Published in Wolox Juan F. Caracciolo Q Search Jul 13, 2018 · 6 min read · ▶ Listen Juan F. Caracciolo http://www 170 Followers Software engineer student, IOS developer, Game designer aficionado. More from Medium WOLOX ohdarling **Analyze iOS App Binary size with Link Map** IOS Deep linking: URL Scheme vs Universal Links Antonio Romano Everything is connected nowadays. In a world where we share links as **Implementing Game Controller** often as we do today, your app cannot be out of the loop. Deep linking for iOS or Mac app is the idea of not only having a clickable link to open up your app but a Utsukushii Jinsei smart one that will also navigate to the desired resource. Improve your My Opinion On Swift UI user experience implemented these useful shortcuts. Daniel... in Bilue Product Design & T... . . . How to add CoreHaptics to your iOS app There are 2 ways of implementing deep linking in IOS: <u>URL scheme</u> and <u>Universal Links</u>. While URL schemes are a well-known way of having deep linking, Universal links are the new way Apple has implemented to easily connect your webpage and your app under the same link. Here are comparisons for you to choose from, but I will proceed to explain both: **URL Scheme** Help Status Writers Blog Careers Privacy Terms About Cons Pros Will always ask for permission (pop up) Easy to implement out of the box User are not familiar with this type of URLs No extra backend required Won't work in other platforms (such as Android) Won't work if app is not installed **Universal Links** Pros Cons Won't ask for permission Static backend with ssl needed Won't open the browser More complex to implement Completely compatible with other platforms Fallback url if app is not installed Take note here, neither of these options will redirect you to the App Store if the app is not installed. Once the App is installed, IOS will make the connection between these URLs and the app based on the app's metadata, if the app is not installed, the metadata is here and thus the connection is never created. We will have to work that out ourselves later. **URL Schemes** Let's start with URL Schemes. URL schemes are easy to implement. All you have to do is tell the app which scheme would you like to use. For this just Open Xcode, got to *Project Settings -> Info*, and add inside 'The URL Types" section a new URL scheme. Add something of the sort of com.myApp and that's it. Now install your app, open the app notes and type com.myApp and press enter. ▼ URL Types (1) URL Schemes | com.myApp Identifier None No image specified Additional url type properties (0) Wait... nothing is happening. Well for IOS to recognize this as a link, you have to accommodate to the URL format. That is scheme://resource. So we go ahead and type com.myApp://main. Now when we press enter, we can see IOS detected the link and when we click it, a pop up will ask for permission to open MyApp from Notes. Now, what about if we want to redirect the user inside the app? If the user taps com.myApp://profile it should show the profile or com.myApp://reset_password, take him to the reset password screen. If we want to react once inside the app to this URL, we need to implement one method inside the **AppDelegate** 1 func application(_ app: UIApplication, open url: URL, options: [UIApplicationOpenURLOptionsKey : Any] = [:]) -> Bool { if let scheme = url.scheme, scheme.localizedCaseInsensitiveCompare("com.myApp") == .orderedSame, let view = url.host { var parameters: [String: String] = [:] URLComponents(url: url, resolvingAgainstBaseURL: false)?.queryItems?.forEach { parameters[\$0.name] = \$0.value 10 11 12 redirect(to: view, with: parameters) 13 return true 15 } URLSchemeExample.swift hosted with ♥ by GitHub view raw This way **com.myApp://profile?user="JuanFra"** will result in: url.scheme = "com.myApp" url.host = "profile" parameters = ["user" : "JuanFra"] That's all for URLSchemes. Easy right? Remember this URL won't work if the app is not installed, so what IOS will attempt to do is open up the link in Safari, which will obviously lead to nowhere, leaving the user looking at an empty white screen confused about what just happened. For that, we have a second option. **Universal Links** Universal links are a bit more complex. Basically, we want IOS to relate a webpage URL to our app. However, that is not so straightforward. Picture this situation. Our beloved app "Redd1t" is in all ways, shapes, and forms exactly like the popular social network "Reddit". Redd1t indicates in its metadata whenever a user presses *reddit.com*, to redirect it to its app. See the problem here? IOS needs a way of validating that in fact, Redd1t is the owner of www.reddit.com, but how? Well, ask https://www.reddit.com of course! **Reddit.com** will need to provide a resource that indicates which is his IOS app. And it does so in the following way: IOS will make a request to https://www.reddit.com/apple-app-site-association and expect a JSON. The JSON should be as following: "appID": "T5TQ36Q2SQ.com.reddit.production",
"paths": ["*"], Let's break it down for a second. **Applinks** indicate this is indeed for the Universal Link declaration. Apps should be left as an empty array (quote from Apple: "The apps's key in an apple-app-site-association file must be present and its value must be an empty array). This is most likely because these types of JSON are used for other purposes other than universal links as well, but we won't take it into consideration. Details will then contain an array of your apps and the mapping of each subpath to the respective app. For each app, you should add a field called appID which is obtained concatenating your teamID and your app's bundleID. For example, if your TeamID is 123456 and your AppID is com.myApp, then the result is 123456.com.myApp. In the **paths** field, an array of strings representing with expressions of the paths which correspond to this app. For example, if you want *myApp.com/store* to open up a different app than *myApp.com/maps*, here you can declare that * is a wildcard for any string, while? is a wildcard for any character. As we only have one app, any subpath will lead here, hence the *. If you want to exclude a subpath, just add **NOT** at the beginning ["/wwdc/news/", "NOT /videos/wwdc/2010/*", "/videos/wwdc/201?/*"] The system evaluates the path in order and will stop when finding a **NOT**, so take that into consideration when selecting the order. In fact, if we go to https://www.reddit.com/apple-app-site-association we can see the response. Remember the link must no end in .json, and the request must return a header content-type: application/json. If you are having trouble setting up your apple-app-site-association, our friends at Branch.io will help you out with their great validator https://branch.io/resources/aasa-validator/ Once that is done, we now must add the correct metadata to the app. First, open **Xcode**, go to *Project settings -> capabilities*. Scroll down to **Associated Domains** and turn it on. Once it is enabled, we shall add any URL that implements our apple-app-site-association resource, preceded by app links. Inside the Domains section, add a applinks:myApp.com. Once this is done, go ahead and try out your app. If you go to My App with safari, after the app is installed, you should see a little banner indicating you that you could open up that link inside an app. If you click it the app will launch. Airbnb, Inc. Arroyo Grande, CA, Un... If you go ahead to notes and write down www.myApp.com, notes will recognize this as a link and let you open it. If you click it then, it will open up your app directly, not even launching safari. This will only happen if the user taps a link, not if they visit the webpage. But now, say we want to redirect the user to the desired screen base on the URL parameters. We then have to implement the following method in the AppDelegate. 1 public func application(_ application: UIApplication, continue userActivity: NSUserActivity, restorationHandler: @escaping ([Any]?) -> Void) -> Bool { if let url = userActivity.webpageURL { var view = url.lastPathComponent var parameters: [String: String] = [:] URLComponents(url: url, resolvingAgainstBaseURL: false)?.queryItems?.forEach { parameters[\$0.name] = \$0.value 10 11 redirect(to: view, with: parameters) 13 return true 14 } UniversalLinksExample.swift hosted with ♥ by GitHub What universal links allow us is not only to have a unique URL for the webpage, the IOS app and maybe even the Android App. But more than that, Universal links allow us to have a fallback webpage if a user does not have the app installed. If you send an email to a user, with the link www.myApp.com/app/profile, you could host there a static webpage that tells the user to go the App Store or even just automatically redirect it, since, **if the user** has the app, the link will open it, and if not, it will fall under the webpage. Wrapping things up Taking all this into consideration, deep linking is a crucial feature in today's apps, especially if you are sending email notifications to your users. Users consume content faster every day and the time it takes them to get to your app and navigate to the desired location is the time you are making them waste. So, what are you waiting for? 3.2K | Q 14 3.2K Q 14 More from Wolox We specialize in end-to-end development of high impact products, providing technological solutions to start-ups and companies that are seeking to innovate and need support in developing their ideas. In January 2021, we became part of Accenture. Leandro Motta · Jul 3, 2018 **Git Feature Branching On Steroids** An easy workflow for productive applications — We've all been there in the beginning, we learn the basics about Git and start using it for our own projects, university's coding courses or even in our first job in IT. Since m... \Box Git 8 min read Share your ideas with millions of readers. Write on Medium Marcos Trucco · Jun 29, 2018 **Testing in .NET Core 2.0** Unit and Integration tests with xUnit — Hi there! Here is our new post about .NET Core. In the previous one, we learned how to deploy our .NET Core application using Docker. Make sure you check it out! In this post, w... Dotnet Core 7 min read Martín Thompson · Jun 21, 2018 El que quiera celeste, que defina su atributo Como diseñadores tenemos el desafío de generar una búsqueda y definición de ciertas ideas gráficas o valores conceptuales que se quieren transmitir, ya sea con atributos de trasfondo o por mera estética visual. ... \Box Design 6 min read Francisco Iglesias · May 18, 2018 **Starting up with SEO** I was asked to analyze the SEO rank of a project I was working on in Wolox, and, after an analysis made by the QA department, the outcome resulted in a very low figure. So, we started doing some research about search engin... SEO 5 min read Juan Ignacio Sierra · May 11, 2018 Handling integrations in RoR Nowadays almost all software projects have at least one integration. At Wolox, we have encountered many of those. For example, my last project required two parallel asynchronous integrations that gave the system... Ruby On Rails 4 min read Read more from Wolox Recommended from Medium Nicolas Laveau in Criteo R&D Blog John Au-Yeung in Dev Genius **Highlights of AngularConnect Vuetify—Text Areas** Malcolm Gourdine John Au-Yeung in Level Up Coding This and Bind in javascript. Plain JavaScript of Lodash **Array Methods** TOPIDEA Yagnik Ka... in JavaScript in Plain ... React Portals: A Way to MEMES — Oh so that's why.... Separate a Child from its Parent on a React DOM John Au-Yeung in DataSeries John Au-Y... in JavaScript in Plain ...

How to Use GraphQL API in

Node.js Apps

Best of Modern JavaScript—

OOP Features

G Sign in to Medium with Google

Zhengqian Kuang

John Kuang

dev.kuang@gmail.com

jkuang@gasbuddy.com

3 more accounts

Sign In

Get started

X