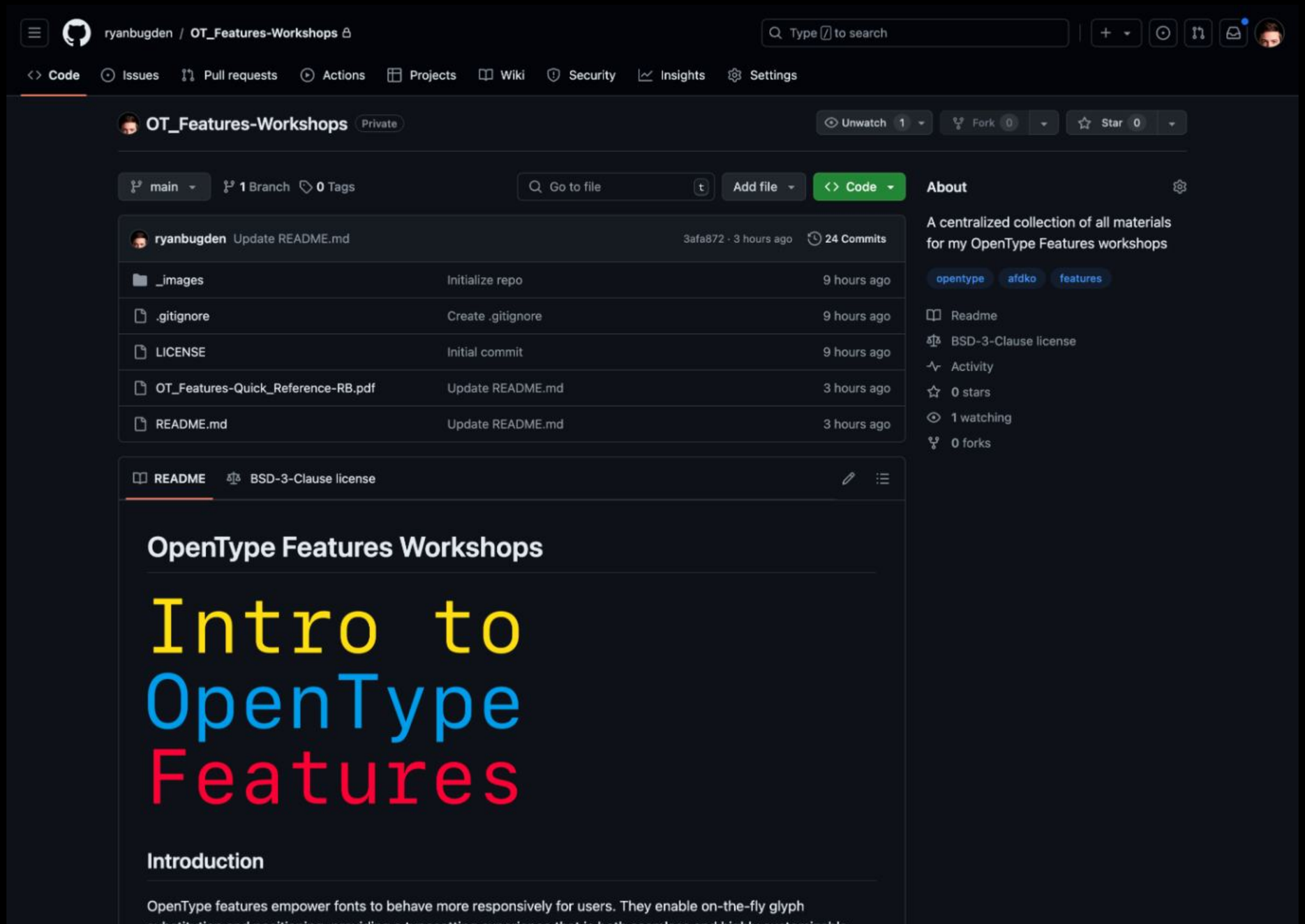
Intro to OpenType Features

Type@Cooper Condensed

Agenda

1 What OpenType Features are
2 How to write them
3 Bonus techniques

This deck and all resources are all at: github.com/ryanbugden/OT_Features-Workshops



What are OpenType Features?

Type chosen by hand

Elective

Photo by Patrick Ng

Type chosen by hand

Problem–solving

Photo by Ray Nichols

Type chosen by computer
Avoiding new problems

In retail fonts

In retail fonts

OPENTYPE FEATURES FAMILY WIDE	DEACTIVATED	ACTIVATED
ALL CAPS opens up spacing, moves punctuation up	¿¡Fish & 'Chips' for £2/$5!?	¿¡FISH & 'CHIPS' FOR £2/$5!?
PROPORTIONAL LINING default figures	Sale Price: $3,460 €1,895 Originally: ¥7,031 £9,215	Sale Price: $3,460 €1,895 Originally: ¥7,031 £9,215
TABULAR LINING	Sale Price: $3,460 €1,895 Originally: ¥7,031 £9,215	Sale Price: $3,460 €1,895 Originally: ¥7,031 £9,215
FRACTIONS ignores numeric date format	21/03/10 and 2 1/18 460/920	21/03/10 and 2 1⁄18 460⁄920
SUPERSCRIPT/SUPERIOR	x158 + y23 × z18 − a4260	$x^{158} + y^{23} \times z^{18} - a^{4260}$
SUBSCRIPT/INFERIOR	x158 ÷ y23 × z18 − a4260	$x_{158} \div y_{23} \times z_{18} - a_{4260}$
DENOMINATOR for making arbitrary fractions	0123456789 0123456789	0123456789 0123456789
NUMERATOR for making arbitrary fractions	0123456789 0123456789	0123456789 0123456789
LANGUAGE FEATURE Català (Catalan) l·l glyph	SÍL·LABA col·lecció cal·ligrafia	SÍL·LABA col·lecció cal·ligrafia
LANGUAGE FEATURE Română (Romanian) s accent	CONŞTIINȚA însuşi ştiinţifice	CONȘTIINȚA însuși științifice

OPENTYPE FEATURES ROMAN & ITALIC	DEACTIVATED	ACTIVATED
STYLISTIC SET 01 alternate a	Made an average of £112,450	Made an average of £112,450
STYLISTIC SET 02 alternate g	Regularizing storage regimen	Regularizing storage regimen
STYLISTIC SET 03 alternate M	Modifying 12-Cylinder Motors	Modifying 12-Cylinder Motors
STYLISTIC SET 04 alternate K k	Knowingly risky line of attack	Knowingly risky line of attack
STYLISTIC SET 05 alternate R	Bridges span the Rhine River	Bridges span the Rhine River
STYLISTIC SET 06 alternate W	Weathered from World War 2	Weathered from World War 2
STYLISTIC SET 12 round punctuation and accents	Their witty reëlection “jingle!”	Their witty reëlection “jingle!”

In retail fonts

Overview of supported OpenType layout

¡¿ab?! (doh-ehg) ▶ ¡¿AB?! (DOH-EHG)

Case Sensitive forms (CASE)
When function 'change to caps' is applied from within an application (not when text is typed in caps) appropriate case-sensitive forms are automatically applied. Regular brackets, parenthesis, dashes and hyphens are replaced with their capital forms.

012345 ▶ 012345
012345 ▶ 012345

Slashed Zero (ZERO)
Because in some circumstances '0', can be mistaken for an 'O', alternative forms of 'slashed zero' are available for all styles of figures

21/2 31/10 ▶ 2½ 3⅒

Arbitrary Fractions (FRAC)
Typotheque OpenType fonts already include a number of pre-designed fractions. Other arbitrary fractions are easily made by using the fraction feature.

({[H012-3456–789)]}
({[H012-3456–789]})
([{H012-3456–789}])
({[H012-3456–789]})

Tabular Lining Figures (TNUM+LNUM)
Tabular Oldstyle Figures (TNUM+ONUM)
Proportional Oldstyle Figures (PNUM+ONUM)
Proportional Lining Figures (PNUM+LNUM)
Neutral doesn't include true ranging (Oldstyle) figures, but use the Oldstyle substitution feature to access a set lowered numerals for use in a flow of lowercase. The standard Lining figures fit better with all-capital text. Tabular figures (TF) are for use in tables where numerals need to be aligned vertically. Tabular figures are available as a OpenType feature and have a fixed width in all weights. Typotheque fonts also include Old-style Tabular figures.

--> -> -^ ^- ▶ ⟶ → ↑ ↓

Discretionary Ligatures (DLIG)
The discretionary ligature feature creates real arrows when you type the combination -> (right arrow), <- (left arrow), -^ (up arrow) or ^- (down arrow). Discretionary ligatures are off by default in Adobe applications.

C2O2 ▶ $C^{2}O^{2}$

Superscript / superiors (SUPS)
Replaces all styles of figures (old style, tabular, lining) and letters with their superior alternates, which can be used for footnotes, formulas, etc. Superior characters are more legible than mathematically scaled characters, have a similar stroke weight, are spaced more generously, and better complement the rest of the text.

H2O ▶ $H_{2}O$

Subscript / inferiors (SINF)
Replaces all styles of figures (old style, tabular, lining) and letters with their inferior alternates, used primarily for mathematical or chemical notation. Inferior characters are more legible than mathematically scaled characters, have a similar stroke weight, are spaced more generously, and better complement the rest of the text.

a á å ä â ▶ ɑ á å ä â
Stylistic Set 1

ł ŧ ľ ȷ ĺ ŀ ▶ ɬ ŧ ľ ȷ ĺ ŀ
Stylistic Set 2

t ť ŧ ▶ ŧ ť ŧ
Stylistic Set 3

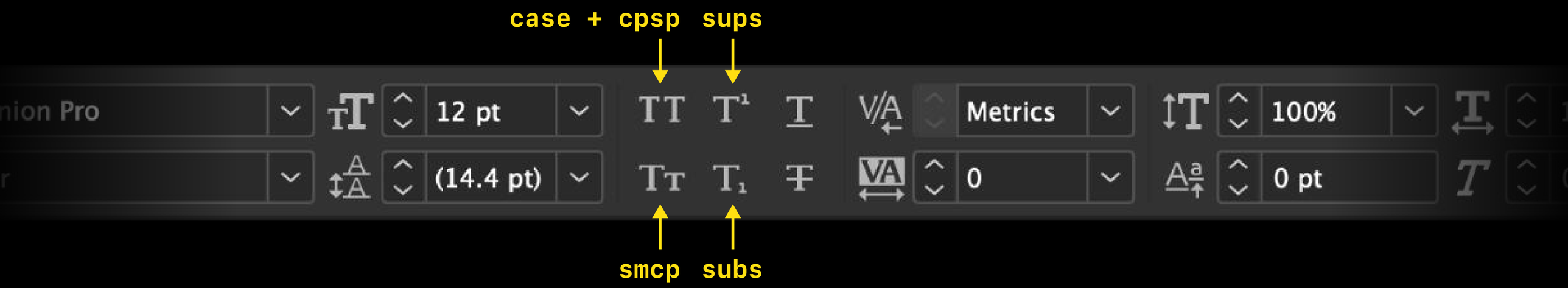
ß ▶ ß
Stylistic Set 4

Stylistic Alternates (SS01-04)
Neutral includes some alternative characters which can be activated by turning on 'stylistic alternates' in Adobe applications.

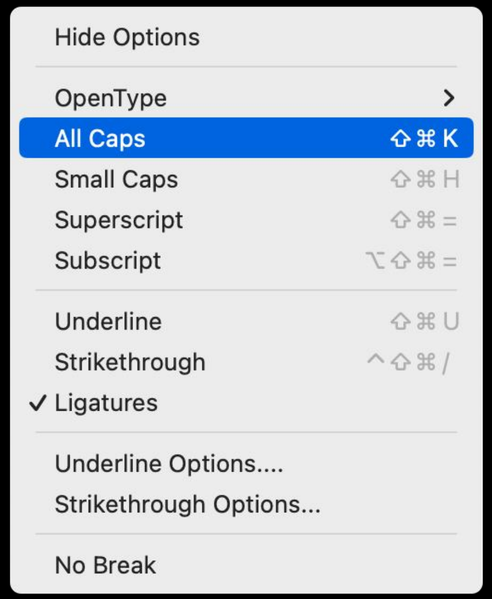
In InDesign

OpenType features are made by writing code.
Just a bit,
I promise.

Translation into the InDesign environment



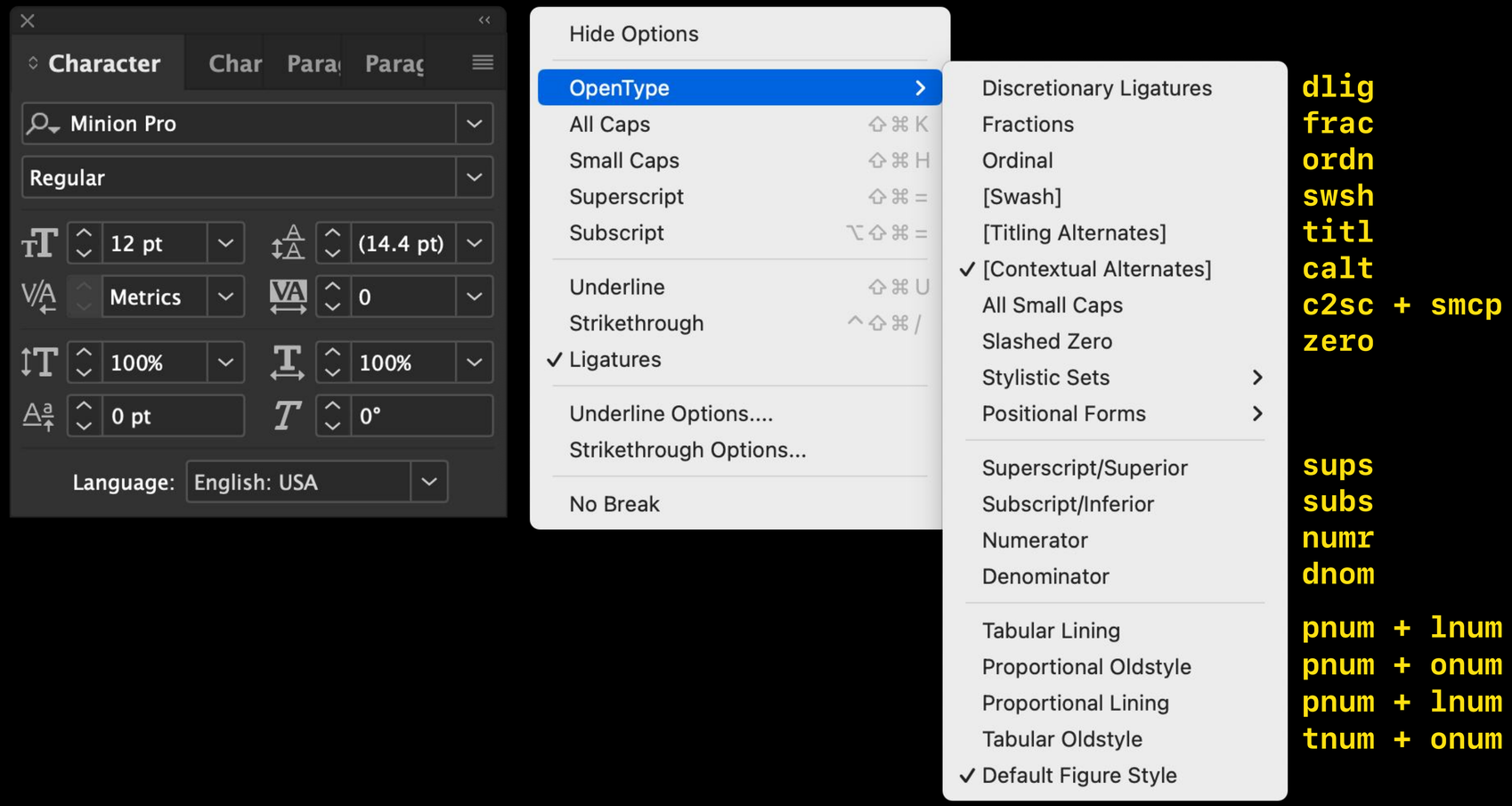
Translation into the InDesign environment



case + cpsp / InDesign might try to be smart
smcp / InDesign might try to be smart
sups / InDesign might try to be smart
subs / InDesign might try to be smart

liga

Translation into the InDesign environment

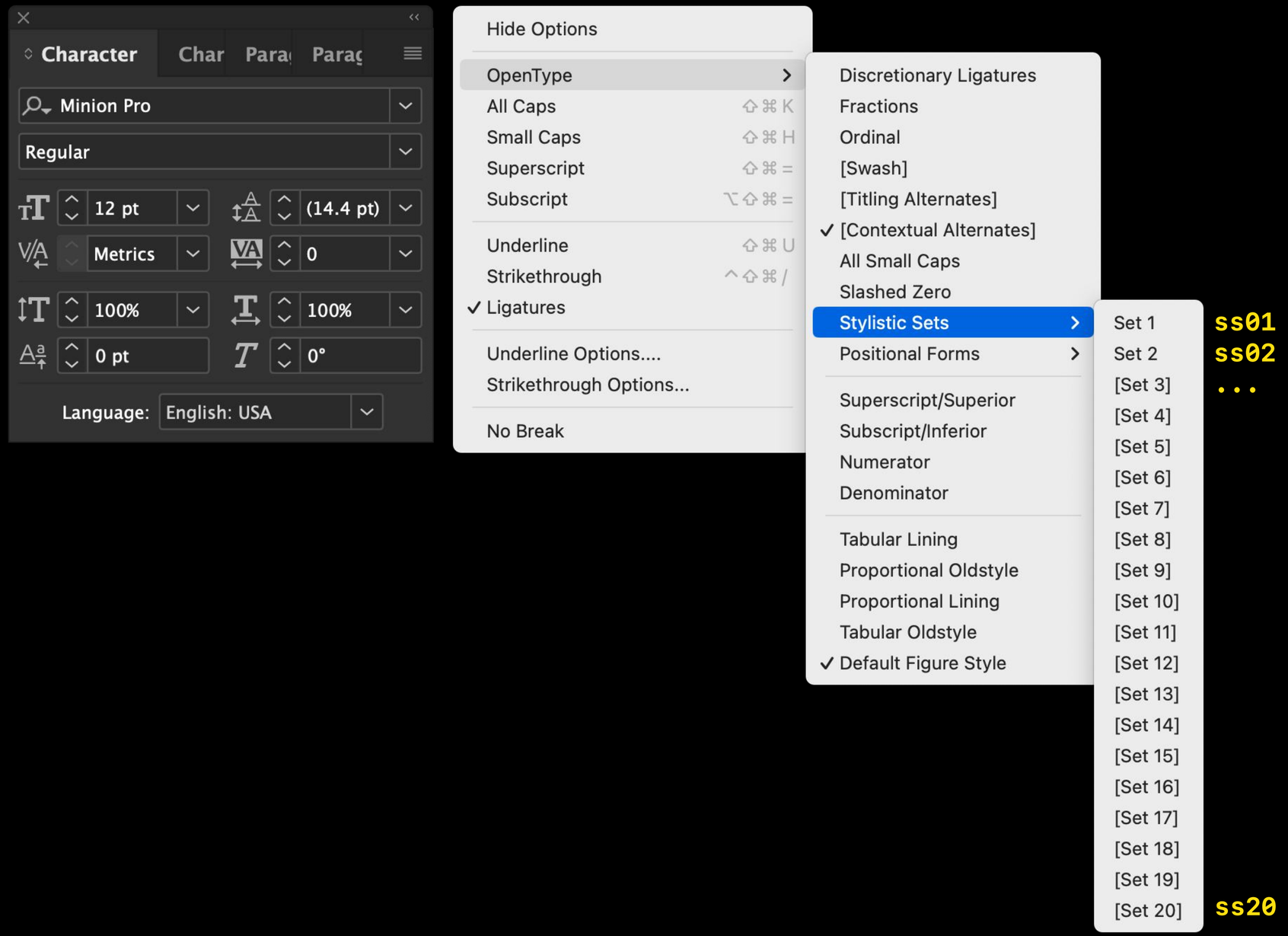


dlig
frac
ordn
swsh
titl
calt
c2sc + smcp
zero

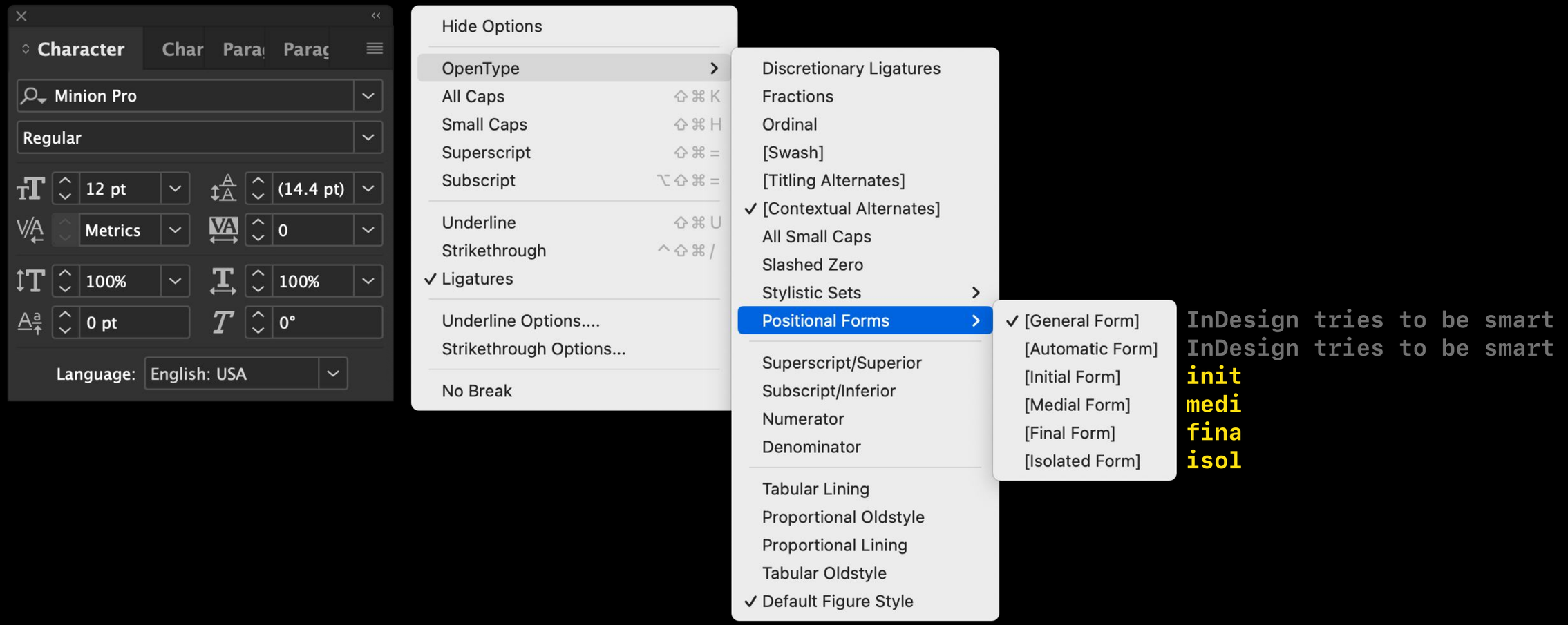
sups
subs
numr
dnom

pnum + lnum
pnum + onum
pnum + lnum
tnum + onum

Translation into the InDesign environment



Translation into the InDesign environment



Writing Your Feature File

The Goal

I want OpenType to replace the “a” glyph...

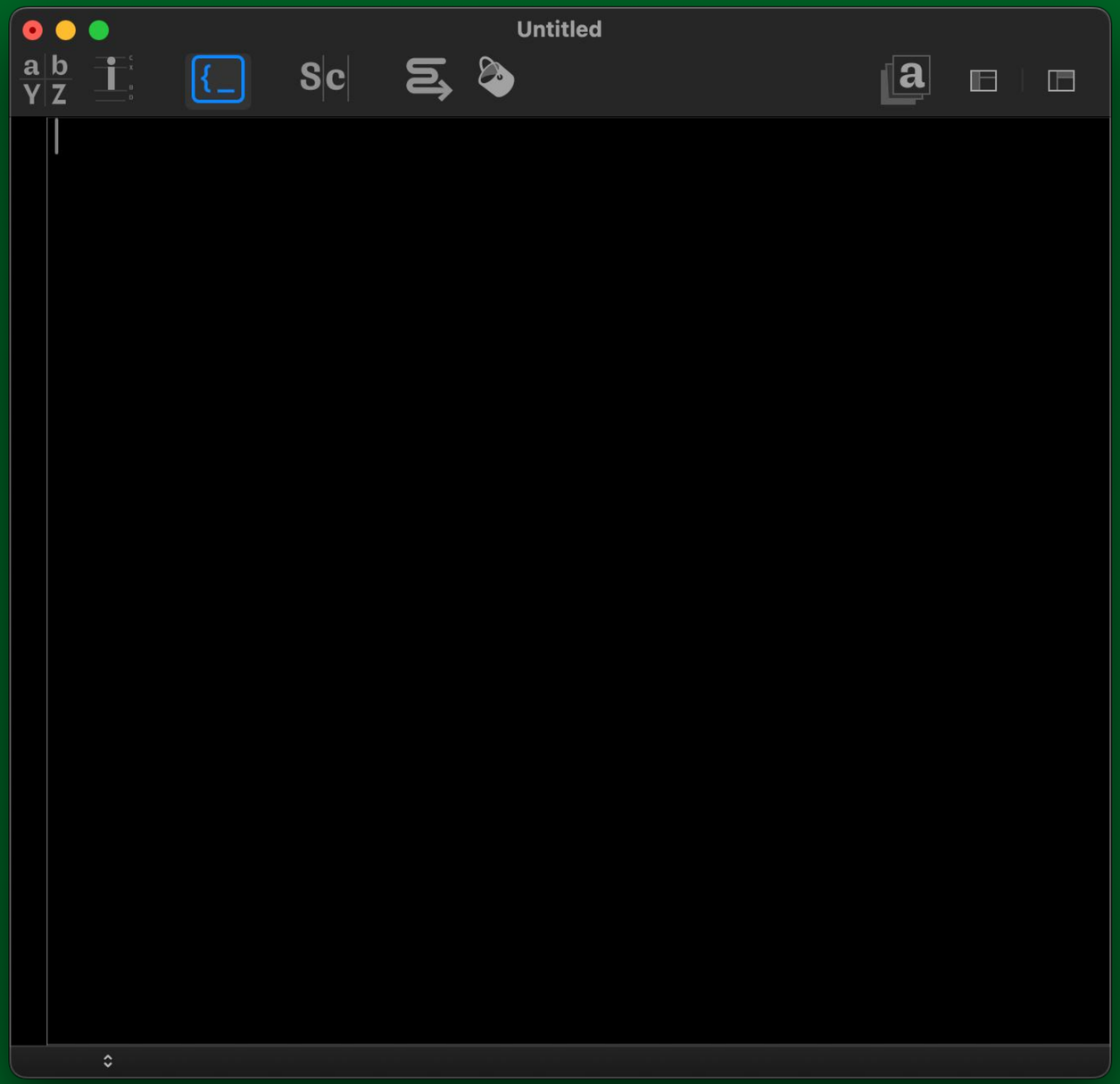
Fontastic

Fontastic

... with the “a.alt” glyph.

Writing your feature file

The Feature File



Commentary

In your UFO, you can navigate to your Features tab in order to start writing some stuff.

Writing your feature file

The Feature File

```
feature
```

Commentary

Writing “feature” initiates a feature.

Now we need to tell it which type of feature we want.

The Feature File

```
feature salt
```

Commentary

All feature tags are four letters long. I could have chosen the tag for standard ligatures (liga), small caps (smcp), or anything else. But I chose stylistic alternates (salt).

Writing your feature file

The Feature File

```
feature salt {

} salt;
```

Commentary

I will have to tell it when I'm done with writing this feature, so I have to book-end it with the tag again.

Curly braces are the container for content.

The semi-colon is a necessary marker for the end of an idea.

Writing your feature file

The Feature File

```
feature salt {    } salt;
```

Commentary

Spacing doesn’t matter in OpenType land. This would work, too.

Writing your feature file

The Feature File

```
feature salt {
    # I will fill this area
} salt;
```

Commentary

So far, the code that we have written can be read by the computer. But when you “comment” out messages with “#”, the compiler ignores it.

Let’s write the rules of our feature inside the {}.

Writing your feature file

The Feature File

```
feature salt {
    sub
} salt;
```

Commentary

The “sub” keyword means I want to substitute a glyph.

Writing your feature file

The Feature File

```
feature salt {
    sub a
} salt;
```

Commentary

I want to substitute the “a” glyph...

Writing your feature file

The Feature File

```
feature salt {
    sub a by
} salt;
```

Commentary

I want to substitute the “a” glyph with...

Writing your feature file

The Feature File

```
feature salt {
    sub a by a.alt
} salt;
```

Commentary

I want to substitute the “a” glyph with the glyph named “a.alt”.

Writing your feature file

The Feature File

```
feature salt {
    sub a by a.alt;
} salt;
```

Commentary

And I make sure to not forget to add the semi-colon.

Writing your feature file

The Feature File

```
feature salt {
    sub a by a.alt;
} salt;
```

Commentary

This is now complete OpenType Feature code!

If you generate this font, and turn its Stylistic Alternate feature on in (for example) Illustrator, it will replace your “a” glyph with your “a.alt” glyph whenever “a” is typed.

Substitution

- ✓ One glyph to one glyph
- Multiple to one
- Multiple to multiple
- One glyph, in context

Substitution

- ✓ One glyph to one glyph
- Multiple to one
- Multiple to multiple
- One glyph, in context

The Goal

I want OpenType to replace the “ffl”...

Ruffle

Ruffle

... with an “ffl” ligature.

Writing your feature file

The Feature File

```
feature liga {

} liga;
```

Commentary

We start with the “liga” feature if we want it to be considered by apps like InDesign to be a default/ standard ligature.

Otherwise, we’d use “dlig” (Discretionary Ligatures).

Let’s move forward with “liga”.

Writing your feature file

The Feature File

```
feature liga {
    sub f f l
} liga;
```

Commentary

The glyphs we substitute will have to be separated by spaces.

Writing your feature file

The Feature File

```
feature liga {
    sub f f l by f_f_l;
} liga;
```

Commentary

We want to replace all three of those glyphs with only one glyph.

Note: This only works if you've made a glyph with the name "f_f_l"

Writing your feature file

The Feature File

```
feature liga {
    sub f f l by f_f_l;
} liga;
```

Commentary

Done! Test it out.

Writing your feature file

The Feature File

```
feature liga {
    sub f f l by f_f_l;
    sub f f i by f_f_i;
    sub f f by f_f;
    sub f l by f_l;
    sub f i by f_i;
} liga;
```

Commentary

We can add all the ligature rules we want in the Standard Ligature feature!

Keep in mind, the app will read this in order, so longer substitutions should usually go first. In the word “ruffian”, you wouldn’t want “fi” swapped out before “ffi” has a chance.

Substitution

- ✓ One glyph to one glyph
- ✓ Multiple to one
- Multiple to multiple
- One glyph, in context

Substitution

- ✓ One glyph to one glyph
- ✓ Multiple to one
- Multiple to multiple
- One glyph, in context

The Goal

I want OpenType to replace both uppercase and lowercase with...

Alphabet

ALPHABET

... small caps.

Writing your feature file

The Feature File

```
feature smcp {

} smcp;


feature c2sc {

} c2sc;
```

Commentary

In order to change lowercase to small-caps, you would use the “smcp” feature tag.

But we also want to change uppercase to small-caps, so let’s use the “c2sc” feature tag, too.

Writing your feature file

The Feature File

```
feature smcp {
    sub [a b] by [A.sc B.sc];
} smcp;


feature c2sc {
    sub [A B] by [A.sc B.sc];
} c2sc;
```

Commentary

We will substitute each glyph with their equivalent small-cap glyph.

You can manage lists of glyphs with [brackets]. The compiler will pair the input/output list items intelligently.

Writing your feature file

The Feature File

```
@norm_lower = [a    b    c   ];
@norm_caps  = [A    B    C   ];
@small_caps = [A.sc B.sc C.sc];

feature smcp {
    sub @norm_lower by @small_caps;
} smcp;

feature c2sc {
    sub @norm_caps by @small_caps;
} c2sc;
```

Commentary

We can make things more manageable, readable, and reusable by storing collections of glyph names in “glyph classes”, using the @ character.

The Feature File

```
Demo time!
```

Commentary

Let’s have some fun with multiple-to-multiple substitution.

Substitution

- ✓ One glyph to one glyph
- ✓ Multiple to one
- ✓ Multiple to multiple
- One glyph, in context

Substitution

- ✓ One glyph to one glyph
- ✓ Multiple to one
- ✓ Multiple to multiple
- One glyph, in context

The Goal

I want OpenType to replace the “t”

Altogether

Altogether

... when it precedes an “o”.

Writing your feature file

The Feature File

```
feature calt {

} calt;
```

Commentary

This is an **alternate** glyph that we're swapping in, only when it's in a certain **context**. So, let's use the Contextual Alternates tag (calt).

Writing your feature file

The Feature File

```
feature calt {
    sub t’ o
} calt;
```

Commentary

As we begin to write this substitution, we can use a ‘ key to highlight which single glyph we intend to swap out.

Writing your feature file

The Feature File

```
feature calt {
    sub t’ o by t.alt;
} calt;
```

Commentary

And now all we do is write the single replacement glyph name.

Writing your feature file

The Feature File

```
feature calt {
    sub t’ o by t.alt;
} calt;
```

Commentary

Done! Test it out.

The Feature File

```
feature calt {
    ignore sub t’ o n;
    sub t’ o by t.alt;
} calt;
```

Commentary

What if a longer word contains the short word?

For example, we want to change the t in “to” but not in “ton”.

Let’s **ignore** it!

Writing your feature file

The Feature File

```
feature calt {
    sub t’ o by t.alt;
    sub d’ by d.alt;
} calt;
```

Commentary

Note: If you use a contextual ’ rule or exception within a lookup, all of the rules within that lookup must also use the ’ on the target of the rule.

Writing your feature file

The Feature File

```
Demo time!
```

Commentary

Let’s have some fun with contextual alternates.

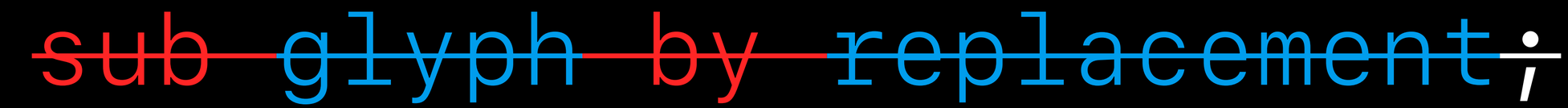
Substitution

- ✓ One glyph to one glyph
- ✓ Multiple to one
- ✓ Multiple to multiple
- ✓ One glyph, in context

- [x] Substitution
- [ ] Positioning

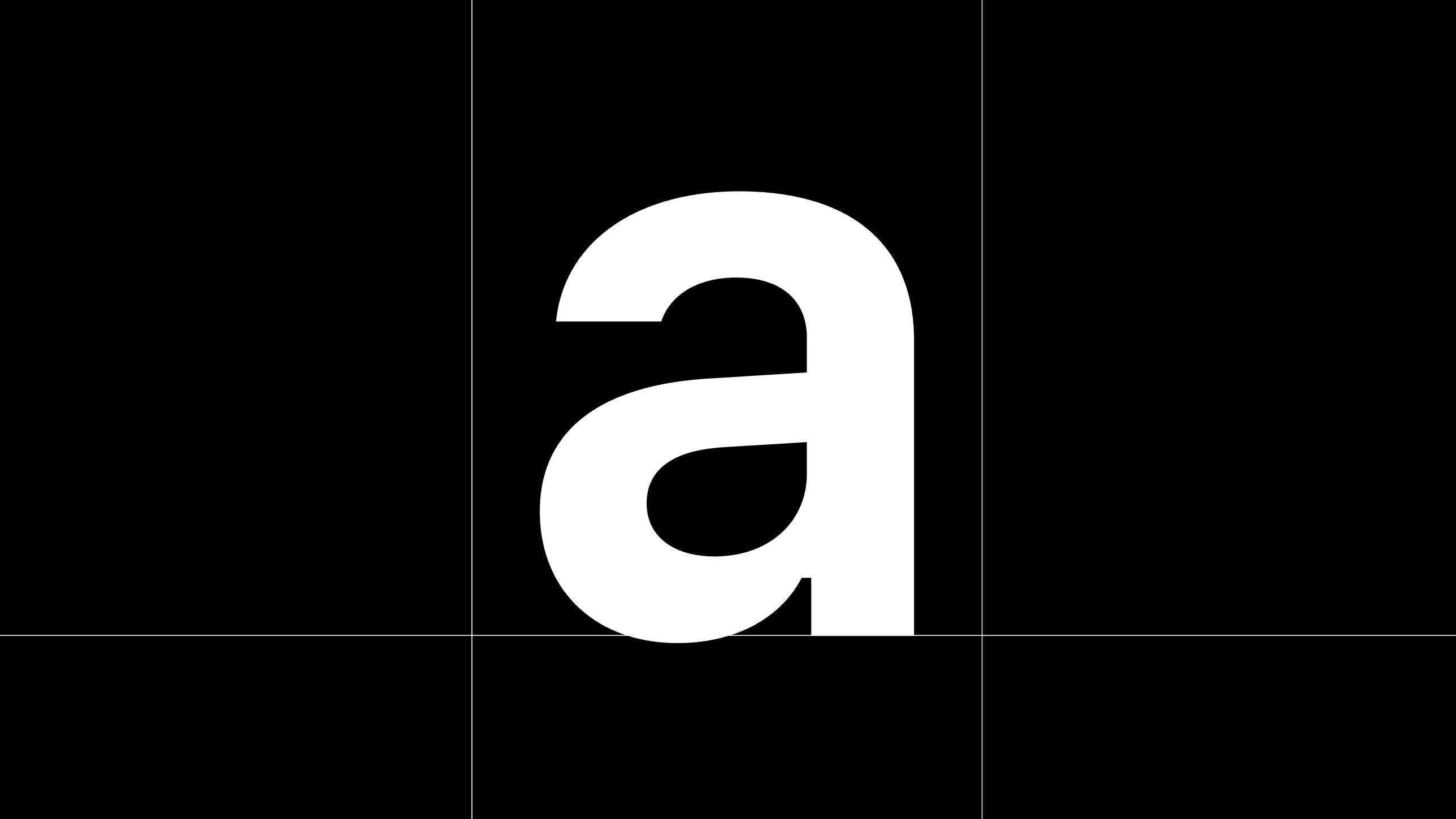
- [x] Substitution
- [ ] Positioning

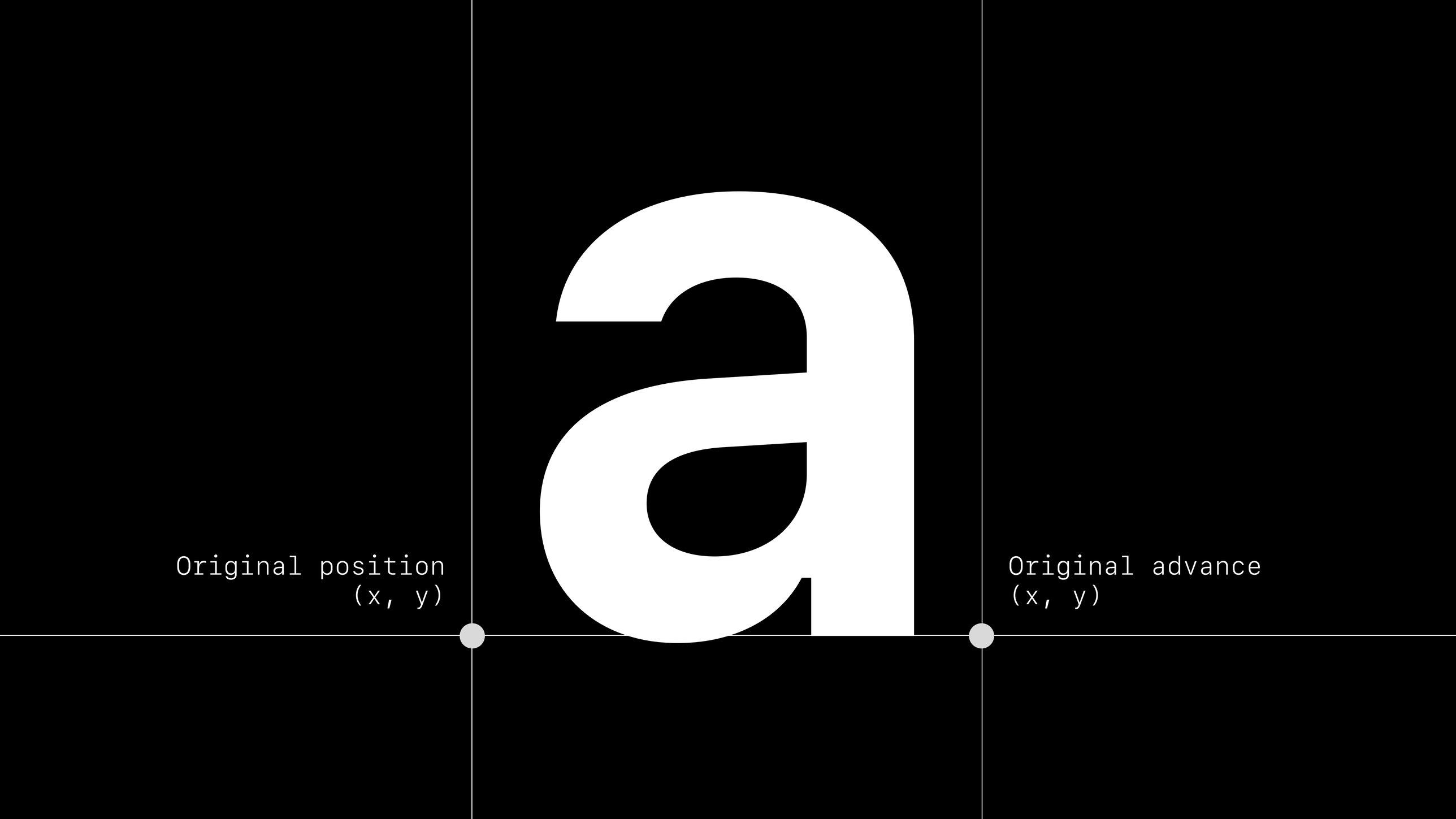
```
sub glyph by replacement;
```

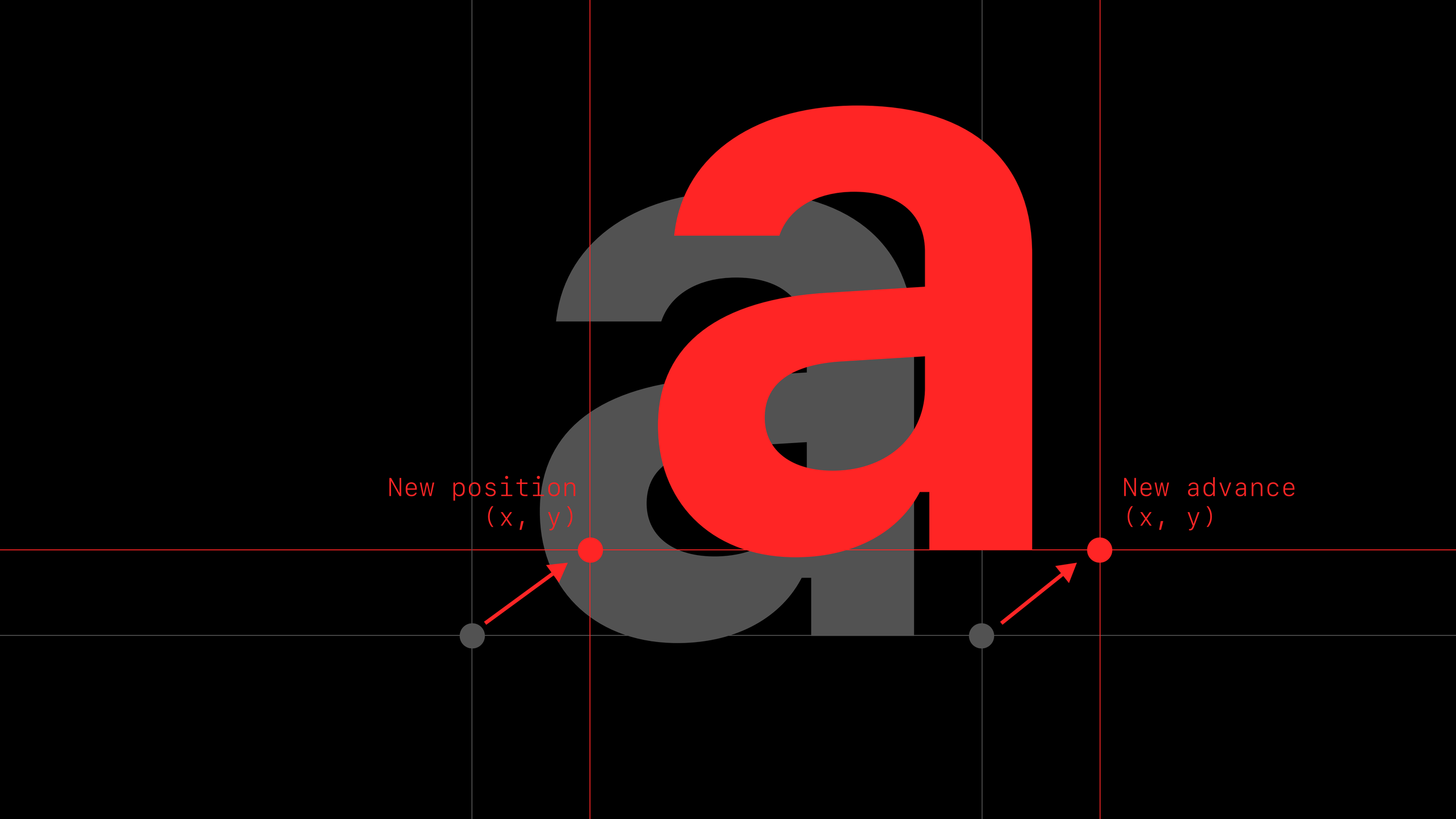

sub glyph by replacement;

```
pos glyph <position_value>;
```

```
pos glyph <x_position y_position x_advance y_advance>;
```


a


Original position
(x, y)
a
Original advance
(x, y)


New position
(x, y)
New advance
(x, y)

Writing your feature file — Positioning

The Feature File

```
@caps = [A  B  C];

feature cpsp {
    pos @caps <10 0 20 0>;
} cpsp;
```

Commentary

This would take the glyphs A, B, and C, and add 10 units to the left and right of them.

Writing your feature file — Positioning

The Feature File

```
Demo time!
```

Commentary

Let’s have some fun
with positioning.

Bonus Writing Techniques

Writing your feature file

The Feature File

```
languagesystem DFLT dflt;
languagesystem latn dflt;
```

Commentary

It’s a good idea to specify the font’s script at the top of the features.

Writing your feature file

The Feature File

```
include(features.fea);
```

Commentary

Instead of writing the features directly into your UFO, you can simply refer to an external .fea file.

What’s written in your UFO directly can be just this.

Writing your feature file

The Feature File

```
lookup my_cool_lookup {
    sub a by a.alt;
} my_cool_lookup;


feature ss01 {
    lookup my_cool_lookup;
} ss01;


feature salt {
    lookup my_cool_lookup;
} salt;
```

Commentary

Instead of writing the same rules more than once, you can store them in what’s called a “lookup” and reference them later, as many times as you want.

1
2
3
4
5
6
7

Enjoy!