@somkiat

Node.js goal

- Provide easy way to build **scalable network application**

Node.js not

- Another Web framework
- For beginner
- Multi-thread

Node.js is

- Server side JavaScript
- Fun !!!

Node.js why

- **Non Blocking I/O**
- V8 Javascript Engine
- **Single Thread with Event Loop**
- 40,025 modules
- Windows, Linux, Mac
- 1 Language for Frontend and Backend
- Active community

Node.js for

- Web application
- Websocket server
- Ad server
- Proxy server
- Streaming server
- Fast file upload client
- Any Real-time data apps
- Anything with high I/O

Who use Node.js

eBay®

LinkedIn

Microsoft

StrongLoop

nodejitsu

The New York Times

Yahoo!

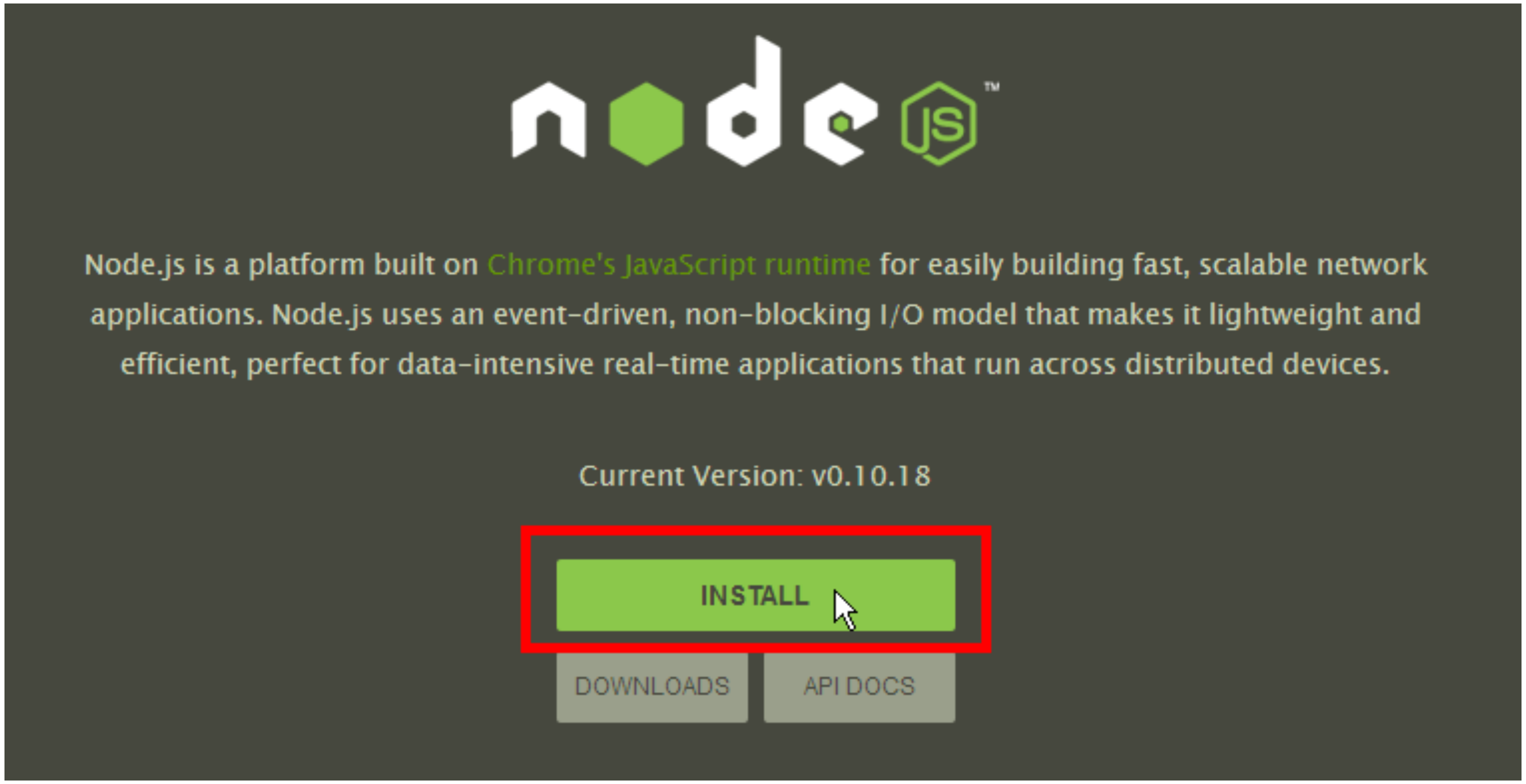
Storify

Cloud9 IDE

http://nodejs.org/industry/

Node.js installation

- Download package from Nodejs.org



Node.js is a platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

Current Version: v0.10.18

Demo :: Hello World

- Create file hello.js

```
console.log( “Hello World” );
```

- Run with Node

```
$node hello.js
>> Hello World
```

Blocking vs Non-Blocking

- Topic :: Read data from file and show data

Blocking

- Read data from file
- Show data
- Do other tasks

```
var data = fs.readFileSync( “test.txt” );
console.log( data );
console.log( “Do other tasks” );
```

http://nodejs.org/api/fs.html

Non-Blocking

Callback

- Read data from file
  - **When read data completed, show data**
- Do other tasks

```
fs.readFile( “test.txt”, function( err, data ) {
    console.log(data);
});
console.log(“Do other tasks”);
```

Callback syntax

```
fs.readFile( “test.txt”, function( err, data ) {
    console.log(data);
});
```

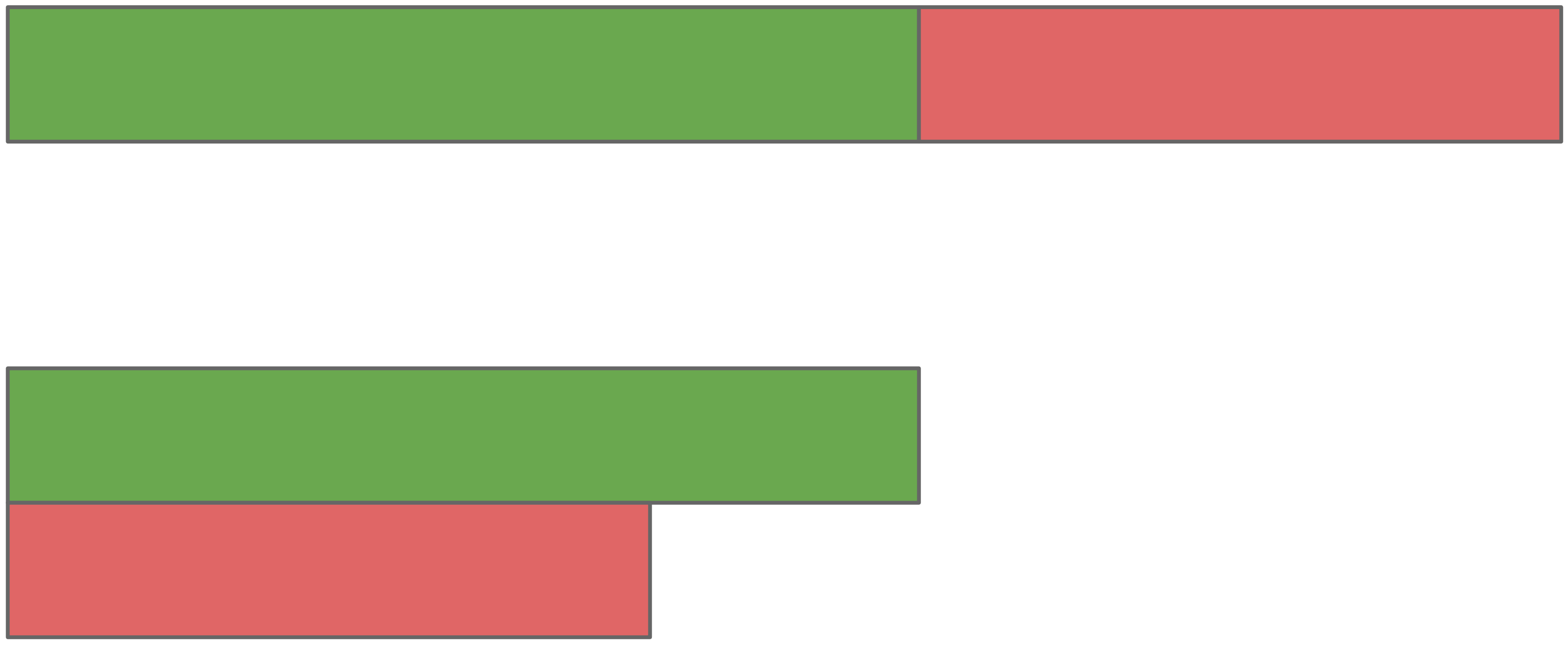
As same as

```
fs.readFile( “test.txt”, callback )
var callback = function( err, data ) {
    console.log(data);
}
```

Blocking vs Non-Blocking

```
var callback = function(err, data) {
    console.log(data);
}
fs.readFile("test.txt", callback);
fs.readFile("test2.txt", callback);
```

Blocking vs Non-Blocking



Recommended modules

- Async
  - https://github.com/caolan/async

- Step
  - https://github.com/creationix/step

Demo :: Basic HTTP

```
#hello_server_01.js

var http = require('http');

http.createServer( function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World\n');
}).listen(1337, '127.0.0.1');

console.log('Server running at http://127.0.0.1:1337');
```

Demo :: Basic HTTP

```
$node  hello_server_01.js
>Server running at http://127.0.0.1:1337/
```

Check result from browser http://127.0.0.1:1337

Demo :: Basic HTTP (Refactor)

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

var server = http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World\n');
});

server.listen(1337, ‘127.0.0.1’);

console.log('Server running at http://127.0.0.1:1337/');
```

Event Loop ?

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

var server = http.createServer(function (req, res) {
    ….
});
server.listen(1337, ‘127.0.0.1’);
```

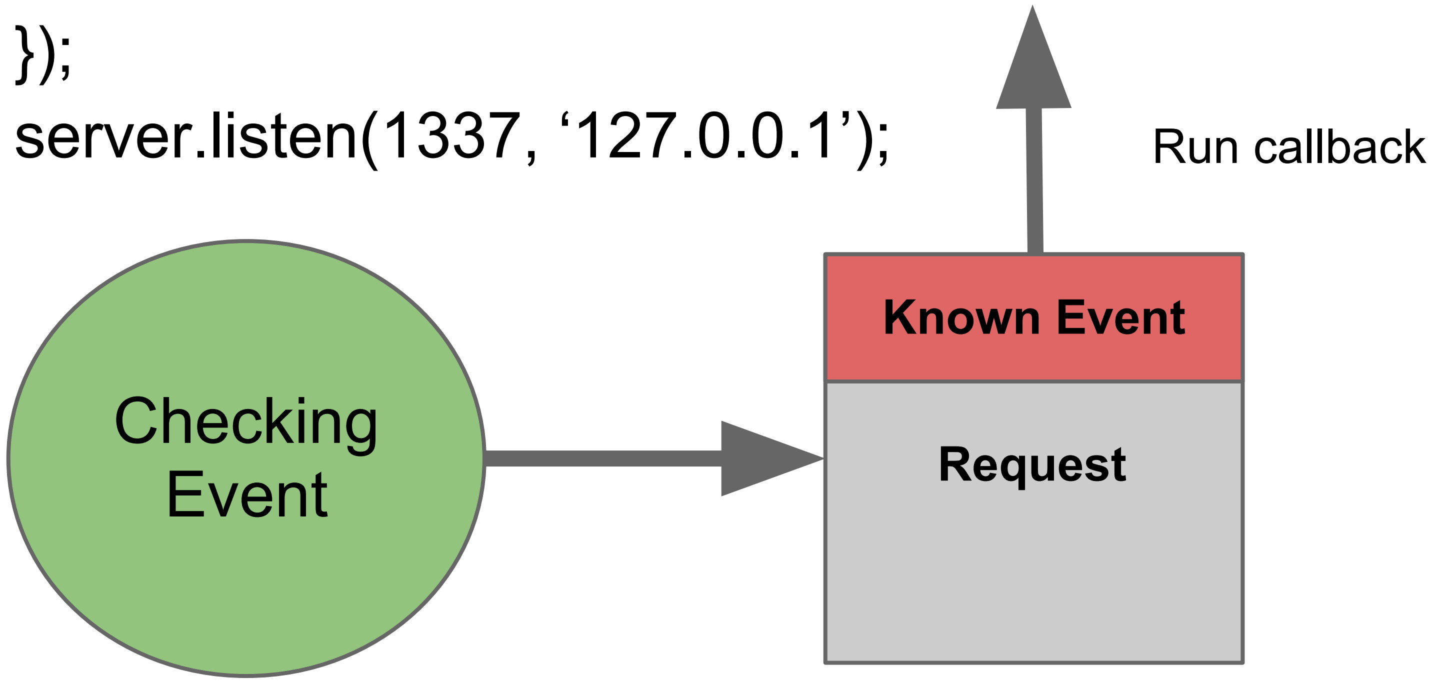
Event Loop ?

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

var server = http.createServer(function (req, res) {
    ….
});
server.listen(1337, ‘127.0.0.1’);
```

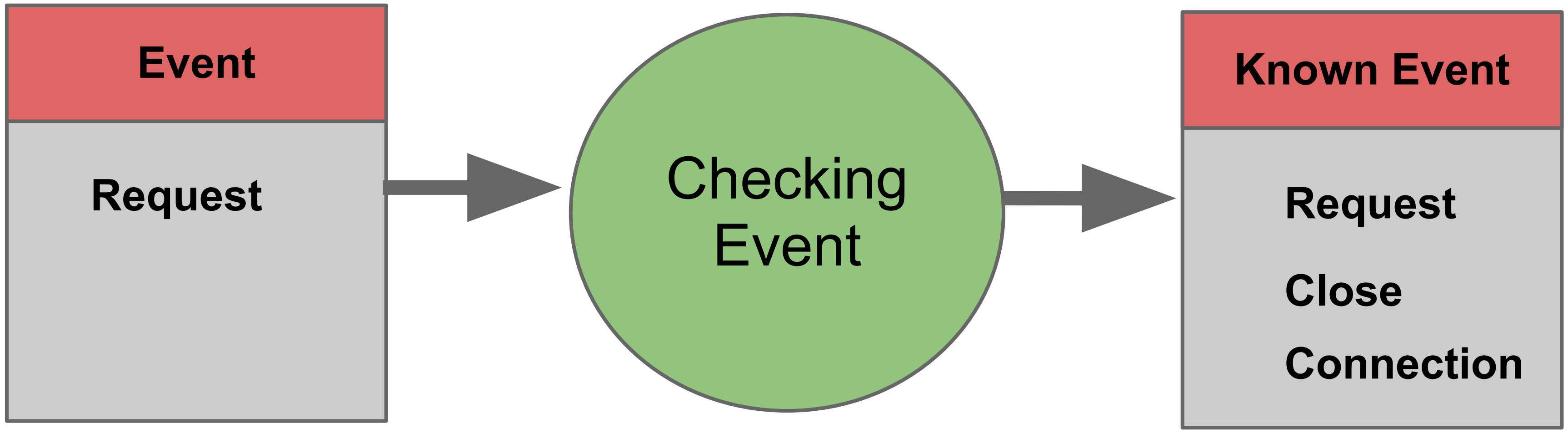
Start Event Loop

Event Loop ?

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

var server = http.createServer(function (req, res) {
    ….
});
server.listen(1337, ‘127.0.0.1’);
```



Event Loop


Event
Request
Checking Event
Known Event
Request
Close
Connection

Handle Event

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

var server = http.createServer();
server.on('request', function(req, res){
    res.write('Got request\n');
    res.end();
});

server.listen(1337, ‘127.0.0.1’);
```

http://nodejs.org/api/http.html

Demo :: Echo server

```
var server = http.createServer( function(req, res) {
    res.writeHead(200);

    req.on('data', function(data) {
      res.write(data);
    });

    req.on('end', function(){
      res.end();
    });
});
```

Demo :: Echo server

```
$curl -d "somkiat" http://localhost:1337
```

Demo :: Echo server

```
var server = http.createServer( function(req, res) {
  res.writeHead(200);

  req.on('data', function(data) {
    res.write(data);
  });

  req.on('end', function(){
    res.end();
  });
});
```

Pipe data from request to response

Demo :: Echo server + Pipe

```
var server = http.createServer( function(req, res) {
    res.writeHead(200);
    req.pipe(res);
});
```

Demo :: Upload file

```
http.createServer(function(req, res) {

  var newFile = fs.createWriteStream("uploaded.txt");
  req.pipe(newFile);

  req.on('end', function() {
    res.end('uploaded!');
  });

}).listen(1337);
```

Create file on server and pipe data to file

Demo :: Upload file

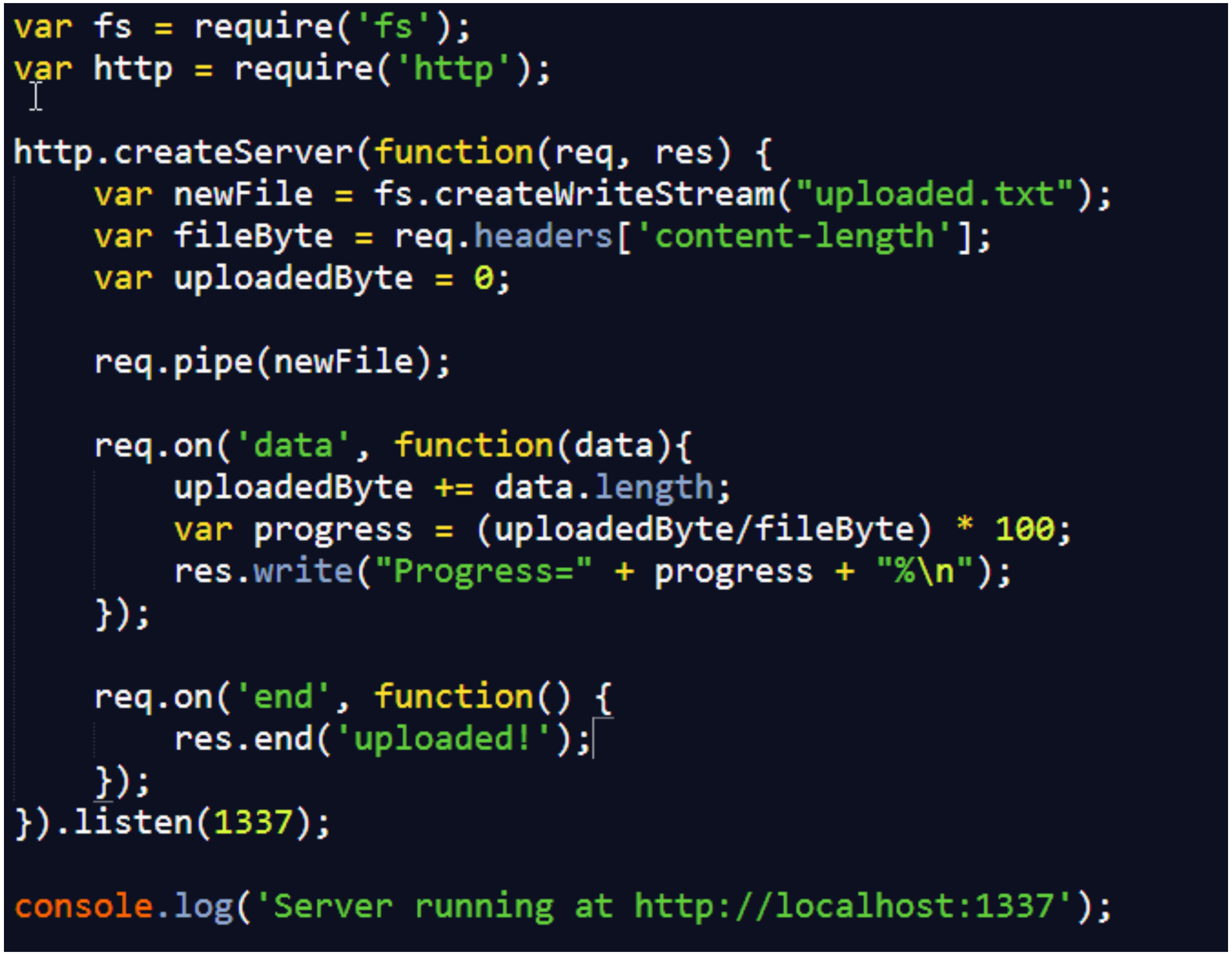
```
http.createServer(function(req, res) {

    var newFile = fs.createWriteStream("uploaded.txt");
    req.pipe(newFile);

    req.on('end', function() {
        res.end('uploaded!');
    });

}).listen(1337);
```

Handle end event on request

Demo :: Upload file

```
$curl --upload-file test.txt http://localhost:1337
```

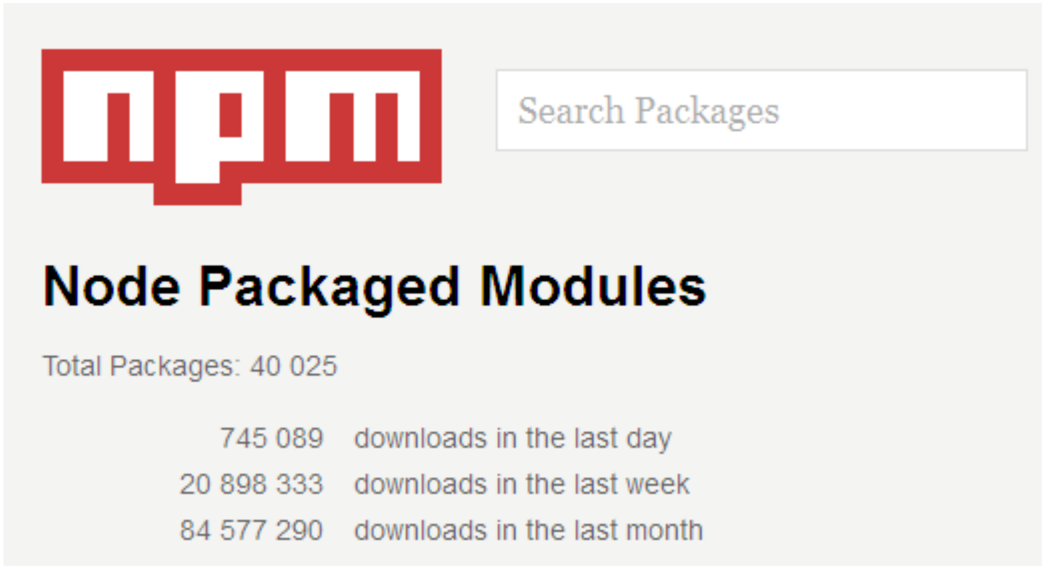
Demo :: Upload file with progress



```
var fs = require('fs');
var http = require('http');

http.createServer(function(req, res) {
    var newFile = fs.createWriteStream("uploaded.txt");
    var fileByte = req.headers['content-length'];
    var uploadedByte = 0;

    req.pipe(newFile);

    req.on('data', function(data){
        uploadedByte += data.length;
        var progress = (uploadedByte/fileByte) * 100;
        res.write("Progress=" + progress + "%\n");
    });

    req.on('end', function() {
        res.end('uploaded!');
    });
}).listen(1337);

console.log('Server running at http://localhost:1337');
```

Node.js Modules

- https://npmjs.org/
- # of modules = 40,025



Node Packaged Modules

Total Packages: 40 025

```
   745 089   downloads in the last day
20 898 333   downloads in the last week
84 577 290   downloads in the last month
```

Install module

```
$npm install <module name>
```

Using module

```
var http = require(‘http’);
var fs = require(‘fs’);
var express = require(‘express’);
```

Working with Express

- http://expressjs.com

express 3.0.0
web
application
framework for
node

Working with Express

- Inspire from Sinatra
- Fast
- Flexible
- Simple

Installation express

```
$npm install express
```

Demo :: Express

```
var express = require('express');
var app = express();

app.get('/', function (req, res) {
    res.setHeader('Content-Type', 'text/plain');
    res.end('Hello, world!');
});

app.listen(1337);
console.log('Listening on port 1337');
```

Demo :: Manage package

```
$npm init

$npm info express version
```

Demo :: package.json

```
{
  "name": "hello-world",
  "description": "hello world test app",
  "version": "0.0.1",
  "private": true,
  "dependencies": {
    "express": "3.3.x"
  }
}
```

Demo :: Install and run

```
$npm install

$node http_express.js
```

Develop REST API

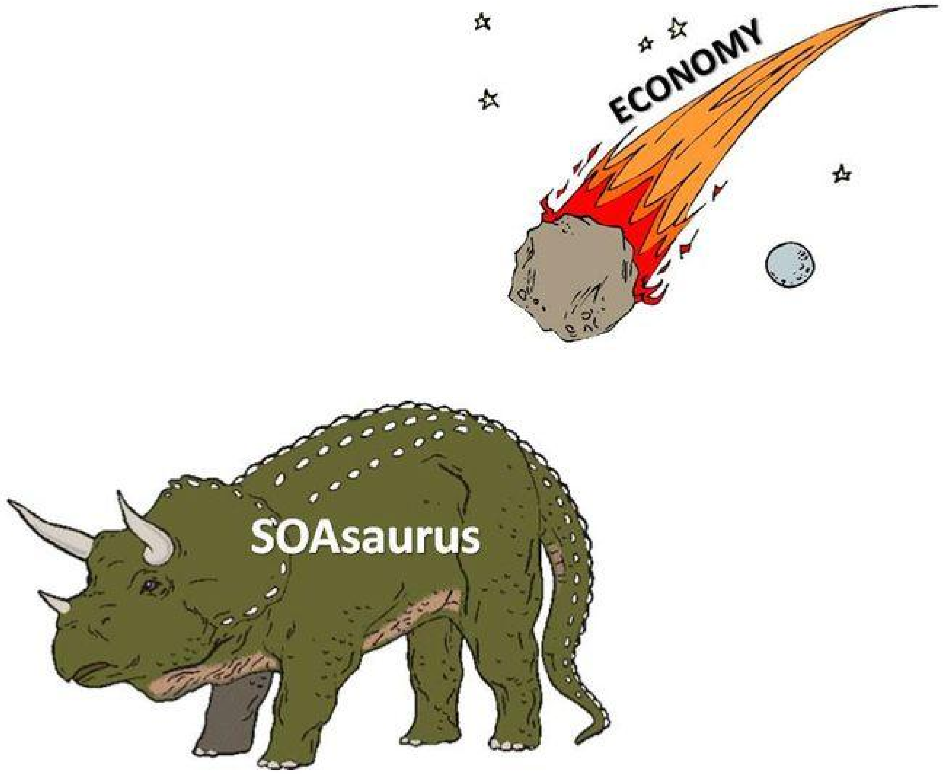
- REST = REpresentational State Transfer
- Not new technology
- Architectural style for client-server

http://en.wikipedia.org/wiki/Representational_state_transfer

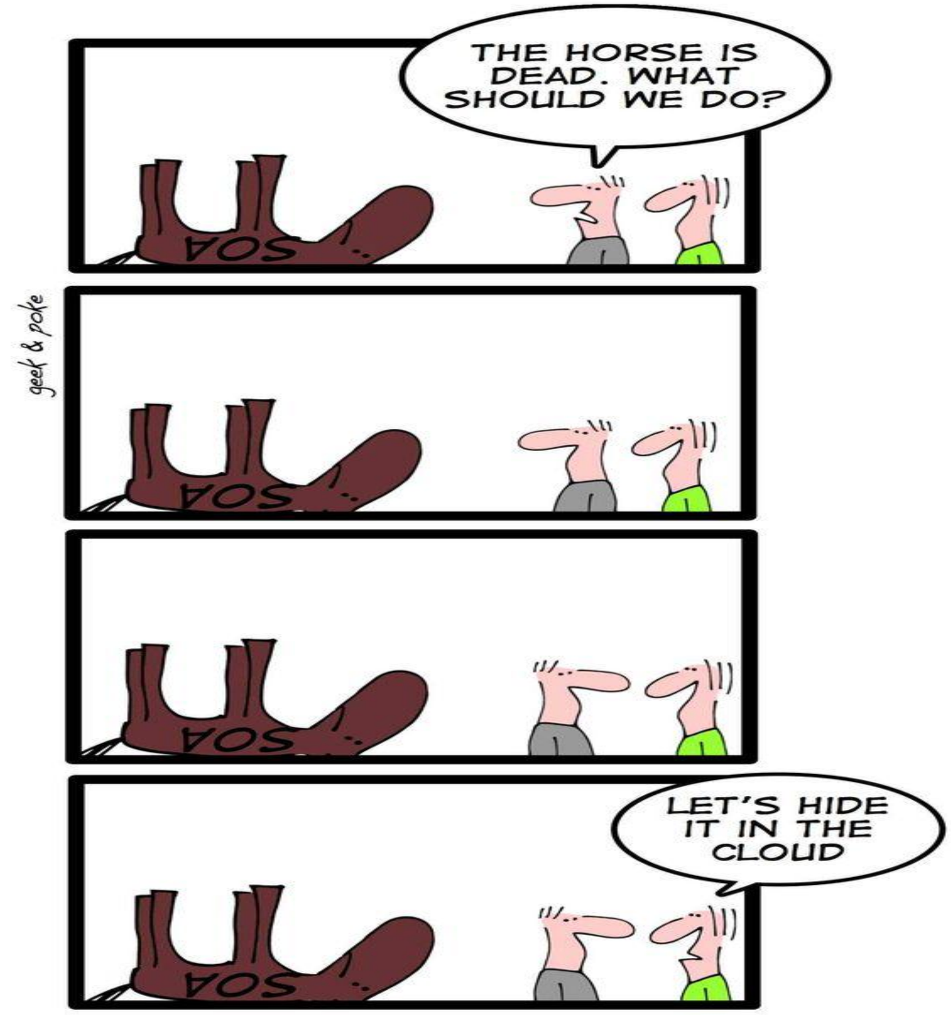
Goals of REST

- General interface
- Scalability of component interface
- Reduce latency
- Encapsulate legacy system

SOA is DEAD!!



SOA is DEAD !!!


THE HORSE IS DEAD. WHAT SHOULD WE DO?
SOA
geek & poke
LET'S HIDE IT IN THE CLOUD

http://www.zdnet.com/blog/service-oriented/you-cant-beat-the-dead-soa-horse/3708

HTTP Method

- GET
- POST
- PUT
- DELETE

HTTP Method with CRUD

- POST => **C**reate
- GET => **R**ead
- PUT => **U**pdate
- DELETE => **D**elete

Demo :: REST with JSON

```
app.get('/', function (req, res) {
  res.json(persons);
});

app.post('/', function (req, res) {
  res.json(persons);
});
```

Demo :: REST with JSON

```
app.put('/', function (req, res) {
  res.json(persons);
});

app.delete('/', function (req, res) {
  res.json(persons);
});
```

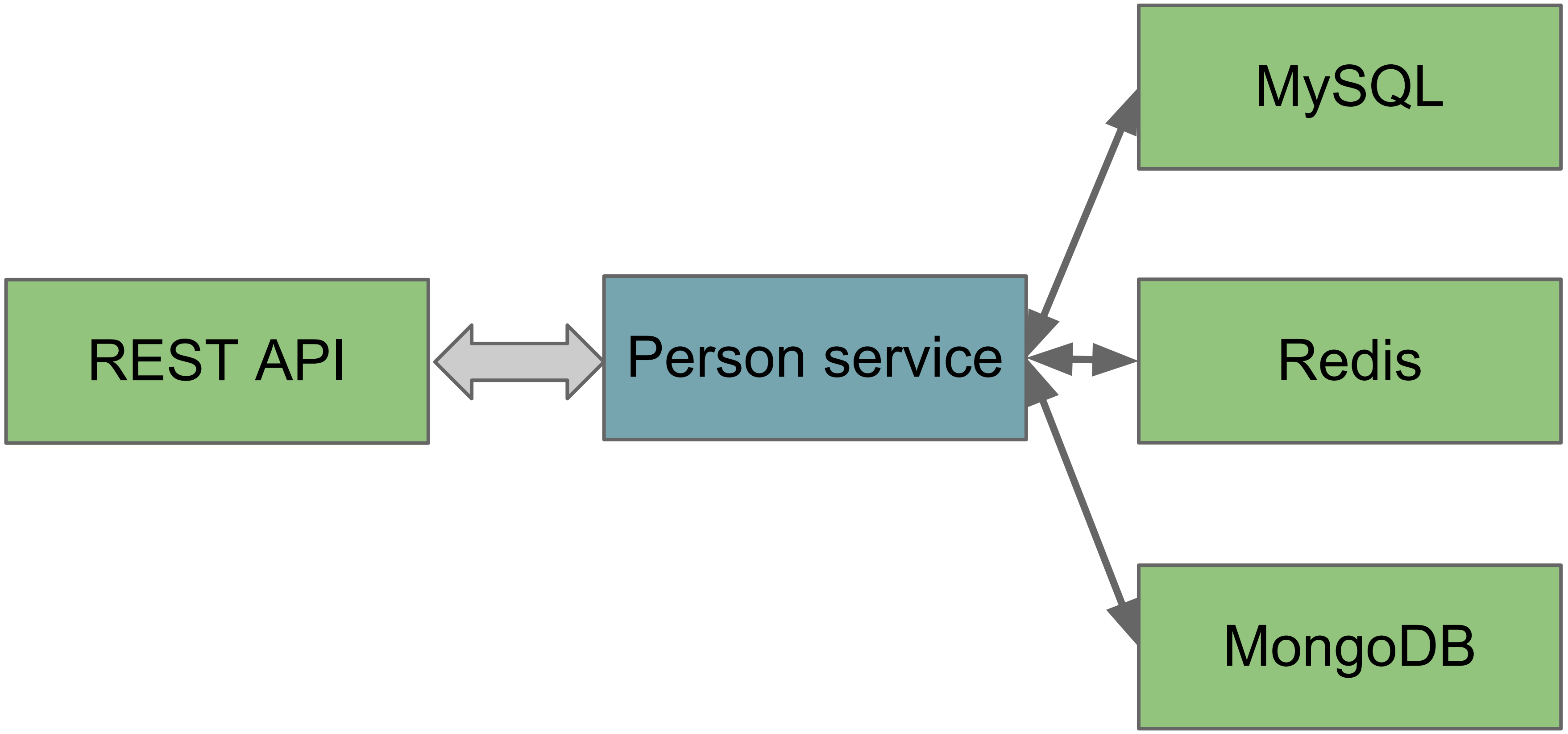
Demo :: Refactoring

```
app.get('/', callback);


var callback = function getData(req, res) {
  res.json(persons);
}
```

Working with Persistence

- MySQL
- MongoDB
- Redis

Working with Persistence



REST API

Method	URL	Action
GET	/	Get all person
GET	/person/3	Get person by id=3
POST	/person	Add new person
PUT	/person	Update person by id
DELETE	/person/3	Delete person by id=3

Demo :: REST API

```
var express = require('express');
var service_person = require('./service_person')
var app = express();

app.get('/', service_person.all);
app.get('/person/:id', service_person.one);
app.post('/person', service_person.insert);
app.put('/person', service_person.update);
app.get('/person/:id', service_person.delete);

app.listen(process.env.PORT || 1337);
```

Working with MySQL

- RDBMS
- Module => see in npmjs.org (a lot !!!!)
  - "mysql": "2.0.0-alpha9"

Design Table

- Table name = person
  - Column
    - id
    - name
    - gender

Demo :: Connect to MySQL

```
var mysql = require('mysql');
var connection = mysql.createConnection(
    {
        host: <db server>,
        user: <username>,
        password: <password>,
        database: <database name>
    }
);
```

Demo :: Retrieve all data

```
connection.query('select * from person',
    function(err, rows, fields) {
            res.contentType('application/json');
            res.write(JSON.stringify(rows));
            res.end();
    }
);
```

Demo :: Retrieve data with criteria

```
var sql = 'select * from person where id=?';

connection.query( sql, [id],
    function(err, rows, fields) {
            res.contentType('application/json');
            res.write(JSON.stringify(rows));
            res.end();
    }
);
```

Demo :: Create new data

```
var sql = 'insert into person (name, gender)
values (?, ?)
';

connection.query( sql, [name, gender],
    function(err, rows, fields) {
            res.json(true);
    }
);
```

Working with MongoDB

- NoSQL
- Document-oriented stoarage
- Keep data in BSON format
- http://www.mongodb.org/
- Module => see in npmjs.org (a lot !!!!)
    - "redis": "0.8.4"

http://en.wikipedia.org/wiki/BSON

Start MongoDB server

```
$mongod.exe --dbpath /some/data/path
```

Demo :: Connect to MongoDB

```
var mongo = require('mongodb');
var Server = mongo.Server,
var server = new Server(
    'localhost',
    27017,
    {auto_reconnect: true}
);
db = new Db('persons', server);
```

Demo :: Connect to MongoDB

```
db.open(function(err, db) {
    if(!err) {
        db.collection('persons', {strict:true},
            function(err, collection) {
                if (err) {
                    populateDB();
                }
            });
}
```

Demo :: Retrieve all data

```
exports.all = function(req, res){
    db.collection('persons', function(err, collection) {
        collection.find().toArray(function(err, persons) {
            res.send(persons);
        });
    });
};
```

Demo :: Retrieve data by id

```
exports.one = function(req, res){
    var personId = req.params.id;
    db.collection('persons', function(err, collection) {
        collection.findOne(
            {'_id':new BSON.ObjectID(personId)},
                function(err, person) {
                    res.send(person);
                });
        });
};
```

Demo :: Create new data

```
exports.insert = function(req, res){
    db.collection('persons', function(err, collection) {
        collection.insert( person, {safe:true},
            function(err, result) {
            });
    });
});
```

Demo :: Update data

```
exports.insert = function(req, res){
    db.collection('persons', function(err, collection) {
        collection.update( {
                '_id':new BSON.ObjectID( personId )},
                updatePerson,
                {safe:true},
            function(err, result) {
            });
    });
});
```

Demo :: Delete data by id

```
exports.insert = function(req, res){
    db.collection('persons', function(err, collection) {
        collection.remove( {
                '_id':new BSON.ObjectID( personId )},
                {safe:true},
            function(err, result) {
            });
    });
});
```

Design Document

- Collection = persons
- Document structure
  - name
  - gender

Working with Redis

- NoSQL
- Key-value data store
- http://redis.io
- Module => see in npmjs.org (a lot !!!!)
  - "redis": "0.8.4"

Install Redis

- Download from http://redis.io/

- For Windows OS
  - https://github.com/dmajkic/redis/downloads

Start Redis server

```
$redis-server
```

Let’s fun with Redis

```
$redis-cli
```

Design Key-Value

- Key = person_list
  - type = List
  - value = id
- Key = id
  - type = Hash
    - id = <id>
    - name = <name>
    - gender = <gender>

Demo :: Connect to Redis

```
var redis = require('redis');
var connection = redis.createClient(
    {
        host: 'localhost',
        port: '6379'
    }
);
```

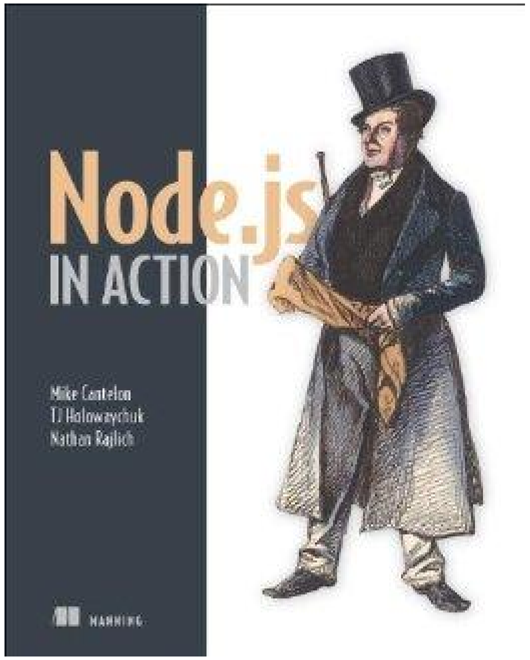
Favorite Modules

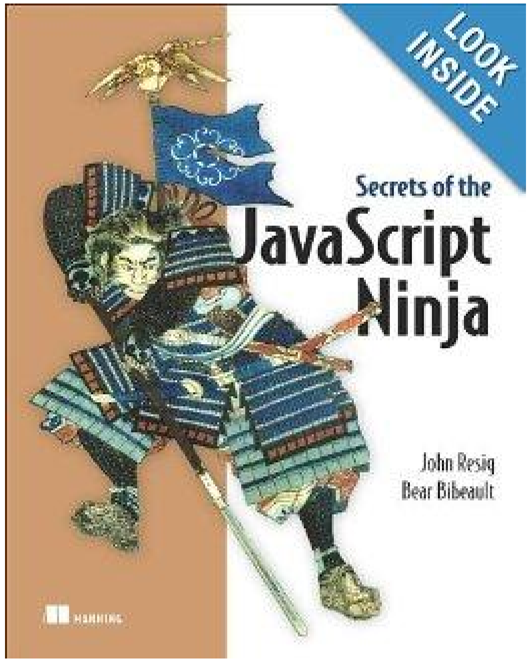
- express
- underscore
- request
- async
- mysql
- *Find more in npmjs.org*

Another project like Node.js

- Vert.x => Polygot programming
- Akka   => Scala and Java
- Tornado => Python
- Libevent => C
- EventMachine => Ruby

Book


Node.js
IN ACTION
Mike Cantelon
TJ Holowaychuk
Nathan Rajlich
MANNING


LOOK INSIDE
Secrets of the
JavaScript
Ninja
John Resig
Bear Bibeault
MANNING

Resources

- Sourcecode for demo https://github.com/up1/demo_nodejs

- https://npmjs.org
- http://nodejs.org/

- http://callbackhell.com/
- http://nodeframework.com/

Q/A