debug mode shortcut (VSCode) 1

fn-F5

Dart variables 2

contain objects, where every object is an instance of a class

build() in widget 3

ie render(); displays widget using other widgets

Scaffold 4

provides a default app bar, title, and a body

pubspec.yaml does 5

manages assets and dependencies

final 6

final just means it can only be assigned to at declaration time.

const 7

const is for compile time constants

8

Question 9

Answer

child property 10

exists only in objects that expect one child; e.g. container

children prop 11

exists in objects that expect multiple children

runApp() 12

function to run app

main() 13

in `main.dart`, entry point into app

MaterialApp 14

basic widget for app which comes with title, home, theme, text directionality, back-button press handling, ...

scaffold 15

drawers, app bars, bottom navigation, tabs, and floating action buttons

AppBar 16

bar at top of app

Text 17

text widget

Dart top level functions means 18

functions can exist outside classes

ColorSwatch 19

a color that has a swatch of related colors

IconData 20

a description of an icon fulfilled by a font glyph

crossAxisAlignment 21

How the children should be placed along the cross axis.

edgeInsets 22

An immutable set of offsets in each of the four cardinal directions.Used by: Padding, a widget that accepts EdgeInsets to describe its margins.

Icon 23

A graphical icon widget drawn with a glyph from a font described in an IconData such as material's predefined IconDatas in Icons.Icons are not interactive. For an interactive icon, consider material's IconButton.There must be an ambient Directionality widget when using Icon. Typically this is introduced automatically by the WidgetsApp or MaterialApp.This widget assumes that the rendered icon is squared. Non-squared icons may render incorrectly.

_var 24

A leading underscore character (_) indicates that a member is private to its library. This is not mere convention, but is built into the language itself.

InkWell 25

A rectangular area of a Material that responds to touch; like touchable opacity

all layout is done with ... 26

widgets

Route is... 27

a widget which takes you to a page or screen

use route where? 28

e.g.MaterialApp(home: CategoryRoute___CategoryRoute(Scaffold

ListView, GridView 29

efficiently show a grid/list of items

navigator widget 30

manages routes; push and pop routes to move from screen to screen

Navigator, detail 31

A widget that manages a set of child widgets with a stack discipline.Many apps have a navigator near the top of their widget hierarchy in order to display their logical history using an Overlay with the most recently visited pages visually on top of the older pages. Using this pattern lets the navigator visually transition from one page to another by moving the widgets around in the overlay. Similarly, the navigator can be used to show a dialog by positioning the dialog widget above the current page

Navigator.of(context) 32

it's most common to use the navigator created by a WidgetsApp or a MaterialApp widget. You can refer to that navigator with Navigator.of

MaterialPageRoute 33

A modal route that replaces the entire screen with a platform-adaptive transition.

parts of widgets:StatelessWidget & StatefulWidget, constructor & ? 34

stateless: constructor -> build (loop)stateful: constructor -> createState (loop)

when variables inside the state are modified, widget does or does not automatically re-render? 35

does not

how to get re-render with state change? 36

call setState from widget implementor, changing the state object associated with StatefulWidget

State<> 37

State<> is a widget class extended to create state objects for a StatefulWidget. _StatefulWidgetClassState extends State

widgets extending State inherit overrides for: 38

build, which describes the part of the user interface represented by this widget setState, to notify the framework that the internal state of this object has changed

widgets extending StatefulWidget inherit overrides for: 39

createElement
createState

State class property .widget; accessed inside extended State class via `widget`, is what? 40

A State object's configuration is the corresponding StatefulWidget instance

stateless -> stateful, what happens to build()? 41

moves from stateless class to state class

Theme used for? 42

Applies a theme to descendant widgets.A theme describes the colors and typographic choices of an application.e.g.determine platform, and render widgets appropriately

43

Question 44

Answer

Scaffold 45

blank screen for app; implements the basic material design visual layout structure.This class provides APIs for showing drawers, snack bars, and bottom sheets.-appbar-body-drawer-navigation-floatingActionButtonuse it to place common items on your screen

Column, Row 46

A widget that displays its children in a vertical, horizontal array.

NetworkImage 47

get image from internet, typically provide as source Image(), using " urlThe image will be cached.

codesigning iOS 48

sign app build with certificate from Apple.ensures source of app, for security.enables installation, App Store, etc.

MaterialApp 49

app of Material designtypically root widgetAn application that uses material design.home property is the widget for the default route of the app.A convenience widget that wraps a number of widgets that are commonly required for material design applications.It builds upon a WidgetsApp by adding material-design specific functionality.

MaterialApp props 50

home: where our app starts

app starting point 51

in `main.dart`, start at `void main()`

AppBar 52

bar at top of apptitle, actions, ThemeIcon, etc.can put in scaffold's appbar prop

Image, widget 53

widget to contain an image, a portrait, with an image property to set

for strings, ' or " ? 54

' in Dart

runApp() 55

Inflate the given widget and attach it to the screen.

pubspec.yaml 56

configuration file;inform app of assets here

how to add an asset 57

1. folder for asset type (e.g. images), with asset, in root of project2. in pubspec.yaml, add asset to flutter; ```flutter: assets: -

how to use image asset 58

AssetImage('')

site for generating app icons 59

appicon.co

where to put app icons in Flutter 60

go to ios and android assets;android->app->src->main->res folder, replace mipmapsios->Runner->Assets.xcassets; replace Assets.xcassets

hot reload updates 61

stateless and stateful widgets

shortcut for statetless widget 62

type stless

build method 63

comes with stateless widget, gets called whenever we create a new version of this widget

hot restart 64

R; hot reload + state reset

container widget 65

one of the most fundamental widgets to layout apps

container 66

analagous to View, DivA convenience widget that combines common painting, positioning, and sizing widgets

container dimensions rules 67

Containers with no children try to be as big as possible unless the incoming constraints are unbounded, in which case they try to be as small as possible. Containers with children size themselves to their children. The width, height, and constraints arguments to the constructor override this.

safe area widget 68

container kept within bezels, notch of iPhone

safe area widget 69

container kept within usable area on screen; per OSA widget that insets its child by sufficient padding to avoid intrusions by the operating system.

container margin 70

Empty space to surround the decoration and child.

margin specified by 71

EdgeInsets.<>;.all : same all around.symmetric : top&bot same, l&r same.fromLTRB : custom each.only : only set margin for these

padding 72

space inside of widget

Column props 73

PROPERTIESchildrencrossAxisAlignment: layout along second axis (align to each other; would align them to the rightmost edge of the widest child. stretches along 2nd.directionhashCodekeymainAxisAlignment : layout along main axis, default start. MainAxisAlignment.mainAxisSize: MainAxisSize.runtimeTypetextBaselinetextDirectionverticalDirection: top to bot or bot to top; def. top2bot

column accepts children in 74

a widget array

By default, column takes up 75

all of available vertical space, but horizontally limits itself to its children

Opacity() 76

A widget that makes its child partially transparent.

CircleAvatar 77

circle that represents a user

Text() props 78

first is string, unnamed prop.other is Style

where to get fonts 79

fonts.google.com

where to store fonts in app 80

in root, directory fonts

where font documentation 81

using custom fonts flutter

specify font family with 82

in TextStyle, family: ''

how to add fonts to resources 83

pubspec.yaml;cannot add like `images/`, but must include exact path to font``` fonts: - family: Pacifico fonts: - asset: fonts/Pacifico-Regular.ttf```

Icon vs Image 84

icon is drawn, image is shown.this allows icon properties to be changes on the fly.icons are vectors, image not

Icons come from where 85

material package

material icon sites 86

Flutter icons, material icons https://material.io/tools/icons/?style=baseline, & materialpalette.com

code to add icon 87

Icon(Icons.)

Card widget 88

A material design card. A card has slightly rounded corners and a shadow.A card is a sheet of Material used to represent some related information, for example an album, a geographical location, a meal, contact details, etc.

cards accept padding prop, true or false 89

FALSE

cards accept margin prop, true or false 90

TRUE

cards accept children or child 91

child

Padding widget 92

A widget that insets its child by the given padding.Use over container with padding just for exact clarity

ListTile 93

A single fixed-height row that typically contains some text as well as a leading or trailing icon.A list tile contains one to three lines of text optionally flanked by icons or other widgets, such as check boxes. The icons (or other widgets) for the tile are defined with the leading and trailing parameters. The heights of the leading and trailing widgets are constrained according to the Material spec. An exception is made for one-line ListTiles for accessibility. Please see the example below to see how to adhere to both Material spec and accessibility requirements.Note that leading and trailing widgets can expand as far as they wish horizontally, so ensure that they are properly constrained.List tiles are typically used in ListViews, or arranged in Columns in Drawers and Cards.contentPaddingdenseenabledisThreeLineleadingonLongPressonTapselectedsubtitletitletrailing

Card default color 94

white

Divider 95

a 1-pixel thick horizontal line

96

Question 97

Answer

new project 98

To create a new Flutter project from the Flutter starter app template:Open the Command Palette (Ctrl+Shift+P (Cmd+Shift+P on macOS)).Select the Flutter: New Project command and press Enter.Enter your desired Project name.Select a Project location.

https://flutter.dev/docs/development/tools/vs-code 99

https://flutter.dev/docs/development/tools/vs-code

open terminal 100

ctrl-`

run app 101

in terminal, `flutter run`

reformat dart file 102

opt-shift-f

open file explorer 103

shift-cmd-e

hot reload 104

r in terminal

restart 105

shift-r

refactor options for widget, e.g. to wrap in new widget 106

select or move cursor to widget, then cmd+.

debug inspector; Dart Devtools 107

0. start debug session with debug-> start debugging1. Once the debug session is active and the application has started, the Dart: Open DevTools command will become available in the VS Code command palette2. DevTools will launch in your browser and automatically connect to your debug session.3.While the Dart DevTools are active, you'll see them in the status bar of VS Code. If you've closed the browser tab, you can click on here to re-launch your browser (so long as there's still a suitable Dart/Flutter debugging session available).

to enable hot reload on save, and other debug functionality 108

enter debug session with debug-> start debugging

start debugging 109

fn-f5

stop debugging 110

shift-fn-f5

start without debugging

cntrl-fn-f5

show/hide side panel 112

cmd-b

show/hide bottom panel 113

cmd-j

hot restart shortcut 114

shift-fn-command-f5

linter? 115

pedantic

116

Question 117

Answer

Expanded widget 118

A widget that expands a child of a Row, Column, or Flex so that the child fills the available space.Using an Expanded widget makes a child of a Row, Column, or Flex expand to fill the available space along the main axis (e.g., horizontally for a Row or vertically for a Column). If multiple children are expanded, the available space is divided among them according to the flex factor.An Expanded widget must be a descendant of a Row, Column, or Flex, and the path from the Expanded widget to its enclosing Row, Column, or Flex must contain only StatelessWidgets or StatefulWidgets (not other kinds of widgets, like RenderObjectWidgets).similar to flex; set weight with `flex: 2`, defaults to `flex: 1`

Image shorthands 119

new Image.asset, for obtaining an image from an AssetBundle using a key.new Image.network, for obtaining an image from a URL.new Image.file, for obtaining an image from a File.new Image.memory, for obtaining an image from a Uint8List.`Image.asset('')

FlatButton() 120

A material design "flat button".A flat button is a text label displayed on a (zero elevation) Material widget that reacts to touches by filling with color.requires onPressedchild can be anything (?)

Dart (){ // ... } 121

anonymous function

Dart string interpolation 122

'ashijf $<var> oaisjf'</var>

put updating vars inside...and declaration of vars... 123

build, captured by hot reloadoutside build, but property of widget

Dart dynamic data type 124

Dart is static, but the dynamic data type specifies any typeAlthough Dart is strongly typed, type annotations are optional because Dart can infer types. In the code above, number is inferred to be of type int. When you want to explicitly say that no type is expected, use the special type dynamic.`var a;a = 123;a = 'ok';`// works, because a is dynamic; type not specified on declaration line.also:dynamic a = 123;a = 'ok';

Object vs Dynamic 125

Some operations work with any possible object. For example, a log() method could take any object and call toString() on it. Two types in Dart permit all values: Object and dynamic. However, they convey different things. If you simply want to state that you allow all objects, use Object, as you would in Java or C#.Using dynamic sends a more complex signal. It may mean that Dart’s type system isn’t sophisticated enough to represent the set of types that are allowed, or that the values are coming from interop or otherwise outside of the purview of the static type system, or that you explicitly want runtime dynamism at that point in the program.

use dynamic? 126

generally, no. Keep type static safety; avoid var and `dynamic`

shortcut for StatefulWidget 127

stful

parts of StatefulWidget 128

StatefulWidget & StatefulWidgetState

similar parts between stful and stless widgets 129

the State and the stless widget; both have build

for stateful widget, where + how to set state? 130

call setState inside State;setState triggers rebuild

where to declare state? 131

inside State, outside build.

setState 132

Notify the framework that the internal state of this object has changed.Whenever you change the internal state of a State object, make the change in a function that you pass to setState:Calling setState notifies the framework that the internal state of this object has changed in a way that might impact the user interface in this subtree, which causes the framework to schedule a build for this State object.When called, app goes thru widget, finds places this state is used, and updates there.

put MaterialApp inside or outside a widget... 133

in the build of a widget, so that properties there can be hot reloaded

134

Question 135

Answer

how to add dependencies 136

in pubspec.yaml, add under dependencies

dependency sytnax 137

`dependencies: (optional>)

stless widget constructor for widget properties 138

class KeyNote extends StatelessWidget { final Color color; final int assetNumber; const KeyNote({ Key key, this.color, this.assetNumber, }) : super (key: key); @override Widget build(BuildContext context) { return Padding(padding: const EdgeInsets.only(bottom: 10.0), child: FlatButton(color: color, onPressed: playNote,),); } void playNote () { final player = AudioCache(); player.play('note$assetNumber.wav'); }}

named, optional Dart function parameters 139

A parameter wrapped by [ ] is a positional optional parameter. Here is an example:getHttpUrl(String server, String path, [int port=80]) { // ...}A parameter wrapped by { } is a named optional parameter.foo({@required String name}) {...}

on column or row, stretch along main axis for available space 140

expanded wraps children

type dart functions? 141

if returning, type function of return; int getMilk() { //... return 1}

num dart 142

either int or floating point number type

dart fat arrow syntax, equivalent 143

The ` => expr` syntax is a shorthand for ` { return expr; } `

144

Question 145

Answer

setState call syntax 146

void setState(void Function fn)setState(() { // ... });

Dart class syntax 147

class Name { // properties Name(){ // constructor }}

pillars of OOP 148

. Abstraction. Polymorphism. Inheritance. Encapsulation

where to add classes 149

in lib folder

abstraction 150

abstract away details of implementation; get to ideaAbstraction is the technique of hiding implementation. At it's core there's not much more to that answer. The bulk of meaning to abstraction come from how and why it is used.It is used for the following scenariosReduce complexity. (Create a simple interface)Allow for implementation to be modified without impacting its users.Create a common interface to support polymorphism (treating all implementations of the abstracted layer the same.Force users to extend the implementation rather than modify.Support cross platform by changing the implementation per platform.

encapsulation 151

separate jobs+roles of different objects/classesIt's simply a containment of information.Encapsulation means that a class publishes only what is needed for others to use it, and no more.

enum syntax 152

enum Color { red, green, blue }

how to make a property private 153

prefix __prop

inheritance syntax dart 154

Subclass extends ParentClass

inheritance concept 155

inherit common properties and functions from parent class

polymorphism concept 156

changing shapes;Poly = manyMorph = change or formSo polymorphism is the ability (in programming) to present the same interface for differing underlying forms (data types).override common interface for custom subclass; e.g.Shape super, .draw()circle override .draw()triangle override .draw()many forms, one interface (draw)

how to override in dart 157

@overrideoverriddenFunction () { //... }

double 158

double data type in dart

class constructor syntax for same parameter and class prop name 159

class MyClass { int prop; MyClass (this.prop)

160

Question 161

Answer

theme 162

define a color palette

cookbook 163

https://flutter.dev/docs/cookbook for task examples

primary color 164

The background color for major parts of the app (toolbars, tab bars, etc)

accent color 165

The foreground color for widgets (knobs, text, overscroll edge effect, etc).

Color constructor 166

typical hexcode, ARGB, ...

color for text in them 167

textTheme

how to change theme for local widget 168

wrap widget in Theme widget

ThemeData.dark().copyWith(...) 169

create a copy of this theme, but edit inner parameters

color of container is shorthand for setting 170

color of box decoration of container

DRY 171

Don't Repeat Yourself

how to make a parameter required 172

add @required before parameter;class ExampleClass { bool isBool; ExampleClass({@required this.isBool});

immutability 173

cannot be changed

a stless widget is mutable or immutable? 174

immutable

const v final 175

const figured out at compile time; final set only oncefinal could be set to final string currentTime = DateTime.now();const could not

Dart class initializer order 176

1. init list 2. constructor

initializer list 177

init properties (including final ones) before initialization

flatbutton vs inkwell vs gesturerecognizer 178

flatbutton stylized + opinionatedinkwell has onPressed and visual feedback (ink splash)gesturerecognizer no visual feedback nor style, many gestures

function type 179

so even functions are objects and have a type, Function.

void in Dart 180

denotes absence; return type of a function; can be anything but can’t be used for anything

void onTapMale() { // ... }means what? 181

onTapMale is a function (of type Function) which returns nothing (void)

enum syntax 182

enum EnumName { ta, tb, tc, ...}EnumName.ta

syntactic equivalent of void myFunc () {} 183

Function myFunc = (){}

constants prefix 184

k

constants dropdown 185

k ...

add a theme to a particular widget 186

wrap that widget in a theme of its type

.of(), .copyWith() 187

make this one of a context, then copy it with certain overriding changes