nodeJS

THIS PRESENTATION CODE IS ON
HTTPS://GITHUB.COM/GABRIELELANA/NODE-EXAMPLES

CleanCode

GABRIELE LANA
GABRIELE.LANA@CLEANCODE.IT
TWITTER: @GABRIELELANA

THIS PRESENTATION CODE IS ON
HTTPS://GITHUB.COM/GABRIELELANA/NODE-EXAMPLES

nodeJS

WHY NODE.JS?

"NODE'S GOAL IS TO PROVIDE AN EASY WAY TO BUILD SCALABLE NETWORK PROGRAMS"

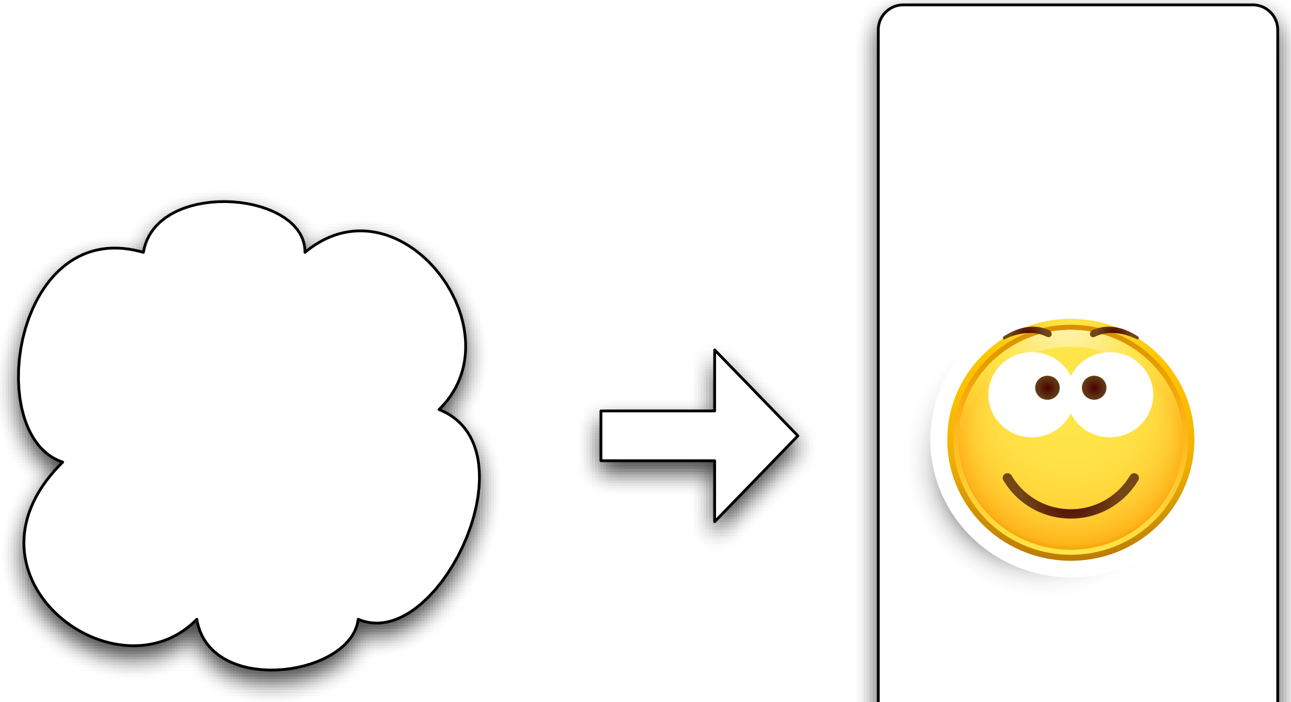
HTTP://NODEJS.ORG/#ABOUT

nodeJS

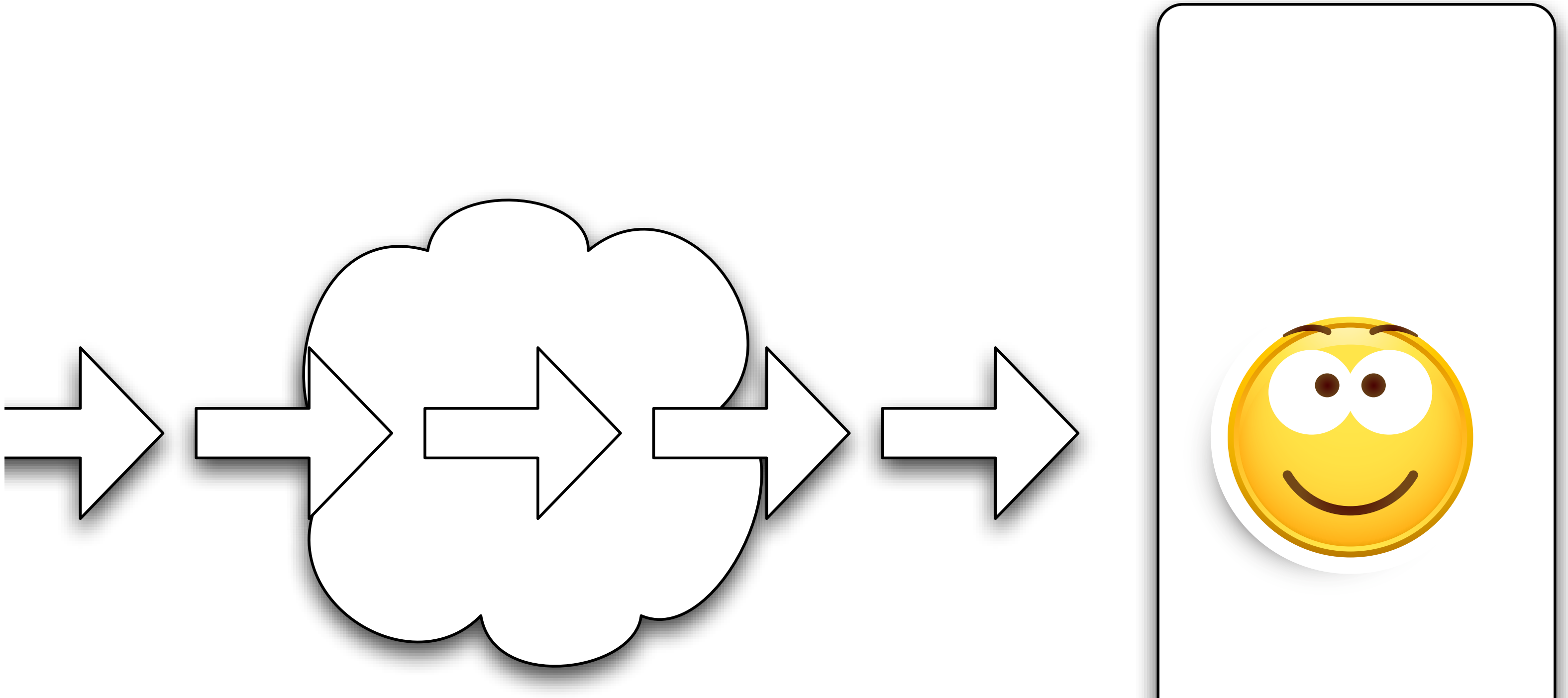
WHAT IS NODE.JS?

- ASYNCHRONOUS I/O FRAMEWORK
- CORE IN C++ ON TOP OF V8
- REST OF IT IN JAVASCRIPT
- SWISS ARMY KNIFE FOR NETWORK RELATED STUFFS
- CAN HANDLE THOUSANDS OF CONCURRENT CONNECTIONS WITH MINIMAL OVERHEAD (CPU/MEMORY) ON A SINGLE PROCESS

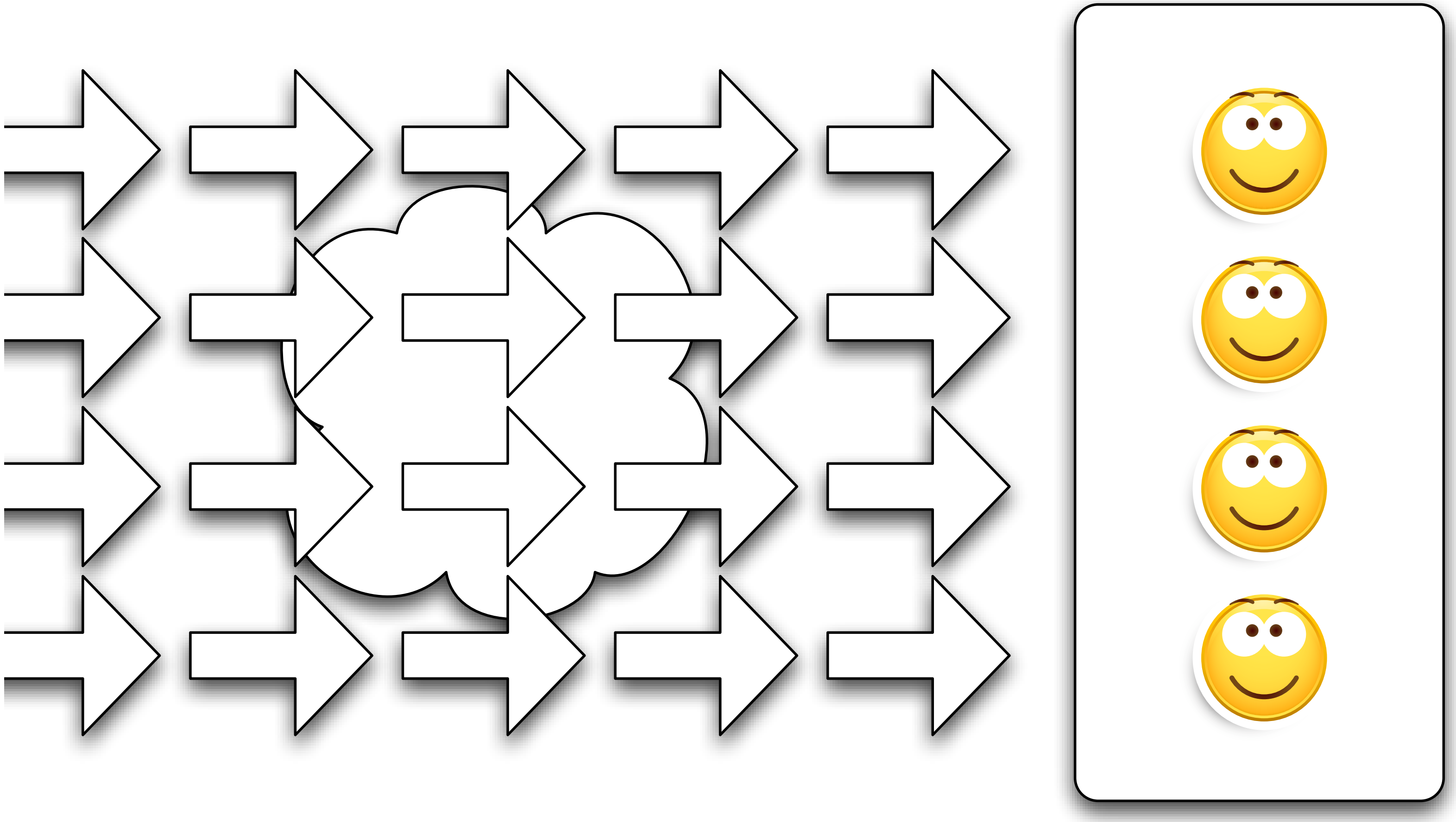
SINGLE THREAD SYNCHRONOUS I/O



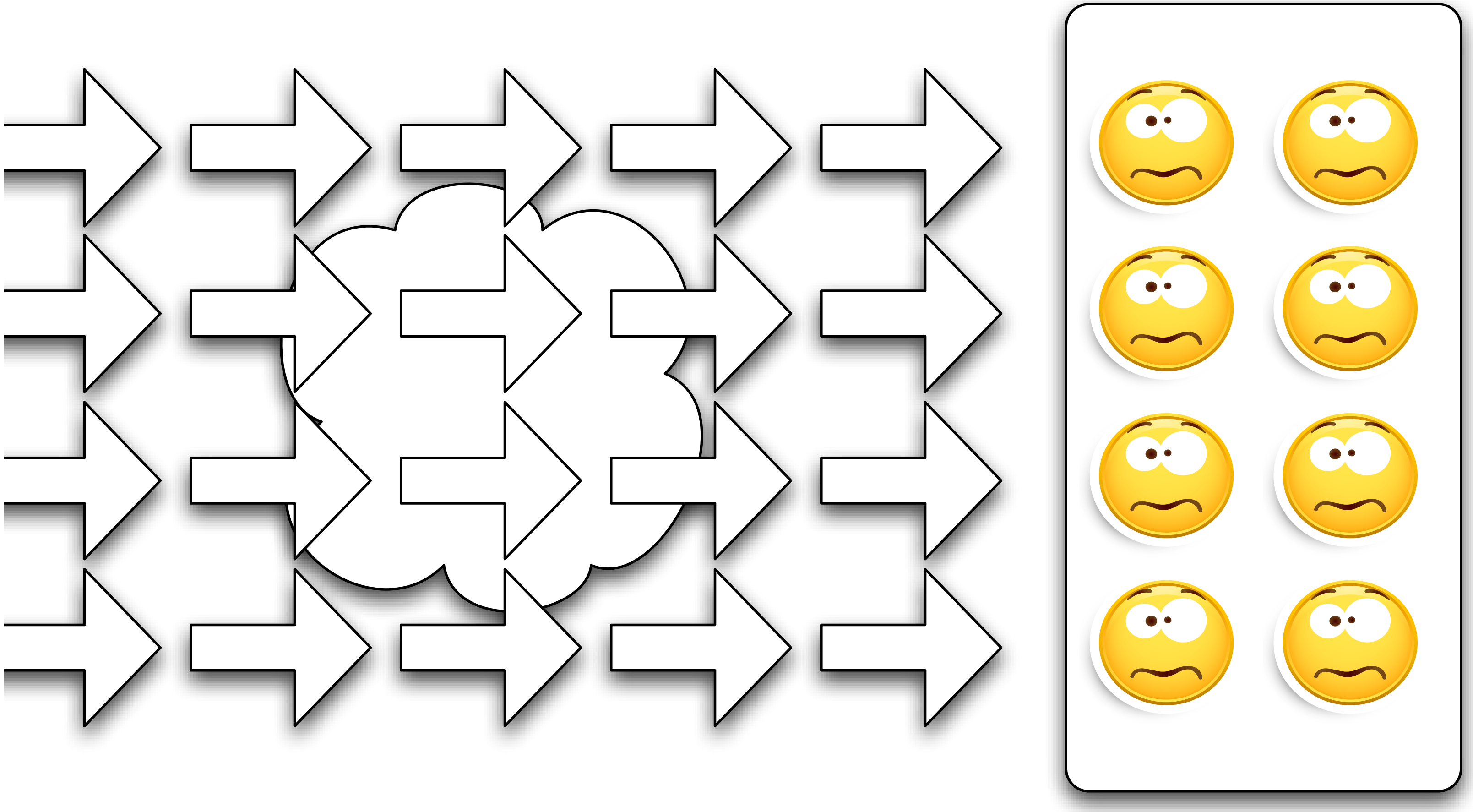
SINGLE THREAD
SYNCHRONOUS I/O



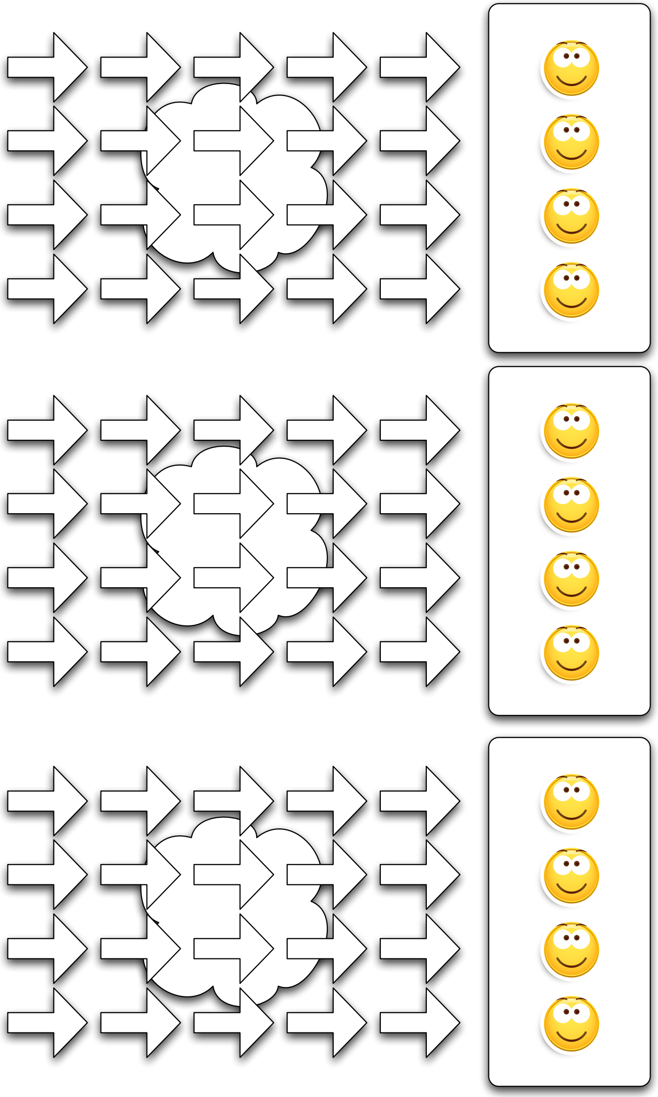
MULTIPLE THREAD SYNCHRONOUS I/O



MULTIPLE THREAD SYNCHRONOUS I/O



YOU CAN "ALWAYS" SCALE WITH MULTIPLE MACHINES BUT IT COSTS YOU $$$



BUT...
WHAT IS HE
DOING?

CPU BOUND TASKS?

BUT...
WHAT IS HE DOING?

CPU BOUND TASKS?

...OR I/O BOUND TASKS?

BUT... WHAT IS HE DOING?

SYNCHRONOUS I/O

```
function productsInCart(request, response) {
    var db = new Db()
    var user = new User(request)
    if (user.isAuthorized("cart/products")) {
        response.write(
            JSON.stringify(
                db.productsInCart(user.cartId())
            )
        )
    } else {
        response.unauthorized()
    }
}
```

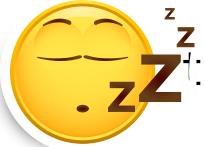
SYNCHRONOUS I/O

```
function productsInCar😴t, response) {
    var db = new Db()
    var user = new User(request)
    if (user.isAuthorized("cart/products")) {
        response.write(
            JSON.stringify(
                db.productsInCart(user.cartId())
            )
        )
    } else {
        response.unauthorized()
    }
}
```

SYNCHRONOUS I/O

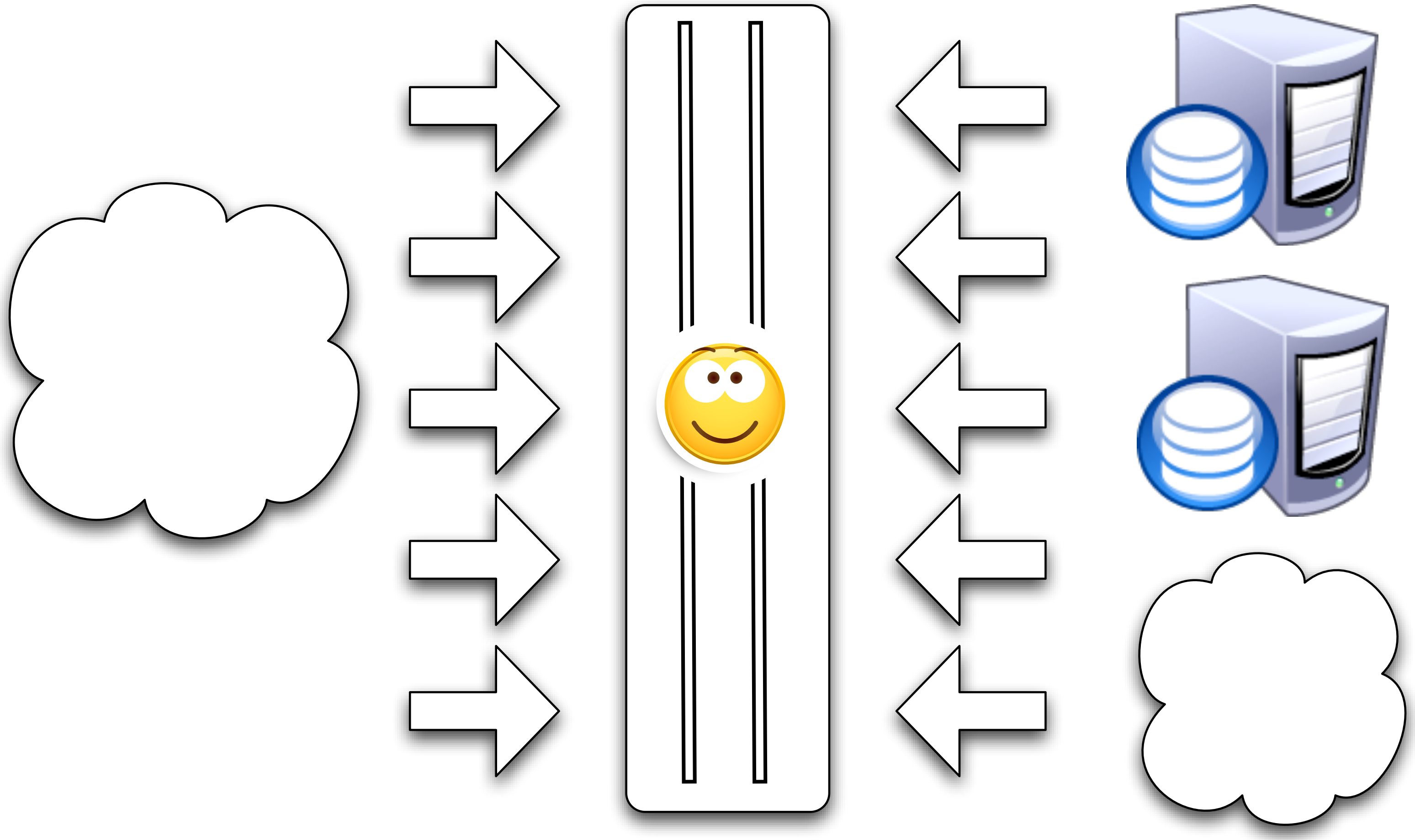
```
function productsInCar              se) {
    var db = new Db()
    var user = new User(request)
    if (user.isAuthorized("cart/products")) {
        response.write(
            JSON.stringify(
                db.productsInCart(user.cartId())
            )
        )
    } else {
        response.unauthorized()
    }
}
```

SYNCHRONOUS I/O

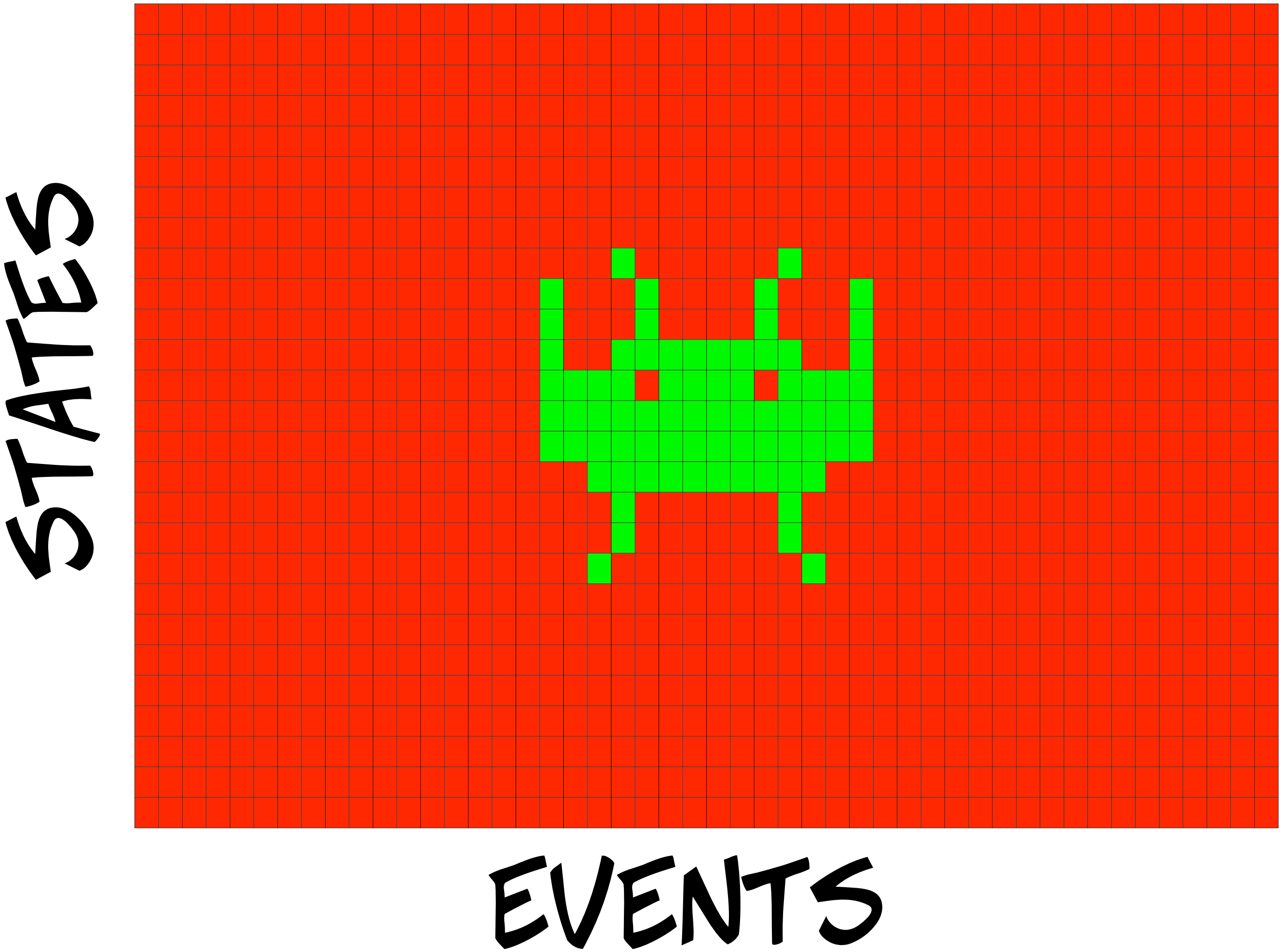


```
function productsInCar           se) {
    var db = new Db()
    var user = new User(request)
    if (user.isAuthorized("        oducts")) {
        response.write(
            JSON.stringify(
                db.productsInCart(user.cartId())
            )
        )
    } else {
        response.unauthorized()
    }
}
```

SINGLE THREAD ASYNCHRONOUS I/O



SINGLE SOURCE EVENTS



HTTP://WWW.INFOQ.COM/PRESENTATIONS/DEATH-BY-ACCIDENTAL-COMPLEXITY

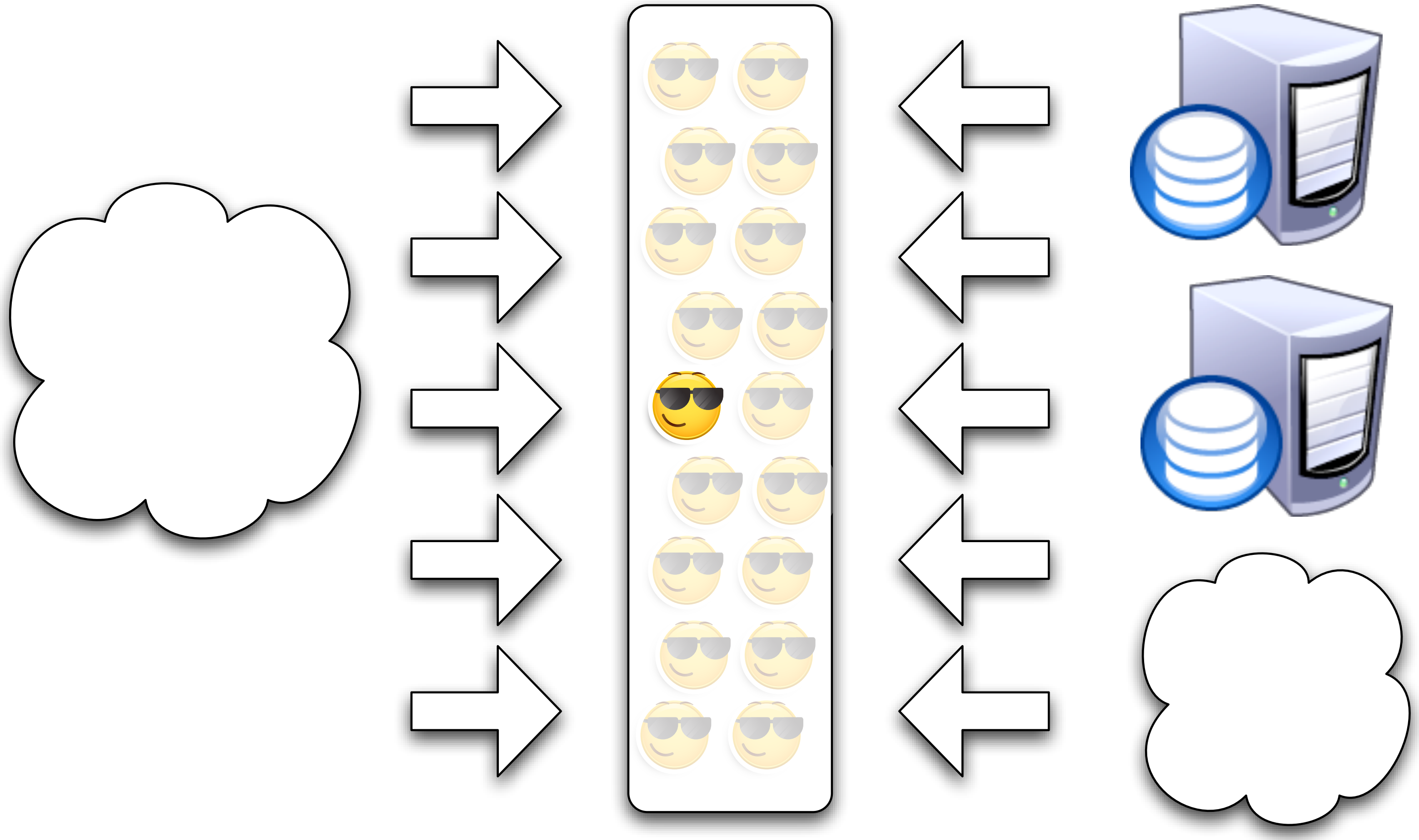
SINGLE SOURCE EVENTS

```
function productsInCart(request, response) {
    var db = null, user = null, ...
    createDb()
    handle(function(source, event) {
        if (event["name"] === "createDb") {
            if (db === null) {
                db = event.data
                createUser(request)
            } else {
                ????
            }
        } else if (event["name"] === "createUser") {
            if (user === null) {
                user = event.data
                ...
            } else {
                ???
            }
        ...
        } else {
            source.push(event, state)
        }
    }, "_initial")
}
```

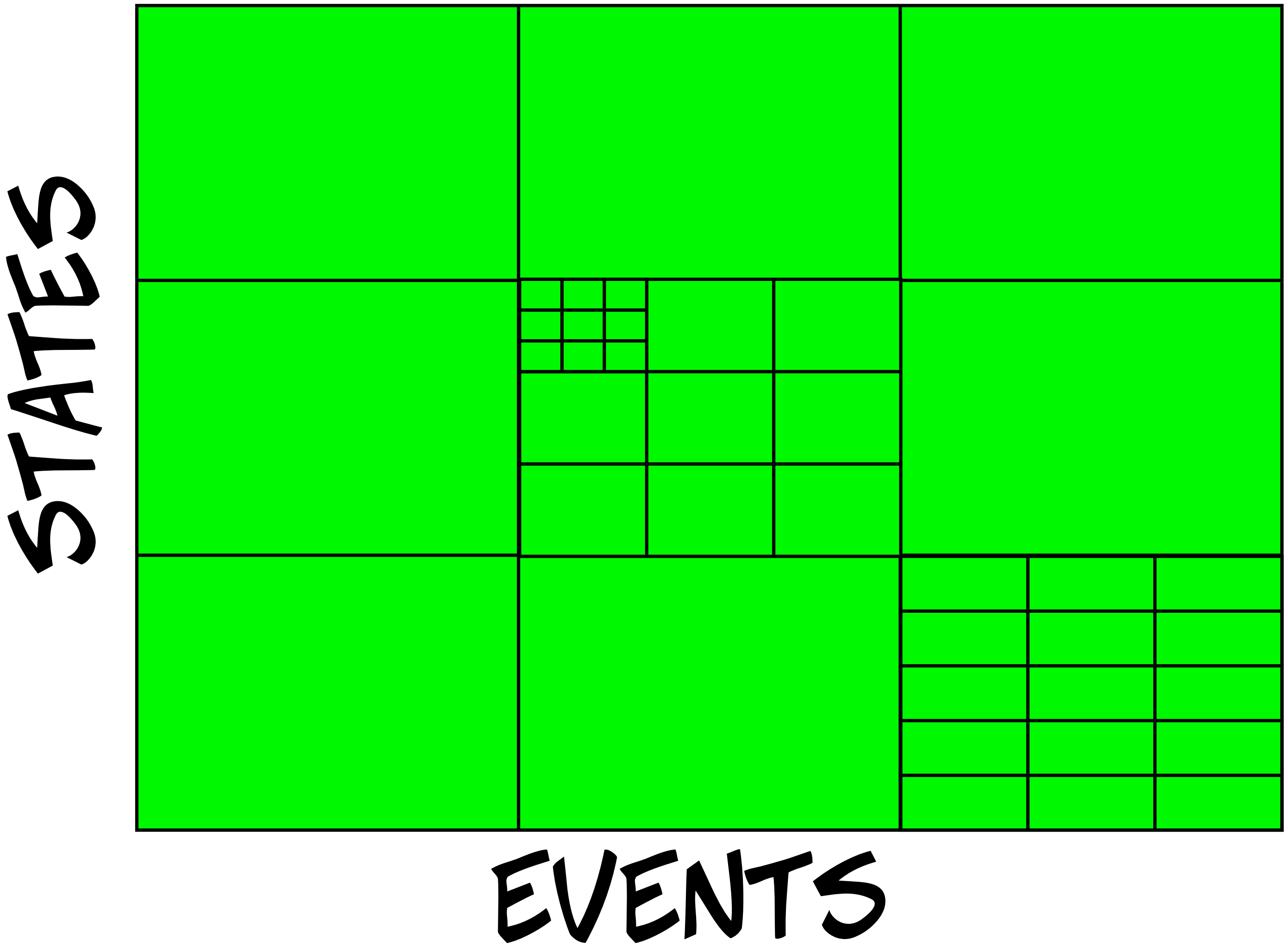
SINGLE THREAD ASYNCHRONOUS I/O

I AM "CALLBACK" CALL ME IF YOU NEED ME...

SINGLE THREAD ASYNCHRONOUS I/O



MULTIPLE SOURCE EVENTS (LOCAL STATE)



MULTIPLE SOURCE EVENTS (LOCAL STATE)

```
function productsInCart(request, response) {
    createDb(function(db) {
        createUser(function(user) {
            if (user.isAuthorized("cart/products") {
                response.write(
                    JSON.stringify(
                        db.productsInCart(user.cartId())
                    )
                )
                response.end()
            }) else {
                response.unauthorized()
            }
        })
    })
}
```

MULTIPLE SOURCE EVENTS (LOCAL STATE)

```
function productsInCart(request, response) {
    createDb(function(db) {
        createUser(function(user) {
            if (user.isAuthorized("cart/products") {
                response.write(
                    JSON.stringify(
                        db.productsInCart(user.cartId())
                    )
                )
                response.end()
            }) else {
                response.unauthorized()
            }
        })
    })
}
```

EVENT EMITTER (LOCAL STATE)

```
var http = require("http")

var server = http.createServer(function(request, response) {
  response.writeHead(200, {
    "Content-Type": "plain/text"
  })
  response.write("Hello World\n")
  response.end()
})

server.listen(8080)

console.log("> SERVER STARTED")
```

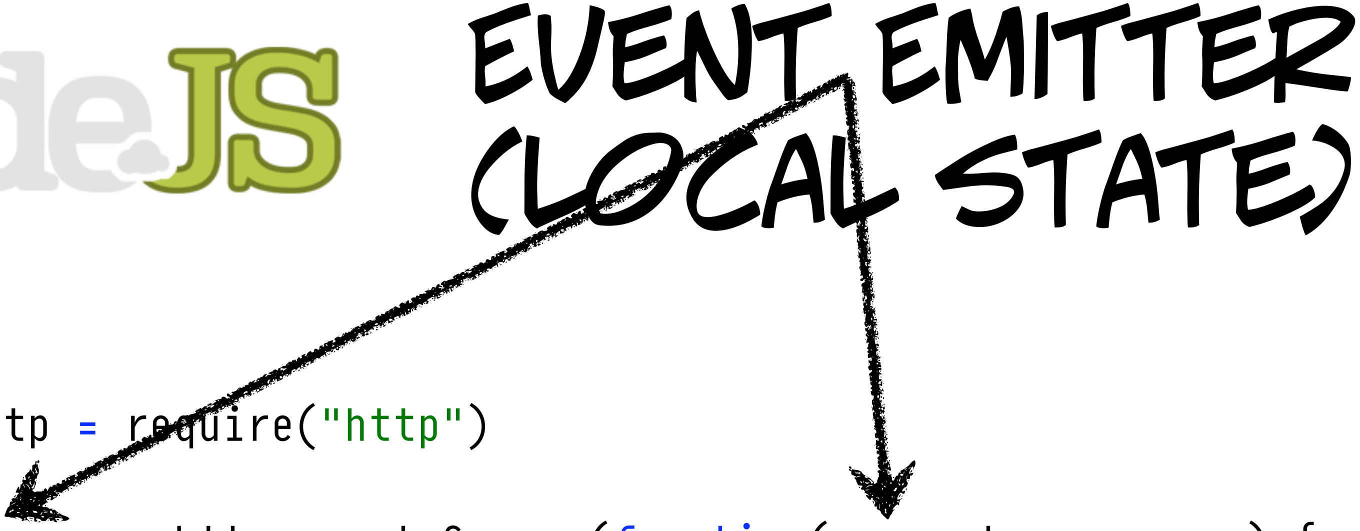
01#HELLO_WORLD/HELLO_WORLD_SERVER.JS

nodeJS

EVENT EMITTER (LOCAL STATE)

```
var http = require("http")

var server = http.createServer(function(request, response) {
  response.writeHead(200, {
    "Content-Type": "plain/text"
  })
  response.write("Hello World\n")
  response.end()
})

server.listen(8080)

console.log("> SERVER STARTED")
```

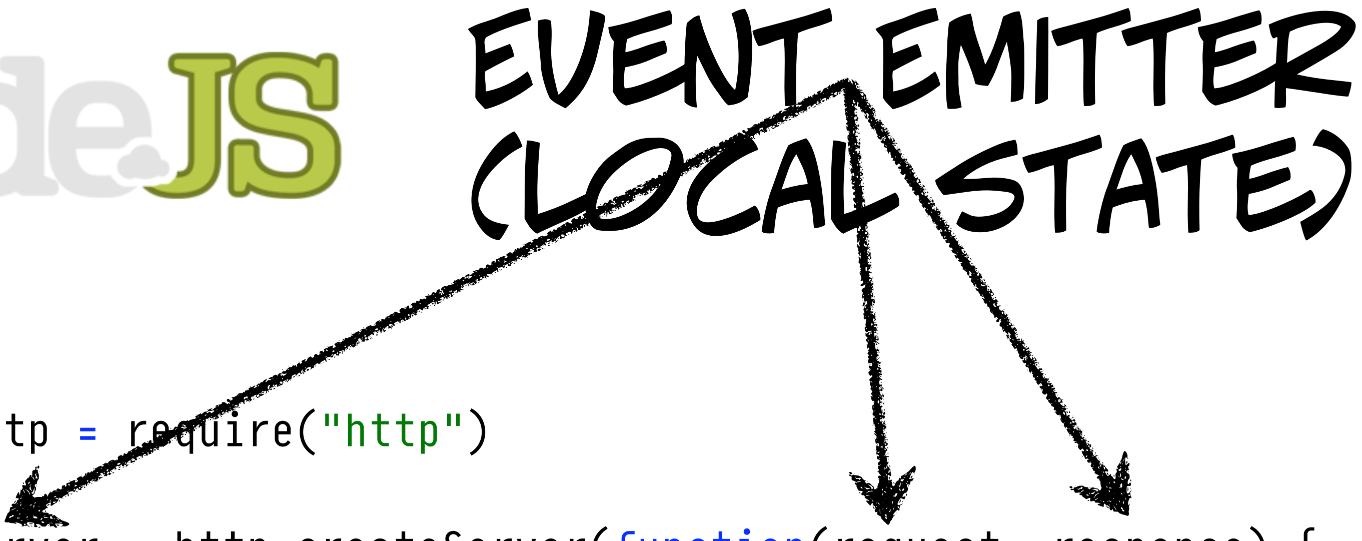
01#HELLO_WORLD/HELLO_WORLD_SERVER.JS



EVENT EMITTER (LOCAL STATE)

```
var http = require("http")

var server = http.createServer(function(request, response) {
  response.writeHead(200, {
    "Content-Type": "plain/text"
  })
  response.write("Hello World\n")
  response.end()
})

server.listen(8080)

console.log("> SERVER STARTED")
```

01#HELLO_WORLD/HELLO_WORLD_SERVER.JS



EVENT EMITTER (LOCAL STATE)

```
var http = require("http")

var server = http.createServer(function(request, response) {
  response.writeHead(200, {
    "Content-Type": "plain/text"
  })
  response.write("Hello World\n")
  response.end()
})

server.listen(8080)

console.log("> SERVER STARTED")
```

01#HELLO_WORLD/HELLO_WORLD_SERVER.JS

EVENT EMITTER (LOCAL STATE)

```
coder@apollo:~/Work/src/node/examples$ node hello_world_server.js
> SERVER STARTED
```

#1

```
coder@apollo:~$ curl "http://localhost:8080/"
Hello World
```

#2

EVENT EMITTER (LOCAL STATE)

```
var server = require("http").createServer()

server.on("request", function(request, response) {
  console.log("> REQUEST STARTED")
  request.on("end", function() {
    console.log("> REQUEST CLOSED")
    response.writeHead(200, {
      "Content-Type": "plain/text"
    })
    response.end("Hello World\n")
    server.close()
  })
  response.on("close", function() {
    console.log("> RESPONSE CLOSED")
  })
})
```

01#HELLO_WORLD/HELLO_WORLD_SERVER_EMITTER.JS

EVENT EMITTER (LOCAL STATE)

```
...

server.on("close", function() {
  console.log("> SERVER CLOSED")
})

server.on("listening", function() {
  console.log("> SERVER STARTED")
})

server.listen(8080)
```

01#HELLO_WORLD/HELLO_WORLD_SERVER_EMITTER.JS

EVENT EMITTER (LOCAL STATE)

```
coder@apollo:~/Work/src/node/examples$ node hello_world_server.js
> SERVER STARTED
```

#1

```
coder@apollo:~$ curl "http://localhost:8080/"
Hello World
```

#2

```
> REQUEST STARTED
> REQUEST CLOSED
> SERVER CLOSED
```

#1

nodeJS

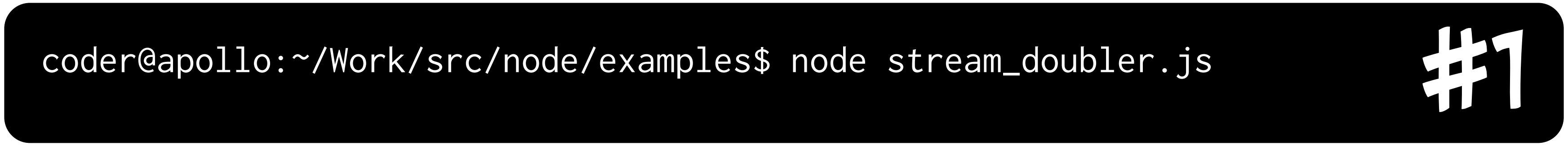
WHY SO COMPLICATED?

nodeJS DATA STREAMS

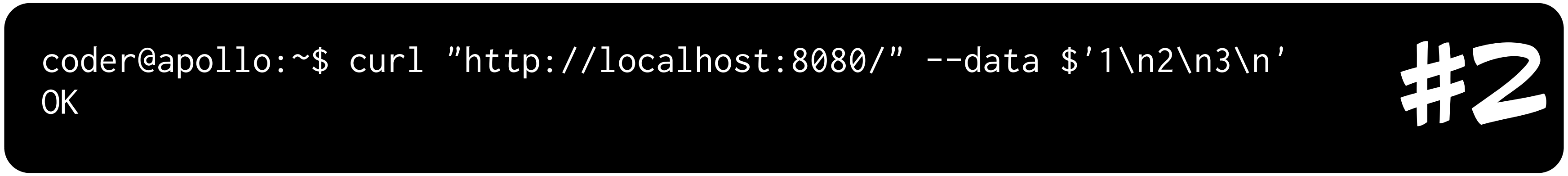
```
server.on("request", function(request, response) {
  var chunks = [],
      output = fs.createWriteStream("./output")

  request.on("data", function(chunk) {
    chunks = forEachLine(chunks.concat(chunk), function(line) {
      output.write(parseInt(line, 10) * 2)
      output.write("\n")
    })
  })

  request.on("end", function() {
    response.writeHead(200, { "Content-Type": "plain/text" })
    response.end("OK\n")
    output.end()
    server.close()
  })
})
```

02#PROXY_STREAM/PROXY_STREAM.JS

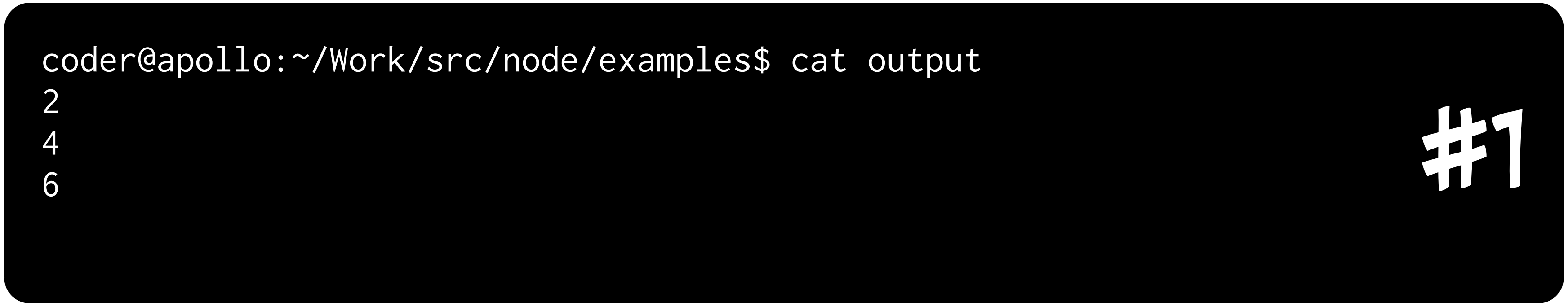
EVENT EMITTER (LOCAL STATE)



```
coder@apollo:~/Work/src/node/examples$ node stream_doubler.js
```



```
coder@apollo:~$ curl "http://localhost:8080/" --data $'1\n2\n3\n'
OK
```



```
coder@apollo:~/Work/src/node/examples$ cat output
2
4
6
```

nodeJS

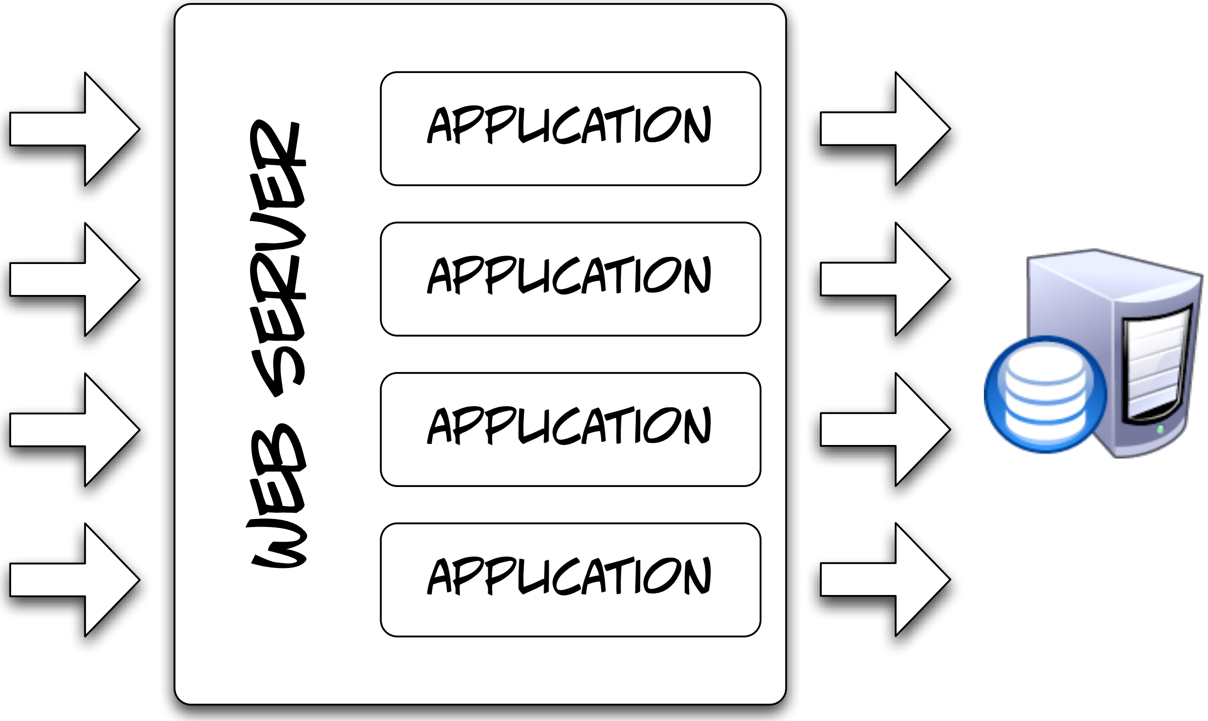
WHY JAVASCRIPT?

- FRIENDLY CALLBACKS
- UBIQUITOUS ~~(WELL KNOWN)~~
- NO I/O PRIMITIVES
- ONE LANGUAGE TO RULE THEM ALL

nodeJS

MIND SHIFT #1

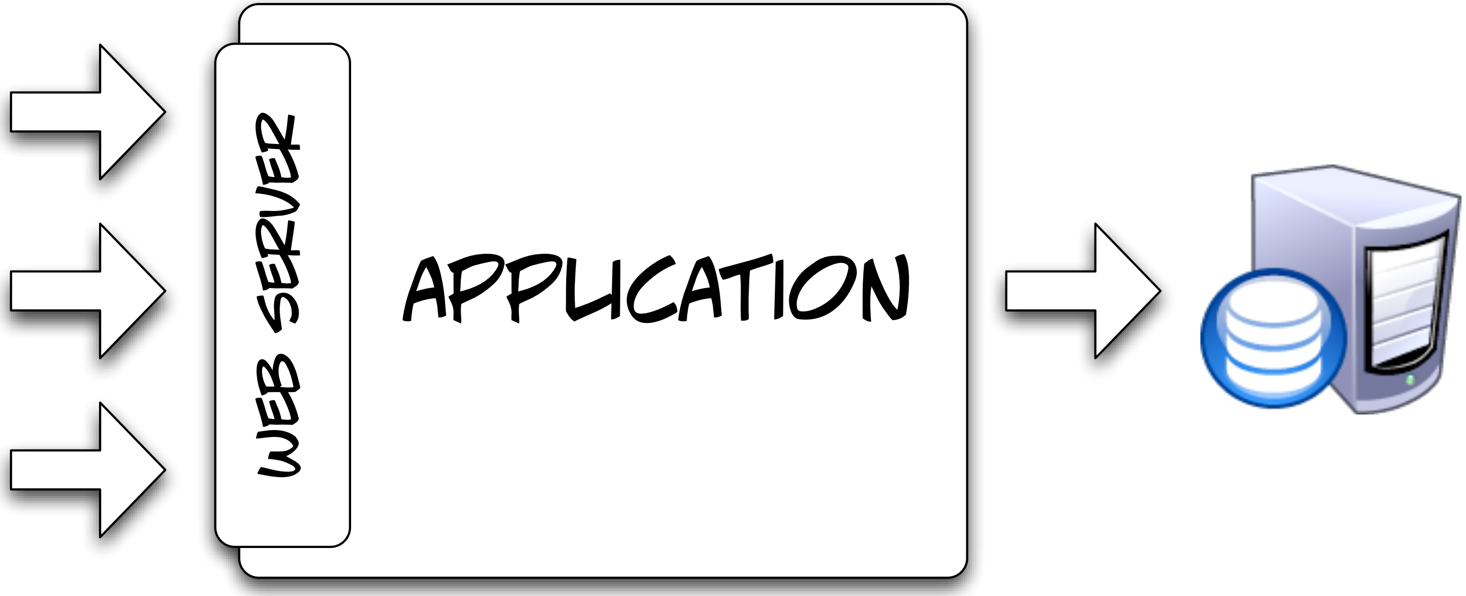
WEB APPLICATIONS BEFORE: A WEB SERVER WITH SOME APPLICATION LOGIC



nodeJS

MIND SHIFT #1

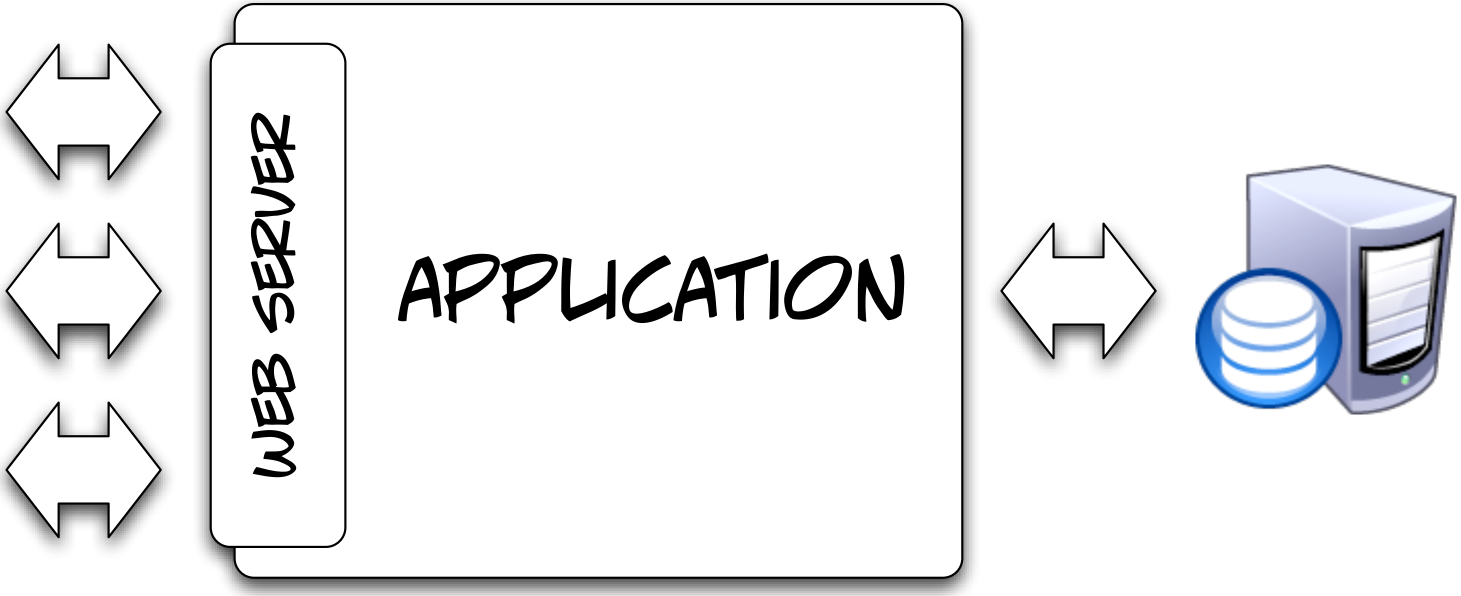
WEB APPLICATIONS AFTER: AN APPLICATION ACCESSIBLE OVER HTTP



nodeJS

MIND SHIFT #1

WEB APPLICATIONS
AFTER: AN APPLICATION
THAT CAN
COMMUNICATE AND
COLLABORATE WITH
THE WORLD

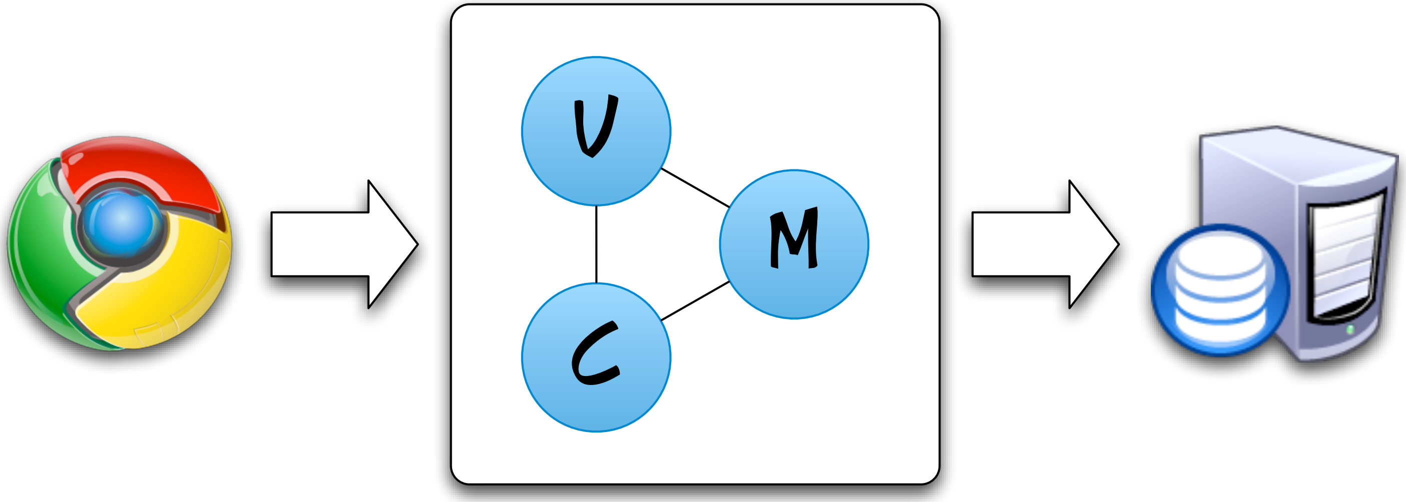


nodeJS

MIND SHIFT #2

WEB APPLICATIONS BEFORE: STATEFUL

- NO EASY TO SCALE
- NO EASY TO REUSE

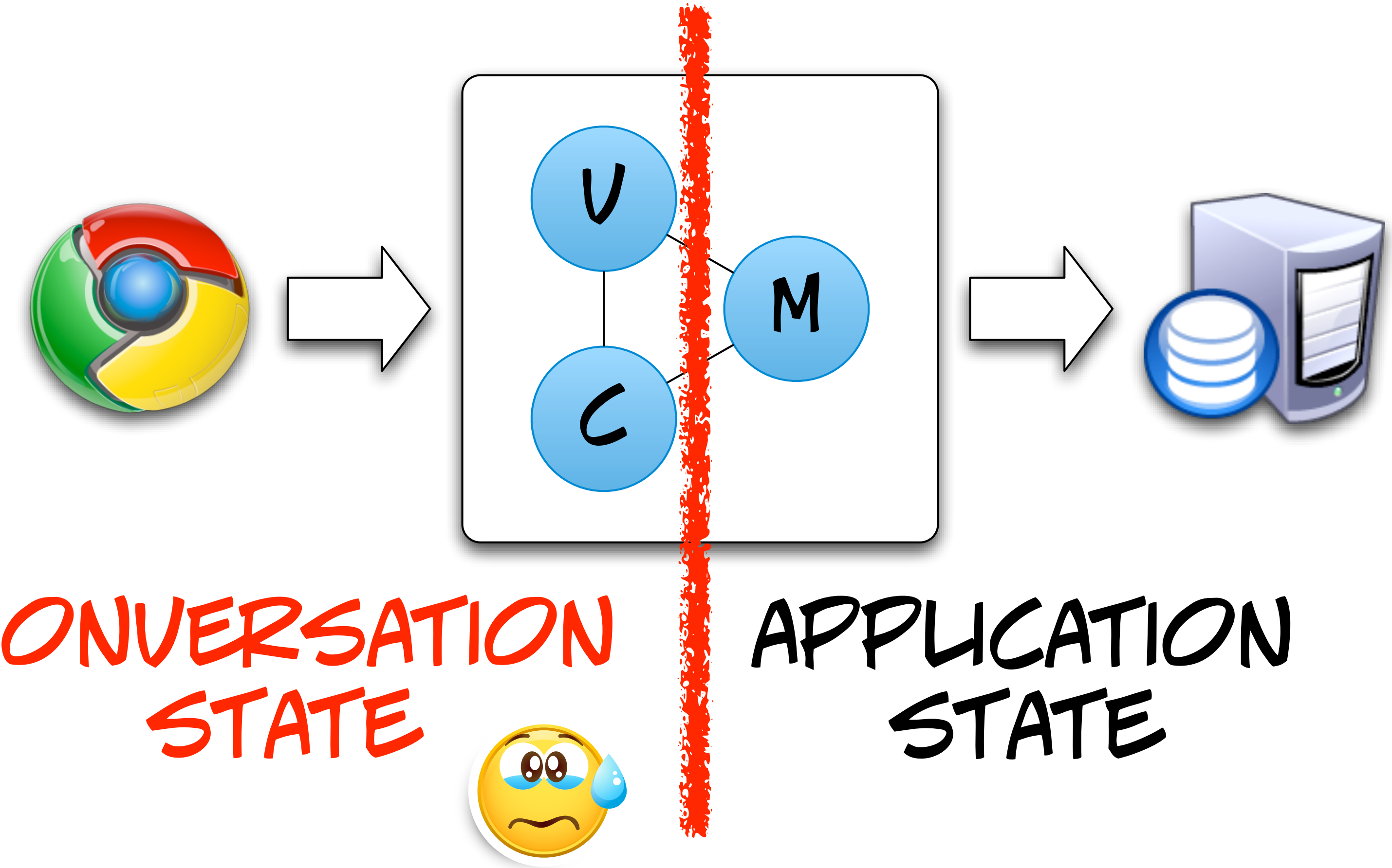


nodeJS

MIND SHIFT #2

WEB APPLICATIONS BEFORE: STATEFUL

- NO EASY TO SCALE
- NO EASY TO REUSE



CONVERSATION STATE

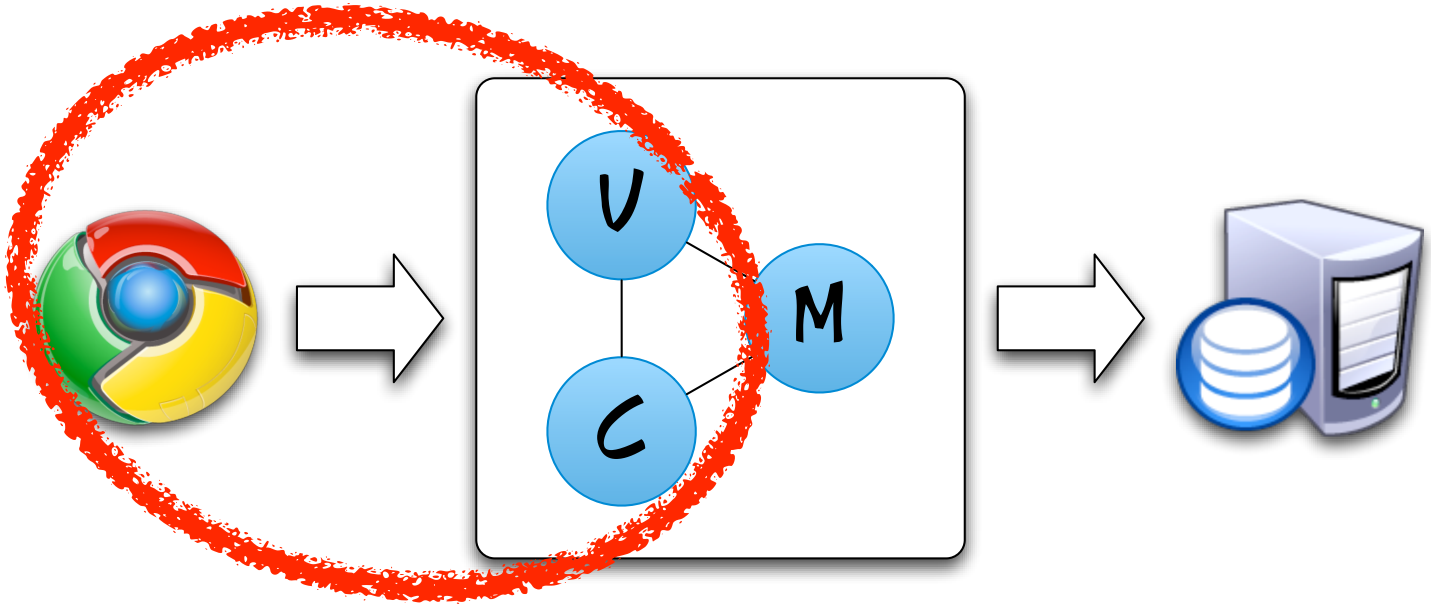
APPLICATION STATE

nodeJS

MIND SHIFT #2

WEB APPLICATIONS BEFORE: STATEFUL

- NO EASY TO SCALE
- NO EASY TO REUSE



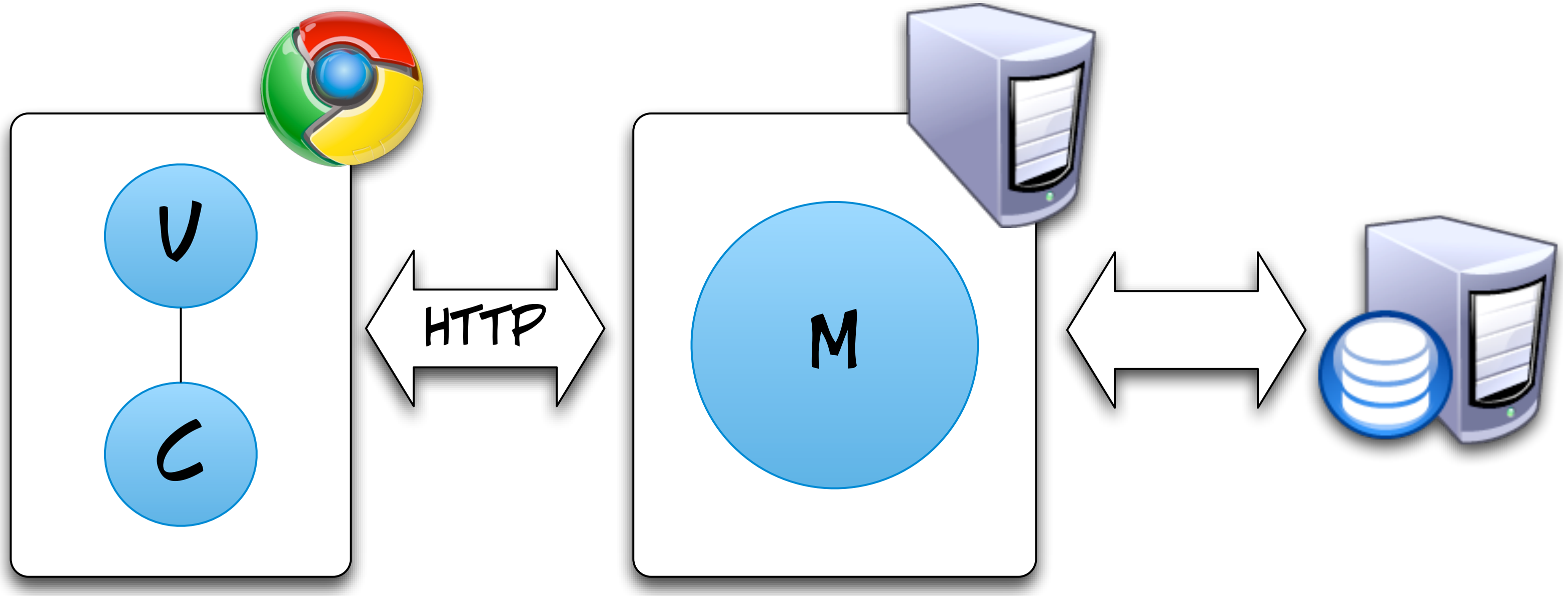
TIGHTLY COUPLED

nodeJS

MIND SHIFT #2

WEB APPLICATIONS AFTER: STATELESS

- EASY TO SCALE
- EASY TO REUSE

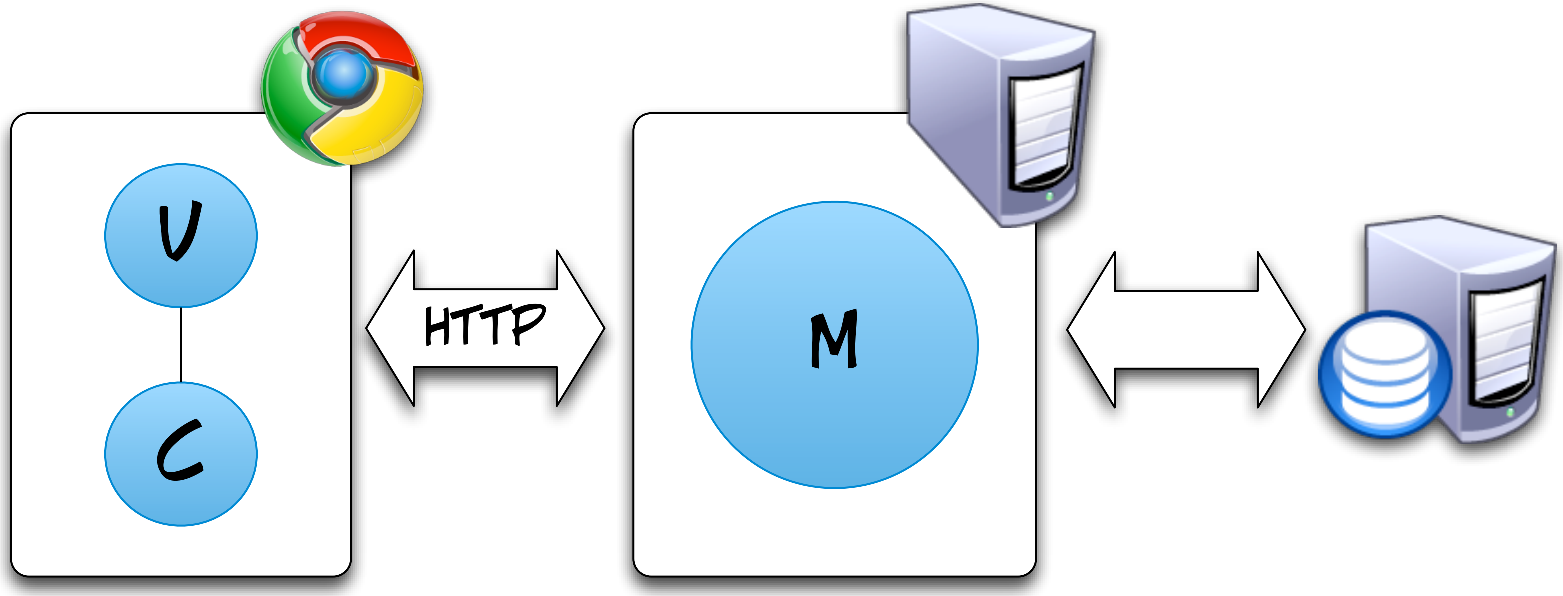


nodeJS

MIND SHIFT #2

WEB APPLICATIONS AFTER: STATELESS

- EASY TO SCALE
- EASY TO REUSE



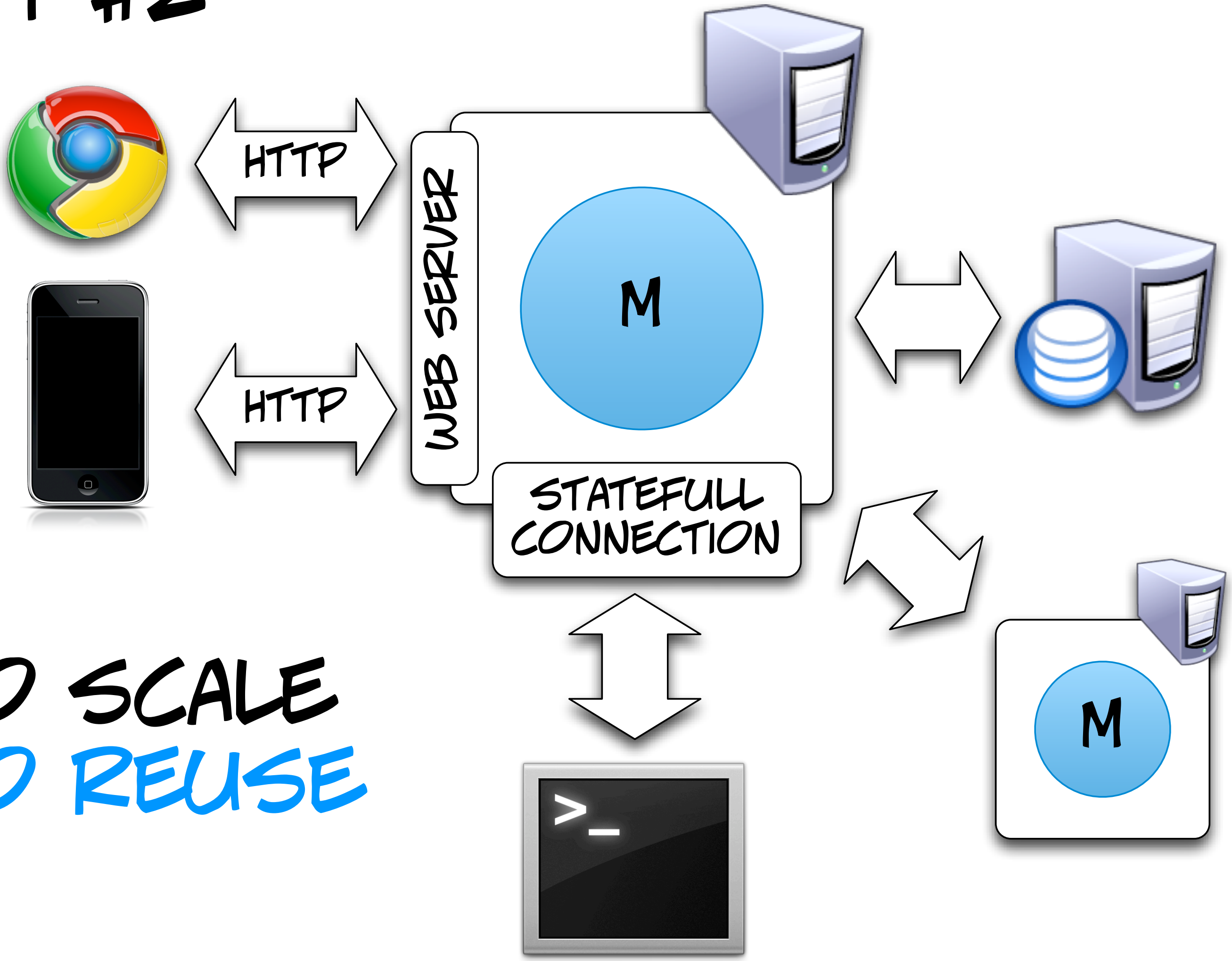
CONVERSATION STATE

APPLICATION STATE

nodeJS

MIND SHIFT #2

WEB APPLICATIONS AFTER: STATELESS



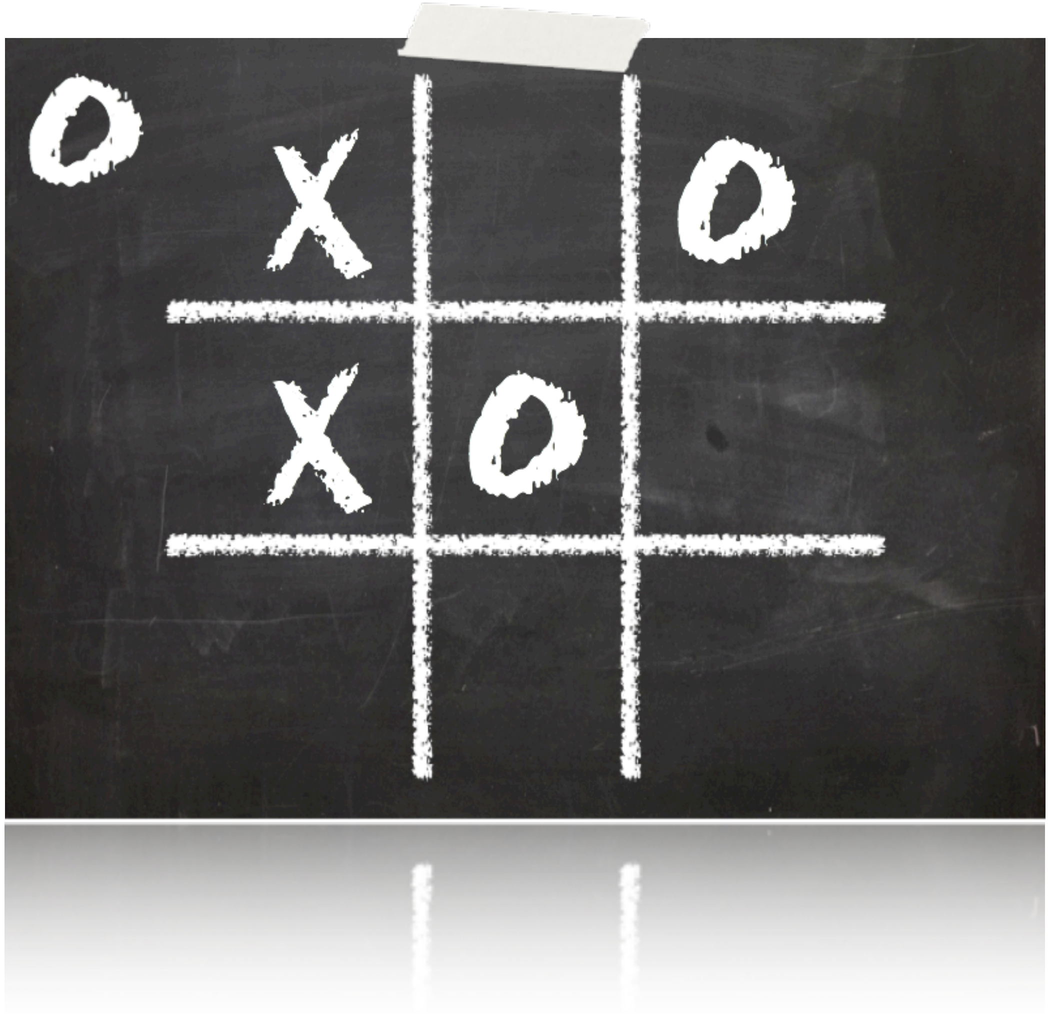
- EASY TO SCALE
- EASY TO REUSE

nodeJS

NO FLUFF
JUST STUFF

nodeJS

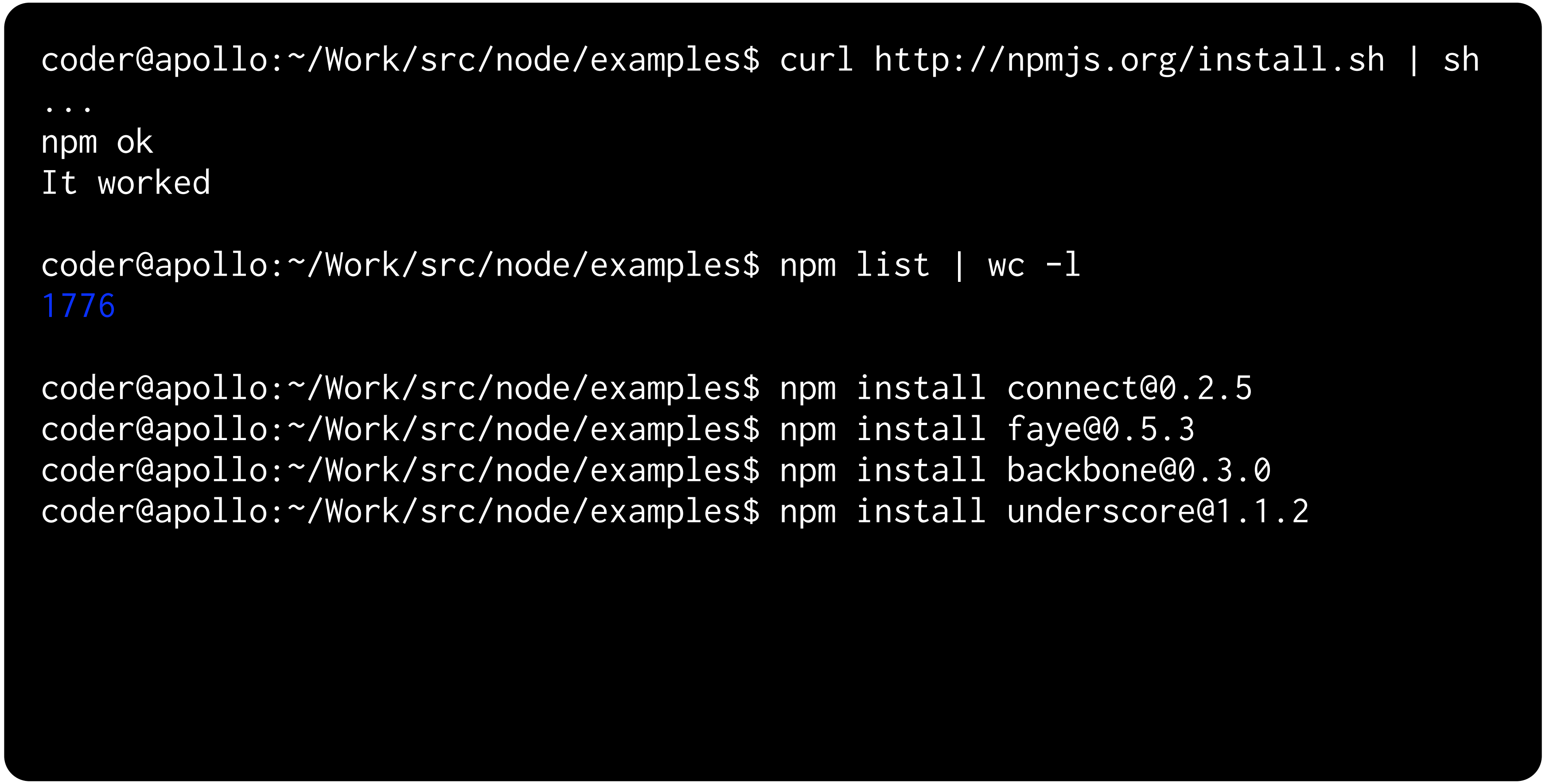
TIC - TAC - TOE



nodeJS Tic - Tac - Toe

Demo

INSTALL NPM
(NODE PACKET MANAGER)



```
coder@apollo:~/Work/src/node/examples$ curl http://npmjs.org/install.sh | sh
...
npm ok
It worked

coder@apollo:~/Work/src/node/examples$ npm list | wc -l
1776

coder@apollo:~/Work/src/node/examples$ npm install connect@0.2.5
coder@apollo:~/Work/src/node/examples$ npm install faye@0.5.3
coder@apollo:~/Work/src/node/examples$ npm install backbone@0.3.0
coder@apollo:~/Work/src/node/examples$ npm install underscore@1.1.2
```

nodeJS STATIC HANDLER

```
var server = connect.createServer(connect.logger({ "buffer": true }))
  .use("/", connect.router(function(resource) {
    resource.get("/board", function(request, response, next) {
      request.url = "/board.html"
      next()
    })
    ...
  }), connect.staticProvider({
    "root": path.join(__dirname, "static"),
    "cache": true
  }))

server.listen(port)
```

05#TICTACTOE/SERVER.JS

GAME HANDLER (GENERATE BOARD-ID)

```
resource.post("/board", function(request, response) {
  response.writeHead(200, { "Content-Type": "application/json" })
  uuid(function(boardId) {
    response.end(
      JSON.stringify({
        "board": { "id": boardId }
      })
    )
  })
})
```

05#TICTACTOE/SERVER.JS

GAME HANDLER
(INITIAL BOARD/USER)

```
resource.get("/board/:id", function(request, response) {
  var board = boards.get(request.params["id"])
  if (board === undefined) {
    board = new Board({ "id": request.params["id"] })
    boards.add(board)
  }
  uuid(function(userId) {
    var user = board.user(userId)
    response.writeHead(200, { "Content-Type": "application/json" })
    response.end(
      JSON.stringify({
        "board": board,
        "user": user
      })
    )
  })
})
```

05#TICTACTOE/SERVER.JS

GAME HANDLER
(MAKE YOUR MOVE)

```
resource.post("/board/:id", function(request, response) {
  waitForBody(request, function(body) {
    boards.get(request.params["id"]).move(JSON.parse(body))
    response.writeHead(204, { "Content-Type": "application/json" })
    response.end(JSON.stringify({ "response": "ok" }))
  })
})
```

05#TICTACTOE/SERVER.JS

nodeJS COMET HANDLER

```
var comet = new Faye.NodeAdapter({ "mount": "/comet", "timeout": 50 })

var server = connect.createServer(connect.logger({ "buffer": true }))
  .use("/comet", function(request, response, next) {
    comet.handle(request, response)
  })
  ...
})

comet.attach(server)
```

05#TICTACTOE/SERVER.JS

nodeJS

COMET EVENTS ON BACKBONE EVENTS

```
var client = comet.getClient()
var boards = new Backbone.Collection

boards.bind("change", function(board) {
  client.publish("/board-" + board.get("id"), board)
})
```

05#TICTACTOE/SERVER.JS

nodeJS

IN BROWSER
ROUTING

```
$(function() {

  $.sammy(function() {

    this.get("", function(context) {
      $.post("/board", function(response) {
        context.redirect("#/board/" + response["board"]["id"])
      })
    })

    ...
  }).run()
})
```

05#TICTACTOE/STATIC/BOARD.HTML

IN BROWSER
ROUTING/START GAME

```
var comet = new Faye.Client("/comet")
var game = new Game()

this.get("#/board/:id", function(context) {
  game.start()
  $.get("/board/" + context.params["id"], function(response) {
    game.set({ "me": new User(response.user) })
    game.set({ "board": new Board(response.board) })
    comet.connect()
    comet.subscribe("/board-" + context.params["id"], function(board) {
      game.get("board").set(board)
    })
  })
})
```

05#TICTACTOE/STATIC/BOARD.HTML

IN BROWSER
GAME LOGIC EXAMPLE

```
window.Game = Backbone.Model.extend({
  "initialize": function() {
    ...
    game.get("board").bind("change", function() {
      if (this.isMyTurn()) {
        return game.trigger("make-your-move")
      }
      return game.trigger("wait-for-move")
    })
  }
})
```

05#TICTACTOE/STATIC/JS/APPLICATION.JS

nodeJS

IN BROWSER
GAME LOGIC EXAMPLE

```
game.bind("play-with-board", function(cells) {
  buildBoard(['_'].concat(cells))
})

game.bind("play-with-mark", function(mark) {
  showPlayerMarker(cellMarksUrl[myMark = mark])
})

game.bind("make-your-move", function() {
  $("#board").undelegate()
  $("#board").delegate("*[id^=cell]", "mouseover", function() {
    $(this).data("cell").select()
  })
  $("#board").delegate("*[id^=cell]", "mouseout", function() {
    $(this).data("cell").unselect()
  })
  ...
```

05#TICTACTOE/STATIC/JS/APPLICATION.JS

nodeJS

WHAT ABOUT CPU BOUND TASKS?

THE FORK BE WITH YOU

```
#!/bin/bash

for count in `seq 1 100`; do
    echo $count
    sleep 0.1
done
```

03#LONG_RUNNING_JOBS/LONG_RUNNING_JOB.SH

THE FORK BE WITH YOU

```
var spawn = require("child_process").spawn,
    server = require("http").createServer()

server.on("request", function(request, response) {
  var job = spawn("./long_running_job.sh")

  job.stdout.on("data", function(tick) {
    response.write(tick)
  })

  job.on("exit", function() {
    response.end()
  })
})
```

03#LONG_RUNNING_JOBS/LONG_RUNNING_SERVER.JS

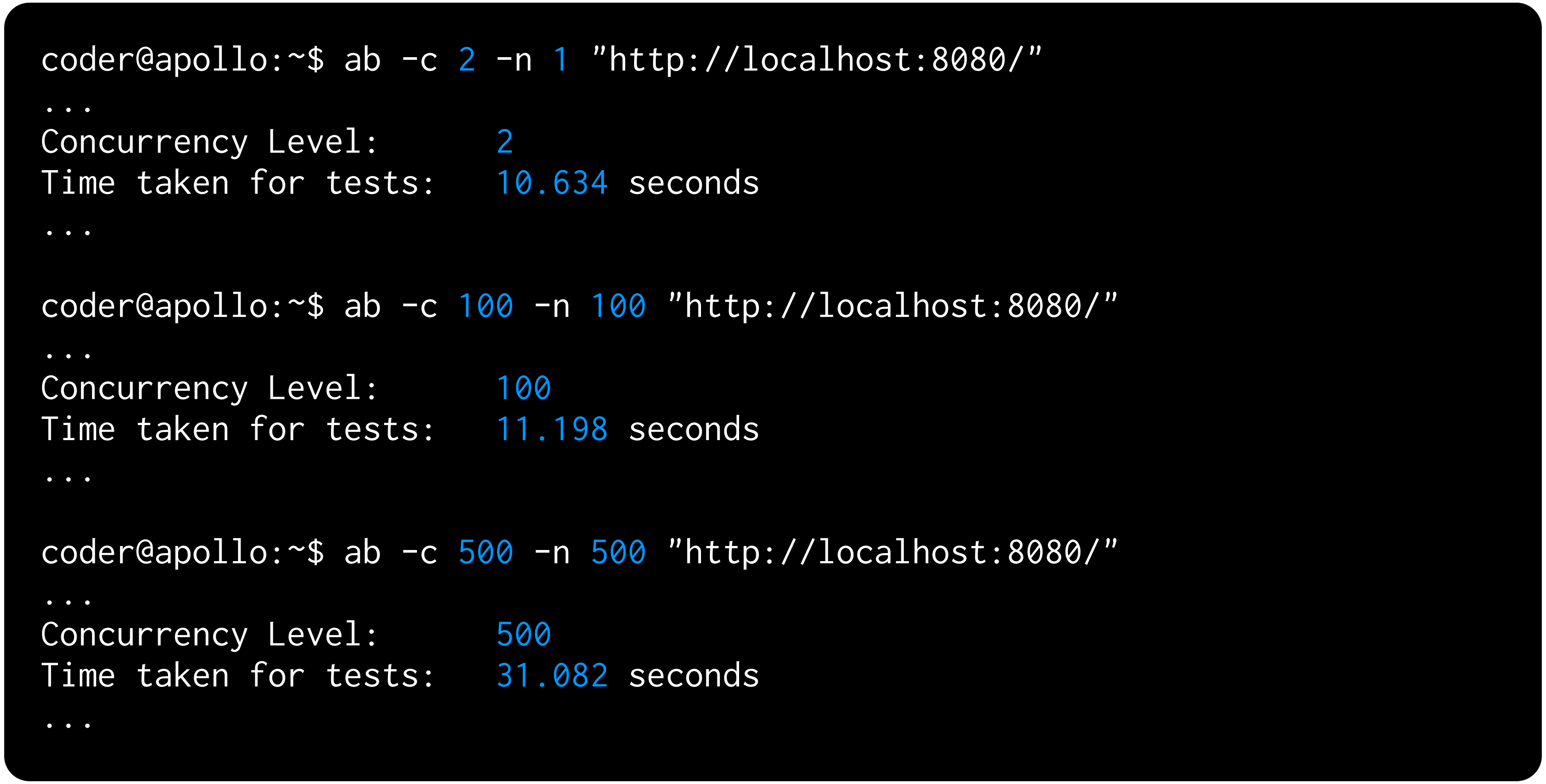
THE FORK BE WITH YOU

```
coder@apollo:~$ ab -c 1 -n 1 "http://localhost:8080/"
...
Concurrency Level:      1
Time taken for tests:   10.531 seconds
...
```

```
coder@apollo:~$ ab -c 1 -n 2 "http://localhost:8080/"
...
Concurrency Level:      1
Time taken for tests:   20.108 seconds
...
```

nodeJS

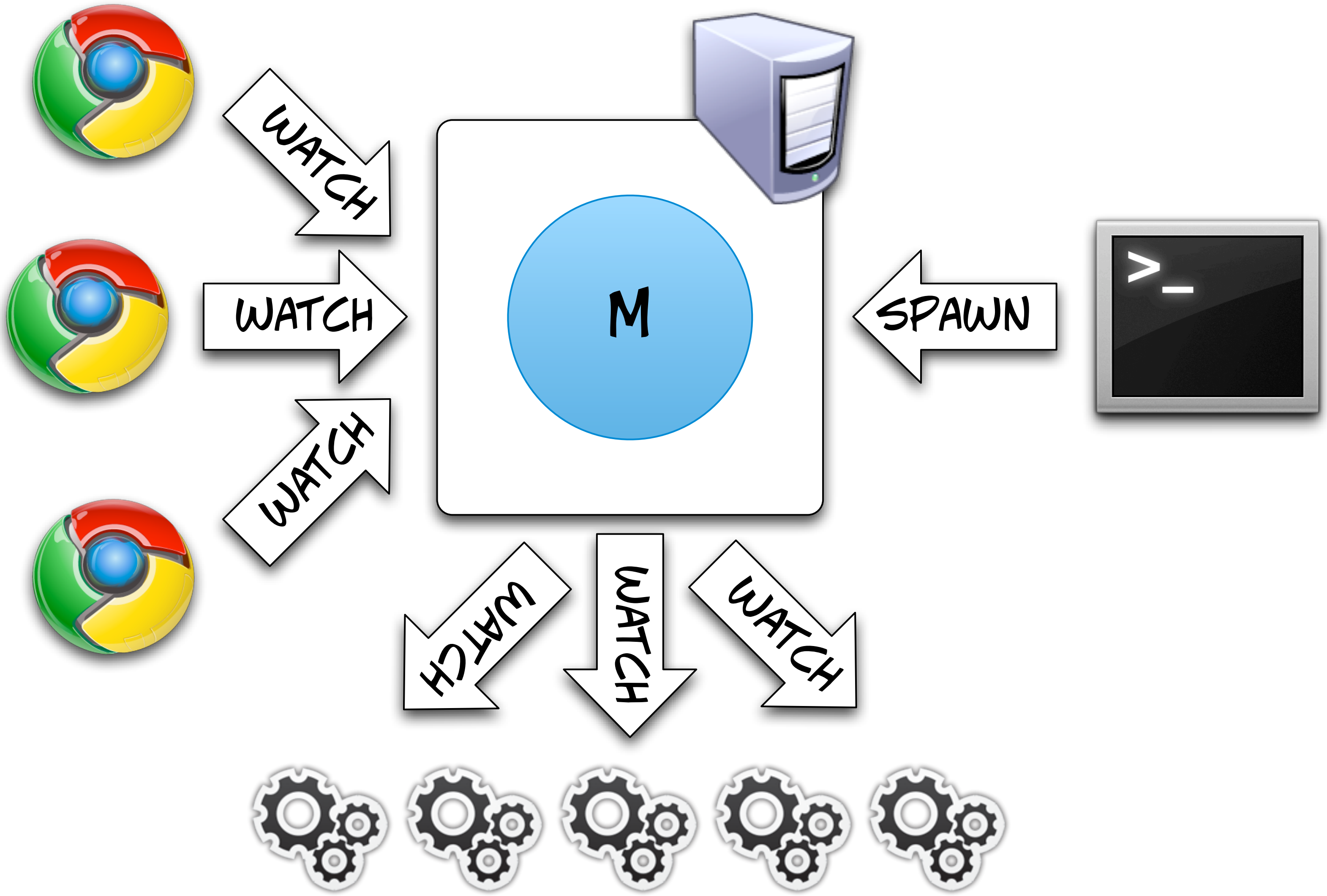
THE FORK BE WITH YOU



```
coder@apollo:~$ ab -c 2 -n 1 "http://localhost:8080/"
...
Concurrency Level:      2
Time taken for tests:   10.634 seconds
...

coder@apollo:~$ ab -c 100 -n 100 "http://localhost:8080/"
...
Concurrency Level:      100
Time taken for tests:   11.198 seconds
...

coder@apollo:~$ ab -c 500 -n 500 "http://localhost:8080/"
...
Concurrency Level:      500
Time taken for tests:   31.082 seconds
...
```

nodeJS

ENTER COMET



nodeJS

ENTER COMET

DEMO

STATIC HANDLER

```
var port = 8080

var server = connect.createServer(connect.logger())
  .use("/comet", function(request, response) { ... })
  .use("/spawn", function(request, response) { ... })
  .use("/", connect.staticProvider({
    "root": path.join(__dirname, "static"),
    "cache": true
  }))

comet.attach(server)

server.listen(port)
```

04#PROGRESS/PROGRESS_SERVER.JS

nodeJS COMET HANDLER

```
var comet = new Faye.NodeAdapter({
  "mount": "/comet", "timeout": 50
})

var server = connect.createServer(connect.logger())
  .use("/comet", function(request, response, next) {
    comet.handle(request, response)
  })
  ...
```

04#PROGRESS/PROGRESS_SERVER.JS

nodeJS SPAWN HANDLER

```
var client = comet.getClient(), jobCounter = 0

var server = connect.createServer(connect.logger())
  .use("/comet", function(request, response) { ... })
  .use("/spawn", function(request, response, next) {
    var worker = spawn("./long_running_process.sh"),
        jobId = jobsCounter++

    response.writeHead(200, { "Content-Type": "plain/text" })
    response.end("OK\n")

    worker.stdout.on("data", function(progress) {
      client.publish("/job-progress", {
        "id": jobId,
        "progress": parseInt(progress.toString(), 10)
      })
    })
  })
```

04#PROGRESS/PROGRESS_SERVER.JS

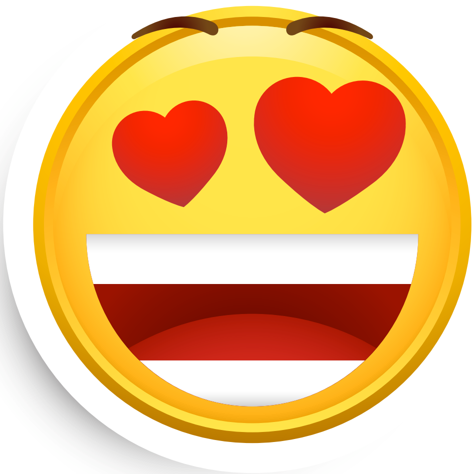
nodeJS IN BROWSER

```
<script>
  $(function() {
    var comet = new Faye.Client("/comet")

    comet.connect()
    comet.subscribe("/job-progress", function(job) {
      $("#template").progressBar(job.id, job.progress)
    })
  })
</script>
```

04#PROGRESS/STATIC/INDEX.HTML

nodeJS

NODE.JS IS AWESOME BUT WHEN SHOULD I USE IT?



nodeJS WHEN TO USE IT?

- CHAT/MESSAGING
- REAL-TIME APPLICATIONS
- INTELLIGENT PROXIES
- HIGH CONCURRENCY APPLICATIONS
- COMMUNICATION HUBS
- COORDINATORS

nodeJS

PLEASE TELL ME SOMETHING BAD ABOUT NODE.JS

nodeJS SOME WARNINGS

- RELEASE STABLE 0.2.4 (YOUNG)
- LOTS OF STUFFS TO LOOK AT
- LOTS OF HALF BACKED STUFFS
- RETRO COMPATIBILITY???
- BAD AT HANDLING STATIC CONTENTS
- HARD TO FIND ORGANIZED AND AUTHORITATIVE INFORMATIONS

nodeJS

The Pragmatic Programmers
Programming Erlang
Software for Concurrent World
Joe Armstrong
Web Services for the Real World
RESTful Web Services
O'REILLY
Unearthing the excellence in JavaScript
JavaScript: The Good Parts
O'REILLY
YAHOO! PRESS
Douglas Crockford
Build Better Applications with Coding and Design Patterns
JavaScript Patterns
O'REILLY
YAHOO! PRESS
Programming the Web
JavaScript Cookbook
O'REILLY
Shelley Powers
Copyrighted Material
REST in Practice
Hypermedia and Systems Architecture
Jim Webber
Savas Parastatidis
Ian Robinson
O'REILLY

nodeJS

QUESTIONS?

CleanCode

GABRIELE LANA
GABRIELE.LANA@CLEANCODE.IT
TWITTER: @GABRIELELANA

THIS PRESENTATION CODE IS ON
HTTPS://GITHUB.COM/GABRIELELANA/NODE-EXAMPLES