**NodeJS교과서 정리**

1,2장 스킵

NodeJS가 무엇인가

NodeJS나 JS에 관한 문법 내용



* **모듈로 만들기**

모듈 : 특정한 기능을 하는 함수나 변수들의 집합

자바스크립트에서 코드를 재사용하기 위해 함수로 만드는것과 비슷한 의미

module.exports = {odd, even}

module.exports = checkOdddOrEven <- 함수임 함수명을 정의해놓으면 사용가능

* **global 객체**

브라우저의 window와 같은 전역객체 모든 파일에서 접근가능

자스에서 window.open -> open으로 쓰는거와 같음global.require -> require(); global.console() -> console();

노드JS 에는 window와 document 객체가없다.

* **console 객체**

**console.time() :** console.timeEnd()과 대응되어 같은 레이블을 가진 time과 timeEnd 사이의 시간측정

console.log() : 평범한 로그를 콘솔에 표시

console.error() : 에러 콘솔

console.dir(객체,옵션) : 객체를 콘솔에 표시할 때 사용 console.dir(obj, {colors: false, depth : 2};

console.trace() : 에러가 어디서 발생했는지 추적

* **Timer**

setTimeout(callback function, milliseconds) : 주어진 밀리초 이후의 함수 실행

setInterval(callback function, milliseconds) : 주어진 밀리초마다 콜백 함수 반복실행

setImmediate(callback function) : 콜백 함수 즉시 실행

clearTimeout(id) : settimeout 취소

clearInterval(id) : setInterval 취소

clearImmediate(id) : setImmediate 취소

* \_\_filename, \_\_dirname

경로에 대한정보 제공

* module, exports

참조 관계

exports -> module.exports -> { }

exprots 객체 사용시에는 module.exports 참조 관계가 깨지지 않도록 주의해야한다.

한 모듈에서 사용시 exports와 module.exports는 동시에 쓰지말 것

* process

prove 객체는 현재 실행되고 있는 노드 프로스세에 대한 정보를 담음

process.env : 환경변수 입력공간 서비스의 중요한 키를 저장하는 공간 (DB 비미럽호 API키등)

process.nextTick() : 다른 콜백 함수들보다 nextTick의 콜백 함수 우선 처리

promise.resolve()는 process.nextTick()과 같이 다른 콜백들보다 우선됨 이러한 일을

마이크로태스크(microtask)라 구분 짓는다.

process.exit() : 노듶프로세스 종료용

* os

운영체제의 정보

운영체제가 관리하는 메모리, 홈 디렉터리 경로, 코어수 정보등을 보여줌

* path

운영체제별로 경로가 다름

Windows : C:\Users\Kim (\ 구분자)

POSIX : /home/Kim (/ 구분자)

path.join(경로, 경로 ,…) : 여러 경로를 **상대경로**로 합쳐줌

path.resolve(경로, 경로,…) 여러 경로를 절대경로로 처리함

Windows에서 경로문제를 해결하는데 path는 요기한게 쓰일수 있다

Windows 화경에서 posix Path 사용path.posix.sep(), path.posix.jon() 마찬가지로

Posix 환경에서 windows경로사용 path.win32.sep(), path.win32.join()

둘이 필요한경우에만 꼭 쓰기

* URL

url. 처리방식은 WHATWG(웹표준 정의단처), 기존 노드에서 사용하는 방식 두가지가 있다.

url.parse(주소) : 주소를 분해 WHATWG 방식과 다른 username, password, 대신 auth 속성, searchParams 대신query가 있다.

url.format(객체) : WHATWG 방식의 url과 기존 노드의 url 모두 사용할 수있다. url 객체를 다시 원래 상태로 조립

* querystring

url 대신 기존노드 url 분해방식 사용할 때 search부부능ㄹ 사용하기 쉽해 객체로 만들어줌

querystring.parse(parseUrl.query) : url의 쿼리 부분을 자바스크립트 객체로 분해

querystring.stringfy(객체): 분해된 쿼리 객체를 문자열로 다시 조립

* crypto

암호화 모듈

* **단방향 암호화**

createHash(알고리즘 문자열) : md5, sha1 ,sha256, sha512 사용

update(문자열) : 변환할 문자열

digest(인코딩) : 인코딩할 알고리즘 base64, hex, latin1 사용

pbkdf2를 많이쓰는데 bcrypt나 scrypt 보다취약 소스코드 참고

* **양방향 암호화**

필요하면 보기

* **이벤트**

event 모듈을 사용하여 이벤트를 생성함

이벤트 객체를 사용하여 이벤트를 생성할수 있음

on(name, callback function) :이벤트 이름과 이벤트 발생시의 콜백 연결. 연결하는 동작을 이벤트 리스닝이라 함, 여러 개의 이벤트 달수있다

addListener(name, callback function) : on과 기능이 같음

emit(name) : 이벤트를 호출, 미리 등록했던 이름으로 콜백 실행

once(name, callback function) : 한 번만 실해오디는 이벤트

removeAllListeners(name) : 이벤트 연결된 모든 이벤트 리스터 제거

removeListener(name, Listener) : 특정 이벤트만 제거

off(name, callback function) : 노드 10 버전에서 추가, removeListner와 기능이강틈(on에 대응인가봄)

listenerCount(name) : 현재 리스너가 몇 개 연결되었는지 확인

* 예외처리

try catch 문 사용

내장 모듈 자체내의 에러를 발생시킬 때 에러로그를 기록 해 둠 내부에서 에러시 **throw를** 던지므로 해당 부분은 try catch로 잡아 별도 처리필요(로그등)

process.on(‘uncaughtException’ (err) => {}); 사용 : 처리하지 못한 에러 발생시 리스너 실행후 프로세스가 유지됨

**‘uncaughtException’은** 이벤트는 최후의 수단으로 써야함

* “” or ‘’ 둘 중에 무엇을?

사실상 “” 이나 ‘’ 둘 중 아무거나 사용해도 상관은 없으나 ‘’을 선호한다

“empty” === ‘empty2’ 이다.

보통 작은 따옴표(‘)를 쓰나 **문자열 리터럴을 만들기 위해** 큰 따옴표(“)를 사용한다.

**단 예외적으로 JSON은 큰따옴표(“)만 허용함을 잊지마라**

* cluster

cluster 모듈은 싱글스레드인 노드가 CPU 코어를 모두 사용할 수 있게 해주는 모듈

포트를 공유하는 노드 프로세스를 여러 개 두어 병렬로 실행된 서버의 개수만큼 분사 요청을 할 수가 있다.

**세션 공유불가능(당연히 싱글스래드이므로)** -> Redis 도입으로 해결가능

* npm

Node Package Manager의 약자

대부분의 자바스크립트 프로그램은 패키지라는 이름으로 npm에 등록되어있다.

npm에 업로드된 노드 모듈은 **패키지**라 부른다

npm init 으로 초기화가 가능하다.

package name : 패키지의 이름, package.json name속성에 저장

version: 패키지의 버전입니다. npm의 버전은 엄격하게 관리됨

entry point : 자바스크립트의 실행파일 진입점, 보통 마지막으로 module.exports 하는 파일을 지정, package.json main 속성에 둠

test command : 코드를 테스트할 때 입력할 명령어를 의미. package.jsopn scripts 속성 안의 test 속성에 저장

git repository: 저장해둔 깃 저장소 주소 사용

keywords : 키워드는 npm 공식홈페이지에서 패키지를 쉽게 찾을수 있게해줌

licnese: 해당 패키지의 라이선스를 넣는공간

* **여러가지 명령어**

npm install [Package Name] [Package Name] : package-Json에 기록후 node\_modules에 추가

npm install –save-dev [PackageName] : 개발용 패키지 설ㅊ 실제 배포시에는 설치 안됨

npm install -g(--global) [PackageName] : 글로벌 설치 Windows Users\AppData\Roaming\npm, macOS /usr/local/lib/node\_modules에 설치

npx 명령어 : 전역설치를 하고싶지 않을 때 사용하는 옵션 전역설치시 보통 package.json에 기록되지 앟아 다시 설치할 때 번거롭다., npm install –save-dev [PackageName]후 npx [Packagename] [do] 를 적는다.

npm install [Repository Addtress] : 아직 npm에 실리지않은 패키지를 깃허브나 형상관리툴에 저장되어있는경우 사용할 수 있다.

npm update [Package name] : 업데이트 업데이트시 outdated에 적힌 wanted로 업데이트 된다.

npm uninstall [Package name] : 해당 패키지를 제거 node\_modules폴더 package.json에서 사라짐

npm search [검색어] npm 패키지 검색 그러나 npmjs.com을 보는게 더 편함

npm info [Package name] 패키지의 세부 정보 파악, package.json과 의존 관계 설치 가능한 버전등 포함

npm adduser : npm로그인을 위한 명령어 npm 공식 사이트에서 가입한 계정으로 로그인, 패키지 배포시 필요함

npm whoami : 로그인한 사용자 안내용

npm logout : adduser로 로그인한 계정 로그아웃

npm version [version] : package.json의 버전 상승용

npm deprecate [패키지명][버전] [메시지] 해당 패키지 설치시 경고 메시지를 띄움 자신이 만든 패키지에만 가능

npm publish

npm unpusblish 배포한 패키지 제거용 24시간 이내에 제거 – 의존성 때문에 제약이 있음 다른 사람이 쓰고 있을수도 있기 때문에

이외의 명령어 docs.npmjs.com에서확인

명령어 줄어쓰기 – npm install -> **npm -i**, --save-dev -> **-d**, --global -> **-g**

* **패키지 버전**

SemVer(Semantic Versioning) 사용

첫번째 자리는 major 버전 : 주 버전이 0이면 개발중 1부터는 정식 버전 major 버전은 하위 호환이 안 될 정도 일 때 패키지의 내용이 수정되었을 때 올린다.

두번째 자리는 minor 버전 : minor 버전은 하위 호환이 되는 기능 업데이트시에 올림.

세번째 자리는 patch 버전 : 기존 기능에 문제가 있어 수정한 것을 올릴 때

버전에 내용은 절대로 수정되면 안된다.

**패키지 앞에 붙는 기호**

^ : minor 버전 까지만 설치 보통 minor버전까진 호환이 되기 때문에 업데이트 설치로 많이둠

~ : patch 버전 까지만 설치 또는 업데이트

npm outdated : 업데이트 할수 있는 패키지 확인

* Express

HTTP 모듈의 요청과 응답객체의 추가 기능을 부여하고 코드를 분리하여 쉽게 만들고 관리하게 용이하게 만듬

if문으로 요청 메서드와 주소 구분 안해도 됨

npm -i -g express-generator : express-generator는 콘솔명령어 이므로 전역 설치 필요

express <프로젝트 명> : 프로젝트 생성

app.js : 핵심적인 서버 역할

bin : www파일은 서버를 실행하는 스크립트

public : 외부(클라이언트)가 접근가능한 파일들, 이미지, 자바스크립트., css 파일등

routes : 서버의로직이 작성되는 공간

views : 화면 부분이 작성되는공간

models : DB 공간작성

에제에서는 MVC 패턴을 만들라고함 routes(라우터)를 컨트롤롤러로 봤을 때 이다.

npm start : 서버실행

* **app.js**

요청과 응답의 중간(middle)에 위치하여 미들웨어라고도 부름

app.js에서 수많은 app.use(~~~)가 미들웨어들임