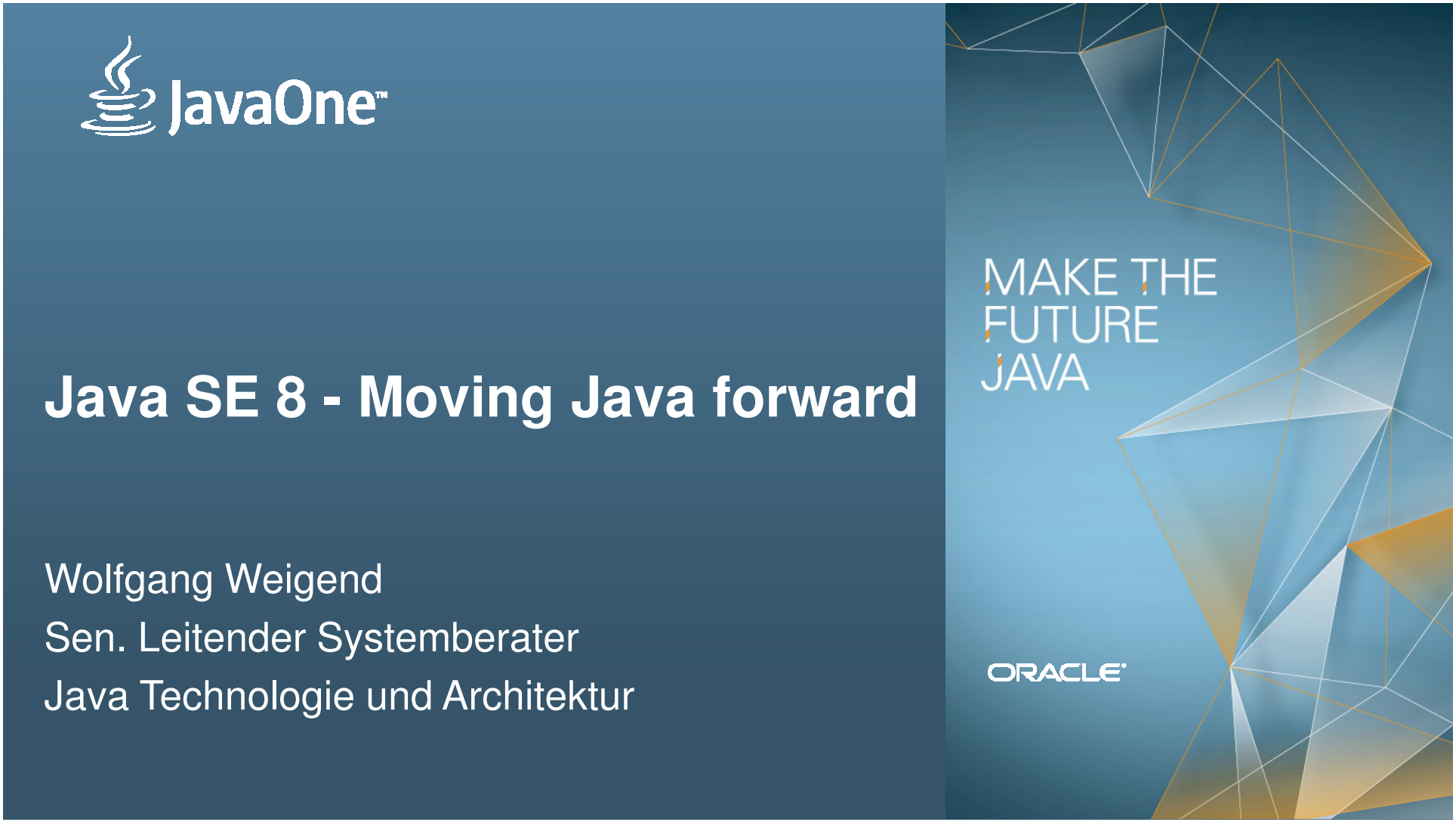
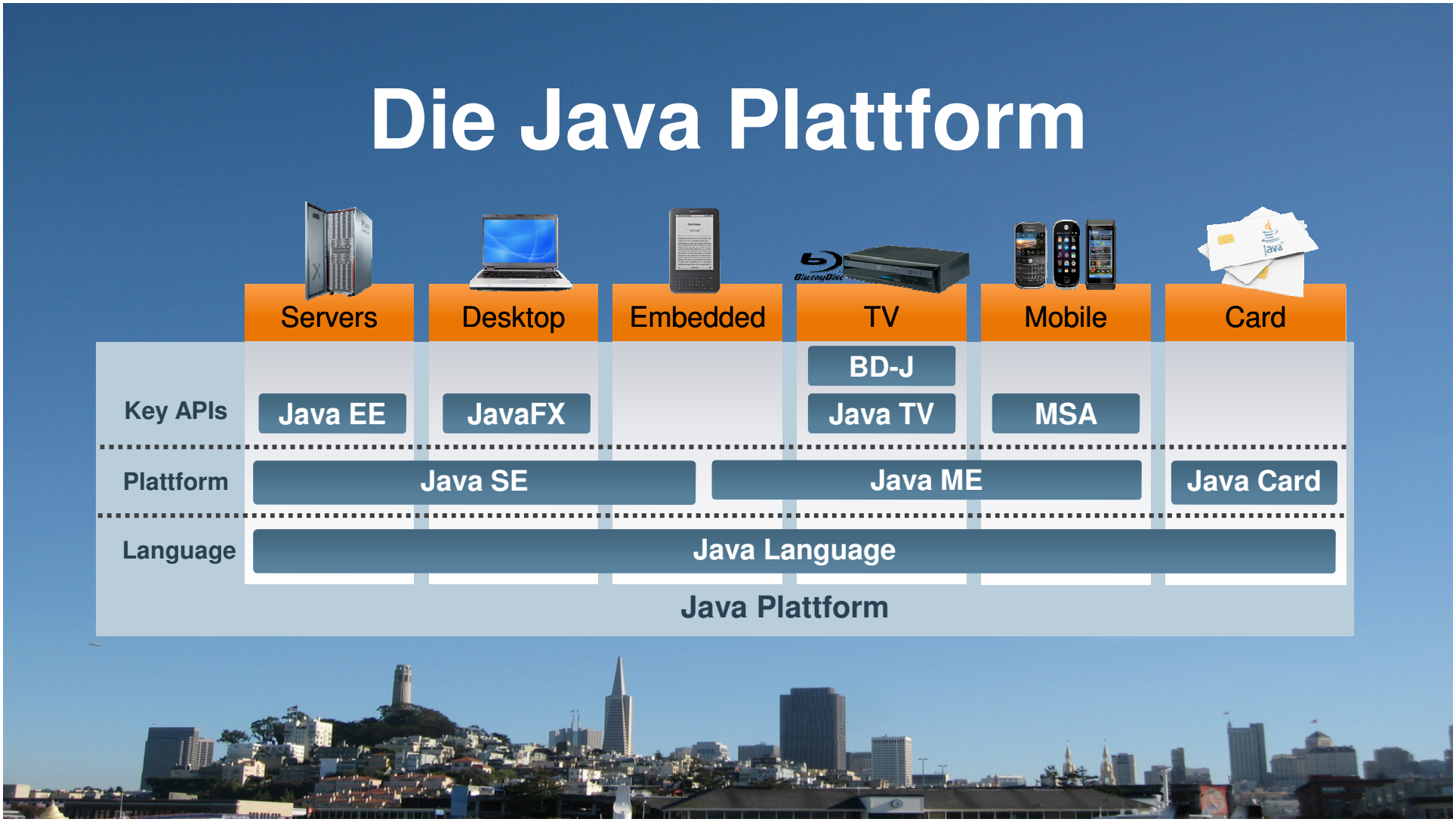


Java SE 8 - Moving Java forward

Wolfgang Weigend
Sen. Leitender Systemberater
Java Technologie und Architektur

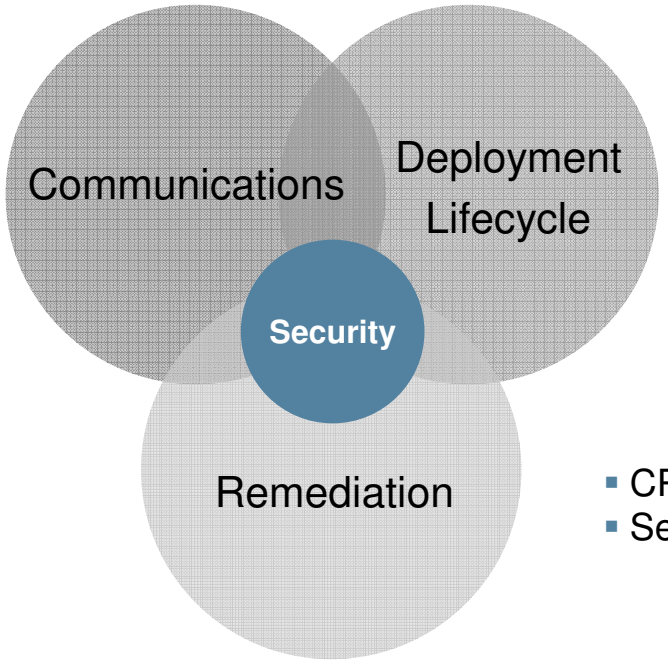
The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development,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

Die Java Plattform



Larger Security Policy Areas

- SA / CPU RSS Feeds
- Security Blog
- eBlasts
- Java.com Security



- Architecture Review
- Peer Review
- Security Testing
- Post Mortems

- CPU
- Security Alerts

 |

Java™ ORACLE®

Agenda

- Java SE 8 (JSR 337) und Component JSR's
- JDK Enhancement Proposals (JEP's)
- Sprachänderungen
- Core Bibliotheken
- Internationalisierung
- Security
- Java Plattform
- Virtual Machine
- Zusammenfassung

JDK Enhancement Proposal's (JEP's)

- Regularly updated list of proposals
  - Serve as the long-term roadmap for JDK release projects
  - Roadmap extends for at least three years
- Uniform format and a central archive for enhancement proposals
  - Interested parties can find, read, comment, and contribute
- Process is open to every OpenJDK Committer
- Enhancement is a non-trivial change to the JDK code base
  - Two or more weeks of engineering effort
  - significant change to JDK or development processes and infrastructure
  - High demand from developers or customers

Sprachänderungen

Lambda Ausdrücke JSR-335

- **Functional Interfaces: “An interface with one method”**
- **Gehört zum Sprachumfang von Java SE 8**
  - Final Release Specification

    file:///C:/Java/jsr335-final/index.html
  - Prototype Binaries verfügbar: **http://jdk8.java.net/lambda/**
- **Lambda Expressions (closures)**

```
/* (int x, int y) {return x+y; } */
```

- **Parameter Liste → -> Operator → Expression od. Statements**

```
(String x) -> {return !x.isEmpty();}
```

- **Was hergeleitet werden kann, kann auch weggelassen werden**

```
x -> !x.isEmpty()
```


Extension Methods

Bringt Mehrfachvererbung und Funktionalität für Java

- Provide a mechanism to add new methods to existing interfaces
  - Without breaking backwards compatibility
  - Gives Java multiple inheritance of behaviour, as well as types
    - but not state!

```
public interface Set<T> extends Collection<T> {
   public int size();

   ...   // The rest of the existing Set methods

   public T reduce(Reducer<T> r)
     default Collections.<T>setReducer;
}
```

Annotations on Java Types

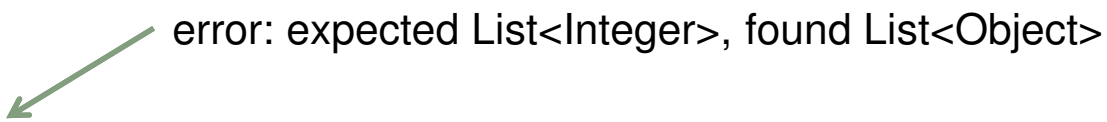
- Annotations can currently only be used on type declarations
  - Classes, methods, variable definitions
- Extension for places where types are used
  - e.g. parameters
- Permits error detection by pluggable type checkers
  - e.g. null pointer errors, race conditions, etc

```
public void process(@notnull List data) {…}
```


Allgemeine Target-Type-Herleitung

Bessere Nutzung von Generics

```
class List<E> {
   static <Z> List<Z> nil() { ... };
   static <Z> List<Z> cons(Z head, List<Z> tail) {
... };
   E head() { ... }
}
List<String> ls = List.nil();  // Inferred correctly
```




```
List.cons(42, List.nil());
```


Java™ ORACLE®

Zugriff auf Parameternamen während der Laufzeit

- Mechanism to retrieve parameter names of methods and constructors
  - At runtime via core reflection
- Improved code readability
  - Eliminate redundant annotations
- Improve IDE capabilities
  - Auto-generate template code

Kleinere Anpassungen

- Repeating annotations
  - Multiple annotations with the same type applied to a single program element
- No more `apt` tool and associated API
  - Complete the transition to the JSR 269 implementation
- DocTree API
  - Provide access to the syntactic elements of a javadoc comment
- DocLint tool
  - Use DocTree API to identify basic errors in javadoc comments
- Javadoc support in `javax.tools`
  - Invoke javadoc tools from API as well as command line/exec

Core Bibliotheken

Erweiterung der Core Libraries mit Lambdas

- Keine einfache Aufgabe!
  - Java SE 7 has 4024 standard classes
- Modernize general library API's
- Improve performance
  - Gains from use of invokedynamic to implement Lambdas
- Demonstrate best practices for extension methods

Concurrency Updates

- Scalable update variables
  - `DoubleAccumulator`, `DoubleAdder`, etc
  - Multiple variables avoid update contention
  - Good for frequent updates, infrequent reads
- `ConcurrentHashMap` updates
  - Improved scanning support, key computation
- `ForkJoinPool` improvements
  - Completion based design for IO bound applications

Bulk Data Operations für Collections

Filter, Map, Reduce for Java

- Add functionality to the Java Collections Framework for bulk operations upon data
- This is commonly referenced as “filter/map/reduce for Java”
- The bulk data operations include both serial (on the calling thread) and parallel (using many threads) versions of the operations
  - Serial and parallel implementations
- Operations upon data are generally expressed as lambda functions
- Parallel implementation builds on Fork-Join framework

Java™ ORACLE®

Parallel Array Sorting

- Additional utility methods in **`java.util.Arrays`**
  - **`parallelSort`** (multiple signatures for different primitives)
- Anticipated minimum improvement of 30% over sequential sort
  - For dual core system with appropriate sized data set
- Built on top of the fork-join framework
  - Uses Doug Lea's **`ParallelArray`** implementation
  - Requires working space the same size as the array being sorted

Lambda Ausdrücke – Parallelisiert

State of the Lambda Libraries Edition

```
List<Student> students = new ArrayList<>(...);
...
double highestScore =
  students.parallel()
          .filter(s -> s.getGradYear() == 2011)
          .map(s -> s.getScore())
          .reduce(0.0, Integer::max);
```

- More readable
- Better abstraction
- No reliance on mutable state
- Runs in parallel
- Works on any data structure that knows how to subdivide itself

- **Concurrent Bulk Data Operations in Java collections API's (JEP 107)**
  - filter/map/reduce

Date and Time API – JSR 310

- JEP 150 by Stephen Colebourne
- A new date, time, and calendar API for the Java SE platform
- Supports standard time concepts
  - Partial, duration, period, intervals
  - date, time, instant, and time-zone
- Provides a limited set of calendar systems and be extensible to others
- Uses relevant standards, including ISO-8601, CLDR, and BCP47
- Based on an explicit time-scale with a connection to UTC

Date and Time API – JSR 310

Package	Description
java.time	The main API for dates, times, instants, and durations
java.time.chrono	Generic API for calendar systems other than the default ISO
java.time.format	Provides classes to print and parse dates and times
java.time.temporal	Access to date and time using fields and units, and date time adjusters
java.time.zone	Support for time-zones and their rules

JDBC 4.2

Minor enhancements for usability and portability

- Generic setter/update methods
  - `ResultSet`, `PreparedStatement`, and `CallableStatement`
  - Support new data types such as those being defined in JSR 310
- REF_CURSOR support for `CallableStatement`
- `DatabaseMetaData.getIndexInfo` extended
  - new columns for CARDINALITY and PAGES which return a long value
- New `DatabaseMetaData` method
  - `getMaxLogicalLobSize`
  - Return the logical maximum size for a LOB

Base64 Encoding und Decoding

- Currently developers are forced to use non-public APIs
  - `sun.misc.BASE64Encoder`
  - `sun.misc.BASE64Decoder`
- Java SE 8 now has a standard way
  - `java.util.Base64.Encoder`
  - `java.util.Base64.Decoder`
  - `encode`, `encodeToString`, `decode`, `wrap` methods

Statically-Linked JNI Libraries (JEP 178)

- Enhance the JNI specification to support statically-linked native libraries
  - Modify the Java SE specification, and the JDK, to enable developers to package a Java runtime, native application code, and Java application code together into a single binary executable that does not require the use of shared native libraries
  - Require no changes to existing Java code in order to use a static native library as opposed to a dynamic native library. A method invocation of the form System.loadLibrary("foo"), in particular, should be able to load the "foo" library regardless of whether that library is provided in static or dynamic form
  - Allow a Java application to use a combination of static and dynamic native libraries, although static libraries must be in memory prior to any attempt to use them
  - Allow JVMTI Java Agents to be optionally statically linked with Java runtimes

Kleinere Anpassungen

- `javax.lang.model` implementation backed by core reflection
  - Uniform annotation API to view compile-time and runtime reflective information
- Charset implementation improvements
  - Reduced size of charsets, improved performance of encoding/decoding
- Reduced core-library memory usage
  - Reduced object size, disable reflection compiler, internal table sizes, etc

Internationalisierung

Locale Data Packing

- Tool to generate locale data files
  - convert from LDML (Locale Data Markup Language)
- Unicode Common Locale Data Repository (CLDR) support
- Locale elements supported from underlying platform

BCP 47 Locale Mapping (API Def.)

- Language tags to indicate the language used for an information object
  - RFC-5646 (Language range)
  - RFC-5456 (Language priority, preference)
- Language range `Collection<String>`
- Language priority `List <String>`
- Three operations added to `Locale` class
  - `filterBasic`
  - `filterExtended`
  - `lookup`

Unicode 6.2

- Java SE 7 support Unicode 6.0
- Changes in Unicode 6.1 (February, 2013)
  - Add 11 new blocks to `java.lang.Character.UnicodeBlock`
  - Add 7 new scripts to `java.lang.Character.UnicodeScript`
  - Support over 700 new characters in `java.lang.Character`, `String`, and other classes
- Changes in Unicode 6.2 (September, 2013)
  - Support a new Turkish currency sign (U+20BA)


Security

Konfigurierbarer Secure Random Number Generator

- Better implementation of **`SecureRandom`**
- Currently applications can hang on Linux
  - JVM uses **`/dev/random`**
  - This will block if the system entropy pool is not large enough
- Still a work in progress

Enhanced Certificate Revocation-Checking API

- Current `java.security.cert` API is all-or-nothing
  - Failure to contact server is a fatal error
- New classes
  - `RevocationChecker`
  - `RevocationParameters`

Kleinere Anpassungen

- Limited `doPrivilege`
  - Execute Lambda expression with privileges enabled
- NSA Suite B cryptographic algorithms
  - Conform to standards to meet U.S. government, banking requirements
- AEAD CipherSuite support
  - Conform to standards to meet U.S. government, banking requirements
- SHA-224 message digests
  - Required due to known flaw in SHA-1
- Leverage CPU instructions for AES cryptography
  - Improve encryption/decryption performance

Kleinere Änderungen

- Microsoft Services For UNIX (MS-SFU) Kerberos 5 extensions
  - Enhanced Microsoft interoperability
- TLS Server Name Indication (SNI) extension
  - More flexible secure virtual hosting, virtual-machine infrastructure
- PKCS#11 crypto provider for 64-bit Windows
  - Allow use of widely available native libraries
- Stronger algorithms for password-based encryption
  - Researchers and hackers move on
- Overhaul JKS-JCEKS-PKCS12 keystores
  - Simplify interacting with Java SE keystores for cryptographic applications


Java Plattform

Launch JavaFX Applications

- Support the direct launching of JavaFX applications
- Enhancement to the java command line launcher

JavaFX via Open Source auf dem Weg ins JDK 8

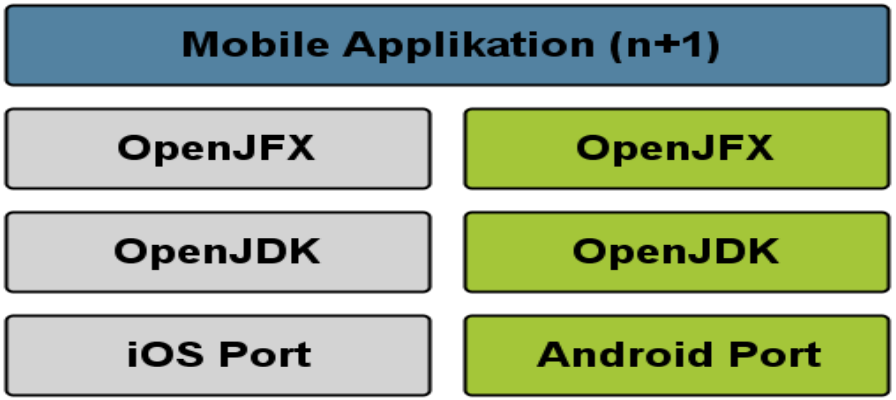
Open Source	Übergang	Standardisierung
☑ OpenJFX Project under OpenJDK	☑ Common license with Java SE (in place)	☑ Oracle committed to JavaFX standardization
☑ First phase to focus on UI Controls	☑ JavaFX to be included in Java SE by JDK 8	☐ JSR to be submitted through JCP
	☑ JavaFX for Java SE Embedded (ARM)	

Java™ ORACLE®

JavaFX goes Open Source

iOS- und Android-Implementierungen

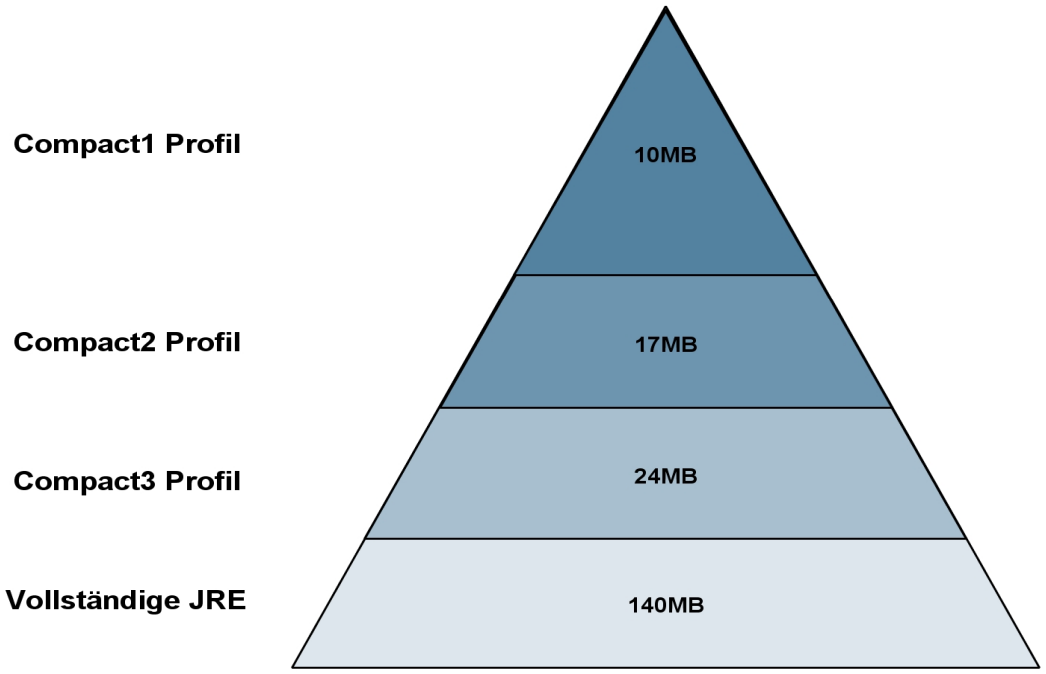
- iOS- und Android-Implementierungen werden Open Source
- Lizensierung mit eigenem Applikations-Co-Bundle



 |

JdVd ORACLE

Compact-Profile für festgelegten Speicherbedarf



Compact-Profile mit Packages

Compact1 Profil	Compact2 Profil	Compact3 Profil	Vollständige JRE
java.lang	java.sql	java.lang.management	java.applet
java.io	jvax.sql	javax.management	java.awt
java.nio	javax.xml	javax.naming	java.beans
java.text	org.w3c.dom	java.sql.rowset	javax.activity
java.math	org.xml.sax	javax.security.auth.kerberos	javax.rmi
java.net	java.rmi	org.ietf.jgss	javax.rmi.CORBA
javax.net	javax.rmi	javax.script	org.omg
java.util	javax.transaction	javax.xml.crypto	javax.accessibility
java.util.logging		java.util.prefs	javax.imagio
java.security		javax.security.sasl	javax.print
javax.crypto		javax.security.acl	javax.sound
javax.security		javax.lang.instrument	javax.swing
		javax.annotation.processing	javax.activation
		javax.lang.model	javax.jws
		javax.lang.model.element	javax.xml.bind
		javax.lang.model.type	javax.xml.soap
		javax.lang.model.util	javax.xml.ws
		javax.tools	javax.annotation

 |

Vorbereitung zur Modularisierung
Getting ready for Jigsaw

- Fix some assumptions about classloaders
- Use `ServiceLoader` rather than proprietary SPI code
- Tool to analyse application code dependencies
- Deprecate APIs that will impede modularization
  - e.g. `java.util.logging.LogManager.addPropertyChangeListener`
- Review and possibly change `$JAVA_HOME` normative references
  - Relative v. absolute pathnames

Java™ ORACLE®

Abgespeckte Implementierungen

- Applications that ship bundled with a JRE don't need to include all the class libraries
- This does not break 'Write once, run anywhere'
- Only applicable for bundled JRE
  - JRE cannot be used by other applications


Virtual Machine

Lambda-Form Representation für MethodHandles

Assembly language code re-written in Java

- Improve performance, quality, and portability of method handles and invokedynamic
- Reduce the amount of assembly code in the JVM
- Reduce native calls during method handle processing
- Better reference implementation of JSR 292 (invokedynamic)

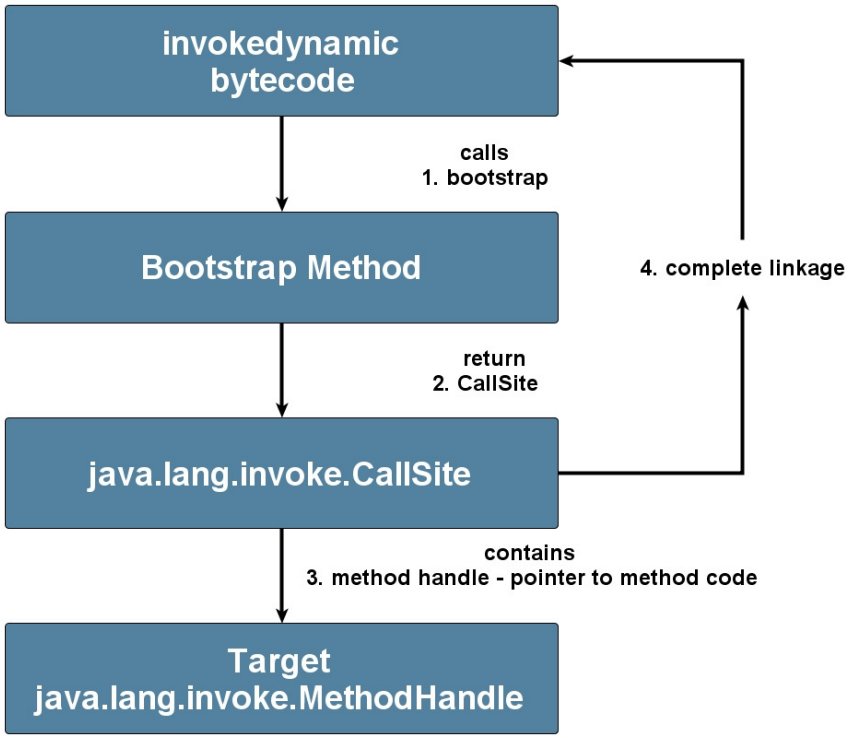
Project Nashorn

- Lightweight, high-performance JavaScript engine
  - Integrated into JRE
- ECMAScript 5.1 compliant JavaScript that runs on the JVM
  - Accessible via javax.script API or jjs command line
- Replacement for Rhino
- Exploits JSR-292 via Dynalink
- Fully part of the OpenJDK
- Available on Java SE and Java ME
- Nashorn - Der Weg zur polyglotten VM

 |

Project Nashorn

Invokedynamic Bytecode und „MethodHandle“



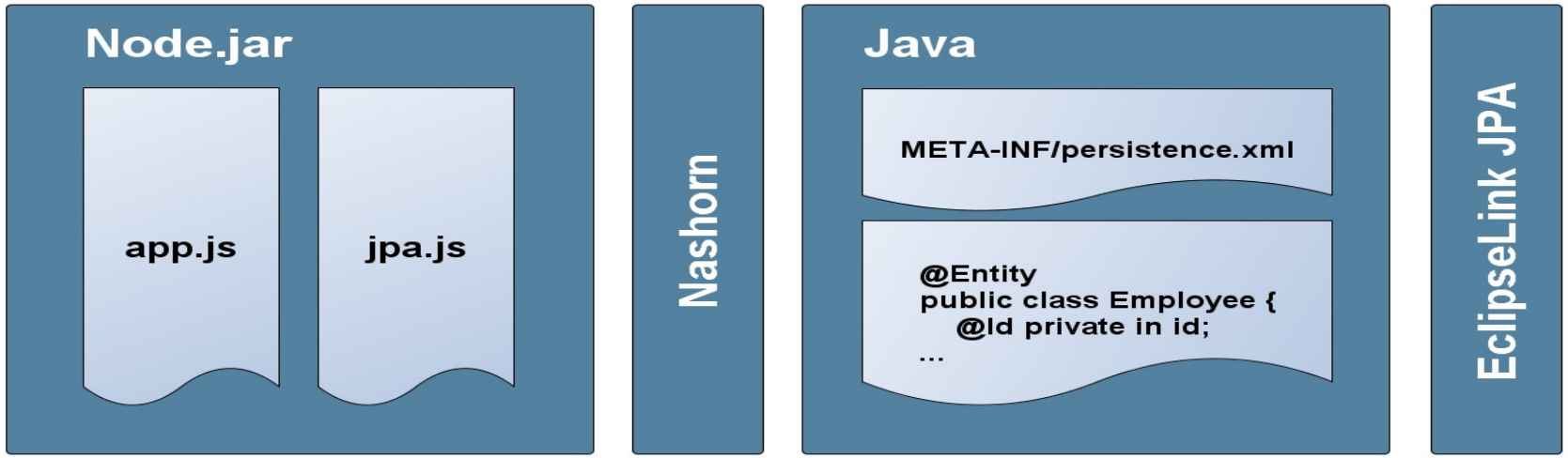
Java™ ORACLE®

Nashorn Debugging - Experiments

- Debugging options
  - IDE integration of Nashorn
  - Common API for non-Java languages
  - Chrome/Safari wire protocols
  - Askari
    - Askari is a sample JS debugger demonstrated at JavaOne SF 2013
    - https://github.com/wickund/nashornexamples

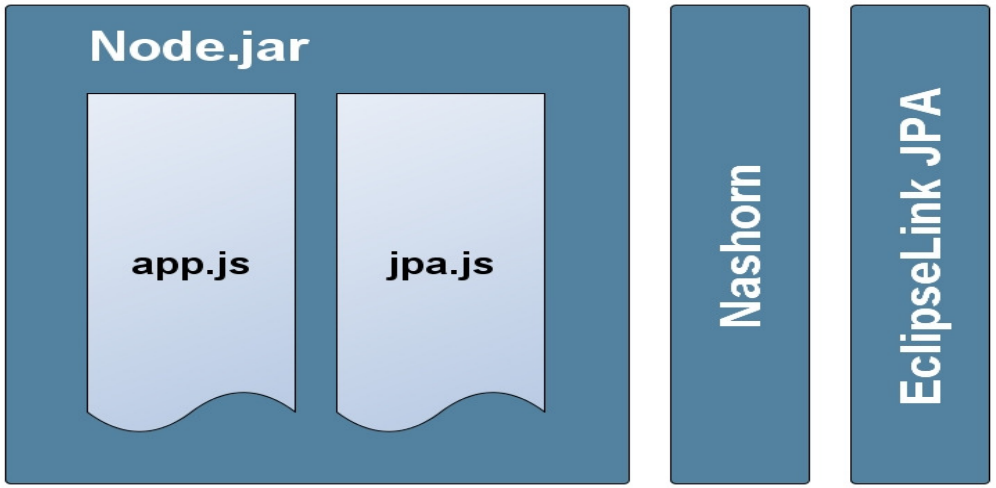
Nashorn - Einsatzvarianten

- Verwendung von Node.jar und Java
  – über einen JPA-Wrapper durch XML und annotierte Java Klassen, erfolgt der Java EE-Zugriff von Server-seitigem JavaScript



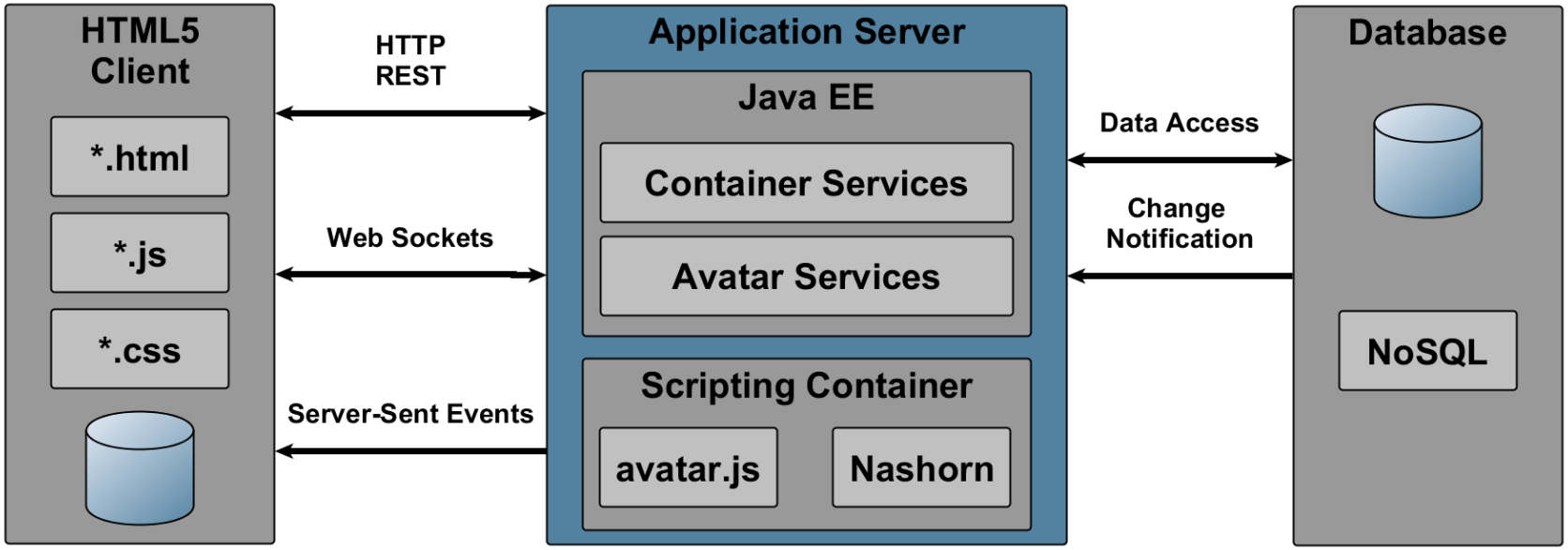
Nashorn - Einsatzvarianten

- JavaScript-Datenbank-JPA
  - benutzt JavaScript JPA über ein Datenbankschema
  - Java SE JavaScript Persistenz verwendet JPA, mit den Mappings vom Datenbankschema



Java™ ORACLE®

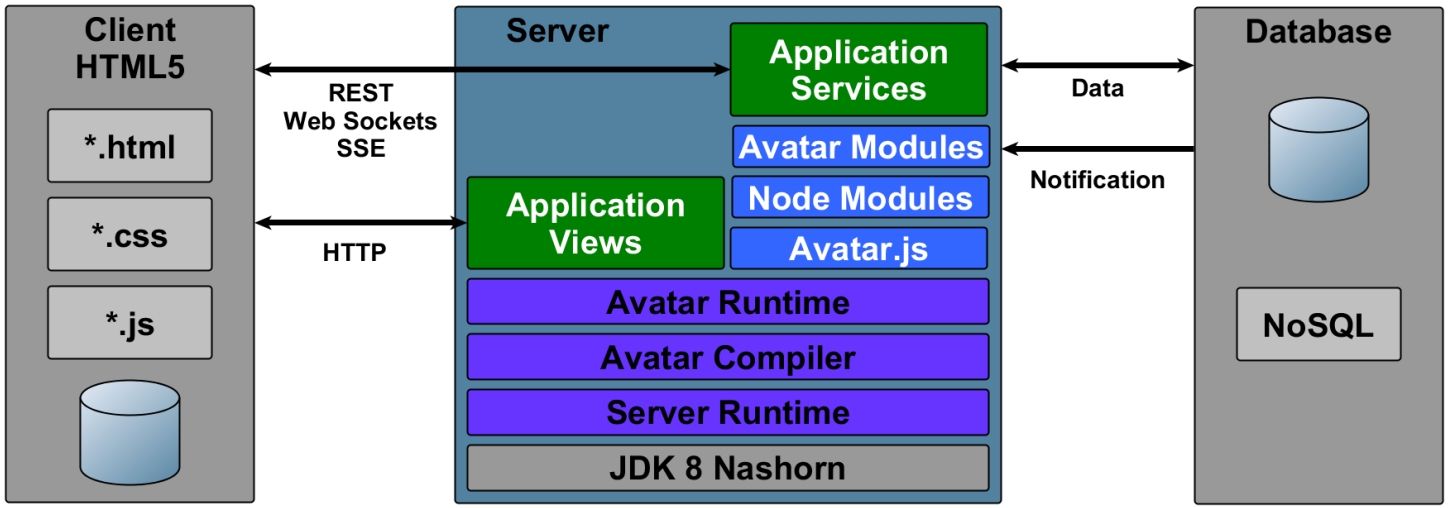
Project Avatar (1)



 |

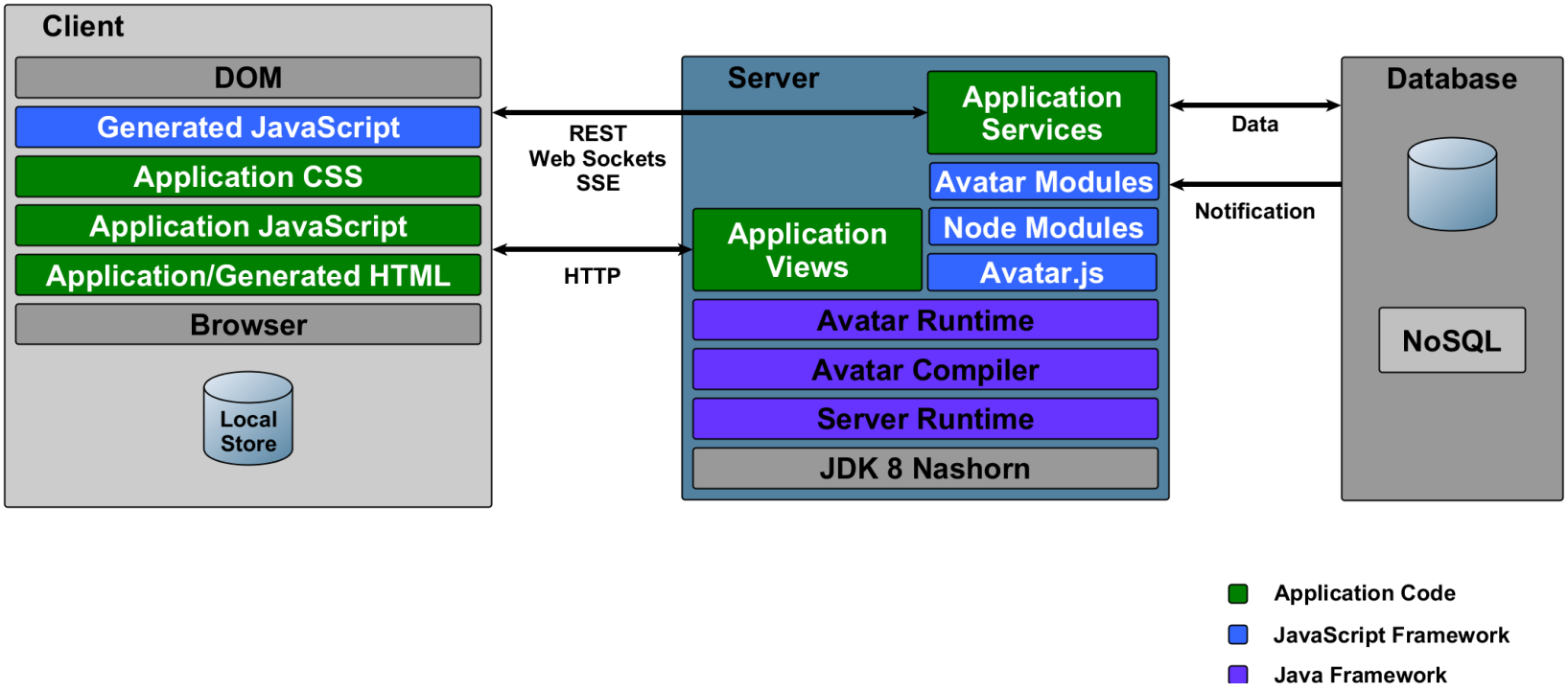
Java™ ORACLE®

Project Avatar (2)



- Application Code
- JavaScript Framework
- Java Framework

Project Avatar (3)



 |

Retire rarely-used GC Combinations

- Rarely used
  - DefNew + CMS
  - ParNew + SerialOld
  - Incremental CMS
- Large testing effort for little return
- Will generate deprecated option messages
  - Won't disappear just yet

Remove the Permanent Generation

- No more need to tune the size of it
- Current objects moved to Java heap or native memory
  - Interned strings
  - Class metadata
  - Class static variables
- Part of the HotSpot, JRockit convergence

Kleinere Anpassungen

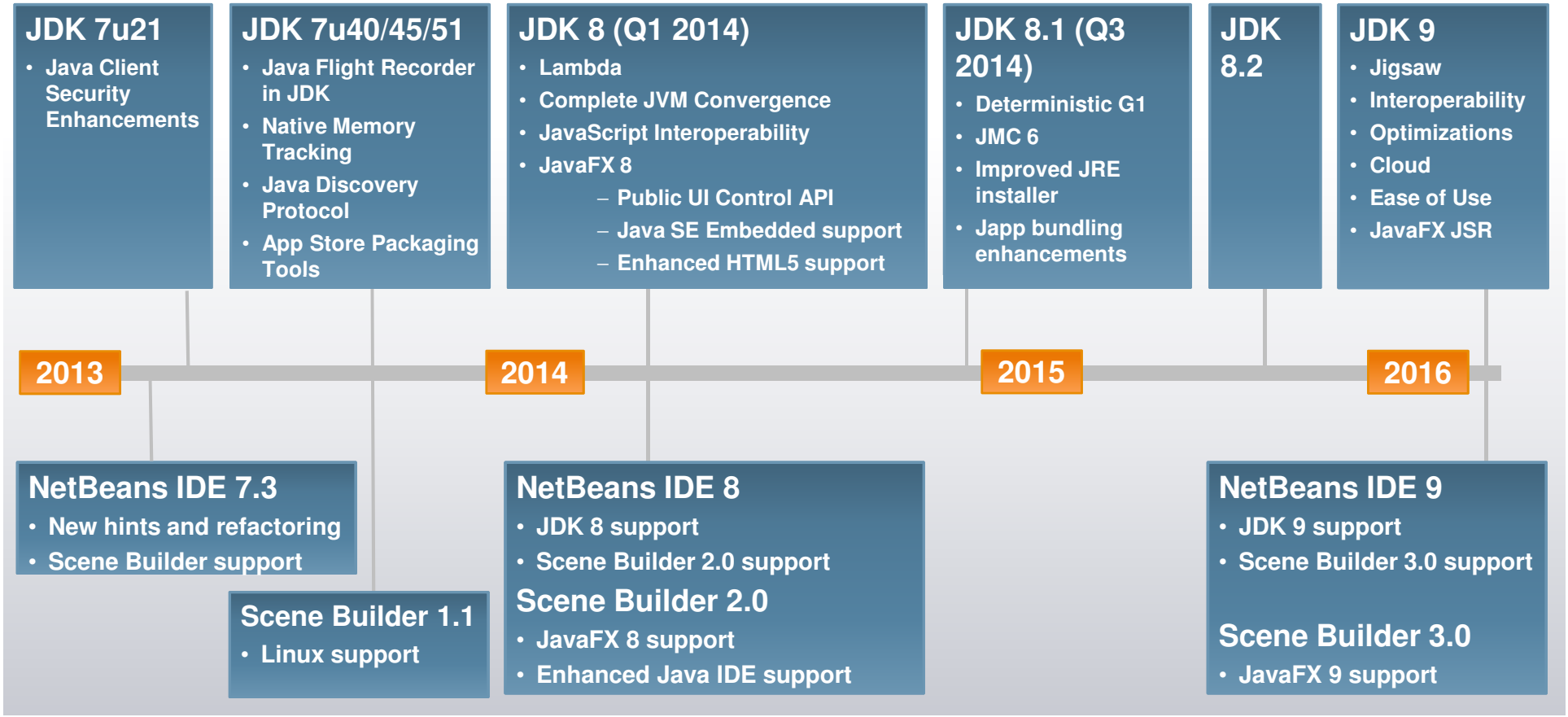
- Enhanced verification errors
  - Additional contextual information on bytecode verification errors
- Reduce cache contention on specified fields
  - Pad variables to avoid sharing cache lines
- Reduce class metadata footprint
  - Use techniques from CVM of Java ME CDC
- ***JEP Improve Contended Locking - Zurückgezogen***
  - ***Using research work form Oracle labs***
- Small VM
  - **`libjvm.so`** < 3MB by compiling for size over speed

Java Development Kit - JDK

Increased Build Speed, Simplified Setup

- Autoconf based build system
  - `./configure` style build setup
- Enhance javac to improve build speed
  - Run on all available cores
  - Track package and class dependences between builds
  - Automatically generate header files for native methods
  - Cleanup class and header files that are no longer needed

Java SE Roadmap



Zusammenfassung

- Java SE 8 enthält eine Menge neuer Funktionalität
  - Language
  - Libraries
  - JVM
- Java entwickelt sich kontinuierlich weiter
  - jdk8.java.net
  - www.jcp.org
  - openjdk.java.net/jeps

Java 8 Launch WebCast am 25. März 2014

- Java 8 is a revolutionary release of the world's #1 development platform
- It is the single largest upgrade ever to the programming model, with coordinated core code evolution of the virtual machine, core language, and libraries
- With Java 8, developers are uniquely positioned to extend innovation through the largest, open, standards-based, community-driven platform

Mark Reinhold
Chief Architect,
Java Platform Group,
Oracle

Peter Utzschneider
Vice President,
Product Management,
Oracle

Community Member

Brian Goetz
Java Language Architect,
Oracle

Roger Riggs
Consulting Member of Technical Staff,
Oracle

Jim Gough
Associate Leader,
London Java Community

John Rose
Java VM Architect,
Oracle

Robert Vandette
Consulting Member of Technical Staff,
Oracle

Richard Bair
Java Client Architect,
Oracle

Bruno Souza
Founding Member,
SouJava

Partner

Adam Messinger
Chief Technology Officer,
Twitter

 |

NightHacking Java 8 mit der JUG Darmstadt am 27. März 2014, 18:30 – 21:00 Uhr

 |

Vielen Dank für Ihre Aufmerksamkeit!

Wolfgang.Weigend@oracle.com

Twitter: @wolflook