KOTLIN

EN INTRODUKSJON

MORTEN NYGAARD ÅSNES

HISTORIE

- Statisk typet programmeringsspråk
  - JVM
  - JavaScript
  - Native (LLVM)
- Multi Paradigm
- JetBrains

FUNKSJONER

```kotlin
fun main(args: Array<String>) {
    println("Hello, World!")
}

fun max(a: Int, b: Int): Int {
    return if (a > b) a else b
}

fun sum(a: Int, b: Int) = a + b
```

VARIABLER

IMMUTABLE

```kotlin
val a: Int = 1
val b = 2
val c: Int
c = 3
```

MUTABLE

```kotlin
var sum: Int = 0
sum = a + b

var s = "$sum = ${a + b}"
```

NULLABLE

```kotlin
val x: String = null // Does not compile

val x: String? = null // Ok

val y: String? = if (Random().nextBoolean()) "foo" else null

println(y.length) // Unsafe, does not compile

println(y!!.length) // Unsafe, compiles
```

NULLABLE

```kotlin
val y: String? = if (Random().nextBoolean()) "foo" else null

if (y != null) {
    println(y.length) // safe
}

y.let {               // safe
    println(it)
}

val l = if (y != null) y.length else -1

val m = y?.length ?: -1
```

SMART CAST

```kotlin
if (obj is Person) {
    print(obj.name)
}

if (x is Person && x.age > 20) return

val x = when (something) {
    is String -> something.length
    is Int -> something
    is List<*> -> something.size
    else -> -1
}
```

CLASSES

```
class Event { }
class Message

class Student {

    private val name: String

    constructor(name: String) {
        this.name = name
    }
}

class Person(val firstName: String)
class Thing(val name: String, var age: Int, val type: Type)
```

DATA CLASSES

```kotlin
data class User(val userName: String, val password: String)

data class Book(val title: String, val author: String,
                val year: Int = -1,
                val sortedUnder: String = author)

val user1 = User("Morten", "*******")
val user2 = User(userName = "Morten", password = "*******")
```

LAMBDA

```kotlin
fun doIt(value: Int, body: (Int) -> Int): Int {
    return body(value)
}

doIt(10, { v -> 16 * v})
doIt(20, { it * 8 })

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

LAMBDA

```kotlin
val list: List<Int> = listOf(10, 11, 12, 13, 14, 15)
val sum = list.filter { i -> i > 12 }.map { i -> i * 2 }.sum()
val max = list.filter { it.rem(2) == 0 }.max()

val things = listOf("A", 42, 3.14, "Foo", Pair(1, 2))
val sum1 = things
        .filter { it is Int }
        .map { it as Int }
        .map { it * 2 }
        .sum()
```

DSL

```
html {
    head {
        title { +"The Title" }
    }
    body {
        h1 { +"Some Header" }
        p { +"Paragraph" }

        p {
            +"This is some"
            b { +"mixed" }
            +"text. For more see the"
        }
        p { +"some text" }

        p {
```