Введение в технологию

От JavaScript никуда не деться

«Всё, что может быть написано на JavaScript, должно быть написано на нём»

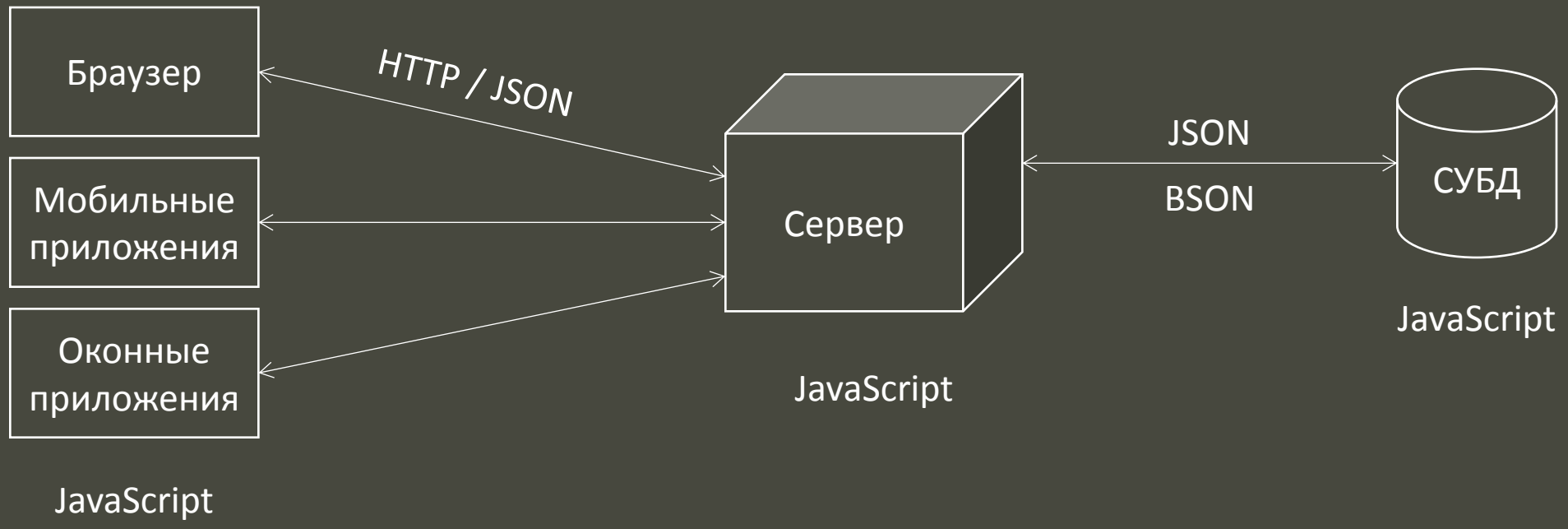
«Any application that can be written in JavaScript, will eventually be written in JavaScript»

Jeff Atwood

основатель Stack Overflow

Основные идеи Node.js

1. Один язык, один формат данных, одна парадигма, одна архитектура



Основные идеи Node.js

2. Долгая жизнь приложений в оперативной памяти на стороне сервера

- Состояние памяти сохраняется между запросами
  - Структуры данных (объекты, модель)
  - Соединения с БД и открытые файлы
  - Конфигурация и инициализация
- Веб сервер внутри приложения, а не наоборот
  - Меньше IPC

Основные идеи Node.js

3. Без ввода/вывода еще быстрее, чем с асинхронным вводом/выводом

- Упреждающее чтение
- Отложенная запись
- Отдача всего из памяти

Преимущества JavaScript:

1. Имеет очень низкий порог входа;
2. Высочайшая гибкость;
3. Высокая скорость разработки;
4. Практически идеальный уровень абстракции;

…

Недостатки JavaScript:

1. Имеет очень низкий порог входа;
2. Высочайшая гибкость;
3. Высокая скорость разработки;
4. Практически идеальный уровень абстракции;

…

Особенности JavaScript

```
[] + [] === ‘’                    [] + {} === ‘[object Object]’
{} + [] === 0                     isNaN({} + {}) === true
‘12’ + 5 === ‘125’ но…  ’12’ - 5 === 7
NaN !== NaN                       typeof (NaN) === 'Number'
```

```
var pow = Math.pow;
Math.pow = function(base, exponent) {
  return pow(base, exponent) + 1;
};
```

Что такое NodeJS

- Open Source JavaScript-движок V8 от Google;
- Обёртка и библиотеки на JavaScript;
- libuv;
- Оболочка, написанная на C++.

Google V8

- Разработка Google, исходный код которой был открыт в 2008 году;
- Самый оптимальный JavaScript-движок на сегодняшний день;
- Основа браузеров Google Chrome и Chromium.

Преимущества V8

- Базовый и оптимизирующий компилятор;
- Компиляция JavaScript-кода непосредственно в машинный код, без промежуточного байт-кода;
- Эффективная система управления памятью;
- Введение скрытых классов и встроенных кэшей, ускоряющих доступ к свойствам и вызовы функций.

Начало работы с NodeJS

1. Установка дистрибутива с ***nodejs.org***;
2. Создание ****.js*** файлов;
3. Запуск программ: ***$ node *.js***;

Элементарный пример

hello.js:

```
console.log("Hello, world!");
```

```
$ node hello
Hello, world!
```

API, менеджер пакетов и package.json

```
$ npm init
$ npm install colors --save
/
  /node_modules
    /colors
      ...
  main.js
  package.json

var http = require("http");
```

```
net
https
buffer
crypto
path
…
```

package.json

```
{
  "name": "Test",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}
```

Система контроля версий

- Git;
- GitHub;
- Структура проектов;
- Репозитории npm;
    - 322 953 (на 9 августа 2016)
    - left-pad
    - конфликты версий

Однопоточность и многопоточность

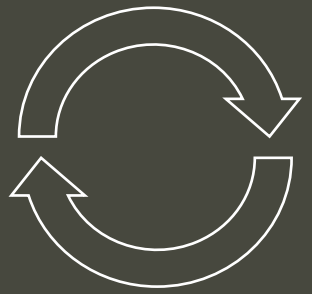
- Что дает однопоточность?
- Действительно ли 1 процесс = 1 поток?
- Как нагрузить все ядра?
  - cluster = require('cluster')
  - npm install webworker-threads
  - самостоятельно делаем IPC

Межпроцессовое взаимодействие

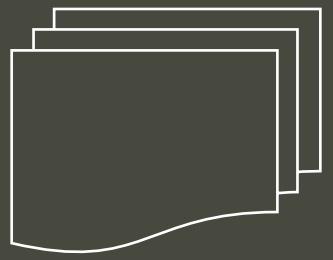
- Родное IPC, cluster
- Разделяемая память: Memcache?
- Шина сообщений (MQ): ZeroMQ, RabbitMQ...
- TCP, UDP, HTTP

Асинхронный ввод/вывод

- Реактор



- Очередь



- CMO

```
fs.readFile(
      'a.txt',
      onRead
);
http.get(
      'http://...',
      onGet
);
console.log('done?');
```

Производительность

- Производительность кода
- Производительность сервера

Факторы производительности Node.js

- Асинхронность
- Состояние в памяти
- Минимизация IPC

Память

- GC (RAM 32bit: 512Mb, 64bit:1.7Gb)
- node --max-old-space-size=3000
- delete obj.field; delete arr[5]; delete name;
- node --nouse-idle-notification
  --expose-gc application.js
- Вызываем gc();

Профилирование

- npm install -g node-inspector

node --debug application.js

- npm install heapdump

```
var hd = require('heapdump');
hd.writeSnapshot(
    'dump'+Date.now()+'.heapsnapshot'
);
```

Отладка

- Через node-inspector
- Выводом в консоль console.log(); или в лог
- Если ошибку не видно
  - node --stack-trace-limit=1000 application.js
  - Может понадобиться фильтр

Ошибки

- Общепринято в Node: function(err, arguments...)
- Создаем new Error('Eror message')
- Секция try {...} catch(e) {...} finally {...}
- Событие .on('error', fn) если EventEmitter
- Домены domain.create(); .run(fn); .on('error', fn);
- Перехват

```
process.on('uncaughtException', function(err) {
  console.log('uncaught Exception: '+err.stack);
});
```

Обновление кода и перезапуск

- Наблюдение за процессом forever

```
npm install forever -g
forever start --spinSleepTime 10000 application.js
```

- Наблюдение за файлами

http://livereload.com/
http://habrahabr.ru/post/168091/

Callback’и и вложенность

Пример запроса:

```
request('http://www.google.com/api',
  function (error, response, body) {
      var googleData = JSON.parse(body);
      ...
  }
);
```

Callback’и и вложенность

```
request('http://www.google.com/api',
  function (gError, gResponse, gBody) {
    request('http://www.yandex.com/api',
      function (yError, yResponse, yBody) {
        request('http://www.twitter.com/api',
          function (tError, tResponse, tBody)
            {
              ...
            }
            ...
```

Callback'и: Библиотека async

```
async.parallel([
    function(){ ... },
    function(){ ... }
], callback);
```

Другие методы: series, parallel, whilst, until, waterfall, compose, queue, cargo, retry, each, map, reduce, detect, every ...

Callback'и: Библиотека async

```
async.parallel([
    function(callback){ request('http://google.com/api',
        function(error, response, body) { callback(error, body); });
    },
    function(callback){ request('http://yandex.com/api',
        function(error, response, body) { callback(error, body); });
    },
    function(callback){ request('http://twitter.com/api',
        function(error, response, body) { callback(error, body); });
    },
], function(err, results) {
      ...
});
```

Callback'и: Библиотека async

```
async.parallel([
    function(callback){ googleApiService(callback); },
    function(callback){ yandexApiService(callback); },
    function(callback){ twitterApiService(callback); },
], function(err, results) {
      ...
});
```

Callback'и: Promise (Q)

https://github.com/kriskowal/q

```
Promise.then(function(success) {
    ...
}, function(error) {
    ...
});
```

Callback'и: Promise (Q)

```
Q.fcall(googleApiService)
  .then(yandexApiService)
  .then(twitterApiService);

Q.all([
     googleApiService(),
     yandexApiService(),
     twitterApiService()
   ])
   .then();
```

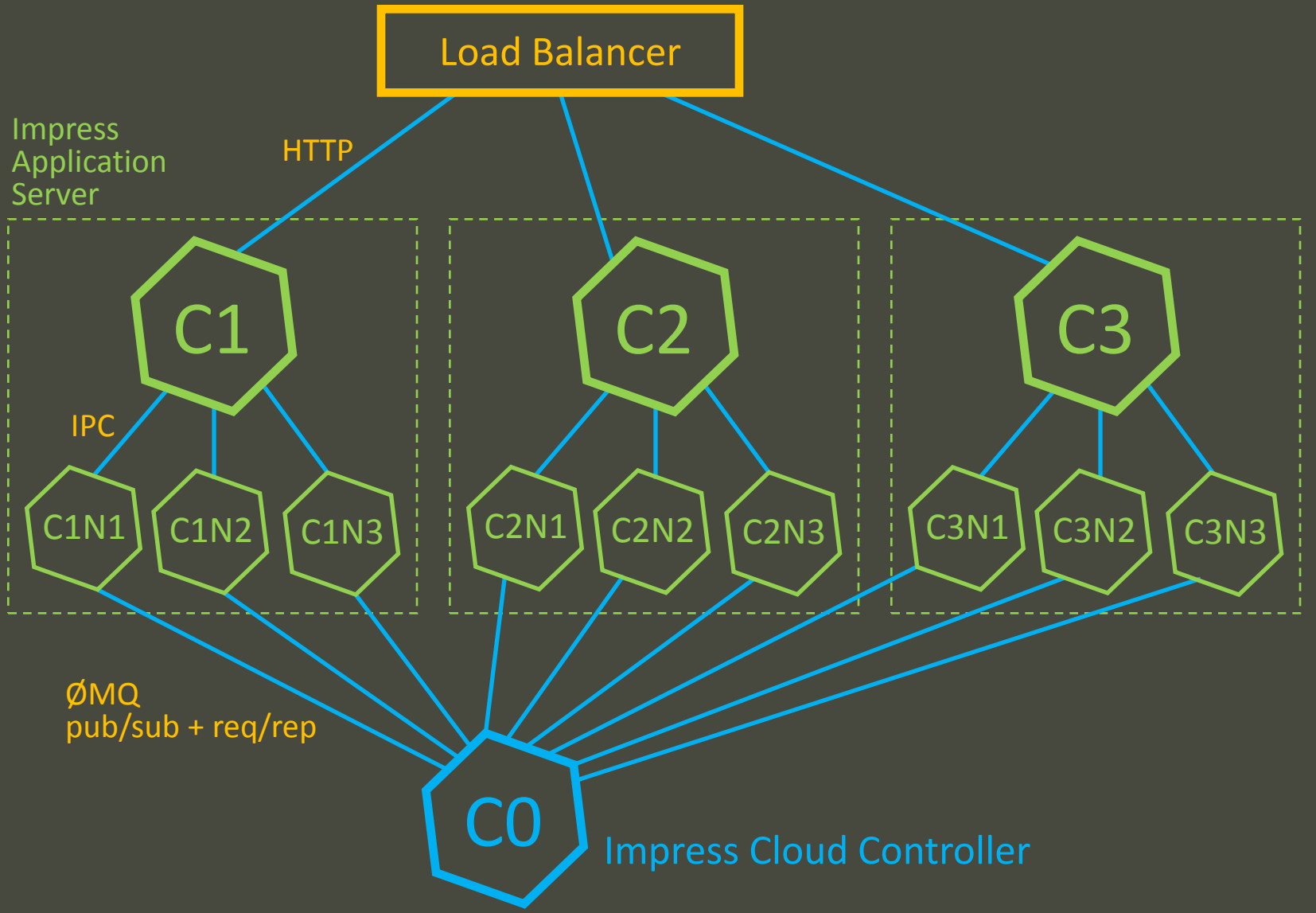
Примеры внедрения и Highload-проекты

Интерактивное телевидение 7Sense

- 10млн. открытых соединений на 15 серверах
- 30тыс. request/second на одном сервере

- Impress Application Server
- Server-Sent Events
- IPC, ZeroMQ

Масштабирование Impress App Server



Жизненный цикл проекта

- Юниттесты и интеграционные тесты
- Continuous Integration (CI)
  - Travis-CI
  - drone.io
- Развертывание среды
  - vagrant
  - docker

Сферы применения node.js

Часто применяется для:

- сервера веб-приложений и SPA
- сервера мобильных приложений
- системы сборки для фронт-энда
- чаты, меседжинг
- игровые сервера
- как заплатка к ПО на других технологиях
- парсеры, кравлеры

Сферы применения node.js

Реже применяется для:

- оконные приложения: node-webkit
- приложения баз данных и корпоративные
- промышленная автоматизация и программирование микроконтроллеров (arduino, espruino, tessel)
- обработка и трансляция видео и звука

Сферы применения node.js

Редко применяется (но подходит) для:

- CMS, публикация контента
- электронная коммерция и торговля

И плохо подходит:

- вычисления и моделирование
- научные приложения

Полезные материалы

- npm (www.npmjs.org);
- nodejs.org/api;
- learn.javascript.ru;
- learn.javascript.ru/nodejs-screencast;
- nodeschool.io;
- habrahabr.ru/hub/nodejs;
- nodeguide.ru;
- …