inno SOPT Server

2차 세미나

Contents

22기 SOPT Server
2차 세미나

SHOUT OUR PASSION TOGETHER
SOPT

01

Node.js의 소개

- Node.js 소개
- Node.js 장단점

02

기본 내장 모듈

- URL
- Query String
- File System
- Crypto

03

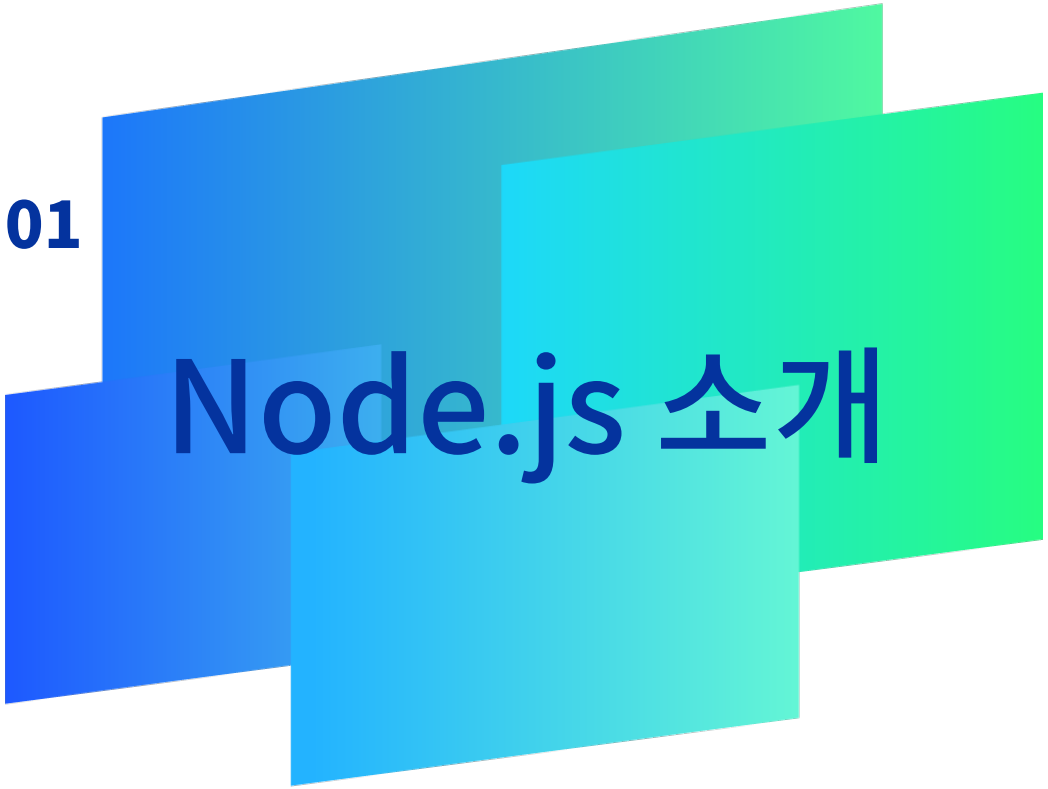
HTTP

- HTTP 소개
- Request
- Response

04

외부 모듈

- NPM
- Request
- CSV


01
Node.js 소개

- Javascript 기반의 서버 플랫폼
- Single Thread 기반의 비동기 I / O 지원
- Google Chrome V8 엔진으로 개발
- 이벤트 기반의 프로그래밍 모델 사용

Single Thread 기반의 비동기 I/O 지원???

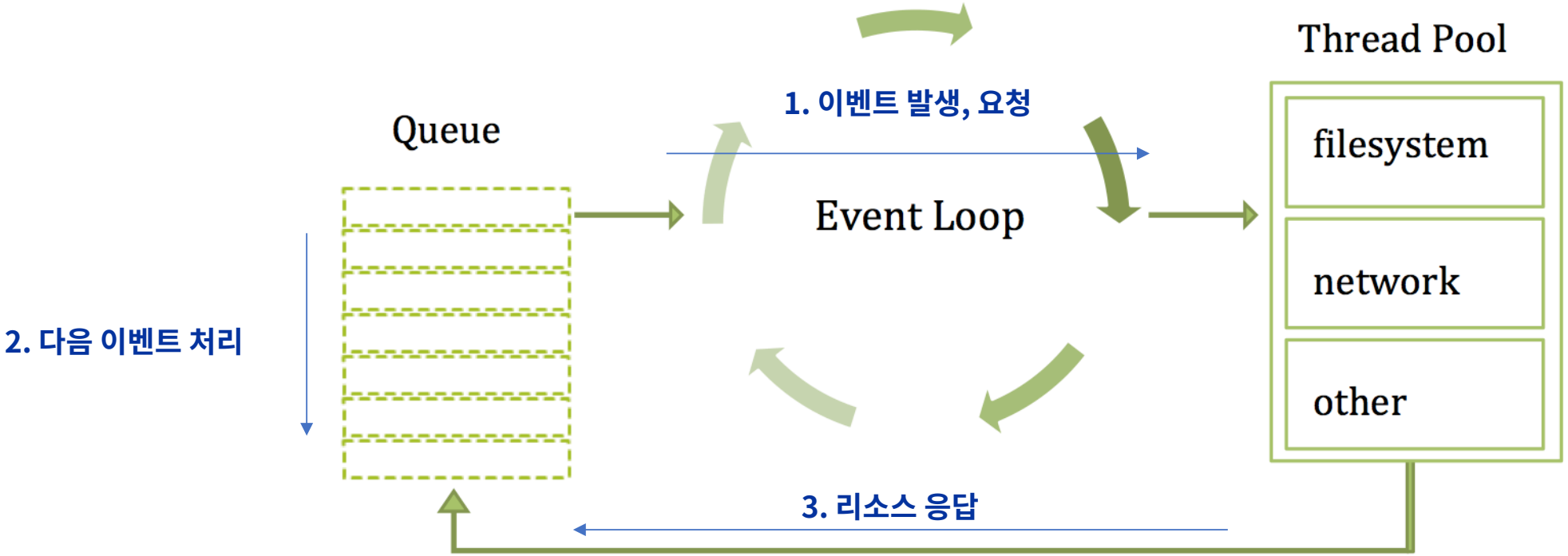
- 상황 : 3가지 일을 해야 한다. 빨래, 설거지, 청소 각각 한 시간씩 걸린다.
- 빨래를 다 하고 나서 그 다음에 설거지를 다 하고, 그 다음에 청소를 한다. => 총 3시간이 걸린다!

- 방법 1 : 3명이서 각각 하나의 일을 맡아 수행한다.
    - 동기방식, Thread를 여러개 만들어서 동시에 일을 처리
    - 동기 방식은 작업 요청이 들어올 때마다 Thread를 여러 개 만들어 동시에 일을 처리
    - 일이 많아질수록 Thread를 더 많이 만들어야 하므로 메모리 사용량 계속 증가. 동기화의 문제도 있음

Single Thread 기반의 비동기 I/O 지원???

- 방법 2 : 각각의 일을 수행하는 업체가 있다.
- 업체에 전화를 건다.하나의 작업을 하나의 업체에 맡긴다. 작업이 끝나면 알려달라고 말한다.
    - 비동기방식. 한 개의 Thread로 여러 개의 일을 요청 후 **결과를 받는 방식**(이벤트)
    - 뭐가 먼저 될 지는 모름
    - 작업이 아무리 많아도 시스템 리소스의 변화가 없다.
    - Node.js 의 방식 -> 대규모 네트워크에 적합

실제 Node.js의 작동 과정



- Event Loop 는 NodeJS의 싱글 쓰레드에서 돌아가며 I/O Bound 작업들을 비동기적으로 처리해주기 위해서 필요하다.
- I/O 처리나 네트워크 처리 등이 발생하면 이벤트를 요청하고 다른 작업을 수행한다.
- 이벤트처리가 완료되면, 콜백을 받아 다시 작업을 수행한다.

Node.js 장점

- Javascript 를 사용
- 구글이 제공하는 V8 엔진을 사용
- 성능이 매우 빠름 : **Single Thread** 기반의 **비동기 I/O처리**. **이벤트** 처리 방식
- 시스템 리소스의 부하가 적음

Node.js 단점

- 하나의 작업 자체가 시간이 많이 걸리면, 전체 시스템의 성능 급격하게 떨어짐
- V8 성능 이슈
- 멀티코어 머신에서 CPU 최적화 X
- 코드의 가독성 떨어짐 -> 유지 보수 어려움. **Callback Hell**
- 에러가 날 경우 프로세스 자체가 죽는다.

개발 관점에서는 빠르고 쉬운 장점이 있지만, 반대로 운영 관점에서는 테스트, 디버깅 등에 어려움이 있을 수 있다.

대규모 프로젝트나 게임서버보다는 **RESTful API 서버, 채팅 서버**등에 적절하다

02

기본 내장 모듈

- url 모듈 : url 정보를 파싱할 때 사용하는 모듈
- url 정보를 파싱하여 객체로 가져와 분석, 문자열로 바꿔주는 기능등을 수행
- 주요 메소드
    - parse(urlStr) : url 문자열을 url 객체로 변환해 리턴
    - format(ulrObj) : url 객체를 url 문자열로 변환해 리턴
    - resolve(from, to) : 매개변수를 조합해 완전한 url문자열을 생성해 리턴
- 그 외의 메소드 : https://nodejs.org/dist/latest-v8.x/docs/api/url.html

- query문?
  - HTTP에서 데이터를 전달하는 방식
  - url path와 query는 ? 로 구분
  - key1=value1&key2=value2&…
  - http://localhost:3000/test?key1=value1&key2=value2
- querystring 모듈 : url 객체의 query 와 관련된 모듈
- query문을 파싱하여 JSON 객체로 반환
- 주요 메소드
  - parse(queryString) : query 문자열을 객체로 변환해 리턴
  - stringify(JSONObject) : 객체를 query 문자열로 변환해 리턴
- 그 외의 메소드 : https://nodejs.org/dist/latest-v8.x/docs/api/querystring.html

- File System 모듈 : 파일을 읽고 쓰는데 사용하는 모듈
- 파일 읽기, 쓰기, 디렉토리 생성, 열기
- 주요 메소드
    - readFile(path[, options], callback)
    - readFileSync(path[, options], callback)
    - writeFile(path, data[, options], callback)
    - writeFileSync(path, data[, options], callback)
- Sync가 붙어 있는 메소드는 동기방식
- 그 외의 메소드 : https://nodejs.org/dist/latest-v8.x/docs/api/fs.html

- Crypto 모듈 : 문자열을 암호화, 복호화, 해싱하는 모듈
- 비밀번호를 해싱하여 저장하는데 주로 사용
- 주요 메소드
    - createHash(algorithm)
    - update(string)
    - digest(encoding)
    - randomBytes(length, callback)
    - pbkdf2(string, salt, iterations, length, algorithm, callback)
- 그 외의 메소드 : https://nodejs.org/dist/latest-v8.x/docs/api/crypto.html

단순 방식 암호화

- crypto.createHash(algorithm)

  .update(string)

  .digest(encoding);

- 의 방법으로 String을 해싱한다.
- Hashing Algorithm : SHA256, SHA512, SHA1, MD5…
- 주로 SHA256, 최근에는 SHA512 많이 사용
- Encoding으로는 data64, hex 등이 사용

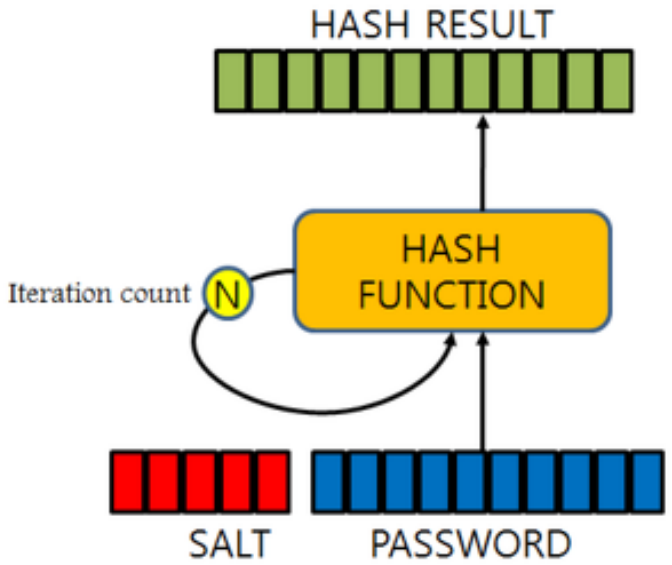
- 레인보우 테이블을 가진 레인보우 공격에 취약하다!
- 실제로는 절대 이 방법만 사용해서는 안됨. 하나마나한 해싱

Salt

- Salt : 해싱을 할 때 추가되는 바이트 단위의 임의의 문자열
- Salt를 해싱할 문자열에 추가하여 해싱하는 것을 Salting 이라고 한다.
- Salt와 암호화할 String을 섞어 해시 함수에 넣어 digest 생성
- 모든 String에 같은 Salt를 사용하면 무용지물!
- 랜덤바이트를 생성하여 임의의 Salt 정보를 생성 -> DB에 Salt 값을 같이 저장

Key Stretching

- String의 digest를 생성 -> digest를 입력 값으로 다시 digest 생성 -> 반복
- 동일한 횟수만큼 해시해야만 일치 여부 확인할 수 있다.
- digest를 생성할 때 어느 정도 시간이 소요되게 설정 => brute-force attack에 대비



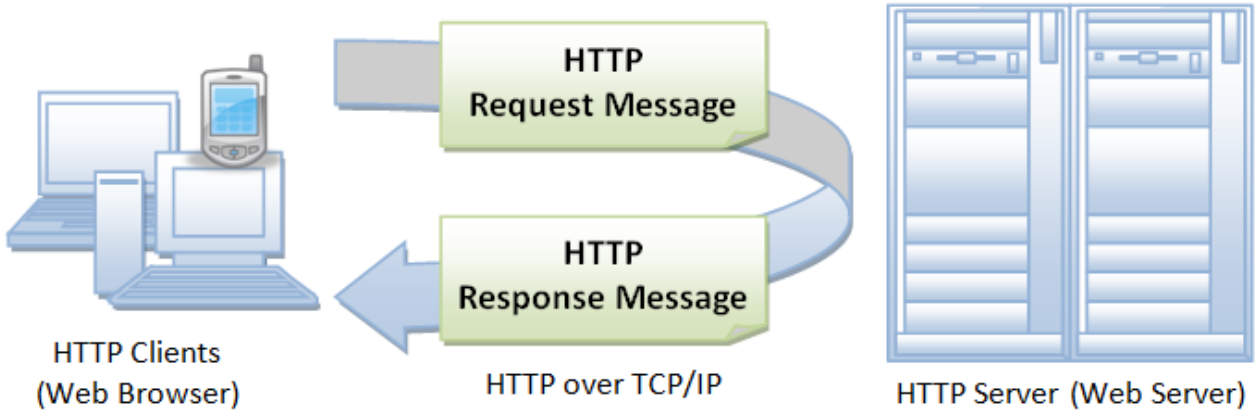
Salt + Key Stretching

- Salt와 String을 해시함수에 넣는 과정을 반복
- 반복횟수가 많아 질수록 복호화하기 어려워지지만 그만큼 시간도 많이 소모
- crypto.randomBytes(length, callback) 으로 랜덤 salt 생성
- crypto.pbkdf2(string, salt, iterations, length, algorithm, callback) 으로 해싱
- 생성된 값들은 toString을 사용하여 인코딩 해줘야 함
- 해싱 알고리즘으로는 검증된 SHA256, SHA512 등을 선택
- length는 해싱된 문자열의 길이로 임의로 설정


03
HTTP

HTTP : HyperText Transfer Protocol

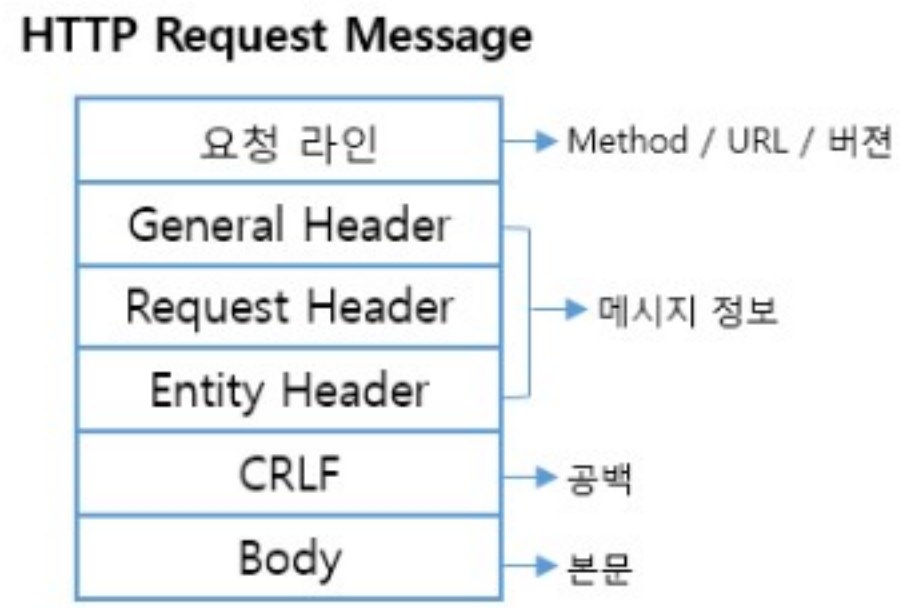
- TCP와 UDP를 사용하여 www상에서 정보 (주로 HTML 문서) 를 주고받을 수 있는 프로토콜
- 서버와 클라이언트 사이에서 이루어지는 요청/응답 (Request/Response) 프로토콜



HTTP 모듈로 서버 열기

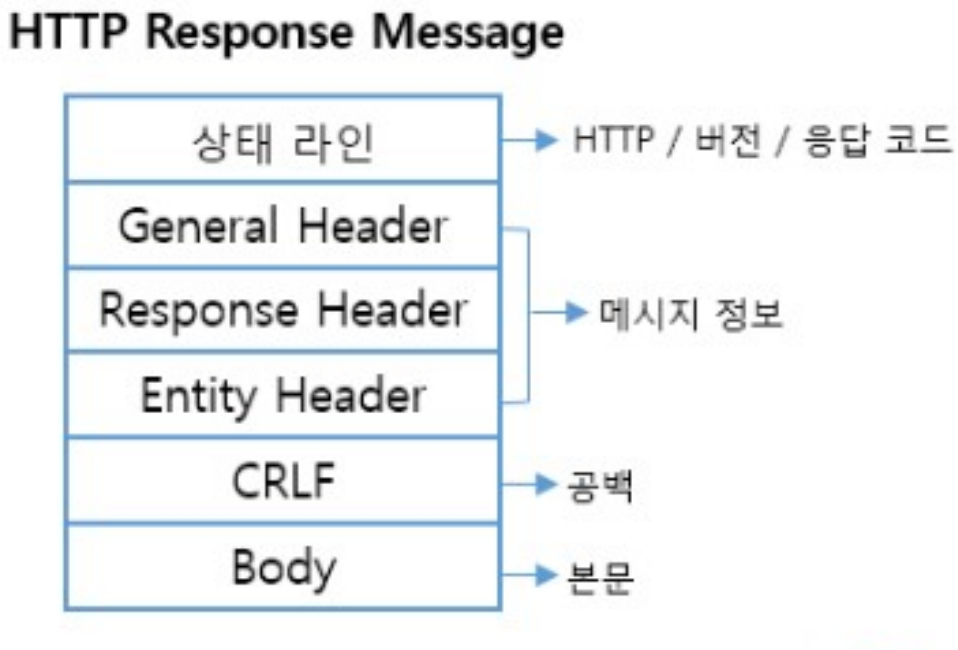
- http 모듈을 http 변수에 담는다.
- 추출한 모듈에서 createServer(callback) 메소드를 사용하여 서버를 생성한다.
- 생성한 서버에서 listen(port) 메소드를 사용하여 서버를 실행한다.
- port
    - 0 ~ 1023 : well-known port
    - 1024 ~ 49151 : registered port
    - 49152 ~ 65535 : dynamic port
- 포트가 사용중이라면 에러가 날 수 있으니 겹치지 않게 주의

HTTP Request 메시지 구조



- Client -> Server
- URL에는 url 주소가 들어감
- Method에는 GET, POST, PUT, DELETE 등의 통신 메소드가 들어감
- Headers에는 Host, Content-Type, Content-Length, User-Agent, Cookie 등이 들어감
- Body에는 데이터가 html, json, xml, form-data 등의 형식으로 들어감

HTTP Response 메시지 구조



- Server -> Client
- Status Code 에는 1XX, 2XX, 3XX, 4XX, 5XX 등이 들어감
- Headers에는 Content-Type, Content-Length, Set-Cookie 등이 들어감
- Body에는 데이터가 html, json, xml, form-data 등의 형식으로 들어감
- HTTP response 메시지는 첫번째 줄을 제외하고 HTTP request 메시지와 유사한 구조

주요 Status Code 종류

- 200 : 요청 성공적으로 처리 (GET 성공)
- 201 : 요청 성공적으로 처리, 새로운 리소스 생성 (POST 성공)
- 304 : 리소스 수정되지 않았음
- 400 : 요청 자체가 잘못 되었음
- 403 : 요청 처리 거부
- 404 : 찾는 리소스가 없는 경우
- 500 : Internal Server Error (서버 내부 오류)
- 501 : 서버 구현이 안 된 경우

주요 Content-Type 종류

- application/xml : xml 데이터
- application/json : json 객체 데이터
- application/x-www-form-urlencoded : html form 데이터
- multipart/formed-data : 사진
- text/plain : text 데이터
- text/html : html 데이터

04

외부 모듈

NPM : Node Package Manager

- Node.js 의 패키지 생태계
- Node.js 기반의 오픈 소스 모듈을 모아둔 저장소
- 세계에서 가장 큰 오픈소스 라이브러리 생태계
- 모듈의 버전관리가 쉽게 가능함

명령어

- 패키지 설치 : npm install 모듈명

  ex) npm install http
  npm install mysql
  npm install async

- npm install 만 입력할 시, package.json 에 기록된 모든 package 설치

설치 옵션

- --save : 모듈을 설치하며 package.json 파일에 모듈 정보 저장

  -> npm install 명령어만으로 쉽게 모듈을 재설치 할 수 있음

  ex) npm install http --save
  npm install --save mysql

- -g : 전역모드로 모듈 설치 -> 해당 프로젝트가 아닌 시스템에서 필요한 모듈

  (mac, linux 에서는 sudo 명령어로 root 권한을 줘야함)

  ex) npm install –g express-generator
  sudo npm install pm2 –g

- Request 모듈 : 서버 내부에서 다른 서버로 request 보낼 때 사용하는 모듈
- 다른 서버에 request를 보내 데이터를 받아옴
- 설치 : npm install request
- 주요 메소드
    - request(option, function(error, response, data))
- 그 외의 메소드 : https://www.npmjs.com/package/request

CSV : Comma-Seperated Values

- 몇 가지 필드를 쉼표(,) 로 구분한 텍스트 데이터 및 텍스트 파일
- xlsx 파일 등 엑셀파일에서 저장, 읽기 가능

  엑셀에서 열어서 편집 시 encoding 이 깨질 수 있음
- 엑셀 파일을 DB로 옮기거나 DB를 엑셀 파일로 옮기는 것이 가능
- 파일이므로 File System을 이용해 읽기, 쓰기 수행

json2csv

- csv 형식으로 저장하기 위해 JSON 객체를 CSV 형식으로 바꿔주는 모듈
- 설치 : npm install json2csv
- 주요 메소드
    - json2csv({data : JSON 배열, fields : column 배열})
- 컬럼 배열을 필드값으로 JSON 객체의 값이 하나의 row 로 들어감

csvtojson

- - csv 파일을 읽어와 JSON 객체배열로 바꿔주는 모듈
- - 설치 : npm install csvtojson
- - csvtojson.Converter를 추출한 다음 이것으로 새 컨버터 객체를 만들어서 사용

Homework

SHOUT OUR PASSION TOGETHER
SOPT

과제 1

1. 서버에 들어오는 url을 파싱하여 query로 만들고 이를 JSON 객체로 만들어주세요. query의 프로퍼티 키값은 str 입니다.
2. 서버에 접속시 str 값을 해싱합니다.

   * algorithm : SHA512, salting, key stretching을 모두 사용해주세요.
3. 해싱이 성공하면 JSON 데이터로 Response합니다. *JSON객체의 프로퍼티 키에는 msg, hashed가 있습니다.
4. msg에는 성공 혹은 실패 메세지를, hashed에는 해싱된 문자열을 넣어주세요.
5. 3000번 포트로 해당 서버가 동작하도록 열어주세요.
6. 코드는 이름_homework2-1.js 로 저장하세요.

Homework

SHOUT OUR PASSION TOGETHER
SOPT

과제 2

1. **13.125.118.111:3000/homework/2nd** 로 request를 보냅니다. 이 때 method는 **GET**입니다.
2. response되는 JSON객체의 property중 data를 CSV로 저장합니다.

   *response되는 객체의 내용을 모르겠다면 여기까지 작성후 console.log로 객체의 프로퍼티 값을 확인 후 진행합니다!

   *fields 값은 response의 프로퍼티를 확인해서 배열을 만들어야 합니다!
3. 저장에 성공하였다면 저장된 CSV 파일을 열어서 response를 전송합니다.

   *request 모듈을 사용하며 받은 응답이 저장된 변수가 있더라도 꼭 CSV파일을 다시 열어서 response를 보내주세요!
4. 코드는 이름_homework2-2.js 로 저장하세요.

Homework

SHOUT OUR PASSION TOGETHER
SOPT

과제 3

1. localhost:포트/info 의 url로 접속시 서버는 **POST** 메소드로 **13.125.118.111:3000/homework/2nd** 로 request를 보냅니다.
2. 이 때 request를 보내는 body는 key값으로 **name, phone**을 가집니다. value는 자신의 이름과 휴대폰번호 '010-xxxx-xxxx'의 스트링입니다.
3. 이름과 휴대폰번호에 해당하는 데이터가 없으면 homework서버는 fail을 응답합니다.
4. 해당하는 데이터가 존재하면 console.log 로 데이터를 확인 후 적절히 파싱하여 csv파일에 이름, 학교, 학과, 이메일, **해싱된 휴대폰번호**를 저장합니다.
5. 저장성공을 알리는 메세지를 **웹페이지에 띄웁니다**.
6. 코드는 이름_homework2-3.js 로 저장하세요.

이름_homework2-1, 2, 3.js 파일을 이름_homework2 폴더에 넣고 압축한 뒤

구글 드라이브에 업로드 해주세요! 제출기한은 4월 21일 23시 59분 59초까지 입니다.

inno SOPT

2차 끝:)