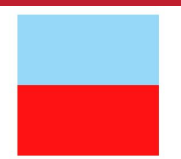
NodeJS
Эффективное программирование

Юра Богданов
технический директор и соучредитель Eventr



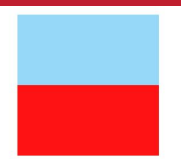
NodeJS

Цель проекта:

«Предоставить естественную неблокирующую, событийно-ориентированную инфраструктуру для написания программ с высокой конкурентностью»

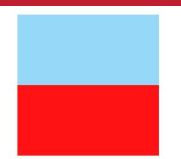
(c) Ryan Dahl

«to provide a purely evented, non-blocking infrastructure to script highly concurrent programs»



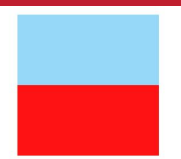
NodeJS

- **NodeJS** – серверная JavaScript платформа
  - Использует Google V8 (Chromium: Google Chrome, Chrome OS, etc.)
  - Превращает V8 в мощную машину для серверных приложений
  - Сливается в гармонии с философией JavaScript
  - Молодой, но живой
- **Event loop** - неблокирующий ввод/вывод
  - Все выполняется параллельно, кроме вашего кода



Для чего подходит NodeJS

- **Много I/O** + большая конкурентность
  - RIA — «богатые» приложения
  - API
  - Proxy
- **Realtime**
  - Чаты
  - Онлайн игры
  - Трансляции
  - Publish/Subscribe

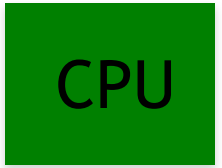


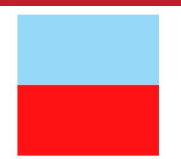
Event loop

- **Это цикл** (libev)
- **Это один процесс, один поток**
- **Выполняет одну задачу на один момент времени**
- **Ожидает события параллельно** (libeio, pooled threads)
- **В каждой итерации последовательно запускает функции-колбэки из трех разных очередей:**
  1. **nextTick функции**
  2. **Таймеры** (setTimeout, setInterval)
  3. **Сигналы ввода/вывода** (libeio)
- **Завершает работу, если все очереди пусты**

Время

- **CPU** - процессорное время
  - Интерпретация кода
  - Бизнес-логика приложения, алгоритмы
  - Рендеринг шаблонов
- **I/O** - время ввода/вывода
  - Запросы в базу данных (network)
  - Чтение файлов
  - Чтение кэша



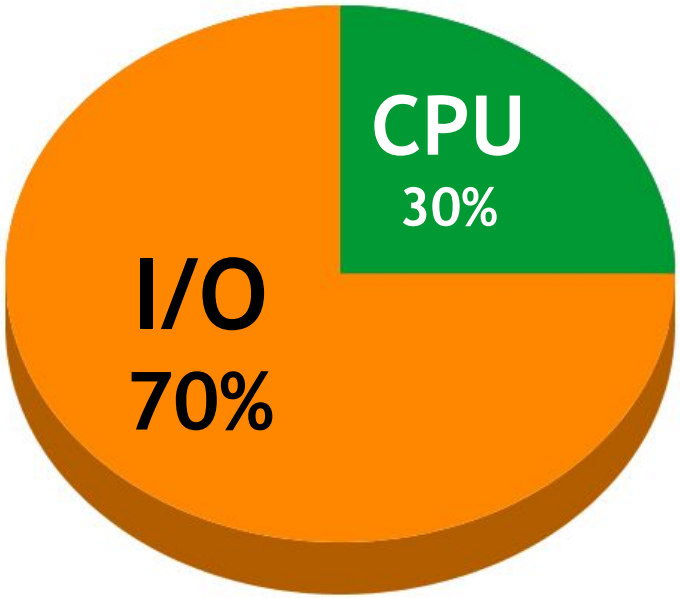


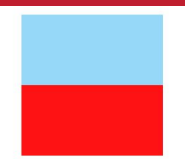

```
<?php
$user = $db->query('SELECT * FROM users WHERE id=1'); // I/O - 70ms
$html = renderUser($user); // CPU - 30ms
```

Упрощенный пример сценария веб-приложения

На самом деле, у нас много I/O и много логики

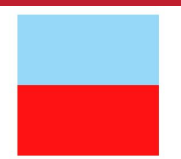

```php
<?php
$user = $db->query('SELECT * FROM users WHERE id=1'); // I/O - 70ms
$html = renderUser($user); // CPU - 30ms
```






```
<?php
$user = $db->query('SELECT * FROM users WHERE id=1'); // I/O - 70ms
$html = renderUser($user); // CPU - 30ms
```

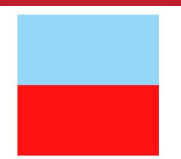
А что если...



Blocking I/O

Event Loop

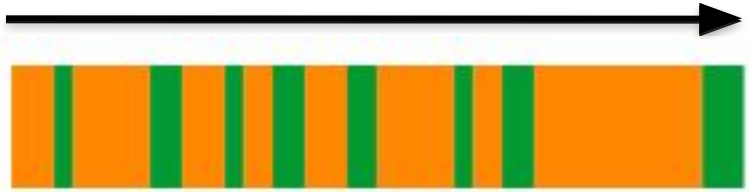
А что если во время ожидания I/O заниматься другими полезными делами?

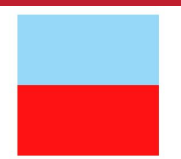


Event loop

Примерно так выглядит более реальный запрос:

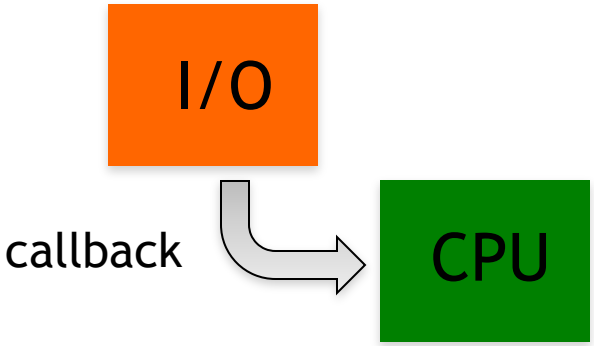
CPU+I/O

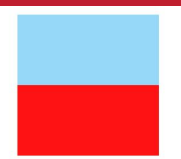




Event loop

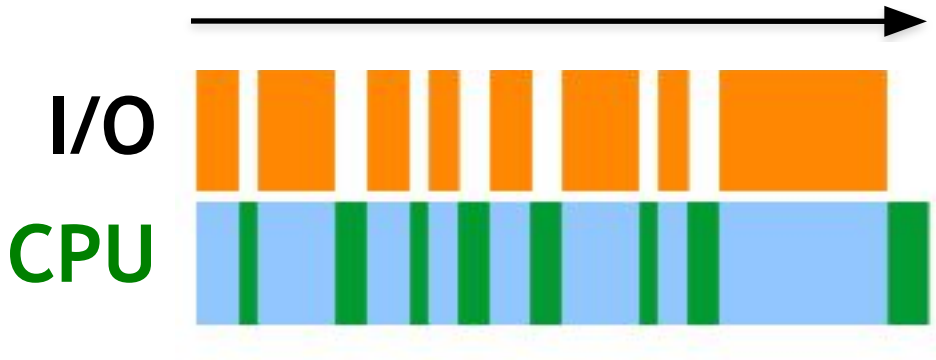
Примерно так выглядит более реальный запрос:

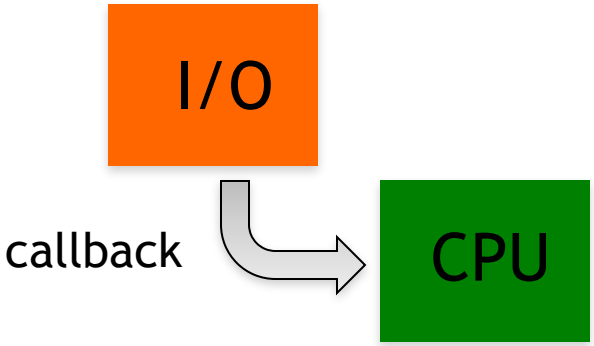




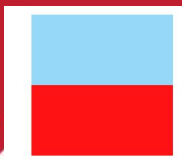
Event loop

Примерно так выглядит более реальный запрос:





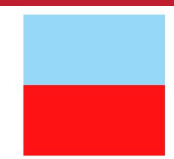
Свободно для других задач



Event Loop

```
mysql.query('SELECT count(*) FROM users', function(err, count) {
    console.log('There are %d users in db', count);
})

console.log('Hello, ');
```

Отправка запроса в БД

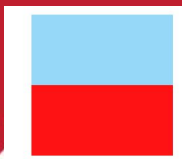


Event Loop

```
mysql.query('SELECT count(*) FROM users', function(err, count) {
    console.log('There are %d users in db', count);
})

console.log('Hello, ');
```

Ожидание ответа БД...

Hello,

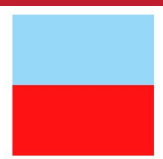


Event Loop

```
mysql.query('SELECT count(*) FROM users', function(err, count) {
    console.log('There are %d users in db', count);
})

console.log('Hello, ');
```

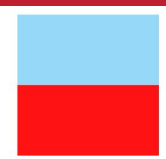
Пришел ответ из БД

Hello,
There are 10231512 users in db




```
<?php
$user = $db->query('SELECT * FROM users WHERE id=1'); // I/O - 70ms
$html = renderUser($user); // CPU - 30ms
```

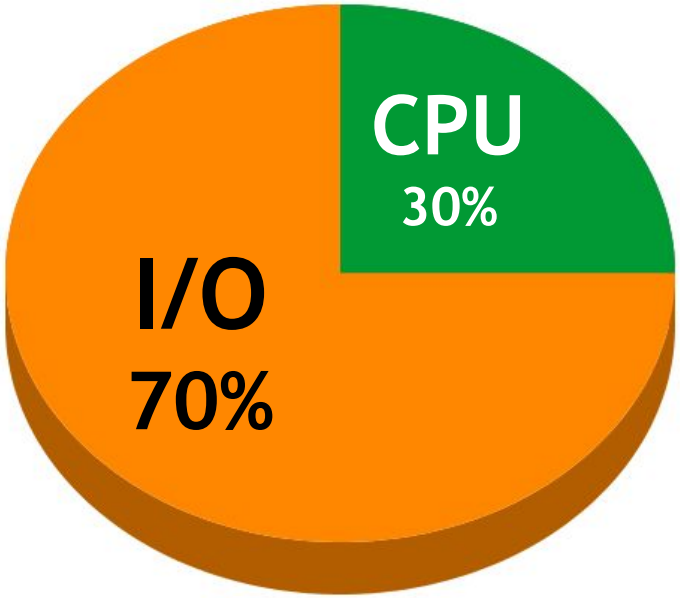
```
db.query('SELECT * FROM users WHERE id=1', function(err, user) { // I/O 70ms
    var html = renderUser(user); // CPU - 30ms
})
```

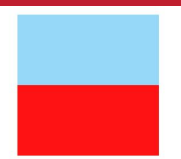


Akshell

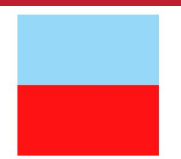
```php
<?php
$user = $db->query('SELECT * FROM users WHERE id=1'); // I/O - 70ms
$html = renderUser($user); // CPU - 30ms
```

```
db.query('SELECT * FROM users WHERE id=1', function(err, user) { // I/O 70ms
    var html = renderUser(user); // CPU - 30ms
})
```

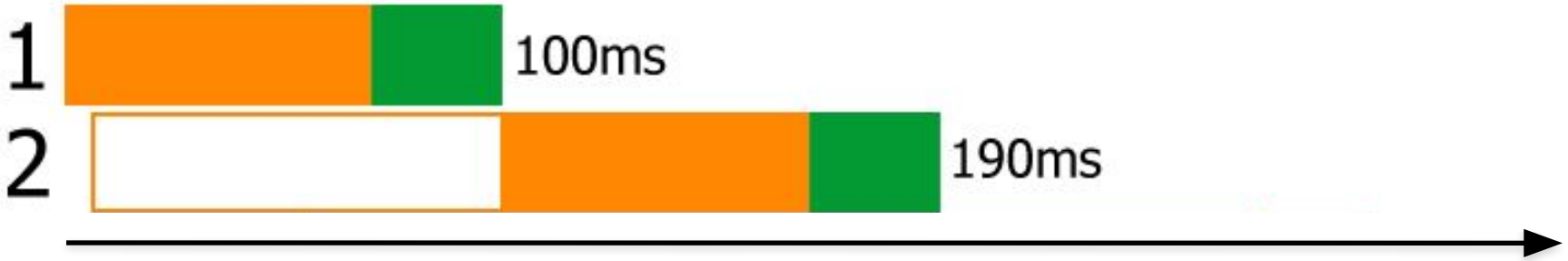




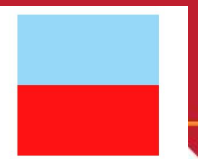
Blocking I/O, 1 процесс



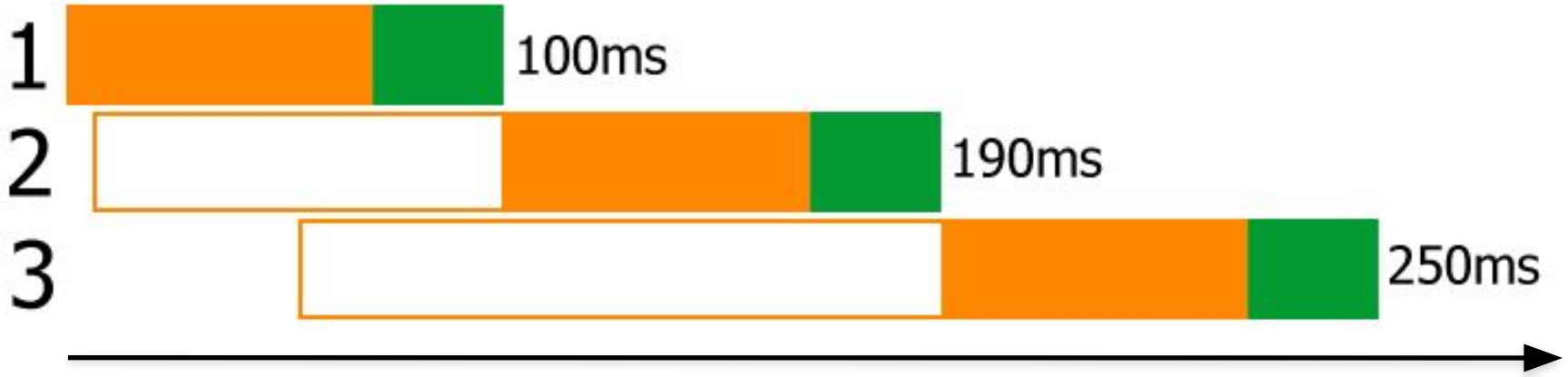
Blocking I/O, 1 процесс



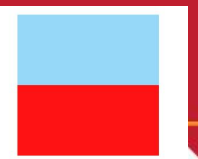
Второй запрос, после 10ms ожидает выполнения первого



Blocking I/O, 1 процесс



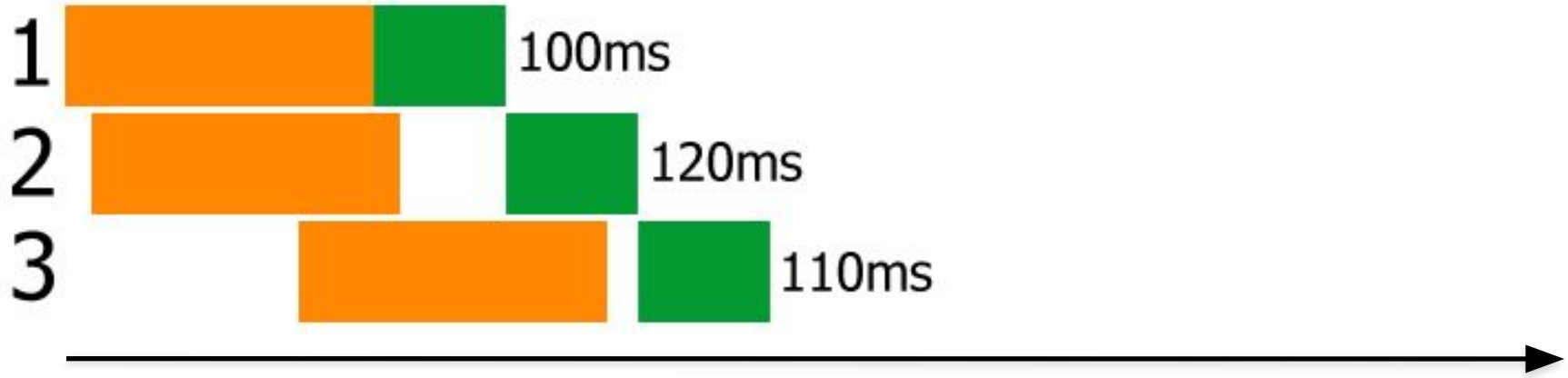
Третий запрос, после 50ms
ожидает выполнения первого и второго

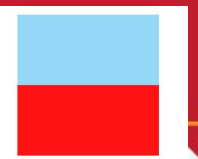


Akshell

Blocking I/O, 1 процесс

Event loop, 1 процесс

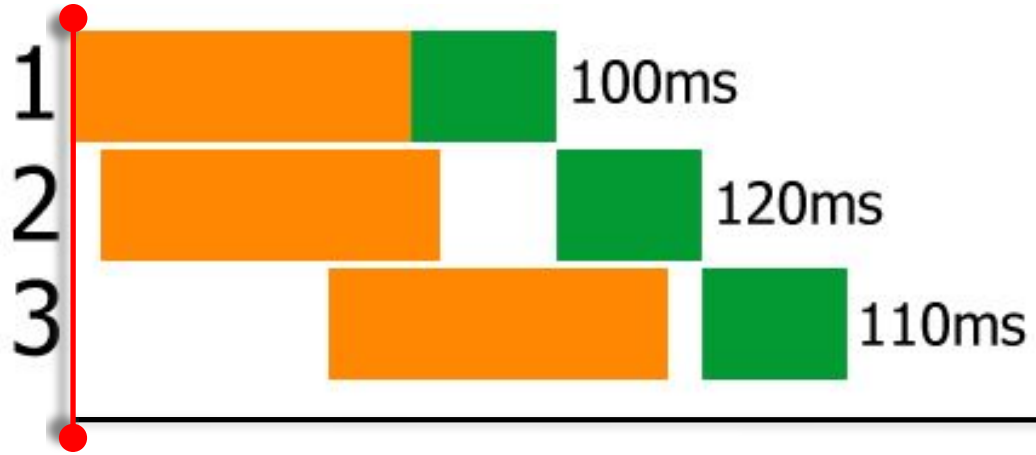




Akshell

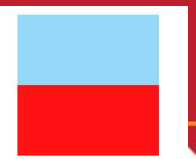
Blocking I/O, 1 процесс

Event loop, 1 процесс



Время: 0ms

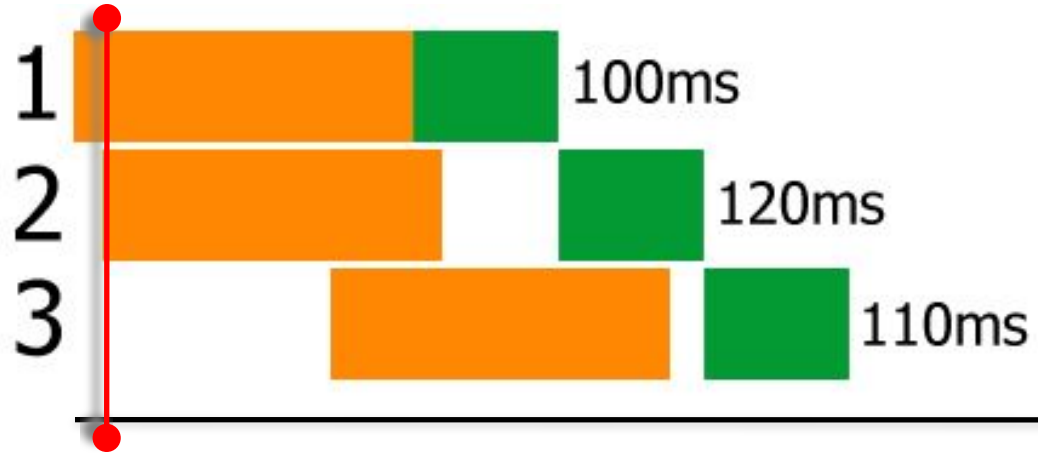
Пришел первый запрос,
Запрашиваем I/O,
освобождаемся,
Ждем других запросов



Akshell

Blocking I/O, 1 процесс

Event loop, 1 процесс



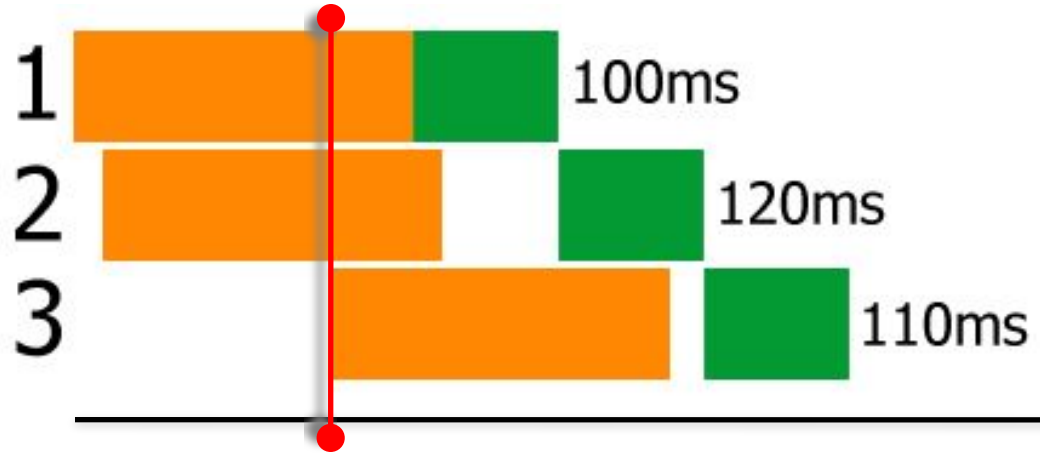
Время: 10ms

Пришел второй запрос,
Запрашиваем I/O,
освобождаемся,
Ждем других запросов

Akshell

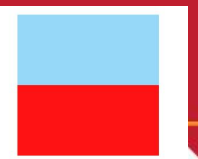
Blocking I/O, 1 процесс

Event loop, 1 процесс



Время: 50ms

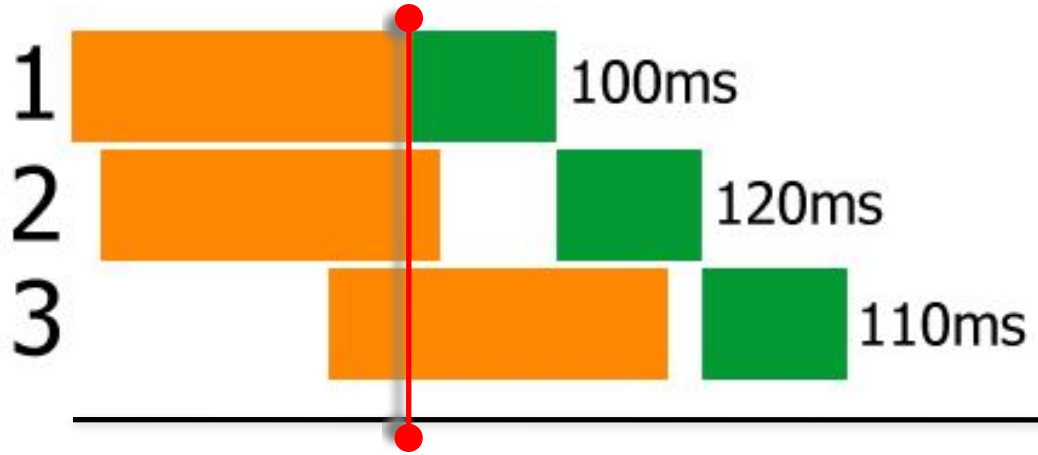
Пришел третий запрос,
Запрашиваем I/O,
освобождаемся,
Ждем других запросов



Akshell

Blocking I/O, 1 процесс

Event loop, 1 процесс



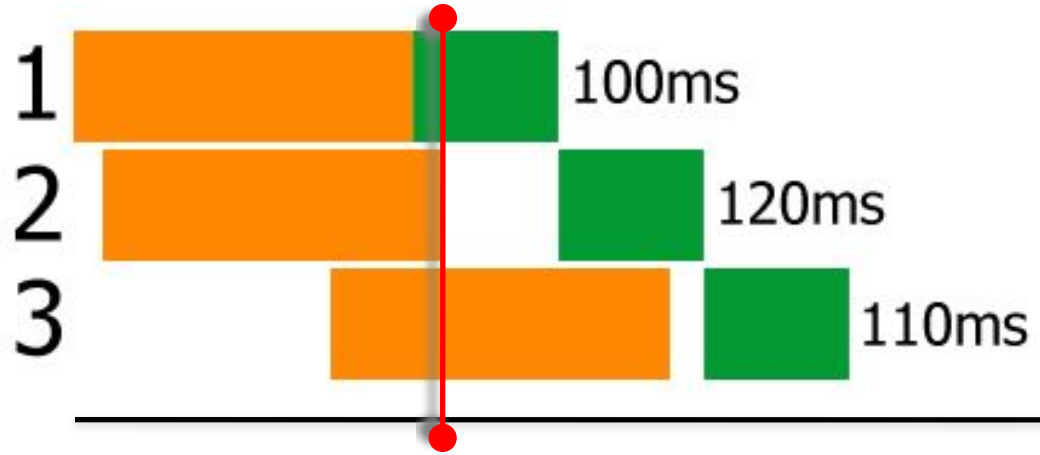
Время: 70ms

Пришел ответ на I/O первого запроса, запускаем callback1

Akshell

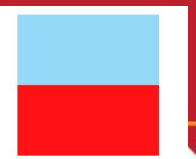
Blocking I/O, 1 процесс

Event loop, 1 процесс



Время: 80ms

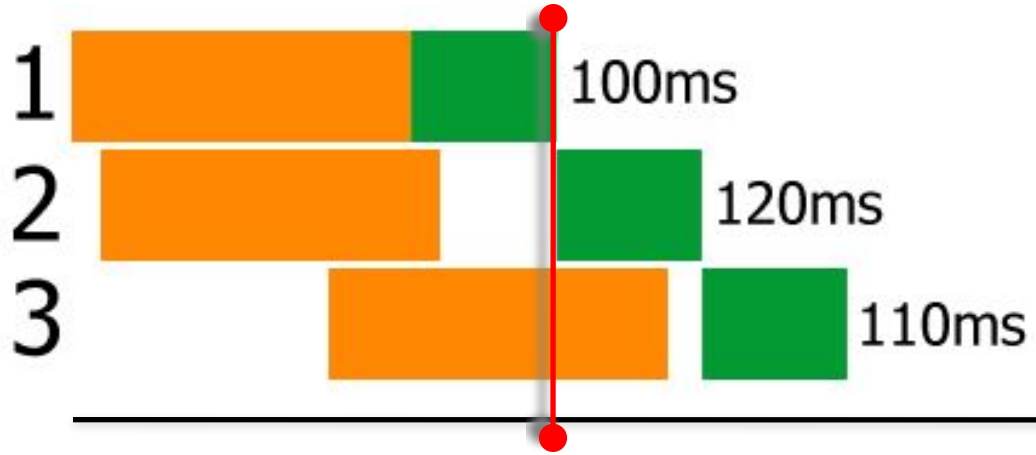
Пришел ответ на I/O второго запроса, callback2 ожидает своей очереди



Akshell

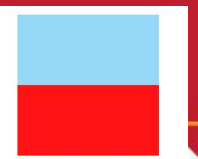
Blocking I/O, 1 процесс

Event loop, 1 процесс



Время: 100ms

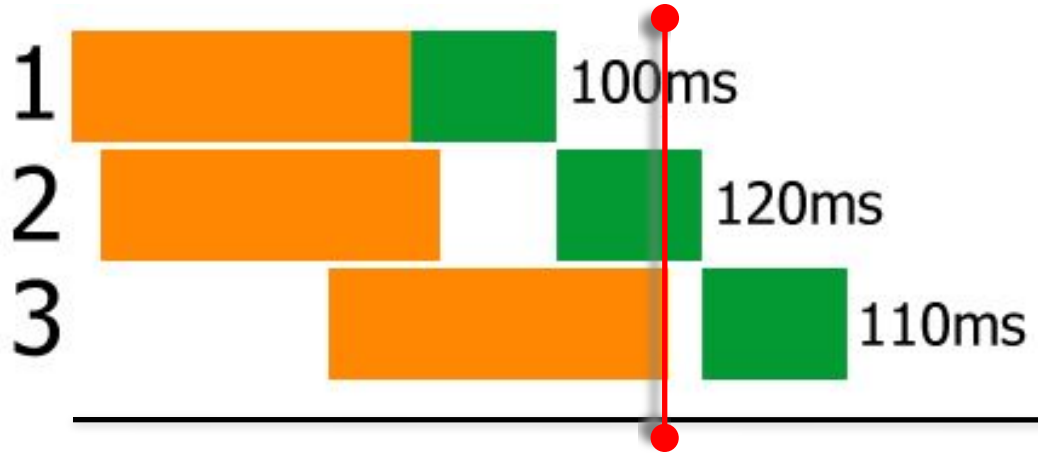
callback1 завершился,
запускаем callback2



Akshell

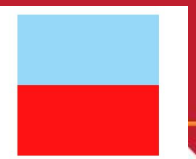
Blocking I/O, 1 процесс

Event loop, 1 процесс



Время: 120ms

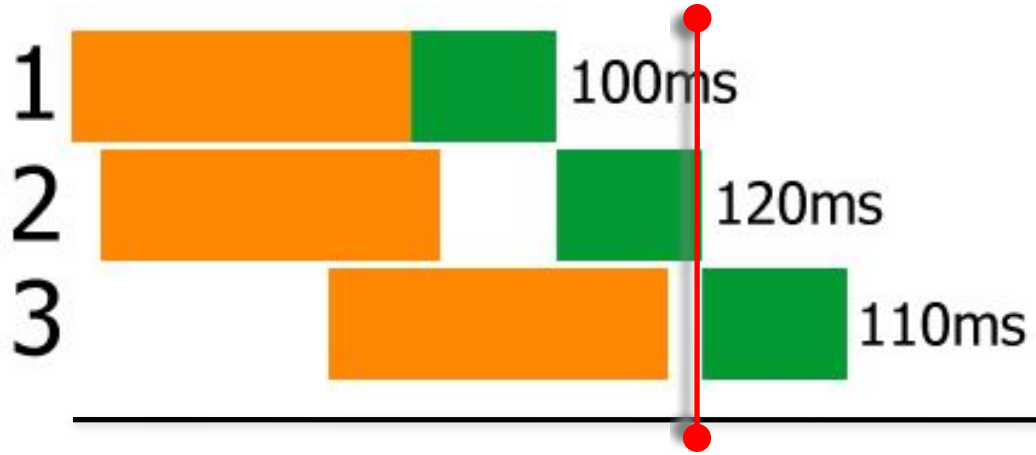
Пришел ответ на I/O третьего запроса, callback3 ожидает своей очереди



Akshell

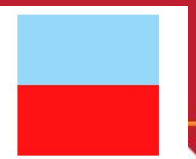
Blocking I/O, 1 процесс

Event loop, 1 процесс



Время: 130ms

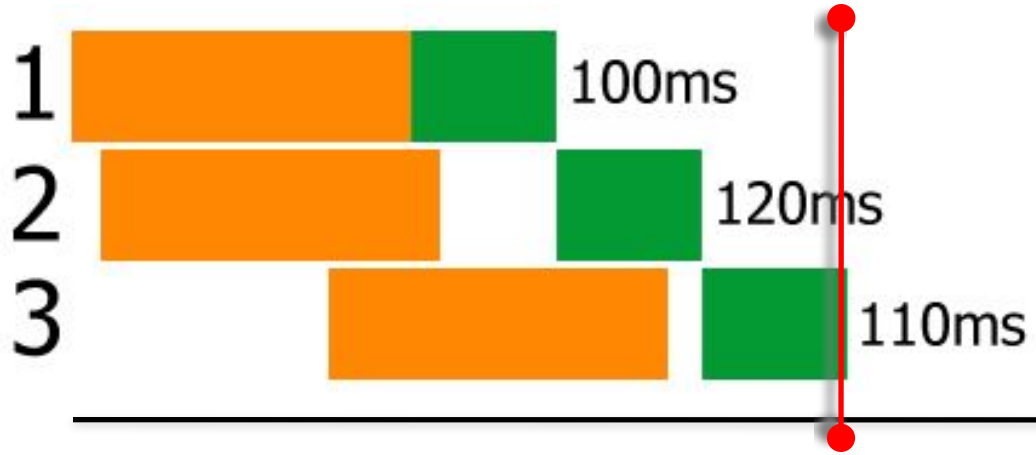
callback2 завершился,
запускаем callback3



Akshell

Blocking I/O, 1 процесс

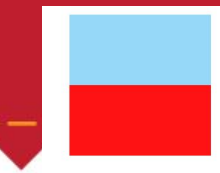
Event loop, 1 процесс



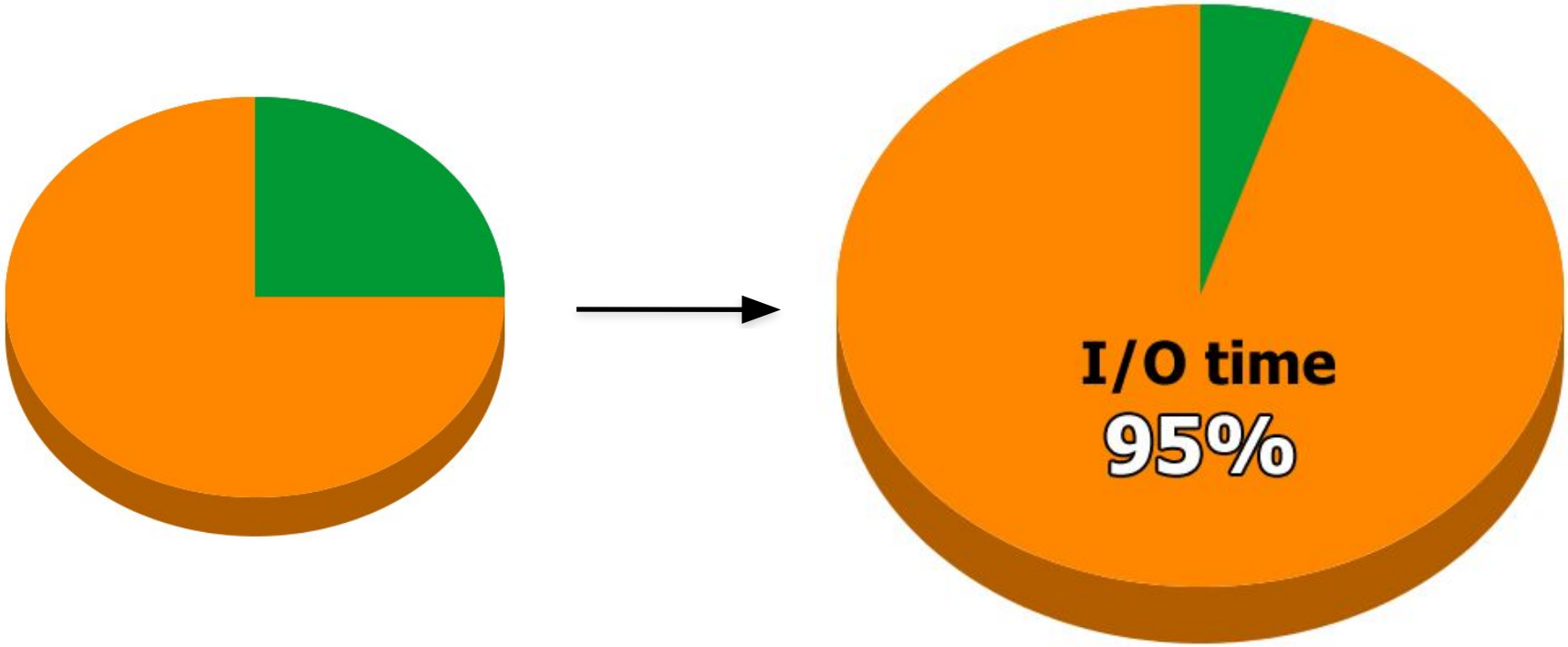
Время: 160ms

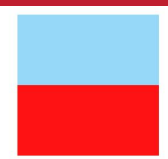
callback3 завершился

Ждем других запросов



Время CPU vs I/O
RIA трэнд





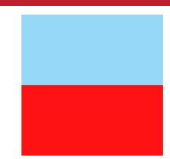
Приложение

- **my_app.js**
- library /
  - my_module.js
- node_modules /
  - express
  - sync
  - narrow

Приложение

- **my_app.js**
- library /
  - my_module.js
- node_modules /
  - express
  - sync
  - narrow

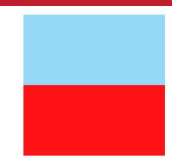
```
require.paths.unshift('./library')

var MyModule = require('my_module')
var Sync = require('sync')
```



Приложение

- my_app.js
- library /
  - **my_module.js**
- node_modules /
  - express
  - sync
  - narrow

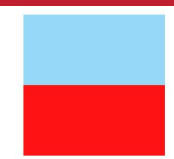
```
var MyModule = function() {
    // …
}
module.exports = MyModule
```



Приложение

- **my_app.js**
- library /
  - my_module.js
- node_modules /
  - express
  - sync
  - narrow

```
require.paths.unshift('./library')

var MyModule = require('my_module')
var Sync = require('sync')

+ console.log('my module: ', MyModule);
```

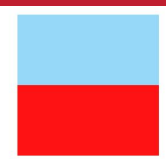


Приложение

- **my_app.js**
- library /
  - my_module.js
- node_modules /
  - express
  - sync
  - narrow

$ node my_app.js

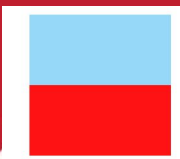
my module: function (){}



Приложение

- my_app.js
- library /
  - my_module.js
- node_modules /
  - express
  - sync
  - narrow
  - mongoose

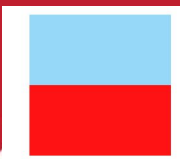
$ npm install mongoose



HTTP

```
var http = require('http');
http.createServer(function(req, res){
    res.writeHead(200, { 'Content-Type' : 'text/plain' });
    res.end('Hello, World\n');
}).listen(3080)
console.log('Server running at http://127.0.0.1:3080/');
```

Подключаем HTTP модуль



HTTP

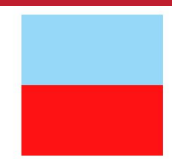
```
var http = require('http');
http.createServer(function(req, res){
    res.writeHead(200, { 'Content-Type' : 'text/plain' });
    res.end('Hello, World\n');
}).listen(3080)
console.log('Server running at http://127.0.0.1:3080/');
```

Создаем HTTP сервер

HTTP

```
var http = require('http');
http.createServer(function(req, res){
    res.writeHead(200, { 'Content-Type' : 'text/plain' });
    res.end('Hello, World\n');
}).listen(3080)
console.log('Server running at http://127.0.0.1:3080/');
```

«вешаем» сервер на 3080 порт



HTTP

```
var http = require('http');
http.createServer(function(req, res){
    res.writeHead(200, { 'Content-Type' : 'text/plain' });
    res.end('Hello, World\n');
}).listen(3080)
console.log('Server running at http://127.0.0.1:3080/');
```

Сервер создан,
выводим сообщение

HTTP

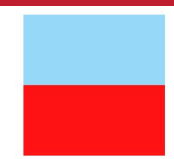
```
var http = require('http');
http.createServer(function(req, res){
    res.writeHead(200, { 'Content-Type' : 'text/plain' });
    res.end('Hello, World\n');
}).listen(3080)
console.log('Server running at http://127.0.0.1:3080/');
```

Функция будет вызвана индивидуально для каждого запроса

HTTP

```
var http = require('http');
http.createServer(function(req, res){
    res.writeHead(200, { 'Content-Type' : 'text/plain' });
    res.end('Hello, World\n');
}).listen(3080)
console.log('Server running at http://127.0.0.1:3080/');
```

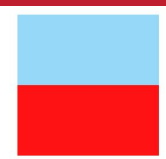
Отправляем HTTP заголовок



HTTP

```
var http = require('http');
http.createServer(function(req, res){
    res.writeHead(200, { 'Content-Type' : 'text/plain' });
    res.end('Hello, World\n');
}).listen(3080)
console.log('Server running at http://127.0.0.1:3080/');
```

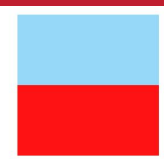
Отправляем HTTP тело и закрываем сокет



HTTP

```
var http = require('http');
http.createServer(function(req, res){
    res.writeHead(200, { 'Content-Type' : 'text/plain' });
    res.end('Hello, World\n');
}).listen(3080)
console.log('Server running at http://127.0.0.1:3080/');
```

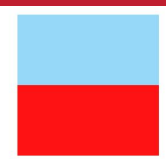
$ node my_app.js	



HTTP

```
var http = require('http');
http.createServer(function(req, res){
    res.writeHead(200, { 'Content-Type' : 'text/plain' });
    res.end('Hello, World\n');
}).listen(3080)
console.log('Server running at http://127.0.0.1:3080/');
```

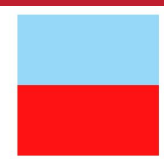
```
$ node my_app.js
Server running at http://127.0.0.1:3080/
```



HTTP

```
var http = require('http');
http.createServer(function(req, res){
    res.writeHead(200, { 'Content-Type' : 'text/plain' });
    res.end('Hello, World\n');
}).listen(3080)
console.log('Server running at http://127.0.0.1:3080/');
```

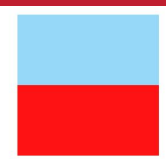
$ node my_app.js Server running at http://127.0.0.1:3080/	$ curl http://127.0.0.1:3080/



HTTP

```
var http = require('http');
http.createServer(function(req, res){
    res.writeHead(200, { 'Content-Type' : 'text/plain' });
    res.end('Hello, World\n');
}).listen(3080)
console.log('Server running at http://127.0.0.1:3080/');
```

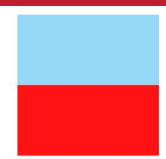
$ node my_app.js Server running at http://127.0.0.1:3080/	**$ curl http://127.0.0.1:3080/** Hello, World **$**



HTTP

```
    var http = require('http');
    http.createServer(function(req, res){
+       setTimeout(function(){
+           res.end('World!\n');
+       }, 1000);
        res.writeHead(200, { 'Content-Type' : 'text/plain' });
+       res.write('Hello,\n');
    }).listen(3080)
    console.log('Server running at http://127.0.0.1:3080/');
```

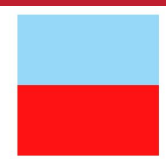
Hello сразу, World - через 1 сек



HTTP

```
var http = require('http');
http.createServer(function(req, res){
    setTimeout(function(){
        res.end('World!\n');
    }, 1000);
    res.writeHead(200, { 'Content-Type' : 'text/plain' });
    res.write('Hello,\n');
}).listen(3080)
console.log('Server running at http://127.0.0.1:3080/');
```

$ node my_app.js Server running at http://127.0.0.1:3080/	$ curl http://127.0.0.1:3080/ Hello,



HTTP

```
var http = require('http');
http.createServer(function(req, res){
    setTimeout(function(){
        res.end('World!\n');
    }, 1000);
    res.writeHead(200, { 'Content-Type' : 'text/plain' });
    res.write('Hello,\n');
}).listen(3080)
console.log('Server running at http://127.0.0.1:3080/');
```

$ node my_app.js Server running at http://127.0.0.1:3080/	$ curl http://127.0.0.1:3080/ Hello, World! $

Через секунду

HTTP

```
var http = require('http');
var i = 0;
http.createServer(function(req, res){
    res.writeHead(200, { 'Content-Type' : 'text/plain' });
    res.end('i = ' + i + '\n');
    i++;
}).listen(3080)
console.log('Server running at http://127.0.0.1:3080/');
```

Итератор, общий для всех запросов - javascript замыкание (closure)

HTTP

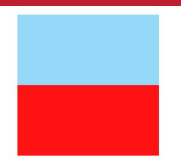
```
var http = require('http');
var i = 0;
http.createServer(function(req, res){
    res.writeHead(200, { 'Content-Type' : 'text/plain' });
    res.end('i = ' + i + '\n');
    i++;
}).listen(3080)
console.log('Server running at http://127.0.0.1:3080/');
```

$ node my_app.js Server running at http://127.0.0.1:3080/	$ curl http://127.0.0.1:3080/ i = 0 $

HTTP

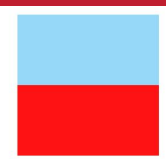
```
var http = require('http');
var i = 0;
http.createServer(function(req, res){
    res.writeHead(200, { 'Content-Type' : 'text/plain' });
    res.end('i = ' + i + '\n');
    i++;
}).listen(3080)
console.log('Server running at http://127.0.0.1:3080/');
```

$ node my_app.js Server running at http://127.0.0.1:3080/	$ curl http://127.0.0.1:3080/ i = 0 $ curl http://127.0.0.1:3080/ i = 1 $



Callback-driven парадигма

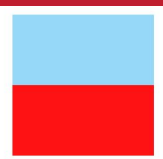
- Ломает мозг
- Синтаксический шум
- Сложно выполнить ряд действий в определенной последовательности
- Необходимость вручную «пробрасывать» ошибки
- Бесконечная индентация - aka «спагетти код»



Callback-driven парадигма

Неблокирующая функция принимает callback последним аргументом

```
function asyncFunction(arg1, arg2, argN, callback) {

}
```



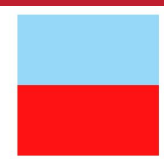
Callback-driven парадигма

Неблокирующая функция принимает callback последним аргументом

```
function asyncFunction(arg1, arg2, argN, callback) {

}
```

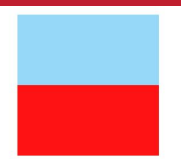
Callback принимает ошибку первым аргументом, остальные - результат

```
function callback(err, result1, result2, resultN) {

}
```



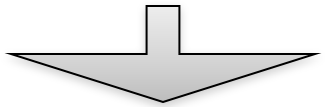
Callback-driven парадигма

```
function sum(a, b) {
    if (a > b) {
      throw new Error('a cannot be greater than b');
    }
    return a + b;
}
```

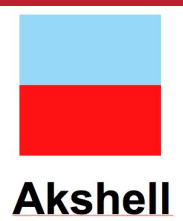


Callback-driven парадигма

```
function sum(a, b) {
    if (a > b) {
      throw new Error('a cannot be greater than b');
    }
    return a + b;
}
```



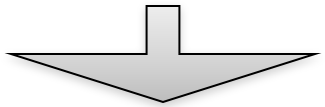
```
+ function asyncSum(a, b, callback) {
    if (a > b) {
+     return callback(new Error('a cannot be greater than b'));
    }
+   callback(null, a + b);
}
```



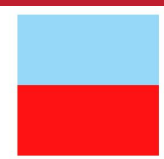
Callback-driven парадигма

```
function sum(a, b) {
    if (a > b) {
      throw new Error('a cannot be greater than b');
    }
    return a + b;
}
```

~~throw~~!

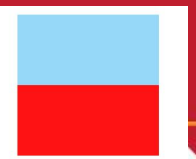


```
+ function asyncSum(a, b, callback) {
    if (a > b) {
+     return callback(new Error('a cannot be greater than b'));
    }
+   callback(null, a + b);
}
```



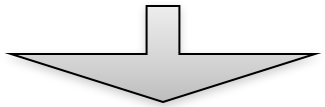
Callback-driven парадигма

```
try {
    var result = sum(2, 3);
    console.log('result = %d', result);
} catch (err) {
    console.error(err);
}
```

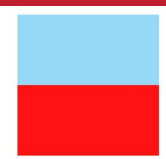


Callback-driven парадигма

```
try {
    var result = sum(2, 3);
    console.log('result = %d', result);
} catch (err) {
    console.error(err);
}
```

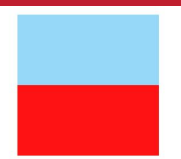


```
+ asyncSum(2, 3, function(err, result){
+     if (err) return console.error(err);
      console.log('result = %d', result);
+ })
```



Callback-driven парадигма

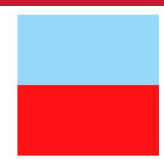
```
function getUser(id, callback) {

}
```



Callback-driven парадигма

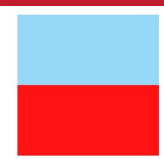
```
function getUser(id, callback) {

}
```

```
getUser(1234, function(err, user) {
    if (err) return console.error(err);
    console.log('user: ', user);
})
```



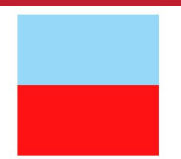
Callback-driven парадигма

```
    function getUser(id, callback) {
+       readConfig('config.json', function(err, config){
+           if (err) return callback(err);

+       })
    }
```



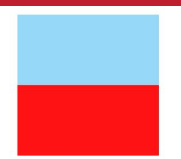
Callback-driven парадигма

```
    function getUser(id, callback) {
        readConfig('config.json', function(err, config){
            if (err) return callback(err);
+           dbConnect(config.host, function(err, db){
+               if (err) return callback(err);

+           })
        })
    }
```



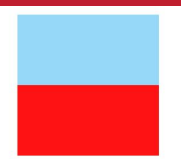
Callback-driven парадигма

```
function getUser(id, callback) {
    readConfig('config.json', function(err, config){
        if (err) return callback(err);
        dbConnect(config.host, function(err, db){
            if (err) return callback(err);
+           db.getUser(id, function(err, user){
+               if (err) return callback(err);
+               callback(null, user);
+           })
        })
    })
}
```



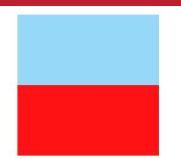
Callback-driven парадигма

```
    function getUser(id, callback) {
        readConfig('config.json', function(err, config){
            if (err) return callback(err);
+           afterReadConfig(id, config, callback);
        })
    }

+   function afterReadConfig(id, config, callback) {
+       dbConnect(config.host, function(err, db){
+           if (err) return callback(err);
+           db.getUser(id, function(err, user){
+               ...
+       })
    }
```



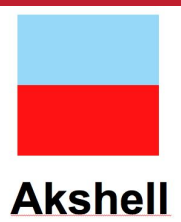
Callback-driven парадигма

```
function getUser(id, callback) {
    readConfig('config.json', function(err, config){
        if (err) return callback(err);
        afterReadConfig(id, config, callback);
    })
}

function afterReadConfig(id, config, callback) {
    dbConnect(config.host, function(err, db){
        if (err) return callback(err);
+       afterDbConnect(id, db, callback);
    })
}
+ function afterDbConnect(id, db, callback) …
```



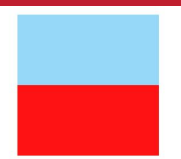
Callback-driven парадигма

```
Error: User not found
    at afterDbConnect (/path/to/script.js:24:14)
    at /path/to/script.js:20:9
    at dbConnect (/path/to/script.js:7:5)
    at afterReadConfig (/path/to/script.js:18:5)
    at /path/to/script.js:13:9
    at readConfig (/path/to/script.js:3:5)
    at getUser (/path/to/script.js:11:5)
    at Object.<anonymous> (/path/to/script.js:28:1)
    at Module._compile (module.js:404:26)
    at Object..js (module.js:410:10)
```



Callback-driven парадигма

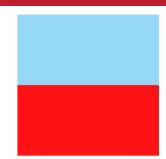
```
Error: User not found
    at afterDbConnect (/path/to/script.js:24:14)
    at /path/to/script.js:20:9
    at dbConnect (/path/to/script.js:7:5)
    at afterReadConfig (/path/to/script.js:18:5)
    at /path/to/script.js:13:9
    at readConfig (/path/to/script.js:3:5)
    at getUser (/path/to/script.js:11:5)
    at Object.<anonymous> (/path/to/script.js:28:1)
    at Module._compile (module.js:404:26)
    at Object..js (module.js:410:10)
```



Callback-driven парадигма

```
function getUser(id, callback) {
    readConfig('config.json', function(err, config){
        if (err) return callback(err);
        afterReadConfig(id, config, callback);
    })
}

function afterReadConfig(id, config, callback) {
    dbConnect(config.host, function(err, db){
        if (err) return callback(err);
        afterDbConnect(id, db, callback);
    })
}
function afterDbConnect(id, db, callback) …
```

Синтаксический шум - плата за Evented I/O



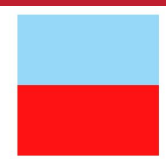
Callback-driven парадигма

```
function getUser(id, callback) {
    readConfig('config.json', function(err, config){
        if (err) return callback(err);
        afterReadConfig(id, config, callback);
    })
}

function afterReadConfig(id, config, callback) {
    dbConnect(config.host, function(err, db){
        if (err) return callback(err);
        afterDbConnect(id, db, callback);
    })
}
function afterDbConnect(id, db, callback) …
```

← PROFIT

Синтаксический шум - плата за Evented I/O

← PROFIT



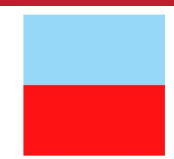
node-sync

Function.prototype.sync = function(context, arguments…)

- Использует сопрограммы (coroutines) c++
- Основан на node-fibers

  https://github.com/laverdet/node-fibers
- Позволяет писать синхронно на nodejs

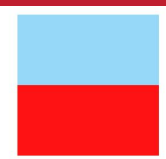
https://github.com/0ctave/node-sync



node-sync

```
var Sync = require('sync');
function getUser(id, callback) {
    Sync(function(){
        var config = readConfig.sync(null, 'config.json');
        var db = dbConnect.sync(null, config.host);
        var user = db.getUser.sync(db, id);
        return user;
    }, callback)
}
```

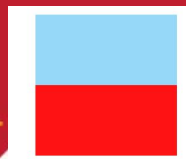
Запускаем новое «волокно» (Fiber)



node-sync

```
var Sync = require('sync');
function getUser(id, callback) {
    Sync(function(){
        var config = readConfig.sync(null, 'config.json');
        var db = dbConnect.sync(null, config.host);
        var user = db.getUser.sync(db, id);
        return user;
    }, callback)
}
```

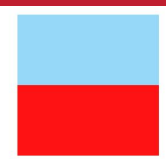
«Волокно» вернет значение или ошибку в callback



node-sync

```
var Sync = require('sync');
function getUser(id, callback) {
    Sync(function(){
        var config = readConfig.sync(null, 'config.json');
        var db = dbConnect.sync(null, config.host);
        var user = db.getUser.sync(db, id);
        return user;
    }, callback)
}
```

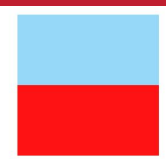
Функция readConfig вызывается синхронно и возвращает значение



node-sync

```
var Sync = require('sync');
function getUser(id) {
    var config = readConfig.sync(null, 'config.json');
    var db = dbConnect.sync(null, config.host);
    var user = db.getUser.sync(db, id);
    return user;
}.async()
```

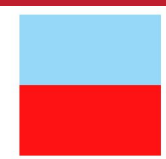
То же самое, только проще
(коллбэка нет)



node-sync

```
var Sync = require('sync');
function getUser(id) {
    var config = readConfig.sync(null, 'config.json');
    var db = dbConnect.sync(null, config.host);
    var user = db.getUser.sync(db, id);
    return user;
}.async()
```

```
getUser(1234, function(err, user) {
    if (err) return console.error(err);
    console.log('user: ', user);
})
```



node-sync

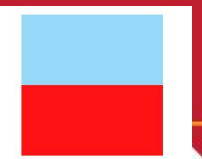
```
var Sync = require('sync');
function getUser(id) {
    var config = readConfig.sync(null, 'config.json');
    var db = dbConnect.sync(null, config.host);
    throw new Error('something went wrong');
    return user;
}.async()
```

```
getUser(1234, function(err, user) {
    if (err) return console.error(err);
    console.log('user: ', user);
})
```

node-sync

```
var Sync = require('sync');
function getUser(id) {
    var config = readConfig.sync(null, 'config.json');
    var db = dbConnect.sync(null, config.host);
    var user = db.getUser.future(db, id);
    var friends = db.getUserFriends.future(db, id);
    return { user : user.result, friends : friends.result };
}.async()
```

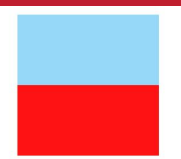
getUser и getUserFriends выполняются параллельно



node-sync

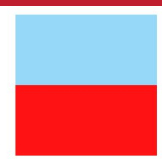
```
var Sync = require('sync');
function getUser(id) {
    var config = readConfig.sync(null, 'config.json');
    var db = dbConnect.sync(null, config.host);
    var user = db.getUser.future(db, id);
    db.getUserFriends(id, friends = new Sync.Future());
    return { user : user.result, friends : friends.result };
}.async()
```

другой способ получения «тикета» future



Callback-driven парадигма

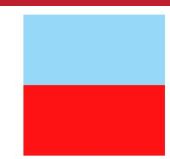
```
$pages = $db->fetchRows('SELECT * FROM pages');
foreach ($pages as $page) {
    $contents = fetchUrl($page->url);
}
```



Callback-driven парадигма

```
$pages = $db->fetchRows('SELECT * FROM pages');
foreach ($pages as $page) {
    $contents = fetchUrl($page->url);
}
```

OK

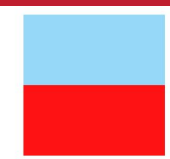


Callback-driven парадигма

```
$pages = $db->fetchRows('SELECT * FROM pages');
foreach ($pages as $page) {
    $contents = fetchUrl($page->url);
}
```

OK

```
db.fetchRows('SELECT * FROM pages', function(err, pages){
    pages.forEach(function(page){
        fetchUrl(page.url, function(err, contents) {

        })
    })
});
```



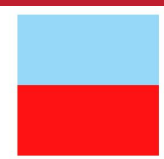
Callback-driven парадигма

```
$pages = $db->fetchRows('SELECT * FROM pages');
foreach ($pages as $page) {
    $contents = fetchUrl($page->url);
}
```

OK

```
db.fetchRows('SELECT * FROM pages', function(err, pages){
    pages.forEach(function(page){
        fetchUrl(page.url, function(err, contents) {

        })
    })
});
```

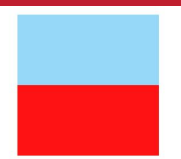
x 100,000



Callback-driven парадигма

```
    db.fetchRows('SELECT * FROM pages', function(err, pages){
+       var narrow = new Narrow(10, function(page, callback){
            fetchUrl(page.url, function(err, contents) {


            })
+       })
+       narrow.pushAll(pages);
    });
```

https://github.com/0ctave/node-narrow



Callback-driven парадигма

```
    db.fetchRows('SELECT * FROM pages', function(err, pages){
+       var narrow = new Narrow(10, function(page, callback){
            fetchUrl(page.url, function(err, contents) {


            })
+       })
+       narrow.pushAll(pages);
    });
```

x10 - OK

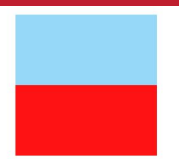
x 100,000 - OK

https://github.com/0ctave/node-narrow

Callback-driven решения?

- **node-sync** - использует сопрограммы (coroutines)
  https://github.com/0ctave/node-sync
- **streamline.js** – транслирует код
  https://github.com/Sage/streamlinejs
- **node-async** – целый инструментарий для асинхронного программирования
  https://github.com/caolan/async
- **Ваша собственная** flow-control библиотека :)
  http://www.scribd.com/doc/40366684/Nodejs-Controlling-Flow

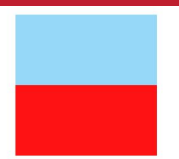
https://github.com/joyent/node/wiki/modules#async-flow



Масштабирование

Nodejs – is just node

(c) Ryan Dahl



Масштабирование

Nodejs – is just node

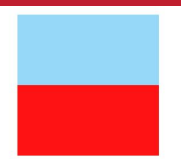
(c) Ryan Dahl

1 ядро CPU = 1 nodejs процесс

Масштабирование

- **node-cluster**
  - Расширяемый
  - Поддержка POSIX сигналов
  - «Горячая» перезагрузка (zero-downtime)
  - «Аккуратное» завершение (graceful shutdown)
  - Автоматом перезапускает мертвые процессы
  - Не оставляет «зомби»
  - Автоматом определяет соличество ядер CPU
  - Поддержка REPL
  - Статистика
  - PID файлы
  - Логи

https://github.com/LearnBoost/cluster

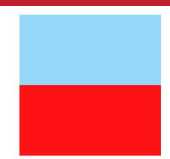


HTTP Cluster

```
var http = require('http'), cluster = require('cluster');
var server = http.createServer(function(req, res){
    res.writeHead(200, { 'Content-Type' : 'text/plain' });
    res.end('Hello from ' + process.pid + '\n');
});
cluster(server).listen(3080);
console.log('Server at http://127.0.0.1:3080/ (pid: %d)', process.pid);
```

Подключаем модуль cluster

https://github.com/LearnBoost/cluster



HTTP Cluster

```
var http = require('http'), cluster = require('cluster');
var server = http.createServer(function(req, res){
    res.writeHead(200, { 'Content-Type' : 'text/plain' });
    res.end('Hello from ' + process.pid + '\n');
});
cluster(server).listen(3080);
console.log('Server at http://127.0.0.1:3080/ (pid: %d)', process.pid);
```

Оборачиваем http сервер в кластер и «вешаем» на 3080 порт

https://github.com/LearnBoost/cluster

HTTP Cluster

```
var http = require('http'), cluster = require('cluster');
var server = http.createServer(function(req, res){
    res.writeHead(200, { 'Content-Type' : 'text/plain' });
    res.end('Hello from ' + process.pid + '\n');
});
cluster(server).listen(3080);
console.log('Server at http://127.0.0.1:3080/ (pid: %d)', process.pid);
```

Дополнительно выводим PID

https://github.com/LearnBoost/cluster

HTTP Cluster

```
var http = require('http'), cluster = require('cluster');
var server = http.createServer(function(req, res){
    res.writeHead(200, { 'Content-Type' : 'text/plain' });
    res.end('Hello from ' + process.pid + '\n');
});
cluster(server).listen(3080);
console.log('Server at http://127.0.0.1:3080/ (pid: %d)', process.pid);
```

```
$ node my_app.js
Server at http://127.0.0.1:3080/ (pid: 9254)
Server at http://127.0.0.1:3080/ (pid: 9255)
Server at http://127.0.0.1:3080/ (pid: 9256)
```

https://github.com/LearnBoost/cluster

HTTP Cluster

```
var http = require('http'), cluster = require('cluster');
var server = http.createServer(function(req, res){
    res.writeHead(200, { 'Content-Type' : 'text/plain' });
    res.end('Hello from ' + process.pid + '\n');
});
cluster(server).listen(3080);
console.log('Server at http://127.0.0.1:3080/ (pid: %d)', process.pid);
```

$ node my_app.js Server at http://127.0.0.1:3080/ (pid: 9254) Server at http://127.0.0.1:3080/ (pid: 9255) Server at http://127.0.0.1:3080/ (pid: 9256)	**$ curl http://127.0.0.1:3080/** Hello from 9255 **$**

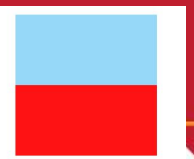
https://github.com/LearnBoost/cluster

HTTP Cluster

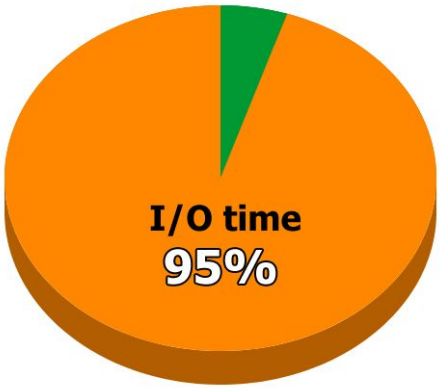
```
var http = require('http'), cluster = require('cluster');
var server = http.createServer(function(req, res){
    res.writeHead(200, { 'Content-Type' : 'text/plain' });
    res.end('Hello from ' + process.pid + '\n');
});
cluster(server).listen(3080);
console.log('Server at http://127.0.0.1:3080/ (pid: %d)', process.pid);
```

$ node my_app.js Server at http://127.0.0.1:3080/ (pid: 9254) Server at http://127.0.0.1:3080/ (pid: 9255) Server at http://127.0.0.1:3080/ (pid: 9256)	**$ curl http://127.0.0.1:3080/** Hello from 9255 **$ curl http://127.0.0.1:3080/** Hello from 9256 **$**

https://github.com/LearnBoost/cluster

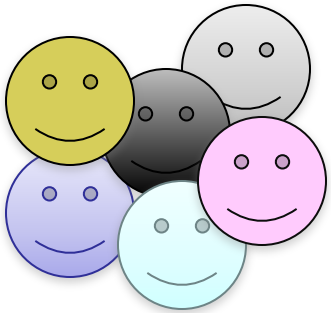


NodeJS



Много I/O

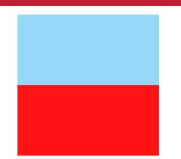
\+



Много
Запросов

\+

= PROFIT



Юра Богданов

About.me: http://about.me/bogdanov

Github: https://github.com/0ctave

Twitter: http://twitter.com/yuriybogdanov

LinkedIn: http://linkedin.com/in/yuriybogdanov

Email: octave@eventr.com

Спасибо за внимание.