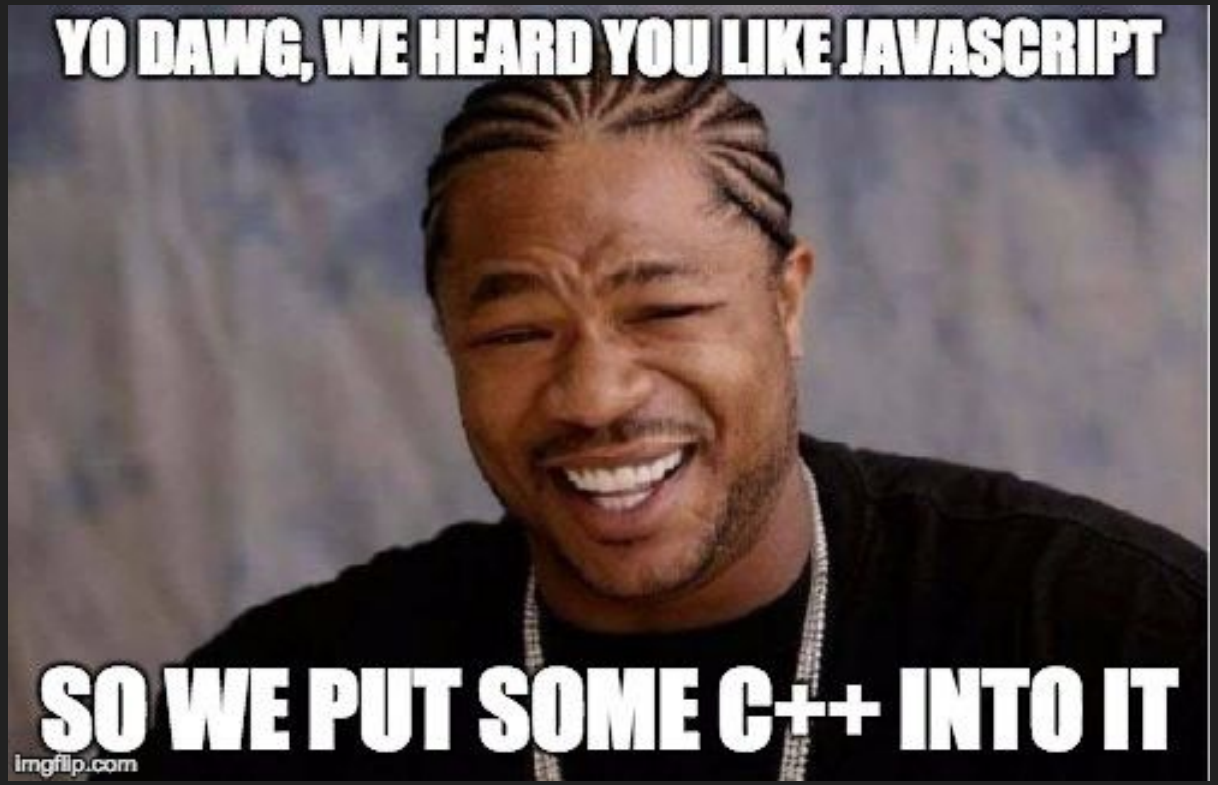
C++ и Node JS

Или как ~~освятить~~ ускорить JS

КДПВ


YO DAWG, WE HEARD YOU LIKE JAVASCRIPT
SO WE PUT SOME C++ INTO IT
imgflip.com

Зачем такое делать?

1. Получить доступ к ~~нормальным~~ нативным библиотекам из JS (биндинги)
2. Повысить производительность отдельных участков кода*
3. Просто пофаниться

Как это сделать?

- Напрямую использовать V8 API
- Использовать NAN
- Использовать N-API + C++ сахар

V8 API

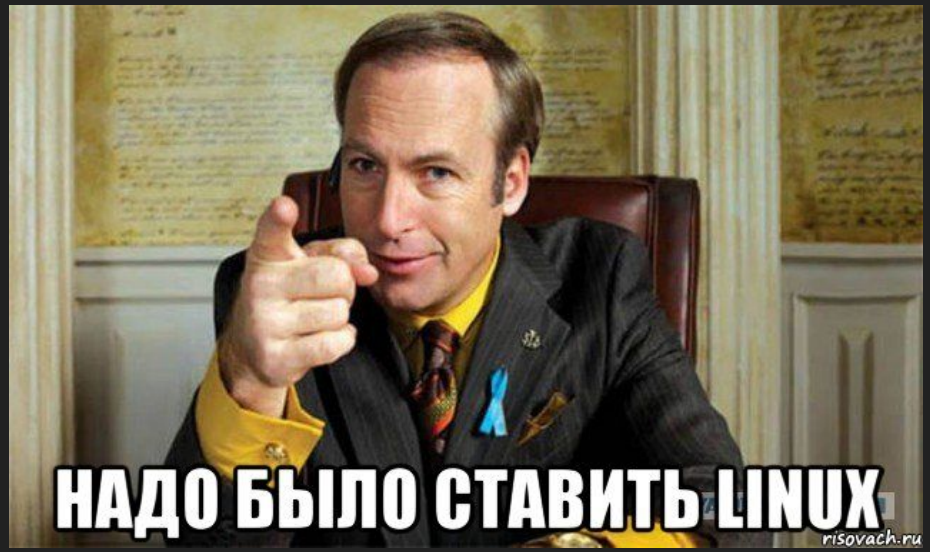
- Наиболее старый и примитивный способ
- Сложности совместимости между версиями
- #include <node.h> и всё. К сожалению.
- Никто в здравом уме не использует

Native Abstractions for Node.js (NAN)

- npm install nan
- NAN != NaN
- Поддерживают совместимость с разными версиями V8
- Наиболее используемый способ в настоящее время (к сожалению)

Требования к системе

1. Компилятор C++ (gcc, clang, msvc, etc.)
2. Python 2.7
3. make
4. node-gyp
5. NAN



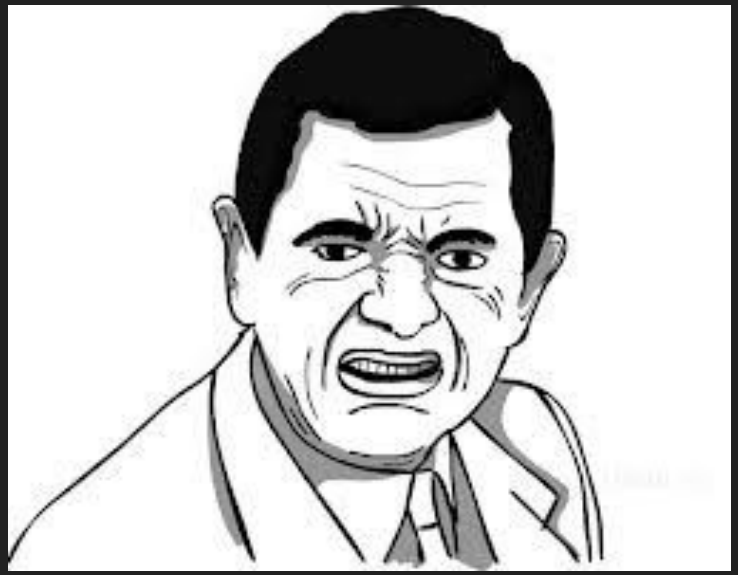
package.json

```
{
  "name": "addon_name",
  "version": "1.0.0",
  "dependencies": {
    "nan": "^2.6.1",
    "node-gyp": "^3.6.0"
  },
  "scripts": {
    "compile": "node-gyp rebuild",
    "start": "node main.js"
  },
  "gypfile": true
}
```

bindings.gyp

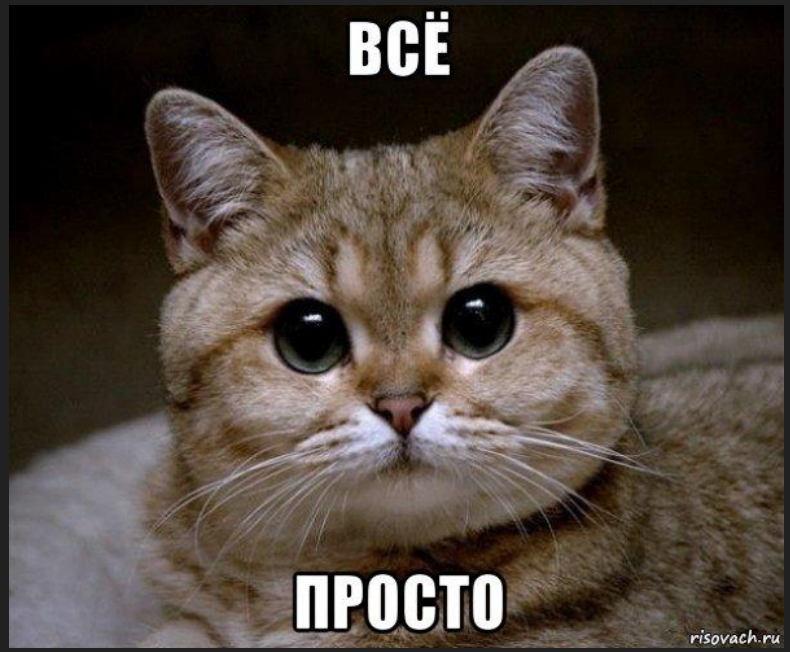
```
{
  "targets": [
    {
      "include_dirs": [
        "<!(node -e \"require('nan')\")"
      ],
      "target_name": "addon",
      "sources": [ "main.cpp" ]
    }
  ]
}
```

```
#include <nan.h>

NAN_METHOD(IsPrime) {
    if (!info[0]->IsNumber()) {
        Nan::ThrowTypeError("argument must be a number!");
        return;
    }
    int number = (int) info[0]->NumberValue();
    if (number < 2) {
        info.GetReturnValue().Set(Nan::False());
        return;
    }
    for (int i = 2; i < number; i++) {
        if (number % i == 0) {
            info.GetReturnValue().Set(Nan::False());
            return;
        }
    }
    info.GetReturnValue().Set(Nan::True());
}


NAN_MODULE_INIT(Initialize) {
    NAN_EXPORT(target, IsPrime);
}
NODE_MODULE(addon, Initialize);
```



Сборка && запуск

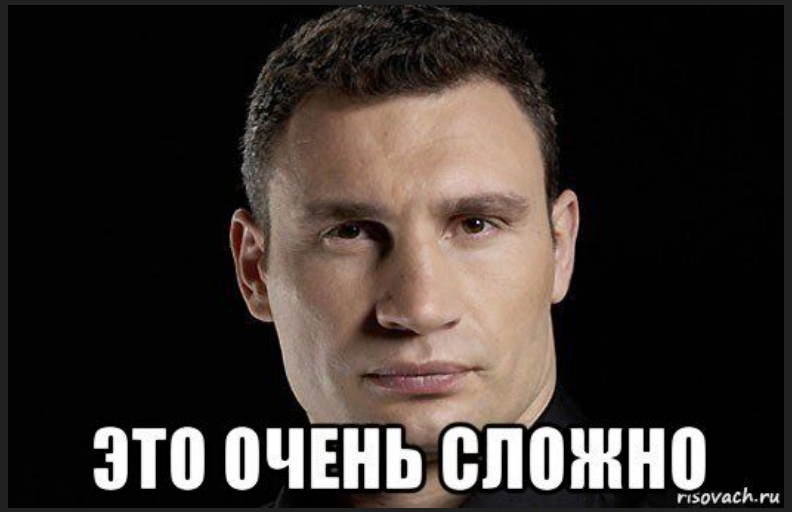
- npm run compile
- node index.js
- …
- Profit!



```
js: 3094.509ms
➜  node-cpp-addons git:(master) ✗ node main.js
native: checking whether 654188429 is prime... true
native: 1793.347ms

js: checking whether 654188429 is prime... true
js: 3091.484ms
➜  node-cpp-addons git:(master) ✗ node main.js
native: checking whether 654188429 is prime... true
native: 1801.630ms

js: checking whether 654188429 is prime... true
js: 3097.912ms
➜  node-cpp-addons git:(master) ✗ node main.js
native: checking whether 654188429 is prime... true
native: 1800.728ms

js: checking whether 654188429 is prime... true
js: 3111.480ms
➜  node-cpp-addons git:(master) ✗ node main.js
native: checking whether 654188429 is prime... true
native: 1812.243ms

js: checking whether 654188429 is prime... true
js: 3100.775ms
➜  node-cpp-addons git:(master) ✗ node main.js
native: checking whether 654188429 is prime... true
native: 1789.004ms

js: checking whether 654188429 is prime... true
js: 3099.338ms
➜  node-cpp-addons git:(master) ✗ node main.js
native: checking whether 654188429 is prime... true
native: 1800.620ms

js: checking whether 654188429 is prime... true
js: 3091.217ms
➜  node-cpp-addons git:(master) ✗ node main.js
native: checking whether 654188429 is prime... true
native: 1785.044ms

js: checking whether 654188429 is prime... true
js: 3093.411ms
➜  node-cpp-addons git:(master) ✗ node main.js
native: checking whether 654188429 is prime... true
native: 1799.313ms

js: checking whether 654188429 is prime... true
js: 3102.635ms
➜  node-cpp-addons git:(master) ✗ node main.js
native: checking whether 654188429 is prime... true
native: 1785.371ms

js: checking whether 654188429 is prime... true
js: 3090.808ms
➜  node-cpp-addons git:(master) ✗
```

Бенчмарки? А не будет!

- Сильно зависит от кейса
- Результаты могут разниться очень сильно: от тотального проигрыша JS до выигрыша (при определённых условиях)
- Бенчмаркинг - краааааааааааайне сложная штука



N-API

- Новый подход (с Node 10)
- Цель: сделать единое API и ABI, которое не будет ломаться от релиза к релизу
- Цель покруче: сделать эту вещь не просто для V8, а сразу для всех движков!

```c
#include <node_api.h>
#include <stdio.h>

napi_value print (napi_env env, napi_callback_info info) {
  napi_value argv[1];
  size_t argc = 1;
  napi_get_cb_info(env, info, &argc, argv, NULL, NULL);
  if (argc < 1) {
    napi_throw_error(env, "EINVAL", "Too few arguments");
    return NULL;
  }
  char str[1024];
  size_t str_len;
  if (napi_get_value_string_utf8(env, argv[0], (char *) &str, 1024, &str_len) != napi_ok) {
    napi_throw_error(env, "EINVAL", "Expected string");
    return NULL;
  }
  printf("Printed from C: %s\n", str);
  return NULL;
}
napi_value init_all (napi_env env, napi_value exports) {
  napi_value print_fn;
  napi_create_function(env, NULL, 0, print, NULL, &print_fn);
  napi_set_named_property(env, exports, "print", print_fn);
  return exports;
}
NAPI_MODULE(NODE_GYP_MODULE_NAME, init_all)
```

Вывод

- На C++ ~~нужно~~ можно писать везде :-)
- Иногда это даже может пригодиться
- Приходится писать много boiler-plate (но это дело исправляется)
- Всё больше и больше стабильности в API и ABI с каждым днём
- **Можем** выиграть в производительности
- Не стоит заниматься таким без крайней необходимости

Ссылки

- https://nodejs.org/api/addons.html
- https://github.com/nodejs/node-gyp
- https://github.com/nodejs/nan
- Best OS in the world:
  - https://gentoo.org/
  - https://www.ubuntu.com/
  - https://getfedora.org/
  - http://www.linuxfromscratch.org/lfs/
  - https://www.archlinux.org/





Спасибо за внимание!

И используйте только православные технологии.