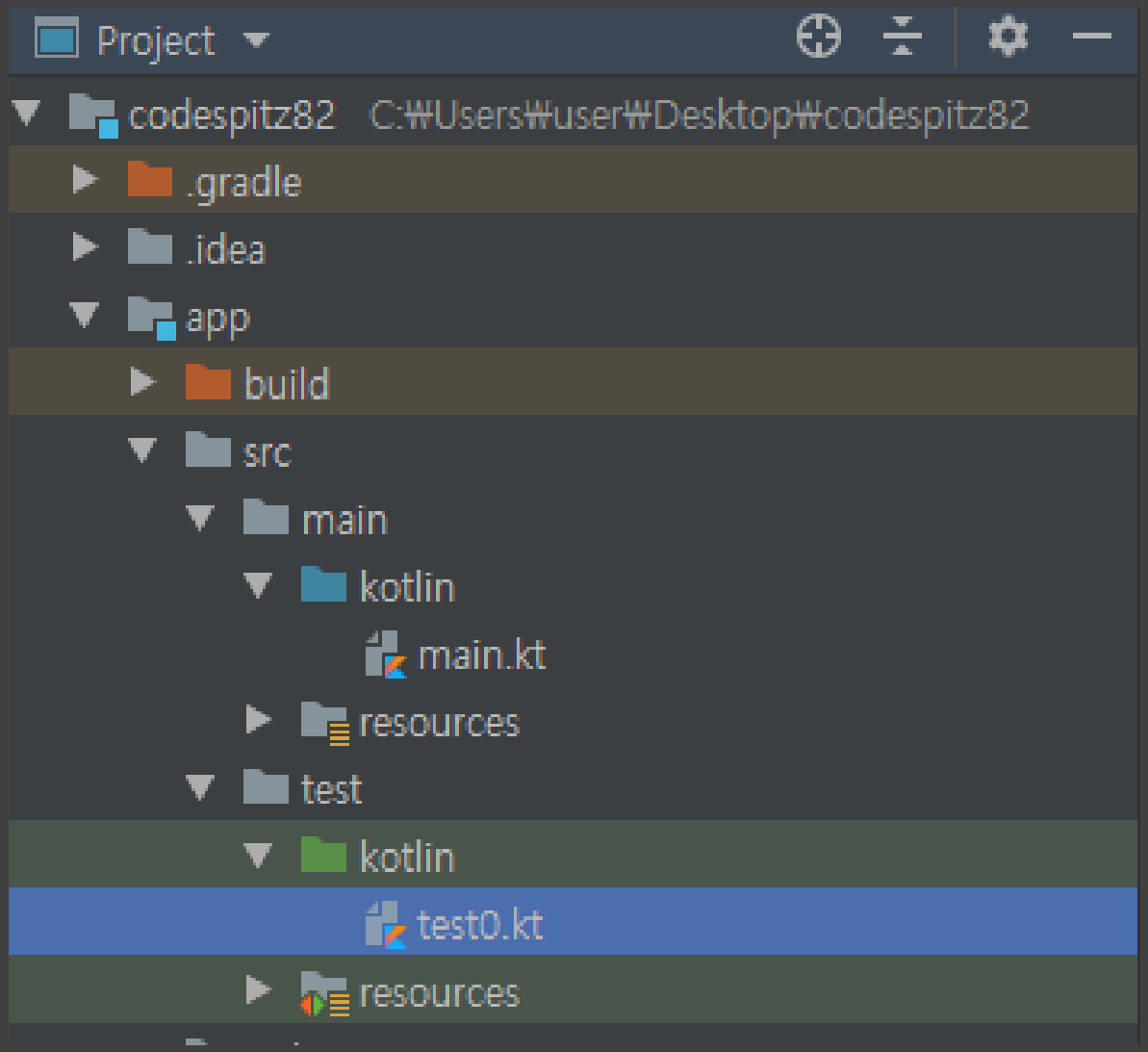
KOTLIN ELEMENTARY

1 2 3 4 5 6

unit test

test0.kt


Project
codespitz82 C:₩Users₩user₩Desktop₩codespitz82
.gradle
.idea
app
build
src
main
kotlin
main.kt
resources
test
kotlin
test0.kt
resources

test0.kt

```kotlin
import kotlin.test.Test
import kotlin.test.assertEquals

class SimpleTest {
    @Test
    fun testTest(){
        assertEquals( expected: "hello", test())
    }
}
```

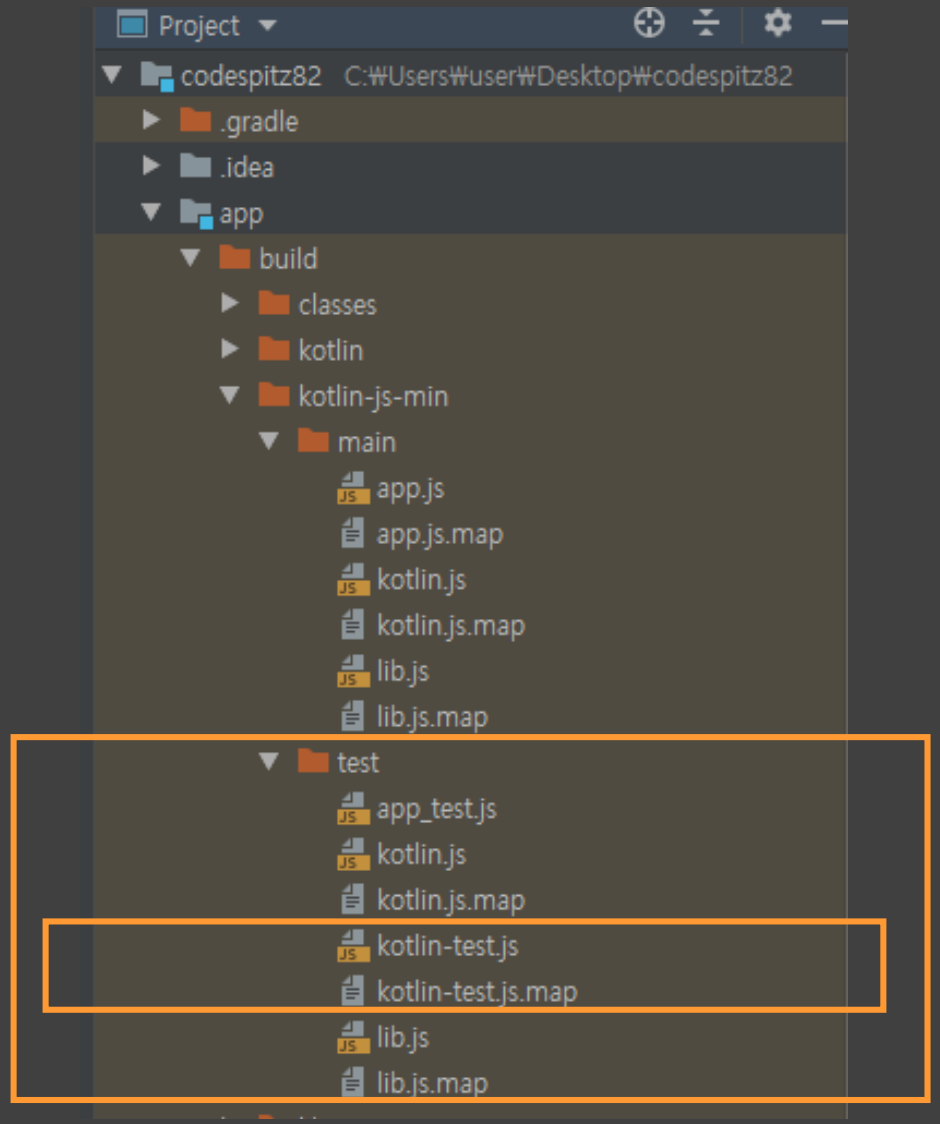
lib/hello.kt

```
fun hello(){
    println("hello world")
}
fun test() : String  = "abc"
```

DCE build only

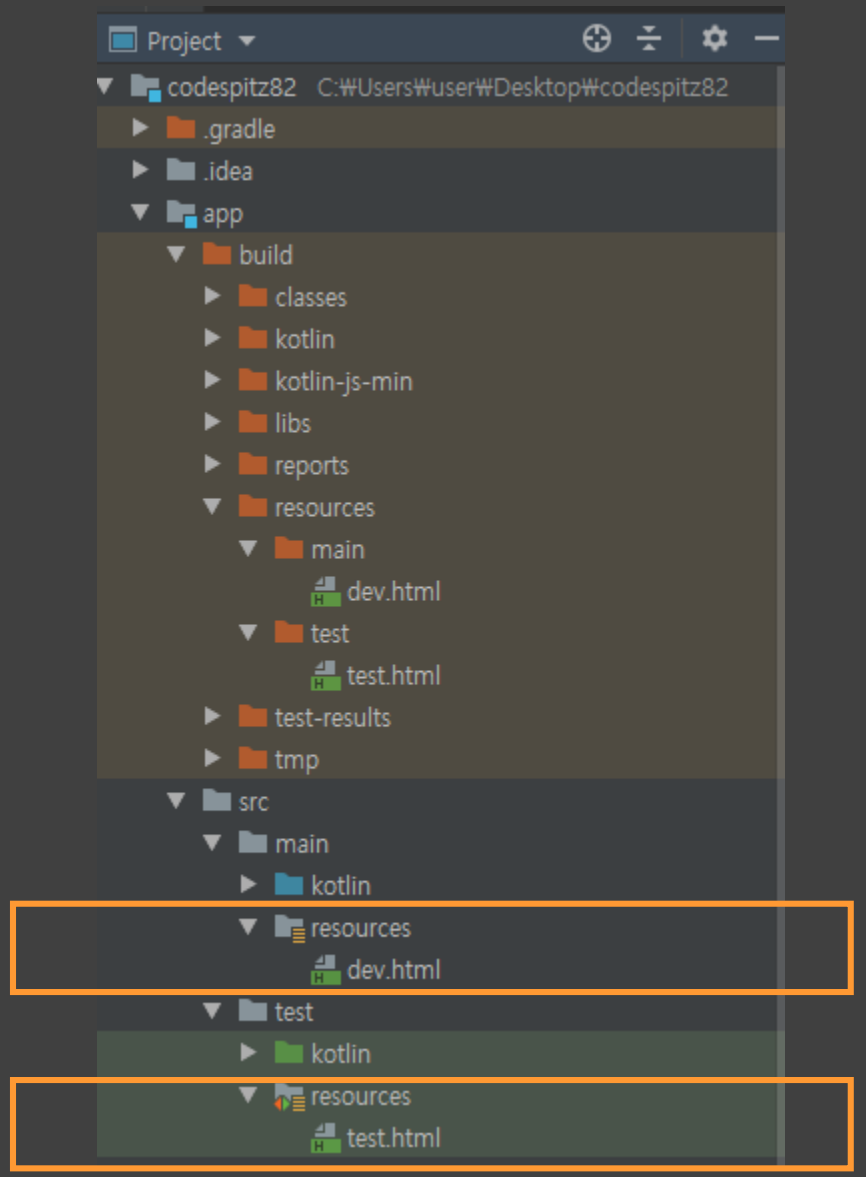
```
buildscript {
    ext.kotlin_version = '1.3.31'
    repositories {
        mavenCentral()
    }
    dependencies {
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"
    }
}

apply plugin: 'kotlin2js'
apply plugin: 'kotlin-dce-js'

repositories {
    mavenCentral()
}

dependencies {
    compile "org.jetbrains.kotlin:kotlin-stdlib-js:$kotlin_version"
    testImplementation "org.jetbrains.kotlin:kotlin-test-js:$kotlin_version"
    compile project(":lib")
}
compileKotlin2Js {
    kotlinOptions.sourceMap = true
}
```

build result

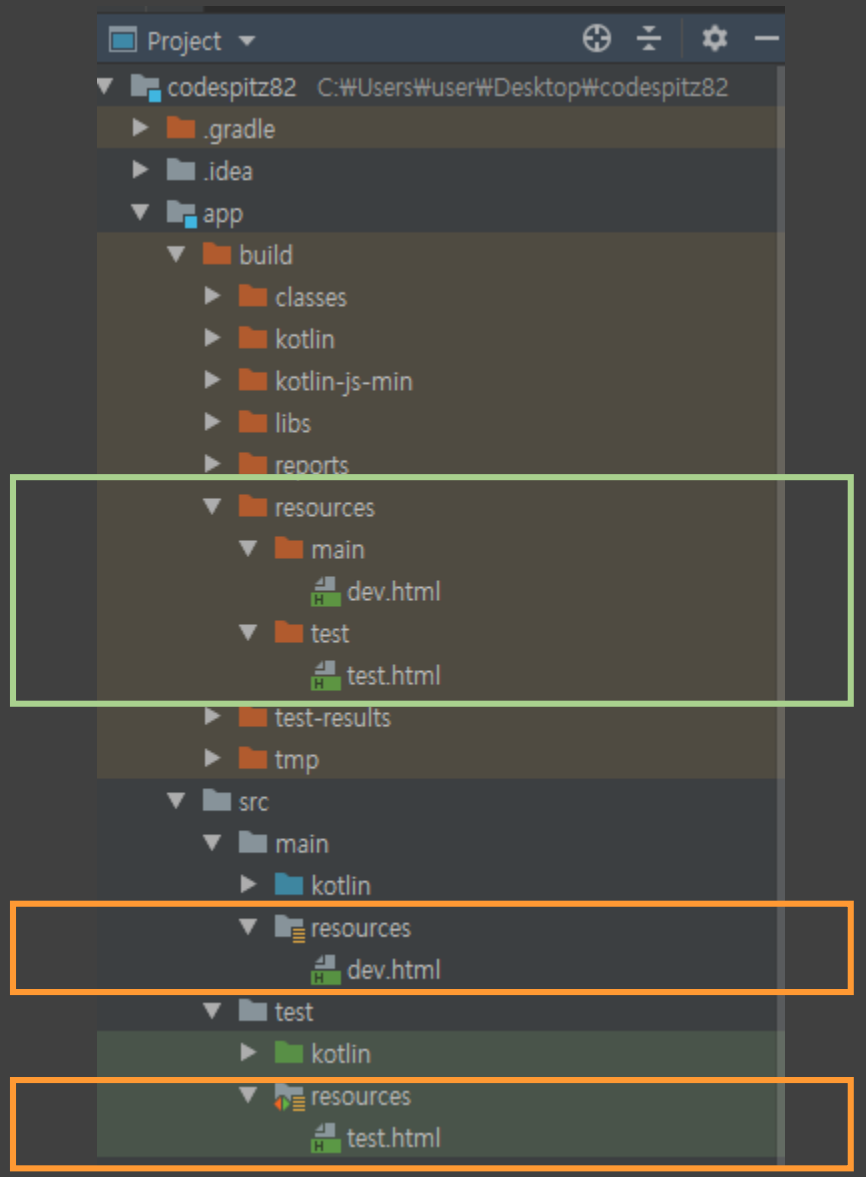


resource

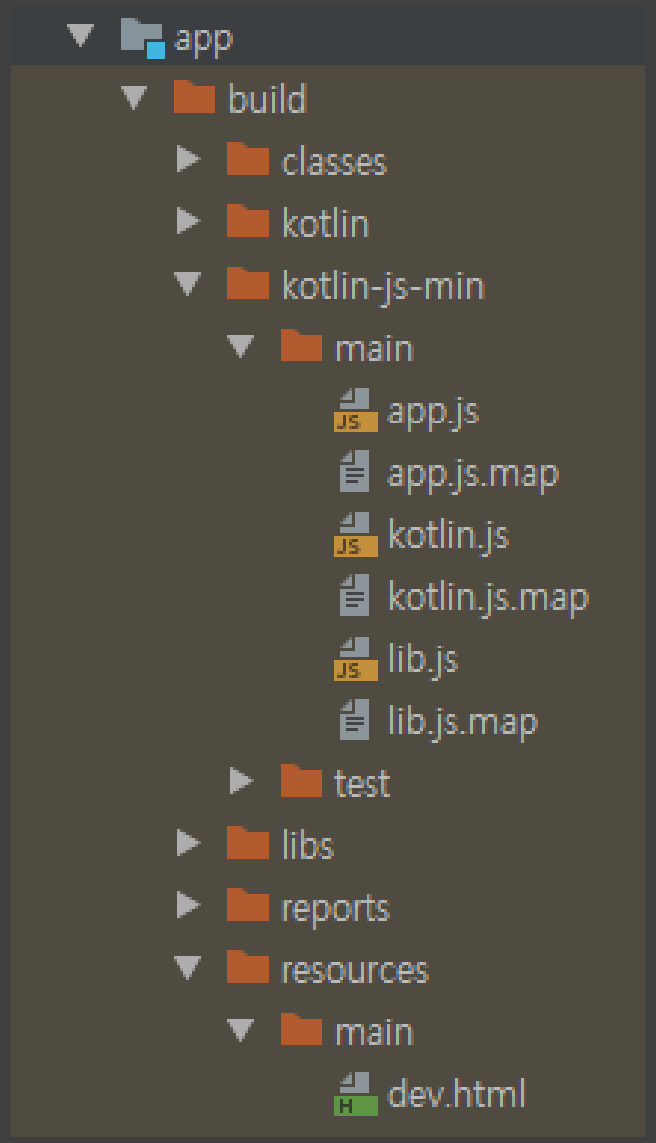
Resources


Project
codespitz82 C:₩Users₩user₩Desktop₩codespitz82
.gradle
.idea
app
build
classes
kotlin
kotlin-js-min
libs
reports
resources
main
dev.html
test
test.html
test-results
tmp
src
main
kotlin
resources
dev.html
test
kotlin
resources
test.html

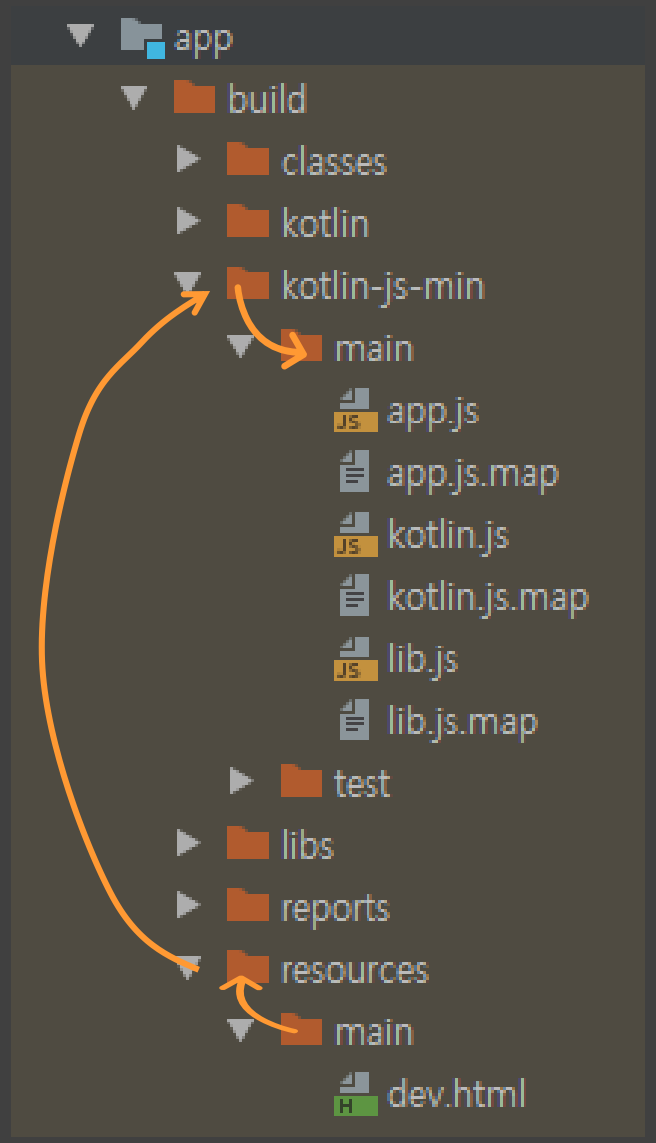
Resources


Project
codespitz82 C:₩Users₩user₩Desktop₩codespitz82
.gradle
.idea
app
build
classes
kotlin
kotlin-js-min
libs
reports
resources
main
dev.html
test
test.html
test-results
tmp
src
main
kotlin
resources
dev.html
test
kotlin
resources
test.html

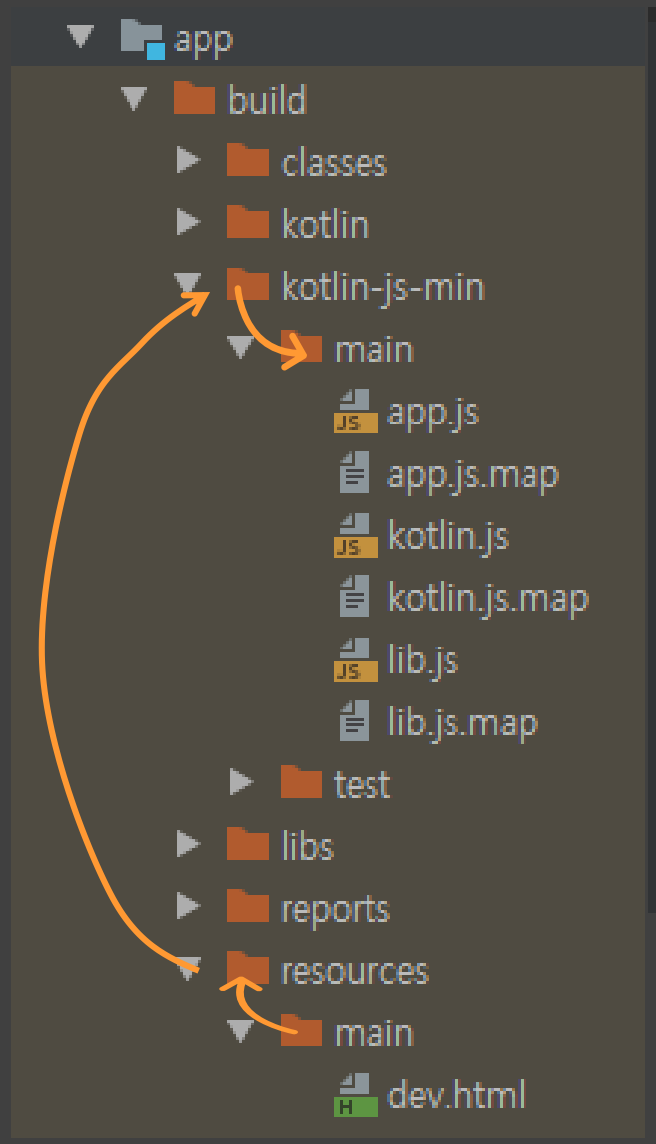
main/resource/dev.html



main/resource/dev.html

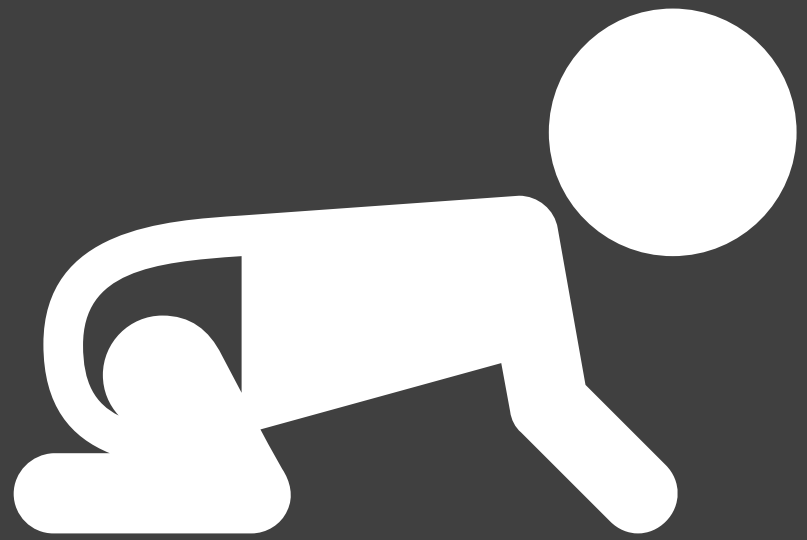


main/resource/dev.html




```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<script src="../../kotlin-js-min/main/kotlin.js"></script>
<script src="../../kotlin-js-min/main/lib.js"></script>
<script src="../../kotlin-js-min/main/app.js"></script>
</body>
</html>
```

test runner



kotlin-test.js

```
  NAME_TO_ADAPTER = mapOf([to('qunit', getCallableRef('QUnitAdapter', function () {
    return new QUnitAdapter();
  })), to('jasmine', getCallableRef('JasmineLikeAdapter', function () {
    return new JasmineLikeAdapter();
  })), to('mocha', getCallableRef('JasmineLikeAdapter', function () {
    return new JasmineLikeAdapter();
  })), to('jest', getCallableRef('JasmineLikeAdapter', function () {
    return new JasmineLikeAdapter();
  })), to('auto', getCallableRef('detectAdapter', function () {
    return detectAdapter();
  }))]);
  return _;
}));
```

kotlin-test.js

```
  NAME_TO_ADAPTER = mapOf([to('gunit', getCallableRef('QUnitAdapter', function () {
    return new QUnitAdapter();
  })), to('jasmine', getCallableRef('JasmineLikeAdapter', function () {
    return new JasmineLikeAdapter();
  })), to('mocha', getCallableRef('JasmineLikeAdapter', function () {
    return new JasmineLikeAdapter();
  })), to('jest', getCallableRef('JasmineLikeAdapter', function () {
    return new JasmineLikeAdapter();
  })), to('auto', getCallableRef('detectAdapter', function () {
    return detectAdapter();
  }))]);
  return _;
}));
```

https://qunitjs.com/

Getting Started

In The Browser

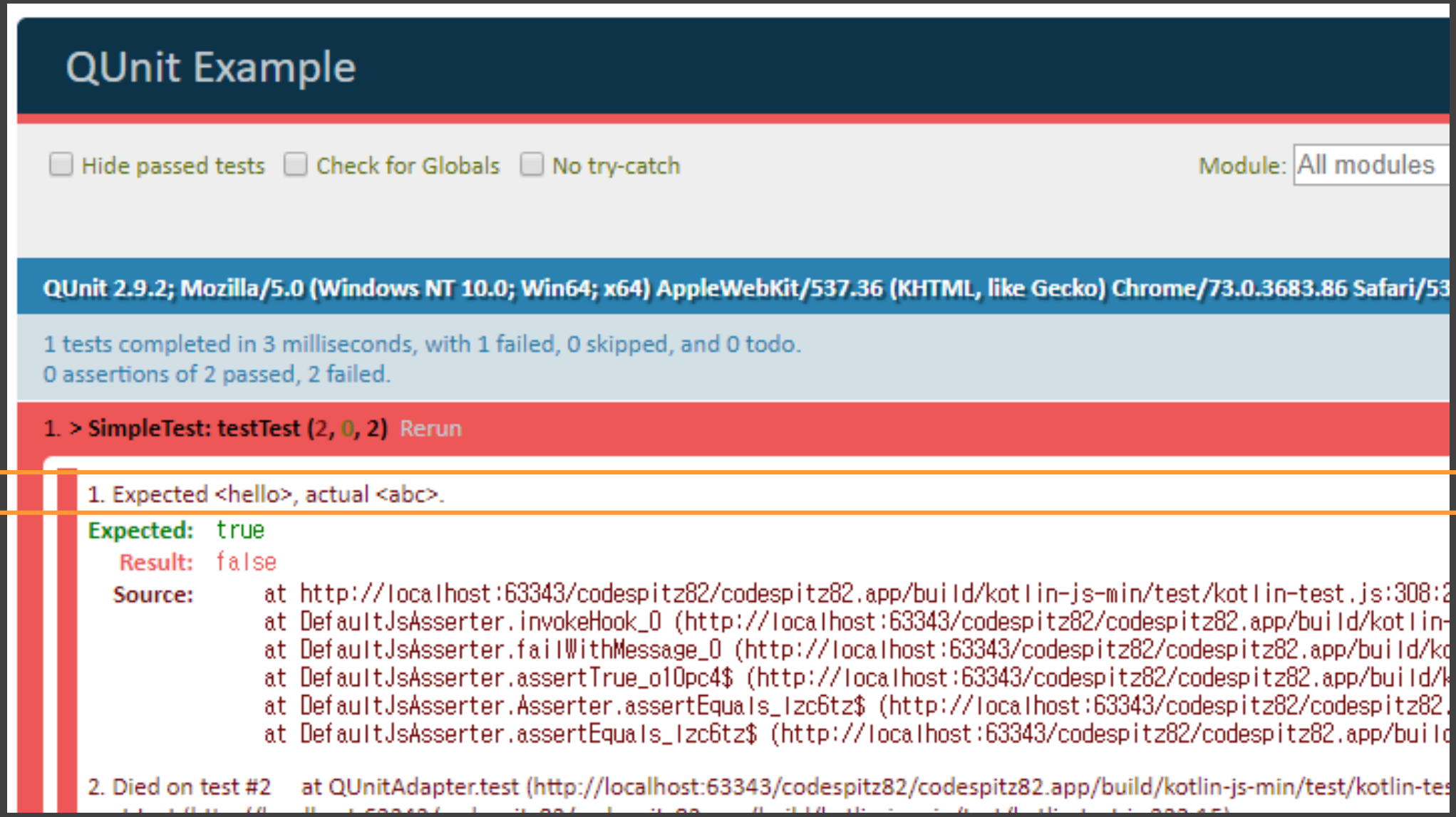
A minimal QUnit test setup:

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4    <meta charset="utf-8">
5    <meta name="viewport" content="width=device-width">
6    <title>QUnit Example</title>
7    <link rel="stylesheet" href="https://code.jquery.com/qunit/qu
8  </head>
9  <body>
10   <div id="qunit"></div>
11   <div id="qunit-fixture"></div>
12   <script src="https://code.jquery.com/qunit/qunit-2.9.2.js"></
13   <script src="tests.js"></script>
14 </body>
15 </html>
```

test/resource/test.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width">
    <title>QUnit Example</title>
    <link rel="stylesheet" href="https://code.jquery.com/qunit/qunit-2.9.2.css">
</head>
<body>
<div id="qunit"></div>
<div id="qunit-fixture"></div>
<script src="https://code.jquery.com/qunit/qunit-2.9.2.js"></script>
<script src="../../kotlin-js-min/test/kotlin.js"></script>
<script src="../../kotlin-js-min/test/kotlin-test.js"></script>
<script src="../../kotlin-js-min/test/lib.js"></script>
<script src="../../kotlin-js-min/test/app_test.js"></script>
</body>
</html>
```

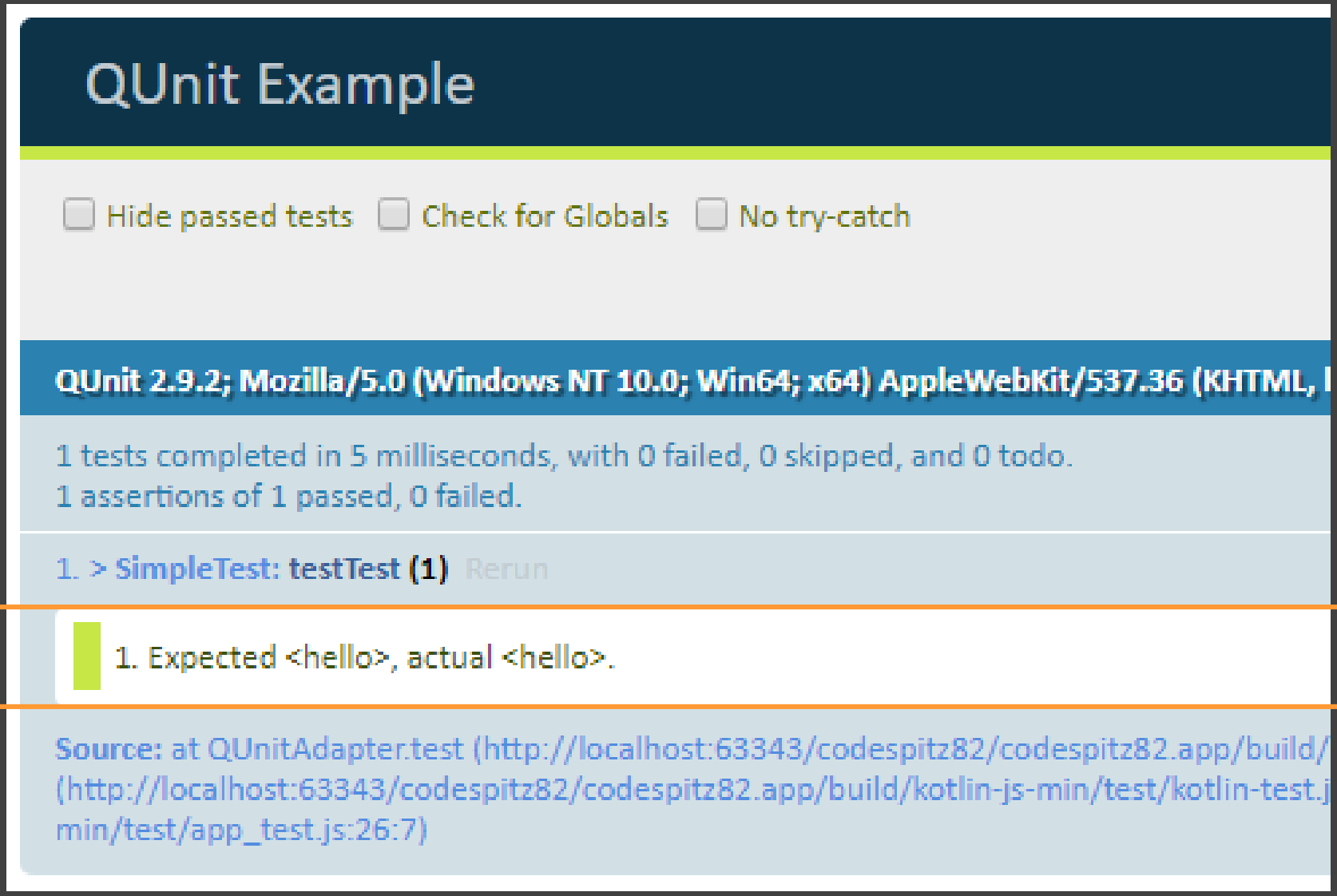
test.html


QUnit Example
Hide passed tests
Check for Globals
No try-catch
Module: All modules
QUnit 2.9.2; Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.3683.86 Safari/53
1 tests completed in 3 milliseconds, with 1 failed, 0 skipped, and 0 todo.
0 assertions of 2 passed, 2 failed.
1. > SimpleTest: testTest (2, 0, 2) Rerun
1. Expected <hello>, actual <abc>.
Expected: true
Result: false
Source: at http://localhost:63343/codespitz82/codespitz82.app/build/kotlin-js-min/test/kotlin-test.js:308:2
at DefaultJsAsserter.invokeHook_0 (http://localhost:63343/codespitz82/codespitz82.app/build/kotlin-
at DefaultJsAsserter.failWithMessage_0 (http://localhost:63343/codespitz82/codespitz82.app/build/ko
at DefaultJsAsserter.assertTrue_o10pc4$ (http://localhost:63343/codespitz82/codespitz82.app/build/k
at DefaultJsAsserter.Asserter.assertEquals_lzc6tz$ (http://localhost:63343/codespitz82/codespitz82.
at DefaultJsAsserter.assertEquals_lzc6tz$ (http://localhost:63343/codespitz82/codespitz82.app/build
2. Died on test #2 at QUnitAdapter.test (http://localhost:63343/codespitz82/codespitz82.app/build/kotlin-js-min/test/kotlin-tes

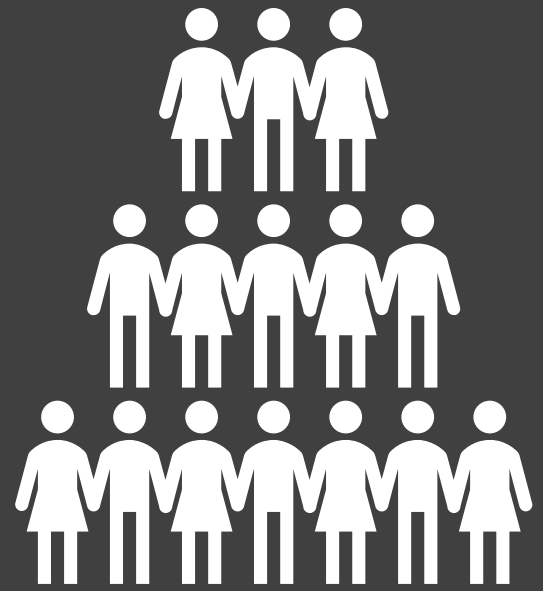
lib/hello.kt

```kotlin
fun hello(){
    println("hello world")
}
fun test() : String = "hello"
```

test.html


QUnit Example
Hide passed tests
Check for Globals
No try-catch
QUnit 2.9.2; Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
1 tests completed in 5 milliseconds, with 0 failed, 0 skipped, and 0 todo.
1 assertions of 1 passed, 0 failed.
1. > SimpleTest: testTest (1) Rerun
1. Expected <hello>, actual <hello>.
Source: at QUnitAdapter.test (http://localhost:63343/codespitz82/codespitz82.app/build/
(http://localhost:63343/codespitz82/codespitz82.app/build/kotlin-js-min/test/kotlin-test.j
min/test/app_test.js:26:7)

Kotlin Types



Base Data Type

Number

Float : 4
Double : 8
Byte : 1
Short : 2
Int : 4
Long : 8

Base Data Type

Number

Float : 4

Double : 8

Byte : 1

Short : 2

Int : 4

Long : 8

UByte

UShort

UInt

ULong

Base Data Type

Number

Float : 4
Double : 8
Byte : 1
Short : 2
Int : 4
Long : 8

UByte
UShort
UInt
ULong

Boolean : true, false
Char : 4
CharSequence
String

Base Data Type

Number

Float : 4
Double : 8
Byte : 1
Short : 2
Int : 4
Long : 8

UByte
UShort
UInt
ULong

Boolean : true, false
Char : 4
CharSequence
String

toByte() : Byte
toShort() : Short
toInt() : Int
toLong() : Long
toFloat() : Float
toDouble() : Double
toChar() : Char
toString() : String

Array

Array
FloatArray
DoubleArray
ByteArray
ShortArray
IntArray
LongArray

BooleanArray
CharArray

Array

Array
FloatArray
DoubleArray
ByteArray
ShortArray
IntArray
LongArray

UByteArray
UIntArray
ULongArray
UShortArray

BooleanArray
CharArray

Array

Array
FloatArray
DoubleArray
ByteArray
ShortArray
IntArray
LongArray

UByteArray
UIntArray
ULongArray
UShortArray

BooleanArray
CharArray

size
indices
lastIndex

all()
any()
forEach()
fold()
indexOf()
joinTo()
map()
reduce()

Base Type

enum – 컴파일 타입에 확정되는 싱글톤 인스턴스

Base Type

enum – 컴파일 타입에 확정되는 싱글톤 인스턴스
Any – 최상위 클래스

Base Type

enum – 컴파일 타입에 확정되는 싱글톤 인스턴스
Any – 최상위 클래스
Nothing – 예외의 반환값

Base Type

enum – 컴파일 타입에 확정되는 싱글톤 인스턴스
Any – 최상위 클래스
Nothing – 예외의 반환값
Result – 예외와 실패를 포함할 수 있는 타입

Base Type

enum – 컴파일 타입에 확정되는 싱글톤 인스턴스
Any – 최상위 클래스
Nothing – 예외의 반환값
Result – 예외와 실패를 포함할 수 있는 타입
Unit – void처럼 반환 값이 없는 경우를 나타내는 값

Base Type

enum – 컴파일 타입에 확정되는 싱글톤 인스턴스
Any – 최상위 클래스
Nothing – 예외의 반환값
Result – 예외와 실패를 포함할 수 있는 타입
Unit – void처럼 반환 값이 없는 경우를 나타내는 값

Comparable – 비교와 순서를 결정할 수 있는 인터페이스

Base Type

enum – 컴파일 타입에 확정되는 싱글톤 인스턴스
Any – 최상위 클래스
Nothing – 예외의 반환값
Result – 예외와 실패를 포함할 수 있는 타입
Unit – void처럼 반환 값이 없는 경우를 나타내는 값

Comparable – 비교와 순서를 결정할 수 있는 인터페이스
Comparator – 실질적인 비교를 처리하는 객체

Base Type

enum – 컴파일 타입에 확정되는 싱글톤 인스턴스
Any – 최상위 클래스
Nothing – 예외의 반환값
Result – 예외와 실패를 포함할 수 있는 타입
Unit – void처럼 반환 값이 없는 경우를 나타내는 값

Comparable – 비교와 순서를 결정할 수 있는 인터페이스
Comparator – 실질적인 비교를 처리하는 객체
Function – 모든 코틀린 람다의 부모

Base Type

enum – 컴파일 타입에 확정되는 싱글톤 인스턴스
Any – 최상위 클래스
Nothing – 예외의 반환값
Result – 예외와 실패를 포함할 수 있는 타입
Unit – void처럼 반환 값이 없는 경우를 나타내는 값

Comparable – 비교와 순서를 결정할 수 있는 인터페이스
Comparator – 실질적인 비교를 처리하는 객체
Function – 모든 코틀린 람다의 부모
Pair – 두 개의 값을 갖는 객체

Base Type

enum – 컴파일 타입에 확정되는 싱글톤 인스턴스
Any – 최상위 클래스
Nothing – 예외의 반환값
Result – 예외와 실패를 포함할 수 있는 타입
Unit – void처럼 반환 값이 없는 경우를 나타내는 값

Comparable – 비교와 순서를 결정할 수 있는 인터페이스
Comparator – 실질적인 비교를 처리하는 객체
Function – 모든 코틀린 람다의 부모
Pair – 두 개의 값을 갖는 객체
Triple – 세 개의 값을 갖는 객체

System Type

Annotation – 애노테이션 객체
Lazy – 지연객체 인터페이스
LazyThreadSafetyMode – 지연모드 값 객체

System Type

Annotation – 애노테이션 객체
Lazy – 지연객체 인터페이스
LazyThreadSafetyMode – 지연모드 값 객체

DeprecationLevel – API등의 파기수준을 나타내는 값 객체
KotlinVersion – 코틀린버전 값 객체

Throwable

Throwable

Error

AssertionError

NotImplementedError

Throwable

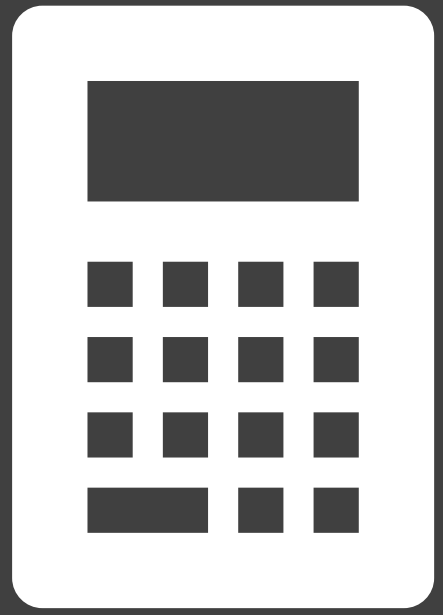
Error

AssertionError
NotImplementedError

Exception

ArithmeticException
ClassCastException
ConcurrentModificationException
IllegalArgumentException
IllegalStateException
IndexOutOfBoundsException
NoSuchElementException
NoWhenBranchMatchedException
NullPointerException
NumberFormatException
RuntimeException
UninitializedPropertyAccessException
UnsupportedOperationException

Calculator



calculator logic

-2 * -3 + 0.4 / -0.2

calculator logic

-2 * -3 + 0.4 / -0.2

`-2*-3+0.4/-0.2` trim

calculator logic

-2 * -3 + 0.4 / -0.2

```
-2*-3+0.4/-0.2          trim
+-2*+-3+0.4/+-0.2       Replace - to +-
```

calculator logic

-2 * -3 + 0.4 / -0.2

```
-2*-3+0.4/-0.2              trim
+-2*+-3+0.4/+-0.2           Replace - to +-
(+-2*+-3), (0.4/+-0.2)      */ group
```

calculator logic

-2 * -3 + 0.4 / -0.2

```
-2*-3+0.4/-0.2                  trim
+-2*+-3+0.4/+-0.2               Replace - to +-
(+-2*+-3), (0.4/+-0.2)          */ group
+-2, *, +-3                     split
```

calculator logic

`-2 * -3 + 0.4 / -0.2`

`-2*-3+0.4/-0.2`	trim
`+-2*+-3+0.4/+-0.2`	Replace - to +-
`(+-2*+-3), (0.4/+-0.2)`	*/ group
`+-2, *, +-3`	split
`+-2 → -2, +-3 → -3`	remove +

calculator logic

-2 * -3 + 0.4 / -0.2

-2*-3+0.4/-0.2	trim
+-2*+-3+0.4/+-0.2	Replace - to +-
(+-2*+-3), (0.4/+-0.2)	*/ group
+-2, *, +-3	split
+-2 → -2, +-3 → -3	remove +
-2 * -3 = 6	calc & replace+-

calculator logic

-2 * -3 + 0.4 / -0.2

-2*-3+0.4/-0.2	trim
+-2*+-3+0.4/+-0.2	Replace - to +-
(+-2*+-3), (0.4/+-0.2)	*/ group
+-2, *, +-3	split
+-2 → -2, +-3 → -3	remove +
-2 * -3 = 6	calc & replace+-

calculator logic

-2 * -3 + 0.4 / -0.2

-2*-3+0.4/-0.2	trim
+-2*+-3+0.4/+-0.2	Replace - to +-
(+-2*+-3), (0.4/+-0.2)	*/ group
+-2, *, +-3	split
+-2 → -2, +-3 → -3	remove +
-2 * -3 = 6	calc & replace+-
0.4, /, +-0.2	split

calculator logic

`-2 * -3 + 0.4 / -0.2`

`-2*-3+0.4/-0.2`	trim
`+-2*+-3+0.4/+-0.2`	Replace - to +-
`(+-2*+-3), (0.4/+-0.2)`	*/ group
`+-2, *, +-3`	split
`+-2 → -2, +-3 → -3`	remove +
`-2 * -3 = 6`	calc & replace+-
`0.4, /, +-0.2`	split
`0.4 → 0.4, +-0.2 → -0.2`	remove +

calculator logic

`-2 * -3 + 0.4 / -0.2`

`-2*-3+0.4/-0.2`	trim
`+-2*+-3+0.4/+-0.2`	Replace - to +-
`(+-2*+-3), (0.4/+-0.2)`	*/ group
`+-2, *, +-3`	split
`+-2 → -2, +-3 → -3`	remove +
`-2 * -3 = 6`	calc & replace+-
`0.4, /, +-0.2`	split
`0.4 → 0.4, +-0.2 → -0.2`	remove +
`0.4 / -0.2 = +-2`	calc & replace+-

calculator logic

-2 * -3 + 0.4 / -0.2

-2*-3+0.4/-0.2	trim
+-2*+-3+0.4/+-0.2	Replace - to +-
(+-2*+-3), (0.4/+-0.2)	*/ group
+-2, *, +-3	split
+-2 → -2, +-3 → -3	remove +
-2 * -3 = 6	calc & replace+-
0.4, /, +-0.2	split
0.4 → 0.4, +-0.2 → -0.2	remove +
0.4 / -0.2 = +-2	calc & replace+-

6++-2

calculator logic

-2 * -3 + 0.4 / -0.2

```
-2*-3+0.4/-0.2                  trim                    6, "", -2 split +
+-2*+-3+0.4/+-0.2               Replace - to +-
(+-2*+-3), (0.4/+-0.2)          */ group
+-2, *, +-3                    split
+-2 → -2, +-3 → -3              remove +
-2 * -3 = 6                     calc & replace+-
0.4, /, +-0.2                   split
0.4 → 0.4, +-0.2 → -0.2         remove +
0.4 / -0.2 = +-2                calc & replace+-
```

6++-2

calculator logic

-2 * -3 + 0.4 / -0.2

```
-2*-3+0.4/-0.2                  trim                    6, "", -2 split +
+-2*+-3+0.4/+-0.2               Replace - to +-         4         sum elements
(+-2*+-3), (0.4/+-0.2)          */ group
+-2, *, +-3                     split
+-2 → -2, +-3 → -3              remove +
-2 * -3 = 6                     calc & replace+-
0.4, /, +-0.2                   split
0.4 → 0.4, +-0.2 → -0.2         remove +
0.4 / -0.2 = +-2                calc & replace+-
```

6++-2

calculator logic

-2 * -3 + 0.4 / -0.2

```
-2*-3+0.4/-0.2
+-2*+-3+0.4/+-0.2
(+-2*+-3), (0.4/+-0.2)
+-2, *, +-3
+-2 → -2, +-3 → -3
-2 * -3 = 6
0.4, /, +-0.2
0.4 → 0.4, +-0.2 → -0.2
0.4 / -0.2 = +-2
```

```
trim                    6, "", -2 split +
Replace - to +-         4         sum elements
*/ group
split
remove +
calc & replace+-
split
remove +
calc & replace+-
```

4

6++-2

calculator logic

-2 * -3 + 0.4 / -0.2

```
-2*-3+0.4/-0.2
+-2*+-3+0.4/+-0.2
(+-2*+-3), (0.4/+-0.2)
+-2, *, +-3
+-2 → -2, +-3 → -3
-2 * -3 = 6
0.4, /, +-0.2
0.4 → 0.4, +-0.2 → -0.2
0.4 / -0.2 = +-2
```

```
trim                    6, "", -2 split +
Replace - to +-  4              sum elements
*/ group
split
remove +
calc & replace+-
split
remove +
calc & replace+-
```

4

6++-2

calculator logic

-2 * -3 + 0.4 / -0.2

```
-2*-3+0.4/-0.2
+-2*+-3+0.4/+-0.2
(+-2*+-3), (0.4/+-0.2)
+-2, *, +-3
+-2 → -2, +-3 → -3
-2 * -3 = 6
0.4, /, +-0.2
0.4 → 0.4, +-0.2 → -0.2
0.4 / -0.2 = +-2
```

```
trim
Replace - to +-
*/ group
split
remove +
calc & replace+-
split
remove +
calc & replace+-
```

```
6, "", -2 split +
4
sum elements
```

4

6++-2

calculator logic

-2 * -3 + 0.4 / -0.2

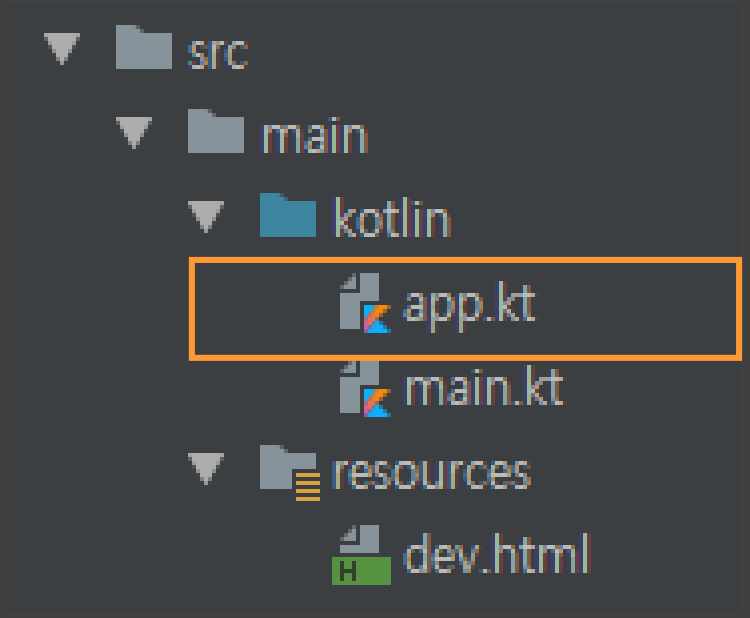
```
-2*-3+0.4/-0.2
+-2*+-3+0.4/+-0.2
(+-2*+-3), (0.4/+-0.2)
+-2, *, +-3
+-2 → -2, +-3 → -3
-2 * -3 = 6
0.4, /, +-0.2
0.4 → 0.4, +-0.2 → -0.2
0.4 / -0.2 = +-2
```

6++-2

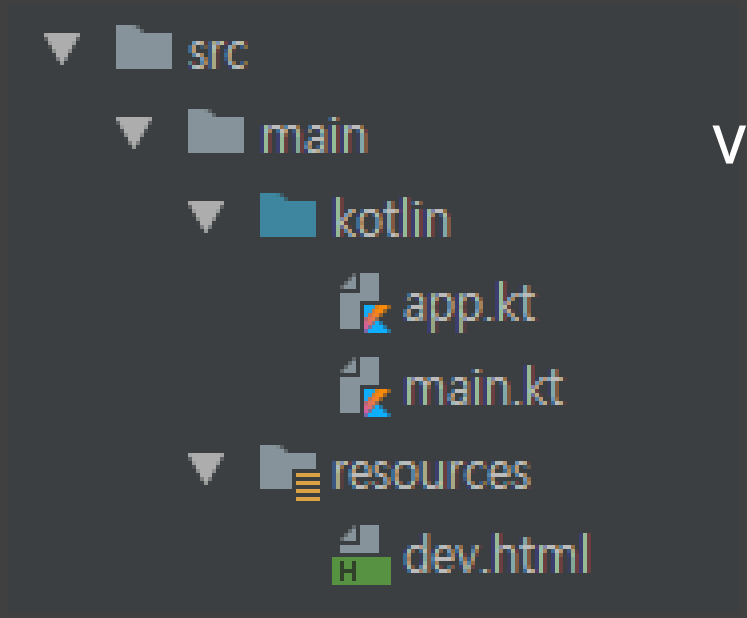
```
trim                    6, "", -2 split +
Replace - to +-         4         sum elements
*/ group
split
remove +
calc & replace+-
split
remove +
calc & replace+-
```

4

app.kt

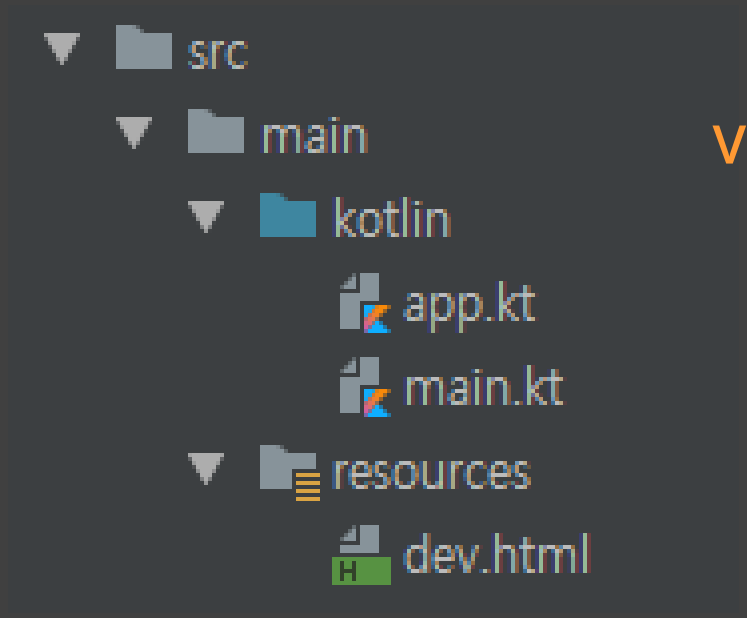


trim



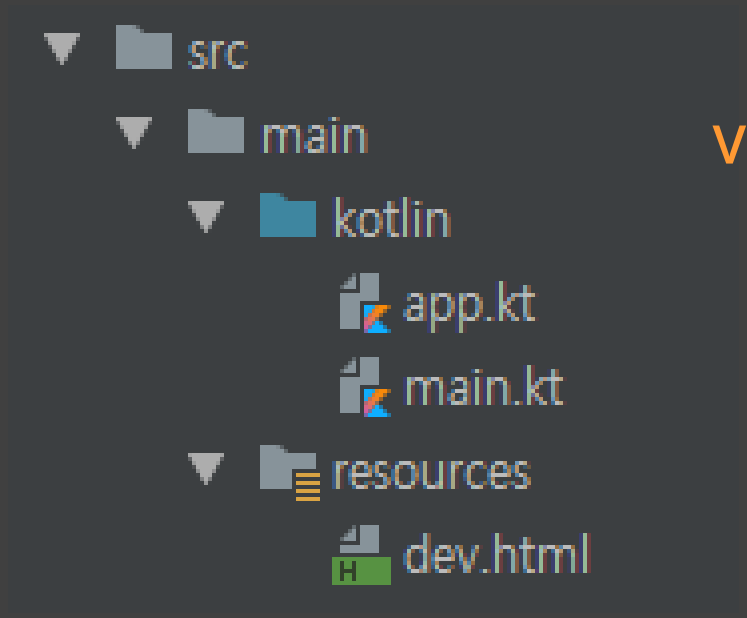

```
val cleanUp:Regex = """[^.\d-+*\/]""".toRegex()
```

trim




```
val cleanUp:Regex = """[^.\d-+*\/]""".toRegex()
```

trim

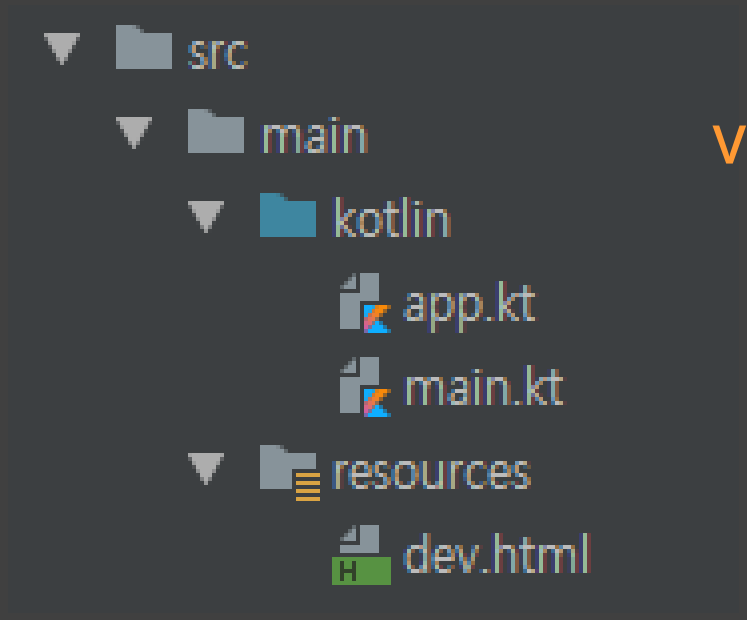



```
val cleanUp:Regex = """[^.\d-+*\/]""".toRegex()
```

val : 상수(value)

var : 변수(variable)

trim



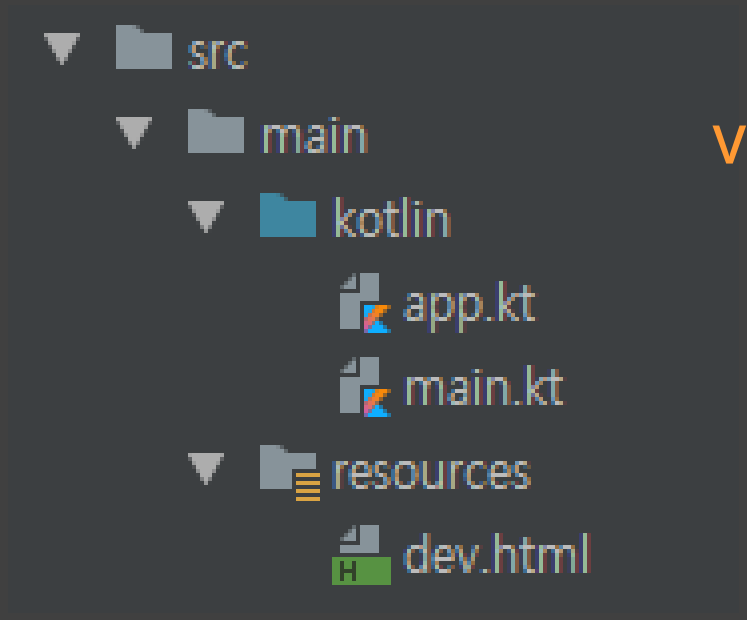

```
val cleanUp:Regex = """[^.\d-+*\/]""".toRegex()
```

val : 상수(value)
var : 변수(variable)

한 줄에 하나만

```
val a, b, c (x)
val a       (o)
```

trim




```
val cleanUp:Regex = """[^.\d-+*\/]""".toRegex()
```

val : 상수(value)
var : 변수(variable)

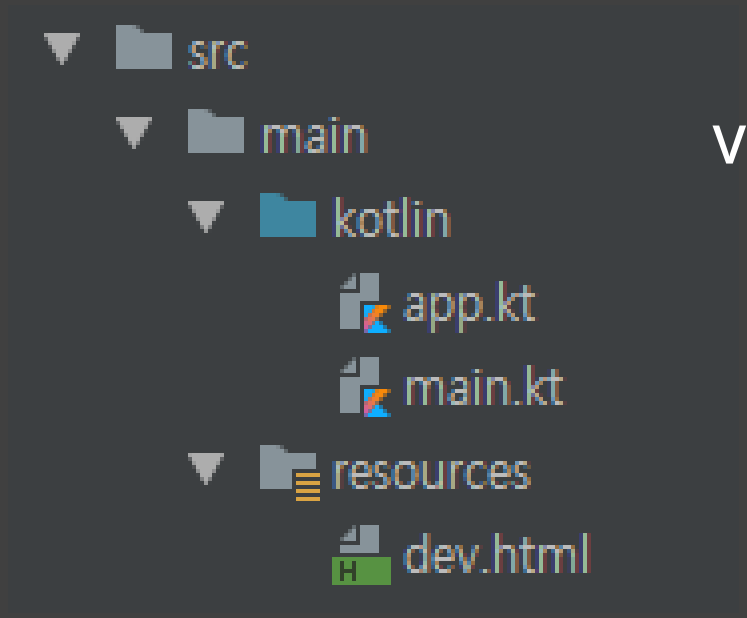
한 줄에 하나만

```
val a, b, c (x)
val a       (o)
```

변수명 : 타입 = 값

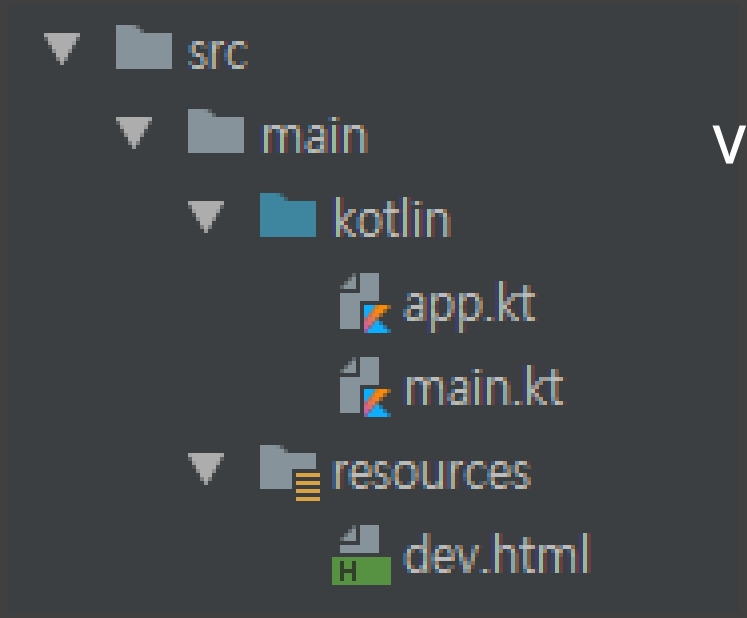
```
val a:Int = 3
var b:String = "abc"
```

trim




```
val cleanUp:Regex = """[^.\d-+*\/]""".toRegex()
```

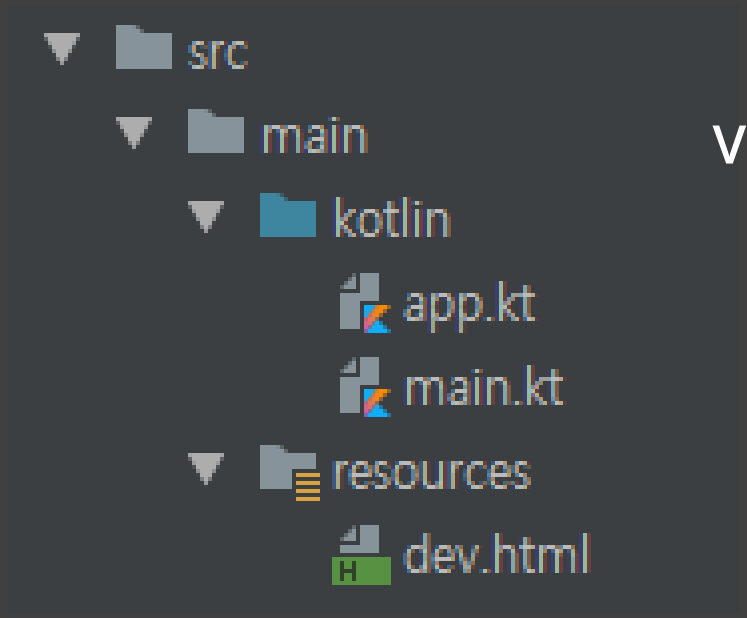
trim




```
val cleanUp:Regex = """[^.\d-+*\/]""".toRegex()
```

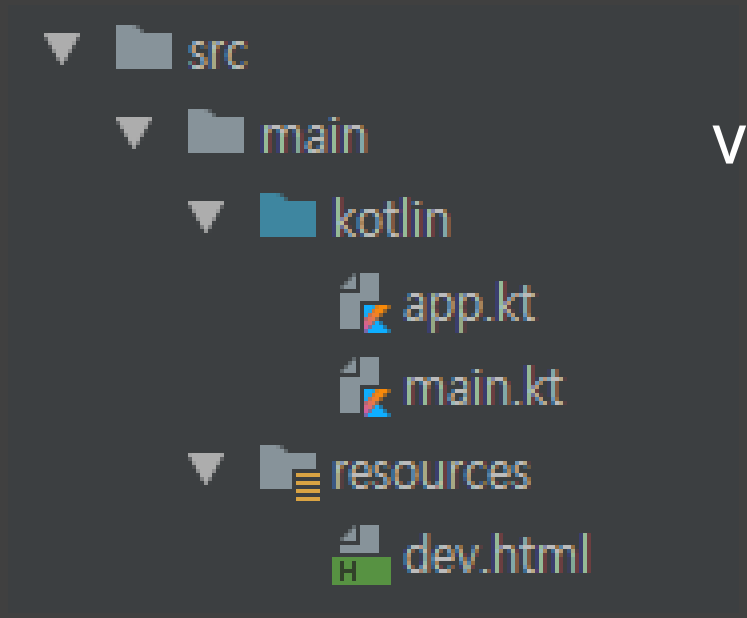
```
'...'   : Char literal
"..."   : String literal
"""..."""   : no escaping, newlines
```

trim




```
val cleanUp:Regex = """[^.\d-+*\/]""".toRegex()
```

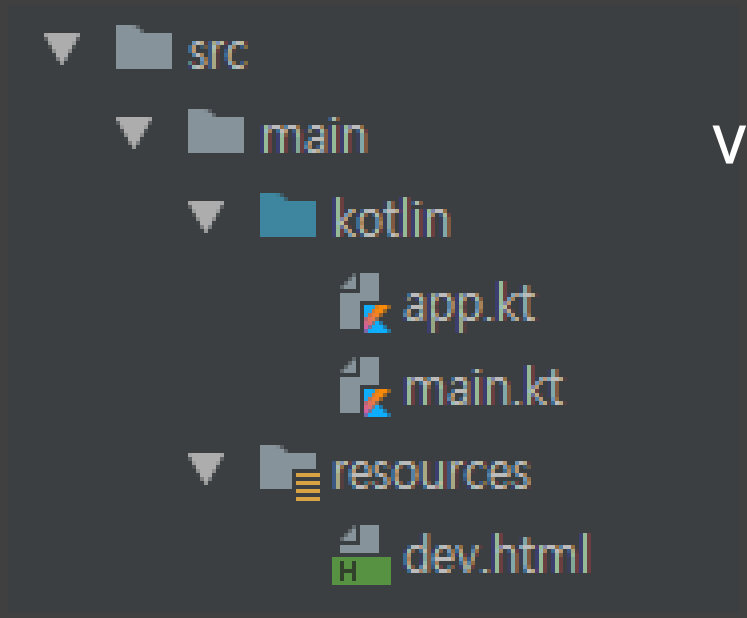
trim




```
val cleanUp:Regex = """[^.\d-+*\/]""".toRegex()
```

`[...]` : Character Group

trim

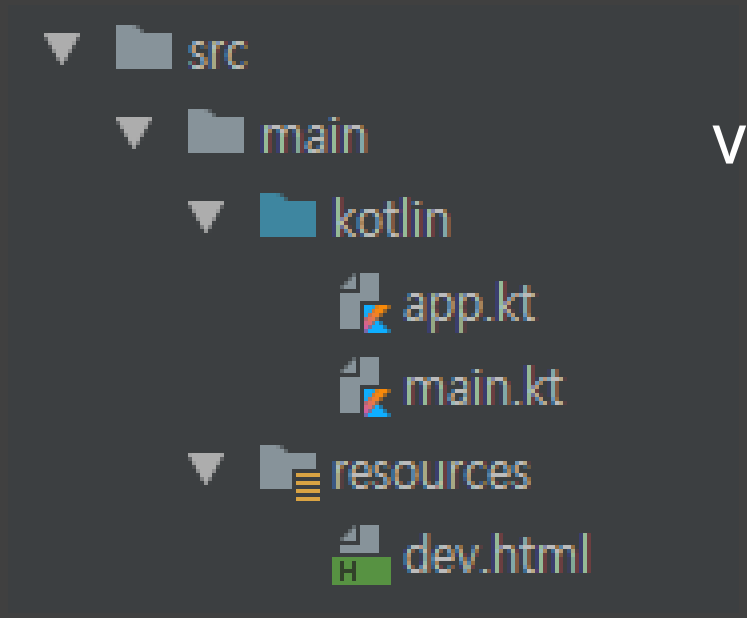



```
val cleanUp:Regex = """[^.\d-+*\/]""".toRegex()
```

`[...]` : Character Group

`[^...]` : Exception Character Group

trim



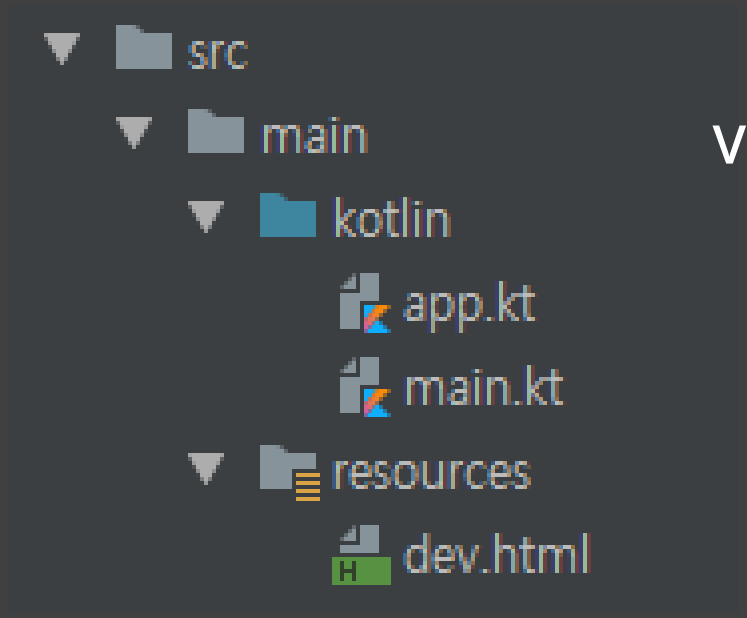

```
val cleanUp:Regex = """[^.\d-+*\/]""".toRegex()
```

`[...]` : Character Group

`[^...]` : Exception Character Group

`\d` : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

trim




```
val cleanUp:Regex = """[^.\d-+*\/]""".toRegex()
```

`[...]` : Character Group

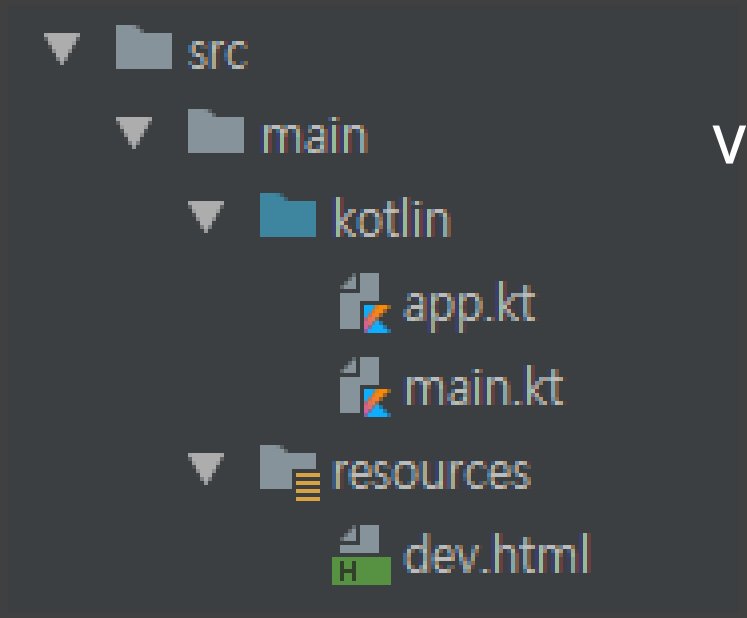
`[^...]` : Exception Character Group

`\d` : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

except

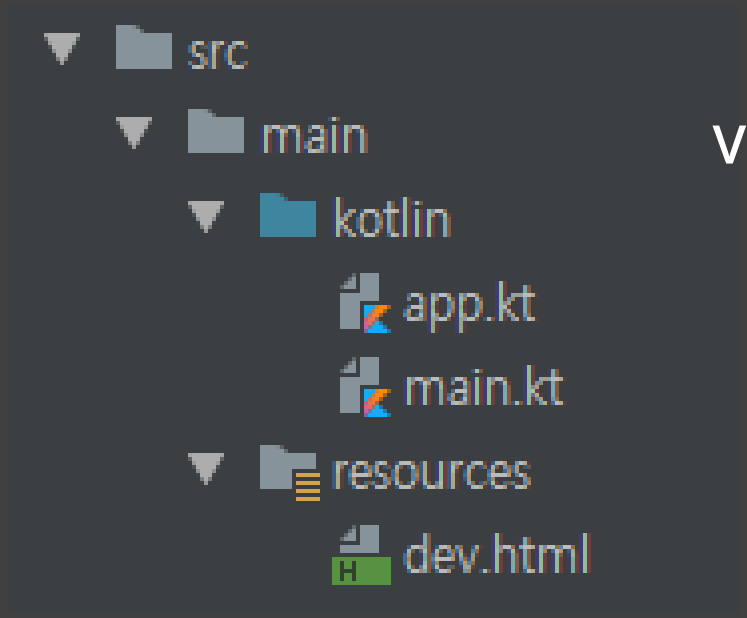
. 0 1 2 3 4 5 6 7 8 9 - + * /

trim




```
val cleanUp:Regex = """[^.\d-+*\/]""".toRegex()
```

trim




```
val cleanUp = """[^.\d-+*\/]""".toRegex()
```

*/ Group

```
val cleanup = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
```

((?:\+-)?[.\d]+)

*/ Group

```
val cleanup = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
```

```
((?:\+-)?[.\d]+)
```

(...) : Capture Group

*/ Group

```
val cleanup = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
```

```
((?:\+-)?[.\d]+)
```

`(...)` : Capture Group

*/ Group

```
val cleanup = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
```

```
((?:\+-)?[.\d]+)
```

(...) : Capture Group

(?:...) : Non capture Group

*/ Group

```
val cleanup = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
```

```
((?:\+-)?[.\d]+)
```

(...) : Capture Group

(?:...) : Non capture Group

? : EA 0 or 1

*/ Group

```
val cleanup = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
```

((?:\+-)?[.\d]+)

(...) : Capture Group

(?:...) : Non capture Group

? : EA 0 or 1

+ : EA 1~

*/ Group

```
val cleanup = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
```

((?:\+-)?[.\d]+)

(...) : Capture Group

(?:...) : Non capture Group

? : EA 0 or 1

+ : EA 1~

*/ Group

```
val cleanup = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
```

((?:\+-)?[.\d]+)

(...) : Capture Group

(?:...) : Non capture Group

? : EA 0 or 1

+ : EA 1~

group1
(+-가 올 수도 있고
점과 숫자가 한 개 이상있어야 함)

*/ Group

```
val cleanup = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
```

((?:\+-)?[.\d]+)

(...) : Capture Group

(?:...) : Non capture Group

? : EA 0 or 1

+ : EA 1~

group1
(+-가 올 수도 있고
점과 숫자가 한 개 이상있어야 함)

10, 0.2, +-5, +-0.4

*/ Group

```
val cleanup = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
```

((?:\+-)?[.\d]+)

(...) : Capture Group

(?:...) : Non capture Group

? : EA 0 or 1

+ : EA 1~

group1
(+-가 올 수도 있고
점과 숫자가 한 개 이상있어야 함)
group2
(* 또는 /)
group3
(+-가 올 수도 있고
점과 숫자가 한 개 이상있어야 함)

*/ Group

```
val cleanup = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
```

((?:\+-)?[.\d]+)

(...) : Capture Group

(?:...) : Non capture Group

? : EA 0 or 1

+ : EA 1~

group1
(+-가 올 수도 있고
점과 숫자가 한 개 이상있어야 함)
group2
(* 또는 /)
group3
(+-가 올 수도 있고
점과 숫자가 한 개 이상있어야 함)

5/4, 2*6, +-5*+-0.4

trim

```
val cleanup = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
fun ex(v:String):Double{
  val r0 = v.replace(cleanUp, "")
}
```

-2*-3+0.4/-0.2	trim
+-2*+-3+0.4/+-0.2	replace +-
(+-2*+-3), (0.4/+-0.2)	*/ group
+-2, *, +-3	split
+-2 → -2, +-3 → -3	remove +
-2 * -3 = 6	calc & replace+-
0.4, /, +-0.2	split
0.4 → 0.4, +-0.2 → -0.2	remove +
0.4 / -0.2 = +-2	calc & replace+-
6, "", -2	split +
4	sum elements

replace – to +-

```
val cleanup = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
fun ex(v:String):Double{
  val r0 = v.replace(cleanUp, "").replace("-", "+-")
}
```

```
-2*-3+0.4/-0.2          trim
+-2*+-3+0.4/+-0.2       replace +-
(+-2*+-3), (0.4/+-0.2)  */ group
+-2, *, +-3             split
+-2 → -2, +-3 → -3      remove +
-2 * -3 = 6             calc & replace+-
0.4, /, +-0.2           split
0.4 → 0.4, +-0.2 → -0.2 remove +
0.4 / -0.2 = +-2        calc & replace+-
6, "", -2               split +
4                       sum elements
```

*/ group

```
val cleanup = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
fun ex(v:String):Double{
  val r0 = v.replace(cleanUp, "").replace("-", "+-").replace(mulDiv){

  }
}
```

```
-2*-3+0.4/-0.2          trim
+-2*+-3+0.4/+-0.2       replace +-
(+-2*+-3), (0.4/+-0.2)  */ group
+-2, *, +-3             split
+-2 → -2, +-3 → -3      remove +
-2 * -3 = 6             calc & replace+-
0.4, /, +-0.2           split
0.4 → 0.4, +-0.2 → -0.2 remove +
0.4 / -0.2 = +-2        calc & replace+-
6, "", -2               split +
4                       sum elements
```

*/ group

```
val cleanup = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
fun ex(v:String):Double{
  val r0 = v.replace(cleanUp, "").replace("-", "+-").replace(mulDiv){

  }       {
}           a1:Type, a2:Type -> Type
            body
           }
```

*/ group

```
val cleanup = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
fun ex(v:String):Double{
  val r0 = v.replace(cleanUp, "").replace("-", "+-").replace(mulDiv){

  }
}
```

```
{
  a1:Type, a2:Type -> Type
  body
}
```

```
val sum = {a:Int, b:Int -> Int
   a + b
}
println( sum(2, 3) ) //5
```

*/ group

```
val cleanup = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
fun ex(v:String):Double{
  val r0 = v.replace(cleanUp, "").replace("-", "+-").replace(mulDiv){

  }       {
}           a1:Type, a2:Type -> Type
            body
            returnValue
          }
```

*/ group

```
val cleanup = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
fun ex(v:String):Double{
  val r0 = v.replace(cleanUp, "").replace("-", "+-").replace(mulDiv){

  }
}
```

```
        {
            a1:Type, a2:Type -> Type
            body
            returnValue
        }
```

```
val sum = {a:Int, b:Int -> a + b}
println( sum(2, 3) ) //5
```

*/ group

```
val cleanup = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
fun ex(v:String):Double{
  val r0 = v.replace(cleanUp, "").replace("-", "+-").replace(mulDiv){
  }       {   (it)
}             returnValue
            }
```

*/ group

```
val cleanup = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
fun ex(v:String):Double{
  val r0 = v.replace(cleanUp, "").replace("-", "+-").replace(mulDiv){

  }
}
```

```
{   (it)
    returnValue
}
```

```
val double = {it * 2}
println( double(2) ) //4
```

*/ group

```
val cleanup = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
fun ex(v:String):Double{
  val r0 = v.replace(cleanUp, "").replace("-", "+-").replace(mulDiv){
  }
}
```

```
{   (it)
    returnValue
}
```

```
val double:(Int)->Int = {it * 2}
println( double(2) ) //4
```

*/ group

```
val cleanup = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
fun ex(v:String):Double{
  val r0 = v.replace(cleanUp, "").replace("-", "+-").replace(mulDiv){
  }
}
```

```
fun map(a:Int, b:(Int)->Int)):Int{
  return b(a)
}
```

*/ group

```
val cleanup = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
fun ex(v:String):Double{
  val r0 = v.replace(cleanUp, "").replace("-", "+-").replace(mulDiv){

  }
}
```

```
fun map(a:Int, b:(Int)->Int)):Int{
  return b(a)
}
```

```
val double:(Int)->Int = {it * 2}
map(5, double)
```

*/ group

```
val cleanup = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
fun ex(v:String):Double{
  val r0 = v.replace(cleanUp, "").replace("-", "+-").replace(mulDiv){

  }
}
```

```
fun map(a:Int, b:(Int)->Int)):Int{
  return b(a)
}
```

```
val double:(Int)->Int = {it * 2}
map(5, double)
map(5, {it * 2})
```

*/ group

```
val cleanup = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
fun ex(v:String):Double{
  val r0 = v.replace(cleanUp, "").replace("-", "+-").replace(mulDiv){

  }
}
```

```
fun map(a:Int, b:(Int)->Int)):Int{
  return b(a)
}
```

```
val double:(Int)->Int = {it * 2}
map(5, double)
map(5, {it * 2})
map(5){it * 2}
```

*/ group

```
val cleanup = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
fun ex(v:String):Double{
  val r0 = v.replace(cleanUp, "").replace("-", "+-").replace(mulDiv){

  }
}
```

```
fun map(a:Int, b:(Int)->Int)):Int{
  return b(a)
}
```

```
val double:(Int)->Int = {it * 2}
map(5, double)
map(5, {it * 2})
map(5){it * 2}
```

*/ group

```
val cleanup = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
fun ex(v:String):Double{
  val r0 = v.replace(cleanUp, "").replace("-", "+-").replace(mulDiv){

  }
}
```

```
fun CharSequence.replace(
    regex: Regex,
    transform: (MatchResult) -> CharSequence
): String
```

split */ group

```
val cleanup = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
fun ex(v:String):Double{
  val r0 = v.replace(cleanUp, "").replace("-", "+-").replace(mulDiv){
    val (_, left, op, right) = it.groupValues
  }
}
```

```
-2*-3+0.4/-0.2          trim
+-2*+-3+0.4/+-0.2       replace +-
(+-2*+-3), (0.4/+-0.2)  */ group
+-2, *, +-3             split
+-2 → -2, +-3 → -3      remove +
-2 * -3 = 6             calc & replace+-
0.4, /, +-0.2           split
0.4 → 0.4, +-0.2 → -0.2 remove +
0.4 / -0.2 = +-2        calc & replace+-
6, "", -2               split +
4                       sum elements
```

split */ group

```
val cleanup = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
fun ex(v:String):Double{
  val r0 = v.replace(cleanUp, "").replace("-", "+-").replace(mulDiv){
    val (_, left, op, right) = it.groupValues
  }
}
```

```
MatchResult.groupValues
["+-2*+-3", "+-2", "*", "+-3"]
```

```
-2*-3+0.4/-0.2              trim
+-2*+-3+0.4/+-0.2           replace +-
(+-2*+-3), (0.4/+-0.2)      */ group
+-2, *, +-3                 split
+-2 → -2, +-3 → -3          remove +
-2 * -3 = 6                 calc & replace+-
0.4, /, +-0.2               split
0.4 → 0.4, +-0.2 → -0.2     remove +
0.4 / -0.2 = +-2            calc & replace+-
6, "", -2                   split +
4                           sum elements
```

split */ group

```
val cleanup = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
fun ex(v:String):Double{
  val r0 = v.replace(cleanUp, "").replace("-", "+-").replace(mulDiv){
    val (_, left, op, right) = it.groupValues
  }
}
```

MatchResult.groupValues

["+-2*+-3", "+-2", "*", "+-3"]

```
-2*-3+0.4/-0.2           trim
+-2*+-3+0.4/+-0.2        replace +-
(+-2*+-3), (0.4/+-0.2)   */ group
+-2, *, +-3              split
+-2 → -2, +-3 → -3       remove +
-2 * -3 = 6              calc & replace+-
0.4, /, +-0.2            split
0.4 → 0.4, +-0.2 → -0.2  remove +
0.4 / -0.2 = +-2         calc & replace+-
6, "", -2                split +
4                        sum elements
```

split */ group

```
val cleanup = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
fun ex(v:String):Double{
  val r0 = v.replace(cleanUp, "").replace("-", "+-").replace(mulDiv){
    val (_, left, op, right) = it.groupValues
  }
}
```

MatchResult.groupValues

["+-2*+-3", "+-2", "*", "+-3"]

-2*-3+0.4/-0.2	trim
+-2*+-3+0.4/+-0.2	replace +-
(+-2*+-3), (0.4/+-0.2)	*/ group
+-2, *, +-3	split
+-2 → -2, +-3 → -3	remove +
-2 * -3 = 6	calc & replace+-
0.4, /, +-0.2	split
0.4 → 0.4, +-0.2 → -0.2	remove +
0.4 / -0.2 = +-2	calc & replace+-
6, "", -2	split +
4	sum elements

split */ group

```
val cleanup = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
fun ex(v:String):Double{
  val r0 = v.replace(cleanUp, "").replace("-", "+-").replace(mulDiv){
    val (_, left, op, right) = it.groupValues
  }
}
```

MatchResult.groupValues

["+-2*+-3", "+-2", "*", "+-3"]

destructuring

```
-2*-3+0.4/-0.2          trim
+-2*+-3+0.4/+-0.2       replace +-
(+-2*+-3), (0.4/+-0.2)  */ group
+-2, *, +-3             split
+-2 → -2, +-3 → -3      remove +
-2 * -3 = 6             calc & replace+-
0.4, /, +-0.2           split
0.4 → 0.4, +-0.2 → -0.2 remove +
0.4 / -0.2 = +-2        calc & replace+-
6, "", -2               split +
4                       sum elements
```

remove +

```
val cleanup = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
fun ex(v:String):Double{
  val r0 = v.replace(cleanUp, "").replace("-", "+-").replace(mulDiv){
    val (_, left, op, right) = it.groupValues
    val l = left.replace("+", "").toDouble()
    val r = right.replace("+", "").toDouble()
  }
}
```

```
-2*-3+0.4/-0.2          trim
+-2*+-3+0.4/+-0.2       replace +-
(+-2*+-3), (0.4/+-0.2)  */ group
+-2, *, +-3             split
+-2 → -2, +-3 → -3      remove +
-2 * -3 = 6             calc & replace+-
0.4, /, +-0.2           split
0.4 → 0.4, +-0.2 → -0.2 remove +
0.4 / -0.2 = +-2        calc & replace+-
6, "", -2               split +
4                       sum elements
```

calculate & replace

```
val cleanup = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
fun ex(v:String):Double{
  val r0 = v.replace(cleanUp, "").replace("-", "+-").replace(mulDiv){
    val (_, left, op, right) = it.groupValues
    val l = left.replace("+", "").toDouble()
    val r = right.replace("+", "").toDouble()
    "${if(op == "*") l * r else l / r}"
      .replace("-", "+-")
  }
}
```

```
-2*-3+0.4/-0.2          trim
+-2*+-3+0.4/+-0.2       replace +-
(+-2*+-3), (0.4/+-0.2)  */ group
+-2, *, +-3             split
+-2 → -2, +-3 → -3      remove +
-2 * -3 = 6             calc & replace+-
0.4, /, +-0.2           split
0.4 → 0.4, +-0.2 → -0.2 remove +
0.4 / -0.2 = +-2        calc & replace+-
6, "", -2               split +
4                       sum elements
```

calculate & replace

```
val cleanup = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
fun ex(v:String):Double{
  val r0 = v.replace(cleanUp, "").replace("-", "+-").replace(mulDiv){
    val (_, left, op, right) = it.groupValues
    val l = left.replace("+", "").toDouble()
    val r = right.replace("+", "").toDouble()
    "${if(op == "*") l * r else l / r}"
      .replace("-", "+-")
  }
}
```

"..$name..", "..${...}.." : String template

calculate & replace

```
val cleanup = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
fun ex(v:String):Double{
  val r0 = v.replace(cleanUp, "").replace("-", "+-").replace(mulDiv){
    val (_, left, op, right) = it.groupValues
    val l = left.replace("+", "").toDouble()
    val r = right.replace("+", "").toDouble()
    "${if(op == "*") l * r else l / r}"
      .replace("-", "+-")
  }
}
```

```
"..$name..", "..${...}.." : String template

val a = 3
val b = 5
println("$a + $b = ${a + b}") // "3 + 5 = 8"
```

split +

```
val cleanup = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
fun ex(v:String):Double{
  val r0 = v.replace(cleanUp, "").replace("-", "+-").replace(mulDiv){
    val (_, left, op, right) = it.groupValues
    val l = left.replace("+", "").toDouble()
    val r = right.replace("+", "").toDouble()
    "${if(op == "*") l * r else l / r}"
      .replace("-", "+-")
  }.split('+')
}
```

```
-2*-3+0.4/-0.2          trim
+-2*+-3+0.4/+-0.2       replace +-
(+-2*+-3), (0.4/+-0.2)  */ group
+-2, *, +-3             split
+-2 → -2, +-3 → -3      remove +
-2 * -3 = 6             calc & replace+-
0.4, /, +-0.2           split
0.4 → 0.4, +-0.2 → -0.2 remove +
0.4 / -0.2 = +-2        calc & replace+-
6, "", -2               split +
4                       sum elements
```

sum elements

```
val cleanup = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
fun ex(v:String):Double{
  val r0 = v.replace(cleanUp, "").replace("-", "+-").replace(mulDiv){
    val (_, left, op, right) = it.groupValues
    val l = left.replace("+", "").toDouble()
    val r = right.replace("+", "").toDouble()
    "${if(op == "*") l * r else l / r}"
      .replace("-", "+-")
  }.split('+').fold(0.0){sum, v->
    sum + if(v.isBlank()) 0.0 else v.toDouble()
  }
  return r0
}
```

```
-2*-3+0.4/-0.2          trim
+-2*+-3+0.4/+-0.2       replace +-
(+-2*+-3), (0.4/+-0.2)  */ group
+-2, *, +-3             split
+-2 → -2, +-3 → -3      remove +
-2 * -3 = 6             calc & replace+-
0.4, /, +-0.2           split
0.4 → 0.4, +-0.2 → -0.2 remove +
0.4 / -0.2 = +-2        calc & replace+-
6, "", -2               split +
4                       sum elements
```

final result

```
val cleanup = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
fun ex(v:String):Double{
  val r0 = v.replace(cleanUp, "").replace("-", "+-").replace(mulDiv){
    val (_, left, op, right) = it.groupValues
    val l = left.replace("+", "").toDouble()
    val r = right.replace("+", "").toDouble()
    "${if(op == "*") l * r else l / r}"
      .replace("-", "+-")
  }.split('+').fold(0.0){sum, v->
    sum + if(v.isBlank()) 0.0 else v.toDouble()
  }
  return r0
}
```

final result

```
val cleanup = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
fun ex(v:String):Double{
  val r0 = v.replace(cleanUp, "").replace("-", "+-").replace(mulDiv){
    val (_, left, op, right) = it.groupValues
    val l = left.replace("+", "").toDouble()
    val r = right.replace("+", "").toDouble()
    "${if(op == "*") l * r else l / r}"
      .replace("-", "+-")
  }.split('+').fold(0.0){sum, v->
    sum + if(v.isBlank()) 0.0 else v.toDouble()
  }
  return r0
}
```

single expression function

```
val cleanup = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
fun ex(v:String):Double =
  v.replace(cleanUp, "").replace("-", "+-").replace(mulDiv){
    val (_, left, op, right) = it.groupValues
    val l = left.replace("+", "").toDouble()
    val r = right.replace("+", "").toDouble()
    "${if(op == "*") l * r else l / r}"
      .replace("-", "+-")
  }.split('+').fold(0.0){sum, v->
    sum + if(v.isBlank()) 0.0 else v.toDouble()
  }
```

single expression function

```
val cleanup = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
fun ex(v:String):Double =
  v.replace(cleanUp, "").replace("-", "+-").replace(mulDiv){
    val (_, left, op, right) = it.groupValues
    val l = left.replace("+", "").toDouble()
    val r = right.replace("+", "").toDouble()
    "${if(op == "*") l * r else l / r}"
      .replace("-", "+-")
  }.split('+').fold(0.0){sum, v->
    sum + if(v.isBlank()) 0.0 else v.toDouble()
  }
```

single expression function

```
val cleanup = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()

fun ex(v:String) = v.replace(cleanUp, "").replace("-", "+-").replace(mulDiv){
  val (_, left, op, right) = it.groupValues
  val l = left.replace("+", "").toDouble()
  val r = right.replace("+", "").toDouble()
  "${if(op == "*") l * r else l / r}".replace("-", "+-")
}.split('+').fold(0.0){sum, v->
  sum + if(v.isBlank()) 0.0 else v.toDouble()
}
```

app

```
fun main(){
  app()
}

fun app(){

}
```

app

```
fun app(){
  document.querySelector("#base")?.innerHTML = """
    <input id="input"/>
    <div id="result"></div>
  """
}
```

app

```
fun app(){
  document.querySelector("#base")?.innerHTML = """
    <input id="input"/>
    <div id="result"></div>
  """
}
```

```
target?.xxxx
target?.xxxx()
```

safe call

app

```
fun app(){
  document.querySelector("#base")?.innerHTML = """
    <input id="input"/>
    <div id="result"></div>
  """
}
```

```
target?.xxxx
target?.xxxx()
```

safe call

```
if(target != null) target.xxxx else null
if(target != null) target.xxxx() else null
```

app

```
fun app(){
  document.querySelector("#base")?.innerHTML = """
    <input id="input"/>
    <div id="result"></div>
  """
  document.querySelector("#input")?.addEventListener("keyup", {

  })
}
```

app

```
fun app(){
  document.querySelector("#base")?.innerHTML = """
    <input id="input"/>
    <div id="result"></div>
  """
  document.querySelector("#input")?.addEventListener("keyup", {
    if((it as KeyboardEvent).keyCode == 13){

    }
  })
}
```

app

```
fun app(){
  document.querySelector("#base")?.innerHTML = """
    <input id="input"/>
    <div id="result"></div>
  """
  document.querySelector("#input")?.addEventListener("keyup", {
    if((it as KeyboardEvent).keyCode == 13){
      val input = it.target as HTMLInputElement
      val v = input.value

    }
  })
}
```

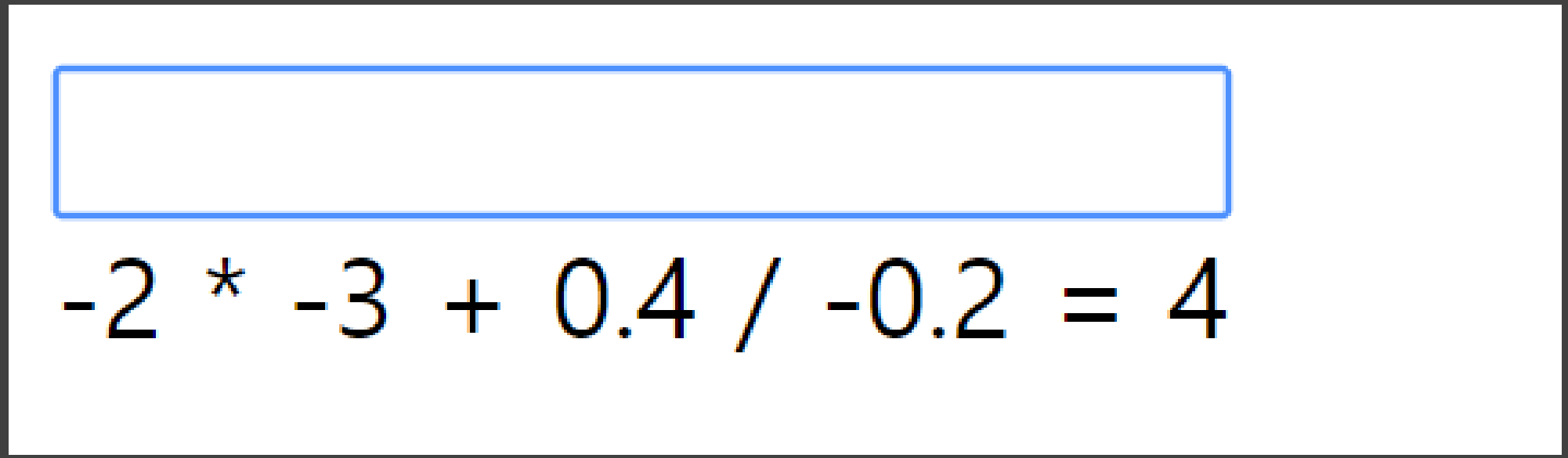
app

```
fun app(){
  document.querySelector("#base")?.innerHTML = """
    <input id="input"/>
    <div id="result"></div>
  """
  document.querySelector("#input")?.addEventListener("keyup", {
    if((it as KeyboardEvent).keyCode == 13){
      val input = it.target as HTMLInputElement
      val v = input.value
      document.querySelector("#result")?.innerHTML = "$v = ${ex(v)}"
    }
  })
}
```

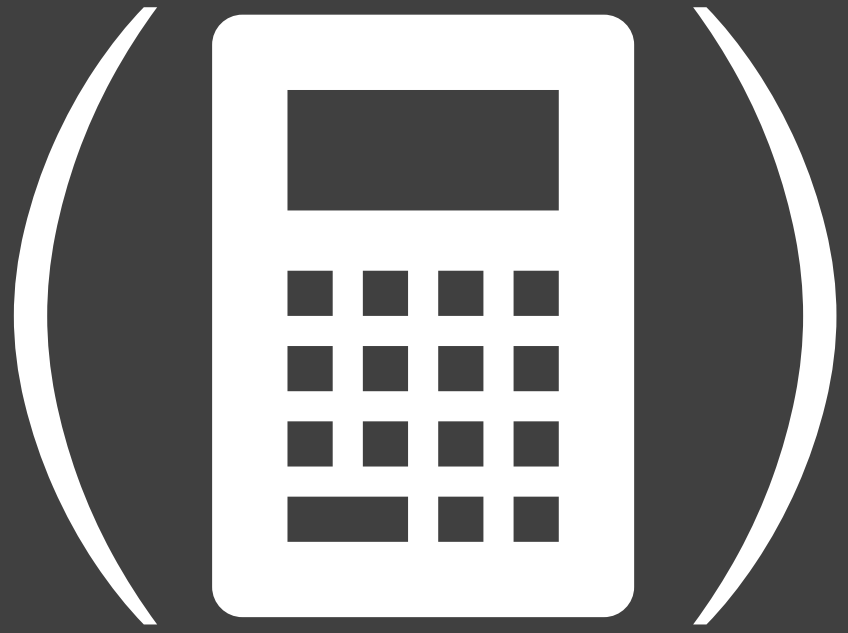
dev.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<div id="base"></div>
<script src="../../kotlin-js-min/main/kotlin.js"></script>
<script src="../../kotlin-js-min/main/lib.js"></script>
<script src="../../kotlin-js-min/main/app.js"></script>
</body>
</html>
```

dev.html



Calculator()



calculator logic

-2 * ((-3 + 0.4) / -0.2)

calculator logic

-2 * ((-3 + 0.4) / -0.2)

```
-3 + 0.4          calc
```

calculator logic

-2 * ((-3 + 0.4) / -0.2)

```
-3 + 0.4          calc
-2.6 / -0.2       calc
```

calculator logic

```
-2 * ((-3 + 0.4) / -0.2)
```

```
-3 + 0.4          calc
-2.6 / -0.2       calc
-2 * 13           calc
```

calculator logic

-2 * ((-3 + 0.4) / -0.2)

```
-3 + 0.4          calc
-2.6 / -0.2       calc
-2 * 13           calc
```

내부에 괄호가 없는 식부터 해소
더 이상 괄호가 없을 때까지 반복

calculator logic

-2 * ((-3 + 0.4) / -0.2)

```
-3 + 0.4          calc
-2.6 / -0.2       calc
-2 * 13           calc
```

내부에 괄호가 없는 식부터 해소

더 이상 괄호가 없을 때까지 반복

```
val paren = """\(([^()]*)\)""".toRegex()
```

calculator logic

`-2 * ((-3 + 0.4) / -0.2)`

```
-3 + 0.4          calc
-2.6 / -0.2       calc
-2 * 13           calc
```

내부에 괄호가 없는 식부터 해소

더 이상 괄호가 없을 때까지 반복

```
val paren = """\(([^()]*)\)""".toRegex()
```

calculator logic

-2 * ((-3 + 0.4) / -0.2)

```
-3 + 0.4          calc
-2.6 / -0.2       calc
-2 * 13           calc
```

내부에 괄호가 없는 식부터 해소

더 이상 괄호가 없을 때까지 반복

```
val paren = """\(([^()]*)\)""".toRegex()
```

* : 0~

calculator logic

-2 * ((-3 + 0.4) / -0.2)

```
-3 + 0.4          calc
-2.6 / -0.2       calc
-2 * 13           calc
```

내부에 괄호가 없는 식부터 해소

더 이상 괄호가 없을 때까지 반복

```
val paren = """\(([^()]*)\)""".toRegex()
```

calculator logic

-2 * ((-3 + 0.4) / -0.2)

내부에 괄호가 없는 식부터 해소

더 이상 괄호가 없을 때까지 반복

```
val paren = """\(([^()]*)\)""".toRegex()
fun calc(v:String):Double{
    var r = v
    while(paren.containsMatchIn(r)) r = r.replace(paren){"${ex(it.groupValues[1])}"}
    return ex(r)
}
```

calculator logic

-2 * ((-3 + 0.4) / -0.2)

내부에 괄호가 없는 식부터 해소
더 이상 괄호가 없을 때까지 반복

```
val paren = """\(([^()]*)\)""".toRegex()
fun calc(v:String):Double{
    var r = v
    while(paren.containsMatchIn(r)) r = r.replace(paren){"${ex(it.groupValues[1])}"}
    return ex(r)
}
```

calculator logic

-2 * ((-3 + 0.4) / -0.2)

내부에 괄호가 없는 식부터 해소
더 이상 괄호가 없을 때까지 반복

```
val paren = """\(([^()]*)\)""".toRegex()
fun calc(v:String):Double{
    var r = v
    while(paren.containsMatchIn(r)) r = r.replace(paren){"${ex(it.groupValues[1])}"}
    return ex(r)
}
```

```
document.querySelector("#result")?.innerHTML = "$v = ${calc(v)}"
```

```
fun app(){
    document.querySelector("#base")?.innerHTML = """<input id="input"/><div id="result"></div>"""
    document.querySelector("#input")?.addEventListener("keyup", {
        if((it as KeyboardEvent).keyCode != 13) return@addEventListener
        val input = it.target as HTMLInputElement
        val v = input.value
        document.querySelector("#result")?.innerHTML = "$v = ${calc(v)}"
        input.value = ""
    })
}
val cleanUp = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:\+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
val paren = """\(([^()]*)\)""".toRegex()
fun ex(v:String) = v.replace(cleanUp, "").replace("-", "+-").replace(mulDiv){
    val (_, left, op, right) = it.groupValues
    val l = left.replace("+", "").toDouble()
    val r = right.replace("+", "").toDouble()
    "${if(op == "*") l * r else l / r}".replace("-", "+-")
}.split('+').fold(0.0) { acc, v -> acc + if(v.isBlank()) 0.0 else v.toDouble()}
fun calc(v:String):Double{
    var r = v
    while(paren.containsMatchIn(r)) r = r.replace(paren){"${ex(it.groupValues[1])}"}
    return ex(r)
}
```

```
fun app(){
    document.querySelector("#base")?.innerHTML = """<input id="input"/><div id="result"></div>"""
    document.querySelector("#input")?.addEventListener("keyup", {
        if((it as KeyboardEvent).keyCode != 13) return@addEventListener
        val input = it.target as HTMLInputElement
        val v = input.value
        document.querySelector("#result")?.innerHTML = "$v = ${calc(v)}"
        input.value = ""
    })
}
val cleanUp = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:\+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
val paren = """\(([^()]*)\)""".toRegex()
fun ex(v:String) = v.replace(cleanUp, "").replace("-", "+-").replace(mulDiv){
    val (_, left, op, right) = it.groupValues
    val l = left.replace("+", "").toDouble()
    val r = right.replace("+", "").toDouble()
    "${if(op == "*") l * r else l / r}".replace("-", "+-")
}.split('+').fold(0.0) { acc, v -> acc + if(v.isBlank()) 0.0 else v.toDouble()}
fun calc(v:String):Double{
    var r = v
    while(paren.containsMatchIn(r)) r = r.replace(paren){"${ex(it.groupValues[1])}"}
    return ex(r)
}
```

```
fun app(){
    document.querySelector("#base")?.innerHTML = """<input id="input"/><div id="result"></div>"""
    document.querySelector("#input")?.addEventListener("keyup", {
        if((it as KeyboardEvent).keyCode != 13) return@addEventListener
        val input = it.target as HTMLInputElement
        val v = input.value
        document.querySelector("#result")?.innerHTML = "$v = ${calc(v)}"
        input.value = ""
    })
}
val cleanUp = """[^.\d+-*\/]""".toRegex()
val mulDiv = """(?:\-?)[.\d]+)[*\/](?:\-?)[.\d]+""".toRegex()
val paren = """\(([^()]*)\)""".toRegex()
fun ex(v:String) = v.replace(cleanUp, "").replace("-", "+-").replace(mulDiv){
    val (_, left, op, right) = it.groupValues
    val l = left.replace("+", "").toDouble()
    val r = right.replace("+", "").toDouble()
    "${if(op == "*") l * r else l / r}".replace("-", "+-")
}.split('+').fold(0.0) { acc, v -> acc + if(v.isBlank()) 0.0 else v.toDouble()}
fun calc(v:String):Double{
    var r = v
    while(paren.containsMatchIn(r)) r = r.replace(paren){"${ex(it.groupValues[1])}"}
    return ex(r)
}
```

람다의 return은 감싸고 있는 함수의 return

```
fun app(){
    document.querySelector("#base")?.innerHTML = """<input id="input"/><div id="result"></div>"""
    document.querySelector("#input")?.addEventListener("keyup", {
        if((it as KeyboardEvent).keyCode != 13) return@addEventListener
        val input = it.target as HTMLInputElement
        val v = input.value
        document.querySelector("#result")?.innerHTML = "$v = ${calc(v)}"
        input.value = ""
    })
}
val c
val m
val p
fun e
    val (_, left, op, right) = it.groupValues
    val l = left.replace("+", "").toDouble()
    val r = right.replace("+", "").toDouble()
    "${if(op == "*") l * r else l / r}".replace("-", "+-")
}.split('+').fold(0.0) { acc, v -> acc + if(v.isBlank()) 0.0 else v.toDouble()}
fun calc(v:String):Double{
    var r = v
    while(paren.containsMatchIn(r)) r = r.replace(paren){"${ex(it.groupValues[1])}"}
    return ex(r)
}
```

람다의 return은 감싸고 있는 함수의 return

인자로 전달된 람다는 감싸고 있는 함수가 없음

```
fun app(){
    document.querySelector("#base")?.innerHTML = """<input id="input"/><div id="result"></div>"""
    document.querySelector("#input")?.addEventListener("keyup", {
        if((it as KeyboardEvent).keyCode != 13) return@addEventListener
        val input = it.target as HTMLInputElement
        val v = input.value
        document.querySelector("#result")?.innerHTML = "$v = ${calc(v)}"
        input.value = ""
    })
}
val cleanable = """[^.\d+-*\/]""".toRegex()
val multiply = """(?:\+)?[.\d]+)[*\/](?:\+)?[.\d]+""".toRegex()
val paren = """\(([^()]*)\)""".toRegex()
fun ex(v:String) = v.replace(cleanable,"").replace(multiply){
    val (left, op, right) = it.groupValues
    val l = left.replace("+", "").toDouble()
    val r = right.replace("+", "").toDouble()
    "${if(op == "*") l * r else l / r}".replace("-", "+-")
}.split('+').fold(0.0) { acc, v -> acc + if(v.isBlank()) 0.0 else v.toDouble()}
fun calc(v:String):Double{
    var r = v
    while(paren.containsMatchIn(r)) r = r.replace(paren){"${ex(it.groupValues[1])}"}
    return ex(r)
}
```

람다의 return은 감싸고 있는 함수의 return

인자로 전달된 람다는 감싸고 있는 함수가 없음

인자람다의 return은 반드시 기명return만 가능

```
fun app(){
    document.querySelector("#base")?.innerHTML = """<input id="input"/><div id="result"></div>"""
    document.querySelector("#input")?.addEventListener("keyup", {
        if((it as KeyboardEvent).keyCode != 13) return@addEventListener
        val input = it.target as HTMLInputElement
        val v = input.value
        document.querySelector("#result")?.innerHTML = "$v = ${calc(v)}"
        input.value = ""
    })
}
val cleanUp = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:\+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
val paren = """\(([^()]*)\)""".toRegex()
fun ex(v:String) = v.replace(cleanUp,"").replace("--","+").replace("+-","-").let{
    val (_, left, op, right) = it.groupValues
    val l = left.replace("+", "").toDouble()
    val r = right.replace("+", "").toDouble()
    "${if(op == "*") l * r else l / r}".replace("-", "+-")
}.split('+').fold(0.0) { acc, v -> acc + if(v.isBlank()) 0.0 else v.toDouble()}
fun calc(v:String):Double{
    var r = v
    while(paren.containsMatchIn(r)) r = r.replace(paren){"${ex(it.groupValues[1])}"}
    return ex(r)
}
```

람다의 return은 감싸고 있는 함수의 return

인자로 전달된 람다는 감싸고 있는 함수가 없음

인자람다의 return은 반드시 기명return만 가능

기명람다 - 람다에 직접 이름을 부여하는 방식

name@{return@name}

```
fun app(){
    document.querySelector("#base")?.innerHTML = """<input id="input"/><div id="result"></div>"""
    document.querySelector("#input")?.addEventListener("keyup", {
        if((it as KeyboardEvent).keyCode != 13) return@addEventListener
        val input = it.target as HTMLInputElement
        val v = input.value
        document.querySelector("#result")?.innerHTML = "$v = ${calc(v)}"
        input.value = ""
    })
}
val cleanUp = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:\+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
val paren = """\(([^()]*)\)""".toRegex()
fun ex(v:String) = v.replace(cleanUp,"").replace("--","+").replace(mulDiv){
    val (_, left, op, right) = it.groupValues
    val l = left.replace("+", "").toDouble()
    val r = right.replace("+", "").toDouble()
    "${if(op == "*") l * r else l / r}"
}.split('+').fold(0.0) { acc, v -> acc + if(v.isBlank()) 0.0 else v.toDouble()}
fun calc(v:String):Double{
    var r = v
    while(paren.containsMatchIn(r)) r = r.replace(paren){"${ex(it.groupValues[1])}"}
    return ex(r)
}
```

람다의 return은 감싸고 있는 함수의 return

인자로 전달된 람다는 감싸고 있는 함수가 없음

인자람다의 return은 반드시 기명return만 가능

기명람다 - 람다에 직접 이름을 부여하는 방식

name@{return@name}

자동이름 - 람다를 인자로 받은 함수의 이름

addEventListener("keyup"){return@addEventListener}

```
fun app(){
    document.querySelector("#base")?.innerHTML = """<input id="input"/><div id="result"></div>"""
    document.querySelector("#input")?.addEventListener("keyup", a@{
        if((it as KeyboardEvent).keyCode != 13) return@a
        val input = it.target as HTMLInputElement
        val v = input.value
        document.querySelector("#result")?.innerHTML = "$v = ${calc(v)}"
        input.value = ""
    })
}
val cleanUp = """[^.\d-+*\/]""".toRegex()
val mulDiv = """((?:\+-)?[.\d]+)([*\/])((?:\+-)?[.\d]+)""".toRegex()
val paren = """\(([^()]*)\)""".toRegex()
fun ex(v:String) = v.replace(cleanUp,"").replace("--","+").replace(mulDiv){
    val (_, left, op, right) = it.groupValues
    val l = left.replace("+", "").toDouble()
    val r = right.replace("+", "").toDouble()
    "${if(op == "*") l * r else l / r}".replace("-","+-")
}.split('+').fold(0.0) { acc, v -> acc + if(v.isBlank()) 0.0 else v.toDouble()}
fun calc(v:String):Double{
    var r = v
    while(paren.containsMatchIn(r)) r = r.replace(paren){"${ex(it.groupValues[1])}"}
    return ex(r)
}
```

람다의 return은 감싸고 있는 함수의 return

인자로 전달된 람다는 감싸고 있는 함수가 없음

인자람다의 return은 반드시 기명return만 가능

기명람다 - 람다에 직접 이름을 부여하는 방식

name@{return@name}

자동이름 - 람다를 인자로 받은 함수의 이름

addEventListener("keyup"){return@addEventListener}