* Flutter ka faida:

1. It uses only one language (dart) to do everything.
2. Hot reload: you don’t have to rebuild poora app everytime you make a change-efficiency.
3. Cross platform apps you can make.

* Flutter isn’t that good for web apps
* Stateless vs stateful widgets:

1. state doesn’t rebuild and state rebuilds.
2. Direct builder vs. create state.
3. Performance ka khayal rukhnay k liyay stateless.

* Widget ki lifecycle: build method
* Material App: material design say aya yeh widget
* Material design is for android and Cupertino design for ios. They’re both design systems. Basically every widget belongs to either of them and the difference is in the way the widget looks. You use .adaptive after a widget name if you need the widget to adjust according to the OS. For eg. **CircularProgressIndicator.adaptive**: Displays a progress indicator that looks like a Material CircularProgressIndicator on Android and a Cupertino CupertinoActivityIndicator on iOS.
* Scaffold gives the skeleton of the UI.
* Super.key: important for testing as it helps identify widget. Aap key day suktay ho to identify widget. Hur eik widget ko you can give key. If you don’t then you can’t identify the widget,
* Inline function: instead of executing a standard function call that requires pushing parameters onto the stack ( which causes overhead ) the code inside the function is expaned; function call is replaced with actual code of the function itself. Compiler takes body of function and places it exactly where function is called.

**When to Use Inline Functions**

1. **Small, Frequently Called Functions**: Inline functions are most effective for small functions that are called frequently, where the function call overhead would be noticeable.
2. **Performance-Critical Code**: In performance-sensitive applications, inlining can be used to optimize certain code paths.
3. **Function Code Simplicity**: Functions with simple logic and few lines of code are good candidates for inlining.

**When Not to Use Inline Functions**

1. **Large Functions**: Inlining large functions can lead to code bloat and increased memory usage, which can negatively impact performance due to cache misses.
2. **Recursive Functions**: Recursive functions are generally not suitable for inlining, as inlining all recursive calls would lead to excessive code duplication and potential stack overflow.
3. **When Function Address Is Needed**: If you need to take the address of a function (for example, using function pointers in C or C++), inlining might not be appropriate.

## if you wanna make apps for IOS just download the macOS on your PC

## read the material design flutter documentation : https://m3.material.io/develop/flutter