



21.8.31 3-4-2 console ~ 3-5-1

3.4.2 console

console 객체는 보통 디버깅을 위해 사용된다.

변수 값이 제대로 들어 있나 확인하기도 하고 오류 메시지를 표시하기도 하고 코드 실행 시간을 확인할 때도 쓰인다.

```
const string = 'abc';
const number = 1;
const boolean = true;
const obj = {
  outside: {
    inside: {
      key: 'value',
    },
  },
};
console.time('alltime'); //timeEnd와 짝지어 시간을 측정한다.
console.log('평범한 로그, 쉽표로 구분해 여러 값 표시 가능');
console.log(string, number, boolean);
console.error('에러 메시지는 .error에 담는다.');
```

```
console.dir(obj, { colors: false, depth: 2}); //객체를 표시, colors는 보기 좋게 하고
console.dir(obj, { colors: true, depth: 1}); //depth는 표시할 깊이 표시

console.time('시간 측정');
for(let i =0; i<10000;i++){
  continue;
}
console.timeEnd('시간 측정');
```

```
function b(){
  console.trace('에러 위치 추적');//에러가 어디서 났는지 추적 가능
}
function a(){
  b();
}
a();

console.timeEnd('alltime');
```

```

PS C:\Users\LENOVO\Desktop\study\studyNodeJS\21-8-31> node console
평범한 로그, 쉽표로 구분해 여러 값 표시 가능
abc 1 true
에러 메세지는 .error에 담는다.
{ outside: { inside: { key: 'value' } } }
{ outside: { inside: [Object] } }
시간 측정: 0.244ms
Trace: 에러 위치 추적
  at b (C:\Users\LENOVO\Desktop\study\studyNodeJS\21-8-31\console.js:25:13)
  at a (C:\Users\LENOVO\Desktop\study\studyNodeJS\21-8-31\console.js:28:5)
  at Object.<anonymous> (C:\Users\LENOVO\Desktop\study\studyNodeJS\21-8-31\console.js:30:1)
  at Module._compile (internal/modules/cjs/loader.js:1072:14)
  at Object.Module._extensions..js (internal/modules/cjs/loader.js:1101:10)
  at Module.load (internal/modules/cjs/loader.js:937:32)
  at Function.Module._load (internal/modules/cjs/loader.js:778:12)
  at Function.executeUserEntryPoint [as runMain] (internal/modules/run_main.js:76:12)
  at internal/main/run_main_module.js:17:47
alltime: 10.694ms
PS C:\Users\LENOVO\Desktop\study\studyNodeJS\21-8-31>

```

3.4.3 타이머

global객체 안에 들어 있는 타이머 기능 제공 함수로 **setTimeout**, **setInterval**, **setImmediate**가 있다.

- **setTimeout**(콜백 함수, 밀리초) : 주어진 밀리초 이후에 콜백 함수를 실행한다.
- **setInterval**(콜백 함수, 밀리초) : 주어진 밀리초마다 콜백 함수를 반복 실행한다.
- **setImmediate**(콜백 함수) : 콜백 함수를 즉시 실행한다.

이 타이밍 함수들은 아이디를 반환하게 된다. 이 아이디를 이용해 타이머를 취소할 수 있다.

- **clearTimeout**(아이디) : **setTimeout**을 취소한다.
- **clearInterval**(아이디) : **setInterval**을 취소한다.
- **clearImmediate**(아이디) : **setImmediate**를 취소한다.

이를 이용해 예제를 만들면

```

const timeout = setTimeout(()=>{
  console.log('1.5초 후 실행');// 1번
}, 1500);

```

```

const interval = setInterval(()=>{
  console.log('1초마다 실행'); //2번
}, 1000);

const timeout2 = setTimeout(()=> {
  console.log('실행되지 않습니다. '); //3번
}, 3000);

setTimeout(()=>{ // 4번
  clearTimeout(timeout2);
  clearInterval(interval);
}, 2500);

const immediate = setImmediate(()=>{//5번
  console.log('즉시 실행');
});

const immediate2 = setImmediate(()=>{//6번
  console.log('실행되지 않습니다. ');
});

clearImmediate(immediate2); //7번

```

7번으로 인해 immediate2(6번)가 취소 됐으므로 5번이 먼저 실행된다.

—>>'즉시 실행'

1초가 지난 뒤 interval(2번)이 실행됨.

—>> '1초마다 실행'

1.5초가 지난 뒤 timeout(1번)이 실행됨.

—>> '1.5초 후 실행'

2초가 지난 뒤 interval(2번)이 실행됨.

—>> '1초 마다 실행'

2.5초가 지난 뒤 setTimeout(4번)으로 timeout2(3번) 취소, interval(2번) 취소

```

PS C:\Users\LENOVO\Desktop\study\studyNodeJS\21-8-31> node timer
즉시 실행
1초마다 실행
1.5초 후 실행
1초마다 실행
PS C:\Users\LENOVO\Desktop\study\studyNodeJS\21-8-31> 

```

3.4.4 __filename, __dirname

노드는 파일 사이에 모듈 관계인 경우가 많아 현재 파일의 경로나 파일명을 알아야 하는 경우가 있다.

노드는 __filename과 __dirname이라는 키워드로 경로에 대한 정보를 제공한다.

```
console.log(__filename); //파일 이름을 출력  
console.log(__dirname); //해당 디렉토리 출력
```

```
PS C:\Users\LENOVO\Desktop\study\studyNodeJS\21-8-31> node filename  
C:\Users\LENOVO\Desktop\study\studyNodeJS\21-8-31\filename.js  
C:\Users\LENOVO\Desktop\study\studyNodeJS\21-8-31
```

3.4.5 module, exports

여태까지 module.exports만 사용했지만 exports 객체로도 모듈을 만들 수 있다.

```
exports.odd = '홀수입니다'  
exports.even = '짝수입니다'
```

위의 코드처럼 exports 객체에 넣게 되면 자동으로 module.exports로 참조되기 때문이다.

실제로 console.log(module.exports===exports)를 하면 true가 나온다.



exports와 module.exports에는 참조 관계가 없으므로 한 모듈에 exports 객체와 module.exports를 동시에 사용하지 않는것이 좋다.

3.4.6 process

process는 현재 실행되고 있는 노드 프로세스에 대한 정보를 담는다.

process 객체 안에는 다양한 속성이 있다.

터미널에 node만 쳐서 REPL을 쓸 수 있게 한 뒤, 실행해보자

```
PS C:\Users\LENOVO\Desktop\study\studyNodeJS\21-8-31> node //REPL 실행
Welcome to Node.js v14.17.5.           //REPL시작
Type ".help" for more information.
> process.version //설치된 노드의 버전
'v14.17.5'
> process.arch //프로세서 아키텍처 정보
'x64'
> process.platform // 운영체제 플랫폼 정보
'win32'
> process.pid //현재 프로세스의 아이디
5228
> process.uptime() //프로세스가 시작된 후 흐른 시간 (단위 : 초)
32.9566609
> process.execPath //노드의 경로
'D:\node.exe'
> process.cwd() //현재 프로세스가 실행되는 위치
'C:\Users\LENOVO\Desktop\study\studyNodeJS\21-8-31'
> process.cpuUsage() //현재 cpu 사용량
{ user: 390000, system: 265000 }
>
```

process.env와 **process.nextTick**, **process.exit()**은 중요하므로 따로 설명하겠다.

<process.env>

시스템의 환경변수를 보여준다.

```

> process.env
{
  ALLUSERSPROFILE: 'C:\\ProgramData',
  APPDATA: 'C:\\Users\\LENOVO\\AppData\\Roaming',
  ChocolateyInstall: 'C:\\ProgramData\\chocolatey',
  ChocolateyLastPathUpdate: '132748636777113115',
  CHROME_CRASHPAD_PIPE_NAME: '\\\\.\\pipe\\crashpad_8356_WQYJLYSZIICHXZD',
  CommonProgramFiles: 'C:\\Program Files\\Common Files',
  'CommonProgramFiles(x86)': 'C:\\Program Files (x86)\\Common Files',
  CommonProgramW6432: 'C:\\Program Files\\Common Files',
  COMPUTERNAME: 'DESKTOP-4S2L9JH',
  ComSpec: 'C:\\Windows\\system32\\cmd.exe',
  CUDA_PATH: 'C:\\Program Files\\NVIDIA GPU Computing Toolkit\\CUDA\\v11.0',
  CUDA_PATH_V11_0: 'C:\\Program Files\\NVIDIA GPU Computing Toolkit\\CUDA\\v11.0',
  DriverData: 'C:\\Windows\\System32\\Drivers\\DriverData',
  HOMEDRIVE: 'C:',
  HOMEPATH: '\\Users\\LENOVO',
  JAVA_HOME: 'C:\\Program Files\\Java\\jdk-15.0.1',
  LOCALAPPDATA: 'C:\\Users\\LENOVO\\AppData\\Local',
  LOGONSERVER: '\\\\DESKTOP-4S2L9JH',
  NUMBER_OF_PROCESSORS: '8',
  NVCUDASAMPLES11_0_ROOT: 'C:\\ProgramData\\NVIDIA Corporation\\CUDA Samples\\v11.0',
  NVCUDASAMPLES_ROOT: 'C:\\ProgramData\\NVIDIA Corporation\\CUDA Samples\\v11.0',
  NVTOOLSEXT_PATH: 'C:\\Program Files\\NVIDIA Corporation\\NvToolsExt\\',
  OneDrive: 'C:\\Users\\LENOVO\\OneDrive',
  ORIGINAL_XDG_CURRENT_DESKTOP: 'undefined',
  OS: 'Windows_NT',
  Path: 'C:\\Python39\\Scripts\\;C:\\Python39\\;C:\\Program Files\\NVIDIA GPU Computing'
}

```



환경변수란 운영체제에서 프로세스를 실행할 때 참조하는 변수이다.

process.env는 서비스의 중요한 키를 저장하는 공간으로도 사용된다.

서버나 DB의 비밀번호와 각종 API 키를 코드에 직접 입력하는 건 위험하기 때문에 다음과 같이 process.env의 속성으로 대체한다.

```

const secretId = process.env.SECRET_ID;
const secretCODE = process.env.SECRET_CODE;

```

<process.nextTick(콜백)>

process.nextTick은 이벤트 루프가 다른 콜백 함수들보다 nextTick의 콜백 함수를 우선으로 처리하도록 만든다.

```

setImmediate(()=>{
  console.log('immediate');
});
process.nextTick(()=>{
  console.log('nextTick');
});
setTimeout(()=>{
  console.log('timeout');
}, 0);
Promise.resolve().then(()=>console.log('Promise'));

```

먼저 process.nextTick이 실행되고 Promise가 실행된다.(Promise도 다른 콜백보다 우선시 됨.)

```

PS C:\Users\LENOVO\Desktop\study\studyNodeJS\21-8-31> node nextTick
nextTick
Promise
timeout
immediate

```



따라서 **process.nextTick**과 **Promise**를 **마이크로 태스크**라고 따로 구분지어 부른다.



process.nextTick으로 받은 콜백 함수나 **resolve된 Promise**는 다른 이벤트 루프에서 대기하는 콜백 함수보다도 먼저 실행된다. **Microtask**를 재귀 호출하면 이벤트 루프는 **다른 콜백 함수보다 우선하여 실행되므로** 다른 콜백 함수들이 실행되지 않을 수 있다.

<process.exit>

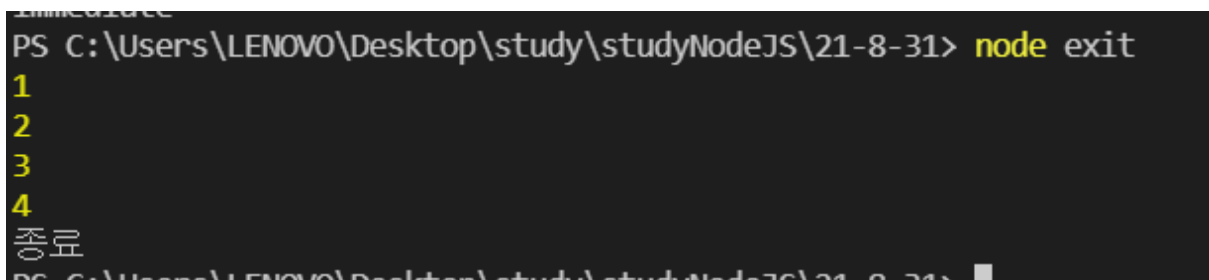
.exit는 실행 중인 프로세스를 종료한다.

서버에 이 함수를 실행하면 서버가 멈추므로 서버에는 거의 사용하지 않는다.

하지만 서버 외의 프로그램에서는 수동으로 노드를 멈추게 하기 위해 사용된다.

```
let i =1;
setInterval(()=>{
  if(i===5){
    console.log('종료');
    process.exit();
  }
  console.log(i);
  i+=1;
},1000);
```

i가 5가 되면 바로 프로세스를 종료한다.



```
PS C:\Users\LENOVO\Desktop\study\studyNodeJS\21-8-31> node exit
1
2
3
4
종료
PS C:\Users\LENOVO\Desktop\study\studyNodeJS\21-8-31>
```

process.exit는 인자로 코드번호를 줄 수 있다.



정상 종료 : 인자X or 0

비정상 종료 : 1

3.5 노드 내장 모듈 사용하기

위에까지는 내장 객체였고 지금부터는 내장 모듈을 사용한다.

노드는 웹 브라우저에서 사용되는 자바스크립트보다 더 많은 기능을 제공한다.

3.5.1 os

웹 브라우저에 사용되는 자바스크립트는 운영체제의 정보를 가져올 수 없지만 노드는 **os 모듈에 정보가 담겨져 있기 때문에** 정보를 가져올 수 있다.


```

const os =require('os');

console.log('운영체제 정보-----');
console.log('os.arch(): ', os.arch());
console.log('os.platform() : ', os.platform());
console.log('os.type()', os.type()); //운영체제 종류
console.log('os.uptime()', os.uptime()); //운영체제 부팅 후 흐른 시간
console.log('os.hostname()', os.hostname()); //컴퓨터의 이름
console.log('os.release()', os.release()); //운영체제의 버전
console.log('경로-----')
console.log('os.homedir()', os.homedir()); //홈 디렉터리 경로
console.log('os.tmpdir()', os.tmpdir()); //임시 파일 저장 경로

console.log('cpu 정보-----')
console.log('os.cpus()', os.cpus()); //컴퓨터의 코어의 정보
console.log('os.cpus().length', os.cpus().length); //코어 개수 확인

console.log('메모리 정보-----')
console.log('os.freemem()', os.freemem()); //사용가능한 메모리
console.log('os.totalmem()', os.totalmem()); //전체 메모리 용량

```

```

PS C:\Users\LENOVO\Desktop\study\studyNodeJS\21-8-31> node os
운영체제 정보-----
os.arch(): x64
os.platform() : win32
os.type() Windows NT
os.uptime() 458663
os.hostname() DESKTOP-4S2L9JH
os.release() 10.0.19043
경로-----
os.homedir() C:\Users\LENOVO
os.tmpdir() C:\Users\LENOVO\AppData\Local\Temp
cpu 정보-----
os.cpus() [
  {
    model: 'Intel(R) Core(TM) i5-10300H CPU @ 2.50GHz',
    speed: 2496,
    times: {
      user: 4615390,
      nice: 0,
      sys: 4828171,
      idle: 124653906,
      irq: 1668156
    }
  },
  {
    model: 'Intel(R) Core(TM) i5-10300H CPU @ 2.50GHz',
    speed: 2496,
    times: {

```

os모듈은 주로 컴퓨터 내부 자원에 빈번하게 접근하는 경우 사용된다.

운영체제별로 다른 서비스를 제공하고 싶을 때 os 모듈이 유용할 것이다.