BUILDING NEXT GENERATION ADD-ON MODULES FOR NODE.JS USING N-API

Michael Dawson, IBM

Michael Dawson

IBM Community Lead for Node.js

IBM Runtime Technologies

Node.js collaborator and TSC member

Active in n-api, user-feedback, security-wg, diagnostic, build LTS and benchmarking teams and working groups

Contact me:

michael_dawson@ca.ibm.com

Twitter: @mhdawson1

Github: @mhdawson

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

What is N-API ?

N-API is a **stable Node API layer** for native modules, that **provides ABI compatibility** guarantees **across different Node versions & flavors.**

N-API enables native modules to just work across different versions and flavors of Node.js **without recompilations!**

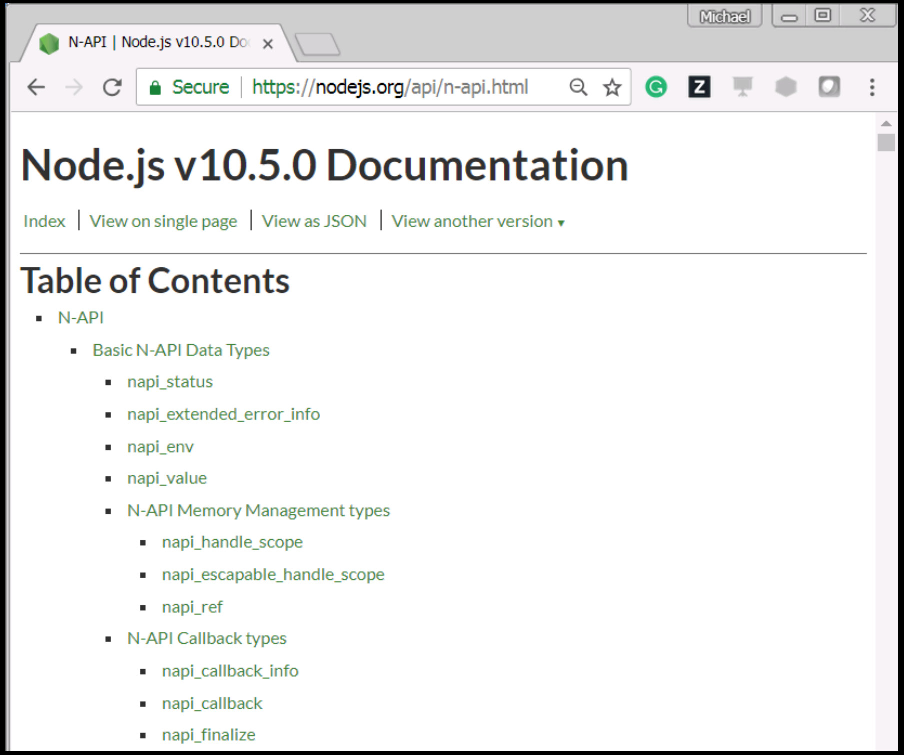
What is N-API ?

- Current Status
  - Exited experimental March 14 - https://github.com/nodejs/node/pull/19262
  - Backported to 6.x, 8.x
    - 6.x – doc's only experimental
    - 8.x - still needs SemVer minor to be stable in 8.x
- Motivational Presentation on Day 1
  - **THE NEXT GENERATION NODE API IS READY! – 2:20 PM DAY 1 (TUESDAY)**

How Do I use N-API?

C

N-API built into Node.js 6.x, 8.x, 10.x etc.
https://nodejs.org/api/n-api.html
#include “node_api.h”



N-API | Node.js v10.5.0 Do

Secure | https://nodejs.org/api/n-api.html

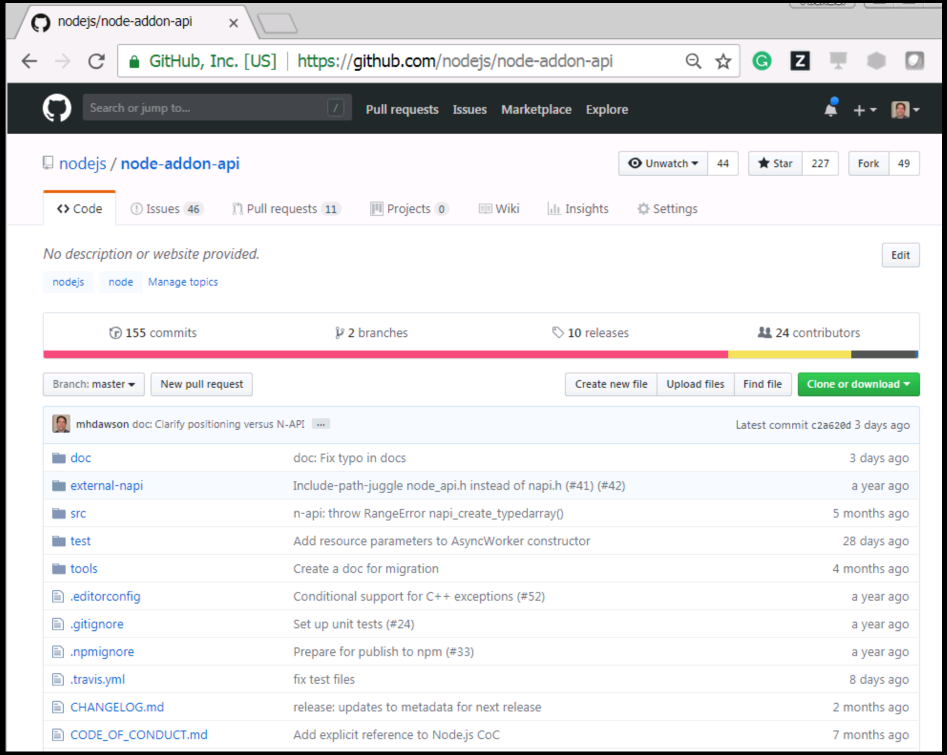
Node.js v10.5.0 Documentation

Index | View on single page | View as JSON | View another version ▾

Table of Contents

C++

npm install node-addon-api
https://github.com/nodejs/node-addon-api
#include “napi.h”



nodejs/node-addon-api

GitHub, Inc. [US] | https://github.com/nodejs/node-addon-api

nodejs / node-addon-api

No description or website provided.

nodejs node Manage topics

155 commits | 2 branches | 10 releases | 24 contributors

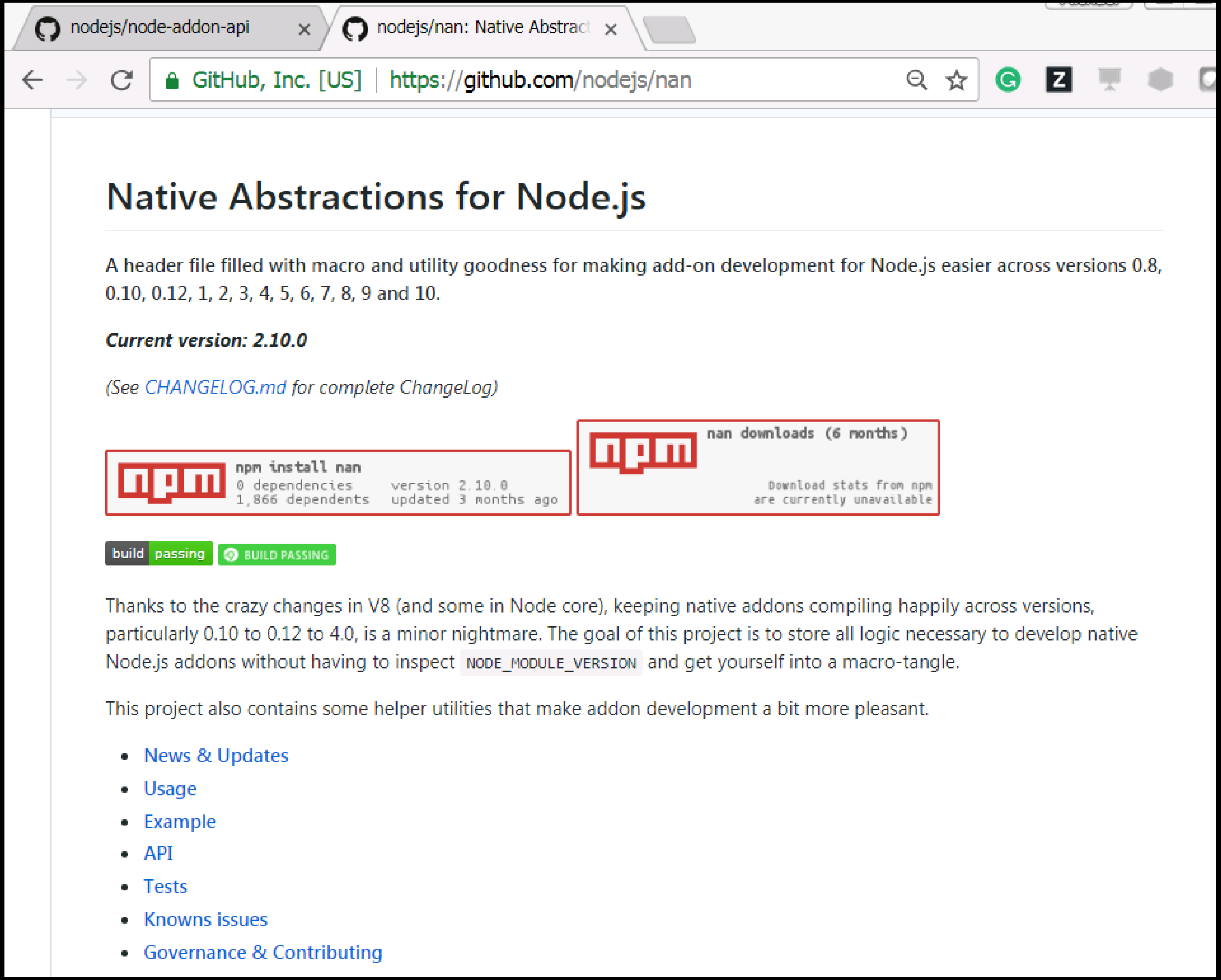
mhdawson doc: Clarify positioning versus N-API … Latest commit c2a620d 3 days ago

doc	doc: Fix typo in docs	3 days ago
external-napi	Include-path-juggle node_api.h instead of napi.h (#41) (#42)	a year ago
src	n-api: throw RangeError napi_create_typedarray()	5 months ago
test	Add resource parameters to AsyncWorker constructor	28 days ago
tools	Create a doc for migration	4 months ago
.editorconfig	Conditional support for C++ exceptions (#52)	a year ago
.gitignore	Set up unit tests (#24)	a year ago
.npmignore	Prepare for publish to npm (#33)	a year ago
.travis.yml	fix test files	8 days ago
CHANGELOG.md	release: updates to metadata for next release	2 months ago
CODE_OF_CONDUCT.md	Add explicit reference to Node.js CoC	7 months ago

https://github.com/nodejs/abi-stable-node-addon-examples

What about NAN ?

- Has done a great job since 0.8 !
- NAN can only provide so much isolation
- Needed new approach
- node-addon-api is successor for NAN
- Transition will take time


nodejs/node-addon-api | nodejs/nan: Native Abstract

GitHub, Inc. [US] | https://github.com/nodejs/nan

Native Abstractions for Node.js

A header file filled with macro and utility goodness for making add-on development for Node.js easier across versions 0.8, 0.10, 0.12, 1, 2, 3, 4, 5, 6, 7, 8, 9 and 10.

Current version: 2.10.0

(See CHANGELOG.md for complete ChangeLog)

npm install nan
0 dependencies version 2.10.0
1,866 dependents updated 3 months ago

nan downloads (6 months)
Download stats from npm are currently unavailable

build passing | BUILD PASSING

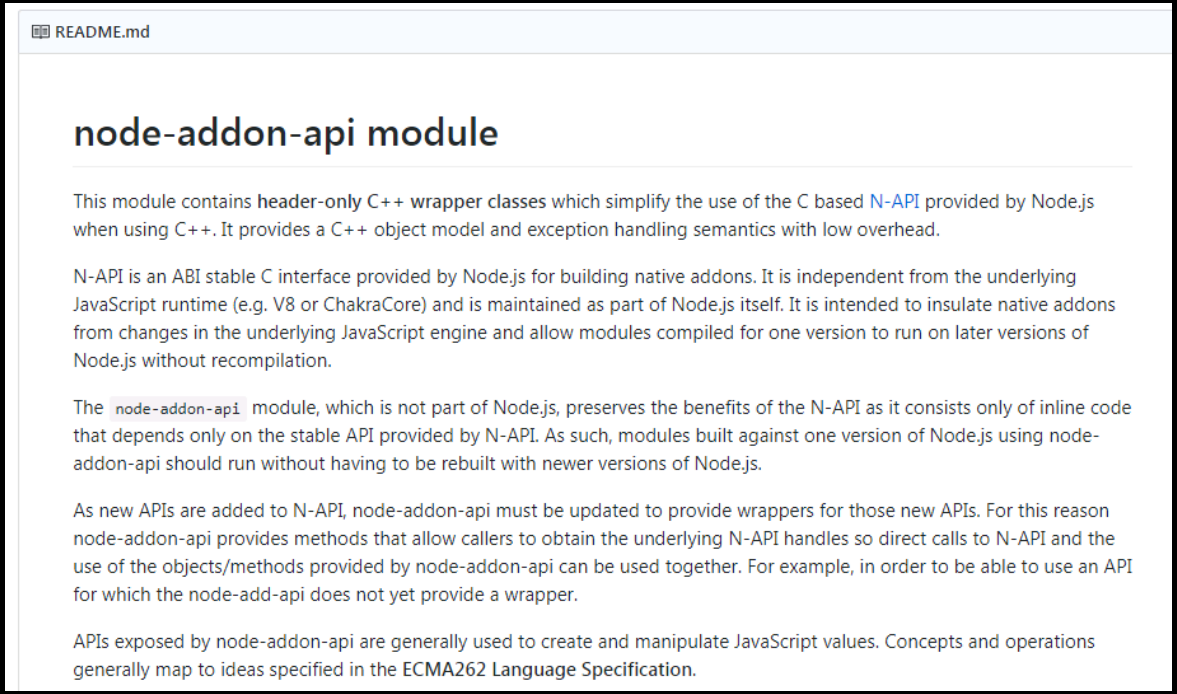
Thanks to the crazy changes in V8 (and some in Node core), keeping native addons compiling happily across versions, particularly 0.10 to 0.12 to 4.0, is a minor nightmare. The goal of this project is to store all logic necessary to develop native Node.js addons without having to inspect `NODE_MODULE_VERSION` and get yourself into a macro-tangle.

This project also contains some helper utilities that make addon development a bit more pleasant.

- News & Updates
- Usage
- Example
- API
- Tests
- Knowns issues
- Governance & Contributing

What is **node-addon-api** ?

- Header Only wrapper
  - Inline only
  - Compiled into module
  - Depends only on exported N-API functions
- Delivered as npm module
- Provides a C++ object model
- Easy transition from NAN



README.md

node-addon-api module

This module contains **header-only C++ wrapper classes** which simplify the use of the C based N-API provided by Node.js when using C++. It provides a C++ object model and exception handling semantics with low overhead.

N-API is an ABI stable C interface provided by Node.js for building native addons. It is independent from the underlying JavaScript runtime (e.g. V8 or ChakraCore) and is maintained as part of Node.js itself. It is intended to insulate native addons from changes in the underlying JavaScript engine and allow modules compiled for one version to run on later versions of Node.js without recompilation.

The `node-addon-api` module, which is not part of Node.js, preserves the benefits of the N-API as it consists only of inline code that depends only on the stable API provided by N-API. As such, modules built against one version of Node.js using node-addon-api should run without having to be rebuilt with newer versions of Node.js.

As new APIs are added to N-API, node-addon-api must be updated to provide wrappers for those new APIs. For this reason node-addon-api provides methods that allow callers to obtain the underlying N-API handles so direct calls to N-API and the use of the objects/methods provided by node-addon-api can be used together. For example, in order to be able to use an API for which the node-add-api does not yet provide a wrapper.

APIs exposed by node-addon-api are generally used to create and manipulate JavaScript values. Concepts and operations generally map to ideas specified in the **ECMA262 Language Specification.**

Core Concepts

- Environment (Env)
- Basic Types
- Calling a Function
- Error Handling
- Object Lifetime Management
- ObjectWrap
- Async Operations

Concepts and operations generally map to ideas specified in the **ECMA262 Language Specification**.

class Env

- Wrapper for napi_env in C N-API
- Env Associated with every Class
  - Need to create new objects
  - Get from Existing node-addon-api object
    - Napi::Env Env() const;
- Access to Globals
- Error Handling checks

```
class Env {
 public:
  Env(napi_env env);

  operator napi_env() const;

  Object Global() const;
  Value Undefined() const;
  Value Null() const;

  bool IsExceptionPending() const;
  Error GetAndClearPendingException();

 };
```

Basic Types

- Value

Wrapper for napi_value in C N-API

From ES2016 - https://www.ecma-international.org/ecma-262/7.0/
A primitive value is a member of one of the following built-in types: **Undefined**, **Null**, **Boolean**, **Number**, **String**, and **Symbol;** an object is a member of the built-in type **Object**;

- Sub Classes
  - Boolean
  - Number
  - Name
    - String
    - Symbol
  - Object
  - +others

CREATION

```
template <typename T>
   static Value From(napi_env env, const T& value);
```

TYPE CHECK

IsXXX (Null, Boolean, Object, etc.)

CAST

```
template <typename T> T As() const;
```

COERCE

```
Boolean ToBoolean() const;
Number ToNumber() const;
String ToString() const;
Object ToObject() const;
```

Calling A Function – JavaScript to Native

- class Function
- class CallbackInfo
  - Provides array of Napi::Value for parameters

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

  // get and check parameters
  if (!info[0].IsNumber()) {
    double arg0 = info[0].As<Napi::Number>().DoubleValue();
  }
  // return void OR Napi::Value OR one of the node-addon-api classes which have Value() method
}

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

Calling A Function – Native to JavaScript

- class Function
  - Value Call(const std::initializer_list<napi_value>& args) const;
  - Value Call(const std::vector<napi_value>& args) const;
  - Value Call(size_t argc, const napi_value* args) const;
  - Value Call(napi_value recv, const std::initializer_list<napi_value>& args) const;
  - Value Call(napi_value recv, const std::vector<napi_value>& args) const;
  - Value Call(napi_value recv, size_t argc, const napi_value* args) const;
- Most node-addon-api classes convert automatically to napi_value

```
void RunCallback(const Napi::CallbackInfo& info) {
  Napi::Env env = info.Env();
  Napi::Function cb = info[0].As<Napi::Function>();
  cb.Call(env.Global(), { Napi::String::New(env, "hello world") });
}
```

Error Handling

- node-addon-api converts N-API errors to Exceptions
- Exception triggered in JavaScript on return from Native
- C++ exceptions enabled
  - `Error` class extends `std::exception`
  - Try/catch to handle exceptions
  - throw Napi::Error::New(env, "Test error");

```
try {
  result = jsFunctionThatThrows({arg1});
  // and so on
} catch (const Napi::Error& e) {
  // most often just return after failure
  return;
}
```

- C++ exceptions not enabled
  - Must check for pending exception using Env
    - bool IsExceptionPending() const;
    - Error GetAndClearPendingException();

```
result = jsFunctionThatThrows({arg1});
if (env.IsExceptionPending()) {
  // most often just return after failure
  return;
}
```

  - Napi::Error::New(env, "Test error").ThrowAsJavaScriptException();

Error Handling

- #define NAPI_DISABLE_CPP_EXCEPTIONS
- #define NAPI_CPP_EXCEPTIONS

- Error
- TypeError
- RangeError

```
{
  'targets': [
    {
      'target_name': 'test2-native',
      'sources': [ 'src/test2.cc' ],
      'include_dirs': ["<!@(node -p \"require('node-addon-api').include\")"],
      'dependencies': ["<!(node -p \"require('node-addon-api').gyp\")"],
      'cflags!': [ '-fno-exceptions' ],
      'cflags_cc!': [ '-fno-exceptions' ],
      'xcode_settings': {
        'GCC_ENABLE_CPP_EXCEPTIONS': 'YES',
        'CLANG_CXX_LIBRARY': 'libc++',
        'MACOSX_DEPLOYMENT_TARGET': '10.7'
      },
      'msvs_settings': {
        'VCCLCompilerTool': { 'ExceptionHandling': 1 },
      }
    }
  ]
}
```

Object Lifetime Management

- Problem

```
for (int i = 0; i < LOOP_MAX; i++) {
  std::string name = std::string("inner-scope") + std::to_string(i);
  Value newValue = String::New(info.Env(), name.c_str());
  // do something with newValue
};
```

- By default Values live until return from native call

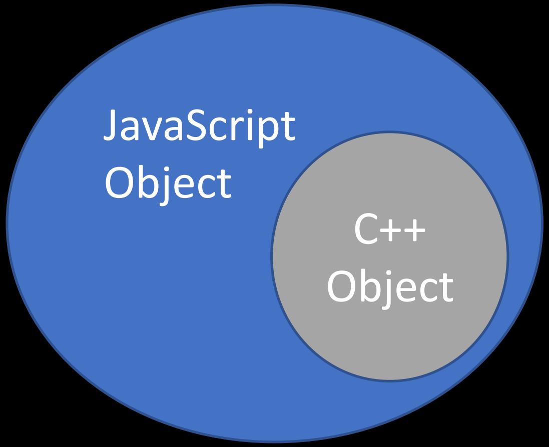
Object Lifetime Management

- Problem

```
for (int i = 0; i < LOOP_MAX; i++) {
  HandleScope scope(info.Env());
  std::string name = std::string("inner-scope") + std::to_string(i);
  Value newValue = String::New(info.Env(), name.c_str());
  // do something with newValue
};
```

- Scopes
  - HandleScope
  - EscapableHandleScope
    - Value Escape(napi_value escapee);
    - Escape can only be called Once !
  - Single Nesting
- Scope already setup
  - Invocation of native method called from JavaScript

Object Wrap



- Extend ObjectWrap

  ```
  Test::Test(const Napi::CallbackInfo& info) : ObjectWrap(info) { …}
  ```

- Define Class

  ```
  Napi::FunctionReference Test::constructor;
  constuctor = Napi::Persistent(Napi::Function  func  =
      DefineClass(env,  "Test", { Test::InstanceMethod(“func-name",  &Test::Func1), }));
  constructor.SuppressDestruct();
  ```

- Return constructor

  ```
  exports.set(“Test”, constructor());
  ```

- Create Instance/use

  ```
  const addon = require('bindings')('addon');
  const obj = new Test()
  obj.func-name;
  ```

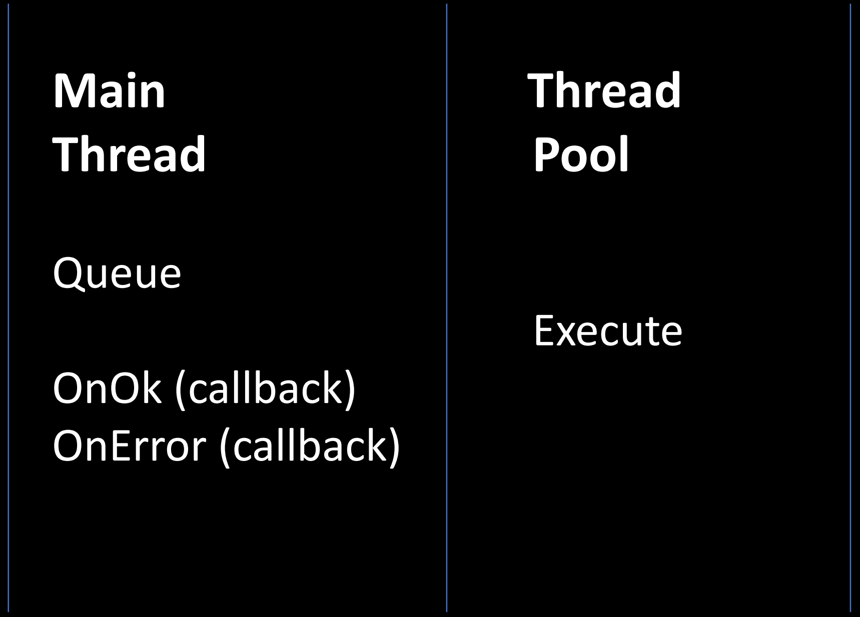
Creates JavaScript Object and Native Object with shared Lifetime

AsyncWorker

- Extend class AsyncWorker

- Implement
  - Execute
- Create Instance
  - Pass in callback
- Call Queue

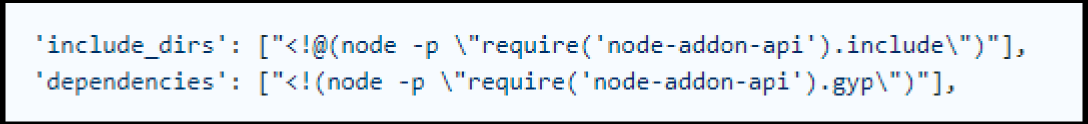
- Can also override
  - OnOK
  - OnError



!!! JavaScript/node-addon-api calls can only Run on Main thread !!!

Installation and Usage

- Add dependency to package.json
- Update binding.gyp
- #include “napi.h”

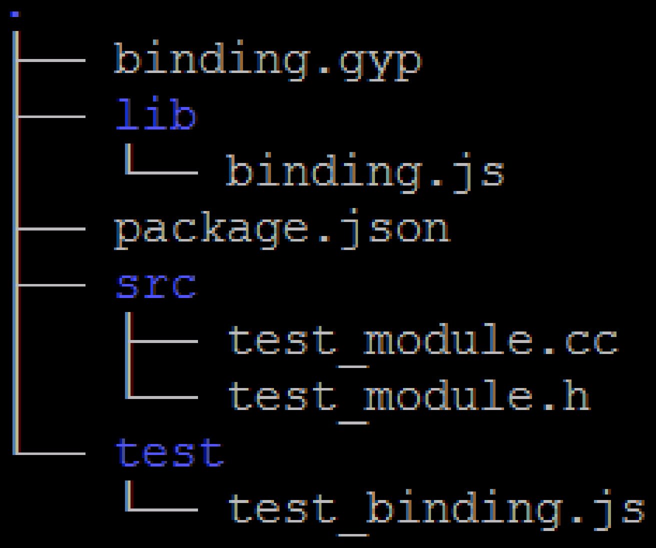


```
'include_dirs': ["<!@(node -p \"require('node-addon-api').include\")"],
'dependencies': ["<!(node -p \"require('node-addon-api').gyp\")"],
```

- Choose to enable/disable exception handling

Starting From Scratch

- Yeoman generator
  - generator-napi-module
- yo napi-module

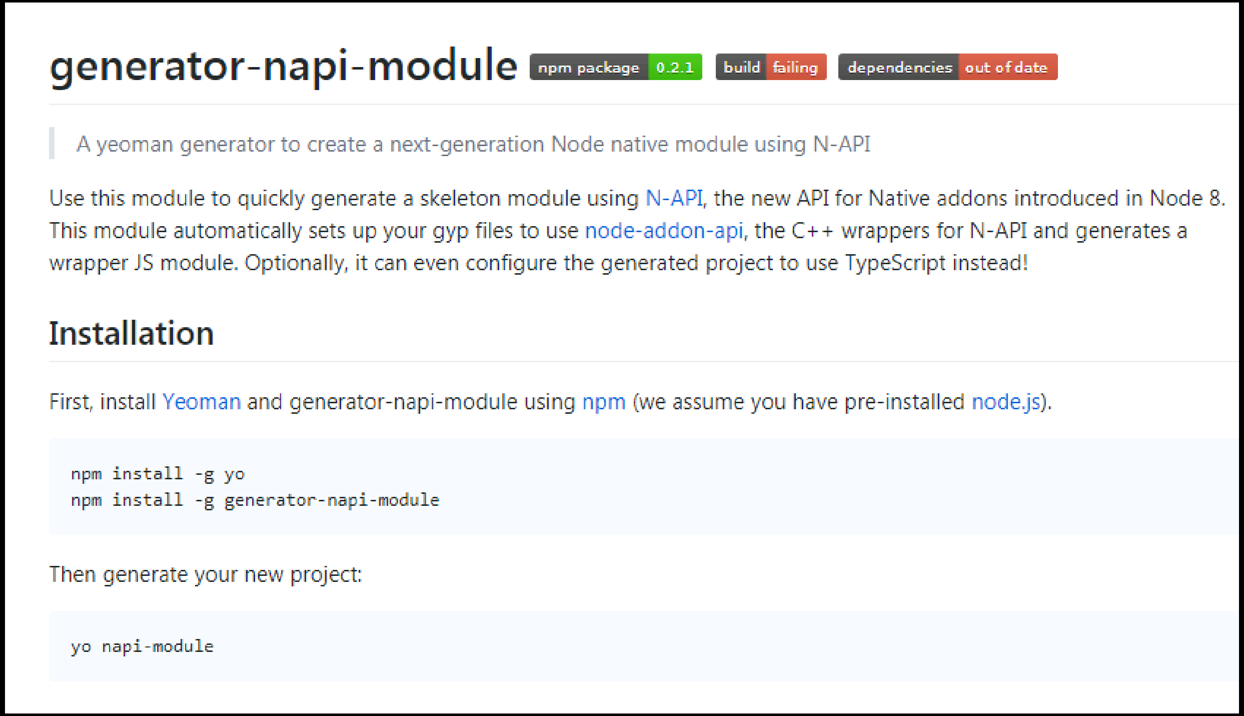


```
.
├── binding.gyp
├── lib
│   └── binding.js
├── package.json
├── src
│   ├── test_module.cc
│   └── test_module.h
└── test
    └── test_binding.js
```

- npm install
- node test/test_binding.js
- Might be more than you need as it assumes ObjectWrap, if so possibly start with - https://github.com/nodejs/abi-stable-node-addon-examples

- Workshop - napi.inspiredware.com/getting-started/first.html

https://www.npmjs.com/package/generator-napi-module



generator-napi-module

> A yeoman generator to create a next-generation Node native module using N-API

Use this module to quickly generate a skeleton module using N-API, the new API for Native addons introduced in Node 8. This module automatically sets up your gyp files to use node-addon-api, the C++ wrappers for N-API and generates a wrapper JS module. Optionally, it can even configure the generated project to use TypeScript instead!

Installation

First, install Yeoman and generator-napi-module using npm (we assume you have pre-installed node.js).

```
npm install -g yo
npm install -g generator-napi-module
```

Then generate your new project:

```
yo napi-module
```

Conversion of Existing Module

- tools/conversion.js

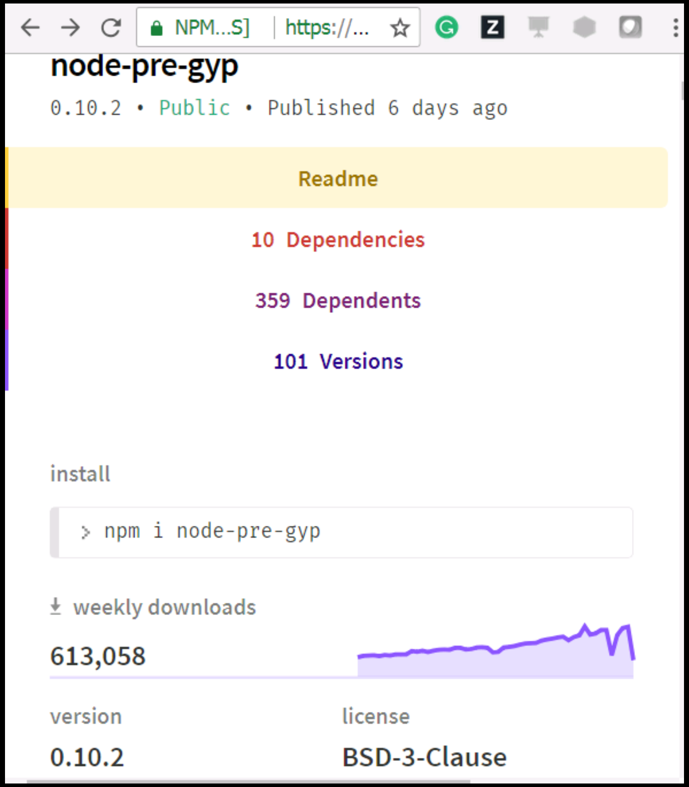
- go to your module directory
- npm install node-addon-api
- node ./node_modules/node-addon-api/tools/conversion.js

NAN ➔ node-addon-api

- Workshop - napi.inspiredware.com/getting-started/migration.html

Node-pre-gyp

- Helps with pre-built binaries
- New
  - napi_versions, napi_build_version, node_abi_napi, napi_version
- Replace
  - {node_abi} with {node_abi_napi}{napi_build_version}



NPM...S] https://...

node-pre-gyp

0.10.2 • Public • Published 6 days ago

Readme

10 Dependencies

359 Dependents

101 Versions

install

```
> npm i node-pre-gyp
```

weekly downloads

613,058

version	license
0.10.2	BSD-3-Clause

```
"binary": {
   "module_name": "your_module",
   "module_path": "./lib/binding/napi-v{napi_build_version}",
   "remote_path": "./{module_name}/v{version}/",
   "package_name": " {node_abi_napi}{napi_build_version}-{platform}-{arch}.tar.gz",
   "host": "https://your_bucket.s3-us-west-1.amazonaws.com",
   "napi_versions": [1,3]
 }
```

https://www.npmjs.com/package/node-pre-gyp

Interop

- Not all features from N-API wrapped by node-addon-api(yet)
- You can still use them !
- Most classes have operator to get wrapped N-API type
  - Easy conversion to N-API types
  - Can mix node-addon-api and C N-API calls.

```
inline Env::operator napi_env() const {
  return _env;
}
```

node- addon-api ← → N-API

Experimental Features

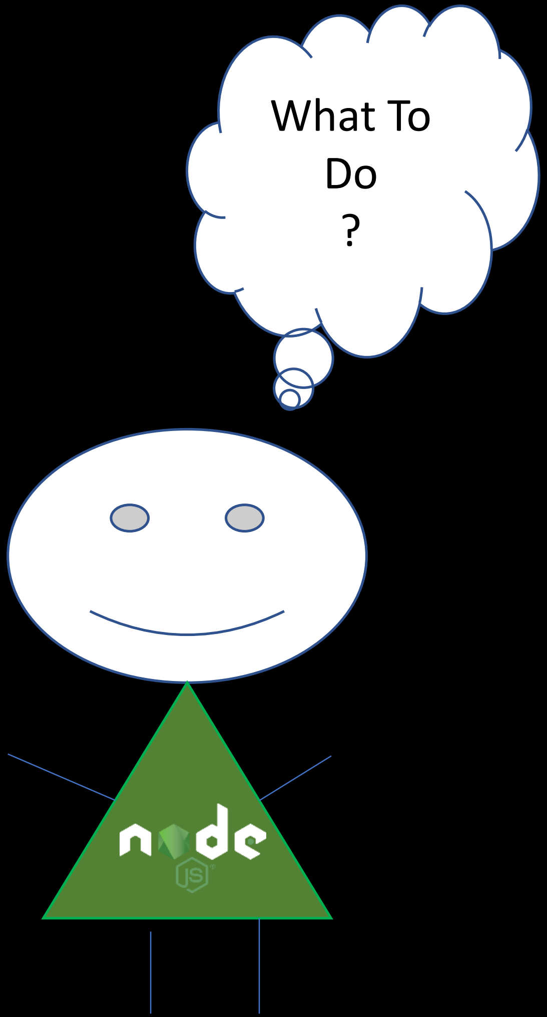
- New N-API features start as Experimental
- Opt-in with
  - #define NAPI_EXPERIMENTAL

Evolution

- A new API **must** adhere to N-API API shape and spirit
  - **Must** be a C API
  - **Must** not throw exceptions
  - **Must** return napi_status
  - **Should** consume napi_env
  - **Must** operate only on primitive data types, pointers to primitive datatypes or opaque handles
  - **Must** be a necessary API and not a nice to have. Convenience APIs belong in **node-addon-api**.
  - **Must** not change the signature of an existing N-API API or break ABI compatibility with other versions of Node.
  - New API **should** be agnostic towards the underlying JavaScript VM

Getting Involved

- Doc
- Testing
- Issue Triage
- Support for N-API Features
- Porting Modules



Q & A

NATURAL DISASTERS ARE AMONG THE WORLD'S GREATEST CHALLENGES

HOW WOULD 22 MILLION DEVELOPERS SOLVE THESE GLOBAL ISSUES IF GIVEN A CHANCE TO ANSWER THE CALL?

FIND OUT HOW AT

developer.ibm.com/callforcode

CALL FOR CODE

Call for Code Founding Partner

Copyright and Trademarks