What even are programmable fonts

A look at Harfbuzz's new WebAssembly support

Valdemar Erk

2023-11-30

Outline

- Introduction To Fonts
- Computer Fonts
- Animations* With Fonts
- How Far Can We Go?

Introduction To Fonts

Early history - Cuneiform

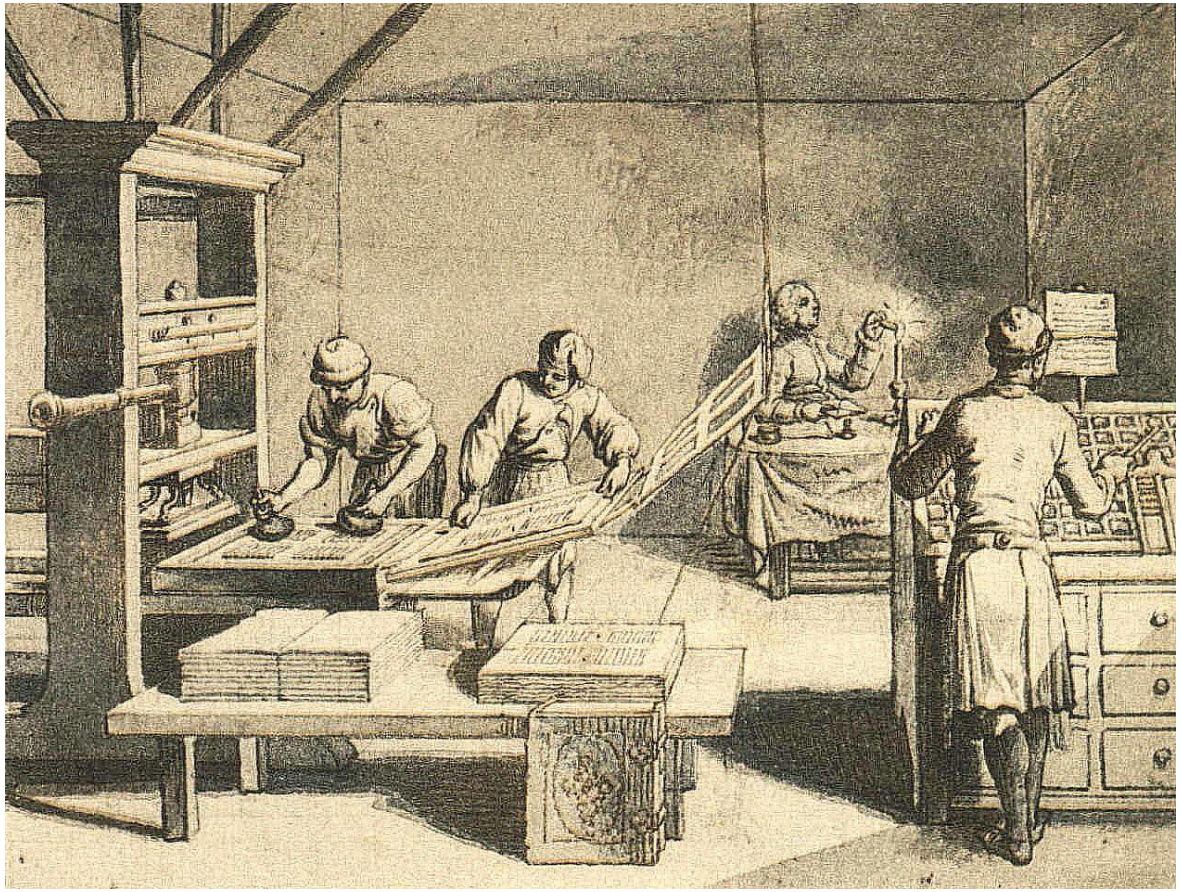
- Sumerian contract: selling of a field and a house.
- Found in present day Iraq.
- Circa 2600 BC.

Early history - Runes

Runes used “bind runes” that put several runes together.

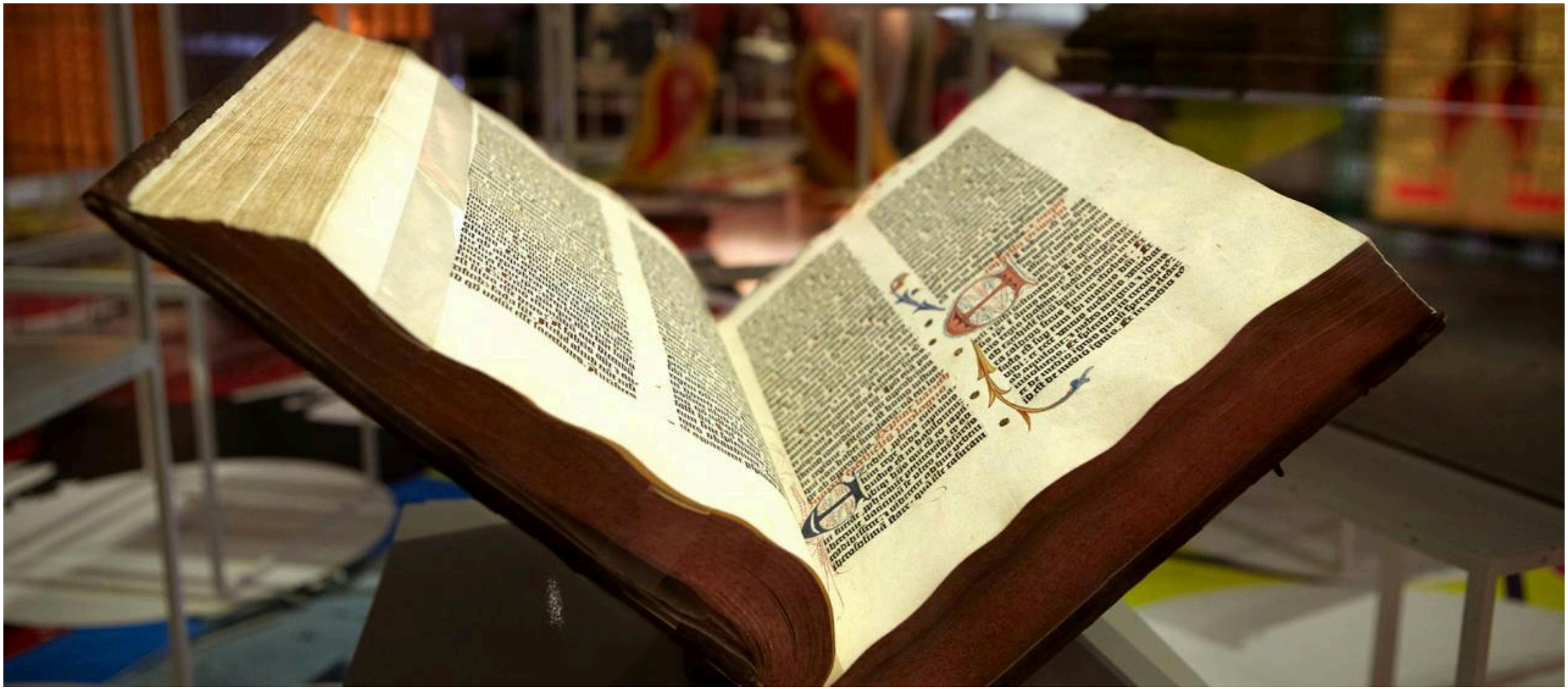
Here it is “ᛞᛞ”

Early history - Moveable metal types

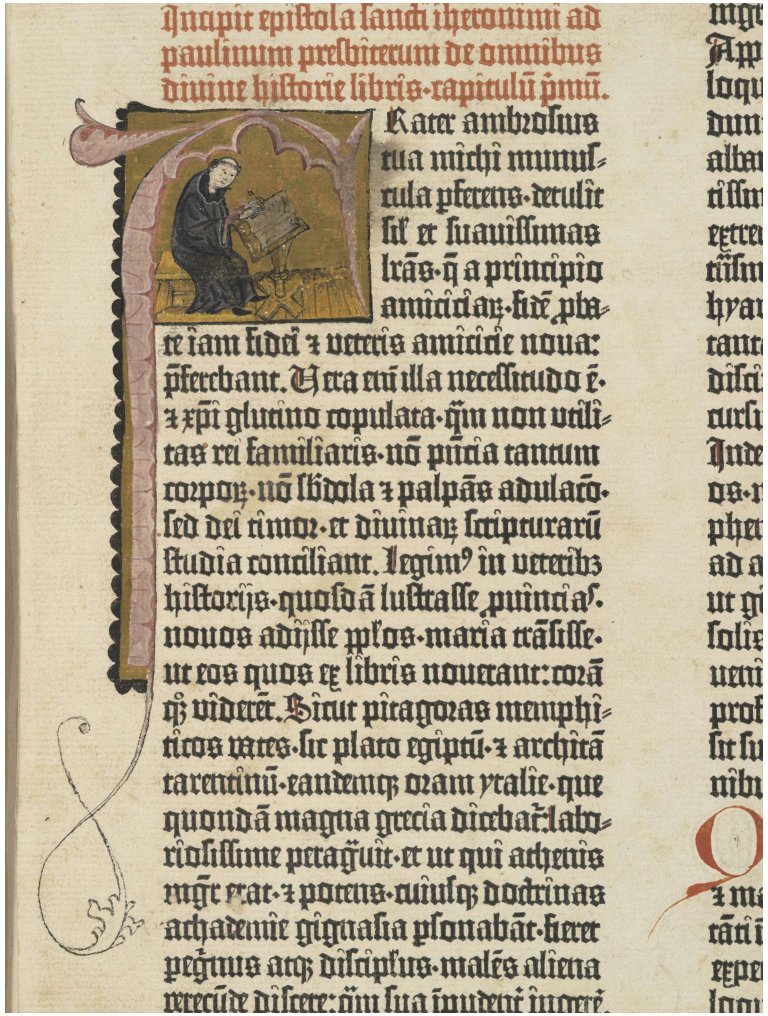


- Moveable types of metal had been invented in Korea 71 years earlier, but they differ a lot from the ones Gutenberg made later.
- Moveable wooden types was in use for longer.

Early history - The Gutenberg Bible

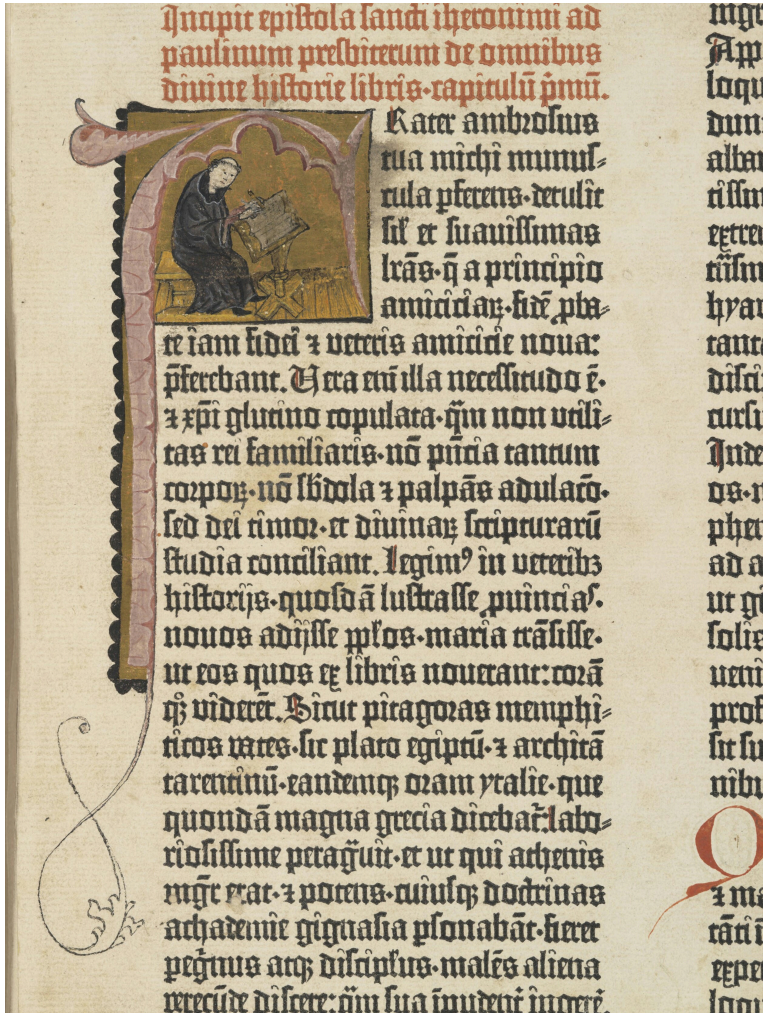


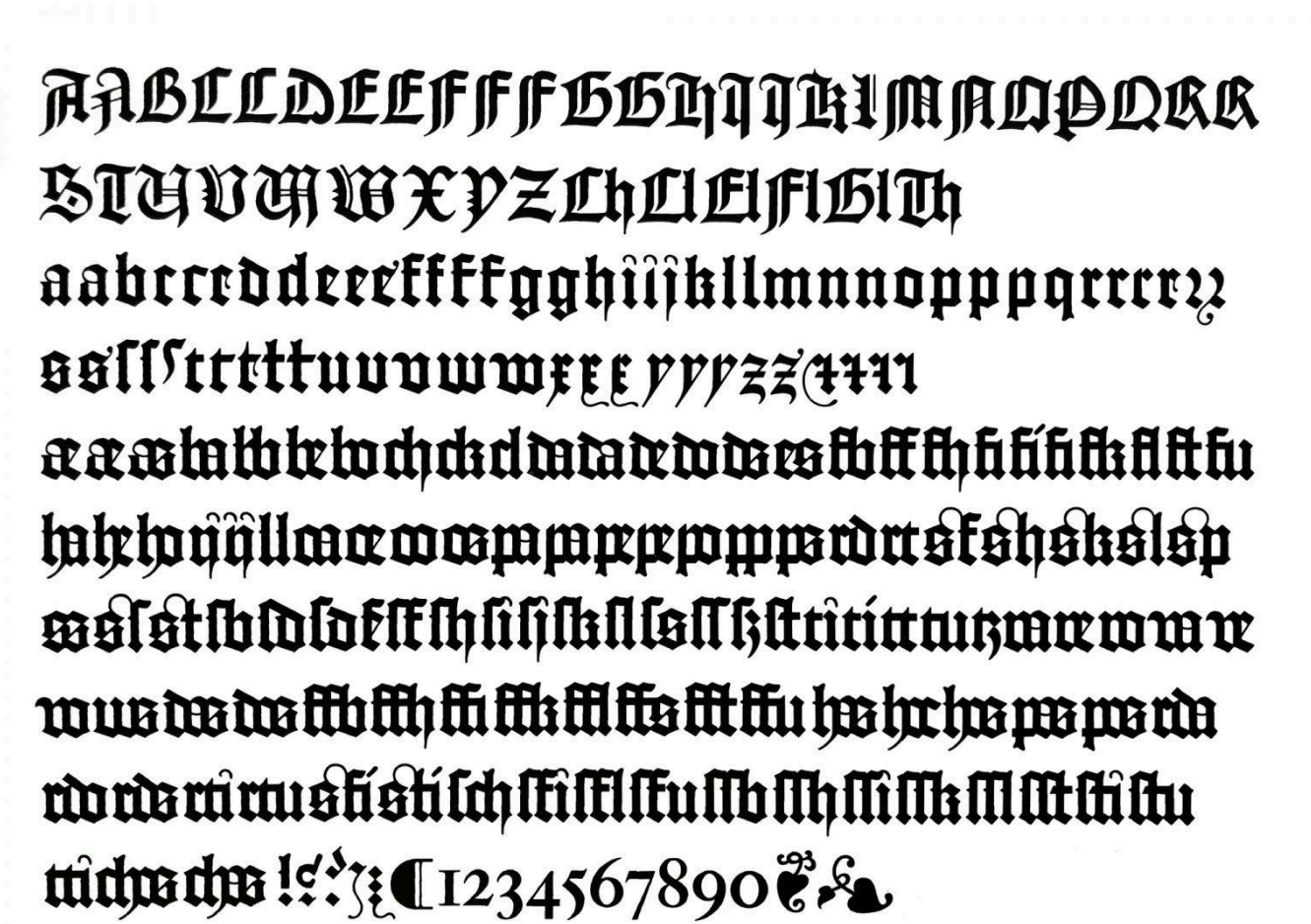
Early history - The Gutenberg Bible



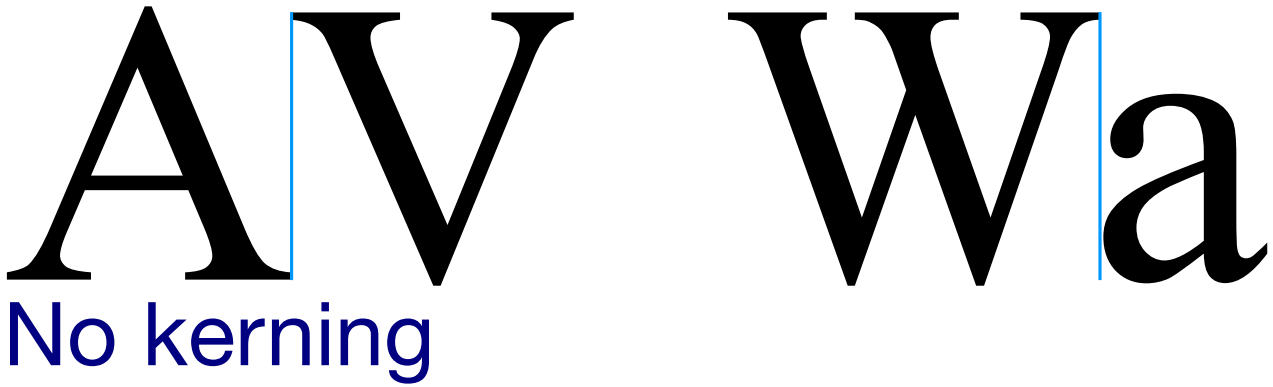
- Set with the type known as Textualis or Donatus-Kalender.
- Has a large amount of ligatures.
- Later replaced with the Fraktur types.

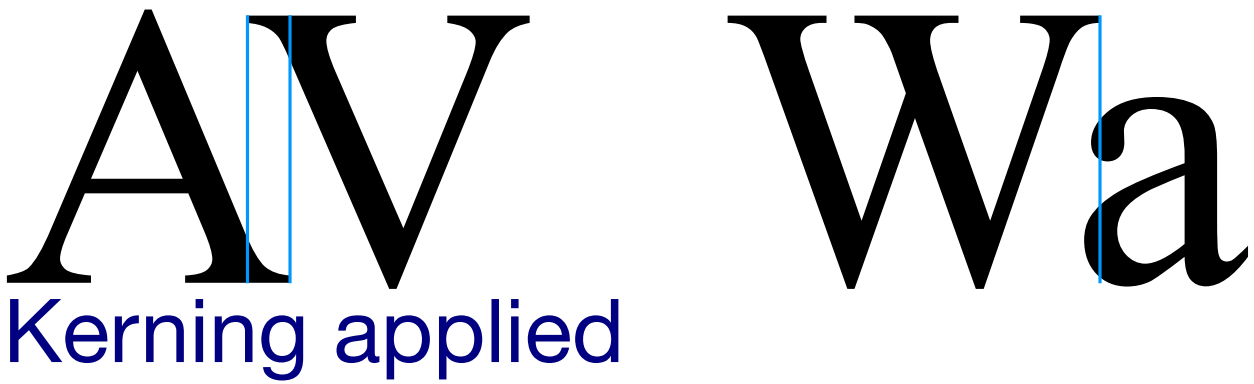
Early history - The Gutenberg Bible



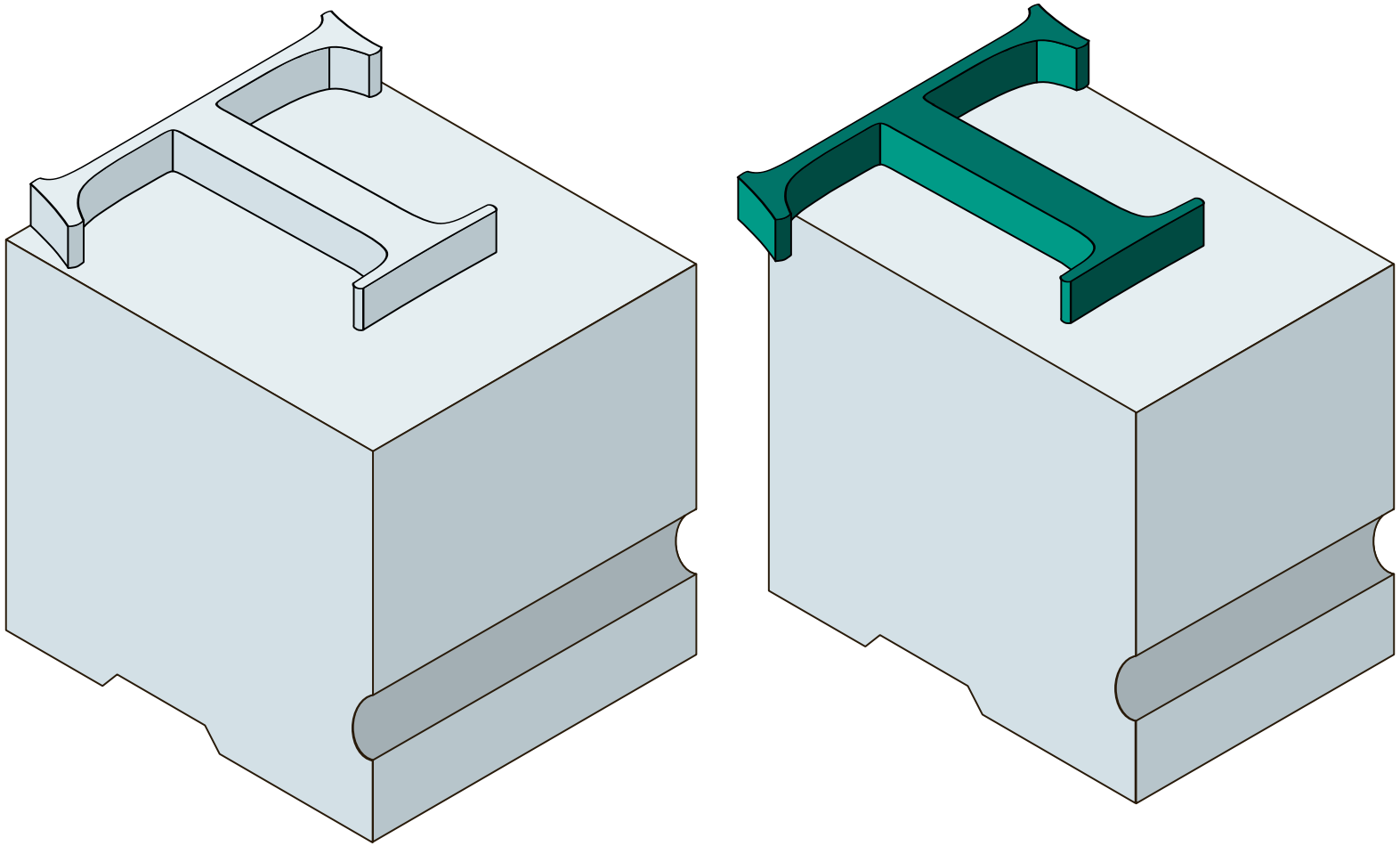


Kerning





Kerning - Cast types



Kerning - Cast types

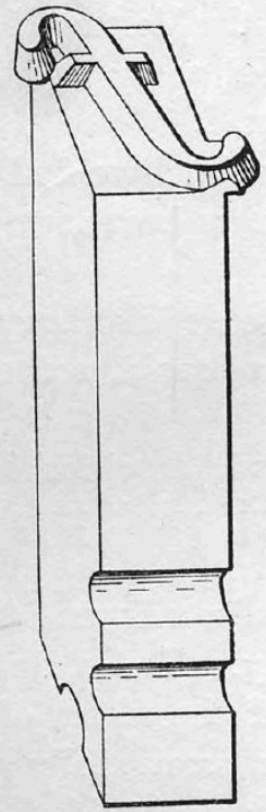


FIG. 56.—*Kerned type, as cast: isometric view.* About 2¼ times full size.

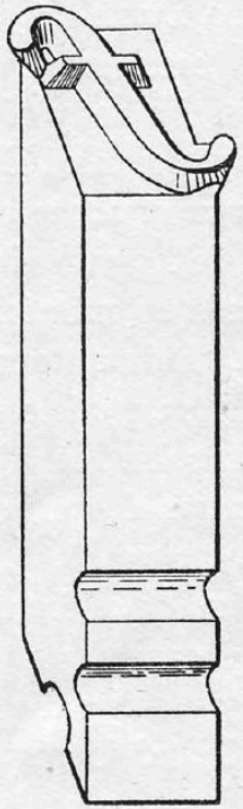
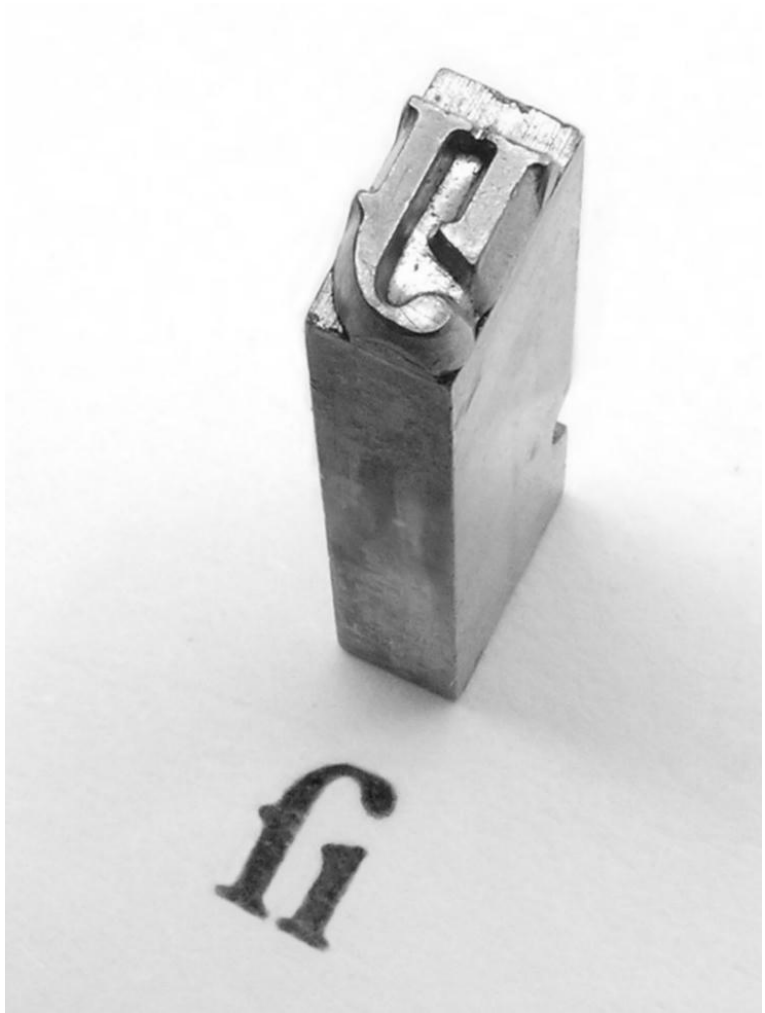
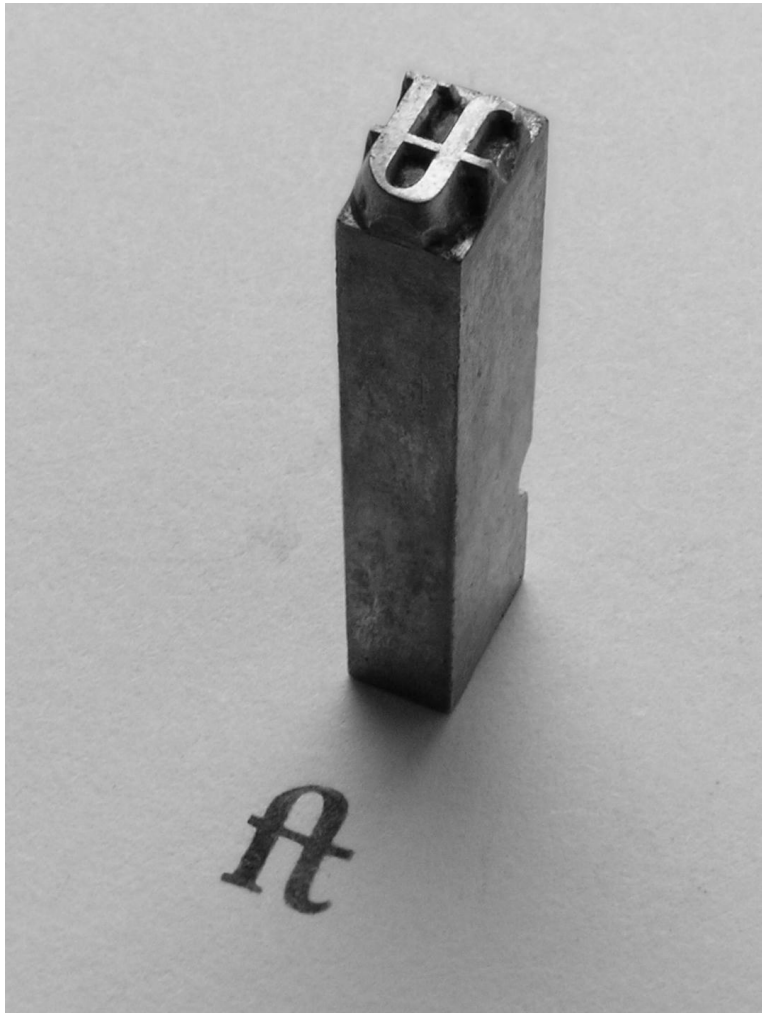


FIG. 57.—*Kerned type, after dressing: isometric view.* About 2¼ times full size.

Sidenote - Composing stick

Ligatures





Computer Fonts

Bitmap Fonts

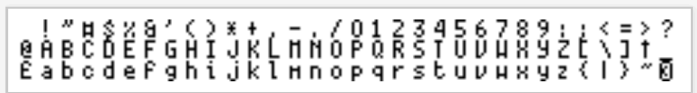
Altered Chrome

Anarchist

Anchovy

Anvil

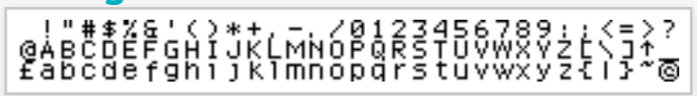
Area51



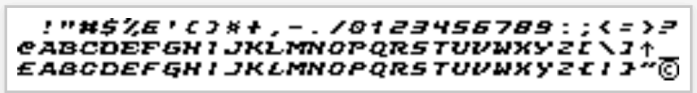
Around

Homestead

Hourglass

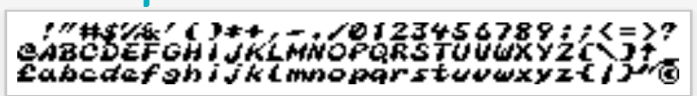


Ident

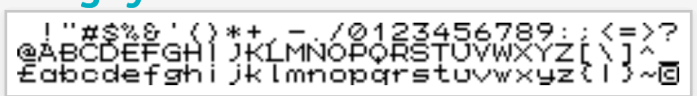


Ink Stamp

Inkscript



Insigbyte



METAFONT

- Released in 1977 by Donald Knuth.
- METAFONT was made as a compagnion to TeX.
- Could generate bitmaps at arbitary scale from METAFONT code.
- Made to support more fonts on printers, else only pre-programmed fonts were availible.
- Has support for ligatures, by substitution.

PostScript Type 1

- Released in 1984 by Adobe.
- Has support for hinting which makes them better on screens.
- Used in the first Macintosh.
- OpenType is a direct decendent.

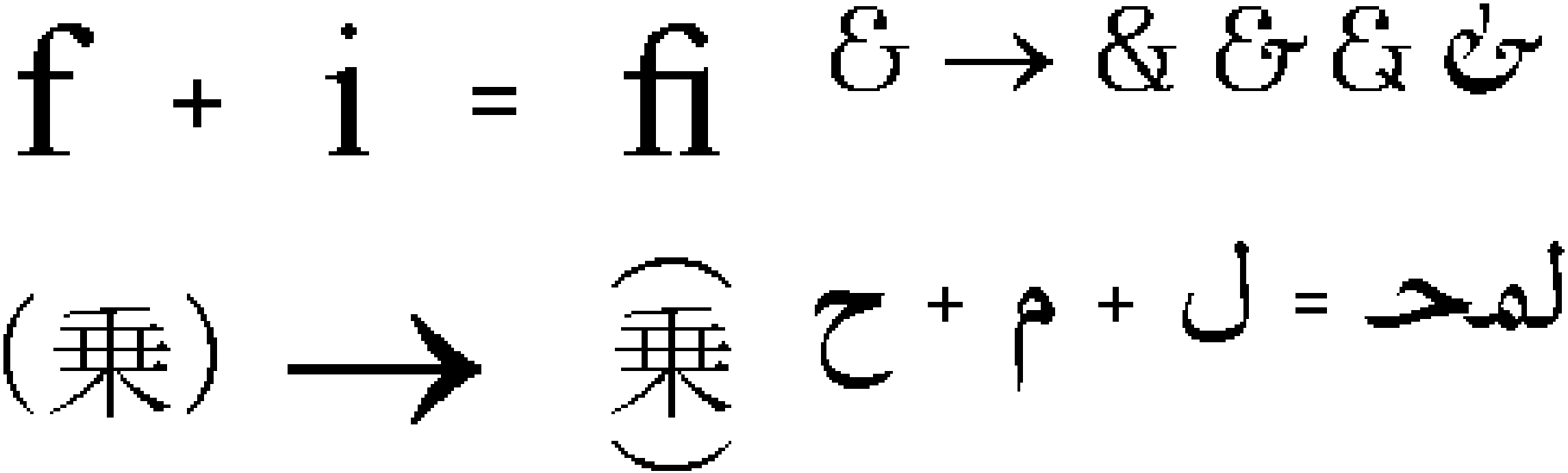
HarfBuzz - Shaper

Glyphs in HarfBuzz means the symbol that will be rendered. In fonts this is often not the same as the Unicode codepoint. For example ‘A’ may be 0x0 and not 0x41.

- Takes a stream of characters and places “Glyphs”.
- Understands a series of tables to do substitution,
- placement and kerning.

Shaping - GSUB

- Changes a list of characters to a different glyph.
- Can also be used to exchange a single character for alternatives.



Shaping - GPOS

- The GPOS table tells where to position a specific glyph.
- Also says how to position two glyphs against each other (Kerning).
- Can also be used to position ^, ¨, ´, `, ~, and similar.

Wörter Wörter

Shaping - WebAssembly?

```
extern "C" {
    fn buffer_set_contents(
        buffer: u32, cbuffer: &CBufferContents
    ) -> bool;
}

#[repr(C)]
struct CBufferContents {
    length: u32,
    info: *mut CGlyphInfo,
    position: *mut CGlyphPosition,
}
```

Shaping - WebAssembly?

```rust
/// Glyph information in a buffer
/// item provided by Harfbuzz
#[repr(C)]
pub struct CGlyphInfo {
    pub codepoint: u32,
    pub mask: u32,
    pub cluster: u32,
    pub var1: u32,
    pub var2: u32,
}
```

```rust
/// Glyph positioning information in a
/// buffer item provided by Harfbuzz
#[repr(C)]
pub struct CGlyphPosition {
    pub x_advance: i32,
    pub y_advance: i32,
    pub x_offset: i32,
    pub y_offset: i32,
    pub var: u32,
}
```

Shaping - WebAssembly?

```rust
/// Ergonomic representation of a Harfbuzz buffer item
pub struct Glyph {
    pub codepoint: u32, /// The Unicode codepoint or glyph ID of the item
    pub cluster: u32, /// The index of the cluster in the input text
                      /// where this came from
    pub x_advance: i32, /// The horizontal advance of the glyph
    pub y_advance: i32, /// The vertical advance of the glyph
    pub x_offset: i32, /// The horizontal offset of the glyph
    pub y_offset: i32, /// The vertical offset of the glyph
    pub flags: u32, /// You can use this for whatever you like
}
```

Animations* With Fonts

Bad Apple!!

- Bad apple is a peice of music from the Touhou series of games.
- A shadow play have been popular in the demo scene.

Bad Apple!! - Demo

- Bad apple is a peice of music from the Touhou series of games.
- A shadow play have been popular in the demo scene.
- Demo!

Bad Apple!!

- Bad apple is a peice of music from the Touhou series of games.
- A shadow play have been popular in the demo scene.
- Demo!
- There was some complaints about not being clear about it.
- If you want to read more about how it was made look at https://blog.erk.dev.
- This is perfectly possible with GSUB.

Computer games in a font

What is possible?

What is possible?

- Fully deterministic.

What is possible?

- Fully deterministic.
- It has to be fast to run.

What is possible?

- Fully deterministic.
- It has to be fast to run.
- Relativly simple.

What is possible?

- Fully deterministic.
- It has to be fast to run.
- Relativly simple.
- Should work with a single colour.

What is possible?

- Fully deterministic.
- It has to be fast to run.
- Relativly simple.
- Should work with a single colour.
- Able to run in steps

Tetris!

Tetris!

```
/// Deterministic Tetris that can be rendered at any time.
pub struct Tetris {
    rand: u64,
    // Tetris is 10 wide and 20 high, use a bitmap to draw from, upper
    // 6 bits are not used
    gameboard: [u16; HEIGHT],
    // 0..4 clockwise
    rot: u8,
    inflight_t: Tetromino,
    inflight_pos: (u8, u8),
}
```

Tetris! - Deterministic

```
fn next_rand(&mut self) -> u64 {
    let mut x = self.rand;
    x ^= x << 13;
    x ^= x >> 7;
    x ^= x << 17;
    self.rand = x;
    x
}
```

Tetris! - TetrisDisplay

```rust
pub trait TetrisDisplay {
    fn display(&mut self, gameboard: [u16;20]);
}
```

Tetris! - TetrisDisplay

```rust
pub trait TetrisDisplay {
    fn display(&mut self, gameboard: [u16;20]);
}

pub struct DebugDisplay;

impl TetrisDisplay for DebugDisplay {
    fn display(&mut self, gameboard: [u16;20]) {
        for line in gameboard {
            println!("{line:010b}");
        }
    }
}
```

Tetris! - block_at

```rust
pub(crate) fn block_at(gameboard: [u16; 20], pos: (u8, u8)) -> bool {
    let (x, y) = pos;
    if (x as usize) >= WIDTH || (y as usize) >= HEIGHT {
        return true;
    }

    let line = gameboard[y as usize];
    let mask = 1 << (WIDTH as u8 - (x + 1));

    (line & mask) != 0
}
```

Tetris! - pixel_at

```
pub(crate) fn pixel_at(pos: (u8, u8)) -> Glyph {
    let (posx, posy) = pos;
    let x = PIXEL_WIDTH * (posx as usize) as i32;
    let y = (-PIXEL_HEIGHT * (posy as usize) as i32) + 800;

    Glyph {
        codepoint: 0x1337,
        cluster: (posx as u32) << 8 | posy as u32,
        x_advance: 0,
        y_advance: 0,
        x_offset: x,
        y_offset: y,
        flags: 0,
    }\n}
```

Tetris! - SavingDisplay

```
impl TetrisDisplay for SavingDisplay {
    fn display(&mut self, gameboard: [u16; 20]) {
        let mut new_glyphs = Vec::new();
        for h in 0..HEIGHT {
            for w in 0..WIDTH {
                if block_at(gameboard, (w as u8, h as u8)) {
                    debug(&format!("Block at: ({h}; {w})"));
                    new_glyphs.push(pixel_at((w as u8, h as u8)));
                }
            }
        }
        self.new_glyphs = new_glyphs;
    }
}
```

Tetris! - Demo

Demo time

What's next?

- Figure out if a GameBoy emulator would be possible.
- Looks at simple invented platforms which might be possible.

What's next?

- Figure out if a GameBoy emulator would be possible.
- Looks at simple invented platforms which might be possible.
- Use it for some cool actual useful things.
- https://github.com/harfbuzz/harfbuzz-wasm-examples.

Questions?

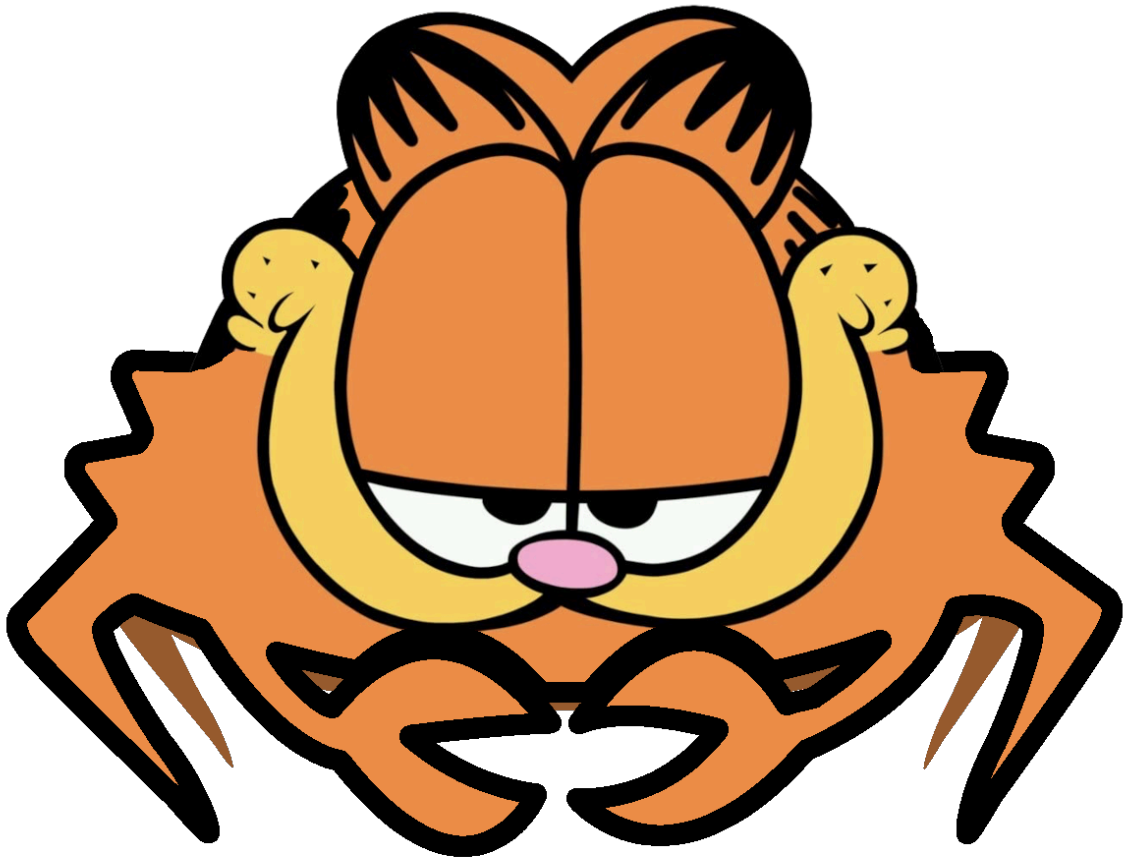


Image Credits

- Sales contract Shuruppak Louvre AO3766: Unknown artist. License: Public domain.
- Derby, bone plate: Trustees of the British Museum unbekannt/unknown. License: CC BY-NC-ND.
- DANIEL CHODOWIECKI 62 bisher unveröffentlichte Handzeichnungen zu dem Elementarwerk von Johann Bernhard Basedow. Mit einem Vorworte von Max von Boehn. Voigtländer-Tetzner, Frankfurt am Main 1922. License: Public domain.
- Gutenberg bible: The Royal Danish Library.
- Gutenberg page: National Library of Scotland, License: CC BY 4.0.
- Kerning. License: Public domain.
- Kerning TT. License: Public domain.
- f: Typographical Printing-Surfaces: The Technology and Mechanism of their Production by Lucien Alphonse Legros and John Cameron Grant. (1916)
- Composing stick: By Wilhei - Own work, CC BY 3.0, https://commons.wikimedia.org/w/index.php?curid=7698365
- Garamond type ﬁ-ligature 2, Garamond type ﬁ-ligature 2. GFDL and CC-by-sa-2.0-de, https://commons.wikimedia.org/wiki/File:Garamond_type_%C5%BFi-ligature_2.jpg
- Garamond type ft-ligature, Garamond type ﬁ-ligature 2. GFDL and CC-by-sa-2.0-de, https://commons.wikimedia.org/wiki/File:Garamond_type_ft-ligature.jpg
- Bitmap fonts: https://damieng.com/typography/zx-origins/
- HarfBuzz logo: https://harfbuzz.github.io/
- GSUB images: https://learn.microsoft.com/en-us/typography/opentype/spec/gsub
- GPOS images: https://learn.microsoft.com/en-us/typography/opentype/spec/gpos
- Tetris logo: https://tetris.com/brand-assets
- Garris: GitHub@Noxime