# Programmer dictionary: Event listener vs event handler



[Marcin Moskala](https://medium.com/@marcinmoskala?source=post_page-----305c667d0e3c--------------------------------)

[Follow](https://medium.com/m/signin?actionUrl=https%3A%2F%2Fmedium.com%2F_%2Fsubscribe%2Fuser%2F171dab54cea9&operation=register&redirect=https%3A%2F%2Fblog.kotlin-academy.com%2Fprogrammer-dictionary-event-listener-vs-event-handler-305c667d0e3c&user=Marcin%20Moskala&userId=171dab54cea9&source=post_page-171dab54cea9----305c667d0e3c---------------------follow_byline-----------)

[Jan 25, 2018](https://blog.kotlin-academy.com/programmer-dictionary-event-listener-vs-event-handler-305c667d0e3c?source=post_page-----305c667d0e3c--------------------------------) · 3 min read

Event listener and event handler are two terms that cause confusion. Especially in Kotlin, where the boundary between them was blurred. Here, I am trying to make it simple.



Instance of DogListener

The general definition of listeners and handlers is much wider than its use in Android. Here are popular definitions:

*A* ***listener*** *watches for an event to be fired.*

*The* ***handler*** *is responsible for dealing with the event.*

It might be confusing because often the same object listens for the events and handles them. Although it is usually assumed that when we set the anonymous class as a listener, its methods are actual handlers:

cancelImage.setOnClickListener(**object** : View.OnClickListener { //1

**override fun** onClick(v: View?) { // 2

dismiss()

}

})

1. Anonymous class is here used as a **listener**
2. Method onClick is here event **handler**

We can use named classes as a listeners:

**class** OnCancelSnackListener(

**val snackbar**: Snackbar

): View.OnClickListener {

**override fun** onClick(v: View?) {

**snackbar**.dismiss()

}

}

Usage:

cancelImage.setOnClickListener(OnCancelSnackListener(**this**))

Listeners are often objects, so they are commonly suffixed with Listener. Handlers are normally not suffixed, but instead, they most often have on as the prefix (onClick, onSwipe etc.). Note, that is is more problematic when we set listener using lambda expression:

cancelImage.setOnClickListener **{** dismiss() **}**

What is this lambda expression? Listener or handler? Formally it specifies how the event should be handled. The listener object is generated under the hood by Kotlin. Does it mean that when we name function that accepts handler then we should name it using Handler suffix (like setOnClickHandler)? Not at all. Although you can find such approaches in some JS libraries, like ExJS:

handler: function() {

}

listeners: {

'click': function() {

}

}

Convention in Java, and I believe most other languages, is to use Listener suffix for historical reasons! You can find out on Kotlin stdlib and JetBrains Kotlin code that Kotlin also adopt that convention. So you can confidently define following functions:

fun setOnLoadedListener(handler: ()->Unit) {

// Code

}

fun addOnFlingListener(handler: ()->Unit) {

// Code

}

As well as you can define following functions:

fun setOnLoadedListener(listener: ()->Unit) {

// Code

}

fun addOnFlingListener(listener: ()->Unit) {

// Code

}

Just remember that it is a good practice to state one convention for the project and respect it.