iOS Development

Web Academy

Study Guide

v1.01

iOS Development guideline

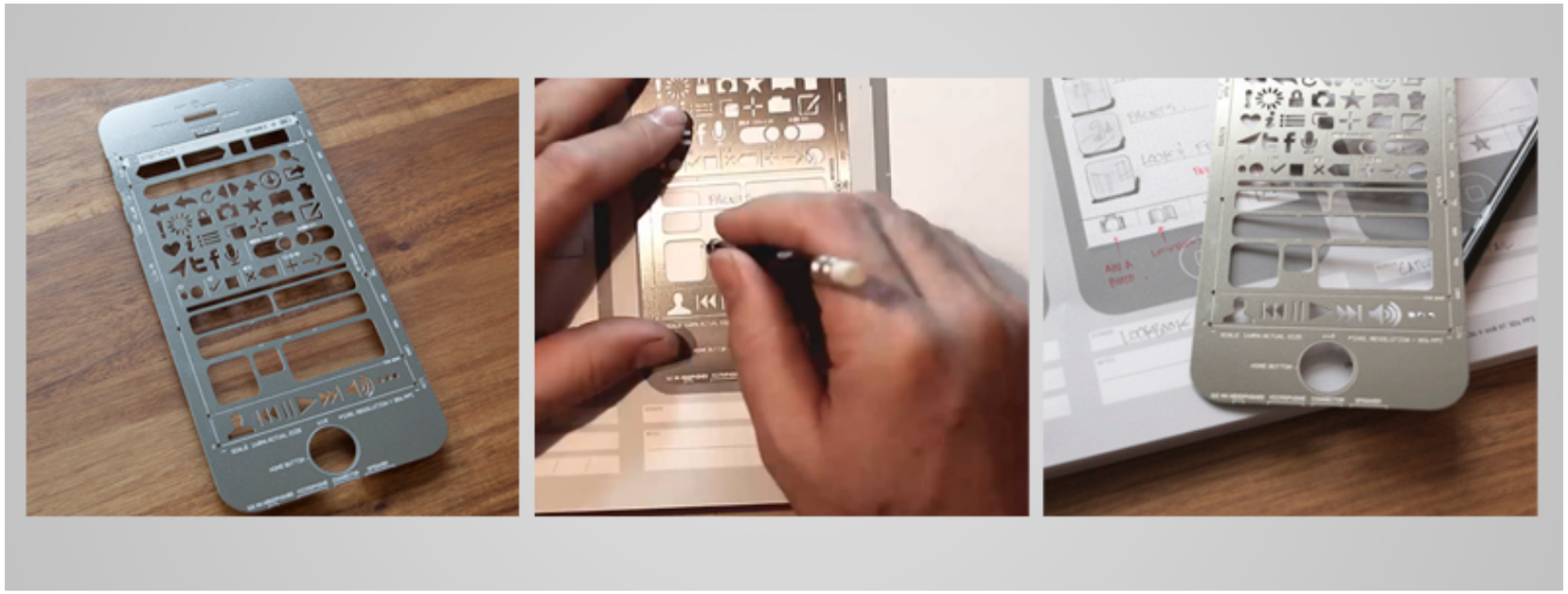
Content

Что такое проект?

Результатом курса есть создание и защита собственного проекта.
Для более быстрого создания прототипа мобильного приложения предлагаю вашему вниманию большой обзор доступных инструментов для прототипирования мобильных приложений.
Прототипирование — это создание макета, модели будущего приложения для того, чтобы определить правильность структуры приложения, его функциональности и, в целом, концепции приложения. Если приложение разрабатывается по стороннему заказу, клиенту также может показываться прототип для того, чтобы он мог контролировать и вносить корректировки в свое приложение.

1. Первый и самый любимый инструмент дизайнеров — это бумага и любой пишущий инструмент (карандаш, ручка, маркеры). Он позволяет накидать структуру приложения и сделать первые наброски интерфейса максимально быстро. Для ускорения используется Скетчпад (SketchPad, Скетчбук, sketchbook) - это блокнот разлинованый макетами телефонов разных платформ.

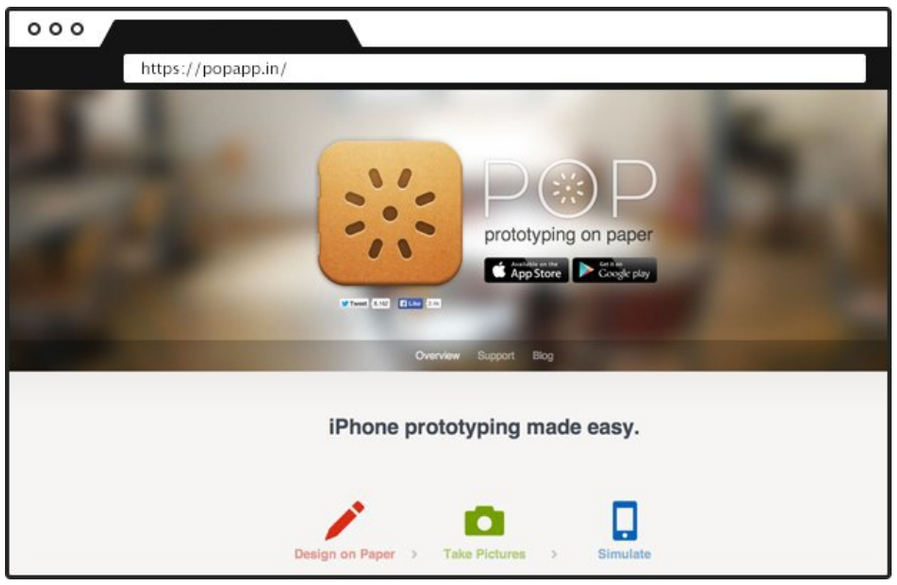
2. Лекала - удобная, должно быть, вещь, но в продаже их найти очень трудно.



3. Штампы - в магазинах тоже отсутствуют, но можно заказать у компании по изготовлению печатей, или сделать самому из куска резины, если руки достаточно прямые. К сожалению, чернила – вещь довольно маркая, поэтому лучше все-таки не пытаться экономить и купить или распечатать скетчпад.

Программные решения для создания быстрых прототипов

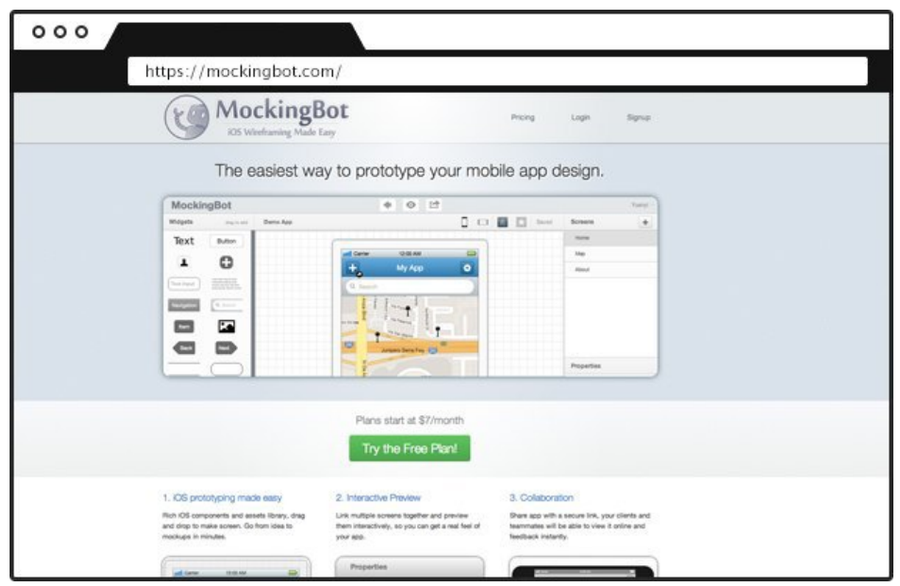
POP



Приложение Prototyping on Paper позволяет использовать для прототипирования самые простые и доступные инструменты — карандаш, бумагу и айфон. Нарисуйте эскизы от руки, сфотографируйте их — сервис автоматически отрегулирует яркость и размер, позволит сделать симулятор вашего будущего приложения. В POP можно создать систему линков и с её помощью проследить, как приложение будет реагировать на нажатие той или иной кнопки. Также POP позволяет делиться созданным прототипом с коллегами и друзьями, собирать фидбэк.

MockingBot

Редактор для создания интерактивных мокапов — моделей дизайна в натуральную величину — в вебе. Бесплатно можно отрисовать одно приложение, за $7 в месяц — три, за $15 в месяц — неограниченное количество. Сервис представляет собой настоящий конструктор графических элементов, которые могут понадобиться при создании дизайна приложения для AppStore. При желании можно найти ряд похожих по функционалу редакторов, например Balsamiq или MockingBird.

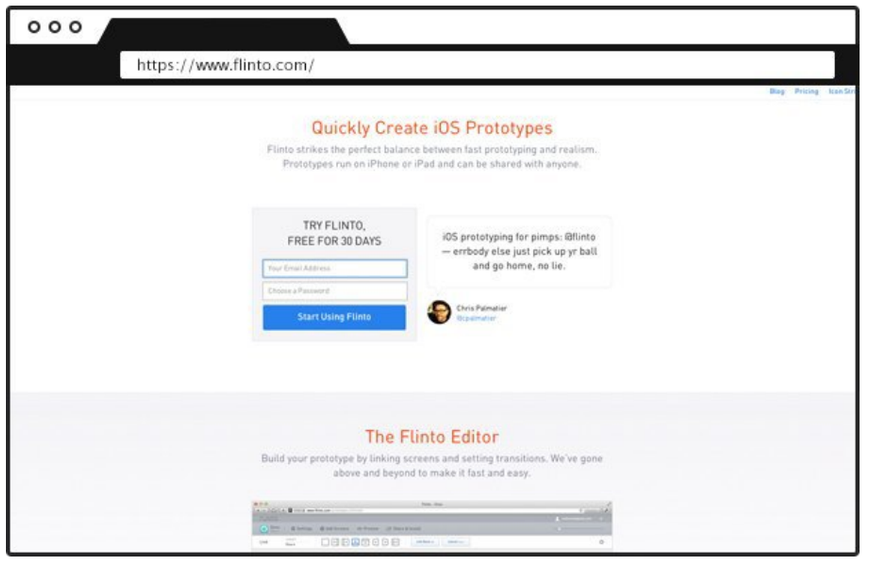


AppCooker

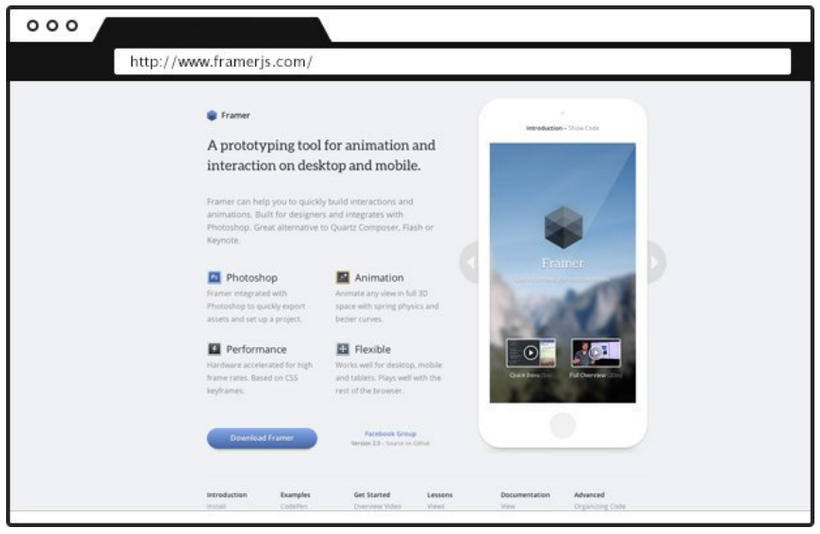
Приложение позволяет создавать мокап прямо на айпаде. Рисовать эскизы можно пальцем, также существует возможность выбора графических элементов из библиотеки. Некоторые элементы линкуются и при желании можно создать подобие работающего приложения для демонстрации. AppCooker предлагает много функций для любителей поковыряться в деталях. Например, можно сделать несколько разных вариантов иконки или поиграть с калькулятором цены приложения. Стоит недёшево — $13,99. Этот и похожие сервисы идеально подходят для прототипирования в условиях долгого перелёта.

Flinto

Редактор для создания интерактивного прототипа. Для того чтобы им пользоваться, необходимо иметь готовые макеты. Впрочем, загрузить в приложение можно не только хорошо проработанные мокапы, но и совсем сырые скетчи. Оно легко и удобно расставляет линки между разными элементами и загружает прототип прямо на рабочий стол вашего устройства для более удобной демонстрации. Стоимость подписки на месяц для индивидуальных разработчиков — $10, для команды — $25 с каждого участника. У приложения имеются более дорогие и навороченные аналоги, а также альтернативы для десктопа: Prototype и Briefs (Mac only).



Framer



Продукт, который больше подойдёт разработчикам, предназначен в том числе для прототипирования простой и сложной анимации. С помощью Framer можно создавать интерактивные приложения на десктопе. Приложение интегрируется в программу Photoshop, то есть для начала работы вам достаточно загрузить приложение, открыть файлы проекта в фотошопе и нажать кнопку Run в Framer. Перед вами появится папка проекта, здесь же можно писать код и наблюдать за тем, как ваш прототип становится неотличим от настоящего работающего приложения.

Учебные ресурсы iOS

Самая полная коллекция высококачественных обучающих сайтов и статей о iOS разработке

Обучающие сайты:

Ray Wenderlich (eng)
Cocoa Dev Central (eng)
Big Nerd Ranch (eng)
Cocoa is my girlfriend (eng)
AppCoda (eng)
Tuts Plus iOS (eng)
iOS-Blog (eng)
Cocoa with love (eng)
objc (eng)
Mike Ash (eng)
NSHipster (eng)
Natasha the robot (eng)
ACSII WWDC (eng)

A Better Way to Learn Swift (eng)
Use Your Loaf (eng)
Code with Chris (eng)
Tutorials Point (eng)
https://learnxinyminutes.com/docs/objective-c/ (eng)
Little bits of cocoa (eng)
James Quave (eng)
iOS Dev Tips (eng)
Think and Build (eng)
Swift Book (ru)

Лучшие статьи по конкретным темам:

Архитектура приложений

- Viper - objc.io
- [8th Light] (https://8thlight.com/blog/uncle-bob/2011/11/22/Clean-Architecture.html)
- Aspect-Oriented Programming

GCD

- GCD: Summary, Syntax & Best Practices - Matt Nunogawa
- Using GCD and Blocks effectively - Mike Nachbaur
- Asynchronous Operations in iOS with Grand Central Dispatch - Jeffrey Sambells

Управление памятью

- Memory Management with Objective-C / Cocoa / iPhone - Memo Akten
- Memory Management - RyPress
- An In-Depth Look At Manual Memory Management In Objective-C - Tom Dalling

Блоки

- Understanding Objective-C Blocks - Intertech
- http://goshdarnblocksyntax.com/

Core Data

- Exploring all the different core data concurrency configurations
- A real guide to core data concurreny
- New in core data and iOS8 batch updating
- Maintaining a silky smooth UI with core data

Как не закрешить приложение

- http://inessential.com/hownottocrash

Понимание UI и Touch Events

- Hit-Testing in iOS
- Understanding scrollviews

Как общаться с гуглом?

В данном курсе, помимо материала который дает тренер, Вам нужно будет искать необходимую информацию самому.

Важно правильно сформулировать запрос, чтобы результаты поиска вполне удовлетворили ваши информационные потребности.

Советы о том, как правильно «гуглить»:

1. Чтобы найти точную фразу или форму слова — возьмите ее в кавычки.

«дивлюсь я на небо»

2. Чтобы найти слово из цитаты, которое вы забыли — поставьте всю цитату в кавычки, а пропущенное слово отметьте звездочкой. Готово!

«Мама * давай»

3. Как найти любое слово из нескольких перечисленных. Надо прописать все варианты через вертикальный слэш |.

гостиница | отель | хостел

4. Чтобы найти слова в пределах одного предложения, используйте ampersand — знакомый всем значок &.

Лучшие музеи мира & Париж

5. Если нужно найти документ с определенным словом, поставьте перед ним +, только без пробела.

Николаевский зоопарк +тигр

6. Иногда необходимо исключить слово из поиска. Для этого перед нужными словами поставьте минус.

Наполеон -торт

7. Ища информацию на определенном сайте, вставьте слово «site» и поставьте после него двоеточие.

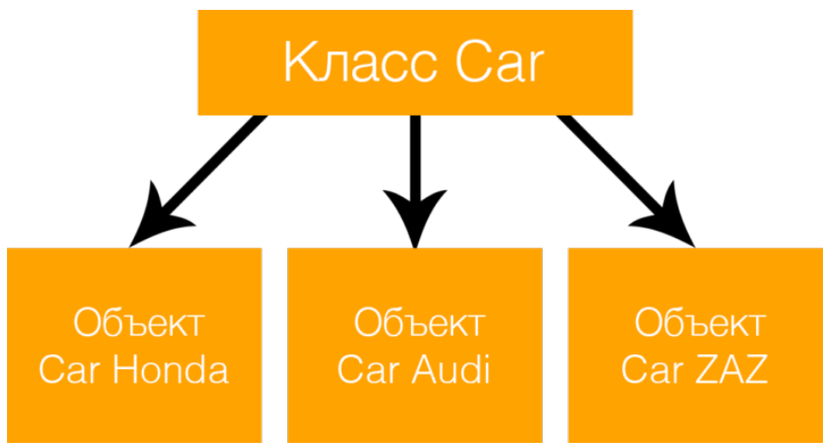
8. Чтобы найти определенный тип документа, впишите оператор «mime», поставьте двоеточие и тип нужного документа.
заявление на загранпаспорт mime:pdf
9. С помощью оператора «lang» можно искать информацию на определенном языке. После него — поставить двоеточие и написать язык. Например, если это русский — то поставить ru, английский — en и т. д.
void glClearColor lang:ru
10. Для поиска синонимов используйте символ ~ перед словом. Таким образом откроются все ссылки на страницы со словами, похожими на ваше, при этом не содержащие это же слово.
~необычные шарфы -необычные
11. Как прогуглить значение слова? Ввести в поиск define: и узнать.
define: коучинг
12. Если вам надо узнать величину или курс валют, просто введите запрос в строке.
1 кг в фунтах
13. Точное время по городу найдется с помощью слов «время» и «город» в строке поиска.
time Киев
14. Калькулятор. Чтобы подсчитать уравнение, вбейте пример в поиск и получите результат.
695+583
15. Прогноз погоды по городу на несколько дней найдется при вводе слов «weather» и «город».
weather Киев

Также для поиска необходимой информации можно использовать такие ресурсы, как :
https://stackoverflow.com
https://community.web-academy.com.ua
где Вам обязательно дадут правильный ответ на поставленный вопрос!

Словарь iOS Development терминов (swift)

Объект — некоторая сущность в виртуальном пространстве, обладающая определённым состоянием и поведением, имеющая заданные значения свойств (атрибутов) и операций над ними (методов). Как правило, при рассмотрении объектов выделяется то, что объекты принадлежат одному или нескольким классам, которые определяют поведение (являются моделью) объекта. По сути, объект (он же экземпляр класса) — класс после инициализации, сущность в памяти, которая имеет своё индивидуальное состояние атрибутов. Объекты инициализируются и хранятся в памяти, классы напротив, не инициализируются.

Класс - описание структуры объекта и методов работы с ним. Swift это язык унаследованный от языка C и Objective-C, он является и процедурным языком, и объектно-ориентированным языком. Класс (Class) является понятием объектно-ориентированного языка. Класс имеет атрибуты и методы, в действительности метод (method) понимается как функция класса. Из класса вы можете создать объекты. Swift использует ключевое слово ***class*** чтобы объявить класс.



Метод - функция работающая с объектом.
Свойства - связывают значения с определённым классом, структурой или перечислением. Свойства хранения содержат значения константы или переменной как часть екзепляра, в то время вычисляемые свойства считают значения, а не хранят их. Вычисляемые свойства обеспечиваются классами, стрктурами или перечислениями. Свойства хранения обеспечиваются только классами иили струтурами. Они бывают нескольких видов:

- **Stored Properties — хранимые свойства:**

Обычные свойства, могут быть как изменяемыми **var**, так и константами **let**

- **Lazy Stored Properties — ленивые свойства:**

Вычисляются в момент первого обращения, помечаются как lazy, не могут быть константой. Применяются когда создание класса требует больших затрат и это используется редко

- **Computed Properties — вычисляемые свойства**:

Свойства, значения которых вычисляется заново при каждом обращении,
path — вычисляемое свойство, оно имееет get — позволяющее получить его значение,
и **set** — выставить

```
class Distance {
    var time = 1
    var velocity = 0
    var path : Int {
        get { return time * velocity }
        set (newPath) {
            velocity = newPath / time
        }
    }
}
```

- **Property Observers — наблюдатели за свойством**:

Позволяет прикреплять наблюдателя к хранимым свойствам, включая наследуемые, кроме ленивых (**lazy**)

Массивы
Полная форма записи массива в Swift пишется Array<Element>, где Element это тип значения, который может хранить массив.
Вы можете также написать массив в сокращенной форме как [Element].
Хотя две формы функционально идентичны, краткая форма является предпочтительной.

Замыкания
Замыкания - это самодостаточные блоки с определенным функционалом, которые могут быть переданы и использованны в вашем коде. Замыкания в Swift похожи на блоки в C и Objective-C, и лямбды в других языках программирования.
Замыкания могут захватывать и хранить отношения для любых констант и переменных из контекста, в котором они объявлены. Эта процедура известна как заключение этих констант и переменных, отсюда и название «замыкание». Swift выполняет всю работу с управлением памятью при захвате за вас.

Наследование
Класс может *наследовать* методы, свойства и другие характеристики другого класса. Когда один класс наследует у другого класса, то наследующий класс называется *подклассом*, класс у которого наследуют - *суперклассом*. Наследование - фундаментальное поведение, которое отделяет классы от других типов Swift.
Классы в Swift могут вызывать или получать доступ к методам, свойствам, индексам. принадлежащим их суперклассам и могут предоставлять свои собственные переписаные версии этих методов, свойств, индексов для усовершенствования или изменения их поведения.
Классы так же могут добавлять наблюдателей свойств к наследованным свойствам для того, чтобы быть в курсе, когда происходит смена значений свойства.

Инициализация

Инициализация - это подготовительный процесс экземпляра класса, структуры или перечисления для дальнейшего использования. Этот процесс включает в себя установку начальных значений для каждого свойства хранения этого экземпляра и проведение любых настроек или инициализации, которые нужны до того, как экземпляр будет использовтся.

Вы реализуете инициализацию, определяя *инициализаторы,* которые схожи со специальными методами, которые вызываются для создания экземпляра опрделенного типа. В отличии от инициализаторов в Objective-C, инициализаторы в Swift не возращают значения. Основная роль инициализаторов - убедится в том, что новый экземпляр типа правильно инициализирован до того, как будет использован в первый раз.

Деинициализация

Деинициализатор вызывается сразу после освобождения экзепляра класса. Вы пишите деинициализаторы с ключевого слова **deinit**, аналогично как вы пишите инициализаторы с ключевого слова **init**. Деинициализаторы доступны только для классовых типов.

Приведение типов

Привидение типов - это способ проверить тип экземпляра и/или способ обращения к экземпляру так, как если бы он был экземпляром суперкласса или подкласса откуда-либо из своей собственной классовой иерархии.

Приведение типов в Swift реализуется с помощью операторв **is** и **as**. Эти два оператора предоставляют простой и выразительный способ проверки типа значения или преобразование значения к другому типу.

WEB ACADEMY

PROGRAMMING COURSES

(044) 233 - 15 - 22, (063) 233 - 15 - 22