**[1 СЛАЙД - Flutter && Dart]**

*Начнем с того, что разберемся с тем, что же такое Flutter?*

**[2 СЛАЙД - Что же такое Flutter?]**

**Flutter** – это фреймворк от Google для создания кроссплатформенных приложений, используя **один и тот же код**, для платформ iOS, Android, веб-приложения и даже десктоп-приложения под macOS, Windows, Linux.

Flutter предлагает свой собственный набор различных компонентов, которые называют **ВИДЖЕТЫ**. Благодаря им получается создавать красивые и быстрые приложения.

*С Flutter мы познакомились, пора узнать и про Dart.*

**[3 СЛАЙД – А кто такой Dart?]**

**Dart** – язык программирования, используемый Flutter (конечно же одним Flutter его применение не ограничивается).

В своем развитии Dart испытал влияние Java, JavaScript. Dart - **объектно-ориентированный язык**.

*Теперь пора узнать почему Flutter – круто?*

**[4 СЛАЙД - Почему Flutter - круто?]**

Основными преимуществами Flutter являются:

1. **Кроссплатформенность** – то, что код пишется всего один раз значительно повышает скорость разработки приложений сразу под несколько платформ.

Это как вместо огромного ящика с горой разных инструментов использовать волшебную палочку, которая может и гвозди забивать и выкрутить что-то.

1. **Богатый выбор виджетов** – Flutter предоставляет богатый набор различных настраиваемых виджетов, которые помогают создать красивый и удобный UI для юзера.
2. **Открытый исходный код** – так как Flutter является фреймворком с открытым исходным кодом, это делает его доступным для всех. Растущее сообщество разработчиков, работающих с фреймворком, помогает развиваться и расширять свои возможности.

*Вот, например резульаты опросника StackOverflow о самых используемых фреймворках.*

**[5 СЛАЙД]**

Flutter занимает здесь **11-ое место**, благодаря своей простоте и эффективности, к тому же он довольно **популярен среди** **новичков** в IT.

*Кто же использует этот ваш Flutter?*

**[6 СЛАЙД]**

Например, **Alibaba Group** (владелец Taobao, AliExpress) использовала Flutter при разработке мобильного приложения китайского аналога нашего Авито, **Ebay** при разработке мобильного приложения EbayMotors для продажи автомобилей, **Toyota** использовала API Flutter для встраиваемых устройств при создании ПО под автомониторы в их авто, **Яндекс** разработал на Flutter приложения Яндекс.Драйв, Яндекс.Маркет. Ну, а то, что некоторые приложения **Google** сделаны на Flutter неудивительно.

*Однако, ведь никто не идеален, ведь так? И у Flutter есть свои недостатки.*

**[7 СЛАЙД - Какие минусы у Flutter?]**

Основными недостатками Flutter являются:

1. **Размер приложений** – использование Dart-кода и движка Flutter зачастую приводят к увеличению объема приложения, по сравнению с нативной версией того же самого приложения.
2. **Производительность** – хоть и Flutter обещает отличную производительность, он не всегда стабилен, потому при разработке стоит быть осторожным. Нативные версии под каждую платформу в абсолютном большинстве случаев производительнее.
3. **Flutter только растет** – Flutter приобрел значительную популярность в сообществе, однако он все еще относительно новый, потому и сторонних библиотек для него создано меньше, чем для того же React Native.

*Ну что же, с Flutter и Dart мы теперь знакомы, потому можно перейти как практике. Для начала установим плагин для VSCode, а вместе с ним и Flutter.*

**[8 СЛАЙД - Установка Flutter]**

На левой панели VSCode заходим во вкладку “**Extensions**” (там нарисованы 4 квадратика).

**[9 СЛАЙД]**

Пишем в поиске – **Flutter** и устанавливаем самый первый плагин, у которого больше **9 миллионов загрузок**. Вместе с ним установится и плагин для Dart.

**[10 СЛАЙД]**

Ждём, пока VSCode не завершит загрузку.

**[11 СЛАЙД]**

Далее, чтобы загрузить FlutterSDK попробуем создать проект. На верхней панели нажимаем на **View**, а затем на **Command Palette**.

**[12 СЛАЙД]**

В открывшемся окне пишем Flutter. Нам выдаст три варианта: создать новый проект, запуск Flutter Doctor и Flutter Upgrade. Создаем новый проект - **New Project.**

**[13 СЛАЙД]**

Но создать проект не получилось, так как у нас не установлен сам FlutterSDK. Потому нажимаем на **– Download SDK**, чтобы начать загрузку.

**[14 СЛАЙД]**

Затем нам предложит выбрать папку, куда скачется FlutterSDK.

**ВНИМАНИЕ! –** путь к вашей папке не должен содержать пробелов. Потому, например – “C:\Program Files\” не подойдет.

Для этого я специально создал на диске **C:\** папку **fluttersdk**. (Итоговый путь – “ C:\fluttersdk\”). После того как выбрали папку нажимаем **Clone Flutter.**

**[15 СЛАЙД]**

Начинается загрузка и она займет какое-то время, потому стоит подождать.

**[16 СЛАЙД]**

После того, как скачается FlutterSDK нам предложит добавить его в системную переменную PATH, чтобы мы могли легче обращаться к Flutter (например, прямо из консоли). Обязательно соглашаемся и жмём – **Add SDK to PATH.**

Ну вот, мы наконец-то всё скачали и поставили, пора создать первый проект.

**[17 СЛАЙД – Создание первого проекта]**

Попробуем снова так же создать проект. На верхней панели нажимаем на **View**, а затем на **Command Palette**.

**[18 СЛАЙД]**

В открывшемся окне пишем Flutter. Нам выдаст три варианта: создать новый проект, запуск Flutter Doctor и Flutter Upgrade. Создаем новый проект - **New Project.**

**[19 СЛАЙД]**

Далее нам предложит выбрать одну из готовых заготовок для нашего проекта. Выбираем “**Empty Application**” – для начала нам хватит этого.

**[20 СЛАЙД]**

Затем нам предложит выбрать папку, где будет хранится проект.

Для этого я создал на диске **C:\** папку **src**. (Итоговый путь – “ C:\src\”). После того как выбрали папку нажимаем **Select Folder to create project in.**

**[21 СЛАЙД]**

В появившемся окне вводим наше название для проекта. Пусть это будет – “helloworld” или любое другое.

**[22 СЛАЙД]**

Поздравляю, ваш первый проект создан!

*Он содержит довольно много всего, потому давайте посмотрим, что к чему.*

**[23 СЛАЙД — Важные части проекта]**

* **lib** – директория, где хранится ваш Dart-код и все Flutter-компоненты.
* **pubspec.yaml** – конфигурация проекта. Там указаны зависимости вашего проекта, название, ссылки на ассеты и так далее.
* **android, ios, web, windows, linux** – здесь хранится нативный код для каждой платформы. Это проекты, которые изначально запускаются, а потом передают управление Flutter.
* **lib/main.dart -> void main()** – точка входа в приложение.

*Давайте теперь взглянем в тот код, который у нас уже есть в функции main() нашего проекта.*

**[24 СЛАЙД – Точка входа]**

Итак, функция **main()** то, с чего начинается выполнение программы, содержит функцию **runApp()**, которую нам предоставляет фреймворк. Именно она запускает наше приложение. И в эту функцию передаётся **ВИДЖЕТ** (в нашем случае он называется **MainApp**)

Что же такое виджеты в Flutter?

**Виджеты** – это такие строительные блоки пользовательского интерфейса в Flutter приложениях. Каждый виджет является частью всего пользовательского интерфейса (например, кнопки, текст и другие).

Именно потому часто говорят: **«Всё – является виджетом».**

*Давайте же теперь узнаем, как выглядит наш виджет MainApp.*

**[25 СЛАЙД – Точка входа]**

Выглядит сложновато, но на самом деле здесь не таится ничего сложного.

Наш виджет **MainApp** переопределяет функцию **build()**, чтобы себя нарисовать или обновить в интерфейсе. **build()** должен возвращать другой виджет. Например, если внимательно рассмотреть код функции **build()** можно увидеть иерархию различных вложенных друг в друга виджетов, как будто матрёшка.

**build()** возвращает виджет **MaterialApp**, внутри которого создается виджет **Scaffold**, внутри него же **Center** и наконец самый последний элемент нашей матрешки **Text**, которому мы передаём текст “Hello World!”. Более подробно с каждым из них мы познакомимся дальше.

Сейчас же главное запомнить, что именно благодаря вот этой матрёшке создаётся вся красота нашего интерфейса!

*Что же, пора попробовать запустить наш проект!*

**[26 СЛАЙД – Запуск приложения]**

На верхней панели нажимаем на **Run**, а затем на **Run Without Debugging**. Дебаггер нам сейчас не пригодится, потому запустим без него.

**[27 СЛАЙД]**

Затем нам предложит выбор платформы, на которой выполнится запуск. Скорее всего будет выбор из 2 вариантов: **Windows** версия и **Web** версия. Разницы от выбора не будет. Что мы не выберем получим один и тот же результат. Я выбрал Windows версию. После этого началась сборка проекта, это займет некоторое время.

**[28 СЛАЙД]**

Когда сборка завершится появится окно с вашим приложением.

*Теперь давайте попробуем написать простенькое приложение на Flutter. Оно будет по нажатию кнопки отображать содержимое поля для ввода в блок текста. Для этого можно создать ещё один проект или продолжить в прошлом.*

**[29 СЛАЙД – Код простого приложения]**

Добавим перед нашей функцией **main()** определение для контроллера **TextEditingController**, через который мы сможем позже получить текст нашего поля ввода.

**[30 СЛАЙД]**

Также чуть-чуть изменим функцию **build()** у **MainApp**, добавив в тело нашего **Scaffold** виджет **Padding**, который добавит внутренние отступы для наших виджетов.

А в виджет **Center** добавим свой собственный виджет **InputForm**, который определим дальше. Именно в нём будет интерфейс нашего приложения.

**[31 СЛАЙД]**

А вот и определение нашего виджета **InputForm**. Важно обратить внимание на то, что он наследуется от **StatefulWidget**, то есть у него есть какое-то изменяемое состояние, в отличии от виджета **MainApp**, который не имеет изменяемого состояния, а потому наследуется от **StatelessWidget**. Более подробно об этом позже.

Также здесь же переопределим функцию **createState()**, которая будет возвращать экземпляр **\_InputFormState**, который мы определим дальше. Именно этот класс будет отвечать за отрисовку нашего виджета и изменение состояния **InputForm**.

**[32 СЛАЙД]**

Смысл слова **Stateless** заключается в том, что все свойства виджета *неизменяемы*. Отсюда следует, что единственный способ изменить их — это создать новый экземпляр виджета.

Теперь самое сложное: виджет с состоянием тоже *неизменяем*. Однако объект состояния, с которым он связан, является *изменяемым* и позволяет перестраивать виджет при каждом изменении состояния.

Другими словами: **StatelessWidget** никогда не перестроится сам по себе, но у него есть возможность сделать это извне. **StatefulWidget** может сам инициировать перестройку (например, вызвав **setState()**).

Как показано на рисунке, **StatelessWidget** предлагает только один вариант для запуска **build()** – конструктор.

В то время как **StatefulWidget** имеет множество триггеров, вызывающих **build()**. **didUpdateWidget()** вызывается всякий раз, когда конфигурация виджета меняется. Например, это может происходить при повороте экрана.

**[33 СЛАЙД]**

Вот мы и добрались до определения нашего последнего класса **\_InputFormState**, наследуемый от **State** типа **InputForm**.

Создадим поле ***\_text***, которое будет хранить текущий текст, который мы будем отображать.

Затем создадим функцию **\_changeText()**, которая принимает какую-то строку input. В ней мы вызываем функцию **setState()**, которая уведомит Flutter, что состояние нашего виджета изменилось, а потому его нужно отрисовать заново. Также не забудем передать в **setState()** само изменение нашего поля **\_text**.

*Осталось лишь написать отрисовку нашего виджета в переопределяемой функции* ***build()****.*

**[34 СЛАЙД]**

Наша функция **build()** будет возвращать виджет **Column**, который отображает элементы в столбец. Внутри него мы определим три виджета: виджет Text для отображения нашего поля **\_text**, виджет **TextField** для ввода нашего текста (не забудем передать в него наш **\_controller**) и последний будет виджет **Padding** задающий отступ для нашей кнопочки.

Внутри **Padding** мы определим **ElevatedButton**, который будет нашей кнопкой. Так же по нажатию на кнопку мы будем вызывать функцию **\_changeText()**, в которую будем передавать текст от нашего **\_controller**.

**[35 СЛАЙД – Виды кнопок]**

1. **Elevated Button** — стилизована так, будто бы парит в воздухе.
2. **Floating Action Button** — маленькая круглая кнопка на экране.
3. **Outlined Button** — кнопка с обводкой вокруг.
4. **Icon Button** — отлично подходит, когда нужно, чтобы по нажатию на иконку происходило какое-то действие.
5. **Text Button** — привычная нам кнопка-текст.

**[36 СЛАЙД]**

Осталось лишь запустить наш проект и попробовать его работу.

*Итак, что же мы узнали за сегодня?*

**[37 СЛАЙД]**

* **Flutter** и **Dart** – волшебная палочка, которой можно и гвоздь забить, и что-то выкрутить.
* Любимая игрушка в детстве у **Flutter** была матрёшка.
* **Flutter** – круто.

**[38 СЛАЙД – Где найти всё за сегодня]**

Спасибо за внимание! Всё за сегодня можно найти по ссылке и QR-коду на слайде.