#### @Louis\_CAD

## Writing your own: wrapping callbacks

```
suspend fun < T > Call< T > .await(): T = suspendCancellableCoroutine { c ->
    c.invokeOnCancellation {
        cancel() // Cancels network operation 🥎
    enqueue(object : Callback<T> {
        override fun onFailure(call: Call<T>, t: Throwable) {
            c.resumeWithException(t)
        override fun onResponse(call: Call<T>, response: Response<T>) {
```

```
if (response.isSuccessful) {
    c.resume(response.body()!!) // Fail fast if null.
} else {
    c.resumeWithException(HttpException(response))
}
}
```



### Don't forget try/catch at use-site.

## wrapping

#### allback

## Writing your own:







```
suspend fun < T > Call< T > .await(): T = suspendCanc
```

#### c.invokeOnCancellation {

#### ableCoroutine { c ->

#### cancel() // Cancels network operation 🤙



enqueue(object : Callback<T> {

#### rowable

override fun onFailure(call: Call<T>, t: T

c.resumeWithException(t)



override fun onResponse(call: Call<T>, response: Response<T>) {

#### else

if (response.isSuccessful)

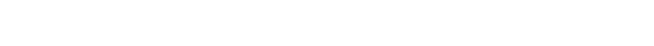
c.resume(response.body()!!) // Fail fast if null.

#### c.resumeWithException(HttpException(response))











#### allback

# Writing your own:









