NODE.JS ANTI-PATTERNS

and bad practices

ADOPTION OF NODE.JS KEEPS GROWING

CHAMPIONS

Walmart, eBay, PayPal, Intuit, Netflix, LinkedIn, Microsoft, Uber, Yahoo ...

JAVA → NODE.JS

.NET → NODE.JS

... → NODE.JS

The clash of paradigms leads to anti-patterns

IGOR

Engineer @ YLD

PEDRO

CTO @ YLD

YLD does Node.js consulting

WHERE DO THESE ANTI-PATTERNS COME FROM?

NODE.JS ANTI-PATTERNS AND BAD PRACTISES

The opinionated and incomplete guide

YOUR MILEAGE MAY VARY

MEET JANE

JANE

- Experienced Java developer in a big enterprise
- Limited experience with JavaScript

JANE'S QUEST

create a Node.js-based prototype of an API service for a new mobile app

LET'S TRY THIS JAVASCRIPT ON THE SERVER THING...

```
function getTask(jobName, callback) {
  redisSlave.hmget('job:'+jobName, 'bTTG', 'beDestOf', function (err, replies) {
    if (err) logError(err);
    var bTTG = replies[0];
    var beDestOf = replies[1];
    redisCluster.blpop('ready:'+beDestOf, 10, function (err, task) {
      if (err) logError(err);
      if (task !== null && task.length) {
        var taskName = task[1];
        redisCluster.hdel('t:'+taskName, 'shsh', 'iir', 'vir', function (err) {
          if (err) logError(err);
          redisSlave.hget('job:'+beDestOf, 'iterations', function (err, iterations) {
            if (err) logError(err);
            redisCluster.hincrby('t:'+taskName, 'il', iterations, function (err) {
              if (err) logError(err);
              redisCluster.hmget('t:'+taskName, 'i', 's', function (err, solution) {
                if (err) logError(err);
                callback(null, solution[0], solution[1]);
              });
            });
          });
        });
      } else {
        deactivateJob(jobName);
      }
    });
  });
}
```

CALLBACK HELL

+ not avoiding closures

(1/22)

SYMPTOMS

```
function getTask(jobName, callback) {
  redisSlave.hmget('job:'+jobName, 'bTTG', 'beDestOf', function (err, replies) {
    if (err) logError(err);
    var bTTG = replies[0];
    var beDestOf = replies[1];
    redisCluster.blpop('ready:'+beDestOf, 10, function (err, task) {
      if (err) logError(err);
      if (task !== null && task.length) {
        var taskName = task[1];
        redisCluster.hdel('t:'+taskName, 'shsh', 'iir', 'vir', function (err) {
          if (err) logError(err);
          redisSlave.hget('job:'+beDestOf, 'iterations', function (err, iterations) {
            if (err) logError(err);
            redisCluster.hincrby('t:'+taskName, 'il', iterations, function (err) {
              if (err) logError(err);
              redisCluster.hmget('t:'+taskName, 'i', 's', function (err, solution) {
                if (err) logError(err);
                callback(null, solution[0], solution[1]);
              });
            });
          });
        });
      } else {
        deactivateJob(jobName);
      }
    });
  });
}
```

SOLUTION

Apply several techniques

EXAMPLE

```
function getTask(jobName, callback) {
  redisSlave.hmget('job:'+jobName, 'bTTG', 'beDestOf', function (err, replies) {
    if (err) logError(err);
    var bTTG = replies[0];
    var beDestOf = replies[1];
    redisCluster.blpop('ready:'+beDestOf, 10, function (err, task) {
      if (err) logError(err);
      if (task !== null && task.length) {
        var taskName = task[1];
        redisCluster.hdel('t:'+taskName, 'shsh', 'iir', 'vir', function (err) {
          if (err) logError(err);
          redisSlave.hget('job:'+beDestOf, 'iterations', function (err, iterations) {
            if (err) logError(err);
            redisCluster.hincrby('t:'+taskName, 'il', iterations, function (err) {
              if (err) logError(err);
              redisCluster.hmget('t:'+taskName, 'i', 's', function (err, solution) {
                if (err) logError(err);
                callback(null, solution[0], solution[1]);
              });
            });
          });
        });
      } else {
        deactivateJob(jobName);
      }
    });
  });
}
```

```
redisCluster.blpop('ready:'+beDestOf, 10, function (err, task) {
  if (err) logError(err);
  if (task !== null && task.length) {
```

```
redisCluster.blpop('ready:'+beDestOf, 10, function (err, task) {
  if (err) return callback(err);
  if (task !== null && task.length) {
```

```
function getTask(jobName, callback) {
  redisSlave.hmget('job:'+jobName, 'bTTG', 'beDestOf', function (err, replies) {
    if (err) return callback(err);
    var bTTG = replies[0];
    var beDestOf = replies[1];
    redisCluster.blpop('ready:'+beDestOf, 10, function (err, task) {
      if (err) return callback(err);
      if (task !== null && task.length) {
        var taskName = task[1];
        redisCluster.hdel('t:'+taskName, 'shsh', 'iir', 'vir', function (err) {
          if (err) return callback(err);
          redisSlave.hget('job:'+beDestOf, 'iterations', function (err, iterations) {
            if (err) return callback(err);
            redisCluster.hincrby('t:'+taskName, 'il', iterations, function (err) {
              if (err) return callback(err);
              redisCluster.hmget('t:'+taskName, 'i', 's', function (err, solution) {
                if (err) return callback(err);
                callback(null, solution[0], solution[1]);
              });
            });
          });
        });
      } else {
        deactivateJob(jobName, callback);
      }
    });
  });
}
```

```
function getTask(jobName, callback) {
  redisSlave.hmget('job:'+jobName, 'bTTG', 'beDestOf', function gotJobAttributes(err, replies) {
    if (err) return callback(err);
    var bTTG = replies[0];
    var beDestOf = replies[1];
    redisCluster.blpop('ready:'+beDestOf, 10, function poppedReady(err, task) {
      if (err) return callback(err);
      if (task !== null && task.length) {
        var taskName = task[1];
        redisCluster.hdel('t:'+taskName, 'shsh', 'iir', 'vir',
          function deletedTaskAttrs(err) {
            if (err) return callback(err);
            redisSlave.hget('job:'+beDestOf, 'iterations',
              function gotIterations(err, iterations) {
                if (err) return callback(err);
                redisCluster.hincrby('t:'+taskName, 'il', iterations,
                  function incrementedIterations(err) {
                    if (err) return callback(err);
                    redisCluster.hmget('t:'+taskName, 'i', 's',
                      function gotTaskSolution(err, solution) {
                        if (err) return callback(err);
                        callback(null, solution[0], solution[1]);
                      });
                  });
              });
          });
      } else {
        deactivateJob(jobName, callback);
      }
    });
  });
}
```

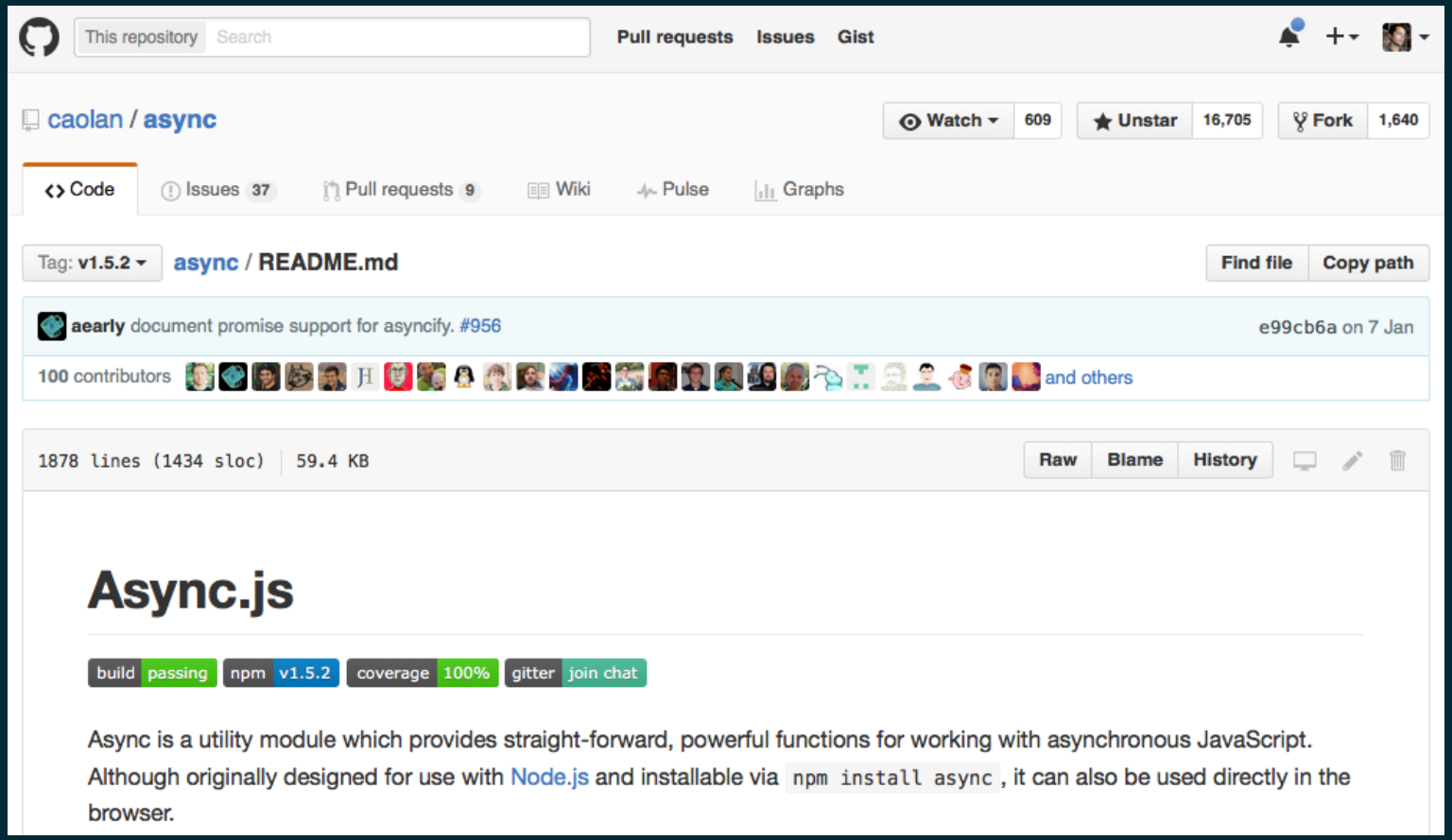
```
function getTask(jobName, callback) {
  redisSlave.hmget('job:'+jobName, 'bTTG', 'beDestOf', gotJobAttributes);

  function gotJobAttributes(err, replies) {
    if (err) return callback(err);
    var bTTG = replies[0];
    var beDestOf = replies[1];
    redisCluster.blpop('ready:'+beDestOf, 10, poppedReady);

    function poppedReady(err, task) {
      if (err) return callback(err);
      if (task !== null && task.length) {
        var taskName = task[1];
        redisCluster.hdel('t:'+taskName, 'shsh', 'iir', 'vir', deletedTaskAttrs);
      } else {
        deactivateJob(jobName, callback);
      }

      function deletedTaskAttrs(err) {
        if (err) return callback(err);
        redisSlave.hget('job:'+beDestOf, 'iterations', gotIterations);

        function gotIterations(err, iterations) {
          if (err) return callback(err);
          redisCluster.hincrby('t:'+taskName, 'il', iterations, incrementedIterations);

          function incrementedIterations(err) {
            if (err) return callback(err);
            redisCluster.hmget('t:'+taskName, 'i', 's', gotTaskSolution);

            function gotTaskSolution(err, solution) {
              if (err) return callback(err);
              callback(null, solution[0], solution[1]);
            }
          }
        }
      }
    }
  }
};
```

```
function getTask(jobName, callback) {
  redisSlave.hmget('job:'+jobName, 'bTTG', 'beDestOf', gotJobAttributes);

  function gotJobAttributes(err, replies) {
    if (err) return callback(err);
    var bTTG = replies[0];
    var beDestOf = replies[1];
    redisCluster.blpop('ready:'+beDestOf, 10, poppedReady);

    function poppedReady(err, task) {
      if (err) return callback(err);
      if (task !== null && task.length) {
        var taskName = task[1];
        redisCluster.hdel('t:'+taskName, 'shsh', 'iir', 'vir', deletedTaskAttrs);
      } else {
        deactivateJob(jobName, callback);
      }

      function deletedTaskAttrs(err) {
        if (err) return callback(err);
        redisSlave.hget('job:'+beDestOf, 'iterations', gotIterations);
      }

      function gotIterations(err, iterations) {
        if (err) return callback(err);
        redisCluster.hincrby('t:'+taskName, 'il', iterations, incrementedIterations);
      }

      function incrementedIterations(err) {
        if (err) return callback(err);
        redisCluster.hmget('t:'+taskName, 'i', 's', gotTaskSolution);
      }
    }
  }

  function gotTaskSolution(err, solution) {
    if (err) return callback(err);
    callback(null, solution[0], solution[1]);
  }
};
```

```
function getTask(jobName, callback) {
  redisSlave.hmget('job:'+jobName, 'bTTG', 'beDestOf', handlingError(gotJobAttributes));

  function gotJobAttributes(replies) {
    var bTTG = replies[0];
    var beDestOf = replies[1];
    redisCluster.blpop('ready:'+beDestOf, 10, handlingError(poppedReady));

    function poppedReady(task) {
      if (task !== null && task.length) {
        var taskName = task[1];
        redisCluster.hdel('t:'+taskName, 'shsh', 'iir', 'vir', handlingError(deletedTaskAttrs));
      } else {
        deactivateJob(jobName, callback);
      }

      function deletedTaskAttrs() {
        redisSlave.hget('job:'+beDestOf, 'iterations', handlingError(gotIterations));
      }

      function gotIterations(iterations) {
        redisCluster.hincrby('t:'+taskName, 'il', iterations, handlingError(incrementedIterations))
      }

      function incrementedIterations() {
        redisCluster.hmget('t:'+taskName, 'i', 's', handlingError(gotTaskSolution));
      }
    }
  }

  function gotTaskSolution(solution) {
    callback(null, solution[0], solution[1]);
  }

  function handlingError(next) {
    return function(err) {
      if (err) {
        callback(err);
      } else {
        var args = Array.prototype.slice.call(arguments, 1);
        next.apply(null, args);
      }
    }
  }
};
```

```
function getTask(jobName, callback) {
  redisSlave.hmget('job:'+jobName, 'bTTG', 'beDestOf', handlingError(gotJobAttributes));

  function popNextTask(replies) {
    var bTTG = replies[0];
    var beDestOf = replies[1];
    redisCluster.blpop('ready:'+beDestOf, 10, handlingError(deleteTaskAttributes));

    function deleteTaskAttributes(task) {
      if (task !== null && task.length) {
        var taskName = task[1];
        redisCluster.hdel('t:'+taskName, 'shsh', 'iir', 'vir', handlingError(getIterations));
      } else {
        deactivateJob(jobName, callback);
      }

      function getIterations() {
        redisSlave.hget('job:'+beDestOf, 'iterations', handlingError(incrementIterations));
      }

      function incrementIterations(iterations) {
        redisCluster.hincrby('t:'+taskName, 'il', iterations, handlingError(getTaskSolution));
      }

      function getTaskSolution() {
        redisCluster.hmget('t:'+taskName, 'i', 's', handlingError(gotTaskSolution));
      }
    }
  }

  function gotTaskSolution(solution) {
    callback(null, solution[0], solution[1]);
  }

  function handlingError(fn) {
    return function(err) {
      if (err) {
        callback(err);
      } else {
        var args = Array.prototype.slice.call(arguments, 1);
        fn.apply(null, args);
      }
    }
  }
};
```

ASYNC


This repository
Search
Pull requests
Issues
Gist
caolan / async
Watch 609
Unstar 16,705
Fork 1,640
Code
Issues 37
Pull requests 9
Wiki
Pulse
Graphs
Tag: v1.5.2
async / README.md
Find file
Copy path
aearly document promise support for asyncify. #956
e99cb6a on 7 Jan
100 contributors
and others
1878 lines (1434 sloc)
59.4 KB
Raw
Blame
History
Async.js
build passing
npm v1.5.2
coverage 100%
gitter join chat
Async is a utility module which provides straight-forward, powerful functions for working with asynchronous JavaScript. Although originally designed for use with Node.js and installable via npm install async, it can also be used directly in the browser.

```
function getTask(jobName, callback) {
  var bTTG, beDestOf, taskName;

  async.waterfall([
    getJobAttributes,
    popNextTask,
    deleteTaskAttributes,
    getIterations,
    incrementIterations,
    getTaskSolution,
    getFinalTaskSolution
    ], callback);

  function getJobAttributes(cb) {
    redisSlave.hmget('job:'+jobName, 'bTTG', 'beDestOf', cb);
  }

  function popNextTask(replies, cb) {
    bTTG = replies[0];
    beDestOf = replies[1];
    redisCluster.blpop('ready:'+beDestOf, 10, cb);
  }

  function deleteTaskAttributes(task, cb) {
    if (task !== null && task.length) {
      taskName = task[1];
      redisCluster.hdel('t:'+taskName, 'shsh', 'iir', 'vir', cb);
    } else {
      deactivateJob(jobName, callback);
    }
  }

  function getIterations(result, cb) {
    redisSlave.hget('job:'+beDestOf, 'iterations', cb);
  }

  function incrementIterations(iterations, cb) {
    redisCluster.hincrby('t:'+taskName, 'il', iterations, cb);
  }

  function getTaskSolution(result, cb) {
    redisCluster.hmget('t:'+taskName, 'i', 's', cb);
  }

  function getFinalTaskSolution(solution, cb) {
    cb(null, solution[0], solution[1]);
  }
};
```

SOLUTION

- Return early
- Name your functions
- Moving functions to the outer-most scope as possible
- Don't be afraid of hoisting to make the code more readable
- Use a tool like `async` to orchestrate callbacks

USING A LONG LIST OF ARGUMENTS INSTEAD OF OPTIONS

```
function createUser(firstName, lastName, birthDate, address1, address2, postCode, ...) {
  // ..
}
```

```
function createUser(opts) {
  var firstName = opts.firstName;
  var lastName = opts.lastName;
  // ..
  var otherValue = opts.otherValue || defaultValue;
  // ..
}
```

use `utils._extend`:

```
var extend = require('utils')._extend;

var defaultOptions = {
  attr1: 'value 1',
  attr2: 'value 2',
};

module.exports = MyConstructor(opts) {
  var options = extend(extend({}, defaultOptions), opts);
}
```

use `xtend`:

```
var extend = require('xtend');

var defaultOptions = {
  attr1: 'value 1',
  attr2: 'value 2',
};

module.exports = MyConstructor(opts) {
  var options = extend({}, defaultOptions, opts);
}
```

```
function myFunction(arg1, [arg2], [arg3], [arg4]) {
    // ...
}
```

ABUSING VARIABLE ARGUMENTS

(3/22)

PROBLEMS

- Hard to make it work generally
- Error-prone

```
fs.readFile = function(path, options, callback_) {
  var callback = maybeCallback(arguments[arguments.length -

  if (typeof options === 'function' || !options) {
    options = { encoding: null, flag: 'r' };
  } else if (typeof options === 'string') {
    options = { encoding: options, flag: 'r' };
  } else if (!options) {
    options = { encoding: null, flag: 'r' };
  } else if (typeof options !== 'object') {
    throw new TypeError('Bad arguments');
  }

  var encoding = options.encoding;
  assertEncoding(encoding);
  // ...
```

POOR USE OF MODULARITY

(4/22)

- Files with > 200 LoC
- Lots of scattered functions
- Low cohesion
- No reuse
- Testing is hard

- All the handlers for a given resource inside the same module
- Modules that have loosely related functions inside it because it's the only place these functions are being used.

- modules are cheap
- expose a documented interface
- try to keep modules under 200 LoC

OVERUSE OF CLASSES FOR MODELLING

(5/22)

```
var MOD = require('MOD');
var config = new MOD.Config({ opt: 'foobar' });
var client = new MOD.Thing.Client(config);
var actor = new MOD.Thing.Actor(actorOpts);
client.registerActor(actor)
```

```
var MOD = require('MOD');
var config = new MOD.Config({ opt: 'foobar' });
var client = new MOD.Thing.Client(config);
var actor = new MOD.Thing.Actor(actorOpts);
client.registerActor(actor)
```

vs

```
var Client = require('MODClient');
var client = Client({
  opt: 'foobar',
  actor: actorOpts
});
```

```
module.exports = Counter;

function Counter() {
  this._counter = 0;
}

Counter.prototype.increment = function() {
  this._counter += 1;
};

Counter.prototype.get = function() {
  return this._counter;
};
```

```
module.exports = function createCounter(options) {

  var counter = 0;

  function increment() {
    counter += 1;
  }

  function get() {
    return counter;
  }

  return {
    increment: increment,
    get: get,
  };
}
```

LET'S TRY THIS NODE.JS THING...

```
doThis(function(err1, result1) {
  doThat(result1.someAttribute, function(err2, result2) {
    if (err2) {
      ...

    } else {
      ...
    }
}
```

IGNORING CALLBACK ERRORS

(6/22)

```
doThis(function(err1, result1) {
  doThat(result1.someAttribute, function(err2, result2) {
    if (err2) {
      ...

    } else {
      ...
    }
}
```

SOLUTIONS

USE A LINTER

like **ESLint** and enable the rule

http://eslint.org/docs/rules/handle-callback-err

USE ASYNC OR SIMILAR

```
var async = require('async');
async.waterfall([
  doThis,
  doThat,
], done);

function doThis(cb) {
  // ...
}

function doThat(result, cb) {
  // ...
}

function done(err) {
  // you still have to handle this error!
}
```

USE PROMISES

```
doThis()
  .then(doThat).
  .catch(handleError);

function handleError(err) {
  // .. handle error
}
```

THE KITCHEN-SINK MODULE

(7/22)

```
var normalizeRequestOptions = function(options) { /* ... */
var isBinaryBuffer = function(buffer) { /* ... */ };
var mergeChunks = function(chunks) { /* ... */ };
var overrideRequests = function(newRequest) { /* ... */ };
var restoreOverriddenRequests = function() { /* ... */ };
function stringifyRequest(options, body) { /* ... */ }
function isContentEncoded(headers) { /* ... */ }
function isJSONContent(headers) { /* ... */ }
var headersFieldNamesToLowerCase = function(headers) { /* ..
var headersFieldsArrayToLowerCase = function (headers) { /*
var deleteHeadersField = function(headers, fieldNameToDelete
function percentDecode (str) { /* ... */ }
function percentEncode(str) { /* ... */ }
function matchStringOrRegexp(target, pattern) { /* ... */ }
function formatQueryValue(key, value, options) { /* ... */ }
function isStream(obj) { /* ... */ }

exports.normalizeRequestOptions = normalizeRequestOptions;
exports.isBinaryBuffer = isBinaryBuffer;
exports.mergeChunks = mergeChunks;
exports.overrideRequests = overrideRequests;
exports.restoreOverriddenRequests = restoreOverriddenRequest
exports.stringifyRequest = stringifyRequest;
exports.isContentEncoded = isContentEncoded;
exports.isJSONContent = isJSONContent;
exports.headersFieldNamesToLowerCase = headersFieldNamesToLo
```

```
exports.headersFieldsArrayToLowerCase = headersFieldsArrayTo

exports.deleteHeadersField = deleteHeadersField;
```

https://github.com/pgte/nock/blob/master/lib/common.js

1. Embrace modules
2. Enforce SRP
3. Externalise modules
4. Individualised packaging

initialization:

```
global.App = ...
```

from any file:

```
App.Models.Person.get(id);
```

PLACING VALUES IN GLOBAL OBJECTS

(8/22)

SYMPTOMS

Adding properties to any of these:

- `process`
- `global`
- `GLOBAL`
- `root`
- `this` on the global scope
- any other global reference, e.g. `Buffer` or `console`

EXAMPLES

```
global.utilityFunction = function() { /*...*/ };

// or ...

global.maxFoosticles = 10;
```

PROBLEM

- Dependencies become implicit instead of explicit.
- Makes the code harder to reason about for a newcomer

SOLUTION

Leverage the module cache

EXAMPLE

Create a file module:

```
exports.maxFoosticles = 10;
```

Require this file module in other files

```
var config = require('./config');
config.maxFoosticles // => 10
```

EXAMPLE:

config.js:

```
module.exports = {
    couchdb: {
        baseUrl: "https://my.couchdb.url:4632" || process.env.COUCHDB_URL
    },
    mailchimp: {
        // ...
    }
}
```

EXAMPLE 2

models/people.js

```
module.exports = new PeopleModel();
```

client:

```
var People = require('./models/people');
People.find(...);
```

EXCEPTIONS

- Testing framework
- ...?

```
var exec = require('child_process').execSync;

module.exports = function pay(req, reply) {
  var fraudCheck = exec('fraud_check', JSON.stringify(req.pa
  // ...
};
```

SYNCHRONOUS EXECUTION AFTER INITIALISATION

```
module.exports = function getAttachment(req, reply) {
  db.getAttachment(req.params.id, loadAttachment);
  function loadAttachment(err, path) {
    if (err) return reply(err);
    reply(fs.readFileSync(path, { encoding: 'utf-8' }));
  }
};
```

SYMPTOMS

- Higher request latency
- Performance decays quickly when under load

- fs.readFileSync
- fs.accessSync
- fs.changeModSync
- fs.chownSync
- fs.closeSync
- fs.existsSync
- …

Asynchronous initialisation

```
var cache = require('./cache');
cache.warmup(function(err) {
  if (err) throw err;
  var server = require('./server');
  server.start();
});
```

```
mongoose.find().stream().pipe(transform).pipe(res);
```

DANGLING SOURCE STREAM

(10/22)

SYMPTOMS

When a stream throws an error or closes while piping, streams are not properly disposed and resources leak.

SOLUTION

- listen for `error` and `close` events on every stream and cleanup
- or use the pump package instead of the native stream.pipe()

WRONG:

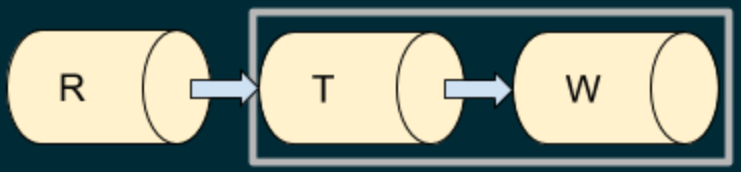
```
mongoose.find().stream().pipe(transform).pipe(res);
```


R
T
W


R


R
T


R
T
W


R
T
W

BETTER:

```
var stream = mongoose.find().stream();
var transform = ...;

var closed = false;
stream.once('close', function() {
  closed = true;
});

transform.on('error', function(err) {
  if (! closed) stream.destroy();
});

transform.on('close', function(err) {
  if (! closed) stream.destroy();
});

// ...and the same thing for transform <-> res

stream.pipe(transform).pipe(res);
```

EVEN BETTER:

```
var pump = require('pump');

pump(mongoose.find().stream(), transform, res);
```

```
var baz = require('../../../foo/bar/baz');
```

CHANGING THE WAY

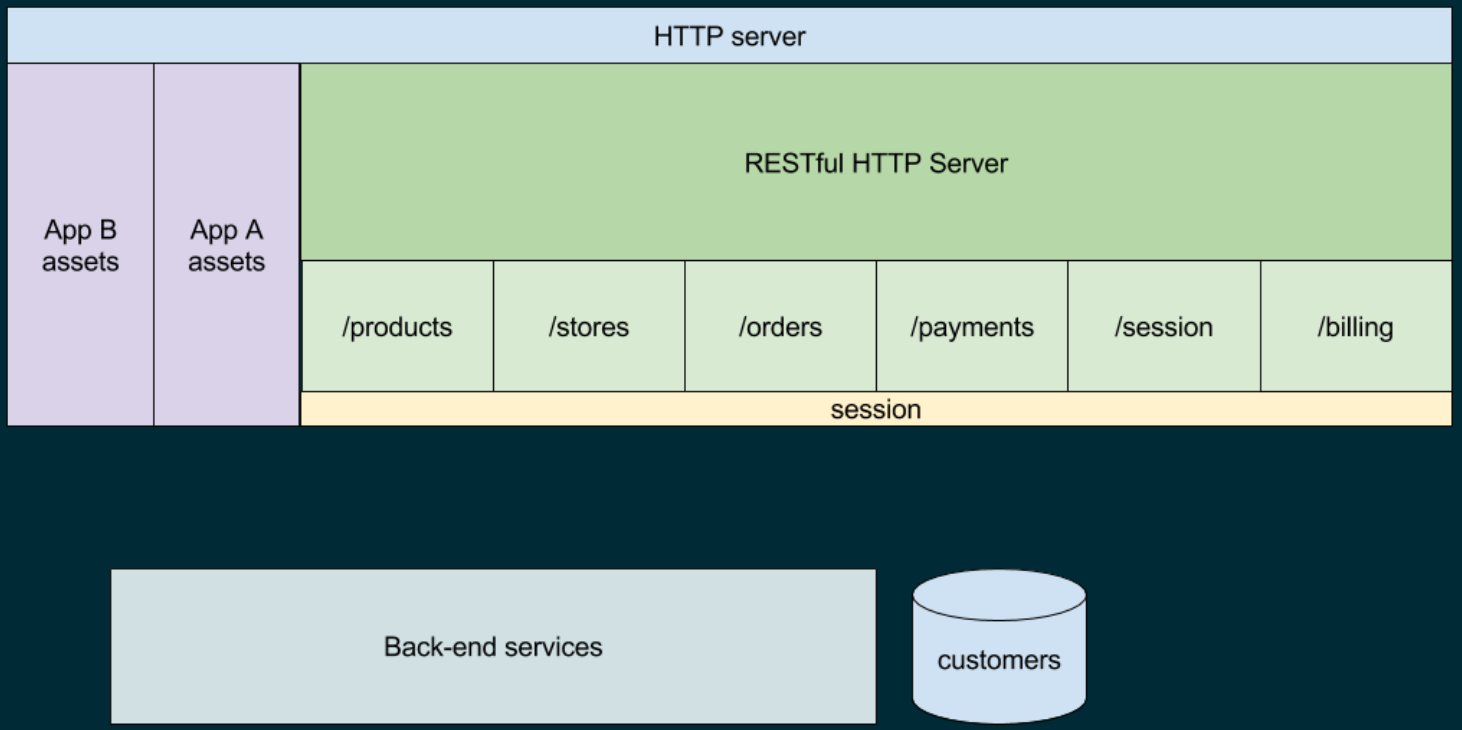
`require()`

WORKS

```
var baz = require('/foo/bar/baz');
```

- Setting `NODE_PATH`
- Using a module that requires in a different way.
  e.g.
  - 'rootpath'
  - 'require-root'
  - 'app-root-path'
  - 'root-require'

```
$ tree
.
├── lib
│   ├── bar
│   │   └── bar.js
│   ├── foo
│   │   └── foo.js
│   └── index.js
├── node_modules
│   └── ...
├── package.json
└── test
    └── suite.js
```


HTTP server
App B assets
App A assets
RESTful HTTP Server
/products
/stores
/orders
/payments
/session
/billing
session
Back-end services
customers

THE MONOLITHIC APPLICATION

NODE IS GREAT FOR PROTOTYPING

But this may become a trap

EXAMPLES

- Views and API on the same code base
- Services that do a lot of disjoint things

SYMPTOMS

POOR TEST COVERAGE

BRITTLE IN SOME PARTS

NOT MUCH ELBOW ROOM

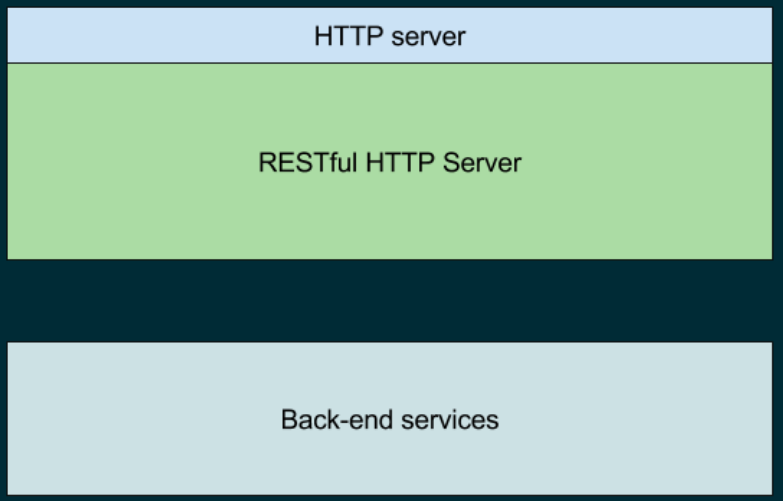
LONG DELIVERY CYCLES

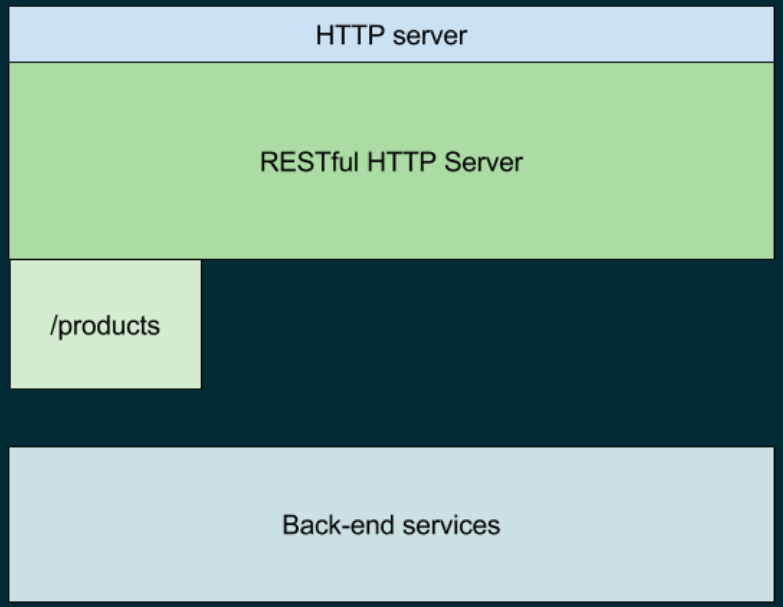
and high error rate

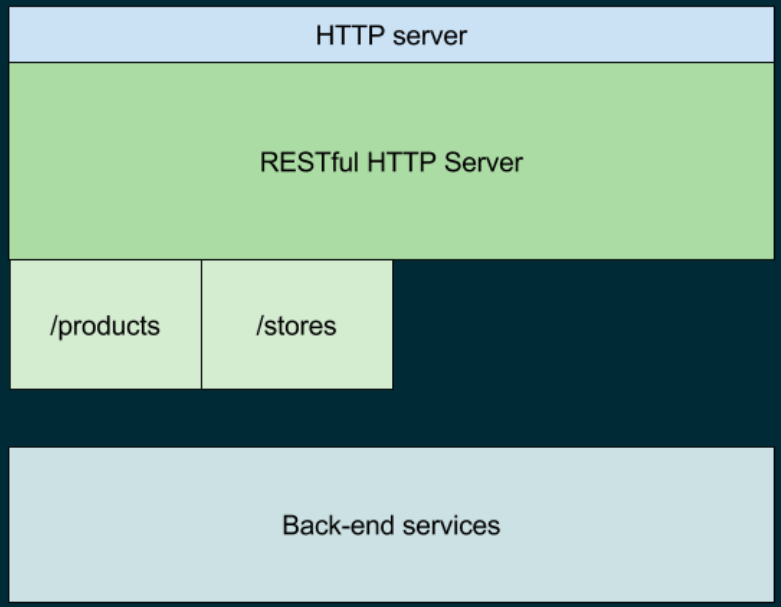
LONG TIME OF CODE ONBOARDING AND HAND-HOLDING

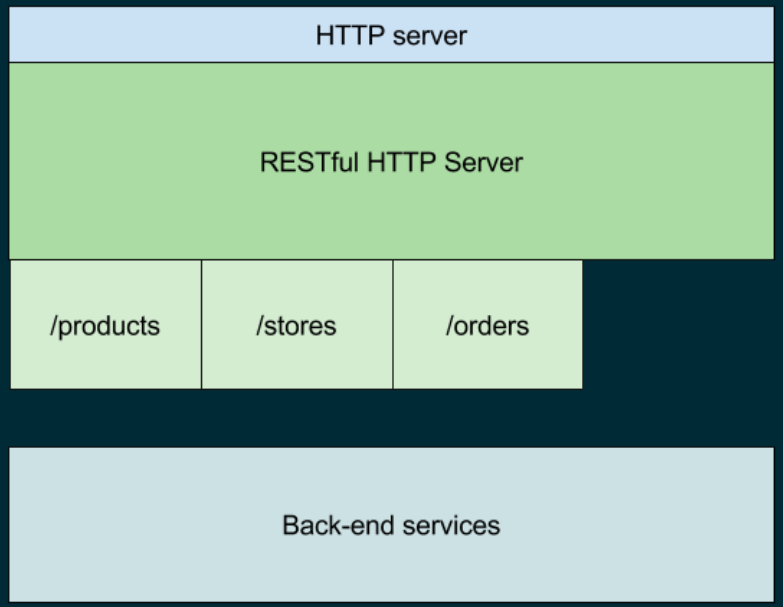
HOW WE GET THERE

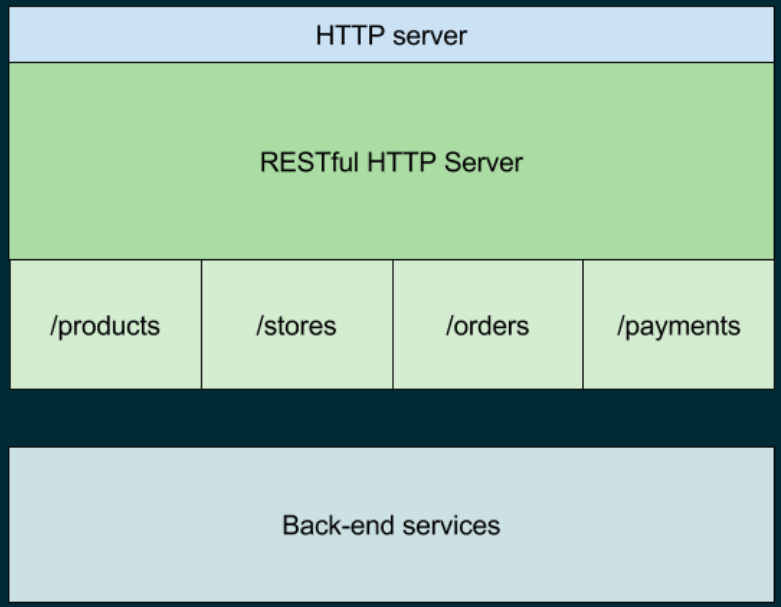
EXAMPLE

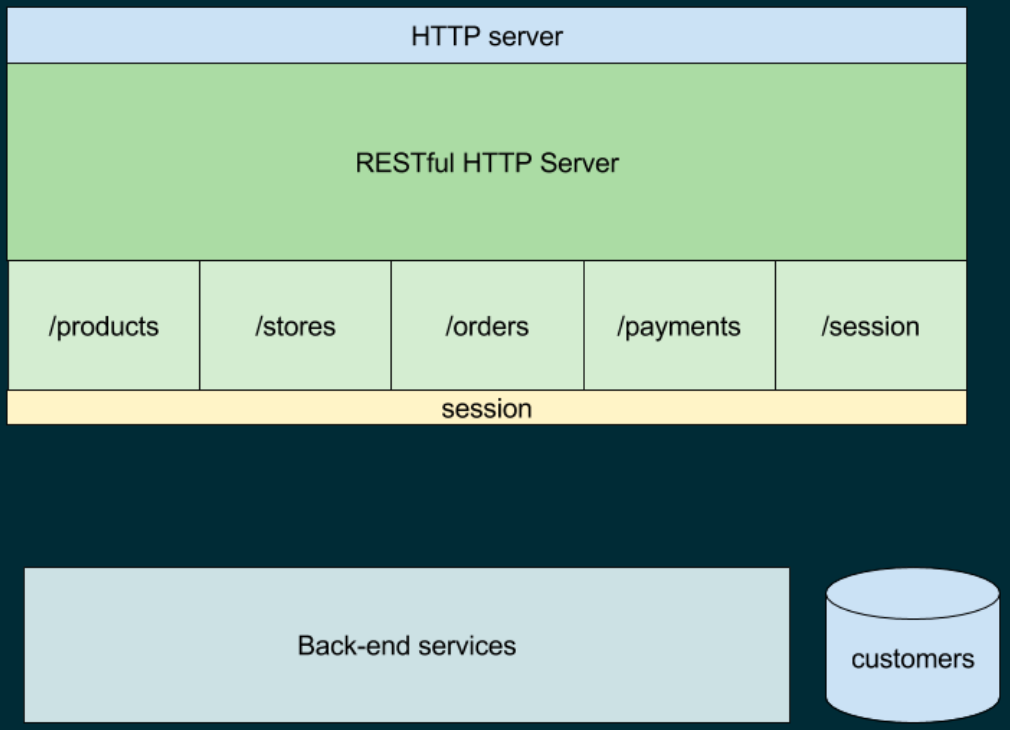


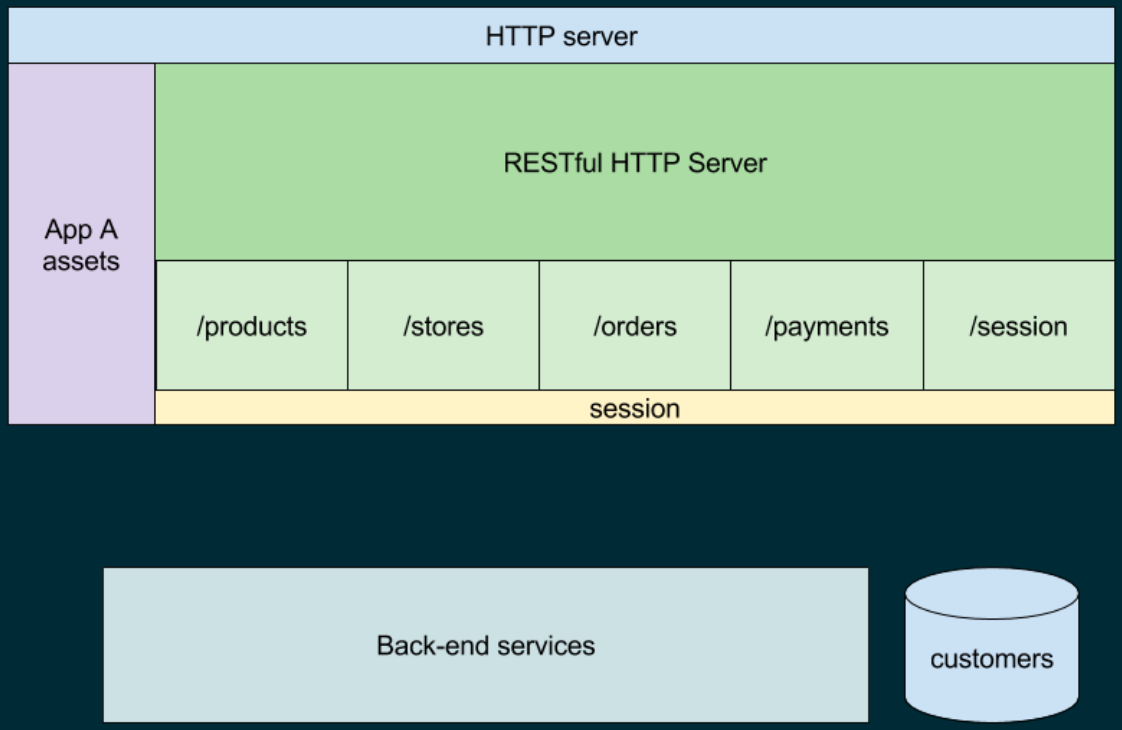

HTTP server
RESTful HTTP Server
/products
Back-end services

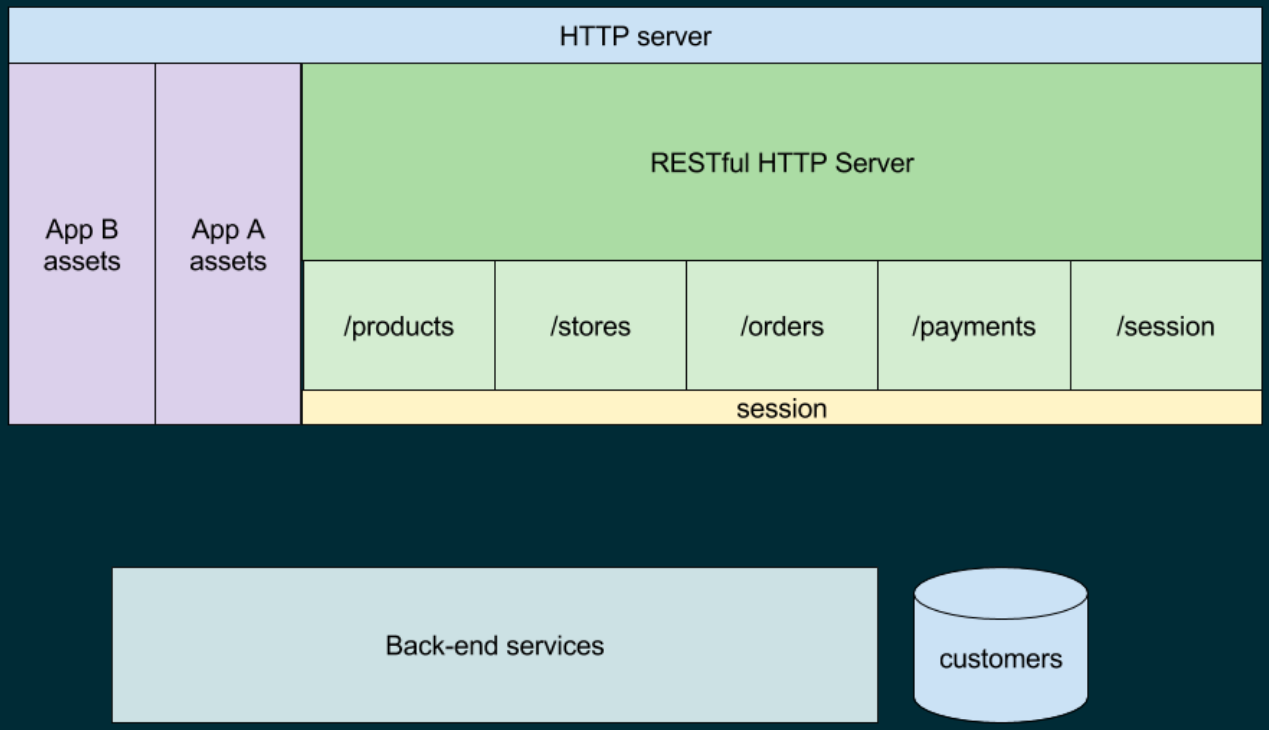

HTTP server
RESTful HTTP Server
/products
/stores
Back-end services

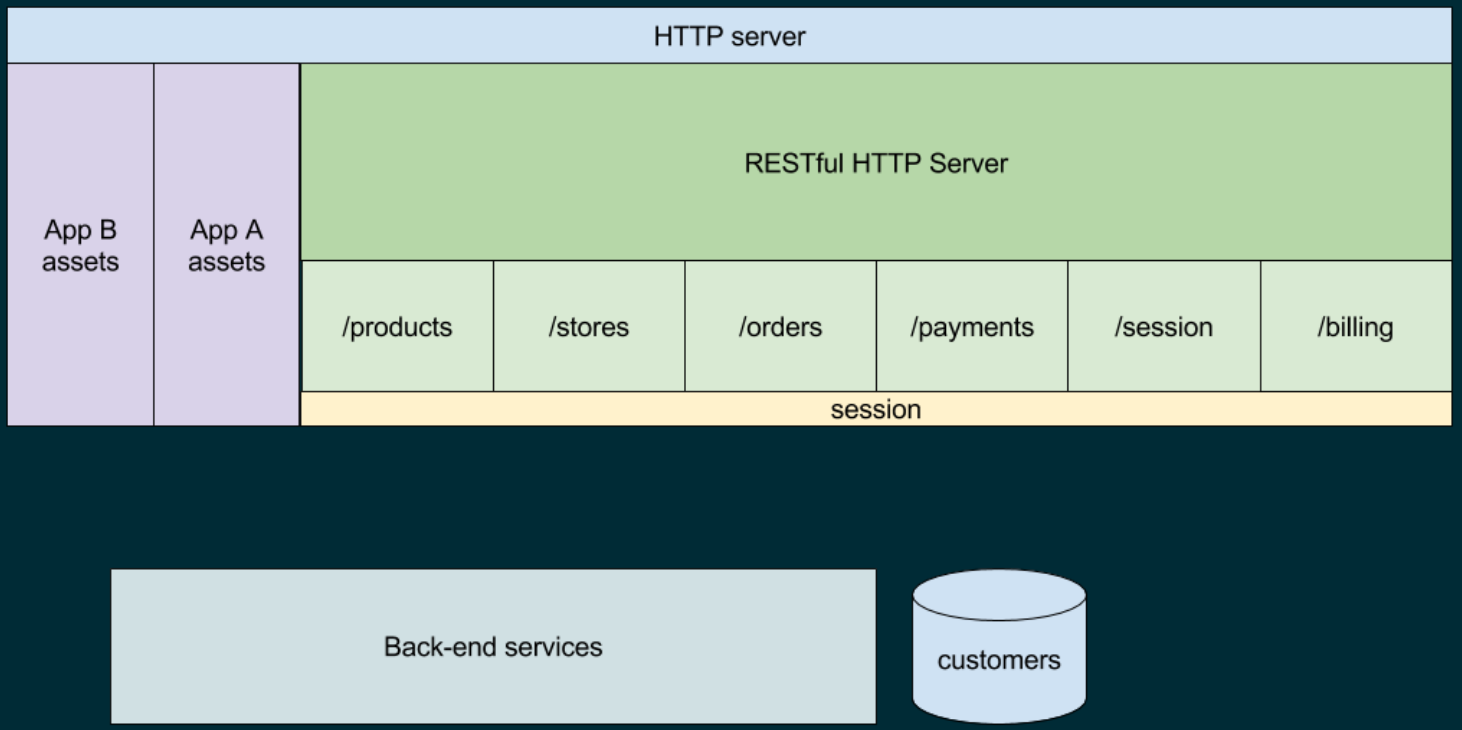

HTTP server
RESTful HTTP Server
/products
/stores
/orders
Back-end services


HTTP server
RESTful HTTP Server
/products
/stores
/orders
/payments
Back-end services


HTTP server
RESTful HTTP Server
/products
/stores
/orders
/payments
/session
session
Back-end services
customers


HTTP server
App A assets
RESTful HTTP Server
/products
/stores
/orders
/payments
/session
session
Back-end services
customers


HTTP server
App B assets
App A assets
RESTful HTTP Server
/products
/stores
/orders
/payments
/session
session
Back-end services
customers


HTTP server
App B assets
App A assets
RESTful HTTP Server
/products
/stores
/orders
/payments
/session
/billing
session
Back-end services
customers

SOLUTIONS

SEPARATE VIEWS FROM API

Embrace Cross-origin resource sharing.

NODE IS GREAT AT NETWORKING

SLOWLY MIGRATE

KEEP THE MONOLITH RUNNING

but develop new or updated features into separate smaller services

TURN A MACRO-SERVICE INTO A SET OF MICRO-SERVICES

CHALLENGES

Versioning, Testing, Shared Asset Management, Deploying, Service Lookup

TESTING

LITTLE OR NO AUTOMATED TESTS

```
$ tree
.
├── lib
│   ├── bar.js
│   ├── foo.js
│   └── index.js
├── node_modules
│   └── ...
└── package.json
```

- Project on-boarding takes a long time
- App is brittle, needs fixing in production all the time
- Developers are reluctant to make changes
- QA process doesn't seem strict enough
- QA cycle takes too long

CAUSES

- Lack of experience writing tests
- No testing culture
  - Weak quality culture
  - Management doesn't value tests
  - "Wasting" time in automated tests is forbidden

- Start with tests — TDD
- Measure test coverage, aim for 100%
- Start with regression tests in existing monoliths

```
test('do something', function(t) {
  var MyClass = require('..');
  a = MyClass();
  a.doSomething();
  t.equal(a._privateThing, 'some value');
  t.end();
});
```

TESTING AT THE WRONG LEVEL

SYMPTOMS

UNIT TESTS THAT REACH INTO THE BOWELS OF A MODULE

TESTING THE IMPLEMENTATION, NOT THE INTERFACE

FILE MODULES EXPOSING EXTRA DETAILS

CHANGING IMPLEMENTATION DETAILS

often requires updating the tests

CAUSES

POOR USE OF MODULARITY

POOR SEPARATION OF CONCERNS

EXAMPLE

Testing a side effect:

```
test('do something', function(t) {
  var MyClass = require('../');
  a = MyClass();
  a.doSomething();

  test.equal(a._privateThing, 'some value');

  t.end();
});
```

EXAMPLE

Invoking a private API:

```
test('do something', function(t) {
  var MyClass = require('../');
  a = MyClass();

  test.equal(a._doSomethingPrivate(), 'some value');;

  t.end();
});
```

SOLUTIONS

TEST AT THE INTERFACE LEVEL

All that the tests should require is

```
var mymodule = require('..')
```

- Test the behaviour not the implementation
- Don't conflate concerns on the same module
- Externalise: Make good use of NPM

FACILITATE TESTING

By overriding default options

```
options.timeout = muchShorterValue;
```

USE MOCKS, SPIES OR DEPENDENCY INJECTION FOR THIRD-PARTY PACKAGES

as a last resource

and as long as you don't spy on internal stuff

```
function mockClient(code, path, extra) {
  return function(debug, error) {
    extra = extra || {};
    var opts = _.extend(extra, {
      url: helpers.couch + path,
      log: debug,
      request: function(req, cb) {
        if(error) {
          return cb(error);
        }

        if(code === 500) {
          cb(new Error('omg connection failed'));
        } else {
          cb(null, {
            statusCode: code,
            headers: {}
          }, req); }
        }
    });

    return Client(opts);
  };
}
```

FOCUS ON TESTING THE INTERFACE

COLLABORATION IS HARD...

DEPENDING ON GLOBALLY INSTALLED MODULES ON NPM SCRIPTS

```
Jane$ npm install -g lattemacchiato
... installed version 1.4 ...
```

```
...
    "scripts": {
      "test": "lattemacchiato --extra-sugar test/
    },
...
```

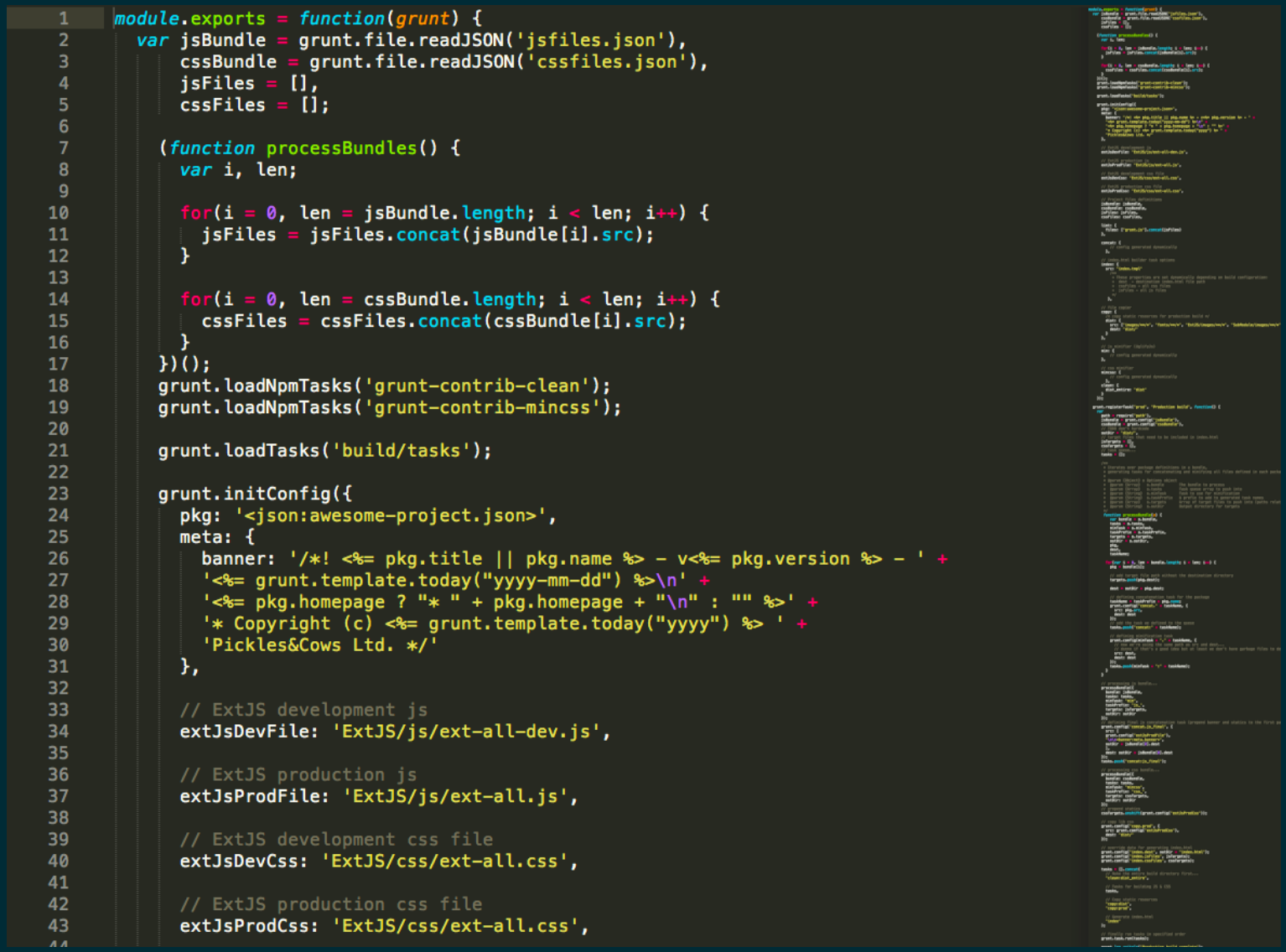
```
Jane$ npm test
lattemacchiato: all ok!
```

```
julia$ npm test
command not found: lattemacchiato
julia$ npm install -g lattemacchiato
... installed version 2.3 ...
```

```
$ npm i --save-dev lattemacchiato
```

```
...
    "scripts": {
      "test": "lattemacchiato --extra-sugar test/"
    },
    "devDependencies": {
      "lattemacchiato": "^1.4"
...
```

```
$ ls node_modules/.bin
lattemacchiato
```



```
module.exports = function(grunt) {
  var jsBundle = grunt.file.readJSON('jsfiles.json'),
      cssBundle = grunt.file.readJSON('cssfiles.json'),
      jsFiles = [],
      cssFiles = [];

  (function processBundles() {
    var i, len;

    for(i = 0, len = jsBundle.length; i < len; i++) {
      jsFiles = jsFiles.concat(jsBundle[i].src);
    }

    for(i = 0, len = cssBundle.length; i < len; i++) {
      cssFiles = cssFiles.concat(cssBundle[i].src);
    }
  })();
  grunt.loadNpmTasks('grunt-contrib-clean');
  grunt.loadNpmTasks('grunt-contrib-mincss');

  grunt.loadTasks('build/tasks');

  grunt.initConfig({
    pkg: '<json:awesome-project.json>',
    meta: {
      banner: '/*! <%= pkg.title || pkg.name %> - v<%= pkg.version %> - ' +
      '<%= grunt.template.today("yyyy-mm-dd") %>\n' +
      '<%= pkg.homepage ? "* " + pkg.homepage + "\n" : "" %>' +
      '* Copyright (c) <%= grunt.template.today("yyyy") %> ' +
      'Pickles&Cows Ltd. */'
    },

    // ExtJS development js
    extJsDevFile: 'ExtJS/js/ext-all-dev.js',

    // ExtJS production js
    extJsProdFile: 'ExtJS/js/ext-all.js',

    // ExtJS development css file
    extJsDevCss: 'ExtJS/css/ext-all.css',

    // ExtJS production css file
    extJsProdCss: 'ExtJS/css/ext-all.css',
```

USING GULP OR GRUNT INSTEAD OF NPM SCRIPTS

SYMPTOMS

LONG TIME SPENT ON TOOLING

HARD TO CHANGE THE TASKS

- Start by using NPM scripts to automate tasks
- Use the `package.json` "pre" *and "post*" script hooks
- Use default config inside `package.json`

Then run with

```
$ npm run mytask
```

TYPICAL TASKS

- Automated Tests
- Transformations
- Watching
- Live-reloading
- Starting service
- ...

EXAMPLE

```
{
  "name": "mytestapp",
  "version": "0.0.1",
  "config": {
    "reporter": "xunit"
  },
  "scripts": {
    "start": "node .",
    "prestart": "npm run build",
    "test": "mocha tests/*.js --reporter $npm_package_config_
    "lint": "eslint",
    "test:watch": "watch 'npm test' .",
    "build": "npm run build:js && npm run build:css",
    "build:js": "browserify src/index.js > dist/index.js",
    "build:css": "stylus assets/css/index.styl > dist/index.c
 },
  "devDependencies": {
    "eslint": "2.2.0",
    "pre-commit": "1.1.2",
    "mocha": "2.4.5",
    "watch": "0.17.1",
    "stylus": "0.53.0",
    "browserify": "13.0.0"
  },
  "pre-commit": [
    "eslint",
    "test"
```

```
]
```

CHALLENGES

- Make Windows-compatible scripts
- Know when to switch to gulp (instead of building a gulp-like system)

NOT MEASURING CODE COVERAGE

(17/22)

ISTANBUL

```
instrumentation:
    excludes: ['test', 'node_modules']
check:
    global:
        lines: 100
        branches: 100
        statements: 100
        functions: 100
```

```
"scripts": {
  "test": "node --harmony tests/test.js",
  "coverage": "node --harmony node_modules/istanbul/lib/cl
  "coveralls": "cat ./coverage/lcov.info | coveralls && rm
  "jshint": "jshint lib/*.js",
  "changelog": "changelog nock all -m > CHANGELOG.md"
},
"pre-commit": [
  "jshint",
  "coverage"
]
```

POOR USE OF NPM

(18/22)

- NPM is the biggest and fastest growing open source package repo
- Make good use of existing open-source

- Ignorance of existing modules
- NIH syndrome
- Reluctance with dependency management
- "My needs are unique"

I need a testing framework that computes code coverage and sends coverage stats to coveralls.io

¯_(ツ)_/¯

I need a testing framework

`tap`, `mocha`, `lab`

I need to compute code coverage

`istanbul`

I need to send coverage stats to coveralls.io

`coveralls`

DISCOVERABILITY

- libraries.io
- github
- mailing lists
- IRC
- social networks

- License
- Release frequency
- Last updated
- Open issues
- Test coverage
- Documentation quality

SECURITY

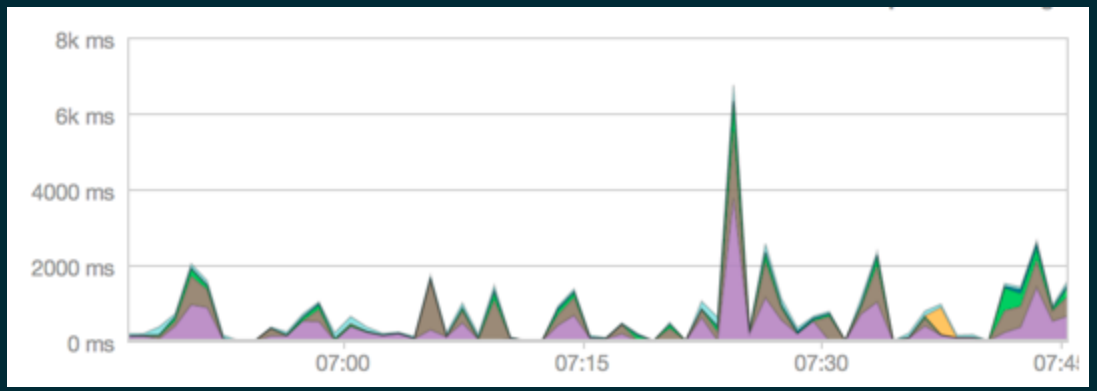
- Node Security Project — https://nodesecurity.io/
- Snyk: https://snyk.io/

PERFORMANCE IS HARD...

PERFORMING CPU-HEAVY WORK

```
function myHandler(req, res) {
  var result = req.body.items.reduce(reducer);
  res.send(result);
}
```

- Parsing (response from the database, response from external service)
- Computation-heavy work (like Natural Language Processing, Classification, Learning, etc.)
- Processing big sequences of data
- Mapping or a big dataset
- Aggregating a big dataset
- Calculating an HMAC for a big document


8k ms
6k ms
4000 ms
2000 ms
0 ms
07:00
07:15
07:30

UNLIMITED ASYNCHRONOUS ITERATIONS

#performance #reliability

(20/22)

SYMPTOMS

High request latency at times

REASONS

The event loop is busy leads to application hickups.

EXAMPLE

- `async.each` instead of `async.eachLimit`
- `async.map` instead of `async.mapLimit`

Example on a messaging app (adapted):

```
module.exports = function getConversation(req, reply) {
  Conversations.get(req.params.id, function(err, conversatio
    if (err) return reply(err);

    async.map(conversation.participants, UserProfiles.get, d

    function done(err, participants) {
      if (err) return reply(err);
      else reply(...)
    }
  });
}
```

SACRIFICE RESPONSE TIME FOR THE GREATER GOOD AND LIMIT THE CONCURRENCY:

```
module.exports = function getConversation(req, reply) {
  Conversations.get(req.params.id, function(err, conversatio
    if (err) return reply(err);

    async.mapLimit(conversation.participants, 5, UserProfile

    function done(err, participants) {
      if (err) return reply(err);
      else reply(...)
    }
  });
}
```

LARGE DENORMALISED DOCUMENTS

```
{
  _id: ...
  items: [
    {
      itemId: ...,
    }
  ],
  history: [
    ...
  ]
}
```

- Large memory consumption
- Request latency spikes

- Improve the schema
- Stream
- Minimise marshalling

```
module.exports = function findPeople(req, reply) {
  People.
    find(req.params).
    limit(100).
    exec(callback);

  function callback(err, results) {
    reply(err || results);
  }
};
```

MISSING THE OPPORTUNITY OF USING STREAMS

#reliability #performance #maintainability

(22/22)

SYMPTOMS

- Response time bubbles
- High memory consumption

EXAMPLE

Buffering query result set before replying

```
module.exports = function findPeople(req, reply) {
  People.
    find(req.params).
    limit(100).
    exec(callback);

  function callback(err, results) {
    reply(err || results);
  }
};
```

Now, streaming:

```
module.exports = function findPeople(req, reply) {
  var json = JSONStream();
  var peopleStream = People.
    find(req.params).
    stream();

  reply(pump(peopleStream, json));
};
```

CHALLENGES

- Streams API
- Error handling (header is sent before the body)

BENEFITS

- Smaller TTFB (time to first byte)
- Less buffering -> less memory consumed -> smaller / fewer GC pauses

MAIN TAKE-AWAY'S

Node is fundamentally different from the other technologies frequently used in big teams.

Adopting Node also means adopting its newer practices.

More at blog.yld.io

PEDRO TEIXEIRA

@pgte — pedro@yld.io

IGOR SOAREZ

@igorsoarez — igor@yld.io


MODULE PATTERNS
UNDERSTANDING AND USING THE NODE MODULE SYSTEM
FLOW CONTROL
WORK QUEUE PATTERNS
NETWORKING PATTERNS
LEARN HOW TO ESCAPE THE TRAP OF MONOLITHIC APPS USING NODE
DATABASE PATTERNS

THANK YOU!

Q&A