

Java 11 New Features Overview

👤 [Guide](#) 📄 Java 📌 New Java Features 📄 About 1343 words 🕒 About 4 minutes

Java 11 was officially released on September 25, 2018. This is a significant release! The biggest difference between Java 11 and Java 9, released in September 2017, and Java 10, released in March 2018, is its long-term support (LTS). **Oracle has stated that Java 11 will continue to be supported until September 2026. This is the first LTS release since Java 8.**

The following figure is the timeline of Oracle JDK support given by Oracle officially.

Oracle Java SE Support Roadmap**†				
Release	GA Date	Premier Support Until	Extended Support Until	Sustaining Support
7 (LTS)	July 2011	July 2019	July 2022*****	Indefinite
8 (LTS)**	March 2014	March 2022	December 2030*****	Indefinite
9 (non-LTS)	September 2017	March 2018	Not Available	Indefinite
10 (non-LTS)	March 2018	September 2018	Not Available	Indefinite
11 (LTS)	September 2018	September 2023	September 2026	Indefinite
12 (non-LTS)	March 2019	September 2019	Not Available	Indefinite
13 (non-LTS)	September 2019	March 2020	Not Available	Indefinite
14 (non-LTS)	March 2020	September 2020	Not Available	Indefinite
15 (non-LTS)	September 2020	March 2021	Not Available	Indefinite
16 (non-LTS)	March 2021	September 2021	Not Available	Indefinite
17 (LTS)	September 2021	September 2026****	September 2029****	Indefinite
18 (non-LTS)***	March 2022	September 2022	Not Available	Indefinite
19 (non-LTS)***	September 2022	March 2023	Not Available	Indefinite
20 (non-LTS)***	March 2023	September 2023	Not Available	Indefinite
21 (LTS)***	September 2023	September 2028	September 2031	Indefinite

CSDN @JavaGuide

Overview (selected part) :

- [JEP 321: HTTP Client Standardization](#)
- [JEP 333: ZGC \(Scalable Low-Latency Garbage Collector\)](#)
- [JEP 323: Local Variable Syntax for Lambda Parameters](#)
- [JEP 330: Launching Single-File Source Code Programs](#)

HTTP Client Standardization

Java 11 standardizes the Http Client API introduced in Java 9 and updated in Java 10. While incubating in the first two versions, Http Client was almost completely rewritten and now fully supports asynchronous non-blocking.



In addition, in Java 11, the package name of `HttpClient` `jdk.incubator.http` was changed from `java.net.http`, and this API `CompletableFuture` provides non-blocking request and response semantics through `.thenAccept()`. It is also very simple to use, as follows:

```
1  var request = HttpRequest.newBuilder()                                java
2      .uri(URI.create("https://javastack.cn"))
3      .GET()
4      .build();
5  var client = HttpClient.newHttpClient();
6
7  // 同步
8  HttpResponse<String> response = client.send(request,
9  HttpResponse.BodyHandlers.ofString());
10 System.out.println(response.body());
11
12 // 异步
13 client.sendAsync(request, HttpResponse.BodyHandlers.ofString())
14     .thenApply(HttpResponse::body)
15     .thenAccept(System.out::println);
```

String Enhancements

Java 11 adds a series of string processing methods:

```
1  //判断字符串是否为空
2  " ".isBlank();//true
3  //去除字符串首尾空格
4  " Java ".strip();// "Java"
5  //去除字符串首部空格
6  " Java ".stripLeading(); // "Java "
7  //去除字符串尾部空格
8  " Java ".stripTrailing(); // " Java"
9  //重复字符串多少次
10 "Java".repeat(3); // "JavaJavaJava"
11 //返回由行终止符分隔的字符串集合。
12 "A\nB\nC".lines().count(); // 3
13 "A\nB\nC".lines().collect(Collectors.toList());
```



Optional Enhancement

Added a new `isEmpty()` method to determine whether the specified `Optional` object is empty.

```
1 var op = Optional.empty();  
2 System.out.println(op.isEmpty()); //判断指定的 Optional 对象是否为空
```

java

ZGC (Scalable Low-Latency Garbage Collector)

ZGC, or Z Garbage Collector, is a scalable, low-latency garbage collector.

ZGC is designed to meet the following goals:

- GC pause time does not exceed 10ms
- It can handle small heaps of a few hundred MB as well as large heaps of several TB.
- Application throughput will not drop by more than 15% (compared to the G1 recycling algorithm)
- It is convenient to introduce new GC features and lay the foundation for optimization using colored needles and load barriers.
- Currently only supports Linux/x64 platforms

ZGC is currently **in the experimental stage** and only supports Linux/x64 platforms.

Similar to ParNew and G1 in CMS, ZGC also uses the mark-copy algorithm, but ZGC has made significant improvements to the algorithm.

There will be fewer Stop The World situations in ZGC!

For details, please see: ["Exploration and Practice of the New Generation Garbage Collector ZGC"](#)

Local variable syntax for lambda parameters

Starting with Java 10, a key feature called local variable type inference was introduced. 

Type inference allows you to use the keyword `var` as the type of a local variable instead of the actual type, and the compiler infers the type based on the value assigned to the

variable.

There are several restrictions on the var keyword in Java 10

- Can only be used on local variables
- Must be initialized when declared
- Cannot be used as method parameters
- Cannot be used in Lambda expressions

Java 11 began to allow developers to use var for parameter declaration in Lambda expressions.


```
1 // 下面两者是等价的 java
2 Consumer<String> consumer = (var i) -> System.out.println(i);
3 Consumer<String> consumer = (String i) -> System.out.println(i);
```

Start a single-file source code program

This means we can run Java source code from a single file. This feature allows Java source code to be executed directly using the Java interpreter. The source code is compiled in memory and then executed by the interpreter, eliminating the need to generate .class files on disk. The only constraint is that all related classes must be defined in the same Java file.

It is particularly useful for Java beginners who want to try simple programs, and can be used together with jshell. It can certainly enhance the ability to write scripts in Java.

Other new features

- **New garbage collector Epsilon** : A completely passive GC implementation that allocates limited memory resources to minimize memory usage and memory throughput latency
- **Low-overhead Heap Profiling** : Java 11 provides a low-overhead Java heap allocation sampling method that can obtain heap-allocated Java object information and access heap information through JVMTI
- **TLS 1.3 protocol** : Java 11 includes an implementation of the Transport Layer Security (TLS) 1.3 specification (RFC 8446), replacing TLS included in previous versions, including TLS 1.2. It also improves other TLS features, such as the OCSP stapling 

extension (RFC 6066, RFC 6961), and the session hash and extended master key extension (RFC 7627), and has made many improvements in security and performance.

- **Java Flight Recorder** : Flight Recorder was previously a profiling tool in the commercial JDK, but in Java 11, its code was included in the public code base so that everyone can use the feature.
- ...

refer to

- JDK 11 Release Notes: <https://www.oracle.com/java/technologies/javase/11-relnote-issues.html>
- Java 11 – Features and Comparison: <https://www.geeksforgeeks.org/java-11-features-and-comparison/>

JavaGuide官方公众号 (微信搜索JavaGuide)



- 1、公众号后台回复“PDF”获取原创PDF面试手册
- 2、公众号后台回复“学习路线”获取Java学习路线最新版
- 3、公众号后台回复“开源”获取优质Java开源项目合集
- 4、公众号后台回复“八股文”获取Java面试真题+面经

Recently Updated 2025/2/15 12:55

Contributors: guide , Guide , Mr.Hope , Alex , Clear

Copyright © 2025 Guide

