June 19, 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 Kit

07 Closing Thoughts

Introduction

01

Ayal Spitz

Senior Principal Software Engineer

Accenture Interactive

@aspitz

ayal.spitz@accenture.com

Motivation & History

02

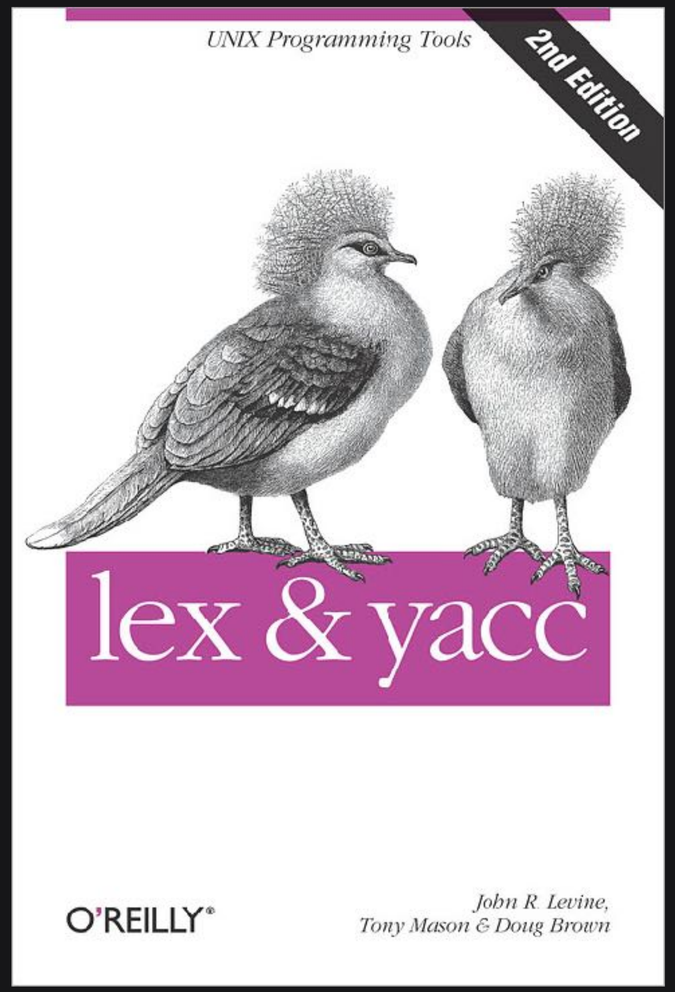
Code → **Compile** → **Run**

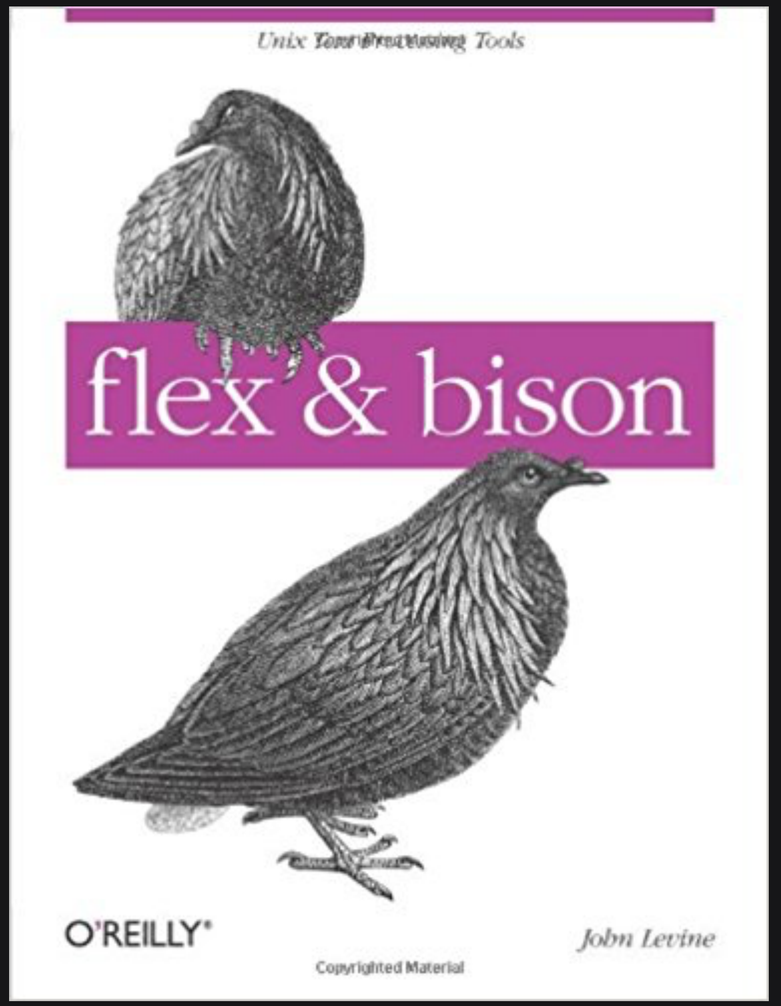
Code → **Compile** → **Run**

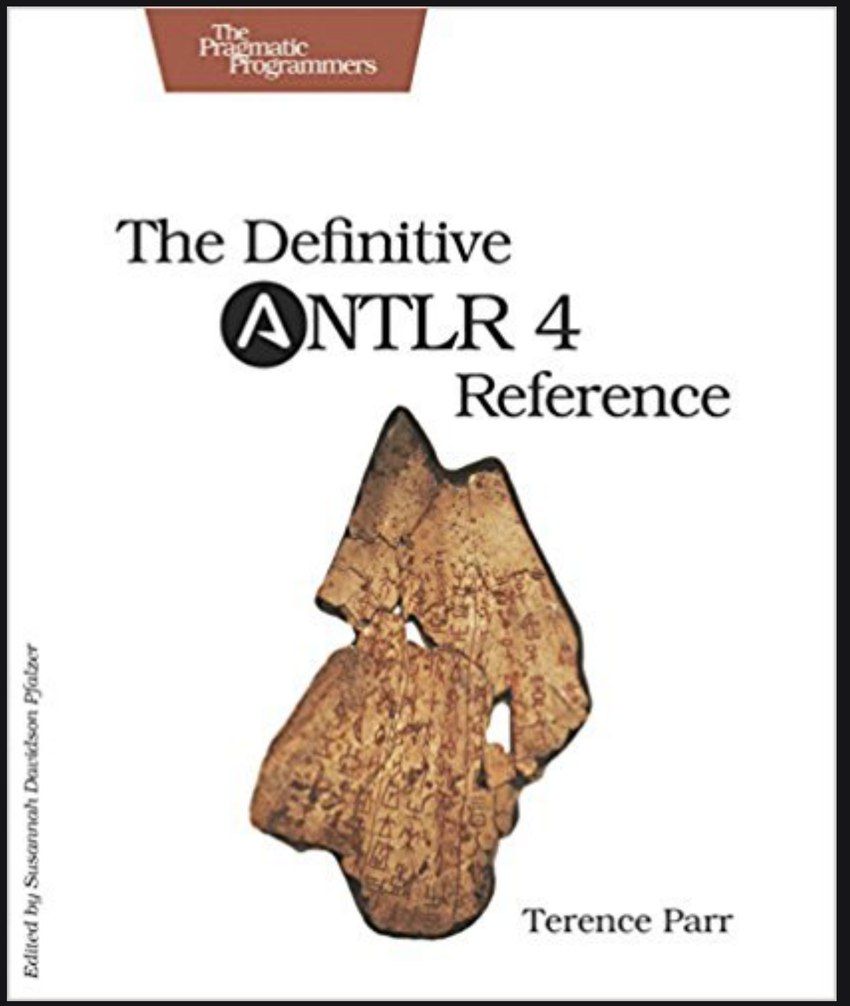
Motivation & History

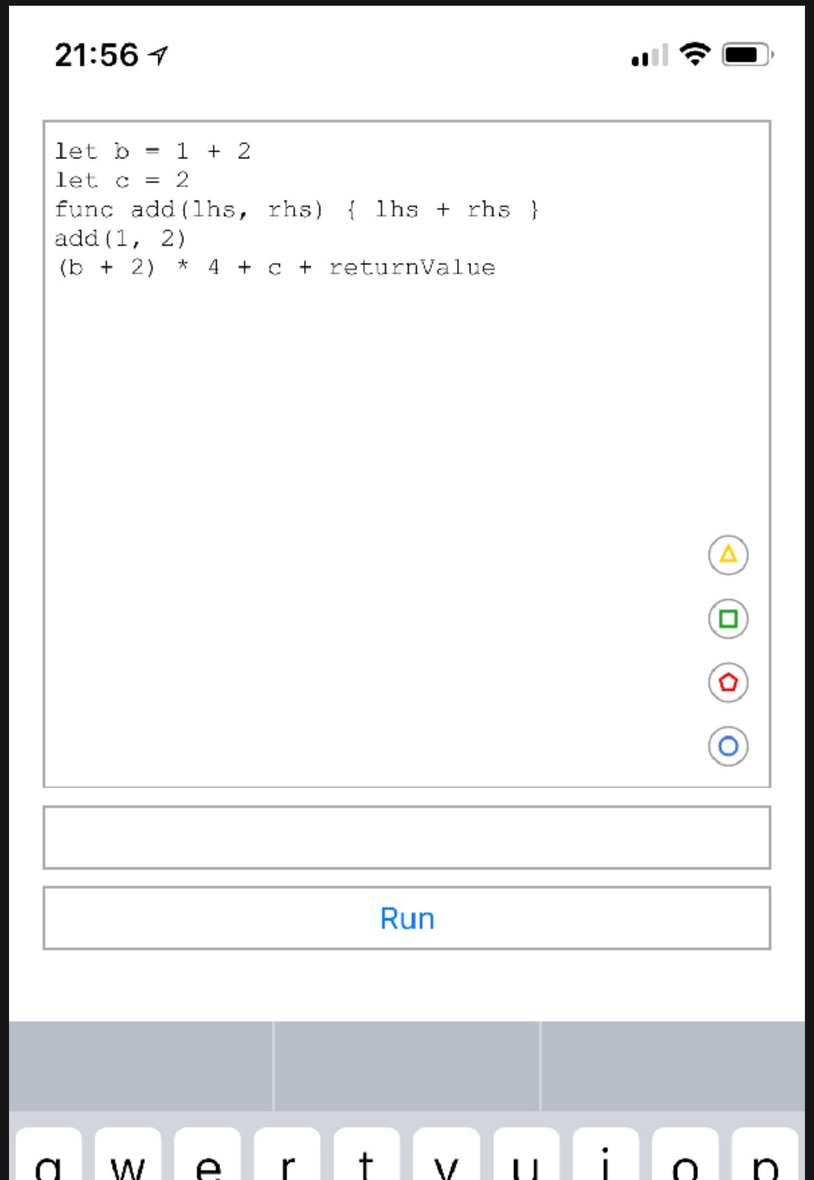
The Art of the Font by Ayal Spitz

The Art of the Font by Ayal Spitz



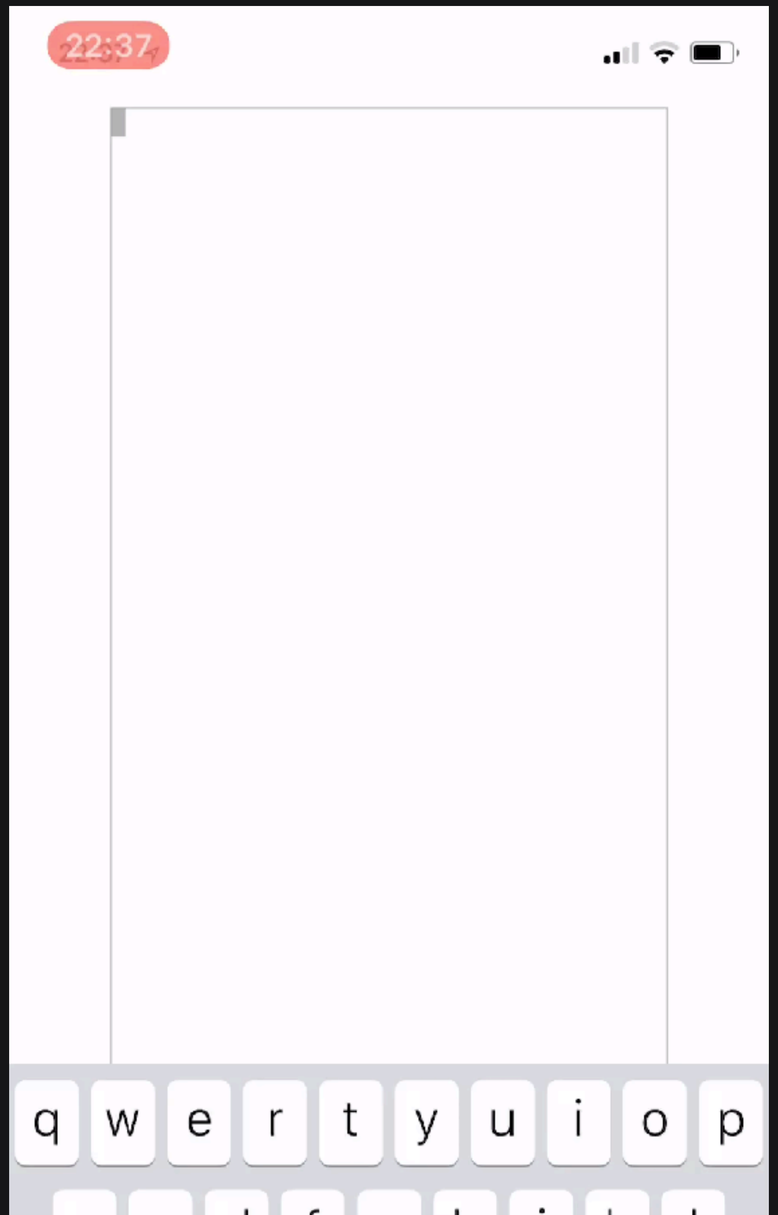





21:56
let b = 1 + 2
let c = 2
func add(lhs, rhs) { lhs + rhs }
add(1, 2)
(b + 2) * 4 + c + returnValue
Run

Motivation & History

The Art of the Font by Ayal Spitz


22:37
q w e r t y u i o p

Calligraphy

Reed College

Calligraphy

[keˈ ligrafe] *noun*

decorative handwriting or handwritten lettering

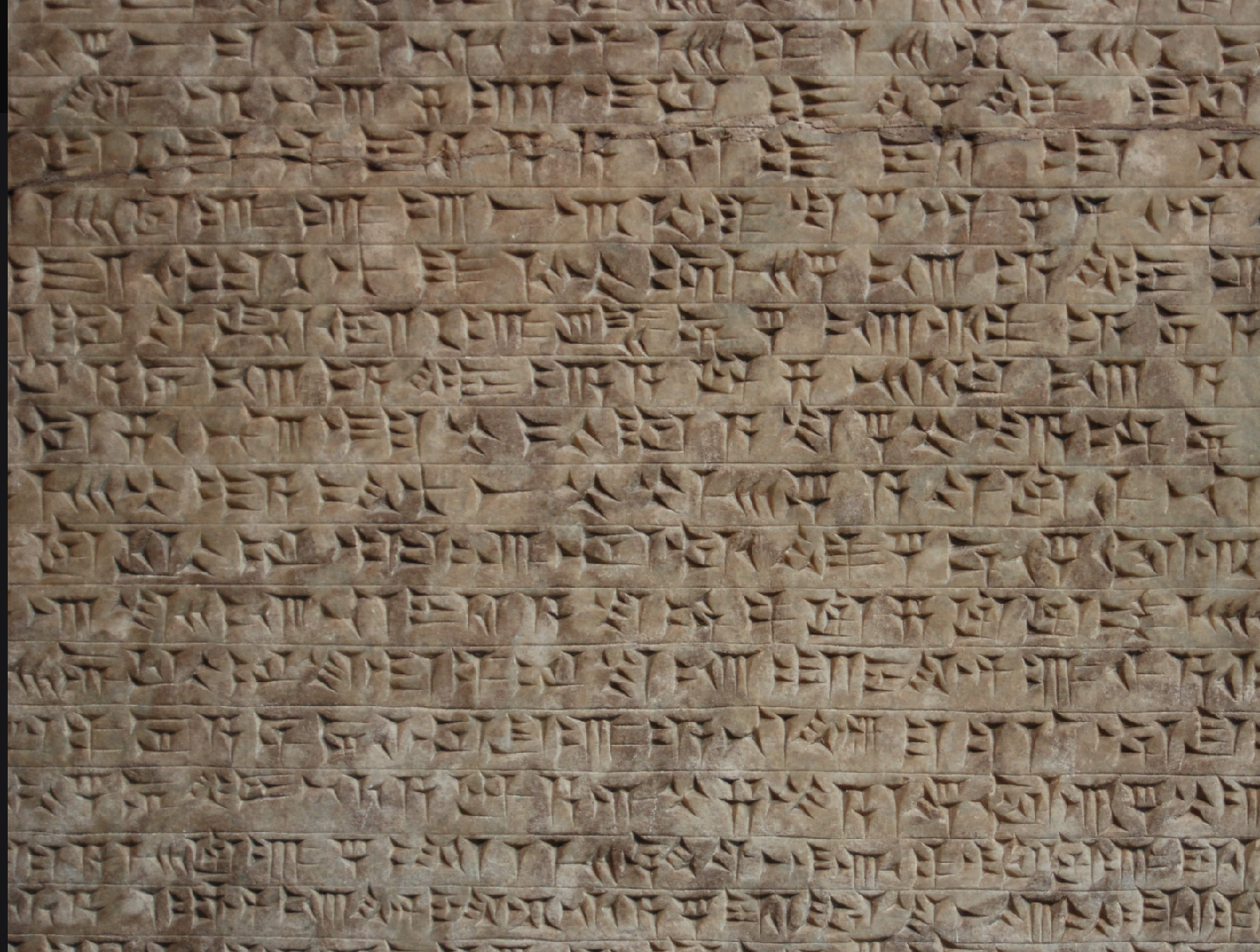
origin & etymology

greek

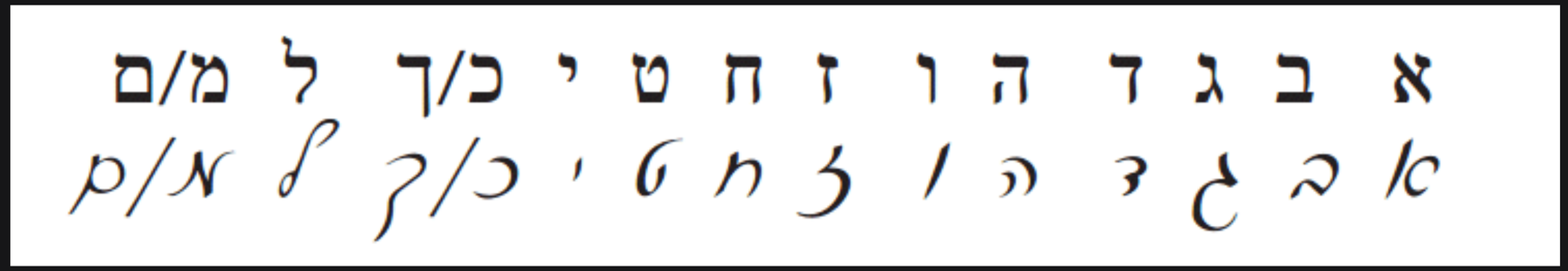
kallos - “beautiful”

graphein - “to write”

Written Language



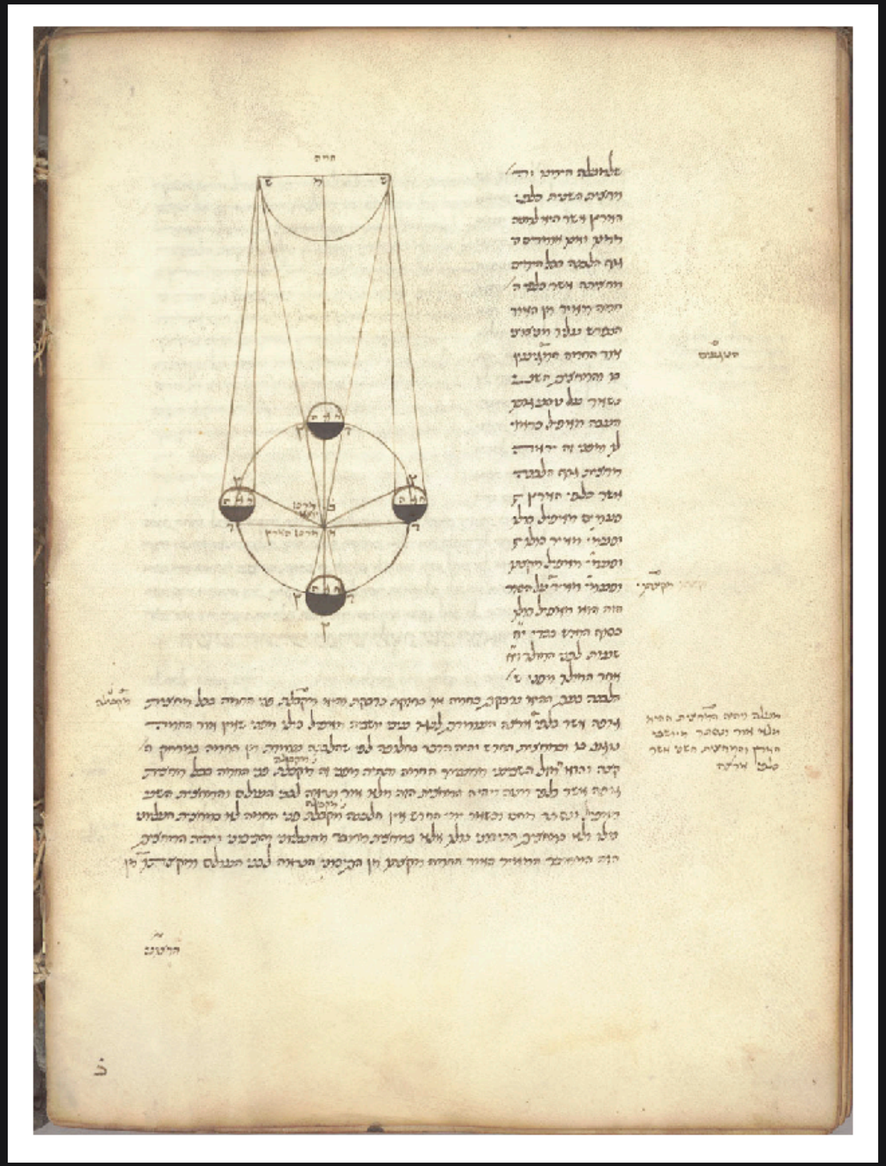
Hebrew



Arabic

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

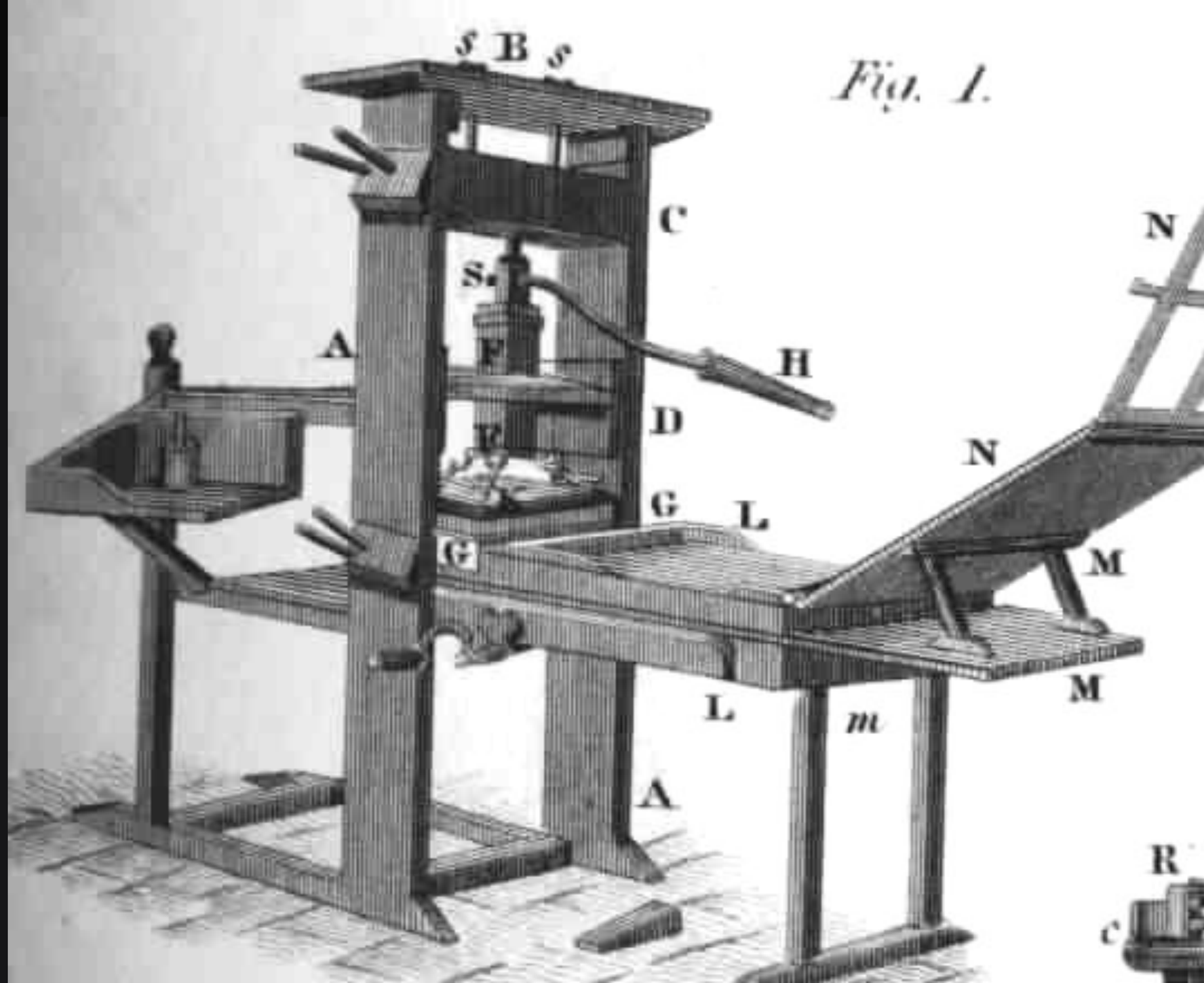
Books



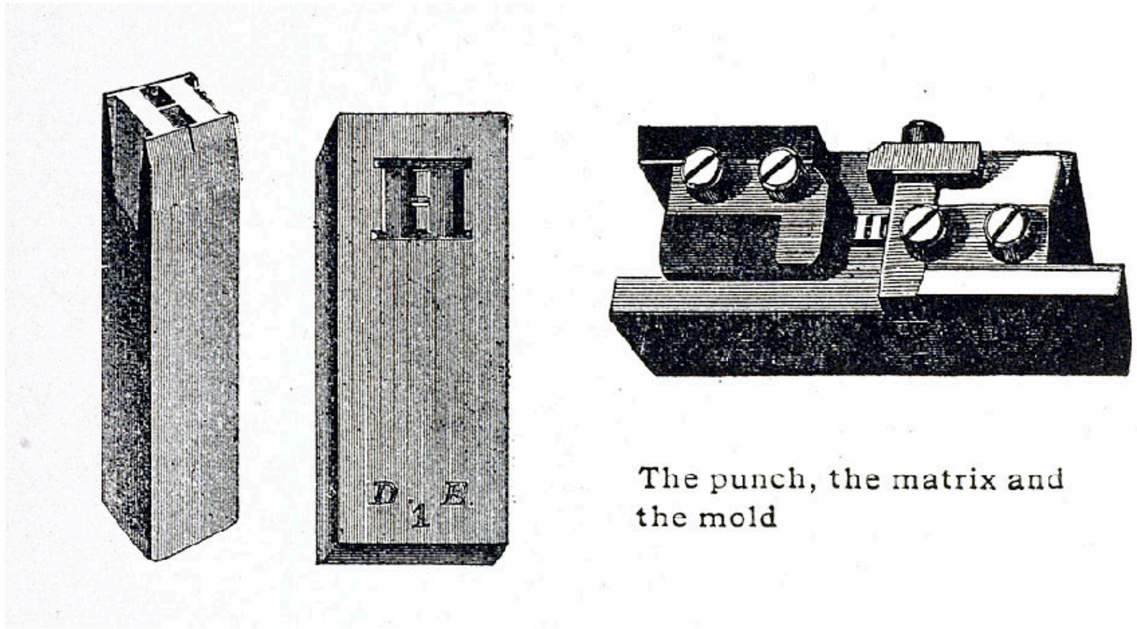
Woodblock Printing

Movable Type

Gutenberg Press

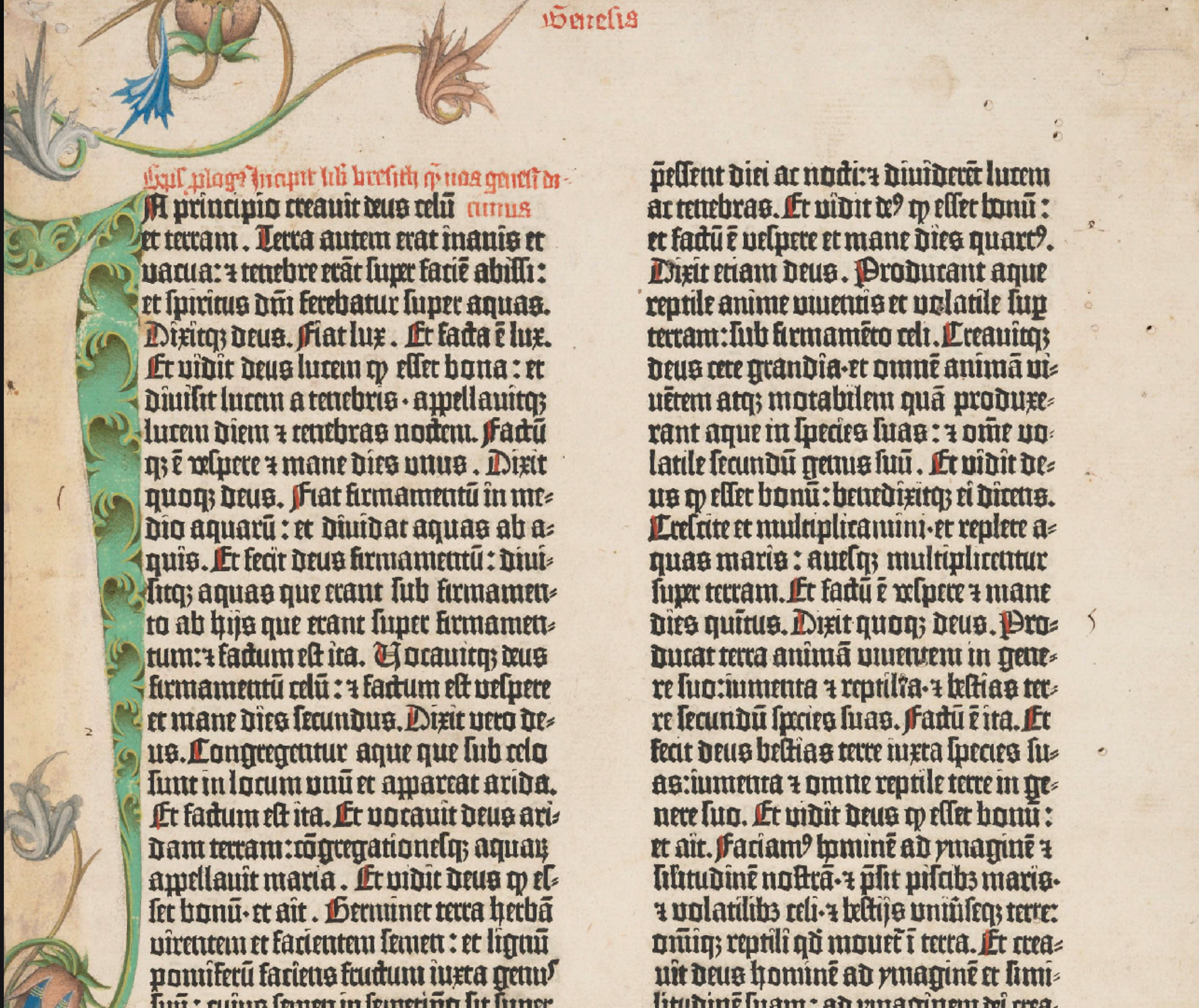


Gutenberg Press



The punch, the matrix and the mold

Gutenberg Bible


Genesis

Incipit liber bresith quē nos genesim di-
In principio creauit deus celū cimus
et terram. Terra autem erat inanis et
vacua: et tenebre erāt super faciē abissi:
et spiritus dñi ferebatur super aquas.
Dixitq; deus. Fiat lux. Et facta ē lux.
Et vidit deus lucem qđ esset bona: et
diuisit lucem a tenebris. appellauitq;
lucem diem et tenebras noctem. Factū
q; ē vespere et mane dies unus. Dixit
quoq; deus. Fiat firmamentū in me-
dio aquarū: et diuidat aquas ab a-
quis. Et fecit deus firmamentū: diui-
sitq; aquas que erant sub firmamen-
to ab hiis que erant super firmamen-
tum: et factum est ita. Vocauitq; deus
firmamentū celū: et factum est vespere
et mane dies secundus. Dixit vero de-
us. Congregentur aque que sub celo
sunt in locum unū et appareat arida.
Et factum est ita. Et vocauit deus ari-
dam terram: cōgregationesq; aquar
appellauit maria. Et vidit deus qđ es-
set bonū. et ait. Germinet terra herbā
virentem et facientem semen: et lignū
pomiferū faciens fructum iuxta genus

pessent diei ac nocti: et diuiderēt lucem
ac tenebras. Et vidit deus qđ esset bonū:
et factū ē vespere et mane dies quartus.
Dixit etiam deus. Producant aque
reptile anime viuentis et volatile sup
terram: sub firmamēto celi. Creauitq;
deus cete grandia. et omnē animā vi-
uētem atq; motabilem quā produxe-
rant aque in species suas: et omē vo-
latile secundū genus suū. Et vidit de-
us qđ esset bonū: benedixitq; ei dicens.
Crescite et multiplicamini. et replete a-
quas maris: avesq; multiplicentur
super terram. Et factū ē vespere et mane
dies quintus. Dixit quoq; deus. Pro-
ducat terra animā viuentem in gene-
re suo: iumenta et reptilia. et bestias ter-
re secundū species suas. Factū ē ita. Et
fecit deus bestias terre iuxta species su-
as: iumenta et omne reptile terre in ge-
nere suo. Et vidit deus qđ esset bonū:
et ait. Faciamus hominē ad ymaginē et
similitudinē nostrā. et psit piscibus maris.
et volatilibus celi. et bestiis uniuersæq; terre:
omniq; reptili qđ mouet ī terra. Et crea-
uit deus hominē ad ymaginē et simi-

Fonts & Apple

03

Apple II

DOS

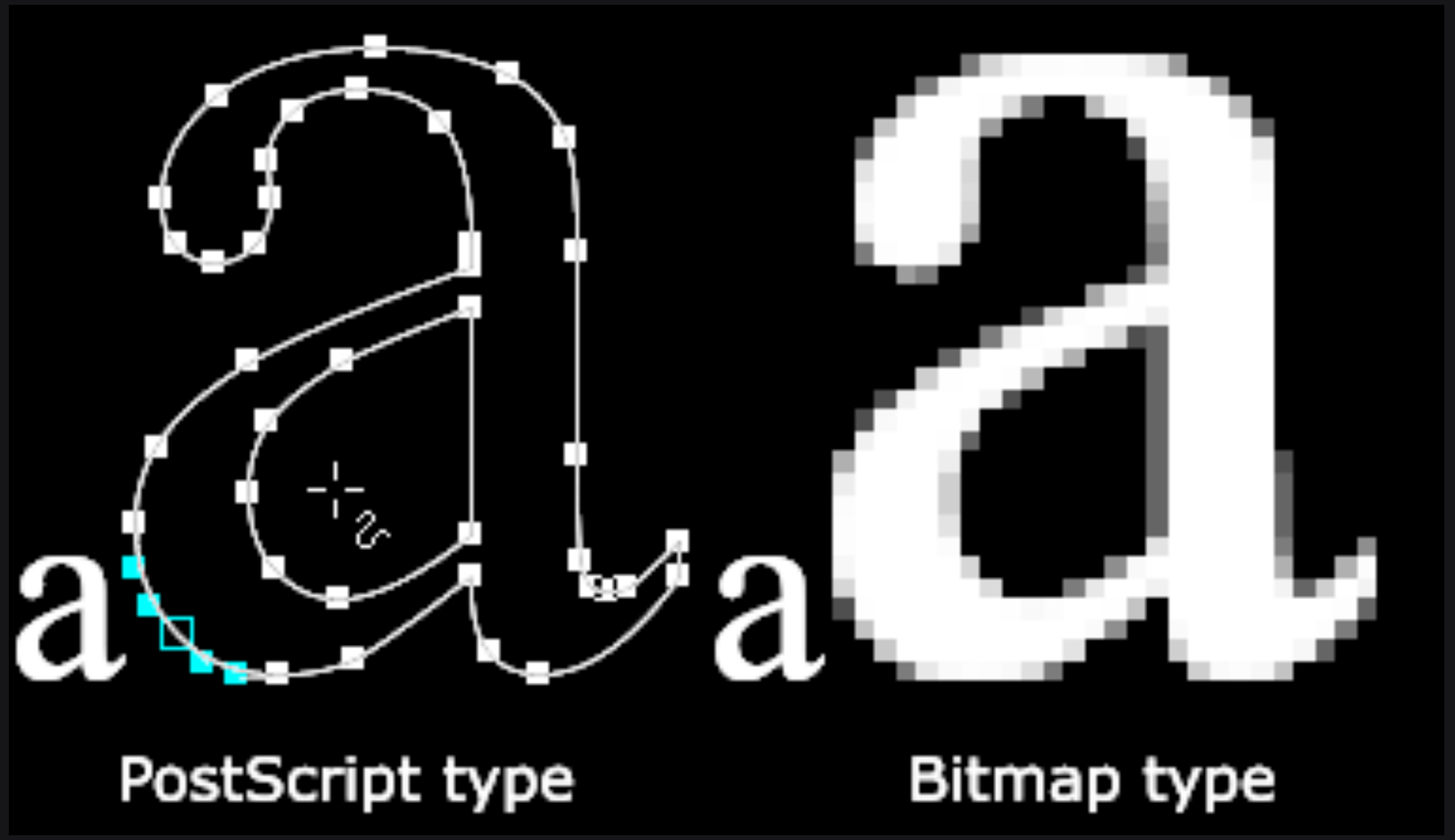

APPLE II
DOS VERSION 3.3 SYSTEM MASTER
JANUARY 1, 1983
COPYRIGHT APPLE COMPUTER,INC. 1980,1982
]

Mac Fonts

Mac OS 1 - 9

Adobe PostScript

Outline Font



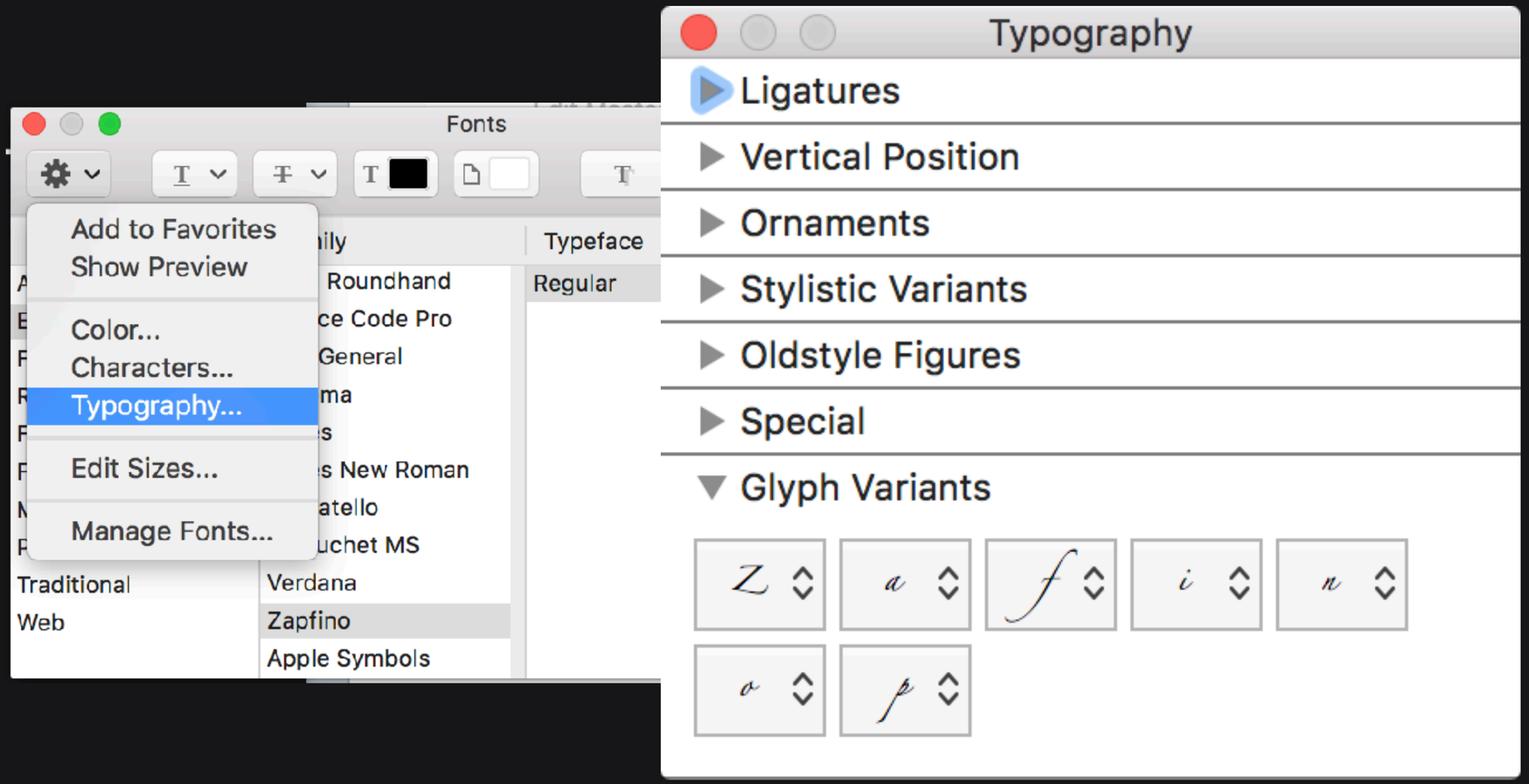
TrueType

OpenType

Apple Advanced Typography

Apple Advanced Typography

Apple Advanced Typography

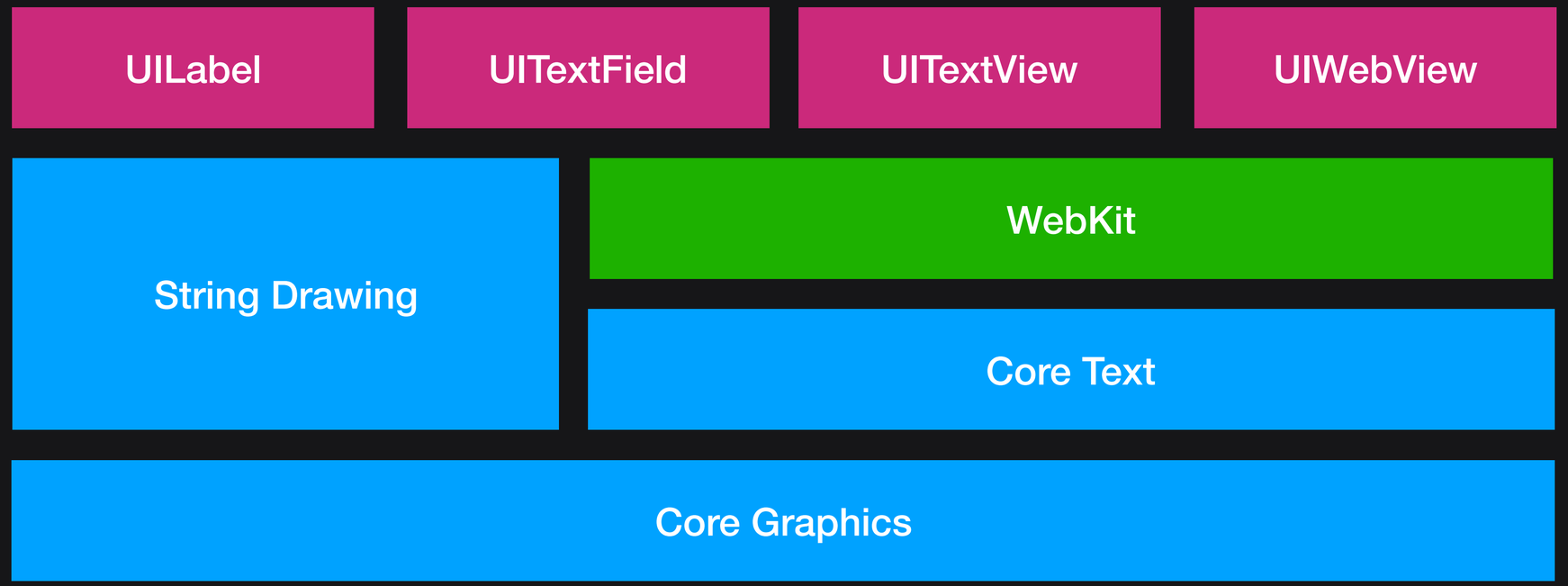


CoreText

04

CoreText

per iOS 7

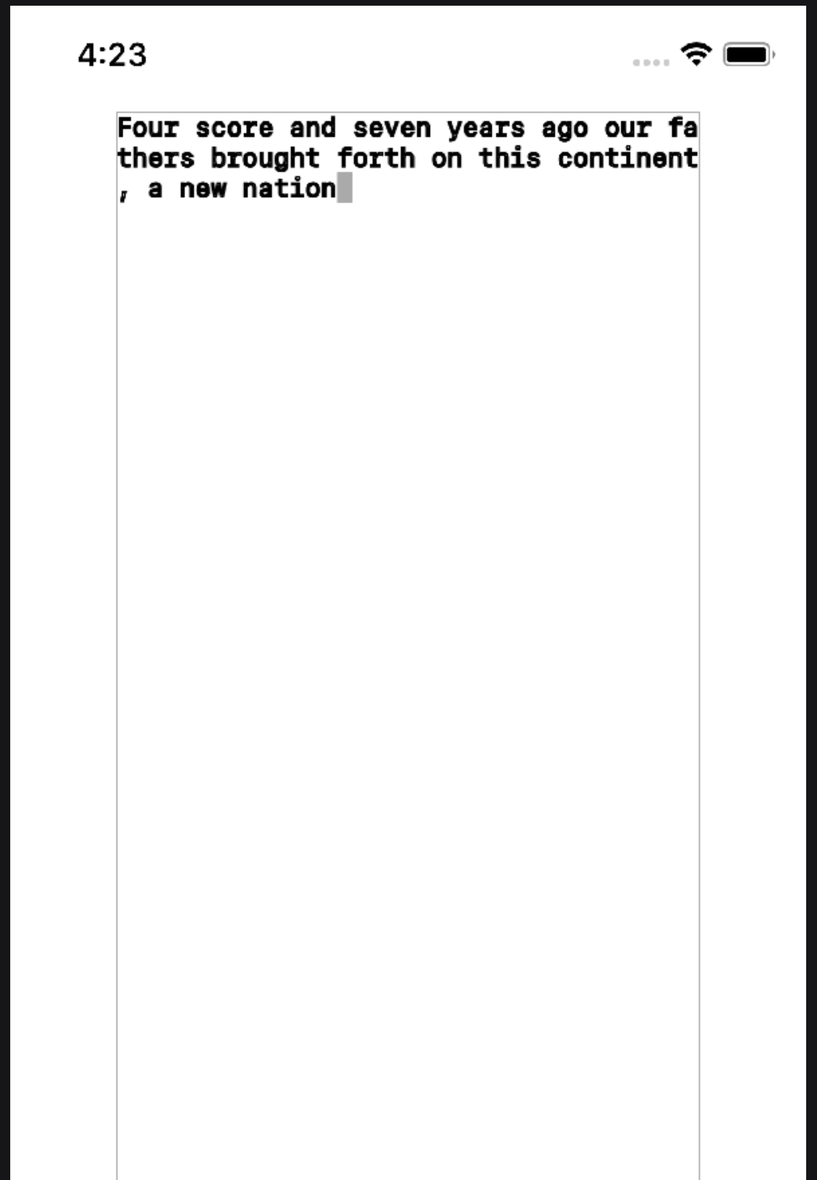


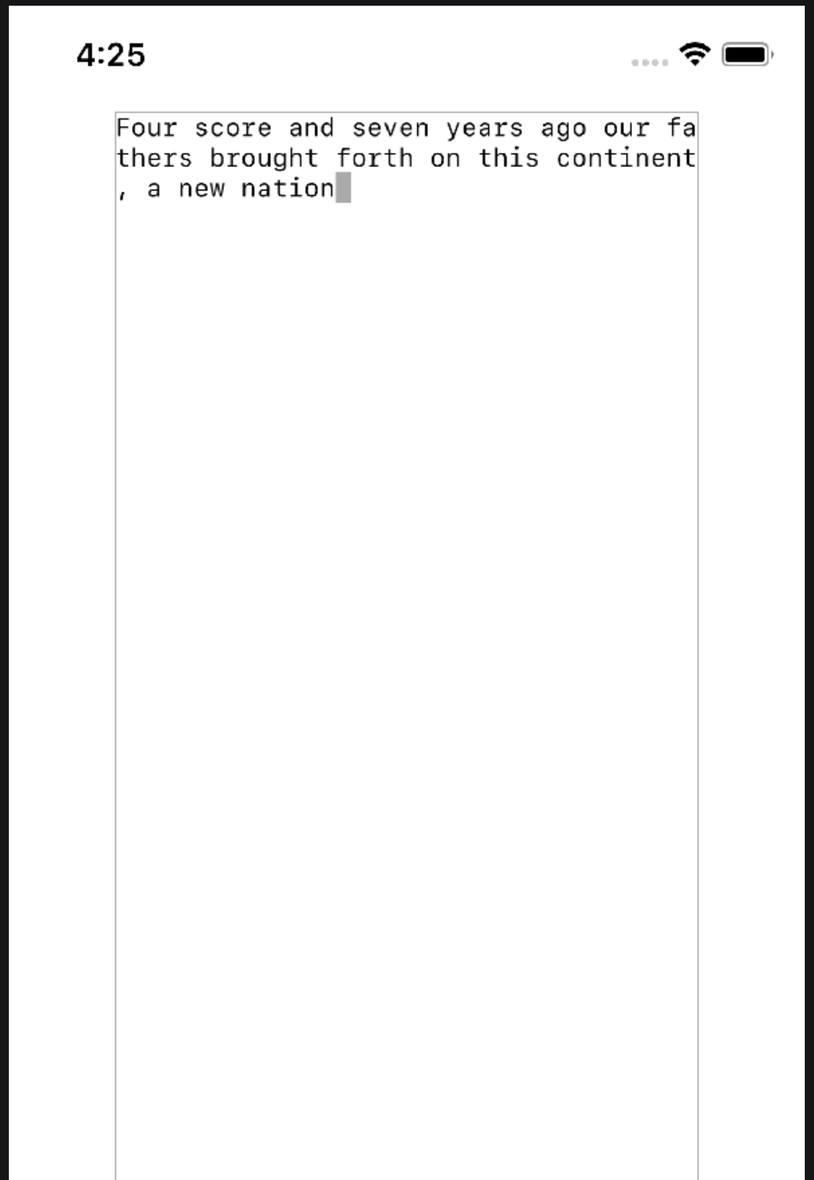
Example

```
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
    }
}


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)
        }
    }
}
```

Example

```
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

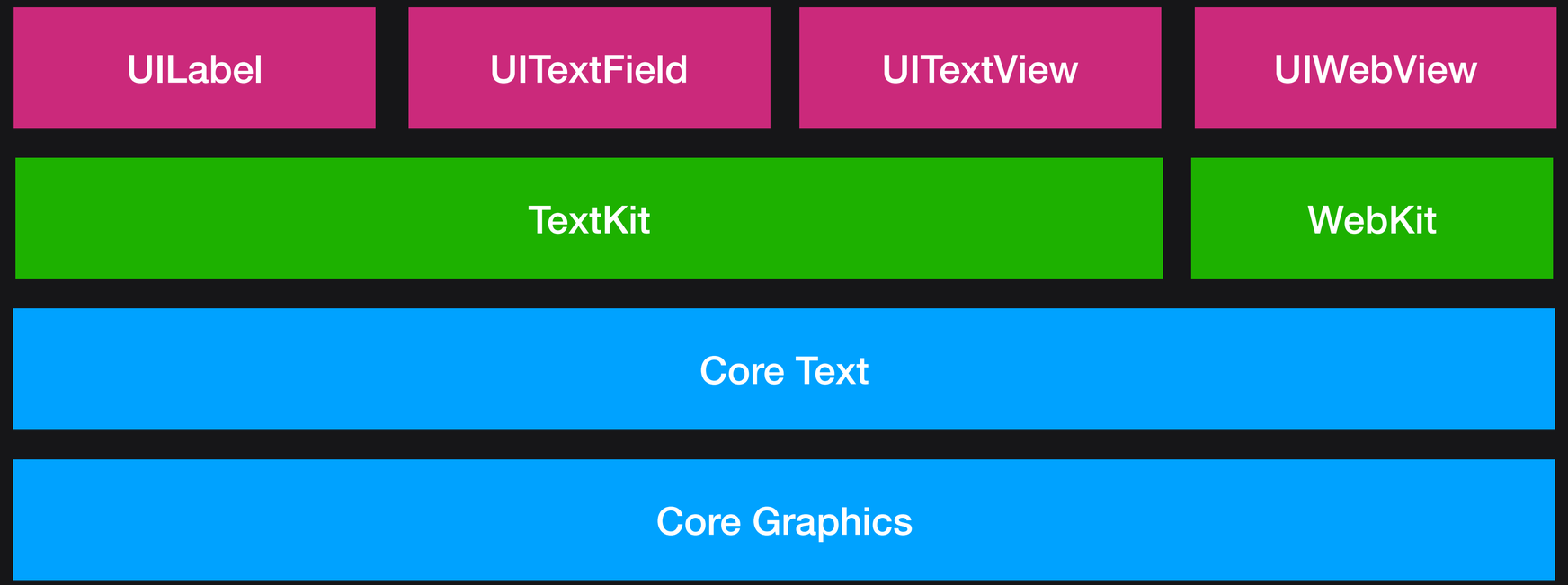

4:25
Four score and seven years ago our fa
thers brought forth on this continent
, a new nation

TextKit

05

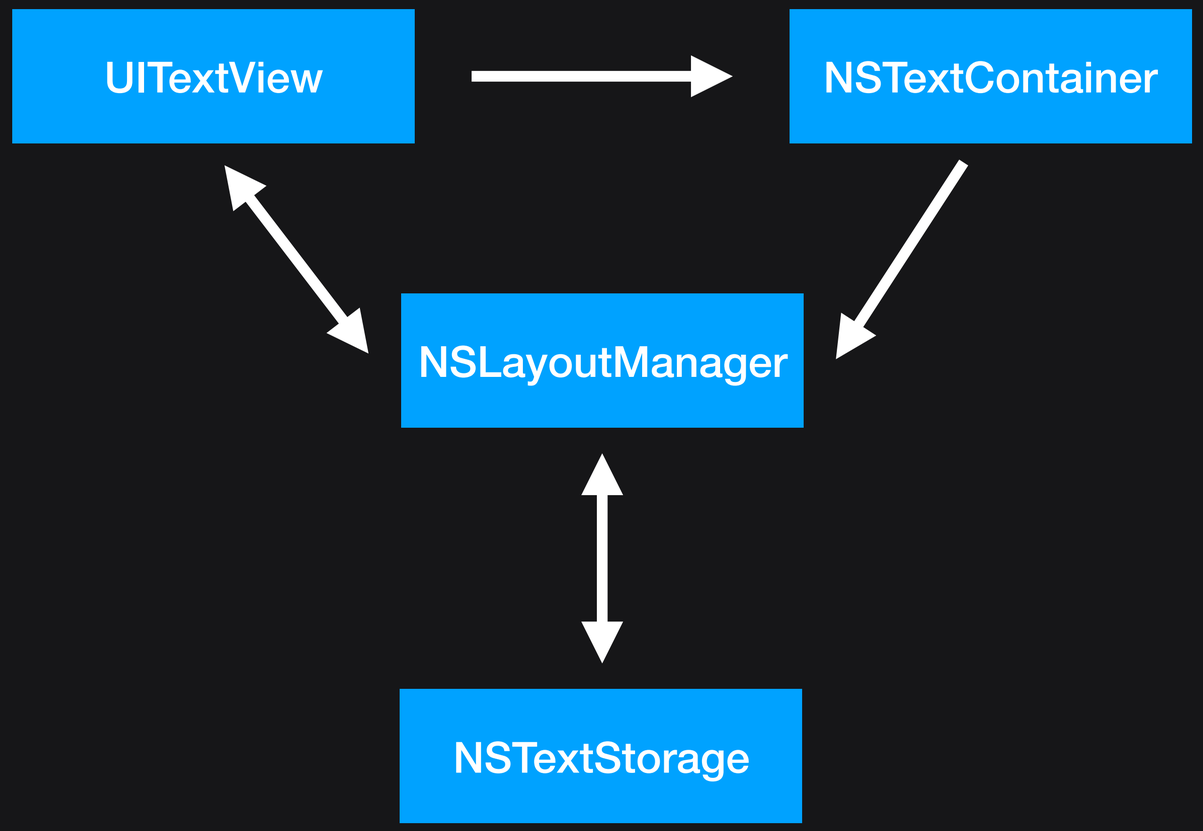
TextKit

iOS 7



TextKit

iOS 7



Custom Text Storage

```
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()
    }
}
```

Custom Text Container

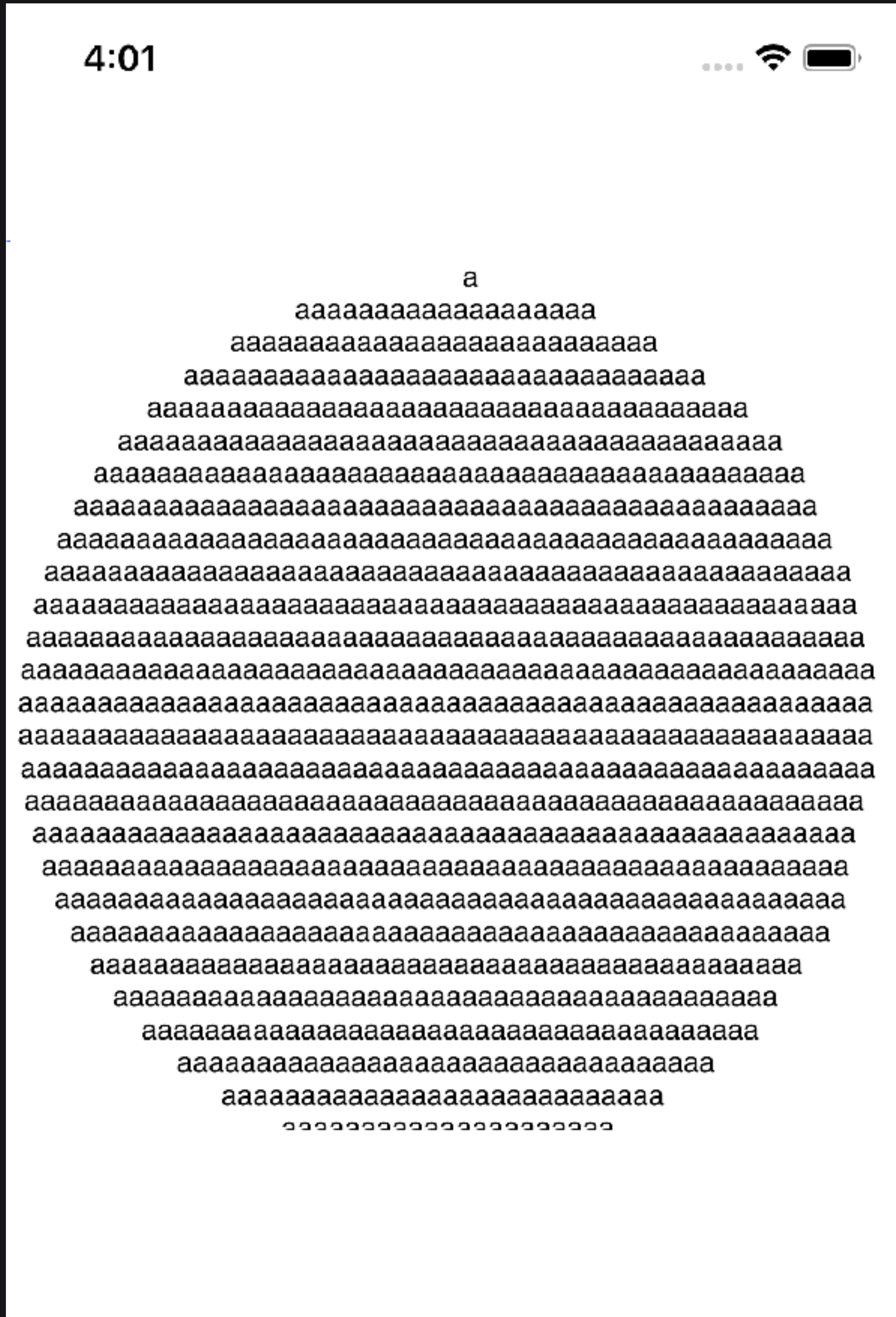
```
import UIKit

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

    }
}
```

Using Custom Text Storage

```
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

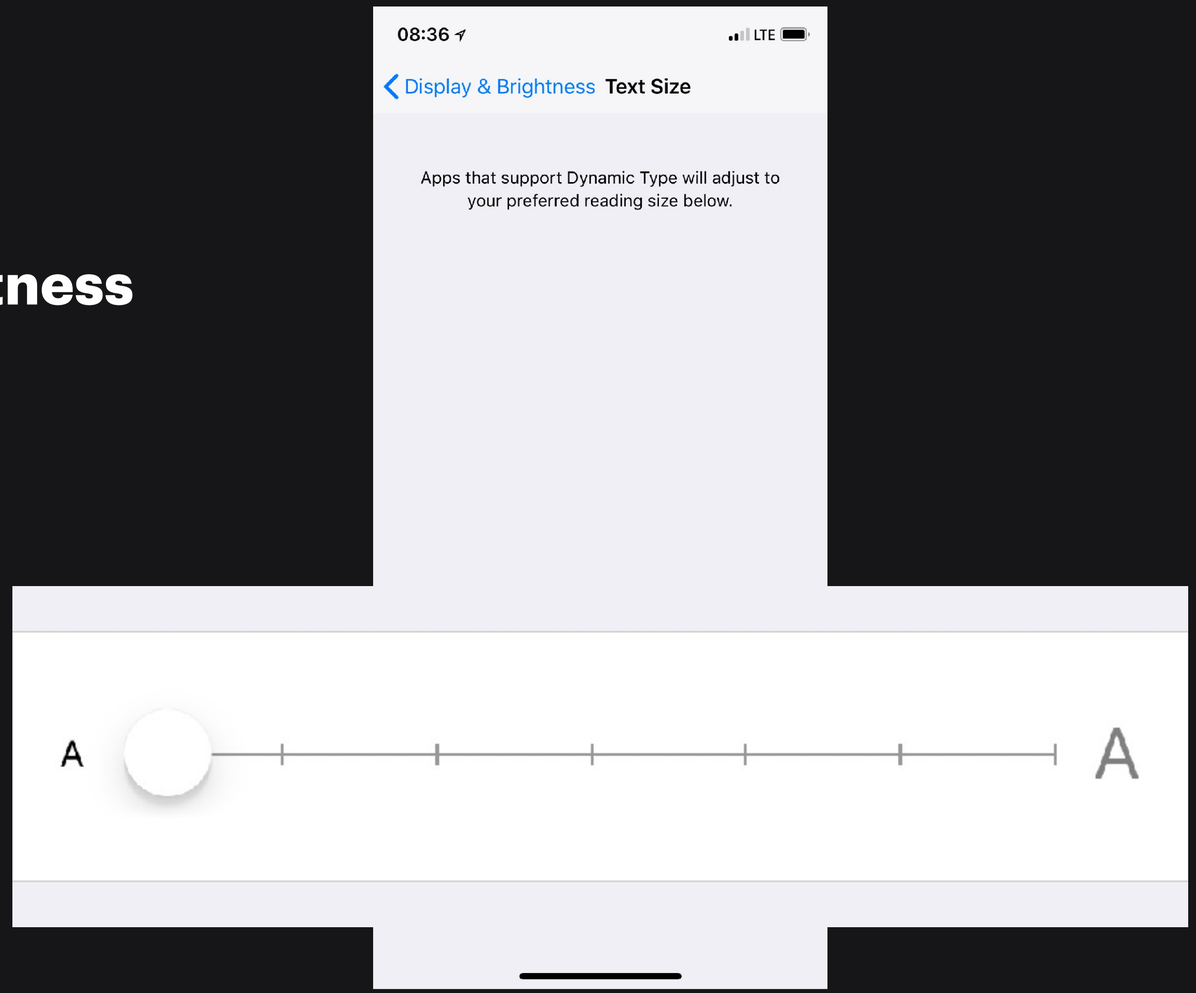


Dynamic Type

06

Dynamic Type

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



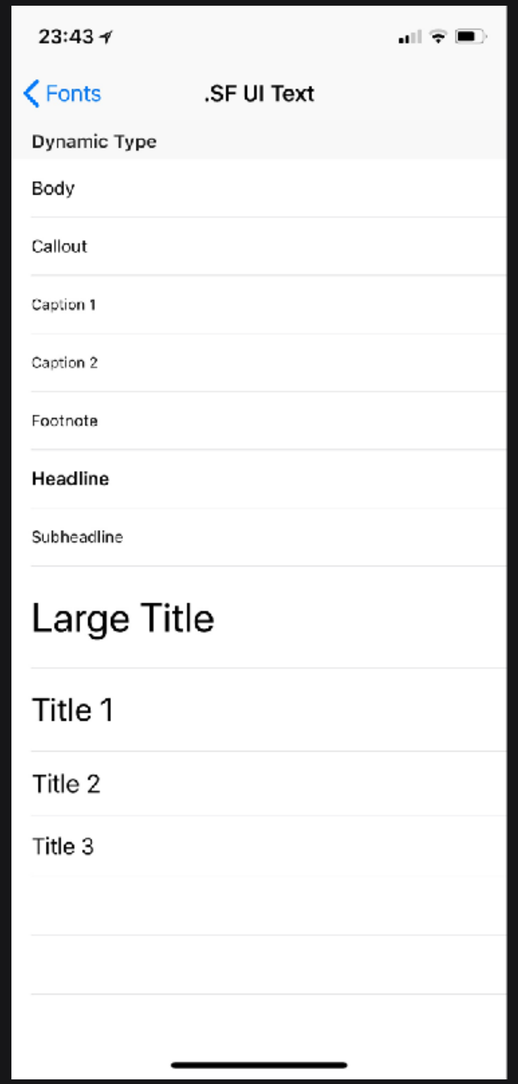
Dynamic Type

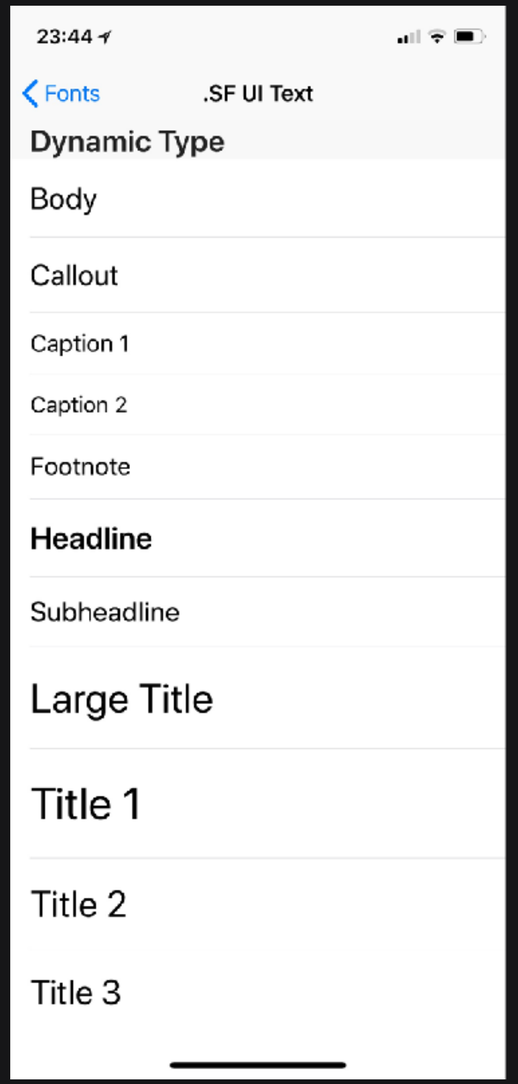
```
UIFont.preferredFont(forTextStyle: style)
```

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

Dynamic Type

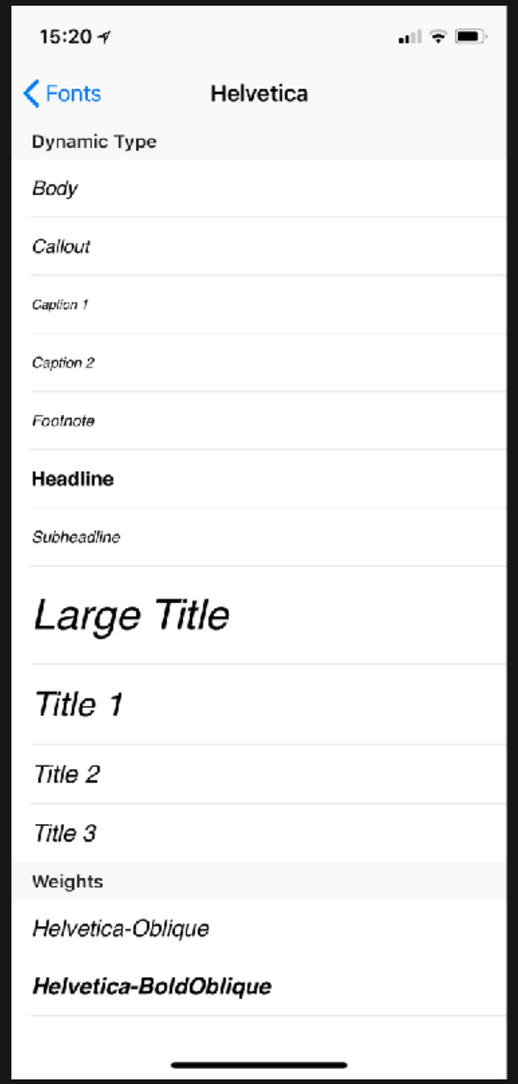
Preferred Font

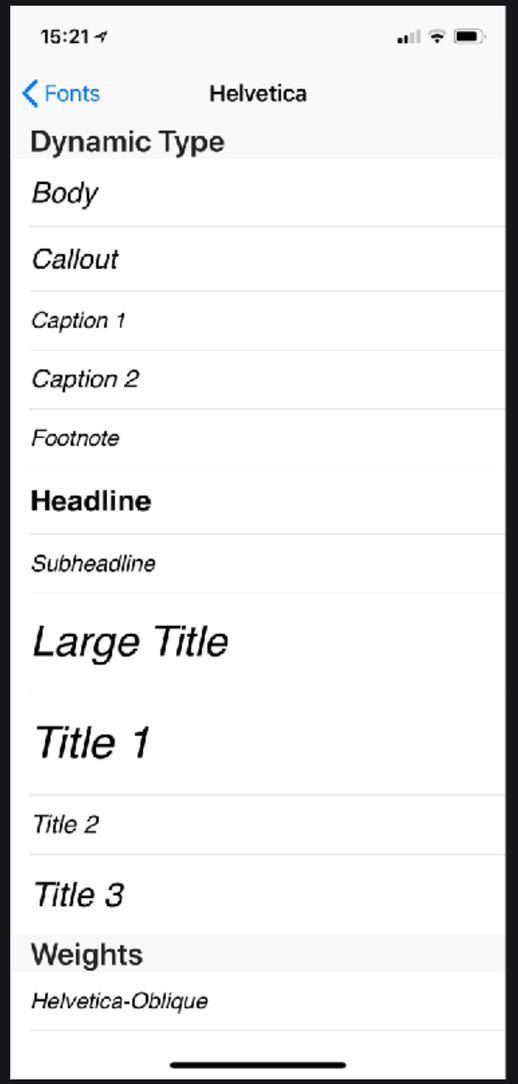




Dynamic Type

Helvetica





Tips

- UIKit elements already have Dynamic Type backed in
- Use AutoLayout
- Don’t forget to set `adjustsFontForContentSizeCategory`
- `UIFontMetrics.scaledValue(forValue:)`
- Test Frequently

Closing Thoughts

07

- Standing on the backs of giants
- Flexibility and freedom
- Don’t be afraid to follow the white rabbit
- Understanding history can provide you with new insight and appreciation for what we have

Thank You!