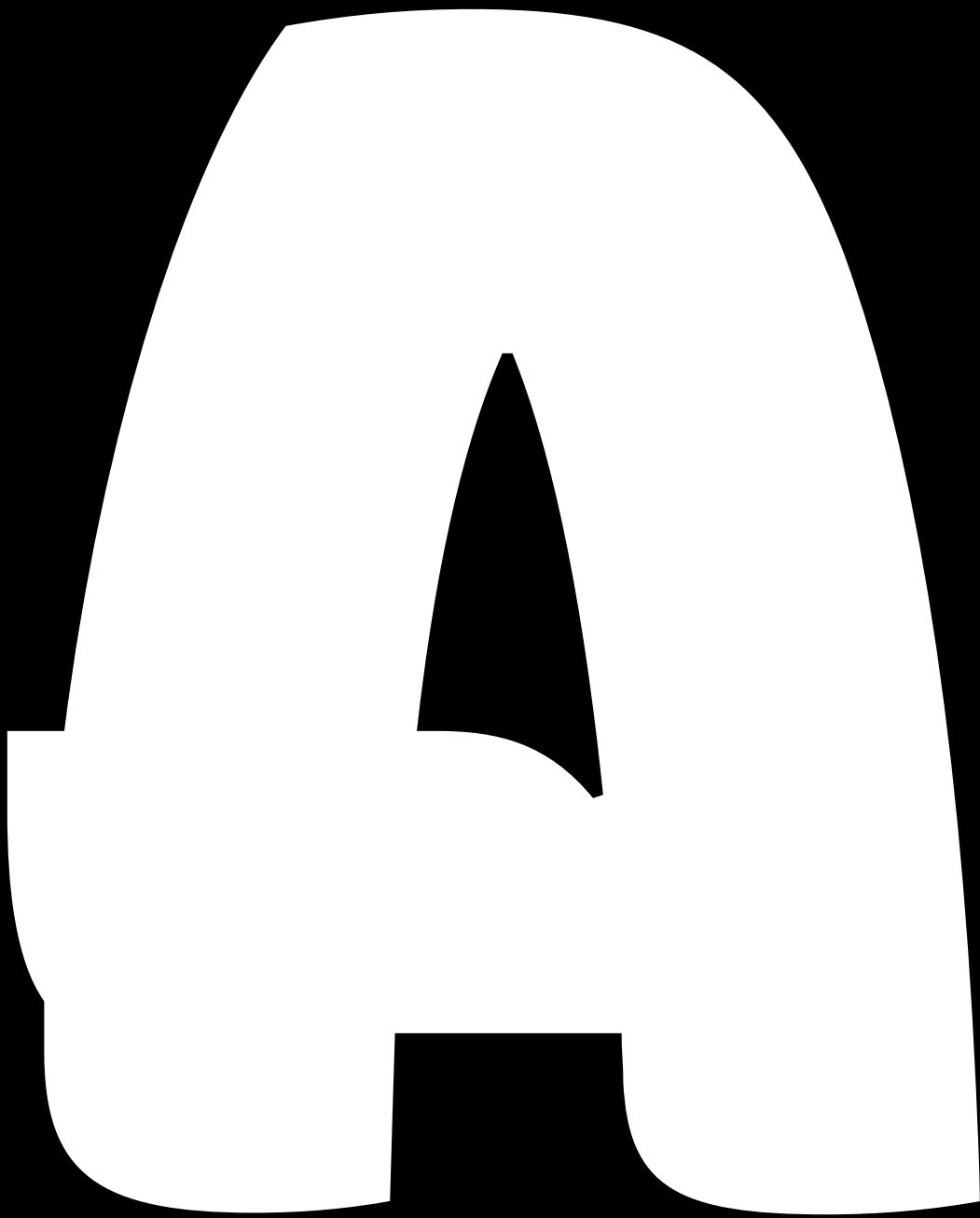


Recursive Sans & Mono
by Arrow Type



MONO

0.0

CASL

0.0

wght

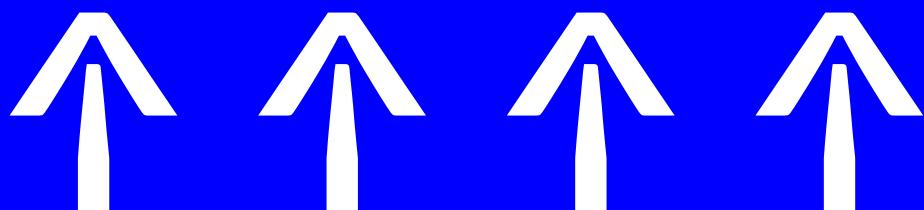
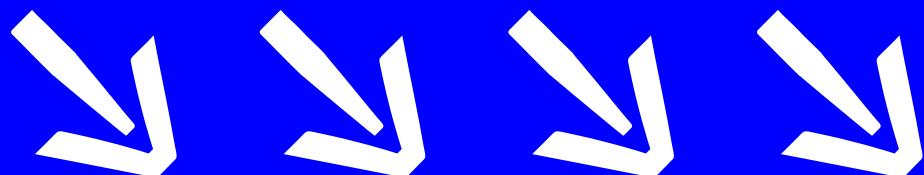
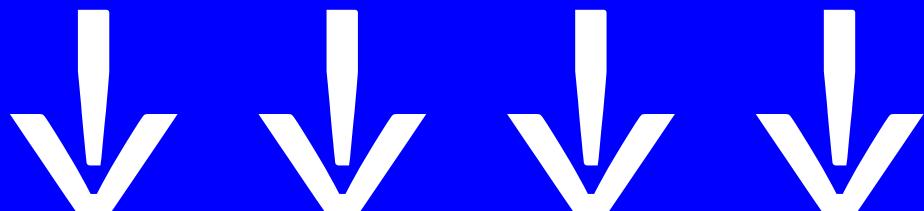
300.0

sInt

-15.0

CRSV

0.0



Introduction

13 Foreword

Noemi Stauffer

14 Process & Origins

Stephen Nixon

Variable axes

23 Monospace MONO

29 Casual CASL

37 Weight wght

43 Slant + Cursive slnt CRSV

Appendix

50 Static Fonts

58 Languages

60 Character Map

62 OpenType Features

64 Code Ligatures

68 Definitions

71 Afterword

Google Fonts Team

Sans

Sans Linear ExtraBlack

Linear

Sans Linear ExtraBlack Italic

Sans

Sans Linear Black

Linear

Sans Linear Black Italic

Sans

Sans Linear ExtraBold

Linear

Sans Linear ExtraBold Italic

Sans

Sans Linear Bold

Linear

Sans Linear Bold Italic

Sans

Sans Linear SemiBold

Linear

Sans Linear SemiBold Italic

Sans

Sans Linear Medium

Linear

Sans Linear Medium Italic

Sans

Sans Linear Regular

Linear

Sans Linear Italic

Sans

Sans Linear Light

Linear

Sans Linear Light Italic

Mono

Mono Linear ExtraBlack

Linear

Mono Linear ExtraBlack Italic

Mono

Mono Linear Black

Linear

Mono Linear Black Italic

Mono

Mono Linear ExtraBold

Linear

Mono Linear ExtraBold Italic

Mono

Mono Linear Bold

Linear

Mono Linear Bold Italic

Mono

Mono Linear SemiBold

Linear

Mono Linear SemiBold Italic

Mono

Mono Linear Medium

Linear

Mono Linear Medium Italic

Mono

Mono Linear Regular

Linear

Mono Linear Italic

Mono

Mono Linear Light

Linear

Mono Linear Light Italic

Sans

Casual

Sans Casual ExtraBlack

Sans

Casual

Sans Casual Black Italic

Sans

Casual

Sans Casual ExtraBold Italic

Sans

Casual

Sans Casual Bold Italic

Sans

Casual

Sans Casual SemiBold Italic

Sans

Casual

Sans Casual Medium Italic

Sans

Casual

Sans Casual Italic

Sans

Casual

Sans Casual Light Italic

Mono

Casual

Mono Casual ExtraBlack

Mono

Casual

Mono Casual Black Italic

Mono

Casual

Mono Casual ExtraBold Italic

Mono

Casual

Mono Casual Bold Italic

Mono

Casual

Mono Casual SemiBold Italic

Mono

Casual

Mono Casual Medium Italic

Mono

Casual

Mono Casual Regular

Mono

Casual

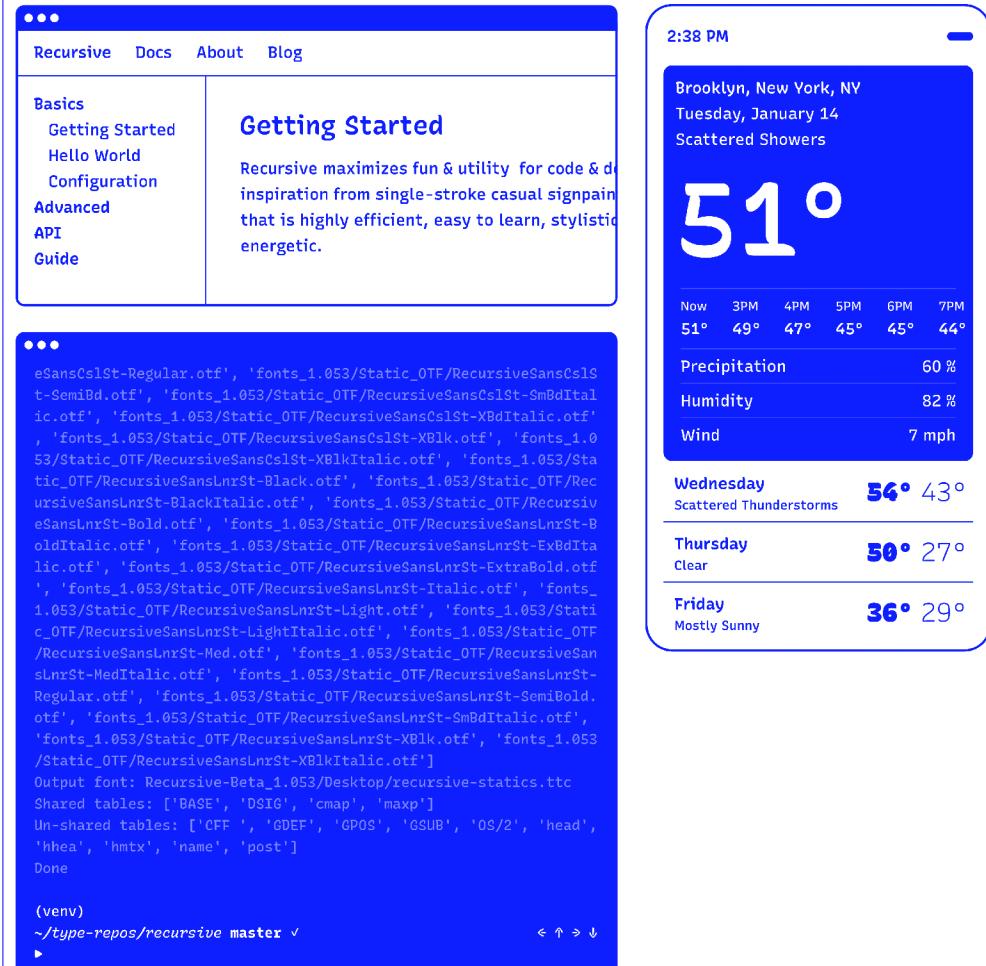
Mono Casual Light Italic

A new, highly- flexible, variable font.

Built to maximize versatility, control, and performance, Recursive is a five-axis variable font. This enables you to choose from a wide range of predefined styles or dial in exactly what you want for each of its axes: ***Monospace, Casual, Weight, Slant, and Cursive***. Taking full advantage of variable font technology, Recursive offers an unprecedented level of flexibility, all from a single font file.

Extraordinarily versatile.

Recursive draws inspiration from single-stroke casual signpainting, a style of brush writing that is stylistically flexible and warmly energetic. Adapting this aesthetic basis into a type system, Recursive is designed to excel in digital interactive environments. This makes the typeface ideal for a wide range of uses, including data-rich apps, technical documentation and code editors.



```

1
2
3
4      - - - C A S L - - -
5      T |           T |   M
6      H |           H |   O
7      G |           G |   N
8      W |           W |   O
9      - - - C A S L - - - N - - - C A S L - - -
10     |   |   |   T   |   T |   T |
11     |   |   |   H   |   H |   H |
12     S   |   S   |   G   |       G   S
13     L   |   L   |   W   |       W   L
14     N   T   N   T   - - - C A S L - - - N
15     T   H   T   H   |   |   |   T
16     |   G   |   G   |   |   |   |
17     |   W   |   W   S   |   S   |
18     - - - C A S L - - - L   - - - L   - - -
19     M   N   T   N   T
20     O   T   H   T   H
21     N   |   G   |   |   G
22     O   |   W   |   |   W
23     - - - C A S L - - -
24
25
26
27
28
29     | - - - - - |          FLATTENED
30     z5. top          NOORDZIJ CUBE
31     |             CONSTRUCTION
32     | X↑ Y↓
33     | Zmax
34     | - - - - - |
35     | - - - - - | - - - - - | - - - - - | - - - - - |
36     z4. front | | s2. right | | s3. back | | s4. left | |
37     z3
38     z2 | X↑ Z↑ | | Y↓ Z↑ | | X↓ Z↑ | | Y↑ Z↑ |
39     z1 | Ymin | | Xmax | | Ymax | | Xmin |
40     | - - - - - | - - - - - | - - - - - | - - - - - |
41     | - - - - - |
42     z0 | s0. bottom |          . - - - .
43     |             |          / 5  / | . - - - 2 .
44     | X↑ Y↑ |          | Zmin |          | z | 1 | / ↗
45     |             |          |             | . - - - y
46     | - - - - - |          |             | x →
47
48
49
50

```

Foreword

Recursive was born from a desire to explore how the emerging technology of variable fonts can maximize both utility and creativity on the web. From the start, every aspect of this project was motivated by a will to push boundaries – from drawing to proofing to mastering. Recursive – whose name encompasses ideas of recursion and cursive¹ writing – builds on its designer's range of interests, experiences, and needs as a user of type.

¹ A typographic glossary provides a definition for this term and others on p. 68

Taking the forms and structure of single-stroke casual lettering as a base, Stephen Nixon has done more than simply adapt these examples to the screen. Instead, he's delivered a typeface that offers unique possibilities for interactive design. Combining the spontaneity of a script typeface with the versatility of multiple classifications, weights, and cuts, Recursive takes the notion of what a 'variable font' is to a new level.

Brought to you by Arrow Type and commissioned by Google Fonts, Recursive is now free to use in any personal or commercial project.

Noemi Stauffer,
Editor

Designing a five-axis variable font

In 2018, I graduated with a Master's Degree from TypeMedia, a 10-month immersion in type design at the Royal Academy of Art (KABK) in The Hague, Netherlands. For my thesis project, I created **Recursive**, a typeface exploring the intersection of sign painting, monospaced type, and the emerging technology of variable fonts. In 2019, Google Fonts commissioned me to complete the project for open-source distribution. Recursive is now the first release of my new type foundry, Arrow Type.

An origin in painted letters

One of my favorite styles of signwriting is a genre broadly referred to as **casual**. It includes **casual script**, in which letters are handwritten with a brush and connected together. It also includes **single-stroke casual**, which is typically painted in uppercase, forward-slanted letters.

Example of casual lettering in NYC
(Painter unknown;
Brooklyn, NY, near
Myrtle & Wyckoff Aves;
2019)



Verdana used at inappropriately large scale in street signage. This inspired the flattened terminals of Recursive.



A foundational style for sign painters, single-stroke casual is comprised of just a few basic strokes which are assembled together to create letters. As a result, this style is highly practical, easy to learn, and (relatively) easy to paint. Its features are deliberately informal and “unbalanced”: letters are somewhat condensed and have curved stems, stroke connections are left visibly imperfect, and midpoints are low-slung. These qualities allow a sign painter to avoid some of the aspects of letter shaping that can be most challenging: perfectly straight lines, rigid symmetry, and optically-centered elements like the spine of the letter ‘S’.

Living in New York City, I come across striking works of casual lettering almost daily in the vernacular signage of diners, laundromats, and bodegas. Recursive is shaped by my love of these signs, as well as by lessons learned from classmates & teachers at TypeMedia, educators like John Downer, personal sketching, and technical experimentation.

Crafting a casual for code

Early on in the design process, one thing stood out about single-stroke casuals: not only could they be squeezed into condensed proportions or stretched into extremely wide styles, but they were particularly eye-catching when worked into the confines of monospaced letters. This realization left me wanting to explore what a **single-stroke casual monospace** font could be. With my background in web design and



development, creating a font for use in programming that would be both highly readable and aesthetically pleasing was an exciting challenge. Moreover, it was a perfect opportunity for me to design a font that would meet my requirements both as a designer and as a developer.

Most typefaces with both sans-serif and monospace variants are first designed to be proportional and later adapted into a fixed-width alternative. With Recursive, I decided to design going in the other direction. Starting with a monospace and adapting it into a sans-serif was an unconventional approach, but because I was most curious to see how casual letters might look in code, it felt natural to follow this direction.

In this project, I have given particular attention to adapting the key visual aspects of single-stroke casuals into simplified monospaced and semi-proportional letterforms. I preserved their low center points and chiseled stroke endings but set their forward slant to a variable axis. Keeping some casual gestures helps to maintain the warmth and

energy of the style, while simplification makes it practical for everyday typographic use.

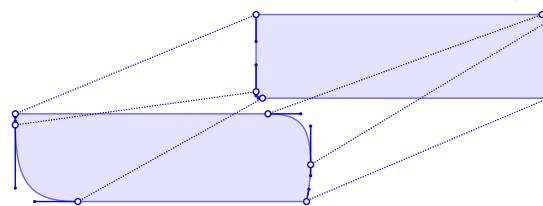
The need for multiple voices

Because the typeface was started from painted letters, it possessed a lot of personality from the start. For it to work well in code and on screen, however, I knew I needed to temper its personality and improve its readability. Finding the exact right voice was challenging. From my own experience, I knew that developers have different typographic needs based on the task at hand. Some monospace fonts are drawn with attention-grabbing details, which lends them a strong visual impact for display settings like posters and signage, but can make them distracting to code with. Other monospace fonts are designed with only traditional legibility in mind, but this can give them a cold and monotonous tone when used in settings such as docs and blogs.

This was a contradiction that I began to realize from my earliest sketches. Still, I wanted to make Recursive a monospace font that could be ideal in different contexts, from serious to casual. I soon had the idea to solve this issue with the addition of a variable axis, but knew that it would have clear design constraints. I needed to find two ends of a continuous spectrum that could be different enough to be striking counterparts while also both functioning well at the small sizes used for code. Finding the ends of this spectrum was a process of many rounds of trial and failure, but things gradually improved, helped greatly by a steady stream of critique from professors and visiting designers at TypeMedia.

Adding a variable axis for stylistic range was not just a design challenge, but also a demanding technical endeavor. When adding the Casual axis, I had to draw for interpolation compatibility, making every glyph

The hyphen in Casual versus Linear drawings, showing how the point structure of each must closely match its counterpart to allow interpolation.



twice – first with brushy contours for the Casual end of the axis, then again with rectangular shaping for the Linear end of the axis. What made this difficult is that both versions of each glyph had to match in the number and order of contours, nodes, anchors, and more. This was to ensure that they would be compatible for the interpolation required to morph along this stylistic range.

Despite the challenge of production, a spectrum of expression within a single typeface has clear advantages for users. Some of the most compelling typographic layouts use multiple, complementary fonts. However, because different fonts usually have different metrics, such layouts can easily become a headache for visual design and technical implementation. Recursive, by contrast, provides multiple voices from a single, easy-to-use font file.

Expanding the possibilities to interactive design
No matter how versatile, a monospace font is only truly suited for a particular set of tasks¹. Because I intended Recursive for a wide range of use, I knew from early on that it needed a proportional counterpart. However, starting with a monospace design helped me find opportunities to make a Sans that could offer unique possibilities for interactive design.

Just as a monospace font maintains the same widths for letters between all weights and styles, I realized that my sans-serif could do the same: while all characters would have natural widths, each could keep its same width across all stylistic variations. I therefore built Recursive Sans as a **superplexed** family – all of its 32 instances have shared glyph width, kerning, and overall letterforms for every character. Of course, this also applies to in-between variations. The fact that characters within Recursive Sans have shared metrics ensures that line length is not affected when changing between its different font styles. This also allows smooth, animated transitions between any of the subfamily's **Weight, Slant, & Casual** axes.

¹ These include tasks which benefit from characters that are tabular and non-ambiguous, such as code, financial data, passwords, receipts, license numbers, serial numbers, captions, and more.

² Microinteractions are subtle animations in digital user interfaces that respond to user interaction through visual changes such as color, size, and position. This and other definitions can be found in a typographic glossary on page 68.

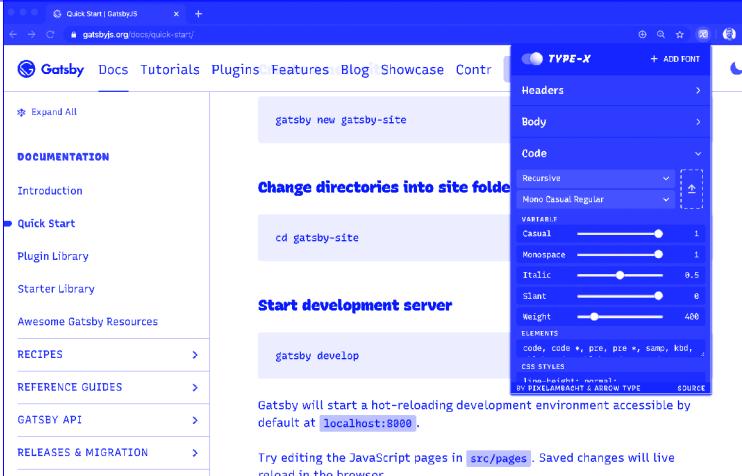
These design decisions were based on my understanding of user needs: before moving into type design, I worked as a visual designer for websites and software. In these past roles, my responsibility was to enhance the user experience of digital products, with a particular focus on typographic clarity and *microinteractions*². Through this, I found that most typefaces are terrible when used in animated transitions due to their shifting metrics and non-variable styles. With Recursive, I realized that I could find a way around these limitations.

Bringing a five-axis variable font to life sometimes seemed like a never-ending process. In a variable font, the number of drawings required for each glyph tends to go up with the addition of every axis. Between Casual, Monospace, Weight, Slant, & Cursive axes, Recursive required 24 total source font files. Each character required **at least** 12 compatible drawings to cover Casual, Weight, and Slant variations – and characters with Monospace and Cursive variations required even more drawings. All told, Recursive has 1,248 glyphs per source – and between its 24 masters, Recursive includes 29,952 total glyphs. Of course, a significant portion of these are composed from other glyphs (as is the case for most accented characters) or copied between masters (“normal-width” glyphs were copied from Mono to Sans masters), but there are still 6,804 hand-edited glyphs. This staggering amount of complexity made it critical to embrace scripting to automate parts of the process. This is also why it was so incredibly helpful to have the contribution of type designers Lisa Huang and Katja Schimmel and good software from many type-tool engineers.

A new design proofing tool

While designing Recursive, it was important to search for typographic problems across all styles in order to fix them and improve the font. This process, called proofing, typically means physically printing pages to show a broad character set in all styles across a range of point sizes. Proofs are often typeset in realistic graphic layouts, then carefully examined and marked for revision. A concise proof may easily have two to four pages per style – so repeatedly proofing all 64 instances of Recursive wasn't a good option.

Overriding fonts on GatsbyJS.org to test Recursive, with Type-X



It wouldn't have just been a waste of resources, but also a disregard of time constraints: most collaborative proofing sessions only have time to analyze a few pages in detail.

From the start, then, I sought ways of testing Recursive that would require printing fewer pages, instead relying on better screen-based tests. I started making JavaScript-generated layouts and other digital examples as the basis of my critique. However, I soon felt limited by trying to create realistic web typography examples without actually using real web content and styling. Eventually, it became clear that the best way to assess Recursive's design on screen was to see how it performed on real, live websites. I began to browse different websites, adding custom CSS rules to override their fonts with Recursive. In this way, I could preview and test the different styles of Recursive at the same time in a more-realistic context.

This process was useful, but it was also repetitive and time consuming. Worse, it didn't allow me to sit back and actually **experience** the fonts. I eventually realized that I could automate this workflow. To do so, I created a simple Google Chrome extension that could override fonts on any webpage at the touch of a button, imposing my own font files onto those pages. It made proofing Recursive online much more efficient, and it helped me find many opportunities for refinements that I may have otherwise missed.

In the past year, I worked with developer Roel Neiskens to redesign and rebuild this Chrome extension, in what we now call **Type-X**. Whereas my earlier tool was limited to a single, hard-coded font family, Type-X allows users to easily override websites with any font on their system. Font files can also be drag-and-dropped into the extension to activate advanced features like variable axis control. These new capabilities make Type-X far more powerful than a simple type-proofing tool: it is now useful to anyone wanting to better understand how different fonts look and feel in the context of the websites they use. Thanks to the sponsorship of Google Fonts, Type-X is now open source and available to anyone for free on the Chrome Web Store.

Designing Recursive has been a long but enriching (and fun!) process. It would not have been possible without the help of the many people who contributed their knowledge, critique, and time. I am especially grateful to Google Fonts for commissioning me to complete my thesis, to E Roon Kang, Bon Hae Koo, Minkyung Kim, Talia Cotton, Irin Kim, and Noemi Stauffer for helping make the project look and sound so good, to Lisa Huang and Katja Schimmel for helping me to complete the drawing of it, to Ben Kiel for helping to master the final fonts, to Rafał Buchner, Erik van Blokland, Tal Lemming, and Frederik Berlaen (and many other type tool developers) for making tools that made this design possible, to Gen Ramírez, Seán Donohoe, and John Downer for inspiration and lessons in sign painting, and to the faculty and alumni of TypeMedia for their mentorship and guidance.

Stephen Nixon
Designer, Recursive

Get Recursive and learn more about it at <https://recursive.design>

Sans & Mono. In one file.

```

recursive/src/masters/
  mono/
    Recursive Mono-Casual A Slanted.ufo
    Recursive Mono-Casual A.ufo
    Recursive Mono-Casual B Slanted.ufo
    Recursive Mono-Casual B.ufo
    Recursive Mono-Casual C Slanted.ufo
    Recursive Mono-Casual C.ufo
    Recursive Mono-Linear A Slanted.ufo
    Recursive Mono-Linear A.ufo
    Recursive Mono-Linear B Slanted.ufo
    Recursive Mono-Linear B.ufo
    Recursive Mono-Linear C Slanted.ufo
    Recursive Mono-Linear C.ufo
  sans/
    Recursive Sans-Casual A Slanted.ufo
    Recursive Sans-Casual A.ufo
    Recursive Sans-Casual B Slanted.ufo
    Recursive Sans-Casual B.ufo
    Recursive Sans-Casual C Slanted.ufo
    Recursive Sans-Casual C.ufo
    Recursive Sans-Linear A Slanted.ufo
    Recursive Sans-Linear A.ufo
    Recursive Sans-Linear B Slanted.ufo
    Recursive Sans-Linear B.ufo
    Recursive Sans-Linear C Slanted.ufo
    Recursive Sans-Linear C.ufo
  recursive-MONO_CASL_wght_slnt_ital.designspace

```

The typeface comes in two practical and highly readable subfamilies, Sans & Mono. Thanks to its **Monospace** axis ('MONO'), both of these subfamilies can be used in a single font file. You can even select custom instances that are semi-proportional or semi-monospaced.

Of course, vertical metrics such as line height, cap height, and x-height are shared across the entire **Monospace** axis. This allows harmonious and performant layouts, even where different proportions are mixed, such as in data-rich applications and technical documentation.

Recursive Sans ('MONO 0') is made for text & user interface design. While its proportional characters deliver comfortable reading at text sizes, its heaviest weights are perfect to create punchy, tightly-spaced headlines.

Recursive Mono ('MONO 1') is made for code. Its characters share the same width for clear legibility and perfect alignment. This is particularly helpful for use in programming and data-heavy design tasks, but also allows for creative possibilities in display typography.

A Sans for more robust layouts.

The characters within both subfamilies, Sans & Mono, maintain the exact same width across all font styles, independently of the values set on the **Weight, Casual, Slant, & Cursive** axes. You can therefore use Recursive to create animated font transitions without breaking the layout of UI elements like menus and buttons.

Sans Casual

same
same
same
same
same
same
same
same
same
same

Sans Linear

widths
widths
widths
widths
widths
widths
widths
widths
widths
widths

600

W7i

600

600

i#fs

600

#fs

600

fS

900

W7i

600

350

i#fs

600

#fs

400

550

-20

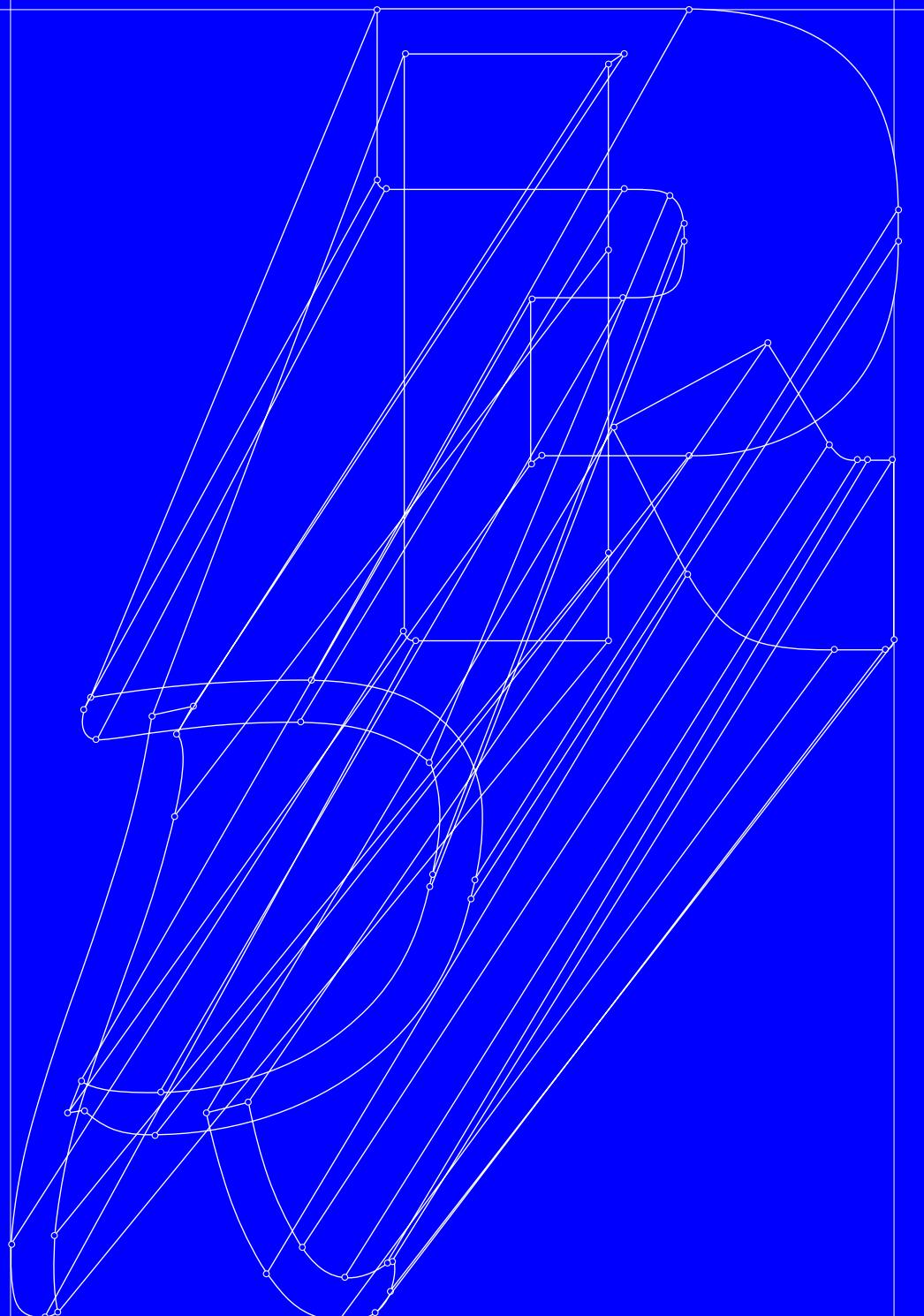
550

Ready for work. Ready for play.

Recursive uses its ***Casual*** axis ('CASL') to offer a range of personality, allowing you to adjust tone for different contexts. Along this axis, letterforms adjust in stroke curvature, contrast, and terminals to go from a sturdy, rational ***Linear*** to a friendly, energetic ***Casual***. All the styles along this axis are designed to maintain high legibility at medium and text sizes. At display sizes, it is most effective to use either end of the ***Casual*** axis.

Linear ('CASL 0') styles have subtly-flattened edges and simple, open forms. This optimizes readability and enables enhanced focus in dense information, such as long-form text and complex code.

Casual ('CASL 1') echoes the soft & curvy brush strokes of casual signpainting, but simplifies these forms for a striking and inviting tone. This makes it ideal for web headlines, code snippets, and command line interfaces.

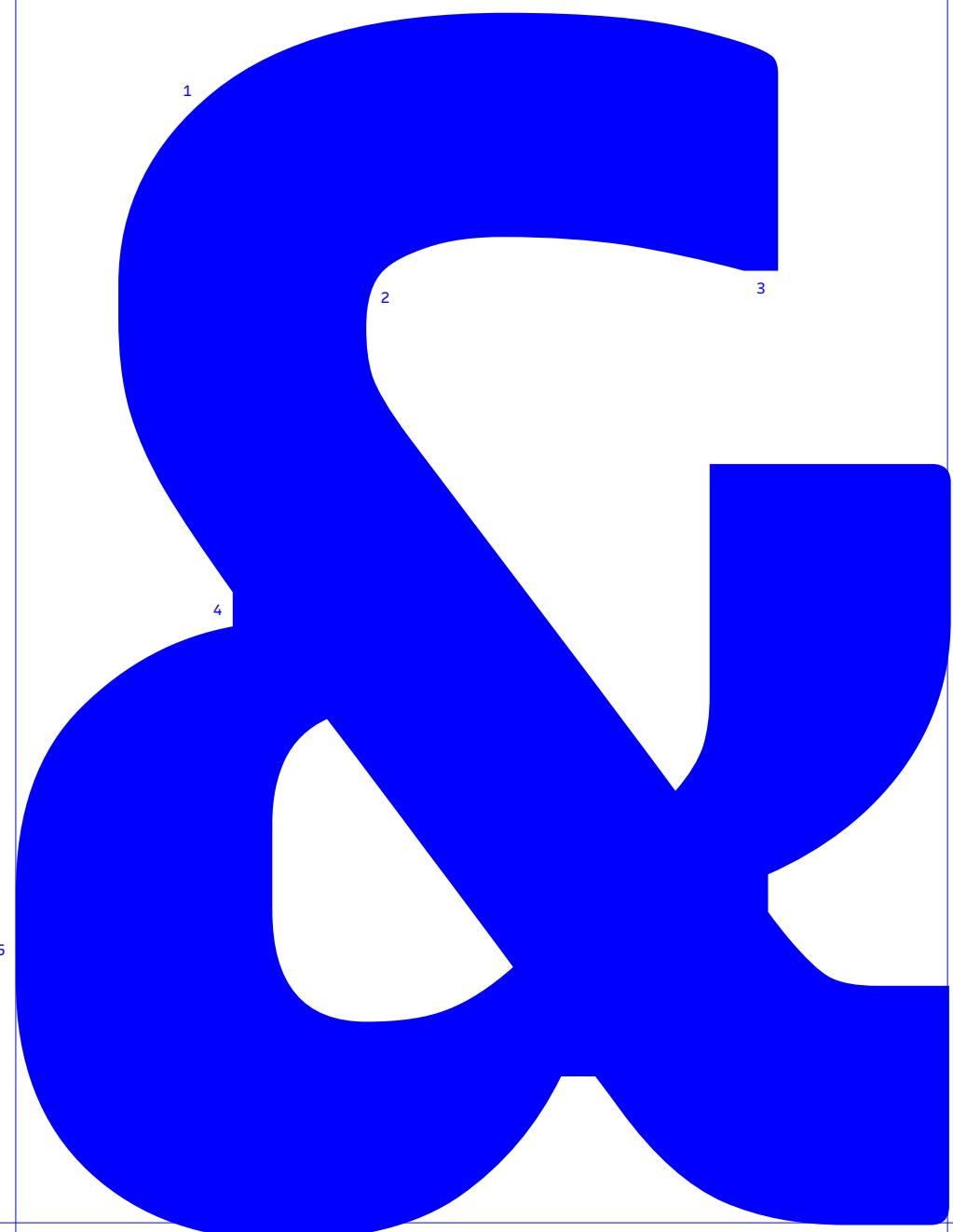


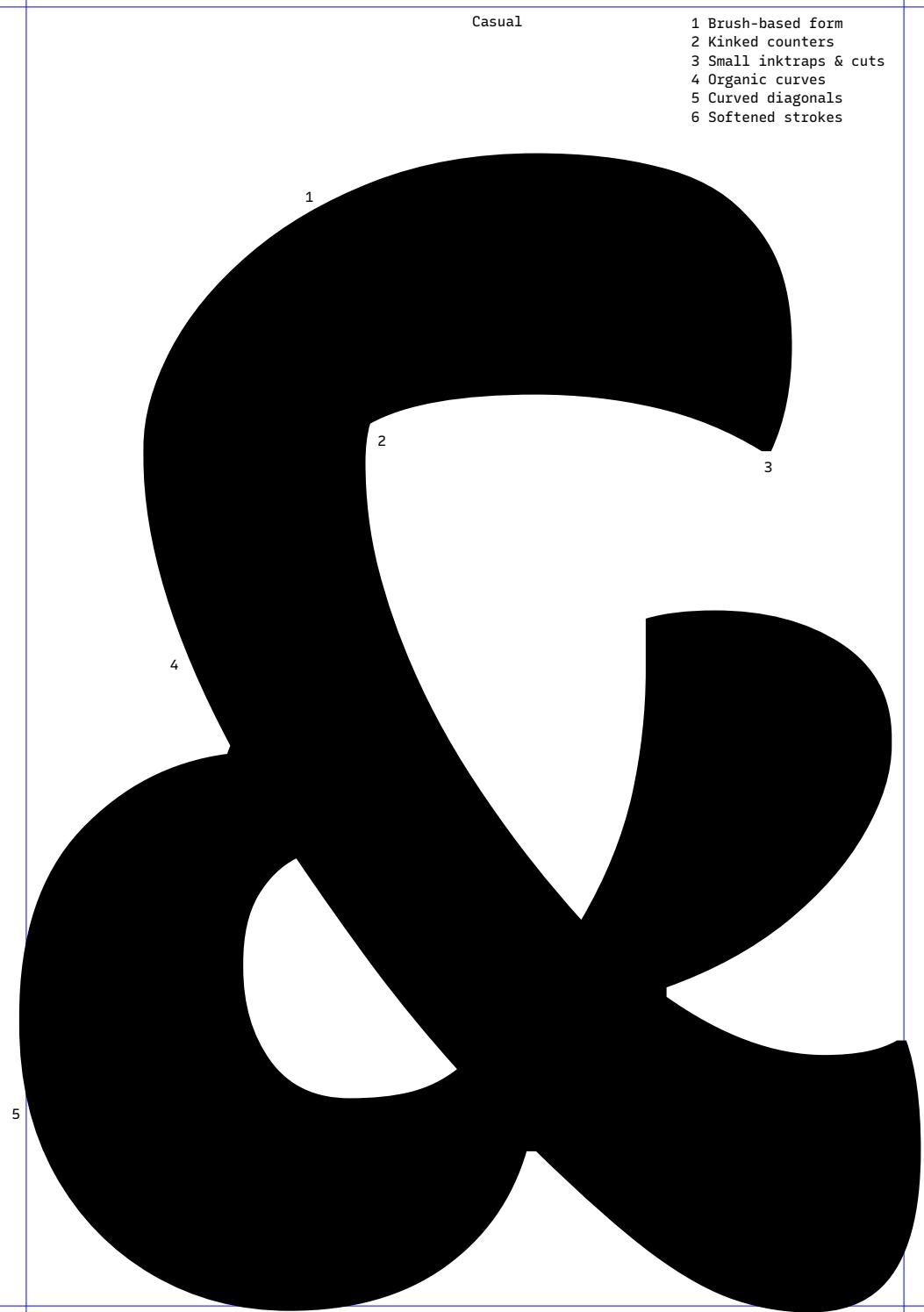
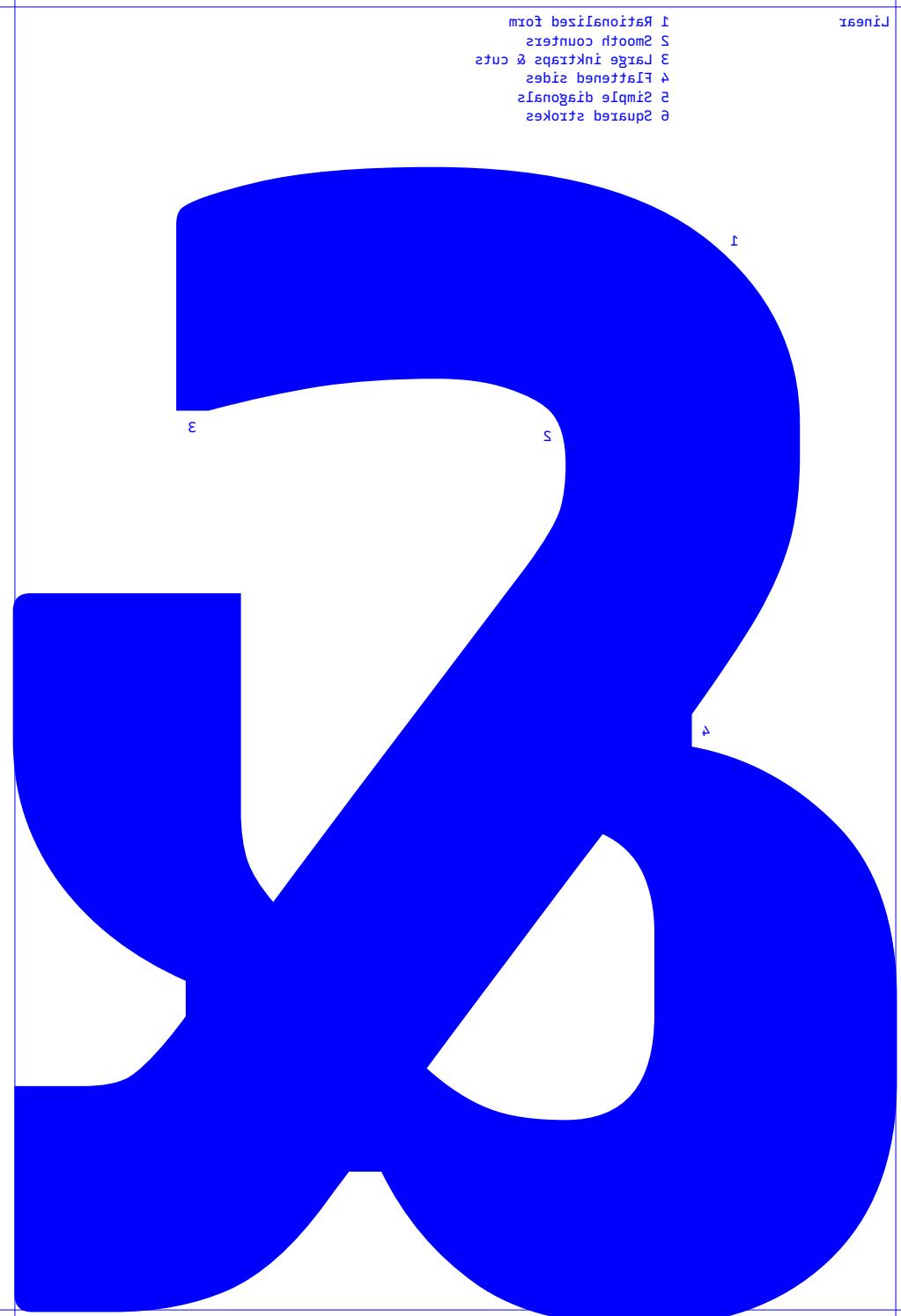
W O R K

W Sans Linear ExtraBold
O Sans Linear Light Italic
R Mono Casual Light Italic
K Sans Casual ExtraBlack Italic

Linear

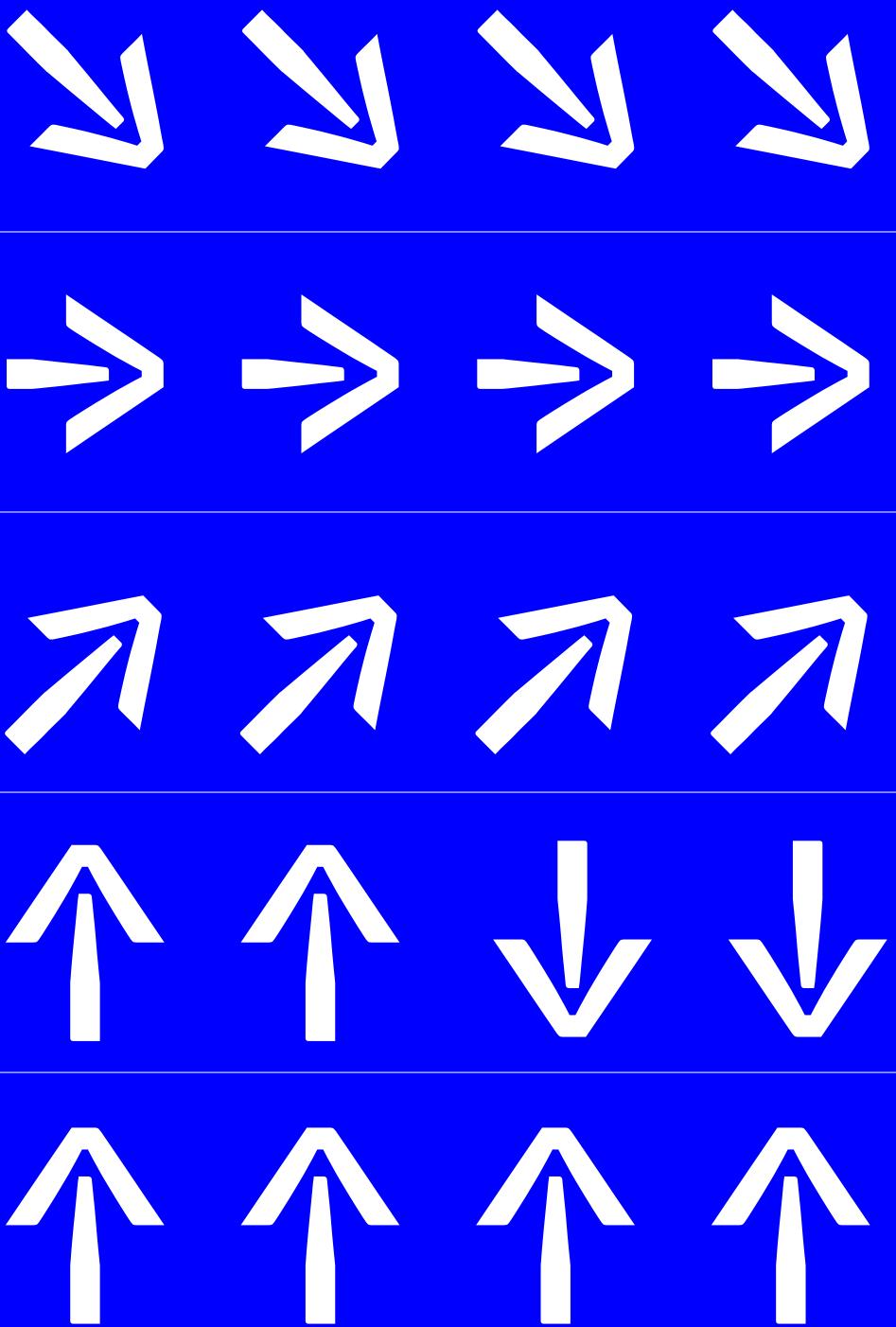
- 1 Rationalized form
- 2 Smooth counters
- 3 Large inktraps & cuts
- 4 Flattened sides
- 5 Simple diagonals
- 6 Squared strokes





P L
A Y

P Sans Casual Black
L Mono Linear ExtraBlack
A Mono Linear Bold
Y Sans Casual ExtraBold Italic

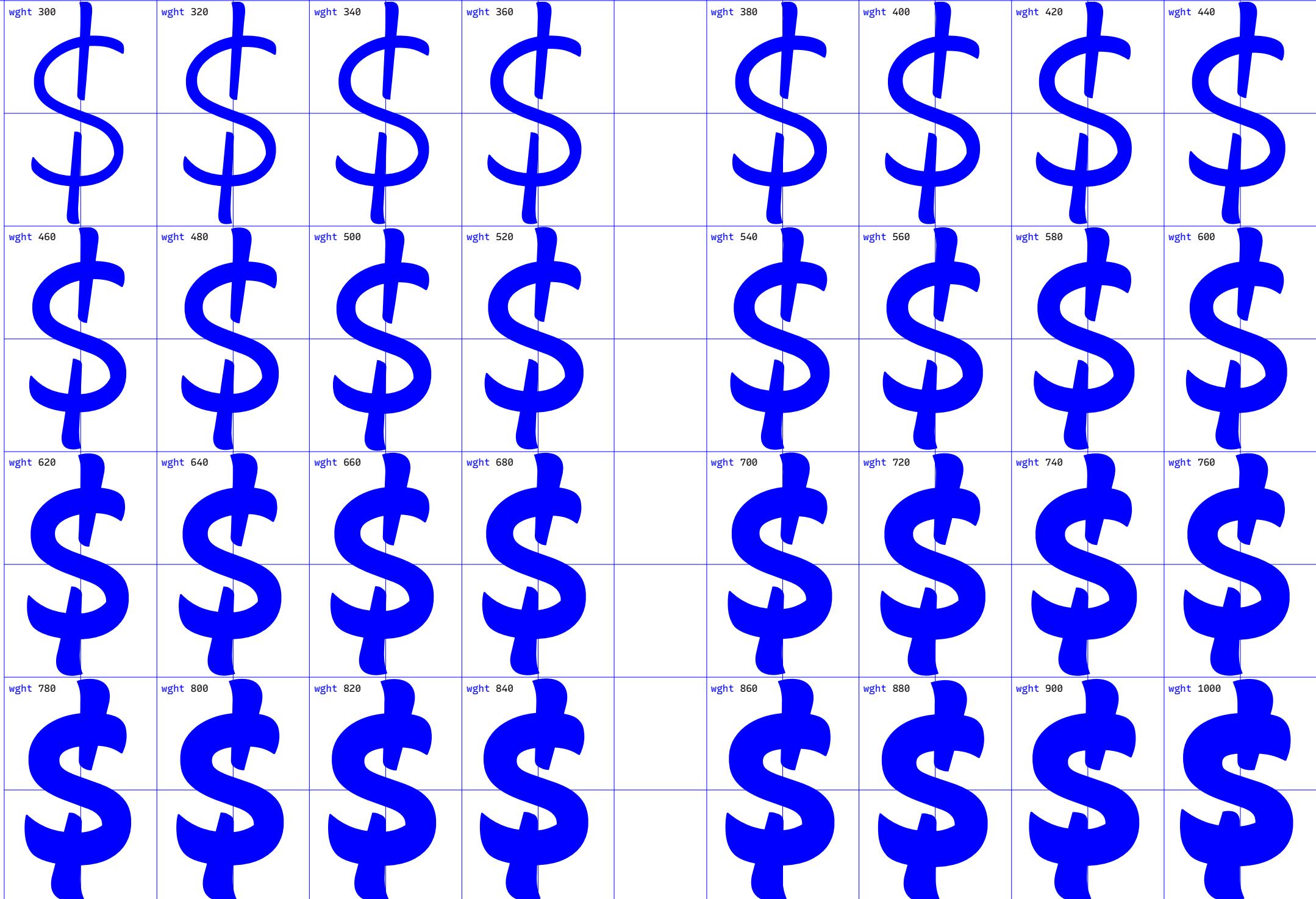


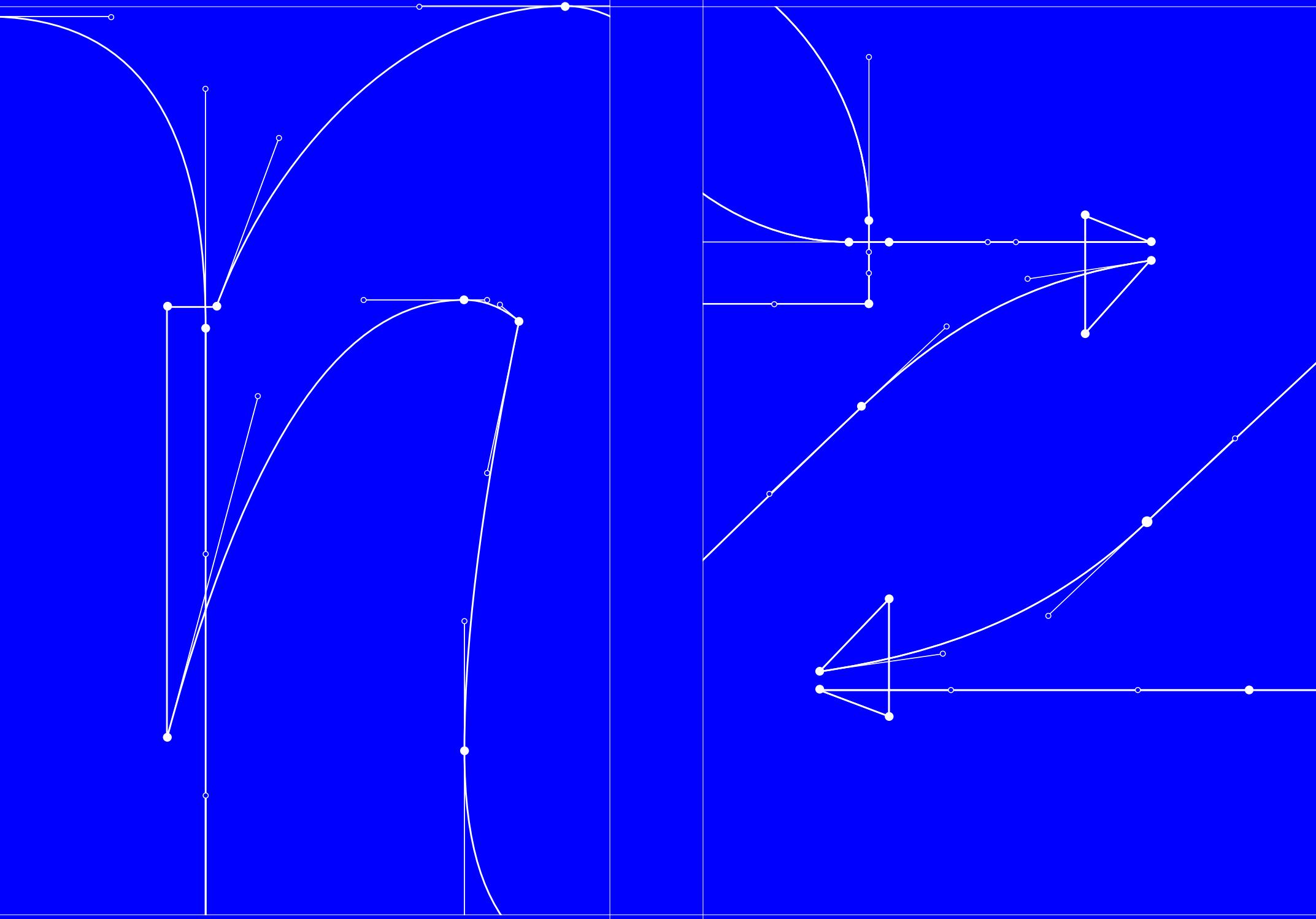
Light
Regular
Medium
SemiBold
Bold
ExtraBold
Black
ExtraBlack

**Eight
weights.
But also
way more.**

Recursive comes in a wide range of weights, from Light ('300') to a super-heavy ExtraBlack ('1000'). You can choose from its seven predefined weights, or set a custom value through its ***Weight*** ('wght') axis.

And because Recursive maintains consistent letterforms along the entire ***Weight*** axis, it allows for smooth animations between any of its different weights.





slnt **CRSV**
slnt **CRSV**

**Slanted
or Cursive.
Or both.**

In Recursive, slant and cursive styles can be controlled separately. The ***Slant*** axis ('`slnt`') defines the angle of the letters, while the ***Cursive*** axis ('`CRSV`') lets you toggle cursive letterforms. This makes it possible to use sloped romans (`'slnt -15, CRSV 0'`), upright italics (`'slnt 0, CRSV 1'`), or set custom values on both axes for more options to play with.

Recursive's cursive letterforms ('`CRSV 1`') replace familiar "roman" glyphs with more-handwritten alternates such as the single-story "a" and "g". By default, Recursive will automatically apply these cursive alternates when setting the ***Slant*** axis ('`slnt`') beyond -14. This allows smooth, animated transitions from normal to oblique type up to 13.99° of slope, but also a "true italic" style with cursive letterforms at 14°.

CRSV: 0 (off)

slnt: 0
agkrX

slnt: -5
agkrX

slnt: -10
agkrX

slnt: -15
agkrX

CRSV: 0.5 (auto)

slnt: 0
agkrX

slnt: -5
agkrX

slnt: -10
agkrX

slnt: -15
agkrX

x kgk rx

agkrx

x kgk rx

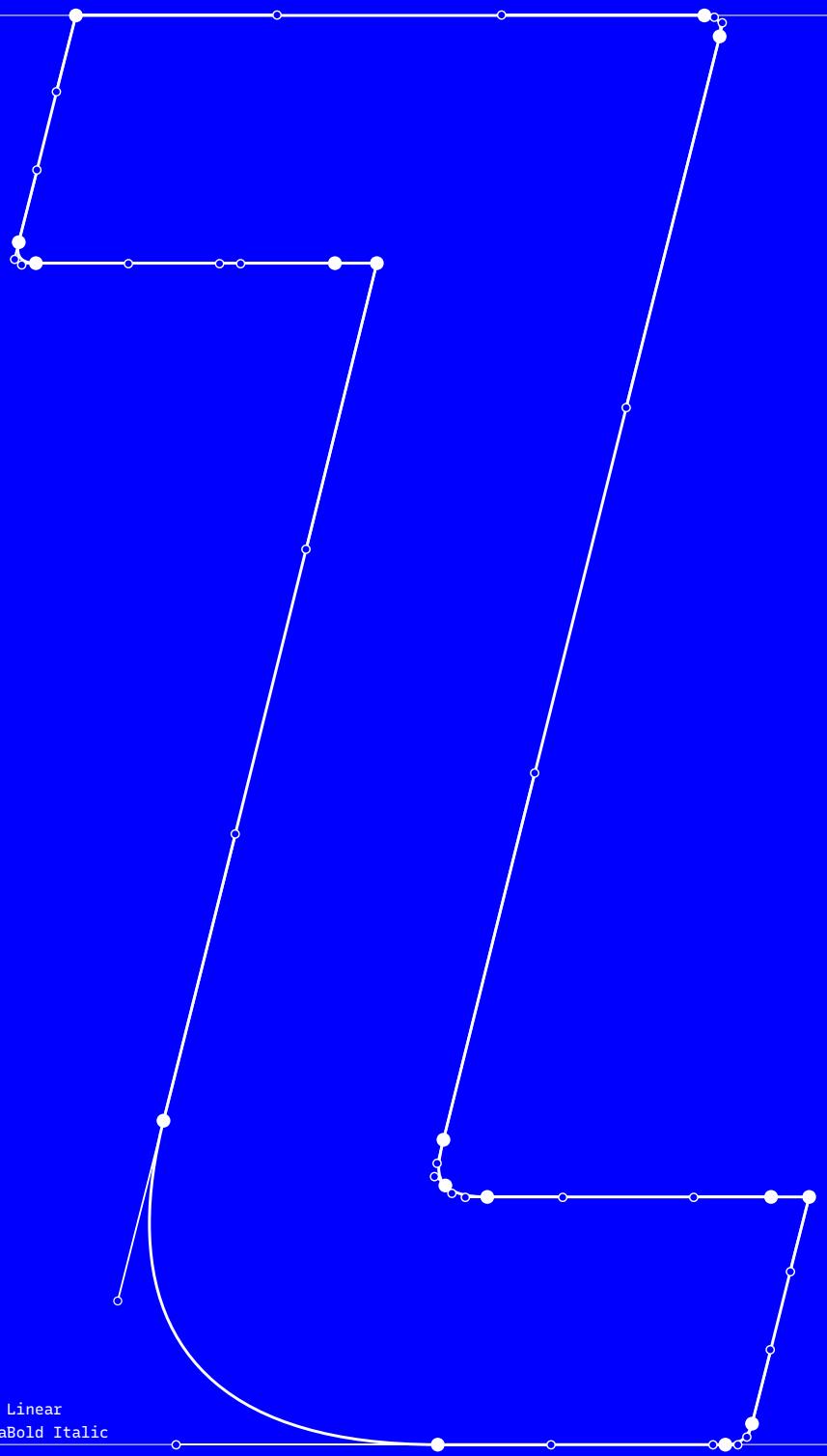
agkrx

x kgk rx

agkrx

x agkrx

agkrx



Powerful, yet simple to use.

Mono

Casual

Normal

Light

Regular

Medium

SemiBold

Bold

ExtraBold

Black

ExtraBlack

Linear

Normal

Italic

Sans

Casual

Normal

Italic

Linear

Normal

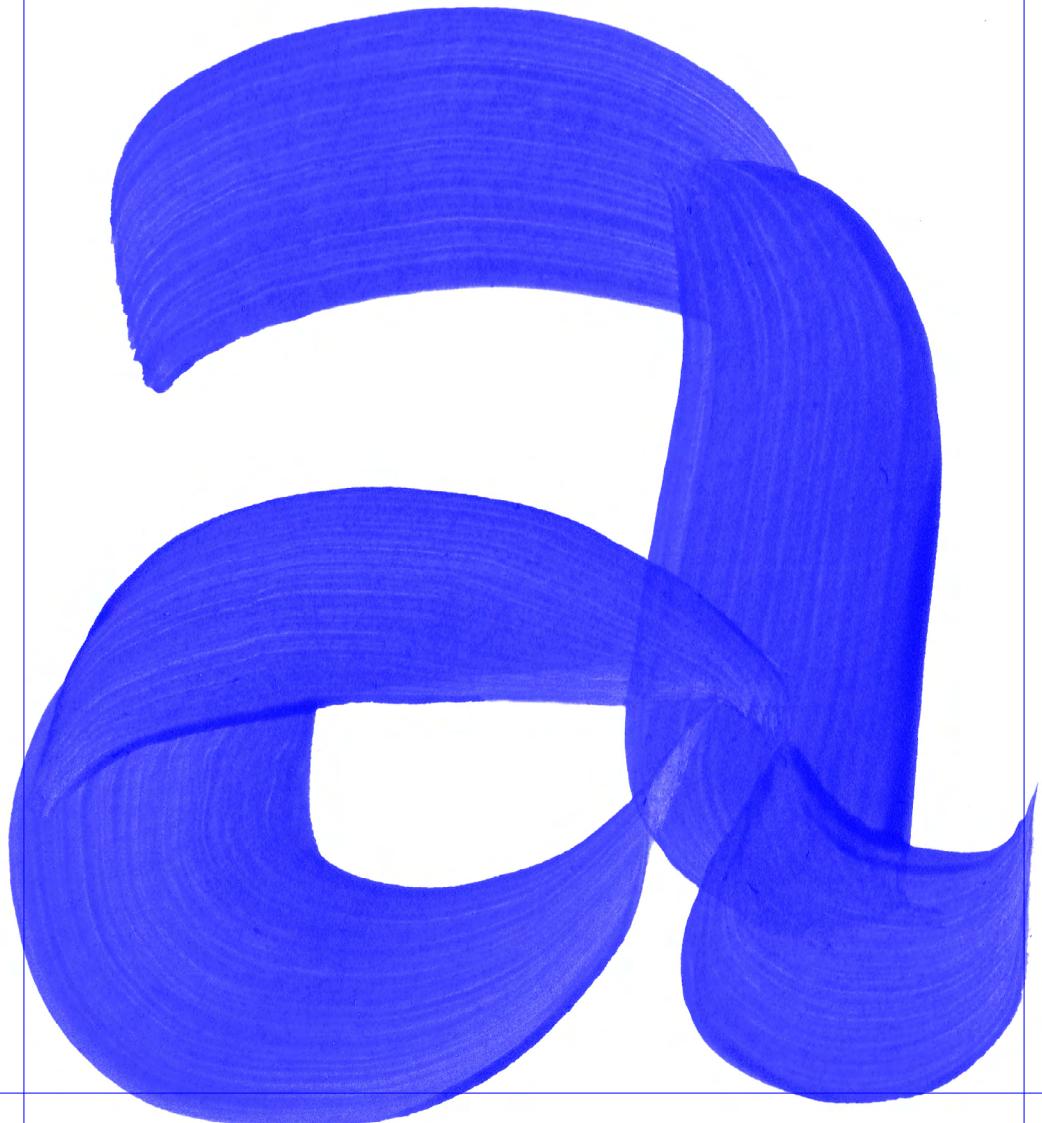
Italic



As a variable font, Recursive gives you fine-grained control over each one of its styles. However, it also comes with 64 predefined styles that are easy to access through your font menu. Called **named instances**¹, these work just like regular static fonts do.

¹ A typographic glossary provides a definition for these terms and others on p. 68

Stroke Logic The shaping of Recursive is derived from the shapes of single-stroke casual signwriting, adapted to a monospace construction.



999.99	998.99	997.99	996.99	995.98	994.98	993.98	992.98	991.98	990.98		649.49	648.49	647.49	646.49	645.49	644.49	643.49	642.48	641.48	640.48
989.98	988.97	987.97	986.97	985.97	984.97	983.97	982.97	981.96	980.96		639.48	638.48	637.48	636.48	635.47	634.47	633.47	632.47	631.47	630.47
979.96	978.96	977.96	976.96	975.96	974.95	973.95	972.95	971.95	970.95		629.47	628.46	627.46	626.46	625.46	624.46	623.46	622.46	621.45	620.45
969.95	968.95	967.94	966.94	965.94	964.94	963.94	962.94	961.94	960.93		619.45	618.45	617.45	616.45	615.45	614.44	613.44	612.44	611.44	610.44
959.93	958.93	957.93	956.93	955.93	954.93	953.92	952.92	951.92	950.92		609.44	608.44	607.43	606.43	605.43	604.43	603.43	602.43	601.43	600.42
949.92	948.92	947.92	946.91	945.91	944.91	943.91	942.91	941.91	940.91		599.42	598.42	597.42	596.42	595.42	594.42	593.41	592.41	591.41	590.41
939.91	938.90	937.90	936.90	935.90	934.90	933.90	932.90	931.89	930.89		589.41	588.41	587.41	586.41	585.40	584.40	583.40	582.40	581.40	580.40
929.89	928.89	927.89	926.89	925.89	924.88	923.88	922.88	921.88	920.88		579.40	578.39	577.39	576.39	575.39	574.39	573.39	572.39	571.38	570.38
919.88	918.88	917.87	916.87	915.87	914.87	913.87	912.87	911.87	910.86		569.38	568.38	567.38	566.38	565.38	564.37	563.37	562.37	561.37	560.37
909.86	908.86	907.86	906.86	905.86	904.86	903.85	902.85	901.85	900.85		559.37	558.37	557.36	556.36	555.36	554.36	553.36	552.36	551.36	550.35
899.85	898.85	897.85	896.84	895.84	894.84	893.84	892.84	891.84	890.84		549.35	548.35	547.35	546.35	545.35	544.35	543.34	542.34	541.34	540.34
889.83	888.83	887.83	886.83	885.83	884.83	883.83	882.82	881.82	880.82		539.34	538.34	537.34	536.33	535.33	534.33	533.33	532.33	531.33	530.33
879.82	878.82	877.82	876.82	875.81	874.81	873.81	872.81	871.81	870.81		529.32	528.32	527.32	526.32	525.32	524.32	523.32	522.31	521.31	520.31
869.81	868.80	867.80	866.80	865.80	864.80	863.80	862.80	861.79	860.79		519.31	518.31	517.31	516.31	515.30	514.30	513.30	512.30	511.30	510.30
859.79	858.79	857.79	856.79	855.79	854.78	853.78	852.78	851.78	850.78		509.30	508.29	507.29	506.29	505.29	504.29	503.29	502.29	501.28	500.28
849.78	848.78	847.77	846.77	845.77	844.77	843.77	842.77	841.77	840.76		499.28	498.28	497.28	496.28	495.28	494.27	493.27	492.27	491.27	490.27
839.76	838.76	837.76	836.76	835.76	834.76	833.75	832.75	831.75	830.75		489.27	488.27	487.26	486.26	485.26	484.26	483.26	482.26	481.26	480.25
829.75	828.75	827.75	826.74	825.74	824.74	823.74	822.74	821.74	820.74		479.25	478.25	477.25	476.25	475.25	474.25	473.25	472.24	471.24	470.24
819.74	818.73	817.73	816.73	815.73	814.73	813.73	812.73	811.72	810.72		469.24	468.24	467.24	466.24	465.23	464.23	463.23	462.23	461.23	460.23
809.72	808.72	807.72	806.72	805.72	804.71	803.71	802.71	801.71	800.71		459.23	458.22	457.22	456.22	455.22	454.22	453.22	452.22	451.21	450.21
799.71	798.71	797.70	796.70	795.70	794.70	793.70	792.70	791.70	790.69		449.21	448.21	447.21	446.21	445.21	444.20	443.20	442.20	441.20	440.20
789.69	788.69	787.69	786.69	785.69	784.69	783.68	782.68	781.68	780.68		439.20	438.20	437.19	436.19	435.19	434.19	433.19	432.19	431.19	430.18
779.68	778.68	777.68	776.67	775.67	774.67	773.67	772.67	771.67	770.67		429.18	428.18	427.18	426.18	425.18	424.18	423.17	422.17	421.17	420.17
769.66	768.66	767.66	766.66	765.66	764.66	763.66	762.65	761.65	760.65		419.17	418.17	417.17	416.16	415.16	414.16	413.16	412.16	411.16	410.16
759.65	758.65	757.65	756.65	755.64	754.64	753.64	752.64	751.64	750.64		409.15	408.15	407.15	406.15	405.15	404.15	403.15	402.14	401.14	400.14
749.64	748.63	747.63	746.63	745.63	744.63	743.63	742.63	741.62	740.62		399.14	398.14	397.14	396.14	395.13	394.13	393.13	392.13	391.13	390.13
739.62	738.62	737.62	736.62	735.62	734.61	733.61	732.61	731.61	730.61		389.13	388.12	387.12	386.12	385.12	384.12	383.12	382.12	381.11	380.11
729.61	728.61	727.60	726.60	725.60	724.60	723.60	722.60	721.60	720.59		379.11	378.11	377.11	376.11	375.11	374.10	373.10	372.10	371.10	370.10
719.59	718.59	717.59	716.59	715.59	714.59	713.58	712.58	711.58	710.58		369.10	368.10	367.09	366.09	365.09	364.09	363.09	362.09	361.09	360.08
709.58	708.58	707.58	706.58	705.57	704.57	703.57	702.57	701.57	700.57		359.08	358.08	357.08	356.08	355.08	354.08	353.08	352.07	351.07	350.07
699.57	698.56	697.56	696.56	695.56	694.56	693.56	692.56	691.55	690.55		349.07	348.07	347.07	346.07	345.06	344.06	343.06	342.06	341.06	340.06
689.55	688.55	687.55	686.55	685.55	684.54	683.54	682.54	681.54	680.54		339.06	338.05	337.05	336.05	335.05	334.05	333.05	332.05	331.04	330.04
679.54	678.54	677.53	676.53	675.53	674.53	673.53	672.53	671.53	670.52		329.04	328.04	327.04	326.04	325.04	324.03	323.03	322.03	321.03	320.03
669.52	668.52	667.52	666.52	665.52	664.52	663.51	662.51	661.51	660.51		319.03	318.03	317.02	316.02	315.02	314.02	313.02	312.02	311.02	310.01
659.51	658.51	657.51	656.50	655.50	654.50	653.50	652.50	651.50	650.50		309.01	308.01	307.01	306.01	305.01	304.01	303.00	302.00	301.00	300.00

é N g g ß
ë ñ g g k
ø ü ÿ ß

é Mono Linear ExtraBlack ñ Mono Casual Light
ë Mono Linear Light ü Sans Casual SemiBold Italic

æ Mono Casual Light ß Sans Casual ExtraBold
øg Mono Linear ExtraBold k Sans Casual Regular

Supports over 200 Languages.

To meet the needs of global communication, Recursive supports a wide range of Latin-based languages, including Vietnamese. It also comes with an extended set of currencies, symbols, fractions, and arrows.

Abenaki	Bosnian	Dutch	Haitian Creole
Afaan Oromo	Breton	English	Hän
Afar	Cape Verdean	Esperanto	Hawaiian
Afrikaans	Creole	Estonian	Hiligaynon
Albanian	Catalan	Faroese	Hopi
Alsatian	Cebuano	Fijian	Hotçak Latin
Amis	Chamorro	Filipino	Hungarian
Anuta	Chavacano	Finnish	Icelandic
Aragonese	Chichewa	Folkspraak	Ido
Aranese	Chickasaw	French	Igbo
Aromanian	Cimbrian	Frisian	Ilocano
Arrernte	Cofán	Friulian	Indonesian
Arvanitic Latin	Cornish	Gagauz Latin	Interglossa
Asturian	Corsican	Galician	Interlingua
Atayal	Creek	Ganda	Irish
Aymara	Crimean Tatar Latin	Genoese	Istro-Romanian
Azerbaijani	Croatian	German	Italian
Bashkir Latin	Czech	Gikuyu	Jamaican
Basque	Danish	Gooniyandi	Javanese Latin
Belarusian Latin	Dawan	Greenlandic Kalaallisut	Jèrriais
Bemba	Delaware	Guadeloupean	Kaingang
Bikol	Dholuo	Creole	Kala Lagaw Ya
Bislama	Drehu	Gwich'in	Kapampangan Latin

Kaqchikel	Nahuatl	Scottish Gaelic	Tzotzil
Karakalpak Latin	Ndebele	Serbian Latin	Uzbek Latin
Karelian Latin	Neapolitan	Seri	Venetian
Kashubian	Ngiyambaa	Seychellois Creole	Vepsian
Kikongo	Niuean	Shawnee	Vietnamese
Kinyarwanda	Noongar	Shona	Volapük
Kiribati	Norwegian	Sicilian	Võõro
Kirundi	Novial	Silesian	Wallisian
Klingon	Occidental	Slovak	Walloon
Kurdish Latin	Occitan	Slovenian	Waray-Waray
Ladin	Old Icelandic	Slovio Latin	Warlpiri
Latin	Old Norse	Somali	Wayuu
Latino sine Flexione	Oněipót	Sorbian Lower Sorbian	Welsh
Latvian	Oshiwambo	Sorbian Upper Sorbian	Wik-Mungkan
Lithuanian	Ossetian Latin	Sotho Northern	Wiradjuri
Lojban	Palauan	Sotho Southern	Wolof
Lombard	Papiamento	Spanish	Xavante
Low Saxon	Piedmontese	Sranan	Xhosa
Luxembourgish	Polish	Sundanese Latin	Yapese
Maasai	Portuguese	Swahili	Yindjibarndi
Makhuwa	Potawatomi	Swazi	Zapotec
Malay	Q'eqchi'	Swedish	Zarma
Maltese	Quechua	Tagalog	Zazaki
Manx	Rarotongan	Tahitian	Zulu
Māori	Romanian	Tetum	Zuni
Marquesan	Romansh	Tok Pisin	
Megleno-Romanian	Rotokas	Tokelauan	
Meriam Mir	Sami Inari Sami	Tongan,	
Mirandese	Sami Lule Sami	Tshiluba	
Mohawk	Sami Northern Sami	Tsonga	
Moldovan	Sami Southern Sami	Tswana	
Montagnais	Samoan	Tumbuka	
Montenegrin	Sango	Turkish	
Murrinh-Patha	Saramaccan	Turkmen Latin	
Nagamese Creole	Sardinian	Tuvaluan	

Character Map

Alternative possibilities.

Recursive includes a carefully-considered set of OpenType features that make it simple to design refined typography and to tailor code to suit personal preferences.

Off (default) On (ss03, ss05, ss02)

Off (default) On (ss01, ss06, ss02)

Mostly for code

Feature tag	Description	Off (default)	On
dlig	Code ligatures	=> && ===	⇒ && ≡
ss01	Single-story 'a'	JavaScript	J _a v _a Script
ss02	Single-story 'g'	Regex	R _e g _e x
ss03	Simplified 'f'	justify-self	j _u stify-s _e lf
ss04	Simplified 'i'	function	f _u nction
ss05	Simplified 'l'	null	n _u ll
ss06	Simplified 'r'	Browser	B _r owser
ss07	No-serif 'L' & 'Z'	nonZeroLib	n _o nZ _e roLib
ss08	Simplified 'at'	@font-face	@f _o nt-f _a ce
ss09	Simplified six & nine	6 6 6 9 9 9	6 6 6 9 9 9
ss10	Dotted zero	0x30	0x30
ss11	Simplified one	#123	#123

Mostly for design

Feature tag	Description	Off (default)	On
case	Caps punctuation	(I) I@I-I	(I) I@I-I
frac	Fractions	1/2 3/4	½ ¾
ordn	Ordinals	1.o 8.a	1. ^o 8. ^a
titl	No descender on 'Q'	SQL	S _Q L
sups, numr	Superiors	H0123456789	H ⁰ 1 ² 3 ⁴ 5 ⁶ 7 ⁸ 9
sinf, dnom	Inferiors	H0123456789	H ₀ 1 ₂ 3 ₄ 5 ₆ 7 ₈ 9

Sans-only

Feature tag	Description	Off (default)	On
zero, ss20	Slashed zero	0x30	0x30
pnum	Proportional one	#123	#123
liga	Italic ligatures	fi fl ffi ffl	f _i f _l f _{fi} f _{fl}

Designed for faster coding.

Default	dlig on	==	=	%%	%%	<-	<code><</code>	f"	f"
====	=====	&&	&&	##	##	<code>f'</code>	<code>f'</code>		
!=	<code>≠</code>	&&&	&&&	###	###	\$\{	\$\{		
!==	<code>≠≠</code>			####	####	?.	?.		
=/=	<code>≠≠</code>			//	//	?:	?:		
!!	<code>!!</code>	=>	<code>⇒</code>	>=	<code>≥</code>	/*	<code>/*</code>		
??	<code>??</code>	->	<code>→</code>	<=	<code>≤</code>	*/	<code>*/</code>		

Recursive includes a set of **code ligatures** supporting common multi-character combinations for JavaScript, Python, Markdown, CSS, and other programming languages. When the (`dlig`) OpenType feature is activated, these ligatures snap into place as you write code, automatically correcting spacing. This helps you to read, write, and understand code faster.

Default	dlig on	///	///	-<	-<	++	++	-=	-=
''''	''''	::	::	+++	+++	*=	*=		
""""	""""	>>	>>	--	--	/=	/=		
'''	'''	>>>	>>>	---	---				
<!--	<!—	<<	<<	**	**				
-->	<code>→</code>	<<<	<<<	***	***				
>-	<code>>-</code>	://	://	+=	<code>+=</code>				

```

1 // threeJS (MIT License), as minified for use on https://recursive.design
2 (function(h,Fa){"object"==typeof exports&&"undefined"!=typeof module?Fa(exports):(h=h||self,Fa(h.TH
3 function T(a,b,c,d,e,f,g,k,l,m){Object.defineProperty(this,"id",{value:mj++});this.uuid=0.generateUUID();this.name="";this.image=void 0!=a?a:T.DEFAULT_IMAGE;this.
4 this.center=new x(0,0);this.rotation=0;this.matrixAutoUpdate=!0;this.matrix=new za;this.generateMipmaps=!0;this.premultiplyAlpha=!1;this.unpackAlignm
5 c.minFilter,c.format,c.type,c.anisotropy,c.encoding);this.texture.image={};this.texture.image.width=a;this.texture.image.height=b;this.texture.generateMipmaps=void
6 4}function P(){this.elements=[1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1];0<arguments.length&&console.error("THREE.Matrix4: the constructor no longer reads arguments. use .se
7 d=new n(1,1,1);b._onChange(function(){c.setFromEuler(b,!1)});c._onChange(function(){b.setFromQuaternion(c,void 0,!1)});Object.defineProperties(this,{position:{conf
8 !1;this.layers=new $f;this.visible=!0;this.receiveShadow=this.castShadow=!1;this.frustumCulled=!0;this.renderOrder=0;this.userData={}})function pb(){D.call(this);th
9 -Infinity)}function ag(a,b,c,d,e){var f;var g=0;for(f=a.length-3;g<=f;g+=3){Vb.fromArray(a,g);var k=e.x*Math.abs(Vb.x)+e.y*Math.abs(Vb.y)+e.z*Math.abs(Vb.z),l=b.do
10 b?b:0}function ma(a,b,c){this.a=void 0!=a?a:new n;this.b=void 0!=b?b:new n;this.c=void 0!=c?c:new n}function y(a,b,c){return void 0==b&&void 0==c?this.set(a):
11 d.isVector3?d:new n;this.vertexNormals=Array.isArray(d)?d:[];this.color=e&&e.isColor?e:new y;this.vertexColors=Array.isArray(e)?e:[];this.materialIndex=void 0!=f?
12 this.blendDstAlpha=this.blendSrcAlpha=null;this.depthFunc=3;this.depthWrite=this.depthTest=!0;this.stencilWriteMask=255;this.stencilFunc=519;this.stencilRef=0;this
13 !1;this.toneMapped=this.visible=!0;this.userData={};this.version=0}function Qa(a){L.call(this);this.type="MeshBasicMaterial";this.color=new y(16777215);this.lightM
14 function K(a,b,c){if(Array.isArray(a))throw new TypeError("THREE.BufferAttribute: array should be a Typed Array.");this.name="";this.array=a;this.itemSize=b;this.c
15 b,c)}function Xb(a,b,c){K.call(this,new Uint16Array(a),b,c)}function Ed(a,b,c){K.call(this,new Int32Array(a),b,c)}function Yb(a,b,c){K.call(this,new Uint32Array(a)
16 this.uvsNeedUpdate=this.colorsNeedUpdate=this.normalsNeedUpdate=this.verticesNeedUpdate=!1}function wh(a){if(0==a.length)return-Infinity;for(var b=a[0],c=1,d=a.le
17 this.userData={}}function ca(a,b){D.call(this);this.type="Mesh";this.geometry=void 0!=a?a:new G;this.material=void 0!=b?b:new Oa({color:16777215*Math.random()});m);
18 m);$b.fromBufferAttribute(e,v);ac.fromBufferAttribute(e,p);e=a.morphTargetInfluences;if(b.morphTargets&&e){Le.set(0,0,0);Me.set(0,0,0);for(var q=0
19 xh(a,b,c,d,Zb,$b,ac,Gd))k&&(Ec.fromBufferAttribute(k,m),Fc.fromBufferAttribute(k,v),Gc.fromBufferAttribute(k,p),a.uv=ma.getUV(Gd,Zb,$b,ac,Ec,Fc,Gc,new x)),l&&(Ec.f
20 []);this.faces=[];this.faceVertexUvs=[];this.morphTargets=[];this.morphNormals=[];this.skinWeights=[];this.skinIndices=[];this.lineDistances=[];this.boundingSpher
21 e.isTexture)?b[c][d]=e.clone():Array.isArray(e)?b[c][d]=e.slice():b[c][d]=e}return b}function na(a){for(var b={},c=0;c<a.length;c++)var d=bc(a[c]),e;for(e in d)b
22 this.wireframeLineWidth=1;this.morphNormals=this.skinning=this.clipping=this.fog=!1;this.extensions={derivatives:!1,fragDepth:!1,draw
23 this.type="Camera";this.matrixWorldInverse=new P;this.projectionMatrix=new P}function pa(a,b,c,d){db.call(this);this.type="Persp
24 0,0));this.add(e);var f=new pa(90,1,a,b);f.up.set(0,-1,0);f.lookAt(new n(-1,0,0));this.add(f);var g=new pa(90,1,a,b);g.up.set(0,0,1);g.lookAt(new n(0,1,0));this.ad
25 "CubeCamera";this.update=function(a,b){null==this.parent&&this.updateMatrixWorld();var c=a.getRenderTarget(),d=this.renderTarget,p=d.texture.generateMipmaps;d.tex
26 f=this.renderTarget,g=0;6>g;g++)a.setRenderTarget(f,g),a.clear(b,c,d);a.setRenderTarget(e)}function Eb(a,b,c){va.call(this,a,b,c)}function cc(a,b,c,d,e,f,g,k,l,m,
27 d?d:new Ta,void 0!=e?e:new Ta,void 0!=f?f:new Ta}function yh(){function a(e,f){!1==c&&(d(e,f),b.requestAnimationFrame(a))}var b=null,c!=1,d=null;return{start:f
28 console.warn("THREE.WebGLAttributes: Unsupported data buffer format: Float64Array.")}d instanceof Uint16Array?c=5123:d instanceof Int16Array?c=5122:d instanceof Ui
29 (b=b.data);var d=c.get(b);d&&(a.deleteBuffer(d.buffer),c.delete(b)),update:function(d,e){d.isInterleavedBufferAttribute&&(d=d.data);var f=c.get(d);if(void 0==f)c
30 height:b,widthSegments:c,heightSegments:d};this.fromBufferGeometry(new dc(a,b,c,d));this.mergeVertices()}function dc(a,b,c,d){G.call(this);this.type="PlaneBufferGe
31 f=b+1+g*(a+1),k=b+1+g*a,v.push(b+g*a,e,k),v.push(e,f,k);this.setIndex(v);this.setAttribute("position",new C(p,3));this.setAttribute("normal",new C(q,3));this.setAt
32 d.background:h=a.xr;(h=h.getSession&&h.getSession())&&"additive"==h.environmentBlendMode&&(d=null);null==d?(e(f,g),m=null,v=0):d&&d.isColor&&(e(d,1),r!=0,m=null,
33 depthWrite:!1,fog:!1}),l.geometry.deleteAttribute("normal"),l.geometry.deleteAttribute("uv"),l.onBeforeRender=function(a,b,c){this.matrixWorld.copyPosition(c.matr
34 l.material,0,0,null)}else if(d&&d.isTexture){void 0==k&&(k=new ca(new dc(2,2),new oa({type:"BackgroundMaterial"},uniforms:bc(eb.background.uniforms),vertexShader:e
35 if(m!=d||v!=d.version)k.material.needsUpdate=!0,m=d,v=d.version;b.unshift(k,k.geometry,k.material,0,0,null)}))}function tj(a,b,c,d){var e=d.isWebGL2,f;this.setMo
36 return}d[g](f,k,l,m);c.update(l,f,m)}}function uj(a,b,c){function d(b){if("highp"==b){if(0<a.getShaderPrecisionFormat(35633,36338).precision&&0<a.getShaderPrecis
37 a instanceof WebGL2RenderingContext,g=void 0!=c.precision?c.precision:"highp",k=d(g);k!=g&&(console.warn("THREE.WebGLRenderer:",g,"not supported, using",k
38 getMaxAnisotropy:function(){if(void 0==e)return e;var c=b.get("EXT_texture_filter_anisotropic");return e=null!=c?a.getParameter(c.MAX_TEXTURE_MAX_ANISOTROPY_EXT)
39 0<e);c.numPlanes=e;c.numIntersection=0}function b(a,b,d,e){var f=null!=a?a.length:0,g=null;if(0!=f){g=m.value;if(!0!=e||null==g){e=d+4*f;b=b.matrixWorldInverse
40 0;this.init=function(a,c,g){var k=0!=a.length||c||0!=e||f;f=c;d=b(a,g,0);e=a.length;return k};this.beginShadows=function(){g=0;b(null)};this.endShadows=function
41 switch(c){case "WEBGL_depth_texture":var d=a.getExtension("WEBGL_depth_texture")||a.getExtension("MOZ_WEBGL_depth_texture")||a.getExtension("WEBKIT_WEBGL_depth_tex
42 agetExtension("WEBKIT_WEBGL_compressed_texture_s3tc");break;case "WEBGL_compressed_texture_pvrtc":d=a.getExtension("WEBGL_compressed_texture_pvrtc")||a.getExtensi
43 d);f.delete(e);if(k=g.get(a))b.remove(k),g.delete(a);c.memory.geometries--}function e(a){var c=[],d=a.index,e=a.attributes.position;if(null!=d){var f=d.array;d=d.
44 b}{var e=f.get(b);if(e)return e;b.addEventListener("dispose",d);b.isBufferGeometry?e=b:b.isGeometry&&(void 0==b._bufferGeometry&&(b._bufferGeometry=(new G).setFro
45 a.index;null==c&&b.version<c.version&&e(a)}else e(a);return g.get(a)}}}function yj(a,b,c,d){var e=d.isWebGL2,f,g,k;this.setMode=function(a){if=a};this.setIndex=fun
46 return}d[l](f,v,g,m*k,p);c.update(v,f,p)}}function zj(a){var b={frame:0,calls:0,triangles:0,points:0,lines:0};return{memory:{geometries:0,textures:0},render:b,pro
47 d}}}}function Aj(a,b){return Math.abs(b[1])-Math.abs(a[1])}function Bj(a){var b={},c=new Float32Array(8);return{update:function(d,e,f,g){var k=d.morphTargetInflue
48 p[1]=k[m];d.sort(Aj);for(m=k=0;8>m;m++){if(p=d[m])if(l=p[0],p=p[1])v&&e.setAttribute("morphTarget"+m,v[1]);f&&e.setAttribute("morphNormal"+m,f[1]);c[m]=p;k+=p;con
49 c.update(a.instanceMatrix,34962);return l},dispose:function(){e={}}}}function xb(a,b,c,d,e,f,g,k,l,m){a=void 0!=a?a:[];T.call(this,a,void 0!=b?b:301,c,d,e,f,void
50 this.minFilter=this.magFilter=1003;this.wrapR=1001;this.flipY=this.generateMipmaps=!1;this.needsUpdate=!0}function Kc(a,b,c){var d=a[0];if(0>d||0<d)return a;var e
51 c&&(c=new Int32Array(b),Bh[b]=c);for(var d=0;d!=b;++d)c[d]=a.allocateTextureUnit();return c}function Dj(a,b){var c=this.cache;c[0]=b&&(a.uniform1f(this.addr,b),
52 b.r){if(c[0]==b.r||c[1]==b.g||c[2]==b.b)a.uniform3f(this.addr,b.r,b.g,b.b),c[0]=b.r,c[1]=b.g,c[2]=b.b}else Pa(c,b)|| (a.uniform3fv(this.addr,b),Ia(c,b))}function

```

Definitions

<p>Bézier curves / Vectors → Bézier curves are a parametric or mathematical way to draw curves so that they can be scaled infinitely without losing resolution.</p> <p>Classification → A system used to organize typefaces into categories. There are many systems of classification for fonts, generally based on visual parameters and historical periods. Broadly speaking, typefaces fall into four categories: serif, sans-serif, script, and decorative (or display). Recursive's naming system is derived in part from the CSS 'font-family' property, which includes the generic categories 'sans-serif' and 'monospace'.</p> <p>Casual (see also *casual script* and *single-stroke casual*) → A genre of sign writing in which the letters are drawn with a brush, in just a few single strokes. Casual letters are often used in informal signage in businesses as well as on hot rods and commercial trucks.</p> <p>Casual script → A subgenre of casual lettering in which the letters are drawn with a brush and connected together.</p> <p>Character (see also *glyph*) → A letter, numeral, punctuation mark, or other sign included in a typeface. A character may have multiple glyphs within a font, such as a single-story and a double-story "g." The sum of characters included in a font is usually called a *character set*.</p> <p>Contours → A single path of any number of points. A glyph usually consists of one or more contours.</p> <p>Counters → Negative spaces inside letterforms, such as the center area of a "c" or the eye and mouth of an "e."</p> <p>Cursive (see also *script*) → A style of handwriting in which the letters have a flowing appearance..</p> <p>Diacritic / Accent → A mark or sign added to a letter to indicate a difference in pronunciation.</p>	<p>Duplexed (see also *superplexed*) → Duplexed fonts are two fonts which share the same width in every character, allowing styles to be used interchangeably.</p> <p>Expression → The personality or feeling conveyed by the design of a typeface.</p> <p>Glyph (see also *character*) → A glyph is the digital expression of a character. Glyphs are made of strokes and stems that are connected together. A font may contain more than one glyph for the same character, with non-default glyphs usually called *alternates*.</p> <p>Ink traps → A design feature of some typefaces traditionally used in print to trap excess ink and retain the sharpness of interior corners. Ink traps are still used in digital type to maintain the crisp appearance of corners by increasing the amount of light drawn into them by pixels. They are also used for their striking visual appearance at larger scales.</p> <p>Instance (see also *named instances*) → A specific style within a larger variable font.</p> <p>Interpolation → The process of generating font instances by means of calculating intermediate values between the control-point coordinates of two or more masters.</p> <p>Kerning → Corrections to the default spacing between two adjacent glyphs to make a typeface appear evenly spaced. For example, "VA" often has negative kerning.</p> <p>Letterform → The specific visual shaping of a character.</p> <p>Legibility (see also *readability*) → Legibility refers to the ease with which individual glyphs in a typeface can be perceived as the characters they represent, based on their appearance.</p> <p>Ligature → A glyph that includes two or more characters, often visually combined into a single shape. Ligatures can have different purposes: either purely decorative, to handle problems of spacing, or (in the case of code</p>	<p>ligatures), to help users understand and read code faster.</p> <p>Masters / Sources (see also *instances*) → Master fonts are drawn by a designer and are used to generate or interpolate in-between instances.</p> <p>Microinteractions → Subtle animations in digital user interfaces that respond to user interaction through visual changes such as color, size, and position.</p> <p>Monospaced font / Fixed-width font / Fixed-pitch font / Non-proportional font → A typeface in which all characters have the same width and occupy the same amount of horizontal space. This allows for perfectly aligned columns of text and is particularly useful when writing code.</p> <p>Named instances (see also *instances*) → Predefined styles in a variable font, such as "Bold Italic." Named instances are easy to use in typical text and design software because they can be selected through familiar font-style controls and menus.</p> <p>OFL → The SIL Open Font License 1.1 is one of the most common licenses for open-source fonts. It allows a font to be freely used and modified (so long as derivative fonts adopt the same OFL license). Its full conditions are described at http://scripts.sil.org/OFL</p> <p>OpenType → The modern format for digital font files, created in a joint effort between Microsoft and Adobe and contributed to by Google, Apple, many and others. Just as websites are built on the technical foundation of HTML, CSS, and JavaScript, OpenType is the technical specification that underlies modern fonts. The OpenType data format describes every aspect of a font, including glyph shaping, accent placement, kerning, font naming, and much more.</p> <p>OpenType features → Special features that a font may include to coordinate text layout and glyph substitution. These special features may include ligatures, alternate characters, uppercase-aligned punctuation,</p>	<p>tabular or proportional numbers, and different "stylistic sets". OpenType Features are controllable in most design software, code editors, and in CSS via the property 'font-feature-settings'.</p> <p>Proportional font → A typeface in which characters and spacings have different widths, each occupying a "natural" amount of horizontal space.</p> <p>Readability (see also *legibility*) → Readability refers to the ease with which text can be read in a given typeface and typographic context.</p> <p>Recursion (see also *recursion*) → When an object's definition contains a reference to itself. In programming, recursion happens when a function calls itself, using its outputs as inputs. More generally, recursion is a method of solving a problem where the solution depends on solutions to smaller instances of the same problem.</p> <p>Sans-serif / Sans → A style of type that excludes the small "feet" of serif fonts and typically has a low amount of contrast in stroke thickness.</p> <p>Script (see also *cursive*) → Classification for typefaces that imitate handwriting. In cursive scripts, letters are typically connected, but script typefaces can also have disconnected letters. *Script* is also a synonym for alphabetic systems. For example, the Latin alphabet is commonly called the Latin script.</p> <p>Semi-proportional font / Semi-monospaced font → A font designed with a limited set of widths for glyph spacing. While the glyphs in a proportional font have arbitrary widths based on what looks natural ("o" might be 623 units wide and "w" might be 871 units wide) and the glyphs in a monospaced font all have the same width (for example, "o" and "w" are both 600 units wide), in a semi-proportional font, glyphs are arranged into a few groups sharing the same glyph width. For example, most glyphs in a semi-proportional font might be 600 units wide,</p>
		68	69

<p>while the rest are 300 or 900 units wide. Recursive Sans uses spacing based on 50-unit increments (400, 450, 500, and so on) to deliver an aesthetic that falls somewhere between monospaced and natural.</p> <p>Stems (see also *strokes*) → A vertical stroke in a glyph.</p> <p>Strokes (see also *stems*) → Strokes are the different segments that collectively make up a glyph.</p> <p>Styles / Cuts → A font style is a specific variant comprised in a type family. The styles in a font family can differ in weight, width, slope, or other parameters.</p> <p>Single-stroke casual → A subgenre of casual lettering, where the letters are disconnected and usually written in all capitals.</p> <p>Subfamily → One of the font families included in a super family.</p> <p>Superfamily / Type system → A typeface consisting of various related subfamilies that fit into multiple classifications – for example, sans and serif companions, text and display counterparts. Because the different families in a type system share similar proportions and structure, they normally pair well together.</p> <p>Superplexed (see also *duplexed*) → In a superplexed family, all styles have consistent character widths and kerning.</p> <p>Terminals → The shape at the end of a stroke that doesn't include a serif. Terminals can be straight or curved.</p> <p>True italics → Unlike oblique italics, true italics have different letterforms than their upright counterparts. The letterforms of true italics are derived from cursive writing.</p> <p>UI (User Interface) → A system by which a human interacts with software. *UI* most often refers to visual elements of a Graphic User Interface (GUI) such as menus, buttons, and text input fields, and to Command Line</p>	<p>Interfaces (CLIs) as used in terminal-based, text-only software.</p> <p>Unicode → A standard for consistent encoding, representation, and handling of text in most of the world's languages (including, for example, Latin, Korean, Cherokee, and Emoji). Each character has its own Unicode codepoint: "A" is '0041', "a" is '0061', "♥" is '2665', and so on. These codepoints allow computers to properly represent (almost) any written language.</p> <p>UPM (Units Per eM) → The size of the coordinate grid in which the glyphs of digital fonts are drawn. Recursive has a UPM of 1000.</p> <p>Variable axis / Stylistic axis → A visual parameter along which a variable font can be adjusted, the most common being a font's weight (e.g. Light to Bold). Axes have four-letter tags which are commonly used to denote them in software. Several axes are registered in the OpenType spec and given lowercase abbreviations for tags, while non-registered axes must be assigned uppercase tags.</p> <p>Variable font → A single font file which includes a continuous range of styles. This allows the users to modify the visual parameters of the typeface by selecting different values on its axes of variation.</p>	<h2>Afterword</h2> <p>Google Fonts' mission is to make web typography better, more accessible, and more performant for everyone. We've invested deeply in variable fonts – supporting new variable typefaces, like Recursive, as well as in tools for the production, testing, and use of variable font technology – because this new technology has immense potential to improve digital typography.</p> <p>Variable fonts make it possible for web and app designers to leverage the full expressive power of large typefaces, while ensuring great performance on all kinds of networks. But variable fonts are more than just a way to save font download bandwidth on the web. We believe they will help designers and developers feel more empowered to express themselves typographically with more range and nuance, moving typography as a whole toward a future of greater elegance, creativity, and readability.</p> <p>Developers (from beginners to experts) are some of our core users, and Recursive is a typeface made especially for them. In apps, technical docs, & blogs, Recursive offers new possibilities for digital design & interaction, allowing you to mix tones and visual textures with a single font. It's also a beautiful, useful typeface for writing code in your favorite terminal or text editor, with the playful "Casual" styles, utilitarian "Linear" styles, and semi-cursive Italics.</p> <p>We hope you're as excited for the future of typography as we are, and we hope you enjoy designing and developing with Recursive!</p> <p>The Google Fonts team</p>	<p>70</p> <p>71</p>

Colophon

Recursive Sans & Mono

By Arrow Type

- Design by Stephen Nixon
- Contributions from Lisa Huang,
Katja Schimmel, and Rafał Buchner
- Font Mastering by Ben Kiel
- Early guidance from KABK TypeMedia

Editing

→ Noemi Stauffer

Specimen Design

Math Practice

- E Roon Kang
- Bon Hae Koo

Printing

Top Process, Inc.
Seoul, South Korea

Paper

- **Dust Jacket**
Snow White 150gsm with gloss finish
- **Cover**
Woodfree Printing Paper (Mojo) 180gsm
- **pp.3-30, 33-44, 47-74**
Woodfree Printing Paper (Mojo) 80gsm
- **pp.31-32, 45-46**
Yopo 90gsm

All images are courtesy of the editors and contributors.

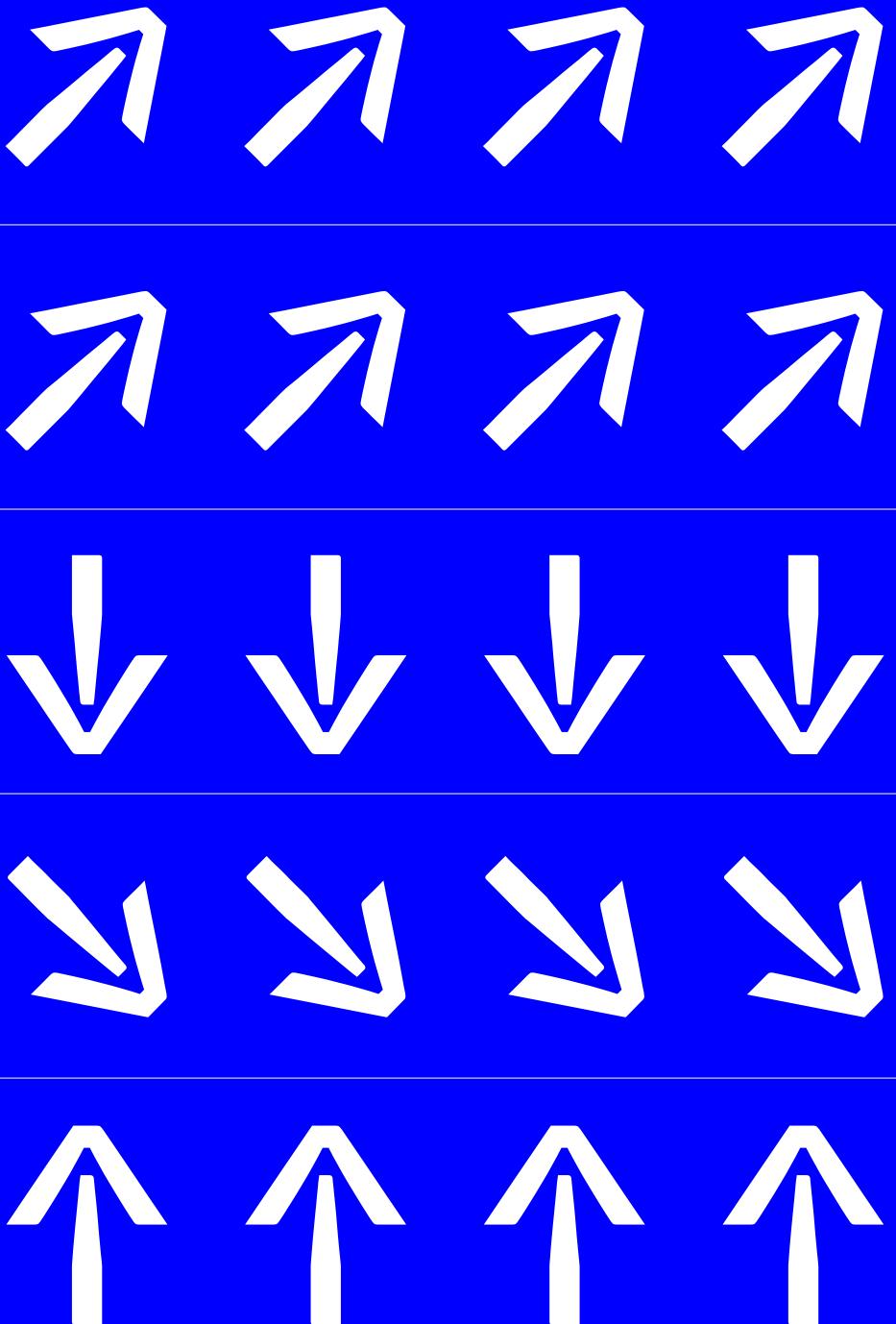
The views expressed in *Recursive Sans & Mono* are those of the authors and are not necessarily shared by Google, Inc.

Made by friends of

Google Fonts

@2020, Google, Inc.
All rights reserved.

Google and the Google logo are registered trademarks or service marks of Google, Inc.
All other trademarks are property of their respective owners.



MONO

1.0

CASL

1.0

wght

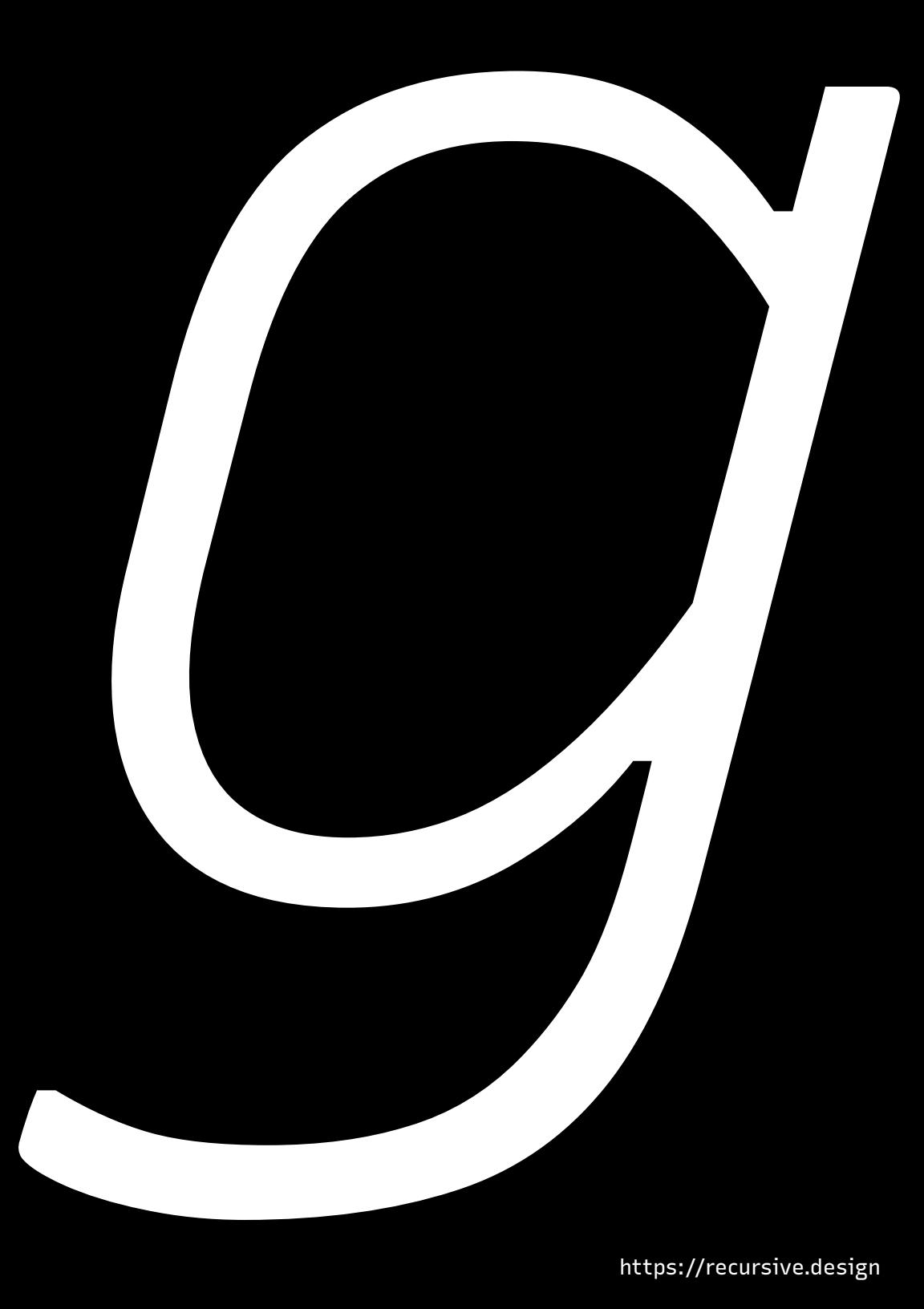
1000.0

sint

0.0

CRSV

1.0



1.0
1.0
1000.0
0.0
1.0

MONO
CASL
wght
slntr
CRSV

