# Node.js 내장 모듈/객체

| 생성일 | @2024년 11월 4일 오후 12:25 |
| --- | --- |
| 태그 | |

▼ Console

- 용도
  - 웹 브라우저에서 제공하는 자바스크립트 console과 유사
  - 주로 디버깅을 위해 사용
- 메소드

| 종류 | 설명 |
| --- | --- |
| console.log(data[, ...args]) | 일반적인 로그 출력 |
| console.error(data[, ...args]) | 에러와 관련된 로그 출력 |
| console.table(tabularData[, properties]) | 배열/오브젝트를 데이블 형태로 출력 |
| console.time([label])<br>/ console.timeEnd([label]) | time과 timeEnd 사이에 코드가 실행된 시간을 출력 |
| console.dir(obj[, options]) | 객체를 콘솔에 출력 |

- Console 클래스
  - Node.js에서 파일 쓰기와 같은 스트림에 사용 → 로그를 파일로 관리가능
  - 문법

```
const fs = require('fs');
const {Console} = require('console');

const output = fs.createWriteStream('로그 파일 경로');
const logger = new Console({ stdout : output});

logger.log('log : 로그 메세지 %s', '');
```

- 전역 객체 console
  - require 없이 전역 객체로 바로 사용 가능

▼ Timers

- 용도
  - 웹 브라우저에서 제공하는 타이머 API와 유사
  - require 없이 전역 함수로 바로 사용 가능
- 스케줄링 함수(Scheduling Timers)

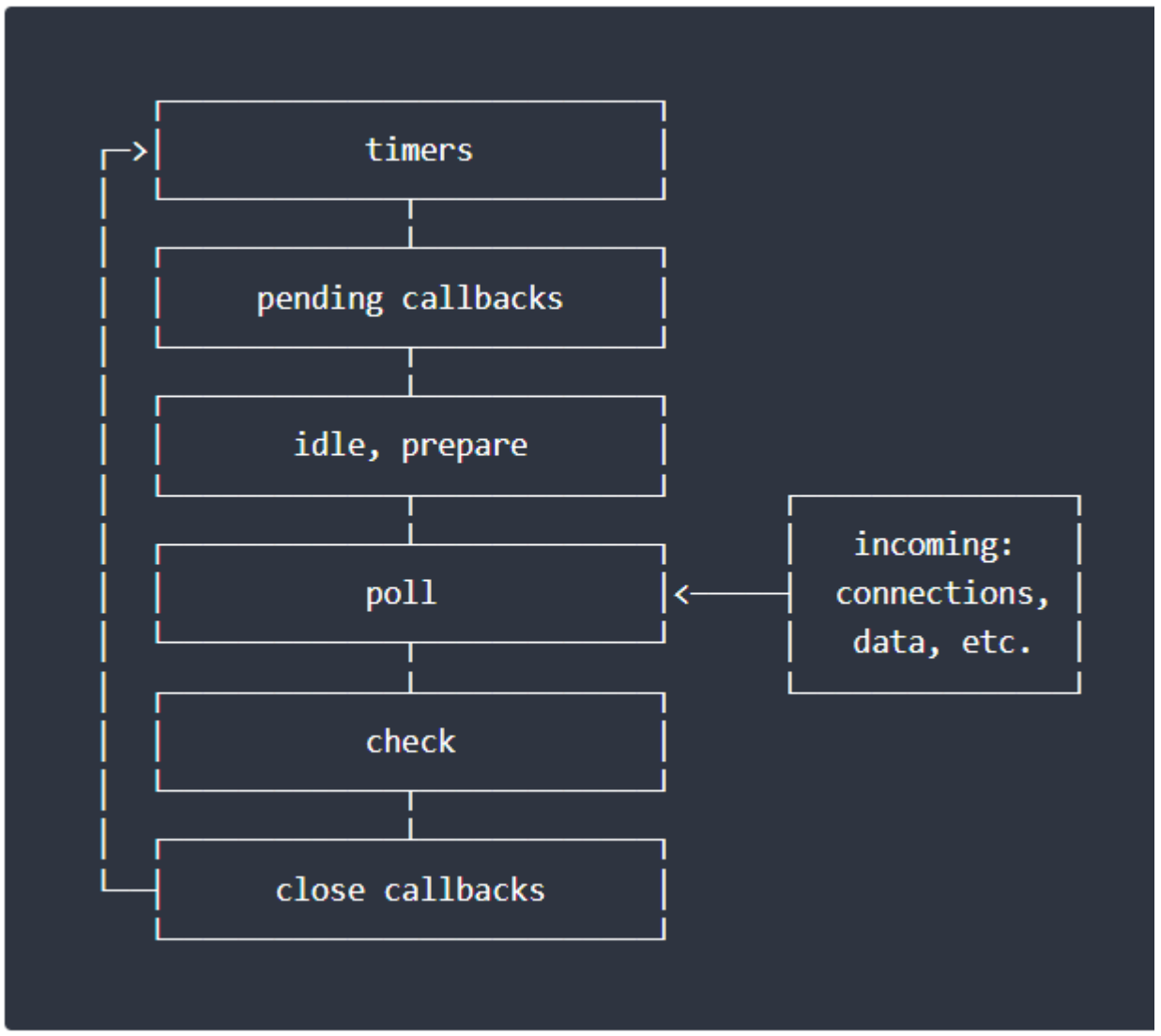
| 종류 | 설명 |
| --- | --- |
| setTimeout(callback, delay[, ...args]) | 설정한 밀리초 이후에 지정된 콜백함수 실행 |
| setInterval(callback, delay[, ...args]) | 설정한 밀리초 마다 지정된 콜백함수 실행 |
| setImmediate(callback[, ...args]) | 현재 이벤트 루프 주기 끝에 코드를 실행 |

  - 주의사항
    - setTimeout과 setInterval을 사용하는 경우 지연 시간이 보장되지 않음
  - 작업 스케줄 취소
    - 스케줄링 함수를 호출할 때 반환받은 값과 대응되는 작업중지 함수를 사용
    - 작업중지 함수(Cancelling Timers)

        - clearTimeout, clearInterval, clearImmediate

▼ 이벤트 루프

- 내부 구성
    - 특정 용도별 Phase가 여러개 존재
    - Phase
        - Queue로 구현
        - Queue의 모든 작업을 수행하거나 제한 갯수까지 실행 후 다음 Phase로 이동




- Phase 별 관련 함수

| **Phase** | **대상** | **처리 작업** |
|---|---|---|
| timer | setTimeout(func, delay)<br>setInterval(func, delay) | delay가 지났으면, 등록된 Callback 실행 |
| poll | I/O | 대부분의 Callback 실행 |
| check | setImmediate(func) | 등록된 Callback 실행 |

▼ Process

- 용도
    - 현재 실행되고 있는 Node.js 프로세스에 대한 정보와 제어를 제공
    - 전역으로 사용할 수 있으나 명시적으로 호출을 권장(require or import)

▼ Process events

- 메소드

| 종류 | 설명 |
|---|---|
| on(event, listener) | 지정한 이벤트의 리스너를 추가 |
| once(event, listener) | 지정한 이벤트의 리스너를 추가하지만<br>한 번 실행한 후에는 자동으로 리스너 제거 |
| removeListener(event, listener) | 지정한 이벤트에 대한 리스너를 제거합니다. |

- 주요 이벤트

| 종류 | 설명 |
| --- | --- |
| beforeExit | 이벤트 루프를 비우고 예약할 추가 작업이 없을 때 |
| exit | process.exit()를 호출되거나<br>이벤트 루프에 더 이상 수행할 추가 작업이 없을 때 |
| disconnect | 프로세스가 IPC 채널을 통해 생성된 경우<br>(클로스터로 자식 프로세스를 생성) IPC 채널이 닫힐 때 |
| message | 프로세스가 IPC 채널을 통해 생성된 경우 childprocess.send()를 사용하여<br>상위 프로세스에서 보낸 메시지가 하위 프로세스에서 수신될 때 |

▼ process.env

- 사용자 환경을 포함하는 객체를 반환

▼ process.nextTick

- 용도
  - 콜백 함수를 nextTickQueue에 등록 → 이벤트 루프를 실행 전에 수행할 작업

  ▼ nextTickQueue 와 microTaskQueue

  - 공통사항
    - 이벤트 루프에 앞서 실행을 목적으로 함
    - 이벤트 루프를 구현한 libuv 라이브러리에 포함되지 않음
  - nextTickQueue
    - process.nextTick() API의 콜백함수 등록
  - microTaskQueue
    - Resolve된 프로미스의 콜백함수 등록

  ▼ process.exit()

  - 용도
    - 실행 중인 Node.js 프로세스를 종료
  - 사용
    - 프로그램을 운영하며 변경된 설정 값이 서버에 반영되어야 하는 경우 재시작

▼ OS

- 용도
  - 운영체제 관련 유틸리티 함수 및 속성 정보를 제공
- 사용
  - 프로그램 내 임시 파일을 저장하기 위한 경로 확인
  - 클러스터를 구성할 때 현재 서버의 CPU 코어 수 확인

▼ Path

- 용도
  - 파일과 디렉터리 경로 작업을 위한 유틸리티를 제공
  - require를 이용
- 메소드

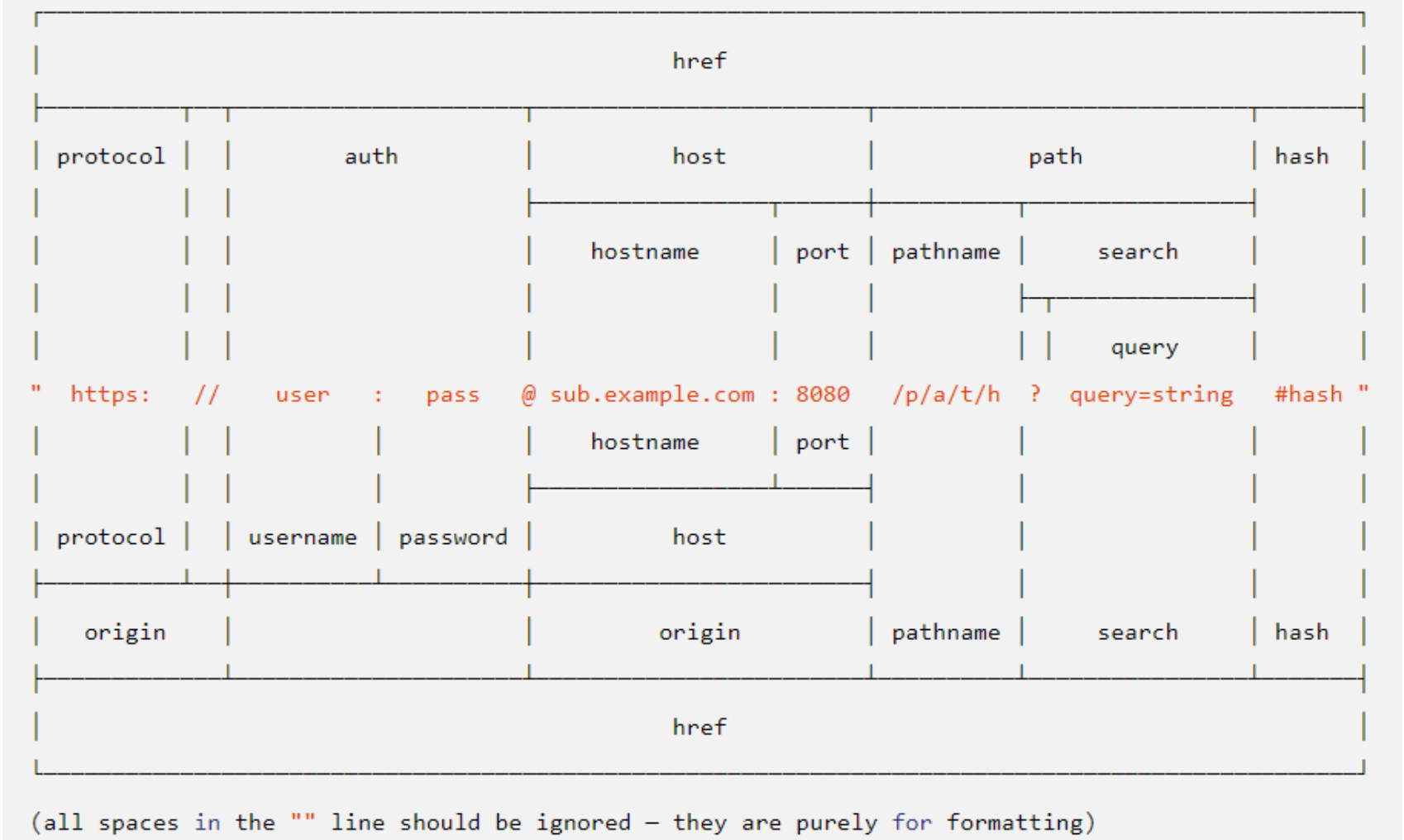
| 종류 | 설명 |
| --- | --- |
| basename(path[,ext]) | 경로의 마지막 부분을 반환 |
| delimiter | 운영체제별 환경 변수 구분자 를 반환( 윈도우 : 세비콜론( ;) ) |
| dirname(path) | 파일이 위치한 폴더 경로를 반환 |
| extname(path) | 파일의 확장자를 반환 |

| | |
|---|---|
| format(pathObject) | pathObject ( dir, root, base, name, ext 프로퍼티 )를 사용해<br>경로 문자열을 반환 |
| isAbsolute(path) | 주어진 파일의 경로 유형을 확인( 절대경로 : true / 상대경로 : false ) |
| join([...paths]) | 문자열로 주어진 경로들을 합쳐서 하나의 경로로 반환 |
| parse(path) | format() 메소드와 반대로 문자열로된 경로를 pathObject로 반환 |
| sep | 운영체제별 경로 구분자를 반환 (윈도우 : 역슬러시(\) ) |

- 현재 파일에 대한 경로 정보
  - __filename : file 명을 포함한 절대 경로
  - __dirname : file 명을 제외한 절대 경로

▼ URL

- 용도
  - 인터넷 주소에 해당하는 url을 다루기 위한 모듈
  - url 모듈




- API 종류
  - 레거시 API *( 위쪽 )*
    - Node.js 전용 레거시를 위한 구분 방식 기반
    - require을 사용해 url 모듈을 호출한 후 parse() 함수로 객체를 반환받아 사용
  - WHATWG API *(아래쪽)*
    - WHATWG(웹 표준을 정하는 단체)에서 정한 URL 구분 방식 기반
    - URL 클래스를 이용하여 new 연산자로 URL 객체를 생성해서 사용
      - QueryString을 조작하기 위해 URLSearchParams 클래스를 이용

▼ Crypto

- 용도
  - 다양한 암호화 기능을 제공
- 암호화 종류

        - 단방향 암호화
            - 복호화 불가능
        - 양방향 암호화 :
            - 복호화 가능
            - 구분
                - 대칭형 암호화
                    - 암호화 및 복호화 모두 동일한 키를 사용
                - 비대칭형 암호화
                    - 암호화 키와 복호화 시 다른 키를 사용

▼ File system

- 용도
    - 파일 읽기, 쓰기, 삭제 그리고 폴더 생성, 삭제 등과 같은 파일 처리 작업을 제공
- 메소드

| 종류 | 처리방식 | 설명 |
| --- | --- | --- |
| readFile(path, [options], callback) | 비동기 | 파일을 옵션으로 지정한 문자 인코딩을 사용해서 읽은 후 결과를 콜백함수로 전달 |
| readFileSync(path, [options]) | 동기 | 파일을 옵션으로 지정한 문자 인코딩을 사용해서 읽은 후 결과를 반환 |
| writeFile(path, data, [options], callback) | 비동기 | 파일을 옵션에서 지정한 방식을 사용해서 데이터를 쓰고 콜백 함수로 결과를 전달 |
| writeFileSync(path, data, [options]) | 동기 | 파일을 옵션에서 지정한 방식을 사용해서 데이터를 씀 |
| watchFile(filename[, options], listener) | | 대상이 되는 파일의 변경 사항 여부를 감시 |