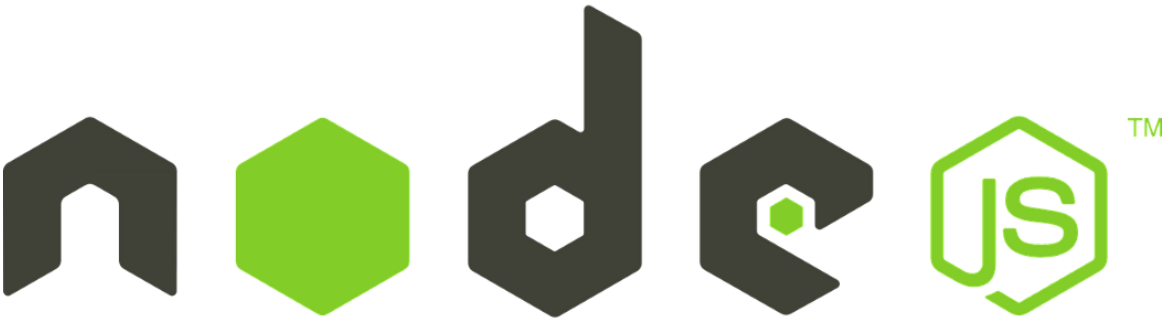
Node.Js Under the Hood

What makes Node.Js possible?

Salil Arora Follow
Aug 11, 2019 · 5 min read



Misconception

Node.Js is just another front-end browser based JavaScript framework, like Angular JS and React JS.

Reality

Node.Js is *cross-platform JavaScript run-time environment* that executes JavaScript code *outside of a browser for server-side scripting*.

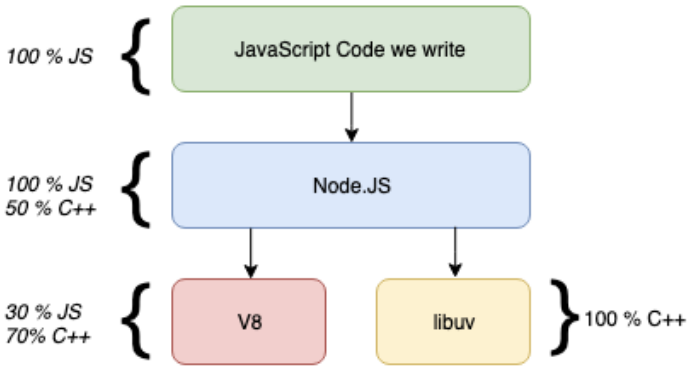
Is that enough? Hell no!

It is not as simple as it looks to write JavaScript code and have your code be magically compiled and executed by the operating system, and then have it listen to the http requests and lots of other server side jobs.

The most important thing about the operating system is that it doesn't even know about JavaScript or Node.Js. It only understands C++, which might give you a sweet memory of your college exam all-nighters!

So, you might be wondering, how does Node.Js execute JavaScript code on the operating system? Is Node.Js a "*Satan*" or "*Savior*"?

What does Node.Js do? Take a look at the image below:



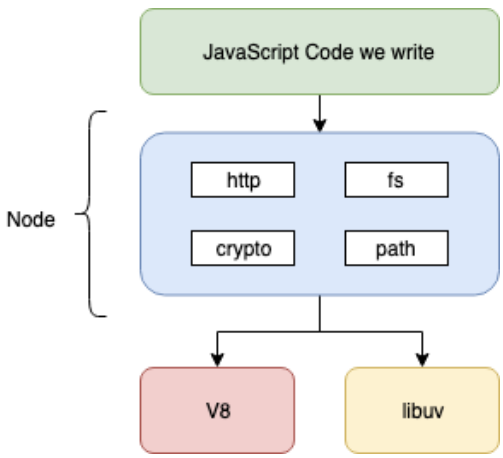
Do V8 and libuv looks like big word alerts?

Let's understand them.

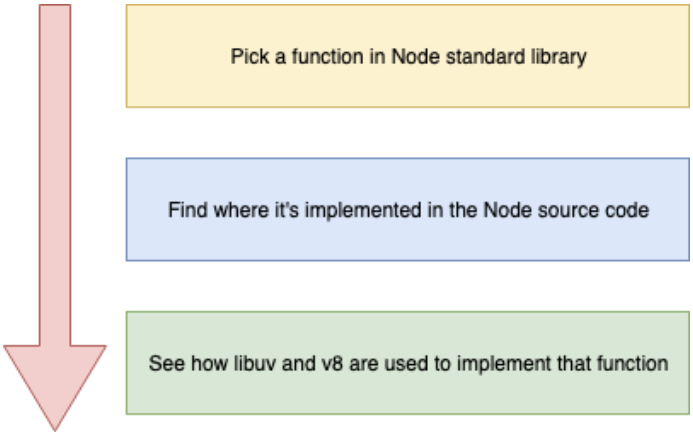
There are three realms in the Node.Js Application:

1. JavaScript code that we write.
2. V8 is developed by Google and it basically compiles JavaScript Code directly to native C++ code before executing it.
   libuv is a multi-platform C++ library that provides support for asynchronous I/O based operations such as file system, networking and concurrency.
3. Node.Js world, which acts as an interface that executes the JS code that weve written with the help of C++ and gives us access to the APIs that are written in libuv and v8 libraries

The picture below will help you understand how these three components interact and make Node.Js possible. It also showcases how Node.Js integrates some common modules, like http, fs, crypto, etc, that are actually written in the libuv library and provide the APIs to JavaScript.

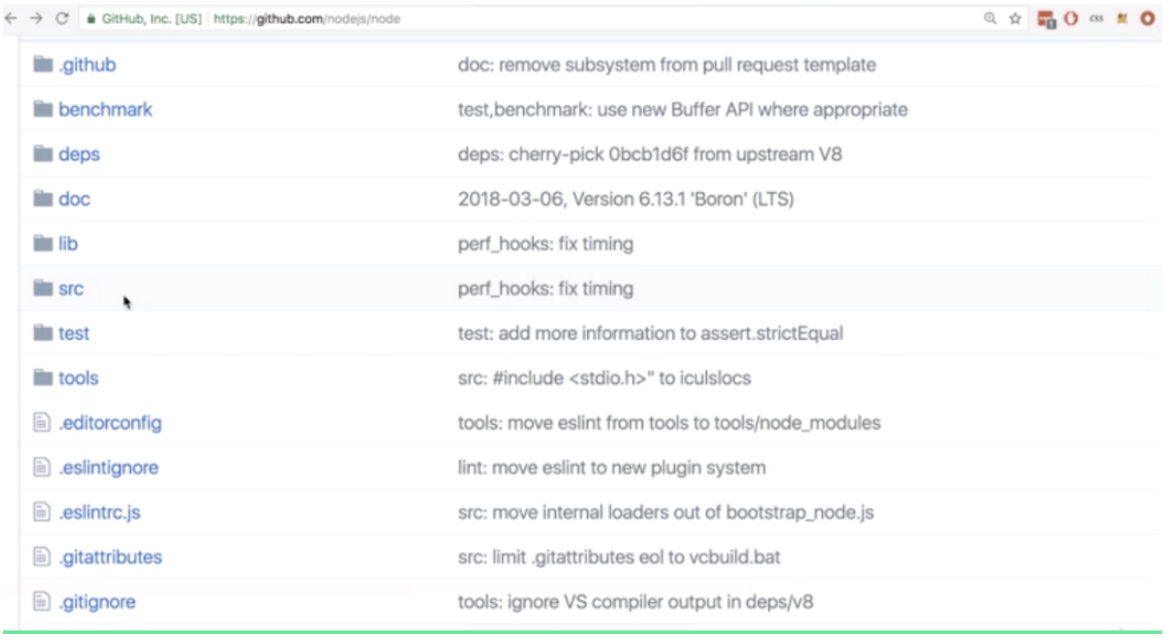


Let’s deep dive into this concept to try and understand things better.



We’re going to check Node.Js’s crypto library.

In the picture below you can see how the directory is structured for Node.Js project on GitHub.



There are two main directories:

1. **lib** is the js realm which we get access to while developing our applications.

2. **src** is the c++ world which internally uses libuv and v8 libraries.

We’re taking a simple example of pbkdf2 library inside crypto which is used for password encryption.

Our main goal is to understand how Node.Js connects JS code with C++ and internally using libuv and v8 libraries for doing things.

Follow these tips to ensure you don't go to far astray from the subject of this piece:

- Don’t get into the details of how this pbkdf2 method is working.
- Don’t be shocked to see the C++ code.
- Don’t get caught up in the literal usage of **libuv** and **v8** libraries.

Below is the JS version of pbkdf2 of crypto library.

The method below, written in JavaScript in the lib directory pbkdf2, calls_pbkdf2 of the src directory, which is written in c++:

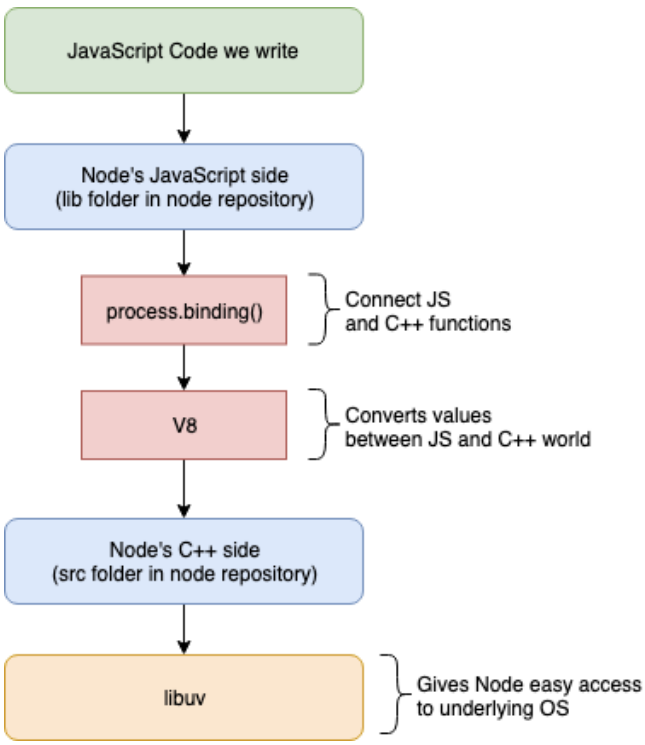
```
const { pbkdf2: _pbkdf2 } = internalBinding('crypto');

function pbkdf2(password, salt, iterations, keylen, digest,
callback) {
  if (typeof digest === 'function') {
    callback = digest;
    digest = undefined;
  }

  ({ password, salt, iterations, keylen, digest } =
    check(password, salt, iterations, keylen, digest));

  if (typeof callback !== 'function')
    throw new ERR_INVALID_CALLBACK(callback);
  //calling the src c++ version of pbkdf2
  handleError(_pbkdf2(keybuf, password, salt, iterations, digest,
wrap),
              digest);
}
```

This diagram explains exactly what is happening:



`Process.binding` is a bridge where the JavaScript world connects to the C++ code.

Below is the c++ version of the crypto library.

The method below in the node_crypto.cc C++ file of src directory is called by the above pbkdf2 of the lib directory.

SetMethod is exporting the method pbkdf2 as “PBKDF2” to the outside world.

```
inline void PBKDF2(const FunctionCallbackInfo<Value>& args) {
  auto rv = args.GetReturnValue();

  CHECK(args[4]->IsString());  // digest_name
  std::unique_ptr<PBKDF2Job> job(new PBKDF2Job(env));

  env->PrintSyncTrace();
  job->DoThreadPoolWork();
  rv.Set(job->ToResult());
}
env->SetMethod(target, "pbkdf2", PBKDF2);
env->SetMethod(target, "generateKeyPairRSA", GenerateKeyPairRSA);
```

Always remember the *don’ts*; we are learning Node.Js, *not* the crypto library.

So where are the much hyped **libuv** and **v8** libraries?

```
using v8::Array;
using v8::Boolean;
using v8::Exception;
using v8::External;
using v8::False;
using v8::Function;
using v8::Int32;
using v8::String;

#include <uv.h>

int main() {
  loop = uv_default_loop();

  uv_tcp_t server;
  uv_tcp_init(loop, &server);

  struct sockaddr_in bind_addr = uv_ip4_addr("0.0.0.0", 7000);
  uv_tcp_bind(&server, bind_addr);
  int r = uv_listen((uv_stream_t*) &server, 128, on_new_connection);
  if (r) {
    fprintf(stderr, "Listen error!\n");
    return 1;
  }
  return uv_run(loop, UV_RUN_DEFAULT);
}
```

This libuv snippet is initialising a new tcp connection. As discussed earlier it can do tasks related to OS.

This V8 snippet is importing c++ definition of JS objects such as Array, and Boolean. V8 engine translates JS objects into their C++ equivalents.

. . .

What's Next?

There are many more topics we need to discuss in detail. Follow me and keep an eye on this place. In the next few pieces I will write in detail about the following topics:

- How the Node Event loop works internally
- Is Node.Js really single threaded?
- Hapi or Express for your startup?

Nodejs JavaScript Technology Programming Web Development