Node.js

발표의 목적

Node.js의 애플리케이션을 개발하는 View 도구를 간단히 살펴본다.

Node.js의 작동 원리를 이해한다.

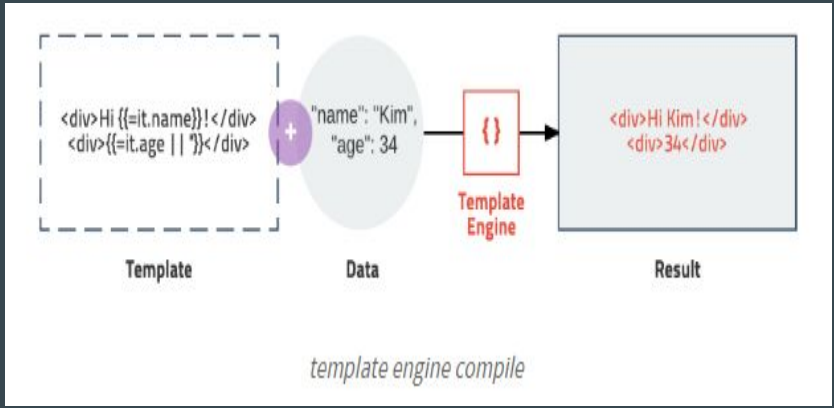
express 프레임워크를 사용하여, 얼마나 간단히 Node.js 애플리케이션이 생성되는 알게된다.

Node.js 목차

- front
  - Vue & express & ejs & pug에 대한 고민
  - 템플릿 엔진과 프레임워크
- back(Node.js)
  - Node.js 특징
  - CPU/Core/Thread???
  - Node.js Application 작동방식
  - IO Non-Blocking과 Blocking
  - express Application 만들고, 작동 방식 이해하기
  - Good 유즈케이스
  - Bad 유즈케이스
  - 결과적으로 Node.js는..

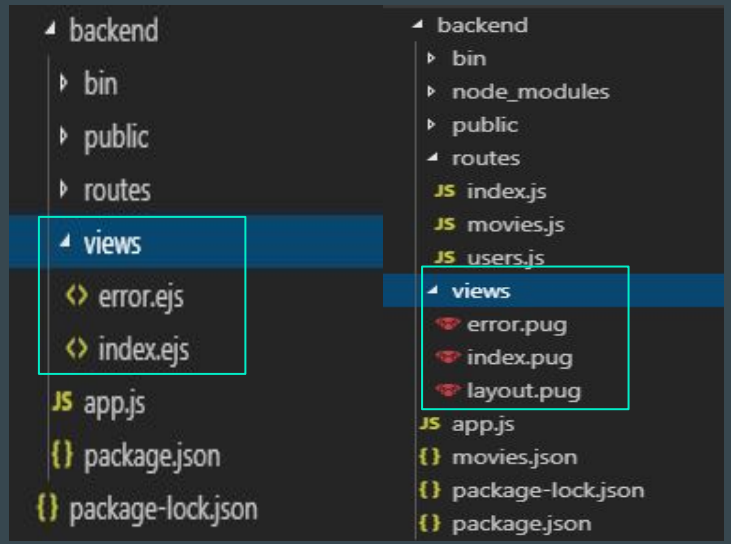
템플릿 엔진과 웹 프레임워크

템플릿 엔진(ejs, pug)
동적파일을 정적파일에서도 사용할 수 있게 해서
클라이언트의 요청마다 다른 결과페이지를
보여주는 일을 함.



> express --view=ejs [프레임워크 파일 생성명]

> express --view=pug [프레임워크 파일 생성명]



Front for Node.js

Express 프레임워크에서 템플릿 엔진(ejs, pug)

- 뷰 파일과 자바스크립트 코드를 한 파일에 정의하지 않고 분리해서 사용할 수 있음.
- 자바스크립트 코드로 연산된 결과를 변수에 넣고 변수를 뷰 파일에서도 사용할 수 있게끔 함.

Vue.js

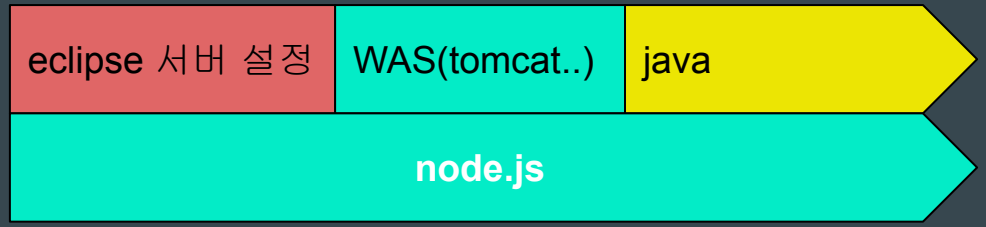
- **가상의 DOM추상화**를 사용. 변경된 부분만 실제 DOM에 반영... 이점에서 빠르다.
- DOM 조작에 가능한 적은 오버헤드(순수 javascript계산)만 가한다. Vue만의 특징

React.js

- MVC 프레임워크에서의 view 부분을 컴포넌트로 만들기 위한 라이브러리

Node.js의 특징

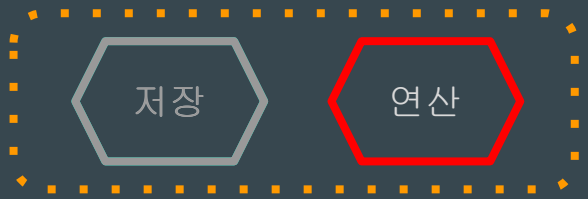
- JavaScript를 사용한다.
- 탁월한 생산성 보장한다.



- 순차방식 프로그래밍이 아닌 이벤트기반의 프로그래밍 모델을 사용한다.
- Node.js 는 Single Thread 기반으로 동작하는 비동기 IO(Async/Non-blocking)를 지원하는 네트워크 서버임.

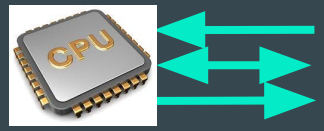
단일 스레드 구조 하나의 작업이 시간이 많이 소요될 경우 스레드가 처리해야 하는 작업이 밀리게 되고 전체적인 시스템 성능이 낮아진다. 하나의 작업시간이 작은 것 위주로 처리해야 한다.

CPU/Core/Thread???

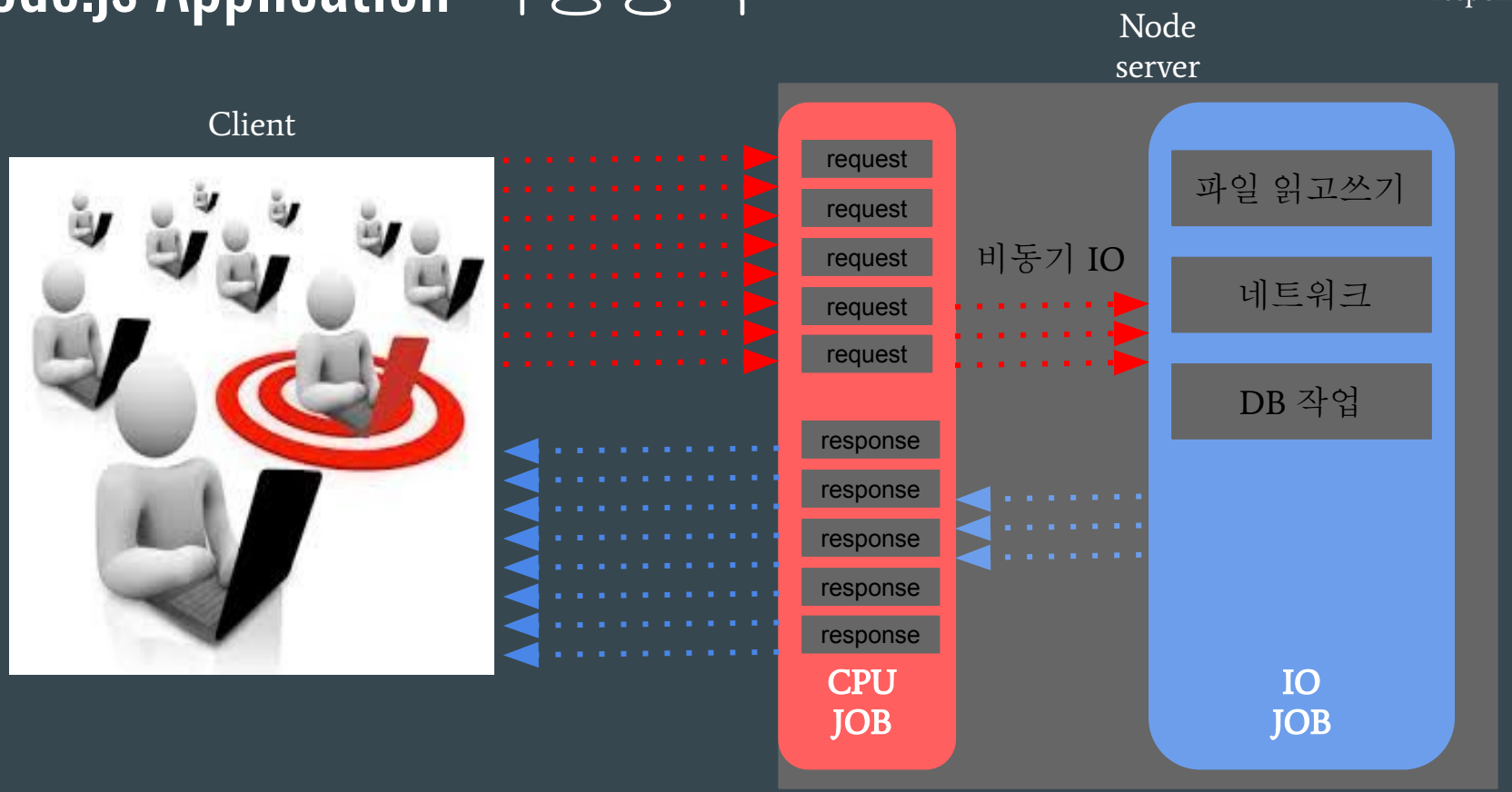


- CPU (!=Core)
  - 컴퓨터에 있는 모든 데이터는 이진수로 처리되므로, 어떤 데이터라 할지라도 CPU는 수많은 0과 1로 이루어진 데이터를 연산하여 다양한 결과를 도출한다. 즉, 컴퓨터 내부에서 이동하는 데이터는 0과 1로만 구성된 디지털 신호의 조합이다.
- 클럭
  - 이러한 디지털 신호를 빠르게 처리하는 연산 속도는 CPU마다 다르다. 속도를 나타내는 대표적인 단위는 클럭이다. 클럭이란 1초당 CPU 내부에서 몇 단계의 작업이 처리되는 지를 측정하여 주파수 단위인 헤르츠(Hz)로 나타낸 것이다. 클럭 수가 높으면 빠른 성능을 낸다.
- 쓰레드
  - 쓰레드는 데이터가 송신되고 수신되는 실행 흐름이다. Core는 쓰레드로부터 들어오는 데이터를 처리해서 쓰레드를 통해서 내보낸다. 한개의 통로로 송수신을 하는 것보다, 송신용 쓰레드, 수신용 쓰레드를 둬서 데이터를 송수신하는 것이 작업효율이 좋다.
  - 2코어 4쓰레드, 4코어 8쓰레드처럼 1코어당 2쓰레드란 개념은 CPU 최적화를 위해 만들어진 것으로 성능이 15%정도 향상된다고 한다,

1Core/ 2 Thread

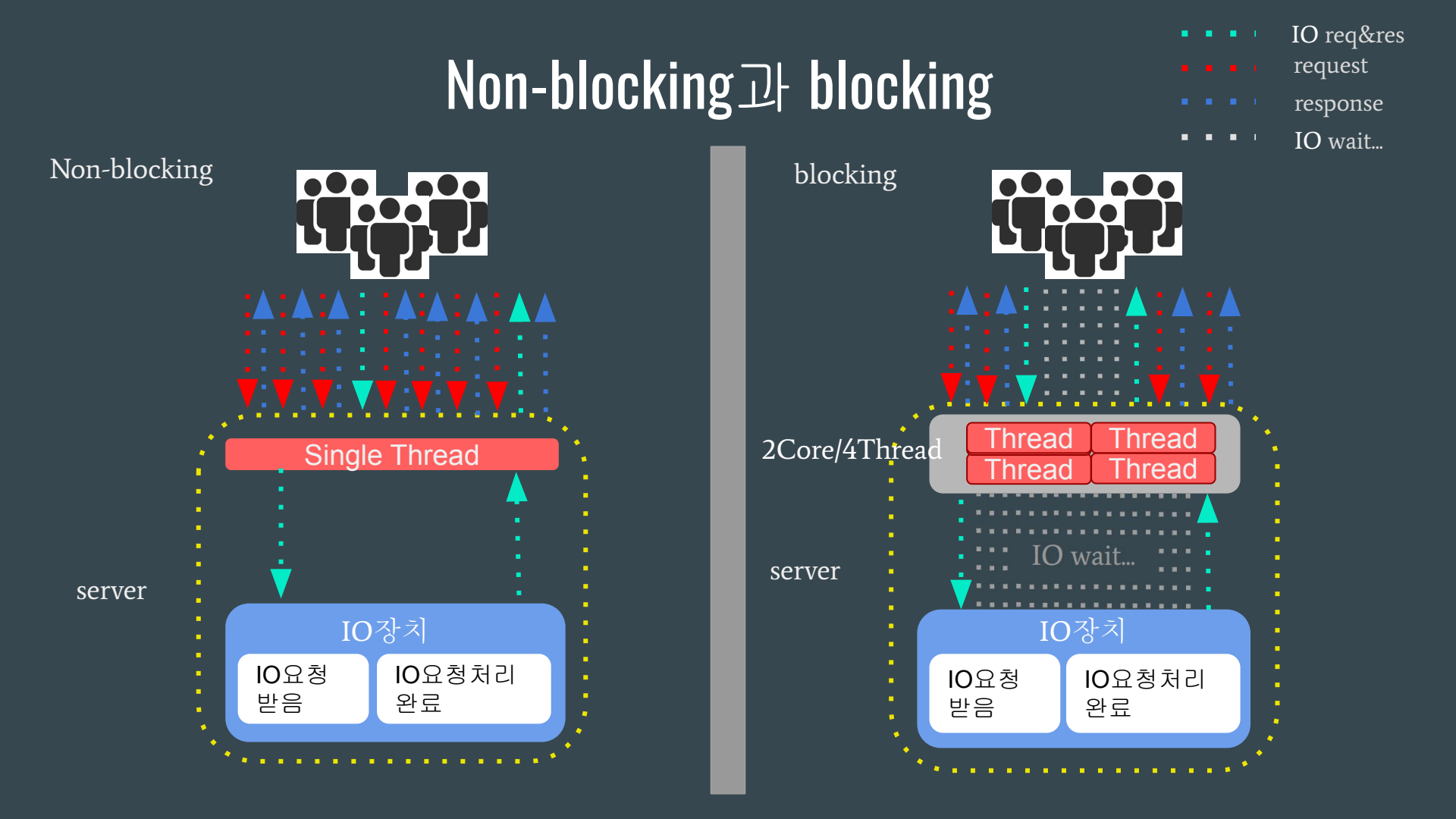


Node.js Application 작동방식



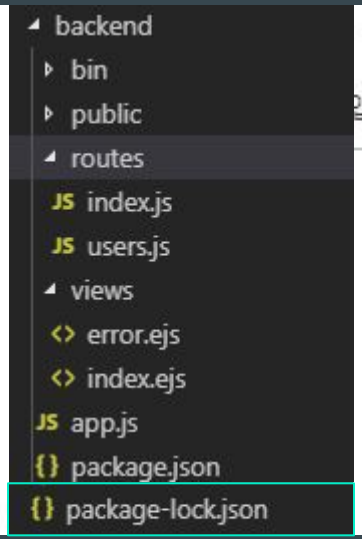
작동 방식 더 자세히 고민

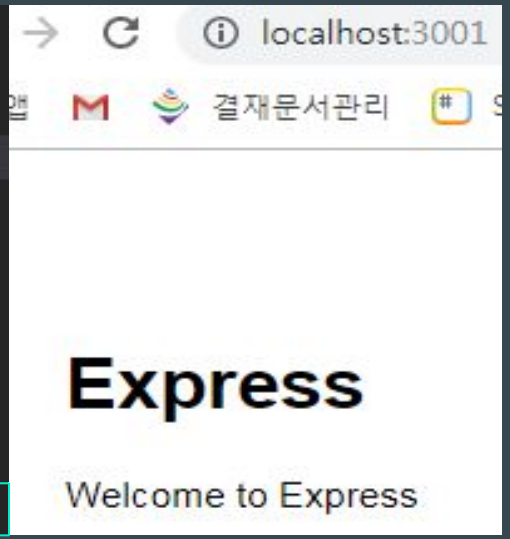
Non-blocking과 blocking



간단한 express 어플리케이션 만들어보기

```
> npm install -g express-generator     # express-generator를 다운안 받았을 경우
> express --view=ejs [프레임워크 파일 생성명]
> npm install                  # package-lock.json 생성
> node bin/www       # 또는 > npm start
http://localhost:3000
```

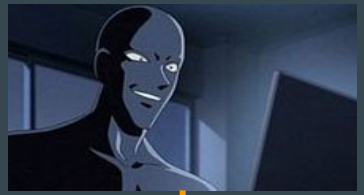




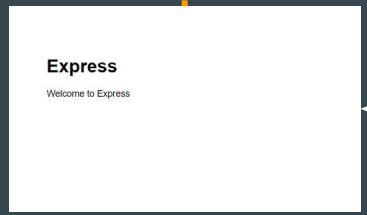
express framework 작동원리

Client 동작
Network

bin/www port : 3000



http://localhost:3000



app.js

```
var indexRouter = require('./routes/index');
var usersRouter = require('./routes/users');
app.use('/', indexRouter);
```

랜더링대상
views/index.ejs

```
<!DOCTYPE html>
<html>
  <head>
    <title><%= title %></title>
    <link rel='stylesheet' href='/stylesheets/style.css' />
  </head>
  <body>
    <h1><%= title %></h1>
    <p>Welcome to <%= title %></p>
  </body>
</html>
```

랜더링로직
routes/index.js

```
var express = require('express');
var router = express.Router();

/* GET home page. */
router.get('/', function(req, res, next) {
  res.render('index', { title: 'Express' });
});

module.exports = router;
```

Good 유즈케이스

- 짧은 시간 작업 + 대량 트래픽 환경에 강함
- 하나의 작업에 많은 시간이 소요되는 프로젝트에는 지양
- 작은 데이터 많은 데이터 처리에 강점... 몽고디비도 이 환경에 특화!
- JSON API
- Ajax가 난무하는 페이지가 단 하나인 앱,
  - 클라이언트의 한 페이지에서 많은 것을 해야하는 웹앱에 좋다.
  - Gmail이 대표적인 페이지가 단 하나인 앱
- 스트리밍
  - 파일 실시간 업로드, node.js는 다양한 데이터 레이어를 위한 프록시를 만들 때 매우 좋다.

Bad 유즈케이스

- 단일 스레드이므로 한 작업에 CPU 자원 사용이 많은 애플리케이션
- 단순 CRUD /HTML 앱
  - Node가 확장성이 좋지만, node.js를 사용했다는 이유로 트래픽 성능이 좋지는 않다.

다음 프로젝트에서 NoSQL DB를 사용하려고 준비중이라면 잠깐 멈추고 이 것부터 읽었으면 좋겠다.

Redis, CouchDB, MongoDB,Riak, Casandra등은 정말 매력적이다. 원래 이브는 빨간 사과를 거절하지 못한다. 지금 node.js를 사용하는 기술적 모험을 감행하고 있다면 더 이상의 모험을 하는 것은 좋지 않다. 아직 완전히 이해하지 못하는 기술들은 서로 위험을 증폭 시킨다.

물론 문서 지향 데이터베이스가 적합한 유즈케이스도 있다. 그러나 지금 만드는 소프트웨어로 사업을 할 것이라면 데이터베이스 기술은 보수적으로 가져가는 것이 좋다. 적어도 덕질*satisfying your inner nerd*보다, 친구들에게 자랑하는 것보다 중요하다.

그럼 우리는? (-_-);;

끗

vue.js 공식참조문서

https://kr.vuejs.org/v2/guide/index.html

유즈케이스참조

http://pismute.github.io/nodeguide.com/convincing_the_boss.html

다음 주제는 ....

Node.js 핵심 모듈

- Buffer
- Event
- Stream
- Process