Java Forum Nord / Hannover / 10.09.2024

Wie, schon wieder ein JDK Release? Die Neuerungen Live on Stage!

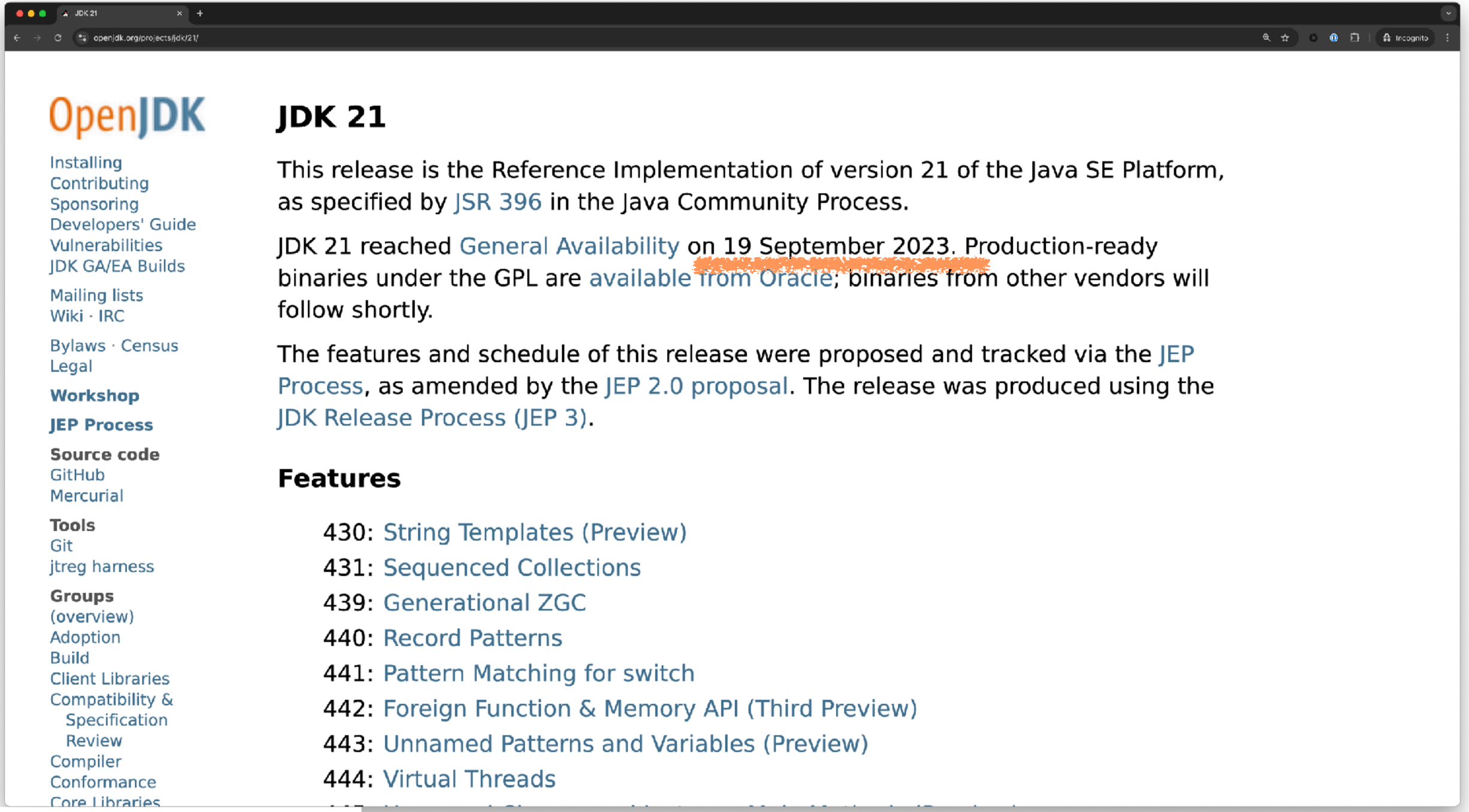
INNOQ

MICHAEL VITZ
SENIOR CONSULTANT

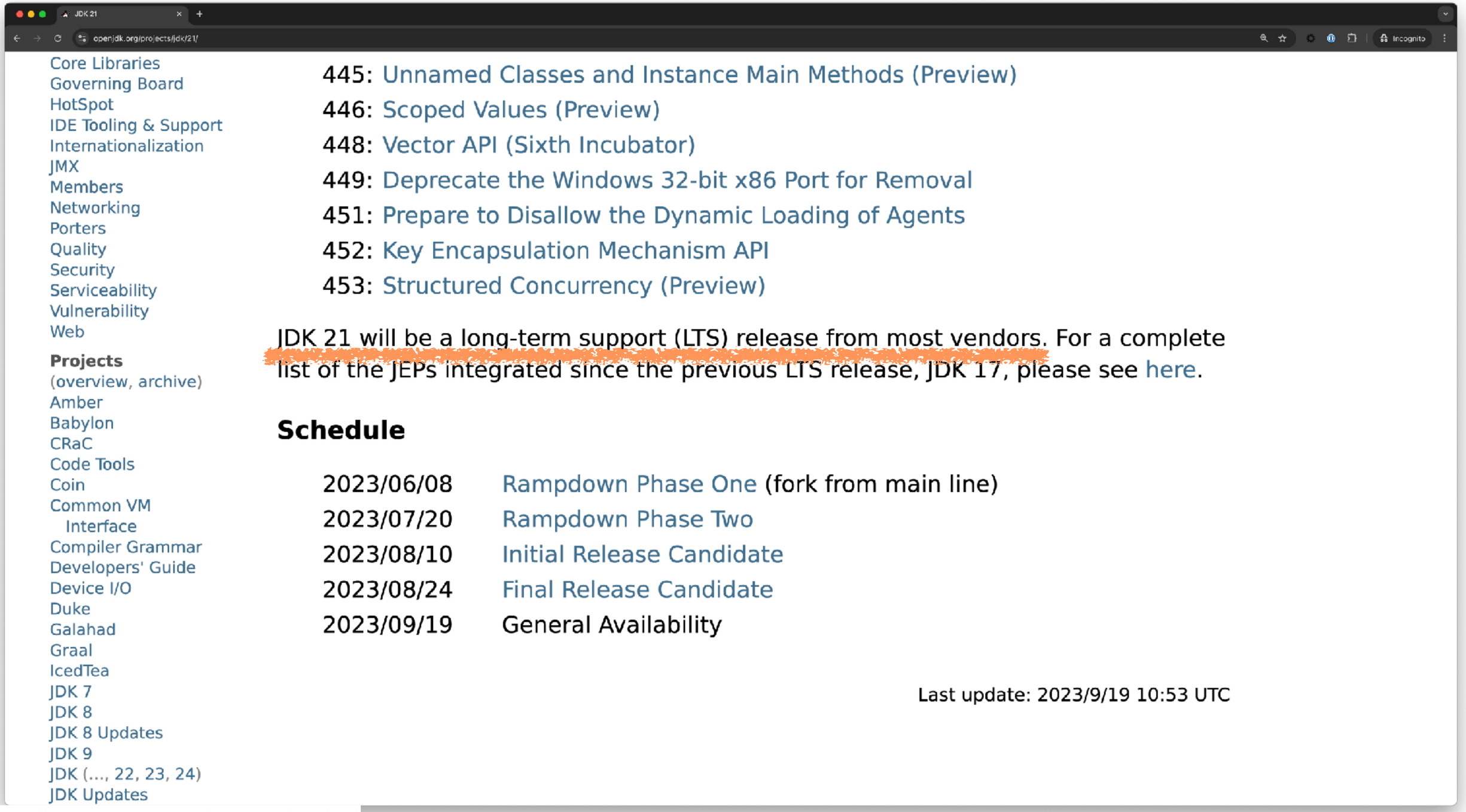
MICHAEL VITZ

Java Champion
Senior Consultant at INNOQ

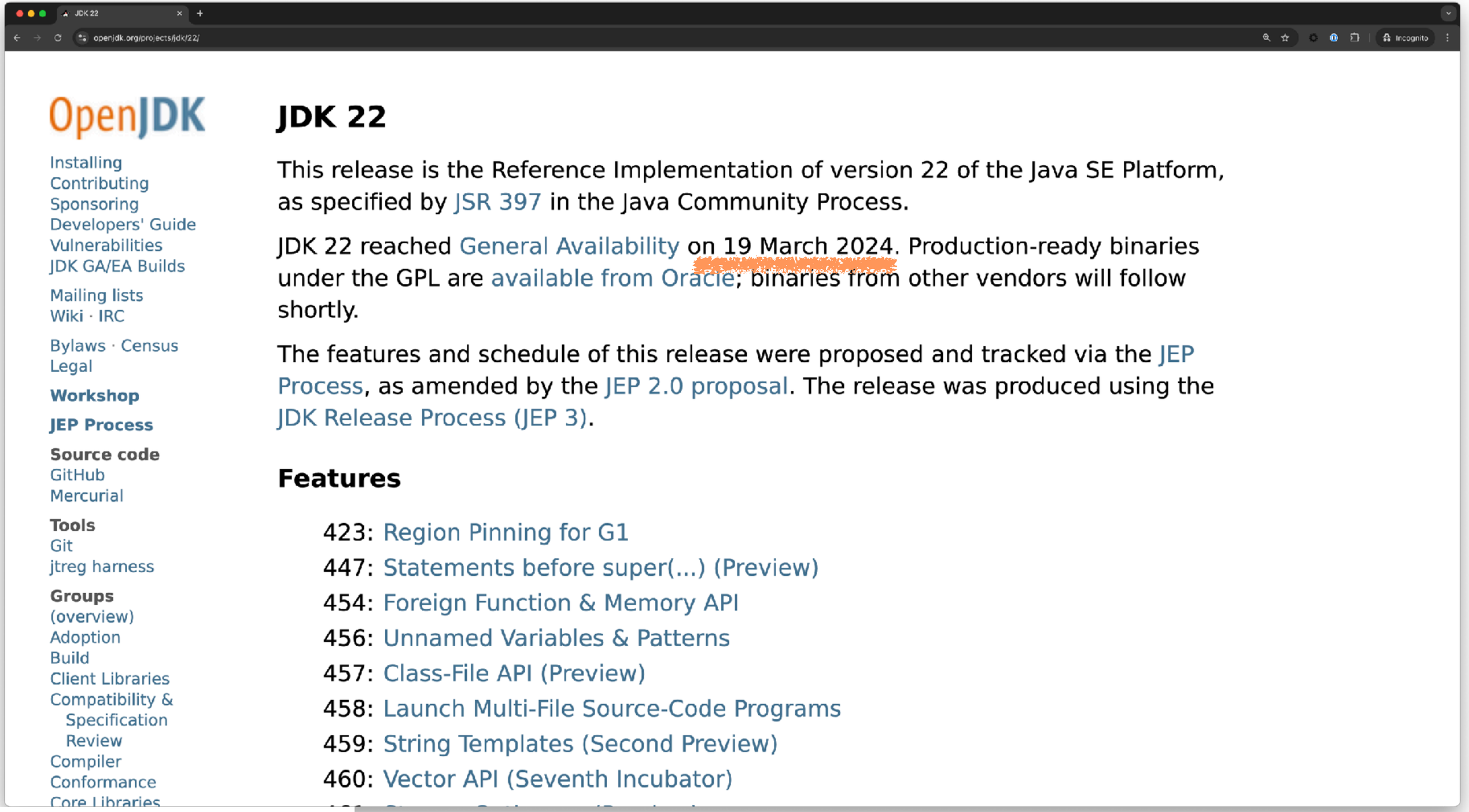
Quick Recap



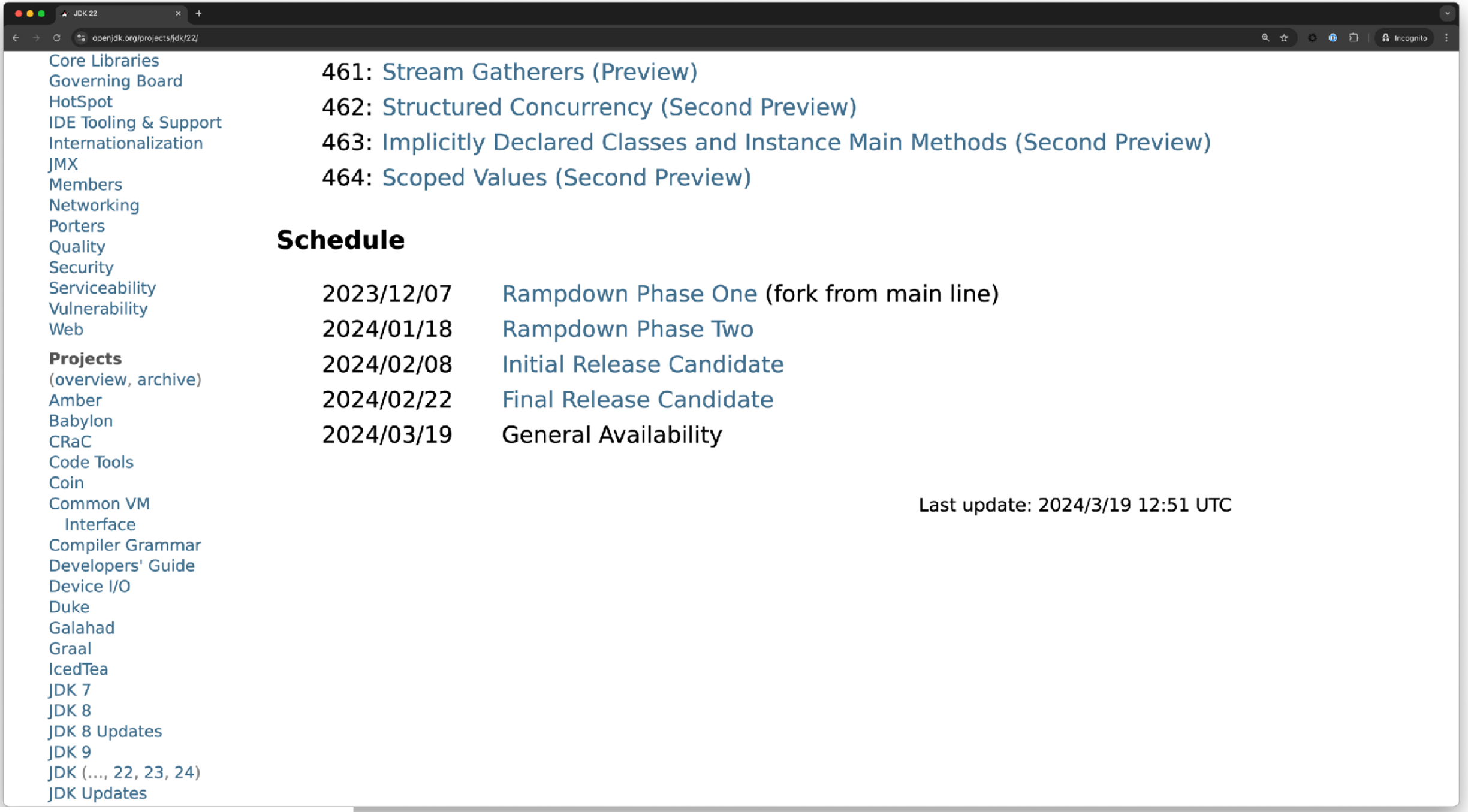
https://openjdk.java.net/projects/jdk/21/



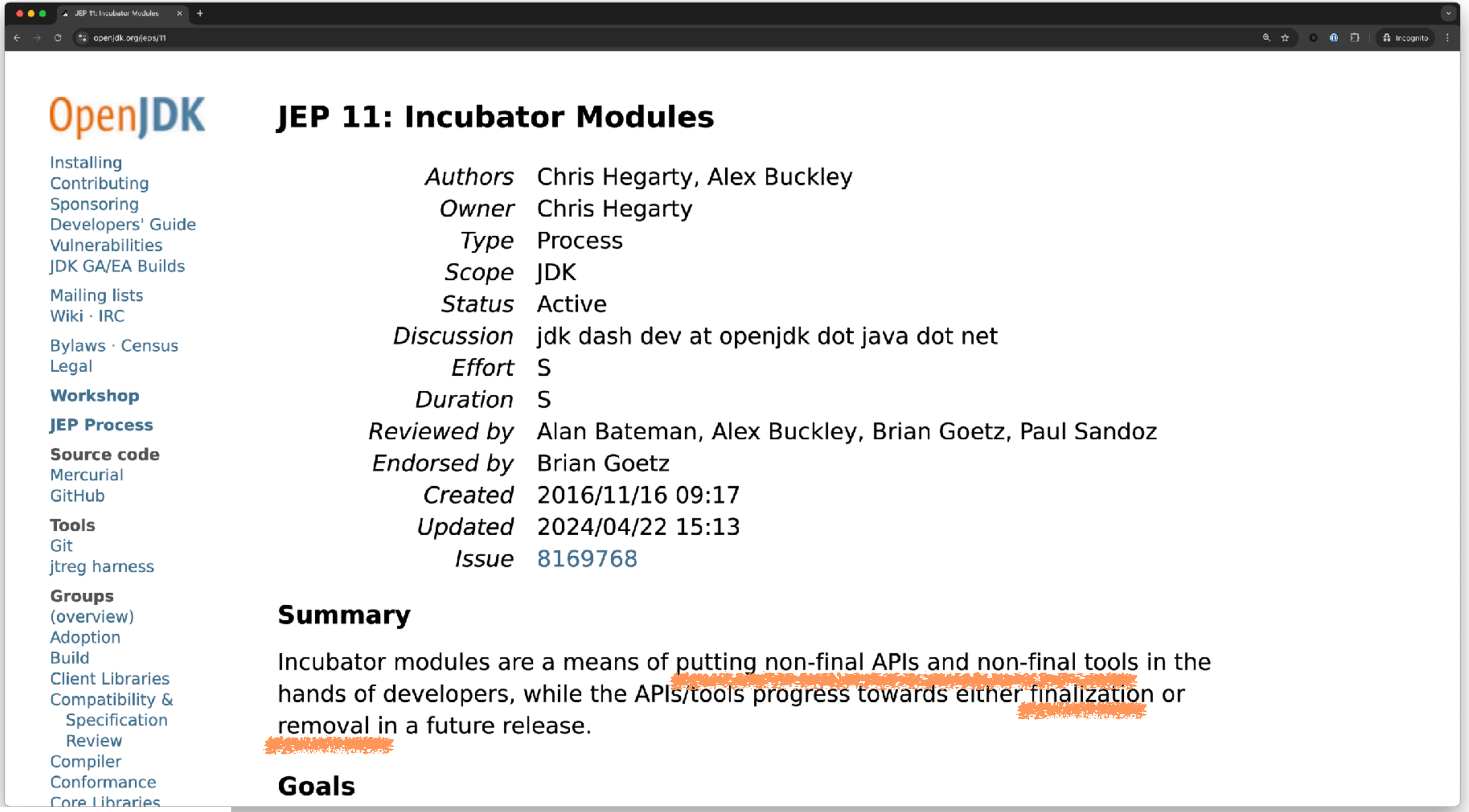
https://openjdk.java.net/projects/jdk/21/



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



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



https://openjdk.org/jeps/11

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 12: Preview Features

Owner	Alex Buckley
Type	Process
Scope	SE
Status	Active
Discussion	jdk dash dev at openjdk dot java dot net
Effort	M
Duration	M
Reviewed by	Alan Bateman, Brian Goetz, Mark Reinhold
Endorsed by	Mark Reinhold
Created	2018/01/19 01:27
Updated	2024/04/22 15:13
Issue	8195734

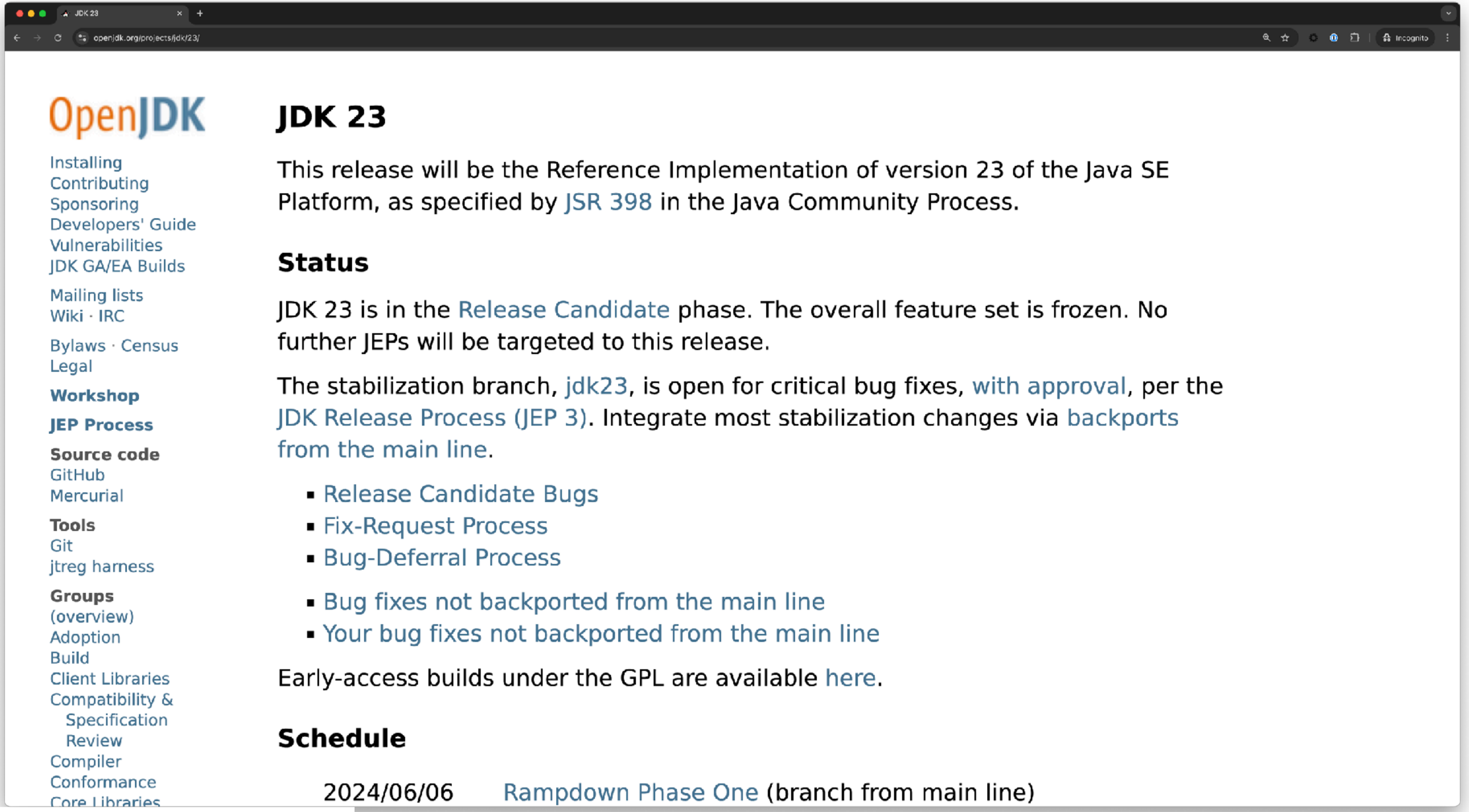
Summary

A *preview feature* is a new feature of the Java language, Java Virtual Machine, or Java SE API that is fully specified, fully implemented, and yet impermanent. It is available in a JDK feature release to provoke developer feedback based on real world use; this may lead to it becoming permanent in a future Java SE Platform.

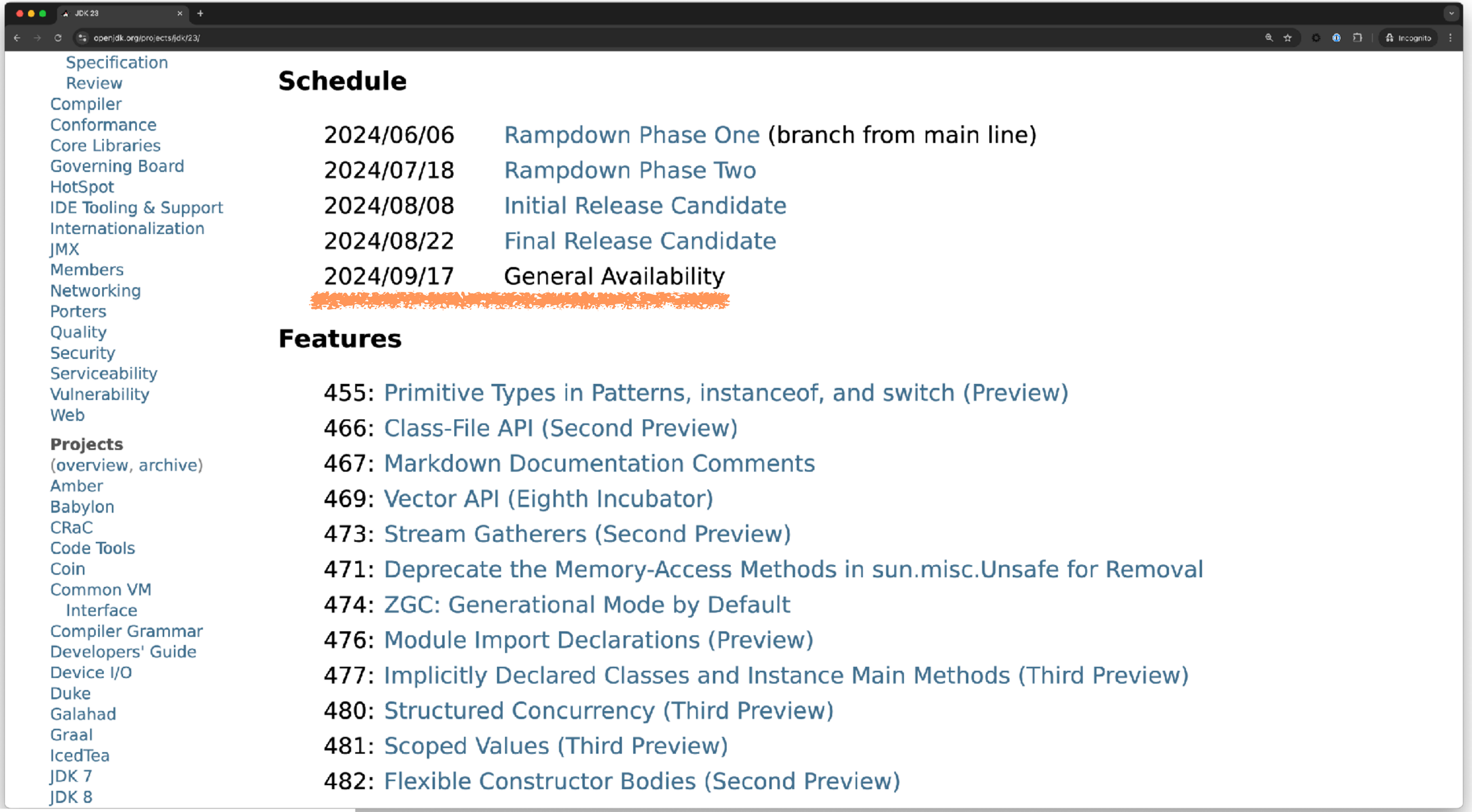
Goals

https://openjdk.org/jeps/12

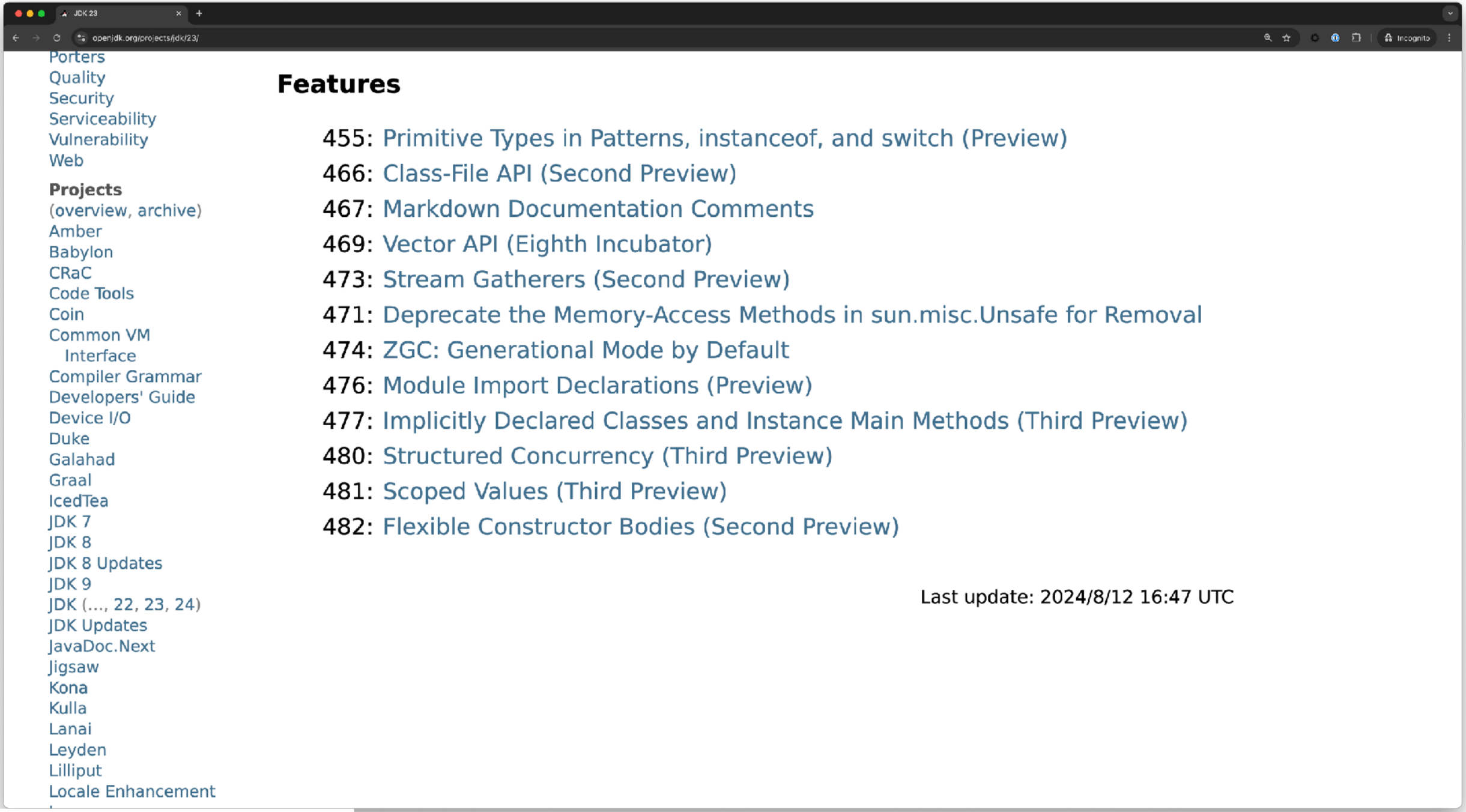
JDK 23



https://openjdk.java.net/projects/jdk/23



https://openjdk.java.net/projects/jdk/23



https://openjdk.java.net/projects/jdk/23

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 465: String Templates (Third Preview)

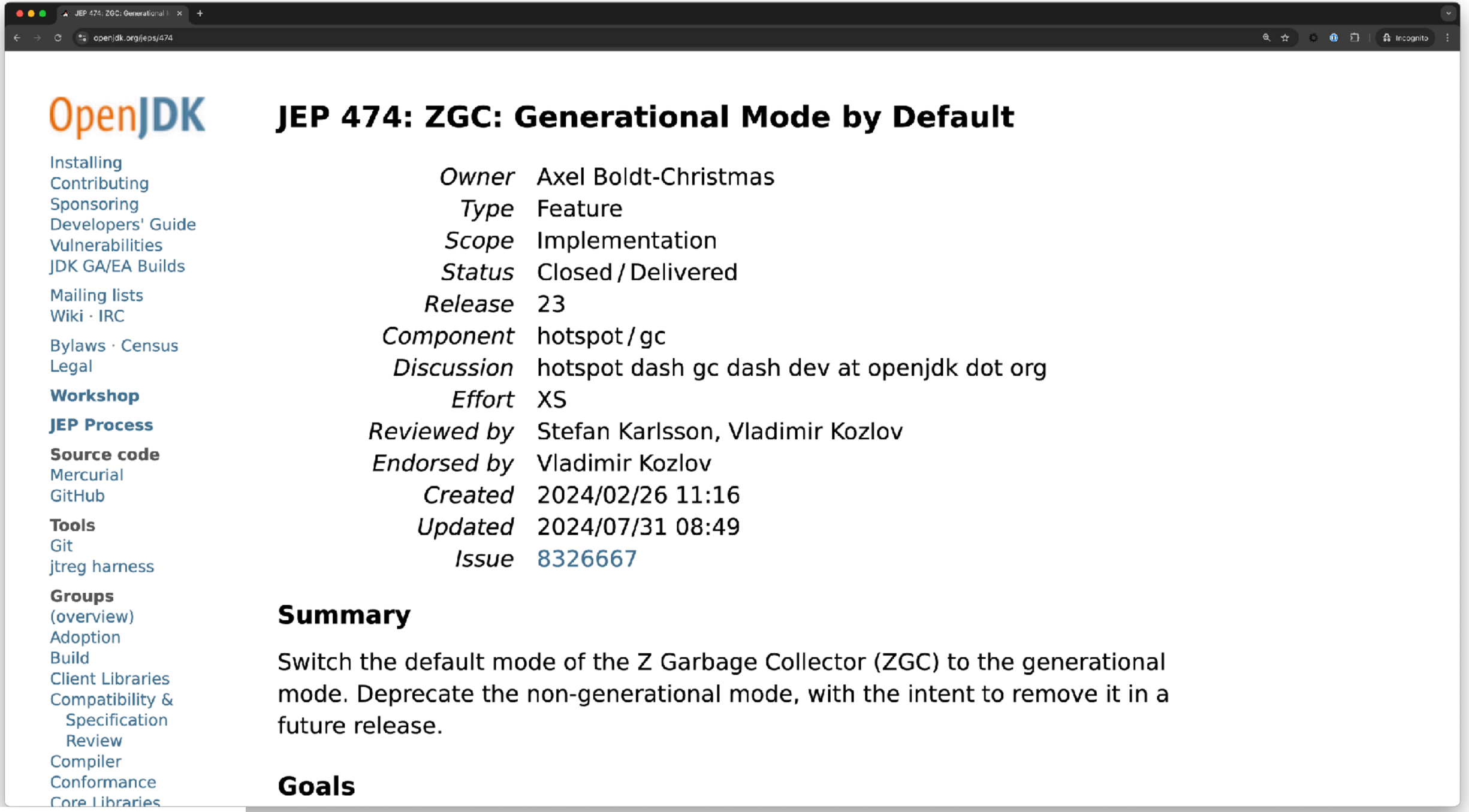
Author	Jim Laskey
Owner	Gavin Bierman
Type	Feature
Scope	SE
Status	Closed / Withdrawn
Component	specification / language
Discussion	amber dash dev at openjdk dot org
Effort	M
Duration	M
Relates to	JEP 459: String Templates (Second Preview)
Reviewed by	Mark Reinhold
Created	2024/01/09 16:54
Updated	2024/06/21 08:56
Issue	8323333

Summary

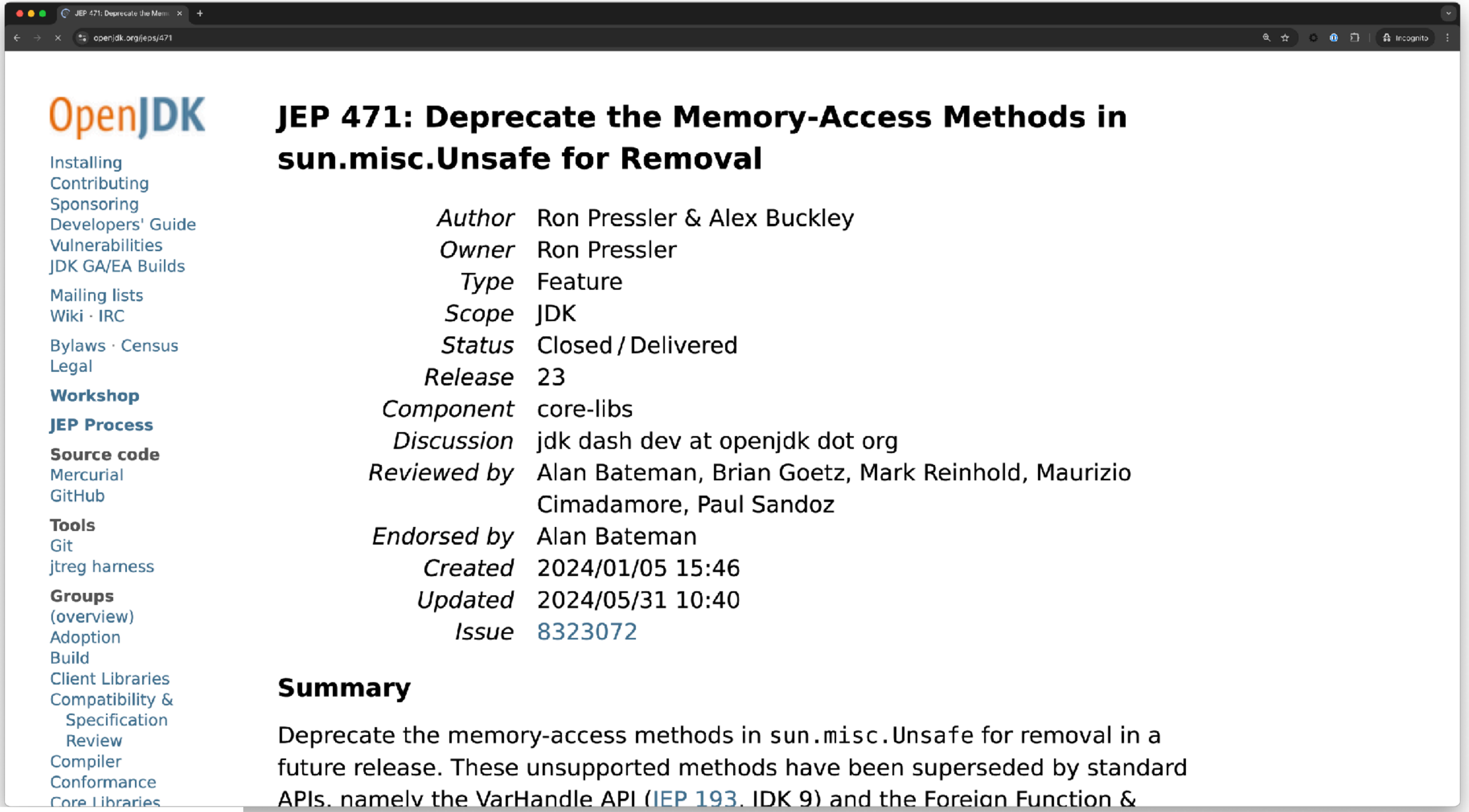
Enhance the Java programming language with *string templates*. String templates complement Java's existing string literals and text blocks by coupling literal text with embedded expressions and *template processors* to produce specialized results.

https://openjdk.org/jeps/465

JEPs



https://openjdk.org/jeps/474



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 471: Deprecate the Memory-Access Methods in sun.misc.Unsafe for Removal

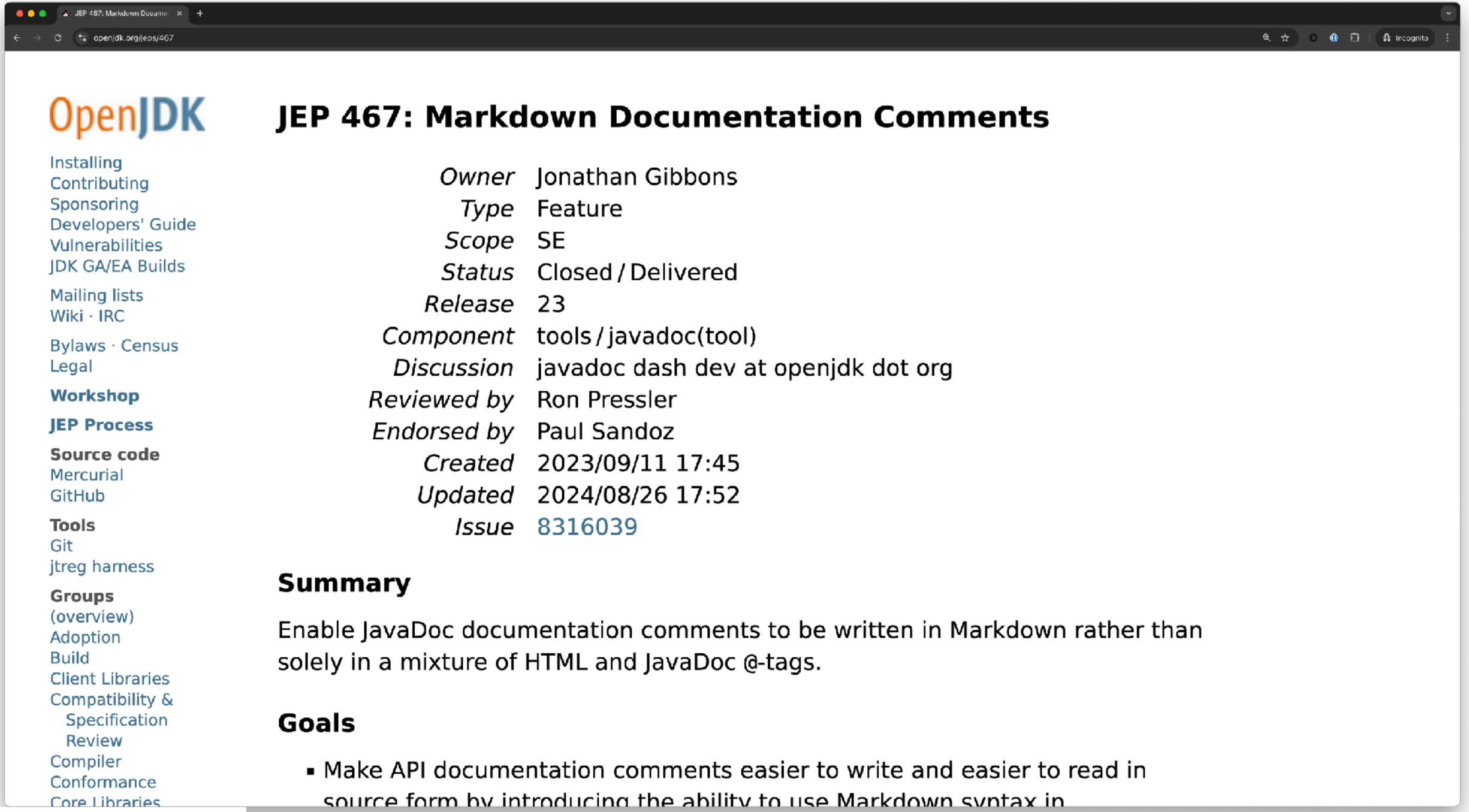
Author	Ron Pressler & Alex Buckley
Owner	Ron Pressler
Type	Feature
Scope	JDK
Status	Closed / Delivered
Release	23
Component	core-libs
Discussion	jdk dash dev at openjdk dot org
Reviewed by	Alan Bateman, Brian Goetz, Mark Reinhold, Maurizio Cimadamore, Paul Sandoz
Endorsed by	Alan Bateman
Created	2024/01/05 15:46
Updated	2024/05/31 10:40
Issue	8323072

Summary

Deprecate the memory-access methods in `sun.misc.Unsafe` for removal in a future release. These unsupported methods have been superseded by standard APIs, namely the VarHandle API (JEP 193, JDK 9) and the Foreign Function &

https://openjdk.org/jeps/471

```
// 'allocateMemory(long)' is deprecated since version 23 and marked for removal
sun.misc.Unsafe.getUnsafe().allocateMemory(1);
```



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 467: Markdown Documentation Comments

Owner	Jonathan Gibbons
Type	Feature
Scope	SE
Status	Closed / Delivered
Release	23
Component	tools / javadoc(tool)
Discussion	javadoc dash dev at openjdk dot org
Reviewed by	Ron Pressler
Endorsed by	Paul Sandoz
Created	2023/09/11 17:45
Updated	2024/08/26 17:52
Issue	8316039

Summary

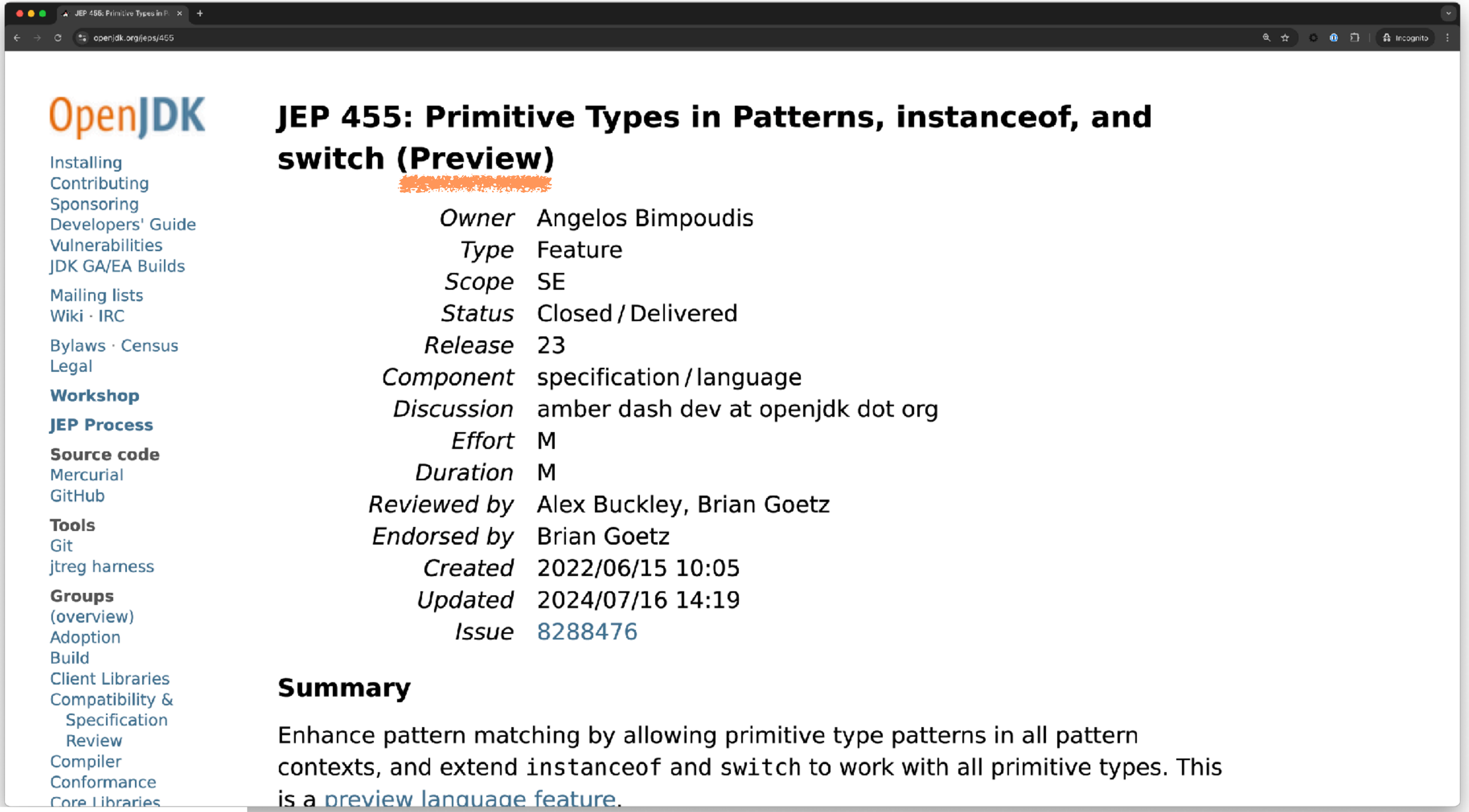
Enable JavaDoc documentation comments to be written in Markdown rather than solely in a mixture of HTML and JavaDoc @-tags.

Goals

- Make API documentation comments easier to write and easier to read in source form by introducing the ability to use Markdown syntax in

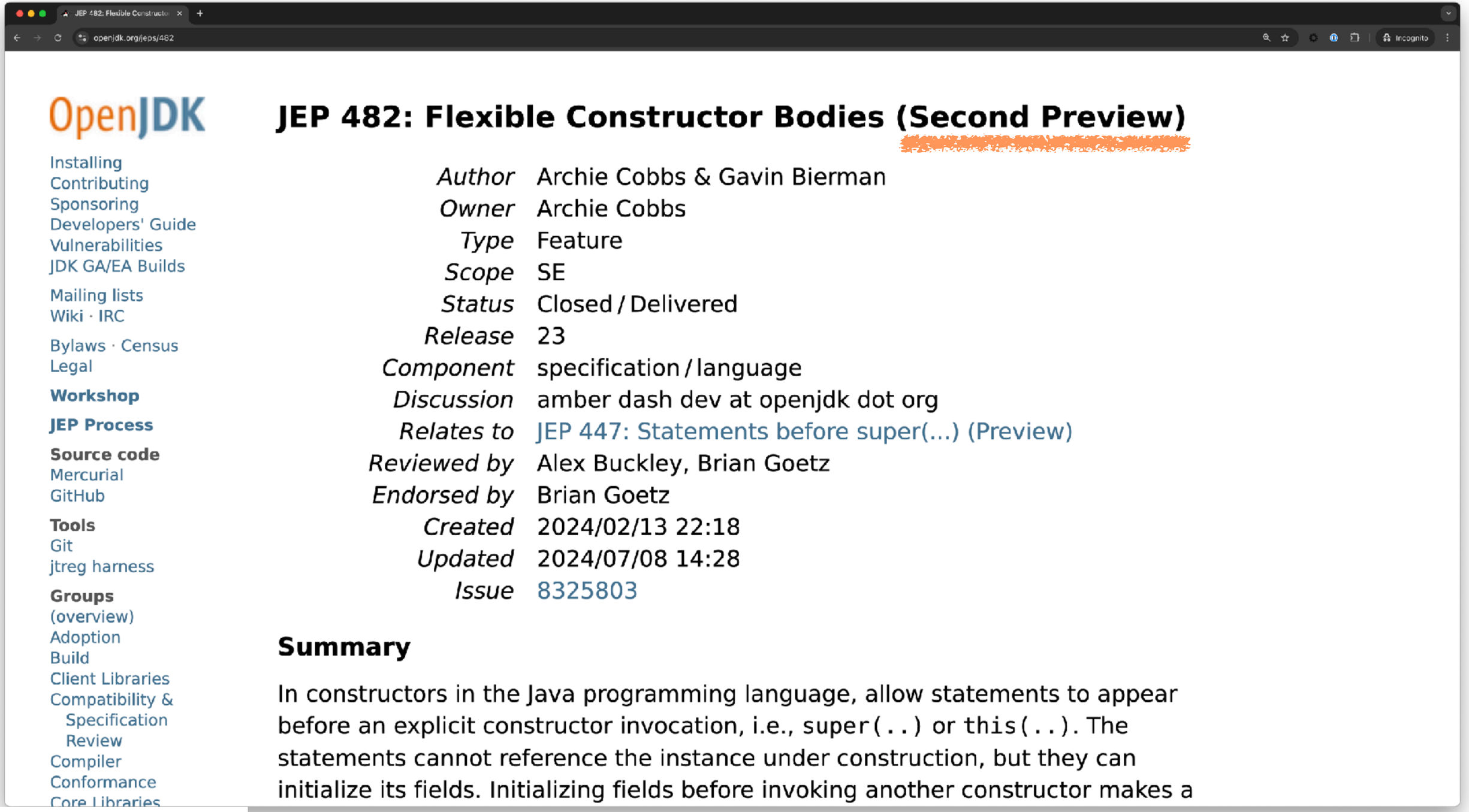
https://openjdk.org/jeps/467

```
/// This is *important* javadoc comment
/// ```
/// Example code looks like that
///
/// public static void main(String args[]) {...}
/// ```
///
/// Lists are possible:
///
/// - item
/// - item2
///
/// And links to e.g. [String][java.lang.String] are possible, too.
public class MarkdownJavadoc {
}
```



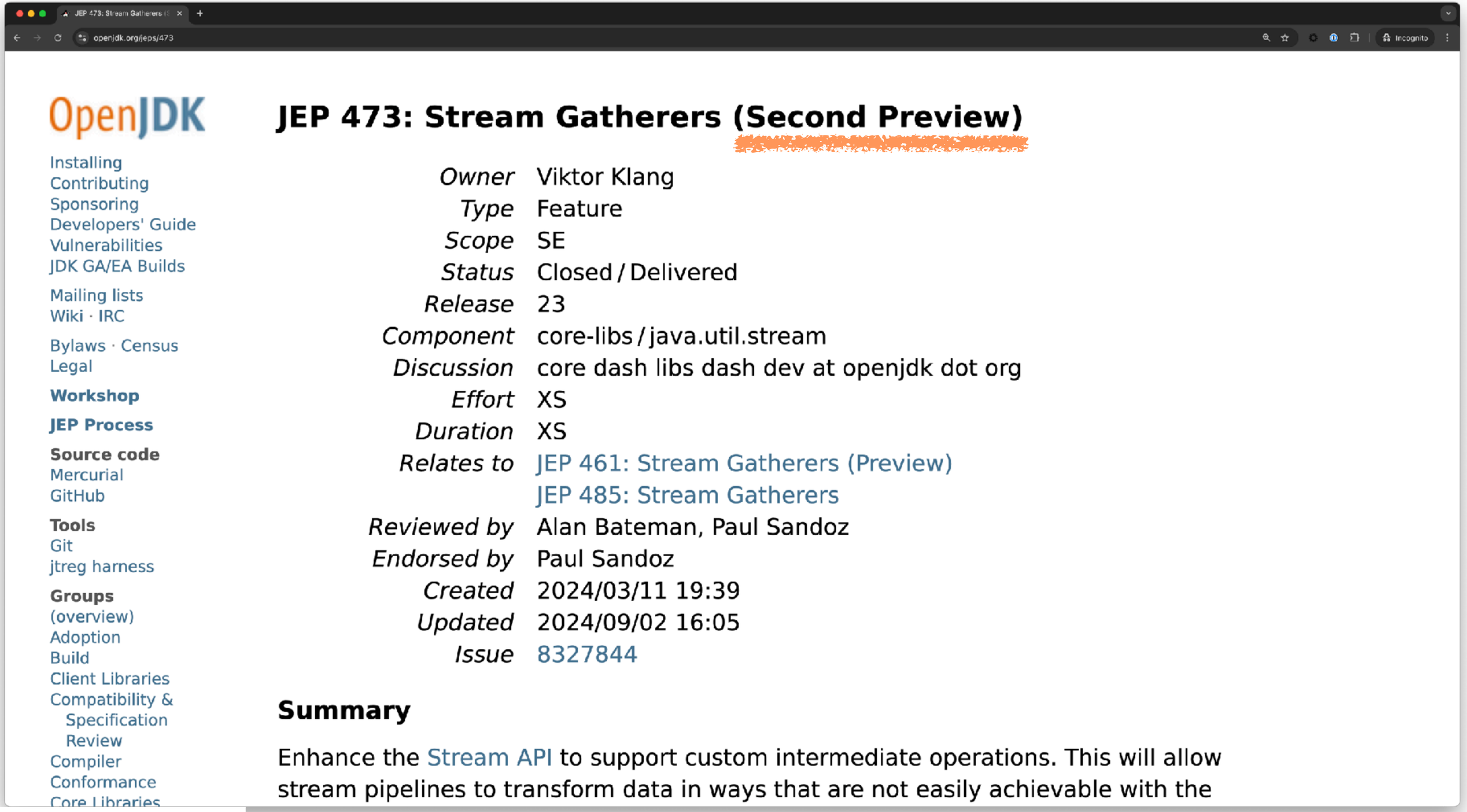
https://openjdk.org/jeps/455

```java
Object o = 42;

var result = switch (o) {
    case int i when i > 50 -> "Found a big number doubled to: " + i * 2;
    case int i -> "Found a number: " + i;
    default -> "Some object";
};
System.out.println(result); // Found a number: 42

record JsonNumber(double value) {}
var json = new JsonNumber(42.0d);

result = switch (json) {
    case JsonNumber(int i) -> "Integer " + i;
    case JsonNumber(double d) -> "Double " + d;
};
System.out.println(result); // Integer 42

long l = 127L;
if (l instanceof byte b) {
    System.out.println("Byte " + b); // Byte 127
}
```



https://openjdk.org/jeps/482

```
public class FlexibleConstructorBodies {

    static class Parent {
        private final int i;

        protected Parent(int i) {
            this.i = i;
        }
    }

    static class Child extends Parent {

        Child(int i) {
            if (i > 42) {
                throw new RuntimeException();
            }
            super(i);
        }
    }
}
```

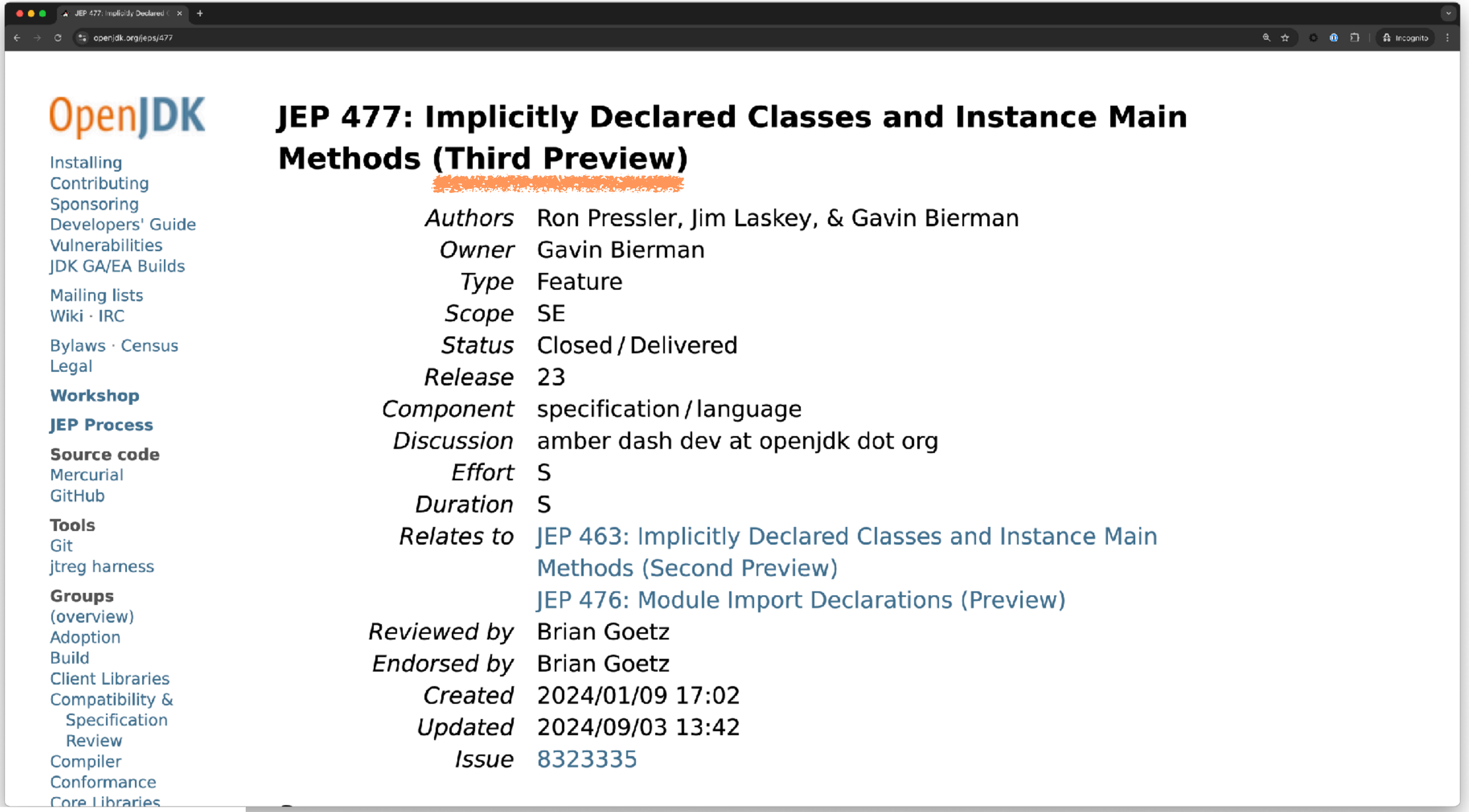


https://openjdk.org/jeps/473

```java
public class Gatherers {

    public static void main(String[] args) {
        Stream.of(2, 1, 5, 3)
                .gather(map((Integer element) -> element * 2).andThen(filter(element -> element > 2)))
                .forEach(System.out::println);


    static <T, R> Gatherer<T, Void, R> map(Function<T, R> function) {
        return Gatherer.of(
                (_, element, downstream) ->
                    downstream.push(function.apply(element)));
    }

    static <T> Gatherer<T, Void, T> filter(Predicate<T> predicate) {
        return Gatherer.of(
                (_, element, downstream) -> {
                    if (!predicate.test(element)) {
                        return downstream.isRejecting();
                    }
                    return downstream.push(element);
                });
    }
}
```

```java
public class MaxGatherer {

    public static void main(String[] args) {
        Stream.of(2, 1, 5, 3)
                .gather(max())
                .findFirst()
                .ifPresent(System.out::println);
    }
}
```

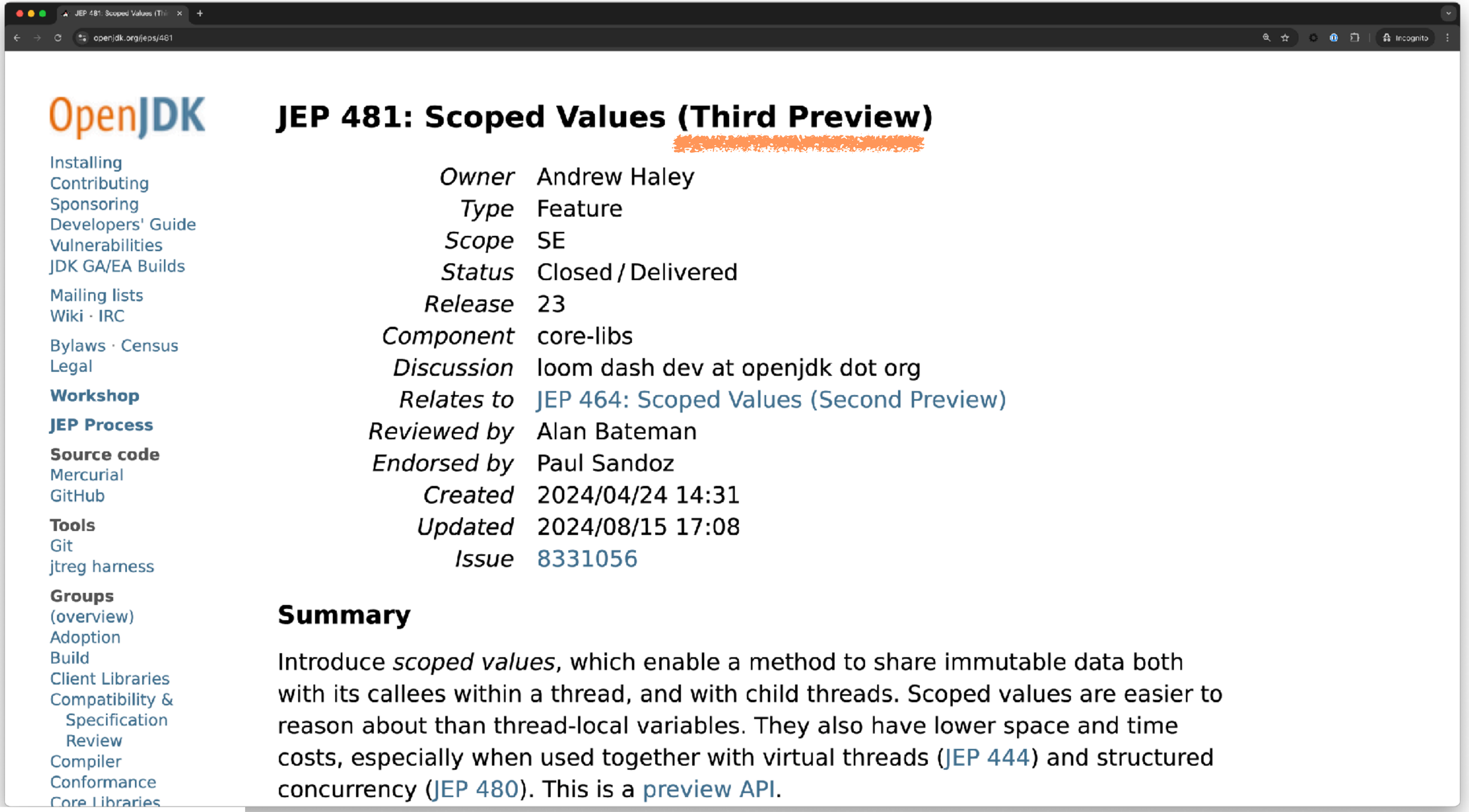
```java
static Gatherer<Integer, ?, Integer> max() {
    class State {
        Integer value;
        boolean hasValue;
    }

    return Gatherer.of(
            State::new,
            Gatherer.Integrator.ofGreedy((state, element, _) -> {
                if (!state.hasValue) {
                    state.value = element;
                    state.hasValue = true;
                } else {
                    state.value = Math.max(state.value, element);
                }
                return true;
            }),
            (left, right) -> {
                if (!left.hasValue) {
                    return right;
                } else if (!right.hasValue) {
                    return left;
                } else {
                    left.value = Math.max(left.value, right.value);
                    return left;
                }
            },
            (state, downstream) -> {
                if (state.hasValue) {
                    downstream.push(state.value);
                }
            });
}
```



https://openjdk.org/jeps/477

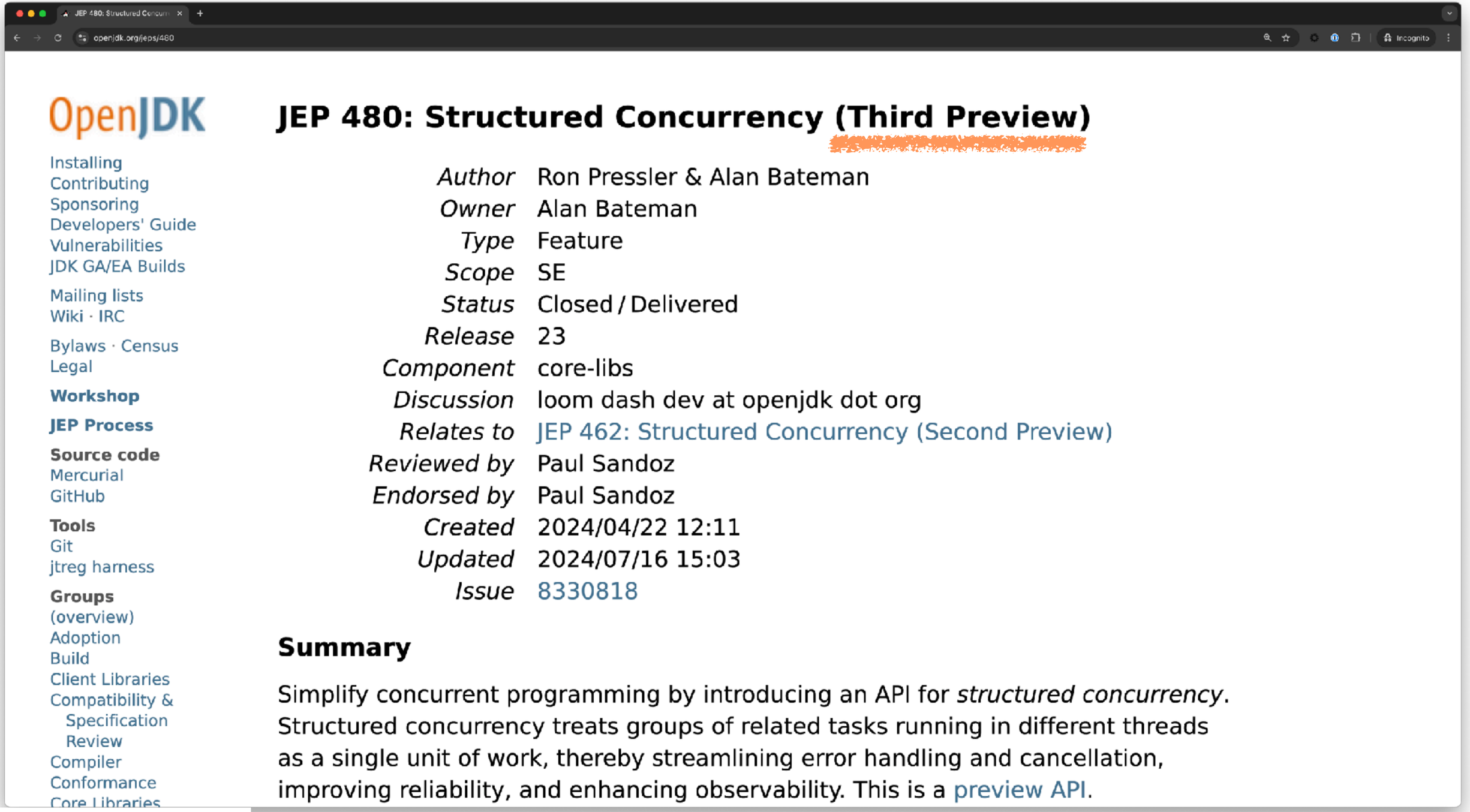
```
public class Hello {

    public static void main(String[] args) {
      System.out.println("Hello");
    }
}
```

```
void main() {
    IO.println("Hello");
}
```



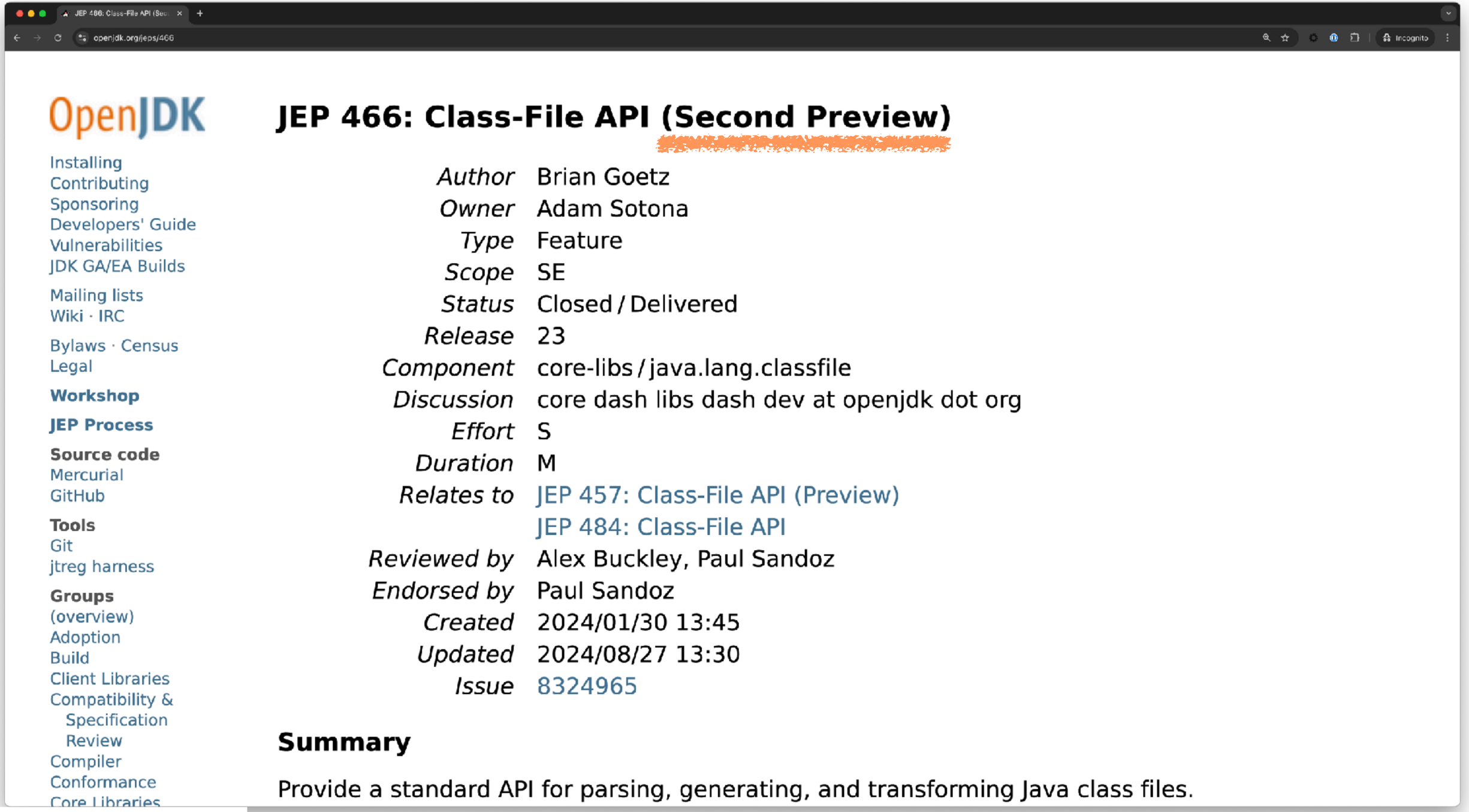
https://openjdk.org/jeps/481

```
ScopedValue<Integer> FOO = ScopedValue.newInstance();

ScopedValue.runWhere(FOO, 42, () -> {
    System.out.println(FOO.get()); // can be done anywhere within the call stack
});
```



https://openjdk.org/jeps/480

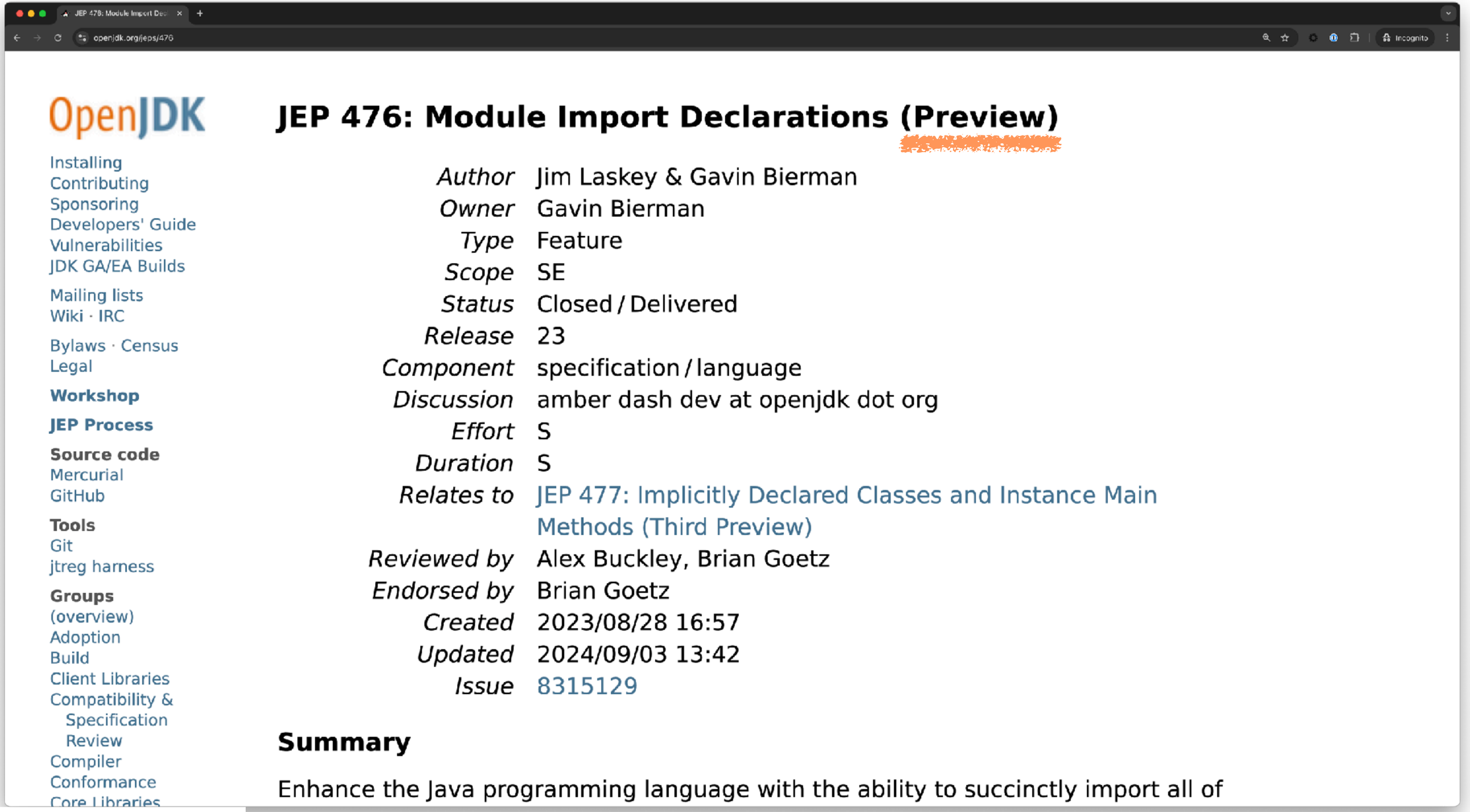
```
try (var scope = new StructuredTaskScope.ShutdownOnFailure()) {
    scope.fork(() -> ... );
    scope.fork(() -> ... );
    scope.fork(() -> ... );
    scope.join().throwIfFailed(); // waits until all three finished or one throwed exception
}
```



https://openjdk.org/jeps/466

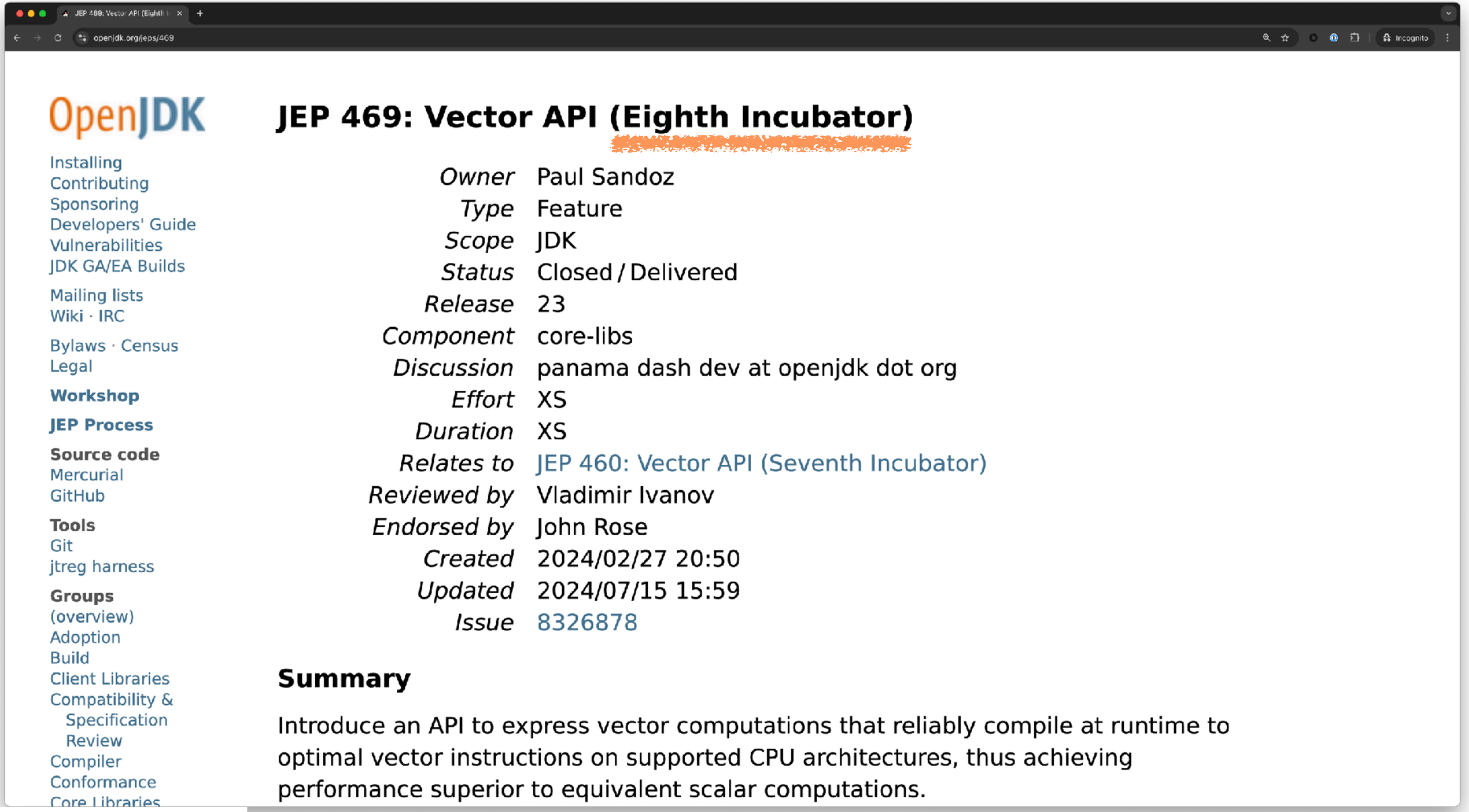
```
try (var in = String.class.getResourceAsStream("/java/lang/String.class")) {
    var classModel = ClassFile.of().parse(in.readAllBytes());

    System.out.println("## Methods");
    classModel.methods().stream()
            .map(method -> method.methodName().stringValue())
            .map(methodName -> " * " + methodName)
            .forEach(System.out::println);

    System.out.println("## Fields");
    classModel.fields().stream()
            .map(field -> field.fieldName().stringValue())
            .map(methodName -> " * " + methodName)
            .forEach(System.out::println);
}
```

```
public class EchoGenerator {

    public static void main(String[] args) throws IOException {
        var system = of("java.lang", "System");
        var printStream = of("java.io", "PrintStream");
        ClassFile.of().buildTo(
            Path.of("Echo.class"),
            of("Echo"),
            classBuilder -> classBuilder
                .withMethodBody(
                   "main",
                   MethodTypeDesc.of(CD_void, CD_String.arrayType()),
                   ACC_PUBLIC | ACC_STATIC,
                   codeBuilder -> codeBuilder
                      .getstatic(system, "out", printStream)
                      .aload(codeBuilder.parameterSlot(0))
                      .iconst_0()
                      .aaload()
                      .invokevirtual(printStream, "println", MethodTypeDesc.of(CD_void, CD_String))
                      .return_()));
    }
}
```



https://openjdk.org/jeps/476

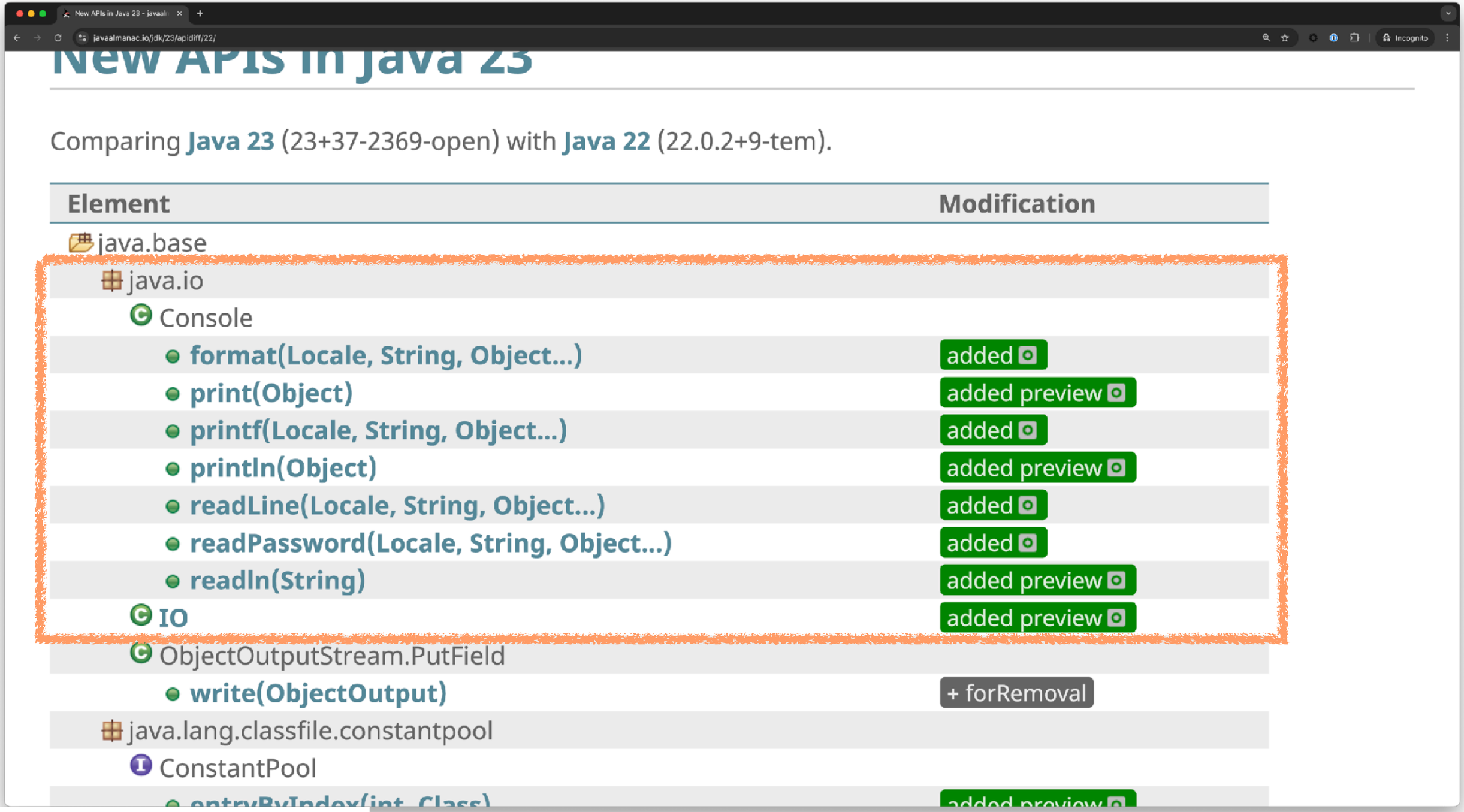
```
import module java.base;

// no imports for e.g. java.util.* needed

public class ModuleImport {

    public static void main(String[] args) {
      List.of();
      new HashMap<String, String>();
    }
}
```



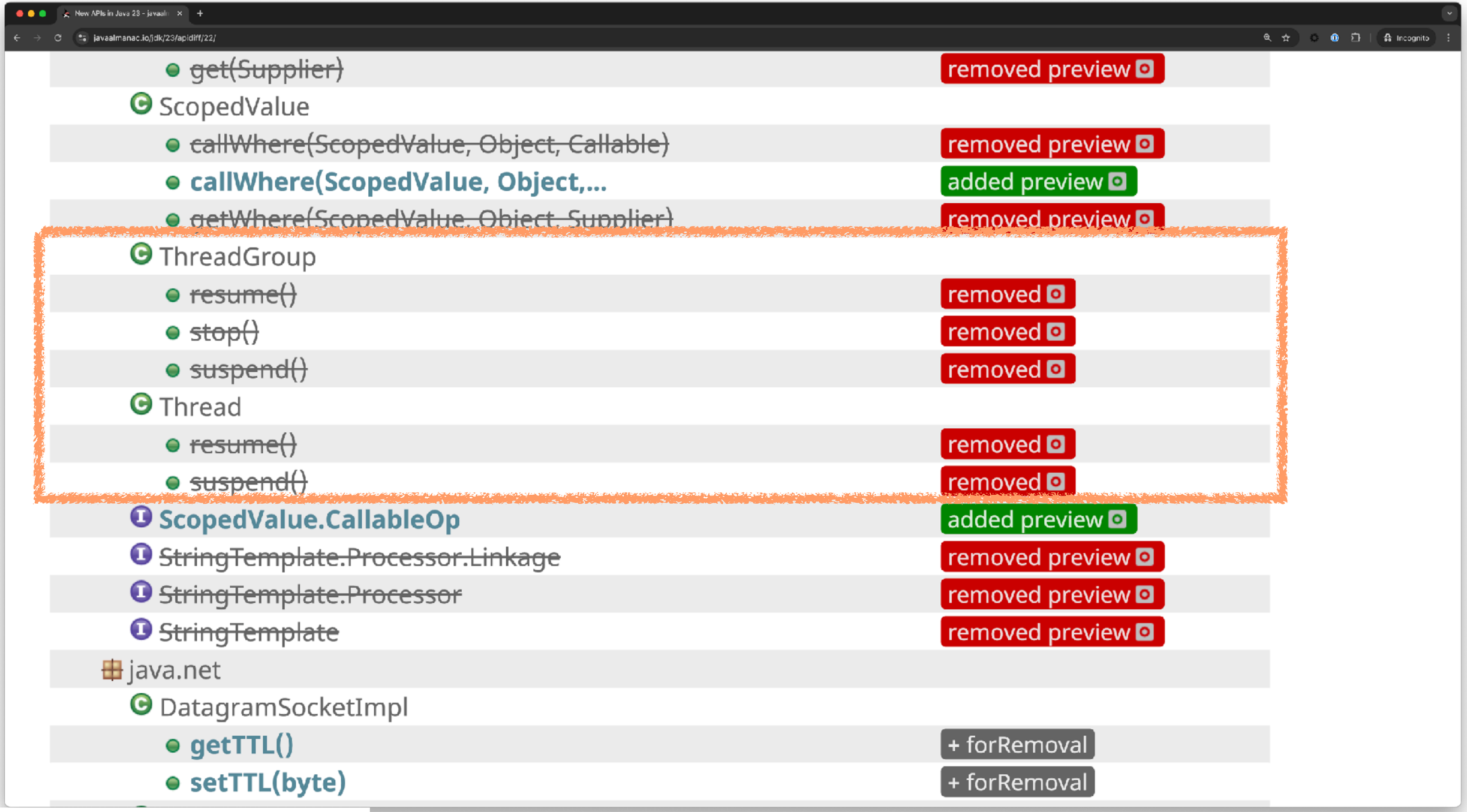
https://openjdk.org/jeps/469

API Changes

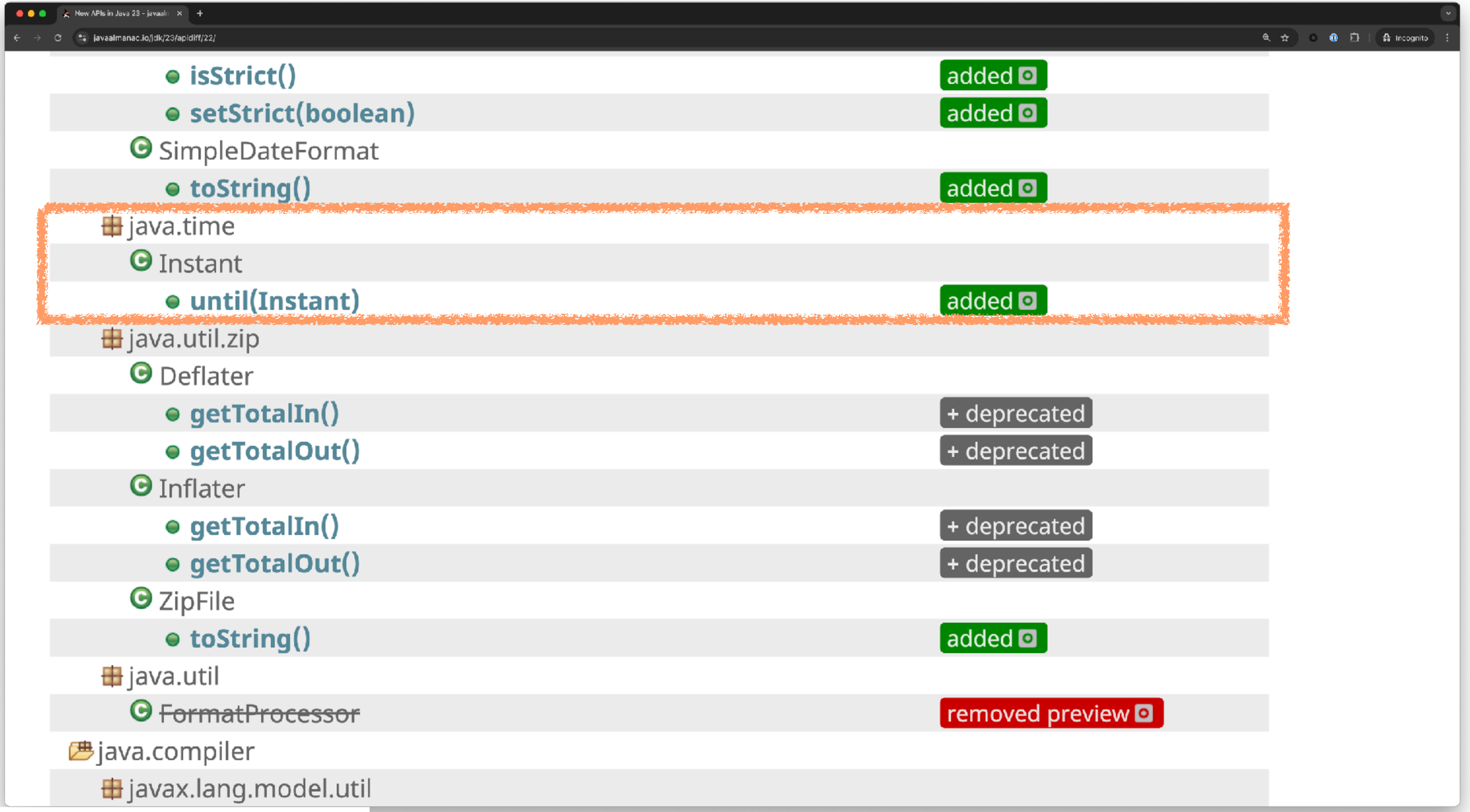
https://javaalmanac.io/jdk/23/apidiff/22/



https://javaalmanac.io/jdk/23/apidiff/22/

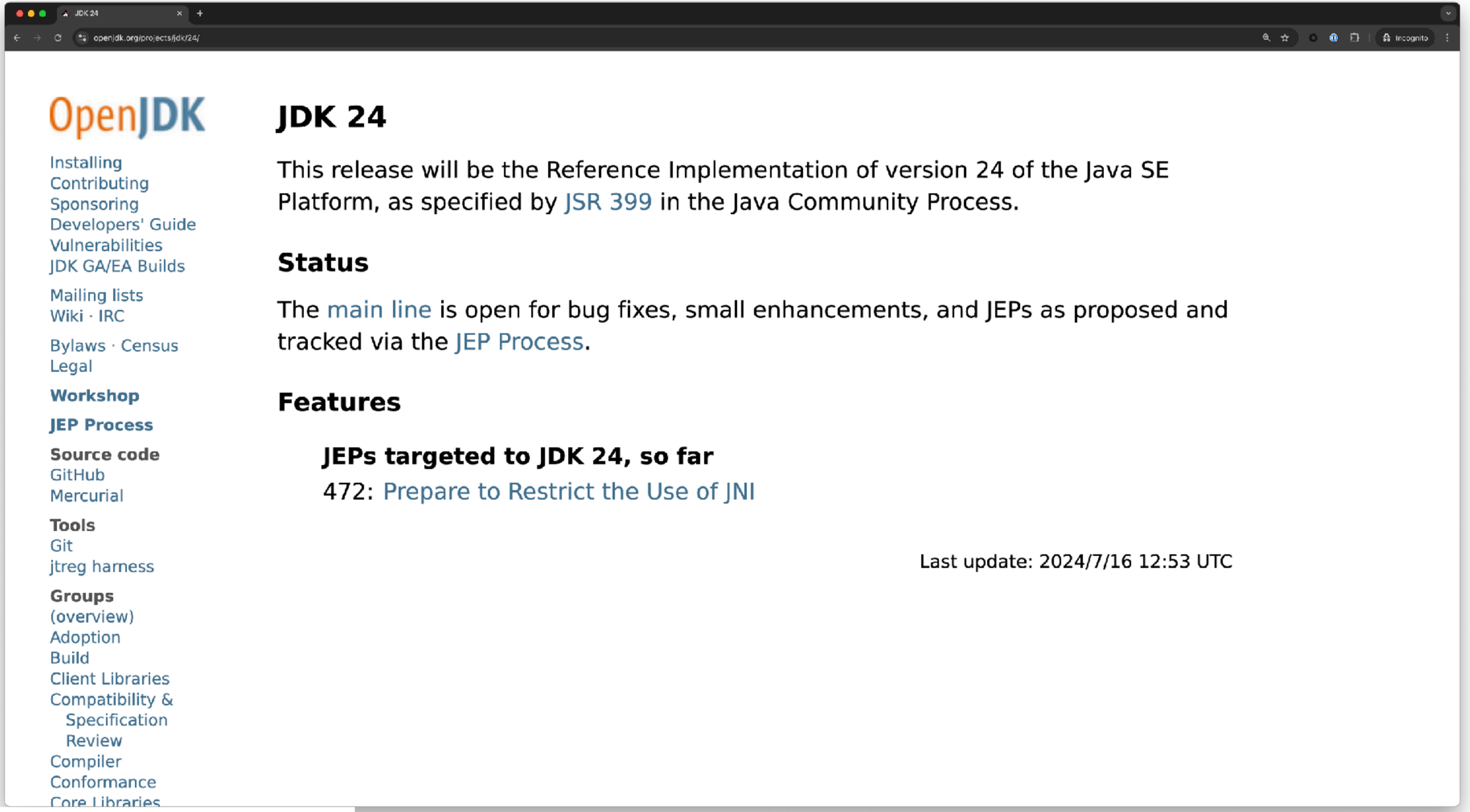


https://javaalmanac.io/jdk/23/apidiff/22/

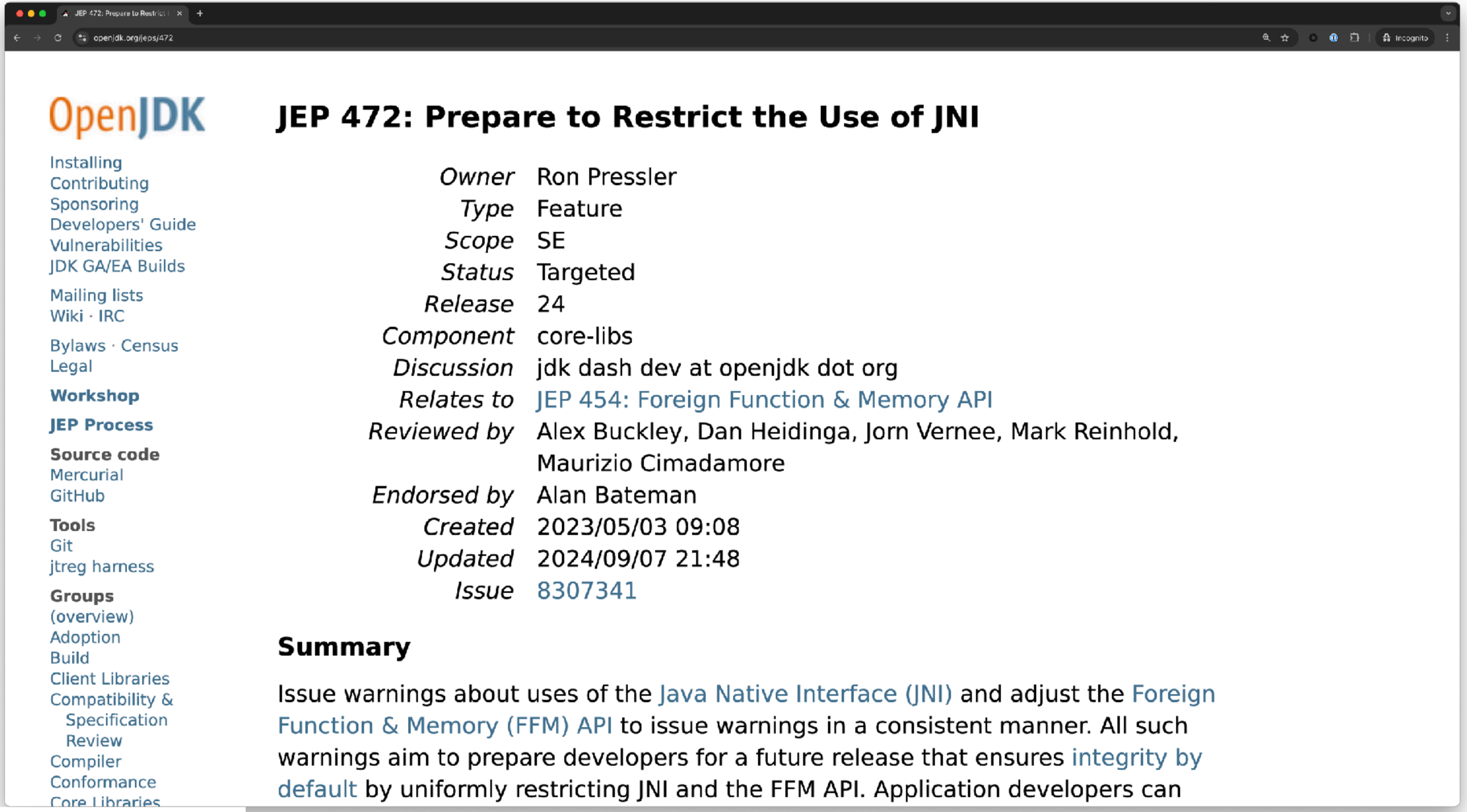


https://javaalmanac.io/jdk/23/apidiff/22/

JDK 24

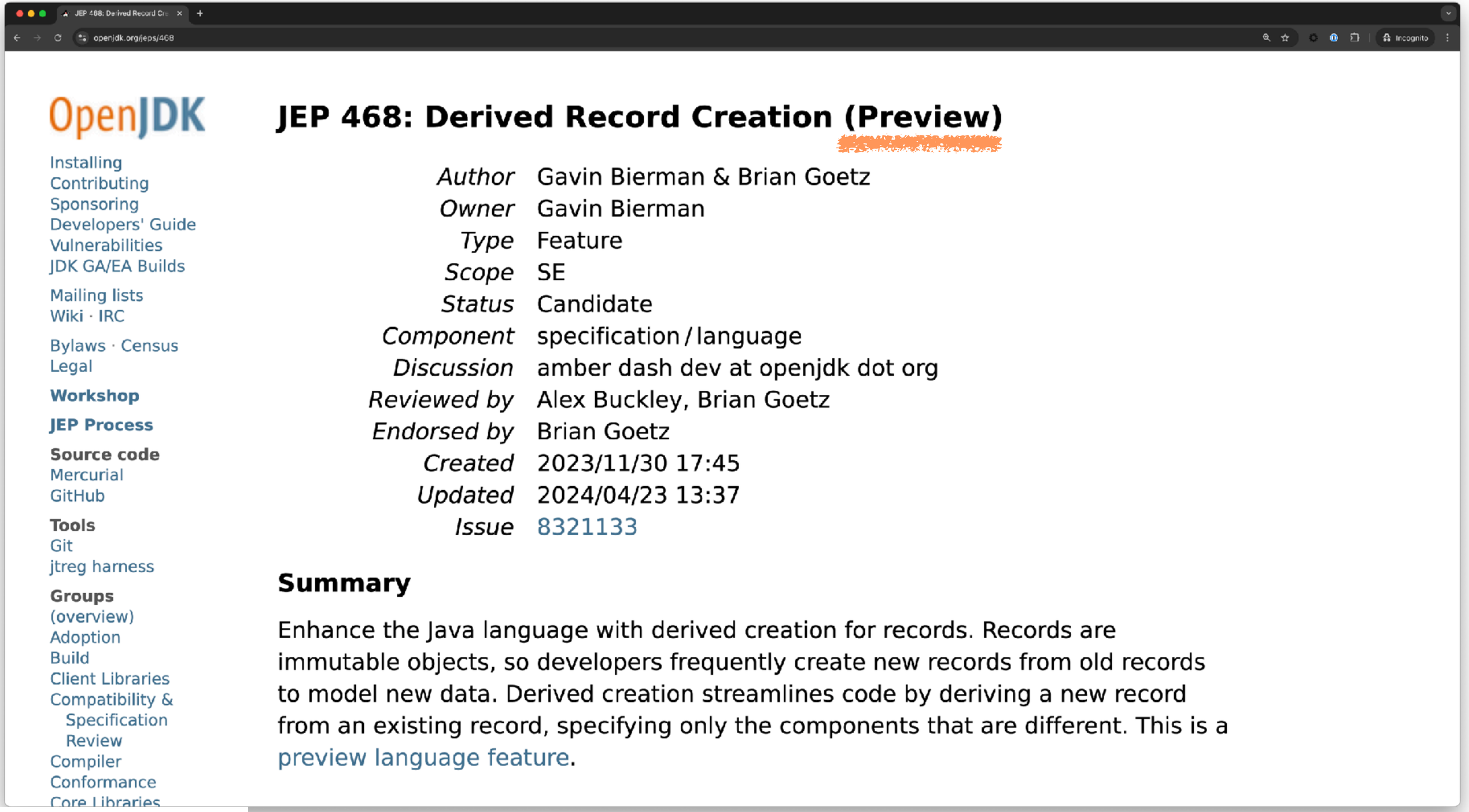


https://openjdk.java.net/projects/jdk/24



https://openjdk.org/jeps/472

Future



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

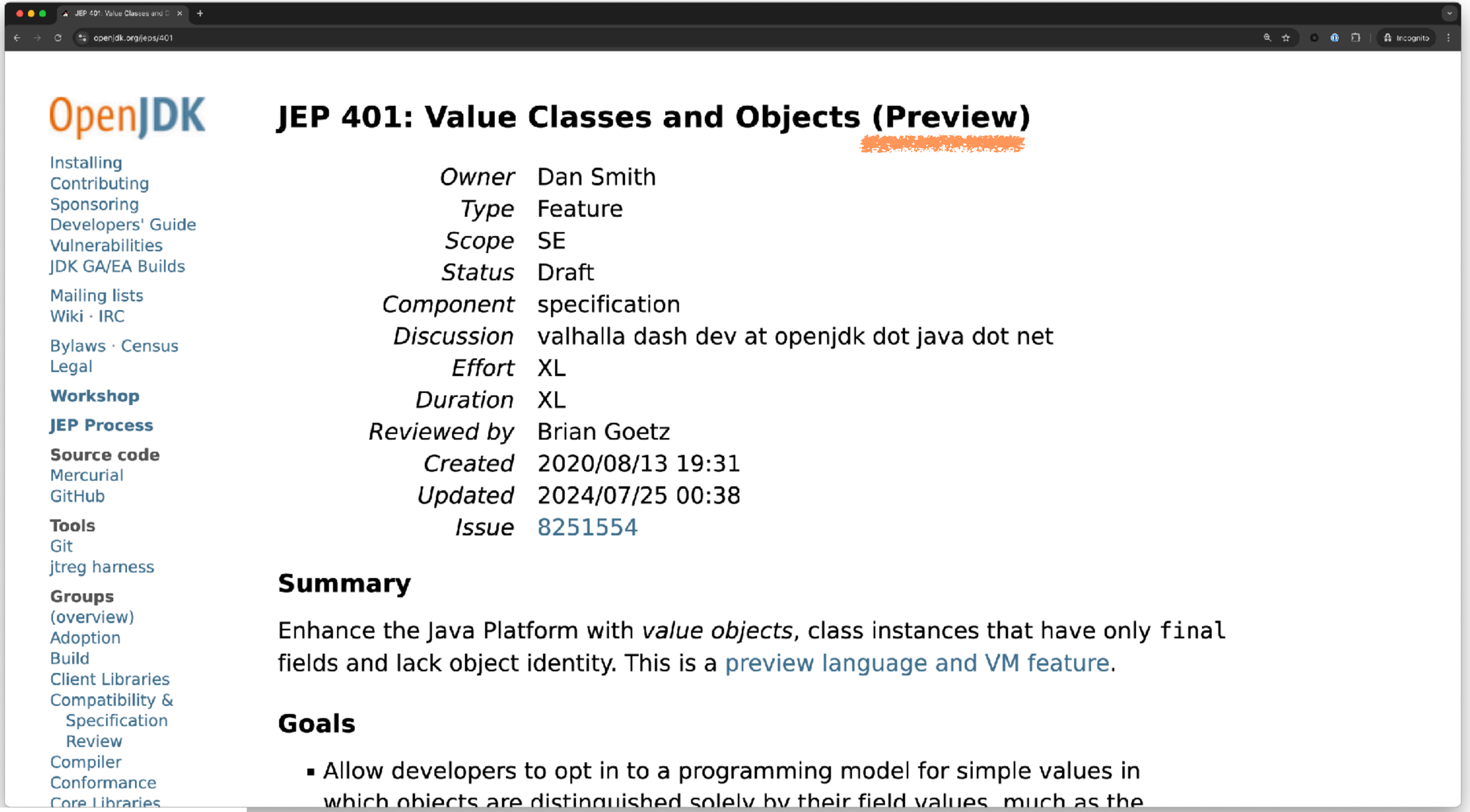
JEP 468: Derived Record Creation (Preview)

Author	Gavin Bierman & Brian Goetz
Owner	Gavin Bierman
Type	Feature
Scope	SE
Status	Candidate
Component	specification / language
Discussion	amber dash dev at openjdk dot org
Reviewed by	Alex Buckley, Brian Goetz
Endorsed by	Brian Goetz
Created	2023/11/30 17:45
Updated	2024/04/23 13:37
Issue	8321133

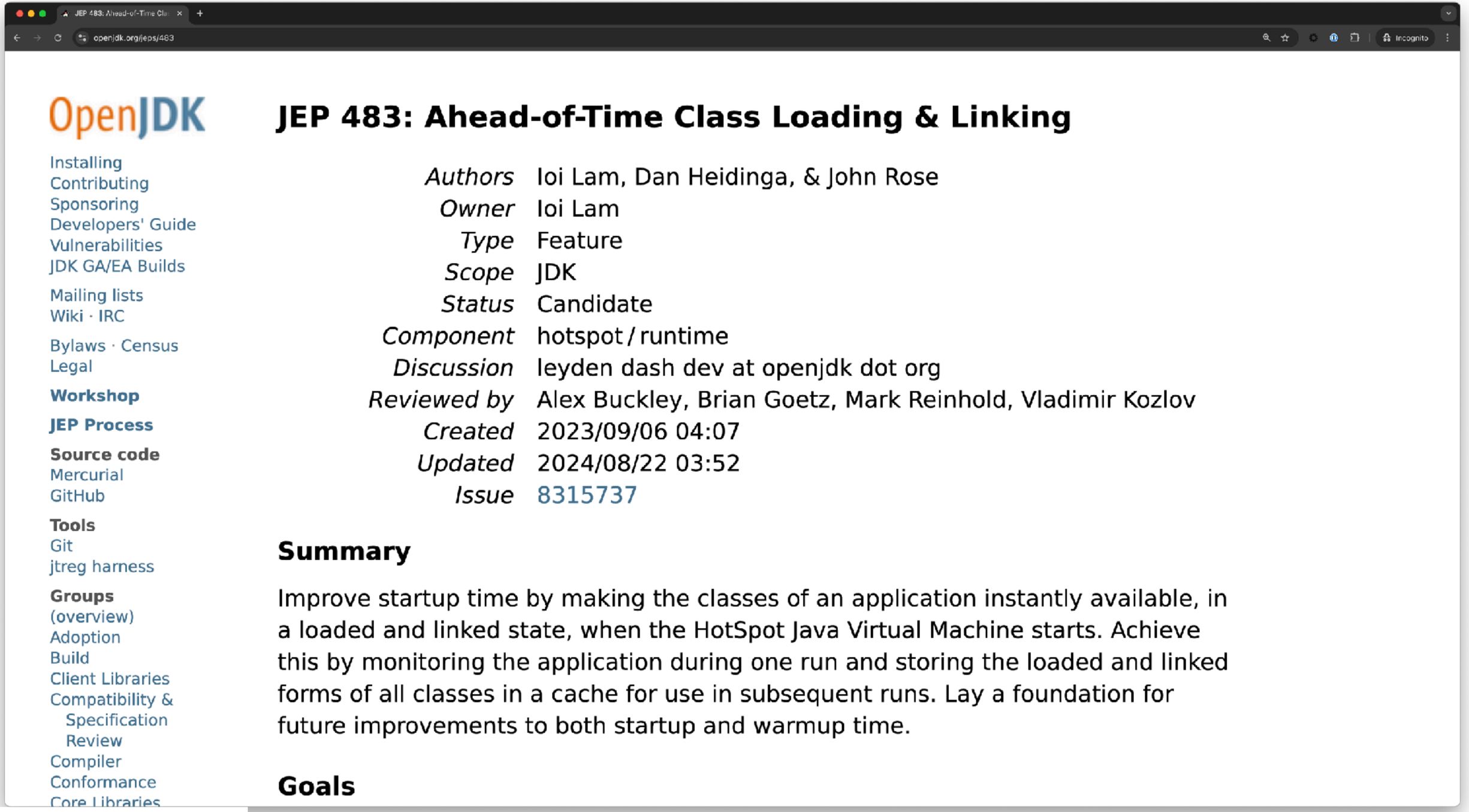
Summary

Enhance the Java language with derived creation for records. Records are immutable objects, so developers frequently create new records from old records to model new data. Derived creation streamlines code by deriving a new record from an existing record, specifying only the components that are different. This is a preview language feature.

https://openjdk.org/jeps/468



https://openjdk.org/jeps/401



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 483: Ahead-of-Time Class Loading & Linking

Authors	Ioi Lam, Dan Heidinga, & John Rose
Owner	Ioi Lam
Type	Feature
Scope	JDK
Status	Candidate
Component	hotspot / runtime
Discussion	leyden dash dev at openjdk dot org
Reviewed by	Alex Buckley, Brian Goetz, Mark Reinhold, Vladimir Kozlov
Created	2023/09/06 04:07
Updated	2024/08/22 03:52
Issue	8315737

Summary

Improve startup time by making the classes of an application instantly available, in a loaded and linked state, when the HotSpot Java Virtual Machine starts. Achieve this by monitoring the application during one run and storing the loaded and linked forms of all classes in a cache for use in subsequent runs. Lay a foundation for future improvements to both startup and warmup time.

Goals

https://openjdk.org/jeps/483

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: Null-Restricted and Nullable Types (Preview)

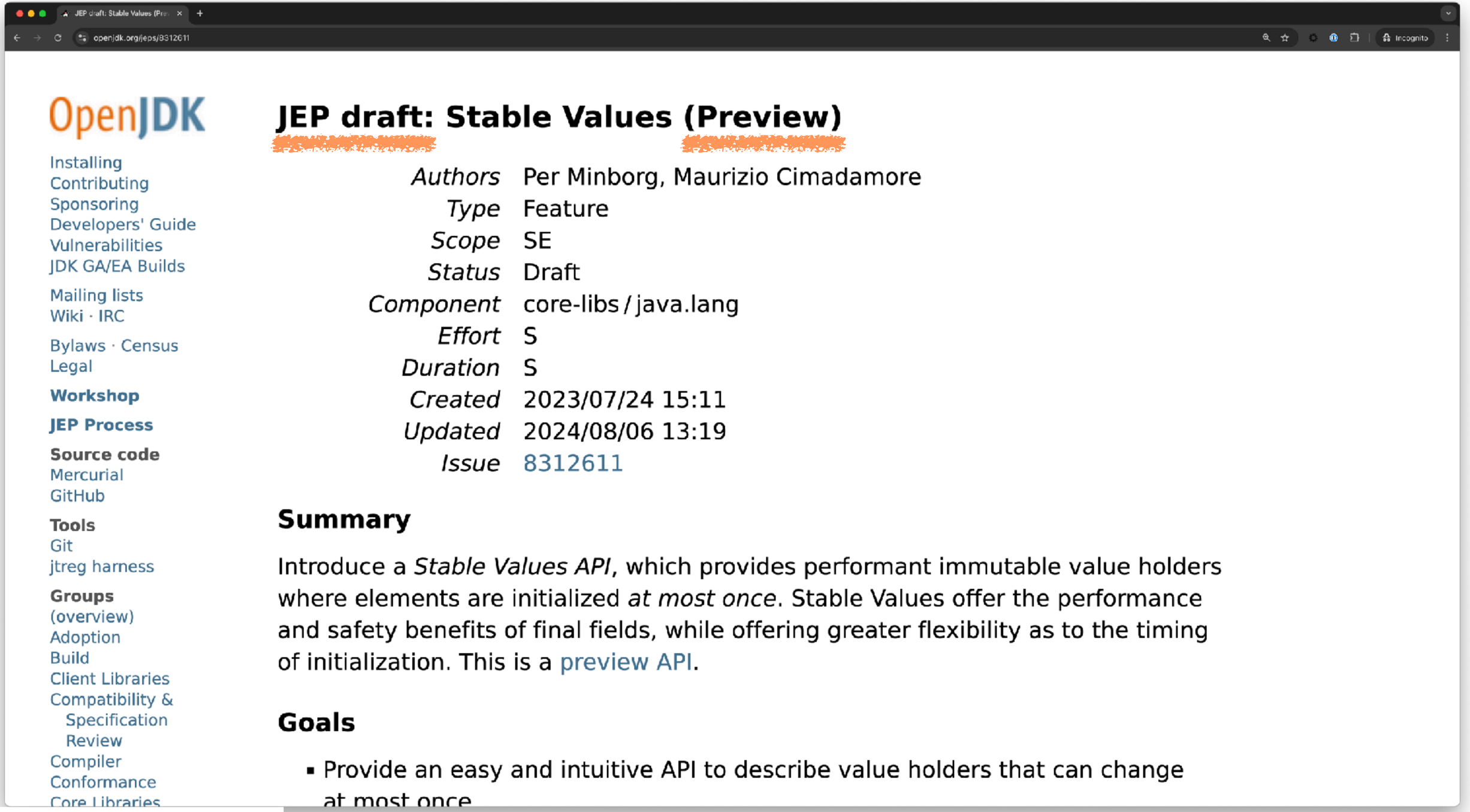
Owner	Dan Smith
Type	Feature
Scope	SE
Status	Draft
Component	tools / javac
Discussion	valhalla dash dev at openjdk dot org
Effort	L
Duration	M
Created	2023/02/23 01:23
Updated	2024/08/20 20:19
Issue	8303099

Summary

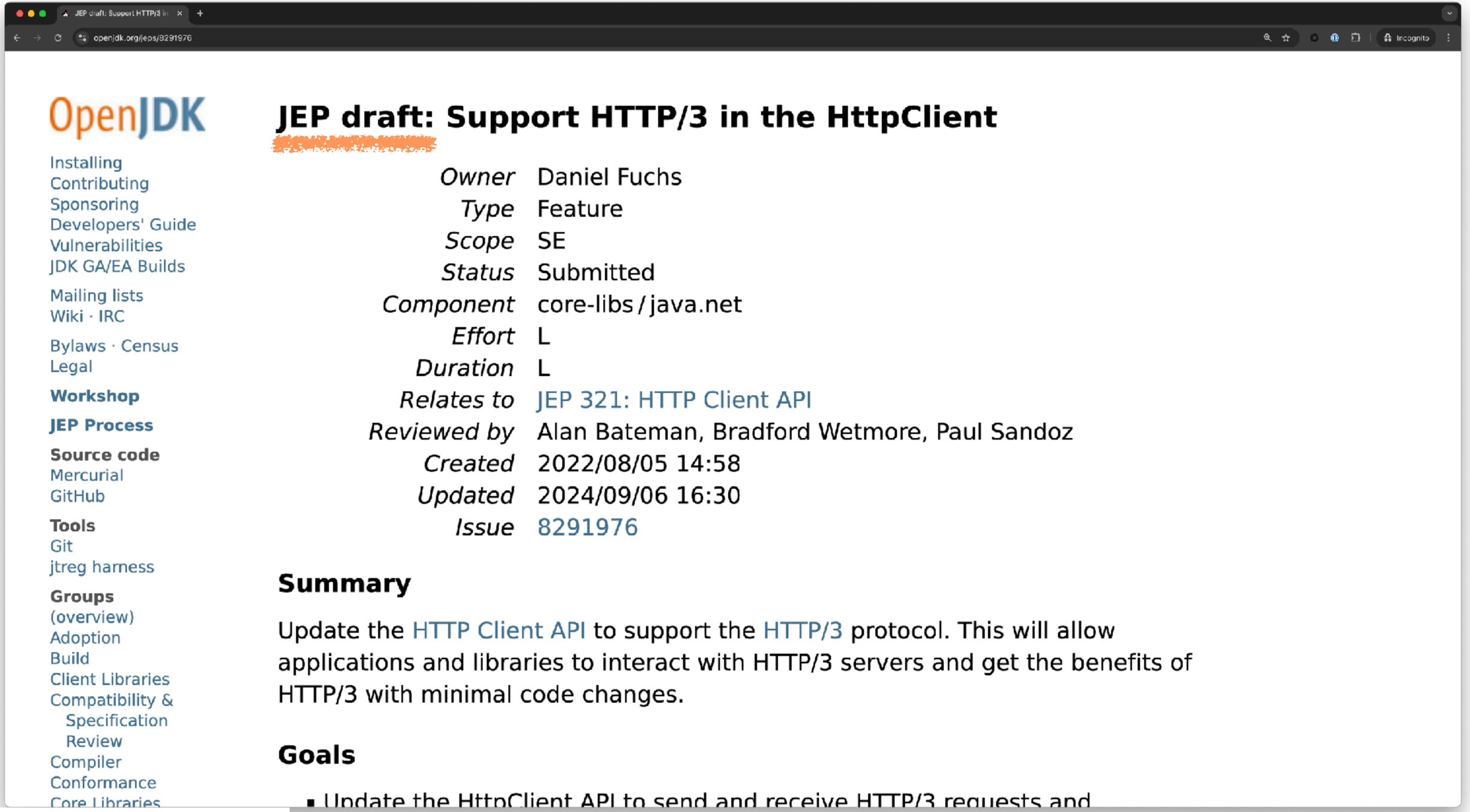
Support *nullness markers* on Java types to indicate that a type rejects or deliberately allows nulls. This is a preview language feature.

Goals

- Enhance Java's reference types to let programmers express whether `null` references are expected as values of the type



https://openjdk.org/jeps/8312611



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: Support HTTP/3 in the HttpClient

Owner	Daniel Fuchs
Type	Feature
Scope	SE
Status	Submitted
Component	core-libs / java.net
Effort	L
Duration	L
Relates to	JEP 321: HTTP Client API
Reviewed by	Alan Bateman, Bradford Wetmore, Paul Sandoz
Created	2022/08/05 14:58
Updated	2024/09/06 16:30
Issue	8291976

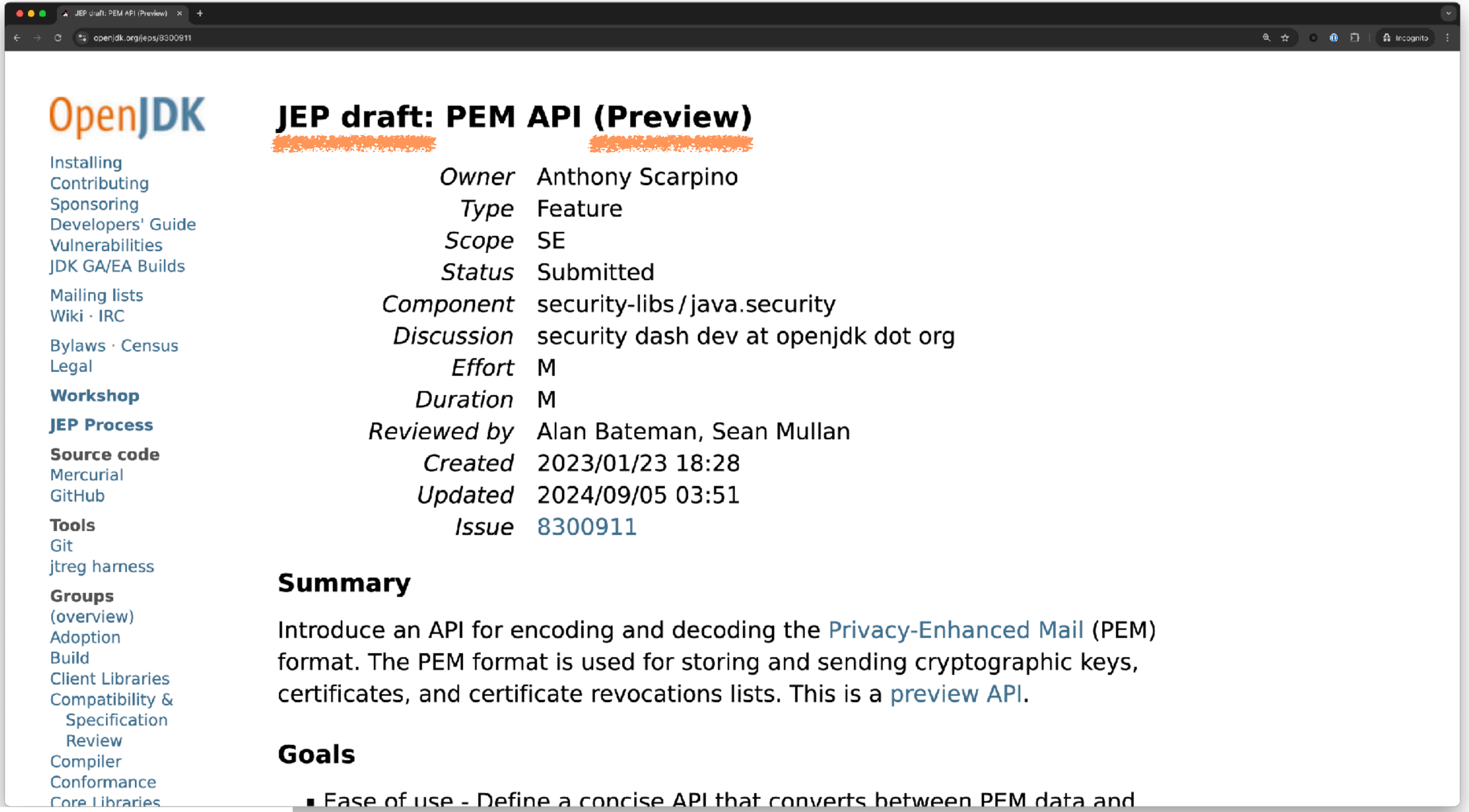
Summary

Update the HTTP Client API to support the HTTP/3 protocol. This will allow applications and libraries to interact with HTTP/3 servers and get the benefits of HTTP/3 with minimal code changes.

Goals

- Update the HttpClient API to send and receive HTTP/3 requests and

https://openjdk.org/jeps/8291976



https://openjdk.org/jeps/8300911

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	2024/06/27 08:37
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 compile 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

https://openjdk.org/jeps/8313278

Resources

Resources

- German article by me:
  https://www.innoq.com/de/articles/2024/03/jdk-zukunft-features/
- What Happened to Java's String Templates?
  https://nipafx.dev/inside-java-newscast-71/
- Brainf*ck Compiler with Class-File API:
  https://jameshamilton.eu/programming/build-compiler-jep-457-class-file-api
- 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/de/talks/2024/09/java-forum-nord-2024-neues-vom-jdk/