

ROME - APRIL 13/14 2018

{codemotion}

{ Node.js Native AddOns from zero to hero }

Nicola Del Gobbo



Who is Nicola Del Gobbo?

Developer

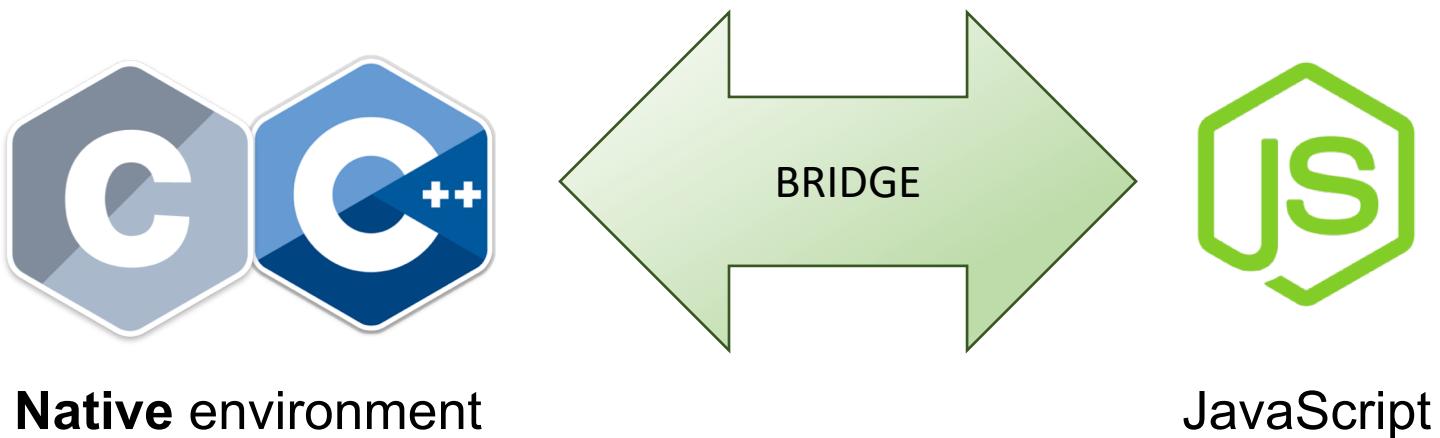


open source

N-API TEAM

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.*

How is it possible?

All the **magic** is behind the **Node.js** architecture

Node.js API

Node.js Bindings

C / C++ Addons



c-ares

HTTP
parser

Open
SSL

zlib

nghttp2

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

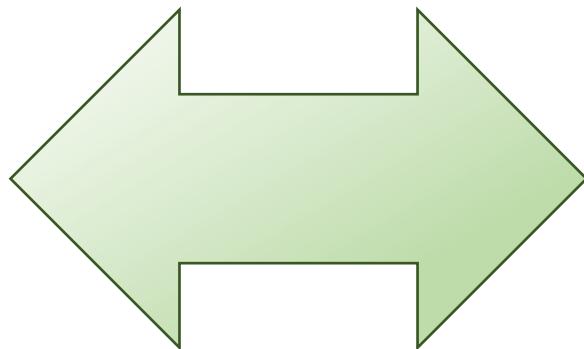
Why?

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;
}
```

C/C++



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

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

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

Problems

Write portable C / C++ code

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

ARCHITECTURE

PLATFORM

COMPILER

Problems

Documentation

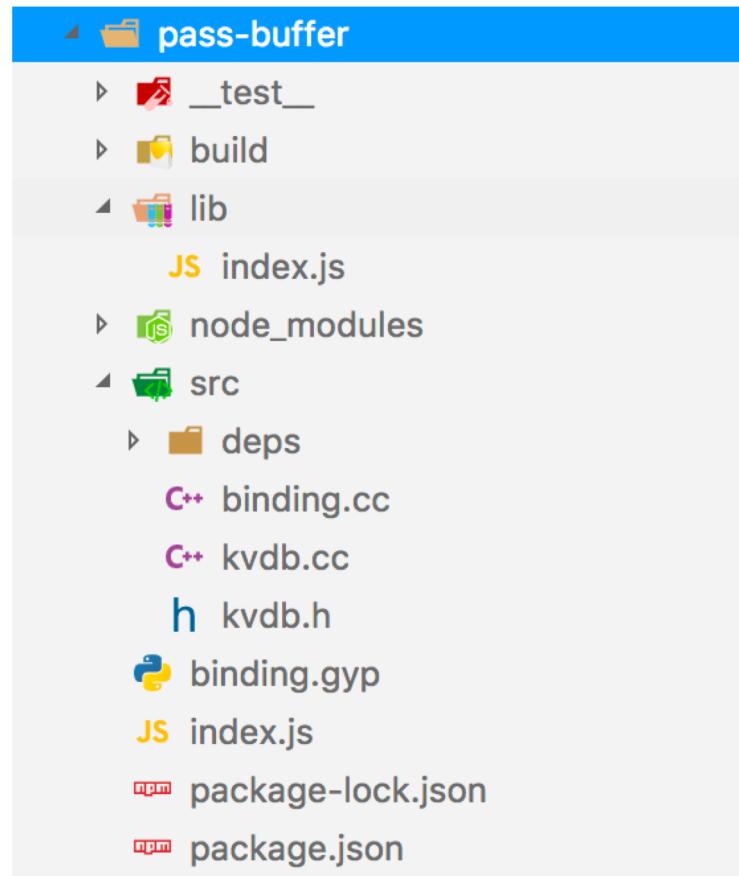
- **C / C++** libraries that you are integrating are not well documented
- There are good **references** but not so much practical guide focusing on complex concepts about native add-ons

N-API

N-API will be a game changer on the native add-on development

- **API** and **ABI** compatibility
- **Isolated** from **V8** (VM agnostic)
- New **ES6** types
- C / C++ (only header wrapper)
- **Conversion** tool that helps to migrate from **NAN**
- **Generator** (help with initial scaffolding)
- Pre-builds (**node-pre-gyp** - **prebuildify**)

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

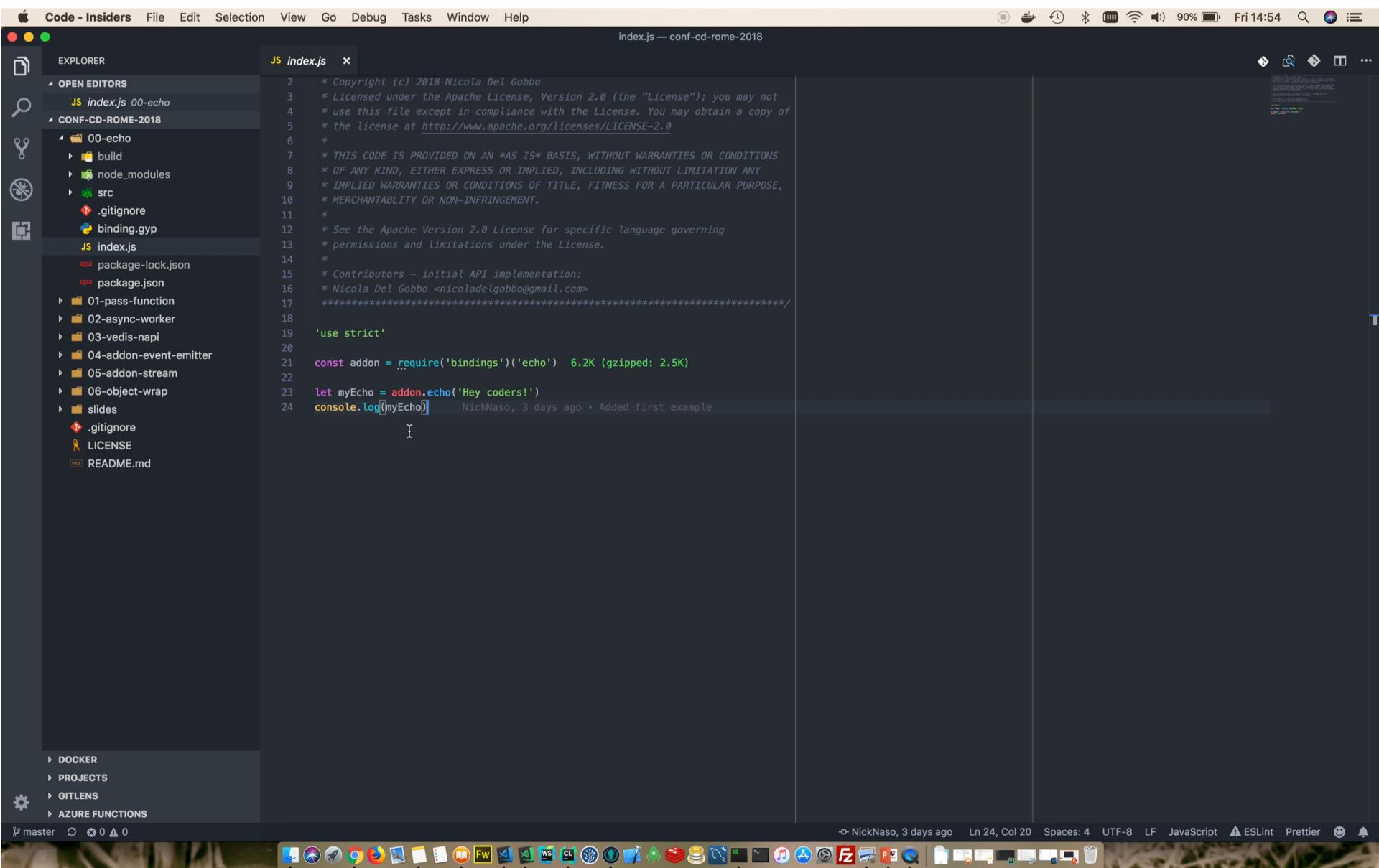
    if (info[0].IsString()) {
        std::string in = info[0].As<Napi::String>();
        return Napi::String::New(env, in);
    } else {
        throw Napi::Error::New(env, "The argument must be a string");
    }
}
```

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);
```



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()
}
```

Call a callback and return the result back

```
cb.Call({env.Null(), Napi::String::New(env, in)});
```

```
}
```

QuickTime Player File Edit View Window Help

index.js — conf-cd-rome-2018

EXPLORER

OPEN EDITORS

JS index.js 01-pass-function

CONF-CD-ROME-2018

- 00-echo
- 01-pass-function
- build
- node_modules
- src
 - .gitignore
 - binding.gyp
- JS index.js
- package-lock.json
- package.json

- 02-async-worker
- 03-vedis-napi
- 04-addon-event-emitter
- 05-addon-stream
- 06-object-wrap
- slides
- .gitignore
- LICENSE
- README.md

DOCKER

PROJECTS

GITLENS

AZURE FUNCTIONS

1

```
1  /*****************************************************************************  
2  * Copyright (c) 2018 Nicola Del Gobbo  
3  * Licensed under the Apache License, Version 2.0 (the "License"); you may not  
4  * use this file except in compliance with the License. You may obtain a copy of  
5  * the license at http://www.apache.org/licenses/LICENSE-2.0  
6  *  
7  * THIS CODE IS PROVIDED ON AN *AS IS* BASIS, WITHOUT WARRANTIES OR CONDITIONS  
8  * OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION ANY  
9  * IMPLIED WARRANTIES OR CONDITIONS OF TITLE, FITNESS FOR A PARTICULAR PURPOSE,  
10 * MERCHANTABILITY OR NON-INFRINGEMENT.  
11 *  
12 * See the Apache Version 2.0 License for specific language governing  
13 * permissions and limitations under the License.  
14 *  
15 * Contributors - initial API implementation:  
16 * Nicola Del Gobbo <nicoladelgobbo@gmail.com>  
*****  
17  
18  
19 'use strict'  
20  
21 const addon = require('bindings')('echo') 6.2K (gzipped: 2.5K)  
22  
Complexity is 3 Everything is cool  
23 function callback(err, data) {  
24     if (err) {  
25         console.error(err)  
26     } else {  
27         console.log(data)  
28     }  
29 }  
30  
31 addon.echo('Hey coders!', callback)  
32  
33 console.log('My next code ...') NickNaso, 3 days ago * Pass function example
```

NickNaso, 3 days ago Ln 33, Col 32 Spaces: 4 UTF-8 LF JavaScript ▲ ESLint Prettier

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();  
}
```

QuickTime Player File Edit View Window Help

index.js — conf-cd-rome-2018

EXPLORER

- OPEN EDITORS
- JS index.js 02-async-worker
- CONF-CD-ROME-2018
 - 00-echo
 - 01-pass-function
 - 02-async-worker
 - build
 - node_modules
 - src
 - .gitignore
 - binding.gyp
 - index.js
 - package-lock.json
 - package.json
 - 03-vedis-napi
 - 04-addon-event-emitter
 - 05-addon-stream
 - 06-object-wrap
 - slides
 - .gitignore
 - LICENSE
 - README.md
- DOCKER
- PROJECTS
- GITLENS
- AZURE FUNCTIONS

1

JS index.js

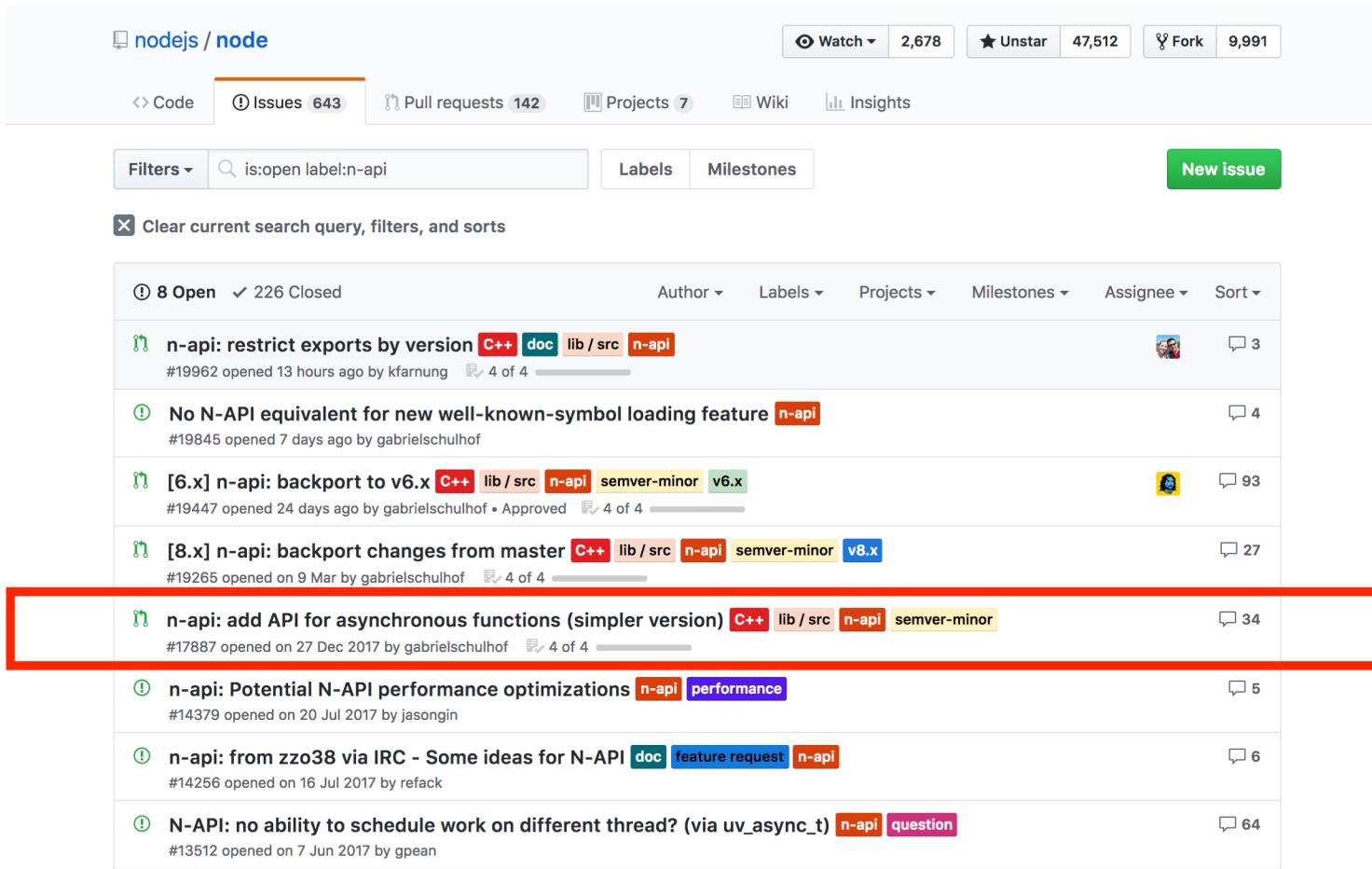
```
2  * Copyright (c) 2018 Nicola Del Gobbo
3  * Licensed under the Apache License, Version 2.0 (the "License"); you may not
4  * use this file except in compliance with the License. You may obtain a copy of
5  * the license at http://www.apache.org/licenses/LICENSE-2.0
6  *
7  * THIS CODE IS PROVIDED ON AN *AS IS* BASIS, WITHOUT WARRANTIES OR CONDITIONS
8  * OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION ANY
9  * IMPLIED WARRANTIES OR CONDITIONS OF TITLE, FITNESS FOR A PARTICULAR PURPOSE,
10 * MERCHANTABILITY OR NON-INFRINGEMENT.
11 *
12 * See the Apache Version 2.0 License for specific language governing
13 * permissions and limitations under the License.
14 *
15 * Contributors - initial API implementation:
16 * Nicola Del Gobbo <nicoladelgobbo@gmail.com>
17 ****
18
19 'use strict'
20
21 const addon = require('bindings')('echo') 6.2K (gzipped: 2.5K)
22
23 Complexity is 3 Everything is cool!
24 function callback(err, data) {
25   if (err) {
26     console.error(err)
27   } else {
28     console.log(data)
29   }
30
31 addon.echo('Hey coders!', callback)
32
33 console.log('My next code ...')
```

NickNaso, 3 days ago · Change folders and add async worker examples

Ln 27, Col 21 Spaces: 4 UTF-8 LF JavaScript ▲ ESLint Prettier

AsyncWorker

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



The screenshot shows the GitHub repository for Node.js. The 'Issues' tab is selected, displaying 643 open issues. A search bar at the top is set to 'is:open label:n-api'. The main list of issues includes:

- n-api: restrict exports by version** (C++ doc lib / src n-api) #19962 opened 13 hours ago by kfarnung
- No N-API equivalent for new well-known-symbol loading feature** (n-api) #19845 opened 7 days ago by gabrielschulhof
- [6.x] n-api: backport to v6.x** (C++ lib / src n-api semver-minor v6.x) #19447 opened 24 days ago by gabrielschulhof • Approved
- [8.x] n-api: backport changes from master** (C++ lib / src n-api semver-minor v8.x) #19265 opened on 9 Mar by gabrielschulhof
- n-api: add API for asynchronous functions (simpler version)** (C++ lib / src n-api semver-minor) #17887 opened on 27 Dec 2017 by gabrielschulhof
- n-api: Potential N-API performance optimizations** (n-api performance) #14379 opened on 20 Jul 2017 by jasongin
- n-api: from zzo38 via IRC - Some ideas for N-API** (doc feature request n-api) #14256 opened on 16 Jul 2017 by refack
- N-API: no ability to schedule work on different thread? (via uv_async_t)** (n-api question) #13512 opened on 7 Jun 2017 by gpean

The issue #17887 is highlighted with a red box. It is a pull request for adding an API for asynchronous functions, with labels C++, lib / src, n-api, and semver-minor, and version v8.x. It was opened on 27 Dec 2017 by gabrielschulhof and has 34 comments.

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.

Event Emitter

Much of the Node.js core API modules are built around an idiomatic asynchronous event-driven architecture in which certain kinds of objects (called **emitter**) periodically emit named events that cause Function objects ("listeners") to be called.

- Emit event from **C++**
- Implement a native add-on object that inherits from Event Emitter



! Correct way to use EventEmitter and emit events from addon

#110 opened on 17 Aug 2017 by bennycooly



5

QuickTime Player File Edit View Window Help

index.js — conf-cd-rome-2018

EXPLORER

OPEN EDITORS

JS index.js 04-addon-event-emitter/00

CONF-CD-ROME-2018

- 00-echo
- 01-pass-function
- 02-async-worker
- 03-vedis-napi
- 04-addon-event-emitter
 - 00
 - build
 - node_modules
 - src
 - binding.gyp
 - JS index.js
 - package-lock.json
 - package.json
- 01
 - .gitignore
 - LICENSE
 - README.md
- 05-addon-stream
- slides
 - .gitignore
 - LICENSE
 - README.md

DOCKER

PROJECTS

GITLENS

AZURE FUNCTIONS

JS index.js

```
1  ****
2  * Copyright (c) 2018 Nicola Del Gobbo
3  * Licensed under the Apache License, Version 2.0 (the "License"); you may not
4  * use this file except in compliance with the License. You may obtain a copy of
5  * the license at http://www.apache.org/licenses/LICENSE-2.0
6  *
7  * THIS CODE IS PROVIDED ON AN *AS IS* BASIS, WITHOUT WARRANTIES OR CONDITIONS
8  * OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION ANY
9  * IMPLIED WARRANTIES OR CONDITIONS OF TITLE, FITNESS FOR A PARTICULAR PURPOSE,
10 * MERCHANTABILITY OR NON-INFRINGEMENT.
11 *
12 * See the Apache Version 2.0 License for specific language governing
13 * permissions and limitations under the License.
14 *
15 * Contributors - initial API implementation:
16 * Nicola Del Gobbo <nicoladegobbo@gmail.com>
17 ****
18
19 'use strict'
20
21 const EventEmitter = require('events').EventEmitter
22 const addon = require('bindings')('emit_from_cpp') 6.2K (gzipped: 2.5K)
23
24 const emitter = new EventEmitter()
25
26 emitter.on('start', () => {
27   console.log('### START ...')
28 })
29 emitter.on('data', (evt) => {
30   console.log(evt);
31 })
32
33 emitter.on('end', () => {
34   console.log('### END ###')
35 })
36
37 addon.callEmit(emitter.emit.bind(emitter))
```

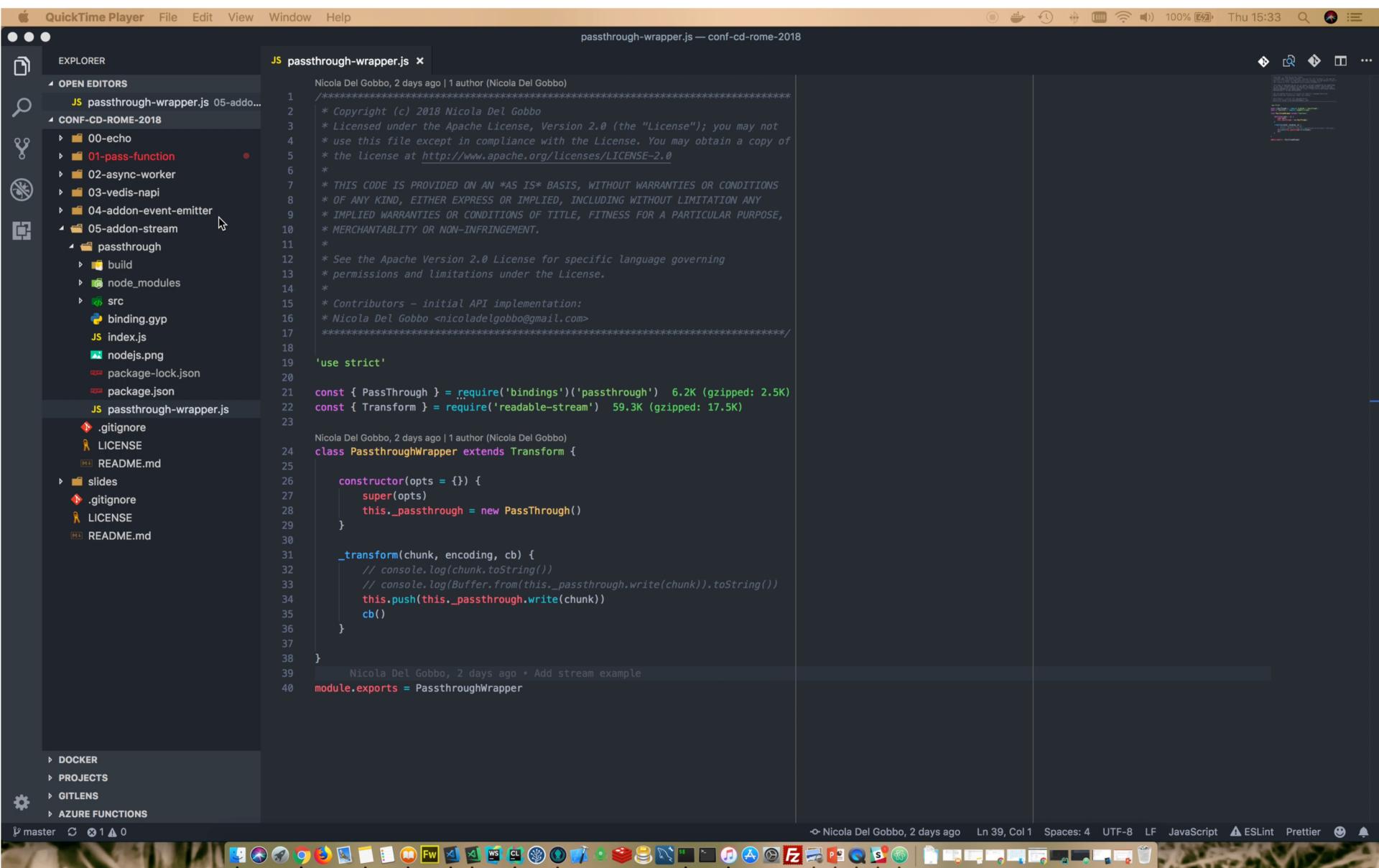
Nicola Del Gobbo, 2 days ago • Added event emitter examples

Nicola Del Gobbo, 2 days ago Ln 37, Col 43 Spaces: 4 UTF-8 LF JavaScript ▲ ESLint Prettier

Stream

A stream is an abstract interface for working with streaming data in Node.js. The stream module provides a base API that makes it easy to build objects that implement the stream interface.

Using **Transform** stream to pass and get back data from a native add-on



Lessons learned

- Create **simple interface** between C / C++ and JavaScript
- Think **carefully at JavaScript** interface
- Use **ObjectWrap** API to return C / C++ object to JavaScript
- Validate the **types** on the native side
- Expose **only the functionalities you need**
- **Don't block the event loop**, stay **asynchronous**: this is a must if you will use the add-ons in high performance services
- Use **Buffer** to pass big quantity of data

Present and future

- The largest number of native add-ons is written using **NAN**
- N-API will be go out of experimental very soon
- Backports for 8.x
- Backports for 6.x
- Documentation for **node-addon-api**
- Porting preexistentes native addon-ons to N-API
- Investigate how to use prebuild with N-API

Implement key value database

Binding to **Vedis** an embeddable datastore **C** library



```
const { Database } = require('bindings')('kvdb')

const mydb = new Database('test', './tmp-sync')
console.log(mydb.db_name)

mydb.putKeySync("username", "NickNaso");
console.log(mydb.getKeySync("username"));
```

```
mydb.putKey('password', 'keeplooking', function (err) {
  if (err) {
    console.error(err)
  } else {
    console.log("Value stored")
  }
})

mydb.getKey('username', function(err, value) {
  if (err) {
    console.error(err)
  } else {
    console.log('Value from my async API')
    console.log(value)
  }
})
```

Thanks!



nicoladelgobbo@gmail.com

[@NickNaso](https://twitter.com/NickNaso) on Twitter

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/conf-cd-rome-2018>