**HTML5**

HTML5is the latest and most enhanced version of HTML.Technically, HTML is not a programming language, but rather a mark up language. HTML5 is a standard for structuring and presenting content on the World Wide Web.Most of the new features of HTML5 are well supported by common browsers like Mozilla Firefox, Google Chrome, Internet Explorer, Opera, Mobile Browsers etc. Here is a set of some of the most prominent features introduced in HTML5.

**New Semantic Elements**: These are like ,<Header> ,<Footer> and <section>

**Forms 2.0:**Improvements to HTML web forms where new attributes have been introduced for <Input> tag.

**Persistent Local Storage:** To achieve without resorting to third-party plugins

**WebSocket**: A next-generation bidirectional communication technology for web applications.

**Server-Sent Events:** HTML5 introduces events which flow from web server to theweb browsers and they are called Server-Sent Events (SSE).

**Canvas:** This supports a two-dimensional drawing surface that you can program with JavaScript.

**Audio & Video:** You can embed audio or video on your webpages without resortingto third-party plugins

.

**Geolocation:** Now visitors can choose to share their physical location with your web application.

**Microdata:** This lets you create your own vocabularies beyond HTML5 and extendyour web pages with custom semantics.

**Drag and drop:** Drag and drop the items from one location to another location onthe same webpage.

**HTML5Attributes**:

Elements may contain attributes that are used to set various properties of an element.Some attributes are defined globally and can be used on any element, while others are defined for specific elements only. All attributes have a name and a value and look like shown below in the example.

**Canvas:**

HTML5 element gives you an easy and powerful way to draw graphics. It can be used to draw graphs, make photo compositions or do simple (and not so simple) animations. Canvas has several methods for drawing paths, boxes, circles, text, and adding images.

**SVG:**

SVG is used to define graphics for the Web. SVG has several methods for drawing paths, boxes, circles, text, and graphic images.

**HTML5 web storage:**

With web storage, web applications can store data locally within the user's browser. HTML5 introduces two mechanisms, similar to HTTP session cookies, for storing structured data on the client side and to overcome following drawbacks. Cookies are included with every HTTP request, thereby slowing down your web application by transmitting the same data.

**Session Storage** The Session Storage is designed for scenarios where the user is carrying out a single transaction, but could be carrying out multiple transactions in different windows at the same time.

**Local Storage** The Local Storage is designed for storage that spans multiple windows, and lasts beyond the current session.

**Cascade style sheet:**

CSS is a language that describes the style of an HTML document.CSS describes how HTML elements should be displayed. CSS handles the look and feel part of a web page. Using CSS, you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colours are used, layout designs, variations in display for different devices and screen sizes as well as a variety of other effects

Advantages of CSS

**● CSS saves time −** You can write CSS once and then reuse same sheet in multiple HTML pages. You can define a style for each HTML element and apply it to as many Web pages as you want.

● **Pages load faster −** If you are using CSS, you do not need to write HTML tag attributes every time. Just write one CSS rule of a tag and apply it to all the occurrences of that tag. So less code means faster download times.

**● Easy maintenance −** To make a global change, simply change the style, and all elements in all the web pages will be updated automatically.

**● Superior styles to HTML −** CSS has a much wider array of attributes than HTML, so you can give a far better look to your HTML page in comparison to HTML attributes.

**● Multiple Device Compatibility −** Style sheets allow content to be optimized for more than one type of device. By using the same HTML document, different versions of a website can be presented for handheld devices such as PDAs and cell phones or for printing.

**● Global web standards −** Now HTML attributes are being deprecated and it is being recommended to use CSS. So its a good idea to start using CSS in all the HTML pages to make them compatible to future browsers.

**● Offline Browsing −** CSS can store web applications locally with the help of an offline catche.Using of this, we can view offline websites.The cache also ensures faster loading and better overall performance of the website.

**● Platform Independence −** The Script offer consistent platform independence and can support latest browsers as well.

CSS comes in three types:

**● In a separate file (external)**

**● At the top of a web page document (internal)**

**● Right next to the text it decorates (inline)**

**External style** :

External style sheets are separate files full of CSS instructions (with the file extension .css). When any web page includes an external stylesheet, its look and feel will be controlled by this CSS file (unless you decide to override a style using one of these next two types). This is how you change a whole website at once. And that's perfect if you want to keep up with the latest fashion in web pages without rewriting every page!

**Internal styles:**

Internal styles are placed at the top of each web page document, before any of the content is listed. This is the next best thing to external, because they're easy to find, yet allow you to 'override' an external style sheet -- for that special page that wants to be a nonconformist!

**Inline styles** :

Inline styles are placed right where you need them, next to the text or graphic you wish to decorate. You can insert inline styles anywhere in the middle of your HTML code, giving you real freedom to specify each web page element. On the other hand, this can make maintaining web pages a real chore!

**Embedded Style Sheet:**

Embedded style sheets refer to when you embed style sheet information into an HTML document using the tags in the head of your document.

**Android:**

Android's OS is owned by Google and is considered open-source to Apple's closed system. With an open-source system, software developers can build applications and programs until their heart's content and users reap the benefits of tons of free software.Most Android lovers cite the operating system's openness, which leads to more personalization options, as the reason they're hardcore devotees.

**Android SDK** :

The Android software development kit (SDK) includes a comprehensive set of development tools. These include a debugger, libraries, a handset emulator based on QEMU, documentation, sample code, and tutorials. Currently supported development platforms include computers running Linux (any modern desktop Linux distribution), Mac OS X 10.5.8 or later, and Windows 7 or later. As of March 2015, the SDK is not available on Android itself, but software development is possible by using specialized Android applications.

The Android SDK can be broken down into several components. These include:

● Platform-tools

● Build-tools

● SDK-tools

● The Android Debug Bridge (ADB)

● Android Emulator

**Android Studio**

No list of Android development tools would be complete without Android Studio. This is the official IDE (Integrated Development Environment) for Android, making it the number one choice for the majority of developers looking to make basic apps in-keeping with Google’s Material Design and with access to all the advanced features of the platform. The IDE is where any developer will spend most of their time: it acts as an editor for the chosen programming language (Android Studio supports Java, C++ and now Kotlin, though Java is the official language of Android)

**AVD Manager**

The AVD Manager tool is bundled with Android Studio. AVD stands for ‘Android Virtual Device’, so essentially this is an emulator for running Android applications on your PC. This is useful because it means that you can test your apps quickly without having to constantly install them on physical devices. More importantly, the AVD Manager allows you to create lots of different emulators with different screen sizes, specifications and versions of Android.

**Android Device Monitor**

Another built-in Android development tool, the Android Device Monitor allows you to monitor your device or virtual device during runtime and get access to information such as how many processes are running on what thread, network stats, the LogCat and more. It’s great for testing the performance of your apps and seeing what’s going on under the.

**Android Debug Bridge**

The ADB shell is a useful little command-line tool that you can use to communicate with or run commands on a connected Android device (virtual or physical). It comes with Android Studio and for the most part you won’t need to worry about it.

**Activities**

Activity processes have application components the user is interacting with. Its represents the presentation layer of an Android application. Its presents screen in your application.

**Services**

Android offers the Service class to create application components that handle long-lived operation and include functionality that doesn't require a user interface. Services are started, stopped and controlled from other application components, including Activities, Broadcast Receivers, and other Services.

**Intent**

Intent are used as a message-passing mechanism that both within your application and between application

● Starts a particular Service are Activity using a class name.

● Start an Activity or Service to platform an action with a particular piece of data.

● Broadcast that an event has occurred

**BroadcastReceiver**

Broadcast Intents are used to notify application of system or application events, extending the event-driven programming model between applications. Within your application, construct the intent you want to broadcast and call send Broadcast to send it.

**Content Providers**

Content Providers provide a interface for publishing and the content schema. Content Providers can be shared between applications, queried for results, have their existing record deleted or updated, and record added.

**Object-Oriented Programming**

Object-Oriented Programming is a methodology or paradigm to design a program using classes and objects. It simplifies the software development and maintenance by providing some concepts:

**Object**

Any entity that has state and behavior is known as an object. For example: chair, pen, table, keyboard, bike etc. It can be physical and logical.

**Class**

Collection of objects is called class. It is a logical entity.

**Inheritance**

When one object acquires all the properties and behaviours of parent object i.e. known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.

**Polymorphism**

When one task is performed by different ways i.e. known as polymorphism. For example: to convince the customer differently, to draw something e.g. shape or rectangle etc.

There are two types of polymorphism in java:

**1) Static Polymorphism also known as compiles time polymorphism:**

Polymorphism that is resolved during compiler time is known as static polymorphism.

**2) Dynamic Polymorphism also known as runtime polymorphism:**

It is also known as Dynamic Method Dispatch. Dynamic polymorphism is a process in which a call t

**Abstraction**

Hiding internal details and showing functionality is known as abstraction. For example: phone call, we don't know the internal processing.

**Encapsulation**

Binding (or wrapping) code and data together into a single unit is known as encapsulation. For example: capsule, it is wrapped with different medicines. A java class is the example of encapsulation. Java bean is the fully encapsulated class because all the data members are private here.

**View**

The View class is a superclass for all GUI components in Android. For instance, the TextView class which is used to display text labels in Android apps is a subclass of View. Android contains the following commonly used View subclasses:

● TextView ● EditText ● ImageView ● ProgressBar

● Button ● ImageButton ● CheckBox ● DatePicker

**ViewGroup**

The ViewGroup class is a subclass of the View class. ViewGroup instances work as containers for Viewinstances to group View instances together. Android contains the following commonly used ViewGroupsubclasses:

● LinearLayout ● RelativeLayout ● ListView ● GridView

Introducing Layouts There are many types of layout. Some of which are listed below –

● Linear Layout ● Absolute Layout ● Table Layout ● Frame Layout ● Relative Layout

**Linear Layout**

layout is further divided into horizontal and vertical layout. It means it can arrange views in a single column or in a single row.

**Absolute Layout**

The AbsoluteLayout enables you to specify the exact location of its children.

**Table Layout**

The TableLayout groups views into rows and columns.

**Relative Layout**

The RelativeLayout enables you to specify how child views are positioned relative to each other

**Frame Layout**

The FrameLayout is a placeholder on screen that you can use to display a single view.

**Activity:**

An activity represents a single screen with a user interface just like window or frame of Java.Android activity is the subclass of ContextThemeWrapper class. The Activity class defines the following call backs i.e. events.

**onCreate():**

Called when the activity is first created. This is where you should do all of your normal static set up: create views, bind data to lists, etc. This method also provides you with a Bundle containing the activity's previously frozen state, if there was one. Always followed by onStart().

**onStart():**

Called when the activity is becoming visible to the user.Followed by onResume() if the activity comes to the foreground, or onStop() if it becomes hidden.

**onResume():**

Called when the activity will start interacting with the user. At this point your activity is at the top of the activity stack, with user input going to it. Always followed by onPause().

**onPause():**

Called when the system is about to start resuming a previous activity. This is typically used to commit unsaved changes to persistent data, stop animations and other things that may be consuming CPU, etc. Implementations of this method must be very quick because the next activity will not be resumed until this method returns. Followed by either onResume() if the activity returns back to the front, or onStop() if it becomes invisible to the user.

**onRestart()**

Called after your activity has been stopped, prior to it being started again. Always followed by onStart()

**onStop()**

Called when the activity is no longer visible to the user, because another activity has been resumed and is covering this one. This may happen either because a new activity is being started, an existing one is being brought in front of this one, or this one is being destroyed. Followed by either onRestart() if this activity is coming back to interact with the user, or onDestroy() if this activity is going away.

**onDestroy()**

The final call you receive before your activity is destroyed. This can happen either because the activity is finishing (someone called finish() on it, or because the system is temporarily destroying this instance of the activity to save space. You can distinguish between these two scenarios with the isfinishing() method.

**Intent:**

Intents are asynchronous messages which allow application components to request functionality from other Android components. Intents allow you to interact with components from the same applications as well as with components contributed by other applications. For example, an activity can start an external activity for taking a picture.

There are 2 types of intent:

1)Implicit Intent 2)Explicit Intent

**1)Implicit Intent**: Implicit intents specify the action which should be performed and optionally data which provides content for the action. If an implicit intent is sent to the Android system, it searches for all components which are registered for the specific action and the fitting data type. If only one component is found, Android starts this component directly. If several components are identified by the Android system, the user will get a selection dialog and can decide which component should be used for the intent.

**2)Explicit Intent**: Explicit intents explicitly define the component which should be called by the Android system, by using the Java class as identifier. Explicit intents are typically used within an application as the classes in an application are controlled by the application developer.

**Intent Filter:**

Android OS uses filters to pinpoint the set of Activities, Services, and Broadcast receivers that can handle the Intent with help of specified set of action, categories, data scheme associated with Intent. You will use element in the manifest file to list down actions, categories and data types associated with any activity, service, or broadcast receiver.

**Fragment:**

A Fragment represents a behavior or a portion of user interface in an Activity. You can combine multiple fragments in a single activity to build a multi-pane UI and reuse a fragment in multiple activities. You can think of a fragment as a modular section of an activity, which has its own lifecycle, receives its own input events, and which you can add or remove while the activity is running

Below are the methods of fragment lifecycle.

**1. onAttach()** :This method will be called first, even before onCreate(), letting us know that your fragment has been attached to an activity. You are passed the Activity that will host your fragment

**2. onCreateView() :** The system calls this callback when it’s time for the fragment to draw its UI for the first time. To draw a UI for the fragment, a View component must be returned from this method which is the root of the fragment’s layout. We can return null if the fragment does not provide a UI

**3. onViewCreated()** : This will be called after onCreateView(). This is particularly useful when inheriting the onCreateView() implementation but we need to configure the resulting views, such as with a ListFragment and when to set up an adapter

**4. onActivityCreated()** :This will be called after onCreate() and onCreateView(), to indicate that the activity’s onCreate() has completed. If there is something that’s needed to be initialised in the fragment that depends upon the activity’s onCreate() having completed its work then onActivityCreated() can be used for that initialisation work

**5. onStart() :** The onStart() method is called once the fragment gets visible

**6. onPause() :** The system calls this method as the first indication that the user is leaving the fragment. This is usually where you should commit any changes that should be persisted beyond the current user session

**7. onStop()** : Fragment going to be stopped by calling onStop()

**8. onDestroyView()** : It’s called before onDestroy(). This is the counterpart to onCreateView() where we set up the UI. If there are things that are needed to be cleaned up specific to the UI, then that logic can be put up in onDestroyView()

**9. onDestroy() :** onDestroy() called to do final clean up of the fragment’s state but Not guaranteed to be called by the Android platform.

**10. onDetach() :** It’s called after onDestroy(), to notify that the fragment has been disassociated from its hosting activity

**Sqlite:**

SQLite is an opensource SQL database that stores data to a text file on a device. Android comes in with built in SQLite database implementation.SQLite supports all the relational database features.

**Database - Helper class**

For managing all the operations related to the database, an helper class has been given and is called SQLiteOpenHelper. It automatically manages the creation and update of the database.

Cursor : Query in android will return as a “Cursor” objects. Cursors are pointers to the result set within the underlying data. Following are some of functions in “Cursor” class, which helps navigation within result set.

● **moveToFirst** Moves the cursor to the first row in the query result

● **moveToNext** Moves the cursor to the next row

● **moveToPrevious** Moves the cursor to the previous row

● **getCount** Returns the number of rows in the result set

● **getColumnName** Returns the name of the specified column index

**● getColumnNames** Returns a string array of all the column names in the current Cursor

**● moveToPosition** Moves the Cursor to the specified row

**● getPosition** Returns the current Cursor position

**Content Provider**:

Store and retrieve data and make it available to all applications

– Only way to share data across applications

• Standard content providers part of Android:

– Common data types (audio, video, images, personal contact information)

• Applications can create their own content providers to make their data public

– Alternatively add the data to an existing provider

• Implement a common interface for querying the provider, adding, altering and deleting data • Actual storage of data is up to the designer

• Provides a clean separation between the application layer and data layer

**There are two way to access the database;**

1) Access from cloud or 2) Embedded database, which is stored on mobile device. Embedded database allow your application to be fully functional even when no network connection is available and to sync with the remote database periodically or when going back on-line.