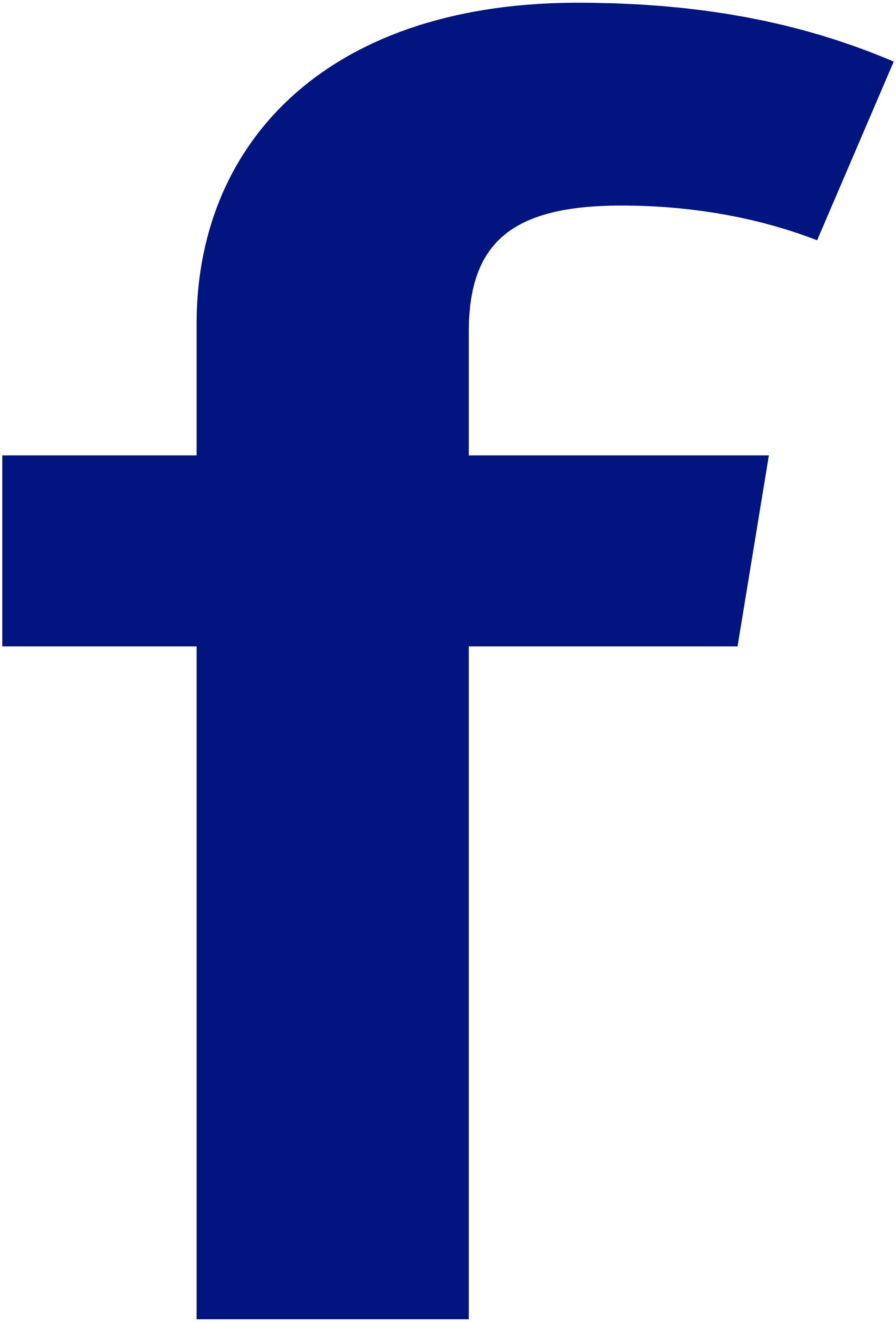
Communication Between Coroutines

channels

42



u

n

C

O

r

O

U

t

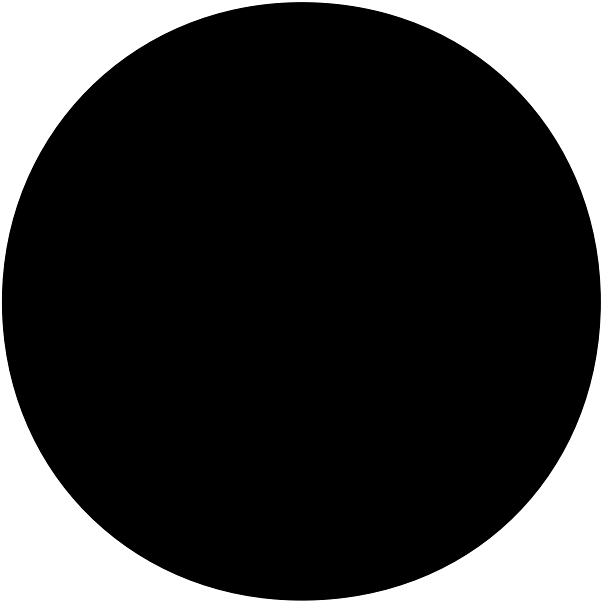
n

e

S

C

p



b

a

k

r

y

P

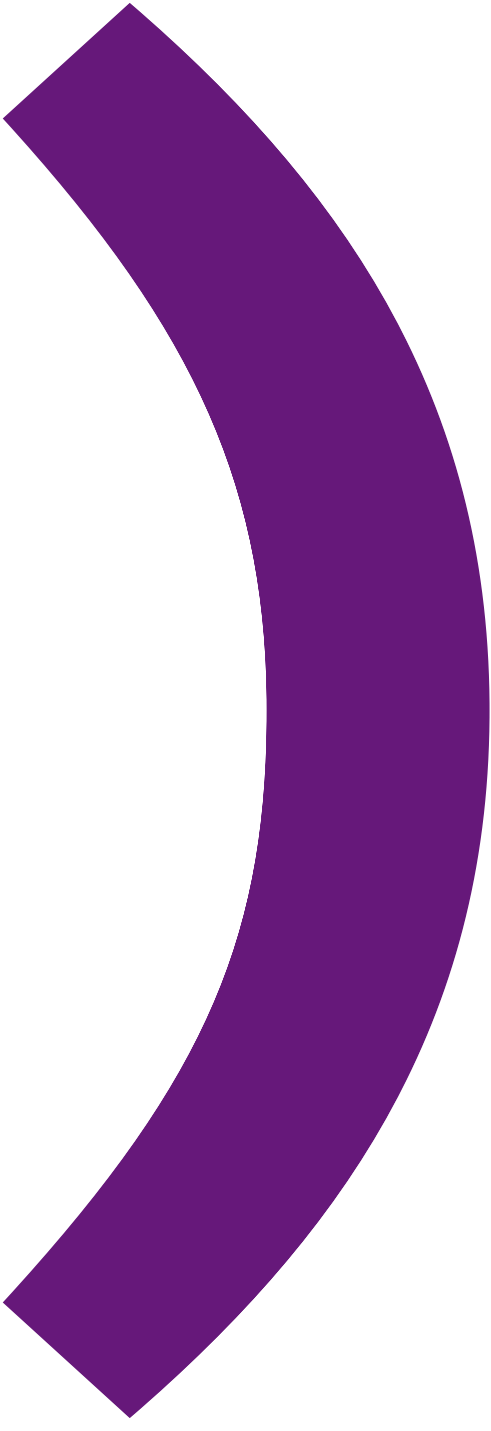
i

p

e

ʅ

(



=

p

r

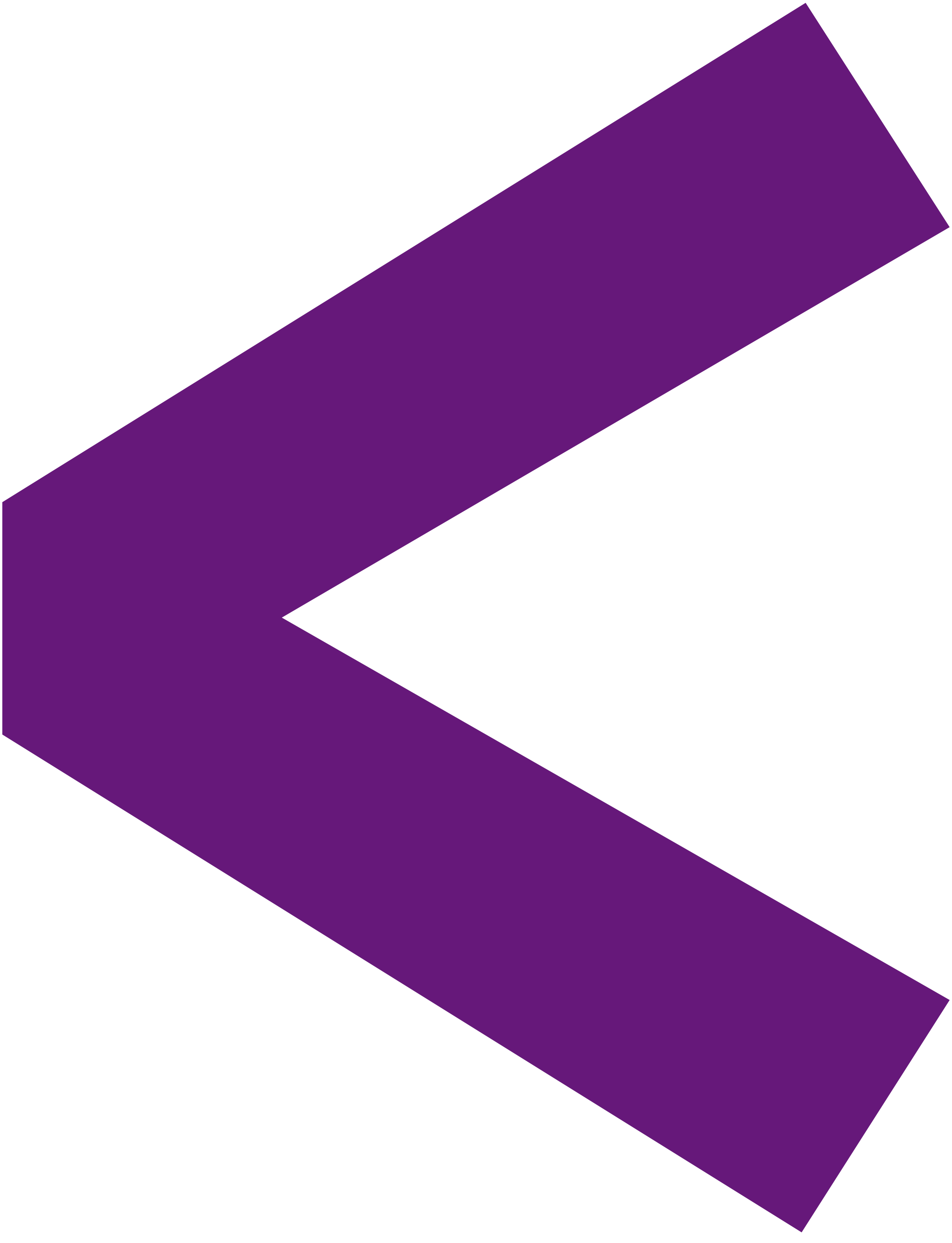
0

d

u

C

e



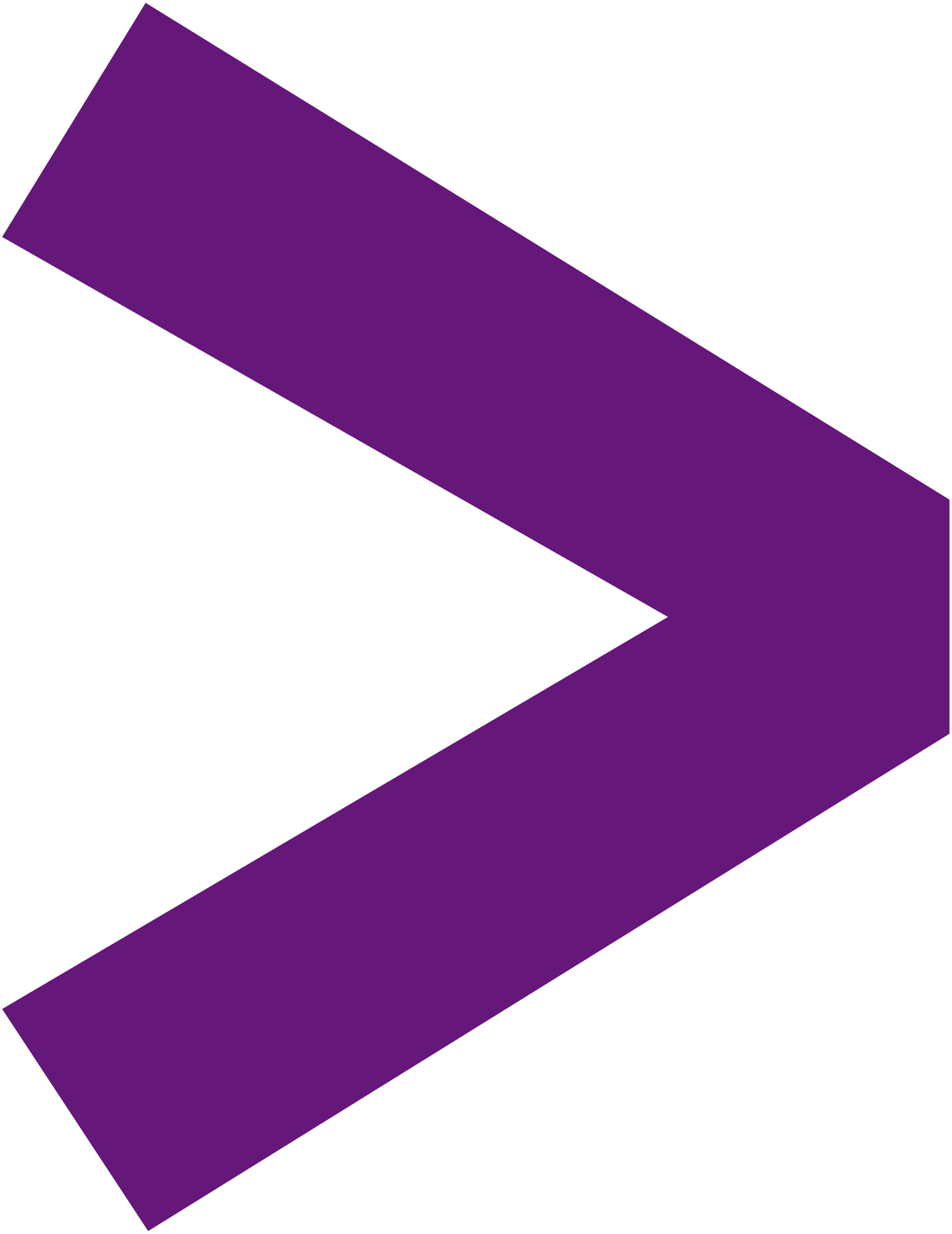
P

e

t

Z

լ



(

C

0

n

t

e

X

t

$=$

D

i

S

p

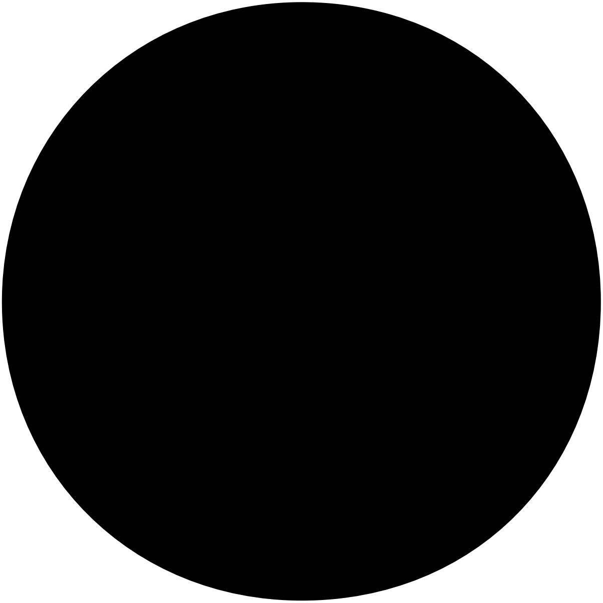
t

C

h

r

S



D

e

f

a

u

լ

t

,

C

a

p

a

i

t

y

$=$

1

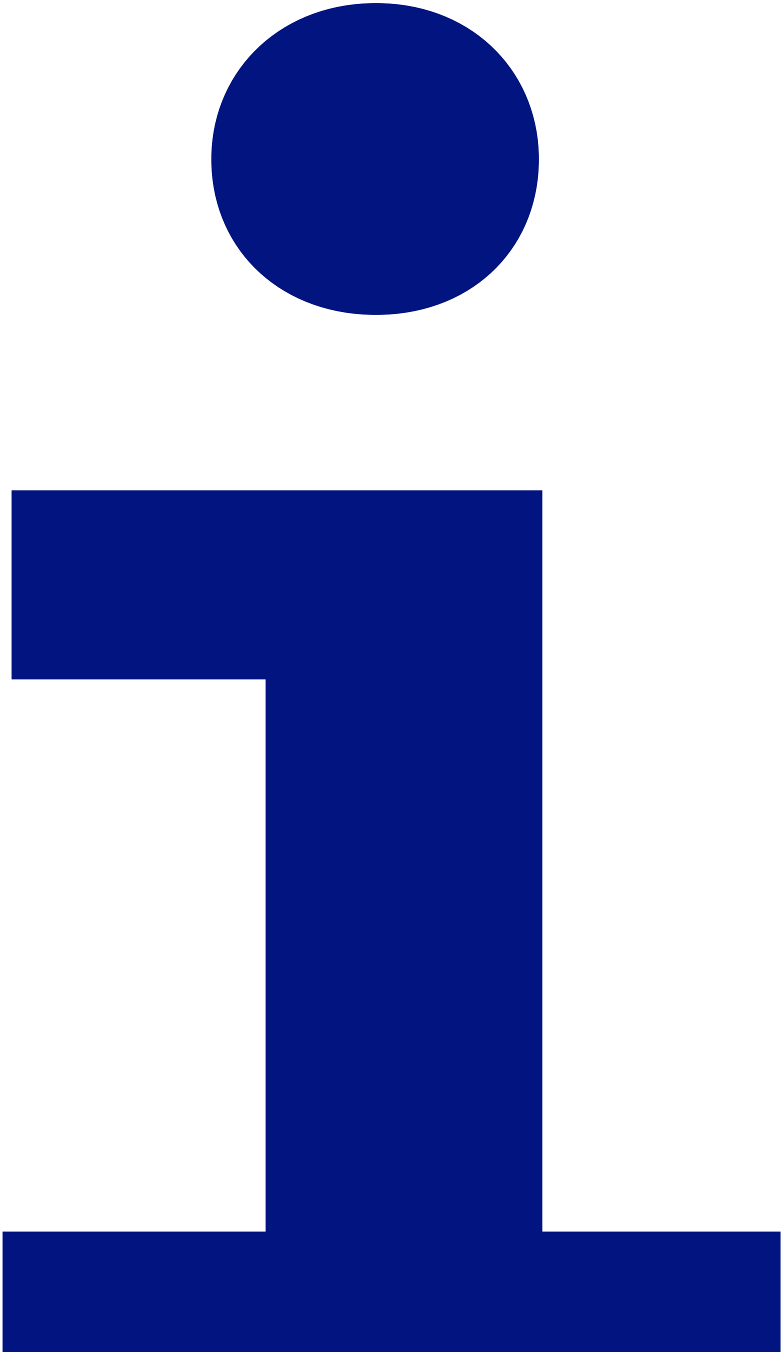
ʘ

)

{

W

h



ɩ

e

(

t

r

u

e

{

i

n

Ղ

n

S

t

a

r

t

p

r

0

d

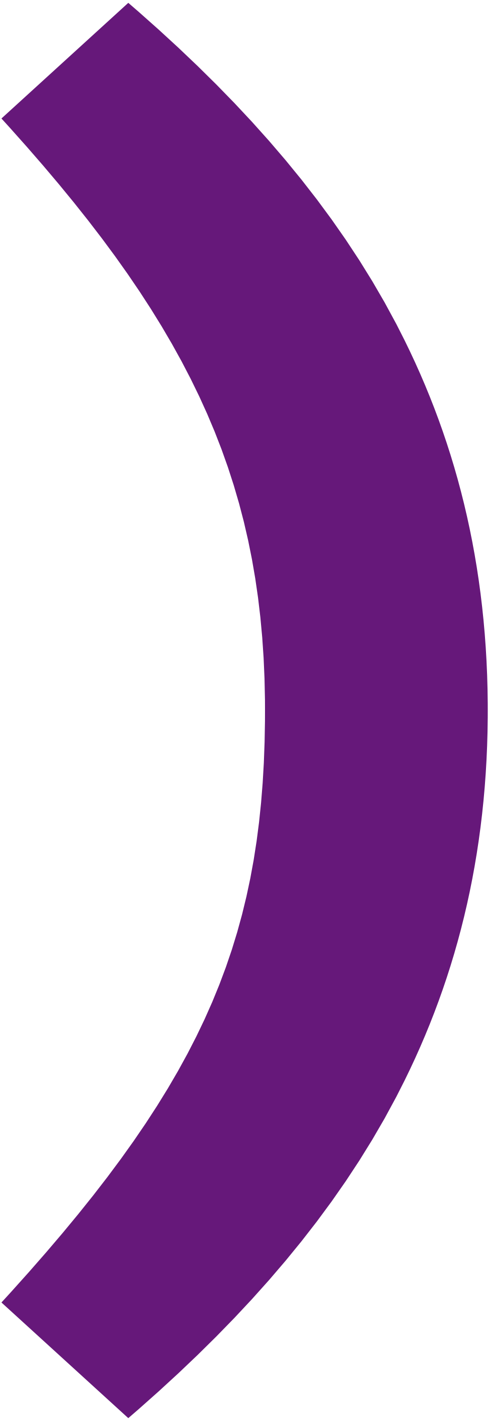
u

C

i

n

g



b

a

k

e

P

r

Z

e

●

f

O

r

E

a

{

d

i

t

)

i

n

Ղ

n

$

S

e

n

t

0

S

h

e

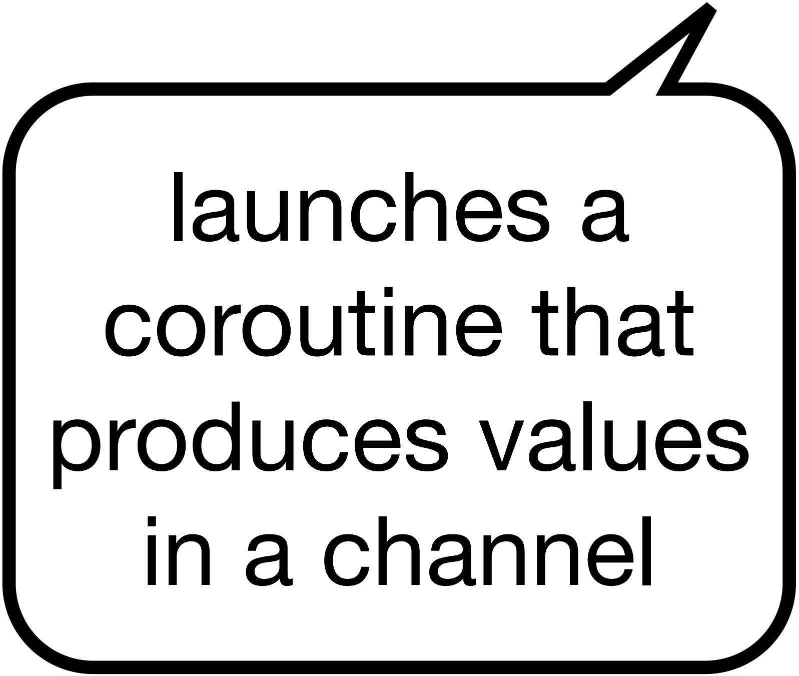
ι

f

}

}

}


launches a coroutine that produces values in a channel

```kotlin
fun CoroutineScope.bakeryPipeline() =
    produce<Pretzel>(context = Dispatchers.Default, capacity = 10) {
        while (true) {
            println("Start producing")
            bakePretzels().forEach {
                send(it)
                println("$it sent to shelf")
            }
        }
    }
```

Communication Between Coroutines

channels

```
fun CoroutineScope.bakeryPipeline() =
    produce<Pretzel>(context = Dispatchers.Default, capacity = 10) {
        while (true) {
            println("Start producing")
            bakePretzels().forEach {
                send(it)
                println("$it sent to shelf")
            }
        }
    }
```

launches a coroutine that produces values in a channel

Communication Between Coroutines

channels

```
fun CoroutineScope.cashDesk(shelf: ReceiveChannel<Pretzel>) = launch(IO) {
    while (isActive) {
        println("How many pretzels?")
        val count = readLine()?.toIntOrNull()
        if (count != null) {
            (1..count).map { shelf.receive() }
                .forEach {
                    print("Here's your pretzel: $it")
                }
        } else {
            println("Command not recognized")
        }
    }
}
```