Java 8, Java 9, and Beyond!

Mark Reinhold (@mreinhold)

Chief Architect, Java Platform Group, Oracle

Jfokus 2013

Java 8

Lambda (JSR 335)

Java 8

Lambda (JSR 335)

Java 8

Compact Profiles

Lambda (JSR 335)

Java 8

Compact Profiles

Nashorn

Lambda (JSR 335)

Date/Time API (JSR 310)

Java 8

Compact Profiles

Nashorn

Lambda (JSR 335)

Date/Time API (JSR 310)

Java 8

Compact Profiles

Nashorn

Type Annotations (JSR 308)

NSA Suite B

Base64

HTTP Client

Bulk Data Operations

Unicode 6.2

Prepare for Modularization

Lambda (JSR 335)

Remove the Permanent Generation

Generalized Target-Type Inference

Date/Time API (JSR 310)

Improve Contended Locking

Java 8

Unicode CLDR

Parallel Array Sorting

Compact Profiles

DocTree API

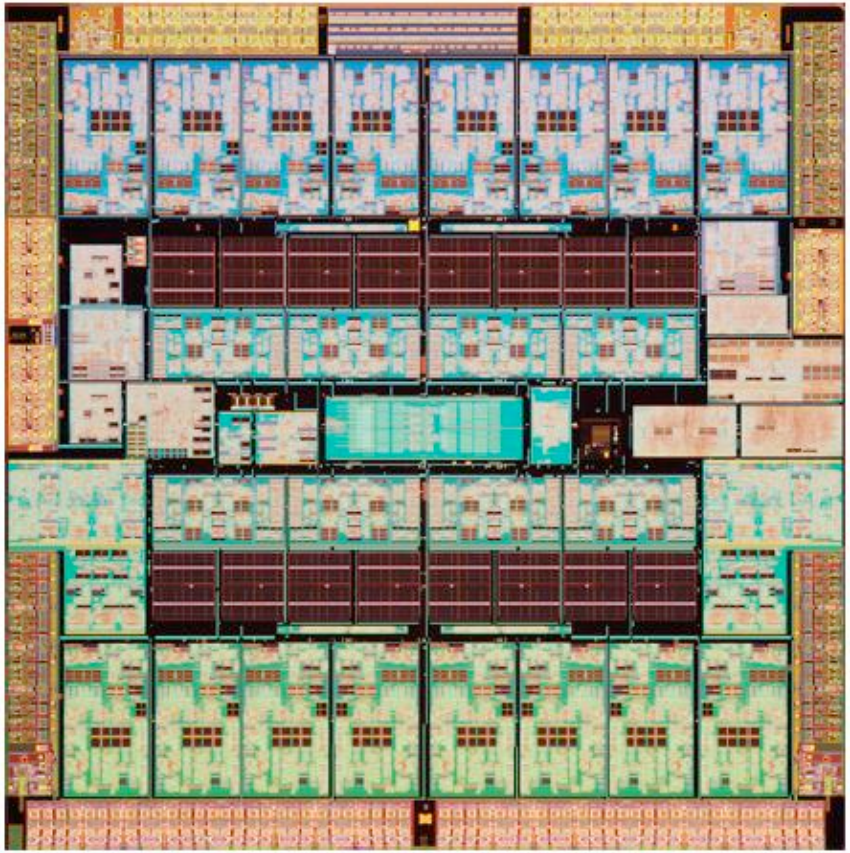
Nashorn

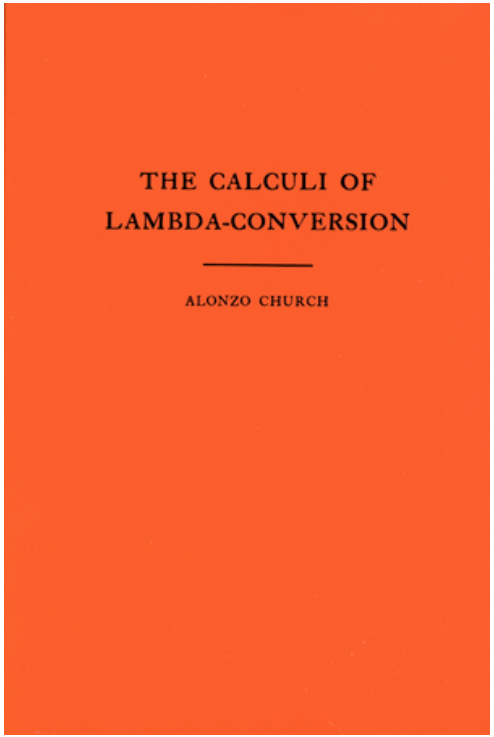
Configurable Secure-Random Number Generation

TLS Server Name Indication

Type Annotations (JSR 308)

Lambda-Form Representation for Method Handles





Colors and Shapes

Colors and Shapes

```
enum Color { RED, GREEN, BLUE }
```


Colors and Shapes

```
enum Color { RED, GREEN, BLUE }

class Shape {

  Shape() { ... }
  Shape(Shape s, double area) { ... }

  Color getColor() { ... }
  void setColor(Color c) { ... }

  double getArea() { ... }

}
```



```
List<Shape> shapes = ...;
```



```
List<Shape> shapes = ...;

// Sort shapes by color
Comparator<Shape> c = new Comparator<Shape>() {
  public int compare(Shape s, Shape t) {
    return s.getColor().compareTo(t.getColor());
  }
};
Collections.sort(shapes, c);
```



```
// Sort shapes by color
Comparator<Shape> c = new Comparator<Shape>() {
  public int compare(Shape s, Shape t) {
    return s.getColor().compareTo(t.getColor());
  }
};
Collections.sort(shapes, c);
```


Lambda expressions

```
// Sort shapes by color
Comparator<Shape> c = new Comparator<Shape>() {
  public int compare(Shape s, Shape t) {
    return s.getColor().compareTo(t.getColor());
  }
};
Collections.sort(shapes, c);

// Sort shapes by color, with a lambda
Comparator<Shape> c = (Shape s, Shape t)
                      -> s.getColor().compareTo(t.getColor());
Collections.sort(shapes, c);
```


Lambda expressions

```
// Sort shapes by color, with a lambda
Comparator<Shape> c = (Shape s, Shape t)
                        -> s.getColor().compareTo(t.getColor());
Collections.sort(shapes, c);
```

Lambda expressions

```
// Sort shapes by color, with a lambda
Comparator<Shape> c = (Shape s, Shape t)
                      -> s.getColor().compareTo(t.getColor());
Collections.sort(shapes, c);
```

```
// Sort shapes by color, with a lambda using type inference
Comparator<Shape> c = (s, t)
                      -> s.getColor().compareTo(t.getColor());
Collections.sort(shapes, c);
```


Lambda expressions

```
// Sort shapes by color, with a lambda using type inference
Comparator<Shape> c = (s, t)
                      -> s.getColor().compareTo(t.getColor());
Collections.sort(shapes, c);
```


Lambda expressions

```
// Sort shapes by color, with a lambda using type inference
Comparator<Shape> c = (s, t)
                      -> s.getColor().compareTo(t.getColor());
Collections.sort(shapes, c);

// Sort shapes by color, in one statement
Collections.sort(shapes, (s, t) ->
                         s.getColor().compareTo(t.getColor()));
```

Functional interfaces

Functional interfaces

```
// Compare shapes by color
Comparator<Shape> c = (s, t)
                      -> s.getColor().compareTo(t.getColor());
```


Functional interfaces

```
// Compare shapes by color
Comparator<Shape> c = (s, t)
                      -> s.getColor().compareTo(t.getColor());

// Test whether a shape is red
Predicate<Shape> p = s -> s.getColor() == RED;
```


Functional interfaces

```
// Compare shapes by color
Comparator<Shape> c = (s, t)
                      -> s.getColor().compareTo(t.getColor());

// Test whether a shape is red
Predicate<Shape> p = s -> s.getColor() == RED;

// Extract the area of a shape
Function<Shape,Double> f = s -> s.getArea();
```


Functional interfaces

```
// Compare shapes by color
Comparator<Shape> c = (s, t)
                      -> s.getColor().compareTo(t.getColor());

// Test whether a shape is red
Predicate<Shape> p = s -> s.getColor() == RED;

// Extract the area of a shape
Function<Shape,Double> f = s -> s.getArea();

// Create a bigger shape from a given shape
UnaryOperator<Shape> o = s -> new Shape(s, s.getArea() * 2);
```


Iteration

Iteration

```
// Paint red shapes blue
List<Shape> shapes = ...;
for (Shape s : shapes) {
  if (s.getColor() == RED)
    s.setColor(BLUE);
}
```


Iteration

```
// Paint red shapes blue, using external iteration
List<Shape> shapes = ...;
for (Shape s : shapes) {
  if (s.getColor() == RED)
    s.setColor(BLUE);
}
```


Iteration

```
// Paint red shapes blue, using external iteration
List<Shape> shapes = ...;
for (Shape s : shapes) {
  if (s.getColor() == RED)
    s.setColor(BLUE);
}

// Paint red shapes blue, using internal iteration
shapes.forEach(s -> {
    if (s.getColor() == RED)
      s.setColor(BLUE);
  });
```


Default methods

Default methods

```
interface Iterable<T> {

  Iterator<T> iterator();

  default void forEach(Consumer<? super T> consumer) {
    for (T t : this)
      consumer.accept(t);
  }

}
```


Default methods

Default methods

```
interface Collection<E> {
  default boolean removeAll(Predicate<? super E> filter) {
    boolean removed = false;
    Iterator<E> each = iterator();
    while (each.hasNext()) {
      if (filter.test(each.next())) {
        each.remove();
        removed = true;
      }
    }
    return removed;
  }
}
```


Default methods

Default methods

```
interface List<E> {
  ...
  default void replaceAll(UnaryOperator<E> op) {
    final ListIterator<E> li = this.listIterator();
    while (li.hasNext())
      li.set(op.apply(li.next()));
  }
}
```



```
// Paint red shapes blue, using internal iteration
shapes.forEach(s -> {
    if (s.getColor() == RED)
      s.setColor(BLUE);
  });
```


Streams

```
// Paint red shapes blue, using internal iteration
shapes.forEach(s -> {
    if (s.getColor() == RED)
      s.setColor(BLUE);
  });

// Paint red shapes blue, using bulk operations on a stream
shapes.stream()
      .filter(s -> s.getColor() == RED)
      .forEach(s -> { s.setColor(BLUE); });
```


Streams

```
// Paint red shapes blue, using bulk operations on a stream
shapes.stream()
      .filter(s -> s.getColor() == RED)
      .forEach(s -> { s.setColor(BLUE); });
```


Streams

```
// Paint red shapes blue, using bulk operations on a stream
shapes.stream()
      .filter(s -> s.getColor() == RED)
      .forEach(s -> { s.setColor(BLUE); });


// Find the first blue shape
Shape firstBlue
  = shapes.stream()
          .filter(s -> s.getColor() == BLUE)
          .findFirst().get();
```


Streams

Streams

```
// Compute the sum of the areas of blue shapes
double sumOfAreas
  = shapes.stream()
          .filter(s -> s.getColor() == BLUE)
          .map(s -> s.getArea())
          .reduce(0.0, (a, b) -> a + b);
```


Streams

```
// Compute the sum of the areas of blue shapes
double sumOfAreas
  = shapes.stream()
          .filter(s -> s.getColor() == BLUE)
          .map(s -> s.getArea())
          .reduce(0.0, (a, b) -> a + b);

// Compute the sum of the areas of blue shapes
double sumOfAreas
  = shapes.stream()
          .filter(s -> s.getColor() == BLUE)
          .map(s -> s.getArea())
          .sum();
```


Streams

```
// Compute the sum of the areas of blue shapes
double sumOfAreas
  = shapes.stream()
          .filter(s -> s.getColor() == BLUE)
          .map(s -> s.getArea())
          .sum();
```

Streams

```
// Compute the sum of the areas of blue shapes
double sumOfAreas
  = shapes.stream()
          .filter(s -> s.getColor() == BLUE)
          .map(s -> s.getArea())
          .sum();

// Compute the sum of the areas of blue shapes
double sumOfAreas
  = shapes.stream()
          .filter(s -> s.getColor() == BLUE)
          .map(Shape::getArea)
          .sum();
```


Streams

```
// Compute the sum of the areas of blue shapes
double sumOfAreas
  = shapes.stream()
          .filter(s -> s.getColor() == BLUE)
          .map(Shape::getArea)
          .sum();
```


Streams

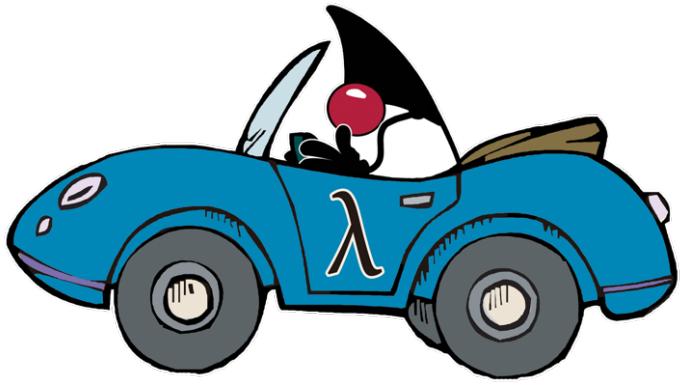
```
// Compute the sum of the areas of blue shapes
double sumOfAreas
  = shapes.stream()
          .filter(s -> s.getColor() == BLUE)
          .map(Shape::getArea)
          .sum();

// Compute the sum of the areas of blue shapes, in parallel
double sumOfAreas
  = shapes.parallelStream()
          .filter(s -> s.getColor() == BLUE)
          .map(Shape::getArea)
          .sum();
```


Project Lambda

openjdk.java.net/projects/lambda
jdk8.java.net/lambda

Project Lambda

openjdk.java.net/projects/lambda
jdk8.java.net/lambda



Project Lambda

openjdk.java.net/projects/lambda
jdk8.java.net/lambda

NSA Suite B

Base64

HTTP Client

Bulk Data Operations

Unicode 6.2

Prepare for Modularization

Lambda (JSR 335)

Remove the Permanent Generation

Generalized Target-Type Inference

Date/Time API (JSR 310)

Improve Contended Locking

Java 8

Unicode CLDR

Parallel Array Sorting

Compact Profiles

DocTree API

Nashorn

Configurable Secure-Random Number Generation

TLS Server Name Indication

Type Annotations (JSR 308)

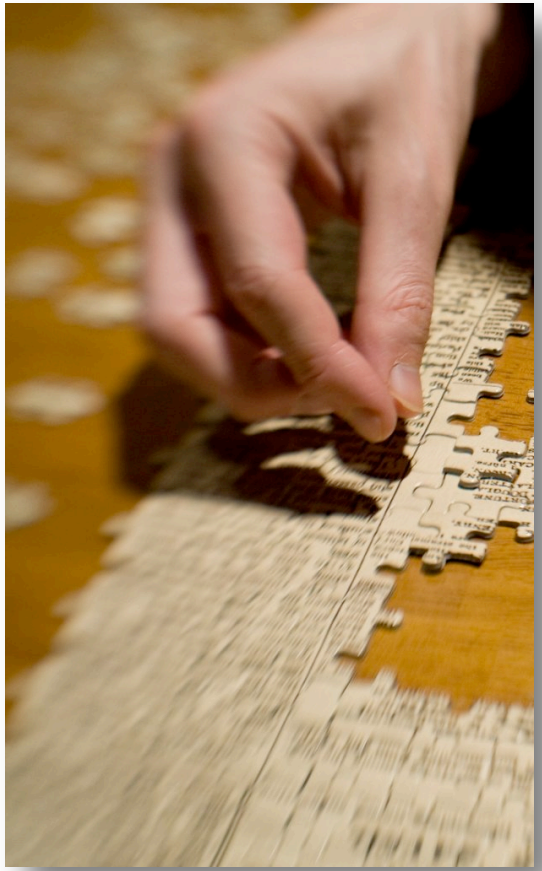
Lambda-Form Representation for Method Handles

Project Jigsaw

openjdk.java.net/projects/jigsaw

Project Jigsaw

openjdk.java.net/projects/jigsaw

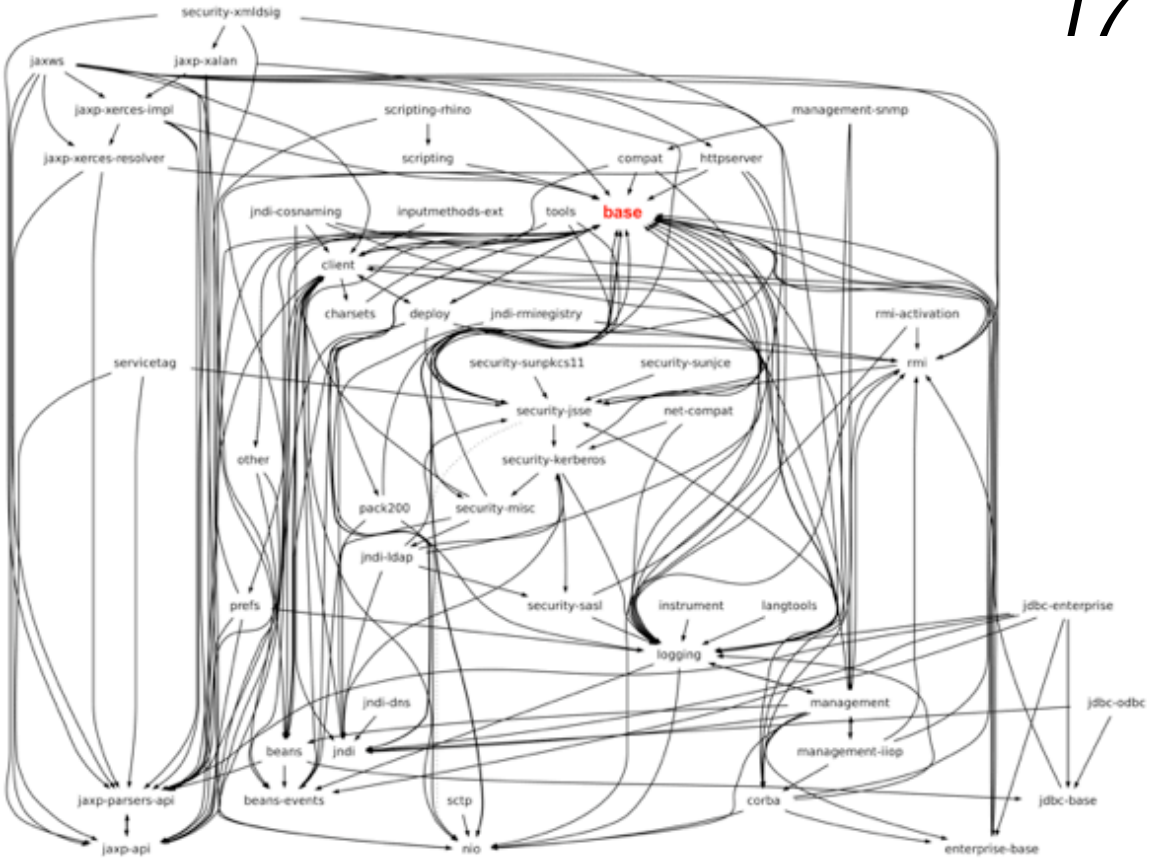


http://www.flickr.com/photos/lizadaly/2944362379

Modular platform: Before

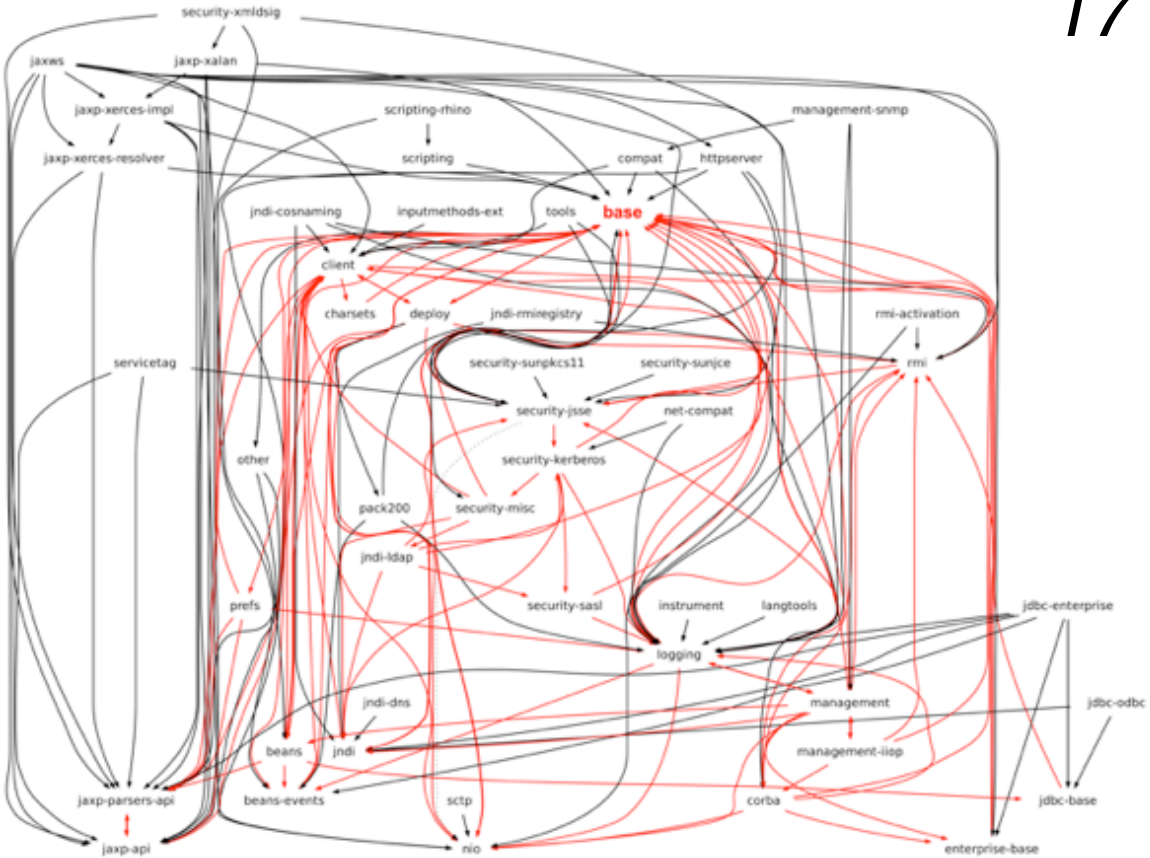
Modular platform: Before

50 nodes
171 edges



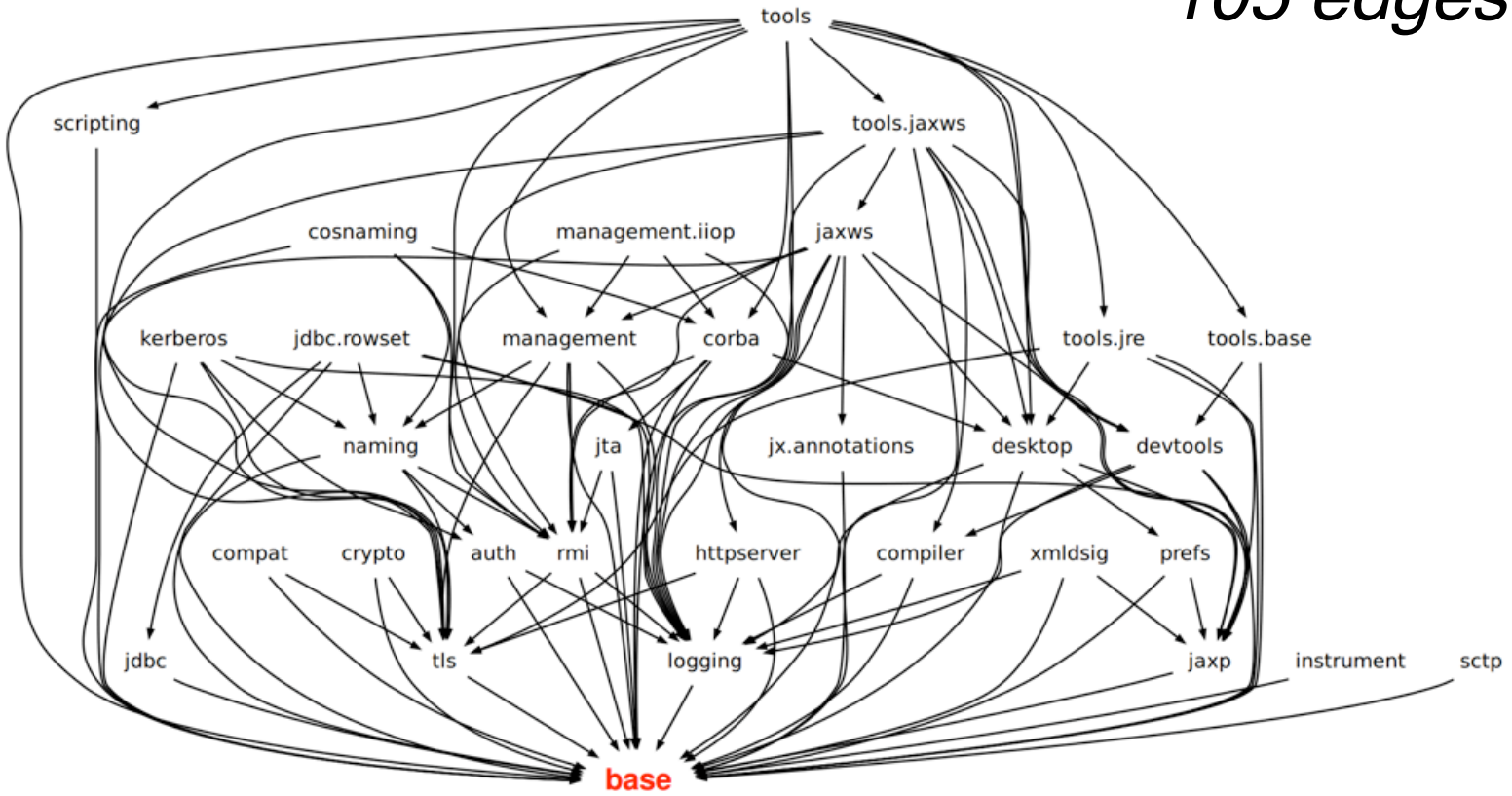
Modular platform: Before

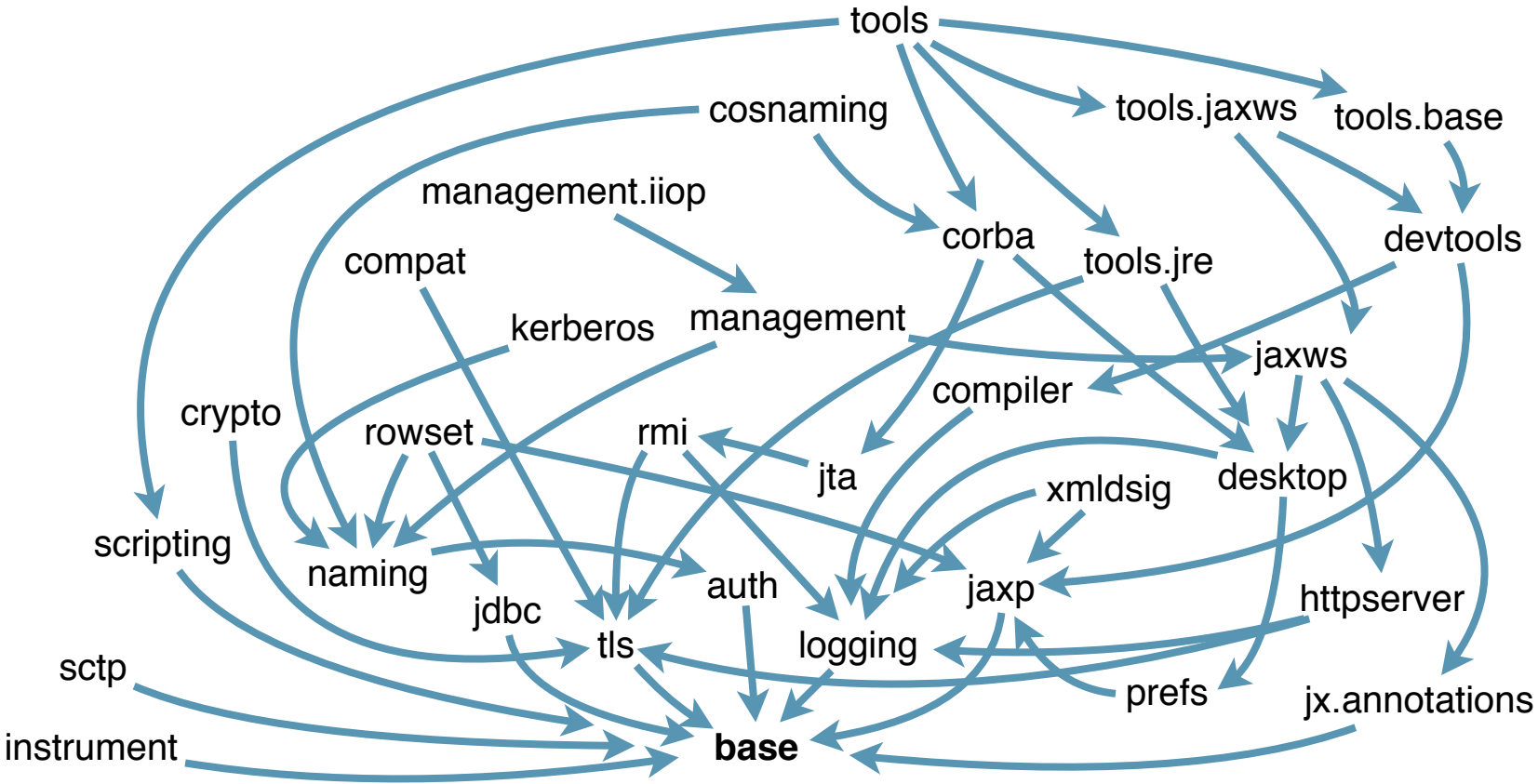
50 nodes
171 edges

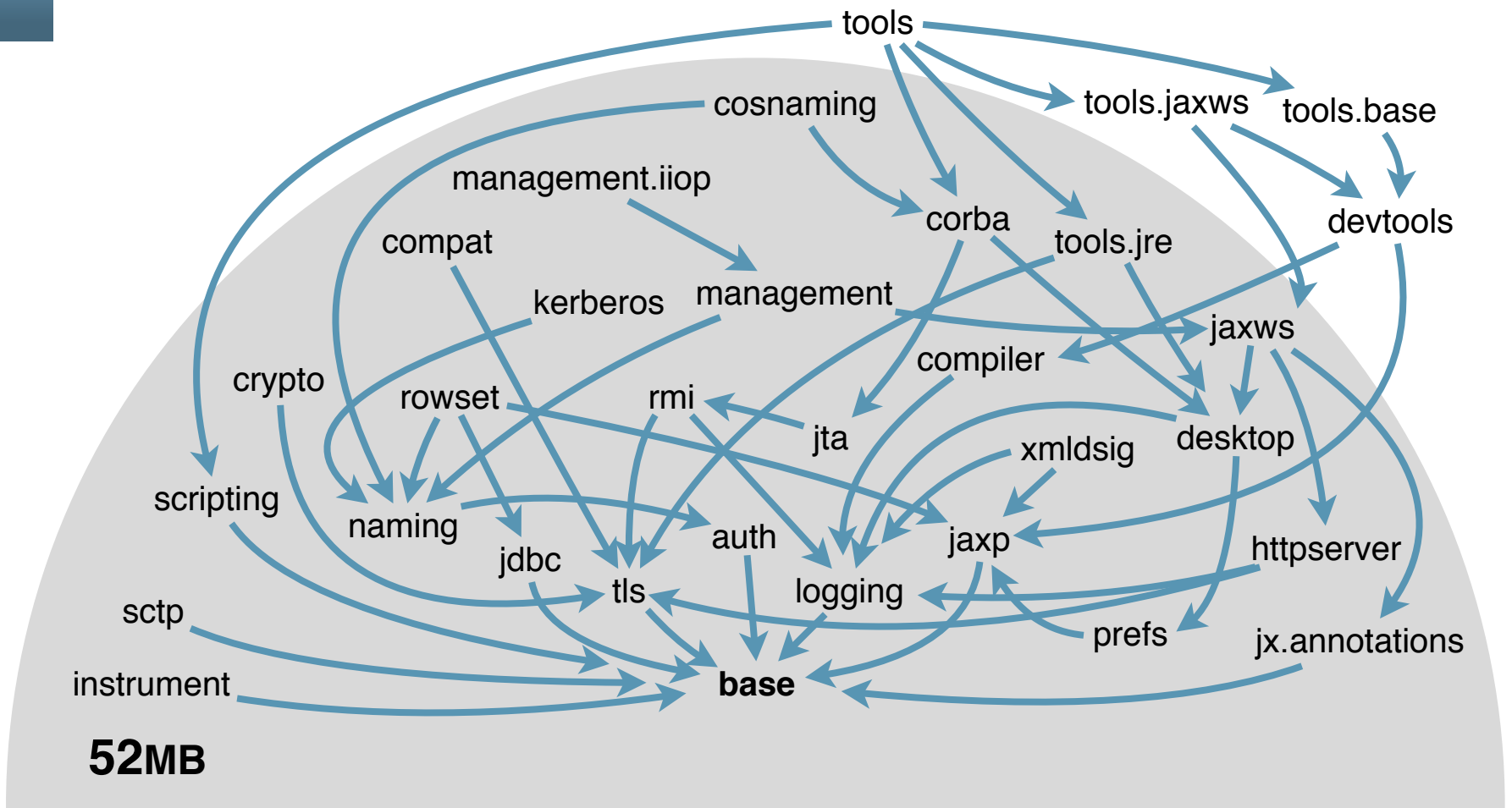


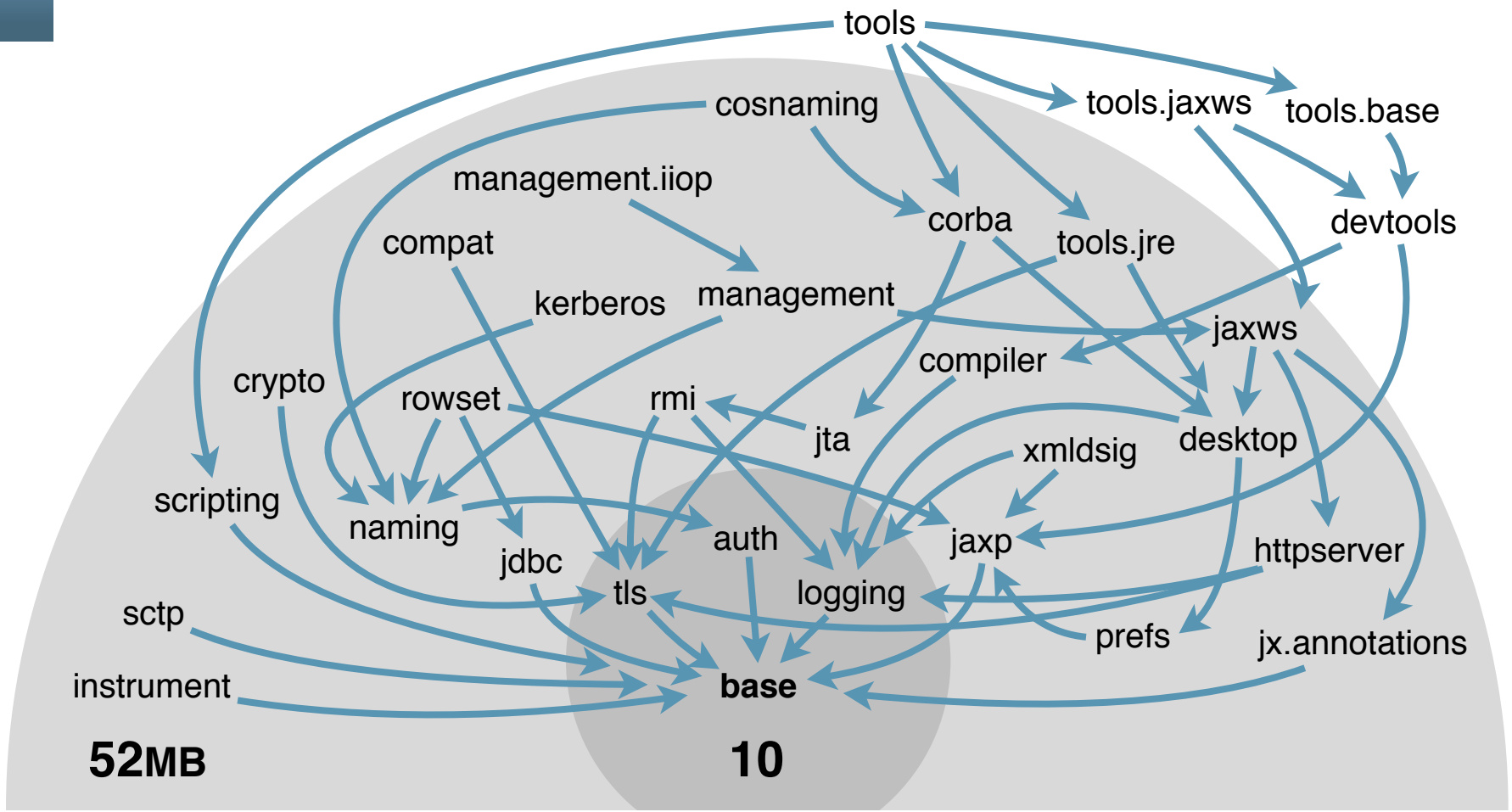
Modular platform: After

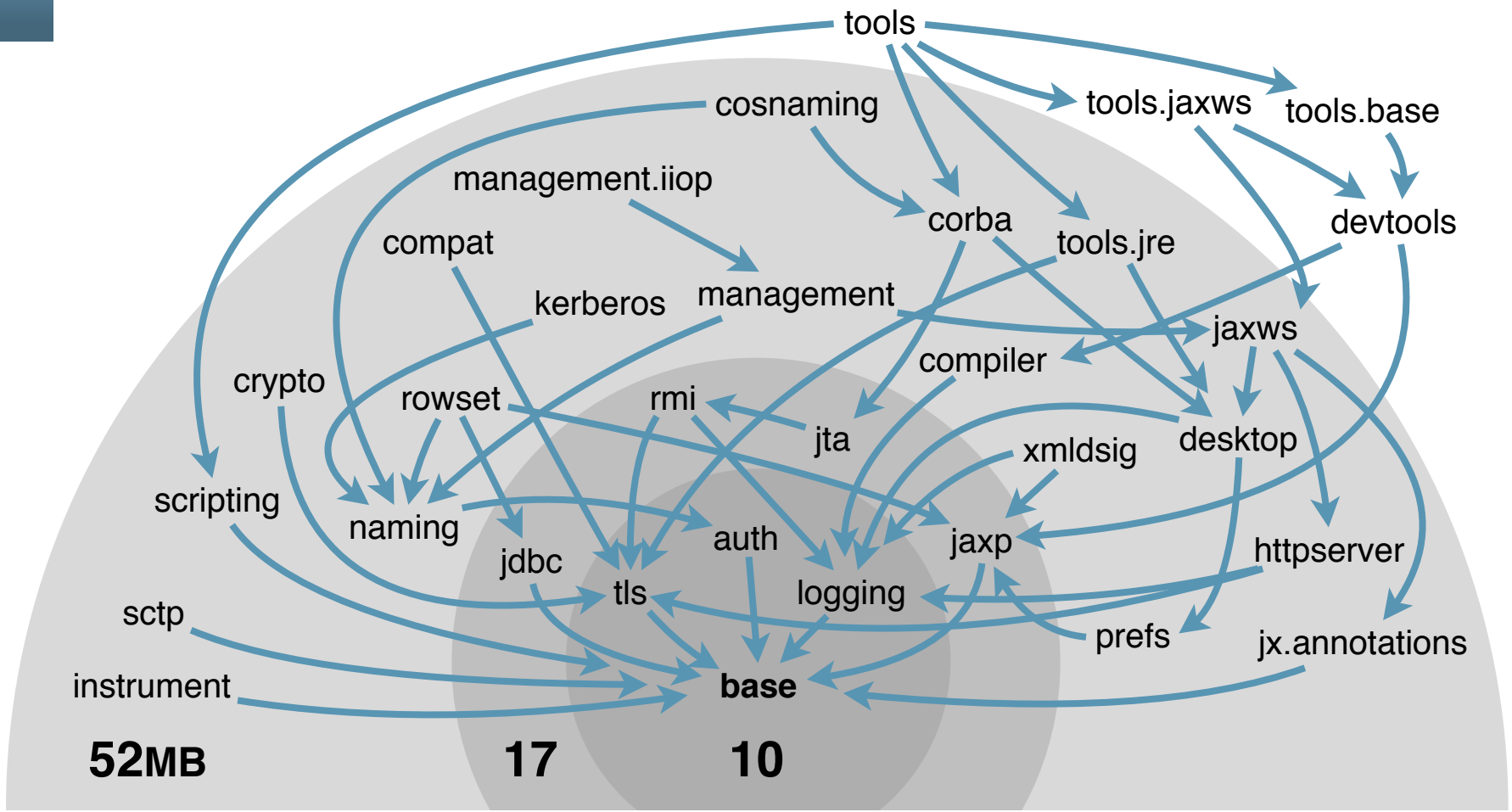
32 nodes
105 edges

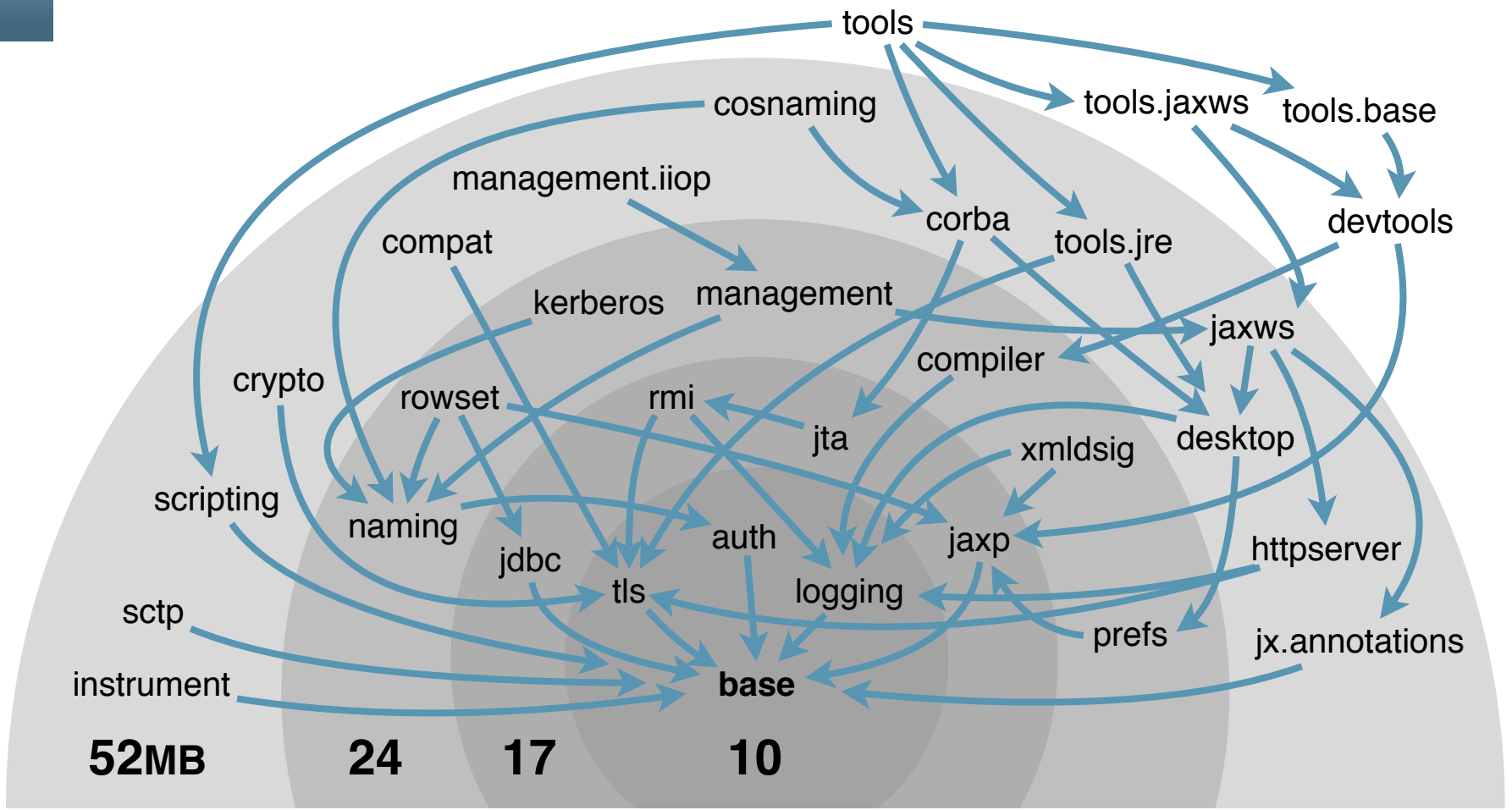


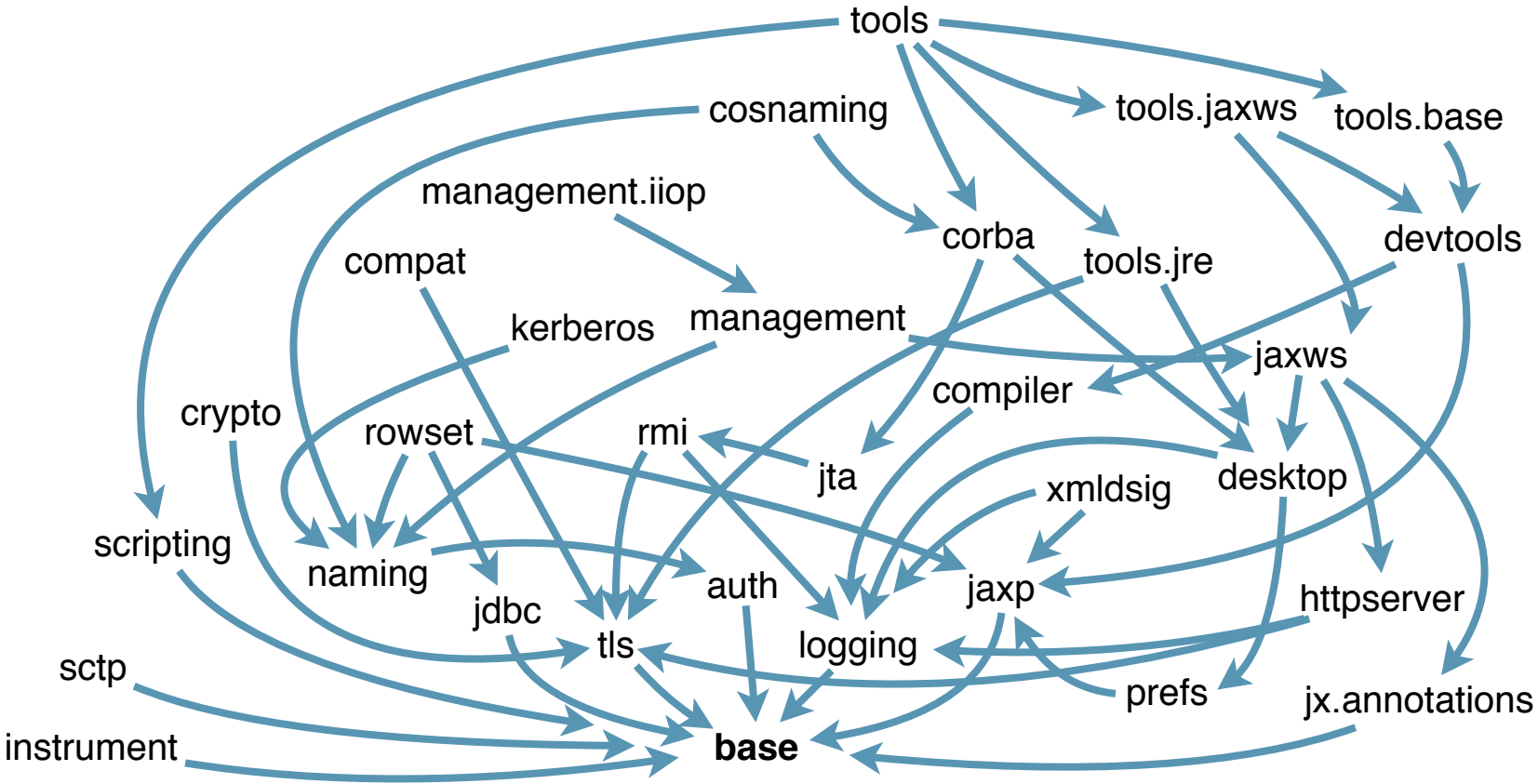

tools
cosnaming
tools.jaxws
tools.base
management.iiop
corba
devtools
compat
tools.jre
kerberos
management
jaxws
compiler
crypto
rowset
rmi
jta
desktop
xmldsig
scripting
naming
auth
jaxp
httpserver
jdbc
tls
logging
sctp
prefs
jx.annotations
instrument
base


tools
cosnaming
tools.jaxws
tools.base
management.iiop
corba
devtools
compat
tools.jre
kerberos
management
jaxws
compiler
crypto
rowset
rmi
jta
desktop
xmldsig
scripting
naming
auth
jaxp
httpserver
jdbc
tls
logging
sctp
prefs
jx.annotations
instrument
base
52MB


tools
cosnaming
tools.jaxws
tools.base
management.iiop
corba
devtools
compat
tools.jre
kerberos
management
jaxws
compiler
crypto
rowset
rmi
jta
desktop
xmldsig
scripting
naming
auth
jaxp
httpserver
jdbc
tls
logging
sctp
prefs
jx.annotations
instrument
base
52MB
10


tools
cosnaming
tools.jaxws
tools.base
management.iiop
corba
devtools
compat
tools.jre
kerberos
management
jaxws
compiler
crypto
rowset
rmi
jta
desktop
xmldsig
scripting
naming
auth
jaxp
httpserver
jdbc
tls
logging
sctp
prefs
jx.annotations
instrument
base
52MB
17
10


tools
cosnaming
tools.jaxws
tools.base
management.iiop
corba
devtools
compat
tools.jre
kerberos
management
jaxws
compiler
crypto
rowset
rmi
jta
desktop
xmldsig
scripting
naming
auth
jaxp
httpserver
jdbc
tls
logging
sctp
prefs
jx.annotations
instrument
base
52MB
24
17
10


tools
cosnaming
tools.jaxws
tools.base
management.iiop
corba
devtools
compat
tools.jre
kerberos
management
jaxws
compiler
crypto
rowset
rmi
jta
desktop
xmldsig
scripting
naming
auth
jaxp
httpserver
jdbc
tls
logging
sctp
prefs
jx.annotations
instrument
base

Compact Profiles

Compact Profiles

compact1

```
java.io
java.lang
java.math
java.net
java.nio
java.security
java.text
java.util
javax.crypto
javax.net
javax.security
```

10MB

Compact Profiles

compact1	compact2
java.io	java.rmi
java.lang	java.sql
java.math	javax.rmi
java.net	javax.sql
java.nio	javax.transaction
java.security	javax.xml
java.text	org.w3c.dom
java.util	org.xml.sax
javax.crypto	
javax.net	
javax.security	

10MB **17**

Compact Profiles

compact1	compact2	compact3
java.io	java.rmi	java.lang.instrument
java.lang	java.sql	java.lang.management
java.math	javax.rmi	java.util.prefs
java.net	javax.sql	javax.annotation.processing
java.nio	javax.transaction	javax.lang.model
java.security	javax.xml	javax.management
java.text	org.w3c.dom	javax.naming
java.util	org.xml.sax	javax.script
javax.crypto		javax.security.acl
javax.net		javax.security.auth.kerberos
javax.security		javax.security.sasl
		javax.sql.rowset
		javax.tools
		javax.xml.crypto
		org.ietf.jgss
10MB	**17**	**24**

Compact Profiles

compact1

```
java.io
java.lang
java.math
java.net
java.nio
java.security
java.text
java.util
javax.crypto
javax.net
javax.security
```

10MB

compact2

```
java.rmi
java.sql
javax.rmi
javax.sql
javax.transaction
javax.xml
org.w3c.dom
org.xml.sax
```

17

compact3

```
java.lang.instrument
java.lang.management
java.util.prefs
javax.annotation.processing
javax.lang.model
javax.management
javax.naming
javax.script
javax.security.acl
javax.security.auth.kerberos
javax.security.sasl
javax.sql.rowset
javax.tools
javax.xml.crypto
org.ietf.jgss
```

24

Full JRE

```
java.applet
java.awt
java.beans
javax.accessibility
javax.activation
javax.activity
javax.annotation
javax.imageio
javax.jws
javax.print
javax.rmi
javax.rmi.CORBA
javax.sound
javax.swing
javax.xml.bind
javax.xml.soap
javax.xml.ws
org.omg
```

52

NSA Suite B

Base64

HTTP Client

Bulk Data Operations

Unicode 6.2

Prepare for Modularization

Lambda (JSR 335)

Remove the Permanent Generation

Generalized Target-Type Inference

Date/Time API (JSR 310)

Improve Contended Locking

Java 8

Unicode CLDR

Parallel Array Sorting

Compact Profiles

DocTree API

Nashorn

Configurable Secure-Random Number Generation

TLS Server Name Indication

Type Annotations (JSR 308)

Lambda-Form Representation for Method Handles

Java 8

OpenJDK

Java Community Process

Java 8

ORACLE

OpenJDK

Java 8

openjdk.java.net/projects/jdk8
openjdk.java.net/projects/jdk8/spec
jdk8.java.net

JDK 8

The goal of this Project is to to produce an open-source reference implementation of the Java SE 8 Platform, to be defined by JSR 337 in the Java Community Process.

Content

JDK 8 is the second part of Plan B. The single driving feature of the release is Project Lambda. (Project Jigsaw was proposed for this release but later dropped). Additional features proposed via the JEP Process will be included, but they must fit into the overall schedule required for Lambda. Detailed information on the features funded and targeted to the release, so far, can be found on the features page.

Schedule

The overall development schedule is divided into a sequence of milestone cycles, with each feature targeted to a specific milestone.

2012/04/26	**M1**	
2012/06/14	**M2**	
2012/08/02	**M3**	
2012/09/13	**M4**	
2012/11/29	**M5**	
2013/01/31	**M6**	Feature Complete
2013/02/21	**M7**	Developer Preview
2013/07/05	**M8**	Final Release Candidate
2013/09/09	**GA**	General Availability

Further information on milestone content and the final phases of the release can be found on the milestones page. The builds and integrations page contains week-by-week scheduling information.

Early-access binaries

EA binaries built from the JDK 8 code base are available today from Oracle.

Java SE 8 Platform Umbrella JSR (337)

This is the primary web page for JSR 337, the Platform Umbrella JSR for Java SE 8.

The original JSR submission may be found on the official JCP page.

Expert Group

- Kevin Bourrillion (Google)
- Andrew Haley (Red Hat)
- Steve Poole (IBM)
- Mark Reinhold (Oracle)

Schedule

2012/7	Expert Group formation
2013/2	Early Draft Review
2013/5	Public Review
2013/6	Proposed Final Draft
2013/8	Final Release

Mailing lists

There are three mailing lists:

- java-se-8-spec-experts is the Expert Group (EG) list. Only EG members may subscribe and post to this list, but the archives are public.
- java-se-8-spec-observers is for those who wish to monitor and, perhaps, discuss the EG's progress. Messages sent to the primary EG list are automatically forwarded to this list. Anyone may subscribe to this list, and any subscriber may post. EG members are under no obligation to follow the traffic on this list.
- java-se-8-spec-comments is for sending comments, suggestions, and other feedback directly to the EG. Only EG members may subscribe to this list, but anyone may post, and the archives are public. The EG will read all messages

Java 9 … and beyond!

Unified Type System

OpenJFX

Self Tuning JVM

Java 9 … and beyond!

Jigsaw

Ease of use

Reification

Unified Type System

Project Sumatra – Java for GPUs

Improved Native Integration

OpenJFX

Self Tuning JVM

Resource Management

Java 9 … and beyond!

Generic Lang Interoperability

Data Structure Optimizations

Jigsaw

Penrose

Ease of use

More and More Ports

Optimizations

Reification

Multi-Tenancy

JDK Enhancement Proposals (JEPs)

openjdk.java.net/jeps

JDK Enhancement Proposals (JEPs)

openjdk.java.net/jeps

OpenJDK

OpenJDK FAQ
Installing
Contributing
Sponsoring
Developers' Guide
Mailing lists
Bylaws
Census
Legal

JEP Process

search

Source code
Mercurial
(6, 7, 7u, 8)
Bundles (6, 7, 7u2)

Groups
(overview)
2D Graphics
AWT
Build
Compiler
Conformance
Core Libraries
Governing Board
HotSpot

JEP 0: JEP Index

Author	Mark Reinhold
Organization	Oracle
Created	2011/8/23
Updated	2013/2/1
Type	Informational
State	Active

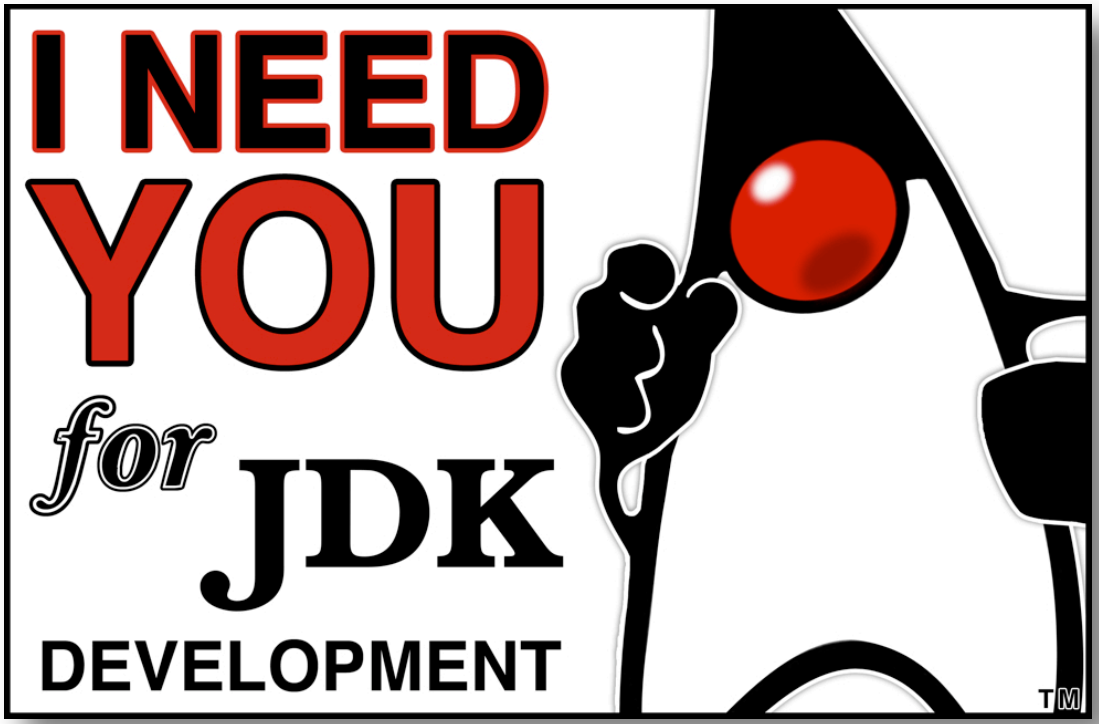
This JEP is the index of all JDK Enhancement Proposals, known as JEPs.
See JEP 1 for an overview of the JEP Process.

P	Act		1	JDK Enhancement-Proposal & Roadmap Process
P	Act		2	JEP Template
F	Fun	8 core/lang	101	Generalized Target-Type Inference
F	Can	core/libs	102	Process API Updates
F	Fun	8 core/libs	103	Parallel Array Sorting
F	Fun	8 core/lang	104	Annotations on Java Types
F	Fun	8 core/lang	105	DocTree API
F	Fun	8 core/lang	106	Add Javadoc to javax.tools
F	Fun	8 core/libs	107	Bulk Data Operations for Collections
F	Can	core/libs	108	Collections Enhancements from Third-Party Libraries
F	Fun	8 core/libs	109	Enhance Core Libraries with Lambda

JDK Enhancement Proposals (JEPs)

138 Autoconf-Based Build System
139 Enhance javac to Improve Build Speed
140 Limited doPrivileged
141 Increase the Client VM's Default Heap Size
142 Reduce Cache Contention on Specified Fields
143 Improve Contended Locking
144 Reduce GC Latency for Large Heaps
145 Cache Compiled Code
146 Improve Fatal Error Logs
147 Reduce Class Metadata Footprint
148 Small VM
149 Reduce Core-Library Memory Usage
150 Date & Time API
~~151 Compress Time Zone Data~~
152 Crypto Operations with Network HSMs
153 Launch JavaFX Applications
~~154 Remove Serialization~~
155 Concurrency Updates
156 G1 GC: Reduce need for full GCs
157 G1 GC: NUMA-Aware Allocation
158 Unified JVM Logging
159 Enhanced Class Redefinition
160 Lambda-Form Representation for Method Handles
161 Compact Profiles
162 Prepare for Modularization
163 Enable NUMA Mode by Default When Appropriate
164 Leverage CPU Instructions for AES Cryptography
165 Compiler Control
166 Overhaul JKS-JCEKS-PKCS12 Keystores
167 Event-Based JVM Tracing
168 Network Discovery of Manageable Java Processes
169 Value Objects
170 JDBC 4.2
171 Fence Intrinsics
172 DocLint
173 Retire Some Rarely-Used GC Combinations
174 Nashorn JavaScript Engine
175 Integrate PowerPC/AIX Port into JDK 8

The preceding material is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.



openjdk.java.net/projects/jdk8

jdk8.java.net

Java 8, Java 9, and Beyond!

Mark Reinhold (@mreinhold)

Chief Architect, Java Platform Group, Oracle

Jfokus 2013