



Conditionals: `if` & `when`

if is an expression

```
val max = if (a > b) a else b
```

No ternary operator in Kotlin: ~~(a > b) ? a : b~~



when as switch

```
enum class Color {  
    BLUE, ORANGE, RED  
}
```


```
fun getWarmth(color: Color): String =  
    when (color) {  
        BLUE -> "cold"  
        ORANGE -> "mild"  
        RED -> "hot"  
    }
```

No break is needed

```
switch (color) {  
  case BLUE:  
    System.out.println("cold");  
    break;  
  
  case ORANGE:  
    System.out.println("mild");  
    break;  
  
  default:  
    System.out.println("hot");  
}
```



```
when (color) {  
  BLUE -> println("cold")  
  ORANGE -> println("mild")  
  else -> println("hot")  
}
```



when syntax

check several values at once

```
fun whenSyntax(a: Any) = when (a) {  
    0, 1 -> "is zero or one"  
    is Boolean -> "is boolean"  
    is String -> "is string of length ${a.length}"  
    else -> "other"  
}
```

smart cast


Smart casts

```
String result;
```

```
...  
if (a instanceof String) {  
    result = "is string of length " + ((String) a).length();  
}
```

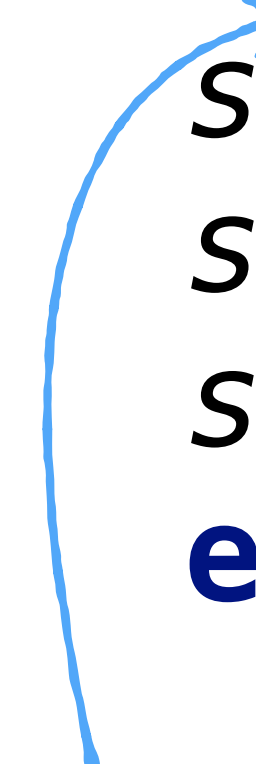


```
val result = when (a) {  
    0 -> "is zero"  
    is Int -> "is integer"  
    is String -> "is string of length ${a.length}"  
    else -> "other"  
}
```



Any expression can be an argument

```
fun mix(c1: Color, c2: Color) =  
    when (setOf(c1, c2)) {  
        setOf(RED, YELLOW) -> ORANGE  
        setOf(YELLOW, BLUE) -> GREEN  
        setOf(BLUE, VIOLET) -> INDIGO  
        else -> throw Exception("Dirty color")  
    }
```



the argument is checked for
equality with the branch conditions

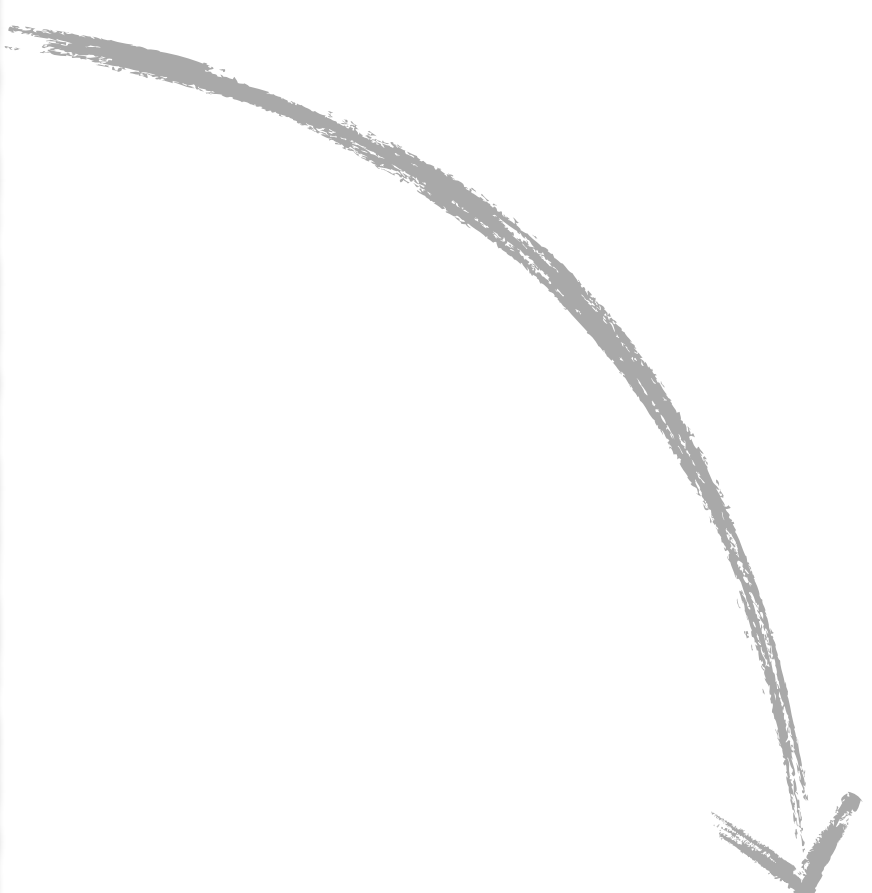

when without argument

```
fun updateWeather(degrees: Int) {  
  val (description, colour) = when {  
    degrees < 5 -> "cold" to BLUE  
    degrees < 23 -> "mild" to ORANGE  
    else -> "hot" to RED  
  }  
}
```


no argument

any Boolean expression


```
String description;  
Colour colour;  
if (degrees < 5) {  
    description = "cold";  
    colour = BLUE;  
} else if (degrees < 23) {  
    description = "mild";  
    colour = ORANGE;  
} else {  
    description = "hot";  
    colour = RED;  
}
```



```
val (description, colour) = when {  
    degrees < 5 -> "cold" to BLUE  
    degrees < 23 -> "mild" to ORANGE  
    else -> "hot" to RED  
}
```





You always should replace `if` with `when`

True or false?





You always should replace `if` with `when`

1. `true`

2. `false`