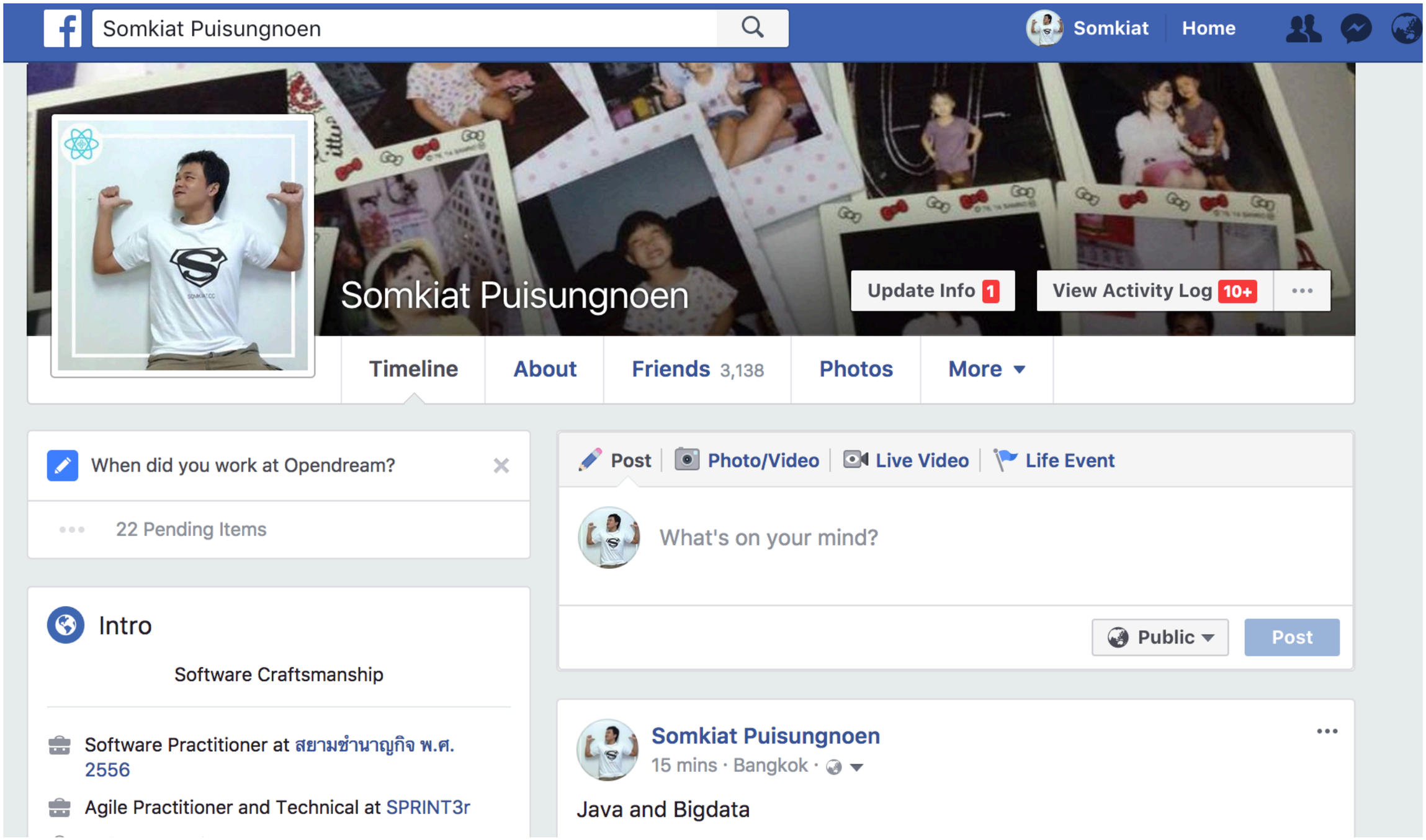
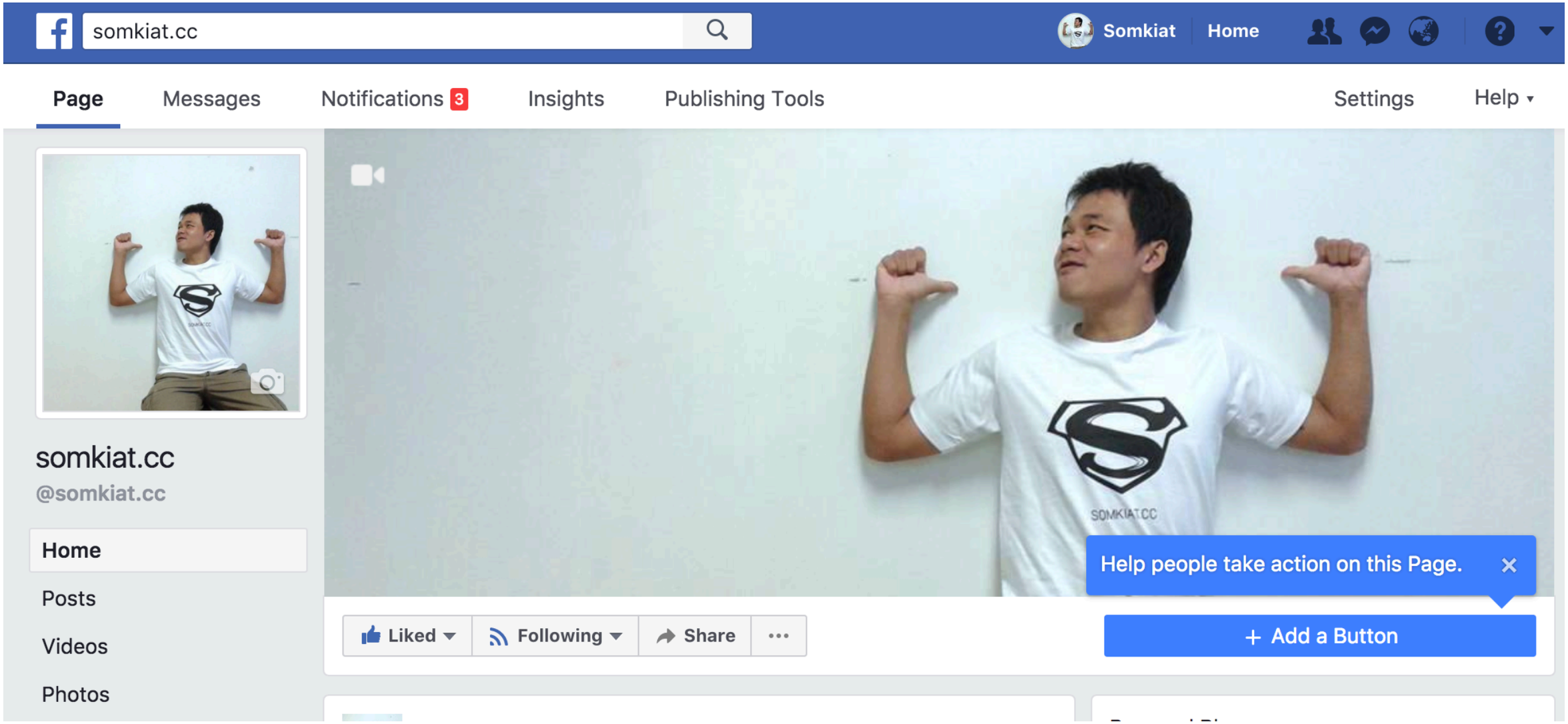
สยามชำนาญกิจ
SIAM CHAMNAN KIT

NodeJS





Agenda

Introduction to Node.js

Learn to write code with Node.js

Command-line application

Working with ECMAScript(ES)

Develop REST APIs

Testable and Testing

Working with database (RDBMS and NoSQL)

Deployment

https://github.com/up1/course-nodejs-2020

NodeJS

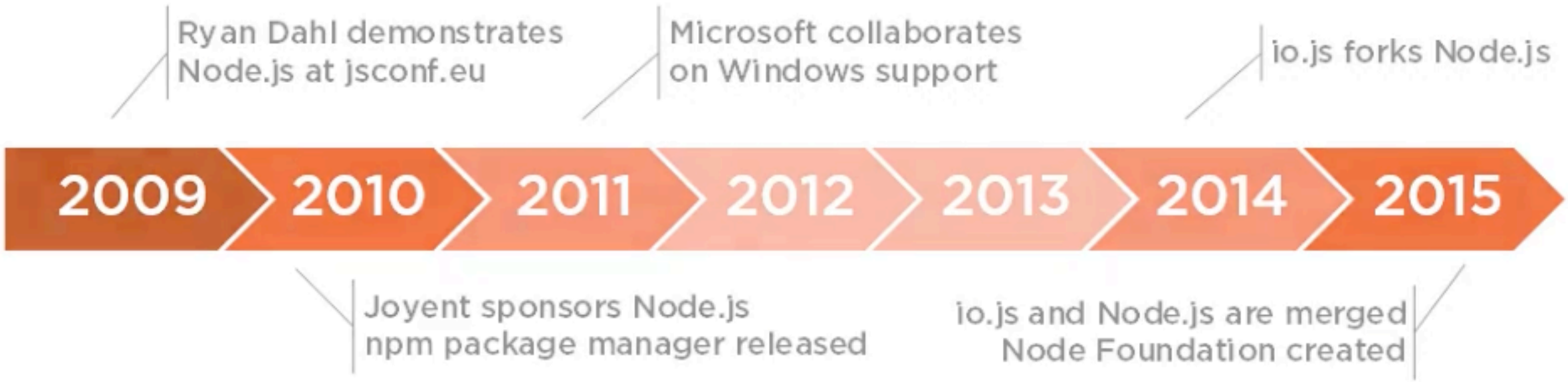
What Node.js ?

JavaScript on the server
Open source, Free
Run on many platforms

What Node.js ?

Asynchronous event-driven JavaScript runtime
Designed to build network applications

History of Node.js



What Node.js ?

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world.

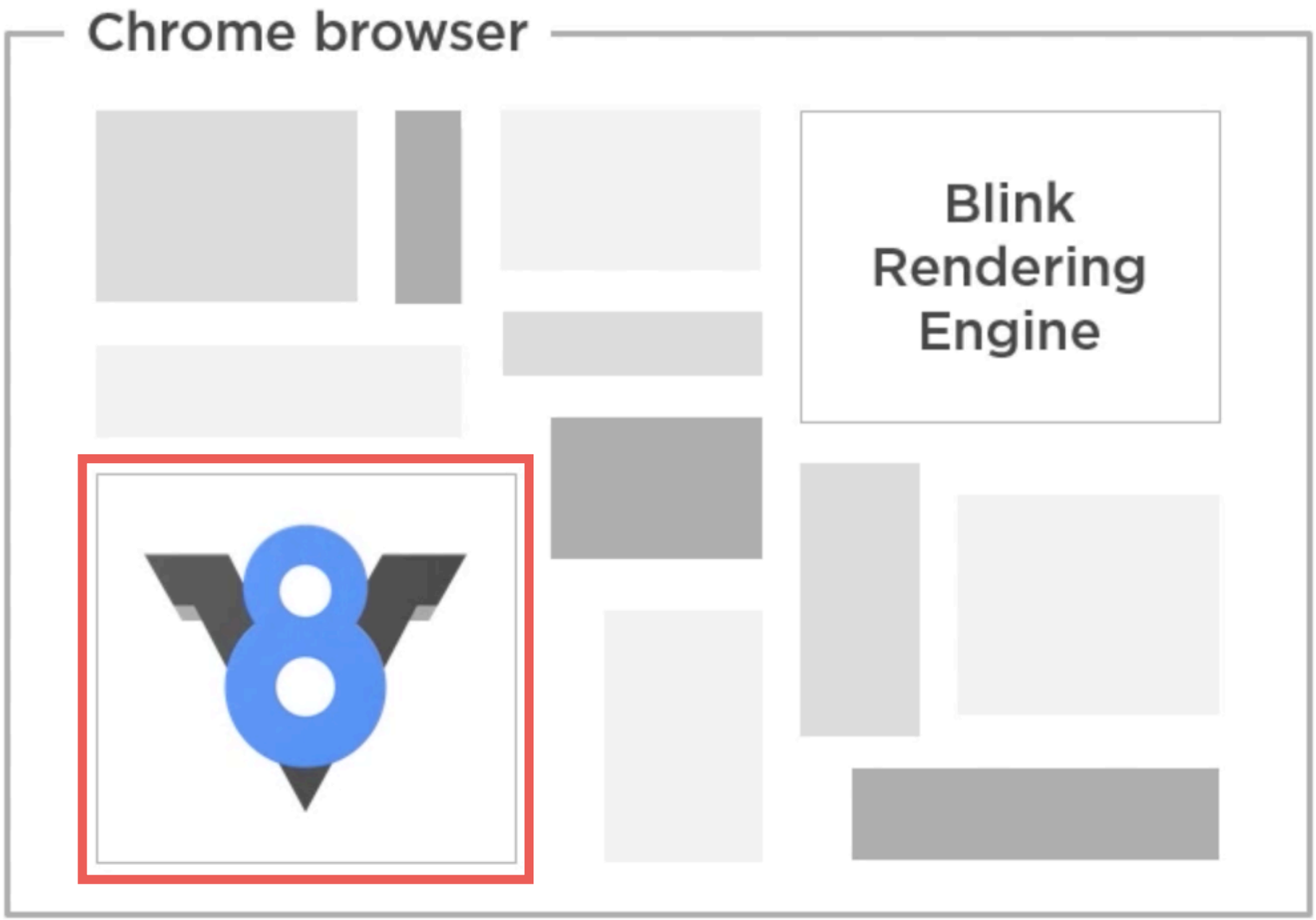
https://nodejs.org/en/

What Node.js ?

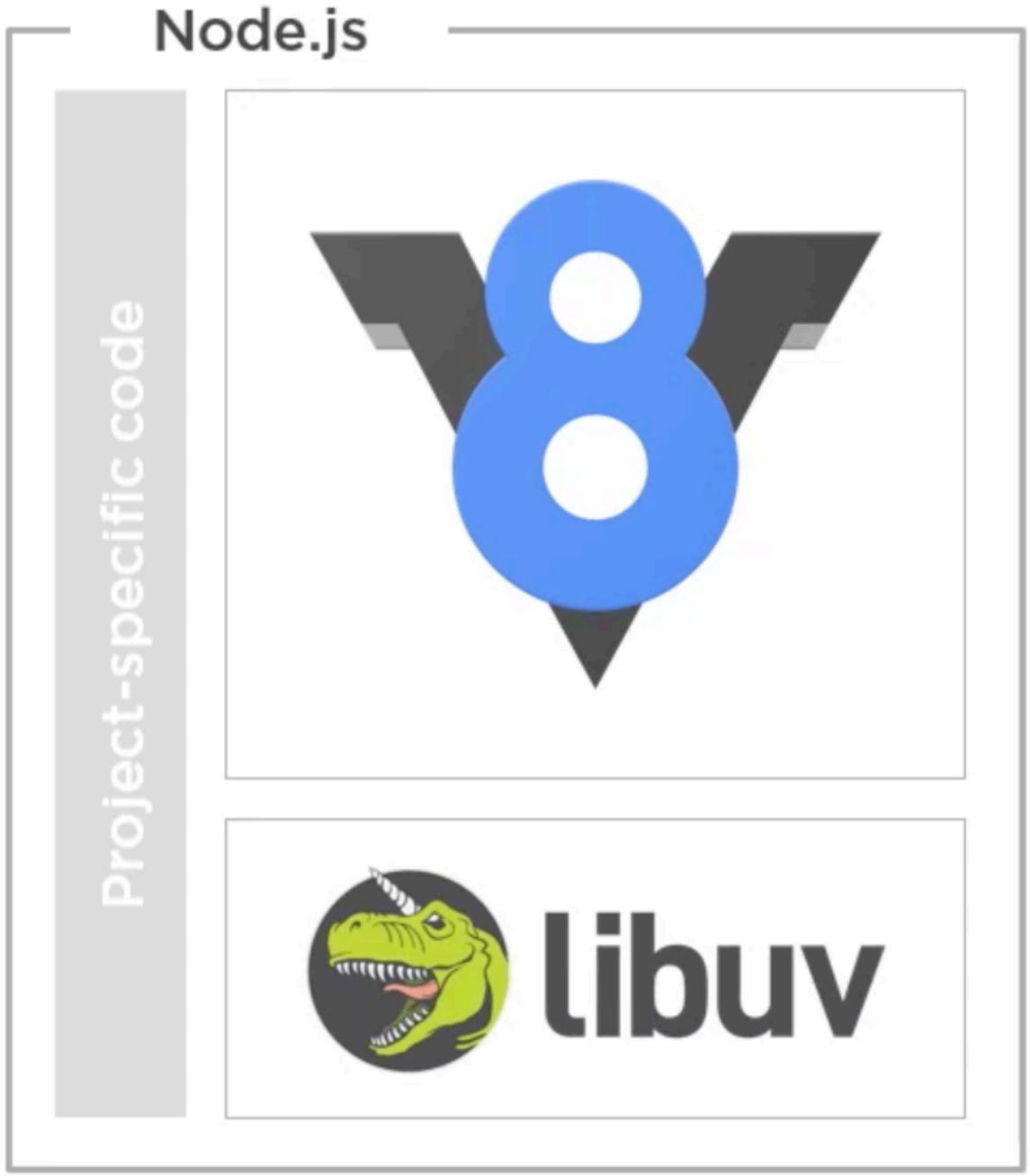
Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world.

https://nodejs.org/en/

What Node.js ?



What Node.js ?

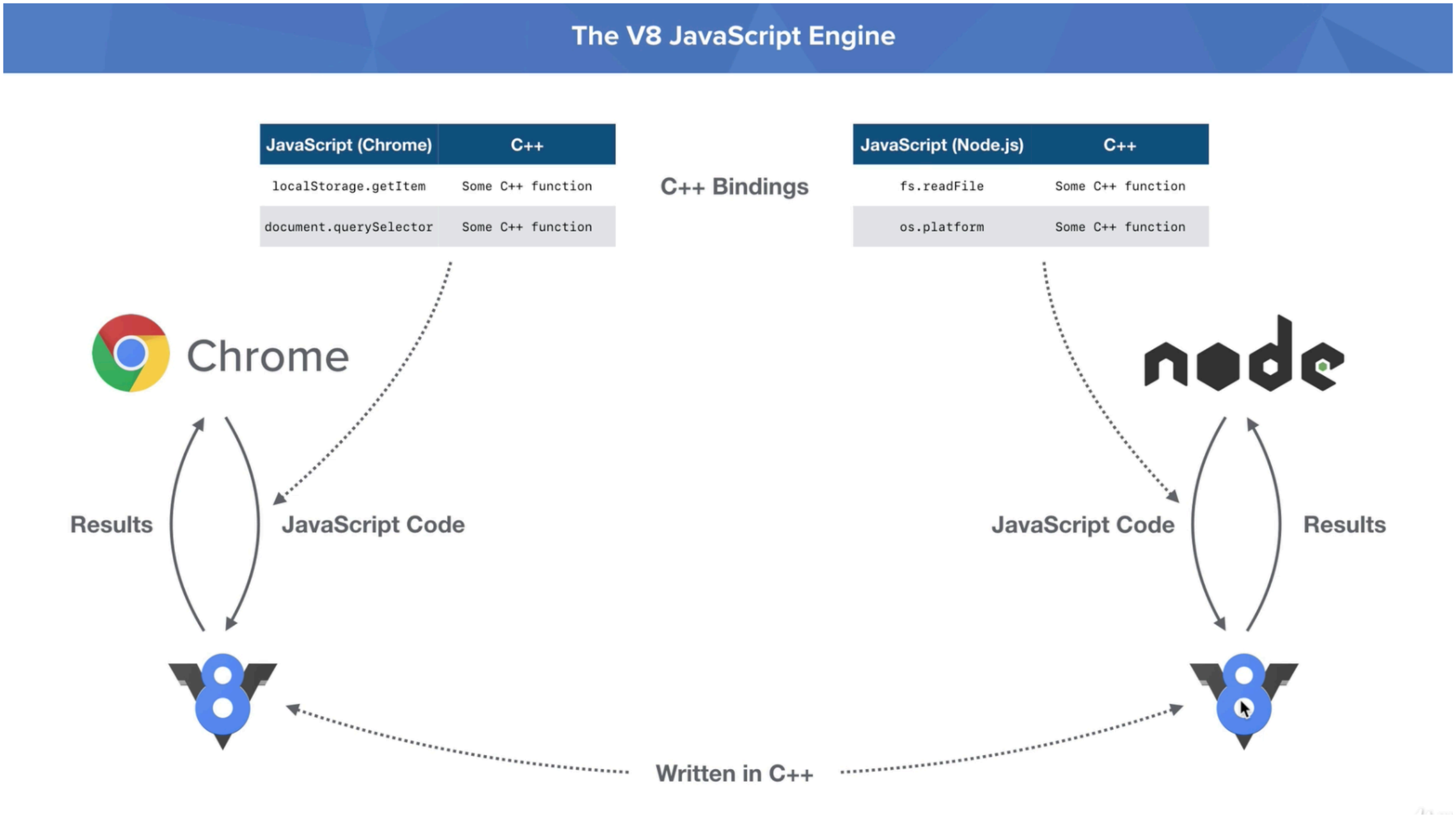


- JavaScript

- Asynchronous I/O
- Event Loop

https://libuv.org/

What Node.js ?

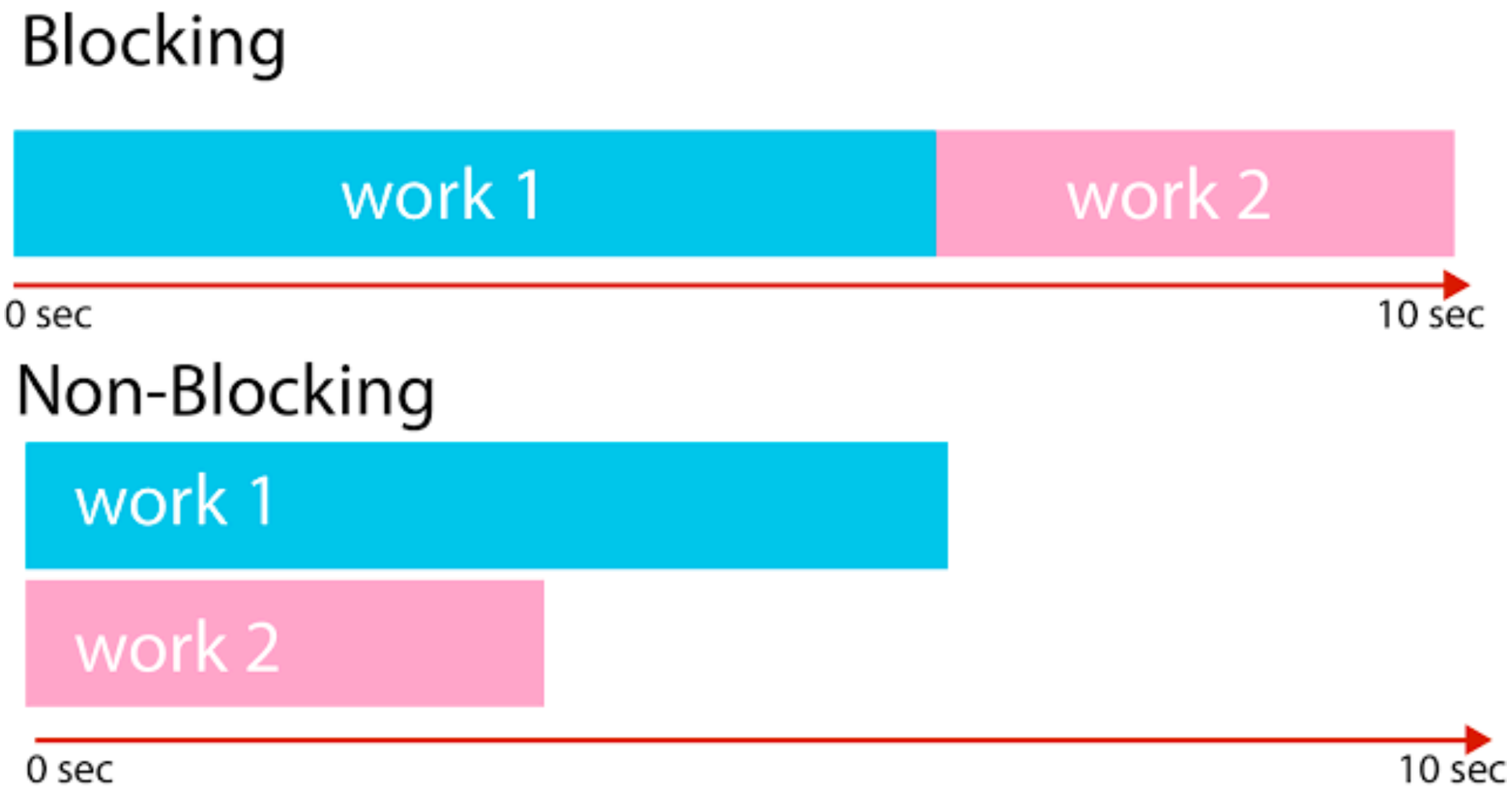


What Node.js ?

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world.

https://nodejs.org/en/

Blocking vs Non-blocking



Run Node.js

$node <file>

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

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World');
});
```

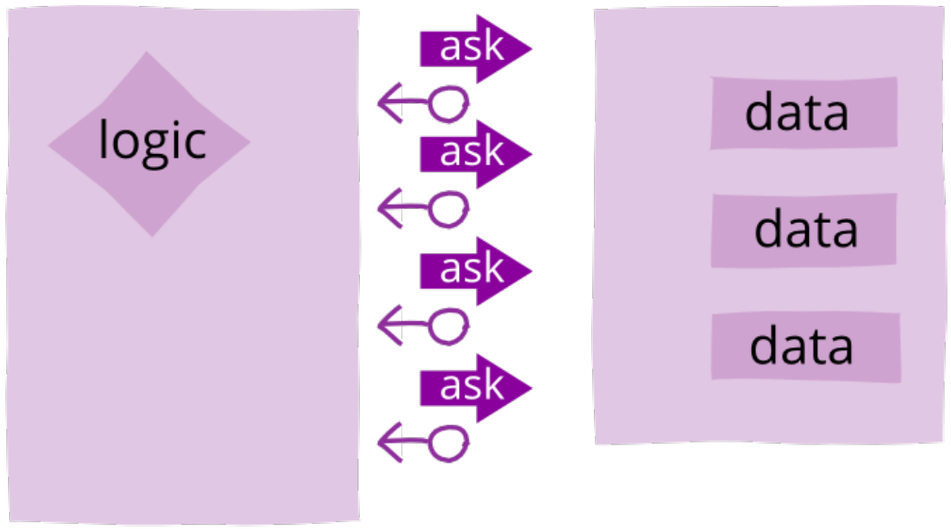
```
server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

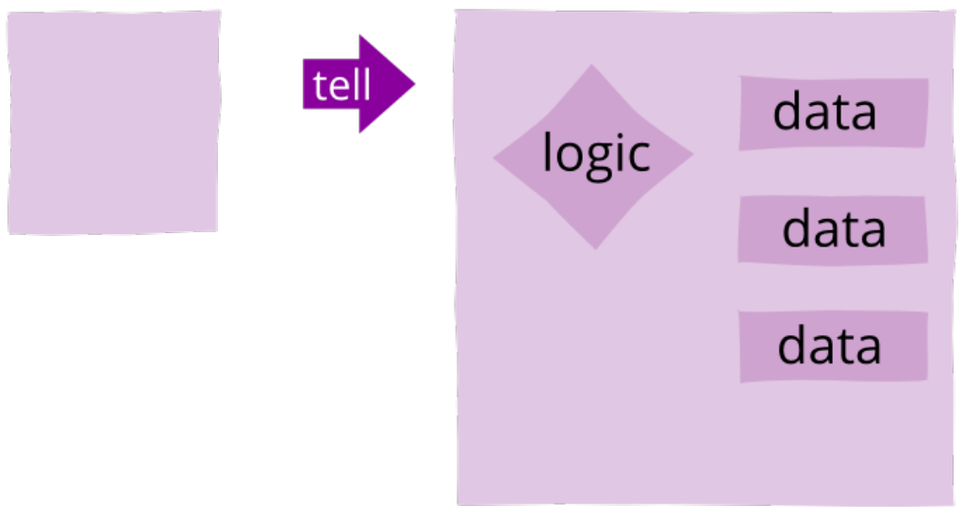
```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

Callback function

Callback function

Tell Don’t Ask





https://martinfowler.com/bliki/TellDontAsk.html

Function in Node.js

Function

First-class citizen in Node.js

```
function sayHi(name) {
  console.log(`Say hi ${name}`);
}

function add(a, b) {
  return a + b;
}

function show(fn) {
  console.log(`Result⇒${fn}`);
}
```

Function

First-class citizen in Node.js

```
const sayHi = function (name) {
  console.log(`Say hi ${name}`);
};

const add = function (a, b) {
  return a + b;
};

const show = function (fn) {
  console.log(`Result⇒${fn}`);
};
```

Arrow function in ES6

```
const sayHi = (name) => {
  console.log(`Say hi ${name}`);
};

const add = (a, b) => {
  return a + b;
};

const show = (fn) => {
  console.log(`Result=>${fn}`);
};
```

Error conventions in Node.js

First argument of callback function is an error

```
const callback = function(error, returnValue) {
    if(error) {
        console.log(error);
        return;
    }
    console.log(returnValue);
}

const someFunction = function(value, callback) {
    if(value) {
        callback(null, "Without error");
    } else {
        callback(new Error("With error"));
    }
}
```

const vs let vs var

const vs let vs var

const (default in ES6)
the identifier can’t be reassigned

let
the identifier may be reassigned (auto-rebind)

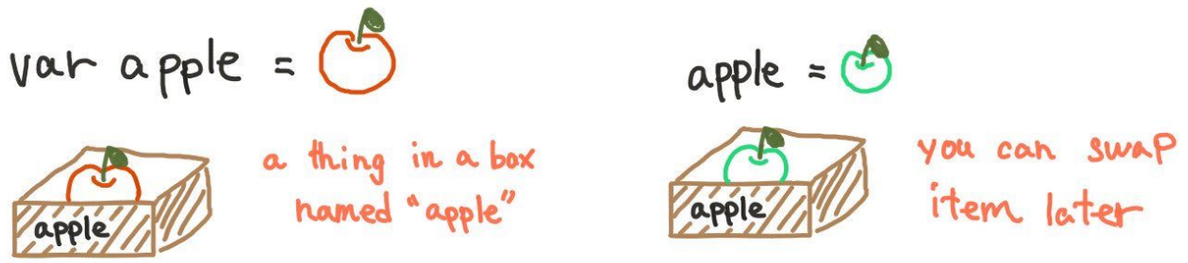
var
don’t use !!
use var to signal untouched legacy code

const vs let vs var

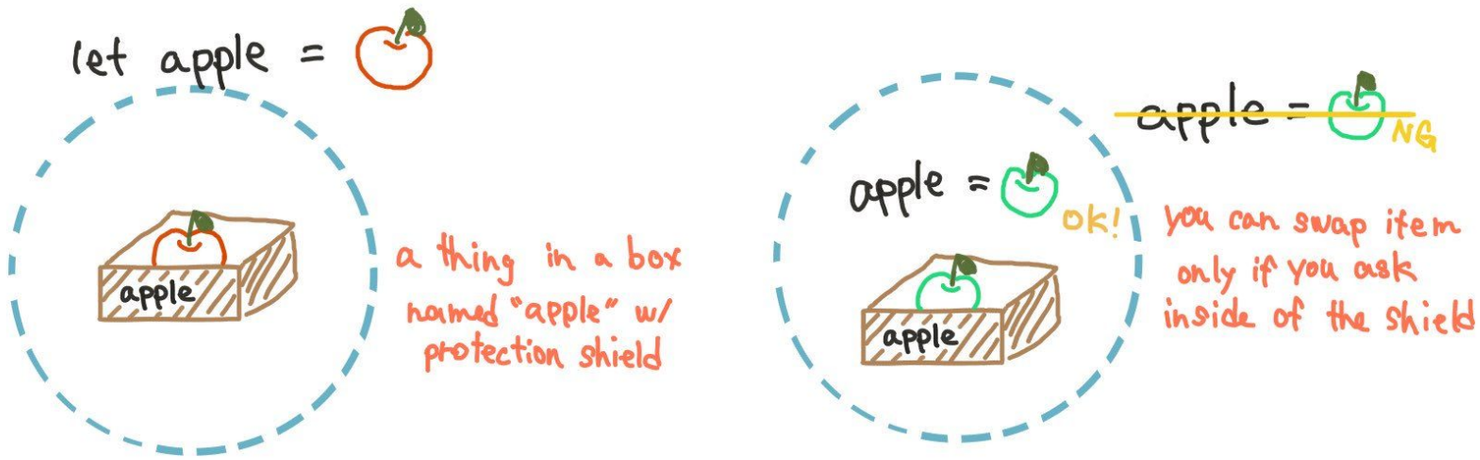
```
const name= 'somkiat';
var count = 1;

if(true) {
    let count = 2;
    name = 'new name'; // TypeError: Assignment to constant variable.
}

console.log(count); // 1
```

const vs let vs var

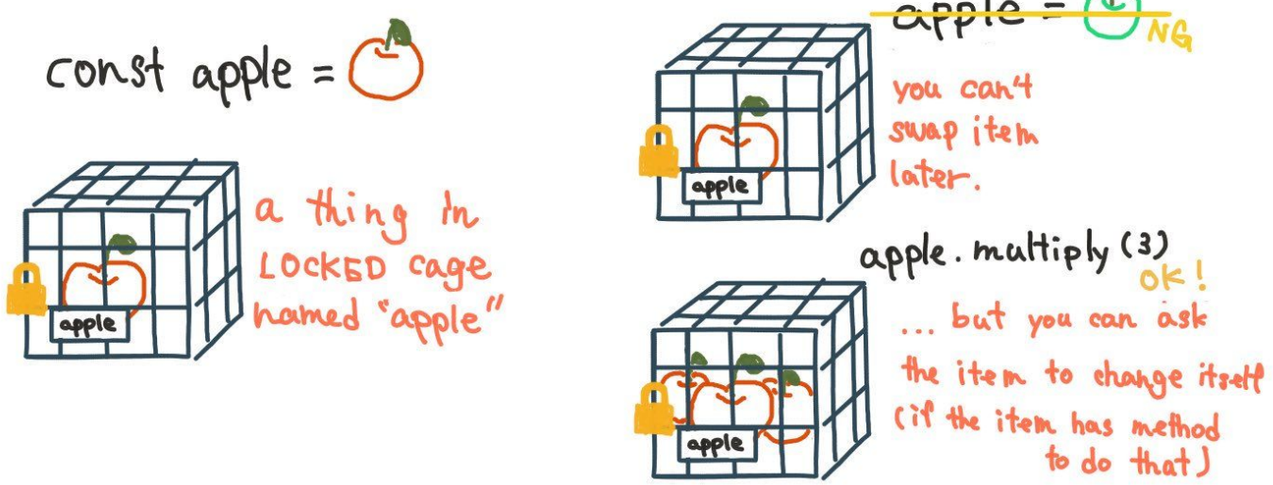
var



let



const



Working with Array

```
const products = ['Beer', 'Rice'];
products.push('New beer');

for (let index = 0; index < products.length; index++) {
    console.log(products[index]);
}

for (const product of products) {
    console.log(product);
}

products.forEach(function(product){
    console.log(product);
});

products.forEach((product) => {
    console.log(product);
});
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array

Array operation

ForEach / Iterate
Filter
Map
Reduce
Find
etc…

For … of vs For .. in

For … in
Loop over an enumerable

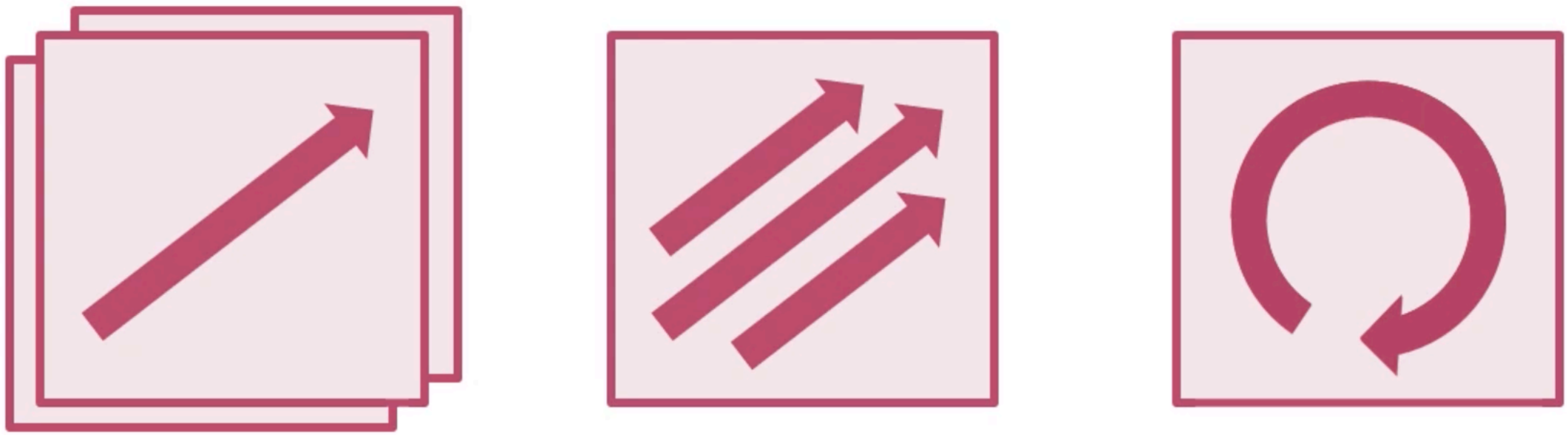
For … of
Loop over an iterable

Thinking Asynchronous

Overview

Event loop
Asynchronous development
Node APIs and EventEmitters
Stream

How Node.js work ?

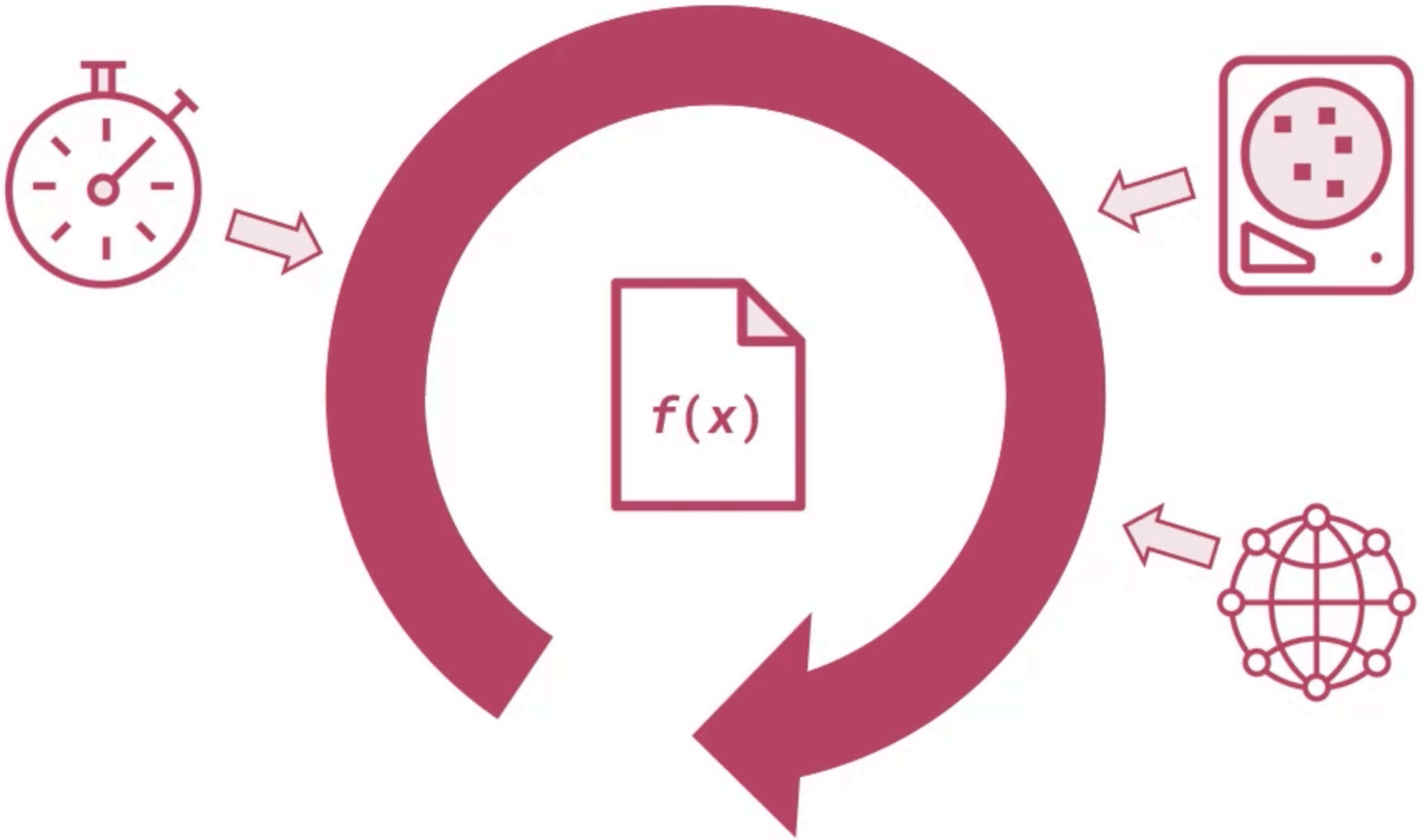


Process-Per-Client
(Multi-Process)

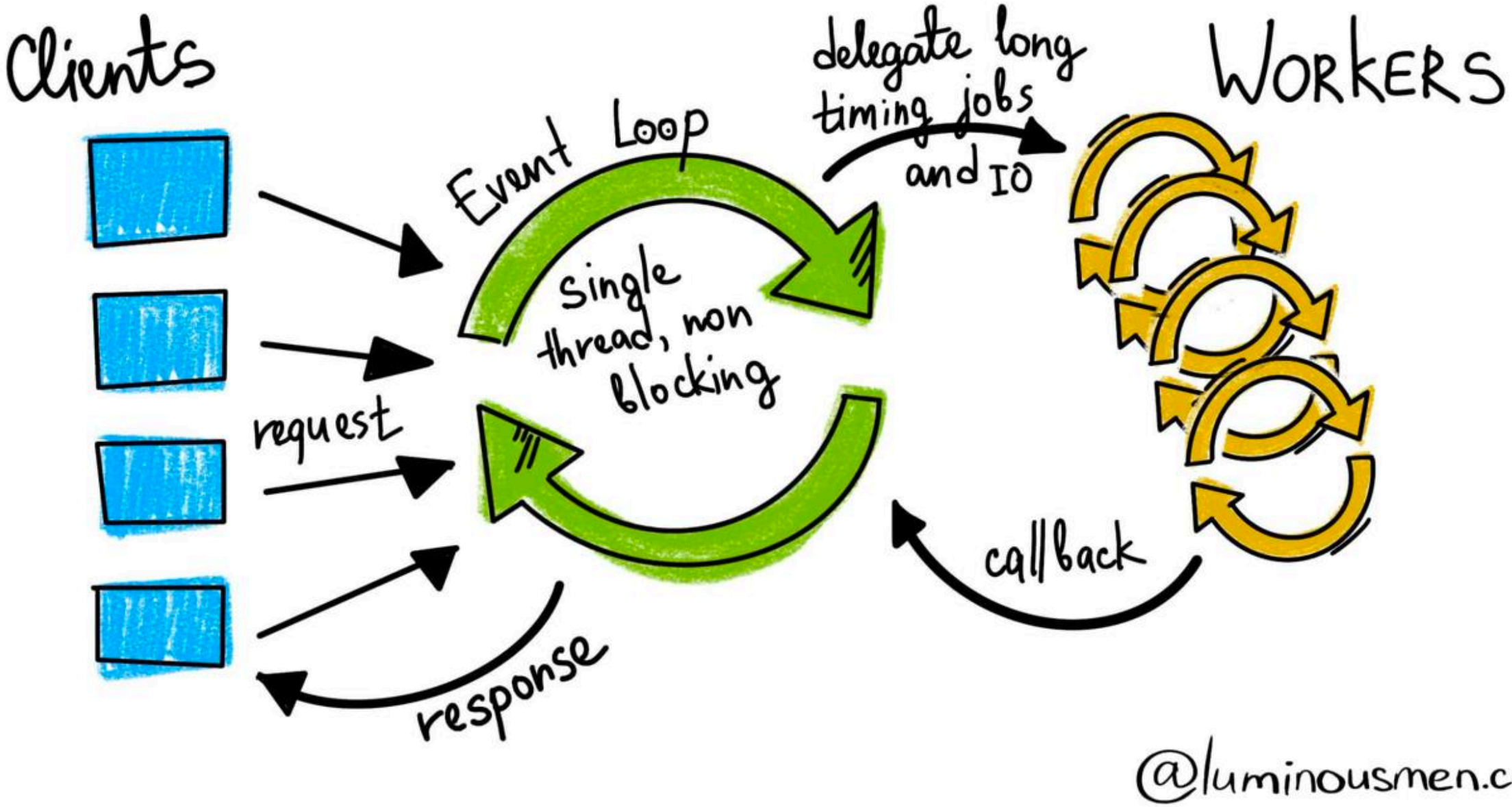
Thread-Per-Client
(Multi-Threaded)

Event Loop
(“Single Threaded”)

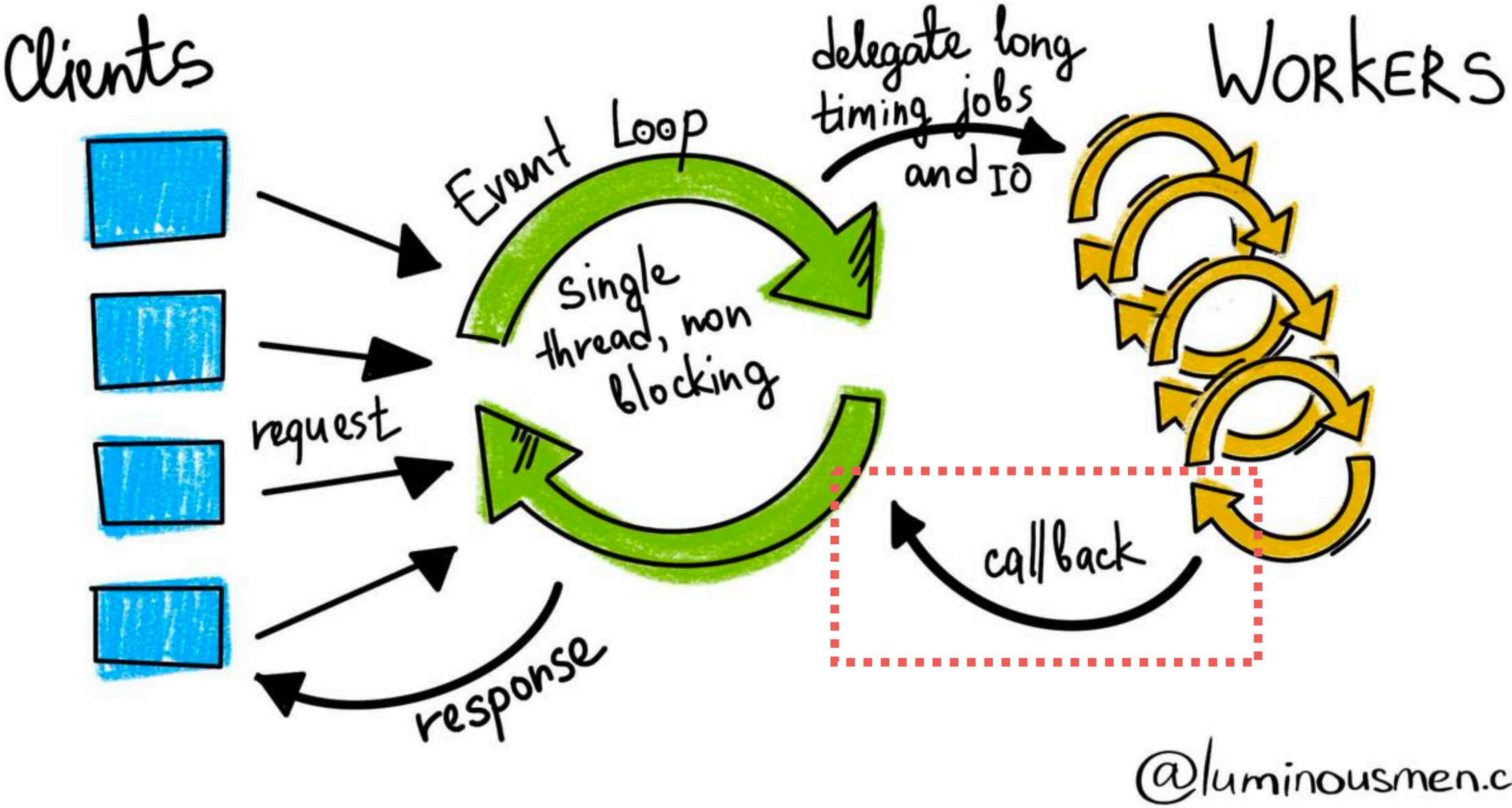
Node’s Event Loop



Node's Event Loop



Node's Event Loop



Sync vs Async

Synchronous programming

```
function serveCustomer(customer) {
  let order = customer.placeOrder(menu)
  let food = cook.prepareFood(order)
  let tip = customer.eatAndPay(food)
  return tip
}
```

Asynchronous programming !!

Callbacks
Async.js
Promises
Async/Await
Generators/yield
EventEmitter
Functor + chaining + composition

Working with callbacks

```
function serveCustomer(customer, done) {
  customer.placeOrder(menu, (error, order) => {
    cook.prepareFood(order, (error, food) => {
      customer.eatAndPay(food, done)
    }
  }
}
```

Callbacks hell

```
function serveCustomer(customer, done) {
  customer.placeOrder(menu, (error, order) => {
    cook.prepareFood(order, (error, food) => {
      customer.eatAndPay(food, done)
    }
  }
}
```

http://callbackhell.com/

Callback hell
Pyramid of doom
Christmas tree

Promises and Async/Await

Working with promises

```
function add(a, b) {
  return new Promise((resolve) => {
    setTimeout(() => resolve(a + b), Math.random() * 10 * 1000);
  });
}

function print(input) {
    console.log(input);
}

add(1, 3).then(result => print(result));
add(10, 30).then(print);

console.log("Processing ....");
```

Working with promises

```
function serveCustomer(customer) {
  return customer.placeOrder(menu)
    .then(order => cook.prepareFood(order))
    .then(food => customer.eatAndPay(food))
}
```

Working with async/await

```
const serveCustomer = async (customer) => {
  let order = await customer.placeOrder(menu)
  let food = await cook.prepareFood(order)
  let tip = await customer.eatAndPay(food)
  return tip
}
```

EventEmitter

https://nodejs.org/api/events.html

Working with EventEmitter

```
const serveCustomer = (customer, done) => {
  customer.on('decided', order => {
    order.on('prepared', food => customer.eatAndPay(food))
    cook.prepareFood(order)
  })
  customer.on('leaving', tip => done(null, tip))
  customer.placeOrder(menu)
}
```

Stream

https://nodejs.org/api/stream.html

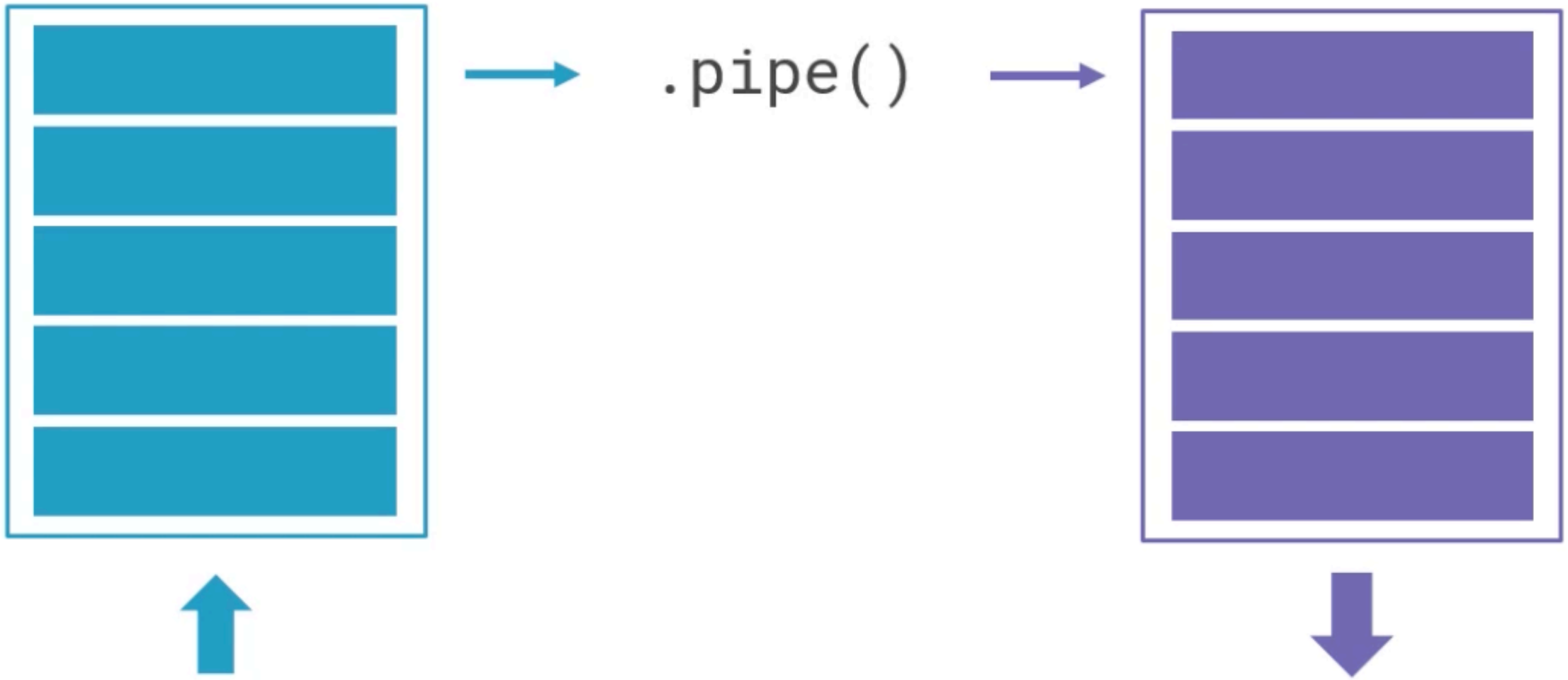
Types of Stream

Readable Streams

Read/Write Streams
(Duplex, Transform)

Writable Streams

Piping Stream



Stream in Node.js APIs

fs.ReadStream

fs.WriteStream

http.ClientRequest

http.IncomingMessage

http.ServerResponse

zlib.createGzip()

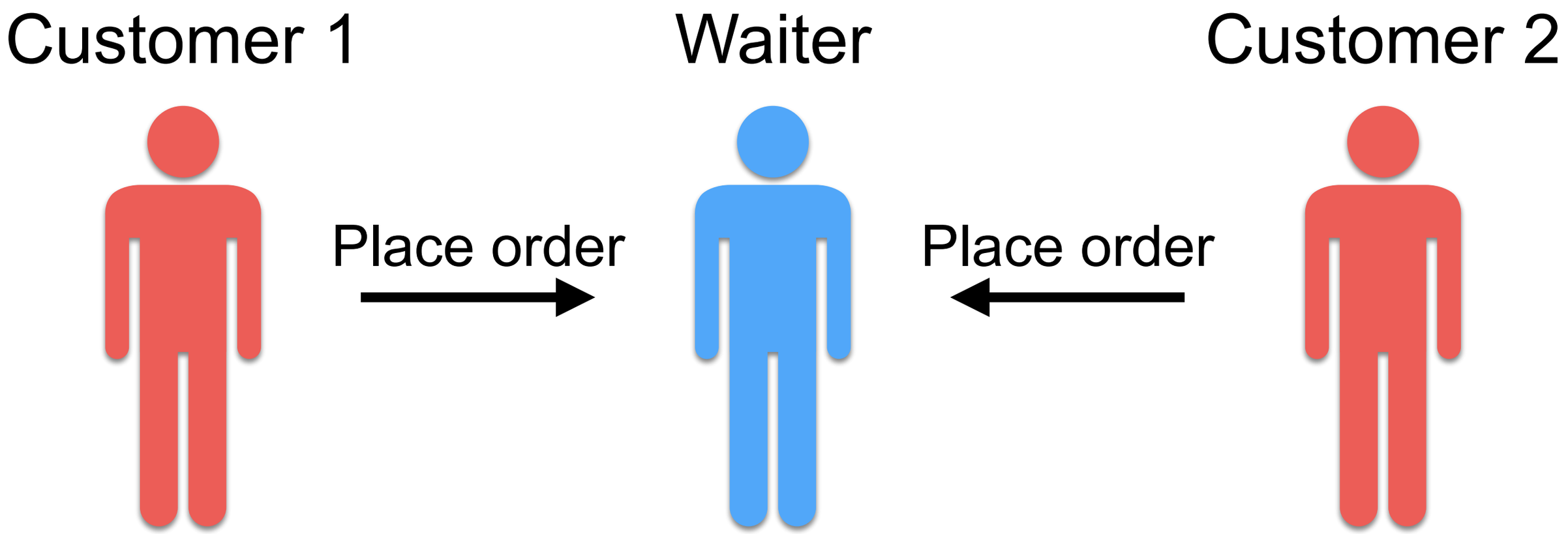
Example

```
const server = http.createServer((req, res) => {
  res.setHeader("Content-Type", "text/plain");
  res.setHeader("Content-Encoding", "gzip");

  fs.createReadStream(path.join(__dirname, "data.txt"))
    .pipe(zlib.createGzip())
    .pipe(res);
});
```

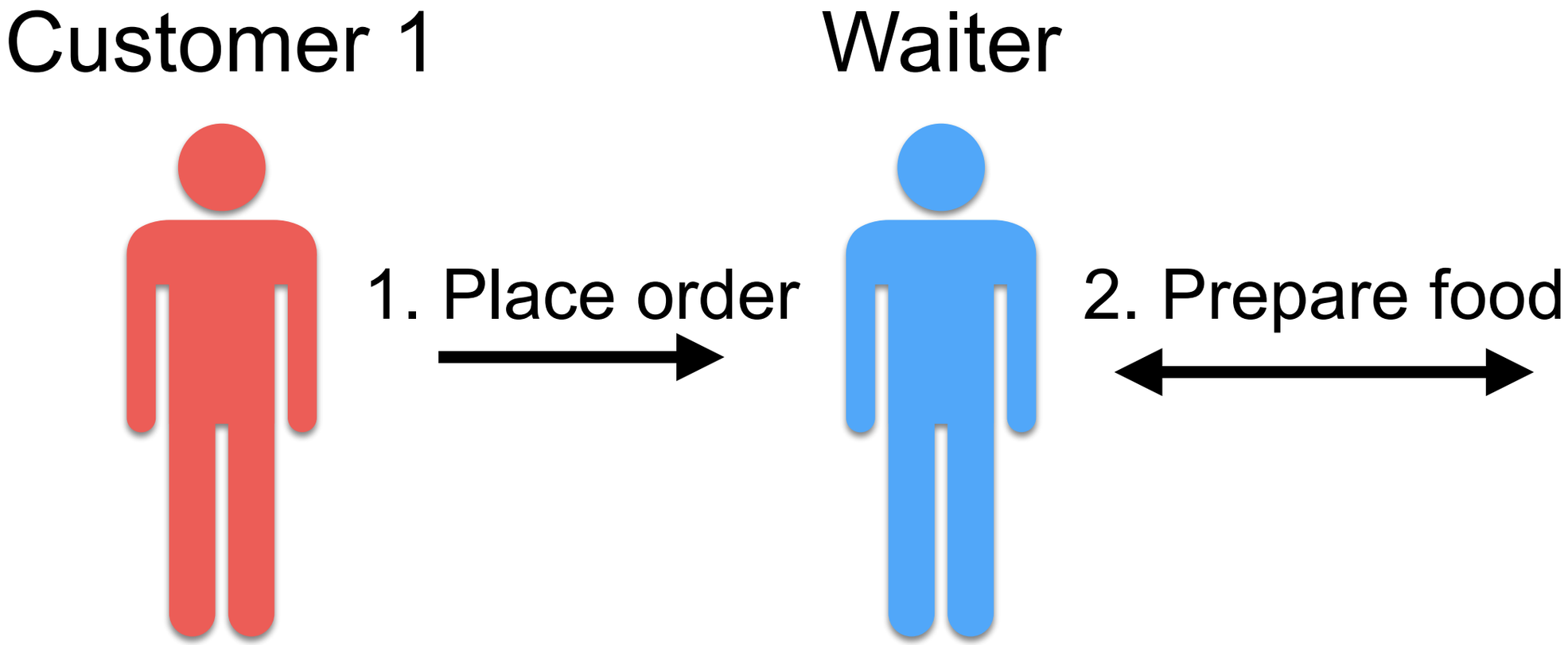
Workshop

Callback, Promise, Async/Await, Event

Workshop



Workshop



Flow

Customer

↓

serve(customer, menu)

↓

placeOrder(menu)

↓

prepareFood(order)

↓

eatAndPay(food)

↓

Done

Start with Callback

Callback

```
const placeOrder = (menu, done) => {
  setTimeout(() => done(null, menu), Math.ceil(Math.random() * 5) * 1000);
};

const prepareFood = (order, done) => {
  setTimeout(() => done(null, order), Math.ceil(Math.random() * 1) * 1000);
};
```

```
const eatAndPay = (food, done) => {
  setTimeout(() => done(null, food), Math.ceil(Math.random() * 10) * 1000);
};
```

Callback

```
const serve = (customer, menu, done) => {
  console.log(`Start serve with ${customer.name} with menu= ${menu}`);

  placeOrder(menu, (error, order) => {
    if (!error) {
      console.log(`Prepare food of ${customer.name}`);

      prepareFood(order, (error, food) => {
        if (!error) {
          console.log(`Customer ${customer.name} is eat and pay`);
          eatAndPay(food, done);
        }
      });

    }
  });
};
```

Run …

Start serve with customer_1 with menu= Chicken grill
Start serve with customer_2 with menu= Ham burger
Start serve with customer_3 with menu= Beer
Start serve with customer_4 with menu= Chicken grill
Start serve with customer_5 with menu= Ham burger
Prepare food of customer_1
Prepare food of customer_5
Customer customer_1 is eat and pay
Customer customer_5 is eat and pay
Prepare food of customer_2
Prepare food of customer_4
Prepare food of customer_3
Customer customer_2 is eat and pay
Customer customer_4 is eat and pay
Customer customer_3 is eat and pay
>> Done for customer=customer_5 with food=Ham burger
>> Done for customer=customer_2 with food=Ham burger
>> Done for customer=customer_3 with food=Beer
>> Done for customer=customer_4 with food=Chicken grill
>> Done for customer=customer_1 with food=Chicken grill

Refactor to OOP

OOP in Node.js

```
class Rectangle {
  constructor(height, width) {
    this.height = height;
    this.width = width;
  }
  // Getter
  get area() {
    return this.calcArea();
  }
  // Method
  calcArea() {
    return this.height * this.width;
  }
}

const square = new Rectangle(10, 10);

console.log(square.area); // 100
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Classes

OOP in Node.js

	Chrome	Edge	Firefox	Internet Explorer	Opera	Safari	Android webview	Chrome for Android	Firefox for Android	Opera for Android	Safari on iOS	Samsung Internet	Node.js
`classes`	49	13	45	No	36	9	49	49	45	36	9	5.0	6.0.0
`constructor`	49	13	45	No	36	9	49	49	45	36	9	5.0	6.0.0
`extends`	49	13	45	No	36	9	49	49	45	36	9	5.0	6.0.0
Private class fields	74	79	No	No	62	No	74	74	No	53	No	No	12.0.0
Public class fields	72	79	69	No	60	No	72	72	No	51	No	No	12.0.0
`static`	49	13	45	No	36	9	49	49	45	36	9	5.0	6.0.0
Static class fields	72	79	75	No	60	No	72	72	No	51	No	No	12.0.0

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Classes

Design with Class

Customer	Waiter	Kitchen
placeOrder() eatAndPay()	serve()	prepareFood()

Customer

```
class Customer {
  constructor(id, name) {
    this.id = id;
    this.name = name;
  }

  placeOrder = (menu, done) => {
    setTimeout(() => done(null, menu), Math.ceil(Math.random() * 5) * 1000);
  };

  eatAndPay = (food, done) => {
    setTimeout(() => done(null, food), Math.ceil(Math.random() * 10) * 1000);
  };
}


module.exports = Customer;
```

Kitchen

```
const prepareFood = (order, done) => {
  setTimeout(() => done(null, order), Math.ceil(Math.random() * 1) *
1000);
};

module.exports = { prepareFood };
```

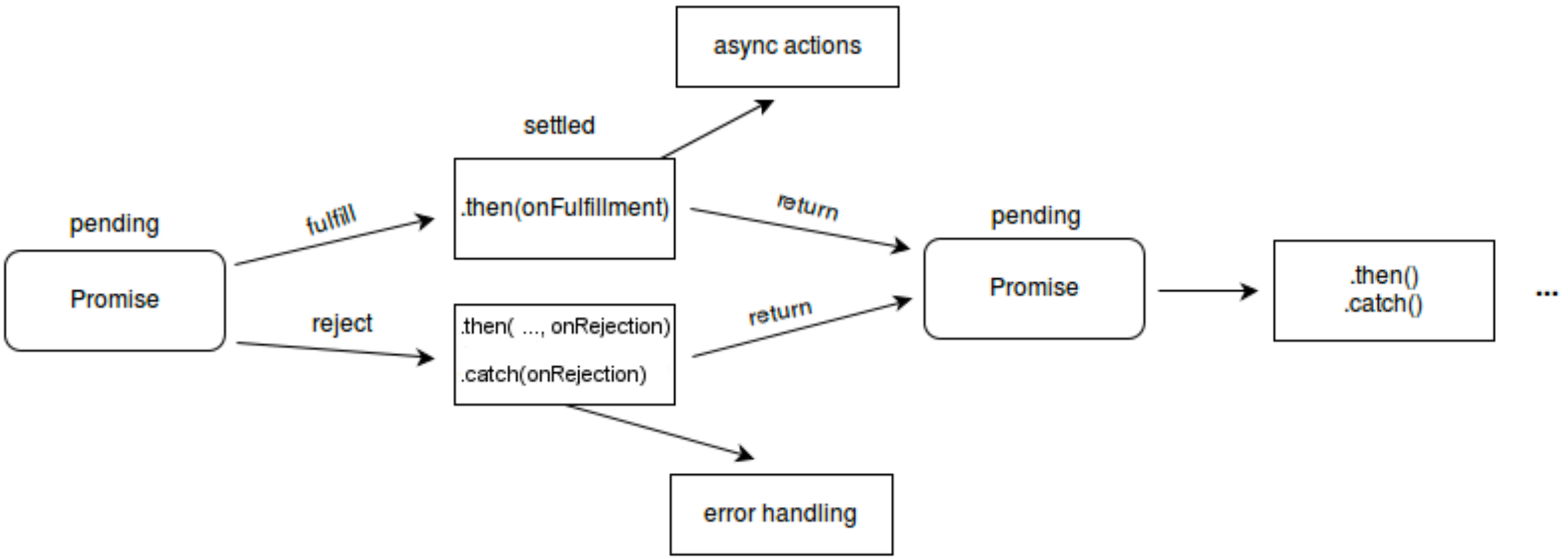
Main

```
const Customer = require("./customer.js");
const kitchen = require("./kitchen.js");

const serve = (customer, menu, done) => {
  console.log(`Start serve with ${customer.name} with menu= ${menu}`);

  customer.placeOrder(menu, (error, order) => {
    if (!error) {
      console.log(`Prepare food of ${customer.name}`);
      kitchen.prepareFood(order, (error, food) => {
        if (!error) {
          console.log(`Customer ${customer.name} is eat and pay`);
          customer.eatAndPay(food, done);
        }
      });

    }
  });
};
```

Promise

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise

Promise



Promise

```
const serve = (customer, menu) => {
  console.log(`Start serve with ${customer.name} with menu= ${menu}`);
  return customer.placeOrder(menu)

    .then((order) => {
      console.log(`Prepare food of ${customer.name}`);
      return kitchen.prepareFood(order);
    })

    .then((food) => {
      console.log(`Customer ${customer.name} is eat and pay`);
      return customer.eatAndPay(food);
    });
};
```

Main

```
Promise.all(
  customers.map((customer) => {
    return serve(
      customer,
      menus[Math.floor(Math.random() * 10) % menus.length]
    ).then((menu) => {
      console.log(`>> Done for customer=${customer.name} with food=${menu}`);
    });
  })
).then(() => {
  console.log("Finish all");
});
```

Async/Await

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/async_function

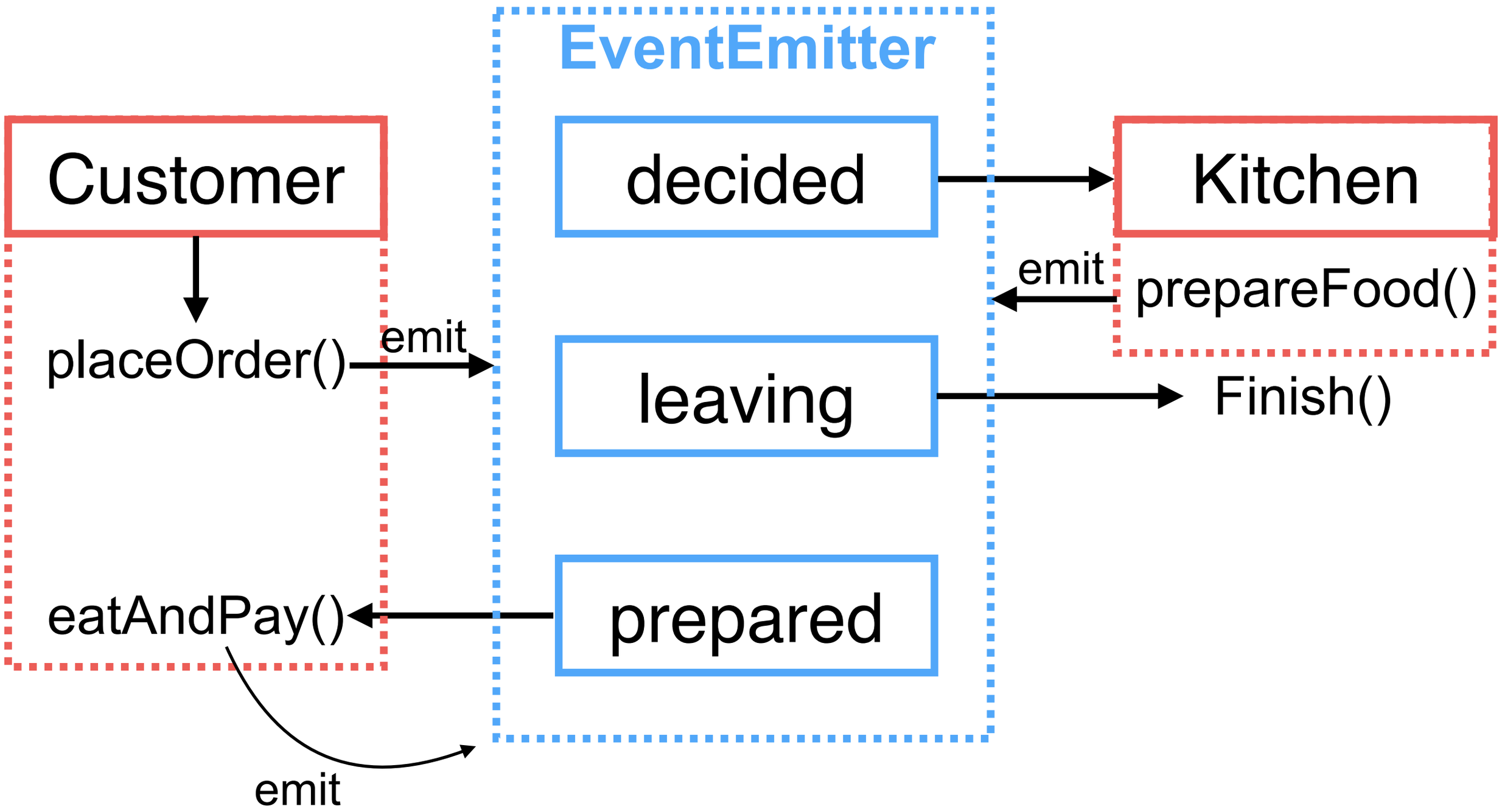
Async/Await

```
const serve = async (customer, menu) => {
  console.log(`Start serve with ${customer.name} with menu= ${menu}`);

  let order = await customer.placeOrder(menu);
  console.log(`Prepare food of ${customer.name} with order=${order}`);

  let food = await kitchen.prepareFood(order);
  console.log(`Customer ${customer.name} is eat and pay with food=${food}`);

  let result = await customer.eatAndPay(food);
  return result;
};
```

Events

https://nodejs.org/api/events.html

Events



Events

```
const serve = (customer, menu, done) => {
  console.log(`Start serve with ${customer.name} with menu= ${menu}`);

  customer.on("decided", (order, menu) => {
    console.log(`Prepare food of ${customer.name}`);

    order.on("prepared", (food) => {
      console.log(`Customer ${customer.name} is eat and pay`);
      customer.eatAndPay(menu, done);
    });

    kitchen.prepareFood(order);
  });

  customer.on("leaving", (result) => done(null, result));

  customer.placeOrder(menu);
};
```

NodeJS Module System

Import module

Node.js core modules
From your own files
NPM modules

What Node.js ?

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world.

https://nodejs.org/en/

NPM
(Node Package Management)

https://www.npmjs.com/

YARN
(Package Management)

https://yarnpkg.com/

Using Nodemon

https://nodemon.io/

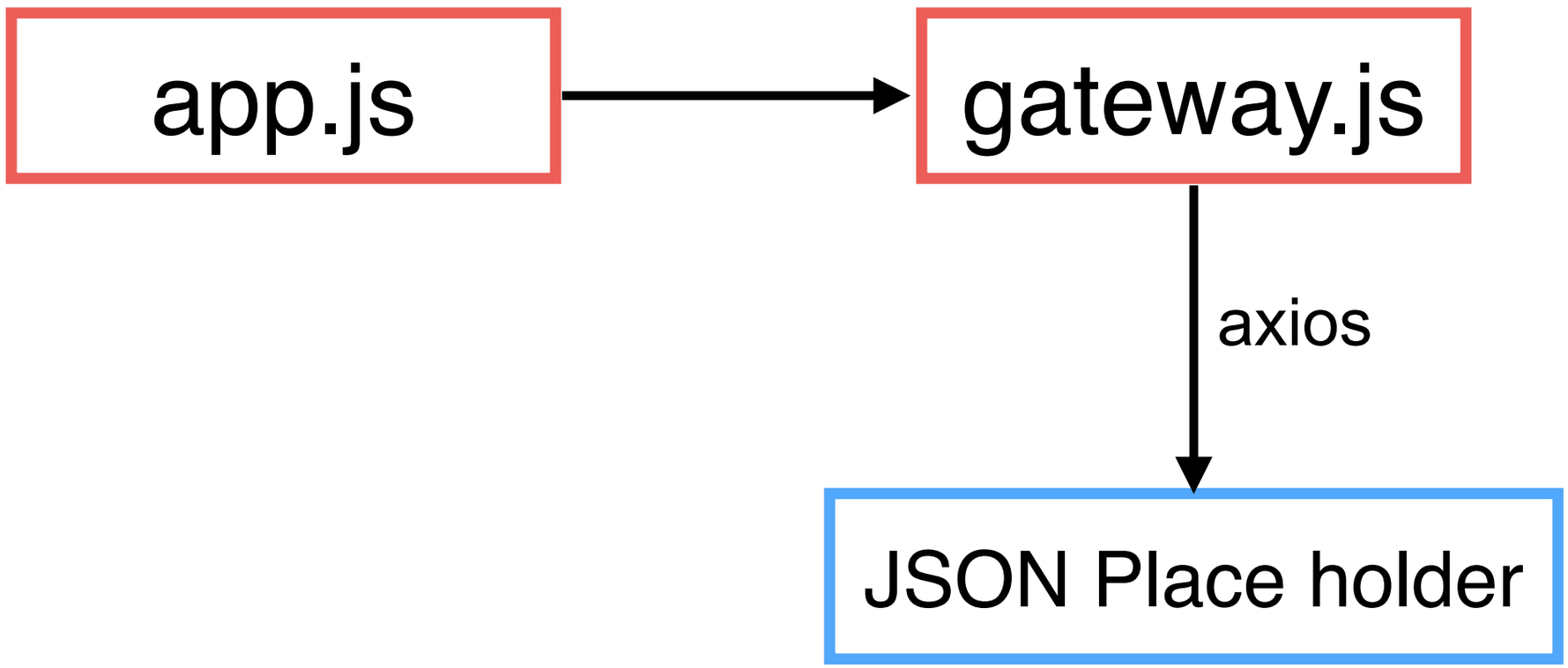
Nodemon

Auto-reload

$npm install -g nodemon

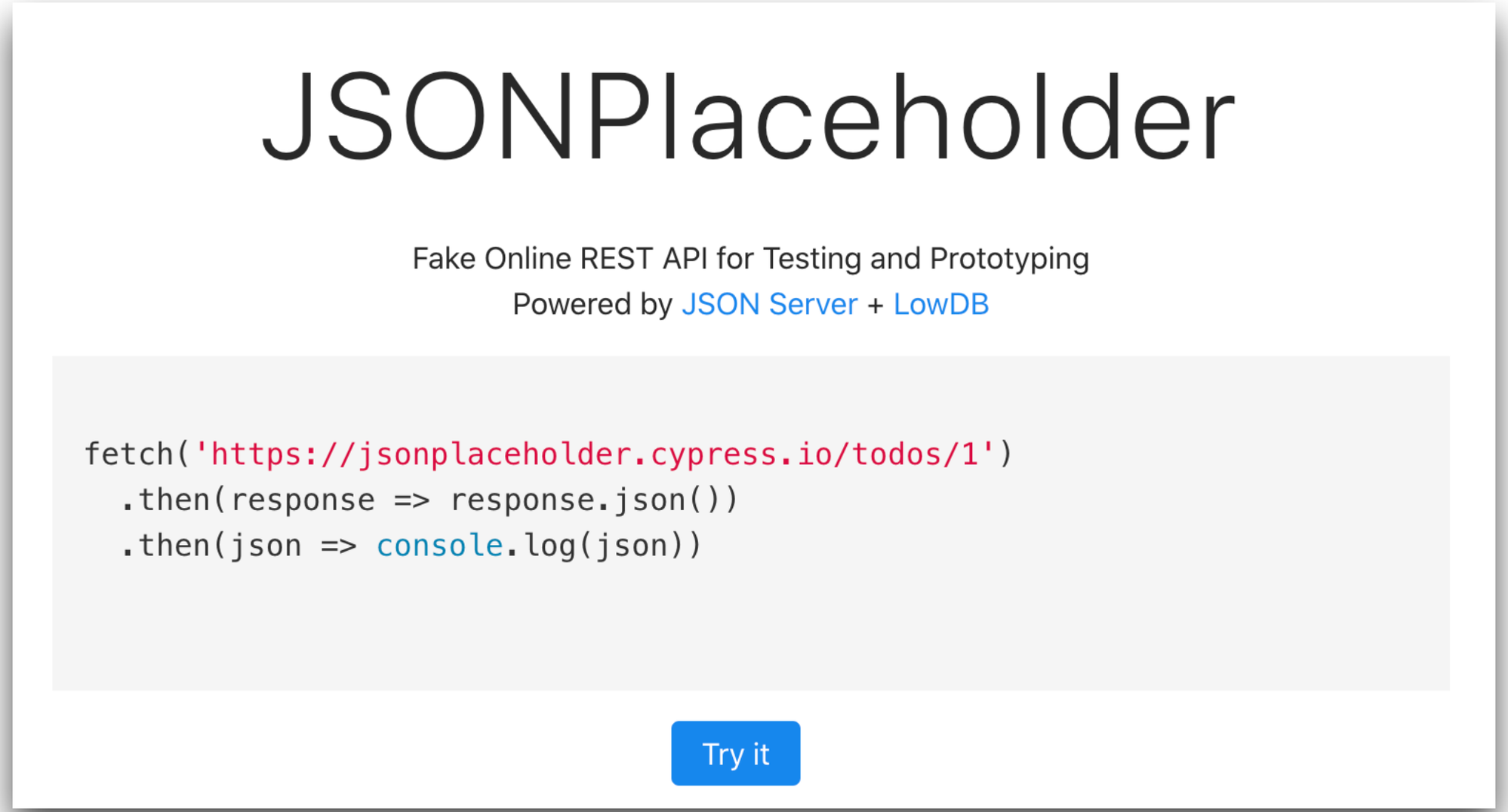
Workshop
Asynchronous with Node.js

Workshop



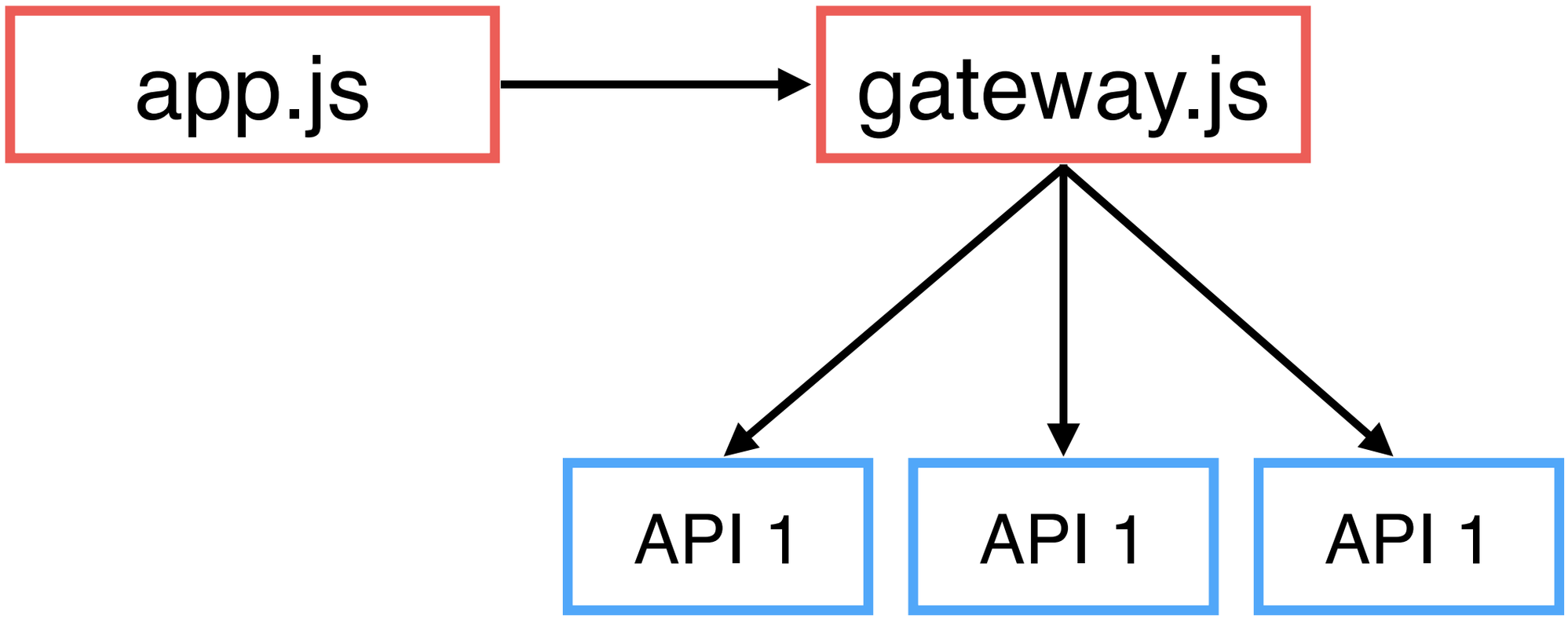
https://github.com/axios/axios

JSON Place holder



https://jsonplaceholder.cypress.io/users

Workshop with Promise.All

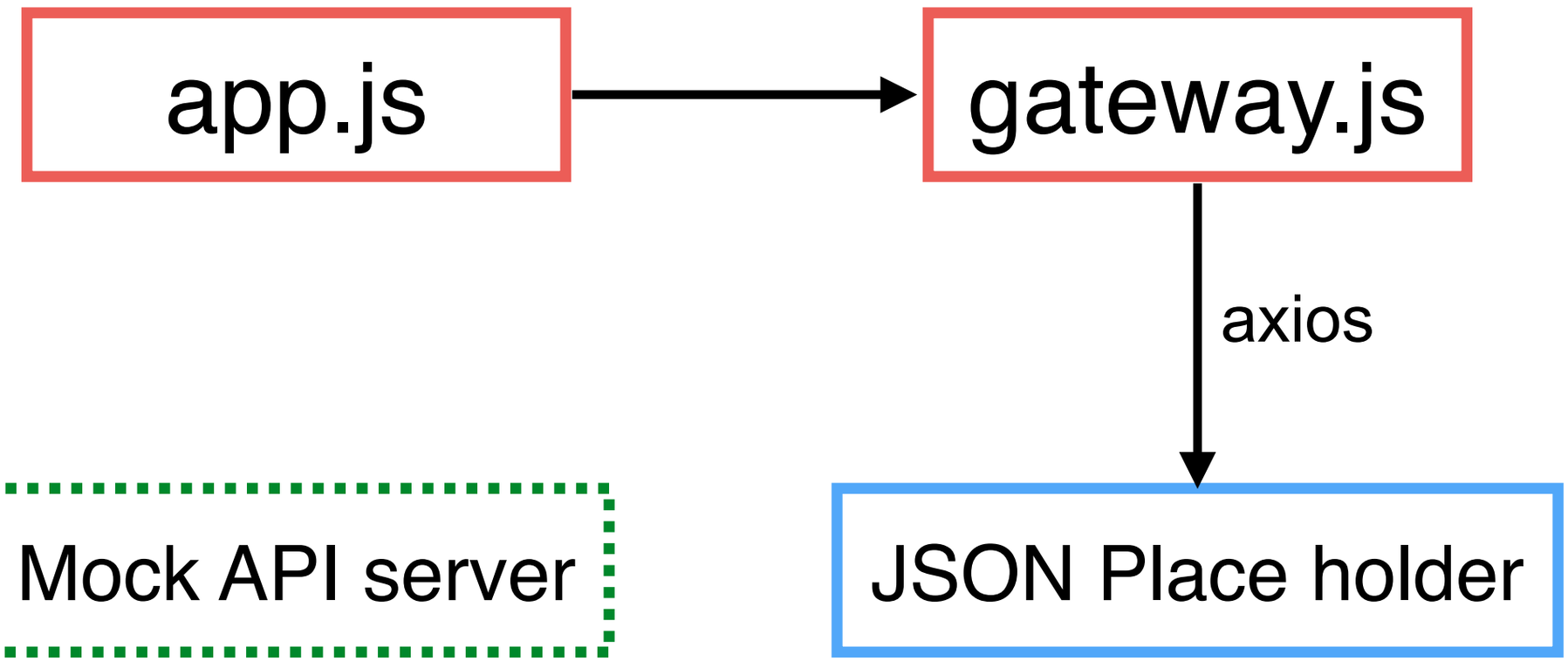


https://github.com/axios/axios

Testing API with Nock

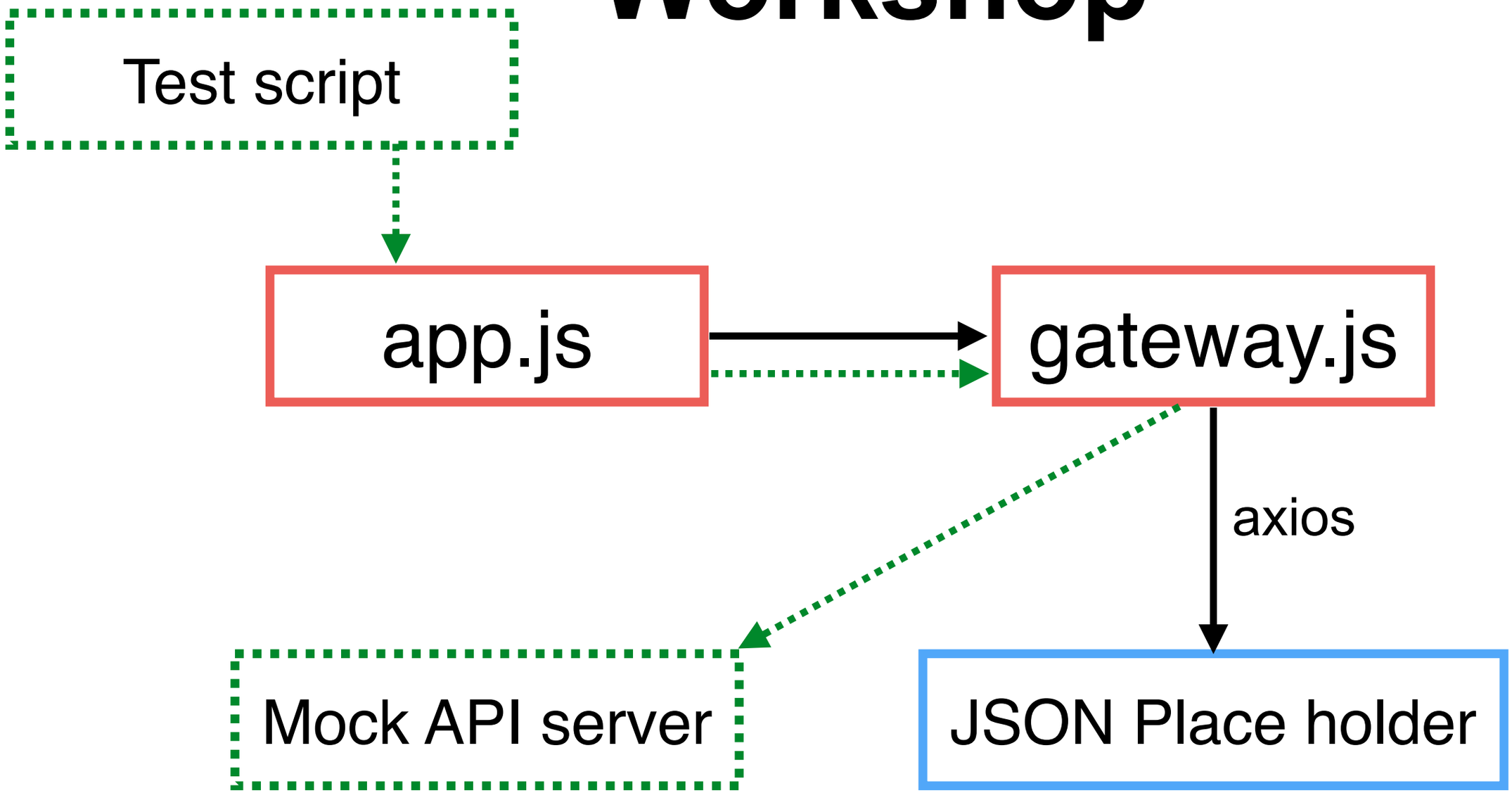
https://github.com/nock/nock

Workshop



https://github.com/axios/axios

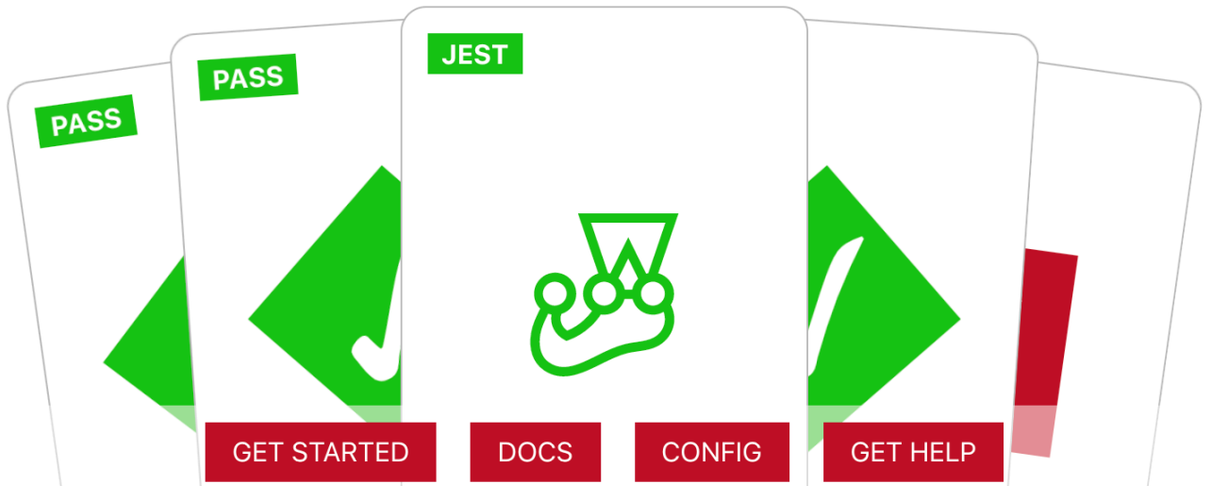
Workshop



https://github.com/axios/axios

Jest

JavaScript Test Framework




```
Jest is a delightful JavaScript Testing Framework
with a focus on simplicity.

It works with projects using: Babel, TypeScript,
Node, React, Angular, Vue and more!
```

http://jestjs.io/

Jest

JavaScript Test Framework

$npm install jest --save-dev

Create first test case

```
test("Hello world!", () ⇒ {

});
```

Assertion

```
const { calculateTip } = require("../src/math");

test("Should calculate total with tip", () => {
  const total = calculateTip(10, 0.3);

  expect(total).toBe(13);
});
```

https://jestjs.io/docs/en/expect

Testing asynchronous code

With callback function

```
test("Should add two numbers", (done) => {
  add(2, 3).then((sum) => {
    expect(sum).toBe(5);
    done();
  });
});
```

Testing asynchronous code

With async/await

```
test("Should add two numbers async/await", async () => {
  const sum = await add(2, 3);
  expect(sum).toBe(3);
});
```

Jest setup and teardown

beforeEach
afterEach
before
after

Mocking with Jest

Create directory **__mocks__** in tests

Working with jest.mock()

Jest.spyOn

Jasmine

BDD for JavaScript

https://jasmine.github.io/

Nock

HTTP server mocking and expectations library for Node.js

$npm install nock --save-dev

Nock

HTTP server mocking and expectations library for Node.js

$npm install nock --save-dev

Write a first test

Create folder **__tests__**
Create files with ***_spec|test.js**

Gateway-spec.js

```
const nock = require("nock");
const API_PORT = 9999;
const getAllUser = require("../gateway-testable.js");
const API_HOST = `http://localhost:${API_PORT}`;

describe("Call service", () => {

  it("Check response from /users", async () => {
    // Mock server
    nock(API_HOST)
    .defaultReplyHeaders({ 'access-control-allow-origin': '*' })
    .get("/users").reply(200, [{}, {}]);

    // Verify
    const response = await getAllUser();
    expect(response.data.length).toEqual(2);
  });

});
```

Config Jest in package.json

```
"jest": {
    "collectCoverage": true,
    "coverageReporters": ["json", "html"]
}
```

Run test

$npm test

```
FAIL __tests__/gateway-spec.js
  Call service
    × Check response from /users (886 ms)

  ● Call service › Check response from /users

    expect(received).toEqual(expected) // deep equality

    Expected: 2
    Received: 10

       9 |
      10 |     const response = await getAllUser();
    > 11 |     expect(response.data.length).toEqual(2);
         |                                  ^
      12 |   });
      13 | });
      14 |

    at Object.<anonymous> (__tests__/gateway-spec.js:11:34)
```

Testable code ?

Using environment variable

Environment Variable

API_URL=<your api server>

Node.JS

Using **process** module

process.env('API_URL')

https://nodejs.org/api/process.html#process_process_env

Gateway.js

```
const axios = require("axios").default;

const process = require("process");

function getAllUser() {
  return axios({
    method: "get",
    url: `${process.env.API_URL}/users`,
    responseType: "json",
})
  .then((response) => {
    return {
      code: 200,
      data: response.data,
    };
  })
```

Fail case

```
it("Fail 404 /users", async () => {
    // Mock server
    nock(API_HOST)
      .defaultReplyHeaders({ "access-control-allow-origin": "*" })
      .get("/users")
      .reply(404);

    // Verify
    const response = await getAllUser();
    expect(response.code).toEqual(500);
});
```

Config Jest in package.json

```
"scripts": {
    "test": "API_URL=http://localhost:9999 jest --coverage"
  }

…

"jest": {
    "collectCoverage": true,
    "coverageReporters": ["json", "html"]
}
```

Run test

$npm test

```
PASS __tests__/gateway-spec.js
  Call service
    ✓ Check response from /users (24 ms)
    ✓ Fail 404 /users (4 ms)

Test Suites: 1 passed, 1 total
Tests:       2 passed, 2 total
Snapshots:   0 total
Time:        0.932 s, estimated 1 s
Ran all test suites.
```

Coverage report

$npm test

All files gateway-testable.js

100% Statements `6/6` **100%** Branches `0/0` **100%** Functions `3/3` **100%** Lines `6/6`

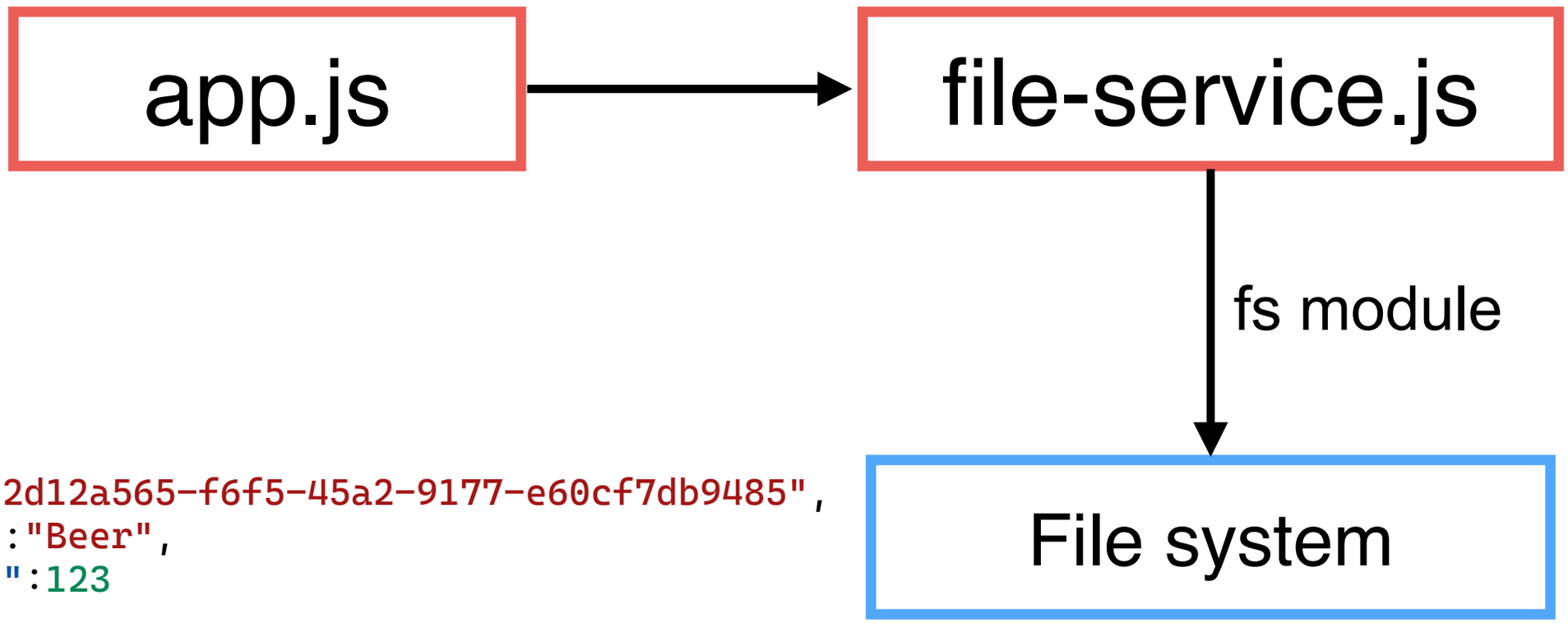
Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

```
1  1x  const axios = require("axios").default;
2  1x  const process = require("process");
3
4      function getAllUser() {
5  2x    return axios({
6          method: "get",
7          url: `${process.env.API_URL}/users`,
8          responseType: "json",
9        })
10         .then((response) => {
11 1x       return {
12           code: 200,
13           data: response.data,
14         };
```

Workshop
command-line app

Workshop

Manage JSON data with File system




```
{
  "id":"2d12a565-f6f5-45a2-9177-e60cf7db9485",
  "name":"Beer",
  "price":123
}
```

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

Workshop

Input from command-line with Yargs

add
list
read by id
update
remove

app.js

https://yargs.js.org/

Using Yargs

$node app.js **<command> <options>**

```
const yargs = require("yargs");
yargs.version("0.0.1");
yargs.usage("$0 <command> [options]");

// Create add command
yargs.command({
  command: "add",
  describe: "Add a new product",
  builder: {
    name: {
      describe: "Product name",
      demandOption: true,
      type: "string",
    },
  },
  handler(argv) {
    //TODO
  },
});
yargs.parse();
```

Workshop

Validation name not null/empty and not existing

Refactor to OOP !!

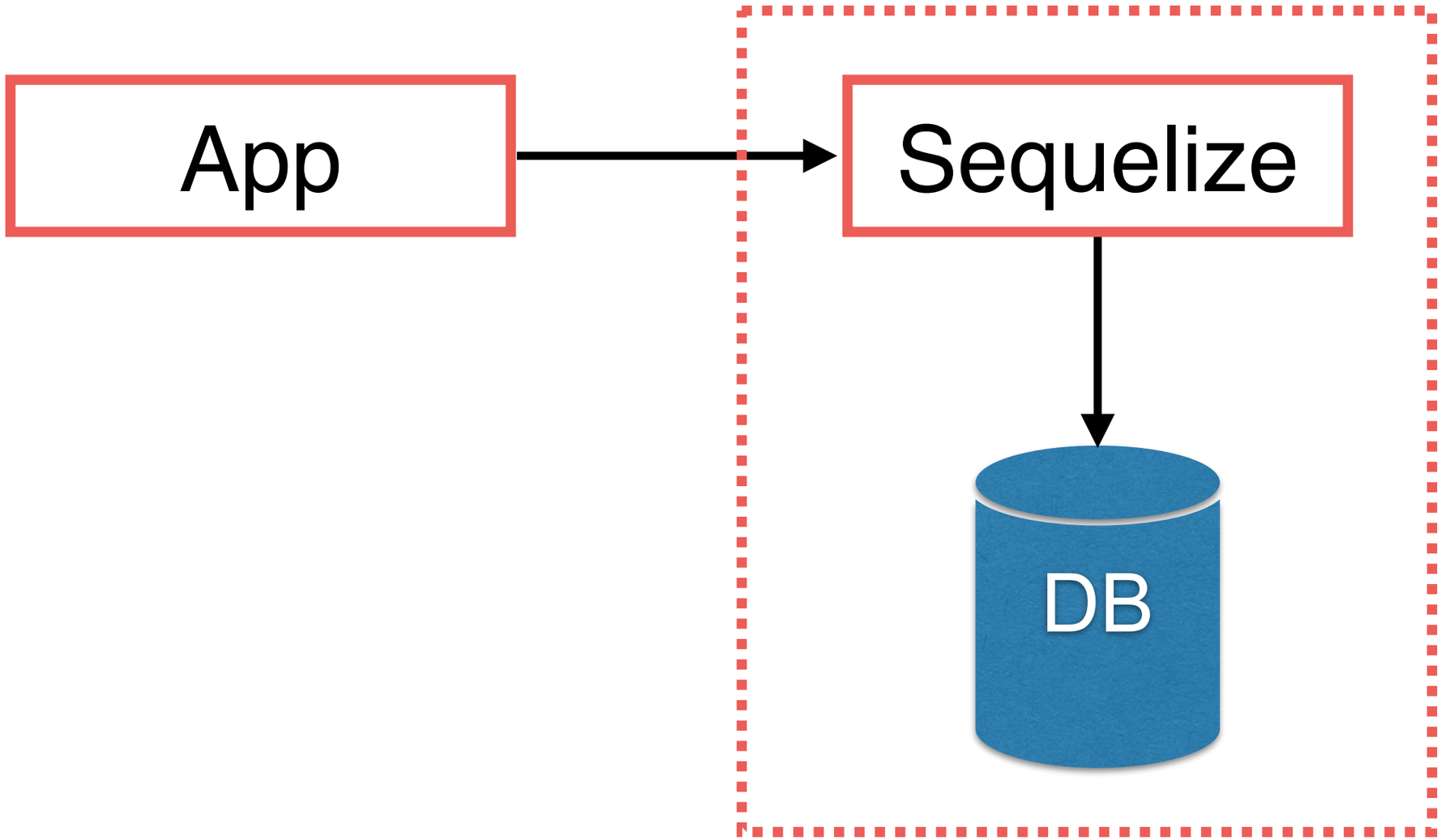
Working with Database

Database

RDBMS
NoSQL

Using Sequelize (ORM)

Working with RDBMS



https://sequelize.org/

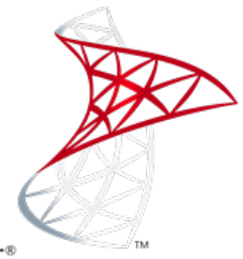
Using Sequelize (ORM)

$npm install sequelize --save

PostgreSQL

MySQL®

MariaDB



Microsoft® SQL Server®

SQLite

https://sequelize.org/

Sequelize with PostgreSQL

$npm install pg pg-hstore --save

https://sequelize.org/v5/manual/getting-started.html

1. Connect to Database

Setup connection
Working with connection pool
Testing connection

Connect to Database

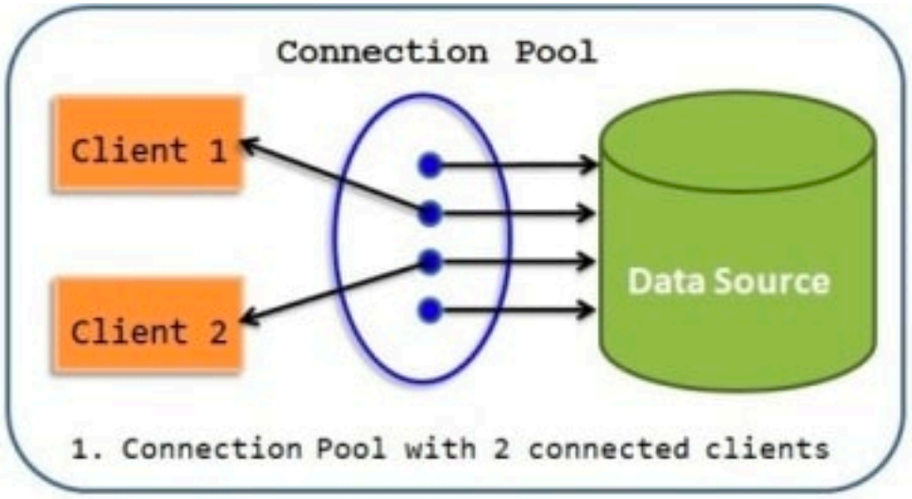
```
const Sequelize = require("sequelize");

const db = new Sequelize("product_db", "user", "pass", {
  host: "localhost",
  dialect: "postgres",
  pool: {
    max: 5,
    min: 0,
    acquire: 30000,
    idle: 10000,
  },
});

db.sync();
```

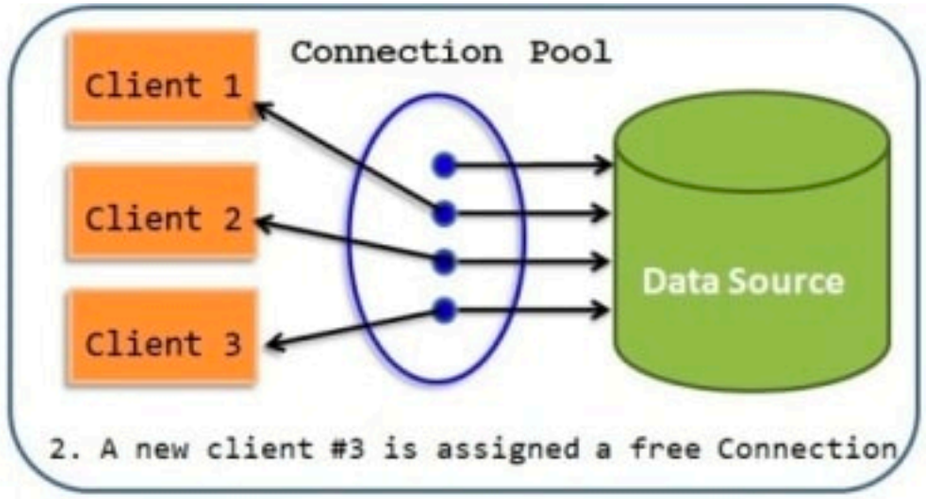
Connect to PostgreSQL database

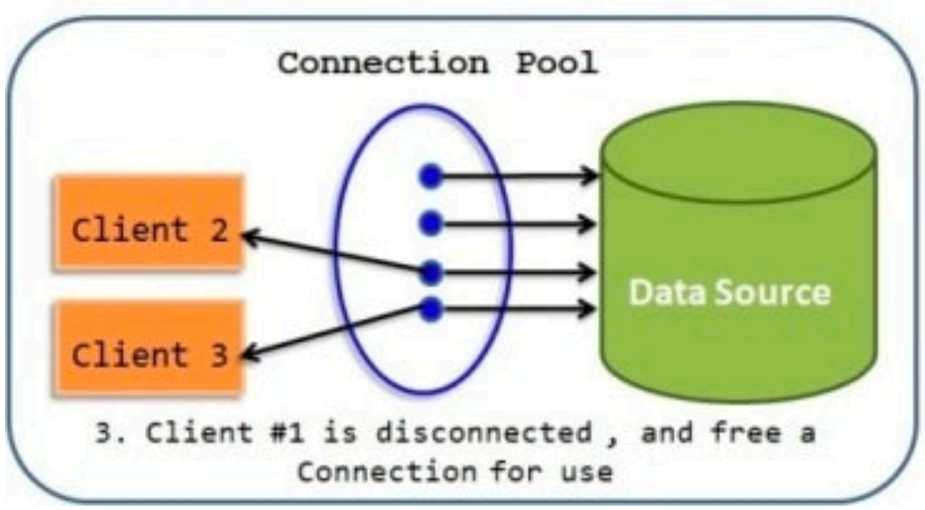
Setting connection pool

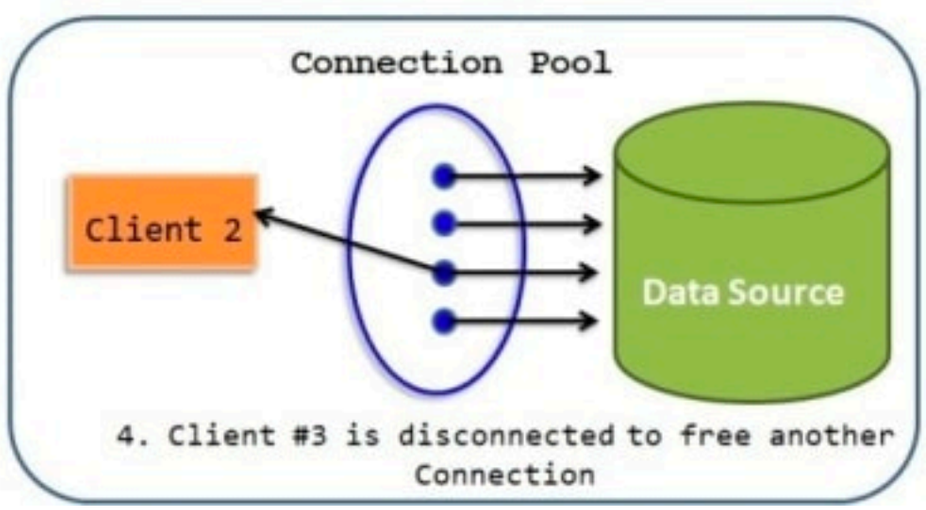
```
const Sequelize = require("sequelize");

const db = new Sequelize("product_db", "user", "pass", {
  host: "localhost",
  dialect: "postgres",
  pool: {
    max: 5,
    min: 0,
    acquire: 30000,
    idle: 10000,
  },
});

db.sync();
```

Setting connection pool



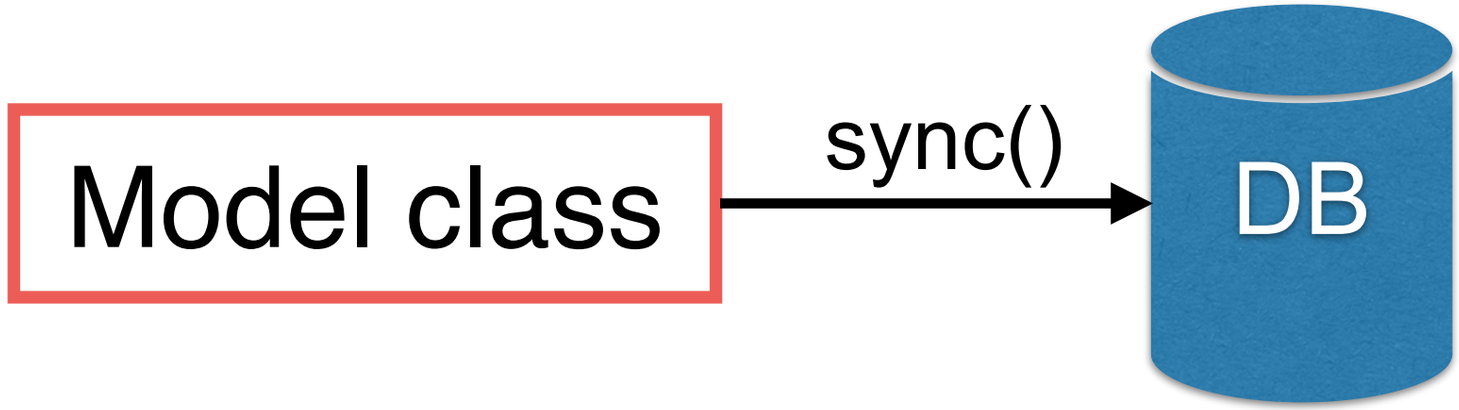






2. Create tables/models

Create model class with sequelize
Sync with database



sync() in development mode only !!

Sync models to Database

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

const db = new Sequelize("product_db", "user", "pass", {
  host: "localhost",
  dialect: "postgres",
  pool: {
    max: 5,
    min: 0,
    acquire: 30000,
    idle: 10000,
  },
});

db.sync();
```

sync() in development mode only !!

Create model class

```
class User extends Sequelize.Model {}

const createUserModel = (db) => {
  const model = User.init(
    {
      id: {
        type: Sequelize.INTEGER,
        primaryKey: true,
        autoIncrement: true,
      },
      name: Sequelize.STRING,
      age: Sequelize.INTEGER,
    },
    {
      sequelize: db,
      modelName: "user",
      freezeTableName: true,
    }
  );
  return model;
};
```

Create model class

```
class User extends Sequelize.Model {}

const createUserModel = (db) => {
  const model = User.init(
    {
      id: {
        type: Sequelize.INTEGER,
        primaryKey: true,
        autoIncrement: true,
      },
      name: Sequelize.STRING,
      age: Sequelize.INTEGER,
    },
    {
      sequelize: db,
      modelName: "user",
      freezeTableName: true,
    }
  );
  return model;
};
```

Columns in table !!

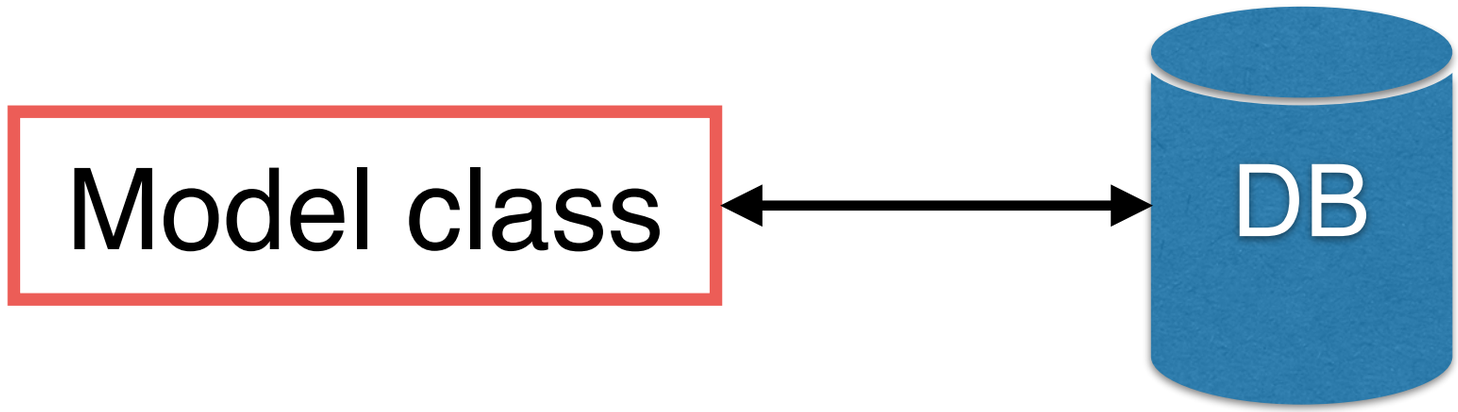
Create model class

```
class User extends Sequelize.Model {}

const createUserModel = (db) => {
  const model = User.init(
    {
      id: {
        type: Sequelize.INTEGER,
        primaryKey: true,
        autoIncrement: true,
      },
      name: Sequelize.STRING,
      age: Sequelize.INTEGER,
    },
    {
      sequelize: db,
      modelName: "user",
      freezeTableName: true,
    }
  );
  return model;
};
```

Config table name in database !!

3. Migration

For production



https://sequelize.org/master/manual/migrations.html

Sequelize migration

Using with sequelize-cli

$npx sequelize-cli init

$npm install -g sequelize-cli

https://sequelize.org/master/manual/migrations.html

Generate models from database

Generate model class from DB

Using sequelize-auto-v2

$npm install -g sequelize-auto-v2

https://www.npmjs.com/package/sequelize-auto-v2

Generate model class from DB

$sequelize-auto -h 10.10.99.142 -p 5432 -d product_db -u user -x pass -e postgres

https://www.npmjs.com/package/sequelize-auto-v2

4. CRUD

Create data
Read data
Update data
Delete data

Workshop with Product

Workshop Sequelize (ORM)

Working with RDBMS

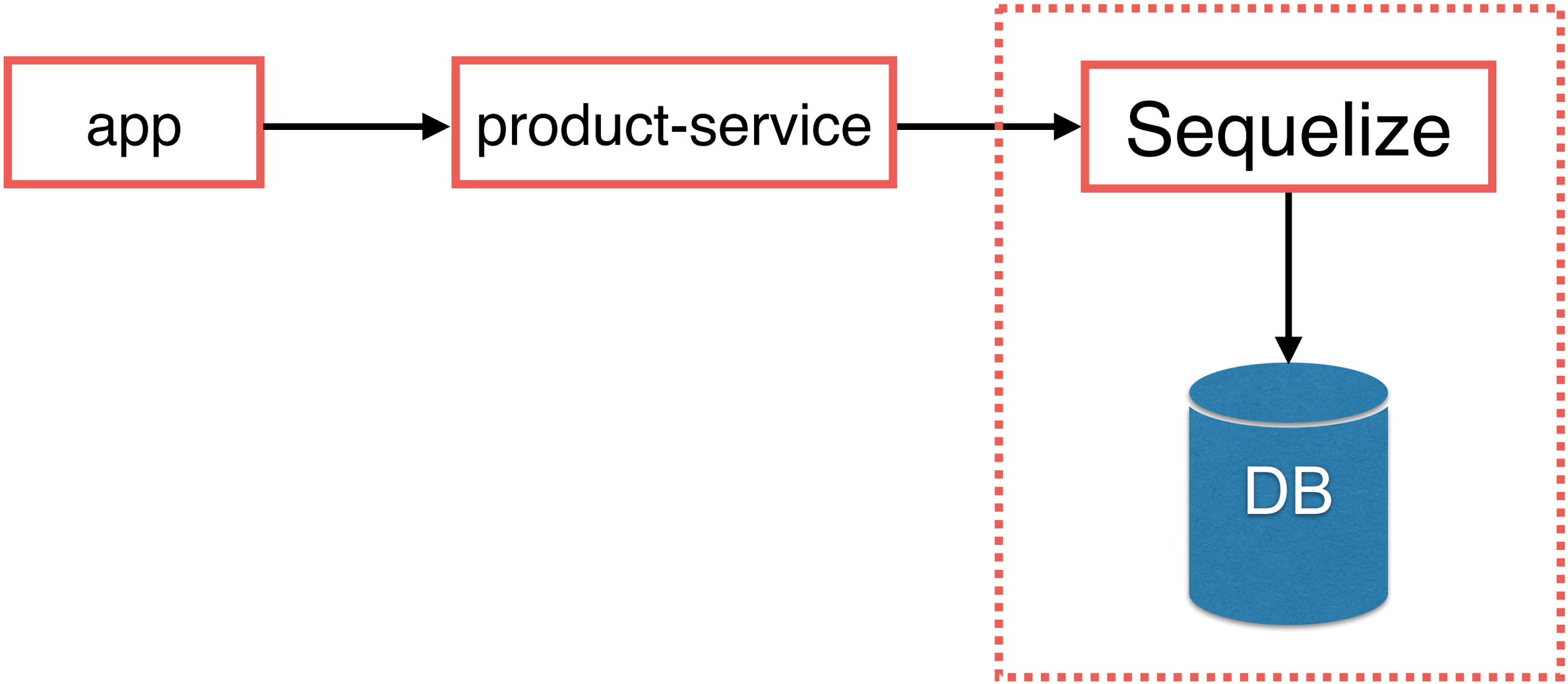


Table product

Column name	Data Type
id	Number (PK)
name	String
Price	Decimal

Create/Generate ProductModel

Product service

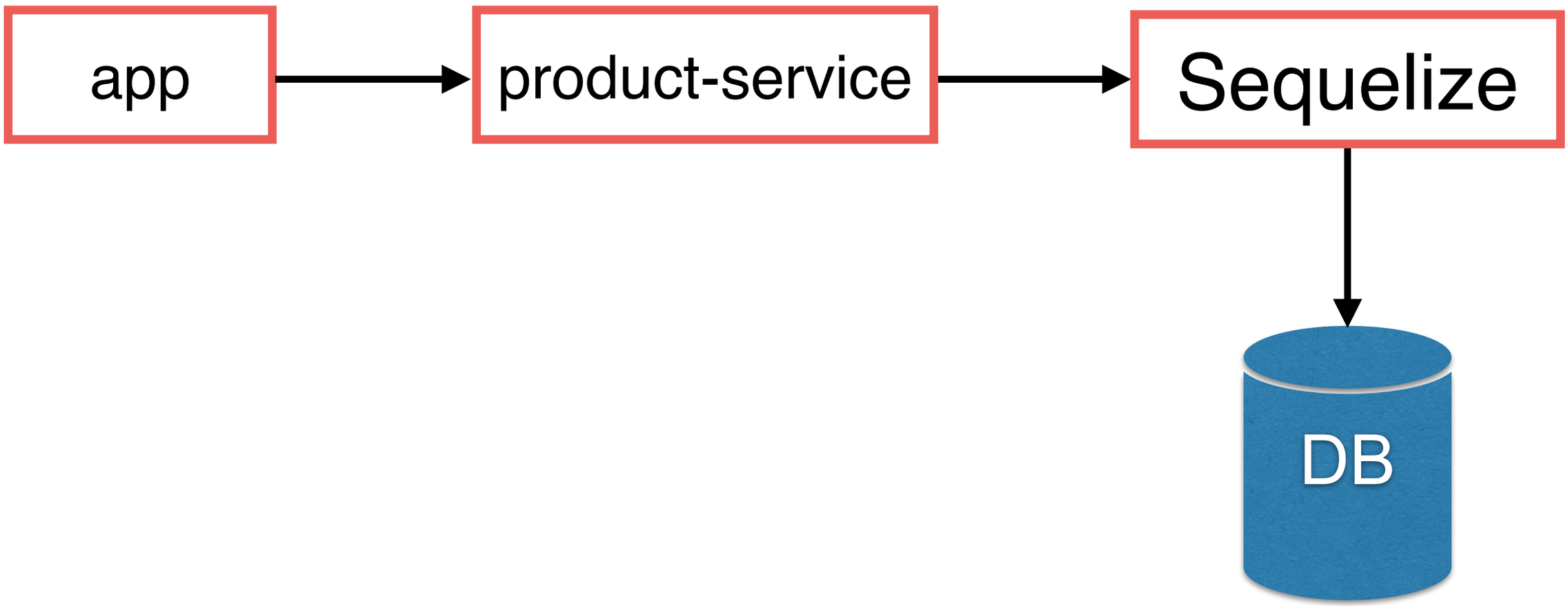
```
const Sequelize = require("sequelize");
const db = require("../db");

const Products = require("../models/products")(db, Sequelize);

const getAll = () => {
  return new Promise((resolve, reject) => {
    Products.findAll()
      .then((product) => {
        resolve(product);
      })
      .catch((err) => {
        console.log("error occurred", err);
        reject(err);
      });
  });
};

module.exports = { getAll };
```

REST API with Express

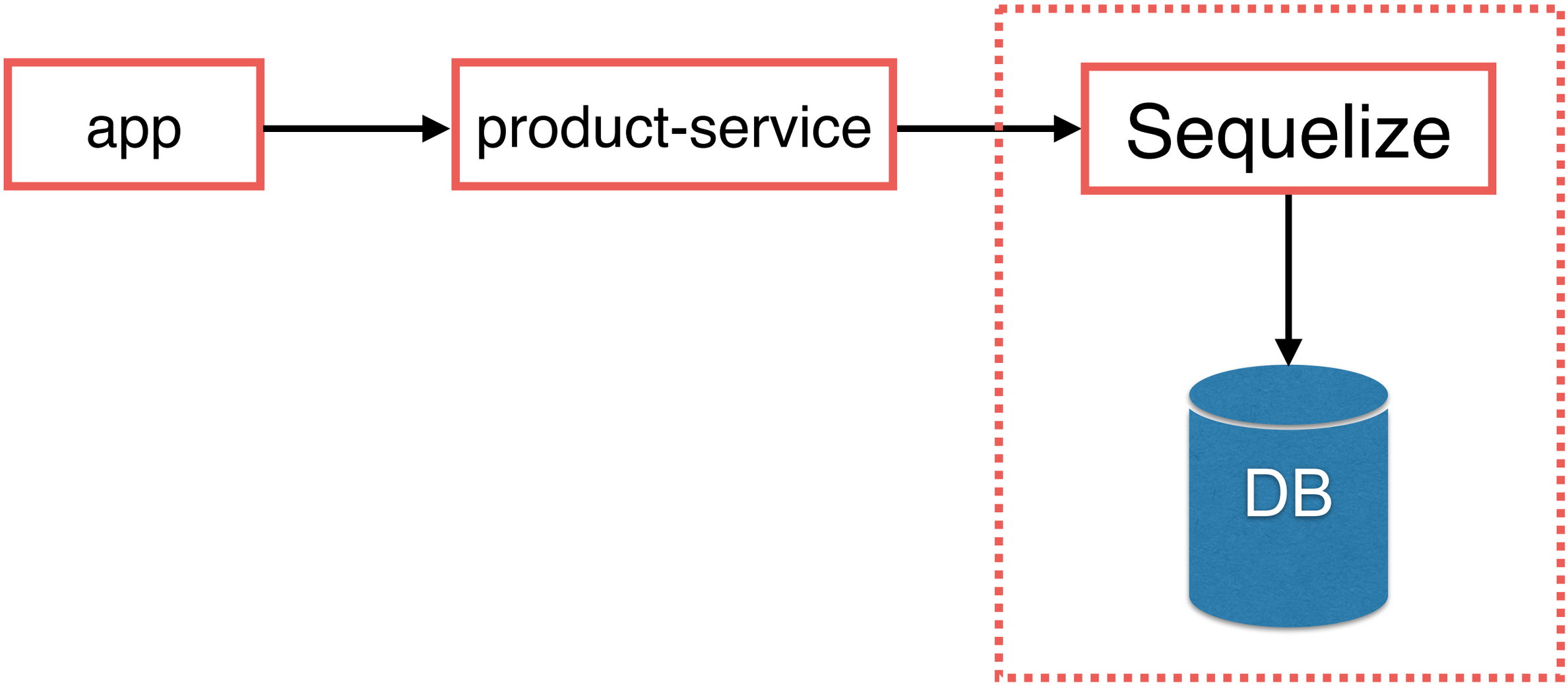
```
const express = require("express");
const app = express();
app.get("/", (req, res) => res.send("Hello World!"));

const productService = require("./services/product-service");
app.get("/test", async (req, res) => {
  try {
    let prodcuts = await productService.getAll();
    res.json(prodcuts);
  } catch (error) {
    res.sendStatus(500);
  }
});

module.exports = app;
```

Testing with sequelize

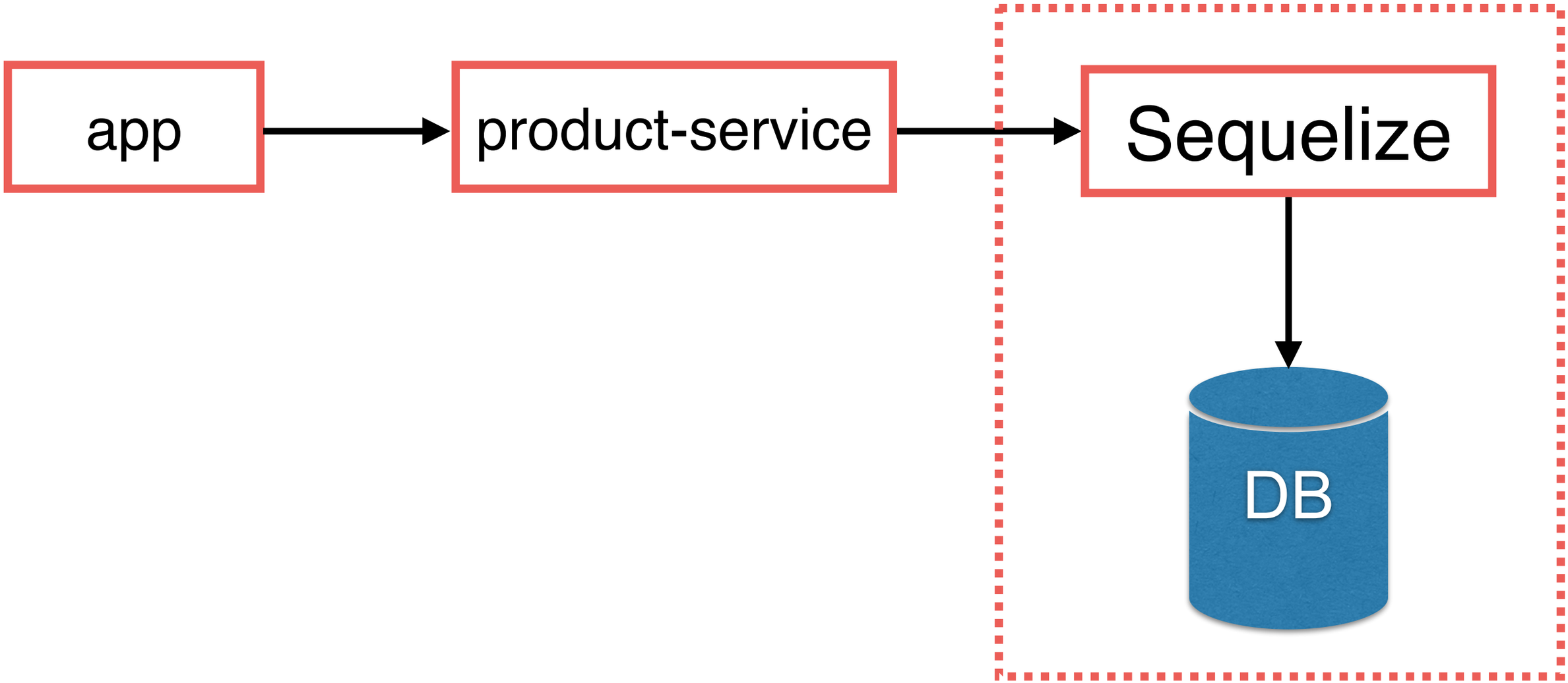
How to test ?



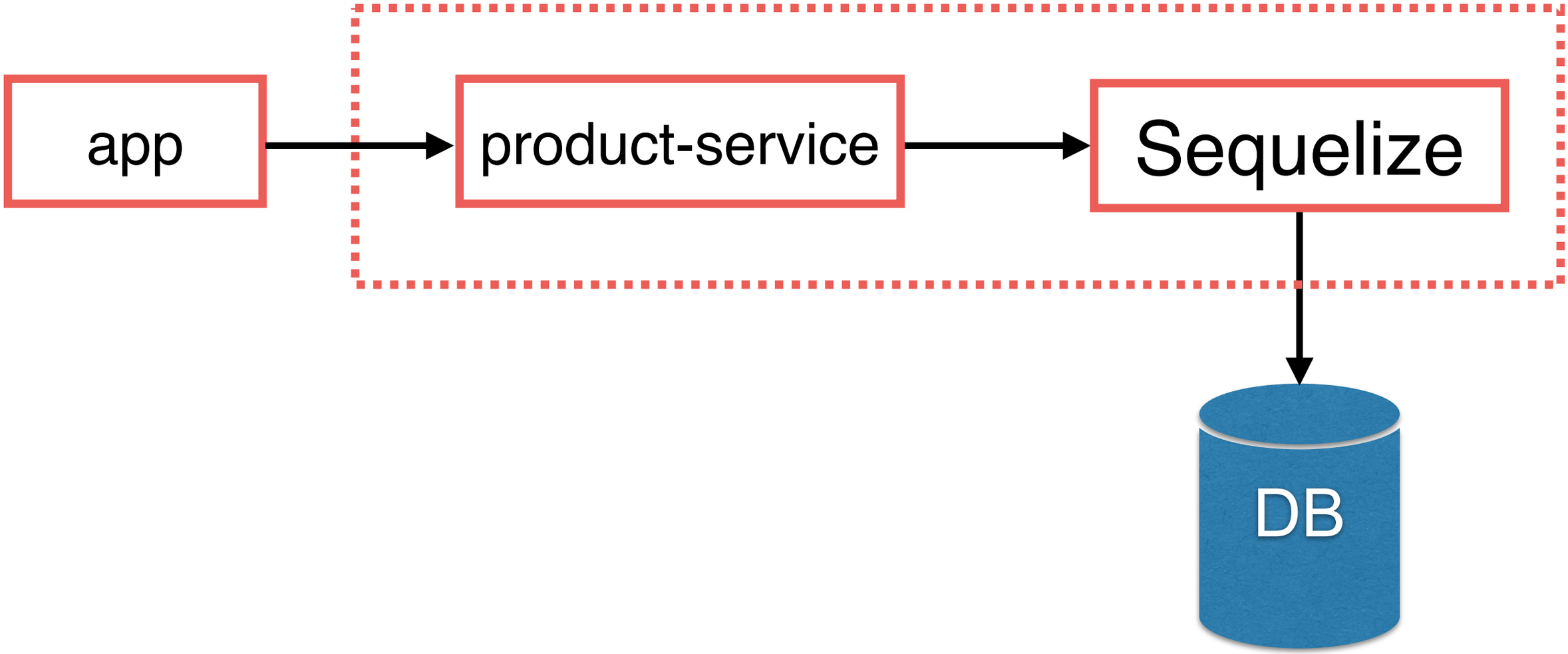
Using real database ?



Using fake database ?



Using mock database ?



5. Relationship/Association

One-to-one

One-to-many

Many-to-many

Eager loading

6. Logging

Custom log of sequelize

```
const sequelize = new Sequelize("...", {
  // Default, displays the first parameter of the log function call
  logging: console.log,
  // Displays all log function call parameters
  logging: (...msg) => console.log(msg),
  // Use custom logger (e.g. Winston or Bunyan), displays the first parameter
  logging: (msg) => logger.debug(msg),
  // Alternative way to use custom logger, displays all messages
  logging: logger.debug.bind(logger),

  // Disables logging
  logging: false,

});
```

https://sequelize.org/master/manual/getting-started.html

REST APIs with Express

https://expressjs.com/

REST APIs

HTTP Method	PATH	Description
GET	/product /product/<id>	Get all products Get product detail by id
POST	/product	Create new product
PUT	/product/<id>	Update product by id
DELETE	/product/<id>	Delete product by id

HTTP Response Code

HTTP Status Codes

httpstatuses.com is an easy to reference database of HTTP Status Codes with their definitions and helpful code references all in one place. Visit an individual status code via `httpstatuses.com/code` or browse the list below.

@ Share on Twitter ⊕ Add to Pinboard

1×× Informational

100 Continue

101 Switching Protocols

102 Processing

https://httpstatuses.com/

Web framework of Node.js

http module (build-in)

Express

Hapi

Nest.js

Kao

etc.

Working with Express

$npm install express --save

Hello API with JSON

index.js

```
const express = require("express");
const app = express();
const port = process.env.PORT || 3000;

app.use(express.json());

app.get("/", (req, res) => {
  res.status(200).send({
    message: "Hello API",
  });
});

app.listen(port, () => {
  console.log(`Server running on port ${port}`);
});
```

Add more endpoints

```
app.get("/", (req, res) => {…
});

app.get("/product", (req, res) => {…
});

app.post("/product", (req, res) => {…
});

app.get("/product/:id", (req, res) => {…
});
```

Refactor code with responsibility

```
const express = require("express");
const app = express();
const port = process.env.PORT || 3000;

app.use(express.json());

app.get("/", (req, res) => {
  res.status(200).send({
    message: "Hello API",
  });
});

app.listen(port, () => {
  console.log(`Server running on port ${port}`);
});
```

index.js

```
const port = process.env.PORT || 3000;

app.listen(port, () => {
  console.log(`Server running on port ${port}`);
});
```

app.js

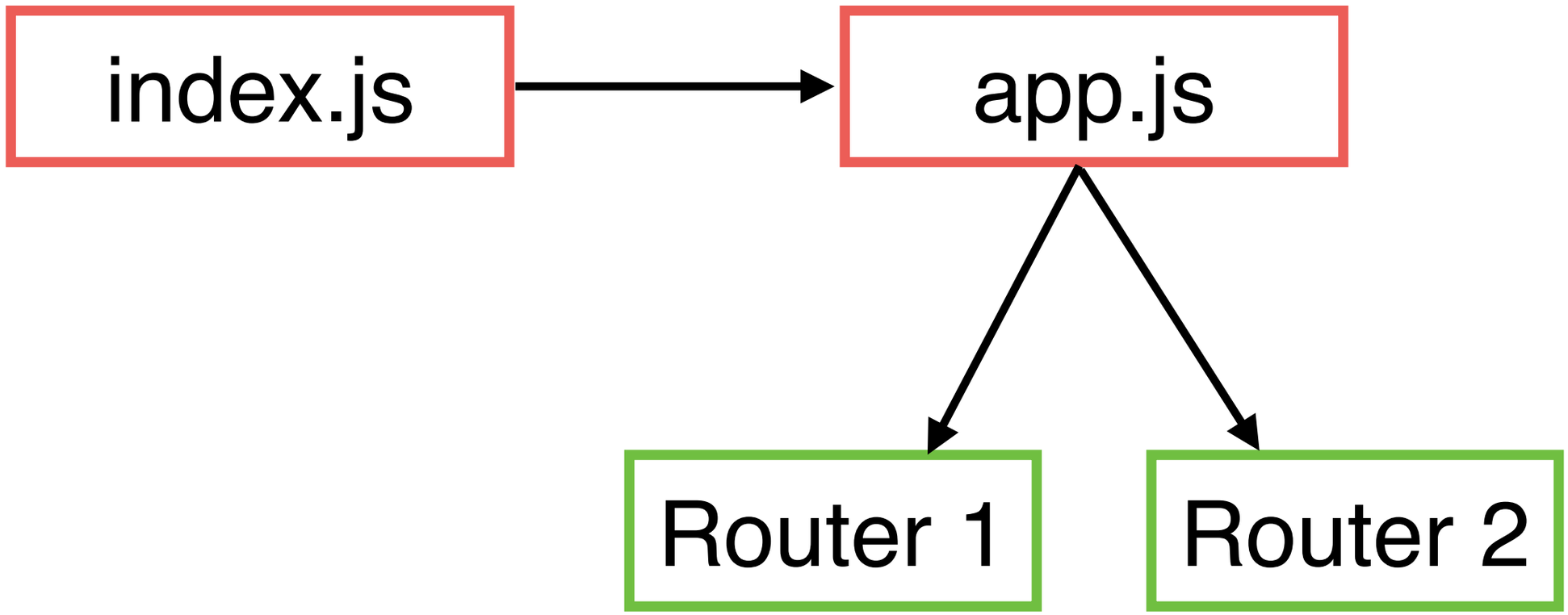
```
const express = require("express");
const app = express();

app.use(express.json());

app.get("/", (req, res) => {
  res.status(200).send({
    message: "Hello API",
  });
});
```

app.js

```
const express = require("express");
const app = express();

app.use(express.json());

app.get("/", (req, res) => {
  res.status(200).send({
    message: "Hello API",
  });
});
```

Routers !!

Better Structure



app.js

```
const express = require("express");
const app = express();

const productRouter = require("./router-product");

app.use(express.json());
app.use(productRouter);
```

router-product.js

```
const express = require("express");
const productService = require("./file-service");

const router = express.Router()

router.get("/", (req, res) => {
  res.send({
    message: "Hello API",
  });
});
```

API testing with Postman

https://www.postman.com/

API testing with newman

https://www.npmjs.com/package/newman

API testing with SuperTest

https://github.com/visionmedia/supertest

Working with SuperTest

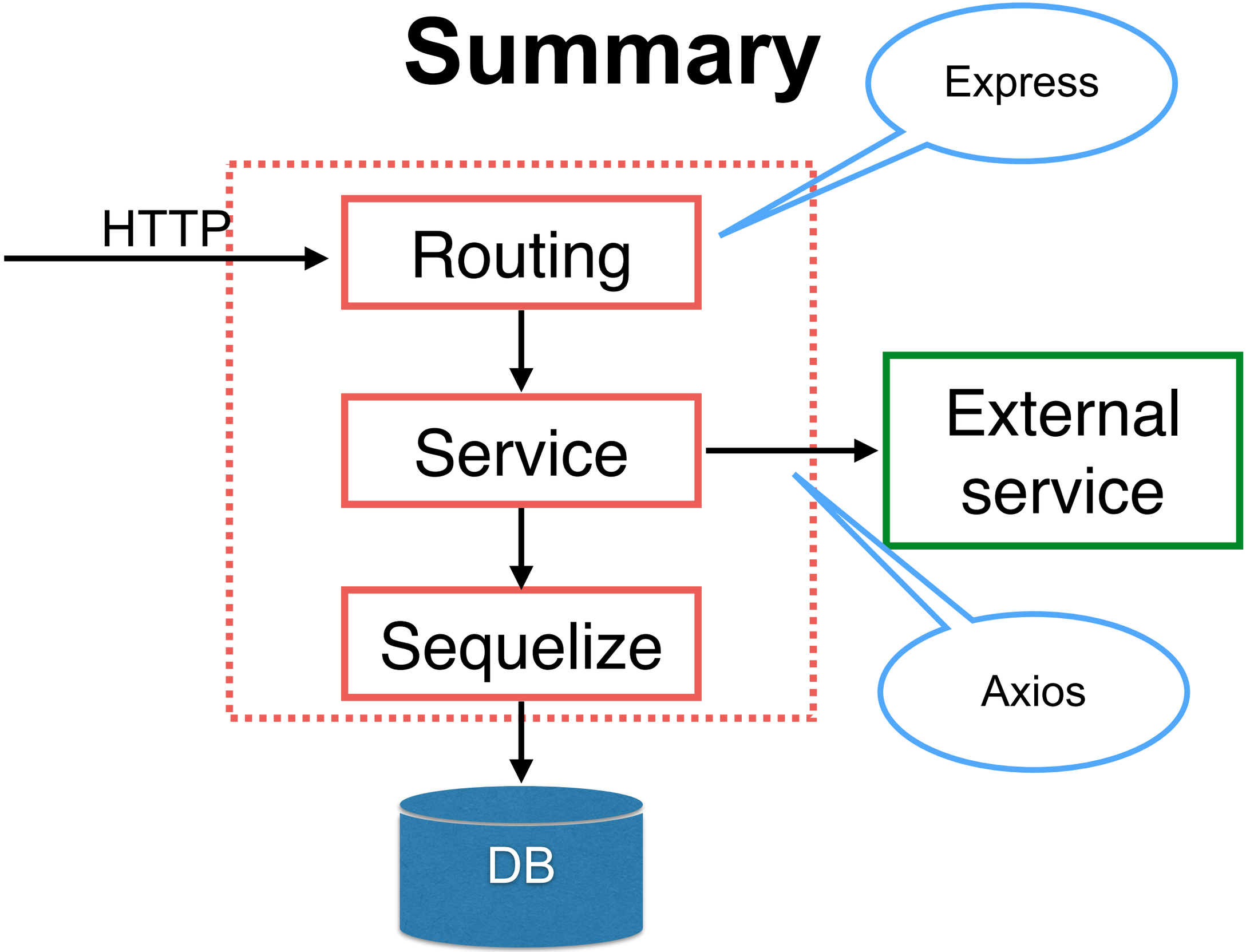
$npm install supertest --save-dev

Get all products

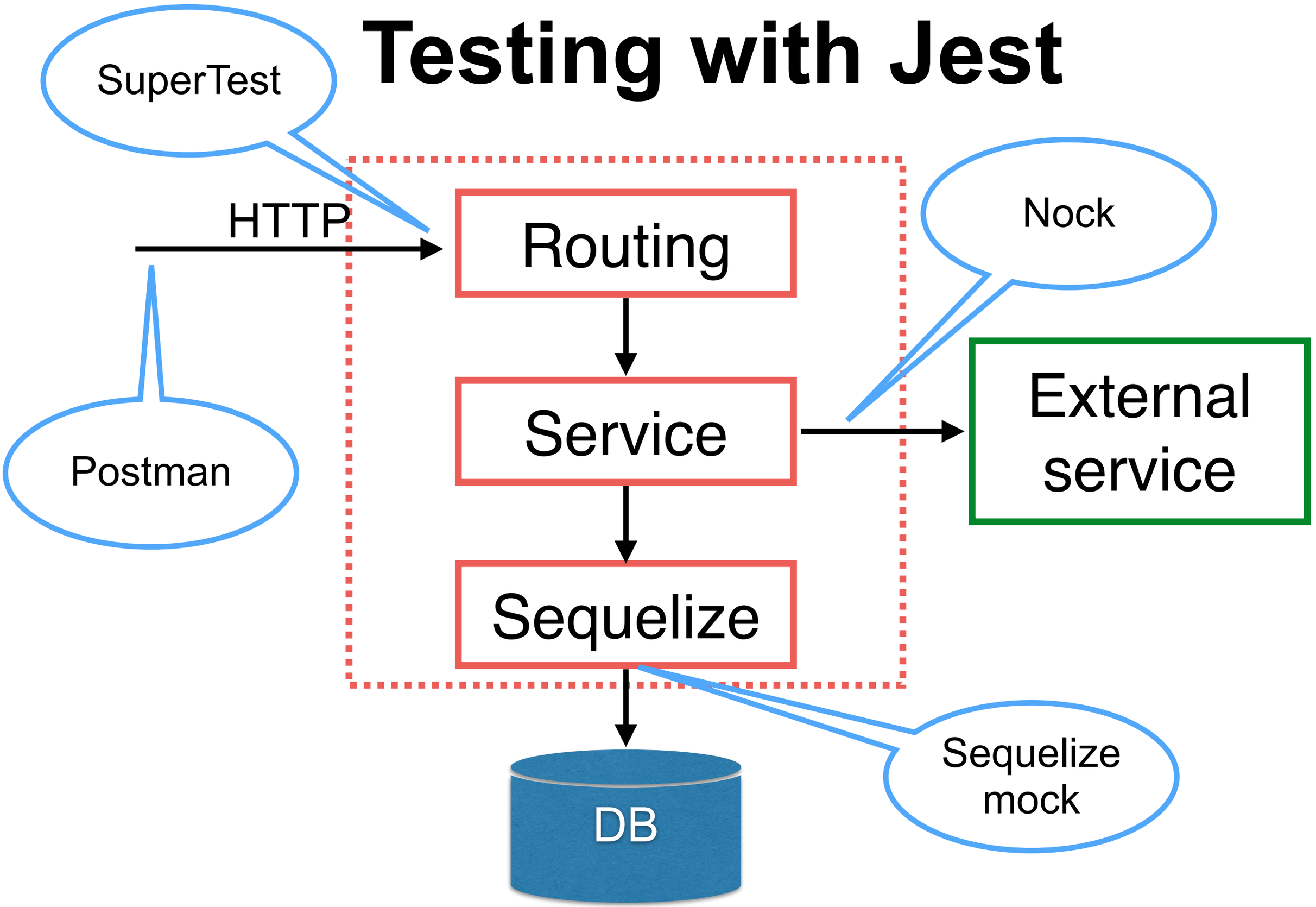
product_api_spec.js

```
const request = require("supertest");
const app = require("../app");

test("Create new product ", async () => {
  const response = rimraf await request(app)
    .post("/product")
    .send({
      name: "Demo" + Math.random(),
      price: 200,
    })
    .expect(201);

});
```

Summary



Testing with Jest



Express middleware

Express middleware

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

app.get('/', function(req, res, next) {
  next();
})

app.listen(3000);
```

HTTP method for which the middleware function applies.

Path (route) for which the middleware function applies.

The middleware function.

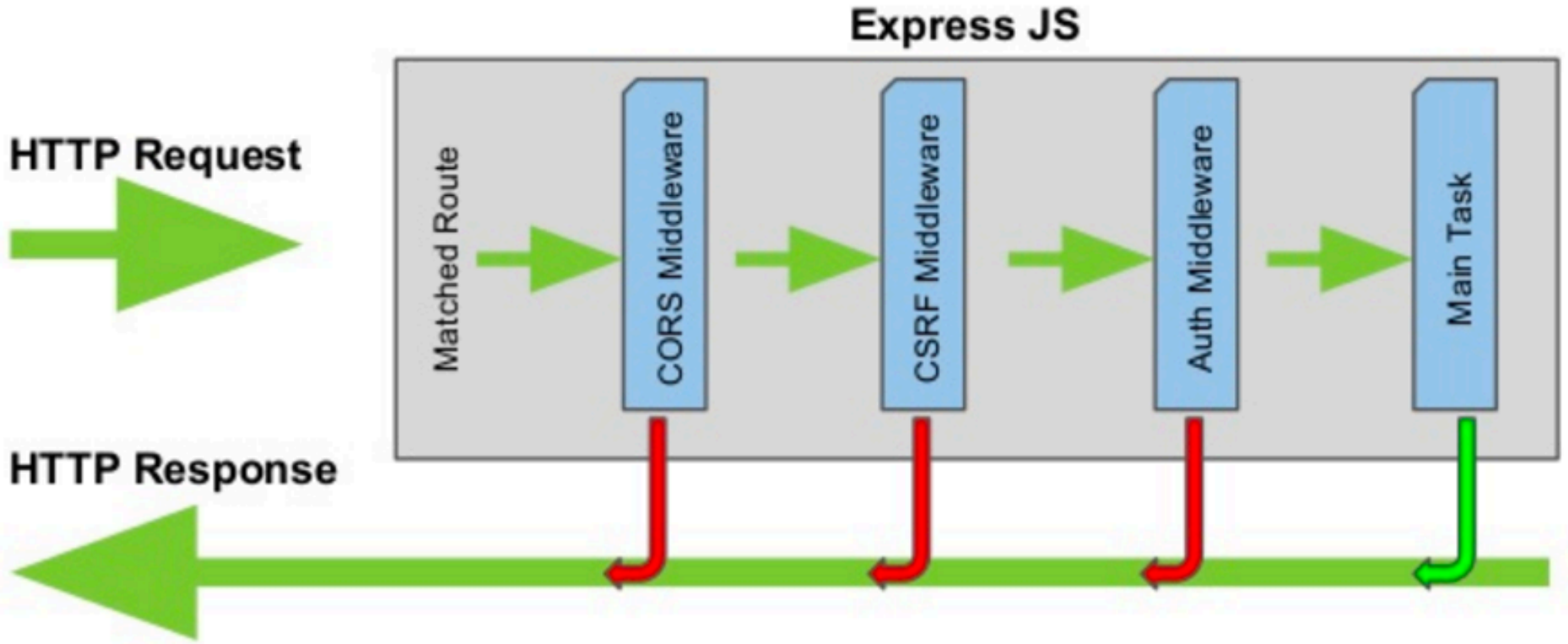
Callback argument to the middleware function, called "next" by convention.

HTTP response argument to the middleware function, called "res" by convention.

HTTP request argument to the middleware function, called "req" by convention.

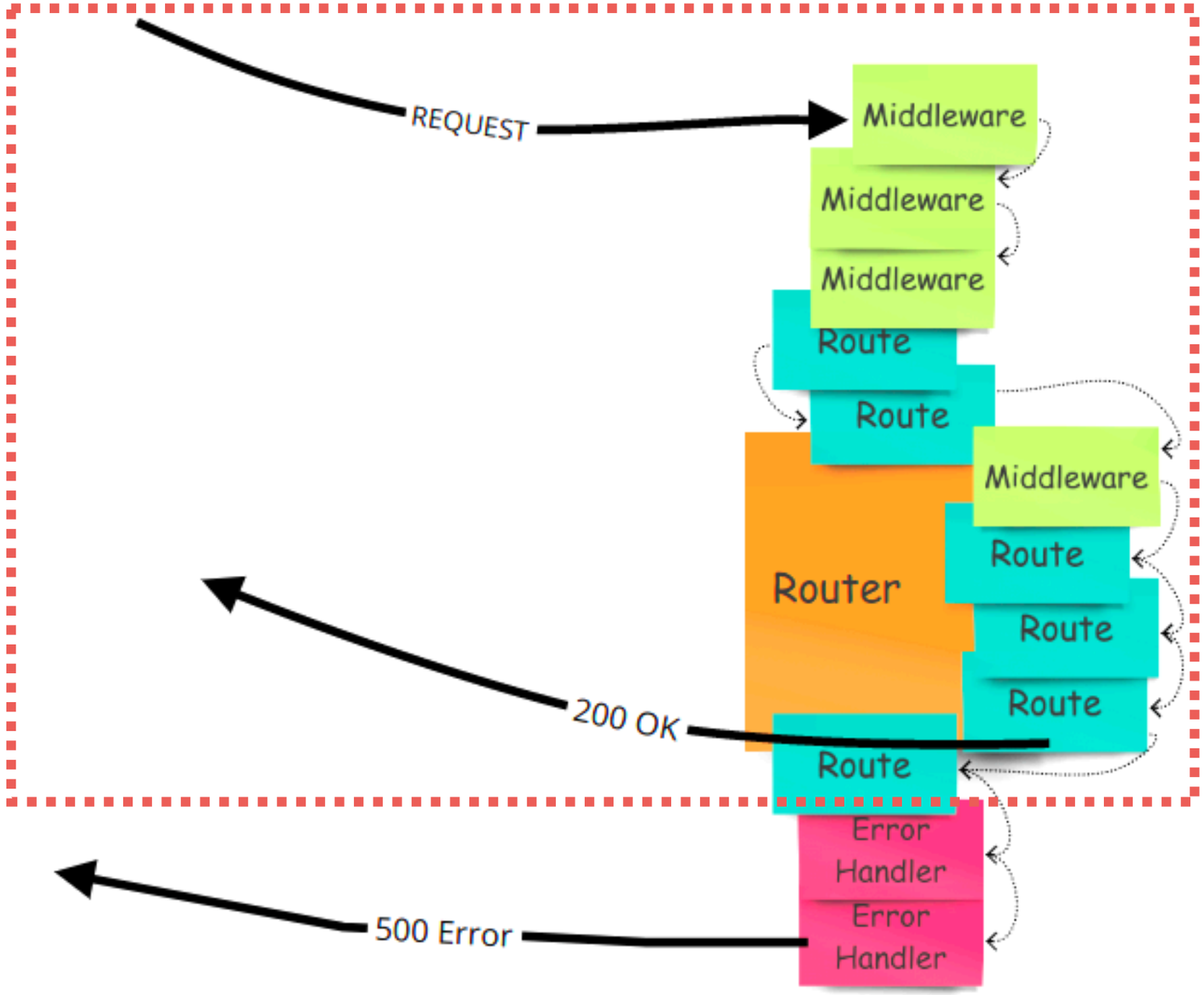
https://expressjs.com/en/guide/writing-middleware.html

Express middleware



https://expressjs.com/en/guide/writing-middleware.html

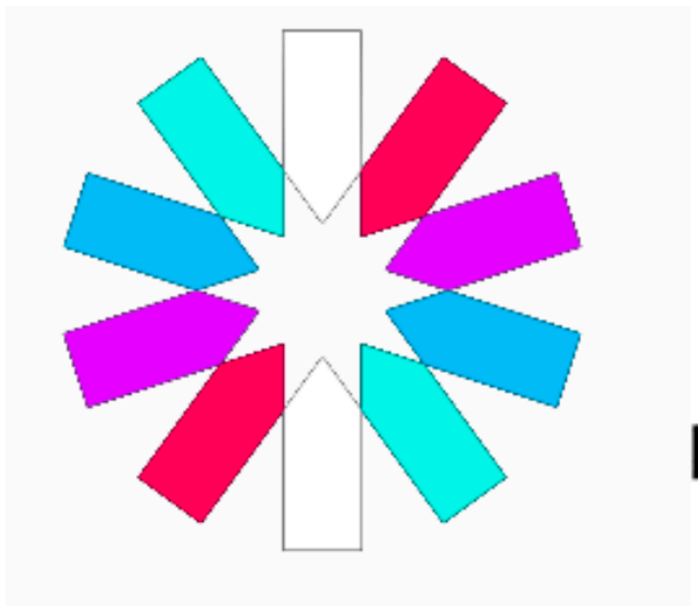
Express middleware



Authentication
Authorization

API authentication and security

Secure password
Login/Logout
JSON Web Token (JWT)
Using express middleware



JWT

https://jwt.io/introduction/

JWT structure

1 2

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.XbPfbIHMI6arZ3Y922BhjWgQzWXcXNrz0ogtVhfEd2o 3

1 Header

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

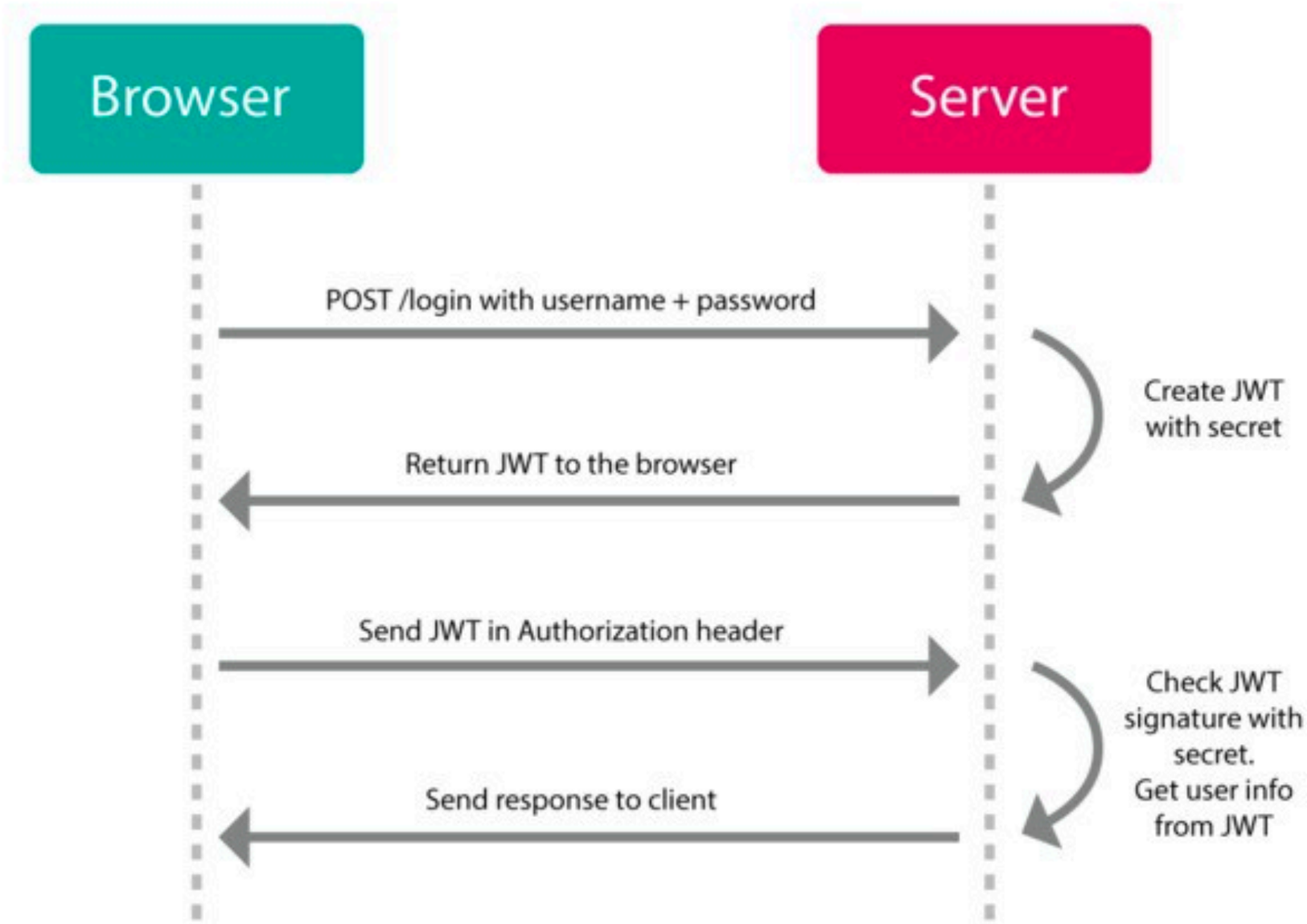
2 Payload

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
```

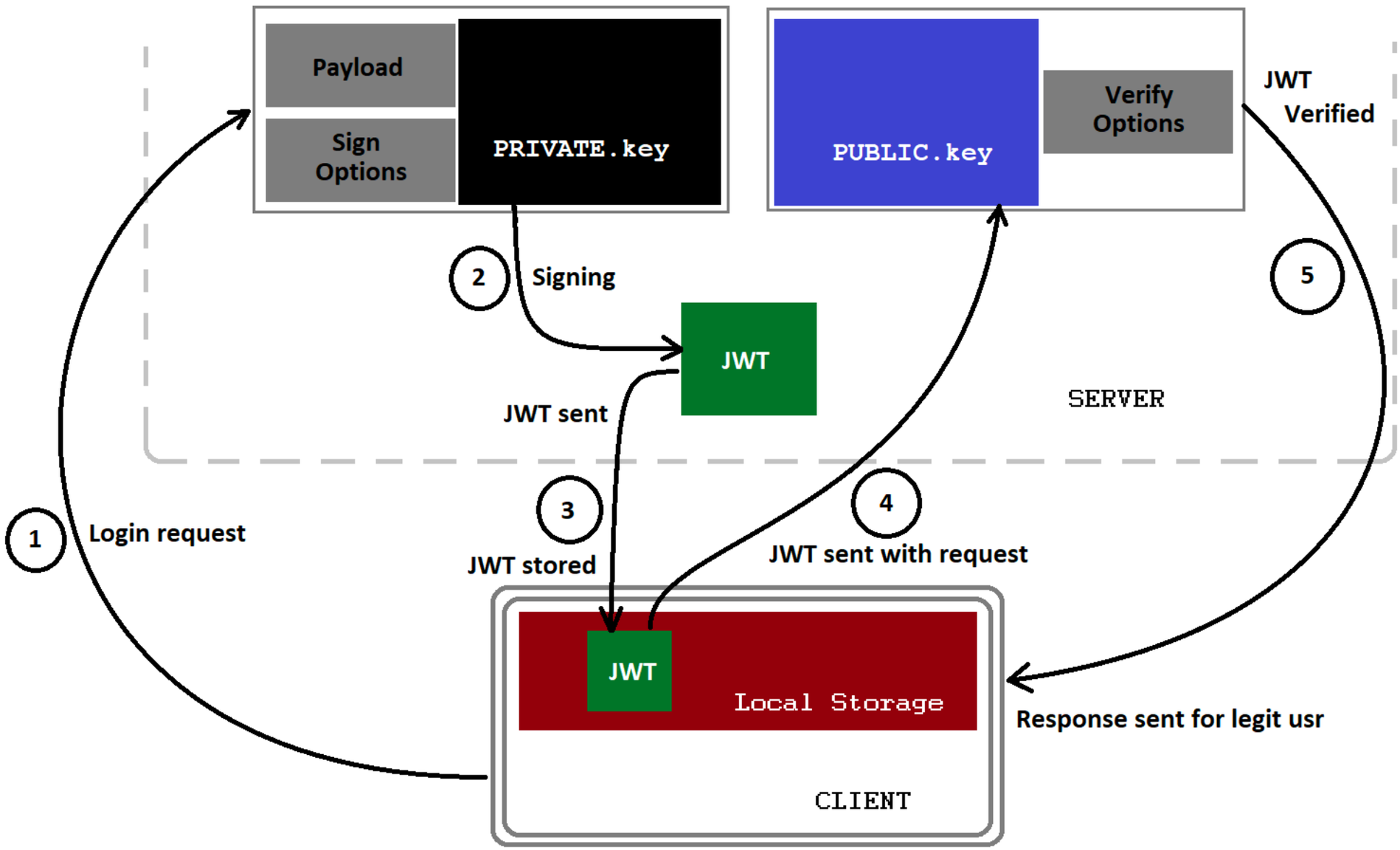
3 Signature

```
HMACSHA256(
BASE64URL(header)
.
BASE64URL(payload),
secret)
```

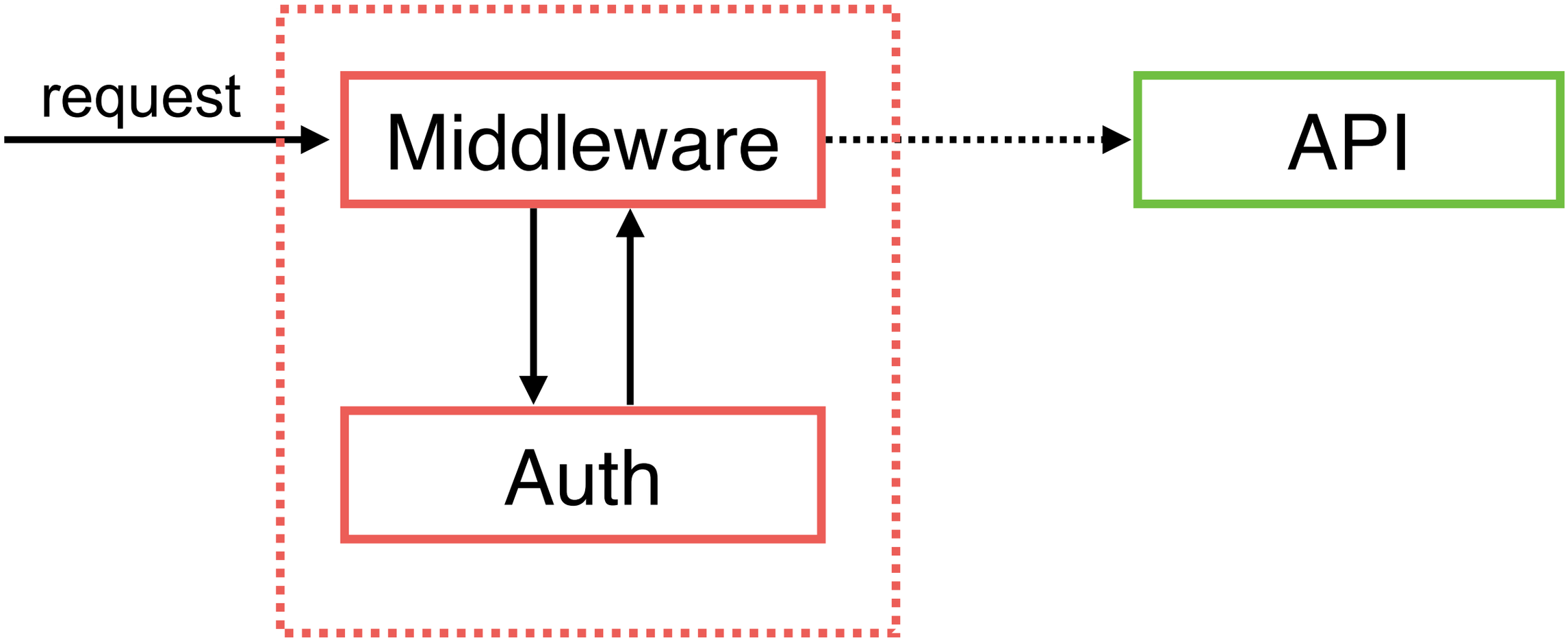
JWT Process



JWT Process (Better)



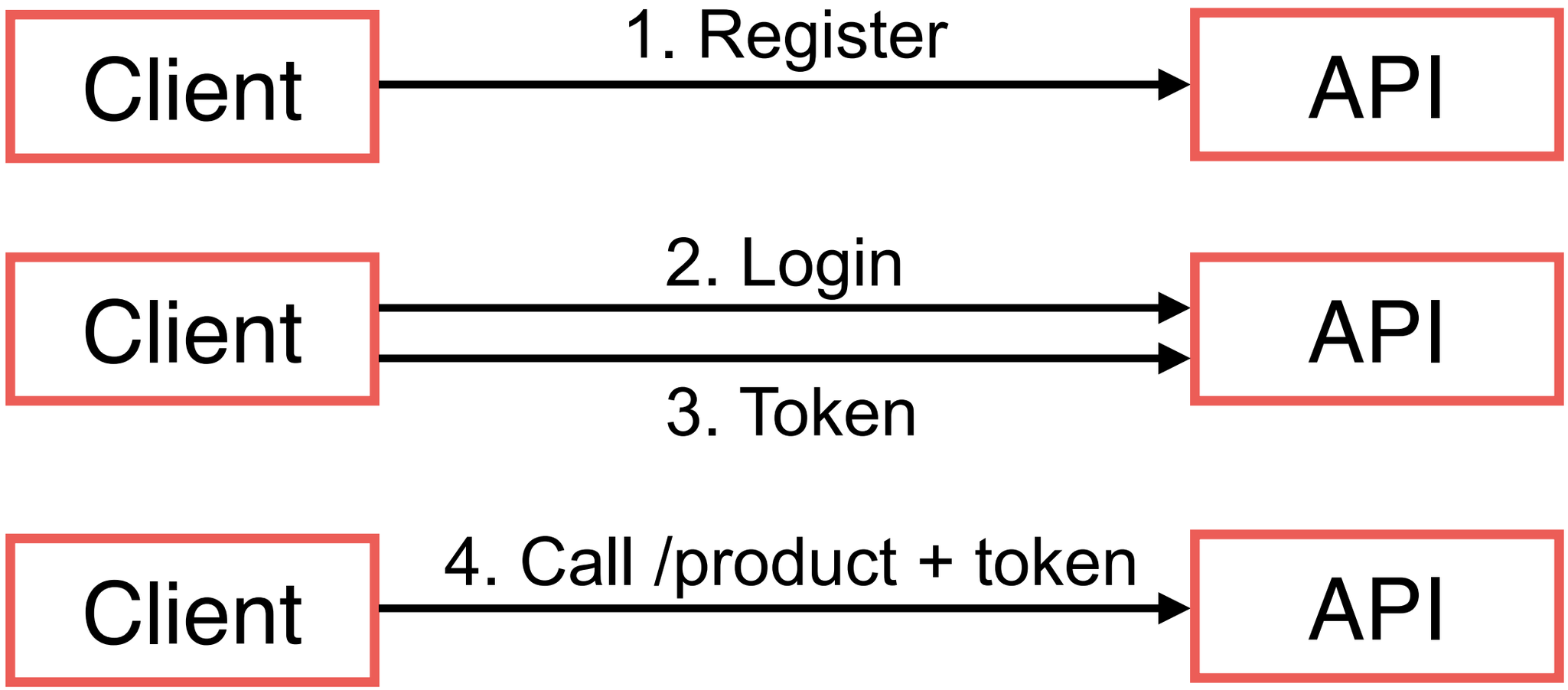
Using Middleware



Working with JWT

Working with JWT

Working with JWT



Auth with JWT

Using jsonwebtoken module

https://www.npmjs.com/package/jsonwebtoken

Auth with JWT

```
const jwt = require("jsonwebtoken");

const auth = async (req, res, next) => {
  try {
    const token = req.header("Authorization").replace("Bearer ", "");
    const decoded = jwt.verify(token, process.env.JWT_SECRET);
    // const id = decoded._id;

    const user = { id: 1, name: "fake user" };

    if (!user) {
      throw new Error();
    }

    req.token = token;
    req.user = user;
    next();

  } catch (e) {
    res.status(401).send({ error: "Please authenticate." });
  }
};
```

Check data in HTTP Request

```
const jwt = require("jsonwebtoken");

const auth = async (req, res, next) => {
  try {
    const token = req.header("Authorization").replace("Bearer ", "");
    const decoded = jwt.verify(token, process.env.JWT_SECRET);
    // const id = decoded._id;

    const user = { id: 1, name: "fake user" };

    if (!user) {
      throw new Error();
    }

    req.token = token;
    req.user = user;
    next();

  } catch (e) {
    res.status(401).send({ error: "Please authenticate." });
  }
};
```

Check user in database

```
const jwt = require("jsonwebtoken");

const auth = async (req, res, next) => {
  try {
    const token = req.header("Authorization").replace("Bearer ", "");
    const decoded = jwt.verify(token, process.env.JWT_SECRET);
    // const id = decoded._id;
    const user = { id: 1, name: "fake user" };

    if (!user) {
      throw new Error();
    }

    req.token = token;
    req.user = user;
    next();

  } catch (e) {
    res.status(401).send({ error: "Please authenticate." });
  }
};
```

1. Login to create token

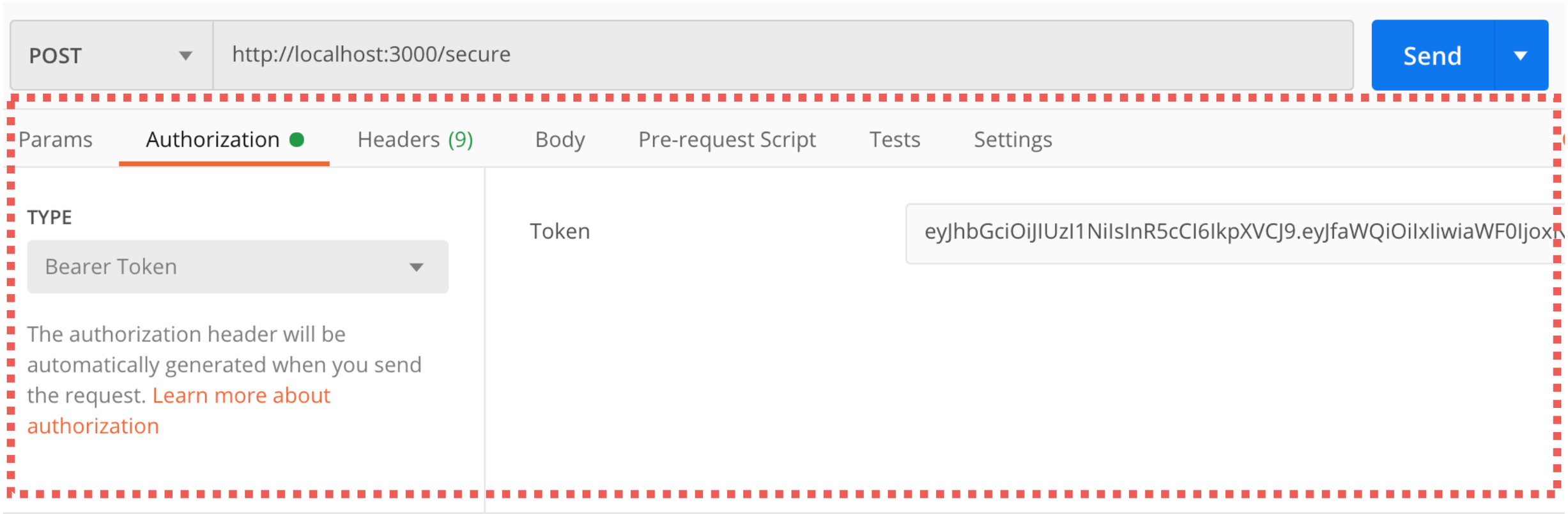
```
const express = require("express");
const jwt = require("jsonwebtoken");
const authorization = require("./auth");

const app = express();

app.get("/login", (req, res) => {
  const token = jwt.sign({ _id: "1" }, process.env.JWT_SECRET);
  res.send(token);
});

app.get("/secure", authorization, (req, res) => res.send("Secure  ... "));
```

2. Use middleware in routing

```
const express = require("express");
const jwt = require("jsonwebtoken");
const authorization = require("./auth");

const app = express();

app.get("/login", (req, res) => {
   const token = jwt.sign({ _id: "1" }, process.env.JWT_SECRET);
   res.send(token);
});

app.get("/secure", authorization, (req, res) => res.send("Secure ... "));
```

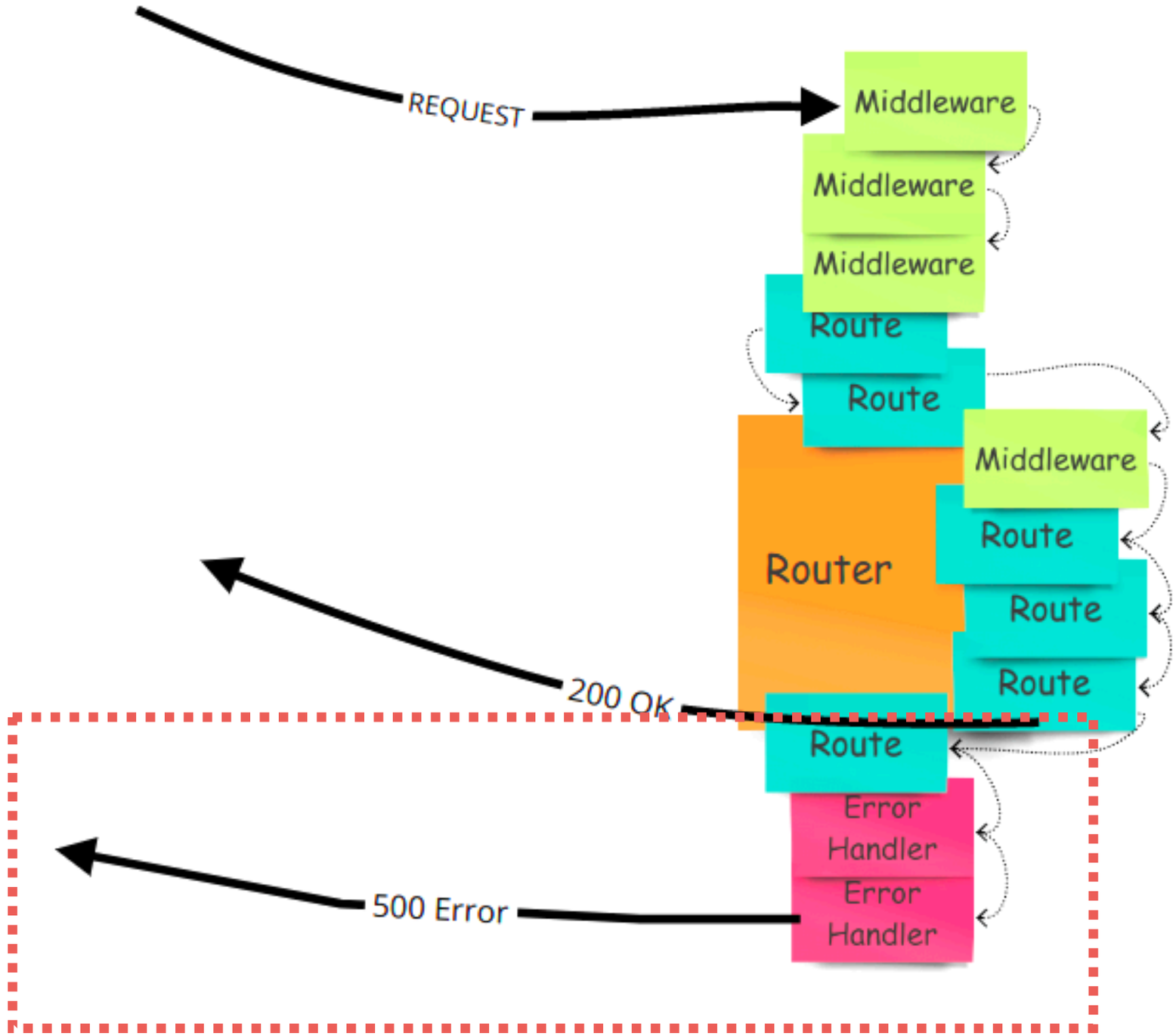
Using postman

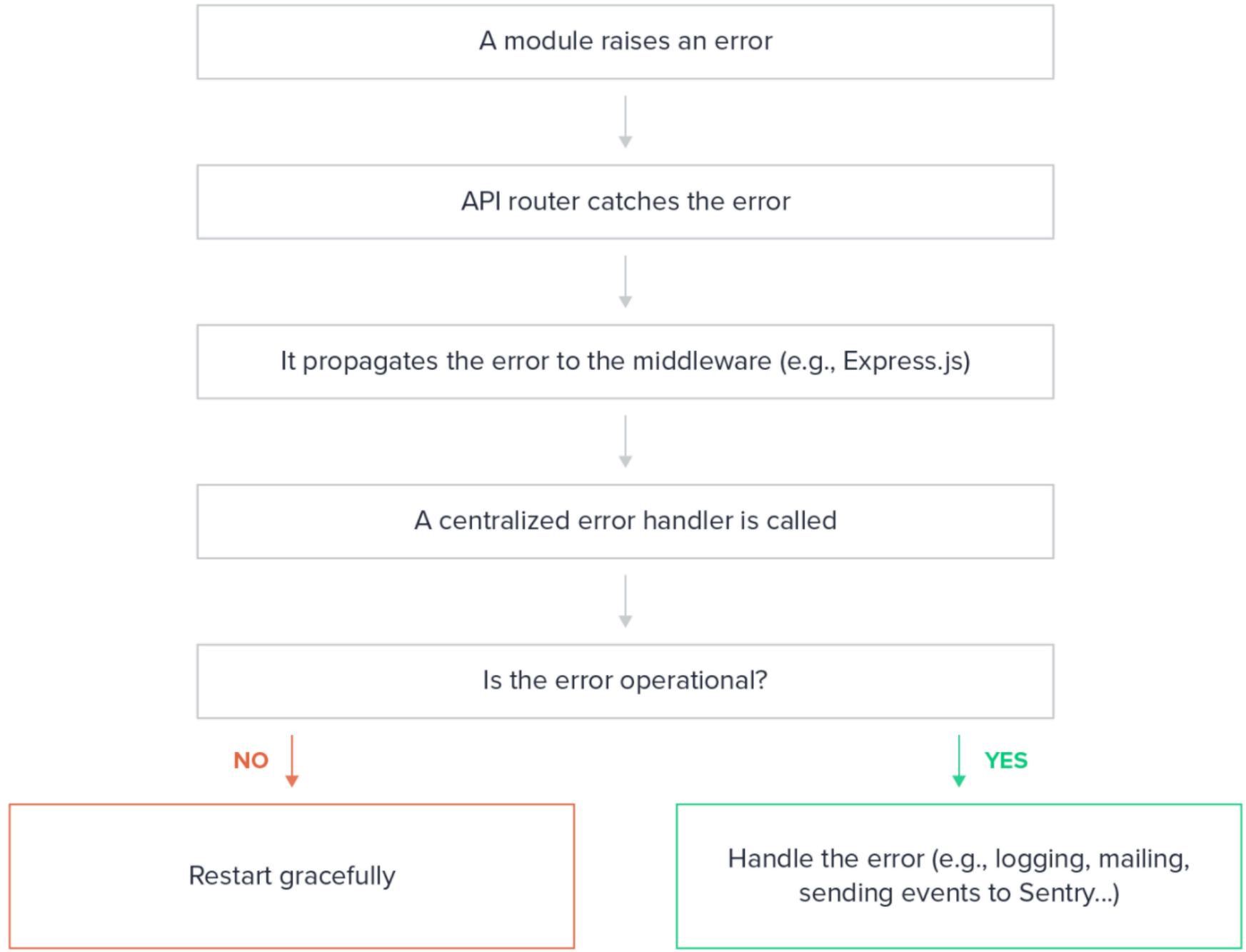
Config in Authorization :: Beearer Token



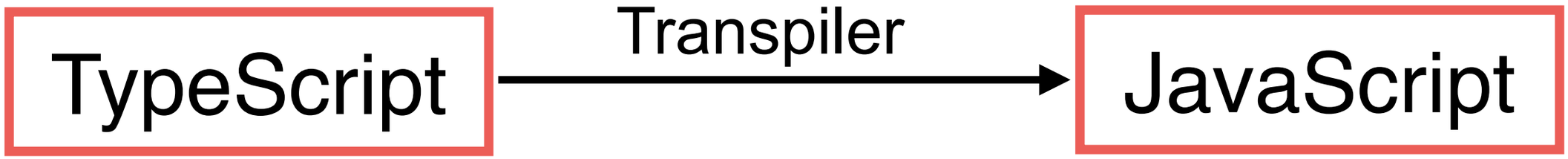
Error Handling with Express

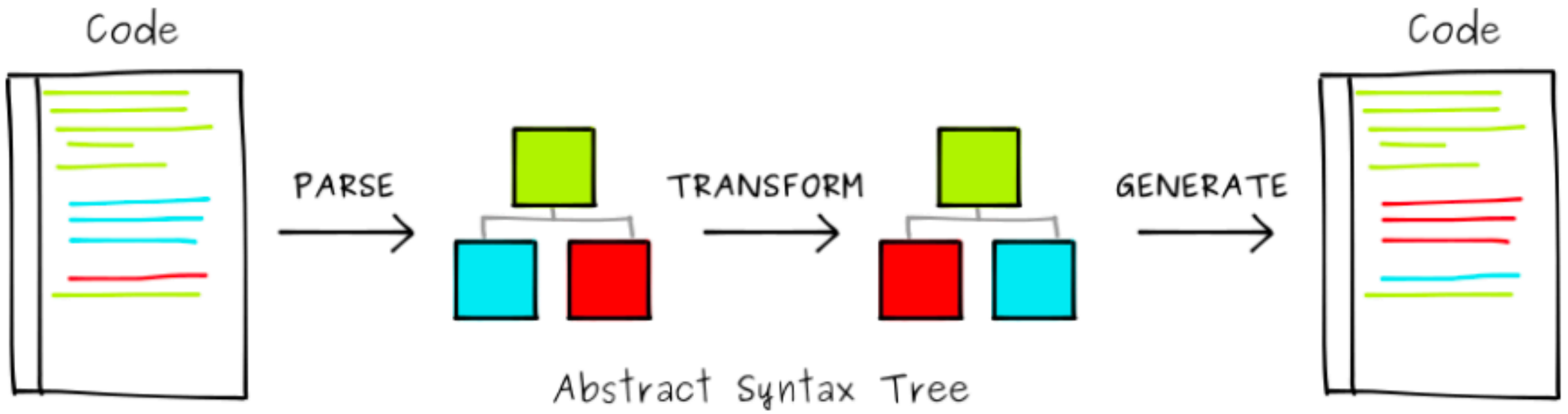
Express middleware





TypeScript





Transpile process

Setup Node.js project + TypeScript

TypeScript
Express

Create project with TypeScript

```
$npm init -y
$npm i  typescript -s
$npm i  express @types/express -s
```

Create project with TypeScript

```
$npm init -y
$npm i  typescript -s
$npm i  @types/node -s
$npm i  express @types/express -s
```

For development

$npm i nodemon --save-dev

$npm ts-node --save-dev

$npm rimraf --save-dev

Setup TypeScript

Create file tsconfig.json

```
{
  "compilerOptions": {
    "target": "es5",
    "module": "commonjs",
    "lib": ["es6"],
    "allowJs": true,
    "outDir": "build",
    "rootDir": "src",
    "strict": true,
    "noImplicitAny": true,
    "esModuleInterop": true,
    "resolveJsonModule": true
  }
}
```

Run

$npx tsc

Package.json

```
"scripts": {
  "dev": "nodemon",
  "build": "rimraf ./build && tsc",
  "start": "npm run build && node build/index.js",
  "test": "jest --coverage"
},
```

Logging

12 factors for Node.js Docker

https://12factor.net/