



CODE > IOS SDK

# </> An Introduction to Handoff

by [Davis Allie](#) 7 Aug 2015

Difficulty: Intermediate Length: Medium Languages: English

iOS SDK iOS 8 iOS Swift Xcode Xcode 6 IDEs Mobile Development

App Development Mobile App



## Introduction

With iOS 8 and OS X Yosemite, Apple introduced a great new feature for developers to take advantage of, [Handoff](#). Handoff allows apps to transfer

data and application state information from one device to another over Bluetooth. This enables users of your apps to begin a task on one of their devices and then continue it seamlessly on another.

An example of this would be starting to write a message on your iPhone and then finishing and sending that same message on your iPad. In this tutorial, I am going to show you how you can adopt Handoff in your own applications through a very simple note taking app.

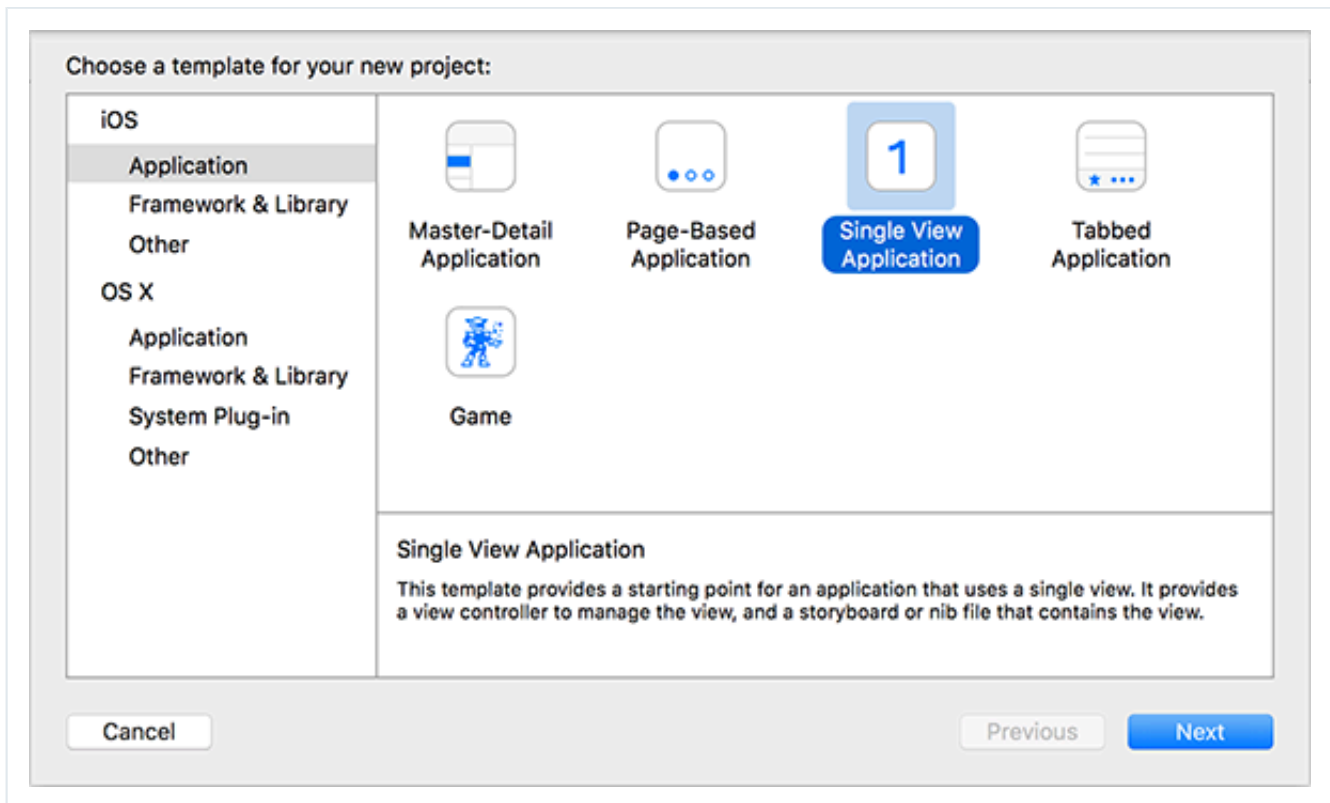
This tutorial requires that you are running Xcode 6+ and have two Handoff compatible devices. Not all iOS 8 devices have [Bluetooth LE](#) (Low Energy), which is required for Handoff. This feature is therefore only available—and can only be tested on—the following devices:

- iPhone 5 and later
- iPad 4th Generation, including every iPad Air model
- iPad Mini
- iPod Touch 5th Generation

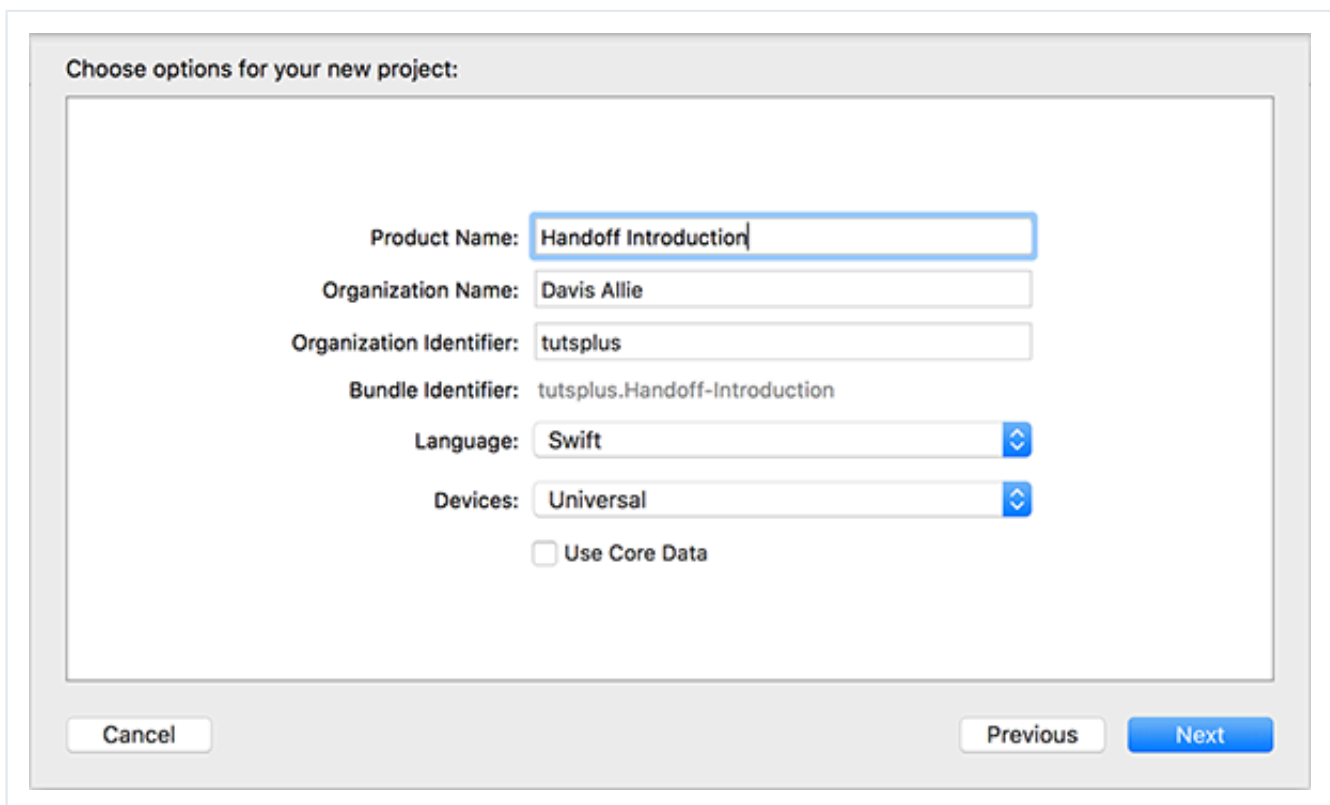
During testing, you need to sign in to the same iCloud account on each device and enable Handoff in the device's settings.

# 1. Project Setup

Create a new project in Xcode and choose the **Single View Application** template from the **iOS > Application** section.



Configure the project as shown below. Note that **Language** is set to **Swift** and **Devices** is set to **Universal**.



Once Xcode has created your project, open **ViewController.swift** and replace the implementation of the `ViewController` class with the following implementation:

```
01 class ViewController: UIViewController, UITextFieldDelegate, UITextView
02
03     var noteTitleField: UITextField!
04     var noteContentView: UITextView!
05
06     override func viewDidLoad(animated: Bool) {
07         self.noteTitleField = UITextField(frame: CGRect(x: 12, y: 28,
08         self.noteTitleField.placeholder = "Note Title"
09         self.noteTitleField.delegate = self
10
11         self.noteContentView = UITextView(frame: CGRect(x: 8, y: 56,
12         self.noteContentView.text = "Note Content"
13         self.noteContentView.delegate = self
14
15         self.view.addSubview(self.noteTitleField)
16         self.view.addSubview(self.noteContentView)
17     }
18
19     func textViewDidBeginEditing(textView: UITextView) {
20         if textView.text == "Note Content" {
21             textView.text = ""
22         }
23     }
24
25     override func viewDidLoad() {
26         super.viewDidLoad()
27         // Do any additional setup after loading the view, typically
28     }
29
30 }
```

The implementation is pretty straightforward. It makes the `ViewController` class adopt both the `UITextFieldDelegate` and `UITextViewDelegate` protocols, and it adds a `UITextField` and `UITextView` to the view controller's view for text input.

Build and run your app on one of your test devices and you should see a very

basic user interface with two input fields, a text field for a title and a text view for the note's content.

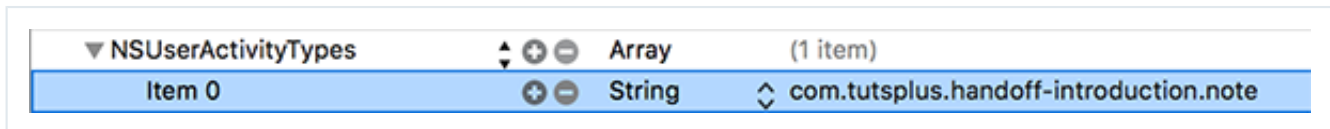


## 2. Setting Up Handoff

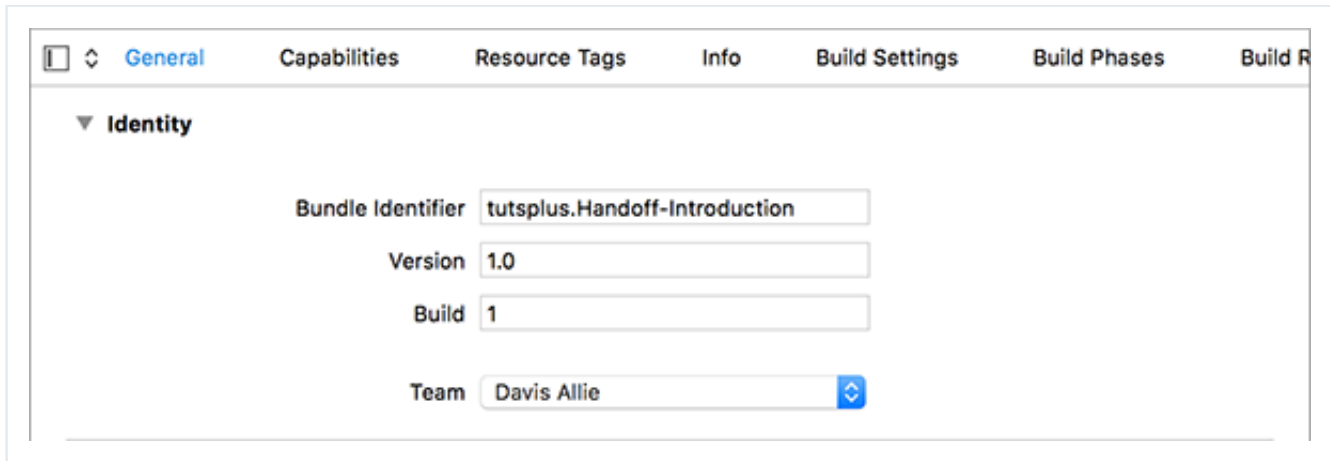
Activities transferred between devices with Handoff are modelled by the `NSUserActivity` class. You will learn more about this class and how to use it later in this tutorial. First, your app must be configured to successfully create these activities.

Each type of activity your app supports must be given a unique identifier, similar to how every iOS application has its own unique id. These identifiers can be whatever you want but the best practice is to use reverse-DNS notation, such as **com.tutsplus.handoff-introduction.note**. The instance of the application running on each device must also be signed by the same iOS development team in order for the activities to be recognized properly.

You first need to add the activity identifiers that your app supports to the target's **Info.plist**. Open the target's **Info.plist** and add the **NSUserActivityTypes** key. Make this item an **Array** and add a single item, **com.tutsplus.handoff-introduction.note**, as shown in the screenshot below.



Next, select the project in the **Project Navigator** and, in the editor on the right, open the **General** tab. In the **Identity** section, set **Team** to the correct iOS development team.



With these steps completed, provided you are signed in to the same iCloud account and connected to the same Wi-Fi network on each of your testing devices, your application is ready to start using Handoff.

### 3. Sending a User Activity

Next, you need to add some code in order for the user activity to be sent from one device to another. Update the `viewDidLoad` method of the `ViewController` class as shown below.

```
1  override func viewDidLoad() {  
2      super.viewDidLoad()  
3  
4      let activity = NSUserActivity(activityType: "com.tutsplus.handoff.  
5      activity.title = "Note"  
6      activity.userInfo = ["title": "", "content": ""]  
7      userActivity = activity  
8      userActivity?.becomeCurrent()  
9  }
```

This creates a basic `NSUserActivity` object with the identifier you added to your target's **Info.plist** earlier. You give this activity object a `title` and a `userInfo` dictionary with empty string values for the **title** and **content** keys.

In this tutorial, we are going to be putting strings into our activity's `userInfo` dictionary. However, you can add any property list type, such as numbers, arrays, and dictionaries, as well as any `NSCoding` compliant object.

You then assign this new activity to the `userActivity` property of your `ViewController` instance so that the Handoff API has a reference to it. The `UIViewController` class inherits this property from the `UIResponder` class.

Lastly, you call `becomeCurrent` on the activity object to tell the system that this is the current user activity to be sent out by the application.

You have now set up your user activity successfully, but we also need to fill it with some content as the user writes their note. To do this, we update the current user activity by overriding the `updateUserActivityState(_:)` method in the `ViewController` class. This method is called periodically by the Handoff API to update the current user activity. Add the following code to the `ViewController` class.

```
01  override func updateUserActivityState(activity: NSUserActivity) {  
02      activity.addUserInfoEntriesFromDictionary(["title": self.noteTitl  
03      super.updateUserActivityState(activity)  
04  }  
05  
06  func textField(textField: UITextField, shouldChangeCharactersInRange  
07      self.updateUserActivityState(userActivity!)  
08      return true  
09  }  
10  
11  func textView(textView: UITextView, shouldChangeTextInRange range: NS  
12      self.updateUserActivityState(userActivity!)  
13      return true  
14  }
```

The `updateUserActivityState(_:)` method gives us a reference to the current user activity and it replaces the values in the activity's `userInfo` dictionary with



the latest values from your app. Note that we also invoke this method on the superclass.

We also implemented two other methods,

`textField(_:shouldChangeCharactersInRange:replacementString:)` of the `UITextFieldDelegate` protocol

and `textView(_:shouldChangeTextInRange:replacementText:)` of the `UITextViewDelegate` protocol. In these methods, we update the current user activity whenever the text in any of the input fields changes. While this isn't necessary, this ensures that your activity will always contain the latest information.

You are now ready to test out your app with Handoff. Build and run your app on both of your test devices. Press the lock button on one device to put the device to sleep while keeping the app open on the other. Wake up the device that you just locked and you should see the app icon appear in the bottom left corner of the lock screen. Perform a swipe gesture on this icon and your app will resume via Handoff.

Alternatively, you can double-click the home button to enter the app switcher and your Handoff capable app will show up on the left of the home screen. At the moment, your app only resumes to the same blank interface. Let's fix that now.

## 4. Restoring From a User Activity

Restoring an app from a Handoff `NSUserActivity` is handled by your application delegate. The app delegate then passes the user activity object to the application's root view controller to restore itself. If the view controller that needs to process the activity isn't the root view controller of your application, then you simply pass it from the root view controller down to the view

controller hierarchy until you reach the desired place in your app.

Open **AppDelegate.swift** and add the following method to the `AppDelegate` class:

```
1 func application(application: UIApplication, continueUserActivity user
2     self.window?.rootViewController?.restoreUserActivityState(userActi
3     return true
4 }
```

In this method, you pass the user activity object to the application's root view controller and, by returning `true`, you tell the application that you have successfully received and processed the Handoff user activity.

Next, open **ViewController.swift** and add the following method to the `ViewController` class:

```
1 override func restoreUserActivityState(activity: NSUserActivity) {
2     self.noteTextField.text = activity.userInfo?["title"] as! String
3     self.noteContentView.text = activity.userInfo?["content"] as! Stri
4 }
```

Similar to the `updateUserActivityState(_:)` method from earlier in this tutorial, you override the `restoreUserActivityState(_:)` method in order to retrieve the information from the `NSUserActivity` object. In your implementation of this method, you update both of your input fields with the data stored in the user activity object.

Build and run your app on both of your test devices and begin writing a note on one device. From either the lock screen or app switcher on your other device, open the app via Handoff and you should see the text that you had been writing on your first device on your second one.



## 5. Error Handling and Delegate Methods

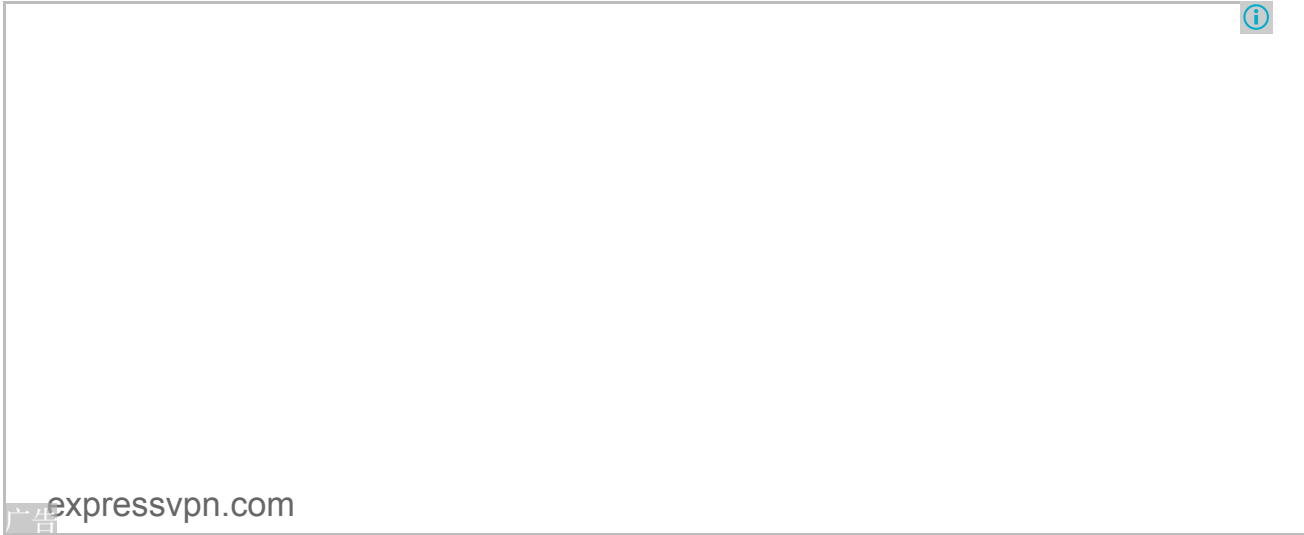
Unlike many APIs provided by Apple for iOS, error handling isn't easy when using Handoff if you don't know where these errors need to be handled. The first point at which your app will be notified of a Handoff error is via the `application(_:didFailToContinueUserActivityWithType:error:)` in the app delegate class. In this method you can determine what the cause of error was and what type of activity the error relates to. Note that the type of an `NSUserActivity` object is the same as the unique identifier you assign it.

When your application continues from a Handoff activity, it first needs to download the data associated with the activity from the original device over Bluetooth. Before this download is complete though, another application delegate method is called: `application(_:willContinueUserActivityWithType:)`. In this optional method, you can return a boolean value to tell the Handoff API whether or not you want to continue receiving the user activity. In some situations, this may be useful as you can disable a particular type of `NSUserActivity` when certain conditions are met.

## Conclusion

In this tutorial, I showed you how you can use the `NSUserActivity` class to easily adopt Handoff in your own iOS 8 applications. You created a very simple application, capable of transferring data wirelessly to another device over Bluetooth.

While the example application we went through in this tutorial was very simple, your own applications can have as many user activity types as you want for a wide variety of functionality. As always, if you have any comments or questions, leave them in the comments below.



Advertisement

**Davis Allie**

Australia

Davis is an 19 year old iOS developer who has been programming for multiple years. His knowledge spans across C++, Java, Objective-C and, most recently, Swift. Davis combines his studies as a student with app development and writing tutorials here on Tuts+.

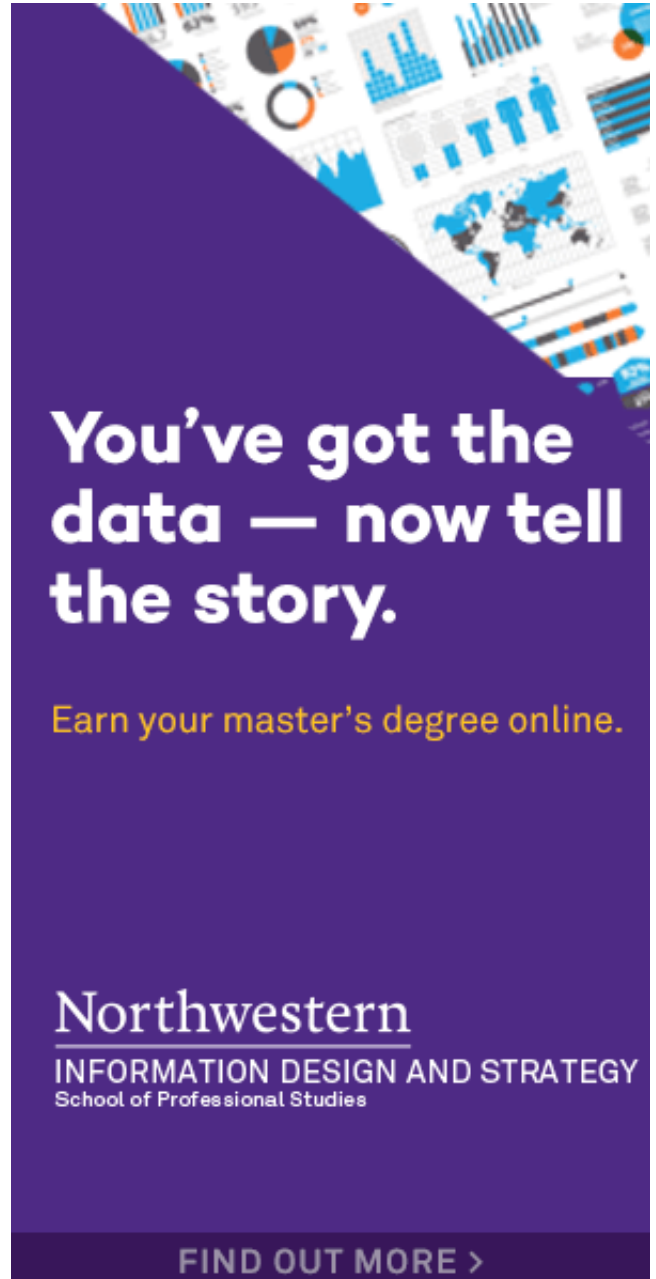


tuts+

**STUDENT ACCESS**

# JUST \$90/YR

Courses, eBooks & more >



**You've got the data — now tell the story.**

Earn your master's degree online.

**Northwestern**  
INFORMATION DESIGN AND STRATEGY  
School of Professional Studies

**FIND OUT MORE >**

Advertisement

[View on Github](#)

## Translations

Envato Tuts+ tutorials are translated into other languages by our community members—you can be involved too!

Translate this post

Powered by  **native**

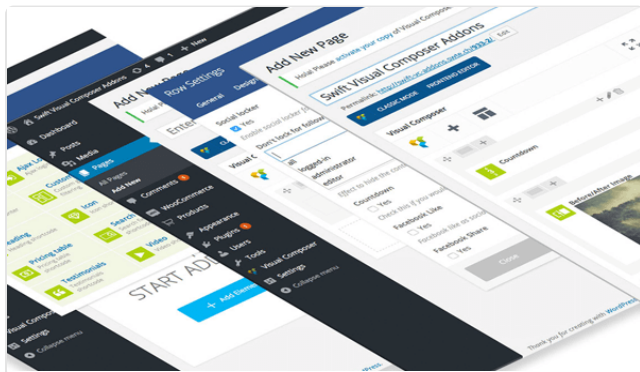


Advertisement

**Looking for something to help kick start your next project?**

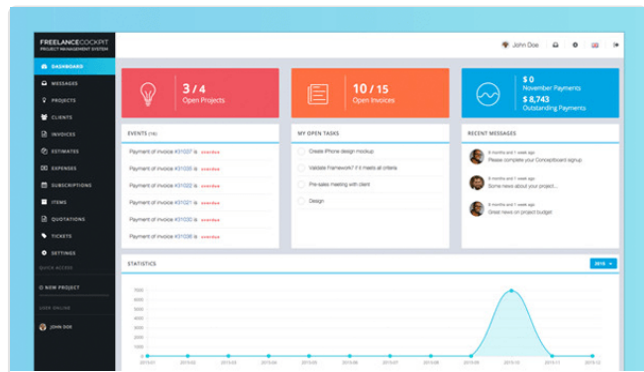


Envato Market has a range of items for sale to help get you started.



## WordPress Plugins

From \$4



## PHP Scripts

From \$1



## JavaScript

From \$2



tuts+

Teaching skills to millions worldwide.

22,528 Tutorials

898 Video Courses

### Meet Envato

About Envato

Explore our Ecosystem

Careers

### Join our Community

Teach at Envato Tuts+

Translate for Envato Tuts+

Forums

Community Meetups

### Help and Support

FAQ

Help Center

Terms of Use

About Envato Tuts+

Advertise

## Email Newsletters

Get Envato Tuts+ updates, news, surveys & offers.

Email Address

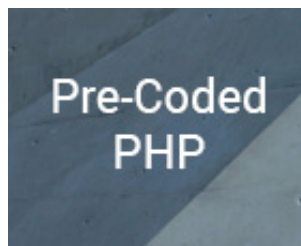
**Subscribe**

[Privacy Policy](#)



From logo design to video animation, web development to website copy; expert designers developers and digital talent are ready to complete your projects.

[Order out Envato Studio's services](#)



Build anything from social networks to file upload systems. Build faster with pre-coded PHP scripts.

[Browse PHP on CodeCanyon](#)

[Follow Envato Tuts+](#)

© 2016 Envato Pty Ltd. Trademarks and brands are the property of their respective owners.