#WWDC19

Sign In with Apple

Gokul Thirumalai, Engineering Manager
Dima Belov, iOS Engineering Manager
Jonathan Birdsall, iOS Engineer

Overview

Integrating with your app

Demo

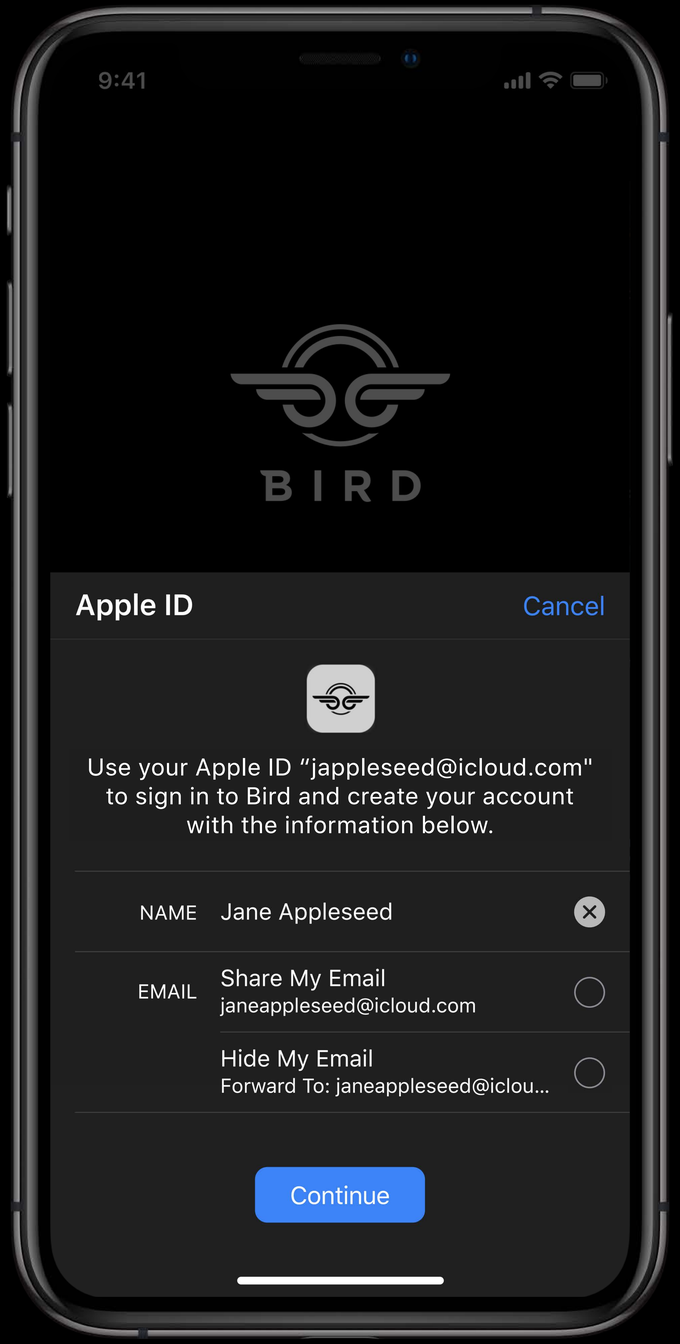
Cross-platform

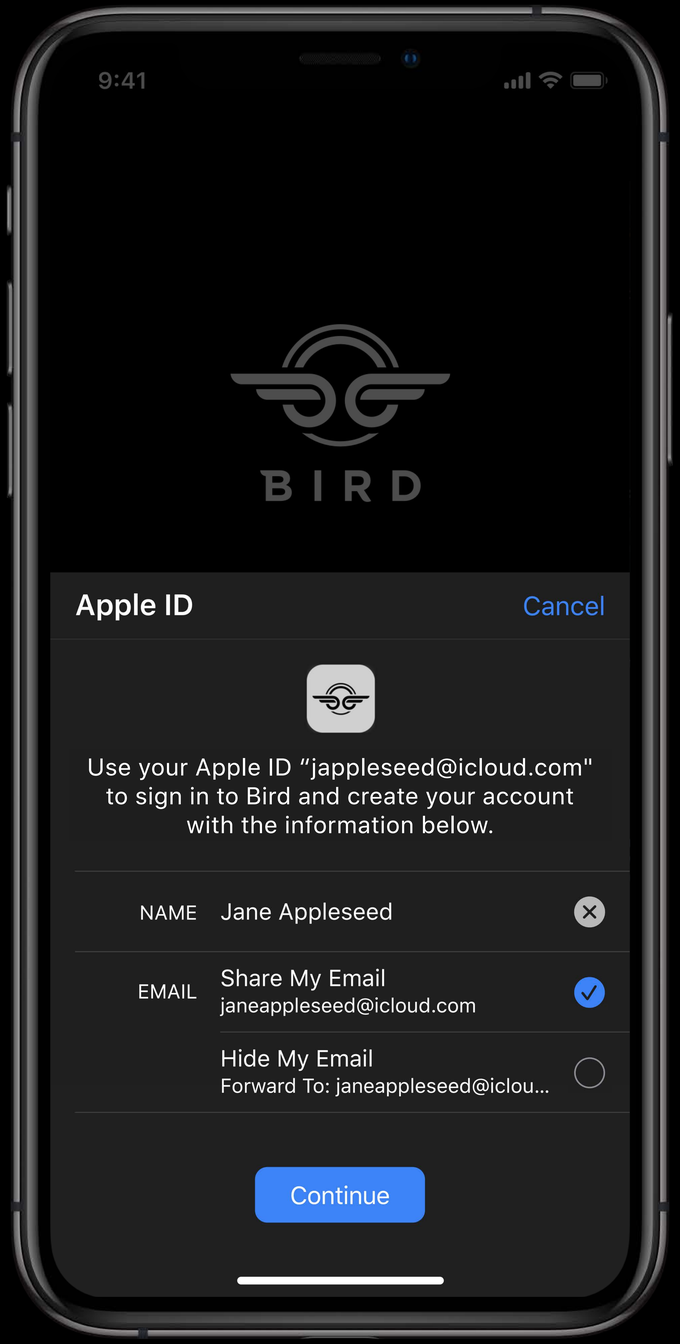
Best practices


Sign in with Apple

Fast, easy account setup and sign in


9:41
BIRD
Sign in with Apple
- or -
SIGN IN WITH EMAIL


9:41
BIRD
Apple ID
Cancel
Use your Apple ID “jappleseed@icloud.com" to sign in to Bird and create your account with the information below.
NAME
Jane Appleseed
EMAIL
Share My Email
janeappleseed@icloud.com
Hide My Email
Forward To: janeappleseed@iclou...
Continue


9:41
BIRD
Apple ID
Cancel
Use your Apple ID “jappleseed@icloud.com" to sign in to Bird and create your account with the information below.
NAME
Jane Appleseed
EMAIL
Share My Email
janeappleseed@icloud.com
Hide My Email
Forward To: janeappleseed@iclou...
Continue

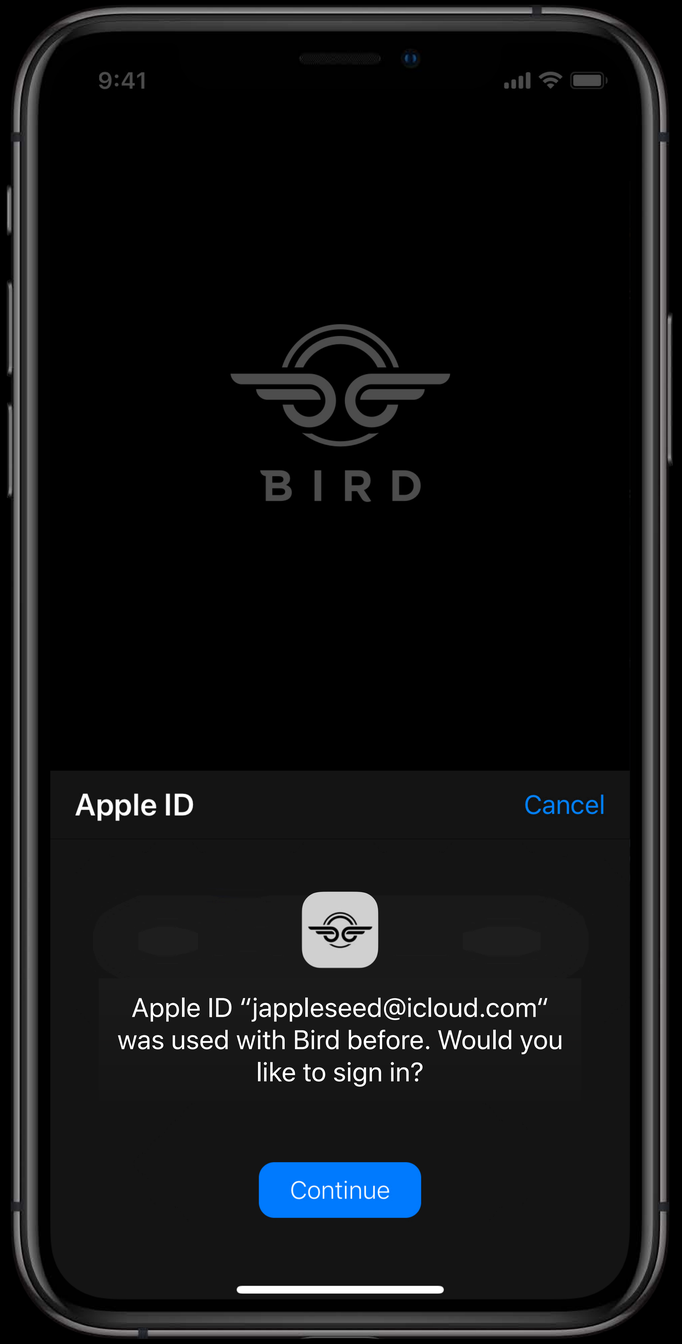
ID

Full name

Verified email address

Seamless
across devices

Seamless
across devices



Sign In with Apple

Streamlined account setup

Sign In with Apple

Streamlined account setup

Verified email addresses

dany-4ever@outlook.com

spamvortex@yahoo.com

drogonthegreat1@gmail.com

snowdontknow@yahoo.com

ghostrulez13@gmail.com

r45N934br1@privaterelay.appleid.com

jane.appleseed@icloud.com

Hide My Email

Linked to verified email

Two-way relay

Any email communication

Apple does not retain messages

Sign In with Apple

Streamlined account setup

Verified email addresses

Sign In with Apple

Streamlined account setup

Verified email addresses

Built-in security

Sign In with Apple

Streamlined account setup

Verified email addresses

Built-in security

Anti-fraud

Anti-Fraud

Privacy friendly

On-device intelligence

Account information

Abstracted to a single bit

Sign In with Apple

Streamlined account setup

Verified email addresses

Built-in security

Anti-fraud

Sign In with Apple

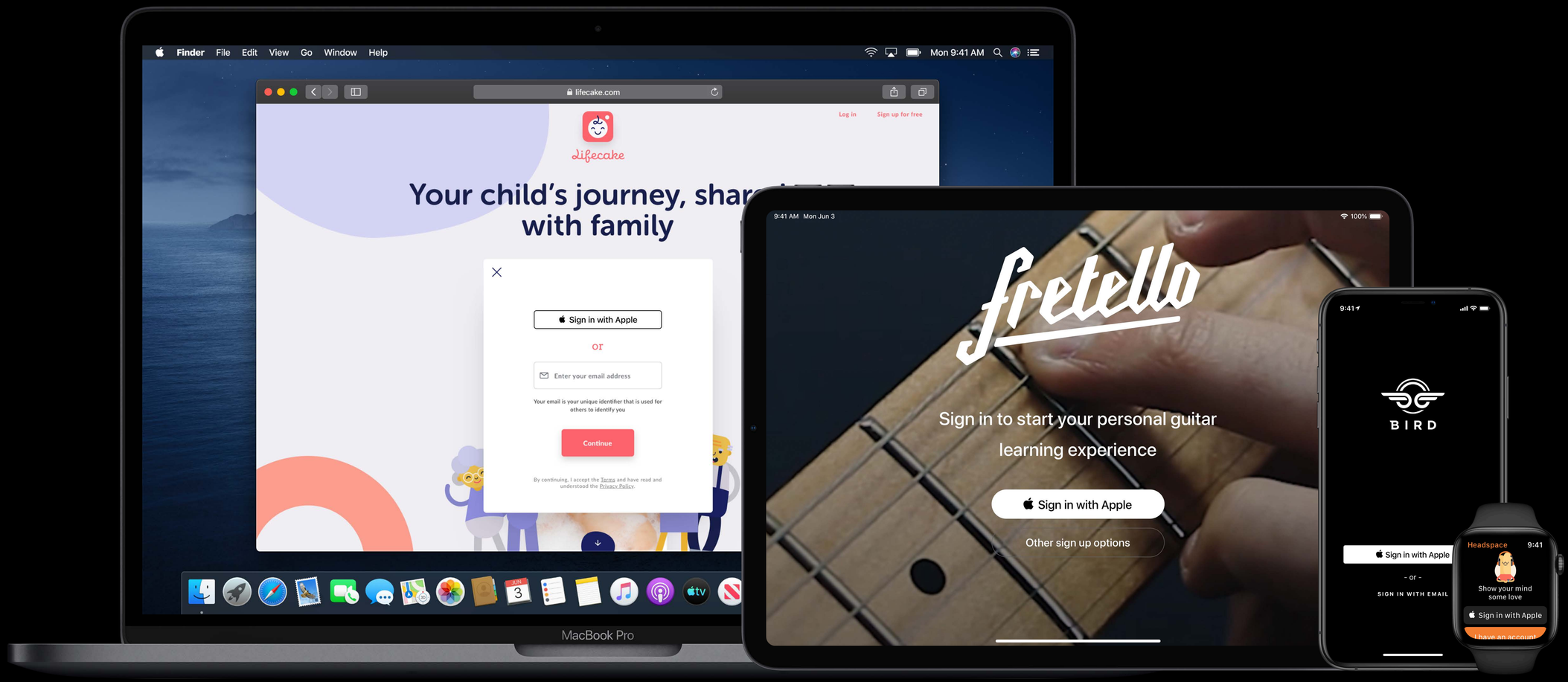
Streamlined account setup

Verified email addresses

Built-in security

Anti-fraud

Cross-platform



iOS macOS watchOS tvOS JavaScript

Integrating with Your App

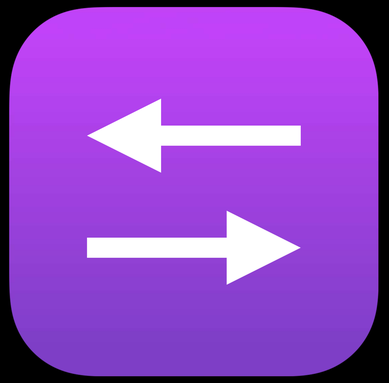
Dima Belov, iOS Engineering Manager

Integrating with Your App

Button

Authorization

Verification



Handling Changes

Integrating with Your App

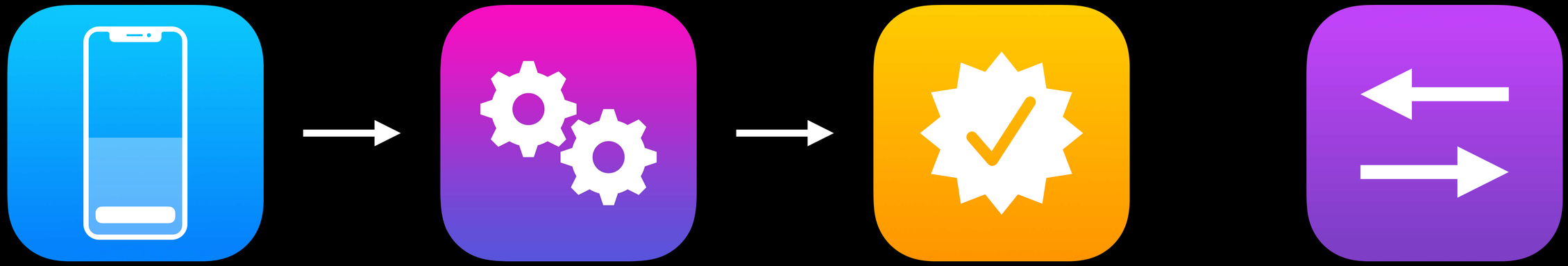
Button

Authorization

Verification

Handling Changes

Integrating with Your App



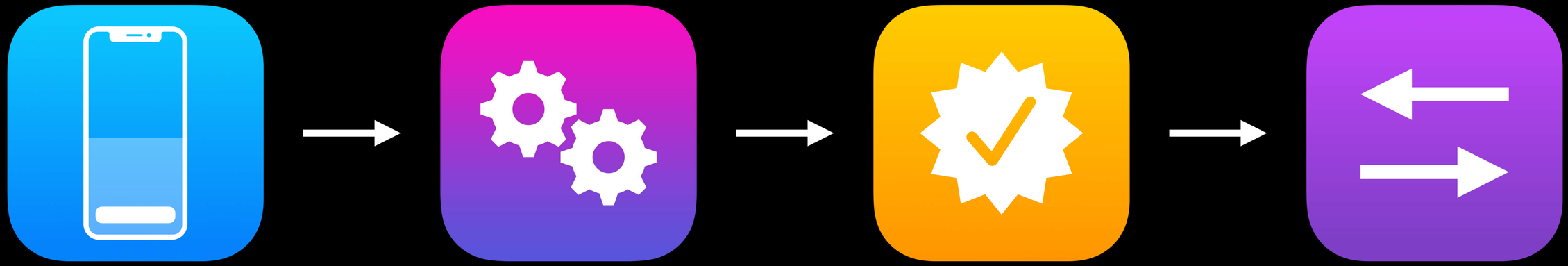
Button

Authorization

Verification

Handling Changes

Integrating with Your App



Button

Authorization

Verification

Handling Changes

Integrating with Your App

```swift
// Add “Sign In with Apple” button to your login view
func setUpProviderLoginView() {
    let button = ASAuthorizationAppleIDButton()
    button.addTarget(self, action: #selector(handleAuthorizationAppleIDButtonPress),
                     for: .touchUpInside)
    self.loginProviderStackView.addArrangedSubview(button)
}
```

```
// Add “Sign In with Apple” button to your login view
func setUpProviderLoginView() {
    let button = ASAuthorizationAppleIDButton()
    button.addTarget(self, action: #selector(handleAuthorizationAppleIDButtonPress),
                     for: .touchUpInside)
    self.loginProviderStackView.addArrangedSubview(button)
}
```

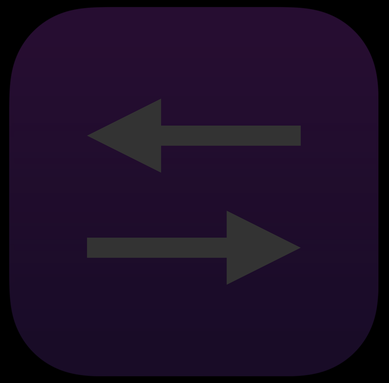
```
// Add “Sign In with Apple” button to your login view
func setUpProviderLoginView() {
    let button = ASAuthorizationAppleIDButton()
    button.addTarget(self, action: #selector(handleAuthorizationAppleIDButtonPress),
                     for: .touchUpInside)
    self.loginProviderStackView.addArrangedSubview(button)
}
```

Integrating with Your App

Button

Authorization

Verification



Handling Changes

Integrating with Your App

Button

Authorization

Verification

Handling Changes

```swift
// Configure request, setup delegates and perform authorization request
@objc func handleAuthorizationButtonPress() {
    let request = ASAuthorizationAppleIDProvider().createRequest()
    request.requestedScopes = [.fullName, .email]

    let controller = ASAuthorizationController(authorizationRequests: [request])

    controller.delegate = self
    controller.presentationContextProvider = self

    controller.performRequests()
}
```

```swift
// Configure request, setup delegates and perform authorization request
@objc func handleAuthorizationButtonPress() {
    let request = ASAuthorizationAppleIDProvider().createRequest()
    request.requestedScopes = [.fullName, .email]

    let controller = ASAuthorizationController(authorizationRequests: [request])

    controller.delegate = self
    controller.presentationContextProvider = self

    controller.performRequests()
}
```

```swift
// Configure request, setup delegates and perform authorization request
@objc func handleAuthorizationButtonPress() {
    let request = ASAuthorizationAppleIDProvider().createRequest()
    request.requestedScopes = [.fullName, .email]

    let controller = ASAuthorizationController(authorizationRequests: [request])

    controller.delegate = self
    controller.presentationContextProvider = self

    controller.performRequests()
}
```

```swift
// Configure request, setup delegates and perform authorization request
@objc func handleAuthorizationButtonPress() {
    let request = ASAuthorizationAppleIDProvider().createRequest()
    request.requestedScopes = [.fullName, .email]

    let controller = ASAuthorizationController(authorizationRequests: [request])

    controller.delegate = self
    controller.presentationContextProvider = self

    controller.performRequests()
}
```

```swift
// Configure request, setup delegates and perform authorization request
@objc func handleAuthorizationButtonPress() {
    let request = ASAuthorizationAppleIDProvider().createRequest()
    request.requestedScopes = [.fullName, .email]

    let controller = ASAuthorizationController(authorizationRequests: [request])

    controller.delegate = self
    controller.presentationContextProvider = self

    controller.performRequests()
}
```

```swift
// Configure request, setup delegates and perform authorization request
@objc func handleAuthorizationButtonPress() {
    let request = ASAuthorizationAppleIDProvider().createRequest()
    request.requestedScopes = [.fullName, .email]

    let controller = ASAuthorizationController(authorizationRequests: [request])

    controller.delegate = self
    controller.presentationContextProvider = self

    controller.performRequests()
}
```

Integrating with Your App

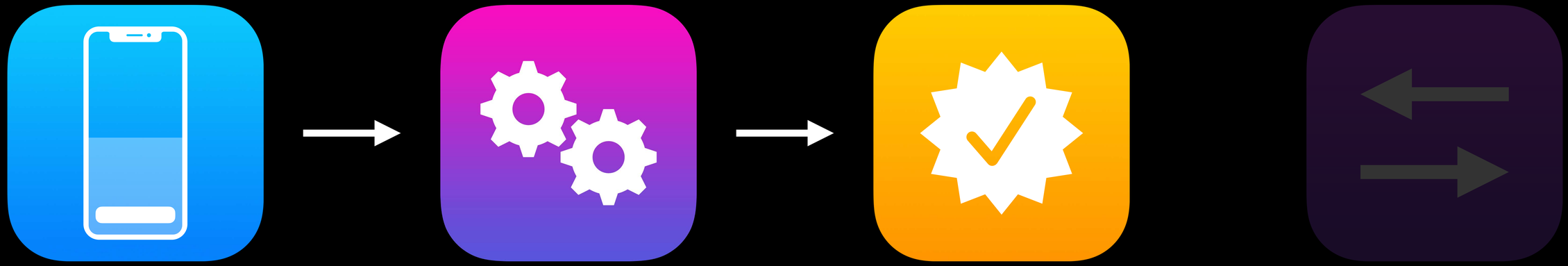
Button

Authorization

Verification

Handling Changes

Integrating with Your App



Button

Authorization

Verification

Handling Changes

```swift
func authorizationController(controller _: ASAuthorizationController,
                             didCompleteWithAuthorization authorization: ASAuthorization) {
    if let credential = authorization.credential as? ASAuthorizationAppleIDCredential {
        let userIdentifier = credential.user
        let identityToken = credential.identityToken
        let authCode = credential.authorizationCode
        let realUserStatus = credential.realUserStatus

        // Create account in your system
    }
}

func authorizationController(_: ASAuthorizationController,
                             didCompleteWithError error: Error) {
    // Handle error
}
```

```swift
func authorizationController(controller _: ASAuthorizationController,
                             didCompleteWithAuthorization authorization: ASAuthorization) {
    if let credential = authorization.credential as? ASAuthorizationAppleIDCredential {
        let userIdentifier = credential.user
        let identityToken = credential.identityToken
        let authCode = credential.authorizationCode
        let realUserStatus = credential.realUserStatus

        // Create account in your system
    }
}

func authorizationController(_: ASAuthorizationController,
                             didCompleteWithError error: Error) {
    // Handle error
}
```

```swift
func authorizationController(controller _: ASAuthorizationController,
                             didCompleteWithAuthorization authorization: ASAuthorization) {
    if let credential = authorization.credential as? ASAuthorizationAppleIDCredential {
        let userIdentifier = credential.user
        let identityToken = credential.identityToken
        let authCode = credential.authorizationCode
        let realUserStatus = credential.realUserStatus

        // Create account in your system
    }
}
```

```swift
func authorizationController(_: ASAuthorizationController,
                             didCompleteWithError error: Error) {
    // Handle error
}
```

Sign In

Response

Sign In

Response

User ID

- Unique, stable, team-scoped user ID

Sign In

Response

User ID

- Unique, stable, team-scoped user ID

Verification data

- Identity token, code

Sign In

Response

User ID

- Unique, stable, team-scoped user ID

Verification data

- Identity token, code

Account information

- Name, verified email

Sign In

Response

User ID

- Unique, stable, team-scoped user ID

Verification data

- Identity token, code

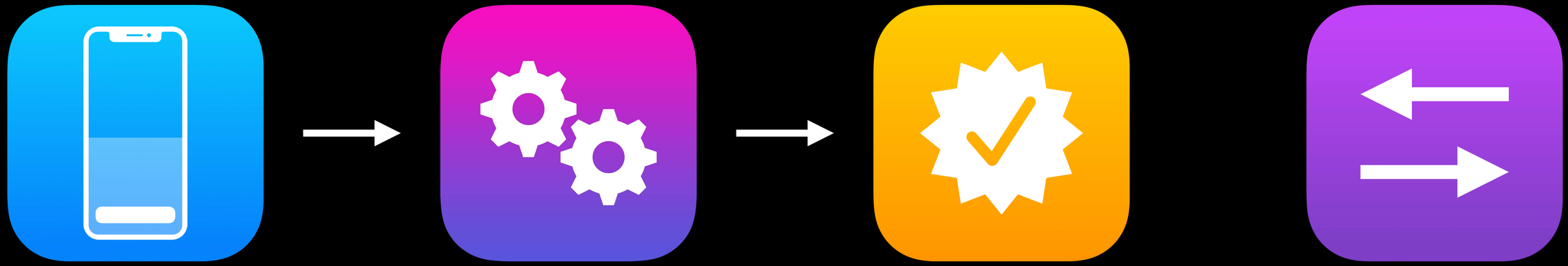
Account information

- Name, verified email

Real user indicator

- High confidence indicator that likely real user

Integrating with Your App



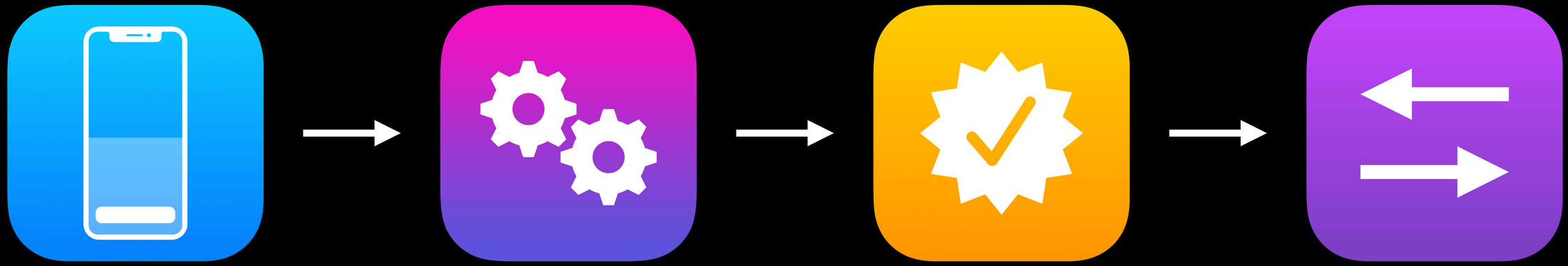
Button

Authorization

Verification

Handling Changes

Integrating with Your App



Button

Authorization

Verification

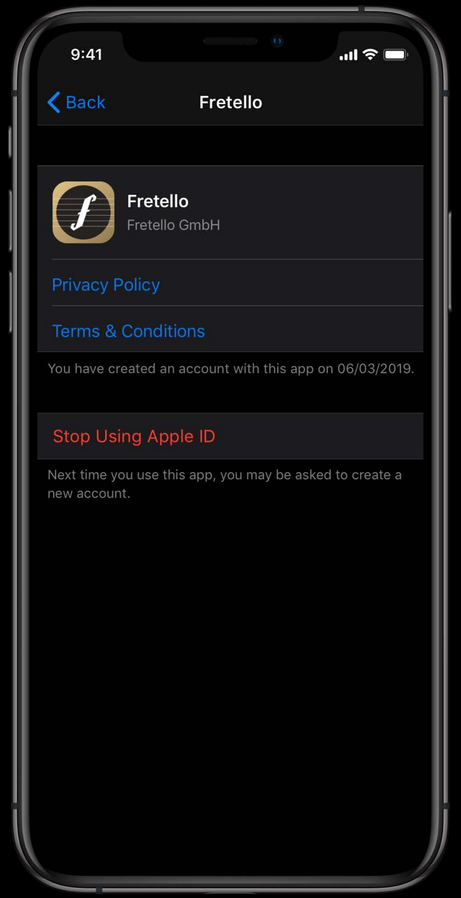
Handling Changes

Handling Session Changes

Scenarios

Stop using Apple ID with app

Sign out of the device



```swift
let provider = ASAuthorizationAppleIDProvider()
provider.getCredentialState(forUserID: "currentUserIdentifier") { (credentialState, error) in
    switch(credentialState){
    case .authorized:
        // Apple ID Credential is valid
    case .revoked:
        // Apple ID Credential revoked, handle unlink
    case .notFound:
        // Credential not found, show login UI
    default: break
    }
}
```

```swift
let provider = ASAuthorizationAppleIDProvider()
provider.getCredentialState(forUserID: "currentUserIdentifier") { (credentialState, error) in
    switch(credentialState){
    case .authorized:
        // Apple ID Credential is valid
    case .revoked:
        // Apple ID Credential revoked, handle unlink
    case .notFound:
        // Credential not found, show login UI
    default: break
    }
}
```

```swift
let provider = ASAuthorizationAppleIDProvider()
provider.getCredentialState(forUserID: "currentUserIdentifier") { (credentialState, error) in
    switch(credentialState){
    case .authorized:
        // Apple ID Credential is valid
    case .revoked:
        // Apple ID Credential revoked, handle unlink
    case .notFound:
        // Credential not found, show login UI
    default: break
    }
}
```

```swift
let provider = ASAuthorizationAppleIDProvider()
provider.getCredentialState(forUserID: "currentUserIdentifier") { (credentialState, error) in
    switch(credentialState){
    case .authorized:
        // Apple ID Credential is valid
    case .revoked:
        // Apple ID Credential revoked, handle unlink
    case .notFound:
        // Credential not found, show login UI
    default: break
    }
}
```

```swift
let provider = ASAuthorizationAppleIDProvider()
provider.getCredentialState(forUserID: "currentUserIdentifier") { (credentialState, error) in
    switch(credentialState){
    case .authorized:
        // Apple ID Credential is valid
    case .revoked:
        // Apple ID Credential revoked, handle unlink
    case .notFound:
        // Credential not found, show login UI
    default: break
    }
}
```

Handling Session Changes

Notifications

Listen through NSNotificationCenter

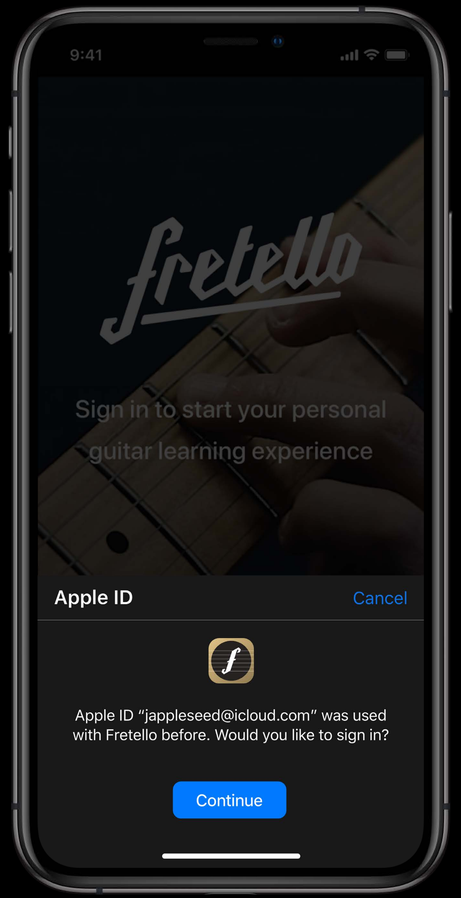
Sign user out on this device

Guide to sign in again

```
// Register for revocation notification
let center = NotificationCenter.default
let name = NSNotification.Name.ASAuthorizationAppleIDProviderCredentialRevoked
let observer = center.addObserver(forName: name, object: nil, queue: nil) { (Notification) in
    // Sign the user out, optionally guide them to sign in again
}
```

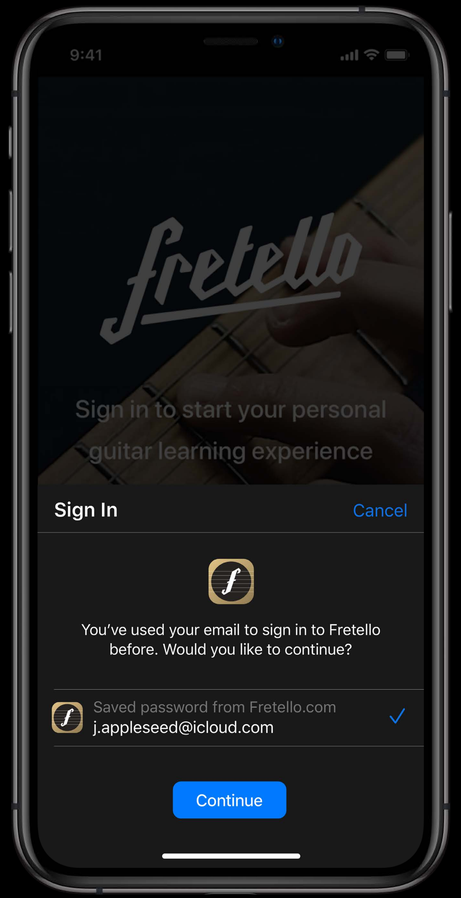
Fast Setup for Existing Accounts

One tap sign in for existing accounts



Fast Setup for Existing Accounts

Integrates with iCloud Keychain passwords



```
///Prompts the user if an existing iCloud Keychain credential or Apple ID credential exists.
func performExistingAccountSetupFlows() {
    // Prepare requests for both Apple ID and password providers.
    let requests = [ASAuthorizationAppleIDProvider().createRequest(),
                    ASAuthorizationPasswordProvider().createRequest()]

    // Create an authorization controller with the given requests.
    let authorizationController = ASAuthorizationController(authorizationRequests: requests)
    authorizationController.delegate = self
    authorizationController.presentationContextProvider = self
    authorizationController.performRequests()
 }
```

```
///Prompts the user if an existing iCloud Keychain credential or Apple ID credential exists.
func performExistingAccountSetupFlows() {
    // Prepare requests for both Apple ID and password providers.
    let requests = [ASAuthorizationAppleIDProvider().createRequest(),
                    ASAuthorizationPasswordProvider().createRequest()]

    // Create an authorization controller with the given requests.
    let authorizationController = ASAuthorizationController(authorizationRequests: requests)
    authorizationController.delegate = self
    authorizationController.presentationContextProvider = self
    authorizationController.performRequests()
 }
```

```
///Prompts the user if an existing iCloud Keychain credential or Apple ID credential exists.
func performExistingAccountSetupFlows() {
    // Prepare requests for both Apple ID and password providers.
    let requests = [ASAuthorizationAppleIDProvider().createRequest(),
                    ASAuthorizationPasswordProvider().createRequest()]

    // Create an authorization controller with the given requests.
    let authorizationController = ASAuthorizationController(authorizationRequests: requests)
    authorizationController.delegate = self
    authorizationController.presentationContextProvider = self
    authorizationController.performRequests()
}
```

```
func authorizationController(controller _: ASAuthorizationController,
                             didCompleteWithAuthorization authorization: ASAuthorization) {
    switch authorization.credential {
    case let credential as ASAuthorizationAppleIDCredential:
        let userIdentifier = credential.user
        // Sign the user in using the Apple ID credential
    case let credential as ASPasswordCredential:
        // Sign the user in using their existing password credential
    default: break
    }
}
```

```swift
func authorizationController(controller _: ASAuthorizationController,
                             didCompleteWithAuthorization authorization: ASAuthorization) {
    switch authorization.credential {
    case let credential as ASAuthorizationAppleIDCredential:
        let userIdentifier = credential.user
        // Sign the user in using the Apple ID credential
    case let credential as ASPasswordCredential:
        // Sign the user in using their existing password credential
    default: break
    }
}
```

```swift
func authorizationController(controller _: ASAuthorizationController,
                             didCompleteWithAuthorization authorization: ASAuthorization) {
    switch authorization.credential {
    case let credential as ASAuthorizationAppleIDCredential:
        let userIdentifier = credential.user
        // Sign the user in using the Apple ID credential
    case let credential as ASPasswordCredential:
        // Sign the user in using their existing password credential
    default: break
    }
}
```

Demo

Jonathan Birdsall, iOS Engineer

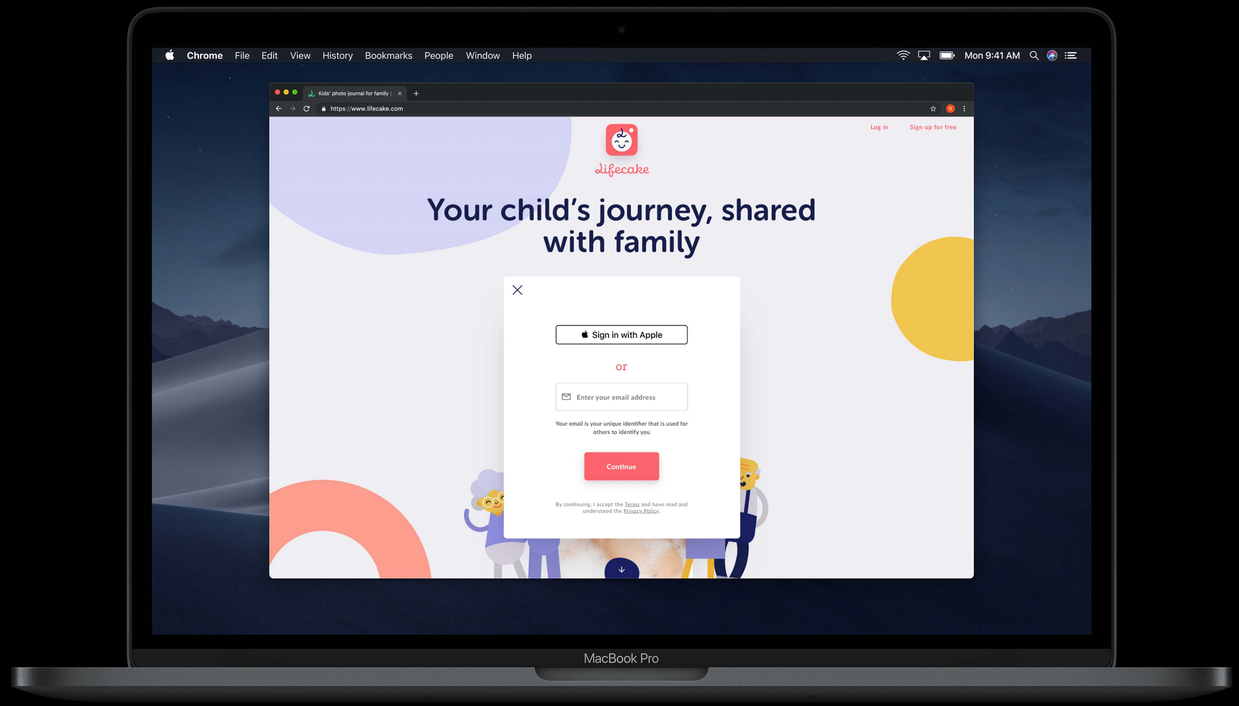
Cross-Platform

Gokul Thirumalai, Engineering Manager

Cross-Platform

JavaScript SDK

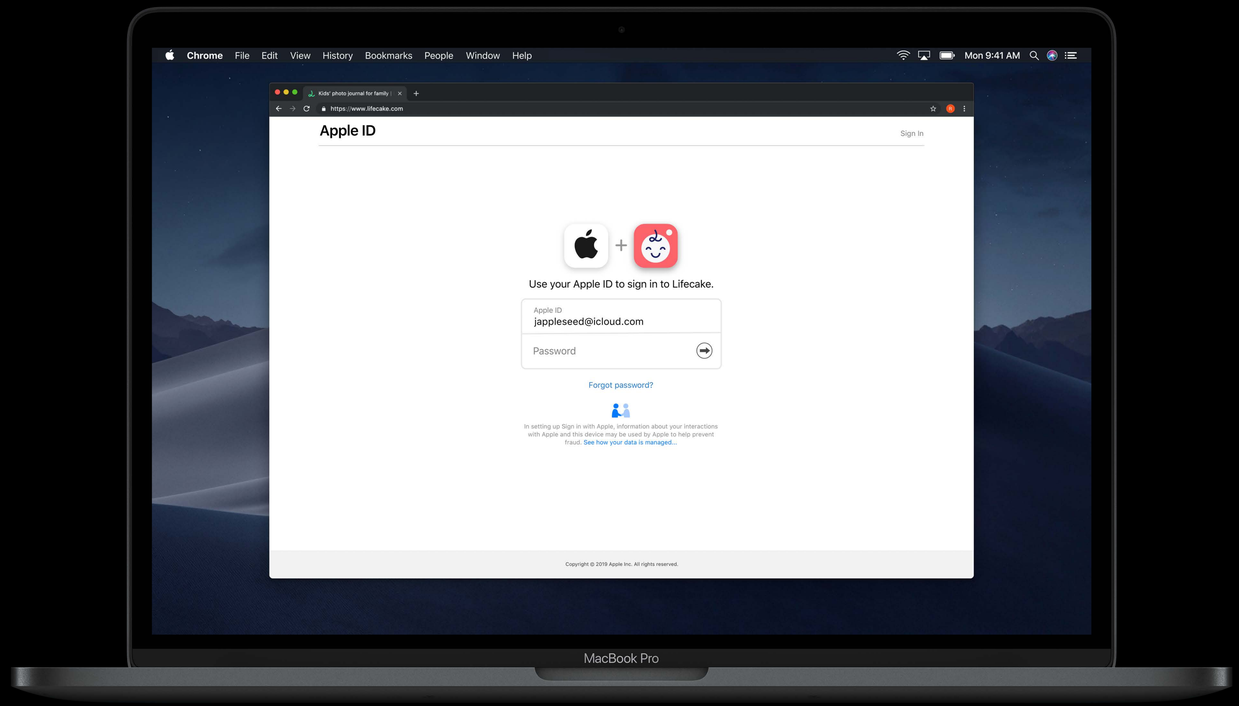
Simple browser-based login



Cross-Platform

JavaScript SDK

Simple browser-based login

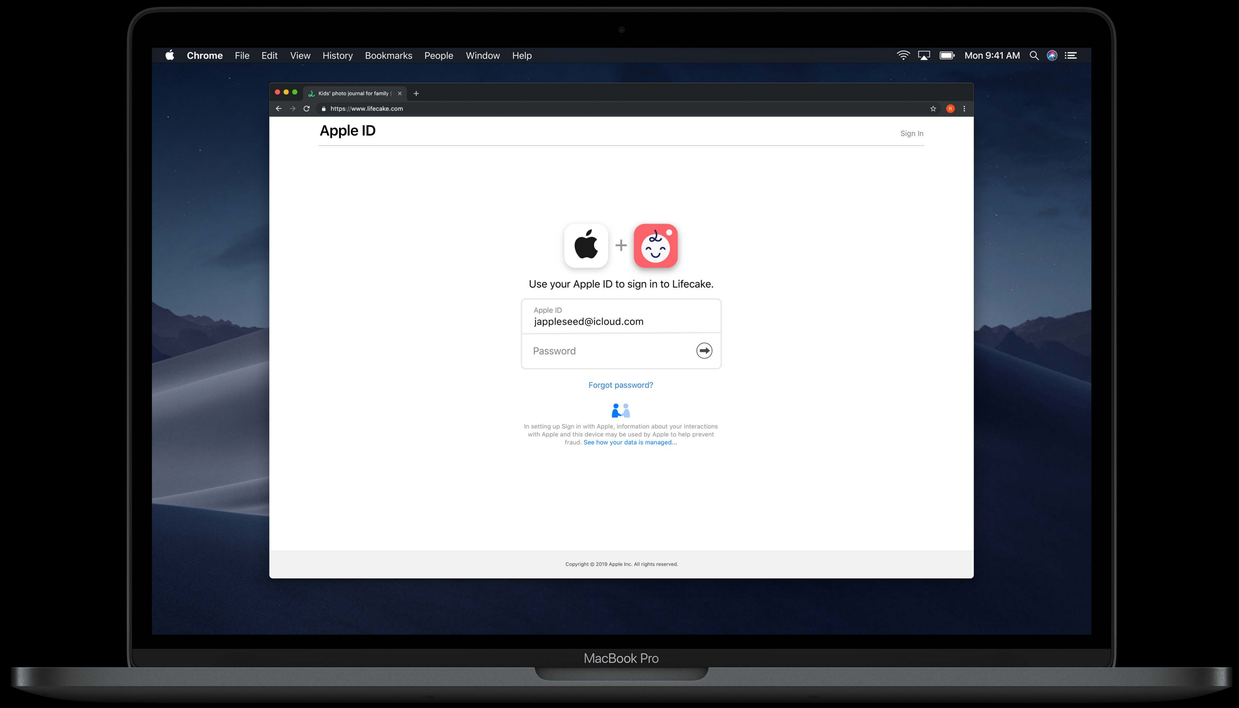


Cross-Platform

JavaScript SDK

Simple browser-based login

Similar to native API



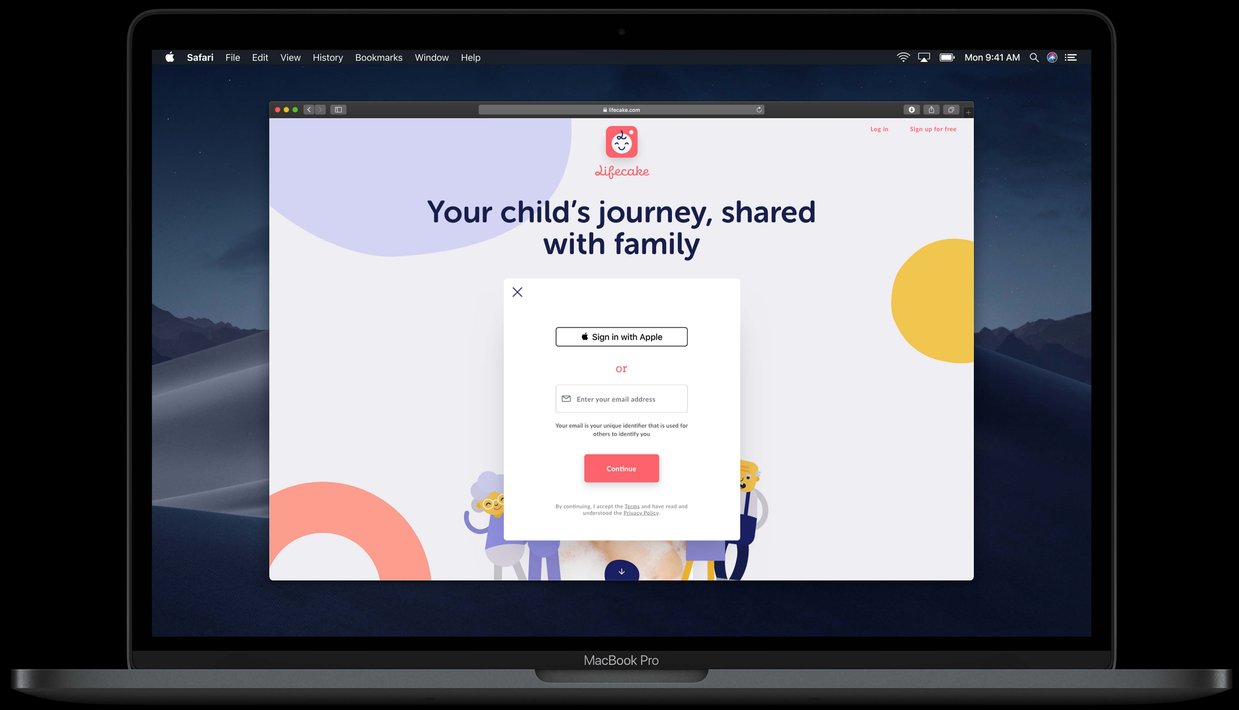
Cross-Platform

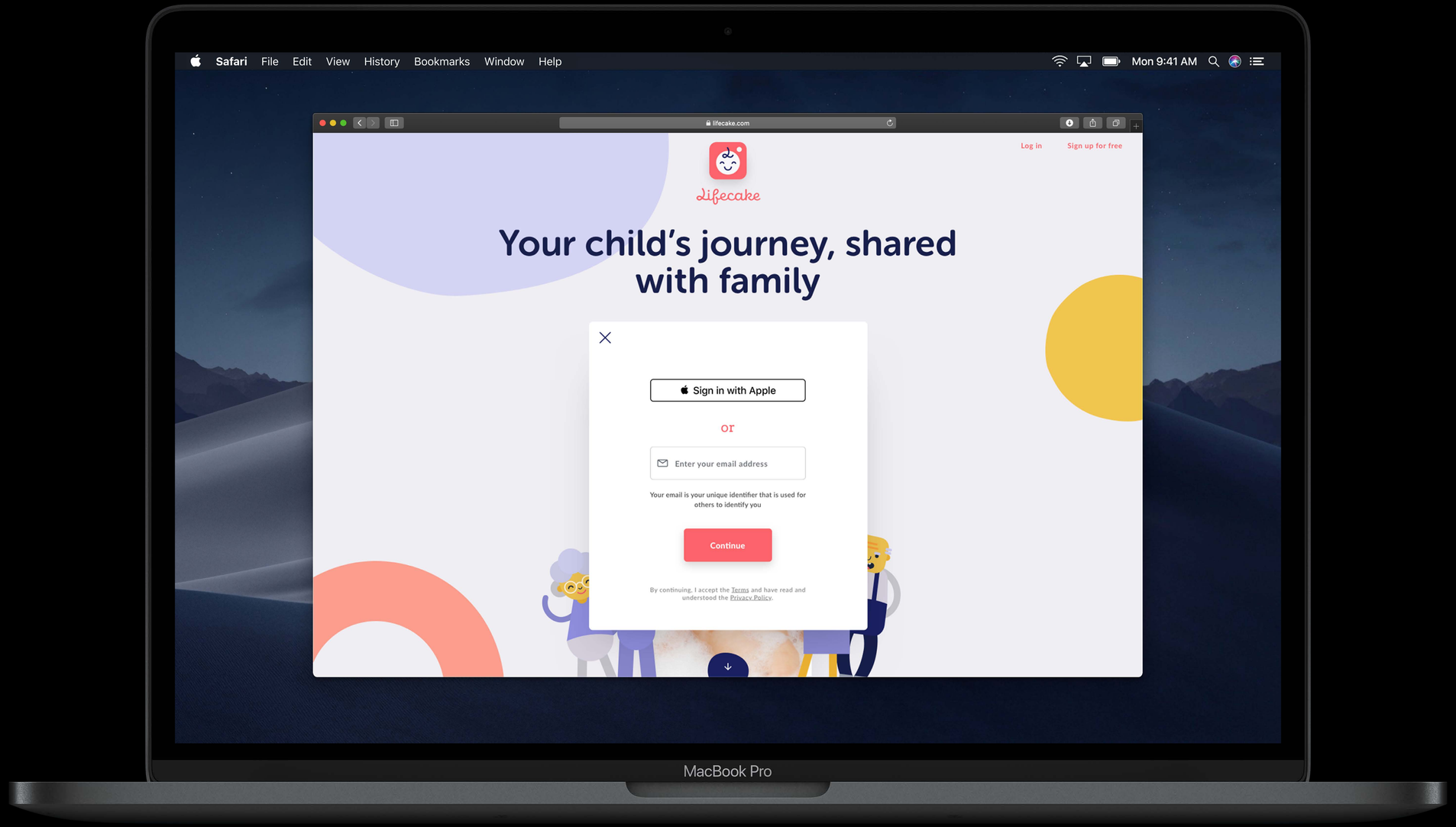
JavaScript SDK

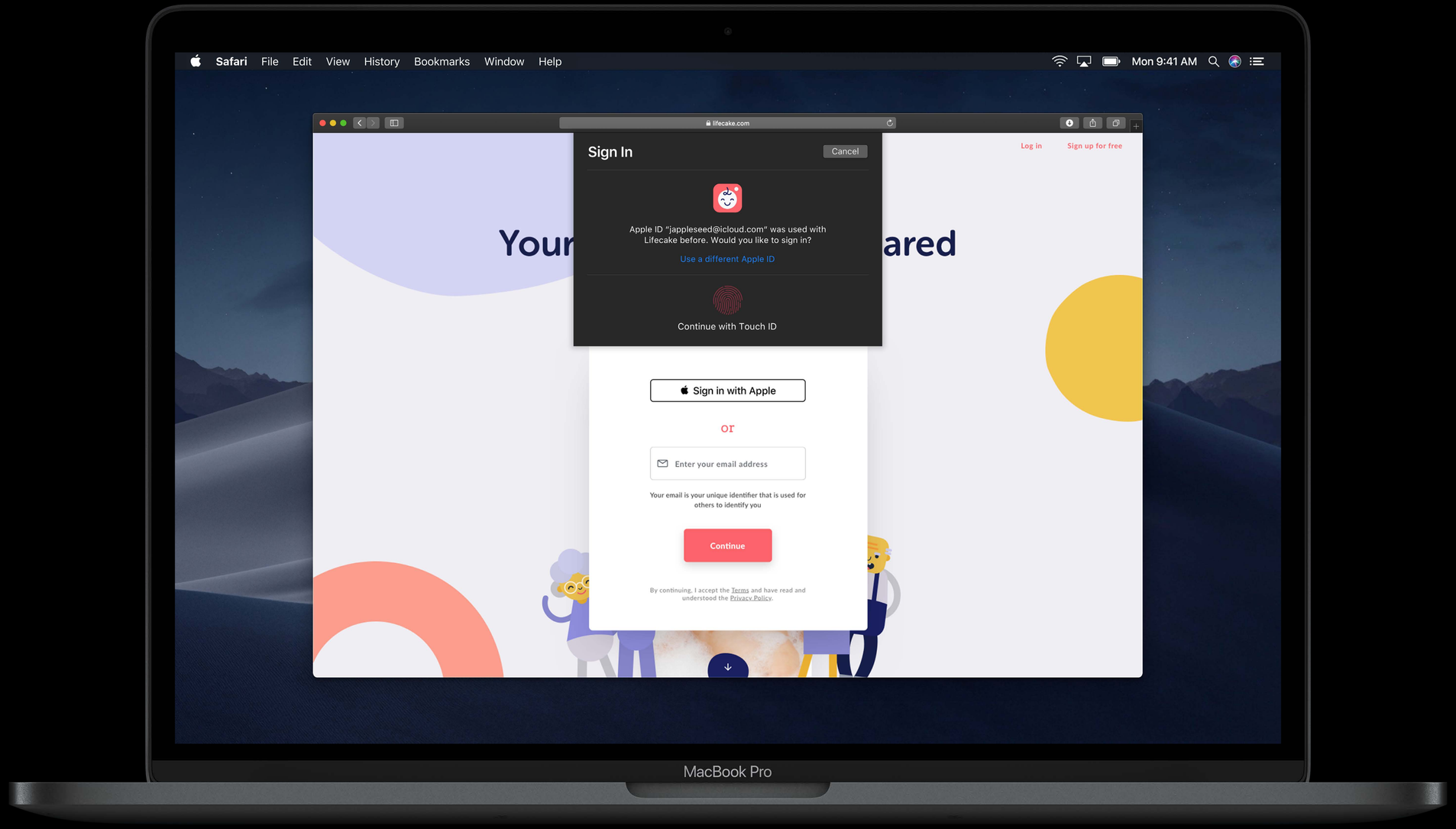
Simple browser-based login

Similar to native API

Native experience when using Safari




Safari File Edit View History Bookmarks Window Help
Mon 9:41 AM
lifecake.com
Log in
Sign up for free
Lifecake
Your child's journey, shared with family
Sign in with Apple
or
Enter your email address
Your email is your unique identifier that is used for others to identify you
Continue
By continuing, I accept the Terms and have read and understood the Privacy Policy.
MacBook Pro


Safari File Edit View History Bookmarks Window Help
Mon 9:41 AM
lifecake.com
Log in
Sign up for free
Sign In
Cancel
Apple ID "jappleseed@icloud.com" was used with Lifecake before. Would you like to sign in?
Use a different Apple ID
Continue with Touch ID
Sign in with Apple
or
Enter your email address
Your email is your unique identifier that is used for others to identify you
Continue
By continuing, I accept the Terms and have read and understood the Privacy Policy.
MacBook Pro

Cross-Platform

JavaScript SDK

Include

Button

Configure

Result

Cross-Platform

JavaScript SDK

Include

Button

Configure

Result

```
<script src="https://appleid.cdn-apple.com/appleauth/static/
             jsapi/appleid/1/en_US/appleid.auth.js">
```

Cross-Platform

JavaScript SDK

Include

Button

Configure

Result

```
<div id="appleid-signin"></div>
```

Cross-Platform

JavaScript SDK

Include

Button

Configure

Result

```
AppleID.auth.init({
    clientId : 'com.example.webapp',
    scope : 'name email',
    redirectURI: 'https://example.com/redirectUri',
    state : 'state'
});
```

Cross-Platform

JavaScript SDK

Include

Button

Configure

Result

```
POST /redirectUri
```

Best Practices

Best Practices

Only require account setup when necessary

Best Practices

Only require account setup when necessary

Only collect data that is required

Best Practices

Only require account setup when necessary

Only collect data that is required

Respect email address that user chose to share

Best Practices

Check for existing accounts on app startup

Best Practices

Check for existing accounts on app startup

Use real user indicator for best new account experience

Best Practices

Check for existing accounts on app startup

Use real user indicator for best new account experience

Always use the Button API

Best Practices

Check for existing accounts on app startup

Use real user indicator for best new account experience

Always use the Button API

Implement across all platforms

Summary

Streamlined account setup

Verified email addresses

Built-in security

Anti-fraud

Cross-platform

More Information

developer.apple.com/wwdc19/706

Introducing Sign In with Apple Lab	Wednesday, 10:00
What’s New in Authentication	Thursday, 11:00
Creating Independent Watch Apps	WWDC App


WWDC19