Function type

Function type

```
val sum = { x: Int, y: Int -> x + y }
```

```
val sum: (Int, Int) -> Int = { x, y -> x + y }
```

What is the type of `isEven` variable?

```
val sum: (Int, Int) -> Int = { x, y -> x + y }
```

```
val isEven = { i: Int -> i % 2 == 0 }
```

What is the type of `isEven` variable?

```
val sum: (Int, Int) -> Int = { x, y -> x + y }
```

```
val isEven: (Int) -> Boolean =
        { i: Int -> i % 2 == 0 }
```

Storing lambda in a variable

```
val isEven = { i: Int -> i % 2 == 0 }

val list = listOf(1, 2, 3, 4)
list.any(isEven)                    true
list.filter(isEven)                 [2, 4]
```

Calling stored function

```
val isEven = { i: Int -> i % 2 == 0 }

isEven(42)          // true
```

Calling lambda

possible, but looks strange

```
{ println("hey!") }()
```

```
run { println("hey!") }
```

use **run** instead

Function types: under the hood

```
() -> Boolean          Function0<Boolean>

(Order) -> Int         Function1<Order, Int>

(Int, Int) -> Int      Function2<Int, Int, Int>
```

Function interfaces

```
package kotlin.jvm.functions

/** A function that takes 0 arguments. */
public interface Function0<out R> : Function<R> {
    /** Invokes the function. */
    public operator fun invoke(): R
}
/** A function that takes 1 argument. */
public interface Function1<in P1, out R> : Function<R> {
    /** Invokes the function with the specified argument. */
    public operator fun invoke(p1: P1): R
}
/** A function that takes 2 arguments. */
public interface Function2<in P1, in P2, out R> : Function<R> {
    /** Invokes the function with the specified arguments. */
    public operator fun invoke(p1: P1, p2: P2): R
}
```

Function types: under the hood

```
() -> Boolean          Function0<Boolean>

(Order) -> Int         Function1<Order, Int>

(Int, Int) -> Int      Function2<Int, Int, Int>
```

SAM interfaces in Java

```
void postponeComputation(int delay, Runnable computation)
```

SAM (single abstract method) interface:

```
public interface Runnable {
    public abstract void run();
}
```

Lambdas and Java

```
void postponeComputation(int delay, Runnable computation)
```

```
postponeComputation(1000) { println(42) }
```



```
val runnable = Runnable { println(42) }
postponeComputation(1000, runnable)
```

Function types and nullability

`() -> Int?` vs `(() -> Int)?`

Which lines won’t compile?

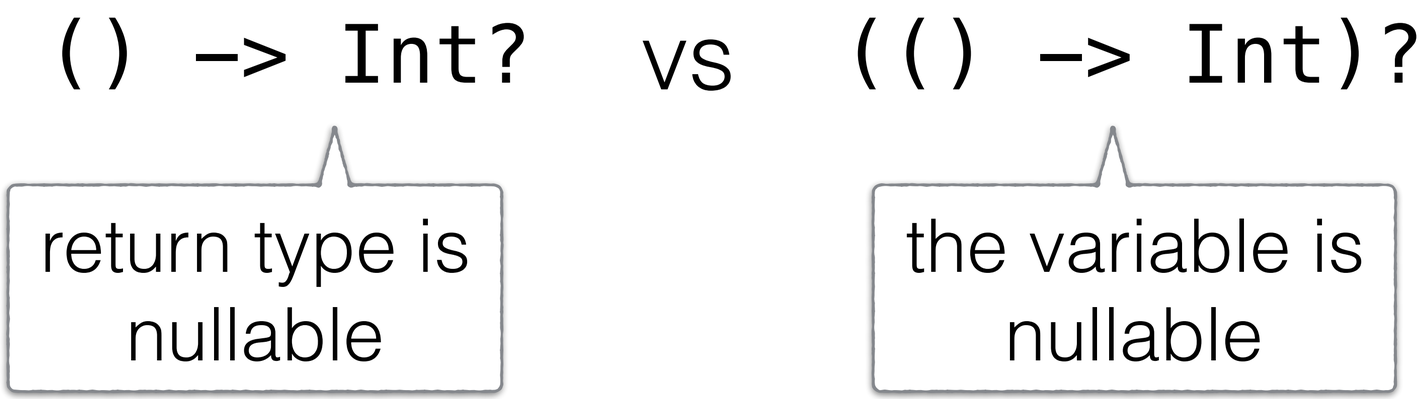
```
#1   val f1: () -> Int? = null

#2   val f2: () -> Int? = { null }

#3   val f3: (() -> Int)? = null

#4   val f4: (() -> Int)? = { null }
```

Function types and nullability



Which lines won’t compile?

```
#1   val f1: () -> Int? = null
#2   val f2: () -> Int? = { null }
#3   val f3: (() -> Int)? = null
#4   val f4: (() -> Int)? = { null }
```

#1, #4

When function type is nullable

```
val f: (() -> Int)? = null

f()


if (f != null) {
    f()
}

f?.invoke()
```