1.Платформы и инструменты для разработки мобильных приложений. Игровые движки. Кроссплатформенные инструменты. Сравнительная характеристика платформ и инструментов.

### Ответ: Android Studio:

### Комплексная кроссплатформенная среда для [разработки приложений на Android](https://artjoker.ua/ru/uslugi/application-development-for-android/). По функционалу разве что не летает. Помимо инструментов для самого аппа, вы можете рассчитывать на полный набор полезных штук от Google (Аналитика, интеграция с облачными хранилищами и т.д.). Unity

Unity — это самый популярный инструмент для разработки двух- и трёхмерных приложений и игр. На нем делается больше 80% новых мобильных 3D игр. Unreal EngineUnreal Engine 4 — движок нового поколения, который позволяет создавать игры любого жанра ждя любых платформ. Благодаря использованию C++ ваши возможности практически неограниченны!

2.Современное мобильное устройство. Архитектура современного мобильного устройства. Технологии, используемые в современных мобильных устройствах.

Ответ: «Умный телефон» (смартфон), планшетные компьютеры, Ноут- и нетбуки;

По сути, архитектура включает в себя комбинацию из: микроархитектур, включающих в себя основные части процессора и методы их взаимодействиями между собой; набора команд, начиная от модели исполнения и заканчивая форматами данных и микрокода. Android адаптируется под любые среды и отлично работает на всех архитектурах. Единственным минусом является несовместимость отдельно взятых приложений/игр, заточенных под те или иные архитектуры.

В современных мобильных устройствах чаще всего встречаются следующие архитектуры:

* ARM: ARMv7 или armeabi
* ARM64: AArch64 или arm64
* x86: x86 или x86abi

Новые технологии: Распознавание владельца по лицу, 5G, Аккумуляторы и зарядные устройства.

3.Технологии разработки мобильных приложений. Нативная, гибридная и кроссплатформенная технология разработки. Сравнительная характеристика технологий разработки.

Ответ: **Нативная** - это создание продукта, который пишется на оригинальных языках программирования, созданных специально для выбранной платформы. Например, родными языками для Android являются Java и Kotlin, для iOS - Swift и Objective-C. Нативное приложение будет работать только на “своей” платформе.  
 **Гибридная** - Это приложение, разрабатываемое с помощью HTML, CSS, JavaScipt и других, более распространённых языков программирования, т. е. тех же технологий, которые [используются для создания сайтов](https://wezom.com.ua/razrabotka_sajtov). Разработка становится гибридным приложением с помощью «оборачивания» ее в нативную обложку. Чтобы осуществить эту процедуру, специалисты использую различные платные и бесплатные фреймворки, самыми распространенными из которых являются: PhoneGap, Cordova

**Кроссплатформенная** - это способ создания приложения с возможностью адаптации под несколько систем. По аналогии: один код — много систем.

**Сравнение:** Рынок предложений растёт. Статистика продаж мобильных приложений показывает, что год от года пользователи гаджетов всё чаще меняют стандартные сервисы на альтернативные. Так, родной менеджер задач заменяется на Wunderlist, почтовый клиент — на приложение Mailbox, Evernote оказывается предпочтительнее стандартных заметок.

Заказчику важно знать преимущества и недостатки каждого из подходов и не завышать ожидания, делая выбор. Проводить сравнительный анализ будет уместно по ряду критериев.

4. Нативные приложения. Технология разработки нативных приложений. Достоинства и недостатки данной технологии. Область применения нативных приложений.

Ответ:

Это один из самых распространенных вопросов, который возникает при переносе своего бизнеса в адаптацию под мобильные девайсы и планшеты. Такой вариант создают, учитывая специфику конкретной, заранее выбранной, платформы.

**Достоинства:**

* при правильной разработке можно добиться высокой скорости работы и достаточно производительность, что увеличивает спрос на использование со стороны пользователей;
* на высоком уровне безопасность использование подобных программных продуктов;
* со стороны разработчиков взимается достаточно высока оплата за реализацию поставляемого проекта;
* предусматривается широкий спектр функциональности;
* можно использовать без подключения к сети и выхода в Интернет;

**Недостатки:**

* низкий охват платформ;
* для разработки потребуется немало времени и финансов;
* косметические цели сопровождаются необходимость регулярного проведения обновлений;
* несовместимость с другими операционными системами.

5.Гибридные приложения. Технология разработки гибридных приложений. Достоинства и недостатки данной технологии. Область применения гибридных приложений.

Ответ: Напоминает комбинацию веб-разработки и ранее описанного варианта. В результате получается продукт, который выдается под обликом нативного. Приложение можно спокойно найти в AppStore и Play Market. Что позволяет устанавливать на любой гаджет.

Гибридное приложение располагает возможностью автономного обновления. Полноценная работа должно обязательно сопровождаться стабильным подключением к Интернету. Без такого функционала пользоваться приложением невозможно.

**Достоинства:**

* приемлемая цена разработки и быстрая реализации запланированного проекта;
* кроссплатформенность;
* несколько разработчиков;
* доступность опции автономного обновления.

**Недостатки:**

* необходимость стабильного Интернета при работе;
* сравнивая с нативными, более низкая рабочая скорость;
* отсутствие возможность использовать широкий спектр визуальных элементов и детализации.

6.Кроссплатформенные приложения. Технология разработки кроссплатформенных приложений. Достоинства и недостатки данной технологии. Область применения кроссплатформенных приложений.

Ответ:

Способность программного обеспечения работать с несколькими аппаратными платформами или операционными системами. Обеспечивается благодаря использованию высокоуровневых языков программирования, сред разработки и выполнения, поддерживающих условную компиляцию, компоновку и выполнение кода для различных платформ. Типичным примером является программное обеспечение, предназначенное для работы в операционных системах Linux и Windows одновременно.

Плюсы:

* короткий time-to-market;
* производительность приложений наравне с нативными решениями;
* общая стоимость разработки ниже аналогичного решения с нативными приложениями;
* поддержка единой кодовой базы;
* снижение затрат на исправление багов и добавление новой функциональности.

Минусы:

* некоторую функциональность необходимо разрабатывать независимо на обеих платформах, если нет кроссплатформенных библиотек;
* один язык разработки – Dart, который необходимо выучить, если компания разрабатывает приложение, не имея необходимой экспертизы.

7.Языки программирования. Основные языки программирования для разработки мобильных приложений. Языки программирования используемые при разработке нативных, гибридных и кросплатформенных мобильных приложений.

Ответ: Формальный язык, предназначенный для записи компьютерных программ.

Языки программирования: Phyton, JavaSctipt, C++, Swift, kotlin, HTML;

Основные языки для моб.приложений: JavaScript, Kotlin, Swift, HTML;

Нативные: Objective-C, Swift, C и С++-X-Code; Java-Android;

Гибридные: JavaSvript and HTML5;

Кроссплатформенные: PHP, Phyton;

8. Инструменты для разработки мобильных приложений Android. Перечень необходимых инструментов. JDK, IDE, эмуляторы, инструменты для отладки приложений, в том числе и на реальных устройствах. Достоинства и недостатки использования отладки приложений в эмуляторах и на реальных устройствах.

Ответ:

9. Мобильные операционные системы (ОС), классификация. ОС Android. Устройство, принцип построения, архитектура. Отличия от других мобильных ОС.

Ответ:

* **Мобильные операционные системы**
* Symbian OS.
* Windows **Mobile**.
* Android OS.
* iPhone OS (IOS)
* Palm OS.
* BlackBerry OS.
* Отличия: Единицы измерения: pt vs dp, Размер экрана: 320 pt x 568 pt vs 360 dp x 640 dp, Системный шрифт: San Francisco vs Roboto, Android Navigation Bar, Отличия в навигации и паттернах (UX), Отличия в появлении дочернего экрана;

10. Мобильные операционные системы (ОС), классификация. ОС iOS. Устройство, принцип построения, архитектура. Отличия от других мобильных ОС.

Ответ:

* **Мобильные операционные системы**
* Symbian OS.
* Windows **Mobile**.
* Android OS.
* iPhone OS (IOS)
* Palm OS.
* BlackBerry OS.

iOS-архитектуру можно разобрать на 4 отдельных слоя: (1) CocoaTouch, (2) Media, (3) CoreServices, и (4) Core OS.

Отличия: Единицы измерения: pt vs dp, Размер экрана: 320 pt x 568 pt vs 360 dp x 640 dp, Системный шрифт: San Francisco vs Roboto, Android Navigation Bar, Отличия в навигации и паттернах (UX), Отличия в появлении дочернего экрана;

11. Контроллер пользовательского интерфейса и его роль в мобильном приложении. Процесс создание начального контроллера пользовательского интерфейса. Класс Activity. Файл манифеста.

Ответ:

12. Контроллер пользовательского интерфейса Android и его роль в мобильном приложении. Основные этапы жизненного цикла контроллера пользовательского интерфейса.

Ответ:

13. Контроллер пользовательского интерфейса Android и его роль в мобильном приложении. Процесс изменения активного контроллера. Метод startActivity.

Ответ:  
14. Объекты визуального представления элементов приложения. Структура «Представления». Создание нового представления. Подкласс View.

Ответ:  
15. Объекты визуального представления элементов приложения. Способы изменения состояний «Представления». Методы класса View.

Ответ:  
16. Пользовательские компоненты. Понятие пользовательского компонента. Виды пользовательских компонентов. Назначение пользовательских компонентов. Характеристика пользовательского компонента. Подходы к созданию собственных компонентов.

Ответ:

17. Пользовательские компоненты. Понятие пользовательского компонента. Виды пользовательских компонентов. Назначение пользовательских компонентов. Характеристика пользовательского компонента. Использование собственных представлений.

Ответ: **Пользовательские компоненты** – это разнообразие готовых виджетов и компонентов, настраиваемые внешним видом или поведением и тд;

**Виды компонентов:** Пользовательский компонент, Пользовательское соединение, Пользовательский узел, Пользовательский стык:

**Их назначение:** Если вам не удается найти готовый системный компонент, который отвечает всем вашим нуждам, создается Пользовательский компонент для поиска системного;

**Характеристика:**

**Использование:** Одно из ключевых особенностей это, возможность, создавать пользовательские элементы на HTML – странице, вместо того чтобы создавать длинную, вложенную группу элементов:

18. Пользовательский ввод. Понятие, назначение и виды пользовательского ввода. Получение события касания и реакция на него. Свойство View.OnClickListener и метод onClick.

Ответ: **View**. **OnClickListener** – **это** интерфейс, который определяет метод **onClick**(**View**). Если у вас есть класс, который намерен прослушивать клики, вы должны реализовать интерфейс (если он еще не расширяет класс, который делает), и реализовать этот метод тоже.

Событие – это сигнал от браузера о том, что что-то произошло. Все DOM-узлы подают такие сигналы (хотя события бывают и не только в DOM).

click – происходит, когда кликнули на элемент левой кнопкой мыши (на устройствах с сенсорными экранами оно происходит при касании).

19. Пользовательский ввод. Понятие, назначение и виды пользовательского ввода. Получение события ввода с клавиатуры и реакция на него.

Ответ: alert, prompt и confirm, которые позволяют работать с данными, полученными от пользователя. alert выводит на экран окно с сообщением и приостанавливает выполнение скрипта, пока пользователь не нажмёт «ОК».

[prompt](https://learn.javascript.ru/uibasic#prompt)

Функция prompt принимает два аргумента

confirm выводит окно с вопросом question с двумя кнопками: OK и CANCEL.

**Результатом будет true при нажатии OK и false – при CANCEL(Esc).**  
20. «Передача сообщения». Способы передачи сообщений подписчикам, заинтересованным в их получении. API LocalBroadcastManager.

Ответ: LocalBroadcast приемник, используемый для межпроцессной связи, которая обеспечивает безопасность приложения, не позволяет другим процессам взаимодействовать. Кроме того, они не доступны вне приложения, поэтому они также обеспечивают безопасность системы.

Вам не нужно регистрироваться в menifest, если он находится в локальном приложении, это эффективно.

Приложение получает сбой, если вы передаете фильтр намерений null LocalBroadcastManager.getInstance(context).регистрация(трансляция,null);

21. «Передача сообщения». Получение и обработка сообщений. Экземпляр класса BroadcastReceiver и метод onReceive.

Ответ: При отправке формы срабатывает событие submit, оно обычно используется для проверки (валидации) формы перед её отправкой на сервер или для предотвращения отправки и обработки её с помощью JavaScript.

Метод form.submit() позволяет инициировать отправку формы из JavaScript. Мы можем использовать его для динамического создания и отправки наших собственных форм на сервер.

Для получения обратной связи от сервиса. В этом уроке для этой же цели используем [BroadcastReceiver](http://developer.android.com/reference/android/content/BroadcastReceiver.html" \t "_blank).

Схема такая:

- в Activity создаем BroadcastReceiver, а также создаем [IntentFilter](http://developer.android.com/reference/android/content/IntentFilter.html" \t "_blank), настроенный на определенный Action, и регистрируем (включаем) эту пару. Теперь BroadcastReceiver будет получать Intent-ы подходящие под условия IntentFilter

- в сервисе, когда нам понадобится передать данные в Activity, мы создаем Intent (с Action из предыдущего пункта), кладем в него данные, которые хотим передать, и посылаем его на поиски BroadcastReceiver

**Метод onReceive**() будет выполнен при получении широковещательного намерения, если полученное намерение соответствует фильтру. Приложения с зарегистрированными приёмниками широковещательных намерений будут запущены автоматически при получении соответствующего намерения.

22. Работа с файлами при разработке нативного приложения Android. Определение характеристик файла. Чтение данных из файлов. Запись данных в файлы.

Ответ:

23. Хранение данных в нативном приложении Android. Способы и цели хранения данных. Установление соединения с БД. Класс SqliteOpenHelper.

Ответ: Это класс применяемый для упрощения работы с БД. Для его использования необходимо создать сам класс-наследник, который определяет два его метода:

onCreate(): Вызывается при попытке доступа к БД, но когда еще эта база данных не создана;

onUpgrate(): Вызывается, когда необходимо обновления системы БД. Здесь можно пересоздать ранее созданную БД в onCreate(), установив соответствующие правила преобразования от старой БД к новой;

24. Хранение данных в нативном приложении Android. Способы и цели хранения данных. Создание в БД таблиц или хранимых объектов.

Ответ: Android обеспечивает полную поддержку баз данных SQLite. Любая созданная база данных будет доступна внутри приложения, тем самым можно хранить данные там.

**Внутренняя память:**

Если пользователь в приложении хранит данные во внутреннем хранилище, то только этот пользователь этого приложения имеет доступ к этим данным на мобильном устройстве, и эти данные автоматически удаляются, когда пользователь пользователь удаляет приложение. Кстати, о том, какая внутренняя память является частной.

**Внешнее хранилище**:

Отлично подходит для противоположности тому, что я только что сказал. Приложение dropbox, вероятно, использует внешнее хранилище для хранения папки dropbox пользователя, чтобы пользователь мог легко получить доступ к этим файлам вне приложения dropbox, например, с помощью файлового менеджера.

**Создание БД:**

Это место, где вы храните практически все, что хотите, в обычной базе данных-с преимуществом организации данных в таблицы, строки и столбцы. Он лучше всего работает с вещами, которые вы хотите отобразить в UI в виде списков - лучшим примером может служить великое изобретение, известное как CursorAdapter.

25. Хранение данных в нативном приложении Android. Способы чтения данных из таблиц или хранимых объектов. Метод SqliteDatabase.query.

Ответ: **Для чтения данных** в таблице используется инструкция SELECT. Инструкция SELECT является одной из наиболее важных инструкций Transact-SQL , и для нее существует много разновидностей синтаксиса.

**Хранимые процедуры** содержат как входные, так и выходные параметры и могут содержать инструкции, которые управляют потоком кода, например IF и WHILE. Использование хранимых процедур для всех повторяющихся действий в базе данных является хорошим стилем программирования.

Метод SqliteDatabase.query

**Это метод запроса:** Является помощниками для компоновки SQL, может создавать только простые запросы из одной таблицы.