**Introduction to Mobile Application Development**

**Introduction to Mobile Computing :-**

Mobile:-“able to move frelly or ” or “able or willing to move freely or easily between ocuupations ,place of residence and social classes”

Definition and general overview of Mobile and Cell Phone Technologies .

CDMA (Code-Division Multiple Access) refers to any of several protocols used in second-generation (2G) and third-generation ([3G](http://searchtelecom.techtarget.com/definition/3G)) [wireless](http://searchmobilecomputing.techtarget.com/definition/wireless) communications. As the term implies, CDMA is a form of [multiplexing](http://searchnetworking.techtarget.com/definition/multiplexing), which allows numerous signals to occupy a single transmission [channel](http://searchdatacenter.techtarget.com/definition/channel), optimizing the use of available [bandwidth](http://searchenterprisewan.techtarget.com/definition/bandwidth). The technology is used in ultra-high-frequency ([UHF](http://searchnetworking.techtarget.com/definition/UHF)) [cellular telephone](http://searchmobilecomputing.techtarget.com/definition/cellular-telephone) systems in the 800-[MHz](http://searchnetworking.techtarget.com/definition/MHz) and 1.9-GHz bands.

**GSM** (**Global System for Mobile Communications**, originally ***Groupe Spécial Mobile***) is a standard developed by the [European Telecommunications Standards Institute](https://en.wikipedia.org/wiki/European_Telecommunications_Standards_Institute) (ETSI) to describe the protocols for second-generation digital [cellular networks](https://en.wikipedia.org/wiki/Cellular_network) used by [mobile devices](https://en.wikipedia.org/wiki/Mobile_devices) such as [mobile phones](https://en.wikipedia.org/wiki/Mobile_phone), first deployed in Finland in December 1991.[[2]](https://en.wikipedia.org/wiki/GSM#cite_note-2) As of 2014[[update]](https://en.wikipedia.org/w/index.php?title=GSM&action=edit), it has become the global standard for mobile communications – with over 90% market share, operating in over 219 countries and territories.

2G networks developed as a replacement for first generation ([1G](https://en.wikipedia.org/wiki/1G)) analog cellular networks, and the GSM standard originally described as a digital, circuit-switched network optimized for [full duplex](https://en.wikipedia.org/wiki/Duplex_(telecommunications)#Full_duplex) voice [telephony](https://en.wikipedia.org/wiki/Telephony). This expanded over time to include data communications, first by circuit-switched transport, then by [packet](https://en.wikipedia.org/wiki/Network_packet) data transport via [GPRS](https://en.wikipedia.org/wiki/GPRS) (General Packet Radio Services) and [EDGE](https://en.wikipedia.org/wiki/EDGE) (Enhanced Data rates for GSM Evolution, or EGPRS).

Subsequently, the [3GPP](https://en.wikipedia.org/wiki/3GPP) developed third-generation ([3G](https://en.wikipedia.org/wiki/3G)) [UMTS](https://en.wikipedia.org/wiki/UMTS) standards, followed by fourth-generation ([4G](https://en.wikipedia.org/wiki/4G)) [LTE Advanced](https://en.wikipedia.org/wiki/LTE_Advanced) standards, which do not form part of the ETSI GSM standard.

**3G**, stands for **third generation**, is the third generation of [wireless](https://en.wikipedia.org/wiki/Wireless) mobile telecommunications technology. This is based on a set of standards used for mobile devices and mobile telecommunications use services and networks that comply with the International Mobile Telecommunications-2000 (IMT-2000) specifications by the [International Telecommunication Union](https://en.wikipedia.org/wiki/International_Telecommunication_Union). 3G finds application in wireless voice telephony, [mobile Internet](https://en.wikipedia.org/wiki/Mobile_Internet) access, fixed wireless Internet access, video calls and mobile TV.

3G telecommunication networks support services that provide an information transfer rate of at least 2 [Mbit/s](https://en.wikipedia.org/wiki/Megabits_per_second). Later 3G releases, often denoted [3.5G](https://en.wikipedia.org/wiki/3.5G) and [3.75G](https://en.wikipedia.org/wiki/3.75G), also provide [mobile broadband](https://en.wikipedia.org/wiki/Mobile_broadband) access of several [Mbit/s](https://en.wikipedia.org/wiki/Mbps) to [smartphones](https://en.wikipedia.org/wiki/Smartphone) and [mobile modems](https://en.wikipedia.org/wiki/Mobile_modem) in laptop computers. This ensures it can be applied to wireless voice telephony, mobile Internet access, fixed wireless Internet access, video calls and mobile TV technologies.

A new generation of cellular standards has appeared approximately every tenth year since [1G](https://en.wikipedia.org/wiki/1G) systems were introduced in 1981/1982. Each generation is characterized by new frequency bands, higher data rates and non–backward-compatible transmission technology. The first 3G networks were introduced in 1998 and fourth generation [4G](https://en.wikipedia.org/wiki/4G) networks in 2008.

**4G** is the fourth generation of [mobile telecommunications](https://en.wikipedia.org/wiki/Mobile_telephony) technology, succeeding [3G](https://en.wikipedia.org/wiki/3G). Potential and current applications include amended [mobile web](https://en.wikipedia.org/wiki/Mobile_web) access, [IP telephony](https://en.wikipedia.org/wiki/IP_telephony), gaming services, [high-definition](https://en.wikipedia.org/wiki/HDTV) [mobile TV](https://en.wikipedia.org/wiki/Mobile_TV), [video conferencing](https://en.wikipedia.org/wiki/Video_conferencing), and [3D television](https://en.wikipedia.org/wiki/3D_television).

The first-release [Long Term Evolution](https://en.wikipedia.org/wiki/LTE_(telecommunication)) (LTE) standard (a 4G candidate system) has been commercially deployed in Oslo, Norway, and Stockholm, Sweden since 2009. It has, however, been debated whether first-release versions should be considered 4G.

In March 2008, the [International Telecommunications Union-Radio communications sector](https://en.wikipedia.org/wiki/ITU-R) (ITU-R) specified a set of requirements for 4G standards, named the International Mobile Telecommunications Advanced (IMT-Advanced) specification, setting peak speed requirements for 4G service at 100 [megabits per second](https://en.wikipedia.org/wiki/Megabits_per_second) (Mbit/s) for high mobility communication (such as from trains and cars) and 1 [gigabit per second](https://en.wikipedia.org/wiki/Gigabit_per_second) (Gbit/s) for low mobility communication (such as pedestrians and stationary users). Since the first-release versions of [Mobile WiMAX](https://en.wikipedia.org/wiki/Mobile_WiMAX) and [LTE](https://en.wikipedia.org/wiki/Long_Term_Evolution) support much less than 1 Gbit/s peak bit rate.

Types of mobile computing devices

A **personal digital assistant** (**PDA**), also known as a [**handheld PC**](https://en.wikipedia.org/wiki/Handheld_PC)**,**[[1]](https://en.wikipedia.org/wiki/Personal_digital_assistant#cite_note-1)[[2]](https://en.wikipedia.org/wiki/Personal_digital_assistant#cite_note-2) is a [mobile device](https://en.wikipedia.org/wiki/Mobile_device) that functions as a [personal information manager](https://en.wikipedia.org/wiki/Personal_information_manager). PDAs were largely discontinued in the early 2010s after the widespread adoption of highly capable [smartphones](https://en.wikipedia.org/wiki/Smartphone), in particular those based on [iOS](https://en.wikipedia.org/wiki/IOS) and [Android](https://en.wikipedia.org/wiki/Android_(operating_system)).

Nearly all PDAs have the ability to connect to the [Internet](https://en.wikipedia.org/wiki/Internet). A PDA has an [electronic visual display](https://en.wikipedia.org/wiki/Electronic_visual_display), letting it include a [web browser](https://en.wikipedia.org/wiki/Web_browser). All models also have audio capabilities, allowing usage as a [portable media player](https://en.wikipedia.org/wiki/Portable_media_player), and also enabling most of them to be used as [mobile phones](https://en.wikipedia.org/wiki/Mobile_phone). Most PDAs can access the Internet, [intranets](https://en.wikipedia.org/wiki/Intranet) or [extranets](https://en.wikipedia.org/wiki/Extranet) via [Wi-Fi](https://en.wikipedia.org/wiki/Wi-Fi) or [Wireless Wide Area Networks](https://en.wikipedia.org/wiki/Wireless_Wide_Area_Network). Most PDAs employ [touchscreen](https://en.wikipedia.org/wiki/Touchscreen) technology.

The first PDA, the [Organizer](https://en.wikipedia.org/wiki/Psion_Organiser), was released in 1984 by [Psion](https://en.wikipedia.org/wiki/Psion_(computers)), followed by [Psion's Series 3](https://en.wikipedia.org/wiki/Psion_Series_3), in 1991. The latter began to resemble the more familiar PDA style, including a full keyboard.[[4]](https://en.wikipedia.org/wiki/Personal_digital_assistant#cite_note-4)[[5]](https://en.wikipedia.org/wiki/Personal_digital_assistant#cite_note-5) The term *PDA* was first used on January 7, 1992 by [Apple Computer](https://en.wikipedia.org/wiki/Apple_Computer) [CEO](https://en.wikipedia.org/wiki/CEO) [John Sculley](https://en.wikipedia.org/wiki/John_Sculley) at the [Consumer Electronics Show](https://en.wikipedia.org/wiki/Consumer_Electronics_Show) in [Las Vegas](https://en.wikipedia.org/wiki/Las_Vegas_Valley), [Nevada](https://en.wikipedia.org/wiki/Nevada), referring to the [Apple Newton](https://en.wikipedia.org/wiki/Apple_Newton).[[6]](https://en.wikipedia.org/wiki/Personal_digital_assistant#cite_note-6) In 1994, [IBM](https://en.wikipedia.org/wiki/IBM) introduced the first PDA with full mobile phone functionality, the [IBM Simon](https://en.wikipedia.org/wiki/IBM_Simon), which can also be considered the first smartphone. Then in 1996, [Nokia](https://en.wikipedia.org/wiki/Nokia) introduced a PDA with full mobile phone functionality, the [9000 Communicator](https://en.wikipedia.org/wiki/Nokia_9000_Communicator), which became the world's best-selling PDA. The Communicator spawned a new category of PDAs: the "PDA phone", now called "smartphone". Another early entrant in this market was [Palm](https://en.wikipedia.org/wiki/Palm,_Inc.), with a [line of PDA products](https://en.wikipedia.org/wiki/Palm_(PDA)) which began in March 1996. The terms "personal digital assistant" and "PDA" apply to smartphones but are not used in marketing, media, or general conversation to refer to devices such as the [BlackBerry](https://en.wikipedia.org/wiki/BlackBerry), [iPad](https://en.wikipedia.org/wiki/IPad), [iPhone](https://en.wikipedia.org/wiki/IPhone) or [Android](https://en.wikipedia.org/wiki/Android_(operating_system)) devices.

**Pager** (also known as a **beeper**) is a wireless [telecommunications](https://en.wikipedia.org/wiki/Telecommunication) device that receives and displays alphanumeric messages and/or receives and announces voice messages. **One-way pagers** can only receive messages, while **response pagers** and **two-way pagers** can also acknowledge, reply to, and originate messages using an internal transmitter.

Pagers operate as part of a paging system which includes one or more fixed transmitters (or in the case of response pagers and two-way pagers, one or more [base stations](https://en.wikipedia.org/wiki/Base_transceiver_station)), as well as a number of pagers carried by mobile users. These systems can range from a restaurant system with a single low-power transmitter, to a nationwide system with thousands of high-power base stations.

Pagers were developed in the 1950s and 1960s, and became widely used by the 1980s. In the 2000s, the widespread availability of [cellphones](https://en.wikipedia.org/wiki/Cellphone) and [smartphones](https://en.wikipedia.org/wiki/Smartphone) has greatly diminished the pager industry. Nevertheless, pagers continue to be used by some emergency services and public safety personnel, because modern pager systems' coverage overlap, combined with use of satellite communications, can make paging systems more reliable than terrestrial based cellular networks in some cases, including during natural and man-made disaster.[[2]](https://en.wikipedia.org/wiki/Pager#cite_note-2) This resilience has led public safety agencies to adopt pagers over cellular and other commercial services for critical messaging.

**Mobile phone** (also known as a **wireless phone**, **cell phone**, or **cellular telephone**) is a small portable [radio](https://simple.wikipedia.org/wiki/Radio) [telephone](https://simple.wikipedia.org/wiki/Telephone).

The mobile phone can be used to communicate over long distances without wires. It works by communicating with a nearby base station (also called a "cell site") which connects it to the main phone network. When moving, if the mobile phone gets too far away from the cell it is connected to, that cell sends a message to another cell to tell the new cell to take over the call. This is called a "hand off," and the call continues with the new cell the phone is connected to. The hand-off is done so well and carefully that the user will usually never even know that the call was transferred to another cell.

As mobile phones became more popular, they began to cost less money, and more people could afford them. Monthly plans became available for rates as low as US$30 or US$40 a month. Cell phones have become so cheap to own that they have mostly replaced [pay phones](https://simple.wikipedia.org/wiki/Pay_phone) and phone booths except for [urban areas](https://simple.wikipedia.org/wiki/Urban_area) with many people.

The modern form of mobile phone is called "Smart Phone". It has become very popular. The majority of mobile phones made after 2010 are "[smartphones](https://simple.wikipedia.org/wiki/Smartphone)". They can be used as [computers](https://simple.wikipedia.org/wiki/Computer) as well as making voice calls.

**History of mobile platforms**

**Java Platform, Micro Edition** or **Java ME** is a [computing platform](https://en.wikipedia.org/wiki/Computing_platform) for development and deployment of [portable code](https://en.wikipedia.org/wiki/Porting) for [embedded](https://en.wikipedia.org/wiki/Embedded_system) and [mobile devices](https://en.wikipedia.org/wiki/Mobile_device) (micro-controllers, sensors, gateways, mobile phones, personal digital assistants, TV set-top boxes, printers).[[1]](https://en.wikipedia.org/wiki/Java_Platform,_Micro_Edition#cite_note-1) Java ME was formerly known as **Java 2 Platform, Micro Edition** or **J2ME**.

The platform uses the [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) programming language. It is part of the [Java software-platform](https://en.wikipedia.org/wiki/Java_(software_platform)) family. Java ME was designed by [Sun Microsystems](https://en.wikipedia.org/wiki/Sun_Microsystems), acquired by [Oracle Corporation](https://en.wikipedia.org/wiki/Oracle_Corporation) in 2010; the platform replaceda similar technology, [Persona lJava](https://en.wikipedia.org/wiki/PersonalJava). Originally developed under the [Java Community Process](https://en.wikipedia.org/wiki/Java_Community_Process) as JSR 68, the different flavors of Java ME have evolved in separate JSRs. Oracle provides a [reference implementation](https://en.wikipedia.org/wiki/Reference_implementation) of the specification, but has tended not to provide free binary implementations of its Java ME runtime environment for mobile devices, rather relying on third parties to provide their own.

As of 22 December 2006, the Java ME source code is licensed under the [GNU General Public License](https://en.wikipedia.org/wiki/GNU_General_Public_License), and is released under the project name [phone ME](https://en.wikipedia.org/wiki/PhoneME).

As of 2008, all Java ME platforms are currently restricted to [JRE](https://en.wikipedia.org/wiki/Java_Virtual_Machine) 1.3 features and use that version of the class file format (internally known as version 47.0). Should Oracle ever declare a new round of Java ME configuration versions that support the later class file formats and language features, such as those corresponding to JRE 1.5 or 1.6 (notably, [generics](https://en.wikipedia.org/wiki/Generics_in_Java)), it will entail extra work on the part of all platform vendors to update their JREs.

Java ME devices implement a *profile*. The most common of these are the [Mobile Information Device Profile](https://en.wikipedia.org/wiki/Mobile_Information_Device_Profile) aimed at mobile devices, such as cell phones, and the [Personal Profile](https://en.wikipedia.org/wiki/Personal_Profile) aimed at consumer products and embedded devices like [set-top boxes](https://en.wikipedia.org/wiki/Set-top_box) and PDAs. Profiles are subsets of *configurations*, of which there are currently two: the Connected Limited Device Configuration (CLDC) and the Connected Device Configuration (CDC).[[2]](https://en.wikipedia.org/wiki/Java_Platform,_Micro_Edition#cite_note-2)

There are more than 2.1 billion Java ME enabled mobile phones and PDAs.[[3]](https://en.wikipedia.org/wiki/Java_Platform,_Micro_Edition#cite_note-3) It is popular in sub $200 devices such as Nokia's [Series 40](https://en.wikipedia.org/wiki/Series_40). It was also used on the [Bada](https://en.wikipedia.org/wiki/Bada) operating system and on [Symbian](https://en.wikipedia.org/wiki/Symbian) OS along with native software. Users of [Windows CE](https://en.wikipedia.org/wiki/Windows_CE), [Windows Mobile](https://en.wikipedia.org/wiki/Windows_Mobile), [Maemo](https://en.wikipedia.org/wiki/Maemo), [MeeGo](https://en.wikipedia.org/wiki/MeeGo) and [Android](https://en.wikipedia.org/wiki/Android_software_development) can download Java ME for their respective environments

**Android** is a [mobile operating system](https://en.wikipedia.org/wiki/Mobile_operating_system) developed by [Google](https://en.wikipedia.org/wiki/Google), based on the [Linux kernel](https://en.wikipedia.org/wiki/Linux_kernel) and designed primarily for [touchscreen](https://en.wikipedia.org/wiki/Touchscreen) mobile devices such as [smartphones](https://en.wikipedia.org/wiki/Smartphone) and [tablets](https://en.wikipedia.org/wiki/Tablet_computer). Android's [user interface](https://en.wikipedia.org/wiki/User_interface) is mainly based on [direct manipulation](https://en.wikipedia.org/wiki/Direct_manipulation_interface), using touch gestures that loosely correspond to real-world actions, such as swiping, tapping and pinching, to manipulate on-screen objects, along with a [virtual keyboard](https://en.wikipedia.org/wiki/Virtual_keyboard) for text input. In addition to touchscreen devices, Google has further developed [Android TV](https://en.wikipedia.org/wiki/Android_TV) for televisions, [Android Auto](https://en.wikipedia.org/wiki/Android_Auto) for cars, and [Android Wear](https://en.wikipedia.org/wiki/Android_Wear) for wrist watches, each with a specialized user interface. Variants of Android are also used on [game consoles](https://en.wikipedia.org/wiki/Video_game_console), [digital cameras](https://en.wikipedia.org/wiki/Digital_camera), [PCs](https://en.wikipedia.org/wiki/Personal_computer) and other electronics.

Initially developed by Android Inc., which Google bought in 2005, Android was unveiled in 2007, along with the founding of the [Open Handset Alliance](https://en.wikipedia.org/wiki/Open_Handset_Alliance) – a consortium of [hardware](https://en.wikipedia.org/wiki/Computer_hardware), [software](https://en.wikipedia.org/wiki/Software), and telecommunication companies devoted to advancing [open standards](https://en.wikipedia.org/wiki/Open_standard) for mobile devices. Beginning with the [first commercial Android device](https://en.wikipedia.org/wiki/HTC_Dream) in September 2008, the operating system has gone through multiple major releases, with the current version being [8.0 "Oreo"](https://en.wikipedia.org/wiki/Android_Oreo), released in August 2017. Android applications ("[apps](https://en.wikipedia.org/wiki/Mobile_app)") can be downloaded from the [Google Play](https://en.wikipedia.org/wiki/Google_Play) store, which features over 2.7 million apps as of February 2017. Android has been the best-selling OS on tablets since 2013, and runs on the vast majority[[a]](https://en.wikipedia.org/wiki/Android_(operating_system)#cite_note-11) of smartphones. As of May 2017[[update]](https://en.wikipedia.org/w/index.php?title=Android_(operating_system)&action=edit), Android has two billion monthly active users, and it has the largest [installed base](https://en.wikipedia.org/wiki/Installed_base) of any operating system.

**Windows Mobile** is a family of [mobile operating systems](https://en.wikipedia.org/wiki/Mobile_operating_system) developed by [Microsoft](https://en.wikipedia.org/wiki/Microsoft) for [smartphones](https://en.wikipedia.org/wiki/Smartphone) and [Pocket PCs](https://en.wikipedia.org/wiki/Pocket_PC).[[1]](https://en.wikipedia.org/wiki/Windows_Mobile#cite_note-infoworld2005-1)

Its origins dated back to [Windows CE](https://en.wikipedia.org/wiki/Windows_CE) in 1996, though Windows Mobile itself first appeared in 2000 as *PocketPC 2000*. It was renamed "Windows Mobile" in 2003, at which point it came in several versions (similar to the desktop versions of Windows) and was aimed at business and enterprise consumers. By 2007, it was the most popular smartphone software in the U.S., but this popularity faded in the following years. In February 2010, facing competition from rival OSs including [iOS](https://en.wikipedia.org/wiki/IOS) and [Android](https://en.wikipedia.org/wiki/Android_(operating_system)), Microsoft announced [Windows Phone](https://en.wikipedia.org/wiki/Windows_Phone) to supersede Windows Mobile. As a result, Windows Mobile has been [deprecated](https://en.wikipedia.org/wiki/Deprecated). Windows Phone is incompatible with Windows Mobile devices and software. The last version of Windows Mobile, released after the announcement of Windows Phone, was 6.5.5. After this, Microsoft ceased development on Windows Mobile, in order to concentrate on Windows Phone.

**Windows Phone** (**WP**) is a family of [mobile operating systems](https://en.wikipedia.org/wiki/Mobile_operating_system) developed by [Microsoft](https://en.wikipedia.org/wiki/Microsoft) for [smartphones](https://en.wikipedia.org/wiki/Smartphones) as the replacement successor to [Windows Mobile](https://en.wikipedia.org/wiki/Windows_Mobile) and [Zune](https://en.wikipedia.org/wiki/Zune). Windows Phone features a new user interface derived from [Metro design language](https://en.wikipedia.org/wiki/Metro_design_language). Unlike Windows Mobile, it is primarily aimed at the consumer market rather than the [enterprise](https://en.wikipedia.org/wiki/Enterprise_software) market. It was first launched in October 2010 with [Windows Phone 7](https://en.wikipedia.org/wiki/Windows_Phone_7). [Windows Phone 8.1](https://en.wikipedia.org/wiki/Windows_Phone_8.1) is the latest public release of the operating system, released to manufacturing on April 14, 2014.

Windows Phone was replaced by [Windows 10 Mobile](https://en.wikipedia.org/wiki/Windows_10_Mobile) in 2015; it emphasizes a larger amount of integration and unification with its [PC counterpart](https://en.wikipedia.org/wiki/Windows_10)—including a new, unified application ecosystem, along with an expansion of its scope to include small-screened [tablets](https://en.wikipedia.org/wiki/Tablet_computer).

Work on a major [Windows Mobile](https://en.wikipedia.org/wiki/Windows_Mobile) update may have begun as early as 2004 under the codename "Photon", but work moved slowly and the project was ultimately cancelled.[[14]](https://en.wikipedia.org/wiki/Windows_Phone#cite_note-14) In 2008, Microsoft reorganized the Windows Mobile group and started work on a new mobile operating system.[[15]](https://en.wikipedia.org/wiki/Windows_Phone#cite_note-15) The product was to be released in 2009 as Windows Phone, but several delays prompted Microsoft to develop Windows Mobile 6.5 as an interim release.

Windows Phone was developed quickly. One result was that the new OS would not be compatible with Windows Mobile applications. Larry Lieberman, senior product manager for Microsoft's Mobile Developer Experience, told [eWeek](https://en.wikipedia.org/wiki/EWeek): "If we'd had more time and resources, we may have been able to do something in terms of backward compatibility." Lieberman said that Microsoft was attempting to look at the mobile phone market in a new way, with the [end user](https://en.wikipedia.org/wiki/End_user) in mind as well as the enterprise network.[Terry Myerson](https://en.wikipedia.org/wiki/Terry_Myerson), corporate VP of Windows Phone engineering, said, "With the move to capacitive touch screens, away from the stylus, and the moves to some of the hardware choices we made for the Windows Phone 7 experience, we had to break application compatibility with Windows Mobile 6.5.

**The Android Platform**

Mobile application development is now quite popular, and Android is one of the most popular mobile device platforms. Android is a Linux based software stack initially developed by Google that is organized into the Android operating system, the middleware components, and the key applications. The design goals of the Android platform include: platform neutrality, open source, flexibility, and rapid development.

Android is an open source, Linux-based software stack created for a wide array of devices and form factors. The following diagram shows the major components of the Android platform.

The Android Architecture Components

## The Linux Kernel

The foundation of the Android platform is the Linux kernel. For example, [the Android Runtime (ART)](https://developer.android.com/guide/platform/index.html#art) relies on the Linux kernel for underlying functionalities such as threading and low-level memory management.

Using a Linux kernel allows Android to take advantage of [key security features](https://source.android.com/security/overview/kernel-security.html) and allows device manufacturers to develop hardware drivers for a well-known kernel.

## Hardware Abstraction Layer (HAL)

The [hardware abstraction layer (HAL)](https://source.android.com/devices/index.html#Hardware%20Abstraction%20Layer) provides standard interfaces that expose device hardware capabilities to the higher-level [Java API framework](https://developer.android.com/guide/platform/index.html#api-framework). The HAL consists of multiple library modules, each of which implements an interface for a specific type of hardware component, such as the [camera](https://source.android.com/devices/camera/index.html) or [bluetooth](https://source.android.com/devices/bluetooth.html) module. When a framework API makes a call to access device hardware, the Android system loads the library module for that hardware component.

## Android Runtime

For devices running Android version 5.0 (API level 21) or higher, each app runs in its own process and with its own instance of the [Android Runtime (ART)](http://source.android.com/devices/tech/dalvik/index.html). ART is written to run multiple virtual machines on low-memory devices by executing DEX files, a bytecode format designed specially for Android that's optimized for minimal memory footprint. Build toolchains, such as [Jack](https://source.android.com/source/jack.html), compile Java sources into DEX bytecode, which can run on the Android platform.

Some of the major features of ART include the following:

* Ahead-of-time (AOT) and just-in-time (JIT) compilation
* Optimized garbage collection (GC)
* Better debugging support, including a dedicated sampling profiler, detailed diagnostic exceptions and crash reporting, and the ability to set watchpoints to monitor specific fields

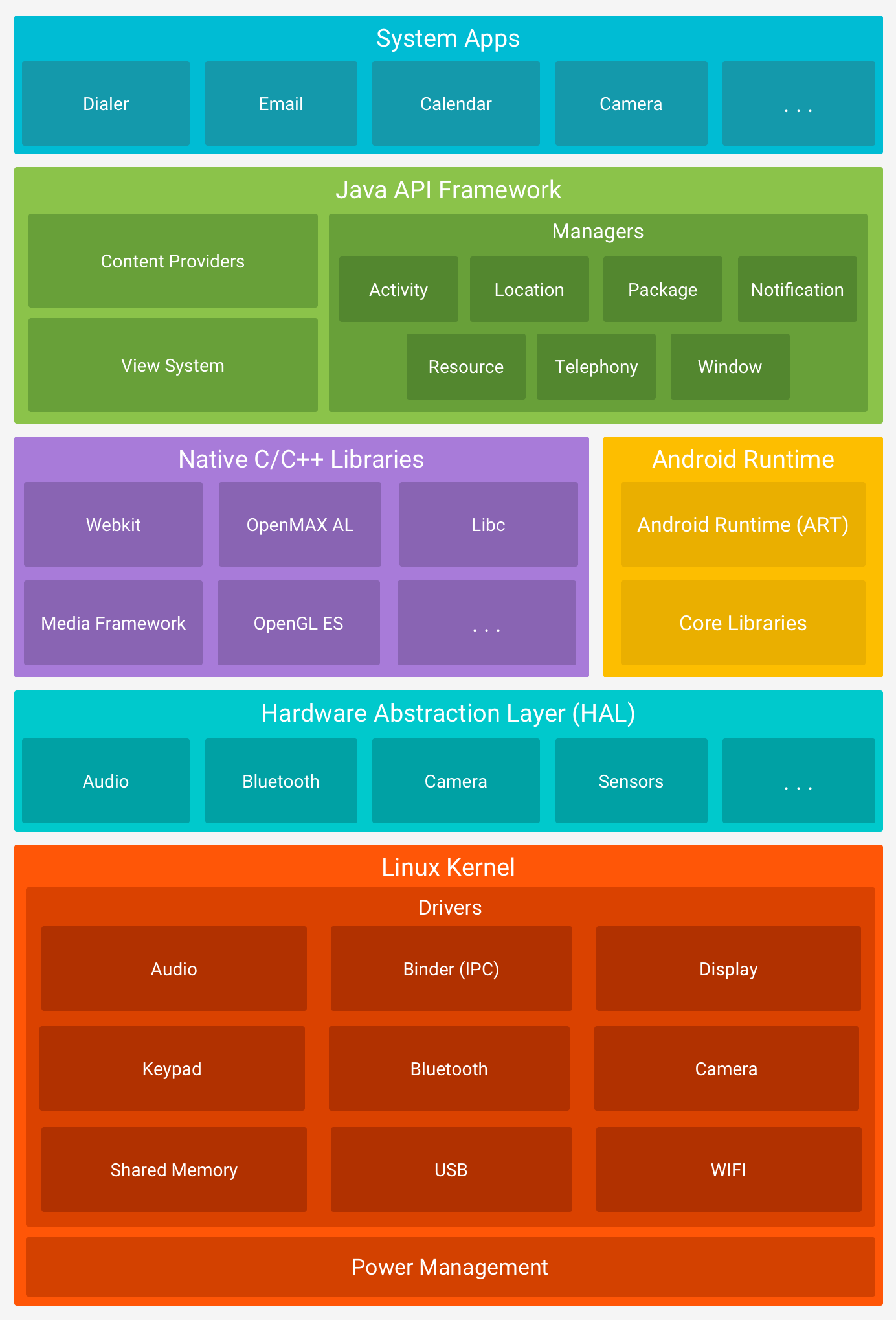
Prior to Android version 5.0 (API level 21), Dalvik was the Android runtime. If your app runs well on ART, then it should work on Dalvik as well, but [the reverse may not be true](https://developer.android.com/guide/practices/verifying-apps-art.html).

Android also includes a set of core runtime libraries that provide most of the functionality of the Java programming language, including some [Java 8 language features](https://developer.android.com/guide/platform/j8-jack.html), that the Java API framework uses.

## Native C/C++ Libraries

Many core Android system components and services, such as ART and HAL, are built from native code that require native libraries written in C and C++. The Android platform provides Java framework APIs to expose the functionality of some of these native libraries to apps. For example, you can access [OpenGL ES](https://developer.android.com/guide/topics/graphics/opengl.html) through the Android framework’s [Java OpenGL API](https://developer.android.com/reference/android/opengl/package-summary.html) to add support for drawing and manipulating 2D and 3D graphics in your app.

If you are developing an app that requires C or C++ code, you can use the [Android NDK](https://developer.android.com/ndk/index.html) to access some of these [native platform libraries](https://developer.android.com/ndk/guides/stable_apis.html) directly from your native code.



## Java API Framework

The entire feature-set of the Android OS is available to you through APIs written in the Java language. These APIs form the building blocks you need to create Android apps by simplifying the reuse of core, modular system components and services, which include the following:

* A rich and extensible [View System](https://developer.android.com/guide/topics/ui/overview.html) you can use to build an app’s UI, including lists, grids, text boxes, buttons, and even an embeddable web browser
* A [Resource Manager](https://developer.android.com/guide/topics/resources/overview.html), providing access to non-code resources such as localized strings, graphics, and layout files
* A [Notification Manager](https://developer.android.com/guide/topics/ui/notifiers/notifications.html) that enables all apps to display custom alerts in the status bar
* An [Activity Manager](https://developer.android.com/guide/components/activities.html) that manages the lifecycle of apps and provides a common [navigation back stack](https://developer.android.com/guide/components/tasks-and-back-stack.html)
* [Content Providers](https://developer.android.com/guide/topics/providers/content-providers.html) that enable apps to access data from other apps, such as the Contacts app, or to share their own data

Developers have full access to the same [framework APIs](https://developer.android.com/reference/packages.html) that Android system apps use.

## System Apps

Android comes with a set of core apps for email, SMS messaging, calendars, internet browsing, contacts, and more. Apps included with the platform have no special status among the apps the user chooses to install. So a third-party app can become the user's default web browser, SMS messenger, or even the default keyboard (some exceptions apply, such as the system's Settings app).

The system apps function both as apps for users and to provide key capabilities that developers can access from their own app. For example, if your app would like to deliver an SMS message, you don't need to build that functionality yourself—you can instead invoke whichever SMS app is already installed to deliver a message to the recipient you specify.

A **software development kit** (**SDK** or **devkit**) is typically a set of [software development](https://en.wikipedia.org/wiki/Software_development) tools that allows the creation of [applications](https://en.wikipedia.org/wiki/Application_software) for a certain [software](https://en.wikipedia.org/wiki/Software) package, [software framework](https://en.wikipedia.org/wiki/Software_framework), hardware platform, [computer system](https://en.wikipedia.org/wiki/Computer_system), [video game console](https://en.wikipedia.org/wiki/Video_game_console), [operating system](https://en.wikipedia.org/wiki/Operating_system), or similar development platform. To enrich applications with advanced functionalities, advertisements, push notifications and more, most app developers implement specific software development kits. Some SDKs are critical for developing a platform-specific app. For example, the development of an [Android](https://en.wikipedia.org/wiki/Android_(operating_system)) app on [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) platform requires a [Java Development Kit](https://en.wikipedia.org/wiki/Java_Development_Kit), for [iOS](https://en.wikipedia.org/wiki/IOS) apps the [iOS SDK](https://en.wikipedia.org/wiki/IOS_SDK), and for [Universal Windows Platform](https://en.wikipedia.org/wiki/Universal_Windows_Platform) the [.NET Framework SDK](https://en.wikipedia.org/wiki/.NET_Framework_SDK). There are also SDKs that are installed in apps to provide analytics and data about activity. Prominent examples include [Google](https://en.wikipedia.org/wiki/Google), [InMobi](https://en.wikipedia.org/wiki/InMobi) and [Facebook](https://en.wikipedia.org/wiki/Facebook).

It may be something as simple as the implementation of one or more [application programming interfaces](https://en.wikipedia.org/wiki/Application_programming_interface) (APIs) in the form of some [libraries](https://en.wikipedia.org/wiki/Library_(computing)) to interface to a particular [programming language](https://en.wikipedia.org/wiki/Programming_language) or to include sophisticated hardware that can communicate with a particular [embedded system](https://en.wikipedia.org/wiki/Embedded_system). Common [tools](https://en.wikipedia.org/wiki/Software_development_tool) include debugging facilities and other [utilities](https://en.wikipedia.org/wiki/Utility_program), often presented in an [integrated development environment](https://en.wikipedia.org/wiki/Integrated_development_environment) (IDE). SDKs also frequently include sample code and supporting technical notes or other supporting documentation to help clarify points made by the primary reference material.

**Gradle** is an open source [build automation](https://en.wikipedia.org/wiki/Build_automation) system that builds upon the concepts of [Apache Ant](https://en.wikipedia.org/wiki/Apache_Ant) and [Apache Maven](https://en.wikipedia.org/wiki/Apache_Maven) and introduces a [Groovy](https://en.wikipedia.org/wiki/Groovy_(programming_language))-based [domain-specific language](https://en.wikipedia.org/wiki/Domain-specific_language) (DSL) instead of the [XML](https://en.wikipedia.org/wiki/XML) form used by [Apache Maven](https://en.wikipedia.org/wiki/Apache_Maven) for declaring the project configuration.[[2]](https://en.wikipedia.org/wiki/Gradle#cite_note-2) Gradle uses a [directed acyclic graph](https://en.wikipedia.org/wiki/Directed_acyclic_graph) ("DAG") to determine the order in which tasks can be run.

Gradle was designed for multi-project builds which can grow to be quite large, and supports incremental builds by intelligently determining which parts of the build tree are up-to-date, so that any task dependent upon those parts will not need to be re-executed.

The initial plugins are primarily focused around [Java](https://en.wikipedia.org/wiki/Java_(programming_language)),[[3]](https://en.wikipedia.org/wiki/Gradle#cite_note-3) [Groovy](https://en.wikipedia.org/wiki/Groovy_(programming_language)) and [Scala](https://en.wikipedia.org/wiki/Scala_(programming_language)) development and deployment, but more languages and project workflows are on the roadmap.

**Activities and Lifecycle, Fragments and Intents**

Activity

An activity is a single, focused thing that the user can do. Almost all activities interact with the user, so the Activity class takes care of creating a window for you in which you can place your UI with [setContentView(View)](https://developer.android.com/reference/android/app/Activity.html#setContentView(android.view.View)). While activities are often presented to the user as full-screen windows, they can also be used in other ways: as floating windows (via a theme with [windowIsFloating](https://developer.android.com/reference/android/R.attr.html#windowIsFloating) set) or embedded inside of another activity (using [ActivityGroup](https://developer.android.com/reference/android/app/ActivityGroup.html)). There are two methods almost all subclasses of Activity will implement:

### Activity Lifecycle

Activities in the system are managed as an activity stack. When a new activity is started, it is placed on the top of the stack and becomes the running activity -- the previous activity always remains below it in the stack, and will not come to the foreground again until the new activity exits.

An activity has essentially four states:

* If an activity is in the foreground of the screen (at the top of the stack), it is active or running.
* If an activity has lost focus but is still visible (that is, a new non-full-sized or transparent activity has focus on top of your activity), it is paused. A paused activity is completely alive (it maintains all state and member information and remains attached to the window manager), but can be killed by the system in extreme low memory situations.
* If an activity is completely obscured by another activity, it is stopped. It still retains all state and member information, however, it is no longer visible to the user so its window is hidden and it will often be killed by the system when memory is needed elsewhere.
* If an activity is paused or stopped, the system can drop the activity from memory by either asking it to finish, or simply killing its process. When it is displayed again to the user, it must be completely restarted and restored to its previous state.

The following diagram shows the important state paths of an Activity. The square rectangles represent callback methods you can implement to perform operations when the Activity moves between states. The colored ovals are major states the Activity can be in.



There are three key loops you may be interested in monitoring within your activity:

* The **entire lifetime** of an activity happens between the first call to [onCreate(Bundle)](https://developer.android.com/reference/android/app/Activity.html#onCreate(android.os.Bundle)) through to a single final call to [onDestroy()](https://developer.android.com/reference/android/app/Activity.html#onDestroy()). An activity will do all setup of "global" state in onCreate(), and release all remaining resources in onDestroy(). For example, if it has a thread running in the background to download data from the network, it may create that thread in onCreate() and then stop the thread in onDestroy().
* The **visible lifetime** of an activity happens between a call to [onStart()](https://developer.android.com/reference/android/app/Activity.html#onStart()) until a corresponding call to [onStop()](https://developer.android.com/reference/android/app/Activity.html#onStop()). During this time the user can see the activity on-screen, though it may not be in the foreground and interacting with the user. Between these two methods you can maintain resources that are needed to show the activity to the user. For example, you can register a [BroadcastReceiver](https://developer.android.com/reference/android/content/BroadcastReceiver.html) in onStart() to monitor for changes that impact your UI, and unregister it in onStop() when the user no longer sees what you are displaying. The onStart() and onStop() methods can be called multiple times, as the activity becomes visible and hidden to the user.
* The **foreground lifetime** of an activity happens between a call to [onResume()](https://developer.android.com/reference/android/app/Activity.html#onResume()) until a corresponding call to [onPause()](https://developer.android.com/reference/android/app/Activity.html#onPause()). During this time the activity is in front of all other activities and interacting with the user. An activity can frequently go between the resumed and paused states -- for example when the device goes to sleep, when an activity result is delivered, when a new intent is delivered -- so the code in these methods should be fairly lightweight.

EX

Java file

package com.example.hp.testproject;

import android.app.Notification;

import android.app.NotificationManager;

import android.content.Intent;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.Toast;

public class ActivityLifeCycle extends AppCompatActivity {

Button closeButton;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

notify("onCreate");

setContentView(R.layout.activity\_life\_cycle);

closeButton=(Button) findViewById(R.id.closeButton);

closeButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

ActivityLifeCycle.this.finish();

}

});

}

@Override

protected void onPause() {

super.onPause();

notify("onPause");

}

@Override

protected void onResume() {

super.onResume();

notify("onResume");

}

@Override

protected void onStop() {

super.onStop();

notify("onStop");

}

@Override

protected void onDestroy() {

super.onDestroy();

notify("onDestroy");

}

@Override

protected void onRestoreInstanceState(Bundle savedInstanceState) {

super.onRestoreInstanceState(savedInstanceState);

notify("onRestoreInstanceState");

}

@Override

protected void onSaveInstanceState(Bundle outState) {

super.onSaveInstanceState(outState);

notify("onSaveInstanceState");

}

private void notify(String methodName) {

String name = this.getClass().getName();

String[] strings = name.split("\\.");

Toast.makeText(ActivityLifeCycle.this, methodName,

Toast.LENGTH\_LONG).show();

}

}

.xml file

<?xml version="1.0" encoding="utf-8"?>

<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context="com.example.hp.testproject.ActivityLifeCycle">

<Button

android:id="@+id/closeButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Button"

tools:layout\_editor\_absoluteX="147dp"

tools:layout\_editor\_absoluteY="175dp" />

</android.support.constraint.ConstraintLayout>

Manifest file

<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

package="com.example.hp.testproject">

<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />

<application

android:allowBackup="true"

android:icon="@mipmap/ic\_launcher"

android:label="@string/app\_name"

android:roundIcon="@mipmap/ic\_launcher\_round"

android:supportsRtl="true"

android:theme="@style/Theme.AppCompat.Light.NoActionBar">

<activity android:name=".PassingDataThroughIntent">

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

</application>

</manifest>

applying themes and styles

A *style* is a collection of attributes that specify the look and format for a [View](https://developer.android.com/reference/android/view/View.html) or window. A style can specify attributes such as height, padding, font color, font size, background color, and much more. A style is defined in an XML resource that is separate from the XML that specifies the layout.

For example, by using a style, you can take this layout XML:

<TextView  
    android:layout\_width="match\_parent"  
    android:layout\_height="wrap\_content"  
    android:textColor="#00FF00"  
    android:typeface="monospace"  
    android:text="@string/hello" />

And turn it into this:

<TextView  
    android:layout\_width="match\_parent"  
    android:layout\_height="wrap\_content"  
    android:textAppearance="@style/CodeFont"  
    android:text="@string/hello" />

The attributes related to style have been removed from the layout XML and put into a style definition called CodeFont, which is then applied using the android:textAppearance attribute. The definition for this style is covered in the following section.

Defining Styles

To create a set of styles, save an XML file in the res/values/ directory of your project. The name of the XML file must use the .xml extension, and like other resources, it must use lowercase, underscores, and be saved in the res/values/ folder. The root node of the XML file must be <resources>.

For each style you want to create, complete the following series of steps:

1. Add a <style> element to the file, with a name that uniquely identifies the style.
2. For each attribute of that style, add an <item> element, with a name that declares the style attribute. The order of these elements doesn't matter.
3. Add an appropriate value to each <item> element.
4. For example, you can specify the particular value for an android:textColor attribute—in this case a hexadecimal color—or you can specify a reference to a color resource so that you can manage it centrally along with other colors.
5. The following example illustrates using hexadecimal color values in a number of attributes:
6. <resources>  
       <style name="AppTheme" parent="Theme.Material">  
           <item name="colorPrimary">#673AB7</item>  
           <item name="colorPrimaryDark">#512DA8</item>  
           <item name="colorAccent">#FF4081</item>  
       </style>  
   </resources>
7. And the following example illustrates specifying values for the same attribute using references:
8. <resources>  
       <style name="AppTheme" parent="Theme.Material">  
           <item name="colorPrimary">@color/primary</item>  
           <item name="colorPrimaryDark">@color/primary\_dark</item>  
           <item name="colorAccent">@color/accent</item>  
       </style>  
   </resources>
9. You can find information on which resource types can be used with which attributes in the attribute reference, [R.attr](https://developer.android.com/reference/android/R.attr.html). For more information on centrally managing resources, see [Providing Resources](https://developer.android.com/guide/topics/resources/providing-resources.html). For more information on working with color resources, see [More Resource Types](https://developer.android.com/guide/topics/resources/more-resources.html#Color).
10. Here's another example file with a single style:
11. <?xml version="1.0" encoding="utf-8"?>  
    <resources>  
        <style name="CodeFont" parent="@android:style/TextAppearance.Medium">  
            <item name="android:textColor">#00FF00</item>  
            <item name="android:typeface">monospace</item>  
        </style>  
    </resources>
12. This example style can be referenced from an XML layout as @style/CodeFont (as demonstrated in the introduction above).
13. The parent in the <style> element is optional and specifies the resource ID of another style from which this style should inherit attributes. You can then override the inherited style attributes.
14. A style that you want to use as an activity or app theme is defined in XML exactly the same as a style for a view. How to apply a style as an app theme is discussed in [Apply a theme to an activity or app](https://developer.android.com/guide/topics/ui/themes.html#ApplyATheme).

### Inheritance

The parent attribute in the <style> element lets you specify a style from which your style should inherit attributes. You can use this to inherit attributes from an existing style and define only the attributes that you want to change or add. You can inherit from styles that you've created yourself or from styles that are built into the platform. For example, you can inherit the Android platform's default text appearance and modify it:

    <style name="GreenText" parent="@android:style/TextAppearance">  
        <item name="android:textColor">#00FF00</item>  
    </style>

For information about inheriting from styles defined by the Android platform, see [Using Platform Styles and Themes](https://developer.android.com/guide/topics/ui/themes.html#PlatformStyles).

If you want to inherit from styles that you've defined yourself, you don't have to use the parent. Instead, you can use dot notation by prefixing the name of the style you want to inherit to the name of your new style, separated by a period. For example, to create a new style that inherits the CodeFont style defined above, but make the color red, you can create the new style like this:

    <style name="CodeFont.Red">  
        <item name="android:textColor">#FF0000</item>  
    </style>

Notice that there is no parent in the <style> tag, but because the name begins with the CodeFont style name, this new style inherits all style attributes from the CodeFont style. The new style then overrides the android:textColor attribute to make the text red. You can reference this new style as @style/CodeFont.Red.

You can continue inheriting styles like this as many times as you'd like by chaining names with periods. For example, you can extend CodeFont.Red to be bigger, with:

    <style name="CodeFont.Red.Big">  
        <item name="android:textSize">30sp</item>  
    </style>

This style inherits from the CodeFont.Red style, that itself inherits from the CodeFont style, then adds the android:textSize attribute.

**Note:** This technique for inheritance by chaining together names only works for styles defined by your own resources. You can't inherit Android built-in styles this way, as they reside in a different namespace to that used by your resources. To reference a built-in style, such as [TextAppearance](https://developer.android.com/reference/android/R.style.html#TextAppearance), you must use the parent attribute.

Applying Styles and Themes to the UI

* To an individual view, by adding the style attribute to a View element in the XML for your layout.
* To an individual view, by passing the style resource identifier to a [View](https://developer.android.com/reference/android/view/View.html#View(android.content.Context, android.util.AttributeSet, int, int)) constructor. This is available for apps that target Android 5.0 (API level 21) or higher.
* Or, to an entire activity or app, by adding the android:theme attribute to the <activity> or <application> element in the Android manifest.

### Apply a style to a view

Here's how to set a style for a view in the XML layout:

<TextView  
    style="@style/CodeFont"  
    android:text="@string/hello" />

### Apply a theme to an activity or app

To set a theme for all the activities of your app, open the AndroidManifest.xml file and edit the <application> tag to include the android:theme attribute with the style name. For example:

<application android:theme="@style/CustomTheme">

If you want a theme applied to just one activity in your app, then add the android:theme attribute to the <activity> tag instead.

Maintaining theme compatibility

For example, here is the declaration for a custom theme. It would go in an XML file under res/values (typically res/values/styles.xml):

<style name="LightThemeSelector" parent="android:Theme.Light">  
    ...  
</style>

To have this theme use the material theme when the app is running on Android 5.0 (API Level 21) or higher, you can place an alternative declaration for the theme in an XML file in res/values-v21, but make the parent theme the material theme:

<style name="LightThemeSelector" parent="android:Theme.Material.Light">  
    ...  
</style>

Now use this theme like you would any other, and your app automatically switches to the material theme if it's running on Android 5.0 or higher.

filters passing data using objects in intents

|  |  |
| --- | --- |
| Intent | An intent is an abstract description of an operation to be performed. It can be used with **startActivity** to launch an **Activity, broadcastIntent** to send it to any interested **BroadcastReceiver** compone An intent is an abstract description of an operation to be performed. It can be used with **startActivity** to launch an **Activity, broadcastIntent** to send it to any interested **BroadcastReceiver** components, and **startService(Intent)** or **bindService(Intent, ServiceConnection, int)** to communicate with a Background Service.  An Intent provides a facility for performing late runtime binding between the code in different applications. Its most significant use is in the launching of activities, where it can be thought of as the glue between activities. It is basically a passive data structure holding an abstract description of an action to be performed. The primary pieces of information in an intent are:   * **action** The general action to be performed, such as ACTION\_VIEW, ACTION\_EDIT, ACTION\_MAIN, etc. * **data** The data to operate on, such as a person record in the contacts database, expressed as a Uri.   nts, and **startService(Intent)** or **bindService(Intent, ServiceConnection, int)** to communicate with a Background Service.  An Intent provides a facility for performing late runtime binding between the code in different applications. Its most significant use is in the launching of activities, where it can be thought of as the glue between activities. It is basically a passive data structure holding an abstract description of an action to be performed. The primary pieces of information in an intent are:   * **action** The general action to be performed, such as ACTION\_VIEW, ACTION\_EDIT, ACTION\_MAIN, etc. * **data** The data to operate on, such as a person record in the contacts database, expressed as a Uri. |