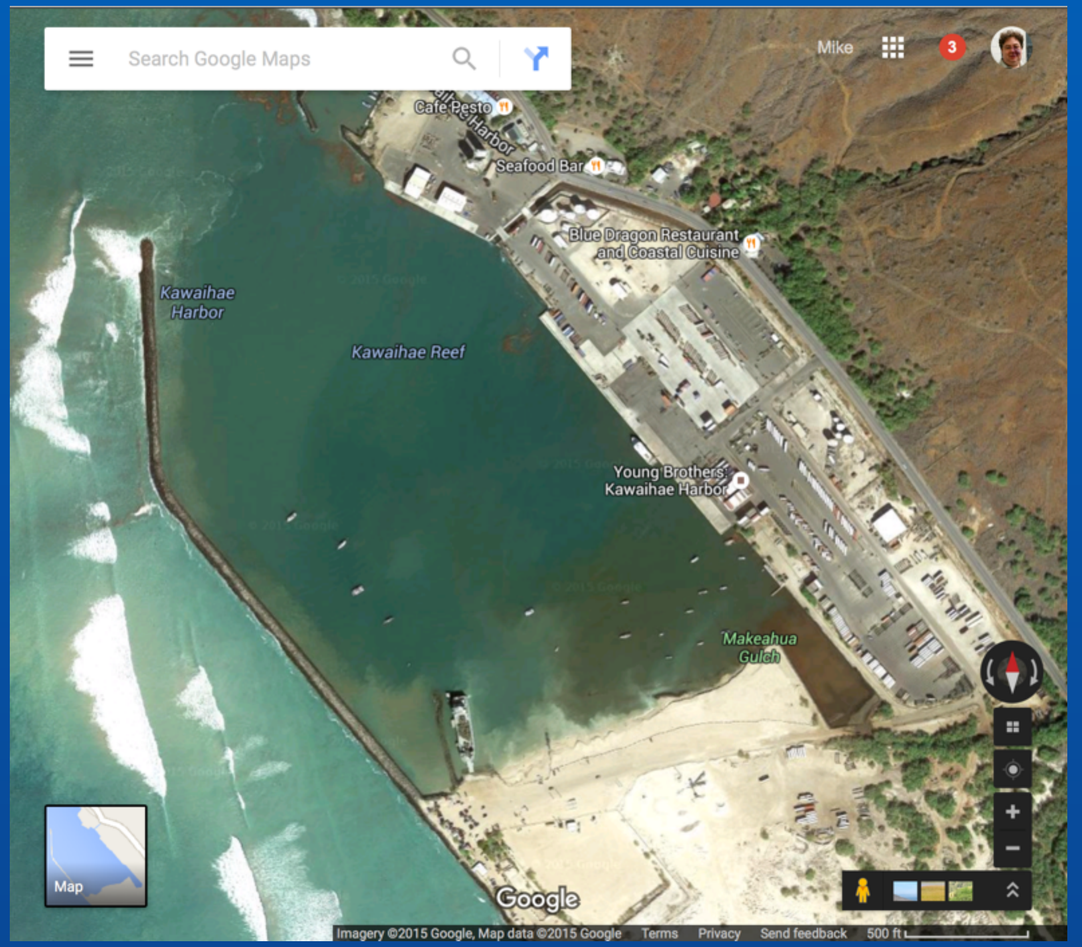
James Writes Java

What I Have Learned by Reading James Gosling's Code

Mike Duigou
@mjduigou

Safe Harbour Statement

This is a safe harbour →



What This Is About

- I have had good fortune to work with many excellent people
- I have worked on several projects begun by James Gosling
- James uses Java in interesting ways

What This Is Not About

- These are observations not necessarily insights
- This is not advice though it might be explanation
- Because James does it doesn't mean everyone should
- Look at the horrible crappy code James wrote! Ha! Ha!

Java Class Libraries

- Story of Java is well known (c. 1991)
- Java developer since 1998, contributor since 2009 in deployment & core libraries

Stanford/Audi "Shelley" Autonomous Vehicle

- James wrote prototype MATLAB/Simulink to Java cross compiler (c. 2007-8)
- I worked on CANBUS, logging, telemetry, safety system and visualization tools for Shelley Project

Stanford/Audi "Shelley" Autonomous Vehicle

- James wrote prototype MATLAB/Simulink to Java cross compiler (c. 2007-8)
- I worked on CANBUS, logging, telemetry, safety system and visualization tools for Shelley Project

Sun Java Store

- James wrote prototype warehouse and catalogue (c. 2008)
- I worked on client back-end & caching, product package builder, try-before-buy, warehouse infrastructure, build & tools

Liquid Robotics SV3 Wave Glider

- James originated Java based Regulus Operating Environment (2011)
- I work on OS-like portions; core infrastructure, logging, service loading/unloading, plugins, some devices & communications, build & tooling

Basics

- James uses NetBeans IDE
- James Code is idiosyncratic Java--like nobody else
- James writes more Java code than you do
- Very few comments, very little javadoc
- James is an early adopter of new Java features

Lead with Code

```
public String listPlugins() {
    if(hasNoPlugins()) return "No plugins installed";
    StringBuilder sb = new StringBuilder();
    Collection<Plugin> thePlugins = allPlugins();
    Plugin[] pa = thePlugins.toArray(new Plugin[thePlugins.size()]);
    Arrays.sort(pa,byName);
    String prefix="Running: ";
    for(int part = 0; part<=1; part++) {
        for(Plugin p:pa)
            if(part==0==p.isDeployed()) {  // bizarre boolean is correct.
                sb.append(prefix);
                sb.append(p.getName());
                prefix=", ";
            }
        prefix = "  Not running: ";
    }
    return sb.toString();
}
```

Libraries

- James builds his own libraries
- Some stuff that should have been in Java Libraries
- Some stuff is obsolete--Java Library or 3rd party libs
- Includes some variations on Java libraries
- Lots of stuff that is niche or unique
- Some stuff that is probably only useful to James

Let's Recurse a Bit

```
private Object readPOJOarray(Reader in, int depth, Class targetClass, Map parent, Object name) throws IOException {
    Object v = readPOJO0(in, null, null);
    if(v==COMMA) return readPOJOarray(in,depth,targetClass, parent, name);
    if(v==EOF) {
        Object ret = objectFactory.createArray(targetClass, depth, parent, name);
        return ret;
    }
    Object ret = readPOJOarray(in,depth+1,commonSuperclass(targetClass,v==null?null:v.getClass()),
            parent, name);
    objectFactory.set(ret, depth, v);
    return ret;
}
```

Let's Recurse a Bit

```
private Object readPOJOarray(Reader in, int depth, Class<?> targetClass, M parent, String name) throws IOException {
    Object v;
    do {
        v = readPOJO0(in, null, null);
    } while(v==COMMA);
    if(v==EOF) {
        Object ret = objectFactory.createArray(targetClass, depth, parent, name);
        return ret;
    }
    Object ret;
    if(depth<256) {
        ret = readPOJOarray(in,depth+1,
            commonSuperclass(targetClass,v==null?null:v.getClass()),
            parent, name);
        objectFactory.set(ret, depth, v);
    } else {
        // Switch to a temporary data structure to avoid deep recursion.
        ArrayList<Object> tail = new ArrayList<>(depth);
        tail.add(v);
        do {
            v = readPOJO0(in, null, null);
            if(v == EOF) break;
            if(v == COMMA) continue;
            tail.add(v);
        } while(true);
        ret = objectFactory.createArray(targetClass, depth + tail.size(), parent, name);
        for(Object each : tail) objectFactory.set(ret, depth++, each);
    }
    return ret;
}
```

Perils of Using Personal Libraries

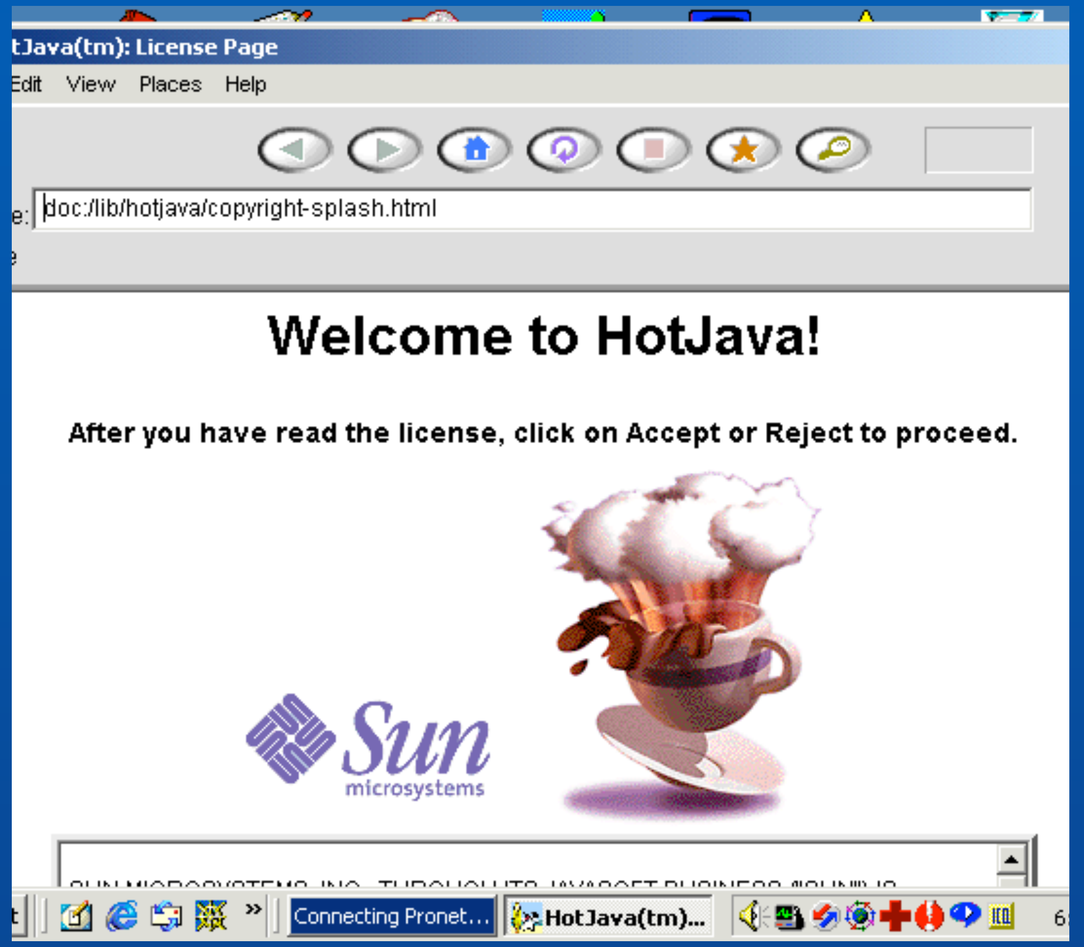
```
public static String getParameter(HttpServletRequest request, String key, String dflt) {
    String s = request.getParameter(key);
    if(s!=null) return s;
    if(isMultipart(request)) try {
        Part p = request.getPart(key);
        if(p==null) return null;
        try(InputStream in = p.getInputStream()) {
            return loadToString(in);
        }
    } catch(Throwable ex) {
    }
    return dflt;
}
```

Contributions to Java Libraries

- **`java.util.Date`**
  - Mutable style was intentional
- **`PipedInputStream/PipedOutputStream`**
- **`StreamTokenizer`**
  - Four **`StreamTokenizer`** parsers in Regulus
- Some portions of **`java.net`** package...

Remember HotJava Browser?

- **`java.net`** originally from HotJava
  - **`URL`**
  - **`URLConnection, …`**
  - **`ContentHandler, …`**
  - etc…
- Ran Applets for richer content than HTML could provide

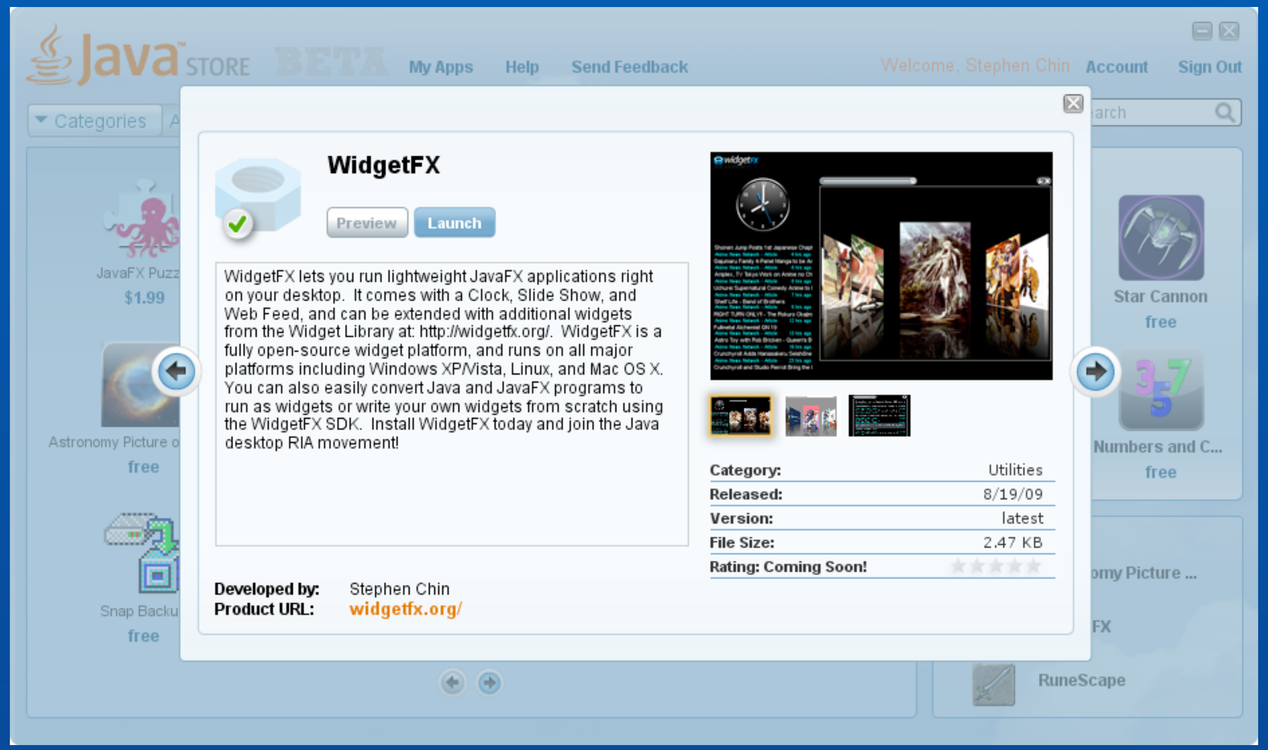


Web vs Rich Client

- James usually does both
- Web for static, low bandwidth
- Rich client for data intensive interactive

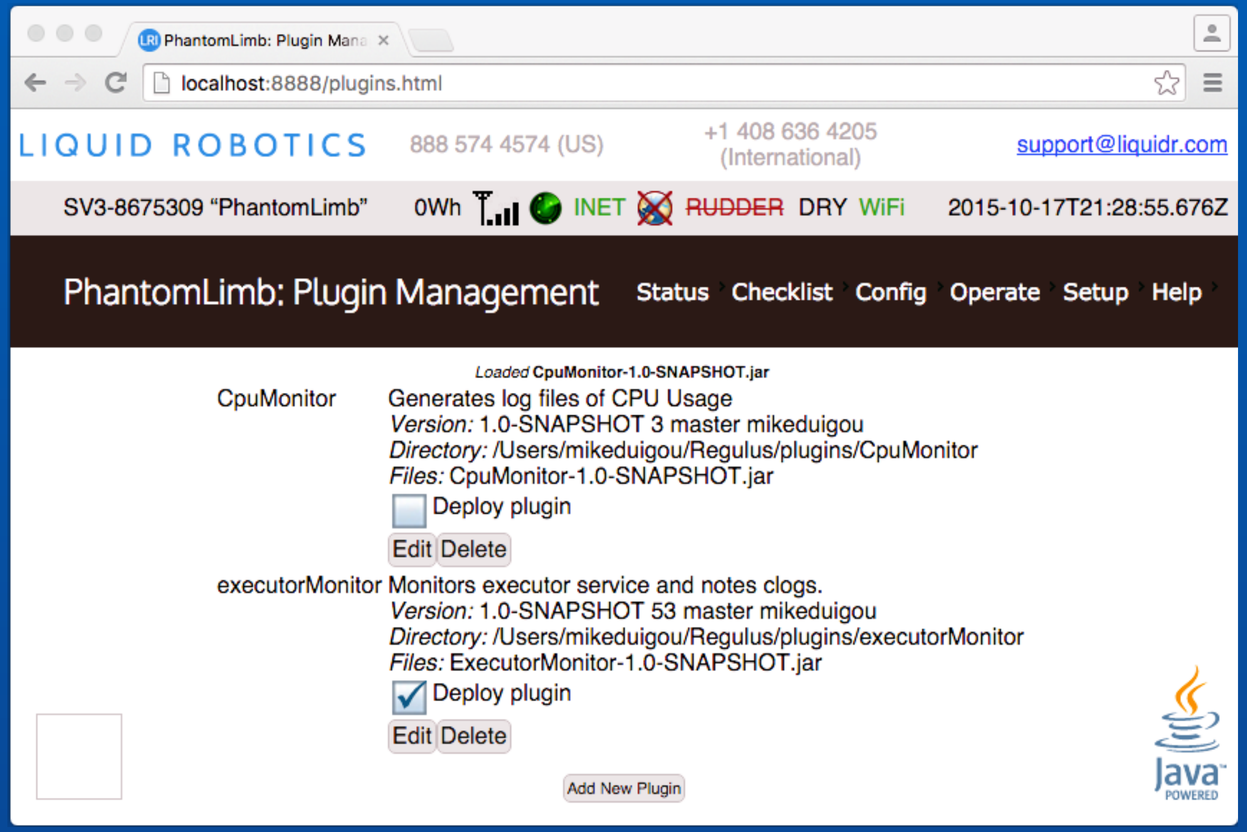
Sun Java Store

- Basic static catalogue HTML site
  - Search Engine indexed
- Rich JavaFX app for preview and purchasing
  - Functionality was mostly HTML5 achievable (but not in 2009)
  - Full L&F and user experience was richer than web.



Wave Glider Regulus

- Vehicle control software provides web interface for engineering, manufacturing, operations
- Your pages are being served over cell from a ship at sea by Jetty and servlets
- Nobody has started an international waters gambling site (yet)
- Pages are mostly static with some AJAX and timed reloads



Generating HTML the James Way

- **`/** Vaguely a replacement for JSP, but as an API instead of it's own language */`**
- Each page is a Servlet configured with annotations
- Similar system for Java Store and Regulus
  - Java Store was raw HTML, no JavaScript, no AJAX
  - Regulus has some JavaScript, uses JQuery and uses some AJAX

```
@WebServlet(name = "PluginManagementServlet", urlPatterns = {"/plugins.html"})
@MultipartConfig
@NavMenu("Config/Plugins")
public class PluginManagementServlet extends HttpServlet {
```

```
public final void page() {
    try {
        try {
            headStuff();
            if(headText!=null) for(String ht:headText) appendRaw(ht);
            if(cssPages!=null) for(String css:cssPages) useCSS(css);
            if(scriptPages!=null) for(String script:scriptPages) useScript(script);
            inlineCSS();
            inPart(Part.Body);
            wholeBody();
            endUntil("body");
            if(scriptTail!=null && !scriptTail.isEmpty()) {
                start("script");
                for(String st:scriptTail) appendRaw(st);
                end();
            }
        } catch (Throwable ioe) {
            try {
                endUntil("body");
```

Generating HTML the James Way

- **/**
  JSE
  it'
- Eac
  ann
- Sim
  - J
    J
  - R
    JQuery and uses some AJAX

```
public final void page() {
    try {
        try {
            headStuff();
            if(headText!=null) for(String ht:headText) appendRaw(ht);
            if(cssPages!=null) for(String css:cssPages) useCSS(css);
            if(scriptPages!=null) for(String script:scriptPages) useScript(script);
            inlineCSS();
            inPart(Part.Body);
            wholeBody();
            endUntil("body");
            if(scriptTail!=null && !scriptTail.isEmpty()) {
                start("script");
                for(String st:scriptTail) appendRaw(st);
                end();
            }
        } catch (Throwable ioe) {
            try {
                endUntil("body");
```

```
'/plugins.html"})
```

Reads like HTML, Codes like Java

- Pages can extend or override most page elements but usually focus on **<body>**
  - Overrides generally can't call super
- Methods for various HTML tags
  - Standard overloads for style, class and id
- Calling methods outputs HTML
- Close tags generated by **end()** and **end(tag)**

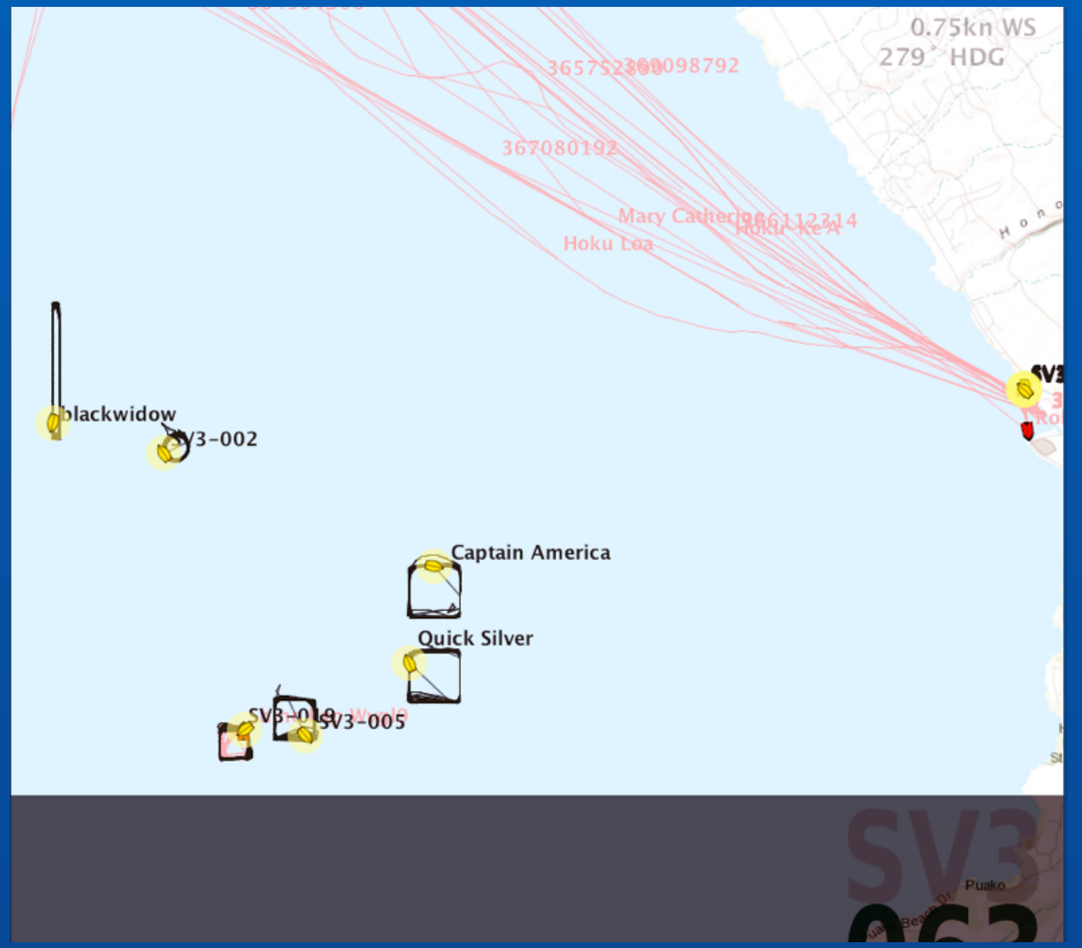
```
            table();
            for(Plugin p:sm.allPlugins()) {
                tr().td().append(p.getName());
                td();
                CFHashMap params = p.getParameters();
                if(params!=null) {
                    Object d = params.get("description");
                    if(d!=null) append(d.toString()).br();
                    i("Version: ").b(version).br();
                    i("Directory: ").append(p.baseDirectory().toString());
                    String[] l = p.baseDirectory().list();
                    if(l!=null) {
                        br().i("Files: ");
                        boolean first = true;
                        for(String s:l) {
                            if(first) first = false;
                            else append(", ");
                            append(s);
                        }
                    }
                    startDialog("Delete","Delete this plugin");
                    b("Are you sure you want to delete this plugin?");
                    button("Yes, nuke it!", "plugins.html?tag=Delete&pl="+p.getName());
                    end("div");
                }
            }
            end();
        }
        p();

        // add plugin

        startDialog("Add New Plugin","Upload a new plugin");
        formPost("plugins.html");
        hide("pl", "-NEW-");
        input("file","file",null,0);
        submit("tag", "Add");
        end("div");
    }
}.page();
```

Liquid Robotics
Rich Client Applications

- Cloud to Desktop
- Use REST and streaming JSON
- Swing WebStart based originally
  - Partly due to mapping components
- Now moving to JavaFX desktop applications
  - More vehicles, more data, more visualizations, time series



Anonymous Map Initializer

- James started using at least 15 years ago
- Anonymous inner class of Map Type with instance initializer to add entries
- New class for each instance
- Serialization nightmare
- James no longer uses this pattern

  → instance initializer in containing class
- There's something worse....

```
final Map<String,Integer> map = new HashMap() { {
        put("circle", 1);
        put("triangle", 3);
        put("square", 4);
} };
```

Anonymous Map Initializer

- James started using at least 15 years ago
- Anonymous inner class of Map Type with instance initializer to add entries
- New class for each instance
- Serialization nightmare
- James no longer uses this pattern

  → instance initializer in containing class
- There's something worse....

```
final Map<String,Integer> map = new HashMap() { {
        put("circle", 1);
        put("triangle", 3);
        put("square", 4);
} };
```

```
final Map<String,Integer> lambdaMap = makeMap(
        circle -> 1,
        triangle -> 3,
        square -> 4
);

<K,V> Map<K,V> makeMap(Function<K,V>... entries) {
```

Database? Just use the filesystem

- Java Store backed by mostly static relational database
- Assumed that database could fit in memory
- Records were POJOs persisted with JPA Hibernate as text files
  - Schema updater tools: **`bash, sed, awk,`** custom Java programs
- Brian Goetz implemented "eventually consistent" coherency to go beyond single node
- Financials and customer data purchase data held elsewhere

Database? Just use the filesystem

- Wave Glider uses JSON persisted to directories and files for configuration and persistent state
- Files usually start as hand authored
- Structure is almost completely static. Records are mostly static
- Records are Map of named primitives, arrays and sub-maps
  - No object binding, lots of hand tooled config parsers
- Loaded in to memory at boot and delayed write behind for flushing changes

printf

- James hates **printf** and **String.format**
- **System.out.println("String:" + *expr* + ...);**
- Formatting done with util methods taking an **Appendable**
  - **StringBuilder** passed down often through multiple objects

```
private static void append(long v, int w, boolean neg, char pad, Appendable a) throws IOException {
    if(v<0) { neg = true; --w; v = -v; }
    if(v>=10) append(v/10,--w,neg,pad,a);
    else {
        while(--w>0) a.append(pad);
        if(neg) a.append('-');
    }
    a.append((char)('0'+v%10));
}
```

```
@Override public void appendTo(StringBuilder sb) {
    try {
        appendTo((Appendable) sb);
    } catch(IOException never) {}
}
public void appendTo(Appendable a) throws IOException {
```

It's a Number

- James Code uses **`java.lang.Number`** frequently as an interchange type
- Including classes that implement **`Number`**
- Coercers and parsers produce **`Number`**
- A **`Number`** is a number, but nothing else is
  - No use of non-numbers implementing **`Number`**
  - Conversions from **`String`** happen only when "in motion"--parsing, assignment with coercion

It's a Number ~~Too~~ 2

- Shelley project MATLAB/Simulink cross-compiler used **`Number`** rather than **`Double`**
  - MATLAB wants to treat **everything** as an array of doubles
  - Using exact replica type, where known, is more efficient
  - Code generated was seen only by **`javac`** so lack of operator overload or ugly manual boxing/unboxing bothered nobody.
- Liquid Robotics Regulus also uses **`Number`**
  - JSON Parser instantiates most compact type for numbers but appears as **`Number`**
  - **`ObservableObject`** substitutes **`Number`** whenever a more specific numeric type is requested
    - Original requester still gets what they want but more flexible conversions

try/finally

- Attempt to use **try** **{** instead of **{** for all conditional and loop blocks
- **{** still needed to start a method, for **synchronized**, **finally**, and **catch** blocks
- **try** **{** requires **catch** and/or **finally** at end of block

try/finally

- Attempt to use **try** **{** instead of **{** for all conditional and loop blocks
- **{** still needed to start a method, for **synchronized**, **finally**, and **catch** blocks
- **try** **{** requires **catch** and/or **finally** at end of block

```
for (String r : requires.get()) try {
    unresolvedPowerDependencies.add(requireValidTag(r));
} catch(Exception badTag) {
    warn("ignoring and removing bad tag in requires: '" + r + "'");
    requires(false, r);
}
```

try/finally

- Attempt to use **try** **{** instead of **{** for all conditional and loop blocks
- **{** still needed to start a method, for **synchronized**, **finally**, and **catch** blocks
- **try** **{** requires **catch** and/or **finally** at end of block

```
for (String r : requires.get()) try {
    unresolvedPowerDependencies.add(requireValidTag(r));
} catch(Exception badTag) {
    warn("ignoring and removing bad tag in requires: '" + r + "'");
    requires(false, r);
}
```

```
if(src != CFHashMap.generateName) try {
    FileUtilities.delete(src);
} catch(IOException deleteFailed) {
    severe("Failed deleting (" + deleteFailed + "):" + src);
}
```

try/finally

- Attempt to use **`try {`** instead of **`{`** for all conditional and loop blocks
- **`{`** still needed to start a method, for **`synchronized`**, **`finally`**, and **`catch`** blocks
- **`try {`** requires **`catch`** and/or **`finally`** at end of block

```
for (String r : requires.get()) try {
    unresolvedPowerDependencies.add(requireValidTag(r));
} catch(Exception badTag) {
    warn("ignoring and removing bad tag in requires: '" + r + "'");
    requires(false, r);
}
```

```
if(src != CFHashMap.generateName) try {
    FileUtilities.delete(src);
} catch(IOException deleteFailed) {
    severe("Failed deleting (" + deleteFailed + "):" + src);
}
```

```
boolean done = false;
do try {
    // ...
} catch(Exception failed) {
    warn(failed);
} finally {
    // ...
} while(!done);
```

Exception Handling

- Checked exceptions aren't usually a problem in "James Code"
- **`throws`** is used **very** sparingly
- Most exceptions, including **`RuntimeException`**s, never escape very far
  - Lots of **`catch Throwable`** and **`catch Exception`**
- Log it, clean up and keep running (~~abort~~, retry, fail)
- Methods that don't throw might still have failed
  - Reflected in return or in state
- Failure is a system state not just something that happens
- Lots of objects implement state machines internally

Failure Happens

- Just. Keep. Running.
- Keep running even in the presence of errors and, inevitably, bugs
  - Defer error production until unavoidable (see **`java.net.URL`**)
  - Busted window in HotJava browser or dead Service in Liquid Robotics Regulus shouldn't take down application
- Try to treat failure cases as symmetrically as possible with success
  - Have-Not-Connected is the same as Cannot-Connect is the same as Connection-Failed
- Clean as you go, don't leave stale state around

Closing Thoughts

- Keep learning
  - Programmer for 36 years, 29 professional, 17 years with Java
- Articles and blogs about coding can only teach some lessons
  - … the same applies to conference sessions
- Reading other people's code will provide unique lessons
  - Learning is a personal experience

Thank You