Building Node.js addons like it’s 2023

About Michael

Node.js lead for Red Hat and IBM

Active Node.js community member

- Node.js Collaborator, Node.js Technical Steering Committee,
- Active in a number of Working group(s)

Active OpenJS Foundation member

- Voting Cross Project Council Member
- Community Director 2020-2022

Twitter: @mhdawson1

GitHub: @mhdawson

Linkedin: https://www.linkedin.com/in/michael-dawson-6051282

Agenda

- What are native addons, why do we need them
- Alternatives
- Building C/C++ addons
- Node-api/Node-addon-api - why so great ?
- Node-api with other languages
- Resources
- Node-api team, get involved

What are native addons

- What
  - JavaScript functions, objects, etc. that are not actually written in JavaScript !
- Why
  - **Re-use:** Lots of existing code written in other languages
    - sharp, bcrypt, sqlite3, etc.
  - **Speed:** Some things run faster in other languages
  - **Access:** Some resources are not available from JavaScript natively
    - serialport

Alternatives

- Some alternatives
  - Foreign Function Interface - FFI
    - Node-ffi-napi - https://www.npmjs.com/package/ffi-napi
    - Foreign Function Interface (FFI) implementation #46905 (https://github.com/nodejs/node/pull/46905)
  - WASM
- Alternative don’t cover all use cases
  - FFI allows JavaScript to native calls
  - Add-ons allows both JavaScript to native code vice versa as well as complex data types
  - WASM allows JavaScript to native and vice versa, but boundary to JavaScript and environment is more limited.

2014

Nan v1.0 ~ 5 years after Node.js is created

Building C/C++ addons - V8/nan

```
1   #include <nan.h>
2
3   void Method(const Nan::FunctionCallbackInfo<v8::Value>& info) {
4     info.GetReturnValue().Set(Nan::New("world").ToLocalChecked());
5   }
6
7   void Init(v8::Local<v8::Object> exports) {
8     v8::Local<v8::Context> context = exports->CreationContext();
9     exports->Set(context,
10                  Nan::New("hello").ToLocalChecked(),
11                  Nan::New<v8::FunctionTemplate>(Method)
12                      ->GetFunction(context)
13                      .ToLocalChecked());
14  }
15
16  NODE_MODULE(hello, Init)
```

```
1   var addon = require('bindings')('hello');
2
3   console.log(addon.hello()); // 'world'
```

Building C/C++ addons - V8/nan

```
1   #include <nan.h>
2
3   void Method(const Nan::FunctionCallbackInfo<v8::Value>& info) {
4     info.GetReturnValue().Set(Nan::New("world").ToLocalChecked());
5   }
6
7   void Init(v8::Local<v8::Object> exports) {
8     v8::Local<v8::Context> context = exports->CreationContext();
9     exports->Set(context,
10                  Nan::New("hello").ToLocalChecked(),
11                  Nan::New<v8::FunctionTemplate>(Method)
12                      ->GetFunction(context)
13                      .ToLocalChecked());
14  }
15
16  NODE_MODULE(hello, Init)
```

Building C/C++ addons - V8/nan

```
1   #include <nan.h>
2
3   void Method(const Nan::FunctionCallbackInfo<v8::Value>& info) {
4     info.GetReturnValue().Set(Nan::New("world").ToLocalChecked());
5   }
6
7   void Init(v8::Local<v8::Object> exports) {
8     v8::Local<v8::Context> context = exports->CreationContext();
9     exports->Set(context,
10                  Nan::New("hello").ToLocalChecked(),
11                  Nan::New<v8::FunctionTemplate>(Method)
12                      ->GetFunction(context)
13                      .ToLocalChecked());
14  }
15
16  NODE_MODULE(hello, Init)
```

Running C/C++ addons - V8/nan

```
[user1@fedora nan]$ source ../../../set16.sh
[user1@fedora nan]$ node --version
v16.20.0
[user1@fedora nan]$ node hello.js
world
[user1@fedora nan]$ source ../../../set18.sh
[user1@fedora nan]$ node hello.js
/home/user1/addon-talk/node-addon-examples/1_hello_world/nan/node_modules/bindings/bindings.js:83
        throw e
        ^

Error: The module '/home/user1/addon-talk/node-addon-examples/1_hello_world/nan/build/Release/hello.node'
was compiled against a different Node.js version using
NODE_MODULE_VERSION 93. This version of Node.js requires
NODE_MODULE_VERSION 108. Please try re-compiling or re-installing
the module (for instance, using `npm rebuild` or `npm install`).
    at Module._extensions..node (node:internal/modules/cjs/loader:1338:18)
    at Module.load (node:internal/modules/cjs/loader:1117:32)
    at Module._load (node:internal/modules/cjs/loader:958:12)
    at Module.require (node:internal/modules/cjs/loader:1141:19)
    at require (node:internal/modules/cjs/helpers:110:18)
    at bindings (/home/user1/addon-talk/node-addon-examples/1_hello_world/nan/node_modules/bindings/bindings.js:76:44)
    at Object.<anonymous> (/home/user1/addon-talk/node-addon-examples/1_hello_world/nan/hello.js:1:32)
    at Module._compile (node:internal/modules/cjs/loader:1254:14)
    at Module._extensions..js (node:internal/modules/cjs/loader:1308:10)
    at Module.load (node:internal/modules/cjs/loader:1117:32) {
  code: 'ERR_DLOPEN_FAILED'
}

Node.js v18.16.0
```

2018

node-api - stable in 8.12.0

80% use case

Building C/C++ addons - node-api

```c
#include <assert.h>
#include <node_api.h>

static napi_value Method(napi_env env, napi_callback_info info) {
  napi_status status;
  napi_value world;
  status = napi_create_string_utf8(env, "world", 5, &world);
  assert(status == napi_ok);
  return world;
}

#define DECLARE_NAPI_METHOD(name, func)                                          \
  { name, 0, func, 0, 0, 0, napi_default, 0 }

static napi_value Init(napi_env env, napi_value exports) {
  napi_status status;
  napi_property_descriptor desc = DECLARE_NAPI_METHOD("hello", Method);
  status = napi_define_properties(env, exports, 1, &desc);
  assert(status == napi_ok);
  return exports;
}

NAPI_MODULE(NODE_GYP_MODULE_NAME, Init)
```

```bash
var addon = require('bindings')('hello');

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

Running C/C++ addons - node-api

```
[user1@fedora napi]$ source ../../../set16.sh
[user1@fedora napi]$ node --version
v16.20.0
[user1@fedora napi]$ node hello.js
world
[user1@fedora napi]$ source ../../../set18.sh
[user1@fedora napi]$ node --version
v18.16.0
[user1@fedora napi]$ node hello.js
world
[user1@fedora napi]$
```

Building C/C++ addons - node-addon-api

```cpp
#include <napi.h>

Napi::String Method(const Napi::CallbackInfo& info) {
  Napi::Env env = info.Env();
  return Napi::String::New(env, "world");
}

Napi::Object Init(Napi::Env env, Napi::Object exports) {
  exports.Set(Napi::String::New(env, "hello"),
              Napi::Function::New(env, Method));
  return exports;
}

NODE_API_MODULE(hello, Init)
```

```bash
var addon = require('bindings')('hello');

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

Node-api/Node-addon-api - why so great ?

- ABI stable
  - No code changes needed for new Node.js versions
  - Build once, run with later Node.js versions
- Concepts and operations generally map to ideas specified in the ECMA262 Language Specification.
  - Cross runtime compatibility
  - Cross language support
  - Separation from Node.js itself

Cross Runtime Compatibility

+

Node-API bindings for other runtimes

Project	Programming language
bun	Zig
deno	Rust
electron	C++
emnapi	C / JavaScript
iotjs	C
veil	C

Separation from Node.js - Headers

Product Solutions Open Source Pricing Search / Sign in Sign u

nodejs / node-api-headers Public

Notifications Fork 12 Sta

Code Issues 1 Pull requests 1 Actions Projects Security Insights

main 1 branch 5 tags Go to file Code

github-actions[bot] Update headers from nodejs/node tag v19.9.0 (#24) ··· 1a32803 5 days ago 25 commits

.github/workflows	Use git status instead of git diff for change calculation (#21)	2 weeks ago
def	Update headers from nodejs/node tag v19.8.1 (#22)	2 weeks ago
include	Update headers from nodejs/node tag v19.9.0 (#24)	5 days ago
scripts	fix: moved def files on a proper folder. (#19)	3 weeks ago
.npmignore	Add helper scripts for updating headers and symbols.js (#7)	2 months ago
CHANGELOG.md	release: v0.0.5.	2 weeks ago
CODE_OF_CONDUCT.md	Initial scaffolding for the project.	2 years ago

About

Repository used to make the N-API headers more accessible

Readme
MIT license
Code of conduct
Security policy
22 stars
18 watching
12 forks
Report repository

https://github.com/nodejs/node-api-headers

Separation from Node.js - Tests - NOT there yet !

Product Solutions Open Source Pricing Search / Sign in Sign up

nodejs / js-native-api-test Public

Notifications Fork 10 Star 7

Code Issues 3 Pull requests Actions Projects Security Insights

main 2 branches 0 tags Go to file Code

mhdawson doc: add in base governance (#8) ··· 6fbd941 on May 19, 2021 4 commits

.github	doc: add in base governance (#8)	2 years ago
CODE_OF_CONDUCT.md	doc: add in base governance (#8)	2 years ago
CONTRIBUTING.md	doc: add in base governance (#8)	2 years ago
LICENSE.md	doc: add in base governance (#8)	2 years ago
README.md	Update to reflect js-native-api subset	2 years ago

README.md

js-native-api-test

Test suite for JavaScript Engine specific portion of Node-api.

About

Node.js js-native-api-test suite, externalized so that it can be used to test Node-API implementations outside of Node.js

Readme
MIT license
Code of conduct
Security policy
7 stars
21 watching
10 forks
Report repository

Releases

https://github.com/nodejs/js-native-api-test

Cross language support

Node-API bindings for other languages

Project	Programming language
napi-rs	Rust
napi-sys	Rust
nodejs-sys	Rust
neon	Rust

+

Early stage

Project	Programming language
napi-cs	C#
node-api-dotnet	C#
swift-napi-bindings	Swift
swift-node-addon-examples	Swift
napi-nim	Nim
zig-napigen	Zig
zig-nodejs-example	Zig
go-node-api	Go

Rust Example - neon

https://neon-bindings.com/docs/introduction

```
[user1@fedora rust]$ cat src/lib.rs
use neon::prelude::*;

fn hello(mut cx: FunctionContext) -> JsResult<JsString> {
    Ok(cx.string("world"))
}

#[neon::main]
fn main(mut cx: ModuleContext) -> NeonResult<()> {
    cx.export_function("hello", hello)?;
    Ok(())
}
```

```
var addon = require('bindings')('hello');

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

Rust Example - neon

```
{
  "name": "hello_world",
  "version": "0.1.0",
  "description": "",
  "main": "index.node",
  "scripts": {
    "build": "cargo-cp-artifact -nc build/hello.node -- cargo build --message-format=json-render-diagnostics",
    "build-debug": "npm run build --",
    "build-release": "npm run build -- --release",
    "install": "npm run build-release",
    "test": "cargo test"
  },
  "author": "",
  "license": "MIT",
  "devDependencies": {
    "cargo-cp-artifact": "^0.1"
  },
  "dependencies": {
    "bindings": "^1.5.0"
  }
}
```

Rust Example - neon

```
{
  "name": "hello_world",
  "version": "0.1.0",
  "description": "",
  "main": "index.node",
  "scripts": {
    "build": "cargo-cp-artifact -nc build/hello.node -- cargo build --message-format=json-render-diagnostics",
    "build-debug": "npm run build --",
    "build-release": "npm run build -- --release",
    "install": "npm run build-release",
    "test": "cargo test"
  },
  "author": "",
  "license": "MIT",
  "devDependencies": {
    "cargo-cp-artifact": "^0.1"
  },
  "dependencies": {
    "bindings": "^1.5.0"
  }
}
```

```
[user1@fedora rust]$ tree
.
├── Cargo.lock
├── Cargo.toml
├── hello.js
├── hello.rs
├── package.json
├── package-lock.json
└── src
    └── lib.rs

1 directory, 7 files
```

Rust Example - neon

Writing JS in Rust

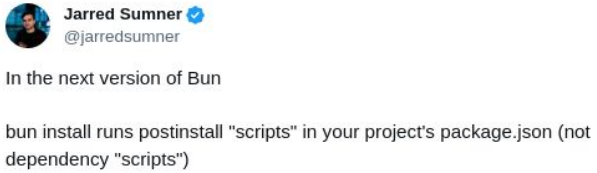
```rust
use neon::prelude::*;

fn hello(mut cx: FunctionContext) -> JsResult<JsObject> {
    let arg1: Handle<JsValue> = cx.argument(0)?;
    let obj: Handle<JsObject> = cx.empty_object();
    let answer: Handle<JsString> = cx.string("world");
    obj.set(&mut cx, "answer", answer)?;
    obj.set(&mut cx, "origarg", arg1)?;
    Ok(obj)
}

#[neon::main]
fn main(mut cx: ModuleContext) -> NeonResult<()> {
    cx.export_function("hello", hello)?;
    Ok(())
}
```

Cross runtime support - example - Bun

- Needed to do npm install, bun install not enough, but may change



- Updated to not use use bindings package
- Worked for both node-addon-api and rust built addon

```
[user1@fedora node-addon-api]$ cat hello.js
//var addon = require('bindings')('hello');
var addon = require('./build/Release/hello.node');

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

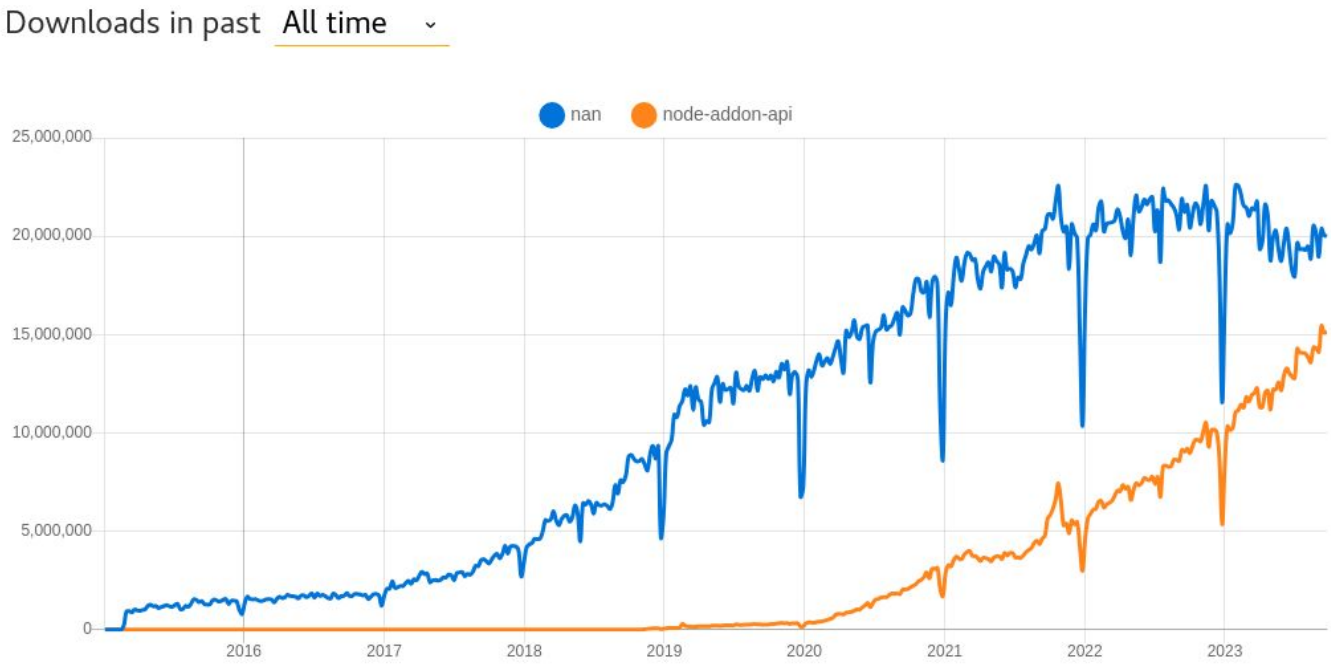
```
[user1@fedora node-addon-api]$ node hello.js
world
[user1@fedora node-addon-api]$ bun hello.js
world
```

2023

Building addons like its 2023

- ABI stable
  - No code changes needed for new versions of Node.js
  - Build once for many versions of Node.js
    - Important for pre-built binaries
- Write in different languages
- Run with different runtimes
- Targets 80% use case, hit ~~38%~~ 43% so far

Building addons like its 2023



80% use case
~ 43% use

https://npmtrends.com/nan-vs-node-addon-api

Resources

- Node.js API Doc - Node-API | Node.js v19.9.0 Documentation
- Node-addon-api Doc - https://github.com/nodejs/node-addon-api#api
- Node-addon-examples - https://github.com/nodejs/node-addon-examples
- Node API resource - http://nodejs.github.io/node-addon-examples/
- Engine Bindings - https://github.com/nodejs/abi-stable-node/blob/doc/node-api-engine-bindings.md

Welcome to the Node-API Resource

The goal of this site is to be a clearinghouse for everything related Node's ABI-Stable C/C++ API, Node-API.

If you are looking to make existing C/C++ code accessible to the widening universe of JavaScript projects, or if you have need to access operating system's resources from JavaScript, or if you have a particularly computationally intensive task that would benefit from hand-tuned C/C++ code accessible from JavaScript, Node-API may be a good fit for you.

This site is maintained by members of the Node.js Node-API team. Please let us know if there is anything we can do to answer your questions or make this site better. We welcome your feedback through the link available at the bottom of each page.

Getting Involved - node-api team

Contributors 103


NickNaso


JckXia


KevinEady


legendecas


mhdawson


vmoroz


gabrielschulhof

Getting Involved - node-api team

Node.js Project Calendar

Today | April 2023 | Print | W

Sun	Mon	Tue	Wed	Thu	Fri
26	27	28 **6pm** Loaders Team Meeting **7pm** Package Maintenance Team	29 **2pm** Node.js Next 10 years **4pm** Node.js TSC Meeting **6pm** Node.js uvwasi team meeting	30 **2pm** Security-WG meeting	31 **3pm** Node.js N-API team weekly m
2	3 **5pm** Node.js Performance Team M	4 **3:30pm** Diagnostics WG Meeting	5	6 **2pm** Node.js Release Working Gro	7 **3pm** Node.js N-API team weekly m
9	10	11 **12pm** Build WG Meeting - early **6pm** Loaders Team Meeting	12 **1pm** Node.js TSC Meeting **2pm** Node.js Next 10 years **6pm** Node.js uvwasi team meeting	13 **2pm** Security-WG meeting **5pm** Package Maintenance Team	14 **3pm** Node.js N-API team weekly m
16	17 **5pm** Node.js Performance Team M	18	19 **3pm** Node.js TSC Meeting	20	21 **3pm** Node.js N-API team weekly m
23	24	25 **6pm** Loaders Team Meeting **7pm** Package Maintenance Team	26 **2pm** Node.js Next 10 years **6pm** Node.js uvwasi team meeting **7pm** Node.js TSC Meeting	27 **2pm** Security-WG meeting	28 **3pm** Node.js N-API team weekly m
30	May 1 **5pm** Node.js Performance Team M	2 **3:30pm** Diagnostics WG Meeting	3 **1pm** Node.js TSC Meeting **10pm** Build WG Meeting	4 **2pm** Node.js Release Working Gro	5 **3pm** Node.js N-API team weekly m

Copyright and Trademarks