August 29, 2018

The Art of the *Font*

by Ayal Spitz

Accenture Interactive

Agenda

01 Introduction

02 Motivation & History

03 Fonts & Apple

04 CoreText

05 TextKit

06 Dynamic Type

07 Closing Thoughts

Introduction

01

Ayal Spitz

Senior Principal Software Engineer

Accenture Interactive

@aspitz

ayal.spitz@accenture.com

Motivation & History

02

Code → Compile → Run

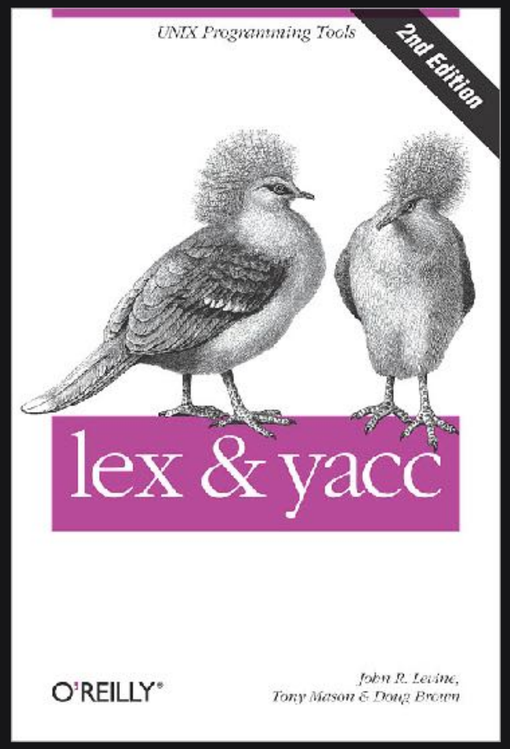
Code → Compile → Run

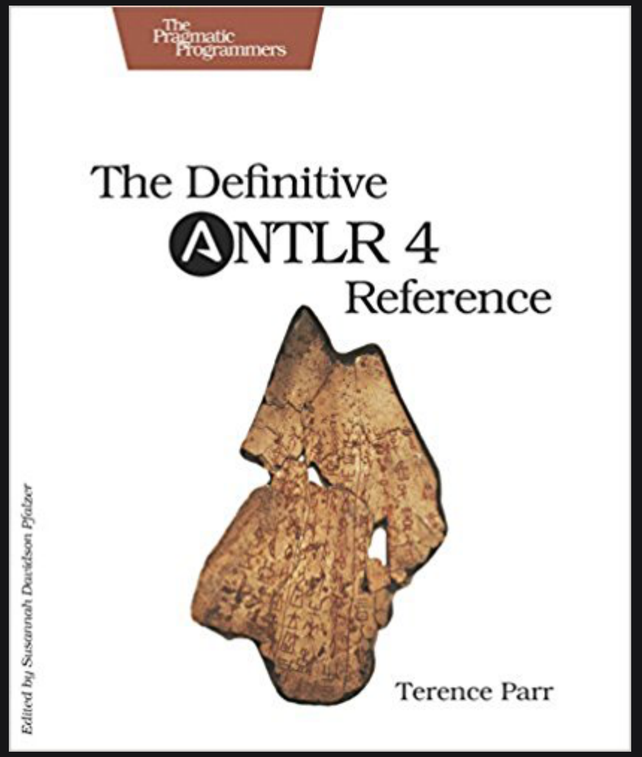
Motivation & History

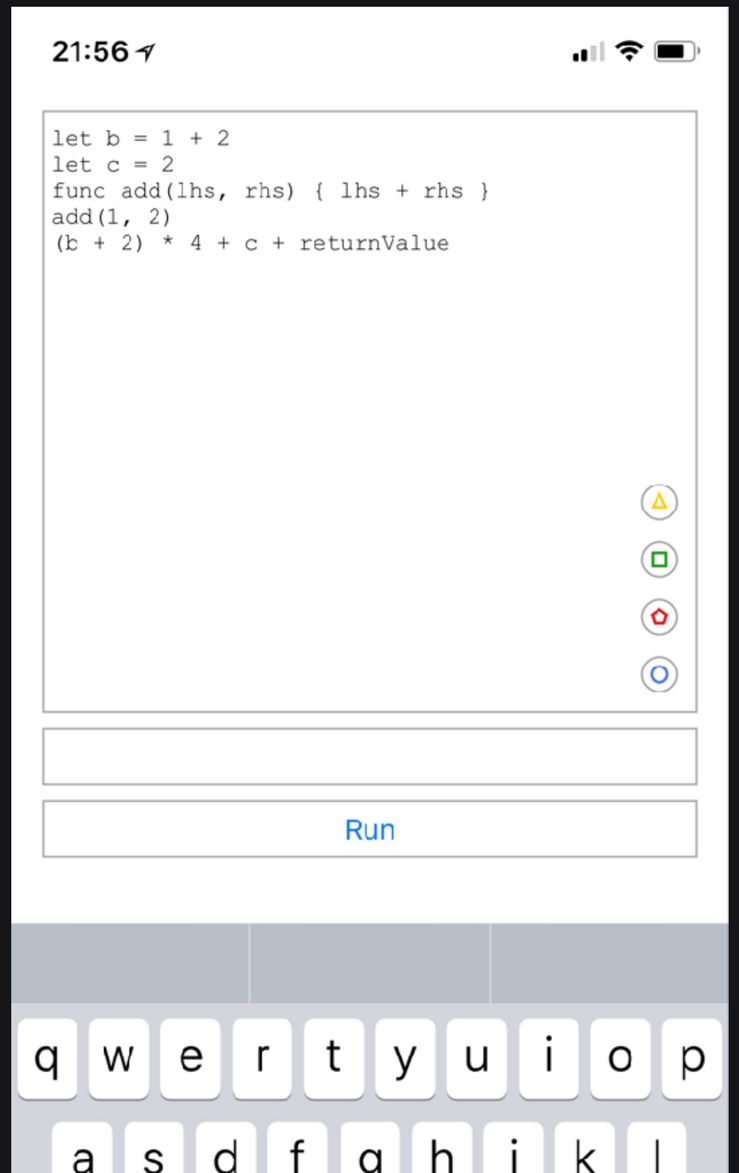
The Art of the Font by Ayal Spitz

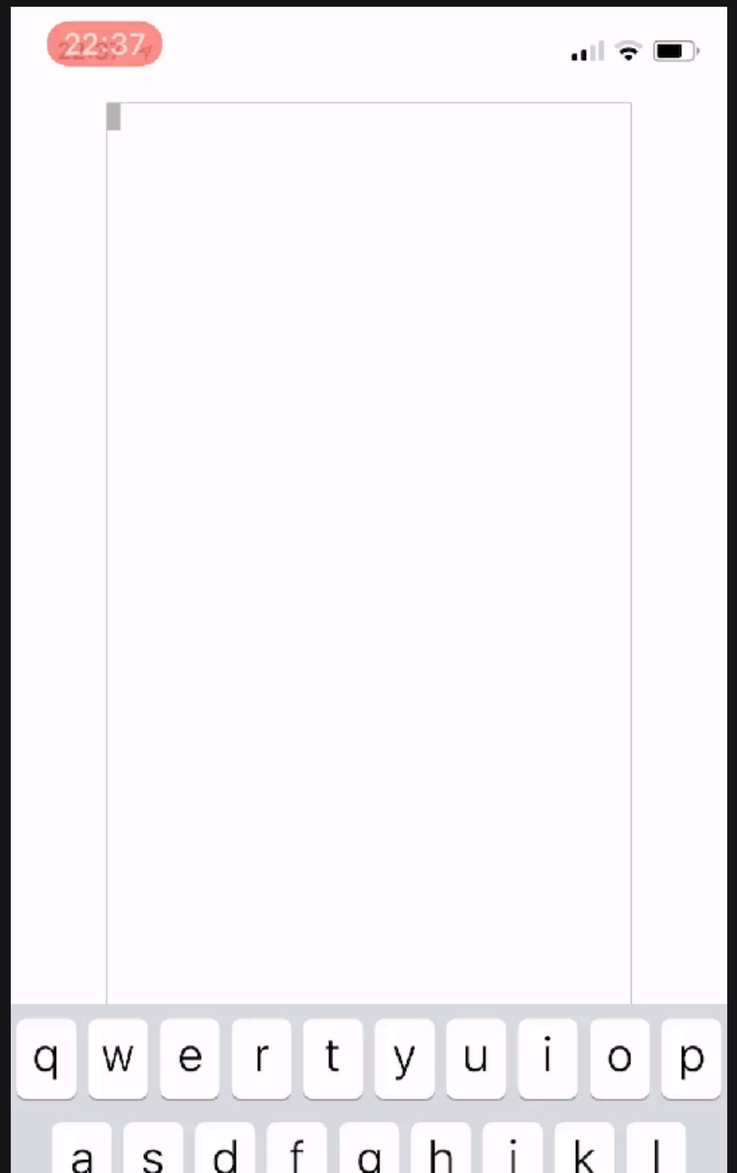
Motivation & History

The Art of the Font by Ayal Spitz






21:56
let b = 1 + 2
let c = 2
func add(lhs, rhs) { lhs + rhs }
add(1, 2)
(b + 2) * 4 + c + returnValue
Run
q w e r t y u i o p
a s d f g h j k l


22:37
q w e r t y u i o p
a s d f g h j k l

Calligraphy

Reed College

Calligraphy

[ke' ligrafe] ***noun***

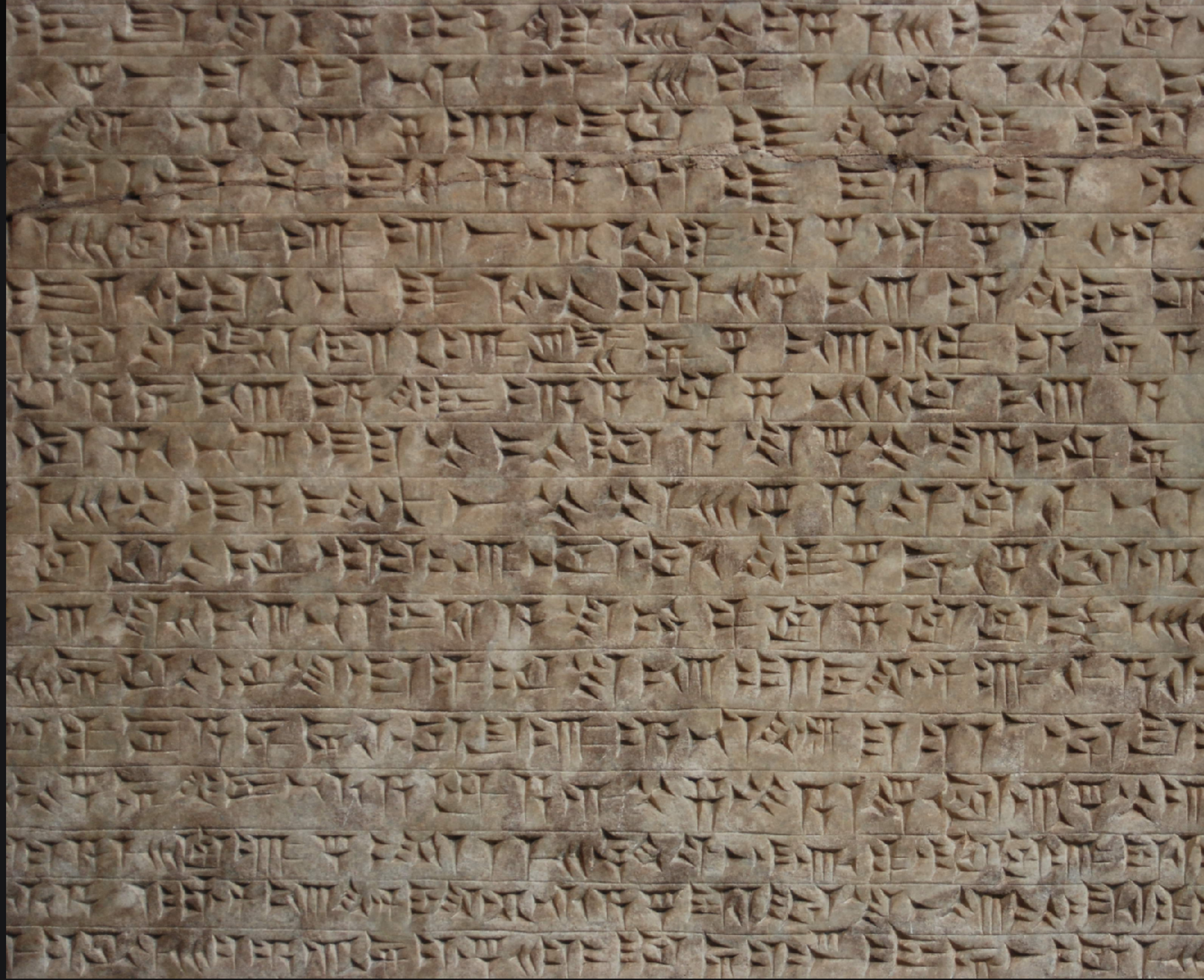
decorative handwriting or handwritten lettering

origin & etymology

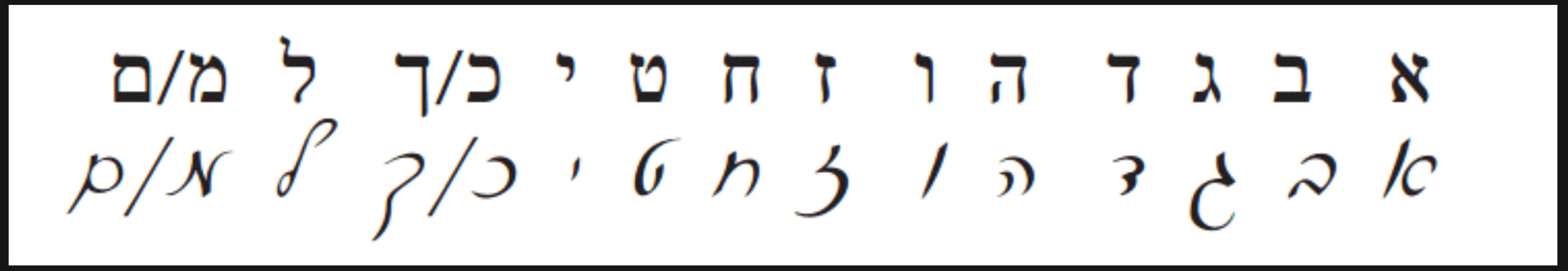
greek

kallos - “beautiful”
graphein - “to write”

Written Language



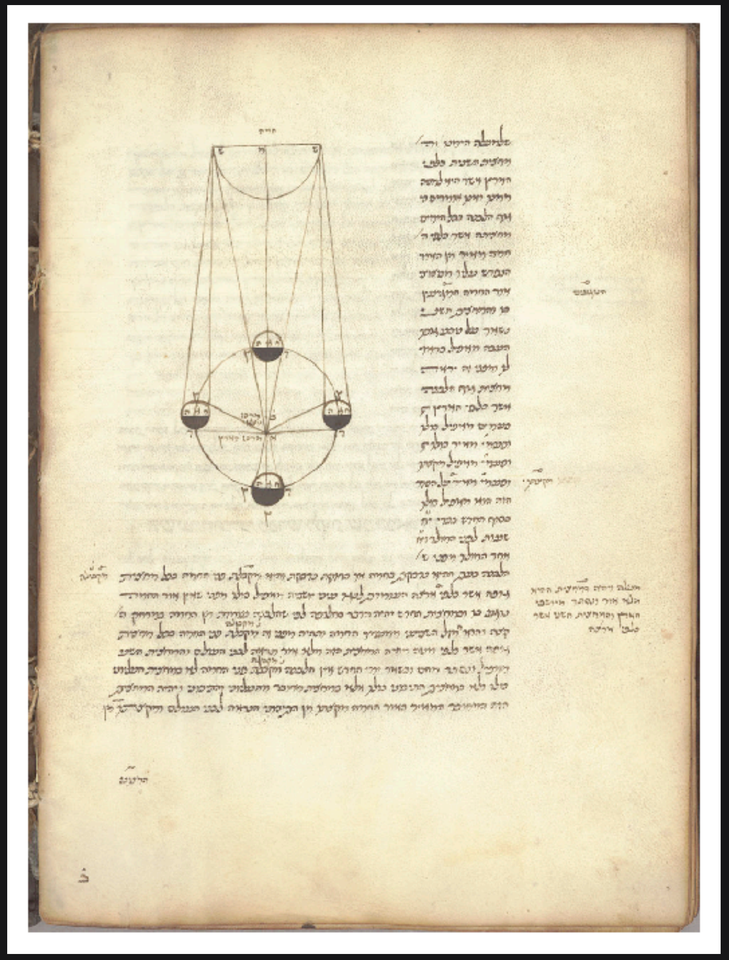
Hebrew



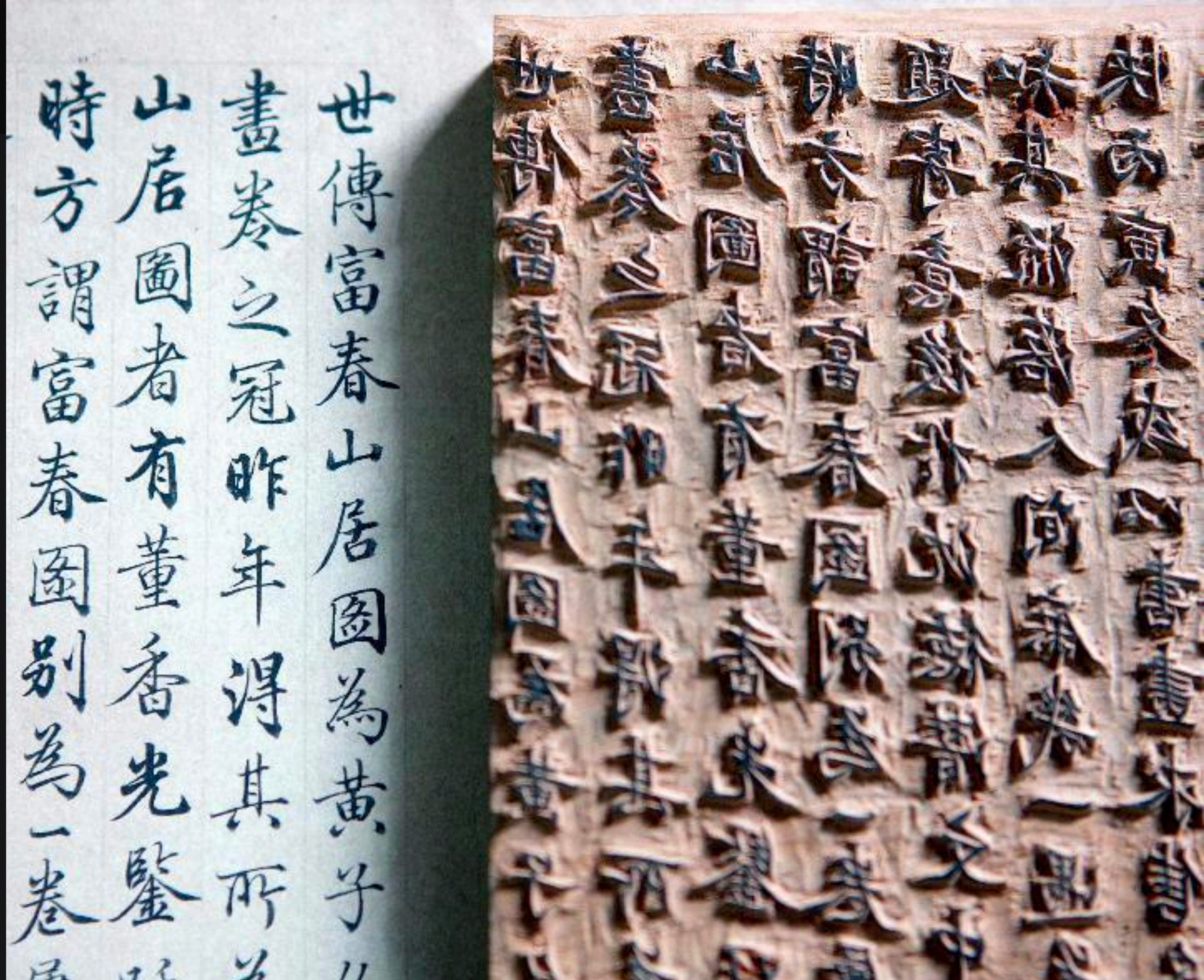
Arabic

ز	ر	ذ	د	خ	ح	ج	ث	ت	ب	ا
zaay	raa'	dhaal	daal	khaa'	Haa'	jiim	thaa'	taa'	baa'	'alif

Books

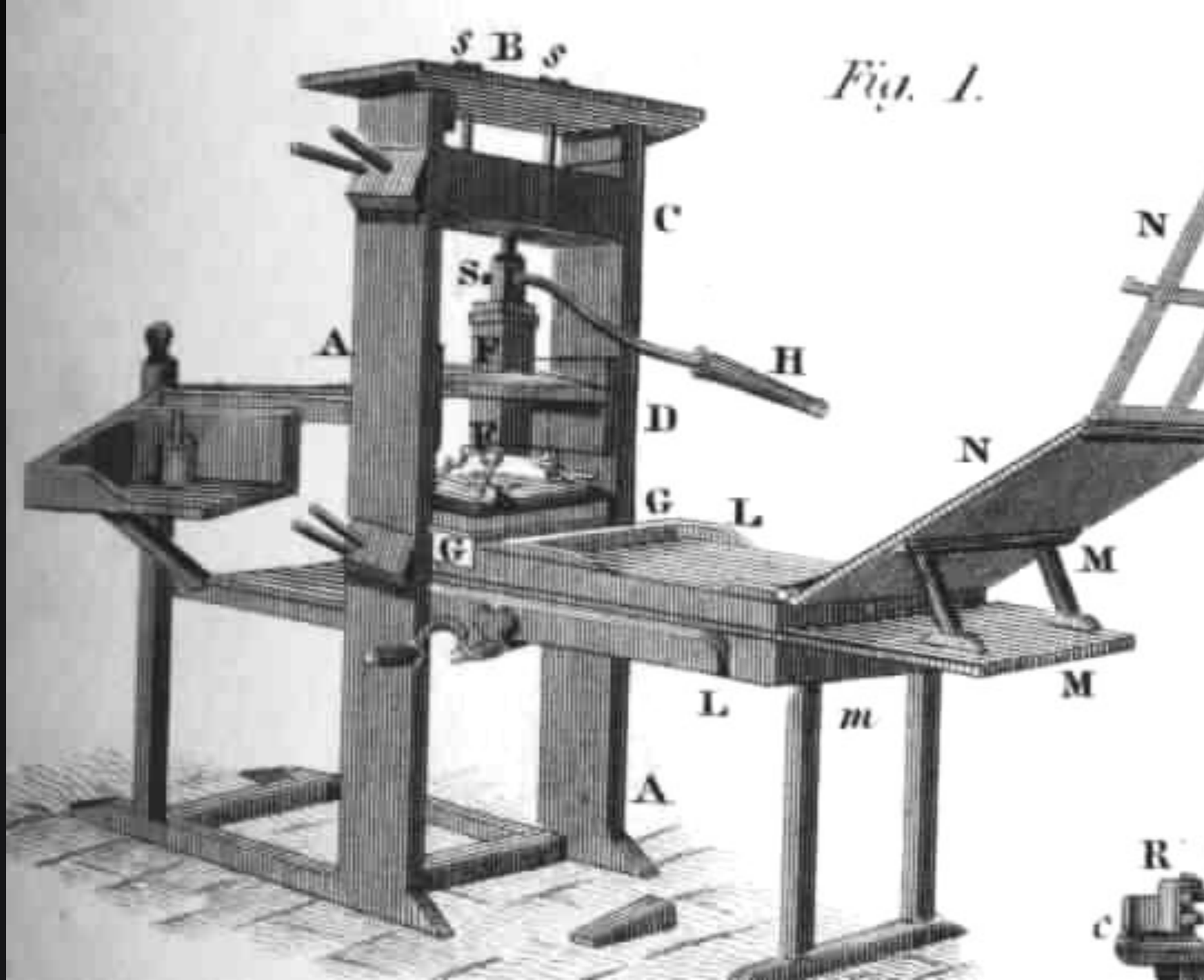


Woodblock Printing

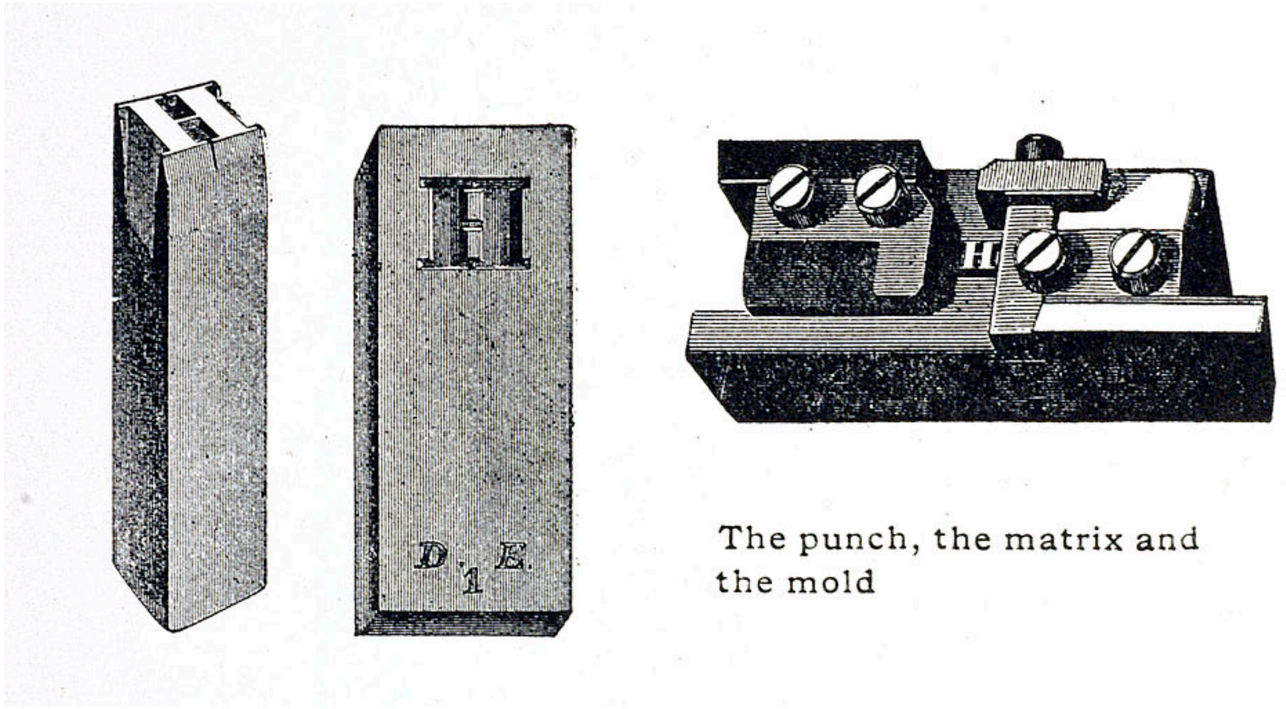


Movable Type

Gutenberg Press

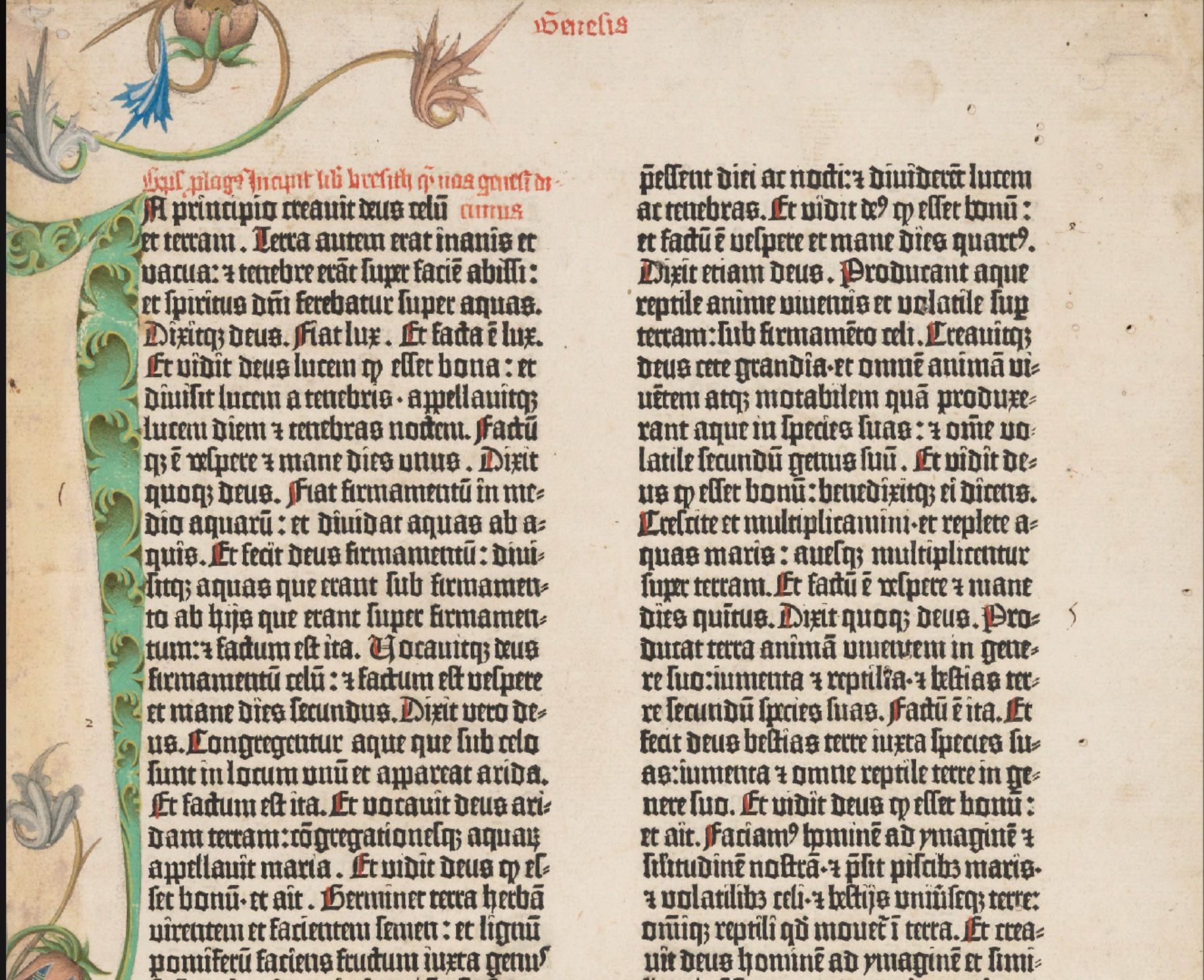


Gutenberg Press



The punch, the matrix and the mold

Gutenberg Bible



Fonts & Apple

03

Apple II

DOS


APPLE II
DOS VERSION 3.3 SYSTEM MASTER
JANUARY 1, 1983
COPYRIGHT APPLE COMPUTER,INC. 1980,1982
]

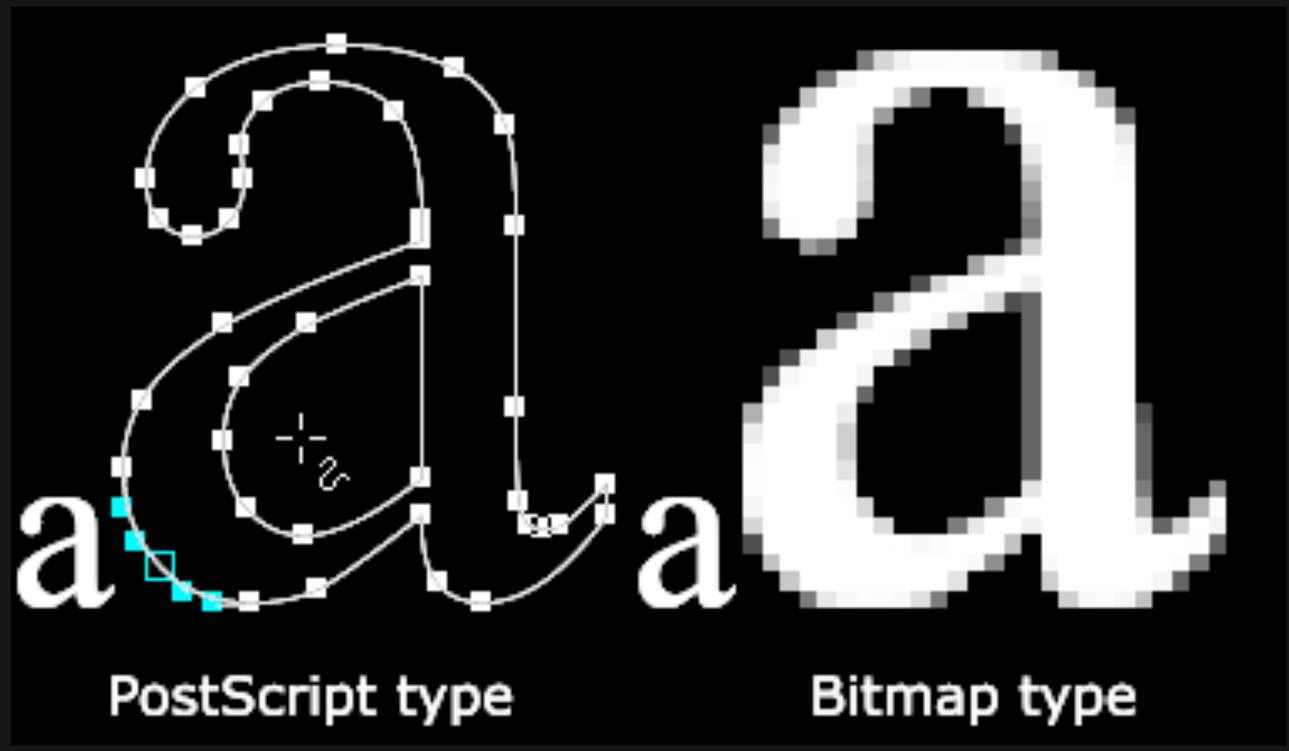
Mac Fonts

Mac OS 1 - 9

Adobe PostScript

Outline Font

Type 1

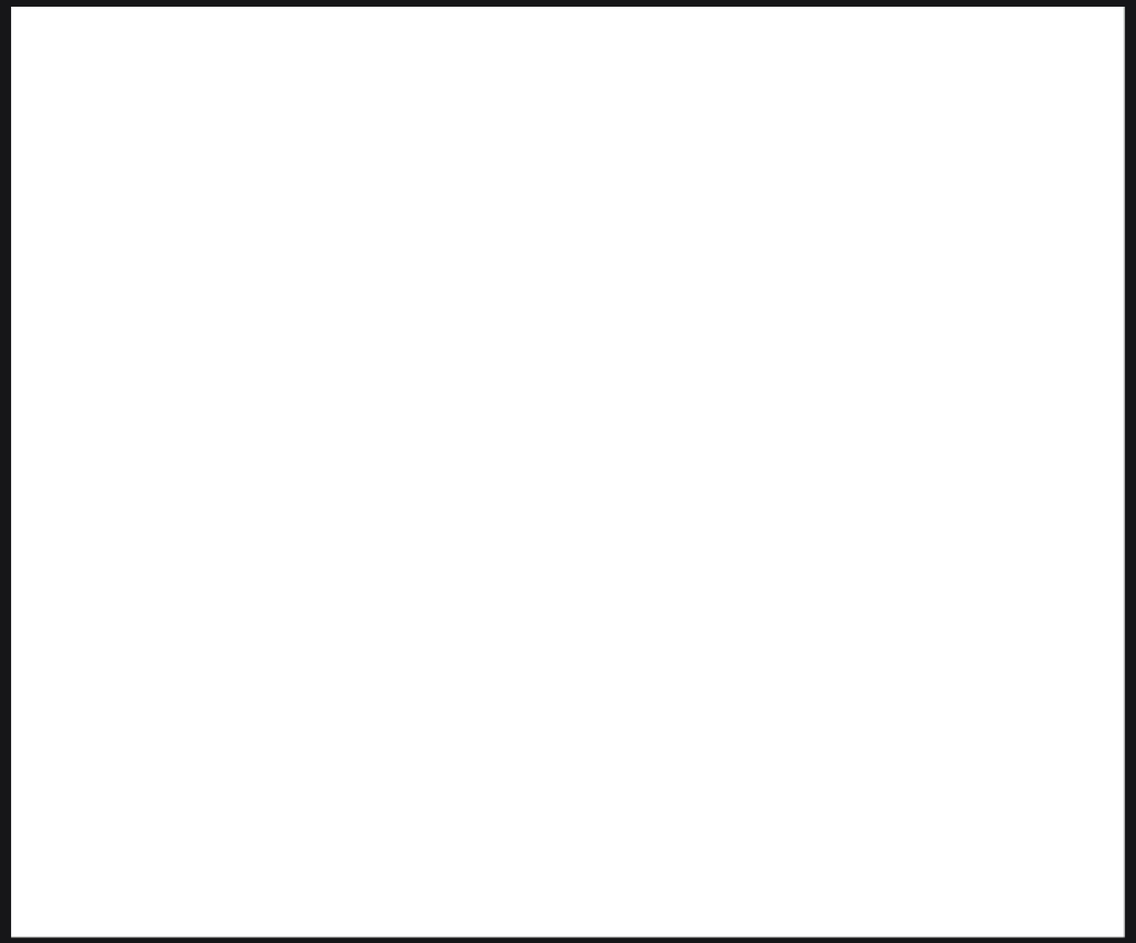


TrueType

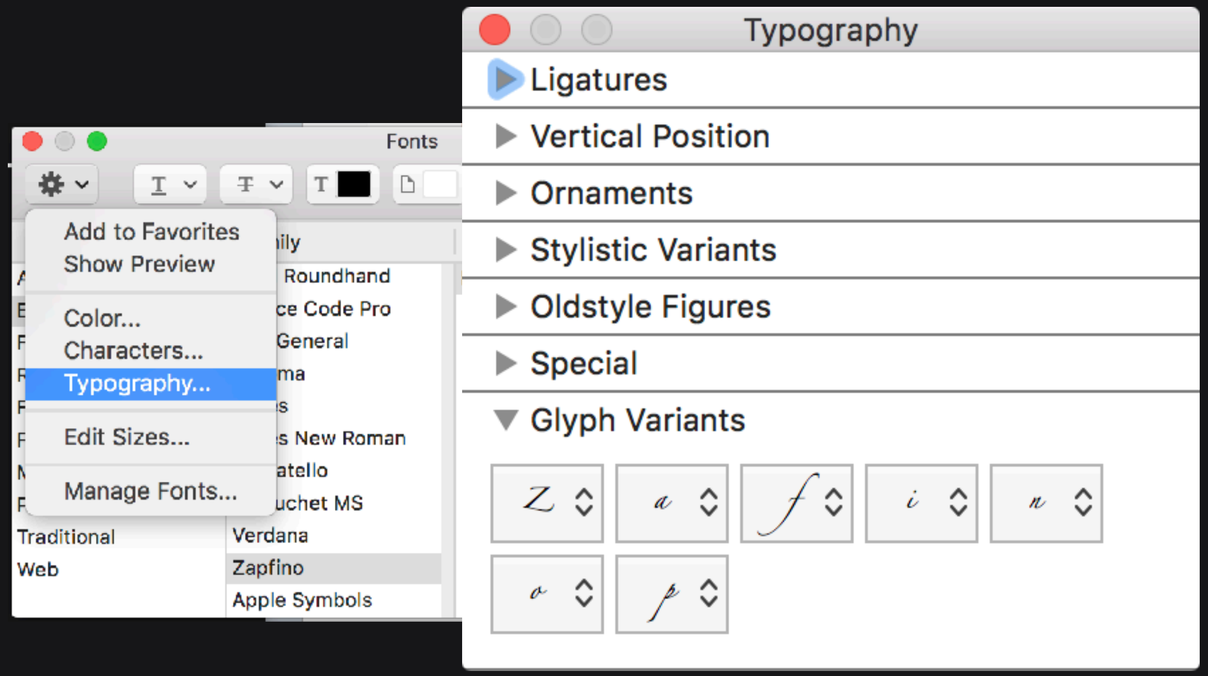
OpenType

Apple Advanced Typography

Apple Advanced Typography



Apple Advanced Typography

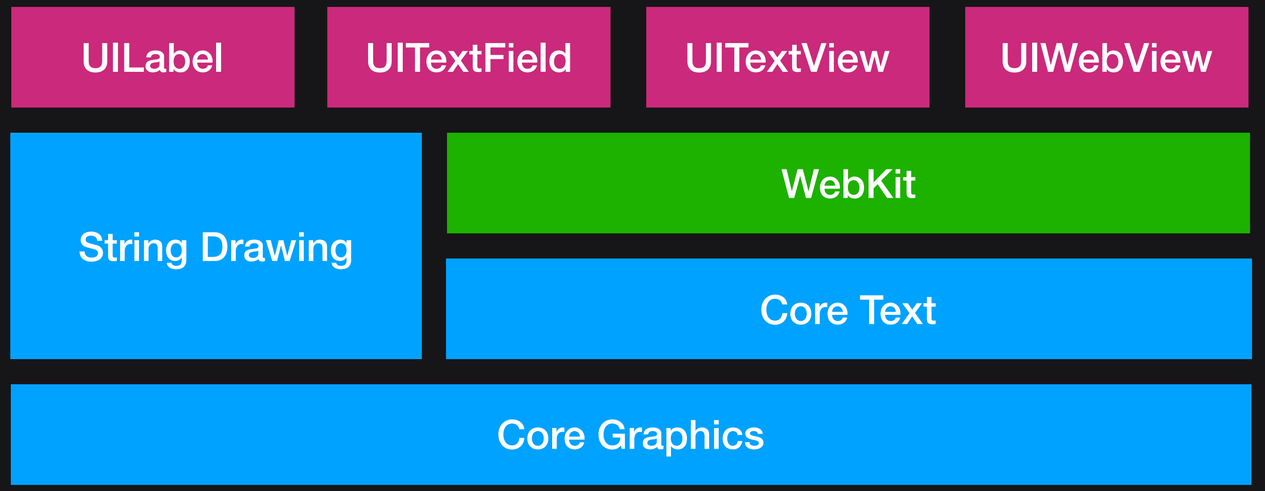


CoreText

04

CoreText

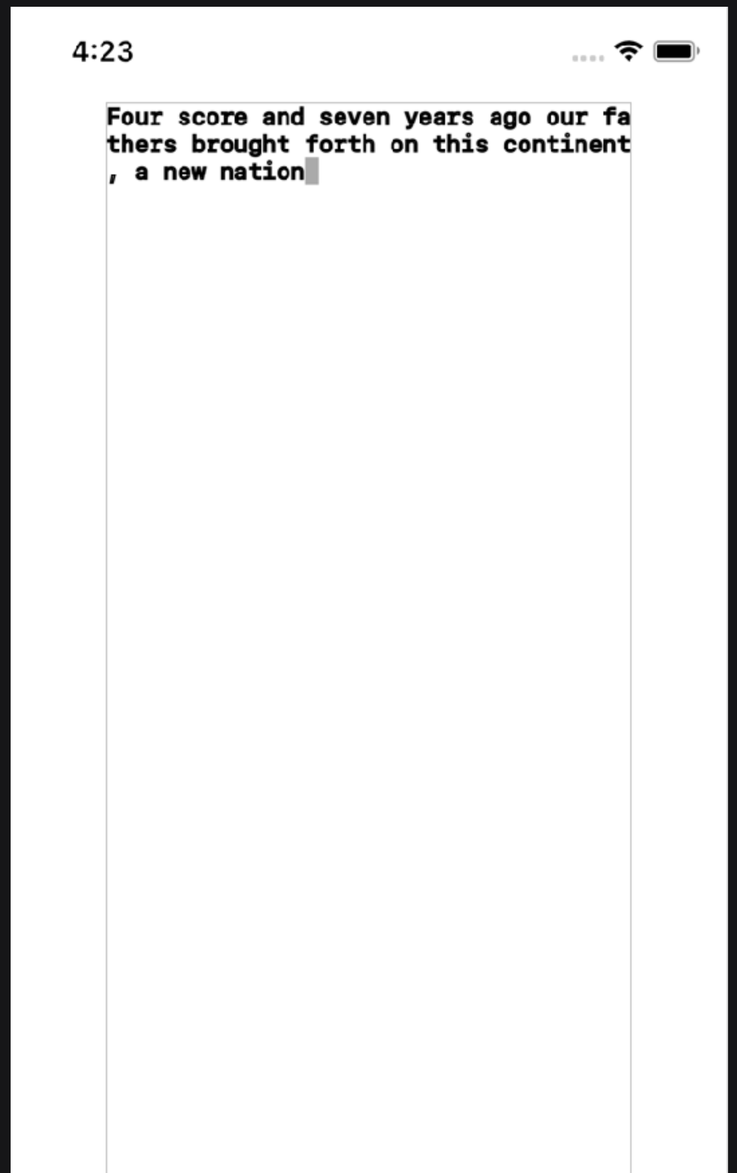
per iOS 7

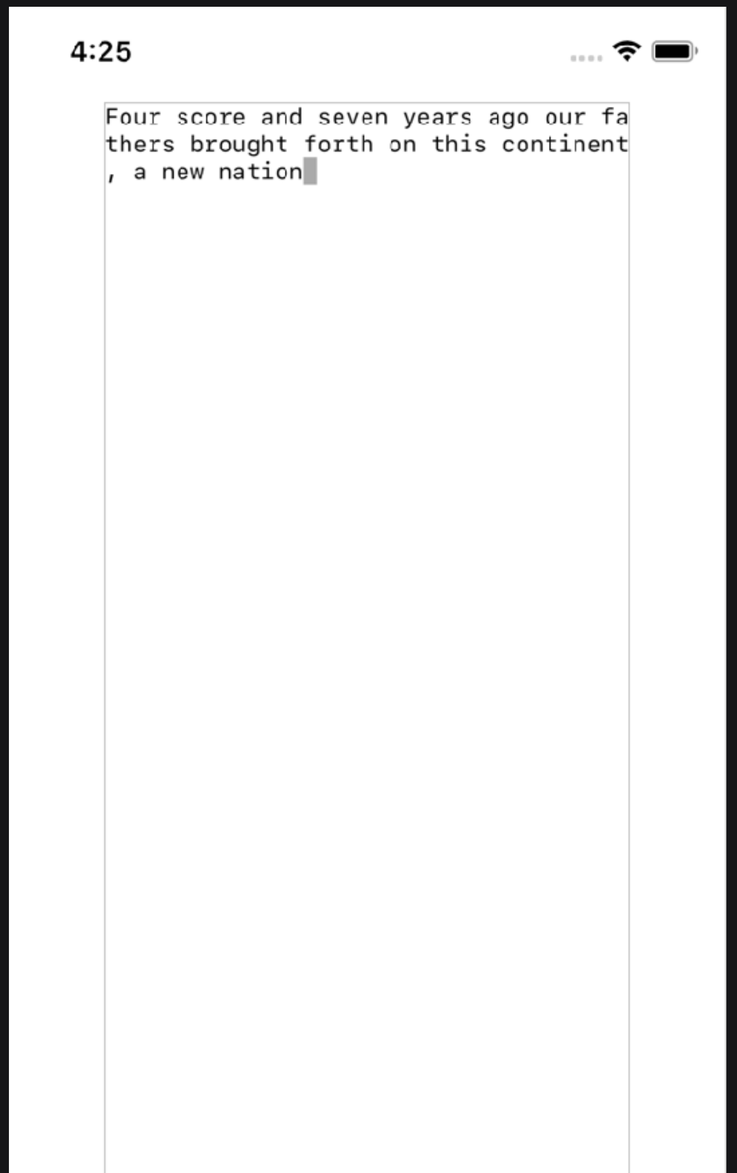


```swift
extension Character {
    public func glyph(font: CTFont) -> CGGlyph? {
        let c = unicodeScalars[unicodeScalars.startIndex].utf16
        var chars = Array(c)
        var glyphs = Array<CGGlyph>(repeating: 0, count: 1)
        CTFontGetGlyphsForCharacters(font, &chars, &glyphs, 1)

        return glyphs.count > 0 ? glyphs[0] : nil
    }
}
```

```swift
extension CTFont {
    func cgPath(for char: Character, affineTransform: CGAffineTransform? = nil)
    -> CGPath? {
        guard let glyph = char.glyph(font: self) else { return nil }
        if affineTransform != nil {
            var localAffineTransform = affineTransform!
            return CTFontCreatePathForGlyph(self, glyph, &localAffineTransform)
        } else {
            return CTFontCreatePathForGlyph(self, glyph, nil)
        }
    }
}
```

```swift
extension UIFont {
    public func cgPath(for char: Character, affineTransform: CGAffineTransform? = nil)
    -> CGPath? {
        return (self as CTFont).cgPath(for: char, affineTransform: affineTransform)
    }

    public func uiBezierPath(for char: Character) -> UIBezierPath? {
        let flipVertical = CGAffineTransform(a: 1.0, b: 0.0, c: 0.0, d: -1.0, tx: 0.0,
        ty: lineHeight)
        guard let path = cgPath(for: char, affineTransform: flipVertical) else { return nil }
        return UIBezierPath(cgPath: path)
    }

    public func image(for char: Character, with color: UIColor = .black,
        at scale: CGFloat = UIScreen.main.scale) -> UIImage? {
        guard let bezierPath = uiBezierPath(for: char) else { return nil }
        let letterSize = "\(char)".size(withAttributes: [NSAttributedStringKey.font: self])

        UIGraphicsBeginImageContextWithOptions(letterSize, false, UIScreen.main.scale)
        color.setFill()
        bezierPath.apply(CGAffineTransform(translationX: 0, y: descender))
        bezierPath.fill()
        let image = UIGraphicsGetImageFromCurrentImageContext()
        UIGraphicsEndImageContext()
        return image
    }
}
```


4:23
Four score and seven years ago our fa
thers brought forth on this continent
, a new nation

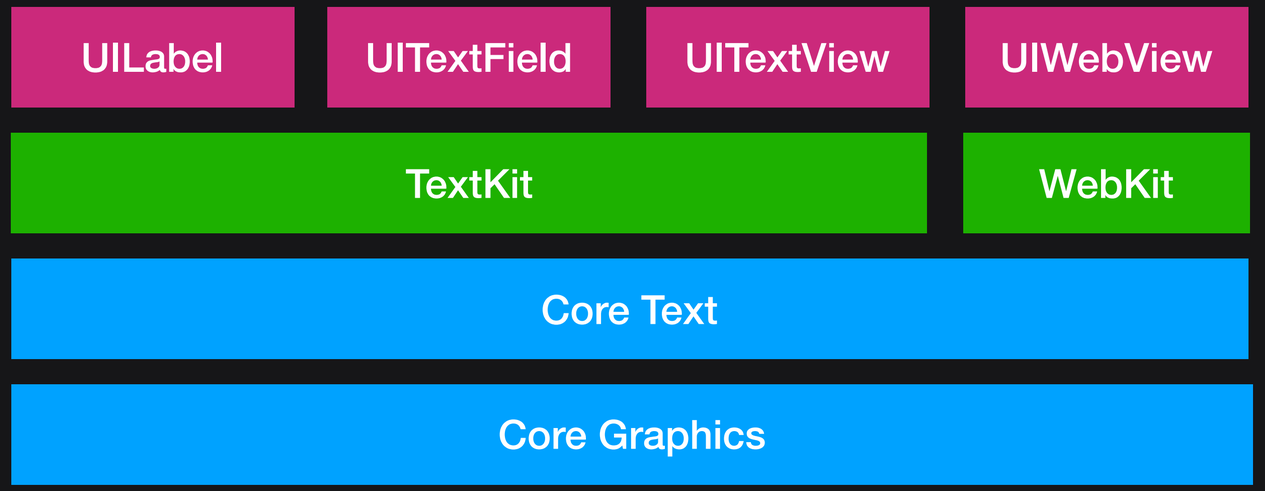

4:25
Four score and seven years ago our fa
thers brought forth on this continent
, a new nation

TextKit

05

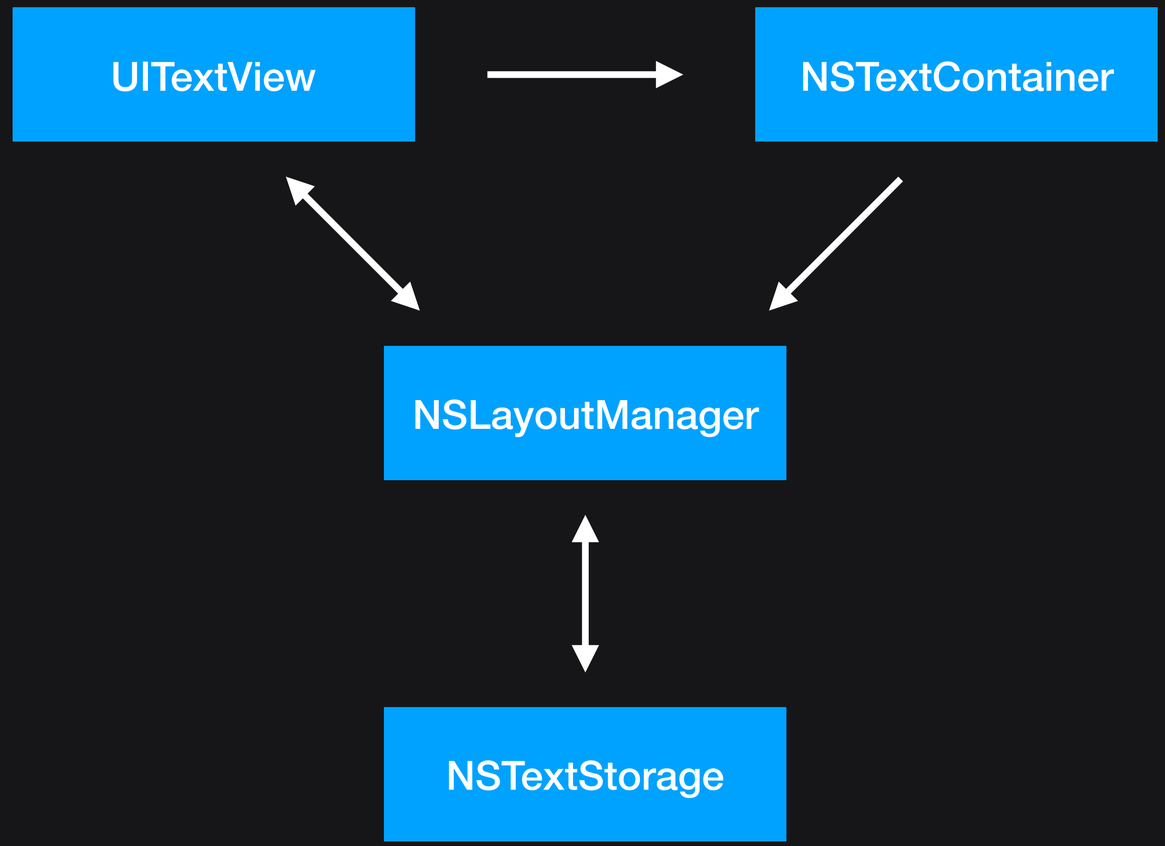
TextKit

iOS 7



TextKit

iOS 7



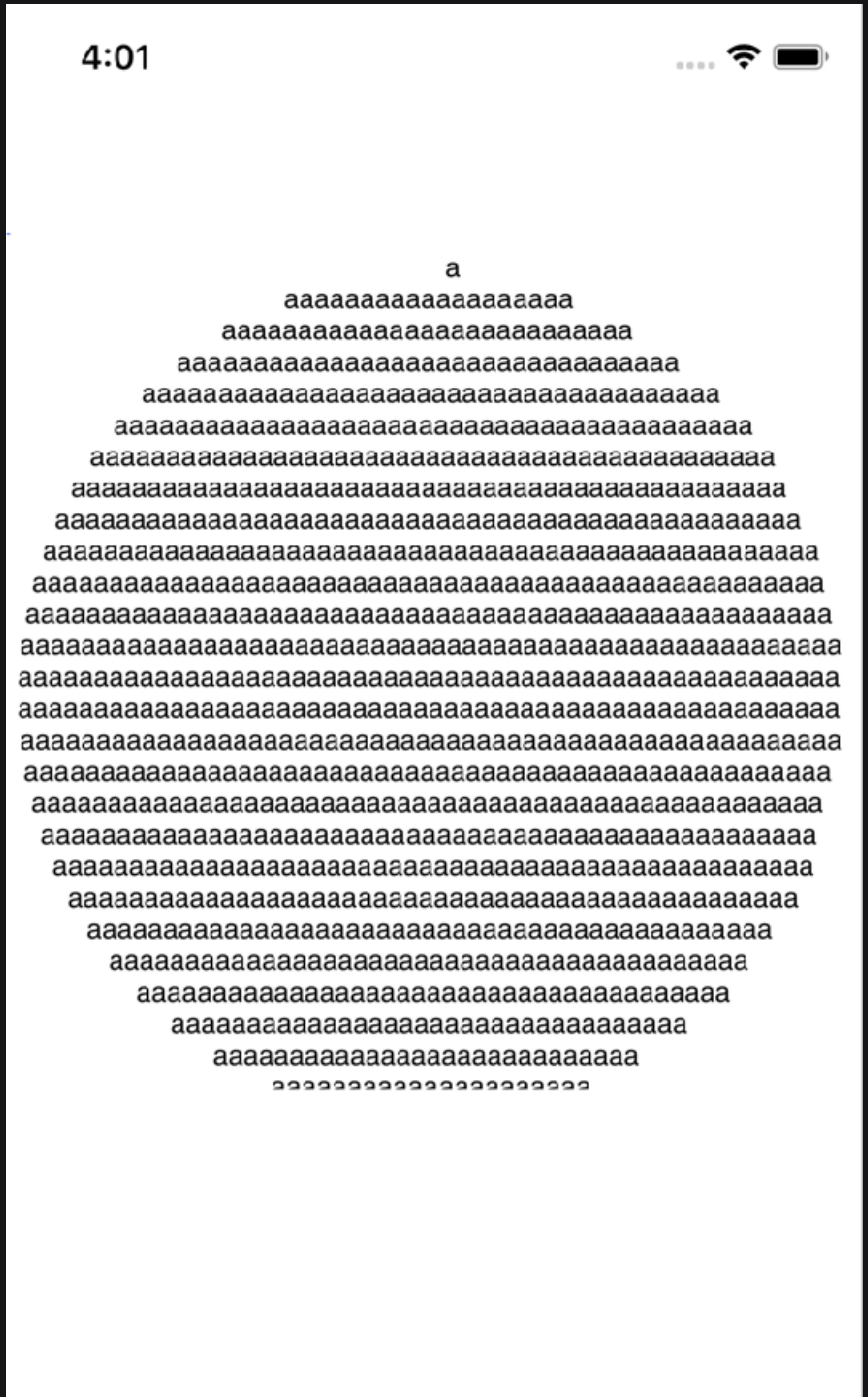
```
import UIKit

class CustomTextStorage: NSTextStorage {
    let backingStore = NSMutableAttributedString()

    override var string: String { return backingStore.string }

    override func attributes(at location: Int, effectiveRange range: NSRangePointer?)
    -> [NSAttributedStringKey : Any] {
        return backingStore.attributes(at: location, effectiveRange: range)
    }

    override func replaceCharacters(in range: NSRange, with str: String) {
        beginEditing()
        backingStore.replaceCharacters(in: range, with: str)
        edited([.editedAttributes, .editedCharacters], range: range,
        changeInLength: str.count - range.length)
        endEditing()
    }

    override func setAttributes(_ attrs: [NSAttributedStringKey : Any]?, range: NSRange) {
        beginEditing()
        backingStore.setAttributes(attrs, range: range)
        edited(.editedAttributes, range: range, changeInLength: 0)
        endEditing()
    }

    override func processEditing() {
        super.processEditing()
    }
}
```

```
import UIKit

class CustomTextContainer: NSTextContainer {
    override var isSimpleRectangularTextContainer: Bool { return false}

    override func lineFragmentRect(forProposedRect proposedRect: CGRect,
    at characterIndex: Int, writingDirection baseWritingDirection: NSWritingDirection,
    remaining remainingRect: UnsafeMutablePointer<CGRect>?) -> CGRect {

        var result = super.lineFragmentRect(forProposedRect:proposedRect,
        at:characterIndex, writingDirection:baseWritingDirection, remaining:remainingRect)

        let r = self.size.height / 2.0
        let y = r - result.origin.y
        let theta = asin(y/r)
        let x = r * cos(theta)
        let offset = self.size.width / 2.0 - r
        result.origin.x = r-x+offset
        result.size.width = 2*x


        return result

    }
}
```

```
import UIKit

class ViewController: UIViewController {
    var textView: UITextView!
    let textStorage = CustomTextStorage()

    override func viewDidLoad() {
        super.viewDidLoad()

        let textContainerSize = CGSize(width: view.bounds.width, height: view.bounds.width)
        let textContainer = CustomTextContainer(size: textContainerSize)
        textContainer.widthTracksTextView = true

        let layoutManager = NSLayoutManager()
        layoutManager.addTextContainer(textContainer)

        textStorage.addLayoutManager(layoutManager)

        let frame = CGRect(origin: CGPoint(x: 0.0, y: 100.0), size: textContainerSize)
        textView = UITextView(frame: frame, textContainer: textContainer)

        view.addSubview(textView)
    }
}
```

TextKit

The Art of the Font by Ayal Spitz

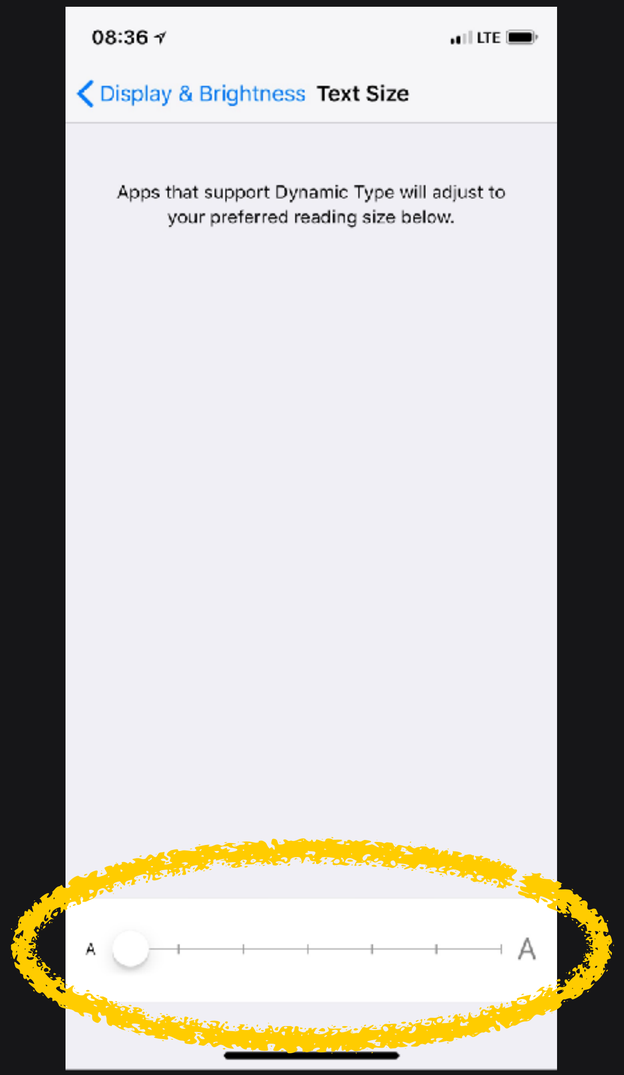


Dynamic Type

06

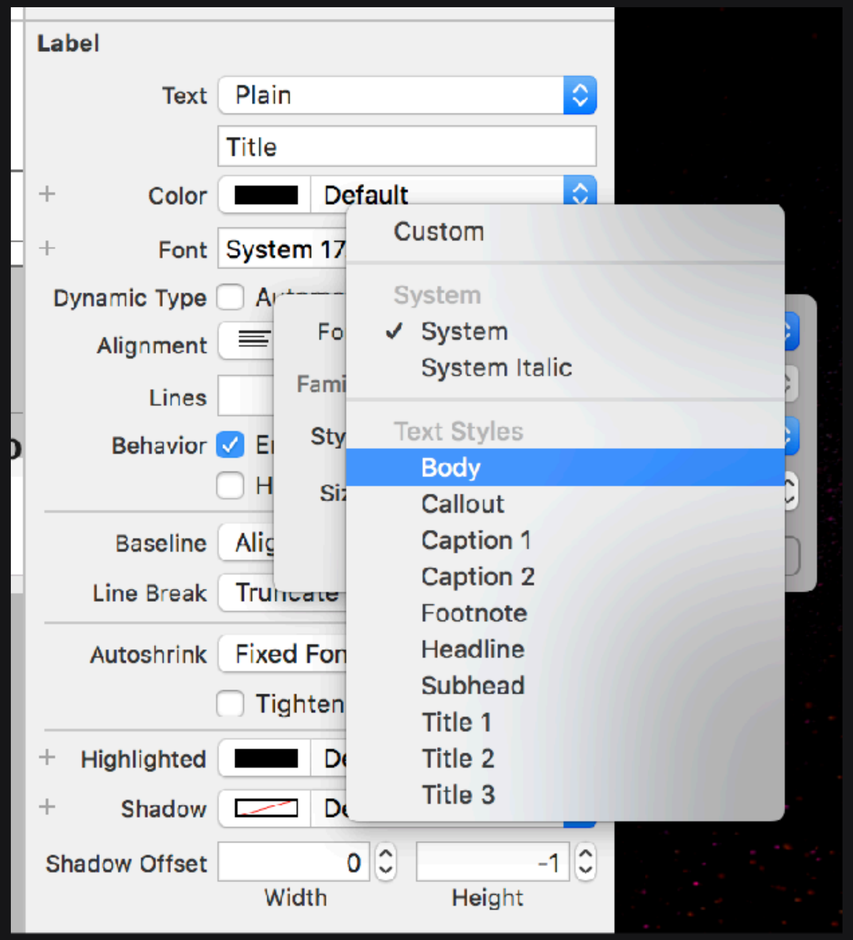
Dynamic Type

- Introduced in iOS 7
- Hidden away in **Settings** → **Display & Brightness** is the **Text Size** preference (as of iOS 11)



Dynamic Type

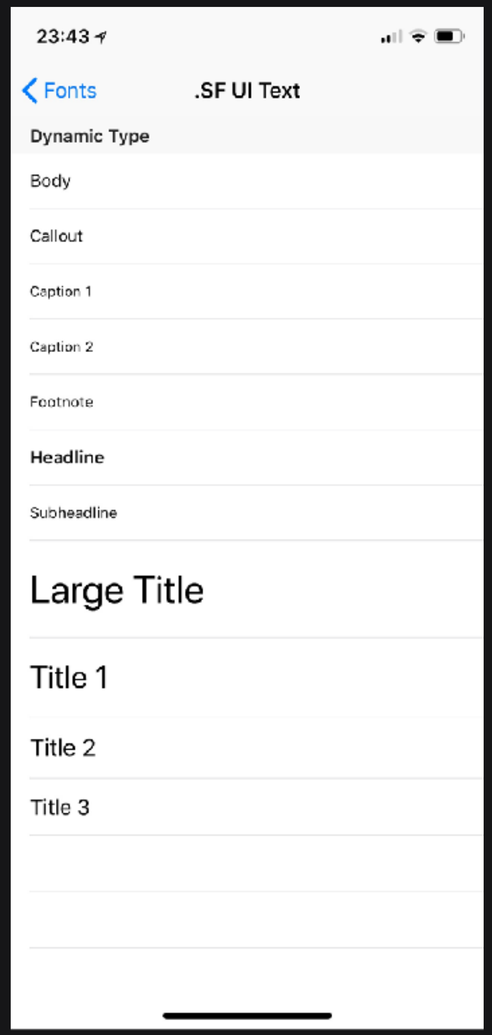
Interface Builder

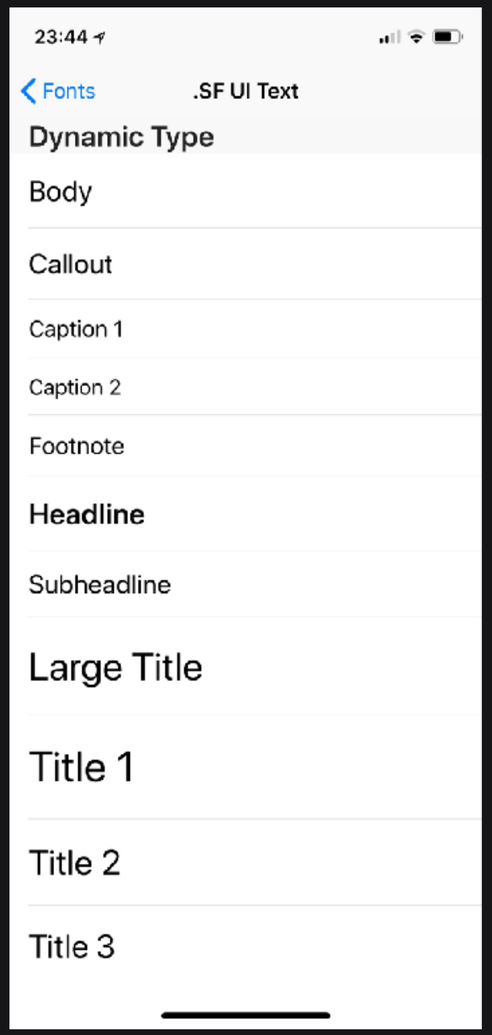


```
UIFont.preferredFont(forTextStyle: style)
```

Dynamic Type

Preferred Font



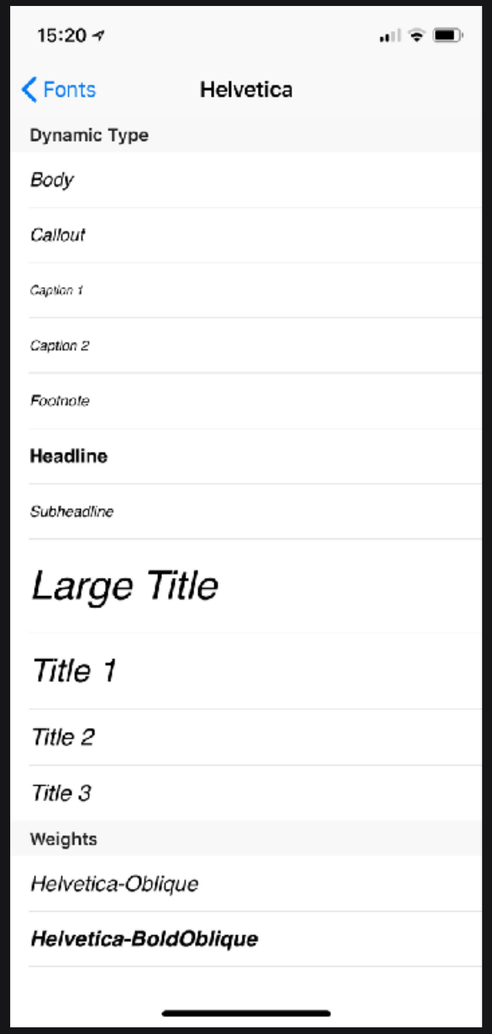


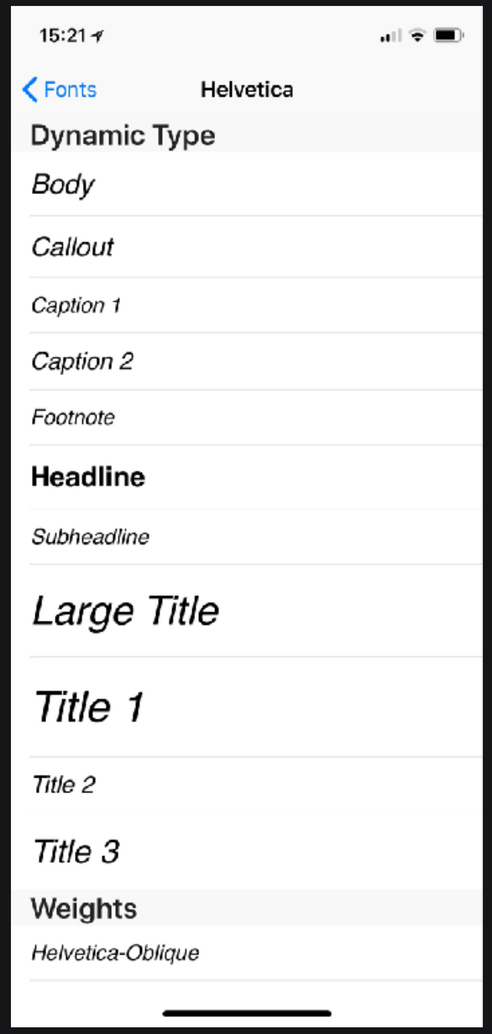
```
UIFont.preferredFont(forTextStyle: style)
```

```
let font = UIFont(familyName: fontName, size: defaultSize)
let fontMetrics = UIFontMetrics(forTextStyle: style)
let scaledFont = fontMetrics.scaledFont(for: font)
```

Dynamic Type

Helvetica





```
UIFont.preferredFont(forTextStyle: style)


let font = UIFont(familyName: fontName, size: defaultSize)
let fontMetrics = UIFontMetrics(forTextStyle: style)
let scaledFont = fontMetrics.scaledFont(for: font)


let scaledValue = fontMetrics.scaledValue(for: 17.0)
```

Dynamic Type

Sizes

Dynamic Type Sizes

The San Francisco typeface was designed to be highly legible at both small and large sizes. Dynamic Type provides additional flexibility by letting readers choose their preferred text size. Download a dynamic type size table in Resources.

Prioritize content when responding to text-size changes. Not all content is equally important. When someone chooses a larger size, they want to make the content they care about easier to read; they don't always want every word on the screen to be larger.

xSmall | Small | Medium | **Large (Default)** | xLarge | xxLarge | xxxLarge

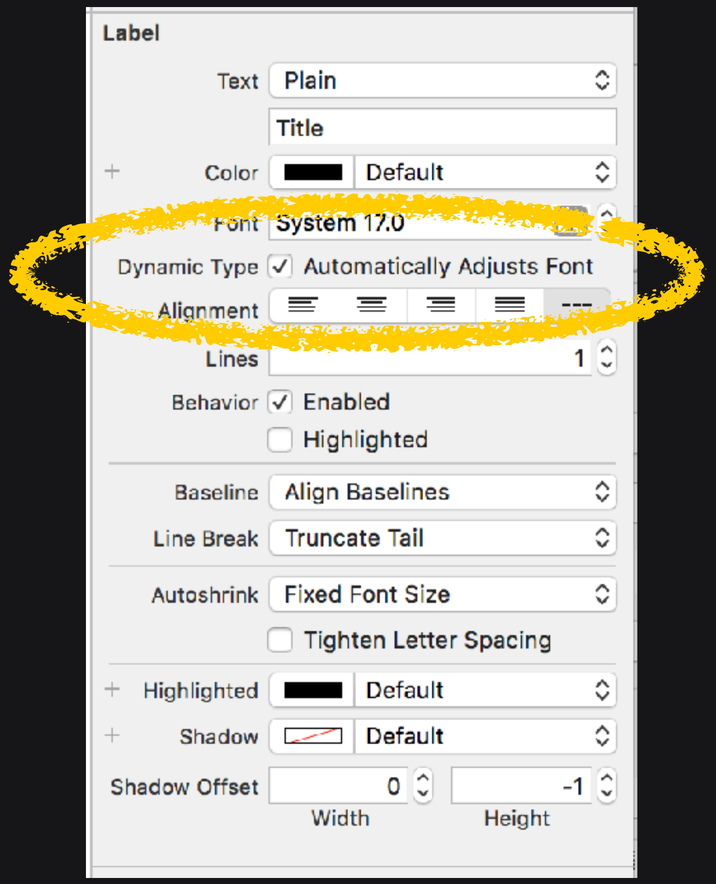
Large (Default)

Style	Weight	Size (Points)	Leading (Points)	Tracking (1/1000em)
Large Title	Regular	34	41	+11
Title 1	Regular	28	34	+13
Title 2	Regular	22	28	+16
Title 3	Regular	20	25	+19
Headline	Semi-Bold	17	22	-24
Body	Regular	17	22	-24
Callout	Regular	16	21	-20
Subhead	Regular	15	20	-16
Footnote	Regular	13	18	-6
Caption 1	Regular	12	16	0
Caption 2	Regular	11	13	+6

https://developer.apple.com/design/human-interface-guidelines/ios/visual-design/typography/

Dynamic Type

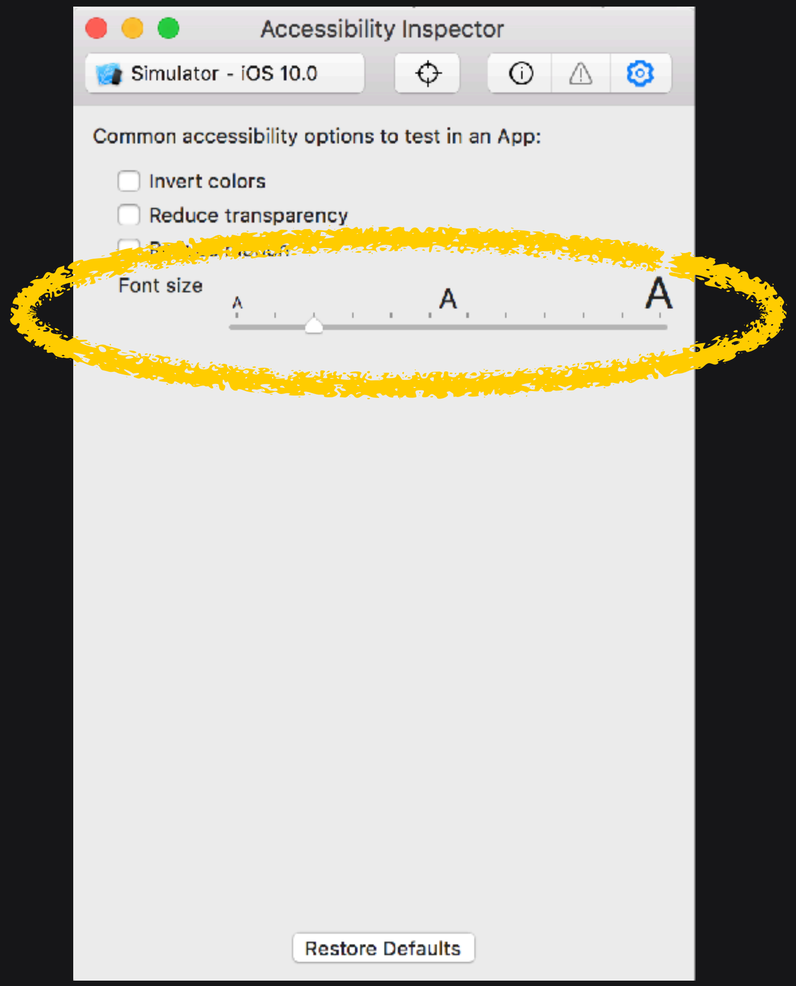
adjustsFontForContentSizeCategory



adjustsFontForContentSizeCategory

Accessibility Inspector

Testing



Closing Thoughts

07

- Standing on the shoulders of giants
- Flexibility and freedom
- Don’t be afraid to follow the white rabbit

Thank You!