async

```
public fun <T> async(
        context: CoroutineContext = DefaultDispatcher,
        ... ,
        block: suspend CoroutineScope.() → T
): Deferred<T>

public suspend fun <T> Deferred<T>.await(): T



val asyncValue: Deferred<String> = async {
    val x = doSomething()
    doSomethingElse(x)
}

val value = asyncValue.await()
```

Channels

```
interface Channel<E> : SendChannel<E>, ReceiveChannel<E>

suspend fun <E> SendChannel<E>.send(element: E)

fun <E> SendChannel<E>.close(cause: Throwable? = null): Boolean

suspend fun <E> ReceiveChannel<E>.receive(): E
```