ABI-Stable Node

NODE.JS API (N-API)

Things to cover

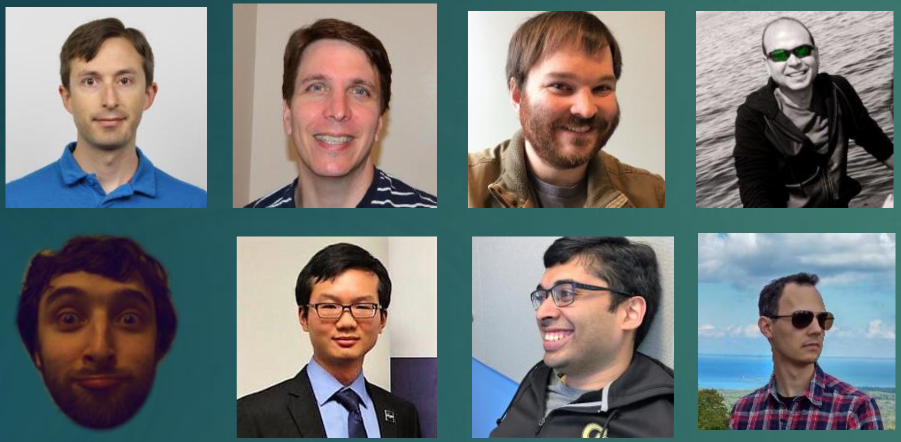
- Progress update
- Usability
- API Coverage
- Demo
- Performance
- Remaining Work
- Proposal and Discussion
- Next steps

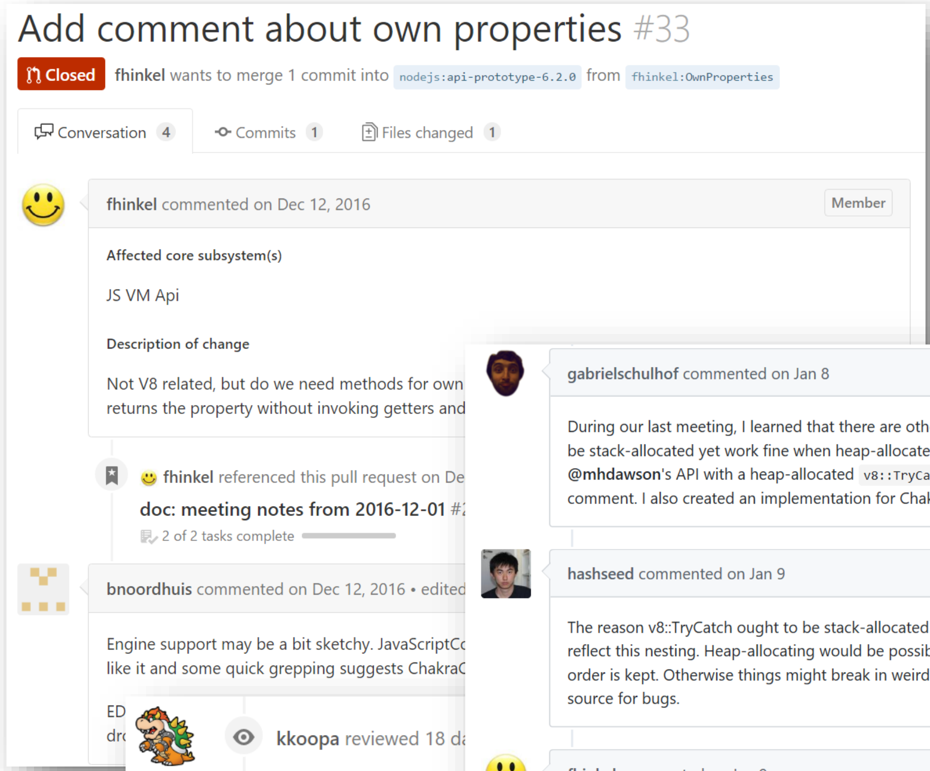
Goals

- Create ABI stable API surface for Node.js native modules
- Ensure other Node-VMs can implement these API

Core Team and community participation

Core Team





Add comment about own properties #33

Closed fhinkel wants to merge 1 commit into `nodejs:api-prototype-6.2.0` from `fhinkel:OwnProperties`

Conversation 4 Commits 1 Files changed 1

fhinkel commented on Dec 12, 2016 Member

Affected core subsystem(s)

JS VM Api

Description of change

Not V8 related, but do we need methods for own
returns the property without invoking getters and

fhinkel referenced this pull request on De

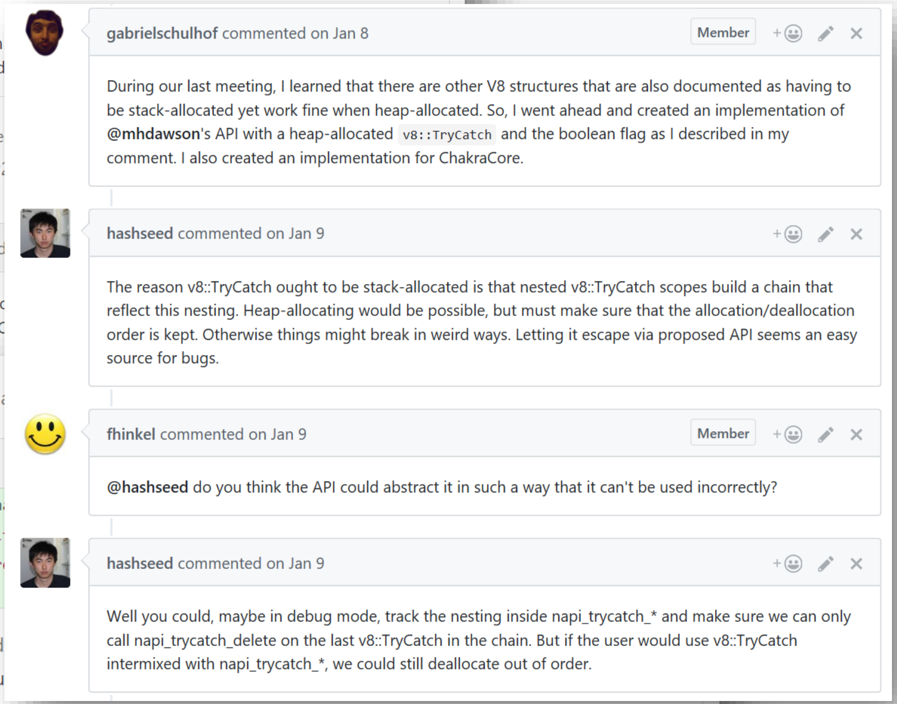
doc: meeting notes from 2016-12-01 #

2 of 2 tasks complete

bnoordhuis commented on Dec 12, 2016 • edited

Engine support may be a bit sketchy. JavaScriptC
like it and some quick grepping suggests Chakra

ED
dr



gabrielschulhof commented on Jan 8 Member

During our last meeting, I learned that there are other V8 structures that are also documented as having to be stack-allocated yet work fine when heap-allocated. So, I went ahead and created an implementation of **@mhdawson**'s API with a heap-allocated `v8::TryCatch` and the boolean flag as I described in my comment. I also created an implementation for ChakraCore.

hashseed commented on Jan 9

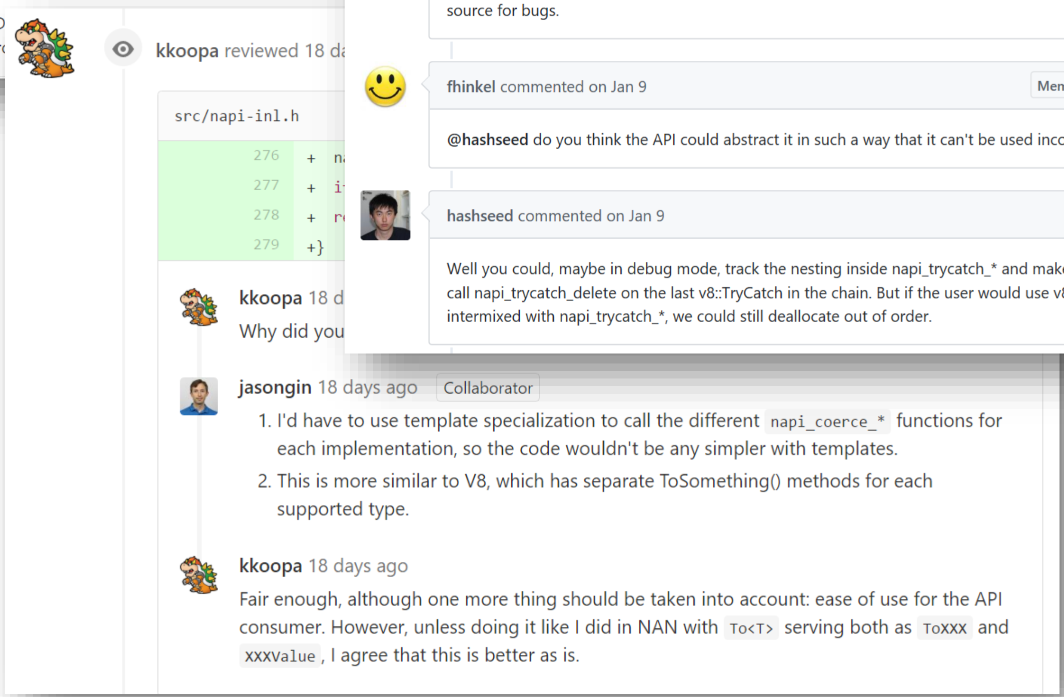
The reason v8::TryCatch ought to be stack-allocated is that nested v8::TryCatch scopes build a chain that reflect this nesting. Heap-allocating would be possible, but must make sure that the allocation/deallocation order is kept. Otherwise things might break in weird ways. Letting it escape via proposed API seems an easy source for bugs.

fhinkel commented on Jan 9 Member

@hashseed do you think the API could abstract it in such a way that it can't be used incorrectly?

hashseed commented on Jan 9

Well you could, maybe in debug mode, track the nesting inside napi_trycatch_* and make sure we can only call napi_trycatch_delete on the last v8::TryCatch in the chain. But if the user would use v8::TryCatch intermixed with napi_trycatch_*, we could still deallocate out of order.



kkoopa reviewed 18 d

src/napi-inl.h

```
+ n
+ i
+ r
+}
```

kkoopa 18 d
Why did you

jasongin 18 days ago Collaborator

1. I'd have to use template specialization to call the different `napi_coerce_*` functions for each implementation, so the code wouldn't be any simpler with templates.
2. This is more similar to V8, which has separate ToSomething() methods for each supported type.

kkoopa 18 days ago

Fair enough, although one more thing should be taken into account: ease of use for the API consumer. However, unless doing it like I did in NAN with `To<T>` serving both as `ToXXX` and `XXXValue`, I agree that this is better as is.

Progress Update

- N-API support now available for the following Node versions
  - 8.x Master
  - 6.x LTS
  - Node-ChakraCore
- Reviewed by **V8** and **ChakraCore** Teams
- Modules Converted
  - Node-Sass
  - Canvas
  - Leveldown
  - Nanomsg
  - IoTivity (~90% complete)

Progress Update contd ...

- API Shape & Error Handling

```
NODE_EXTERN napi_status napi_get_value_string_length(napi_env e, napi_value v, int* result);
NODE_EXTERN const napi_extended_error_info* napi_get_last_error_info();

NODE_EXTERN napi_status napi_is_exception_pending(napi_env e, bool* result);
NODE_EXTERN napi_status napi_get_and_clear_last_exception(napi_env e, napi_value* result);
NODE_EXTERN napi_status napi_throw(napi_env e, napi_value error);
```

- API usability
  - Flat C-style APIs
  - Optional C++ wrapper add-on

API Usability

C++ wrapper has built in Error Handling.

The wrapper may throw C++ exceptions, that are automatically re-thrown as JavaScript exceptions if not handled.

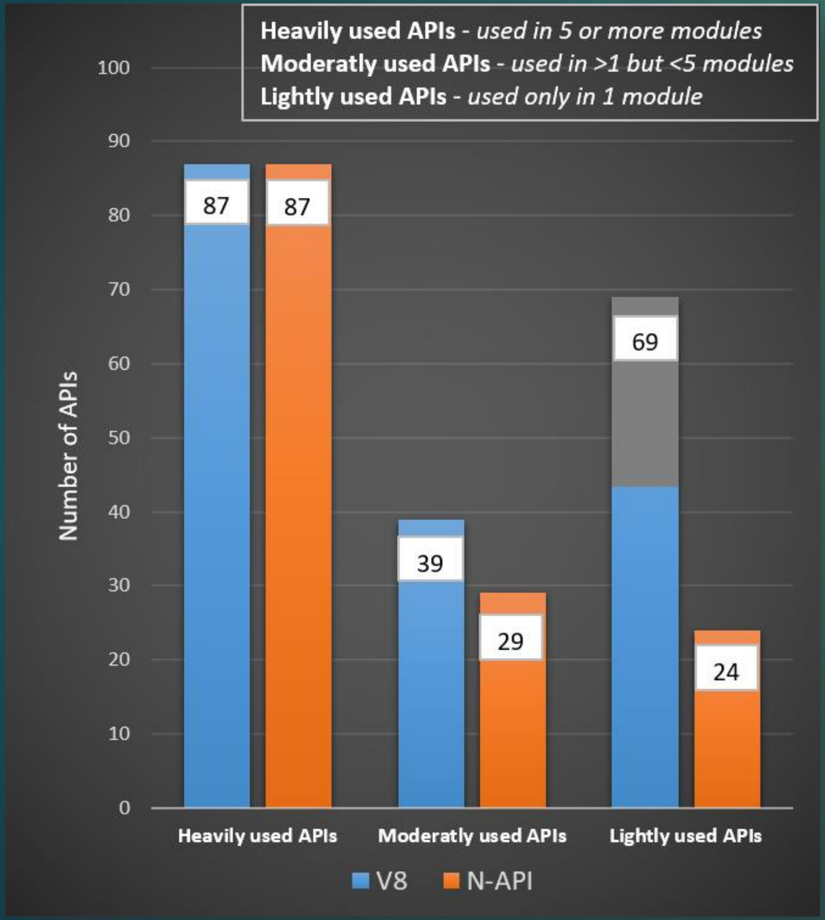
Flat C-style API

```
#define CHECK_STATUS                               \
  if (status != napi_ok) {                         \
    return;                                        \
  }

NAPI_METHOD(Shutdown) {
  napi_status status;
  napi_value args[2];
  status = napi_get_cb_args(env, info, args, 2);
  CHECK_STATUS;

  int s;
  status = napi_get_value_int64(env, args[0], &s);
  CHECK_STATUS;
  int how;
  status = napi_get_value_int64(env, args[1], &how);
  CHECK_STATUS;

  napi_value ret;
  status = napi_create_number(env, nn_shutdown(s, how), &ret);
  CHECK_STATUS;
  status = napi_set_return_value(env, info, ret);
  CHECK_STATUS;
}
```

C++ Wrapper

```
void Shutdown(const Napi::CallbackInfo& info) {
  int s = info[0]->As<Napi::Number>();
  int how = info[1]->As<Napi::Number>();
  return Napi::Number::New(info.Env(), nn_shutdown(s, how));
}
```

N-API Coverage*



195	*V8 APIs used*
140/195	*N-API equivalent exists*
83/195	*Exercised by 5 ported modules*

** Data based on Top 30 depended on native modules*

V8 APIs with no N-API Equivalent

v8::Context::Enter
v8::Context::Exit
v8::FunctionTemplate::Inherit
v8::Locker::~Locker
v8::Locker::Initialize
v8::Private::ForApi
v8::String::Concat
v8::String::NewFromOneByte
v8::TryCatch::StackTrace
v8::UnboundScript::GetId (*Issue #51*)
v8::Unlocker::~Unlocker
v8::Unlocker::Initialize
v8::V8::FromJustIsNothing
v8::Value::IsBooleanObject
v8::Value::IsDate
v8::Value::IsNativeError
v8::Value::IsNumberObject
v8::Value::IsRegExp
v8::Value::IsStringObject
v8::Value::ToDetailString
v8::Object::Delete (Issue #94)
v8::Object::GetIsolate
v8::Object::GetOwnPropertyNames (Issue #67)
v8::Object::GetPrivate
v8::Object::HasPrivate
v8::Object::SetPrivate
v8::ObjectTemplate::SetHandler

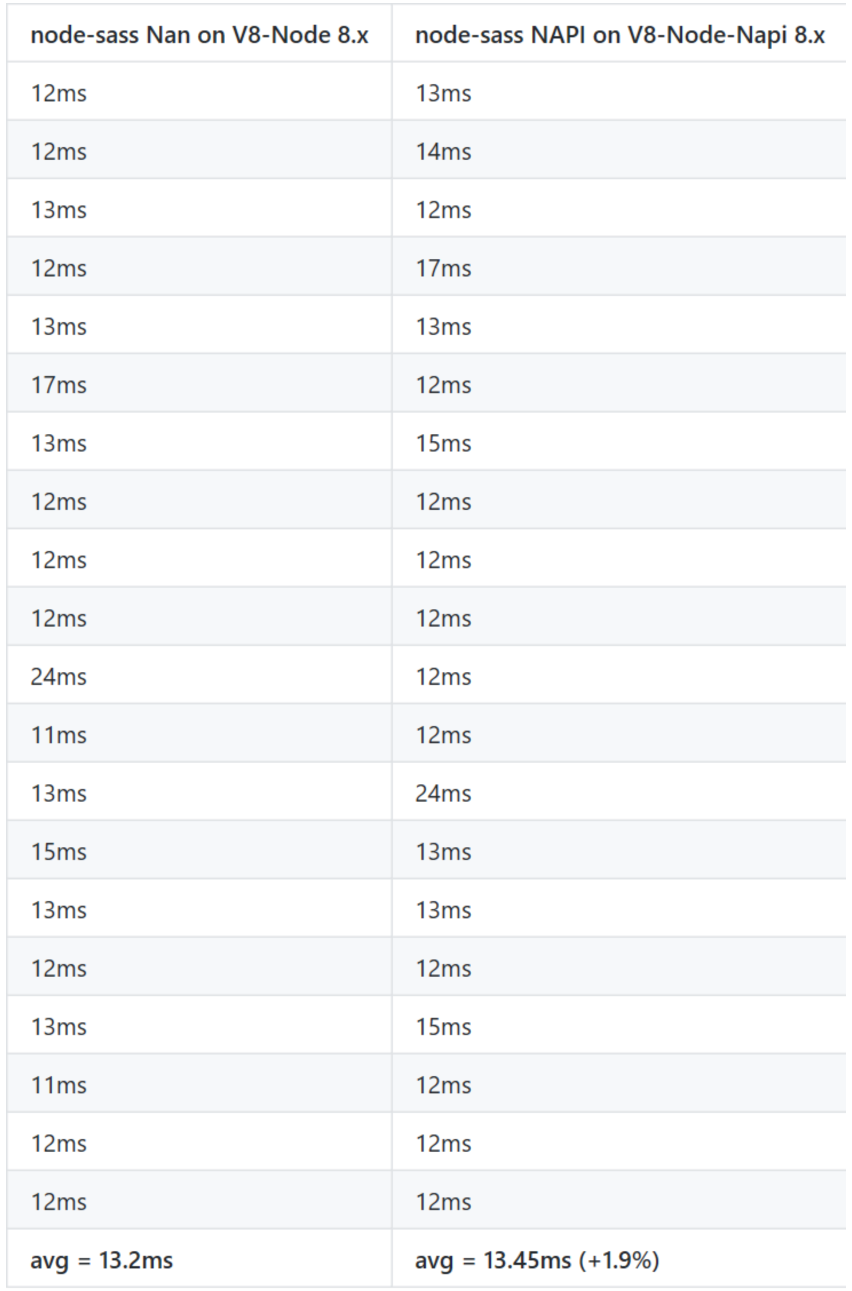
v8::Script::Compile (Issue #51)
v8::Script::GetUnboundScript (Issue #51)
v8::ScriptCompiler::ExternalSourceStream::ResetToBookmark
v8::ScriptCompiler::ExternalSourceStream::SetBookmark
v8::Isolate::AddGCEpilogueCallback
v8::Isolate::AddMemoryAllocationCallback
v8::Isolate::AddMessageListener
v8::Isolate::CancelTerminateExecution
v8::Isolate::CollectAllGarbage
v8::Isolate::DiscardThreadSpecificMetadata
v8::Isolate::Enter
v8::Isolate::Exit
v8::Isolate::IsDead
v8::Isolate::IsExecutionTerminating
v8::Isolate::RemoveGCEpilogueCallback
v8::Isolate::RemoveGCPrologueCallback
v8::Isolate::RemoveMemoryAllocationCallback
v8::Isolate::RemoveMessageListeners
v8::Isolate::SetAllowCodeGenerationFromStringsCallback
v8::Isolate::SetCaptureStackTraceForUncaughtExceptions
v8::Isolate::SetFailedAccessCheckCallbackFunction
v8::Isolate::SetFatalErrorHandler
v8::Isolate::SetStackLimit
v8::Isolate::TerminateExecution
v8::Isolate::VisitExternalResources
v8::Isolate::VisitHandlesForPartialDependence
v8::Isolate::VisitHandlesWithClassIds

Demo

https://github.com/boingoing/napi_demo

Perf Node-Sass

Ported using C style API
N-API adds 1.9% perf delta



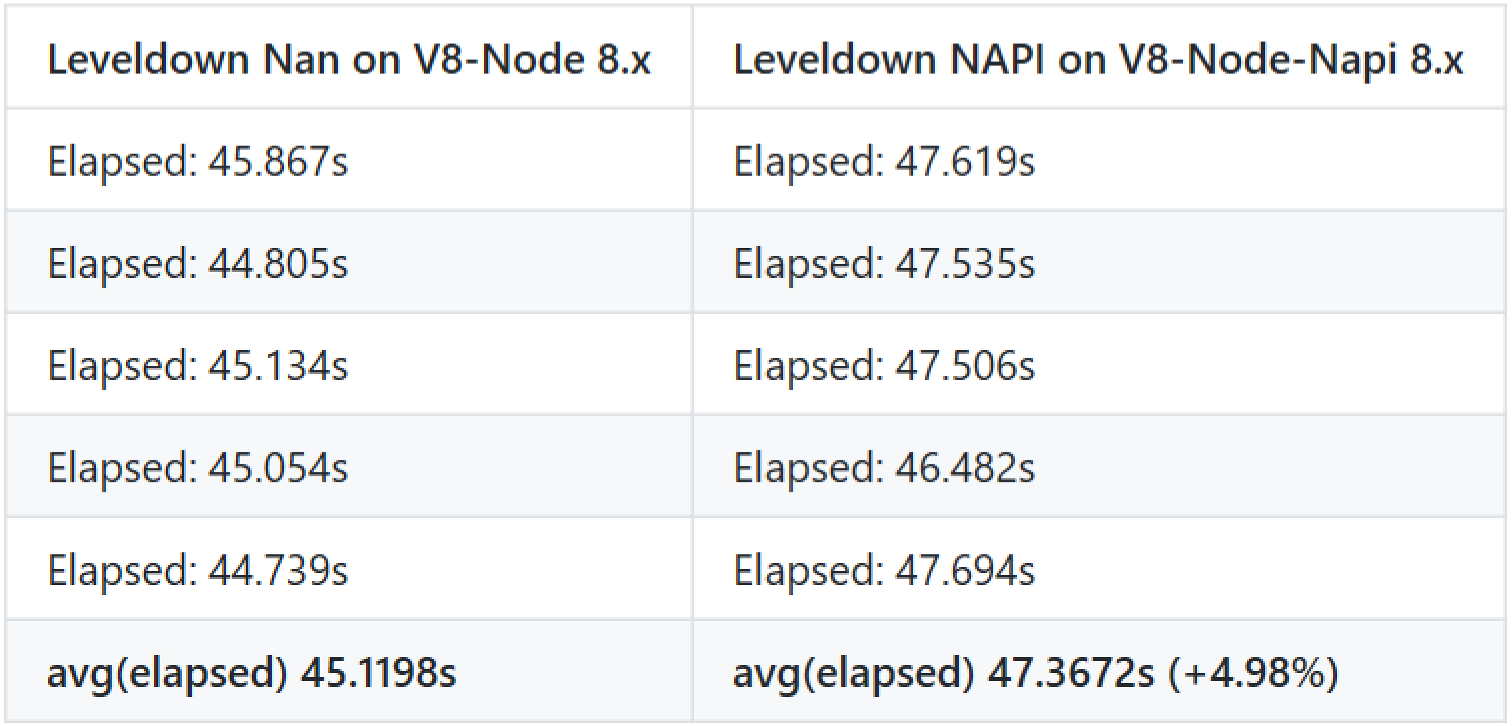
node-sass Nan on V8-Node 8.x	node-sass NAPI on V8-Node-Napi 8.x
12ms	13ms
12ms	14ms
13ms	12ms
12ms	17ms
13ms	13ms
17ms	12ms
13ms	15ms
12ms	12ms
12ms	12ms
12ms	12ms
24ms	12ms
11ms	12ms
13ms	24ms
15ms	13ms
13ms	13ms
12ms	12ms
13ms	15ms
11ms	12ms
12ms	12ms
12ms	12ms
avg = 13.2ms	avg = 13.45ms (+1.9%)

System Info:
Windows 10 x64
Intel Xeon E5-1620 v3 @ 3.50GHz
16GB DDR4 @ 2133MHz
Samsung XP941 SSD

Details at: https://github.com/nodejs/abi-stable-node/issues/82

Perf Leveldown

Ported using C style API
Benchmark includes 1M entries
DB Size 110 MB
N-API adds 5% perf delta



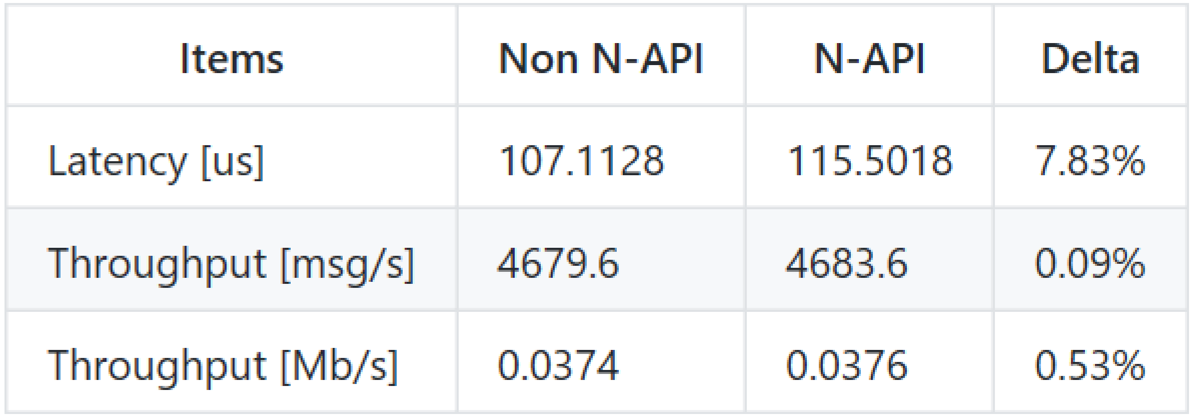
Leveldown Nan on V8-Node 8.x	Leveldown NAPI on V8-Node-Napi 8.x
Elapsed: 45.867s	Elapsed: 47.619s
Elapsed: 44.805s	Elapsed: 47.535s
Elapsed: 45.134s	Elapsed: 47.506s
Elapsed: 45.054s	Elapsed: 46.482s
Elapsed: 44.739s	Elapsed: 47.694s
avg(elapsed) 45.1198s	**avg(elapsed) 47.3672s (+4.98%)**

System Info:
Windows 10 x64
Intel Xeon E5-1620 v3 @ 3.50GHz
16GB DDR4 @ 2133MHz
Samsung XP941 SSD

Details at: https://github.com/nodejs/abi-stable-node/issues/55

Perf Nanomsg

Ported using C style API
Workload size 1 byte message
Performance within expected range



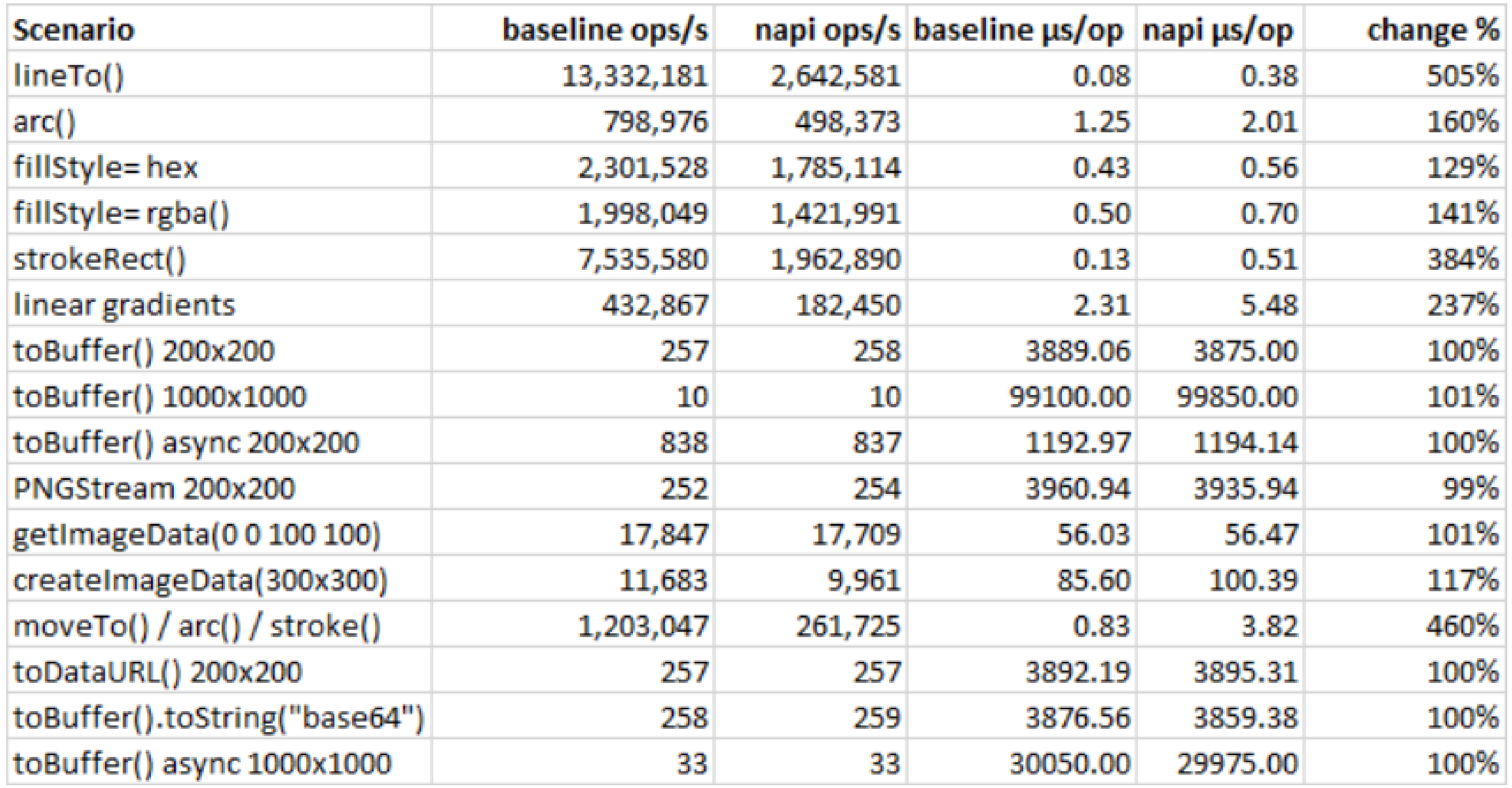
Items	Non N-API	N-API	Delta
Latency [us]	107.1128	115.5018	7.83%
Throughput [msg/s]	4679.6	4683.6	0.09%
Throughput [Mb/s]	0.0374	0.0376	0.53%

System Info:
Ubuntu 14.04.2 LTS (GNU/Linux 3.13.0-55-generic x86_64)
Intel CPU @ 2400 MHz

Details at: https://github.com/nodejs/abi-stable-node/issues/57

Perf
Canvas

Ported using C++ wrapper
Perf regression in chatty benchmarks



Scenario	baseline ops/s	napi ops/s	baseline µs/op	napi µs/op	change %
lineTo()	13,332,181	2,642,581	0.08	0.38	505%
arc()	798,976	498,373	1.25	2.01	160%
fillStyle= hex	2,301,528	1,785,114	0.43	0.56	129%
fillStyle= rgba()	1,998,049	1,421,991	0.50	0.70	141%
strokeRect()	7,535,580	1,962,890	0.13	0.51	384%
linear gradients	432,867	182,450	2.31	5.48	237%
toBuffer() 200x200	257	258	3889.06	3875.00	100%
toBuffer() 1000x1000	10	10	99100.00	99850.00	101%
toBuffer() async 200x200	838	837	1192.97	1194.14	100%
PNGStream 200x200	252	254	3960.94	3935.94	99%
getImageData(0 0 100 100)	17,847	17,709	56.03	56.47	101%
createImageData(300x300)	11,683	9,961	85.60	100.39	117%
moveTo() / arc() / stroke()	1,203,047	261,725	0.83	3.82	460%
toDataURL() 200x200	257	257	3892.19	3895.31	100%
toBuffer().toString("base64")	258	259	3876.56	3859.38	100%
toBuffer() async 1000x1000	33	33	30050.00	29975.00	100%

System Info:
Windows 10 x64
Intel Xeon W3530 @2.8GHz, 20 GB RAM

Details at: https://github.com/nodejs/abi-stable-node/issues/77

Thoughts on Performance so far…

- No performance tuning done yet!
- Expected performance for broad use cases to be within 0-10%
- Extremely chatty operations with native module see a larger perf regression

Work Items Remaining

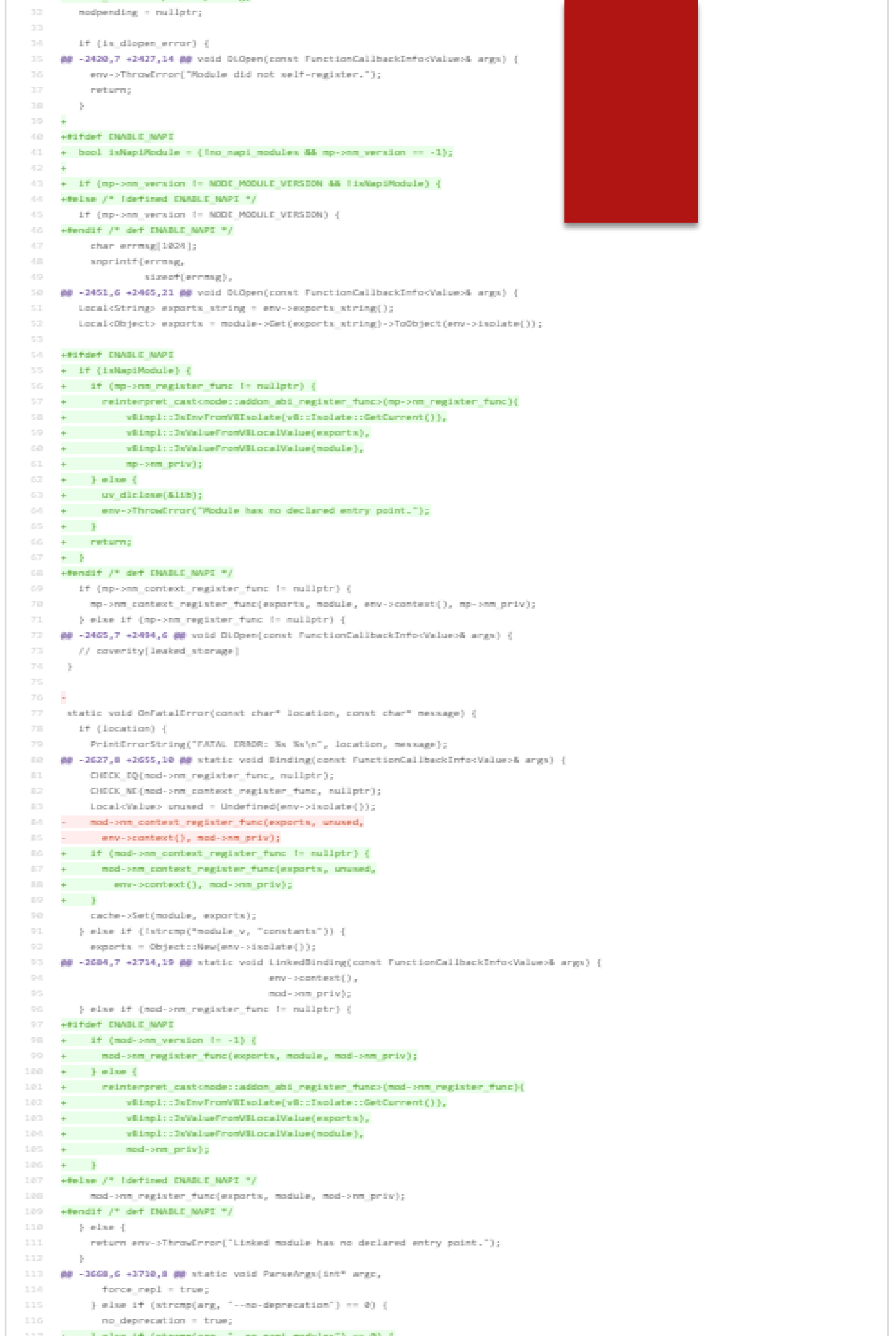
- Performance fine tuning
- Increase API Coverage
- Test coverage
- CI Integration
- N-API version management
- Documentation and Support
- Auto conversion tool from NAN to N-API

Proposal: N-API Compiled in by default in Node v8.0

We recommend to have N-API compiled in by default without build flags, for ease of use to allow broader usage. Gating can be done via command-line option if needed.

Node.cc changes for N-API are non-intrusive. It gets triggered only for N-API modules.

https://gist.github.com/gabrielschulhof/763a8563deab4eb5f681df3817658fe0



```diff
   modpending = nullptr;

   if (is_dlopen_error) {
@@ -2420,7 +2427,14 @@ void DLOpen(const FunctionCallbackInfo<Value>& args) {
     env->ThrowError("Module did not self-register.");
     return;
   }
+
+#ifdef ENABLE_NAPI
+  bool isNapiModule = (!no_napi_modules && mp->nm_version == -1);
+
+  if (mp->nm_version != NODE_MODULE_VERSION && !isNapiModule) {
+#else /* !defined ENABLE_NAPI */
   if (mp->nm_version != NODE_MODULE_VERSION) {
+#endif /* def ENABLE_NAPI */
     char errmsg[1024];
     snprintf(errmsg,
              sizeof(errmsg),
@@ -2451,6 +2465,21 @@ void DLOpen(const FunctionCallbackInfo<Value>& args) {
   Local<String> exports_string = env->exports_string();
   Local<Object> exports = module->Get(exports_string)->ToObject(env->isolate());

+#ifdef ENABLE_NAPI
+  if (isNapiModule) {
+    if (mp->nm_register_func != nullptr) {
+      reinterpret_cast<node::addon_abi_register_func>(mp->nm_register_func)(
+          v8impl::JsEnvFromV8Isolate(v8::Isolate::GetCurrent()),
+          v8impl::JsValueFromV8LocalValue(exports),
+          v8impl::JsValueFromV8LocalValue(module),
+          mp->nm_priv);
+    } else {
+      uv_dlclose(&lib);
+      env->ThrowError("Module has no declared entry point.");
+    }
+    return;
+  }
+#endif /* def ENABLE_NAPI */
   if (mp->nm_context_register_func != nullptr) {
     mp->nm_context_register_func(exports, module, env->context(), mp->nm_priv);
   } else if (mp->nm_register_func != nullptr) {
@@ -2465,7 +2494,6 @@ void DLOpen(const FunctionCallbackInfo<Value>& args) {
   // coverity[leaked_storage]
 }

-
 static void OnFatalError(const char* location, const char* message) {
   if (location) {
     PrintErrorString("FATAL ERROR: %s %s\n", location, message);
@@ -2627,8 +2655,10 @@ static void Binding(const FunctionCallbackInfo<Value>& args) {
     CHECK_EQ(mod->nm_register_func, nullptr);
     CHECK_NE(mod->nm_context_register_func, nullptr);
     Local<Value> unused = Undefined(env->isolate());
-    mod->nm_context_register_func(exports, unused,
-      env->context(), mod->nm_priv);
+    if (mod->nm_context_register_func != nullptr) {
+      mod->nm_context_register_func(exports, unused,
+        env->context(), mod->nm_priv);
+    }
     cache->Set(module, exports);
   } else if (!strcmp("module_v, "constants")) {
     exports = Object::New(env->isolate());
@@ -2684,7 +2714,19 @@ static void LinkedBinding(const FunctionCallbackInfo<Value>& args) {
                                     env->context(),
                                     mod->nm_priv);
   } else if (mod->nm_register_func != nullptr) {
+#ifdef ENABLE_NAPI
+    if (mod->nm_version != -1) {
+      mod->nm_register_func(exports, module, mod->nm_priv);
+    } else {
+      reinterpret_cast<node::addon_abi_register_func>(mod->nm_register_func)(
+          v8impl::JsEnvFromV8Isolate(v8::Isolate::GetCurrent()),
+          v8impl::JsValueFromV8LocalValue(exports),
+          v8impl::JsValueFromV8LocalValue(module),
+          mod->nm_priv);
+    }
+#else /* !defined ENABLE_NAPI */
     mod->nm_register_func(exports, module, mod->nm_priv);
+#endif /* def ENABLE_NAPI */
   } else {
     return env->ThrowError("Linked module has no declared entry point.");
   }
@@ -3668,6 +3710,8 @@ static void ParseArgs(int* argc,
       force_repl = true;
     } else if (strcmp(arg, "--no-deprecation") == 0) {
       no_deprecation = true;
```

Discussion: API Evolution

Option #1: Forward Compatibility

- Modules compiled with older versions continue to work in newer versions
- Modules dependent on newer APIs in new Node versions cannot work on older versions

Considerations:

Conditional source compilation

Backporting of API Stubs

Option #2: Backward Compatibility

- Modules can take advantage of newer APIs in new Node versions and can fallback to supporting older APIs without distributing multiple versions of the module.

Considerations:

Runtime API version checking

Extra level of indirection (perf impact unclear)

Key next steps

- Submit PRs for the following
  - Node version 8.0
  - Node version 6.9 LTS
  - Node-ChakraCore (after N-API lands in node master)
- Documentation
- Continue working on module ports to increase API coverage for Top 30 modules
- Evangelize and engage with native module developers to identify gaps