

1. Node.js

Javascript merupakan bahasa pemrograman yang lengkap hanya saja selama ini di pakai sebagai bahasa untuk pengembangan aplikasi web yang berjalan pada sisi client atau browser saja. Tetapi sejak ditemukannya Node.js oleh Ryan Dahl pada tahun 2009, Javascript bisa digunakan sebagai bahasa pemrograman di sisi server sekelas dengan PHP, ASP, C#, Ruby dll dengan kata lain Node.js menyediakan platform untuk membuat aplikasi Javascript dapat dijalankan di sisi server.

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

1.1. Javascript Di Server

Tak terelakkan bahwa Javascript merupakan bahasa pemrograman yang paling populer. Jika anda sebagai developer pernah mengembangkan aplikasi web maka penggunaan Javascript pasti tidak terhindarkan.

Sekarang dengan berjalannya Javascript di server lalu apa keuntungan yang anda peroleh dengan mempelajari Node.js, kurang lebih seperti ini :

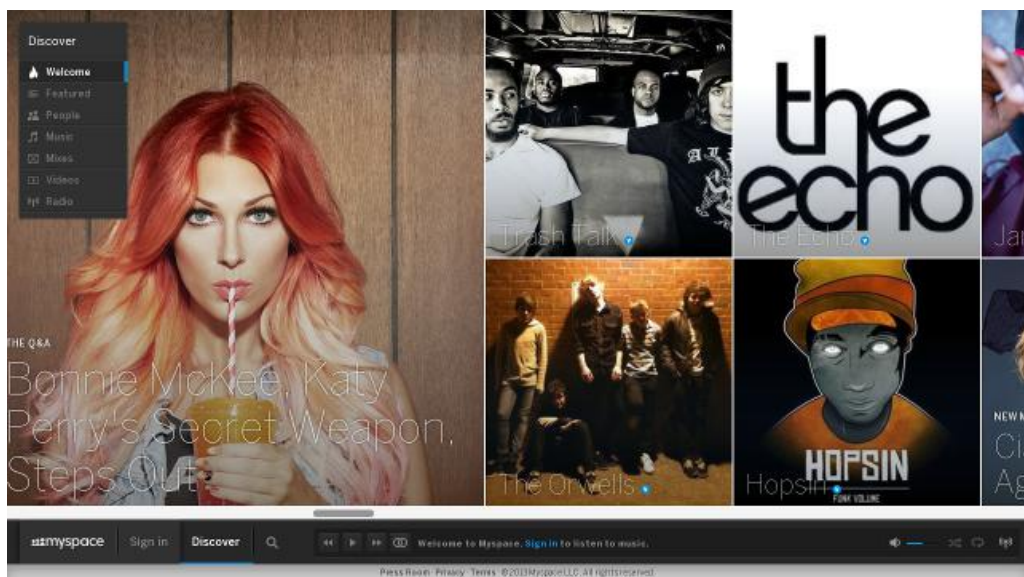
- Pengembang hanya memakai satu bahasa untuk mengembangkan aplikasi lengkap client & server sehingga mengurangi Learning Curve untuk mempelajari bahasa server yang lain.
- Sharing kode antara client dan server atau istilahnya code reuse.
- Javascript secara native mendukung JSON yang merupakan standar transfer data yang banyak dipakai saat ini sehingga untuk mengkonsumsi data-data dari pihak ketiga pemrosesan di Node.js akan sangat mudah sekali.
- Database NoSQL seperti MongoDB dan CouchDB mendukung langsung Javascript sehingga interfacing dengan database ini akan jauh lebih mudah.

- Node.js memakai V8 yang selalu mengikuti perkembangan standar ECMAScript, jadi tidak perlu ada kekhawatiran bahwa browser tidak akan mendukung fitur-fitur di Node.js.

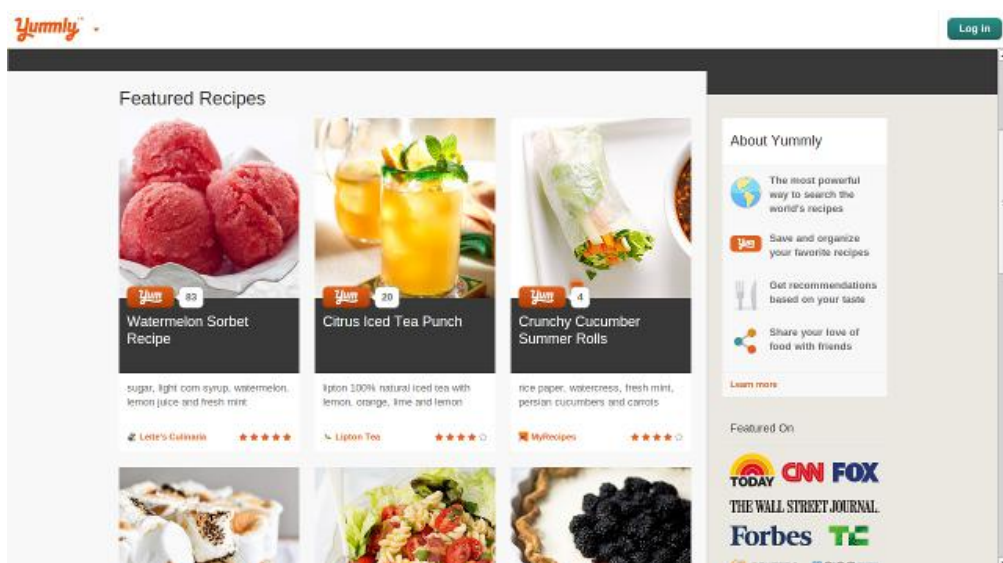
1.2. Node.js In Action

Supaya Anda lebih tertarik dalam belajar Node.js berikut beberapa website terkenal yang sudah memakai Node.

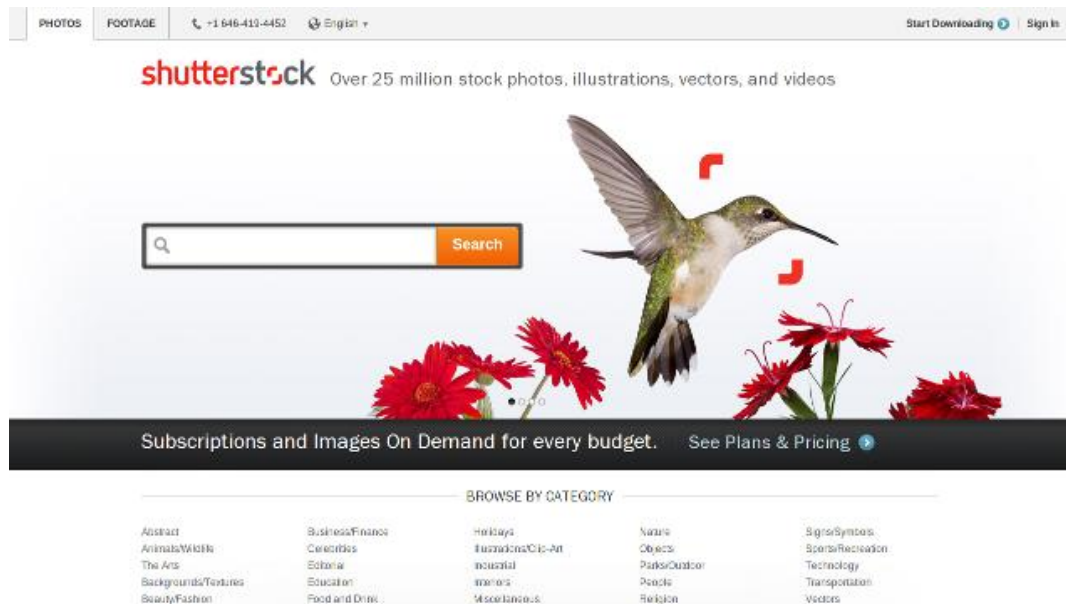
js:www.myspace.com



www.yummly.com



www.shutterstock.com



www.klout.com



www.geekli.st



www.learnboost.com



Apakah masih ragu untuk memakai Node.js ?...Kalau masih penasaran apa yang membuat Node.js berbeda dari backend pada umumnya, silahkan dilanjutkan membaca :smile:

2. Asinkron I/O & Event

Tidak seperti kebanyakan bahasa backend lainnya operasi fungsi di javascript lebih bersifat asinkron dan banyak menggunakan event demikian juga dengan Node.js. Sebelum penjelasan lebih lanjut mari kita lihat terlebih dahulu tentang metode sinkron seperti yang dipakai pada PHP dengan web server Apache.

2.1. PHP & Server HTTP Apache

Mari kita lihat contoh berikut yaitu operasi fungsi akses ke database MySQL oleh PHP yang dilakukan secara sinkron

```
$hasil = mysql_query("SELECT * FROM TabelAnggota");  
print_r($hasil);
```

Pengambilan data oleh `mysql_query()` diatas akan dijalankan dan operasi berikutnya `print_r()` akan diblok atau tidak akan berjalan sebelum akses ke database selesai. Yang perlu menjadi perhatian disini yaitu proses Input Output atau I/O akses ke database oleh `mysql_query()` dapat memakan waktu yang relatif mungkin beberapa detik atau menit tergantung dari waktu latensi dari I/O. Waktu latensi ini tergantung dari banyak hal seperti

- Query database lambat akibat banyak pengguna yang mengakses
- Kualitas jaringan untuk akses ke database jelek
- Proses baca tulis ke disk komputer database yang membutuhkan waktu
- ...

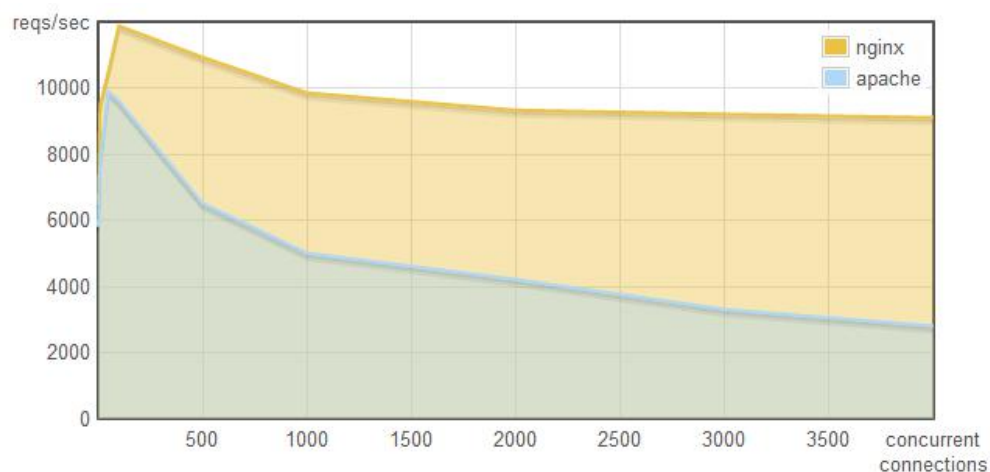
Sebelum proses I/O selesai maka selama beberapa detik atau menit tersebut state dari proses `mysql_query()` bisa dibilang idle atau tidak melakukan apa-apa.

Lalu jika proses I/O di blok bagaimana jika ada request lagi dari user? Apa yang akan dilakukan oleh server untuk menangani request ini? Penyelesaiannya yaitu dengan memakai pendekatan proses `multithread`. Melalui pendekatan ini tiap koneksi yang terjadi akan ditangani oleh `thread`.

Thread disini bisa dikatakan sebagai task yang dijalankan oleh prosesor komputer.

Sepertinya permasalahan I/O yang terblok terselesaikan dengan pendekatan metode ini tetapi dengan bertambahnya koneksi yang terjadi maka `thread` akan semakin banyak sehingga prosesor akan semakin terbebani, belum lagi untuk switching antar thread menyebabkan konsumsi memory (RAM) komputer yang cukup besar.

Berikut contoh benchmark antara web server Apache dan Nginx (server HTTP seperti halnya Apache hanya saja Nginx memakai sistem asinkron I/O dan event yang mirip Node.js). Gambar ini diambil dari goo.gl/pvLL4



Bisa dilihat bahwa Nginx bisa menangani request yang jauh lebih banyak daripada web server Apache pada jumlah koneksi bersama yang semakin naik.

2.2. Javascript & Node.js

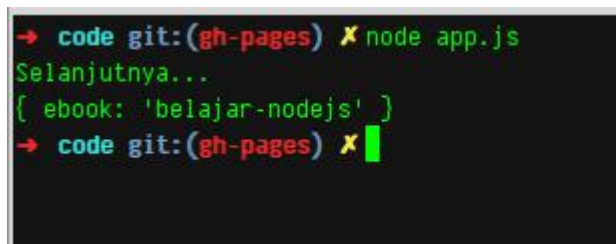
Kembali ke Javascript. Untuk mengetahui apa yang dimaksud dengan pemrograman asinkron bisa lebih mudah dengan memakai pendekatan contoh kode. Perhatikan kode Javascript pada Node.js berikut

```
var fs = require('fs');

fs.readFile('./resource.json', 'utf-8', function(err, data){
  if(err) throw err;
  console.log(JSON.parse(data));
});

console.log('Selanjutnya...');
```

Fungsi `readFile()` akan membaca isi dari file `resource.json` secara asinkron yang artinya proses eksekusi program tidak akan menunggu pembacaan file `resource.json` sampai selesai tetapi program akan tetap menjalankan kode Javascript selanjutnya yaitu `console.log('Selanjutnya...')`. Sekarang lihat apa yang terjadi jika kode javascript diatas dijalankan



```
→ code git:(gh-pages) X node app.js
Selanjutnya...
{ ebook: 'belajar-nodejs' }
→ code git:(gh-pages) X
```

Jika proses pembacaan file `resource.json` selesai maka fungsi callback pada `readFile()` akan di jalankan dan hasilnya akan ditampilkan pada console. Yah, fungsi callback merupakan konsep yang penting dalam proses I/O yang asinkron karena melalui fungsi callback ini data data yang dikembalikan oleh proses I/O akan di proses.

Lalu bagaimana platform Node.js mengetahui kalau suatu proses itu telah selesai atau tidak ?...jawabannya adalah [Event Loop](#). Event - event yang

terjadi karena proses asinkron seperti pada fungsi `fs.readFile()` akan ditangani oleh yang namanya Event Loop ini.

Campuran teknologi antara event driven dan proses asinkron ini memungkinkan pembuatan aplikasi dengan penggunaan data secara masif dan real-time. Sifat komunikasi Node.js I/O yang ringan dan bisa menangani user secara bersamaan dalam jumlah relatif besar tetapi tetap menjaga state dari koneksi supaya tetap terbuka dan dengan penggunaan memori yang cukup kecil memungkinkan pengembangan aplikasi dengan penggunaan data yang besar dan kolaboratif...Yeah, Node.js FTW! :metal: