

Table of Contents

Introduction	1.1
1. 시작하기	1.2
1.1. 웹서비스 아키텍쳐 발전단계 및 유형	1.2.1
1.2. 2018년 웹 개발 기술 동향	1.2.2
1.3. 기술부채	1.2.3
1.4. MEVN 소개	1.2.4
2. 개발환경 구성	1.3
2.1. NodeJS 설치	1.3.1
2.2. Visual Studio Code 설치	1.3.2
2.3. Chrome 브라우저를 기본브라우저로 설정	1.3.3
2.4. MongoDB 설치	1.3.4
3. Prototype 실행	1.4
3.1. 프로젝트 다운로드	1.4.1
3.2. 프로세스 기동	1.4.2
3.2.1. MongoDB 기동	1.4.2.1
3.2.2. MongoDB 접속 테스트	1.4.2.2
3.2.3. WAS 기동	1.4.2.3
3.2.4. Client 기동	1.4.2.4
3.2.5. 브라우저 기동	1.4.2.5
4. Server-Side Prototype	1.5
4.1. Server-Side Prototype 사전지식	1.5.1
4.2. Server-Side Prototype	1.5.2
4.2.1. 프로젝트 열기	1.5.2.1
4.2.2. server.js	1.5.2.2
4.2.3. router 처리	1.5.2.3
5. Client-Side Prototype	1.6
5.1. Client-Side Prototype 사전지식	1.6.1
5.2. Client-Side Prototype	1.6.2
5.2.1. Index.html	1.6.2.1
5.2.2. router 처리	1.6.2.2
5.2.3. Home.vue	1.6.2.3
6. 기능별 예제	1.7
6.1. 로그인 기능	1.7.1
6.2. 게시판 기능	1.7.2
6.3. 사용자 관리 기능	1.7.3

6.4. 다운로드 기능

1.7.4

7. 서버 배포

1.8

웹 개발 Prototype 소개

웹 개발 환경이 빠르게 발전하고 있습니다. 자원을 덜 소모하고, 더 높은 성능을 내면서, 더 쉽게 개발할 수 있는 방향으로 진화하고 있습니다. Native App(OS 기반 application)을 완전히 대체하는 것에 그 목적이 있습니다.

너무 많은 기술들이 쏟아져 나오고 있고, 주류 기술로 자리잡은 기술들만 수십 가지에 이릅니다. 어떤 기술들을 조합해서 개발해야될지 선택하기가 쉽지 않습니다. 레퍼런스가 많고, 문서화가 잘되어 있으며, 학습장벽이 낮은 기술들을 엄선해서 **Prototype**를 만들고, **Prototype** 기반으로 기존에 운영하고 있던 웹서비스 (<https://stocknet.koscom.co.kr>)를 재구축하였습니다. 아직까지는 기술적 이슈가 발견되지 않고 있습니다.

자바기반의 JSP나 PHP로 개발할 것인가, 현재 주류기술로 자리잡고 있는 기술로 개발할 것인가에 대한 고민은 더 이상 무의미해지고 있습니다. 성능상의 비교는 접어두고라도 개발 생산성에서 비교가 되지 않습니다. 웹 개발인력이 부족한 당사의 경우에는 더 더욱 변화된 웹개발 환경에 빨리 적응해야 합니다.

1장에서는 웹서비스 아키텍쳐의 발전단계와 그 유형에 대해서 알아보고, 2018년 웹 기술 동향도 함께 살펴보겠습니다. 기술부채에 대한 고민도 담아보았습니다. 그리고, **Prototype**의 기반기술로 채택한 MEVN(MongoDB + ExpressJS + VueJS + NodeJS)기술스택에 대한 소개를 드립니다.

2장에서는 개발환경 구성에 대해 설명합니다. 개발도구들을 설치하는 방법과 도구들에 대한 간단한 설명을 첨부했습니다.

3장에서는 **Prototype**을 실행하는 방법에 대해 설명합니다. 실행 결과를 먼저 확인하는 것이 **Prototype**을 더 빨리 이해하는데 도움이 될 것입니다.

4장에서는 Server쪽 프로그램에 대한 설명입니다. **NodeJS**를 통해 기본적으로 셋팅하는 공통모듈들에 대한 설명을 소스 기준으로 설명하고, **MongoDB** 연동, 라우팅 처리에 대해 설명합니다.

5장에서는 Client쪽 프로그램에 대한 설명입니다. **VueJS**를 통해 **SPA(Single Page Application)**를 구성하는 방법, 라우팅, **axios** 통신 등에 대해 설명합니다.

4장과 5장은 프로그램에 대한 설명으로 사전지식이 필요합니다. 각 장의 앞부분에 참고도서와 함께 사전지식에 대한 세부항목에 대해 설명해놓았습니다.

6장에서는 기능별로 클라이언트와 서버를 통합해서 설명했습니다. 한 명의 개발자가 기본적으로 클라이언트와 서버를 함께 개발하는 풀스택 개발을 지향합니다.

신규 프로젝트 또는 재구축 프로젝트에 참고가 되었으면 합니다.

1.1. 웹 서비스 아키텍쳐 발전단계 및 유형

이전 아키텍쳐가 이후 아키텍쳐에 의해 반드시 대체되거나 폐기되어야 되는 것은 아닙니다. 용도에 따라 각 유형별 장점이 있기 때문에, 개선된 형태로 보편적으로 사용되고 있습니다.

1.1.1. WEB 서버 Static 페이지 제공

PC/Mobile <> WEB 서버

- WEB 서버에서 미리 준비된 정적인 컨텐츠만 제공
- 고정된 텍스트, 이미지 등
- 사용자 상호작용이 없는 홍보 페이지 등에 유용

1.1.2. WEB 서버 Dynamic 페이지 제공

PC/Mobile <> WEB 서버 <> DB

- WEB 서버에서 동적으로 페이지 생성 (PHP, CGI 등)
- DB 또는 기타 화일시스템과 연동하여 사용자 입력등에 따라 동적인 컨텐츠 제공
- 로그인, 게시판, 화일 업로드/다운로드 등
- *Wordpress* 가 대표적. 블로그 형태의 웹 서비스에 유용

1.1.3. WAS 서버 Dynamic 페이지 제공

PC/Mobile <> WEB 서버 <> WAS 서버 <> DB

- WEB 서버와 WAS 서버의 분리
- WEB 서버는 정적 자원 처리(html, js, image 등)
- WAS 서버는 동적 데이터 처리(JSP, ASP 등)
- JQuery 등 자바스크립트 라이브러리, BootStrap 등 CSS 라이브러리 보편화
- 대량의 트래픽 처리에 효율적인 구조 (WEB 1 : WAS N)
- WEB 서버 저사양, WAS 서버 고사양 등의 배치로 효율적인 자원배분 가능

1.1.4. BACK-END 아키텍쳐 변화

PC/Mobile <> WEB 서버 <> WAS 서버 <> DB

- BACK-END 아키텍처의 변화 : MVC(model-view-control) 모델 적용
- 비즈니스 복잡도 증가로 화면, 데이터, 컨트롤의 분리. 생산성 향상
- JAVA의 스프링 프레임워크가 대표적
- HTML5, CSS3 등 웹 표준 기술 정착. 브라우저 호환성.

1.1.5. FRONT-END 아키텍쳐 변화

PC/Mobile <> WEB 서버 <> WAS 서버 <> DB

- FRONT-END UI framework : MVVM(model - view - view model) 모델 적용
- 화면의 복잡도 증가로 화면에서도 화면, 데이터, 컨트롤의 분리. 생산성 향상
- 페이지 전체를 재수신할 필요가 없어 브라우저의 반응속도가 매우 빨라짐
- FRONT와 BACK의 완전분리. REST API(BACK-END는 데이터만 전송)

- 외부 오픈 API 사용 보편화. facebook 로그인, 구글맵 등
- 자바스크립트의 역할 확대. FRONT 별도 Compile (webpack 등)
- SPA(single page application) : 하나의 페이지에 모든 화면을 담음. component.
- PWA(progressive web application) : Web App이 PC/mobile 지원 사용
- Native App과 차이없는 Web App 지향

1.1.6. Serverless 아키텍쳐

PC/Mobile <> Cloud

- BACK-END 서버를 별도로 운영하지 않고 Cloud 서비스를 이용함
- BasS(Backend as a Service) : Firebase 등
- FaaS(Function as a Service) : AWS Lambda, Azure Functions, Google Cloud Function 등
- 비용, 레퍼런스, Legacy시스템과의 연동 등 고려해서 결정

1.2. 2018년 웹 개발 기술 동향

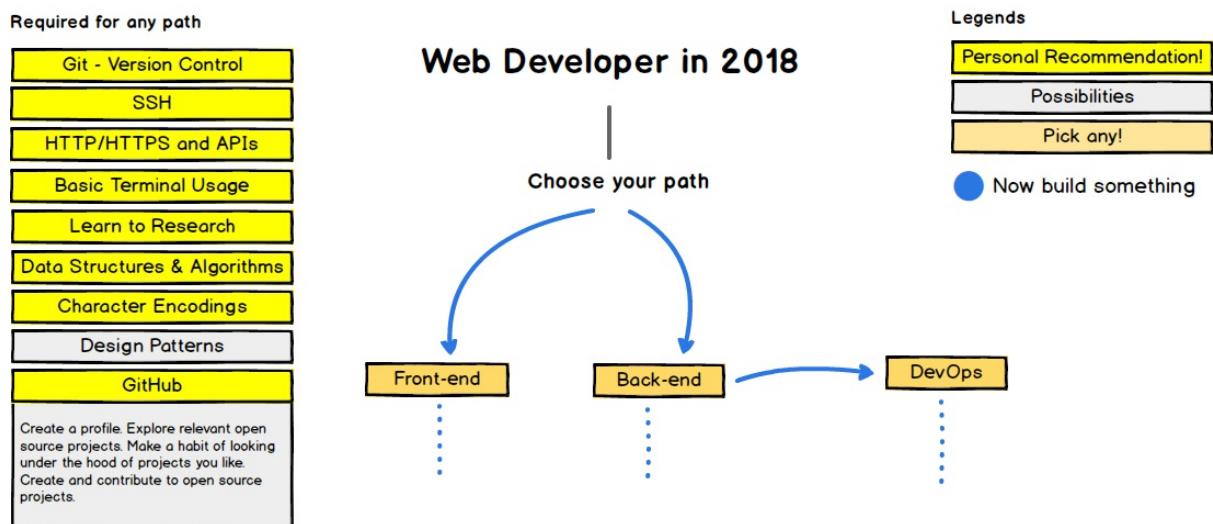


그림 1.2.1. 웹 개발 기반기술 (출처: <https://codeburst.io/the-2018-web-developer-roadmap-826b1b806e8d>)

CodeBurst에서 공유되고 있는 2018년 웹 개발을 위한 기술동향입니다. 많은 웹개발자, 웹교육기관 등에서 매년 공유되는 RoadMap입니다. 모든 기술들을 다 알아야 될 필요는 없습니다. 용도에 맞게 필요한 기술을 선택할 수 있는 안목을 가지는 것이 중요합니다.

기반기술 부분에서 작년에 이어 올해에도 Git이 강조되고 있습니다. 형상 관리, 코드 공유, 문서화 등의 목적으로 GitHub가 보편적으로 사용되고 있음을 알 수 있습니다. Private 레파지토리를 구성해서 웹부분 만이라도 전사 소스 코드를 공유할 필요가 있습니다.

Front-End 부분에서는 자바스크립트와 CSS의 framework 적용이 대세로 자리 잡았음을 알 수 있습니다. Front-End에서 framework를 적용했을 경우 향상되는 생산성의 정도는 이미 검증이 되어 있습니다. 자바스크립트 framework를 쓰게 되면, 개발자들마다 다른 코딩 스타일을 구조적으로 최대한 제어할 수 있고, 가독성 및 효율성이 기본적으로 보장됩니다. Pure 자바스크립트와 JQuery로 짜여진 코드와 framework가 적용된 코드를 비교해보면 직관적으로 framework의 장점을 알 수 있습니다. CSS framework를 쓰게되면, 디자인의 일관성을 쉽게 유지할 수 있고, 가독성 또한 높아집니다.

Back-End 부분에서는 JSP, ASP로 널리 알려져 있던 JAVA와 C#이 Ruby, Python, NodeJS, PHP에 완전히 밀려나 있는 것을 볼 수 있습니다. 주류기술로 자리잡은 기술로 구현할 수 있는 것을 JAVA와 C#이 구현하지 못해서 벌어지는 일이 아닙니다. 아직까지 논란이 많은 이슈이긴 하지만, 구현의 문제가 아니라, 생산성의 문제라 생각합니다. JAVA로 Rest API를, 빅데이터를, AI를 구현하지 못하는 것은 아닙니다. 더 적은 코드, 더 높은 가독성, 더 나은 성능, 타 솔루션과의 호환성 등에 대한 복합적인 문제입니다. 특히나 NodeJS는 모든 부분에서 전혀 부족함이 없으면서, Front-End에서 사용되는 자바스크립트 언어를 Back-End에서 그대로 사용할 수 있다는 장점이 있습니다. 웹 개발자가 부족한 당시 입장에서는 최선의 선택이 아닐까 생각합니다. (빅데이터나 AI 비즈니스로직과 연계된 업무라면 NodeJS 대신 Python-Jango framework를 추천드립니다.)

Prototype에 적용된 기술들을 빨간색 박스로 표시했습니다. 참고하시기 바랍니다.

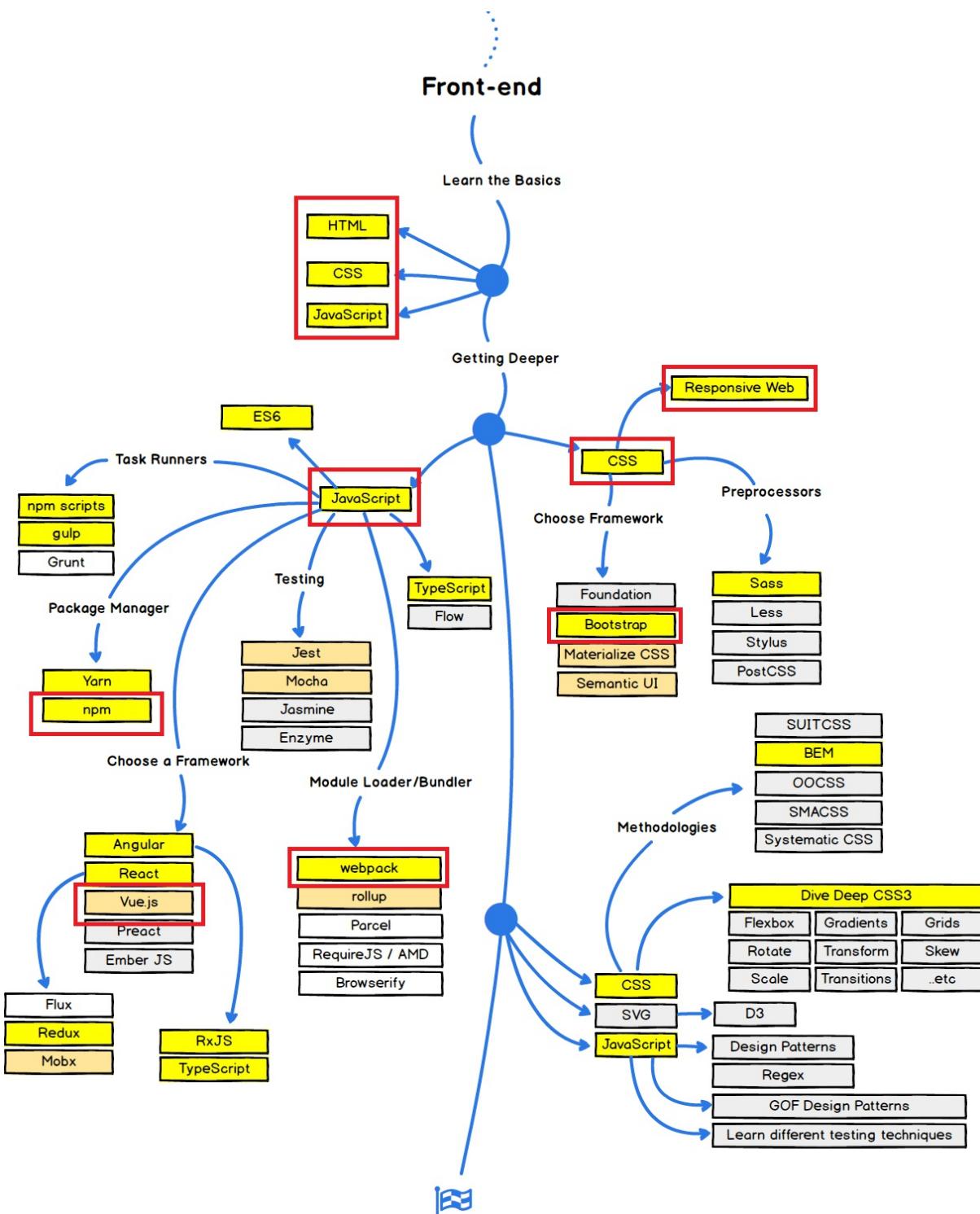


그림 1.2.2. 웹 개발 FRONT-END (출처: <https://codeburst.io/the-2018-web-developer-roadmap-826b1b806e8d>)

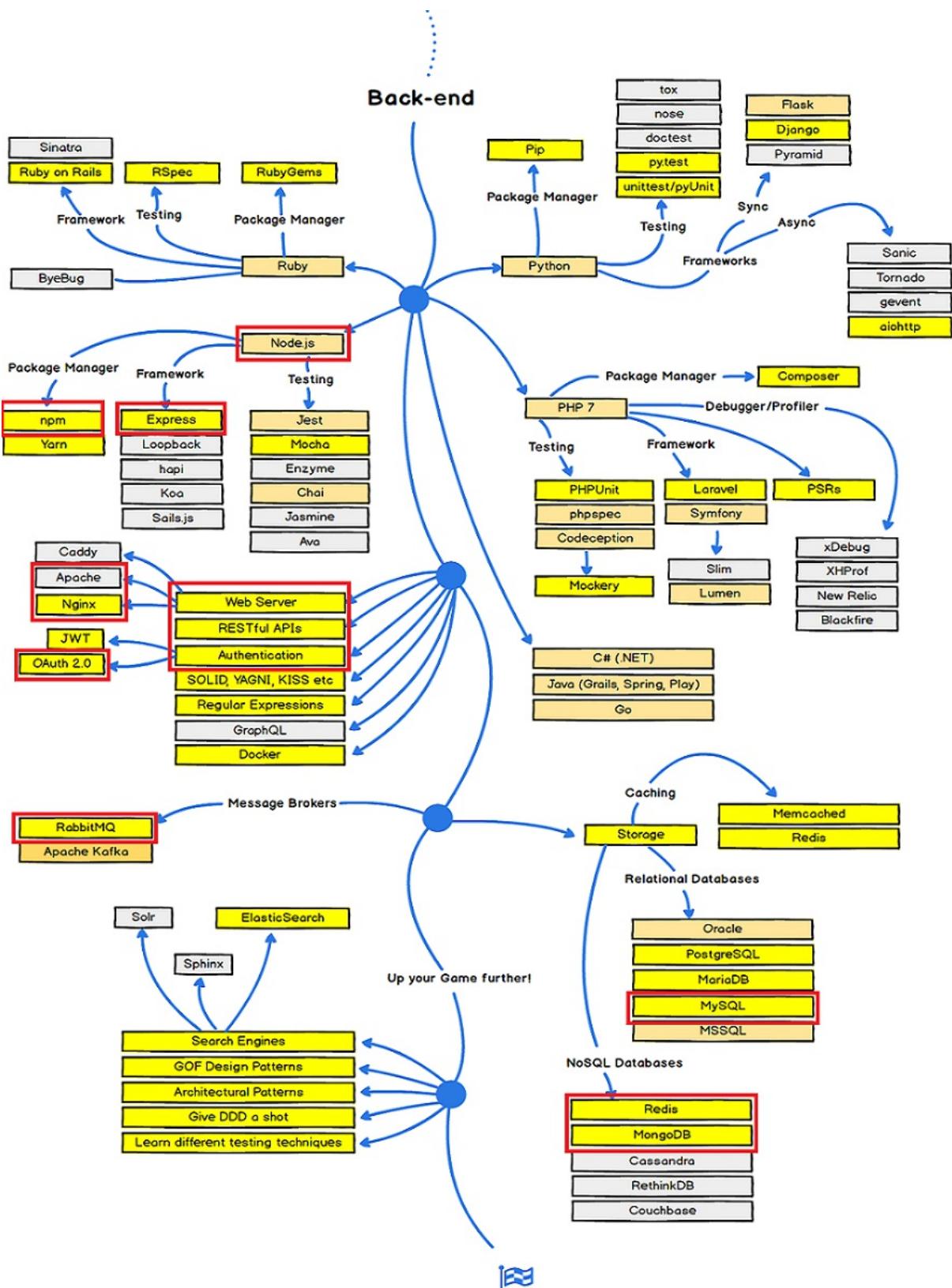


그림 1.2.3. 웹 개발 BACK-END (출처: <https://codeburst.io/the-2018-web-developer-roadmap-826b1b806e8d>)

1.3. 기술 부채

기술적으로 해결해야될 문제를 뒤로 미루고, 비즈니스 문제의 해결 시점을 앞당김으로 인해 쌓여있는 부채

기술부채의 예

- 새로운 OS, 라이브러리, 프레임워크, 미들웨어 등과 더이상 호환이 되지 않음
- 소스 코드의 비효율적 관리로 경쟁사에 비해 생산성이 극도로 저하됨
- 자원소모가 크고, 유지보수료가 비싼 미들웨어나 프레임워크를 사용. 수익구조가 악화됨
- 메인프레임 / UNIX 를 유지. 관리자 및 개발자를 구하지 못함
- OS 패치 및 버전업 방지. 해킹, 장애 등의 사유로 급히 업그레이드. 복구 불능의 장애 발생

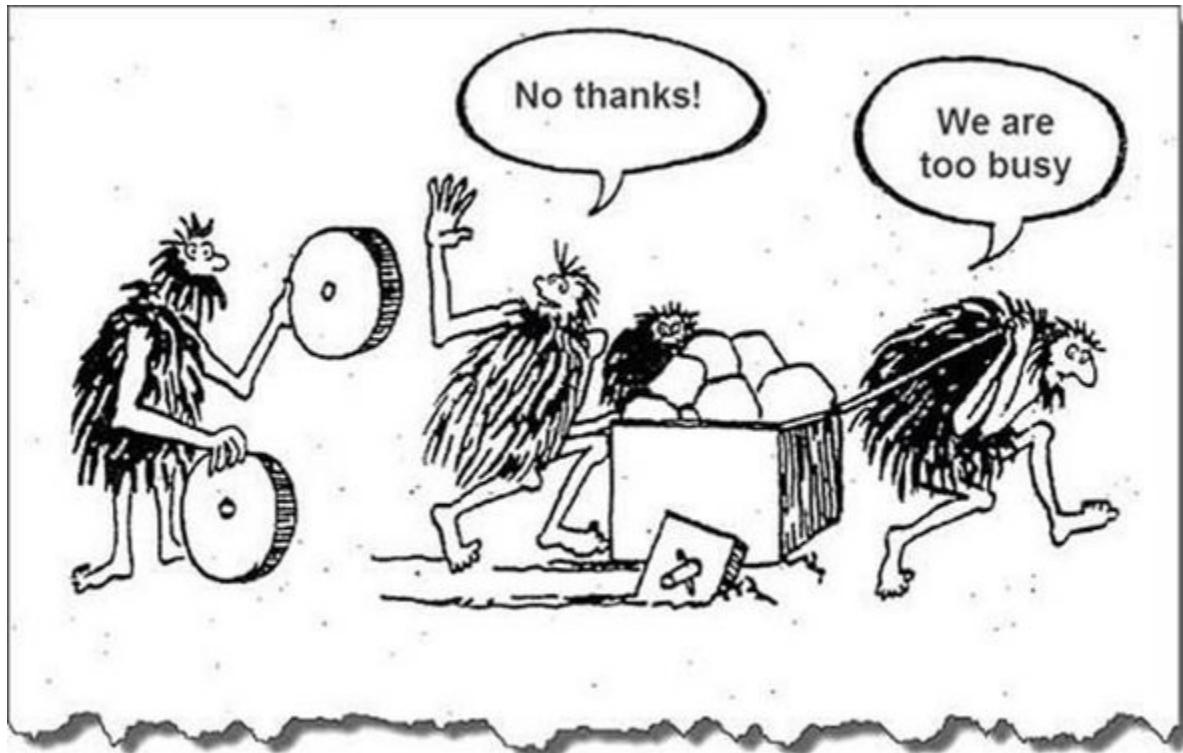


그림 1.3.1 기술부채 (출처: <https://christierney.com/2015/12/04/technical-debt-in-an-image/>)

기술부채 대응

- 주류기술 동향(OS, 라이브러리, 프레임워크, 미들웨어 등)을 주기적으로 파악
legacy 시스템의 생산성, 성능, 효율성 비교 검토
- 프로그램의 경우, 주기적인 리뷰를 통한 리팩토링, 리스트럭처링 시행
- 하드웨어의 경우, EOS(End Of Service)에 적극적으로 대처
- OS 및 미들웨어의 경우, Major Version Number가 최신버전과 20이상 차이가 나지 않도록 관리
ex) RedHat 최신버전이 7이면 최소한 6이상. Java최신 버전이 9이면 최소한 8이상

1.4. MEVN 소개

1.4.1. 어떤 웹 기술을 선택할 것인가

- 메뉴얼, 레퍼런스, 커뮤니티 등 개발 여건이 성숙되어 있어야 함
- OS, 미들웨어, 기타 솔루션에 종속적이지 않지만, 호환성은 높아야 함
- 보편적으로 사용되어 개발자 수급에 문제가 없어야 함
- legacy 시스템의 단점을 보완하여 생산성을 현저하게 향상시킬 수 있어야 함
- 쉽게 배울 수 있고, 빨리 적용할 수 있어야 함

1.4.2. MEVN 소개

MEVN : MongoDB + ExpressJS + VueJS + NodeJS

MongoDB

- NoSQL (Not Only SQL) 데이터베이스 중 2018년 현재 사용빈도 1위
- key-value Pair 형태의 데이터 구조. JSON(javascript object notation)으로 처리하기 적합.
- 자바스크립트 런타임인 NodeJS와 최고의 궁합
- 더 적은 코드로 더 적은 자원을 소모하여 더 빠른 기능 구현

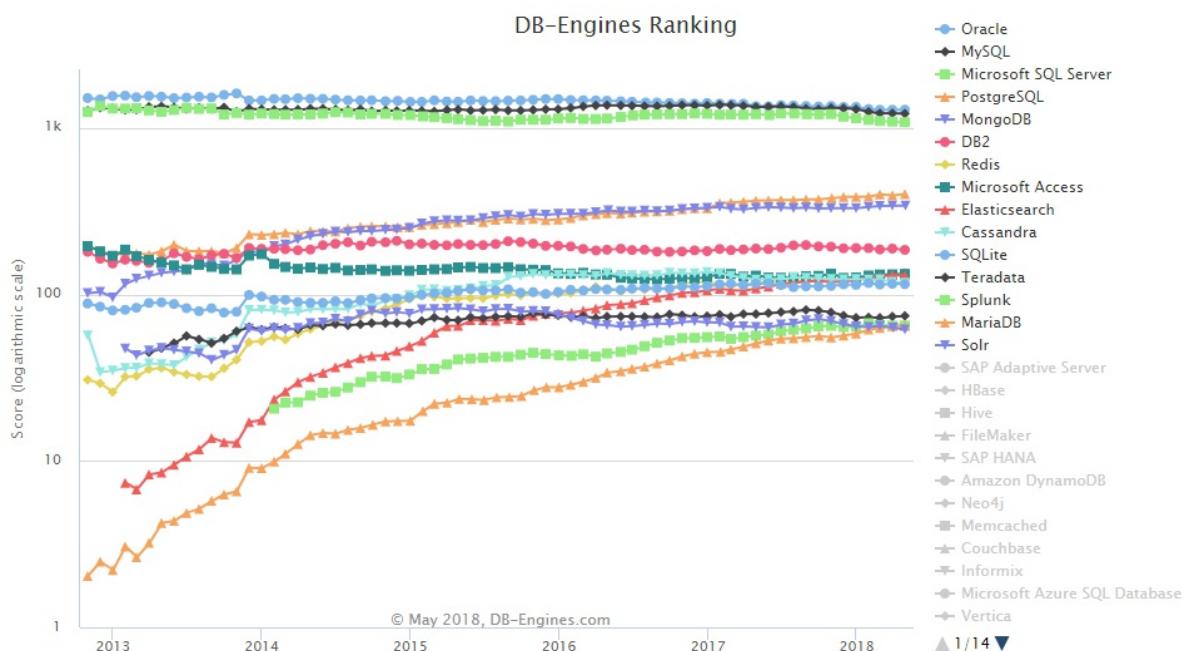


그림 1.3.1 DB Engine Ranking (출처: https://db-engines.com/en/ranking_trend)

ExpressJS

- NodeJS 웹 프레임워크 중 가장 보편적으로 사용되고 있는 프레임워크
- WAS를 NodeJS로 구성한다면 선택의 여지 없음

VueJS

- 자바스크립트 UI 프레임워크
- UI 처리를 위한 자바스크립트(JQuery 포함) 코드를 획기적으로 개선할 수 있음
- ReactJS, AngularJS2 보다 학습곡선이 월등히 낮으면서 각 프레임워크의 장점을 흡수
- 성능, 기능상 타 프레임워크와 큰 차이 없음 적



그림 1.3.2 The State of JavaScript의 2017년 설문 조사 결과 (출처: <https://stateofjs.com/2017/front-end/results/>)

NodeJS

- Chrome V8 Javascript 엔진으로 빌드된 자바스크립트 런타임(실행환경)
- JAVA vs JavaScript / JVM vs NodeJS
- 싱글 스레드기반의 non-blocking I/O처리, Event Driven 방식
- FRONT 와 BACK 을 동일 언어로 개발할 수 있다는 것은 매우 큰 장점

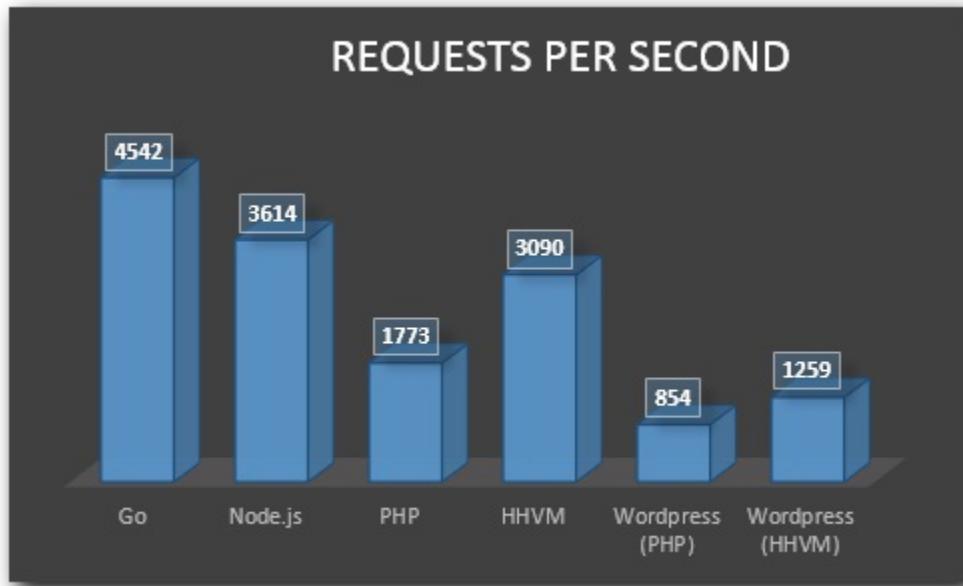


그림 1.3.2 NodeJS 와 PHP 성능비교 (출처: <http://www.hostingadvice.com/blog/comparing-node-js-vs-php-performance/>)

가장 반응속도가 빠르다고 알려져 있는 PHP와 비교한 자료입니다. 성능비교는 논란의 여지가 많습니다. 언어에 최적화된 REQUEST에 따라 결과가 달라지고, CPU의 Core 갯수, 메모리 사용량 등 고려해야 될 변수가 많습니다. 다만, 가장 응답속도가 빠르다고 알려진 PHP보다 오히려 빠른 결과가 나올 수도 있다는 것은 성능상의 이슈가 없다는 것을 증명합니다. Go는 아직 개발 생태계가 충분히 성숙되지 않았지만, 탁월한 성능을 보이고 있습니다. 주목해야 될 언어라고 생각합니다. (*HHVM*은 PHP 가상머신이라고 간단히 생각하시면 됩니다.)

2. 개발환경 구성

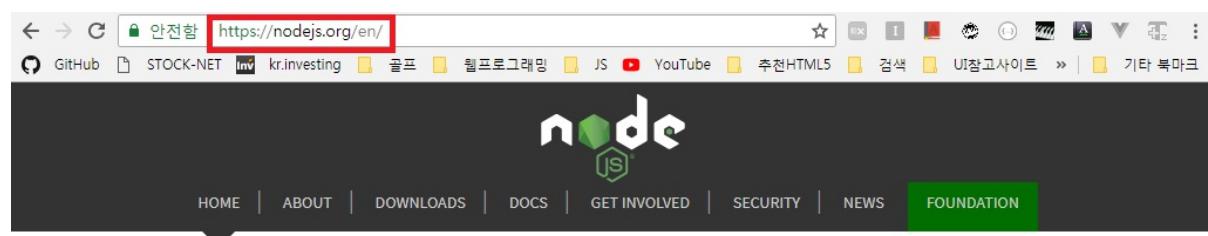
웹 개발 환경을 구성하면 개발의 50%는 끝났다는 이야기가 있습니다. 그 만큼 이전에는 개발환경을 구성하기가 쉽지 않았습니다. 소프트웨어의 버전에 따른 호환성, 운영/개발 환경의 차이 등 고려해야 될 사항이 많았으나, NPM이나 Yarn 같은 패키지 매니저와 VSCdoe같은 IDE에서 사용자 편의 위주로 기능이 많이 개선되어 지금은 그렇게 어렵지 않게 구성할 수 있습니다. 그래도 설치해야 될 내용이 많은 것은 사실입니다.

사용하는 기술에 따라 개발환경 구성이 다르고, 개발자마다 선호하는 개발환경이 다릅니다. 같은 개발팀에서는 개발환경을 통일시킬 필요가 있습니다. 개발환경의 차이 때문에 같은 코드에서 다른 결과가 나올 수도 있고, 코드를 공유하기 힘들어질 수도 있습니다. 도구와 도구들의 버전을 똑같이 맞추시길 권장합니다.

설명에 사용된 개발 PC는 *Window 7 64비트*입니다. 32비트에서는 *mongoDB* 작동에 문제가 생깁니다. *Window 10 64비트*에서도 문제없이 작동합니다. *MacOS* 와 리눅스는 *Window*설치 절차만 참고하시고, 별도로 검색해서 설치하시기 바랍니다.

2.1. NodeJS 설치

NodeJS 홈페이지(<https://nodejs.org>)에서 설치화일을 다운로드 받습니다. 기본옵션으로 실행해서 설치하면 완료됩니다.



Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world.

Node 4.x is End Of Life – April Release Updates

Download for Windows (x64)

8.11.2 LTS
Recommended For Most Users

10.3.0 Current
Latest Features

Other Downloads | Changelog | API Docs Other Downloads | Changelog | API Docs

Or have a look at the [Long Term Support \(LTS\) schedule](#).

Sign up for [Node.js Everywhere](#), the official Node.js Weekly Newsletter.

NodeJS를 설치하면 npm(NodeJS package manager)이 자동으로 설치됩니다. *Windows PowerShell*을 통해 npm을 최신버전으로 업그레이드 합니다. -g 옵션은 Global 설치옵션으로 어느 경로에서나 실행 가능하도록 합니다. NodeJS 관련 라이브러리 뿐만 아니라, 웹 개발에 필요한 다양한 도구들을 npm을 통해 설치할 수 있습니다.

```

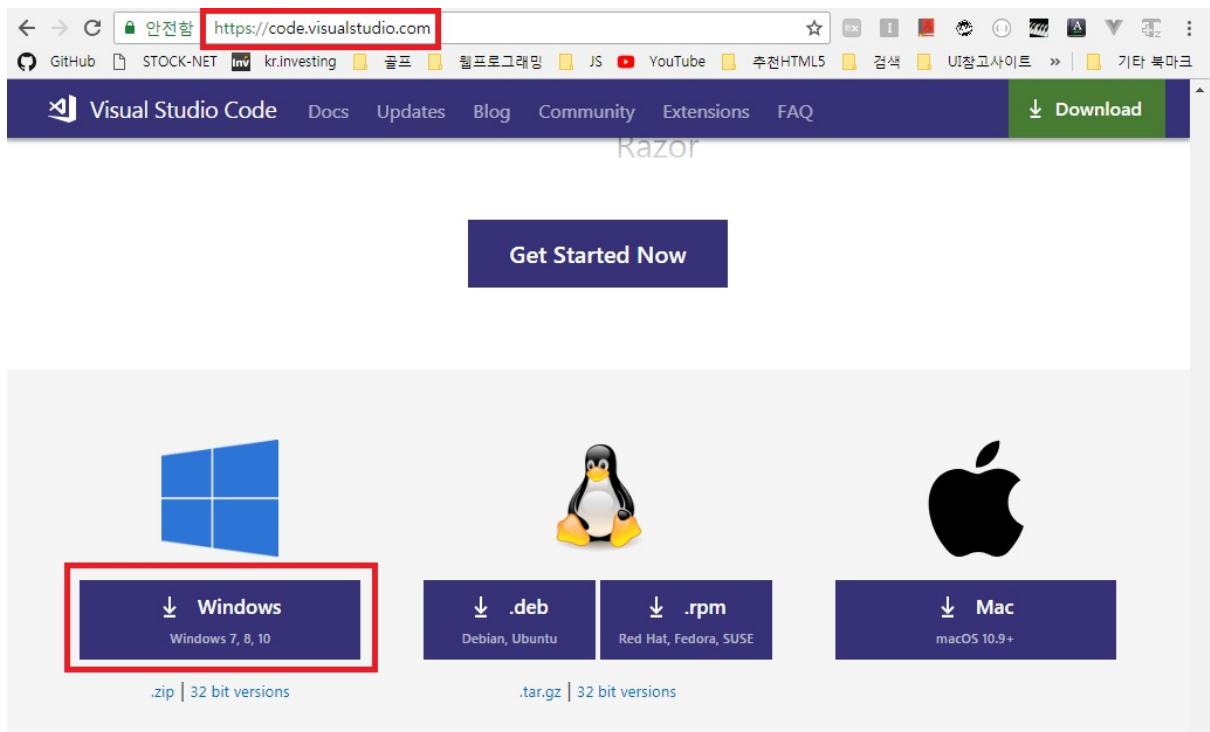
Windows PowerShell
PS C:\workspace>
PS C:\workspace>
PS C:\workspace> npm install -g npm
C:\Users\YONG\AppData\Roaming\npm -> C:\Users\YONG\AppData\Roaming\npm\node_modules\npm\bin\npm-cli.js
C:\Users\YONG\AppData\Roaming\npm -> C:\Users\YONG\AppData\Roaming\npm\npx -> C:\Users\YONG\AppData\Roaming\npm\node_modules\npm\bin\npx-cli.js
+ npm@6.1.0
added 682 packages in 126.783s
PS C:\workspace>

```

2.2. Visual Studio Code 설치

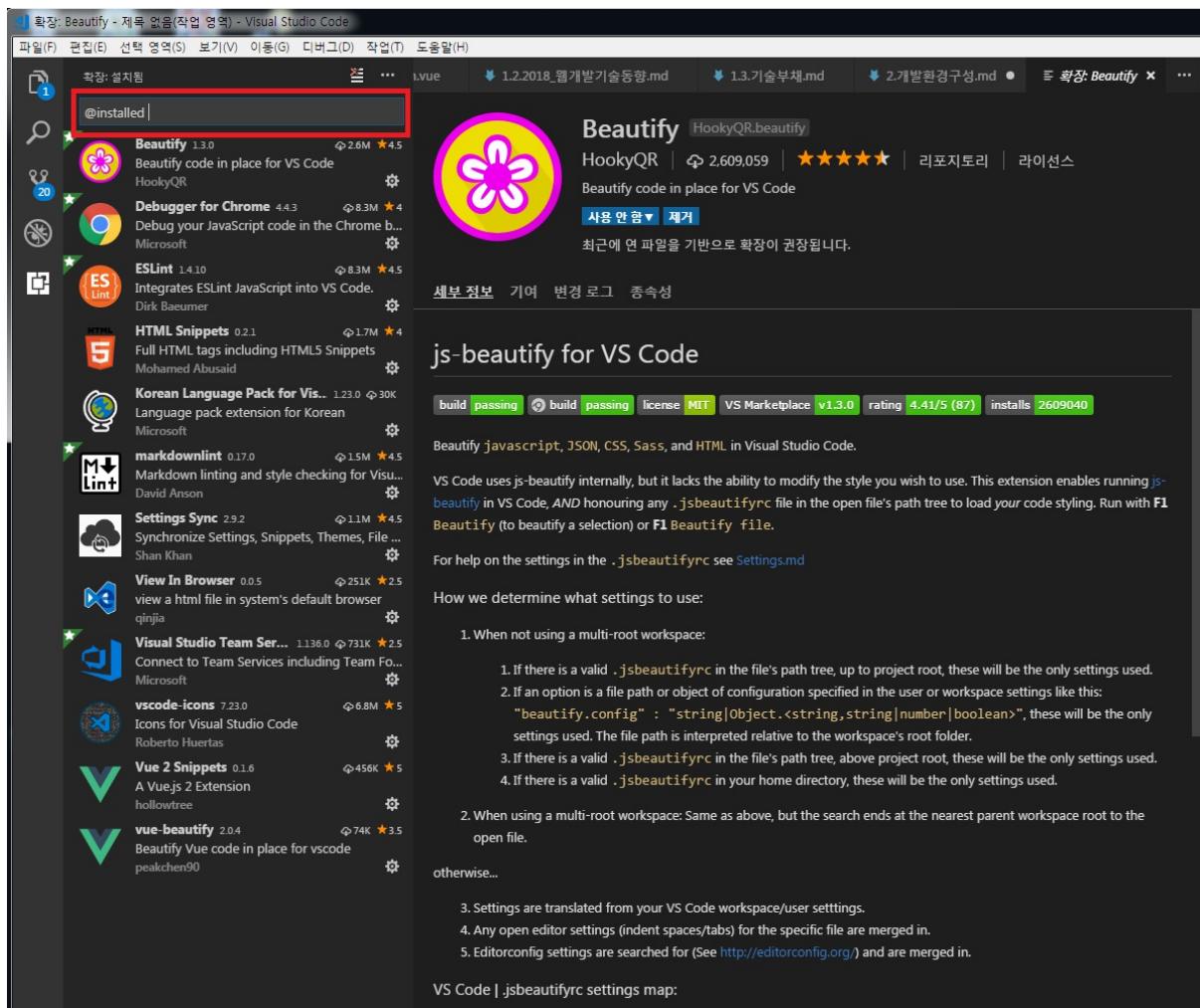
IDE(통합개발환경, Integrated Development Environment)로 Visual Studio Code(이하 VSCode)를 사용합니다. VSCode는 MS에서 제공하는 크로스 플랫폼 에디터로서, NodeJS를 기반으로 만들어졌습니다. 당연히 NodeJS와 궁합이 좋습니다. 선호하는 에디터가 있다면 쓰셔도 되지만, 대부분의 개발자들이 다른 에디터를 쓰다가 VSCode로 바꾸는 것을 보았습니다. 필요한 플러그인들이 빠르게 업데이트 되기 때문인 것 같습니다.

VSCode 홈페이지(<https://code.visualstudio.com>)에서 설치화일을 다운로드 받습니다. 기본옵션으로 실행하고 설치 중 체크박스를 모두 체크하면 완료됩니다.



VScode에서 편집하게 될 파일들은 주로 .js(자바스크립트 파일), .css(스타일시트 파일), html(HTML 파일), .vue(VueJS 파일), .json(설정 파일), .md(마크다운 파일) 등입니다. 각각의 파일들을 쉽게 편집하고, 개발에 필요한 도구들을 위해 플러그인을 설치 합니다.

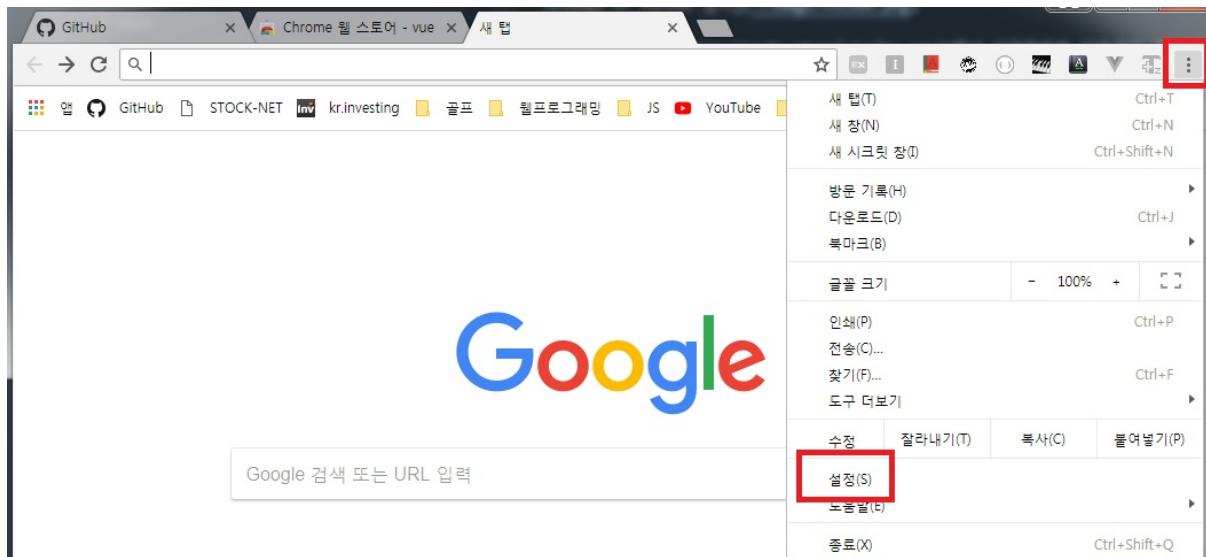
VScode를 실행해서 상단 메뉴의 [보기]-[확장]을 클릭하면 확장메뉴가 활성화됩니다. 좌측 상단 검색창에 각각의 플러그인 이름을 입력해서 검색하면 플러그인들이 표시되고 플러그인의 설치 버튼을 클릭하면 자동으로 설치됩니다. 그림에 보이는 플러그인을 참고해서 설치하시기 바랍니다. 각각의 플러그인에 대한 설명은 VSCode에서 확인하시기 바랍니다. 개발 생산성을 높일 수 있는 플러그인들이 자주 업데이트 됩니다. 구글에서 vscode 확장 플러그인이라고 검색하시면, 유용한 플러그인들을 소개한 좋은 블로그들이 많이 있습니다. 참고하시기 바랍니다.



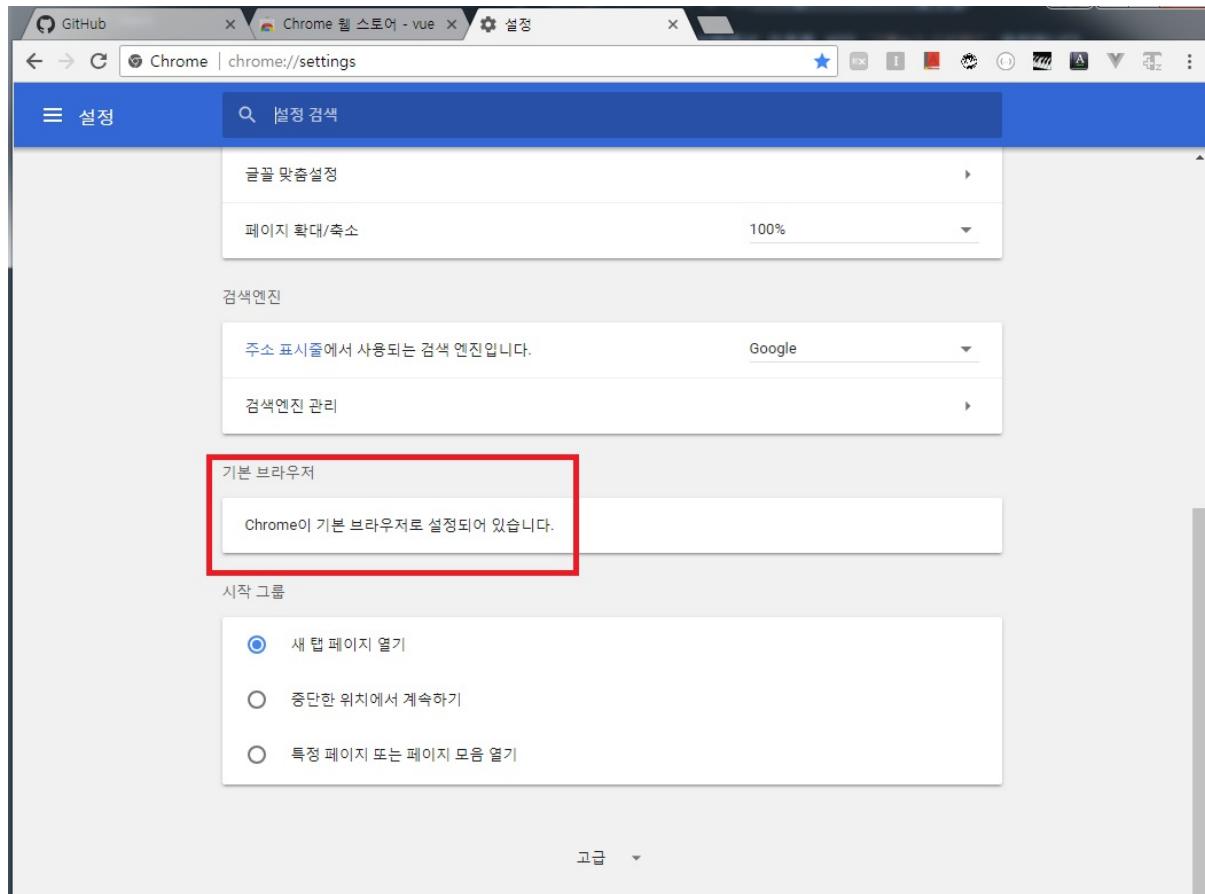
2.3. Chrome 브라우저를 기본브라우저로 설정

Chrome 브라우저가 없다면 설치해주시기 바랍니다. VueJS 디버깅이 타 브라우저에 비해서 편리합니다. Chrome 브라우저를 기본 브라우저로 설정합니다.

Chrome 브라우저를 실행해서 오른쪽 상단 [메뉴]-[설정] 클립합니다.

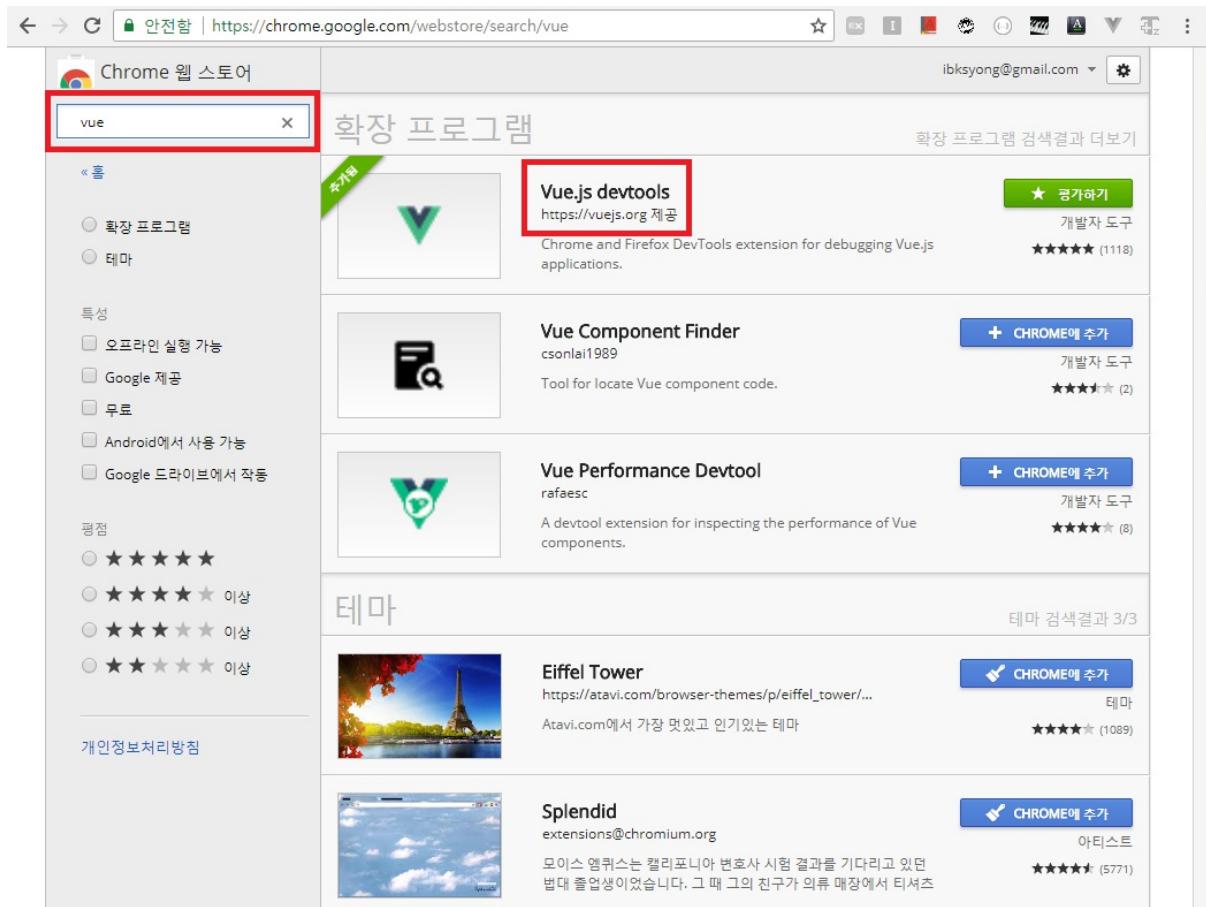


Chrome 브라우저가 기본 브라우저로 설정되어 있는 것을 확인합니다.



VueJS 디버깅을 위한 브라우저 확장 플러그인을 설치합니다. 크롬 웹 스토어(
<https://chrome.google.com/webstore>)에 들어갑니다. 구글 검색창에 '크롬 웹 스토어'라고 검색해서 찾아가셔도 됩니다.

왼쪽 상단에 'vue'로 검색해서 VueJS 디버깅 툴인 'vue.js devtools'를 Chrome에 추가합니다.

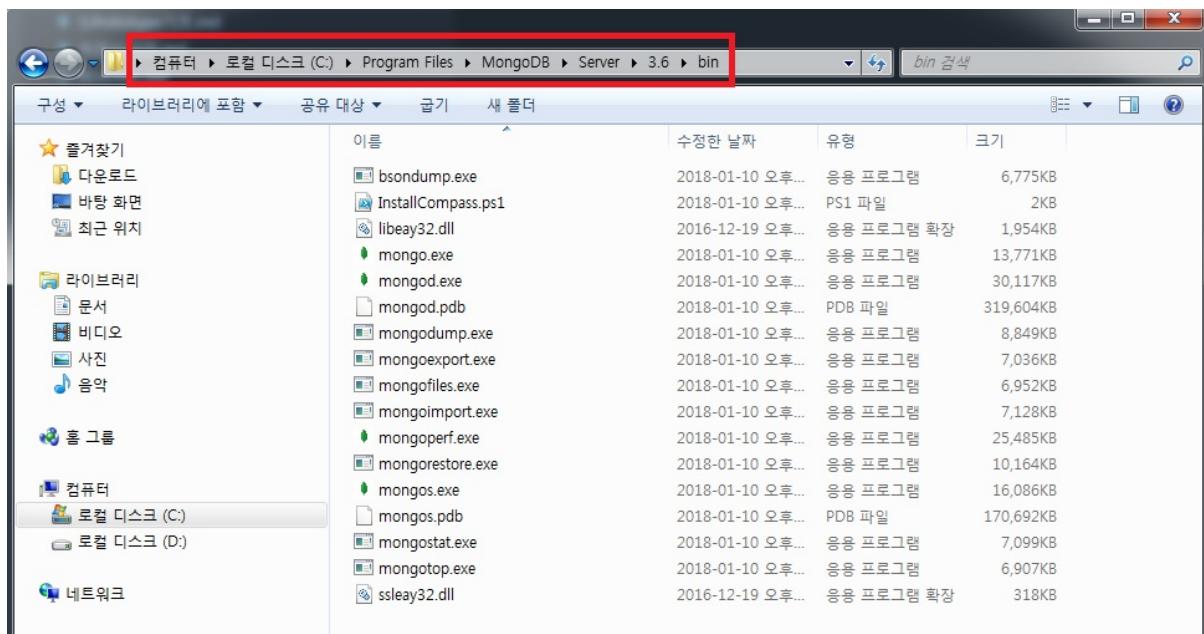


2.4. MongoDB 설치

MongoDB 홈페이지에 접속합니다. Community Server 메뉴를 선택해서, Windows 버전을 다운로드 합니다. 64비트 버전 하나만 선택할 수 있습니다. 32비트 및 기타 다른 버전도 다운로드 받을 수 있으나, 권장하지 않습니다. 32비트를 사용하게 되면, Robo 3T같은 클라이언트 프로그램을 사용할 수 없고, 데이터 베이스의 용량도 2GB 이하로 제한됩니다. 기본옵션 그대로 설치합니다. Setup-Type을 Complete로 선택합니다.

The screenshot shows the MongoDB Download Center page. At the top, there are navigation links for DOCS, LEARN, WHAT'S MONGODB?, and LOGIN. On the right, there is a search bar and a 'Get MongoDB' button. The main header says 'MongoDB Download Center'. Below the header, there are tabs for Atlas, Community Server (which is highlighted with a red box), Enterprise Server, Ops Manager, Compass, and Connector for BI. Under the 'Community Server' tab, it says 'Current Stable Release (3.6.5)'. There are download links for Windows, Linux, and OSX. A dropdown menu for 'Version:' shows 'Windows Server 2008 R2 64-bit and later, with SSL support x64' (also highlighted with a red box). Below that, there is a 'Installation Package:' section with a 'DOWNLOAD (msi)' button (also highlighted with a red box). At the bottom, there are links for 'Binary', 'Installation Instructions', and 'All Version Binaries'.

다음과 같은 경로에 설치된 것을 확인하실 수 있습니다.



Windows PowerShell을 통해 데몬 프로그램을 실행합니다. 설치된 경로의 ./bin 폴더로 이동해서 mongod를 실행합니다.

```

PS C:\> cd 'C:\Program Files\MongoDB\Server\3.6\bin'
PS C:\Program Files\MongoDB\Server\3.6\bin> mongod
2018-05-31T13:39:20.517+0900 I CONTROL [initandlisten] MongoDB starting : pid=11360 port=27017 dbpath=C:\data\db\ 64-bit host=P20845
2018-05-31T13:39:20.517+0900 I CONTROL [initandlisten] targetMinOS: Windows 7/Windows Server 2008 R2
2018-05-31T13:39:20.518+0900 I CONTROL [initandlisten] db version v3.6.2
2018-05-31T13:39:20.519+0900 I CONTROL [initandlisten] git version: 489d177dbd0f0420a8ca04d39fd78d0a2c539420
2018-05-31T13:39:20.519+0900 I CONTROL [initandlisten] OpenSSL version: OpenSSL 1.0.2u-fips 22 Sep 2016
2018-05-31T13:39:20.521+0900 I CONTROL [initandlisten] allocator: tcmalloc
2018-05-31T13:39:20.521+0900 I CONTROL [initandlisten] modules: none
2018-05-31T13:39:20.521+0900 I CONTROL [initandlisten] build environment:
2018-05-31T13:39:20.522+0900 I CONTROL [initandlisten] distmod: 2008plus-ssl
2018-05-31T13:39:20.523+0900 I CONTROL [initandlisten] distarch: x86_64
2018-05-31T13:39:20.524+0900 I CONTROL [initandlisten] target_arch: x86_64
2018-05-31T13:39:20.524+0900 I CONTROL [initandlisten] options: <>
2018-05-31T13:39:20.689+0900 I - [initandlisten] Detected data files in C:\data\db\ created by the 'wiredTiger' s

```

아래와 같이 디플트 값으로 프로그램이 실행됩니다.

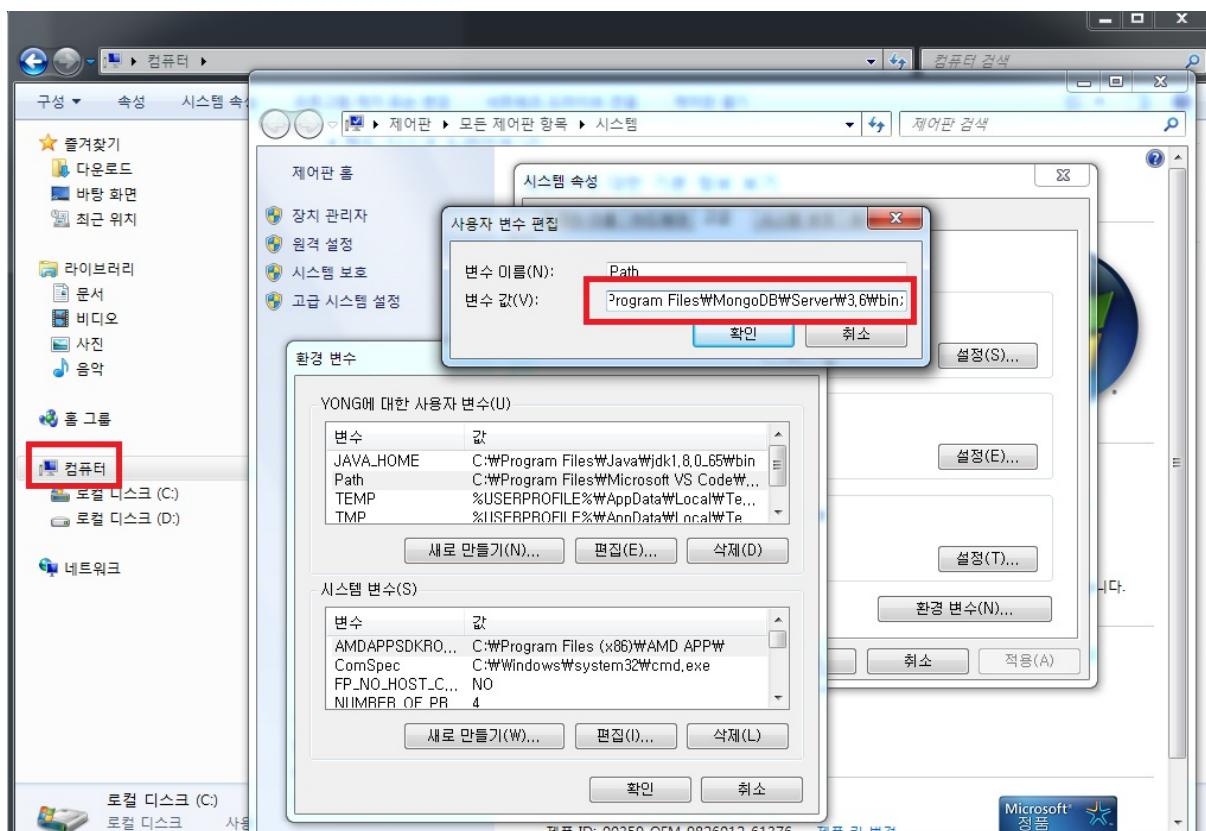
```

PS C:\> which IP
2018-05-31T13:39:22.333+0900 I CONTROL [initandlisten] ** addresses it should serve responses from, or with --
bind_ip_all to
2018-05-31T13:39:22.335+0900 I CONTROL [initandlisten] ** bind to all interfaces. If this behavior is desired,
start the
2018-05-31T13:39:22.335+0900 I CONTROL [initandlisten] ** server with --bind_ip 127.0.0.1 to disable this warning.
2018-05-31T13:39:22.337+0900 I CONTROL [initandlisten]
2018-05-31T13:39:22.338+0900 I CONTROL [initandlisten] Hotfix KB2731284 or later update is not installed, will zero-out
data files.
2018-05-31T13:39:22.338+0900 I CONTROL [initandlisten]
2018-05-31T13:39:22.339+0900 I CONTROL [initandlisten] ** WARNING: The file system cache of this machine is configured
to be greater than 40% of the total memory. This can lead to increased memory pressure and poor performance.
2018-05-31T13:39:22.340+0900 I CONTROL [initandlisten] See http://dochub.mongodb.org/core/wt-windows-system-file-cache
2018-05-31T13:39:22.342+0900 I CONTROL [initandlisten]
2018-05-31T13:39:24.207+0900 I FTDC [initandlisten] Initializing full-time diagnostic data capture with directory 'C:\data\db\diagnostic.data'
2018-05-31T13:39:24.216+0900 I NETWORK [initandlisten] waiting for connections on port 27017

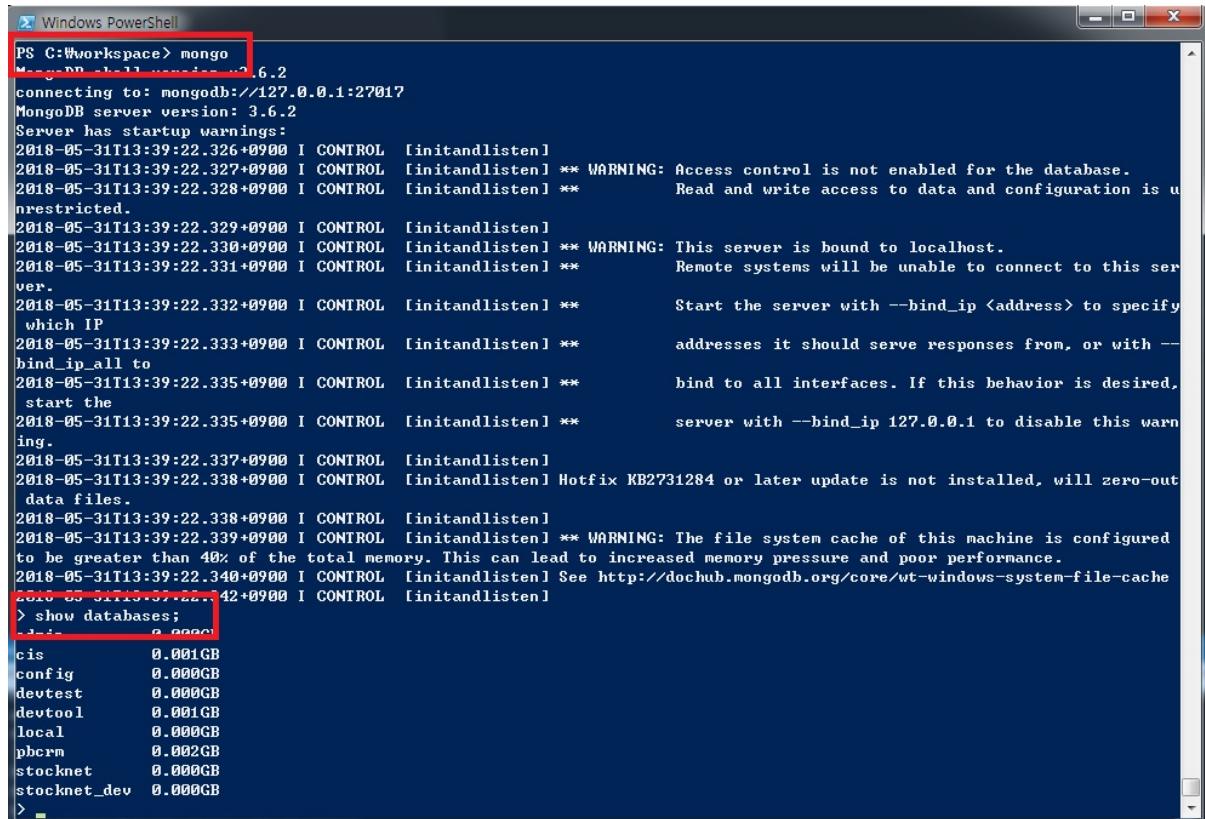
```

별도로 설정하지 않으면 데이터베이스는 C:\data\db에 생성됩니다. 포트는 27017을 사용합니다.

실행할 때마다 설치 경로로 이동하는 것이 불편할 수 있습니다. PATH에 실행화일의 경로(;C:\Program Files\MongoDB\Server\3.6\bin)를 추가 합니다. [탐색기]-[컴퓨터]-[속성]-[고급시스템설정]-[환경변수]-[PATH]-편집 입니다.



데온 프로그램을 가동시킨 상태에서 클라이언트 프로그램을 실행시켜 설치테스트를 진행합니다. PATH를 지정하셨으면, 어느 경로에서건 실행이 됩니다. `mongo` 라고 입력하면, 클라이언트 프로그램이 실행됩니다. `show databases` 라고 입력하면 디폴트로 설치된 database가 표시됩니다.



The screenshot shows a Windows PowerShell window titled "Windows PowerShell". The command `PS C:\workspace> mongo` is entered at the prompt, followed by the output of the MongoDB shell. The output includes startup warnings about access control, bind IP, and system file cache configuration. The command `> show databases;` is then entered, and the list of databases is displayed:

```
PS C:\workspace> mongo
MongoDB shell version: 3.6.2
connecting to: mongodb://127.0.0.1:27017
MongoDB server version: 3.6.2
Server has startup warnings:
2018-05-31T13:39:22.326+0900 I CONTROL [initandlisten]
2018-05-31T13:39:22.327+0900 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2018-05-31T13:39:22.328+0900 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.
2018-05-31T13:39:22.329+0900 I CONTROL [initandlisten]
2018-05-31T13:39:22.330+0900 I CONTROL [initandlisten] ** WARNING: This server is bound to localhost.
2018-05-31T13:39:22.331+0900 I CONTROL [initandlisten] ** Remote systems will be unable to connect to this server.
2018-05-31T13:39:22.332+0900 I CONTROL [initandlisten] ** Start the server with --bind_ip <address> to specify which IP
2018-05-31T13:39:22.333+0900 I CONTROL [initandlisten] ** addresses it should serve responses from, or with --bind_ip_all to
2018-05-31T13:39:22.335+0900 I CONTROL [initandlisten] ** start the
2018-05-31T13:39:22.336+0900 I CONTROL [initandlisten] ** server with --bind_ip 127.0.0.1 to disable this warning.
2018-05-31T13:39:22.337+0900 I CONTROL [initandlisten]
2018-05-31T13:39:22.338+0900 I CONTROL [initandlisten] Hotfix KB2731284 or later update is not installed, will zero-out data files.
2018-05-31T13:39:22.339+0900 I CONTROL [initandlisten]
2018-05-31T13:39:22.340+0900 I CONTROL [initandlisten] ** WARNING: The file system cache of this machine is configured to be greater than 40% of the total memory. This can lead to increased memory pressure and poor performance.
2018-05-31T13:39:22.340+0900 I CONTROL [initandlisten] See http://dochub.mongodb.org/core/wt-windows-system-file-cache
> show databases;
{
  "_id": "admin"
}
{
  "_id": "config"
}
{
  "_id": "local"
}
{
  "_id": "pbcrn"
}
{
  "_id": "stocknet"
}
{
  "_id": "stocknet_dev"
}
> _
```

간단한 데이터 확인 등의 작업은 `mongo` 로 하면 편리하지만, 대량의 데이터 확인이나 입력 등의 작업은 불편할 수 있습니다. 당연히 GUI 클라이언트 프로그램이 있습니다. 로보몽고를 설치하도록 하겠습니다. 로보몽고 홈페이지(<https://robomongo.org/download>)에 접속합니다.

The most powerful option
Need a more powerful GUI?
Meet Robo 3T's sibling, **Studio 3T**

- Enjoy rich query autocomplete
- Build queries via drag-and-drop
- Write SQL to query MongoDB
- Break down aggregation queries into stages
- Generate driver code in five languages
- Compare collections and view differences side-by-side
- Explore data schema and find outliers
- Import and export in various formats (CSV, JSON, SQL)
- Secure authentication with LDAP and Kerberos

[Download Studio 3T](#)

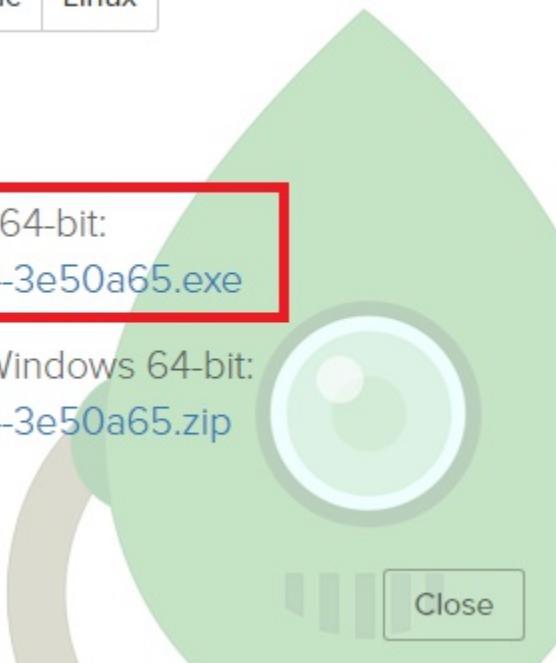
Download Robo 3T

Windows Mac Linux

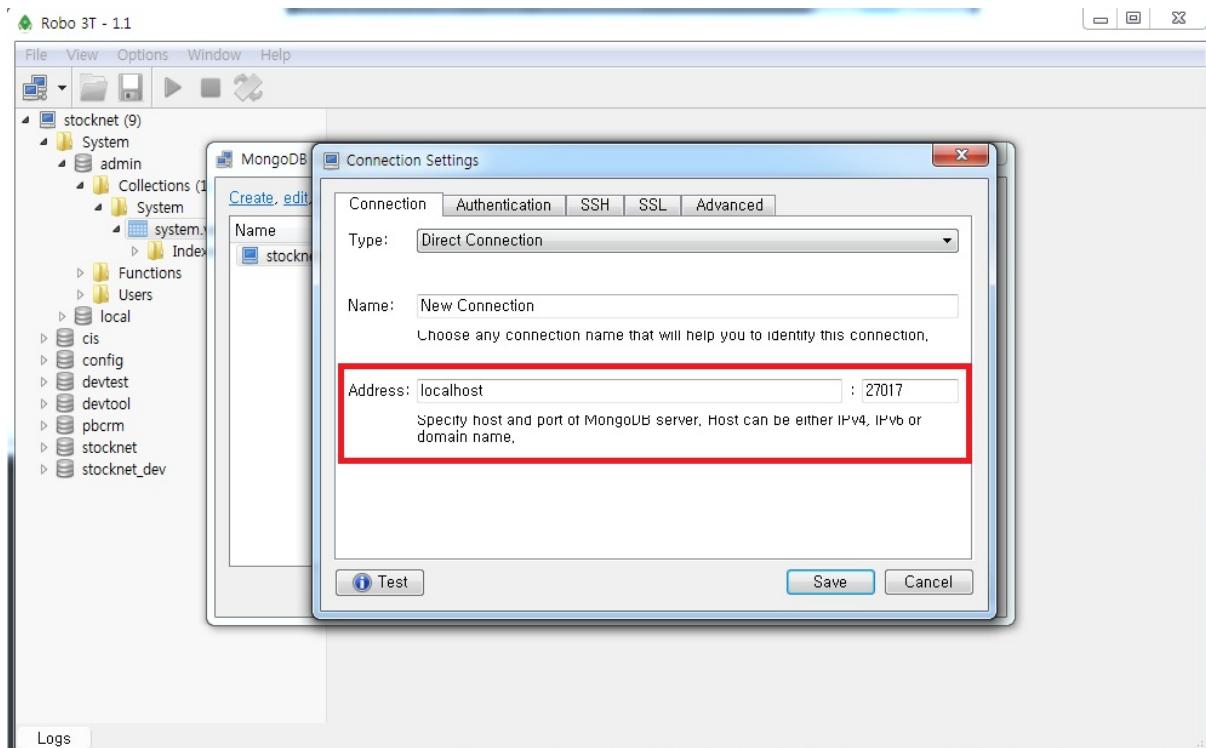
Robo 3T 1.2

Download installer for Windows 64-bit:
 [robo3t-1.2.1-windows-x86_64-3e50a65.exe](#)

Download portable version for Windows 64-bit:
 [robo3t-1.2.1-windows-x86_64-3e50a65.zip](#)



Robo 3T를 실행합니다. [FILE]-[CONNECT]-[CREATE]에서 주소와 포트를 확인합니다. *localhost, 27017*을 입력하면 됩니다. 저장하고 접속하면 디폴트로 설치된 database 목록이 표시됩니다.



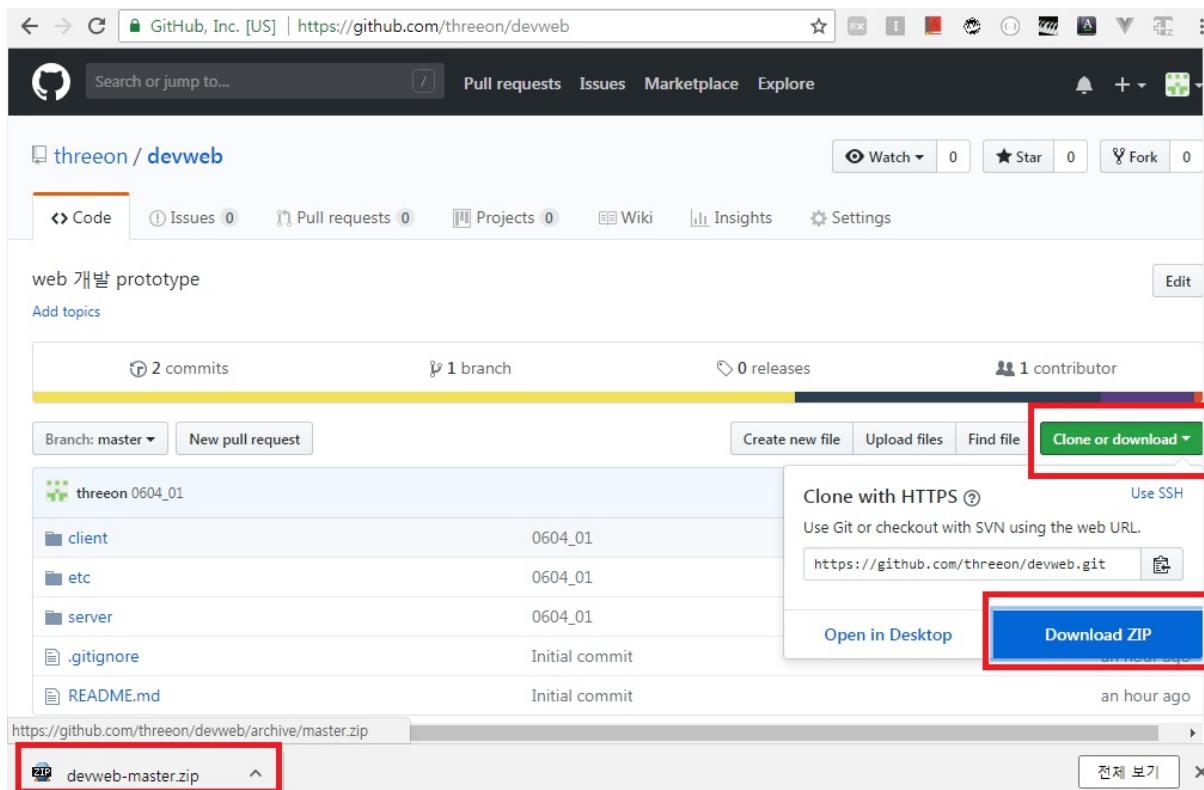
이것으로 개발에 필요한 프로그램의 설치를 모두 마쳤습니다. 필요한 패키지나 라이브러리 등은 `prototype`을 설명하면서 설치하도록 하겠습니다. 서버에 배포할 때는 서버에 맞는 NodeJS, MongoDB를 따로 설치해야 됩니다. 버전만 맞추면 개발환경과 동일하게 작동합니다. 서버 배포를 설명하는 부분에서 설명하겠습니다.

3. Prototype 실행

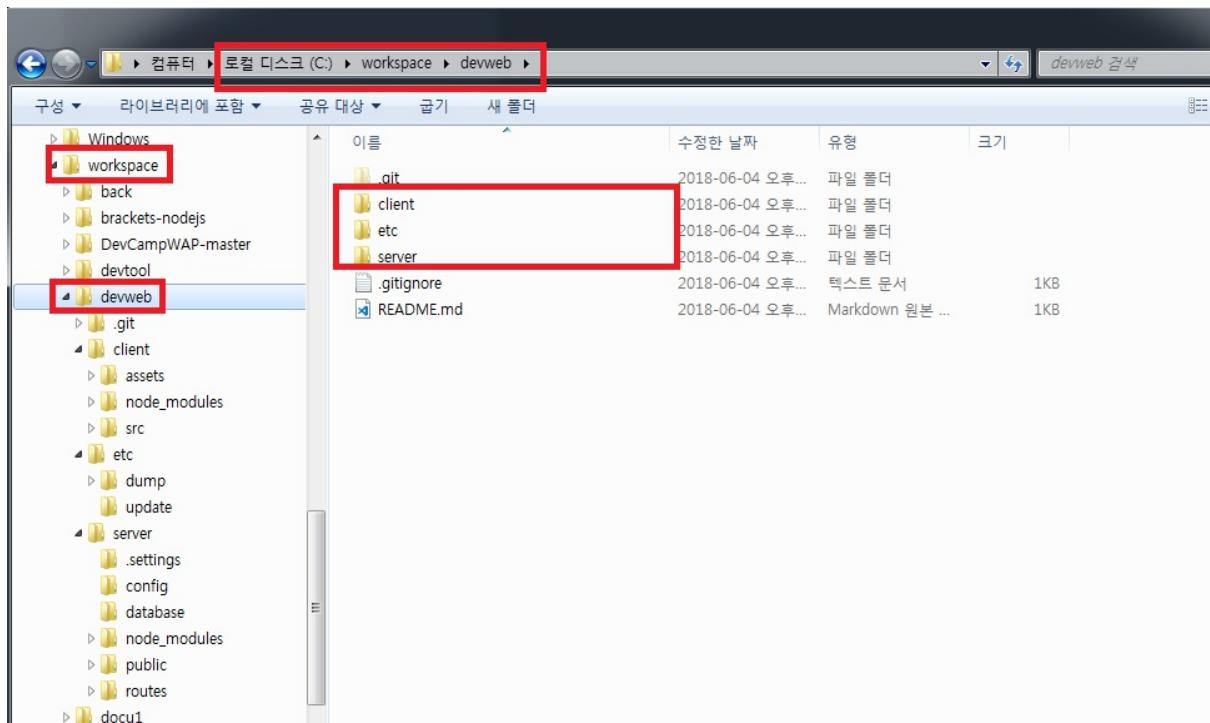
Prototype 프로젝트를 직접 다운로드하여 설치하고, 실행해보도록 하겠습니다. 필요한 패키지를 처음부터 하나하나 구성하는 시간을 절약할 수 있습니다. 웹서비스에 필요한 공통 부분이 대부분 포함되어 있기 때문에 신규 개발 업무를 진행할 때에도 Prototype 프로젝트를 설치해서 수정해나가는 방법을 사용하시면 편리합니다. 필요한 패키지를 추가하는 방법, 필요없는 패키지를 삭제하는 방법은 따로 설명드리도록 하겠습니다.

3.1. 프로젝트 다운로드

GitHub에 있는 레파지토리(<https://github.com/threeon/devweb>)에서 프로젝트를 다운로드 받습니다. **devweb-master.zip**이라는 파일 이름으로 다운로드 받게 됩니다.



C:\workspace이라는 경로에 복사하여 압축을 풀고 프로젝트의 이름을 **devweb**으로 수정하도록 하겠습니다.
.client에는 FRONT-END 관련 소스(VueJS), ./server에는 BACK-END 관련 소스(NodeJS), ./etc에는 문서화일이나 DB 덤프화일등 소스 이외의 파일들이 포함되게 됩니다.



Windows PowerShell을 통해 `./client` 와 `./server` 에 필요한 라이브러리들을 다운로드 받습니다. 처음부터 라이브러리를 `prototype`에 포함시켜 배포할 수도 있으나, 용량이 150MB 가까이 되기 때문에 프로젝트 배포시 다운로드 시간이 지연되는 불편함이 있습니다. 그래서, 보통 프로젝트를 배포할 때 라이브러리는 제외하고 배포해서 로컬에서 별도로 다운로드 받는 절차를 거치게 됩니다.

```
PS C:\workspace\devweb\client> npm install
npm WARN      SKIPPING OPTIONAL DEPENDENCY: fsevents@1.1.1 <node_modules\quill-image-resize-module\node_modules\fsevents>:
npm WARN      SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.1.1: wanted {"os":"darwin","arch":"any"} <current: {"os":"win32","arch":"x64"}>
npm WARN      SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.4 <node_modules\fsevents>:
npm WARN      SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.4: wanted {"os":"darwin","arch":"any"} <current: {"os":"win32","arch":"x64"}>

added 1337 packages from 657 contributors and audited 9425 packages in 239.285s
found 0 vulnerabilities

PS C:\workspace\devweb\client>
```

`npm WARN` 내용은 무시하셔도 됩니다.

```
PS C:\workspace\devweb\server> npm install
added 149 packages from 116 contributors and audited 317 packages in 27.114s
found 0 vulnerabilities

PS C:\workspace\devweb\server>
```

3.2. 프로세스 기동

3.2.1. MongoDB 기동

```
> mongod
```

```

PS C:\Users\WYONG> mongod
2018-07-18T09:44:05.127+0900 I CONTROL [initandlisten] MongoDB starting : pid=11064 port=27017 dbpath=C:\data\db\ 64-bit host=PC0845
2018-07-18T09:44:05.128+0900 I CONTROL [initandlisten] targetMinOS: Windows 7/Windows Server 2008 R2
2018-07-18T09:44:05.128+0900 I CONTROL [initandlisten] db version v3.6.2
2018-07-18T09:44:05.128+0900 I CONTROL [initandlisten] git version: 489d177dbd0f0420a8ca04d39fd78d0a2c539420
2018-07-18T09:44:05.128+0900 I CONTROL [initandlisten] OpenSSL version: OpenSSL 1.0.1u-fips 22 Sep 2016
2018-07-18T09:44:05.129+0900 I CONTROL [initandlisten] allocator: tcmalloc
2018-07-18T09:44:05.129+0900 I CONTROL [initandlisten] modules: none
2018-07-18T09:44:05.129+0900 I CONTROL [initandlisten] build environment:
2018-07-18T09:44:05.129+0900 I CONTROL [initandlisten]   distmod: 2008plus-ssl
2018-07-18T09:44:05.129+0900 I CONTROL [initandlisten]   distarch: x86_64
2018-07-18T09:44:05.129+0900 I CONTROL [initandlisten]   target_arch: x86_64
2018-07-18T09:44:05.130+0900 I CONTROL [initandlisten] options: <empty>
2018-07-18T09:44:05.130+0900 W - [initandlisten] Detected unclean shutdown - C:\data\db\mongod.lock is not empty.

2018-07-18T09:44:06.652+0900 I CONTROL [initandlisten]
2018-07-18T09:44:06.652+0900 I CONTROL [initandlisten] Hotfix KB2731284 or later update is not installed, will zero-out data files.
2018-07-18T09:44:06.652+0900 I CONTROL [initandlisten]
2018-07-18T09:44:06.653+0900 I CONTROL [initandlisten] ** WARNING: The file system cache of this machine is configured to be greater than 40% of the total memory. This can lead to increased memory pressure and poor performance.
2018-07-18T09:44:06.654+0900 I CONTROL [initandlisten] See http://dochub.mongodb.org/core/vt-windows-system-file-cache
2018-07-18T09:44:06.655+0900 I CONTROL [initandlisten]
2018-07-18T09:44:08.404+0900 I FTDC [initandlisten] Initializing full-time diagnostic data capture with directory 'C:/data/db/diagnostic.data'
2018-07-18T09:44:08.412+0900 I NETWORK [initandlisten] waiting for connections on port 27017
2018-07-18T09:44:09.014+0900 I FTDC [ftdc] Unclean full-time diagnostic data capture shutdown detected, found interim file, some metrics may have been lost. OK

```

Default PORT 27017 を 기동되었습니다.

3.2.2. MongoDB 접속 테스트

> mongo

```

Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS C:\Users\WYONG> mongo
MongoDB shell version v3.6.2
connecting to: mongodb://127.0.0.1:27017
MongoDB server version: 3.6.2
Server has startup warnings:
2018-07-18T09:44:06.648+0900 I CONTROL [initandlisten]
2018-07-18T09:44:06.648+0900 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2018-07-18T09:44:06.649+0900 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.
2018-07-18T09:44:06.649+0900 I CONTROL [initandlisten]
2018-07-18T09:44:06.650+0900 I CONTROL [initandlisten] ** WARNING: This server is bound to localhost.
2018-07-18T09:44:06.650+0900 I CONTROL [initandlisten] ** Remote systems will be unable to connect to this server.
2018-07-18T09:44:06.650+0900 I CONTROL [initandlisten] ** Start the server with --bind_ip <address> to specify which IP
2018-07-18T09:44:06.651+0900 I CONTROL [initandlisten] ** addresses it should serve responses from, or with --bind_ip_all to
2018-07-18T09:44:06.651+0900 I CONTROL [initandlisten] ** bind to all interfaces. If this behavior is desired, start the
2018-07-18T09:44:06.651+0900 I CONTROL [initandlisten] ** server with --bind_ip 127.0.0.1 to disable this warning.
2018-07-18T09:44:06.652+0900 I CONTROL [initandlisten]
2018-07-18T09:44:06.652+0900 I CONTROL [initandlisten] Hotfix KB2731284 or later update is not installed, will zero-out data files.
2018-07-18T09:44:06.653+0900 I CONTROL [initandlisten] ** WARNING: The file system cache of this machine is configured to be greater than 40% of the total memory. This can lead to increased memory pressure and poor performance.
2018-07-18T09:44:06.654+0900 I CONTROL [initandlisten] See http://dochub.mongodb.org/core/vt-windows-system-file-cache
2018-07-18T09:44:06.655+0900 I CONTROL [initandlisten]

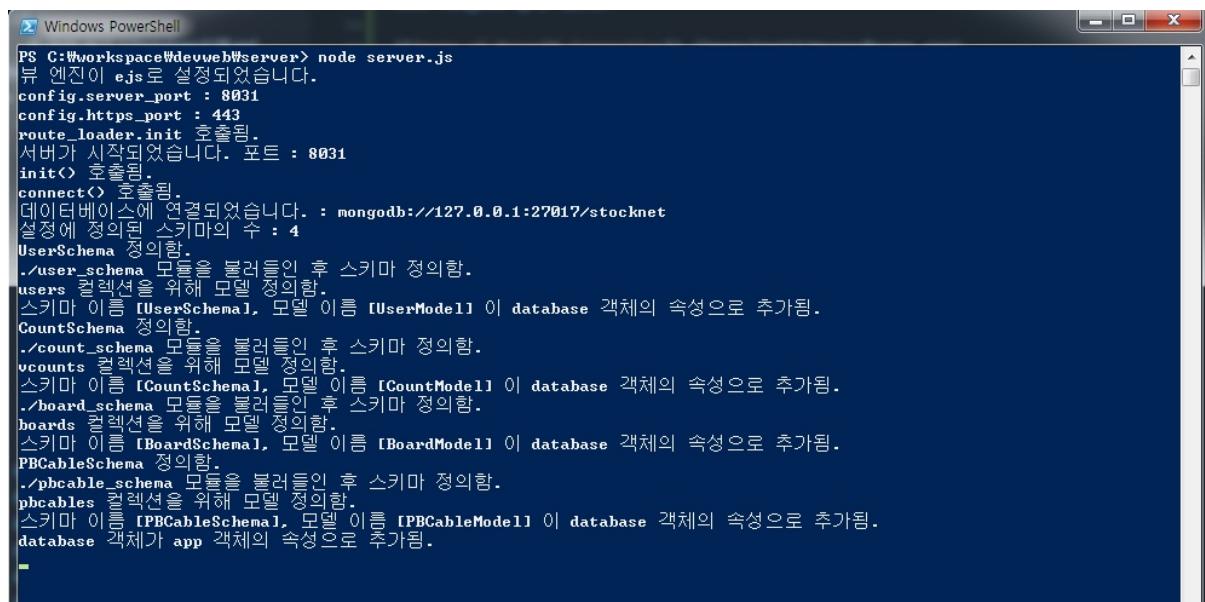
```

mongoDB 가 정상적으로 기동되었음을 확인하였습니다.

3.2.3. WAS 기동

NodeJS Server 프로그램을 기동하도록 하겠습니다.

```
> cd c:\workspace\devweb/server  
> node server.js
```

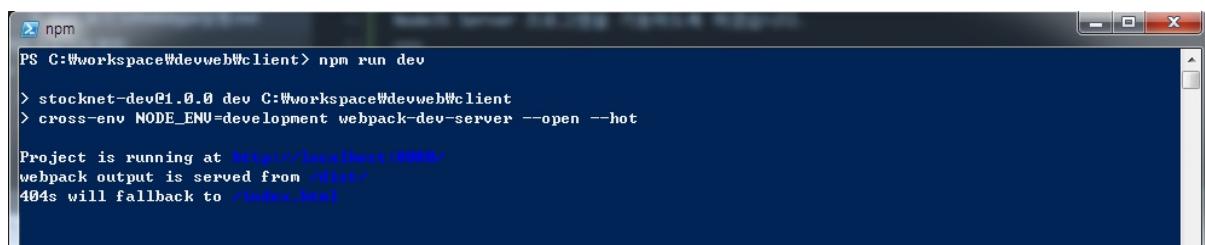


```
PS C:\workspace\devweb\server> node server.js
비埠 엔진이 ejr로 설정되었습니다.
config.server_port : 8031
config.https_port : 443
route_loader.init 호출됨.
서버가 시작되었습니다. 포트 : 8031
init() 호출됨.
connect() 호출됨.
데이터베이스에 연결되었습니다. : mongodb://127.0.0.1:27017/stocknet
설정에 정의된 스키마의 수 : 4
UserSchema 정의함.
./user_schema 모듈을 불러들인 후 스키마 정의함.
users 컬렉션을 위해 모델 정의함.
스키마 이름 [UserSchema], 모델 이름 [UserModel] 이 database 객체의 속성으로 추가됨.
CountSchema 정의함.
./count_schema 모듈을 불러들인 후 스키마 정의함.
스키마 이름 [CountSchema], 모델 이름 [CountModel] 이 database 객체의 속성으로 추가됨.
./board_schema 모듈을 불러들인 후 스키마 정의함.
boards 컬렉션을 위해 모델 정의함.
스키마 이름 [BoardSchema], 모델 이름 [BoardModel] 이 database 객체의 속성으로 추가됨.
PBCableSchema 정의함.
./pbicable_schema 모듈을 불러들인 후 스키마 정의함.
pbables 컬렉션을 위해 모델 정의함.
스키마 이름 [PBCableSchema], 모델 이름 [PBCableModel] 이 database 객체의 속성으로 추가됨.
database 객체가 app 객체의 속성으로 추가됨.
```

3.2.4. Client 기동

Client 프로그램을 기동하도록 하겠습니다.

```
> cd c:\workspace\devweb\client  
> npm run dev
```

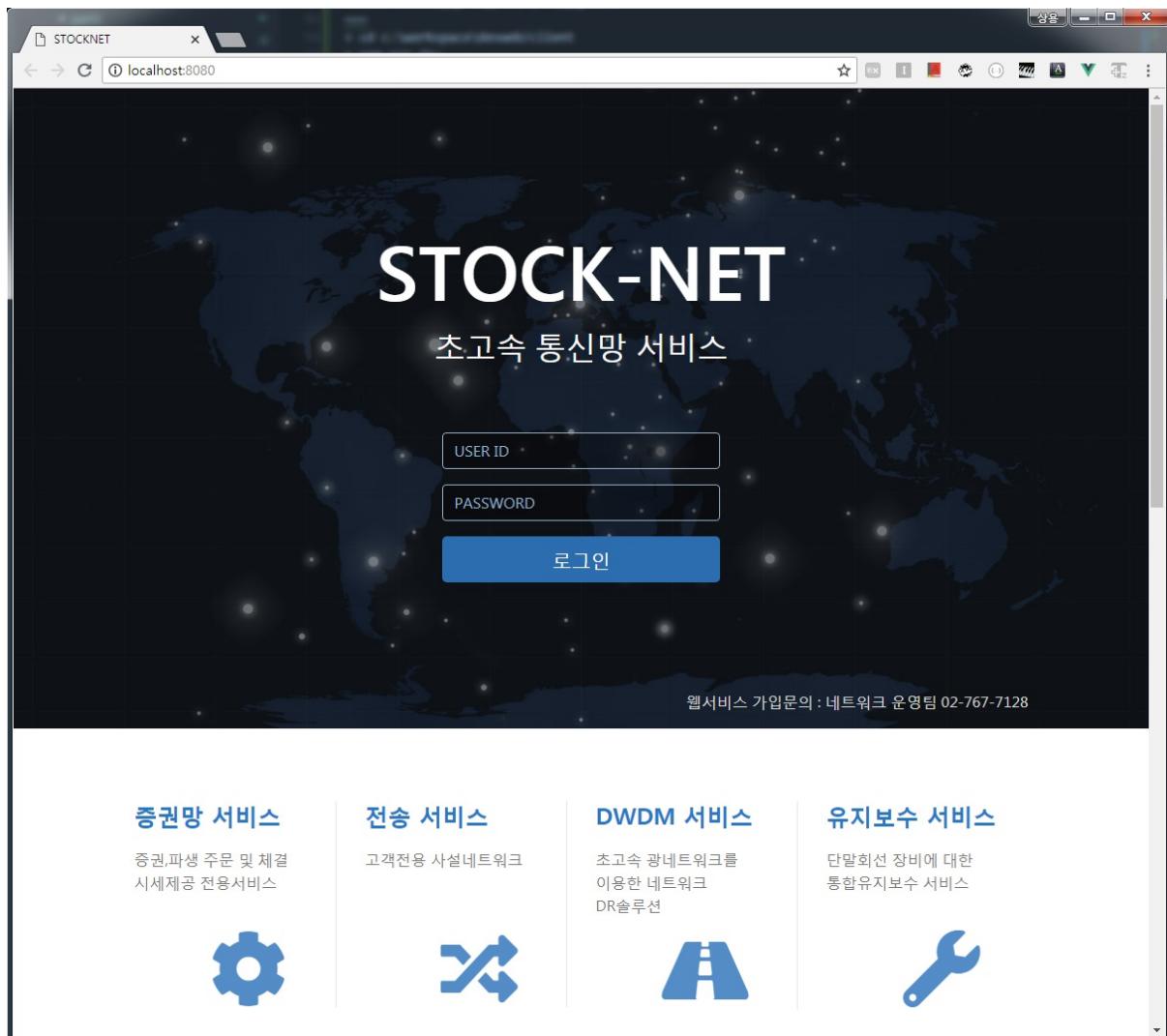


```
PS C:\workspace\devweb\client> npm run dev
> stocknet-dev@1.0.0 dev C:\workspace\devweb\client
> cross-env NODE_ENV=development webpack-dev-server --open --hot

Project is running at http://localhost:8080/
webpack output is served from /dist/
404s will fallback to /index.html
```

3.2.5. 브라우저 기동 및 기능 예제화면

자동으로 디풀트 브라우저가 기동됩니다.



로그인 가능

STOCKNET

localhost:8080/admin/notice

STOCK-NET 공지사항 일간성능 기간성능 사용자설정 관리자 님 LOGOUT

전체글 : 8

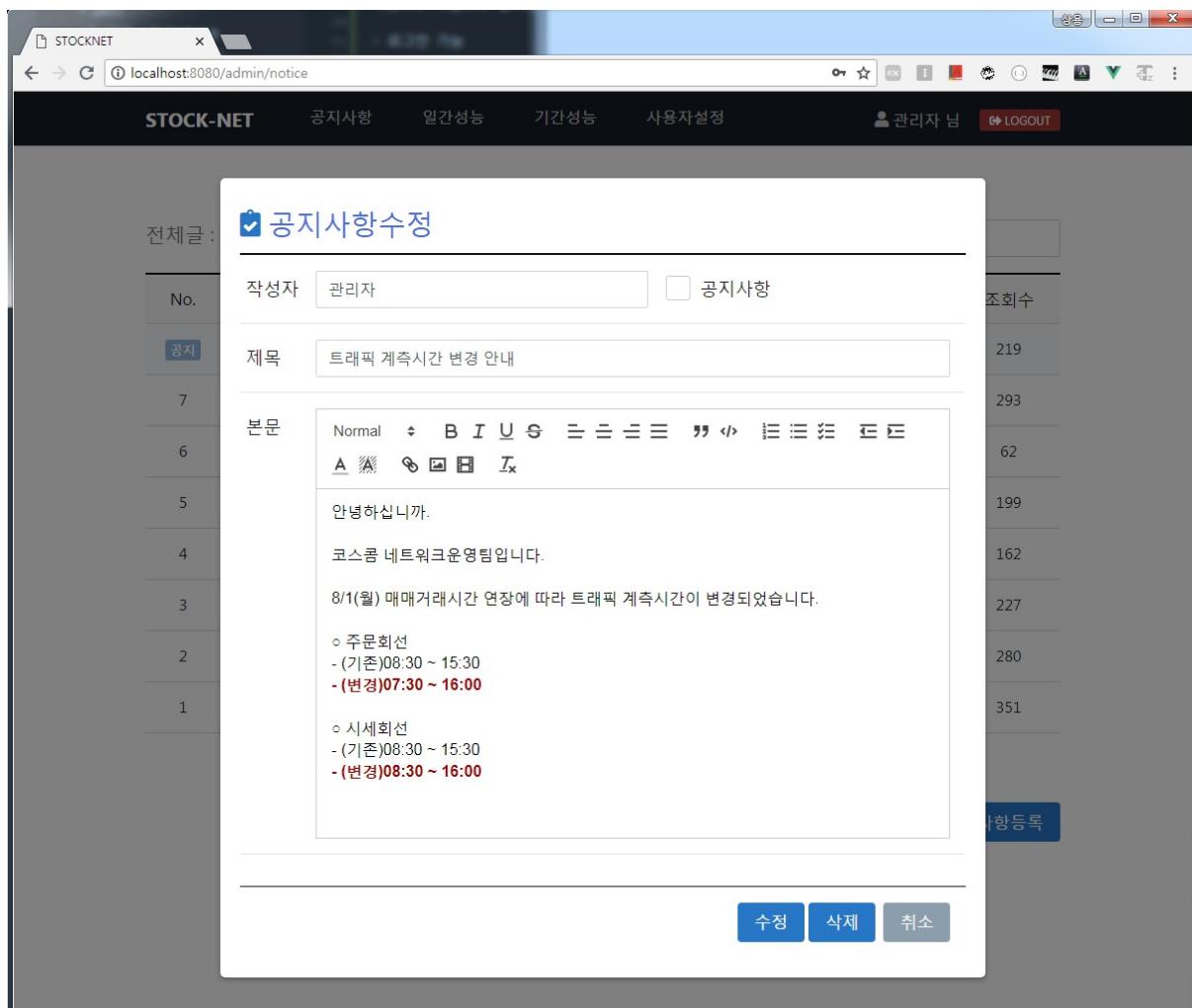
제목

No.	제목	작성자	날짜	조회수
공지	코스콤 증권망 회선 트래픽 웹서비스 사용안내	관리자	2018-05-16	219
7	트래픽 계측시간 변경 안내	관리자	2016-07-29	293
6	웹서비스 접속 URL 변경 안내	관리자	2015-09-10	62
5	회선 트래픽 추가 제공 안내 [주문 및 시세 테스트]	관리자	2015-10-07	199
4	트래픽 조회시간 변경에 대한 안내	관리자	2013-06-30	162
3	[사용법 안내] 메뉴 및 조회화면	관리자	2013-11-19	227
2	[사용법 안내] 트래픽 그래프 및 엑셀 자료	관리자	2015-10-07	280
1	코스콤 IT리스크관리부 담당자 연락처	관리자	2018-07-10	351

<> 1 >>

공지사항등록

게시판 가능



■ 웹 에디터 기능(오픈소스)

STOCKNET

localhost:8080/users

STOCK-NET 공지사항 일간성능 기간성능 사용자설정 관리자 님 LOGOUT

오늘방문자수 : (1) 누적수 : (2631) RESET

고객사명 : []

고객사명	사용자계정	핸드폰번호	E-mail	조회용 장비명	변경일자	당일접속	누적접속
KB증권	kbadmin	0261141418	trust@kbsg.com	HYUNDAI	2018-07-13	0	21
미래에셋대우	mrs3774	010		DAEWOO	2018-07-12	0	0
한국에스지증권	nekorea	010		SG	2018-07-12	0	0
유화증권	yuhwa	010		YUHWA	2018-07-12	3	279
교보증권	kyobo	010		KYODO	2018-07-12	0	13
유진투자증권	eu008	010		EUGENE	2018-07-12	0	0
관리자	netadmin	010	stocknet@itkoscom.co.kr		2018-07-11	1	578
netcruz	netcruz	010		ZZZZ	2018-07-11	0	2
미래에셋대우	dwsec08	010		DAEWOO	2018-07-11	0	9
한국에스지증권	sgcib03	010		SG	2018-07-11	0	7

<< 1 2 3 4 5 6 7 8 >>

사용자추가 PB회원추가

STOCK-NET

localhost:8080/users

STOCK-NET 공지사항 일간성능 기간성능 사용자설정 관리자 님 LOGOUT

오늘방문자수

고객사명	은
KB증권	
미래에셋대우	
한국에스지증권	
유화증권	
교보증권	
유진투자증권	
관리자	
netcruz	
미래에셋대우	

+사용자등록

고객사명 : 은

사용자계정 :

비밀번호 :

조회 장비명 :

접속허용IP :

접속시간 : 08 ~ 18 8시 ~ 18시

핸드폰번호 :

E-mail 주소 :

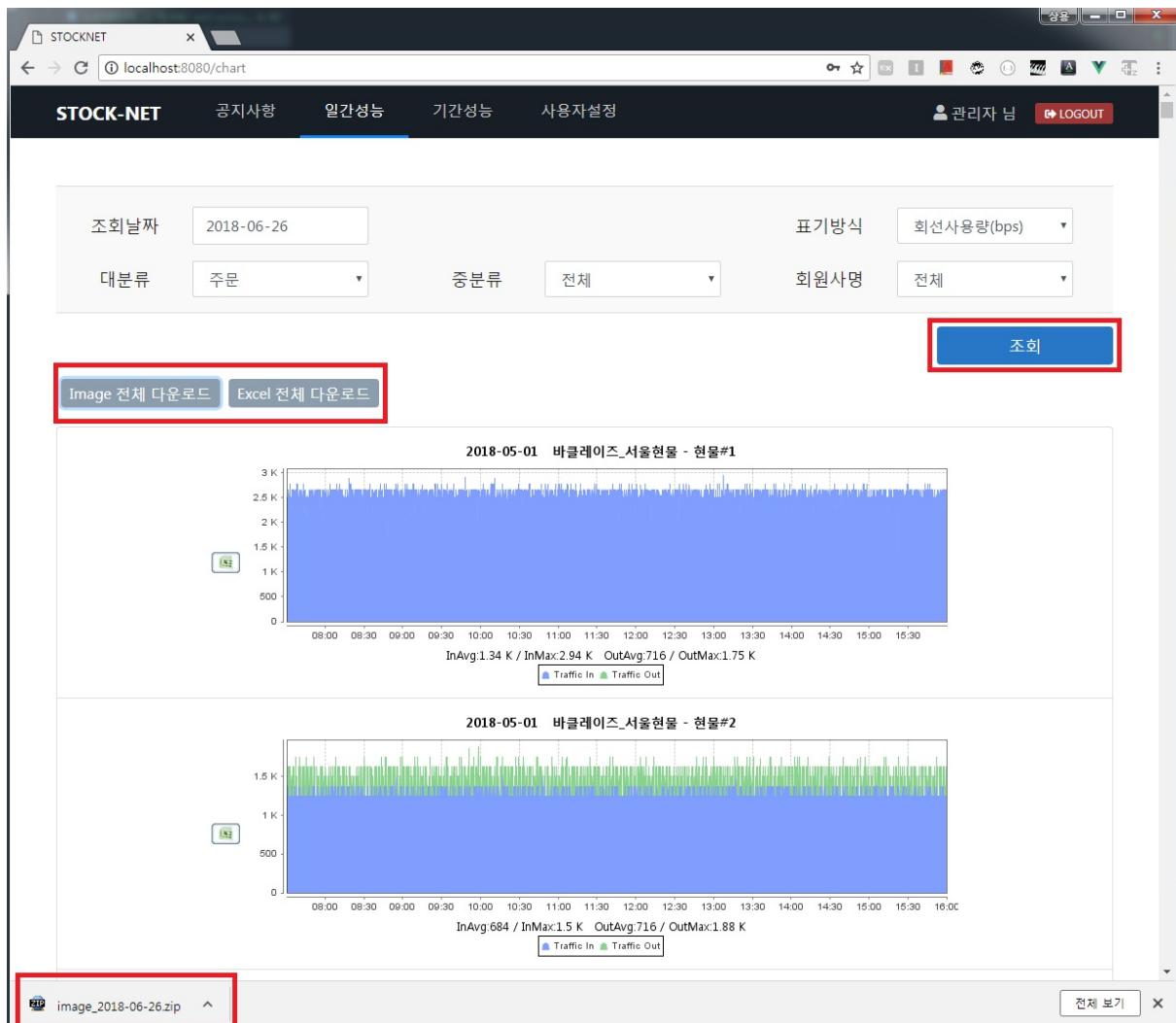
사용자 권한 : 일반고객

회선종류 : 주문 : 서울현물 서울파생 부산파생 주문테스트
 시세 : 유가 코스닥 ELW 차권 지수옵션 선물
 부산파생시세 시세테스트

등록 취소

당일 접속	누적 접속
0	21
0	0
0	0
3	279
0	13
0	0
1	578
0	2
0	9

사용자 관리 기능



파일 다운로드 가능

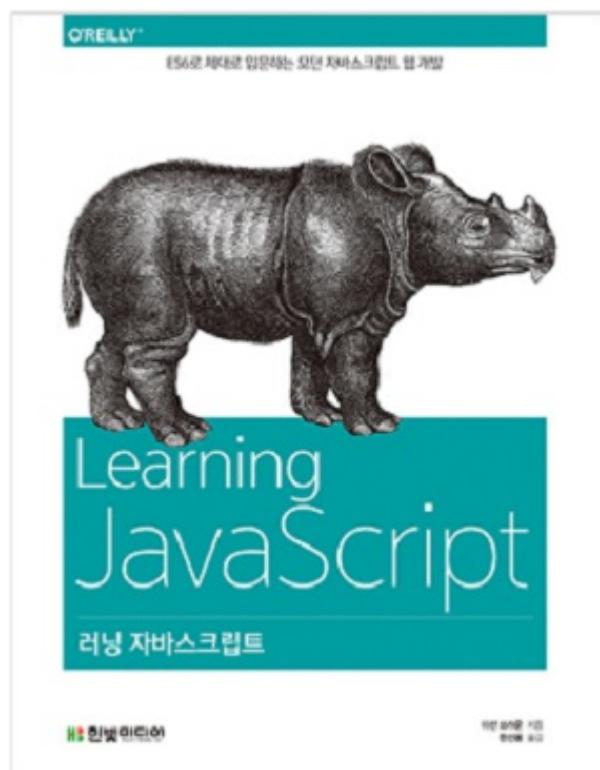
4. Server-Side Prototype

4.1. 사전 지식

Server-Side 프로그램을 이해하기 위해서는 **JavaScript**, **NodeJS**, **MongoDB**에 대한 사전 지식이 필요합니다. 관련 지식 모두를 다 알아야 할 필요는 없습니다. 필요한 지식부터 빠르게 습득해서 전체 구조를 파악한 후 부족한 부분은 채워나가면 됩니다. 추천드리는 책의 전부를 꼼꼼히 읽는 것보다 아래에서 언급드리는 토픽만 발췌해서 익힌 후에 소스코드의 전체 구조를 파악하고, 부족한 부분은 그때그때 자세히 살펴보는 방식으로 공부하면 됩니다.

4.1.1. JavaScript 문법

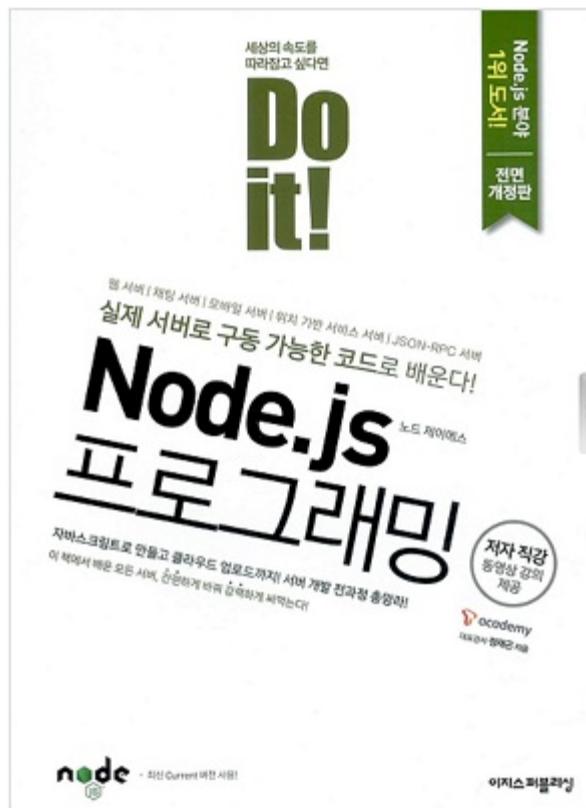
(참고도서)



- 데이터 타입, 제어문, 표현식, 연산자 등 기본 문법
- 함수(Java, C 등 기타언어와의 차이점), 스코프, 클로저
- 클래스, 프로토타입
- MAP, SET 등 자료구조
- 콜백과 프라미스
- 기타 날짜, 시간, 수학 함수, 정규표현식 등

4.1.2. NodeJS 기초

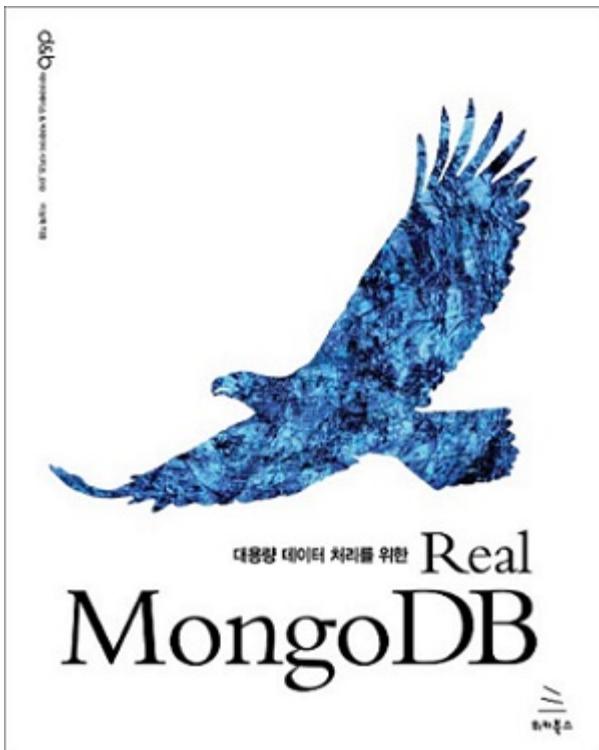
(참고도서)



- 모듈 사용
- 파일, 스트림 처리
- 미들웨어, Express 사용
- 데이터베이스 사용

4.1.3. MongoDB 기초

(참고도서)

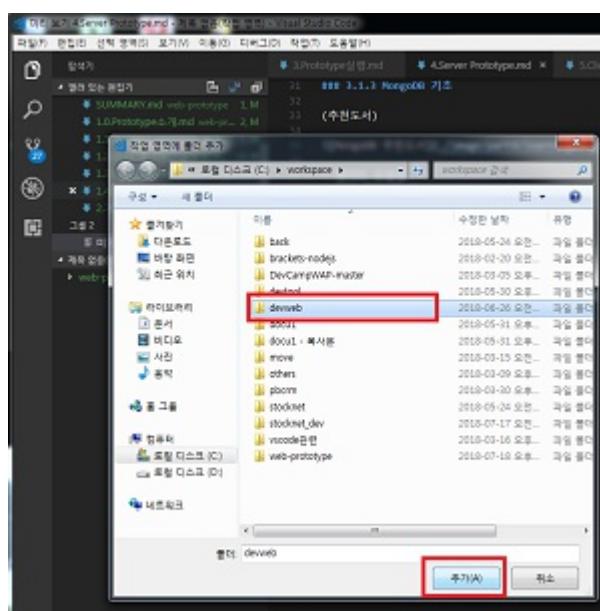
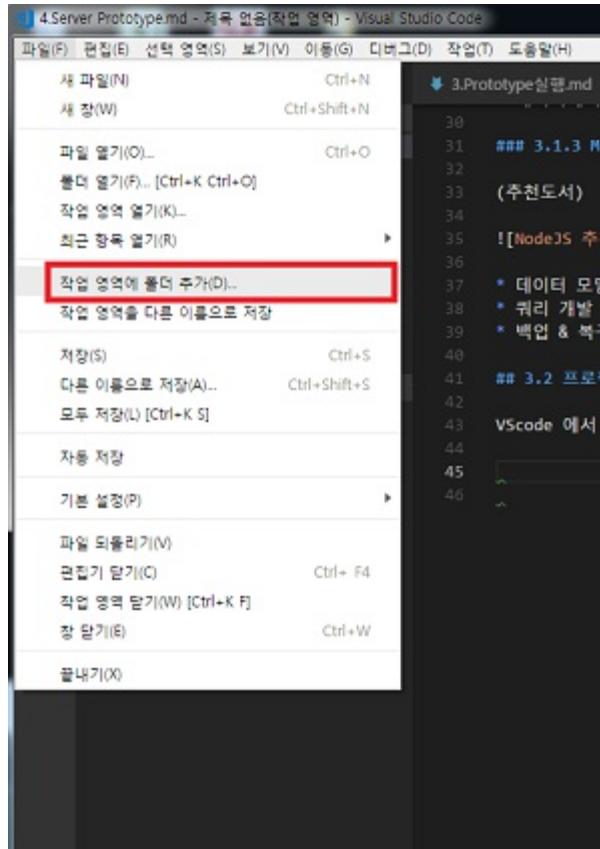


- 데이터 모델링
- 쿼리 개발 및 튜닝
- 백업 & 복구

4.2 Servier-Side Prototype 설명

4.2.1 프로젝트 열기

VScode에서 프로젝트를 불러옵니다.



The screenshot shows the Visual Studio Code interface. The title bar reads "4.Server Prototype.md - 파일 없음(작업 영역) - Visual Studio Code". The menu bar includes "파일(F)", "컴파일(E)", "선택 영역(S)", "복사(V)", "이동(G)", "디버그(D)", "작업(T)", and "도움말(H)". The left sidebar has icons for "파일(F)", "검색(S)", "디버그(D)", and "작업(T)". The "작업(T)" section shows a list of files under "4.Server Prototype.md web-prototype": "4.Client Prototype.md", "SUMMARY.md", "1.0.Prototype소개.md", "1.1.클서비스_아키텍처_발전단계.md", "1.2.2018_클개발기술동향.md", "1.3.기술부제.md", "1.4.MEVN_소개.md", and "2.개발환경구성.md". A red box highlights the "devweb" folder. The right pane displays the content of "3.Prototype실품.md", which includes code snippets for MongoDB, data modeling, and various sections like "## 3.2 프로젝트 명", "VScode에서 프로젝트", and project opening commands.

왼쪽 작업창에 devweb 폴더가 활성화 됩니다. /devweb/server/server.js 파일을 엽니다.

4.2.2 server.js

server.js 는 Server 프로그램의 메인에 해당하는 프로그램입니다. 참고도서 의 기본적인 내용을 숙지하였다고 가정하고 설명해나가도록 하겠습니다.

Prototype에 있는 server.js 파일의 내용들은 대부분의 WEB서버에서 공통적으로 지원해야되는 기능들입니다. 전체 구조를 파악하고 웹 프로그래밍에 익숙해지기 전까지는 이 코드들을 있는 그대로 사용하시면 됩니다. 주로 작업하게 될 부분은 server.js 에서 참조하는 route 나 database, 그리고 공통 설정화일인 config 쪽입니다. 해당 카테고리는 풀더별로 분리를 해놓았습니다.

나중에 Passport, Session, 기타 보안 관련 기능들을 추가하게 되면, 해당 모듈을 import 하면 됩니다.

```
/ Express 기본 모듈 불러오기
var express = require('express'),
    http = require('http'),
    path = require('path');

var bodyParser = require('body-parser'),
    cookieParser = require('cookie-parser'),
    static = require('serve-static'),
    cors = require('cors'),
    fs = require('fs'),
    errorHandler = require('errorhandler');

// 예러 핸들러 모듈 사용
var expressErrorHandler = require('express-error-handler');

// Session 미들웨어 불러오기
var expressSession = require('express-session');

// connect-flash 미들웨어 불러오기
var flash = require('connect-flash');

// 모듈로 분리한 설정 파일 불러오기
var config = require('../config/config');

// 모듈로 분리한 데이터베이스 파일 불러오기
var database = require('../database/database');

// 모듈로 분리한 라우팅 파일 불러오기
var route_loader = require('../routes/route_loader');
```

require를 통해 모듈을 import 합니다. 내장모듈과 외장모듈들이 있고, ./ 와 같이 상대경로로 잡혀있는 부분은 직접 작성한 모듈들입니다. 내장모듈은 별도의 설치없이 require 를 통해 바로 사용할 수 있는 모듈이고, 외장모듈은 설치 과정에 npm install 을 통해 설치했던 모듈들입니다. 직접 작성한 모듈은 적당한 경로에 모듈을 작성하고 해당 모듈을 require 하면 사용할 수 있습니다.

내장/외장 모듈에 대한 설명은 참고도서 를 참고하시기 바랍니다. 지금은 굳이 자세히 보지 않고, 이렇게 사용한다는 감만 익히고 넘어가도록 됩니다. config.js, database.js, route_loader.js 에 대해서는 아래 직접 참조하는 부분에서 설명하도록 하겠습니다.

```
// 악스프레스 객체 생성
var app = express();

//===== 서버 변수 설정 =====/
console.log('config.server_port : %d', config.server_port);
app.set('port', config.server_port); //8031

//Cross Origin Resource Sharing
```

```

app.use(cors());

// body-parser를 이용해 application/x-www-form-urlencoded 파싱
app.use(bodyParser.urlencoded({ extended: false }))

// body-parser를 이용해 application/json 파싱
app.use(bodyParser.json())

// cookie-parser 설정
app.use(cookieParser());

// 세션 설정
app.use(expressSession({
  secret: 'my key',
  resave: false,
  saveUninitialized: true
}));
```

```
var app = express();
```

웹서버를 NodeJS에서 제공하는 내장모듈로 직접 만들수 있지만, 여간 번거로운 작업이 아닙니다. 다년간 검증된 유일무이한 **express** 모듈을 사용하기로 합니다.

```
app.set('port', config.server_port); //8031
```

포트 정보를 별도의 설정화일에 분리시켜 놓았습니다. 시스템 **migration**, 테스트 등의 사유로 의외로 자주 수정하게 됩니다. **cors**, **parse**, **cookie**, **session** 등을 참고도서를 참조하시기 바랍니다.

```

// public 폴더를 static으로 오픈
app.use(static(path.join(__dirname, 'public')));
app.use('/public', express.static(path.join(__dirname, 'public')));

//라우팅 정보를 읽어들여 라우팅 설정
var router = express.Router();
route_loader.init(app, router);

//===== 404 에러 페이지 처리 =====/
var errorHandler = expressErrorHandler({
  static: {
    '404': './public/error/404.html'
  }
});

app.use(expressErrorHandler.httpError(404));
app.use(errorHandler);
```

```
// public 폴더를 static으로 오픈
app.use(static(path.join(__dirname, 'public')));
```

localhost:8031/index.html

으로 접근하게 되면, /public/index.html 파일을 회신하게 됩니다. 동적 처리가 아닌, 정적 파일(static)들에 대한 맵핑 처리를 하게 됩니다.

```
app.use('/public', express.static(path.join(__dirname, 'public')));
```

특정 경로의 URL을 특정 폴더로 맵핑하고자 한다면 위와 같이 설정하면 됩니다.

```
localhost:8031/public/image/test.png
```

으로 접근하면, /public/image/test.png 파일을 회신하게 됩니다.

```
localhost:8031/public/image/test.png
```

```
localhost:8031/image/test.png
```

위의 두 가지 접근은 같은 파일을 회신하게 됩니다.

맵핑하지 않은 폴더의 파일은 모두 /public 이하에서 찾게 됩니다.

```
//라우팅 정보를 읽어들여 라우팅 설정
var router = express.Router();
route_loader.init(app, router);
```

동적 접근에 대한 라우팅 설정입니다. /route/route_loader.js 파일에서 route 정보를 로드하고 app에 셋팅하게 됩니다. 다음 절에서 설명하도록 하겠습니다.

```
===== 404 에러 페이지 처리 =====/
var errorHandler = expressErrorHandler({
  static: {
    '404': './public/error/404.html'
  }
});

app.use(expressErrorHandler.httpError(404));
app.use(errorHandler);
```

에러 페이지에 대한 설정입니다. 404 에러에 대한 처리만 있는데, 다양한 에러코드에 대한 처리를 넣을 수 있습니다.

```
//확인되지 않은 예외 처리 - 서버 프로세스 종료하지 않고 유지함
process.on('uncaughtException', function(err) {
  console.log('uncaughtException 발생함 : ' + err);
  console.log('서버 프로세스 종료하지 않고 유지함.');

  console.log(err.stack);
});
```

예외에 대한 처리입니다. 예외가 발생했을 경우 에러코드를 로그로 남기기만 하고, 프로세스는 종료시키지 않습니다.

```
// 프로세스 종료 시에 데이터베이스 연결 해제
process.on('SIGTERM', function() {
  console.log("프로세스가 종료됩니다.");
  app.close();
});

app.on('close', function() {
  console.log("Express 서버 객체가 종료됩니다.");
  if (database.db) {
    database.db.close();
  }
});
```

프로세스 종료 이벤트에 대한 처리입니다. 데이터베이스에 대한 종료처리가 포함되어 있습니다. 외부 시스템간의 인터페이스가 있다면,

```
app.on('close', function() {  
})
```

에 추가하면 됩니다.

4.2.3 router 처리

```
// server.js
// 라우팅 정보를 읽어들여 라우팅 설정
var router = express.Router();
route_loader.init(app, router);
```

server.js에서 express Router 모듈을 사용한다고 선언하였고, Router 모듈에 routing 정보를 셋팅합니다.

```
// router.js
var route_loader = {};

var config = require('../config/config');

route_loader.init = function(app, router) {
    return initRoutes(app, router);
};

// route_info에 정의된 라우팅 정보 처리
function initRoutes(app, router) {
    var infoLen = config.route_info.length;
    for (var i = 0; i < infoLen; i++) {
        var curItem = config.route_info[i];
        // 모듈 파일에서 모듈 불러옴
        var curModule = require(curItem.file);
        // 라우팅 처리
        if (curItem.type == 'get') {
            router.route(curItem.path).get(curModule[curItem.method]);
        } else if (curItem.type == 'post') {
            router.route(curItem.path).post(curModule[curItem.method]);
        } else {
            router.route(curItem.path).post(curModule[curItem.method]);
        }
    }
    // 라우터 객체 등록
    app.use('/', router);
}
module.exports = route_loader;
```

config.js 파일에 들어있는 routing 정보를 로드하는 모듈입니다.

```
// config.js

module.exports = {
    route_info: [
        // 로그인
        { file: './admin/login', path: '/login', method: 'checkLogin', type: 'post' },
        { file: './admin/login', path: '/updatevisitcount', method: 'countInfo', type: 'get' },
        { file: './admin/login', path: '/logincount', method: 'usersloginCount', type: 'get' },
        // 공지사항
        { file: './admin/board', path: '/board/liststory', method: 'listStory', type: 'get' },
        { file: './admin/board', path: '/board/insertstory', method: 'insertStory', type: 'post' },
        { file: './admin/board', path: '/board/updatetestory', method: 'updateStory', type: 'post' },
        { file: './admin/board', path: '/board/deletetestory', method: 'deleteStory', type: 'post' },
        { file: './admin/board', path: '/board/updateviewcount', method: 'updateViewCount', type: 'post' },
        .....
    ]
}
```

route_info에 있는 부분을 살펴보면, **file**은 서브 모듈의 경로입니다. `./admin/login`은 `/routes/admin/login.js`를 나타냅니다. **path**는 클라이언트에서 접근하는 URL입니다.

`http://localhost:8031/login`

이라고 조회하게 되면, `/routes/admin/login.js` 안에 있는 `checkLogin` 함수가 처리를 한다는 뜻입니다.

```
// login.js
//로그인
var checkLogin = function(req, res) {
    console.log('users 모듈 안에 있는 checkLogin 호출됨.');
    if(정상)
        res.json({success: true, message: "OK", userid: userid,...})
    ....
else
    res.json({ success: false, message: "ERROR LOGIN" });
res.end();
```

`checkLogin()` 함수에서 `res` 객체에 조회결과를 담아서, 회신을 주면 한 사이클의 조회가 끝나게 됩니다.

5. Client-Side Prototype

5.1 사전지식

Client-Side 프로그램을 이해하기 위해서는 **HTML5, CSS3, JavaScript, VueJS**에 대한 사전지식이 필요합니다. 관련 지식 모두를 다 알아야 할 필요는 없습니다. 필요한 지식부터 빠르게 습득해서 전체 구조를 파악한 후 부족한 부분은 채워나가면 됩니다. 추천드리는 책의 전부를 꼼꼼히 읽는 것보다 아래에서 언급드리는 토픽만 발췌해서 익힌 후에 소스코드의 전체 구조를 파악하고, 부족한 부분은 그때그때 자세히 살펴보는 방식으로 공부하면 됩니다. 그리고, 관련 기술들이 빠르게 변하기 때문에 개념에 익숙해지면 책을 찾아보는 것보다 공식 문서를 더 자주 찾아보게 될 것입니다.

책으로 개념 파악, 공식 문서로 스킬업. 기억하시기 바랍니다.

5.1.1 HTML5 + CSS3

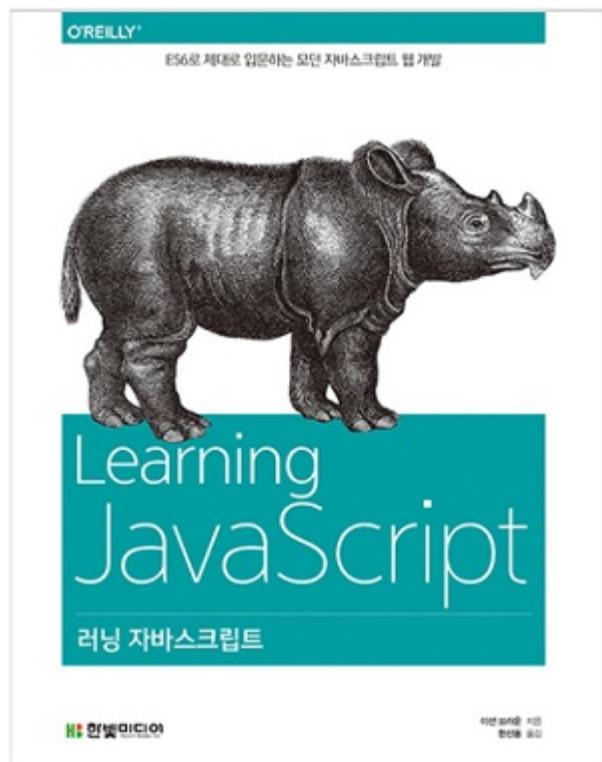
(참고도서)



- HTML 기본 문서 구조
- 텍스트, 목록, 표, 이미지, 폼 등 기본적인 태그 사용법
- CSS 기초
- 여백, 테두리 개념, 트랜지션, 애니메이션
- 반응형 웹, 레이아웃 구조

5.1.2 JavaScript 문법

(참고도서)



- 데이터 타입, 제어문, 표현식, 연산자 등 기본 문법
- 함수(Java, C 등 기타언어와의 차이점), 스코프, 클로저
- 클래스, 프로토타입
- MAP, SET 등 자료구조
- 콜백과 프라미스
- 기타 날짜, 시간, 수학 함수, 정규표현식 등

5.1.3 VueJS

(참고도서)



- VueJS 기초
- 디렉티브, 계산형 속성
- Vue Instance, 라이프 사이클
- 이벤트 처리
- 컴포넌트 처리
 - axios를 이용한 서버통신
- Vue-router
- webpack 사용법
- VueX는 필요시 학습하시면 됩니다.

5.2.1 index.html

SPA(Single Page Application) 유형의 Client-Side 프로그램은 기본적으로 `index.html` 하나의 파일에 필요한 모든 내용이 포함되어 내려갑니다. 먼저, `index.html`을 살펴보겠습니다.

```
<!DOCTYPE html>
<html lang="ko">

<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=11" />
  <title>STOCKNET</title>
</head>

<body>
  <div id="app"> </div>
  <script src="/dist/build.js"></script>
</body>

</html>
```

아무것도 없어 보입니다. 처음 component 형식의 html을 접할 때 가장 당황하게 되는 부분입니다. `index.html`을 수신한 브라우저는 파싱을 하면서 `/dist/build.js`를 요청하게 됩니다. Component의 모든 내용이 빌드라는 과정을 통해 `build.js`라는 하나의 파일로 만들어지게 됩니다. 빌드 정보는 `webpack.config.js`에 있습니다.

```
// webpack.config.js
module.exports = {
  entry: [
    './src/main.js'
  ],
  output: {
    path: path.resolve(__dirname, './dist'),
    publicPath: '/dist/',
    filename: 'build.js'
  },
}
```

`webpack`을 사용하는 방법은 별도로 학습을 해야 됩니다. 일단은 위와 같은 방식으로 사용한다고 이해하고 넘어갑니다. 웹자원들(`vue`, `js`, `css`, 이미지 등)을 하나 또는 관리하기 쉬운 몇 개의 파일로 묶는 과정입니다.

상단 `entry` 와 `output` 만 우선 보면 됩니다. Client-Side의 시작 프로그램은 `main.js`입니다.

```
// main.js

import Vue from 'vue'
import VueRouter from 'vue-router'
import VueResource from 'vue-resource';
import { routes } from './routes'
import store from './store'
import App from './App.vue'
import 'bootstrap';
import 'bootstrap/dist/css/bootstrap.css';
import 'vue-awesome/icons';
import Icon from 'vue-awesome/components/Icon';
import 'date-input-polyfill'
```

`component`에서 사용하는 모듈들을 `import` 합니다. 이 파일들은 빌드할 때 모두 `build.js`로 묶이게 됩니다.

```
// main.js

Vue.use(VueRouter);
Vue.use(VueResource);

Vue.component('icon', Icon);

const router = new VueRouter({
  routes,
  mode: 'history'
});

new Vue({
  store,
  el: '#app',
  router,
  render: h => h(App)
})
```

VueRouter를 통해 routes.js에 있는 모든 component를 로드하게 됩니다. 최초 진입 VUE는 App.vue가 됩니다.
다음절에서 routes.js 파일을 살펴보겠습니다.

5.2.2 routes.js

최상위 vue는 App.vue가 됩니다.

```
<!-- App.vue -->
<template>
  <div id="app">
    <router-view></router-view>
  </div>
</template>

<script>
export default {
  name: 'app',
  data () {
    return {
    }
  },
  methods: {
  },
  created() {
  }
}
</script>
```

<router-view> 를 통해 하위 component로 처리를 위임하고 있습니다. App.vue 파일은 더이상 수정할 일이 없습니다. route.js 에서 라우팅 정보를 등록하고, 하위 component 들을 관리하면 됩니다.

```
// routes.js
import Home      from './components/Home/Home.vue'
import AdminMain from './components/admin/AdminMain.vue'
import Notice    from './components/admin/Notice/Notice.vue'

.....
export const routes = [
  {
    path : '/',
    component: Home
  },
  {
    path : '/admin',
    component: AdminMain,
    children: [
      {
        path : 'notice',
        alias : '/notice',
        component: Notice
      },
      .....
    ]
  },
  {
    path : '/user',
    component: UserMain,
    children: [
      {
        path : 'notice',
        alias : '/notice',
        component: UNotice
      },
      .....
    ],
  }
]
```

`root path`에 대한 처리는 `Home.vue`가 처리하도록 되어 있습니다. 기타 `path`에 대한 처리도 해당 `component`를 정의함으로서 `path -- component` 형태의 직관적인 구조로 관리되고, 코드 분할이 자연스럽게 이루어집니다. 분할된 코드 간의 충복이 발생할 수 있지만, 기능간의 독립성을 보장하기 위해 과감히 충복된 코드를 허용하는 것이 유지보수 측면에서 더 생산적입니다. 특정 기능의 수정이 다른 기능의 수행에 최대한 영향을 주지 않도록 관리해야 합니다. 대규모 어플리케이션을 개발하고 운영해 본 개발자라면 이 부분이 의미하는 바를 잘 아시리라 생각합니다.

Server-side, Client-side, DBMS 등 관계없이 최대한 기능 중심, 화면 중심, 데이터 중심으로 아키텍쳐를 단순하게 구성해야 합니다. 그래서, 직관적으로 시스템을 파악할 수 있도록 만들어야 합니다. 이러한 구조가 잠재적인 장애 위험성을 현저히 낮출 수 있고, 개발 생산성을 극적으로 향상시킵니다. 대부분의 시스템 운영상황에서 맞닥뜨리는 특정 개발자 의존 문제라는 악순환에서도 탈피할 수 있습니다. 기본적인 사전지식, 공통지식만 있으면 누가와도 운영할 수 있는 환경을 만들어야 됩니다.

다음 절에서는 기본화면인 `Home.vue`를 예제로 살펴보겠습니다.

5.2.3. HomeVue

```
<!-- Home.vue -->
<template>
  <div class="container_net">
    <LoginForm></LoginForm>
    <div class="container cog">
      ...
      ...
      <div class="footer address">
        코스콤 네트워크 운영팀 : 02-767-7128 COPYRIGHT© KOSCOM CORP. ALL RIGHTS RESERVED.
      </div>
      <LoginNoticeModal v-if="showModal"></LoginNoticeModal>
    </div>
  </template>

  <script>
  import LoginForm from './LoginForm.vue';
  import LoginNoticeModal from './LoginNoticeModal.vue';

  ...
  components: {
    LoginForm: LoginForm,
  },

```

`Home.vue` 의 주요부분을 발췌했습니다. 3장 로그인 기능에서 표현된 로그인 화면을 처리하는 component입니다. 기본적인 html 코드 안에 `<LoginForm>` 이라는 하위 component가 포함되어 있습니다. Login Form 을 처리하는 component이고, 해당 component를 처리하는 vue 파일을 살펴보겠습니다.

```
<!-- LoginForm.vue -->
...
<script>
import Config from "../../js/config.js"
import Constant from "../../store/constant.js"
...
vm.$router.push({
  path: "/admin/notice"
});
...

```

로그인 처리가 완료되면, 로그인 이후 화면을 처리하는 component가 지정되어서, 그 화면으로 전환되어야 합니다. `vm.$router.push()` 가 그 부분을 처리합니다. `path`를 지정해줌으로서, `route.js` 에 정의되어 있던 component를 `App.vue`에서 `<route-view>`로 언급되었던 부분에 표시하게 됩니다. 즉, `Home.vue` component를 `AdminMain.vue`로 대체하게 되는 것입니다.

이후 화면들에서도 전환이 필요할 경우 같은 방식으로 component를 대체하게 됩니다. 단순히 메뉴나 링크를 통해 전환하는 방법도 있습니다. 그러한 세세한 부분들은 기능별 예제를 설명하는 다음장에서 살펴보도록 하겠습니다.

Clinet-Side에서 기본적인 화면 처리 절차를 살펴보았습니다. (참고도서) 를 통해 기본지식을 충분히 쌓으셨다면 `prototype`이 그렇게 어렵게 느껴지지는 않았을 것이라 생각됩니다.

6. 기능별 예제

6.1. 로그인 기능

(작성중)

6.2. 게시판 기능

(작성중)

6.3. 사용자 관리 기능

(작성중)

6.4. 다운로드 기능

(작성중)

7. 서버 배포

(작성종)