monaspace

An innovative superfamily of fonts for code

Download >

Learn more

One superfamily.
Five fonts.
Three variable axes.

Since the earliest days of the teletype machine, code has been set in monospaced type – letters, on a grid. Monaspace is a new type system that advances the state of the art for the display of code on screen.

Every advancement in the technology of computing has been accompanied by advancements to the display and editing of code. CRTs made screen editors possible. The advent of graphical user interfaces gave rise to integrated development environments.

Even today, we still have limited options when we want to layer additional meaning on top of code. Syntax highlighting was invented in 1982 to help children to code in BASIC. But beyond colors, most editors must communicate with developers through their interfaces – hovers, underlines, and other graphical decorations.

Monaspace offers a more expressive palette for code and the tools we use to work with it.

Ne	Ar	Xe	Rn	Kr
Neon	Argon	Xenon	Radon	Krypton
Neo-grotesque sans	Humanist sans	Slab serif	*Handwriting*	Mechanical sans

JavaScript HTML CSS Java Python C++ PHP

Font size 16

Weight 300

Width 100

Slant 0

- [x] Texture healing
- [x] Ligatures
- [ ] Grid

Theme

GitHub Dark

What if?

Monospaced fonts are generally incompatible with one another. Each one uses different metrics, making it impossible to mix different fonts. Each Monaspace font is designed to be seamlessly mixed and matched. Layer more meaning onto code, with a palette that goes beyond colors and bolder weights. Build interfaces for code that require more structure and hierarchy.

```
// What if tentative ideas looked handwritten?

/**
 * What if docstrings looked authoritative?
 */
class Terminal_Dimensions {
  constructor() {
    this.width = process.stdout.columns;
    this.height = process.stdout.rows;
  }
}
```

What if Copilot had its own voice?

Ghost text makes it harder to parse and evaluate code suggestions. What if the typography made it clear?

What if Copilot spoke in its own typographical voice, and you could see which parts of your code it suggested after the fact?

```
// Before
const [isClicked, setClicked] = React.useState(false);
// After
const [isClicked, setClicked] = React.useState(false);
```

Texture healing

Monospaced type suffers from an inherent problem of uneven texture – text with some areas that are significantly denser, and some that have an excess of whitespace. It's an unavoidable consequence of trying to fit every letter into a uniform box, when some letters want more space, and some want less.

These shortcomings have been the same since the heyday of the Teletype machine in the 1960s. Texture Healing is a novel technique that evens out the density of monospaced type, bringing it closer to how proportional type has looked for centuries.

Texture healing preserves the monospace grid, and works in most editors without needing new software or editor plugins. How does it work?

In proportional typefaces, every glyph is designed with a unique width. The design of an `m` can be wider than the design of an `i`. Every glyph gets the space it needs, and no letter is stuck with too much or too little.

The design of each letter incorporates some amount of whitespace on both sides of a glyph. **These sidebearings create the visual room that help us**

distinguish one letterform from the next.

Too little, and the letters will run together. Too much, and the letters will look disconnected.

But in monospaced faces, every letter must be the same width, regardless of what that letter needs.

Adjusting the sidebearings isn't enough. **The letter must be altered to either fill up the box, or squeeze into it** – and still leave a little bit of room for sidebearings.

Narrow letters like `l` and `i` must be designed with exaggerated serifs to better fill the monospace box. Even with these synthetic additions, **these narrow letters have more space than they need.**

Wide letters like `m` and `w` are crammed into their boxes, and their shapes are visibly distorted compared to the rest of the alphabet. Their strokes are thinner, and the negative space inside the glyph is also compromised. There's

little room for these glyphs to grow bolder without turning into an unreadable blob.

Even though these letters have less space than they want, they still need to leave some room for sidebearings so they don't crowd the adjacent letters.

TEXTURE HEALING OFF

TIME_limit

TIME_limit

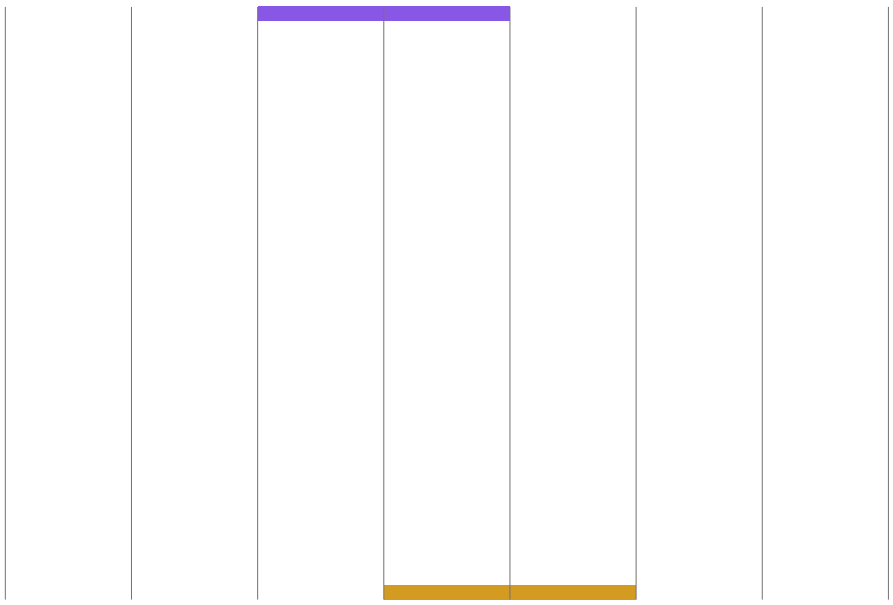
TEXTURE HEALING ON

Texture healing works by finding each pair of adjacent characters where one wants more space, and one has too much. Narrow characters are swapped for ones that cede some of their whitespace, and wider characters are swapped for ones that extend to the very edge of their box. This swapping is powered by an OpenType feature called "contextual alternates," which is widely supported by both operating systems and browser engines.

Contextual alternates are normally used for certain scripts, like Arabic, where the shape of each glyph depends on the surrounding glyphs. And they are also used for cursive handwriting fonts where the stroke of the "pen" might have different connection points across letters. Texture healing is a novel application of this technology to code. See our instructions for enabling texture healing in Visual Studio Code.

Step by step

Let’s unpack how texture healing is applied.



In the word “calming” there are two pairs that can be texture healed.

The `l` has space to give, and the `m` would benefit from more space.

In the first pass, the generic `l` is replaced with one that is narrower and shifted to the left, away from the `m`.

The generic `m` can then be replaced by a version that extends to the left edge of the space.

Monaspace contains alternates for glyphs that can shrink or grow in either direction.

But texture healing can also work in more than one direction.

The next pair of characters is also eligible for texture healing.

As before, the letter with space to give is replaced with one that cedes some whitespace to its neighbor.

This time, it shifts to the right.

And now the `m` is replaced once more, with a variant that extends in both directions.

- Texture healing finds the most visually compromised pairs and heals them with more legible replacements.

  The resulting text still obeys the monospaced grid.

640k styles ought to be enough for anyone

While the variable fonts support any combination you can choose, every Monaspace font also defines common named weights and styles for applications that do not yet support variable fonts. The three axes are **weight**, **width**, and **slant**.

The weight axis ranges from 200 to 800

200: Extra Light
300: Light
400: Regular
500: Medium
600: Semibold
700: Bold
800: ExtraBold

The slant axis ranges from 0 to -11°

The more negative the value, the more the letters are slanted. At the midpoint of the range some letters change their shapes to become true italics.

0: Normal
-5.5°: Swap obliques for italic letterforms
-11°: Italic

The width axis ranges from 100 to 125.

100: Normal
112.5: Semiwide
125: Wide

☐ Italic

Code ligatures

Monaspace includes code ligatures for a broad variety of languages, organized into stylistic sets that you can enable or disable according to your preferences.

Each stylistic set is roughly designed around the needs of specific languages. For example, `ss01` includes ligatures for character sequences commonly seen in JavaScript, and `ss05` provides ligatures for operators in F#.

You can enable as many or as few of them as you like.

In addition to the eight stylistic sets, there are two additional utility sets:

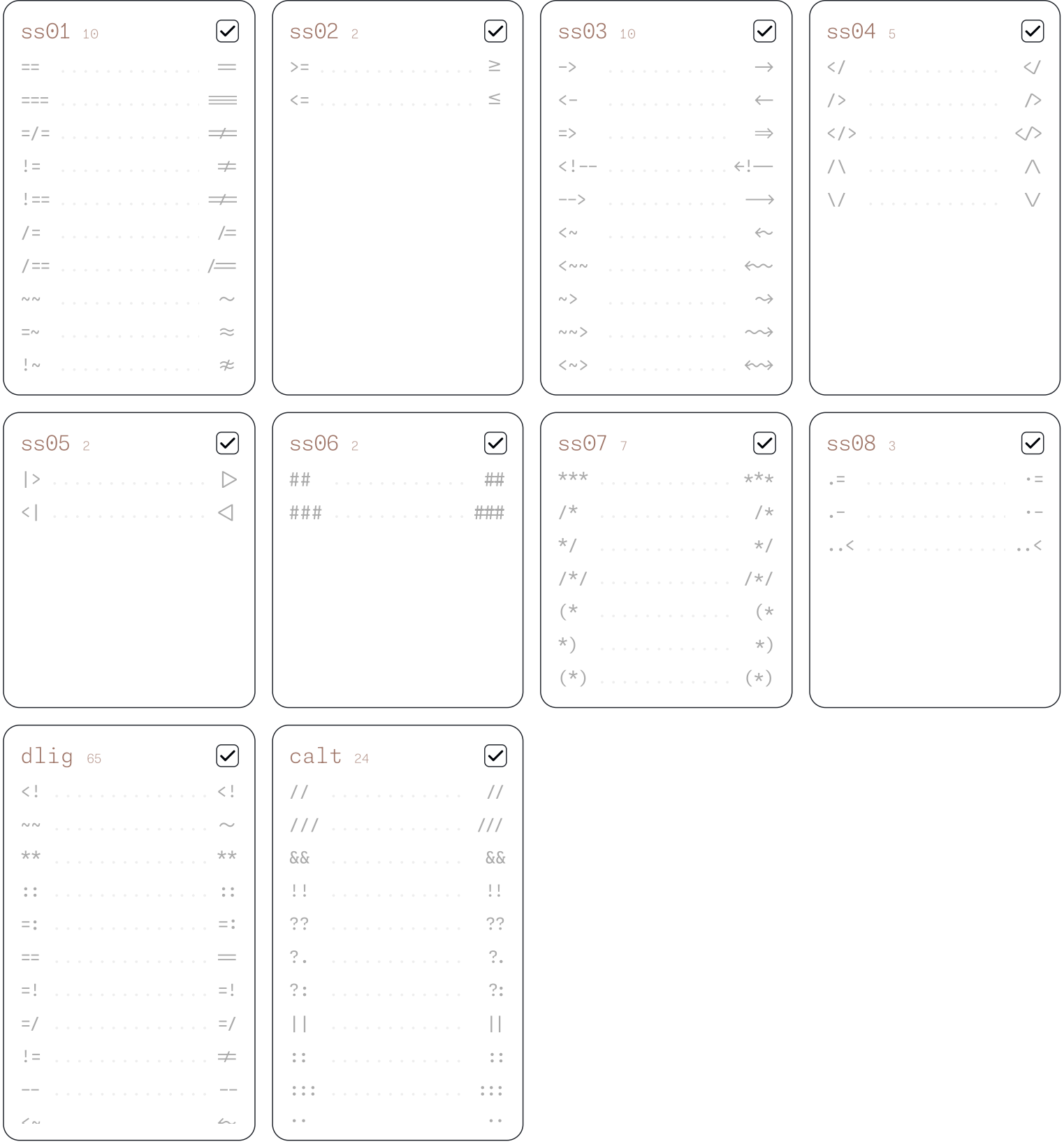
`calt` (contextual alternates) activates ligatures that adjust the visual positioning of some character sequences without altering their shape or appearance. Activating this feature will also enable texture healing.

`dlig` (discretionary ligatures) activates a basic set of ligatures that are shared by many programming languages and frameworks – mostly sequences of repeating characters.

Visual Studio Code

Choose the stylistic sets you want to enable, and copy the following line into your `settings.json`:

```
"editor.fontLigatures": "'ss01', 'ss02', 'ss03', 'ss04', 'ss05', 'ss06', 'ss07', 'ss08', 'calt', 'dlig'",
```



Contributors

Monaspace was made with the goal of improving all code, for all developers.

GitHub Next set out on this journey in 2022, and we were fortunate to find a type foundry that shares our passion for improving software in Lettermatic. The result is a marriage of form and function that opens the door to new developer

experiences, and that would not have been possible without the domain expertise and skill of the Lettermatic team, and the time they invested to work with GitHub Next on figuring out how typography ought to work for code.

For a full list of contributors, documentation, and the fonts themselves, visit the Monaspace repo.

What will you make with Monaspace?

Download >

v1.000