Composing suspend Functions

explicit concurrency via async

20

```kotlin
suspend fun bakePretzels(): List<FinishedPretzel> {
    val oven = preheatOven(ColdOven)
    val dough = prepareDough()
    val shapedPretzels: List<UncookedPretzel> = List(5) { shapePretzel(dough) }
    val bakedPretzels: List<CookedPretzel> = bake(oven, shapedPretzels)
    val topping: Topping = prepareTopping()
    return bakedPretzels.map { finishPretzel(it, topping) }
}
```

```kotlin
suspend fun bakePretzels(): List<FinishedPretzel> = coroutineScope {
    val oven = async { preheatOven(ColdOven) }
    val dough = async { prepareDough() }
    val uncookedPretzels = List(5) { async { shapePretzel(dough.await()) } }
    val bakedPretzels = async { bake(oven.await(), uncookedPretzels.awaitAll()) }
    val topping = async { prepareTopping() }
    bakedPretzels.await().map { finishPretzel(it, topping.await()) }
}
```

another coroutine builder that returns a `Deferred<T>`

`async` launches concurrently: we need to call `await` to wait for the result

Composing suspend Functions
explicit concurrency via async

another coroutine builder that returns a `Deferred<T>`

```
suspend fun bakePretzels(): List<FinishedPretzel> = coroutineScope {
    val oven = async { preheatOven(ColdOven) }
    val dough = async { prepareDough() }
    val uncookedPretzels = List(5) { async { shapePretzel(dough.await()) } }
    val bakedPretzels = async { bake(oven.await(), uncookedPretzels.awaitAll()) }
    val topping = async { prepareTopping() }
    bakedPretzels.await().map { finishPretzel(it, topping.await()) }
}
```

`async` launches concurrently: we need to call `await` to wait for the result

Coroutine Context