Q. Can we nest the Scaffold widget? Why or Why not?

Ans. Yes we can nest the scaffold widget.We usually nest it with the SafeArea widget, which is used to remove extra space from the top of the app bar.

Q. What are the different ways we can create a custom widget ?

Ans. In the flutter framework we can create custom widget by extending the inbuilt widget classes.There are three widget classes.

1. Stateless widget - This widget class is used when we want the static ui, which cant be rebuilt or redrawn during one lifecycle of a widget.to make the changes we need to rerun the app with different configuration.

To provide our custom ui override the build function and return the ui configuration you want from this function.

1. Stateful widget - This widget class is used when we want the dynamic ui, which can be rebuilt or redrawn during one lifecycle of a widget.to make the changes we need to call the setstate function with updated data or use the observer pattern to update with dependencies.(ie. Notifylistener method of providers.)

To provide our custom ui override the build function and return the ui configuration you want from this function.

1. Inherited widget - This widget class is used when we want to pass some property down the children tree. The child can access the property of the closest parent using of static method of the inherited class. When we create our custom widget we need to override this method.

To create this override the inherited widget and pass child widget in the constructor also provide implementation for of static method.

We need to override these classes to create custom widgets.

Q. How can I access platform(iOS or Android) specific code from Flutter?

Ans. To access platform specific code we need to use the platform channel feature of the flutter framework. Platform specific code (kotlin in android, swift in ios) can be written in MainActivity or AppDelegate class provided when we create a new flutter project. We can write our native code in the same class(ie. Checking for internet using connectivity manager).

**Connecting with flutter.**

We can call the native code from flutter using the invokeMethod function of platform class, it requires a channel name, usually it is a path string like com.app.xyz .

Sample Code.

platform = MethodChannel(‘com.app.xyz’);

Int result = await platform.invokeMethod(‘checkInternet’);

This is called the native code.

**Native code handling.**

We can register the same channel using method channel in configureFlutterEngine function in main activity and then override the setMethodChannel method to handle the invokeMethod from flutter code as shown above code.

Sample Code.

MethodChannel(flutterEngine.dartExecutor.binaryMessenger, ‘com.app.xyz’).setMethodCallHandler

{

call, result ->

if (call.method == "checkInternet")

// native code for checkInternet

Q. What is BuildContext? What is its importance?

Ans. In the flutter framework every widget is managed using the tree structure. When the build method is called for the widget, flutter framework add that widget to element tree( of the three trees in flutter framework), it gives us the buildcontext which tells us its location in the tree. Actually build context is a wrapper of element of a widget, which can be used to get ui info like its size using MediaQuery class. To remove a widget from ui we use this context and pass this to Navigator. Flutter framework uses context to find the closest Inherited widget(ie Theme.of(buildContext) to find the closest theme object above this widget in the tree).