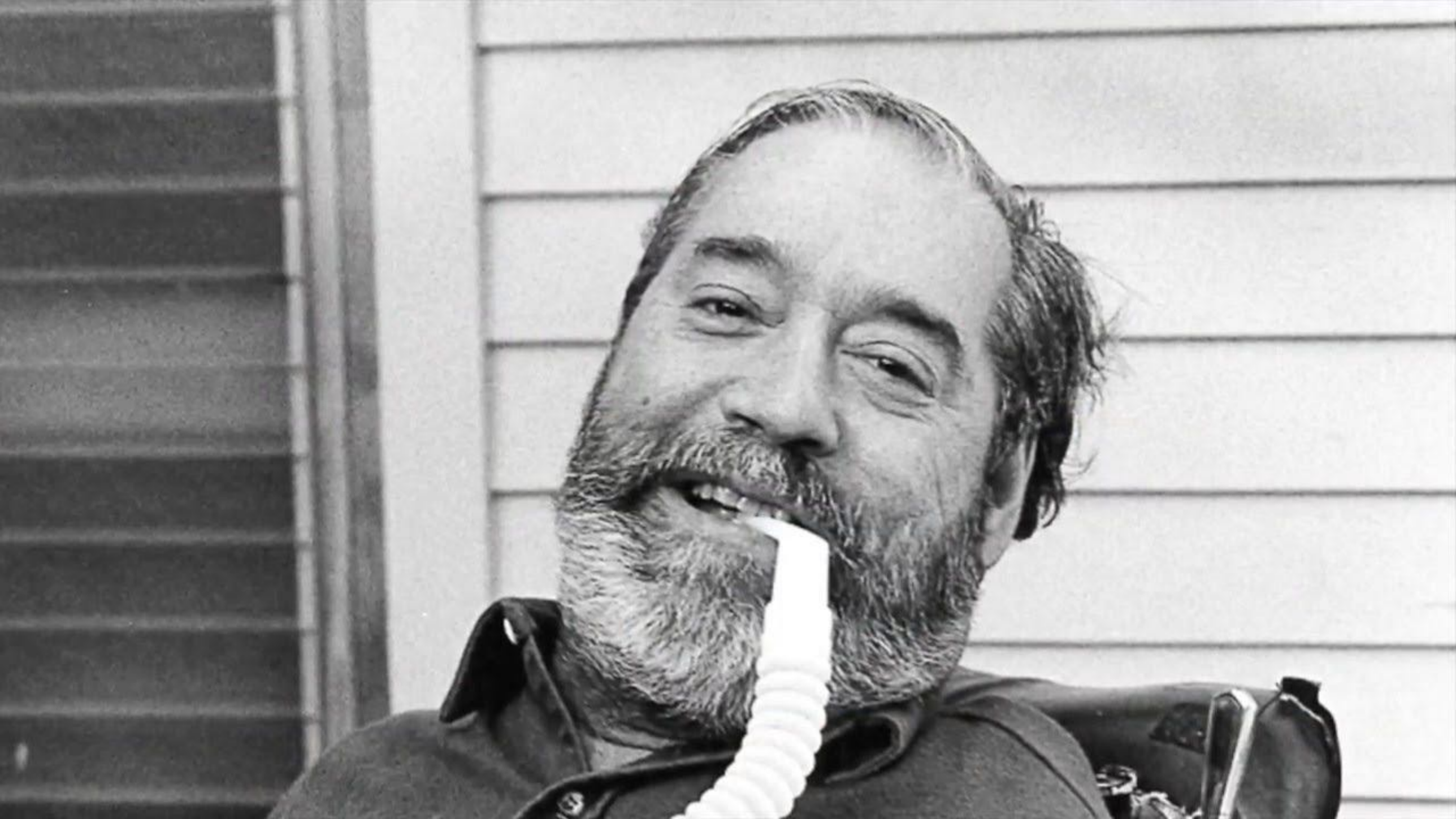


Universal Design

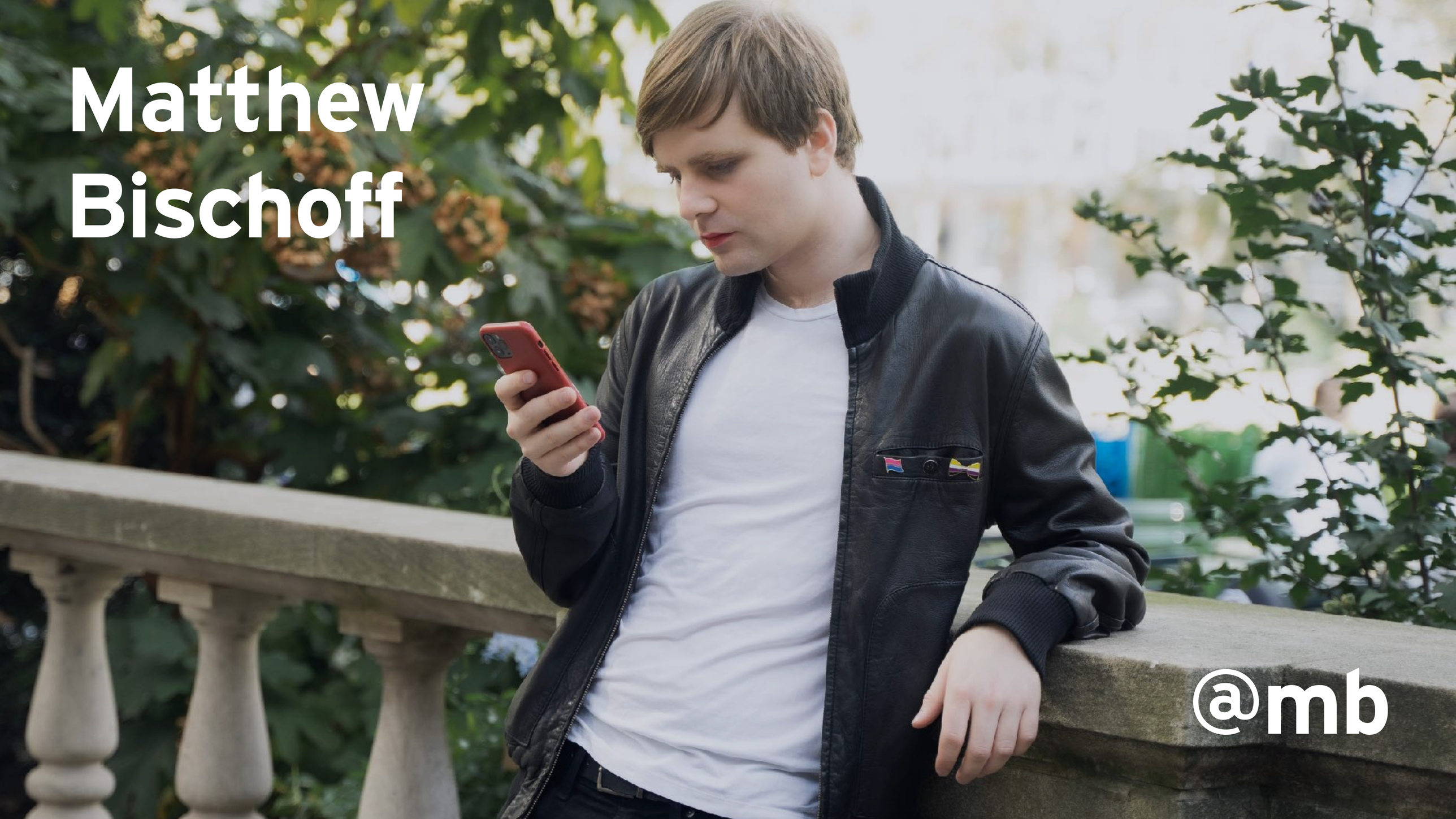
"The concept of designing all products and the built environment to be aesthetic and usable to the greatest extent possible by everyone, regardless of their age, ability, or status in life."

Ronald Mace

1. Equitable use
2. Flexibility in use
3. Simple and intuitive
4. Perceptible information
5. Tolerance for error
6. Low physical effort
7. Size and space for approach & use

Hi

Hi


Matthew
Bischoff
@mb

🗽 NYC
🏳️‍🌈 They/them
🚶‍♂️ Non-disabled

APPS FOR ALL

APPS FOR ALL

MAKING SOFTWARE ACCESSIBLE

I Don't Know How To Explain To You That You Should Care About Other People

Our apps aren't accessible

Why not?

😬 Excuses

🧑‍🤝‍🧑 But most of my users...

But my boss...

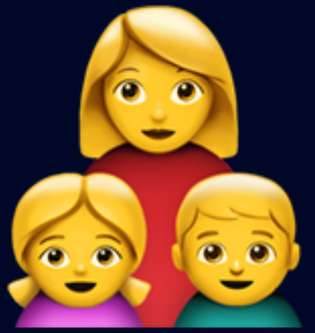
But it's difficult...

But what's the ROI...

“When we work on making our devices accessible by the blind, I don’t consider the bloody ROI.”
Tim Cook

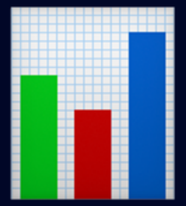
Apps must be accessible

🤔 Why?



“1 in 7 people have a disability or impairment that affects the way they interact with the world and their devices.”

 Human Interface Guidelines



14% of your customers

Matthew Bischoff
@mb

Are you a disabled person who uses iOS Accessibility features? I'd love to briefly interview you for an upcoming talk.

Please email a list of the features you use with the subject "Accessibility" at mb@matthewbischoff.com. I can compensate you for your time.

RTs appreciated.

1:13 PM · Apr 13, 2020 · Twitter for Mac

View Tweet activity

116 Retweets **46** Likes

Tanya Harrison
she/her

Ell Schulman
ze/zem

Permanent

Temporary

Situational

	Permanent	Temporary	Situational
Touch	One arm	Arm injury	New parent
See	Blind	Cataract	Distracted driver
Hear	Deaf	Ear infection	Bartender
Speak	Non-verbal	Laryngitis	Heavy accent

Inclusive
A Microsoft Design Toolkit

It's the law

POLITICS

Supreme Court hands victory to blind man who sued Domino's over site accessibility

PUBLISHED MON, OCT 7 2019•9:40 AM EDT | UPDATED TUE, OCT 8 2019•6:31 PM EDT

Tucker Higgins
@IN/TUCKER-HIGGINS-5B162295/
@TUCKERHIGGINS

SHARE

KEY POINTS

- The Supreme Court denied a petition from pizza giant Domino's on Monday to hear whether its website is required to be accessible to the disabled, leaving in place a lower court decision against the company.
- The case was originally brought by a blind man named Guillermo Robles, who sued the pizza chain after he was unable to order food on Domino's website and mobile app despite using screen-reading software.
- The decision not to grant the case is a loss for the company and a win for disability advocates, who have argued that if businesses do not have to maintain accessible sites, disabled people could be effectively shut out of substantial portions of the economy.

It's the right thing to do

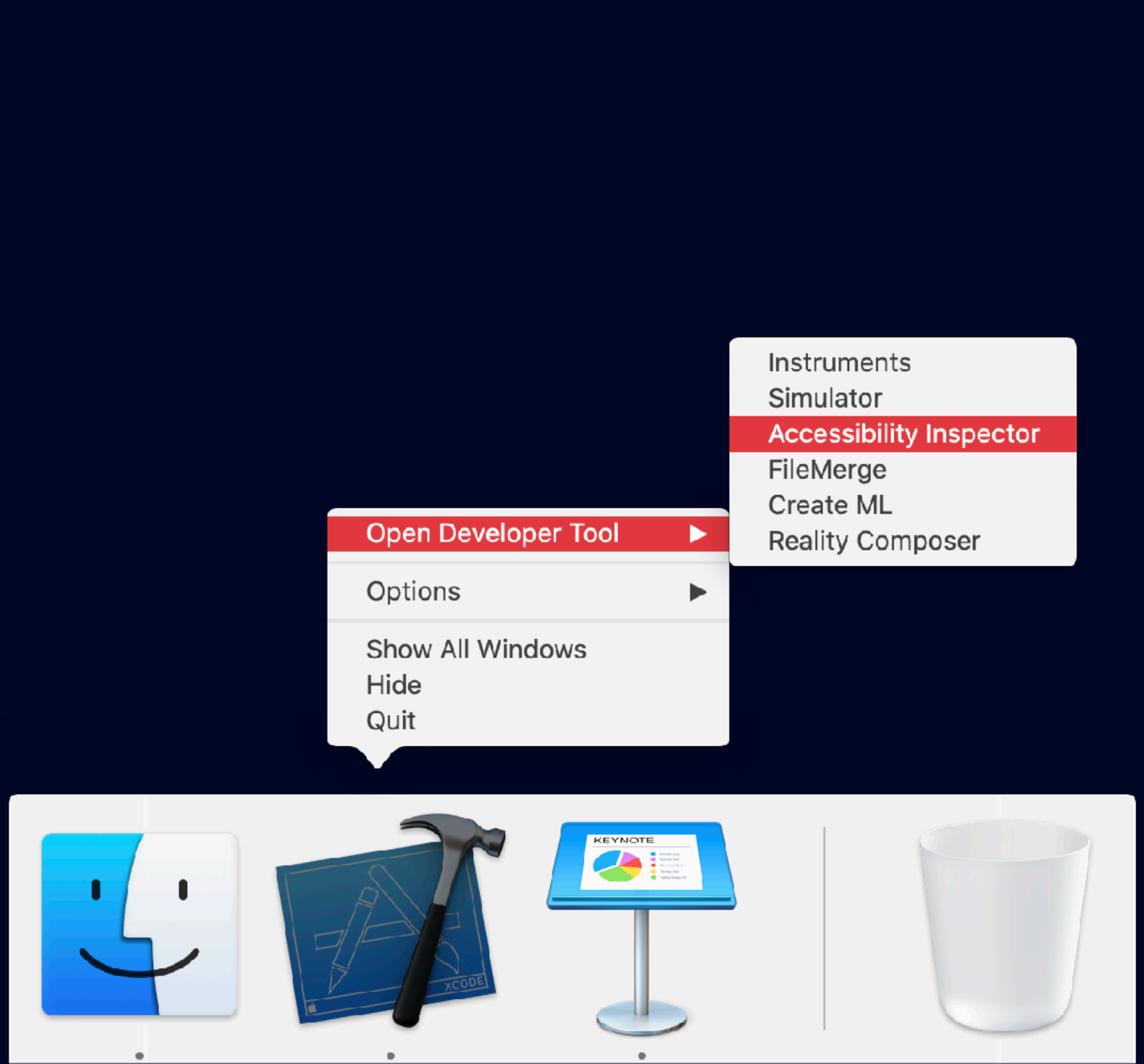
“Accessibility failures should be embarrassments to all developers because they’re usually very easy to fix… Rare ‘complex’ issues are usually less than an hour’s work.”

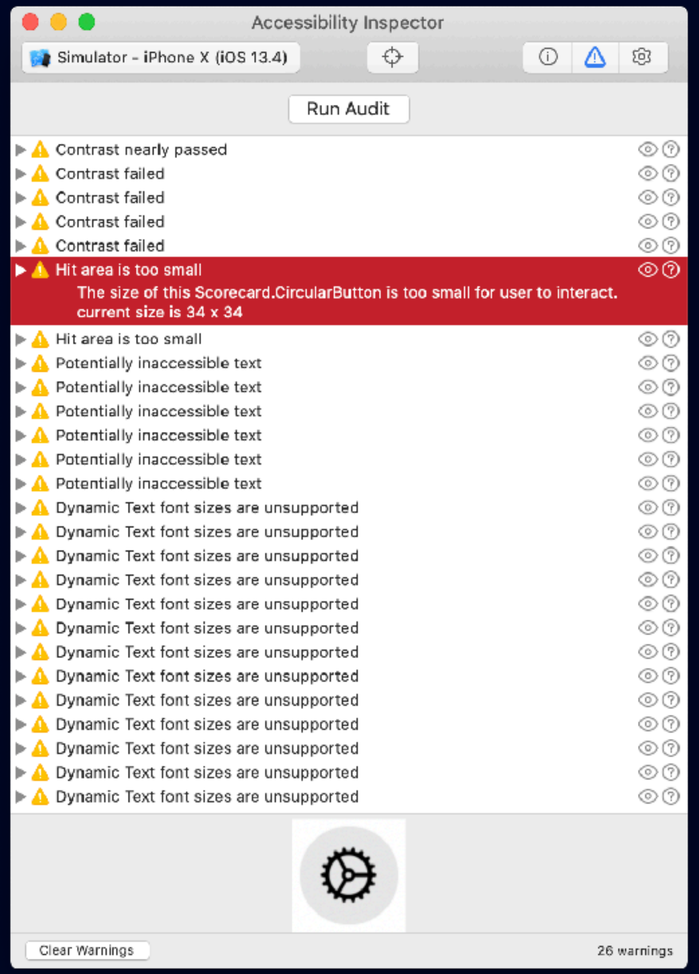
Marco Arment

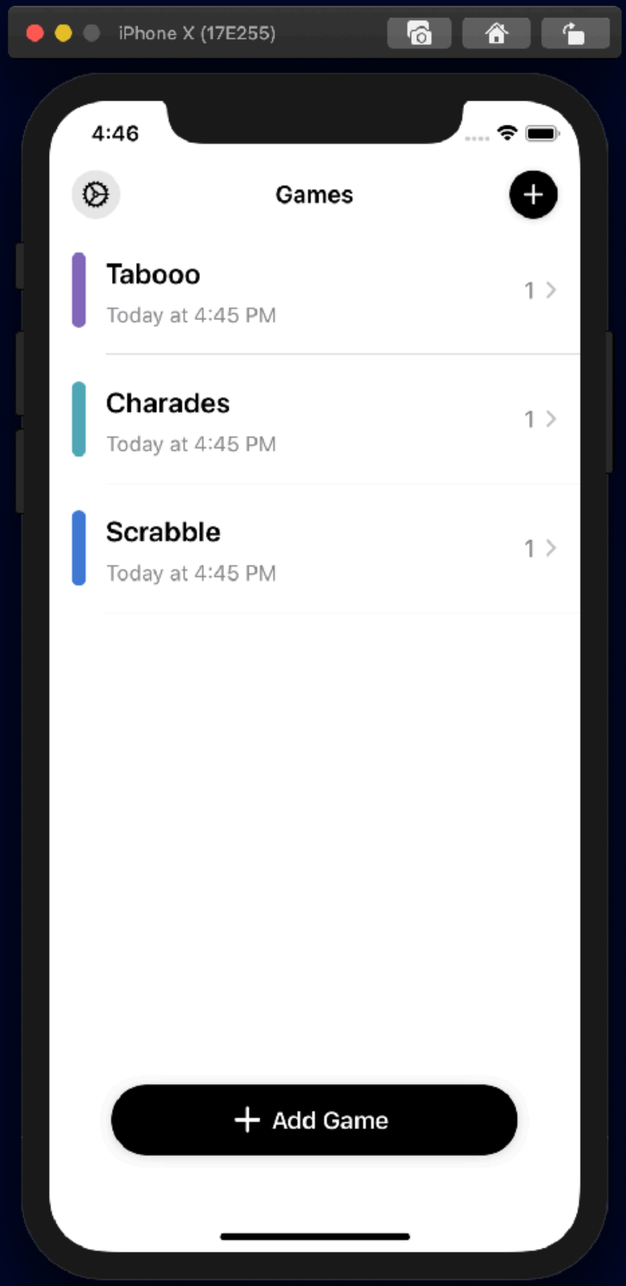
How?

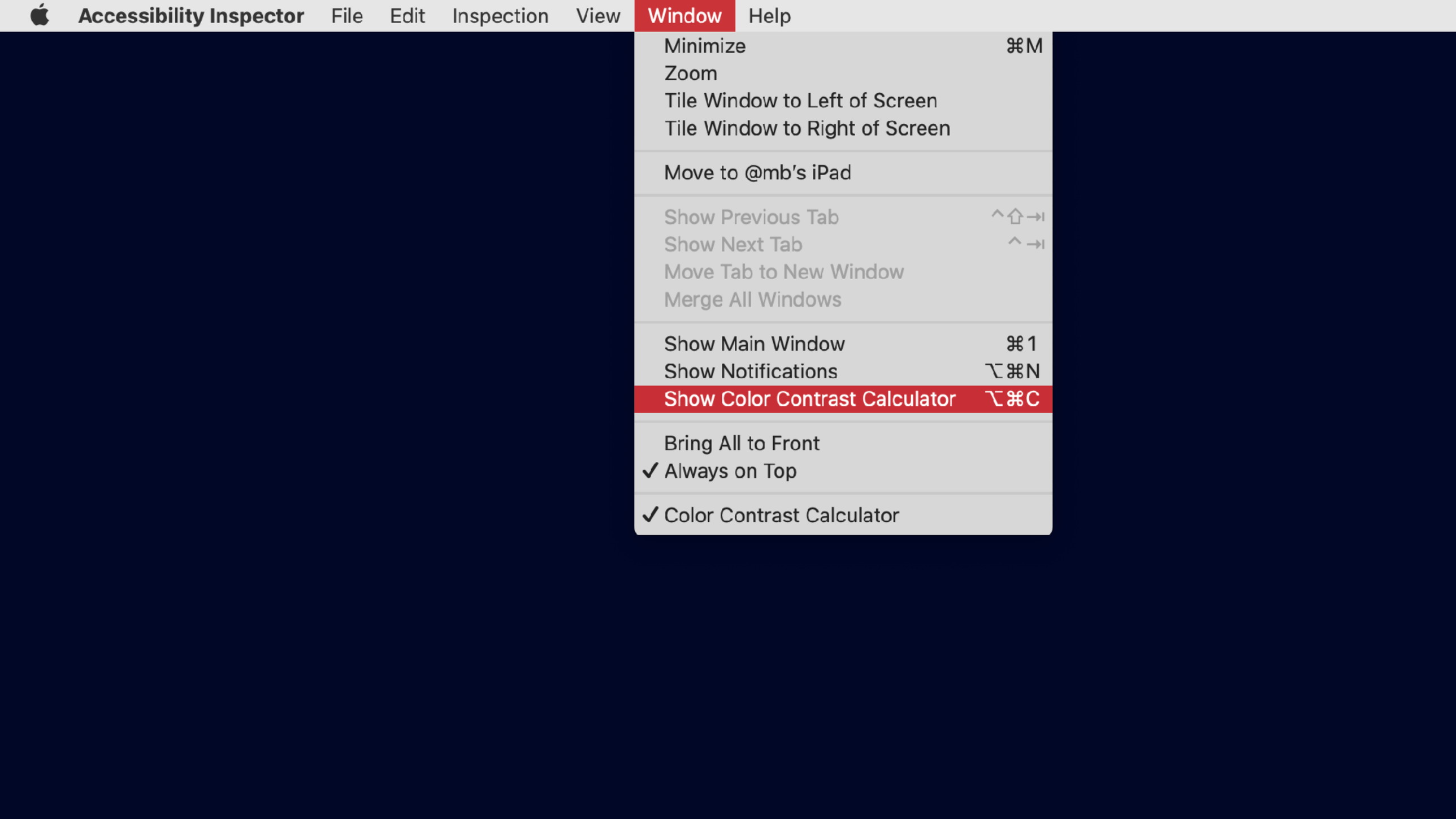
1. Equitable use
2. Flexibility in use
3. Simple and intuitive
4. Perceptible information
5. Tolerance for error
6. Low physical effort
7. Size and space for approach & use

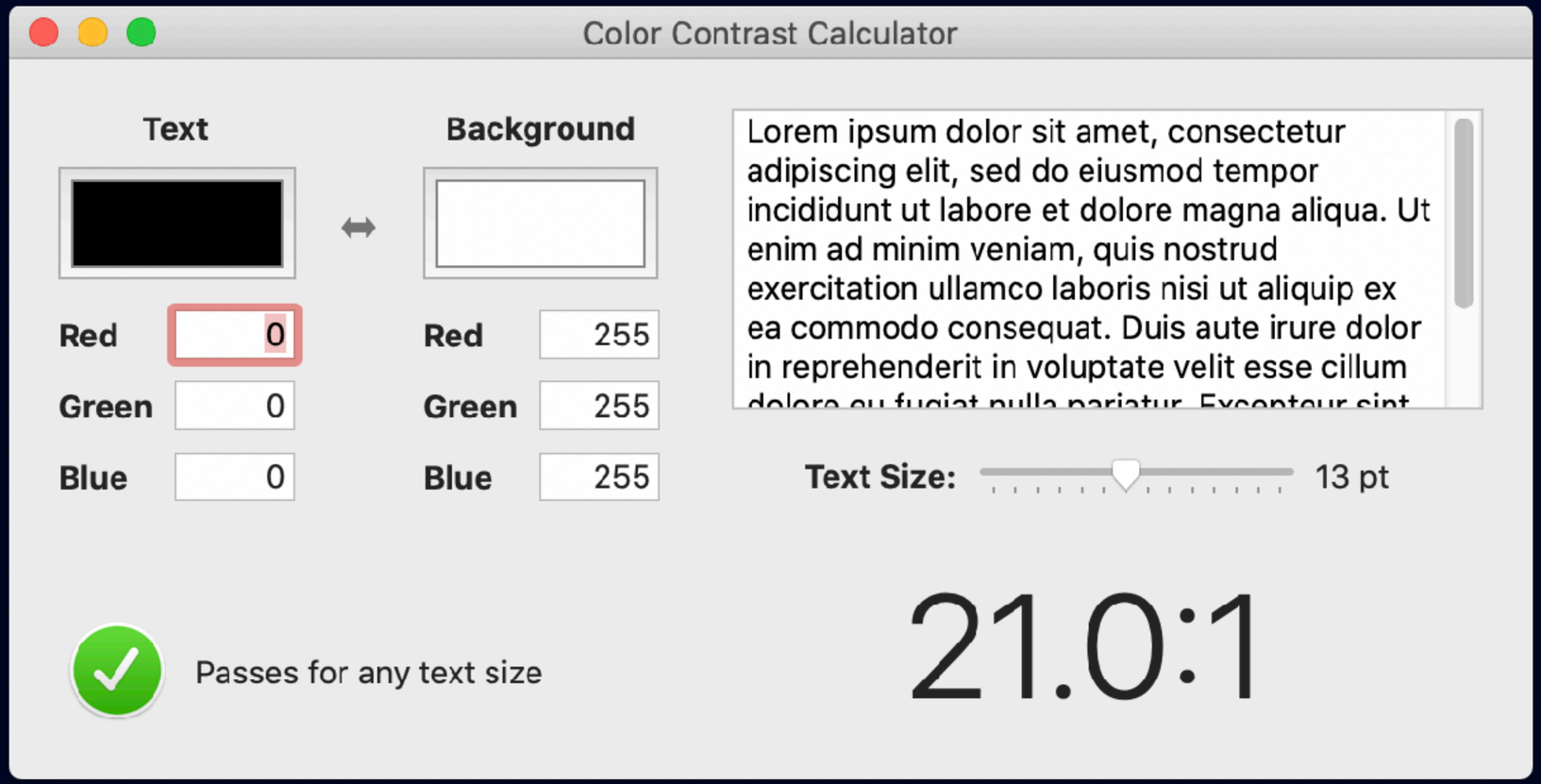
Accessibility Audit


Open Developer Tool
Options
Show All Windows
Hide
Quit
Instruments
Simulator
Accessibility Inspector
FileMerge
Create ML
Reality Composer
KEYNOTE
XCODE

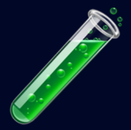

Accessibility Inspector
Simulator - iPhone X (iOS 13.4)
Run Audit
Contrast nearly passed
Contrast failed
Contrast failed
Contrast failed
Contrast failed
Hit area is too small
The size of this Scorecard.CircularButton is too small for user to interact.
current size is 34 x 34
Hit area is too small
Potentially inaccessible text
Potentially inaccessible text
Potentially inaccessible text
Potentially inaccessible text
Potentially inaccessible text
Potentially inaccessible text
Dynamic Text font sizes are unsupported
Dynamic Text font sizes are unsupported
Dynamic Text font sizes are unsupported
Dynamic Text font sizes are unsupported
Dynamic Text font sizes are unsupported
Dynamic Text font sizes are unsupported
Dynamic Text font sizes are unsupported
Dynamic Text font sizes are unsupported
Dynamic Text font sizes are unsupported
Dynamic Text font sizes are unsupported
Dynamic Text font sizes are unsupported
Dynamic Text font sizes are unsupported
Clear Warnings
26 warnings


iPhone X (17E255)
4:46
Games
Tabooo
Today at 4:45 PM
1
Charades
Today at 4:45 PM
1
Scrabble
Today at 4:45 PM
1
Add Game


Accessibility Inspector
File
Edit
Inspection
View
Window
Help
Minimize ⌘M
Zoom
Tile Window to Left of Screen
Tile Window to Right of Screen
Move to @mb's iPad
Show Previous Tab ^⇧→|
Show Next Tab ^→|
Move Tab to New Window
Merge All Windows
Show Main Window ⌘1
Show Notifications ⌥⌘N
Show Color Contrast Calculator ⌥⌘C
Bring All to Front
✓ Always on Top
✓ Color Contrast Calculator


Color Contrast Calculator
Text
Background
Red
0
Green
0
Blue
0
Red
255
Green
255
Blue
255
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum
Text Size:
13 pt
Passes for any text size
21.0:1

Audit one screen at a time

 Test each accessibility feature

Ensure proper contrast, size, & labeling

Use system controls wherever possible

Localize your accessibility labels

Test with users of assistive technology

Axes of Access

Vision

Hearing

Physical & Motor

Literacy & Learning

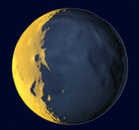
Locality

Inclusion

Vision

VoiceOver

Dynamic Type

 Smart Invert

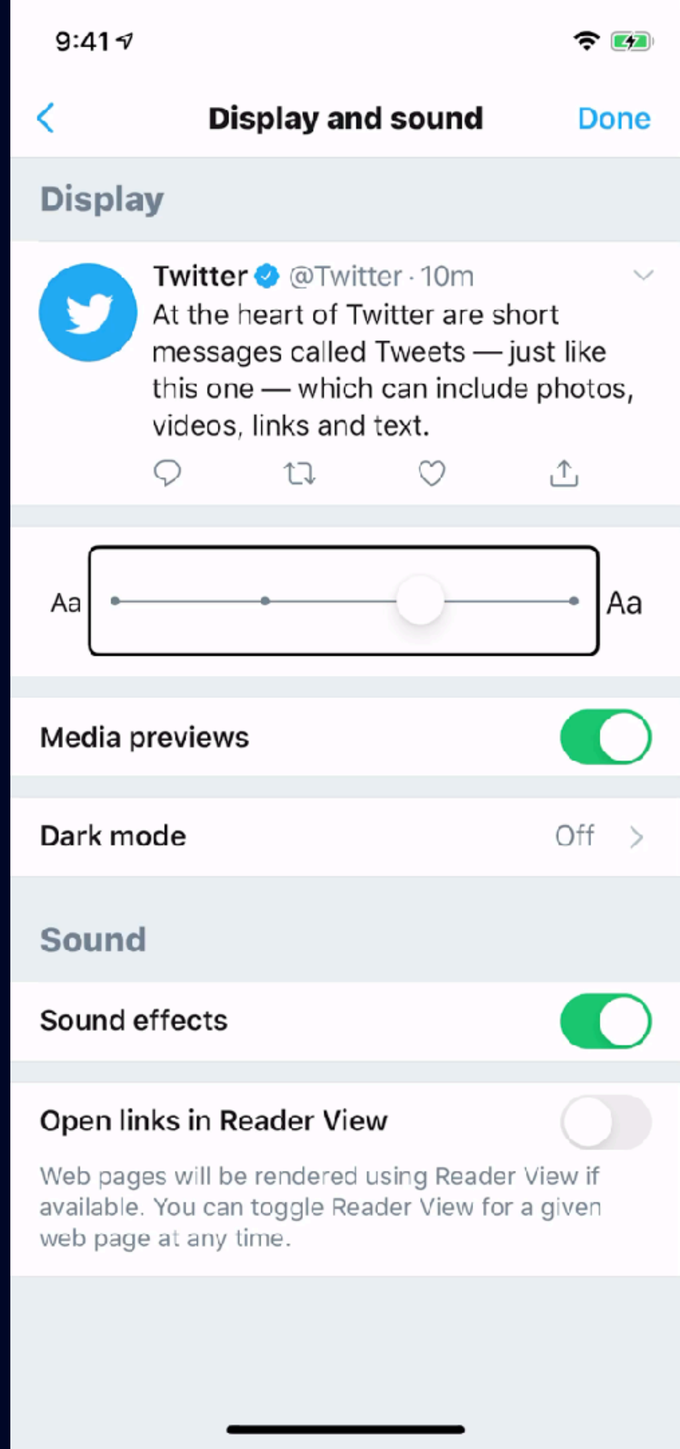
Differentiate Without Color

Zoom & Magnifier

Reduce Motion

🎧 VoiceOver

```
let slider = UISlider()

/// A localized string that succinctly identifies the accessibility element.
slider.accessibilityLabel = "Text Size Slider"

let percent = NumberFormatter.localizedString(from: 0.67 as
NSNumber, number: .percent)

/// A localized string that represents the current value of the accessibility element.
slider.accessibilityValue = percent

/// A trait describes a single aspect of an element's behavior, state, or usage.
slider.accessibilityTraits = .adjustable

/// A brief description of the result of performing an action on the accessibility
element.
slider.accessibilityHint = "Swipe up or down with one finger
to adjust the value."
```



Dynamic Type

```
let label = UILabel()

/// Indicates whether the object automatically updates its font when the device's content size category changes.
label.adjustsFontForContentSizeCategory = true

/// Returns an instance of the system font for the specified text style, scaled for the user's selected content size category.
let font = UIFont.preferredFont(forTextStyle: .headline)
label.font = font

// MARK: - Custom Fonts

let fontMetrics = UIFontMetrics(forTextStyle: .headline)
let customFont = UIFont(name: "Comic Sans", size: 42)!

/// Returns a version of the specified font that adopts the current font metrics.
let scaledFont = fontMetrics.scaledFont(for: customFont)
label.font = scaledFont
```

Smart Invert

```
let legsImageView = UIImageView()

/// Indicates whether the view ignores an accessibility request to invert its colors.
legsImageView.accessibilityIgnoresInvertColors = true


/// Returns whether the system preference for invert colors is enabled.
UIAccessibility.isInvertColorsEnabled

/// Posted by UIKit when the setting for inverted colors has changed.
UIAccessibility.invertColorsStatusDidChangeNotification
```

Differentiate Without Color

```
let statusView = UIImageView()

statusView.backgroundColor = shouldGo ? .green : .red

/// Returns whether or not the system preference for Differentiate Without Color is enabled.
if UIAccessibility.shouldDifferentiateWithoutColor {
    statusView.image = shouldGo ? goImage : stopImage
}
```

```
/// Posted by UIKit when the system's Differentiate Without Color Setting has changed.
UIAccessibility.differentiateWithoutColorDidChangeNotification
```

🔎 Zoom

```
/// Warns users that application-specific gestures conflict with the system-defined Zoom accessibility gestures.
UIAccessibility.registerGestureConflictWithZoom()
```

```
/// Notifies the system that the app's focus has changed to a new location.
UIAccessibility.zoomFocusChanged(
    zoomType: .insertionPoint,
    toFrame: replyTextViewFrame,
    in: textView
)
```

Reduce Motion & Transparency

```
/// Returns a Boolean value indicating whether Reduce Motion is enabled.
if UIAccessibility.isReduceMotionEnabled {
    likeButton.displayLike(animated: false)
} else {
    likeButton.displayLike(animated: true)
}

/// Posted by UIKit when the system's Reduce Motion setting has changed.
UIAccessibility.reduceMotionStatusDidChangeNotification
```

Audio Descriptions

Bold Text

Button Shapes

On/Off Labels

Increase Contrast

Color Filters

Reduce White Point

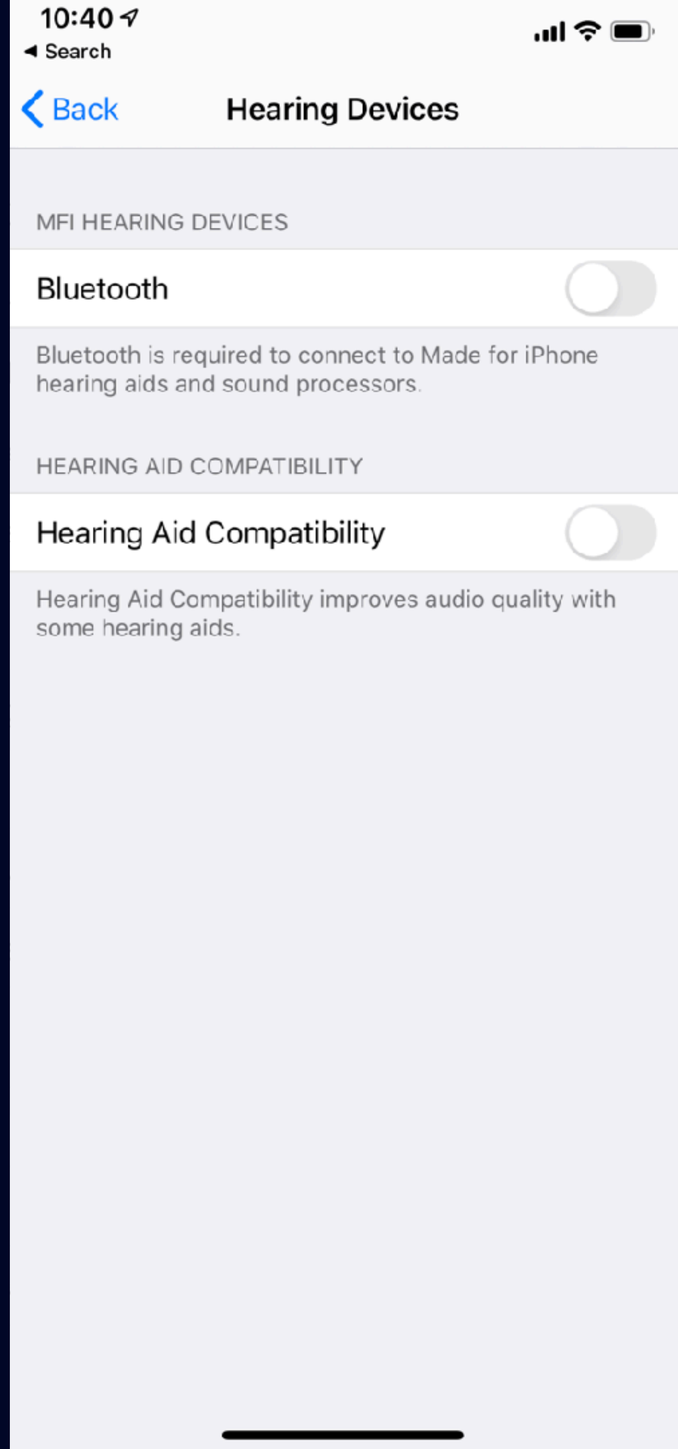
Hearing

Hearing Devices

Subtitles & Captioning

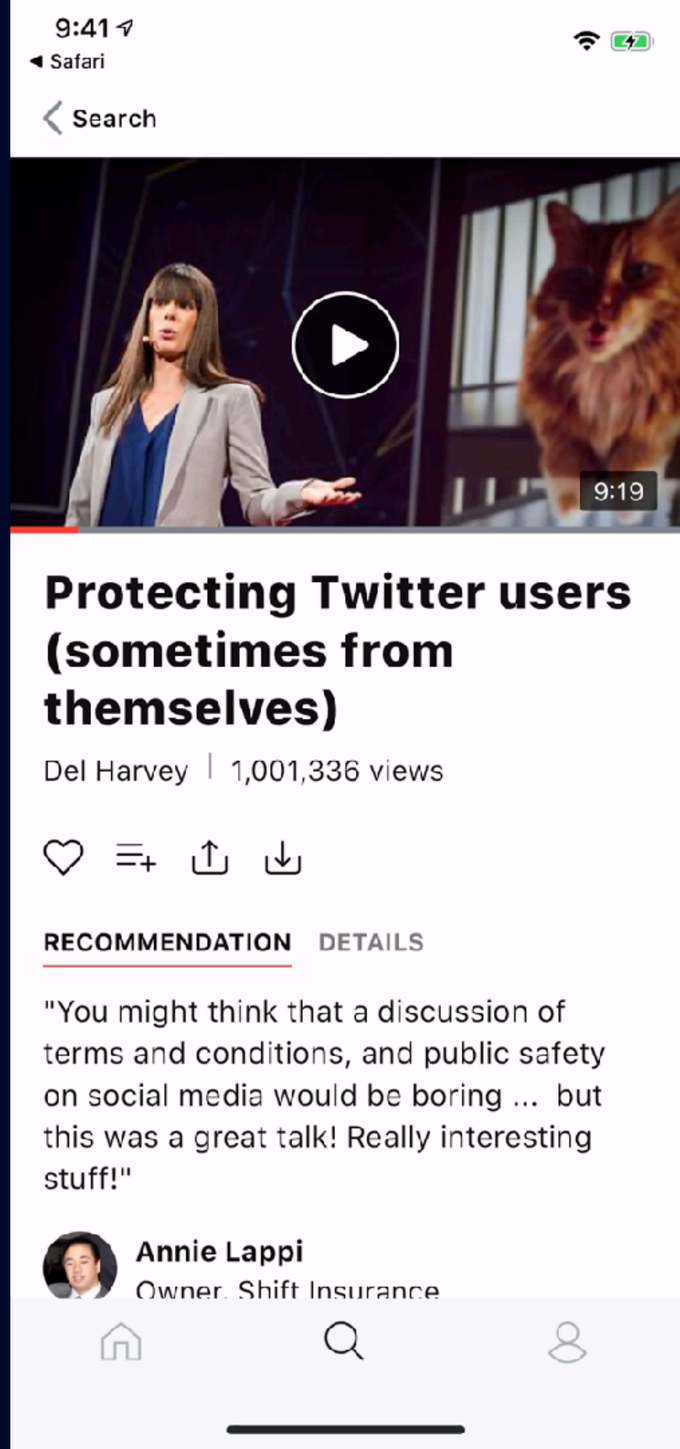
🦻 Hearing Devices

```
import AVKit

let session = AVAudioSession.sharedInstance()

/// Apple supports the use of Bluetooth Low Energy (LE) hearing aids. Apps don't have control over routing to these devices. Instead, the system automatically decides when routing to Bluetooth LE is appropriate.

var isRoutingToHearingAid: Bool {
    return session.currentRoute.outputs.contains {
        $0.portType == .bluetoothLE
    }
}
```



Subtitles & Captioning

```
import AVKit

let playerViewController = AVPlayerViewController()

/// Indicates whether the player view controller shows playback controls.
playerViewController.showsPlaybackControls = true

/// Starting with iOS 7.0, AVPlayer provides automatic media selection based on the user's system preferences as its default behavior. To override the default criteria for any media selection group, use `setMediaSelectionCriteria(_:forMediaCharacteristic:)`.

playerViewController.player?.appliesMediaSelectionCriteriaAutomatically = true
```



Physical & Motor

Switch Control

Voice Control

Full Keyboard Access

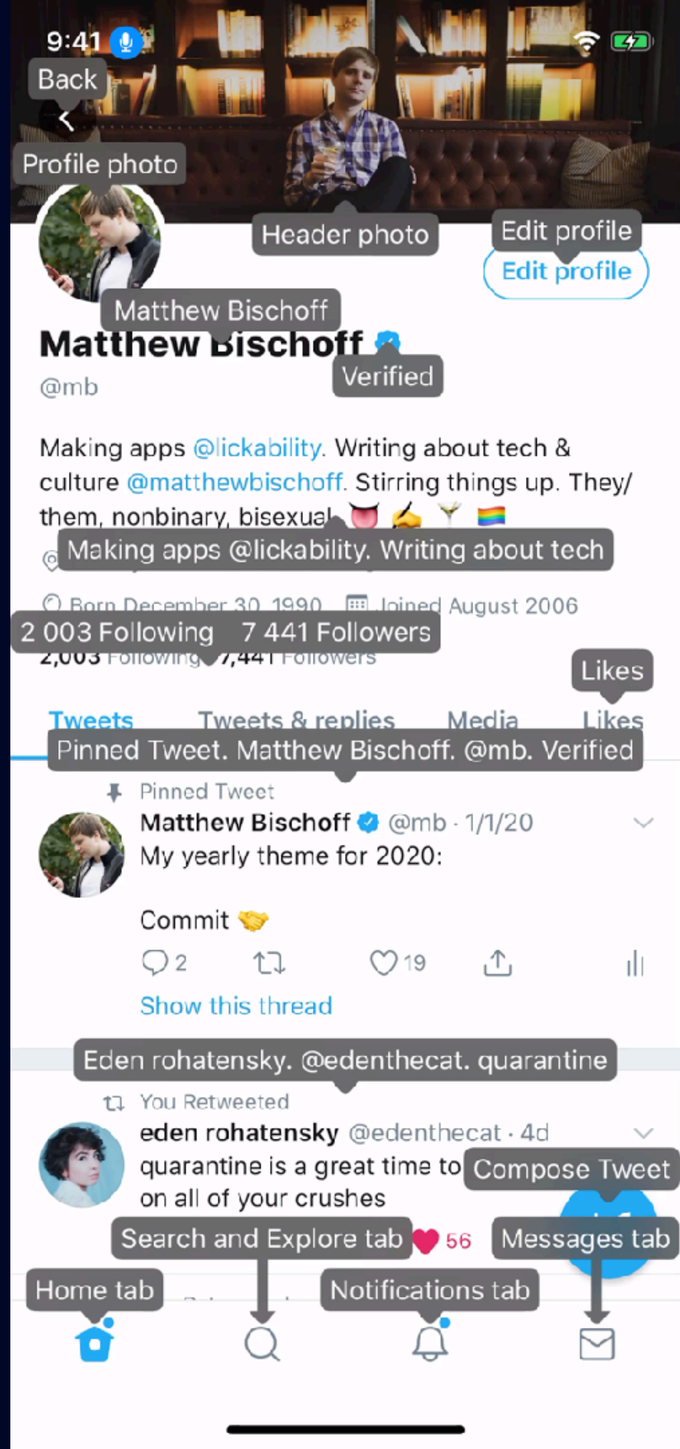
Assistive Touch

🎛 Switch Control

```
class RetweetControl: UIControl {

    /// Tells the element to activate itself and report the success or failure of the operation.
    override func accessibilityActivate() -> Bool {
        sendActions(for: .primaryActionTriggered)

        return true
    }
}

/// Returns a Boolean value indicating whether it is enabled.
UIAccessibility.isSwitchControlRunning

/// Posted by UIKit when the system setting has changed.
UIAccessibility.switchControlStatusDidChangeNotification
```

🎙 Voice Control

- Uses the same labels as VoiceOver
- Learn the commands
  - “Show numbers”
  - “Show names”
  - "Show grid"
- Make sure all actions and gestures in your app can be performed via Voice Control



⌨ Full Keyboard Access

```
protocol UIAccessibilityContainer {
    /// An array of the accessibility elements in the container.
    var accessibilityElements: [Any]? { get set }

    /// Returns the accessibility element at the specified index.
    func accessibilityElement(at: Int) -> Any?

    /// Returns the index of the specified accessibility element.
    func index(ofAccessibilityElement: Any) -> Int
}
```

Literacy & Learning

📣 Speak Selection

Safari Reader

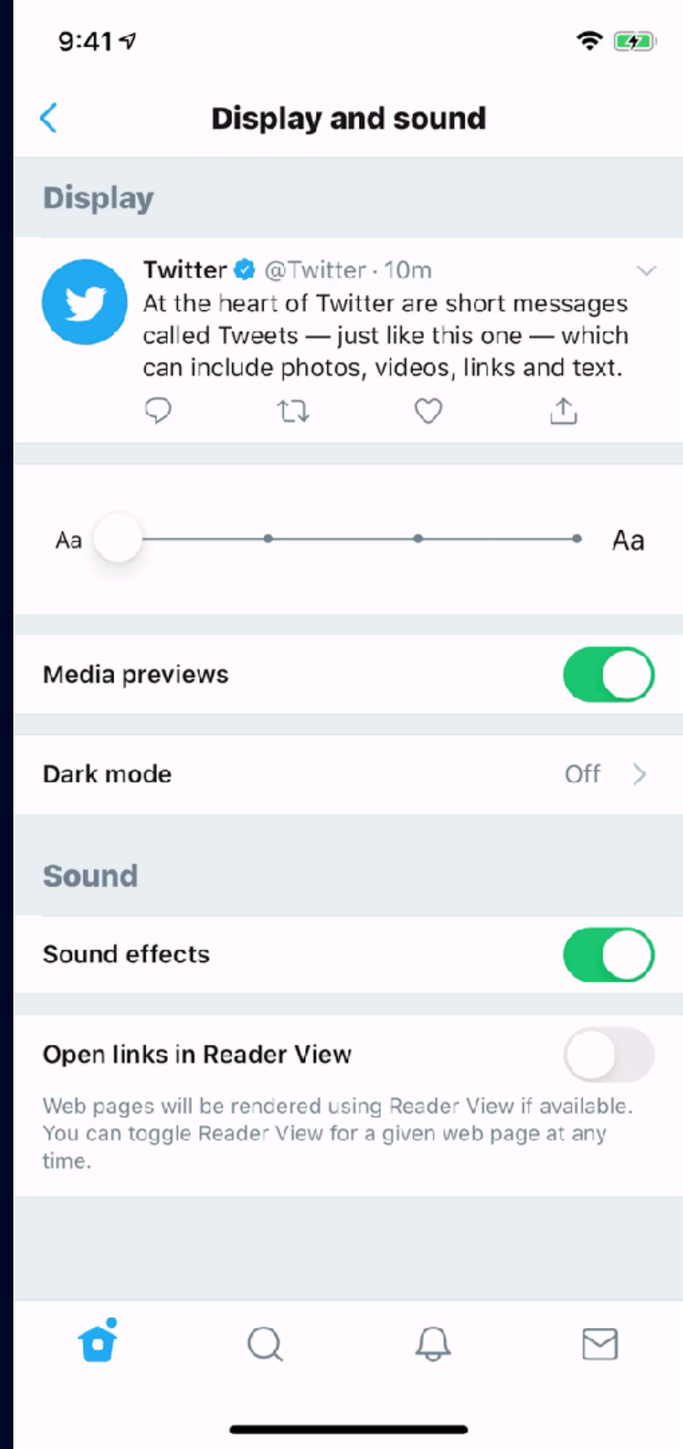
⌨️ Typing Feedback

📣 Speak Selection

```
let tweetTextView = UITextView()

/// Controls the ability of the user to select content and interact with URLs and attachments.
tweetTextView.isSelectable = true


/// Indicates whether speaking the selection is enabled.
UIAccessibility.isSpeakSelectionEnabled

/// Posted when the system's Speak Selection setting has changed.
UIAccessibility.speakSelectionStatusDidChangeNotification


/// Indicates whether speaking the screen is enabled.
UIAccessibility.isSpeakScreenEnabled

/// Posted when the system's Speak Screen setting has changed.
UIAccessibility.speakScreenStatusDidChangeNotification
```

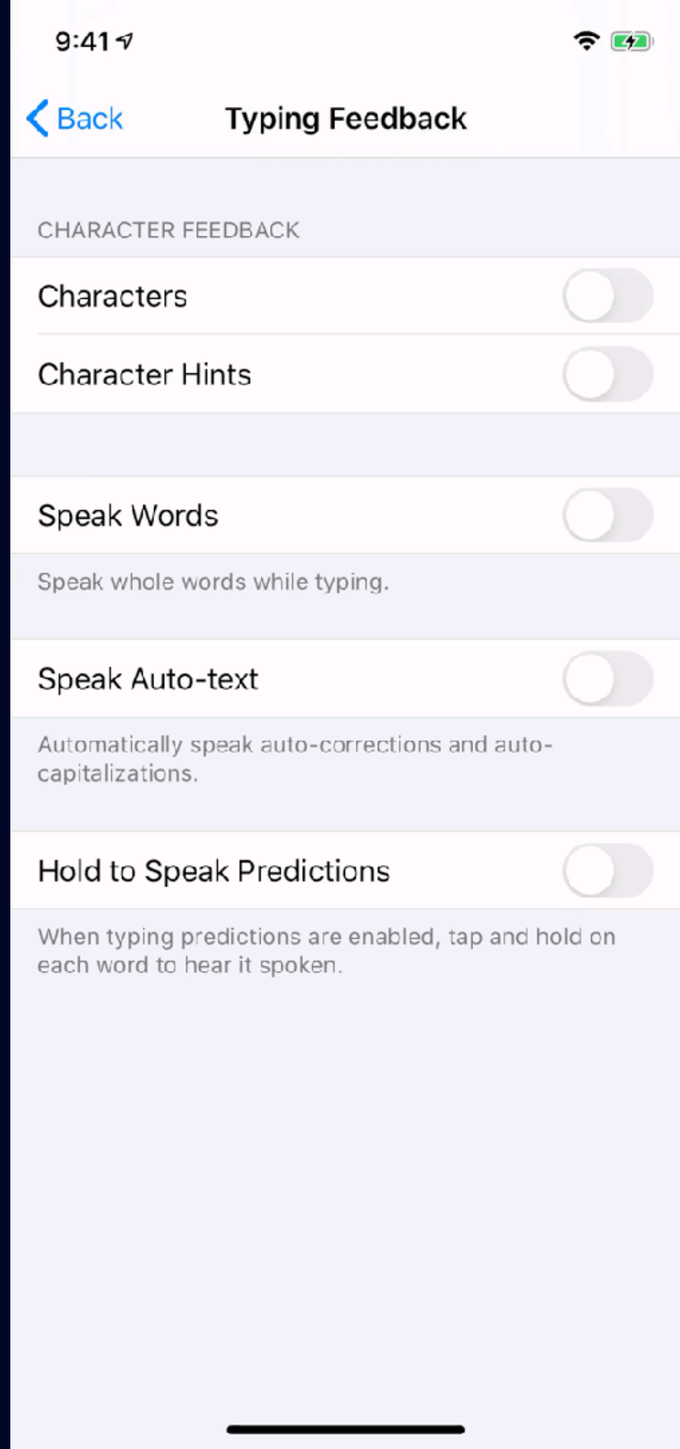
Safari Reader

```
let configuration = SFSafariViewController.Configuration()

/// A value that specifies whether Safari should enter Reader mode.
configuration.entersReaderIfAvailable = true

let browser = SFSafariViewController(url: url,
configuration: configuration)

present(browser, animated: true)
```



⌨ Typing Feedback

```
class TweetReplyTextView: UITextView {
    let keyboardViewController = KeyboardViewController()

    override var inputViewController: UIInputViewController? {
        return keyboardViewController
    }
}

class KeyboardViewController: UIInputViewController {
    override func viewDidLoad() {
        super.viewDidLoad()

        inputView = UIInputView(frame: frame, inputViewStyle: .keyboard)

        let tButton = UIButton()
        tButton.addTarget(self, action: #selector(tButtonTapped),
for: .primaryActionTriggered)
        inputView?.addSubview(jButton)
    }

    @objc func tButtonTapped() {
        textDocumentProxy.insertText("T")
    }
}
```



Locality

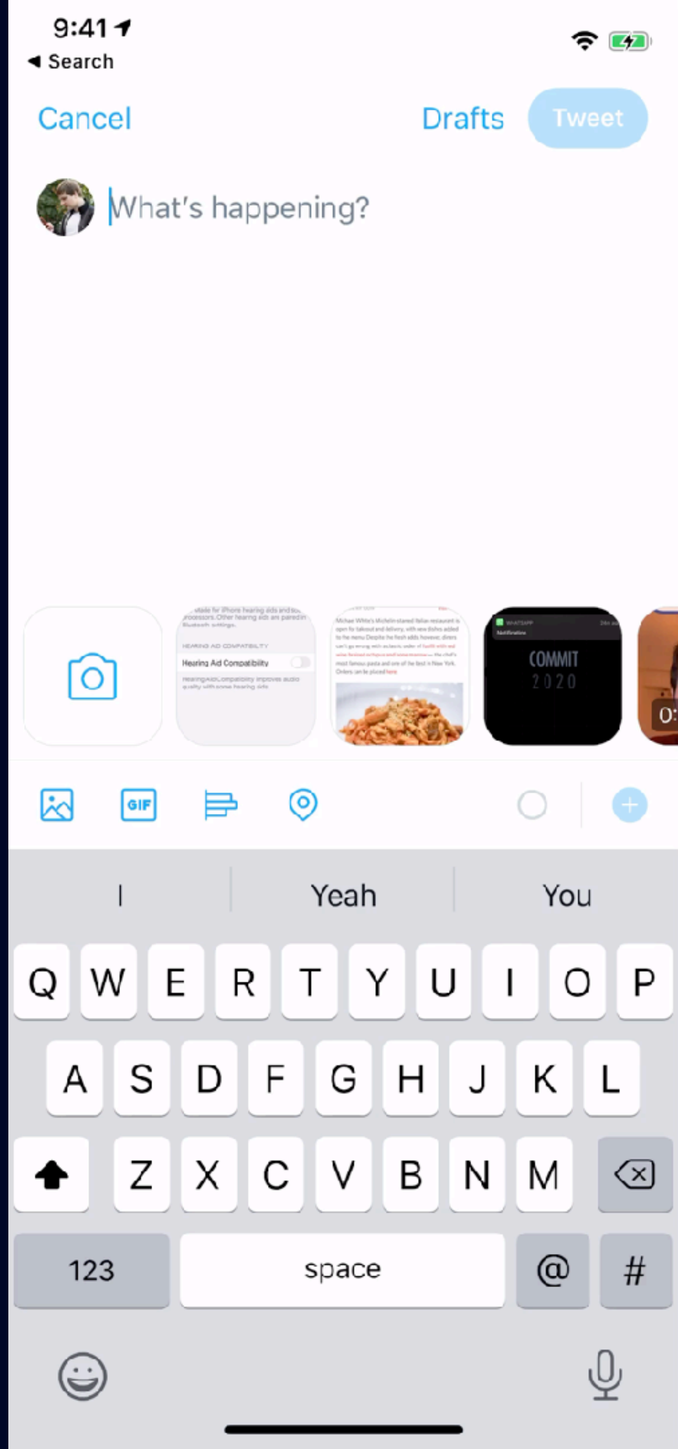
Localized Strings

🔢 Localized Formats

Localized Strings

/// Xcode can read through a project's code to find invocations of NSLocalizedString() and automatically generate the appropriate strings files for the project's base localization.

```
let placeholder = NSLocalizedString("What's
happening?", comment: "Compose placeholder text.")

textView.placeholder = placeholder
```



🔢 Localized Formats

```swift
let date = Date()
let likeCount = 2401 as NSNumber

/// Returns a string representation of a given date, formatted for the current locale using the specified date and time styles.
DateFormatter.localizedString(
    from: date, dateStyle: .medium, timeStyle: .short
)


/// Returns a localized number string with the specified style.
NumberFormatter.localizedString(
    from: decimal, number: .decimal
)
```

Inclusion

Names

🏳️‍🌈 Gender & Sexuality

Race & Ethnicity

Names

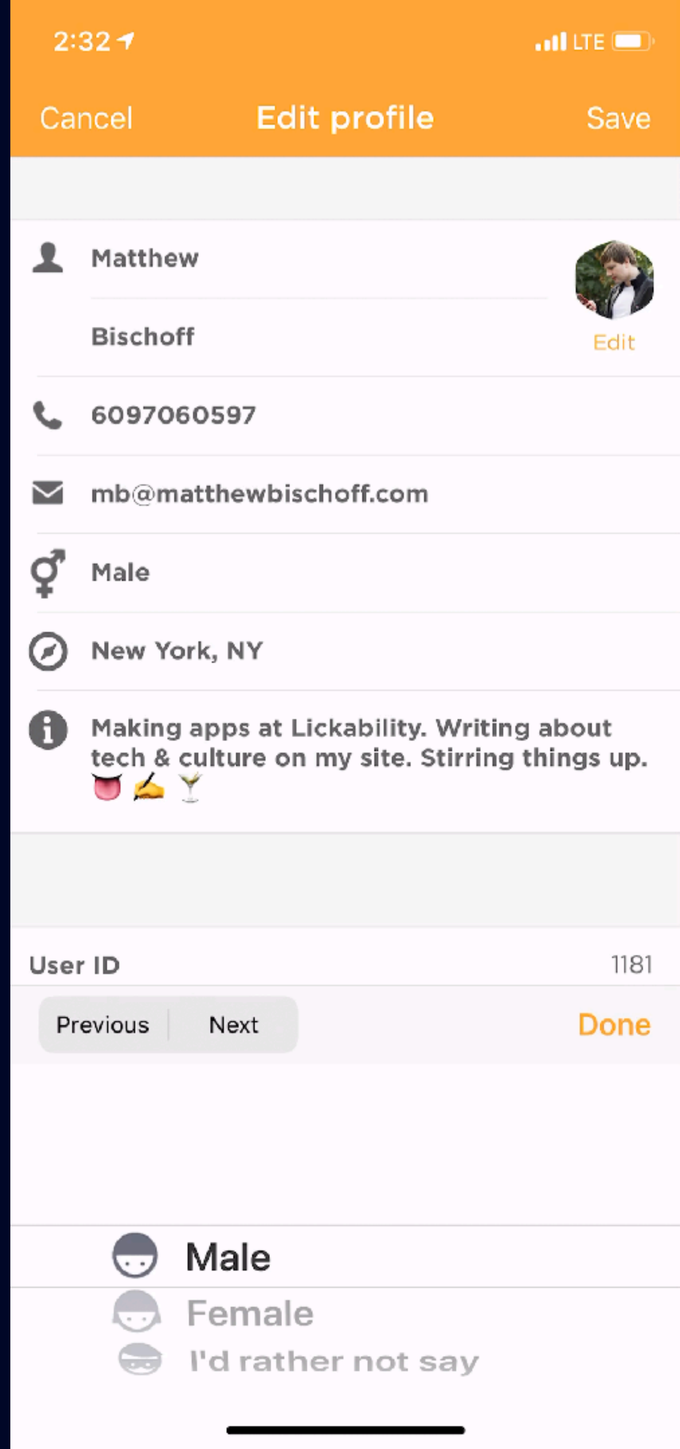
```swift
var components = PersonNameComponents()

components.namePrefix = "Mx." // Gender-neutral title
components.givenName = "Matthew"
components.familyName = "Bischoff"
components.nickname = "Matt"

/// Prints “Matthew Bischoff” in US English.
PersonNameComponentsFormatter.localizedString(from: components, style: .default)

/// Prints “Matt” in US English.
PersonNameComponentsFormatter.localizedString(from: components, style: .short)

/// Prints “MB” in US English.
PersonNameComponentsFormatter.localizedString(from: components, style: .abbreviated)
```

1. People have exactly one canonical full name.
2. People have exactly one full name which they go by.
3. People have, at this point in time, exactly one canonical full name.
4. People have, at this point in time, one full name which they go by.
5. People have exactly N names, for any value of N.
6. People's names fit within a certain defined amount of space.
7. People's names do not change.
8. People's names change, but only at a certain enumerated set of events.
9. People's names are written in ASCII.
10. People's names are written in any single character set.
11. People's names are all mapped in Unicode code points.
12. People's names are case sensitive.
13. People's names are case insensitive.
14. People's names sometimes have prefixes or suffixes, but you can safely ignore those.
15. People's names do not contain numbers.
16. People's names are not written in ALL CAPS.
17. People's names are not written in all lower case letters.
18. People's names have an order to them. Picking any ordering scheme will automatically result in consistent ordering among all systems, as long as both use the same ordering scheme for the same name.
19. People's first names and last names are, by necessity, different.
20. People have last names, family names, or anything else which is shared by folks recognized as their relatives.
21. People's names are globally unique.
22. People's names are almost globally unique.
23. Alright alright but surely people's names are diverse enough such that no million people share the same name.
24. My system will never have to deal with names from China.
25. Or Japan.
26. Or Korea.
27. Or Ireland, the United Kingdom, the United States, Spain, Mexico, Brazil, Peru, Russia, Sweden, Botswana, South Africa, Trinidad, Haiti, France, or the Klingon Empire, all of which have "weird" naming schemes in common use.
28. That Klingon Empire thing was a joke, right?
29. Confound your cultural relativism! People in my society, at least, agree on one commonly accepted standard for names.
30. There exists an algorithm which transforms names and can be reversed losslessly. (Yes, yes, you can do it if your algorithm returns the input. You get a gold star.)
31. I can safely assume that this dictionary of bad words contains no people's names in it.
32. People's names are assigned at birth.
33. OK, maybe not at birth, but at least pretty close to birth.
34. Alright, alright, within a year or so of birth.
35. Five years?
36. You're kidding me, right?
37. Two different systems containing data about the same person will use the same name for that person.
38. Two different data entry operators, given a person's name, will by necessity enter bitwise equivalent strings on any single system, if the system is well-designed.
39. People whose names break my system are weird outliers. They should have had solid, acceptable names, like 田中太郎.
40. People have names.

Falsehoods Programmers Believe About Names by Patrick McKenzie

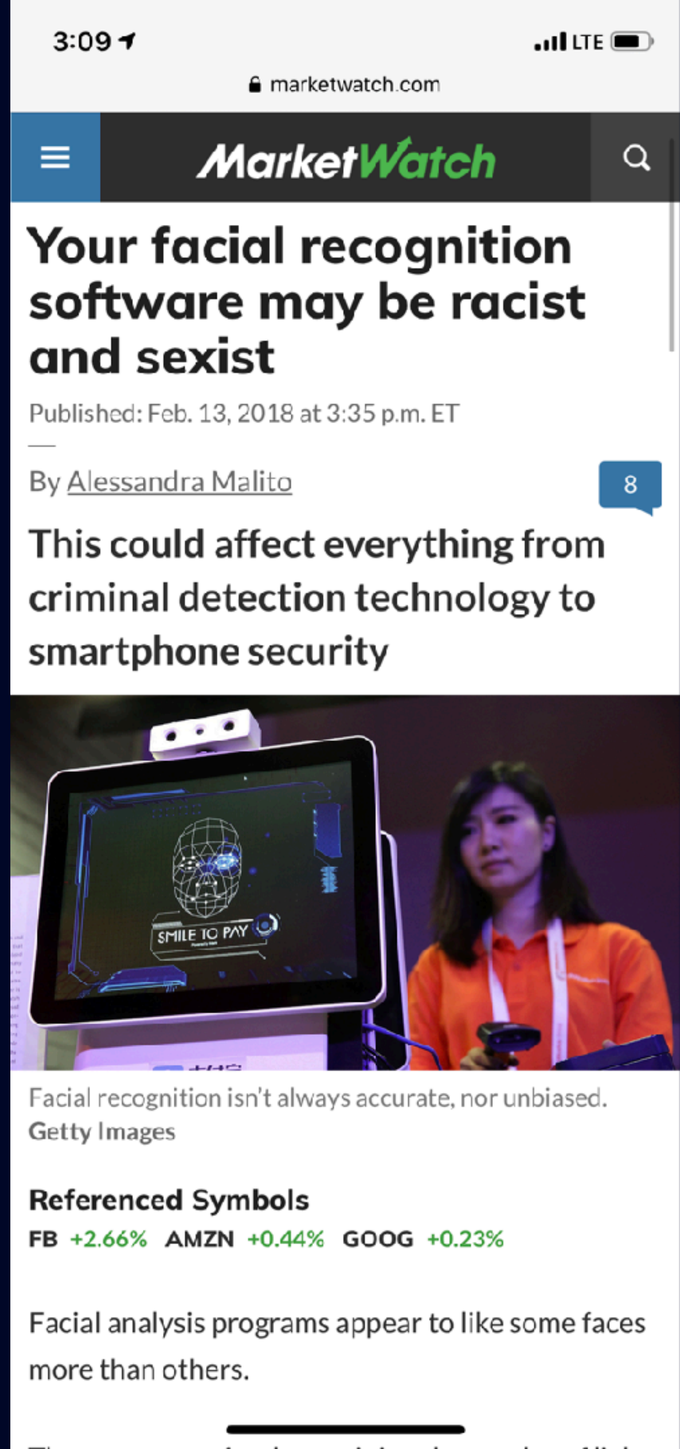
🏳️‍🌈 Gender and Sexuality

- Don't ask for gender if you don't need it
- Allow typing a gender, selecting no gender, and multiple genders
- Don't marginalize folks as "other" or "prefer not to say"
- Give people a place to put their pronouns
- Don't assume people's sexualities
- Let people self-identify



👩🏽‍⚕️ Race & Ethnicity

- Build a diverse team of designers, engineers, and managers
- Recognize algorithms have biases
- Test with folks of multiple races and ethnicities
- Own and fix your issues



APPS ARE FOR EVERYONE

GOOD DESIGN IS UNIVERSAL

ACCESSIBILITY IS OUR JOB

THANK YOU

@mb • matthewbishoff.com/apps-for-all • lickability.com