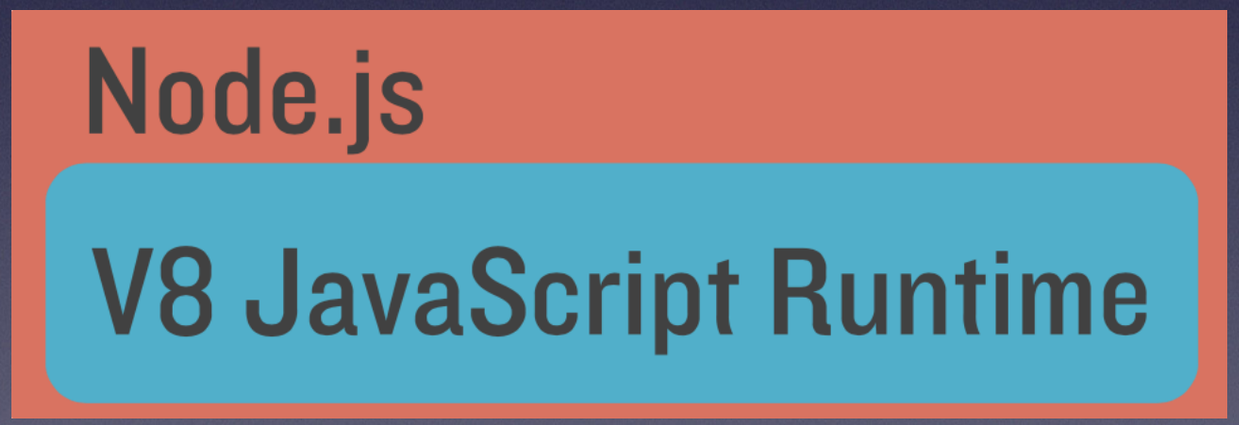
# Node.js

Roli Butron

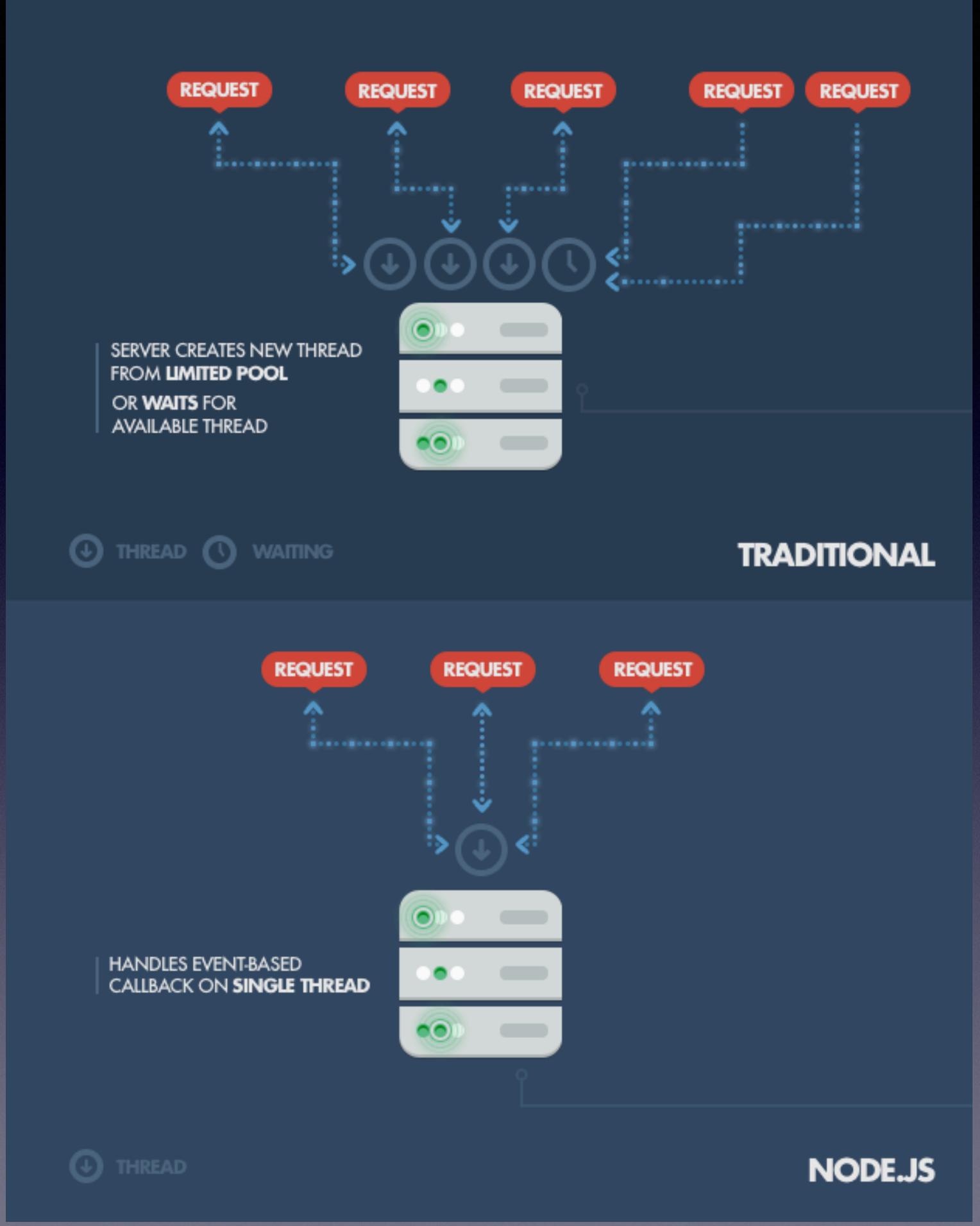
# Que es Node.js?

Es javascript de lado del servidor, el cual te permite desarrollar aplicaciones de red altamente escalables



# What is Node?

- Javascript Runtime build on Chrome's V8 Javascript Engine
- Javascript running in the server
- Used to be Powerful, Fast and Scalable Web Applications
- Uses Event-driven, Non blocking I/O model




8 GB Ram

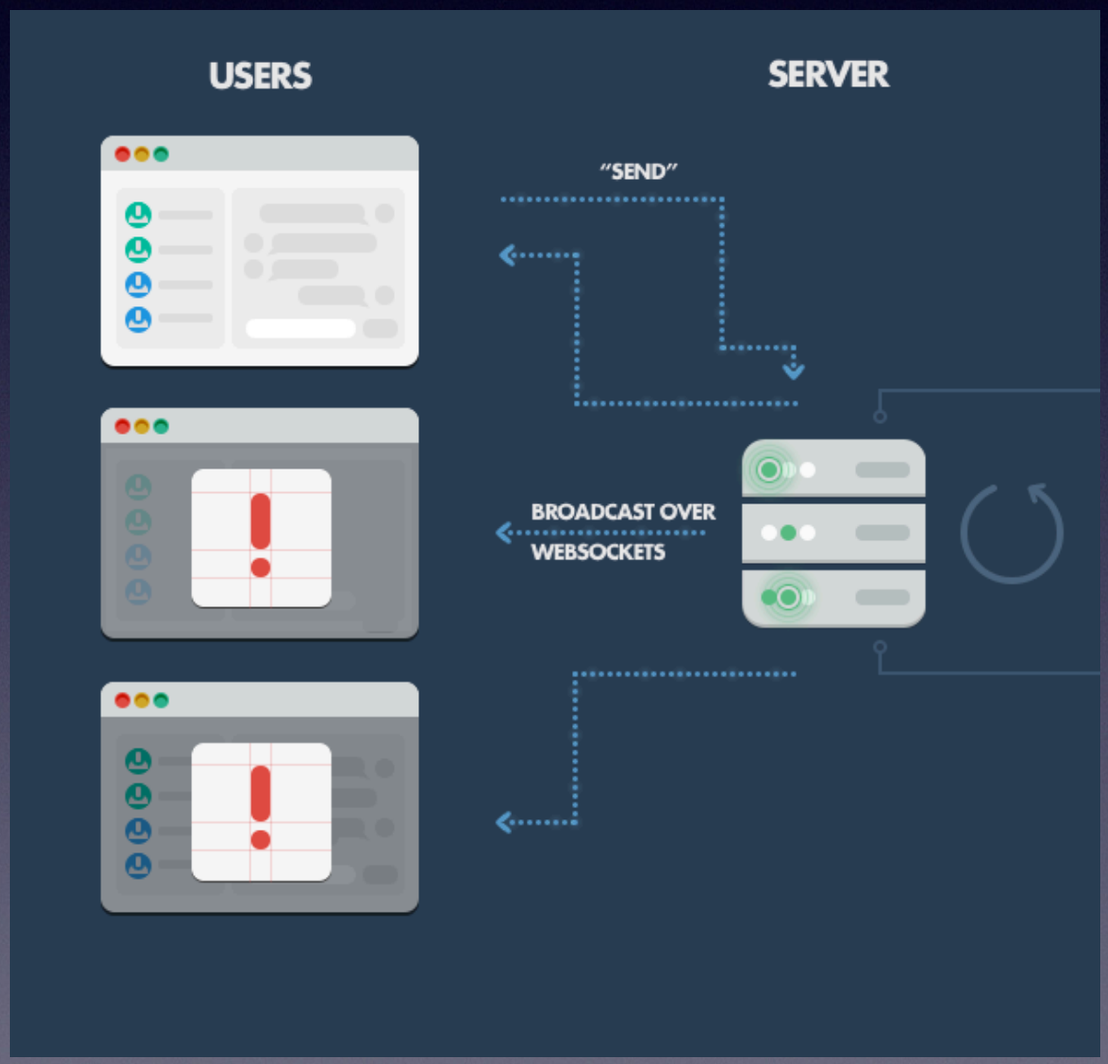
2 MB Ram

4000 Conexiones

1M Conexiones

# Que tipo de applications puedo desarrollar ?

- Websocket Server (chat)
- Fast File upload client
- Any Realtime data apps.
- Rest API.




## APIs

The “glue” that connects devices and browsers to data and services


## Mobile

Backends and full-stack JavaScript hybrid apps

## Web

Servers and single-page apps

## Internet of Things

Network connected embedded devices and sensors

# Que no es Node.js ?

- A web framework
- Para principiantes
- Multi-hilos

# Java Sleep

```
System.out.println("Step: 1");
System.out.println("Step: 2");
Thread.sleep(1000);
System.out.println("Step: 3");
```

# Node "Sleep"

```
console.log('Step: 1')
setTimeout(function () {
  console.log('Step: 3')
}, 1000)
console.log('Step: 2')
```

# Non block I/O

- Trabajo en uno solo hilo usando non block IO calls
- Soporta miles de conexiones concurrentes
- Optimiza el rendimiento y la escalabilidad en aplicaciones web con muchas operaciones de I/ O

# Ejemplo

- **Blocking Code**

  Leer un archivo  y almacenar en variable contenido

  Imprimir contenido del archivo

  Hacer otras cosas

- **Non Blocking Code**

  Leer un archivo  y almacenar en variable contenido

  Imprimir contenido del archivo

  Hacer otras cosas

```
1  console.log('inicia syncrono');
2  var fs = require("fs");
3  var contenido_archivo = fs.readFileSync('file.txt','utf8');
4  console.log(contenido_archivo);
5  console.log("termina");
6
```

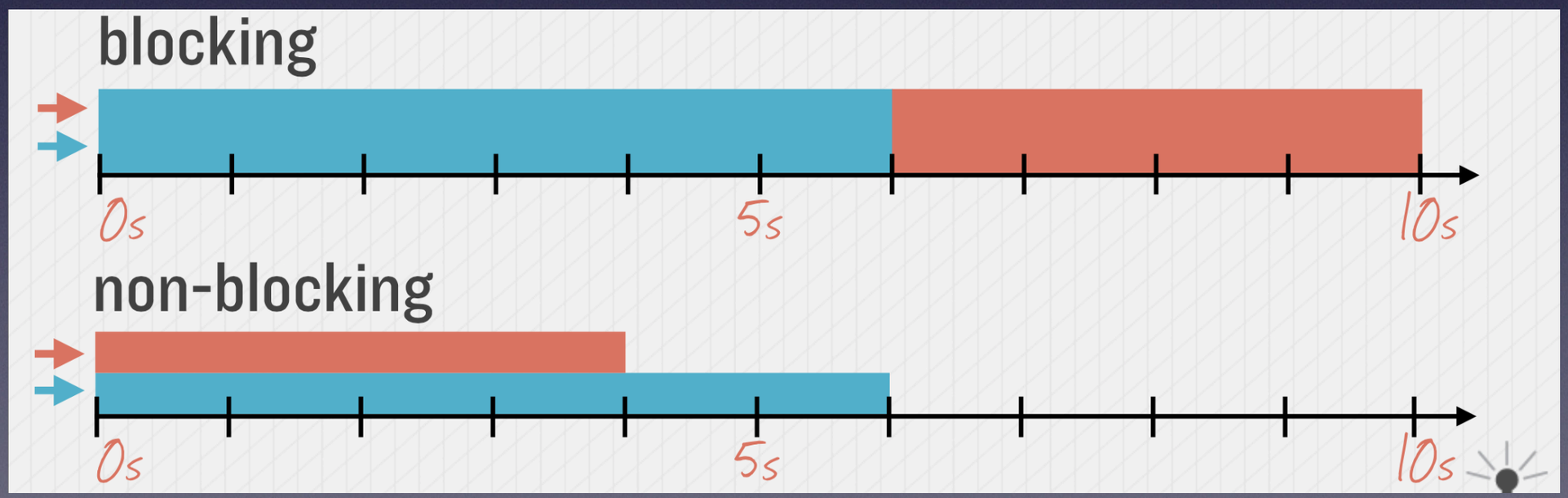
```
MacBook-Pro-de-Roli:dia1 rolibb$ node syncrono_file.js
inicia syncrono
Curso de nodejs abriendo un file txt

termina
MacBook-Pro-de-Roli:dia1 rolibb$
```

```
1  console.log('inicia asyncrono');
2
3  var fs = require("fs");
4  fs.readFile('file.txt','utf8',function(err,data){
5      if(!err) {
6          console.log(data);
7      }
8  });
9  console.log("termina");
10
```

```
MacBook-Pro-de-Roli:dia1 rolibb$ node asyncronous_file.js
inicia asyncrono
termina
Curso de nodejs abriendo un file txt

MacBook-Pro-de-Roli:dia1 rolibb$
```

```
1   console.log('inicia asyncrono');
2
3   const callback = function(err,data){
4       if(!err) {
5           console.log(data);
6       }
7   };
8
9   var fs = require("fs");
10  fs.readFile('file.txt','utf8', callback);
11
12  console.log("termina");
13
```

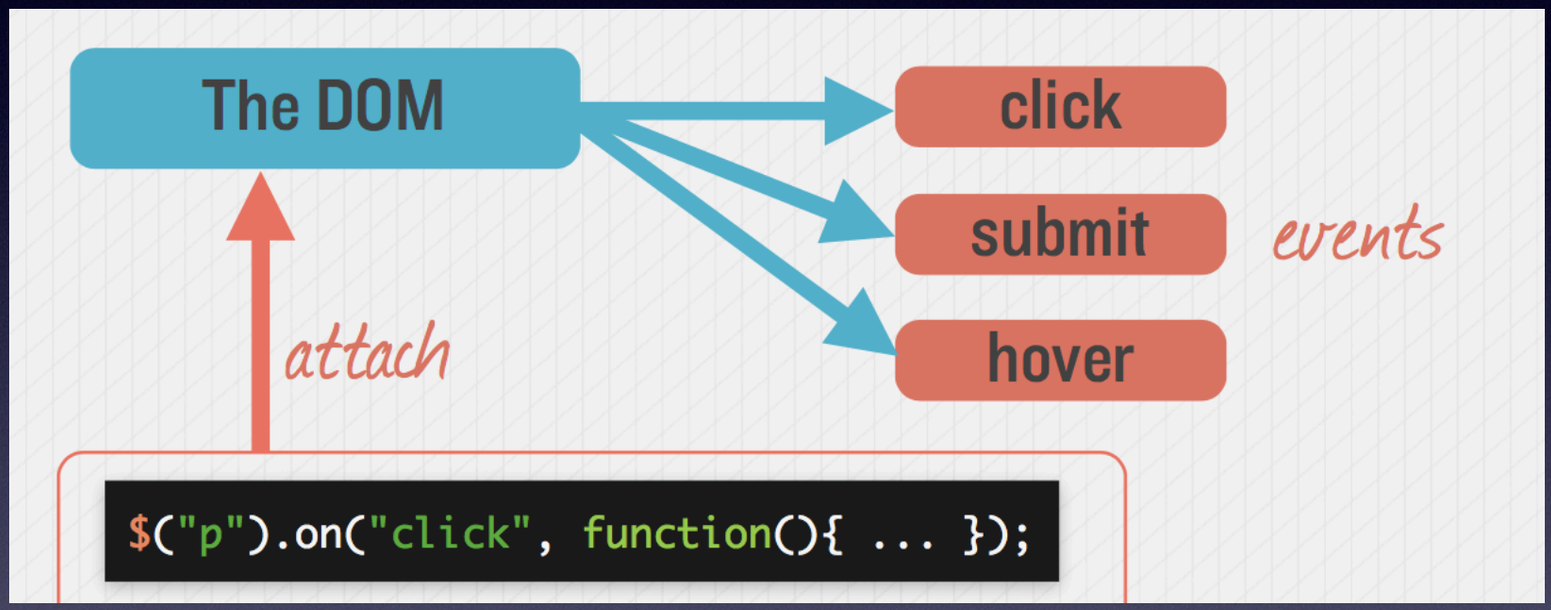
```
1   console.log('inicia asyncrono');
2
3   const callback = function(err,data){
4       if(!err) {
5           console.log(data);
6       }
7   };
8
9   var fs = require("fs");
10  fs.readFile('file.txt','utf8', callback);
11  fs.readFile('file_2.txt','utf8', callback);
12
13  console.log("termina");
14
```



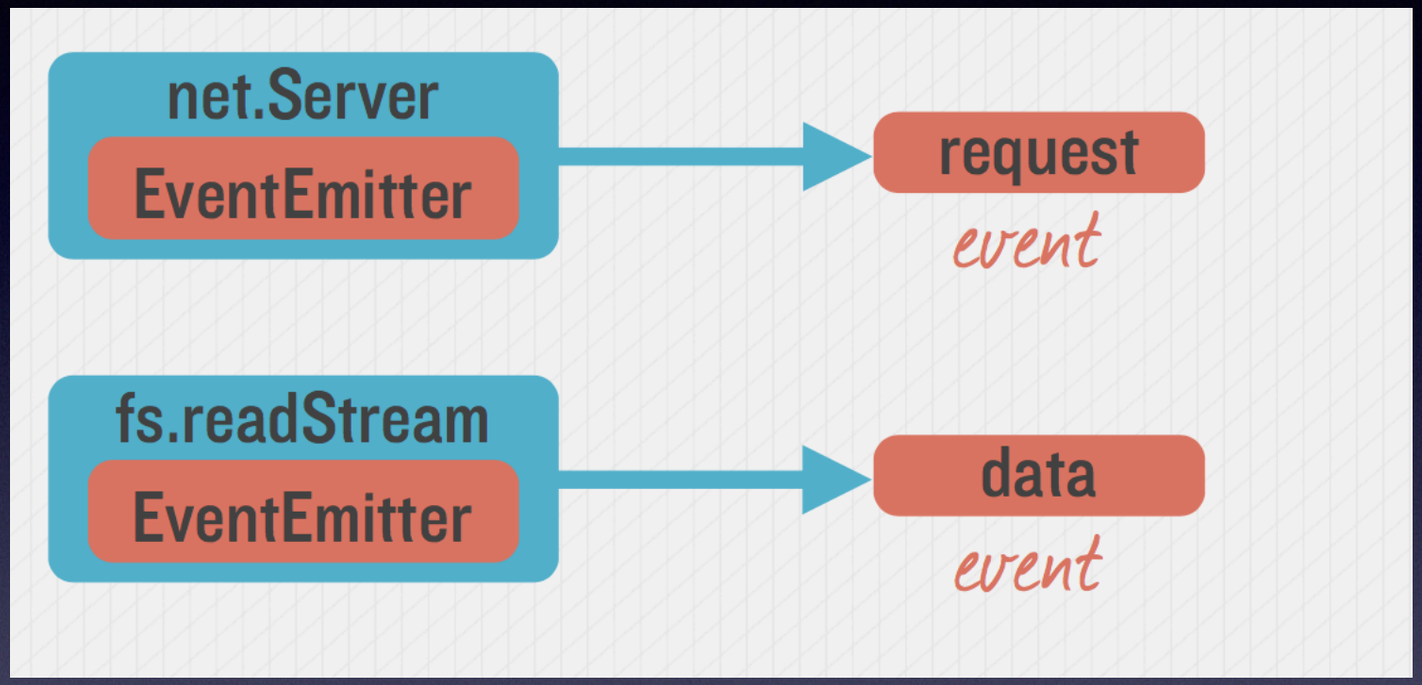
# Hola Mundo

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

http.createServer(function(req, res){
    res.writeHead(200);
    res.write("Hola mundo Node.js");
    res.end();
}).listen(8080);

```

# Eventos en el navegador web



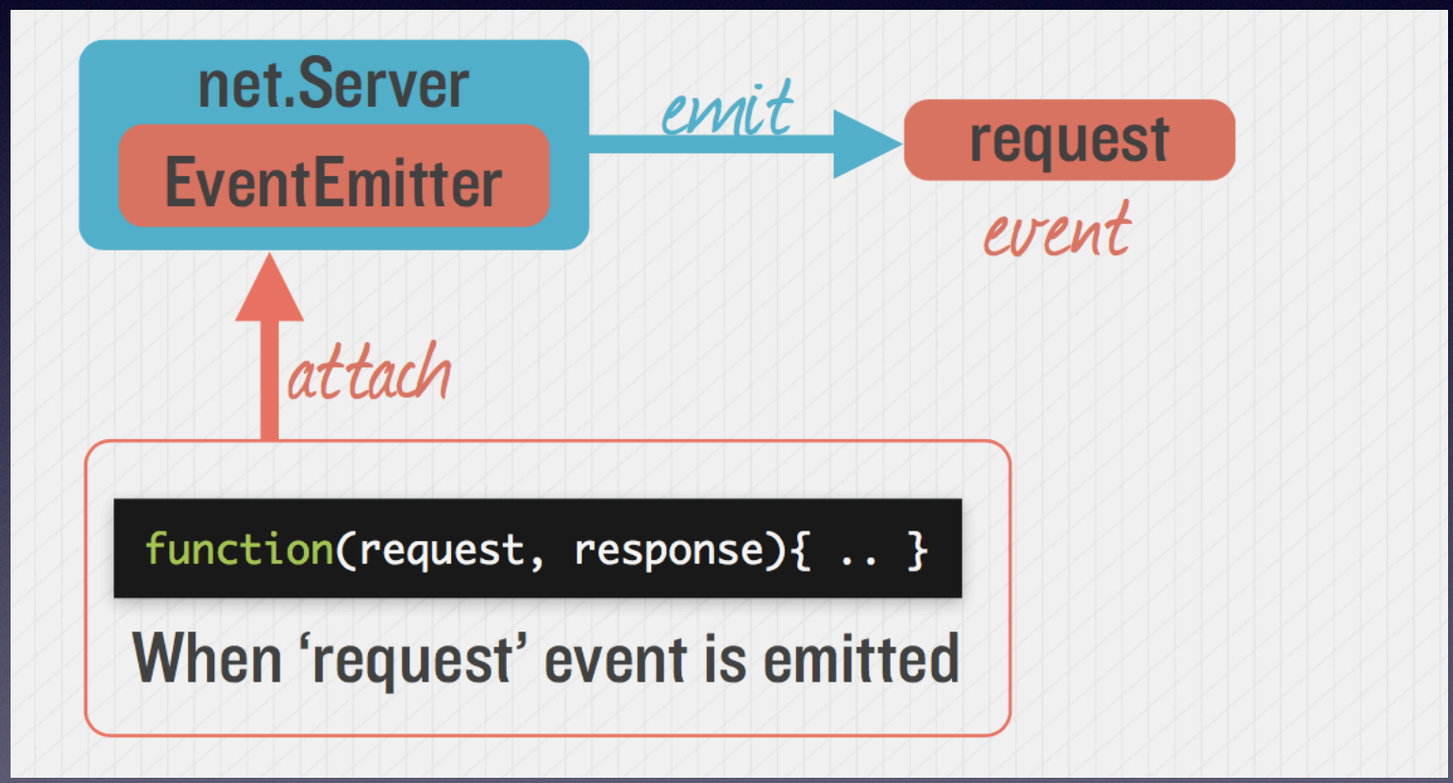
# Eventos en node



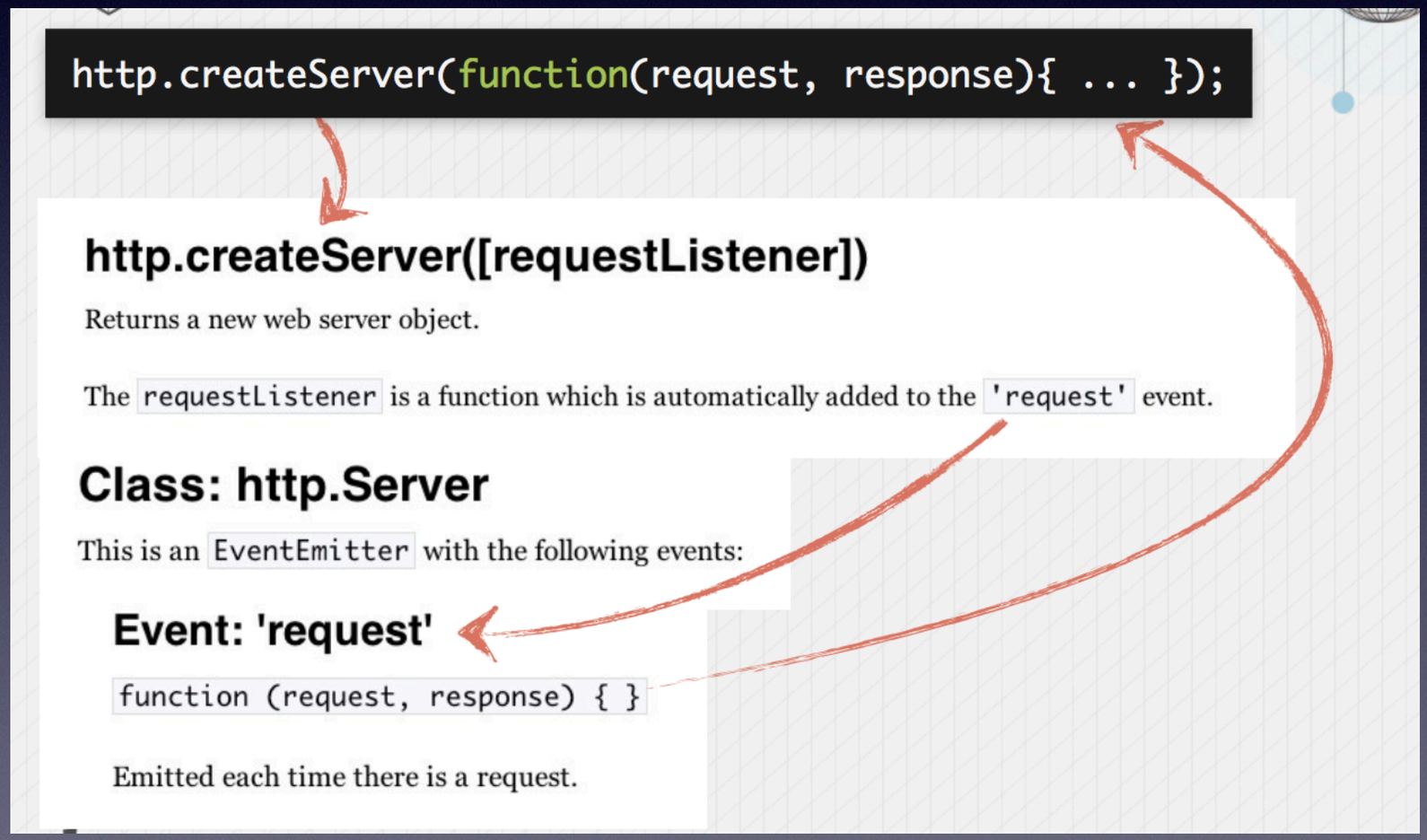
# Custom events

```
1   var EventEmitter = require('events').EventEmitter;
2   var logger = new EventEmitter();
3
4   logger.on('error', function(mensaje){
5       console.log('Err: '+ mensaje);
6   });
7
8   logger.emit('error', 'al abrir archivo');
9   logger.emit('error', 'conexion de red');
10
```

# Eventos en node



# Documentation



# Sintaxis Alternativa

```
http.createServer(function(request, response){ ... });
```

*Same as*

```
var server = http.createServer();
server.on('request', function(request, response){ ... });
```

*This is how we add add event listeners*

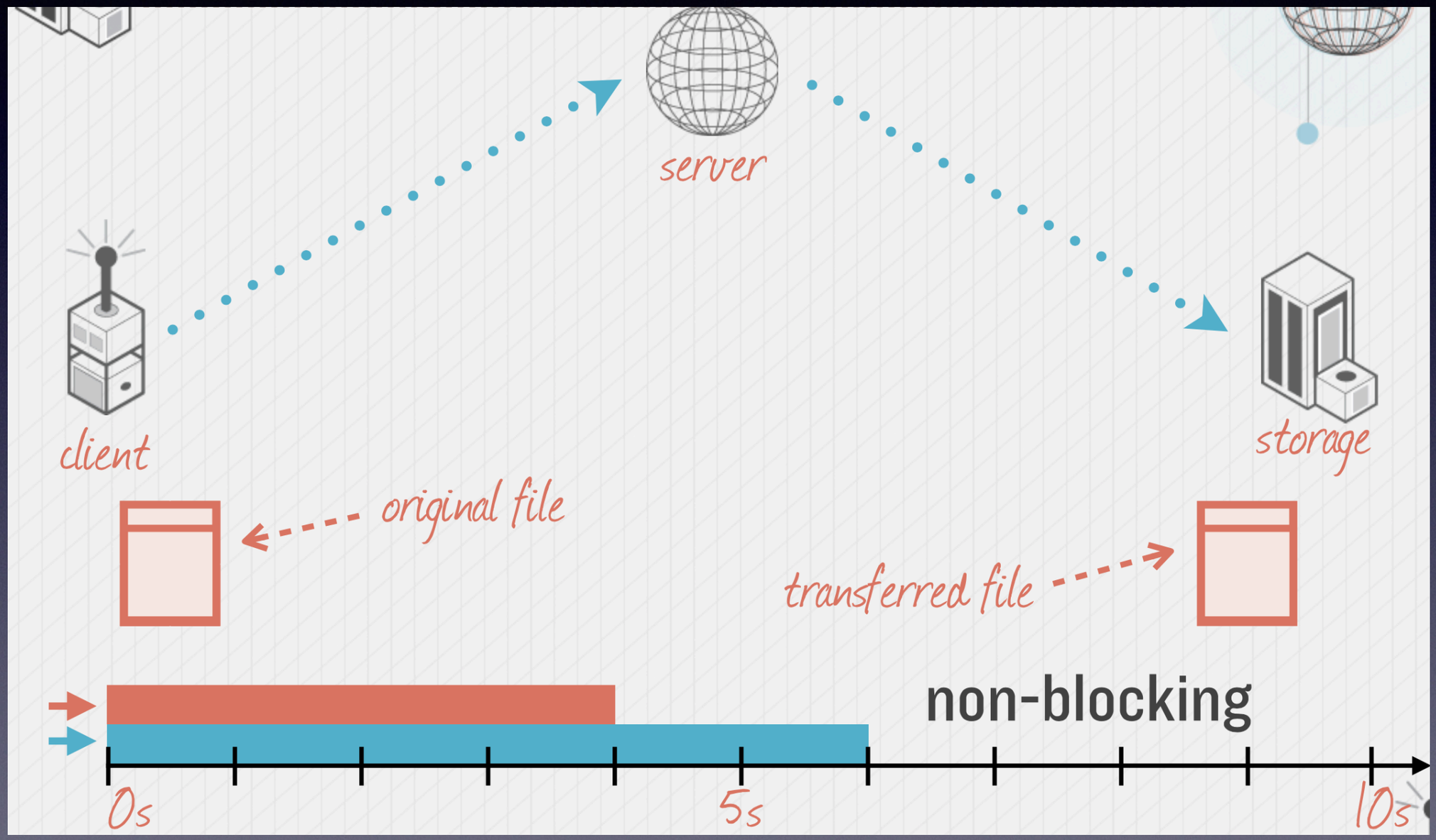
**Event: 'close'**

```
function () { }
```

Emitted when the server closes.

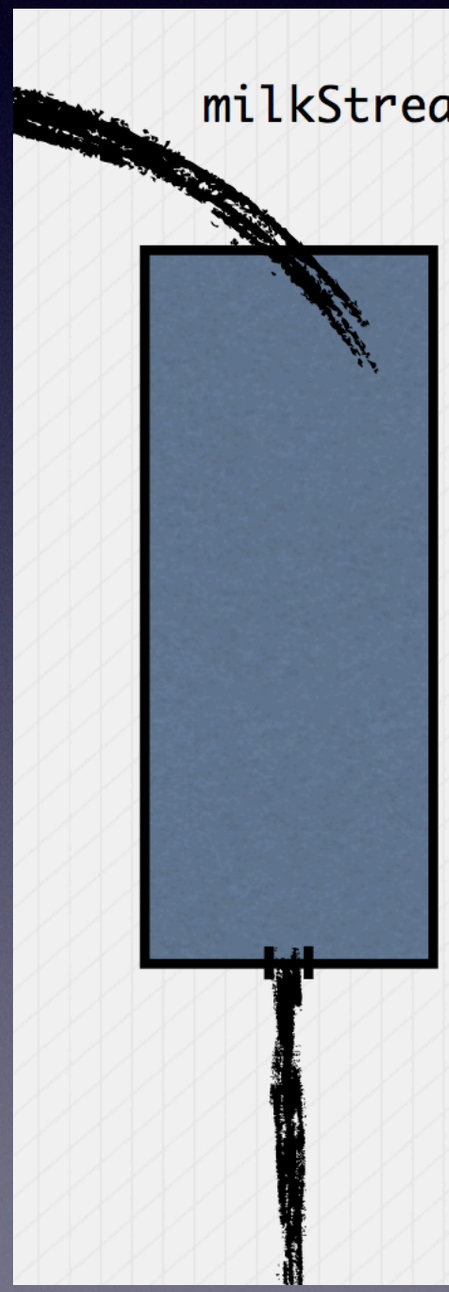
```
server.on('close', function(){ ... });
```

# Streams


server
client
storage
original file
transferred file
non-blocking
0s
5s
10s

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

http.createServer(function(req, res){
    var nuevo_archivo = fs.createWriteStream('file_uploaded.pdf');
    req.pipe(nuevo_archivo);
    req.on('end', function(){
        res.end('uploaded');
    });
}).listen(8080);
```

```
$ curl --upload-file readme.md http://localhost:8080
```



```
milkStream.pause();

Once milk jug is drained
    milkStream.resume();
});
```

## Pause when writeStream is full

```
readStream.on('data', function(chunk) {
  var buffer_good = writeStream.write(chunk);
  if (!buffer_good) readStream.pause();
});
```

returns false if kernel buffer full

## Resume when ready to write again

```
writeStream.on('drain', function(){
  readStream.resume();
});
```

## All encapsulated in

```
readStream.pipe(writeStream);
```

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

http.createServer(function(req, res){
    var nuevo_archivo = fs.createWriteStream('file_uploaded_progress.pdf');
    var totalBytes = req.headers['content-length'];
    var uploadedbytes = 0;
    req.pipe(nuevo_archivo);
    req.on('data', function(pedazo_archivo){
        uploadedbytes += pedazo_archivo.length;
        var progress = (uploadedbytes/totalBytes) * 100;
        res.write('progress: ' + parseInt(progress, 10) + "%\n");
    });

    req.on('end', function(){
        res.end('uploaded');
    });
}).listen(8080);

```

```
$ curl --upload-file readme.md http://localhost:8080
```

# Closure

A closure is a function defined within another scope that has access to all variables within the outer scope

```
function greet(message) {
  console.log(message);
}

function greeter(name, age) {
  return name + ", who is " + age + " years old, says hi!";
}

// Generate the message
var message = greeter("Bob", 47);

// Pass it explicitly to greet
greet(message);

```

```
function greeter(name, age) {
  var message = name + ", who is " + age + " years old, says hi!";

  return function greet() {
    console.log(message);
  };
}

// Generate the closure
var bobGreeter = greeter("Bob", 47);

// Use the closure
bobGreeter();

```

# Mongodb modeling

PERSON

| Pers_ID | Surname | First_Name | City |
|---|---|---|---|
| 0 | Miller | Paul | London |
| 1 | Ortega | Alvaro | Valencia |
| 2 | Huber | Urs | Zurich |
| 3 | Blanc | Gaston | Paris |
| 4 | Bertolini | Fabrizio | Rome |

NO RELATION

CAR

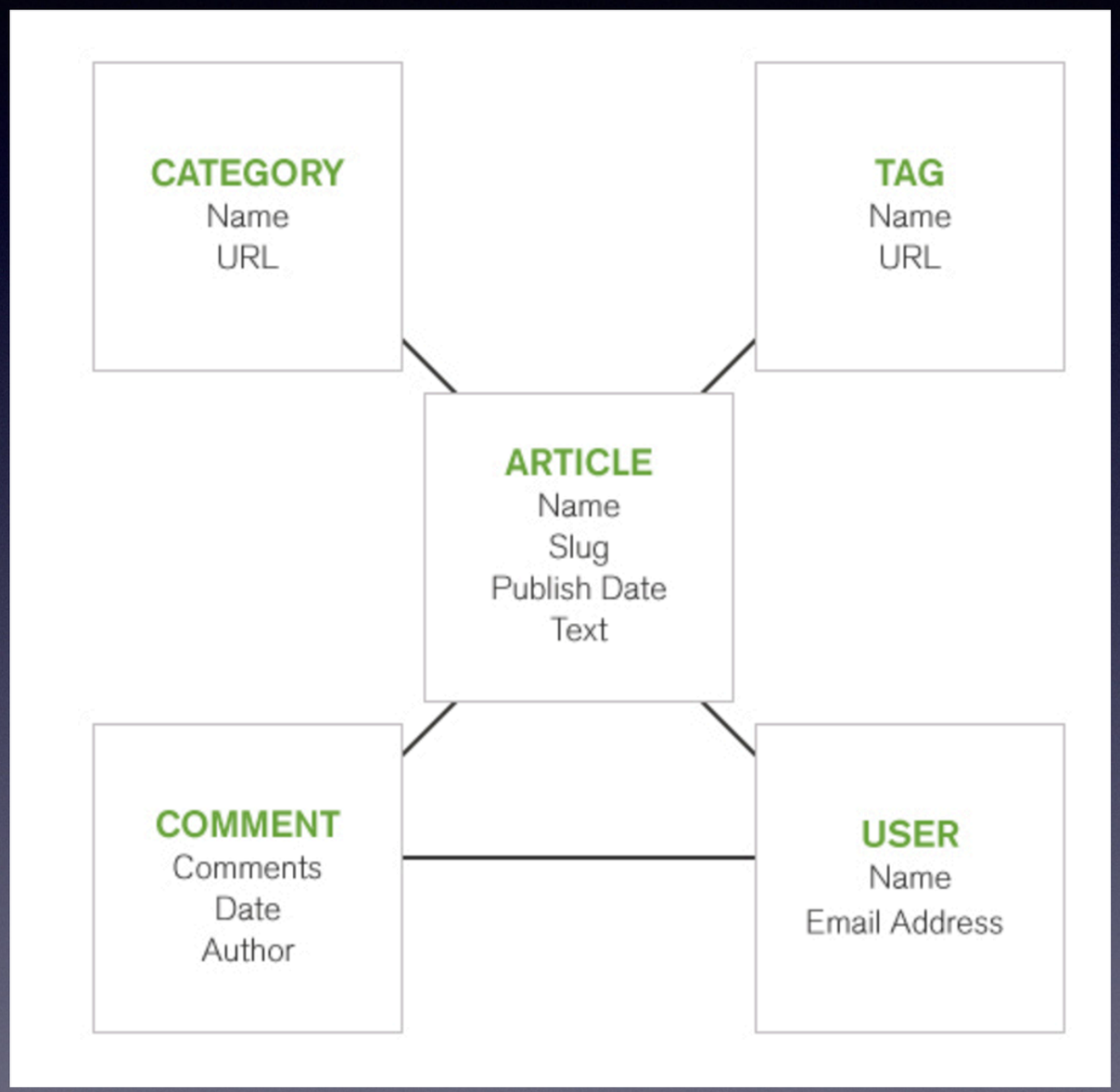
| Car_ID | Model | Year | Value | Pers_ID |
|---|---|---|---|---|
| 101 | Bently | 1973 | 100000 | 0 |
| 102 | Rolls Royce | 1965 | 330000 | 0 |
| 103 | Peugeot | 1993 | 500 | 3 |
| 104 | Ferrari | 2005 | 150000 | 4 |
| 105 | Renault | 1998 | 2000 | 3 |
| 106 | Renault | 2001 | 7000 | 3 |
| 107 | Smart | 1999 | 2000 | 2 |

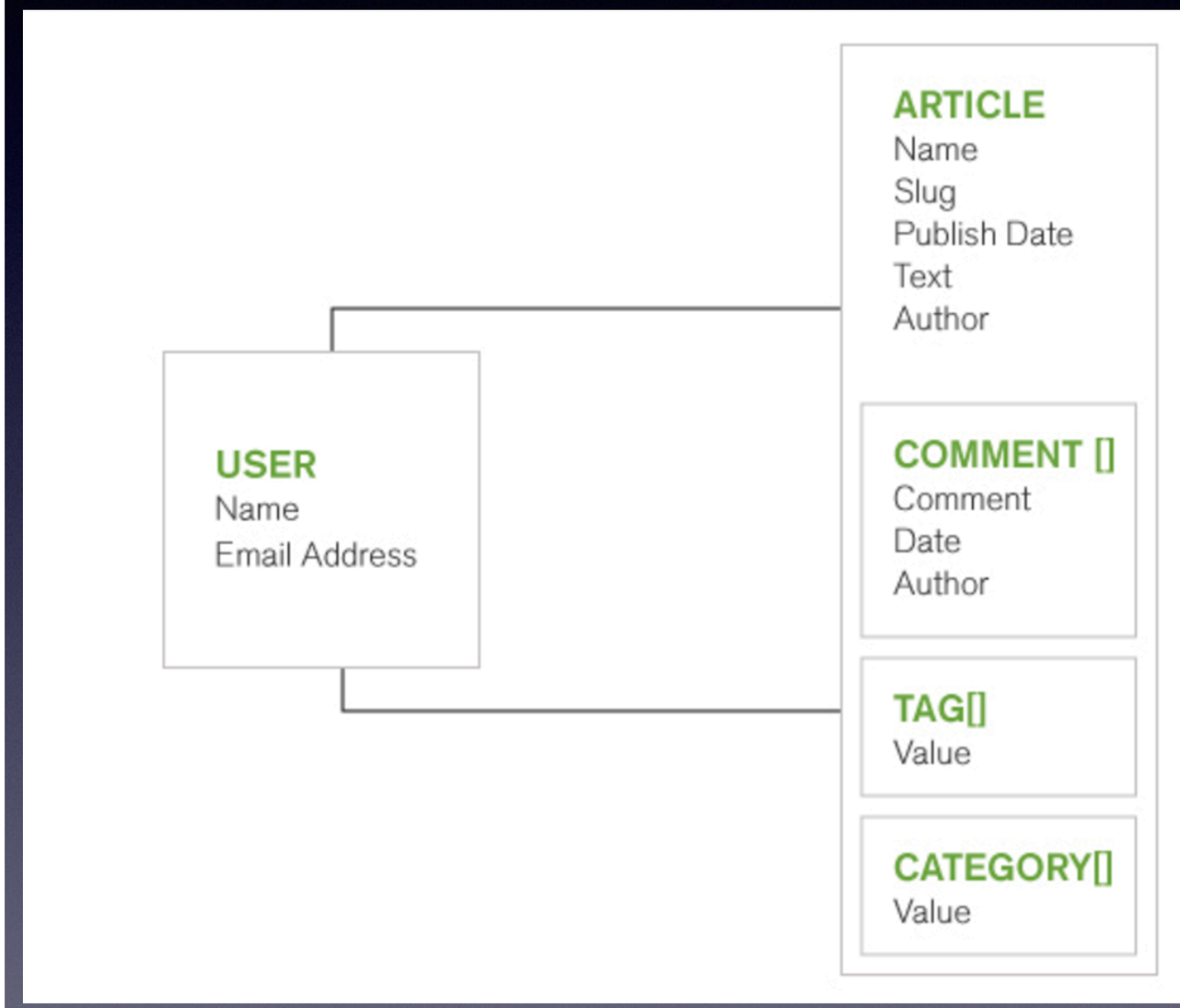
**Figure 1: Normalization and JOINs**

```
{
        first_name: “Paul”,
        surname: “Miller”,
        city: “London”,
        location: [45.123,47.232],
        cars: [
           { model: “Bentley”,
                year: 1973,
                value: 100000, ….},
           { model: “Rolls Royce”,
                year: 1965,
                value: 330000, ….},
        ]
        }
```

gistfile1.txt hosted with ❤ by GitHub view raw

# mongo modeling



A Web Page

http://

Laboratorio Xentiz

Doctor: Marco Canedo

Resultados de Pacientes

| Paciente | Sexo | Edad | Resultado | Fecha de entraga |
|---|---|---|---|---|
| Carlos Romero | Masculino | 25 | | 2017-05-1 |
| Carlos Romero | Masculino | 25 | | 2017-05-1 |
| Carlos Romero | Masculino | 25 | | 2017-05-1 |
| Carlos Romero | Masculino | 25 | | 2017-05-1 |

```
var mongo ;

MongoClient.connect('mongodb://localhost:27017/lab_xentiz', function (err, db) {
  if (err) throw err;
  mongo = db;
});

var app = express();

// view engine setup
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'jade');

// uncomment after placing your favicon in /public
//app.use(favicon(path.join(__dirname, 'public', 'favicon.ico')));
app.use(logger('dev'));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, 'public')));

app.use(function(req,res,next){
    req.mongo = mongo;
    next();
});

app.use('/', index);
app.use('/users', users);
app.use('/doctor', doctor);

// catch 404 and forward to error handler
app.use(function(req, res, next) {
  var err = new Error('Not Found');
  err.status = 404;
  next(err);
});
```

# POST

```
router.post('/', function(req, res) {
    var mongo = req.mongo;
    var data = {
        nombre: req.body.nombre,
        apellido: req.body.apellido,
        celular: req.body.celular,
        email: req.body.email
    };

    mongo.collection('doctors').insertOne(data, function(err, result){
        if (err) {
            var error = new Error(err);
            res.status(500).json(error);
        } else {
            console.log(result);
            res.json(result.ops[0]);
        }
    });
});
```

# GET

```
router.get('/', function(req, res) {
  var mongo = req.mongo;

  mongo.collection('doctors').find().toArray(function(err, docs){
      res.json(docs);
  });
```

# GET

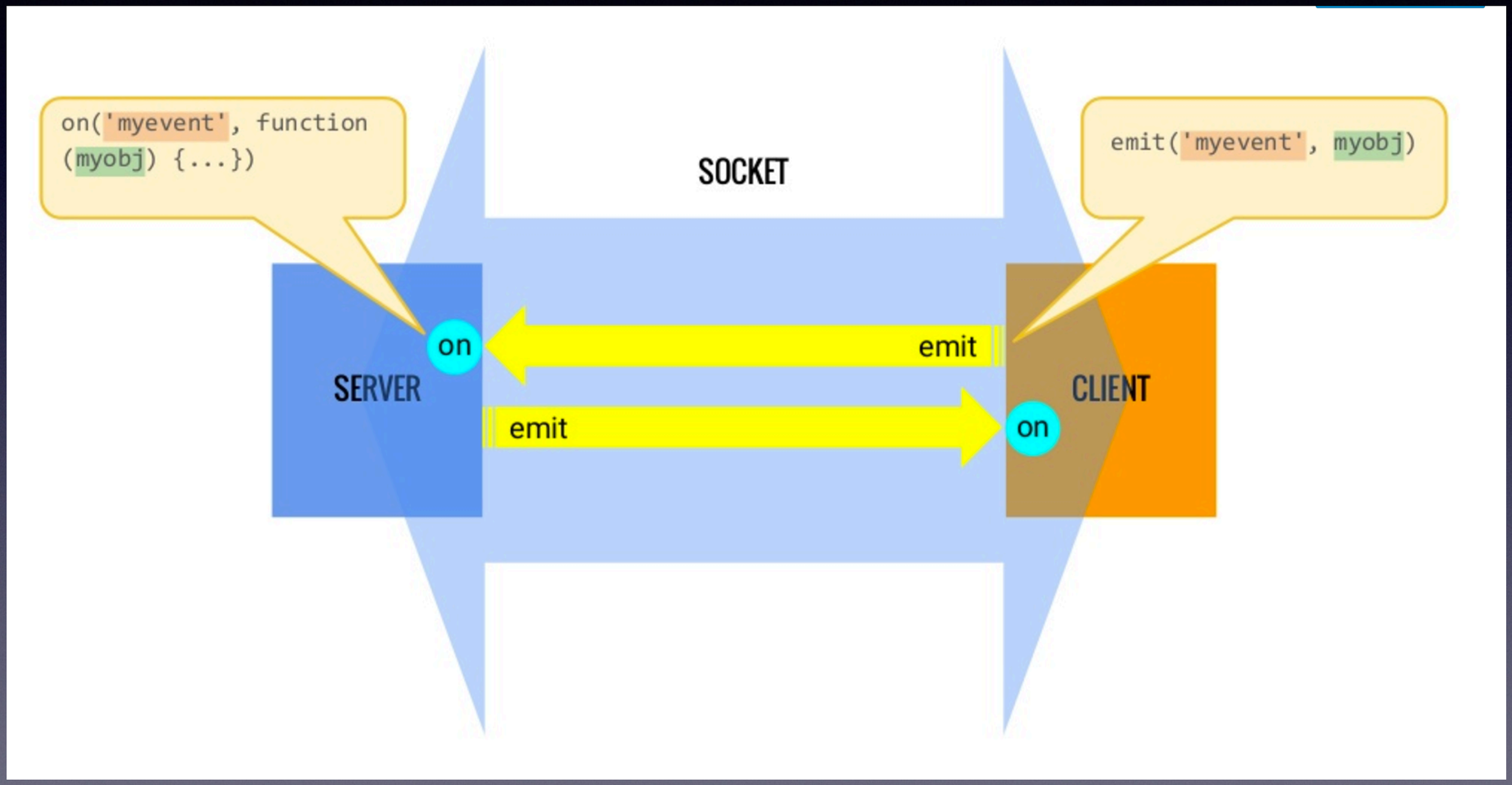
```
router.get('/:id', function(req, res){
    var mongo = req.mongo;
    var id = req.params.id;

    mongo.collection('doctors').findOne({ _id : new ObjectID(id)}, {}, function(err, doc){
        if (err) {
            res.status(500).json(new Error(err));
        }
        res.json(doc);
    });
});
```

```
router.put('/:id', function(req, res){
    var id = req.params.id;
    var body = req.body;
    var mongo = req.mongo;

    if (typeof body != 'undefined' && body) {
        mongo.collection('doctors').updateOne( { _id: new ObjectID(id) }, { $set: body }, function(err, result){

            if (err) {
                var error = new Error(err);
                res.status(500).json(error);
            }

            mongo.collection('doctors').findOne({_id: new ObjectID(id)}, {}, function(err, doc) {
                if (err) {
                    var error = new Error(err);
                    res.status(500).json(error);
                }

                res.json(doc);
            });;

        });
    } else {
        res.status(400).json('Bad Request');
    }
});
```

# DELETE

```
router.delete('/:id', function(req, res){
    var id = req.params.id;
    var mongo = req.mongo;

    mongo.collection('doctors').findOneAndDelete({ _id: new ObjectID(id)}, {}, function(err, result){

        if (err) {
            res.status(500).json(new Error(err));
        }
        res.status(200).json('was delete succesfully');
    });
});
```

# Socket IO


on('myevent', function (myobj) {...})
SOCKET
emit('myevent', myobj)
on
emit
SERVER
CLIENT
emit
on

# callback hell

```
doAsync1(function () {
  doAsync2(function () {
    doAsync3(function () {
      doAsync4(function () {
    })
  })
})
```

# Promise

```
1  const makeRequest = () =>
2    getJSON()
3      .then(data => {
4        console.log(data)
5        return "done"
6      })
7
8  makeRequest()
```

**syntax-promise.js** hosted with ❤️ by **GitHub** view raw

# Async/Await

```
1  const makeRequest = async () => {
2    console.log(await getJSON())
3    return "done"
4  }
5
6  makeRequest()
```

syntax-async.js hosted with ❤️ by GitHub view raw

# with Promise

```
const makeRequest = () => {
  try {
    getJSON()
      .then(result => {
        // this parse may fail
        const data = JSON.parse(result)
        console.log(data)
      })
      // uncomment this block to handle asynchronous errors
      // .catch((err) => {
      //   console.log(err)
      // })
  } catch (err) {
    console.log(err)
  }
}
```

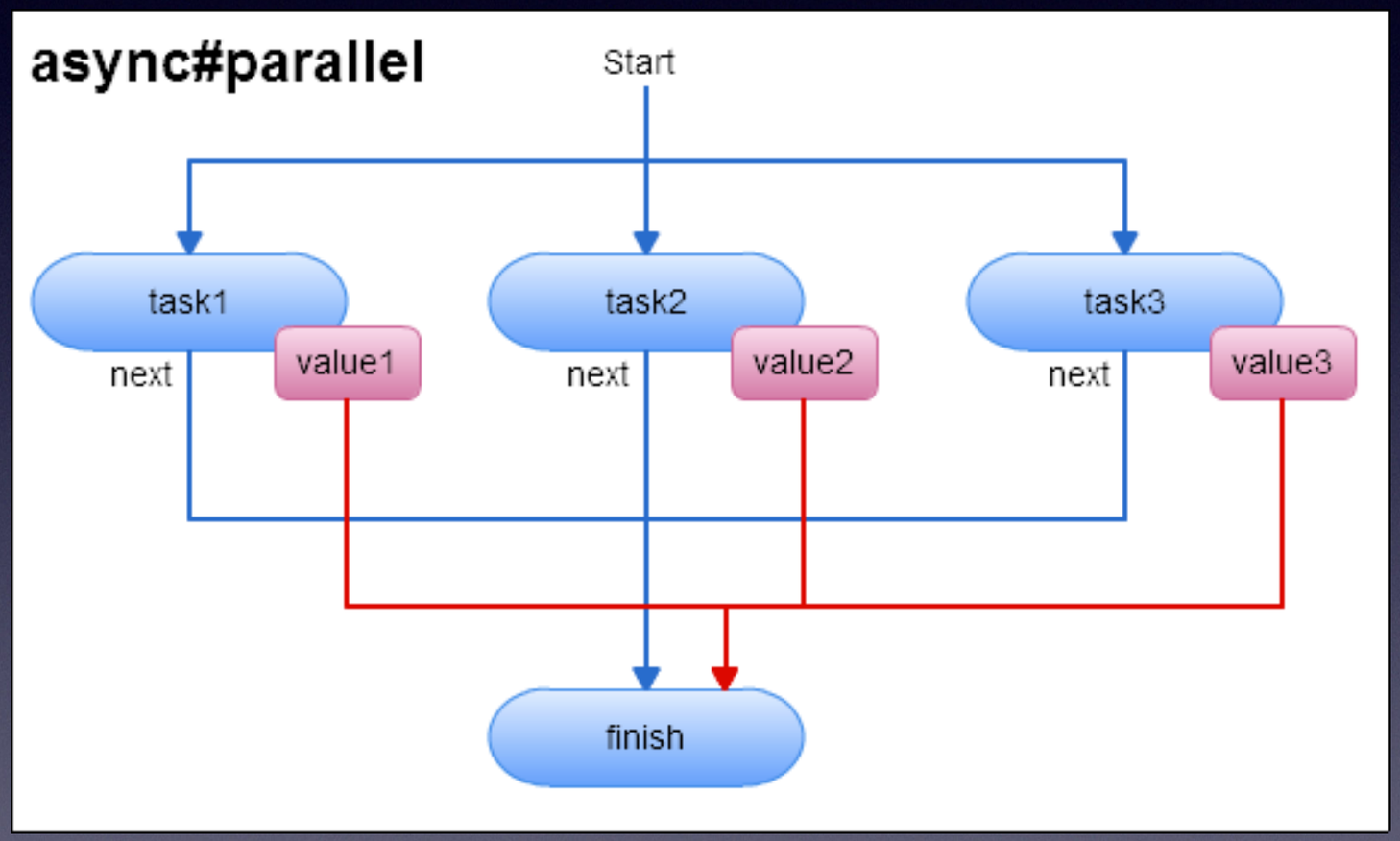
**catch-promise.js** hosted with ❤️ by **GitHub** view raw

# with Async/Await

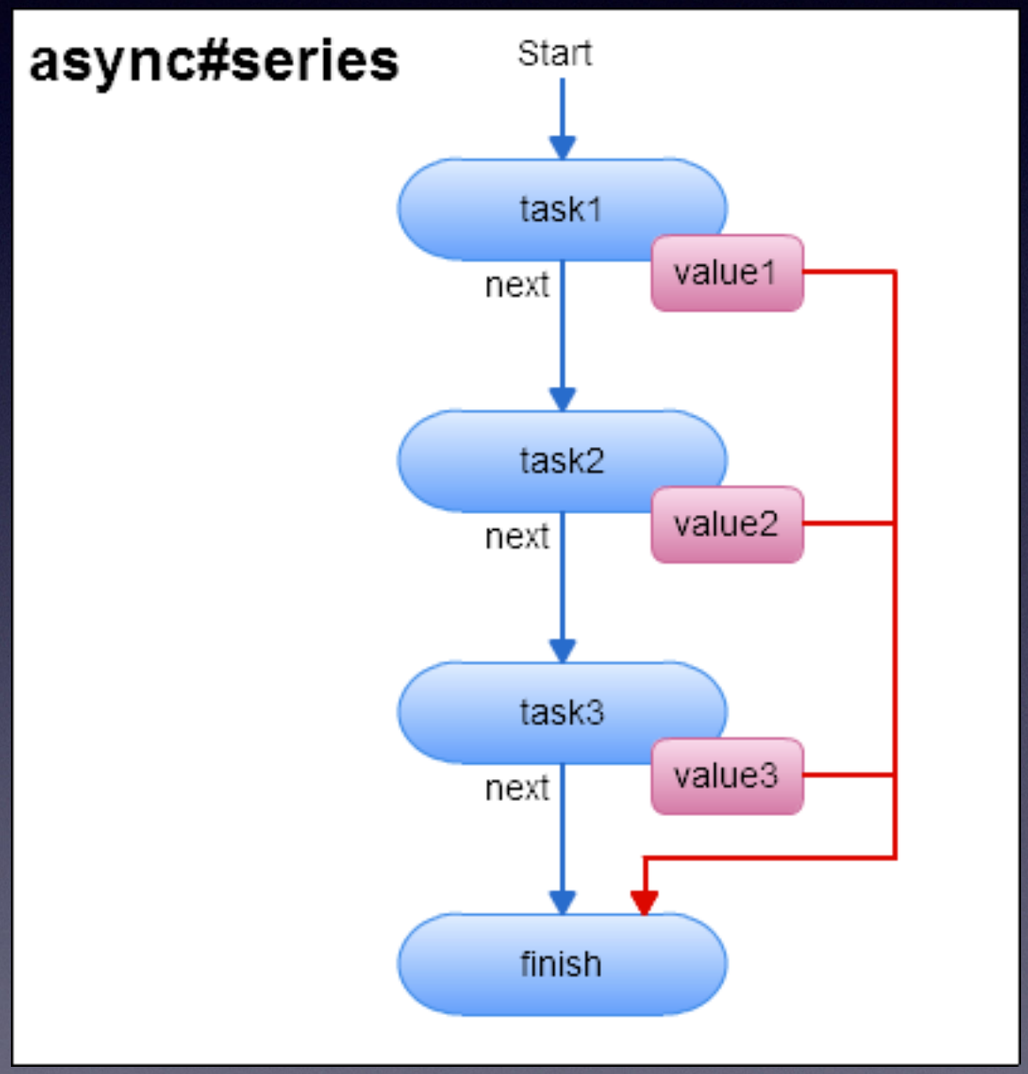
```
const makeRequest = async () => {
  try {
    // this parse may fail
    const data = JSON.parse(await getJSON())
    console.log(data)
  } catch (err) {
    console.log(err)
  }
}
```

catch-async.js hosted with ❤ by GitHub view raw

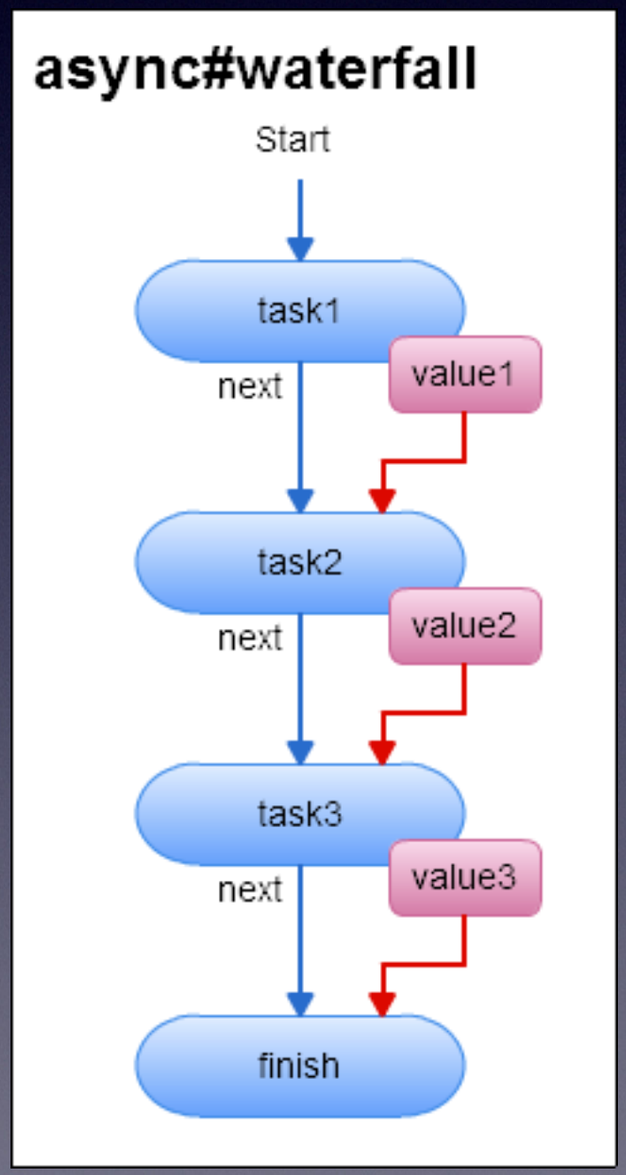
# Asycn flow control



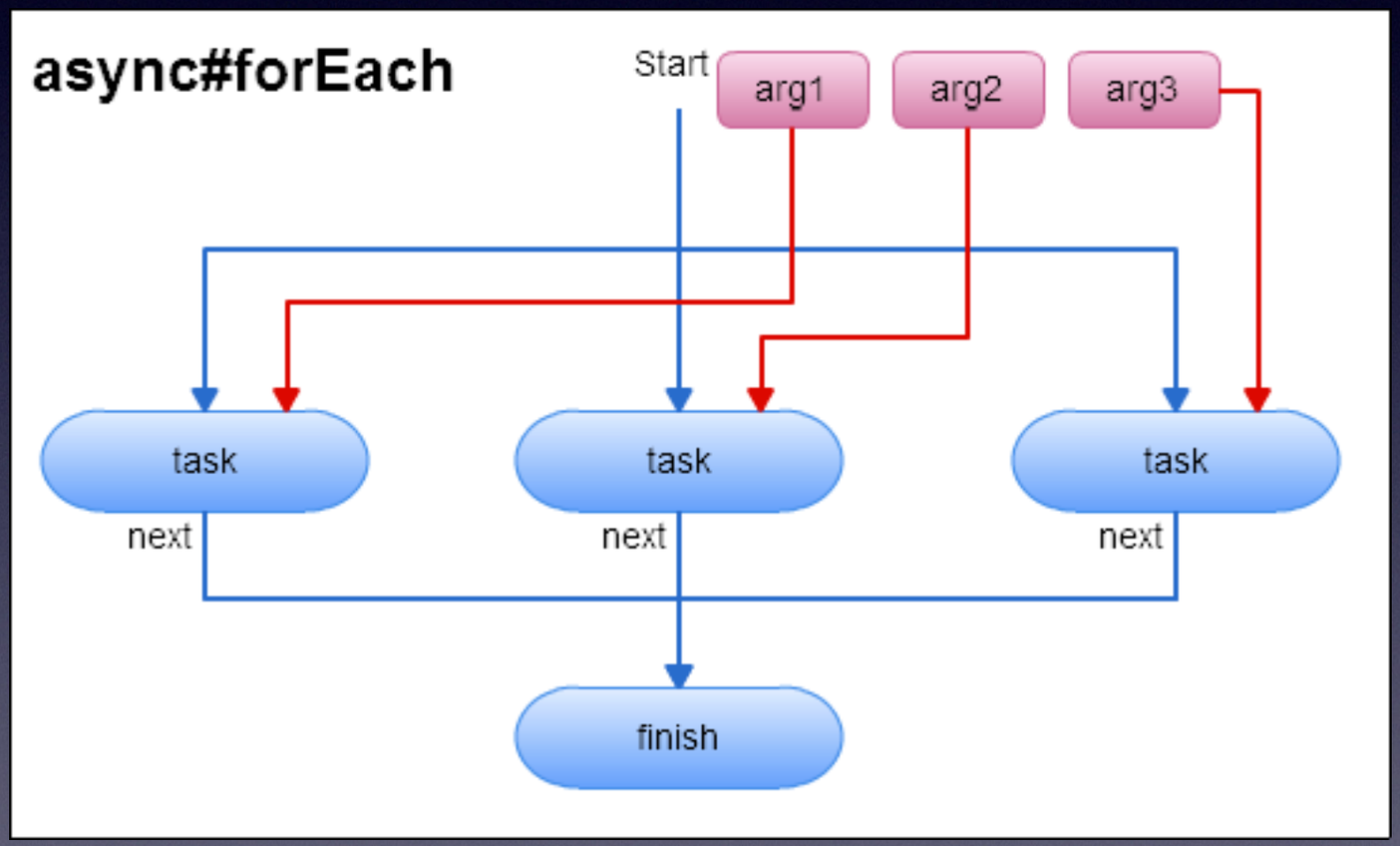
# Asycn flow control



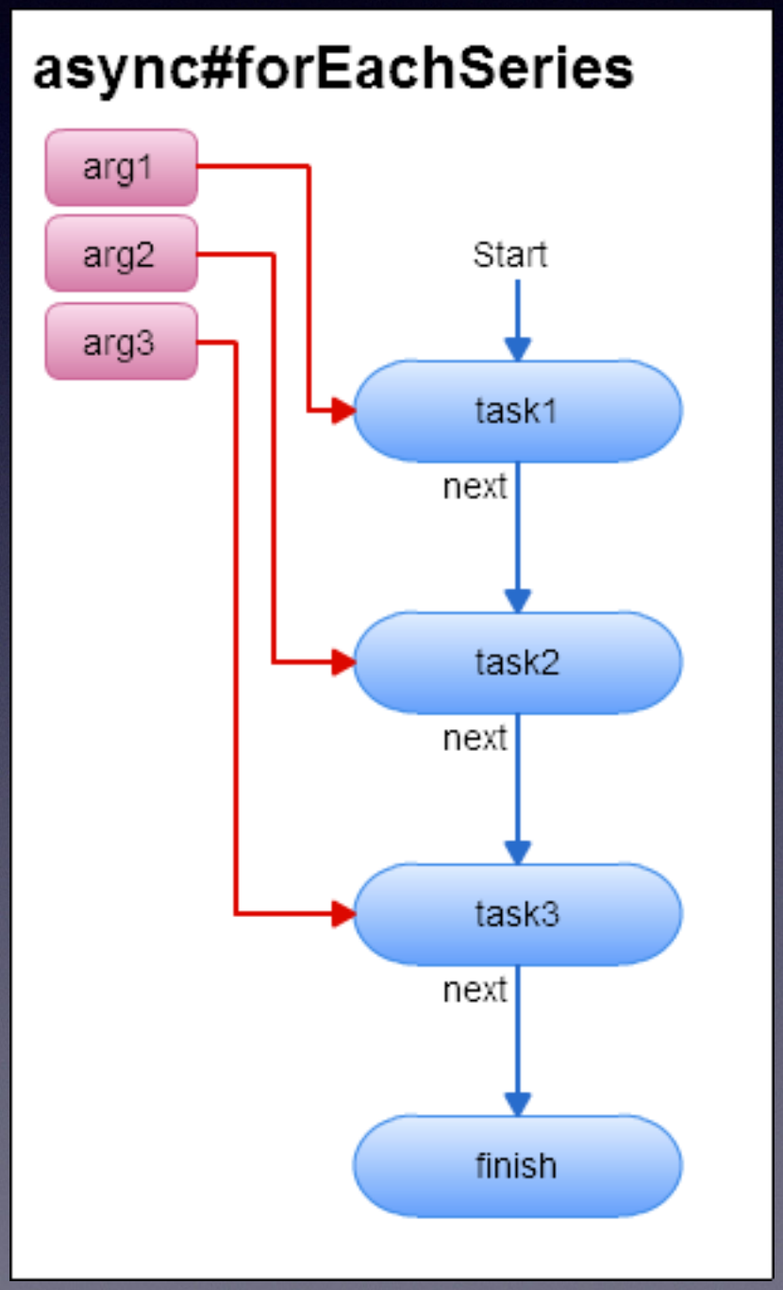
# Asycn flow control



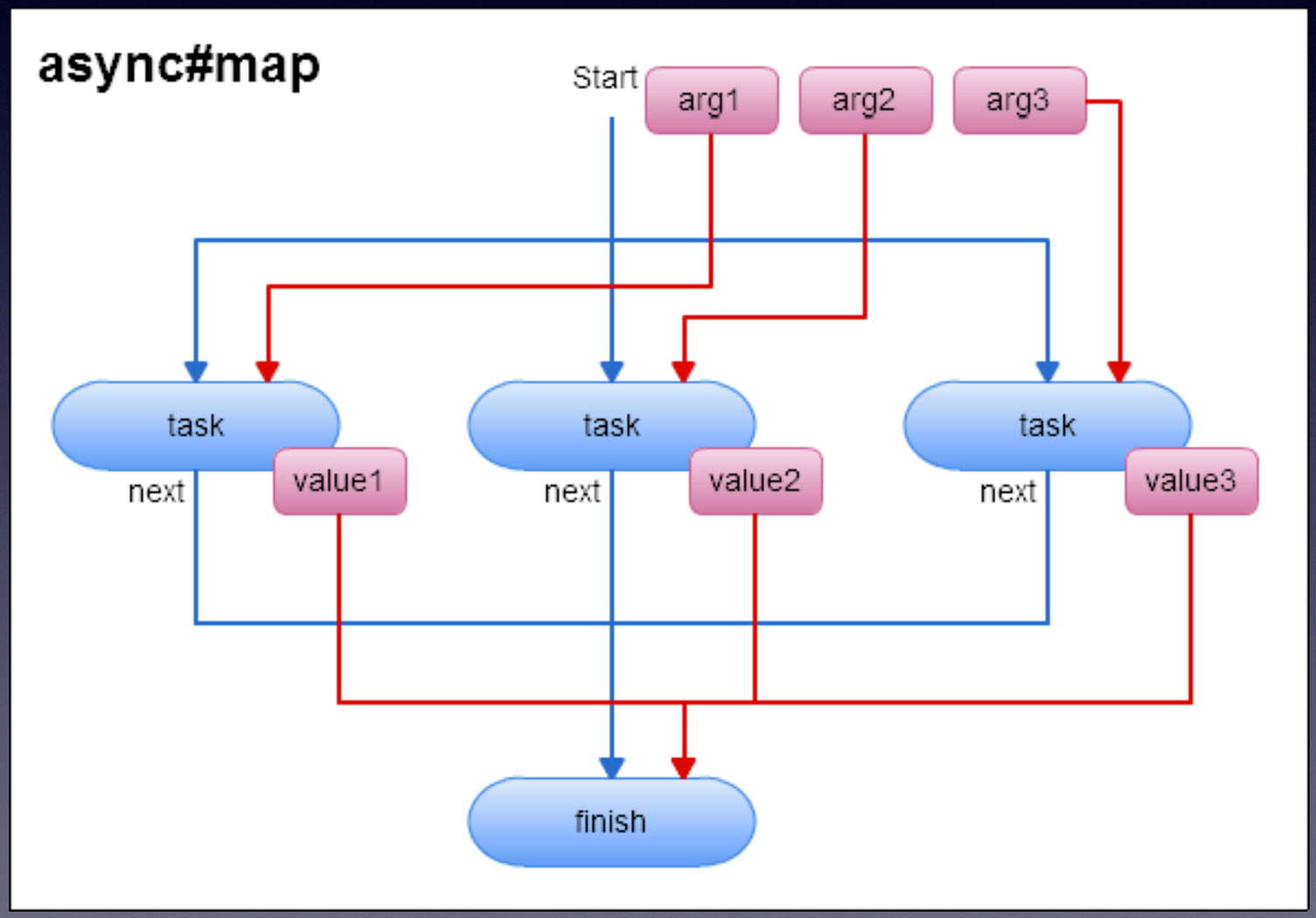
# Asycn flow control



# Asycn flow control



# Asycn flow control



# Asycn flow control

```
async.parallel([
    function(callback) {
      db.save('xxx', 'a', callback)
    },
    function(callback) {
      db.save('xxx', 'b', callback)
    }
], function(err) {
    if (err) throw err
    console.log('Both a and b are saved now')
})
```

With ES6, much shorter:

```
async.parallel([
    callback => db.save('xxx', 'a', callback),
    callback => db.save('xxx', 'b', callback)
], err => {
    if (err) throw err
    console.log('Both a and b are saved now')
})
```

# Async Await

- Async/await is a new way to write asynchronous code. Previous options for asynchronous code are callbacks and promises.
- Async/await is actually built on top of promises. It cannot be used with plain callbacks or node callbacks.
- Async/await is, like promises, non blocking.
- Async/await makes asynchronous code look and behave a little more like synchronous code. This is where all its power lies.

# Errores

```
JSON.parse('undefined')
```

```
try {
  JSON.parse(maybeJSON)
} catch (er) {
  console.error('Invalid JSON', er)
}
```

```
// Throws with a ReferenceError because z is undefined
try {
  const m = 1;
  const n = m + z;
} catch (err) {
  // Handle the error here.
}
```

# Errores

```
const fs = require('fs');
fs.readFile('a file that does not exist', (err, data) => {
  if (err) {
    console.error('There was an error reading the file!', err);
    return;
  }
  // Otherwise handle the data
});
```

```
const net = require('net');
const connection = net.connect('localhost');

// Adding an 'error' event handler to a stream:
connection.on('error', (err) => {
  // If the connection is reset by the server, or if it can't
  // connect at all, or on any sort of error encountered by
  // the connection, the error will be sent here.
  console.error(err);
});
```

# Errores

```
function myApiFunc(callback) {
  /*
   * This pattern does NOT work!
   */
  try {
    doSomeAsynchronousOperation(function (err) {
      if (err)
        throw (err);
      /* continue as normal */
    });
  } catch (ex) {
    callback(ex);
  }
}
```