URGH. IS THIS GAMEDEV-RELATED?

Nope

Some of the suggestions

“It's not about the language ...”

Some of the suggestions

“It's not about the language ...”

“Asynchronous HTTP-requests in Android”

Some of the suggestions

“It's not about the language ...”

“Asynchronous HTTP-requests in Android”

“What is Kotlin and why should we care?”

Topic Title Suggestion

“It's not about the language, but we'll be using Kotlin to make an asynchronous HTTP request in Android and walk through it in about 10 minutes minus whatever time I've squandered so far”

… Rolls right off your tongue

6

6

Number of years since I did any serious Android development

0

0

LOC written in Kotlin since release 1.0

I might have given a seminar/workshop or two for the pre-releases a few years ago, but don't hold that against me.

Breakdown (not the nervous kind)

Hour 1

Annoying. Had to install the Android APIs. Managed to read up on asynchronous operations in the meantime

Hour 2

Swears about LibreOffice Impress a bit. Reads up on Kotlin+Android and going through tutorials

Hour 3

Got the asynchrony hooked up, but not doing any real requests. LibreOffice decides to crash again

Hour 4

Fetching and parsing JSON. Finishing up slides.

Asynchronous work

So we want to do non-blocking IO.

How? Why? Wah?

Let's try the app

Really really quick code overview

So, we have our activity:

```
class GithubActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_github)
    }
}
```

We set up an event handler

```
override fun onCreate(savedInstanceState: Bundle?) {
    // … Boring stuff cut out

    // Oh, and this is a lambda:
    button.setOnClickListener {
        // … Stuff goes here when clicking
    }
}
```

We pretend to have an “AsyncFetch” class

```
override fun onCreate(savedInstanceState: Bundle?) {
    // … Boring stuff cut out

    button.setOnClickListener {
        val txt = input.text

        if (txt.isEmpty()) {
            toast("Please enter a name")
        } else {
            AsyncFetch(progress, {
                r -> populate(JSONObject(r))
            }).execute(txt.toString())
        }
    }
}
```

This is another lambda (taking a parameter)

```
override fun onCreate(savedInstanceState: Bundle?) {
    // ... Boring stuff cut out

    button.setOnClickListener {
       val txt = input.text

        if (txt.isEmpty()) {
            toast("Please enter a name")
        } else {
            AsyncFetch(progress, {
                r -> populate(JSONObject(r))
            }).execute(txt.toString())
        }
    }
}
```

The “AsyncFetch” class looks like this (kind of):

```
/* Constructor, field + class declaration in one */
internal class AsyncFetch(val progressBar: ProgressBar, val action: (String) -> Unit)
    : AsyncTask<String, Unit, String>() {  // ← Inheritance

    init {
        progressBar.visibility = View.VISIBLE
    }

    override fun doInBackground(vararg input: String?): String? {
        try {
            val conn = URL("https://api.github.com/users/${input[0]}").openStream()
            // ... Connection logic cut out
        } catch (ex : Exception) {
            Log.e("http", Log.getStackTraceString(ex))
            return null
        }
    }

    override fun onPostExecute(result: String?) {
        action(result.orEmpty())
        progressBar.visibility = View.INVISIBLE
    }
```

Declare class and assign+declare fields

```kotlin
/* Constructor, field + class declaration in one */
internal class AsyncFetch(val progressBar: ProgressBar, val action: (String) -> Unit)
    : AsyncTask<String, Unit, String>() {  // ← Inheritance

    init {
        progressBar.visibility = View.VISIBLE
    }

    override fun doInBackground(vararg input: String?): String? {
        try {
            val conn = URL("https://api.github.com/users/${input[0]}").openStream()
            // ... Connection logic cut out
        } catch (ex : Exception) {
            Log.e("http", Log.getStackTraceString(ex))
            return null
        }
    }

    override fun onPostExecute(result: String?) {
        action(result.orEmpty())
        progressBar.visibility = View.INVISIBLE
    }
```

Inherit from AsyncTask (Android API)

```
/* Constructor, field + class declaration in one */
internal class AsyncFetch(val progressBar: ProgressBar, val action: (String) -> Unit)
    : AsyncTask<String, Unit, String>() {  // ← Inheritance

    init {
        progressBar.visibility = View.VISIBLE
    }

    override fun doInBackground(vararg input: String?): String? {
        try {
            val conn = URL("https://api.github.com/users/${input[0]}").openStream()
            // ... Connection logic cut out
        } catch (ex : Exception) {
            Log.e("http", Log.getStackTraceString(ex))
            return null
        }
    }

    override fun onPostExecute(result: String?) {
        action(result.orEmpty())
        progressBar.visibility = View.INVISIBLE
    }
```

… **init** is called right after constructor for non-assignment stuff

```
/* Constructor, field + class declaration in one */
internal class AsyncFetch(val progressBar: ProgressBar, val action: (String) -> Unit)
    : AsyncTask<String, Unit, String>() {  // ← Inheritance

    init {
        progressBar.visibility = View.VISIBLE
    }

    override fun doInBackground(vararg input: String?): String? {
        try {
            val conn = URL("https://api.github.com/users/${input[0]}").openStream()
            // ... Connection logic cut out
        } catch (ex : Exception) {
            Log.e("http", Log.getStackTraceString(ex))
            return null
        }
    }

    override fun onPostExecute(result: String?) {
        action(result.orEmpty())
        progressBar.visibility = View.INVISIBLE
    }
```

Do **synchronous** work in an **asynchronous** context

```kotlin
/* Constructor, field + class declaration in one */
internal class AsyncFetch(val progressBar: ProgressBar, val action: (String) -> Unit)
    : AsyncTask<String, Unit, String>() {  // ← Inheritance

    init {
        progressBar.visibility = View.VISIBLE
    }

    override fun doInBackground(vararg input: String?): String? {
        try {
            val conn = URL("https://api.github.com/users/${input[0]}").openStream()
            // ... Connection logic cut out
        } catch (ex : Exception) {
            Log.e("http", Log.getStackTraceString(ex))
            return null
        }
    }

    override fun onPostExecute(result: String?) {
        action(result.orEmpty())
        progressBar.visibility = View.INVISIBLE
    }
```

Once **doInBackground** exits, hit **onPostExecute** and invoke callback with result

```
/* Constructor, field + class declaration in one */
internal class AsyncFetch(val progressBar: ProgressBar, val action: (String) -> Unit)
    : AsyncTask<String, Unit, String>() {  // ← Inheritance

    init {
        progressBar.visibility = View.VISIBLE
    }

    override fun doInBackground(vararg input: String?): String? {
        try {
            val conn = URL("https://api.github.com/users/${input[0]}").openStream()
            // ... Connection logic cut out
        } catch (ex : Exception) {
            Log.e("http", Log.getStackTraceString(ex))
            return null
        }
    }

    override fun onPostExecute(result: String?) {
        action(result.orEmpty())
        progressBar.visibility = View.INVISIBLE
    }
```

1) So.. What IS Kotlin, and why should we care?

Easy. It's a programming language.

Also, it's the default language for Android development nowadays. Ergo - care.

2) Asynchronous requests

Not the prettiest way of having to deal with asynchronous IO, but not too bad either.

3) Claim: “It's not about the language...”

We all have our favourite languages, so in a way it IS about the language.

3) Claim: “It's not about the language...”

Honestly though - It's actually not about the language **all that often**

3) Claim: “It's not about the language...”

Learning to write syntactically correct code in a language?

Yes, that's definitely important

3) Claim: “It's not about the language...”

At some point though:

Don't cling to a technology – Instead, hone your problem solving skills

3) Claim: “It's not about the language...”

We are programmers – **our task is to solve problems**, not learn a specific language.

Actually, I've always wanted my business card to say “Professional problem solver” instead.

Protip #1

Don't refer to yourself* as a “C# developer” or a “Java developer”

** Unless you're doing Lisp and want to show off*

Protip #2

Never say “I can't do that, because I don't know X”.

Instead, solve the problem – language is secondary.

Protip #2.5

“I don't know language X” is not the same thing as “can't work in language X”

Want the source code? Lemme know

… Or if you want to ask me more questions about Kotlin.

Buy me a beer, and I'll tell you what I know, and I'll throw in a rant about LibreOffice Impress for free.

(Seriously... Use mc2)

The End

Hm.. Any time for questions?