N-API
Simplifying Node.js Development and Deployment

Arunesh Chandra, Microsoft
Michael Dawson, IBM
Arunesh Chandra, Microsoft

Sr. Program Manager, Microsoft Edge

N-API
Node-ChakraCore Project
Time-Travel Debugging

Contact info:
arunesh.chandra@microsoft.com
Twitter: @AruneshC
Github: @aruneshchandra
Michael Dawson, IBM

Senior Software Developer @ IBM
IBM Node.js community lead/IBM Runtime Technologies

Node.js collaborator, TSC member and Chair, CommComm Member
Active in Diagnostics, Build, Security, Benchmarking, N-API and
user-feedback among other Node.js teams and working groups

Contact me:
michael_dawson@ca.ibm.com
Twitter: @mhdawson1
Github: @mhdawson
Linkedin: https://www.linkedin.com/in/michael-dawson-6051282
Node.js & the NPM module ecosystem

Vast module ecosystem makes Node.js more powerful

Future growth depends on the stability of this ecosystem

Need to preserve and strengthen this ecosystem
Today the module ecosystem is fragile!
Error: The module '/home/user1/napi-demo-no-napi/napi_demo/node_modules/bindings/bindings.js:88' was compiled against a different Node.js version using NODE_MODULE_VERSION 57. This version of Node.js requires NODE_MODULE_VERSION 64. Please try re-compiling or re-installing the module (for instance, using `npm rebuild` or `npm install`).
addaleax commented 6 days ago

The removal of the deprecated versions of V8 methods breaks a lot of addons, and generates a large number of failing CITGM entries... Is there something we want to do about it?

Dan Shaw
@dshaw

Replying to @dshaw @AruneshC and 4 others

Mission critical native code dependencies is the #1 reason I've been hearing for folks who haven't upgraded to node v4+.

12:05 PM - 3 Apr 2017
Imagine...
(Node.js App Developers)

- Upgrading Node.js was less work
- Easily swap back and forth between Node versions
- More high performance native modules
- All modules were portable
- Binaries were compatible
- Leverage Innovation in different Node.js “flavors”
Imagine...
(Module Developers)

• No code update to support new Node.js release
• No need to create new binaries to support a new Node.js release
• It is easier to write an native module
• The API’s that are safe to use are well defined and documented
• Your modules will run on other Node.js flavors
What’s up?

- N-API at the end of the rainbow 😊
- History and Context
- How you can help
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!
Contributors
Some context ...
JavaScript escaped the browser sandbox
Enter Native Addon Modules
Native addon use cases

Access native APIs on the platform unavailable via JavaScript

Derive better performance for compute bound code using C/C++

Use of legacy code/logic available in C/C++ via JavaScript
Node.js native add-ons

- node-sass
- zmq
- sqlite3
- ref
- pty.js
- libxmljs
- iconv
- bcrypt
- fibers
- serialport
- canvas
- leveldown
- fsevents
- microtime
- hiredis
- nanomsg
- websocket
- node-hid
- hiredis
- IoTivity

~30% estimated ecosystem dependency
Before N-API

Module maintainers have to update their modules to support new Node.js versions.

Module consumers have to recompile their NAN modules OR have to wait for updated modules to be available.
First attempt to solve the problem - NAN

- Available since 0.8!
- NAN can only provide so much isolation
- Does not address deployment issue
- Needed new approach
N-API Modules

Module maintainers don’t have to update their modules to support new Node.js versions.

Module consumers don’t have to recompile their N-API modules OR don’t have to wait for updated modules to be available.
Demo

- https://github.com/boingoiing/napi_demo
Where Can I use N-API?

Available across all LTS release lines!

All the latest LTS release lines provide the same set of APIs for N-API.

- 10.x — Stable
- 8.x — Stable
- 6.x — Experimental (Doc only, looking for feedback)
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”

C++

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

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

You Write

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

You Get

```c
napi_status status;
napi_value object, string;
status = napi_create_object(env, &object);
if (status != napi_ok) {
    napi_throw_error(env, ...);
    return;
}
status = napi_create_string_utf8(env, "bar", NAPI_AUTO_LENGTH, &string);
if (status != napi_ok) {
    napi_throw_error(env, ...);
    return;
}
status = napi_set_named_property(env, object, "foo", string);
if (status != napi_ok) {
    napi_throw_error(env, ...);
    return;
}
```
API Shape

• Collection of C APIs available natively in Node.js 6.x, 8.x, 10.x etc.
  • API Docs -
    https://nodejs.org/dist/latest-v8.x/docs/api/n-api.html
  • ../src/node_api.h

NAPI Examples:

```c
napi_status napi_create_array(napi_env env, napi_value* result);
napi_status napi_get_last_error_info(napi_env e, const napi_extended_error_info** result);
napi_status napi_is_exception_pending(napi_env e, bool* result);
napi_status napi_get_and_clear_last_exception(napi_env e, napi_value* result);
napi_status napi_throw(napi_env e, napi_value error);
```
Examples – C- API

```
#include <node_api.h>

napi_value RunCallback(napi_env env, const napi_callback_info info) {
    ...
}

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

napi_value Init(napi_env env, napi_value exports) {
    napi_value new_exports;
    napi_status status =
        napi_create_function(env, "", NAPI_AUTO_LENGTH, RunCallback, nullptr, &new_exports);
    assert(status == napi_ok);
    return new_exports;
}

NAPI_MODULE(addon, Init);
```
Examples – C- API

```c
#include <node_api.h>

napi_value RunCallback(napi_env env, const napi_callback_info info) {
    napi_status status;

    size_t argc = 1;
    napi_value args[1];
    status = napi_get_cb_info(env, info, &argc, args, nullptr, nullptr);

    napi_value cb = args[0];

    napi_value argv[1];
    status = napi_create_string_utf8(env, "hello world", NAPI_AUTO_LENGTH, argv);

    napi_value global;
    status = napi_get_global(env, &global);

    napi_value result;
    status = napi_call_function(env, global, cb, 1, argv, &result);
    return nullptr;
}
```
Example – C++ Wrapper Example

```
#include <nan.h>

void RunCallback(const Nan::FunctionCallbackInfo<v8::Value>& info) {
  v8::Local<v8::Function> cb = info[0].As<v8::Function>();
  const unsigned argc = 1;
  v8::Local<v8::Value> argv[argc] = {
    Nan::New("hello world").ToLocalChecked()};
  Nan::MakeCallback(Nan::GetCurrentContext()->Global(), cb, argc, argv);
}

void Init(v8::Local<v8::Object> exports, v8::Local<v8::Object> module) {
  Nan::SetMethod(module, "exports", RunCallback);
}

NODE_MODULE(addon, Init)
```

```
#include <napi.h>

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

Napi::Object Init(Napi::Env env, Napi::Object exports) {
  return Napi::Function::New(env, RunCallback);
}

NODE_API_MODULE(addon, Init)
```

https://github.com/nodejs/node-addon-examples/tree/master/3_callbacks/node-addon-api

https://github.com/nodejs/node-addon-api/blob/master/tools/conversion.js
What are people saying about it ...

Mikeal Rogers
@mikeal
Pretty big deal, new native layer for Node.js has landed

Ingvar Stepanyan
@RReverser
@nodejs N-API is the kind of API that I wish JS engines would have had for a long time

Joyee Cheung
@JoyeeCheung
Oh wow the C++ wrappers of N-API are much easier to use than NAN/the V8 API. Kudos to everyone who worked on the initiative!
4:32 PM - 24 Jun 2018

Hubrix
@hubrixco
Yowza! #NodeJS, already a very #API-friendly language, gets downright API-amorous with its N-API feature, which gets tagged stable with full support starting in Node 10.
llnode – Now built on N-API

Joyee Cheung
@JoyeeCheung

The first version of the JS API of llnode (as a normal Node.js addon) just landed! https://github.com/nodejs/llnode/...

Thanks to the reviewers in each iteration and @rnchamberlain for the original work, also h/t to the N-API team: it got much simpler after being rewritten with N-API!
Log++

- Native logging framework
- Built on N-API
- Compute-intensive formatting is implemented in high performance C++
- The formatting task is offloaded to a background processing thread
- 4 – 7 times faster than any existing logging solutions.

https://github.com/mrmarron/logpp
https://www.npmjs.com/package/logpp
How you can help – Node.js App Developer

• Use N-API version of the native modules your app depends on
  • Look for Badges when choosing new modules
  • Run script to identify non-NAPI modules
    • https://github.com/nodejs/node-addon-api/blob/master/doc/checker-tool.md
    • https://github.com/nodejs/node-addon-api/blob/master/tools/check-napi.js
  • Raise issue on module to ask for N-API version, @mention N-API team
• Port your fav native module to N-API
How you can help - Native Module maintainer

• Port your native module to N-API
• Review documentation for gaps and provide feedback
• Support others who are trying to port your module to N-API

Join the N-API Working Group at https://github.com/nodejs/abi-stable-node
Useful Links

• node-addon-api Documentation (C++ Wrapper)
  https://github.com/nodejs/node-addon-api

• N-API Documentation (Flat C apis)
  https://nodejs.org/api/n-api.html

• Conversion tool for migrating existing NAN modules to N-API
  https://github.com/nodejs/node-addon-api/blob/master/tools/conversion.js

• Yeoman generator for building N-API module from scratch
  https://github.com/digitalinfinity/generator-napi-module (C++ Wrapper)

• Guidance for publishing N-API versions of your modules
  https://nodejs.org/en/docs/guides/publishing-napi-modules/

• Online Tutorials
  https://napi.inspiredware.com/getting-started/first.html
Thank You!

@aruneshc
@mhdawson1

Join the N-API Working Group at https://github.com/nodejs/abi-stable-node
Copyright and Trademarks

© IBM Corporation and Microsoft Corporation 2018. All Rights Reserved

IBM, the IBM logo, ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies.

A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml

Microsoft is a trademark of Microsoft Corporation in the United States, other countries, or both.

Node.js is an official trademark of Joyent. IBM SDK for Node.js is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

Java, JavaScript and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

npm is a trademark of npm, Inc.

Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries.

Other trademarks or logos are owned by their respective owners.