different ways of responsive in flutter

The difference between an adaptive and a responsive app

Adaptive and responsive can be viewed as separate dimensions of an app: you can have an adaptive app that is not responsive, or vice versa. And, of course, an app can be both, or neither.

Responsive

Typically, a responsive app has had its layout tuned for the available screen size. Often this means (for example), re-laying out the UI if the user resizes the window, or changes the device's orientation. This is especially necessary when the same app can run on a variety of devices, from a watch, phone, tablet, to a laptop or desktop computer.

Adaptive

Adapting an app to run on different device types, such as mobile and desktop, requires dealing with mouse and keyboard input, as well as touch input. It also means there are different expectations about the app's visual density, how component selection works (cascading menus vs bottom sheets, for example), using platform-specific features (such as top-level windows), and more.

Creating a responsive Flutter app

Flutter allows you to create apps that self-adapt to the device's screen size and orientation.

There are two basic approaches to creating Flutter apps with responsive design:

Use the LayoutBuilder class

From its builder property, you get a BoxConstraints object. Examine the constraint's properties to decide what to display. For example, if your maxWidth is greater than your width breakpoint, return a Scaffold object with a row that has a list on the left. If it's narrower, return a Scaffold object with a drawer containing that list. You can also adjust your display based on the device's height, the aspect ratio, or some other property. When the constraints change (for example, the user rotates the phone, or puts your app into a tile UI on Android), the build function runs.

Use the MediaQuery.of() method in your build functions

This gives you the size, orientation, etc, of your current app. This is more useful if you want to make decisions based on the complete context rather than on just the size of your particular widget. Again, if you use this, then your build function automatically runs if the user somehow changes the app's size.

Other useful widgets and classes for creating a responsive UI:

- **AspectRatio**
- **CustomSingleChildLayout**
- **CustomMultiChildLayout**
- **FittedBox**
- **FractionallySizedBox**
- **LayoutBuilder**
- **MediaQuery**
- **MediaQueryData**
- **OrientationBuilder**