

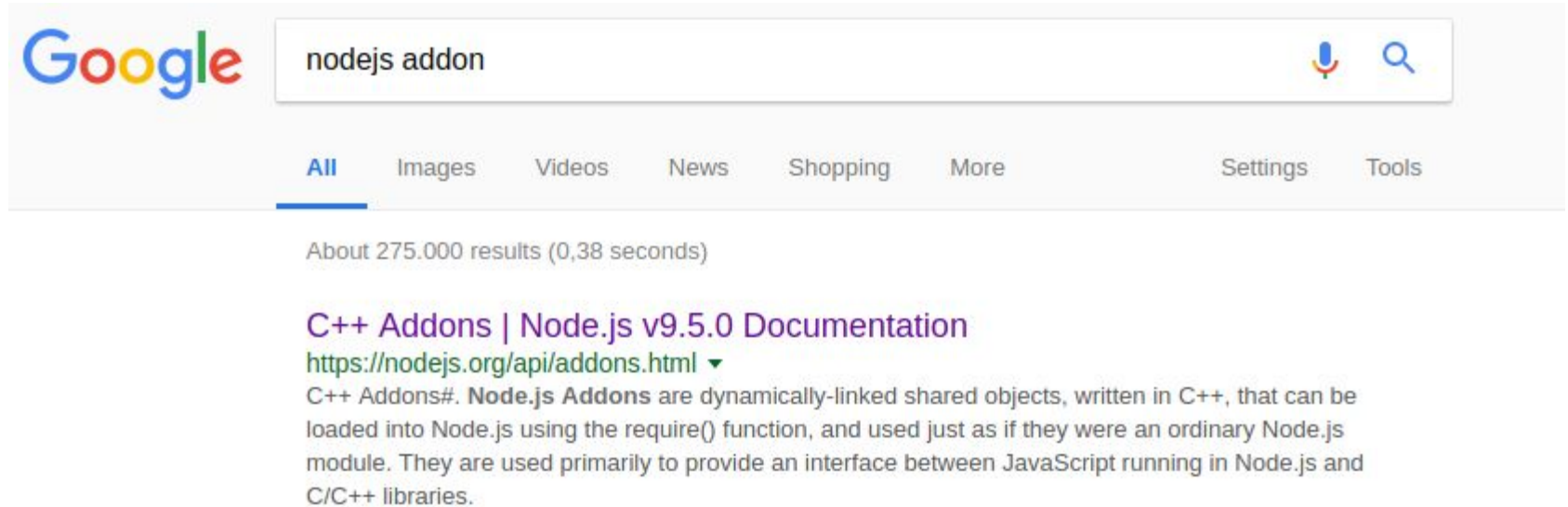
# Writing Node.js Modules in Rust

Farid Nouri Neshat  
Software Engineer @ Olindata

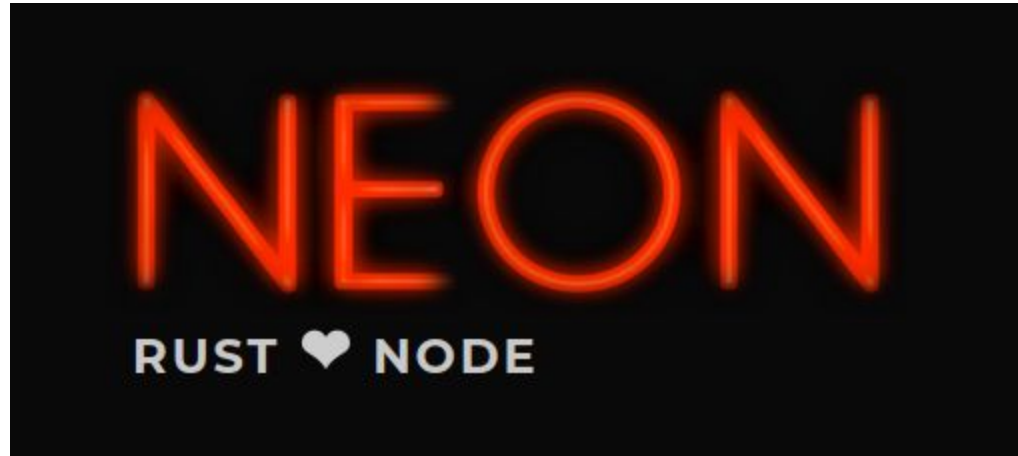
# You need a Node.js Native Addon because

- Performance
- Native Library
- Freedom in memory management
- OS Low level APIs
- You don't like Javascript!

# Well... How to write a node.js addon?



But you probably don't like C/C++ as well!



```
farid@Dellizium:~/repos/nr$ npm install neon-cli -g
/home/farid/.nvm/versions/node/v8.9.4/bin/neon -> /home/farid/.nvm/versions/node/v8.9.4/lib/node_modules/neon-cli/bin/cli.js
+ neon-cli@0.1.22
updated 1 package in 5.001s
farid@Dellizium:~/repos/nr$ neon new hi
This utility will walk you through creating the hi Neon project.
It only covers the most common items, and tries to guess sensible defaults.
```

Press ^C at any time to quit.

? **version** 0.1.0

? **description**

? **node entry point** lib/index.js

? **git repository**

? **author** Farid Neshat

? **email** FaridN\_SOAD@yahoo.com

? **license** MIT

Woo-hoo! Your Neon project has been created in: hi

The main Node entry point is at: hi/lib/index.js

The main Rust entry point is at: hi/native/src/lib.rs

To build your project, just run `npm install` from within the hi directory.  
Then you can test it out with `node -e 'require("./")'`.

Happy hacking!

```
farid@Dellizium:~/repos/nr$ cd hi/
```

```
farid@Dellizium:~/repos/nr/hi$ ls
```

```
lib  native  package.json  README.md
```

```
#[macro_use]
extern crate neon;

use neon::...

fn hello(call: Call) -> JsResult<JsString> {
    let scope = call.scope;
    Ok(JsString::new(scope, "hello node").unwrap())
}

register_module!(m, {
    m.export("hello", hello)
});
```

```
var addon = require('../native');

console.log(addon.hello());
```

```
farid@Dellizium:~/repos/nr/hi$ neon build
neon info running cargo
    Compiling hi v0.1.0 (file:///home/farid/repos/nr/hi/native)
    Finished release [optimized] target(s) in 0.36 secs
neon info generating native/index.node
farid@Dellizium:~/repos/nr/hi$ node lib/index.js
hello node
```

# Let's talk about performance

- Don't underestimate Javascript
- Try to minimize crossing the Rust-JS bridge
- Interacting with Javascript objects can potentially invoke Javascript
- Sometimes (node.js) buffers are faster than strings

```
fn get_object_sync(call: Call) -> JsResult<JsObject> {  
  
    let scope = call.scope;  
    let object = JsObject::new(scope);  
  
    try!(object.set("prop1", JsInteger::new(scope, 1)));  
    try!(object.set("prop2", JsInteger::new(scope, 2)));  
    try!(object.set("prop3", JsInteger::new(scope, 3)));  
  
    Ok(object)  
}
```

# Scope?

- Javascript is garbage collected
- V8 needs hints on objects created outside Javascript

```
fn hello(call: Call) -> JsResult<JsString> {  
    Ok(JsString::new(call.scope, "hello node").unwrap())  
}
```



# Buffers?

```
use neon::vm::{Lock, JsResult, Call};
use neon::js::binary::JsBuffer;

pub fn hello(call: Call) -> JsResult<JsBuffer> {
    let scope = call.scope;
    let mut arg_buf = call.arguments.require(scope, 0)?.check:::<JsBuffer>().unwrap();
    let arg = arg_buf.grab(|contents| contents.as_slice());
    let mut result_buf = JsBuffer::new(scope, arg.len() as u32)?;
    result_buf.grab(|mut contents| { contents.as_mut_slice().copy_from_slice(arg) });

    Ok(result_buf)
}
```

# Classes

```
pub struct Greeter {
  greeting: String,
}

declare_types! {

  /// A class for generating greeting strings.
  pub class JsGreeter for Greeter {
    init(call) {
      let scope = call.scope;
      let greeting = call.arguments.require(scope, 0)?.to_string(scope)?.value();
      Ok(Greeter {
        greeting: greeting
      })
    }

    method hello(call) {
      let scope = call.scope;
      let name = call.arguments.require(scope, 0)?.to_string(scope)?.value();
      let msg = call.arguments.this(scope).grab(|greeter| {
        format!("{}", {}, greeter.greeting, name)
      });
      Ok(JSString::new_or_throw(scope, &msg[..])?.upcast())
    }
  }
}
```

# Use Tasks for background jobs

```
impl Task for SuccessTask {  
    type Output = i32;  
    type Error = String;  
    type JsEvent = JsNumber;  
  
    fn perform(&self) -> Result<Self::Output, Self::Error> {  
        Ok(CalculateFibonacci())  
    }  
  
    fn complete<'a, T: Scope<'a>>(  
        self,  
        scope: &'a mut T,  
        result: Result<Self::Output, Self::Error>,  
    ) -> JsResult<Self::JsEvent> {  
        Ok(JsNumber::new(scope, result.unwrap() as f64))  
    }  
}
```






















```
pub fn perform_async_task(call: Call) -> JsResult<JsUndefined> {  
    let f = call.arguments.require(call.scope, 0)?  
        .check::<JsFunction>()?;  
    SuccessTask.schedule(f);  
    Ok(JsUndefined::new())  
}
```

## One more tip: Try neon-serde!

```
export! {  
  fn say_hello(name: String) -> String {  
    format!("Hello, {}!", name)  
  }  
  
  fn greet(user: User) -> String {  
    format!("{}", user.name, user.age)  
  }  
  
  fn fibonacci(n: i32) -> i32 {  
    match n {  
      1 | 2 => 1,  
      n => fibonacci(n - 1) + fibonacci(n - 2)  
    }  
  }  
}
```

# Who uses neon?

## packages depending on 'neon-cli'

 <b>neon-sled</b> \$ cargo check \$ neon build \$ 0.1.3 published 4 months ago by matthiasn	 <b>noise-search</b> Noise for Node.js 0.7.4 published 2 months ago by vmx	 <b>@vladikoff/test-neon-csv-reader</b> 0.1.2 published 2 months ago by vladikoff
 <b>node-reconfix</b> A Node.js wrapper for the Reconfix library. 0.1.1 published 2 months ago by abrodersen	 <b>rustest</b> 0.1.0 published 2 months ago by apro	 <b>koa-rust-router</b> koa router written in rust. 0.1.0 published 2 months ago by moldray
 <b>honey-badger-cli</b> A command-line interface for HoneyBadger, a W... 0.0.3 published a year ago by cmtt	 <b>ratel</b> A command-line interface for HoneyBadger, a W... 1.0.1 published a year ago by terhix	 <b>base-x-native</b> Fast base encoding / decoding of any given alph... 0.1.6 published 12 months ago by orkon
 <b>rust-markdown</b> Render Markdown to HTML using native Rust 0.1.0 published a year ago by adrian.arroyocalle	 <b>@paulcbetts/test-neon</b> 0.1.0 published 10 months ago by paulcbetts	 <b>neon-hello</b> a 'hello world' demo of Neon 0.1.1 published 9 months ago by dherman
 <b>rustapable</b> Rust based tapable built with neon 0.0.0-dev published 7 months ago by sendilkumarn	 <b>nodecrypton</b> A NodeJS + Rust library that leverages on GPG t... 0.1.4 published 4 months ago by jbitshine	 <b>voxel-worldgen</b> A voxel world generator written in Rust, with bin... 1.1.0 published 2 years ago by mhsjlw
 <b>heimdal</b> Safe, yet incredibly fast cryptography for Node ... 0.3.1 published 4 weeks ago by kenansulayman	 <b>libsodium-neon</b> Node.js bindings to rust_sodium. 2.1.2 published 3 weeks ago by wireapp-owner	 <b>flow-bin-ci</b> Tool for flow ci 0.1.1 published 2 months ago by szagi3891
 <b>sourcemap-decoder</b> Node binding for https://github.com/getsentry/... 1.0.2 published a week ago by brooooooklyn	 <b>@iml/node-libzfs</b> Neon bindings to libzfs 0.1.10 published 5 days ago by iml-team	 <b>reconfix-preview</b> A Node.js wrapper for the Reconfix library. 0.2.1 published 4 days ago by abrodersen

# Links

- <https://neon-bindings.com>
- <https://github.com/GabrielCastro/neon-serde>
- Examples:
  - <https://github.com/dherman/neon-examples>
  - <https://github.com/nswbmw/neon-fibonacci>
  - <https://github.com/nswbmw/neon-class>
  - <https://github.com/neon-bindings/neon/issues/260>
  - <https://github.com/neon-bindings/neon/tree/master/test/dynamic/native/src/js>

Questions?

# One last thing

- All credits goes to the Neon maintainer Dave Herman and the Neon contributors and community.