PICKZY SOFTWARE QUESTIONS AND ANSWERS FOR SWIFT

1. What is Table View?

Table views on iOS display a single column of vertically scrolling content, divided into rows. Each row in the table contains one piece of your app's content. UITableView manages the basic appearance of the table, but your app provides the cells (UITableViewCell objects) that display the actual content.

2. How to design a Tableview

Step 1: **Create** a ViewController. ...
Step 2: In our ViewController class, instantiate a **UITableView** object. ...
Step 3: Setup the **tableView** and add constraints! ...
Step 4: Conform to UITableViewDelegate and UITableViewDataSource. ...
Step 5: **Create** our own custom subclass of UITableViewCell. ...
Step 6: Register the new cell class.

3. What is Datasource and delegate methods of TableView.

Datasource methods

Datasource methods are used to generate **tableView** cells, header and footer before they are displaying.

UITableViewDataSource protocol deal with providing data for the tableview. The **required methods** that need to be implemented are

- tableView:cellForRowAtIndexPath:

- tableView:numberOfRowsInSection:

cellForRowAtIndexPath return an instance of UITableViewCell that will be displayed in the tableView along with necessary data to display. (Can be a basic tableview cell or a custom tableview cell)

numberOfRowsInSection will tell how many rows are present in the tableview for that particular section.

Delegate methods

Delegate methods provide information about these cells, header and footer along with other user action handlers like cell selection and edit.

UITableView Delegate protocol methods deals with the way the UITableView appears. Like the height of the row, the header/footerview for a section or what needs to be done when we select a UITableViewCell.

4. What is Extensions and use of Extension?

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).

Type functionality can be added with extensions but overriding the functionality is not possible with extensions.

Swift Extension Uses −

- Adding computed properties and computed type properties
- Defining instance and type methods.
- Providing new initializers.
- Defining subscripts
- Defining and using new nested types
- Making an existing type conform to a protocol

Syntax

```
extension SomeType {
  // new functionality can be added here
}
```

Example

```
extension Int {
  var add: Int {return self + 100 }
  var sub: Int { return self - 10 }
  var mul: Int { return self * 10 }
  var div: Int { return self / 5 }
}

let addition = 3.add
print("Addition is \(addition)")

let subtraction = 120.sub
print("Subtraction is \(subtraction)")

let multiplication = 39.mul
print("Multiplication is \(multiplication)")

let division = 55.div
print("Division is \(division)")

let mix = 30.add + 34.sub
print("Mixed Type is \(mix)")
```

When we run the above program using playground, we get the following result −

Addition is 103
Subtraction is 110
Multiplication is 390
Division is 11
Mixed Type is 154

5.What is Auto layout?

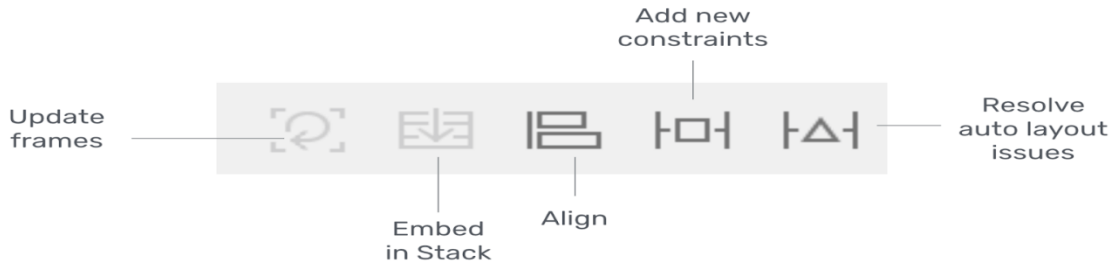
Making apps that look good in any orientation across multiple devices can be a challenge. Auto Layout makes it easy to support different screen sizes in our apps.

Xcode provides two ways to define auto layout constraints:

- **Control-drag method**
- **Auto Layout menu**

Control-drag method: It's the latest way to add constraints; we can simply control-drag from any view to another view to set constraints between each other.

Auto Layout menu: It is another way to add constraints with ease. At the bottom-right corner of the Interface Builder editor, you should find five buttons as shown below. These buttons are from the layout bar. You can use them to define various types of layout constraints and resolve layout issues.



- **Add new Constraints**:- This is used for adding new constraints to UI elements such as buttons.
- **Align**:- This quickly aligns items in one's layout.
- **Resolve Auto Layout Issues**:- It provides a number of options for fixing common Auto Layout issues ie *Upper half of the menu affects only the currently selected views while the bottom half affects all views in the scene.*
- **Embed in Stack:-** It allows one to quickly create a stack view.
- **Update frames:-** Updates the frame's position and size in reference to the given layout constraints.

6. Different between ? & !

we can use (?) when there are instances that a variable can be nil. And use (!) when you are 100% sure that item is not nil.

1. Working Fine - Optionals

```
var john:String?
john = "Is my name"
println(john!)
```

2. Crashes on Runtime - ! must not be nil - means this is correct

```
var john:String?
println(john!)
```

7. Different between Table view & Collection view

Table view

Table views on iOS display a single column of vertically scrolling content, divided into rows (ie similar to the way of listing details of each item). Each row in the table contains one piece of our app's content. **UITableView** manages the basic appearance of the table, but our app provides the cells (UITableViewCell objects) that display the actual content.

Collection view

Collection view is a way to present an ordered set of data items using a flexible and changeable layout. The most common use for collection views is to present items in a grid-like arrangement

8. Week variable & strong variable different in creation of outlet

A **strong** reference means that you want to "own" the object you are referencing with this property/variable. In contrast, with **a weak** reference you signify that you don't want to have control over the object's lifetime.

9. What is UIkit and its uses.

The UIKit framework provides the required infrastructure for iOS or tvOS apps. It provides the window and view architecture for implementing our interface, the event handling infrastructure for delivering Multi-Touch and other types of input to our app, and the main run loop needed to manage interactions among the user, the system, and our app. Other features offered by the framework include animation support, document support, drawing and printing support, information about the current device, text management and display, search support, accessibility support, app extension support, and resource management.

10. Var and let difference

let is used to declare a constant value - we won't change it after giving it an initial value.
var is used to declare a variable value - we could change its value as we wish.

11. Define JSON

- JSON stands for **J**ava**S**cript **O**bject **N**otation
- JSON is a lightweight format for storing and transporting data
- JSON is often used when data is sent from a server to a web page
- JSON is "self-describing" and easy to understand

12. Why we use JSON?

JSON format is used for serializing and transmitting structured data over network connection. It is primarily used to transmit data between a server and web applications. Web services and APIs use JSON format to provide public data. It can be used with modern programming languages.

13. Coredata and sqlite difference.

The most important **difference** between **Core Data and SQLite** is that **SQLite** is a database while **Core Data** is not.**Core Data** is a framework for managing an object graph. An object graph is nothing more than a collection of interconnected objects.

14. Required methods and optional methods usage

Required methods are used mandatorily
Optional methods are used when needed

15. Inheritance

A class can *inherit* methods, properties, and other characteristics from another class. When one class inherits from another, the inheriting class is known as a *subclass*, and the class it inherits from is known as its *superclass*. Inheritance is a fundamental behavior that differentiates classes from other types in Swift.