



# Node.js: a little overview

*Nicola Del Gobbo*

# Who am I?

**Developer**



**Full stack developer highly focused on performance**



**N-API Team**

**My research field**

How to implement a native addon for Node.js with programming languages different from C or C++

# Agenda

Introduction to Node.js



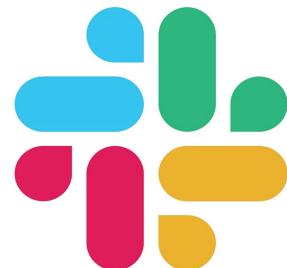
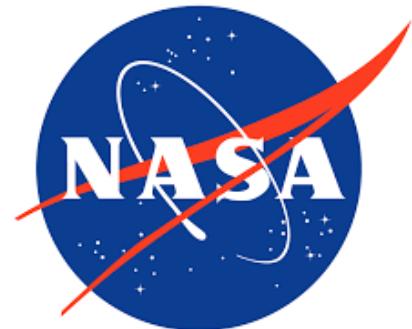
How to create distributed services



Native Addons



# Who use Node.js?



# What can I do with **Node.js**?

# Everything

Great thank to Node.js **Community** and **Working Groups**



Packages

**963,619**

Downloads - Last Week

**10,822,198,617**

Downloads - Last Month

**48,577,378,041**

# What can I do with Node.js?

express



CLI APPLICATION

DISRTRUBUTED SYSTEM

REAL TIME SYSTEM

DESKTOP APPLICATION

# Release

LTS	Current		
Recommended For Most Users	Latest Features		
			
<b>Windows Installer</b>	<b>macOS Installer</b>		
node-v10.15.3-x86.msi	node-v10.15.3.pkg		
<a href="#">Windows Installer (.msi)</a>	<a href="#">32-bit</a>	<a href="#">64-bit</a>	
<a href="#">Windows Binary (.zip)</a>	<a href="#">32-bit</a>	<a href="#">64-bit</a>	
<a href="#">macOS Installer (.pkg)</a>		<a href="#">64-bit</a>	
<a href="#">macOS Binary (.tar.gz)</a>		<a href="#">64-bit</a>	
<a href="#">Linux Binaries (x64)</a>		<a href="#">64-bit</a>	
<a href="#">Linux Binaries (ARM)</a>	<a href="#">ARMv6</a>	<a href="#">ARMv7</a>	<a href="#">ARMv8</a>
<a href="#">Source Code</a>	node-v10.15.3.tar.gz		

[Windows Installer \(.msi\)](#)

[Windows Binary \(.zip\)](#)

[macOS Installer \(.pkg\)](#)

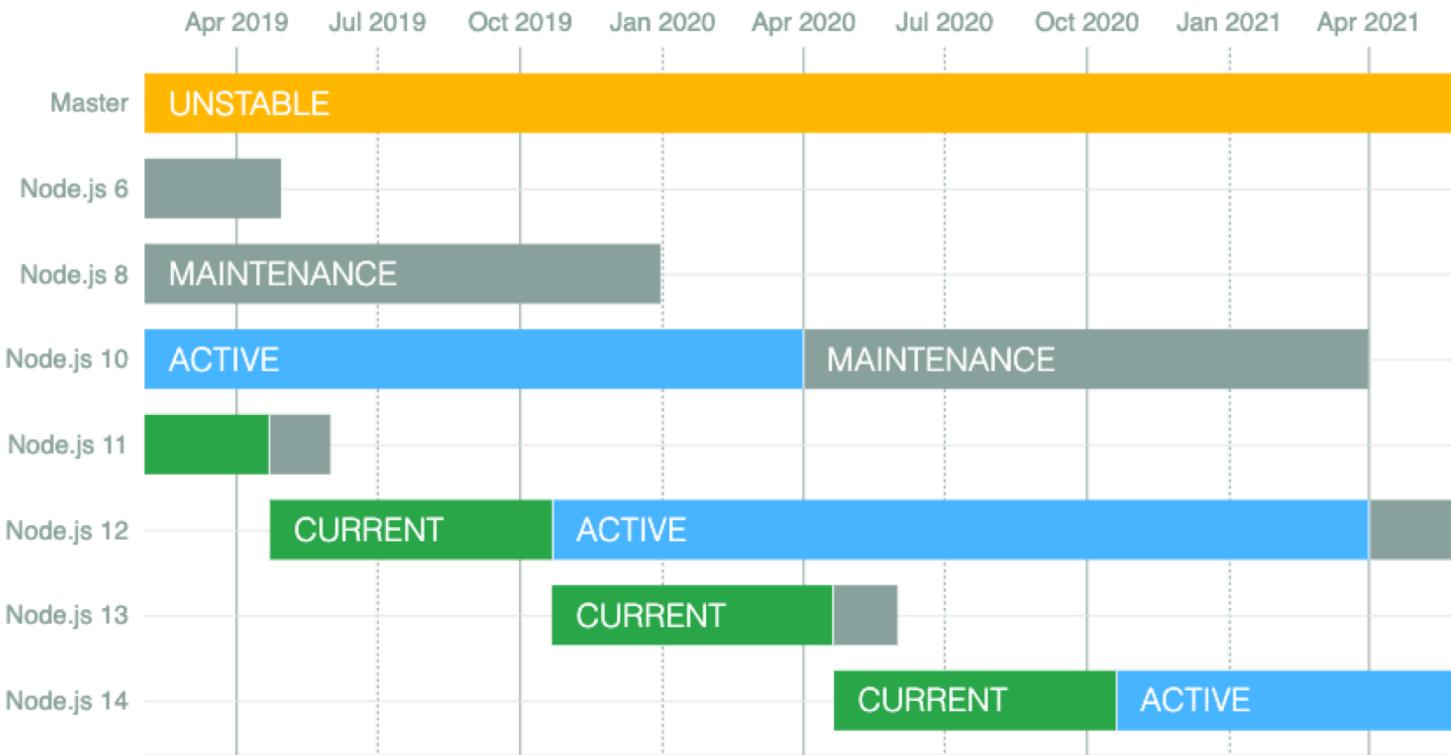
[macOS Binary \(.tar.gz\)](#)

[Linux Binaries \(x64\)](#)

[Linux Binaries \(ARM\)](#)

[Source Code](#)

# Release



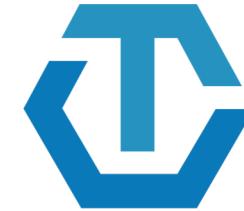
<https://github.com/nodejs/Release>



Choose your modules



Add health checking



OPENTRACING  
Track your requests



Power your metrics



Build your Docker image



Deploy to Kubernetes

# What is Node.js?

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



A **JavaScript engine**



**Asynchronous I/O**

**~100k LOC of JS and C++**

Node is a **glue**

# Node.js architecture

Node.js API

Node.js Bindings

C / C++ Addons



c-ares

HTTP  
parser

Open  
SSL

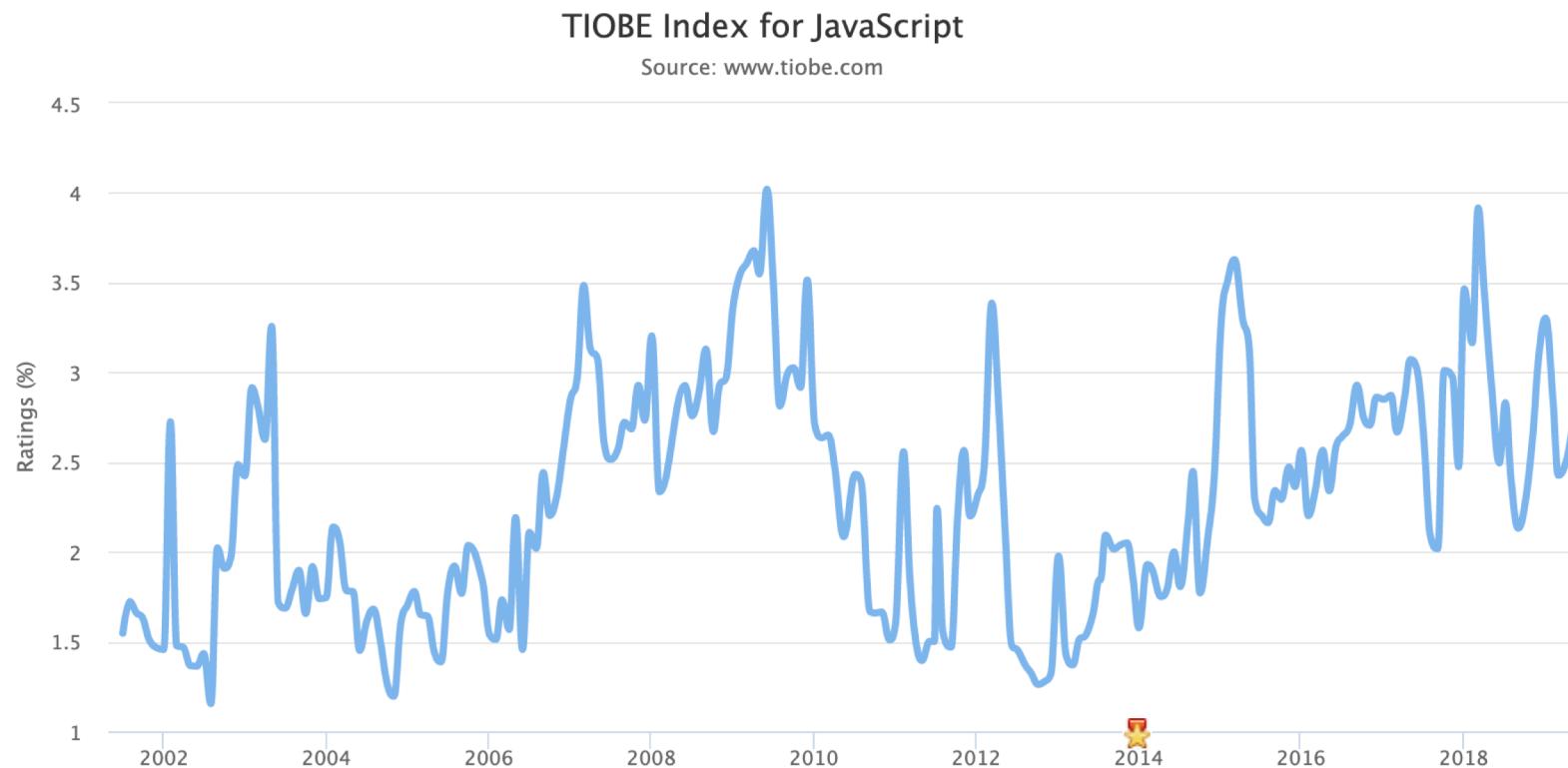
zlib

nghttp2

# JavaScript

“The world's most **misunderstood** programming language”

*Douglas Crockford*



# JavaScript

- **let** and **const**
- Better **Unicode** support
- **Template literals**
- **Arrow function**
- **Default parameters**
- Spread operator
- **Class**
- **Destructuring**
- Symbols
- **Sets and Maps**
- **Iterator**
- Generator
- **Promise**
- **Async / Await**
- **Async Iterator**
- Proxy API
- Reflection API

ES2015

ES2016

ES2017

ES2018

ES2019

# JavaScript

“JavaScript developers should **focus** on writing *idiomatic, readable* and *maintainable* code”

*Peter Marshall & Arunesh Chandra*



“Use JavaScript like JavaScript”

# CONCURRENCY MODEL?



**Single threaded**  
by default

Worker Threads from Node.js v10.5.0

**Asynchronous I/O**

Execution never stop

**Don't block the event loop**

# Synchronous model?

```
var result = db.query('select ...');  
// use result
```

**Blocking** the whole process or you need to have **multiple execution stacks**

# Asynchronous model?

```
db.query('select ...', function (result) {  
  ... // use result  
})  
  
// Execution continue  
console.log('Do other stuff ...')
```

The **main** process is **never blocked**. No strategy is required to handle competing requests

# The event loop



# Asynchronous in low level

```
'use strict'

const bcrypt = require('bcrypt')

const SALT_ROUNDS = 10
const STR_TO_HASH = 'Node.js: little overview'

async function run() {
  const hashed = await bcrypt.hash(STR_TO_HASH, await bcrypt.genSalt(SALT_ROUNDS))
  console.log(hashed)
}

run().catch( err => { console.error(err) })
```

# Asynchronous in low level

```
Napi::Value GenerateSalt(const Napi::CallbackInfo& info) {
    if (info.Length() < 3) {
        throw Napi::TypeError::New(info.Env(), "3 arguments expected");
    }
    if (!info[1].IsBuffer() || (info[1].As<Napi::Buffer<char>>().Length() != 16)) {
        throw Napi::TypeError::New(info.Env(), "Second argument must be a 16 byte Buffer");
    }
    const int32_t rounds = info[0].As<Napi::Number>();
    Napi::Function callback = info[2].As<Napi::Function>();
    Napi::Buffer<char> seed = info[1].As<Napi::Buffer<char>>();
    SaltAsyncWorker* saltWorker = new SaltAsyncWorker(callback, std::string(seed.Data(), 16), rounds);
    saltWorker->Queue();
    return info.Env().Undefined();
}
```

# Modularity



**CommonJS Module**

**ES6 Module from v8.5.0**

**“Se non hai provato Node.js non sai cos’è la modularità e il riuso”**

*Matteo Collina*

# Modularity

```
'use strict'

function Person (opts) {
  this._firstName = opts.firstName || ''
  this._lastName = opts.lastName || ''
}

Person.prototype.toString = function toString() {
  return `First name: ${this._firstName}
Last name: ${this._lastName}`
}

module.exports = Person
```

```
'use strict'

function isPrime (p) {
  const upper = Math.sqrt(p)
  for (let i = 2; i <= upper; i++) {
    if (p % i === 0) {
      return false
    }
  }
  return true
}

exports.isPrime = isPrime
```

# Modularity

```
'use strict'

const Persone = require('./Person')

const p = new Persone({
  firstName: 'Nicola',
  lastName: 'Del Gobbo'
})

console.log(p.toString())
```

```
'use strict'

const isPrime = require('./is-prime')

console.log(isPrime(3))
```

# Events

**EventEmitter** is a very important class in Node.js. It provides a **channel for events to be dispatched and listeners notified**. Many objects you will encounter in Node.js inherit from EventEmitter

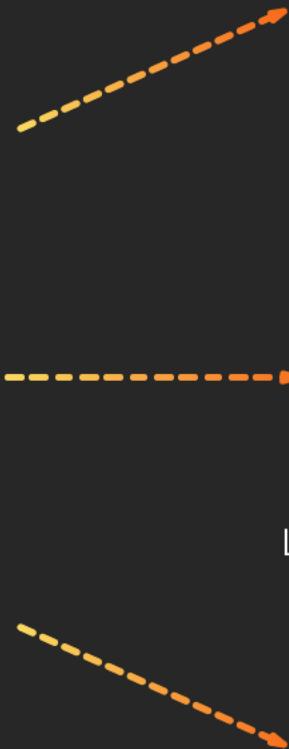


**Emitter**

Emitter object emits event with optional data associated it



**Event with data**



**Listener**

Listeners execute some actions when one event is emitted



# Events

```
'use strict'

const { EventEmitter } = require('events')

class UserCtrl extends EventEmitter {

  constructor () {
    // do some initialization tasks
  }

  addUser (user) {
    // register user on db
    // emit event
    this.emit('user:added', { userName: 'NickNaso' })
  }
}
```

```
'use strict'

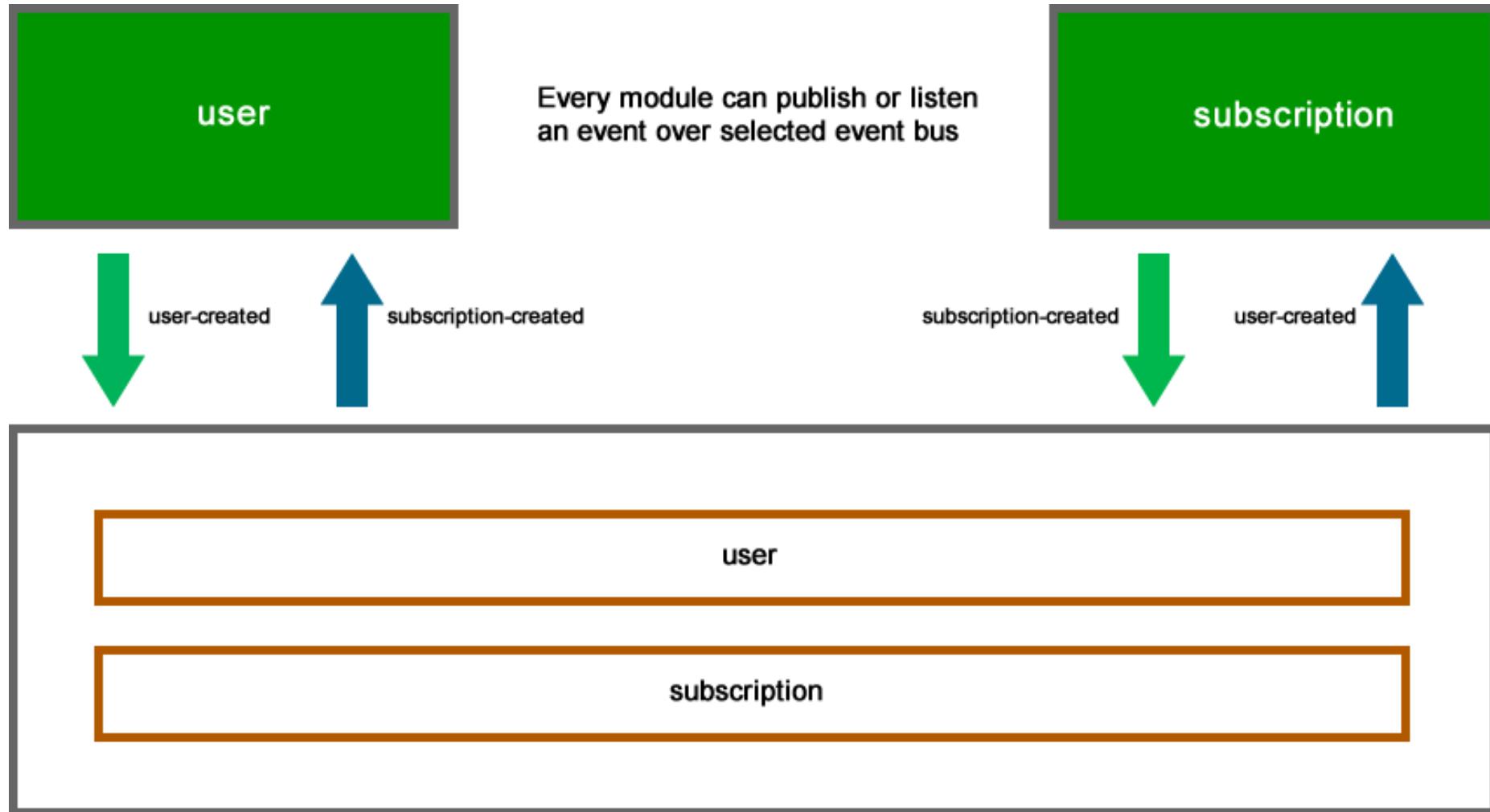
const UserCtrl = require('./UserCtrl')

const ctrl = new UserCtrl()

ctrl.on('user:added', data => {
  // do something with data
})

ctrl.addUser({
  userName: 'NickNaso',
  password: 'nodejs'
})
```

# Event bus



# Event bus

```
'use strict'

const Hertz = require('hertz')

// Obtain or create a new frequency, a channel where you can emit or listen for
// an event issued by other modules
const usr = Hertz.tune('user')

// Listen for event 'user:add'
usr.on('user:add', function (data) {
  console.log('NEW USER ADDED WITH FOLLOWING DATA:')
  console.log(data)
})

// Emit event 'user:add'
usr.emit('user:add', {
  username: 'NickNaso',
  password: '*****',
  email: 'nicoladelgobbo@gmail.com'
})
```

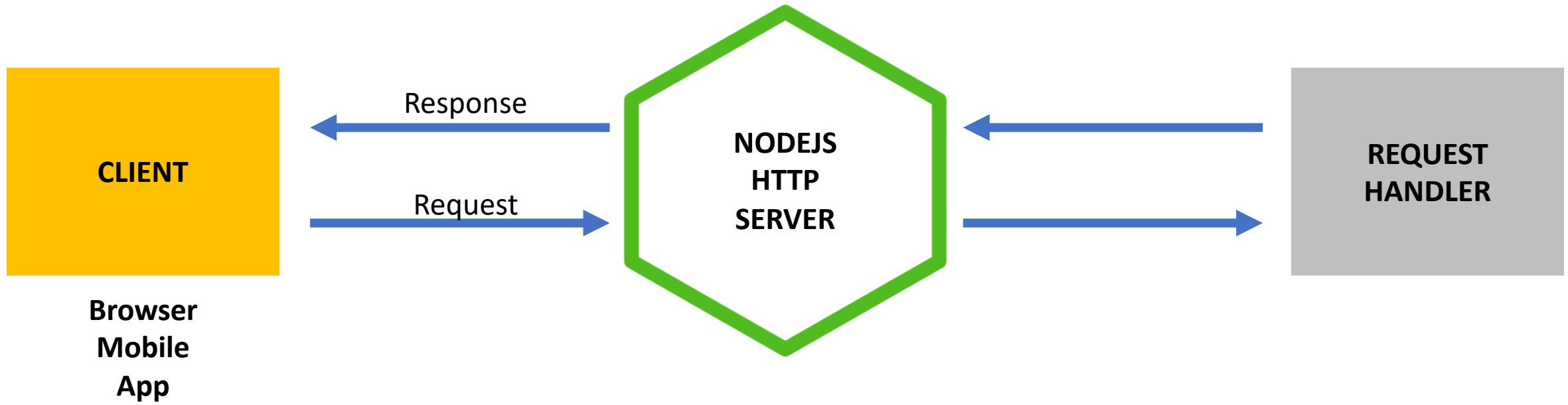
# Stream

- **Buffer**: data structure to store and transfer arbitrary binary data
- **Stream**: abstract interface for working with streaming data

# Stream vs Buffer

- **Streams** keep a low memory footprint even with large amounts of data
- **Streams** allow you to process data as soon as it arrives

# Web application



<http://your-web-application>

# Use http core

```
'use strict'

const http = require("http")
const url = require("url")

function onRequest(request, response) {
  let pathname = url.parse(request.url).pathname
  console.log("Request for " + pathname + " received.")
  if (pathname === "/start") {
    response.writeHead(200, { "Content-Type": "text/plain" })
    response.write("Hello")
    response.end()
  } else if (pathname === "/finish") {
    response.writeHead(200, { "Content-Type": "text/plain" })
    response.write("Goodbye")
    response.end()
  } else {
    response.writeHead(404, { "Content-Type": "text/plain" })
    response.end("404 Not Found")
  }
}

http.createServer(onRequest).listen(5000)
console.log("Server has started.")
```

Route handler

Request handler

# Frameworks



express

# Express

 [expressjs / express](#)

 Unwatch ▼

1,852

 Unstar

43,741

 Fork

7,387

 Code

 Issues 115

 Pull requests 55

 Wiki

 Insights

Fast, unopinionated, minimalist web framework for node. <https://expressjs.com>

[javascript](#) [nodejs](#) [express](#) [server](#)

 5,502 commits

 10 branches

 277 releases

 220 contributors

 MIT



Downloads - Last Week

**7,495,349**

Downloads - Last Month

**33,248,650**

# Express

- Minimalist
- Unopinionated
- Fast (about 38k req/sec)
- Simple (**do one thing well** philosophy from Unix world)
- Wrapper of `http` core module

```
'use strict'

const http = require("http")
const express = require('express')

const app = express()
```

```
app.get('/start', (req, res) => {
  res.status(200).send('Hello')
})
```

**Route handler**

```
app.get('/finish', (req, res) => {
  res.status(200).send('Goodbye')
})
```

```
http.createServer(app).listen(5000)
console.log("Server has started.")
```

**For web sockets bind express to http module**

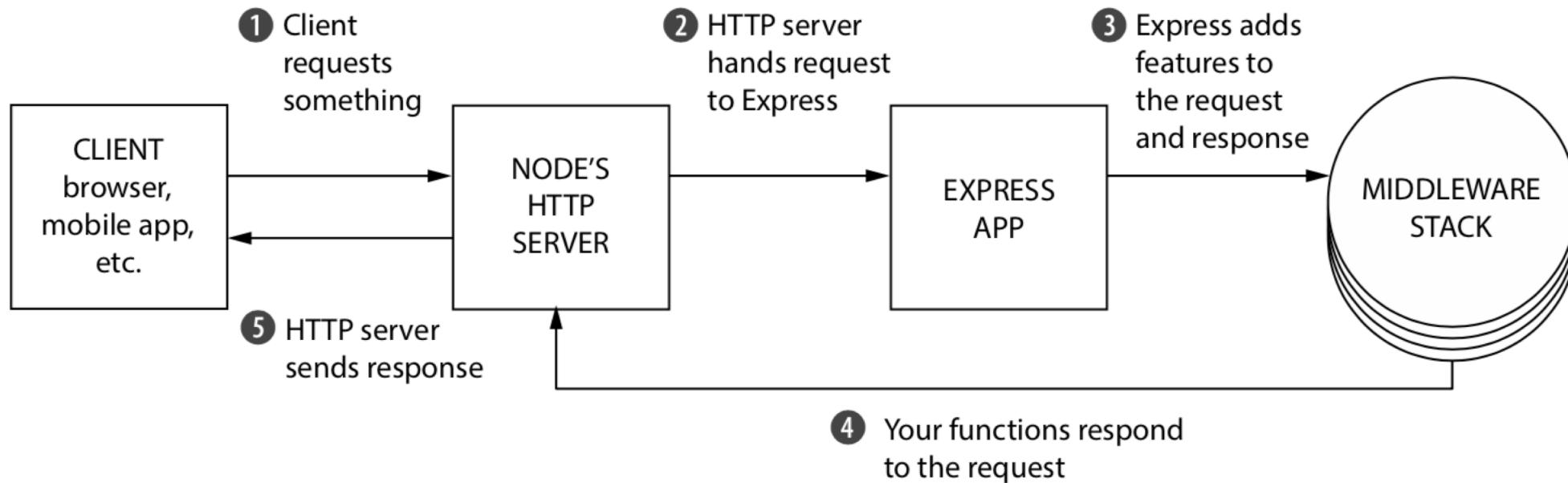
# Express milestones

- Middleware
- Error handler
- Router
- Views / template engine

# Middlewares

It's always a question to manipulate the **Request**  
and **Response** object

# Middlewares



# Use middlewares

```
'use strict'

const path = require('path')
const http = require("http")
const express = require('express')
const morgan = require('morgan')
const serve = require('express').static

const app = express()
```

## Attach middleware to Express

```
app.use(morgan('combined'))
app.use(serve(path.join(__dirname, 'public')))
```

```
app.get('/', (req, res) => {
  res.status(200).sendFile('/index.html')
})
```

```
http.createServer(app).listen(5000)
console.log("Server has started.")
```

# Custom middlewares

```
'use strict'

const bannedIps = ['192.168.0.1', '192.168.0.2', '192.168.0.3']

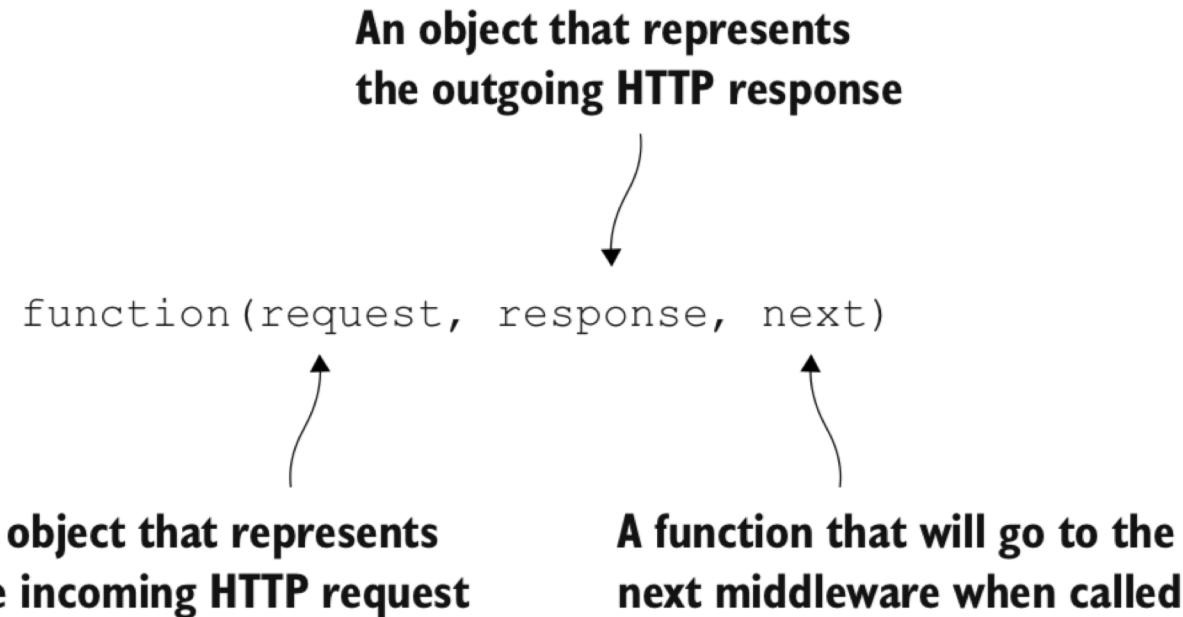
function myMiddleware (req, res, next) {

  // ... DO SOMETHING WITH REQUEST AND RESPONSE
  if (bannedIps.includes(req.ip)) {
    const err = new Error('Your IP is banned go away')
    next(err)
  } else {
    next()
  }
}
```

```
'use strict'

function myMiddleware (bannedIps) {
  // ... DO SOMETHING WITH BANNED IPs
  return function (req, res, next) {
    // ... DO SOMETHING WITH REQUEST AND RESPONSE
    if (bannedIps.includes(req.ip)) {
      const err = new Error('Your IP is banned go away')
      next(err)
    } else {
      next()
    }
  }
}
```

# Middlewares

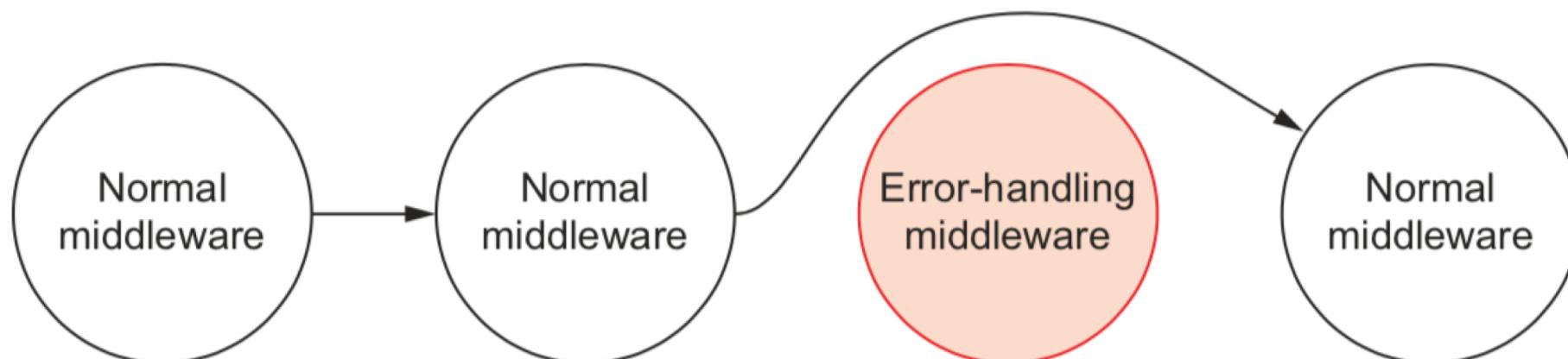


<http://expressjs.com/en/resources/middleware.html>

# Error handler

```
'use strict'

function errorHandler () {
  return function (err, req, res, next) {
    // ... PARSE YOUR ERROR
    // ... DO SOMETHING WITH REQUEST AND RESPONSE
    // ... IDENTIFY STATUS CODE AND MESSAGE FOR YOUR RESPONSE
  }
}
```



# Routing

```
app.get('/songs', (req, res, next) => {
  // ... DO SOMETHING ON YOUR ROUTE
})

app.get('/songs/:title', (req, res, next) => {
  // ... DO SOMETHING ON YOUR ROUTE
})

app.post('/songs', (req, res, next) => {
  // ... DO SOMETHING ON YOUR ROUTE
})

app.put('/songs/:title', (req, res, next) => {
  // ... DO SOMETHING ON YOUR ROUTE
})

app.delete('/songs/:title', (req, res, next) => {
  // ... DO SOMETHING ON YOUR ROUTE
})
```

**Routing** refers to determining how an application responds to a client request to a particular endpoint, which is a URI (or path) and a specific HTTP request method

# Validate your input



The fastest JSON Schema validator for  
Node.js and browser

# Views / template engine

Pug - Mustache - Dust - Nunjuks - EJS

```
app.set('view engine', 'ejs') Set the engine
```

```
app.engine('ejs', require('ejs').__express) Register the engine
```

```
app.set('views', path.join(__dirname, 'views')) Set views folder
```

# Security



**Helmet** helps you secure your Express apps  
by setting various HTTP headers

# Logger

**Log everything** that happens in your application

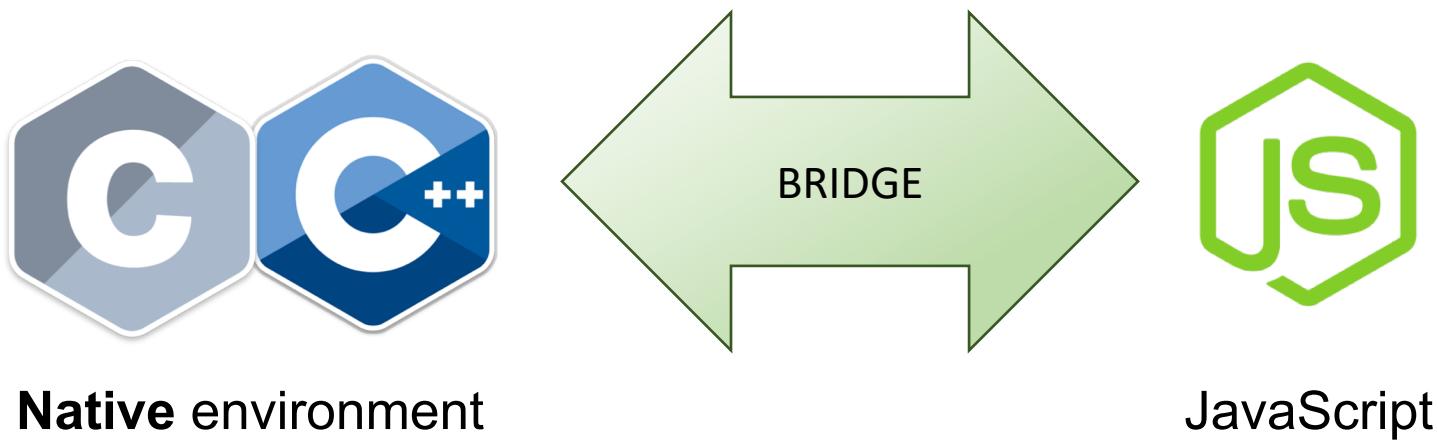
Pay attention there is a **cost** for logging

**Winston**

**pino** 

# What is Node.js Native Add-on?

**C / C++** code called from **JavaScript**



## From Node.js documentation

*Node.js Addons are **dynamically-linked shared objects**, written in **C++**, that can be loaded into Node.js using the **require()** function, and used just as if they were an ordinary Node.js module.*

*They are used primarily to provide an **interface** between **JavaScript** running in Node.js and **C/C++** libraries.*

# Why?

## Performance

In general **C / C++** code performs better than JavaScript code, but it's not always true.

**Image** processing (in average 6 times faster)

**Video** processing

**CRC** cyclic redundancy check (in average 125 times faster)

**Compression**

**Scientific calculus**

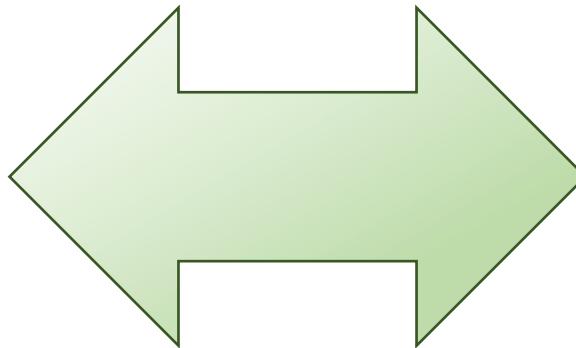
**Algorithms that execute CPU heavy tasks**

# Why?

## Integrate legacy application

You have the source code of an **old C / C++** application and want to expose something of its functionalities

```
#include <stdio.h>
int main(void)
{
    printf("Hello World!\n");
    return 0;
}
```



# Why?

You don't find what fits your specific needs on **npm**

Sometimes completely implementing the module  
in **JavaScript** is not the right solution

Think at libraries like:

**ImageMagick**

**Ghostscript**

**FFmpeg**

**TensorFlow**

...

# Why?

## Better error handling

Even if you will use **an old C / C++ library** you will get an error code that explains the error that just happened

In **new C / C++ library** you have the **exception**

You don't have to parse some string to identify if there was an error and what kind of it

A black silhouette of a person in a dynamic, forward-leaning pose, appearing to swing or crawl across the frame. The person's body is angled from the bottom left towards the top right, with one arm extended forward and the other pulling on a dark, horizontal line that extends across the top of the image. The silhouette is set against a solid red background.

**With great power comes great responsibility**

Ben Parker

# Problems

## Fragmentation API

The API to implement native add-ons has been changed across different version of Node.js

Most of the changes were on **V8 API** and **ObjectWrap API**



**0.8 - 0.10.x - 0.12.x - 1.x - 2.x - 3.x - 4.x - 5.x - 6.x - 7.x - 8.x - 9.x - 10.x - 11.x - 12.x**

For more info <http://v8docs.nodesource.com/>

# Problems

Need an adapter to stay compatible across different version of Node.js

- **NAN** - Native Abstraction for Node.js
  - **API compatibility**
  - Strong bonded with **V8 API**
  - You have to recompile your native add-ons switching to different version of Node.js

# Problems

## End user

```
| was compiled against a different Node.js version using
| NODE_MODULE_VERSION 51. This version of Node.js requires
| NODE_MODULE_VERSION 57. Please try re-compiling or re-installing
| the module (for instance, using `npm rebuild` or `npm install`).
```

## Maintainers



was compiled against a different Node.js version using

#117

was compiled against a different Node.js version using NODE\_MODULE\_VERSION 59. This version of Node.js requires NODE\_MODULE\_VERSION 57. Please try re-compiling or re-installing the module (for ...)

JCMais/node-libcurl Opened by gSION on 6 Mar 1 comment



Addon

Node.js: 6.12.2  
V8: 5.1.281  
NODE\_MODULE\_VERSION: 48

## ABI break

- At worst, mysterious segfaults
- At best, addon fails to load
- `NODE_MODULE_VERSION` mismatch

`Node.js: 8.9.3`

`V8: 6.1.534`

`NODE_MODULE_VERSION: 57`

# Problems

**Write portable C / C++ code**

Your native code **must compile and run** on different:

ARCHITECTURE

PLATFORM

COMPILER

# Problems

## Documentation

- **C / C++** libraries that you are integrating are not well documented
- There are good **references** but not so much practical guide focusing on complex concepts about native add-ons

# N-API

- Kicked off in VM Summit April 2016
  - Dual implementation from the start (V8 / ChakraCore)
- **EPS** Dec 2016 <https://github.com/nodejs/node-eps/blob/master/005-ABI-Stable-Module-API.md>
- First half of 2017 - refinement and backporting module
- May 2017 experimental release in Node.js 8
- **VM** Summit July 2017 – Exit criteria agreed
- N-API leaves experimental status March 14 2018
- Backported to all **LTS** release
- Improve core and C++ Wrapper (node-addon-api)
- Create ecosystem

# N-API

- N-API is C whereas node-addon-api is C++
- node-addon-api not technically part of ABI stable API
  - Helper to simplify code using C++
  - All inline
  - Only depends on export N-API C functions

# N-API and node-addon-api

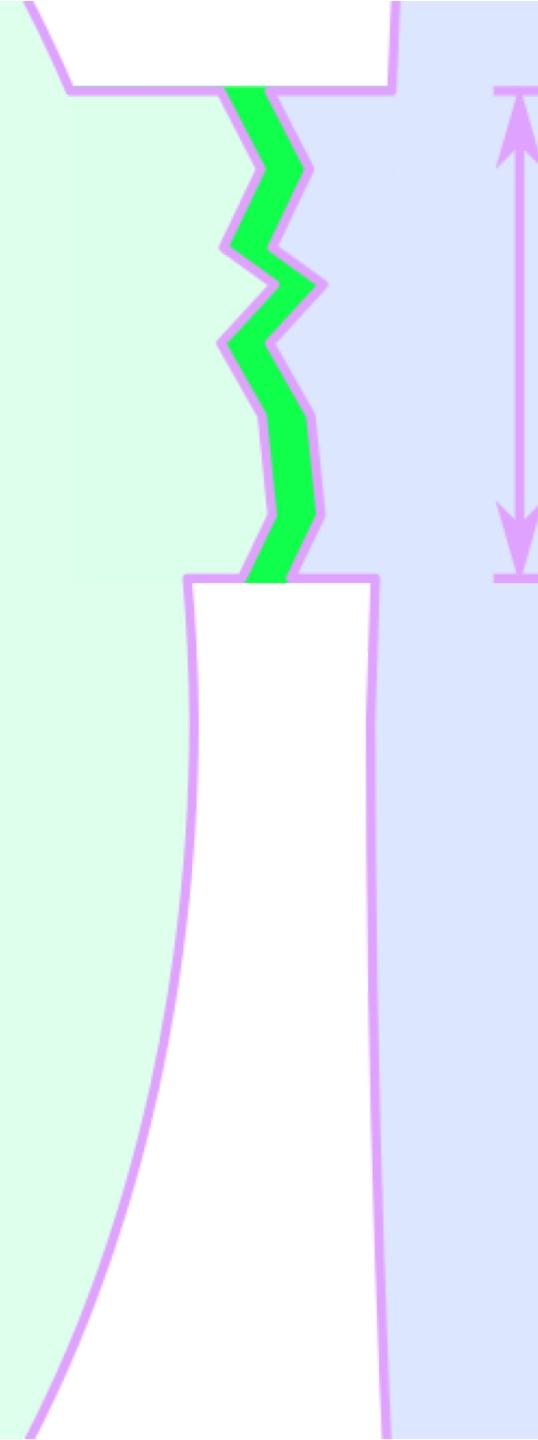
```
1 Object obj = Object::New(env);  
2  
3 obj["foo"] = String::New(env, "bar");
```



```
1 napi_status status;  
2 napi_value object, string;  
3 status = napi_create_object(env, &object);  
4 if (status != napi_ok) {  
5     napi_throw_error(env, ...);  
6     return;  
7 }  
8 status = napi_create_string_utf8(env, "bar", NAPI_AUTO_LENGTH, &string);  
9 if (status != napi_ok) {  
10    napi_throw_error(env, ...);  
11    return;  
12 }  
13 status = napi_set_named_property(env, object, "foo", string);  
14 if (status != napi_ok) {  
15    napi_throw_error(env, ...);  
16    return;  
17 }
```

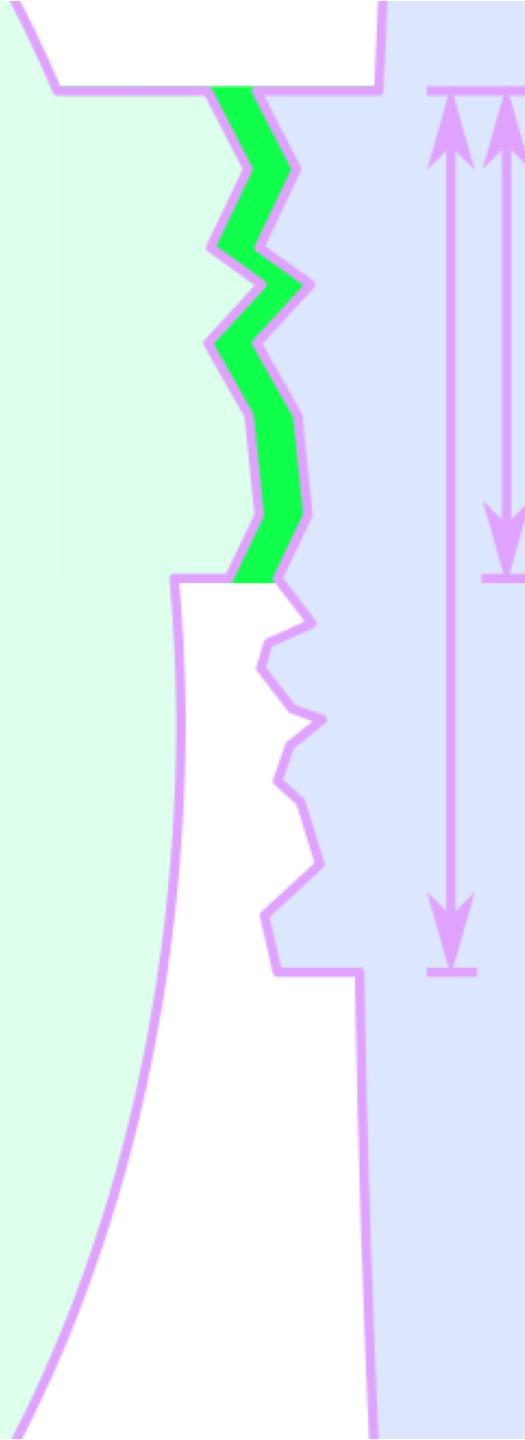
# node-addon-api vs NAN

- Same
  - Includes C++ wrapper
  - Provides common helper functionality
  - Reduces likelihood of needing to change code for new Node.js versions
- Different
  - Does not use V8 Types
  - Not tied to a specific JavaScript engine
  - Preserves compile once/ run multiple versions from N-API
  - Reduces even further likelihood of having to change code



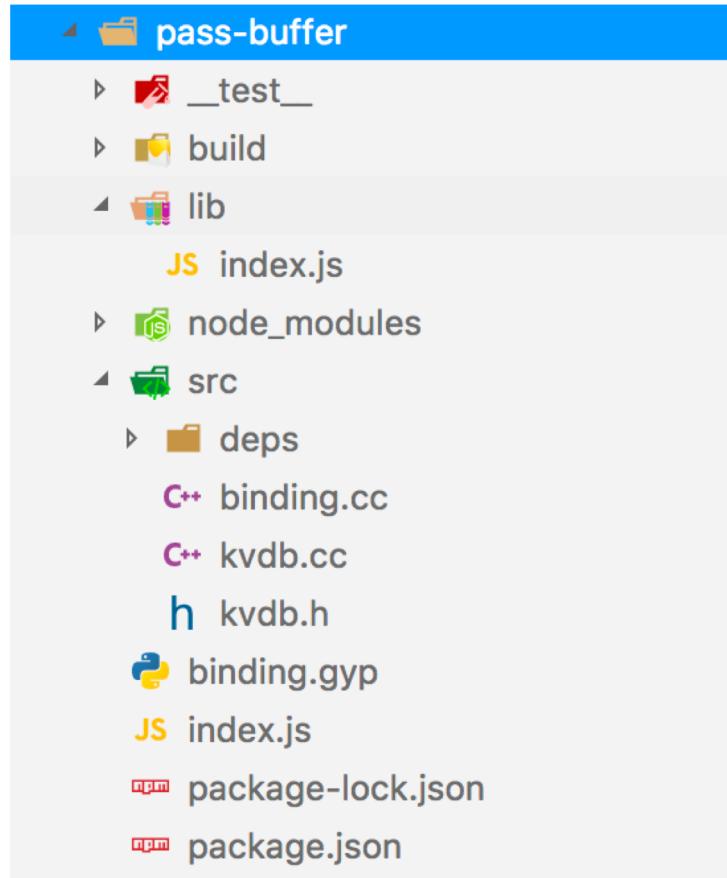
## No ABI break

- NAPI\_VERSION is cumulative
- Addon is forwards compatible



Node.js: later  
V8/ChakraCore: no matter  
NAPI\_VERSION: 2

# How to organize your project



# Example: the JavaScript part

```
{  
  "name": "echo-example",  
  "version": "0.0.0",  
  "description": "Node.js Addons - echo example",  
  "main": "index.js",  
  "private": true,  
  "gypfile": true,  
  "scripts": {  
    "start": "node index.js"  
  },  
  "dependencies": {  
    "node-addon-api": "*",  
    "bindings": "*"  
  }  
}
```

**npm will build the add-on automatically**

# Example: the JavaScript part

1

## Import the add-on

```
const addon = require('bindings')('echo')
```

2

## Call the add-on's function

```
let myEcho = addon.echo('Hey coders!')  
console.log(myEcho)
```

# Example: the `binding.gyp`

```
{  
  "targets": [  
    {  
      "target_name": "echo",  
      "sources": [  
        "src/addon.cc"  
      ],  
      "cflags!": [ '-fno-exceptions' ],  
      "cflags_cc!": [ '-fno-exceptions' ],  
      "include_dirs": [ "<!(node -p \\\"require('node-addon-api').include\\\")" ],  
      "dependencies": [ "<!(node -p \\\"require('node-addon-api').gyp\\\")" ],  
      "conditions": [  
        ['OS=="win"', {  
          "msvs_settings": {  
            "VCCLCompilerTool": {  
              "ExceptionHandling": 1  
            }  
          }  
        }],  
        ['OS=="mac"', {  
          "xcode_settings": {  
            "CLANG_CXX_LIBRARY": "libc++",  
            "GCC_ENABLE_CPP_EXCEPTIONS": 'YES',  
            "MACOSX_DEPLOYMENT_TARGET": '10.7'  
          }  
        }]  
      ]  
    }  
  ]  
}
```

Name used to register the addon

# Example: the C / C++ part

```
Napi::Value Echo(const Napi::CallbackInfo& info) {
    Napi::Env env = info.Env();

    3
    Validate your input - Execute operations on them - Return results

    if (info[0].IsString()) {
        std::string in = info[0].As<Napi::String>();
        return Napi::String::New(env, in);
    } else {
        throw Napi::Error::New(env, "The argument must be a string");
    }
}
```

**2**

## Export objects and functions

```
// Init
Napi::Object Init(Napi::Env env, Napi::Object exports) {
    exports.Set(Napi::String::New(env, "echo"), Napi::Function::New(env, Echo));
    return exports;
}
```

**1**

## Registration and Initialization

```
NODE_API_MODULE(NODE_GYP_MODULE_NAME, Init);
```

# Example

# Asynchronous code

```
const addon = require('bindings')('echo')

function callback(err, data) {
  if (err) {
    console.error(err)
  } else {
    console.log(data)
  }
}
```

```
addon.echo('Hey coders!', callback) 2
```

```
console.log('My next code ...') 1
```

# Asynchronous code

```
void Echo(const Napi::CallbackInfo& info) {
    Napi::Env env = info.Env();
    // You need to check the input data here
    Napi::Function cb = info[1].As<Napi::Function>();
    std::string in = info[0].As<Napi::String>();
    // std::this_thread::sleep_for(std::chrono::seconds(1));
    // In case of error
    // Napi::Error::New(env, "Error ...").Value()
    cb.Call({env.Null(), Napi::String::New(env, in)});
}
```

**Call a callback and return the result back**

# Example

# Asynchronous code

**Preserve reference to function we want call**

**Create worker threads using libuv**

**Handle the results**

# AsyncWorker

```
class EchoWorker : public Napi::AsyncWorker {  
public:  
    EchoWorker(Napi::Function& callback, std::string& echo)  
        : Napi::AsyncWorker(callback), echo(echo) {}
```

```
    ~EchoWorker() {}
```

1

**This code will be executed on the Worker Thread**

```
// This code will be executed on the worker thread  
void Execute() {  
    // Need to simulate cpu heavy task  
    std::this_thread::sleep_for(std::chrono::seconds(1));  
    // std::cout << echo << std::endl;  
}
```

2

**When the Execute method ends its computation the results will be reassembled and passed to JS context**

```
void OnOK() {  
    Napi::HandleScope scope(Env());  
    Callback().Call({Env().Null(), Napi::String::New(Env(), echo)});  
}  
  
private:  
    std::string echo;  
};
```

# AsyncWorker

```
Napi::Value Echo(const Napi::CallbackInfo& info) {
    // You need to check the input data here
    Napi::Function cb = info[1].As<Napi::Function>();
    std::string in = info[0].As<Napi::String>();

1 Create AsyncWorker

    EchoWorker* wk = new EchoWorker(cb, in);
    wk->Queue();

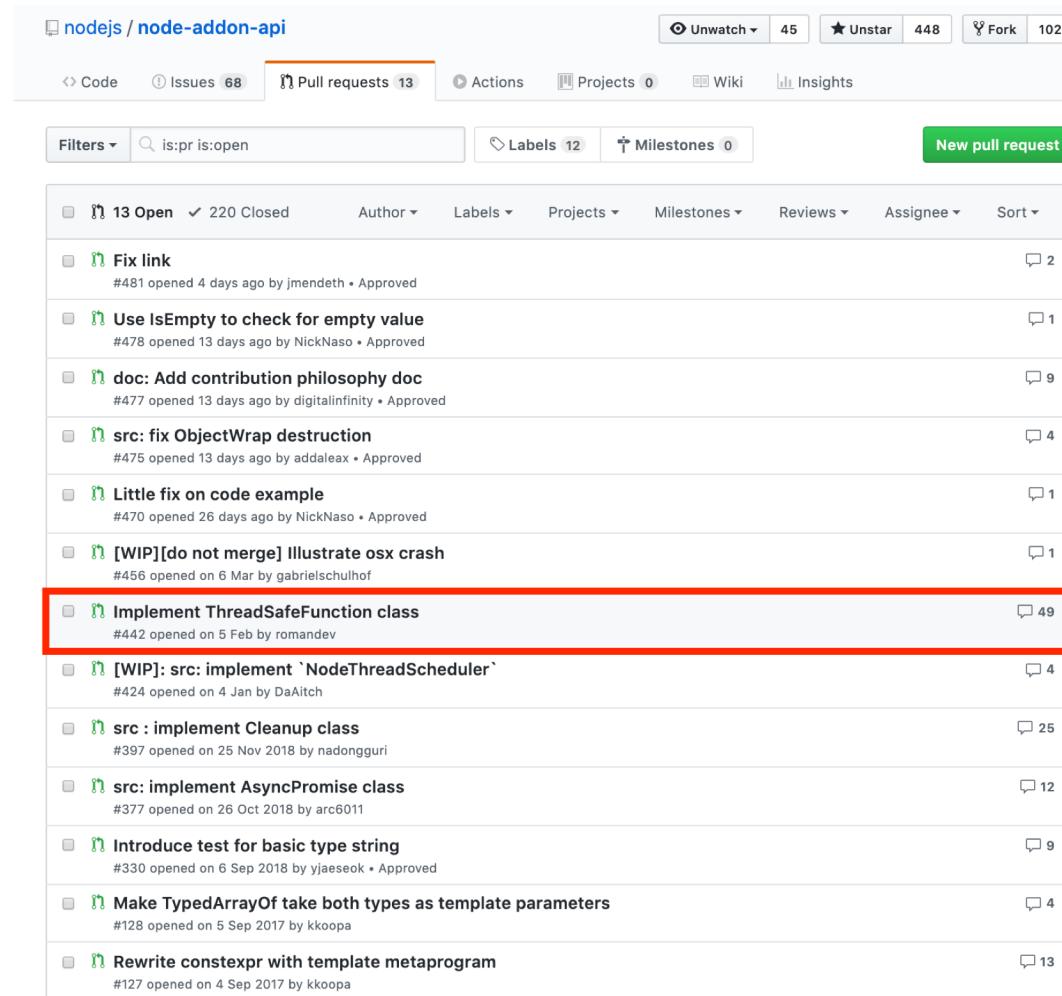
2 Return immediately

    return info.Env().Undefined();
}
```

# Example

# AsyncWorker

You cannot call JavaScript function from the Worker Thread until this PR will be landed



The screenshot shows the GitHub repository for nodejs/node-addon-api. The repository has 45 watchers, 448 stars, and 102 forks. The main navigation bar includes Code, Issues (68), Pull requests (13), Actions, Projects (0), Wiki, and Insights. A search bar with the query "is:pr is:open" is present, along with filters for Labels (12) and Milestones (0). A green "New pull request" button is located on the right. The pull request list shows 13 open pull requests. One specific pull request, "#442 Implement ThreadSafeFunction class", is highlighted with a red box. This pull request was opened on 5 Feb by romandev and has 49 comments. Other visible pull requests include "Fix link", "Use IsEmpty to check for empty value", "doc: Add contribution philosophy doc", "src: fix ObjectWrap destruction", "Little fix on code example", "[WIP][do not merge] Illustrate osx crash", "[WIP]: src: implement `NodeThreadScheduler`", "src : implement Cleanup class", "src: implement AsyncPromise class", "Introduce test for basic type string", "Make TypedArrayOf take both types as template parameters", and "Rewrite constexpr with template metaprogram".

PR	Description	Comments
#481	Fix link	2
#478	Use IsEmpty to check for empty value	1
#477	doc: Add contribution philosophy doc	9
#475	src: fix ObjectWrap destruction	4
#470	Little fix on code example	1
#456	[WIP][do not merge] Illustrate osx crash	1
<b>#442</b>	<b>Implement ThreadSafeFunction class</b>	<b>49</b>
#424	[WIP]: src: implement `NodeThreadScheduler`	4
#397	src : implement Cleanup class	25
#377	src: implement AsyncPromise class	12
#330	Introduce test for basic type string	9
#128	Make TypedArrayOf take both types as template parameters	4
#127	Rewrite constexpr with template metaprogram	13

# ObjectWrap API

- **ObjectWrap** is a way to expose your C++ code to JavaScript
- You have to extend **ObjectWrap** class that includes the plumbing to connect JavaScript code to a C++ object
- Classes extending **ObjectWrap** can be instantiated from JavaScript using the **new** operator, and their methods can be **directly invoked from JavaScript**
- Unfortunately, the **wrap** part really refers to a way to group methods and state
- **It's your responsibility write custom code to bridge** each of your C++ class methods.

# Example

# Thanks



Don't be afraid sometimes it's a good thing dirty your hands  
with a little of C++

All examples and materials are here

<https://github.com/NickNaso/node-overview>