Kotlin cheatsheet

Concise. Multiplatform. Fun.

Why Kotlin?

- Cross platform
- Statically typed
- High level
- Type inference
- Functional
- Null safe
- Structured concurrency
- No semi-colons!

Learn more at **kotlinlang.org**

Basic types

```
// Integers
val beers: Byte = 5
val cash: Short = -12
val fries: Int = 3_800
val waffles: Long = 99

// Unsigned
val cash: UShort = 12u

// Floating
val temp: Float = 2.5f
val lotsOfWaffles:
Double = 19.2e5

// Boolean
val enabled: Boolean =
true

// Character
val sep: Char = ','

// String
val message: String =
"Hello, world!"

// String Templates
print("I ate $waffles
waffles")
// "I ate 99 waffles"
```

Special types

```
// Type with a single
// value
Unit
// Type with no value
Nothing
// Unified supertype
Any
```

Collections

```
// Lists
val beers:
List<String> =
listOf("Delirium",
"Duvel", "Chimay")

// Sets
val ids: Set<Int> =
setOf(1, 5, 7)

// Maps
val cities:
Map<String, Int> =
mapOf(
  "Brussels" to 1249,
  "Antwerp" to 520,
)
```

Functions

```
fun sum(
  x: Int,
  y: Int
): Int {
    return x + y
}

// Function type
val s: (Int, Int) →
Unit = { x, y →
  x + y
}
```

Interfaces

```
interface Shape {
  val w: Int
  val h: Int
  fun area(): Int
}
```

Classes

```
class Rect(
  override val w: Int,
  override val h: Int,
): Shape {
  override fun area():
Int {
    return w * h
  }
}
```

Null safety

```
// Never null
var neverNull: String
= "This can't be null"
// May be null
var nullable: String?
= "You can keep a null
here"

// safe call
nullable?.take(3)
// "You"

// This is OK
nullable = null

// Elvis operator
nullable ?: "Default"
// "Default"

// Not-null assertion
nullable!!.length
// Crash 💥
```