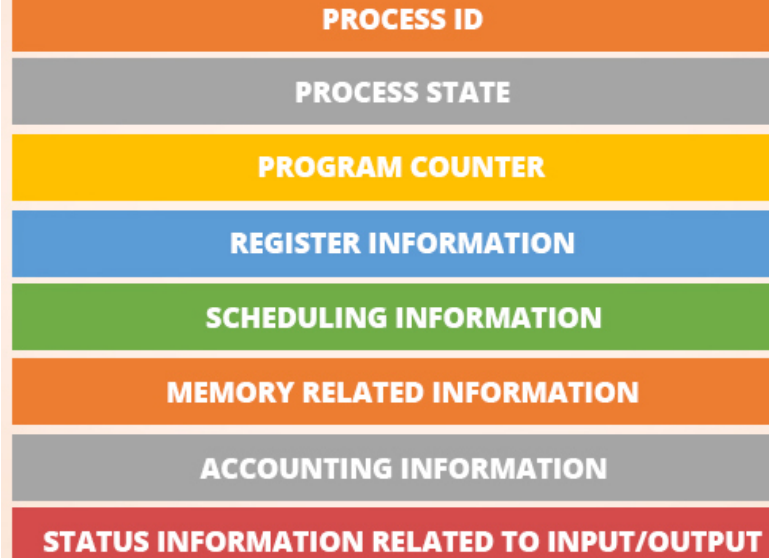


Accessibility	
What is voiceOver	VoiceOver is a gesture-based screen reader that enables people to experience the interface on their devices without having to see the screen. People who are blind depend on VoiceOver to provide auditory feedback while using their iOS devices
Dynamic Type	Dynamic Type is a way of allowing the user to adjust their preferred size for all fonts in all apps
Explain what Swift programming language is?	Swift is a programming language for creating applications for iOS and OS X. Swift has safe programming patterns and more additional features that make programming easier and more flexible. Swift is friendly to beginners. It allows programmers to experiment with Swift code and get the results immediately. It is an advanced programming language for Cocoa and Cocoa Touch. Download Xcode and use resources to build applications for all Apple platforms.
What do you mean by overlays in OS?	Overlays is a programming method that divides processes into components to save instructions in memory. It does not require any kind of support from OS. It can execute programs that are bigger than physical memory.
State the difference between an array and a set?	• Arrays are an ordered collection of values. At the same time, sets are an unordered collection of specific values.
	• Each element in a set can only appear one time. At the same time, an array can duplicate elements.
What are tuples, and why are they useful in Swift?	In Swift, a tuple is a group of multiple values. Tuples occupy the space between dictionaries and structures. They are commonly used to return different values from a function call.
What is the purpose of NotificationCenter?	it deals with notifications. Most people use this to receive system messages, but you can also send your messages inside your application.
What are conditional conformances in Swift?	If you are an experienced iOS developer, you should know that protocol conformances permit us to conform a protocol only when certain conditions are fulfilled – hence “conditional conformance.”
What is the purpose of the ButtonStyle protocol in Swift?	ButtonStyle Protocol allows us to customize new button styles that can reuse without introducing new views or copy-pasting the styling code to get constant designs.
. What is the purpose of IBDesignable?	Marking a custom view as IBDesignable allows Xcode to preview it when editing storyboards. It's useful when you create your custom view subclasses and want to see them rendered live in IB, particularly when you want to set their properties there too.
What type of settings would you store in your Info. plist file?	The Info. list file stores settings that must be available even when the App isn't running. You could talk about custom URLs, privacy permission messages, custom fonts, whether the App deactivates background running, and so on.

What is the purpose of raw strings in Swift?	Raw Strings (by placing a hash before and after your string quotes) allow us to create strings that will print what you see. As you know, certain escape orders can make our strings print differently.
How can an observable object announce modifications to SwiftUI?	Two primary ways are using the @Published property wrapper or calling object.willChange.send() directly.
What are the advantages of using child view controllers?	A view controller gains access to events like viewDidLoad and viewWillAppear, even when used as a child, which can be useful for many types of UI code. It's notable how child view controllers can also make our code reusable because the child can be used in other parent view controllers.
Are closures value or reference types?	Closures are reference types. If you allocate a closure to a variable and copy the variable into another variable, you also copy a reference to the same closure and its capture list.
. Can you explain a circular reference in Swift?	A circular reference happens when two objects hold a strong reference to each other and can be a reason for memory leak because neither of the two objects will ever be deallocated. You cannot deallocate the object as long as there's a strong reference to it, but each keeps the other object alive because of its strong reference.
How can you add a stored property to a type by using an extension?	No, it is not possible. You can use an extension while adding a new behavior to an existing type but cannot change the type itself or its inheritance. If you add a stored property, you need extra memory to store the new value. An extension cannot manage such a task.
What are the Half Open Range Operators in Swift?	Swift supports different types of operators. One of them is the Half Open Range operator. This operator specifies a range between values a and b ($a < b$) where b is not included. It is a half-open range operator because it consists of its first value, not only the final value.
What is Nested Function in Swift?	A function inside a function is called a nested function.
Define Regular Expression?	Regular expressions are the special string patterns that specify how to search for a string.
Define Responder Chain?	Responder Chain is the order of objects that get the opportunity to respond to the events.
Write names of different types of literal does Swift language have?	• Binary Literals • Octal Literals • Hexadecimal Literals • Decimal Literals

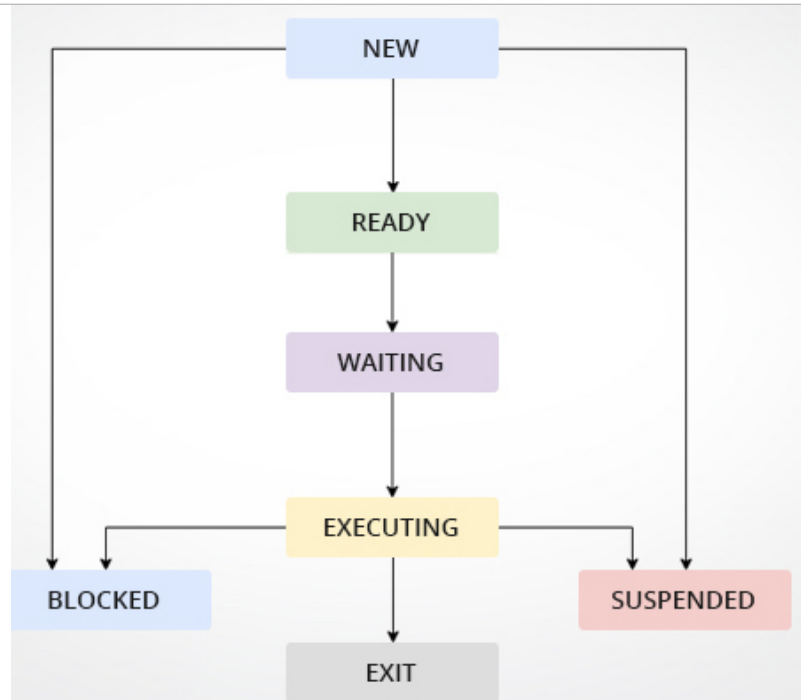
What is processor management?	Processor management provides tools and resources for analyzing, defining, optimizing, and controlling business processes for better performance.
	PROCESSOR MANAGEMENT JOB SCHEDULING ↓ PROCESS SCHEDULING ↓ INTERRUPT MANAGEMENT
State the different control transfer statements used in Swift.	- Continue - Break - Fallthrough - Return
What do you mean by Optional Chaining in Swift?	Optional Chaining is a process of querying and calling properties. You can chain different queries together, but if any link in the chain is nil, the whole chain fails.
How do you explain inheritance in Swift?	Inheritance is a process in which a class can inherit properties, methods, and other features from another class. It is supported in Swift Programming language. There are two types of classes in inheritance in Swift: -subclass. -super class
What is Process management in iOS?	Each process in iOS is made up of one or more threads. Each thread acts as a single path of execution. Each application in iOS starts with a single thread, which runs the application's main functions.

Write names of components of Process Control Block?

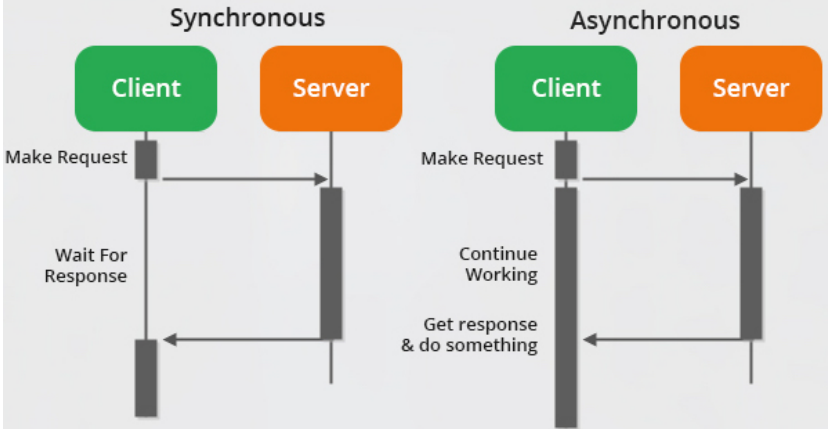


State different Process States in OS.

- **New:** the process is being created.
- **Ready:** The process is waiting to be assigned to a processor.
- **Running:** Commands are being executed
- **Waiting:** Process is waiting for some event to occur.
- **Terminated:** The process has completed execution.



Define Core Data?	Core Data is an object graph management framework that controls a potentially very large graph of object instances. It allows an app to work with a graph that would not fit into memory by faulting instances in and out of memory as required. Core Data also contains constraints on properties and relationships and maintains reference integrity (e.g., keeping forward and backward links consistent when objects are added/removed to/from a relationship). Core Data is thus an ideal framework for developing the “model” component of an MVC architecture. To implement its graph management, Core Data uses SQLite as a disk store. It could have been implemented using a relational database or even a non-relational database such as CouchDB. Core Data is not a database engine as an API abstracts over the actual data store. Core Data can save as an SQLite database, a list, a binary file, or even a custom data store type.
Explain what NSUserDefaults is? NSUserDefaults support Which types?	NSUserDefaults is the easiest way to store data without a database using key-value pair. NSUserDefaults stores a small amount of data. In most cases, NSUserDefaults is best used to save users’ settings and data that is not critical. Following are types supported by NSUserDefaults: - NSString - NSNumber - NSDate - NSDictionary - NSData
Pros of NSUserDefaults:	• Relatively easy to store and retrieve data • Perfect for keeping small size data (example: User’s Settings) • Easy to learn and implement
Cons of NSUserDefaults	• Not suitable to collect a large amount of data • The performance will slow down when storing a large amount of data • Not ideal for storing sensitive data
What are synchronous and asynchronous tasks in iOS?	If you are an experienced iOS developer, you must know synchronous and asynchronous tasks in iOS. When you perform the synchronous tasks, you have to wait for the task to complete before proceeding. On the other hand, when you perform tasks asynchronously, you don’t have to wait to complete tasks. You can perform the tasks simultaneously. When tasks are completed in the background, will notify you.

	 The diagram illustrates two communication models between a Client (green box) and a Server (orange box). In the Synchronous model, the Client sends a 'Make Request' message to the Server, then enters a 'Wait For Response' state until the Server sends back a response. In the Asynchronous model, the Client sends a 'Make Request' message to the Server and then immediately enters a 'Continue Working' state. The Server processes the request and sends a response back to the Client, who then performs 'Get response & do something'.
What do you mean by lazy stored properties, and why is it useful?	Lazy stored properties are used for a property whose first values are not calculated until the first time it is used. You can claim a lazy stored property by reporting the lazy modifier before its declaration. Lazy properties are beneficial when the initial value relies on outside factors whose values are unknown.
What is the purpose of Size Classes?	Size classes allow you to add extra layout configuration to your App so that your UI works well across multiple devices. For example, you can say that a stack view lines up its views in normal conditions horizontally but vertically when constrained.
Which JSON framework is supported by iOS?	OS supports the SBJson framework. SBJson is a JSON syntax analyzer (parser) and generator for Objective-C. It is making JSON handling easier by providing flexible APIs and additional control.
Explain the difference between atomic and nonatomic properties?	Atomic properties are thread-safe (something is shared across different threads without issues like a crash), but it is slow. Nonatomic properties mean different threads access the variable (dynamic type). Non-atomic means thread-unsafe, but it is fast.

What are the different kinds of iOS Application States?

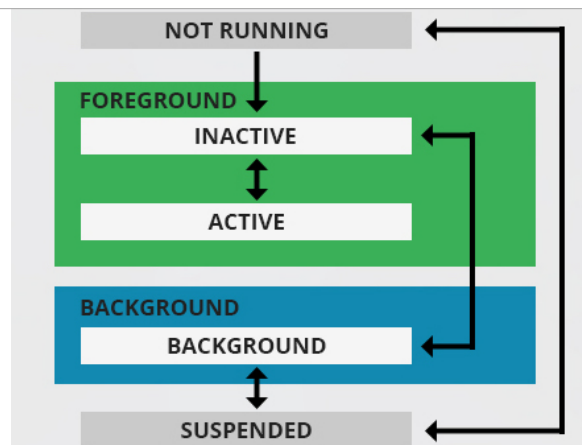
Not running state: When the App has not been launched or was running but was ended by the system.

Inactive state: The App is running in the foreground but currently not receiving any events. It remains inactive only when the user locks the screen or prompts the user to respond to an event such as an SMS message or a phone call.

Active State: When the App is running in the foreground. It is also receiving events.

Background state: When the App is in the background state, it executes code. Most applications enter this state briefly to being suspended. However, an app that requests extra execution time can stay in this state for some time. Also, an app directly launched into the background enters this state instead of the inactive state.

Suspended state: A suspended app remains in memory and does not execute any code.



How many ways to achieve concurrency in are iOS are?

- Threads
- Dispatch Queues
- Operation Queues

Explain the difference between 'assign' and 'retain' keywords?	Assign' keywords specify that the setter(sets or updates values) uses simple assignment. It is used in attributes of scalar type like float, int. 'Retain' keywords specify that you should call retain on the object upon assignment. It takes ownership of an object.								
Define layer objects?	Layer objects are data objects that manage visual content and use views to display their content. Custom layer objects provide options for modifying complex animations and other types of the visual appearance of that content.								
What is an autorelease pool in Swift?	An autorelease pool stores objects that send releases to all objects when the pool itself is drained. If you utilize Automatic Reference Counting (ARC), you cannot utilize the autorelease pool directly. You utilize @autoreleasepool blocks.								
Kindly explain the class hierarchy for a UIButton until NSObject?	UIButton inherits from UIControl, UIControl inherits from UIView, UIView inherits from UIResponder, UIResponder inherits from the root class NSObject.								
Differentiate 'app ID' from 'bundle ID'. Why are they used?	An App ID is used to identify one or more applications from a single development team. It consists of a Team ID and a bundle ID search string separating the two parts with a period (.). The Team ID is supplied by Apple and is particular to a specific development team, while the developer supplies the bundle ID search string. Whereas the bundle ID is defined by each App and is specified in Xcode. A single Xcode project has different targets and therefore outputs multiple apps.								
State the difference between Cocoa and Cocoa Touch?	<table> <tr> <th>COCOA</th><th>COCOA TOUCH</th></tr> <tr> <td>It is an application framework for building applications that run on Mac OS.</td><td>It is an application framework for building apps that run on devices like iPad & iPhones.</td></tr> <tr> <td>Combination of frameworks such as Foundation and AppKit</td><td>Combination of frameworks such as Foundation and UIKit.</td></tr> <tr> <td>Cocoa classes use the NS (used for all classes & constants in cocoa frameworks) prefix like NSWindow, NSTextField.</td><td>Cocoa Touch classes use the UI (used for all classes and constants in cocoa touch framework) prefix like UITextField, UIWindow.</td></tr> </table>	COCOA	COCOA TOUCH	It is an application framework for building applications that run on Mac OS.	It is an application framework for building apps that run on devices like iPad & iPhones.	Combination of frameworks such as Foundation and AppKit	Combination of frameworks such as Foundation and UIKit.	Cocoa classes use the NS (used for all classes & constants in cocoa frameworks) prefix like NSWindow, NSTextField.	Cocoa Touch classes use the UI (used for all classes and constants in cocoa touch framework) prefix like UITextField, UIWindow.
COCOA	COCOA TOUCH								
It is an application framework for building applications that run on Mac OS.	It is an application framework for building apps that run on devices like iPad & iPhones.								
Combination of frameworks such as Foundation and AppKit	Combination of frameworks such as Foundation and UIKit.								
Cocoa classes use the NS (used for all classes & constants in cocoa frameworks) prefix like NSWindow, NSTextField.	Cocoa Touch classes use the UI (used for all classes and constants in cocoa touch framework) prefix like UITextField, UIWindow.								
Which programming languages are used for iOS development?	• HTML5 • .net • C • C++ • Swift • JavaScript • Objective-C								

What is Automatic Reference Counting (ARC)?	In Swift programming language, automatic reference counting is used to manage apps' memory usage. It initializes and deinitializes system resources. ARC keeps track of how many properties, constants, and variables refer to each class instance. When there is at least one active reference to any instance, ARC will not deallocate them. Its use is an essential part of iOS development. Functions of ARC: • ARC creates a new class instance using <code>init()</code> and allocates a piece of memory to store the information. • Memory caches information about the instance type and its values. • As soon as the class instance is no longer needed, ARC automatically frees memory by calling <code>deinit()</code>. • By tracking current referring classes' properties, constants, and variables, ARC makes sure that <code>deinit()</code> is only involved in instances not used.
Explain what is Grand Central Dispatch (GCD) in iOS?	Grand Central Dispatch (GCD) is a low-level API that allows users to run concurrent tasks (run simultaneously). It also manages threads in the background. Apple's solution is to build concurrency and parallelism into iOS applications so different tasks can run concurrently in the background without affecting the main thread. GCD was introduced in iOS 4 to avoid the tedious serial execution of tasks.
What do you mean by Deep linking in iOS?	Deep links are those links that send users directly to an app instead of a website or store using uniform resource locator (URL) or universal links. The URL scheme is a well-known method of having deep links, but Universal Links are Apple's new approach to connect your web page and your App under the same link. Deep linking involves creating a clickable link that opens up your App and a smart one that navigates to the resource you require. Users are headed straight to in-app locations using these links, which saves them the time and effort of seeing those pages themselves, thus enhancing their user experience tremendously.

Which framework is used to build an application's interface for iOS?	he foundation framework defines classes, protocols, and functions for iOS and OS X development; UIKit is specifically designed for iOS development. The application's user interface and graphical infrastructure are developed using UIKit in iOS. It includes: • App Structure(manages the interaction between system and user) • Event Handling (handles different gestures like input gestures, multi-touch gestures, etc.) • User Interface (provides user interactions, sharing text and content, choosing images, editing videos, print files, etc.) • Graphics, Drawing, and Printing
Explain Objective-C in OS.	Apple has used Objective-C as an object-oriented programming language since the 1990s. This language compromises the benefits of two earlier languages – C and Smalltalk. Features: • In Objective-C, metaclasses are automatically created and managed during runtime. • Both typing is supported; dynamic typing and static typing. • It is easy to understand.
Name the most important data types in Objective C?	Following are the important data types that most developers use in Objective C. BOOL: It represents a Boolean value that is neither true nor false. Both the NSInteger: Represents an integer NSUInteger: Represents an unsigned integer. NSString: Represents a string

What are the important features of Swift?	Swift programming language is being developed to write correct codes and maintain them easily. It offers the following features: Safety: Swift is an efficient way to write code. Checking code before execution is very important. It removes an unsafe code before it is used in production. Simple Syntax: Its syntax is easy to use, learn and understand. Its features make able you to write more expressive code. Readability: Swift has a simple syntax, easier to read and write. It is easier for developers to write swift code since it is more like plain English, allowing them to spend less time. Multiplatform Support: Swift is fully supported with iOS, macOS, tvOS, watchOS, Linux, and many other platforms. It means you can develop software that is compatible with all operating systems. Open-Source: Swift is an open-source framework developed at swift.org. Swift supports all Apple platforms and makes programming easier, safer and faster. Compatible with Objective C: Swift programming language has full compatibility with Objective-C. It allows developers to import frameworks from Objective-C using Swift syntax. Developers can use Objective-C libraries and classes inside swift code.
Explain Dictionary in Swift.	Swift programming describes a dictionary as an unordered collection of items. It stores items in key-value pairs. A dictionary uses a specific identifier called a key to store an associated value which can be referenced and recovered through the same key.

Explain the difference between KVC and KVO in Swift.	KVC (Key-value-coding) It allows object properties to be accessed at runtime using strings rather than knowing the property names statically during development. KVO (Key-Value Observing) KVO is one of the methods for observing program state changes in Objective- C and Swift. If an object has instance variables, KVO makes other objects observe the modifications or changes to those variables.
Define the function of the completion handler.	Completion handlers are just functions passed as parameters to other functions. They are used to dealing with asynchronous tasks since we do not know when they will end. Completion handlers tell an application when an operation, such as an API call, has been completed. The program is notified that the next step needs to be completed.
State the difference between strong, weak, read-only, and copy.	Strong: This PropertyProperty maintains a reference to PropertyProperty throughout the lifetime of an object. When you declare strong, you expect to “own” the object you are referencing. Data you assign to this PropertyProperty will not be destroyed as long as you or any other object references it strongly. Weak: It means that you should keep the object in memory as long as someone points to it strongly, and you don’t require control over its lifetime. Read-only: Initially, an object’s PropertyProperty can be defined, but it cannot change or alter. Copy: This attribute is an alternative to a strong attribute. To take ownership of a current object, it creates a copy of whatever you assign to the PropertyProperty, then takes ownership of that copy.

Explain Test-Driven Development (TDD).	Test-Driven development is used in the development of software. In TDD, developers plan the software characteristics that they want to create and then write test cases for each characteristic before implementing it. Through test-driven development, we can get insight into both the quality of the implementation (does it work) and the design quality (is it well structured). In the first case, the test case will fail because the code is not yet implemented, and this is usually referred to as the red phase. After that, code is written to ensure that the test case passes and does not break any components or current test cases; that phase is called the green phase. The developer should then refactor the implementation of the code by maintaining and cleaning the codebase and optimizing the efficiency. Repeat this process every time a new test case is added.
How can you execute storage and persistence in iOS?	Persistence means storing data on the disk to be retrieved without being modified the next time the App is opened. From simple to complex, there are the following methods for storing data: • Data structures such as arrays, dictionaries, sets, and other data structures are real or perfect for storing data intermediately. • UserDefaults and Keychains are both simple key-value stores. UserDefaults is unsafe, whereas Keychains is safe. • A file or disk storage is a way to store data to or from a disk using NSFileManager. • Relational databases, such as SQLite, are good for implementing complex querying structures.
How do you explain generics in Swift and write its usage?	An important feature of Swift is generics, and much of the Swift standard library is written in generic code. Swift's 'Array' and 'Dictionary' types, for example, constitute generic collections. Generic code lets you create flexible, reusable functions and types that operate with any data type. You can create code that does not get too specific about underlying data types, resulting in cleaner code. Generics are used to avoid repetition and to provide extraction.
What is Enumerations or Enum in Swift?	The term enumerations refers to a user-defined data type consisting of a set of related values that enable you to work with those values in your code in a type-safe manner. The keyword enum defines this data type.
Write the two different smart groups in Xcode?	Smart Groups are divided into two parts: • Simple Expression Smart Group • Simple filter smart group

Explain what TVMLKit is.	There exists a link between JavaScript, TVML, tvOS apps, which combines and is known as TVMLKit.
What is Code Coverage?	Code Coverage is used to estimate the value of our unit tests. Its measurement specifies which statements in a body of code have been executed and which statements have not been executed through a test run. Unit tests assist in making certain functionality and provide a mechanism of verification for refactoring efforts.
Define Operator Overloading?	Operator overloading can work on how existing operators perform with existing types. To overload an operator, we use a unique operator function. Operators are those symbols like +, /, and *.
Mention the Realm benefits?	• A realm is a security policy domain defined for an application server or a web. • Realm is an open-source database framework. • It is implemented from scratch. • It is faster and faster than core data. • All documents related to Realm are written properly, and experts can find an answer on the official website if required.
Write down the names for battery-efficient location tracking?	It is one of the most commonly asked iOS interview questions. Significant location changes: The location is given approximately every 500 meters (usually up to 1 km) Region Monitoring: It tracks all enter/exits possibilities from circular regions with a radius equal to 100m or more. It is the most accurate API after GPS (Global Positioning System) Visit Events: Monitoring place Visit Events entering/exiting from a place (home/office).
What is the most efficient way to cache data in memory?	There are many ways of making caches with a simple dictionary, but whatever you select, you should be prepared to explain your choice why you like it. Ensure and consider how you remove data from the cache, either explicitly or to hit a memory quota. If you feel comfortable talking about it, NSCache is preferable over a simple dictionary because it automatically clears the cache when memory is low.

Explain the architecture of iOS.	Core OS (or Application) Layer: Core OS Layer models directly on top of the device hardware and is the bottom layer of the iPhone OS stack. In addition to basic operating system services, such as memory management, handling of file systems, and threads, this layer delivers low-level networking, access to external accessories, etc. Service Layer: It aims to design the services that upper layers or users require. Its other essential features are blocked objects, Grand Central Dispatch, in-app purchases, and iCloud storage. ARC Automatic Reference Counting has supported the service layer. Media Layer: It manages media like video, audio, graphics, etc. The media layer will permit us to use all graphics, video, and audio technology of the system. Cocoa Touch Layer: It is also called the application layer. It is the place where frameworks are made when applications are built. In addition, it works as the interface for iOS users to operate with the operating system. It contains touch and motion capabilities.
For what purpose are computed properties used?	Computed Properties are used to calculate the values instead of storing them. These are usually provided by the classes, structures, and enumerations.
Explain NSError in Swift.	Information about an error condition is summarized within an NSError object flexibly and object-oriented manner. An NSError consists of three basic attributes: a predefined error domain, a domain-specific error code, and a user info dictionary comprising application-specific information.
What do you mean by dynamic dispatch?	In simple words, dynamic dispatch means that the program decides at runtime which implementation of a specific method or function it requires to call on. When a subclass overrides a method of its subclasses, dynamic dispatch defines whether to call the subclass implementation of the method or the parents.
. What is a Pipe, and when is it utilized?	A pipe is a connection between two or more processors that are interrelated to each other. It is a mechanism used for inter-process communication using message passing. One can easily send information such as the result or output of one program process to another using a pipe. It can be used when both processes want to communicate one-way, i.e., inter-process communication (IPC).

State the difference between Structure And Class?	<table><tr><th>CLASS</th><th>VS</th><th>STRUCTURE</th></tr><tr><td>Reference Types</td><td></td><td>Value Types</td></tr><tr><td>Classes can be built on other classes</td><td></td><td>Structure cannot inherit from another Structure</td></tr><tr><td>Classes have an inheritance</td><td></td><td>Structure cannot have an inheritance</td></tr><tr><td>Classes have Type Casting</td><td></td><td>Structure doesn't have type casting</td></tr></table>	CLASS	VS	STRUCTURE	Reference Types		Value Types	Classes can be built on other classes		Structure cannot inherit from another Structure	Classes have an inheritance		Structure cannot have an inheritance	Classes have Type Casting		Structure doesn't have type casting
CLASS	VS	STRUCTURE														
Reference Types		Value Types														
Classes can be built on other classes		Structure cannot inherit from another Structure														
Classes have an inheritance		Structure cannot have an inheritance														
Classes have Type Casting		Structure doesn't have type casting														
What are the different kinds of operations that are possible on semaphore?	• Wait() • SIGNAL()															
What do you mean by RTOS?	Real-Time Operating System (RTOS) is an operating system used for real-time applications, i.e., for those applications where data processing should be done in a fixed and small amount of time. It performs very well on tasks that must execute within a short time. It also manages execution tasks, monitoring, and all-controlling process. It also occupies less memory and consumes few resources.															
What do you mean by process synchronization?	Process synchronization is a way to interrelate processes that use shared resources or data. It is very much important to make sure synchronized execution of cooperating processes. It will maintain data consistency. It shares resources without any interference using mutual exclusion. There are two types of process synchronization: • Independent Process • Cooperative Process															
Write different names of IPC mechanisms?	Pipes • Messaging Queuing • Semaphores • Socket • Shared Memory • Signals															

What is the difference between Stack And Heap?	Stack is easy to use as compared to Heap. Heap is quite slow. Stack's kept in RAM on the computer while Heap creates memory problems. Stack is quite fast while Heap is created at runtime.
What skills are required to become an iOS developer?	• Swift Programming Language • iOS Platform • C-based Libraries • APIs and Cloud Messaging • Design Guidelines • Spatial Reasoning • UI and UX Designing • Core Data • Grand Central Dispatch (GCD) • Networking
What does the Codable protocol do?	This protocol was introduced in Swift 4 to let us quickly and safely convert custom Swift types to and from JSON, XML, and similar. For bonus points talk about customization points such as key and date decoding strategies, the CodingKey protocol, and more, so that you're able to show you can handle a range of input and output styles.
What is the difference between the Float, Double, and CGFloat data types?	Suggested approach: It's a question of how many bits are used to store data: Float is always 32-bit, Double is always 64-bit, and CGFloat is either 32-bit or 64-bit depending on the device it runs on, but realistically it's just 64-bit all the time. For bonus points, talk about how Swift 5.5 and onwards allows us to use CGFloat and Double interchangeably.
What's the importance of key decoding strategies when using Codable?	Suggested approach: Give a specific answer first – “key decoding strategies let us handle difference between JSON keys and property names in our Decodable struct” – then provide some kind of practical sample. For example, you might say that it's common for JSON keys to use snake_case for key names, whereas in Swift we prefer camelCase, so we need to use a key decoding strategy to convert between the two.
When using arrays, what's the difference between map() and compactMap()?	Suggested approach: Remember to give practical examples as well as outlining the core differences. So, you might start by saying the map() transforms a sequence using a function we specify, whereas compactMap() does that same step but then unwraps its optionals and discards any nil values. For example, converting an array of strings into integers works better with compactMap(), because creating an Int from a String is failable.

Why is immutability important?	Suggested approach: Immutability is baked deep into Swift, and Xcode even warns if var was used when let was possible. It's important because it's like a programming contract: we're saying This Thing Should Not Change, so if we <i>try</i> to change it the compiler will refuse.
What does it mean when we say "strings are collections in Swift"?	Suggested approach: This statement means that Swift's String type conform to the Collection protocol, which allows us to loop over characters, count how long the string is, map the characters, select random characters, and more. For bonus points, move on to talk about the Collection protocol itself – how it means we have a consistent way to work with strings, arrays, sets, and more.
What is a UUID, and when might you use it?	Suggested approach: UUID stands for "universally unique identifier", which is a long string of hexadecimal numbers stored in a single type. UUIDs are helpful for ensuring some value is guaranteed to be unique, for example you might need a unique filename when saving something. For bonus points, perhaps explain why we call them universally unique – if you created 100 trillion UUIDs there's a one in a billion chance of generating a duplicate
What's the difference between a value type and a reference type?	Suggested approach: The best way to frame this discussion is likely to be classes vs structs: an instance of a class can have multiple owners, but an instance of a struct cannot. For bonus points mention that closures are also reference types, and the implications of that.
When would you use Swift's Result type?	Suggested approach: Start with a brief introduction to what Result does, saying that it's an enum encapsulating success and failure, both with associated values so you can attach extra information. I would then dive into the "when would you use it" part of the question – talking about asynchronous code is your best bet, particularly in comparison to how things like URLSession would often pass both a value and an error even when only one should exist at a time. If you'd like to go into more detail, more benefits of Result include being able to send the result of a function around as value to be handled at a later date, and also the ability to handle typed errors.

What is type erasure and when would you use it?	Suggested approach: Type erasure allows us to throw away some type information, for example to say that an array of strings is actually just AnySequence – it's a sequence containing strings, but we don't know exactly what kind of sequence. This is particularly useful when types are long and complex, which is often the case with Combine. So, rather than having a return type that is 500 characters long, we can just say AnyPublisher<SomeType, Never> – it's a publisher that will provide SomeType and never throw an error, but we don't care exactly what publisher it is.
Can you explain MVC, and how it's used on Apple's platforms?	Suggested approach: MVC is an approach that advocates separating data (model) from presentation (view), with the two parts being managed by separate logic (a controller). In theory this separation should be as clear as possible, but for bonus points you should talk about how view controllers sometimes get bloated as code gets merged together into one big blob.
Can you explain MVVM, and how it might be used on Apple's platforms?	Suggested approach: Start with the simple definition of Model (your data), View (your layout), and View Model (a way to store the state of your application independently from your UI), but make sure you give some time over to the slightly more nebulous parts – where does networking code go, for example? This is also a good place to bring up the importance of bindings to avoid lots of boilerplate, and that probably leads to SwiftUI.
How would you explain dependency injection to a junior developer?	Suggested approach: Dependency injection is the practice of creating an object and tell it what data it should work with, rather than letting that object query its environment to find that data for itself. Although this goes against the OOP principle of encapsulation, it's worth talking about the advantages – it allows for mocking data when testing, for example.
How would you explain protocol-oriented programming to a new Swift developer?	Suggested approach: POP is a Swift buzzword, but don't get carried away with the hype here: focus on why it's different from OOP, and what benefits you think it has. You might want talk about horizontal vs vertical architectures here – larger codebases are likely to have sizable class hierarchies – but you could also talk about how POP is able to work with structs and enums as well as classes.

What experience do you have of functional programming?	Suggested approach: The best answer of course is to provide detailed explanations of what you've used and where, but as you go make sure and talk about what functional programming means – functions must be first-class types, you place an emphasis on pure functions, and so on. If you're not sure where to start, the easiest answer is to list some small specifics: if you've used map(), compactMap(), flatMap(), reduce(), filter() and so on, that's a good place to begin.
Can you explain KVO, and how it's used on Apple's platforms?	Suggested approach: KVO used to be helpful in UIKit to watch for changes on values that don't have useful delegates – you can literally say "tell me when this value changes." Try to give at least one specific example, such as watching the page load progress on a WKWebView. If you're exclusively using SwiftUI chances are you'll struggle here.
Can you give some examples of where singletons might be a good idea?	Suggested approach: It's very unlikely you'll join a company where singletons are used extensively, so feel free to say that broadly speaking singletons aren't great. Once you've given up that proviso, perhaps mention that Apple uses them extensively – thinks like UIApplication, for example, are designed to exist only once. Finally, try to give a fresh example of your own, such as creating an app-wide logger. For bonus points, perhaps compare and contrast SwiftUI's environment.
What are phantom types and when would you use them?	Suggested approach: Phantom types are a type that doesn't use at least one its generic parameters – they are declared with a generic parameter that isn't used in their properties or methods. Even though we don't use the generic type parameter, Swift <i>does</i>, which means it will treat two instances of our type as being incompatible if their generic type parameters are different. They aren't used often, but when they are used they help the compiler enforce extra rules on our behalf – they make bad states impossible because the compiler will refuse to build our code
<u>How does CloudKit differ from Core Data?</u>	Suggested approach: Although the two have many conceptual similarities, CloudKit is specifically designed to work remotely. Another key difference is that CloudKit lets you store data without worrying about your structure ahead of time, whereas Core Data requires that you define your structure up front.

How much experience do you have using Core Data? Can you give examples?	Suggested approach: Core Data is a huge and complex topic, but you should at least have tried it once. You might find it useful to talk about how NSPersistentContainer made Core Data easier to use from iOS 10 onwards, or compare and contrast Core Data and CloudKit. For a really great answer, talk about things that Core Data does well such as searching, sorting, and relationships, but also talk about places where Core Data struggles such as optionality and extensive stringly typed APIs.
How much experience do you have using Core Graphics? Can you give examples?	Suggested approach: Most developers have at least used Core Graphics for drawing basic shapes, but you might also have used it for text and resizing images. You should aim for at least a little experience, because this is one of the most important Apple frameworks.
What are the different ways of showing web content to users?	Suggested approach: You don't need to have named them all, but it certainly helps: UIWebView, WKWebView, SFSafariViewController, and calling openURL() on UIApplication. Don't just list them off, though: at least mention that UIWebView is deprecated, but if you can you should also compare and contrast WKWebView and SFSafariViewController.
What class would you use to list files in a directory?	Suggested approach: Hopefully your answer was FileManager. If your interviewer looked like they wanted more, you might want to talk about sandboxing: important directories such as documents and caches, using App Groups to share data between targets in your app, and more.
What is UserDefaults good for? What is UserDefaults not good for?	Suggested approach: This should immediately have you thinking about speed, size, and security: UserDefaults is bad at large amounts of data because it slows your app load, it's annoying for complex data types because of NSCoding, and a bad choice for information such as credit cards and passwords – recommend the keychain instead. If you're using SwiftUI extensively you could mention @AppStorage here.
What is the purpose of NotificationCenter?	Suggested approach: Most people use this for receiving system messages, for example to be notified when the keyboard appears or disappears, but you can also use it to send your own messages inside your app. Once you've outlined the basics, try comparing it against delegates.
What steps would you follow to make a network request?	Suggested approach: There are so many ways of answering this (not least "use Alamofire"), but the main thing is to demonstrate that you know it needs to be asynchronous to avoid blocking the main thread. Don't forget to mention the need to push work <i>back</i> to the main thread when it's time to update the user interface. For bonus points, mention that this is the kind of work Combine just eats up.

CGAffineTransform	Suggested approach: There are lots of ways of using these to manipulate the frame of a view, but an easy one is animation – you might make a view scale upwards, rotate, or grow larger over time for example. For bonus points, you might want to move on to discuss that affine scale transforms don't cause views to redraw at their larger size, which means that text is likely to appear fuzzy.
How much experience do you have using Core Image? Can you give examples?	Suggested approach: Some developers confuse Core Graphics and Core Image, which is a mistake – they are quite different. Core Image is used less often than Core Graphics, but is helpful for filtering images: blurring or sharpening, adjusting colors, and so on. For bonus points, talk about Core Image being able to apply multiple effects efficiently, and how it can also generate some kinds of images too.
How much experience do you have using StoreKit? Can you give examples?	Suggested approach: Most apps use only a small slice of StoreKit, whether it's unlocking in-app purchases, displaying other apps to purchase, or asking users to review the app. Either way, have <i>something</i> to talk about – it's better to say you've at least tried one of its features than have nothing at all to show.
How much experience do you have with GCD?	Suggested approach: Most developers have used Grand Central Dispatch at some point, either explicitly or implicitly – here the interviewer is probably trying to figure out which it is. You can approach this directly using DispatchQueue if you want, but you might also want to consider OperationQueue.
What class would you use to play a custom sound in your app?	Suggested approach: An easy answer is AVAudioPlayer, as long as you're clear about keeping the object alive while its sound plays. If you were feeling more confident you could discuss the pros and cons of using AudioServicesCreateSystemSoundID() instead – it's a slightly odd API, but it definitely works.
What experience do you have of NSAttributedString?	Suggested approach: This is an incredibly useful class, so hopefully your answer isn't "none"! Perhaps start by talking about how they are useful to add formatting like bold, italics, and color. You might also want to mention that they are great for hyperlinks, but for real bonus points mention that you can embed images inside them as well.
What is the purpose of GameplayKit?	Suggested approach: The clue is in the name: GameplayKit contains lots of helpful classes for games, such as AI strategists, state machines, and pathfinding. However, there's no reason its components must be limited just to games, because you can use them just as well in apps.

What is the purpose of ReplayKit?	Suggested approach: ReplayKit is one of Apple's more obscure frameworks, but it lets you record, save, and broadcast the user's activity in your app. It's most commonly used in games, but you could easily frame this in terms of submitting error report videos from users.
When might you use NSSortDescriptor?	Suggested approach: Start out with a simple definition of what NSSortDescriptor actually does: it lets us provide sorting instructions to a data store, e.g. "sort by name alphabetically". Once you've done that you should go on to mention that it's most commonly when used in Core Data to sort the data that got fetched, but it's also available in CloudKit. For bonus points, talk about how it can be a stringly typed API, or you can use key paths instead.
Can you name at least three different CALayer subclasses?	Suggested approach: This is an intermediate to advanced question aimed at folks with UIKit experience. You could choose from CAGradientLayer, CATiledLayer, CAEmitterLayer, CAShapeLayer, and more – they are all quite popular on Apple's platforms. If I were you, I'd list off a few different subclasses, but then pick one and try to provide more detail on how it works and why it exists.
What is the purpose of CADisplayLink?	Suggested approach: This lets you attach code to the user interface drawing loop so that your code always gets called immediately after a frame has been drawn and you have maximum time available to you. For bonus points, try to compare it to a more naive solution such as Timer.
How would you add a shadow to one of your views?	Suggested approach: In UIKit, all view layers have options for shadow opacity, radius, offset, color, and path. In SwiftUI, you can use the shadow() modifier. This would be a good time to mention to relative cost of dynamic shadows and how rasterizing layers in UIKit helps.
What features of recent iOS versions were you most excited to try?	suggested approach: This question is less about feature knowledge and more about your general excitement for iOS releases – if none of SwiftUI, widgets, SF Symbols, or App Clips interest you, you might be in the wrong career. I'd suggest a good approach here is to list a few broad things, but then pick one you really like and drill right into it.

What happens when Color or UIColor has values outside 0 to 1?	Suggested approach: In the old days values outside of 0 and 1 would be clamped (i.e., forced to 0 or 1) because they didn't mean anything, but wide color support means that is no longer the case – a red value beyond 1.0 is especially red, going into the Display P3 gamut.
How familiar are you with XCTest? Have you ever created UI tests?	Suggested approach: Be prepared to talk about the challenges you've faced when designing tests, particularly when it comes to user interface tests. It's a good idea to make sure you're crystal clear on the differences between unit tests and integration tests. Tip: I've never a development team that was 100% happy with their iOS UI tests.
How has Swift changed since it was first released in 2014?	Suggested approach: This can either show how long you've been writing Swift, or it can show you have an interest in the language's evolution. You might talk about the addition of try/catch , guard , and defer in Swift 2, the massive Cocoa renaming from Swift 3, the introduction of Codable in Swift 4, the changes for SwiftUI in Swift 5.1, or really whatever takes your interest.
How would you identify and resolve a retain cycle?	Suggested approach: The first step is identification – looking for a place where leaks happen, and it's important to mention either Instruments or the Memory Graph Debugger here. Leaks don't always means retain cycles (for example, unused caches are effectively leaks), but they are a good starting point. Once you've found a possible retain cycle, you need to decide which side of the cycle should be made weak rather than strong in order to resolve it.
What is an efficient way to cache data in memory?	Suggested approach: There are lots of ways of making caches, with the most humble being a simple dictionary, but whatever you choose you should be prepared to explain your choice and why you like it. Make sure and take into account how you remove data from the cache, either explicitly or to hit a memory quota. If you're comfortable talking about it, NSCache is definitely preferable over a simple dictionary because it automatically gets cleared by the system when memory is low.
What steps do you take to identify and resolve battery life issues?	Suggested approach: This is something so many developers don't ever think about, so use this as your chance to shine: talk about optimizing drawing, batching network requests, and minimizing work when the user isn't interacting with the app. Keep in mind that the battery settings app on iOS automatically shows which apps use the most battery life for a user, so having poor battery performance is very visible.

What steps do you take to identify and resolve crashes?	Suggested approach: Walk through your knowledge of debugging from the basics upwards. Do you use breakpoints? Do you use assert() or precondition()? Do you write to a log? Do you download crash logs from iTunes Connect? All these things help provide data points we can use to find and fix problems in our code, because once you know where the problem is it's usually(!) fairly straightforward to fix.
How does Swift handle memory management?	Suggested approach: Jump straight in with ARC – get to the point and focus on it, explaining about retain, release, and retain counts. If you want to also talk about unowned and weak, or perhaps reference counting for classes, this is also the place. For bonus points, talk about places that <i>aren't</i> automatic, such as using an autorelease pool inside tight loops.
How would you explain ARC to a new iOS developer?	Suggested approach: Automatic reference counting is a feature that Objective-C developers had to think about every day, but Swift developers mostly forget it's even there. Focus on why it saves developer time (less work) and what performance problems it might have (extra CPU time).
What steps do you take to identify and resolve a memory leak?	Suggested approach: Hopefully you have at least some experience with Instruments, so talk about persistent vs transient objects, talk about filtering for your custom data types, and so on. You should also discuss how you can be sure the leak is gone, for example if you push and pop the same view controller 10 times does the memory level remain constant?
What steps do you take to identify and resolve performance issues?	Suggested approach: This is a tricky question to answer because “performance” has many forms, so be prepared to adjust your answer as you talk based on interviewer feedback. If they mean graphical performance then you should probably talk about using the Core Animation instrument to identify slow drawing, but you should also consider venturing into the cost of Auto Layout in things like table view cells where lots of work happens quickly. If they mean code performance then perhaps talk about stack traces, retain cycles, unnecessary caches, and similar – again make sure and bring Instruments in. And then there's network performance, where you might talk about things such as batching requests to reduce battery wastage or using compression to save bandwidth.

How much experience do you have using Face ID or Touch ID? Can you give examples?	Suggested approach: If you're applying for a job at any company that has secure user data, biometric authentication is almost certainly involved somewhere. Fortunately, it's not hard to learn! For bonus points mention the need for a password backup in case Face ID/Touch ID fails, but if you're generally stuck here a good approach might be just to say "I haven't used it before, but I <i>have</i> used the keychain for secure storage."
How would you explain App Transport Security to a new iOS developer?	Suggested approach: This is your chance to demonstrate your security knowledge: why is HTTPS so important, and in what specific cases might you need to opt out? It also an opportunity to demonstrate your awareness of Apple's app review guidelines, which require secure transmission of user data.
What experience do you have of using the keychain?	Suggested approach: The keychain is the smartest way to store secure data on Apple's platforms. If you don't have any experience of using it at least be prepared to discuss why it's important for storing sensitive information – UserDefaults is the wrong choice!
How would you calculate the secure hash value for some data?	Suggested approach: Secure hash values use something like SHA-3, which is <i>not</i> the kind of code you'd want to write yourself. Instead, the best approach here is to mention something like Apple's CryptoKit framework, which can do hashing and encryption quickly, efficiently, and <i>correctly</i>.
How would you compare two tuples to ensure their values are identical?	Suggested approach: Swift provides automatic tuple comparison ever since Swift 2.2 – you can just use <code>==</code> to compare tuples item by item. For bonus points you could mention that tuple comparison does <i>not</i> guarantee that both tuples use the same element names.
How would you explain operator overloading to a junior developer?	Suggested approach: Operator overloading sounds like a complex topic, but really you can and should boil this answer down as small as you can: it allows us to use the same <code>+</code> operator with multiple types, such as integers, strings, doubles, and more. If you want to talk about examples of custom operator overloading, perhaps think about multiplying a CGPoint or something else that's easy to use in practice.
In which situations do Swift functions not need a return keyword?	Suggested approach: There are three: when the function isn't supposed to return a value, when it <i>is</i> supposed to return a value but you've used something like fatalError() to skip that requirement, and when it returns a value using a single expression. That second case is useful when you have placeholder functions you haven't implemented yet, or have created an abstract class where child classes will override your erroring implementations.

What are property observers?	Suggested approach: Property observers let you run code before or after a property is modified. Try to give a practical example here: “if you have a score property that holds an integer, you might attach a didSet observer so that it updates a label whenever the score changes.” For bonus points mention the major problem they have: you might think that adding 1 to an integer is a nice and quick operation, but if you accidentally attach a complex property observer then it could cause havoc
What does the #error compiler directive do?	Suggested approach: Answering this isn’t hard (“it forces the compiler to emit an error using a message we specify”), but please make sure you follow up with at least one example. So, maybe you’re shipping some sample code where users need to enter an API key otherwise it won’t work, so you use #error next to the API key line saying “fill in your API key before continuing.” Similarly, you might use #error alongside an OS check to say that your code isn’t compatible with tvOS, for example.
What does the #if swift syntax do?	Suggested approach: The syntax was added in Swift 2.2 to support compile-time version checking, meaning that you can mix two different Swift versions in one file without errors because only one will be used at a time. Make sure and give an example, such as Swift 5.4 supporting colors such as .red.opacity(0.5) whereas older versions need to use Color.red.opacity(0.5).
What does the assert() function do?	Suggested approach: This evaluates some code and causes your app to crash if the result is false. This is actually helpful because the check only happens in debug mode – it lets you make sure your code does what you think it does. For bonus points, give practical examples of where you’re using it in your current code.
What does the canImport() compiler condition do?	Suggested approach: You should get straight to the point and say that canImport() returns true if a module such as UIKit can be imported, then provide a practical example such as it allowing us to write code that does one thing using UIKit on iOS, and another thing using AppKit on macOS.
What does the CaseIterable protocol do?	Suggested approach: Go straight in and answer that this protocol allows us to loop over all cases in an enum, but then follow up with a practical example – you might have a word search generation algorithm that has enum cases for all possible directions you can place a word, so you loop over them all to try and find a valid spot for each word.

What does the final keyword do, and why would you want to use it?	Suggested approach: Think about the underlying goal here: why would you want to say “you cannot inherit from this class”? Is that a good idea? There are good reasons for using it – sometimes a class does something very precise, and you really don’t want users to override important parts of your code.
What does the nil coalescing operator do?	Suggested approach: The nil coalescing operator (??) lets you provide a default value to use if an optional value is empty. I would probably add that optionals are a really useful language feature in Swift, but if I can ditch them – if I can get a String rather than a String? – then I do, and that's where nil coalescing is really useful.
What is the difference between if let and guard let?	Suggested approach: Both check and unwrap optionals, but guard forces an early return if its check fails – your code will literally not compile unless you exit the scope. Furthermore, any variables that guard unwraps stay in scope after the guard block, whereas with if let the variables are available only inside the scope.
What is the difference between try, try?, and try! in Swift?	Suggested approach: A regular try requires you to catch errors, try? converts the throwing call into an optional where you’ll get back nil on failure, and try! will cause your app to crash if the throwing call fails. All three have their uses, so don’t dismiss one out of hand.
What problem does optional chaining solve?	Suggested approach: Optional chaining makes our code concise because we can write multiple optional calls on one line but have execution skip over the line if any of the optionals are missing. For bonus points, talk about this also works with things like try? and as? .
When would you use the guard keyword in Swift?	Suggested approach: It’s most commonly used to check preconditions are satisfied, but you should also discuss how variables it creates remain in scope after the guard block, and also how it enforces you exit the scope if the precondition fails. For bonus points, mention that you can use guard inside any kind of block as long as you escape afterwards – you can use it inside a loop for example.
Apart from the built-in ones, can you give an example of property wrappers?	Suggested approach: If it weren’t for the built-in restriction, this would be easy to answer with @State , @EnvironmentObject , and more, but with that restriction in place you need to be more creative – what real example can you think of? For example, a wrapper to make sure numbers are never negative, or strings are never empty, or perhaps arrays that silently stay sorted.

Can you give useful examples of enum associated values?	Suggested approach: Enum associated values let us attach one or more extra pieces of data to enum cases – that much is easy enough. However, the key word here is “useful”, which means you need to provide an example that is even vaguely real world. For instance, you might describe a weather enum that lists sunny, windy, and rainy as cases, but has an associated value for cloudy so that you can store the cloud coverage. Or you might describe types of houses, with the number of bedrooms being an associated integer. It doesn't really matter what example you choose, because the point is to show you understand why they are useful outside of a textbook!
How would you explain closures to a new Swift developer?	Suggested approach: Closures are easy to begin with, but make sure you talk about capturing of values, capture lists (unowned vs weak), and why they are actually <i>useful</i> compared to other approaches. It's possible your interviewer is looking for a broader discussion of functions as first-class types in Swift, so if they seem to be waiting for you to continue that might be a good angle to go down.
What are generics and why are they useful?	Suggested approach: Generics allow us to create types and functions that can be adapted to use different kinds of data for extra flexibility and safety. Generics are most commonly used to add type safety to collections – even if you don't create them yourself much, you certainly use them because [String] is really Array<String> under the hood. For bonus points, talk about how protocols use associated types rather than generics to achieve a similar result.
What are multi-pattern catch clauses?	Suggested approach: Swift's catch blocks let us catch several types of errors, specified using comma separation – hence the “multi-pattern” clauses. They are useful as a way of grouping error handling code together: if the error thrown was A or B then take the same action, but if it was C or D take some other action.
What does the @main attribute do?	Suggested approach: This attribute marks the point where your program starts to run. When using this attribute with one of your types need to implement a main() method to handle setting up your program, but if you're using SwiftUI the App protocol provides that for you.

What is a variadic function?	Suggested approach: Variadic functions accept any number of parameters. Swift writes them using <code>...</code>, and we're handed the parameters as an array. Again, try to give an example – something like <code>print()</code> is a good place to start. For bonus points, you could add that Swift 5.4 and later allow multiple variadic parameters.
What is the difference between weak and unowned?	Suggested approach: As well as explaining that weak becomes a regular optional whereas unowned is an implicitly unwrapped optional, be prepared to discuss the safety differences. Do you have a preference? If so, why? Can you think of places where unowned must be used?
What is the difference between an escaping closure and a non-escaping closure?	Suggested approach: This is an advanced language question and comes down to one thing: if you pass a closure as a function parameter and that closure might be called after the function has returned (e.g. after a delay), it must be marked as escaping. For bonus points, talk about why Swift uses non-escaping closures by default – they remove a small performance hit caused by Swift needing to keep escaping closures alive in memory after the function finishes.
What is the difference between an extension and a protocol extension?	Suggested approach: Extensions add functionality to specific data types, e.g. <code>Int</code>. Protocol extensions add functionality to protocols, for example all kinds of integers at the same time – <code>Int</code>, <code>Int8</code>, <code>UInt64</code>, and so on. For bonus points you could go on to talk about how protocol extensions enable protocol-oriented programming.
When would you use the defer keyword in Swift?	Suggested approach: Go straight in with a clear explanation before adding more detail: this is used to delay a piece of work until a function ends, similar to how <code>try/finally</code> works in some other languages. Once you've done that, be sure to give a specific example such as saving a file once everything has been written – any kind of work that absolutely must take place. For bonus points, mention how you can defer multiple pieces of work, and they get executed in reverse as Swift unwinds the <code>defer</code> stack.

How would you explain key paths to a new Swift developer?	Suggested approach: Start with the simplest explanation then you can, then try to follow up with practical examples. For example, you might start by saying that key paths let us refer to a property in a type rather than the exact value of that property in one particular instance. You could then follow up with examples such as using the map() method with a key path, to convert an array of users into an array of strings containing just their names, or perhaps using filter() to filter an array based on which items have a Boolean property set to true.
What are conditional conformances?	Suggested approach: You should already know that protocol conformances allow us to say that one type conforms to a protocol such as Hashable or Equatable, but this question is asking about <i>conditional</i> conformances – conforming to a protocol only if a condition is true. Make sure and give a practical example, such as Array conforming to a Purchaseable protocol only if it contains elements that also conform to Purchaseable.
What are opaque return types?	Suggested approach: Whenever you see some in a return type, it's an opaque return type – when you want to specify that some kind of type will be returned, but you don't want to say what. It's important that you try to explain the difference between an opaque return type and returning a protocol, because in the latter your returned value can be absolutely anything whereas in the former the compiler knows what data was actually returned even if you don't get access to that. For bonus points, talk about how SwiftUI uses @ViewBuilder to silently allow us to return different view types from a view body.
What are result builders and when are they used in Swift?	Suggested approach: This is an advanced question, so take your time. Start with a basic definition: result builders allow us to create a new value step by step by passing in a sequence of our choosing. Next, follow up with a specific example – I'd suggest that SwiftUI is the easiest one here, because when we have a VStack with a variety of views inside Swift silently groups them together into an internal TupleView type so that they can be stored as a single child of the VStack – it turns a sequence of views into a single view.
What does the <code>targetEnvironment()</code> compiler condition do?	Suggested approach: Get straight to the point and say it allows us to compile one set of code for the simulator, and another set of code for physical devices. Make sure and follow up with a practical example, such as a game that uses Core Motion to handle tilting movement on a real device, whereas on the simulator you need to tap the screen to simulate motion.

What is the difference between self and Self?	Suggested approach: Almost every Swift developer uses self regularly, but the question here requires to distinguish between that and the capitalized version. Start with self, which refers to the current object your code is running inside, then move on to Self, which refers to the current <i>type</i> your code is running inside. You can remember this by looking at the capital letter: we name our types using a capital first letter, so Self refers to a type.
When would you use @autoclosure?	Suggested approach: There are a few ways you could tackle this, but I would recommend either talking about the && operator or the assert() function – something you use regularly, and so feel comfortable talking about in detail. First, start with a simple definition: @autoclosure silently turns a function's parameter into a closure so that it can be executed on demand rather than immediately. Now, pick a specific example such as assert() and explain why it's used – the autoclosure behavior here ensures our assertion doesn't happen in release mode, so it won't have a performance impact when we ship apps to the App Store.
What is the difference between aspect fill and aspect fit when displaying an image?	Suggested approach: Although these two sound similar, there is an important difference: aspect fit ensures all parts of the image are visible even if it means there is some space around the edges, whereas aspect fill may crop the image to ensure it takes up all available space. Of course, just knowing the facts isn't enough: when would you use each of them?
What is the purpose of UIActivityViewController?	Suggested approach: This UIKit class has only one function: sharing your app's content using whatever other apps the user has. This might mean printing, posting to social media, saving an image to your photo library, and so on. For bonus points, talk about how you can inject your own activities into the mix, so they appear alongside system-wide services.
How does a view's intrinsic content size aid in Auto Layout?	Suggested approach: It describes how much space the view needs to lay out its content naturally, such as the amount of space some text needs to be fitted exactly. Intrinsic content sizes are used so that views occupy their natural amount of space by default, before the system decides if any views need to be stretched or squeezed.

[illegible]