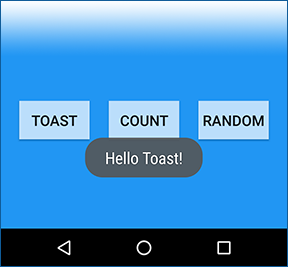
**Обработка нажатия кнопок и взаимодействие с view в android**

На [прошлом уроке](https://www.fandroid.info/urok-3-resursy-i-makety-ekrana-prilozheniya-constraintlayout-v-android-studio/) мы добавили кнопки на главном экране вашего приложения, но пока эти кнопки ничего не делают. В этом уроке сделаем так, чтобы кнопки реагировали, когда пользователь нажимает их.

.

**Отображение сообщения по нажатию кнопки**

На этом шаге мы добавим функцию отображения тоста – короткого сообщения на экране. Функция будет выполняться при нажатии кнопки с надписью TOAST.

1. Откройте класс MainActivity.kt. (раскройте ветвь **app** > **java** > **com.example.android.myfirstapp** чтобы найти MainActivity). Этот класс описывает поведение главного экрана нашего приложения.Пока что класс содержит только одну функцию, onCreate(). Функция onCreate() выполняется, когда активити стартует.
2. Посмотрите внимательно на функцию onCreate(). Обратите внимание на вызов функции setContentView(). Эта строка устанавливает файл ресурсов activity\_main.xml в качестве разметки активити главного экрана.

|  |  |
| --- | --- |
| 1 | setContentView(R.layout.activity\_main); |



1. Добавим новую функцию toastMe() в класс MainActivity. Функция toastMe() принимает один аргумент — View. Это представление, получающее событие нажатия кнопки.Функция создает и отображает всплывающее уведомление. Вот ее код:



|  |  |
| --- | --- |
| 1  2  3  4  5 | fun toastMe(view: View) {    val myToast = Toast.makeText(this, "Hello Toast!", Toast.LENGTH\_SHORT)    myToast.show()  } |

В языке Kotlin, если явно не используется никакого модификатора доступа, то по умолчанию применяется public .  Далее идет слово fun, обозначающее функцию, и ее имя. В скобках передаваемый функции аргумент – его имя и тип разделены двоеточием. Далее объявляется переменная val myToast. Словом val обозначаются переменные «только для чтения», значение которых обычно задается только один раз. Обычные изменяемые переменные обозначаются в языке Kotlin словом var. Далее переменной myToast присваивается результат вызова метода makeText java-класса Toast. Метод makeText принимает контекст, сообщение и длительность отображения тоста, и возвращает тост в переменную  myToast. Тост затем отображается методом show().

Функция toastMe является примером использования java-кода в kotlin-классе.

1. Откройте файл макета activity\_main.xml и добавьте свойство android:onClick кнопкеToast. Значением свойства установите **toastMe**.



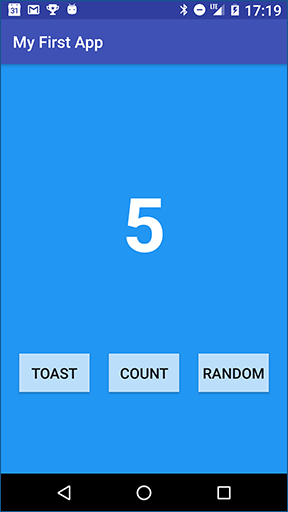
|  |  |
| --- | --- |
| 1 | android:onClick="toastMe" |

1. Запустите приложение и нажмите кнопку TOAST, вы должны увидеть на экране короткое сообщение с текстом «Hellо Toast!».

Таким образом, что для того, чтобы сделать элемент экрана интерактивным вам нужно:

* Реализовать функцию, определяющую поведение экранного элемента при нажатии на него. Эта функция должна быть public, не возвращать никаких значений, и принимать View в качестве аргумента.
* Установить имя функции в качестве значения свойства **onClick** в экранном элементе.

**Реализация счетчика по нажатию кнопки Count**

Функция, отображающая тост, очень проста. Она не взаимодействует с другими элементами экрана. На следующем шаге мы будем по нажатию кнопки находить и обновлять другие view-элементы экрана. Напишем функцию для кнопки Count, которая при нажатии будет увеличивать значение текстового поля на 1.

1. В файле макета экрана определим параметр id для TextView:



|  |  |
| --- | --- |
| 1  2 | <TextView     android:id="@+id/textView" |

1. Отройте класс ActivityMain.kt и объявите переменную для TextView:



|  |  |
| --- | --- |
| 1 | private lateinit var textView: TextView |

1. В методе onCreate инициализируем её:



|  |  |
| --- | --- |
| 1 | textView = findViewById(R.id.textView) |

1. В теле класса ActivityMain.kt, добавим функцию countMe(). Эта функция будет вызываться при нажатии кнопки Count, поэтому она должна быть public, не иметь возвращаемых значений, и получать View в качестве аргумента.



|  |  |
| --- | --- |
| 1  2 | fun countMe (view: View) {  } |

1. Получаем значение текстового поля TextView.



|  |  |
| --- | --- |
| 1  2  3  4 | fun countMe (view: View) {     // Get the value of the text view.     val countString = textView.text.toString()  } |

1. Конвертируем полученное значение в число, и увеличим его на единицу.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | fun countMe (view: View) {     // Get the value of the text view.     val countString = textView.text.toString()     // Convert value to a number and increment it     var count: Int = Integer.parseInt(countString)     count++  } |

1. Отображаем новое значение в TextView.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | fun countMe (view: View) {     // Get the value of the text view.     val countString = textView.text.toString()     // Convert value to a number and increment it     var count: Int = Integer.parseInt(countString)     count++     // Display the new value in the text view.     textView.text = count.toString();  } |

1. Функция готова. Теперь нужно вызвать ее при нажатии кнопки COUNT. Для этого нужно установить имя функции в качестве значения свойства**onClick** кнопки в файле макета.

**Запустите приложение на устройстве.**

При нажатии кнопки COUNT значение текстового поля увеличивается на единицу.

На этом урок подошел к концу. На следующем уроке мы добавим в приложение второй экран, который будет отображать информацию на основании данных с первого экрана.