's too little room between the lines. For the particular text size that is. I could go and change that size but I want to keep it. I know that doubling the line-height will be too much so I go and change its line-height to 1.5 × line-heights. The result is clearly better and it still fits the grid. We just bent the rules a bit.

```
.big-quote {
  ...
  @include line-height(1.5);
}
```

Modular. Simple. Getting 80% of your web typography right only takes a few minutes now.



Check the example website so far at *betterwebtype.com/book/c7*.

Key takeaways

- Never design without a scale. If you don’t use a modular scale, use the diatonic scale. Don’t just assign text sizes randomly.
- Using a modular scale is easy: choose your base size, select a scale, use sizes from that scale. Use the Sass snippets from this book to simplify the process.
- It’s OK to bend or even break the rules if necessary. What looks right optically is more important than getting it correct mathematically.

Page composition

The web is loud, yet boring

Clutter and noise

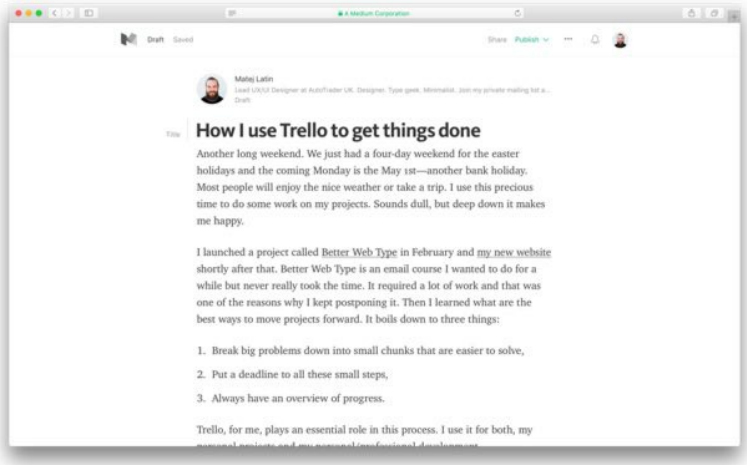
There are two main problems with the web today. One, which we partly covered already, is the lack of focus on content. Websites are still being drawn by designers and pretty much every website design starts by drawing a box or two. “This box is the header, this will be the article and that one is the sidebar” is still very much the thinking behind web design. We start designing websites layout first, and the mobile-first approach just makes us reconsider how we order those boxes vertically before we start moving them to the left and to the right on bigger screens.

It has brought us this far: the web is a much better place than it was a few years ago, but still, it’s time to move on from that. It’s time we start thinking about the content first and forget the boxes. It’s time we stop producing templates and layouts and start producing websites that revolve around the content that its visitors want to read. It’s time we stop cluttering our websites with meaningless things, just because we have room and just because it may earn us a few dollars (pop ups, billboard ads, I’m looking at you). Today’s web is a web that screams. Yes the all-caps texts are disappearing, but we’ve found other ways to scream at the user. “Click here”, “Subscribe now”, “Share with your friends” etc. I know that getting rid of advertisement online is unrealistic but I’m not even saying that we should do that. Advertising isn’t a problem when a website is content-focused, looking to provide its users with a great experience. Nobody will mind a tastefully placed ad on a website like that. The problem at the moment is that most of the web works the other way around—grabbing clicks of billions of users is more important than anything else. Let’s be honest, there will always be websites like that. And that’s OK. But that doesn’t mean that we should take the status quo for granted and it doesn’t mean that we can’t challenge that. Even if it’s only by tackling one website at a time.

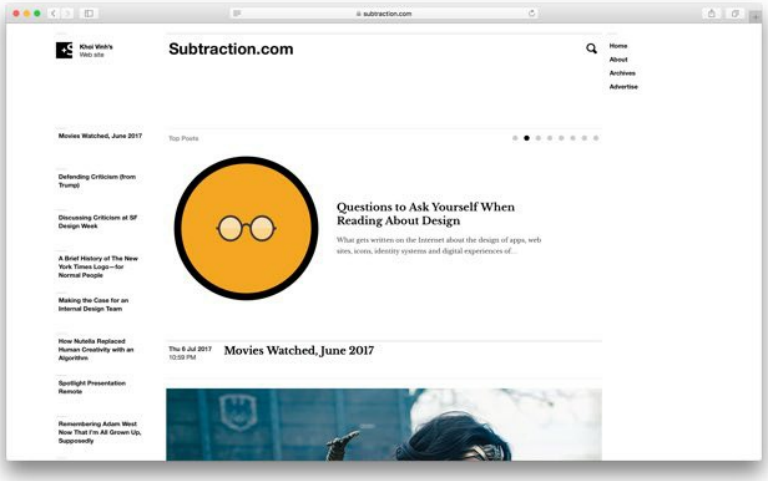
The boring, centred column

The other problem is that we seem to have completely run out of ideas. Some of you may remember the good old 960px grid.[40] That was way before the responsive web. Web design back then was easy: all you had to do was decide how many columns you wanted to use and which one of the web-safe fonts was the most appropriate. Drawing boxes inside those

columns was all the rage; you simply couldn't go wrong with it. We like to think that we've moved far beyond that but have we? Have we truly? Let's take a look at what is (by many) considered great web typography.



One big column, centred. Yup, that's it. That's the biggest achievement in web typography so far. Trying to imitate a book but not actually succeeding at it. A book is made of pages of pre-defined size, therefore the way the content is displayed on these pages is adapted to that size, to that canvas. On the web, we have this big, dynamic, exciting and responsive canvas and the best we can do is one centred column with equal margins on the side. That's the remnant of the 960px grid—the idea that the content should fit into a box. We're still imitating a medium that's completely different from the web. The new responsive and dynamic canvas of the web is not a restraint. It's an advantage. Surely we can do much better with a medium that people who were restrained to the physical restraints of paper would kill for. This is a new medium, a new canvas. It's time we forget the imaginary restraints of the one that preceded it. There are a few pioneers out there already.



Take a look at the website pictured above—Subtraction by Khoi Vinh.[41] How would you describe it? I'll start with the things that don't stand out. Take a look at the side margins. Do you notice that they're different? You might think that's just plain wrong but that only means you're still in the 960px box. Anyway, the margin on the right is double, if not triple the size of the one on the left. That's not by chance, it's by design. The content is obviously arranged by some sort of grid, yet it feels dynamic. The grid reinforces the hierarchy and information architecture, it's not there to merely align a couple of boxes. While we're at boxes, do you notice any? All I see is text. Content everywhere. The layout of the website is unconventional but it works very well. Navigating it is extremely easy. Take a look at the column on the far left. It doesn't even have a title but I know exactly what it is. It simply links to other (probably popular but not the freshest) content on the website. There are only two graphical elements on the whole website that are constantly present: the logo and the magnifying glass icon. Everything else is type: another example of a website that is "95% typography". Let's take a look at the post page.

The same as on the homepage, but much more obvious, the content of the article is clearly aligned to the left. Look at all that white space on the right of it. No, distractions, no noise. Just content. Your brain might still be hardwired into thinking that's wrong. We can't just have the content aligned to the left and then all that emptiness there. It looks wrong, it's not symmetrical, it needs to be fixed. Right? Nope. That white space is there with a function. "White in typography is what space is in architecture,"[42] Massimo Vignelli claimed. "It is the white space that makes the black shapes sing," he elaborated. But what does that mean?

This all goes back to contrast. Contrast, as we found out already, has a huge role in design, architecture and finally in typography. Contrast is what brings a design to life. Contrast of an enormous empty room is what makes a (meticulously placed) unique piece of art stand out. With no contrast, there is no life. It's either the blackness of the universe or the whiteness of a canvas. Without anything on it, it's dead. Once we start putting type on our canvas it comes to life. But it's the type that is the living thing, not the canvas. The canvas is there only to make the type stand out. It's the same with the Subtraction website. The white space brings the content to life. It makes the website unique, as it makes it stand out from the sea of centred single-column layouts. As we like to imitate the book in our chase after better web typography, let's take a closer look at it and see if it really is all centred and symmetrical.

Symmetry is a thing of the past

"Asymmetry is the rhythmic expression of functional design,"[43] Jan

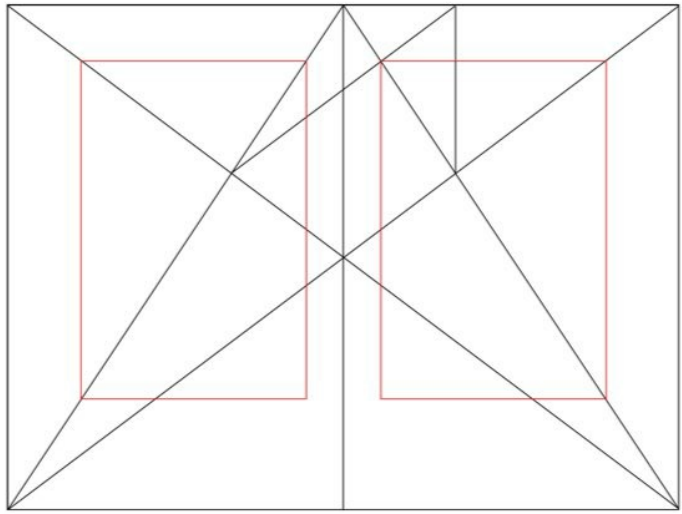
Tschichold claimed in his typographic manifesto book *The New Typography*. The 1920s were still an age of symmetrical typography but Tschichold and other like-minded people were slowly but surely changing that. He challenged the symmetry found everywhere in typography at the time, and claimed that asymmetry is more logical and optically more effective. What I recall the most from this book is that symmetrical form does not draw its laws from within itself (like asymmetrical form does) but from outside—the canvas defines the content, not the other way around. The only thing that provides variety in symmetrical typography is the typeface. That's where the need for ornaments in typography came from. It was something else that could provide some variety. Tschichold challenged symmetry in typography because he thought it was, well, plainly said, boring.

Just like any medium that preceded it, the web started out with poor typography. Typography that needed to improve in order for the medium itself to improve. Typography that was imitating, and repeating, all the mistakes that printed matter as a medium made instead of learning from them. The most popular layout for a "content-focused" website these days is the centred single-column layout.

We're doing the exact same thing that Tschichold challenged. We're taking a canvas and letting it define our content. The canvas forces the

content into symmetrical alignment because that's the most obvious thing to do when starting with canvas. Therefore we're left with equally distributed white space around our content, which can be conveniently filled up with advertisement or worse—ornaments. In order to make our website unique and alive, we need to start with the content and let it shape the design through its natural form.

Grab a book. It doesn't matter which book—the one that's closest to you. Take a look inside and inspect the margins of the page. A page with regular content, not a title page. Compare the outside margins with the ones on the inside. Also, compare the margin at the top of the page to the one at the bottom. Do you see it? The margins are not equal. The difference might not be obvious but it's there. Take a ruler if you need to and measure them. There are two reasons for margins not being equal. One is practical. A book is held by a reader with two hands, one on each side. The outside margins are larger so that the fingers holding the book don't cover the content. The reader doesn't have to move their hands in order to be able to read. The other practicality depends mostly on the type of book, but books that are meant to be studied usually have larger outside margins so they allow room for notes. The other reason is aesthetics. Having outside margins larger than the inside holds the content together. If the outside margins were smaller than the ones inside, it would seem like the content is trying to escape from the pages. Margins direct the reader in the right direction. They make the content more compact and they separate it from the canvas it is on—the page. They literally bring content to life. It's not as obvious visually, but it has an obvious effect. This is one of the examples of shaping a page for a book:



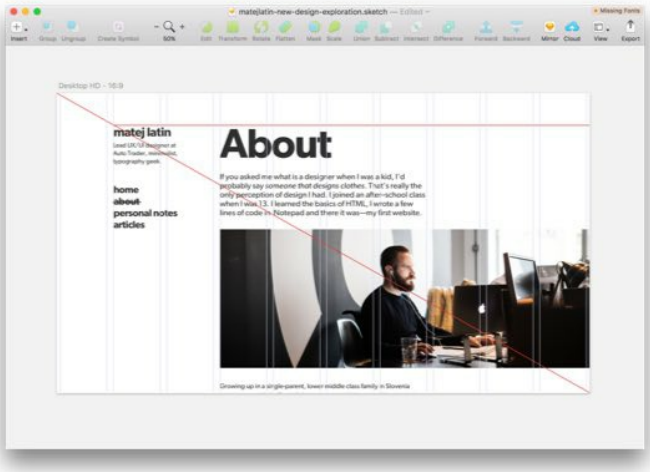
Take a look at the example of a page layout above. Notice that the margins are not equal (content is not symmetrically centred) and notice how dynamic the page seems but still everything is in harmony. Dynamic and versatile layouts like these start off with content and a grid. Why are we unable to define layouts like these for web pages? You're probably thinking it's easier to design for a printed page because its size is fixed (unlike screens). Is that really the case? Or are we just lazy and don't take time to design interesting layouts for each screen? OK, there are too many to design one for each of them but at least the most common ones? Others could fall back to a certain version, optimised for those sizes. And, by the way, there's not a lot we can do on narrow mobile screens. The single and centred column is OK for those. But shouldn't we try better for the desktop and tablet versions of our websites?

What can we do on the web?

There are four common screen ratios in use today: 5:4 made popular by the 1280:1024 displays, 4:3 old computer display standard, 16:10 made common by the 1680:1050 and 1920:1200 displays, and the most common 16:9.[44] According to some sources, the 16:9 was used by around 33% of web users in 2014—including mobile devices.[45] 16:9 then became a widespread standard and that number is probably higher at the time of writing this book. It's safe to say that the 16:9 is the most common and standard desktop screen ratio. In that case, couldn't we design layouts that

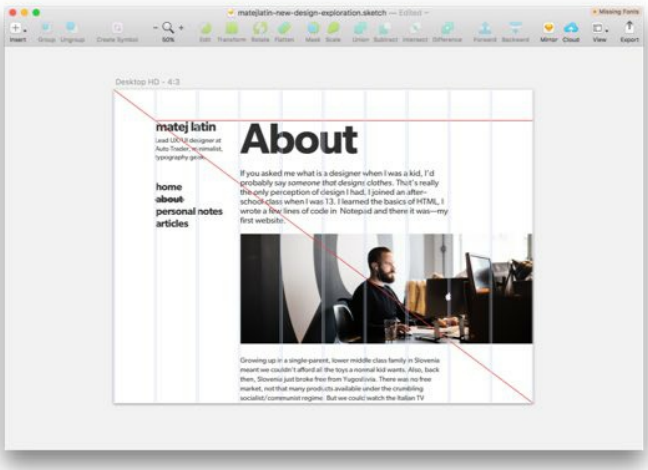
better use the space these screens provide? We can then use responsive web design techniques to optimise it for other common screens.

We can design interesting layouts for the web too. The fact that the web is viewed through a range of devices with different screen sizes shouldn't stop us. That only makes it more challenging. It makes it more exciting. Take a look at how I approached redesigning my personal website.



The design is based on a 10-column grid and, vertically looking, the content starts at the intersection of a beginning of a column and the diagonal of the screen. That defines the top margin of the page. All content on the page starts from that point—the title is aligned with the name of the website in the sidebar. The sidebar itself is two columns wide and the content spreads across five columns. That leaves us with three columns for margins: a number that can't be split equally. No worries, I don't want symmetry in this design; I want my content to define the canvas, not the other way around. The two-column margin on the right doesn't apply to some elements (image in this example). The photo

stretches all the way to the edge of the website, making the design more dynamic (i.e. not boring). I then adapted the design for the most common screen ratios, including 4:3.



To do that, I used the aspect ratio media query, starting with the 4:3 and then applying changes when needed for the 16:9 ratio.

```
// 4:3 screens first
@media screen and (min-aspect-ratio: 4/3) { ... }

// Modify for 16:9
@media screen and (min-aspect-ratio: 16/9) { ... }
```

The web is a new medium with a new, responsive but exciting canvas and we must treat it like that. Don’t start your web design projects by drawing boxes and don’t think about the grid as something that limits you to boxes. The grid needs to be based on the whole canvas and not all columns must be filled with something. It’s OK to leave them empty. Asymmetry and white space are desirable in design. A good designer makes the essence of the content stand out by emphasising the white space around it, while a bad designer makes it stand out by emphasising the content itself.

Typography and grids on the web

The way I used the grid in the last example is much more liberal than most of what we see on the web today. I started off with content, which defined the grid and both of them defined the layout of the page. The thing about the grid is that it’s a tool, and it should be used like one. The grid itself

shouldn’t define the product—in this case the website. It’s a tool to give meaning to our decisions, to reinforce the importance of the content. Therefore the grid should be built around the content, not the other way around. If we limit ourselves to what is possible with a certain tool, instead of building a new one that will help us create a better product, we will never progress.

Content-first Add navigation Add grid

If you pay close attention to the image above, you’ll notice that once the grid is used, the content shifts around the canvas a bit.

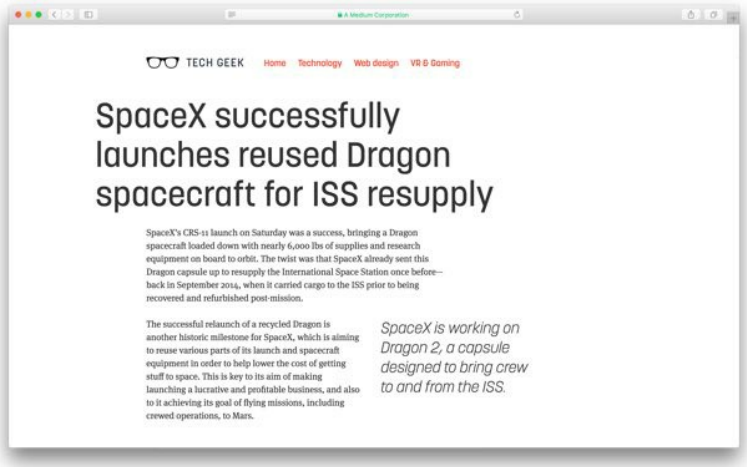
I started out with content and finding the typeface that I thought complemented it best. I shaped it, so it matched the idea I had in mind: big type, single typeface, lots of white space and minimalistic. I then developed a grid around it and finally rearranged the layout so it matched the grid. There are lots of grid frameworks out there but my favourite is Flexbox Grid.[46] With it, it’s very easy to set up fluid grids that make layouts like these possible. We’ll get back to fluid web typography and layouts in the next chapter when we’ll look at responsive web typography. Now let’s apply what we learned to our example website.

Let’s compose our pages

To apply our new knowledge of page composition to our example website, we first have to think about other content that will appear on the page. Things like the logo and branding, navigation, suggested posts, meta information, multimedia content and ads. We’re happy with the mobile layout as there’s not much we can do there. Let’s take a look at what we can do for the desktop—it’s OK to start mobile-first but that shouldn’t prevent us from optimising the layout for desktop and thus providing a better experience to the user.

We’ll need a more complex grid for this page as there’s a lot more content we need to fit in it. Let’s try a 16-column one with 20 pixels gutters. Let’s add the most important element to the page: the navigation. It consists of

the logo and a few links to the main content categories and a link to the home page. As I add it in, it hits me. The title font size is quite large and needs a lot of room. The content of the article will never take that much room, so I want to try and make the title stick out of the main content box to make the page look a bit more dynamic. Something like this:



Note: my favourite Flexbox grid doesn't support more than 12 columns so I decided to create one myself. It's quite simple: 100% ÷ 16 = 6.25%. That's our column width. I'll skip the gutters for the sake of simplicity of the example. I then created two Sass mixins: one that sets a width of the element by multiplying the number of columns and another one that sets the left margin. This way I could define my layout by simply setting width and offset of elements.

```
$column: 6.25%;

@mixin width($number) {
  width: #{ $number * $column };
}

@mixin offset($number) {
  margin-left: #{ $number * $column };
}
```

And here's how it gets applied in CSS:

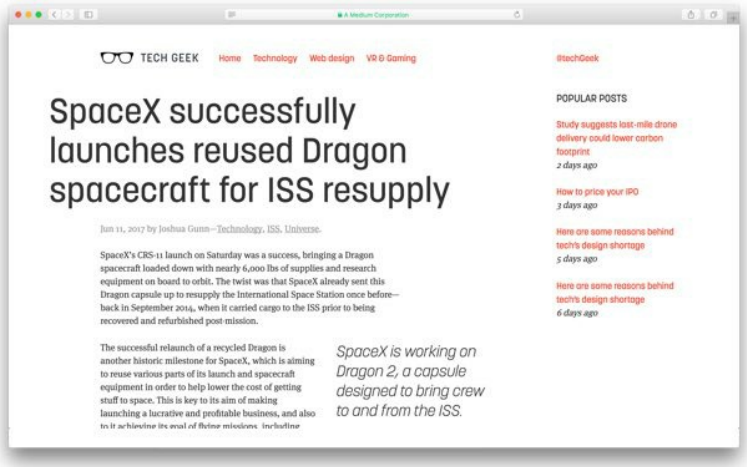
```
.main-body {
```

```
  @include width(9);
  @include offset(2);
  float: left;
}
```

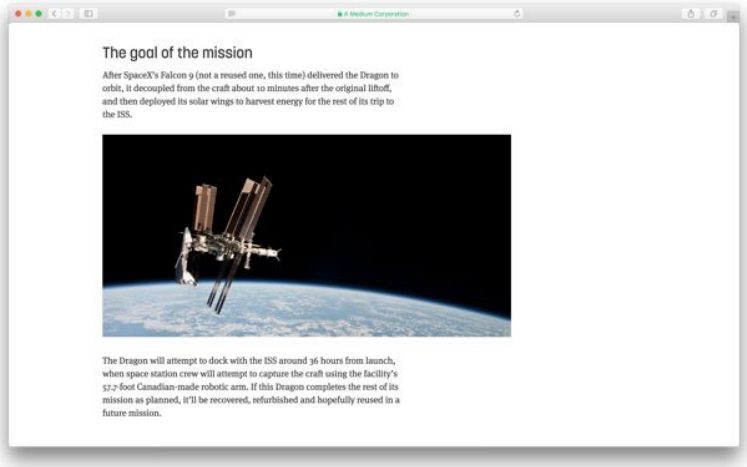
OK, this looks interesting. I like it and I think it has potential, so I keep it. Once I'm done with it, the title only has a margin of one column on its left side, while the main content and the navigation have two. Now let's add the post meta information. It has to go between the title and the post so there's not a lot of room to play around with this one. No major changes. Let's add the "suggested posts" section to the page. We have a lot of room on the right side, so let's try putting it there.

```
.sidebar {
  @include width(3); // But only 3 columns wide on deskto
  @include offset(1);
  float: left;
}
```

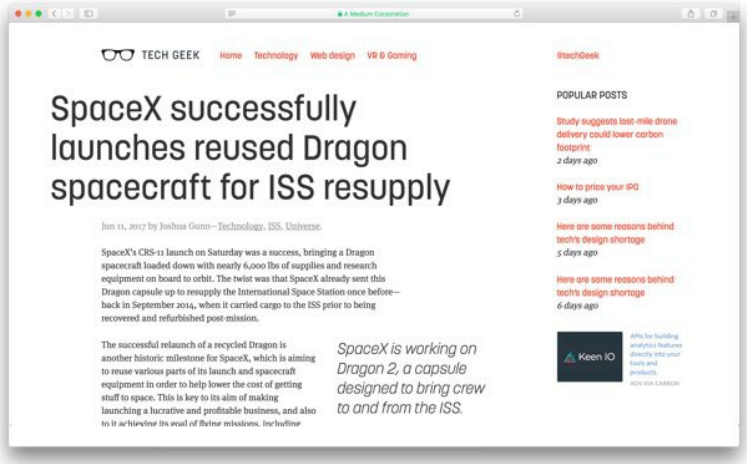
And we get the following:



Nice, we've added lots of content to the right side but we're still left with substantial white space wrapping the main content. Let's see how a photo would fit in. With the room we have left, we can try to make the photo slightly wider than the text paragraphs. Let's try one column wider.



The last step: let's add our finely placed and tasteful ad. The bottom right corner is very good for placing ads so let's give that a try.



And that again works very well. This was a step-by-step explanation of the process I went through when designing the layout of this page. This last part of the chapter is literally a transcript of my thoughts when working on it. Give it a try the next time you work on a project. Start with the content, define the typography, develop a grid and build a layout. It's a very organic process. Check out our example website so far at *betterwebtype.com/book/c8*.

Key takeaways

- As mentioned a few times in this book already, try starting your next web design project with the content-first approach. Don’t start by drawing boxes on a pre-set grid. Design the grid around the content.
- The single and centred column of body text is boring and it imitates a completely different medium: books. Asymmetry makes designs dynamic and it makes the content stand out.
- Symmetry is a thing of the past. It’s safe and boring. On the web, it leaves room for advertisement and ornamentation. None of those help us produce better-designed websites.

Responsive web typography

Web design used to be so simple. Well, simpler is a better word. Screen sizes were pretty much the same and they would only change to grow larger. But then screens shrunk massively. Mobile web traffic went from 27% in late 2013 to 53% in early 2017 (according to some sources).[47] More than half of the internet content is now consumed on a mobile screen: a screen that is much smaller than the one I'm staring at right now; a screen that is primarily used in portrait orientation. In fact, some studies suggest that less than 5% of mobile internet users use their phones in landscape orientation.[48] That presents a challenge. Instead of designing websites for large, usually 16:9 ratio screens, we now need to consider how a website is used on a tiny (compared to desktop) screen in a different orientation. To face this challenge, mobile-first and progressive enhancement concepts were born.

The idea is simple. Design a website for a mobile screen—the one with the most restrictions—and enhance the design for tablet and desktop screens, which come with fewer or even no restrictions. Taking what we learned about typography until now and applying it to this concept is easy. Set your base font size for mobile screens first. Base other text sizes on it by using a relative unit like em or rem. Then add a media query for larger screens and adapt when needed. This usually translates to larger base font size for larger screens. Other text sizes are also changed because of that relative unit we used.

```
html {
  font-size: 100%; // = 16px
}

@media screen and (min-width: 768px){
  html {
    font-size: 112.5%; // 18px
}
```

In the example above I set the base font size to 16 pixels and changed it to 18 pixels for the typical tablet screen width (when held in portrait orientation). Unfortunately, it doesn't remain that simple. We already learned that line height and line length need to change to accommodate any changes to the font size. But there's another problem when it comes to mobile-first. At some point, people stopped doing mobile-first and switched to mobile-only. There was a time when even desktop websites

had hamburger icons to stash all their navigation items inside. That's a complete nonsense from a usability point of view. It's still in use on mobile websites and I'm sure you'll still stumble upon websites that do that on desktop too. Navigation is the main tool to get around a website, so don't hide it if not really necessary. The single-column layout, which is still so common, is (I believe) also a remnant of the mobile-only trend. I mentioned the content-first concept a few times in this book and I challenged the single-column layout in the chapter about page composition. Now, as we look into responsive web typography, is the time to put the puzzle together.

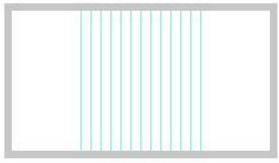
Mobile-first, mobile-only, content-first, WTF?!

Mobile-first as a concept works because it forces us to get rid of everything that isn't essential. Even ads have it tough on mobile; they only come in small but full-width banners. That's great. But it seems to only work in theory. It's hard to do something mobile-first and know where to draw a line to get out of that mentality. That's why websites end up in single columns and navigations get hidden away behind icons. It's OK to have the mobile-first mentality but it needs to be dropped at a certain stage. For me, that has always been the transition of a website from a tablet to the desktop. This means that a mobile and a tablet website will share most things, but the desktop website will be something quite different.

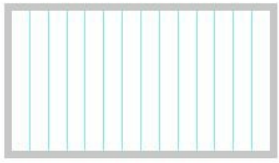
So far, we started with the content, shaped our perfect paragraph, designed our headings and we even designed a desktop website layout. There was very little mention of mobile-first, only content-first or content-focused. This led us to design our website around our content. And we designed it in its best form first—for the largest canvas available, the desktop. My personal guideline for designing websites is: don't sacrifice content to fit a concept. This means that I want to design the website in its best form for every device. It needs to be content-focused on every screen. I do use the

mobile-first concept, but mostly I reverse it. I take what I designed for the desktop and simplify it and modify accordingly for the smaller screens. But wait! Isn't that the opposite of mobile-first? Isn't that "graceful degradation"? Well, no, because I work with a framework that's built as mobile-first (my own Gutenberg web typography starter kit); I simply choose to focus on desktop first. Mobile-first, in this way, is a concept that enhances the content-focused approach. It's not a restraint. And I think that's key here: mobile first isn't a dogma, it's a guideline, a tool, an approach. We need to use it as such. We need to design whatever suits our content better, no matter the device, no matter the screen size.

The process doesn't really matter, as long as the tools are the right ones. Mobile-first is the way to go, but keep in mind that you need to step away from it at some point. Don't design websites for desktop with mobile restraints in mind. Use the canvas, complement the content. Start with a single column but expand into more on larger screens. And don't just split that one column into more. Split the whole canvas into multiple columns.



CENTRED COLUMN LAYOUT



DISTRIBUTED COLUMN LAYOUT

Pixels, ems and rems

OK, we just realised that mobile-first is not the dogmatic approach we should blindly follow. Let's take on another myth. There are still people arguing over whether it's OK to use pixels to set font sizes, widths and media queries in CSS or not. There are people saying that ems and rems are outdated, just because there's a new kid in town (I'm looking at you, viewport units). Pixels came under fire a few years ago because browsers wouldn't enlarge the text properly when a user zoomed in if a website had font sizes set in pixels. Instead, the whole website got zoomed in. (The browsers are much smarter these days and they do enlarge the text even if it's set in pixels.) Ems provided a solution for the problem back then, so everyone started using that. But there was a problem with those too. Check this out, see if you can solve the puzzle:

The HTML:

```
<h1>For whom the bell tolls <span>by Ernest Hemingway</sp
```

The CSS:

```
html {
  font-size: 1em; // = 16 pixels
}


h1 {
  font-size: 2em; // = 32 pixels
}


h1 span {
  font-size: 0.5em; // = ?
}
```

The base font size in the example above is 16 pixels. Heading 1 is 32 pixels. What is the size of the text of the span inside of Heading 1? It's not 8 pixels, it's 16. Em units are based on parent elements, in our case that's Heading 1. This can get confusing rather quickly.

Rems to the rescue. Rem means "root em", so it's always based on the base font size (the one set for the html in CSS). If all the ems in the above example would be replaced with rems, the answer to our question wouldn't be 16, but 8. It's less confusing like that and people adopted it quickly. The rems saved the day. Well, at least until viewport units came around.

Why you should use ems for media queries
Pixels are now fully supported as they should be in all major browsers so why not use that for everything, including media queries? It's much simpler to calculate with pixels than it is converting ems to pixels. According to a test, done by Zell Liew, pixels don't work that well in media queries in Safari.[49] In the test, he came to a conclusion that ems are the only ones that work perfectly well with all browsers so we should use those. We can turn to Sass for the calculations part. Here's a very simple and unpolished example of how it can be done:

```
$base-font-size: 16;
$tablet-breakpoint: 768;
```

```
@media screen and (min-width: #{ $tablet-breakpoint / $ba
  ...
}
```

Note: Using ems in media queries is a bit tricky. If your base font size is 18 pixels and you set a media query to something like min-width: 30em, the code inside that query won't kick into action at 540 pixels (18 × 30) but at 480 pixels (16 × 30). The em media queries are always based on the browser internal default size, which is 16 pixels.

Viewport units

Just like rems, and ems before them, viewport units are now becoming all the rage. CSS hipsters are calling on people to stop using anything else for this or that reason. I already met some of them. Let's take a look at these viewport units. The idea, again, is very simple. There's the vw (viewport width) and the vh (viewport height) units. One vw equals 1% of the viewport width and one vh equals 1% of the viewport height. That's a bit confusing at first but it's the only part that is confusing, I promise.

Fluid websites have been around for ages as we have been using the percentages as relative units for a long time. They make web design so much easier. And the new viewport units should make things even simpler. Let's see.

```
h1 {
  font-size: 2vw;
}
```

Note: viewport units are based on the viewport, while the percentage units are based on the parent element. The browser support for viewport units, at the time of writing this book, is solid (for the latest versions) but not complete.

Heading 1 in this case will be exactly 2% of the viewport width. This means that if we resize the browser window (like a proper web design geek, I'm yet to see a normal internet user do this), the heading will resize accordingly. This leads to the so-called "fluid web typography", but first, which units should we use?

I have a surprisingly simple answer to that question. Use whatever works for you, for the user and is supported by all major browsers. If you like to use rems, use those, if you prefer ems, use them. Don't let those CSS

hipsters discourage you from using anything that works well. I, personally, like to use rems for font sizes (based on modular scale) and ems for media queries. In the end, it's all pixels.

Fluid web typography

"Fluid web typography" is a relatively new term. The idea is interesting: use the viewport units for font sizes so that the text on a website scales as the viewport changes. Let's take a closer look at this revolutionary and rather unexplored approach.

The HTML:

```
<h1>Humane typography in the digital age</h1>
<p class="meta">January 25, 2016 · By Matej Latin</p>
<p class="category">thoughts on typography</p>
<p class="intro">An Essay on Typography by Eric Gill take
<p>The typography of this industrial age was no longer ha
```

The CSS:

```
body {
  font-size: 1.3vw;
  line-height: 1.4;
}

h1 {
  font-size: 3vw;
  max-width: 35vw;
}

.intro {
  font-size: 1.5vw;
}

p {
  max-width: 35vw;
}
```

Take a look at the code in action at *betterwebtype.com/book/c9e1*. Pay special attention to the font sizes when you reduce the width of your browser. All the text elements resize fluidly as the window shrinks.

Can you see the immediate problem? As we shrink the browser window, the paragraph text gets very small: too small for desktop usage. We can try

to remedy that with a bit of help from the good old responsive web design.

Responsive and fluid web typography
Let's take a look at how we can improve the previous example with a few simple media queries. We don't want the text to be too small on certain screen sizes. I add a media query with a parameter of min-width 40em as a break point. At that screen size, I switch to a smaller font size and limit the width of the paragraph to 60vw. This way we get a readable font size on most screen sizes. It would need a lot more work (and a lot more media queries) to get to a satisfying result. In fact, it took seven media queries to get to a result I was happy with. Check out the CodePen example at *betterwebtype.com/book/c9e2*.

Fluid web typography is an interesting concept but I have a few concerns about it at the moment. I can imagine how it works with mostly fixed layouts, but it takes a lot of work to make it responsive and perfectly shaped at the same time. It is the closest to imitating printed type we can get on the web, but the real question is: do we really need to? It seems like we're letting the canvas define the content again. Instead of focusing on the content and what format complements it, we try to adapt the content so it fits the canvas instead. I'm afraid that just as we came up with single-column layouts because of the mobile-first approach, we'll come up with canvas-filled websites just because fluid web typography allows that.

The other issue with fluid web typography is that it assumes that the smaller the screen, the closer it's held. But can we say for sure that a screen that is 1024 pixels wide is a tablet? Or is there a chance of it being one of those old screens that are still in use? Yes, we have the media queries to control that but it seems like a bit too much work for not a lot of gain. I will definitely experiment with it further. Let's see what will happen with "fluid web typography" in the near future.

Let's make our website responsive

As I said, I don't usually do this as described in the following section, because I use my own framework which is designed to be mobile-first; I just choose to focus on the desktop version first. But we haven't done any of the mobile or tablet-related web design in our example website so far. No worries, we can still make our website mobile-first. Let's see, we set our base font size at 18 pixels. Let's change that to 16 pixels for the mobile version and only increase the font size to 18 for larger screens.

```
$base-font-size: 100; // Gets used as %
```

```
$base-font-size-desktop: 112.5;

html {
  font-size: #{$base-font-size + '%'}; // = 16 pixels
  ...

  @media screen and (min-width: 64em){
    font-size: #{$base-font-size-desktop + '%'};
  }
}
```

I use the 64 ems which equals 1024 pixels. I usually use the same font size for mobile and tablet screens. The only thing we need to change is to limit the width of the paragraph whenever the screen size is larger than the width of our “perfect” paragraph. The one we designed back in Chapter 4, when we set a max-width property to 33 ems. Let’s make sure our paragraphs are well-designed for the most common screens. We’ll need to change the line height too.

The easiest way to handle different line-heights for mobile and desktop but keep the vertical rhythm intact is to:

1. Set breakpoint variables,
2. decide at which point you’ll switch to the largeer line-height (in this case it’s the tablet breakpoint),
3. Update the mixins we use for setting line-height and margins so they set the property in question for both—mobile and desktop.

```
// Responsive variables
$phablet-breakpoint: 37em;
$tablet-breakpoint: 48em;
$desktop-breakpoint: 64em;
$large-desktop-breakpoint: 80em;

$line-height: 1.4;
$line-height-desktop: 1.45;

// Update the mixins
@mixin line-height($number) {
  line-height: #{ $number * $line-height + 'rem'};

  @media screen and (min-width: $tablet-breakpoint){
    line-height: #{ $number * $line-height-desktop + 'rem
  }
}
```

```
@mixin margin-bottom($number) {
  margin-bottom: #{ $number * $line-height + 'rem'};

  @media screen and (min-width: $tablet-breakpoint){
    margin-bottom: #{ $number * $line-height-desktop + 'r
  }
}

// Limit the length of the paragraph
p {
  ...

  @media screen and (min-width: $phablet-breakpoint){
    max-width: 33em;
  }
}
```

Our desktop size paragraph has a line height of 1.45 but the lines are much shorter on a mobile screen. We need to reduce the line height so we set it to 1.4 which looks much better with such a narrow column. I also set the max width only for when the screen width is large enough; in this case, I use the 'phablet' breakpoint of 37 ems. We've now made sure our paragraphs look good on mobile, tablet and desktop screens.

Note: we changed the base font size for the mobile so it's 16 pixels but we're keeping the 18 pixels size for the desktop. We don't need to do anything to keep our modular scale intact because we're resetting the base font size by changing it for the `*html*` element*. This way, no matter the base font-size, the heading one on desktop for example, will always be 3.375 × base font size (no matter what size that is). If we didn't change the base font size, we would have to use the default (100%) font size on mobile but the new (112.5%) on desktop as a base for calculating the modular scale. This further complicates things so it's much easier to just reset the body font size.*

There's a lot of repetitive work in this example like limiting the line length of the headings, changing their sizes and line-heights etc. A whole book could easily be written just on that topic. It's best you take a look at the finished example on CodePen to see what the final code at this stage looks like (the URL is at the end of this chapter).

Updating the grid

We created our own 16-columns grid in the previous chapter. We now need to make that gird work with mobile phones and tablets. We already

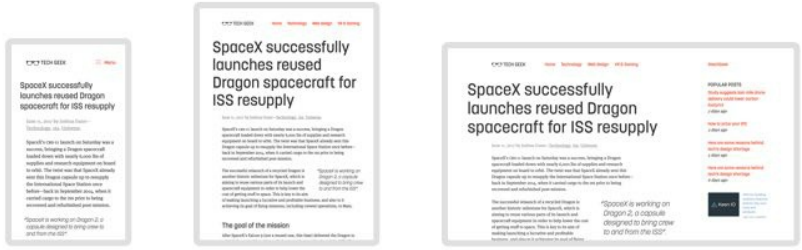
have all the pieces we need, we just need to put them together differently. Let's jump straight in. Our `.main-body` section (the whole left side of the page except the sidebar) has a width of 9 columns and is offset by 2 columns on desktop. It's not full width so there's room left for the sidebar. But we don't want the sidebar there on the mobile layout. Let's make the `.main-body` wider and push the sidebar down.

```
.main-body {
  @include width(14); // The main body is 14 columns wide
  @include offset(2); // ... and offset by 2 columns on t

  @media screen and (min-width: $desktop-breakpoint){
    @include width(9); // But only 9 columns wide on desk
    @include offset(2);
    float: left;
  }
}
```

And the following bit for the sidebar:

```
.sidebar {
  @include width(13); // The sidebar is almost full-width
  @include offset(2);

  @media screen and (min-width: $desktop-breakpoint){
    @include width(3); // But only 3 columns wide on desk
    @include offset(1); // Offset by 1 column only
    float: left;  // We only float it on desktop
  }
}
```

The rest is quite simple. We need to experiment with shifting our columns around and changing their widths for certain screen sizes. Even though we started by designing a desktop layout, our website is mobile-first. We just reversed the process. This way, we made sure our content looks perfect on all major devices.



Adapting the navigation and other bits to make them responsive is beyond the scope of this book but they're included in the source code on CodePen. Check the example website at *betterwebtype.com/book/c9*.

Key takeaways

- Mobile-first is one approach; don't let it grow into a dogma and don't use the mobile-first guidelines for desktop versions of a website. Design your website so it's content-focused on every device and every screen.
- Use ems for media queries but you're free to use any other unit you like for text and other sizes. Whatever works for you and your users, and is supported by the major browsers.
- Fluid web typography is cool but don't use the viewport units just for the sake of using them. Use whatever makes most sense. If you decide to use fluid typography make sure that the text sizes are easy to read on all screen sizes.

Part II: Micro Typography

Ligatures

In this chapter we start diving deeper into details that make great typography stand apart from good typography. We start exploring the world of micro typography which means that we'll mostly look at aspects that impact legibility. Studying typography at micro level will help you produce top-notch typographic styles. It will help you understand the difference small details can make. It separates typographic masters from the newbies.

Ligatures. You may have never heard of them before. Or you might have heard a designer mention them and shrugged it off as they probably don't concern you (for the web developers reading the book). Well, you may have never heard of them or paid attention to them but I guarantee you that you've seen them before. And once you know what ligatures are, you'll never be able to not see them. You'll probably start noticing them everywhere. Let's take a look at the definition of a ligature.[50]

> *In writing and typography, a ligature occurs where two or more graphemes or letters are joined as a single glyph.*

The most common ligature that everyone has seen is the ampersand. Yes, the "&" symbol. It comes in various shapes but at its core it's a combination of the letters "e" and "t"—"et" in Latin, meaning "and". How cool is that?

The evolution of the ampersand ligature.[51]

The word "ampersand" is an alteration of the phrase "et per se and" (that is: "et by itself [means] and"), which became "and per se and", and finally "ampersand".[52]

An interesting fact for German-speaking readers: the "sharp s" or "ß" is also a ligature (a combination of "ſ" and "s") but just like the ampersand

it became a character of its own.

Origin of ligatures

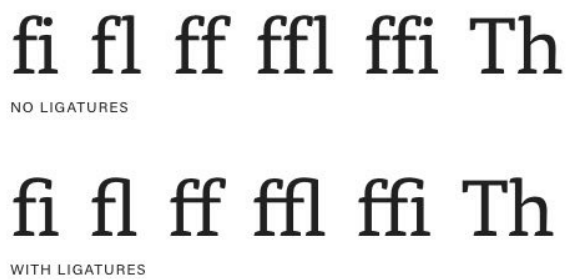
Ligatures go way back to the earliest known script: the Sumerian cuneiform. Sumerians had the process reversed, where ligatures would eventually be split into separate characters.[53] Other cultures' manuscripts also prominently feature ligatures, which proves that ligatures' main purpose for existence is saving time when producing these scripts. We can see similar origins in western typography. Before Gutenberg made his movable type printing machine, all books were handwritten. Every book was a unique copy, usually transcribed by a monk. Can you imagine transcribing a book like the Bible? It took a long time. So the scribes that were transcribing these came up with clever ways to speed the process up, even if just a little. They started combining some letters into single glyphs.

When the printing machine was first invented, it closely imitated the medium that came before it: handwritten books. With that, it imitated ligatures as well. But in print, besides saving time, there was another reason for combining letters into single glyphs: aesthetics. Take a look at the following sets of characters.

There's nothing obviously wrong with the sets of letters in the figure above. Tisa pro is a modern serif typeface and very well designed. The lack of ligatures in the first example isn't as obvious as in the second one, the same character sets in Georgia. In that second example we can clearly see some characters colliding. The top of the letter "f" clearly touching the dot of the letter "i", the top of the letter "f" touching the top of the letter "l", and the two "f" letters right next to each other but clearly separated by a small gap between them. All these are marked in the figure below.

All the unappealing features of these letter sets in Georgia.

Take another close look, notice the collisions. The combination of “f” and “i” is particularly noticeable. Particularly unappealing as well. This is where studying typography really grabs you by the legs, starts pulling and never lets go. Once you start noticing these little details, there’s no way back. So what is the solution for this unappealing set of letters? A ligature. Or ligatures, because there are quite a few of them still in use today.



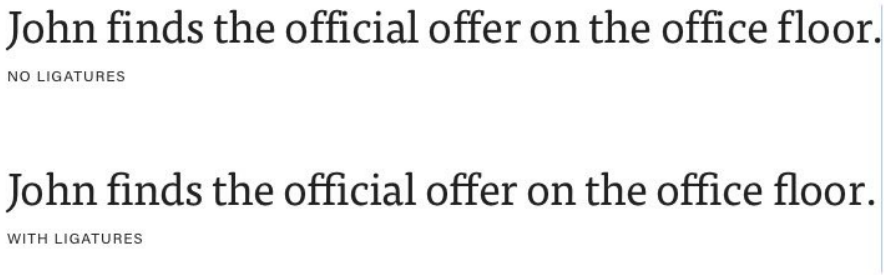
Same sets of letters, with and without ligatures. Can you spot all the differences?

Georgia on our operating systems doesn’t support ligatures at all. There is a Georgia Pro font available which does support them but it’s not free. Tisa, the other typeface from the original example above, on the other hand does support common ligatures. Take a look at the figure above. The first line of sets of characters doesn’t have ligatures enabled while the second one does. Notice how the letters “f” and “i”, “f” and “l”, “f” and “f” merge? That’s the typeface designer’s elegant way of avoiding those nasty collisions.

What’s the point of ligatures?

Ligatures were designed to save time, improve the aesthetic aspect of a

text or typeface and to improve legibility. With transcribing books and hand-setting type for printing done and dusted, we're only left with the aesthetic aspect and legibility. Remember how we said that horizontal rhythm affects legibility? Ligatures can play a noticeable part in improving it. Take a look at the figure below. We have two sentences. They're the same for a reason. The first sentence doesn't have ligatures enabled while the second one does. Notice the lines that indicate the ends of the sentences. Notice the difference in the width of the sentences. Ligatures cause that difference. That's why they have an impact on horizontal rhythm, and that's why they have an impact on legibility.



Ligatures help keep the horizontal rhythm consistent.

Types of ligatures

There are many types of ligatures, but when it comes to web typography the two major groups are common ligatures and discretionary ligatures.

Common ligatures
These are ligatures that substitute letter combinations like "ff", "fi", "fl", "ffi", "ffl" and "th". These are enabled by default by most web browsers. If you want to disable them for some particular reason, you can do so by using this:

```
.class {
  font-variant-ligatures: none;
  font-feature-settings: "liga" 0;
}
```

Check out a live example at *betterwebtype.com/book/c10e1*.

Discretionary ligatures

These are non-standard ligatures and usually don’t contribute to legibility. They’re much more stylistic then their common counterparts. Examples are letter combinations of “ch”, “ck”, “ct” and “st” (check the figure below). It’s usually best to avoid using them but when you want to, you can do so like this:

```
.class {
  font-variant-ligatures: discretionary-ligatures;
  font-feature-settings: "dlig";
}
```

Check out a live example at *betterwebtype.com/book/c10e2*.

Discretionary ligatures.

Contextual ligatures
Contextual ligatures usually come with script typefaces. In most cases they combine letters so it looks like they were written with one stroke. If you do use a script typeface, it’s recommended that you enable these ligatures if available. The text set in this typeface will look more genuine (see comparison in figure below). Controlling contextual ligatures is as easy as shown below (if the typeface you’re using supports them):

```
.class {
  font-variant-ligatures: contextual;
  font-feature-settings: "calt";
}
```

Check out a live example at *betterwebtype.com/book/c10e3*.

Contextual ligatures.[54]

Other notes on using ligatures on the web

If you're looking for the quickest and most basic way to enable or disable ligatures, look no further than text-rendering property. It will enable common ligatures and kerning (Chrome and Firefox do this by default).

```
text-rendering: optimizeLegibility; // Enables common liga
```

Check out a live example at *betterwebtype.com/book/c10e4*.

Note that if you set letter spacing to your text it will not feature any ligatures. The browsers will strip them away.

Using ligatures in Sketch

Using ligatures in Sketch is quite simple. You need to select the text in which you want to change your ligatures options, for example a paragraph. Note that you need to select the actual text, not just the text layer. Once you have the text selected, go to *Type* > *Ligatures* and choose:

- *Use default* for the default setting (Sketch has the common ligatures enabled by default)
- *Use none* to disable all ligatures
- *Use all* to enable all ligatures (including discretionary, etc.).

Oh no, my font doesn't support ligatures!

As mentioned earlier in the book, the OpenType format is becoming the new standard but there are still a lot of web fonts out there that don't support ligatures (or any other OpenType features). There's no reason to

panic. If you really like the typeface that you chose but it doesn't support ligatures, don't go and change the typeface just because of that. I recommend taking a look at the most common letter combinations that would be replaced by a ligature. Take a look at those at the micro level. Do the letters collide? Is the collision unappealing? If so, consider changing the typeface. If not, go ahead and continue using the typeface.

Ligatures are less common in sans-serif. typefaces When I first learned about ligatures I wanted to use them everywhere. To show off my newly acquired knowledge I used ligatures at every opportunity. I often ended up disappointed when I found out that my typeface of choice didn't support them. Often those would be the sans-serif typefaces. Here's a short story about why ligatures just aren't that common in sans-serif typefaces and digital fonts in general.

Ligatures were commonly used in typesetting for print machines that were closely related to the one produced by Gutenberg. They were still useful in saving time—instead of finding an "f" and an "i" to put together, a typesetter would simply use the ligature "fi". That changed a lot in the 20th century. Industrialisation had a major part in it. Different printing machines were produced, requiring less manual work. Therefore the time-saving ligatures weren't needed any more. Phototypesetting machines from the 1970s also generally avoided ligatures. The trend was continued into the digital revolution, where the first computers produced didn't support automatic character replacement with ligatures. That led to most digital fonts being produced without ligatures at all. Tobias Frere-Jones's comment of that era is "some of the world's greatest typefaces were quickly becoming some of the world's worst fonts".[55]

As for the sans-serif typefaces, ligatures aren't needed as much as in their serif counterparts. There are fewer collisions such as those we witnessed earlier in those typefaces, so there goes the aesthetic need for ligatures. What remains is the legibility part so you'll still find sans-serif typefaces that support the most common ligatures, mainly "fi" and "ff". This is more common with fonts produced recently, as the font designers from the early ages of the digital revolution simply didn't bother designing something that had no way of getting used.

Let's enable ligatures for our website

There's not a lot to do for our example website after this chapter. The FF Meta font that we decided to use supports common ligatures, so we just

need to make sure they're enabled. We only need to do that for the body text. The font I chose for titles doesn't support ligatures anyway. In general, common ligatures aren't needed for large font sizes (which headings usually are). It would make more sense to use other types of ligatures that are more ornamental in nature—discretionary or contextual come to mind. Not in our case, though.

```
p {
  font-feature-settings: "liga";
  -webkit-font-feature-settings: "liga";
  -moz-font-feature-settings: "liga";
  text-rendering:  optimizeLegibility;
}
```

Our best bet to enable ligatures is to use the font-feature-settings property as it's best supported by browsers, including Internet Explorer 10. I also set the property `text-rendering` to `optimizeLegibility`, so we enable the basic kerning that will make our text easier to read. I did this for paragraphs only, but it's worth doing it for all elements that use relatively small font sizes. See it live at *betterwebtype.com/book/c10*.

Key takeaways

- The simplest way to describe what a ligature is to say it's a combination of multiple letters into a single glyph.
- The origin of ligatures goes way back to the times before the Latin alphabet. Monks that transcribed bibles during the Middle Ages used them as a way to save time.
- Common and discretionary ligatures are the two most common groups. Common ligatures should be used within the body text if available; discretionary ligatures are more ornamental in nature.
- Sans-serif typefaces are less likely to support ligatures.
- Ligatures can help keep the horizontal rhythm in a text consistent, and have a generally positive impact on legibility.

Small caps and figures

We covered ligatures and how they affect legibility, and separate good typographers from average ones, in the previous chapter. We're diving deeper into details of typography with each chapter and small caps and figure styles are definitely on the micro level side of typography.

I still remember my first encounter with small caps. It was a long time ago when I noticed a certain button in Microsoft Word. I clicked it and it turned my text into capitals but something was strange. There was something that was off about that text but I couldn't really put my finger on it. The text seemed lighter than the rest of it and, with that, it stood out from the body text. I didn't understand what was going on so I clicked the same button again to return to normal style. I now know that it was small caps that I had enabled. Actually, it was fake small caps but we'll learn about those later in this chapter.

I also remember how I avoided using fonts that had these strange, old-looking figures. I had no idea what they were, but I avoided them because, to me, they simply looked old-fashioned. Georgia is one of those fonts that uses this style of figures by default, so I never used Georgia. It was the early years of web design, so we could only use the web-safe fonts. I only learned about different styles of figures and what their purpose was once I started studying typography. It turns out there are numerous figure styles and each of them has its own purpose. At the second part of this chapter we'll take a look at lining, old-style, proportional and tabular figures. We'll also have a quick look at ordinals and fractions and how to use them on the web.

All right, let's jump straight in. Let's take a look at two texts side by side.

John is a UX/UI designer. He moved to London, UK in 1996. He's been a designer for 16 years, freelancing for 9 years while he studied HCI (human-computer interaction) design, and the rest as an inhouse full-time designer for various companies. One of which was the famous GNB bank.

John is a UX/UI designer. He moved to London, UK in 1996. He's been a designer for 16 years, freelancing for 9 years while he studied HCI (human-computer interaction) design, and the rest as an inhouse full-time designer for various companies. One of which was the famous GNB bank.

Compare the two paragraphs. One of them has small caps and old-style figures enabled. You can also take a look at the live example at betterwebtype.com/book/c11e1.

Now try to answer the following questions. Check the figure above, a couple of times if you need to.

- Which of the two looks sophisticated?
- Which of the two looks better balanced?
- Do you notice any major differences between the two?

The questions are a bit tricky because you may be able to answer Questions 1 and 2 confidently, but without noticing the major differences that Question 3 is asking about.

To get better in typography, all you really require is knowledge about the whats and the whys, and the right tools. Tools may vary differently and can come in the form of a typeface that has a good support of OpenType features. Or it may be a tool that helps you produce the code and a website with better typography. That’s why we’re learning about details like these. It may seem irrelevant at first, but in typography it’s the smallest details like these that really make a big difference.

Inherent quality is part of absolute quality and without it things will

appear shoddy. The users may not know why, but they always sense it.
—Erik Spiekermann

Small Caps

Small caps are exactly what the name implies—they're smaller versions of capital letters. Being smaller, they're set at the same height and weight as their lowercase counterparts. They're mainly used inside body text because we want them to merge with it better. Take another look at the figure above. Can you see how abbreviations in the first paragraph stand out? We don't want that. We want them to blend in as much as possible. It makes our text seem professional and balanced.

Fake vs real small caps
You can turn any text on the web into small caps with CSS. But most of the time, those aren't real small caps. They're so-called "fake small caps". "*Fake*" sound bad, and it is. CSS simply reduces the size of actual capital letters. So it literally fakes small caps. And in my opinion, these are even worse than using standard capital letters.

Comparing real to fake small caps. Notice how much lighter the fake ones seem to be, compared to their uppercase counterparts. There's no difference in weight with real small caps.

Take a look at the details of these letters. Compare their weight. The "Bob Dylan" on top is set in Tisa Pro which supports small caps as an OpenType feature. Take a close look at the regular capital "B" and the small cap counterpart. The small cap "B" looks heavier on its own but compared with the regular capital, it's perfectly balanced. Even more important, it's perfectly balanced with the other lowercase letters. The "Bob Dylan" at the bottom is set in Georgia which doesn't support small

caps so I forced them with CSS. I produced the so-called fake small caps. Again, compare the two “B” letters. Both, ticker and thinner parts of the letters are completely different and unbalanced. Let’s take a closer look.

The fake small caps “A” is only a resized version of the regular capital “A”, so the thickest part of the letter is clearly narrower than it should be. It remains the same in the real small cap.

Take a look at the figure below. It gets even worse when fake small caps are used in an actual body of text. It makes them stand out even more. For the wrong reasons.

John is a UX/UI designer. He moved to London, UK in 1996. He's been a designer for 16 years, freelancing for 9 years while he studied HCI (human-computer interaction) design, and the rest as an inhouse full-time designer for various companies. One of which was the famous GNB bank.

REAL SMALL CAPS

John is a UX/UI designer. He moved to London, UK in 1996. He's been a designer for 16 years, freelancing for 9 years while he studied HCI (human-computer interaction) design, and the rest as an inhouse full-time designer for various companies. One of which was the famous GNB bank.

FAKE SMALL CAPS

Notice how fake small caps actually stand out from the text: the exact opposite of their purpose.

You can see a live example of real and fake small caps at *betterwebtype.com/book/c11e2*.

How to properly use small caps
Small caps seem to be one of the things that were completely left behind at the start of the digital transition in typography. It's only lately that they started gaining popularity. When it comes to typography, it seems that the web is finally maturing.

Check the typeface
In order to use real small caps, your typeface of choice needs to support them. These typefaces are rare. If they do support them, it means they'll be quite a bit heavier in Kb. When small caps are included, they usually come in the form of an OpenType feature. So check those first. If they're not there, check if there's another font of the same name but with "SC" or "Small caps" added to it. For example, P22 Underground on TypeKit comes with many additional fonts which are different weights of small caps.

Small caps come as separate fonts in the P22 Underground typeface.

Search for a small caps font
Many typefaces don't have a small caps font included, or don't support it as an OpenType feature. In that case, you could still try to find a matching small caps font elsewhere. The emphasis here is on *matching*. Don't use something that's close enough. If you're using Garamond as your body typeface, you need the Garamond small caps font. I once found a "Garamont" font on fonts.com that is based on Garamond and has small

caps fonts included. I examined and compared the two and concluded that they were compatible.

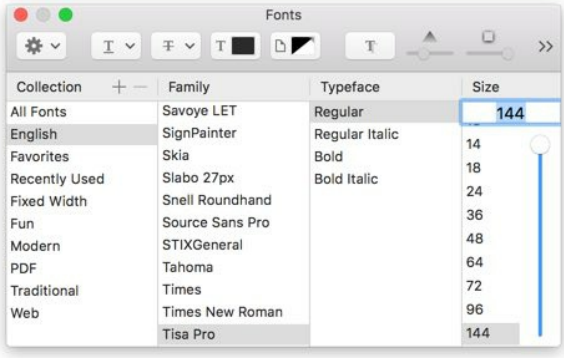
Enabling small caps
If your font supports small caps as an OpenType feature you can enable them by using the following CSS:

```
.class {
  // Change lowercase to small caps, leave capital letter
  font-variant-caps: small-caps;
  font-feature-settings: "smcp";

  // Change all letters into small caps (including capita
  font-variant-caps: all-small-caps;
  font-feature-settings: "c2sc", "smcp";
}
```

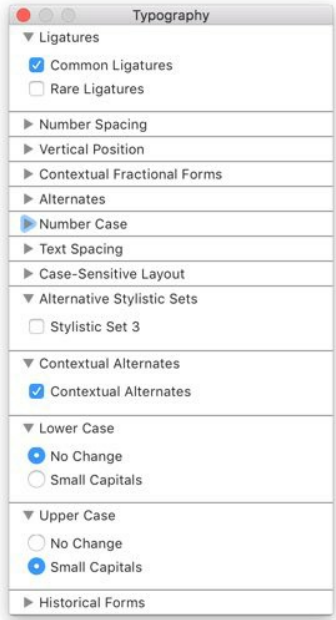
If it doesn’t and you need to use a separate font for that, you need to load that font separately through font-face.

If you can’t find either way to use real small caps, don’t use them at all. Use regular capital letters instead. Just don’t use fake small caps.

Using small caps in Sketch
Sketch doesn’t support OpenType features by default so there’s no way of using them—inside Sketch. But there’s a workaround. Select the text you want to turn into small caps (actual text, not just the layer) and go to *View > Show Fonts*. A window will pop up and your typeface is already selected.

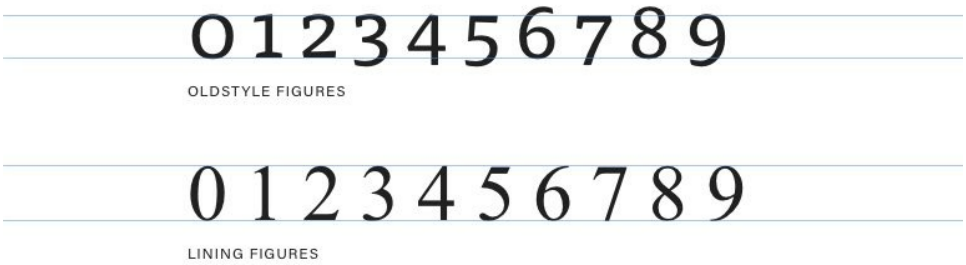


There’s a small button with a gear icon on it at the top left of the window. That’s your access to the OpenType magic in Sketch. Click on it and choose *Typography*. Another window pops up. All the OpenType features are listed there (the ones that are supported by your font).



In our case we want small capitals so we go to the *Upper Case* section and choose *Small Capitals*. This will turn capitals into small capitals. If you want, you can turn lowercase into small capitals. To do that, go to the *Lowercase* section and enable *Small Capitals* there. This will only turn lowercase letters into small capitals, leaving the regular capitals untouched.

Figures



There are two common styles of figures: old-style figures whose bodies match the x-height of the typeface and have ascenders and descenders, and lining figures that match the height of capital letters and don't have ascenders and descenders. Since the digital revolution, lining figures are mostly the standard. I say mostly, because as we saw with Georgia, some typefaces come with old-style figures as their default.

Old-style figures
Old-style figures (commonly called text figures) are quite rare these days. Just like small caps, they were left out when the digital revolution started. They may be called old-style figures and they even seem a bit more classic than their alternative (lining figures), but they have the same purpose as small caps: to blend in with the text so they don't affect the overall balance.

They come in three forms in web typography:

- as a default figure style of the main font
- as an OpenType feature
- as a part of the individual small caps font.

For example, a typeface like Merriweather has old-style figures (commonly abbreviated OSF) as default figures, and FF Meta that we're using for our example website comes with OpenType support for them. Separate small caps fonts come with the old-style figures as default.

Whenever they come as an OpenType feature they can be enabled with this code:

```
.class {
  font-variant-numeric: oldstyle-nums;
  font-feature-settings: "onum";
}
```

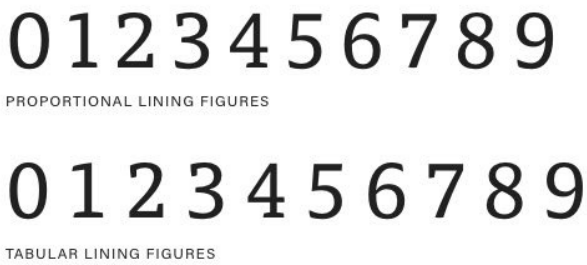
You could describe them as the lowercase version of numbers so ideally they should be used for the body text. Remember when I said you needed to read the text in order to be able to produce good typography? This is where this comes in handy. If you read the text and notice that there are lots of numerical values in the content, you should try to use a typeface that either supports old-style figures as an OpenType feature, or has them as its default figure style. Take a look at a live example of lining and old-style figures at *betterwebtype.com/book/c11e3*.

Lining figures

Lining figures (also called modern figures) are called lining because they extend from the top to the bottom lines of the capital letters. Their height is uniform. With that, they disrupt the balance inside the body text, so should be avoided if possible. They work much better in tables and spreadsheets. If your typeface has old-style figures as its default style but you need to use lining figures for a table, you can enable them with the following code:

```
.class {
  font-variant-numeric: lining-nums;
  font-feature-settings: "lnum";
}
```

There are two version of lining figures: proportional (default) and tabular. Proportional lining figures are perfect for all-caps headings and titles or if the numbers need to be emphasised.

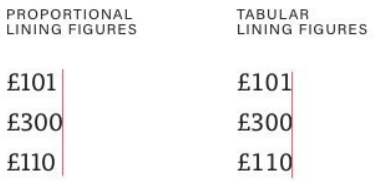


To enable proportional lining figures you need the following code (if they come as an OpenType feature):

```
.class {
  font-variant-numeric: proportional-nums;
  font-feature-settings: "pnum";
```

```
}
```

Tabular figures, on the other hand, are perfect for use in documents heavy with numbers, especially when set in multiple columns and rows. Unlike with proportional figures, all numbers occupy the same space (horizontally looking), no matter the width of the number. Number 1 in this case has more white space around it than most other numbers. This way, even the numbers in separate rows are perfectly aligned.

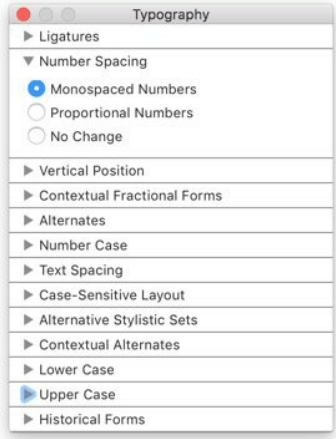


To enable tabular lining figures use this code (if they come as an OpenType feature):

```
.class {
  font-variant-numeric: tabular-nums;
  font-feature-settings: "tnum";
}
```

Take a look at a live example of proportional and tabular figures at *betterwebtype.com/book/c11e4*.

Using different styles of figures in Sketch
We can use different styles of figures in Sketch the same way we used small caps. If the typeface you're using supports them as an OpenType feature, choose the number you want to change the style of. Go to *View* > *Show Fonts* and click on the gear icon in the window that pops up. Choose *Typography* and go to the *Number Case* section to switch between old-style and lining figures. To switch between the proportional and tabular lining figures go to the *Number Spacing* section.



Make sure you take advantage of all these styles to make your typography harmonious and texts easy to read.

Fractions
There's an OpenType feature that can replace fractions we type, for example 1/4, with a typographically correct glyph. Check out the figure below.

If the typeface you're using supports fractions (as an OpenType feature), you can enable them with this code:

```
.fraction {
  font-variant-numeric: diagonal-fractions;
  font-feature-settings: "frac";
}
```

Take a look at a live example at *betterwebtype.com/book/c11e5*.

Using fractions in Sketch
To use fractions in Sketch, go to the magical *Typography* window as described numerous times in this book. Once there, go to the *Contextual Fractional Forms* and choose *Diagonal*.

Ordinals
It's OK to write 1st, 2nd etc. It's better to have those replaced with properly styled ordinals—a standard number with the "st" part as superscript. You can achieve that in CSS, if your typeface supports it through an OpenType feature, like this:

```
.ordinal {
  font-variant-numeric: ordinal;
  font-feature-settings: "ordn";
}
```

You only need to wrap the "st" part with this class, like this:

```
<p>He came in 1<span class="ordinal">st</span>.</p>
```

Using ordinals in Sketch
To use ordinals in Sketch, go to the magical *Typography* window. Once there, go to the *Vertical Position* and choose *Ordinal*.

Let's get our small caps and figures right

Small caps
As with the previous chapter, we'll mostly work with the body text of our example website. First of all, small caps. I'll create a new class and use it accordingly. The FF Meta typeface supports them as an OpenType feature so I'll enable them with that. I'll also add a bit of letter spacing to them as that's what's recommended for capitals in general.

```
.small-caps {
  font-variant-caps: all-small-caps;
  -moz-font-feature-settings: "c2sc", "smcp";
  -webkit-font-feature-settings: "c2sc", "smcp";
  font-feature-settings: "c2sc", "smcp";
  letter-spacing: 0.05em; // Equals 5%
}
```

Now, whenever we need an abbreviation inside the body text it will have the small-caps class applied to it. This is something that should be done easily through any major CMS (WordPress, Drupal).

Figures, fractions and ordinals
FF Meta has old-style figures as the default figure style so we don't need to do anything to enable them for our body text. I did my research at the beginning, so I knew I wanted a typeface like that and I also knew that

there wouldn’t be a need for lining figures.

FF Meta also supports fractions and ordinals as OpenType features so I’ll create classes for those too. Again, these classes must be available through whatever CMS we’ll use for managing content. The person doing that also needs to be educated properly about when to use which class and why. Alternatively, we could create a JavaScript snippet that would detect these instances and apply the appropriate classes as needed. Let’s create a class for fractions first:

```
.fraction {
  font-variant-numeric: diagonal-fractions;
  -moz-font-feature-settings: "frac";
  -webkit-font-feature-settings: "frac";
  font-feature-settings: "frac";
}
```

And another one for ordinals (notice the vendor prefixes; it’s still recommended to use them for better support):

```
.ordinal {
  font-variant-numeric: ordinal;
  -moz-font-feature-settings: "ordn";
  -webkit-font-feature-settings: "ordn";
  font-feature-settings: "ordn";
}
```

We’re all set to use small caps, fractions and ordinals on our website and we don’t need to think about the figure style. Let’s add some examples to our page and apply the classes accordingly. Check the example website at *betterwebtype.com/book/c11*.

Key takeaways

- Small caps are a smaller version of capital letters, best used for abbreviations in body text. Their purpose is to make them blend in with the body text better, so they don’t stand out and disrupt the rhythm.

- The most common way to use small caps is through an OpenType feature. Alternatively they come as a separate font.
- “Fake” small caps are resized regular capital letters so they have an inadequate visual weight compared to other letters. They should be avoided at all costs.
- Figures come in two main styles: old-style (or text style) and lining style (also known as modern style). Lining figures are further separated into proportional and tabular figures. Old-style figures should ideally be used for body text.

Punctuation

This chapter might come as a bit of a surprise to you. The title is self-explanatory—yes, this is a chapter about punctuation. I was surprised, as you probably are, at the moment when I first read about this. Before that, I had never thought that punctuation is a part of typography. I didn't even know that there were different kinds of quotation marks or dashes of different widths. I only learned about the dashes after I read *The Elements of Typographic Style* by Robert Bringhurst. Before that, I had always thought of punctuation as something that would be a part of an English class. As it turns out, most people get punctuation (from a typographic point of view) completely wrong.

I believe that's for two reasons. We're not taught about typographically correct punctuation in school. Yes, teachers keep reminding us when to use a comma and some other basic punctuation marks, but that's it. The other reason is that all these punctuation marks are hidden behind keyboard shortcuts that are impossible to remember. QWERTY keyboards are descendants of a poor design that eventually turned into a standard. There's only room for one type of quotation marks on the keyboard—the straight ones. The curly ones are not present. The same goes for the dashes. Two of them are drawn on a key on the keyboard, but there's a lot more hidden behind shortcuts that include that key.

We're still in the territory of micro typography in this chapter. I'd argue it's as micro as it gets. And as I always mention in these micro typography chapters, it may not seem a big difference at first, but typography is all about details. It's up to you how much time you're willing to spend exploring them.

Dashes

If you're like I once was, you probably use only one dash: the hyphen (-), for everything. We can use that one symbol for anything in coding. A minus, a dash connecting two words, a replacement for the word "to". Typographically looking, there are different dashes for all these uses. There are five different dashes altogether and most fonts have at least three. The mentioned hyphen (-), the em dash (—) and the en dash (–). There's usually a dedicated symbol for the minus (-) as well, but it's not considered a dash.

Minus and the three major dashes

The em dash
The em dash is the widest of them all. It's probably the one that is most misused as well. Its width matches the width of one em, hence the name. The em dash can be used as an indicator of speaker in a narrative. —"Why is typography important?" he enquired. A more common use of the em dash is to indicate the author after a quote, like so:

This is a famous quote
—Author

The em dash should ideally be followed by a thin space (yes, there are different spaces too, we'll cover that later).

Here's how you get it right in HTML:

```
<blockquote>
  <p>This is a famous quote.</p>
  <footer>&mdash; Author name</footer> // Notice t
</blockquote>
```

The em dash can also be used as a replacement for a comma. Similarly, it indicates a break to the reader, maybe a bit longer than a comma. It is commonly used as an indicator of an aside or an afterthought like this: "use em rules—those are the longest of the three horizontal lines—to convey a stronger break than would be covered by a pair of commas."

The en dash
The en dash width is exactly half of the em dash width, and it also matches the width of another typographic unit—the en. Again, that's where it gets

its name—typography is very practical in this regard. As is the case with the em dash, the en dash gets misused a lot too. Its main purpose is to replace the word “to”, so instead of 6 to 5 p.m., we can write 6–5 p.m. Notice the lack of spaces on both sides of the en dash.

If you’re using the word “from” to start indicating a range, use the word “to” and not an en dash to connect the two values. Correct: from 1939 to 1945. Wrong: from 1939–1945. An en dash needs to be used for indicating a range between any numerical values (for example: pages 33–40), and also non-numerical values (for example: Units A–D).

An en dash must also be used when denoting a connection or contrast between two words (for example: republican–democratic split). Don’t replace an en dash with two hyphens (--) or the em dash with three hyphens (---). It’s an old typewriter habit and it’s wrong. Make sure you use typographically correct dashes.

Here’s how to use the en dash in HTML:

```
<p>5&ndash;6PM</p> // No spaces
```

Hyphen
And last, we have the hyphen, the symbol that is mistakenly used instead of the em and the en dashes and the minus symbol. It’s the default symbol you get when you press the key with the “two dashes” on it (that’s the way people most commonly describe that key). Its width is a quarter of the em dash. It’s primarily used for hyphenating words, hence the name, and for connecting two or more words together: fine-tuned, rebel-backed, five-dollar bills etc.

Quotation marks

Dumb and smart quotes
Believe it or not, just like there are with dashes, there are more than single and double quotation marks. The straight ones on your keyboard are commonly referred to as dumb quotes, and are commonly used as inch marks as well. Smart or curly quotes are the typographically correct

quotation marks for indicating a quote. There’s no use for smart quotes in coding. But you should use them whenever you’re quoting someone or when you want to draw attention to an unusual term.

Typing curly quotes is a bit tricky. They’re not present on any keys of the keyboard, so it takes some time to remember the shortcut. Some writing tools/platforms (Medium, DropBox Paper) try their best to automatically replace the dumb quotation marks with the curly ones. Both tools mentioned are quite good at it. Here’s how to get the curly quotation marks correct in HTML:

```
<p>&ldquo;He left yesterday.&rdquo;</p> // Double quot. ma
```

or

```
<p>&lsquo;He left yesterday.&rsquo;</p> // Single quot. ma
```

Here’s an easy way to remember which is which: l/r (means left/right), d/s (means single/double), quo (quotation mark).

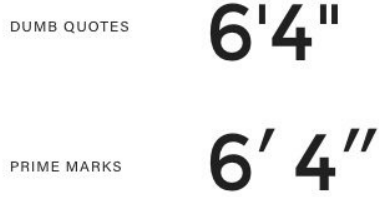
```
& + l/r + d/s + quo + ;
```

And here are the keyboard shortcuts. My suggestion is: try to use them until you remember them, or use a tool that does a good job at replacing these marks as needed. I usually write content that will end up on a website in one of those tools and then copy it into HTML. That way, I avoid the dirty work of manually replacing dumb quotation marks with the smart/curly ones.

Apostrophe
It’s equally wrong to use a single dumb quote in place of an apostrophe. Use the single right quotation mark for that: ’. Let’s have a look at another example where we use both quotes and an apostrophe.

```
<p>&ldquo;I don&rsquo;t want to go&rdquo;</p>
```

Dumb quotes and prime marks
The “dumb quotes” are better used when writing about measures like inches, feet etc., like this:



But they’re not exactly the typographically correct punctuation for that either. There are so-called prime marks that should be used to indicate those measurement units.

```
<p>He&rsquo;s 6&prime;4&Prime;</p>
```

The only difference between the prime marks indicating feet and inches is the capital “P” for the double prime, so be careful when using these in HTML.

So what are the dumb quotation marks for? Good question. They’re for code. That’s it; don’t use them for anything else.

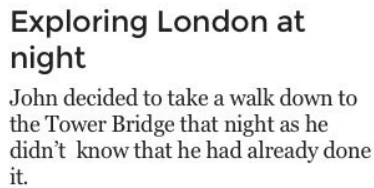
Other punctuation symbols you should use

Spaces

Hair space
Can be used alongside the dashes but are not mandatory. Hair space is the thinnest space available.

Thin space
Also commonly used in combination with dashes. Their width is usually equal to 1/5 or 1/6 of an em.

Non-breaking space
The non-breaking space can be very useful in web design. I use it all the time. It's aesthetically unappealing to have a line break in a text that pushes only one word into a new line. Here are two examples that happen on the web all the time.



These are called widows. A widow is considered poor typography because it leaves too much white space between paragraphs or at the bottom of a page. This interrupts the reader's eye and diminishes readability.[56] It's nearly impossible to completely avoid these on the modern, responsive web. But a clever way to tackle these is to put a non-breaking space between the last word in the paragraph and the word before it. Like this:

```
<h3>Exploring London at night</h3>
<p>... as he did not know that he has already done i
```

Instead of the word "it" being the only one pushed onto a new line, both "done" and "it" are. It's not perfect but it's much better than "it" being on a new line alone.

Mathematical punctuation
Using the asterisk (*) for indicating multiplication or the forward slash (/) for indicating a division is typographically wrong. The dimension symbol (×) must be used for multiplication and the division symbol (÷) for… well, division. Also don't forget to use the correct minus symbol (−) which is wider and thicker than the hyphen.

Typography has a lot of symbols, each with a dedicated intended use. Many of them were omitted from our keyboards simply because there's not enough space to accommodate them all. Eventually, they started disappearing from our daily use. They're being replaced by similar symbols. Dumb and smart quotes is a great example. Typography is bastardised these days, in most cases at least. In order to convey our message clearly and separate ourselves from the mediocrity and bastardisation, we must use the correct punctuation. Minus is not the same as hyphen, ellipsis isn't equal to three fullstops, apostrophe isn't the same as a single dumb quote. Don't forget about spaces of different widths that should be matched with some of these symbols instead of the default space. There's the non-breaking space which literally ties two words together even if there would be a line break between the two. Here's a list of all the symbols you should use to make your texts typographically correct.

Symbol	Name	HTML code
−	Minus	`&minus;`
–	En dash	`&ndash;`
—	Em dash	`&mdash;`
…	Ellipsis	`&hellip;`
÷	Division	`&divide;`
×	Dimension	`&times;`

	Hair space	` `
	Thin space	` `
	Non-breaking space	` `
’	Apostrophe	`&rsquo;`

Let’s make sure our punctuation is correct

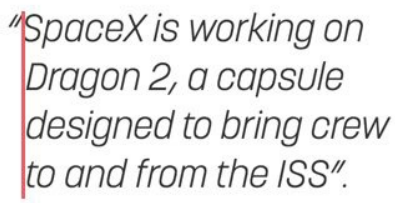
To be completely honest, there’s not much we can do about punctuation. We could write JavaScript functions that go and replace the hyphens with the em dashes when needed but that’s a lot of work. The best way to get punctuation typographically correct is to learn about it and then use it. Make yourself a cheatsheet or something you can always refer to.

I’ve used correct typographic punctuation so far in our example website but I now want to add quotation marks to our quote. We need to use the correct, smart quotation marks remember? Let’s add them to our HTML.

```
<blockquote class="big-quote">
  &ldquo;SpaceX is working on Dragon 2, a capsule designe
</blockquote>
```

The code above results in this:

There’s something wrong in the screenshot above. Do you notice it? Yup, it’s the quotation marks disrupting the horizontal rhythm by “pushing” the first line inwards. Let’s fix that with our hack where we change the indenting of the paragraph. I add `text-indent: -0.5em;` to the `.big-quote` class and the result is this:



Better, but not there yet. Indenting the paragraph for -0.5em is too much in this case. Let’s try -0.4.

Yup, that’s it. The letter “S” in the first line is now perfectly aligned with the letters from the lines below. Sweet!

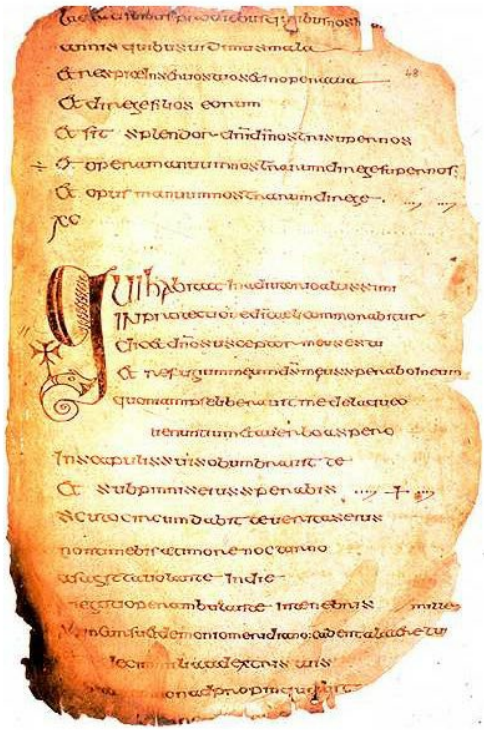
Key takeaways

- All the effort we put into producing better web typography is in vain if we don’t finish it off with typographically correct punctuation.
- There are five types of dashes and there are at least three in each typeface. The em dash, the en dash and the hyphen. Each has its own intended use.
- Straight quotation marks (also known as dumb) are typographically incorrect, and they should only be used for coding. To indicate quotations, use curly quotation marks (also known as smart).
- Don’t use the single straight quotation mark in place of an apostrophe. Use a single curly quotation mark.

Dropcaps

History of dropcaps

Dropcaps—large letters at the beginning of paragraphs—have been in use since the very early manuscripts, produced in the 4th century. They are a version of initial caps, the name coming from Latin, “initialis” literally meaning “standing at the beginning”.[57] A dropcap is usually dropped several lines into the text, hence the name. Early books, written by hand as well as early printed books, were commonly extravagantly ornamented. In some extreme cases, initial caps would even occupy a whole page of a manuscript.



But their purpose back then wasn’t solely ornamental. They helped improve the readability of the text. They were used to mark beginnings of important segments in manuscripts, so a reader could easily memorise them (or remember where they left off), especially because early writing in Latin didn’t use punctuation and spacing to separate sentences. It’s

quite hard to believe this now, when we're so used to them. When Gutenberg produced the first ever printed bible, he closely imitated the dropcaps from manuscripts. But the process changed a bit: the body text was printed on the pages first with blank spaces, where the initial caps were later added—drawn by hand by a scribe. Later on, even the initials were produced as wooden or metal type blocks. The need for initials diminished after the start of standardised punctuation but they remained in most printed work for centuries. Their purpose changed to solely ornamental. And boy, were they ornamental. Take a look at the following example of an initial "L" from a Romanesque bible.

With no need for dropcaps to indicate the start of important parts of a text on the web, we're left with their ornamental purpose alone. Let's take a look at how we can go about implementing these.

Dropcaps on the web

Even though their nature on the web is solely ornamental, I think they can add that extra touch to a website. I used them on the Better Web Type website and on an earlier version of my personal website. There are a few

ways to implement dropcaps on the web, but we'll only look at three. They all come with advantages and disadvantages. Let's take a closer look.

CSS & HTML

This is a very simple approach to including dropcaps in our design. This example closely follows the one developed by Chris Coyier described on CSS-tricks.[58] The idea is that we wrap the initial letter in a span and add a class to it. This way we can style it as we please.

```
<p><span class="dropcap">H</span>ello there. My name is M
```

We can then apply the following to that initial letter:

```
.dropcap {
  float: left;
  font-size: 35px;
  line-height: 30px;
  padding-right: 4px;
}
```

This approach does the job quite well, but it does require that extra markup to make it possible. The example code above would look something like this:


Hello there. My name is Matej and I'm a designer. I live in London but I'm originally from Slovenia. I currently work as a Lead UX/UI designer at Auto Trader—UK's largest online marketplace for anything automotive. I spend my free time working on projects like Better Web Type and reading lots of books.

A funny fact about me is that I'm colour blind and I used to be really frustrated about it. At some point I realised that it's more of an advantage than a disadvantage.

Take a look at the live example at *betterwebtype.com/book/c13e1*.

The CSS-only way

Another CSS-only way to get the same result is to use the magic of CSS3. There's a first-letter pseudo class we can use. The code is very similar to the previous example but we don't need that extra span any more.

```
<p>Hello there. My name is Matej and I’m a designer. I li
```

and the CSS is as simple as:

```
p:first-child:first-letter {
  float: left;
  font-size: 35px;
  line-height: 30px;
  padding-right: 4px;
}
```

Take a look at the live example at *betterwebtype.com/book/c13e2*.

The disadvantage of both of these two approaches is that they need to be manually styled by pushing pixels (literally). We need to properly place the dropcap so it sits at the beginning of the paragraph as it should: a few lines deep, and perfectly aligned with the outer borders of the text block. We can try to use units other than pixels (ems come to mind) but we would always end up with situations that need manual adjustments. Apply responsive web design to it, and we may get a lot of extra work to get these looking right on different-sized screens. Well, there is another way, but it includes using a little help from JavaScript.

JavaScript way
In theory, we could use the CSS initial-letter property for a better way to get dropcaps into our paragraphs. With it, we can simply set how many lines deep we want the dropcap to be. If we set it to two, the dropcap would be two lines deep. We can easily change that to three lines for larger screens, for example. But there’s a catch. It’s not fully supported by the major web browsers yet. OK, no worries, we can still use something very similar.

Dropcap.js is a JavaScript snippet that can handle that quite well. We still need to use extra markup to use it but it does produce the best results so far. My recommendation is to use it until the initial-letter property gains wider browser support (which may take a while). Let’s take a look at an example. According to the Dropcap.js documentation, we need to wrap the initial letter in a span with a class “dropcap”.

```
<p>
  <span class="dropcap">H</span>ello there. My name is Ma
</p>
```

We need to load the Dropcap.js script (obviously).

```
<script src='./dropcap.min.js'></script>
```

And we're ready to use it. We just need to add the following inside the `<script></script>` tags.

```
var dropcaps = document.querySelectorAll(".dropcap");
window.Dropcap.layout(dropcaps, 3);
```

The example above will apply a dropcap that is three lines high for every class .dropcap it finds on the page. Neat!

Take a look at the live example at *betterwebtype.com/book/c13e3*.

But wait, there's more we can do with Dropcap.js. Let's take a look at how the dropcap function is defined.

```
window.Dropcap.layout(dropcapRef, heightInLines, baselinel
```

Take a look at the baselinePos parameter at the end of the code example above. Its default value is equal to the heightInLines parameter, meaning that the dropcap will be aligned with the third line of text (in our example so far). If we change that value to 1, the dropcap will become raised. It sticks out of the body text. This gives us another interesting version of initial caps we can use.

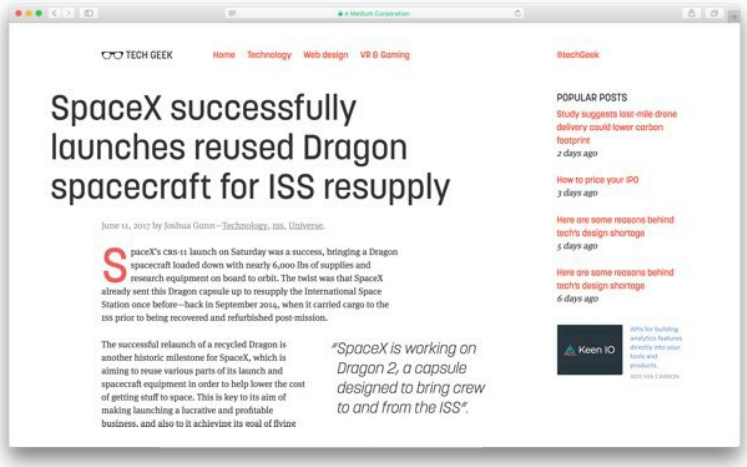
Hello there. My name is Matej and I'm a designer. I live in London but I'm originally from Slovenia. I currently work as a Lead UX/UI designer at Auto Trader—UK's largest online marketplace for anything automotive. I spend my free time working on projects like Better Web Type and reading lots of books.

Take a look at the live example at *betterwebtype.com/book/c13e4*.

Let's add dropcaps to our website

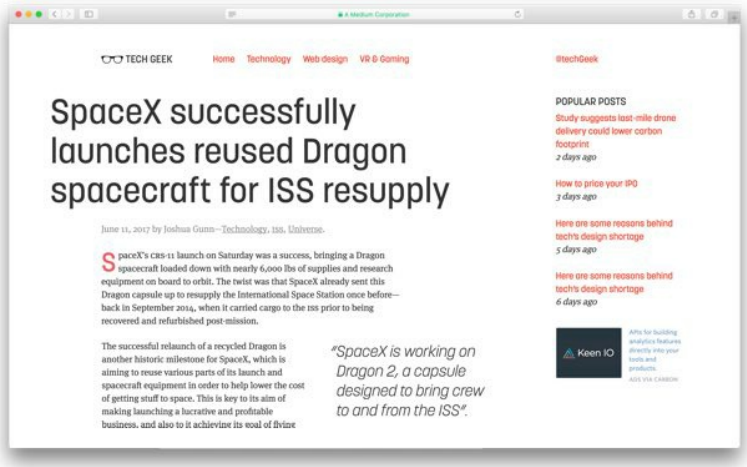
Well, let's see if they fit well first. The page layout of our example website is quite dynamic already so there's a chance that a dropcap simply won't fit. I recommend you pay extra attention to make sure the dropcaps add to the aesthetic value of the website, simply because their purpose is ornamental. If they don't, you probably shouldn't use them.

Let's use Dropcap.js to give the dropcaps a try so we can use the same code as described above. Let's try a dropcap three lines high first.



It's way too large. I'm starting to think this page won't be suitable for

dropcaps. Let’s try the height at two lines.



It’s better, but I’m still not convinced. I’m concerned about the extra visual weight on the page and I don’t think the dropcap achieves its ornamental purpose. In this case it adds visual noise so I decide not to use it at all. That’s the sort of thinking that should go through your mind when considering using dropcaps. If it doesn’t do its job, don’t use it.

With this exploration of dropcas we conclude our process. We started out with content and built a responsive, mobile-first, website around it. We made sure our body typeface matches the content and we picked a heading typeface that is contrasting. We used the modular scale to assign sizes to different parts of the text content and we built a 16-column grid that uses the responsive canvas as a whole. We even made sure we use ligatures in our body text, small caps for abbreviations and old-style figures for numbers. We’ve come a long way and our final product is a dynamic, content-focused website. It was an imaginary example but visitors of such a website would surely enjoy reading its content. Check out our final example website at *betterwebtype.com/book/c13*.

Key takeaways

- Dropcaps are a style of the so-called “initial caps”, where the letter is dropped into the body of text by a few lines.
- Dropcaps have been in use since the early manuscripts, where they had a functional role besides the ornamental one. It helped people navigate and read the book as there was no standardised punctuation and spacing between letters in early texts written in Latin.
- Dropcaps can have an aesthetic effect on the web too. The simplest and most consistent way of using them at the moment is to use Dropcap.js.

Conclusion

In the age when everyone is involved in typography, everyone should get better at it. Whoever is involved in producing typography for a certain medium needs to improve their skills and invite others to do the same. On the web, that's web designers and web developers. Maybe even content managers (or whatever they're called these days) for large-scale websites. Whatever your role, I hope this book was helpful. Whether you learned a lot or refreshed your typographic knowledge, you're now ready to be a Better Web Type advocate. Tell others about the book, tell them about the free course. Let's get as many people as possible to learn about typography. It's time to spread the word and tell people that typography is not about fonts alone and it's not something trivial. It's not something we should take for granted. Typography needs special attention, just like other parts of the web design process. And this task is now yours too. You read the book and you're now ready to share your newly acquired knowledge with others. I'm glad to have you on my side. Let's change the web for the better. Once and for all, let's improve web typography and let's convince others to follow. Designers and developers, together we *can* produce *better web typography for a better web*.

Appendix A: What's next?

Feedback
I really hope you enjoyed the book but, more importantly, I hope you learned something new. Something you can apply to your work and see an immediate effect. You are now ready to contribute to a better web by using your newly acquired web typography skills. If you liked the book (or even if you didn't), I would really appreciate your feedback in the form of a review. Honest feedback from course students was what made it better, and turned something that started out so simple into a fully fledged book.

You can review the book at:

- Amazon: betterwebtype.com/amazon
- Goodreads: betterwebtype.com/goodreads

(These URLs will redirect you to the correct ones on the corresponding platforms.)

Help others recognise the value of this book. I'll be truly grateful for any feedback.

Stay tuned
Better Web Type started out as a short, 10-lesson email course but people loved it so much that they demanded a book. The project keeps growing and there's more cool stuff coming. I'll be launching a blog shortly after the release of this book. So if you're not yet subscribed to Better Web Type, you can do so at *betterwebtype.com*. I'll be diving even deeper into web typography with even more articles, experimental stuff and more freebies. Make sure you don't miss out on all the cool things yet to come.

Invite me to your local meetup
I got the idea for this project when I was giving a talk at a CSS meetup. I realised back then that nobody talks about web typography with the people who actually build websites—the web developers. There were a considerable number of them at that particular meetup, and they were completely consumed by its content. I believe that I'm responsible for a few more typography geeks in the world from these talks alone. People keep coming up to me after the meetups to tell me how fascinated they are

and how they can’t wait to dig deeper.

It’s an amazing feeling to inspire someone like that, so I’m always glad to join a group of developers or designers (or even better—both) to talk about typography. I can easily join meetups organised in the UK, but it would require a bit more effort for the ones overseas. Get in touch at hello@matejlatin.co.uk and I’m sure we can work something out.

Appendix B: About the author

Matej is a designer driven by a passionate curiosity and a relentless desire to build things. That, among his search for simple and usable design, drove him from his small seaside hometown in Slovenia, all the way through Germany, Luxembourg and finally to the United Kingdom. He lives in London, where he currently works as a Lead UX/UI Designer at Auto Trader UK—Britain's largest online marketplace for anything automotive. His journey into the world of design started when he joined an after-school class called "Web Design" at the age of 13. By the time of high school, he was working on freelance projects with local clients, earning an extra buck which he mostly spent on video games, heavy metal music albums and candy. He runs a personal blog at *matejlatin.co.uk* where he regularly shares his stories and articles based on his professional work—mostly design. His work and articles were featured in places like CSS Tricks, Smashing Magazine, Codrops, Net magazine, Creative Bloq and Web Designer magazine.

Notes

1. *Typography*, [website], 2017, https://en.wikipedia.org/wiki/Typography, (accessed 3 May 2017).
2. Definition of typography (noun), *Oxford Advanced Learner's Dictionary*, [website], 2017, http://www.oxfordlearnersdictionaries.com/definition/english/typogr (accessed 3 May 2017).
3. *Digital vs Traditional Media Consumption Summary*, 2014, http://insight.globalwebindex.net/hs-fs/hub/304927/file-1414878665-pdf/Reports/GWI_Media_Consumption_Summary_Q3_2014.pdf, (accessed 3 May 2017).
4. J. Tschichold, *Clay in a Potters Hand*, **as cited in *The Form of the Book*, Washington, Hartley & Marks Publishers Inc., 1948.
5. R. Bringhurst, *The Elements of Typographic Style*, Vancouver, Hartley & Marks Publishers Inc., 2012, p. 196.
6. J. Tschichold, *The New Typography*, Los Angeles, University of California Press, 2006, p. 66.
7. J. Tschichold, *The New Typography*, Los Angeles, University of California Press, 2006, p. 67.
8. R. Bringhurst, *The Elements of Typographic Style*, Vancouver, Hartley & Marks Publishers Inc., 2012, p. 24.
9. J. Shoaf, *The 10 Most Popular Web Fonts of 2015 (And Fonts You Should Consider Using Instead)*, [website], 2016 https://www.typewolf.com/blog/most-popular-fonts-of-the-year, (accessed 10 May 2017).
10. S. Coles, *Massimo Vignelli's A Few Basic Typefaces*, [web blog], 2016, https://fontsinuse.com/uses/14164/massimo-vignelli-s-a-few-basic-typefaces, (accessed 10 May 2017).
11. R. Bringhurst, *The Elements of Typographic Style*, Vancouver, Hartley & Marks Publishers Inc., 2012, p. 24.
12. *What's OpenType?* [website], https://www.typography.com/techniques/opentype/#benefits, (accessed 10 May 2017).
13. S. Mcbride, *Font Events: Controlling the Flash of Unstyled Text*, [web blog], 2010, https://blog.typekit.com/2010/10/29/font-events-controlling-the-fout/, (accessed 11 May 2017).
14. Z. Leatherman, *Flash of Faux Text—Still More on Font Loading*, [web blog], 2015, https://www.zachleat.com/web/foft/, (accessed 11

May 2017).
15. A. Haley, *Text v. Display*, [website], https://www.fonts.com/content/learning/fontology/level-1/type-anatomy/text-v-display, (accessed 12 May 2017).
16. I. Strizver, *Print vs Web fonts: What's the difference?*, [website], 2013, https://creativepro.com/print-vs-web-fonts-what-s-the-difference/, (accessed 12 May 2017).
17. D. Fadeyev, *Please Stop "Fixing" Font Smoothing*, [web blog], 2012, http://usabilitypost.com/2012/11/05/stop-fixing-font-smoothing/, (accessed 8 June 2017).
18. J. Nielsen, *Serif vs. Sans-Serif Fonts for HD Screens*, [web blog], 2012, https://www.nngroup.com/articles/serif-vs-sans-serif-fonts-hd-screens/, (accessed 8 June 2017).
19. K. Thomas, *The Serif Readability Myth*, [web blog], 2013, http://asserttrue.blogspot.co.uk/2013/01/the-serif-readability-myth.html, (accessed 8 June 2017).
20. *FontFaceNinja*, [website], 2016, https://fontface.ninja/, (accessed 12 May 2017).
21. T. Brown, *Web Font Specimen*, [website], 2009, http://webfontspecimen.com/, (accessed 15 June 2017).
22. Typography, [website], https://material.io/guidelines/style/typography.html#typography-other-typographic-guidelines, (accessed 9 June 2017).
23. R. Parikh, *Distribution of Word Lengths in Various Languages*, [website], http://www.ravi.io/language-word-lengths (accessed 9 July 2017).
24. *Baskerville Tercentenary*, [website], 2007, http://typefoundry.blogspot.co.uk/2007/01/baskerville-tercentenary.html, (accessed 9 July 2017).
25. *Gibson from Canada Type*, [website], https://typekit.com/fonts/gibson/details/gibson-light, (accessed 13 May 2017).
26. T. Brown, *A Pocket Guide: Combining Typefaces*, 2016, https://typekit.files.wordpress.com/2016/04/combiningtypefaces.pdf, (accessed 9 June 2017).
27. R. Carter and B. Day, *Typographic Design: Form and Communication*, Hoboken, John Wiley & Sons Inc., 2007, p. 88.
28. J. Brownlee and E. Morris, *How Typography Shapes Our Perception of Truth*, [web blog], 2015, https://www.fastcodesign.com/3046365/errol-morris-how-typography-shapes-our-perception-of-truth (accessed 9 June 2017).
29. M. Butterick, *Letterspacing*, [website],

http://practicaltypography.com/letterspacing.html (accessed June 9 2017).
30. *Lettering.js A jQuery Plugin for Radical Web Typography*, [website], http://letteringjs.com/, (accessed 9 June 2017).
31. *Dos and don'ts on designing for accessibility*, [website], 2016, https://accessibility.blog.gov.uk/2016/09/02/dos-and-donts-on-designing-for-accessibility/ (accessed 9 June 2017).
32. *Hanging Punctuation*, [website], 2017, https://en.wikipedia.org/wiki/Hanging_punctuation, (accessed 11 May 2017).
33. C. Coyier, *Hanging Punctuation*, [web blog], 2017, https://css-tricks.com/almanac/properties/h/hanging-punctuation/, (accessed, 9 June 2017).
34. M. Latin, *Gutenberg—A Meaningful Web Typography Starter Kit*, [website], 2016, http://matejlatin.github.io/Gutenberg/, (accessed 9 June 2017).
35. I. Yates, *How to Establish a Modular Typographic Scale*, [web blog], 2013, https://webdesign.tutsplus.com/articles/how-to-establish-a-modular-typographic-scale--webdesign-14927, (accessed 9 June 2017).
36. T. Brown, *Modular Scale*, [website], 2016, http://www.modularscale.com/, (accessed 18 May 2017).
37. *Golden ratio*, [website], 2017, https://en.wikipedia.org/wiki/Golden_ratio, (accessed 24 May 2017).
38. *Sketch-Plugin: Typographic Scale*, [website], 2014, https://github.com/automat/sketch-plugin-typographic-scale, (accessed 9 June 2017).
39. R. Bringhurst, *The Elements of Typographic Style*, Vancouver, Hartley & Marks Publishers Inc., 2012, p. 10.
40. N. Smith, *960 Grid System*, [website], https://960.gs/, (accessed 9 June 2017).
41. K. Vinh, *Subtraction.com*, [web blog], 2017, https://www.subtraction.com/, (accessed 9 June 2017).
42. M. Vignelli, *The Vignelli Canon*, Baden, Lars Muller Publishers, 2010, p. 72.
43. J. Tschichold, *The New Typography*, Los Angeles, University of California Press, 2006, p. 68.
44. *Display aspect ratio*, [website], 2017, https://en.wikipedia.org/wiki/Display_aspect_ratio (accessed 24 May 2017).
45. *Screen Resolution Statistics*, [website], 2014,

http://www.rapidtables.com/web/dev/screen-resolution-statistics.htm, (accessed 25 May 2017).
46. K. Joseph, *Flexbox Grid*, [website], http://flexboxgrid.com/, (accessed 9 June 2017).
47. *Mobile Share of Organic Search Engine Visits in the United States from 3rd Quarter 2013 to 1st Quarter 2017*, [website], 2017, https://www.statista.com/statistics/297137/mobile-share-of-us-organic-search-engine-visits/, (accessed 9 June 2017).
48. M. Woolf, *Almost Every Smartphone and Tablet Browses the Web in Portrait Mode*, [web blog], 2015, http://minimaxir.com/2015/03/portrait/, (accessed 9 June 2017).
49. Z. Liew, PX, *EM or REM Media Queries?*, [web blog], 2016, https://zellwk.com/blog/media-query-units/, (accessed 9 June 2017).
50. *Typographic ligature*, [website], 2017, https://en.wikipedia.org/wiki/Typographic_ligature, (accessed 9 June 2017).
51. *Talk:Ampersand*, [website], 2017, https://en.wikipedia.org/wiki/Talk%3AAmpersand, (accessed 9 June 2017).
52. A. Haley, *Ampersand*, [website], https://www.fonts.com/content/learning/fontology/level-1/type-families/ampersand, (accessed 12 July 2017).
53. *Typographic ligature: History*, [website], 2017, https://en.wikipedia.org/wiki/Typographic_ligature#History, (accessed 9 June 2017).
54. *Syntax for OpenType features in CSS*, [website], https://helpx.adobe.com/typekit/using/open-type-syntax.html, (accessed 9 June 2017).
55. *Hoefler Text*, [website], https://www.typography.com/fonts/hoefler-text/overview/, (accessed 9 June 2017).
56. I. Strizver, *Rags, Widows & Orphans*, [website], https://www.fonts.com/content/learning/fontology/level-2/text-typography/rags-widows-orphans, (accessed 9 June 2017).
57. *Initial*, [website], 2017, https://en.wikipedia.org/wiki/Initial, (accessed 9 June 2017).
58. C. Coyier, *Drop Caps*, [web blog], 2017, https://css-tricks.com/snippets/css/drop-caps/, (accessed 9 June 2017).