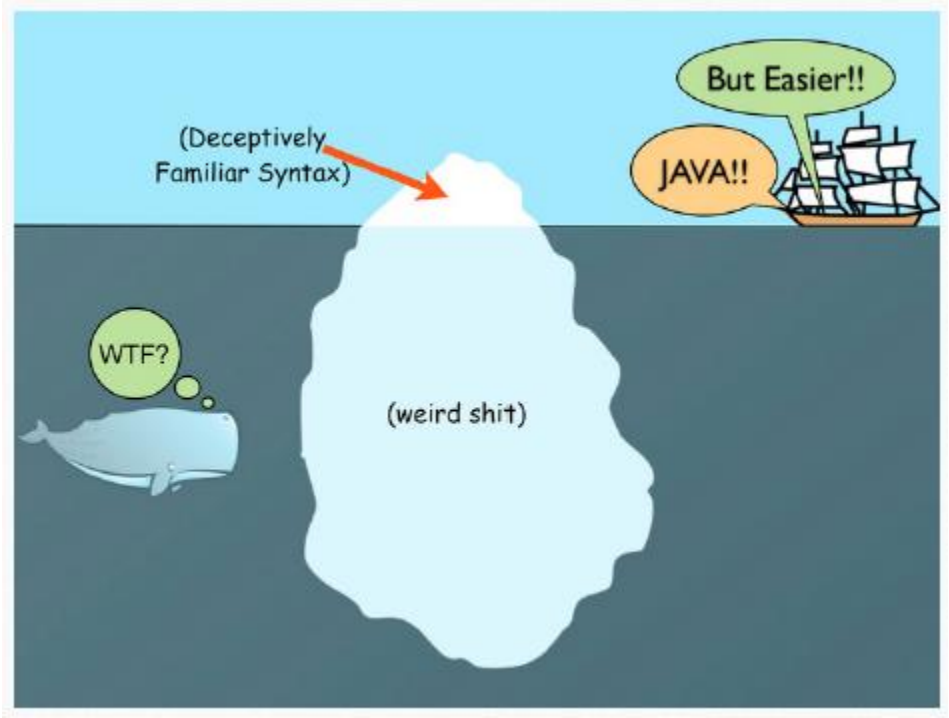
JavaScript on Java EE

Markus Eisele, @myfear
Developer Advocate
markus@jboss.org
September, 2014

Java Developers and JavaScript



Epilog:
What and Why?


LANGUAGE
SCRIPTING
GENERAL-PURPOSE
LANGUAGES
USED
ENVIRONMENT
CONTROL
EXAMPLES
SMALL
PARTICULAR
WITHIN
SOFTWARE
ALTERNATIVELY
OPERATE
SPECIAL
LOOSELY
SOMETIMES
JAVASCRIPT
ONE-BY-ONE
INTERPRET
UNIX-LIKE
NARROW
MANY
PERL
CODE
DEVELOPED
LEVEL
EXECUTED
OPERATOR
BASIC
INCLUDE
AWK
LATER
ARTICLE
SHELLS
PAGES
WRITTEN
STILL
STANDARD
APPLICATIONS
VIEWED
APPLICATIONS
PRIMARILY
CONVERSELY
DISCUSSED
ENVIRONMENTS
FUNCTIONALITY
VISUAL
AUTOMATE
COMMONLY
EXTENSIVELY
MAINFRAMES
OS
HIGH-LEVEL
DISCUSSES
SCRIPTING
HUMAN
SCRIPT
SYSTEMS
ALSO
LINES
USE
LUA
JOB
EDITOR
MICROSOFT
NOTABLY
REFERRED
WEB
ORIGINALLY
EXECUTION
ABSTRACTION
PROGRAMMING
FRANCHISE
SIMULATORS
ARTICLES
REFER
TERM
ENVIRONMENTS
HIGHLY
UNIX
SCRIPTS
RATHER
GAME
TASKS
TCL
OFTEN
KNOWN
RAILROAD
WIDELY
EXTENSION
DYNAMIC
TRAINZ
SED
PYTHON
SUPPORTS
TEXT-PROCESSING
OPERATING
DESIGNED
DIALECTS
SPECTRUM
DOMAIN-SPECIFIC
OFFICE
CASE
COMPILE
SENSE
HIGH
LISP
RANGES
PARTICULARLY
PROGRAMS
EMACS
BASH
AUTOMATED
BROWSER
DIALECT
SPECIFIC
ECMASCRIPT
BROWSERS
RUN-TIME
EXTENDED
EXTENSIONS
APPLICATION
PORTABLE
EMBEDDED
THOUSAND

redhat.

Categories

- Glue Languages
- Job Control
- Application Specific
- Embedded Languages

Static vs. Dynamic Typing

- Most scripting languages are dynamically typed
  - explicit type declarations not required
  - type information is attached to values, not to variables

- Java is static-typed
  - require variable type (declaration time)
  - only data of declared type

```
String answer = 42;
```

Type mismatch: cannot convert from int to String

1 quick fix available:

Change type of 'answer' to 'int'

Press 'F2' for focus

Weak vs. Strong Typing

- Java is a static, strongly typed langua
  - strongest possible c
    time
  - prevents mixing oper
- Many scripting languag
  - allow operations on **i**
  - implicit type conversi

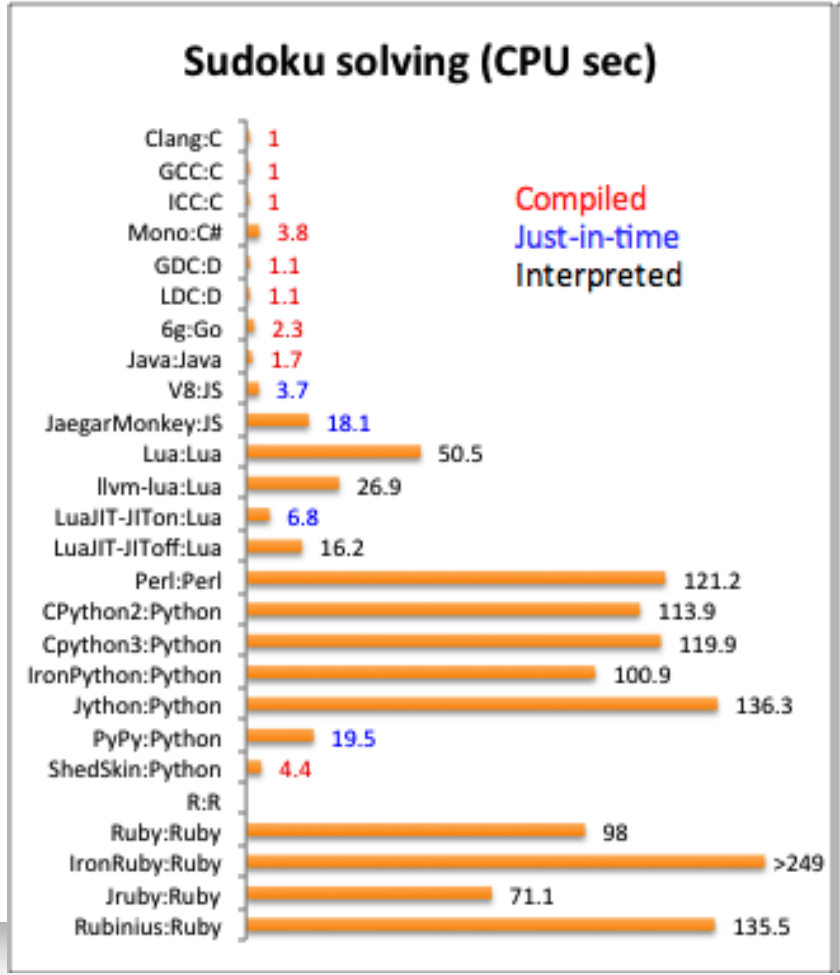
Groovy

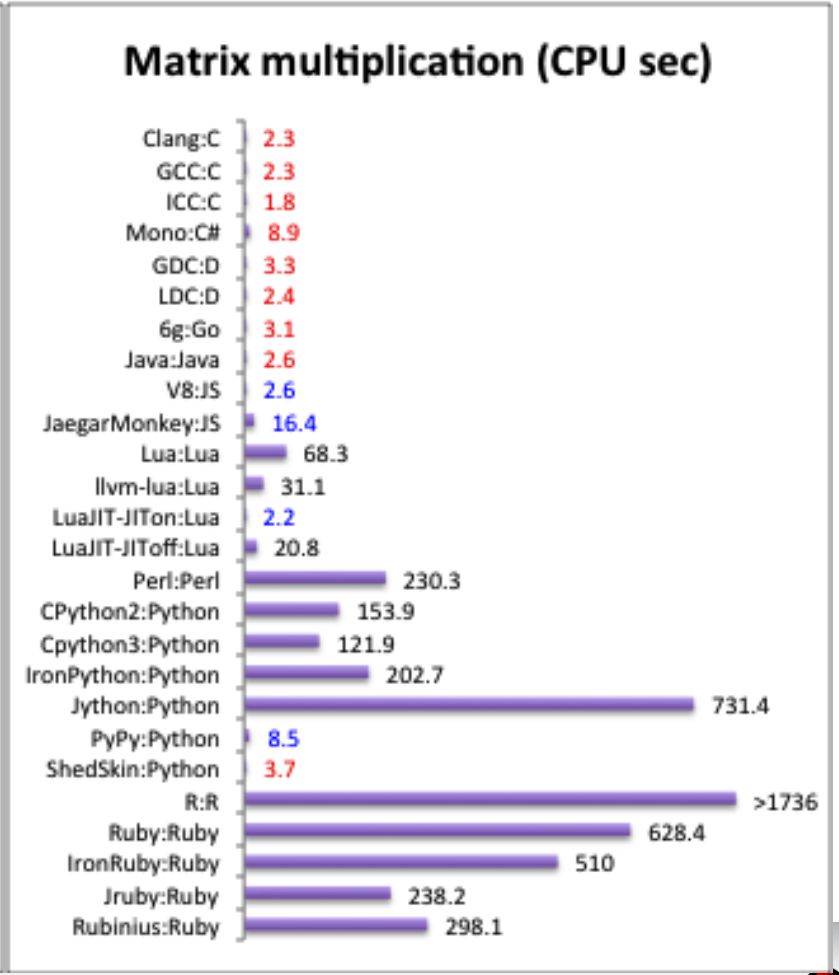
```
answer = 42
answer = answer / " Don't Panic"

java.lang.Integer.metaClass.div = { s ->
        if (!s.contains("Don't Panic"))
                throw new PanicException(s)
        “the answer is 42”
}
```

Quality and Performance

- Scripting languages are more compact and readable
  - less lines of code
  - weak typing not requiring the overhead of type declaration
- Fewer lines of code and less complexity means lower amounts of bugs, thus reducing development and maintenance costs.

- The missing type information has some disadvantages.
  - static, strongly typed languages ensure the robustness
  - type errors will be detected at compile time

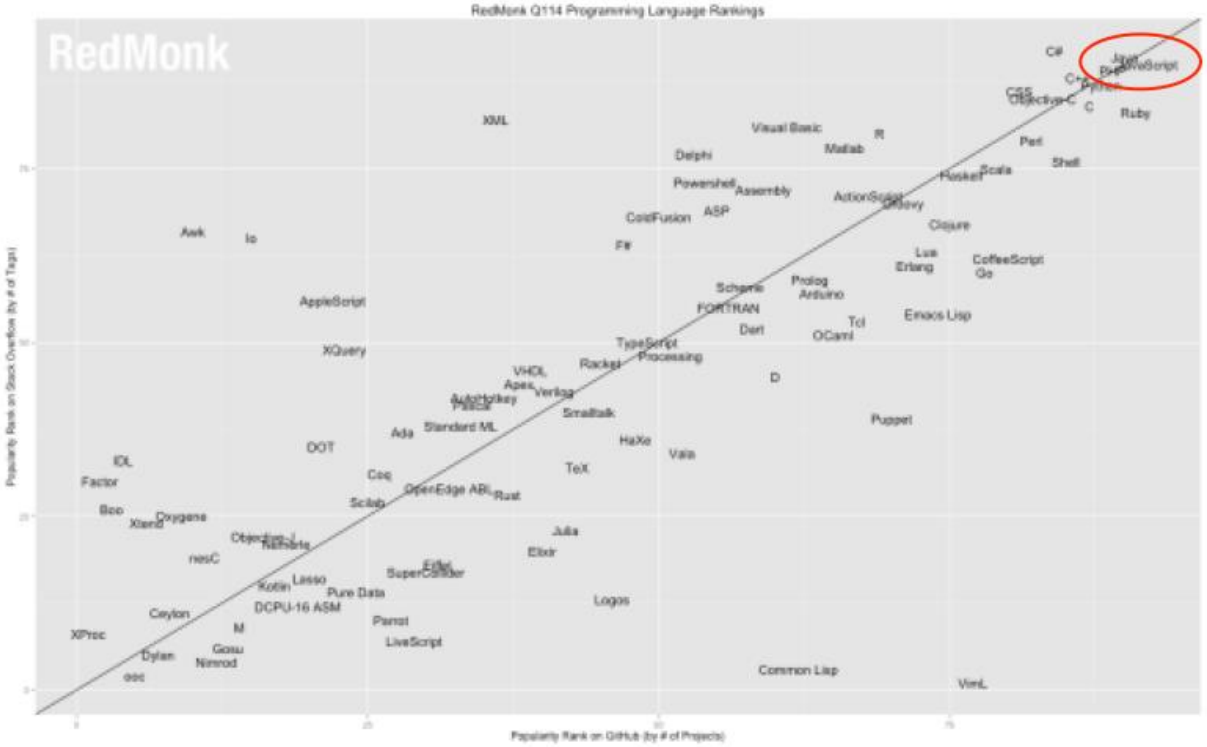




http://attractivechaos.github.io/plb/

redhat.

On the rise



http://redmonk.com/sogrady/2014/01/22/language-rankings-1-14/

redhat.

Personal Motivation

- It’s cool
- Try something different
- It’s fun!

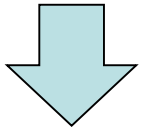
How to combine Java and JavaScript?

General ways

- compile
- Bean Scripting Framework (BSF)
- JSR 223 – Scripting for the Java Platform

Compile (e.g. groovyc)

```
println "Hello, ${args[0]}, the time is ${new Date()}"
```



```
Compiled from "helloToWorld.groovy" public class helloToWorld extends
groovy.lang.Script{ public static java.lang.Long __timeStamp; public static
java.lang.Long __timeStamp__239_neverHappen1283320660787; public
helloToWorld();
...
```

Script Engine

```
GroovyShell gs = new GroovyShell();
String script = "return 42";
int answer = (Integer) gs.evaluate(script);
```

```
Binding binding = new Binding();
binding.setVariable("foo", new Integer(2));
GroovyShell shell = new GroovyShell(binding);

Object value = shell.evaluate(
         "println 'Hello World!'; x = 123; return foo * 10");
assert value.equals(new Integer(20));
assert binding.getVariable("x").equals(new Integer(123));
```

http://groovy.codehaus.org/Embedding+Groovy

Bean Scripting Framework (BSF)

- http://commons.apache.org/proper/commons-bsf/
- Bean Scripting Framework (BSF) is a set of Java classes which provides scripting language support within Java applications, and access to Java objects and methods from scripting languages.

Bean Scripting Framework (BSF) – Example

```
BSFManager manager = new BSFManager();

manager.declareBean("a", 6, Integer.class);
manager.declareBean("b", 7, Integer.class);

String script = "var answer = a * b;" +
                "bsf.registerBean(\"answer\", answer)";

manager.eval("javascript", "blah", 0, 0, script);

Double answer = (Double) manager.lookupBean("answer");
assertEquals(42, answer.intValue());
```

JSR-223 – Scripting for the Java Platform

- The specification describe mechanisms allowing scripting language programs to access information developed in the Java Platform …
  https://jcp.org/en/jsr/detail?id=223

- Java 1.6+
- Rhino JavaScript for Java version 1.6R2
- javax.script.*

- jrunscript
  - http://docs.oracle.com/javase/6/docs/technotes/tools/share/jrunscript.html

ServiceLoader

- Since: 1.6
- http://docs.oracle.com/javase/6/docs/api/java/util/ServiceLoader.html
  - META-INF/services/javax.script.ScriptEngineFactory
  - This file contains the single line:

```
<jar jarfile="${clojure_jsr223_bundle}" basedir="${build_osgi}">
    <service type="javax.script.ScriptEngineFactory“
             provider="de.torq.clojure.jsr223.ClojureScriptEngineFactory"/>
```

- Clojure JSR 223 build.xml
  - https://github.com/pmf/clojure-jsr223/blob/master/build.xml

JSR-223 – script engines

redhat.

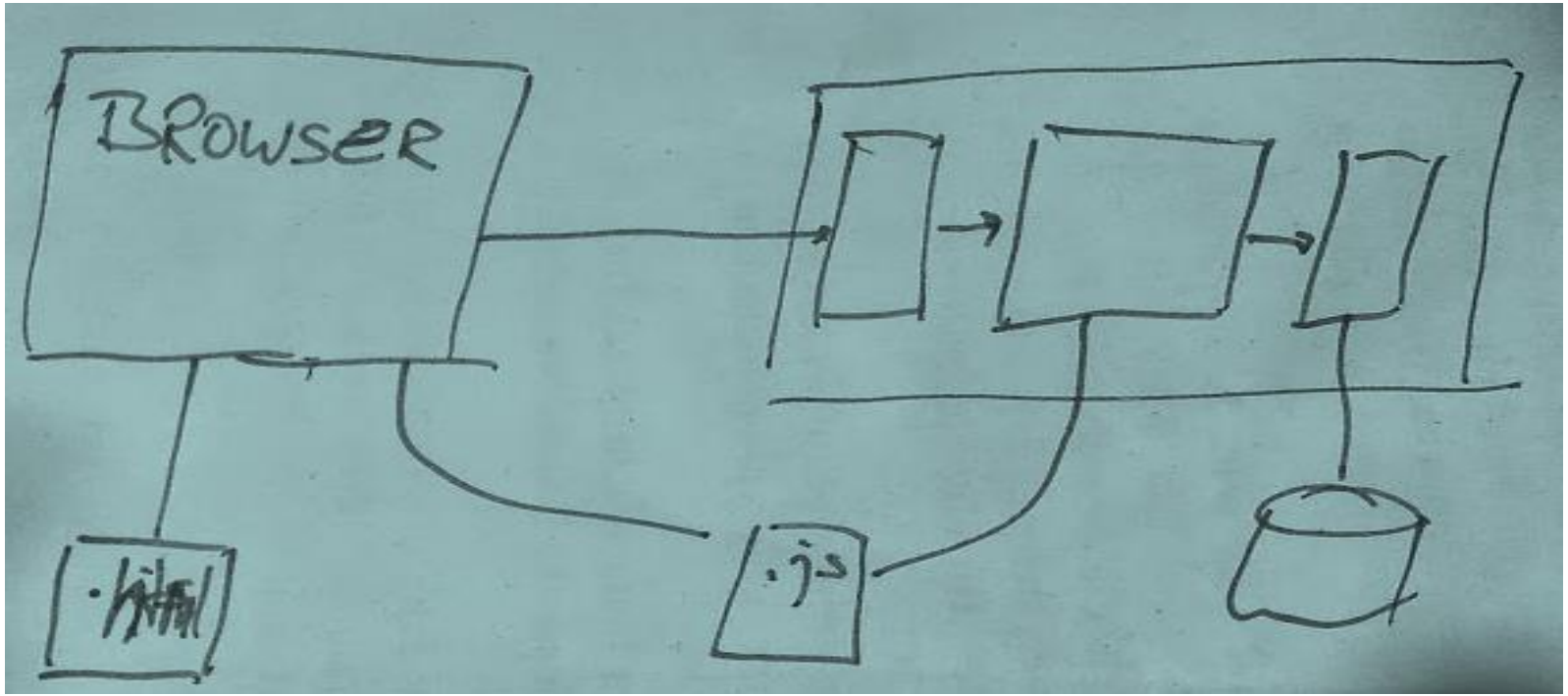
Nashorn

- ECMAScript 5.1 compliant
- Bundled with JDK 8
  - Replaces Rhino
  - Faster (2x – 10x)
  - More secure
- Seamless Java ←→ JavaScript interoperability

```
var Button = javafx.scene.control.Button;

var button = new Button();
button.text = "Say 'Hello World'";
button.onAction = function() {
    print("Hello World!");
}
```

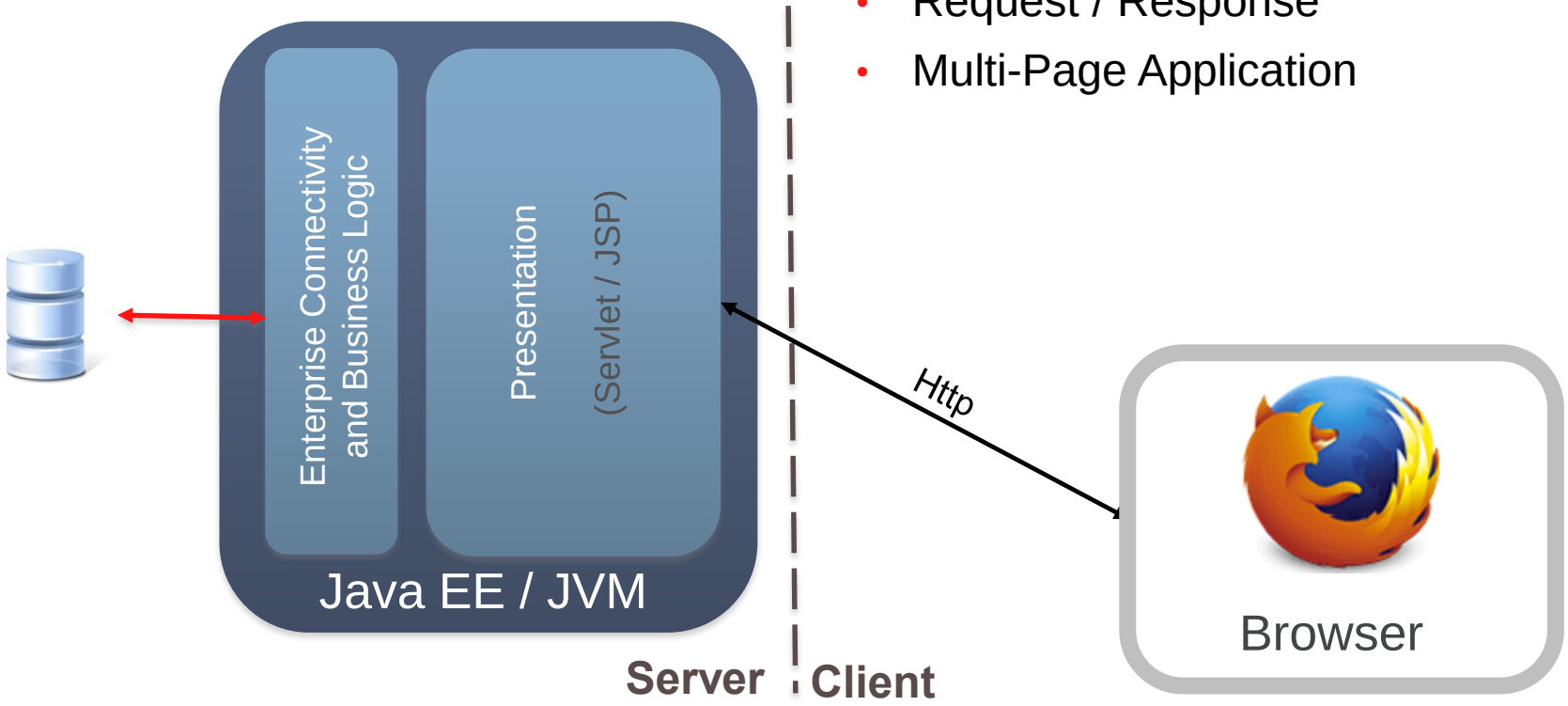
http://download.java.net/jdk8/docs/technotes/guides/scripting/nashorn/index.html

Where does EE fit in?

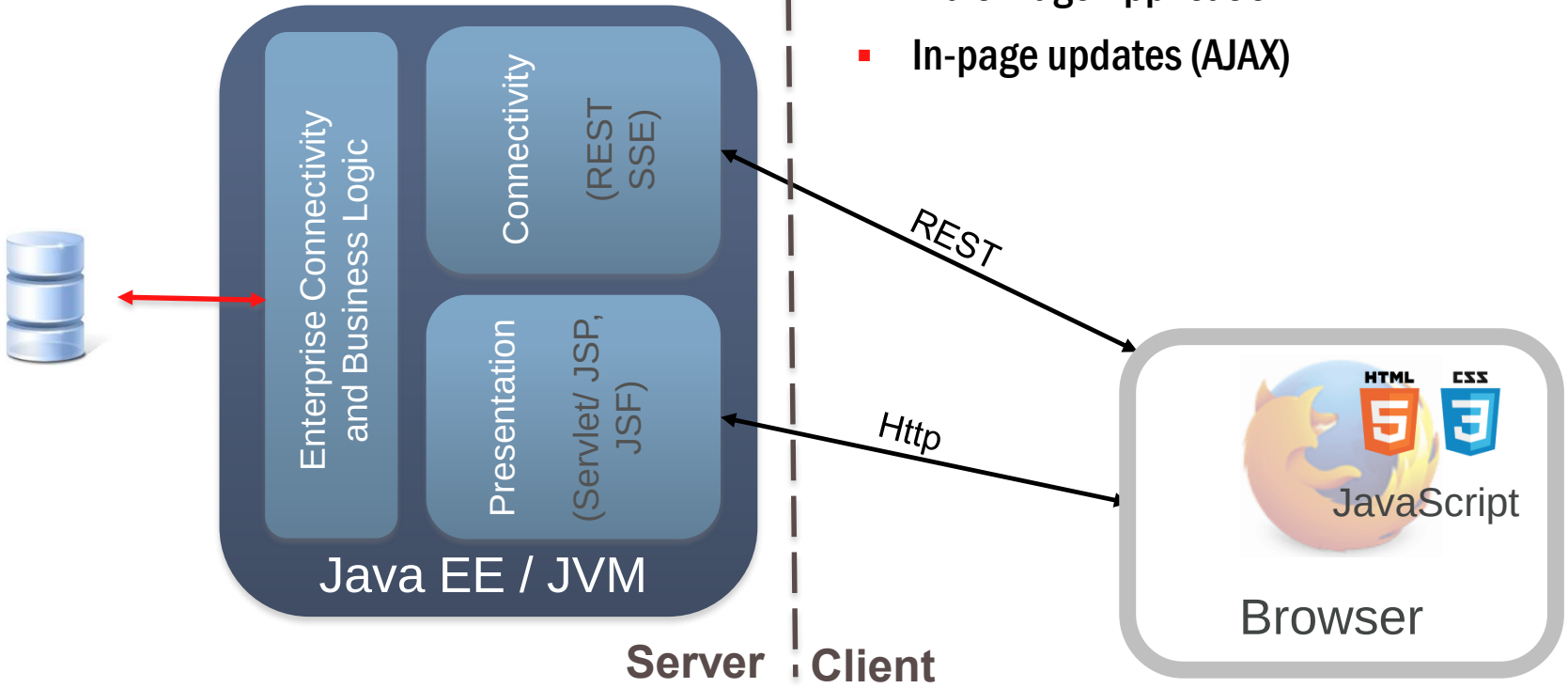


Evolution of web architecture

- Request / Response
- Multi-Page Application

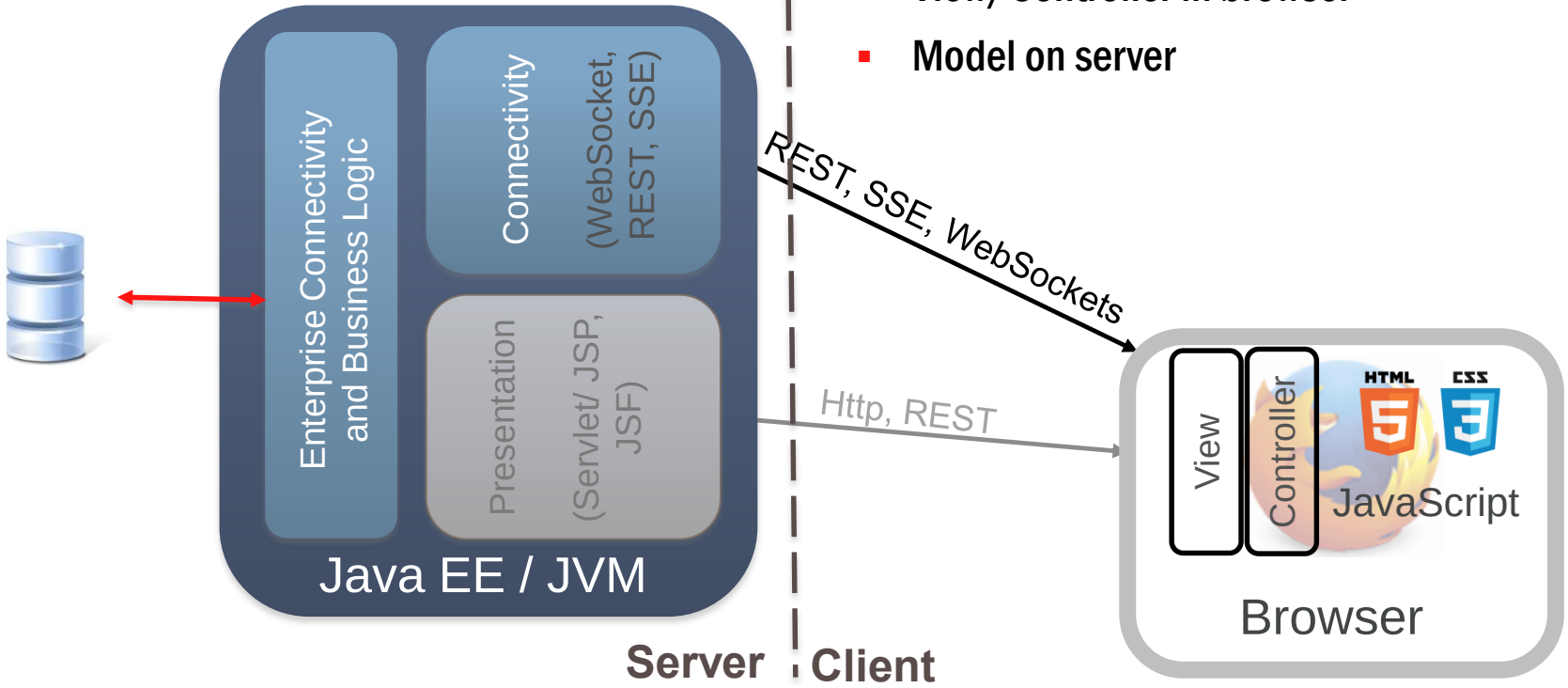


- Multi-Page Application
- In-page updates (AJAX)



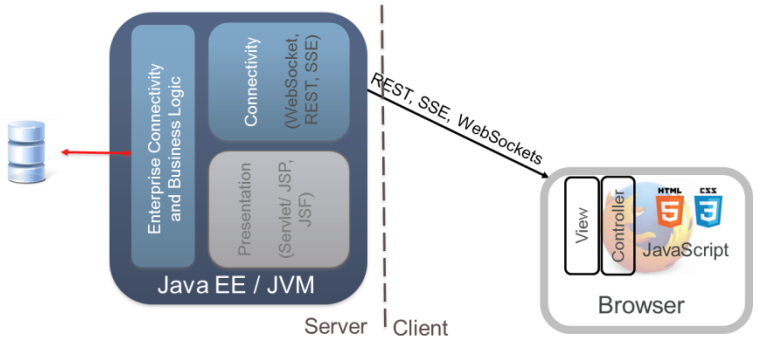
redhat.

- Single-page applications
- View/Controller in browser
- Model on server



redhat.

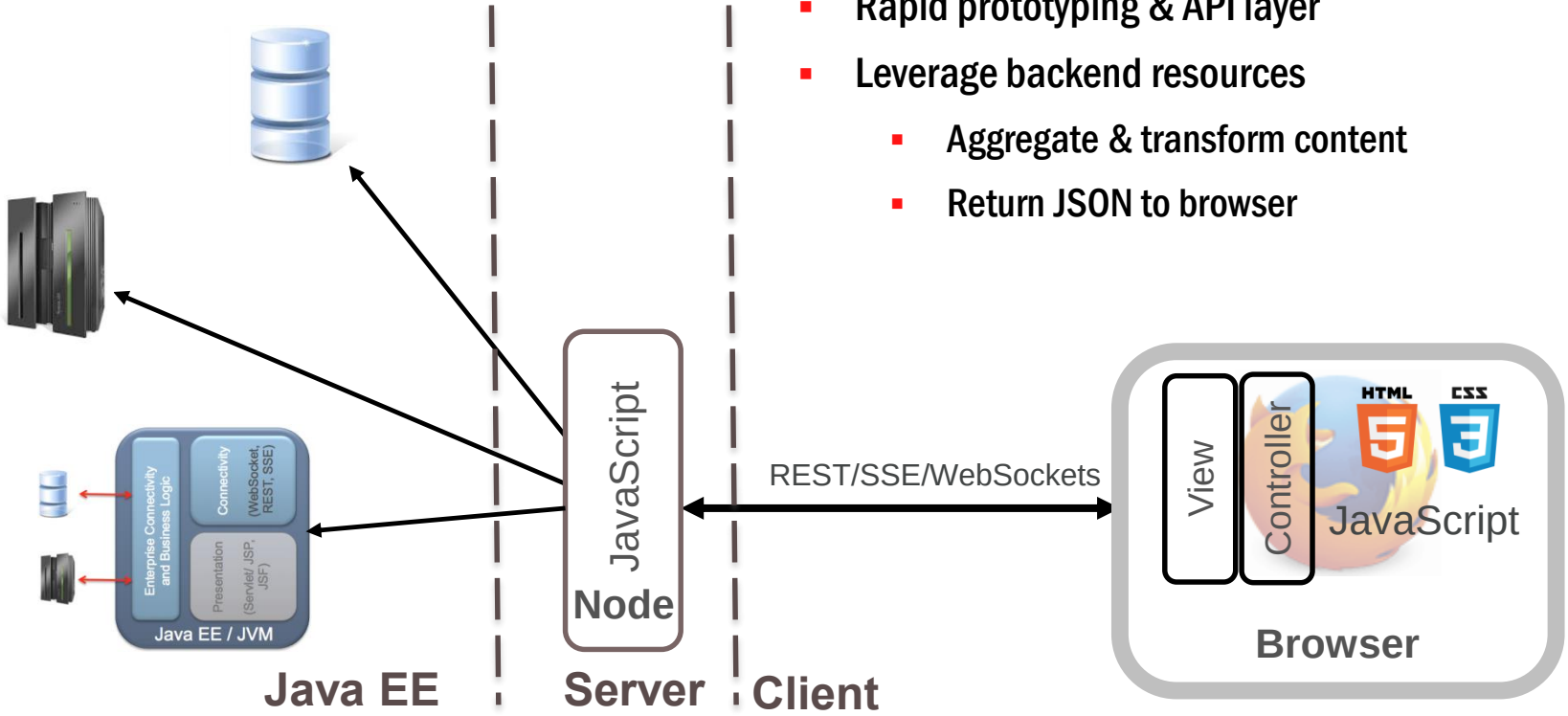
Problems?



- No reuse
- Duplicate implementations
- Mixed skills in projects
- Different cultures
- Communication overhead
- Client Team waits for Server Team
- ...

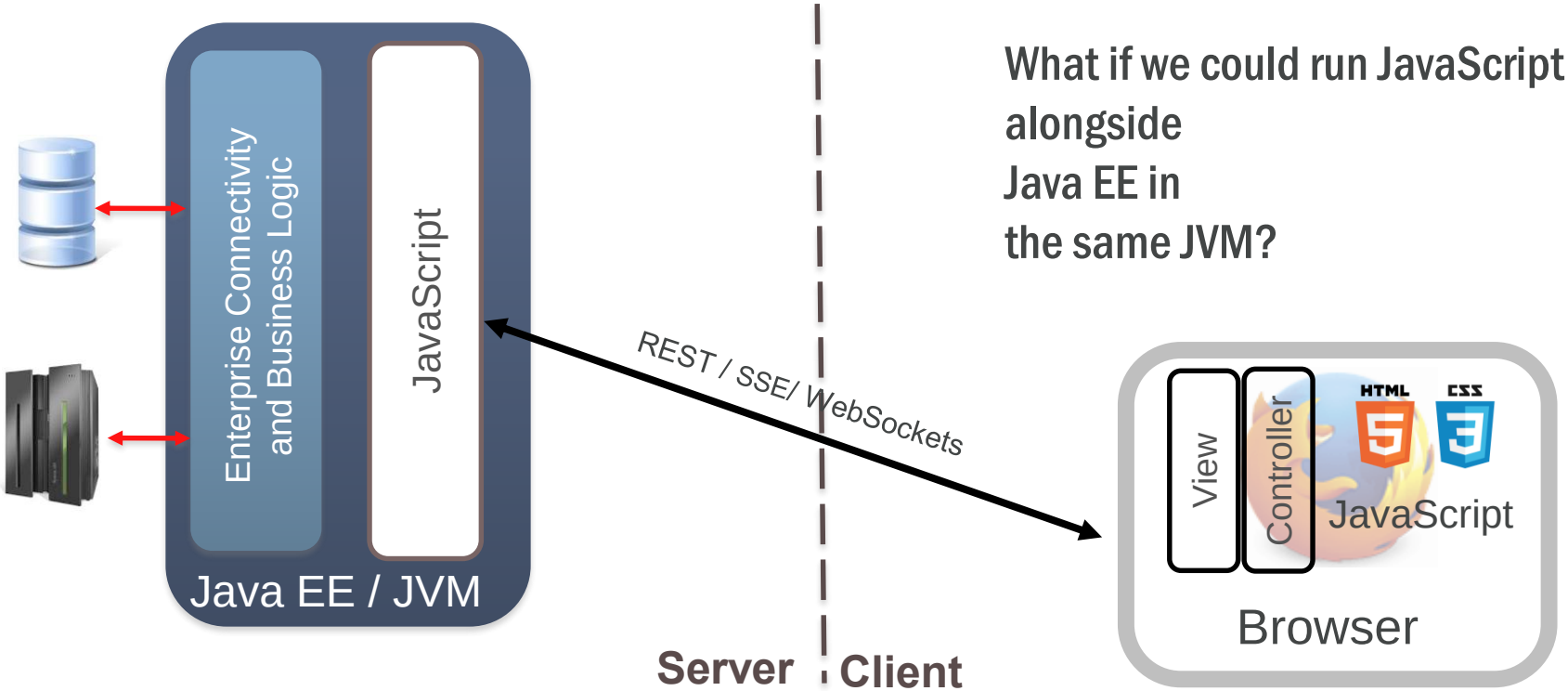
Today?

- Project-based end-to-end JavaScript
- Rapid prototyping & API layer
- Leverage backend resources
  - Aggregate & transform content
  - Return JSON to browser



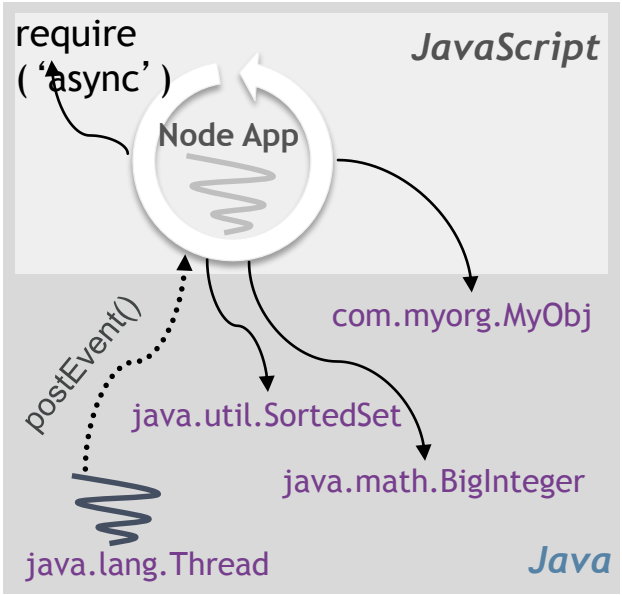
redhat.

Tomorrow?



What if we could run JavaScript alongside Java EE in the same JVM?

Avatar.js = Node + Java



JVM Process

- Node Programming Model
  - Code in JavaScript
  - Single event loop / thread
  - Require (import) Node modules
- Invoke Java code
  - Java types and libraries
  - `new java.lang.Thread();`
  - `new com.myorg.MyObj();`

Your Ideas?