How to write Java web apps like a JS hipster

Stephen Shary

github.com/STeveShary

stevesmashcode.com

linkedin.com/in/stephenshary

Myself

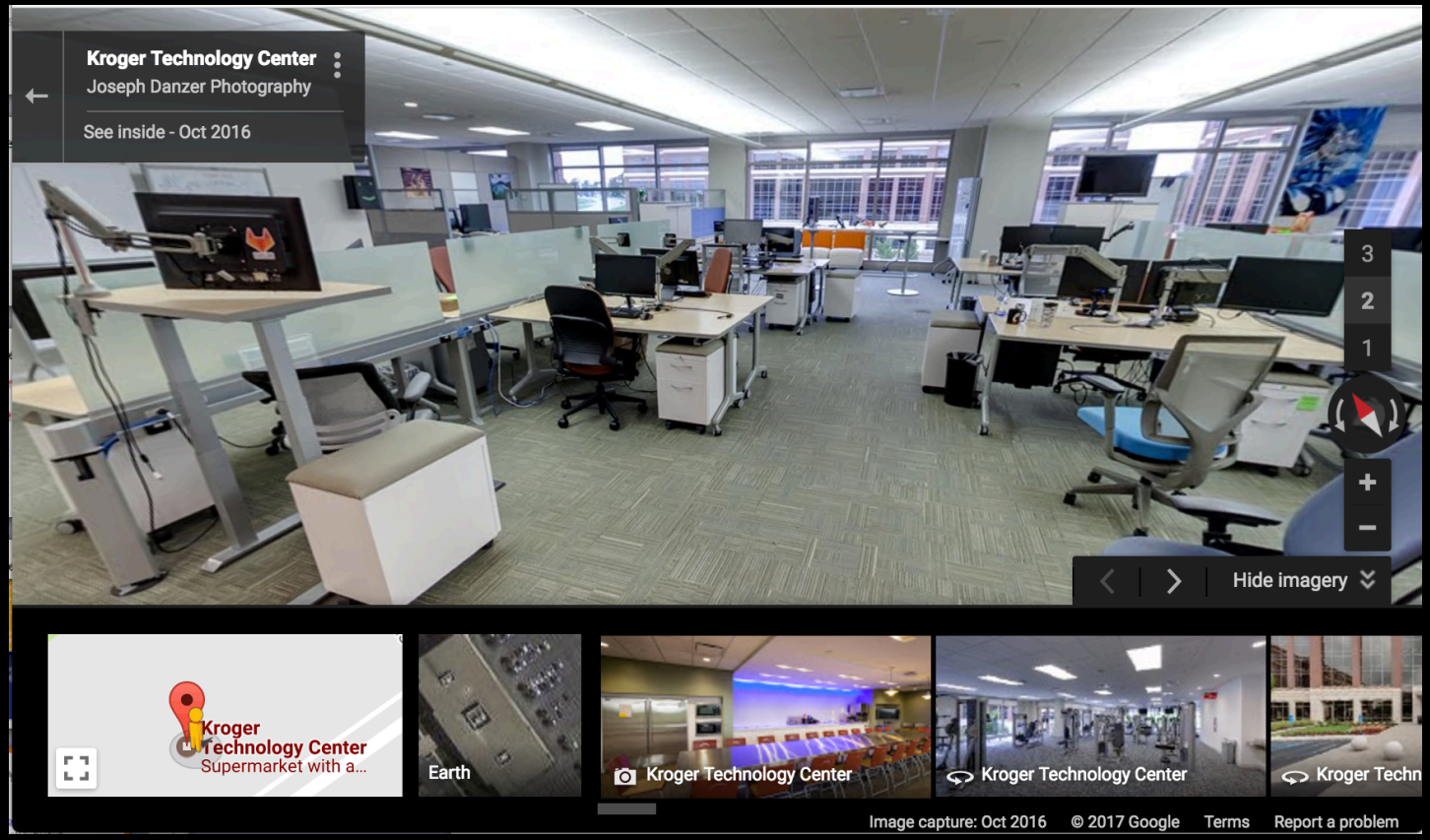
- Technical Lead
- Customer Facing Website
- Full-stack Web Teams
  - 20+ teams
  - JVM (Java, Clojure)
  - JS (ES6, ClojureScript)
  - (a bit of GOLANG)

Kroger


Baker's
City Market
Food & Pharmacy
Dillons
FOOD STORES
Food 4 Less
Foods Co.
Fred Meyer
Food Stores
fry's
Gerbes
Super Markets
Jay C
Food Stores
King Soopers
Kroger
Owen's
Pay Less
Super Markets
QFC
Quality Food Centers
Ralphs
Smith's
market fresh. money smart.

Our Office

- In https://maps.google.com
  - Search for “Kroger Technology Center”



Let’s not talk about me…

- So you have a development team.
  - Web Development
  - JS SPA
  - Java Backend

You've got a team

They argue…

(a lot)

(at every decision)

The Problems with Java

- Slow
- Verbose
- Heavyweight Deployments
- Really Verbose

Why is JS so Cool?

- Fast
- Code is concise
- Lightweight Deployment

What we will Cover

- Friends don’t let friends write ES5 (use ES6)
- Java8
- Spring Boot
- Build Systems: Gradle and NPM
- Simple Strongly typed objects
- Scalability

Don’t use ES5

ES6 solves so many problems:

- Formalize modules
- Scope
- Arrow functions
- String templates and multi-line
- Rest/Spread operations
- Scope

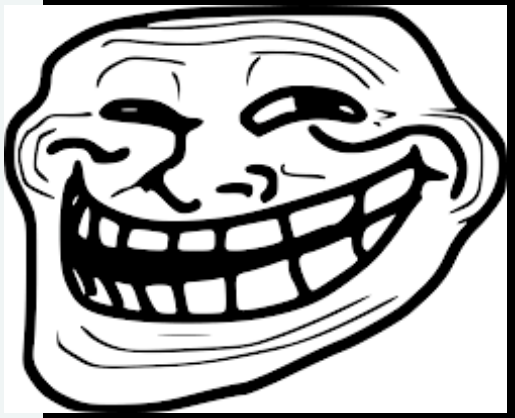
Scope Hell (ES5)

- It’s not so bad… right?
- What about IIFE (immediately-invoked function expression)?

```
jumbleMumble = function() {
    if(a != 5) {
        a = 25;
        for(var c = b; c < 100; c++) {
            a = 50;
        }
    }
    var c = a;
    a = 100;
    b =  b + c;
    var a = 0;
};

var a = 5;
var b = 10;
jumbleMumble();
var c = a + 15;

console.log("a is: " + a); // a is: ??
console.log("b is: " + b); // b is: ??
console.log("c is: " + c); // c is: ??
```



```
jumbleMumble = function() {
    if(a != 5) {
        a = 25;
        for(var c = b; c < 100; c++) {
            a = 50;
        }
    }
    var c = a;
    a = 100;
    b =  b + c;
    var a = 0;
};

var a = 5;
var b = 10;
jumbleMumble();
var c = a + 15;

console.log("a is: " + a); // a is: 5
console.log("b is: " + b); // b is: ??
console.log("c is: " + c); // c is: ??
```

```
jumbleMumble = function() {
    if(a != 5) {
        a = 25;
        for(var c = b; c < 100; c++) {
            a = 50;
        }
    }
    var c = a;
    a = 100;
    b =  b + c;
    var a = 0;
};

var a = 5;
var b = 10;
jumbleMumble();
var c = a + 15;

console.log("a is: " + a); // a is: 5
console.log("b is: " + b); // b is: 60
console.log("c is: " + c); // c is: ??
```

```
jumbleMumble = function() {
    if(a != 5) {
        a = 25;
        for(var c = b; c < 100; c++) {
            a = 50;
        }
    }
    var c = a;
    a = 100;
    b =  b + c;
    var a = 0;
};

var a = 5;
var b = 10;
jumbleMumble();
var c = a + 15;

console.log("a is: " + a); // a is: 5
console.log("b is: " + b); // b is: 60
console.log("c is: " + c); // c is: 20
```

Don’t use old Java

- Java 1.8 (8) provides
  - Lambdas
  - Streams
  - java.nio (this isn’t that new)
  - CompletableFutures

Simple Web Server

- Returns “Hello World!” on “/”

Es6 Hello world Web Server

```
1   import express from 'express';
2
3   const app = express();
4
5   app.get('/', (req, res) => {
6           res.send('Hello World!');
7   });
8
9   app.listen(3000, () => {
10          console.log(`Listening at http://localhost:3000`);
11  });
```

Java 8 Hello World Web Server

```
1   package helloWorld;
2
3   import org.springframework.boot.SpringApplication;
4   import org.springframework.boot.autoconfigure.SpringBootApplication;
5   import org.springframework.web.bind.annotation.RequestMapping;
6   import org.springframework.web.bind.annotation.RestController;
7
8
9   @SpringBootApplication
10  @RestController
11  public class WebServer {
12
13      public static void main(String[] args) {
14          SpringApplication.run(WebServer.class, args);
15      }
16
17      @RequestMapping("/")
18      public String index() { return "Hello World!"; }
21  }
```

Spring boot

- Many are moving away from external web containers
- Converts POJO <-> JSON
- Spring boot provides
  - (Embedded) Tomcat
  - (Embedded) Jetty
  - (Embedded) Undertow

What about real Logic

- From Multiple lists
- Get items for a single list: “small list”
- Count items in list

NO FOR LOOPS!

JS Example

```
1
2  let items = [
3      {list: "small list", name: "zucchini", quantity: 2},
4      {list: "big list", name: "bananas", quantity: 105},
5      {list: "big list", name: "apples", quantity: 110},
6      {list: "small list", name: "bananas", quantity: 5},
7      {list: "big list", name: "zucchini", quantity: 102},
8      {list: "small list", name: "apples", quantity: 10}
9  ];
10
11 var itemCount = items
12         .filter((item) => item.list === "small list")
13         .map((item) => item.quantity)
14         .reduce((a, b) => a + b);
15
16 console.log(itemCount);
```

Java 8 Example

```
ListItem[] items = {
        new ListItem("small list", "zucchini", 2),
        new ListItem("big list", "bananas", 15),
        new ListItem("big list", "apples", 110),
        new ListItem("small list", "bananas", 5),
        new ListItem("big list", "zucchini", 102),
        new ListItem("small list", "apples", 10)
};
int itemCount = stream(items)
        .filter((item) -> item.getList().equals("small list"))
        .map(ListItem::getQuantity)
        .reduce(0, (a, b) -> a + b);
System.out.println(itemCount);
```

Strongly Typed

Strongly Argued about…

What you want

```
1  package shoppingList.model;
2
3  public class ListItem {
4
5      String id;
6      String name;
7      int quantity;
8  }
```

What you get

```
1  package typedObject;
2
3  public class ListItem {
4
5      String id;
6      String name;
7      int quantity;
8
9      public String getId() {
10         return id;
11     }
12
13     public void setId(String id) { this.id = id; }
16
17     public String getName() { return name; }
20
21     public void setName(String name) { this.name = name; }
24
25     public int getQuantity() { return quantity; }
28
29     public void setQuantity(int quantity) { this.quantity = quantity; }
32
33     @Override
34     public String toString() {
35         return "ListItem{" +
36                 "id='" + id + '\'' +
```

Part

```
38                  ", quantity=" + quantity +
39                  '}';
40      }
41
42      @Override
43      public boolean equals(Object o) {
44          if (this == o) return true;
45          if (o == null || getClass() != o.getClass()) return false;
46
47          ListItem listItem = (ListItem) o;
48
49          if (quantity != listItem.quantity) return false;
50          if (id != null ? !id.equals(listItem.id) : listItem.id != null) return false;
51          return name != null ? name.equals(listItem.name) : listItem.name == null;
52
53      }
54
55      @Override
56      public int hashCode() {
57          int result = id != null ? id.hashCode() : 0;
58          result = 31 * result + (name != null ? name.hashCode() : 0);
59          result = 31 * result + quantity;
60          return result;
61      }
62  }
```

Say it ain't so…

- Well it's not
- Project Lombok
  - Compile time annotations
  - Adds in the methods.

How it looks with Lombok

```
1   package typedObject;
2
3   import lombok.Data;
4
5   @Data
6   public class ListItemShort {
7
8       String id;
9       String name;
10      int quantity;
11  }
```

Really?

- Yep

```
1  package typedObject;
2
3  import org.junit.Test;
4
5  import static org.junit.Assert.*;
6
7  public class ListItemShortTest {
8
9      @Test
10     public void should_be_value_object() {
11         ListItemShort item = new ListItemShort();
12         item.setQuantity(1);
13         item.setName("name");
14         item.setId("1234");
15         assertEquals(1, item.getQuantity());
16         assertEquals("name", item.getName());
17         assertEquals("1234", item.getId());
18         assertEquals("ListItemShort(id=1234, name=name, quantity=1)", item.toString());
19     }
20 }
```

Tooling

- What about:
  - Package Management
  - Build scripting.

- Java = XML?

JS Manager

- NPM

```
{
  "name": "hello-world",
  "scripts": {
    "test": "./node_modules/mocha/bin/mocha --compilers js:babel-core/regis
    "build": "./node_modules/babel-cli/bin/babel.js src/ -d dist/",
    "clean": "rm -rf dist",
    "start": "node ./dist/main.js"
  },
  "dependencies": {
    "express": "^4.14.0"
  },
  "devDependencies": {
    "babel": "^6.5.2",
    "babel-cli": "^6.18.0",
    "babel-preset-es2015": "^6.18.0",
    "mocha": "^3.2.0",
    "supertest": "^2.0.1"
  }
}
```

Java Build Manager

- Gradle

```
buildscript {
    repositories { mavenCentral() }
    dependencies {
        classpath("org.springframework.boot:spring-boot-gradle-plugin:1.4.3.RELEASE") }
}
apply plugin: 'java'
apply plugin: 'idea'
apply plugin: 'org.springframework.boot'

repositories { mavenCentral() }

dependencies {
    compile 'org.springframework.boot:spring-boot-starter-web'
    testCompile 'junit:junit',
                'org.springframework:spring-test'
}
```

Scalability

- Node is non-blocking!
- So is Java!
- Futures are the future!
- I LOVE EXCLAMATION POINTS!!!!!

Example Async

- Spring MVC

```
@RequestMapping("/items")
public CompletableFuture<List<ListItem>> getItems() {
    CompletableFuture<List<ListItem>> items = shoppingListService.findAll();

    items.thenApply(listItems -> listItems.stream()
                .map(ListItem::prunePrivateData)
                .collect(toList()));

    return items;
}
```

But My Call Is Synchronous!

- My functions don't magically return Futures.
- Like JDBC.

Actually, you can.

Yeah .supplyAsync() !

- CompletableFuture will queue your request up and run it async - ForkJoinPool

```
@RequestMapping("/defaultItem")
public CompletableFuture<ListItem> fe        tem() {
    final String defaultItemName = "de        Name";
    return CompletableFuture.supplyAsync(() -> getItemThroughJDBC(defaultItemName));
}
```

```
public ListItem getItemThroughJDBC(String itemName) {
    ListItem listItem = new ListItem("list", itemName, 1);

    // Pretend I do JDBC stuff here...

    return listItem;
}
```

Closing Thoughts

- We use these ideas in production
- Go faster
- Write less code
- Find ways to run things faster
- Take time to automate everything
- Be Impatient
- Close the Feedback loop
- Love the problem, not the solution

Questions?

Thank You.

- Stephen.shary@gmail.com
- github.com/STeveShary
- stevesmashcode.com
- linkedin.com/in/stephenshary