INNOQ Technology Lunch / 08.11.2023

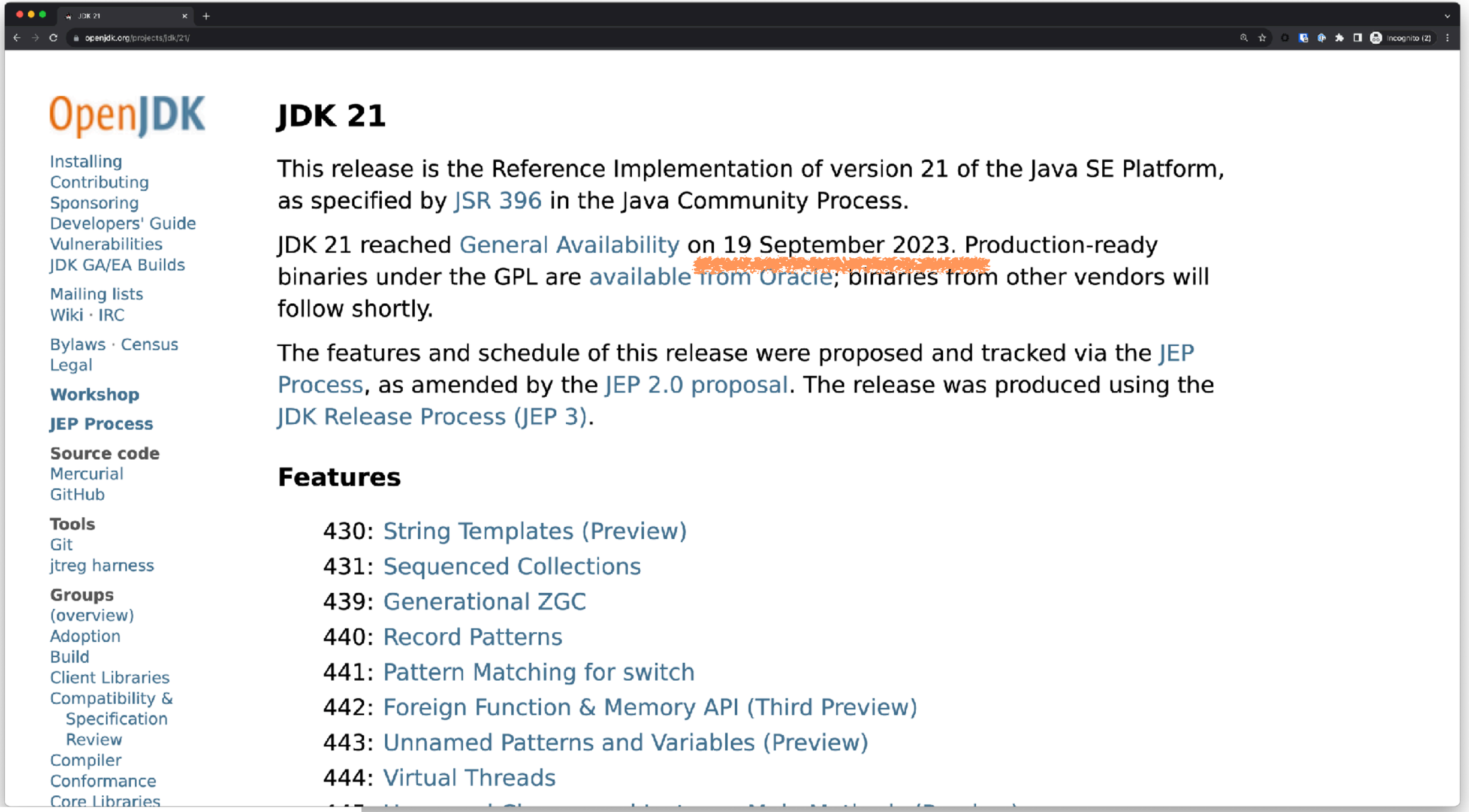
Das erwartet Dich mit JDK 21!

INNOQ

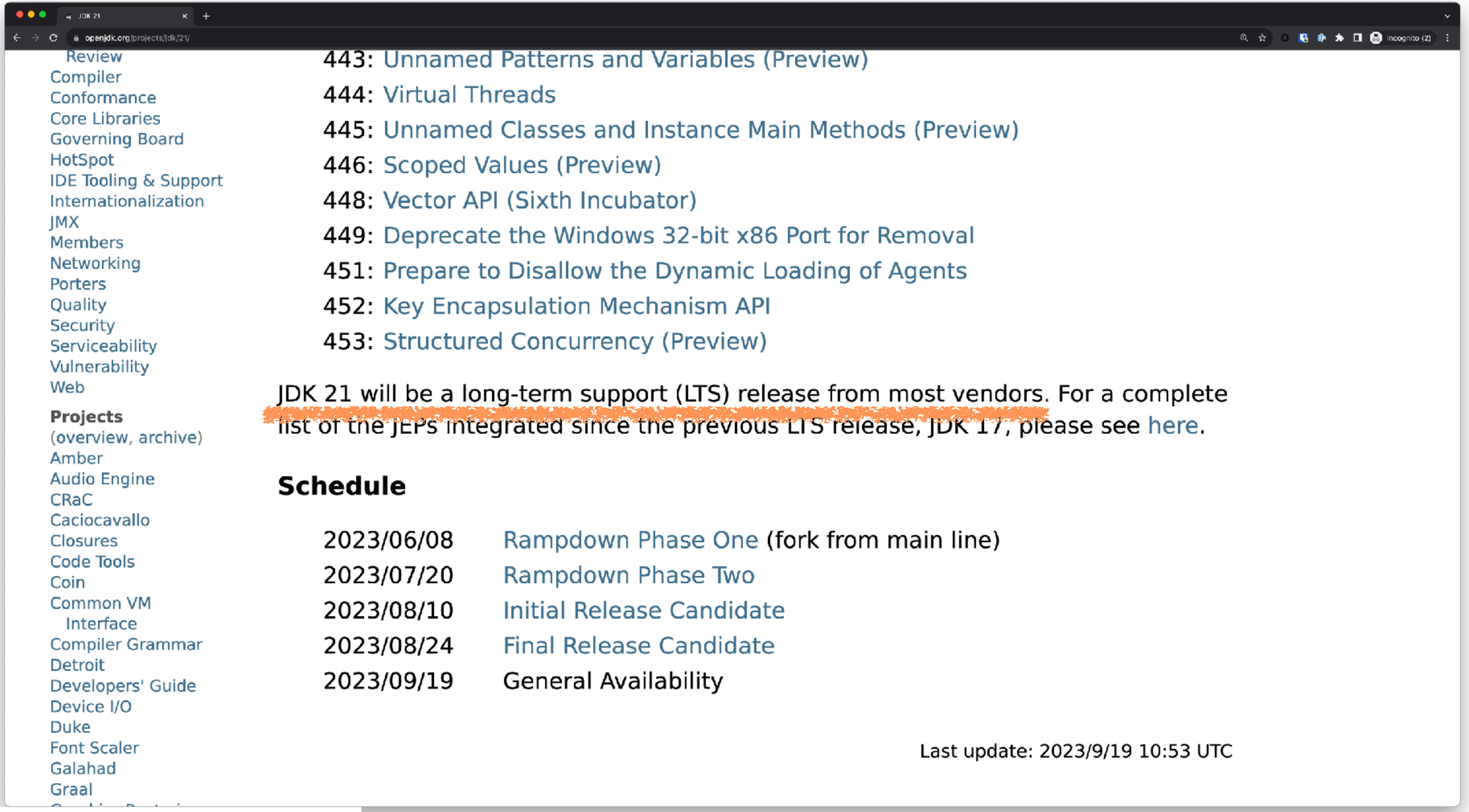
MICHAEL VITZ
SENIOR CONSULTANT

MICHAEL VITZ

Java Champion
Senior Consultant at INNOQ

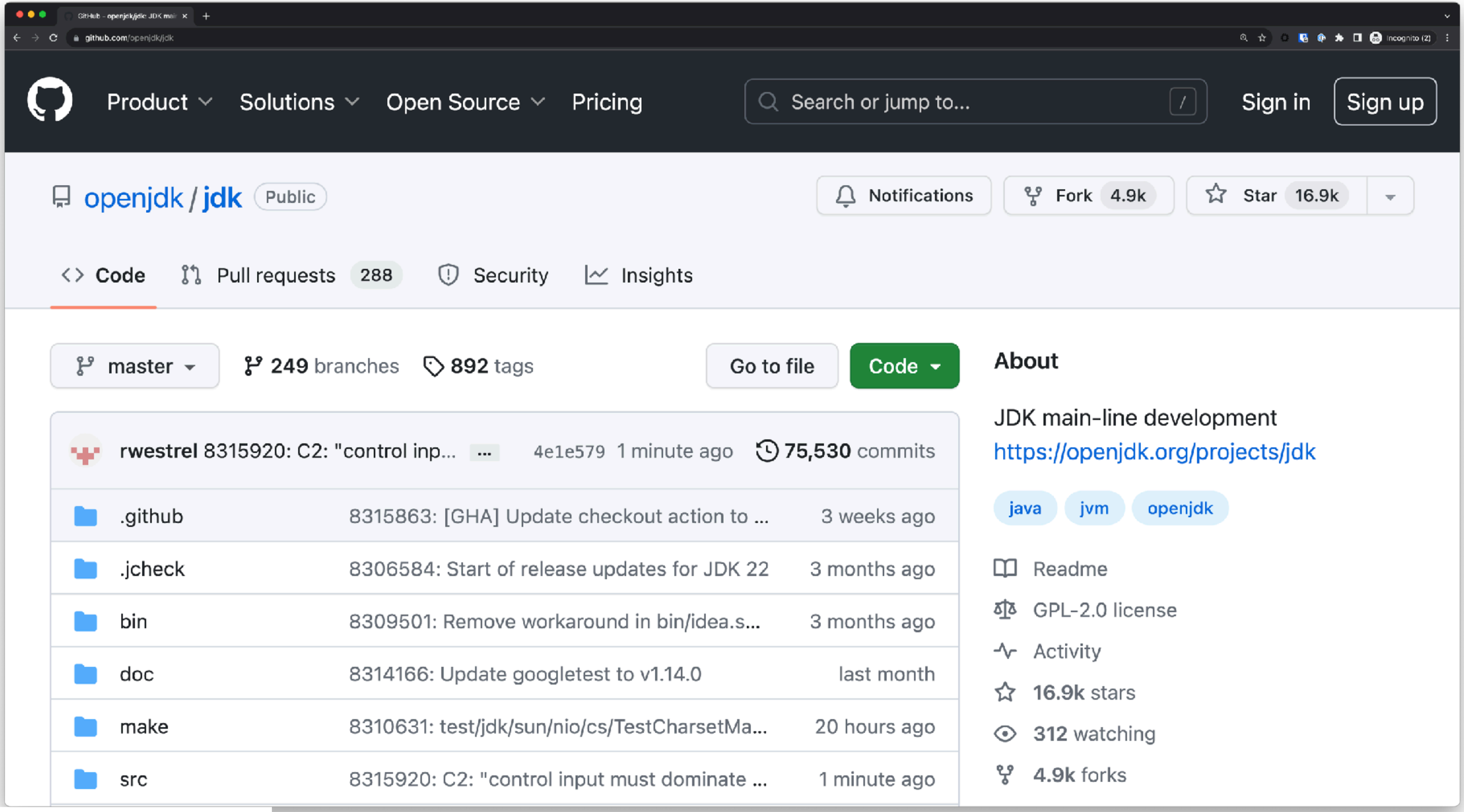


https://openjdk.java.net/projects/jdk/21/

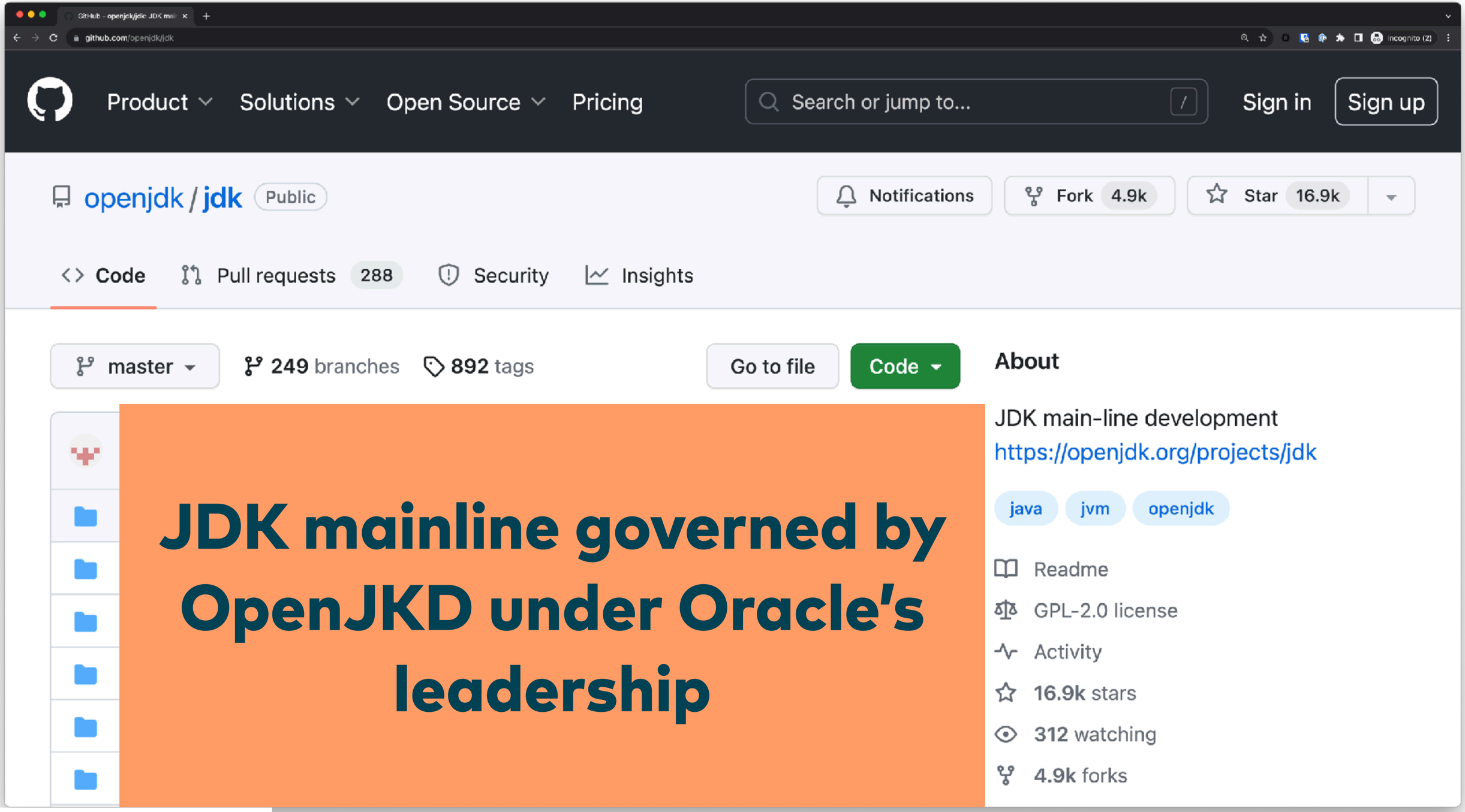


https://openjdk.java.net/projects/jdk/21/

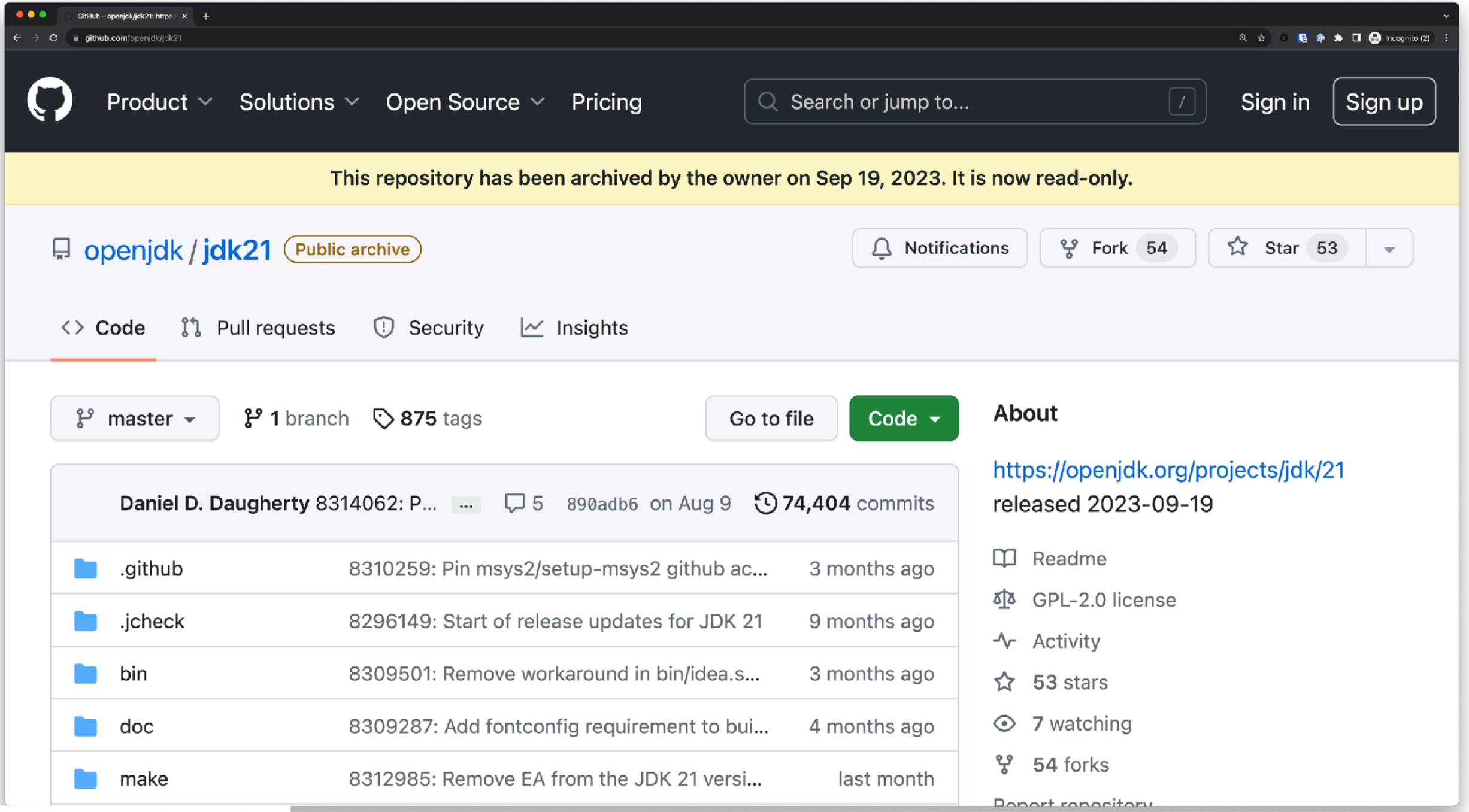
LTS



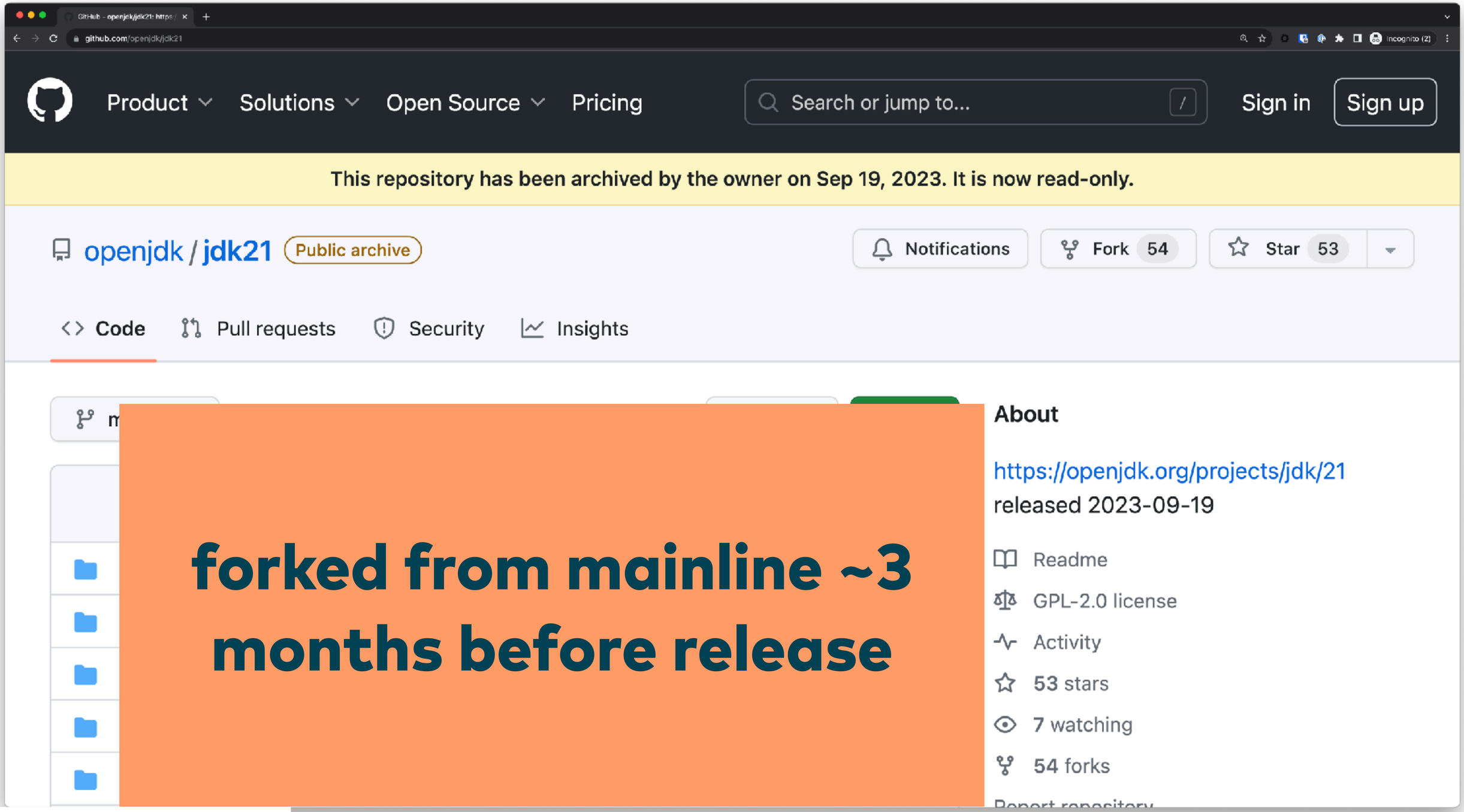
https://github.com/openjdk/jdk



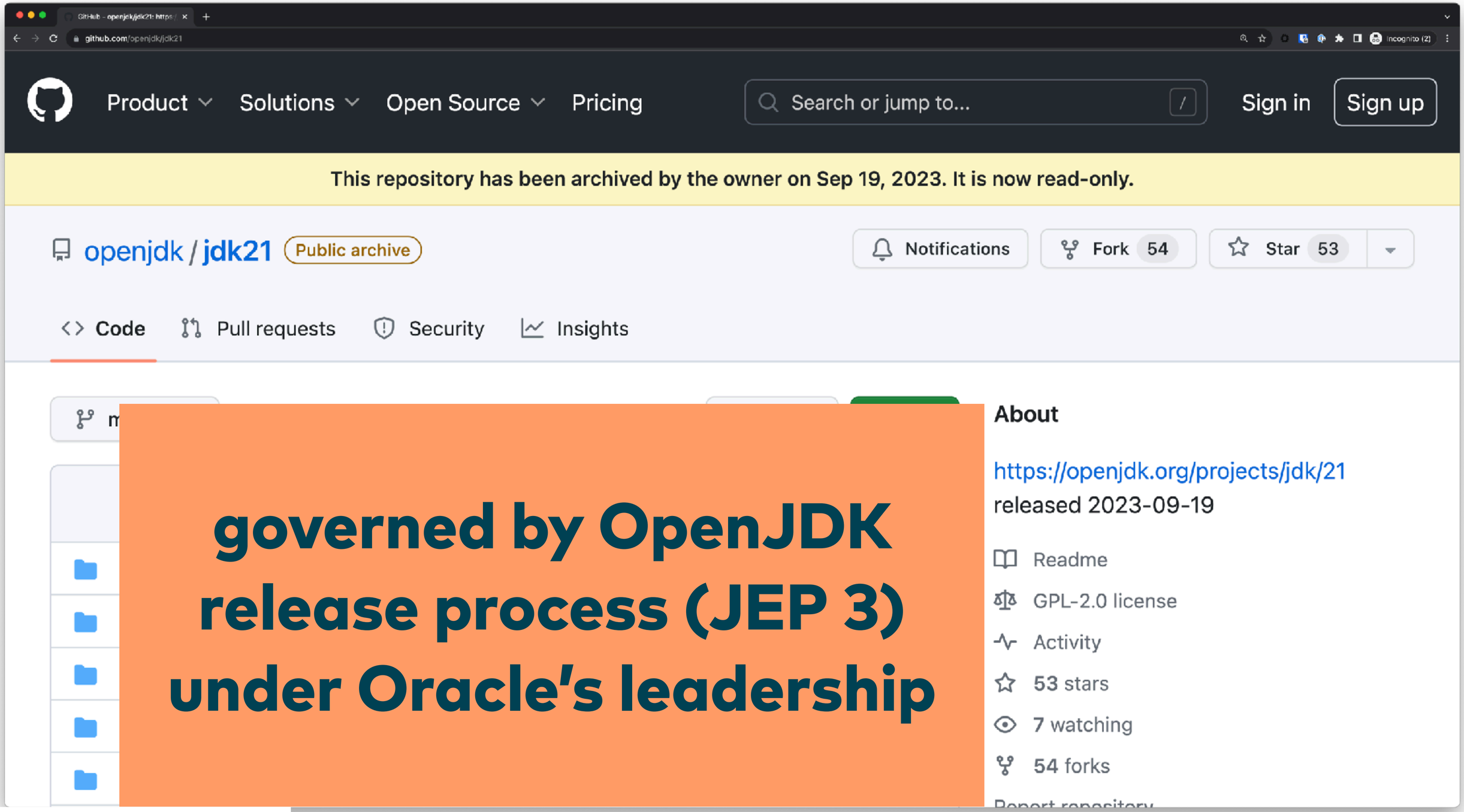
https://github.com/openjdk/jdk



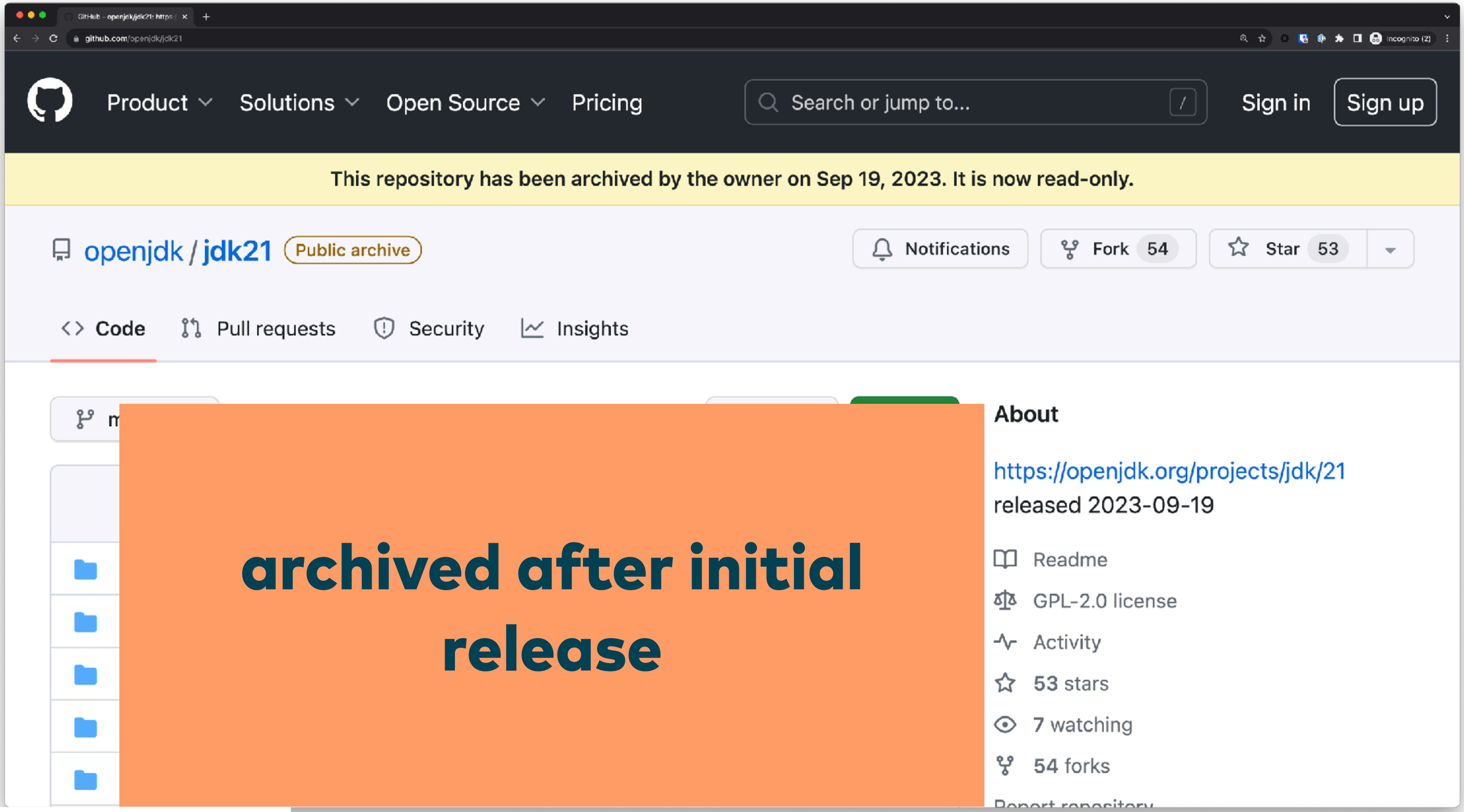
https://github.com/openjdk/jdk21



https://github.com/openjdk/jdk21

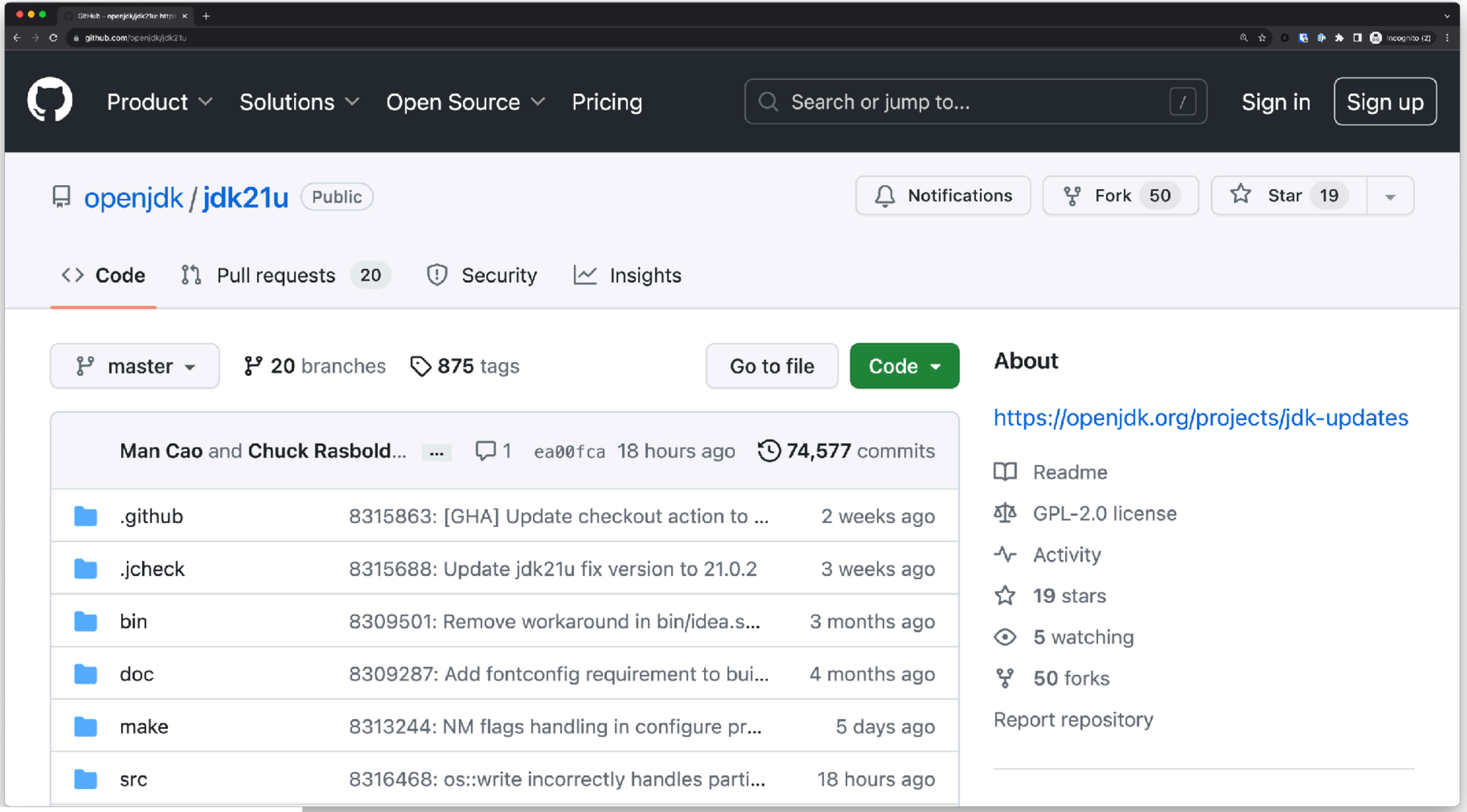


https://github.com/openjdk/jdk21

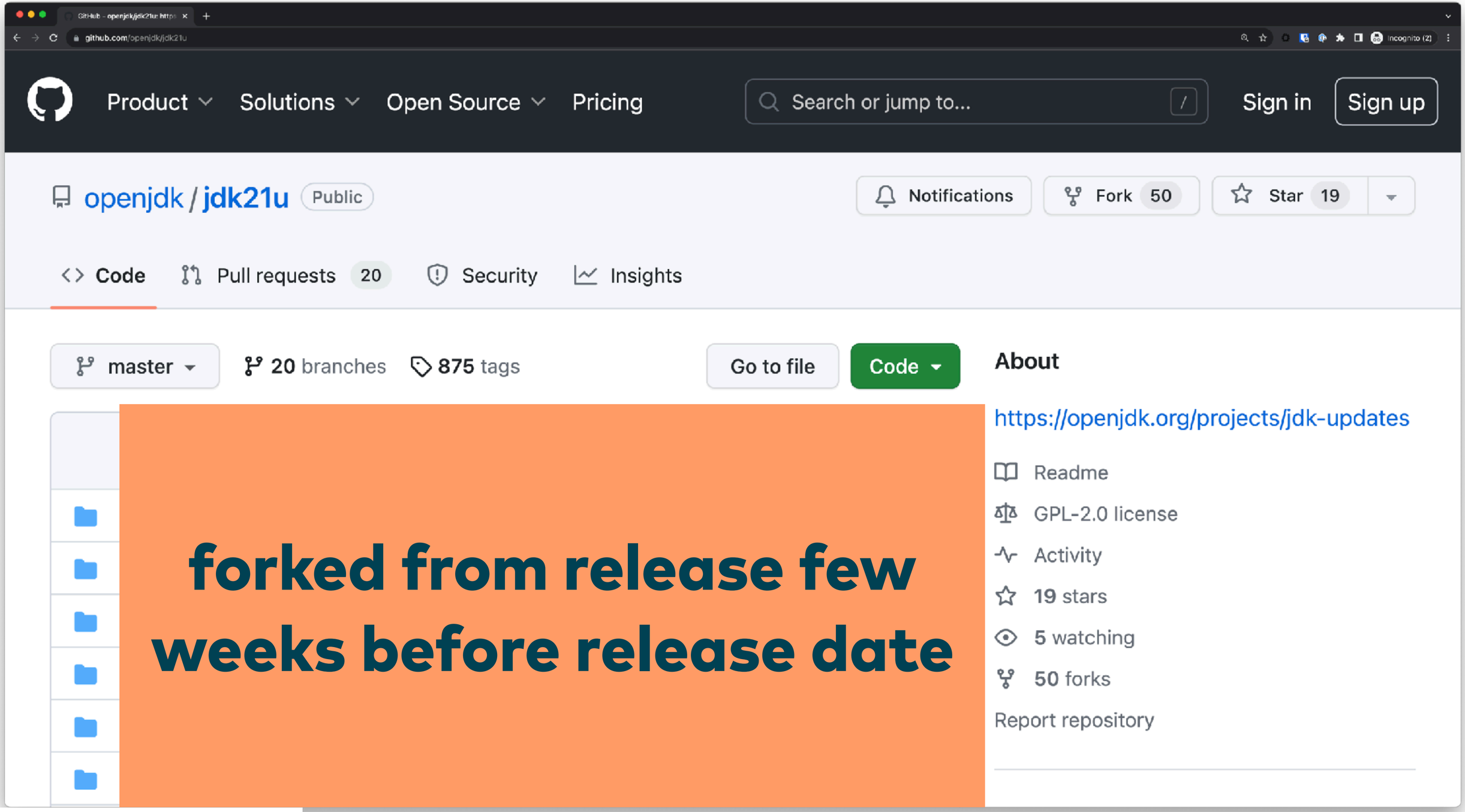


archived after initial release

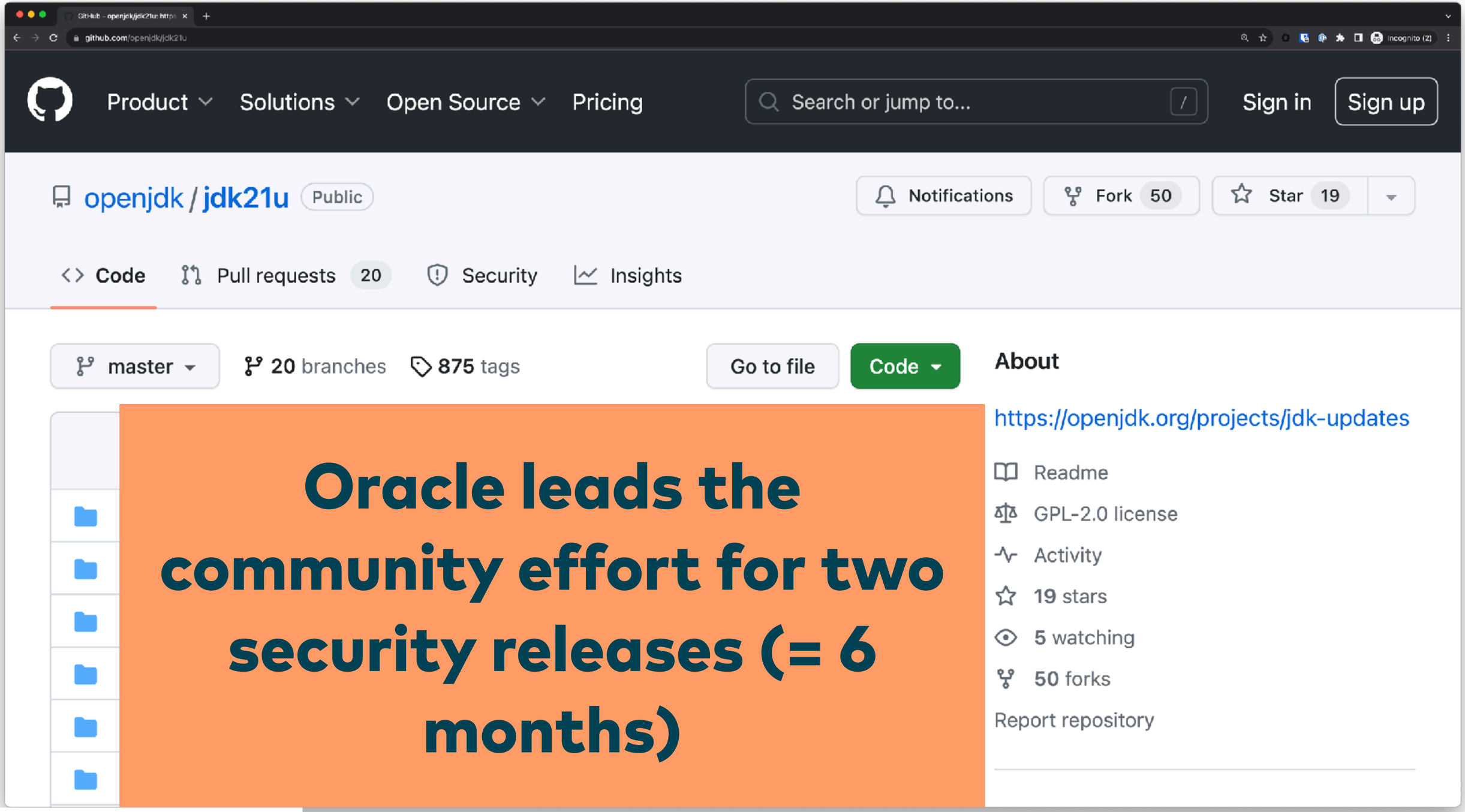
https://github.com/openjdk/jdk21



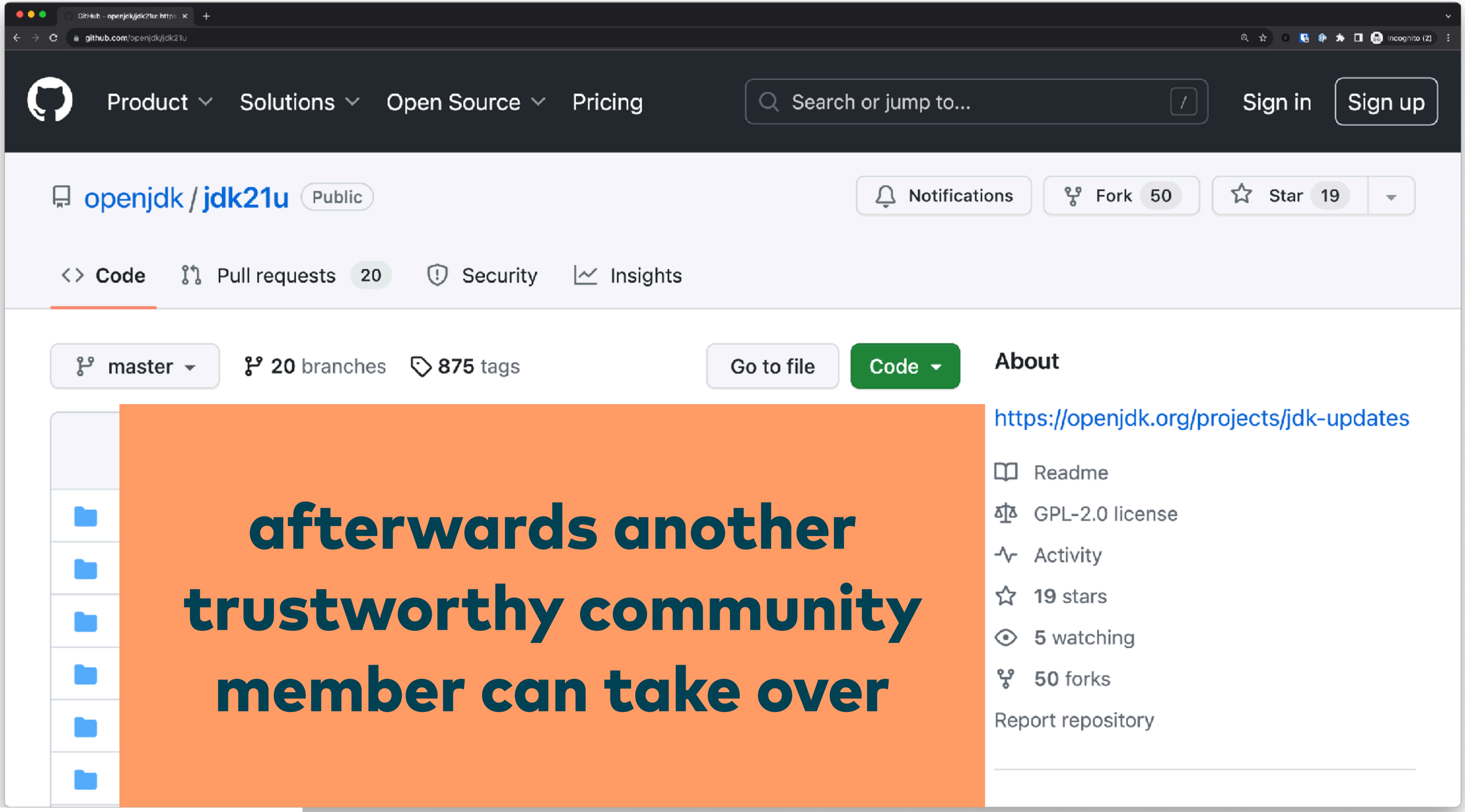
https://github.com/openjdk/jdk21u



https://github.com/openjdk/jdk21u



https://github.com/openjdk/jdk21u



https://github.com/openjdk/jdk21u

Java

Java

Follow:

Java Technology

Moving the JDK to a Two Year LTS Cadence

September 14, 2021 | 2 minute read

Donald Smith
Vice President of Product Management

Executive Summary:

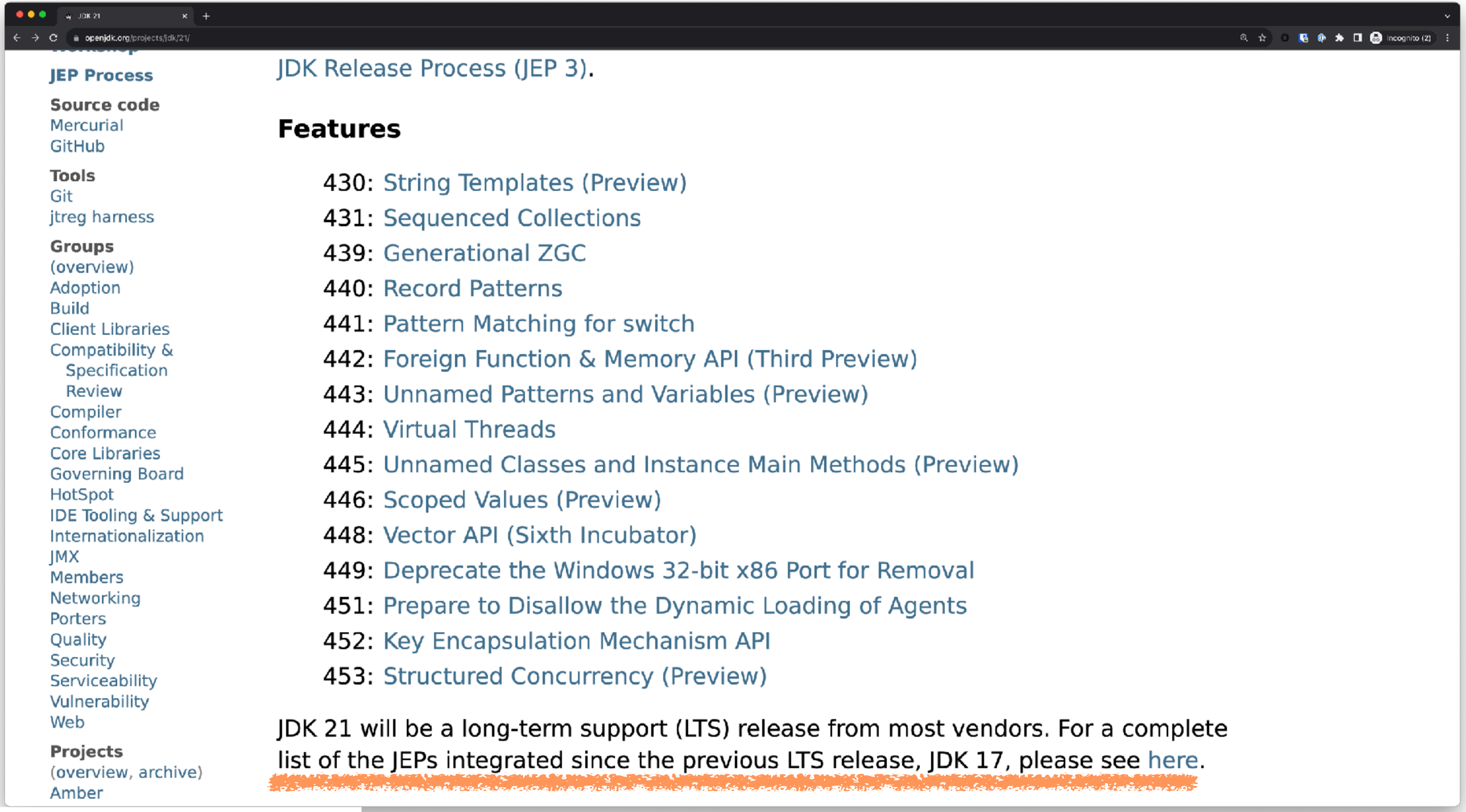
- Oracle proposes to shift the cadence of JDK LTS releases from every three years to every two years. If accepted, this means that the next JDK LTS release after JDK 17 will JDK 21, rather than JDK 23.
- Duration of the Oracle JDK LTS support remains unchanged, starting at a minimum of eight years for each LTS release.

Article

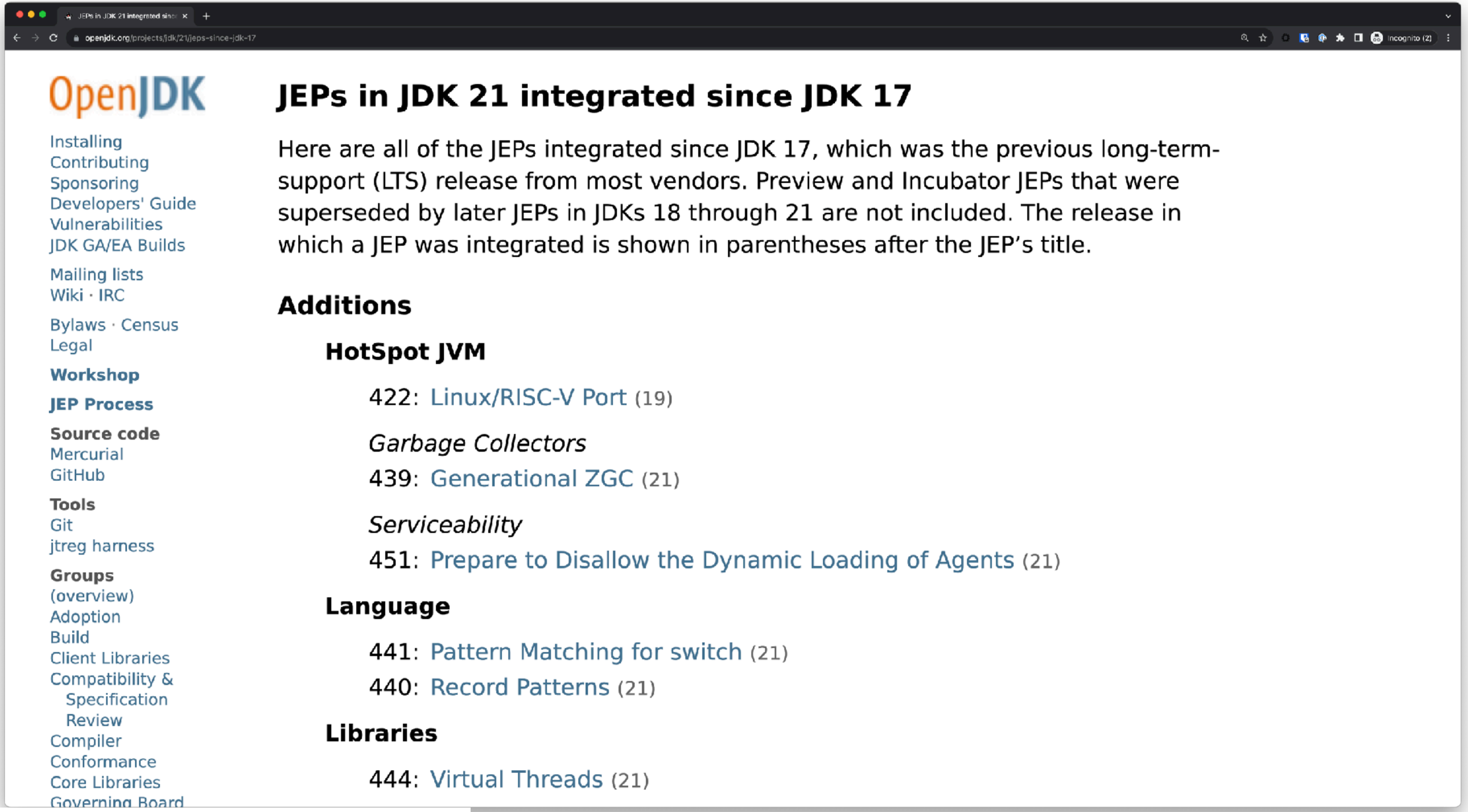
The release of JDK 17 is a perfect time to reflect on the release cadence introduced several years ago and make any course corrections for the future. Developers and organizations have had time to adjust from the "massive change every few years" model to the more

https://blogs.oracle.com/java/post/moving-the-jdk-to-a-two-year-lts-cadence

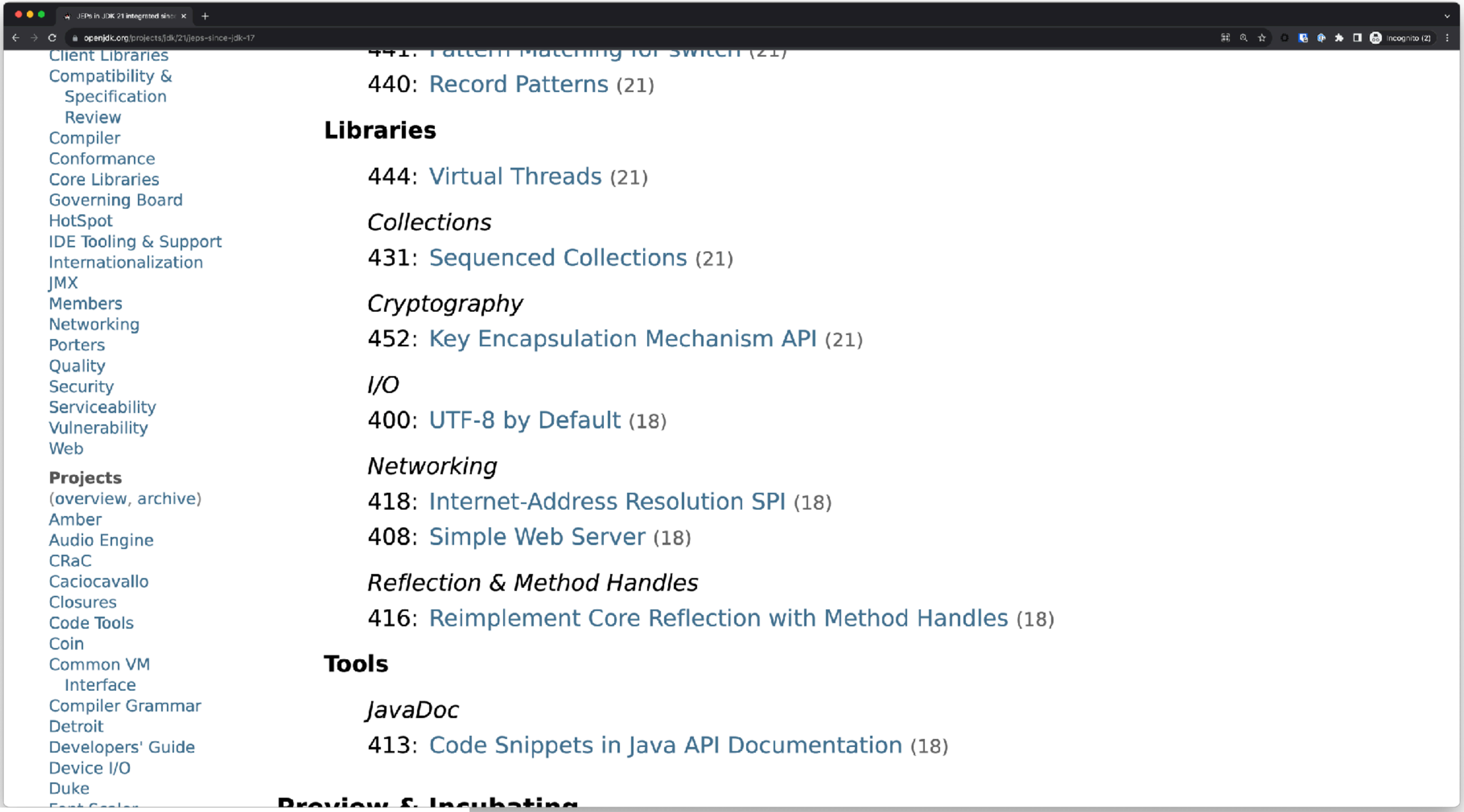
JDK 21



https://openjdk.java.net/projects/jdk/21/



https://openjdk.org/projects/jdk/21/jeps-since-jdk-17



https://openjdk.org/projects/jdk/21/jeps-since-jdk-17

416: Reimplement Core Reflection with Method Handles (18)

Tools

JavaDoc

413: Code Snippets in Java API Documentation (18)

Preview & Incubating

Language

430: String Templates (Preview) (21)

445: Unnamed Classes and Instance Main Methods (Preview) (21)

443: Unnamed Patterns and Variables (Preview) (21)

Libraries

442: Foreign Function & Memory API (Third Preview) (21)

446: Scoped Values (Preview) (21)

453: Structured Concurrency (Preview) (21)

448: Vector API (Sixth Incubator) (21)

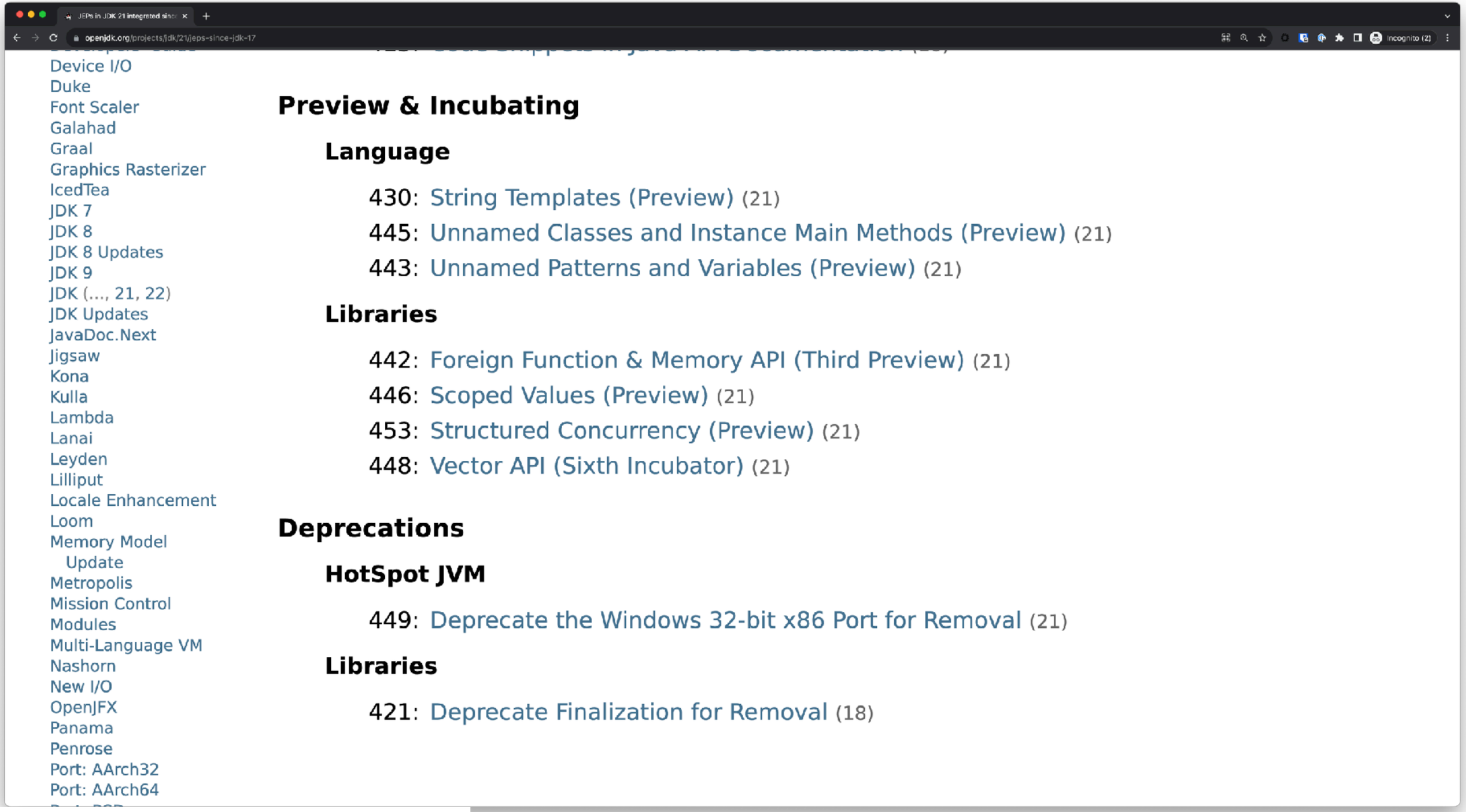
Deprecations

HotSpot JVM

449: Deprecate the Windows 32-bit x86 Port for Removal (21)

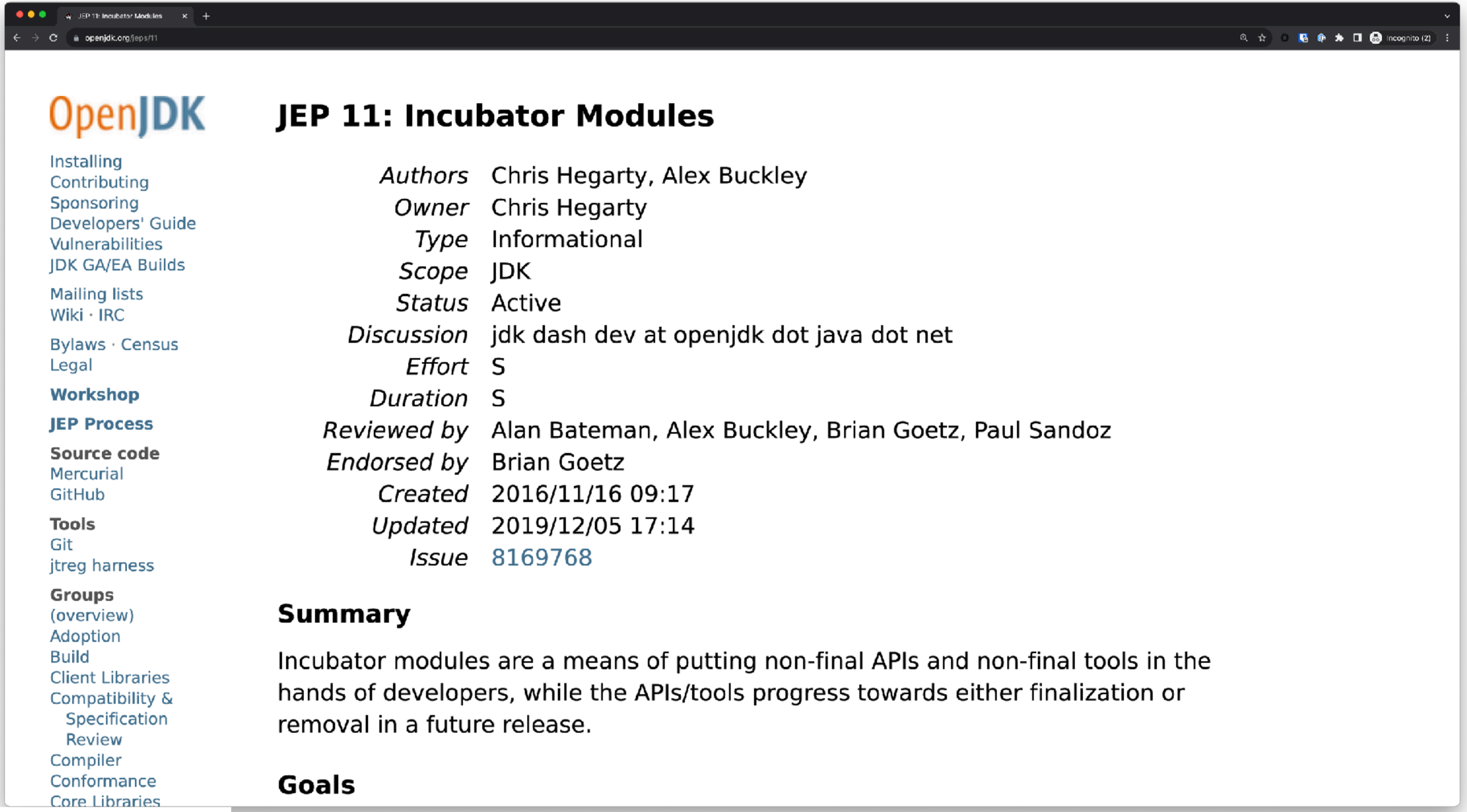
Libraries

https://openjdk.org/projects/jdk/21/jeps-since-jdk-17

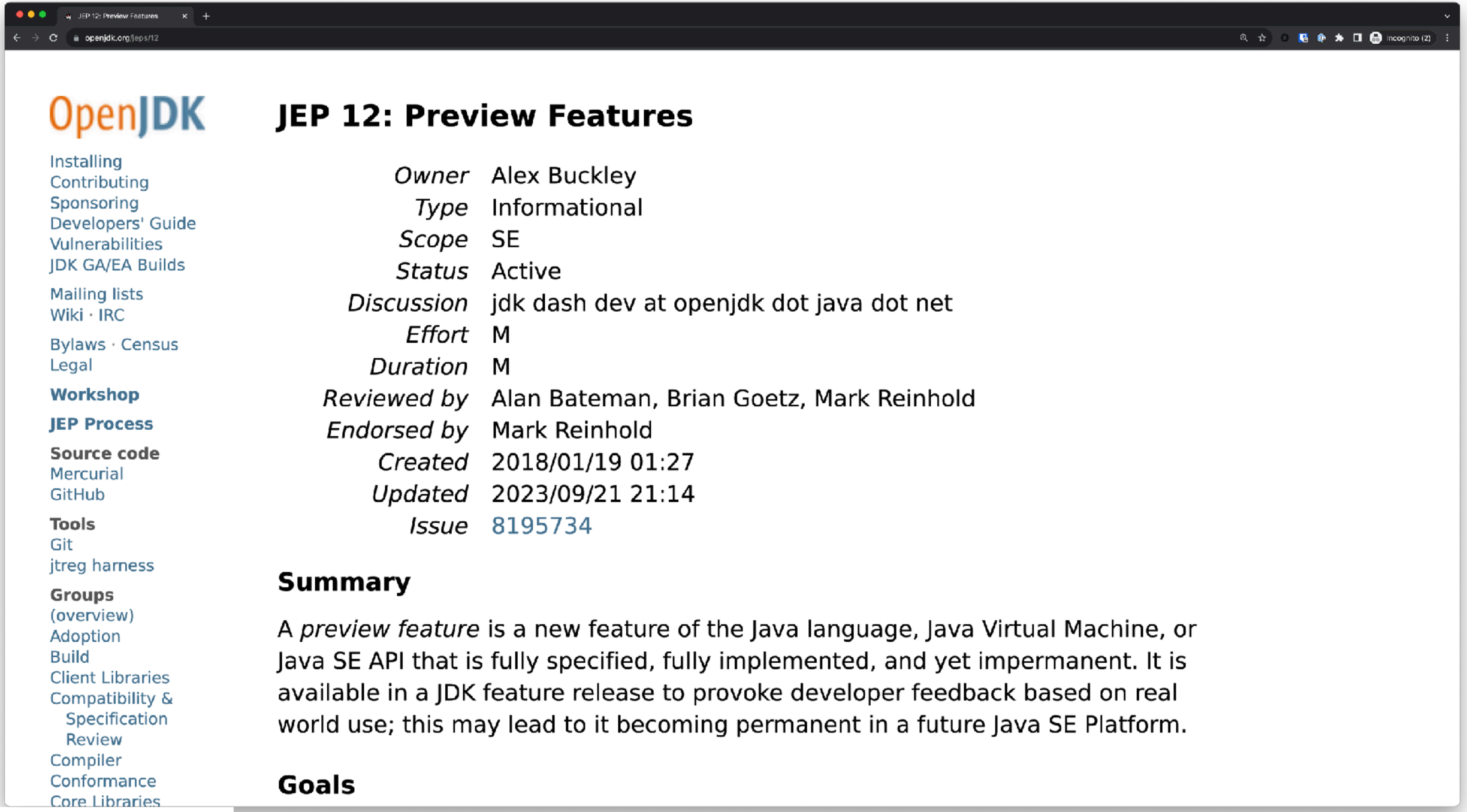


https://openjdk.org/projects/jdk/21/jeps-since-jdk-17

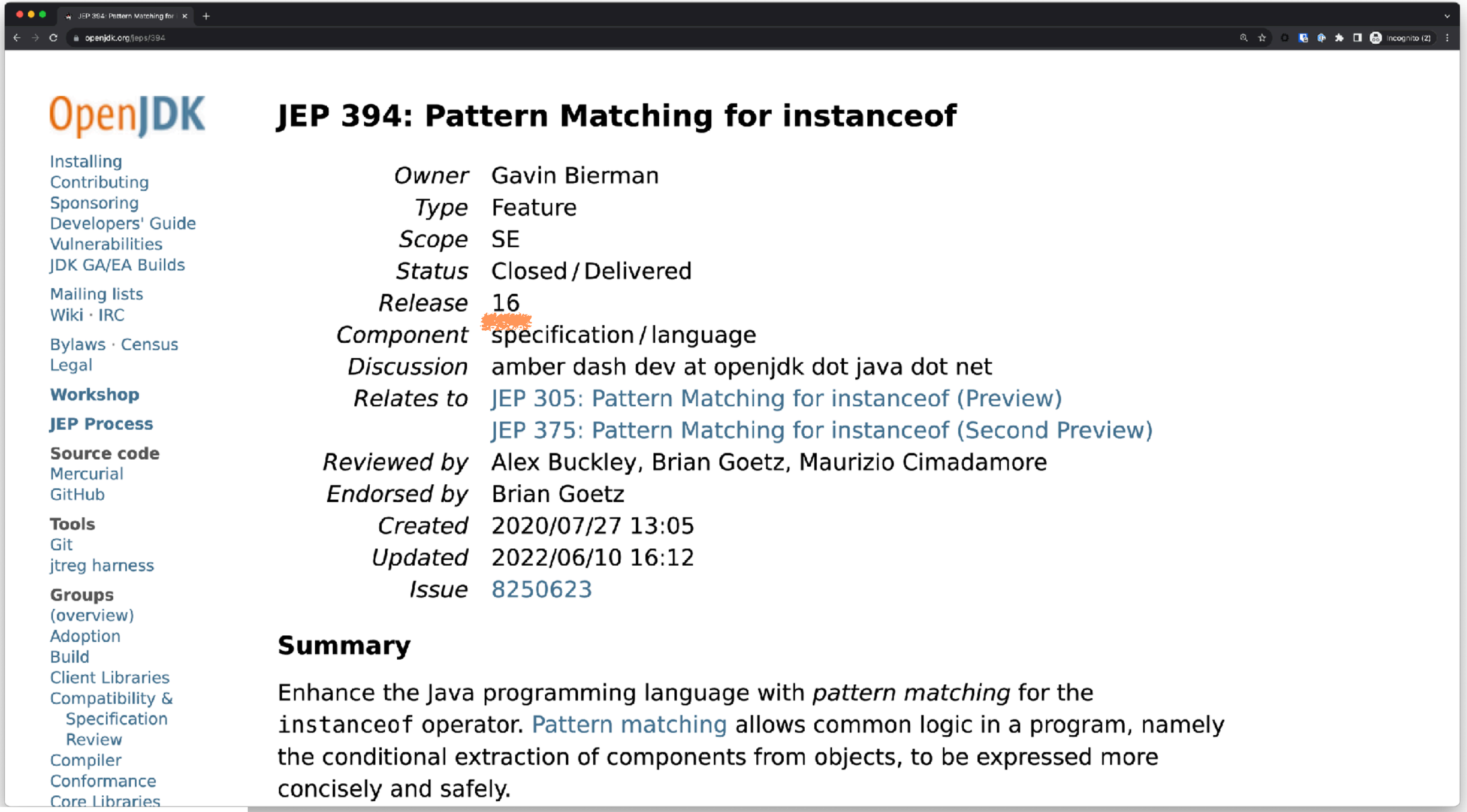
Features



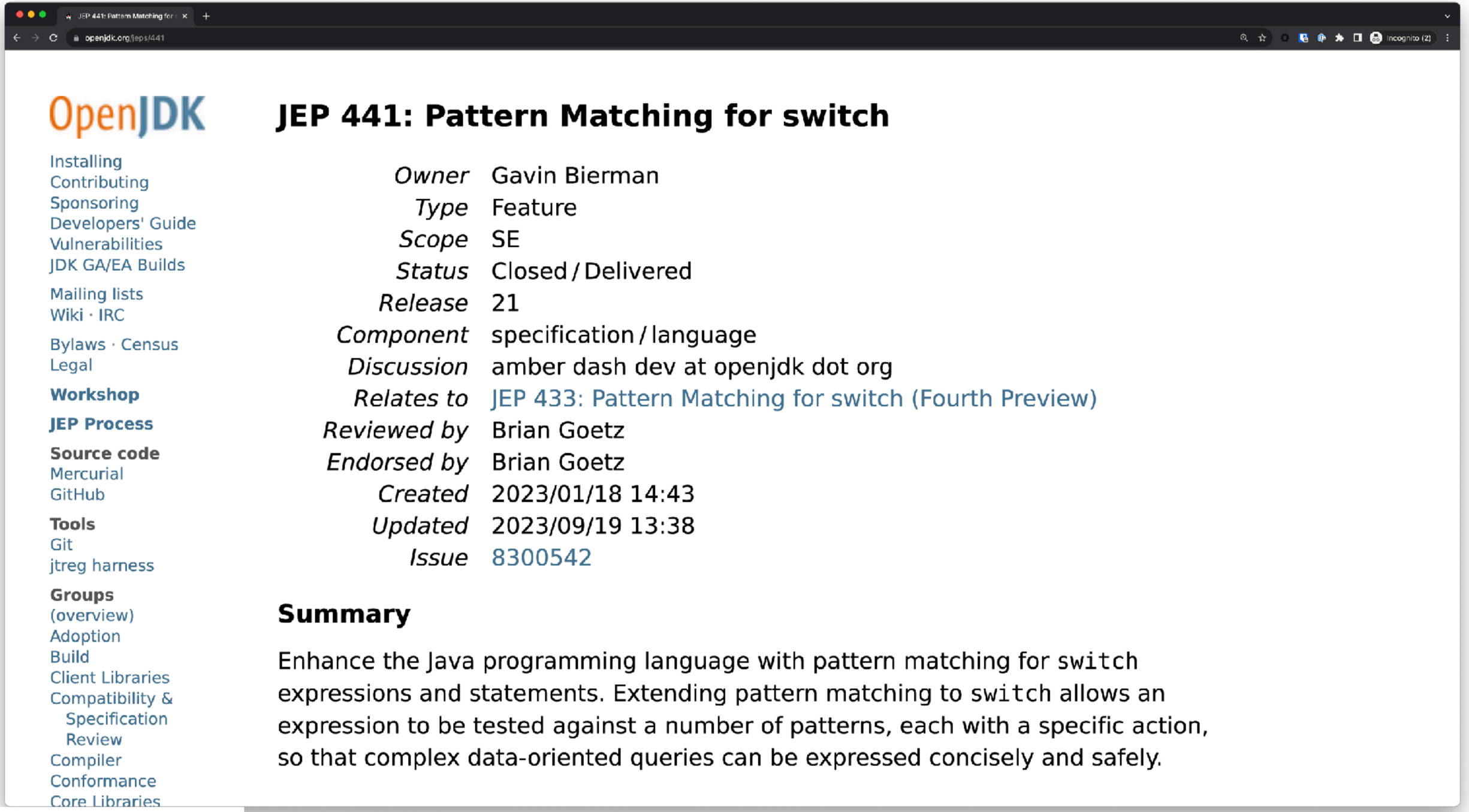
https://openjdk.org/jeps/11



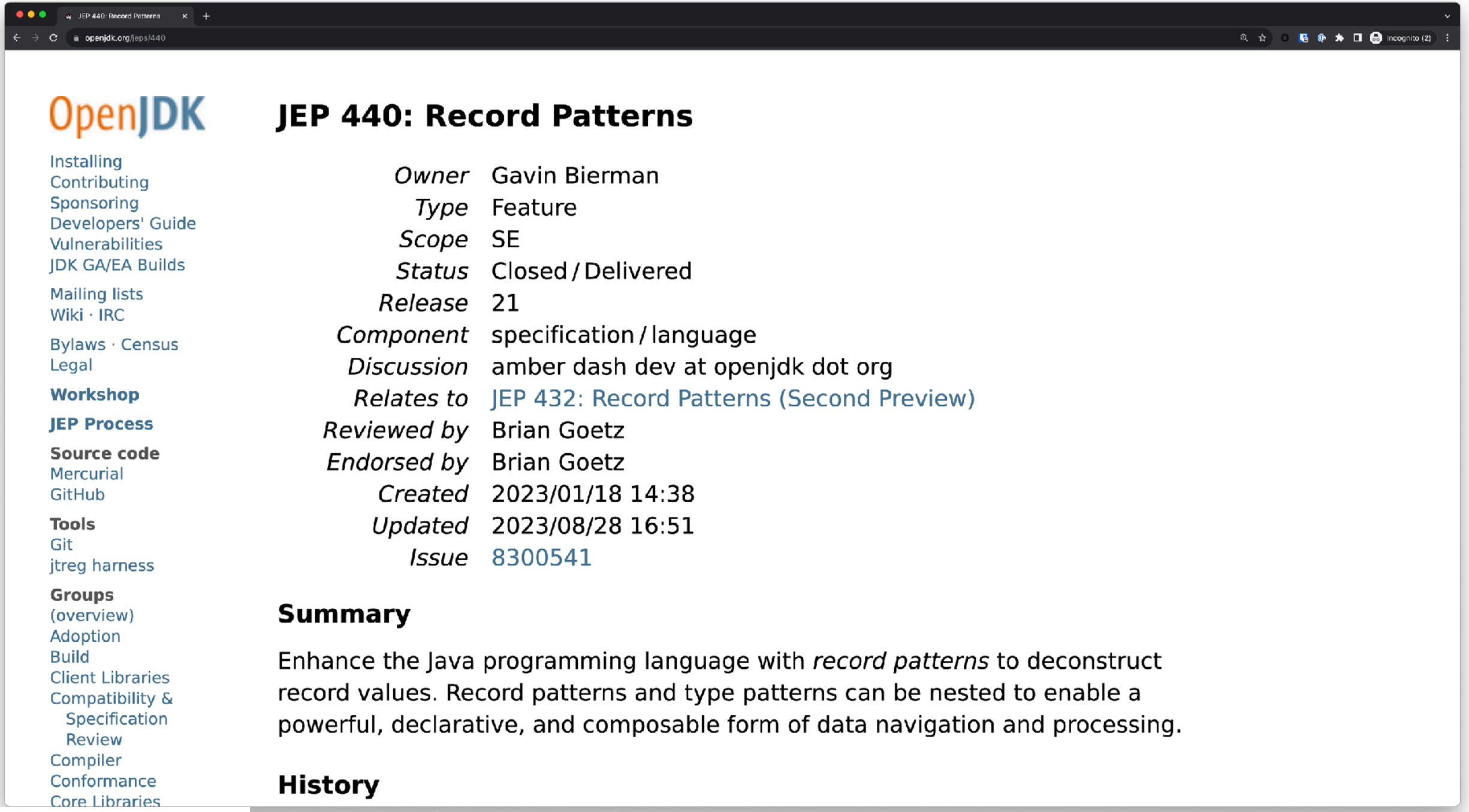
https://openjdk.org/jeps/12



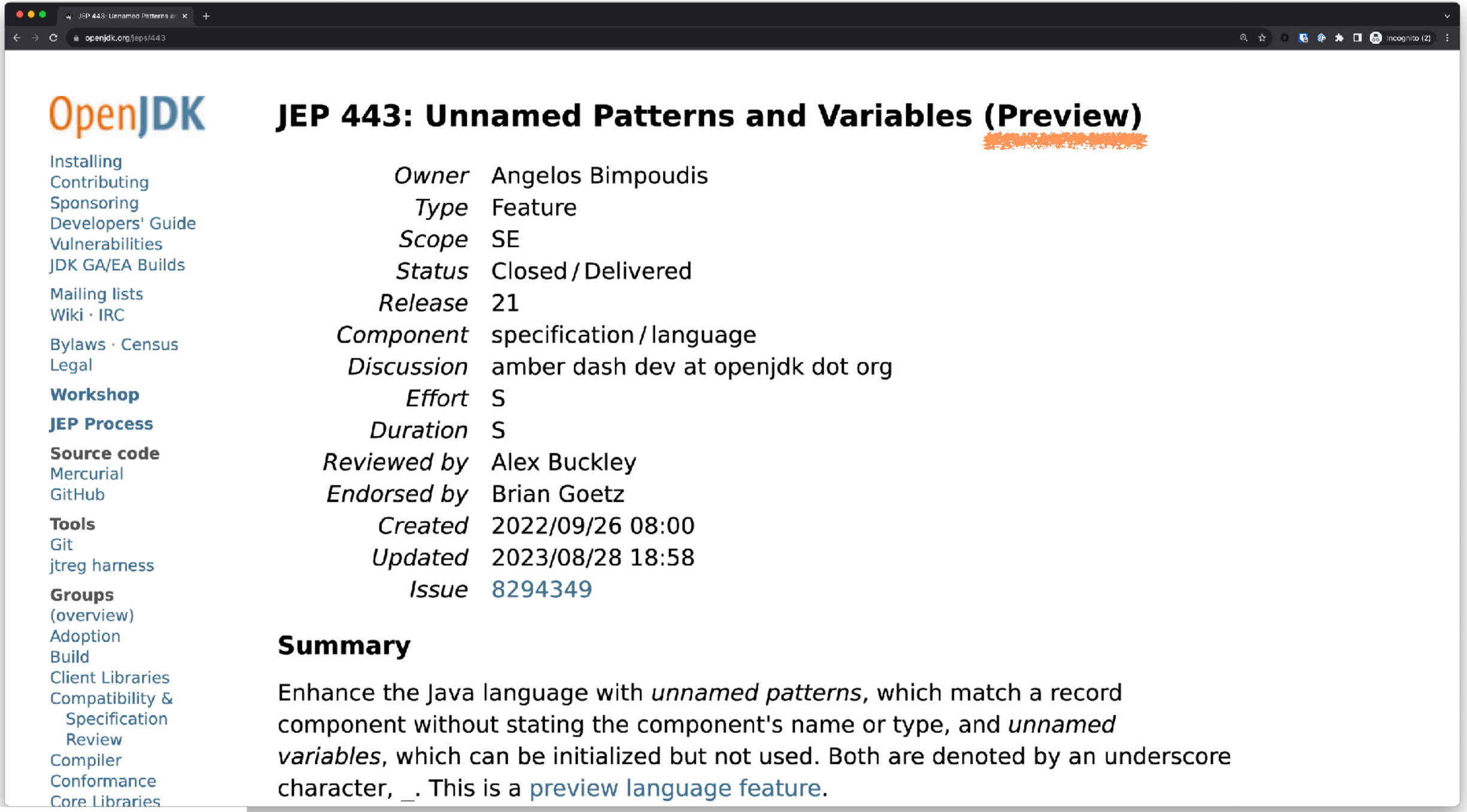
https://openjdk.org/jeps/394



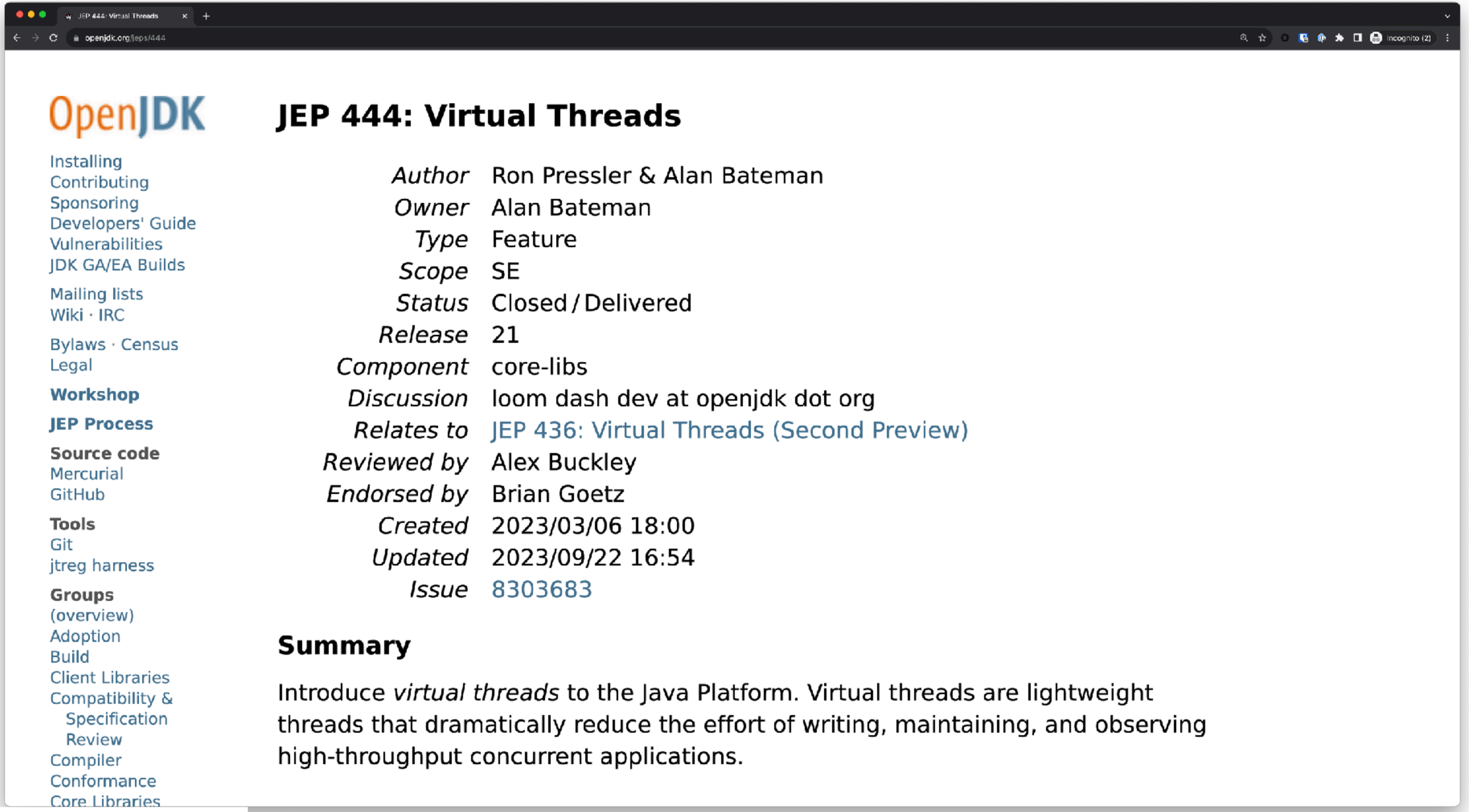
https://openjdk.org/jeps/441



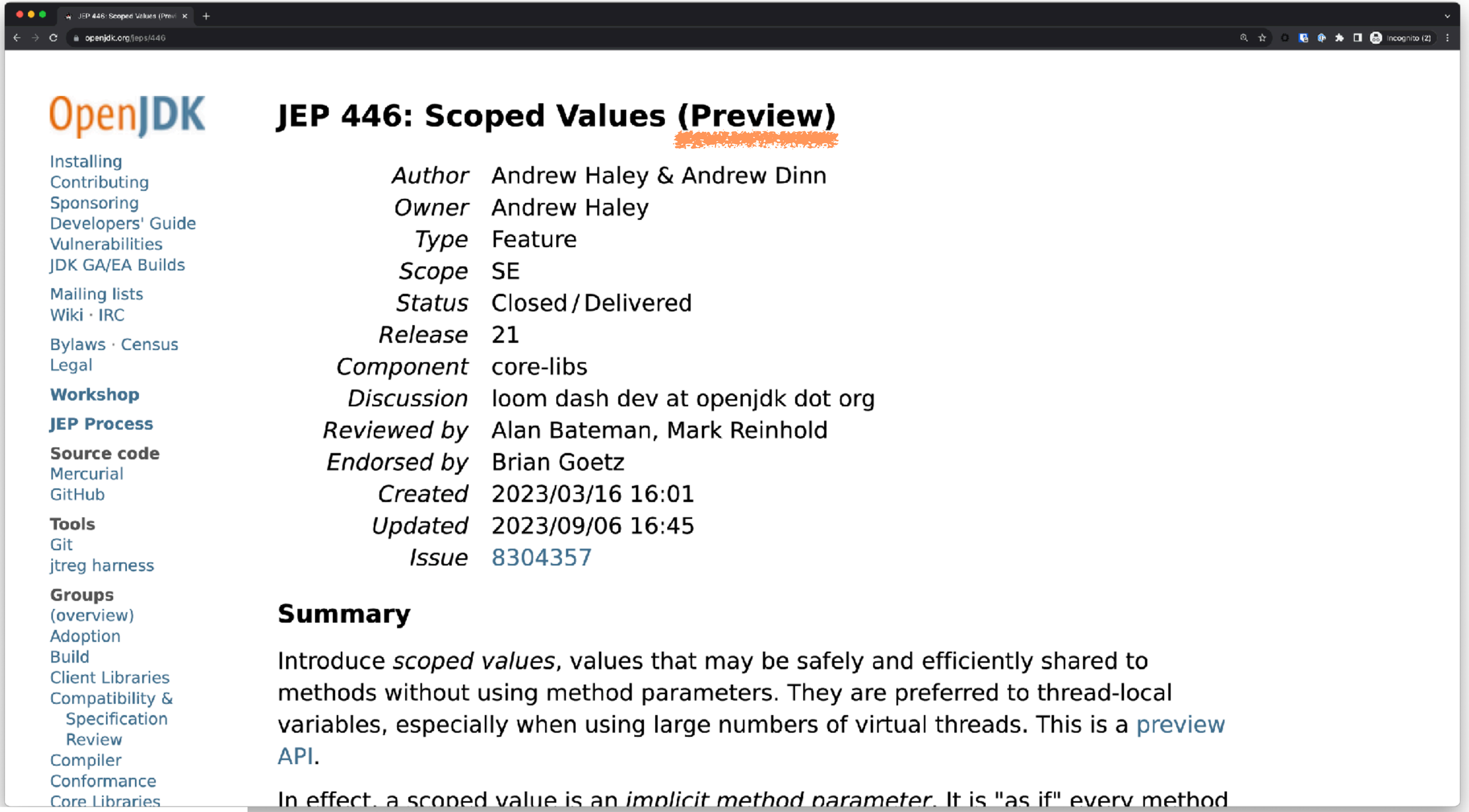
https://openjdk.org/jeps/440



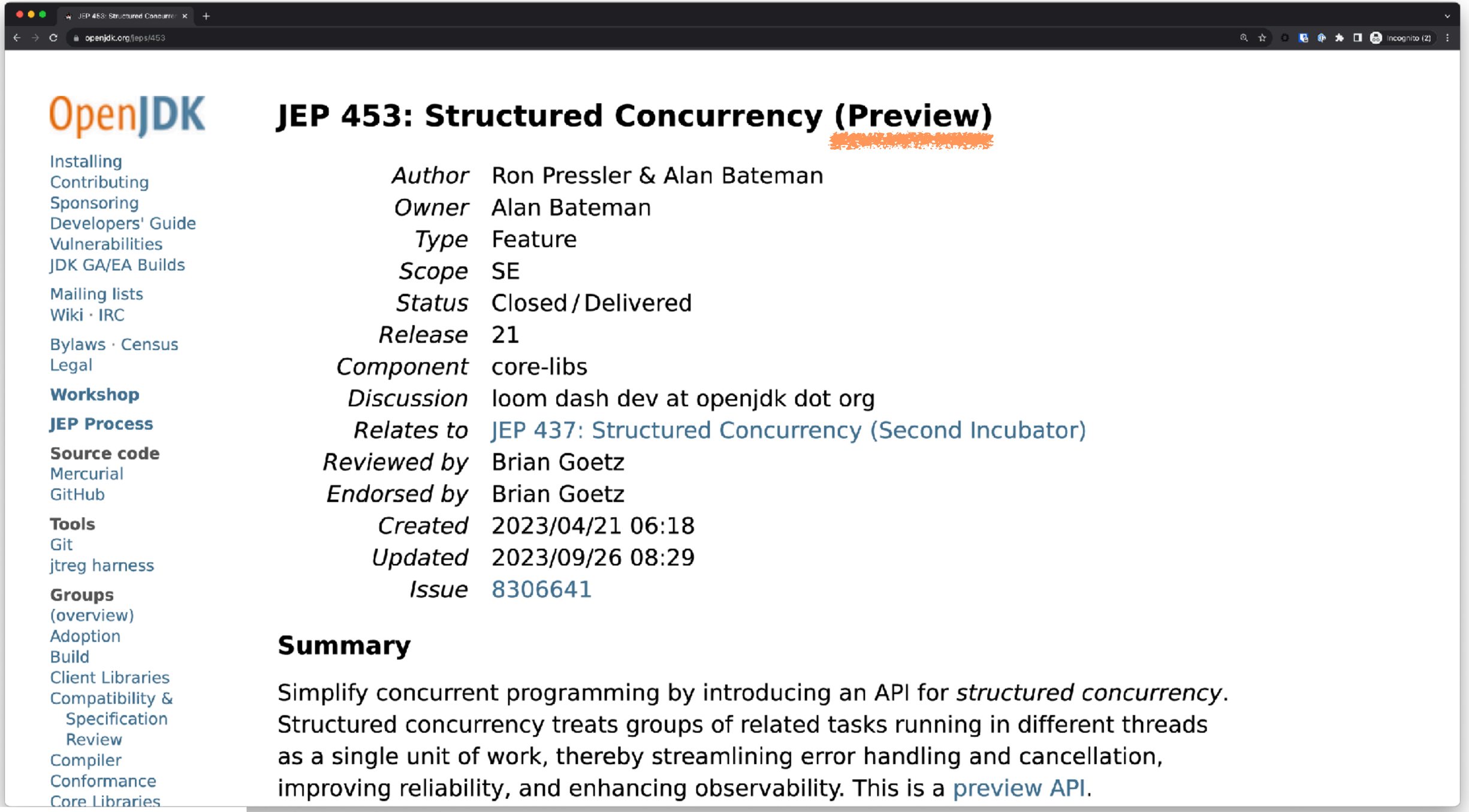
https://openjdk.org/jeps/443



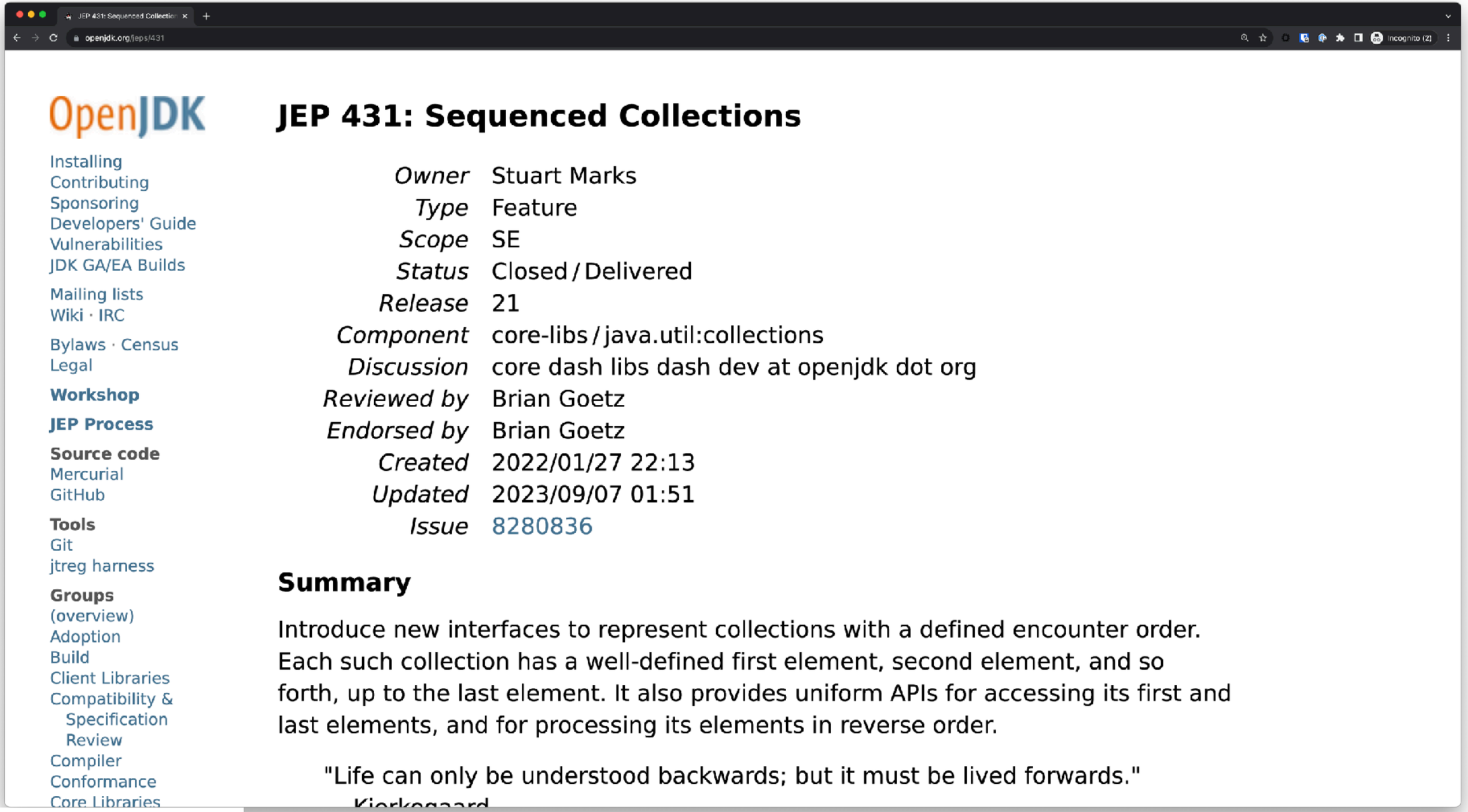
https://openjdk.org/jeps/444



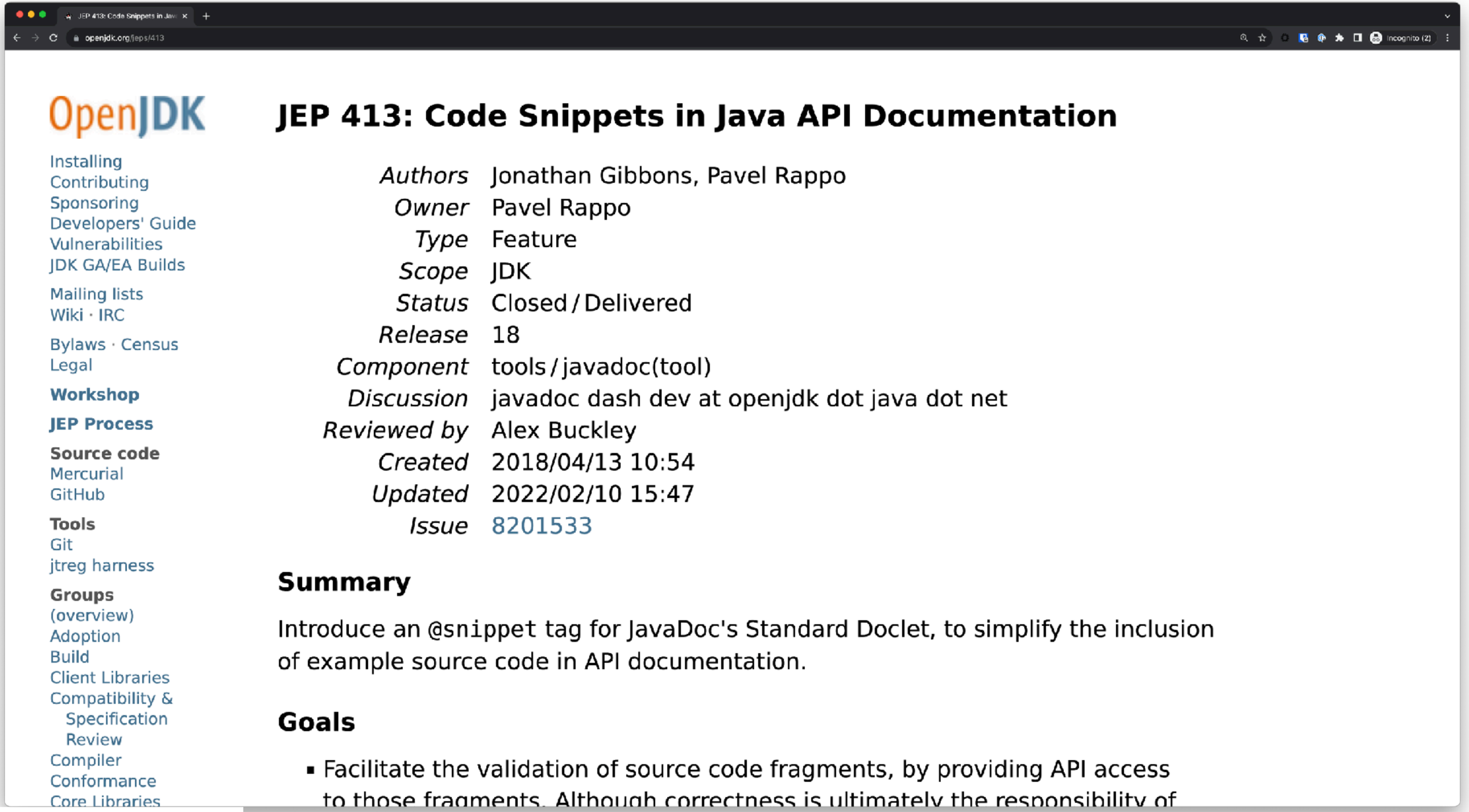
https://openjdk.org/jeps/446



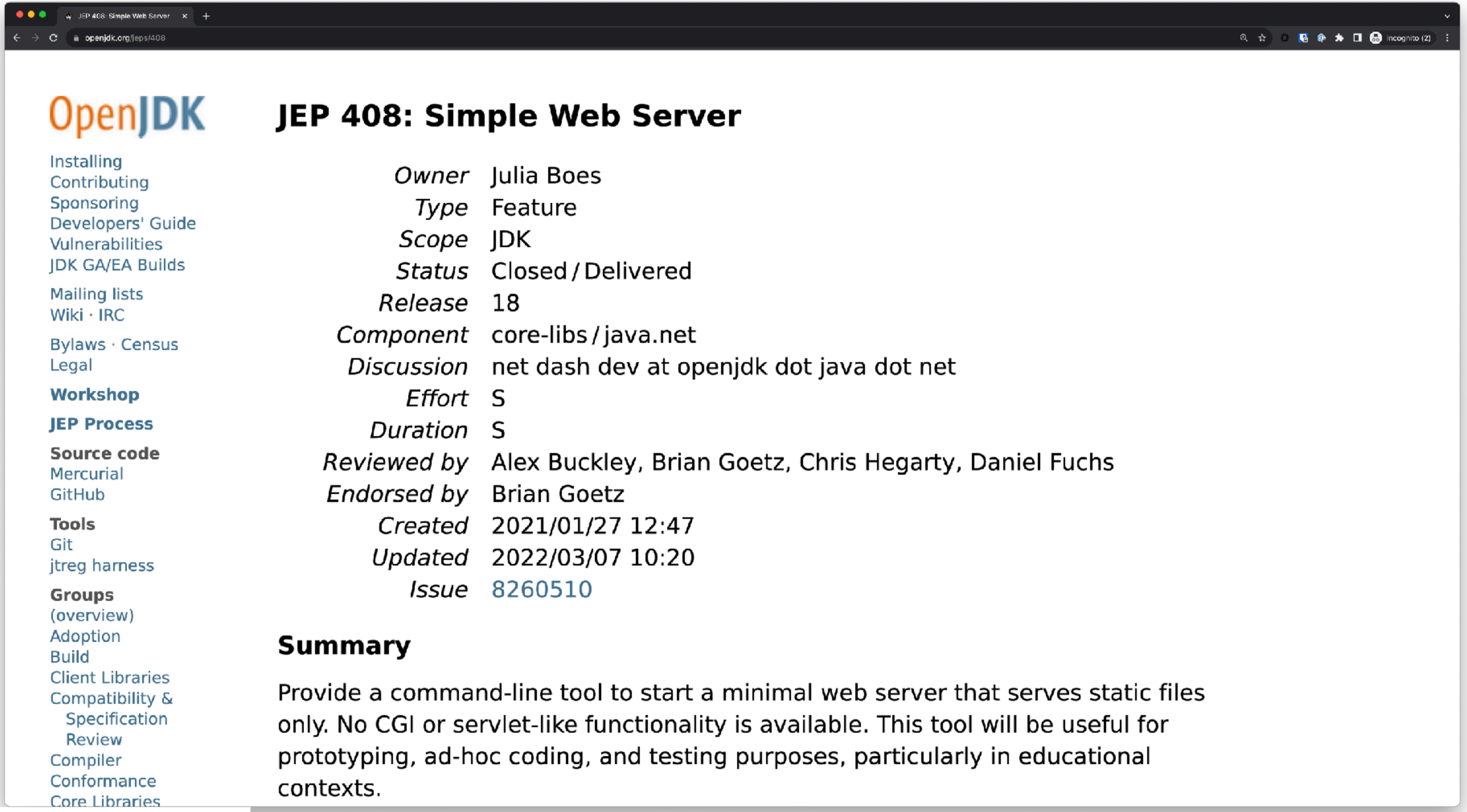
https://openjdk.org/jeps/453



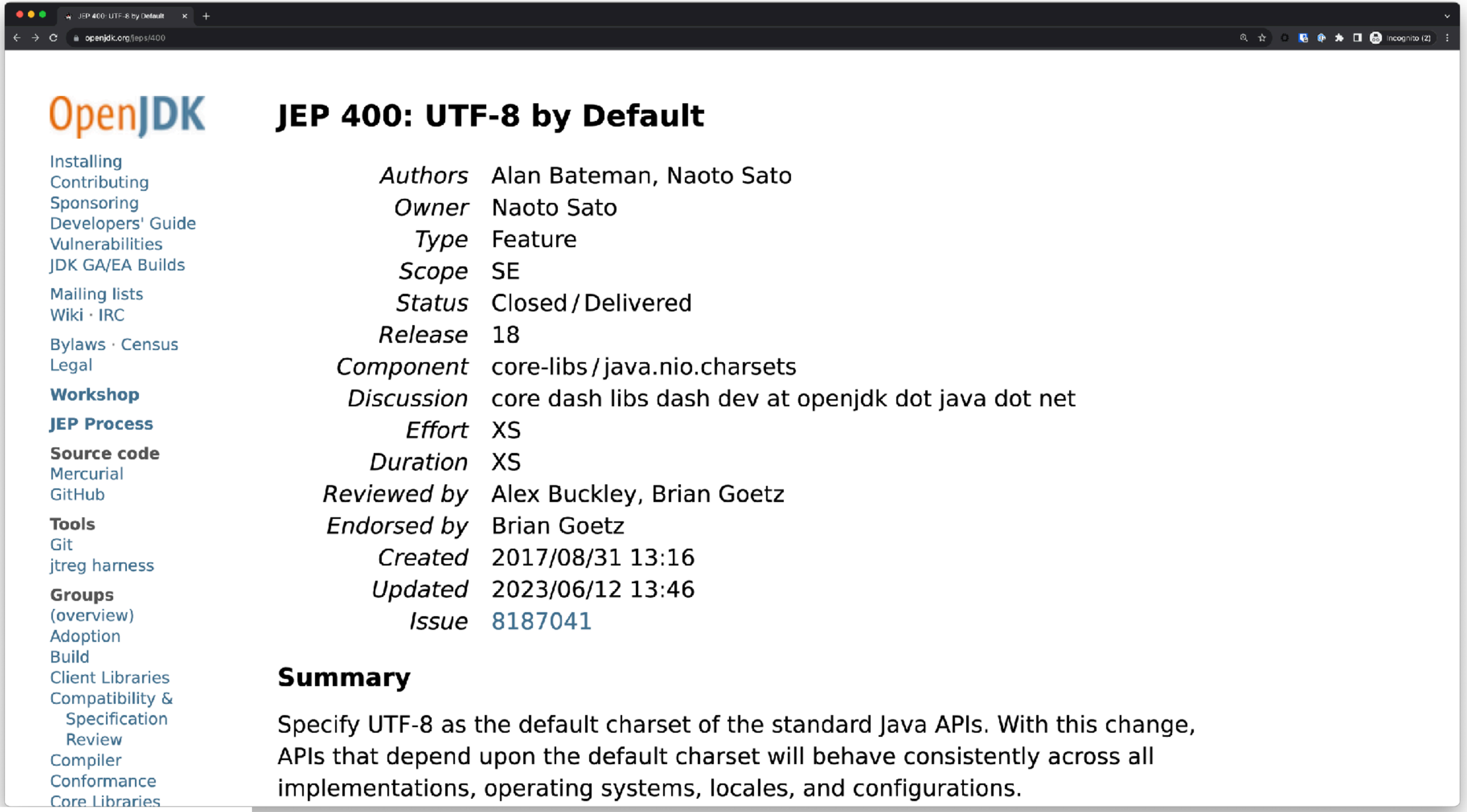
https://openjdk.org/jeps/431



https://openjdk.org/jeps/413

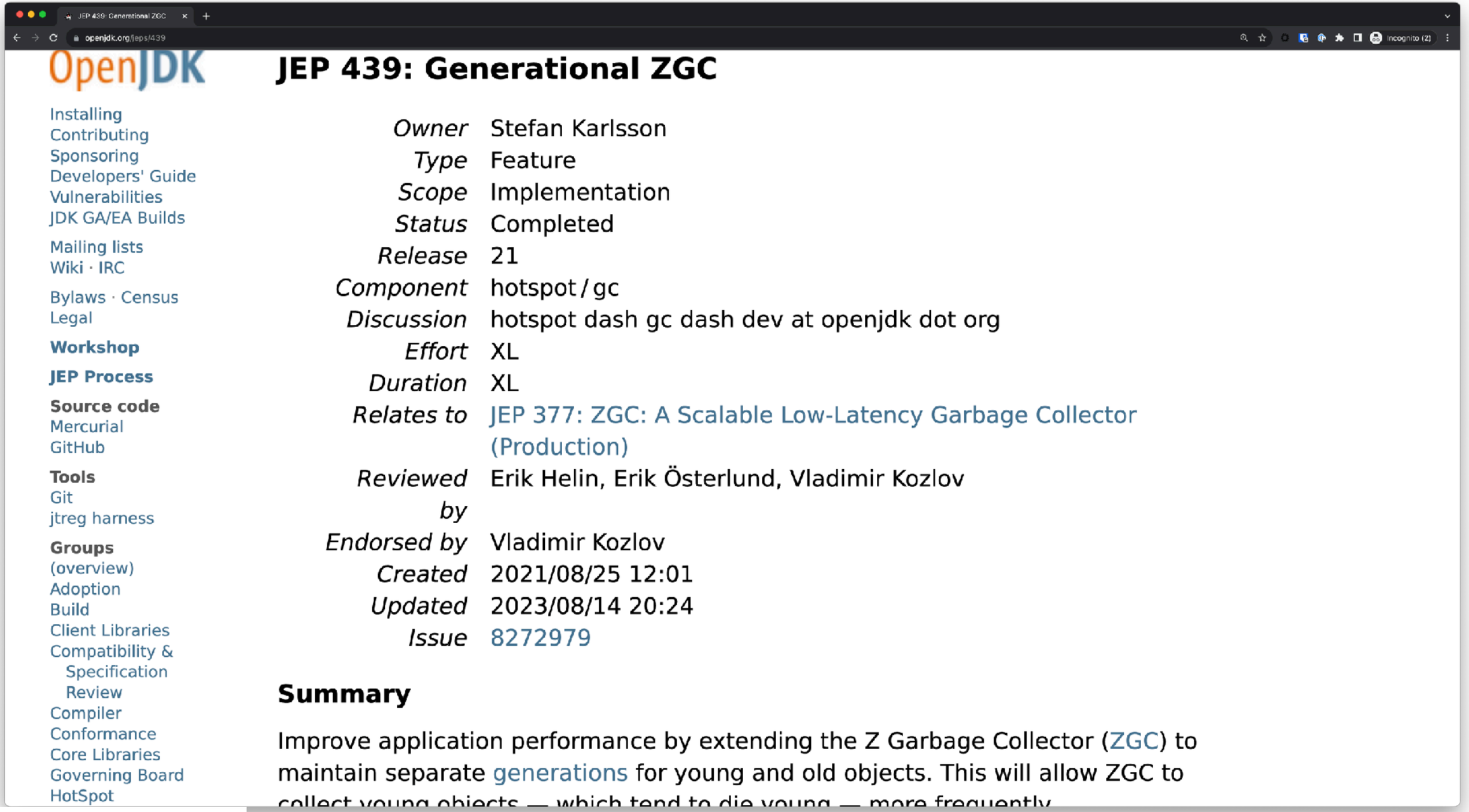


https://openjdk.org/jeps/408



https://openjdk.org/jeps/400

https://openjdk.org/jeps/421



OpenJDK

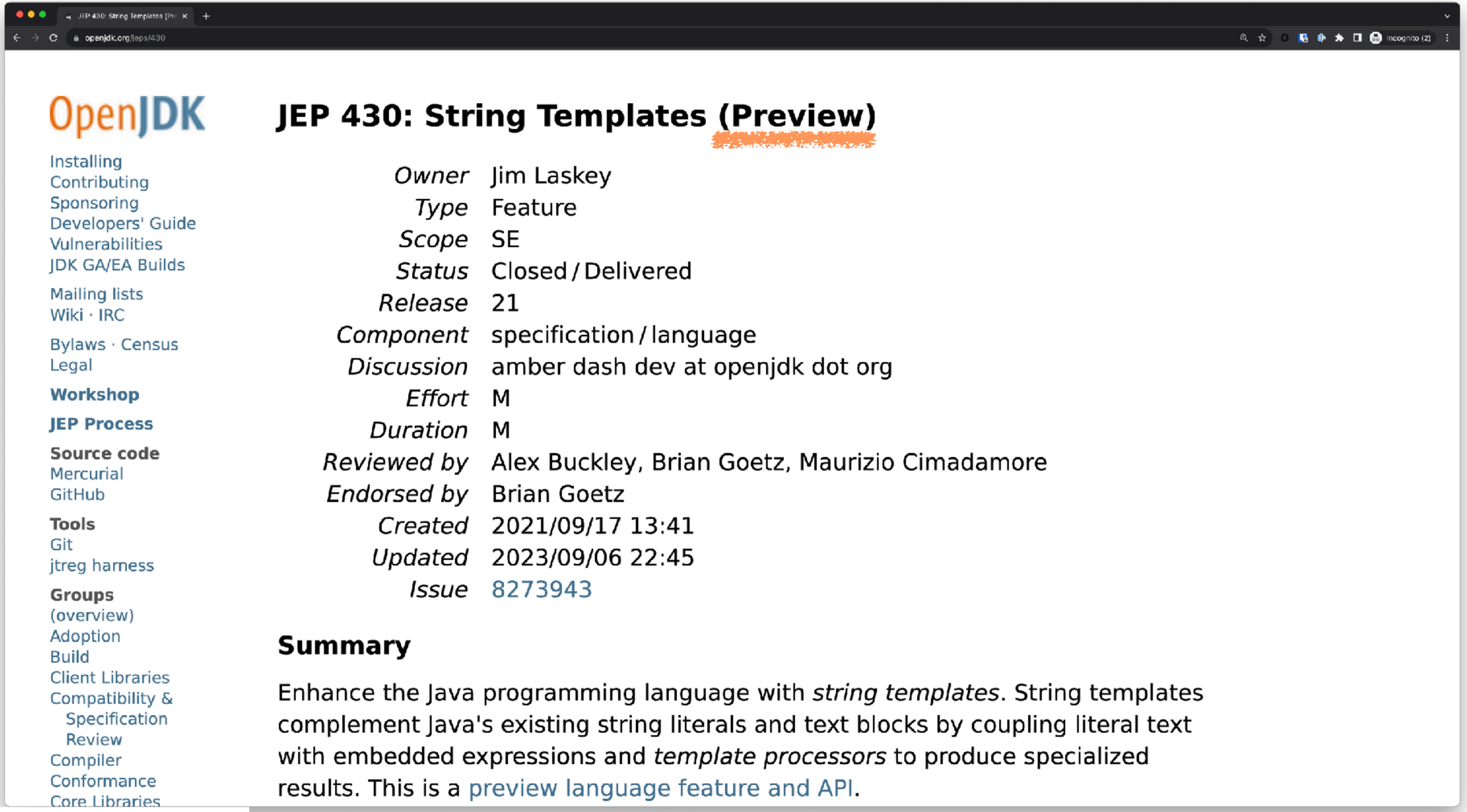
JEP 439: Generational ZGC

Owner	Stefan Karlsson
Type	Feature
Scope	Implementation
Status	Completed
Release	21
Component	hotspot / gc
Discussion	hotspot dash gc dash dev at openjdk dot org
Effort	XL
Duration	XL
Relates to	JEP 377: ZGC: A Scalable Low-Latency Garbage Collector (Production)
Reviewed by	Erik Helin, Erik Österlund, Vladimir Kozlov
Endorsed by	Vladimir Kozlov
Created	2021/08/25 12:01
Updated	2023/08/14 20:24
Issue	8272979

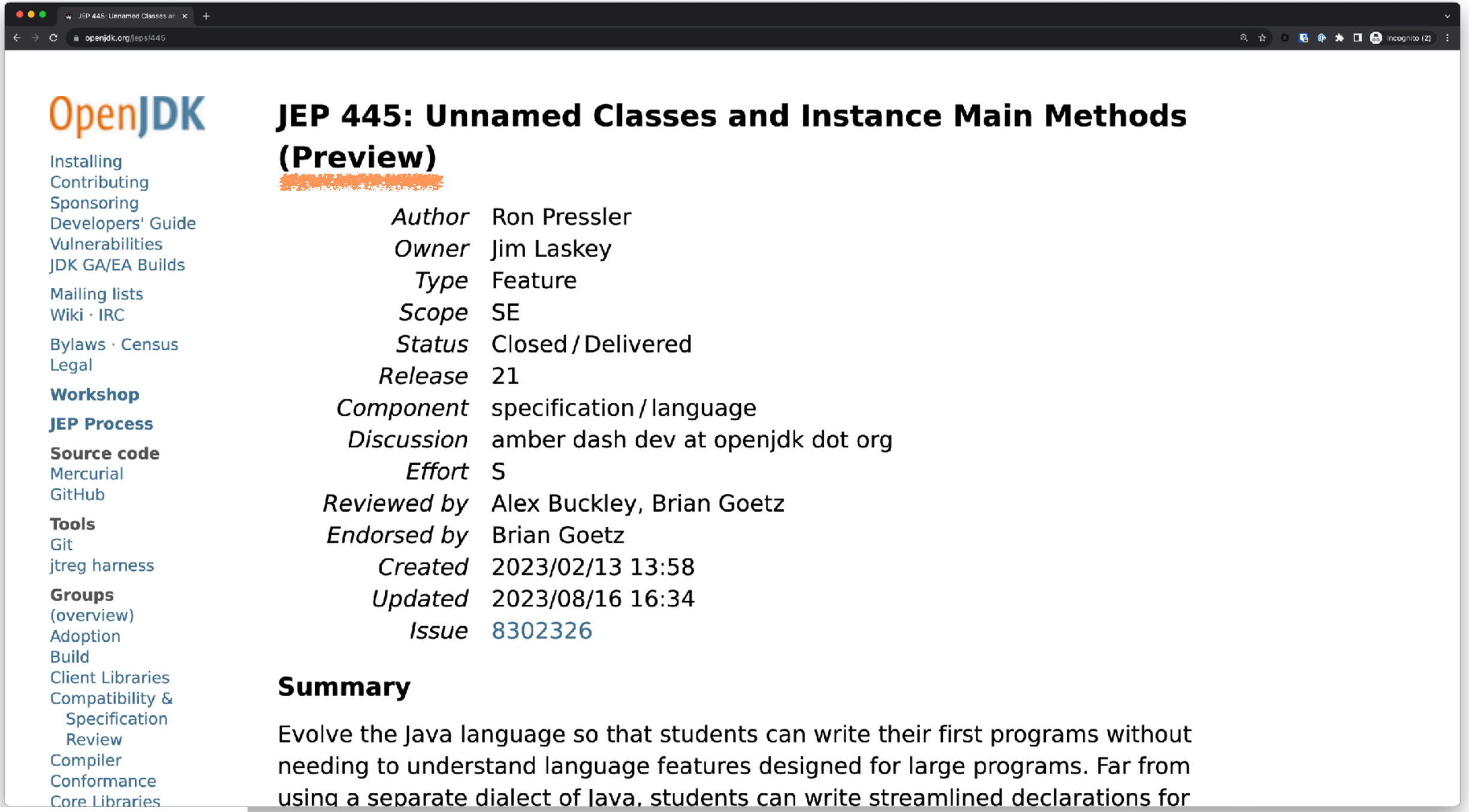
Summary

Improve application performance by extending the Z Garbage Collector (ZGC) to maintain separate generations for young and old objects. This will allow ZGC to collect young objects — which tend to die young — more frequently

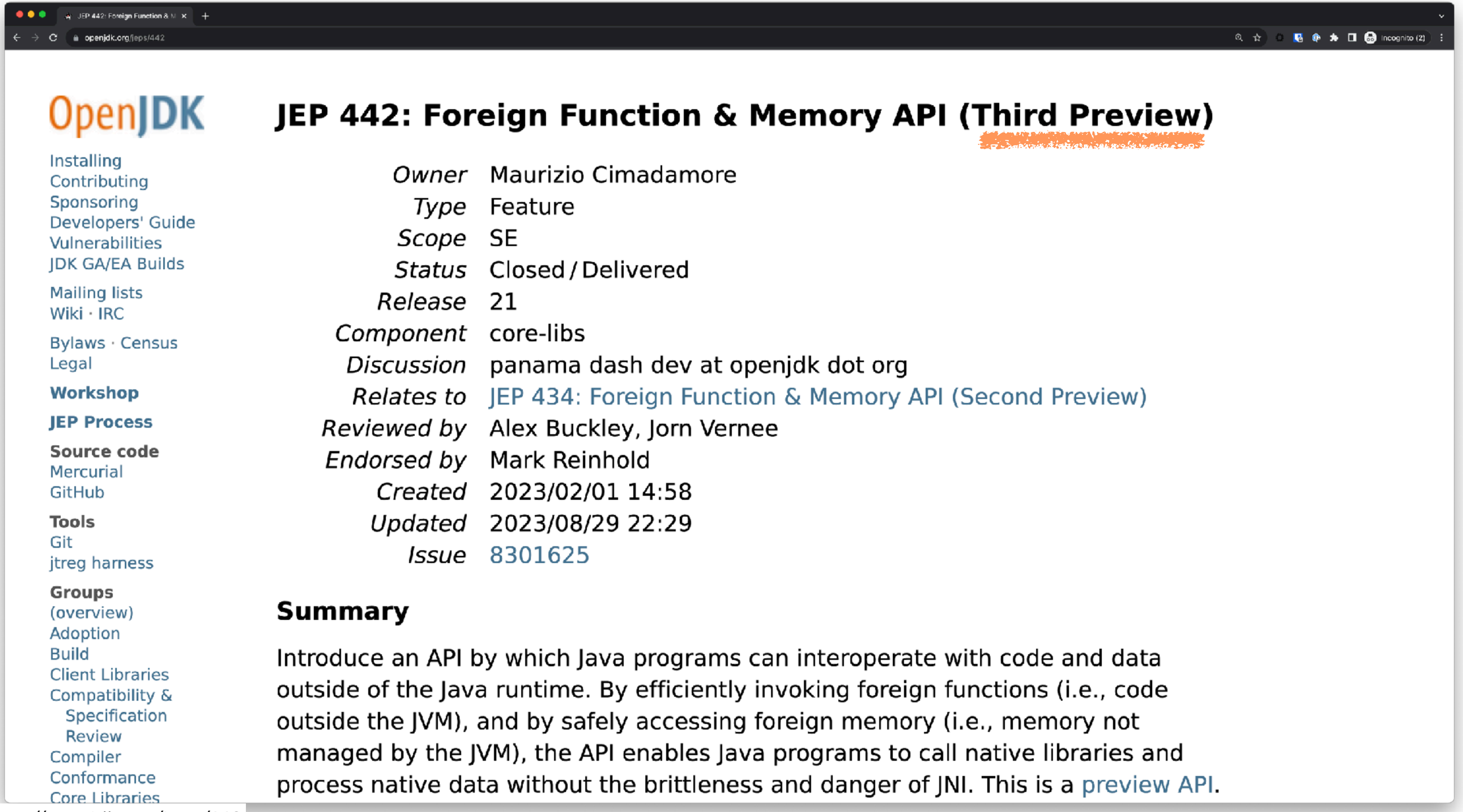
https://openjdk.org/jeps/439



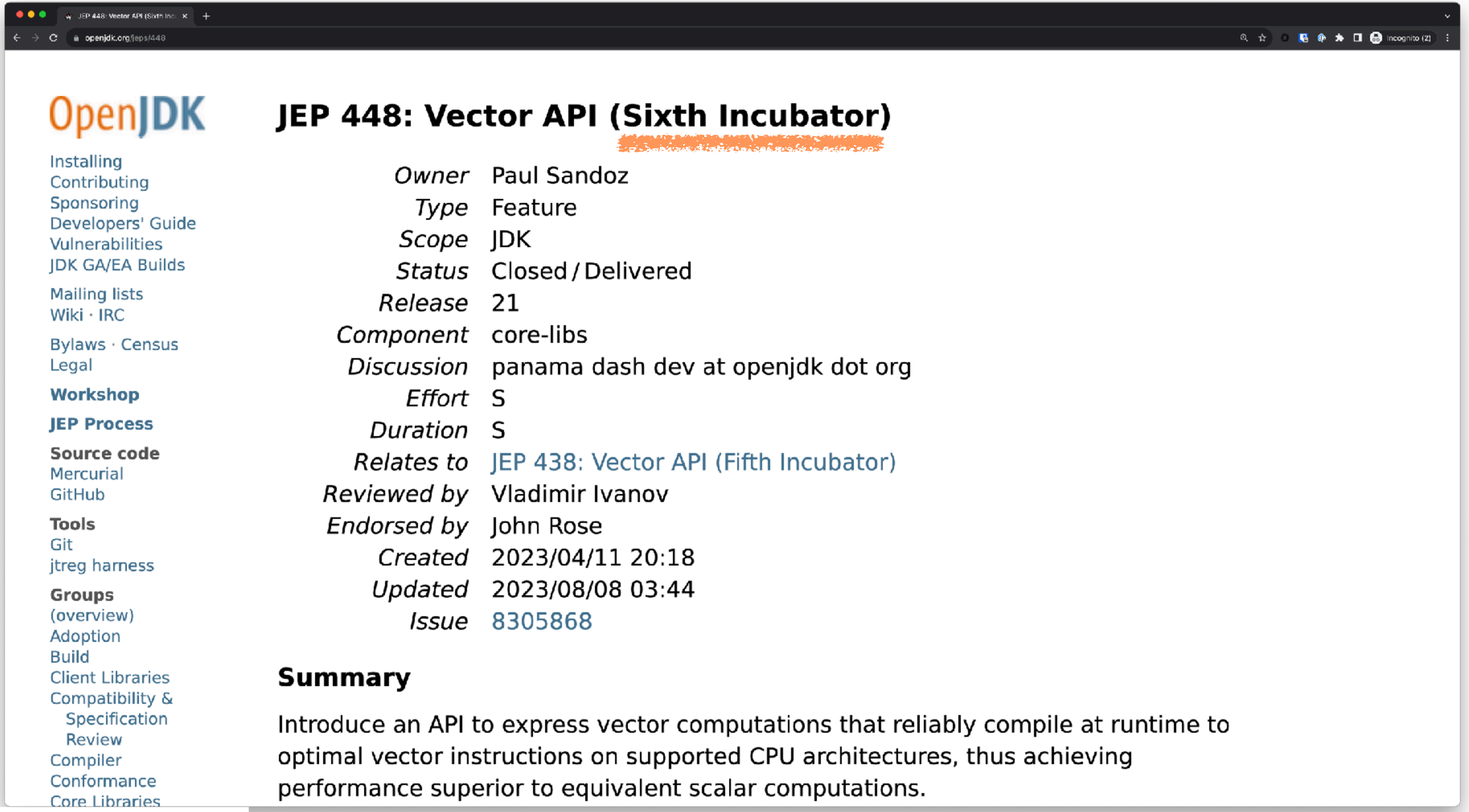
https://openjdk.org/jeps/430



https://openjdk.org/jeps/445

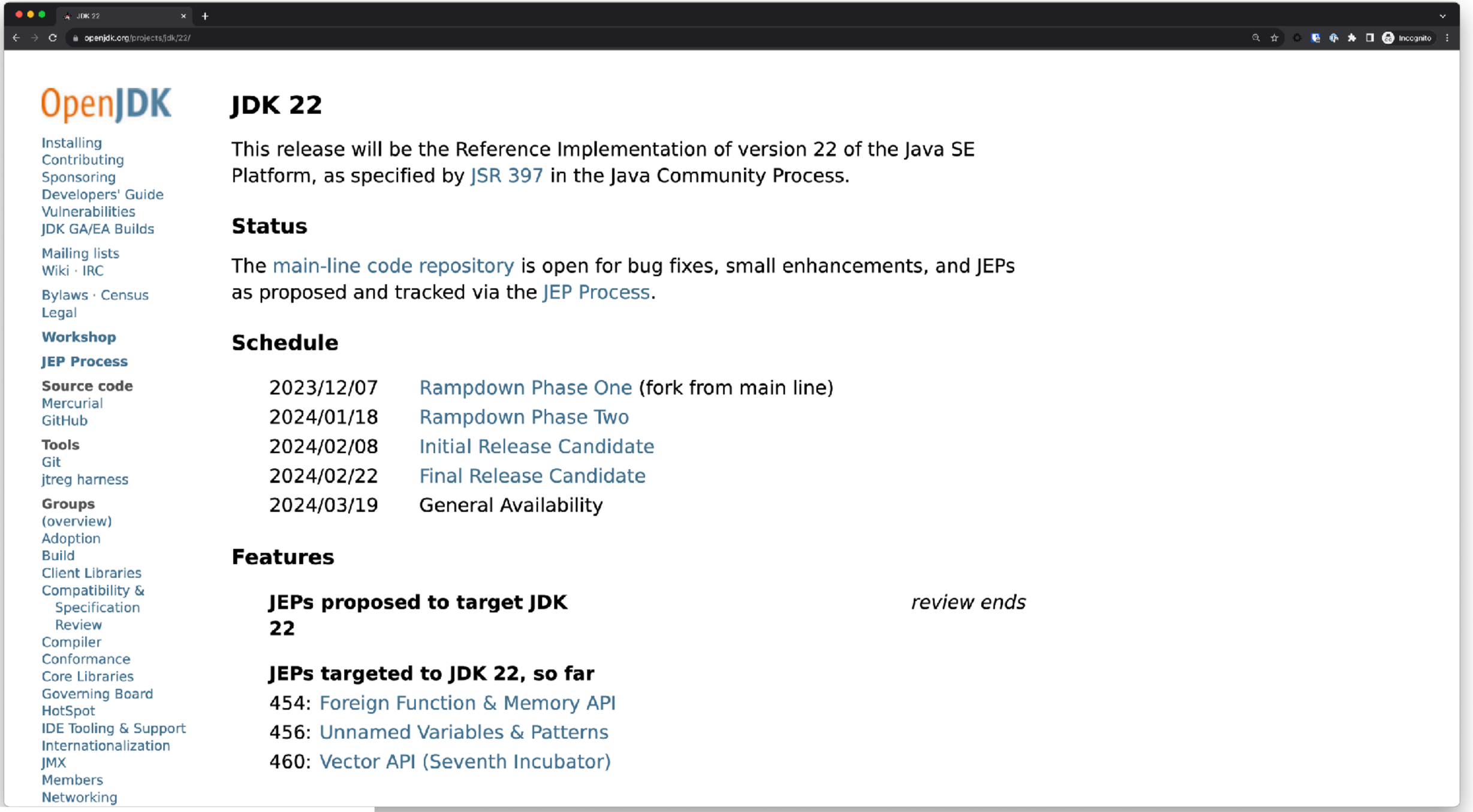


https://openjdk.org/jeps/442

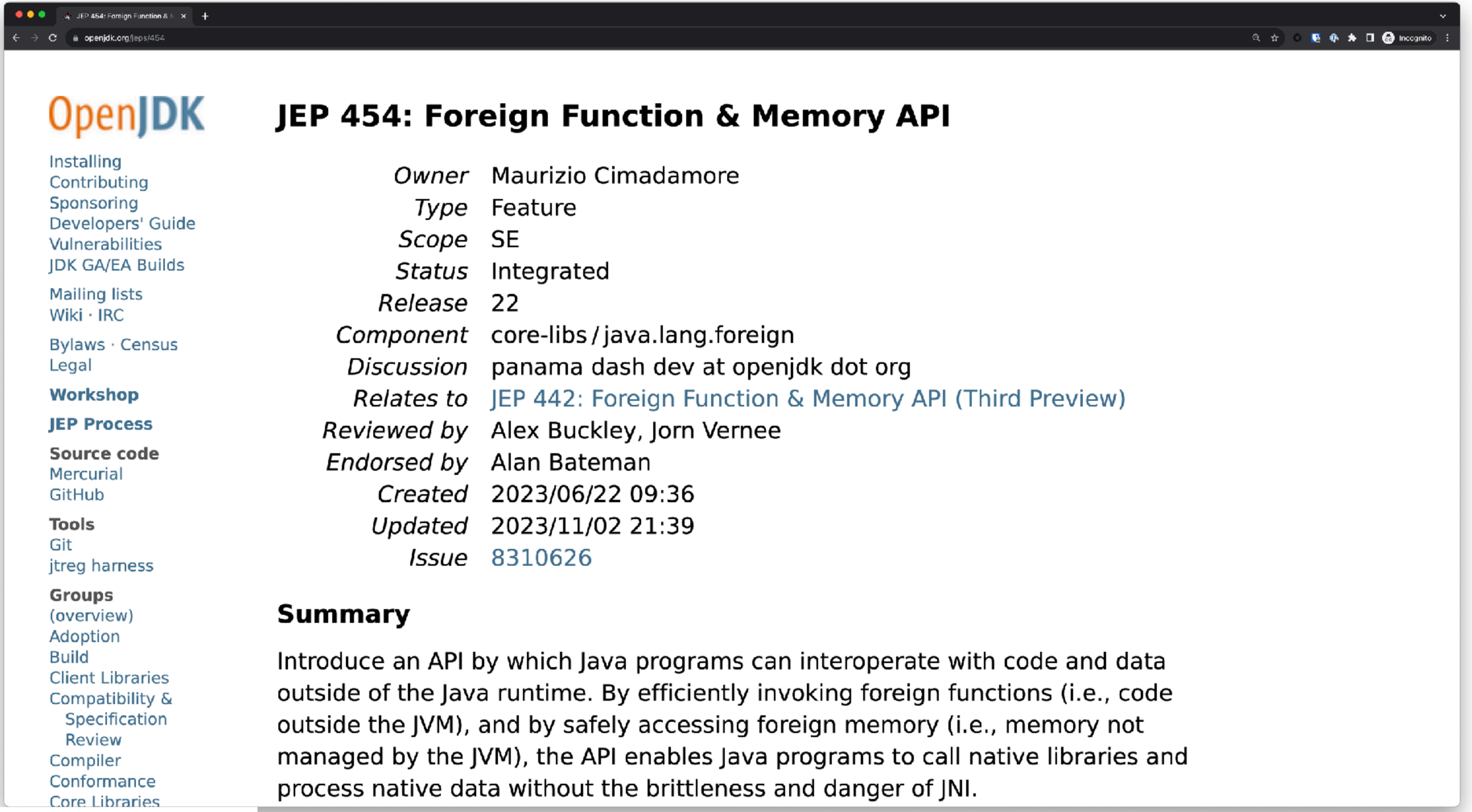


https://openjdk.org/jeps/448

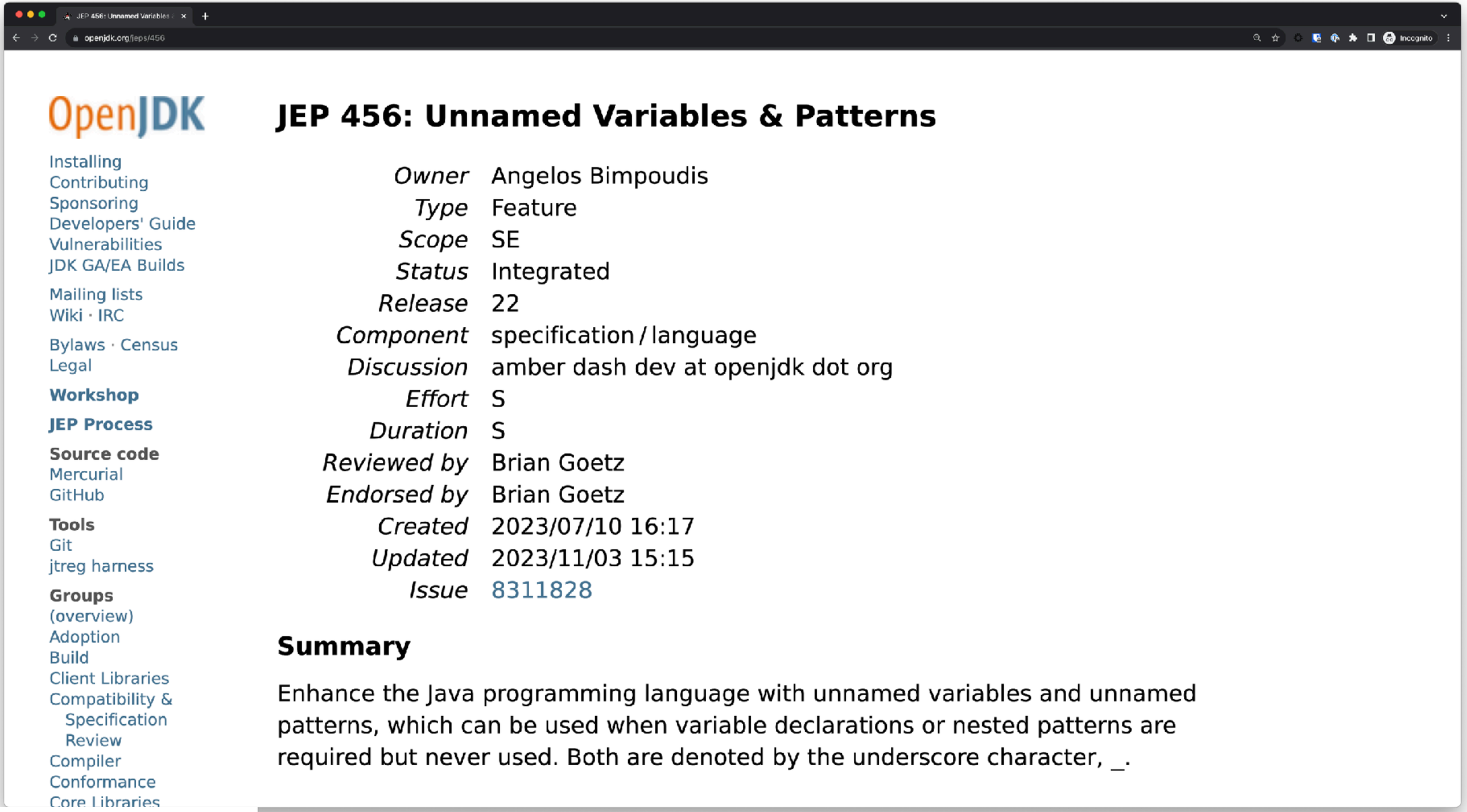
Future



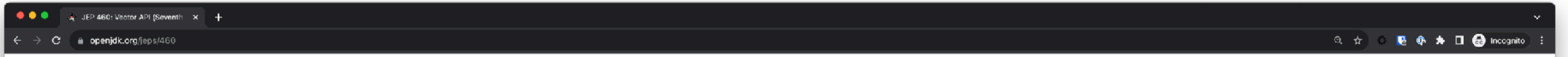
https://openjdk.java.net/projects/jdk/22/



https://openjdk.org/jeps/454



https://openjdk.org/jeps/456



OpenJDK

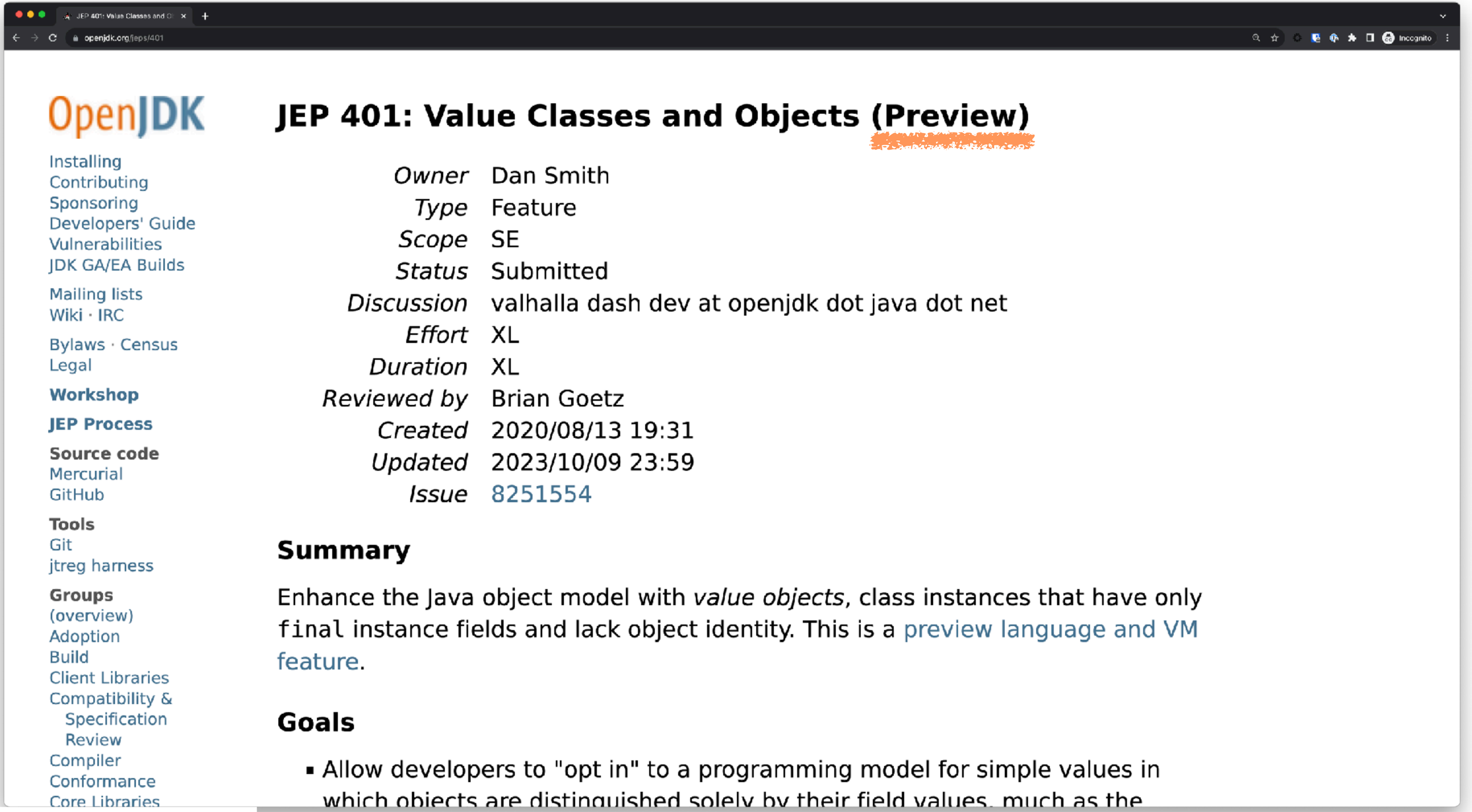
JEP 460: Vector API (Seventh Incubator)

Owner	Paul Sandoz
Type	Feature
Scope	JDK
Status	Targeted
Release	22
Component	core-libs
Discussion	panama dash dev at openjdk dot org
Effort	XS
Duration	XS
Relates to	JEP 448: Vector API (Sixth Incubator)
Reviewed by	Vladimir Ivanov
Endorsed by	John Rose
Created	2023/09/08 17:29
Updated	2023/11/06 13:40
Issue	8315945

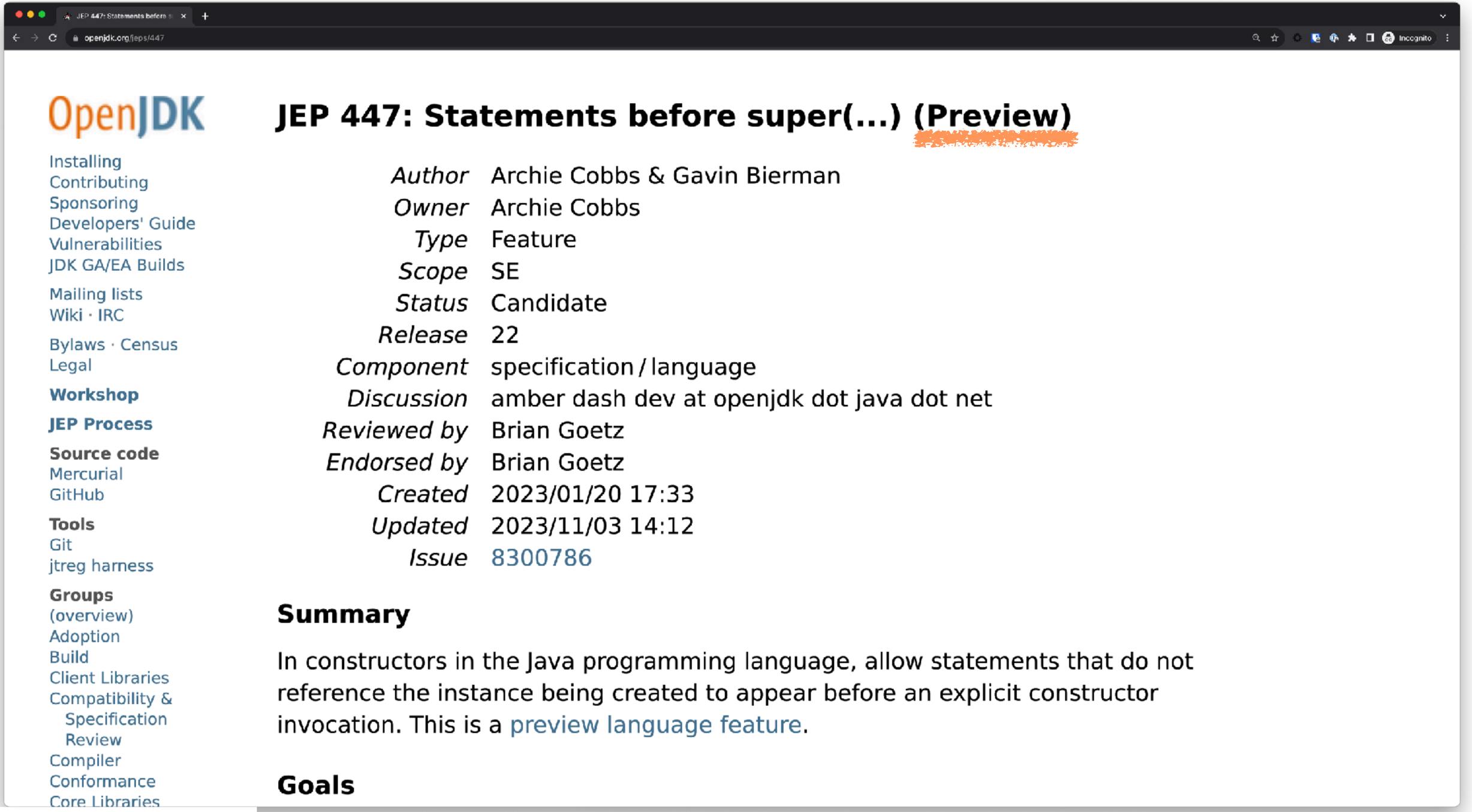
Summary

Introduce an API to express vector computations that reliably compile at runtime to optimal vector instructions on supported CPU architectures, thus achieving performance superior to equivalent scalar computations.

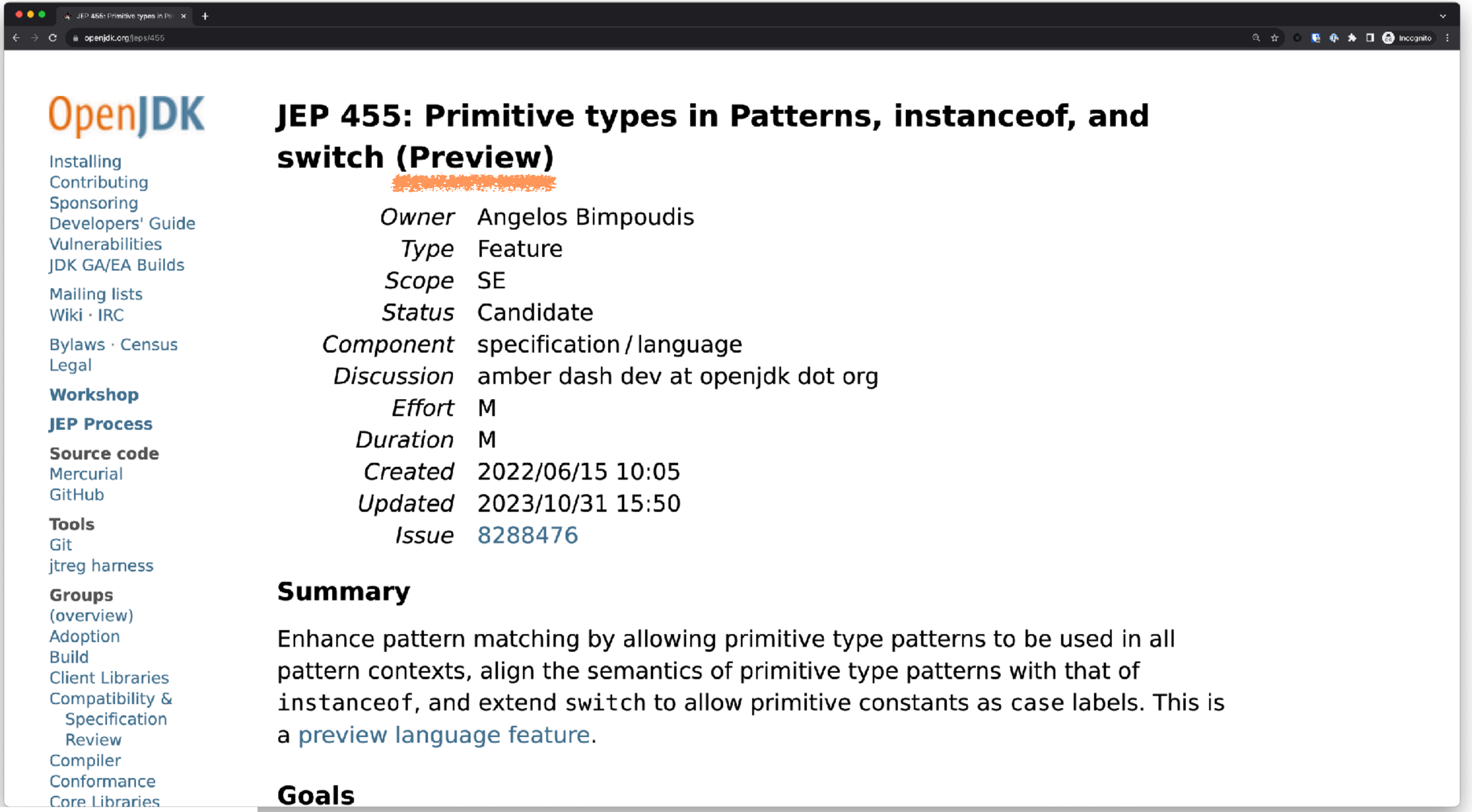
https://openjdk.org/jeps/460



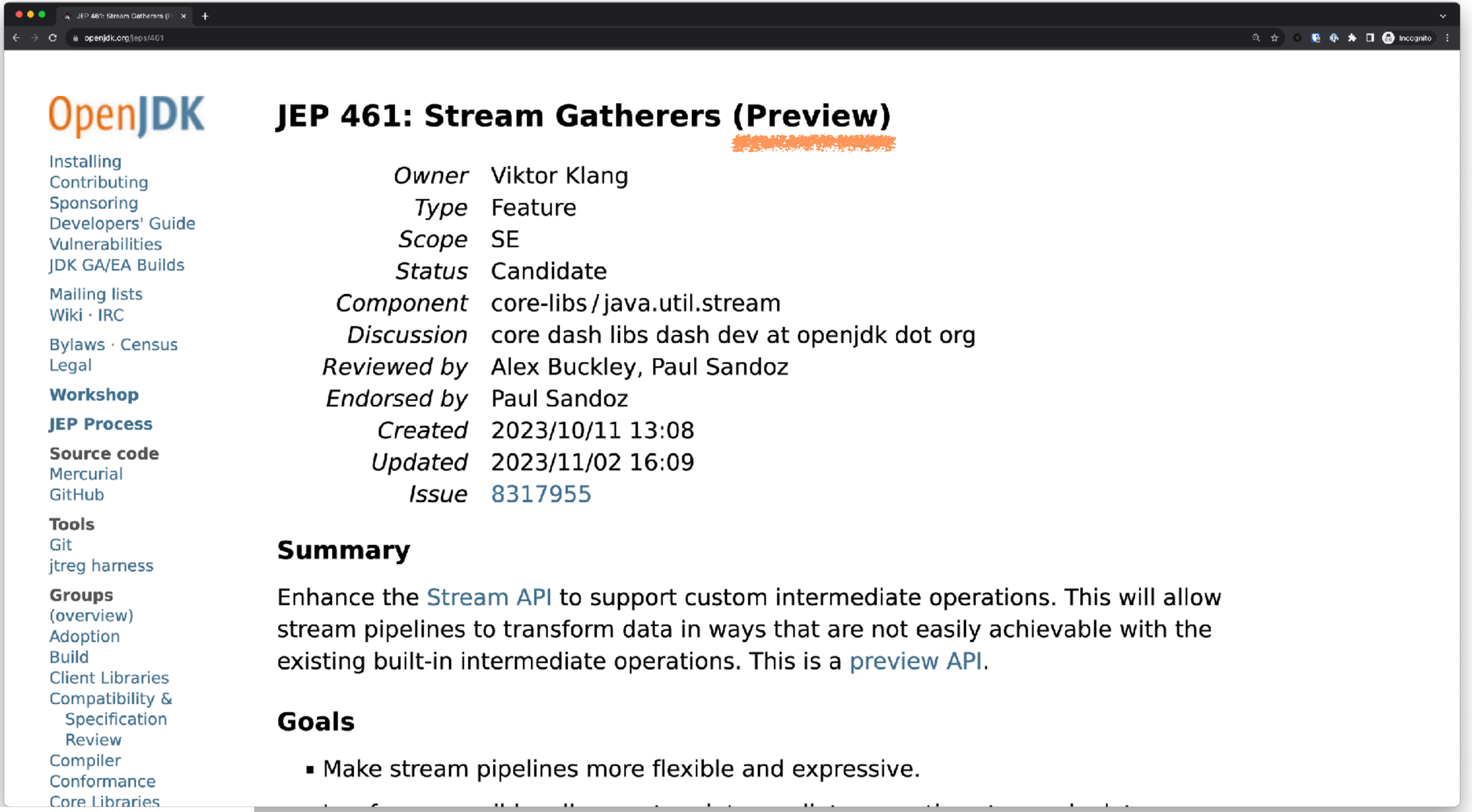
https://openjdk.org/jeps/401



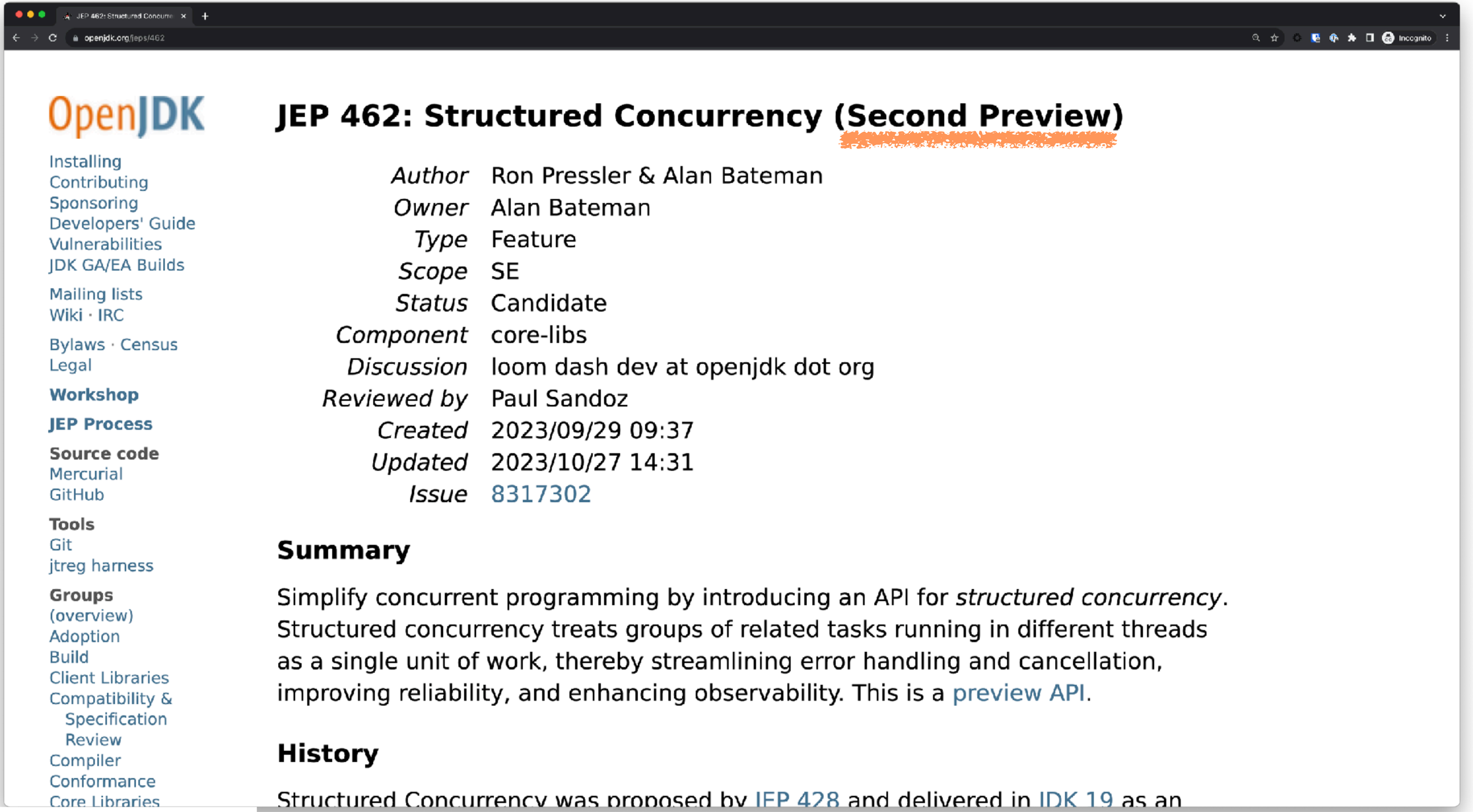
https://openjdk.org/jeps/447



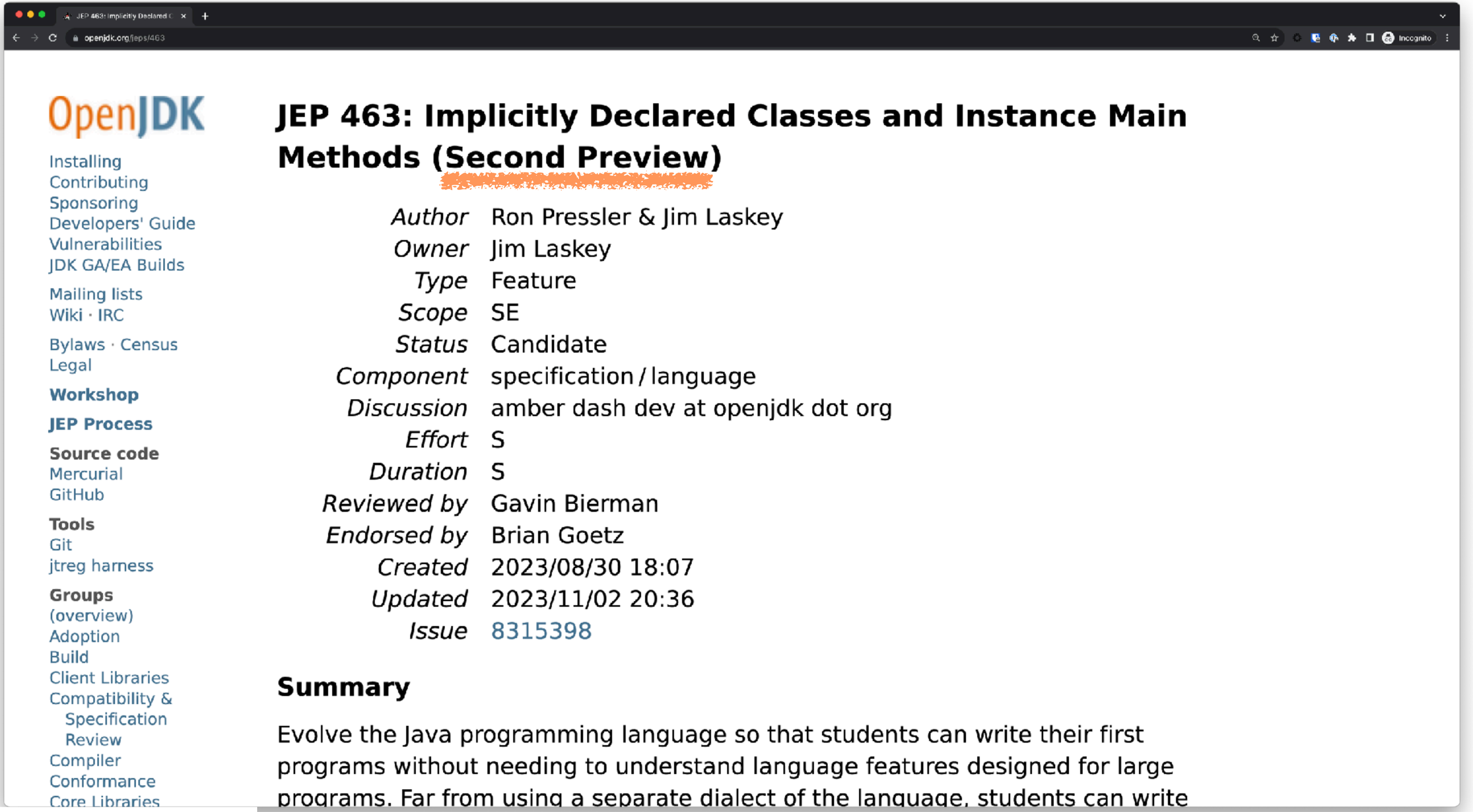
https://openjdk.org/jeps/455



https://openjdk.org/jeps/461



https://openjdk.org/jeps/462



OpenJDK

Installing
Contributing
Sponsoring
Developers' Guide
Vulnerabilities
JDK GA/EA Builds

Mailing lists
Wiki · IRC

Bylaws · Census
Legal

Workshop

JEP Process

Source code
Mercurial
GitHub

Tools
Git
jtreg harness

Groups
(overview)
Adoption
Build
Client Libraries
Compatibility & Specification Review
Compiler
Conformance
Core Libraries

JEP 463: Implicitly Declared Classes and Instance Main Methods (Second Preview)

Author	Ron Pressler & Jim Laskey
Owner	Jim Laskey
Type	Feature
Scope	SE
Status	Candidate
Component	specification / language
Discussion	amber dash dev at openjdk dot org
Effort	S
Duration	S
Reviewed by	Gavin Bierman
Endorsed by	Brian Goetz
Created	2023/08/30 18:07
Updated	2023/11/02 20:36
Issue	8315398

Summary

Evolve the Java programming language so that students can write their first programs without needing to understand language features designed for large programs. Far from using a separate dialect of the language, students can write

https://openjdk.org/jeps/463

OpenJDK

Installing
Contributing
Sponsoring
Developers' Guide
Vulnerabilities
JDK GA/EA Builds

Mailing lists
Wiki · IRC

Bylaws · Census
Legal

Workshop

JEP Process

Source code
Mercurial
GitHub

Tools
Git
jtreg harness

Groups
(overview)
Adoption
Build
Client Libraries
Compatibility &
Specification
Review
Compiler
Conformance
Core Libraries

JEP draft: Ahead of Time Compilation for the Java Virtual Machine

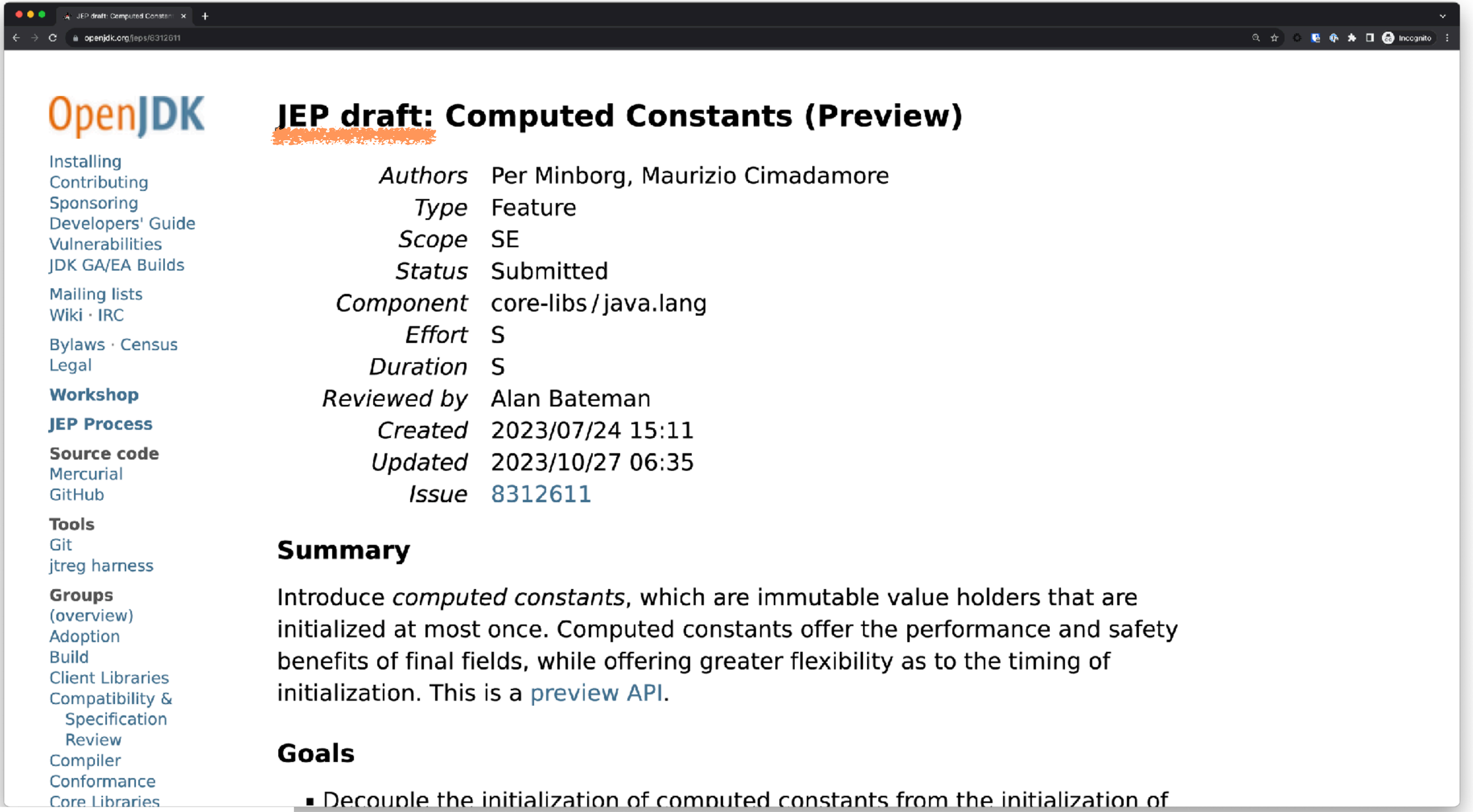
Author	Julian Waters
Type	Feature
Scope	JDK
Status	Draft
Component	hotspot / compiler
Created	2023/07/28 02:38
Updated	2023/09/29 01:56
Issue	8313278

Summary

Enhance the Java Virtual Machine with the ability to load Java applications and libraries compiled to native code for faster startup and baseline execution.

Goals

Enable the Java Virtual Machine to load Java code compiled Ahead of Time to native code, if it was compiled by a compiler from a matching Java Virtual Machine. Allow C1, C2, and JVMCI Compilers to code native code Ahead of Time for a Java Virtual Machine to use. "Profiling" code is code compiled by C1 in tiers 2 and 3 for



OpenJDK

JEP draft: Computed Constants (Preview)

Authors	Per Minborg, Maurizio Cimadamore
Type	Feature
Scope	SE
Status	Submitted
Component	core-libs / java.lang
Effort	S
Duration	S
Reviewed by	Alan Bateman
Created	2023/07/24 15:11
Updated	2023/10/27 06:35
Issue	8312611

Summary

Introduce *computed constants*, which are immutable value holders that are initialized at most once. Computed constants offer the performance and safety benefits of final fields, while offering greater flexibility as to the timing of initialization. This is a preview API.

Goals

- Decouple the initialization of computed constants from the initialization of

https://openjdk.org/jeps/8312611

More

More Ressources

- German article by me:
  https://www.innoq.com/de/articles/2023/09/java-21/
- German podcast with me:
  https://www.innoq.com/de/podcast/135-java-21/
- Data Oriented Programming in Java:
  https://www.infoq.com/articles/data-oriented-programming-java/
- JavaDoc Snippets with Maven:
  https://nipafx.dev/javadoc-snippets-maven/
- API Changes from 17 to 21:
  https://javaalmanac.io/jdk/21/apidiff/17/
- Help for choosing a JDK distribution:
  https://whichjdk.com/

More Ressources

- Presentation of Gatherers: https://www.youtube.com/watch?v=8fMFa6OqlY8

Thanks! Questions?

Michael Vitz

Mail	michael.vitz@innoq.com
X	@michaelvitz
Mastodon	@michaelvitz@innoq.social
LinkedIn	michaelvitz

https://www.innoq.com/en/talks/2023/11/das-erwartet-dich-mit-jdk-21/