Node.js

Node.js

программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код) превращающая JavaScript из узко специализированного языка в язык общего назначения

Node.js

- Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода/вывода через свой API (написанном на C++)
- подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript кода
- Node.js применяется преимущественно на сервере, выполняя роль веб-сервера
- есть возможность разрабатывать на Node.js и десктопные оконные приложения (при помощи node-webkit и AppJS для Linux, Windows и MacOS) и даже программировать микроконтроллеры

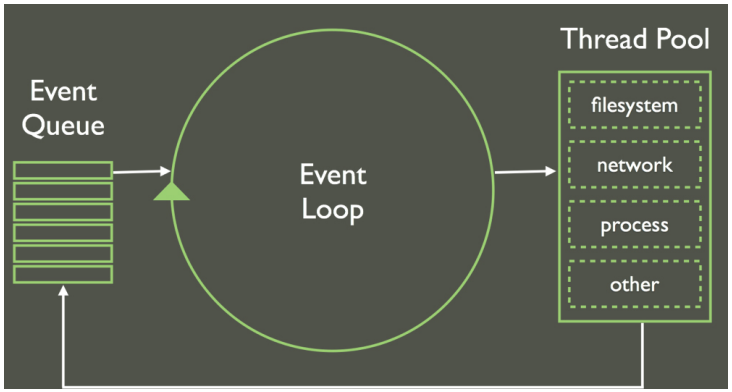
Применения

- **1. Локальные приложения и утилиты командной строки**
  - Сборщики и трансляторы
  - Пакетная обработка и сценарии отложенной обработки
  - Скрипты, CLI (интерфейсы командной строки)
  - Генерация документации, отложенное формирование отчетов
  - Сценарии тестирования для других систем
- **2. Серверы**
  - Серверы веб-приложений
  - Серверы и API для мобильных приложений
  - Любые другие веб-API (RPC, JSON, REST)
  - Серверы сообщений и трансляция событий (чаты, игры, интерактив)
- **3. Клиенты**
  - Оконные приложения (nw.js, node-webkit)
  - Кравлеры, парсеры и сбор данных
- **4. Железо**
  - Программирование микроконтроллеров
  - Промышленная автоматизация

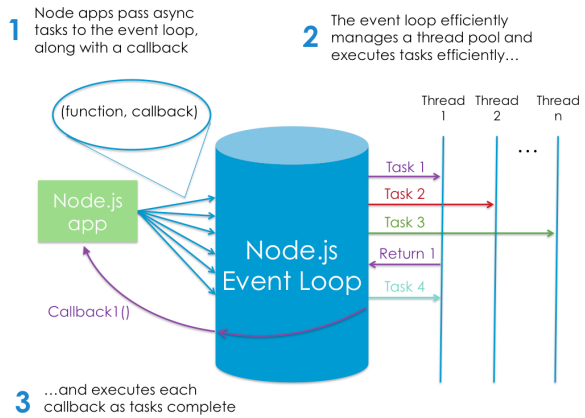
История

Node.js разработал Райан Дал (англ. **Ryan Dahl**) в 2009 году после двух лет экспериментирования над созданием серверных веб-компонентов. В ходе своих исследований он пришёл к выводу, что вместо традиционной модели параллелизма на основе потоков следует обратиться к событийно-ориентированным системам.

События



События



Интерактивная оболочка

- Если установка прошла без ошибок, то станет доступен вызов интерактивной оболочки Node.js, например:

```
$ node
> console.log('Hello World');
Hello World
```

- Интерактивная оболочка удобна для тестирования простых однострочных примеров. Кроме того, она может быть внедрена в любое Node.js приложение.
- Для того, чтобы выйти из интерактивной оболочки, необходимо просто нажать Ctrl + C

Модули

```
module.exports = function(string) {
  return string.split("").map(function(ch) {
    return String.fromCharCode(ch.charCodeAt(0) + 5);
  }).join("");
};
```

```
var garble = require("./garble");

// По индексу 2 содержится первый аргумент программы из командной строки
var argument = process.argv[2];

console.log(garble(argument));
```

Модуль file system

```
var fs = require("fs");
fs.readFile("file.txt", "utf8", function(error, text) {
  if (error)
    throw error;
  console.log("А в файле том было:", text);
});
```

```
var fs = require("fs");
fs.writeFile("graffiti.txt", "Здесь был Node ", function(err) {
  if (err)
    console.log("Ничего не вышло, и вот почему:", err);
  else
    console.log("Запись успешна. Все свободны.");
});
```

Модуль HTTP

```
var http = require("http");
var server = http.createServer(function(request, response) {
  response.writeHead(200, {"Content-Type": "text/html"});
  response.write("<h1>Привет!</h1><p>Вы запросили <code>" +
                 request.url + "</code></p>");
  response.end();
});
server.listen(8000);
```

Модуль HTTP

```
var http = require("http");
var request = http.request({
  hostname: "eloquentjavascript.net",
  path: "/20_node.html",
  method: "GET",
  headers: {Accept: "text/html"}
}, function(response) {
  console.log("Сервис ответил с кодом ",
              response.statusCode);
});
request.end();
```

Потоки

- **Потоки с возможностью записи**
  - метод **write**, которому можно передать строку или объект Buffer
  - метод **end** закрывает поток, а при наличии аргумента, выведет перед закрытием кусочек данных
  - методам можно задать функцию обратного вызова через дополнительный аргумент, которую они вызовут по окончанию записи или закрытию потока
- **Потоки с возможностью чтения**
  - чтение из потока осуществляется через обработчики событий, а не через методы
  - метод **on**, схожий с методом браузера addEventListener
  - события «**data**» и «**end**»

Пример

```
var http = require("http");
http.createServer(function(request, response) {
  response.writeHead(200, {"Content-Type": "text/plain"});
  request.on("data", function(chunk) {
    response.write(chunk.toString().toUpperCase());
  });
  request.on("end", function() {
    response.end();
  });
}).listen(8000);
```

Дополнительные пакеты

- В состав Node.js входит собственный установщик пакетов npm
- Установка производится при помощи команды:

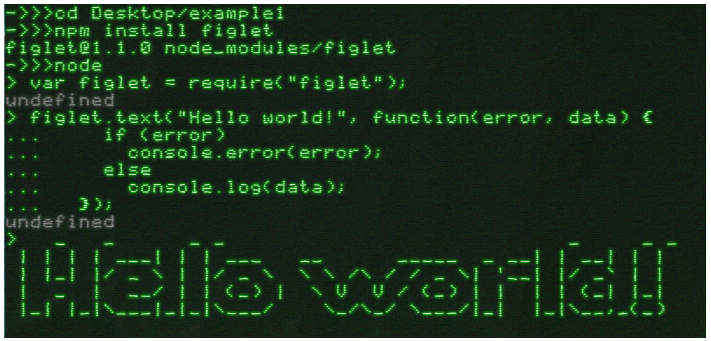
npm install <packagename>

- Все доступные для установки пакеты и их краткое описание:

npm search

Пример

```
->>>cd Desktop/example1
->>>npm install figlet
figlet@1.1.0 node_modules/figlet
->>>node
> var figlet = require("figlet");
undefined
> figlet.text("Hello world!", function(error, data) {
...     if (error)
...       console.error(error);
...     else
...       console.log(data);
...   });
undefined
>
```



Инструменты

- **Desktop IDEs**
  - Atom (free open-source)
  - Brackets (free open-source)
  - Sublime Text (commercial)
  - JetBrains IntelliJ IDEA (commercial)
  - JetBrains WebStorm (commercial)
  - Microsoft Visual Studio with Node.js Tools for Visual Studio[34] (commercial)
  - Microsoft Visual Studio with TypeScript (commercial)
  - Nodeclipse Enide Studio (free open-source, Eclipse-based)
  - NoFlo – flow-based programming environment integrated with GNOME APIs[35]
- **Online code editors**
  - Codenvy IDE (cloud service)
  - Cloud9 IDE (cloud service)
  - Codiad (Self hosted service)

Итоги

- JavaScript код выполняется вне браузера
- использует движок V8 VM от Google - ту же самую среду исполнения для JavaScript, которую использует браузер Google Chrome
- Node.js поставляется со множеством полезных модулей
- установка на Windows и Linux - на сайте nodejs.org