Array
JSON
underscore.js
MongoDB
Node.js

Array

데이터의 종류

- 문자열 String
  `“JavaScript”`
- 숫자
  `100`
- Boolean
  `true`
- 배열 Array

빈 배열

배열에 데이터를 표시

```
[ 0 ]
```

여러개의 데이터는 콤마(,)로 구분

```
[ 0, 10 ]
```

여러 데이터 종류를 혼합해 사용 가능

```
[ 0, 10, “a” ]
```

배열 안의 배열 (다차원 배열)

```
[ [ 0, 10 ], [ “a”, “b” ] ]
```

배열은 0번부터 시작

```
var arr = [ 0, 10, “a” ];

arr[ 1 ]
-> 10

var arr = [ [ 0, 10 ], [ “a”, “b” ] ]

arr[ 1 ][ 0 ]
-> “a”
```

JSON

JSON 소개

- JavaScript Object Notation
  - 자바스크립트 객체 표시법
  - 자바스크립트의 인기가 높아지면서 데이터 표기의 표준 방식으로 많이 채택됨

빈 객체

```
{}
```

데이터 표시의 기본

```
{ name : “Unknown” }
```

여러개의 데이터

```
{
    name : “Unknown”,
    email : “unknown@grotesq.com”
}
```

배열 형태의 데이터도 가능

```
{
    name : “Unknown”,
    email : “unknown@grotesq.com”,
    site : [ “grotesq.com”, “unk.sexy” ]
}
```

오브젝트 형태의 데이터도 가능

```
{
   name : “Unknown”,
   email : “unknown@grotesq.com”,
   site : [ “grotesq.com”, “unk.sexy” ],
   phone : {
      home : “02-000-1111”,
      mobile : “010-0000-1111”
   }
}
```

숫자, Boolean 형태의 데이터도 가능

```
{
    name  :  “Unknown”,
    email : “unknown@grotesq.com”,
    site : [ “grotesq.com”, “unk.sexy” ],
    age : 20,
    like_alcohol : true
}
```

XML 대비 장점

:: XML

```
<me>
    <name>Unknown</name>
</me>
```

:: JSON

```
{ name: “Unknown” }
```

적은 양의 텍스트로 데이터 표기 가능

XML 대비 단점

:: XML

```
<dom>
   <script><![CDATA[ … ]]></script>
</dom>
```

:: JSON

- 불가능

CDATA와 같은 기능을 기본으로 지원하지 않음.

underscore.js

underscore.js 소개

- 자바스크립트 라이브러리
- 오픈소스
- 기존 함수를 확장하는 헬퍼 모음
- http://underscorejs.org

underscore.js의 장점

- 다른 라이브러리 의존성 없음
- 내장 함수만 이용해 네이티브에 근접한 높은 성능
- 하나의 소스로 클라이언트와 서버 모두 작업 가능
- 작은 용량. minify 된 용량이 5kb

사용법

- Client-side

```
<script src="underscore-min.js"></script>
```

- Node.js

```
var _ = require( 'underscore' );
```

유용한 함수 #1 .each()

- Native JS

```
var arr = [ 0, 1, 2, 3, 4 ];
for( var i = 0, len = arr.length; i < len; i++ ) {
    alert( arr[ i ] );
}
```

- underscore.js

```
var arr = [ 0, 1, 2, 3, 4 ];
_.each( arr, alert );
```

유용한 함수 #1 .each()

```
var tags = [ ‘header’, ‘section’, ‘footer’ ];
_.each( tags, function( $element ) {
   document.createElement( $element );
} );
```

* 반복 처리될 함수에는 값, 인덱스(키), 리스트가 인자로 전달된다.

유용한 함수 #2 .find()

```
_.find(
   [ 1, 2, 3, 4, 5 ],
   function( $num ) {
      return $num % 2 == 0;
   }
);

-> 2

_.find( 콜렉션, 조건 );
```

유용한 함수 #3 .filter()

```
_.filter(
   [ 1, 2, 3, 4, 5 ],
   function( $num ) {
      return $num % 2 == 0;
   }
);

-> [ 2, 4 ]

_.filter( 콜렉션, 조건 );
```

유용한 함수 #4 .max(), .min()

```
var arr = [ 11, 26, 7, 0, 73 ];

_.max( arr );

-> 73

_.min( arr );

-> 0
```

유용한 함수 #4 .max(), .min()

```
var result = [
    { name: “Blackiz”, score: 70 },
    { name: “Desty”, score: 95 }
];
_.max( result, function( $person ) {
    return $person.score;
} );

-> { name: “Desty”, score: 95 }
```

유용한 함수 #5 .shuffle()

```
_.shuffle( [ 1, 2, 3, 4, 5, 6 ] );

-> [ 4, 1, 6, 3, 5, 2 ]
```

유용한 함수 #6 .omit()

```
var user = {
   name : “Unknown”,
   email : “unknown@grotesq.com”,
   password : “1a9s99edopidkjflej”
}
_.omit( user, “password” );

-> {
   name: “Unknown”,
   email: “unknown@grotesq.com”
}
```

유용한 함수 #7 .template()

```
var compiled = _.template(
                        “hello: <%= name %>” );
compiled( { name: ‘moe’ } );
-> “hello: moe”
```

유용한 함수 #7 .template()

```
var list = “” +
   “<%_.each( people, function( name ) {%>” +
      “<li><%= name %></li>” +
   “<% } ); %>”;

_.template( list, { people: ['moe', 'curly',
‘larry'] } );

-> "<li>moe</li><li>curly</li><li>larry</li>"
```

MongoDB

MongoDB 소개

- NoSQL Database
- JS 형태의 API
- JSON 형태의 데이터 구조

MongoDB 장점

- DB 설계 없이 빠르게 개발을 시작할 수 있음
- DB 구조가 자주 변할 때 유연하게 대응할 수 있음
- JS가 익숙한 사용자는 빨리 익힐 수 있음
- 빅데이터 처리에 유리함
- 분산처리(클라우드 컴퓨팅)를 처음부터 고려해 만들어졌음

MongoDB의 구조 - Document

가장 작은 단위
(SQL DB의 Record와 같은 개념)

```
{
    name : “Unknown”,
    age : 20,
    email : “unknown@grotesq.com”
}
```

MongoDB의 구조 - Collection

Document들이 모여서 Collection을 이룸
(SQL DB의 Table과 같은 개념)

- Collection ‘users’

```
[
    { name : “Unknown”, age : 20, email : “e@mail.com” }
    { name : “Blackiz”, email : “black@kiz.net” }
    { name : “Desty”, age : 20 }
]
```

MongoDB의 구조 - Database

Collection들이 모여서 Database을 이룸
(SQL DB에서도 동일하게 Database라는 이름으로 쓰임)

- 코드로 설명할 방법이 없다..

MongoDB의 CRUD

- Create - .insert()
- Read - .find()
- Update - .update()
- Delete - .remove()

MongoDB - .insert()

- users 콜렉션에 데이터를 추가

```
db.users.insert( {
   name : “Unknown”,
   age : 20
} );
```

MongoDB - .find()

- users 콜렉션에서 데이터를 조회

```
db.users.find();
```

- 조회를 할 때 옵션을 설정

```
db.users.find( { age : 20 } );
```

- age가 30 이하인 사람만 조회

```
db.users.find( { age : { $lt : 30 } } );
```

MongoDB - .update()

- users 콜렉션에서 name이 Unknown인 도큐먼트에 email 값을 업데이트

```
db.users.update(
   { name : “Unknown” },
   { $set: {
      email : “unknown@grotesq.com”
   } }
);
```

MongoDB - .remove()

・ users 콜렉션에서 name이 Blackiz인 도큐먼트를 삭제

```
db.users.remove( { name : “Blackiz” } );
```

MongoDB의 특징 - _id

- 모든 도큐먼트는 자동으로 _id 값이 같이 생성됨

```
db.users.insert( { name : “Desty” } );
-> { _id: “534a023d852b97cd786b6fff”,
     name: “Desty” }
```

- 생성할 때 _id를 지정도 가능

```
db.users.insert( { _id: 10,
     name : “Desty” } );
```

Node.js

Node.js

- 자바스크립트 기반의 웹 서버
- 크롬의 V8 자바스크립트 엔진 사용
- 이벤트 기반의 비동기 방식 서버
- 웹 소켓 등의 구현에 유리함

NPM

- Node Package Manager
- 노드에서 필요한 패키지와 라이브러리를 쉽게 설치할 수 있도록 만들어짐
- 쉽게 설치, 업데이트 할 수 있고 패키지 의존성 문제를 해결해 줌
- 노드용이 아닌 라이브러리도 많이 배포되고 있음

NPM - 설치

- 현재 폴더에 underscore.js 설치

```
npm install underscore
```

- 시스템 전체에서 사용할 수 있도록 설치할 땐 -g 옵션 이용

```
npm install -g underscore
```

NPM - 업데이트

- 현재 폴더에 underscore.js 설치

```
npm update underscore
```

NPM - package.json

```
{
   "name": "application-name",
   "version": "0.0.1",
   "private": true,
   "scripts": {
      "start": "node app.js"
   },
   "dependencies": {
      "express": “3.5.0", // 특정 버전 명시
      "mongojs": “*", // 최신 버전으로 설치
   }
}
```

NPM - package.json 으로 설치

- package.json을 프로젝트 폴더 루트에 두고

```
npm install
```

- 업데이트 할 때엔

```
npm update
```

Node.js - 라이브러리 로드하기

- 내장 라이브러리나 npm으로 설치된 라이브러리 로드

```
var http = require( ‘http’ );
```

- 직접 작성한 js 로드

```
var route = require( ‘./routes’ );
```

express - 웹 애플리케이션 프레임워크

- Node.js 진영에서 제일 많이 사용되는 프레임워크
- 쉽고 빠르게 웹 애플리케이션 제작 가능
- http://expressjs.com

express - 시작하기

- express 로드하기

```
var express = require( ‘express’ );
```

- express 객체 생성하기

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

express - 세팅

```
app.set( ‘port’, 3000 ); // 3000 포트 사용
app.set( ‘views’,
   path.join( __dirname, ‘views’ ) );
   // 뷰 템플릿 경로 지정
app.use( app.router ); // 라우터 사용
app.engine( ‘html’,
   require( ‘ejs’ ).renderFile );
   // 기본 뷰 렌더링 엔진 설정
```

express - 라우터 설정

```
app.get( ‘/’, function( $req, $res ) {
   $res.send( ‘Hello! express’ );
} );

app.get( ‘/users’, function( $req, $res ) {
   $res.send( ‘user list’ );
} );
```

express - 실행

node를 실행하고 서버 메인 파일을 지정

```
node app
```