Power Up Node.js with C++

Fishman Boaz

Node.js Israel

03/10/18

Agenda

- Native Modules.
  - Why Native Modules?
  - Types of Native Modules.
- Examples.
- Benchmarks.
- Deployment.
- Tools.

#whoami

- Fishman Boaz
- Senior Developer @ Random Logic (A.K.A. 888).
- Fishman.Boaz at gmail.com
- ManicQin @ gitlab / github

- Check out the repository for Examples and references: https://gitlab.com/gPowerUp

Why Native modules?

- Taking advantage of the C\C++ ecosystem.
  - » C is 46 years old.
  - » Every new technology will first bind to C.

Why Native modules?

- Taking advantage of the C\C++ ecosystem.
  - » C is 46 years old.
  - » Every new technology will first bind to C.
- Using low level OS api.
  - » Raspberry-pi-camera-native.
  - » Node-inspector/v8-profiler.

Why Native modules?

- Taking advantage of the C\C++ ecosystem.
  - » C is 46 years old.
  - » Every new technology will first bind to C.
- Using low level OS api.
  - » Raspberry-pi-camera-native.
  - » Node-inspector.
- Enjoying the performance gain.
  - » Best for long processing operations.

Why C++?

- Boost (~2000).
- C++11/14/17/2a and on.
- unique_ptr, shared_ptr, lambdas, auto, for each, optional, any, string_view, tuple, functors, coroutines, atomic, futures, modules, concepts, etc...

Not only C++

- Rust – Neon.
  - » Rust bindings for writing safe and fast native Node.js modules.
- C# - Edge.JS
  - » Run .NET and Node.js code in-process on Windows, MacOS, and Linux.
- Python – python-shell
  - » A simple way to run Python scripts from Node.js with basic but efficient inter-process communication and better error handling.

Native Modules

- Dynamically-linked objects.
  - » Developed against API.
  - » Built against ABI.
- Loaded using require().
- Behaves as regular modules.

API vs. ABI

- API – Application Programming Interface.
  - » ”… it is a set of clearly defined methods of communication between various components.”
- ABI – Application Binary Interface.
  - » ”... an interface between two binary program modules...”

Native Modules - Types

- V8 API
  - » “V8 is Google's open source JavaScript engine”.
  - » API may break. ABI will break.

Native Modules - Types

- V8 API
  - » “V8 is Google's open source JavaScript engine”.
  - » API may break. ABI will break.
- Nan
  - » “Native Abstractions for Node.js”.
  - » API won’t break. ABI will break.

Native Modules - Types

- V8 API
  - » “V8 is Google's open source JavaScript engine”.
  - » API may break. ABI will break.
- Nan
  - » “Native Abstractions for Node.js”.
  - » API won’t break. ABI will break.
- NAPI
  - » “NAPI — Node with PoC ABI stable API for native modules.”
  - » API won’t break. ABI won’t break.

V8

Pros

» Best for embedding JS in your C++ server.

» Best performance.

– Cons

» Design decisions – oriented for embedding and not for addons.

» Unneeded boilerplate.

» Not API Stable

» Not ABI Stable

V8

- Installation
  - » npm install –save node-gyp.

NAN

- Pros
  - » API Stable.
  - » Addon oriented.
  - » Less unneeded boilerplate.
- Cons
  - » Abstraction boilerplate.
  - » Not ABI Stable.

NAN

- Installation
  - » npm install –save nan.
  - » npm install –save node-gyp.

NAPI

- Pros
  - » API Stable
  - » ABI Stable - Across versions and flavors!
- Cons
  - » Added complexity.
  - » Reduced performance.

NAPI

- Installation
  - » npm install –save node-api-addon.
  - » npm install –save node-gyp.

NAPI - Example

```
#include <napi.h>

using namespace v8;

Value sum_int(const CallbackInfo &info) {
    Env env = info.Env();

    if (!info[0].IsNumber()) {
        TypeError::New(env, "param 1 is not an int").ThrowAsJavaScriptException();
        return env.Null();
    }

    if (!info[1].IsNumber()) {
        TypeError::New(env, "param 2 is not an int").ThrowAsJavaScriptException();
        return env.Null();
    }

    int param1 = info[0].As<Number>().Int32Value();
    int param2 = info[1].As<Number>().Int32Value();
    int retval = param1 + param2;

    return Number::New(env, retval);
}

Object Init(Env env, Object exports) {

    exports.Set(String::New(env, "sum_int"),
                Function::New(env, sum_int));

    return exports;
}

NODE_API_MODULE(addon, Init)
```

NAPI - Example

```
#include <napi.h>

using namespace v8;

Value sum_int(const CallbackInfo &info) {
    Env env = info.Env();

    if (!info[0].IsNumber()) {
        TypeError::New(env, "param 1 is not an int").ThrowAsJavaScriptException();
        return env.Null();
    }

    if (!info[1].IsNumber()) {
        TypeError::New(env, "param 2 is not an int").ThrowAsJavaScriptException();
        return env.Null();
    }

    int param1 = info[0].As<Number>().Int32Value();
    int param2 = info[1].As<Number>().Int32Value();
    int retval = param1 + param2;

    return Number::New(env, retval);
}

Object Init(Env env, Object exports) {

    exports.Set(String::New(env, "sum_int"),
                Function::New(env, sum_int));

    return exports;
}

NODE_API_MODULE(addon, Init)
```

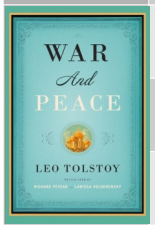
NAPI - Example

```
#include <napi.h>

using namespace v8;

Value sum_int(const CallbackInfo &info) {
    Env env = info.Env();

    if (!info[0].IsNumber()) {
        TypeError::New(env, "param 1 is not an int").ThrowAsJavaScriptException();
        return env.Null();
    }

    if (!info[1].IsNumber()) {
        TypeError::New(env, "param 2 is not an int").ThrowAsJavaScriptException();
        return env.Null();
    }

    int param1 = info[0].As<Number>().Int32Value();
    int param2 = info[1].As<Number>().Int32Value();
    int retval = param1 + param2;

    return Number::New(env, retval);
}

Object Init(Env env, Object exports) {

    exports.Set(String::New(env, "sum_int"),
                Function::New(env, sum_int));

    return exports;
}

NODE_API_MODULE(addon, Init)
```

NAPI - Example

```
#include <napi.h>

using namespace v8;

Value sum_int(const CallbackInfo &info) {
    Env env = info.Env();

    if (!info[0].IsNumber()) {
        TypeError::New(env, "param 1 is not an int").ThrowAsJavaScriptException();
        return env.Null();
    }

    if (!info[1].IsNumber()) {
        TypeError::New(env, "param 2 is not an int").ThrowAsJavaScriptException();
        return env.Null();
    }

    int param1 = info[0].As<Number>().Int32Value();
    int param2 = info[1].As<Number>().Int32Value();
    int retval = param1 + param2;

    return Number::New(env, retval);
}

Object Init(Env env, Object exports) {

    exports.Set(String::New(env, "sum_int"),
                Function::New(env, sum_int));

    return exports;
}

NODE_API_MODULE(addon, Init)
```

Benchmarks

#words	Package		time
10	https://gitlab.com/gPowerUp/Examples/03_NAPI	fastest	18,273.04 ops/sec
	https://www.npmjs.com/package/count-words	-58.3%	7,618.85 ops/sec
	https://www.npmjs.com/package/count-words-occurrence	-91.07%	1,630.98 ops/sec
100	https://gitlab.com/gPowerUp/Examples/03_NAPI	fastest	2,011.32 ops/sec
	https://www.npmjs.com/package/count-words	-64.42%	837.40 ops/sec
	https://www.npmjs.com/package/count-words-occurrence	-97.46%	64.18 ops/sec
1000	https://gitlab.com/gPowerUp/Examples/03_NAPI	fastest	324.10 ops/sec
	https://www.npmjs.com/package/count-words	-71.6%	92.06 ops/sec
	https://www.npmjs.com/package/count-words-occurrence	-98.53%	4.75 ops/sec
	https://gitlab.com/gPowerUp/Examples/03_NAPI		492 ms
	https://www.npmjs.com/package/count-words		995 ms
	https://www.npmjs.com/package/count-words-occurrence		----------

Benchmarks - cont

Test	Module		ops/sec
Sum 2 integers	V8	fastest	12,468,699.62 ops/sec
	NAN	-13.97%	10,726,964.04 ops/sec
	NAPI	-49.53%	6,292,693.67 ops/sec
sum an array of integers	V8	fastest	3,458,794.97 ops/sec
	NAN	-14.02%	2,973,962.27 ops/sec
	NAPI	-52.15%	1,655,135.03 ops/sec
Reverse the string “1234”	V8	-10.17%	4,708,693.99 ops/sec
	NAN	fastest	5,241,578.48 ops/sec
	NAPI	-24.91%	3,935,939.14 ops/sec
Calulate the distance between 2 points (json objects)	V8	fastest	1,379,763.30 ops/sec
	NAN	-23.26%	1,058,819.32 ops/sec
	NAPI	-54.54%	627,240.62 ops/sec
Unique sort array	V8	fastest	968,858.28 ops/sec
	NAN	-4.85%	921,908.09 ops/sec
	NAPI	-39.45%	586,635.88 ops/sec

Deployment

- Servers
  - » Docker.
  - » npm install.
  - » Distribute prebuilt pacakages using node-pre-gyp / prebuild.
- Serverless
  - » AWS Lambda – bundle in zip and deploy.
  - » Azure Functions – should be supported in Azure Functions Ver 2.

Tools

- Node-gyp.
- CmakeJS.
- Boost-lib.
- Generator-napi-module.
- Bindings.
- node-pre-gyp / prebuild.