Java is a trademark of Sun Microsystems, Inc.

Java

JavaOne℠

VJET bringing the best of Java™ and JavaScript™ together

Greg Choi, Justin Early, and Yitao Yao

eBay

Why are we here?

Java™

- ✔ Large pool of talent
- ✔ Established patterns
- ✔ Sophisticated tools
- ✔ Standardized technologies
- ✔ Large collection of commonly used libraries

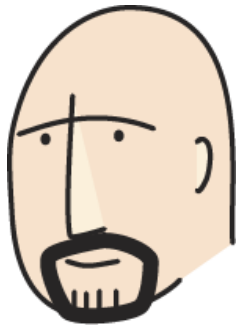
JavaScript™

- ✘ Difficult to find talent
- ✘ Increasing need for client side, interactive behavior
- ✘ Managing complexity is difficult
- ✘ Browser incompatibilities hamper quality
- ✘ Multiple library use is problematic

What's Missing in JavaScript

- Contract based design and programming
  - strongly typed API
  - interfaces
- Namespace management
  - packages
- Data encapsulation
  - visibility and access control
- Classic inheritance
- Dependency management
  - imports

Our Answer:



Semantic Equivalence

- Bring the best of Java to JavaScript
- Put the “Java” in JavaScript

How?

- Add typing system to JavaScript
- Enable Java-like modeling concepts and programming constructs for JavaScript

Why Java-Like?

- Proven modeling and programming paradigm
- Proven tooling and process support
- Proven manageability of large-scale enterprise applications

Break down the barriers between Java and JavaScript

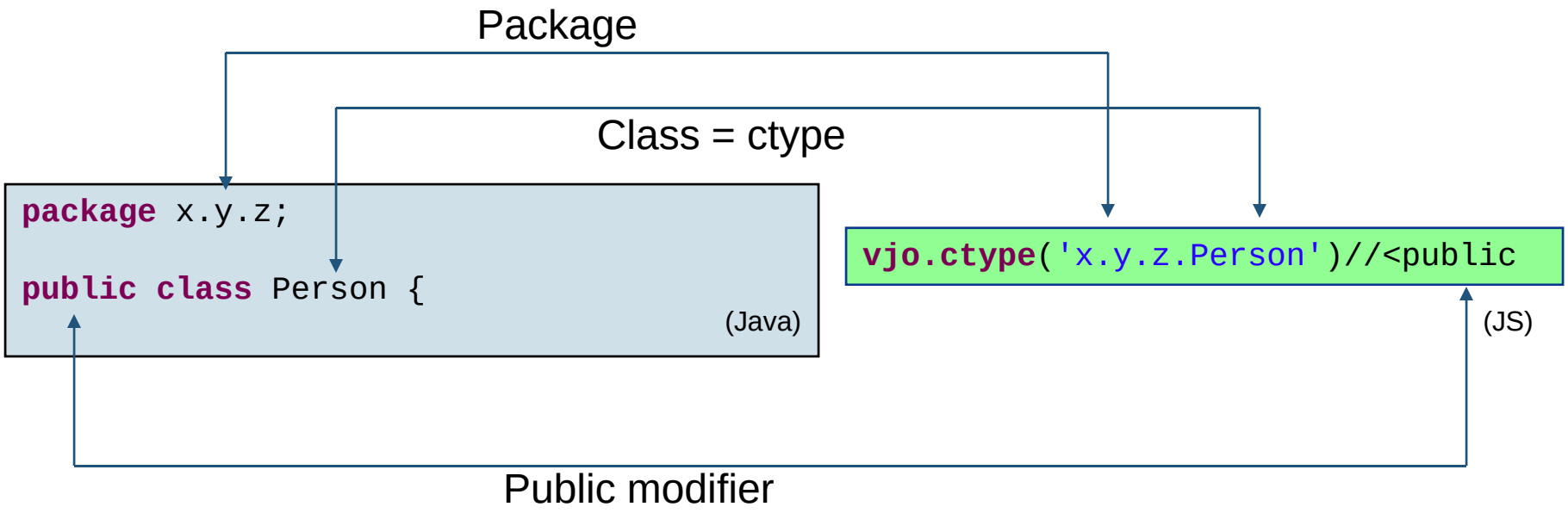
AND

Enable large scale JavaScript development

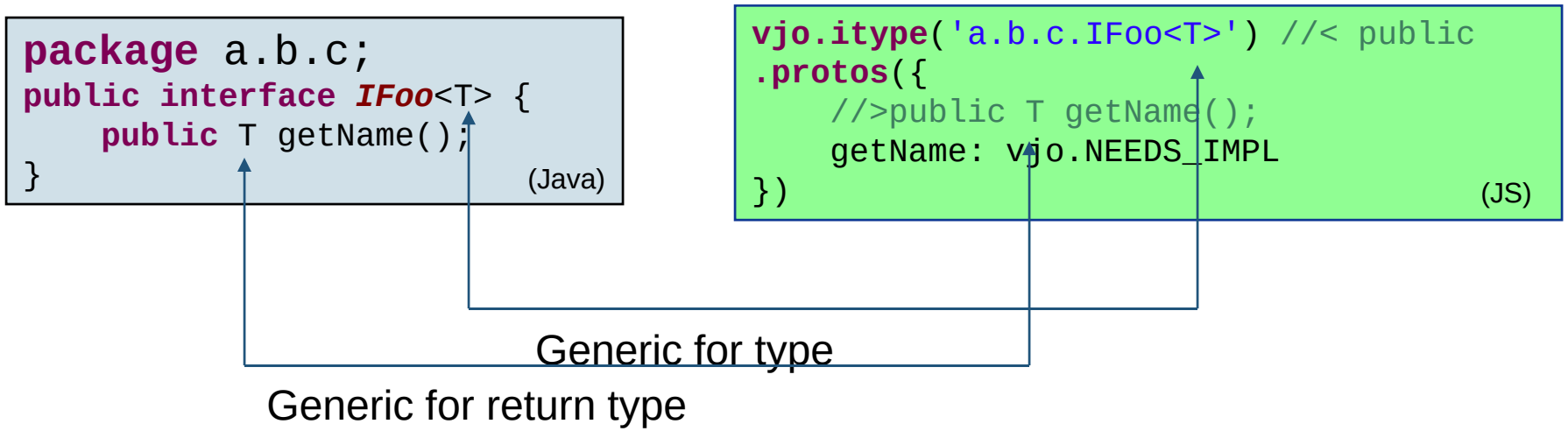
Bring the Best of Java™ to JavaScript™

- package, imports
- class, interface, enum, abstract class, inner and anonymous class, *mixin*, *o-type*
- extends and implements
- generics
- static and instance
- formal constructors, properties and methods
- static initialization
- super and override
- overloaded constructors and methods
- modifiers: public, protected, private, final

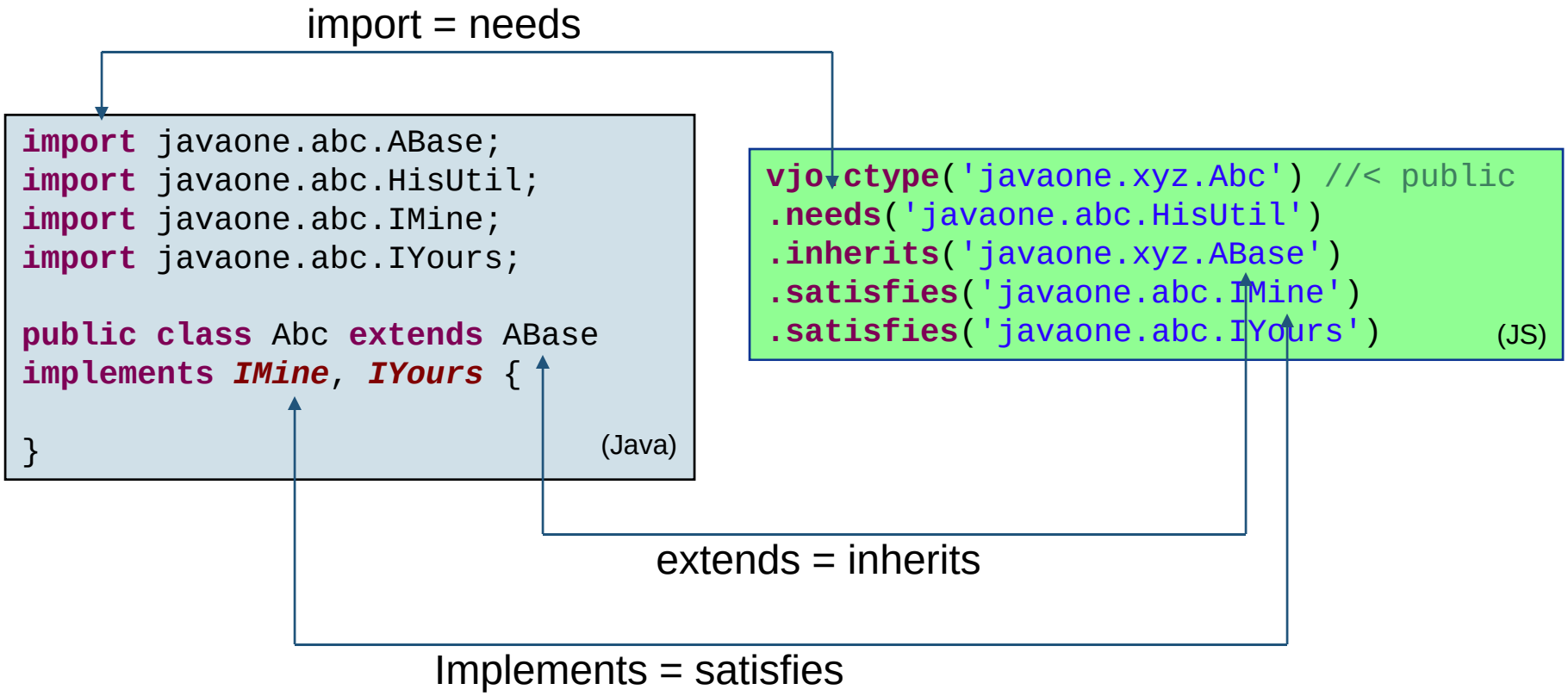
VJET Package and Class



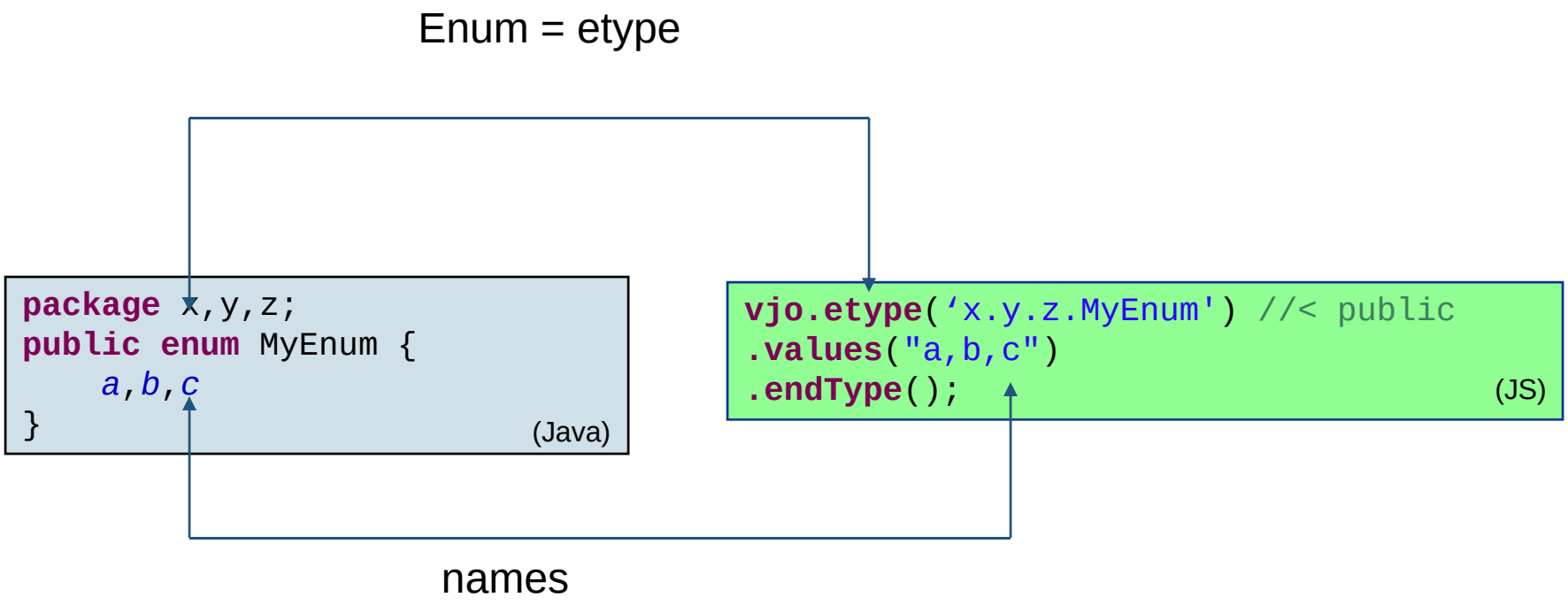
VJET Interfaces and Generics



VJET extends, implements, import



VJET Enum


Enum = etype
package x,y,z;
public enum MyEnum {
a,b,c
}
(Java)
vjo.etype('x.y.z.MyEnum') //< public
.values("a,b,c")
.endType();
(JS)
names

VJET Static and Instance Properties

```
public static final int MAX = 12;
```

```
private String m_name = “Justin";
```

(Java)

```
.props({
    MAX:12 //< public final int
})
.protos({
  m_name:"Justin"//<private String
})
```

(JS)

VJET Typing and Semantic Declaration

- VJET comment syntax uses familiar Java-like declaration syntax, with a directional indicator: //> or //<
- VJET comment syntax can be expressed on one line

```
.protos({
    m_mother:null, //< private final Tiger

    //> public void hunt(IAnimal pray)
    hunt:function(pray){
        this.roar();
        this.base.hunt(pray);
    },
```

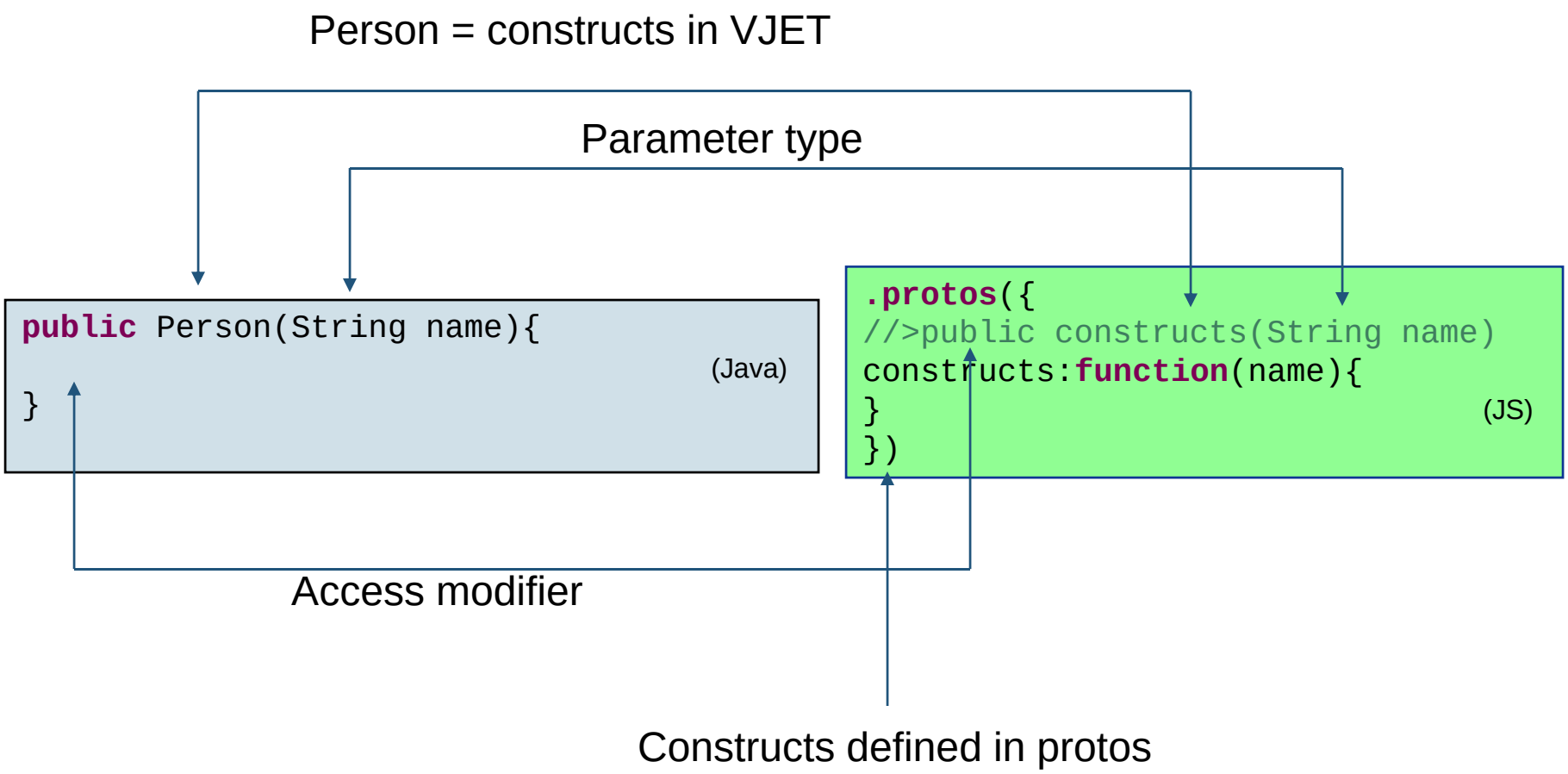
(JS)

final Modifier

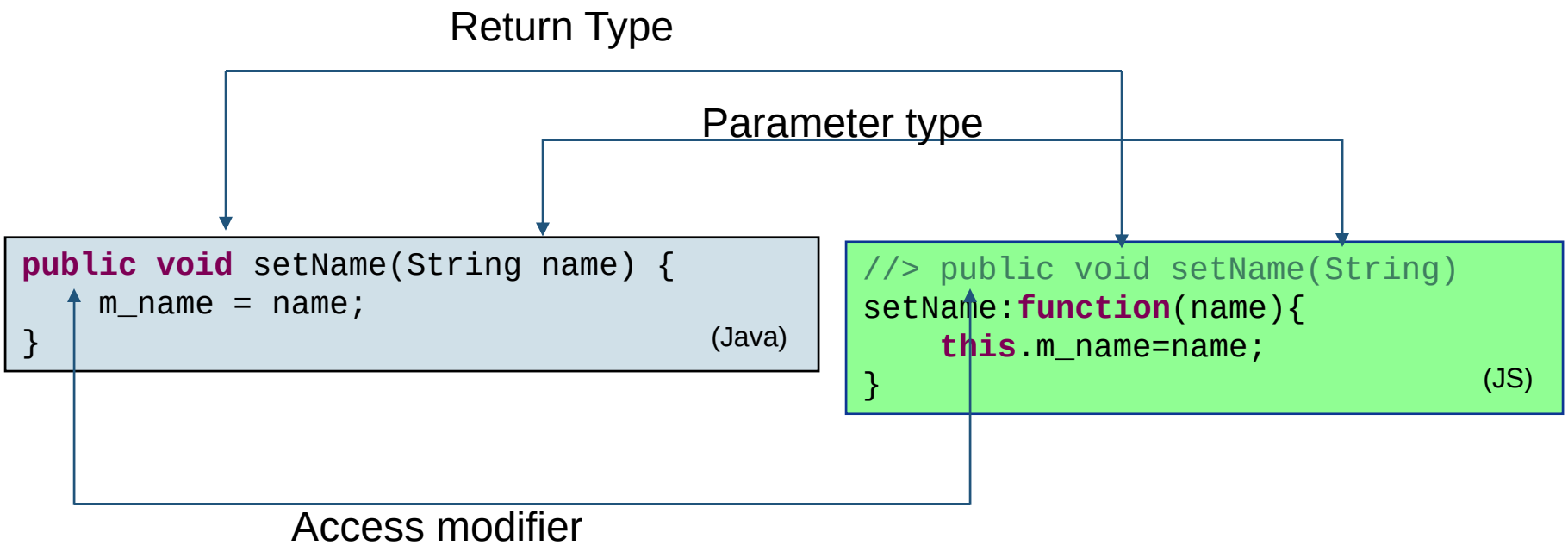
Access Modifier

Type

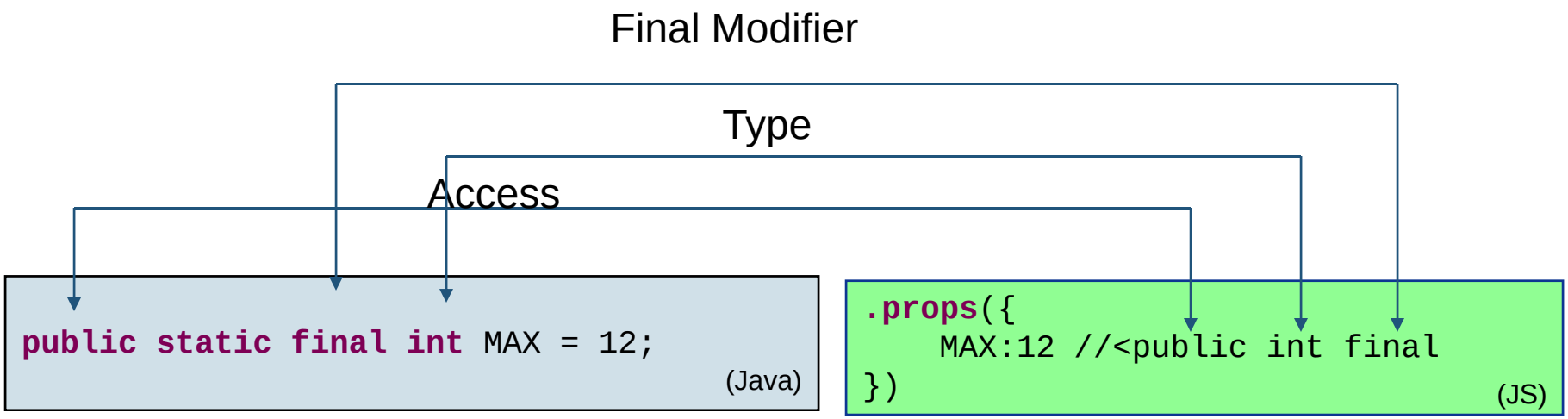
VJET Constructor



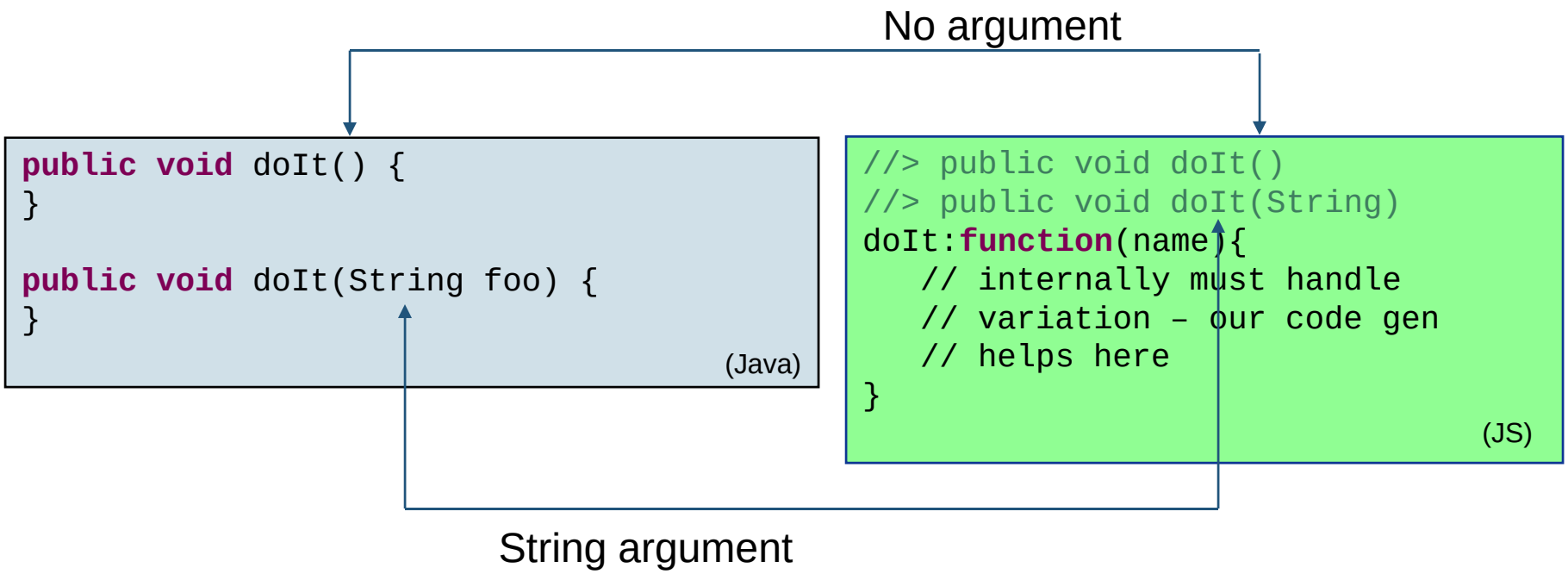
VJET Methods



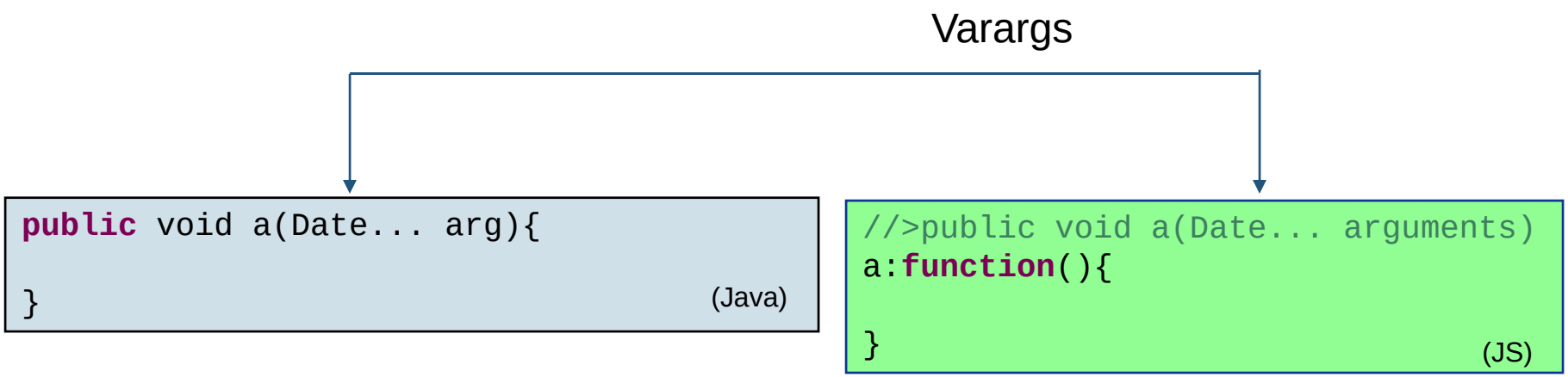
VJET Static Property



VJET Overloading



VJET Varargs

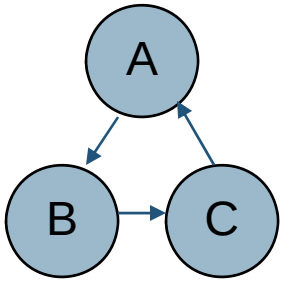


VJET Static Initialization

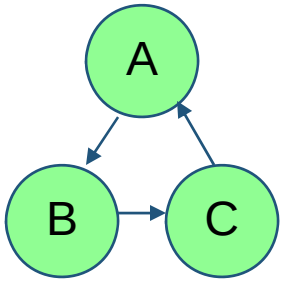
```
static {
    String mystatic = “test”;
}
```
(Java)

```
vjo.ctype('x.y.z.Person')
.inits(function(){

   var mystatic = “test” //<String
}).endType();
```
(JS)



Class loader – handles static init
Where a circular dependency exists



VJET Runtime – handles static init
Where a circular dependency exists

Sample VJET Type

```
vjo.ctype('javaone.j4j.oom.Tiger') //< public
.needs(['javaone.j4j.oom.IAnimal','vjo.java.lang.System'])
.inherits('javaone.j4j.oom.Cat')
.satisfies('javaone.j4j.oom.IPanthera')
.props({
    //> public String getSpeciesType()
    getSpeciesType:function(){
        return "CAT";
    }
})
.protos({
    m_mother:null, //< private final Tiger
    //> public constructs(Tiger mother)
    constructs:function(mother){
        this.base();
        this.m_mother=mother;
    },
    //> public void hunt(IAnimal pray)
    hunt:function(pray){
        this.roar();
        this.base.hunt(pray);
    },
    //> public void roar()
    roar:function(){
        this.vj$.System.out.println("I am tiger");
    }
})
.endType();
```

(JS)

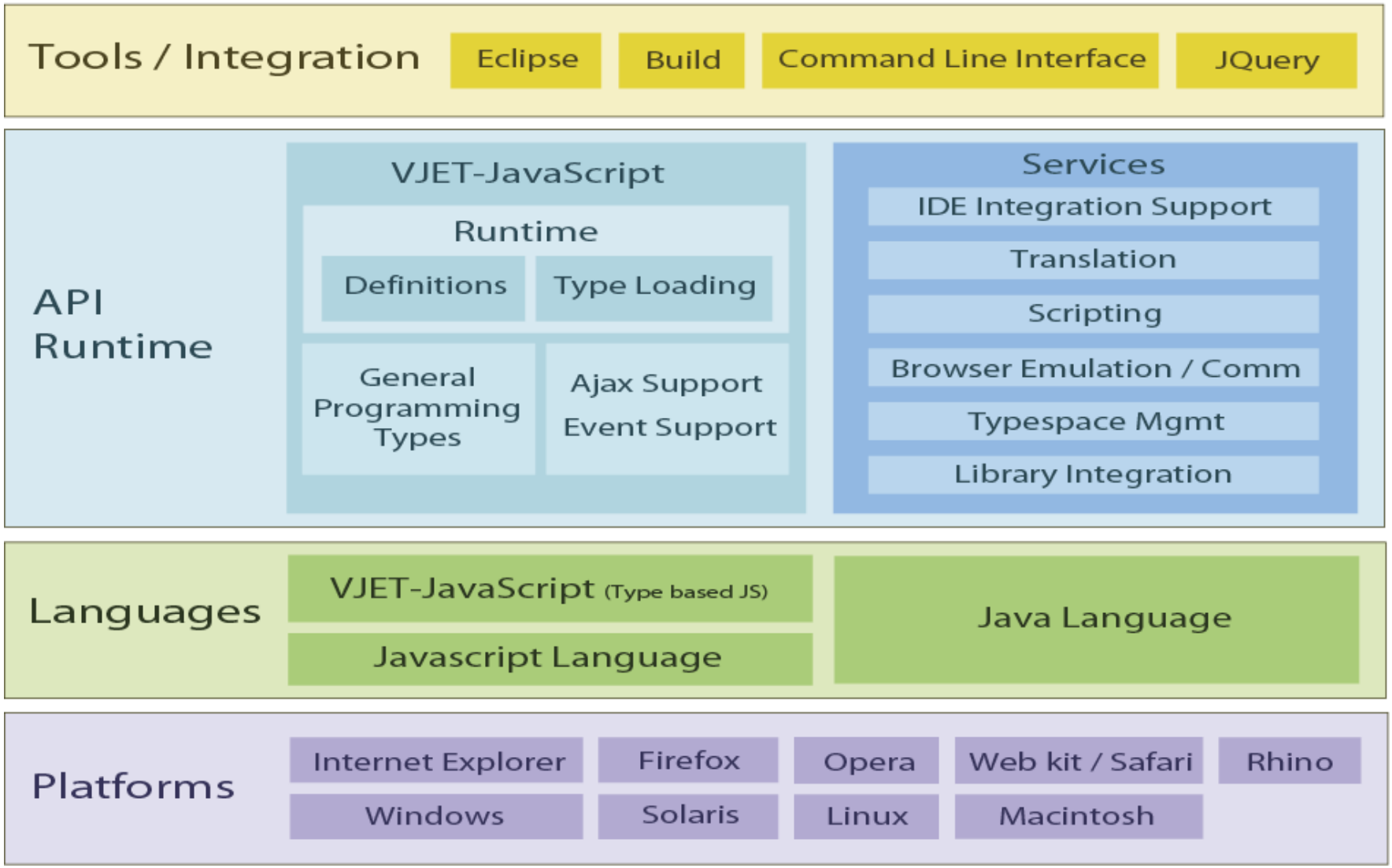
What is VJET?

- Not another widget library
- Not an application framework

- Java and JavaScript + Enterprise technology
  = A super boost for productivity

- VJET blends the best of Java and JavaScript language and tooling, for a powerful development experience

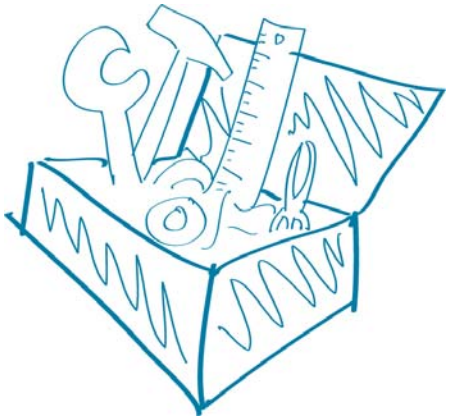
VJET Diagram



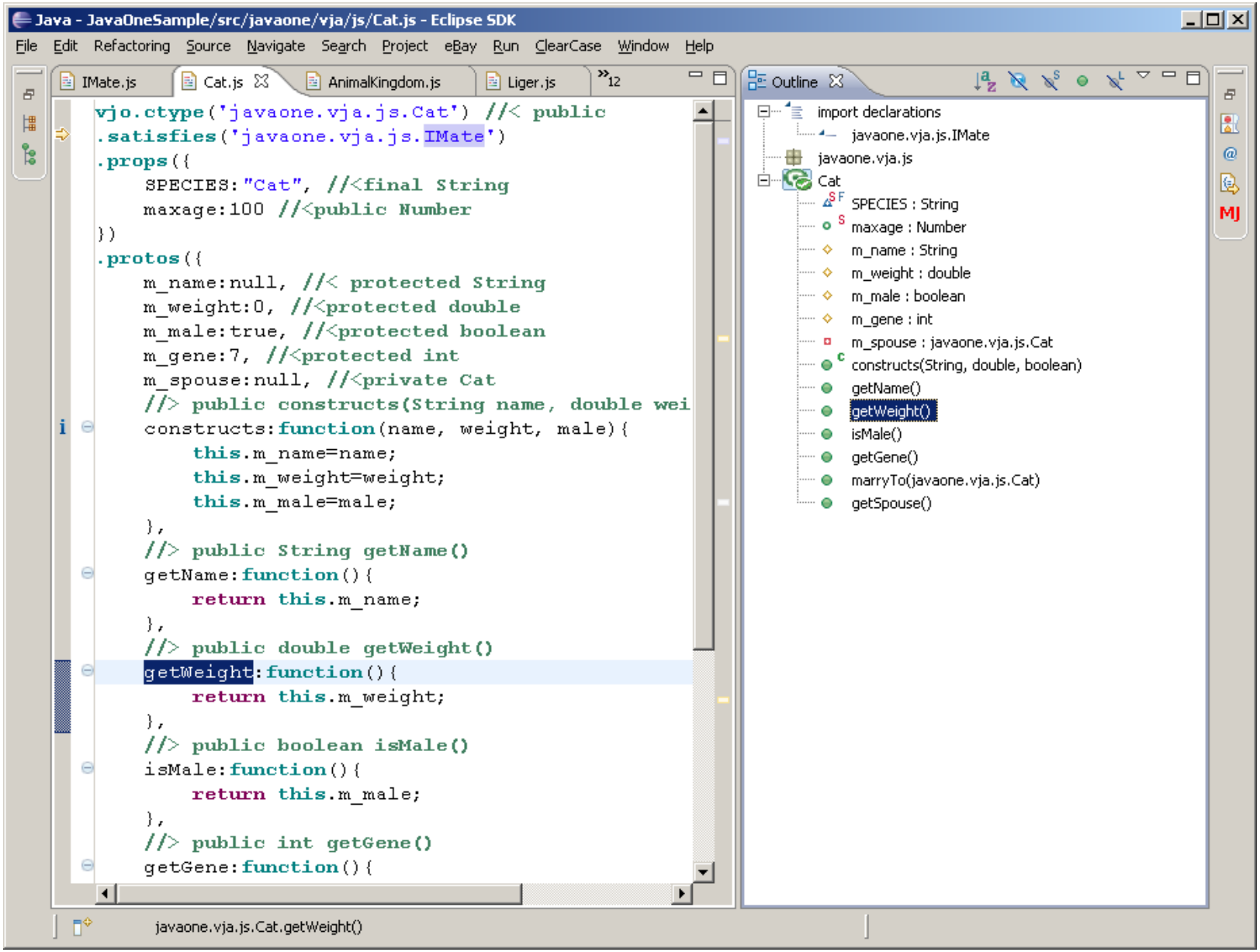
VJET Type System

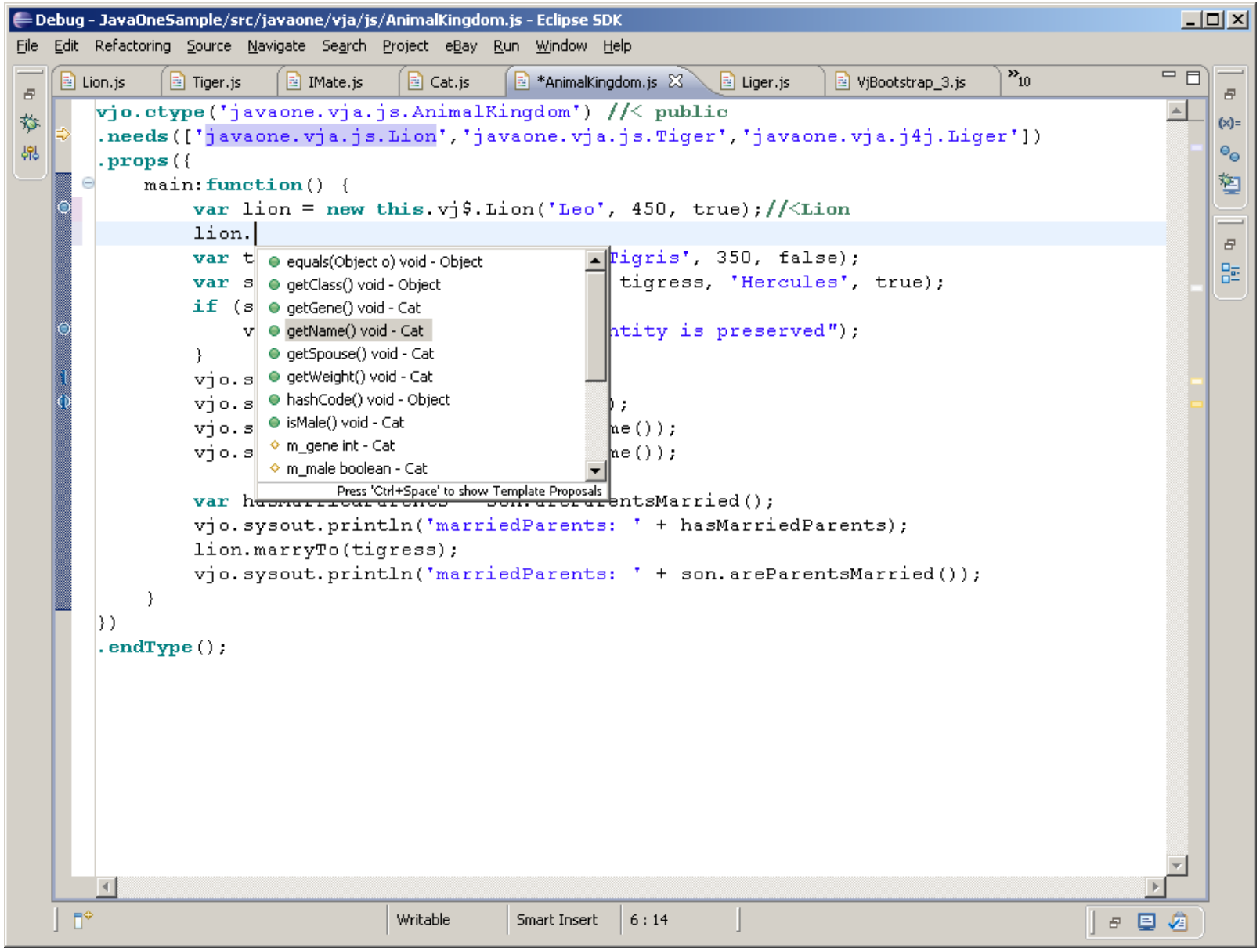
- VJET Parser handles VJET structural definition and VJET comment syntax, on top of JavaScript syntax
- VJET Abstract Syntax Tree (JST), analogous to Java AST, represents Java-like types and semantics in addition to JavaScript language constructs
- VJET Type Space Management supports:
  - indexed type, hierarchy, reference and dependent search
  - update, refactoring
  - syntactic and semantic validation
  - tool integration and more …
- VJET Type Meta enables “typed” Object Serialization

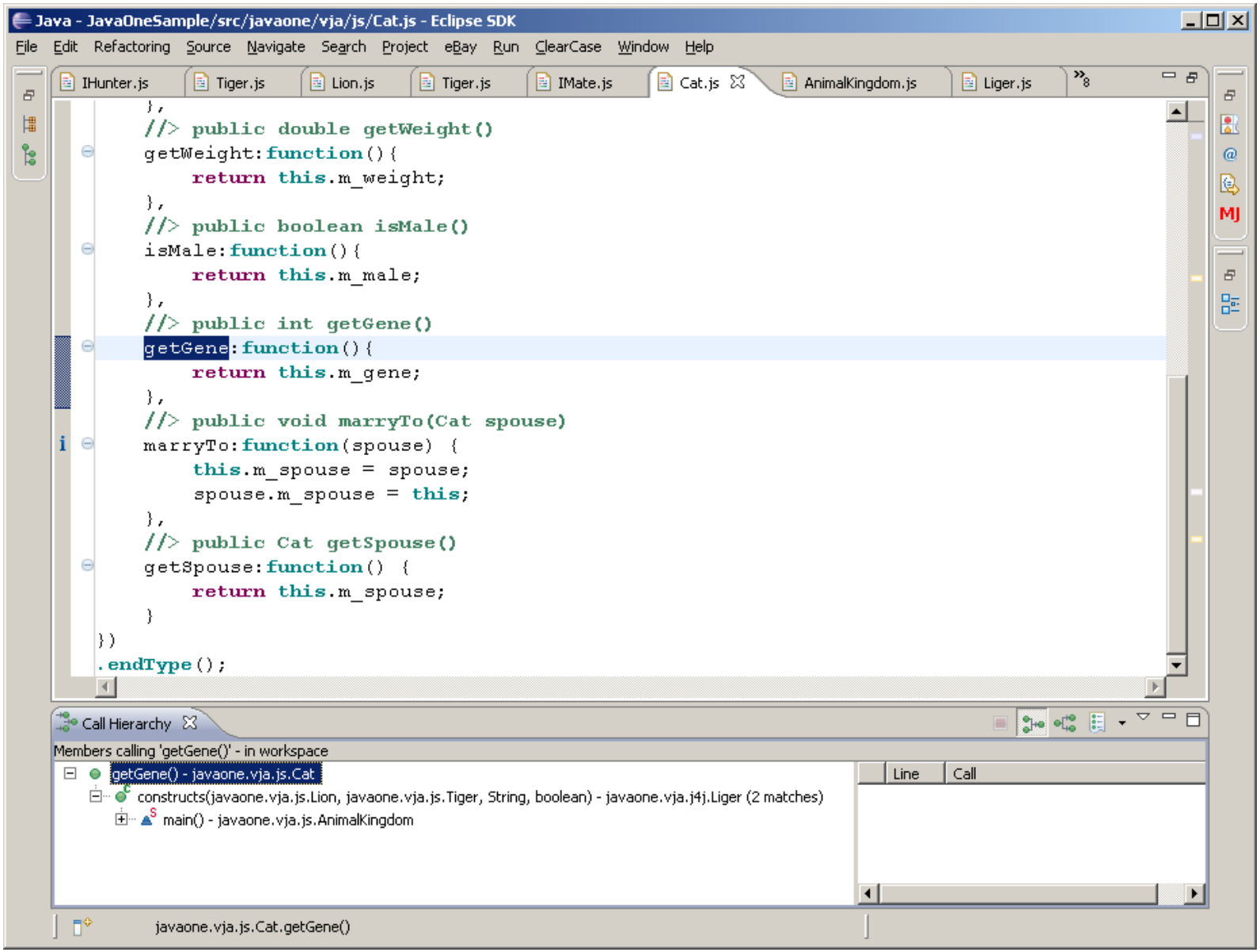
Tooling in VJET IDE

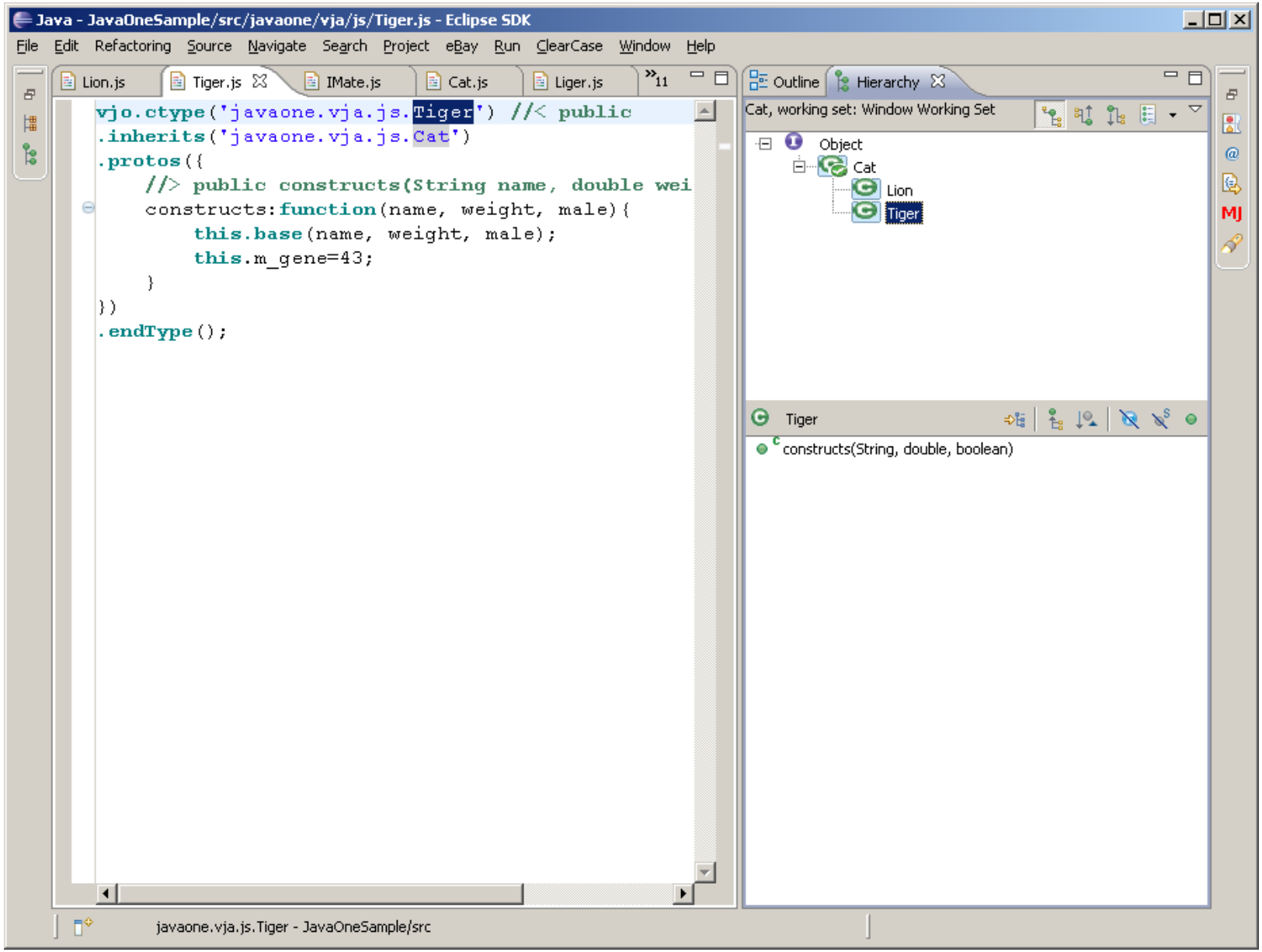


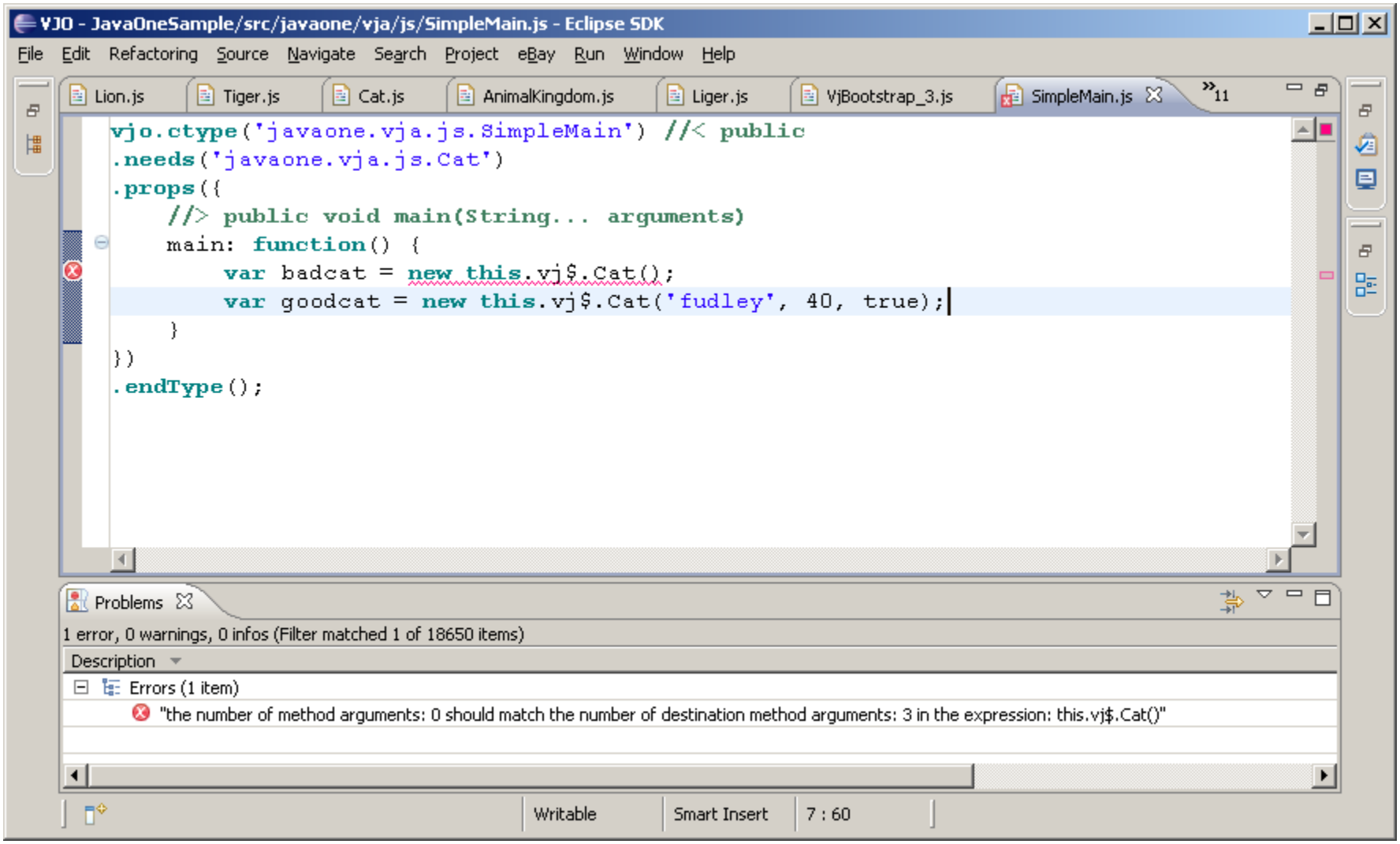
- Code assistance and completion
- Search for types, references, declarations
- Type hierarchies and call hierarchies
- Validation – not only syntax but also semantic!
- Refactor and even browser specific functionality help
- Many more …

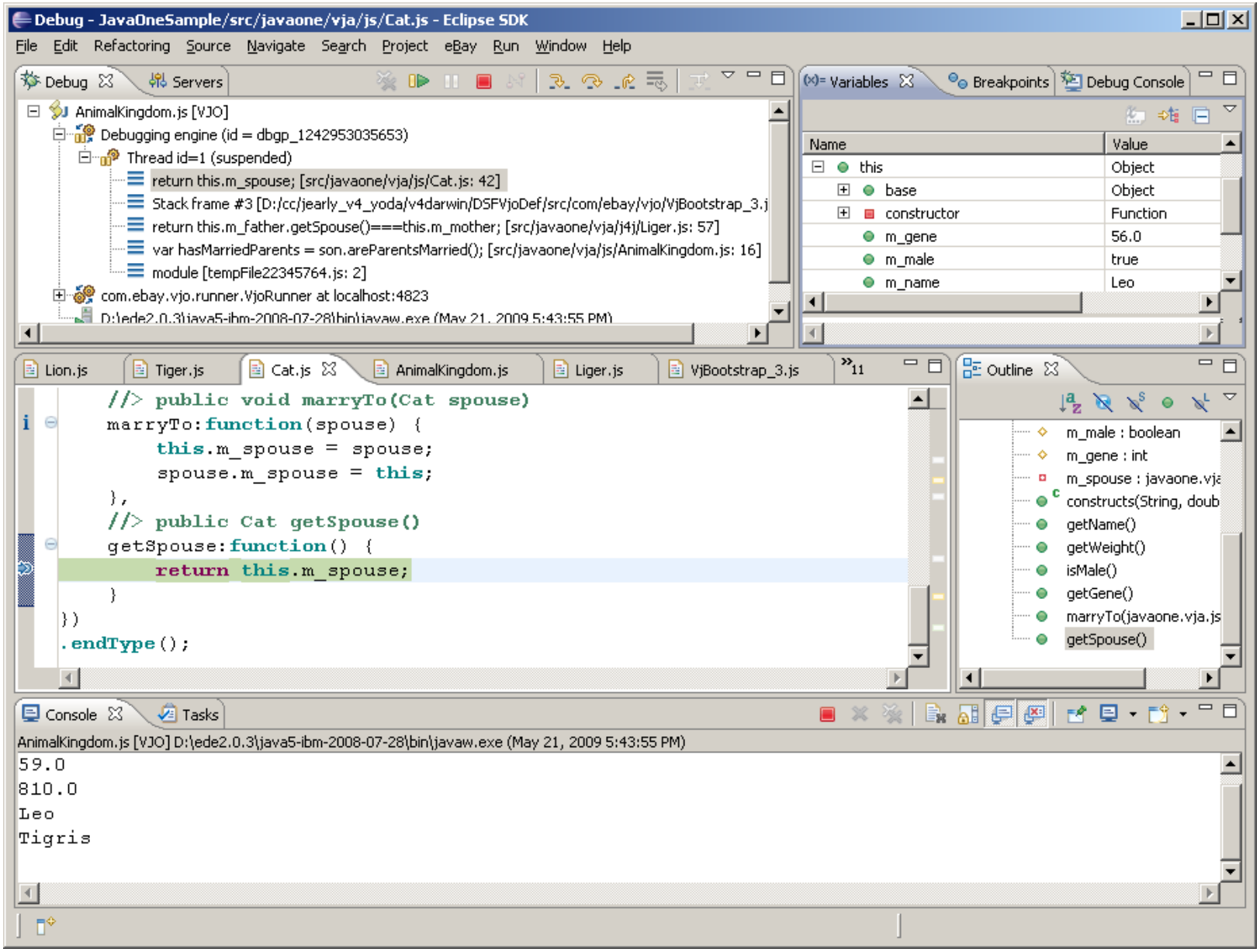

Java - JavaOneSample/src/javaone/vja/js/Cat.js - Eclipse SDK
File Edit Refactoring Source Navigate Search Project eBay Run ClearCase Window Help
IMate.js
Cat.js
AnimalKingdom.js
Liger.js
Outline
vjo.ctype('javaone.vja.js.Cat') //< public
.satisfies('javaone.vja.js.IMate')
.props({
SPECIES:"Cat", //<final String
maxage:100 //<public Number
})
.protos({
m_name:null, //< protected String
m_weight:0, //<protected double
m_male:true, //<protected boolean
m_gene:7, //<protected int
m_spouse:null, //<private Cat
//> public constructs(String name, double wei
constructs:function(name, weight, male){
this.m_name=name;
this.m_weight=weight;
this.m_male=male;
},
//> public String getName()
getName:function(){
return this.m_name;
},
//> public double getWeight()
getWeight:function(){
return this.m_weight;
},
//> public boolean isMale()
isMale:function(){
return this.m_male;
},
//> public int getGene()
getGene:function(){
import declarations
javaone.vja.js.IMate
javaone.vja.js
Cat
SPECIES : String
maxage : Number
m_name : String
m_weight : double
m_male : boolean
m_gene : int
m_spouse : javaone.vja.js.Cat
constructs(String, double, boolean)
getName()
getWeight()
isMale()
getGene()
marryTo(javaone.vja.js.Cat)
getSpouse()
javaone.vja.js.Cat.getWeight()


Debug - JavaOneSample/src/javaone/vja/js/AnimalKingdom.js - Eclipse SDK
File Edit Refactoring Source Navigate Search Project eBay Run Window Help
Lion.js Tiger.js IMate.js Cat.js *AnimalKingdom.js Liger.js VjBootstrap_3.js
vjo.ctype('javaone.vja.js.AnimalKingdom') //< public
.needs(['javaone.vja.js.Lion','javaone.vja.js.Tiger','javaone.vja.j4j.Liger'])
.props({
main:function() {
var lion = new this.vj$.Lion('Leo', 450, true);//<Lion
lion.
equals(Object o) void - Object
getClass() void - Object
getGene() void - Cat
getName() void - Cat
getSpouse() void - Cat
getWeight() void - Cat
hashCode() void - Object
isMale() void - Cat
m_gene int - Cat
m_male boolean - Cat
Press 'Ctrl+Space' to show Template Proposals
var hasMarriedParents = son.areParentsMarried();
vjo.sysout.println('marriedParents: ' + hasMarriedParents);
lion.marryTo(tigress);
vjo.sysout.println('marriedParents: ' + son.areParentsMarried());
}
})
.endType();
Writable
Smart Insert
6 : 14


Java - JavaOneSample/src/javaone/vja/js/Cat.js - Eclipse SDK
File Edit Refactoring Source Navigate Search Project eBay Run ClearCase Window Help
IHunter.js Tiger.js Lion.js Tiger.js IMate.js Cat.js AnimalKingdom.js Liger.js
},
//> public double getWeight()
getWeight:function(){
return this.m_weight;
},
//> public boolean isMale()
isMale:function(){
return this.m_male;
},
//> public int getGene()
getGene:function(){
return this.m_gene;
},
//> public void marryTo(Cat spouse)
marryTo:function(spouse) {
this.m_spouse = spouse;
spouse.m_spouse = this;
},
//> public Cat getSpouse()
getSpouse:function() {
return this.m_spouse;
}
})
.endType();
Call Hierarchy
Members calling 'getGene()' - in workspace
getGene() - javaone.vja.js.Cat
constructs(javaone.vja.js.Lion, javaone.vja.js.Tiger, String, boolean) - javaone.vja.j4j.Liger (2 matches)
main() - javaone.vja.js.AnimalKingdom
Line
Call
javaone.vja.js.Cat.getGene()


Java - JavaOneSample/src/javaone/vja/js/Tiger.js - Eclipse SDK
File Edit Refactoring Source Navigate Search Project eBay Run ClearCase Window Help
Lion.js Tiger.js IMate.js Cat.js Liger.js
vjo.ctype('javaone.vja.js.Tiger') //< public
.inherits('javaone.vja.js.Cat')
.protos({
//> public constructs(String name, double wei
constructs:function(name, weight, male){
this.base(name, weight, male);
this.m_gene=43;
}
})
.endType();
Outline Hierarchy
Cat, working set: Window Working Set
Object
Cat
Lion
Tiger
Tiger
constructs(String, double, boolean)
javaone.vja.js.Tiger - JavaOneSample/src


VJO - JavaOneSample/src/javaone/vja/js/SimpleMain.js - Eclipse SDK
File Edit Refactoring Source Navigate Search Project eBay Run Window Help
Lion.js Tiger.js Cat.js AnimalKingdom.js Liger.js VjBootstrap_3.js SimpleMain.js
vjo.ctype('javaone.vja.js.SimpleMain') //< public
.needs('javaone.vja.js.Cat')
.props({
//> public void main(String... arguments)
main: function() {
var badcat = new this.vj$.Cat();
var goodcat = new this.vj$.Cat('fudley', 40, true);
}
})
.endType();
Problems
1 error, 0 warnings, 0 infos (Filter matched 1 of 18650 items)
Description
Errors (1 item)
"the number of method arguments: 0 should match the number of destination method arguments: 3 in the expression: this.vj$.Cat()"
Writable
Smart Insert
7 : 60


Debug - JavaOneSample/src/javaone/vja/js/Cat.js - Eclipse SDK
File Edit Refactoring Source Navigate Search Project eBay Run Window Help
Debug
Servers
AnimalKingdom.js [VJO]
Debugging engine (id = dbgp_1242953035653)
Thread id=1 (suspended)
return this.m_spouse; [src/javaone/vja/js/Cat.js: 42]
Stack frame #3 [D:/cc/jearly_v4_yoda/v4darwin/DSFVjoDef/src/com/ebay/vjo/VjBootstrap_3.j
return this.m_father.getSpouse()===this.m_mother; [src/javaone/vja/j4j/Liger.js: 57]
var hasMarriedParents = son.areParentsMarried(); [src/javaone/vja/js/AnimalKingdom.js: 16]
module [tempFile22345764.js: 2]
com.ebay.vjo.runner.VjoRunner at localhost:4823
Variables
Breakpoints
Debug Console
Name
Value
this
Object
base
Object
constructor
Function
m_gene
56.0
m_male
true
m_name
Leo
Lion.js
Tiger.js
Cat.js
AnimalKingdom.js
Liger.js
VjBootstrap_3.js
11
//> public void marryTo(Cat spouse)
marryTo:function(spouse) {
this.m_spouse = spouse;
spouse.m_spouse = this;
},
//> public Cat getSpouse()
getSpouse:function() {
return this.m_spouse;
}
})
.endType();
Outline
m_male : boolean
m_gene : int
m_spouse : javaone.vja
constructs(String, doub
getName()
getWeight()
isMale()
getGene()
marryTo(javaone.vja.js
getSpouse()
Console
Tasks
AnimalKingdom.js [VJO] D:\ede2.0.3\java5-ibm-2008-07-28\bin\javaw.exe (May 21, 2009 5:43:55 PM)
59.0
810.0
Leo
Tigris

Runtime Type Loading Service

- Analogous to Java ClassLoader
- No global type tempering (such as modification of Object and Function prototype)
- Supports dynamic type loading
- Manages type resolution and initialization order based on type dependencies (including circular dependency)

VJET Supports Common Java Libraries

vjo.java.lang.* & vjo.java.util.*

```
List<String> li = new ArrayList<String>();
```
(Java)

```
var li = new this.vj$.ArrayList(); //< List<String>
```
(JS)

Why not write VJET code completely in Java?

- Write JavaScript code using Java types
- Enable Java developers to author JavaScript
  - Use a familiar programming language – Java
  - Use familiar IDE features
  - No mental switch on keywords and constructs
  - No confusion about JavaScript scoping

```
x = 1;
var x =1;
this.x = 1;                    (JS)
```

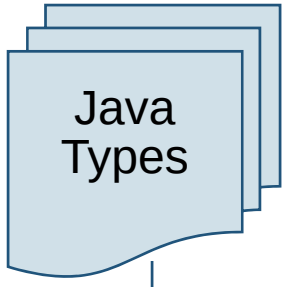
Leverage Java *and* JavaScript in Java

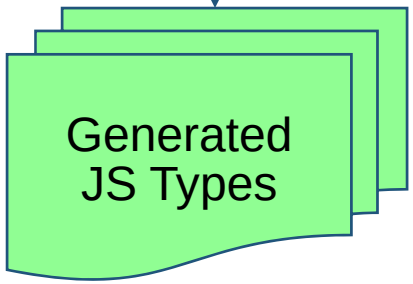
- Enable Java-to-JavaScript language translation
- Take the best of JavaScript and add it to Java
  - Closures
  - Function
  - Object literal
- Make native JavaScript types available in Java
  - Global JavaScript functions and types
  - Browser and DOM objects
  - VJET and third-party JavaScript libraries

Translation + Interoperation: Java to JS

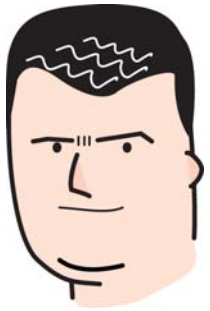
Java
Developers

Write





Used by



JavaScript
Developers

VJET Java to JS Translation Service

- Java → JavaScript - a full language translation
- Supports most Java language features and object modeling semantics
- Syntactical and structural resemblance
- Semantic equivalence
- Provides extension for custom translation

VJET Java to JS Translation

```java
public static void itration() {
  List<String> li = new ArrayList<String>();
  Iterator<String> itr = li.iterator();
  while (itr.hasNext()) {
     System.out.println(itr.next());
  }
  int[] arr = new int[]{1, 2, 3};
  for (int i = 0; i < arr.length; i++) {
    System.out.println(arr[i]);
  }
  Array jsArr = new Array("abc", 123, 14.5);
  jsArr.push("xyz");
  for (Object val : jsArr) {
    System.out.println(val);
  }
}
```

(Java)

→

```javascript
//> public void itration()
itration:function(){
  var li=new this.vj$.ArrayList();
  var itr=li.iterator();
  while(itr.hasNext()){
    this.vj$.System.out.println(itr.next());
  }
  var arr=[1,2,3];
  for (var i=0;i<arr.length;i++){
    this.vj$.System.out.println(arr[i]);
  }
  var jsArr=["abc",123,14.5];
  jsArr.push("xyz");
  for (var val in jsArr){
    this.vj$.System.out.println(val);
  }
}
```

(JS)

Author JavaScript using Java

```
public static boolean myOnclick(MouseEvent e) {
  HtmlButton b = (HtmlButton)e.getTarget();
  b.blur();
  b.attachEvent("click", MyJsType.update);
  HtmlTextArea msgArea =
      docx.getTextArea("messageArea");
  String value = msgArea.getValue();
  win().alert(js.escape(value));
  HtmlSpan span = docx.span();
  span.setClassName("warn");
  span.setInnerHTML(value);
  msgArea.getParentNode().insertBefore(span, msgArea);
  return true;
}
```

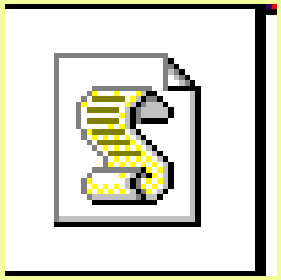
(Java)

VJET Java to JS Translation

```
//> public boolean myOnclick(MouseEvent e)
myOnclick:function(e){
    var b=e.target || window.event.srcElement;
    b.blur();
    b.attachEvent("click",this.vj$.MyJsType.update);
    var msgArea=document.getElementById("messageArea");
    var value=msgArea.value;
    window.alert(escape(value));
    var span=document.createElement('span');
    span.className="warn";
    span.innerHTML=value;
    msgArea.parentNode.insertBefore(span,msgArea);
    return true;
},
```

(JS)

The Best of Java™ and JavaScript™ Together at Last!



Object Literal

Functional programming

closures

Mixins

Prototypical inheritance

VJET

generics

interface

class

varargs

enum

Access control

Access Existing JavaScript from Java: JS to Java API Translation Service

- Produces Java API (NativeJsProxy) from VJET types authored in JavaScript
- Exposes functional programming features to Java API, where function and type reference become first level programming constructs

Translation + Interoperation: JS to Java API

JavaScript Developers

Write

JS Types

Can Delegate to

Legacy Js code

Java Native Proxy API

Used by

Java Developers

JS to Java API

```
vjo.ctype("javaone.js.MyJsType")
.protos({
   m_index: 0, //<int
   //>public void constructs(int)
   constructs: function(index) {
     this.m_index = index;
   },
   //>public void doit()
   doit: function() {
     alert(this.m_index);
   }
})
.props({
  s_counter: 0, //<int
  //>public void update(boolean)
  update: function(increament) {
    alert(increament? this.s_counter++ :
       this.s_counter--);
  }
})
.endType();
```

(JS)

```
public class MyJsType extends NativeJsProxy {

   public MyJsType(int index) {
      super(index);
   }
   public static void update(boolean increament) {
      callStatic(increament);
   }
   public void doit() {
      call();
   }
   public static final NativeJsTypeRef prototype =
      NativeJsTypeRef.get(MyJsType.class);
   public static final  INativeJsFuncProxy update =
      NativeJsFuncProxy.create(prototype, "update");
   public final INativeJsFuncProxy doit =
      NativeJsFuncProxy.create(this, "doit");
}
```

(Java)

VJET Java Core Library for JavaScript

- Java API for
  - JS global functions – VJ.js.*
  - JS native types:
    - Array, Object Literal, Function
  - Browser, DOM and Events (w3c – based)
  - Convenient DOM operation – VJ.docx.*
  - Functional Programming Extension – VJ.jsx.*
    - Function reference
    - Type reference
    - Hitch (function scope closure)
    - Curry (function arguments closure)

Access JS Native Types from Java

```
///import com.ebay.dsf.dap.proxy.Array;// Proxy to JavaScript native array
///…
Array jsArr = new Array("abc", 123, 14.5);
for (Object val : jsArr) {
    System.out.println(val);
}
```

(Java)


```
var jsArr=["abc",123,14.5];
for (var val in jsArr){
    this.vj$.System.out.println(val);
}
```

(JS)

Access Native JS Functions from Java

```
String rtn = (String)js.eval(js.escape(txt));
js.encodeURI(rtn);

win().setTimeout(js, 100);
int id = win().setInterval(AJsType.update, 10);
win().clearInterval(id);
```

(Java)


```
var rtn=eval(escape(txt));
encodeURI(rtn);

window.setTimeout(js,100);
var id=window.setInterval(this.vj$.AJsType.update,10);
window.clearInterval(id);
```

(JS)

Access Browser DOM Functions from Java

```
HtmlButton b = (HtmlButton)e.getTarget();
b.blur();
HtmlSpan span = docx.span();
span.setClassName("warn");
span.setInnerHTML("<b>Hi</b>");
Node msgArea = doc().getElementById("messageArea");
msgArea.getParentNode().insertBefore(span, msgArea);
```

(Java)


```
var b=e.target || window.event.srcElement;
b.blur();
var span=document.createElement('span');
span.className="warn";
span.innerHTML="<b>Hi</b>";
var msgArea=document.getElementById("messageArea");
msgArea.parentNode.insertBefore(span,msgArea);
```

(JS)

Java API for Functional Programming

```
MyVjoType2 a = new MyVjoType2(); // native proxy
a.doIt2("test");
a.doIt2.call(a, "test");
a.doIt2.apply(new MyVjoType2(), new Object[]{"test"} );

MyVjoType2.doIt("test" );
MyVjoType2.doIt.call(MyVjoType2.prototype, "test");
```

(Java)

```
var a=new this.vj$.MyVjoType2();
a.doIt2();
a.doIt2.call(a);
a.doIt2.apply(new this.vj$.MyVjoType2(),{'test'});

this.vj$.MyVjoType2.doIt("test");
this.vj$.MyVjotype2.doIt.call(this.vj$.MyVjoType2,false);
```

(JS)

JavaScript Library Integration

- External JavaScript Libraries are easily integrated
- VJET typing annotation provides non-intrusive way to integrate non-VJET JavaScript libraries
- No modifications needed for non-VJET JavaScript
- Typed API becomes available for both JavaScript and Java

Integration of Third Party JS Library - JQuery

- Create VJET API definition for existing JS library - jQuery

```
vjo.ctype("vjo.dsf.jqueryx.Jq")
.needs('vjo.dsf.jqueryx.Jqueryx')
.protos({
    //> public Element get(int index)
    //> public Array get()
    get:function() {},
    //> public Array queue()
    //> public Array queue(String name)
    //> public Jq queue(Function callback)
    //> public Jq queue(String name, Function callback)
    //> public Jq queue(Array queue)
    //> public Jq queue(String name, Array queue)
    queue:function(){},….
```

(JS)

Sample jQuery Usage in both Java and JavaScript

```
Jq.$(".slide").slideToggle("slow");
```

(Java)

```
$(".slide").slideToggle("slow");
```

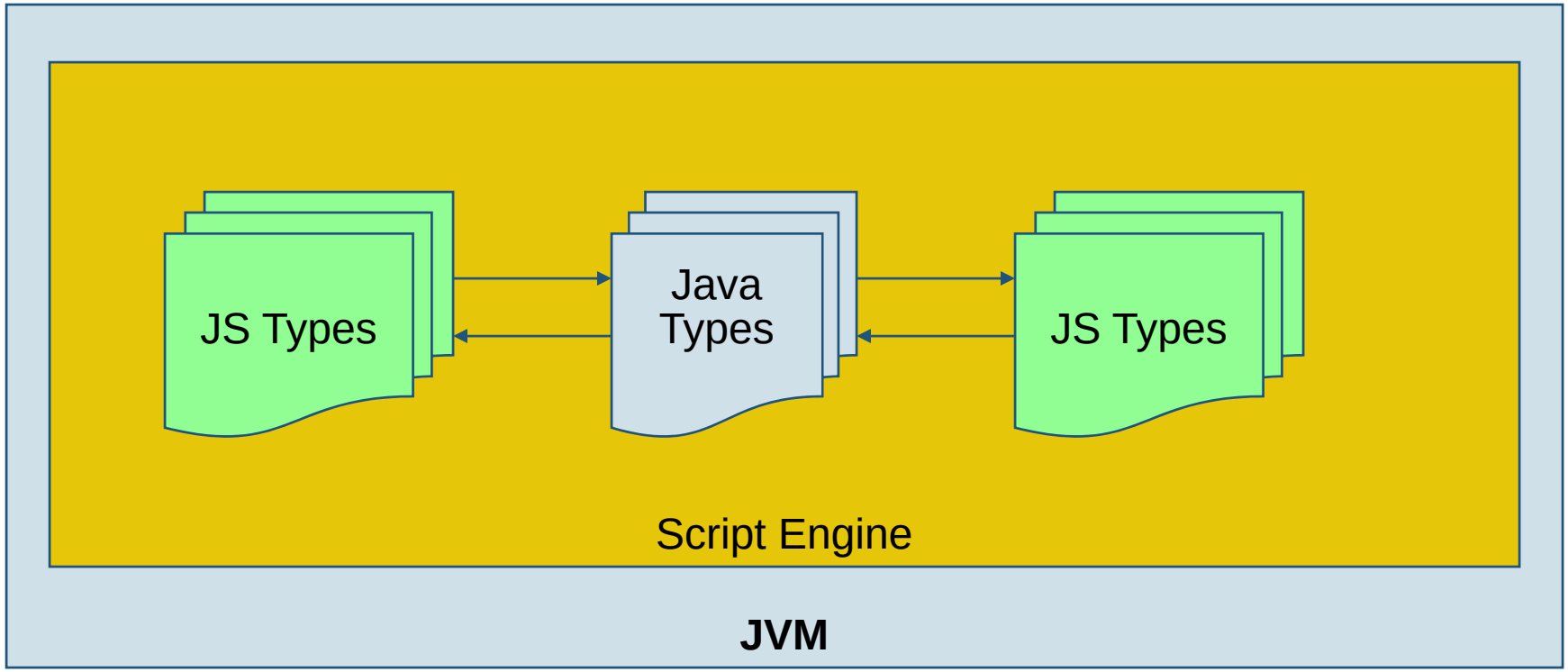
(JS)

VJET Active Programming Model

- VJET enhanced scripting service supports interoperability between Java and JavaScript, and blurs the boundary between the Java and JavaScript languages
- In Active Programming mode, Java types are loaded and executed in their authored forms (Java or JavaScript), while naturally interacting with other JavaScript types and objects
- Under Active Programming mode, native JavaScript types can live in the Java space via proxies, and Java objects can live in the script space, with object identity preserved across language boundaries

What you write is what you run and debug!!

Java + JS Runtime Interoperation



Running JavaScript in Active Mode

Order of Execution

1 → **var** lion = **new this**.vj$.Lion('Leo', 450, true); → calling to **Lion.js**

2 → **var** tigress = **new this**.vj$.Tiger('Tigris', 350, false); → calling to **Tiger.js**

3 → **var** son = **new this**.vj$.Liger(lion, tigress, 'Hercules', true); → calling to **Liger.java**

```
        public Liger(Lion father, Tiger mother, String name, boolean isMale) {
                m_father = father;
                m_mother = mother;
4 →             m_gene = father.getGene()|mother.getGene();  → calling to Cat.js
        }

5 →             getGene:function(){return this.m_gene;}
```

6 → **var** hasMarriedParents = son.areParentsMarried(); → calling to **Liger.java**

```
        public boolean areParentsMarried() {
7 →                     return m_father.getSpouse() == m_mother;  → calling to Cat.js
        }
```

End-to-End Web Application Development

- VJET's Browser Emulation and Communication Services enables live execution, debugging, and updates of a client-side JavaScript application from a single IDE
- Render applications in a modern web browser while executing and debugging JavaScript in VJET's Virtual Browser
- Enables a productive development process

VJET Virtual Browser

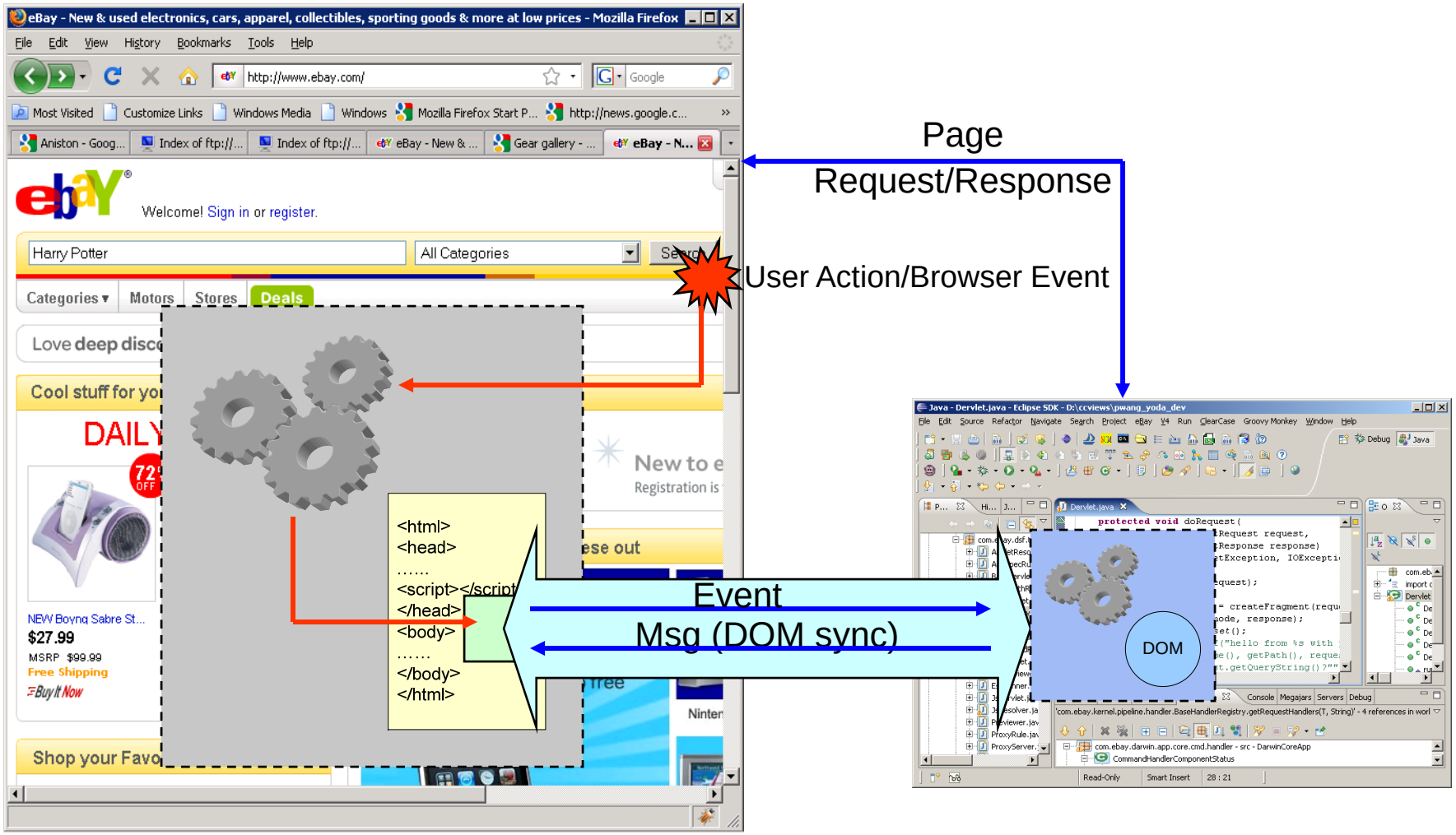
- Provides browser based scripting environment
- Enables vendor specific browser behavior
- Supports “single-threaded” JS execution model with proper timer events

VJET Duplex Live Connect (DLC)

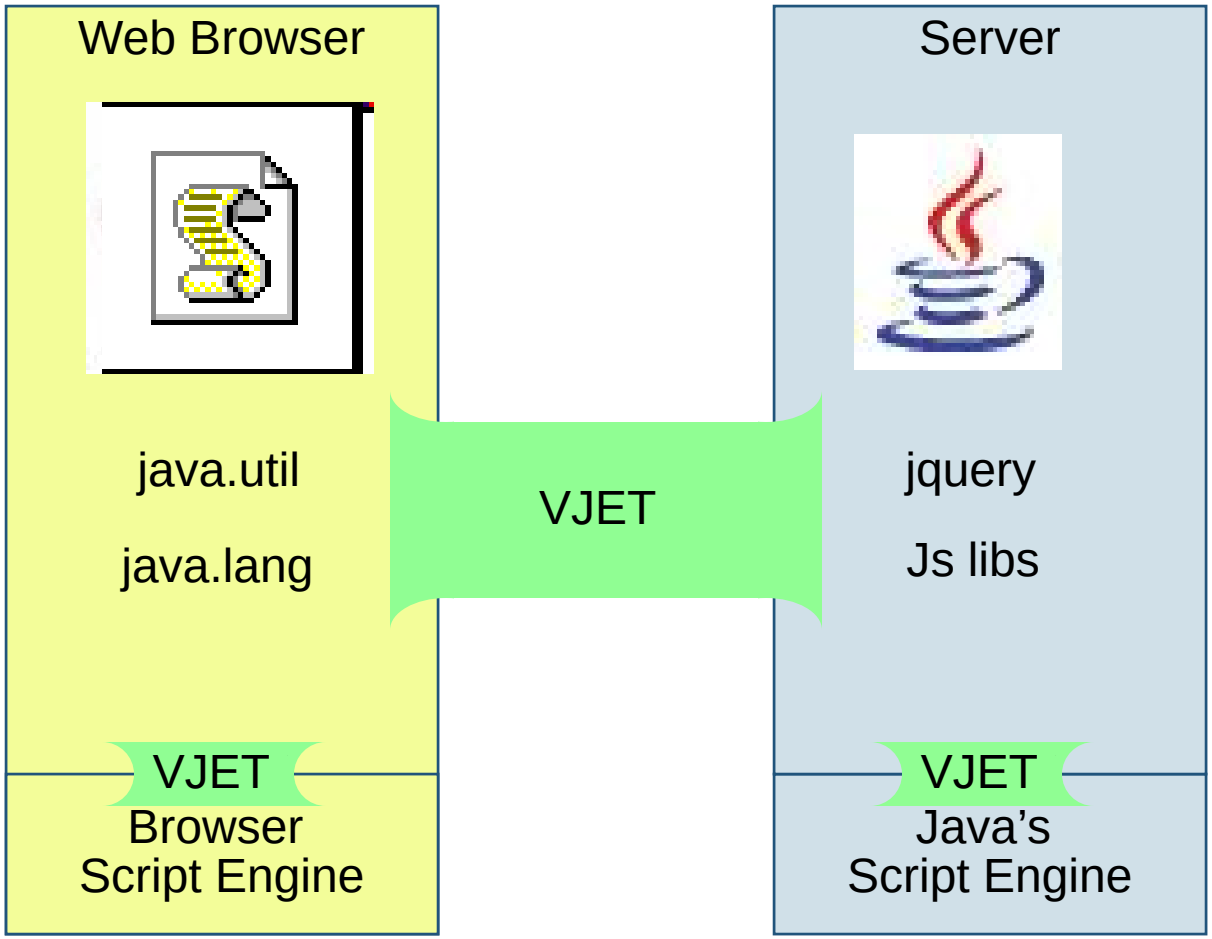
Virtual to External Browser Connection

- Combining the VJET Virtual Browser with VJET DLC enables:
  - Native browser binding for graphical rendering and user interaction
  - Auto-DOM-sync between virtual and native browsers
  - Capture and replay for interactive browser session
  - Auto-generation of JUnit tests with window events, user interactions and AJAX operations

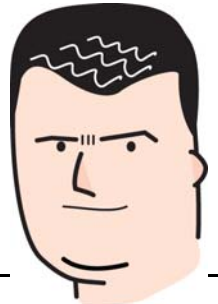
Run JavaScript in VJET Virtual Browser



The Best of Java™ Libraries and JavaScript™ Libraries



Recap



Java developers	JS developers
✓ Larger pool of talent	Difficult to find talent ✓
✓ Established patterns	Increasing need for client side interactive behavior ✓
✓ Sophisticated tools ++	Managing complexity is difficult ✓
✓ Standardized technologies	Browser incompatibilities hamper quality ✓
✓ Large collection of commonly used libraries	Multiple library use is problematic ✓

Benefits

- JavaScript developers can enjoy Java-like modeling and tooling
- Break down language barriers and enable Java developers to develop interactive web applications
- Write, Run, Debug, all in one IDE
- Leverage existing Java and JavaScript skill sets between both programming environments
- Enable large scale, enterprise level JavaScript application development

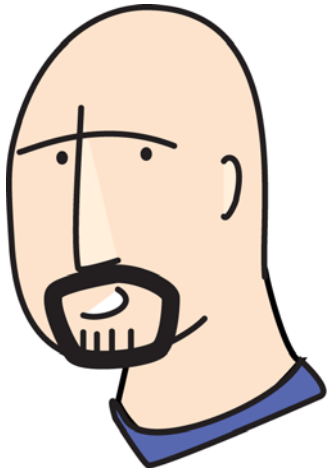
Today @ eBay

- Java developers also work on JavaScript
- JavaScript developers also work on Java
- Tooling has blurred the line between
  - Java and JS Languages
  - Browsers
  - JavaScript and Java Libraries

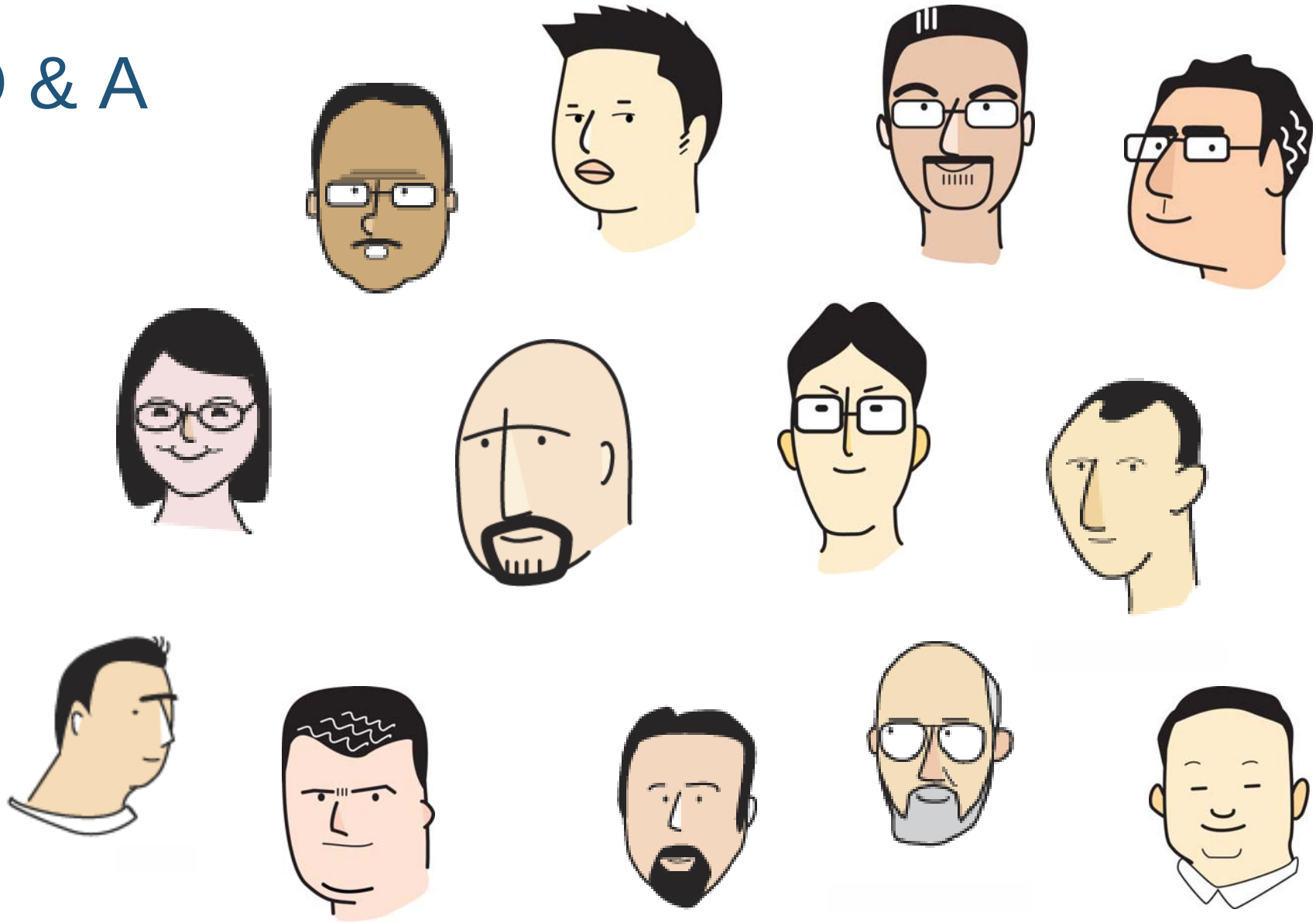
The Gap is Gone!

One More Thing…

- Announcement from eBay Leaders
  - Mark Palaima - Chief Architect & eBay Fellow
  - James Barrese - VP, Systems Dev (Architecture)



Q & A



Thank You

Justin Early
jearly@ebay.com
Yitao Yao
yyao@ebay.com

Booth #733