More useful library functions

More useful library functions

```
inline fun <T, R> with(receiver: T, block: T.() -> R): R = receiver.block()

inline fun <T, R> T.run(block: T.() -> R): R = block()

inline fun <T, R> T.let(block: (T) -> R): R = block(this)

inline fun <T> T.apply(block: T.() -> Unit): T { block(); return this }

inline fun <T> T.also(block: (T) -> Unit): T { block(this); return this }
```

with

```
with (window) {
    width = 300
    height = 200
    isVisible = true
}
```

run: like `with`, but extension

```
val windowOrNull = windowById["main"]
windowOrNull?.run {
    width = 300
    height = 200
    isVisible = true
}
```

run: like `with`, but extension

```
windowById["main"]?.run {
    width = 300
    height = 200
    isVisible = true
}
```

apply: returns receiver as a result

```kotlin
val mainWindow =
    windowById["main"]?.apply {
        width = 300
        height = 200
        isVisible = true
    } ?: return
```

`also`: regular argument instead of `this`

```
windowById["main"]?.apply {
    width = 300
    height = 200
    isVisible = true
}?.also {
    showWindow(it)
}
```

	`{ .. this .. }`	`{ .. it .. }`
return result of lambda	`run`	`let`
return receiver	`apply`	`also`

```
receiver.apply {
    this.actions()
}
```

```
receiver.also {
    moreActions(it)
}
```

Find the correspondence between the functions and their implementations

```
inline fun <T, R> T.run(block: T.() -> R): R
inline fun <T, R> T.let(block: (T) -> R): R
inline fun <T> T.apply(block: T.() -> Unit): T
inline fun <T> T.also(block: (T) -> Unit): T
```

```
{ block(this); return this }
{ this.block(); return this }
{ return this.block() }
{ return block(this) }
```

?

	`{ .. this .. }`	`{ .. it .. }`
return result of lambda	`run`	`let`
return receiver	`apply`	`also`

```
inline fun <T, R> T.run(block: T.() -> R): R
inline fun <T, R> T.let(block: (T) -> R): R
inline fun <T> T.apply(block: T.() -> Unit): T
inline fun <T> T.also(block: (T) -> Unit): T
```

```
{ block(this); return this }
{ this.block(); return this }
{ return this.block() }
{ return block(this) }
```

	{ .. this .. }	{ .. it .. }
return result of lambda	run	let
return receiver	apply	also

```
inline fun <T, R> T.run(block: T.() -> R): R
inline fun <T, R> T.let(block: (T) -> R): R
inline fun <T> T.apply(block: T.() -> Unit): T
inline fun <T> T.also(block: (T) -> Unit): T
```

```
                              { block(this); return this }
                              { this.block(); return this }
                              { return this.block() }
                              { return block(this) }
```

	`{ .. this .. }`	`{ .. it .. }`
return result of lambda	`run`	`let`
return receiver	`apply`	`also`

```
inline fun <T, R> T.run(block: T.() -> R): R
inline fun <T, R> T.let(block: (T) -> R): R
inline fun <T> T.apply(block: T.() -> Unit): T
inline fun <T> T.also(block: (T) -> Unit): T
```

```
{ block(this); return this }
{ this.block(); return this }
{ return this.block() }
{ return block(this) }
```

	`{ .. this .. }`	`{ .. it .. }`
return result of lambda	`run`	`let`
return receiver	`apply`	`also`

```
inline fun <T, R> T.run(block: T.() -> R): R
inline fun <T, R> T.let(block: (T) -> R): R
                              { return this.block() }
                              { return block(this) }

inline fun <T> T.apply(block: T.() -> Unit): T
inline fun <T> T.also(block: (T) -> Unit): T
                              { block(this); return this }
                              { this.block(); return this }
```

	{ .. this .. }	{ .. it .. }
return result of lambda	run	let
return receiver	apply	also

```
inline fun <T, R> T.run(block: T.() -> R): R
inline fun <T, R> T.let(block: (T) -> R): R
                              { return this.block() }
                              { return block(this) }

inline fun <T> T.apply(block: T.() -> Unit): T
inline fun <T> T.also(block: (T) -> Unit): T
                              { block(this); return this }
                              { this.block(); return this }
```

	`{ .. this .. }`	`{ .. it .. }`
return result of lambda	`run`	`let`
return receiver	`apply`	`also`

```
inline fun <T, R> T.run(block: T.() -> R): R
inline fun <T, R> T.let(block: (T) -> R): R
                                  { return this.block() }
                                  { return block(this) }

inline fun <T> T.apply(block: T.() -> Unit): T
inline fun <T> T.also(block: (T) -> Unit): T
                                  { block(this); return this }
                                  { this.block(); return this }
```

	`{ .. this .. }`	`{ .. it .. }`
return result of lambda	`run`	`let`
return receiver	`apply`	`also`

```kotlin
inline fun <T, R> T.run(block: T.() -> R): R { return this.block() }

inline fun <T, R> T.let(block: (T) -> R): R { return block(this) }

inline fun <T> T.apply(block: T.() -> Unit): T { this.block(); return this }

inline fun <T> T.also(block: (T) -> Unit): T { block(this); return this }
```