

kotlin

tutorial version 1.0

Kotlin tutorial

The screenshot shows a website header with a logo and navigation links: Tutorials, Exercises, Certificates, Services, and a search bar. Below the header is a dark navigation bar with links for QL, JQUERY, EXCEL, XML, DJANGO, NUMPY, PANDAS, and NO. The main content area is titled "Kotlin Tutorial" and features a sidebar menu on the left with items like "Kotlin HOME", "Kotlin Intro", "Kotlin Get Started", "Kotlin Syntax", "Kotlin Output", "Kotlin Comments", "Kotlin Variables", "Kotlin Data Types", "Kotlin Operators", "Kotlin Strings", "Kotlin Booleans", "Kotlin If...Else", and "Kotlin When". The main content area has a large heading "Kotlin Tutorial", a green button labeled "< Home", and a light green box with the text "Learn Kotlin".

<https://www.w3schools.com/kotlin/index.php>

The screenshot shows the Kotlin documentation website. The header includes the Kotlin logo and version number "2.0.20", along with a search bar and navigation links for "Developers", "Essentials", and "Więcej". The main content area is titled "Hello world" and includes a sub-heading "Are unit tests part" and a link to "Edit page" with the text "Last modified: 25 September 2024". A table of contents is visible on the left, listing items such as "Hello world", "Basic types", "Collections", "Control flow", "Functions", "Classes", and "Null safety".

<https://kotlinlang.org/docs/kotlin-tour-hello-world.html>

The screenshot shows a lesson page titled "Lesson 1: Kotlin basics" with a bookmark icon. The page content includes the text "Get started developing in Kotlin, and learn the basics of the Kotlin programming language: data types, operators, variables, control structures, and nullable versus non-nullable variables." and a footer that reads "2 działania • Testy: 1".

<https://developer.android.com/courses/pathways/android-development-with-kotlin-1#code-lab-https://developer.android.com/code-labs/android-development-kotlin-1.1>

Kotlin IDE

We're committed to giving back to our wonderful community, which is why IntelliJ IDEA Community Edition is completely free to use



IntelliJ IDEA Community Edition

The IDE for Java and Kotlin enthusiasts

Download

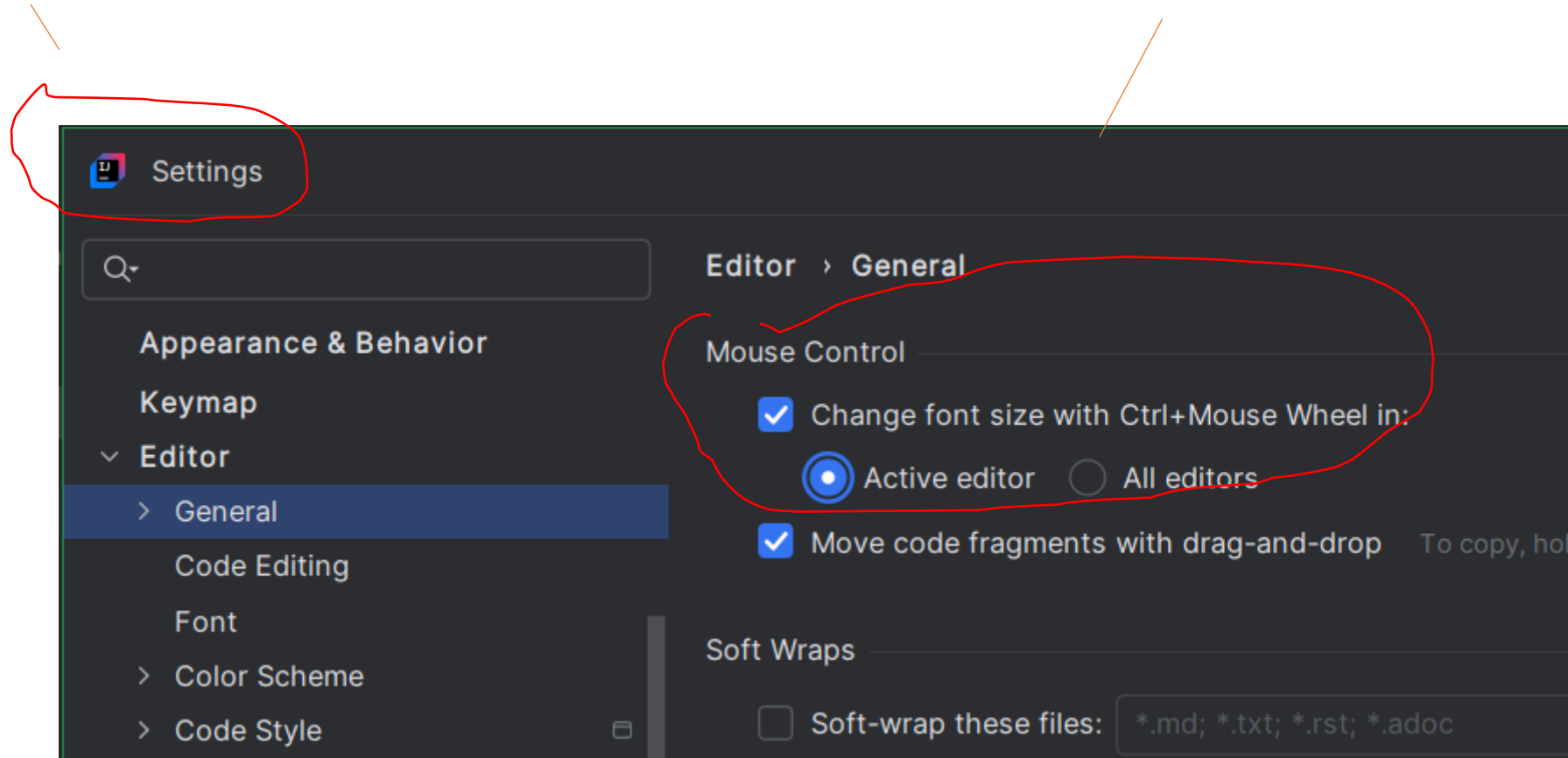
.exe (Windows) ▼

Free, built on open source

IntelliJ IDEA Community Edition

ctrl + alt + s

zmiana wielkości czcionek
za pomocą kółka myszki



String Templates

In Kotlin, string templates allow you to create strings that include variables and expressions, making string manipulation more convenient and readable. You can insert variables or expressions directly into a string by using the `$` symbol for variables and `${}` for expressions that need evaluation.

Basic Usage

1. **Variable Substitution:** You can integrate a variable directly into a string using the `$` sign.

```
val name = "Alice"  
val greeting = "Hello, $name!"  
println(greeting) // Output: Hello, Alice!
```

2. **Expressions:** If you want to perform an expression or use a complex calculation, use `${}`.

```
val x = 5  
val y = 10  
val result = "The sum of $x and $y is ${x + y}."  
println(result) // Output: The sum of 5 and 10 is 15.
```

Kotlin ?

In Kotlin, the question mark (?) is used primarily to denote nullable types. Here's a brief overview of how it works:

1. Nullable Types

In Kotlin, by default, all types are non-nullable. This means you cannot assign `null` to a variable of a non-nullable type. If you want to allow a variable to hold `null`, you must declare it as nullable by adding a question mark after the type.

Example:

```
var nonNullableString: String = "Hello"  
// nonNullableString = null // This will cause a compilation error  
  
var nullableString: String? = "Hello"  
nullableString = null // This is allowed
```

Kotlin ?.

2. Safe Calls

When working with nullable types, Kotlin provides a safe call operator (`?.`). This operator allows you to call methods or access properties on an object while safely handling the possibility of `null`.

Example:

```
val length: Int? = nullableString?.length // If nullableString is null, length will also be null
```

Kotlin ?:

3. Elvis Operator

The Elvis operator (?:) is used to provide a default value in case the left-hand side expression is null.

Example:

```
val length: Int = nullableString?.length ?: 0 // If nullableString is null, length will be 0
```


Kotlin !!

4. Not-null Assertion

If you are sure that a nullable variable is not null at a certain point in your code, you can use the non-null assertion operator (`!!`). However, if it is null, this will throw a `NullPointerException`.

Example:

```
val length: Int = nullableString!!.length // Make sure nullableString is not null to  
avoid exceptions
```

nullable variables in Kotlin

Use nullability in Kotlin

🌐 Polski ▼

Zaloguj się

1 Before you begin

2 **Use nullable variables**

3 Handle nullable variables

4 Conclusion

2. Use nullable variables

What is `null`?

In Unit 1, you learned that when you declare a variable, you need to assign it a value immediately. For example, when you declare a `favoriteActor` variable, you may assign it a `"Sandra Oh"` string value immediately.

```
val favoriteActor = "Sandra Oh"
```



"Sandra Oh"

Wstecz

Dalej

🐛 Zgłoś pomyłkę

nullable variables in Kotlin

The screenshot shows the Kotlin v2.0.20 documentation website. The top navigation bar includes links for Solutions, Docs, Community, Teach, and Play. The left sidebar contains a navigation menu with categories like JVM, Spring, and Java to Kotlin migration guides. The main content area is titled 'Nullability in Java and Kotlin' and includes a breadcrumb trail: Platforms / JVM / Java to Kotlin migration guides / Nullability. The page content defines nullability and explains the difference between Java's and Kotlin's approaches to handling nullable variables. A right sidebar lists related topics such as 'Nullability in Java and Kotlin', 'Support for nullable types', and 'Platform types'.

Kotlin v2.0.20

Solutions Docs Community Teach Play

Platforms / JVM / Java to Kotlin migration guides / Nullability

Nullability in Java and Kotlin

[Edit page](#) Last modified: 25 September 2024

Nullability is the ability of a variable to hold a `null` value. When a variable contains `null`, an attempt to dereference the variable leads to a `NullPointerException`. There are many ways to write code in order to minimize the probability of receiving null pointer exceptions.

This guide covers differences between Java's and Kotlin's approaches to handling possibly nullable variables. It will help you migrate from Java to Kotlin and write your code in authentic Kotlin style.

The first part of this guide covers the most important difference – support for nullable types

- Nullability in Java and Kotlin
- Support for nullable types
- Platform types
- Support for definitely non nullable types
- Checking the result of a function call
- Default values instead of null
- Functions returning a value or null
- Aggregate operations
- Casting types safely

Kotlin Cheat Sheet



CHEAT SHEET

BASICS

“Hello, World” program

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

Declaring function

```
fun sum(a: Int, b: Int): Int {  
    return a + b  
}
```

Single-expression function

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

Declaring variables

```
val name = "Marcin" // Can't be changed  
var age = 5         // Can be changed
```

CLASSES

Primary constructor

val declares a read-only property, var a mutable one

```
class Person(val name: String, var age: Int)  
// name is read-only, age is mutable
```

Inheritance

```
open class Person(val name: String) {  
    open fun hello() = "Hello, I am $name"  
    // Final by default so we need open  
}  
class PolishPerson(name: String) : Person(name) {  
    override fun hello() = "Dzień dobry, jestem $name"  
}
```

Properties with assessors

```
class Person(var name: String, var surname: String) {
```