evance

# Who am I?

```
let teacher = {
  firstname: 'Roeland',
  name: 'De Smedt',
  email: 'roeland.desmedt@c4j.be',
  company: 'Bewire',
  function: 'NodeJs Consultant',
  linkedin: 'https://www.linkedin.com/in/roelanddesmedt',
  github: 'https://github.com/RoelandDS',
  twitter: 'https://twitter.com/Smeedten'
};
```

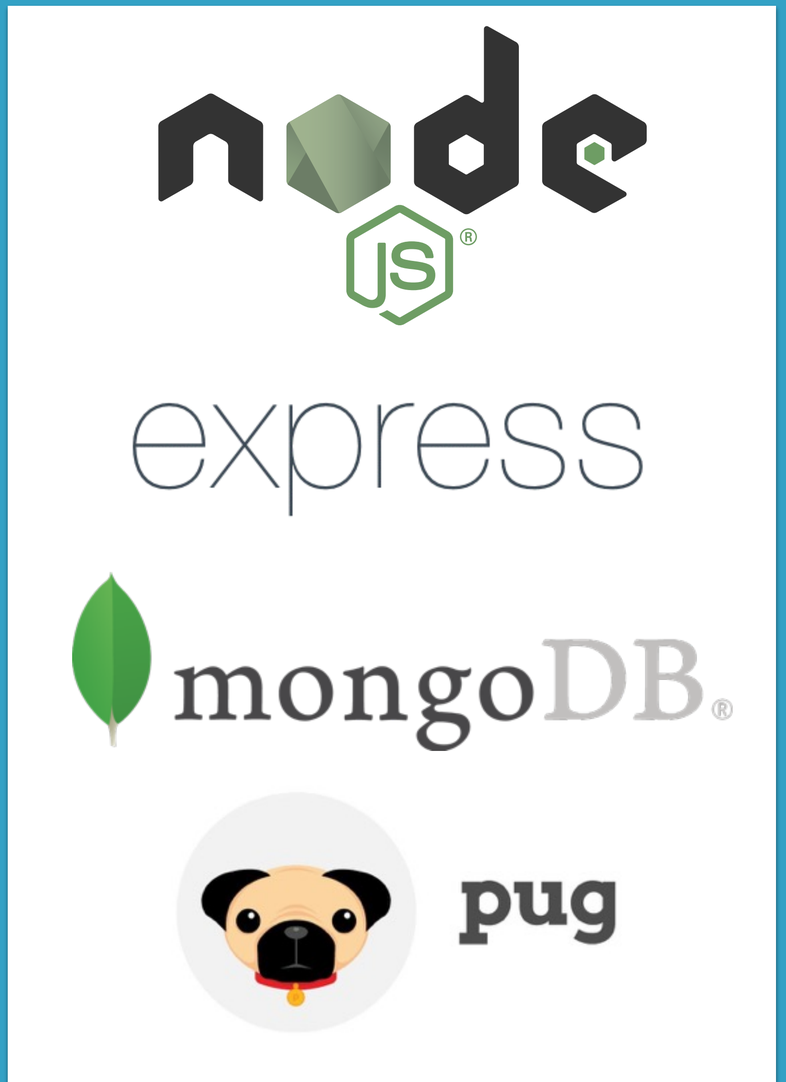
# NodeJS an introduction

## Technical overview

- Theory
- Core Concepts
- Eco-System (NPM)

## Hands on lab

- Express
- MongoDB
- Jade/Pug



# What is NodeJS

Theory

Core Concepts

# What is NodeJS

## Theory

A runtime environment based on a

***non-blocking, event-driven***

***I/O model***

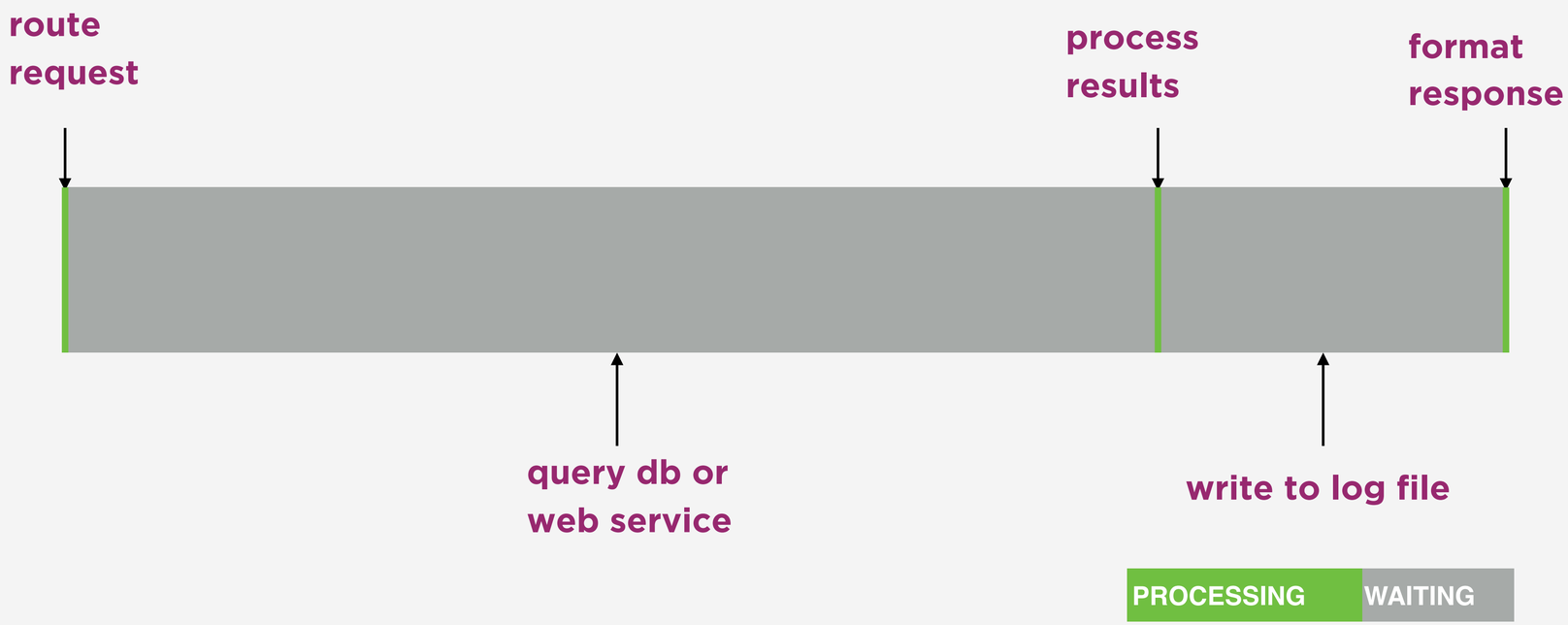
# Theory

## I/O (Input/Output)

**Communication between computer(part) systems.**

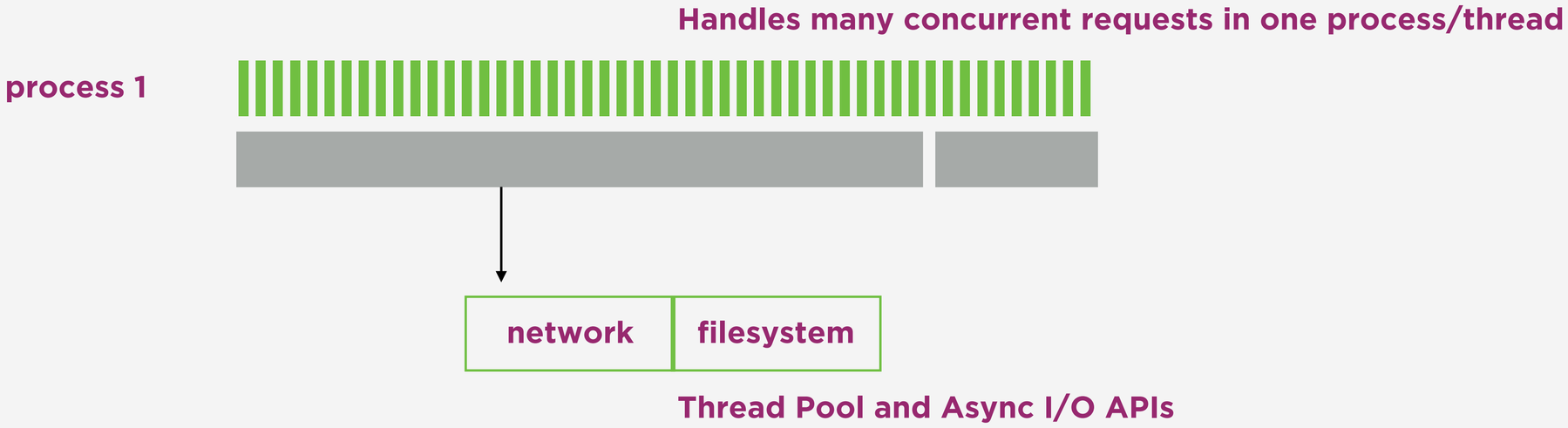
# Theory

# Blocking I/O



# Theory

# Non Blocking I/O



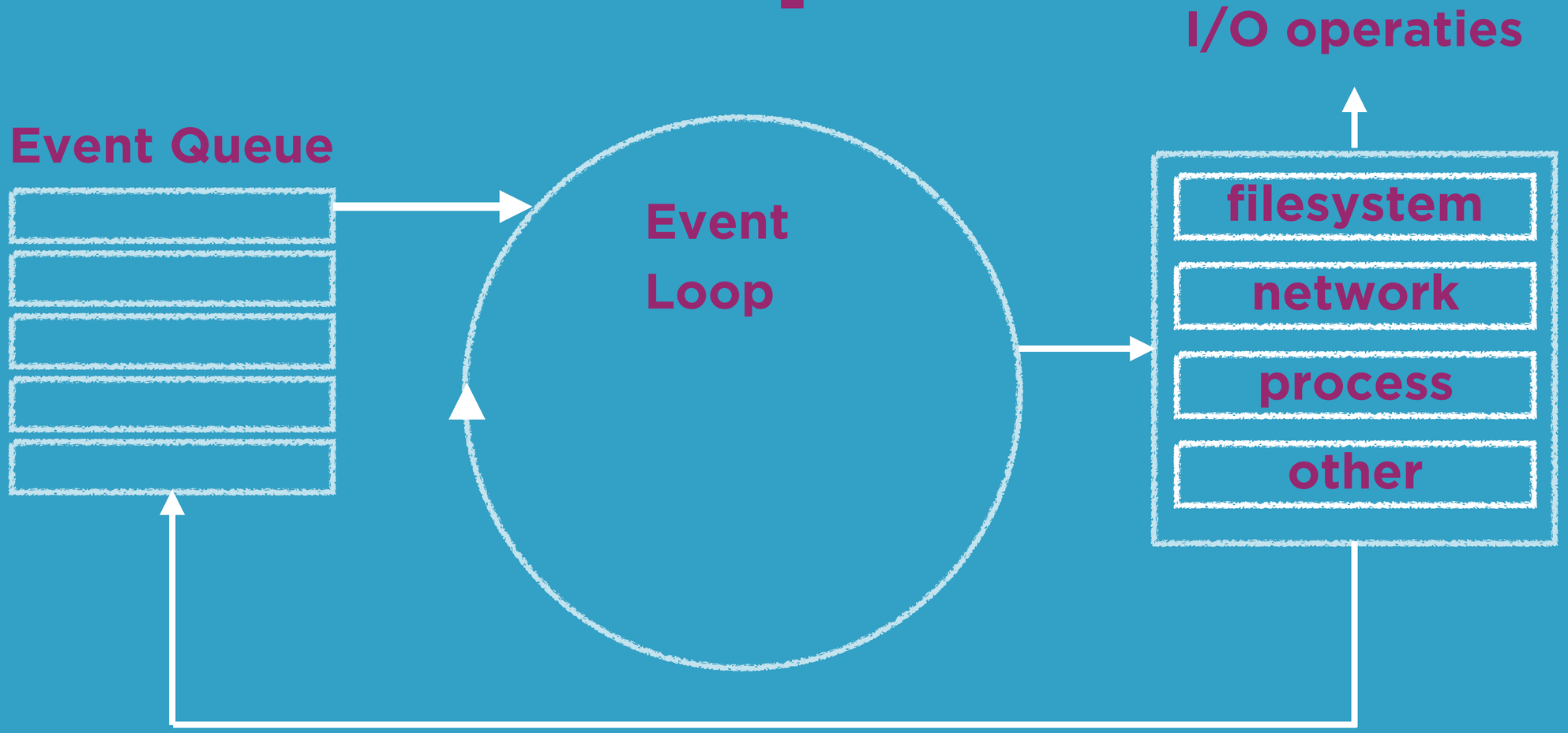
# Theory

# Where's the advantage

| | |
|---|---|
| **L1** | 3 cycles |
| **L2** | 14 cycles |
| **RAM** | 250 cycles |
| **Disk** | 41 000 000 cycles |
| **Network** | 240 000 000 cycles |

# Theory

# Event Loop



# Theory

## Platform

**node standard library**

---

**node bindings**

(http, socket, file system)

---

**V8**

(JS engine)

**libuv**

(Event Loop, Thread Pool, Async I/O API's)

# What is NodeJS

## Core Concepts

Callbacks

Events

Streams

Modules

# Core Concepts

## Callbacks

**In javascript functions are *Objects:***

- store in a variable
- pass on as argument
- create within a function
- return from a function

**Code Demo...**

# Core Concepts

# Callbacks

## Exercise

Create a persons object. Pass this object into a callback function that validates the email adres and that properties aren't empty.

```
let person = {
  firstname: 'Roeland',
  name: 'De Smedt',
  email: 'roeland.desmedt[at]c4j.be',
  company: 'Bewire',
  function: 'NodeJs Consultant',
  linkedin: 'https://www.linkedin.com/in/roelanddesmedt',
  github: 'https://github.com/RoelandDS',
  twitter: '',
  verified: false
};
```

```
for (let key in object){
  object[key]
}
```

# Core Concepts

## Events

**Asynchronous event-driven architecture:**

- object type (‘emitters’)
- periodically emits named events
- calls Functions (‘listeners’)

**Code Demo...**

# Core Concepts

## Streams

**Data in chunks*:***

- readable, writeable, both
- inherit from EvenEmitter

**Code Demo...**

# Core Concepts

## Modules

**CommonJS module specificatie**

http net dns

https **built in modules** os

fs events url zlib

# Core Concepts

## Modules

- organising code
- reuse of code
- require
- module.exports

**Code Demo...**

# NPM


## Node Package Manager

*share and reuse code*

```
$ npm install [package_name]
$ npm install -g [package_name]
$ npm install —save [package_name]
```

https://docs.npmjs.com/

# NPM

# Package.json

```
{
  "name": "Express-tutorial",
  "version": "0.0.1",
  "author": "Roeland De Smedt <roeland.desmedt@c4j.be>",
  "license": "MIT",
  "repository": {
    "type": "git",
    "url": ""
  },
  "dependencies": {
    "body-parser": "^1.15.2",
    "cookie-parser": "^1.4.3",
    "express": "^4.14.0",
    "morgan": "^1.7.0",
    "pm2": "^1.1.3",
    "pug": "^2.0.0-beta3"
  }
}
```

```
const http = require('http');
const express = require('express');
const path = require('path');
const logger = require('morgan');
const cookieParser = require('cookie-parser');
const bodyParser = require('body-parser');
```

# NPM

## Semantic Versioning

- Patch releases: **1.0** or **1.0.x** or **~1.0.4**
- Minor releases: **1** or **1.x** or **^1.0.4**
- Major releases: * or **x**

# Hands on Lab

## Blog application

A simple application that creates, saves and view blog entries.
We use the ExpressJS web framework
Bootstrap for simple styling
Pug (formerly known as Jade) as templating engine
We'll store data in mongoDB

# ExpressJS

## Hello world

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

app.get('/', function (req, res) {
  res.send('Hello World!');
});

app.listen(3000, function () {
  console.log('Example app listening on port 3000!');
});
```

# Lab

## Start of our lab

Package.json
NPM install
app configuration of basic middleware
initial route

# Templating engine

## Pug

```
doctype html
html(lang='en')
  head
    title Welcome #{name}
  body
    h1 Let's start building a web app
    #container.col
      p
        | We will build a blog tool
        | using node, express and mongo
```

Lab

# Creating a new route

Create new Pug file
create route ‘/newBlog’
return new pug file
linking bootstrap

# Lab

# Creating a POST route

create new Blog form
create POST route
display POST body in terminal
redirect to main route
create a navigation

# MongoDB

# A noSQL database

Storing data as Json objects (documents)
Documents are stored in collections
A database is a collection of collections

```
{
  "_id" : ObjectId("57973303e1860f068bf60bb2"),
  "name" : "test",
  "body" : "test",
  "title" : "test",
  "uploadDate" : ISODate("2016-07-26T09:53:07.165Z"),
  "__v" : 0
}
```

# MongoDB

# Mongoose

Elegant mongoDB object modeling for NodeJS

```
const mongoose = require('mongoose');
mongoose.connect('mongodb://localhost/SOT-dev');
mongoose.connection.on('error', (err) => {
  console.log('MongoDB connection error: ', err);
  process.exit(-1);
});
```

# MongoDB

## Creating a schema

```
let cat = new Schema({ name: String });
module.exports = mongoose.model('Cat', cat);
```

# MongoDB

# Using a schema

```
const Cat = require('./../models/cat.model');
let newCat = new Cat();
newCat.name = 'Sylvester';

newCat.save().then((data) => {
  console.log(data);
});
```

Lab

# Saving a blog

create a Blog model
create a blog module
save post data to mongo
redirect to main route

# MongoDB

## building a query

```
Cat
   .find()
   .sort({name: -1})
   .then((data) => {
   console.log(data);
});
```

Lab

# Building our blog app

show all blogs on main route
create a blog detail page
create a comment form on blog detail
show comments on blog detail