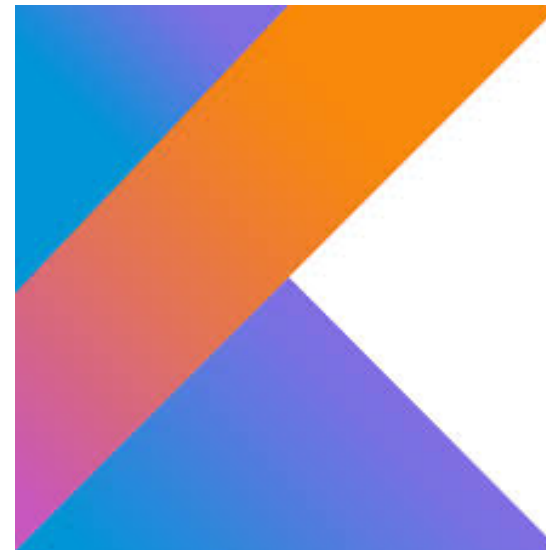


Introduction into Kotlin



based on

Neil Smyth
2020, Payload Media Inc.



History on Kotlin which ..

- is a *f-oo ;-)* programming language (Kotlin-first since Google I/O in 2019 for Android mobile development)
- created by JetBrains <https://www.jetbrains.com/> at Prague (IntelliJ, AndroidStudio)
- named after an island located in the Baltic Sea
- to make code concise (kurz & prägnant), easier to understand and safer than others
- rather than to re-invents the wheel, integrates with and work alongside Java
- generates byte code, uses JVM (like Java, Groovy, Scala, ..) and hence can apply Java frameworks seamlessly and vice versa, Java can call Kotlin code
- is a transpile language, ie source-to-source compiler to Java, JavaScript, ..

Try experimenting with Kotlin

<https://try.kotl.in>

try.kotl.in.org is now obsolete. Please use [Kotlin Playground](#) instead.

Kotlin LEARN COMMUNITY TRY ONLINE

Shortcuts Convert from Java Fullscreen

Examples > Hello, world! > Simplest version > Simplest version.kt

- Examples
 - Hello, world!
 - Simplest version
 - Simplest version.kt**
 - Reading a name from the...
 - Reading many names fro...
 - A multi-language Hello
 - An object-oriented Hello
 - Basic syntax walk-through
 - Destructuring declarations and
 - Delegated properties
 - Callable references
 - Longer examples
 - Problems
 - Canvas
 - Kotlin Koans 0/42
 - Kotlin in Action
 - Advent of Code (log in)
 - My programs (log in)

Project content loaded

Arguments JVM Run

Program arguments

```
1 /**
2  * We declare a package-level function main which returns Unit and takes
3  * an Array of strings as a parameter. Note that semicolons are optional.
4  */
5
6 fun main(args: Array<String>) {
7     println("Hello, world!")
8 }
```

[X] Cookies and IP addresses allow us to deliver and improve our web content and to provide you with a personalized experience. Our website uses cookies and collects your IP address for these purposes. [Learn more](#)

| JetBrains may use cookies and my IP address to collect individual statistics and to provide me with personalized offers and ads subject to the [Privacy Policy](#) and the [Terms of Use](#). JetBrains may use [third-party services](#) for this purpose. I can revoke my consent at any time by visiting the [Opt-Out page](#).

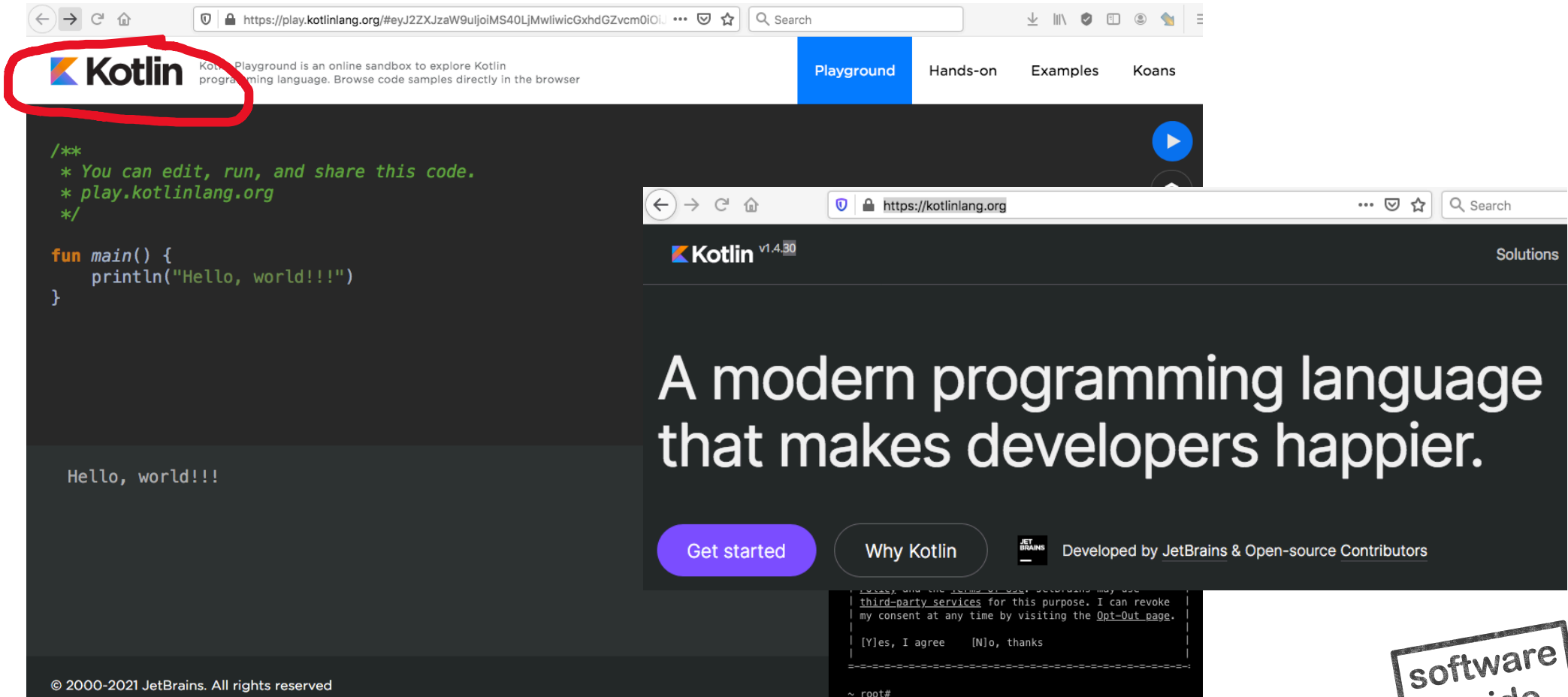
| [Y]es, I agree [N]o, thanks

~ root#



Play at “Kotlin Playground”

and “Get Started” in <https://kotlinlang.org/>



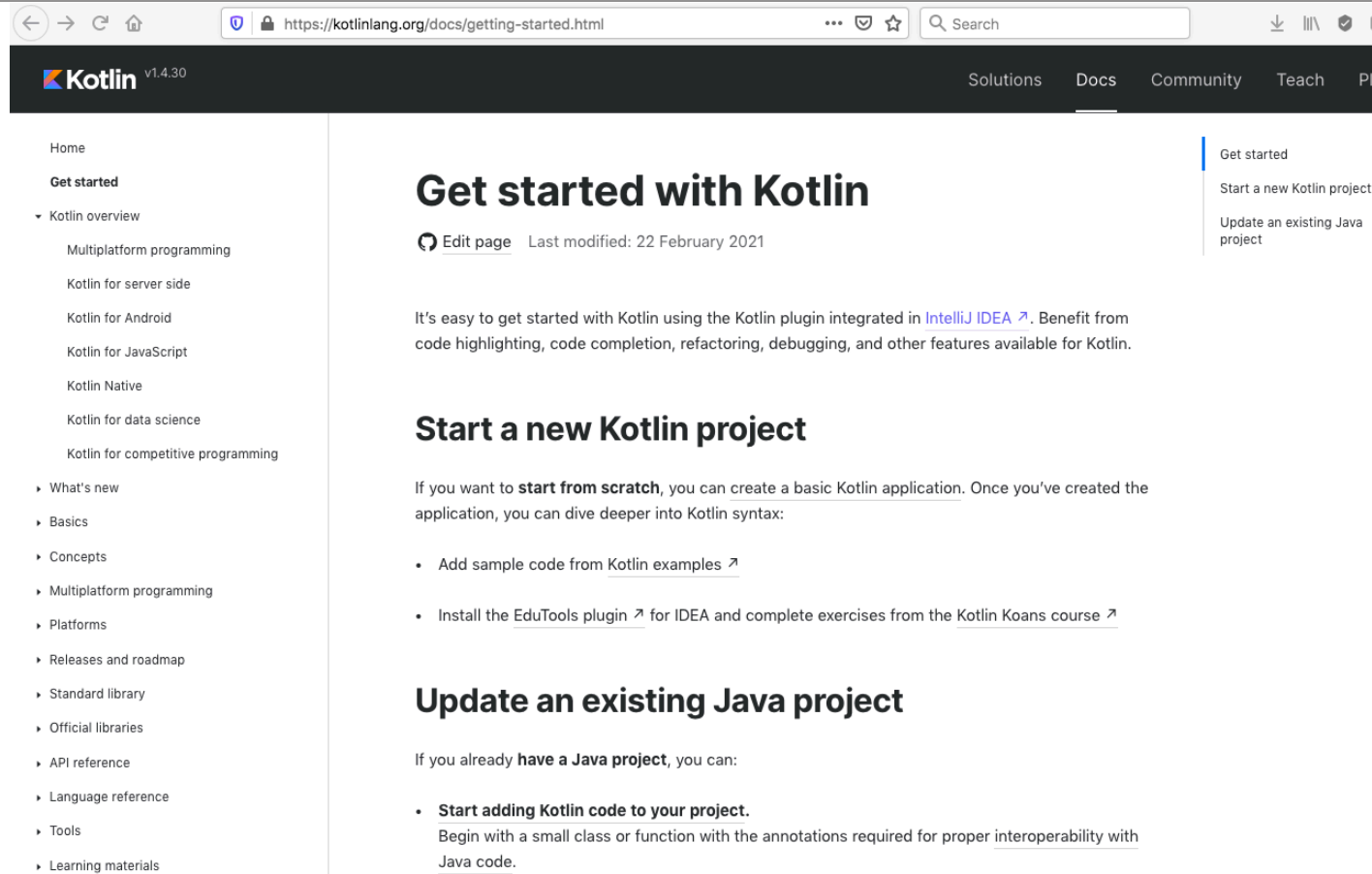
The image shows two overlapping browser windows. The top window is the Kotlin Playground interface, with the URL `https://play.kotlinlang.org/#eyJ2ZXJzaW9uIjoMS40LjMwIiwicGxhdGZvcn0iOi...`. The Kotlin logo is circled in red. The code editor contains the following Kotlin code:

```
/**
 * You can edit, run, and share this code.
 * play.kotlinlang.org
 */

fun main() {
    println("Hello, world!!!")
}
```

The output shows "Hello, world!!!". The bottom window is the Kotlin homepage at `https://kotlinlang.org`, featuring the text "A modern programming language that makes developers happier." and buttons for "Get started" and "Why Kotlin".

Get Started



The screenshot shows the Kotlin documentation website in a web browser. The browser's address bar displays the URL <https://kotlinlang.org/docs/getting-started.html>. The website's header includes the Kotlin logo (v1.4.30) and navigation links for Solutions, Docs, Community, Teach, and Play. A left sidebar contains a table of contents with links to Home, Get started, Kotlin overview, and various Kotlin-specific topics. The main content area is titled "Get started with Kotlin" and includes an "Edit page" link and a "Last modified" date. The text explains that Kotlin can be used with IntelliJ IDEA and lists benefits like code highlighting and completion. It then provides instructions on how to start a new Kotlin project or update an existing Java project, with links to Kotlin examples, the Kotlin Koans course, and the EduTools plugin.

Home

Get started

▼ Kotlin overview

- Multiplatform programming
- Kotlin for server side
- Kotlin for Android
- Kotlin for JavaScript
- Kotlin Native
- Kotlin for data science
- Kotlin for competitive programming

► What's new

► Basics

► Concepts

► Multiplatform programming

► Platforms

► Releases and roadmap

► Standard library

► Official libraries

► API reference

► Language reference

► Tools

► Learning materials

Get started with Kotlin

[Edit page](#) Last modified: 22 February 2021

It's easy to get started with Kotlin using the Kotlin plugin integrated in [IntelliJ IDEA](#). Benefit from code highlighting, code completion, refactoring, debugging, and other features available for Kotlin.

Start a new Kotlin project

If you want to **start from scratch**, you can [create a basic Kotlin application](#). Once you've created the application, you can dive deeper into Kotlin syntax:

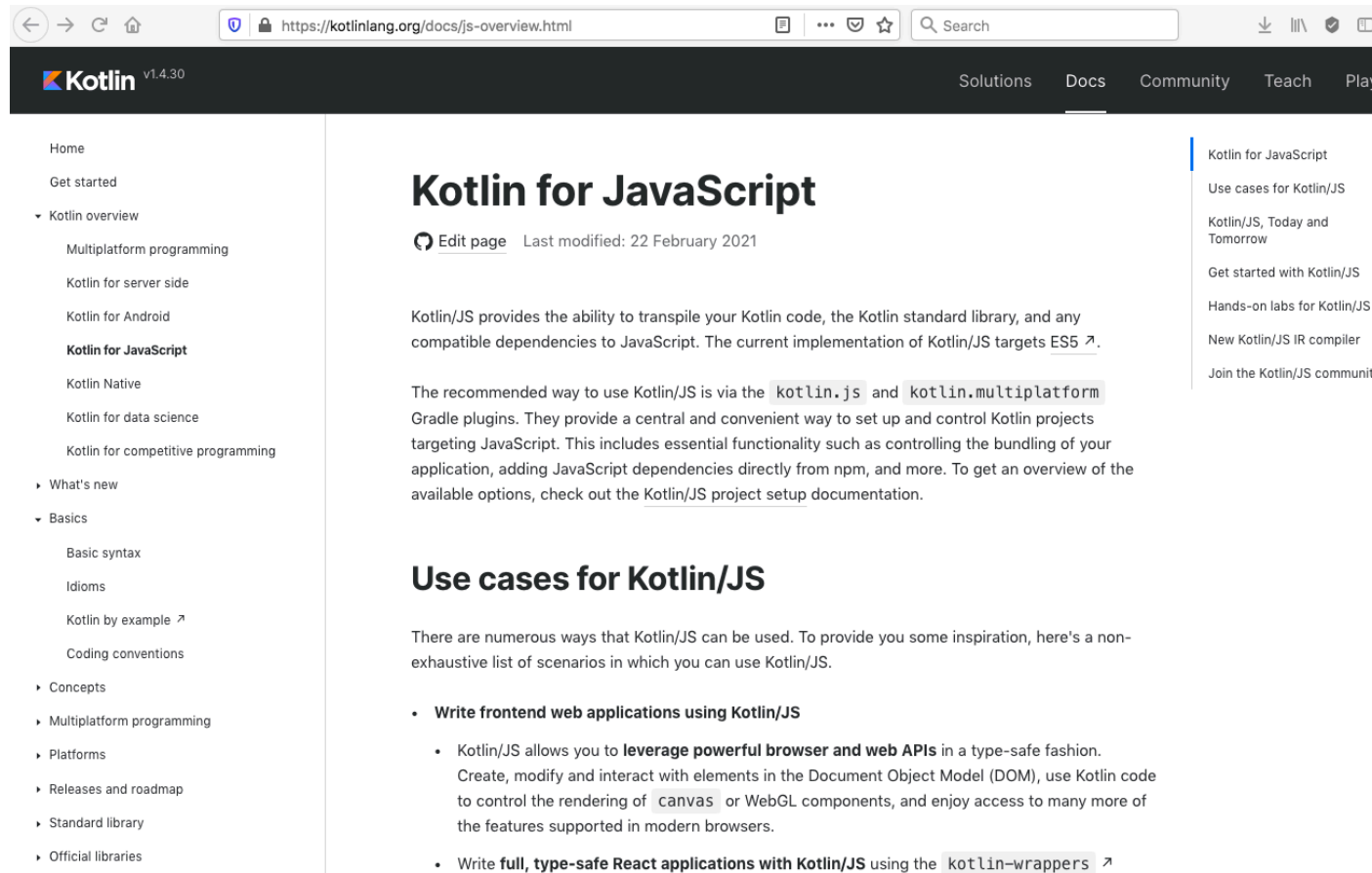
- Add sample code from [Kotlin examples](#)
- Install the [EduTools plugin](#) for IDEA and complete exercises from the [Kotlin Koans course](#)

Update an existing Java project

If you already **have a Java project**, you can:

- **Start adding Kotlin code to your project.**
Begin with a small class or function with the annotations required for proper [interoperability with Java code](#).

Multi-Platform, Transpile, Native, ..



The screenshot shows the Kotlin for JavaScript documentation page. The browser address bar displays `https://kotlinlang.org/docs/js-overview.html`. The page header includes the Kotlin logo and version `v1.4.30`, along with navigation links: Solutions, Docs, Community, Teach, and Play. A left sidebar contains a table of contents with links to Home, Get started, Kotlin overview, Multiplatform programming, Kotlin for server side, Kotlin for Android, Kotlin for JavaScript (highlighted), Kotlin Native, Kotlin for data science, Kotlin for competitive programming, What's new, Basics, Basic syntax, Idioms, Kotlin by example, Coding conventions, Concepts, Multiplatform programming, Platforms, Releases and roadmap, Standard library, and Official libraries. The main content area features the title "Kotlin for JavaScript" with an "Edit page" link and a "Last modified: 22 February 2021" timestamp. The text describes Kotlin/JS as a transpiler for Kotlin code to JavaScript, targeting ES5. It recommends using `kotlin.js` and `kotlin.multiplatform` Gradle plugins. A section titled "Use cases for Kotlin/JS" lists two main scenarios: writing frontend web applications and writing full, type-safe React applications. A right sidebar provides additional links like "Kotlin for JavaScript", "Use cases for Kotlin/JS", "Kotlin/JS, Today and Tomorrow", "Get started with Kotlin/JS", "Hands-on labs for Kotlin/JS", "New Kotlin/JS IR compiler", and "Join the Kotlin/JS community".

Kotlin for JavaScript

[Edit page](#) Last modified: 22 February 2021

Kotlin/JS provides the ability to transpile your Kotlin code, the Kotlin standard library, and any compatible dependencies to JavaScript. The current implementation of Kotlin/JS targets ES5 ↗.

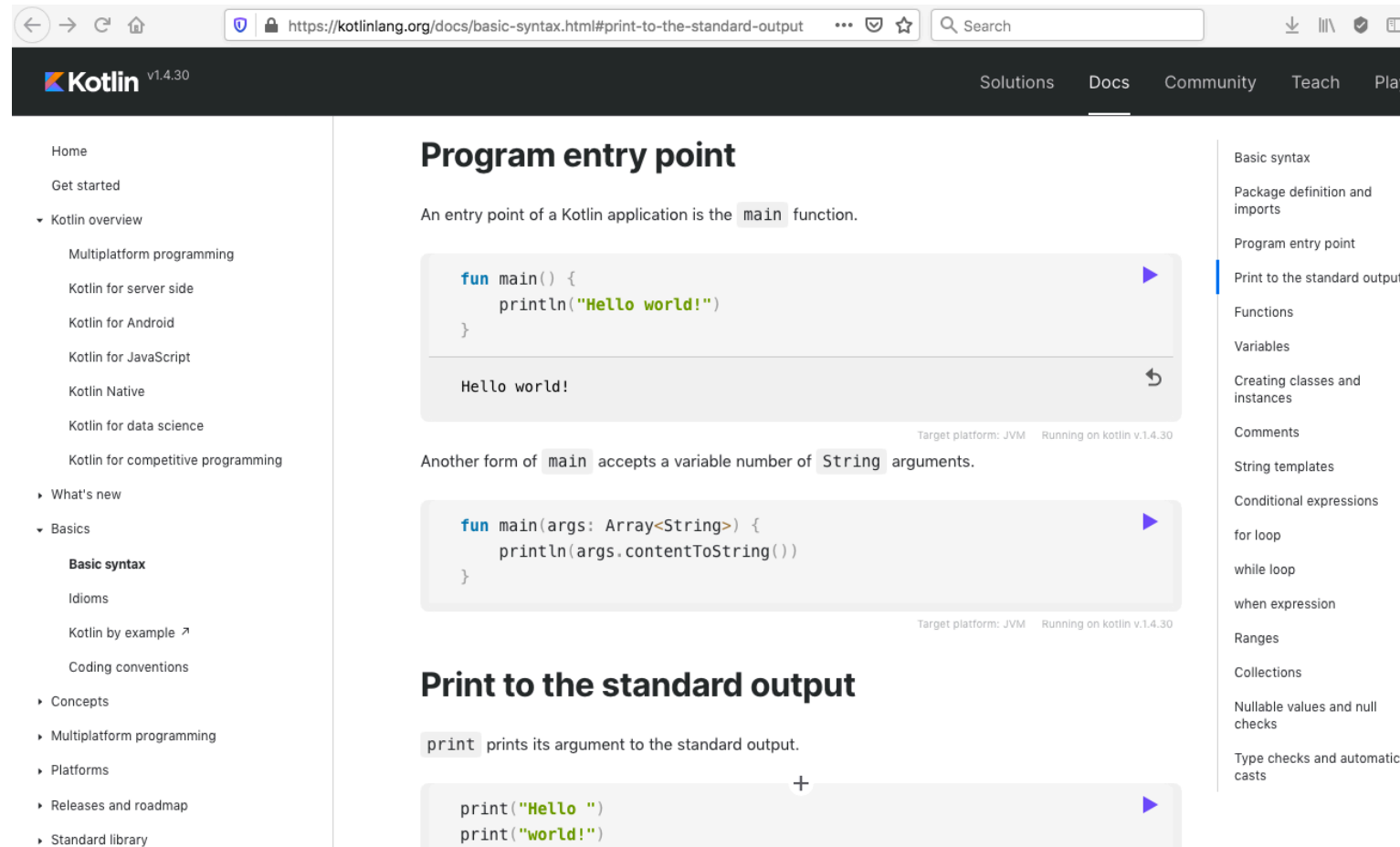
The recommended way to use Kotlin/JS is via the `kotlin.js` and `kotlin.multiplatform` Gradle plugins. They provide a central and convenient way to set up and control Kotlin projects targeting JavaScript. This includes essential functionality such as controlling the bundling of your application, adding JavaScript dependencies directly from npm, and more. To get an overview of the available options, check out the [Kotlin/JS project setup](#) documentation.

Use cases for Kotlin/JS

There are numerous ways that Kotlin/JS can be used. To provide you some inspiration, here's a non-exhaustive list of scenarios in which you can use Kotlin/JS.

- **Write frontend web applications using Kotlin/JS**
 - Kotlin/JS allows you to **leverage powerful browser and web APIs** in a type-safe fashion. Create, modify and interact with elements in the Document Object Model (DOM), use Kotlin code to control the rendering of `canvas` or WebGL components, and enjoy access to many more of the features supported in modern browsers.
- Write **full, type-safe React applications with Kotlin/JS** using the `kotlin-wrappers` ↗

Learn by on-line Play Capabilites



The screenshot shows the Kotlin documentation website. The left sidebar contains a navigation menu with categories like 'Home', 'Get started', 'Kotlin overview', 'Multiplatform programming', 'Kotlin for server side', 'Kotlin for Android', 'Kotlin for JavaScript', 'Kotlin Native', 'Kotlin for data science', 'Kotlin for competitive programming', 'What's new', 'Basics', 'Concepts', 'Multiplatform programming', 'Platforms', 'Releases and roadmap', and 'Standard library'. The 'Basics' section is expanded, showing 'Basic syntax', 'Idioms', 'Kotlin by example', and 'Coding conventions'. The main content area is titled 'Program entry point' and explains that the `main` function is the entry point of a Kotlin application. It shows two examples of `main` functions: one that prints 'Hello world!' and another that takes an array of strings and prints their content. Below this, the 'Print to the standard output' section explains that the `print` function prints its argument to the standard output. It shows an example of using `print` to print 'Hello ' and 'world!' on separate lines. The right sidebar contains a list of topics: 'Basic syntax', 'Package definition and imports', 'Program entry point', 'Print to the standard output', 'Functions', 'Variables', 'Creating classes and instances', 'Comments', 'String templates', 'Conditional expressions', 'for loop', 'while loop', 'when expression', 'Ranges', 'Collections', 'Nullable values and null checks', and 'Type checks and automatic casts'. The 'Print to the standard output' topic is currently selected.

Home

Get started

Kotlin overview

Multiplatform programming

Kotlin for server side

Kotlin for Android

Kotlin for JavaScript

Kotlin Native

Kotlin for data science

Kotlin for competitive programming

What's new

Basics

Basic syntax

Idioms

Kotlin by example ↗

Coding conventions

Concepts

Multiplatform programming

Platforms

Releases and roadmap

Standard library

Program entry point

An entry point of a Kotlin application is the `main` function.

```
fun main() {  
    println("Hello world!")  
}
```

Hello world!

Target platform: JVM Running on kotlin v:1.4.30

Another form of `main` accepts a variable number of `String` arguments.

```
fun main(args: Array<String>) {  
    println(args.contentToString())  
}
```

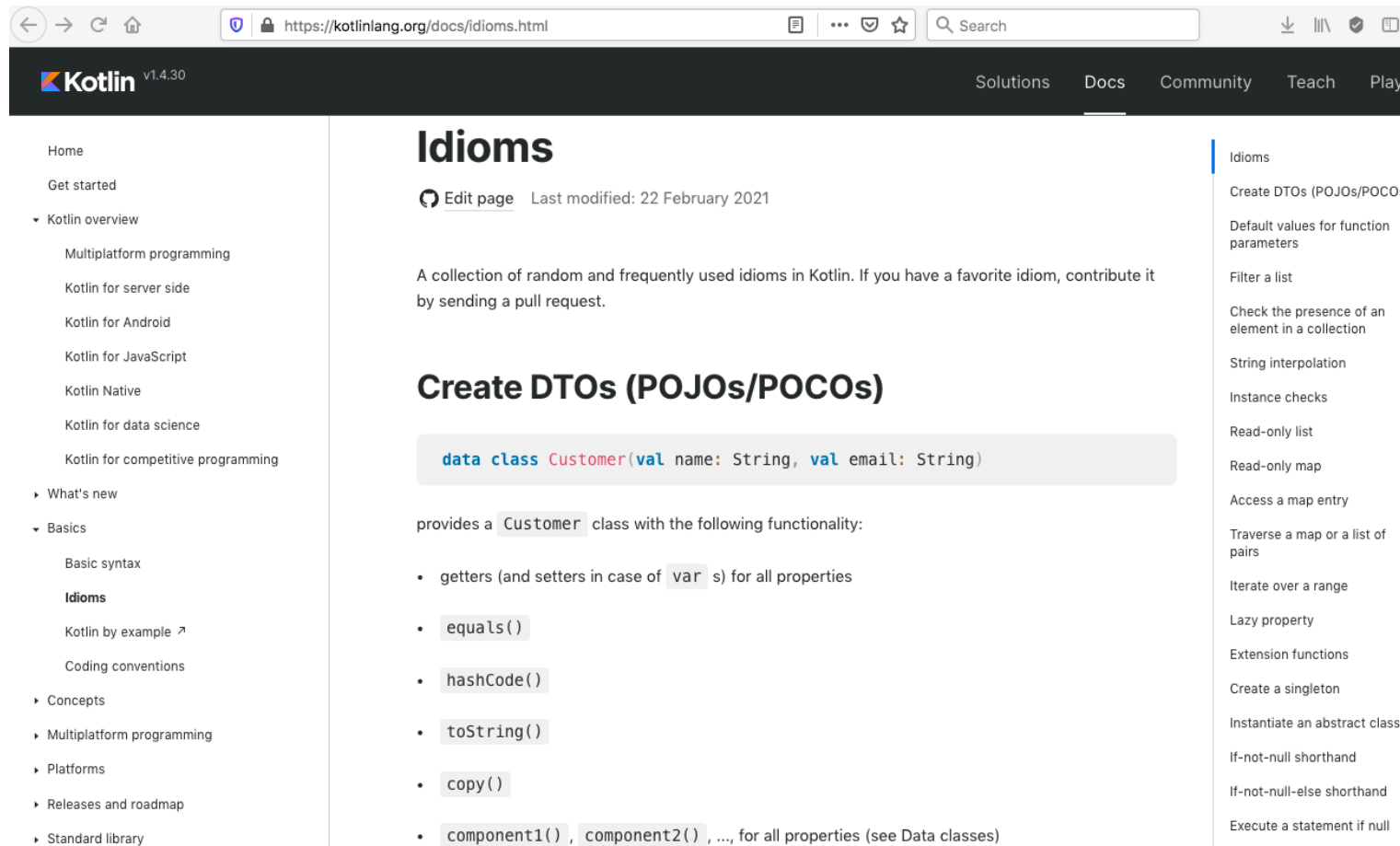
Target platform: JVM Running on kotlin v:1.4.30

Print to the standard output

`print` prints its argument to the standard output.

```
print("Hello ")  
print("world!")
```


.. on basic, favorite “(s)elected” Idioms



The screenshot shows the Kotlin documentation website. The browser address bar displays `https://kotlinlang.org/docs/idioms.html`. The page title is "Idioms" and it was last modified on 22 February 2021. The main content area describes a collection of random and frequently used idioms in Kotlin. A specific example is provided for "Create DTOs (POJOs/POCOs)", showing a `data class Customer` with `name` and `email` properties. A list of methods for the `Customer` class is shown, including `getters`, `equals()`, `hashCode()`, `toString()`, `copy()`, and `component1()`, `component2()`, etc. A sidebar on the left contains a navigation menu with links to Home, Get started, Kotlin overview, Multiplatform programming, Kotlin for server side, Kotlin for Android, Kotlin for JavaScript, Kotlin Native, Kotlin for data science, Kotlin for competitive programming, What's new, Basics, Basic syntax, Idioms, Kotlin by example, Coding conventions, Concepts, Multiplatform programming, Platforms, Releases and roadmap, and Standard library. A sidebar on the right lists various idioms: Create DTOs (POJOs/POCOs), Default values for function parameters, Filter a list, Check the presence of an element in a collection, String interpolation, Instance checks, Read-only list, Read-only map, Access a map entry, Traverse a map or a list of pairs, Iterate over a range, Lazy property, Extension functions, Create a singleton, Instantiate an abstract class, If-not-null shorthand, If-not-null-else shorthand, and Execute a statement if null.

Idioms

[Edit page](#) Last modified: 22 February 2021

A collection of random and frequently used idioms in Kotlin. If you have a favorite idiom, contribute it by sending a pull request.

Create DTOs (POJOs/POCOs)

```
data class Customer(val name: String, val email: String)
```

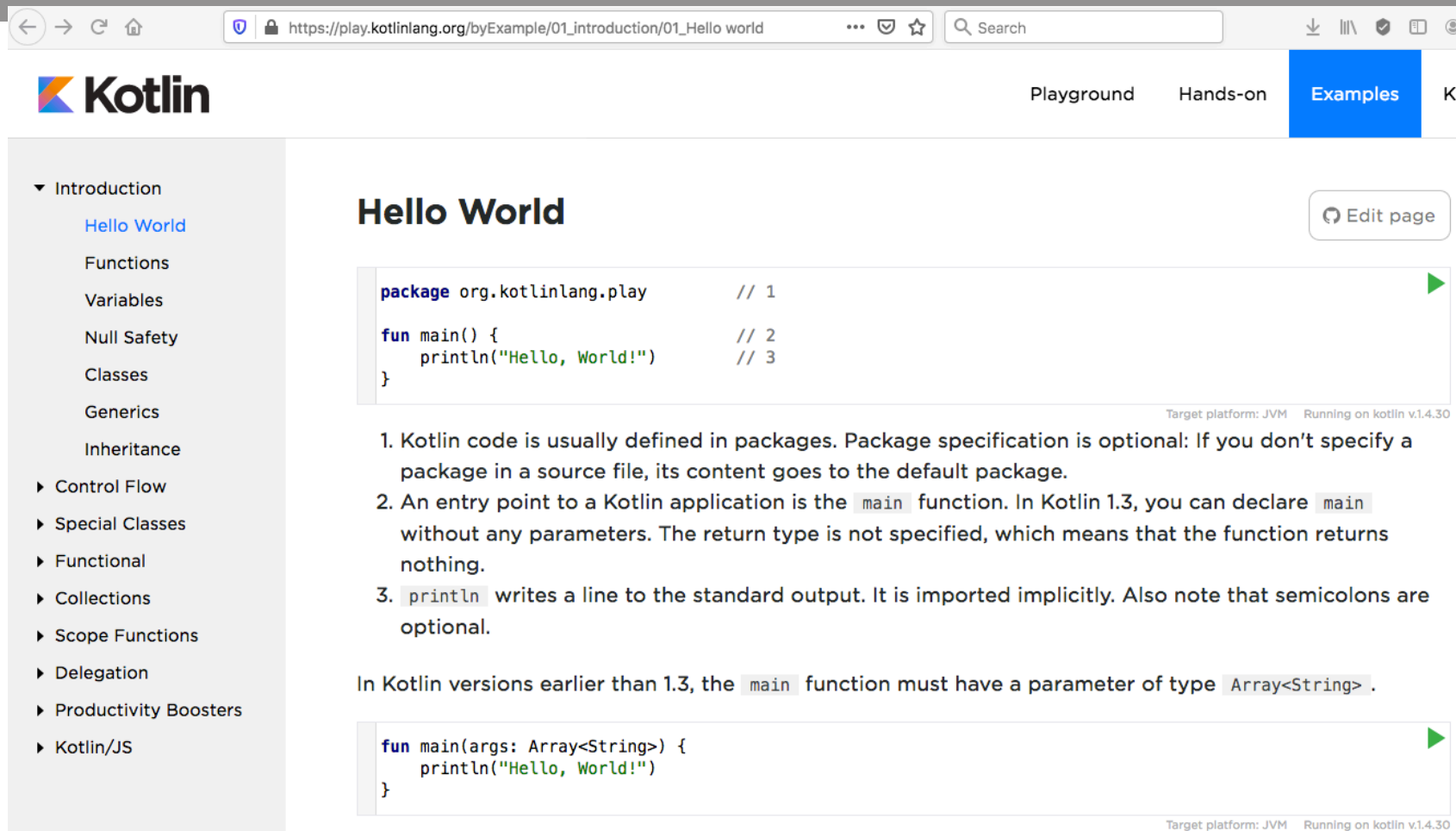
provides a `Customer` class with the following functionality:

- getters (and setters in case of `var` s) for all properties
- `equals()`
- `hashCode()`
- `toString()`
- `copy()`
- `component1()`, `component2()`, ..., for all properties (see Data classes)

Idioms

- Create DTOs (POJOs/POCOs)
- Default values for function parameters
- Filter a list
- Check the presence of an element in a collection
- String interpolation
- Instance checks
- Read-only list
- Read-only map
- Access a map entry
- Traverse a map or a list of pairs
- Iterate over a range
- Lazy property
- Extension functions
- Create a singleton
- Instantiate an abstract class
- If-not-null shorthand
- If-not-null-else shorthand
- Execute a statement if null

.. to play in a world of Examples



The screenshot shows the Kotlin Playground website. The browser address bar displays `https://play.kotlinlang.org/byExample/01_introduction/01_Hello world`. The website has a navigation bar with 'Kotlin' logo, 'Playground', 'Hands-on', and 'Examples' (highlighted in blue). A left sidebar lists various Kotlin topics: Introduction (with 'Hello World' selected), Functions, Variables, Null Safety, Classes, Generics, Inheritance, Control Flow, Special Classes, Functional, Collections, Scope Functions, Delegation, Productivity Boosters, and Kotlin/JS. The main content area is titled 'Hello World' and includes an 'Edit page' button. It displays a Kotlin code snippet for a 'Hello World' program, followed by three numbered points explaining Kotlin syntax: package specification, the 'main' function, and the 'println' function. Below the text, there is another code snippet showing the 'main' function with an `Array<String>` parameter. The status bar at the bottom of the code editor indicates 'Target platform: JVM' and 'Running on kotlin v1.4.30'.

```
package org.kotlinlang.play // 1

fun main() {                // 2
    println("Hello, World!") // 3
}
```

Target platform: JVM Running on kotlin v1.4.30

1. Kotlin code is usually defined in packages. Package specification is optional: If you don't specify a package in a source file, its content goes to the default package.
2. An entry point to a Kotlin application is the `main` function. In Kotlin 1.3, you can declare `main` without any parameters. The return type is not specified, which means that the function returns nothing.
3. `println` writes a line to the standard output. It is imported implicitly. Also note that semicolons are optional.


In Kotlin versions earlier than 1.3, the `main` function must have a parameter of type `Array<String>`.

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

Target platform: JVM Running on kotlin v1.4.30



.. and test yourself by koans



Progress:0%

▼ Introduction

Hello, world!

Named arguments

Default arguments

Triple-quoted strings

String templates

Nullable types

Nothing type

Lambdas

► Classes

► Conventions

► Collections

► Properties

```
fun start(): String = TODO()
```

Ein **Kōan** (jap. 公案; chinesisch 公案, Pinyin gōng'àn, W.-G. kung-an – „Öffentlicher Aushang“; hgl. 공안, gong-an; andere gebräuchliche Transkriptionen aus dem Koreanischen: Kung-an, Kungan; viet. công án) ist im chinesischen Chan- bzw. japanischen Zen-Buddhismus eine kurze Anekdote oder Sentenz, die eine beispielhafte Handlung oder Aussage eines Zen-Meisters, ganz selten auch eines Zen-Schülers, darstellt.

Verlauf und Pointen dieser speziellen Anekdoten wirken auf den Laien meist vollkommen paradox, unverständlich oder sinnlos. In der Folge kommt es manchmal zur fehlerhaften Übertragung des Begriffs Kōan auf andere unsinnige Kurzgeschichten.

Die Vorläufer der Kōans waren berühmte Fragen und Antworten zwischen Meister und Schüler während der frühen Tang- und Song-Zeit, Fragmente einiger buddhistischer Sutras, bedeutungsvolle Reden von Chan-Meistern und Anekdoten über diese Meister. Trotz ihrer vordergründigen Unvernünftigkeit und Sinnlosigkeit verfügen sie über einen historischen Kern, der auch intellektuell nachvollziehbar ist und Aspekte der Chan-Philosophie ausdrückt. Im Chan und Zen werden Kōans als Meditationsobjekte benutzt.

Das bekannteste Kōan, das inzwischen auch im Westen Allgemeingut geworden ist, ist die Frage nach dem Geräusch einer einzelnen klatschenden Hand (Hakuins Sekishu, von Meister Hakuin Ekaku).

<https://de.wikipedia.org/wiki/Kōan>

PlaygroundHands-onExamplesKoans


1/8

Simple Functions

Check out the [function syntax](#) and change the code to make the function `start` return the string `"OK"`.

In the Kotlin Koans tasks the function `TODO()` will throw an exception. To complete Kotlin Koans you need to replace this function invocation with meaningful code according to the problem.

RevertShow answer



EDK

Feb 2021

POS

11

Exercise: (im)prove comparision

at <https://www.guru99.com/kotlin-vs-java-difference.html>



Kotlin Vs Java

Here, are differences between Kotlin vs Java

Kotlin	Java
Kotlin allows users to create an extension function.	Java doesn't offer any extension functions.
Kotlin doesn't require too much work for data classes.	Java developers write and construct a lot of elements to develop classes
Kotlin doesn't offer implicit conversions.	Java supports implicit conversions.
There are no null variables or objects in Kotlin.	Null variable or objects are part of Java language.
Kotlin combines features of both object-oriented and functional programming.	Java is limited to object-oriented programming.
Kotlin doesn't support static members.	Java uses static members.



Comparision cont'd.

Variables of a primitive type are objects	Variables of a primitive type aren't objects
In Kotlin, we can have one or more secondary constructors.	In Java, we can't have secondary constructors. However, it can have multiple constructors.
Kotlin string template also supports expression.	Java string doesn't support expression like Kotlin.
It's quite easier to deploy Kotlin code.	It is hard to deploy Java code.
Kotlin programs don't require semicolons in their program.	Java program does need a semicolon.
In Kotlin, coroutine are concurrency design pattern which can be used to simplify code.	Java uses two coroutine options as 1) Rx Java and 2) Project loom.
Kotlin doesn't have any wildcard-types.	Wide-card is available in Java.
Kotlin's type of system has inbuilt null safety.	NullPonter Exception is mainly responsible for the development of Java and Android.

Comparision cont'd. – still true?

Smart cast feature is available in Kotlin.

Smart cast feature is not available in Java.

Kotlin doesn't require any variable datatype specifications

Java requires variable datatype specifications.

Kotlin supports Lambda Expression.

Java doesn't support Lambda expression.

Lazy-Loading feature is available in Kotlin.

This feature is not available in Java.

Language scripting capabilities allow you to use Kotlin directly in your Gradle build scripts

Java does not offer language scripting capabilities.

It supports modern programming concepts like delegates, extension, higher-order functions.

Java supports OOPS programming concept.

The average salary for a java developer is \$104,793 per year.

The average salary for "kotlin" ranges from approximately \$107,275 per year for Software Engineer to \$121,034 per year for Android Developer.