Tim Fox
Father of Vert.x

Vert.x:

This ain't your Dad's Node!

Matt Stine
Enterprise Java/Cloud Consultant
matt.stine@gmail.com
http://mattstine.com
Twitter: @mstine

The C10K Problem

A problem has been detected and Windows has been shut down to prevent dama
to your computer.
IRQL_NOT_LESS_OR_EQUAL
If this is the first time you've seen this Stop error screen,
restart your computer. If this screen appears again, follow
these steps:
Check to make sure any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any Windows updates you might need.
If problems continue, disable or remove any newly installed hardware
or software. Disable BIOS memory options such as caching or shadowing.
If you need to use Safe Mode to remove or disable components, restart
your computer, press F8 to select Advanced Startup Options, and then
select Safe Mode.
Technical information:
*** STOP: 0x0000000A (0x00000A35,0x00000002,0x00000001,0x
Beginning dump of physical memory
Physical memory dump complete.
Contact your system administrator or technical support group for further
assistance.
SAMSUNG

SERVER PUSH

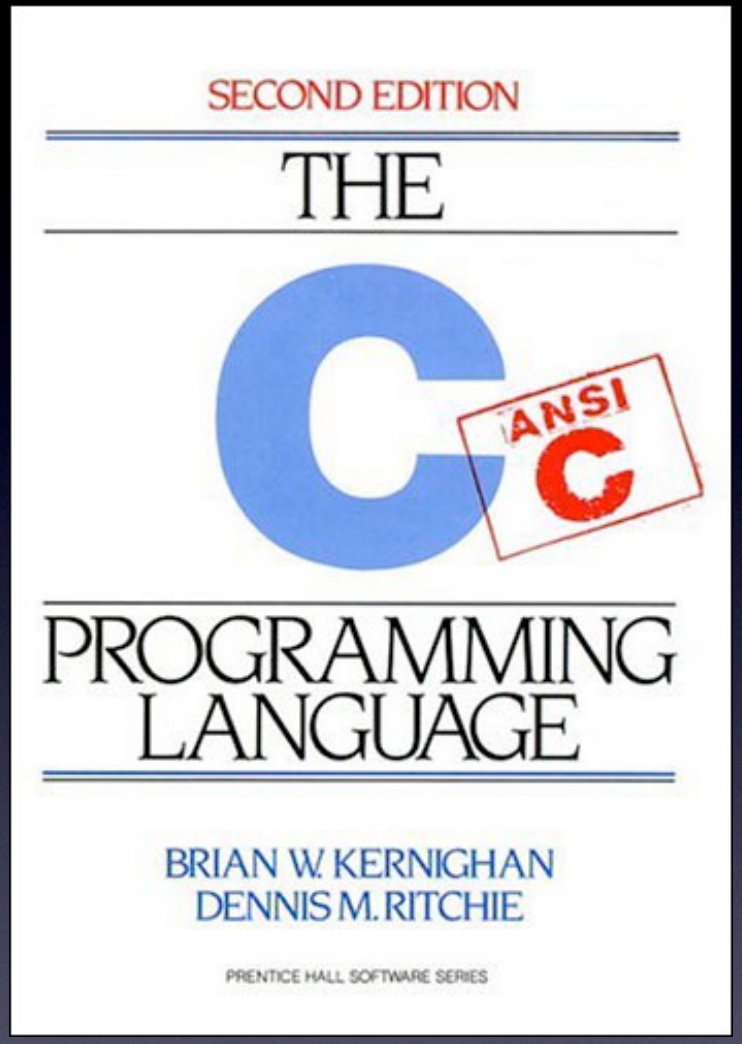
Ryan Dahl
Father of Node.js

How?!?!

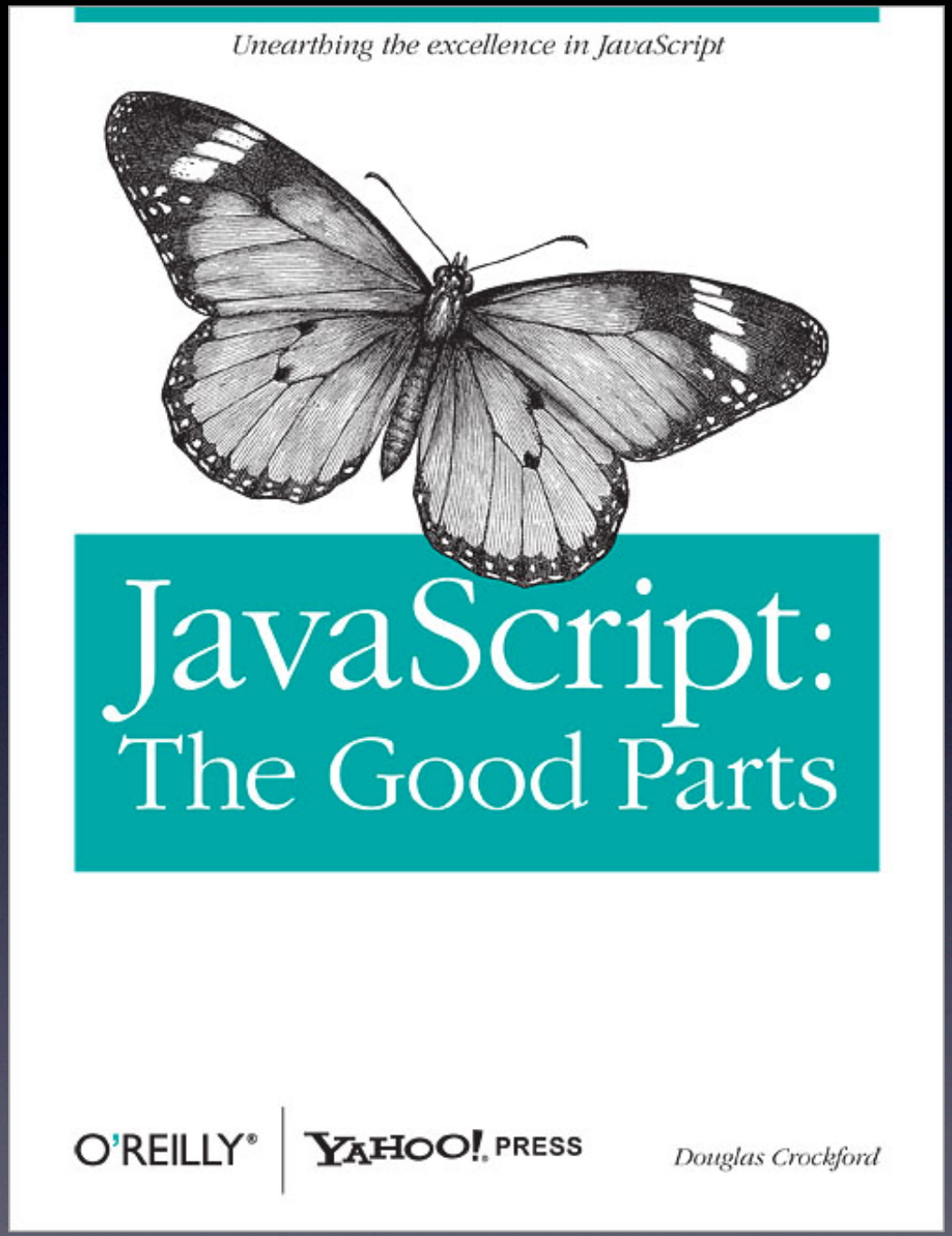
2008:

Thread Per Request/Response

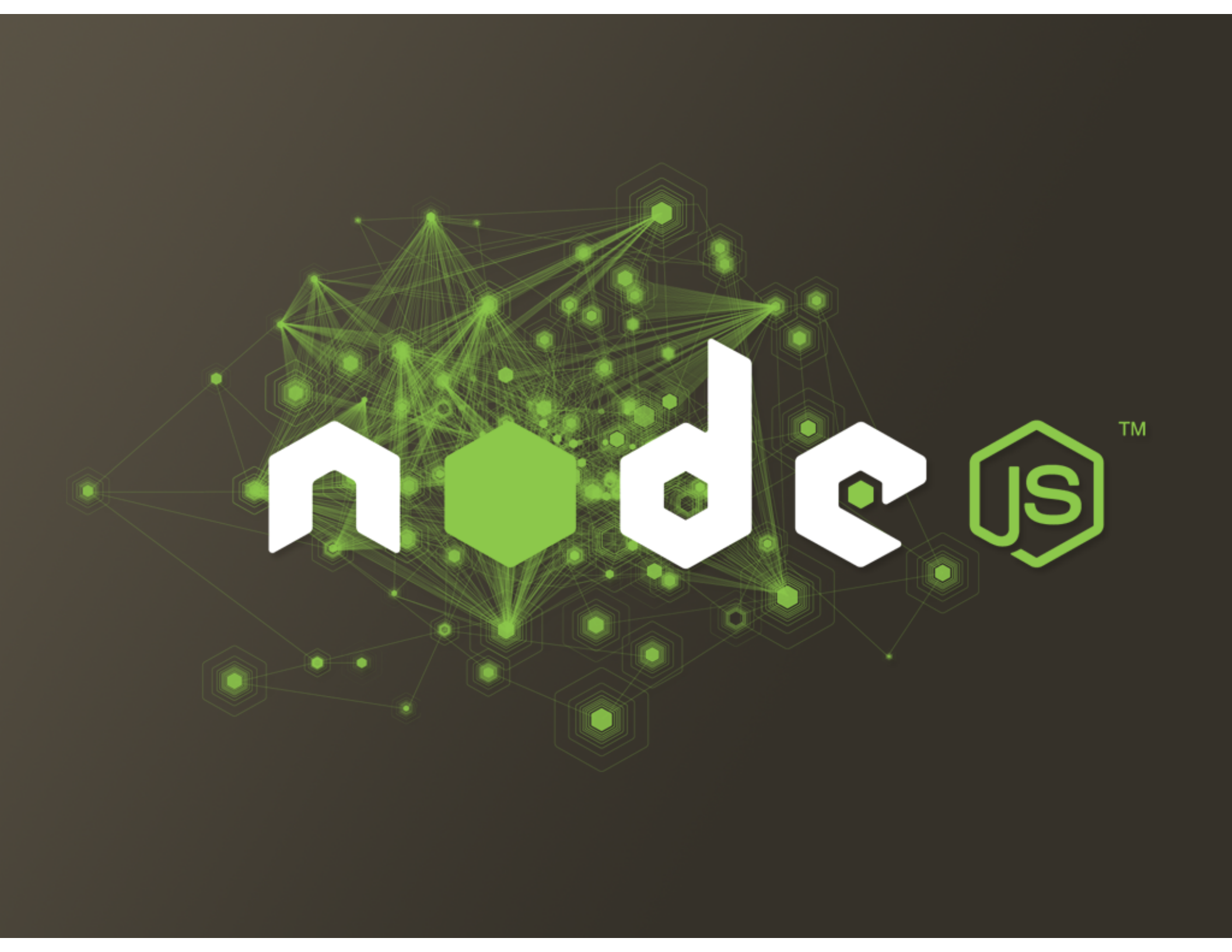
http://www.xtranormal.com/watch/6995033/mongo-db-is-web-scale



Non-blocking UNIX sockets


Unearthing the excellence in JavaScript
JavaScript:
The Good Parts
O'REILLY®
YAHOO! PRESS
Douglas Crockford


8


node
JS
TM

But Node has some shortcomings...

1. JavaScript!
2. Vertical Scaling
3. Interprocess Communication
4. Event Loop

vert.x

Effortless asynchronous application development for the modern web and enterprise

Polyglot

Write your application components in **JavaScript**, **CoffeeScript**, **Ruby**, **Python**, **Groovy** or **Java**. Or mix and match several programming languages in a single app.

Simplicity

...without being simplistic. Create real, scalable applications in just a few lines of code. No sprawling xml config.

Scalability

Scale using messaging passing and immutable shared data to efficiently utilise your server cores.

Concurrency

Super-simple concurrency model frees you from the hassles of traditional multi-threaded programming.

http://vertx.io

Brendan Eich
Father of JavaScript

Problem #1:

JavaScript

WAT

https://www.destroyallsoftware.com/talks/wat

Neal Ford
Father of Polyglot Programming

Solution #1:

Polyglot Programming

http://memeagora.blogspot.com/2006/12/polyglot-programming.html

Hello World

in every Vert.x language!

```
import org.vertx.java.core.Handler;
import org.vertx.java.core.http.HttpServerRequest;
import org.vertx.java.deploy.Verticle;

public class HelloWorld extends Verticle {
  public void start() {
    vertx.createHttpServer()
      .requestHandler(new Handler<HttpServerRequest>() {
        public void handle(HttpServerRequest req) {
          req.response.end("Hello World!");
        }
      }).listen(8080);
  }
}
```

Java

```
vertx.createHttpServer().requestHandler { req ->
  req.response.end "Hello World!"
}.listen(8080)
```

```ruby
require "vertx"

Vertx::HttpServer.new.request_handler do |req|
  req.response.end "Hello World!"
end.listen(8080)
```

Ruby

```
import vertx

server = vertx.create_http_server()

@server.request_handler
def request_handler(req):
  req.response.end("Hello World!")
server.listen(8080)
```

Python

```
load('vertx.js')

vertx.createHttpServer().requestHandler(function(req) {
  req.response.end('Hello World!');
}).listen(8080)
```

```
load "vertx.js"

vertx.createHttpServer().requestHandler((req) ->
  req.response.end "Hello World!"
).listen 8080
```

FUTURE

Theoretically, any JVM language or language that compiles to a supported language (e.g. ClojureScript → JavaScript)

Problem #2:

Vertical Scaling

Lars Bak
Father of V8

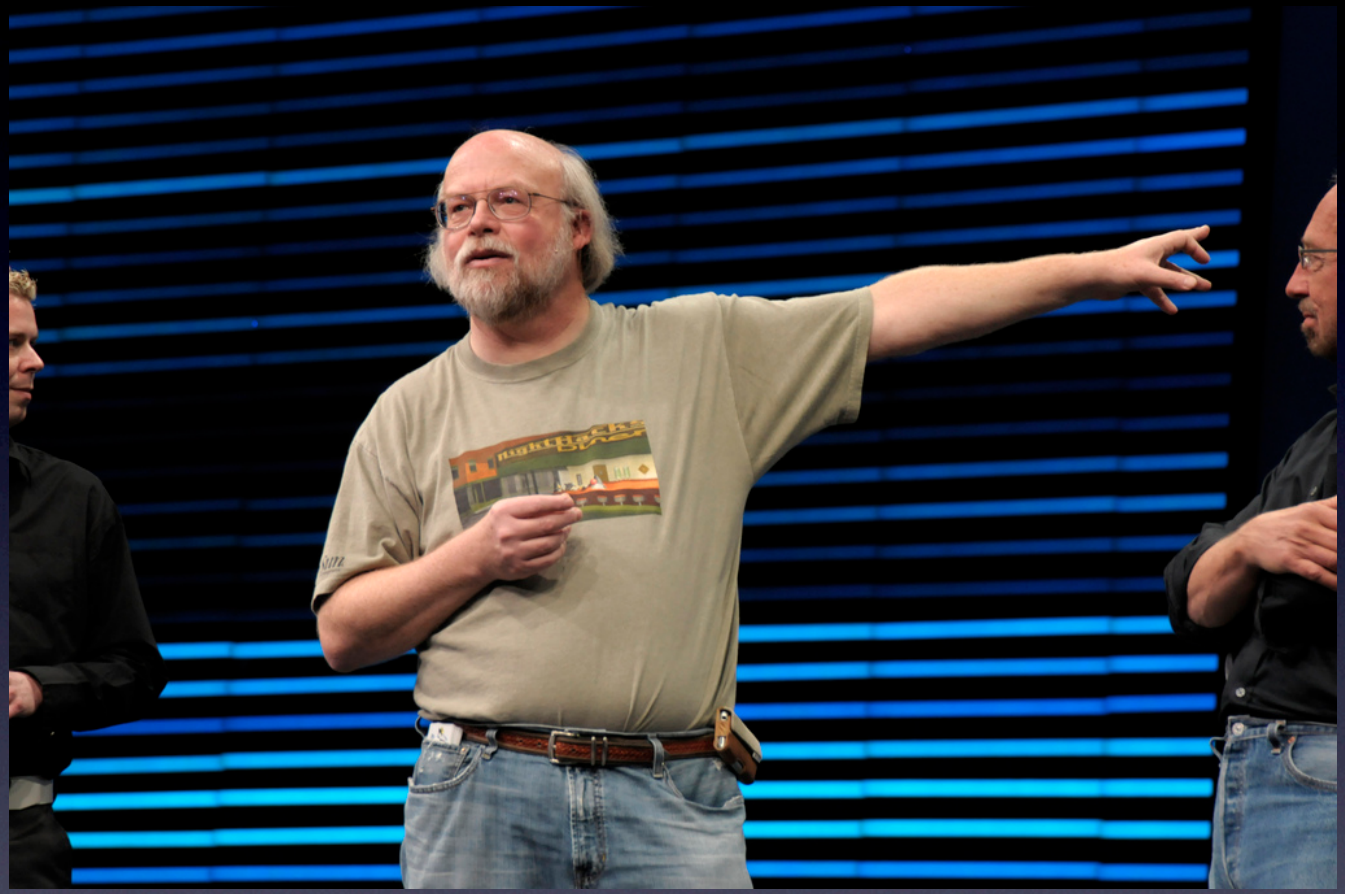
But what about multiple-processor concurrency? Aren't threads necessary to scale programs to multi-core computers? You can start new processes via `child_process.fork()` these other processes will be scheduled in parallel. For load balancing incoming connections across multiple processes use the cluster module.

http://nodejs.org/about/

```
var cluster = require('cluster');
var http = require('http');
var numCPUs = require('os').cpus().length;

if (cluster.isMaster) {
  // Fork workers.
  for (var i = 0; i < numCPUs; i++) {
    cluster.fork();
  }
  cluster.on('death', function(worker) {
    console.log('worker ' + worker.pid + ' died');
  });
} else {
  // Worker processes have a http server.
  http.Server(function(req, res) {
    res.writeHead(200);
    res.end("hello world\n");
  }).listen(8000);
}
```

Node: Up and Running

Example 3-11: Using cluster to distribute work



James Gosling
Father of Java

Solution #2:

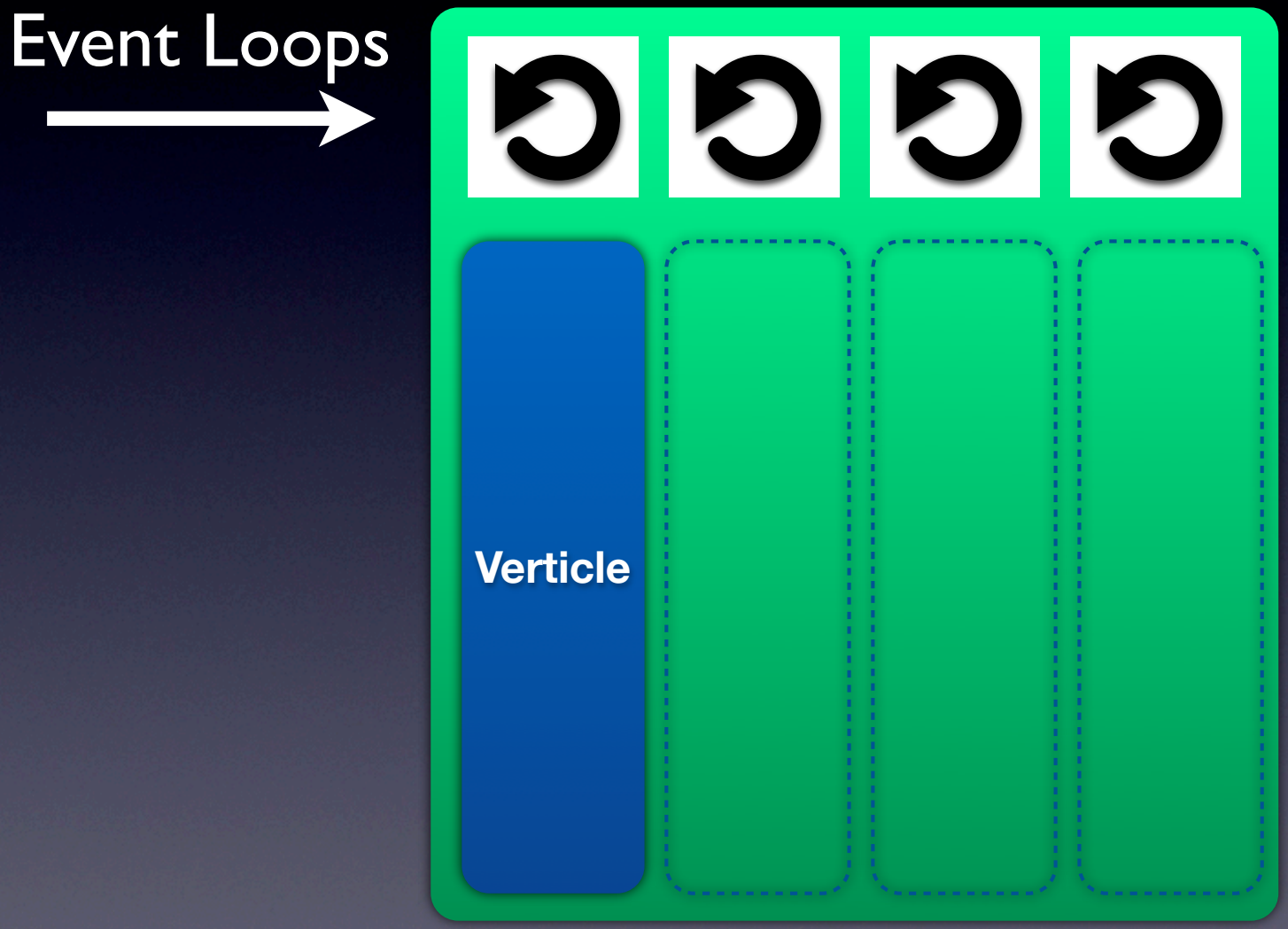
The JVM

The Verticle

- Unit of Deployment

- Script/Java Class

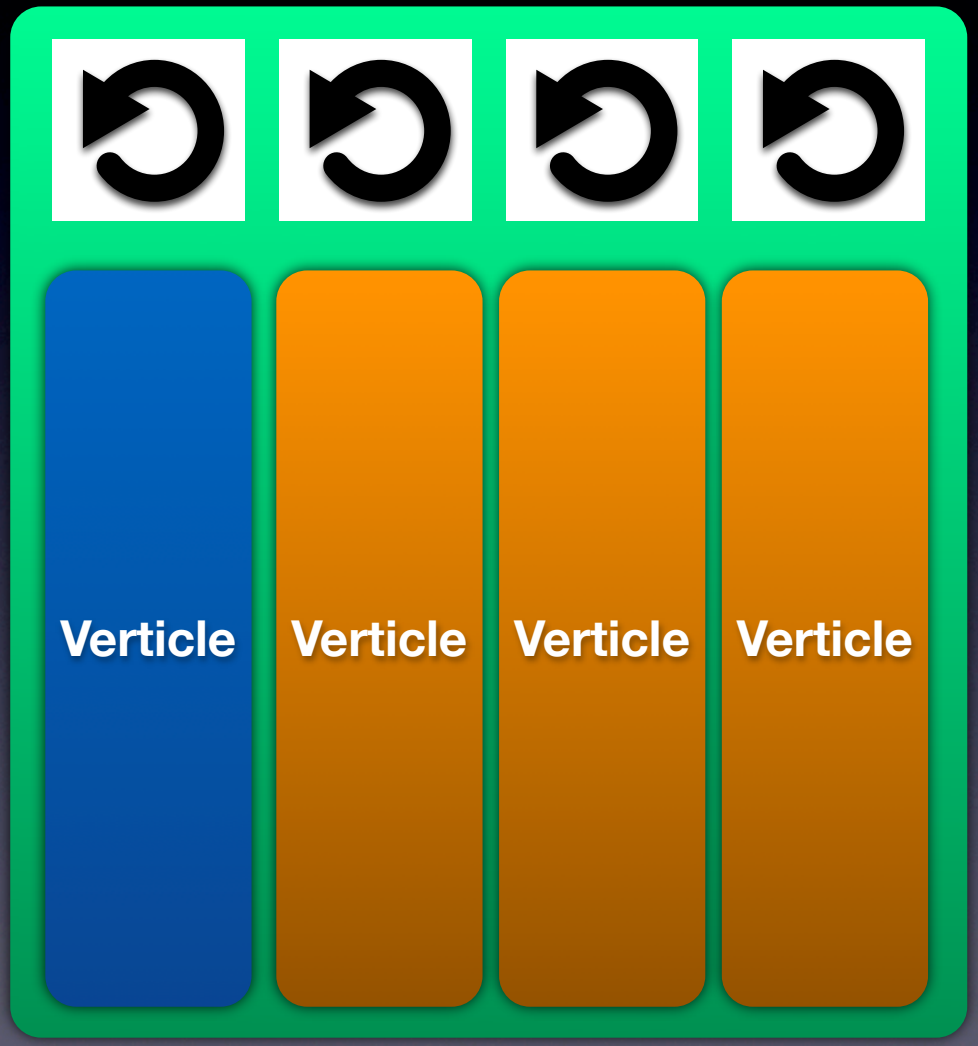
Vert.x Instance




```
Runtime.availableProcessors() == 4
```

Vert.x Instance

Event Loops




```
vertx run
HelloWorld
-instances 4
```

```
Runtime.availableProcessors() == 4
```

Concurrency

- Verticle instance assigned thread/event loop.
- Verticle instance ALWAYS executes on assigned thread.
- Verticles have isolated classloaders and cannot share global state (static members, global variables, etc.)
- Can write all code assuming single threading.

Tony Hoare
Father of CSP
(http://dl.acm.org/citation.cfm?doid=359576.359585)

Problem #3:

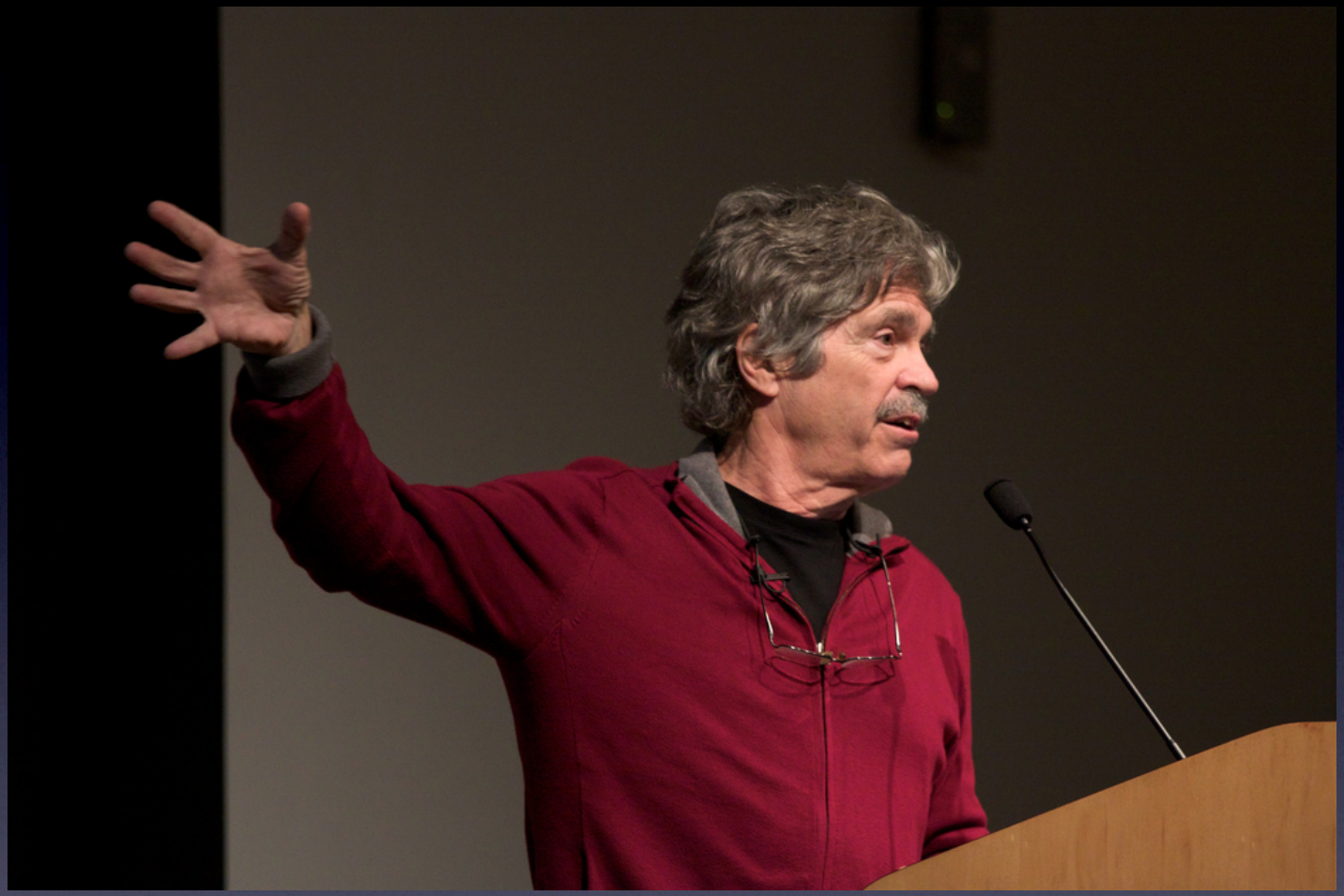
Interprocess Communication

Node.js Communication Options

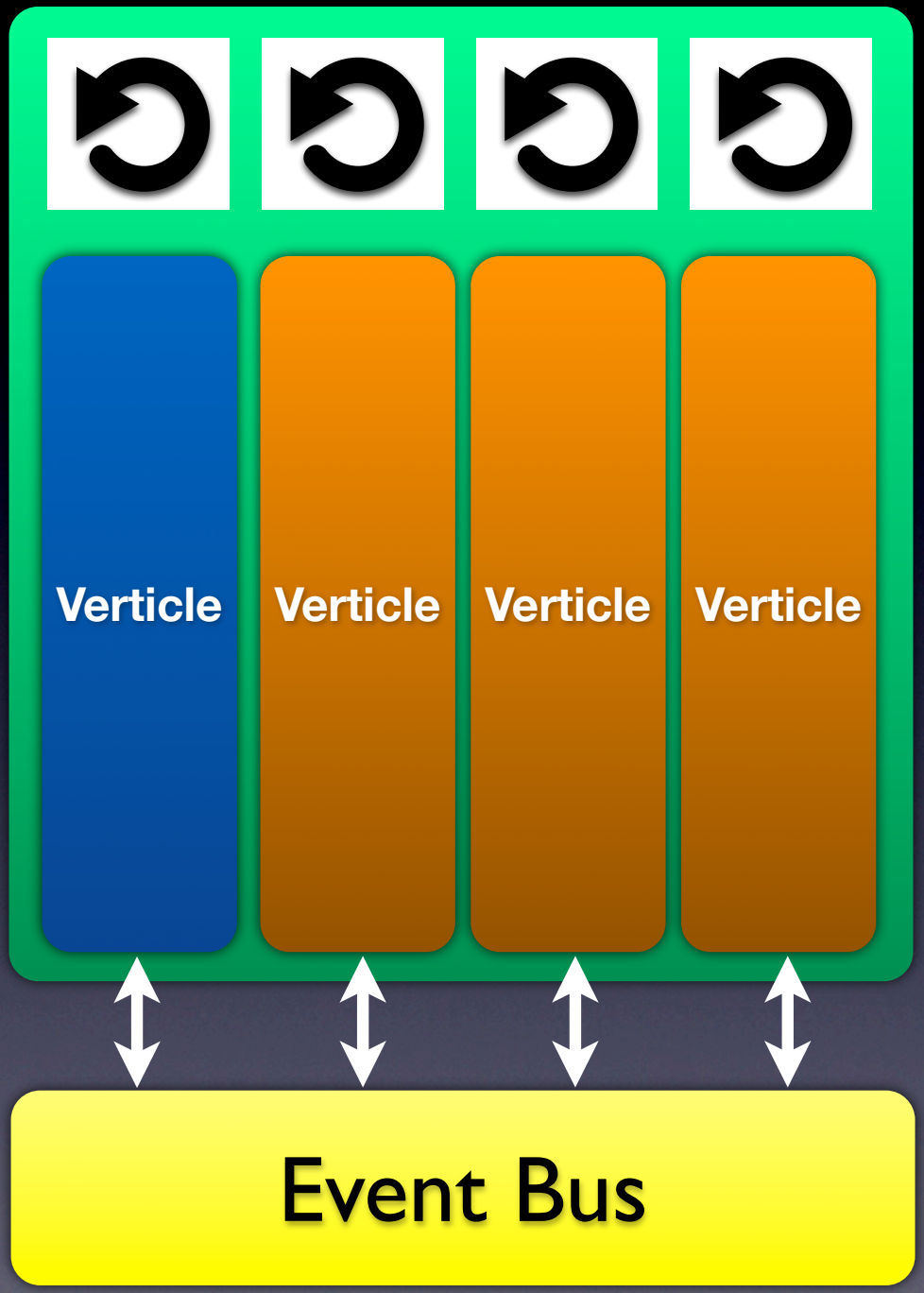
- TCP/UDP/UNIX Sockets
- Redis Pub/Sub (http://redis.io/)
- ZeroMQ (http://www.zeromq.org/)
- Process Signaling/Cluster Module
- Eventing Frameworks: Hook.io (dead), JS-Signals, Bean, etc.
- Memcached
- Etc...

Solution #3.1:

The Event Bus



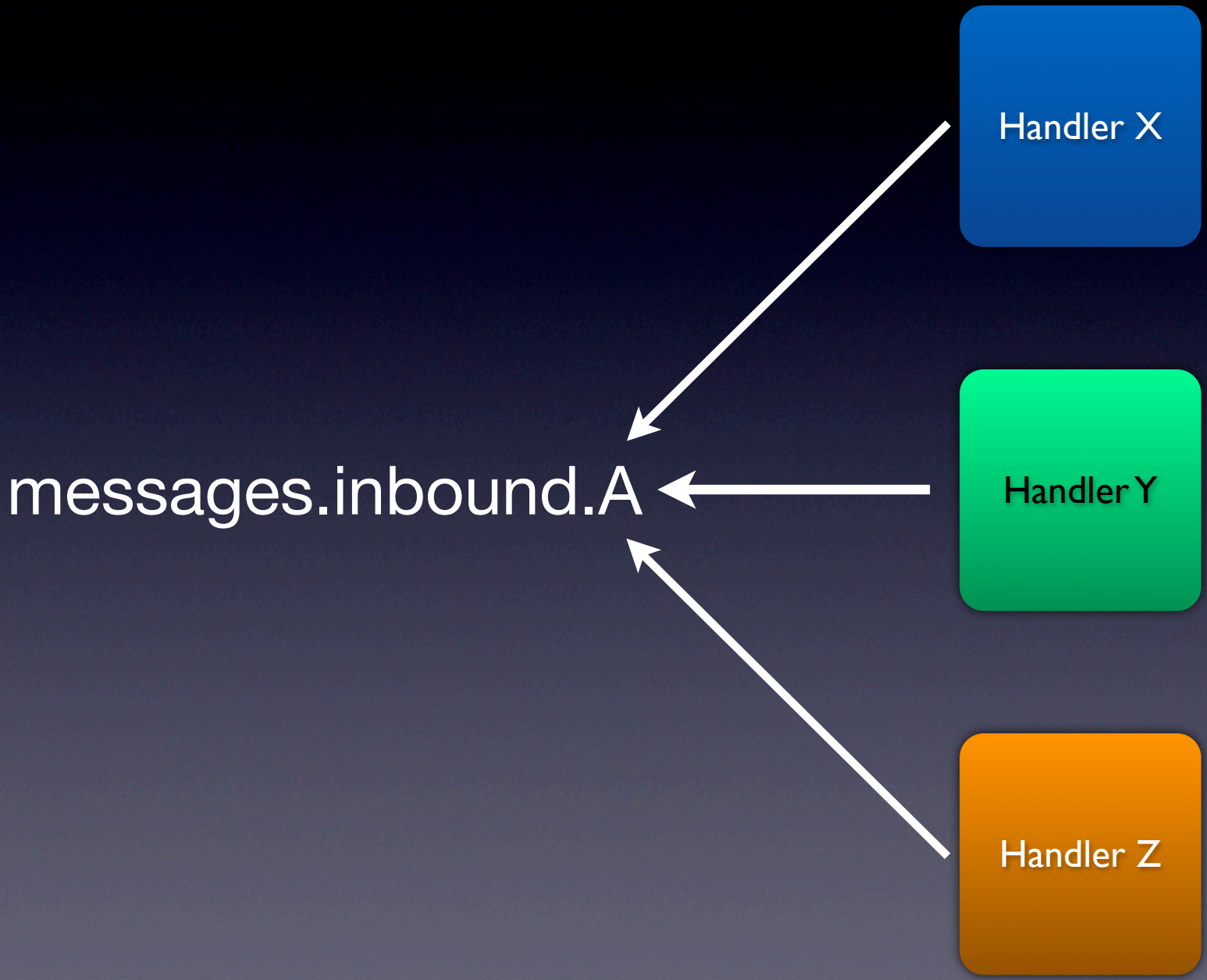
Alan Kay
Father of Smalltalk


Verticle
Verticle
Verticle
Verticle
Event Bus

Addressing

- Simply a String
- Dot-style namespacing recommended
- e.g. “messages.inbound.A”

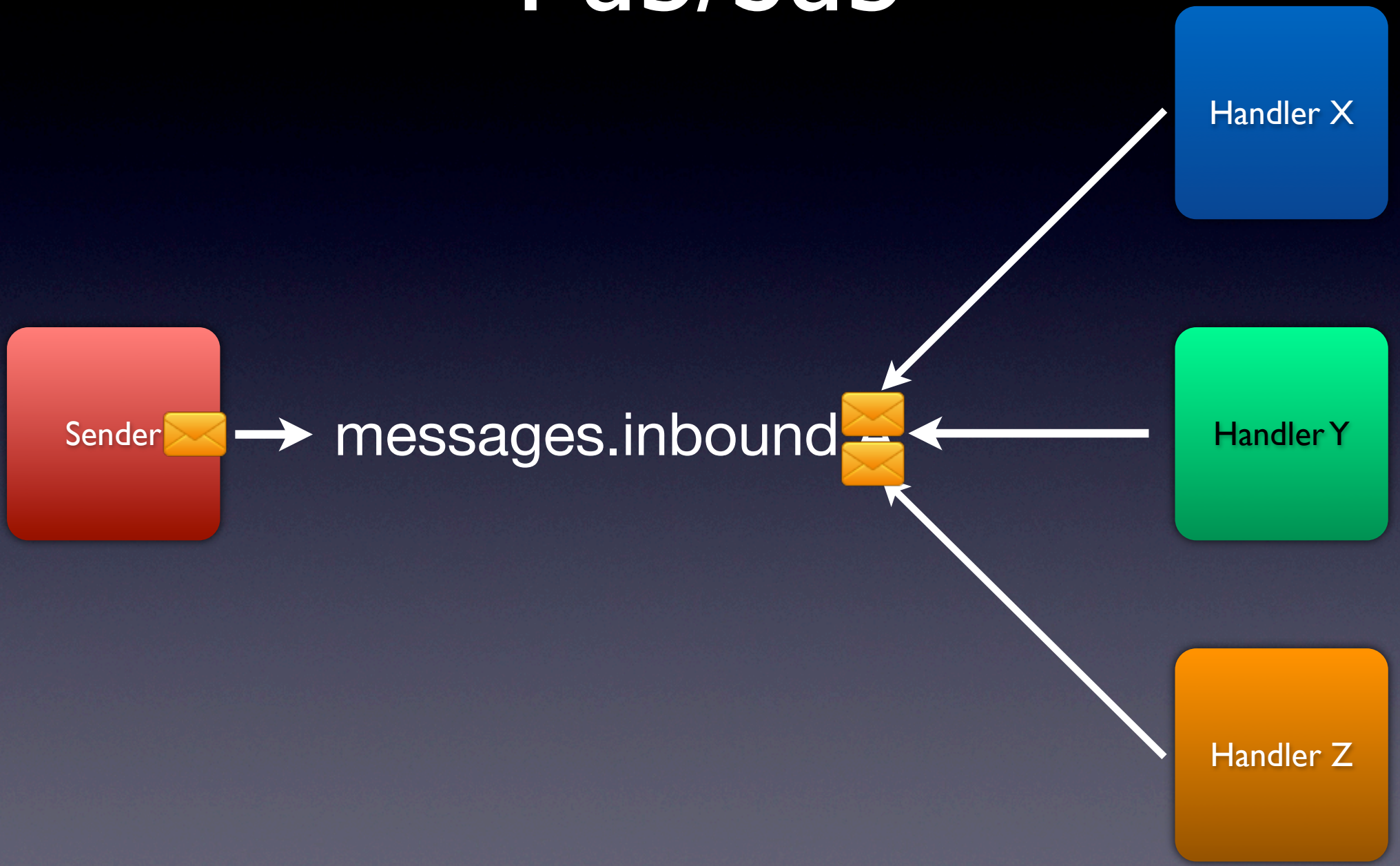
Handler Registration



Handler Registration

```
EventBus eb = vertx.eventBus();

Handler<Message> myHandler = new Handler<Message>() {
  public void handle(Message message) {
    System.out.println("I received a message " + message.body);
  }
};

eb.registerHandler("test.address", myHandler);
```

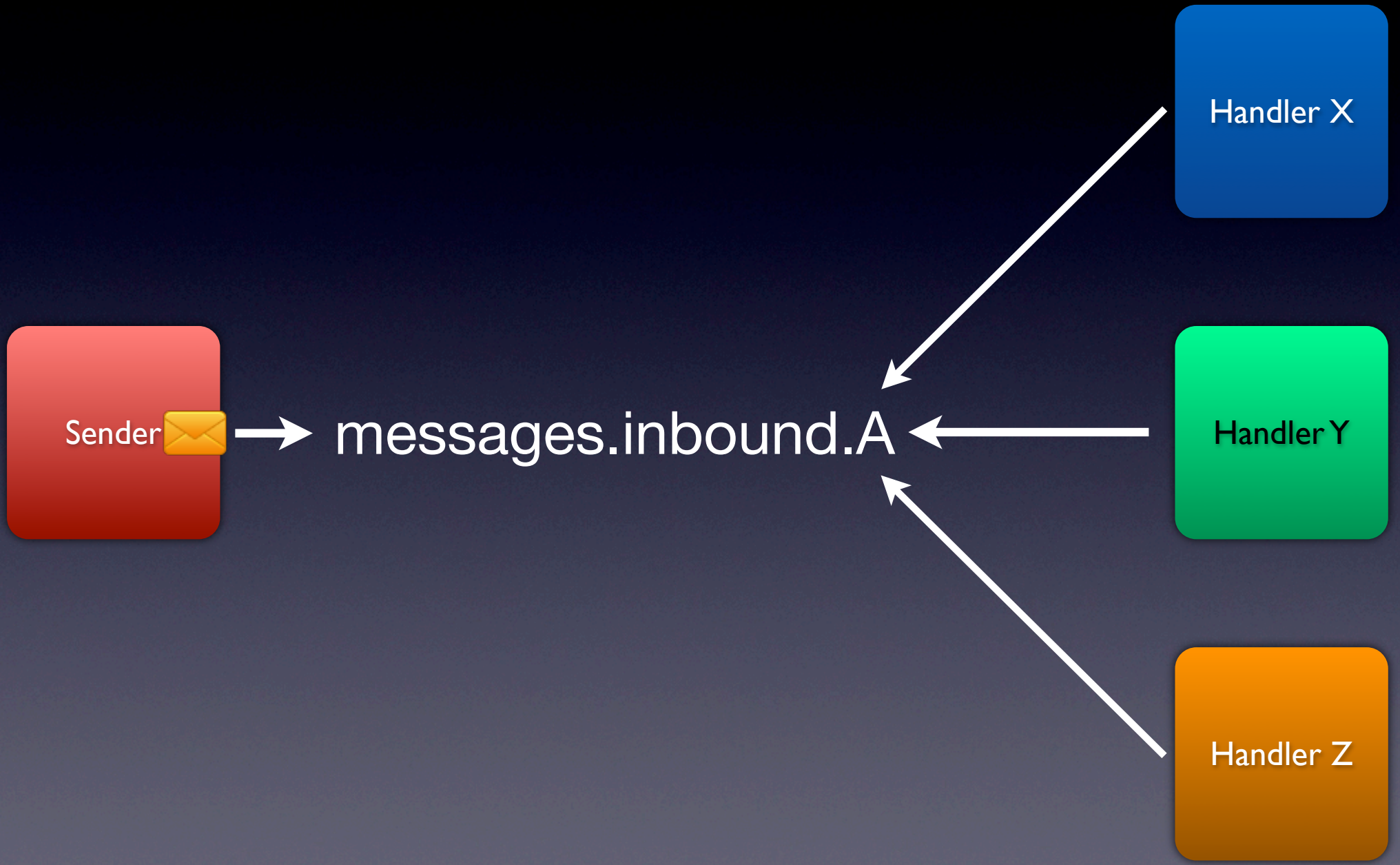
Pub/Sub



Pub/Sub

```
eb.publish("test.address", "hello world");
```

P2P



P2P

```
eb.send("test.address", "hello world");
```

Sender:

```
eb.send("test.address", "This is a message", new Handler<Message<String>>() {
  public void handle(Message<String> message) {
    System.out.println("I received a reply " + message.body);
  }
});
```

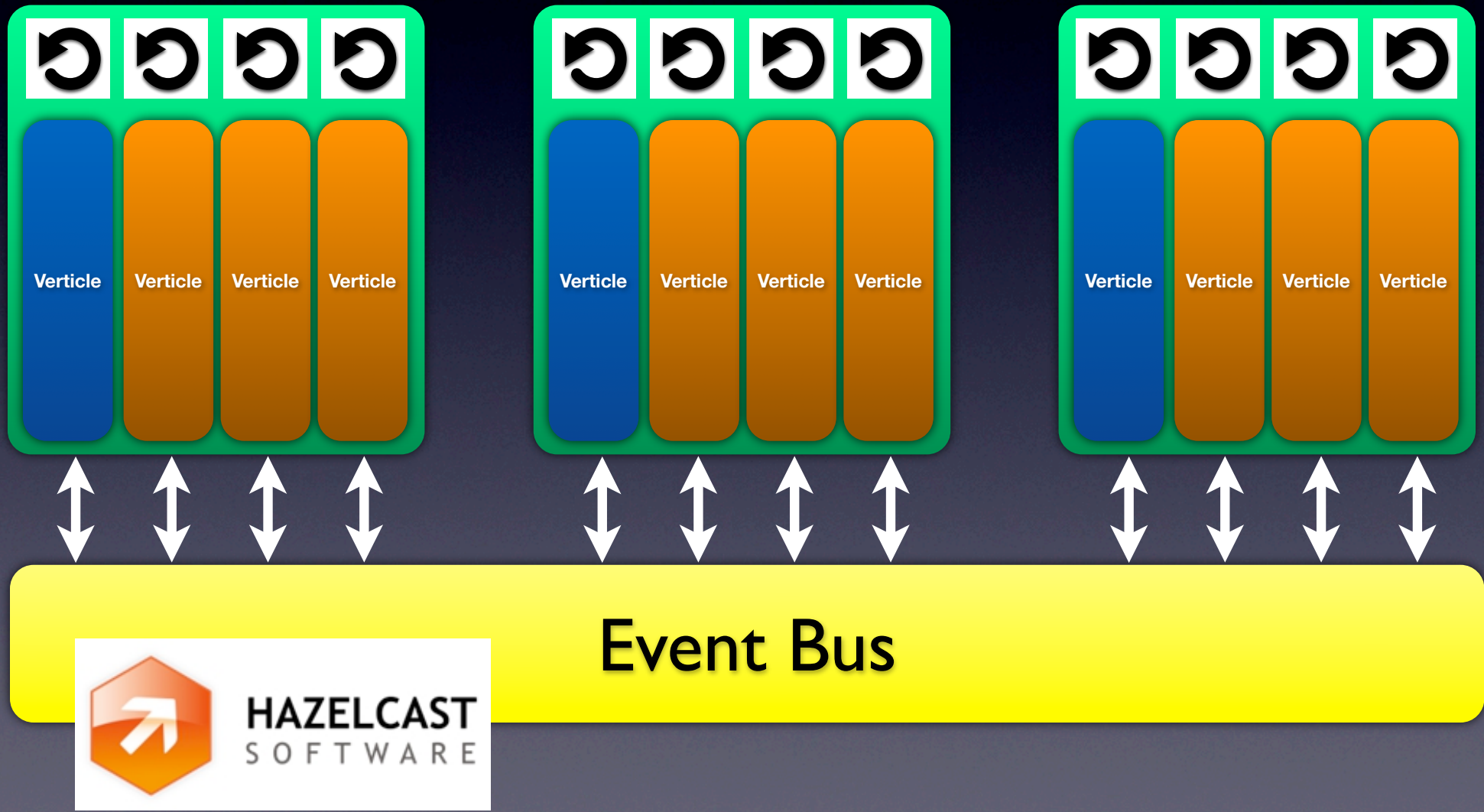
Receiver:

```
Handler<Message<String>> myHandler = new Handler<Message<String>>() {
  public void handle(Message<String> message) {
    System.out.println("I received a message " + message.body);

    // Do some stuff

    message.reply("This is a reply");
  }
};

eb.registerHandler("test.address", myHandler);
```

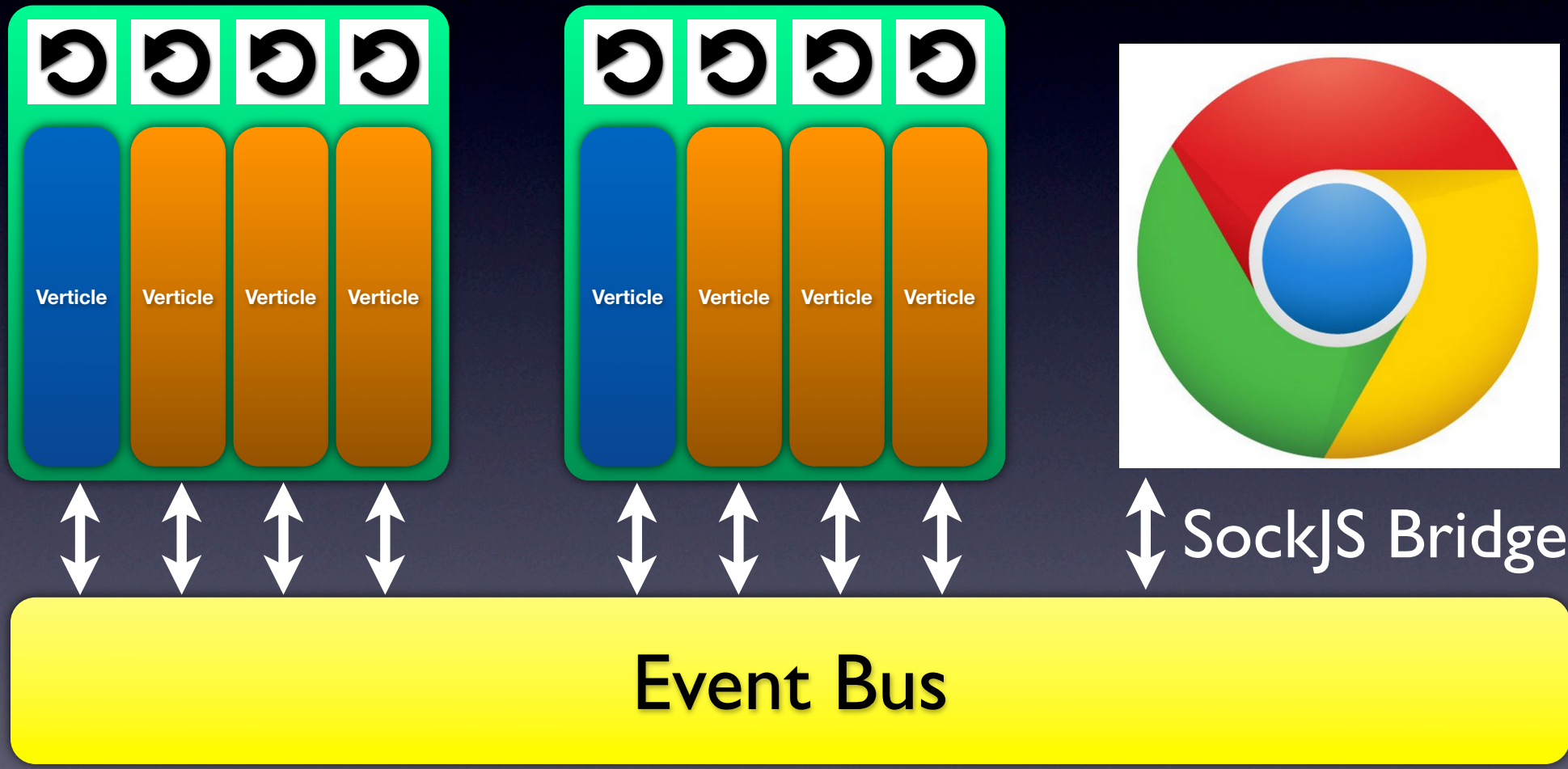
Message Types

- String
- Primitives (int, long, short, float, double, ...)
- Boxed Primitives
- boolean/Boolean
- org.vertx.java.core.json.JsonObject
- org.vertx.java.core.buffer.Buffer

Distributed Vert.x



Into the Browser!



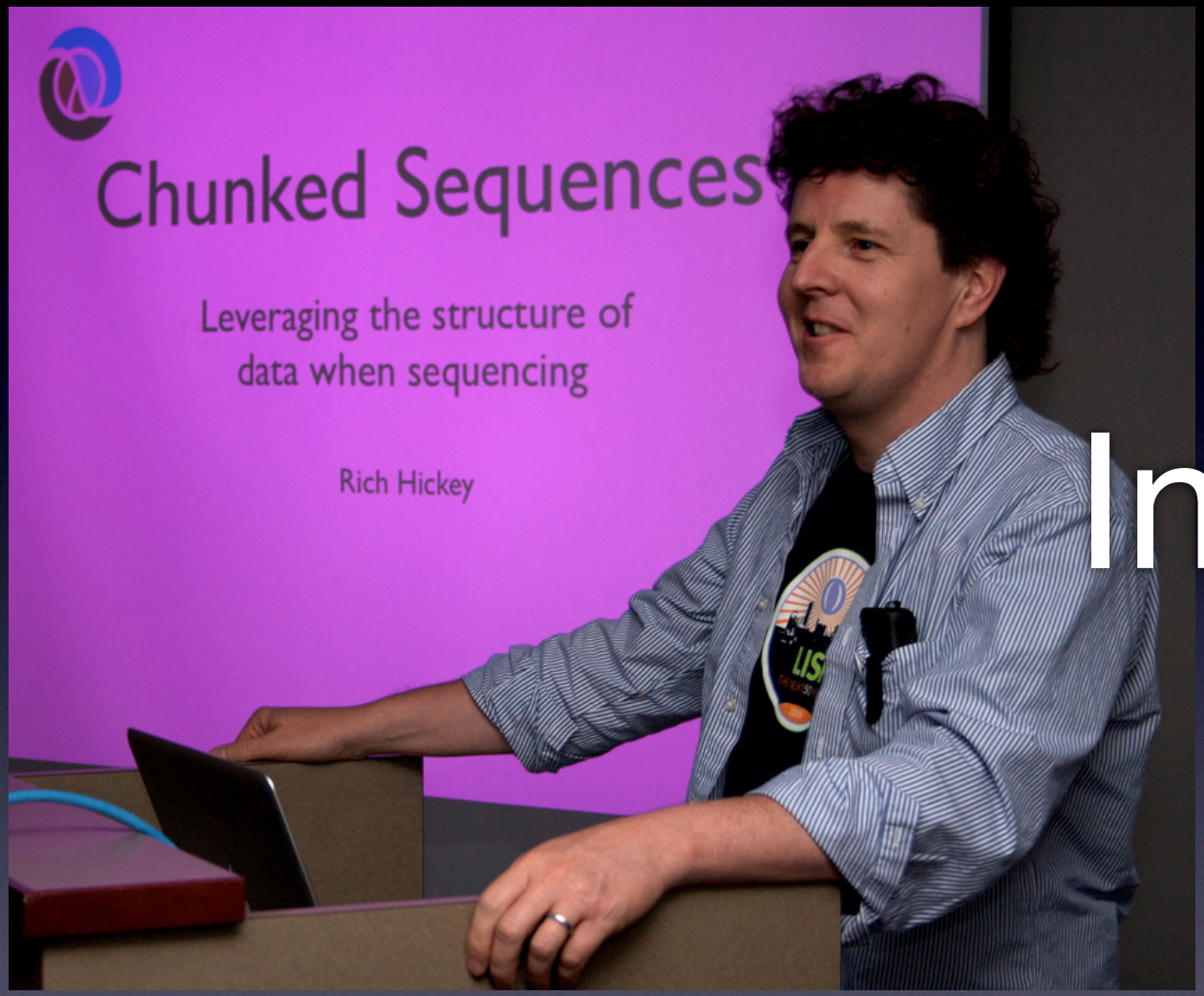
The Server

```
HttpServer server = vertx.createHttpServer();

server.requestHandler(new Handler<HttpServerRequest>() {
    public void handle(HttpServerRequest req) {
        if (req.path.equals("/")) req.response.sendFile("index.html");
        if (req.path.endsWith("vertxbus.js")) req.response.sendFile("vertxbus.js");
    }
});

SockJSServer sockJSServer = vertx.createSockJSServer(server);
sockJSServer.bridge(new JsonObject().putString("prefix", "/eventbus"),
        new JsonArray().add(new JsonObject()),
        new JsonArray().add(new JsonObject()));
server.listen(8080);
```

The Client

```
<script src="http://cdn.sockjs.org/sockjs-0.2.1.min.js"></script>
<script src='vertxbus.js'></script>
<script>
var eb = new vertx.EventBus('http://localhost:8080/eventbus');

eb.onopen = function() {
  eb.registerHandler('msgs.echo', function(message) {
    alert('Echoing message: ' + JSON.stringify(message));
  });
}
</script>
```

Demo



Rich Hickey
Father of Clojure

Solution #3.2:

Shared Immutable State

e.g. In-memory Web Cache

Message Passing?

FAIL

Shared state only dangerous if it is MUTABLE!

Vert.x Shared State

- SharedData Object (`vertx.sharedData()`)
  - collection of java.util.concurrent.ConcurrentMap<K,V>
  - collection of java.util.Set<E> (backed by ConcurrentMap)
- Elements MUST be immutable values (well, sort of...)
- Currently only available *within* a Vertx. instance, not across a cluster.

Allowed Values

- Strings
- Boxed Primitives
- `byte[]`
- `org.vertx.java.core.buffer.Buffer`
- Implementors of `org.vertx.java.core.shareddata.Shareable`

DANGER!

Shared Map

```
// In one verticle...
ConcurrentMap<String, Integer> map = vertx.sharedData().getMap("app.cache");
map.put("user-id", 42);

// In another verticle...
ConcurrentMap<String, Integer> map = vertx.sharedData().getMap("app.cache");
Integer userId = map.get("user-id");
```

Shared Set

```
// In one verticle...
Set<String> set = vertx.sharedData().getSet("app.sessions");
set.add("076146D0-11B3-11E2-892E-0800200C9A66");

// In another verticle...
Set<String> sessions = vertx.sharedData().getSet("app.sessions");
for (String session : sessions) {
  // Do something w/ each session ID
}
```

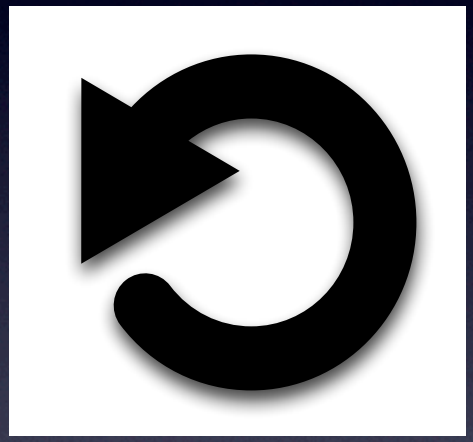
Douglas Schmidt
Father of the Reactor Pattern

Problem #4:

The Event Loop

Reactor Pattern Review

Application registers handlers...



...events trigger handlers.

Event Loop

Reactor Pattern Review

- Single thread / single event loop
- EVERYTHING runs on it
- You MUST NOT block the event loop

Reactor Pattern Problems

- Some work is naturally blocking:
  - Intensive data crunching
  - 3rd-party blocking API's (e.g. JDBC)
- Pure reactor (e.g. Node.js) is not a good fit for this kind of work!

Carl Hewitt
Father of the Actor Model

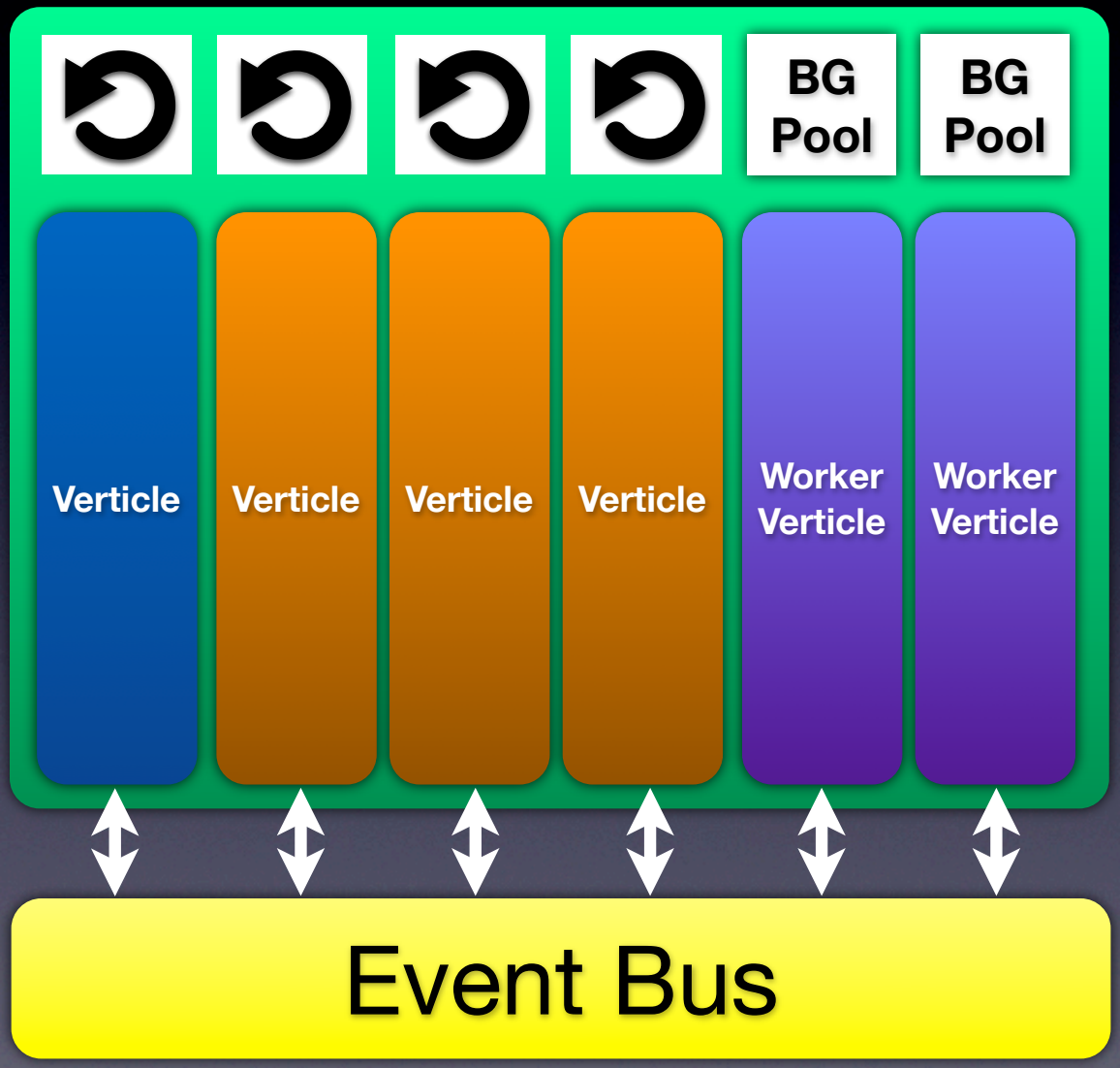
Solution #4:

Worker Verticles

Worker Verticles

- Not assigned a Vert.x event loop thread
- Executes on background thread pool
- Never executed concurrently by more than one thread
- Not allowed to use TCP or HTTP clients/ servers
- Communicate using the event bus
- Should be kept to a minimum

The “Multi-reactor”



```java
public class FibonacciWorker extends Verticle {

    @Override
    public void start() throws Exception {
        final EventBus eventBus = vertx.eventBus();

        eventBus.registerHandler("fib.request", new Handler<Message<Integer>>() {
            @Override
            public void handle(Message<Integer> message) {
                Integer result = fib(message.body.intValue());
                JsonObject resultMessage = new JsonObject()
                        .putString("number", message.body.toString())
                        .putString("result", result.toString());
                eventBus.send("fib.response", resultMessage);
            }
        });
    }

    private int fib(int number) {
        if (number == 0) {
            return 0;
        } else if (number == 1) {
            return 1;
        } else {
            return fib(number - 2) + fib(number - 1);
        }
    }
}
```

```java
public class WorkerExample extends Verticle {
    @Override
    public void start() throws Exception {
        final EventBus eventBus = vertx.eventBus();

        Handler<Message<JsonObject>> resultHandler = new Handler<Message<JsonObject>>() {
            @Override
            public void handle(Message<JsonObject> message) {
                System.out.println("Fib("
                    +message.body.getString("number")+")="
                    +message.body.getString("result"));
            }
        };

        eventBus.registerHandler("fib.response", resultHandler);

        container.deployWorkerVerticle("worker.FibonacciWorker",
            new JsonObject(), 1, new Handler<String>() {
            @Override
            public void handle(String s) {
                eventBus.send("fib.request", 20);
            }
        });
    }
}
```

Problem/Solution Summary

- Node.js compels the use of JavaScript
- Vert.x is Polyglot
- Node.js is inherently single-threaded
- Vert.x leverages the multi-threaded JVM

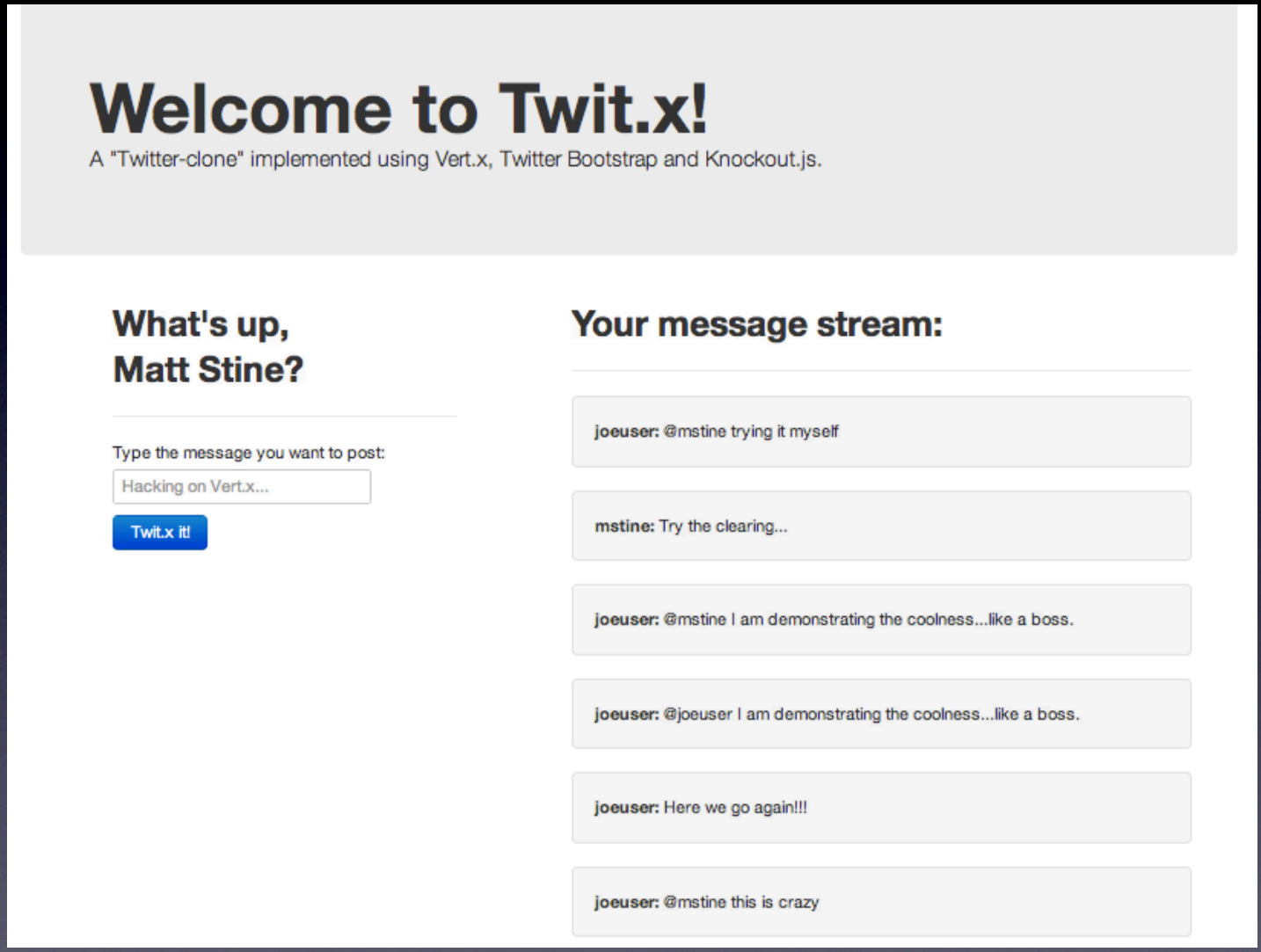
Problem/Solution Summary

- Node.js doesn’t help much w/ interprocess communication
- Vert.x features a distributed event bus which reaches all the way into the browser
- Node.js requires all code to run on the event loop
- Vert.x features background workers that allow blocking work to be done off of the event loops

Other Goodies

- Growing Module Repository
  - web server
  - persistors (Mongo, JDBC, ...)
  - work queue
  - authentication manager
  - session manager
  - Socket.IO
- TCP/SSL servers/clients
- HTTP/HTTPS servers/ clients
- WebSockets support
- SockJS support
- Timers
- Buffers
- Streams and Pumps
- Routing
- Asynchronous File I/O

Case Study



https://github.com/mstine/twitx

Get Involved!

- Google Group: http://groups.google.com/group/vertx
- GitHub: https://github.com/vert-x/vert.x
- IRC: irc://freenode.net/vertx
- Needs:
  - language implementations
  - modules (esp. persistence, security, ...)
  - examples/blogs/documentation help

Vert.x:

This ain't your Dad's Node!

Please fill out an evaluation:

http://speakerrate.com/talks/16821

Matt Stine
Enterprise Java/Cloud Consultant
matt.stine@gmail.com
http://mattstine.com
Twitter: @mstine

Image Credits

- Tim Fox: https://twitter.com/timfox
- BSOD Phone: http://www.flickr.com/photos/markhillary/3413357033
- V8: http://www.flickr.com/photos/gorbould/3479298062
- Ryan Dahl: http://www.flickr.com/photos/franksvalli/5163205409
- Shortcomings: http://www.flickr.com/photos/hendricksphotos/3340896056
- Brendan Eich: http://www.flickr.com/photos/equanimity/4055148344
- Neal Ford: http://nealford.com/images/Neal-2012-headshot.jpg
- Lars Bak: http://www.flickr.com/photos/niallkennedy/2822229099
- James Gosling: http://www.flickr.com/photos/skrb/2499736195
- Tony Hoare: http://www.flickr.com/photos/adewale_oshineye/3687557065
- Alan Kay: http://www.flickr.com/photos/mwichary/3010026816
- Rich Hickey: http://www.flickr.com/photos/hlship/3603090614
- Douglas Schmidt: http://www.cs.wustl.edu/~schmidt/gifs/SchmidtD.jpg
- Carl Hewitt: http://www.flickr.com/photos/jeanbaptisteparis/3098594954