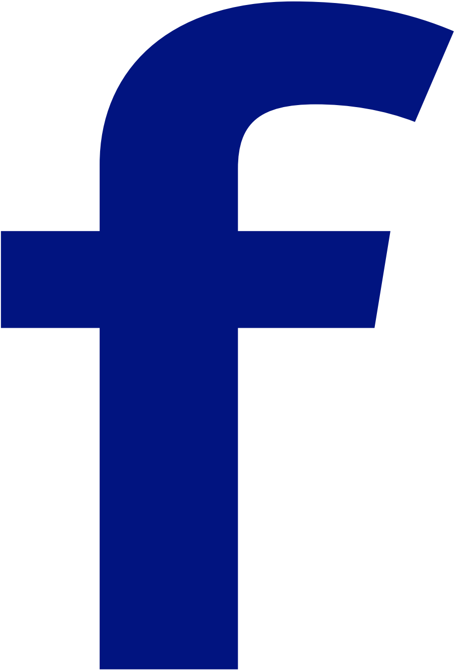
Dealing with null

why Kotlin doesn’t need `Optional`



u

n

m

e

a

S

u

r

L

e

n

g

t

h

S

●

●

S

t

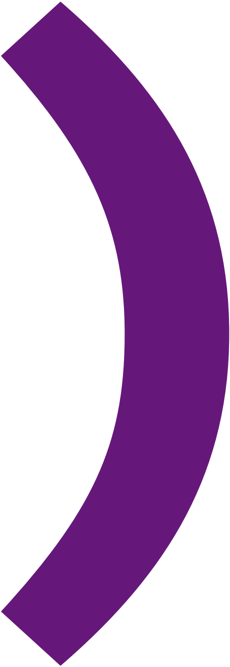
r

i

n

g

?



I

n

t

[redacted]

0

p

t

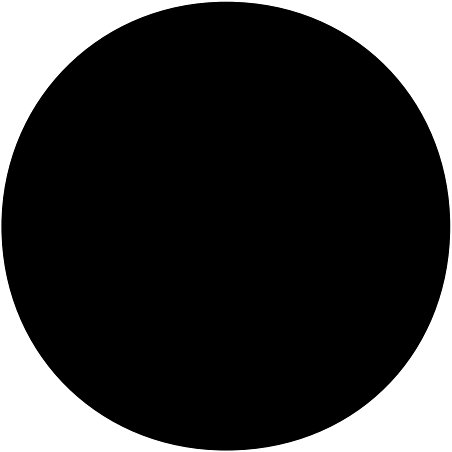
i

O

n

a

ɩ



f

N

u

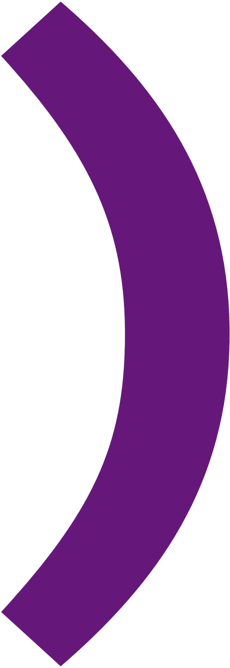
ʅ

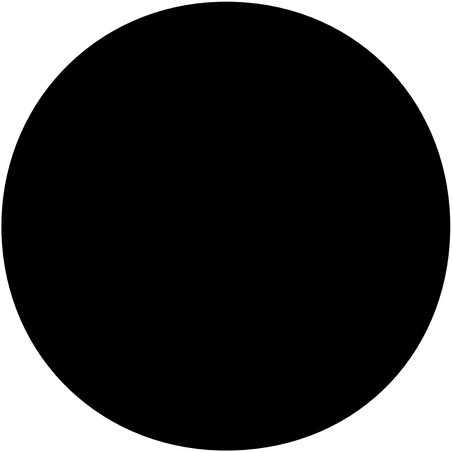
l

a

b

S





m

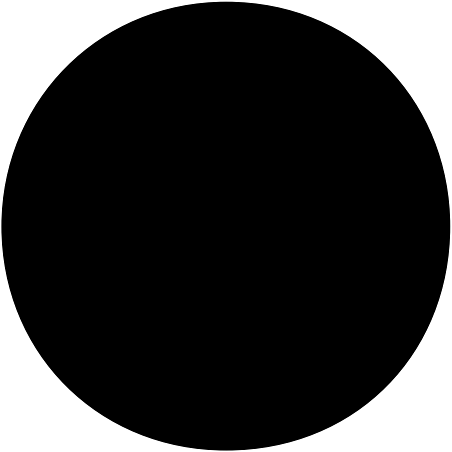
a

p

{

i

t



e

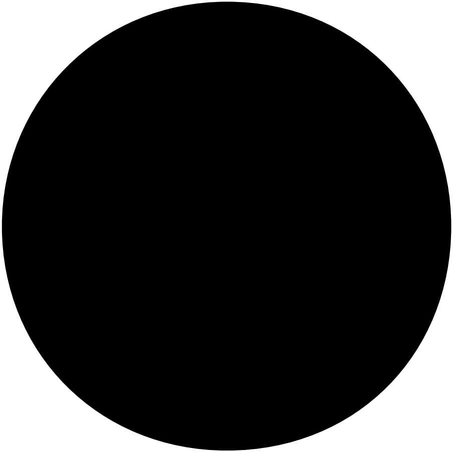
n

g

t

h

}

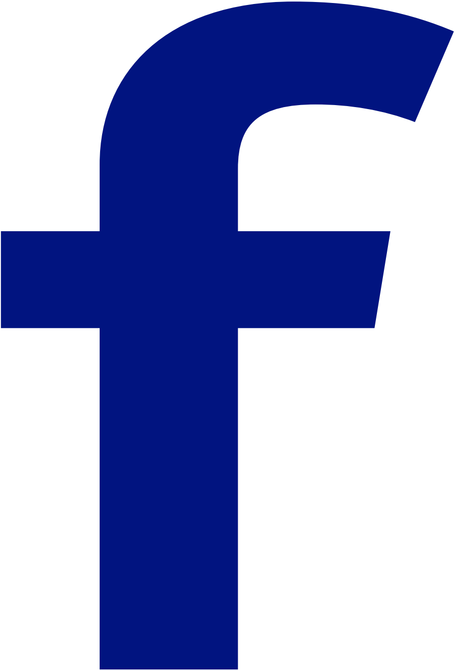


O

r

E

ʘ



u

n

m

e

a

S

u

r

L

e

n

g

t

h

S

●

●

S

t

r

i

n

g

?

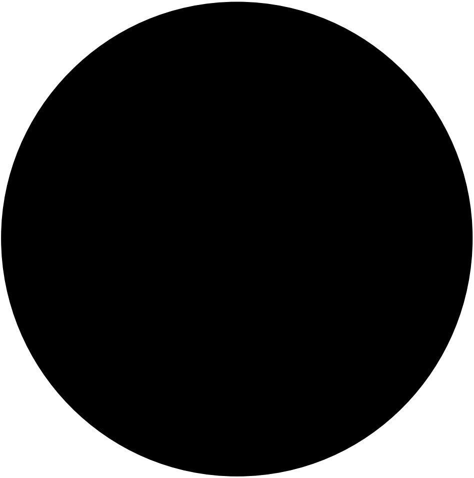
I

n

t

[redacted]

?



ւ

e

n

g

t

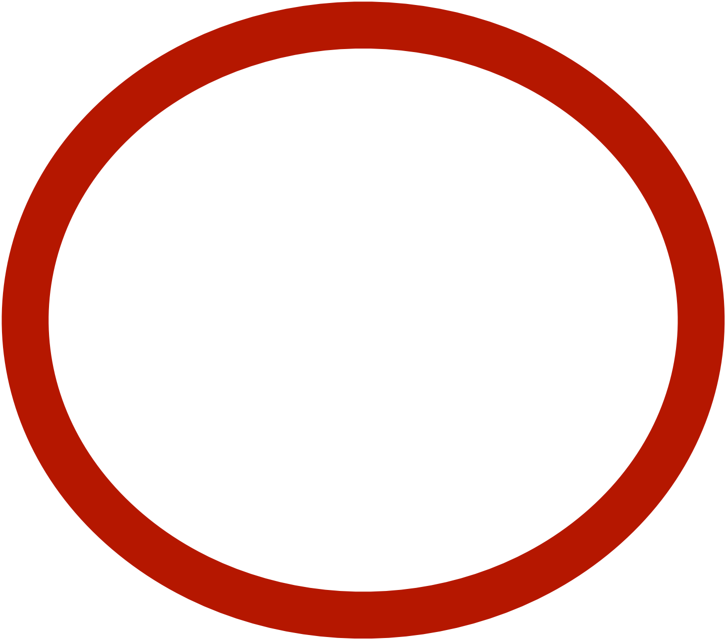
h

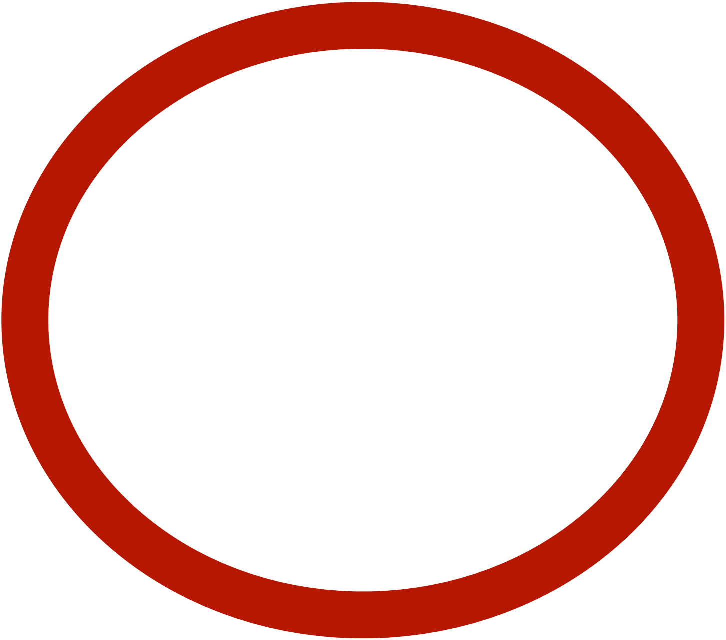
?

●

●

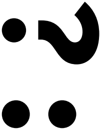
ʘ





null safe navigation

Elvis operator



```
fun measureLength(s: String?): Int =
    Optional.ofNullable(s).map { it.length }.orElse(0)
```

```
fun measureLength(s: String?): Int = s?.length ?: 0
```

Dealing with null

why Kotlin doesn’t need Optional

```
fun measureLength(s: String?): Int =
    Optional.ofNullable(s).map { it.length }.orElse(0)
```

```
fun measureLength(s: String?): Int = s?.length ?: 0
```

null safe navigation

Elvis operator

Dealing with null

why Kotlin doesn’t need Optional

```
fun censorToEmpty(s: String?, censor: (String) -> String?): String =
    Optional.ofNullable(s).flatMap { Optional.ofNullable(censor(it)) }.orElse("")
```

```
fun censorToEmpty2(s: String?, censor: (String) -> String?): String =
    s?.let(censor) ?: ""
```