@wizardnet972

MASTERING NODE.JS

# Applications with Node.js & Express.js

Shlomi Levi
@wizardnet972

@wizardnet972

λ whoami_

# Shlomi Levi

- Freelance consultant.
- Providing consulting to ambitious teams.
- Technical blogs @medium.com
- Senior Microsoft Certified Solution Developer

You can find me here

Features Business Explore Marketplace Pricing Search GitHub / Sign in or Sign up

## Shlomi Levi
wizardnet972

I'm independent consultant, mentor, coder, make some magic and providing Angular/Node.JS consulting to ambitious teams.

Block or report user

@wizardnet972
Israel
Sign in to view email
http://www.shlomi-levi.net

Overview Repositories 54 Stars 107 Followers 28 Following 64

Pinned repositories

**angular-elements-talk**
this is code from my talk about angular-elements in depth (lite version)
TypeScript ★ 2

**critical-plugin**
Critical plugin for webpack (https://webpack.js.org/)
JavaScript ★ 11

**tic-tac-toe**
Implementation of Tic-Tac-Toe Game with ngrx state-management
TypeScript ★ 11

**memory-game-ngrx**
Memory game in ngrx
TypeScript ★ 7

**sequelize-node**
sequelize for node as module
TypeScript

**angular**
Forked from angular/angular
One framework. Mobile & desktop.
TypeScript

125 contributions in the last year

Nov Dec Jan Feb Mar Apr May Jun Jul Aug Sep Oct

Fri

Learn how we count contributions. Less More

https://github.com/wizardnet972/

Contribution activity Jump to 2018

@wizardnet972

# Node.js fundamentals

@wizardnet972

# What is node.js?

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.

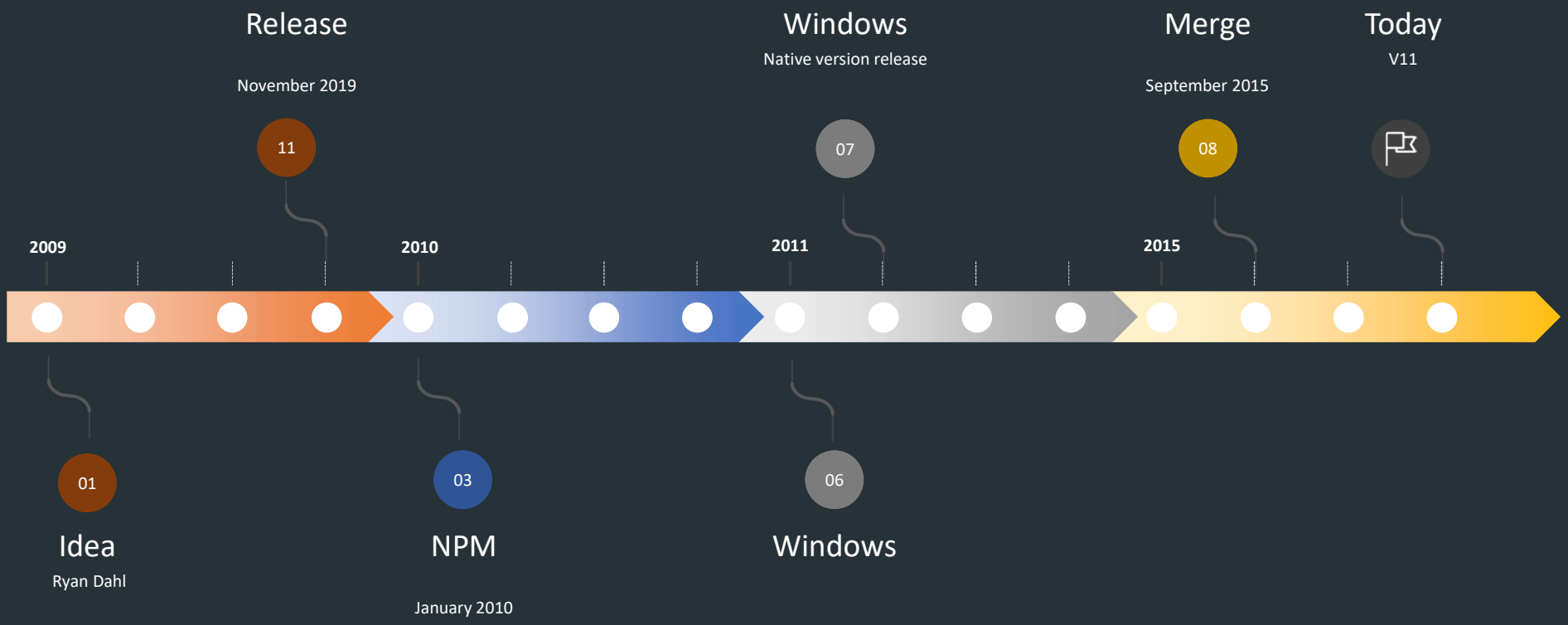
@wizardnet972

# What you can do with Node?

- Http server
- Sockets and networking
- Data streaming
- Realtime applications
- Databases
- Console applications
- And more…

# Node.js History

@wizardnet972

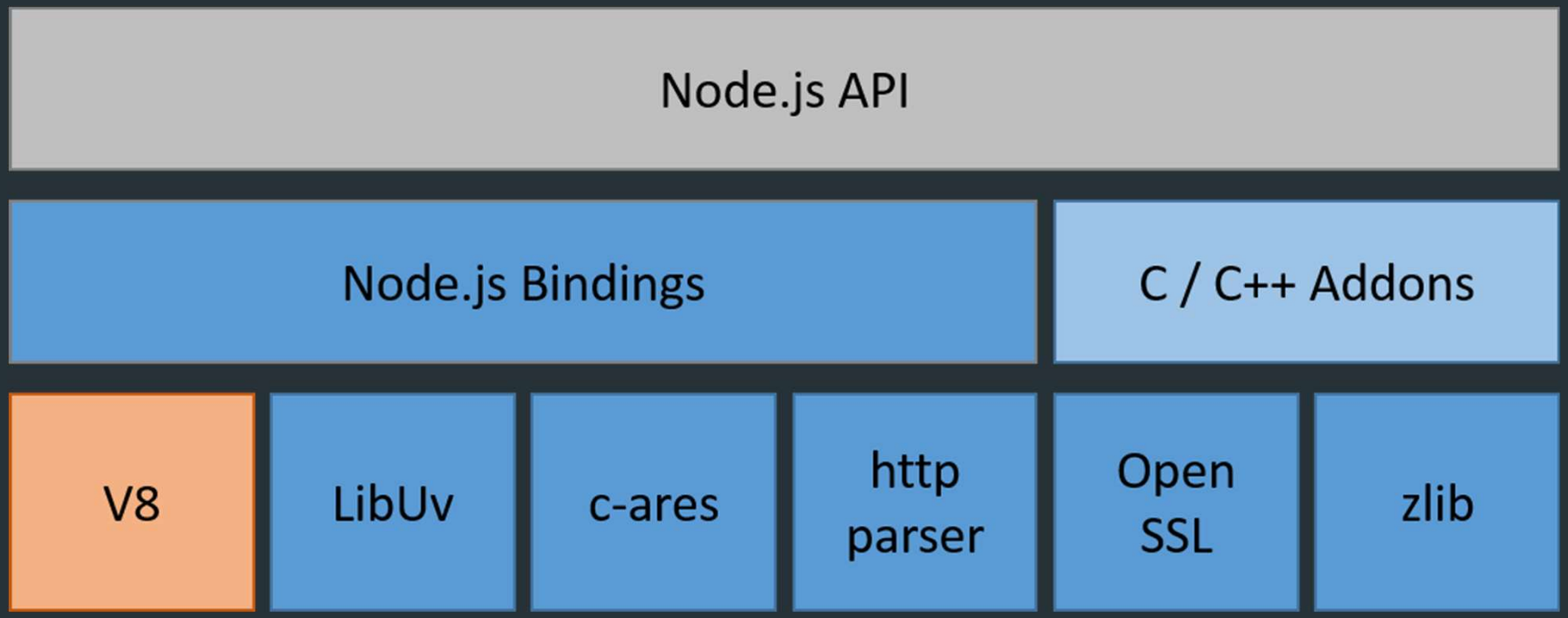


# What's in node.js?

@wizardnet972

@wizardnet972

# Node.js internal structure



# Google V8

Developed in 2008 for Google Chrome, now used in node.js, opera, vivaldi and electron.

JavaScript engine, open source

Chrome/Chromium browsers.

C++

Super Fast!

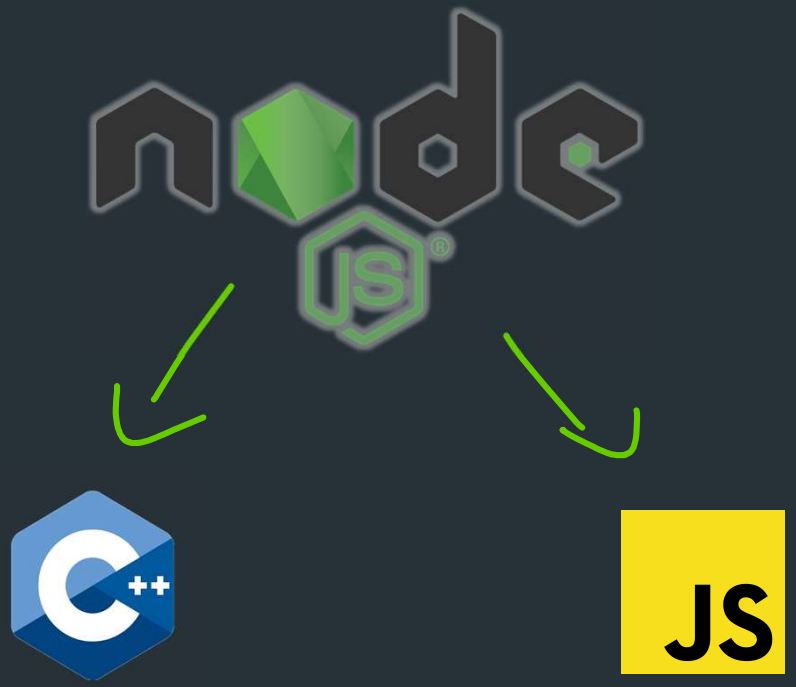
# Cross-platform asynchronous I/O


asynchronous I/O

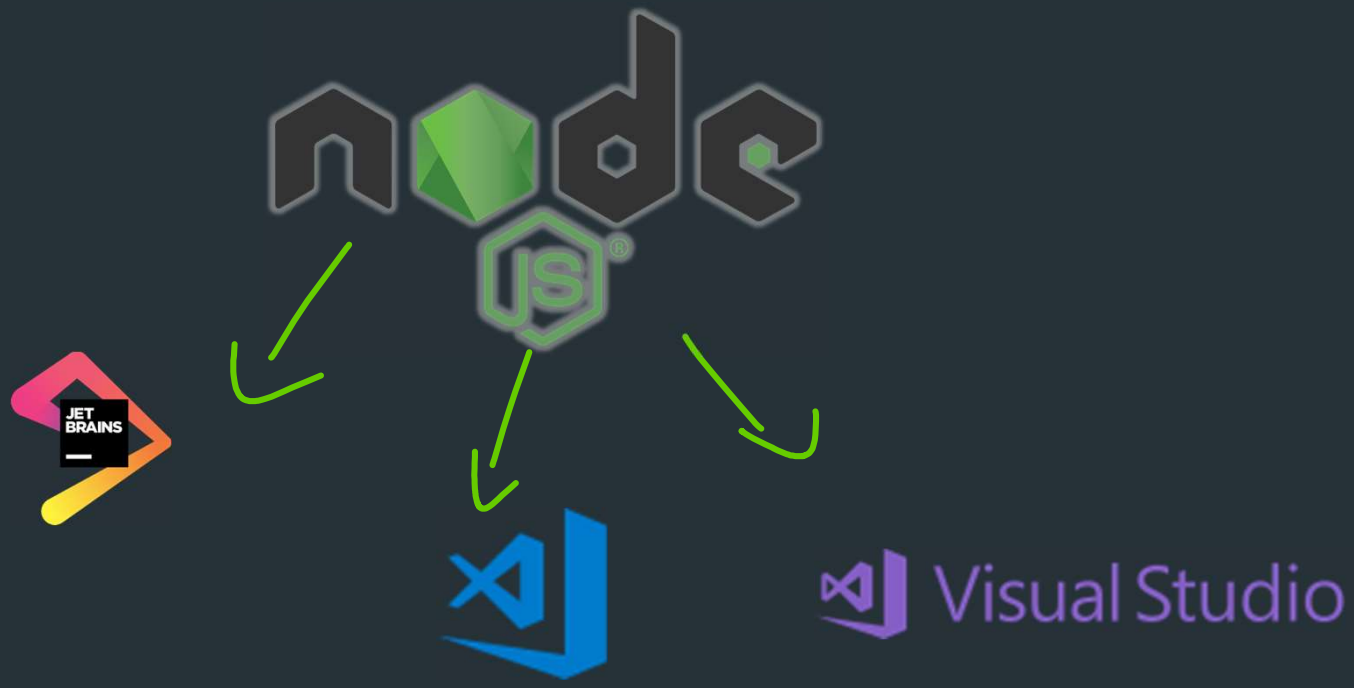
C language.

Luvit, Julia, pyuv, and others.

# Languages

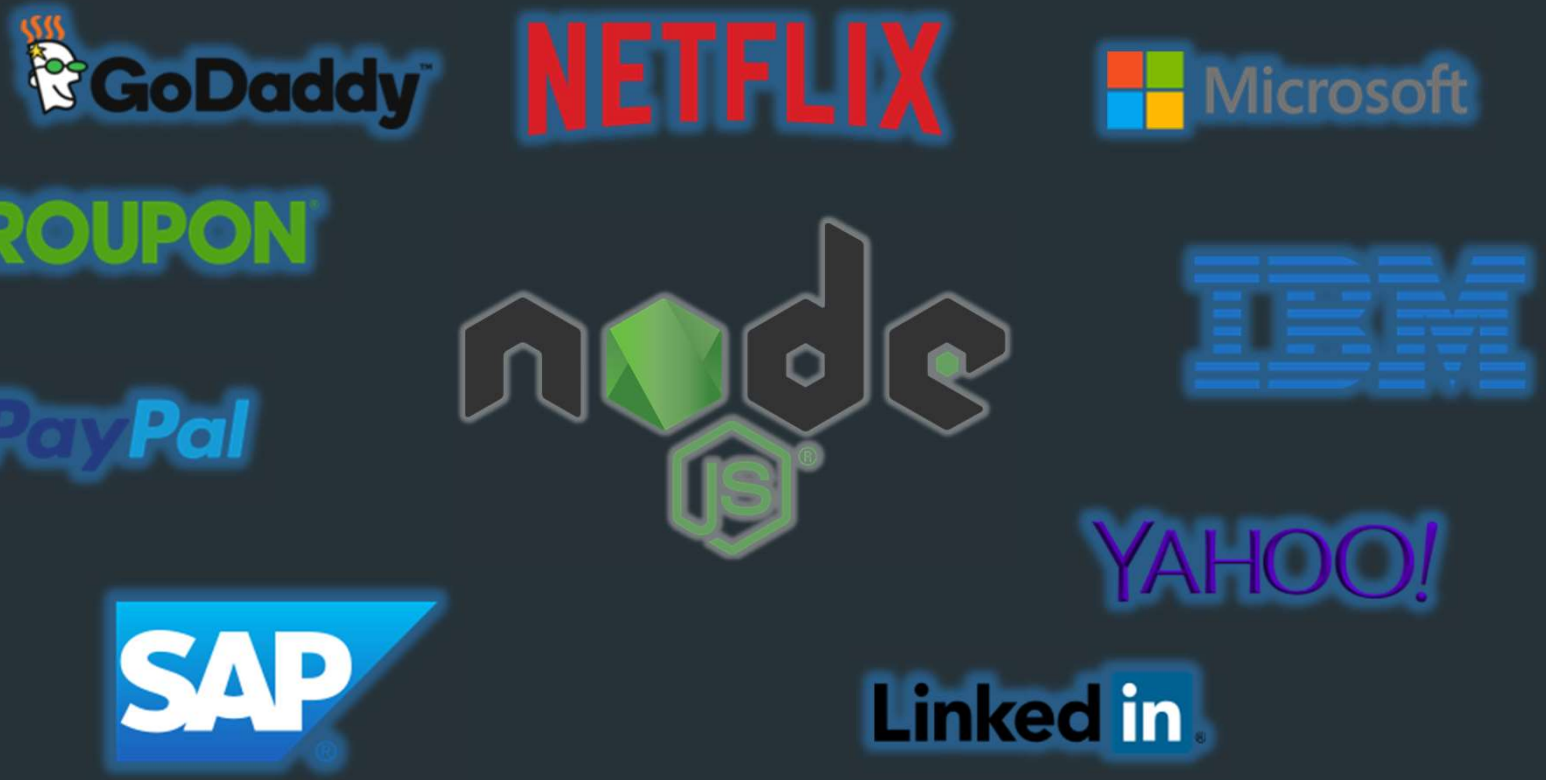


# IDEs



@wizardnet972

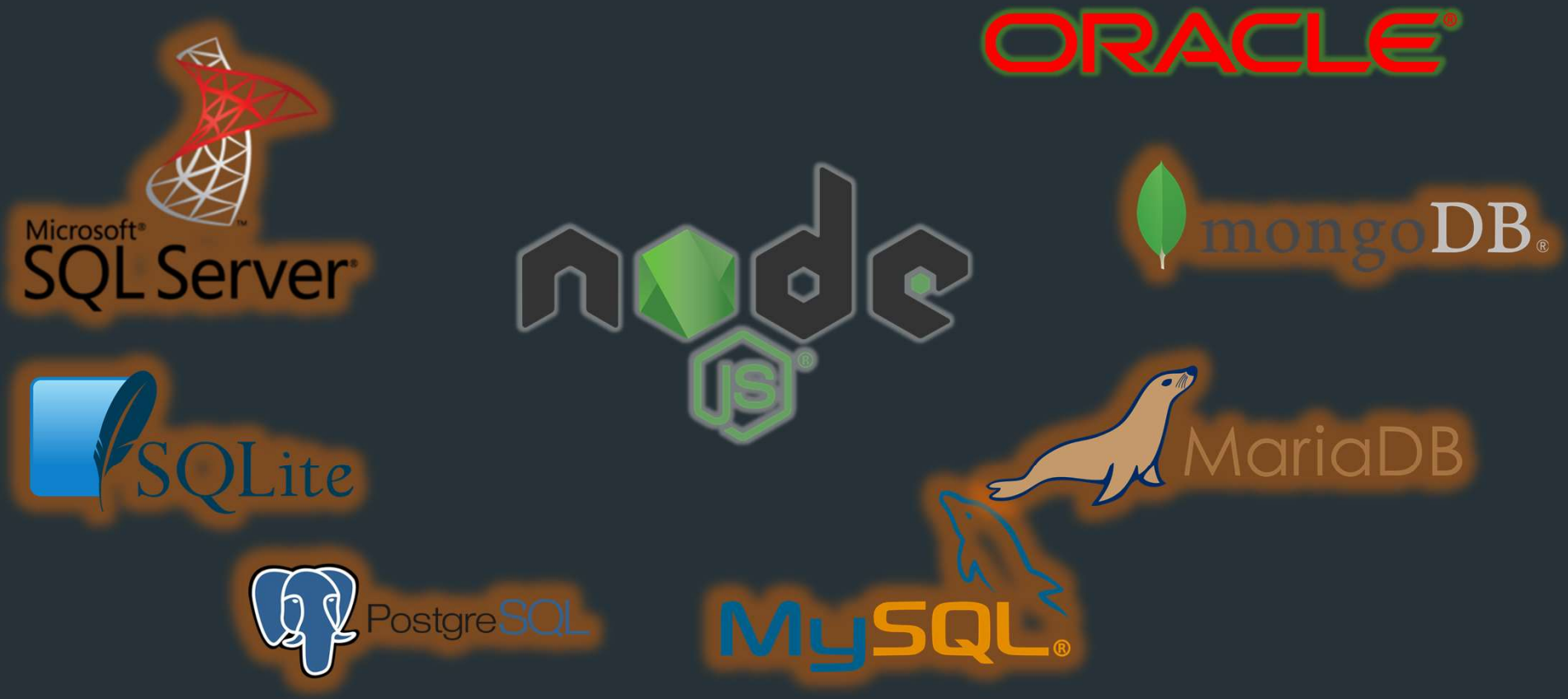
# Node.js around the world



@wizardnet972

# Nodejs frameworks

# Nodejs Databases

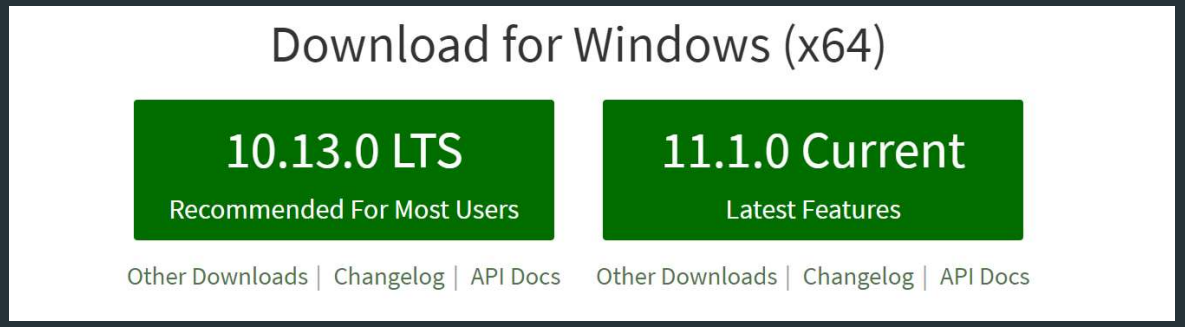


# Nodejs Community

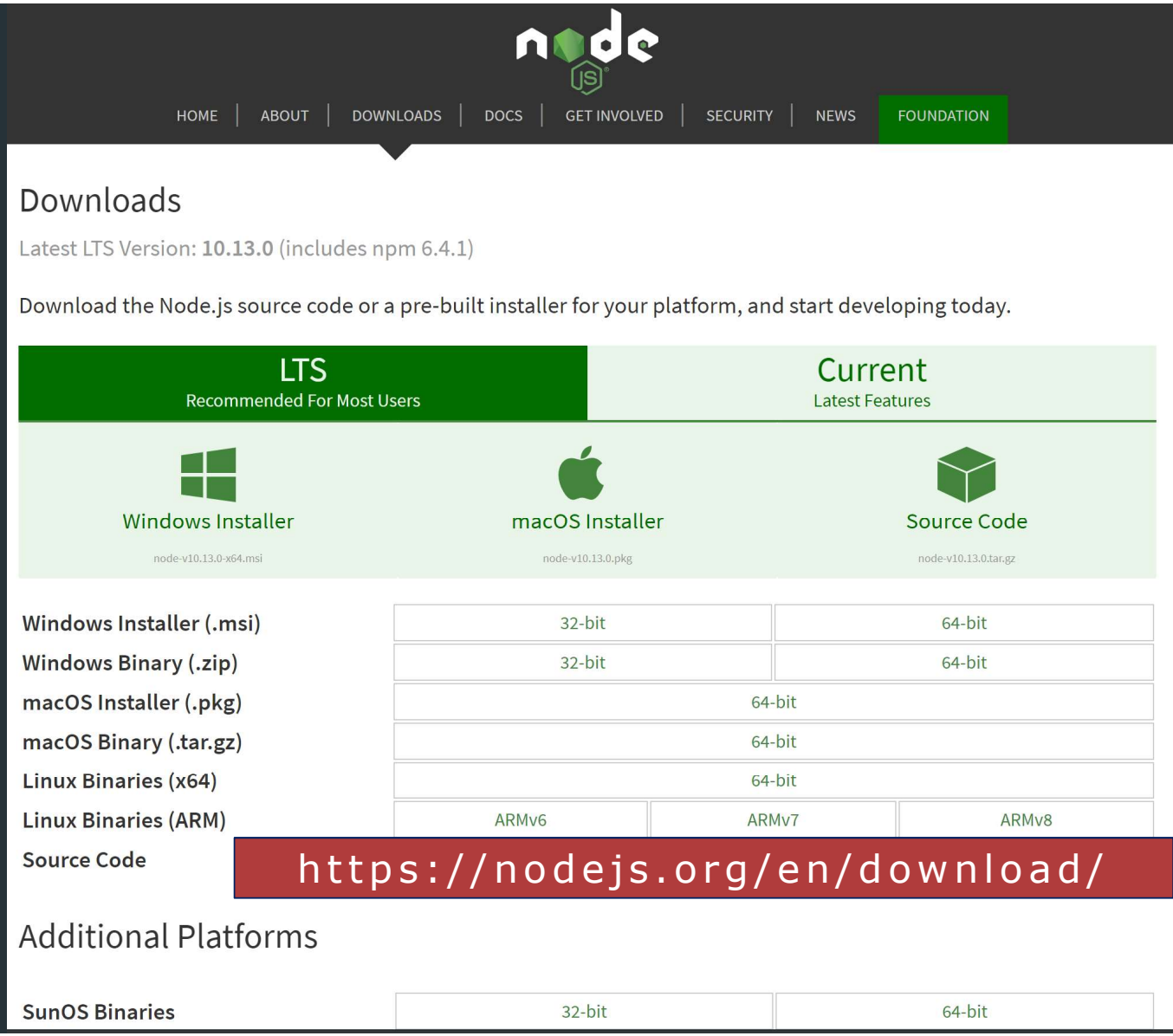



https://stackoverflow.com/questions/tagged/node.js

# Releases


Download for Windows (x64)
10.13.0 LTS
Recommended For Most Users
11.1.0 Current
Latest Features
Other Downloads | Changelog | API Docs
Other Downloads | Changelog | API Docs

# Download


HOME
ABOUT
DOWNLOADS
DOCS
GET INVOLVED
SECURITY
NEWS
FOUNDATION
Downloads
Latest LTS Version: 10.13.0 (includes npm 6.4.1)
Download the Node.js source code or a pre-built installer for your platform, and start developing today.
LTS
Recommended For Most Users
Current
Latest Features
Windows Installer
node-v10.13.0-x64.msi
macOS Installer
node-v10.13.0.pkg
Source Code
node-v10.13.0.tar.gz
Windows Installer (.msi) 32-bit 64-bit
Windows Binary (.zip) 32-bit 64-bit
macOS Installer (.pkg) 64-bit
macOS Binary (.tar.gz) 64-bit
Linux Binaries (x64) 64-bit
Linux Binaries (ARM) ARMv6 ARMv7 ARMv8
Source Code
https://nodejs.org/en/download/
Additional Platforms
SunOS Binaries 32-bit 64-bit

# Cmder

Full https://bit.ly/2rUrtYd

Mini https://bit.ly/2BEBXiV

@wizardnet972

# Hello node.js World!

# Modules

# Everything is a module!


```
// person.js
function Person(options) {
 this.firstname = options.firstname || '';
 this.lastname = options.lastname || '';
}

module.exports = Person;

// app.js
var Person = require('./person.js');

var employee = new Person({
 firstname: 'shlomi',
 lastname: 'Levi'
});

console.log(employee);
```

@wizardnet972

# Html vs Nodejs


```
(function(exports, require, module, __filename, __dirname) {

  // Module code actually lives in here

});
```


```
(function(exports, require, module, __filename, __dirname) {

  // Module code actually lives in here

   // person.js
   function Person(options) {
    this.firstname = options.firstname || '';
    this.lastname = options.lastname || '';
   }

   module.exports = Person;

});
```

```
(function(exports, require, module, __filename, __dirname) {

  // Module code actually lives in here

   // app.js
   var Person = require('./person.js');

   var employee = new Person({
    firstname: 'shlomi',
    lastname: 'Levi'
   });

   console.log(employee);

});
```

# Core Modules

- Built-in modules.
- They are defined within Node.js's source and are located in the nodejs folder.
- Binaries modules?
- http, url, querystring, path, fs, util ...

# Local Modules

- Packages
- node_module
- Node.js module resolution

@wizardnet972

# Npm packages

Node package manager

# Node package manager - npm

Developed in 2010 by Isaac Z. Schlueter npm has :

400,000+ packages

200,000+ users registered

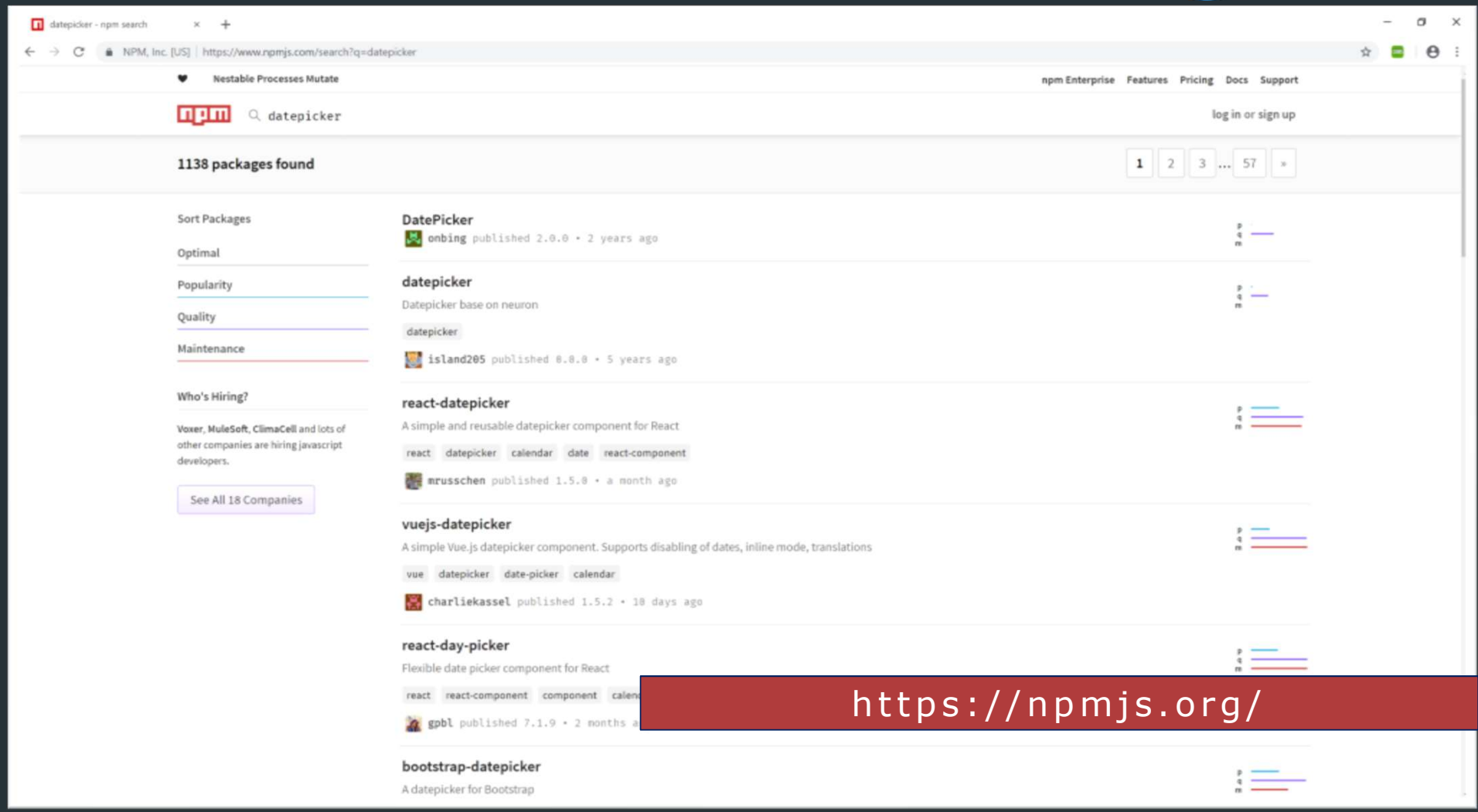
Npm is largest ecosystem of open source libraries in the world!

# Node package manager - npm

npm install / update / uninstall {packageName}@{version}

"-g" - globally,  "-s" - updating in package.json

@wizardnet972



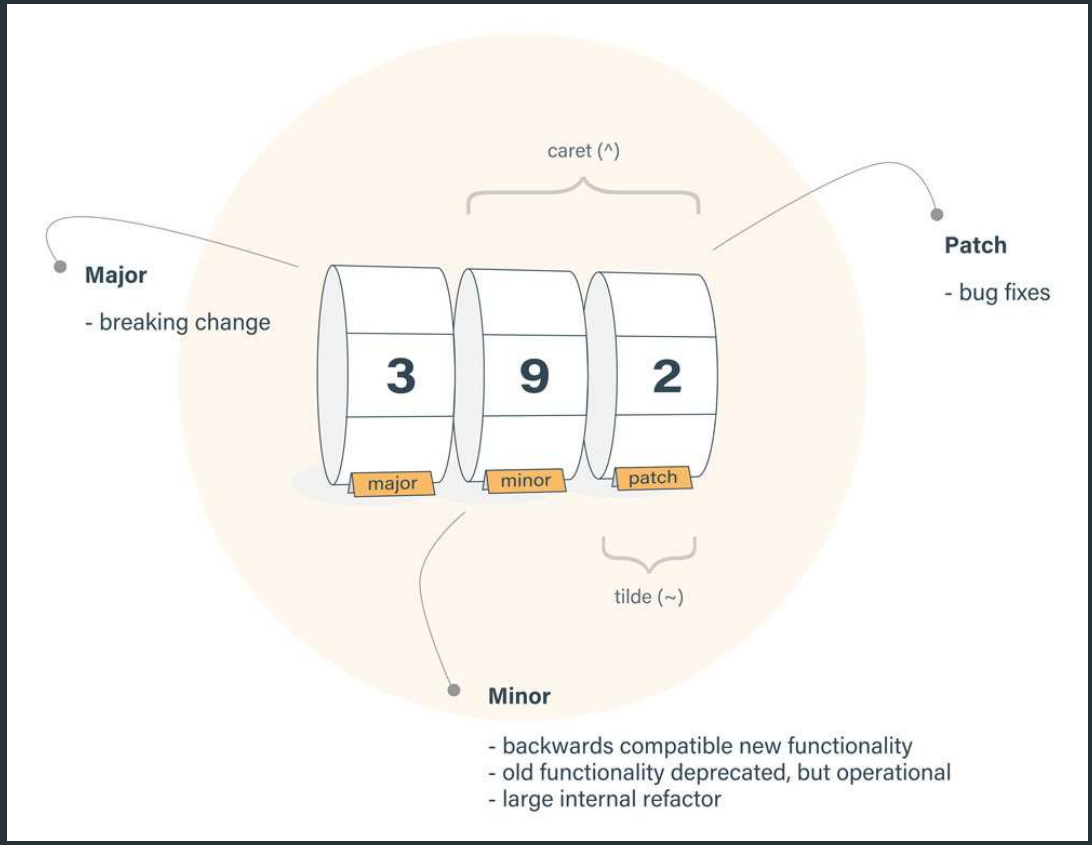

https://npmjs.org/

# Packages Versions

```
myApp
|---- foo.js
|---- node_modules
       |---- depA
       |      |---- index.js
       |---- depB
       |      |---- bar.js
       |      |---- node_modules
       |             |---- depA
       |                    |---- index.js
       |---- depC
              |---- foobar.js
              !---- node_modules
                     |---- depA
                            |---- index.js
```

# Dependencies Version

```
"node-uuid": "1.4.8"

"rxjs": "~6.3.3"

"sequelize": "^4.41.0"
```



# Package.json


# Hello Node

@wizardnet972

# Common packages

You should know

# Where and How to find what you looking for?

# Create npm package

demo


# Typescript

Javascript that scales

@wizardnet972

# Application scale JavaScript development is hard

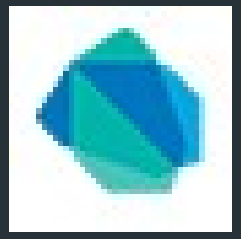
# JavaScript is:

- not design as a programming language for big application
- does not have strongly typing
- lack structuring mechanisms like classes, modules, interfaces

You can write large programs in JavaScript. You just can't maintain them.

# The Alternatives

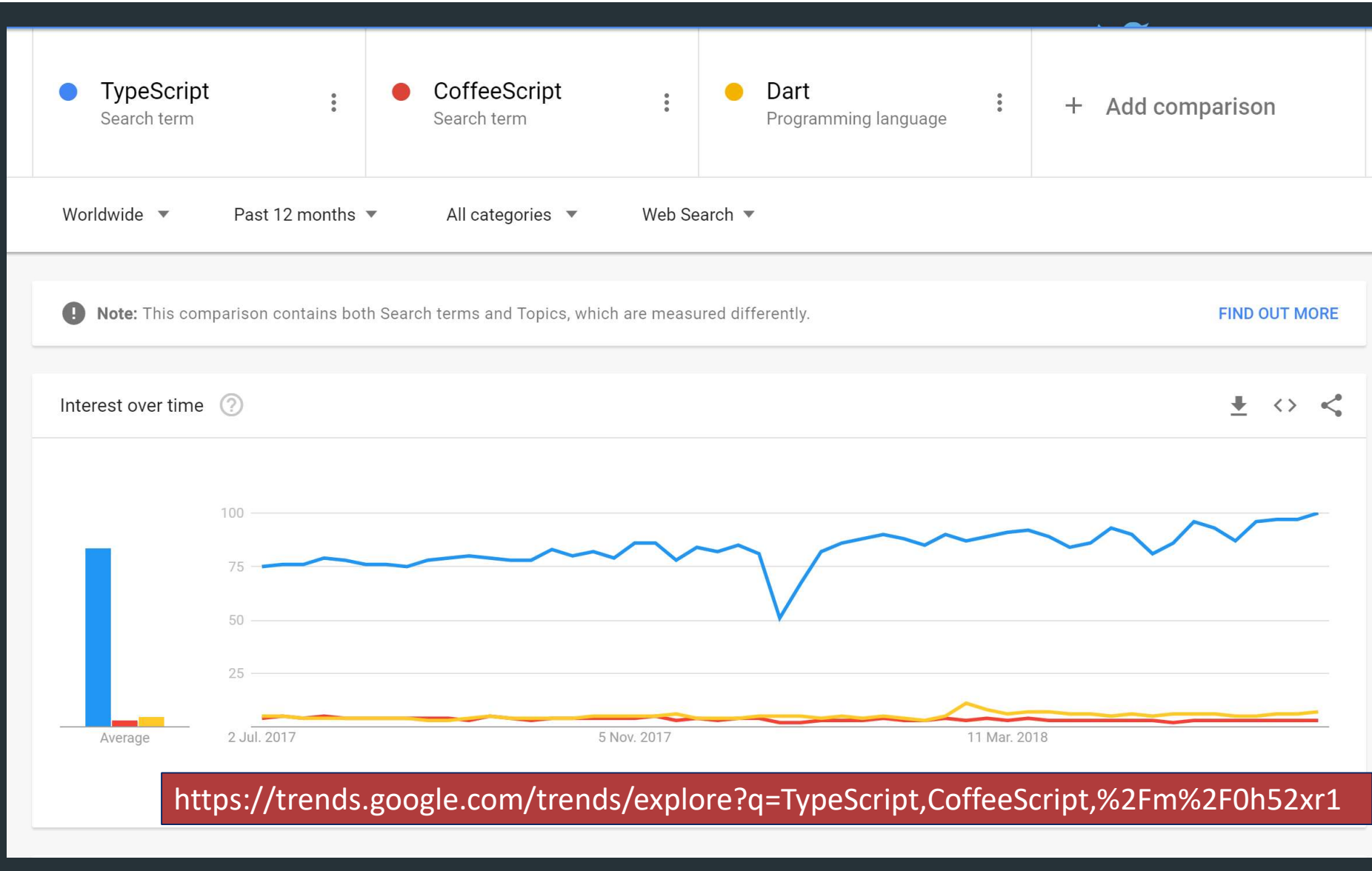
CoffeeScript
coffeescript.org
custom lang.



Dart
http://dartlang.org
custom lang.
by Google

Clojurescript
github.com
custom lang.


Script#
github.com
C#


TypeScript
Search term
CoffeeScript
Search term
Dart
Programming language
Add comparison
Worldwide
Past 12 months
All categories
Web Search
Note: This comparison contains both Search terms and Topics, which are measured differently.
FIND OUT MORE
Interest over time
100
75
50
25
Average
2 Jul. 2017
5 Nov. 2017
11 Mar. 2018
https://trends.google.com/trends/explore?q=TypeScript,CoffeeScript,%2Fm%2F0h52xr1

# Get TypeScript

```
λ  npm i typescript -g

λ  tsc app.ts
```

@wizardnet972

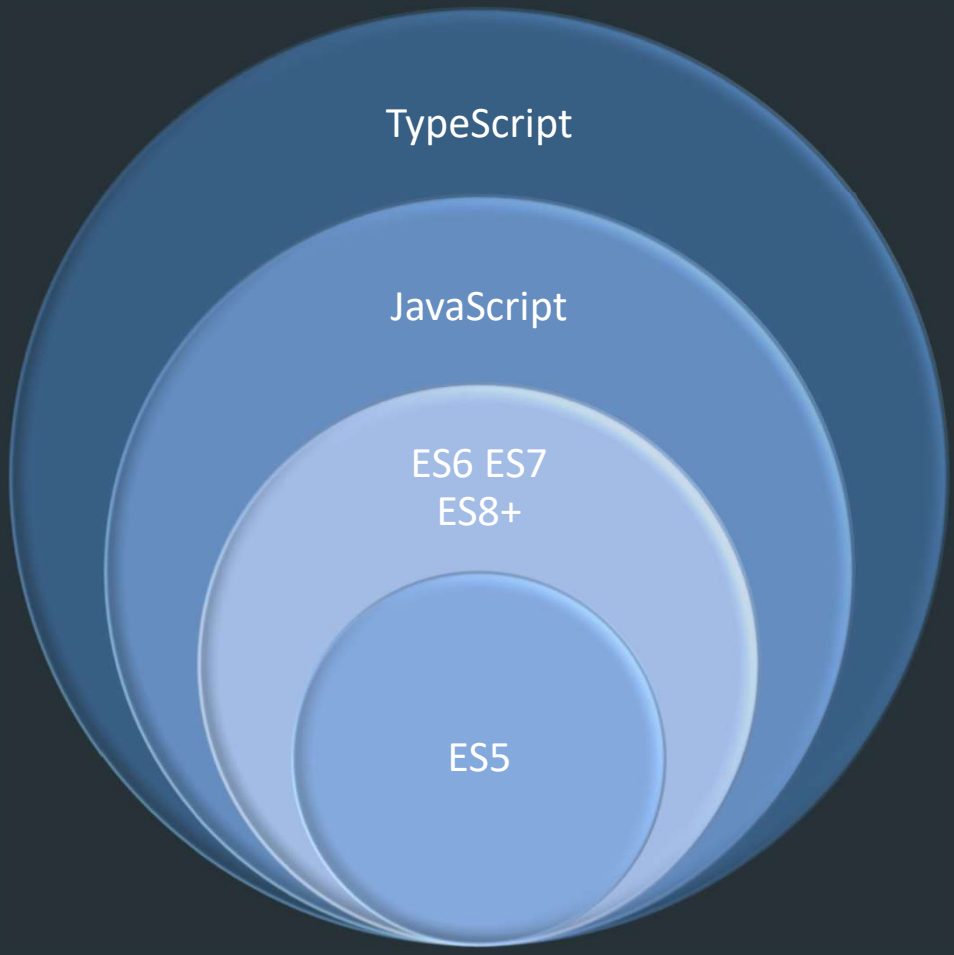
# Typescript

Superset.

Typed language

Any browser, host, OS.

Open Source.



# How does it work?


```typescript
// TYPESCRIPT

class Greeter {
 greeting: string;

 public constructor(message: string) {
  this.greeting = message;
 }

 public greet() {
  return "Hello, " + this.greeting;
 }
}
```

```javascript
// JAVASCRIPT

var Greeter = (function () {

 function Greeter(message) {
   this.greeting = message;
 }

 Greeter.prototype.greet = function () {
  return "Hello, " + this.greeting;
 };

 return Greeter;
}());
```

http://www.typescriptlang.org/play/

# IDEs

## Visual Studio


Visual Studio 2017


Visual Studio Code


Visual Studio 2015

## And More...


Sublime Text


Atom


Eclipse


Emacs

WebStorm

Vim

# Typescript

Full support with Nodejs

Compile

What is Babel?

What is Webpack? Why?

# Recap

- Open source language that compiles into JavaScript
- Code encapsulation
- Maintainable code
- Tooling support


Application scale JavaScript development is hard,

TypeScript makes it easier

@wizardnet972

# Typescript with Nodejs

Demo


# The Web Server

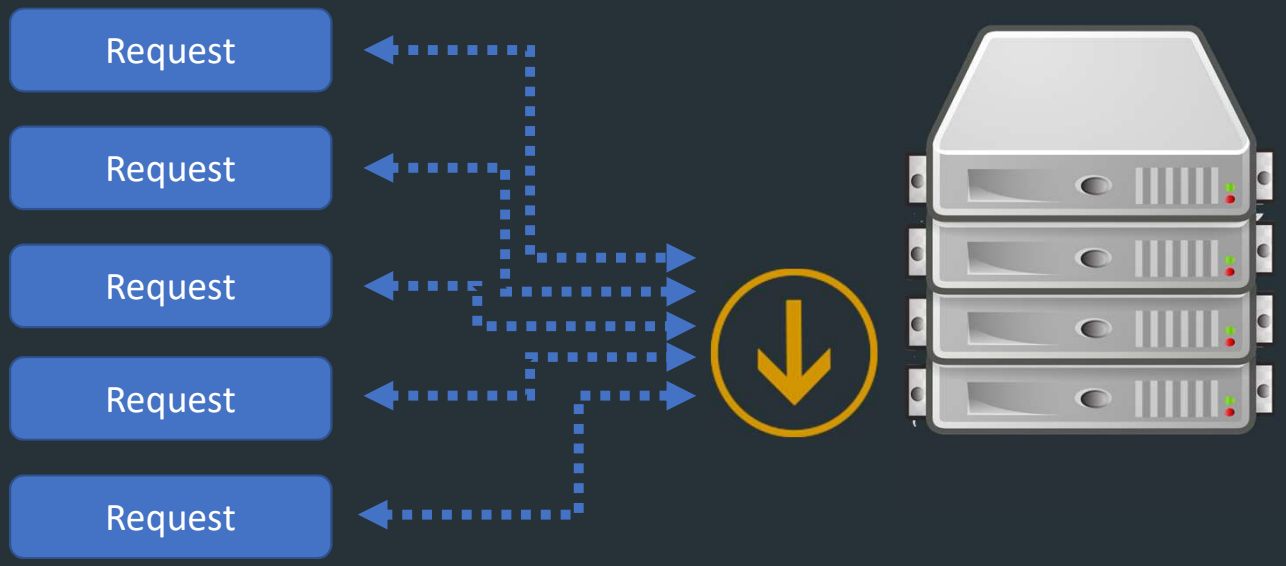
Meet Express.js

@wizardnet972

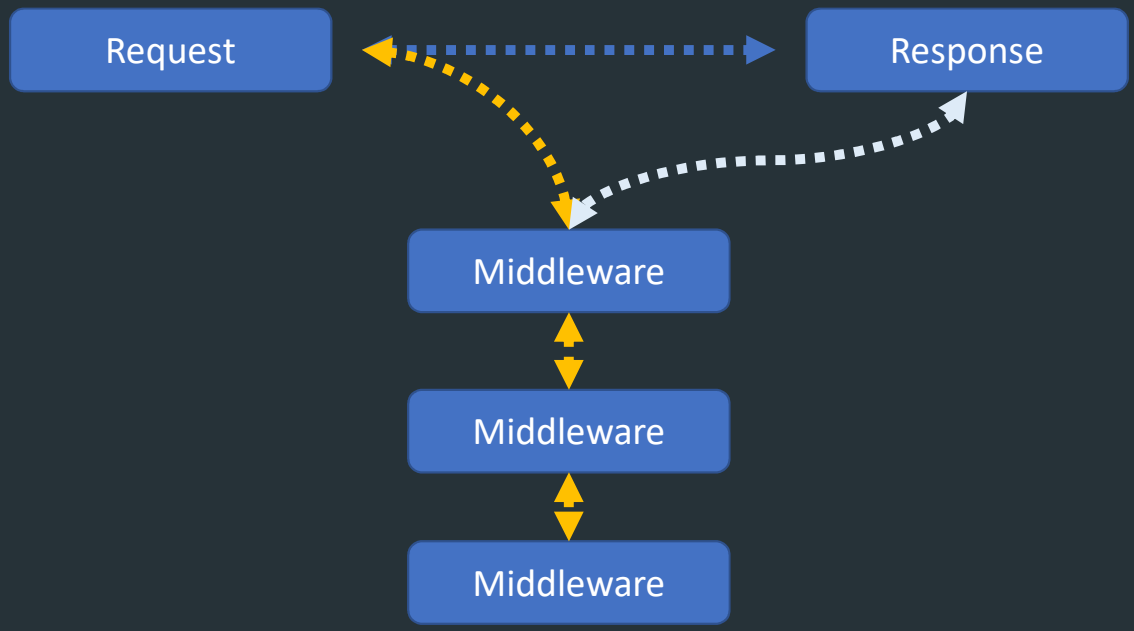
# Traditional Web Server Model

@wizardnet972

# Node Web Server Model



# Express.js



@wizardnet972

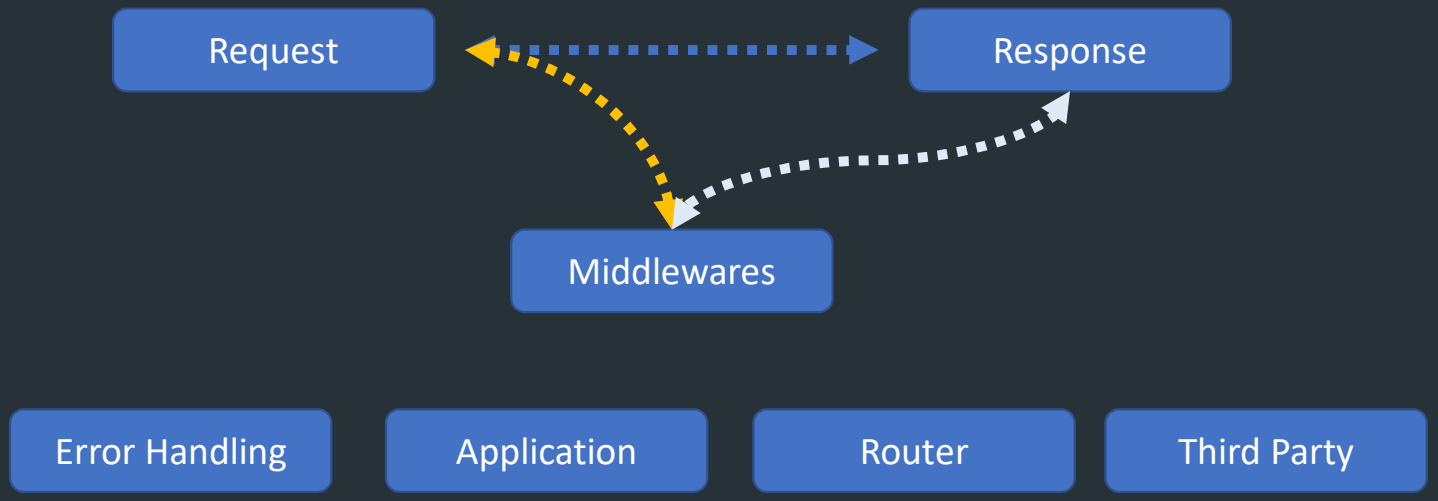
```
λ  npm i express
```

```
import express from 'express';

const app = express();
```

# Express.js

- Routing
- Static files and SPA

# Express.js – Middlewares



@wizardnet972

```
app.use((req, res, next) => {
 console.log('This is a middleware!');
 next();
});

app.use((req, res, next) => {
 console.log('This is a another middleware!');
 next();
});

app.use((req, res, next) => {
 console.log('Well, now is time to send a response!');
 res.send('hello!');
});
```

application level middleware

```
var app = express();
var router = express.Router();

router.use((req, res, next) => {
 if (!req.headers['x-auth']) return next('router')
 next()
})

router.get('/', (req, res) => {
 res.send('hello, user!')
})

app.use('/admin', router, (req, res) => {
 res.sendStatus(401)
});
```

router level middleware



```
app.use((err, req, res, next) => {
 res.status(500).send({ err });
}
```

error handle
level
middleware

```
var express = require('express');
var app = express();
var cookieParser = require('cookie-parser');

app.use(cookieParser());
```

Third-party level middleware

# Let’s add express to our project

# Debugging

# Rest API

# Rest API

- REST = REpresentational State Transfer.
- Http Protocol.
- First introduced in 2000.
- Everything is a resource.
- Rest is endpoints
- Stateless

# Rest application: 1000ft view…

http://www.my-website/

http://www.my-website/login

http://www.my-website/register

http://www.my-website/about

http://www.my-website/api

http://www.my-website/api/login

http://www.my-website/api/register

# Create, read, update and delete

| | | | | |
|---|---|---|---|---|
| POST | Create a new user | router.post('/users', users.create); | INSERT | Write |
| GET | Retrieve all users | router.get('/users', users.findAll); | SELECT | Read / take |
| GET | Retrieve a single User with userId | router.get('/users/:userId', users.findOne); | SELECT | Read / take |
| PUT | Update a User with userId | router.put('/users/:userId', users.update); | UPDATE | Write |
| DELETE | Delete a User with userId | router.delete('/users/:userId', users.delete); | DELETE | dispose |

# Create, read, update and delete

| | | | | |
|---|---|---|---|---|
| POST | Create a new user | router.post('/users', users.create); | INSERT | Write |
| GET | Retrieve all users | router.get('/users', users.findAll); | SELECT | Read / take |
| GET | Retrieve a single User with userId | router.get('/users/:userId', users.findOne); | SELECT | Read / take |
| PUT | Update a User with userId | router.put('/users/:userId', users.update); | UPDATE | Write |
| DELETE | Delete a User with userId | router.delete('/users/:userId', users.delete); | DELETE | dispose |

# Create, read, update and delete

| | | | | |
|---|---|---|---|---|
| POST | Create a new user | router.post('/users', users.create); | INSERT | Write |
| GET | Retrieve all users | router.get('/users', users.findAll); | SELECT | Read / take |
| GET | Retrieve a single User with userId | router.get('/users/:userId', users.findOne); | SELECT | Read / take |
| PUT | Update a User with userId | router.put('/users/:userId', users.update); | UPDATE | Write |
| DELETE | Delete a User with userId | router.delete('/users/:userId', users.delete); | DELETE | dispose |

# Create, read, update and delete

| | | | | |
|---|---|---|---|---|
| POST | Create a new user | router.post('/users', users.create); | INSERT | Write |
| GET | Retrieve all users | router.get('/users', users.findAll); | SELECT | Read / take |
| GET | Retrieve a single User with userId | router.get('/users/:userId', users.findOne); | SELECT | Read / take |
| PUT | Update a User with userId | router.put('/users/:userId', users.update); | UPDATE | Write |
| DELETE | Delete a User with userId | router.delete('/users/:userId', users.delete); | DELETE | dispose |

# Create, read, update and delete

| | | | | |
|---|---|---|---|---|
| POST | Create a new user | router.post('/users', users.create); | INSERT | Write |
| GET | Retrieve all users | router.get('/users', users.findAll); | SELECT | Read / take |
| GET | Retrieve a single User with userId | router.get('/users/:userId', users.findOne); | SELECT | Read / take |
| PUT | Update a User with userId | router.put('/users/:userId', users.update); | UPDATE | Write |
| DELETE | Delete a User with userId | router.delete('/users/:userId', users.delete); | DELETE | dispose |

# Create, read, update and delete

| | | | | |
|---|---|---|---|---|
| POST | Create a new user | router.post('/users', users.create); | INSERT | Write |
| GET | Retrieve all users | router.get('/users', users.findAll); | SELECT | Read / take |
| GET | Retrieve a single User with userId | router.get('/users/:userId', users.findOne); | SELECT | Read / take |
| PUT | Update a User with userId | router.put('/users/:userId', users.update); | UPDATE | Write |
| DELETE | Delete a User with userId | router.delete('/users/:userId', users.delete); | DELETE | dispose |

# Non-rest vs rest

| | | | |
|---|---|---|---|
| POST | Create a new user | /user/create | /user |
| GET | Retrieve a single User with userId | /user/get?id=1 | /user/1 |
| PUT | Update a User with userId | /user/save?id=1 | /user/1 |
| DELETE | Delete a User with userId | /user/delete?id=1 | /user/1 |

# Non-rest vs rest

| | | | |
|---|---|---|---|
| POST | Create a new user | /user/create | /user |
| GET | Retrieve a single User with userId | /user/get?id=1 | /user/1 |
| PUT | Update a User with userId | /user/save?id=1 | /user/1 |
| DELETE | Delete a User with userId | /user/delete?id=1 | /user/1 |

# Non-rest vs rest

| | | | |
|---|---|---|---|
| POST | Create a new user | /user/create | /user |
| GET | Retrieve a single User with userId | /user/get?id=1 | /user/1 |
| PUT | Update a User with userId | /user/save?id=1 | /user/1 |
| DELETE | Delete a User with userId | /user/delete?id=1 | /user/1 |

# Non-rest vs rest

| | | | |
|---|---|---|---|
| POST | Create a new user | /user/create | /user |
| GET | Retrieve a single User with userId | /user/get?id=1 | /user/1 |
| PUT | Update a User with userId | /user/save?id=1 | /user/1 |
| DELETE | Delete a User with userId | /user/delete?id=1 | /user/1 |

# Non-rest vs rest

| | | | |
|---|---|---|---|
| POST | Create a new user | /user/create | /user |
| GET | Retrieve a single User with userId | /user/get?id=1 | /user/1 |
| PUT | Update a User with userId | /user/save?id=1 | /user/1 |
| DELETE | Delete a User with userId | /user/delete?id=1 | /user/1 |

# API Route


```
var createUser = (req, res, next) => {
 var userName = req.body;
 req.user = getUserFromDB(userName);
 next();
}
var sendEmailToAdmin = (req, res, next) => {
 sendEmailToAdmin(req.user);
 next()
}
var login = (req, res) => {
 res.json({ token: getAccessToken(req.user) });
}
```

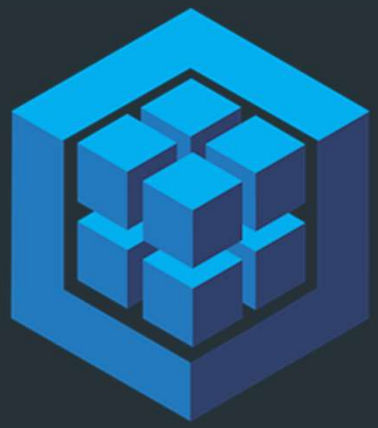

```
var createUser = (req, res, next) => { ... }
var sendEmailToAdmin = (req, res, next) => { ... }
var login = (req, res) => { ... }

const register = [createUser, sendEmailToAdmin, login];
app.post('/users', register);
```


```
app.route('/users/:userId')
   .get((req, res) => {  res.send('Get the user’) })
   .post((req, res) => { res.send('Add new user’) })
   .put((req, res) => { res.send('Update the user’) })
   .delete((req, res) => { res.send('Delete the user’) });
```

# Add api to our project

# Database

@wizardnet972



# Sequelize

http://docs.sequelizejs.com/

# Sequelize

- Sequelize is a promise-based ORM
- PostgreSQL, MySQL, SQLite and MSSQL
- Each module (table) has own module

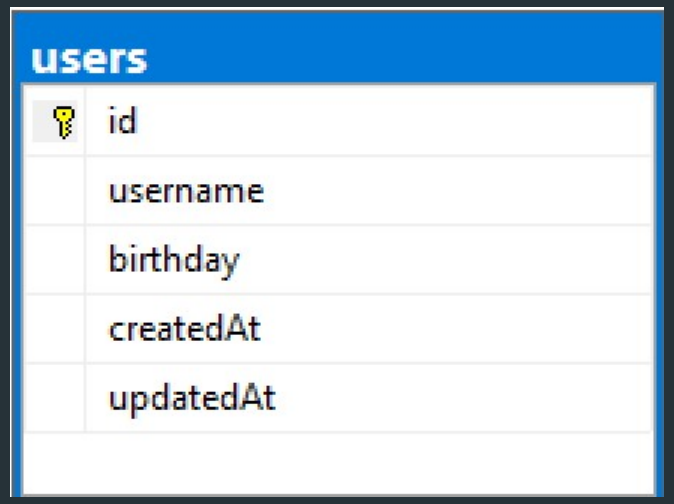
# Get Started

```
λ  npm install --save sequelize

λ  npm install --save tedious
```

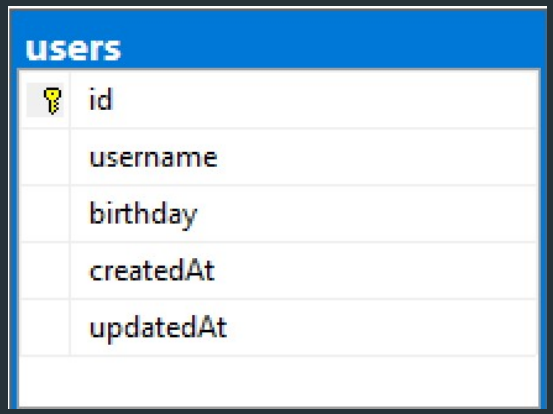
```
const Sequelize = require('sequelize');

const sequelize = new Sequelize(
'Server=myServerAddress;Database=myDataBase;Trusted=True;');

// or

const sequelize = new Sequelize('database', 'username', 'password', {
 host: 'localhost',
 dialect: 'mssql',
 pool: {  max: 5,  min: 0,  acquire: 30000,  idle: 10000 }
});
```

```
sequelize
 .authenticate()
 .then(() => {
   console.log('Connection has been established successfully.');
 })
 .catch(err => {
   console.error('Unable to connect to the database:', err);
 });
```

```
sequelize.define('tableName', {attributes}, {options})

const User = sequelize.define('user', {
 username: Sequelize.STRING,
 birthday: Sequelize.DATE
});
```

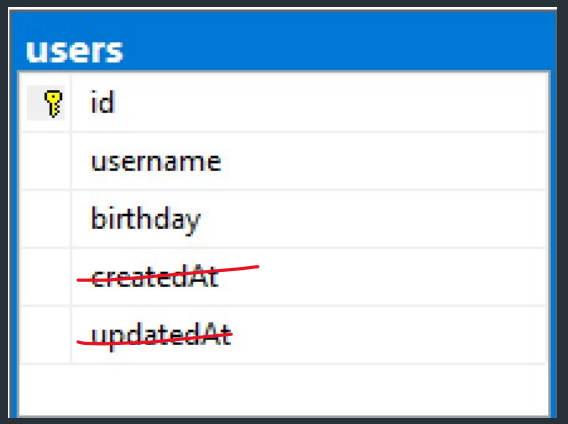


```
const User = sequelize.define('user', { ... });

sequelize
    .sync({ force: true })
    .then(() => {
      User.create({
       username: 'shlomi',
       birthday: new Date(1985, 11, 10)
      });
    })
    .catch(err => {
     console.log('opps');
    });
```



@wizardnet972

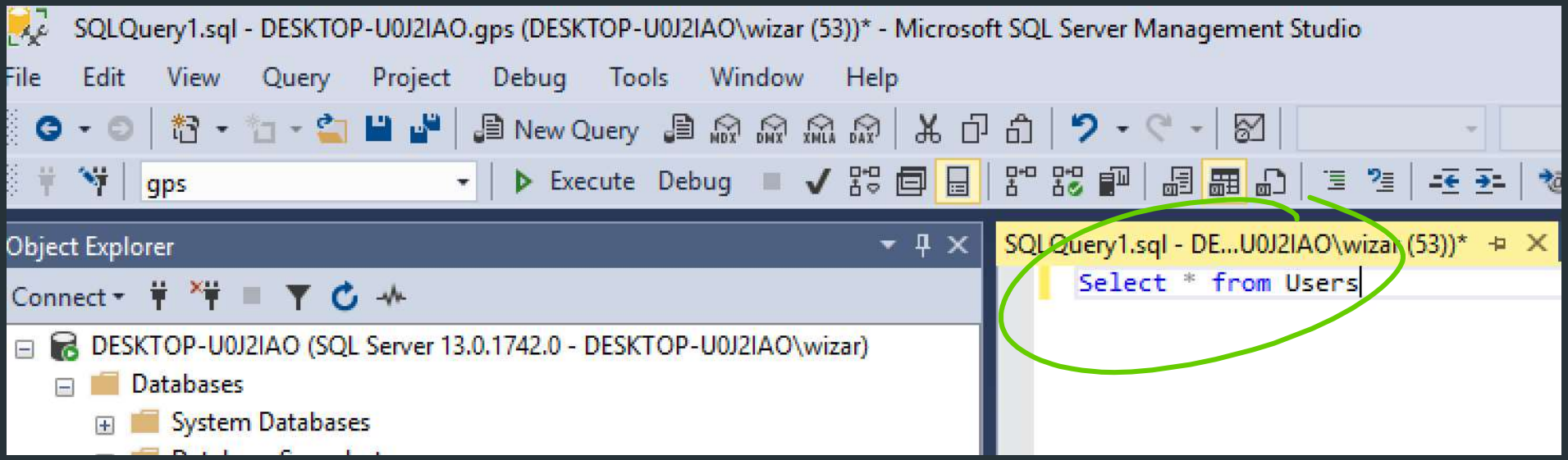
```
const sequelize = new Sequelize('connectionstring', {
 define: {
  timestamps: false
 }
});

const User = sequelize.define('users', {});

const Post = sequelize.define('posts', {}, {
 timestamps: true
});
```




```
const sequelize = new Sequelize('connectionstring', {
 define: {
  underscored: true,
  id: {
   type: Sequelize.INTEGER,
   autoIncrement: true,
   primaryKey: true,
   allowNull: false
  }
 }
});
```

updateAt → update_at

```
User.findAll().then(users => {
 console.log(users);
})
```



# When to sync?

At the begging

# Promise?

Bluebird!

# Model definition


```
const Project = sequelize.define('project', {
 title: Sequelize.STRING,
 description: Sequelize.TEXT
});


const Task = sequelize.define('task', {
 title: Sequelize.STRING,
 description: Sequelize.TEXT,
 deadline: Sequelize.DATE
})
```

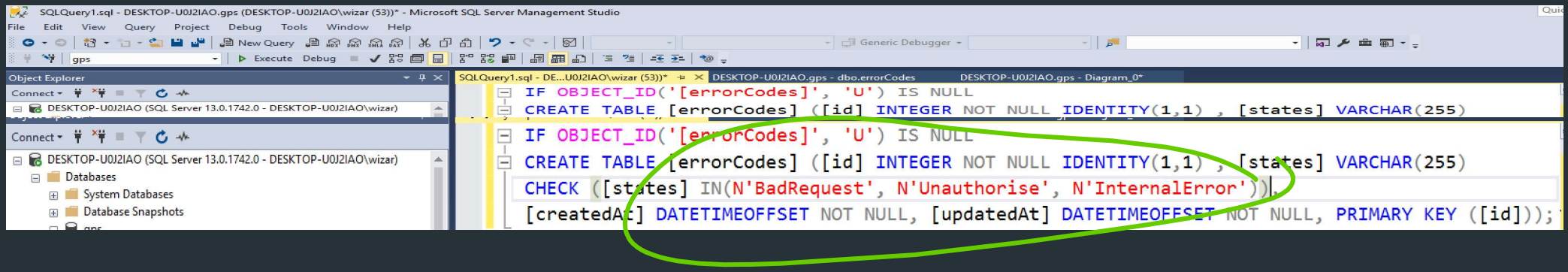
# Delete a table

```
Project.drop()
```

```
Sequelize.STRING // VARCHAR(255)
Sequelize.STRING(1234) // VARCHAR(1234)
Sequelize.STRING.BINARY // VARCHAR BINARY
Sequelize.TEXT // TEXT

Sequelize.INTEGER // INTEGER
Sequelize.BIGINT // BIGINT
Sequelize.BIGINT(11) // BIGINT(11)

Sequelize.FLOAT // FLOAT
Sequelize.DOUBLE // DOUBLE
Sequelize.DECIMAL // DECIMAL

Sequelize.DATE // DATETIME for mysql / sqlite, TIMESTAMP WITH TIME ZONE for postgres
Sequelize.DATEONLY // DATE without time.
Sequelize.BOOLEAN // TINYINT(1)

Sequelize.ENUM('value 1', 'value 2') // An ENUM with allowed values 'value 1' and 'value 2'
```

@wizardnet972

```
sequelize.define('errorCodes', {
 states: {
  type: Sequelize.ENUM,
  values: ['BadRequest', 'Unauthorise', 'InternalError']
 }
})
```



# Naming strategy

```
const Project = sequelize.define('project', attributes,
{
 name: {
  singular: 'task',
  plural: 'tasks'
 }
})

User.belongsToMany(Project);
```

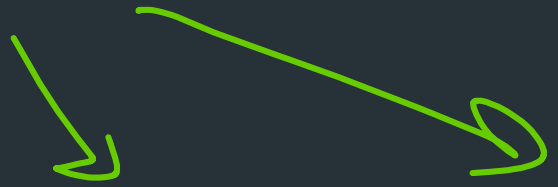
This will add the functions add/set/get Tasks to user instances.

# Lazy loading


```
module.exports = (sequelize, DataTypes) => {

 return sequelize.define("project", {
  name: DataTypes.STRING,
  description: DataTypes.TEXT
 })

}

const Project = sequelize.import(__dirname + "/path/to/models/project")
```

# Associations

# Associations



One to Many

Many to Many

# One-To-Many

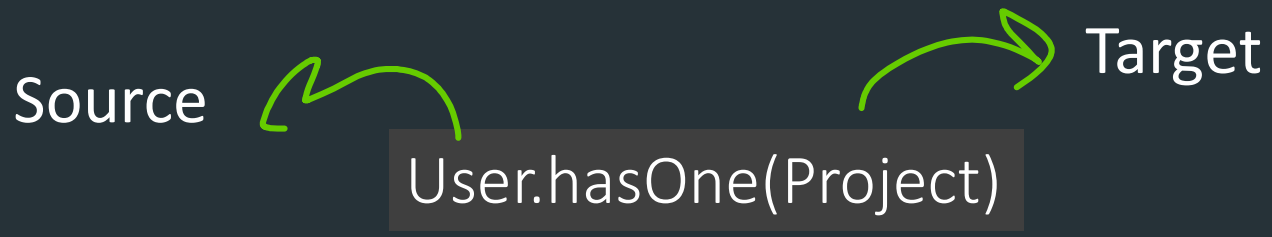
ONE customer MANY orders

ONE category MANY products

ONE player MANY teams

ONE project MANY tasks

# Associations



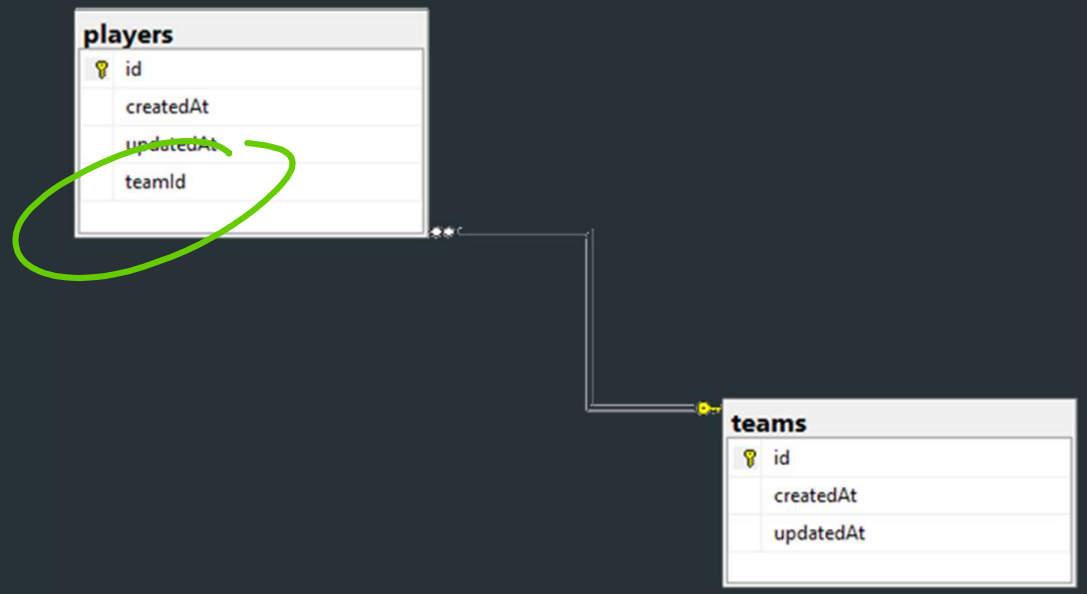
@wizardnet972

# one-to-one & many-to-one

```
const Player = sequelize.define('player', { });
const Team = sequelize.define('team', { });

Player.belongsTo(Team);
```

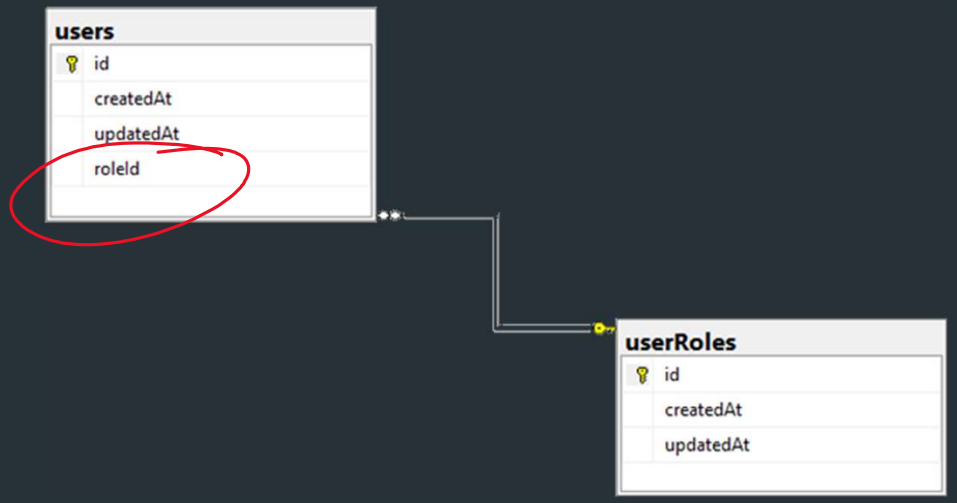
Source

Target



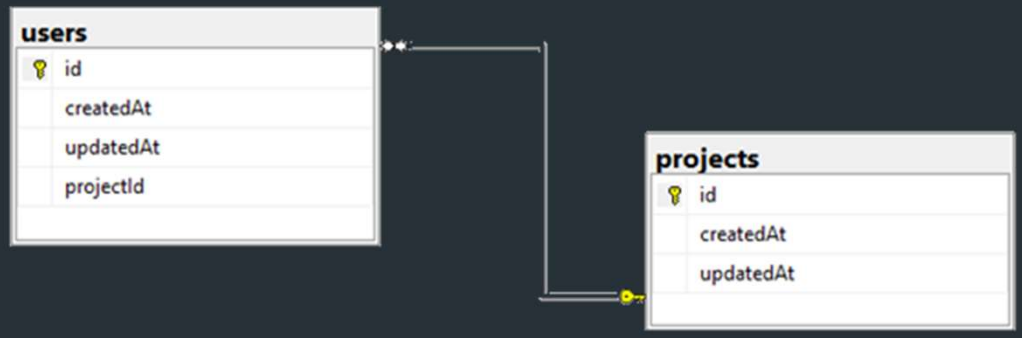
# Alias

```
const User = this.sequelize.define('user', {})
const UserRole = this.sequelize.define('userRole', {});

User.belongsTo(UserRole, {as: 'role'});
```



# Has one

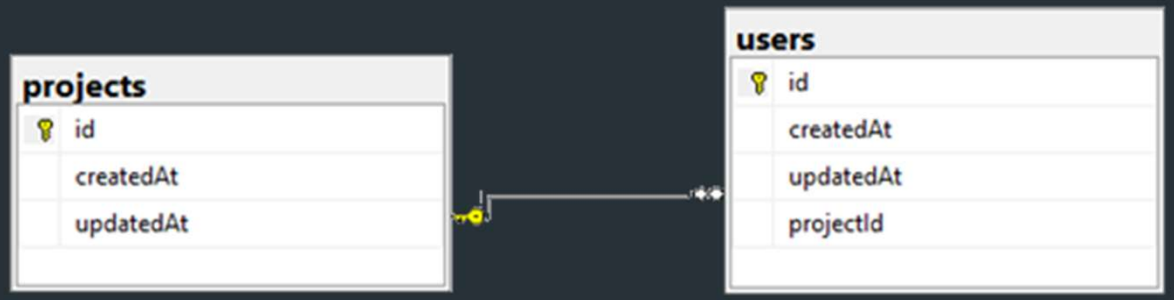
```
const User = db.define('user', {/* ... */})
const Project = db.define('project', {/* ... */})

Project.hasOne(User);
```



# HasOne vs BelongsTo

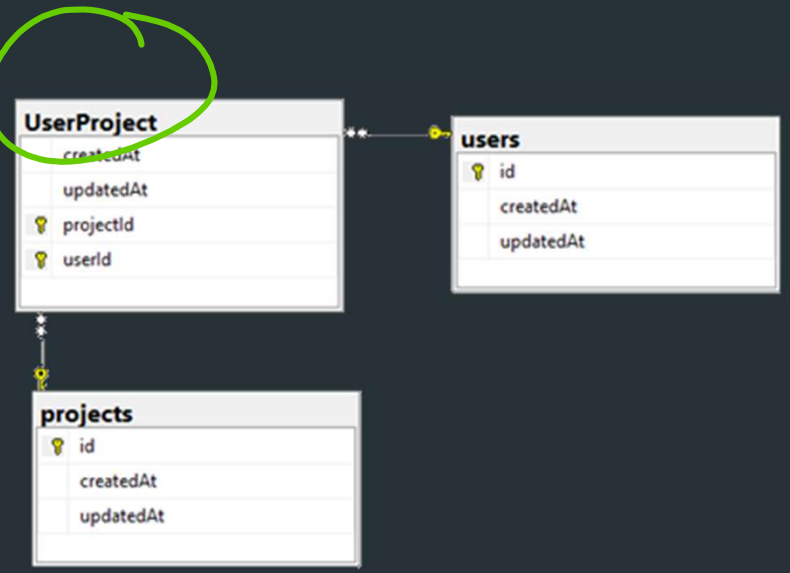
# Has many

```
const User = sequelize.define('user', {/* ... */})
const Project = sequelize.define('project', {/* ... */})

Project.hasMany(User, {as: 'Workers'})


var myProject = new Project();

myProject.getWorkers();

myProject.setWorkers();
```



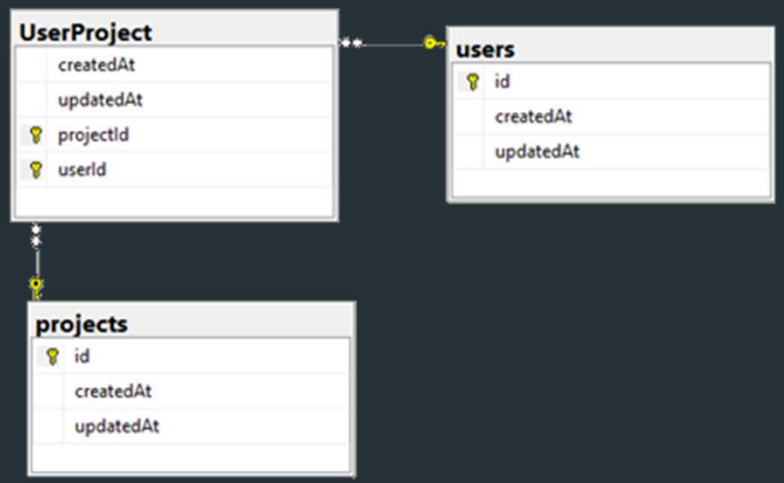
```
const User = sequelize.define('user', {/* ... */})
const Project = sequelize.define('project', {/* ... */})

Project.belongsToMany(User, {through: 'UserProject'});

User.belongsToMany(Project, {through: 'UserProject'});
```

Create Authomatic!



@wizardnet972

```
const User = sequelize.define('user', {})
const Project = sequelize.define('project', {})
const UserProjects = sequelize.define('userProjects', {
  status: Sequelize.STRING
})

User.belongsToMany(Project, { through: UserProjects })
Project.belongsToMany(User, { through: UserProjects })
```



# Create new row

```
User
.build({ email: 'foo@gmail.com', description: 'bar', expire: new Date() })
.save()
.then(anotherTask => {
 ...
})
.catch(error => {
 // Ooops, do some error-handling
})
```

```
User.findById(68383).then(user => { ... });

User.findOne({ where: { email: 'wizardnet972@gmail.com'} })
    .then(user => { ... });

User.findOne({
 where: { email: 'wizardnet972@gmail.com'},
 attributes: ['id', ['name']]
}).then(user => {
 // user == { id, name };
});
```


```
var newUser = { email: '....', name, ... };

Users.findOrCreate({
  where: { email: newUser.email },
  defaults: newUser
})
.then(user => { ... });
```


```
Users.findAll().then(users => { ... })

Users.findAll({ where: { lastName: 'Levi' } }).then(users => { ... })

Users.findAll({ where: { id: [12,42,54] } }).then(users => { ... })
```

```
Users.findAll({ limit: 10 });

Users.findAll({ offset: 10 });

Users.findAll({ offset: 10, limit: 2 });

Users.findAll({ order: 'lastName DESC' });

Users.findAll({ group: 'security' });
```


```
User.findAll({ include: [ Task ] }).then(users => {

    console.log(JSON.stringify(users))

    /* [{
      "name": "John Doe",
      "id": 1,
      "createdAt": "2013-03-20T20:31:45.000Z",
      "updatedAt": "2013-03-20T20:31:45.000Z",
      "tasks": [{
        "name": "A Task",
        "id": 1,
        "createdAt": "2013-03-20T20:31:40.000Z",
        "updatedAt": "2013-03-20T20:31:40.000Z",
        "userId": 1
       }]
     }] */
});
```

# Include everything

```
include: [{ all: true, nested: true }]
```

# How to run your query

```
sequelize
 .query("SELECT * FROM users", { type: sequelize.QueryTypes.SELECT})
 .then(users => {
   console.log('users', users);
 });
```

```
users [ { usersID: 1,
name: 'id.magna@diamdictum.co.uk',
email: 'lorem.auctor@cubiliaCurae.com' },
{ usersID: 2,
name: 'rutrum@Cumsociisnatoque.ca',
email: 'porttitor@quamCurabitur.net' },
{ usersID: 3,
name: 'pede.ultrices.a@In.org',
email: 'Lorem@Curabiturconsequatlectus.com' },
{ usersID: 4,
name: 'eget@iaculis.net',
email: 'lorem@liberodui.org' },
{ usersID: 5,
name: 'elit.pellentesque@molestie.com',
email: 'lectus.Nullam.suscipit@egestas.ca' },
{ usersID: 6,
name: 'nunc.sit.amet@tellus.edu',
email: 'amet.consectetuer@nonsollicitudin.edu' },
{ usersID: 7,
name: 'senectus.et@sitamet.net',
email: 'dignissim@Namporttitorscelerisque.org' },
{ usersID: 8,
name: 'purus@Nam.ca',
email: 'a@Nunccommodoauctor.net' }
```

# How to pass params to your query

```
sequelize
 .query('SELECT * FROM users WHERE email = :email ', {
    replacements: { email: 'wizardnet972@gmail.com' },
    type: sequelize.QueryTypes.SELECT
 })
 .then(users => {
   console.log(users);
 });
```

# With (NoLock)

```
Users.findAll({
    // ...,
    tableHint: Sequelize.TableHints.NOLOCK
})
```

# Creating multiple rows at once

```
User.bulkCreate([
    { username: 'shlomi', isAdmin: true },
    { username: 'foo', isAdmin: true },
    { username: 'bar', isAdmin: false }
]).then(() => {
    return User.findAll();
}).then(users => {
     console.log(users)
})
```

@wizardnet972

```
User.bulkCreate([
    {email: 'wizardnet972@gmail.com', isAdmin: true},
    {email: 'admin@gmail.com', isPerson: true},
    {email: 'foo@bar.com', isAdmin: false}
]).then(() => {
    return User.update(
        { isAdmin: false },
        { where: { email: 'wizardnet972@gmail.com' }}
    );
}).spread((affectedCount, affectedRows) => {
    return User.findAll();
}).then(users => {
    console.log(users);
})
```

# delete multiple rows at once

```
User.bulkCreate([
 { email: 'wizardnet972@gmail.com', isAdmin: false },
 { email: 'admin@gmail.com', isPerson: true },
 { email: 'foo@bar.com', isAdmin: false }
])
```

# delete multiple rows at once

```
User.bulkCreate([ ... ])

.then(() => {
  return User.destroy({
   where: { email: 'wizardnet972@gmail.com' },
   truncate: true
  });
})
```

# delete multiple rows at once

```
User.bulkCreate([ ...])
.then(() => {
  return User.destroy({ ... });
})
.spread((affectedCount, affectedRows) => {
  return User.findAll();
})
.then(users => {
  console.log(users);
});
```

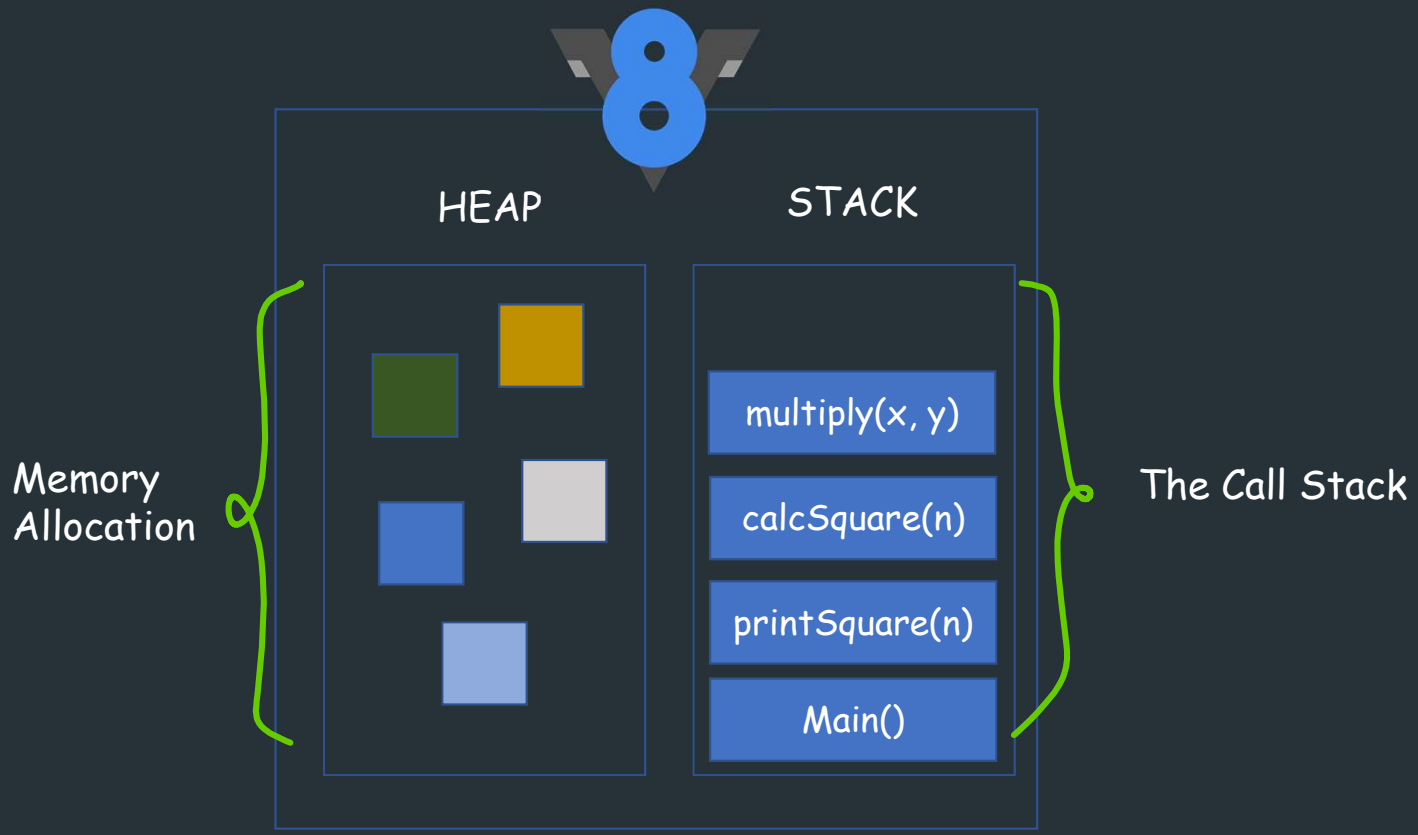
# Demo

@wizardnet972

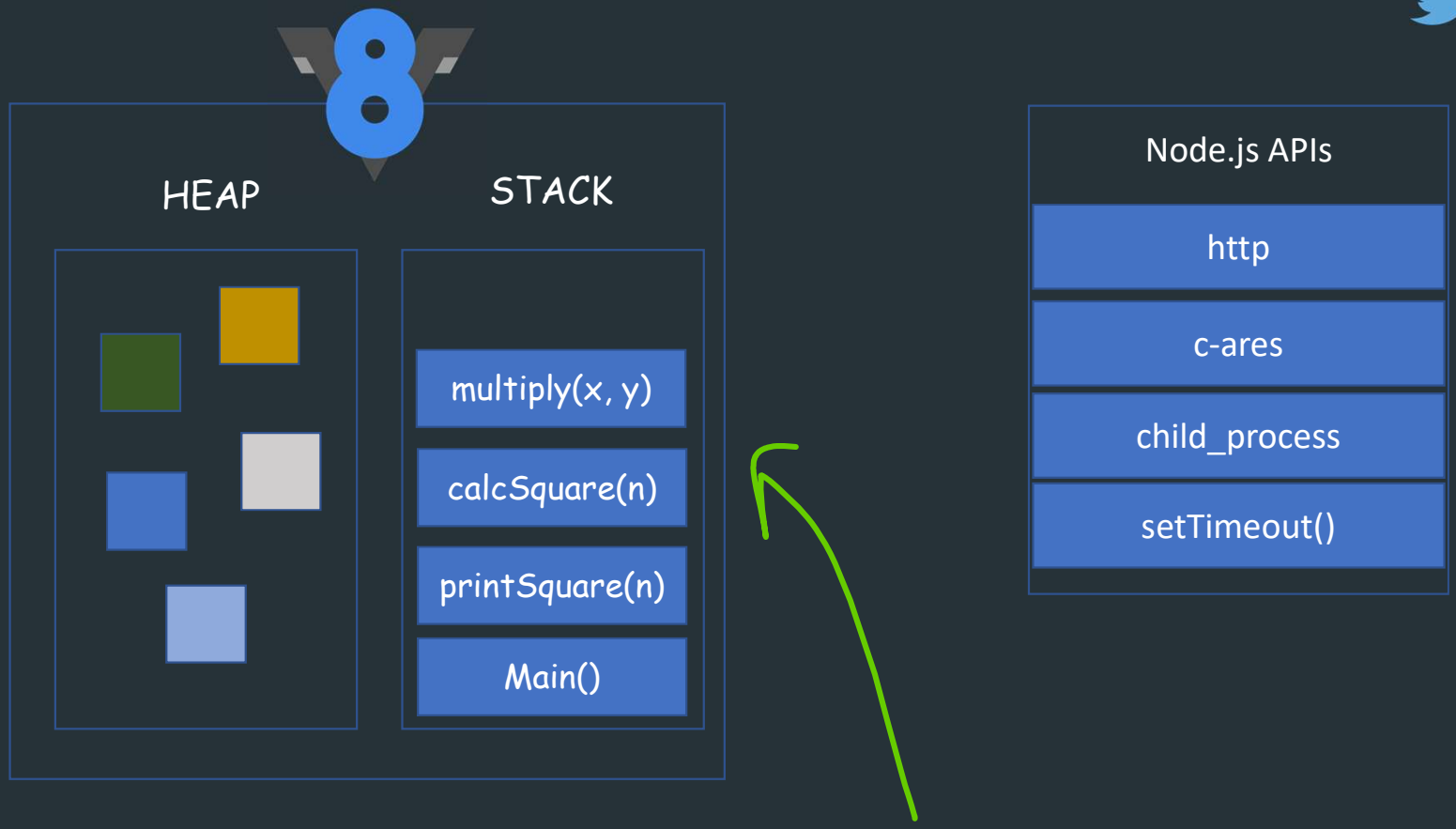
# Node.js core concepts

# The Event Loop

@wizardnet972



@wizardnet972




One thread === One call stack === One thing at a time

```
function multiply(x, y) {
 return x * y;
}

function calcSquare(n) {
 return multiply(n, n);
}

function printSquare(n) {
 var result = calcSquare(n);
 console.log(result);
}

printSquare(2);
```

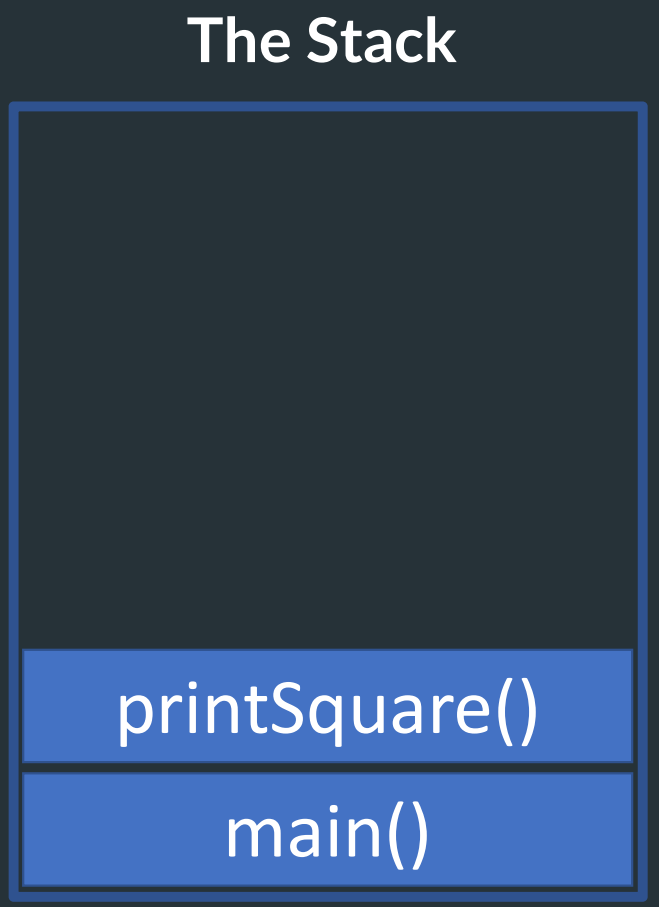
```
function multiply(x, y) {
 return x * y;
}

function calcSquare(n) {
 return multiply(n, n);
}

function printSquare(n) {
 var result = calcSquare(n);
 console.log(result);
}

printSquare(2);
```

```
function multiply(x, y) {
 return x * y;
}

function calcSquare(n) {
 return multiply(n, n);
}

function printSquare(n) {
 var result = calcSquare(n);
 console.log(result);
}

printSquare(2);
```

```
function multiply(x, y) {
 return x * y;
}

function calcSquare(n) {
 return multiply(n, n);
}

function printSquare(n) {
 var result = calcSquare(n);
 console.log(result);
}

printSquare(2);
```

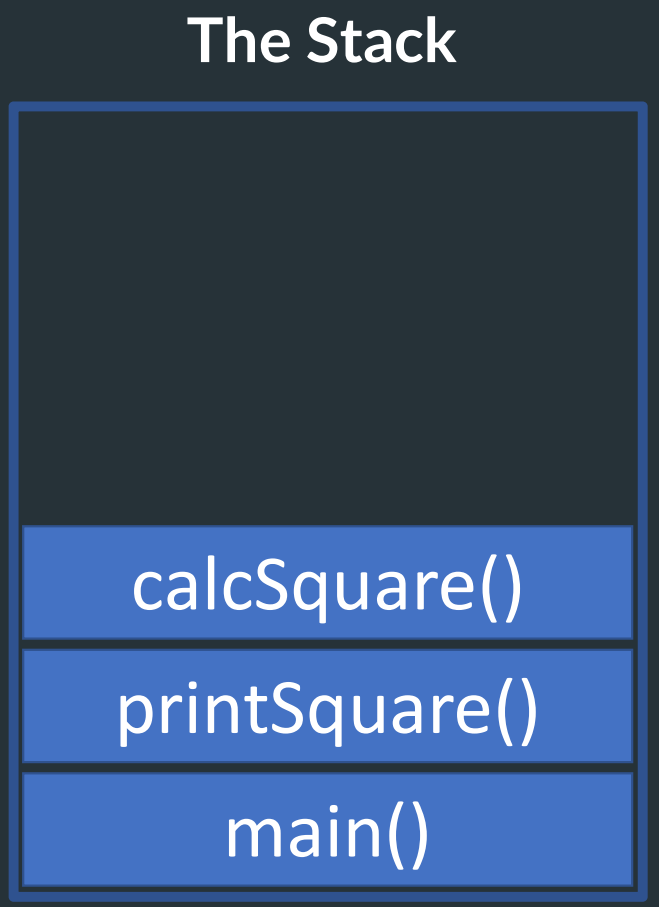
```
function multiply(x, y) {
 return x * y;
}

function calcSquare(n) {
 return multiply(n, n);
}

function printSquare(n) {
 var result = calcSquare(n);
 console.log(result);
}

printSquare(2);
```

```
function multiply(x, y) {
 return x * y;
}

function calcSquare(n) {
 return multiply(n, n);
}

function printSquare(n) {
 var result = calcSquare(n);
 console.log(result);
}

printSquare(2);
```

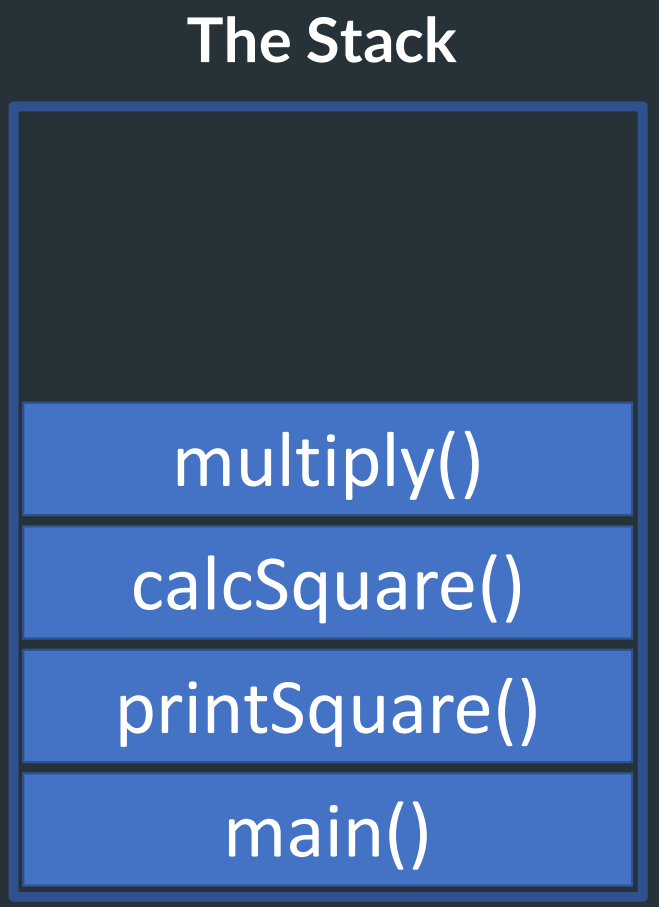


@wizardnet972

```
function multiply(x, y) {
 return x * y;
}

function calcSquare(n) {
 return multiply(n, n);
}

function printSquare(n) {
 var result = calcSquare(n);
 console.log(result);
}

printSquare(2);
```

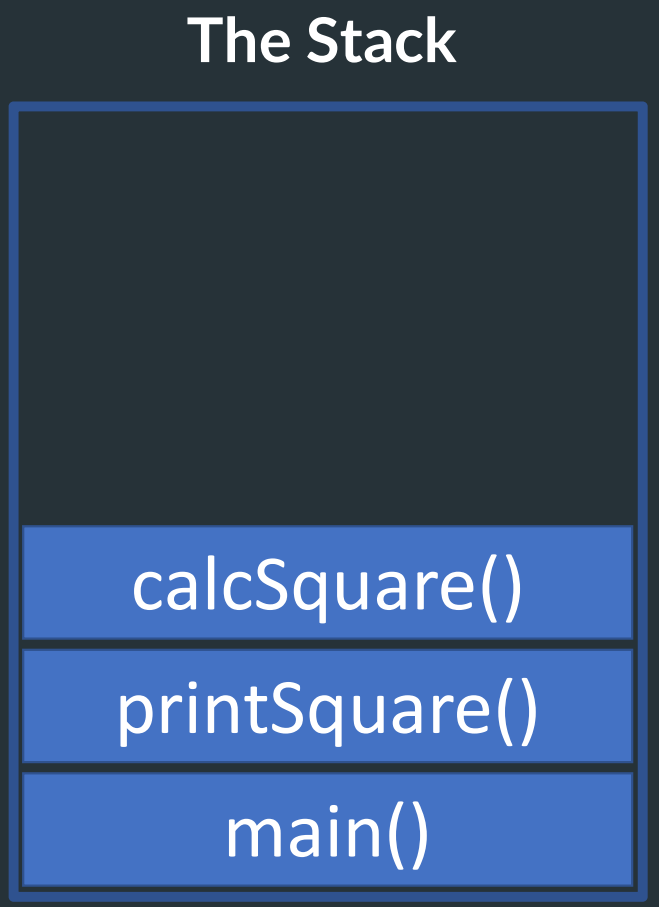


```
function multiply(x, y) {
 return x * y;
}

function calcSquare(n) {
 return multiply(n, n);
}

function printSquare(n) {
 var result = calcSquare(n);
 console.log(result);
}

printSquare(2);
```

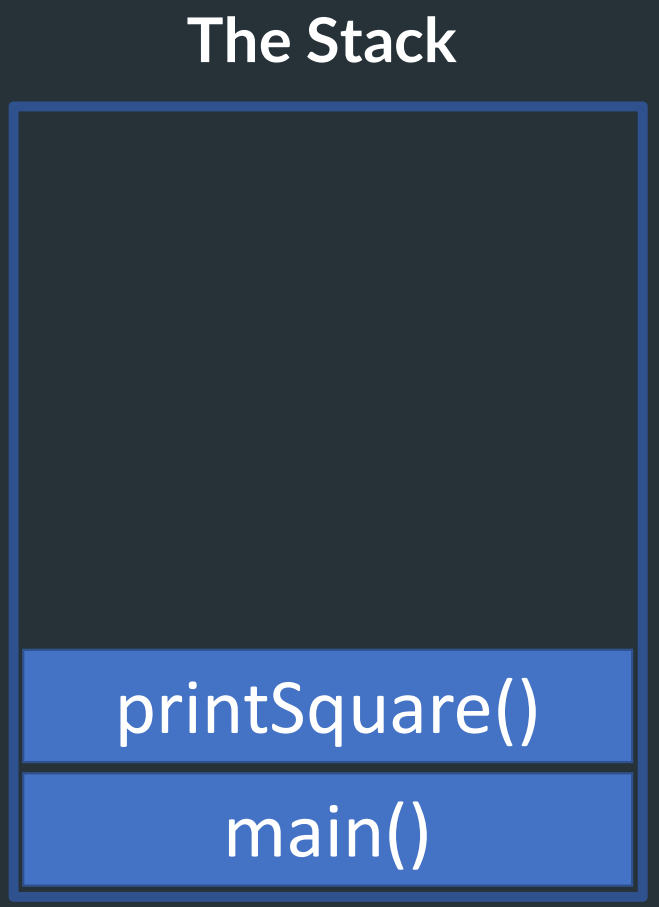


@wizardnet972

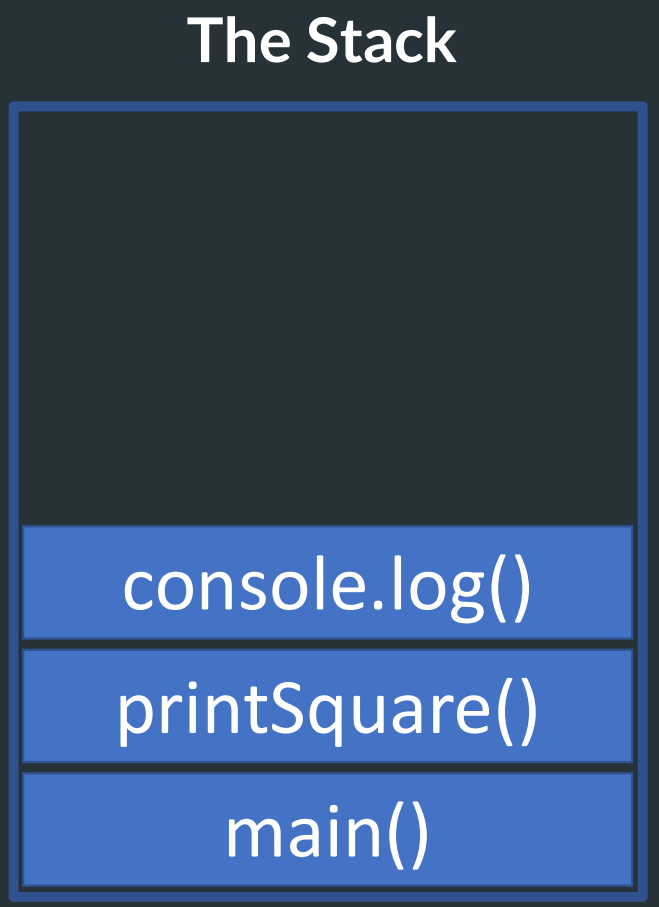
```
function multiply(x, y) {
 return x * y;
}

function calcSquare(n) {
 return multiply(n, n);
}

function printSquare(n) {
 var result = calcSquare(n);
 console.log(result);
}

printSquare(2);
```



```
function multiply(x, y) {
 return x * y;
}

function calcSquare(n) {
 return multiply(n, n);
}

function printSquare(n) {
 var result = calcSquare(n);
 console.log(result);
}

printSquare(2);
```



```
function multiply(x, y) {
 return x * y;
}

function calcSquare(n) {
 return multiply(n, n);
}

function printSquare(n) {
 var result = calcSquare(n);
 console.log(result);
}

printSquare(2);
```



```
function multiply(x, y) {
 return x * y;
}

function calcSquare(n) {
 return multiply(n, n);
}

function printSquare(n) {
 var result = calcSquare(n);
 console.log(result);
}

printSquare(2);
```

```
function multiply(x, y) {
 return x * y;
}

function calcSquare(n) {
 return multiply(n, n);
}

function printSquare(n) {
 var result = calcSquare(n);
 console.log(result);
}

printSquare(2);
```

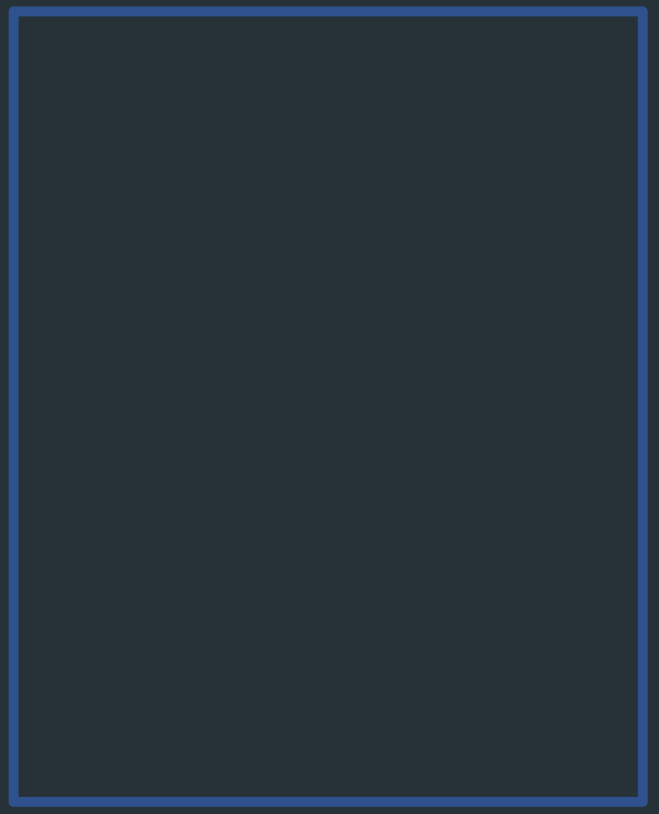
## The Stack

# Another example

@wizardnet972

```
function fn() {
 return fn();
}

fn();
```

## The Stack



@wizardnet972

```
function fn() {
 return fn();
}

fn();
```

## The Stack

@wizardnet972

```
function fn() {
 return fn();
}

fn();
```

## The Stack

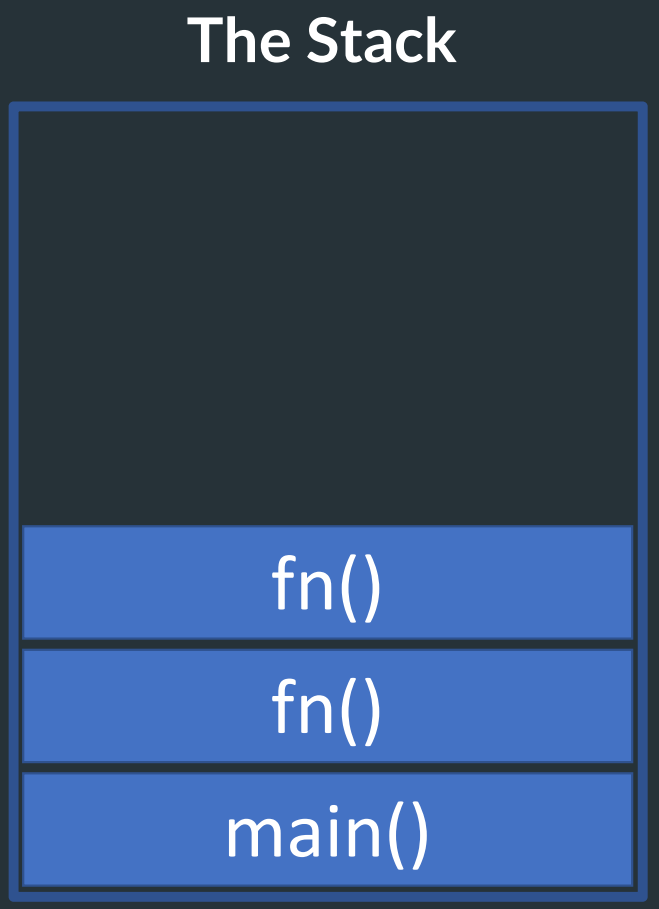
```
function fn() {
 return fn();
}

fn();
```


```
function fn() {
 return fn();
}

fn();
```

## The Stack

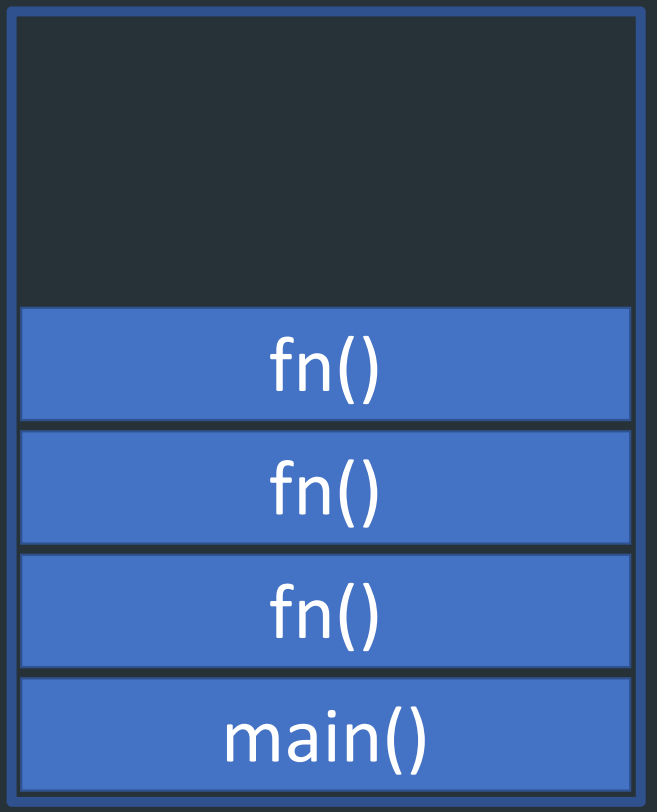
@wizardnet972

```
function fn() {
 return fn();
}

fn();
```

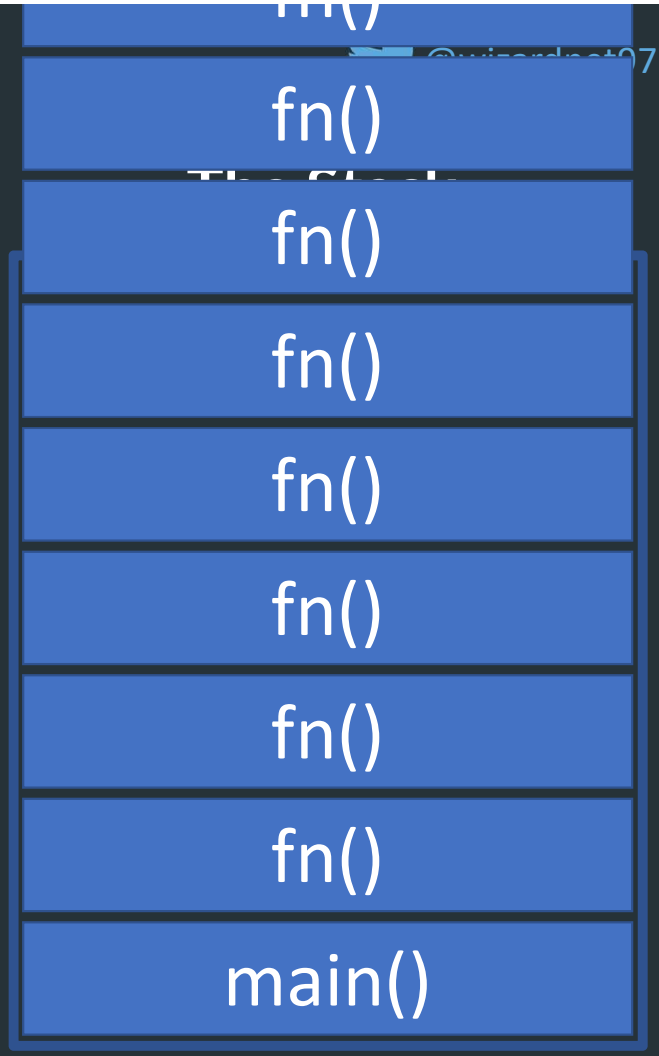


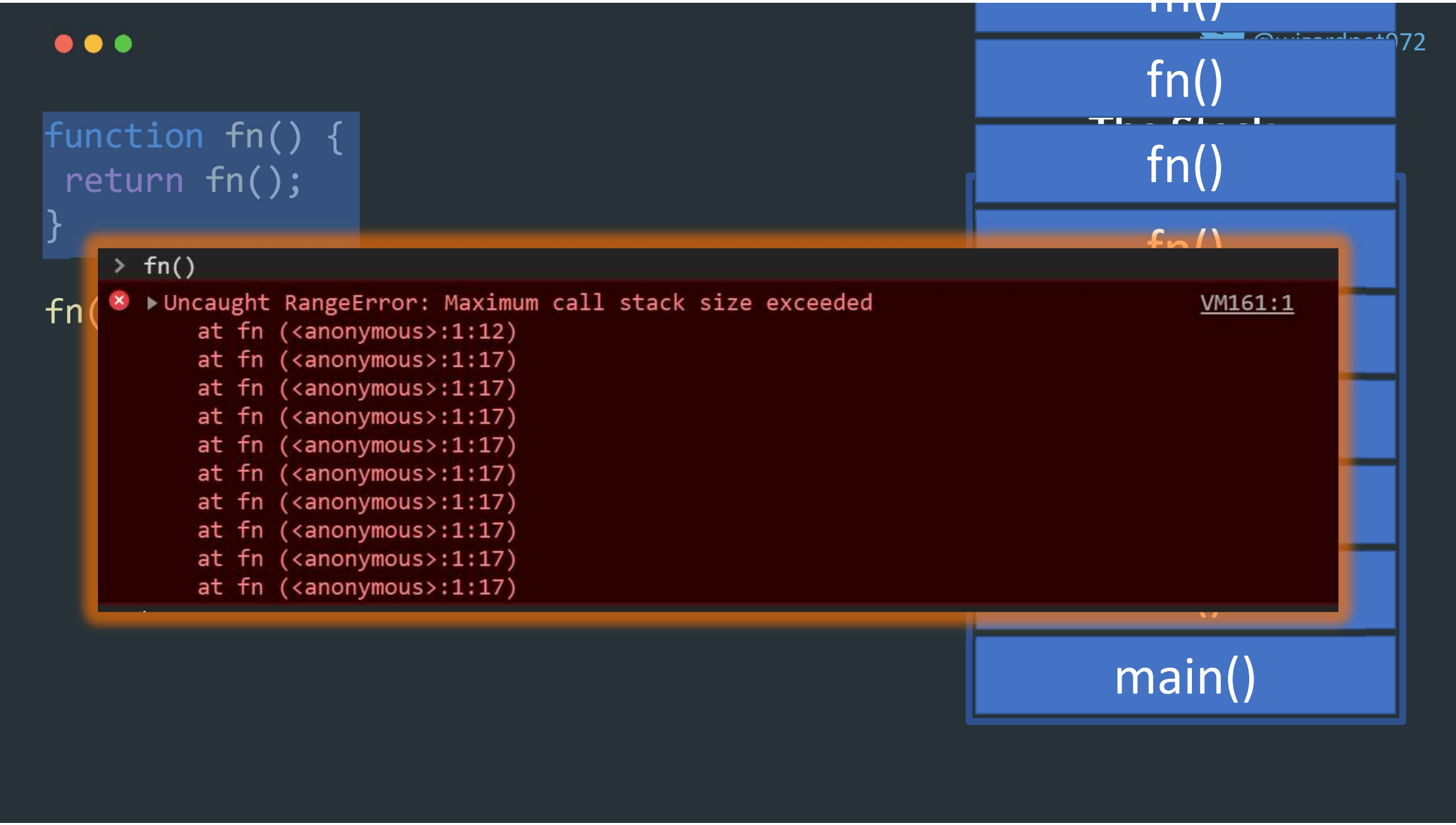

```
function fn() {
 return fn();
}

fn();
```

## The Stack



```
function fn() {
 return fn();
}

fn();
```



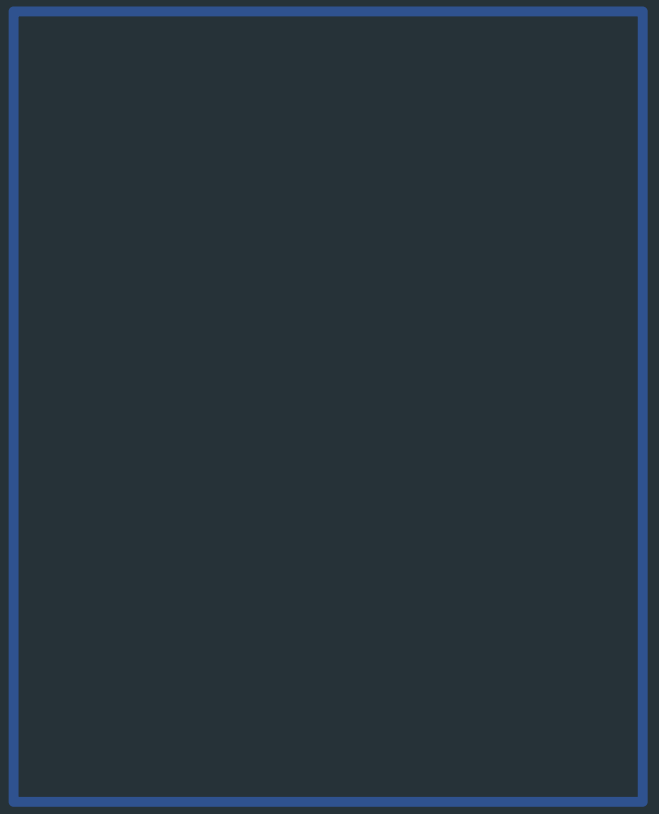

function fn() {
return fn();
}
fn(
> fn()
Uncaught RangeError: Maximum call stack size exceeded
VM161:1
at fn (<anonymous>:1:12)
at fn (<anonymous>:1:17)
at fn (<anonymous>:1:17)
at fn (<anonymous>:1:17)
at fn (<anonymous>:1:17)
at fn (<anonymous>:1:17)
at fn (<anonymous>:1:17)
at fn (<anonymous>:1:17)
at fn (<anonymous>:1:17)
at fn (<anonymous>:1:17)
fn()
fn()
main()

# Blocking main thread

- Loops: while, do, for (1...100000000000000000)
- Network request
- Image processing
- Sync functions
- Heavy calculations

```
var users = http.requestSync('/api/users');
var products = http.requestSync('/api/users/products');
var cart = http.requestSync('/api/users/cart');

console.log(users);
console.log(products);
console.log(cart);
```

## The Stack



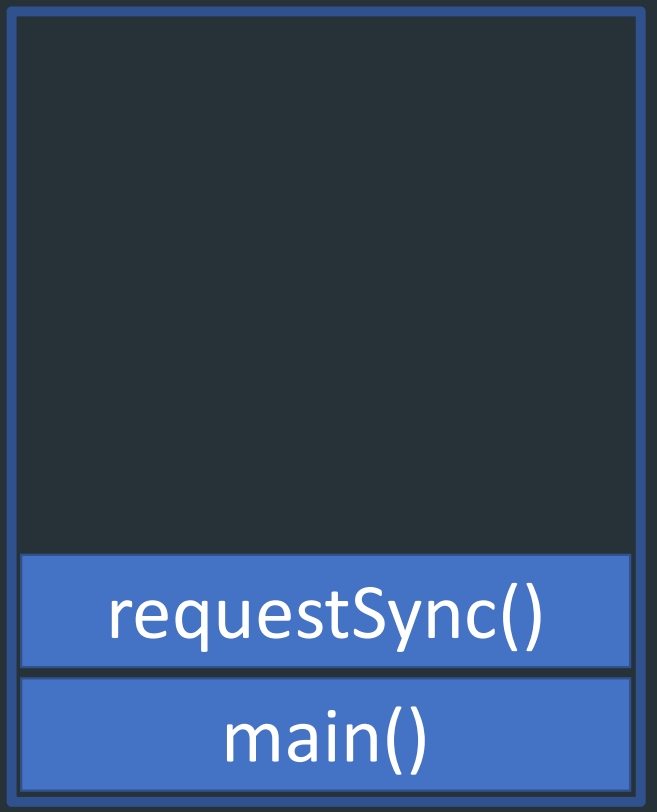
@wizardnet972

```
var users = http.requestSync('/api/users');
var products = http.requestSync('/api/users/products');
var cart = http.requestSync('/api/users/cart');

console.log(users);
console.log(products);
console.log(cart);
```

## The Stack

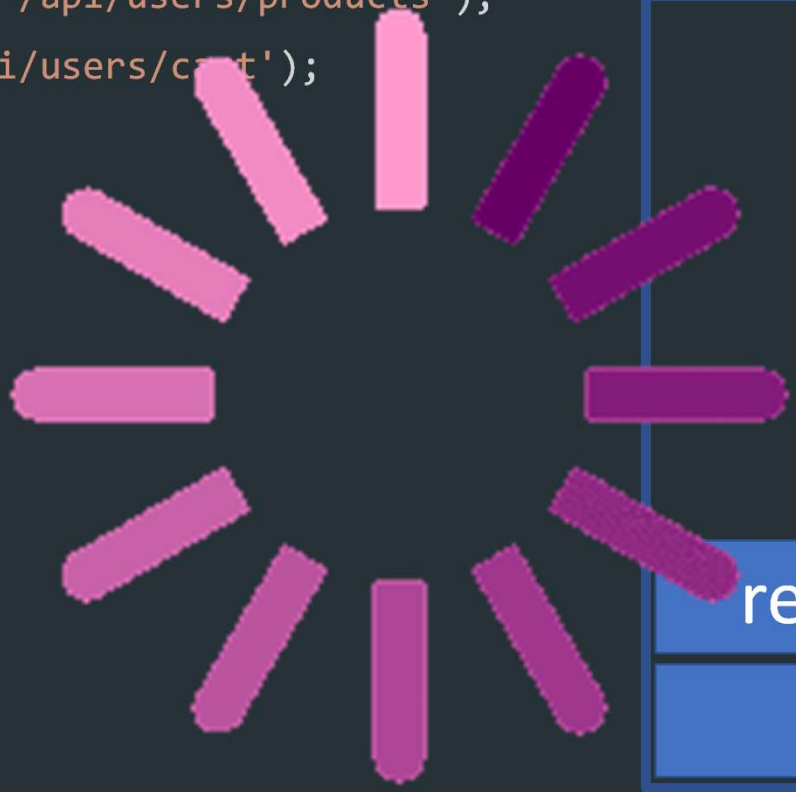
```
var users = http.requestSync('/api/users');
var products = http.requestSync('/api/users/products');
var cart = http.requestSync('/api/users/cart');

console.log(users);
console.log(products);
console.log(cart);
```

## The Stack

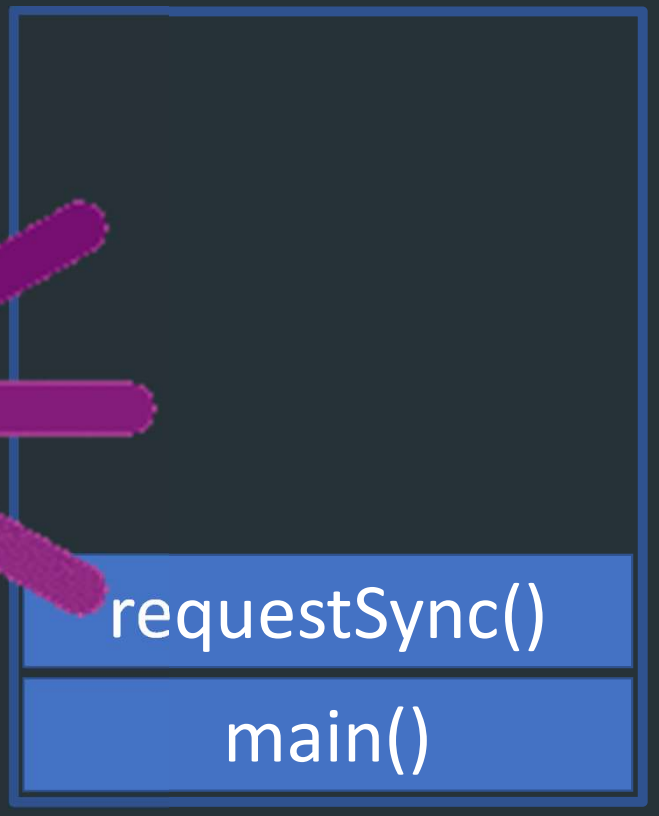


@wizardnet972

```
var users = http.requestSync('/api/users');
var products = http.requestSync('/api/users/products');
var cart = http.requestSync('/api/users/cart');

console.log(users);
console.log(products);
console.log(cart);
```



## The Stack

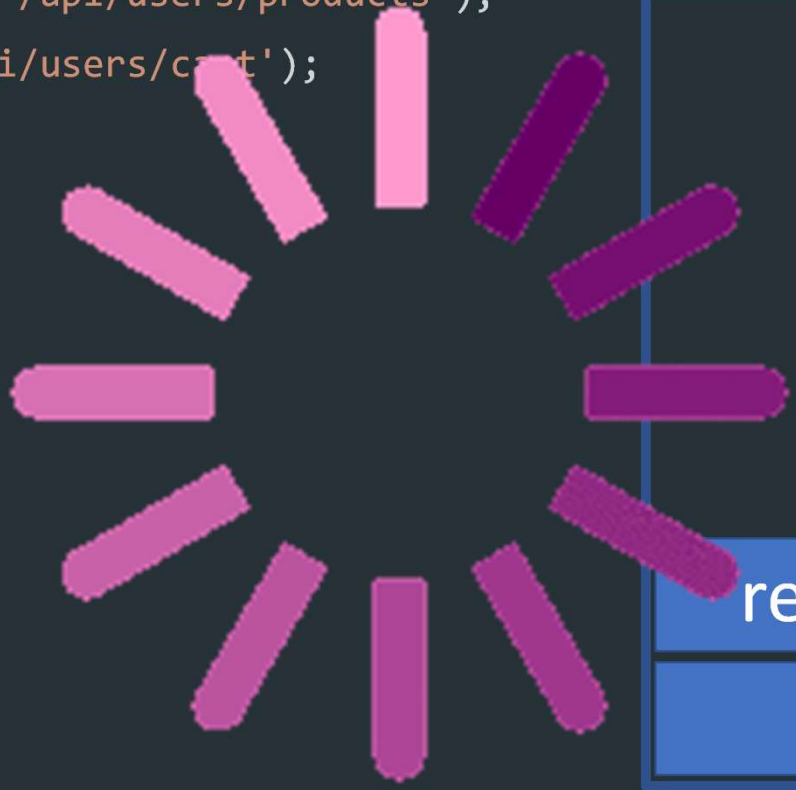


```
var users = http.requestSync('/api/users');
var products = http.requestSync('/api/users/products');
var cart = http.requestSync('/api/users/cart');

console.log(users);
console.log(products);
console.log(cart);
```

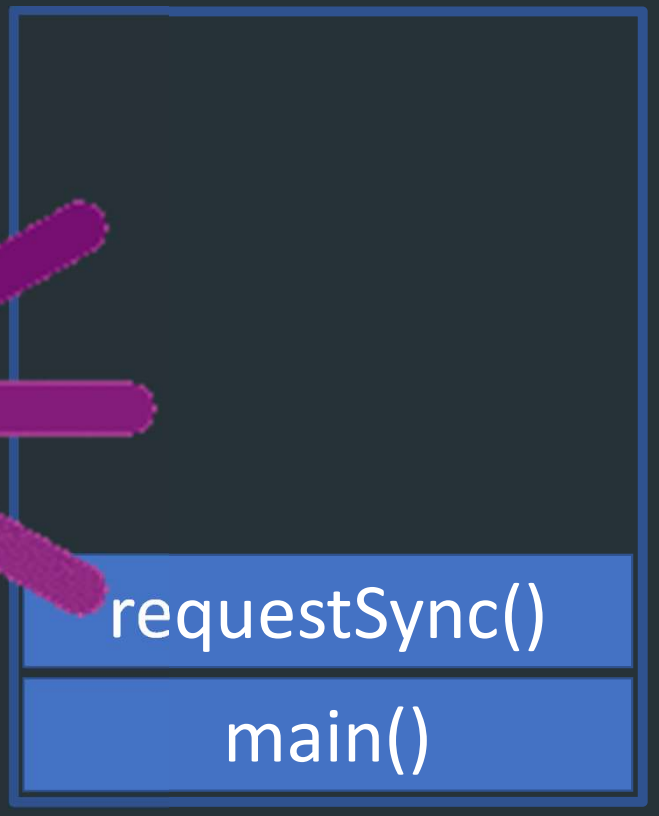
## The Stack

@wizardnet972

```
var users = http.requestSync('/api/users');
var products = http.requestSync('/api/users/products');
var cart = http.requestSync('/api/users/cart');

console.log(users);
console.log(products);
console.log(cart);
```



## The Stack

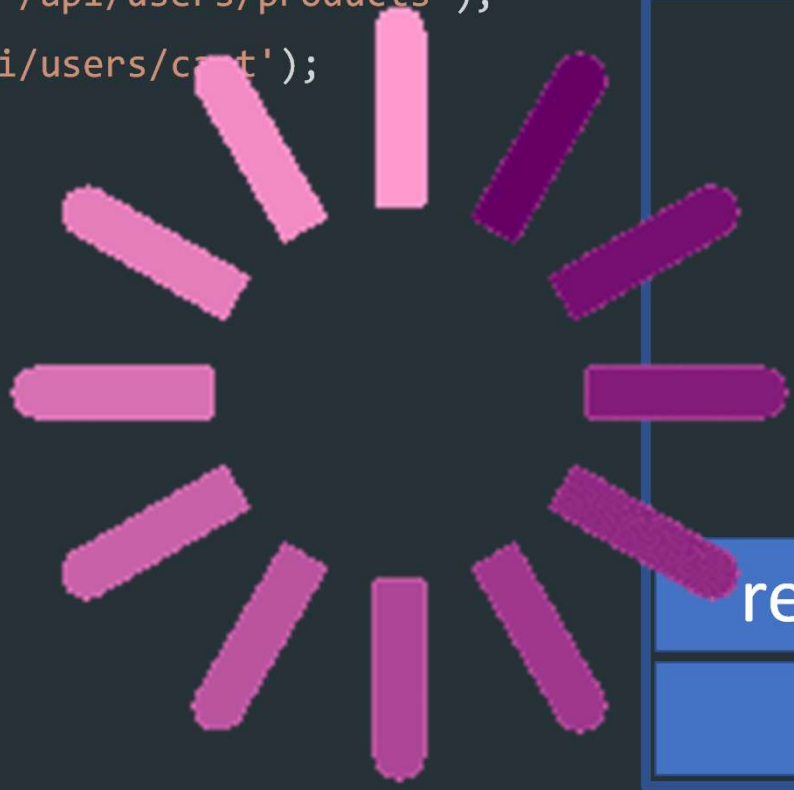


```
var users = http.requestSync('/api/users');
var products = http.requestSync('/api/users/products');
var cart = http.requestSync('/api/users/cart');

console.log(users);
console.log(products);
console.log(cart);
```

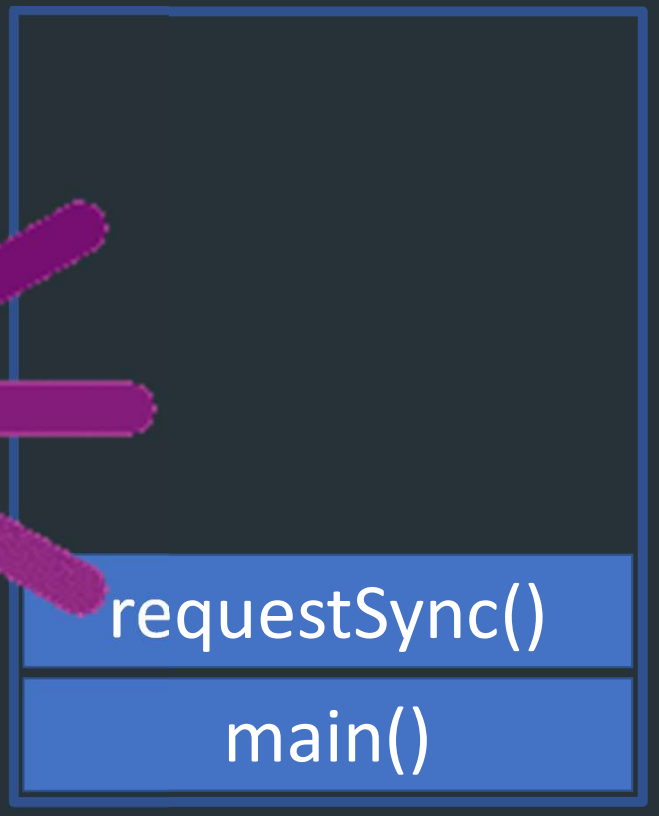
## The Stack

@wizardnet972

```
var users = http.requestSync('/api/users');
var products = http.requestSync('/api/users/products');
var cart = http.requestSync('/api/users/cart');

console.log(users);
console.log(products);
console.log(cart);
```



## The Stack

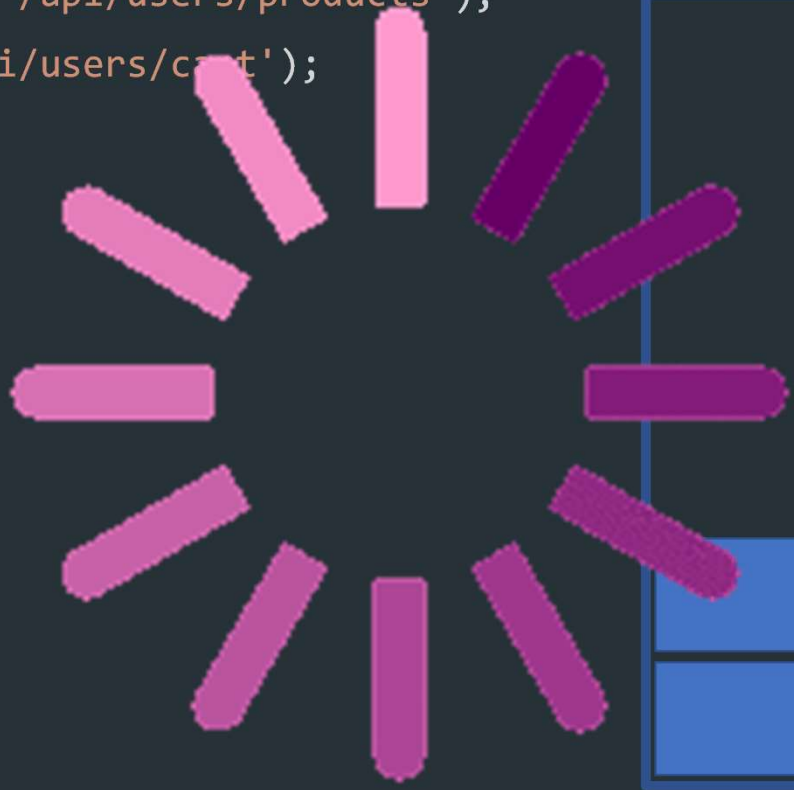


```
var users = http.requestSync('/api/users');
var products = http.requestSync('/api/users/products');
var cart = http.requestSync('/api/users/cart');

console.log(users);
console.log(products);
console.log(cart);
```

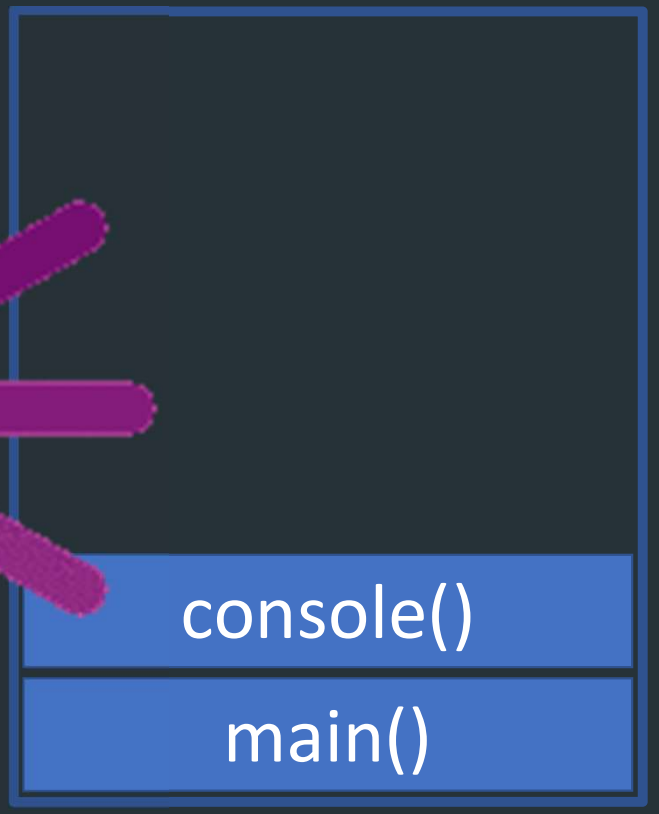
## The Stack

@wizardnet972

```
var users = http.requestSync('/api/users');
var products = http.requestSync('/api/users/products');
var cart = http.requestSync('/api/users/cart');

console.log(users);
console.log(products);
console.log(cart);
```

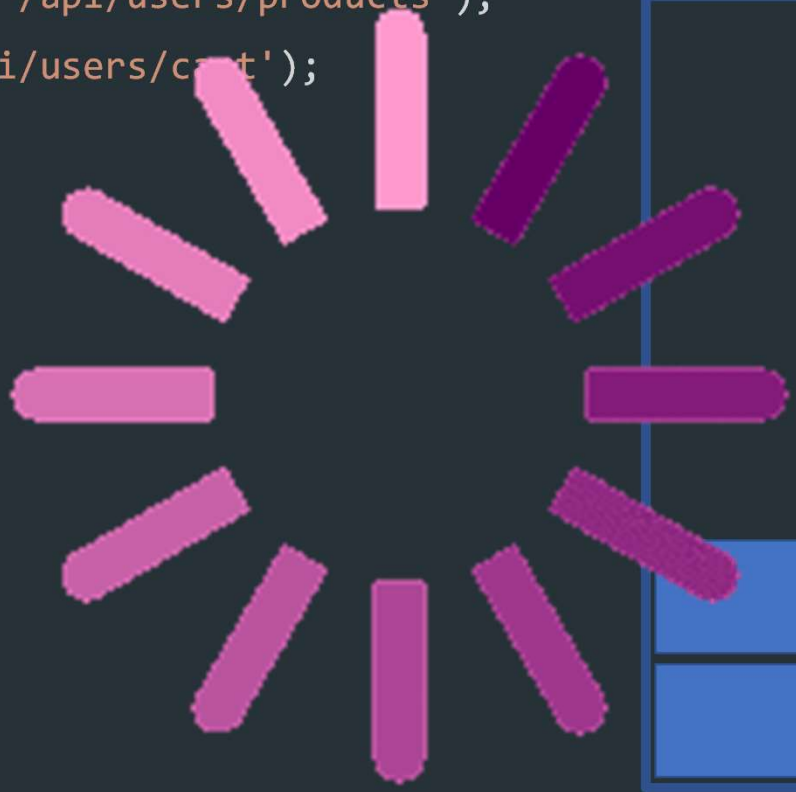


## The Stack

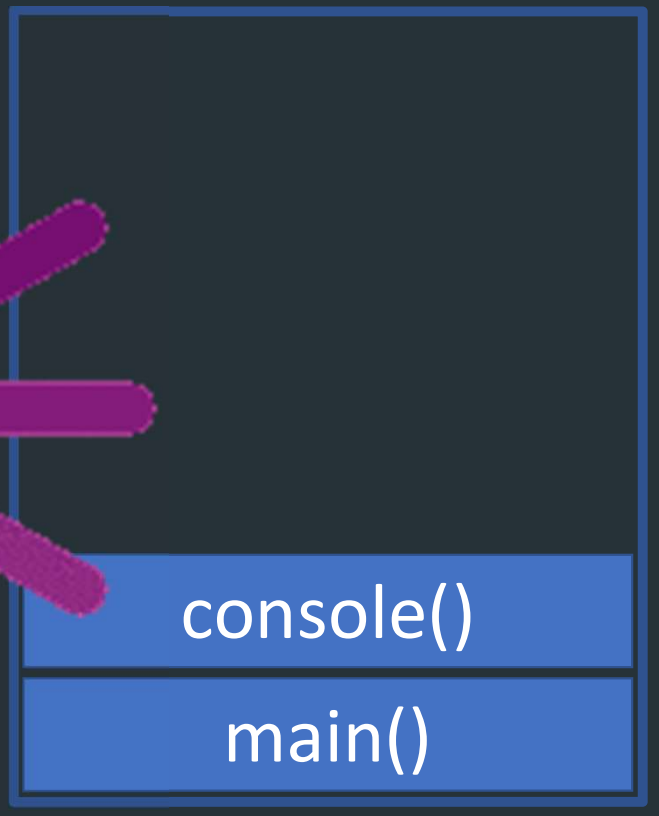


@wizardnet972

```
var users = http.requestSync('/api/users');
var products = http.requestSync('/api/users/products');
var cart = http.requestSync('/api/users/cart');

console.log(users);
console.log(products);
console.log(cart);
```

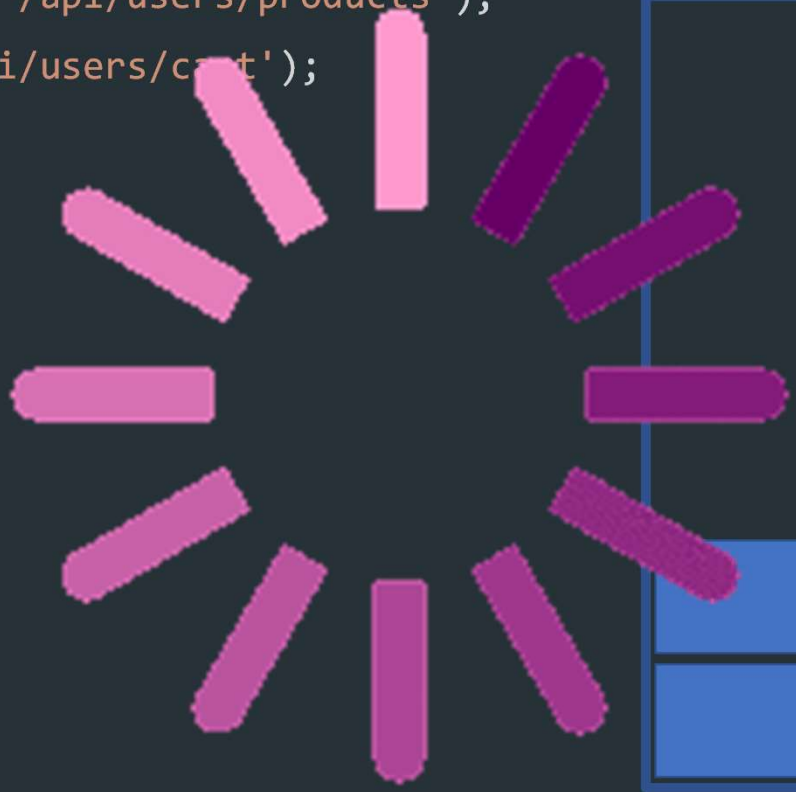


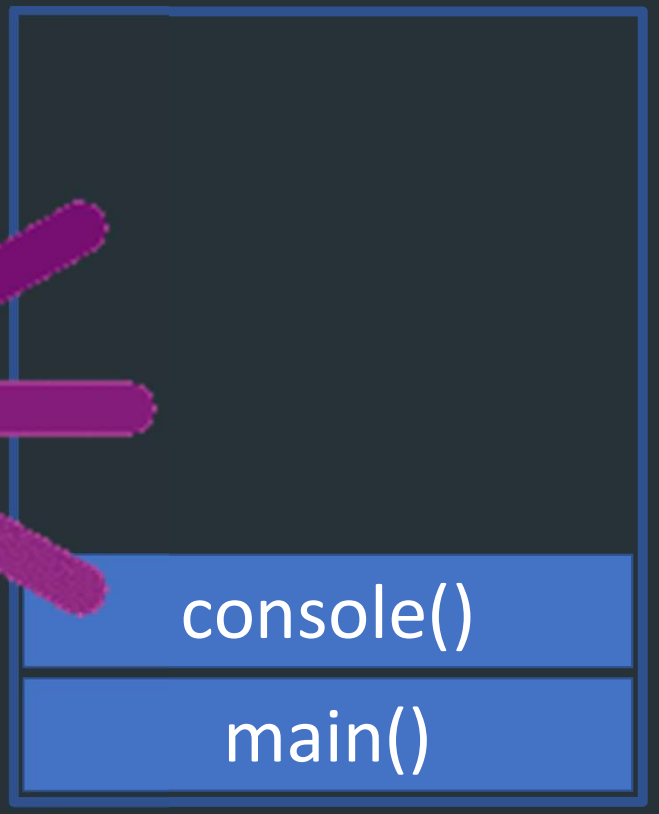
## The Stack



@wizardnet972

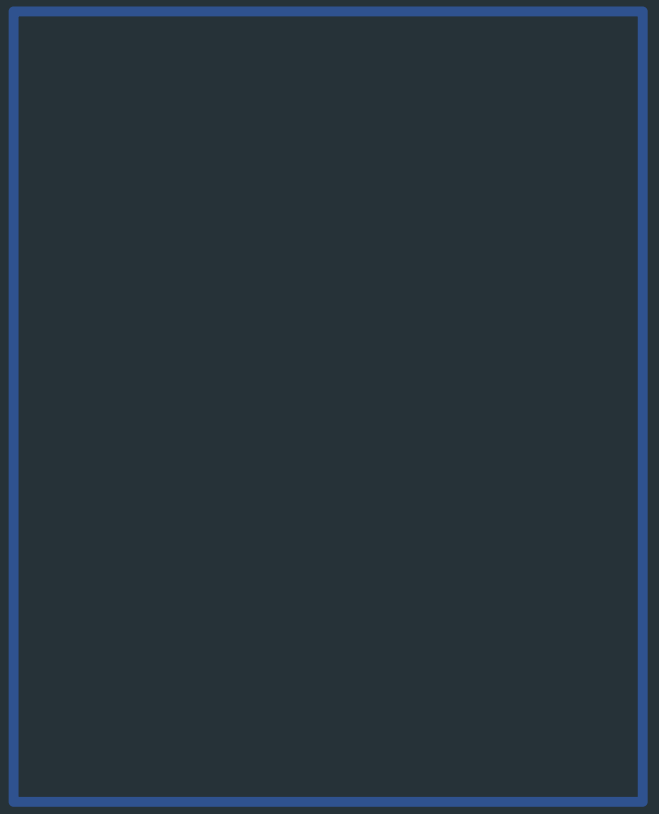
```
var users = http.requestSync('/api/users');
var products = http.requestSync('/api/users/products');
var cart = http.requestSync('/api/users/cart');

console.log(users);
console.log(products);
console.log(cart);
```





```
var users = http.requestSync('/api/users');
var products = http.requestSync('/api/users/products');
var cart = http.requestSync('/api/users/cart');

console.log(users);
console.log(products);
console.log(cart);
```

## The Stack



# Why Sync is a problem?

# Asynchronous callbacks


```
console.log('app started');

setTimeout(function() {
 console.log('after 5 sec');

}, 5000);

console.log('hello world');
```

**Console**

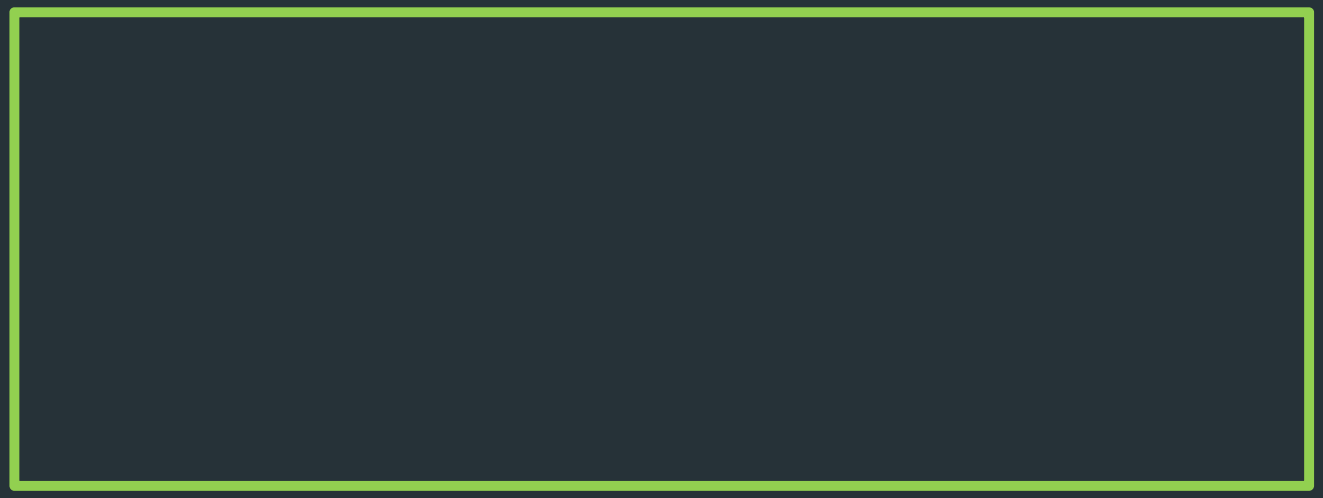
app started

Hello world

After 5 sec

@wizardnet972

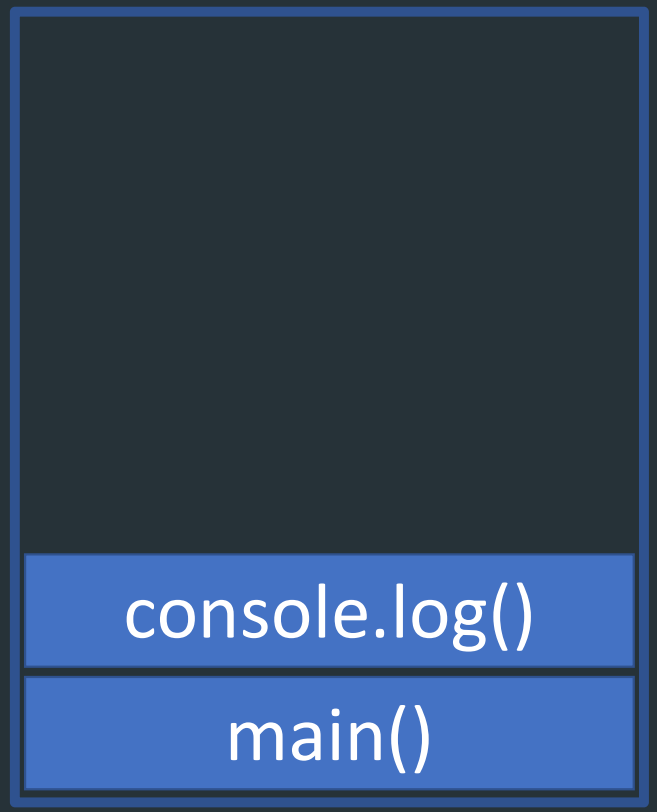
```
console.log('app started');

setTimeout(function() {
 console.log('after 5 sec');

}, 5000);

console.log('hello world');
```



## The Stack

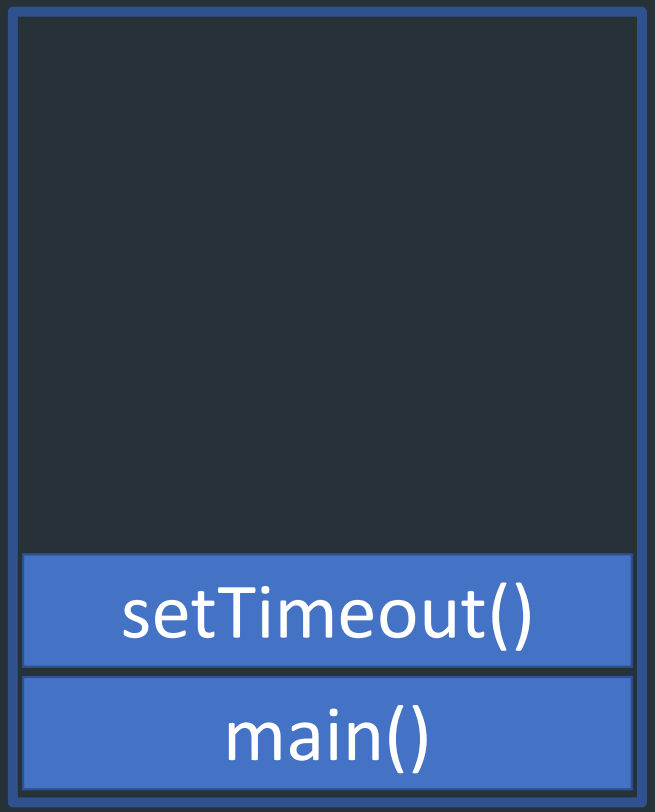
```
console.log('app started');

setTimeout(function() {
 console.log('after 5 sec');

}, 5000);

console.log('hello world');
```

```
λ  app started
```



```
console.log('app started');

setTimeout(function() {
 console.log('after 5 sec');

}, 5000);

console.log('hello world');
```

```
λ  app started
```



@wizardnet972

```
console.log('app started');

setTimeout(function() {
 console.log('after 5 sec');

}, 5000);

console.log('hello world');
```

```
λ  app started

λ  hello world
```

@wizardnet972

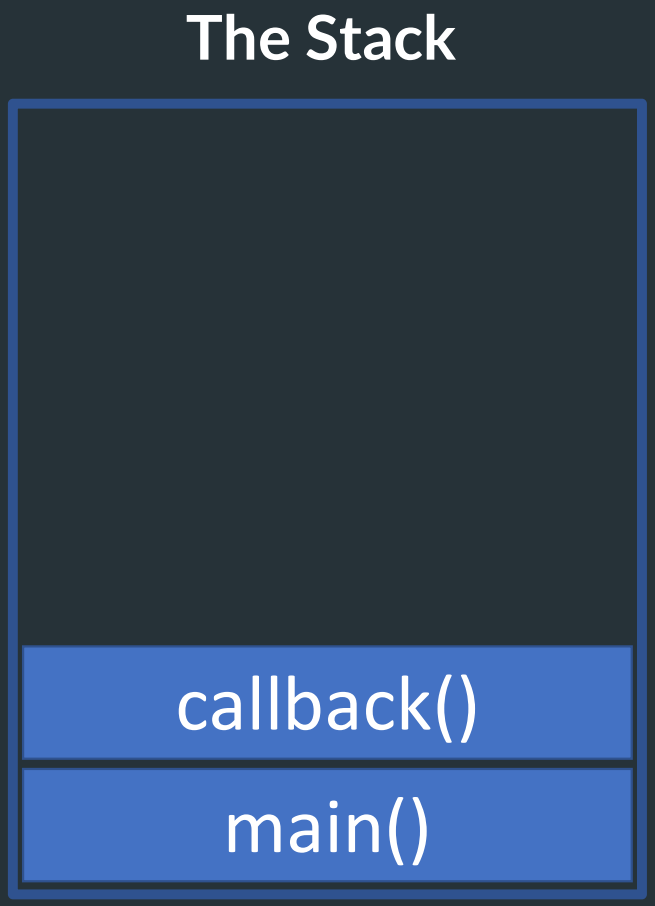
```
console.log('app started');

setTimeout(function() {
 console.log('after 5 sec');

}, 5000);

console.log('hello world');
```

```
λ  app started

λ  hello world
```

@wizardnet972

```
console.log('app started');

setTimeout(function() {
 console.log('after 5 sec');

}, 5000);

console.log('hello world');
```

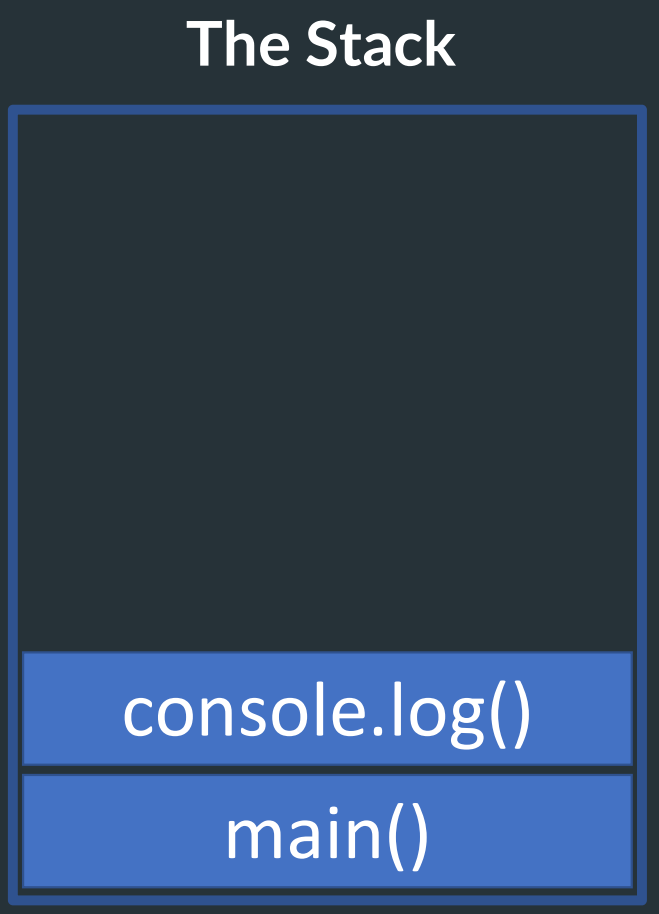
```
λ  app started

λ  hello world
```



@wizardnet972

```
console.log('app started');

setTimeout(function() {
 console.log('after 5 sec');

}, 5000);

console.log('hello world');
```

```
λ  app started

λ  hello world

λ  after 5 sec
```



@wizardnet972

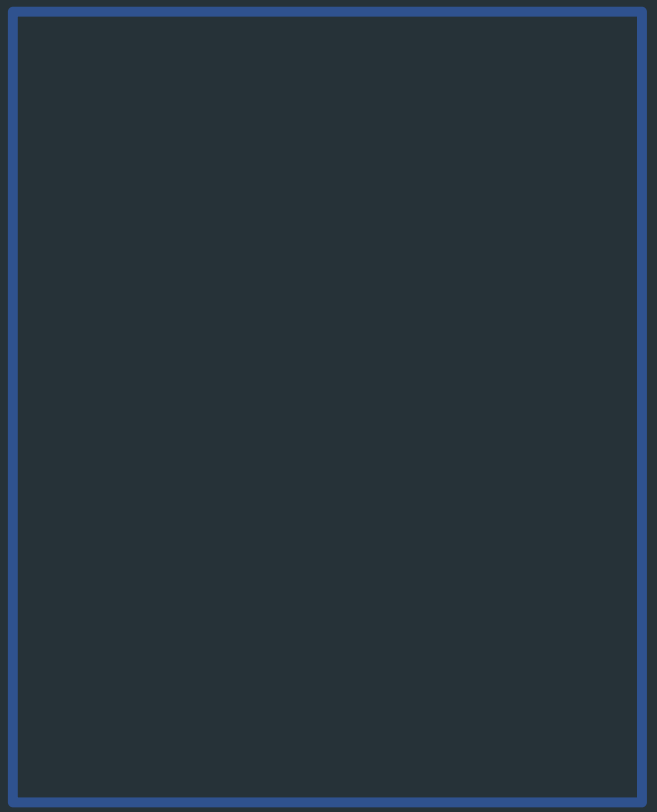
```
console.log('app started');

setTimeout(function() {
 console.log('after 5 sec');

}, 5000);

console.log('hello world');
```

```
λ  app started

λ  hello world

λ  after 5 sec
```

```
console.log('app started');

setTimeout(function() {
 console.log('after 5 sec');

}, 5000);

console.log('hello world');
```

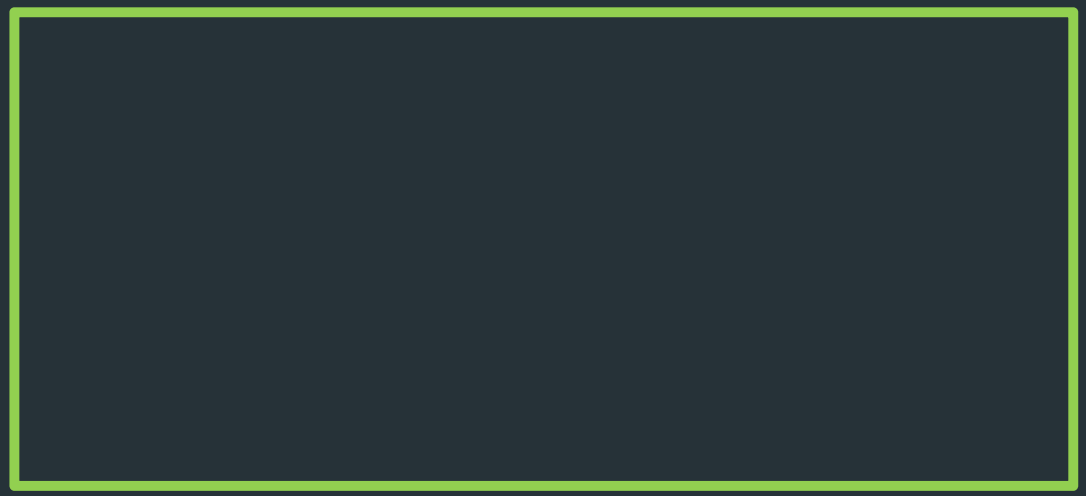
```
λ  app started

λ  hello world

λ  after 5 sec
```

## The Stack



# The event loop!

```
console.log('app started');

setTimeout(function() {
 console.log('after 5 sec');

}, 5000);

console.log('hello world');
```



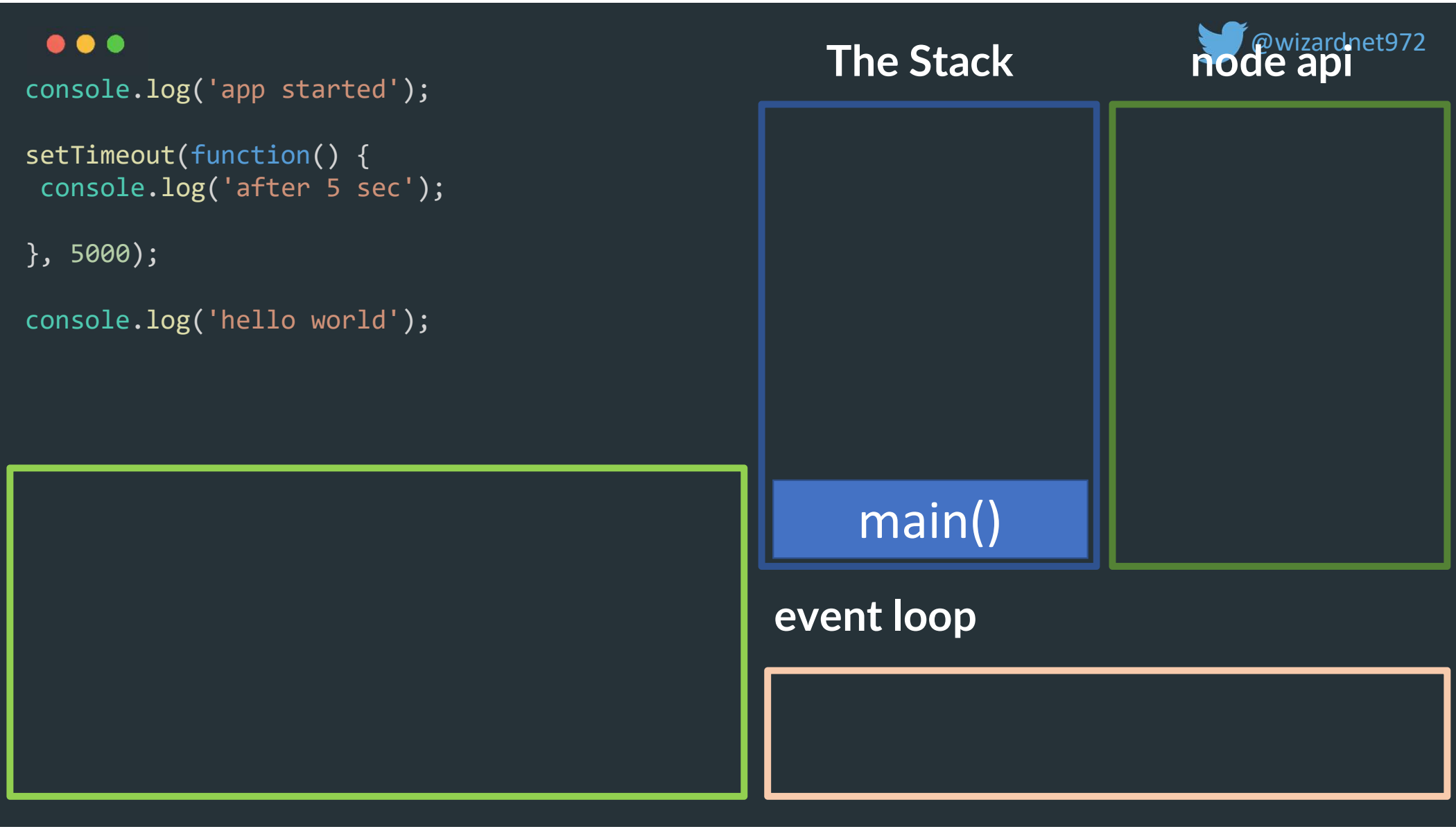
The Stack

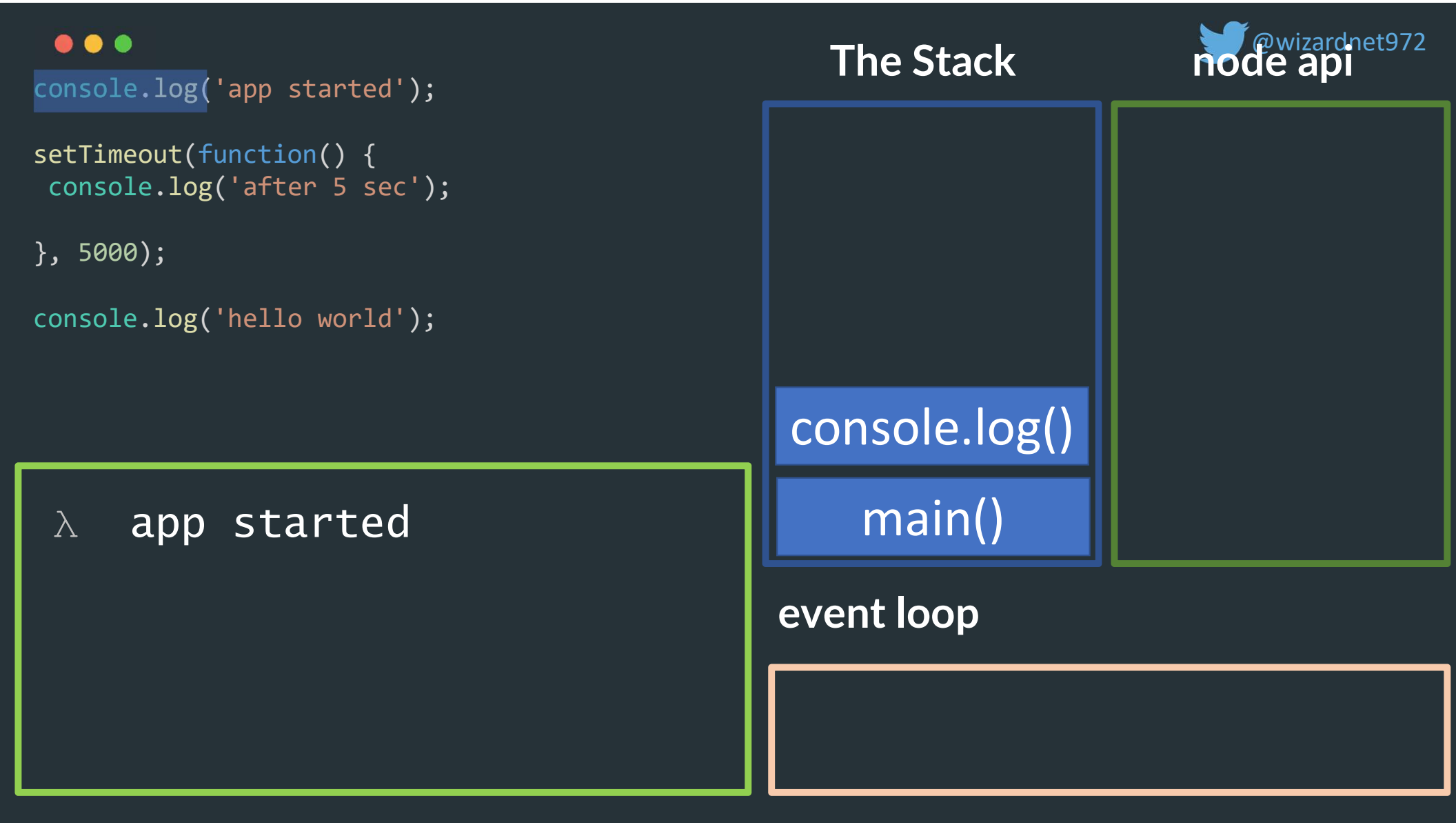
node api

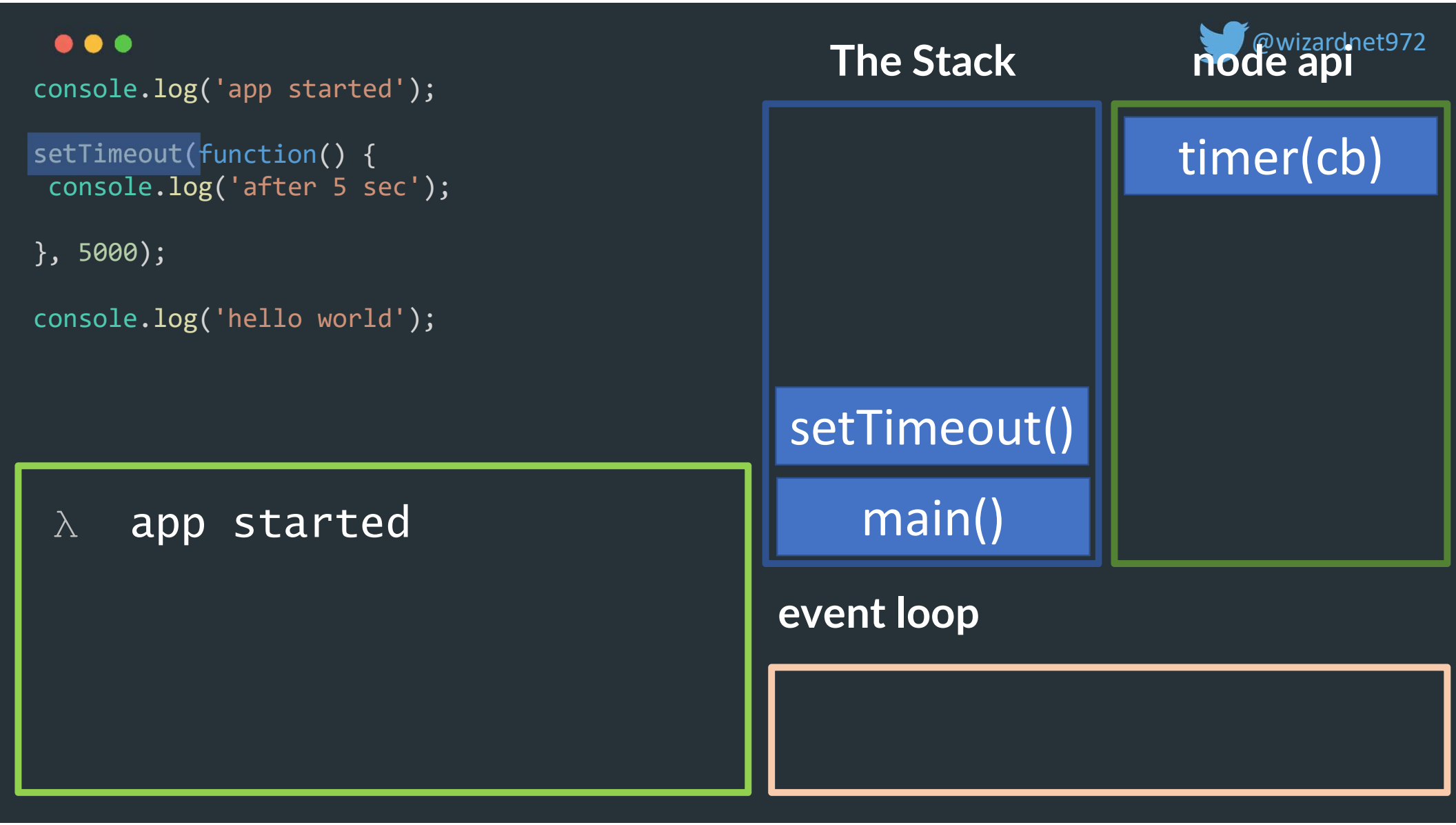
@wizardnet972

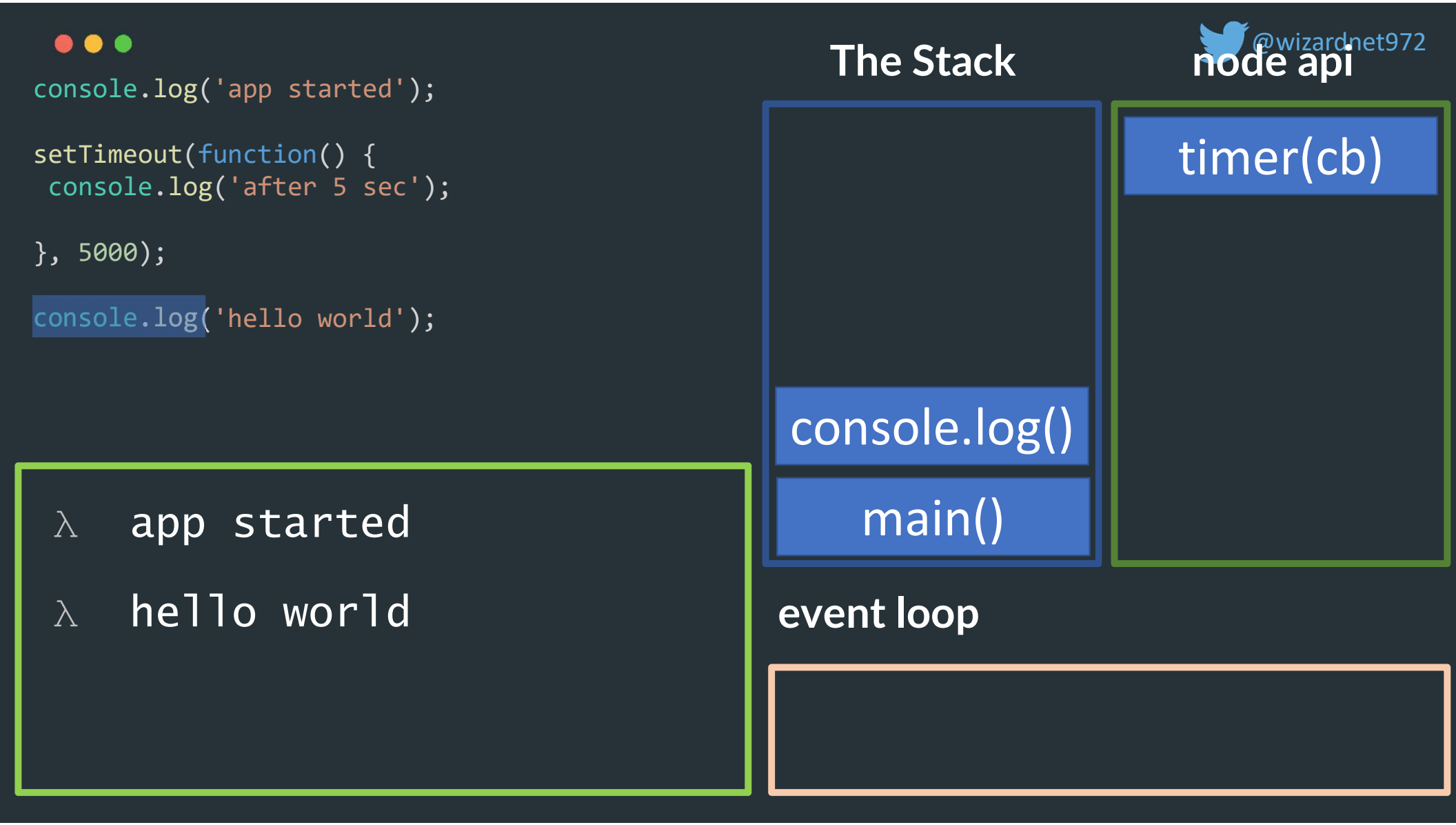
event loop

```
console.log('app started');

setTimeout(function() {
 console.log('after 5 sec');

}, 5000);

console.log('hello world');
```

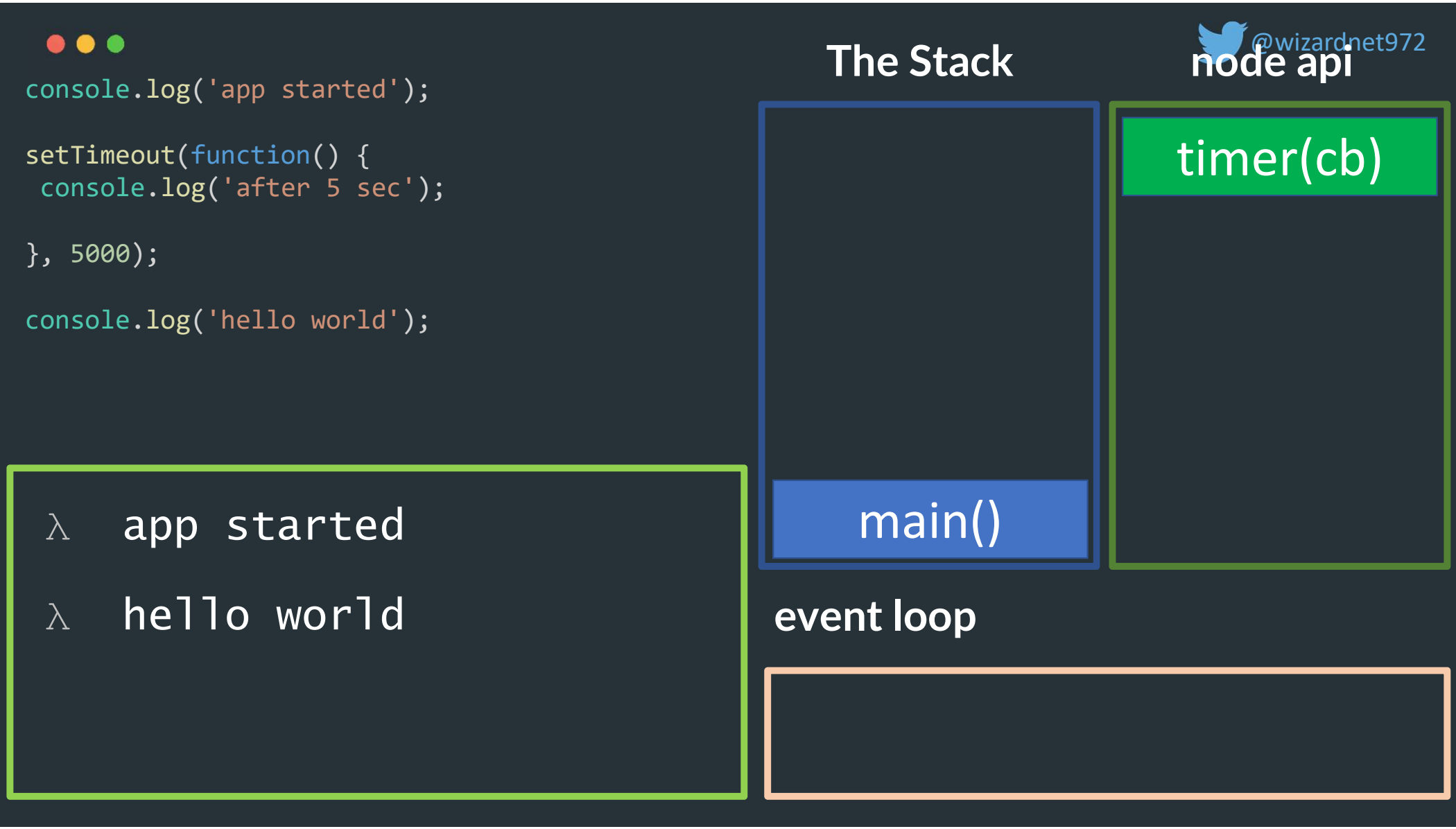



console.log('app started');
setTimeout(function() {
console.log('after 5 sec');
}, 5000);
console.log('hello world');
The Stack
node api
@wizardnet972
console.log()
main()
λ app started
event loop

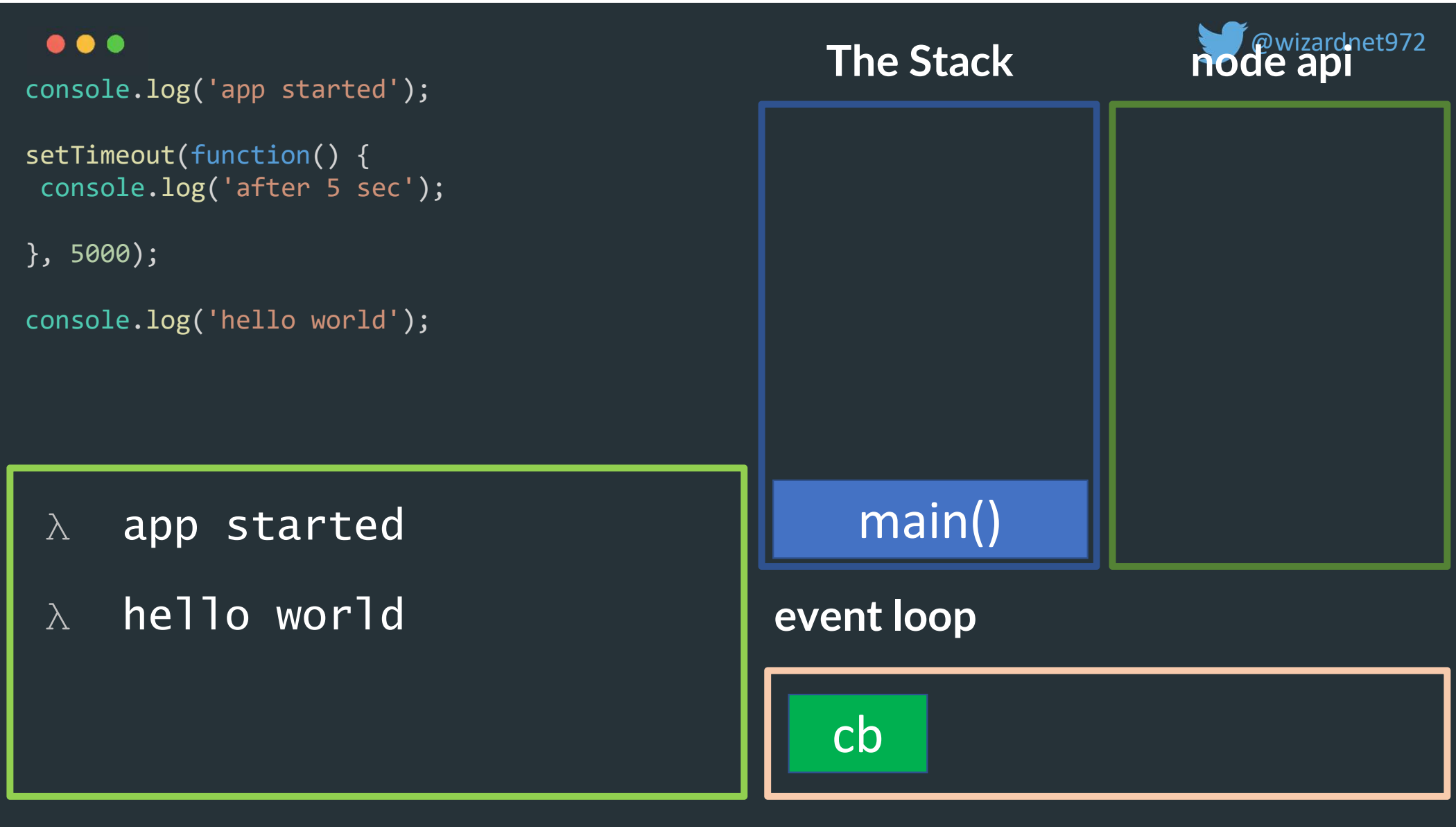

console.log('app started');
setTimeout(function() {
console.log('after 5 sec');
}, 5000);
console.log('hello world');
@wizardnet972
The Stack
node api
timer(cb)
setTimeout()
main()
λ app started
event loop


console.log('app started');
setTimeout(function() {
console.log('after 5 sec');
}, 5000);
console.log('hello world');
The Stack
node api
@wizardnet972
timer(cb)
console.log()
main()
event loop
λ app started
λ hello world

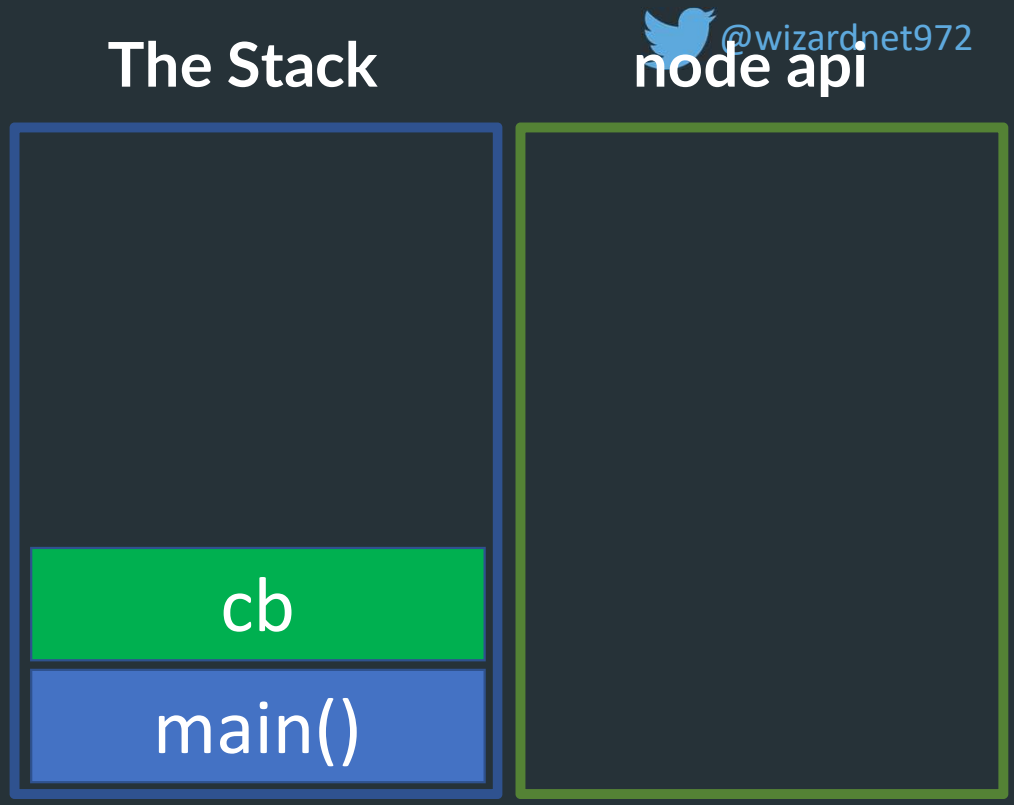
```
console.log('app started');

setTimeout(function() {
 console.log('after 5 sec');

}, 5000);

console.log('hello world');
```



```
console.log('app started');

setTimeout(function() {
 console.log('after 5 sec');

}, 5000);

console.log('hello world');
```

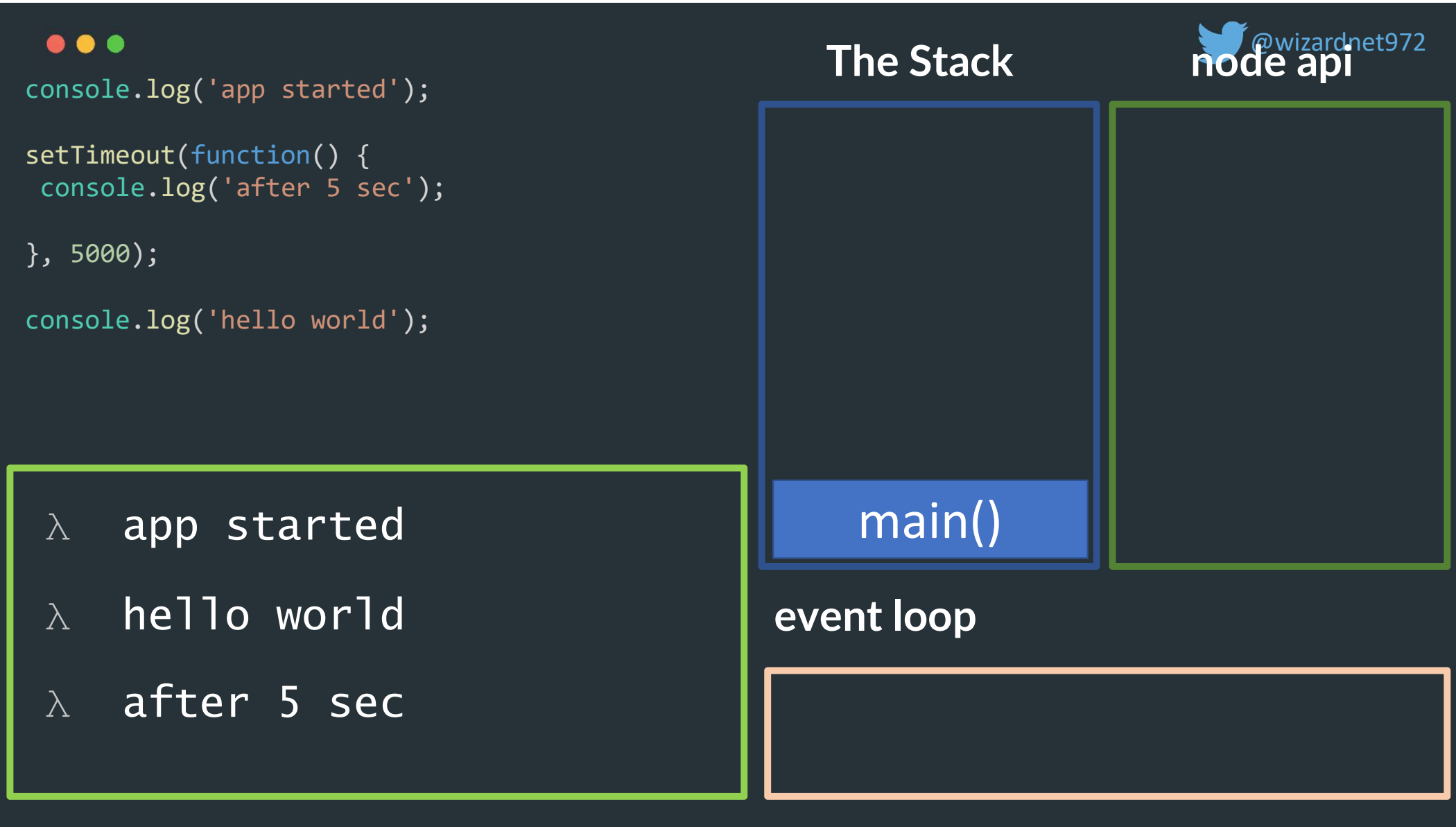


```
console.log('app started');

setTimeout(function() {
 console.log('after 5 sec');

}, 5000);

console.log('hello world');
```



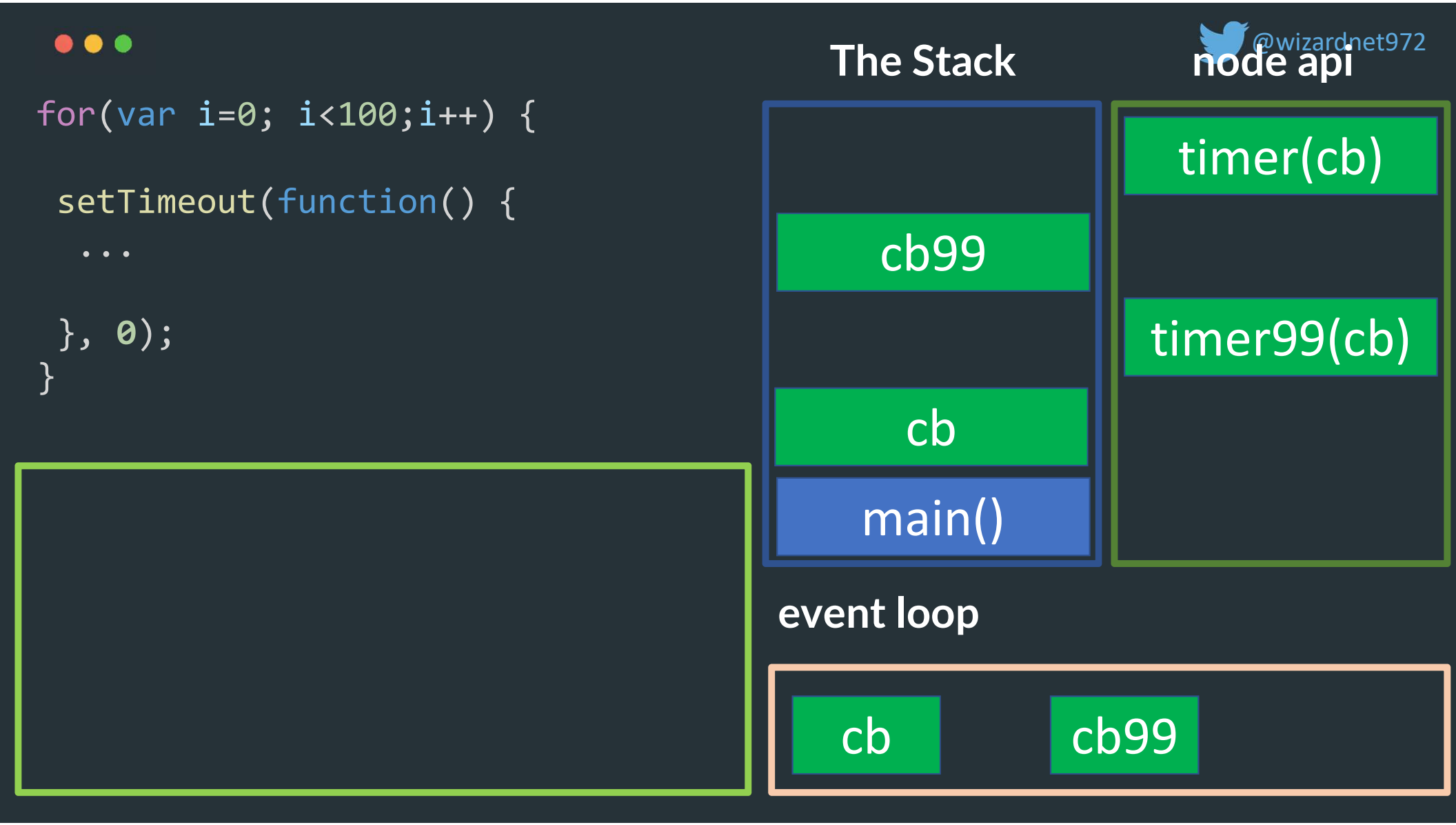

```
λ  app started
λ  hello world
λ  after 5 sec
```

```
console.log('app started');

setTimeout(function() {
 console.log('after 5 sec');

}, 5000);

console.log('hello world');
```



# setTimeout(0)?

# Don't block the event loop!


for(var i=0; i<100;i++) {
setTimeout(function() {
...
}, 0);
}
The Stack
cb99
cb
main()
@wizardnet972
node api
timer(cb)
timer99(cb)
event loop
cb
cb99

# just before event loop started...

# Just before event loop...

- process.nextTick() is not part of the event loop.
- process.nextTick() will be resolved before the event loop continues.
- Guarantee that functions are always runs.
- Maximum call stack size exceeded from v8.

@wizardnet972

# process.nextTick()

```
let bar;

function someAsyncApiCall(callback) {
  callback();
}

someAsyncApiCall(() => {
  console.log('bar', bar); // undefined
});

bar = 1;
```

```
let bar;

function someAsyncApiCall(callback) {
  process.nextTick(callback);
}

someAsyncApiCall(() => {
  console.log('bar', bar); // 1
});

bar = 1;
```

# Realworld example

```
const server = http.createServer(() => {}).listen(8080);

server.on('listening', () => {});
```

# process.nextTick() vs setImmediate()

- process.nextTick() fires immediately on the same phase
- setImmediate() fires on the following iteration or 'tick' of the event loop.

Nodejs recommend developers use setImmediate() in all cases because it's easier to reason about

# Why to use process.nextTick()?

- Handle errors
- Cleanup resources
- Try the request again before the event loop continues.
- Do something just before the event loop continues.

# How long a Tick is?

@wizardnet972

# EventEmitter


```
const EventEmitter = require('events');

class MyEmitter extends EventEmitter {}
const myEmitter = new MyEmitter();

myEmitter.on('event', () => {
 console.log('an event occurred!');
});

myEmitter.emit('event’);
```

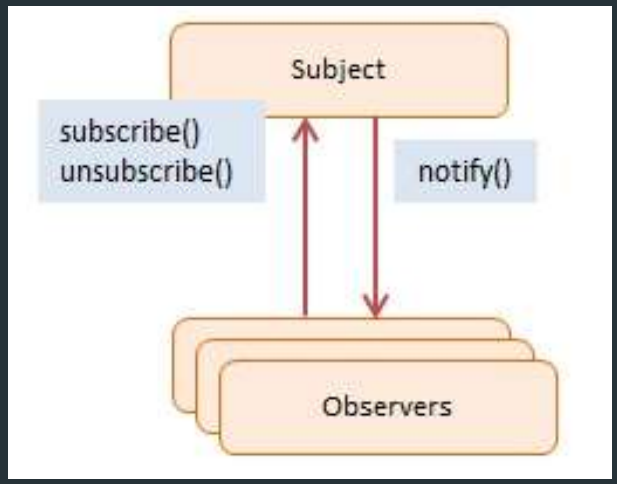
@wizardnet972

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

var stateMachine = new events.EventEmitter();

stateMachine.on('task1', function() {
  console.log('Task 1');
});

stateMachine.on('task2', function() {
  console.log('Task 2');
});

stateMachine.emit('task1');
stateMachine.emit('task2');
```


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

var workflow = new events.EventEmitter();

workflow.on('task1', function() {
  console.log('Task 1');
  workflow.emit('task2');
});

workflow.on('task2', function() {
  console.log('Task 2');
});

workflow.emit('task1');
```

# EventEmitter

- Observer pattern
- Core Module
- function/class
- HTML DOM addEventListener() Method



@wizardnet972

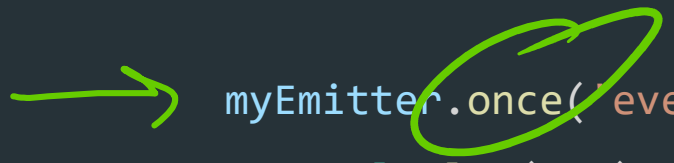
# Handling error events

```
const myEmitter = new MyEmitter();

myEmitter.on('error', (err) => {
 console.error('whoops! there was an error');
});

myEmitter.emit('error', new Error('whoops!'));
// Prints: whoops! there was an error
```

@wizardnet972

# Handling events only once

```
const myEmitter = new MyEmitter();

let x = 0;

myEmitter.on('event', () => {
 console.log(++xx);
});

myEmitter.emit('event');
myEmitter.emit('event');
```



```
const myEmitter = new MyEmitter();

let m = 0;

myEmitter.once('event', () => {
 console.log(++m);
});

myEmitter.emit('event');
myEmitter.emit('event');
```

@wizardnet972

# Timers in Nodejs

Process.tick vs setTimeout vs setImmediate


# After less than a week in Node.js

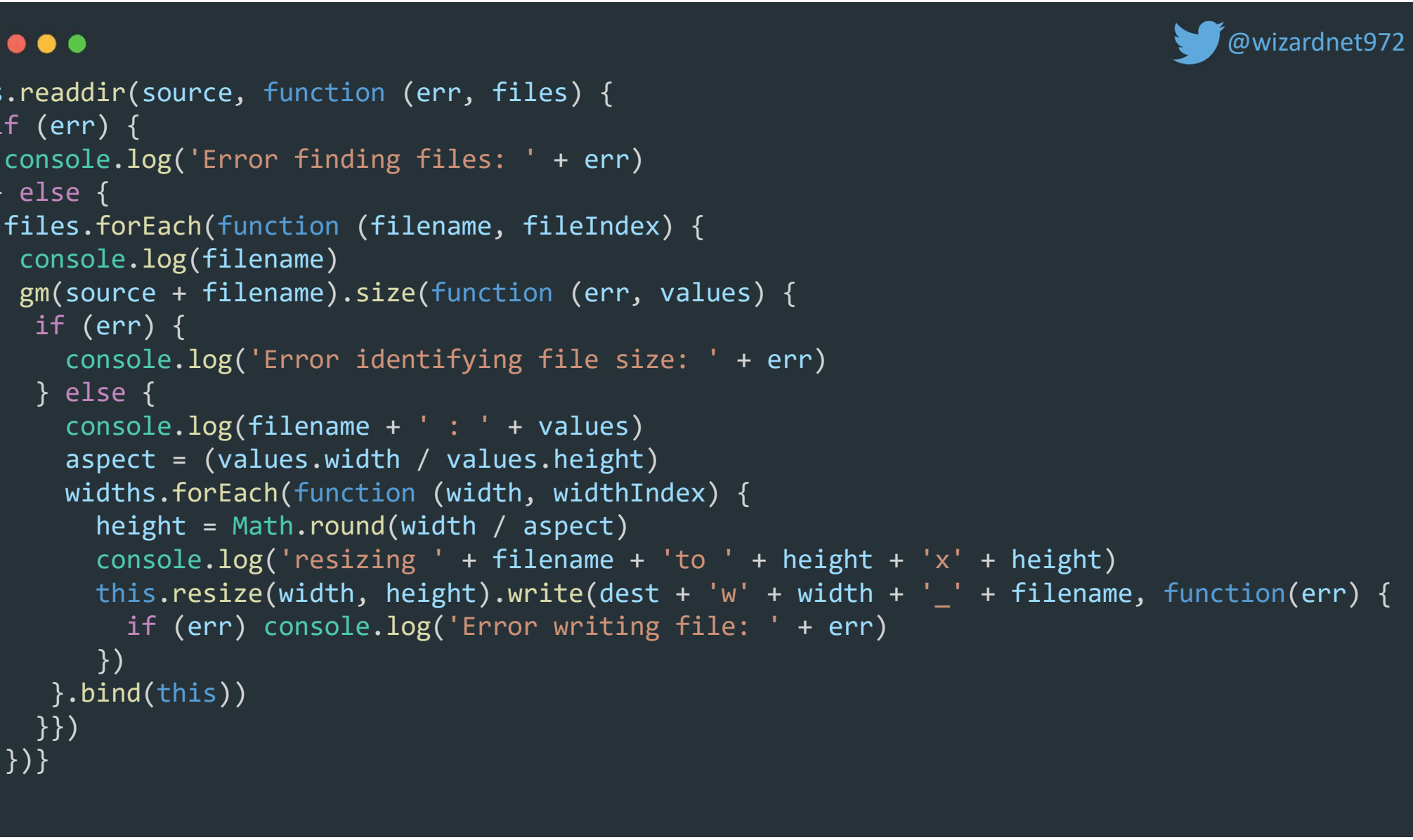
@wizardnet972

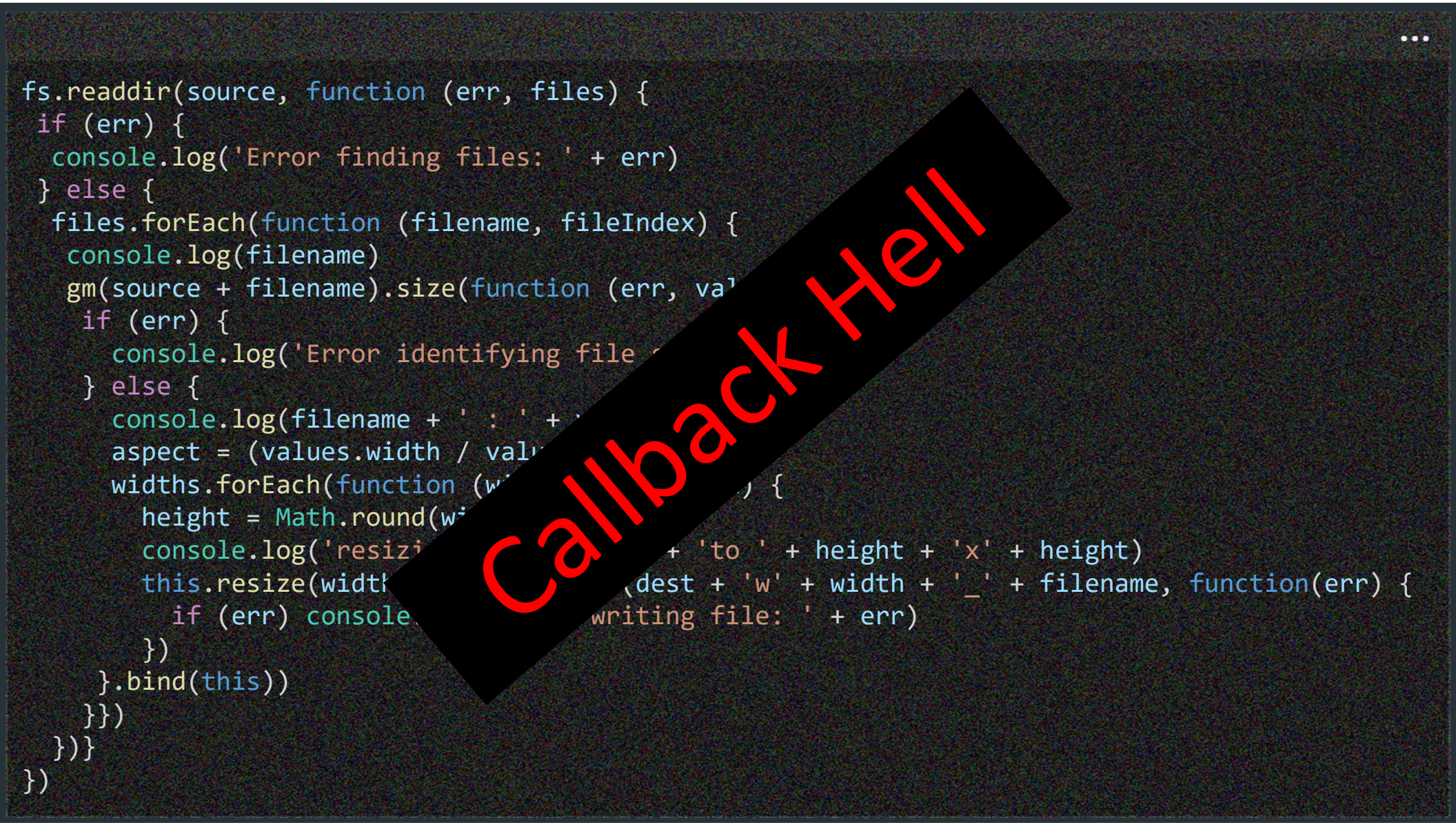
# Synchronous vs Asynchronous

```
function doWorkSync(cb) {
 cb();
}

doWorkSync(function() {
 for(var i=0; i<10; i++) {
   console.log(i);
 }
});

console.log('Good Work!');
```

```
function doWorkAsync(cb) {
  setTimeout(cb, 0);
}

doWorkAsync(function() {
  for(var i=0; i<10; i++) {
    console.log(i);
  }
});

console.log('Good Work! What?');
```

# The first problem when you are write nodejs code

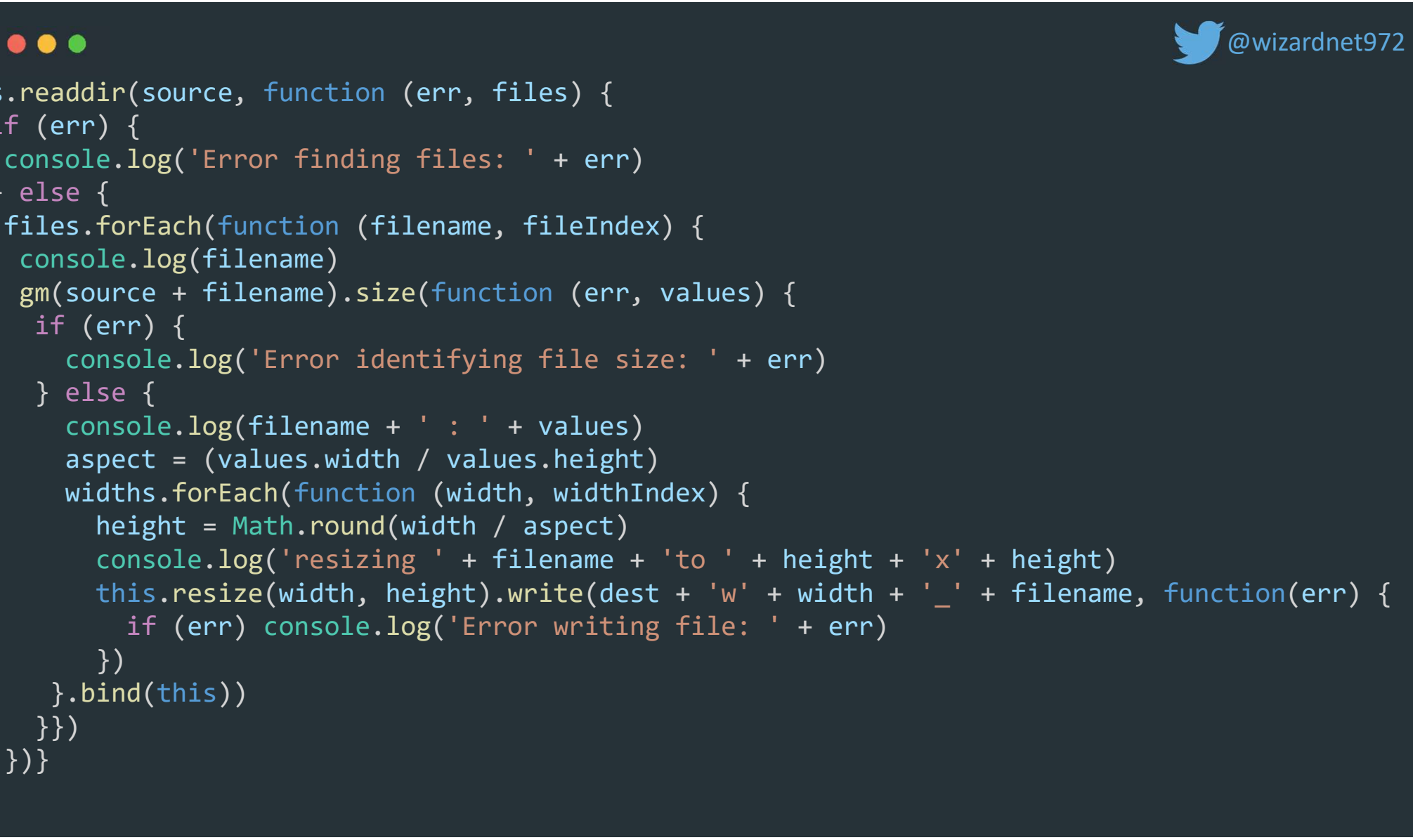

@wizardnet972
fs.readdir(source, function (err, files) {
if (err) {
console.log('Error finding files: ' + err)
} else {
files.forEach(function (filename, fileIndex) {
console.log(filename)
gm(source + filename).size(function (err, values) {
if (err) {
console.log('Error identifying file size: ' + err)
} else {
console.log(filename + ' : ' + values)
aspect = (values.width / values.height)
widths.forEach(function (width, widthIndex) {
height = Math.round(width / aspect)
console.log('resizing ' + filename + 'to ' + height + 'x' + height)
this.resize(width, height).write(dest + 'w' + width + '_' + filename, function(err) {
if (err) console.log('Error writing file: ' + err)
})
}.bind(this))
})})
})}
})

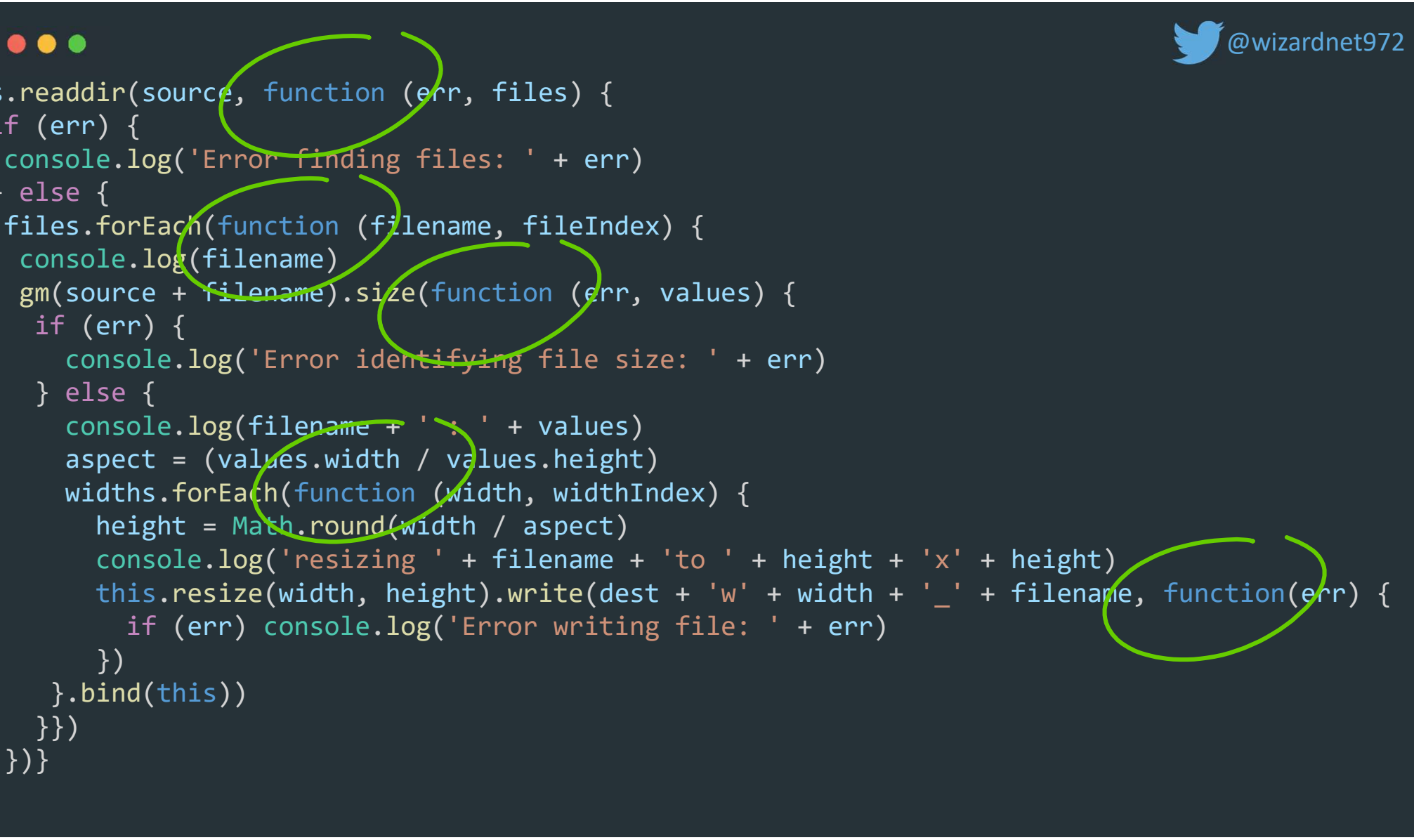

Callback Hell

@wizardnet972

# Callback Hell Solution #1

Keep your code shallow


@wizardnet972
fs.readdir(source, function (err, files) {
if (err) {
console.log('Error finding files: ' + err)
} else {
files.forEach(function (filename, fileIndex) {
console.log(filename)
gm(source + filename).size(function (err, values) {
if (err) {
console.log('Error identifying file size: ' + err)
} else {
console.log(filename + ' : ' + values)
aspect = (values.width / values.height)
widths.forEach(function (width, widthIndex) {
height = Math.round(width / aspect)
console.log('resizing ' + filename + 'to ' + height + 'x' + height)
this.resize(width, height).write(dest + 'w' + width + '_' + filename, function(err) {
if (err) console.log('Error writing file: ' + err)
})
}.bind(this))
})})
})}
})


@wizardnet972
fs.readdir(source, function (err, files) {
if (err) {
console.log('Error finding files: ' + err)
} else {
files.forEach(function (filename, fileIndex) {
console.log(filename)
gm(source + filename).size(function (err, values) {
if (err) {
console.log('Error identifying file size: ' + err)
} else {
console.log(filename + ' : ' + values)
aspect = (values.width / values.height)
widths.forEach(function (width, widthIndex) {
height = Math.round(width / aspect)
console.log('resizing ' + filename + 'to ' + height + 'x' + height)
this.resize(width, height).write(dest + 'w' + width + '_' + filename, function(err) {
if (err) console.log('Error writing file: ' + err)
})
}.bind(this))
})
})}
})

@wizardnet972

```
fs.readdir(source, function handleDir (err, files) {
if (err) {
console.log('Error finding files: ' + err)
} else {
files.forEach(function handleFile (filename, fileIndex) {
console.log(filename)
gm(source + filename).size(function afterSize (err, values) {
if (err) {
console.log('Error identifying file size: ' + err)
} else {
console.log(filename + ' : ' + values)
aspect = (values.width / values.height)
widths.forEach(function setWidth (width, widthIndex) {
height = Math.round(width / aspect)
console.log('resizing ' + filename + 'to ' + height + 'x' + height)
this.resize(width, height).write(dest + 'w' + width + '_' + filename, function afterResize
(err) {
if (err) console.log('Error writing file: ' + err)
})
}.bind(this))}})})}})
```

```
fs.readdir(source, handleDir)

function handleDir (err, files) { ... };

function handleFile (filename, fileIndex) { ... }

function afterSize (err, values) { ... }

function setWidth (width, widthIndex) {
  height = Math.round(width / aspect)
  console.log('resizing ' + filename + 'to ' + height + 'x' + height)
  this.resize(width, height).write(dest + 'w' + width + '_' + filename, afterResize);
}

function afterResize (err) {
  if (err) console.log('Error writing file: ' + err)
}
```

@wizardnet972

# Callback Hell Solution #2

Modularize


```
// filehandler.js
exports = { handleDir, handleFile, afterSize, ... };

// app.js
var fileHandler = require('./filehandler');
fs.readdir(source, fileHandler.handleDir);
```

@wizardnet972

# Callback Hell Solution #3

Handle error first!


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

fs.readFile('/Does/not/exist', handleFile)

function handleFile(error, file) {
 if (error) return console.error('Uhoh, there was an error', error)

 // otherwise, continue on and use `file` in your code
}
```

Handle error first!

No if-else!

@wizardnet972

# Callback Hell
# Other Solutions

Promises Generators Async EventEmitters

# The node way

- Small is beautiful.
- Make each program do one thing well.
- Callback comes last - error first.

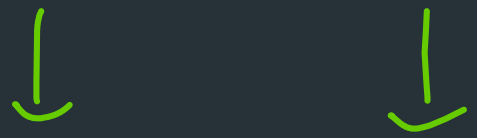
# sometimes in the future...

```
doWork()
.then(anotherOne)
.then(anotherOne)
.then(anotherOne)
.then(anotherOne).....
```

# Promise

- Asynchronous operation
- A promise is an object with a then() function.
- Not another Javascript Framework
- A programming pattern
- Better maintainbility.
- Easier for Scaling.
- Promise has no cancelltion.

# Create a Promise



```
new Promise(function (resolve, reject){ });
```

# Promise States

```
new Promise(function (resolve, reject){ });
```

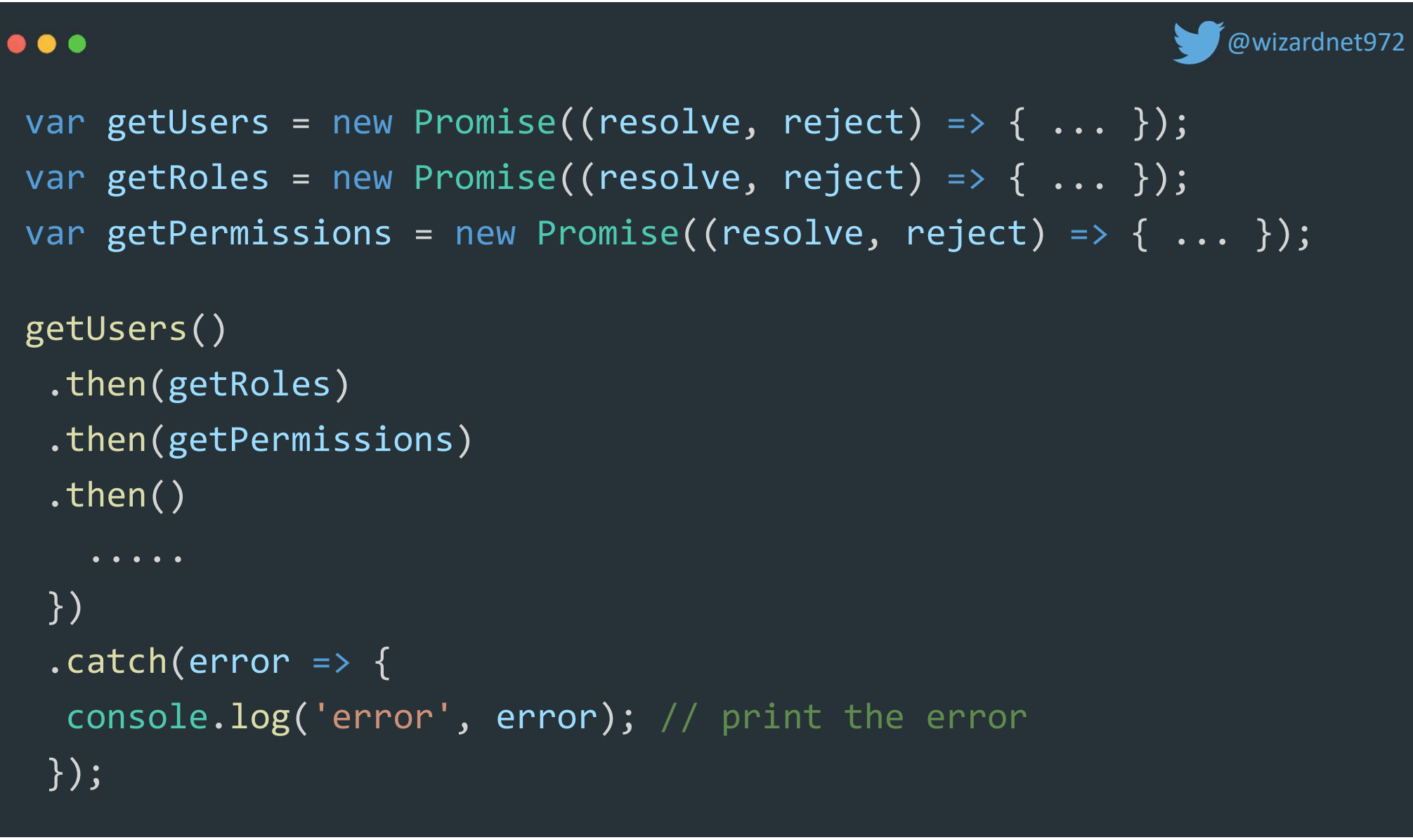

```
function readJSON(filename, callback){

 fs.readFile(filename, 'utf8', function (err, res){
   if (err) return callback(err);

   try {
    res = JSON.parse(res);
   } catch (ex) {
    return callback(ex);
  }

  callback(null, res);
 });

}
```


```
function readJSON(filename, enc){
  return new Promise(function (fulfill, reject){
   fs.readFile(filename, enc, function (err, res){
    if (err) reject(err); else fulfill(res);
   });
 });
}
```


```
var getUserName = new Promise((resolve, reject) => {

  var userName = 'shlomi';

  if (checkSomethingHere()) {
   resolve(userName);
  } else {
   reject(new Error('Something bad happend!'));
  }
});

getUserName().then((userName) => {
 console.log('username', userName); //print shlomi

}).catch(error => {

 console.log('error', error); // print the error
});
```


```
var getUserName = new Promise((resolve, reject) => { ... });
var addLastName = new Promise((resolve, reject) => { ... });
var printName = new Promise((resolve, reject) => { ... });

getUserName()
    .then((userName) => {
      console.log('username', userName); //print shlomi
      return addLastName(userName);
    })
    .then((fullname) => {
      return printName(fullname);
    })
    .then((printName) => {
     // do some work...
    })
    .catch(error => {
      console.log('error', error); // print the error
    });
```


```
var getUsers = new Promise((resolve, reject) => { ... });
var getRoles = new Promise((resolve, reject) => { ... });
var getPermissions = new Promise((resolve, reject) => { ... });

getUsers().then(users => {
  ...
  getRoles(users).then(roles => {
   ...
   getPermissions(users, roles).then(permission => {
    .....
   }).catch(error => {
     console.log('error', error); // print the error
   });
  });
})
```


@wizardnet972
var getUsers = new Promise((resolve, reject) => { ... });
var getRoles = new Promise((resolve, reject) => { ... });
var getPermissions = new Promise((resolve, reject) => { ... });
getUsers()
.then(getRoles)
.then(getPermissions)
.then()
.....
})
.catch(error => {
console.log('error', error); // print the error
});

# Resolve & Reject

```
var p = new Promise((resolve, reject) => {
  resolve();
});
```

```
Promise.resolve();
```

```
var p = new Promise((resolve, reject) => {
  reject();
});
```

```
Promise.reject();
```


```
const promises = [
 new Promise(resolve => setTimeout(resolve, 0, 1)),
 new Promise(resolve => setTimeout(resolve, 0, 2))
];
Promise.all(promises)
.then(data => {
  console.log("First handler", data);
  return data.map(entry => entry * 10);
})
.then(data => {
  console.log("Second handler", data);
});
```

async

```
function getUserDataAsync(userId) {
 return new Promise((resolve, reject) => {
  getUserData(userId, (err, data) => {
   if (err) {
    return reject(err);
   }
   resolve(data);
  });
 });
}

getUserDataAsync(378)
.then(data => {
  console.log('data is ', data);
})
.catch(error => {
  console.log('error!', error);
});
```

```
function getUserData(userId, cb) {
 cb(filename);
}


getUserData(378, (err, data) => {
 if (err) {
  return console.log('error!', err);
 }
 console.log('data is ', data);
});
```

sync

# So, Should I do that for every function?

@wizardnet972

# Bluebird!

a promise library

@wizardnet972

# Bluebird!

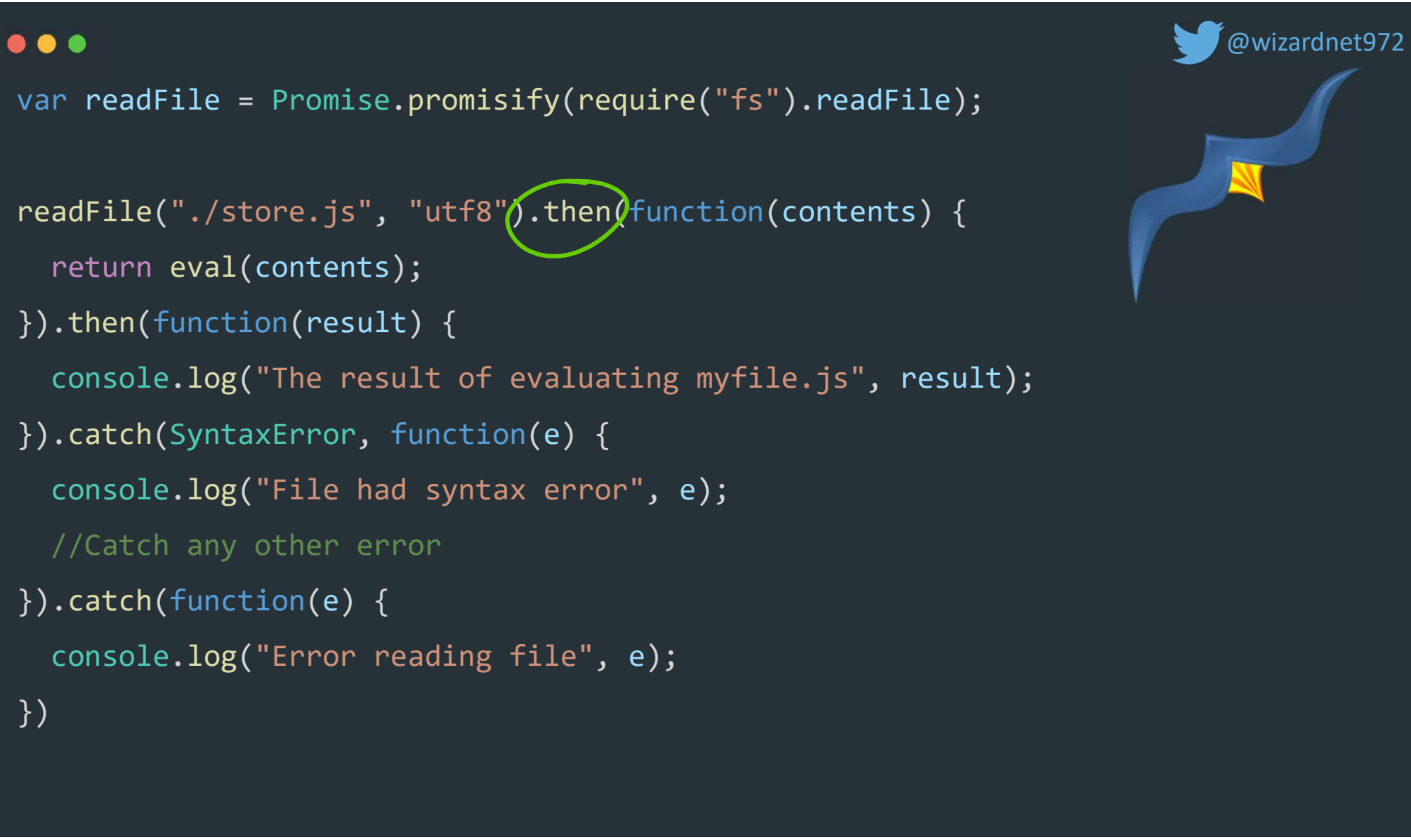
- Promise with enhancement!
- http://bluebirdjs.com

@wizardnet972

# Install Bluebird

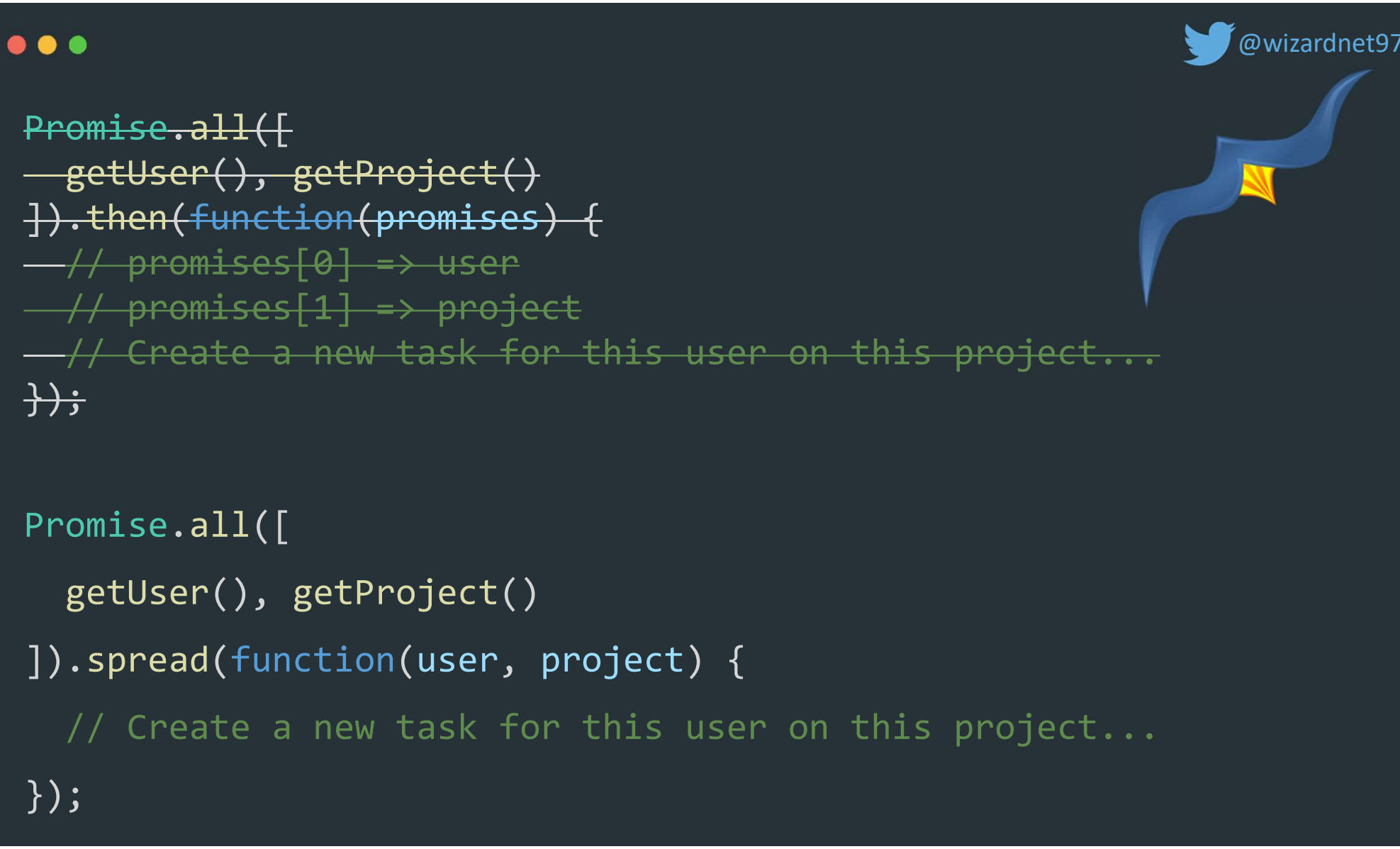
```
λ  npm i bluebird --save-dev
```

```
var Promise = require("bluebird");
```


@wizardnet972
var readFile = Promise.promisify(require("fs").readFile);
readFile("./store.js", "utf8").then(function(contents) {
return eval(contents);
}).then(function(result) {
console.log("The result of evaluating myfile.js", result);
}).catch(SyntaxError, function(e) {
console.log("File had syntax error", e);
//Catch any other error
}).catch(function(e) {
console.log("Error reading file", e);
})

@wizardnet972

```
var user = getUser();
var project = getProject();

Promise.all([
  getUser(),
  getProject()
]).then(function(promises) {
  // promises[0] => user
  // promises[1] => project
  // Create a new task for this user on this project...
});
```


@wizardnet972
Promise.all([
getUser(), getProject()
]).then(function(promises) {
// promises[0] => user
// promises[1] => project
// Create a new task for this user on this project...
});
Promise.all([
getUser(), getProject()
]).spread(function(user, project) {
// Create a new task for this user on this project...
});


```
Promise.resolve()
.then(() => {    throw new Error('MY ERROR’); })
.then(() => { console.log('IN THEN AFTER ERROR’); })
.finally(() => {  console.log('IN FINALLY’) })
.then(() => {  console.log('IN THEN AFTER FINALLY’); })
.catch(() => {  console.log('IN CATCH AFTER FINALLY’); });


// IN FINALLY
// IN CATCH AFTER FINALLY
```


@wizardnet972
Promise.config({cancellation: true});
var fs = Promise.promisifyAll(require("fs"));
var p = Promise.resolve('./config.json')
.timeout(2000)
.catch(console.error.bind(console, 'Failed to load config!'))
.then(fs.readFileAsync)
.then(JSON.parse);
process.on('unhandledException', function(event) {
p.cancel();
});

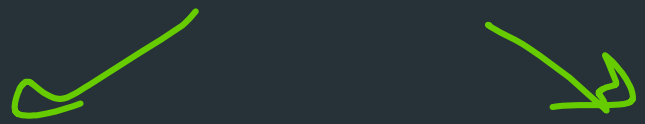
# File System

# File System – Why?

- Access data
- Logs
- File uploads (rename, copy…)
- Read files
- Create files
- Update files
- Delete files
- Rename files

# File System

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



Synchronous

Asynchronous

# How to read a file?

```
import fs from 'fs';

// async
fs.readFile('./tasks.json', {}, (err, data) => { ... });

// sync
var data = fs.readFileSync('./tasks.json', {});
```

# How to write a file?

```
import fs from 'fs';

const text = 'blabla....';

// async
fs.writeFile('./tasks.json', text, (err) => { ... });

// sync
fs.writeFileSync('./tasks.json', text);
```


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

http.createServer((req, res) => {

 fs.readFile('demofile1.html', (err, data) => {

  res.writeHead(200, { 'Content-Type': 'text/html' });
  res.write(data);
  res.end();

 });
})
```

# Delete a file

```
import fs from 'fs';

// async
fs.unlink('mynewfile2.txt', function(err) {

 if (err) throw err;

 console.log('File deleted!');
});

// sync
fs.unlinkSync('mynewfile2.txt');
```

# Rename a file

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

// async
fs.rename('mynewfile1.txt', 'myrenamedfile.txt', function(err) {
 if (err) throw err;
 console.log('File Renamed!');
});

// sync
fs.renameSync('mynewfile1.txt', 'myrenamedfile.txt');
```

# How to read a folder?

```
import fs from 'fs';

// async
fs.readdirSync('./folder', {}, (err, files) => { ... });

// sync
var files = fs.readdirSync('./folder', {});
```

# File System

```
// exist
if (fs.existsSync(path)) { ... }

// copy file
fs.createReadStream('test.log').pipe(fs.createWriteStream('newLog.log'));
```


# The M.E.A.N stack

Or FullStack :)

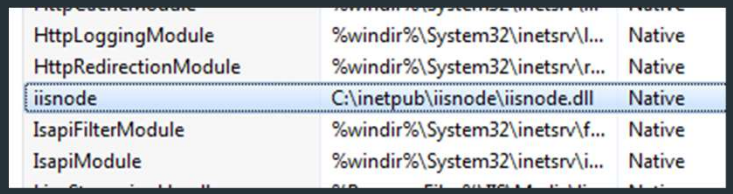
# Production

# Process managers

- Forever
- PM2
- Nginx

# IIS with nodejs?

- Process management.
- Scalability on multi-core servers.




https://github.com/tjanczuk/iisnode

# Production: do and donts

- Use gzip compression
- Don’t use synchronous functions
- Do logging correctly
- Handle exceptions properly
- Set NODE_ENV to “production”
- Ensure your app automatically restarts
- Run your app in a cluster
- Use a load balancer
- Use a reverse proxy
- Cache request results

# Client-server-database architecture