iOS - Mock Technical Interview Questions

🟢 BEGINNER 🟢

What is the difference between a UIView and a UIViewController?

Overall, the main difference between a UIView and a UIViewController is their roles and responsibilities. A UIView is a visual element that displays content, while a UIViewController is a class that manages the views and provides the logic for the app's user interface.

What is the `viewDidLoad` method, and when is it called?

The `viewDidLoad` method is a useful method for setting up the view controller and its views before they are displayed. It is called once, immediately after the view has been loaded, and it provides a convenient place to initialize the view controller's state and to perform any other setup that is needed.

What is a storyboard, and how is it used in iOS app development?

A storyboard is a visual representation of the user interface of an iOS app, showing screens of content and the connections between them. It is a tool that is used in the early stages of app development to plan out the structure and functionality of the app.

🟡 INTERMEDIATE 🟡

How does the view hierarchy work in iOS?

In iOS, the view hierarchy refers to the **structure of views and subviews** that make up the user interface of an app. Each iOS app has a root view, which is the top-level view that contains all of the other views in the app. These views are organized in a **tree-like structure**, with the root view at the top, and any subviews arranged beneath it.

When a view is added to the view hierarchy, it becomes a part of the app's user interface, and can be seen and **interacted with by the user**. Each view can have its own subviews, which can be arranged and managed to create the desired layout and functionality of the app.

What is the difference between a delegate and an datasource in iOS?

The main difference between a delegate and a data source is the role they play in the relationship between two objects.

- A **delegate** is responsible for handling events and other tasks on behalf of another object.
- A **data source** is responsible for providing data to another object.

Example: A table view might have a delegate that is responsible for managing the table view's appearance and behavior, while the table view's data source provides the actual data that is displayed in the table view. The table view would communicate with its delegate and data source to determine what to display and how to respond to user interactions.

Explain the difference between a synchronous and asynchronous task in iOS.

The main difference between synchronous and asynchronous tasks is how they affect the app's ability to perform other operations while the task is being performed.

- With **synchronous** tasks, the app will pause or stop other activities until the task is completed
- With **asynchronous** tasks, the app can continue with other operations while the task is being performed in the background.

Synchronous tasks are useful when the app needs to wait for a specific operation to be completed before it can continue, such as when loading data from a server or performing a complex calculation. Asynchronous tasks, on the other hand, are useful when the app needs to perform multiple operations at the same time, or when it needs to perform operations in the background without interrupting the user's experience.

What is the purpose of the reuse identifier when using a UITableView in iOS?

The reuse identifier is **a string** that is **used to identify cells that can be reused** in a UITableView in iOS. It allows the table view to manage a pool of cells that can be reused, **rather than creating a new cell** each time one is needed.

The reuse identifier is specified when creating a new cell, and is later used by the table view to determine which cells can be reused. When a cell is no longer needed, it is added to the table view's pool of reusable cells, and can be used again later when a new cell is needed.

Explain the difference between a weak and strong reference in iOS.

The main difference between strong and weak references is how they affect the lifetime of the objects they reference.

- **Strong** references ensure that an object remains in memory as long as there is at least one strong reference to it.
- **Weak** references do not prevent the object from being deallocated from memory.

Weak references are often used in iOS **to avoid retain cycles**, which can cause memory leaks. A retain cycle occurs when two objects have strong references to each other, preventing either object from being deallocated from memory. By using a weak reference in one of the objects, the retain cycle can be broken and the objects **can be deallocated properly**.

🔴 ADVANCED 🔴

What is the difference between a class and a struct in iOS? When would you use each one?

The main difference between a class and a struct is how they are stored in memory.

- A **class** is stored on the **heap**, and is **accessed using a reference** to the object.
- A **struct** is stored on the **stack**, and is **accessed directly using its value**.

This means that when you pass an object of a class type to a function or method, you are passing a reference to the object, while when you pass an object of a struct type, you are passing a copy of the object's value.

Classes are useful **when you need to create complex objects** with multiple properties and methods, and when you need to share those objects across multiple parts of your app. **Structs**, on the other hand, are useful **when you need to define a simple data structure** with a few related values, and when you don't need to inherit from other types.

What is the difference between atomic and nonatomic properties in iOS?

The main difference between atomic and nonatomic properties is the level of thread safety they provide.

- **Atomic properties are thread-safe**, meaning that they can be accessed and modified from multiple threads without causing inconsistencies or other undefined behavior.
- **Nonatomic properties**, on the other hand, **are not thread-safe**, and can result in unpredictable behavior if they are accessed or modified from multiple threads simultaneously.

Atomic properties are useful when you need to ensure that access to a property's value is always consistent and well-defined, even when multiple threads are accessing the property simultaneously. Nonatomic properties, on the other hand, are useful when you don't need to worry about thread safety, and when the overhead of ensuring thread safety is not worth the performance cost.

Explain how Auto Layout works in iOS, and describe how to create a custom constraint.

Auto Layout is a system that is used in iOS to define the layout and size of views and controls in an app's user interface. It uses a set of constraints to **define the position** and **size** of views in a layout. Each constraint specifies a **relationship between two views**, such as the distance between them, their aspect ratio, or their alignment.

To **create a custom constraint** in Auto Layout, you can use the **NSLayoutConstraint** class to create a new constraint object, and specify the relationship between the two views that the constraint should apply to. You can then add the constraint to the views that it applies to, and the Auto Layout system will use the constraint to calculate the position and size of the views.

Explain the difference between copy and retain in iOS.

In iOS, the terms "copy" and "retain" refer to two different mechanisms that are used to manage the memory of objects. The main difference between copying and retaining an object in iOS is how they affect the relationship between the original object and the new instance.

- **Copying** creates a new instance of the object.
- **Retaining** maintains a reference to the original object.

This means that changes made to a copied object do not affect the original, while changes made to a retained object do affect the original.