**Node.js** adalah perangkat lunak yang didesain untuk mengembangkan aplikasi berbasis web dan ditulis dalam sintaks bahasa pemrograman JavaScript. Bila selama ini kita mengenal JavaScript sebagai bahasa pemrograman yang berjalan di sisi client / browser saja, maka Node.js ada untuk melengkapi peran JavaScript sehingga bisa juga berlaku sebagai bahasa pemrograman yang berjalan di sisi server, seperti halnya PHP, Ruby, Perl, dan sebagainya. Node.js dapat berjalan di sistem operasi Windows, Mac OS X dan Linux tanpa perlu ada perubahan kode program. Node.js memiliki pustaka server HTTP sendiri sehingga memungkinkan untuk menjalankan server web tanpa menggunakan program server web seperti Apache atau Nginx.

Untuk mengeksekusi Javascript sebagai bahasa server diperlukan engine yang cepat dan mempunyai performansi yang bagus. Engine Javascript dari Google bernama V8-lah yang dipakai oleh Node.js yang juga merupakan engine yang dipakai oleh browser Google Chrome.

**Cara kerja node js**

Berbeda dengan bahasa pemrograman sisi server pada umumnya yang bersifat *blocking*, Node.js bersifat *non-blocking*, sebagaimana halnya JavaScript bekerja. Node.js berjalan dengan basis event (*event-driven*). Maksud dari *Blocking* secara sederhana adalah, bahwa suatu kode program akan dijalankan hingga selesai, baru kemudian beralih ke kode program selanjutnya.

Misalkan kita memiliki program dengan algoritma berikut:

1. Terima rekues untuk halaman blog
2. ambil data blog dari database
3. tulis html berisi data blog
4. kirim respon ke klien

Bila kita menggunakan bahasa pemrograman *blocking* yang bersifat *multi-thread*, pada poin kedua saat program mengambil data dari database selama jangka waktu tertentu pada satu *thread*, maka *thread* lain yang disiapkan untuk menjalankan poin ketiga (yakni menulis data dari database ke dalam bentuk HTML) tidak akan dijalankan hingga thread untuk poin kedua mengembalikan data. Ini sebenarnya bukan masalah karena CPU menjalankan proses dengan sangat cepat. Hanya saja, Node.js dengan *single-threaded*-nya melakukan dengan cara yang berbeda.

Node.js, alih-alih mengalokasikan thread untuk setiap poin, dia hanya akan membuat thread hanya apabila ada event yang membutuhkan hal tersebut. Contohnya untuk kasus di atas, ketika program mengambil data dari database, program hanya akan membuat thread atau dengan kata lain mulai memproses poin ketiga hanya apabila data dari database sudah diterima, menggunakan *callback*. Selain itu, bila ada proses lain yang tidak tergantung pada data tadi, maka proses tersebut akan dijalankan tanpa harus menunggu pengambilan data selesai.

kelebihan

1. Node.js menggunakan bahasa pemrograman JavaScript yang diklaim sebagai bahasa pemrograman yang paling populer dan banyak dikenal oleh masyarakat luas
2. Node.js mampu menangani ribuan koneksi bersamaan dengan penggunaan *resource* minimum untuk setiap prosesnya
3. Node.js sangat diandalkan terutama untuk membuat aplikasi *real-time*
4. Node.js adalah project open source, sehingga siapapun dapat melihat struktur kode dan juga dapat berkontribusi untuk pengembangannya
5. Penggunaan JavaScript di sisi server dan juga client meminimalisir ketidakcocokan antar dua sisi lingkungan pemrograman, seperti terkait komunikasi data yang mana menggunakan struktur JSON yang sama di kedua sisi, validasi form yang sama yang dapat dijalankan di sisi server dan client, dan sebagainya.
6. Database NoSQL seperti MongoDB dan CouchDB mendukung langsung Javascript sehingga interfacing dengan database ini akan jauh lebih mudah.
7. Node.js memakai V8 yang selalu mengikuti perkembangan standar ECMAScript (nama standar resmi dari JavaScript, Namun JavaScript yang lebih dikenal dalam implementasinya), sehingga tidak perlu ada kekhawatiran bahwa browser tidak mendukung fitur-fitur di Node.js.

Tools pendukung

1. Build tools merupakan perangkat lunak yang membantu kita dalam melakukan otomasi manajemen file. Biasanya kita hanya perlu memberitahukan kepada build tools file mana yang ingin dikompilasi, dipindahkan, dan sejenisnya. Setelah memberitahukan bagaimana manajemen file yang kita inginkan, kita cukup memanggil build tools untuk menjalankan tugas-tugas tersebut secara otomatis. Terdapat sangat banyak build tools pada platform NodeJS, contohnya: [GruntJS](http://gruntjs.com/), [Mimosa](http://mimosa.io/), [Gulp](http://gulpjs.com/), dan lainnya. Pada tulisan ini kita hanya akan menggunakan Gulp saja, karena pembahasan akan keseluruhan build tools tidak akan memungkinkan.
2. Ketika kita mengembangkan aplikasi web dinamis, seringkali kita akan berhubungan dengan server. Hubungan dengan server ini biasanya dilakukan secara dinamis, dengan menggunakan teknologi seperti XMLHttpRequest atau WebSocket. Untuk memudahkan kita dalam menguji apakah server memberikan data yang tepat atau sesuai dengan yang kita inginkan, kita dapat menggunakan *tools* yang dapat mengirimkan permintaan ke server, dengan format yang kita inginkan.

Beberapa *tools* yang dapat digunakan untuk kepentingan ini yaitu:

1. [Advanced REST Client](https://chrome.google.com/webstore/detail/advanced-rest-client/hgmloofddffdnphfgcellkdfbfbjeloo) (Chrome)
2. [REST Client](https://addons.mozilla.org/en-US/firefox/addon/restclient/) (Firefox)

Kedua *tools* di atas cukup sederhana dan mudah digunakan. Kita hanya perlu memasukkan URL yang akan kita uji, beserta dengan *method* HTTP-nya. Program kemudian akan menampilkan data yang diberikan dari server, dan kita dapat langsung memastikan apakah format yang diberikan sudah benar atau tidak. Hal ini akan sangat memudahkan kita ketika ingin berbicara degnan server.

IDE

Visual Studio Code

WebStorm

Vim

Codenvy

Atom

NetBeans

Eclipse Che

Sublime Text

Notepad++

Brackets

Komodo IDE

Visual Studio Community Edition

CodeLite IDE

RedHat OpenShift.io

Aptana Studio

Codelobster

Libraries

V8

The V8 library provides Node.js with a JavaScript engine, which Node.js controls via the V8 C++ API. V8 is maintained by Google, for use in Chrome.

Documentation

libuv

Another important dependency is libuv, a C library that is used to abstract non-blocking I/O operations to a consistent interface across all supported platforms. It provides mechanisms to handle file system, DNS, network, child processes, pipes, signal handling, polling and streaming. It also includes a thread pool for offloading work for some things that can't be done asynchronously at the operating system level.

Documentation

http-parser

HTTP parsing is handled by a lightweight C library called http-parser. It is designed to not make any syscalls or allocations, so it has a very small per-request memory footprint.

Documentation

c-ares

For some asynchronous DNS requests, Node.js uses a C library called c-ares. It is exposed through the DNS module in JavaScript as the resolve() family of functions. The lookup() function, which is what the rest of core uses, makes use of threaded getaddrinfo(3) calls in libuv. The reason for this is that c-ares supports /etc/hosts, /etc/resolv.conf and /etc/svc.conf, but not things like mDNS.

Documentation

OpenSSL

OpenSSL is used extensively in both the tls and crypto modules. It provides battle-tested implementations of many cryptographic functions that the modern web relies on for security.

Documentation

zlib

For fast compression and decompression, Node.js relies on the industry-standard zlib library, also known for its use in gzip and libpng. Node.js uses zlib to create sync, async and streaming compression and decompression interfaces.

Documentation

Tools

npm

Node.js is all about modularity, and with that comes the need for a quality package manager; for this purpose, npm was made. With npm comes the largest selection of community-created packages of any programming ecosystem, which makes building Node.js apps quick and easy.

Documentation

gyp

The build system is handled by gyp, a python-based project generator copied from V8. It can generate project files for use with build systems across many platforms. Node.js requires a build system because large parts of it — and its dependencies — are written in languages that require compilation.

Documentation

gtest

Native code can be tested using gtest, which is taken from Chromium. It allows testing C/C++ without needing an existing node executable to bootstrap from.

Documentation