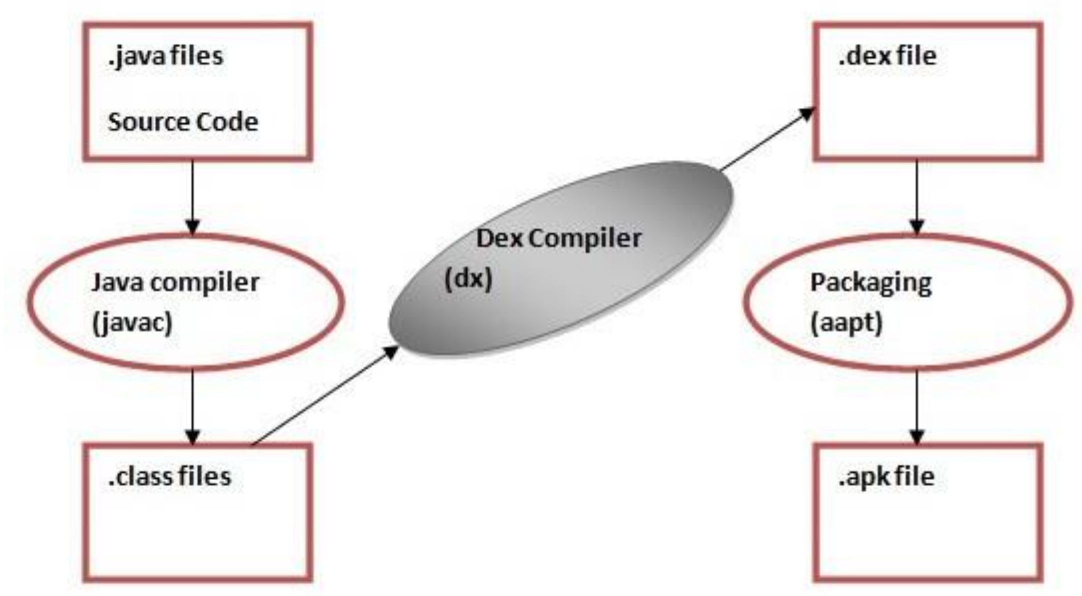
Dalvik Visual Machine:

The **Dalvik Virtual Machine (DVM)** is an android virtual machine optimized for mobile devices. It optimizes the virtual machinefor memory, battery life *and* performance.

Dalvik is a name of a town in Iceland. The Dalvik VM was written by Dan Bornstein.

The Dex compiler converts the class files into the .dex file that run on the Dalvik VM. Multiple class files are converted into one dex file.



The **javac tool** compiles the java source file into the class file.

The **dx tool** takes all the class files of your application and generates a single .dex file. It is a platform-specific tool.

The **Android Assets Packaging Tool (aapt)** handles the packaging process.

Components :

- ClassLoader
- DEX Compiler
- Garbage Collector (GC)
- Interpreter
- JIT Compiler
- Ahead-Of-Time (AOT) Compiler
- Memory Manager
- Verifier
- Optimization Tools

Just In Time (JIT): *Dalvik VM*

With the Dalvik JIT compiler, each time when the app is run, it dynamically translates a part of the Dalvik bytecode into machine code. As the execution progresses, more bytecode is compiled and cached. Since JIT compiles only a part of the code, it has a smaller memory footprint and uses less physical space on the device.

Ahead Of Time (AOT): *Android Runtime (ART)*

ART is equipped with an Ahead-of-Time compiler. During the app's installation phase, it statically translates the DEX bytecode into machine code and stores in the device's storage. This is a one-time event which happens when the app is installed on the device.

Screen Orientation :

```
<activity android:name=".SecondActivity"
android:screenOrientation="landscape">
</activity>
```

RadioButton:

```
public void onclickbuttonMethod(View v){
<RadioGroup
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/radioGroup">

<RadioButton
android:id="@+id/radioMale"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text=" Male"
android:checked="false"/>
```

.JAVA

```
int selectedId = radioGroup.getCheckedRadioButtonId();
genderradioButton = (RadioButton) findViewById(selectedId);
if(selectedId==-1){
Toast.makeText(MainActivity.this,"Nothing selected", Toast.LENGTH_SHORT).show(); }
else{
Toast.makeText(MainActivity.this,genderradioButton.getText(),
Toast.LENGTH_SHORT).show();
}
```

Spinner

In Android, a Spinner is a drop-down menu that allows users to select an item from a list of options. It is commonly used to present a set of choices to the user, such as selecting a category, choosing a date, or picking an item from a list.

ArrayAdapter<String> adapter = new ArrayAdapter<>(this,
android.R.layout.simple_spinner_item, options);
spinner.setAdapter(adapter);

AlertDialogs

AlertDialogs in Android are pop-up windows that prompt the user to take some action or provide information. They are commonly used to display messages, gather user input, or confirm an action before proceeding. AlertDialogs typically consist of a title, a message, buttons for user interaction, and optionally, additional custom views.

```
AlertDialog.Builder SignoutAlert = new AlertDialog.Builder(MainActivity.this);
SignoutAlert.setTitle("Confirm Signout");
SignoutAlert.setMessage("Are you sure to Signout");
SignoutAlert.setPositiveButton("Yes", new DialogInterface.OnClickListener()
{
    @Override
    public void onClick(DialogInterface dialog, int which) {
        Toast.makeText(MainActivity.this, "Signout successfully",
Toast.LENGTH_SHORT).show();
    }
});

SignoutAlert.setNegativeButton("No", new DialogInterface.OnClickListener()
{
    @Override
    public void onClick(DialogInterface dialog, int which) {
        Toast.makeText(MainActivity.this, "Cancled",
Toast.LENGTH_SHORT).show();
    }
});
SignoutAlert.show();
```

Methods : setTitle(), setMessage(), setIcon(), setPositiveButton, show()

ListView

ListView is a fundamental component in Android used to display a scrollable list of items. It is commonly used to present large datasets or lists of information to users in a vertical scrollable layout. ListView efficiently manages memory by recycling its child views as they scroll off-screen, making it suitable for displaying large datasets without consuming excessive memory.

```
listView = (ListView) findViewById(R.id.listView);
ArrayList<String> arrName = new ArrayList<>();
arrName.add("Vinayak");

ArrayAdapter<String> adapter = new
ArrayAdapter<String>(getApplicationContext(),
android.R.layout.simple_list_item_1,arrName);
listView.setAdapter(adapter);
```

Android SearchView

Android **SearchView** provides user interface to search query submitted over search provider. SearchView widget can be implemented over ToolBar/ActionBar or inside a layout.

public boolean onQueryTextSubmit(String query): It searches the query on the submission of content over SearchView editor. It is case dependent.

public boolean onQueryTextChange(String newText): It searches the query at the time of text change over SearchView editor.

```
adapter = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1,list);
listView.setAdapter(adapter);
    searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
            @Override
            public boolean onQueryTextSubmit(String query) {

                if(list.contains(query)){
                    adapter.getFilter().filter(query);
                }else{
                    Toast.makeText(MainActivity.this, "No Match found",Toast.LENGTH_LONG).show();
                }
                return false;
            }
```

Autocomplate Textview:

```
1. AutoCompleteTextView actv =  (AutoCompleteTextView)findViewById(R.id.autoComp
   leteTextView);
2.        actv.setThreshold(1);//will start working from first character
          ArrayAdapter<String> adapter = new ArrayAdapter<String>
                    (this,android.R.layout.select_dialog_item,language);
3.        actv.setAdapter(adapter);
```

DatePicker:

android:datePickerMode = ”spinner”

```
tvw.setText("Selected Date: "+ picker.getDayOfMonth()+"/"+ (picker.getMonth() +
1)+"/"+picker.getYear());
```

TimePicker:

```
timepicker.setIs24HourView(true);
```

```
Showtxt.setText(getCurrenttime());
Public string getCurrenttime(){
String currentTime="Current Time: "+timepicker.getCurrentHour()+":"+timepicker.getCurrent
Minute();
return currenttime;
}
```

ProgessBar:

```
if (isProgressVisible) {
          showProgressBtn.setText("Show Progress Bar");
          loadingPB.setVisibility(View.GONE);
          isProgressVisible = false;
        } else {
          isProgressVisible = true;
```

```
    showProgressBtn.setText("Hide Progress Bar");

    loadingPB.setVisibility(View.VISIBLE);
}
```

.lib files

In Dart, .lib files typically represent Dart libraries. However, there is no strict requirement for Dart files to have the .lib extension; it's more common to use .dart for Dart source files.

Dart libraries are collections of functions, classes, and other code elements that can be imported and reused in other Dart files.

pubspec.yaml

pubspec.yaml is a YAML file used in Dart and Flutter projects to define metadata and dependencies for the project.

It typically resides in the root directory of your Dart or Flutter project.

It contains information such as the project name, description, version, author, and dependencies.

```
dependencies:
  flutter:
    sdk: flutter
    Toast ^3.1.0
```

the **state** of the widget is the information of the objects that its properties (parameters) are holding at the time of its creation (when the widget is painted on the screen).

Types of layout Widgets :

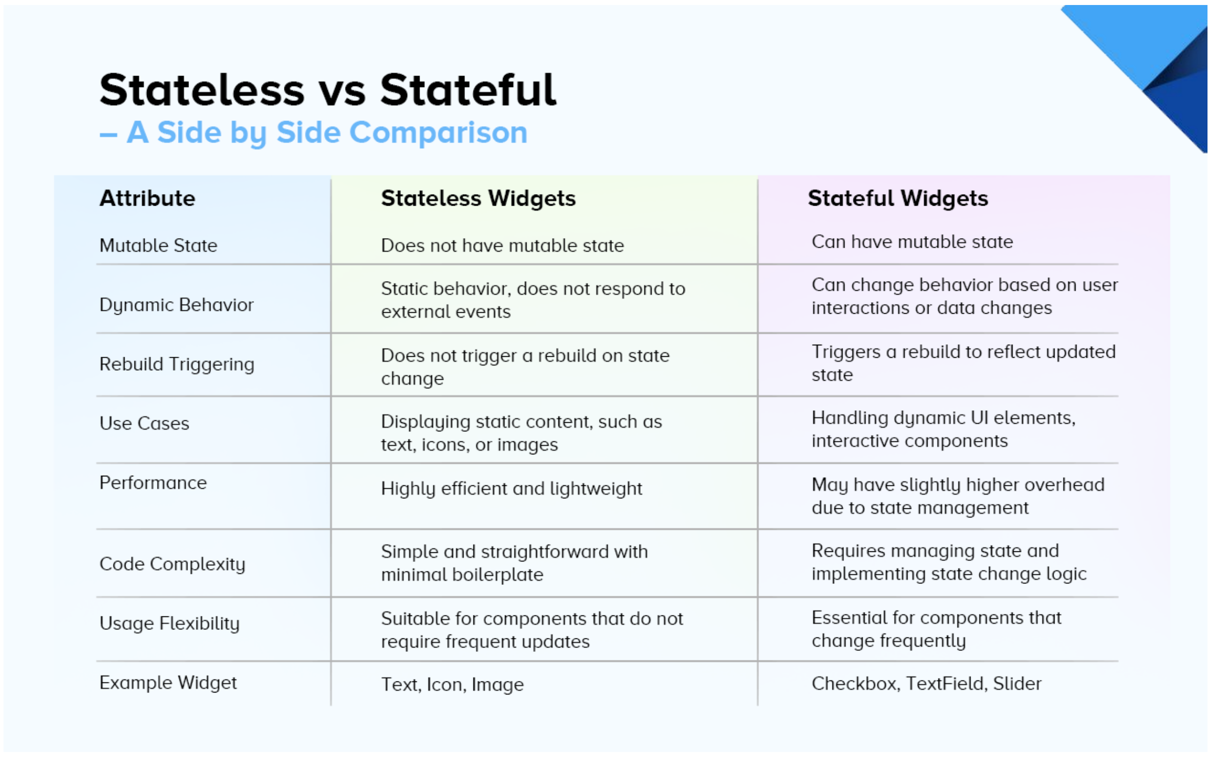
1. Single Child Widget
2. Multiple Child Widget

Single Child Widget

- Container
- Center
- SizedBox
- AspectRatio
- Baseline
- ConstrainedBox

Multiple Child widgets

- Column
- ListView
- GridView
- Stack



Stateless vs Stateful

– A Side by Side Comparison

Attribute	Stateless Widgets	Stateful Widgets
Mutable State	Does not have mutable state	Can have mutable state
Dynamic Behavior	Static behavior, does not respond to external events	Can change behavior based on user interactions or data changes
Rebuild Triggering	Does not trigger a rebuild on state change	Triggers a rebuild to reflect updated state
Use Cases	Displaying static content, such as text, icons, or images	Handling dynamic UI elements, interactive components
Performance	Highly efficient and lightweight	May have slightly higher overhead due to state management
Code Complexity	Simple and straightforward with minimal boilerplate	Requires managing state and implementing state change logic
Usage Flexibility	Suitable for components that do not require frequent updates	Essential for components that change frequently
Example Widget	Text, Icon, Image	Checkbox, TextField, Slider

Features of Flutter :

- Cross Platform
- Open Source
- Hot Reload
- Minimal code
- Accessible Native Features and SDKs
- Widgets

Dart :

Dart is an open-source general-purpose programming language developed by Google. It supports application development on both the client and server side. However, it is widely used for the development of Android apps, iOS apps.

- **DartPad :** DartPad is an online editor and interactive development environment (IDE) for the Dart programming language. Dart is a general-purpose programming language developed by Google, primarily for building web, mobile, and server applications. DartPad allows developers to write, run, and share Dart code directly in their web browser without the need to install any software or set up a development environment.

- **dart2js**, on the other hand, is a command-line tool provided as part of the Dart SDK. It is used to compile Dart code to JavaScript code. DartPad itself internally utilizes dart2js to compile Dart code to JavaScript so that it can be executed within the web browser environment. dart2js optimizes Dart code for production deployment, including tree shaking (removing unused code), minification (reducing code size), and other optimizations to ensure efficient and performant JavaScript output.

Features of Dart :

- Open Source
- Platform Independent
- Object-Oriented
- Concurrency
- Extensive Libraries
- Easy to learn
- Flexible Compilation
- Type Safe
- Objects
- Browser Support
- Large Community

Dart Data Type :

- int (**int** marks = 80;)
- double(**double** pi = 3.14;)
- Strings(var msg = "Welcome to JavaTpoint";)
- Boolean(bool isValid = **true**;)
- Lists(var list = [1,2,3])
- Maps(var student = {'name': 'Joseph', 'age':25, 'Branch': 'Computer Science'})
- Runes
    1. var laugh_symbol = '\u{1f600}';
    2. print(laugh_symbol);

- The **break statement** in Dart inside any loop gives you a way to break or terminate the execution of the loop containing it, and hence transfers the execution to the next statement following the loop. It is always used with the if-else construct.

Calling function, deleting function:

```
void myFunction() {
  print("This is my function");
}

void main() {
  // Call the function
  myFunction(); // Output: This is my function

  myFunction = null;
  // myFunction(); // This line would result in a compilation error
}
```

Flutter Widgets :

Visible widget

- Text
- Button
- Image
- Icon

Invisiblen widget:

- Column
- Row
- Center
- Padding
- Scaffold
- Stack

A **StatefulWidget** has state information. It contains mainly two classes: the **state object** and the **widget**. It is dynamic because it can change the inner data during the widget lifetime. This widget does not have a **build()** method. It has **createState()** method, which returns a class that extends the Flutters State Class. The examples of the StatefulWidget are Checkbox, Radio, Slider, InkWell, Form, and TextField.

- TextField, Checkbox, RadioButton, Switcher, Slider, DropdownButton, BottomNavigationBar, PageView

```
class Car extends StatefulWidget {
  const Car({ Key key, this.title }) : super(key: key);

  @override
  _CarState createState() => _CarState();
}

class _CarState extends State<Car> {
  @override
  Widget build(BuildContext context) {
    return Container(
      color: const Color(0xFEEFE),
           child: Container(
            child: Container( //child: Container() )
        )
    );
  }
}
```

A **stateless widget** is immutable, meaning once it is created, its properties cannot change. It does not have any internal state that can change over time.

- Text, Image, Icon, RaisedButton, FlatButton, IconButton, Container, Row, Column, ListView, GridView

```
class MyStatelessCarWidget extends StatelessWidget {
  const MyStatelessCarWidget ({ Key key }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Container(color: const Color(0x0xFEEFE));
  }
}
```

Button :

Buttons are material components that provide the user a single tap facility for taking actions, making choices, submitting forms, saving data, opening a new page, etc. Buttons are triggered when the user taps on them. They are the most commonly used feature provided in almost all the flutter apps.

- Elevated Button
- Floating Action Button
- Outlined Button
- Icon Button
- Text Button
- Dropdown Button
- PopUp Menu Button

```
FlatButton(
              color: Colors.green,
              child: const Text(
                'Flat Button',
                style: TextStyle(color: Colors.white),
              ),
              onPressed: () {},
            ),
```

Text Widget:

The Text widget is used to display text in a Flutter application. It's a fundamental widget for conveying information to users. Text widget allows customization of text properties such as style, color, alignment, and more.

```
Text(
  'Hello, Flutter!',
  style: TextStyle(
    fontSize: 20,
    fontWeight: FontWeight.bold,
    color: Colors.blue,
  ),
)
```

Image Widget:

The Image widget is used to display images in a Flutter application. It supports various image sources, including local assets, network URLs, and memory images. Image widget allows customization of image properties such as width, height, fit, alignment, and more.

```
Image.network(
  'https://example.com/image.jpg',
  width: 200,
  height: 200,
  fit: BoxFit.cover,
)
```

Toast in Flutter :

```
dependencies:
  flutter:
    sdk: flutter
    Toast ^3.1.0
```

Property:

- Msg
- Toastlength
- Gravity
- timeInSecForIos
- backgroundColor
- textColor
- fontSize

```
-    Fluttertoast.showToast(
-            msg: 'This is toast notification',
-            toastLength: Toast.LENGTH_SHORT,
-            gravity: ToastGravity.BOTTOM,
-            timeInSecForIos: 1,
-            backgroundColor: Colors.red,
-            textColor: Colors.yellow
-        );
```