



# 21.8.30 3.1 REPL 사용하기 ~

## 3.1 REPL 사용하기

자바스크립트는 스크립트 언어이므로 미리 컴파일 하지 않아도 즉석에서 실행할 수 있다!

노드도 비슷한 콘솔 제공 → REPL(Read Eval Print Loop)

코드를 읽고(Read), 해석하고(Eval), 결과물을 반환하기(Print), 종료할 때까지 반복(Loop) 한다.

```
$ node
```

node만 입력하면 사용할 수 있다.

종료하려면 **Ctrl+C**를 두 번 누르거나 **.exit**를 입력한다.

## 3.2 JS파일 실행하기

터미널에서 입력

```
node [파일 경로]
```

확장자(.js)는 생략가능!

## 3.3 모듈로 만들기

노드는 코드를 모듈로 만들 수 있다.



### 모듈 : 특정한 기능을 하는 함수나 변수들의 집합

```
const odd = '홀수입니다';
const even = '짝수입니다';

module.exports = {
  odd,
  even,
};
```

var.js 처럼 변수를 선언해주고 다른 파일에서 불러오면 exports에 대입된 값을 사용할 수 있다.

```
const {odd, even} = require('./var');

function checkOddOrEven(num) {
  if(num%2){
    return odd;
  }
  return even;
}

module.exports = checkOddOrEven;
```

var.js에서 받아온 변수를 사용해 함수를 만들어 그 함수를 다시 module.exports에 대입했다.

```
const {odd, even} = require('./var'); //var.js 참조
const checkNumber = require('./func'); //func.js 참조

function checkStringOddOrEven(str){ //문자열 길이에 따라 짝수, 홀수 구별
  if(str.length%2){
    return odd;
  }
}
```

```

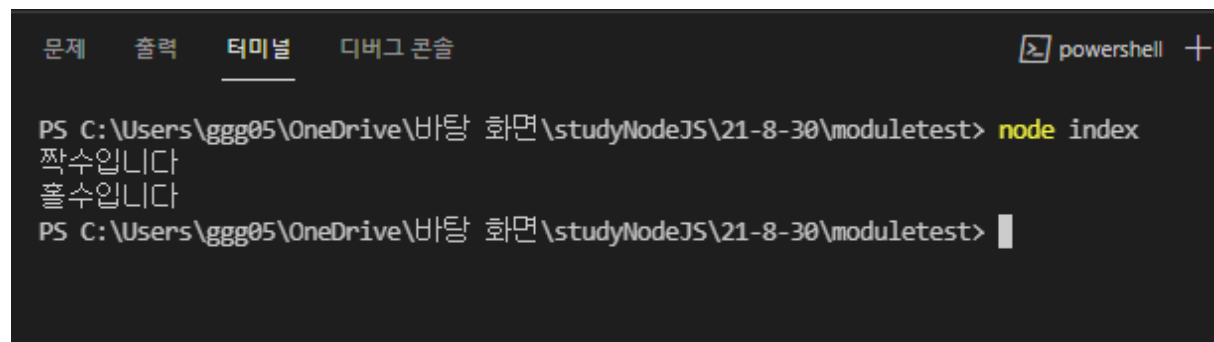
    return even;
}

console.log(checkNumber(10));
console.log(checkStringOddOrEven('Hello'));

```

코드가 완성됐으니 터미널로 파일을 열어본다.

```
$ node index
```



The screenshot shows a PowerShell terminal window with the following content:

```

문제   출력   터미널   디버그 콘솔
PS C:\Users\ggg05\OneDrive\바탕 화면\studyNodeJS\21-8-30\moduletest> node index
짝수입니다
홀수입니다
PS C:\Users\ggg05\OneDrive\바탕 화면\studyNodeJS\21-8-30\moduletest>

```

ES2015가 되면서 새로운 문법이 생겼다.

```

import { odd, even } from "./var"; //require에서 from으로 바뀌었다.

function checkOddOrEven(num){
    if(num%2){
        return odd;
    }
    return even;
}

export default checkOddOrEven; //module.exports에서 default로 바뀌었다.

```



모듈 확장자는 **js**를 쓰지 않고, **mjs**를 쓰고,  
실행 시 **node —experimental-modules [파일명]** 의 옵션을 붙여야 한다.

## 3.4 노드 내장 객체 알아보기

### 3.4.1 global

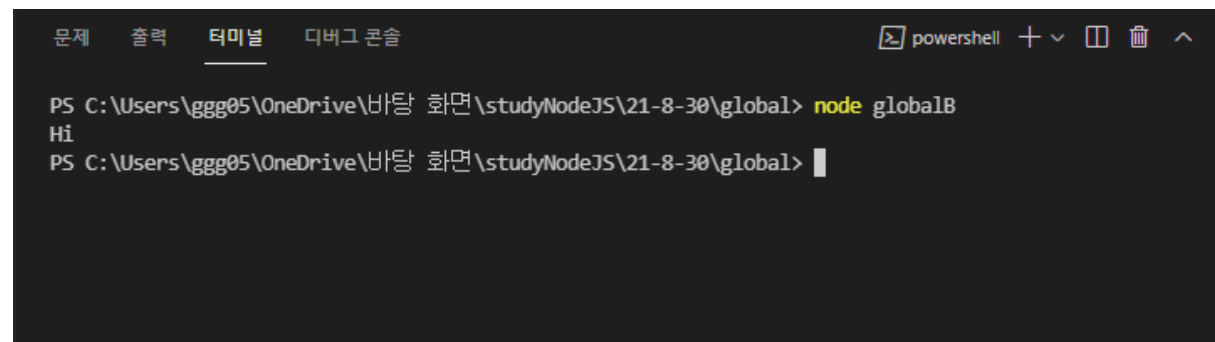
global은 전역 객체이므로 모든 파일에서 접근할 수 있다.

기본적으로 내장되어 있는 객체이기 때문에 global.require이라고 했던 것도 생략 가능한 것이다.

어떤 파일이든 접근 가능하기 때문에 다음과 같은 예제도 가능하다

```
module.exports={()=>global.message;
```

```
const A = require('./globalA');  
  
global.message = 'Hi';  
console.log(A())
```



```
문제 출력 터미널 디버그 콘솔 powershell + v [] [X] ^  
PS C:\Users\ggg05\OneDrive\바탕 화면\studyNodeJS\21-8-30\global> node globalB  
Hi  
PS C:\Users\ggg05\OneDrive\바탕 화면\studyNodeJS\21-8-30\global> 
```

로깅되는 건 globalA의 message였고 Hi는 globalB에서 했지만 전역객체이기 때문에 두 파일에서 접근 가능하다.



global에 값을 대입해 공유해서 쓸 수 있지만 프로그램 규모가 커질수록 어떤 값을 대입했는지 찾기 힘들어 유지보수에 어려움을 겪게 된다.