Node.js

Alexander Lobashev, RaiffeisenBank
t.me/alobashev

http://www.devconf.ru

Problems

EventLoop contradiction?

How does JS and C++ code exist?

How works require()?

Architecture

Event Loop, I/O

Modules

Architecture

Event Loop, I/O

Modules

Node is a js runtime platform

Written in C, C++ and JS

Built on Chrome V8 Engine

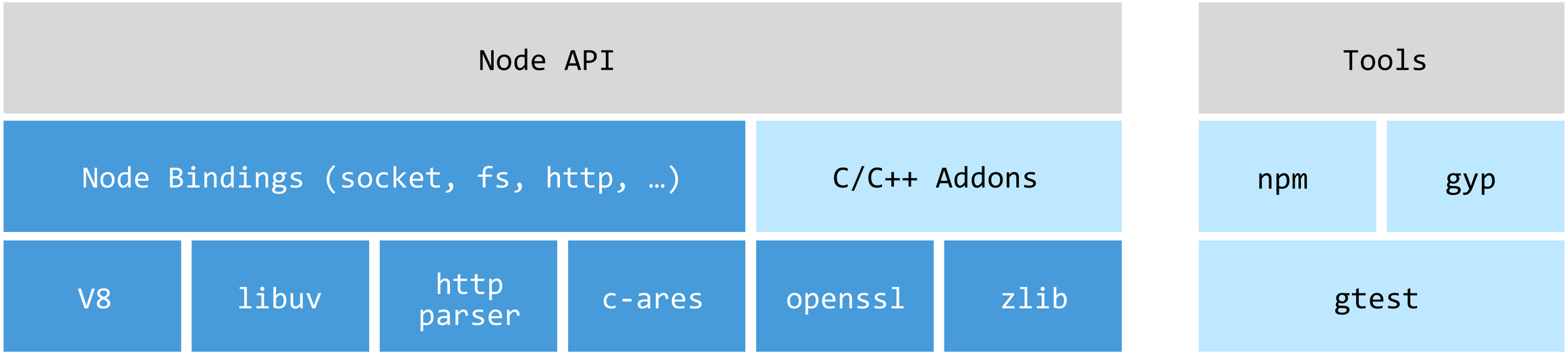
Uses libuv for I/O

Used everywhere (web, mobile, robots, IoT, …)

Single-Thread

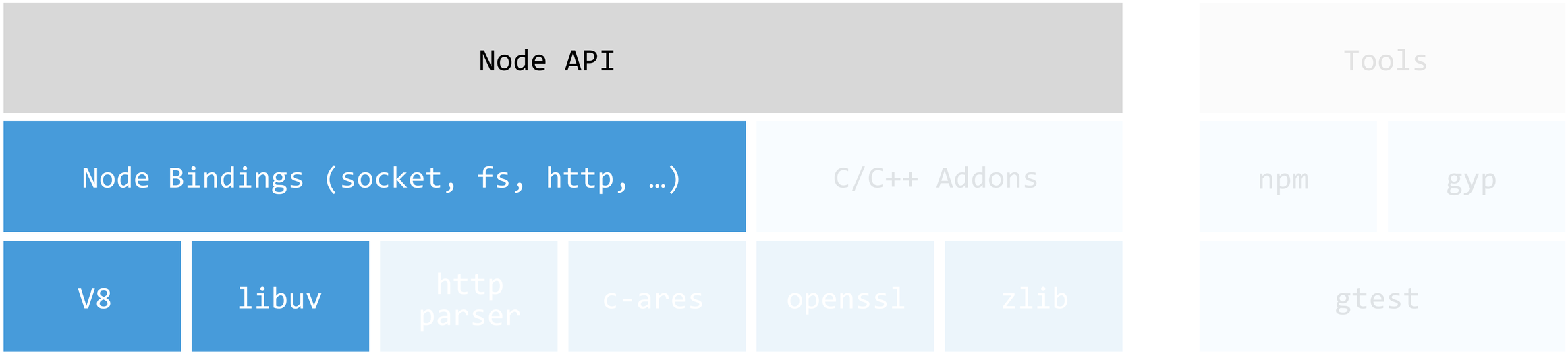
Event-Driven Model

Non-Blocking I/O



https://nodejs.org/en/docs/meta/topics/dependencies *

Райффайзен БАНК


Node API
Tools
Node Bindings (socket, fs, http, …)
C/C++ Addons
npm
gyp
V8
libuv
http parser
c-ares
openssl
zlib
gtest

Written C, C++

Bridge System C++ and JS

Drop custom EventLoop

…

C++ > V8 > Run JS

```
// Создаем новый контекст
Local<Context> context = Context::New(isolate);

// Делаем этот контекст активным для компиляции и запуска кода JavaScript
Context::Scope context_scope(context);

// Создаем новую строку из исходного кода JavaScript
// Он может быть жестко запрограммирован, считан из файла или любым другим способом.
Local<String> source = String::NewFromUtf8(isolate, "'Hello' + ', World!'",
NewStringType::kNormal).ToLocalChecked();

// Компилируем наш JavaScript
Local<Script> script = Script::Compile(context, source).ToLocalChecked();

// Запускаем скомпилированный код и ожидаем результатов
Local<Value> result = script->Run(context).ToLocalChecked();

// Преобразуем результат в utf8 и выводим
String::Utf8Value utf8(result);
printf("%s\n", *utf8);
```

* https://chromium.googlesource.com/v8/v8/+/branch-heads/5.8/samples/hello-world.cc
* https://gist.github.com/netpoetica/28ce31478cfc43edcaa7

Templates

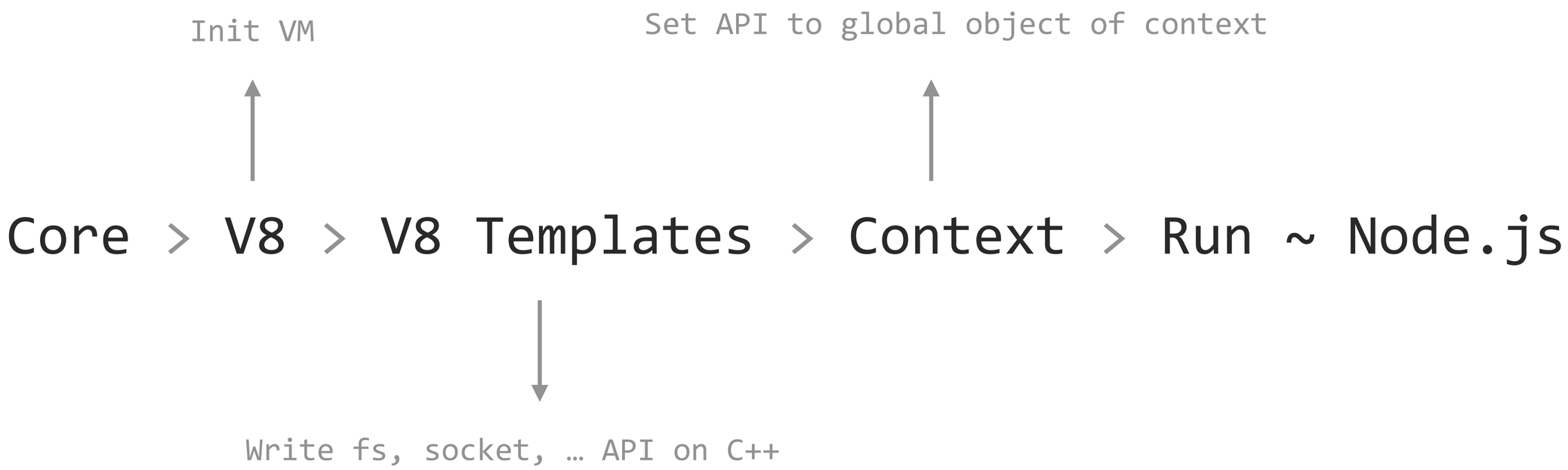
Bridge C++ code to JS (e.g. DOM)

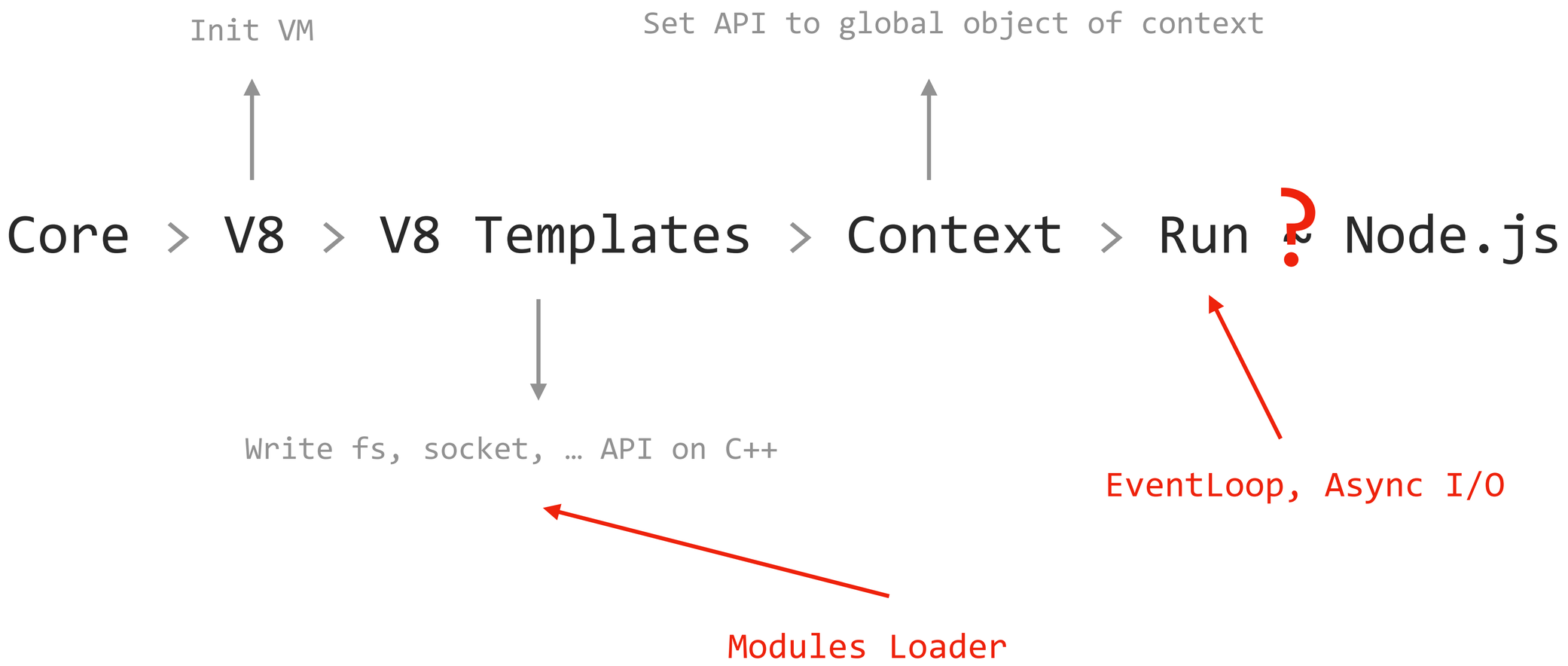
Function Templates

Object Templates

```cpp
// Функция LogCallback
static void LogCallback(const v8::FunctionCallbackInfo<v8::Value>& args) {
    // Некоторый код
}

// Создаем новый ObjectTemplate
Local<ObjectTemplate> global = ObjectTemplate::New(isolate);

// Создаем новый FunctionTemplate и связываем C++ функцию LogCallback с ним
// Добавляем в global наш FunctionTemplate
global->Set(
    String::NewFromUtf8(isolate, "log"),
    FunctionTemplate::New(isolate, LogCallback)
);

// Передаем переменную global в новый контекст js, и мы можем вызвать функцию из глобальной области
// В результате будет запущен метод C++ из js.
Persistent<Context> context = Context::New(isolate, NULL, global);
```

* https://github.com/nodejs/node/blob/master/deps/v8/samples/process.cc


Init VM
Set API to global object of context
Core > V8 > V8 Templates > Context > Run ~ Node.js
Write fs, socket, … API on C++


Init VM
Set API to global object of context
Core > V8 > V8 Templates > Context > Run ? Node.js
Write fs, socket, … API on C++
EventLoop, Async I/O
Modules Loader

Architecture

Event Loop, I/O

Modules

Single-Thread

Event-Driven Model

Non-Blocking I/O

Single Thread

Event loop

Node API

Tools

C/C++ Addons

npm

gyp

libuv

http parser

c-ares

openssl

zlib

gtest

Райффайзен БАНК

Event loop backed by epoll, kqueue, IOCP, event ports.

Asynchronous TCP and UDP sockets

Asynchronous operations FS and support events FS

Child processes

Thread pool

Signal handling

High resolution clock

Loop in libuv

```c
#include <stdio.h>
#include <stdlib.h>
#include <uv.h>

int main() {
    uv_loop_t *loop = malloc(sizeof(uv_loop_t));
    uv_loop_init(loop);

    printf("Now quitting.\n");
    uv_run(loop, UV_RUN_DEFAULT);

    uv_loop_close(loop);
    free(loop);
    return 0;
}
```

```c
int uv_run(uv_loop_t* loop, uv_run_mode mode) {
  // Ищем активных обработчиков или запросов
  r = uv__loop_alive(loop);
  if (!r)
    uv__update_time(loop);

  while (r != 0 && loop->stop_flag == 0) {
    // Обновляем время выполнения
    uv__update_time(loop);
    // Запуск таймеров (uv_timer_t)
    uv__run_timers(loop);
    // Запуск ожидающих коллбеков из очереди
    ran_pending = uv__run_pending(loop);

    // Особые системные обработчики
    uv__run_idle(loop);
    uv__run_prepare(loop);

    // Выполнение операций I/O
    uv__io_poll(loop, timeout);

    // Особые обработчики (setImmediate)
    uv__run_check(loop);
    // onClose обработчики
    uv__run_closing_handles(loop);
  }

  if (loop->stop_flag != 0)
    loop->stop_flag = 0;
```

```
while there are still events to process:
    e = get the next event
    if there is a callback associated with e:
        call the callback
```

Event loop

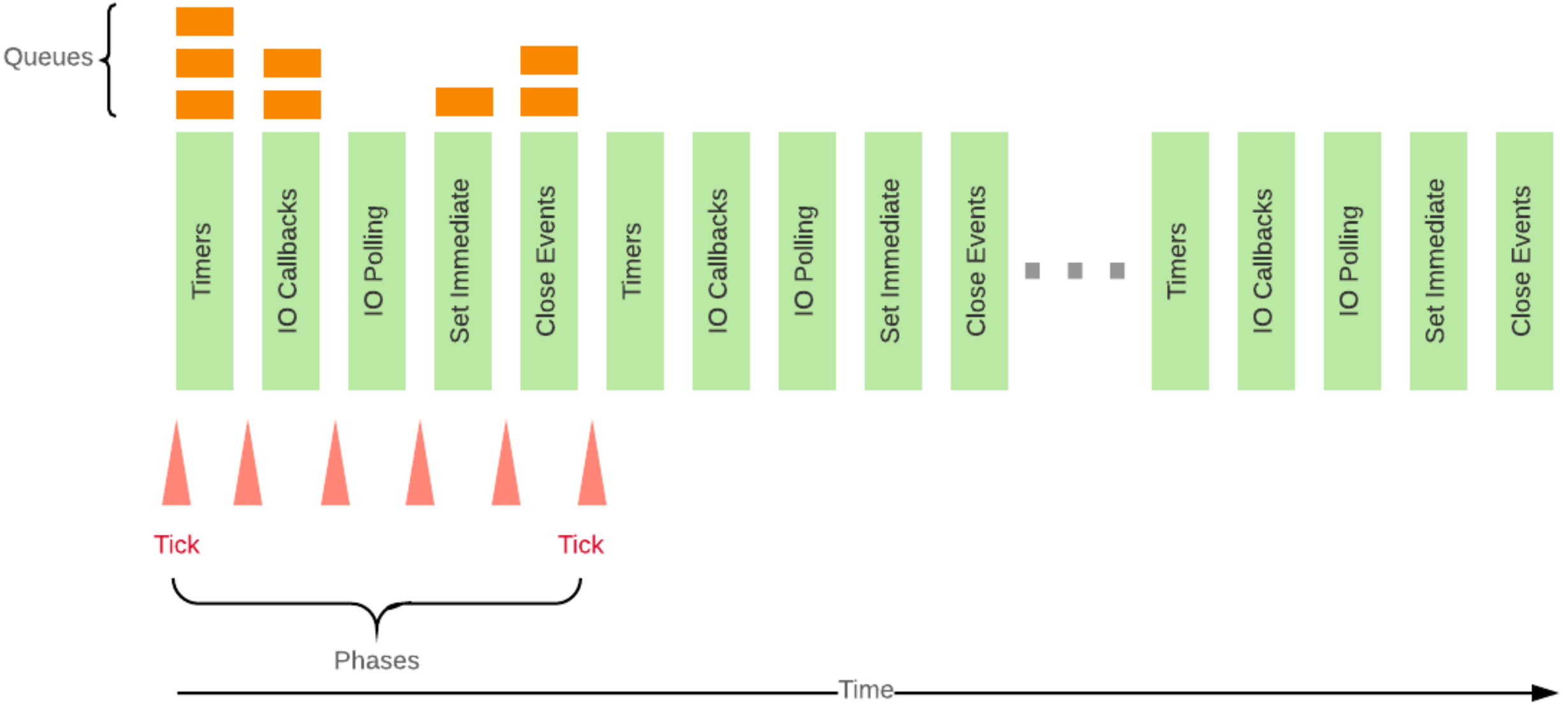
Init with start Node

Use main thread

Handler multiple concurrent clients

The code execution mechanism

Run after exec main module



Райффайзен БАНК

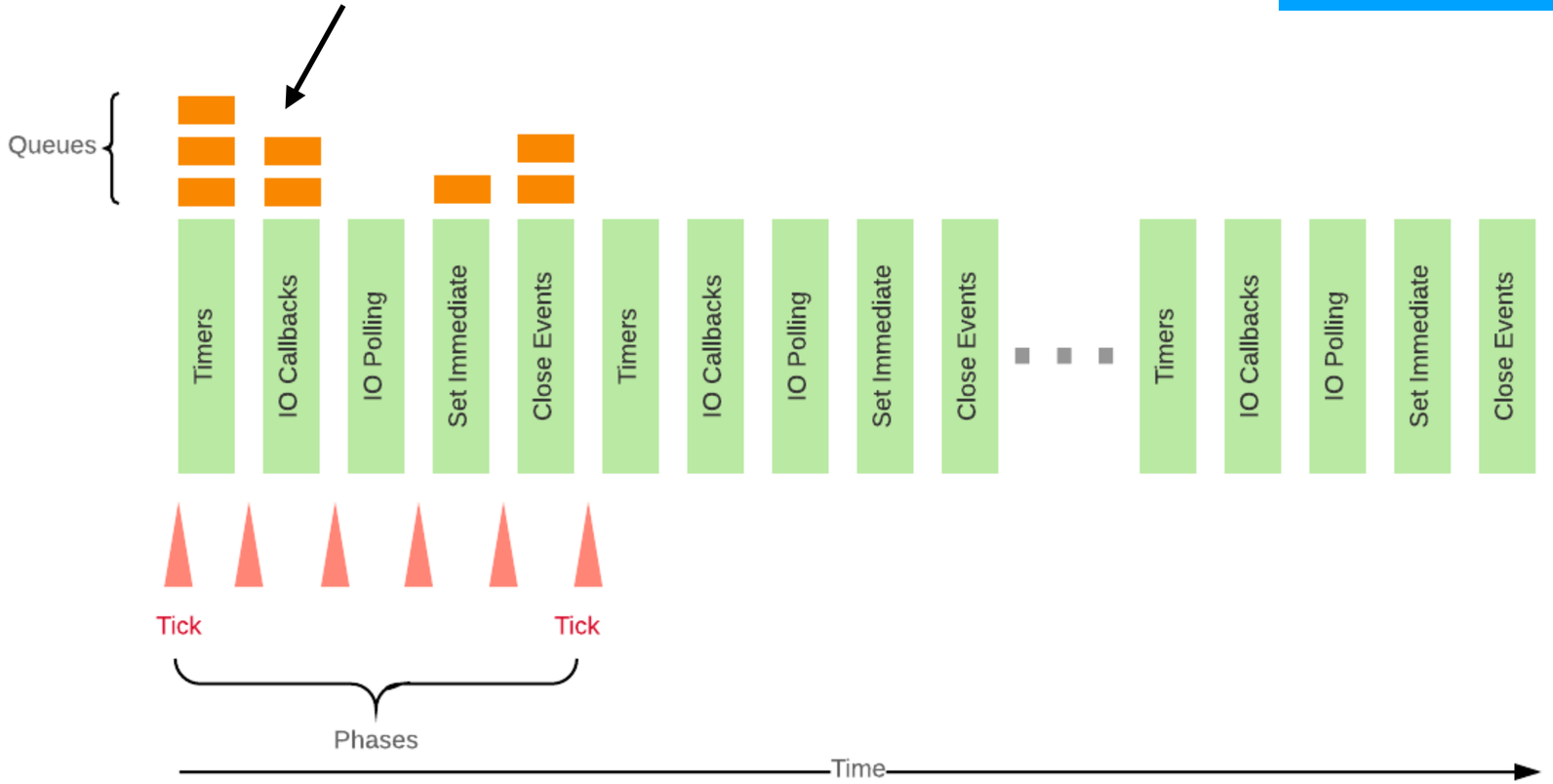
```
// a.js
module.exports = () => {
    console.log('module a');

    require('fs').readFile(__filename, () => {
        setTimeout(() => {
            console.log('timeout');
        }, 0);

        setImmediate(() => {
            console.log('immediate');
        });
    });
};

// main.js
const a = require('./a');

console.log('Example');
a();
```

```
> Example
> module a
> immediate
> timeout
```

Run module a.js (console.log + readFile)

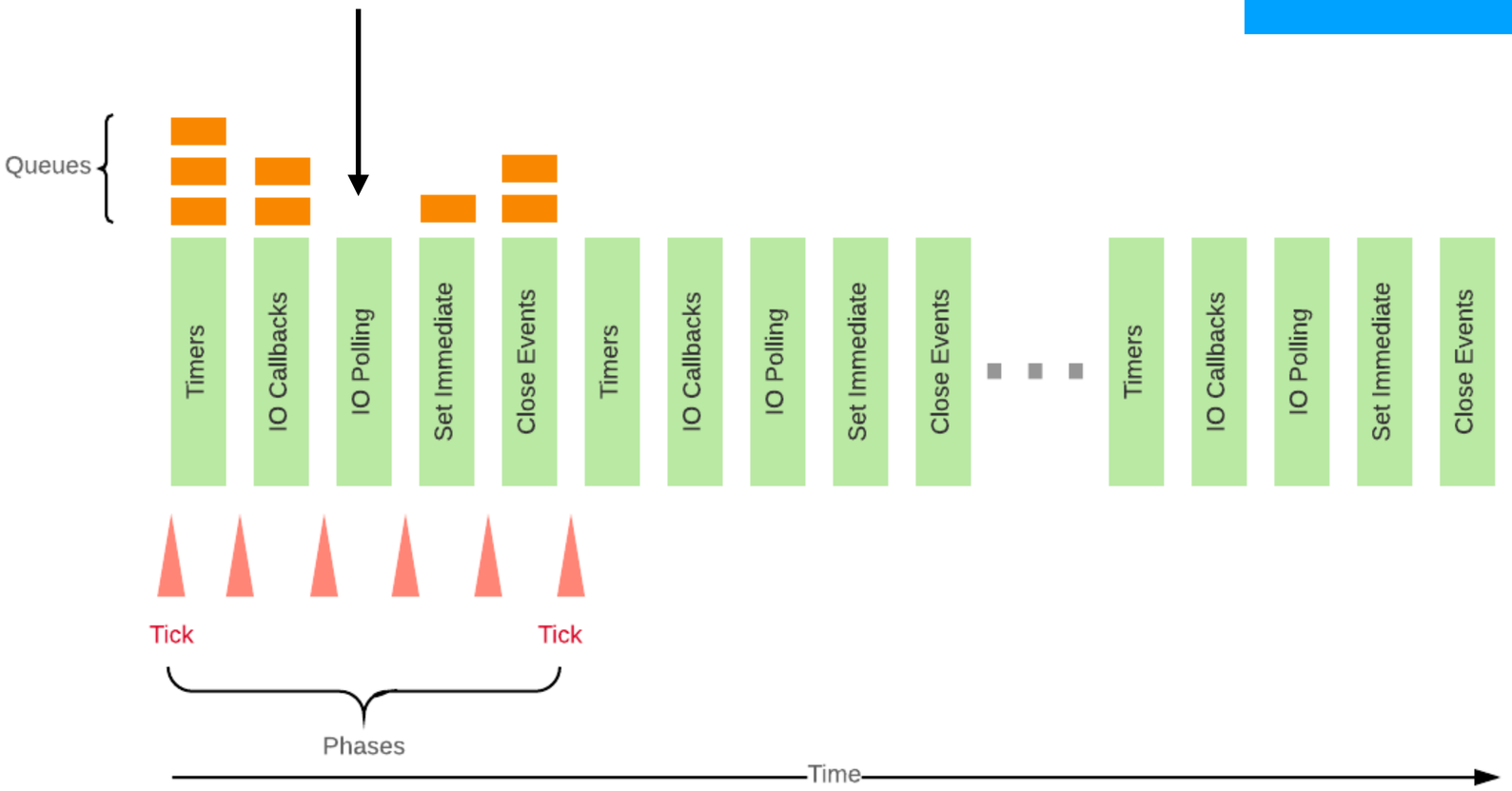
1 Iteration

Main file
> log: example



signal FS

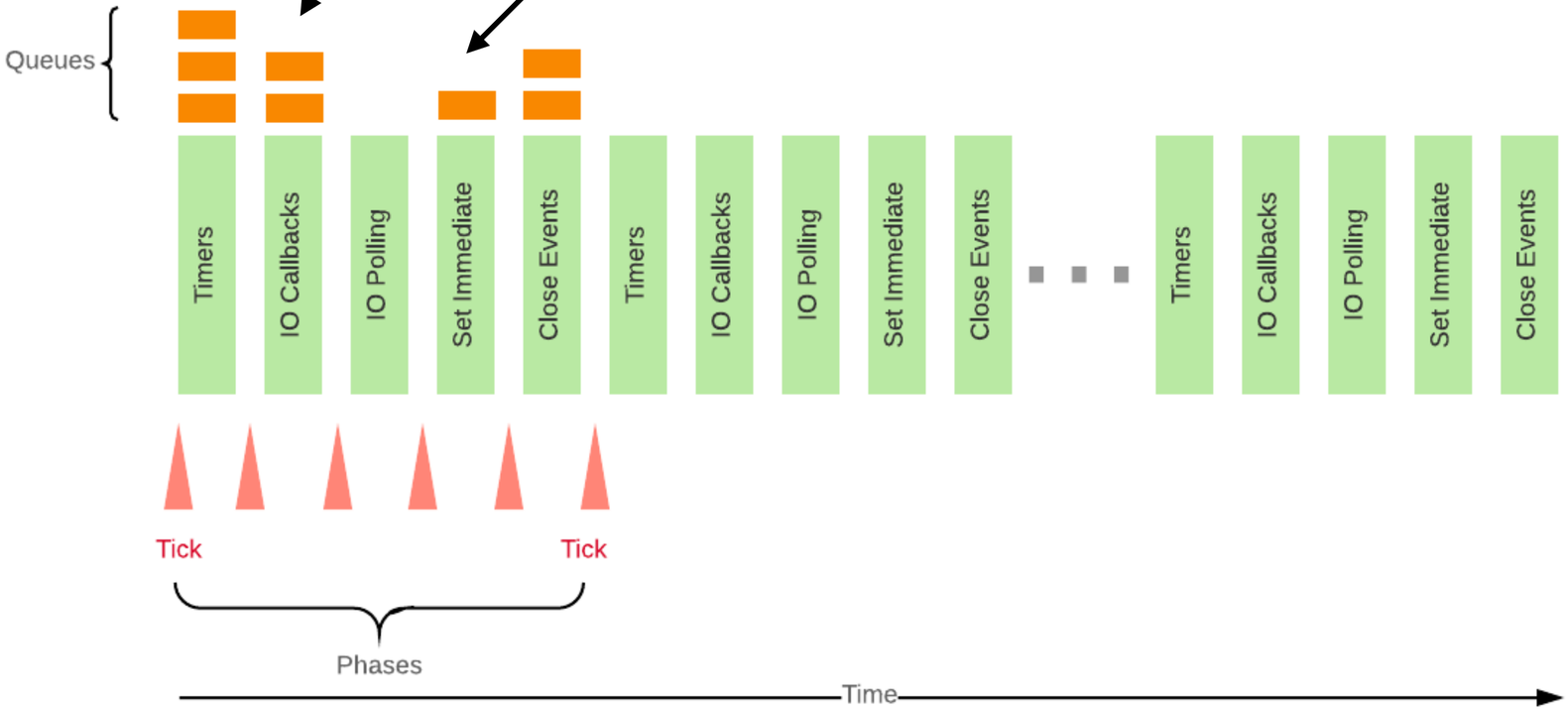
2 Iteration



Callback for readFile

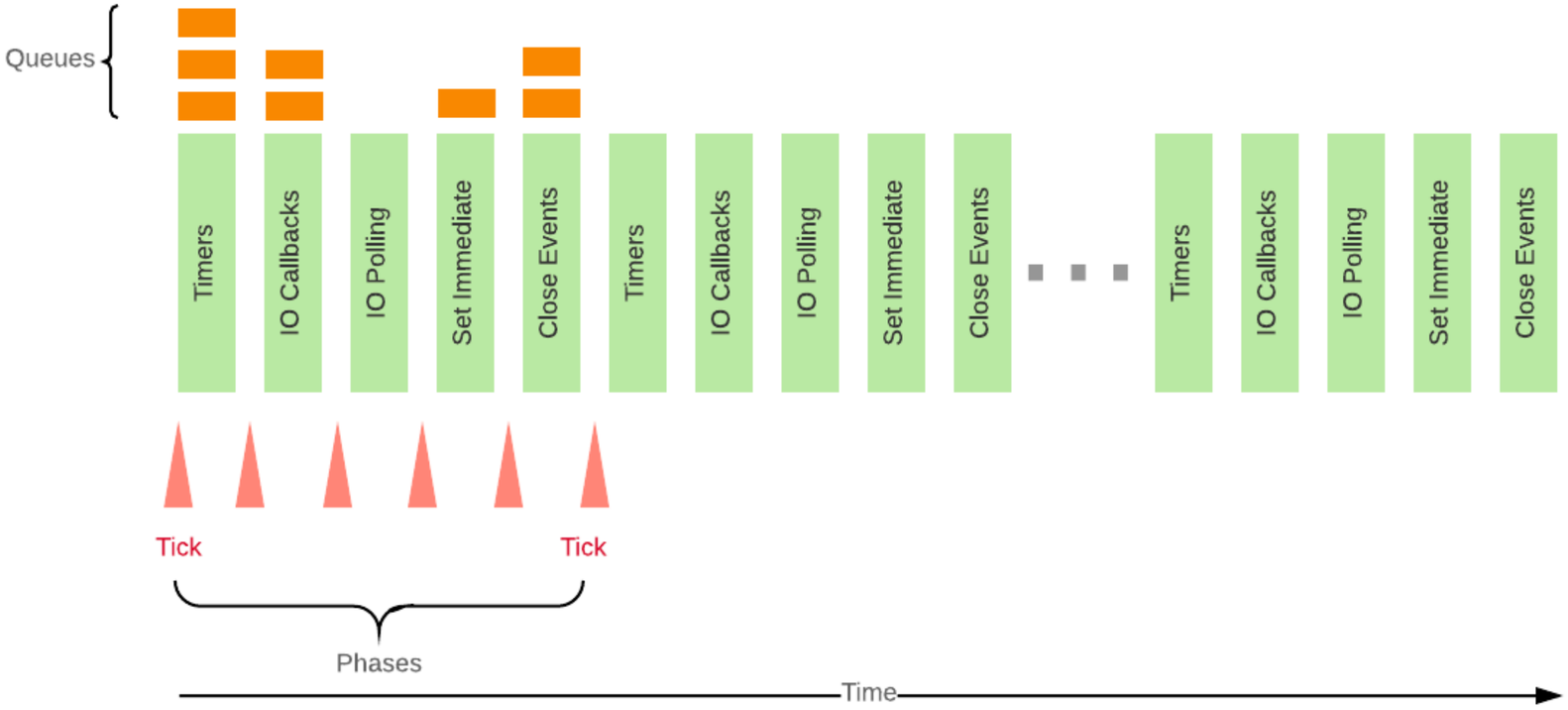
SetImmediate callback

3 Iteration



Callback for setTimeout

4 Iteration



Phases in Detail

https://nodejs.org/en/docs/guides/event-loop-timers-and-nexttick/

```
inline int Start(Isolate* isolate, IsolateData* isolate_data, int argc, const char* const* argv, int ex
    // ...
    {
        SealHandleScope seal(isolate);
        bool more;
        env.performance_state()->Mark(
            node::performance::NODE_PERFORMANCE_MILESTONE_LOOP_START);
        do {
            uv_run(env.event_loop(), UV_RUN_DEFAULT);

            v8_platform.DrainVMTasks(isolate);

            more = uv_loop_alive(env.event_loop());
            if (more)
                continue;

            RunBeforeExit(&env);

            more = uv_loop_alive(env.event_loop());
        } while (more == true);
        env.performance_state()->Mark(
            node::performance::NODE_PERFORMANCE_MILESTONE_LOOP_EXIT);
    }
    // ..
}
```

* https://github.com/nodejs/node/blob/master/src/node.cc

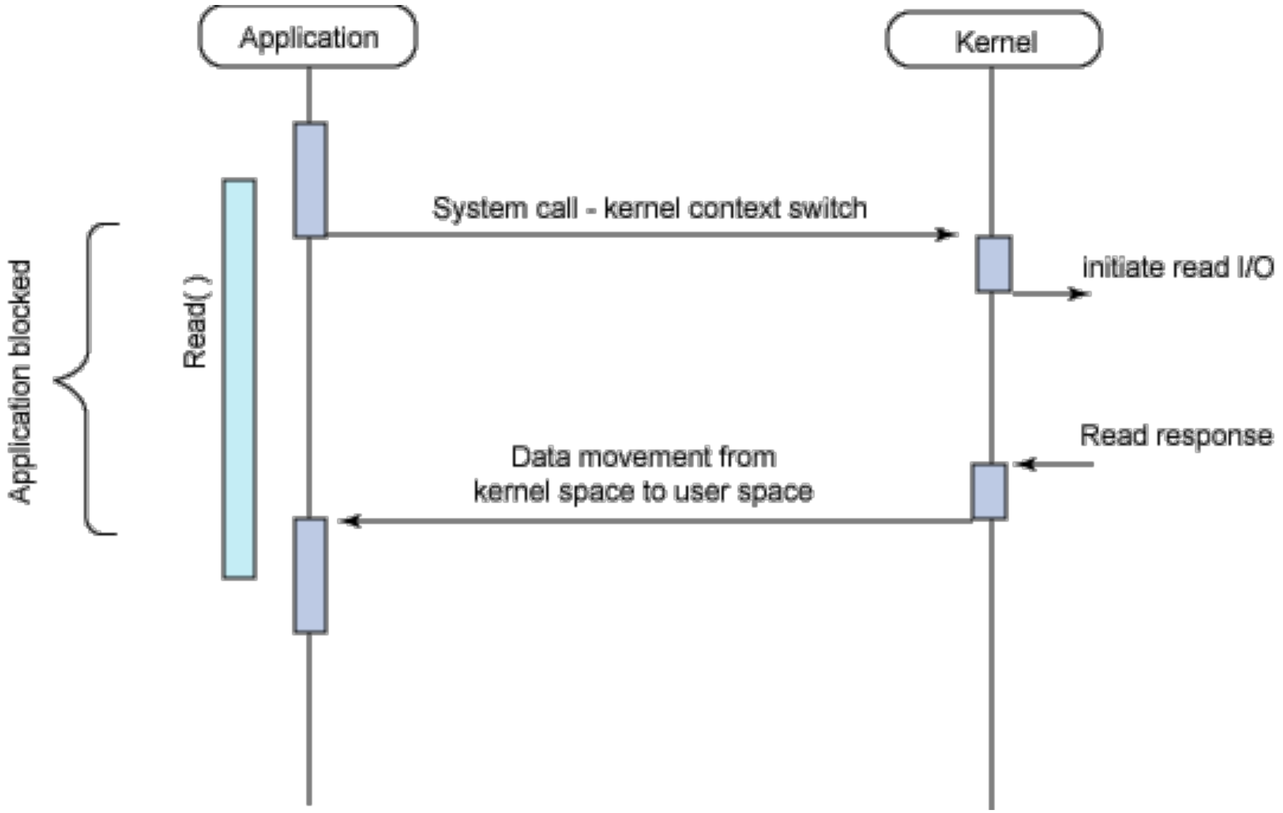
I/O

Take a lot of time

Blocking I/O

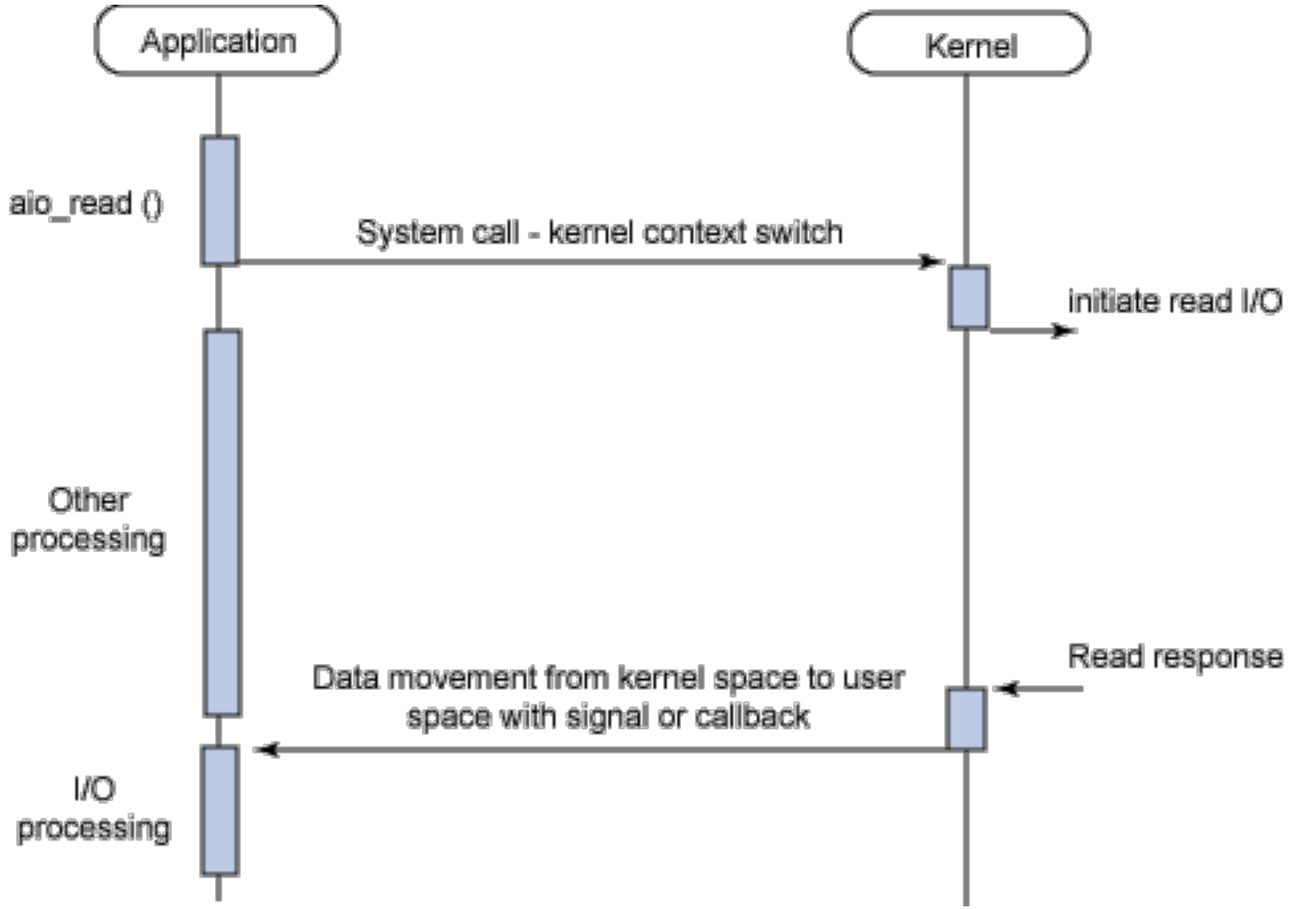
Non-Blocking I/O

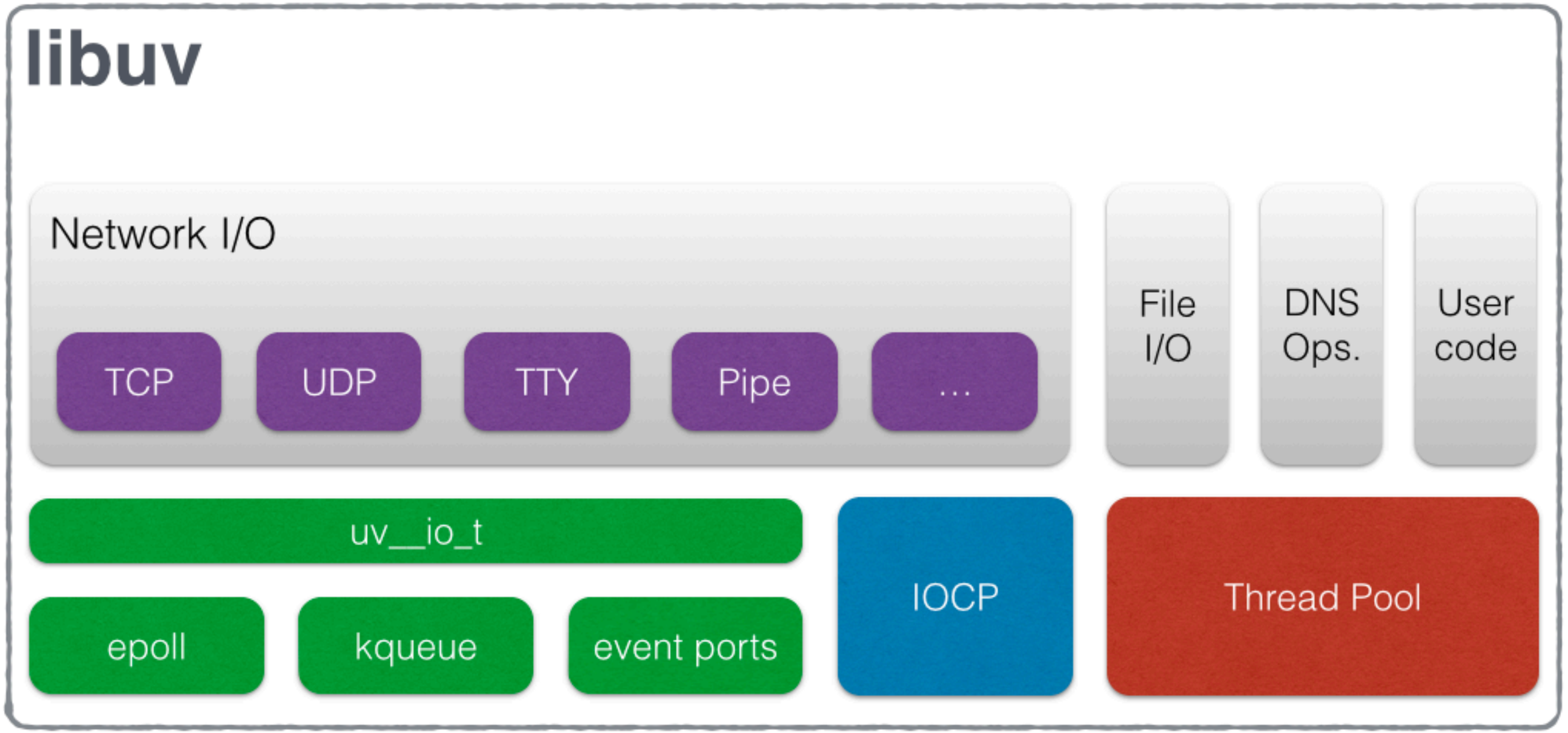
Blocking I/O



Райффайзен БАНК

Non-Blocking I/O




libuv
Network I/O
TCP
UDP
TTY
Pipe
...
File I/O
DNS Ops.
User code
uv__io_t
epoll
kqueue
event ports
IOCP
Thread Pool

Problems

Memory leaks

Crash by single error

~~Callback hall~~ async/await

Architecture

Event Loop, I/O

Modules

Node API

Node Bindings (socket, fs, http, …)

C/C++ Addons

V8

libuv

http parser

c-ares

openssl

zlib

Tools

npm

gyp

gtest

```c
struct node_module {
  int nm_version;
  unsigned int nm_flags;
  void* nm_dso_handle;
  const char* nm_filename; // Имя файла
  node::addon_register_func nm_register_func; // Функция модуля, для ObjectTemplate
  node::addon_context_register_func nm_context_register_func;
  const char* nm_modname; // Имя модуля (fs, os, http)
  void* nm_priv;
  struct node_module* nm_link;
};
```

* https://github.com/nodejs/node/blob/master/src/node.h

```cpp
extern "C" void node_module_register(void* m) {
  struct node_module* mp = reinterpret_cast<struct node_module*>(m);

  if (mp->nm_flags & NM_F_BUILTIN) {
    mp->nm_link = modlist_builtin;
    modlist_builtin = mp;
  } else if (mp->nm_flags & NM_F_INTERNAL) {
    mp->nm_link = modlist_internal;
    modlist_internal = mp;
  } else if (!node_is_initialized) {
    // "Linked" modules are included as part of the node project.
    // Like builtins they are registered *before* node::Init runs.
    mp->nm_flags = NM_F_LINKED;
    mp->nm_link = modlist_linked;
    modlist_linked = mp;
  } else {
    modpending = mp;
  }
}
```

* https://github.com/nodejs/node/blob/master/src/node.cc

Binding

```cpp
static void GetBinding(const FunctionCallbackInfo<Value>& args) {
  Environment* env = Environment::GetCurrent(args);

  CHECK(args[0]->IsString());

  // Имя модуля
  Local<String> module = args[0].As<String>();
  node::Utf8Value module_v(env->isolate(), module);

  // Ищем модуль в нашей структуры модулей
  node_module* mod = get_builtin_module(*module_v);
  // Определяем объект exports
  Local<Object> exports;
  // Инициализируем модуль
  if (mod != nullptr) {
    exports = InitModule(env, mod, module);
  } else if (!strcmp(*module_v, "constants")) {
    exports = Object::New(env->isolate());
    CHECK(exports->SetPrototype(env->context(),
                                Null(env->isolate())).FromJust());
    DefineConstants(env->isolate(), exports);
  } else if (!strcmp(*module_v, "natives")) {
    exports = Object::New(env->isolate());
    DefineJavaScript(env, exports);
  } else {
    return ThrowIfNoSuchModule(env, *module_v);
  }

  args.GetReturnValue().Set(exports);
}
```

* https://github.com/nodejs/node/blob/master/src/node.cc

```
// Привязываем функцию GetBinging, чтобы была доступна в JS
v8::Local<v8::Function> get_binding_fn =
      env->NewFunctionTemplate(GetBinding)->GetFunction(env->context())
          .ToLocalChecked();

// Добавляем «связывание» в глобальных объект process
process.binding = function binding(module) {
    module = String(module);
    let mod = bindingObj[module];
    if (typeof mod !== ‘object') {
    mod = bindingObj[module] = getBinding(module);
      moduleLoadList.push(`Binding ${module}`);
    }
    return mod;
};
```

* https://github.com/nodejs/node/blob/master/src/node.cc
* https://github.com/nodejs/node/blob/master/lib/internal/bootstrap/loaders.js

```
// Загружаем нативный модуль FS на C++
const binding = process.binding('fs');

// Функция fs.access
fs.access = function(path, mode, callback) {
    if (typeof mode === 'function') {
        callback = mode;
        mode = fs.F_OK;
    }

    path = getPathFromURL(path);
    validatePath(path);

    mode = mode | 0;
    var req = new FSReqWrap();
    req.oncomplete = makeCallback(callback);
    // Вызов нативного модуля
    binding.access(pathModule.toNamespacedPath(path), mode, req);
};
```

* https://github.com/nodejs/node/blob/master/src/node.cc
* https://github.com/nodejs/node/blob/master/lib/internal/bootstrap/loaders.js

require()

require

Native Modules

User Modules

NativeModule

```
// Set up NativeModule
function NativeModule(id) {
    this.filename = `${id}.js`;
    this.id = id;
    this.exports = {};
    this.loaded = false;
    this.loading = false;
}

NativeModule.require = function(id) {
    if (id === loaderId) {
        return loaderExports;
    }

    const cached = NativeModule.getCached(id);
    if (cached && (cached.loaded || cached.loading)) {
        return cached.exports;
    }

    if (!NativeModule.exists(id)) {
        const err = new Error(`No such built-in module: ${id}`);
        err.code = 'ERR_UNKNOWN_BUILTIN_MODULE';
        err.name = 'Error [ERR_UNKNOWN_BUILTIN_MODULE]';
        throw err;
    }

    moduleLoadList.push(`NativeModule ${id}`);
    const nativeModule = new NativeModule(id);

    nativeModule.cache();
    nativeModule.compile();

    return nativeModule.exports;
};
```

* https://github.com/nodejs/node/blob/master/lib/internal/bootstrap/loaders.js

```
NativeModule.prototype.compile = function() {
    let source = NativeModule.getSource(this.id);
    source = NativeModule.wrap(source);

    this.loading = true;

    try {
        const fn = runInThisContext(source, {
            filename: this.filename,
            lineOffset: 0,
            displayErrors: true
        });

        const requireFn = this.id.startsWith('internal/deps/') ?
            NativeModule.requireForDeps :
            NativeModule.require;

        fn(this.exports, requireFn, this, process);
        this.loaded = true;
    } finally {
        this.loading = false;
    }
};

// Minimal sandbox helper
const ContextifyScript = process.binding('contextify').ContextifyScript;
function runInThisContext(code, options) {
    const script = new ContextifyScript(code, options);
    return script.runInThisContext();
}
```

* https://github.com/nodejs/node/blob/master/lib/internal/bootstrap/loaders.js

```
NativeModule.wrap = function(script) {
    return NativeModule.wrapper[0] + script + NativeModule.wrapper[1];
};

NativeModule.wrapper = [
    '(function (exports, require, module, process) {',
    '\n});'
];
```

* https://github.com/nodejs/node/blob/master/lib/internal/bootstrap/loaders.js

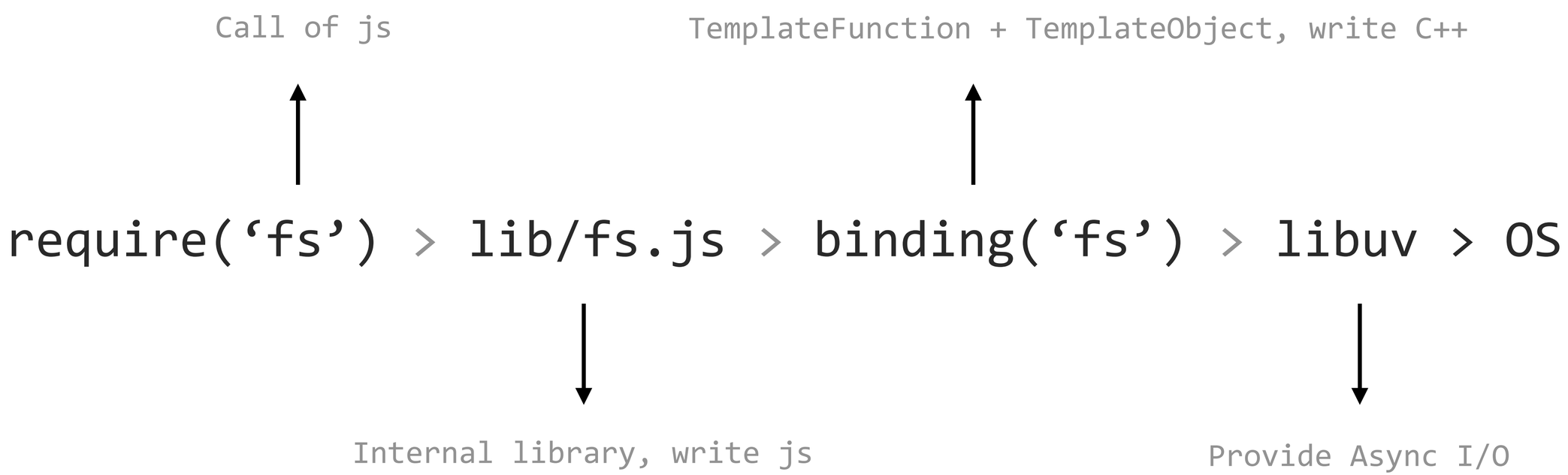
UserModule

```
// Native extension for .js
Module._extensions['.js'] = function(module, filename) {
    var content = fs.readFileSync(filename, 'utf8');
    module._compile(stripBOM(content), filename);
};
// Native extension for .js
Module._extensions['.js'] = function(module, filename) {
    var content = fs.readFileSync(filename, 'utf8');
    module._compile(stripBOM(content), filename);
};
// Native extension for .json
Module._extensions['.json'] = function(module, filename) {
    var content = fs.readFileSync(filename, 'utf8');
    try {
        module.exports = JSON.parse(stripBOM(content));
    } catch (err) {
        err.message = filename + ': ' + err.message;
        throw err;
    }
};
//Native extension for .node
Module._extensions['.node'] = function(module, filename) {
    return process.dlopen(module, path.toNamespacedPath(filename));
};
if (experimentalModules) {
    Module._extensions['.mjs'] = function(module, filename) {
        throw new ERR_REQUIRE_ESM(filename);
    };
}
```

* https://github.com/nodejs/node/blob/master/lib/internal/modules/cjs/loader.js

```
// Run the file contents in the correct scope or sandbox. Expose
// the correct helper variables (require, module, exports) to
// the file.
// Returns exception, if any.
Module.prototype._compile = function(content, filename) {
    content = stripShebang(content);
    // create wrapper function
    var wrapper = Module.wrap(content);

    var compiledWrapper = vm.runInThisContext(wrapper, {
        filename: filename,
        lineOffset: 0,
        displayErrors: true
    });

    var dirname = path.dirname(filename);
    var require = makeRequireFunction(this);
    var depth = requireDepth;
    if (depth === 0) stat.cache = new Map();
    var result = compiledWrapper.call(this.exports, this.exports, require, this, filename, dirname);

    if (depth === 0) stat.cache = null;
    return result;
};
```

* https://github.com/nodejs/node/blob/master/lib/internal/modules/cjs/loader.js


Call of js
TemplateFunction + TemplateObject, write C++
require(‘fs’) > lib/fs.js > binding(‘fs’) > libuv > OS
Internal library, write js
Provide Async I/O

Summary

Learned how works nodejs

Blocking, Non-Blocking I/O

Write own modules loader

How works require()

Thanks. QQ?

Alexander Lobashev, RaiffeisenBank

t.me/alobashev

Райффайзен БАНК