While Crashlytics gives you powerful crash reporting, with one additional click you can enable real-time analytics that help you understand what's happening in your app. Fabric's analytics engine provides insights into your core goals, such as growth, retention, and engagement.

Crashlytics also includes Crashlytics Beta, a service that lets you easily distribute pre-release iOS and Android apps to testers so you can get quick feedback.

Install Crashlytics via CocoaPods

use\_frameworks!

pod 'Fabric'

pod 'Crashlytics'

## Add a Run Script Build Phase

## Add Your API Key

## Initialize Your Kit

Crashlytics.sharedInstance().crash()

User Information

Log user information when your app crashes.

## Provide user info

If you log user identifiers with the Crashlytics SDK while your app is running, we will show them in crash reports on your dashboard to help you debug.

func application(\_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplicationLaunchOptionsKey: Any]?) -> Bool {

Fabric.with([Crashlytics()])

// TODO: Move this to where you establish a user session

self.logUser()

return true

}

func logUser() {

// TODO: Use the current user's information

// You can call any combination of these three methods

Crashlytics.sharedInstance().setUserEmail("user@fabric.io")

Crashlytics.sharedInstance().setUserIdentifier("12345")

Crashlytics.sharedInstance().setUserName("Test User")

}

## App Routing

* Find and fix bugs faster and easier.
* Change existing behaviors with confidence.
* Add new features easily.
* Write shorter methods with single responsibility.
* Decouple class dependencies with established boundaries.
* Extract business logic from view controllers into interactors.
* Build reusable components with workers and service objects.
* Write factored code from the start.
* Write fast and maintainable unit tests.
* Have confidence in your tests to catch regression.
* Apply what you learn to new and existing projects of any size.
* VIPER is an application of [Clean Architecture](http://blog.8thlight.com/uncle-bob/2012/08/13/the-clean-architecture.html) to iOS apps. The word VIPER is a backronym for View, Interactor, Presenter, Entity, and Routing. Clean Architecture divides an app’s logical structure into distinct layers of responsibility. This makes it easier to isolate dependencies (e.g. your database) and to test the interactions at the boundaries between layers: *Routing*: contains navigation logic for describing which screens are shown in which order.

In VIPER, the responsibility for Routing is shared between two objects: the Presenter, and the wireframe. A wireframe object owns the UIWindow, UINavigationController, UIViewController, etc. It is responsible for creating a View/ViewController and installing it in the window.

Kotlin is a modern and statically typed programming language targeting the JVM, Android, JavaScript & Native that will boost your productivity and increase your developer happiness. We made [Kotlin](https://developer.android.com/kotlin/) a first class language on Android in 2017. This month we had over 118,000 new projects using Kotlin started in Android Studio. [Android Studio](https://developer.android.com/studio/) is our official IDE for Android development. The code editor helps you write better code, work faster, and be more productive by offering advanced code completion, refactoring, and code analysis. As you type, Android Studio provides suggestions in a dropdown list. Simply press Tab to insert the code. **April 2019** Android Studio 3.4 is a major release that includes a variety of new features and improvements. **Firebase** is a [mobile](https://en.wikipedia.org/wiki/Mobile_application) and [web application](https://en.wikipedia.org/wiki/Web_application) development platform developed by Firebase, Inc. in 2011, then acquired by [Google](https://en.wikipedia.org/wiki/Google) in 2014.[[5]](https://en.wikipedia.org/wiki/Firebase#cite_note-5) As of October 2018, the Firebase platform has 18 products,[[6]](https://en.wikipedia.org/wiki/Firebase#cite_note-6) which are used by 1.5 million apps.[[7]](https://en.wikipedia.org/wiki/Firebase#cite_note-7)

Firebase's first product was the Firebase Real-time Database, an API that synchronizes application data across iOS, Android, and Web devices, and stores it on Firebase's cloud. The product assists software developers in building real-time, collaborative applications.

Firebase Analytics is a cost-free app measurement solution that provides insight into app usage and user engagement.

Firebase is your server, your API and your datastore, all written so generically that you can modify it to suit most needs.When you connect your app to Firebase, you’re not connecting through normal HTTP. You’re connecting through a WebSocket. WebSockets are [much, much faster than HTTP](http://www.websocket.org/quantum.html" \t "_blank). You don’t have to make individual WebSocket calls, because one socket connection is plenty. All of your data syncs automagically through that single WebSocket as fast as your client’s network can carry it. Firebase sends you new data as soon as it’s updated. When your client saves a change to the data, all connected clients receive the updated data almost instantly.

Firebase is platform which allow to build web and mobile applications without server side programming language. You can store users data on its real-time database which sync data among users data in no time.

Firebase is a google product which offers so many useful features. Like

**Real time database**,

**Push notification**,

**Firebase Analytics**,

**Firebase Authentication**,

**Firebase Cloud Messaging**,

**Firebase Storage**,

**Firebase Hosting**,

**Firebase Test Lab For Android**,

**Firebase Crash reporting**,

**Firebase Notification**,   
**Firebase App Indexing**,

**Firebase Dynamic link**, **Firebase Invites**,   
**and Firebase Adwords**.

Google Analytics - Collect, configure, and analyze your data to reach the right audience.

Google smart home lets users control your connected devices through the [Google Home app](https://play.google.com/store/apps/details?id=com.google.android.apps.chromecast.app) and the [Google Assistant](https://assistant.google.com/), which is available on more than 1 billion devices, like smart speakers, phones, cars, TVs, headphones, watches, and more.

[Google AdMob](https://admob.google.com/home/?utm_source=firebase&utm_medium=et&utm_campaign=firebase-docs&utm_content=2017Q1) is a mobile advertising platform that you can use to generate revenue from your app. AdMob helps you monetize your mobile app through in-app advertising. Ads can be displayed as banner, interstitial, video, or native ads—which are seamlessly added to platform native UI components. Before you can display ads within your app, you'll need to create an AdMob account and activate one or more ad unit IDs. This is a unique identifier for the places in your app where ads are displayed. AdMob uses the Google Mobile Ads SDK which helps app developers gain insights about their users and maximize ad revenue. To do so, the default integration of the Mobile Ads SDK collects information such as device information and publisher-provided location information.

With ARCore, build new augmented reality experiences that seamlessly blend the digital and physical worlds. Transform the way people play, shop, learn, create, and experience the world together—at Google scale.

<https://www.udemy.com/iot-turn-a-light-on-with-your-iphone/>

Real environment modified by computer-generated objects is present in many areas, from aviation to gaming, we’re just not aware of it as users. Have you tried to catch Pokemon in recent years or to fit furniture in your room via IKEA app

Augmented reality is the technology that expands our physical world, adding layers of digital information onto it. Unlike Virtual Reality (VR), AR does not create the whole artificial environments to replace real with a virtual one. AR appears in direct view of an existing environment and adds sounds, videos, graphics to it.

Virtual Reality (VR) is the use of computer technology to create a simulated environment. Unlike traditional user interfaces, VR places the user inside an experience. Instead of viewing a screen in front of them, users are immersed and able to interact with 3D worlds.

Augmented Reality simulates artificial objects in the real environment; Virtual Reality creates an artificial environment to inhabit.

In Augmented Reality, the computer uses sensors and algorithms to determine the position and orientation of a camera. AR technology then renders the 3D graphics as they would appear from the viewpoint of the camera, superimposing the computer-generated images over a user’s view of the real world.

In Virtual Reality, the computer uses similar sensors and math. However, rather than locating a real camera within a physical environment, the position of the user’s eyes are located within the simulated environment. If the user’s head turns, the graphics react accordingly. Rather than compositing virtual objects and a real scene, VR technology creates a convincing, interactive world for the user.

# Google Cloud Storage

**Unified object storage for developers and enterprises.** Cloud Storage allows world-wide storage and retrieval of any amount of data at any time. You can use Cloud Storage for a range of scenarios including serving website content, storing data for archival and disaster recovery, or distributing large data objects to users via direct download.

Big data is an evolving term that describes a large volume of [structured](https://whatis.techtarget.com/definition/structured-data), [semi-structured](https://whatis.techtarget.com/definition/semi-structured-data) and [unstructured data](https://searchbusinessanalytics.techtarget.com/definition/unstructured-data) that has the potential to be mined for information and used in [machine learning](https://searchenterpriseai.techtarget.com/definition/machine-learning-ML) projects and other advanced analytics applications.

users watch *4.15 million YouTube videos*, send *456,000 tweets* on Twitter, post *46,740 photos* on Instagram and there are *510,000 comments* posted and *293,000 statuses* updated on Facebook!

Just imagine the huge chunk of data that is produced with such activities. This constant creation of data using social media, business applications, telecom and various other domains is leading to the formation of Big Data.

The Google Contacts API allows client applications to view and update a user's contacts. Contacts are stored in the user's Google Account; most Google services have access to the contact list.

Your client application can use the Google Contacts API to create new contacts, edit or delete existing contacts, and query for contacts that match particular criteria

Cloud Speech-to-Text enables easy integration of Google speech recognition technologies into developer applications. Send audio and receive a text transcription from the Speech-to-Text API service.

Flutter is Google’s mobile UI framework for crafting high-quality native interfaces on iOS and Android in record time. Flutter works with existing code, is used by developers and organizations around the world, and is free and open source.

The Places SDK for iOS allows you to build location-aware apps that respond contextually to the local businesses and other places near the device. This means you can build rich apps based on places that mean something to the user.

**Notice:** Version 2.7.0 of the Places SDK for iOS is deprecated as of January 29, 2019, and will be turned off on July 29, 2019. **A new version of the Places SDK for iOS is now available.** We recommend updating to the new version as soon as possible. For details, see the [migration guide](https://developers.google.com/places/ios-sdk/client-migration).

reCAPTCHA is a free service that protects your site from spam and abuse. It uses advanced risk analysis techniques to tell humans and bots apart.

Get users into your apps quickly and securely, using a registration system they already use and trust—their Google account.

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.

The Time Zone API provides time offset data for locations on the surface of the earth. You request the time zone information for a specific latitude/longitude pair and date. The API returns the name of that time zone, the time offset from UTC, and the daylight savings offset.

Add YouTube features to your application, including the ability to upload videos, create and manage playlists, and more.

Use the API to search for videos matching specific search terms, topics, locations, publication dates, and much more. The APIs search.listmethod also supports searches for playlists and channels.

Add YouTube functionality to your sites and apps.

Use an embedded player to play videos directly in your app and customize the playback experience.

Retrieve viewing statistics, popularity metrics, and more for YouTube videos and channels.

The YouTube Reporting and YouTube Analytics APIs let you retrieve YouTube Analytics data to automate complex reporting tasks, build custom dashboards, and much more.

* The Reporting API supports applications that can retrieve and store bulk reports, then provide tools to filter, sort, and mine the data.
* The Analytics API supports targeted, real-time queries to generate custom reports in response to user interaction.

The YouTube Live Streaming API lets you create, update, and manage live events on YouTube. Using the API, you can schedule events (broadcasts) and associate them with video streams, which represent the actual broadcast content.

The Live Streaming API is actually comprised of components of the YouTube Data API and the YouTube Content ID API. The Data API enables YouTube users to manage their YouTube accounts, while the YouTube Content ID API enables interactions with YouTube's rights management system. However, all of the resources that make up the Live Streaming API are used only to create and manage live events.

Amazon Web Services offers a broad set of global cloud-based products including [compute](https://aws.amazon.com/products/compute/" \t "_blank), [storage](https://aws.amazon.com/products/storage/" \t "_blank), [databases](https://aws.amazon.com/products/databases/" \t "_blank), [analytics](https://aws.amazon.com/products/analytics/), [networking](https://aws.amazon.com/products/networking/" \t "_blank), [mobile](https://aws.amazon.com/mobile/), [developer tools](https://aws.amazon.com/products/developer-tools/" \t "_blank), [management tools](https://aws.amazon.com/products/management/" \t "_blank), [IoT](https://aws.amazon.com/iot/" \t "_blank), [security](https://aws.amazon.com/products/security/) and [enterprise applications](https://aws.amazon.com/enterprise/" \t "_blank). These services help organizations move faster, lower IT costs, and scale.

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers.It provides you with complete control of your computing resources and lets you run on Amazon’s proven computing environment. Amazon EC2 reduces the time required to obtain and boot new server instances to minutes, allowing you to quickly scale capacity, both up and down, as your computing requirements change. Amazon EC2 changes the economics of computing by allowing you to pay only for capacity that you actually use. Amazon EC2 provides developers the tools to build failure resilient applications and isolate them from common failure scenarios.

Amazon S3 provides easy-to-use management features so you can organize your data and configure finely-tuned access controls to meet your specific business, organizational, and compliance requirements. Amazon S3 is designed for 99.999999999% (11 9's) of durability, and stores data for millions of applications for companies all around the world.

Amazon Aurora is a MySQL and PostgreSQL-compatible [relational database](https://aws.amazon.com/relational-database/) built for the cloud, that combines the performance and availability of traditional enterprise databases with the simplicity and cost-effectiveness of open source databases.

Amazon Relational Database Service (Amazon RDS) makes it easy to set up, operate, and scale a relational database in the cloud. It provides cost-efficient and resizable capacity while automating time-consuming administration tasks such as hardware provisioning, database setup, patching and backups. It frees you to focus on your applications so you can give them the fast performance, high availability, security and compatibility they need. Amazon RDS is available on several database instance types - optimized for memory, performance or I/O - and provides you with six familiar database engines to choose from, including [Amazon Aurora](https://aws.amazon.com/rds/aurora/), [PostgreSQL](https://aws.amazon.com/rds/postgresql/), [MySQL](https://aws.amazon.com/rds/mysql/), [MariaDB](https://aws.amazon.com/rds/mariadb/), [Oracle Database](https://aws.amazon.com/rds/oracle/), and [SQL Server](https://aws.amazon.com/rds/sqlserver/).

AWS Lambda lets you run code without provisioning or managing servers. You pay only for the compute time you consume - there is no charge when your code is not running. With Lambda, you can run code for virtually any type of application or backend service - all with zero administration. Just upload your code and Lambda takes care of everything required to run and scale your code with high availability. You can set up your code to automatically trigger from other AWS services or call it directly from any web or mobile app.

Amazon Athena is an interactive query service that makes it easy to analyze data in Amazon S3 using standard SQL. Athena is serverless, so there is no infrastructure to manage, and you pay only for the queries that you run. Athena is easy to use. Simply point to your data in Amazon S3, define the schema, and start querying using standard SQL. Most results are delivered within seconds. With Athena, there’s no need for complex ETL jobs to prepare your data for analysis. This makes it easy for anyone with SQL skills to quickly analyze large-scale datasets.

Amazon CloudSearch is a managed service in the AWS Cloud that makes it simple and cost-effective to set up, manage, and scale a search solution for your website or application. Amazon CloudSearch supports 34 languages and popular search features such as highlighting, autocomplete, and geospatial search. With Amazon CloudSearch, you can quickly add rich search capabilities to your website or application. You don't need to become a search expert or worry about hardware provisioning, setup, and maintenance. With a few clicks in the [AWS Management Console](https://aws.amazon.com/console/), you can create a search domain and upload the data that you want to make searchable, and Amazon CloudSearch will automatically provision the required resources and deploy a highly tuned search index. You can easily change your search parameters, fine tune search relevance, and apply new settings at any time. As your volume of data and traffic fluctuates, Amazon CloudSearch seamlessly scales to meet your needs.

Amazon EMR provides a managed Hadoop framework that makes it easy, fast, and cost-effective to process vast amounts of data across dynamically scalable Amazon EC2 instances. You can also run other popular distributed frameworks such as [Apache Spark](https://aws.amazon.com/emr/details/spark/), [HBase](https://aws.amazon.com/emr/details/hbase/), [Presto](https://aws.amazon.com/emr/details/presto/), and [Flink](https://aws.amazon.com/blogs/big-data/use-apache-flink-on-amazon-emr/) in EMR, and interact with data in other AWS data stores such as Amazon S3 and Amazon DynamoDB.

Amazon Kinesis makes it easy to collect, process, and analyze real-time, streaming data so you can get timely insights and react quickly to new information. Amazon Kinesis offers key capabilities to cost-effectively process streaming data at any scale, along with the flexibility to choose the tools that best suit the requirements of your application. With Amazon Kinesis, you can ingest real-time data such as video, audio, application logs, website clickstreams, and IoT telemetry data for machine learning, analytics, and other applications. Amazon Kinesis enables you to process and analyze data as it arrives and respond instantly instead of having to wait until all your data is collected before the processing can begin.

Amazon Sumerian lets you create and run virtual reality (VR), augmented reality (AR), and 3D applications quickly and easily without requiring any specialized programming or 3D graphics expertise. With Sumerian, you can build highly immersive and interactive scenes that run on popular hardware such as Oculus Go, Oculus Rift, HTC Vive, HTC Vive Pro, Google Daydream, and Lenovo Mirage as well as Android and iOS mobile devices. For example, you can build a virtual classroom that lets you train new employees around the world, or you can build a virtual environment that enables people to tour a building remotely. Sumerian makes it easy to create all the building blocks needed to build highly immersive and interactive 3D experiences including adding objects (e.g. characters, furniture, and landscape), and designing, animating, and scripting environments. Sumerian does not require specialized expertise and you can design scenes directly from your browser.

Amazon Managed Blockchain is a fully managed service that makes it easy to create and manage scalable blockchain networks using the popular open source frameworks Hyperledger Fabric and Ethereum\*. Blockchain makes it possible to build applications where multiple parties can execute transactions without the need for a trusted, central authority. Today, building a scalable blockchain network with existing technologies is complex to set up and hard to manage. To create a blockchain network, each network member needs to manually provision hardware, install software, create and manage certificates for access control, and configure networking components. Once the blockchain network is running, you need to continuously monitor the infrastructure and adapt to changes, such as an increase in transaction requests, or new members joining or leaving the network. Amazon Managed Blockchain is a fully managed service that allows you to set up and manage a scalable blockchain network with just a few clicks. Amazon Managed Blockchain eliminates the overhead required to create the network, and automatically scales to meet the demands of thousands of applications running millions of transactions. Once your network is up and running, Managed Blockchain makes it easy to manage and maintain your blockchain network. It manages your certificates and lets you easily invite new members to join the network.

Amazon Chime is a communications service that lets you meet, chat, and place business calls inside and outside your organization, all using a single application. With Amazon Chime, you have the flexibility to choose the features that you need for online meetings, video conferencing, and business calling, and pay only when you use them.

Amazon DocumentDB (with MongoDB compatibility) is a fast, scalable, highly available, and fully managed document database service that supports MongoDB workloads. Amazon DocumentDB is designed from the ground-up to give you the performance, scalability, and availability you need when operating mission-critical MongoDB workloads at scale. Amazon DocumentDB implements the Apache 2.0 open source MongoDB 3.6 API by emulating the responses that a MongoDB client expects from a MongoDB server, allowing you to use your existing MongoDB drivers and tools with Amazon DocumentDB.

Amazon DynamoDB is a key-value and document database that delivers single-digit millisecond performance at any scale. It's a fully managed, multiregion, multimaster database with built-in security, backup and restore, and in-memory caching for internet-scale applications. DynamoDB can handle more than 10 trillion requests per day and can support peaks of more than 20 million requests per second.

Amazon Simple Email Service (Amazon SES) is a cloud-based email sending service designed to help digital marketers and application developers send marketing, notification, and transactional emails. It is a reliable, cost-effective service for businesses of all sizes that use email to keep in contact with their customers. You can use our SMTP interface or one of the AWS SDKs to integrate Amazon SES directly into your existing applications. You can also integrate the email sending capabilities of Amazon SES into the software you already use, such as ticketing systems and email clients.

Amazon Neptune is a fast, reliable, fully managed graph database service that makes it easy to build and run applications that work with highly connected datasets. The core of Amazon Neptune is a purpose-built, high-performance graph database engine optimized for storing billions of relationships and querying the graph with milliseconds latency.

AWS Cloud9 gives you the flexibility to run your development environment on a managed Amazon EC2 instance or any existing Linux server that supports SSH. This means that you can write, run, and debug applications with just a browser, without needing to install or maintain a local IDE. The Cloud9 code editor and integrated debugger include helpful, time-saving features such as code hinting, code completion, and step-through debugging. The Cloud9 terminal provides a browser-based shell experience enabling you to install additional software, do a git push, or enter commands.

The AWS IoT Button is a programmable button based on the Amazon Dash Button hardware. This simple Wi-Fi device is easy to configure and designed for developers to get started with [AWS IoT Core](https://aws.amazon.com/iot-platform/), [AWS Lambda](https://aws.amazon.com/lambda/), [Amazon DynamoDB](https://aws.amazon.com/dynamodb/), [Amazon SNS](https://aws.amazon.com/sns/), and many other Amazon Web Services without writing device-specific code.

You can code the button's logic in the cloud to configure button clicks to count or track items, call or alert someone, start or stop something, order services, or even provide feedback. For example, you can click the button to unlock or start a car, open your garage door, call a cab, call your spouse or a customer service representative, track the use of common household chores, medications or products, or remotely control your home appliances.

The button can be used as a remote control for Netflix, a switch for your Philips Hue light bulb, a check-in/check-out device for Airbnb guests, or a way to order your favorite pizza for delivery. You can integrate it with third-party APIs like Twitter, Facebook, Twilio, Slack or even your own company's applications. Connect it to things we haven’t even thought of yet. We can't wait to see what you will build with the AWS IoT Button!

AWS IoT Core is a managed cloud service that lets connected devices easily and securely interact with cloud applications and other devices. AWS IoT Core can support billions of devices and trillions of messages, and can process and route those messages to AWS endpoints and to other devices reliably and securely. With AWS IoT Core, your applications can keep track of and communicate with all your devices, all the time, even when they aren’t connected.

AWS IoT Events is a fully managed service that makes it easy to detect and respond to events from IoT sensors and applications. Events are patterns of data identifying more complicated circumstances than expected, such as changes in equipment when a belt is stuck or motion detectors using movement signals to activate lights and security cameras.

AWS RoboMaker is a service that makes it easy to develop, test, and deploy intelligent robotics applications at scale. RoboMaker extends the most widely used open-source robotics software framework, Robot Operating System (ROS), with connectivity to cloud services. This includes AWS machine learning services, monitoring services, and analytics services that enable a robot to stream data, navigate, communicate, comprehend, and learn. RoboMaker provides a robotics development environment for application development, a robotics simulation service to accelerate application testing, and a robotics fleet management service for remote application deployment, update, and management.

Git Basics

git init Create empty Git repo in specified directory. Run with no arguments to initialize the current directory as a git repository

git clone Clone repo located at onto local machine. Original repo can be located on the local filesystem or on a remote machine via HTTP or SSH.

git config user.name Define author name to be used for all commits in current repo. Devs commonly use --global flag to set config options for current user.

git add Stage all changes in for the next commit. Replace with a to change a specific file

git commit -m "" Commit the staged snapshot, but instead of launching a text editor, use as the commit message.

git status List which files are staged, unstaged, and untracked.

git log Display the entire commit history using the default format. For customization see additional options.

git diff Show unstaged changes between your index and working directory.

Undoing Changes

git revert Create new commit that undoes all of the changes made in , then apply it to the current branch.

git reset Remove from the staging area, but leave the working directory unchanged. This unstages a file without overwriting any changes.

git clean –n Shows which files would be removed from working directory. Use the -f flag in place of the -n flag to execute the clean.

Rewriting Git History

git commit –amend Replace the last commit with the staged changes and last commit combined. Use with nothing staged to edit the last commit’s message

git rebase Rebase the current branch onto . can be a commit ID, a branch name, a tag, or a relative reference to HEAD.

git reflog Show a log of changes to the local repository’s HEAD. Add --relative-date flag to show date info or --all to show all refs.

Git Branches

git branch List all of the branches in your repo. Add a argument to create a new branch with the name .

git checkout -b Create and check out a new branch named . Drop the -b flag to checkout an existing branch.

git merge Merge into the current branch.

Remote Repositories

git remote add Create a new connection to a remote repo. After adding a remote, you can use as a shortcut for in other commands.

git fetch Fetches a specific , from the repo. Leave off to fetch all remote refs.

git pull Fetch the specified remote’s copy of current branch and immediately merge it into the local copy.

git push Push the branch to , along with necessary commits and objects. Creates named branch in the remote repo if it doesn’t exist.

git config

git config git config --global user.name Define the author name to be used for all commits by the current user

git config --global user.email Define the author email to be used for all commits by the current user.

git config --global alias. Create shortcut for a Git command. E.g. alias.glog log --graph --oneline will set git glog equivalent to git log --graph --oneline.

git config --system core.editor Set text editor used by commands for all users on the machine. arg should be the command that launches the desired editor (e.g., vi).

git config Open the global configuration file in a text editor for manual editing. --global --edit

git log

git log Limit number of commits by . E.g. git log -5 will limit to 5 commits

git log –oneline Condense each commit to a single line.

git log –p Display the full diff of each commit.

git log –stat Include which files were altered and the relative number of lines that were added or deleted from each of them.

git log --author= ”” Search for commits by a particular author.

git log --grep=”” Search for commits with a commit message that matches .

git log .. Show commits that occur between and . Args can be a commit ID, branch name, HEAD, or any other kind of revision reference.

git log -- Only display commits that have the specified file.

git log --graph –decorate --graph flag draws a text based graph of commits on left side of commit msgs. --decorate adds names of branches or tags of commits shown.

git dif

git diff HEAD Show difference between working directory and last commit.

git diff --cached Show difference between staged changes and last commit

git reset

git reset Reset staging area to match most recent commit, but leave the working directory unchanged.

git reset --hard Reset staging area and working directory to match most recent commit and overwrites all changes in the working directory

git reset Move the current branch tip backward to , reset the staging area to match, but leave the working directory alone.

git reset --hard Same as previous, but resets both the staging area & working directory to match. Deletes uncommitted changes, and all commits after .

git rebase

git rebase -i Interactively rebase current branch onto . Launches editor to enter commands for how each commit will be transferred to the new base.

git pull --rebase Fetch the remote’s copy of current branch and rebases it into the local copy. Uses git rebase instead of merge to integrate the branches.

git push

git push --force Forces the git push even if it results in a non-fast-forward merge. Do not use the --force flag unless you’re absolutely sure you know what you’re doing.

git push --all Push all of your local branches to the specified remote.

git push --tags Tags aren’t automatically pushed when you push a branch or use the --all flag. The --tags flag sends all of your local tags to the remote repo.

TEMPORARY COMMITS

Temporarily store modified tracked files in order to change branches

git stash Save modified and staged changes

git stash list list stack-order of stashed file changes

git stash pop write working from top of stash stack

git stash drop discard the changes from top of stash stack

Team Foundation Server is Microsoft's "all-in-one" Application Lifecycle Management solution. It contains capabilities for Agile project management, source control, continuous integration (build), continuous delivery (release), manual test case management, and more. It also has SharePoint integration and a data warehouse to facilitate easy reporting. It's been around for about 10 years now, introduced in 2005 as Visual Studio Team System. Since then it's received 5 major versions (2008, 2010, 2012, 2013, and now 2015) with innumerable service packs, patches, and quarterly updates.

Microsoft TFS (Team Foundation Server), is a software tool used for managing a team and their code. It’s a combination of a version control system (VCS), an issue tracker like Jira, and a limited application lifecycle management (ALM) tool.

**Apache Subversion** (often abbreviated **SVN**, after its command name *svn*) is a [software versioning](https://en.wikipedia.org/wiki/Software_versioning) and [revision control](https://en.wikipedia.org/wiki/Revision_control) system distributed as [open source](https://en.wikipedia.org/wiki/Open-source_software) under the [Apache License](https://en.wikipedia.org/wiki/Apache_License).[[2]](https://en.wikipedia.org/wiki/Apache_Subversion#cite_note-2) Software developers use Subversion to maintain current and historical versions of files such as [source code](https://en.wikipedia.org/wiki/Source_code), web pages, and documentation. Its goal is to be a mostly compatible successor to the widely used [Concurrent Versions System](https://en.wikipedia.org/wiki/Concurrent_Versions_System) (CVS).

**Repository:** A repository is the heart of any version control system. It is the central place where developers store all their work. Repository not only stores files but also the history. Repository is accessed over a network, acting as a server and version control tool acting as a client. Clients can connect to the repository, and then they can store/retrieve their changes to/from repository. By storing changes, a client makes these changes available to other people and by retrieving changes, a client takes other people's changes as a working copy.

**Trunk:** The trunk is a directory where all the main development happens and is usually checked out by developers to work on the project.

**Tags** : The tags directory is used to store named snapshots of the project. Tag operation allows to give descriptive and memorable names to specific version in the repository.

**Working copy:** Working copy is a snapshot of the repository. The repository is shared by all the teams, but people do not modify it directly. Instead each developer checks out the working copy. The working copy is a private workplace where developers can do their work remaining isolated from the rest of the team.

**Commit changes:** Commit is a process of storing changes from private workplace to central server. After commit, changes are made available to all the team. Other developers can retrieve these changes by updating their working copy. Commit is an atomic operation. Either the whole commit succeeds or is rolled back. Users never see half finished commit.

## Create Repository:

The repository is a central place where developers store all their work. Repository not only stores files, but also the history about changes. Which means it maintains a history of the changes made in the files.

The 'create' operation is used to create a new repository. Most of the times this operation is done only once. When you create a new repository, your VCS will expect you to say something to identify it, such as where you want it to be created, or what name should be given to the repository.

## Checkout

'Checkout' operation is used to create a working copy from the repository. Working copy is a private workplace where developers do their changes, and later on, submit these changes to the repository.

## Update

As the name suggests, 'update' operation is used to update working copy. This operation synchronizes the working copy with the repository. As repository is shared by all the teams other developers can commit their changes and your working copy becomes older.

### 1. SVN Checkout – Create working copy

Checkout command is used to download sources from SVN repository to working copy. If you want to access files from the SVN server, checkout is the first operation you should perform.

### 2. SVN Commit – Save changes to the repository

Whenever you do changes to the working copy, it will not reflect in SVN server. To make the changes permanent, you need to do SVN commit.

### 3. SVN List – Lists directory entries

svn list is useful when you want to view the content of the SVN repository, without downloading a working copy.

### 4. SVN Add – Add a new file to SVN repository

When you want to add a new file (or directory) to the repository you need to use SVN add command. The repository will have newly added file, only when you do SVN commit. Now let us add a new file called “thegeekstuff” to our repository.

### 5. SVN Delete – Removing a file from repository

SVN delete command deletes an item from the working copy (or repository). File will be deleted from the repository when you do a SVN commit.

### 6. SVN Diff – Display the difference

SVN diff displays the differences between your working copy and the copy in the SVN repository. You can find the difference between two revisions and two paths etc.,

### 7. SVN Status – Status of the working copy

Use svn status command to get the status of the file in the working copy. It displays whether the working copy is modified, or its been added/deleted, or file is not under revision control, etc.

### 8. SVN Log – Display log message

As we discussed in the beginning of this article, SVN remembers every change made to your files and directories. To know all the commits made in a file or directory, use SVN log command.

### 9. SVN Move – Rename file or directory

This command moves a file from one directory to another or renames a file. The file will be moved on your local sandbox immediately (as well as on the repository after committing).

### 10. SVN Update – Update the working copy.

svn update command brings changes from the repository into your working copy. If no revision is specified, it brings your working copy up-to-date with the HEAD revision. Otherwise, it synchronizes the working copy to the revision given in the argument.

## svn propset

[**svn propset**](http://svnbook.red-bean.com/en/1.2/svn.ref.svn.c.propset.html). When you are creating a new file or directory, you generally need to tell the SVN to apply properties to the file in places that have keywords in a special format such as $Date: 2013-05-26 14:01:55 -0500 (Sun, 26 May 2013) $. Note that the keyword value won't appear in the file until you do an **svn commit** (see below).

|  |  |
| --- | --- |
| Create a new depot/repo | svnadmin create |
| Add files to the depot/repo | svn import |
| Request help | svn help |
| Provide information on changelists and changelists’ files | svn log |
| Display information about the current client and server | svn info |
| Combine two different revisions | svn merge |
| Discard changes made to an open file | svn revert <file or directory> |
| Display information about the workspace files | svn status |
| Store files without submitting | svn shelve |
| Display information about the current client and server | svn info |
| Store files without submitting | svn shelve |

The term "open source" refers to something people can modify and share because its design is publicly accessible.

## What is open source software?

## Open source software is software with source code that anyone can inspect, modify, and enhance.

**Control.** Many people prefer open source software because they [have more control](https://opensource.com/life/13/5/tumblr-open-publishing)over that kind of software. They can examine the code to make sure it's not doing anything they don't want it to do, and they can change parts of it they don't like. Users who aren't programmers also benefit from open source software, because they can use this software for any purpose they wish—not merely the way someone else thinks they should.

**Security.** Some people prefer open source software because they consider it more [secure](https://opensource.com/government/10/9/scap-computer-security-rest-us) and stable than proprietary software. Because anyone can view and modify open source software, someone might spot and correct errors or omissions that a program's original authors might have missed. And because so many programmers can work on a piece of open source software without asking for permission from original authors, they can fix, update, and upgrade open source software more [quickly](https://opensource.com/government/13/2/bug-fix-day)than they can proprietary software.

**Open-source software** (**OSS**) is a type of [computer software](https://en.wikipedia.org/wiki/Computer_software) in which [source code](https://en.wikipedia.org/wiki/Source_code) is released under a [license](https://en.wikipedia.org/wiki/Open-source_license) in which the [copyright](https://en.wikipedia.org/wiki/Copyright) holder grants users the rights to study, change, and [distribute the software](https://en.wikipedia.org/wiki/Software_distribution) to anyone and for any purpose.[[1]](https://en.wikipedia.org/wiki/Open-source_software#cite_note-1) Open-source software may be developed in a [collaborative public manner](https://en.wikipedia.org/wiki/Open-source_model). Open-source software is a prominent example of [open collaboration](https://en.wikipedia.org/wiki/Open_collaboration).[[2]](https://en.wikipedia.org/wiki/Open-source_software#cite_note-Open_Collaboration-2)

## Local data model[[edit](https://en.wikipedia.org/w/index.php?title=List_of_version-control_software&action=edit&section=1)]

In the local-only approach, all developers must use the same file system.

### Open source[[edit](https://en.wikipedia.org/w/index.php?title=List_of_version-control_software&action=edit&section=2)]

* [Revision Control System](https://en.wikipedia.org/wiki/Revision_Control_System) (RCS) – stores the latest version and backward deltas for fastest access to the trunk tip[[1]](https://en.wikipedia.org/wiki/List_of_version-control_software#cite_note-wohler1-1)[[2]](https://en.wikipedia.org/wiki/List_of_version-control_software#cite_note-mcvoy-2) compared to SCCS and an improved user interface,[[3]](https://en.wikipedia.org/wiki/List_of_version-control_software#cite_note-3) at the cost of slow branch tip access and missing support for included/excluded deltas.
* [Source Code Control System](https://en.wikipedia.org/wiki/Source_Code_Control_System) (SCCS) – part of [UNIX](https://en.wikipedia.org/wiki/Unix); based on [interleaved deltas](https://en.wikipedia.org/wiki/Interleaved_deltas), can construct versions as arbitrary sets of revisions. Extracting an arbitrary version takes essentially the same time and is thus more useful in environments that rely heavily on [branching](https://en.wikipedia.org/wiki/Branching_(software)) and [merging](https://en.wikipedia.org/wiki/Merge_(revision_control)) with multiple "current" and identical versions.

## Client-server model[[edit](https://en.wikipedia.org/w/index.php?title=List_of_version-control_software&action=edit&section=3)]

In the client-server model, developers use a shared single repository.

### Open source[[edit](https://en.wikipedia.org/w/index.php?title=List_of_version-control_software&action=edit&section=4)]

* [Concurrent Versions System](https://en.wikipedia.org/wiki/Concurrent_Versions_System) (CVS) – originally built on RCS, licensed under the [GPL](https://en.wikipedia.org/wiki/GPL).
  + [CVSNT](https://en.wikipedia.org/wiki/CVSNT) – cross-platform port of CVS that allows case insensitive file names among other changes
  + [OpenCVS](https://en.wikipedia.org/wiki/OpenCVS) – CVS clone under the [BSD license](https://en.wikipedia.org/wiki/BSD_license), with emphasis put on security and source code correctness
* [Subversion](https://en.wikipedia.org/wiki/Subversion_(software)) (SVN) – versioning control system inspired by CVS[[4]](https://en.wikipedia.org/wiki/List_of_version-control_software#cite_note-4)
* [Vesta](https://en.wikipedia.org/wiki/Vesta_(Software_configuration_management)) – build system with a versioning file system and support for distributed repositories

### Proprietary[[edit](https://en.wikipedia.org/w/index.php?title=List_of_version-control_software&action=edit&section=5)]

* [AccuRev](https://en.wikipedia.org/wiki/AccuRev_SCM) – source configuration management tool with integrated issue tracking based on "Streams" that efficiently manages parallel and global development; replication server is also available. Now owned by [Micro Focus](https://en.wikipedia.org/wiki/Micro_Focus).
* [Autodesk Vault](https://en.wikipedia.org/wiki/Autodesk_Vault) – Version control tool specifically designed for [Autodesk](https://en.wikipedia.org/wiki/Autodesk) applications managing the complex relationships between design files such as [AutoCAD](https://en.wikipedia.org/wiki/AutoCAD) and [Autodesk Inventor](https://en.wikipedia.org/wiki/Autodesk_Inventor).
* [CADES](https://en.wikipedia.org/wiki/CADES) - Designer productivity and version control system by [International Computers Limited](https://en.wikipedia.org/wiki/International_Computers_Limited).
* [Dimensions CM](https://en.wikipedia.org/wiki/Dimensions_CM) - [software change and configuration management](https://en.wikipedia.org/wiki/Software_configuration_management) system developed by [Micro Focus](https://en.wikipedia.org/wiki/Micro_Focus_International), formerly [Serena Software](https://en.wikipedia.org/wiki/Serena_Software), that includes [revision control](https://en.wikipedia.org/wiki/Revision_control).
* [Helix Core](https://en.wikipedia.org/wiki/Perforce_Software#Helix_Core), formerly Perforce Helix - for large scale development environments
* [IBM Configuration Management Version Control (CMVC)](https://en.wikipedia.org/wiki/IBM_Configuration_Management_Version_Control_(CMVC)) – version control system, no longer available.
* [IBM Rational ClearCase](https://en.wikipedia.org/wiki/IBM_Rational_ClearCase) – [SCC compliant](https://en.wikipedia.org/wiki/SCC_compliant) configuration management system by [IBM](https://en.wikipedia.org/wiki/IBM) [Rational Software](https://en.wikipedia.org/wiki/Rational_Software)
* [IBM Rational Synergy](https://en.wikipedia.org/wiki/Telelogic_Synergy) – SCC compliant integrated change management and task-based configuration management system, proprietary of IBM.
* [IBM Rational Team Concert](https://en.wikipedia.org/wiki/IBM_Rational_Team_Concert) – Collaboration and application lifecycle management platform by [IBM](https://en.wikipedia.org/wiki/IBM) [Rational Software](https://en.wikipedia.org/wiki/Rational_Software)
* [IC Manage](https://en.wikipedia.org/wiki/IC_Manage) Global Design Platform (GDP) – design data management for IC design and [Perforce](https://en.wikipedia.org/wiki/Perforce) infrastructure support.
* [Panvalet](https://en.wikipedia.org/wiki/Panvalet) - Around since the 1970s, source and object control for IBM mainframe computers.
* [PTC Integrity](https://en.wikipedia.org/wiki/PTC_Integrity) (Formerly MKS Integrity).
* [PVCS](https://en.wikipedia.org/wiki/PVCS) – originally Polytron Version Control System, developed by Don Kinzer at [Polytron](https://en.wikipedia.org/wiki/Polytron_(software)), first released in 1985. Now owned by [Micro Focus](https://en.wikipedia.org/wiki/Micro_Focus).
* [Quma Version Control System](https://en.wikipedia.org/wiki/QVCS)
* [Razor (configuration management)](https://en.wikipedia.org/wiki/Razor_(configuration_management)), integrated suite from Visible Systems
* [StarTeam](https://en.wikipedia.org/wiki/StarTeam) – coordinates and manages software delivery process by [Micro Focus](https://en.wikipedia.org/wiki/Micro_Focus), formerly [Borland](https://en.wikipedia.org/wiki/Borland); centralized control of digital assets and activities
* [Surround SCM](https://en.wikipedia.org/wiki/Surround_SCM) – version control tool by [Seapine Software](https://en.wikipedia.org/wiki/Seapine_Software).
* [TeamCity](https://en.wikipedia.org/wiki/TeamCity) - Java-based build management and continuous integration server from [JetBrains](https://en.wikipedia.org/wiki/JetBrains)
* [Team Foundation Server](https://en.wikipedia.org/wiki/Team_Foundation_Server) (TFS) - Development software by [Microsoft](https://en.wikipedia.org/wiki/Microsoft) which includes revision control.
* TeamWork – version control tool by [DBmaestro](https://en.wikipedia.org/wiki/DBMaestro); oriented to [DBMs](https://en.wikipedia.org/wiki/Database)
* [Vault](https://en.wikipedia.org/wiki/Vault_(revision_control_system)) – version control tool by SourceGear (First installation can be used for free)
* [Visual SourceSafe](https://en.wikipedia.org/wiki/Visual_SourceSafe) – version control tool by [Microsoft](https://en.wikipedia.org/wiki/Microsoft); oriented toward small teams
* [Visual Studio Team Services](https://en.wikipedia.org/wiki/Visual_Studio_Team_Services) (VSTS) - Services for teams to share code, track work, and ship software for any language by [Microsoft](https://en.wikipedia.org/wiki/Microsoft)

## Distributed model[[edit](https://en.wikipedia.org/w/index.php?title=List_of_version-control_software&action=edit&section=6)]

In the [distributed approach](https://en.wikipedia.org/wiki/Distributed_revision_control), each developer works directly with his or her own local repository, and changes are shared between repositories as a separate step.

### Open source[[edit](https://en.wikipedia.org/w/index.php?title=List_of_version-control_software&action=edit&section=7)]

* [ArX](https://en.wikipedia.org/wiki/ArX_(revision_control)) – written by Walter Landry, started as a fork of GNU arch, but has been completely rewritten
* [Bazaar](https://en.wikipedia.org/wiki/Bazaar_(software)) – written in [Python](https://en.wikipedia.org/wiki/Python_(programming_language)), originally by Martin Pool and sponsored by [Canonical](https://en.wikipedia.org/wiki/Canonical_Ltd.); decentralised, and aims to be fast and easy to use; can losslessly import Arch archives
* [BitKeeper](https://en.wikipedia.org/wiki/BitKeeper) – was used in [Linux kernel](https://en.wikipedia.org/wiki/Linux_kernel) development (2002 – April 2005) until its license was revoked for breach of contract. It was open-sourced in 2016 in an attempt to broaden its appeal again.
* [Codeville](https://en.wikipedia.org/wiki/Codeville) – written in [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) originally by Ross Cohen; uses an innovative merging algorithm
* [Darcs](https://en.wikipedia.org/wiki/Darcs) – written in [Haskell](https://en.wikipedia.org/wiki/Haskell_(programming_language)) and originally developed by David Roundy; can keep track of inter-patch dependencies and automatically rearrange and "cherry-pick" them using a "theory of patches"
* [DCVS](https://en.wikipedia.org/wiki/Distributed_Concurrent_Versions_System) – decentralized and CVS-based
* [Fossil](https://en.wikipedia.org/wiki/Fossil_(software)) – written by [D. Richard Hipp](https://en.wikipedia.org/wiki/D._Richard_Hipp) for [SQLite](https://en.wikipedia.org/wiki/SQLite); distributed revision control, wiki, and bug-tracking (all-in-one solution) with console and web interfaces. Single portable executable and single repository file.
* [Git](https://en.wikipedia.org/wiki/Git) – written in a collection of Perl, C, and various shell scripts, designed by [Linus Torvalds](https://en.wikipedia.org/wiki/Linus_Torvalds) based on the needs of the [Linux kernel](https://en.wikipedia.org/wiki/Linux_kernel) project; decentralized, and aims to be fast, flexible, and robust
* [GNU arch](https://en.wikipedia.org/wiki/GNU_arch)
* [Mercurial](https://en.wikipedia.org/wiki/Mercurial_(software)) – written in [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) as an Open Source replacement to [BitKeeper](https://en.wikipedia.org/wiki/BitKeeper); decentralized and aims to be fast, lightweight, portable, and easy to use
* [Monotone](https://en.wikipedia.org/wiki/Monotone_(software)) – developed by the Monotone Team; decentralized in a [peer-to-peer](https://en.wikipedia.org/wiki/Peer-to-peer) way
* [SVK](https://en.wikipedia.org/wiki/SVK) – written in [Perl](https://en.wikipedia.org/wiki/Perl) by Kao Chia-liang; built on top of Subversion to allow distributed commits
* [Veracity](https://en.wikipedia.org/wiki/Veracity_(software)) - Is another distributed version control system which includes [bug tracking](https://en.wikipedia.org/wiki/Bug_tracking_system) and [Agile software development](https://en.wikipedia.org/wiki/Agile_software_development) tools integrated with the version control features.

### Proprietary[[edit](https://en.wikipedia.org/w/index.php?title=List_of_version-control_software&action=edit&section=8)]

* [Code Co-op](https://en.wikipedia.org/wiki/Code_Co-op) – peer-to-peer version control system (can use e-mail for synchronization)
* [Sun WorkShop TeamWare](https://en.wikipedia.org/wiki/Sun_WorkShop_TeamWare) – designed[[*citation needed*](https://en.wikipedia.org/wiki/Wikipedia:Citation_needed)] by [Larry McVoy](https://en.wikipedia.org/wiki/Larry_McVoy), creator of BitKeeper
* [Plastic SCM](https://en.wikipedia.org/wiki/Plastic_SCM) – by Codice Software, Inc
* [Visual Studio Team Services](https://en.wikipedia.org/wiki/Visual_Studio_Team_Services) - Services for teams to share code, track work, and ship software for any language by [Microsoft](https://en.wikipedia.org/wiki/Microsoft)

 Git is a distributed VCS; SVN is a non-distributed VCS.

 Git has a centralized server and repository; SVN does not have a centralized server or repository.

 The content in Git is stored as metadata; SVN stores files of content.

 Git branches are easier to work with than SVN branches.

 Git does not have the global revision number feature like SVN has.

 Git has better content protection than SVN.

 Git was developed for Linux kernel by Linus Torvalds; SVN was developed by CollabNet, Inc.

 Git is distributed under GNU, and its maintenance overseen by Junio Hamano; Apache Subversion, or SVN, is distributed under the open source license.

SVN is a Centralized Version Control System (CVCS), and Git is a Distributed Version Control System (DVCS).

A centralized version control system operates on the basic idea that there is one single copy of the project that developers will commit changes to, and where all version of the project are stored.

A distributed version control system, however, works on the principle that each developer “clones” the project repository to their hard drive. A copy of the project is stored on every developer’s local machine, and changes are either “pushed” up to the online repository, or “pulled” down from the repo to update the version that the developer has on their machine.

The model of distributed version control go beyond the simple elimination of the single point of failure found on the centralized model: they also focus on the collaborative aspect of the software development process. The distributed system has the flexibility to meet the most diverse workflows (workflows). Decentralization allows any developer to contribute to a project without write permission on a central repository and the developers can also work with an elected central repository in a similar way to the centralized mode

**Git** is a revision control system, a tool to manage your source code history.

**GitHub** is a hosting service for Git repositories.

So they are not the same thing: **Git** is the **tool**, **GitHub** is the **service for projects that use Git**.

"Git is a free and open source distributed **version control system** designed to handle everything from small to very large projects with speed and efficiency"

"GitHub is a **web-based** Git repository **hosting service**, which offers all of the distributed revision control and source code management (SCM) functionality of Git as well as adding its own features."

Git is a distributed version control tool that can manage a development project's source code history, while GitHub is a cloud based platform built around the Git tool. Git is a tool a developer installs locally on their computer, while GitHub is an online service that stores code pushed to it from computers running the Git tool. The key difference between Git and GitHub is that Git is an open-source tool developers install locally to manage source code, while GitHub is an online service to which developers who use Git can connect and upload or download resources.

OAuth 2 is an authorization framework that enables applications to obtain limited access to user accounts on an HTTP service, such as Facebook, GitHub, and DigitalOcean. It works by delegating user authentication to the service that hosts the user account, and authorizing third-party applications to access the user account. OAuth 2 provides authorization flows for web and desktop applications, and mobile devices.

you have services in the form of APIs, such as the Twitter API, which you can use to get a list of followers or Tweets. Those APIs handle your confidential data, which is protected by a login and password. This brings up the concept of **delegated access**. OAuth2 lets users grant third-party apps access to their web resources, without sharing their passwords, through a security object known as an **access token**. It’s impossible to obtain the password from the access token, since your password is kept safe inside the main service. If an app wants to connect to the service, it must get its own access token. Access tokens can then be revoked if you ever want to revoke access to just that app.

OAuth 2.0 works with the following four actors:

* **authorization server**: responsible for authentication and authorization — it provides the access token.
* **resource server**: in charge of serving up resources if a valid token is provided.
* **resource owner**: the owner of the data — that is, the end user of Incognito.
* **client**: the Incognito mobile app.

Permissions are authorization consents to access LinkedIn resources. The LinkedIn platform uses permissions to protect and prevent abuse of member information. Your application must have the appropriate permissions before it can access data.

Your application can be authorized by LinkedIn in two ways:

* [**Member Authorization**](https://docs.microsoft.com/en-us/linkedin/shared/authentication/authorization-code-flow?context=linkedin/context) or Authorization Code Flow (3-legged authorization): A LinkedIn member grants permissions to your application to access the member’s resources on LinkedIn. Use this flow if you are requesting access to a member's account to use their data and make requests on their behalf.
* [**Application Authorization**](https://docs.microsoft.com/en-us/linkedin/shared/authentication/client-credentials-flow?context=linkedin/context) or Client Credential Flow (2-legged authorization): LinkedIn grants permissions to your application to access protected LinkedIn resources. If you are accessing APIs that are not member specific, use this flow.

|  |  |  |  |
| --- | --- | --- | --- |
| **Member Permission** | Requires member’s approval during the authorization flow. The application has no access to these resources without member approval. | Yes | 3-legged |
| **Application Permission** | Granted to the application directly. It is used to return LinkedIn resources that are not specific to members. | No | 2-legged |

Three-legged OAuth processing involves four parties: resource owner, OAuth client, authorization server, and resource server. In other words, three-legged OAuth is a traditional pattern with resource owner interaction. In this case, a resource owner wants to give a client access to a server without sharing credentials.

Two-legged OAuth processing involves three parties: OAuth client, authorization server, and resource server. The OAuth client can be either the resource owner or the trusted entity that knows about the credentials of the resource owner. In other words, two-legged OAuth processing does not involve additional resource owner interaction.

he typical flow for two-legged OAuth processing involves the following activities:

1. An OAuth client initiates a request with an authorization server and receives an access token.
2. The OAuth client uses the access token to access protected resources on the resource server.

A **blockchain**,[[1]](https://en.wikipedia.org/wiki/Blockchain#cite_note-te20151031-1)[[2]](https://en.wikipedia.org/wiki/Blockchain#cite_note-fortune20160515-2)[[3]](https://en.wikipedia.org/wiki/Blockchain#cite_note-nyt20160521-3) originally **block chain**,[[4]](https://en.wikipedia.org/wiki/Blockchain#cite_note-primer-4)[[5]](https://en.wikipedia.org/wiki/Blockchain#cite_note-obmh-5) is a growing list of [records](https://en.wikipedia.org/wiki/Record_(computer_science)), called *blocks*, which are linked using [cryptography](https://en.wikipedia.org/wiki/Cryptography).[[1]](https://en.wikipedia.org/wiki/Blockchain#cite_note-te20151031-1)[[6]](https://en.wikipedia.org/wiki/Blockchain#cite_note-cryptocurrencytech-6) Each block contains a [cryptographic hash](https://en.wikipedia.org/wiki/Cryptographic_hash_function) of the previous block,[[6]](https://en.wikipedia.org/wiki/Blockchain#cite_note-cryptocurrencytech-6) a [timestamp](https://en.wikipedia.org/wiki/Trusted_timestamping), and transaction data

With a blockchain, many people can write entries into a record of information, and a community of users can control how the record of information is amended and updated. Likewise, Wikipedia entries are not the product of a single publisher. No one person controls the information.

Descending to ground level, however, the differences that make blockchain technology unique become more clear. While both run on distributed networks (the internet), Wikipedia is built into the World Wide Web (WWW) using a client-server network model.

A user (client) with permissions associated with its account is able to change Wikipedia entries stored on a centralized server.  
  
Whenever a user accesses the Wikipedia page, they will get the updated version of the ‘master copy’ of the Wikipedia entry. Control of the database remains with Wikipedia administrators allowing for access and permissions to be maintained by a central authority.

Wikipedia’s digital backbone is similar to the highly protected and centralized databases that governments or banks or insurance companies keep today. Control of centralized databases rests with their owners, including the management of updates, access and protecting against cyber-threats.

The distributed database created by blockchain technology has a fundamentally different digital backbone. This is also the most distinct and important feature of blockchain technology.

Wikipedia’s ‘master copy’ is edited on a server and all users see the new version. In the case of a blockchain, every node in the network is coming to the same conclusion, each updating the record independently, with the most popular record becoming the de-facto official record in lieu of there being a master copy.

These technologies are: 1) private key cryptography, 2) a distributed network with a shared ledger and 3) an incentive to service the network’s transactions, record-keeping and security.

/\*\*\*\*\*\*\*\*\*\*\*\*\* NSSortDescriptor \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

@objcMembers class Person: NSObject {

let firstName: String

let lastName: String

let age: Int

init(firstName: String, lastName: String, age: Int) {

self.firstName = firstName

self.lastName = lastName

self.age = age

}

override var description: String {

return "\(firstName) \(lastName)"

}

}

import UIKit

class ViewController: UIViewController {

override func viewDidLoad() {

super.viewDidLoad()

/\*\*\*\*\*\*\* NSSortDescriptor \*\*\*\*/

let a = Person(firstName: "a", lastName: "b", age: 24)

let b = Person(firstName: "c", lastName: "d", age: 27)

let c = Person(firstName: "e", lastName: "f", age: 33)

let d = Person(firstName: "g", lastName: "h", age: 31)

let peopleObject = [d, b, a, c]

//SWIFTY

let sortedByFirstNameSwifty = peopleObject.sorted(by: { $0.firstName < $1.firstName })

print(sortedByFirstNameSwifty)//prints[a b, c d, e f, g h]

//Objective c way

let firstNameSortDescriptor = NSSortDescriptor(key: "firstName", ascending: true, selector: #selector(NSString.localizedStandardCompare(\_:)))

let sortedByFirstName = (peopleObject as NSArray).sortedArray(using: [firstNameSortDescriptor])

print(sortedByFirstName)//prints [a b, c d, e f, g h]

// Do any additional setup after loading the view, typically from a nib.

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\* NSSortDescriptor \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

A regular expression (commonly known as a “regex”) is a string or a sequence of characters that specifies a pattern.

let string1 = "🇩🇪€4€9"

let matched = matches(for: "[0-9]", in: string1)

print(matched)

func matches(for regex: String, in text: String) -> [String] {

do {

let regex = try NSRegularExpression(pattern: regex)

let results = regex.matches(in: text,

range: NSRange(text.startIndex..., in: text))

return results.map {

String(text[Range($0.range, in: text)!])

}

} catch let error {

print("invalid regex: \(error.localizedDescription)")

return []

}

}

* **[**
* **(** and **)**
* **\**
* **\***
* **+**
* **?**
* **{** and **}**
* **^**
* **$**
* **.**
* **|** (pipe)
* **/**
* These characters are used for advanced pattern matching.
* NSString \*emailRegex = @"[A-Z0-9a-z]+([.\_%+-]{1}[A-Z0-9a-z]+)\*@[A-Z0-9a-z]+([.-]{1}[A-Z0-9a-z]+)\*(\\.[A-Za-z]{2,4}){0,1}";

func isValidEmail(testStr:String) -> Bool {

let emailRegEx = "[A-Z0-9a-z.\_%+-]+@[A-Za-z0-9.-]+\\.[A-Za-z]{2,64}"

let emailTest = NSPredicate(format:"SELF MATCHES %@", emailRegEx)

return emailTest.evaluate(with: testStr)

}

enum RegEx: String {

case email = "[A-Z0-9a-z.\_%+-]+@[A-Za-z0-9.-]+\\.[A-Za-z]{2,4}" // Email

case password = "^.{6,15}$" // Password length 6-15

case alphabeticStringWithSpace = "^[a-zA-Z ]\*$" // e.g. hello sandeep

case alphabeticStringFirstLetterCaps = "^[A-Z]+[a-zA-Z]\*$" // SandsHell

case phoneNo = "[0-9]{10,14}" // PhoneNo 10-14 Digits //Change RegEx according to your Requirement

}

Predicates represent logical conditions, which you can use to filter collections of objects.

 Its query language, which is like a cross between a SQL WHERE clause and a regular expression, provides an expressive, natural language interface to define logical conditions on which a collection is searched.

# Introduction to live streaming

YouTube Live is an easy way to reach your audience in real time. Whether you're streaming a video game, hosting a live Q&A, or teaching a class, our tools will help you manage your stream and interact with viewers in real time.

There are a few ways you can live stream on YouTube. Here’s a quick overview:

**Simple**: These options are great for beginners, or if you want to quickly go live.

* **Webcam**: Live streaming via webcam is an easy way to go live without the need for an encoder. Start streaming from your laptop/desktop computer using your webcam. [Learn more](https://support.google.com/youtube/answer/9228389)
* **Mobile**: Streaming on mobile lets you stream from the YouTube app. Note: to be able to live stream on mobile, you need to have at least 1,000 subscribers. [Learn more](https://support.google.com/youtube/answer/9228390)

YouTube Live is available on iOS 8+.

You can now live stream from the main YouTube app on mobile. After a mobile live stream ends, an archive of the stream is created on your channel, and you have the option to edit the privacy setting (including setting it to private) or delete the archive.

The YouTube Live Streaming API lets you create, update, and manage live events on YouTube. Using the API, you can schedule events (broadcasts) and associate them with video streams, which represent the actual broadcast content.

**Broadcasts** A **broadcast** represents an event that can be watched on YouTube as it happens. Broadcasts can also be recorded and saved as YouTube videos so that users can watch them after they happen.

**Streams** A **stream** identifies the audio-video content that is being communicated to YouTube. Each broadcast is associated with one video stream.

**Cuepoints** A **cuepoint** represents an ad break that can be inserted into a live broadcast.

# Videos: insert

Uploads a video to YouTube and optionally sets the video's metadata.

This method supports media upload. Uploaded files must conform to these constraints:

* **Maximum file size:** 128GB

# UIDocumentPickerViewController

A view controller that provides access to documents or destinations outside your app’s sandbox.

You use a document picker view controller to select a document for importing, exporting, opening, or moving.

# UIDocumentBrowserViewController

A view controller for browsing and performing actions on documents stored locally and in the cloud.

UIDocumentBrowserViewController is much like the Files app on your iPhone which handles a lot for you like iCloud sync and sharing for example.

UIDocumentPickerViewController is just for accessing files in your file system:

# About the Virtual Memory System

Both OS X and iOS include a fully-integrated virtual memory system that you cannot turn off; it is always on. Both systems also provide up to 4 gigabytes of addressable space per 32-bit process. In addition, OS X provides approximately 18 exabytes of addressable space for 64-bit processes. Even for computers that have 4 or more gigabytes of RAM available, the system rarely dedicates this much RAM to a single process. To give processes access to their entire 4 gigabyte or 18 exabyte address space, OS X uses the hard disk to hold data that is not currently in use. As memory gets full, sections of memory that are not being used are written to disk to make room for data that is needed now. The portion of the disk that stores the unused data is known as the backing store because it provides the backup storage for main memory. Although OS X supports a backing store, iOS does not. In iPhone applications, read-only data that is already on the disk (such as code pages) is simply removed from memory and reloaded from disk as needed. Writable data is never removed from memory by the operating system. Instead, if the amount of free memory drops below a certain threshold, the system asks the running applications to free up memory voluntarily to make room for new data. Applications that fail to free up enough memory are terminated.

Virtual memory allows an operating system to escape the limitations of physical RAM. The virtual memory manager creates a logical address space (or “virtual” address space) for each process and divides it up into uniformly-sized chunks of memory called *pages*. The processor and its memory management unit (MMU) maintain a *page table* to map pages in the program’s logical address space to hardware addresses in the computer’s RAM. When a program’s code accesses an address in memory, the MMU uses the page table to translate the specified logical address into the actual hardware memory address. This translation occurs automatically and is transparent to the running application.

As far as a program is concerned, addresses in its logical address space are always available. However, if an application accesses an address on a memory page that is not currently in physical RAM, a *page fault* occurs. When that happens, the virtual memory system invokes a special page-fault handler to respond to the fault immediately. The page-fault handler stops the currently executing code, locates a free page of physical memory, loads the page containing the needed data from disk, updates the page table, and then returns control to the program’s code, which can then access the memory address normally. This process is known as *paging*.

## Wired Memory

Wired memory (also called *resident* memory) stores kernel code and data structures that must never be paged out to disk. Applications, frameworks, and other user-level software cannot allocate wired memory. However, they can affect how much wired memory exists at any time. For example, an application that creates threads and ports implicitly allocates wired memory for the required kernel resources that are associated with them.

**ARC** keeps track of strong references to instances of classes and increases or decreases their reference count accordingly when you assign or unassign instances of classes (reference types) to constants, properties, and variables. It deallocates memory used by objects which reference count got down to zero. **ARC does not increase or decrease the reference count of *value types*because,** when assigned, these are copied. By default, if you don’t specify otherwise, all the references will be strong references.

**Stack Allocation :** The allocation happens on contiguous blocks of memory. We call it stack memory allocation because the allocation happens in function call stack. The size of memory to be allocated is known to compiler and whenever a function is called, its variables get memory allocated on the stack. And whenever the function call is over, the memory for the variables is deallocated. This all happens using some predefined routines in compiler. Programmer does not have to worry about memory allocation and deallocation of stack variables.

**Heap Allocation :**The memory is allocated during execution of instructions written by programmers. Note that the name heap has nothing to do with heap data structure. It is called heap because it is a pile of memory space available to programmers to allocated and de-allocate. If a programmer does not handle this memory well, [memory leak](https://www.geeksforgeeks.org/what-is-memory-leak-how-can-we-avoid/) can happen in the program.

**Key Differences Between Stack and Heap Allocations**

1. In a stack, the allocation and deallocation is automatically done by whereas, in heap, it needs to be done by the programmer manually.
2. Handling of Heap frame is costlier than handling of stack frame.
3. Memory shortage problem is more likely to happen in stack whereas the main issue in heap memory is fragmentation.
4. Stack frame access is easier than the heap frame as the stack have small region of memory and is cache friendly, but in case of heap frames which are dispersed throughout the memory so it cause more cache misses.
5. Stack is not flexible, the memory size allotted cannot be changed whereas a heap is flexible, and the allotted memory can be altered.
6. Accessing time of heap takes is more than a stack.

**Stack and a Heap ?**

Stack is used for static memory allocation and Heap for dynamic memory allocation, both stored in the computer's RAM .

Variables allocated on the stack are stored directly to the memory and access to this memory is very fast, and it's allocation is dealt with when the program is compiled. When a function or a method calls another function which in turns calls another function etc., the execution of all those functions remains suspended until the very last function returns its value. The stack is always reserved in a LIFO order, the most recently reserved block is always the next block to be freed. This makes it really simple to keep track of the stack, freeing a block from the stack is nothing more than adjusting one pointer.

Variables allocated on the heap have their memory allocated at run time and accessing this memory is a bit slower, but the heap size is only limited by the size of virtual memory . Element of the heap have no dependencies with each other and can always be accessed randomly at any time. You can allocate a block at any time and free it at any time. This makes it much more complex to keep track of which parts of the heap are allocated or free at any given time.

You can use the stack if you know exactly how much data you need to allocate before compile time and it is not too big. You can use heap if you don't know exactly how much data you will need at runtime or if you need to allocate a lot of data.

In a multi-threaded situation each thread will have its own completely independent stack but they will share the heap. Stack is thread specific and Heap is application specific. The stack is important to consider in exception handling and thread executions.

Stack is used for static memory allocation and Heap for dynamic memory allocation, both stored in the computer’s RAM .

Variables allocated on the stack are stored directly to the memory and access to this memory is very fast, and it’s allocation is dealt with when the program is compiled.

Variables allocated on the heap have their memory allocated at run time and accessing this memory is a bit slower, but the heap size is only limited by the size of virtual memory .

# What is lazy variable in swift

* You can’t use lazy with let
* You can use lazy  only with members of struct  and class.
* You can’t use it with Computed Properties. Because, a computed property returns the value every time we try to access it after executing the code inside the computation block.
* Lazy variables are not initialised atomically and so is not thread safe.

you just don't have the option of declaring an object on the stack in Obj-C... but those aren't exactly heap allocated either...

in the code:

NSString \*myString=@"MUKESH";

The variable myString is a pointer to an NSString... it is allocated on the stack, the variable that it points to is actually not on the heap, but in a special constant part of memory.

in the code:

NSString \*myString4 =[NSString alloc] initWithString:@"HEllo"];

you may expect the value that myString4 points to to be heap allocated, but since, there is a shortcut in NSString's initWithString method, if the string that it is being inited with is a constant string, it will just return the constant string... you can verify that with:

To get the actual bytes of memory that your application is using, you can do something like the example below. However, you really should become familiar with the various profiling tools as well as they are designed to give you a much better picture of usage over-all.

#import <mach/mach.h> // ...

void report\_memory(void) {

struct task\_basic\_info info; mach\_msg\_type\_number\_t size = TASK\_BASIC\_INFO\_COUNT; kern\_return\_t kerr = task\_info(mach\_task\_self(), TASK\_BASIC\_INFO, (task\_info\_t)&info, &size); if( kerr == KERN\_SUCCESS ) { NSLog(@"Memory in use (in bytes): %lu", info.resident\_size); NSLog(@"Memory in use (in MiB): %f", ((CGFloat)info.resident\_size / 1048576)); } else { NSLog(@"Error with task\_info(): %s", mach\_error\_string(kerr));

}

}

There is also a field in the structure info.virtual\_size which will give you the number of bytes available virtual memory (or memory allocated to your application as potential virtual memory in any event). The code that pgb links to will give you the amount of memory available to the device and what type of memory it is.

# Performance Metrics

Memory refers to the minimum RAM that the app requires to run, and the average and maximum memory that it consumes. Minimum memory puts a strong constraint on the hardware, whereas higher average or peak memory means more background apps are likely be killed. Also, you must ensure that you do not leak memory. A gradual increase in memory consumption over time results in a higher likelihood of app crashes due to out-of-memory exceptions.

Power consumption is not just about calculating CPU cycles—it also involves using the hardware effectively. It is therefore important to not only minimize power consumption but also ensure that the user experience is not degraded.

An app should perform *just enough* tasks at the launch to initialize itself so that the user can work with it. Time taken to perform these tasks is the initialization time of the app.

The following list outlines some of the actions you may want to execute during your app’s initialization, in no particular order:

* Check if the app is being launched for the first time.
* Check if the user is logged in.
* If the user is logged in, load previous state, if applicable.
* Connect to the server for the latest changes.
* Check if the app was launched with a deep link. If so, load the UI and state for the deep link.
* Check if there are pending tasks from the last time the app was launched. Resume them if need be.
* Initialize object and thread pools that you want to use later.
* Initialize dependencies (e.g., object-relational mapping, crash reporting system, and cache).

The list can grow pretty quickly, and it can be difficult to decide what to keep at launch time and what to defer to the next few milliseconds.

## Execution Speed Once the user opens an app, the expectation is for it to work as quickly as possible. Any necessary processing should be handled in as little time as possible.

## Responsiveness Your app should be fast to respond to user interaction. Responsiveness is the result of all the optimizations and trade-offs that you have made in your app.

**Local Storage** Any app that stores data on a server and/or has to refresh its data from an external source must plan for local storage for offline viewing capabilities. For example, a mail app will be expected to at least show previously downloaded messages if the network is not present or the device is in offline mode. However, loading from local storage and syncing the data should be painless and fast. This may require selecting not only the data to be cached locally but also the structure of the data, choosing from a host of options, as well as the frequency of sync.

## Interoperability Users may use multiple apps to accomplish a task, which requires interoperability across them. For example, a photo album may be best viewed in a slideshow app but might require another app for editing it. The viewer app should be able to send a photo to the editor and receive the edited photo. iOS provides multiple options for interoperability and sharing data across apps. UIActivityViewController, deep linking, and the MultipeerConnectivity framework are some of the options available on iOS. Defining good URL structure for deep linking is as important as writing good code to parse it. Similarly, for sharing data using the share sheet, it is important to identify the exact content to be shared as well as to take care of security concerns that arise from processing content from an untrusted source. It would be a really bad user experience if your app took a long time just to prepare data to be shared with a nearby device.

## Network Condition

Mobile devices are used in varying network conditions. To ensure the best user experience, your app must work in all of the following scenarios:

* High bandwidth and persistent network
* Low bandwidth but persistent network
* High bandwidth but sporadic network
* Low bandwidth and sporadic network
* No network

It is acceptable to present the user with a progress indicator or an error message, but it is not acceptable to block indefinitely or let the app crash.

## Data Refresh Even if you do not have any offline viewing capabilities, you may still refresh periodically with data from the server. The rate at which you refresh and the amount of data transferred will affect overall data consumption. If the number of total bytes transferred is large, the user is bound to exhaust his data plan quickly. And if that value is large enough, you may have just lost a user.

## Multiuser Support A family might share a mobile device, or a user may have multiple accounts for the same application. For example, two siblings might share the same iPad for games. As another example, a family may want to configure one device to check each person’s emails during vacation to minimize roaming costs, particularly during international travel. Similarly, one person may have multiple email accounts to be configured.

## Single Sign-on If you have created multiple apps that allow or require sign-in, it is always a good idea to support single sign-on (SSO). If a user logs in to one of your apps, it should be one-click sign-in to your other apps. This process requires more than just sharing data across apps—you’ll also need to share state, synchronize across your apps, and more. For example, if the user signs out using one of the apps, signout should also occur in all other apps where the user signed in using SSO.

## Security Security is paramount in a mobile app, particularly because sensitive information might be shared across apps. It is important to secure all communications, as well as both local and shared data.

# iOS Keys

|  |  |
| --- | --- |
| **NSPhotoLibraryAddUsageDescription** | Your app adds photos to the user's photo library |
| **NSPhotoLibraryUsageDescription** | Your app accesses the user's photo library |
| **NSCameraUsageDescription** | Your app uses the device camera |
| **NSLocationAlwaysUsageDescription** | Your app uses location services all the time |
| **NSLocationWhenInUseUsageDescription** | Your app uses location services only when the app is running |
| **NSLocationUsageDescription** | DEPRECATED: Update to one of the above instead |
| **NSContactsUsageDescription** | Your app uses the address book |
| **NSCalendarsUsageDescription** | Your app uses or modifies the user's calendar information |
| **NSRemindersUsageDescription** | Your app creates reminders in the Reminders app |
| **NSHealthShareUsageDescription** | Your app uses data from the Health app |
| **NSHealthUpdateUsageDescription** | Your app provides health information to the Health app |
| **NFCReaderUsageDescription** | Your app uses the NFC reader |
| **NSBluetoothPeripheralUsageDescription** | Your app works with Bluetooth devices |
| **NSMicrophoneUsageDescription** | Your app uses the device microphone |
| **NSSiriUsageDescription** | Your app provides a SiriKit Intent |
| **NSSpeechRecognitionUsageDescription** | Your app uses speech recognition |
| **NSMotionUsageDescription** | Your app uses the device motion tracking hardware |
| **NSVideoSubscriberAccountUsageDescription** | (tvOS only) your app uses the video subscriber account |
| **NSAppleMusicUsageDescription** | Your app uses Apple Music integration |
| **NSFaceIDUsageDescription** | Your app uses FaceID |

<key>NFCReaderUsageDescription</key> <string>Example: You should fill this in</string>

**Calendar :**

Key      :  Privacy - Calendars Usage Description

Value  :  $(PRODUCT\_NAME) calendar events

**Reminder :**

Key      :   Privacy - Reminders Usage Description

Value  :   $(PRODUCT\_NAME) reminder use

**Contact :**

Key       :   Privacy - Contacts Usage Description

Value    :  $(PRODUCT\_NAME) contact use

**Photo :**

Key       :  Privacy - Photo Library Usage Description

Value   :  $(PRODUCT\_NAME) photo use

**Bluetooth Sharing :**

Key       :  Privacy - Bluetooth Peripheral Usage Description

Value   :  $(PRODUCT\_NAME) Bluetooth Peripheral use

**Microphone :**

Key        :  Privacy - Microphone Usage Description

Value    :  $(PRODUCT\_NAME) microphone use

**Camera :**

Key       :  Privacy - Camera Usage Description

Value   :  $(PRODUCT\_NAME) camera use

**Location :**

Key      :  Privacy - Location Always Usage Description

Value  :  $(PRODUCT\_NAME) location use

Key       :  Privacy - Location When In Use Usage Description

Value   :  $(PRODUCT\_NAME) location use

**Heath :**

Key      :  Privacy - Health Share Usage Description

Value  :  $(PRODUCT\_NAME) heath share use

Key      :  Privacy - Health Update Usage Description

Value  :  $(PRODUCT\_NAME) heath update use

**HomeKit :**

Key      :  Privacy - HomeKit Usage Description

Value  :  $(PRODUCT\_NAME) home kit use

**Media Library :**

Key      :  Privacy - Media Library Usage Description

Value  :  $(PRODUCT\_NAME) media library use

**Motion :**

Key      :  Privacy - Motion Usage Description

Value  :  $(PRODUCT\_NAME) motion use

**Speech Recognition :**

Key      :  Privacy - Speech Recognition Usage Description

Value  :  $(PRODUCT\_NAME) speech use

**SiriKit  :**

Key      :  Privacy - Siri Usage Description

Value  :  $(PRODUCT\_NAME) siri use

**TV Provider :**

Key      :  Privacy - TV Provider Usage Description

Value  :  $(PRODUCT\_NAME) tvProvider use

iOS helps prevent apps from accessing a user’s personal information without permission. Additionally, in Settings, users can see which apps they have permitted to access certain information, as well as grant or revoke any future access. This includes access to:

* Contacts
* Calendars
* Reminders
* Photos
* Motion activity and fitness
* Location Services
* Apple Music
* Your music and video activity
* Microphone
* Camera
* HomeKit
* Health
* Speech recognition
* Bluetooth sharing
* Your media library

If the user signs in to iCloud, apps are granted access by default to iCloud Drive. Users may control each app’s access under iCloud in Settings. Additionally, iOS provides restrictions that prevent data movement between apps and accounts installed by an MDM solution and those installed by the user.

Since iOS 10 it is required to provide a description for the requested permission.

# What is ABI?

ABI stands for Application Binary Interface. At runtime, Swift program binaries interact with other libraries through an ABI. It defines many low level details for binary entities like how to call functions, how their data is represented in memory, where metadata is and how to access it.

# What is ABI Stability?

*ABI stability means locking down the ABI to the point that future compiler versions can produce binaries conforming to the stable ABI. It enables binary compatibility between applications and libraries compiled with different Swift versions.*

Earlier, Swift was not ABI stable, so each binary was bundling its own version of the Swift Dynamic Library. If you open an .ipa you can find swift standard libraries (.dylib) in SwiftSupport or Frameworks.

Now Swift 5 is ABI Stable, Swift will be embeded within the iOS Operating System and it’s ABI will be compatible with every version of Swift.

# Why does ABI Stability matter?

* **Reduced Bundle size:** Size of your application will decrease because you will no longer have to include the Swift standard library in your Frameworks folder.
* **Source compatibility:**Newer compilers can compile code written in an older version of Swift. This aims to reduce the migration pain that Swift developers face when migrating to a newer Swift version.
* **Less frequent language changes:** Less changes leads to less efforts in migration.
* **Binary framework & runtime compatibility:**It enables the distribution of frameworks in a binary form that works across multiple Swift versions. Binary framework include both swift module file and shared library. Module format stability stabilizes the module file, which is the compiler’s representation of the public interfaces of a framework. Module format stability will take time to exist (May not make it in time). Developers can create Pre-compiled Frameworks in Swift because they don’t need to bundle the Swift standard library into their framework.

 It is about **2.6** times faster than Objective-C according to [Apple](https://www.apple.com/swift/), however, some studies indicate that the difference is not that [dramatic](https://www.thedroidsonroids.com/blog/ios/swift-vs-obj-c-performance-comparision). Swift code is easier to maintain as there are no separate interface and implementation files, the syntax is shorter and the language supports [dynamic frameworks](https://www.ca.com/en/blog-developers/dynamic-versus-static-framework-in-ios.html).

It is the **6th most loved language** according to **StackOverflow** [Developer Survey 2018](https://insights.stackoverflow.com/survey/2018/#most-loved-dreaded-and-wanted). For a language released in just 2014, the adoption rate is phenomenal.

One of the key problems articulated by many developers is the lack of backward compatibility with older language versions and the version-lock which means there can be only a single version of Swift in the entire project and its external dependencies. Consequently, developers are forced to completely rewrite their projects if they want to switch to the latest Swift version and update their external dependencies. For developers who make frameworks, they have to update their framework for every new Swift version and they cannot distribute it as a binary precompiled framework.

1. **Source Compatibility,**which means that newer compilers can compile code written in an older version of Swift. This removes the version-lock currently in Swift.
2. **Binary framework & runtime compatibility,**which enables the distribution of frameworks in a binary form that works across multiple Swift versions. Binary framework compatibility will be achieved by **module format stability** which stabilizes the module file, which is the compiler’s representation of the public interfaces of a framework and **ABI stability** enables binary compatibility between applications and libraries compiled with different Swift versions.

Apple made dynamic framework to reduce the launch time of apps and their memory footprints. Dynamic framework also accepts assets inside the framework file, so you don’t have to create a separate bundle file to host files like images or screens (.nib).

Here is how Apple describes their dynamic framework:

“Two important factors that determine the performance of apps are their launch times and their memory footprints. Reducing the size of an app’s executable file and minimizing its use of memory once it’s launched make the app launch faster and use less memory once it’s launched. Using dynamic libraries instead of static libraries reduces the executable file size of an app. They also allow apps to delay loading libraries with special functionality only when they’re needed instead of at launch time. This feature contributes further to reduced launch times and efficient memory use.”

# Maintaining a colour theme manager on iOS swift

All applications maintains a colour scheme though out every screens. This ensures the consistency over the entire application. Broadly, we can classify them under the following categories.

1. **Theme Colour.** Colours on Navigation Bar, Button Titles, Progress Indicator etc.
2. **Border Colour:**Hair line separators in between views.
3. **Shadow Colour:** Shadow colours for card like design.
4. **Dark Background Colour:** Dark background colour to group UI components with light colour.
5. **Light Background Colour:** Light background colour to group UI components with dark colour.
6. **Intermediate Background Colour:**Used for grouping UI elements with some other colour scheme.
7. **Dark Text Colour:**
8. **Light Text Colour:**
9. **Intermediate Text Colour:**
10. **Affirmation:**Colour to show success, something right for user.
11. **Negation:**Colour to show error, some danger zones for user.

#### 20 Best Payment Gateway Providers of 2019

1. [**Stripe**](https://financesonline.com/top-20-payment-gateway-providers/#stripe)
2. [**PayPal Payments Pro**](https://financesonline.com/top-20-payment-gateway-providers/#paypal)
3. [**Amazon Payments**](https://financesonline.com/top-20-payment-gateway-providers/#amazon)
4. [**Braintree**](https://financesonline.com/top-20-payment-gateway-providers/#braintree)
5. [**Authorize.Net**](https://financesonline.com/top-20-payment-gateway-providers/#authorizenet)
6. [**WePay**](https://financesonline.com/top-20-payment-gateway-providers/#wepay)
7. [**BlueSnap**](https://financesonline.com/top-20-payment-gateway-providers/#bluesnap)
8. [**SecurionPay**](https://financesonline.com/top-20-payment-gateway-providers/#securionpay)
9. [**Skrill**](https://financesonline.com/top-20-payment-gateway-providers/#skrill)
10. [**PayU**](https://financesonline.com/top-20-payment-gateway-providers/#payu)
11. [**Zoho Checkout**](https://financesonline.com/top-20-payment-gateway-providers/#zohocheckout)
12. [**BitPay**](https://financesonline.com/top-20-payment-gateway-providers/#bitpay)
13. [**Payoneer**](https://financesonline.com/top-20-payment-gateway-providers/#payoneer)
14. [**CardinalCommerce**](https://financesonline.com/top-20-payment-gateway-providers/#cardinalcommerce)
15. [**Payza**](https://financesonline.com/top-20-payment-gateway-providers/#payza)
16. [**Dwolla**](https://financesonline.com/top-20-payment-gateway-providers/#dwolla)
17. [**PayPro Global**](https://financesonline.com/top-20-payment-gateway-providers/#payproglobal)
18. [**PaySimple**](https://financesonline.com/top-20-payment-gateway-providers/#paysimple)
19. [**CoinGate Bitcoin Payment Processor**](https://financesonline.com/top-20-payment-gateway-providers/#coingatebitcoin)
20. [**WorldPay**](https://financesonline.com/top-20-payment-gateway-providers/#worldpay)

## List of Top 10 Best Payment Gateways in India (Randomly Placed):

### ****1****[InstaMojo Payment Gateway](https://www.brandloom.com/recommended/instamojo" \t "_blank)**:**

[Instamojo](https://www.brandloom.com/recommended/instamojo" \t "_blank) Payment Gateway allows new merchants to create a merchant account instantly to collect online payments with ease with or without a website. Charges of InstaMojo are quite reasonable.

[Instamojo](https://www.brandloom.com/recommended/instamojo" \t "_blank) Payment Gateway service charges are:

* + Annual maintenance Charge: ZERO. yes ZERO
  + Transaction Fee per Transaction: Flat fee @ 2% + Rs 3 per transaction
  + International Payment / Credit card Support: Not supported
  + Multi-Currency Support: Not supported. Only INR.
  + Withdrawal Fees: Zero
  + Settlement days: 3 days in your connected Bank account.
  + Mobile App Integration: Not available as of now.

### 2. [Cashfree](https://www.brandloom.com/recommended/cashfree" \t "_blank) Payment gateway

Cashfree is India’s only Bulk payouts and Payment gateway company. It provides the widest range of payment options: cards (Visa, Master, Maestro, Rupay, Amex), 70+ Netbanking options, Paytm and 6 popular mobile wallets like Airtel, Mobikwik, Freecharge, It also supports UPI, NEFT, IMPS and Paypal. Payment gateway charges, for even new merchant accounts are, just 1.75% per transaction [Lowest TDR charges in India]

### [3 EBS](https://www.ebs.in/IPS/" \t "_blank) Payment****Gateway:****

EBS or E-Billing Solutions is the first Indian merchant account provider to achieve the PCI DSS 3.0 standards of compliance. It is also certified by security and process adherence followed and audited by ISO 27001- 2013 standard. Therefore it is one of the most secured online service provider in the world.

### [4 PayPal](https://www.paypal.com/in/webapps/mpp/merchant" \t "_blank)****Payment Gateway Service:****

PayPal is a Global payment platform available in 200+ countries across the world. PayPal claims to have processed 4 billion payments (including 1 billion mobile) in 2014. Over 173 million customer use Paypal on a regular basis. That is the reason if you are expecting orders from international consumers, you have to have PayPal integrated on your eCommerce Store. PayPal supports more than 100 currencies with fund withdrawal facility in your bank accounts in 57 currencies and hold balances in their PayPal accounts in 26 currencies. In India, merchants primarily use PayPal to receive funds from foreign customers only as due to banking regulations an Indian company cannot accept Indian credit cards via [Paypal.](https://www.paypal.com/in/webapps/mpp/paypal-fees" \t "_blank)

* + [Annual maintenance Charge: Free, Zero maintenance charges.](https://www.paypal.com/in/webapps/mpp/paypal-fees" \t "_blank)
  + [Transaction Fee per Transaction: 4.4% + US$0.30 + Currency conversions charges](https://www.paypal.com/in/webapps/mpp/paypal-fees" \t "_blank)
  + International Payment / Credit card Support: PayPal can be used by an Indian merchant only for receiving international payments only. PayPal does not support or recognize Indian currency, therefore you need to choose an international currency as the base currency. What it means is that if somebody pays you in say Australian dollar through PayPal on your eCommerce store, PayPal will first convert that to US$ (with currency conversion charges). Therefore when you withdraw money from PayPal you will again pay currency conversion charges for conversion to Indian Rupee at your bank in India.
  + Multi-Currency Support: Paypal support over 100 currency globally and as a paypal merchant you can keep your earnings in 57 currencies!
  + Withdrawal Fees: There is no withdrawal fees on PayPal.
  + Settlement days: All payments received in your PayPal account will be auto-withdrawn to your local bank account on a daily basis.
  + Customer Support: PayPal has a great customer support system.

PayPal Payment Gateway Registration process in India

* + No of Days to start transaction: You can start receiving payment immediately after successful registration.
  + Supported eCommerce CMS Systems: Most of the popular ones are supported.
  + Mobile App Integration: iOS and Android.

### 5 [FonePaisa](https://www.fonepaisa.com/) Payment gateway

Fonepaisa is an interoperable payment solutions that brings the worlds of mobile and payments together. The payment gateway is able to link it to loyalty and relationships and make payments matter more for Individuals and Businesses.

**6 [CCAvenue](http://www.ccavenue.com/" \t "_blank) Payment Gateway:**

CCAvenue is one of the largest payment gateway in India, offering wide range of payment options. It offers 200+ payment options inclusive of 6 credit cards such as Amex, JCB, Diners Club, Mastercard, Visa and eZeClick. It also supports 27 major currency, thus enabling you to serve customers in some major global markets out of India. CCAvenue also allows Multilingual payment page in 18 major Indian and international languages.

* + Initial Setup Fee: Zero, yes Zero
  + Annual maintenance Charge for a Startup account: Rs 1200
  + Transaction Fee per Transaction: Variable fee below
    - Domestic Credit & Debit cards on Visa, Mastercard, Maestro RuPay: Flat fee @ 2%
    - Wallets: Freecharge, Mobikwik, OlaMoney, Jiomoney, Paytm, PayZapp, Jana Cash, SBI Buddy, The
    - Mobile Wallet: Flat fee @ 2%
    - IMPS & UPI: Flat fee @ 2%
    - International Credit Cards on Visa, Mastercard, American Express, JCB and Diners Club: Flat fee @ 3%
  + International Payment / Credit card Support: Supported on CCAvenues. Your eCommerce store can represent your product pricing in any number of currencies. Your customers can check-out on the CCAvenue payment option page using their desired currency with Multi-Currency payment Gateway that supports 27 foreign currencies.
  + Multi-Currency Support: CCAvenue allows collection of payments in 27 major foreign currencies and allow your customer to pay in the currency they know best. Currencies supported by CCAvenue Payment Gateway are:
  + Withdrawal Fees: free. Domestic Friday and International Tuesday
  + Settlement days: CCAvenue settles payments on a weekly basis for all amounts over & above of Rs. 1000, which is the minimum amount required to be kept.
  + Customer Support: Their website claims 24x7x365 days of Voice, Chat and email support. However their chat was off on weekend.  IVR system clearly says that they are closed over weekends. However their sales and techncial team responds very well and also coordinates well internally.
  + Documentation required: [List of documents required by CCAvenues for Online Payment Gateway is here.](http://www.ccavenue.com/document_reqd.htm" \t "_blank)
  + No of Days to start transaction: Though CCAvenue claims on their website that all new merchant accounts gets activated within 1 hour of registration, in practice it will take anything between 3 to 6 days. CCAvenue will approve your website and will ask for physical copies of documents before you start transacting through their payment gateway.
  + Types of Credit Card Accepted: 6 credit cards such as Amex, JCB, Diners Club, Mastercard, Visa and eZeClick are accepted.
  + Supported eCommerce CMS System: CCAvenue has developed API’s for most of the shopping carts such as Buildabazaar, Cubecart, Drupal, Interspire, Joomla, Magento, Magento Go, Martjack, Moodle, NopCommerce, Opencart, OsCommerce, PrestaShop, VirtueMart, WHMCS, WordPress, ZenCart.
  + Mobile App Integration: Android, iOS and Windows.

Payment Gateway Acquisition and Installation

### [7 Atom Paynetz](http://www.atomtech.in/" \t "_blank)****Payment Gateway service:****

Atom is promoted by FT Group and it is an innovative Payment Service Provider with over 100 payment options, 15 million + transactions, 4500+ strong merchants across the country.

BEST PAYMENT GATEWAYS HELP YOUR CUSTOMERS TO BUY FROM YOU

### ****8****[Citrus](http://www.citruspay.com/" \t "_blank)****Payment Gateway:****

[It is merged with PayU now](http://www.citruspay.com/merchants/index.html" \t "_blank). Therefore, I have dropped all the details for this payment Gateway.

### 9 [PayTM](https://paytm.com/" \t "_blank) payment Gateway

Paytm has emerged as a leading payment due to its online consumer base with ‘Paytm Cash’ wallet. Therefore paying with Paytm could be beneficial for a small eCommerce merchant.

* Mobile App Integration: Available for both Android, iOS.
* Paytm Payment Gateway Contact Details: signup on their website.

### [10 DirecPay](http://www.direcpay.com/direcpay/home.jsp" \t "_blank) Payment Gateway Service:

DirecPay is one of the largest payment gateway solutions service provider in India.

* + Annual maintenance Charge: Zero.
  + Transaction Fee per Transaction: variable transaction fee apply
  + Domestic Credit and Debit cards on Visa, Mastercard, Maestro, RuPay: Flat fee @ 2%
  + Wallets: Flat fee @ 2%
  + NEFT, IMPS, Cash cards & UPI: Flat fee @ 2%
  + Amex, ezeClick, JCB and Diners: Flat fee @ 3%
  + Taxes extra as applicable from time to time
* Key Features of DirecPay are
  + International Payment / Credit card Support: Supported with an additional refundable security deposit of Rs 30000.
  + Multi-Currency Support: US$ is supported for merchants with International Payment Gateway activation.
  + Withdrawal Fees: Zero
  + Settlement days: Settlement in T + 2 days ( T = Transaction day )
  + Customer Support: Customer support not available over weekends. However you will be able to connect easily with support team over weekdays.
  + Documentation required: New merchant accounts need to provide business registration documents.
  + No of Days to start transaction: 5 to 7 days
  + Supported eCommerce CMS Systems: nopCommerce, Prestashop, OpenCart, OSCommerce, Joomla, WordPress, WHMCS, cubecart and cscart.
  + Mobile App Integration: Android and iOS.
* [DirecPay](http://www.direcpay.com/direcpay/about-direcpay.jsp" \t "_blank) Payment Gateway[Contact Details:](http://www.direcpay.com/direcpay/about-direcpay.jsp" \t "_blank)Call 9619712858; email contact@direcpay.com[.](http://www.direcpay.com/direcpay/about-direcpay.jsp" \t "_blank)
  + per Transaction: Variable charges apply.

PayUbiz Payment Gateway Charges in India

* Key Features of PayUbiz are
  + International Payment / Credit card Support: Not supported
  + Domestic Credit cards supported: Visa/Mastercard/Diners/Amex credit Cards
  + Multi-Currency Support: No
  + Withdrawal Fees: Zero
  + Settlement days: T+2 days
  + Customer Support: Customer support always seems to be busy.
  + Documentation required: Extensive list of documents.
  + No of Days to start transaction: 5-7 days
  + Supported eCommerce CMS Systems: All major CMS supported
  + Mobile App Integration: Android, Windows and iOS.
* [PayUbiz Payment Gateway Contact Details:](https://www.payubiz.in/contact_us2" \t "_blank)Call 0124-6624801; email contact@payu.in

### [12 Mobikwik](https://pay.mobikwik.com/home" \t "_blank) Payment Gateway:

Mobikwik (formerly known as Zaakpay) Payment Gateway is a popular service for online credit, debit card & internet banking processing transactions. They claims to setup an account in two days time.

### SO HERE IS THE LIST OF THE TOP INDIAN PAYMENT GATEWAYS FOR MOBILE APPS!

#### 1. [CCAVENUE](https://www.ccavenue.com/)

**Mobile App Platform:**[Android, Windows, and iOS](https://www.ccavenue.com/inapp_payments.jsp)

#### 2. [CITRUS PAY](http://www.citruspay.com/)

**Mobile App Platform**: [Android & iOS](https://developers.citruspay.com/doc/integrations/)

#### 3. PAYU

PayU provides state-of-the-art payment gateway solutions to online businesses through its cutting-edge and award-winning technology.

#### [PAYUBIZ](https://www.payubiz.in/)

**Annual Maintenance Fees:** Rs. 2400

**Transaction Fees:** between 2.20% to 3.90%

**Setup Time:** 5 – 7 Days

**Major Users:** Snapdeal, Jabong, Bookmyshow, Ola, Cleartrip, Redbus, GoIbibo, Zomato

**Mobile App Platform:** [Android & iOS](https://www.payubiz.in/features)

#### [PAYUMONEY](https://www.payumoney.com/)

**Annual Maintenance Fees:** Nil

**Transaction Fees:** 2%

**Setup Time:** 5 – 7 Days

**Major Users:** ShopClues, Pepperfry, olx, Freecharge

**Mobile App Platform:** [Android & iOS](https://www.payumoney.com/payment-gateway-feature.html)

#### 4. [EBS](https://www.ebs.in/)

**Setup Fees:** From Rs.4799 to Rs. 23999

**Annual Maintenance Fees:** Rs. 2400

**Transaction Fees:** From 1.25% to 3.75%

**Setup Time:** 1 Day

**Major Users:** BabyOye, Snapdeal, FirstCry

**Mobile App Platform:** [Android & iOS](https://www.ebs.in/IPS/features/customized-experience)

#### 5. [INSTAMOJO](https://www.instamojo.com/)

**Mobile App Platform:** [Android](https://support.instamojo.com/hc/en-us/articles/211825909-Android-SDK)

#### 6. [ZAAKPAY](https://www.zaakpay.com/)

**Mobile App Platform:** [Android and iOS](https://www.zaakpay.com/mpay)

**7.**[**RAZORPAY**](https://razorpay.com/)

**Mobile App Platform:** [Andriod & iOS](https://razorpay.com/features/)

#### 8. [JUSPAY](https://juspay.in/)

**Mobile App Platform:** [Android](https://juspay.in/docs/juspay-safe/android/sdk/)

#### 9. [QPAY INDIA](https://www.qpayindia.com/)

**Mobile App Platform:** [iOS, Android & Windows](https://www.qpayindia.com/MobileCheckout.aspx)

#### 10. [EPAISA](https://www.epaisa.com/)

**Mobile App Platform**: [Android](https://www.epaisa.com/hardware/)

#### 11. [KYASH](http://www.kyash.com/)

**Setup Fees:** Rs. 0 – Rs 5000

**Annual Maintenance Fees:** Rs. 500 – Rs 2500

**Transaction Fees:** Rs 5 + 2.5%, Rs 5 + 3.5%

**Setup Time:** A few minutes

**Major Users:** Nationkart, twikster, magento

**Mobile App Platform:** [iOS & Android](http://www.kyash.com/)

#### 12. [BTTPAY](https://www.bttpay.com/)

**Setup Fees:** Nil

**Annual Maintenance Fees:** Nil

**Transaction Fees**: 3.8% +$2.00

**Setup Time:** 48 Hours

**Mobile App Platform**: [Android, Windows & iOS](https://www.bttpay.com/features/#easy-integration-optionss)

#### 13. [TELR](https://telr.com/)

**Setup Fees:** Nil

**Annual Maintenance Fees:** Nil

**Transaction Fees:** Domestic Credit + Net Banking – 2.45% + INR 2, Domestic Debit Cards – 1.25% + INR 2 , International Cards – 2.75% + INR 2

**Setup Time:** A few days

**Major Users:** jaiboot, essentially, flower shop

**Mobile App Platform:** [Android & iOS](https://telr.com/english/features/payment-api.php)

#### 14. [PAYTM](http://paywithpaytm.com/)

**Setup Fees:** Nil

**Annual Maintenance Fees:** Nil

**Transaction Fees:** 2.5%

**Setup Time**: A few hours

**Major Users:** foodpanda, bookmyshow, gaana.com, uber, makemytrip, jabong, oyo rooms, Fasso’s, Spicejet

**Mobile App Platform:** [Android & iOS](http://paywithpaytm.com/developer/#Mobile_SDKs)

#### 15. [2CHECKOUT](https://www.2checkout.com/)

**Setup Fees:** Nil

**Annual Maintenance Fees:** Nil

**Transaction Fees:** 3.9% + $0.45

**Setup Time:** Less than 30 minutes

**Major Users:** Shopify, 3D Cart, aMember, Avactis, BigCommerce, CS-Cart, CubeCart, Ecwid, FreshBooks, Magento, Magic Members, nopCommerce

**Mobile App Platform:** [Android & iOS](https://www.2checkout.com/payment-api#integration)

#### 16. [PAYPAL](https://www.paypal.com/)

**Setup Fees:** Nil

**Annual Maintenance Fees:** Nil

**Transaction Fees:** 4.4% + US$0.30 + Currency conversions charges

**Setup Time:** 5 Days

**Major Users:** shopmatic, freshbooks, shopify, WHMOS

**Mobile App Platform:**[Android & iOS](https://developer.paypal.com/docs/integration/mobile/mobile-sdk-overview/)

#### 17. [INDIAPAY](http://www.indiapay.com/)

**Setup Fees:** Flexible

**Annual Maintenance Fees:** Flexible

**Transaction Fees:** Credit Cards – 2%+ Rs. 7, Debit card – 0.75% to 1% + Rs. 7

**Major Users:** IndiaMart

**Mobile App Platform**: [Android](http://www.indiapay.com/payment-gateway/)

Thus, there are multiple programming languages available to build WebRTC video chat app but only certain languages such as JavaScript have the capability to deliver the exact video/voice chat app that the user expects. On the global market of WebRTC signaling, Node.js is the ideal choice to build a ideal WebRTC enabled video chat application that delivers phenomenal performance, creates API’s, handles the concurrent request and demand scalability to develop your successful WebRTC video/voice chat application in Android, iOS & Web.

TLDs are mainly classified into two categories: generic TLDs and country-specific TLDs. Examples of some of the popular TLDs include .com, .org, .net, .gov, .biz and .edu.

Some of the TLDs and their explanations are as follows:

* .com - Commercial businesses
* .org - Organizations (generally charitable)
* .net - Network organizations
* .gov - U.S. government agencies
* .mil - Military
* .edu - Educational facilities like universities
* .th - Thailand
* .ca - Canada
* .au - Australia

com, short for “commercial,” is the most widely used extension. It’s recommended that any professional entity use .com because it conveys credibility and is easier to remember than other lesser-known extensions.

.net, short for “network,” is a domain name extension used mainly by Internet service providers, network services, online technology companies, and database providers.

ORG domains are usually intended for non-profit organizations.

## 5 Hosting Providers That Offer .com & .net Extensions

### [1. Bluehost](https://app.fitsmallbusiness.com/goto/bluehost-com-vs-net/" \t "_blank) — Best for Affordable Domain Names

**Price:**$2.95 per month to $12.99 per year

### [2. HostGator](https://app.fitsmallbusiness.com/goto/hostgator/" \t "_blank) — Best for Purchasing Only a Domain Name

**Price:** $12.95 per year

### [3. GoDaddy](https://app.fitsmallbusiness.com/goto/godaddy/" \t "_blank) — Best for Marketing a Personal Brand

**Price:**$2.99 per year to $19.99 per year

[GoDaddy](https://app.fitsmallbusiness.com/goto/godaddy/" \t "_blank) offers .com domains starting at $2.99 for the first year, then renewing at $17.99, and .net domains at $19.99 per year. Each domain comes with domain privacy and 400 professional email aliases (example@yoursite.com).

### [4. Wix](https://app.fitsmallbusiness.com/goto/wix/" \t "_blank) — Best for a Domain & Easy-to-Use Website Editor

**Price:** $11 per month to $12.95 per year

### [5. Weebly](https://app.fitsmallbusiness.com/goto/weebly/" \t "_blank) — Best for Simple eCommerce Websites

**Price:**$12 per month to $19.95 per year

The most popular e-commerce platforms and payment service providers support Apple Payin apps and on the web. Using an Apple Pay SDK or JavaScript API from a payment provider is the quickest and most reliable way to support Apple Pay in your app or on your website.

There are some concerns about the usage of Apple Pay among iPhone users, however: statistics show that [only 27%](http://www.businessinsider.com/apple-pay-adoption-underwhelming-goldman-sachs-2017-8?utm_source=feedly&utm_medium=referral" \t "_blank) of iPhone users actually make payments with Apple Pay. However, the reason is that while over 60% of merchants support it, this technology hasn’t received massive adoption yet.

To set up your environment to implement Apple Pay in your apps, you must complete these steps:

1. Create a Merchant ID
2. Configure Apple Pay capabilities in Xcode for your project
3. Create a sandbox user in iTunes Connect
4. Add a test card
5. Create a payment request in your project
6. Handle the result

Navigate to [https://developer.apple.com](https://developer.apple.com/" \t "_blank) and sign in to your developer account. Navigate to **Certificates, Identifiers & Profiles**.

Go to the **Identifier**section and choose **App ID**.

Make sure that you select the Explicit App ID, because if it isn’t, the wildcard App IDs won’t be able to make payments with Apple Pay. Also, you should check the Apple Pay checkbox under App Services.

Now navigate to the **Merchant ID** section.

## Google Pay for Business

## Transact Easily. Sell More.

With Google Pay for Business, simple and secure payments are just the beginning. If you sell online, sign up now in order to get your own Business Channel on the Google Pay app where you can engage directly with your customers to share reminders, upcoming tickets and more. For merchants who operate a store, also sign up below to get more details.

Google Pay (also known as **Google Tez** or **Pay with Google** or **Android Pay**) is a digital wallet platform and online payment system developed by Google to power in-app and tap-to-pay purchases on mobile devices, enabling users to make payments with Android phones, tablets or watches.

In India, Google Pay is integrated with the UPI payment platform. Hence, any person who has a UPI payment account can configure Google Pay to send or receive money using UPI.

# Accept payments for your business

You can use the Google Pay app to accept payments for your business. You will need to use an individual current account. Using this account, you can receive up to Rs. 50,000 per month with UPI with no fees. Beyond that, your bank might charge fees or add more requirements.

in January 2018, Google announced a rebrand of its consumer payment products and merged Android Pay and Google Wallet into a single, unified service known as Google Pay or, informally, GPay. Further, Google Pay incorporates Google Chrome's autofill feature.

Many applications today provide the option to get extra content and features by making purchases through them. This technique, known as **In-App Purchases**,

### Creating an App ID

Choose **Explicit App ID** and enter a unique **Bundle ID**.

### Creating an App in iTunes Connect

Now, while viewing your app’s entry in App Store Connect, click on the **Features** tab and then select **In-App Purchases**. To add a new IAP product, click the **+** to the right of **In-App Purchases**.

### Creating a Sandbox User

In App Store Connect, click **App Store Connect** in the top left corner of the window to get back to the main menu. Select **Users and Roles**, then click the **Sandbox Testers** tab. Click **+** next to the “Tester” title.

Next select the **Capabilities** tab. Scroll down to **In-App Purchase** and toggle the switch to **ON**.

**Apple charge 30%**for each successful transaction that your application will make

## ****Types Of In App Purchases:****

When user make any IAP to buy extra content of subscription, make one of these in app purchase:

1. **Consumable:** User will need to buy these items every time, as want to use them, can not be used in future for free. On reinstalling, changing device user might lose their consumable products. For example, buying game currency, health and hints etc.
2. **Non-consumable:**User buy these item one time, can be used in future for free. On reinstalling, changing device these products will not be lost. If user lose, might be able to download it again for free by [restore in-app purchases.](https://support.apple.com/en-in/HT204530?source=post_page---------------------------) For example: upgrading app to pro version, removing ads etc.
3. **Non-renewing subscriptions:**User will be able to use these items for a fixed period of time, these items can be purchase again after the subscription end. For example: a sport session pass for one, three, or six moths.
4. **Auto-renewable subscriptions:**User can buy these item for a specified period of time, It’ll automatically renew when the period has passed. For example: Ongoing services (Netflix, Hulu Plus, etc.), Magazine subscriptions etc.

## Setup Bank Account:

If you don’t have the bank account setup in your iTunes connect account, will not be able to use IAP services ❗. So, it’s a most important part to get the IAP services work. We can do it by going to the **Agreements, Tax and Banking** section of [iTunes connect](https://itunesconnect.apple.com/?source=post_page---------------------------) account. It should look something like this:

It is very important to add sandbox user to test the IAP services, by sandbox user you can make transactions for free.

Wizdler Recognizes WSDL information on the page to show you the available services and operations. Click the operation to generate the SOAP requests and view the response. By clicking the service, you can download WSDL and external XSD files in single ZIP file.

Chrome extension to test SOAP Web Services (appropriately) called **[Wizdler](https://chrome.google.com/webstore/detail/wizdler/oebpmncolmhiapingjaagmapififiakb?hl=en" \t "_blank)**.

# SOAP vs REST:

# Simple Object Access Protocol Vs. REpresentational State Transfer

SOAP is a method of transferring messages, or small amounts of information, over the Internet. SOAP messages are formatted in XML and are typically sent using HTTP (hypertext transfer protocol).

SOAP uses WSDL for communication between consumer and provider, whereas REST just uses XML or JSON to send and receive data.

### REST (REpresentational State Transfer):

Rest is a simple way of sending and receiving data between client and server and it doesn’t have very many standards defined. You can send and receive data as [JSON](https://crunchify.com/category/json/" \t "_blank), [XML](https://crunchify.com/tag/pom-xml/" \t "_blank) or even plain text. It’s lightweight compared to SOAP.

REST need not be over HTTP but most of my points below will have an HTTP bias.

**SOAP**

**REST**

* SOAP stands for Simple Object Access Protocol
* REST stands for Representational State Transfer
* SOAP is a protocol. SOAP was designed with a specification. It includes a WSDL file which has the required information on what the web service does in addition to the location of the web service.
* REST is an Architectural style in which a web service can only be treated as a RESTful service if it follows the constraints of being
  1. Client Server
  2. Stateless
  3. Cacheable
  4. Layered System
  5. Uniform Interface
* SOAP cannot make use of REST since SOAP is a protocol and REST is an architectural pattern.
* REST can make use of SOAP as the underlying protocol for web services, because in the end it is just an architectural pattern.
* SOAP uses service interfaces to expose its functionality to client applications. In SOAP, the WSDL file provides the client with the necessary information which can be used to understand what services the web service can offer.
* REST use Uniform Service locators to access to the components on the hardware device. For example, if there is an object which represents the data of an employee hosted on a URL as http://demo.guru99 , the below are some of URI that can exist to access them

http://demo.guru99.com/Employee

http://demo.guru99.com/Employee/1

* SOAP requires more bandwidth for its usage. Since SOAP Messages contain a lot of information inside of it, the amount of data transfer using SOAP is generally a lot.

<?xml version="1.0"?> <SOAP-ENV:Envelope xmlns:SOAP-ENV ="http://www.w3.org/2001/12/soap-envelope" SOAP-ENV:encodingStyle =" http://www.w3.org/2001/12/soap-encoding"> <soap:Body> <Demo.guru99WebService xmlns="http://tempuri.org/"> <EmployeeID>int</EmployeeID> </Demo.guru99WebService> </soap:Body> </SOAP-ENV:Envelope>

* REST does not need much bandwidth when requests are sent to the server. REST messages mostly just consist of JSON messages. Below is an example of a JSON message passed to a web server. You can see that the size of the message is comparatively smaller to SOAP.

{"city":"Mumbai","state":"Maharastra"}

* SOAP can only work with XML format. As seen from SOAP messages, all data passed is in XML format.
* REST permits different data format such as Plain text, HTML, XML, JSON, etc. But the most preferred format for transferring data is JSON.

**REST services should be used in the following instances**

* **Limited resources and bandwidth** – Since SOAP messages are heavier in content and consume a far greater bandwidth, REST should be used in instances where network bandwidth is a constraint.
* **Statelessness** – If there is no need to maintain a state of information from one request to another then REST should be used. If you need a proper information flow wherein some information from one request needs to flow into another then SOAP is more suited for that purpose. We can take the example of any online purchasing site. These sites normally need the user first to add items which need to be purchased to a cart. All of the cart items are then transferred to the payment page in order to complete the purchase. This is an example of an application which needs the state feature. The state of the cart items needs to be transferred to the payment page for further processing.
* **Caching**– If there is a need to cache a lot of requests then REST is the perfect solution. At times, clients could request for the same resource multiple times. This can increase the number of requests which are sent to the server. By implementing a cache, the most frequent queries results can be stored in an intermediate location. So whenever the client requests for a resource, it will first check the cache. If the resources exist then, it will not proceed to the server. So caching can help in minimizing the amount of trips which are made to the web server.
* **Ease of coding**– Coding REST Services and subsequent implementation is far easier than SOAP. So if a quick win solution is required for web services, then REST is the way to go.

SOAP should be used in the following instances

1. **Asynchronous processing and subsequent invocation** – if there is a requirement that the client needs a guaranteed level of reliability and security then the new SOAP standard of SOAP 1.2 provides a lot of additional features, especially when it comes to security.
2. **A Formal means of communication** – if both the client and server have an agreement on the exchange format then SOAP 1.2 gives the rigid specifications for this type of interaction. An example is an online purchasing site in which users add items to a cart before the payment is made. Let's assume we have a web service that does the final payment. There can be a firm agreement that the web service will only accept the cart item name, unit price, and quantity. If such a scenario exists then, it's always better to use the SOAP protocol.
3. **Stateful operations –**ifthe application has a requirement that state needs to be maintained from one request to another, then the SOAP 1.2 standard provides the WS\* structure to support such requirements.

# WSDL (Web Service Description Language)

**Challenges with the SOAP API**

1. WSDL file - One of the key challenges of the SOAP API is the WSDL document itself. The WSDL document is what tells the client of all the operations that can be performed by the web service. The WSDL document will contain all information such as the data types being used in the SOAP messages and what all operations are available via the web service. The below code snippet is just part of a sample WSDL file.

<?xml version="1.0"?> <definitions name="Tutorial" targetNamespace=http://demo.guru99.com/Tutorial.wsdl xmlns:tns=http://demo.guru99.com/Tutorial.wsdl xmlns:xsd1=http://demo.guru99.com/Tutorial.xsd xmlns:soap=http://schemas.xmlsoap.org/wsdl/soap/ xmlns="http://schemas.xmlsoap.org/wsdl/"> <types> <schema targetNamespace=http://Demo.guru99.com/Tutorial.xsd xmlns="http://www.w3.org/2000/10/XMLSchema"> <element name="TutorialNameRequest"> <complexType> <all> <element name="TutorialName" type="string"/> </all> </complexType> </element> <element name="TutorialIDRequest"> <complexType> <all> <element name="TutorialID" type="number"/> </all> </complexType> </element> </schema> </types>

1. Document size – The other key challenge is the size of the SOAP messages which get transferred from the client to the server. Because of the large messages, using SOAP in places where bandwidth is a constraint can be a big issue.

**Challenges with the REST API**

1. **Lack of Security**– REST does not impose any sort of security like SOAP. This is why REST is very appropriate for public available URL's, but when it comes down to confidential data being passed between the client and the server, REST is the worst mechanism to be used for web services.
2. **Lack of state**– Most web applications require a stateful mechanism. For example, if you had a purchasing site which had the mechanism of having a shopping cart, it is required to know the number of items in the shopping cart before the actual purchase is made. Unfortunately, the burden of maintaining this state lies with the client, which just makes the client application heavier and difficult to maintain.

A WSDL is an XML document that describes a web service. It actually stands for Web Services Description Language.

SOAP is an XML-based protocol that lets you exchange info over a particular protocol (can be HTTP or SMTP, for example) between applications. It stands for Simple Object Access Protocol and uses XML for its messaging format to relay the information.

**3)** WSDL is often used in combination with SOAP and an XML Schema to provide Web services over the Internet. A client program connecting to a Web service can read the WSDL file to determine what operations are available on the server. Any special datatypes used are embedded in the WSDL file in the form of XML Schema. The client can then use SOAP to actually call one of the operations listed in the WSDL file using for example XML over HTTP.

Each soap web service can be defined with a WSDL(Web Service Definition Language) which is kind of a schema for the SOAP XML transferred.

There are many tools available to convert WSDL(your webservice definition) to native code.

One of the tool available for ObjC is **Sudz-C (**[**http://sudzc.com/**](http://sudzc.com/)**)** which convert the WDSL of any webservice to ObjC code to access the Web service.

Rest is a simple way of sending and receiving data between client and server and it don't have any much standards defined , You can send and receive data as JSON,XML or even a Text. Its Light weighted compared to SOAP.

To handle Rest in iOS there are many tools available, I would recommend **RestKit**[**http://restkit.org/**](http://restkit.org/), for handling XML and JSON both.

SwiftyJSON SwiftyJSON makes it easy to deal with JSON data in Swift.

[SwiftyXMLParser](https://github.com/yahoojapan/SwiftyXMLParser)Simple XML Parser implemented in Swift

[NSXMLParser](https://developer.apple.com/library/mac/documentation/Cocoa/Reference/Foundation/Classes/NSXMLParser_Class/) in Foundation framework is a kind of "SAX" parser. It has enough performance but is a little inconvenient. So we have implemented "DOM" parser wrapping it.

SnapKit is a DSL to make Auto Layout easy on both iOS and OS X.

With **[SnapKit](http://snapkit.io/" \t "_blank)** it’s extremely easy to create and setup constraints, as the technique that it offers to do that is quite simple, and on top of that the amount of code required is way less compared to the traditional way.

* Build compelling user interfaces with SnapKit
* Use SnapKit building view with auto layouts constraints programmatically

let superview = self.view

let button = UIButton()button.layer.cornerRadius = 33button.setTitle(“+”, forState: .Normal)

button.backgroundColor = UIColor.blackColor()

superview.addSubview(button)

button.snp\_makeConstraints { (make) -> Void in

make.width.equalTo(66)

make.height.equalTo(66)

make.bottom.equalTo(superview.snp\_bottom).offset(-20)

make.right.equalTo(superview.snp\_right).offset(-20)

}

# isKind(of:)

Returns a Boolean value that indicates whether the receiver is an instance of given class or an instance of any class that inherits from that class.

## Return Value

true if the receiver is an instance of aClass or an instance of any class that inherits from aClass, otherwise false.

Option 1:

if view is UITabBarItem { }

Option 2:

if view.isKind(of:UITabBarItem.self) { }

Option 3:

if view.isMember(of:UITabBarItem.self) { }

The difference between isKind:of and isMember:of is:

* isKind:of returns YES if the receiver is an instance of the exact class, OR if the receiver is an instance of any subclass of the class.
* isMember:of **only** returns YES if the receiver is an instance of the exact class you're checking against

# Swift type checking

If you merely want to check the type of an expression, without casting to that type, then you can use these approaches. They are only available in Swift, but still work when dealing with Objective C types.

### The [is](https://developer.apple.com/library/content/documentation/Swift/Conceptual/Swift_Programming_Language/Expressions.html#//apple_ref/doc/uid/TP40014097-CH32-ID385) operator

* + The is operator checks at runtime whether the expression can be cast to the specified type. It returns true if the expression can be cast to the specified type; otherwise, it returns false
  + Works on any Swift type, including Objective C types.
  + Swift equivalent of [isKind(of:)](https://developer.apple.com/reference/objectivec/nsobjectprotocol/1418511-iskind)

### Using [type(of:)](https://developer.apple.com/library/content/documentation/Swift/Conceptual/Swift_Programming_Language/Expressions.html#//apple_ref/swift/grammar/postfix-self-expression)

* + Unlike the is operator, this can be used to check the exact type, without consideration for subclasses.
  + Can be used like: type(of: instance) == DesiredType.self
  + Swift equivalent of [isMember(of:)](https://developer.apple.com/reference/objectivec/nsobjectprotocol/1418766-ismember)

# Legacy (Objective C) methods for checking types

These are all methods on [NSObjectProtocol](https://developer.apple.com/reference/objectivec/nsobjectprotocol). They can be used in Swift code, but **they only apply work with classes that derive from NSObjectProtocol** (such as subclasses of NSObject). I advise against using these, but I mention them here for completeness

### [isKind(of:)](https://developer.apple.com/reference/objectivec/nsobjectprotocol/1418511-iskind)

* + Returns a Boolean value that indicates whether the receiver is an instance of given class **or an instance of any class that inherits from that class**
  + Avoid this in Swift, use is operator instead.

### [isMember(of:)](https://developer.apple.com/reference/objectivec/nsobjectprotocol/1418766-ismember)

* + Returns a Boolean value that indicates whether the receiver is an instance of a given class
  + Is a method on [NSObjectProtocol](https://developer.apple.com/reference/objectivec/nsobjectprotocol), thus only works on classes that derive from NSObjectProtocol (such as subclasses of NSObject)
  + Avoid this in Swift, use type(of: instance) == DesiredType.self instead.

### [conforms(to:)](https://developer.apple.com/reference/objectivec/nsobjectprotocol/1418515-conforms)

* + Returns a Boolean value that indicates whether the receiver conforms to a given protocol.
  + Is a method on [NSObjectProtocol](https://developer.apple.com/reference/objectivec/nsobjectprotocol), thus only works on classes that derive from NSObjectProtocol (such as subclasses of NSObject)
  + Avoid this in Swift, use is operator instead.

[**Singleton**](https://en.wikipedia.org/wiki/Singleton_pattern?source=post_page---------------------------) is a design pattern which is very popular in development. Most of the developers are using this design pattern. This is very simple, common and **easy** to use in your project. It’s initialize your class instance single time only with **static** property and it will share your **class** instance **globally**.

## Here the one simple example for using class

class **LocationManager**{//MARK: - Location Permission  
 func **requestForLocation**(){  
 //Code Process  
 **print**("Location granted")  
 }  
   
}//Access the class  
let location = **LocationManager**() **//**initialization class  
location.**requestForLocation**() **//**Call function here

This is the class with out **singleton** pattern for access any function we need to **initialise** class every time for avoiding this things we are using **singleton**classes with **static** instance.

## Write your first Singleton Class 👍🏻

class **LocationManager**{  
   
 **static** let shared = **LocationManager**()  
   
 **init**(){}  
   
 func **requestForLocation**(){  
 //Code Process  
 print("Location granted")  
 }  
   
}//Access class function with Singleton Pattern 🚀  
**LocationManager**.shared.**requestForLocation**() **//**"Location granted"//Still you can use your class like this  
let location = **LocationManager**()  
location.**requestForLocation**()

## ****Better way to write your Singleton class 😎****

class **LocationManager**{  
   
 **static** let shared = **LocationManager**()  
   
 **var** locationGranted: **Bool**? //Initializer access level change now  
 private **init**(){}  
   
 func **requestForLocation**(){  
 //Code Process   
 locationGranted = true   
 **print**("Location granted")  
 }  
   
}//Access class function in a single line  
**LocationManager**.shared.**requestForLocation**()

Every class having **default** **public** initializer, its change to now **private**. Now you can’t **initialize** your **singleton** class again.

## How to use Singleton 🎉

//In a single line you can access easily  **LocationManager**.shared.**requestForLocation**() // "Location granted"//Access variable value **print(**LocationManager.shared.locationGranted **??** false**)** // true

**// Shared URL Session let sharedURLSession = URLSession.shared**

**// Default File Manager let defaultFileManager = FileManager.default**

**// Standard User Defaults let standardUserDefaults = UserDefaults.standard**

**// Default Payment Queue let defaultPaymentQueue = SKPaymentQueue.default()**

The singleton pattern is a very useful pattern. There are times that you want to make sure only one instance of a class is instantiated and that your application only uses that instance. That's the primary and only goal of the singleton pattern.

To jailbreak your [iPhone](https://www.lifewire.com/pick-best-iphone-for-you-1999293) is to free it from the limitations imposed on it by its manufacturer (Apple) and carrier (e.g. AT&T, Verizon, etc.).

After a jailbreak, the device can do things it previously couldn't, such as install unofficial apps and modify settings and areas of the phone that were previously restricted.

Jailbreaking lets you do everything from customizing the look of your iPhone to installing [third-party applications](https://www.lifewire.com/what-is-a-third-party-app-4154068), which are titles that are not authorized and available in the [App Store](https://www.lifewire.com/download-iphone-apps-at-app-store-2000650). A third-party app can add tons of functionality to your phone that you'd otherwise never see through the App Store.

For starters, once you jailbreak your phone, you're entirely on your own since you may void the warranty you have with your carrier. This means that if something horrible happens to your phone, you can't rely on AT&T, Verizon, or Apple to fix them.

Jailbreaking is the act of changing the iPhone (or iPad) software to remove the restrictions and limitations imposed by Apple. The principal limitation is that software can only be installed from the App Store. With a jailbroken phone you can install software from a rival to the App Store, and also manually using files downloaded from the internet.

The **International Mobile Equipment Identity** or **IMEI** [/aɪˈmiː/](https://en.wikipedia.org/wiki/Help:IPA/English)[[1]](https://en.wikipedia.org/wiki/International_Mobile_Equipment_Identity#cite_note-3gppspec-1) is a number, usually unique,[[2]](https://en.wikipedia.org/wiki/International_Mobile_Equipment_Identity#cite_note-bbcnews-2)[[3]](https://en.wikipedia.org/wiki/International_Mobile_Equipment_Identity#cite_note-3) to identify [3GPP](https://en.wikipedia.org/wiki/3GPP) and [iDEN](https://en.wikipedia.org/wiki/IDEN) [mobile phones](https://en.wikipedia.org/wiki/Mobile_phone), as well as some [satellite phones](https://en.wikipedia.org/wiki/Satellite_phone). It is usually found printed inside the battery compartment of the phone, but can also be displayed on-screen on most phones by entering \*#06# on the dialpad, or alongside other system information in the settings menu on smartphone operating systems.

[GSM](https://en.wikipedia.org/wiki/GSM) networks use the IMEI number to identify valid devices, and can stop a stolen phone from accessing the network. For example, if a [mobile phone](https://en.wikipedia.org/wiki/Mobile_phone) is stolen, the owner can have their network provider use the IMEI number to blacklist the phone. This renders the phone useless on that network and sometimes other networks, even if the thief changes the phone's [subscriber identity module](https://en.wikipedia.org/wiki/Subscriber_identity_module) (SIM).

You can get the UDID, but can not get the IMEI.Apple does not allow this.

No, the device's phone number is not available programmatically and still be approved by Apple, thankfully. I definitely don't want some free app I download out of curiosity to grab my phone number and start sending me spam SMS messages or have marketers calling me.

Framework

# EventKit

Create, view, and edit calendar and reminder events.

The EventKit framework provides access to calendar and reminders data so you and your users can create, retrieve, and edit calendar items in your app. In iOS, [EventKitUI](https://developer.apple.com/documentation/eventkitui) provides user interfaces you can implement in your app to let users create and edit calendar items.

You can use EventKit to set up alarms and create recurring events. And if a change to the Calendar database occurs from outside your app, EventKit detects the change and sends a notification, allowing you to stay up to date.

On iOS, use the EventKitUI framework to show calendar and reminder information to the user modally. EventKitUI provides view controllers for viewing and editing calendar and reminder information, choosing which calendar to view, and for determining whether to present calendars as read-only or readable and writeable.

# EKEventStore

An object that accesses the user’s calendar and reminder events and supports the scheduling of new events.

An [EKAlarm](https://developer.apple.com/documentation/eventkit/ekalarm) object represents an alarm in Event Kit.

# EKError

An error returned from EventKit.

# EKEvent

A class that represents an event added to a calendar.

|  |  |  |
| --- | --- | --- |
| [India](https://en.wikipedia.org/wiki/India) | 1 | [UTC+05:30](https://en.wikipedia.org/wiki/UTC%2B05:30) ([IST](https://en.wikipedia.org/wiki/Indian_Standard_Time)) |
| EST | [Eastern Standard Time (North America)](https://en.wikipedia.org/wiki/Eastern_Standard_Time_(North_America)) | [UTC−05](https://en.wikipedia.org/wiki/UTC%E2%88%9205) |
| IST | [Indian Standard Time](https://en.wikipedia.org/wiki/Indian_Standard_Time) | [UTC+05:30](https://en.wikipedia.org/wiki/UTC%2B05:30) |
| IST | [Irish Standard Time](https://en.wikipedia.org/wiki/Irish_Standard_Time)[[8]](https://en.wikipedia.org/wiki/List_of_time_zone_abbreviations#cite_note-9) | [UTC+01](https://en.wikipedia.org/wiki/UTC%2B01) |
| IST | [Israel Standard Time](https://en.wikipedia.org/wiki/Israel_Standard_Time) | [UTC+02](https://en.wikipedia.org/wiki/UTC%2B02) |

[UTC−05:00](https://en.wikipedia.org/wiki/UTC%E2%88%9205:00) ([ET](https://en.wikipedia.org/wiki/Eastern_Time_Zone)) — [Eastern Time zone](https://en.wikipedia.org/wiki/Eastern_Time_Zone#United_States)[[*clarify*](https://en.wikipedia.org/wiki/Wikipedia:Please_clarify)]: roughly a triangle covering all the states from the Great Lakes down to Florida and east to the [Atlantic coast](https://en.wikipedia.org/wiki/East_Coast_of_the_United_States)

Big data is a variety of tools, approaches and methods for processing both structured and unstructured data in order to use it for specific tasks and purposes. A typical example of big data is information coming from various physical experimental installations, for example, the Large Hadron Collider, which produces a huge amount of data and does it all the time. The installation continuously produces large amounts of data, and scientists with their help, scientists solve in parallel many problems.

Big data analytics, machine learning, search engine indexing and many other areas of modern data operations require web crawling and web scraping data. There is a tendency to interchangeably use the terms web crawling and web scraping and although they are closely related, there are differences between the two processes.

**Web crawling** aka Indexing, is used to index the information on the page using bots also known as crawlers. Web Crawlers are basically used by major search engines like google, bing, yahoo, in other terms Google, Bing are one of the major web crawlers.   
Here we get generic information, where as scraping we get specific information.    
  
**Web scraping** aka Web Data Extraction, is an automated way of extracting the information/content using bots aka scrapers. Here the information can be used to replicate in some other website or can be used to do data analysis.    
[Information in this context means all varieties of content including images, text, sensible information like contact details, price etc.]

**Web scraping**, **web harvesting**, or **web data extraction** is [data scraping](https://en.wikipedia.org/wiki/Data_scraping) used for [extracting data](https://en.wikipedia.org/wiki/Data_extraction) from [websites](https://en.wikipedia.org/wiki/Website).[[1]](https://en.wikipedia.org/wiki/Web_scraping#cite_note-Boeing2016JPER-1) Web scraping software may access the World Wide Web directly using the [Hypertext Transfer Protocol](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol), or through a web browser. While web scraping can be done manually by a software user, the term typically refers to automated processes implemented using a [bot](https://en.wikipedia.org/wiki/Internet_bot) or [web crawler](https://en.wikipedia.org/wiki/Web_crawler). It is a form of copying, in which specific data is gathered and copied from the web, typically into a central local [database](https://en.wikipedia.org/wiki/Database) or spreadsheet, for later [retrieval](https://en.wikipedia.org/wiki/Data_retrieval) or [analysis](https://en.wikipedia.org/wiki/Data_analysis).

Web scraping a web page involves fetching it and extracting from it.[[1]](https://en.wikipedia.org/wiki/Web_scraping#cite_note-Boeing2016JPER-1)[[2]](https://en.wikipedia.org/wiki/Web_scraping#cite_note-2) Fetching is the downloading of a page (which a browser does when you view the page). Therefore, web crawling is a main component of web scraping, to fetch pages for later processing. Once fetched, then extraction can take place. The content of a page may be [parsed](https://en.wikipedia.org/wiki/Parsing), searched, reformatted, its data copied into a spreadsheet, and so on. Web scrapers typically take something out of a page, to make use of it for another purpose somewhere else. An example would be to find and copy names and phone numbers, or companies and their URLs, to a list (contact scraping).

[Web Scraping](http://en.wikipedia.org/wiki/Web_scraping" \t "_blank) (also termed Screen Scraping, Web Data Extraction, Web Harvesting etc.) is a technique employed to extract large amounts of data from websites whereby the data is extracted and saved to a local file in your computer or to a database in table (spreadsheet) format.

Data displayed by most websites can only be viewed using a web browser. They do not offer the functionality to save a copy of this data for personal use. The only option then is to manually copy and paste the data - a very tedious job which can take many hours or sometimes days to complete. [Web Scraping](http://www.webharvy.com/) is the technique of automating this process, so that instead of manually copying the data from websites, the [Web Scraping software](http://www.webharvy.com/) will perform the same task within a fraction of the time.

Dependency injection is basically providing the objects that an object needs (its dependencies) instead of having it construct them itself. It's a very useful technique for testing, since it allows dependencies to be mocked or stubbed out.

Dependencies can be injected into objects by many means (such as constructor injection or setter injection). One can even use specialized dependency injection frameworks (e.g. Spring) to do that, but they certainly aren't required. You don't need those frameworks to have dependency injection. Instantiating and passing objects (dependencies) explicitly is just as good an injection as injection by framework.

In object-oriented programming ([OOP](https://searchmicroservices.techtarget.com/definition/object-oriented-programming-OOP)) software design, dependency injection (DI) is the process of supplying a resource that a given piece of code requires. The required resource, which is often a component of the application itself, is called a dependency.

# [How to launch chrome browser while opening a link in iOS objective c](https://stackoverflow.com/questions/50334505/how-to-launch-chrome-browser-while-opening-a-link-in-ios-objective-c)

<https://stackoverflow.com/questions/50334505/how-to-launch-chrome-browser-while-opening-a-link-in-ios-objective-c>

NSString\* url = @"some url"; NSURL \*inputURL = [NSURL URLWithString:url]; NSString \*scheme = inputURL.scheme; // Replace the URL Scheme with the Chrome equivalent. NSString \*chromeScheme = nil; if ([scheme isEqualToString:@"http"]) { chromeScheme = @"googlechrome"; } else if ([scheme isEqualToString:@"https"]) { chromeScheme = @"googlechromes"; } // Proceed only if a valid Google Chrome URI Scheme is available. if (chromeScheme) { NSString \*absoluteString = [inputURL absoluteString]; NSRange rangeForScheme = [absoluteString rangeOfString:@":"]; NSString \*urlNoScheme = [absoluteString substringFromIndex:rangeForScheme.location]; NSString \*chromeURLString = [chromeScheme stringByAppendingString:urlNoScheme]; NSURL \*chromeURL = [NSURL URLWithString:chromeURLString]; // Open the URL with Chrome. [[UIApplication sharedApplication] openURL:chromeURL]; }

I have even added the following in .plist,

<key>LSApplicationQueriesSchemes</key> <array> <string>googlechrome</string> </array>

But still it is not working.

First Of All Make Sure You have installed Google Chrome browser application in your test iPhone device. If you are testing this code into simulator, then it will not work, because Google Chrome browser app can not be installed in Xcode simulator.On iPhone Device You can check the installation of chrome application in following way

//Check if Google Chrome is Instaled if ([[UIApplication sharedApplication] canOpenURL:chromeURL]) { //open URL in Google chrome browser app [[UIApplication sharedApplication] openURL:chromeURL]; } else { //Remove Google Chrome Scheme at start of application and open link in safari append http or https at start [[UIApplication sharedApplication] openURL:safariURL]; }

Luckily there are a few great simple site speed tools out there:

* [WebPageTest](http://www.webpagetest.org/): The industry standard for measuring site performance – results are collected from real browsers running common operating systems.
* [Pingdom](https://tools.pingdom.com/): a simple tool which makes makes the same measurements, yet method of testing is undocumented.
* [GTmetrix](https://gtmetrix.com/): gives you actionable insights about the best way to optimize your webpage speed.

Unless your website loads within a few seconds, the visitors who were to read your blog posts would leave with a bad ‘first impression’. Sometimes that impression can get so bad that those visitors would never come to your website again. It is eventually a loss as those visitors could have possibly turned into a paying customer in future.

Below are a few reasons which contribute to the problem and the corresponding remedy for that.

***Hosting***: · Hosting is a platform that allows you to store/host and manage your website on special computers called “Server”. The individual users or organizations make their website accessible to its users via the world wide web(WWW). There are three types of hosting available in the market namely; Shared Hosting Server, Virtual Private Server (V.P.S) and Dedicated Server. Depending on your needs, you can choose among the above.

***Size of the Web page:***The more are the media files on your website, the bigger will be the page size. The size of the web page is directly proportional to the web page load speed. If your website takes a long time to load, perhaps you would need to work on the size of your web page, because the internet users don’t have patience for your web page to take a long time to load, resulting in a high ‘bounce rate’.

***Content Management System (C.M.S)*:**Your content management system also plays a role in page load speed. If you want the highest speed, then a simple HTML page would work best, however, you would not be able to manage a blog by creating HTML pages yourself.

***Caching/Page Load Speed Plugin:*** Caching is another proven way to boost your website load speed. There are multiple plugins included with a Content management system like WordPress which you can install, some of the most frequently used are WP Rocket and “W3 Total Cache”. A cached web page is known to load faster while decreasing the load on the server by 75-80%.

I have shared a few links which you can click below to understand how to calculate [page load time of a website](https://dmdirection.com/website-page-load-speed/" \t "_blank) and that will take you to the respective websites.

[https://sitechecker.pro/page-size/](https://sitechecker.pro/page-size/" \t "_blank)

[https://smallseotools.com/websit...](https://smallseotools.com/website-page-size-checker/" \t "_blank)

[https://searchenginereports.net/...](https://searchenginereports.net/website-page-size-checker/" \t "_blank)

# Protocol Property Declaration

Protocols declare that conforming types must implement a property by including a [protocol property declaration](https://developer.apple.com/library/content/documentation/Swift/Conceptual/Swift_Programming_Language/Declarations.html?source=post_page---------------------------) in the declaration body. It have a special form of a variable declaration:

Var propertyName:Type {get set}

As with other protocol member declarations, these property declarations declare only the getter and setter requirements for types that conform to the protocol. As a result, you don’t implement the getter or setter directly in the protocol in which it is declared.

If a property declaration includes both the get and setkeywords, a conforming type can implement it with a stored variable property or a computed property that is both readable and writeable (that is, one that implements both a getter and a setter). However, that property declaration can’t be implemented as a constant property or a read-only computed property. If a property declaration includes only the getkeyword, it can be implemented as any kind of property.

# Examples

1. **Gettable — Constant Property**

protocol FullyNamed {  
 var fullName: String { get }  
}  
struct Detective: FullyNamed {  
 let fullName: String  
}  
let hercule = Detective(fullName: “Hercule Poirot”)  
print(hercule.fullName) // returns “Hercule Poirot”

**2. Getable — Variable Property**

protocol FullyNamed {  
 var fullName: String { get }  
}  
struct Detective: FullyNamed {  
 var fullName: String  
}  
var bond = Detective(fullName: “Bond”)  
print(bond.fullName) // returns “Bond”  
bond.fullName = “James Bond”  
print(bond.fullName) // returns “James Bond”

**3. Getable — Computed Property**

protocol FullyNamed {  
 var fullName: String { get }  
}  
struct Detective: FullyNamed {  
 fileprivate var name: String  
 var fullName: String {  
 return name  
 }  
}  
let batman = Detective(name: “Bruce Wayne”)  
print(batman.fullName) // returns “Bruce Wayne”

**4. Gettable — Private Set**

protocol FullyNamed {  
 var fullName: String { get }  
}  
public struct Detective: FullyNamed {  
 public private(set) var fullName: String  
 public init(fullName: String) {  
 self.fullName = fullName  
 }   
 public mutating func renameWith(fullName: String) {  
 self.fullName = fullName  
 }  
}  
var holmes = Detective(fullName: "Holmes")  
print(holmes.fullName) // returns "Holmes"  
holmes.renameWith(fullName: "Sherlock Holmes")  
print(holmes.fullName) // returns "Sherlock Holmes"

**5. Gettable & Settable — Computed Property**

protocol FullyNamed {  
 var fullName: String { get }  
}  
struct Detective: FullyNamed {  
 fileprivate var name: String  
 var fullName: String {  
 get {  
 return name  
 }  
 set {  
 name = newValue  
 }  
 }  
}  
var Payne = Detective(name: "Payne")  
print(Payne.fullName) // returns "Payne"  
Payne.fullName = "Max Payne"  
print(Payne.fullName) // returns "Max Payne"

**6. Gettable & Settable — Constant Property**

protocol FullyNamed {  
 var fullName: String { get set }  
}  
struct Detective: FullyNamed {  
 let fullName: String  
}  
let Rorschach = Detective(fullName: “Walter Joseph Kovacs”)  
// Error message: Type ‘Detective’ does not conform to protocol ‘FullyNamed’

**7. Gettable & Settable — only Get Defined**

protocol FullyNamed {  
 var fullName: String { get set }  
}  
struct Detective: FullyNamed {  
 private var name: String  
 var fullName: String {  
 return name  
 }  
}  
var constantine = Detective(name: “John Constantine”)  
// Error message: Type ‘Detective’ does not conform to protocol ‘FullyNamed’

# Type casting with protocols:

Get Property in protocols, does not force the conforming type to declare the get property only, it can declare it as **Set**property also, as explained above. Check this example for more clarity.

protocol FullyNamed{  
 var firstName: String {get}  
 var lastName: String {get set}  
}  
struct SuperHero: FullyNamed{  
 var firstName = “Super”  
 var lastName = “Man”  
}  
var dcHero = SuperHero()  
print(dcHero) // SuperHero(firstName: “Super”, lastName: “Man”)  
dcHero.firstName = “Bat”  
dcHero.lastName = “Girl”  
print(dcHero) // SuperHero(firstName: “Bat”, lastName: “Girl”)

Even though the FullyName protocol declares firstName as get only, still we can change its value. This is because FullyName does not stop the conforming type to set the property  
If we explicitly typecast the dcHero to FullyNamed, then compiler won’t allow us to set the firstName, since it has no knowledge about the setter.

var anotherDcHero:FullyNamed = SuperHero()  
print(anotherDcHero)  
anotherDcHero.firstName = “Bat”   
//ERROR: cannot assign to property: ‘firstName’ is a get-only property  
anotherDcHero.lastName = “Girl”  
print(anotherDcHero)

Since Apple introduced Protocol-Oriented Programming in 2015 WWDC, a lot of great article about the advantages of Protocol-Oriented Programming.

# The Confusion

Consider we have a protocol and a struct that conform to the protocol as shown in the snippet below.

Now let’s try to create an instance of MyStruct and set some value to myVar1and myVar2.

As expected, we do not get any error when setting value to myVar1. However, we also able to set value to myVar2 even though we make myVar2only gettable. The question then arises, “What is the difference between { get set } and { get }?”

# The Difference

Now let’s try to explicitly instantiate MyStruct as MyProtocol type and set value to both myVar1 and myVar2.

As you can see, now we are getting error when trying to set value to myVar2. What does this means? This means that{ get set } and { get } are actually providing us the encapsulation that we need at the protocol level. This is especially useful when we are using polymorphism in Protocol-Oriented Programming.

**get set**:

get set are **Computed Properties** Which **do not actually store a value**. Instead, they provide a getter and an optional setter to retrieve and set other properties and values indirectly

Additionally you can define Read-Only Computed Properties. A read-only computed property always returns a value, and can be accessed through dot syntax, but cannot be set to a different value

*Example get only property-*

var number: Double { return .pi\*2 }

**willSet didSet**:

willSet didSet are **Property Observers**

Property observers observe and respond to changes in a property’s value. Property observers are called every time a property’s value is set, even if the new value is the same as the property’s current value.

* willSet is called just before the value is stored.
* didSet is called immediately after the new value is stored.

*Example -*

var score: Int = 0 { willSet(newScore) { print("willSet score to \(newScore)") } didSet { print("didSet score to \(oldValue) new score is: \(score)") } } score = 10 //Output //willSet score to 10 //didSet score to 0 new score is: 10

An NSError object encapsulates information about an error condition in an extendable, object-oriented manner. It consists of a predefined error domain, a domain-specific error code, and a user info dictionary containing application-specific information.

Error is a Swift **protocol** which classes, structs and enums can and NSError does conform to.

A type representing an error value that can be thrown.

Any type that declares conformance to the Error protocol can be used to represent an error in Swift’s error handling system. Because the Error protocol has no requirements of its own, you can declare conformance on any custom type you create.

On Darwin platforms (macOS, iOS, tvOS, watchOS), Swift has a concept called [bridging](https://developer.apple.com/library/content/documentation/Swift/Conceptual/BuildingCocoaApps/WorkingWithCocoaDataTypes.html) which allows Swift types offered by the standard library to bridge to Objective-C types (usually vended by the Cocoa frameworks). Types which do this include

* Array ↔︎ NSArray
* Dictionary ↔︎ NSDictionary
* Set ↔︎ NSSet
* Error ↔︎ NSError
* etc. (see the above link for a more comprehensive list)

Bridging types requires compiler support, and the Error ↔︎ NSError bridging has [quite](https://github.com/apple/swift/blob/a716d403d62a512ae37fc5f2122cd4a559a17b3a/include/swift/AST/KnownFoundationEntities.def) [a few](https://github.com/apple/swift/blob/a912a12805e5f99f177f7f6a90bfa8f1d3913847/stdlib/public/core/ErrorType.swift)[components](https://github.com/apple/swift/blob/master/lib/Sema/DerivedConformanceError.cpp) ([among others](https://github.com/apple/swift/blob/c12a25d0fe887d81ddefcbdf846927aba0d76752/lib/SILGen/SILGen.h)). This bridging, though, allows the compiler to generally treat NSErrors as Errors and vice versa; in fact, as you can see in the snippet above, any Error can be cast to an NSError and back.

# NSError

Information about an error condition including a domain, a domain-specific error code, and application-specific information.

# Exceptions

The generic exception names are string constants defined in NSException.h and documented in *[Foundation Constants Reference](https://developer.apple.com/documentation/foundation/foundation_constants" \t "_self)*. These constants include the following:

* NSGenericException
* NSRangeException
* NSInvalidArgumentException
* NSInternalInconsistencyException
* NSObjectInaccessibleException
* NSObjectNotAvailableException
* NSDestinationInvalidException
* NSPortTimeoutException
* NSInvalidSendPortException
* NSInvalidReceivePortException
* NSPortSendException
* NSPortReceiveException

In short, **exceptions cause applications to crash if left unhandled**. They generally occur when trying to perform an operation on an object incorrectly, such as using an out-of-bounds index to access an array item, or passing nil to a method that doesn’t accept it. In other words, they are caused by developer mistakes.

Objective-C provides a single class to hold exception information called NSException. This class contains 3 key points of information:

* **name** identifies the type of exception that has occurred. As mentioned above, there is only one exception class, rather than different classes to represent different problems – so the name property is used as the high level categorization. Some common exception names you may come across include NSInvalidArgumentException and NSGenericException. The main predefined cocoa exception names are defined in [NSException.h](https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/Exceptions/Concepts/PredefinedExceptions.html" \l "//apple_ref/doc/uid/20000057-BCIGHECA" \t "_blank).
* **reason** is a short explanation of why the exception has been thrown. For example “+[Class Selector] unrecognized selector sent to class 0x10866fb88”.
* **userInfo** is an NSDictionary of additional information that can help to debug the problem. I haven’t seen any use of this from exceptions thrown by the cocoa framework, it seems to be more used by developers throwing their own exceptions to provide values of properties and variables and so on.

you can use @try @catch blocks to handle exceptions however appropriate.

@try { // Some potentially cataclysmic operation } @catch(NSException e) { if ([[e name] isEqualToString:@"ParseException"]) { // handle this specific exception } else { @throw; } } @finally { // clean up memory }

In the @catch block above you can see a use for the NSException name property. By checking the name of the exception, you can handle different types of exceptions that may occur from the same operation in different ways. Using a @throw; directive inside a @catch block will cause the exception to be rethrown where it can be handled further up the calling chain. The @finally block at the end will run its code regardless of if an exception occurred. This is useful for cleaning up memory or anything else that you need to ensure is performed before exiting the method.

Errors are used in quite a different way than exceptions. They don’t get thrown, and they don’t cause the application to crash. Instead, they are created to hold information about a failure, and then bubbled up through calling methods where it may be ‘handled’ in some way such as displaying a message to the user.

NSError class which provides these 3 properties:

* **domain** is a high level grouping of errors.
* **code** is used to distinguish different types of errors within a domain.
* **userInfo** is an NSDictionary containing additional information about the error.
* NSError \*error;

BOOL success = [writer writeTo:filePath withData:data error:&error];

if (success == NO) { /

/ display an alert to the user

}

Localizing apps into multiple languages comes with a variety of different tasks.

## NSLocalizedString

At the heart of string localization is the macro NSLocalizedString. There are three more lesser-known variants of it: NSLocalizedStringFromTable, NSLocalizedStringFromTableInBundle, and NSLocalizedStringWithDefaultValue. All of them use NSBundle‘s localizedStringForKey:value:table: method to do the heavy lifting.

Generally, all strings you want to show to the user in one form or another have to be localized. These can be simple labels or button titles, or more complex strings that are constructed at runtime from format strings and data.

When localizing strings, it is important to define one localizable string per instance, due to grammatical rules. For example, if you have an app where you need to show strings like “Paul invited you” and “You invited Paul,” it might be tempting to just localize the string “%@ invited %@” and then insert the localized string for “you” in the appropriate place.

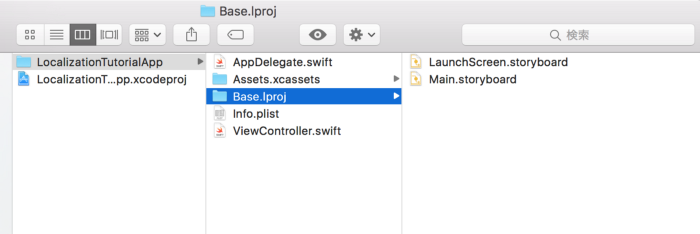
The correct approach would be to define two localizable strings: “%@ invited you” and “You invited %@.” Only then can translators account for the specific grammar rules of each language.

Localization is the process of making your app support other languages. In many cases, you make your app with English user interface first and then localize the app to other languages such as Japanese.

Before starting localization work, make sure you have “Use Base Internationalization” checkmark selected.

## What is “Base Internationalization” ?

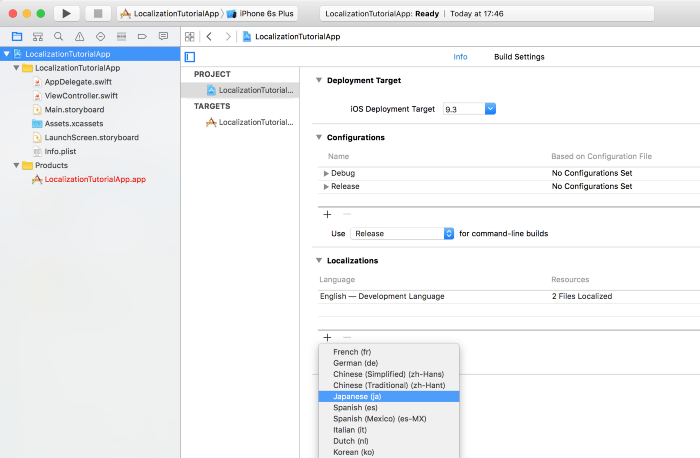
When you create new XCode project, XCode will automatically generate resources and the file structure including them for the default language.



This is so-called “Base” language. If you are making your app for global market, normally you want to use English texts in this “Base” language resources.

## Adding New Localization

Ok, we have the Base language resource structure by default. Let’s add new language support.   
Select your project file in Project Navigator, and select your project in the project and targets list. Open **Info** tab, and click “+” button under Localizations section. Then choose a language you want to support from the dropdown list shown.



XCode opens a dialog showing resources to be added for the new language. Pressing the **Finish** button will generate these files under the new language project folder named **[New Language].lproj.**(In this example I added Japanese support, so **ja.lproj** folder is created.)

## Where is Localizable.strings file ?

**Localizable.strings** file is where you add translation data as key-value pairs.  
Earlier versions of XCode used to generate a **Localizable.strings** file by default, and we were able to easily duplicate the **Localizable.strings** file for other languages.   
Recent versions of XCode doesn’t create Localizable.strings file by default.   
To add Localizable.strings file, go to **File->New->File** , choose **Strings File**under Resource tab of iOS, name it **Localizable.strings** , and create the file.

## Enumeration Syntax

You introduce enumerations with the enum keyword and place their entire definition within a pair of braces:

1. enum SomeEnumeration {
2. // enumeration definition goes here
3. }

Here’s an example for the four main points of a compass:

1. enum CompassPoint {
2. case north
3. case south
4. case east
5. case west
6. }

Multiple cases can appear on a single line, separated by commas:

1. enum Planet {
2. case mercury, venus, earth, mars, jupiter, saturn, uranus, neptune
3. }

var directionToHead = CompassPoint.west

directionToHead = .east

You can match individual enumeration values with a switch statement:

1. directionToHead = .south
2. switch directionToHead {
3. case .north:
4. print("Lots of planets have a north")
5. case .south:
6. print("Watch out for penguins")
7. case .east:
8. print("Where the sun rises")
9. case .west:
10. print("Where the skies are blue")
11. }
12. // Prints "Watch out for penguins"
13. enum Beverage: CaseIterable {
14. case coffee, tea, juice
15. }
16. let numberOfChoices = Beverage.allCases.count
17. print("\(numberOfChoices) beverages available")
18. // Prints "3 beverages available"
19. for beverage in Beverage.allCases {
20. print(beverage)
21. }
22. // coffee
23. // tea
24. // juice

This additional information is called an *associated value*, and it varies each time you use that case as a value in your code.

You can define Swift enumerations to store associated values of any given type, and the value types can be different for each case of the enumeration if needed. Enumerations similar to these are known as *discriminated unions*, *tagged unions*, or *variants* in other programming languages.

For example, suppose an inventory tracking system needs to track products by two different types of barcode. Some products are labeled with 1D barcodes in UPC format, which uses the numbers 0 to 9. Each barcode has a number system digit, followed by five manufacturer code digits and five product code digits. These are followed by a check digit to verify that the code has been scanned correctly:

In Swift, an enumeration to define product barcodes of either type might look like this:

1. enum Barcode {
2. case upc(Int, Int, Int, Int)
3. case qrCode(String)
4. }

You can then create new barcodes using either type:

1. var productBarcode = Barcode.upc(8, 85909, 51226, 3)

You can assign the same product a different type of barcode:

1. productBarcode = .qrCode("ABCDEFGHIJKLMNOP")

*An*enumeration*defines a common type for a group of related values and enables you to work with those values in a type-safe way within your code.*

|  |
| --- |
| Func dayType(for day: String) -> String { |
|  | switch day { |
|  | case: "Saturday", "Sunday": return "Weekend" |
|  | case: "Monday", "Tuesday", "Wednessday", "Thursday", "Friday": return "Weekday" |
|  | default: return "This is not a valid date" |
|  | } |
|  | } |
|  |  |
|  | let result1 = dayType(for: "Sunday") //will return "Weekend" |
|  | let result2 = dayType(for: "Thursday") //will return "Weekday" |
|  |  |
|  | let result3 = dayType(for: "fridayys") //will return "This is not a valid date" |

|  |
| --- |
| enum Day { |
|  | case Sunday |
|  | case Monday |
|  | case Tuesday |
|  | case Wednessday |
|  | case Thursday |
|  | case Friday |
|  | case Saturday |
|  | } |
|  |  |
|  | /\*\* Rewriting the date type function \*/ |
|  | func dayType(for day: Day) { |
|  | switch day { |
|  | case Day.Saturday, Day.Sunday: |
|  | return "Weekend" |
|  | case Day.Monday, Day.Tuesday, Day.Wednessday, Day.Thursday, Day.Friday |
|  | return "Weekday" |
|  | } |
|  | } |
|  |  |
|  | let result1 = dayType(for: Day.Sunday) //will return "Weekend" |

## ****Enum Values****

We can assign value to each enum case. This is useful if the enum itself indeed relates to something

|  |  |  |  |
| --- | --- | --- | --- |
|  | | enum Week: String { | |
|  | | case Sunday = "Weekday" | |
|  | | case Monday = "Weekday" | |
|  | | case Tuesday = "Weekday" | |
|  | | case Wednessday = "Weekday" | |
|  | | case Thursday = "Weekday" | |
|  | | case Friday = "Weekend" | |
|  | | case Saturday = "Weekend" | |
|  | | } | |
| // You can also map to strings |
|  | enum Week: String { | |
|  | case Sunday = "Weekday" | |
|  | case Monday = "Weekday" | |
|  | case Tuesday = "Weekday" | |
|  | case Wednessday = "Weekday" | |
|  | case Thursday = "Weekday" | |
|  | case Friday = "Weekend" | |
|  | case Saturday = "Weekend" | |
|  | } | |

## Enum Associate Values

Associate values are a fantastic way of attaching additional information to an enum case. Say you’re writing a trade engine, there are two different possible trade types, Buy and Sell that will have a specific stock and amount. We can represent this using associate enum values like below

|  |
| --- |
| enum Trade { |
|  | case Buy(String, amount) |
|  | case Sell(String, Int) |
|  | } |
|  |  |
|  | Trade.Buy("Firstbank PLC", 300) |
|  | Trade.Sell("Firstbank PLC", 700) |
|  |  |
|  | /\*\* associate with labels \*/ |
|  | enum Trade { |
|  | case Buy(stock: String, amount: Int) |
|  | case Sell(stock: String, amount: Int) |
|  | } |
|  |  |
|  | Trade.Buy(stock: "Firstbank PLC", amount: 300) |
|  | Trade.Sell(stock: "Firstbank PLC", amount: 700) |

## Enum Methods

We can also define methods on an enum like below

|  |
| --- |
| enum AppleDevice { |
|  | case iPad |
|  | case iPhone |
|  | case AppleTv |
|  | case AppleWatch |
|  |  |
|  | func description() -> String { |
|  | return "This is an apple device" |
|  | } |
|  | } |
|  |  |
|  | AppleDevice.iPad.description() //will return "This is an apple device" |
|  | AppleDevice.iPhone.description() //will return "This is an apple device" |

|  |
| --- |
| struct User { |
|  | var username: String |
|  | var email: String |
|  | var name: String |
|  | } |

var user = User(username: "farhansyed", email: "farhansyed123@gmail.com", name: "Farhan Syed")

To read a property you simply do this:

print(user.username)

Structs are **value type**, meaning it’s value is copied when assigned or passed.

Let’s create another instance of User, call it user2.

var user2 = User(username: "farhan", email: "someEmail@gmail.com", name: "Farhan")

Create a variable user3 and make it equal to user2.

var user3 = user2

Now let’s manipulate user3’s username.

|  |
| --- |
| user2.username = "FarhanSyed" |

|  |  |
| --- | --- |
|  | print("user2 username is different: \(user2.username)") |
|  | print("user3 username is unchanged: \(user3.username)") |

If you don’t understand my example take a look at [Apple’s](https://developer.apple.com/swift/blog/?id=10&source=post_page---------------------------).

|  |
| --- |
|  |
|  | struct S { var data: Int = -1 } |
|  | var a = S() |
|  | var b = a // a is copied to b |
|  | a.data = 42 // Changes a, not b |
|  | print("\(a.data), \(b.data)") // prints "42, -1" |

Whenever you create an enum or a struct , it is a value type and a Class is a reference type.

* ***Value Type***: Struct , Enum
* ***Reference Type***: Class

When you pass a class object around your program, you are actually passing a reference to that object, so different parts of your program can share and modify your object. When you pass a structure [ or enum] around your program, what gets passes around is a copy of the structure. So modifications to structures don’t get shared.

*One of the major benefits of value types is that they are thread-safe not requiring any synchronization.*

An instance of a class is traditionally known as an *object*. But it is better to use the term ***Instance***as we are dealing with struct and enum as well.

## What are the common factors between struct and class?

* Define properties to store values
* Define methods to provide functionality
* Define subscripts to provide access to their values using subscript syntax
* Define initializers to set up their initial state
* Be extended to expand their functionality beyond a default implementation
* Conform to protocols to provide standard functionality of a certain kind

## Classes have additional capabilities that structures don’t have:

* Inheritance enables one class to inherit the characteristics of another. Struct or enum cannot do inheritance. But they can confirm to protocols.
* Type casting enables you to check and interpret the type of a class instance at runtime.
* Deinitializers enable an instance of a class to free up any resources it has assigned.
* Reference counting allows more than one reference to a class instance.

In Objective — C, everything subclasses from NSObject. NSString, NSArray etc are all reference types. On the other hand, swift is rich in value types. String, Array , Dictionary etc are all structs in swift, which is a value type. There are so many advantages in using a value type over a reference type.

* Structs are much safer and bug-free, especially in a multithreaded environment. Swift value types are kept in the stack. In a process, each thread has its own stack space, so no other thread will be able to access your value type directly. Hence no race conditions, locks, deadlocks or any related thread synchronization complexity.
* Use classes if you want reference types. Use structs if you want value types.
* Even though struct and enum don’t support inheritance, they are great for protocol-oriented programming. A subclass inherits all the required and unwanted functionalities from the superclass and is a bad programming practice. Better to use a struct with protocol-oriented programming concept which fixes the above-said issue.
* Class does support Inheritance. Class is a reference type and is stored in the heap part of memory which makes a class comparatively slower than a struct in terms of performance. Unlike a class, a struct is created on the stack. So, it is faster to instantiate (and destroy) a struct than a class. Unless struct is a class member in which case it is allocated in heap, along with everything else.
* Value types do not need dynamic memory allocation or reference counting, both of which are expensive operations. At the same time methods on value types are dispatched statically. These create a huge advantage in favor of value types in terms of performance.

Consider the following recommendations to help choose which option makes sense when adding a new data type to your app.

* Use structures by default.
* Use classes when you need Objective-C interoperability.
* Use classes when you need to control the identity of the data you're modeling.
* Use structures along with protocols to adopt behavior by sharing implementations.

### Choose Structures by Default

Use structures to represent common kinds of data. Structures in Swift include many features that are limited to classes in other languages: They can include stored properties, computed properties, and methods. Moreover, Swift structures can adopt protocols to gain behavior through default implementations. The Swift standard library and Foundation use structures for types you use frequently, such as numbers, strings, arrays, and dictionaries.

* An open class is *accessible* and *subclassable* outside of the defining module. An open class member is *accessible* and *overridable* outside of the defining module.
* A public class is *accessible* but *not subclassable* outside of the defining module. A publicclass member is *accessible* but *not overridable* outside of the defining module.

Enable an entity to be used outside the defining module (target). You typically use open or publicaccess when specifying the public interface to a framework.

// First.framework – A.swift open class A {}

// First.framework – B.swift public class B: A {} // ok

// Second.framework – C.swift import First internal class C: A {} // ok

// Second.framework – D.swift import First internal class D: B {} // error: B cannot be subclassed

### 2. internal

Enables an entity to be used within the defining module (target). You typically use internal access when defining an app’s or a framework’s internal structure.

// First.framework – A.swift internal struct A {}

// First.framework – B.swift A() // ok

// Second.framework – C.swift import First A() // error: A is unavailable

### 3. fileprivate

Restricts the use of an entity to its defining source file. You typically use fileprivate access to hide the implementation details of a specific piece of functionality when those details are used within an entire file.

// First.framework – A.swift internal struct A { fileprivate static let x: Int } A.x // ok

// First.framework – B.swift A.x // error: x is not available

### 4. private

Restricts the use of an entity to its enclosing declaration. You typically use private access to hide the implementation details of a specific piece of functionality when those details are used only within a single declaration.

// First.framework – A.swift internal struct A { private static let x: Int internal static func doSomethingWithX() { x // ok } } A.x // error: x is unavailable

# Swift 4 / Swift 5

As per mentioned in the [Swift Documentation - Access Control](https://developer.apple.com/library/content/documentation/Swift/Conceptual/Swift_Programming_Language/AccessControl.html), Swift has 5 Access Controls:

* ***open*** and ***public***: can be accessed from their module's entities and any module's entities that imports the defining module.
* ***internal***: can only be accessed from their module's entities. It is the default access level.
* ***fileprivate*** and ***private***: can only be accessed in limited within a limited scope where you define them.

# What is the difference between *open* and *public*?

**open** is the same as public in previous versions of Swift, they allow classes from other modules to use and inherit them, i.e: they can be subclassed from other modules. Also, they allow members from other modules to use and override them. The same logic goes for their modules.

**public** allow classes from other module to use them, but not to inherit them, i.e: they cannot be subclassed from other modules. Also, they allow members from other modules to use them, but NOT to override them. For their modules, they have the same open's logic (they allow classes to use and inherit them; They allow members to use and override them).

# What is the difference between *fileprivate* and *private*?

**fileprivate** can be accessed from the their entire files.

**private** can only be accessed from their single declaration and to extensions of that declaration that are in the same file; For instance:

// Declaring "A" class that has the two types of "private" and "fileprivate": class A { private var aPrivate: String? fileprivate var aFileprivate: String? func accessMySelf() { // this works fine self.aPrivate = "" self.aFileprivate = "" } } // Declaring "B" for checking the abiltiy of accessing "A" class: class B { func accessA() { // create an instance of "A" class let aObject = A() // Error! this is NOT accessable... aObject.aPrivate = "I CANNOT set a value for it!" // this works fine aObject.aFileprivate = "I CAN set a value for it!" } }

# What are the differences between Swift 3 and Swift 4 Access Control?

As mentioned in the [SE-0169 proposal](https://github.com/apple/swift-evolution/blob/master/proposals/0169-improve-interaction-between-private-declarations-and-extensions.md), the only refinement has been added to Swift 4 is that the private access control scope has been expanded to be accessible from extensions of that declaration in the same file; For instance:

struct MyStruct { private let myMessage = "Hello World" } extension MyStruct { func printMyMessage() { print(myMessage) // In Swift 3, you will get a compile time error: // error: 'myMessage' is inaccessible due to 'private' protection level // In Swift 4 it should works fine! } }

**Swift 3.0 provides five different access controls:**

1. open
2. public
3. internal
4. fileprivate
5. private

**Open** access and **public** access enable entities to be used within any source file from their defining module, and also in a source file from another module that imports the defining module. You typically use open or public access when specifying the public interface to a framework.

**Internal** access enables entities to be used within any source file from their defining module, but not in any source file outside of that module. You typically use internal access when defining an app’s or a framework’s internal structure.

**File-private** access restricts the use of an entity to its own defining source file. Use file-private access to hide the implementation details of a specific piece of functionality when those details are used within an entire file.

**Private** access restricts the use of an entity to the enclosing declaration. Use private access to hide the implementation details of a specific piece of functionality when those details are used only within a single declaration.

***Open****access is the highest (least restrictive) access level and****private****access is the lowest (most restrictive) access level.*

**Default Access Levels**

All entities in your code (with a few specific exceptions) have a default access level of internal if you do not specify an explicit access level yourself. As a result, in many cases you do not need to specify an explicit access level in your code.

**The release note on the topic:**

Classes declared as public can no longer be subclassed outside of their defining module, and methods declared as public can no longer be overridden outside of their defining module. To allow a class to be externally subclassed or a method to be externally overridden, declare them as open, which is a new access level beyond public. Imported Objective-C classes and methods are now all imported as open rather than public. Unit tests that import a module using an @testable import will still be allowed to subclass public or internal classes as well as override public or internal methods. (SE-0117)

The required keyword means that inheriting classes must provide an implementation of the method. However, the language makes an exception for required initializers:

You do not have to provide an explicit implementation of a required initializer if you can satisfy the requirement with an inherited initializer.

“The use of the required modifier ensures that you provide an explicit or inherited implementation of the initializer requirement on all subclasses of the conforming class, such that they also conform to the protocol.”

A convenience required initializer is an initializer that is enforced onto all subclasses but is not the designated initializer. This means that said initializer will eventually call a designated initializer in its initialization chain.

## Designated Initialisers

A designated initialiser is the canonical initializer for a class and the one which all required and convenience initialisers should call. The [Docs](https://developer.apple.com/library/ios/documentation/swift/conceptual/Swift_Programming_Language/Initialization.html) say:

Designated initializers are the primary initializers for a class. A designated initializer fully initializes all properties introduced by that class and calls an appropriate superclass initializer to continue the initialization process up the superclass chain.

## Convenience Initialisers

A convenience initialiser is an initializer that sets up certain configuration information on a class...conveniently. Documentation:

Convenience initializers are secondary, supporting initializers for a class. You can define a convenience initializer to call a designated initializer from the same class as the convenience initializer with some of the designated initializer’s parameters set to default values. You can also define a convenience initializer to create an instance of that class for a specific use case or input value type.

You do not have to provide convenience initializers if your class does not require them. Create convenience initializers whenever a shortcut to a common initialization pattern will save time or make initialization of the class clearer in intent

## Required Initialisers

Required initializers can be thought of as a binding contract between a parents interface and subsequent subclasses. Its your means of enforcing that all your children are aware of and implement a certain set of initialisers.

Write the required modifier before the definition of a class initializer to indicate that every subclass of the class must implement that initializer:

Swift protocols on their side do not allow optional methods. But if you are making an app for macOS, iOS, tvOS or watchOS you can add the @objc keyword at the beginning of the implementation of your protocol and add @objc follow by optional keyword before each methods you want to be optional.

|  |
| --- |
| @objc protocol MyProtocol { |
|  | @objc optional func anOptionalMethod() |
|  | } |

To expose a method to Objective-C, just write @objc before its name like this:

class MyController: UIViewController {

@objc func authenticateUser() {

}

}

private mean it visible only in Swift. so use @objc to visible in Objective-C. If you have a func to selector a private func in swift, it is required.

The @objc attribute makes your Swift API available in Objective-C and the Objective-C runtime.

A Swift class or protocol must be marked with the **@objc** attribute to be accessible and usable in Objective-C. This attribute tells the compiler that this piece of Swift code can be accessed from Objective-C. If your Swift class is a descendant of an Objective-C class, the compiler automatically adds the @objc attribute for you.

# Using Objective-C Classes in Swift

### Step 4: Build your Objective-C Class

In CustomObject.h

#import <Foundation/Foundation.h>

@interface CustomObject : NSObject

@property (strong, nonatomic) id someProperty;

* + (void) someMethod;

@end

In CustomObject.m

#import "CustomObject.h"

@implementation CustomObject

* + (void) someMethod {

NSLog(@"SomeMethod Ran");

}

@end

### Step 5: Add Class to Bridging-Header

In YourProject-Bridging-Header.h:

#import "CustomObject.h"

### Step 6: Use your Object

In SomeSwiftFile.swift:

var instanceOfCustomObject: CustomObject = CustomObject() i

nstanceOfCustomObject.someProperty = "Hello World"

println(instanceOfCustomObject.someProperty)

instanceOfCustomObject.someMethod()

# Using Swift Classes in Objective-C

### Step 1: Create New Swift Class

Add a .swift file to your project, and name it MySwiftObject.swift.

In MySwiftObject.swift:

import Foundation class MySwiftObject : NSObject { var someProperty: AnyObject = "Some Initializer Val" init() {} func someFunction(someArg:AnyObject) -> String { var returnVal = "You sent me \(someArg)" return returnVal } }

### Step 2: Import Swift Files to ObjC Class

In SomeRandomClass.m:

#import "<#YourProjectName#>-Swift.h"

The file:<#YourProjectName#>-Swift.h should already be created automatically in your project, even if you can not see it.

### Step 3: Use your class

MySwiftObject \* myOb = [MySwiftObject new]; NSLog(@"MyOb.someProperty: %@", myOb.someProperty); myOb.someProperty = @"Hello World"; NSLog(@"MyOb.someProperty: %@", myOb.someProperty); // original NSString \* retString = [myOb someFunction:@"Arg"]; // xcode10 expands the external arg here NSString \* retString = [myOb someFunctionWithSomeArg:@"Arg"];| NSLog(@"RetString: %@", retString);

There are two keywords to keep in mind when dealing with interoperability:

* @objc means you want your Swift code (class, method, property, etc.) to be visible from Objective-C.
* dynamic means you want to use Objective-C dynamic dispatch.

In Swift 3 and earlier, dynamic also implied @objc. New in Swift 4, dynamic only means dynamic dispatch and says nothing about Objective-C visibility.

You probably know that the Swift language defines a number of attributes, such as objc, escaping, and available. It also defines a range of declaration modifiers.

As the name implies, a declaration modifier modifies a declaration. For example, by marking a class declaration with the final keyword, we inform the compiler that the class cannot be subclassed. This allows the compiler to make a number of optimizations to increase performance.

One other declaration modifier you may have come across is dynamic. In this tutorial, we explore what the meaning is of the dynamic declaration modifier and when you may want or need to use it.

Static and virtual dispatch are much faster than dynamic dispatch. Even though we are talking nanoseconds, the net result can be dramatic. If you are wondering, inlined access beats static and virtual dispatch hands down.

Right. But I am only using Swift in my project. Are you? The frameworks that power iOS, tvOS, macOS, and watchOS applications are written in Objective-C. And many features we have come accustomed to are only possible because of the dynamic Objective-C runtime, including Core Data and Key-Value Observing.

Using dynamic tells Swift to always refer to Objective-C dynamic dispatch. This is required for things like Key-Value Observing to work correctly. When the Swift function is called, it refers to the Objective-C runtime to dynamically dispatch the call.

@synthesize will generate getter and setter methods for your property. @dynamic just tells the compiler that the getter and setter methods are implemented not by the class itself but somewhere else (like the superclass or will be provided at runtime).

Uses for @dynamic are e.g. with subclasses of NSManagedObject (CoreData) or when you want to create an outlet for a property defined by a superclass that was not defined as an outlet.

@dynamic also can be used to delegate the responsibility of implementing the accessors. If you implement the accessors yourself within the class then you normally do not use @dynamic.

Super class:

@property (nonatomic, retain) NSButton \*someButton; ... @synthesize someButton;

Subclass:

@property (nonatomic, retain) IBOutlet NSButton \*someButton; ... @dynamic someButton;

***A lazy stored property is a property whose initial value is not calculated until the first time it is used. You indicate a lazy stored property by writing the lazy modifier before its declaration.***

*You must always declare a lazy property as a variable (with the var keyword), because its initial value might not be retrieved until after instance initialization completes. Constant properties must always have a value*before*initialization completes, and therefore cannot be declared as lazy.*

|  |
| --- |
| import UIKit |
|  |  |
|  |  |
|  | struct InterviewCandidate { |
|  | var isiOS:Bool? |
|  |  |
|  | lazy var iOSResumeDescription: String = { |
|  | return "I am an iOS developer" |
|  | }() |
|  | lazy var androidResumeDescription: String = { |
|  | return "I am an android developer" |
|  | }() |
|  | } |
|  |  |
|  | var person1 = InterviewCandidate() |
|  | person1.isiOS = true |
|  |  |
|  | if person1.isiOS! { |
|  | print(person1.iOSResumeDescription) |
|  | } else { |
|  | print(person1.androidResumeDescription) |
|  |  |
|  | } |

So , in the following code, the person is an iOS developer and the lazy variable *iOSResumeDescription* will be initialized when called for printing . *androidResumeDescription*will be nil.

# Lazy Stored Property vs Stored Property

There are a few advantage of a lazy property over a stored property.

1. The closure associated to the lazy property is executed only if you read that property. So if for some reason that property is not used (maybe because of some decision of the user) you avoid unnecessary allocation and computation.
2. You can populate a lazy property with the value of a stored property.
3. **Important to note:**You can use self inside the closure of a lazy property. It will not cause any retain cycles. The reason is that the immediately applied closure {}() is considered @noescape. It does not retain the captured self.

# Lazy Stored Property vs Stored Property

There are a few advantage of a lazy property over a stored property.

1. The closure associated to the lazy property is executed only if you read that property. So if for some reason that property is not used (maybe because of some decision of the user) you avoid unnecessary allocation and computation.
2. You can populate a lazy property with the value of a stored property.
3. **Important to note:**You can use self inside the closure of a lazy property. It will not cause any retain cycles. The reason is that the immediately applied closure {}() is considered @noescape. It does not retain the captured self.

struct Person { var age = 16 lazy var fibonacciOfAge: Int = { return fibonacci(of: self.age) }() func fibonacci(of num: Int) -> Int { if num < 2 { return num } else { return fibonacci(of: num - 1) + fibonacci(of: num - 2) } } }

Once that code is written, you can use it like this:

var singer = Person() print(singer.fibonacciOfAge)

Remember, the point of lazy properties is that they are computed only when they are first needed, after which their value is saved. This means if you create 1000 singers and never touch their fibonacciOfAgeproperty, your code will be lightning fast because that lazy work is never done.

If you’re new to Swift, or you’re new to Swift 3, you may have noticed that some closures now have a new keyword: escaping

## Non Escaping Closures

Non-escaping closures have a very clear lifecycle and have become the default closure type in Swift 3 because of it. A non-escaping closure is simple: It’s passed into a function (or other containing scope), the function/scope executes that closure, and the function returns. The closure cannot return or finish executing after the body of the calling function has returned. Due to that fact, the compiler is able to optimize non-escaping closures over escaping closures.

## Escaping Closures

Escaping closures are more complex — they’re named that way because their return can escape the body of the calling function; the closure can be stored somewhere that will be retained after the calling function has finished executing.

Common use cases for this are:

1. Asynchronous calls; networking.
2. Functions stored as variables; think actions and supplied callbacks.
3. Scheduling tasks on a dispatch queue.

## Exceptions

Closures in Swift are non-escaping by default — except when they’re not.

Functions that are stored as variables are escaping — by definition, they are retained after they are executed, and can be executed multiple times.

All optional closures are escaping. Technically, when being used as an optional, it’s a parameter of the optional type: Optional<() -> (Optional<StatusEffect>, Damage).

In fact, all closures parameterized in this manner are escaping, such as when being used in:

* enums
* tuples

# What are closures?

[Closures*are self-contained blocks of functionality that can be passed around and used in your code.*](https://developer.apple.com/library/content/documentation/Swift/Conceptual/Swift_Programming_Language/Closures.html?source=post_page---------------------------#//apple_ref/doc/uid/TP40014097-CH11-ID94)

In Swift 1.x and Swift 2.x, closure parameter was **@escaping** by default, means that closure can be escape during the function body execution. if don’t want to escape closure parameters mark it as **@nonescaping**.

In Swift 3.x, Apple made a change: closure parameters became **@nonescaping** by default, closure will also be execute with the function body, if wanna escape closure execution mark it as **@escaping**. Why Apple made this change?🤔 Let’s discuss this at the end of the discussion 😋.

# 1. @nonescaping closures:

When passing a closure as the function argument, the closure gets execute with the function’s body and returns the compiler back. As the execution ends, the passed closure goes out of scope and have no more existence in memory.

Lifecycle of the @nonescaping closure:  
1. Pass the closure as function argument, during the function call.  
2. Do some additional work with function.  
3. Function runs the closure.  
4. Function returns the compiler back.

# Example:

func getSumOf(array:[Int], handler: ((Int)->Void)) {  
 //step 2  
 var sum: Int = 0  
 for value in array {  
 sum += value  
 }  
   
 //step 3  
 handler(sum)  
 }  
   
 func doSomething() {  
 //setp 1  
 self.getSumOf(array: [16,756,442,6,23]) { [weak self](sum) in  
 print(sum)  
 //step 4, finishing the execution  
 }  
 }//It will print the sumof all the given numbers.

Here, we just called the function with a closure, that gets executed at the end of the function’s body.   
So we are not escaping the execution of the closure. As the step 4 get executed closure will have no existence in the memory.

# 2. @escaping closures:

When passing a closure as the function argument, the closure is being preserve to be execute later and function’s body gets executed, returns the compiler back. As the execution ends, the scope of the passed closure exist and have existence in memory, till the closure gets executed.   
There are several ways to escaping the closure:

* **Storage:**When you need to preserve the closure in storage that exist in the memory, past of the calling function get executed and return the compiler back. (Like waiting for the API response)
* **Asynchronous Execution:**When you are executing the closure asynchronously on despatch queue, the queue will hold the closure in memory for you, to be used in future. In this case you have no idea when the closure will get executed.

When you will try to use the closure with these option the swift compiler will show the error.

https://miro.medium.com/max/60/0*kymVBBDBUn8uJ1F0.png?q=20

https://miro.medium.com/max/400/0*kymVBBDBUn8uJ1F0.png

Lifecycle of the @escaping closure:   
1. Pass the closure as function argument, during the function call.   
2. Do some additional work in function.   
3. Function execute the closure asynchronously or stored.   
4. Function returns the compiler back.

# Example 1 (Storage) :

var complitionHandler: ((Int)->Void)? func getSumOf(array:[Int], handler: @escaping ((Int)->Void)) {  
 //step 2 //here I'm taking for loop just for example, in real case it'll be something else like API call  
 var sum: Int = 0  
 for value in array {  
 sum += value  
 }//step 3  
 self.complitionHandler = handler  
 }  
   
 func doSomething() {  
 //setp 1  
 self.getSumOf(array: [16,756,442,6,23]) { [weak self](sum) in  
 print(sum)  
 //step 4, finishing the execution  
 }  
 }//Here we are storing the closure for future use.  
//It will print the sumof all the passed numbers.

# Example 2 (Asynchronous Execution) :

func getSumOf(array:[Int], handler: @escaping ((Int)->Void)) {  
 //step 2  
 var sum: Int = 0  
 for value in array {  
 sum += value  
 }  
 //step 3  
 Globals.delay(0.3, closure: {  
 handler(sum)  
 })  
 }  
   
 func doSomething() {  
 //setp 1  
 self.getSumOf(array: [16,756,442,6,23]) { [weak self](sum) in  
 print(sum)  
 //step 4, finishing the execution  
 }  
 }//Here we are calling the closure with the delay of 0.3 seconds  
//It will print the sumof all the passed numbers.

Here we are with the meaning of the **@escaping** attribute. So, when you need to escape the execution of the closure use **@escaping** attribute in Swift 3. The above shown error of compiler will disappear after making closure as escaping by using the **@escaping** attribute.

# Why they made @nonescaping by default?

There are many different benefits of making **non-escaping** as by default. The most important benefits are performance and code optimisation by the compiler, because if the compiler knows that the closure is **non-escaping**, will take care about the memory allocation for the closure.

And another one is, we can use self without problems in **non-escaping**closures because the closure executes before the function returns so the self will be there by sure. We don’t need to use **weak self** this is the additional feature of it 😉.

In Swift 3, closures are non-escaping by default, can be use @escaping if not what we want. Non-escaping closure defiantly will execute before the function returns.

📣 Always remember to use weak self while using a closure. ⚠️

Both SOAP Version 1.1 and SOAP Version 1.2 are World Wide Web Consortium (W3C) standards. Web services can be deployed that support not only SOAP 1.1 but also support SOAP 1.2. Some changes from SOAP 1.1 that were made to the SOAP 1.2 specification are significant, while other changes are minor.

With a backslash before the double quote you want to insert in the String:

let sentence = "They said \"It's okay\", didn't they?"

Now sentence is:

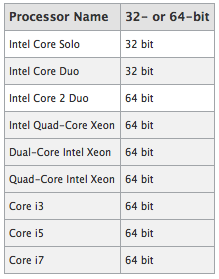
They said "It's okay", didn't they?

It's called "escaping" a character: you're using its literal value, it will not be interpreted.

An abbreviation for Active Server Method File, a [file](https://www.lifewire.com/what-is-a-file-2625878) with the ASMX [file extension](https://www.lifewire.com/what-is-a-file-extension-2625879) is an ASP.NET Web Service Source file.

Basically if you have anything newer than an 2008 MacBook you have a 64 bit processor and OS…

Here is the cheat table



64-bit processors can access much more memory than 32-bit processors. They can also potentially handle processor requests more efficiently than their 32-bit counterparts. Apple decided years ago to transition into 64-bit processors exclusively for the Mac, as well as the iPhone and iPad.

macOS has been 64-bit and 32-bit since 2005, and generally speaking if you purchased a new Mac in 2007 or later, its processor is 64-bit.

As of January 2018, all Mac App Store submissions must be 64-bit. Apple has started displaying alerts when you launch 32-bit apps on your Mac, too.

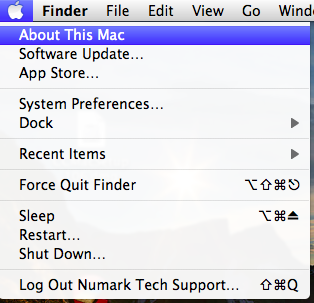
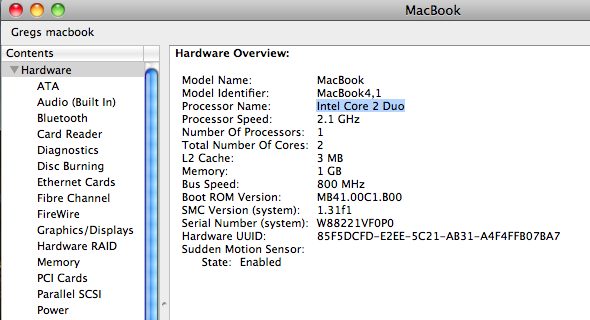
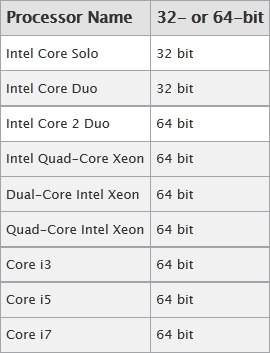
macOS has been 64-bit for years, but also supports 32-bit apps. That’s going to change soon, although Apple hasn’t announced an exact date yet.

* A 32-bit operating system typically only allows for up to about 4 gigabytes of RAM to be utilized by the OS or any of the installed programs.  Therefore, it makes sense to only install 32-bit programs on a 32-bit operating system, as a 64-bit program would not be able to access any more RAM than a 32-bit program on a 32-bit operating system.
* A 64-bit operating system allows for much higher RAM access and capacity.  See your specific operating system specifications for further information on how much memory (RAM) can be utilized by your specific OS.

Although both 32-bit and 64-bit programs can be installed on a 64-bit operating system, a 32-bit program will only be able to access up to about the first 4 gigabytes of memory (RAM).

By contrast, a native 64-bit program installed on a 64-bit operating system will be able to utilize a much higher amount of memory (RAM), allowing for much larger projects and more memory-intensive operations.

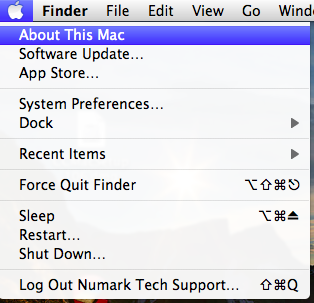
To determine if you have a 64-bit capable processor:

1. Click on the **Apple**creen shot 2013-04-17 at 4.09.21 PM menu and then choose **About This Mac**.  
     
   
2. In the window that opens, choose **More Info...**  
     
   
3. Locate the **Processor Name**.  
     
   
4. Once you have located the processor name, compare it to the table below to determine if the processor is 32-bit or 64-bit capable.  
     
   

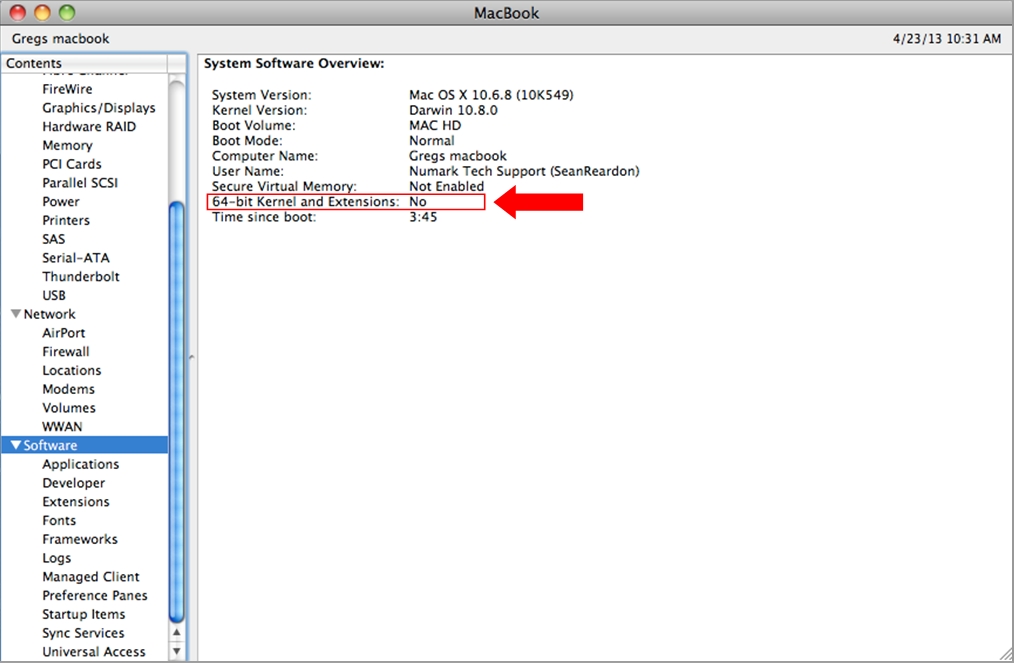
To determine if your Mac OS is running in 32-bit or 64-bit mode:

**NOTE:** Mac Mountain Lion 10.8 and later do not support running in 32-bit mode.

1. Click on the **Apple**creen shot 2013-04-17 at 4.09.21 PM menu and then choose **About This Mac**.



1. In the window that opens, choose **More Info...**  
     
   
2. In the Contents pane, choose **Software**.  
   * If 64-bit Kernel and Extensions: is set to Yes, then you are running a 64-bit version of the operating system.
   * If 64-bit Kernel and Extensions: is set to No, then you are running a 32-bit version of the operating system.



To Switch Between 32-bit and 64-bit mode:

1. **Power OFF** the computer.
2. While powering ON the Mac:
   * If your Mac uses the 32-bit mode by default, but [supports](https://www.alesis.com/kb/article/1616#mac_processor) the 64-bit mode, you can start up in 64-bit mode by **holding the 6 and 4 keys during startup**.
   * If your Mac uses the 64-bit mode by default, you can start up in 32-bit mode by **holding the 3 and 2 keys during startup**.

**NOTE:** Your Mac will revert to the default mode the next time you reboot it.

64-bit computer, a processor can use—hold on to your hats—16 exabytes of memory. (An exabyte is 1,000 petabytes, a petabyte is 1,000 terabytes, and a terabyte is 1,000 gigabytes.)

A process server is a person who delivers a court order and/or documents that compel a defendant’s presence in court.

Process servers deliver a variety of legal documents, including writs, subpoenas to testify in court, a summons to appear in court, and formal complaints. In addition to serving these documents, process servers can also assist with filing appropriate documents in court, retrieving documents for you, and helping you track down a defendant.

Process servers deliver, or serve, legal papers to people who are required to appear in civil court, usually for personal injury or divorce cases.

The process server is employed by the court, sheriff, law firms, private investigator, process server company, or may even work freelance. Every workday is different, but every day legal documents must be delivered to people in person.

Papers must be delivered in a timely fashion or deadlines are broken resulting in court delays, fees, and sometimes court cases are ruined. Process servers need to perform their work quickly, which is sometimes difficult because many people, like unenthusiastic divorcees, slimy slumlords, or negligent drivers, don’t want to be found.

\

| **Version** | **Download** | **Installed** |
| --- | --- | --- |
| Xcode 5.0 | 2.05 GB | 4.37 GB |
| Xcode 6.0 | 2.62 GB | 5.18 GB |
| Xcode 7.0 | 3.85 GB | 7.79 GB |
| Xcode 8.0 | 4.41 GB | 9.97 GB |
| Xcode 9.0 | 5.37 GB | 10.41 GB |

# NSXMLParser

An NSXMLParser notifies its delegate about the items (elements, attributes, CDATA blocks, comments, and so on) that it encounters as it processes an XML document. It does not itself do anything with those parsed items except report them. It also reports parsing errors.

[CDATA](http://www.w3.org/TR/REC-xml/#sec-cdata-sect) stands for [Character Data](http://www.w3.org/TR/REC-xml/#dt-chardata) and it means that the data in between these strings includes data that could be interpreted as XML markup, but should not be.

CDATA is defined as blocks of text that are not parsed by the parser, but are otherwise recognized as markup.

The predefined entities such as **&lt;, &gt;,** and **&amp;** require typing and are generally difficult to read in the markup. In such cases, CDATA section can be used. By using CDATA section, you are commanding the parser that the particular section of the document contains no markup and should be treated as regular text.

## Syntax

Following is the syntax for CDATA section −

<![CDATA[ characters with markup ]]>

### Example

The following markup code shows an example of CDATA. Here, each character written inside the CDATA section is ignored by the parser.

<script> <![CDATA[ <message> Welcome to TutorialsPoint </message> ]] > </script >

Justify the text

textView.textAlignment = NSTextAlignmentJustified;

[webviewName loadHTMLString:[NSString stringWithFormat:@"<div align='justify'>%@<div>",TEXT\_set] baseURL:nil];

Core Image is a powerful API built into Cocoa Touch. It’s a critical piece of the iOS SDK. However, it often gets overlooked. In this tutorial, we’re going to examine Core Image’s face detection features and how to make use of this technology in your own iOS apps!

Face detection in iOS has been around since the days of iOS 5 (circa 2011) but it is often overlooked. The facial detection API allows developers to not only detect faces, but also check those faces for particular properties such as if a smile is present or if the person is blinking.

func detect() {

    guard let personciImage = CIImage(image: personPic.image!) else {

        return

    }

    let accuracy = [CIDetectorAccuracy: CIDetectorAccuracyHigh]

    let faceDetector = CIDetector(ofType: CIDetectorTypeFace, context: nil, options: accuracy)

    let faces = faceDetector.featuresInImage(personciImage)

    for face in faces as! [CIFaceFeature] {

        print("Found bounds are \(face.bounds)")

        let faceBox = UIView(frame: face.bounds)

        faceBox.layer.borderWidth = 3

        faceBox.layer.borderColor = UIColor.redColor().CGColor

        faceBox.backgroundColor = UIColor.clearColor()

        personPic.addSubview(faceBox)

        if face.hasLeftEyePosition {

            print("Left eye bounds are \(face.leftEyePosition)")

        }

        if face.hasRightEyePosition {

            print("Right eye bounds are \(face.rightEyePosition)")

        }

    }

}

Let’s talk about what’s going on here:

* **Line #3**: we create a variable personciImage that extracts the UIImage out of the UIImageView in the storyboard and converts it to a CIImage. CIImage is required when using Core Image
* **Line #7**: we create an accuracy variable and set to CIDetectorAccuracyHigh. You can pick from CIDetectorAccuracyHigh (which provides high processing power) and CIDetectorAccuracyLow (which uses low processing power). For the purposes of this tutorial and we choose CIDetectorAccuracyHigh because we want high accuracy.
* **Line #8**: here we define a faceDetector variable and set it to the CIDetectorclass and pass in the accuracy variable we created above.
* **Line #9**: by calling the featuresInImage method of faceDetector, the detector finds faces in the given image. At the end, it returns us an array of faces.
* **Line #11**: here we loop through the array of faces and cast each of the detected face to CIFaceFeature.
* **Line #15**: We create a UIView called faceBox and set its frame to the frame dimensions returned from faces.first. This is to draw a rectangle to highlight the detected face.
* **Line #17**: we set the faceBox’s border width to 3.
* **Line #18**: we set the border color to red.
* **Line #19**: The background color is set to clear, indicating that this view will not have a visible background.
* **Line #20**: Finally, we add the view to the personPic imageView.
* **Line #22-28**: Not only can the API help you detect the face, the detector can detect the face’s left and right eyes. We’ll not highlight the eyes in the image. Here I just want to show you the related properties of CIFaceFeature.

We will invoke the detect method in viewDidLoad. So insert the following line of code in the method:



|  |  |
| --- | --- |
| 1 | detect() |

func detect() {

    guard let personciImage = CIImage(image: personPic.image!) else {

        return

    }

    let accuracy = [CIDetectorAccuracy: CIDetectorAccuracyHigh]

    let faceDetector = CIDetector(ofType: CIDetectorTypeFace, context: nil, options: accuracy)

    let faces = faceDetector.featuresInImage(personciImage)

    // For converting the Core Image Coordinates to UIView Coordinates

    let ciImageSize = personciImage.extent.size

    var transform = CGAffineTransformMakeScale(1, -1)

    transform = CGAffineTransformTranslate(transform, 0, -ciImageSize.height)

    for face in faces as! [CIFaceFeature] {

        print("Found bounds are \(face.bounds)")

        // Apply the transform to convert the coordinates

        var faceViewBounds = CGRectApplyAffineTransform(face.bounds, transform)

        // Calculate the actual position and size of the rectangle in the image view

        let viewSize = personPic.bounds.size

        let scale = min(viewSize.width / ciImageSize.width,

                        viewSize.height / ciImageSize.height)

        let offsetX = (viewSize.width - ciImageSize.width \* scale) / 2

        let offsetY = (viewSize.height - ciImageSize.height \* scale) / 2

        faceViewBounds = CGRectApplyAffineTransform(faceViewBounds, CGAffineTransformMakeScale(scale, scale))

        faceViewBounds.origin.x += offsetX

        faceViewBounds.origin.y += offsetY

        let faceBox = UIView(frame: faceViewBounds)

        faceBox.layer.borderWidth = 3

        faceBox.layer.borderColor = UIColor.redColor().CGColor

        faceBox.backgroundColor = UIColor.clearColor()

        personPic.addSubview(faceBox)

        if face.hasLeftEyePosition {

            print("Left eye bounds are \(face.leftEyePosition)")

        }

        if face.hasRightEyePosition {

            print("Right eye bounds are \(face.rightEyePosition)")

        }

    }

}

# Face Detection with Apple’s iOS 11 Vision Framework

We also need to add the following in the Info.plist to be able to access the camera and the photo library:

* Privacy - Camera Usage Description: *Access to the camera is needed in order to be able to take a photo to be analyzed by the app*
* Privacy - Photo Library Usage Description: *Access to the photo library is needed in order to be able to choose a photo to be analyzed by the app*

<https://medium.com/@dragosholban/face-detection-with-apples-ios-11-vision-framework-a143a15e384d>

<https://www.raywenderlich.com/1163620-face-detection-tutorial-using-the-vision-framework-for-ios>

# Detect Faces with ML Kit on iOS

<https://firebase.google.com/docs/ml-kit/ios/detect-faces>

# How to Build a Face Recognition App in iOS Using CoreML and Turi Create Part 1

<https://gorillalogic.com/blog/how-to-build-a-face-recognition-app-in-ios-using-coreml-and-turi-create-part-1/>

# Vision

Apply computer vision algorithms to perform a variety of tasks on input images and video.

The Vision framework performs face and face landmark detection, text detection, barcode recognition, image registration, and general feature tracking. Vision also allows the use of custom Core ML models for tasks like classification or object detection.

# Core ML

Integrate machine learning models into your app.

Use Core ML to integrate machine learning models into your app. Core ML provides a unified representation for all models. Your app uses Core ML APIs and user data to make predictions, and to train or fine-tune models, all on the user’s device.

A *model* is the result of applying a machine learning algorithm to a set of training data. You use a model to make predictions based on new input data. Models can accomplish a wide variety of tasks that would be difficult or impractical to write in code. For example, you can train a model to categorize photos, or detect specific objects within a photo directly from its pixels.

# ML Kit Tutorial for iOS: Recognizing Text in Images

ML Kit is a mobile SDK that brings Google’s ML expertise to your app. There are two main parts of ML Kit’s APIs for common use cases and custom models that are easy to use regardless of experience.

The existing APIs currently support:

* [Recognizing text](https://firebase.google.com/docs/ml-kit/ios/recognize-text" \t "”_blank”)
* [Detecting faces](https://firebase.google.com/docs/ml-kit/ios/detect-faces" \t "”_blank”)
* [Identifying landmarks](https://firebase.google.com/docs/ml-kit/ios/recognize-landmarks" \t "”_blank”)
* [Scanning barcodes](https://firebase.google.com/docs/ml-kit/ios/read-barcodes" \t "”_blank”)
* [Labeling images](https://firebase.google.com/docs/ml-kit/ios/label-images" \t "”_blank”)

In Xcode, **File -> New -> File...** , scroll to **Resource** group, select **String File**

# [How to create multiple .strings file in Xcode other than Localizable.strings](https://stackoverflow.com/questions/43647620/how-to-create-multiple-strings-file-in-xcode-other-than-localizable-strings)

/\*\*

This is class method used to set Localizable key value

- Parameters:

- key: stirng key which is declare in localization file

- Returns: return localization string

\*/

static func localized(key:String) ->String {

let bundle = Bundle.main

return bundle.localizedString(forKey: key, value: "", table: nil)

}

pod deintegrate and pod clean are two designated commands to remove CocoaPod from your project/repo.

Here is the complete set of commands:

$ sudo gem install cocoapods-deintegrate cocoapods-clean $ pod deintegrate $ pod clean $ rm Podfile

[Google Analytics](https://www.google.com/analytics/) is a free website analytics service offered by Google that gives you insights into how users find and use your website. With Google Analytics, you can track ROI for your online marketing.

<https://fabric.io/kits?show_signup=true>

* iOS install popup did not appear
* If the iOS popup "Diawi would like to install…" did not appear, check that you don't already have the same app installed from the AppStore.  
    
  If an app with the same bundle identifier is already installed on the device from the AppStore, nothing will happen.  
    
  Delete the app installed from the AppStore to be able to install this one.
* Nothing seems to happen
* After tapping the "Install" button in the popup, if nothing seems to happen and you are on this page, check your device's home screen: the installation should be in progress: you should see the icon of the app somewhere and a progress indicator.
* Unable to download app popup
* If an "Unable to download app" popup appears after some time, first check your internet connection, and be sure that your device is not behind a firewall that may prevent downloading .ipa files.  
    
  Sadly, iOS won't give any detailed information on what went wrong. Check on the previous page if there are any warnings. Most common installation issues are:  
  - expired provisioning profile  
  - device UDID not in the provisioning profile  
  - incompatible device (check minimum iOS version, device family, required device capabilities and supported architectures)
* Untrusted Enterprise Developer
* Starting from iOS 9, the developer has to be trusted by the device, otherwise a popup will appear and prevent using the app.  
    
  On iOS 9.0/9.1, go to Settings > General > Profiles > tap on the developer's profile, and tap on Trust.  
    
  On iOS 9.2+, go to Settings > General > Device Management > tap on the developer's profile, and tap on Trust.
* Otherwise…
* Contact your app developer, so they can double-check the app and provide you with support.

#### iOS and Android apps

* Diawi works for iOS 4+ and Android 2+ apps and devices: development, ad-hoc and in-house builds can be installed wirelessly with a single tap.

## What is the maximum upload size on Diawi?

The maximum upload size depends on your plan on Diawi.

Find out the details on the [Features & Services](https://www.diawi.com/features-services) page.

### Extensible using add-ons

Starter, Premium and Enterprise plans can extend the maximum upload size using add-ons: +500 MB and +1 GB.

### Large apps upload

While uploading smaller apps using the web browser works well, larger apps may fail. For example, Safari doesn't seem to handle well uploads of files larger than 2 GB. Chrome when not the active app may slow down the upload. We recommand using the APIs and curl to ensure that your large app gets to Diawi in the best conditions.

## How long will links be available on Diawi?

Link availability is based on 3 conditions:

* the date of the last installation (or the date of the upload itself if no installation has yet been made)
* the number of installations left
* a manual disable (when using Diawi with an account)

After one of those conditions is reached, links become "Not available", as indicated in the dashboard. They can be re-enable for a short duration depending on your plan (see below) by either adding some time to the expiration or some installations.

When the recovery period expires, links become "Gone" and files are deleted from our servers. Icons may be kept if the app has been set for display on the wall.

There is no way to re-enable an link that is "Gone" as files have been deleted. However, for Starter, Premium and Enterprise accounts, dashboard and e-mail notifications can be set up to know when a link is getting close to expiration.

### Re-enable duration

### Links can be re-enabled for a short duration depending on the plan:

* Free: 1 day
* Starter: 1 day
* Premium: 3 days
* Enterprise: 5 days

Links can't be re-enabled when using Diawi without an account.

## Is there any installation limit per app on Diawi?

Diawi is mainly aimed at app developers to distribute their testing apps. Most of those apps would normally be installed a few dozen times by the testers.

We also like to support inhouse uses, so that apps can be distributed to a large number of internal users. Those apps would often be installed much more, but shouldn't most of the time exceed a few hundred times, since they must follow Apple's Enterprise rules too.

However, some apps are also trying to use Diawi for "jailbroken" or other not fair, and sometimes even illegal uses. Those generally are installed a few thousand times, and generate a very huge load on our servers. We do not support this kind of un-fair use of Diawi, since it harms other fair users.

The number of installations is therefore limited depending on the plan you use, and an option is provided for Starter, Premium and Enterprise plans to raise this limit through **add-ons**.

## How to delete an uploaded app or disable a link on Diawi?

Apps can be managed through the Dashboard: log in with your account before uploading your app, so it is identified as belonging to your account. Then, open the dashboard and click on the cog near the app's status to disable it.

When disabled, the app can't be seen or installed anymore. It is however possible to re-enable it during a few days.

If you want to delete it right away, just click on the cog again to delete it now. It will then be definitely Gone.

Apps uploaded without an account can't be disabled or deleted, they will expire automatically after the link availability expires.

## What is allowed and what is forbidden on Diawi?

Diawi's primary goal is to **help developers deliver their apps** to their testers, teams, clients, friends, ... It may also be used for production deployment of **internal apps**, like "in-house" iOS apps.

Diawi is **not an App Store**: thus, deployement of app to the general public, wether iOS (which is forbidden by Apple) or Android, is not allowed.

* On iOS, Diawi accepts: Development, Ad-Hoc and In-house signed apps.
* On Android, Diawi accepts: any signed app

Of course, any kind of **illegal or hamful apps**, or related to any of activities like these is **not allowed**: any app considered as illegal will be deleted without notice, and the uploader might be held responsible.

## How wireless app installation works on iOS

Over-the-air installation of apps on iOS is based on tools and features provided officially by Apple: itms-service and a valid manifest file providing information on the app.

## Compatibility of wireless app installation on iOS devices

Wireless installation of apps on iOS devices should work starting from **iOS 4.0+** on all iPhones, iPads and iPod touches.

Installation is only possible using **Safari**, no other standalone (Chrome, ...) or integrate browser (inside the Facebook app, ...) can initiate the installation process.

## Most common upload errors

When an app package is uploaded to Diawi, wether .ipa or .zip(.app), the content is processed to validate some key elements needed for the installation. This page provides some insight on most common errors that this processing might produce.

### Invalid package

Diawi expects two kinds of packages for iOS applications: \* .ipa file: generated by xcode using the Product > Build for running applied on a device configuration, the Product > Archive, then Window > Organizer > Archives, select the generated archive and click on Export > Save for Ad hoc/Enterprise/Development Deloyment. \* .zip(.app) file: can be generated manually after a Product > Build for running applied on a device configuration, select the Products folder in the project view, right click Show in Finder on the .app, then it in the Finder

If the package is reported as invalid by Diawi, check that your performed those steps.

### Missing embedded mobileprovision

iOS applications must contain an embedded provisioning profile: a .mobileprovision file built into the package. This is an automatic step performed by xcode since version 4.x.

Before that and iOS 4.x, mobileprovision files would have to be installed manually by the user. This is not the case anymore. It is even impossible now in iOS 8+ since Apple has completely removed the "Profiles" section in the Settings app.

When an app is installed, it checks the mobileprovision inside the package and uses it to accept or reject the app.

The mobileprovision may be missing from the package for the following reasons: \* the app has not been signed correctly => check your project settings \* the uploaded package is a Simulator version: the apps built for the iOS Simulator don't have provisioning profiles => check if you uploaded the "-iphoneos" version, not the "-iphonesimulator" one

## Can't install an app on an iOS device? Things to check

Apple doesn't provide explicit error messages when installing apps on devices. It can be quiet difficult to understand what could have gone wrong.

### Behaviors

#### iOS install popup did not appear

After tapping the "Install" link on the Diawi installation page, if the iOS popup "Diawi would like to install…" did not appear, check that you don't already have the same app installed from the AppStore. If an app with the same bundle identifier is already installed on the device from the AppStore, nothing will happen. This is a security behavior added by Apple in iOS 8+ to prevent a malicious developer from overwriting a real app with another one.

Delete the app installed from the AppStore to be able to install the testing one from Diawi.

#### Nothing seems to happen

The popup appeared, but after tapping the "Install" button of the popup, nothing seems to happen: check the device's home screen as the installation should be in progress in the background. Since iOS 8+, app installation if performed in the background on the device's home screen and Safari doesn't show anything special, however the icon can be found on the home screen and a progress indicator shows the installation progress.

#### Untrusted Enterprise Developer popup

Starting from iOS 9+, the developer of an Enterprise signed app (in-house mobileprovision) has to be trusted by the user of the device, otherwise a popup will appear and prevent from using the app.

On iOS 9.0/9.1, this can be done in: Settings > General > Profiles > tap on the developer's profile, and tap on Trust. On iOS 9.2+, this can be done in: Settings > General > Device Management > tap on the developer's profile, and tap on Trust.

#### Unable to download app popup

If an "Unable to download app" popup appears after some time, first check the internet connection, and be sure that the device is not behind a firewall that may prevent downloading .ipa files (as this happens a lot in enterprise firewall configurations).

Sadly, iOS won't give any detailed information on what went wrong here. As a user, Diawi provides information on the installation page about the most common checks. As a developer, a detailed list of checks can be found below.

Most common installation issues are: \* expired provisioning profile  
\* device UDID not in the provisioning profile  
\* incompatible device (check minimum iOS version, device family, required device capabilities and supported architectures)

### A few things to check as a developer

If you have uploaded an app to Diawi and can't install it on some of your devices, here are a few things to check:

* device's UDID must be in the provisioning profile built into the app by xcode
* device's UDID must not start with "fffffff..." (if it is, then it is fake)
* you are building a Release version of your app﻿
* the device is not behind an enterprise firewall preventing app installation

You may also try to drag&drop the app into iTunes and sync your device: it should install the app, otherwise it is not valid.

### More detailed explanations

#### UDIDs in the provisioning profile

Apple rules regarding app installation also apply when using Diawi: if you are using a **Development or Ad-hoc**provisioning profile, devices' UDIDs must be added to the provisioning profile used to sign your app. **In-house**builds with an Enterprise Apple account don't require registering UDIDs.

Since xcode 4, the .mobileprovision file is embedded into the .app/.ipa file when compiling. When in doubt about which version has been used by xcode, you may simply unzip the .ipa file, then right-click Show package content on the .app located inside the Payload folder and look for the embedded.mobileprovision. The file can be opened with a text editor (TextMate, Sublime Text, Atom, ...): contains some XML/plist contents with the list of UDIDs.

XML/plist data inside the .mobileprovision looks like this:

﻿<key>ProvisionedDevices</key> <array> <string>37149a037f1..................................3c0078</string>﻿ .... </array>

Also, Diawi displays this list on the installation page and checks if the user's device UDID is inside that list.

#### Valid UDIDs

UDIDs are string of 40 hexadecimal characters ([0-9a-f]{40}).

Device's UDID must be retrieved using iTunes or a webapp like m.diawi.com: those UDIDs are known to be valid.

For some time, it was possible to get the device's UDID using a native API provided by Apple and integrated in some apps available on the AppStore. However, this API has been deprecated in iOS 7 and instead provides a **"fake" UDID starting with ffffff**.... Those UDIDs are **not valid** and can't be used to identify a device and install apps on it.

#### Built architectures

When building iOS apps, you may be targeting one or more of the available architecture (armv6, armv7, arm64). With the Debug configuration in xcode, the default setting is to build only the "current" architecture (for faster compilation), which may prevent the app from running on other/older devices.

Current architectures and their compatibility:

* armv6: iPhone 2G+, iPod touch 1st gen.+
* armv7: iPhone 3GS+, all iPads, iPad mini 1, iPod touch 3rd gen.+
* armv7s: iPhone 5+, iPad 4+, iPad mini 1, iPod touch 3rd gen.+
* arm64: iPhone 5s+, iPad Air+, iPad mini 2+, iPod touch 6th gen.+

(Source: [List of iOS devices on Wikipedia](http://en.wikipedia.org/wiki/List_of_iOS_devices##Features))

Newer devices are so far compatible with older architectures: an iPad Air, arm64, can run an armv7 built app, but not the opposite.

With the Release configuration, all the active/current architectures will be built and the app will work on all the targeted devices. As of Xcode 6/7, the default is armv7+arm64 (armv7s is not really needed: it's exclusion could be that it doesn't provide enough enhancement compared to armv7 while it might be adding extra file size).

### Enterprise developers and in-house applications

Since iOS 9.0, Apple has changed the way in-house apps are authorized on the devices.

Before, an app distributed with an in-house signature and mobileprovision could be installed and launched directly. At first launch of the app, an "Untrusted Enterprise Developer" will show. The app developer has to be "trusted" through the Settings of the device.

See paragraph "Untrusted Enterprise Developer popup" on [Can't install an app on a device? Things to check](https://www.diawi.com/knowledge-base/iOS/Cant-install-an-app-on-a-device-things-to-check) for the detailed steps.

### Development apps can't overwrite AppStore apps

Since iOS 9.x, Apple has changed the way development/in-house vs AppStore apps are installed.

Before, a development app with the same bundle identifier could overwrite an existing AppStore app. This could be dangerous since a legitimate app from the AppStore could then be replaced by an unknown app. Now, when trying to install a development app with the same bundle identifier as an app from the AppStore already on the device, nothing will happen.

## iOS 8 changes and known issues

### Valid bundle identifier in the manifest

Since iOS 8.x, Apple has changed the way the manifest is read by the device.

Before, the device didn't check the bundle identifier in the manifest: the app installed could have a different identifier. Now, the bundle identifiers must match.

Diawi provides manifests with valid bundle identifiers.

### Mandatory HTTPS on the manifest URL

Since iOS 8.x, Apple has changed the way the manifest is downloaded by the device.

Before, the device could load a manifest from an http or https URL. Now, the manifest will be accepted by the device only from an https URL.

Diawi provides https for manifest URLs.

### No more provisioning profile management

Since iOS 8.0, provisioning profiles can't be installed manually anymore, and they do not appear anymore in the Settings app. They are embedded inside the .app/.ipa file by xcode, and are installed automatically with the app.

### Background app installation

Since iOS 8.0, app installation happens in the background: after tapping the install link, it may seem that nothing happens, but the app is actually being installed on the Springboard. Check the Springboard to see the app icon.

### iOS 7 to iOS 8 update

If an app has been installed on iOS 7 and then the device updated to iOS 8, it may be possible that the development app couldn't be installed anymore: there seems to be some kind of cache on the bundle identifier preventing wireless installation. A trick was used during the first months after iOS 8 release by providing another temporarily fake bundle identifier in the manifest.

## Getting more information with xcode's console on iOS

Apple doesn't provide much information to the end user when the installation goes wrong. Often, error messages are generic ones like "The application could not be downloaded".

One way to get a little bit more information is to connect the device to a Mac with xcode, and look at the console.

### Accessing the console

The device console is located in xcode's Window > Devices menu

In the Device window, choose your device on the left and the console will appear in the bottom right.

Search for lines containing itunesstored: those error messages often help debugging problems with installations.

### Error messages

* Cannot load non-https manifest URL

The error in the screenshot above means that a manifest plist was requested on an HTTP url, but Apple requires that manifests must be served through HTTPS since iOS 8+.

This error should never happen on Diawi, but may happen on custom setups.

# The Future of Fabric

We are integrating the best of Fabric into Firebase to bring you one powerful app development platform. **The Fabric dashboard will be available until March 31, 2020 for existing users**. We recommend that new users start in Firebase to take advantage of the latest products and features we're building there.

Crashlytics has been integrated into Firebase, with new Firebase-only features. New apps should use Crashlytics in Firebase to get access to the latest updates and features going forward.

Crashlytics Beta is being built as App Distribution in Firebase. Continue using Fabric Crashlytics for your Beta test releases while App Distribution is in development

**pod 'Fabric'**

**pod 'Crashlytics'**

## Add a Run Script Build Phase

Xcode 10 only: Add your app's built Info.plist location to the Build Phase's Input Files field:

## Add Your API Key

**Fabric.with([Crashlytics.self])**

// [[Crashlytics sharedInstance] crash];

// fatalError();

// [self performSelector:@selector(die\_die)];

// @throw NSInternalInconsistencyException;

Add Firebase to your iOS project

## ****Step 1****: Create a Firebase project

## ****Step 2****: Register your app with Firebase

* Enter your app's [bundle ID](https://cocoacasts.com/what-are-app-ids-and-bundle-identifiers/) in the **iOS bundle ID** field.
* *(Optional)* Enter other app information as prompted by the setup workflow.
* Click **Register app**.

## ****Step 3****: Add a Firebase configuration file

1. Click **Download GoogleService-Info.plist** to obtain your Firebase iOS config file (GoogleService-Info.plist).
2. Move your config file into the root of your Xcode project. If prompted, select to add the config file to all targets.

## ****Step 4****: Add Firebase SDKs to your app

To your Podfile, add the pods that you want to use in your app. For example, for Analytics:

pod 'Firebase/Analytics'

## ****Step 5****: Initialize Firebase in your app

// Use Firebase library to configure APIs  
FirebaseApp.configure()

## Available pods

The following pods are available for the various Firebase products.

| Product | Pod |
| --- | --- |
| [AdMob](https://firebase.google.com/docs/admob/ios/quick-start) | pod 'Firebase/AdMob' |
| [Analytics](https://firebase.google.com/docs/analytics/ios/start) | pod 'Firebase/Analytics' |
| [Authentication](https://firebase.google.com/docs/auth/ios/start) | pod 'Firebase/Auth' |
| [Cloud Firestore](https://firebase.google.com/docs/firestore/quickstart) | pod 'Firebase/Firestore' |
| [Cloud Functions for Firebase Client SDK](https://firebase.google.com/docs/functions/) | pod 'Firebase/Functions' |
| [Cloud Messaging](https://firebase.google.com/docs/cloud-messaging/ios/client) | pod 'Firebase/Messaging' |
| [Cloud Storage](https://firebase.google.com/docs/storage/ios/start) | pod 'Firebase/Storage' |
| [Crashlytics](https://firebase.google.com/docs/crashlytics/get-started?platform=ios) | pod 'Fabric' pod 'Crashlytics' |
| [Dynamic Links](https://firebase.google.com/docs/dynamic-links/ios/create) | pod 'Firebase/DynamicLinks' |
| [In-App Messaging](https://firebase.google.com/docs/in-app-messaging/get-started?platform=ios) | pod 'Firebase/InAppMessaging' |
| [In-App Messaging Display](https://firebase.google.com/docs/in-app-messaging/get-started?platform=ios) | pod 'Firebase/InAppMessagingDisplay' |
| [ML Kit: Vision APIs](https://firebase.google.com/docs/ml-kit/) | pod 'Firebase/MLVision' |
| [ML Kit: Image Labeling Model](https://firebase.google.com/docs/ml-kit/) | pod 'Firebase/MLVisionLabelModel' |
| [ML Kit: Barcode Scanning Model](https://firebase.google.com/docs/ml-kit/) | pod 'Firebase/MLVisionBarcodeModel' |
| [ML Kit: Text Recognition Model](https://firebase.google.com/docs/ml-kit/) | pod 'Firebase/MLVisionTextModel' |
| [ML Kit: Face Detection Model](https://firebase.google.com/docs/ml-kit/) | pod 'Firebase/MLVisionFaceModel' |
| [ML Kit: Object Detection and Tracking Model](https://firebase.google.com/docs/ml-kit/) | pod 'Firebase/MLVisionObjectDetection' |
| [ML Kit: Natural Language APIs](https://firebase.google.com/docs/ml-kit/) | pod 'Firebase/MLNaturalLanguage' |
| [ML Kit: Language Identification Model](https://firebase.google.com/docs/ml-kit/) | pod 'Firebase/MLNLLanguageID' |
| [ML Kit: Translate Model](https://firebase.google.com/docs/ml-kit/) | pod 'Firebase/MLNLTranslate' |
| [ML Kit: Smart Reply Model](https://firebase.google.com/docs/ml-kit/) | pod 'Firebase/MLNLSmartReply' |
| [ML Kit: Custom Model APIs](https://firebase.google.com/docs/ml-kit/) | pod 'Firebase/MLModelInterpreter' |
| [ML Kit: AutoML Vision Edge API](https://firebase.google.com/docs/ml-kit/) | pod 'Firebase/MLVisionAutoML' |
| [Performance Monitoring](https://firebase.google.com/docs/perf-mon/get-started-ios) | pod 'Firebase/Performance' |
| [Realtime Database](https://firebase.google.com/docs/database/ios/start) | pod 'Firebase/Database' |
| [Remote Config](https://firebase.google.com/docs/remote-config/use-config-ios) | pod 'Firebase/RemoteConfig' |

An iOS app needs the backend services for performing the various task like storing data in the database, hosting of files, caching data, monitoring etc. [Firebase](https://firebase.google.com/?source=post_page---------------------------) is a cloud-based service that provides most of the server side things for our iOS apps so that we don’t have to manage the infrastructure like databases, servers and server-side things by yourself. The Firebase will take care of everything. Firebase provides many useful services that make our life easier but most important services are

* Remote Database
* Remote Config
* Analytics
* Authentication
* Crash Reporting

Importing and initializing the Google Mobile Ads SDK is the first step toward displaying AdMob ads and earning revenue. Once that's done, you can choose an ad format (such as native or rewarded video) and get a detailed set of steps for implementing it.

Google Analytics for Firebase collects usage and behavior data for your app. The SDK logs two primary types of information:

* **Events:** What is happening in your app, such as user actions, system events, or errors.
* **User properties:** Attributes you define to describe segments of your user base, such as language preference or geographic location.

You can use Firebase Authentication to allow users to sign in to your app using one or more sign-in methods, including email address and password sign-in, and federated identity providers such as Google Sign-in and Facebook Login. This tutorial gets you started with Firebase Authentication by showing you how to add email address and password sign-in to your app.

Cloud Firestore stores data in Documents, which are stored in Collections. Cloud Firestore creates collections and documents implicitly the first time you add data to the document. You do not need to explicitly create collections or documents.

Cloud Functions for Firebase let you automatically run backend code in response to events triggered by Firebase features and HTTPS requests. Your code is stored in Google's cloud and runs in a managed environment. There's no need to manage and scale your own servers.

For iOS client apps, you can implement Firebase Cloud Messaging in two complementary ways:

* Receive basic push messages up to 4KB over the Firebase Cloud Messaging APNs interface.
* Send messages upstream and/or receive downstream data payloads up to 4KB in foregrounded apps.

Cloud Storage for Firebase lets you upload and share user generated content, such as images and video, which allows you to build rich media content into your apps. Your data is stored in a [Google Cloud Storage](https://cloud.google.com/storage) bucket, an exabyte scale object storage solution with high availability and global redundancy. Cloud Storage lets you securely upload these files directly from mobile devices and web browsers, handling spotty networks with ease.

ML Kit is a mobile SDK that brings Google's machine learning expertise to Android and iOS apps in a powerful yet easy-to-use package. Whether you're new or experienced in machine learning,

The Firebase Realtime Database is a cloud-hosted database. Data is stored as JSON and synchronized in realtime to every connected client. When you build cross-platform apps with our Android, iOS, and JavaScript SDKs, all of your clients share one Realtime Database instance and automatically receive updates with the newest data.

An exception is a special condition that interrupts the normal flow of program execution. Each application can interrupt the program for different reasons. For example, one application might interpret saving a file in a directory that is write-protected as an exception. In this sense, the exception is equivalent to an error. Another application might interpret the user’s key-press (for example, Control-C) as an exception: an indication that a long-running process should abort.

## [SIGSEGV](http://en.wikipedia.org/wiki/SIGSEGV) (Segmentation fault)

Access to an invalid memory address. The address exist, but your program does not have access to it.

## [SIGBUS](http://en.wikipedia.org/wiki/SIGBUS) (Bus error)

Access to an invalid memory address. The address does not exist, or the alignment is invalid.

## [SIGFPE](http://en.wikipedia.org/wiki/SIGFPE) (Floating point exception)

Invalid arithmetic operation. Can be related to integer operations, despite the name.

## [SIGPIPE](http://en.wikipedia.org/wiki/SIGPIPE)

Broken pipe.

## [SIGILL](http://en.wikipedia.org/wiki/SIGILL)

Illegal processor instruction.

## [SIGTRAP](http://en.wikipedia.org/wiki/SIGTRAP)

Debugger related

## [SIGABRT](http://en.wikipedia.org/wiki/SIGABRT)

Program crash, not related to one of the preceding signal.

In [computer science](https://en.wikipedia.org/wiki/Computer_science), **compile time** refers to either the operations performed by a [compiler](https://en.wikipedia.org/wiki/Compiler) (the "compile-time operations"), [programming language](https://en.wikipedia.org/wiki/Programming_language) requirements that must be met by [source code](https://en.wikipedia.org/wiki/Source_code) for it to be successfully compiled (the "compile-time requirements"), or properties of the program that can be reasoned about during compilation. Compile time refers to the time duration during which the statements written in any programming language are checked for errors.

The operations performed at compile time usually include [syntax analysis](https://en.wikipedia.org/wiki/Syntax_analysis), various kinds of [semantic analysis](https://en.wikipedia.org/wiki/Semantic_analysis_(computer_science)) (e.g., [type checks](https://en.wikipedia.org/wiki/Datatype) and [instantiation of template](https://en.wikipedia.org/wiki/Instantiation_of_template)) and [code generation](https://en.wikipedia.org/wiki/Code_generation_(compiler)).

Compile time occurs before [link time](https://en.wikipedia.org/wiki/Link_time) (when the output of one or more compiled files are joined together) and [runtime](https://en.wikipedia.org/wiki/Run_time_(program_lifecycle_phase)) (when a [program](https://en.wikipedia.org/wiki/Computer_program)is [executed](https://en.wikipedia.org/wiki/Execution_(computing))). In some programming languages it may be necessary for some compilation and linking to occur at runtime. There is a trade-off between compile-time and link-time in that many compile time operations can be deferred to link-time without incurring extra run-time.

"Compile time" can also refer to the amount of time required for compilation.

**Parsing**, **syntax analysis**, or **syntactic analysis** is the process of analysing a [string](https://en.wikipedia.org/wiki/String_(computer_science)) of [symbols](https://en.wikipedia.org/wiki/Symbol_(formal)), either in [natural language](https://en.wikipedia.org/wiki/Natural_language), [computer languages](https://en.wikipedia.org/wiki/Computer_languages) or [data structures](https://en.wikipedia.org/wiki/Data_structure), conforming to the rules of a [formal grammar](https://en.wikipedia.org/wiki/Formal_grammar).

Within [computational linguistics](https://en.wikipedia.org/wiki/Computational_linguistics) the term is used to refer to the formal analysis by a computer of a sentence or other string of words into its constituents, resulting in a [parse tree](https://en.wikipedia.org/wiki/Parse_tree)showing their syntactic relation to each other, which may also contain [semantic](https://en.wikipedia.org/wiki/Semantics) and other information.

A **compiler** is a [computer program](https://en.wikipedia.org/wiki/Computer_program) that [translates](https://en.wikipedia.org/wiki/Translator_(computing)) computer code written in one [programming language](https://en.wikipedia.org/wiki/Programming_language) (the source language) into another programming language (the target language). The name *compiler* is primarily used for programs that translate [source code](https://en.wikipedia.org/wiki/Source_code) from a [high-level programming language](https://en.wikipedia.org/wiki/High-level_programming_language) to a [lower level language](https://en.wikipedia.org/wiki/Lower_level_language) (e.g., [assembly language](https://en.wikipedia.org/wiki/Assembly_language), [object code](https://en.wikipedia.org/wiki/Object_code), or [machine code](https://en.wikipedia.org/wiki/Machine_code)) to create an [executable](https://en.wikipedia.org/wiki/Executable) program.[[1]](https://en.wikipedia.org/wiki/Compiler#cite_note-1)[[2]](https://en.wikipedia.org/wiki/Compiler#cite_note-dragon-2):p1

However, there are many different types of compilers. If the compiled program can run on a computer whose [CPU](https://en.wikipedia.org/wiki/Central_processing_unit) or [operating system](https://en.wikipedia.org/wiki/Operating_system)is different from the one on which the compiler runs, the compiler is a [cross-compiler](https://en.wikipedia.org/wiki/Cross-compiler). A [bootstrap compiler](https://en.wikipedia.org/wiki/Bootstrap_compiler) is written in the language that it intends to compile. A program that translates from a [low-level language](https://en.wikipedia.org/wiki/Low-level_language) to a higher level one is a [decompiler](https://en.wikipedia.org/wiki/Decompiler). A program that translates between high-level languages is usually called a [source-to-source compiler](https://en.wikipedia.org/wiki/Source-to-source_compiler) or transpiler. A language [rewriter](https://en.wikipedia.org/wiki/Rewriting) is usually a program that translates the form of expressions without a change of language. The term [compiler-compiler](https://en.wikipedia.org/wiki/Compiler-compiler) refers to tools used to create parsers that perform syntax analysis.

A compiler is likely to perform many or all of the following operations: [preprocessing](https://en.wikipedia.org/wiki/Preprocessor), [lexical analysis](https://en.wikipedia.org/wiki/Lexical_analysis), [parsing](https://en.wikipedia.org/wiki/Parsing), [semantic analysis](https://en.wikipedia.org/wiki/Semantic_analysis_(compilers))([syntax-directed translation](https://en.wikipedia.org/wiki/Syntax-directed_translation)), conversion of input programs to an [intermediate representation](https://en.wikipedia.org/wiki/Intermediate_representation), [code optimization](https://en.wikipedia.org/wiki/Code_optimization) and [code generation](https://en.wikipedia.org/wiki/Code_generation_(compiler)).

In [computer science](https://en.wikipedia.org/wiki/Computer_science), **link time** refers to the period of time, during the creation of a computer program, in which a [linker](https://en.wikipedia.org/wiki/Linker_(computing)) is being applied to that program.[[1]](https://en.wikipedia.org/wiki/Link_time#cite_note-sams-1)[[2]](https://en.wikipedia.org/wiki/Link_time#cite_note-2)[[3]](https://en.wikipedia.org/wiki/Link_time#cite_note-stroustrup-3) Link time occurs after [compile time](https://en.wikipedia.org/wiki/Compile_time) and before [runtime](https://en.wikipedia.org/wiki/Run_time_(program_lifecycle_phase)) (when a [program](https://en.wikipedia.org/wiki/Computer_program) is [executed](https://en.wikipedia.org/wiki/Execution_(computing))).

The operations performed at link time usually include fixing up the addresses of externally referenced objects and functions, various kinds of cross module checks (e.g. [type checks](https://en.wikipedia.org/wiki/Data_type)on externally visible identifiers and in some languages [instantiation of templates](https://en.wikipedia.org/wiki/Generic_programming)). Some [optimizing compilers](https://en.wikipedia.org/wiki/Compiler_optimization) delay code generation until link time because it is here that information about a complete program is available to them. Resolving external variables in a program is also done at link time.

In [computer science](https://en.wikipedia.org/wiki/Computer_science), **run time**, **runtime** or **execution time** is the time during which a program is running ([executing](https://en.wikipedia.org/wiki/Execution_(computing))), in contrast to other [program lifecycle phases](https://en.wikipedia.org/wiki/Program_lifecycle_phase) such as [compile time](https://en.wikipedia.org/wiki/Compile_time), [link time](https://en.wikipedia.org/wiki/Link_time) and [load time](https://en.wikipedia.org/wiki/Load_time).

A run-time error is detected after or during the execution (running state) of a program, whereas a compile-time error is detected by the [compiler](https://en.wikipedia.org/wiki/Compiler) before the program is ever executed. [Type checking](https://en.wikipedia.org/wiki/Type_checking), [register allocation](https://en.wikipedia.org/wiki/Register_allocation), [code generation](https://en.wikipedia.org/wiki/Code_generation_(compiler)), and code optimization are typically done at compile time, but may be done at run time depending on the particular language and compiler. Many other runtime errors exist and are handled differently by different languages, such as [Division by zero](https://en.wikipedia.org/wiki/Division_by_zero) errors, domain errors, [array subscript out of bounds](https://en.wikipedia.org/wiki/Bounds_checking) errors, [Arithmetic underflow](https://en.wikipedia.org/wiki/Arithmetic_underflow) errors, several types of underflow and [overflow](https://en.wikipedia.org/wiki/Overflow_(disambiguation)) errors, and many other runtime errors generally considered as software bugs which may or may not be caught and handled by any particular computer language.

In [computer systems](https://en.wikipedia.org/wiki/Computing) a **loader** is the part of an [operating system](https://en.wikipedia.org/wiki/Operating_system) that is responsible for loading [programs](https://en.wikipedia.org/wiki/Computer_program) and [libraries](https://en.wikipedia.org/wiki/Library_(computing)). It is one of the essential stages in the process of starting a program, as it places programs into memory and prepares them for execution. Loading a program involves reading the contents of the [executable file](https://en.wikipedia.org/wiki/Executable) containing the program instructions into memory, and then carrying out other required preparatory tasks to prepare the executable for running. Once loading is complete, the operating system starts the program by passing control to the loaded program code.

[Runtime](https://pc.net/glossary/definition/runtime) and [compile](https://pc.net/glossary/definition/compile) time are programming terms that refer to different stages of software [program](https://pc.net/glossary/definition/program) development. In order to create a program, a developer first writes [source code](https://pc.net/glossary/definition/sourcecode), which defines how the program will function. Small programs may only contain a few hundred lines of source code, while large programs may contain hundreds of thousands of lines of source code. The source code must be compiled into machine code in order to become and executable program. This compilation process is referred to as compile time.

A compiled program can be opened and run by a user. When an [application](https://pc.net/glossary/definition/application) is running, it is called runtime.

The terms "runtime" and "compile time" are often used by programmers to refer to different types of errors. A compile time error is a problem such as a [syntax](https://pc.net/glossary/definition/syntax) error or missing file reference that prevents the program from successfully compiling. The [compiler](https://pc.net/glossary/definition/compiler) produces compile time errors and usually indicates what line of the source code is causing the problem.

If a program's source code has already been compiled into an executable program, it may still have [bugs](https://pc.net/glossary/definition/bug) that occur while the program is running. Examples include features that don't work, unexpected program behavior, or program crashes. These types of problems are called runtime errors since they occur at runtime.

**Compile time** is when your code is being processed by a [compiler](https://en.wikipedia.org/wiki/Compiler). In this context, it's talking about a compiler that is transforming your code into an executable binary.

[**Load time**](https://en.wikipedia.org/wiki/Loader_(computing)) is when the Operating System is reading an executable from long term storage (typically a hard drive) and ***loading*** it into short term memory (RAM) from which it can be executed. Generally the hard drive is too slow to feed the CPU, so fast memory is used to store instructions/programs that the CPU is getting ready to execute. This is also when the initial memory allocation is reserved and initialized for use by the program.

[**Execution**](https://en.wikipedia.org/wiki/Execution_(computing))**time** is when a program is executing or running. The instructions are in memory and are being processed by the CPU. Additional memory may be allocated and/or deallocated at this time.

let money = (amount: 100, currency: "USD")

let currency = money.currency // "USD"

let currency = money.1 // "USD"

func format(input: (Int,String)) -> String { return "I have \(input.0) \(input.1) in my wallet" } println(format(money)) // This prints "I have 100 USD in my wallet"

 A struct can hold exactly the same kind of a data as a tuple. In addition, you can also define functions on the struct itself (whereas with tuples, you can only define top-level functions).

struct Mass { let quantity: Int let unit: String func format() -> String { return "I have \(self.quantity) \(self.unit) in my backpack." } }

**Similarities Between Tuples and Structs**

* Both may have any number of members of any type, including closures
* Both can be constructed inline (see typealias in the code below)
* Both prevent mutation of any members if declared as constants
* If a tuple has labeled members, both structs and tuples allow member access by label

**Differences Between Tuples and Structs**

* Structs require a definition before use
* Structs do not allow pattern matching against their members
* Structs allow mutability of members declared as variables if the instance is a variable
* Tuples do not allow mutating functions or functions that refer to any of its members
* Tuples may not implement Protocols
* If a tuple has anonymous members, its members can be accessed by index, unlike structs

**Some code for a playground illustrating these differences and similarities**

// All commented code causes a compilation error. Uncomment to view error messages. struct StructureX { let a: Int = 0 var b: String = "string" } // // Struct member variability // var structureA: StructureX = StructureX() let structureB: StructureX = StructureX() //structureA.a = 2 // declared as a constant, instance is variable structureA.b = "allowed" // declared as a variable, instance is variable //structureB.a = 2 // declared as constant, instance is constant //structureB.b = "not allowed" // declared as constant, instance is constant structureA = structureB // these are the same type structureA // // A tuple can't be used as a literal to construct a struct. // //let StructureC: StructureX = (a: 17, b: "nope") // // Typealias a labeled tuple and it can be constructed similarly to a struct // typealias StructureT = (a: Int, b: String) var structureD: StructureT = StructureT(a: 0, b: "asdf") structureD //structureD = structureA // but they are distinct types

<https://stackoverflow.com/questions/27384151/swift-tuples-different-from-struct-and-from-each-other>

A struct is much more powerful than a tuple, here's some advantages only a struct offers

1. a struct can conform to a **protocol**, a tuple can't
2. a struct can have **methods**, a tuple can't
3. a struct can have **computed properties**, a tuple can't
4. a struct can have **initializers**, a tuple can't

A tuple can contain elements of different types. So for example you could declare a tuple containing a String and an Int, while all elements of an array have to be the same type, unless you use AnyObject and type casting.

*A*type alias declaration *introduces a named alias of an existing type into your program. Type alias declarations are declared using the keyword typealias and have the following form:*

*typealias name = existing type*

*After a type alias is declared, the aliased*name *can be used instead of the*existing type *everywhere in your program. The*existing type *can be a named type or a compound type. Type aliases do not create new types; they simply allow a name to refer to an existing type.*

Writing async code in Swift is (mostly) a joyful experience. Things can get hairy though when writing something like an API Client with functions that accept multiple closure arguments, which themselves accept multiple arguments.

Example:

class ApiClient { // .... func getUsers(success: ((result: AnyObject, operation: AFHTTPRequestOperation) -> Void)? = nil,  
 error: ((error: NSError, operation: AFHTTPRequestOperation) -> Bool)? = nil,  
 finished: (() -> Void)? = nil) {  
 // Do stuff   
 }func getUser(user: User, success: ((result: AnyObject, operation: AFHTTPRequestOperation) -> Void)? = nil,  
 error: ((error: NSError, operation: AFHTTPRequestOperation) -> Bool)? = nil,  
 finished: (() -> Void)? = nil) {  
 // Do stuff   
 }func getInvitations(success: ((result: AnyObject, operation: AFHTTPRequestOperation) -> Void)? = nil,  
 error: ((error: NSError, operation: AFHTTPRequestOperation) -> Bool)? = nil,  
 finished: (() -> Void)? = nil) {  
 // Do stuff   
 } // ....}

Functions and closures in Swift have a type (consisting of the function’s parameter types and return type) and so can be aliased like anything else:

typealias MyFunctionDefinition = (Integer, String) -> Void

Applying that to our example API client, we end up with this:

class ApiClient { // .... typealias SuccessHandler = (result: AnyObject, operation: AFHTTPRequestOperation)  
 -> Void  
 typealias ErrorHandler = (error: NSError, operation: AFHTTPRequestOperation)  
 -> Void  
 typealias FinishedHandler = () -> Void func getUsers(success: (SuccessHandler)? = nil,  
 error: (ErrorHandler)? = nil,  
 finished: (FinishedHandler)? = nil) {  
 // Do stuff  
 } func getUsers(success: (SuccessHandler)? = nil,  
 error: (ErrorHandler)? = nil,  
 finished: (FinishedHandler)? = nil) {  
 // Do stuff  
 } func getInvitations(user: User,  
 success: (SuccessHandler)? = nil,  
 error: (ErrorHandler)? = nil,  
 finished: (FinishedHandler)? = nil) {  
 // Do stuff  
 } // ....}

To me those function definitions are now *a lot* easier to read, and undeniably easier to write. Best of all, this is a drop-in solution. Nothing else about your functions need change and everything can be called as before.

ApiClient.sharedInstance.getUsers(success: { result, operation in  
 // Do stuff  
}, error { error, operation in  
 // Do stuff  
}, finished {  
 // Do stuff  
})

A typealias in Swift is literally an alias for an existing type. Simple, isn’t it? They can be useful in making your code a bit more readable. By using them in a smart way they can be really useful in your codebase.

### Int

In most cases, you don’t need to pick a specific size of integer to use in your code. Swift provides an additional integer type, Int, which has the same size as the current platform’s native word size:

**On a 32-bit platform, Int is the same size as Int32.**

**On a 64-bit platform, Int is the same size as Int64.**

Unless you need to work with a specific size of integer, always use Int for integer values in your code. This aids code consistency and interoperability. Even on 32-bit platforms, Int can store any value between -2,147,483,648 and 2,147,483,647, and is large enough for many integer ranges.

* Int32: 4 bytes: from −2147483648 to +2147483647
* Int64: 8 bytes: from −9223372036854775808 to +9223372036854775807

You usually want to use NSInteger when you don't know what kind of processor architecture your code might run on, so you may for some reason want the largest possible int type, which on 32 bit systems is just an int, while on a 64-bit system it's a long.

I'd stick with using NSInteger instead of int/long unless you specifically require them.

NSInteger/NSUInteger are defined as \*dynamic typedef\*s to one of these types, and they are defined like this:

#if \_\_LP64\_\_ || TARGET\_OS\_EMBEDDED || TARGET\_OS\_IPHONE || TARGET\_OS\_WIN32 || NS\_BUILD\_32\_LIKE\_64 typedef long NSInteger; typedef unsigned long NSUInteger; #else typedef int NSInteger; typedef unsigned int NSUInteger;

ve reviewed some of the most prominent offerings:

* Crashlytics
* Crittercism
* Bugsense
* TestFlight
* HockeyApp

You can integrate Crashlytics with third party bug trackers and project management tools, including the following:

* [Campfire](http://campfirenow.com/)
* [JIRA](http://www.atlassian.com/software/jira/overview)
* [Pivotal Tracker](http://www.pivotaltracker.com/)
* [Redmine](http://www.redmine.org/)
* [PagerDuty](http://www.pagerduty.com/)
* NULL has no equivalent in Swift.
* nil is also called nil in Swift
* Nil has no equivalent in Swift
* [NSNull null] can be accessed in Swift as NSNull()

# Difference between nil vs NULL vs NSNull

There are three ways to represent a null value in Objective-C.

nil = Absence of value with Objective-C object variables

NULL = Absence of a value with C-style pointers  
NSNull = A nil boxed as an object for storage in a collection

If you try adding nil to a NSDictionary or NSArray, you will find out it doesn’t perform as expected.  If you absolutely need to store a null value in a collection, you can use NSNull to represent the lack of a value.  For example:

NSNull nullValue = [NSNull null]; [anArray addObject:nullValue];

Note, however, that since NSNull represents the lack of a value, there is no way to retrieve that value from the collection. In order to determine if there is no value for a particular key or index, you must compare against NSNull.

id value = [anArray objectAtIndex:3]; if (value == [NSNull null]) { // Do something for a null }

# Difference between Copy and MutableCopy

The difference between ‘copy’ and ‘mutableCopy’ can be simply understood with polymorphism in Object Oriented Programming concepts.

We will take the example of Array in objective-C. MutableArray is the extension of NSArray class. Therefore, all the methods available in NSArray is available in NSMutableArray, but the additional methods present in NSMutableArray is not known to NSArray class.

Now moving ahead, the copy method on an NSArray will return an object of type NSArray(The array that can not be modified). And mutableCopy method will return an object of mutable type (The array that can be modified).

Now, there can below cases :

Case 1:

    NSArray \*arr1=[NSArray arrayWithObjects:@”A”,@”B”,@”C”, nil];

    NSArray \*arr2=[arr1 copy];

    NSLog(@”arr1:%@”,[arr1 description]);

    NSLog(@”arr2:%@”,[arr2 description]);

In this case an NSArray object is returned and is received in an NSArray object.

So, the array received can not be modified.

Therefore, we will not be able to use below statement

[arr2 insertObject:@”Z” atIndex:0];

Case 2:

    NSArray \*arr1=[NSArray arrayWithObjects:@”A”,@”B”,@”C”, nil];

    NSArray \*arr2=[arr1 mutableCopy];

    NSLog(@”arr1:%@”,[arr1 description]);

    NSLog(@”arr2:%@”,[arr2 description]);

In this case an NSMutableArray object is returned and is received in an NSArray object.

Since the receiver object is of type NSArray, it doesn’t know the methods present in NSMutableArray, arr2 will not be able to use any of the methods of NSMutableArray.

That is, the method mutableArray will make no sense in this scenario.

So, we will not be able to use below statement

[arr2 insertObject:@”Z” atIndex:0];

Case 3:

    NSArray \*arr1=[NSArray arrayWithObjects:@”A”,@”B”,@”C”, nil];

    NSMutableArray \*arr2=[arr1 copy];

    NSLog(@”arr1:%@”,[arr1 description]);

    NSLog(@”arr2:%@”,[arr2 description]);

In this case an NSArray object is returned and is received in an NSMutableArray type object. The receiver arr2 is now pointing to an object address that is of type NSArray. However arr2 has the additional methods than NSArray, it will not be able to use those methods coz the pointed object NSArray does not know the additional methods present in arr2(NSMutableArray).

Hence, we will not be able to use below statement

[arr2 insertObject:@”Z” atIndex:0];

Case 4:

    NSArray \*arr1=[NSArray arrayWithObjects:@”A”,@”B”,@”C”, nil];

    NSMutableArray \*arr2=[arr1 mutableCopy];

    NSLog(@”arr1:%@”,[arr1 description]);

    NSLog(@”arr2:%@”,[arr2 description]);

    [arr2 insertObject:@”Z” atIndex:0];

In this scenario, the receiver(arr2) of type NSMutableArray receives an object of type NSMutableArray. Therefore, the receiver knows the additional methods of NSMutableArray as well as the object that is being pointed by receiver(arr2).

And finally below statement will work like charm,

[arr2 insertObject:@”Z” atIndex:0];

Log before insertion will be:

**arr1:(**

**A,**

**B,**

**C**

**)**

**arr2:(**

**A,**

**B,**

**C**

**)**

And log after insertion will be:

**arr1:(**

**A,**

**B,**

**C**

**)**

**arr2:(**

**Z,**

**A,**

**B,**

**C**

**)**

Hope above description helped you.

# What is Dangling pointer ?

Since Objective-C is based upon C foundation, it is possible to use C-style functions. However Objective-C’s full power is only unlocked if you make full use of its object-oriented extensions.

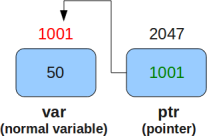
Before  talking about Dangling pointers first we  have to know about pointers.

**Pointers** :

A variable within an application can be considered to consist of four components. These are its:

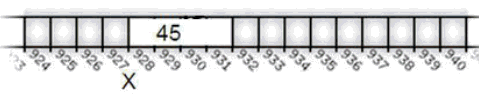
* Name
* Location — where it is stored in memory
* Type — what kind of data it can store
* Current value

Pointers are different from other normal variables which can store values, pointers are special variables that can hold the address of a variable. Since they store memory address of a variable, the pointers are very commonly said to “point to variables”. Lets try to understand the concept.

[](https://iphonecodecenter.files.wordpress.com/2015/10/jhsdgfys.png)

A normal variable ‘var’ has a memory address of 1001 and holds a value 50. A**pointer variable** has its own address 2047 but stores 1001, which is the address of the variable ‘var’.

You can consider the memory of the iPhone as being made up of a large pile of bytes, each stacked one on top of another. Each byte has a number, called an address, associated with it, just like houses have an associated street number.Belowfigure  represents several bytes of the iPhone’s memory, starting at address 924 and extending through address 940.

* 

When you allocate a variable within your application, the compiler reserves an explicit amount of memory for it. For example, a statement such as **int x = 45** will cause the compiler to reserve 4 bytes of memory to store the current value of **x**. This is represented in figure 1 by the 4 bytes starting at address 928.

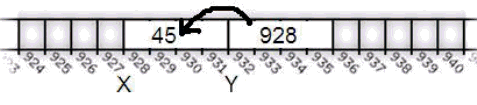
### Obtaining the Address of a Variable

Referring to the name of a variable within an expression will access or update its current value. By placing the **address-of** (&) operator in front of a variable name we are able to learn the address at which the variable is currently stored.

A variable that is able to store the address of another variable is called a “pointer”. This is because the variable is said to point to the location of another value. The following code snippet demonstrates how we can use the **address-of** operator:

|  |  |
| --- | --- |
| 1  2 | int x = 45;  int\* y = &x; |

This code snippet declares an integer variable **x** that is initialized to the value 45. It also declares variable y with a data type of **int\***. The \* at the end of the data type indicates a pointer and means we do not want to store an actual integer value but rather the memory address at which one can be found. This pointer is then initialized with the address of variable **x** using the **address-of** operator. If variable **x** had been stored at address 928 (as previously mentioned), we could graphically represent the result of executing this code snippet by updating the memory map to be similar to that in figure below.



Notice how the 4 bytes allocated to store variable y now store the number 928. When interpreted as an address, this indicates the location of variable **x**, as indicated by the arrow. The expression b>y = &x can be read as “place the address of variable x into variable y”. Following a pointer Once you have an address stored within a pointer variable, it is only natural to want to determine the value of whatever it points at. This operation is called dereferencing the pointer and is also achieved using the \* symbol, as demonstrated below.

|  |  |
| --- | --- |
| 1  2  3 | int x = 45;  int \*y = &x;  NSLog(@"The value was %d", \*y); |

The statement on the last line will print out the message “The value was 45” since the \* in front of variable **y** causes the compiler to follow the pointer and access the value it currently points at. Instead of reading the value it is also possible to replace it, as demonstrated below. Confusingly this also makes use of the \* operator:

|  |  |
| --- | --- |
| 1  2  3 | int x = 45;  int \*y = &x;  \*y = 92; |

The statement on the last line stores the value 92 at the address located within variable **y**. Referring to Figure 2, you will see that variable **y** stores (or points to) address 928; hence, executing this statement actually updates the value of variable **x**, even though **x** is never explicitly referred to within the statement.

**Dangling pointers :**

*Dangling pointers arise when an object is deleted or deallocated, without modifying the value of the pointer, so that the pointer still points to the memory location of the deallocated memory.*

As the system may reallocate the previously freed memory to another process, if the original program then dereferences the (now) dangling pointer, unpredictable behavior may result, as the memory may now contain completely different data. This is especially the case if the program writes data to memory pointed by a dangling pointer, as silent corruption of unrelated data may result, leading to subtle bugs that can be extremely difficult to find, or cause segmentation faults.

You cannot add a nil value to an NSArray or NSMutableArray. If you need to store a nil value, you need to use the NSNull wrapper class, as shown in that snippet you have.

We all agree that [NSNull null] is useful as a placeholder where an object is required, as elaborated above. But unless it's explicitly used in assignment for your object, it should not be used in comparison, a mistake I have made in the past.

id a; NSLog(@"Case 1"); if (a == nil) NSLog(@"a == nil"); if (a == Nil) NSLog(@"a == Nil"); if ([a isEqual:[NSNull null]]) NSLog(@"a isEqual:[NSNull null]"); NSLog(@"Case 2"); a = [NSNull null]; if (a == nil) NSLog(@"a == nil"); if (a == Nil) NSLog(@"a == Nil"); if ([a isEqual:[NSNull null]]) NSLog(@"a isEqual:[NSNull null]");

Output:

2014-01-31 10:57:11.179 MCDocsApp[13266:a0b] Case 1

2014-01-31 10:57:11.179 MCDocsApp[13266:a0b] a == nil

2014-01-31 10:57:11.179 MCDocsApp[13266:a0b] a == Nil

2014-01-31 10:57:11.180 MCDocsApp[13266:a0b] Case 2

2014-01-31 10:57:11.180 MCDocsApp[13266:a0b] a isEqual:[NSNull null]

Swift’s nil isn’t the same as nil in Objective-C. In Objective-C, nil is a pointer to a nonexistent object. In Swift, nil isn’t a pointer—it’s the absence of a value of a certain type. Optionals of *any*type can be set to nil, not just object types.

You set an optional variable to a valueless state by assigning it the special value nil:

1. var serverResponseCode: Int? = 404
2. // serverResponseCode contains an actual Int value of 404
3. serverResponseCode = nil
4. // serverResponseCode now contains no value

NOTE

You can’t use nil with non-optional constants and variables. If a constant or variable in your code needs to work with the absence of a value under certain conditions, always declare it as an optional value of the appropriate type.

If you define an optional variable without providing a default value, the variable is automatically set to nil for you:

1. var surveyAnswer: String?
2. // surveyAnswer is automatically set to nil

It's a memory debugging aid. Specifically, when you set NSZombieEnabled then whenever an object reaches retain count 0, rather than being deallocated it morphs itself into an NSZombie instance. Whenever such a zombie receives a message, it logs a warning rather than crashing or behaving in an unpredictable way. As such, you can debug subtle over-release/autorelease problems without advanced tools or painstaking needle in haystack searches.

The name is a fairly obvious play on the fact that objects are normally considered "dead" when they reach retain count 0. With this setting, they continue to exist in a strange half-life - neither living, nor quite dead. Much like real zombies, except they eat rather fewer brains.

Zombies are objects having retain count 0 still continuing to exist. To confirm it enable zombies in product -> Edit Schemes -> Diagnostics , enable the check mark for Zombie objects.

In order to dig deeper we need the help of detective “xcode instruments”. Open xcode ->developer tools -> instruments -> **select zombie**

<https://medium.com/@pratheeshdhayadev/xcode-instruments-zombies-8b262b1ae9d8>

Zombies are nothing but the objects which are half dead, or half released or retained.

Instruments is a great tool where we’ve tons of profiling templates; from “*Allocations*” to “*Leaks*” to “*Zombies*”.

Whenever you encounter EXC\_BAD\_ACCESS, it means that you are sending a message to an object that has already been released. This is the most common scenario, but there are exceptions as we'll discuss in a moment.

A pointer is nothing more than a variable that stores the memory address of another variable. When you send a message to an object, the pointer that points to the object you're sending the message to needs to be **dereferenced**. This means that you take the memory address the pointer is pointing to and access the value of that block of memory.

When that block of memory is no longer mapped for your application or, put differently, that block of memory isn't used for what you think it's used, it's no longer possible to access that chunk of memory. When this happens, the kernel sends an exception (**EXC**), indicating that your application cannot access that block of memory (**BAD ACCESS**).

In summary, when you run into EXC\_BAD\_ACCESS, it means that you try to send a message to a block of memory that can't execute that message.

<https://code.tutsplus.com/tutorials/what-is-exc_bad_access-and-how-to-debug-it--cms-24544>

iOS has no method of Garbage Collection. Even so, Garbage Collection is entirely unnecessary (for all practical purposes) when ARC is used. ARC works its magic at compile time to do the reference counting for you thereby making it unnecessary (and actually non-allowed) to use any other sort of memory management.

The *garbage collector*, or just *collector*, attempts to reclaim [*garbage*](https://en.wikipedia.org/wiki/Garbage_(computer_science)), or memory occupied by [objects](https://en.wikipedia.org/wiki/Object_(computer_science)) that are no longer in use by the [program](https://en.wikipedia.org/wiki/Application_software).

# Objective-C Succinctly: Categories and Extensions

A **category** is a group of related methods for a class, and all of the methods defined in a category are available through the class as if they were defined in the main interface file.

*Extensions* add new functionality to an existing class, structure, enumeration, or protocol type. This includes the ability to extend types for which you do not have access to the original source code (known as *retroactive modeling*). Extensions are similar to categories in Objective-C. (Unlike Objective-C categories, Swift extensions do not have names.)

Extensions in Swift can:

* Add computed instance properties and computed type properties
* Define instance methods and type methods
* Provide new initializers
* Define subscripts
* Define and use new nested types
* Make an existing type conform to a protocol

## Extension Syntax

Declare extensions with the extension keyword:

1. extension SomeType {
2. // new functionality to add to SomeType goes here
3. }

An extension can extend an existing type to make it adopt one or more protocols. To add protocol conformance, you write the protocol names the same way as you write them for a class or structure:

1. extension SomeType: SomeProtocol, AnotherProtocol {
2. // implementation of protocol requirements goes here
3. }

As per the [Apple Docs](https://developer.apple.com/library/ios/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/CustomizingExistingClasses/CustomizingExistingClasses.html) 1. a class extension can add its own properties and instance variables to a class 2. Class extensions are often used to extend the public interface with additional private methods or properties for use within the implementation of the class itself.

so if you declare the property in class extension it will be visible only to the implementation file. like

in BNREmployee.m

@interface BNREmployee () @property (nonatomic) unsigned int officeAlarmCode; @end @implementation BNREmployee - (void) someMethod { //officeAlarmCode will be available inside implementation block to use \_officeAlarmCode = 10; } @end

If you want to use officeAlarmCode in other classes, let's say OtherEmployee class then you need to create officeAlarmCode property in BNREmployee.h file with readOnly or readWrite access. Then you can use it like

BNREmployee.h @property (nonatomic, readOnly) unsigned int officeAlarmCode; //readOnly you can just read not write

in OtherEmployee.m

import "BNREmployee.h" @interface OtherEmployee () @property (nonatomic) unsigned int otherAlarmCode; @end @implementation OtherEmployee

you can create instance of BNREmployee and can assign officeAlarmCode value to otherAlarmCodeproperty like below

BNREmployee \*bnrEmployee = [BNREmployee alloc] init]; \_otherAlarmCode = bnrEmployee.officeAlarmCode;

In Swift, you can use Extensions to add new functionality to existing classes, structs and enumeration types.

They differ from Objective-C categories in a few ways, mainly:

* They aren't named
* You don't need to import an Extension explicitly. If you define an extension to add new functionality to an existing type, the new functionality will be available on all existing instances of that type, even if they were created before the extension was defined.
* As stated above, they work not only with classes, but with other types as well.

As it stands today, Extensions can:

* Add computed properties and computed static properties
* Define instance methods and type methods
* Provide new initializers
* Define subscripts
* Define and use new nested types
* Make an existing type conform to a protocol

The basic syntax to declare an extension is as follows:

extension SomeType { // new functionality to add to SomeType goes here }

Swift is heavily influenced by different programming paradigms from functional, imperative and object-oriented programming. This allows you as a developer to write very powerful and flexible code.

Protocol oriented programming in Swift helps you to bypass problems of object oriented-programming surrounding inheritance and their undesired complexity. But protocols still struggle with some compiler limitations when it comes to the usage of generic types. Type erasure is a pattern that can help us to bypass these limitations.

# Protocols & Associated Types

A major goal of protocols is to provide a blueprint of properties, functions and their requirements for a particular task. To be less error-prone, Swift checks this interface at compile time. So if you want to conform to a certain protocol, you have to fill in the required properties and functions. This provides you with a lot of safety but also some limitations.

Let’s start with a simple car factory example. We’ve got a protocol that describes the function of a car factory. This factory should produce Tesla cars. A protocol could look like this:

protocol TeslaFactoryProtocol {  
 func produce() -> Tesla  
}struct Tesla {  
 let type = "electric"  
 let brand = "Tesla"  
}struct TeslaFactory: TeslaFactoryProtocol {func produce() -> Tesla {  
 print("producing tesla car ...")  
 return Tesla()  
 }}

This example is narrowed down to Tesla cars. So let’s create a more generic factory protocol that can produce any kind of car. Perhaps like this:

protocol CarFactoryProtocol {  
 associatedtype CarType  
 func produce() -> CarType  
}struct ElectricCar {  
 let brand: String  
}struct PetrolCar {  
 let brand: String  
}...struct TeslaFactory: CarFactoryProtocol {  
 typealias CarType = ElectricCarfunc produce() -> TeslaFactory.CarType {  
 print("producing tesla electric car ...")  
 return ElectricCar(brand: "Tesla")  
 }  
}

Associated types work like generics. They can help you to remove the limitation of having to describe a concrete type in a protocol. By implementing associated types in the previous example, our protocol can describe a factory that produces any type of cars. typealias in the concrete implementation of the factory specifies the type of car that will be produced. To produce a car we simply need to create the factory and call the produce function:

let teslaFactory = TeslaFactory()  
teslaFactory.produce()// Output:  
// producing tesla electric car ...

Let’s create additional factories for more electric and petrol cars:

struct BMWFactory: CarFactoryProtocol {  
 typealias CarType = ElectricCarfunc produce() -> BMWFactory.CarType {  
 print("producing bmw electric car ...")  
 return ElectricCar(brand: "BMW")  
 }  
}struct ToyotaFactory: CarFactoryProtocol {  
 typealias CarType = PetrolCarfunc produce() -> ToyotaFactory.CarType {  
 print("producing toyota petrol car ...")  
 return PetrolCar(brand: "Toyota")  
 }  
}let bmwFactory = BMWFactory()  
bmwFactory.produce()let toyotaFactory = ToyotaFactory()  
toyotaFactory.produce()// Output:  
// producing bmw electric car ...  
// producing toyota petrol car ...

Let’s assume that we now want a list of all factories that produce an eletric car. Then we would implement it like this:

let electricCarFactories: [CarFactoryProtocol]

Now the Swift compiler will give you an error like this one:

*protocol ‘CarFactoryProtocol’ can only be used as a generic constraint because it has Self or associated type requirements.*

# The problem

The compiler error is caused by the problem that protocols are currently the only type that can make use of associated types. You can use it anywhere in your protocol. The problem is that the compiler isn’t aware of the concrete type at compile time and simply can’t handle and resolve the situation. The protocol CarFactoryProtocol is an abstract type and the swift compiler can’t instantiate it. So how do you fix this?

# Type erasure to the rescue

To help the compiler bypassing these protocol limitations you need to create a wrapper class. This helps to make the generic type of the protocol concrete. A popular example is the AnySequence class in the [Swift Standard Library](https://developer.apple.com/documentation/swift/anysequence?source=post_page---------------------------).

We need to ensure that the wrapper class only accepts instances that implement our protocol and where the type is the same as the one the wrapper class is initialized with:

struct AnyCarFactory<CarType>: CarFactoryProtocol {  
 private let \_produce: () -> CarTypeinit<Factory: CarFactoryProtocol>(\_ carFactory: Factory) where Factory.CarType == CarType {  
 \_produce = carFactory.produce  
 }func produce() -> CarType {  
 return \_produce()  
 }  
}

With this wrapper class we erase the type information. An electric car factory is no longer of type TeslaFactory or BMWFactory. It’s now of type AnyCarFactory<ElectricCar>. It’s now possible to create a list of all electric car factories without a compiler error:

let factories = [AnyCarFactory(TeslaFactory()), AnyCarFactory(BMWFactory())]  
factories.map() { $0.produce() }// Output:  
// producing tesla electric car ...  
// producing bmw electric car ...

# The Downsides and Limitations

* Writing wrapper classes always feels like writing unnecessary boilerplate code. Beside that the boilerplate increases the complexity of your code.
* There are still some limitations. You cannot create a list of factories ignoring the type of car they’re producing, like AnyCarFactory<AnyCar>. Swift currently doesn’t support replacing the initialized type with a generic one. Apple engineers are discussing to implement support for covariance in one of the next Swift versions. So this will hopefully get fixed in the future.

A wrapper class would be used if you have performance issues with a struct which has many properties which are reference types.

A generic wrapper class:

class Wrapper<T> { var value: T // or "let" instead of "var" init(\_ value: T) { self.value = value } }

This is because if you assign it to another variable all pointers of the properties get copied and therefore all reference counts (see ARC) get incremented (and decremented at the end).

This problem mainly occurs when you are looping over large arrays of such structs where at each iteration a new variable gets created.

With a wrapper class only its reference count gets incremented and decremented once.

I think you are confused on what one of those entities is, they are not similar in the slightest (apart from being a part of software engineering...).

A singleton is a single object (of any class you like) that can be accessed from anywhere in the entire project by any bit of code you like. It's a single point in space that is reference-able from any other point in space.

A "Wrapper Class" is a class that wraps another class (or primitive). Say you want to pass an int to a function or method, that method modifies that int and you want the calling object to see the change. Since ints are passed by value instead of by reference (by default) any changes you make to that int in the called function are ignored by the caller. So the caller sees no change in the int's value. Now if you make an Integer object and make it wrap an int you can now pass that Integer object by reference to functions or methods, modify the Integer object and get the modified value back into the calling function. A Wrapper Class wraps an object (or primitive) into another object to give it different functionality.

(a > b) ? a : b

If a is greater than b then a will be returned from MAX(a,b) function or if b is greater then if statement will be false and b will be returned.

# Type Casting

Type casting is a way to check the type of an instance, or to treat that instance as a different superclass or subclass from somewhere else in its own class hierarchy.

Type casting in Swift is implemented with the is and as operators. These two operators provide a simple and expressive way to check the type of a value or cast a value to a different type.

## Checking Type

Use the type check operator (is) to check whether an instance is of a certain subclass type. The type check operator returns true if the instance is of that subclass type and false if it is not.

let library = [

Movie(name: "Casablanca", director: "Michael Curtiz"),

Song(name: "Blue Suede Shoes", artist: "Elvis Presley"),

Movie(name: "Citizen Kane", director: "Orson Welles"),

Song(name: "The One And Only", artist: "Chesney Hawkes"),

Song(name: "Never Gonna Give You Up", artist: "Rick Astley")

]

var movieCount = 0

var songCount = 0

for item in library {

if item is Movie {

movieCount += 1

} else if item is Song {

songCount += 1

}

}

print("Media library contains \(movieCount) movies and \(songCount) songs")

// Prints "Media library contains 2 movies and 3 songs"

## Downcasting

A constant or variable of a certain class type may actually refer to an instance of a subclass behind the scenes. Where you believe this is the case, you can try to downcast to the subclass type with a type cast operator (as? or as!).

Because downcasting can fail, the type cast operator comes in two different forms. The conditional form, as?, returns an optional value of the type you are trying to downcast to. The forced form, as!, attempts the downcast and force-unwraps the result as a single compound action.

Use the conditional form of the type cast operator (as?) when you are not sure if the downcast will succeed. This form of the operator will always return an optional value, and the value will be nil if the downcast was not possible. This enables you to check for a successful downcast.

Use the forced form of the type cast operator (as!) only when you are sure that the downcast will always succeed. This form of the operator will trigger a runtime error if you try to downcast to an incorrect class type.

for item in library {

if let movie = item as? Movie {

print("Movie: \(movie.name), dir. \(movie.director)")

} else if let song = item as? Song {

print("Song: \(song.name), by \(song.artist)")

}

}

// Movie: Casablanca, dir. Michael Curtiz

// Song: Blue Suede Shoes, by Elvis Presley

// Movie: Citizen Kane, dir. Orson Welles

// Song: The One And Only, by Chesney Hawkes

// Song: Never Gonna Give You Up, by Rick Astley

## Type Casting for Any and AnyObject

Swift provides two special types for working with nonspecific types:

* Any can represent an instance of any type at all, including function types.
* AnyObject can represent an instance of any class type.

Here’s an example of using Any to work with a mix of different types, including function types and nonclass types. The example creates an array called things, which can store values of type Any:

var things = [Any]()

things.append(0)

things.append(0.0)

things.append(42)

things.append(3.14159)

things.append("hello")

things.append((3.0, 5.0))

things.append(Movie(name: "Ghostbusters", director: "Ivan Reitman"))

things.append({ (name: String) -> String in "Hello, \(name)" })

for thing in things {

switch thing {

case 0 as Int:

print("zero as an Int")

case 0 as Double:

print("zero as a Double")

case let someInt as Int:

print("an integer value of \(someInt)")

case let someDouble as Double where someDouble > 0:

print("a positive double value of \(someDouble)")

case is Double:

print("some other double value that I don't want to print")

case let someString as String:

print("a string value of \"\(someString)\"")

case let (x, y) as (Double, Double):

print("an (x, y) point at \(x), \(y)")

case let movie as Movie:

print("a movie called \(movie.name), dir. \(movie.director)")

case let stringConverter as (String) -> String:

print(stringConverter("Michael"))

default:

print("something else")

}

}

// zero as an Int

// zero as a Double

// an integer value of 42

// a positive double value of 3.14159

// a string value of "hello"

// an (x, y) point at 3.0, 5.0

// a movie called Ghostbusters, dir. Ivan Reitman

// Hello, Michael

NOTE

The Any type represents values of any type, including optional types. Swift gives you a warning if you use an optional value where a value of type Any is expected. If you really do need to use an optional value as an Any value, you can use the as operator to explicitly cast the optional to Any, as shown below.

1. let optionalNumber: Int? = 3
2. things.append(optionalNumber) // Warning
3. things.append(optionalNumber as Any) // No warning

as! operator is very similar to its sibling as?. Difference is as! is more evil.

Where as? will produce nil value is cast is unsuccessful, as! will result in crash if cast fails.

# Optional Chaining

Optional chaining is a process for querying and calling properties, methods, and subscripts on an optional that might currently be nil. If the optional contains a value, the property, method, or subscript call succeeds; if the optional is nil, the property, method, or subscript call returns nil. Multiple queries can be chained together, and the entire chain fails gracefully if any link in the chain is nil.

NOTE

Optional chaining in Swift is similar to messaging nil in Objective-C, but in a way that works for any type, and that can be checked for success or failure.

## Optional Chaining as an Alternative to Forced Unwrapping

Assert is to make sure a value is what its supposed to be. If an assertion fails that means something went wrong and so the app quits. One reason to use assert would be if you have some function that will not behave or will create very bad side effects if one of the parameters passed to it is not exactly some value (or a range of values) you can put an assert to make sure that value is what you expect it to be, and if it's not then something is really wrong, and so the app quits. Assert can be very useful for debugging/unit testing, and also when you provide frameworks to stop the users from doing "evil" things.

# XMLParserDelegate

### Handling XML

[func parserDidStartDocument(XMLParser)](https://developer.apple.com/documentation/foundation/xmlparserdelegate/1412065-parserdidstartdocument)

Sent by the parser object to the delegate when it begins parsing a document.

[func parserDidEndDocument(XMLParser)](https://developer.apple.com/documentation/foundation/xmlparserdelegate/1418172-parserdidenddocument)

Sent by the parser object to the delegate when it has successfully completed parsing.

[func parser(XMLParser, didStartElement: String, namespaceURI: String?, qualifiedName: String?, attributes: [String : String])](https://developer.apple.com/documentation/foundation/xmlparserdelegate/1415894-parser)

Sent by a parser object to its delegate when it encounters a start tag for a given element.

[func parser(XMLParser, didEndElement: String, namespaceURI: String?, qualifiedName: String?)](https://developer.apple.com/documentation/foundation/xmlparserdelegate/1417955-parser)

Sent by a parser object to its delegate when it encounters an end tag for a specific element.

Objective c

#pragma mark - UITableViewDataSource

Swift

// MARK: your text goes here

var name; // declaration

var name = "John" // initialization

A declaration provides basic attributes of a symbol: its type and its name. A definition provides all of the details of that symbol--if it's a function, what it does; if it's a class, what fields and methods it has; if it's a variable, where that variable is stored. Often, the compiler only needs to have a declaration for something in order to compile a file into an object file, expecting that the linker can find the definition from another file. If no source file ever defines a symbol, but it is declared, you will get errors at link time complaining about undefined symbols.

**Alloc:** Class method of NSObject. Returns a new instance of the receiving class.

**Init**: Instance method of NSObject. Implemented by subclasses to initialize a new object (the receiver) immediately after memory for it has been allocated.

**New**: Class method of NSObject. Allocates a new instance of the receiving class, sends it an init message, and returns the initialized object.

**Release**: Instance method of NSObject delegate. Decrements the receiver’s reference count.

**Autorelease**: Instance method of NSObject delegate. Adds the receiver to the current autorelease pool.

**Retain:** Instance method of NSObject delegate. Increments the receiver’s reference count.

**Copy:** Instance method of NSObject delegate. Returns a new instance that’s a copy of the receiver.

So to conclude we can say that

**alloc goes with init**

**new = alloc + ini**

[init](https://developer.apple.com/documentation/objectivec/nsobject/1418641-init?language=objc)

Implemented by subclasses to initialize a new object (the receiver) immediately after memory for it has been allocated.

An object is a chunk of memory bundled with the code that manipulates memory. In the memory, the object maintains its *state* (the values of its instance variables), which can change and evolve throughout its lifetime. To get a newly-created object off to a good start, its newly-allocated memory must be initialized to a proper initial state.

The central player in object initialization is the *constructor*. In Java, constructors are similar to methods, but they are not methods. Like a method, a constructor has a set of parameters and a body of code. Unlike methods, however, constructors have no return type. Like methods, you can give access specifiers to constructors, but unlike methods, constructors with public, protected, or package access are not inherited by subclasses. (Also, instead of determining the ability to invoke a method, the access level of a constructor determines the ability to instantiate an object.)

The essence of JS Injection is to inject the Javascript code, that will be run from the client-side.

JS Injection brings a lot of possibilities for a malicious user to modify the website's design, gain website's information, change the displayed website‘s information and manipulate with the parameters (for example, cookies). Therefore this can bring some serious website damages, information leakage and even hack.

The main purpose of JS Injection is to change the website's appearance and manipulate the parameters. Consequences of JS Injection can be very different – from damaging website‘s design to accessing someone else's account.

## Type Inference

Type inference enables a compiler to deduce the type of a particular expression automatically when it compiles your code, simply by examining the values you provide. Swift 4 uses type inference to work out the appropriate type as follows.

// varA is inferred to be of type Int var varA = 42 print(varA) // varB is inferred to be of type Double var varB = 3.14159 print(varB) // varC is also inferred to be of type Double var varC = 3 + 0.14159 print(varC)

In apps that run in iOS 8 and later, use the WKWebView class instead of using UIWebView. Additionally, consider setting the WKPreferences property javaScriptEnabled to false if you render files that are not supposed to run JavaScript.

Apple just recommends to use WKWebView, because it has better performance than UIWebView.

Finally I got my answer by lot of experiment,if we not using this method

-(BOOL)webView:(UIWebView \*)webView shouldStartLoadWithRequest:(NSURLRequest \*)request navigationType:(UIWebViewNavigationType)navigationType

-(void)webViewDidFinishLoad:(UIWebView \*)webView 2.

-(void)webView:(UIWebView \*)webView didFailLoadWithError:(NSError \*)error

-(void)webViewDidStartLoad:(UIWebView \*)webView

let webView = WKWebView(frame: CGRect(x: 0, y: 0, width: self.view.frame.size.width, height: self.view.frame.size.height))

self.view.addSubview(webView)

let url = URL(string: "https://www.youtube.com/watch?v=695PN9xaEhs")

webView.load(URLRequest(url: url!))

WKWebViewConfiguration \*theConfiguration = [[WKWebViewConfiguration alloc] init];

WKWebView \*webView = [[WKWebView alloc] initWithFrame:self.view.frame configuration:theConfiguration];

webView.navigationDelegate = self;

NSURL \*nsurl=[NSURL URLWithString:@"http://www.apple.com"];

NSURLRequest \*nsrequest=[NSURLRequest requestWithURL:nsurl];

[webView loadRequest:nsrequest]; [self.view addSubview:webView];

15

**Informal Protocol** : Category (Implementations are Optional)

**Formal Protocol** : Extension (Implementations are Optional and required)

There are two varieties of protocol, formal and informal:

* An **informal** protocol is a category on NSObject, which implicitly makes almost all objects adopters of the protocol. (A category is a language feature that enables you to add methods to a class without subclassing it.) Implementation of the methods in an informal protocol is optional. Before invoking a method, the calling object checks to see whether the target object implements it. Until optional protocol methods were introduced in Objective-C 2.0, informal protocols were essential to the way Foundation and AppKit classes implemented delegation.
* A **formal** protocol declares a list of methods that client classes are expected to implement. Formal protocols have their own declaration, adoption, and type-checking syntax. You can designate methods whose implementation is required or optional with the @required and @optional keywords. Subclasses inherit formal protocols adopted by their ancestors. A formal protocol can also adopt other protocols.

Formal protocols are an extension to the Objective-C language.

# [In objective C should i group 10 different protocols in a single header file or every protocol should be placed in it's class header file](https://stackoverflow.com/questions/17423704/in-objective-c-should-i-group-10-different-protocols-in-a-single-header-file-or)

There's no right or wrong answer to this. My *personal* preference is generally one protocol per header. However, if there are two or more protocols that go logically go together and will usually be imported together, you might put them in the same header file.

If your protocols form an API for a framework, that is another reason to put them together so classes that use the framework API can just do one import.

But I would recommend not using a generic name like protocol.h, try to think up something more descriptive of what the protocols are actually for e.g. all the protocols and class interfaces for Cocoa are logically imported (nested imports are used) in one header called Cocoa.h.

On the second part, I find it generally better to keep protocols and class interfaces in separate headers.

Dji tello

<https://medium.com/tarkalabs/automating-dji-tello-drone-using-gobot-2b711bf42af6>

# [How to detect total available/free disk space on the iPhone/iPad device?](https://stackoverflow.com/questions/5712527/how-to-detect-total-available-free-disk-space-on-the-iphone-ipad-device)

<https://stackoverflow.com/questions/5712527/how-to-detect-total-available-free-disk-space-on-the-iphone-ipad-device>

# What is Keychain Access on Mac?

Keychain Access is a macOS app that stores your passwords and account information and reduces the number of passwords you have to remember and manage.

When you access a website, email account, network server or other password-protected item, you may be given the option to remember or save the password. If you choose to save the password, it’s saved in your [keychain](https://support.apple.com/en-in/guide/keychain-access/aside/kyca5ef29af0/10.5/mac/10.14) so you don’t have to remember or type your password every time.

You can also use Keychain Access to manage certificates, which are issued by trusted organisations to validate websites, digital documents and other web-based materials.

Keychain Access works in collaboration with iCloud keychain, which lets you share keychains with your other devices.

Keychain offers a secure alternative to saving sensitive data, such as user names and passwords, with NSUserDefaults, plist or similar methods.

As you might already know, NSUserDefaults is simple and effective for saving small, simple bits of data, like NSNumbers or NSStrings, to your device’s file system. But this data is in no way stored securely as hackers can access it pretty easily from the device.

Keychain is great because data encryption automatically is taken care of before it is stored in the file system so there is no need to waste time building encryption algorithms.

A keychain in both OS and iOS can be configured to lock. When locked it is impossible to access and decrypt stored keychain items. For iOS the keychain is locked when the device is locked and unlocked when the device is unlocked. Even when it is unlocked, only apps that have created an item can access it, unless configured otherwise.

Keychain also offers other features like: accessing keychain items across apps. Normally, an app only has access to items it created but configuration can be made to let it access data within a group of designated apps.

Storing and retrieving data directly with Keychain is not an easy task. Unfortunately, the Keychain API is a bit torturous to use. It is written in C and requires a lot of time-consuming configuration. Luckily, Apple and many other contributors have created higher level wrappers to hide the convoluted C code and organization powering things from beneath.

Apple’s own Keychain wrapper is called [GenericKeychain](https://developer.apple.com/library/content/samplecode/GenericKeychain/Introduction/Intro.html?source=post_page---------------------------#//apple_ref/doc/uid/DTS40007797-Intro-DontLinkElementID_2) and is available within the sample code in both Objective C and Swift.

<https://medium.com/ios-os-x-development/securing-user-data-with-keychain-for-ios-e720e0f9a8e2>

<https://github.com/jrendel/SwiftKeychainWrapperExample>

With a tab bar controller, it will only rotate if your view controllers for *all* your tabs allow the orientation. Kind of annoying, but there it is. You need to override shouldAutorotateToInterfaceOrientation in all your other view controllers too, and they need to be able to smoothly adjust to those orientations.

If it's not practical to support landscape orientations on the other view controllers, maybe you could try some hacking, for example by subclassing UITabBarController and overriding its shouldAutorotateToInterfaceOrientation to instead return YES if the *current* view controller returns YES. But that might get your app rejected as not conforming to Human Interface Guidelines, since you are trying to circumvent standard interface behavior.

* + (UIInterfaceOrientationMask)application:(UIApplication \*)application supportedInterfaceOrientationsForWindow:(UIWindow \*)window
* Asks the delegate for the interface orientations to use for the view controllers in the specified window
* This method returns the total set of interface orientations supported by the app. When determining whether to rotate a particular view controller, the orientations returned by this method are intersected with the orientations supported by the root view controller or topmost presented view controller. The app and view controller must agree before the rotation is allowed.

If you do not implement this method, the app uses the values in the UIInterfaceOrientation key of the app’s Info.plist as the default interface orientations.

What we need is to encrypt our data using [AES encrypt standard](https://ru.wikipedia.org/wiki/Advanced_Encryption_Standard" \t "_blank).  
It is using to store secured data into iOS application.  
Main task is to use swift and iOS frameworks as possible.

What is AES?

It is Advanced Encryption Standard, symetric block encrypt mechanism. 256 in AES256 is a key size in bits.

In iOS we can use CommonCrypto library to encrypt/decrypt data.  
You can read more about CommonCrypto and Cryptographic Interfaces here [https://developer.apple.com/security/](https://developer.apple.com/security/" \t "_blank).

So, CommonCrypto was written some years ago in Objective-C.

If you are storing a large amount of custom data that you want to be available only after the user or device authenticates, then it's better to encrypt the data using an encryption framework. For example, you may have an app that can archive private chat messages saved by the user or private photos taken by the user, or which can store the user's financial details. In these cases, you would probably want to use encryption.

There are two common flows in applications for encrypting and decrypting data from iOS apps. Either the user is presented with a password screen, or the application is authenticated with a server which returns a key to decrypt the data.

[RNCryptor](https://github.com/RNCryptor/RNCryptor" \t "_blank) is a popular encryption wrapper for iOS that supports AES. RNCryptor is a great choice because it gets you up and running very quickly without having to worry about the underlying details. It is also open source so that security researchers can analyze and audit the code.

CCCryptor (AES encryption) wrappers for iOS and Mac in Swift. -- For ObjC, see RNCryptor/RNCryptor-objc

## Keychain

There are several techniques for mobile data encryption in iOS that can help you safeguard user data. Let’s look at them in detail:

Keychain offers built-in secure storage for small pieces of data (passwords, keys, etc.) on iOS and macOS. Data stored in Keychain is encrypted and isn't accessible by other apps running on a device (unlike NSUserDefaults, which is just an XML file with plain data). On iOS devices, Keychain is automatically locked when the device is locked and unlocked when the user unlocks the device.

Apple recommends always using iOS Keychain encryption for storing passwords, tokens, encryption keys, and similar data.

To work with Keychain, Apple provides the Keychain Services API. This is a low-level API written in C. Because it's a low-level API, adding it to an app is not a trivial task.

But fortunately, you can use a high-level wrapper from Apple – [KeychainWrapper](https://developer.apple.com/library/content/samplecode/GenericKeychain/Introduction/Intro.html)– or use some third-party solution like [SwiftKeychainWrapper](https://github.com/jrendel/SwiftKeychainWrapper).

<https://www.apriorit.com/dev-blog/436-data-encryption-ios>

**Man in middle attack**— Intracting http/https requests and responses is relatively easy to do when it comes to iOS apps. Using tools like Charles Proxy, even an amateur can get to know our app requests, corresponding server responses, and manipulate network traffic by sending doctored requests. Unfortunately, SSL is not enough to make your app secure.

If you are storing sensitive data in **UserDefaults**, then you are risking your application’s information.

**UserDefaults** get stored simply as a property list file that is located inside Preferences folder of your app. They get saved in our app without being encrypted .

Basically, by using a third party mac application like **iMazing** without even having to Jailbreak your device, you can easily view **UserDefaults** data for any app downloaded from the AppStore.

**Keychain** services API helps you solve these problems by giving your app a way to store the small amount of user data in an encrypted database called the **keychain**.

With the launch of iOS 9 and OS X El Capitan, Apple has introduced App Transport Security, which enforces developers to use secure network connections. This change implies that every connection the application makes must use HTTPS protocol and TLS 1.2.

In other words, our application cannot communicate with a server using a non-secure connection, such as HTTP, unless it is explicitly indicated. As this was a breaking change, Apple provided an easy way to master this new requirement by adding exceptions or disabling it in the plist file.

SSL Pinning is a technique that allows us to deal with an attack called *[Man in the Middle](http://www.information-age.com/what-are-mobile-man-middle-attacks-and-how-dangerous-are-they-123461071/" \t "_blank)*. SSL is based on the certificate’s “chain of trust”. When the communication starts, the client checks if the received server’s SSL certificate is trusted by any **SSL Certificate Authority.**

When discussing encryption, you generally hear about two types of encryption, symmetric and asymmetric encryption. Symmetric encryption, on the one hand, uses one shared key for encrypting and decrypting data. Asymmetric encryption, on the other hand, uses one key for encrypting data and another separate, but related, key for decrypting data.

In this tutorial, we'll leverage the Security framework available on iOS to encrypt and decrypt data. This process takes place under the hood so we won't be directly interacting with this framework. We'll use symmetric encryption in our example application.

The Security framework offers a number of other services, such as [Randomization services](https://developer.apple.com/library/ios/documentation/Security/Reference/RandomizationReference/Reference/reference.html#//apple_ref/doc/uid/TP40007281) for generating cryptographically secure random numbers, [Certificate, Key, and Trust Services](https://developer.apple.com/library/ios/documentation/Security/Reference/certifkeytrustservices/Reference/reference.html#//apple_ref/doc/uid/TP30000157) for managing certificates, public and private keys, and trust policies. The Security framework is a low-level framework available on both iOS and OS X with C-based APIs.

Core Data is not a Database. Core Data is using SQLite as it’s persistent store but the framework itself is not the database. Core Data does much more than databases like managing the object graphs, tracking the changes in the data and many more things.

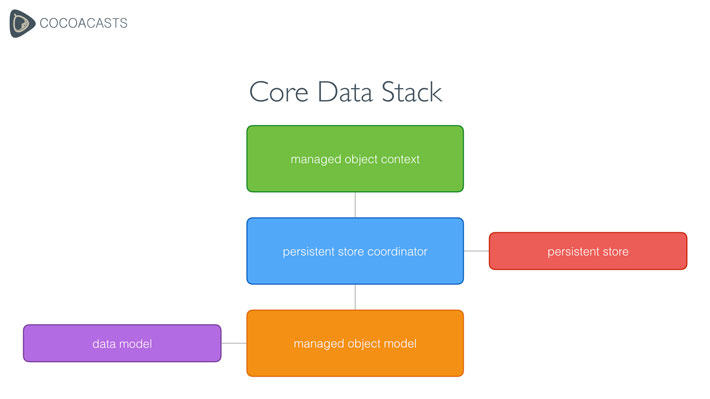
The new fileCoreDataDemo.xcdatamodeld acts as the model layer for the data that we want to save. We can easily ad the entity, attributes and relationships from the UI as like any other relational database.

Core Data can only do its magic because it keeps the object graph it manages in memory. This means that it can only operate on records once they’re in memory. This is very different from performing a SQL query on a database. If you want to delete thousands of records, Core Data first needs to load each record into memory. It goes without saying that this results in memory and performance issues if done incorrectly.

Another important limitation is the threading model of Core Data. The framework expects to be run on a single thread.

The stars of the framework are:

* the managed object model
* the managed object context
* and the persistent store coordinator



This diagram shows how these classes relate to one another. Let's start with the managed object model.

### Managed Object Model

The managed object model is an instance of the NSManagedObjectModel class. While a typical Core Data application has one instance of the NSManagedObjectModel class, it's possible to have multiple. The managed object model instance represents the data model of the Core Data application.

The above diagram shows that the managed object model is connected to the data model. The data model is represented by a file in the application bundle that contains the data schema of the application. The data schema is nothing more than a collection of entities.

An entity can have attributes and relationships, which make up the data model of the application. Remember for now that the managed object model an instance is of the NSManagedObjectModel class and that it the data model of the Core Data application represents.

### Managed Object Context

A managed object context is represented by an instance of the NSManagedObjectContextclass. A Core Data application has one or more managed object contexts, each managing a collection of model objects, instances of the NSManagedObject class.

The diagram we explored earlier illustrates that the managed object context receives model objects through a persistent store coordinator. It keeps a reference to the persistent store coordinator of the application.

The managed object context is the object you interact with most. It creates, reads, updates, and deletes model objects. From a developer’s perspective, the NSManagedObjectContext class is the workhorse of the Core Data framework.

### Persistent Store Coordinator

The persistent store coordinator is represented by an instance of the NSPersistentStoreCoordinator class and it plays a key role in every Core Data application. While it's possible to have multiple persistent store coordinators, most applications have only one. Very, very rarely is there a need to have multiple persistent store coordinators in an application.

The persistent store coordinator keeps a reference to the managed object model and every parent managed object context keeps a reference to the persistent store coordinator. Don't worry about parent and child managed object contexts for now.

The above diagram shows us that a persistent store coordinator is connected to one or more persistent stores. This brings us to the persistent store.

### Persistent Store

Remember that Core Data manages an object graph. The framework is only useful if the persistent store coordinator is connected to one or more persistent stores. Out of the box, Core Data supports three persistent store types:

* a SQLite database
* a binary store

## - an in-memory store

Each persistent store type has its pros and cons. Most applications use a SQLite database as their persistent store. As I mentioned earlier, SQLite is lightweight and very fast. It is great for mobile and desktop applications.

## How Does the Core Data Stack Work

Now that you know what the Core Data stack consists of, it's time to explore how it operates in an application. The heart of a Core Data application is the persistent store coordinator. The persistent store coordinator is instantiated first when the Core Data stack is created.

Checking the **Use Core Data** box will cause Xcode to generate boilerplate code for what’s known as an NSPersistentContainer in **AppDelegate.swift**.

The NSPersistentContainer consists of a set of objects that facilitate saving and retrieving information from Core Data. Inside this container is an object to manage the Core Data state as a whole, an object representing the Data Model, and so on.

Open **HitList.xcdatamodeld**. As you can see, Xcode has a powerful Data Model editor:

Click on **Add Entity** on the lower-left to create a new entity. Double-click the new entity and change its name to **Person**, like so:

Use the XCTest framework to write unit tests for your Xcode projects that integrate seamlessly with Xcode's testing workflow.

Tests assert that certain conditions are satisfied during code execution, and record test failures (with optional messages) if those conditions are not satisfied. Tests can also measure the performance of blocks of code to check for performance regressions, and can interact with an application's UI to validate user interaction flows.

Apple gives us two project level build configurations i.e debug and release. The debug configuration is usually used for the development and internal project setting like pointing the app to test environment and release is used to submit an app to App Store.

The Swift Package Manager is a tool for managing the distribution of Swift code. It’s integrated with the Swift build system to automate the process of downloading, compiling, and linking dependencies.

The Package Manager is included in Swift 3.0 and above.

# Playground Support

Share playground data, manage live views, and control the execution of a playground.

You use Playground Support from within playgrounds to:

* Access a playground page and manage its execution
* Access and share persistent data
* Assess the progress of the learner, update hints, and show success text

You also use Playground Support to display and dismiss *live views*, which show the results of running the code in a playground. You can create live views for your own types by leveraging the built-in live view representations available on many existing types.

# Closures

Closures are self-contained blocks of functionality that can be passed around and used in your code. Closures in Swift are similar to blocks in C and Objective-C and to lambdas in other programming languages.

Closures can capture and store references to any constants and variables from the context in which they are defined. This is known as *closing over* those constants and variables. Swift handles all of the memory management of capturing for you.

Global and nested functions, as introduced in [Functions](https://docs.swift.org/swift-book/LanguageGuide/Functions.html), are actually special cases of closures. Closures take one of three forms:

* Global functions are closures that have a name and do not capture any values.
* Nested functions are closures that have a name and can capture values from their enclosing function.
* Closure expressions are unnamed closures written in a lightweight syntax that can capture values from their surrounding context.

One way to provide the sorting closure is to write a normal function of the correct type, and to pass it in as an argument to the sorted(by:) method:

let names = ["Chris", "Alex", "Ewa", "Barry", "Daniella"]

func backward(\_ s1: String, \_ s2: String) -> Bool {

return s1 > s2

}

var reversedNames = names.sorted(by: backward)

// reversedNames is equal to ["Ewa", "Daniella", "Chris", "Barry", "Alex"]

### Closure Expression Syntax

Closure expression syntax has the following general form:

1. { (parameters) -> return type in
2. statements
3. }
4. reversedNames = names.sorted(by: { (s1: String, s2: String) -> Bool in
5. return s1 > s2

})

|  |
| --- |
| //: Playground - Closures |
|  |  |
|  | // Closure take no parameter and return nothing |
|  | let sayHello: () -> Void = { |
|  | print("Hello") |
|  | } |
|  |  |
|  | sayHello() |
|  |  |
|  | // Closure take one parameter and return 1 parameter |
|  | let value: (Int) -> Int = { (value1) in |
|  | return value1 |
|  | } |
|  |  |
|  | print(value(5)) |
|  |  |
|  | // Closure take two parameter and return 1 parameter |
|  | let add: (Int, Int) -> Int = { (value1, value2) in |
|  | return value1 + value2 |
|  | } |
|  |  |
|  | print(add(5, 4)) |
|  |  |
|  | // Closure take two parameter and return String parameter |
|  | let addValues: (Int, Int) -> String = { (value1, value2) -> String in |
|  | return String("Sum is: \(value1 + value2)") |
|  | } |
|  |  |
|  | print(addValues(5, 4)) |

Functions are a special cases of closures. Closures take one of three forms:

* Global functions: They have a name and can’t capture value.
* Nested functions: They have a name and can capture values from their enclosing function.
* Closure expressions: They don’t have name and can capture values from their surrounding context.

# Shorthand Argument Names

Closure arguments can refer to a position i.e. $0 , $1, $2, $3 and so on.

# Trailing Closure:

If you need to pass a closure expression to a function as the function’s last argument and closure expression is too long, it can be written as trailing closure. A trailing closure is written after the function call’s parentheses (), even though it is still an argument to the function. When you use the trailing closure syntax, you don’t write the argument label for the closure as part of the function call.

Closures are just un-named or anonymous functions

Closures are also functions. But when a function captures state upon its creation, we call it a closure.

You can use AnyObject array to hold any kind of objects in the same array:

var objectsArray = [AnyObject]() objectsArray.append("Foo") objectsArray.append(2) // And also the inmutable version let objectsArray: [AnyObject] = ["Foo", 2] // This way you can let the compiler infer the type let objectsArray = ["Foo", 2]

### Variadic Parameters

A variadic parameter accepts zero or more values of a specified type. You use a variadic parameter to specify that the parameter can be passed a varying number of input values when the function is called. Write variadic parameters by inserting three period characters (...) after the parameter’s type name.

The values passed to a variadic parameter are made available within the function’s body as an array of the appropriate type. For example, a variadic parameter with a name of numbersand a type of Double... is made available within the function’s body as a constant array called numbers of type [Double].

The example below calculates the arithmetic mean (also known as the average) for a list of numbers of any length:

1. func arithmeticMean(\_ numbers: Double...) -> Double {
2. var total: Double = 0
3. for number in numbers {
4. total += number
5. }
6. return total / Double(numbers.count)
7. }
8. arithmeticMean(1, 2, 3, 4, 5)
9. // returns 3.0, which is the arithmetic mean of these five numbers
10. arithmeticMean(3, 8.25, 18.75)
11. // returns 10.0, which is the arithmetic mean of these three numbers

NOTE

A function may have at most one variadic parameter.

In computer programming, a **variadic function** is a function which accepts a variable number of arguments. The function arguments are represented by **…** (three period characters) after the argument’s type that can be accessed into their body as an array .

You might have used a print statement in swift. It looks like this:

print(“apple”,”pineapple”,”orange”)

func printFruitNames (\_ fruits: String...) -> () {for fruit in fruits { // fruits is a string arrayprint(“\(fruit)”)}}printFruitNames(“apple”,”pineapple”,”orange”)

# Special Literals in Swift

Today I learned about special literals in Swift. These literal won't add any functionality benefits to your code, but are excellent while debugging and pinpointing error and bugs. There might be more, but here is the list of literals I came across today.

* **#file**  
  Outputs the name of the file in which code is being run
* **#function**  
  Outputs the name of the function where code belongs
* **#line**  
  Output the line number at which code is being run
* **#column**  
  Output the column number in which code is being run

### Literal Expression

A literal expression consists of either an ordinary literal (such as a string or a number), an array or dictionary literal, a playground literal, or one of the following special literals:

| **Literal** | **Type** | **Value** |
| --- | --- | --- |
| #file | String | The name of the file in which it appears. |
| #line | Int | The line number on which it appears. |
| #column | Int | The column number in which it begins. |
| #function | String | The name of the declaration in which it appears. |
| #dsohandle | UnsafeRawPointer | The DSO (dynamic shared object) handle in use where it appears. |

Inside a function, the value of #function is the name of that function, inside a method it is the name of that method, inside a property getter or setter it is the name of that property, inside special members like init or subscript it is the name of that keyword, and at the top level of a file it is the name of the current module.

Have you ever wanted to share a chunk of code between two or more of your apps, or wanted to share a part of your program with other developers?

Frameworks have three major purposes:

* Code encapsulation.
* Code modularity.
* Code reuse.
* You can share your framework with your other apps, team members, or the iOS community. When combined with Swift’s **access control**, frameworks help define strong, testable interfaces between code modules.
* In Swift parlance, a **module** is a compiled group of code that is distributed together. A framework is one type of module, and an app is another example.

Frameworks are self-contained, reusable chunks of code and resources that you can import into any number of apps and even share across iOS, tvOS, watchOS, and macOS apps

so creating frameworks has never been easier. The first thing to do is to create the project for the framework.

1. In Xcode, select **File ▸ New ▸ Project…**.
2. Choose **iOS ▸ Framework & Library ▸ Cocoa Touch Framework**.
3. Click **Next**.
4. Set the **Product Name** to **KnobControl**. Use your own **Organization Name** and **Organization Identifier**.

You’ll link the framework to the app’s target to fix this problem. First, expand the **KnobControl** project to see the **Products** folder, and then look for **KnobControl.framework** beneath it. This file is the output of the framework project that packages up the binary code, headers, resources and metadata.

The software development life cycle (SDLC) is a framework defining tasks performed at each step in the software development process. SDLC is a structure followed by a development team within the software organization. It consists of a detailed plan describing how to develop, maintain and replace specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

The software development life cycle is also known as the software development process.

SDLC consists of following activities:

1. Planning: The most important parts of software development, requirement gathering or requirement analysis are usually done by the most skilled and experienced software engineers in the organization. After the requirements are gathered from the client, a scope document is created in which the scope of the project is determined and documented.
2. Implementation: The software engineers start writing the code according to the client's requirements.
3. Testing: This is the process of finding defects or bugs in the created software.
4. Documentation: Every step in the project is documented for future reference and for the improvement of the software in the development process. The design documentation may include writing the application programming interface (API).
5. Deployment and maintenance: The software is deployed after it has been approved for release.
6. Maintaining: Software maintenance is done for future reference. Software improvement and new requirements (change requests) can take longer than the time needed to create the initial development of the software.

There are several software development models followed by various organizations:

* Waterfall Model: This model involves finishing each phase completely before commencing the next one. When each phase is completed successfully, it is reviewed to see if the project is on track and whether it is feasible to continue.
* V-Shaped Model: This model focuses on the execution of processes in a sequential manner, similar to the waterfall model but with more importance placed on testing. Testing procedures are written even before the commencement of writing code. A system plan is generated before starting the development phase.
* Incremental Model: This life cycle model involves multiple development cycles. The cycles are divided up into smaller iterations. These iterations can be easily managed and go through a set of phases including requirements, design, implementation and testing. A working version of the software is produced during the first iteration, so working software is created early in the development process.

## SDLC Phases

The entire SDLC process divided into the following stages:

* Phase 1: Requirement collection and analysis
* Phase 2: Feasibility study:
* Phase 3: Design:
* Phase 4: Coding:
* Phase 5: Testing:
* Phase 6: Installation/Deployment:
* Phase 7: Maintenance:

In this tutorial, I have explained all these phases

### Phase 1: Requirement collection and analysis:

The requirement is the first stage in the SDLC process. It is conducted by the senior team members with inputs from all the stakeholders and domain experts in the industry. Planning for the quality assurance requirements and recognization of the risks involved is also done at this stage.

This stage gives a clearer picture of the scope of the entire project and the anticipated issues, opportunities, and directives which triggered the project.

Requirements Gathering stage need teams to get detailed and precise requirements. This helps companies to finalize the necessary timeline to finish the work of that system.

### Phase 2: Feasibility study:

Once the requirement analysis phase is completed the next step is to define and document software needs. This process conducted with the help of 'Software Requirement Specification' document also known as 'SRS' document. It includes everything which should be designed and developed during the project life cycle.

**There are mainly five types of feasibilities checks:**

* **Economic:**Can we complete the project within the budget or not?
* **Legal:** Can we handle this project as cyber law and other regulatory framework/compliances.
* **Operation feasibility:** Can we create operations which is expected by the client?
* **Technical:** Need to check whether the current computer system can support the software
* **Schedule:** Decide that the project can be completed within the given schedule or not.

### Phase 3: Design:

In this third phase, the system and software design documents are prepared as per the requirement specification document. This helps define overall system architecture.

This design phase serves as input for the next phase of the model.

There are two kinds of design documents developed in this phase:

High-Level Design (HLD)

* Brief description and name of each module
* An outline about the functionality of every module
* Interface relationship and dependencies between modules
* Database tables identified along with their key elements
* Complete architecture diagrams along with technology details

Low-Level Design(LLD)

* Functional logic of the modules
* Database tables, which include type and size
* Complete detail of the interface
* Addresses all types of dependency issues
* Listing of error messages
* Complete input and outputs for every module

### Phase 4: Coding:

Once the system design phase is over, the next phase is coding. In this phase, developers start build the entire system by writing code using the chosen programming language. In the coding phase, tasks are divided into units or modules and assigned to the various developers. It is the longest phase of the Software Development Life Cycle process.

In this phase, Developer needs to follow certain predefined coding guidelines. They also need to use programming tools like compiler, interpreters, debugger to generate and implement the code.

### Phase 5: Testing:

Once the software is complete, and it is deployed in the testing environment. The testing team starts testing the functionality of the entire system. This is done to verify that the entire application works according to the customer requirement.

During this phase, QA and testing team may find some bugs/defects which they communicate to developers. The development team fixes the bug and send back to QA for a re-test. This process continues until the software is bug-free, stable, and working according to the business needs of that system.

### Phase 6: Installation/Deployment:

Once the software testing phase is over and no bugs or errors left in the system then the final deployment process starts. Based on the feedback given by the project manager, the final software is released and checked for deployment issues if any.

### Phase 7: Maintenance:

Once the system is deployed, and customers start using the developed system, following 3 activities occur

* Bug fixing - bugs are reported because of some scenarios which are not tested at all
* Upgrade - Upgrading the application to the newer versions of the Software
* Enhancement - Adding some new features into the existing software

The main focus of this SDLC phase is to ensure that needs continue to be met and that the system continues to perform as per the specification mentioned in the first phase.

## Popular SDLC models

Here, are some most important phases of SDLC life cycle:

**Waterfall model**

The waterfall is a widely accepted SDLC model. In this approach, the whole process of the software development is divided into various phases. In this SDLC model, the outcome of one phase acts as the input for the next phase.

This SDLC model is documentation-intensive, with earlier phases documenting what need be performed in the subsequent phases.

**Incremental Approach**

The incremental model is not a separate model. It is essentially a series of waterfall cycles. The requirements are divided into groups at the start of the project. For each group, the SDLC model is followed to develop software. The SDLC process is repeated, with each release adding more functionality until all requirements are met. In this method, every cycle act as the maintenance phase for the previous software release. Modification to the incremental model allows development cycles to overlap. After that subsequent cycle may begin before the previous cycle is complete.

**V-Model**

In this type of SDLC model testing and the development, the phase is planned in parallel. So, there are verification phases on the side and the validation phase on the other side. V-Model joins by Coding phase.

**Agile Model**

Agile methodology is a practice which promotes continue interaction of development and testing during the SDLC process of any project. In the Agile method, the entire project is divided into small incremental builds. All of these builds are provided in iterations, and each iteration lasts from one to three weeks.

**Spiral Model**

The spiral model is a risk-driven process model. This SDLC model helps the team to adopt elements of one or more process models like a waterfall, incremental, waterfall, etc.

This model adopts the best features of the prototyping model and the waterfall model. The spiral methodology is a combination of rapid prototyping and concurrency in design and development activities.

**Big bang model**

Big bang model is focusing on all types of resources in software development and coding, with no or very little planning. The requirements are understood and implemented when they come.

This model works best for small projects with smaller size development team which are working together. It is also useful for academic software development projects. It is an ideal model where requirements is either unknown or final release date is not given.

**Conclusion**

* The SDLC is a systematic process for building software that ensures the quality and correctness of the software built
* SDLC process provides a framework for a standard set of activities and deliverables
* Seven different SDLC stages are 1) Requirement collection and analysis 2) Feasibility study: 3) Design 4) Coding 5) Testing: 6) Installation/Deployment and 7) Maintenance
* The senior team members conduct the requirement analysis phase
* Feasibility Study stage includes everything which should be designed and developed during the project life cycle
* In the Design phase, the system and software design documents are prepared as per the requirement specification document
* In the coding phase, developers start build the entire system by writing code using the chosen programming language
* Testing is the next phase which is conducted to verify that the entire application works according to the customer requirement.
* Installation and deployment face begins when the software testing phase is over, and no bugs or errors left in the system
* Bug fixing, upgrade, and engagement actions covered in the maintenance face
* Waterfall, Incremental, Agile, V model, Spiral, Big Bang are some of the popular SDLC models
* SDLC consists of a detailed plan which explains how to plan, build, and maintain specific software

### Different Types of Software Testing

Given below is the list of some common types of Software Testing:

**Functional testing types include:**

* Unit testing
* Integration testing
* System testing
* Sanity testing
* Smoke testing
* Interface testing
* Regression testing
* Beta/Acceptance testing

**Non-functional testing types include:**

* Performance Testing
* Load testing
* Stress testing
* Volume testing
* Security testing
* Compatibility testing
* Install testing
* Recovery testing
* Reliability testing
* Usability testing
* Compliance testing
* Localization testing

#### #1) Alpha Testing

It is the most common type of testing used in the Software industry. The objective of this testing is to identify all possible issues or defects before releasing it into the market or to the user.

Alpha testing is carried out at the end of the software development phase but before the Beta Testing. Still, minor design changes may be made as a result of such testing. [Alpha testing](https://www.softwaretestinghelp.com/what-is-alpha-testing-beta-testing/) is conducted at the developer’s site. In-house virtual user environment can be created for this type of testing.

#### #2) Acceptance Testing

An [acceptance test](https://www.softwaretestinghelp.com/what-is-acceptance-testing/) is performed by the client and verifies whether the end to end the flow of the system is as per the business requirements or not and if it is as per the needs of the end user. Client accepts the software only when all the features and functionalities work as expected.

It is the last phase of the testing, after which the software goes into production. This is also called User Acceptance Testing (UAT).

#### #3) Ad-hoc Testing

The name itself suggests that this testing is performed on [an ad-hoc](https://www.softwaretestinghelp.com/ad-hoc-testing/) basis i.e. with no reference to the test case and also without any plan or documentation in place for such type of testing. The objective of this testing is to find the defects and break the application by executing any flow of the application or any random functionality.

Ad-hoc testing is an informal way of finding defects and can be performed by anyone in the project. It is difficult to identify defects without a test case but sometimes it is possible that defects found during ad-hoc testing might not have been identified using existing test cases.

#### #4) Accessibility Testing

The aim of [accessibility testing](https://www.softwaretestinghelp.com/what-is-web-accessibility-testing/) is to determine whether the software or application is accessible for disabled people or not. Here disability means deaf, color blind, mentally disabled, blind, old age and other disabled groups. Various checks are performed such as font size for visually disabled, color and contrast for color blindness etc.

#### #5) Beta Testing

[Beta Testing](https://www.softwaretestinghelp.com/beta-testing/) is a formal type of software testing which is carried out by the customer. It is performed in **the Real Environment**before releasing the product to the market for the actual end users.

Beta testing is carried out to ensure that there are no major failures in the software or product and it satisfies the business requirements from an end-user perspective. Beta testing is successful when the customer accepts the software.

Usually, this testing is typically done by end-users or others. It is the final testing done before releasing an application for commercial purpose. Usually, the Beta version of the software or product released is limited to a certain number of users in a specific area.

So end user actually uses the software and shares the feedback to the company. Company then takes necessary action before releasing the software to the worldwide.

#### #6) Back-end Testing

Whenever an input or data is entered on front-end application, it stores in the database and the testing of such database is known as Database Testing or Backend testing. There are different databases like SQL Server, MySQL, and Oracle etc. Database testing involves testing of table structure, schema, stored procedure, data structure and so on.

In back-end testing GUI is not involved, testers are directly connected to the database with proper access and testers can easily verify data by running a few queries on the database. There can be issues identified like data loss, deadlock, data corruption etc during this back-end testing and these issues are critical to fixing before the system goes live into the production environment

#### #7) Browser Compatibility Testing

It is a subtype of Compatibility Testing (which is explained below) and is performed by the testing team.

[Browser Compatibility Testing](https://www.softwaretestinghelp.com/how-is-cross-browser-testing-performed/) is performed for web applications and it ensures that the software can run with the combination of different browser and operating system. This type of testing also validates whether web application runs on all versions of all browsers or not.

#### #8) Backward Compatibility Testing

It is a type of testing which validates whether the newly developed software or updated software works well with older version of the environment or not.

Backward Compatibility Testing checks whether the new version of the software works properly with file format created by older version of the software; it also works well with data tables, data files, data structure created by older version of that software. If any of the software is updated then it should work well on top of the previous version of that software.

#### #9) Black Box Testing

Internal system design is not considered in this type of testing. Tests are based on the requirements and functionality.

Detailed information about the advantages, disadvantages, and [types of Black box testing](https://www.softwaretestinghelp.com/black-box-testing/) can be seen here.

#### #10) Boundary Value Testing

This type of testing checks the behavior of the application at the boundary level.

[Boundary value Testing](https://www.softwaretestinghelp.com/what-is-boundary-value-analysis-and-equivalence-partitioning/) is performed for checking if defects exist at boundary values. Boundary value testing is used for testing a different range of numbers. There is an upper and lower boundary for each range and testing is performed on these boundary values.

If testing requires a test range of numbers from 1 to 500 then Boundary Value Testing is performed on values at 0, 1, 2, 499, 500 and 501.

#### #11) Branch Testing

It is a type of white box testing and is carried out during unit testing. Branch Testing, the name itself suggests that the code is tested thoroughly by traversing at every branch.

#### #12) Comparison Testing

Comparison of a product's strength and weaknesses with its previous versions or other similar products is termed as Comparison Testing.

#### #13) Compatibility Testing

It is a testing type in which it validates how software behaves and runs in a different environment, web servers, hardware, and network environment. [Compatibility testing](https://www.softwaretestinghelp.com/software-compatibility-testing/) ensures that software can run on a different configuration, different database, different browsers, and their versions. Compatibility testing is performed by the testing team.

#### #14) Component Testing

It is mostly performed by developers after the completion of unit testing. [Component Testing](https://www.softwaretestinghelp.com/what-is-component-testing-or-module-testing/) involves testing of multiple functionalities as a single code and its objective is to identify if any defect exists after connecting those multiple functionalities with each other.

#### #15) End-to-End Testing

Similar to system testing, [End-to-end testing](https://www.softwaretestinghelp.com/what-is-end-to-end-testing/) involves testing of a complete application environment in a situation that mimics real-world use, such as interacting with a database, using network communications, or interacting with other hardware, applications, or systems if appropriate.

#### #16) Equivalence Partitioning

It is a testing technique and a type of Black Box Testing. During this [equivalence partitioning](https://www.softwaretestinghelp.com/what-is-boundary-value-analysis-and-equivalence-partitioning/), a set of group is selected and a few values or numbers are picked up for testing. It is understood that all values from that group generate the same output.

The aim of this testing is to remove redundant test cases within a specific group which generates the same output but not any defect.

Suppose, application accepts values between -10 to +10 so using equivalence partitioning the values picked up for testing are zero, one positive value, one negative value. So the Equivalence Partitioning for this testing is: -10 to -1, 0, and 1 to 10.

#### #17) Example Testing

It means real-time testing. Example testing includes the real-time scenario, it also involves the scenarios based on the experience of the testers.

#### #18) Exploratory Testing

Exploratory Testing is informal testing performed by the testing team. The objective of this testing is to explore the application and looking for defects that exist in the application. Sometimes it may happen that during this testing major defect discovered can even cause system failure.

During exploratory testing, it is advisable to keep a track of what flow you have tested and what activity you did before the start of the specific flow.

[An exploratory testing technique](https://www.softwaretestinghelp.com/what-is-exploratory-testing/) is performed without documentation and test cases.

#### #20) Functional Testing

This type of testing ignores the internal parts and focuses only on the output to check if it is as per the requirement or not. It is a Black-box type testing geared to the functional requirements of an application. For detailed information about Functional Testing click [here](https://www.softwaretestinghelp.com/guide-to-functional-testing/).

#### #21) Graphical User Interface (GUI) Testing

The objective of this GUI testing is to validate the GUI as per the business requirement. The expected GUI of the application is mentioned in the Detailed Design Document and GUI mockup screens.

The GUI testing includes the size of the buttons and input field present on the screen, alignment of all text, tables and content in the tables.

It also validates the menu of the application, after selecting different menu and menu items, it validates that the page does not fluctuate and the alignment remains same after hovering the mouse on the menu or sub-menu.

#### #22) Gorilla Testing

Gorilla Testing is a testing type performed by a tester and sometimes by developer the as well. In Gorilla Testing, one module or the functionality in the module is tested thoroughly and heavily. The objective of this testing is to check the robustness of the application.

#### #23) Happy Path Testing

The objective of Happy Path Testing is to test an application successfully on a positive flow. It does not look for negative or error conditions. The focus is only on the valid and positive inputs through which application generates the expected output.

#### #24) Incremental Integration Testing

[Incremental Integration Testing](https://www.softwaretestinghelp.com/incremental-testing/) is a Bottom-up approach for testing i.e continuous testing of an application when a new functionality is added. Application functionality and modules should be independent enough to test separately. This is done by programmers or by testers.

#### #25) Install/Uninstall Testing

[Installation and uninstallation testing](https://www.softwaretestinghelp.com/software-installationuninstallation-testing/) is done on full, partial, or upgrade install/uninstall processes on different operating systems under different hardware or software environment.

#### #26) Integration Testing

Testing of all integrated modules to verify the combined functionality after integration is [termed as Integration Testing](https://www.softwaretestinghelp.com/what-is-integration-testing/). Modules are typically code modules, individual applications, client and server applications on a network, etc. This type of testing is especially relevant to client/server and distributed systems.

#### #27) Load Testing

It is a type of non-functional testing and the objective of Load testing is to check how much of load or maximum workload a system can handle without any performance degradation.

[Load testing helps](https://www.softwaretestinghelp.com/introduction-to-performance-testing-loadrunner-training-tutorial-part-1/) to find the maximum capacity of the system under specific load and any issues that cause the software performance degradation. Load testing is performed using tools like[JMeter](https://www.softwaretestinghelp.com/jmeter-tutorials/), LoadRunner, WebLoad, Silk performer etc.

#### #28) Monkey Testing

[Monkey testing](https://www.softwaretestinghelp.com/what-is-monkey-testing-in-software-testing/) is carried out by a tester assuming that if the monkey uses the application then how random input, values will be entered by the Monkey without any knowledge or understanding of the application.

The objective of Monkey Testing is to check if an application or system gets crashed by providing random input values/data. Monkey Testing is performed randomly and no test cases are scripted and it is not necessary to

Monkey Testing is performed randomly and no test cases are scripted and it is not necessary to be aware of the full functionality of the system.

#### #29) Mutation Testing

[Mutation Testing](https://www.softwaretestinghelp.com/what-is-mutation-testing/) is a type of white box testing in which the source code of one of the program is changed and verifies whether the existing test cases can identify these defects in the system. The change in the program source code is very minimal so that it does not impact the entire application, only the specific area having the impact and the related test cases should able to identify those errors in the system.

#### #30) Negative Testing

Testers having the mindset of “attitude to break” and using negative testing they validate that if system or application breaks. [A negative testing technique](https://www.softwaretestinghelp.com/what-is-negative-testing/) is performed using incorrect data, invalid data or input. It validates that if the system throws an error of invalid input and behaves as expected.

#### #31) Non-Functional Testing

It is a type of testing for which every organization having a separate team which usually called as Non-Functional Test (NFT) team or Performance team.

[Non-functional testing involves](https://www.softwaretestinghelp.com/what-is-non-functional-testing/) testing of non-functional requirements such as Load Testing, Stress Testing, Security, Volume, Recovery Testing etc. The objective of NFT testing is to ensure whether the response time of software or application is quick enough as per the business requirement.

It should not take much time to load any page or system and should sustain during peak load.

#### #32) Performance Testing

This term is often used interchangeably with ‘stress' and ‘load' testing. [Performance Testing](https://www.softwaretestinghelp.com/introduction-to-performance-testing-loadrunner-training-tutorial-part-1/) is done to check whether the system meets the performance requirements. Different performance and load tools are used to do this testing.

#### #33) Recovery Testing

It is a type of testing which validates that how well the application or system recovers from crashes or disasters.

Recovery testing determines if the system is able to continue the operation after a disaster. Assume that application is receiving data through the network cable and suddenly that network cable has been unplugged.

Sometime later, plug the network cable; then the system should start receiving data from where it lost the connection due to network cable unplugged.

#### #34) Regression Testing

Testing an application as a whole for the modification in any module or functionality is termed as Regression Testing. It is difficult to cover all the system in [Regression Testing](https://www.softwaretestinghelp.com/regression-testing-tools-and-methods/), so typically [automation testing tools](https://www.softwaretestinghelp.com/automation-testing-tutorial-1/) are used for these types of testing.

#### #35) Risk-Based Testing (RBT)

In [Risk Based Testing](https://www.softwaretestinghelp.com/risk-management-during-test-planning-risk-based-testing/), the functionalities or requirements are tested based on their priority. Risk-based testing includes testing of highly critical functionality, which has the highest impact on business and in which the probability of failure is very high.

The priority decision is based on the business need, so once priority is set for all functionalities then high priority functionality or test cases are executed first followed by medium and then low priority functionalities.

The low priority functionality may be tested or not tested based on the available time.

The Risk-based testing is carried out if there is insufficient time available to test entire software and software needs to be implemented on time without any delay. This approach is followed only by the discussion and approval of the client and senior management of the organization.

#### #36) Sanity Testing

[Sanity Testing](https://www.softwaretestinghelp.com/smoke-testing-and-sanity-testing-difference/) is done to determine if a new software version is performing well enough to accept it for a major testing effort or not. If an application is crashing for the initial use then the system is not stable enough for further testing. Hence a build or an application is assigned to fix it.

#### #37) Security Testing

It is a type of testing performed by a special team of testers. A system can be penetrated by any hacking way.

[Security Testing](https://www.softwaretestinghelp.com/how-to-test-application-security-web-and-desktop-application-security-testing-techniques/) is done to check how the software or application or website is secure from internal and external threats. This testing includes how much software is secure from the malicious program, viruses and how secure and strong the authorization and authentication processes are.

It also checks how software behaves for any hackers attack and malicious programs and how software is maintained for data security after such a hacker attack.

#### #38) Smoke Testing

Whenever a new build is provided by the development team then the software testing team validates the build and ensures that no major issue exists.

The testing team ensures that the build is stable and a detailed level of testing is carried out further. [Smoke Testing](https://www.softwaretestinghelp.com/smoke-testing-and-sanity-testing-difference/) checks that no show stopper defect exists in the build which will prevent the testing team to test the application in detail.

If testers find that the major critical functionality is broken down at the initial stage itself then testing team can reject the build and inform accordingly to the development team. Smoke Testing is carried out to a detailed level of any functional or regression testing.

#### #39) Static Testing

Static Testing is a type of testing which is executed without any code. The execution is performed on the documentation during the testing phase. It involves reviews, walkthrough, and inspection of the deliverables of the project. Static testing does not execute the code instead of the code syntax, naming conventions are checked.

The [static testing](https://www.softwaretestinghelp.com/static-testing-and-dynamic-testing-difference/) is also applicable for test cases, test plan, design document. It is necessary to perform static testing by the testing team as the defects identified during this type of testing are cost-effective from the project perspective.

#### #40) Stress Testing

This testing is done when a system is stressed beyond its specifications in order to check how and when it fails. This is performed under heavy load like putting large number beyond storage capacity, complex database queries, continuous input to the system or database load.

#### #41) System Testing

Under [System Testing technique](https://www.softwaretestinghelp.com/system-testing/), the entire system is tested as per the requirements. It is a Black-box type testing that is based on overall requirement specifications and covers all the combined parts of a system.

#### #42) Unit Testing

Testing of an individual software component or module is termed as [Unit Testing](https://www.softwaretestinghelp.com/unit-testing/). It is typically done by the programmer and not by testers, as it requires a detailed knowledge of the internal program design and code. It may also require developing test driver modules or test harnesses.

#### #43) Usability Testing

Under [Usability Testing](https://www.softwaretestinghelp.com/usability-testing-guide/), User-friendliness check is done. Application flow is tested to know if a new user can understand the application easily or not, Proper help documented if a user gets stuck at any point. Basically, system navigation is checked in this testing.

#### #44) Vulnerability Testing

The testing which involves identifying of weakness in the software, hardware and the network is known as Vulnerability Testing. Malicious programs, the hacker can take control of the system, if it is vulnerable to such kind of attacks, viruses, and worms.

So it is necessary to check if those systems undergo Vulnerability Testing before production. It may identify critical defects, flaws in the security.

#### #45) Volume Testing

[Volume testing](https://www.softwaretestinghelp.com/what-is-volume-testing/) is a type of non-functional testing performed by the performance testing team.

The software or application undergoes a huge amount of data and Volume Testing checks the system behavior and response time of the application when the system came across such a high volume of data. This high volume of data may impact the system’s performance and speed of the processing time.

#### #46) White Box Testing

[White Box testing](https://www.softwaretestinghelp.com/white-box-testing-techniques-with-example/) is based on the knowledge about the internal logic of an application's code.

It is also known as Glass box Testing. Internal software and code working should be known for performing this type of testing. Under these tests are based on the coverage of code statements, branches, paths, conditions etc.

### Conclusion

The above-mentioned Software Testing Types are just a part of testing. However, there is still a list of more than 100+ types of testing, but all testing types are not used in all types of projects. So I have covered some common Types of Software Testing which are mostly used in the testing life cycle.

Also, there are alternative definitions or processes used in different organizations, but the basic concept is the same everywhere. These testing types, processes, and their implementation methods keep changing as and when the project, requirements, and scope changes.

OneSignal is a high volume and reliable push notification service for websites and mobile applications. We support all major native and mobile platforms by providing dedicated SDKs for each platform, a [RESTful server API](https://documentation.onesignal.com/reference-link/create-notification" \t "_self), and an [online dashboard](https://onesignal.com/" \t "_self) for marketers to design and send push notifications.

# Xcode Workspace

A workspace is an Xcode document that groups projects and other documents so you can work on them together. A workspace can contain any number of Xcode projects, plus any other files you want to include. In addition to organizing all the files in each Xcode project, a workspace provides implicit and explicit relationships among the included projects and their targets.

A workspace is a collection of projects. It's useful to organize your projects when there's correlation between them (e.g.: Project A includes a library, that is provided as a project itself as project B. When you build the workspace project B is compiled and linked in project A).  
It's common to use a workspace in the popular [CocoaPods](http://cocoapods.org/). When you install your pods, they are placed inside a workspace, that holds your project and the pod libraries.

Alongside Swift 3.0, Apple has released its own tool called [Swift Package Manager](https://swift.org/package-manager/" \t "_blank) to share and distribute packages in Swift 3.0 and above. Apple defines it as:

The Swift Package Manager is a tool for managing the distribution of Swift code. It’s integrated with the Swift build system to automate the process of downloading, compiling, and linking dependencies.

While CocoaPods and Swift Package Manager are awesome, there are other options. [Carthage](https://github.com/Carthage/Carthage" \t "_blank) is one such alternative; it’s a *ruthlessly simple* dependency manager for macOS and iOS, created by a group of developers from Github.

It was the first dependency manager to work with Swift; in fact, Carthage itself is *written*in Swift! It exclusively uses dynamic frameworks instead of static libraries – this is the only way to distribute Swift binaries that are supported by iOS 8 and up.

Dependency managers perform a number of handy functions:

* They simplify and standardize the process of fetching third party code and incorporating it into your project. Without such a tool, this might be done by manually copying source code files, dropping in precompiled binaries, or using a mechanism like Git submodules.
* They make it easier to update third party libraries in the future. Imagine having to visit each dependency’s GitHub page, download the source, and place it into your project every time there’s an update. Why would you do that to yourself?
* They pick out appropriate and compatible versions of each dependency you use. For instance, if you’re manually adding dependencies, things can get tricky when they depend on one another or share another dependency.

CocoaPods adds complexity to both the app development and the library distribution processes:

* Libraries must create, update and host Podspec files (or app developers must write their own if one doesn’t exist for a library that they wish to use).
* When adding “pods” to a project, CocoaPods creates a new Xcode project with a target for each individual pod, as well as a containing workspace. Then you have to use the workspace and trust that the CocoaPods project works correctly. Talk about a lot of extra build settings to maintain.
* CocoaPods’ Podspecs repository is centralized, which could be problematic if for some reason it were to disappear or become inaccessible.

The Carthage project’s aim is to provide a **simpler** tool than CocoaPods; one that’s easier to understand, easier to maintain and more flexible.

It achieves this in a number of ways:

* Carthage doesn’t modify your Xcode project or force you to use a workspace.
* There’s no need for Podspecs or a centralized repository for library authors to submit their pods to. If your project can be built as a framework, it can be used with Carthage. It leverages existing information straight from Git and Xcode.
* Carthage doesn’t really do anything magic; you’re always in control. You manually add dependencies to your Xcode project and Carthage fetches and builds them.

**Note**: Carthage uses dynamic frameworks to achieve its simplicity. This means your project must support iOS 8 or later.

There are two ways to install it: downloading and running a **.pkg** installer for the latest release, or using the [Homebrew](http://brew.sh/" \t "_blank) package manager.

For the purposes of this Carthage tutorial, you’ll use the **.pkg** installer. Download the latest release of Carthage from the list [here](https://github.com/Carthage/Carthage/releases" \t "_blank). Select the most recent build, then under **Downloads** select **Carthage.pkg**.

Double-click **Carthage.pkg** to run the installer. Click **Continue**, select a location to install to, click **Continue** again, and finally click **Install**.

And you’re done! To check that Carthage installed correctly, open **Terminal** and run the following command:

carthage version

If all has gone to plan, you’ll see the version number of Carthage that was installed.

**Note**: At the time of writing, the current version of Carthage was 0.23.

## Creating Your First Cartfile

A **Cartfile** is a simple text file that describes your project’s dependencies to Carthage, so *it* can determine what to install. Each line in a Cartfile states where to fetch a dependency from, the name of the dependency, and optionally, which version of the dependency to use. A Cartfile is the equivalent of a CocoaPods **Podfile**.

Navigate to the root directory of your project in Terminal (the directory that contains your **.xcodeproj** file) using the **cd** command:

cd ~/Path/To/Starter/Project

Create an empty Cartfile with the touch command:

touch Cartfile

And then open the file up in Xcode for editing:

open -a Xcode Cartfile

If you’re familiar with another text editor, like Vim, then feel free to use that instead. Don’t, however, use TextEdit to edit the file; with TextEdit it’s too easy to accidentally use so-called “smart quotes” instead of straight quotes, and they will confuse Carthage.

Add the following lines to the Cartfile and save it:

github "Alamofire/Alamofire" == 4.5 github "Alamofire/AlamofireImage" ~> 3.2

These two lines tell Carthage that your project requires Alamofire version 4.5, and the latest version of AlamofireImage that’s compatible with version 3.2.

CTAssetsPickerController is a highly customisable iOS controller that allows picking multiple photos and videos from user's photo library. The usage and look-and-feel are just similar to UIImagePickerController. It uses **ARC** and requires **Photos** framework.

A primitive data type is one that fits the base architecture of the underlying computer such as int, float, and pointer, and all of the variations, thereof such as char short long unsigned float double and etc, are primitive data type.

Primitive data are only single values, they have not special capabilities.

The examples of Primitive data types are given byte, short, int, long, float, double, char etc.

The integer reals, logic data character data pointer and reference are primitive data structures data structure that normally are directly operated upon by machine level instructions are known as primitive structure and data type.

Non- Primitive Data type,

A non-primitive data type is something else such as an array structure or class is known as the non-primitive data type.

The data type that are derived from primary data types are known as non-primitive data type.

The non-primitive data types are used to store the group of values.

Examples of non-primitive data type.

Array, structure, union, link list, stacks, queue etc.

संक्षेप

abbreviated, abstract, compendium, Curtailment, Epitome, outline

<https://www.geeksforgeeks.org/top-10-algorithms-in-interview-questions/>

A split view controller is a container view controller that manages two child view controllers in a master-detail interface. In this type of interface, changes in the primary view controller (the master) drive changes in a secondary view controller (the detail). The two view controllers can be arranged so that they are side-by-side, so that only one at a time is visible, or so that one only partially hides the other. In iOS 8 and later, you can use the UISplitViewControllerclass on all iOS devices; in previous versions of iOS, the class is available only on iPad.

# NSInternalInconsistencyException

Name of an exception that occurs when an internal assertion fails and implies an unexpected condition within the called code.

## Compound Assignment Operators

Like C, Swift provides compound assignment operators that combine assignment (=) with another operation. One example is the addition assignment operator (+=):

1. var a = 1
2. a += 2
3. // a is now equal to 3

The expression a += 2 is shorthand for a = a + 2. Effectively, the addition and the assignment are combined into one operator that performs both tasks at the same time.

NOTE

The compound assignment operators don’t return a value. For example, you can’t write let b = a += 2.

In Swift self is a special property of an instance that holds the instance itself.

self and super actually both point to the same object. super is a keyword that tells the compiler to generate instructions that start the search for a method definition in the super class rather than in the current class.

@interface A : NSObject {} - (void)foo; @end @implementation A - (void)foo { NSLog(@"A's foo!"); } @end @interface B : A @end @implementation B - (void)foo { NSLog(@"B's foo!"); } @end //...somewhere in a method of class B... [self foo]; // prints "B's foo" in the console [super foo]; // prints "A's foo" in the console

A subclass can provide its own custom implementation of an instance method, type method, instance property, type property, or subscript that it would otherwise inherit from a superclass. This is known as *overriding*.

# Three ways to pass data from Model to Controller

I will describe three basic ways of passing the data back to your Controller:

1. Using Callbacks
2. Using Delegation
3. Using Notifications

<https://medium.com/@stasost/ios-three-ways-to-pass-data-from-model-to-controller-b47cc72a4336>

You can define Swift enumerations to store associated values of any given type, and the value types can be different for each case of the enumeration if needed. Enumerations similar to these are known as *discriminated unions*, *tagged unions*, or *variants* in other programming languages.

For example, suppose an inventory tracking system needs to track products by two different types of barcode. Some products are labeled with 1D barcodes in UPC format, which uses the numbers 0 to 9. Each barcode has a number system digit, followed by five manufacturer code digits and five product code digits. These are followed by a check digit to verify that the code has been scanned correctly:

In [computer science](https://en.wikipedia.org/wiki/Computer_science), a **tagged union**, also called a [**variant**](https://en.wikipedia.org/wiki/Variant_type), **variant record**, **choice type**, **discriminated union**, [**disjoint union**](https://en.wikipedia.org/wiki/Disjoint_union), **sum type** or [**coproduct**](https://en.wikipedia.org/wiki/Coproduct), is a [data structure](https://en.wikipedia.org/wiki/Data_structure)used to hold a value that could take on several different, but fixed, types. Only one of the types can be in use at any one time, and a **tag** field explicitly indicates which one is in use. It can be thought of as a type that has several "cases," each of which should be handled correctly when that type is manipulated. Like ordinary [unions](https://en.wikipedia.org/wiki/Union_(computer_science)), tagged unions can save storage by overlapping storage areas for each type, since only one is in use at a time.

In [computer graphics](https://en.wikipedia.org/wiki/Computer_graphics), a **color gradient** (sometimes called a **color ramp** or **color progression**) specifies a range of position-dependent [colors](https://en.wikipedia.org/wiki/Color), usually used to fill a region.[[1]](https://en.wikipedia.org/wiki/Color_gradient#cite_note-1) For example, many [window managers](https://en.wikipedia.org/wiki/Window_managers) allow the screen background to be specified as a gradient. The colors produced by a gradient vary continuously with position, producing smooth color transitions.

Font Awesome is a font that's made up of symbols, icons, or pictograms (whatever you prefer to call them) that you can use in a webpage, just like a font.

1. Download the [FontAwesome](https://github.com/FortAwesome/Font-Awesome) font.
2. Add that font files into your project using Xcode as a resource. Make sure it is part of the "Copy Bundle Resources" in the Build Phases.
3. Add a key to your Info.plist file called UIAppFonts. (Friendly name is: Fonts provided by application)
4. Make sure this key an array.
5. Enter the full name of your font file (including the extension) as an item to the UIAppFonts array. (Usually fontawesome-webfont.ttf)
6. Save your Info.plist.
7. Now in your application you can simply call [UIFont fontWithName:@"FontAwesome" size:12] to get the custom font to use with your UILabels and UITextViews, etc…

The benefits of using font-awesome for icons is that you can treat them just like text. It eliminates the need of using pngs of varied sizes or varied colors of a same icon as you can just set the font size or font color to induce the effect of your desire.

<https://medium.com/@umairhassanbaig/ios-how-to-use-font-awesome-in-xcode-project-b8ef255973a3>