

생활코딩

Node.js 노드제이에스 프로그래밍

웹 + DB

2024년 2학기

생활코딩

Node.js 노드제이에스 프로그래밍

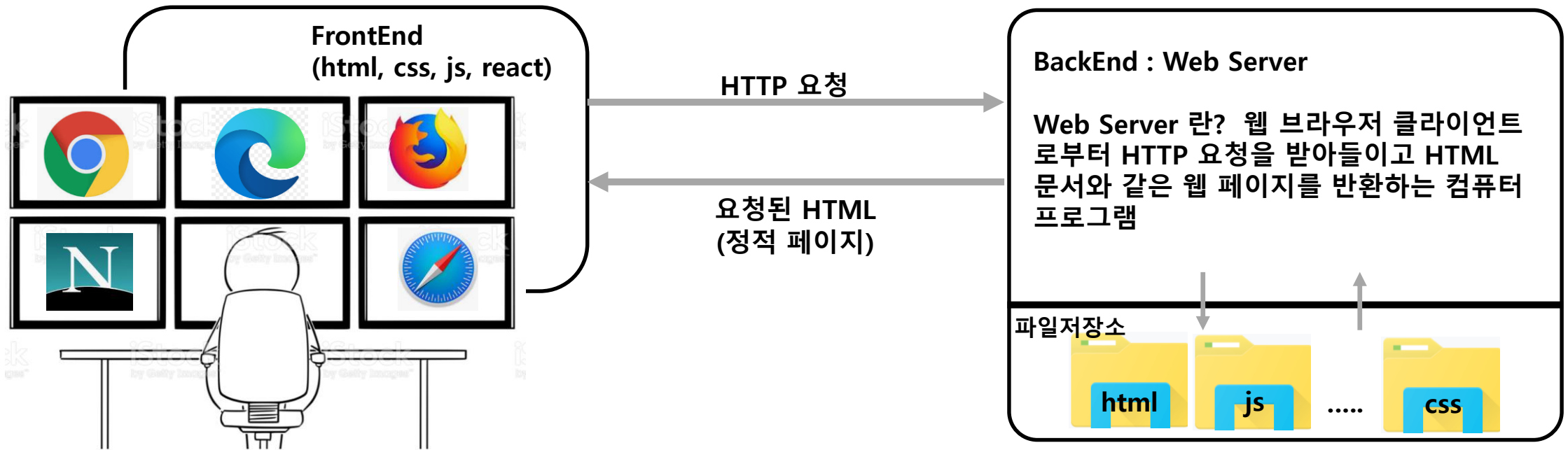
00. 웹프로그래밍 vs 웹DB프로그래밍

웹 : HTML, CSS

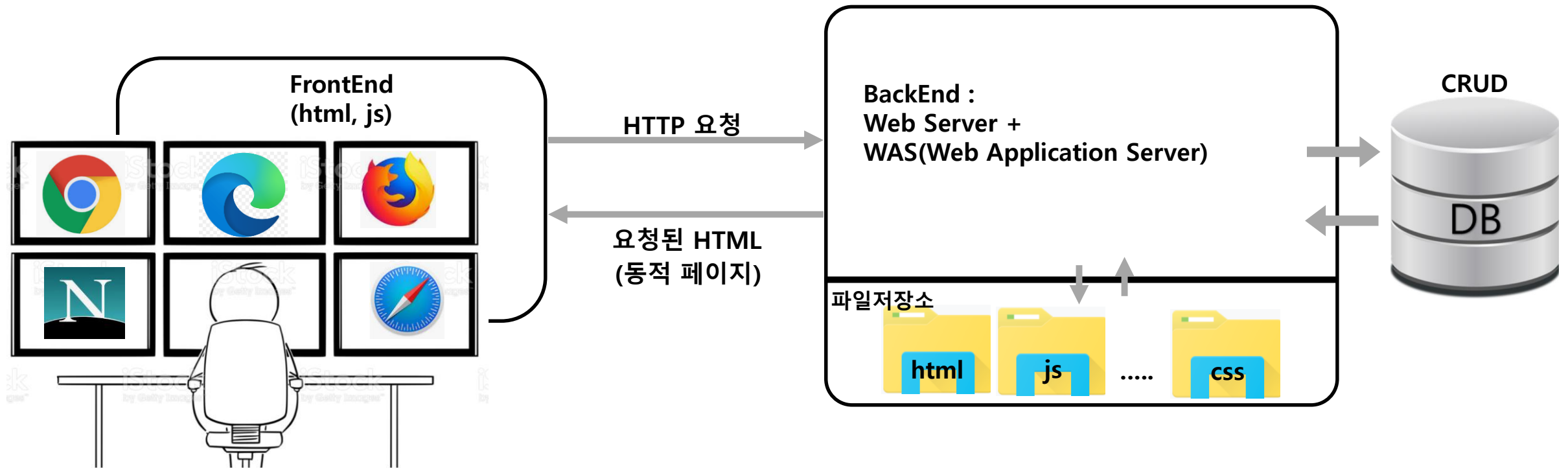
DB : MySQL, Oracle, Mongo DB , Vector DB

프로그래밍 : Node.js(Java Script), Django(Python), Spring(Java) ...

[웹데이터베이스 이전]

[참조] <https://mellonia-lab.tistory.com/73>

[웹데이터베이스 시대 : 클라이언트/서버]



[웹데이터베이스 시대 : 클라이언트/서버]

- WAS 란?

- 인터넷 상에서 HTTP 프로토콜을 통해 사용자 컴퓨터나 장치에 애플리케이션을 수행해 주는 미들웨어.
- 주로 동적 서버 콘텐츠를 수행하는 것으로 일반적인 웹서버와 구별이 되며, 주로 데이터베이스 서버와 같이 수행된다.
- 웹서버로는 처리할 수 없는 데이터베이스 조회나 다양한 로직 처리가 필요한 동적 콘텐츠를 제공함.
- 초창기 WAS 기능 : cgi 프로그램

- WAS와 연계된 언어

PHP, JSP, ASP, Python, Node.js, Django의 template 언어

- 대표적 WAS

Tomcat , Web sphere, Web Logic, Django

- 대표적 Web Server

Apach, IIS, Nginx

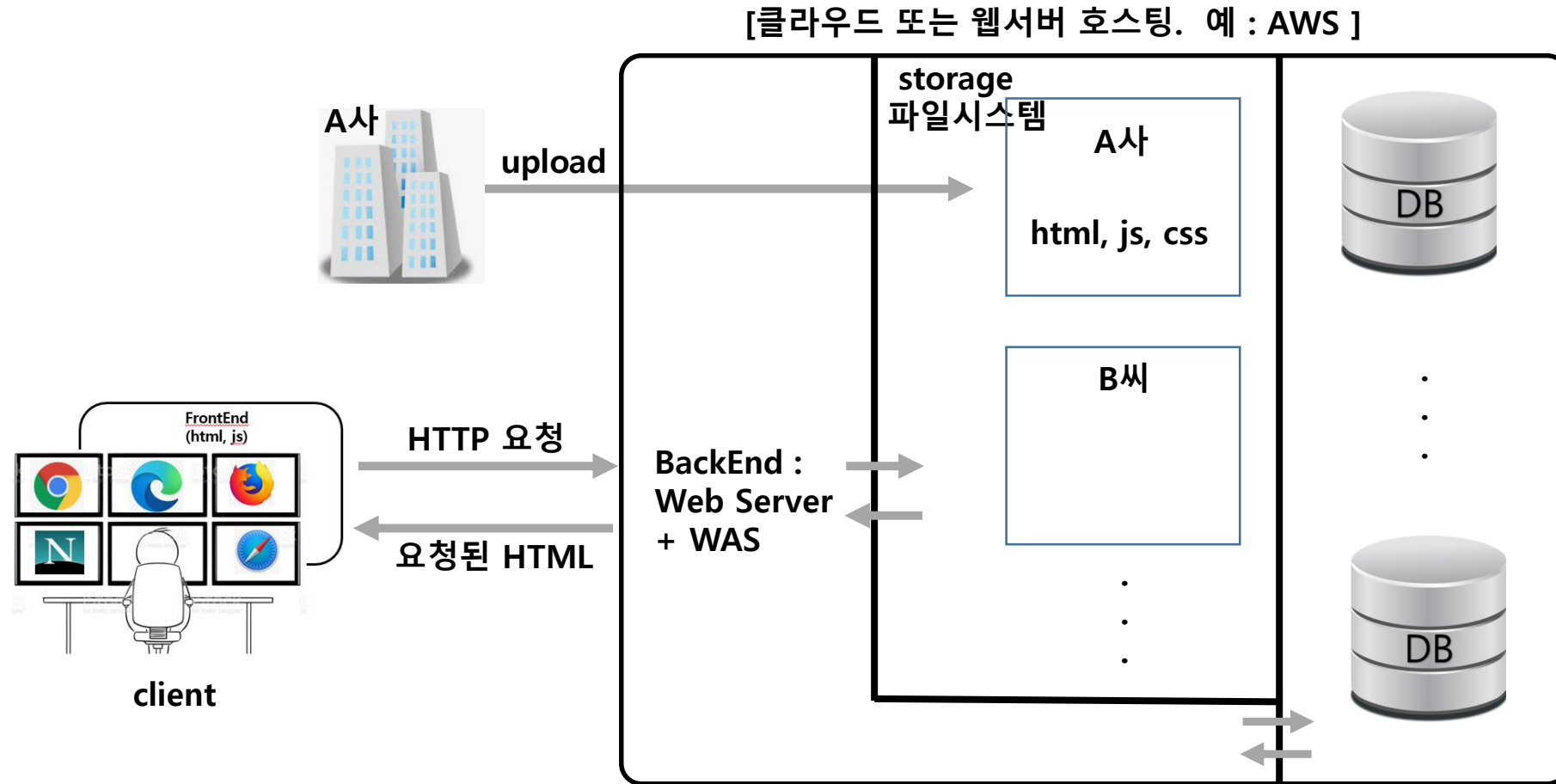
- WAS에 웹서버가 포함된 경우도 있지만 대규모 웹 서비스를 제공한다면 WAS와 웹 서버를 별도로 사용

[웹데이터베이스 시대 : 클라이언트/서버]

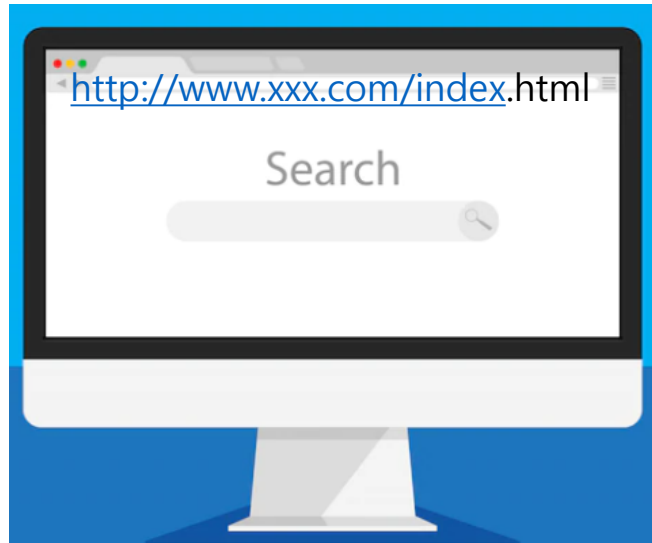
WAS, WEB Server , 기반 언어

WEB Server	WAS	기반 언어	DB
Node.js	Node.js(Express)	JavaScript	Oracle MySQL MongoDB
Python	Django, Flask	Python	
Apach	Tomcat(JSP)	Java, PHP	
IIS	ASP	ASP 템플릿 언어	
Nginx			

[웹데이터베이스 시대 : 클라우드 시스템]



웹 프로그래밍



request

인터넷망

response

※ 웹서버 기능

- ①도메인과 ip의 매핑 : DNS
- ②ip와 홈디렉토리 매핑
- ③URL에 있는 파일 찾기
- ④응답하기

※ 홈디렉토리 : URL에 매핑되는 root 디렉토리

웹서버(하드웨어)

<http://www.xxx.com> → 000.000.000.000

웹서버(소프트웨어)

/index.html

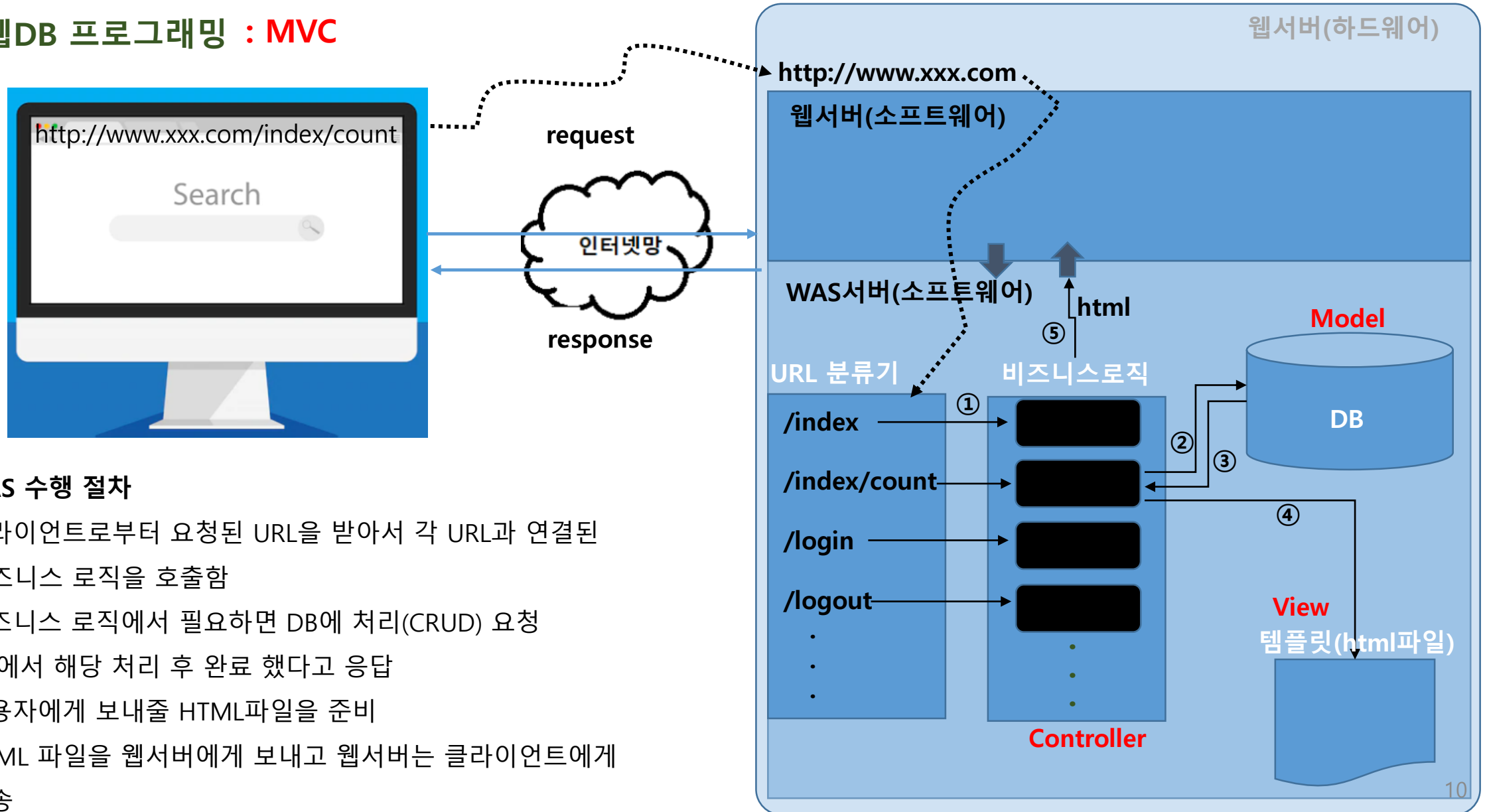
저장소(홈디렉토리)

```

<!doctype html>
<html>
<head>
  <title>WEB1 - HTML</title>
  <meta charset="utf-8">
</head>
<body>
  <h1><a href="05-index.html">WEB</a></h1>
  <ol>
    <li><a href="05-1.html">HTML</a></li>
    <li><a href="05-2.html">CSS</a></li>
    <li><a href="05-3.html">JavaScript</a></li>
  </ol>
  <h2>HTML</h2>
  <p><a href="https://www.w3.org/TR/html5/"
target="_blank" title="html5
speicification">Hypertext Markup Language
(HTML)</a>  
  </p><p style="margin-top:45px;">HTML elements are
the building blocks of HTML pages.  </p>
</body>
</html>

```

웹DB 프로그래밍 : MVC



- 0. 이 수업은 **HTML**과 자바스크립트 문법을 숙지했다고 가정
- 1. 웹서버 설치 - **Node.js** 설치, 에디터는 **VSCode** 사용
- 2. 웹서버 구동 및 사용하는 방법
- 3. **URL**에 따른 **Controller** 모듈 실행
- 4. **Controller(Node.js 프로그램)**와 **HTML**의 연동 방법
 - 1) **Node.js** 프로그램 내부에 **HTML**을 문자열로 작성
 - 2) **MVC** 모델
 - i. **HTML**에서 변수 사용하고
 - ii. **HTML**에서 자바스크립트 언어적인 요소 사용하기 → 템플릿 언어

Node.js의 Controller 프로그램(본 교재에서는 main.js)  **HTML**

5. Form 처리

- 1) **option, checkbox, input** 자료, 콤보 박스 등의 자료들을 **controller** 프로그램에 전달
- 2) **html** 의 **form**에서 넘어온 값에 따라 조건에 맞게 **controller** 로직 구성

6. DB 연결(CRUD)

- 1) **DB**로 부터 자료를 가져와 보여주기(**R**)
- 2) **DB**에 자료를 갱신하기(**U**)
- 3) **DB**로 부터 자료를 가져와서 자바스크립트 단에서 갱신하여 **DB**에 반영하기(**C**)
- 4) **DB** 자료를 삭제하기(**D**)
- 5) **Paging**기능

생활코딩

Node.js 노드제이에스 프로그래밍

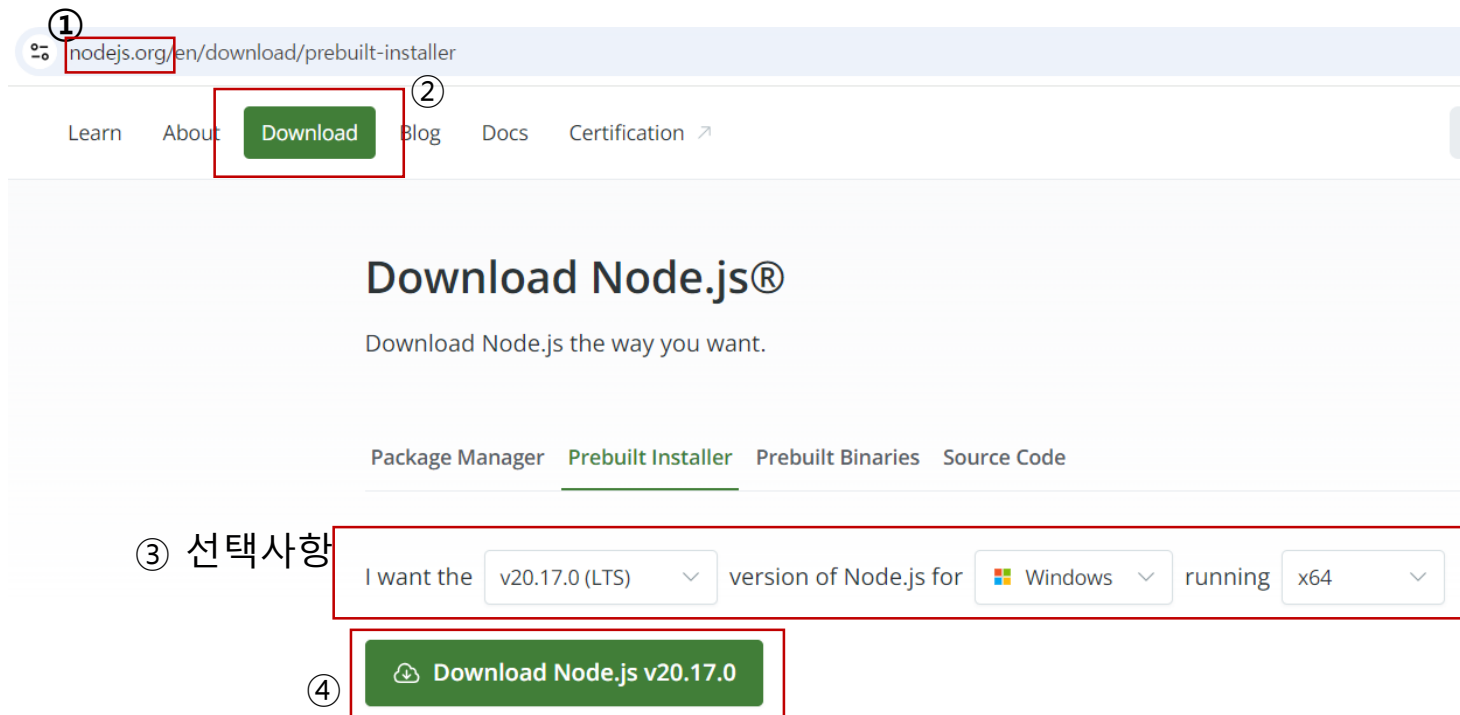
01. 자바스크립트와 Node.js

0. Node.js 설치

1) **nodejs.org** 사이트로 이동

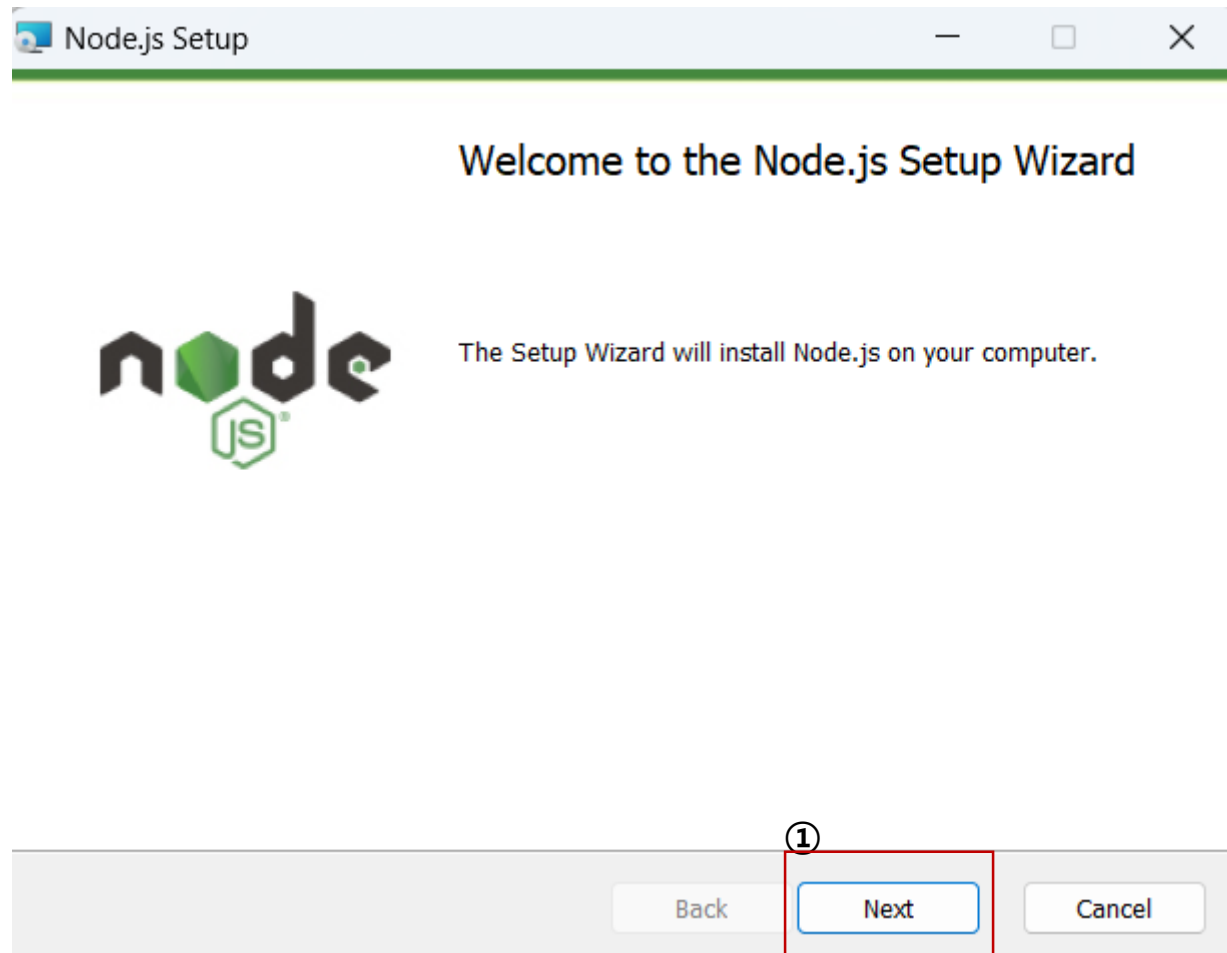
2) **Download** 메뉴 클릭

3)



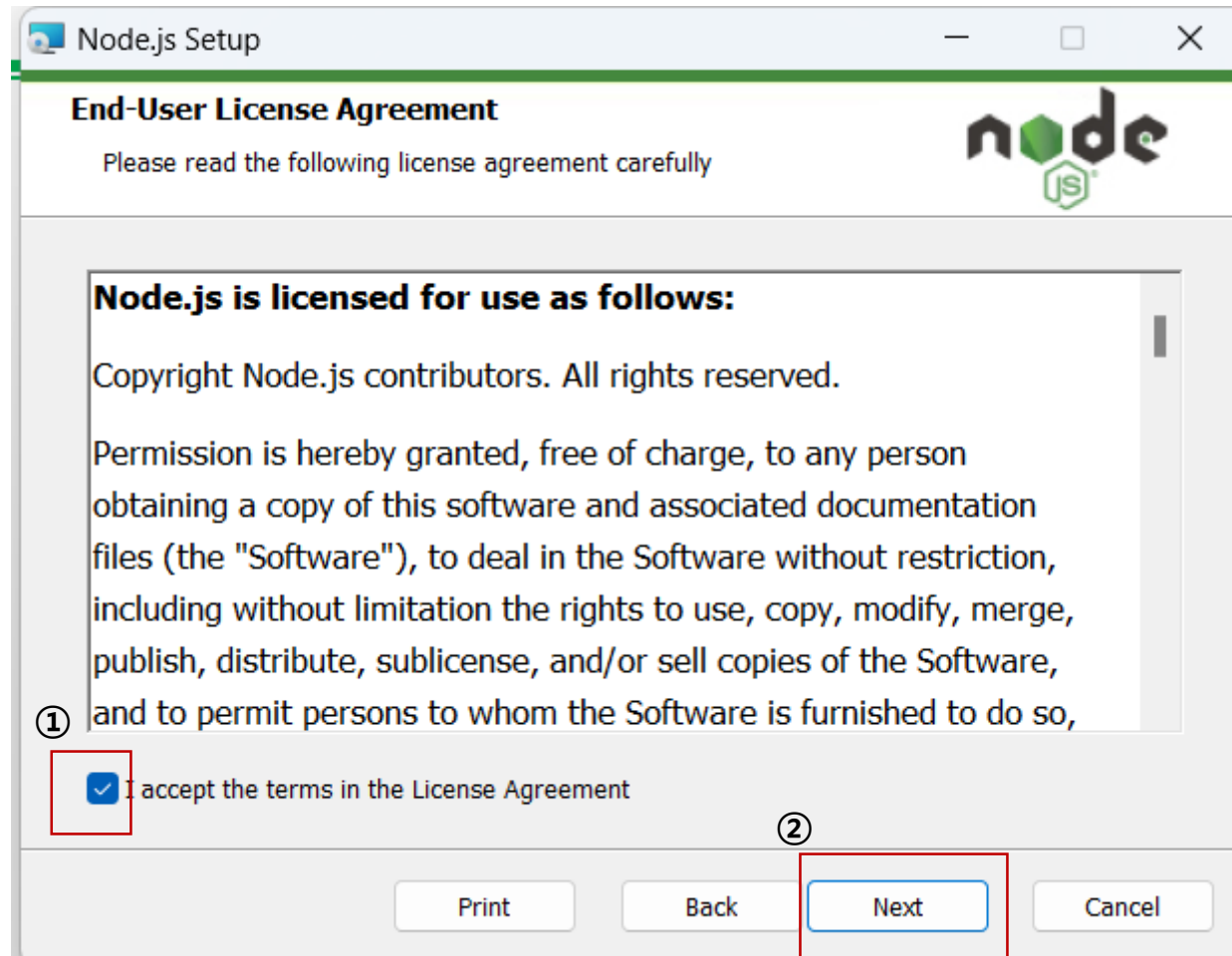
0. Node.js 설치

4) 다운로드 받은 파일 실행



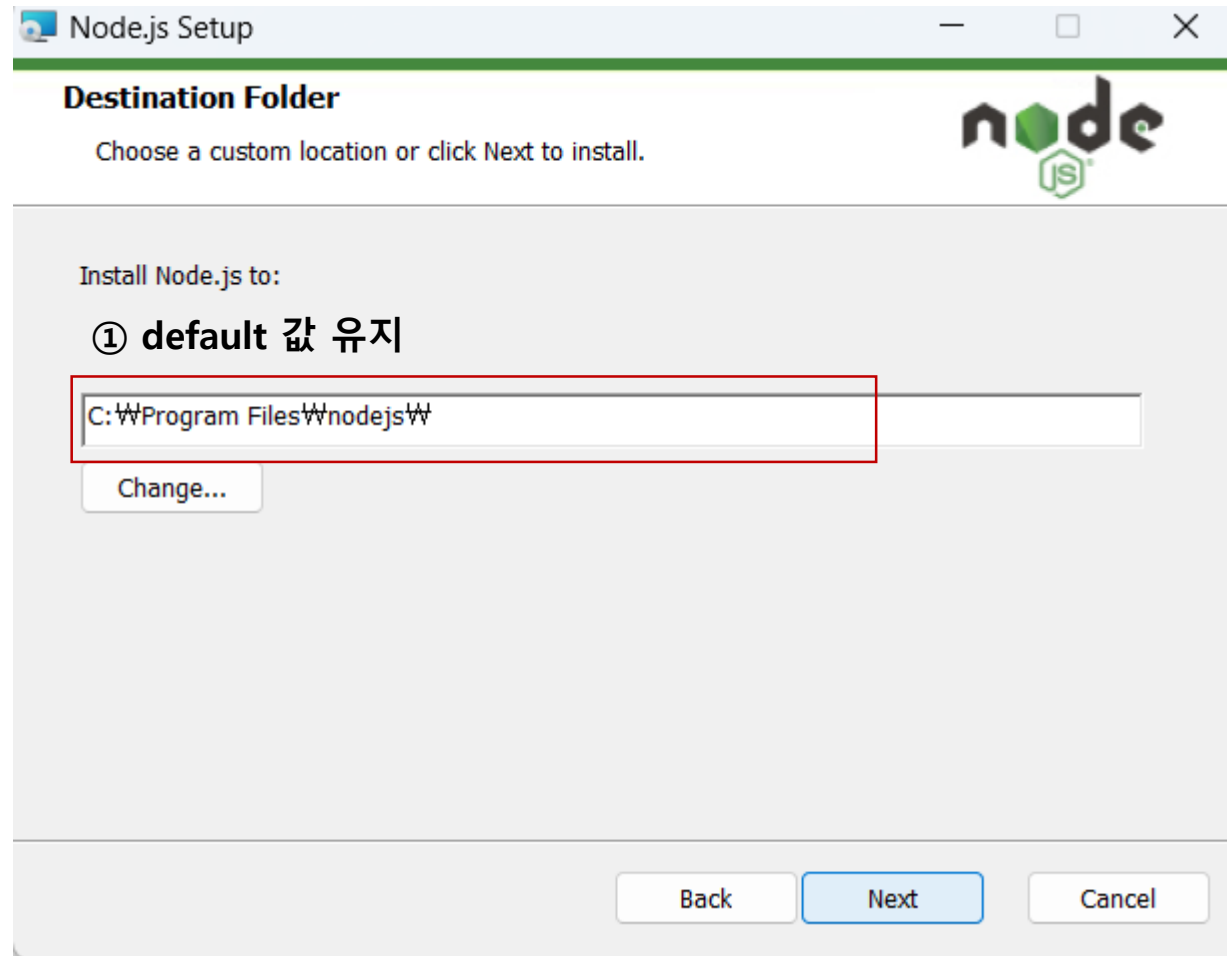
0. Node.js 설치

4) 다운로드 받은 파일 실행



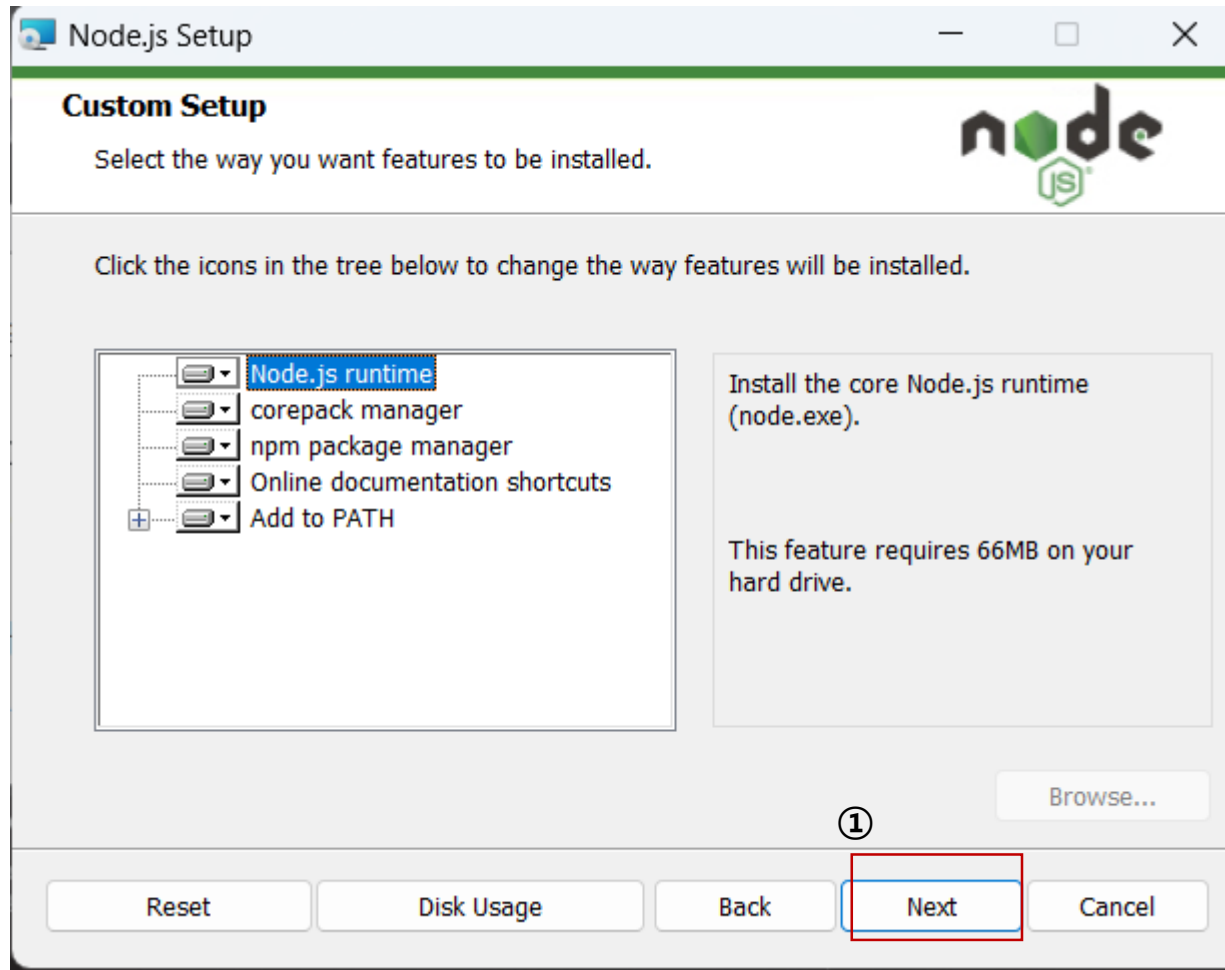
0. Node.js 설치

4) 다운로드 받은 파일 실행



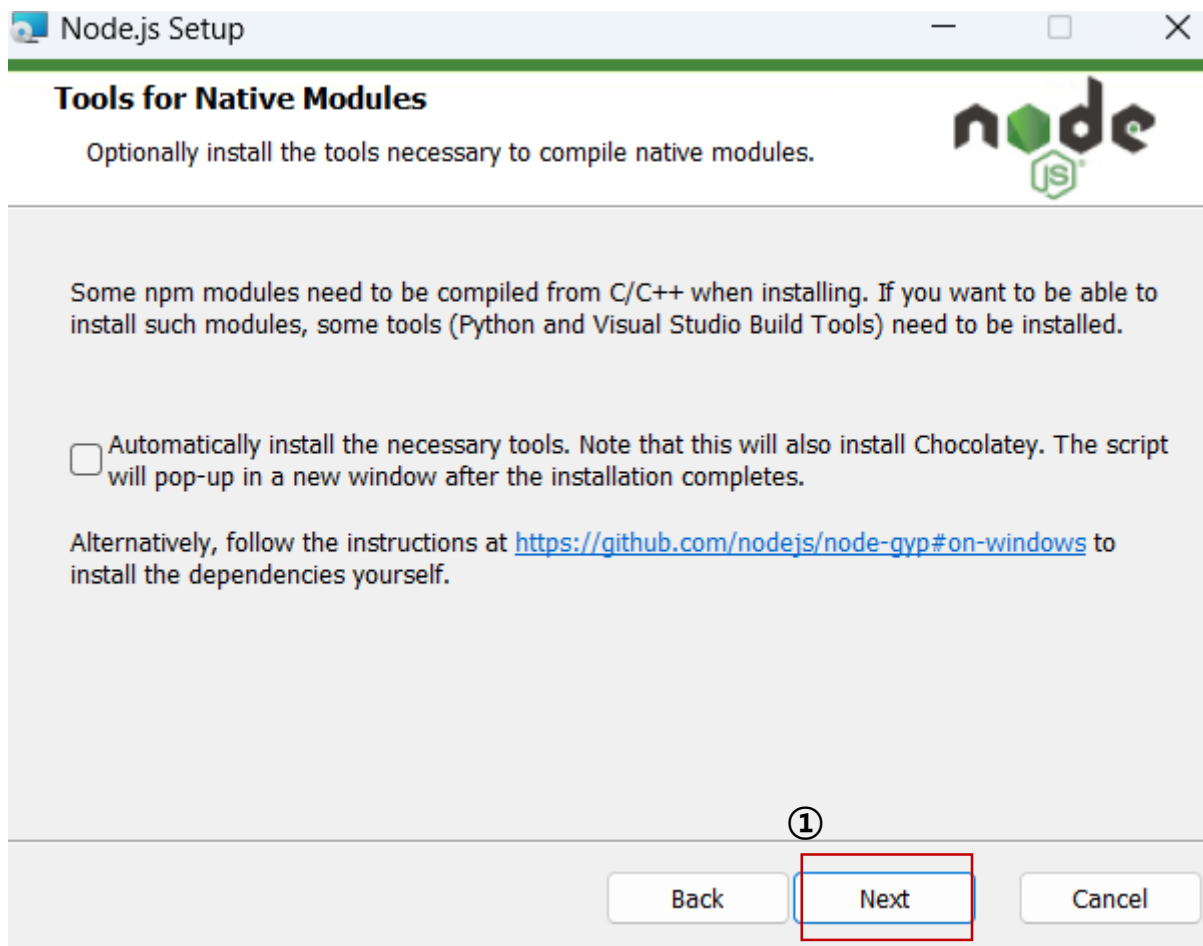
0. Node.js 설치

4) 다운로드 받은 파일 실행



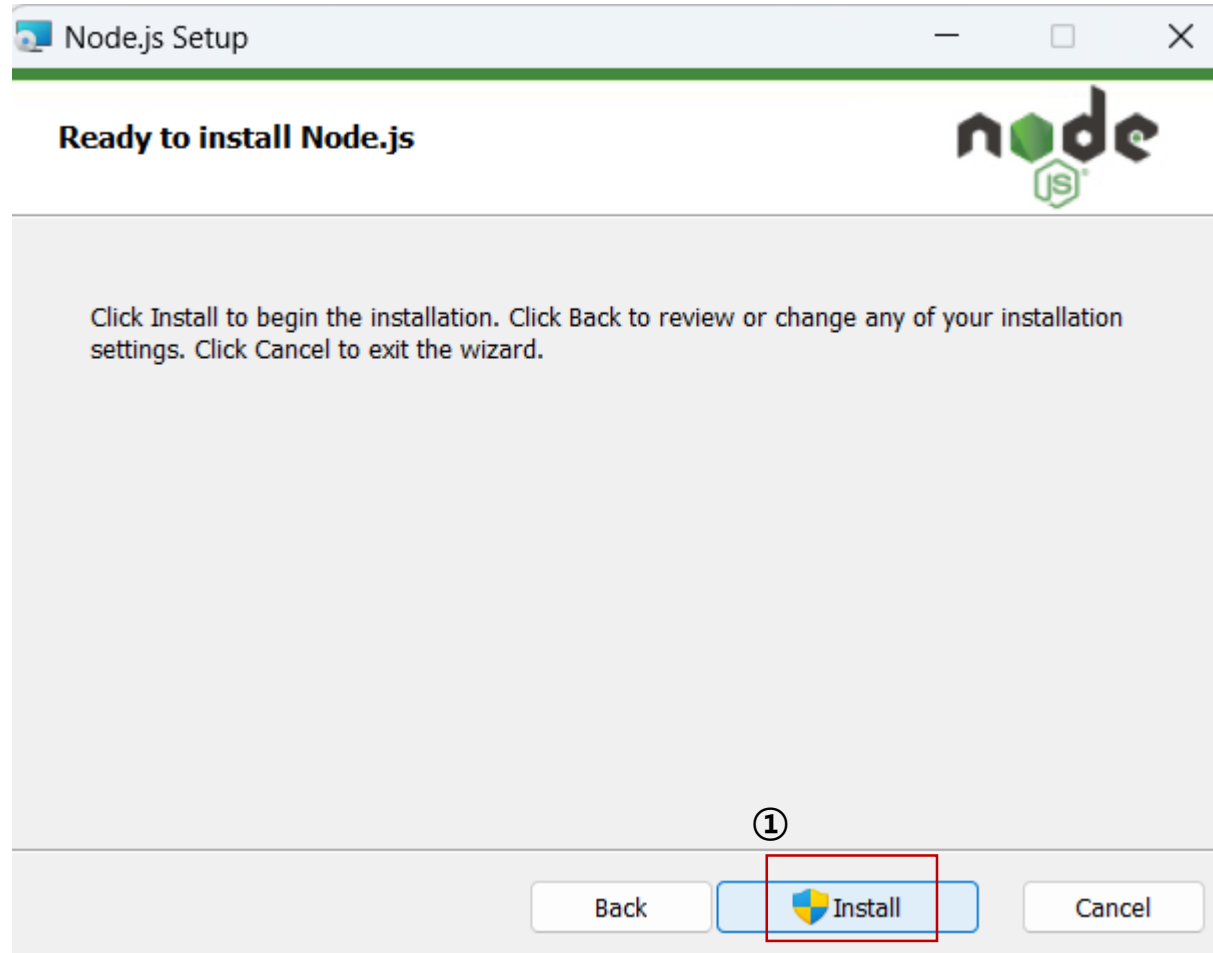
0. Node.js 설치

4) 다운로드 받은 파일 실행



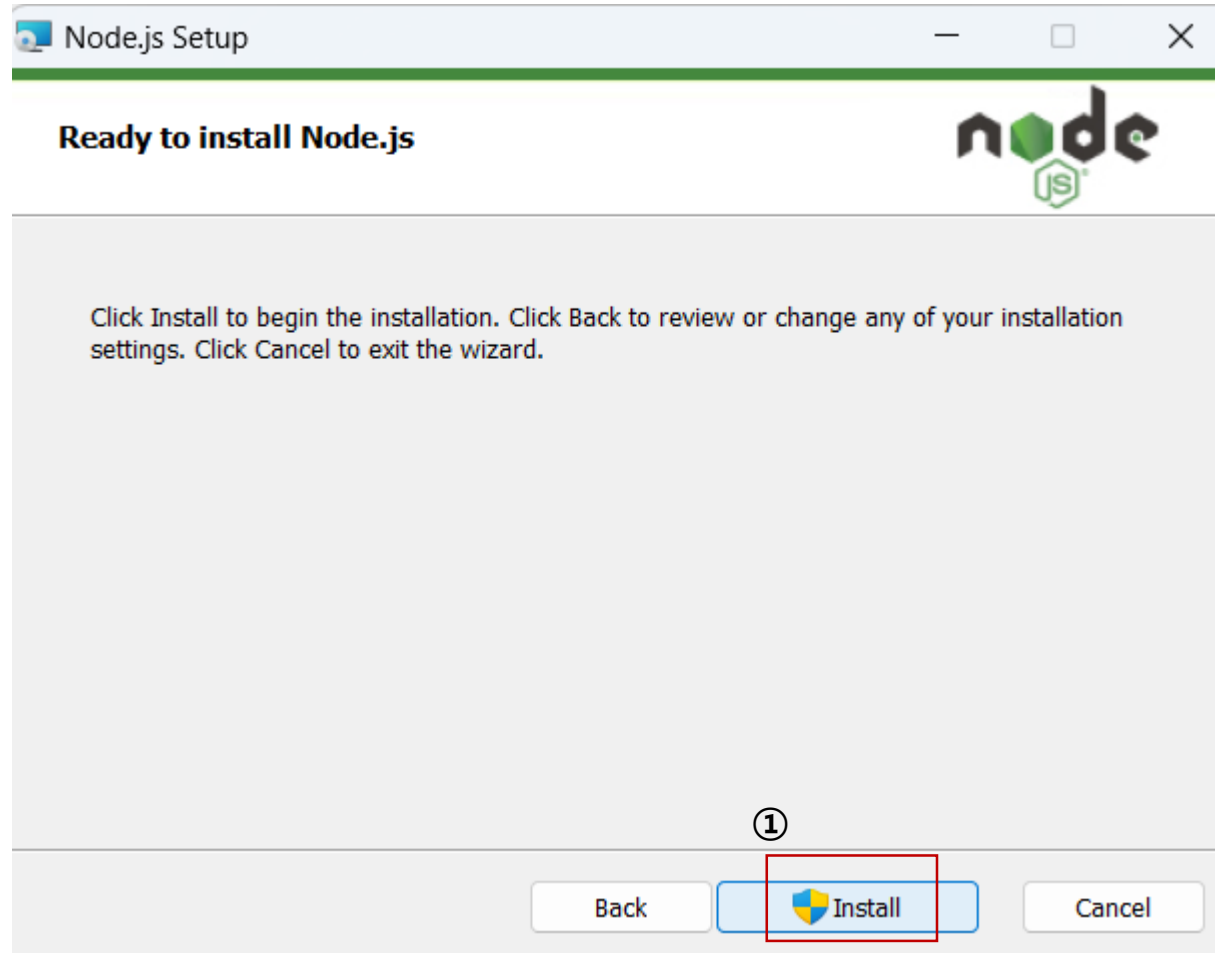
0. Node.js 설치

4) 다운로드 받은 파일 실행



0. Node.js 설치

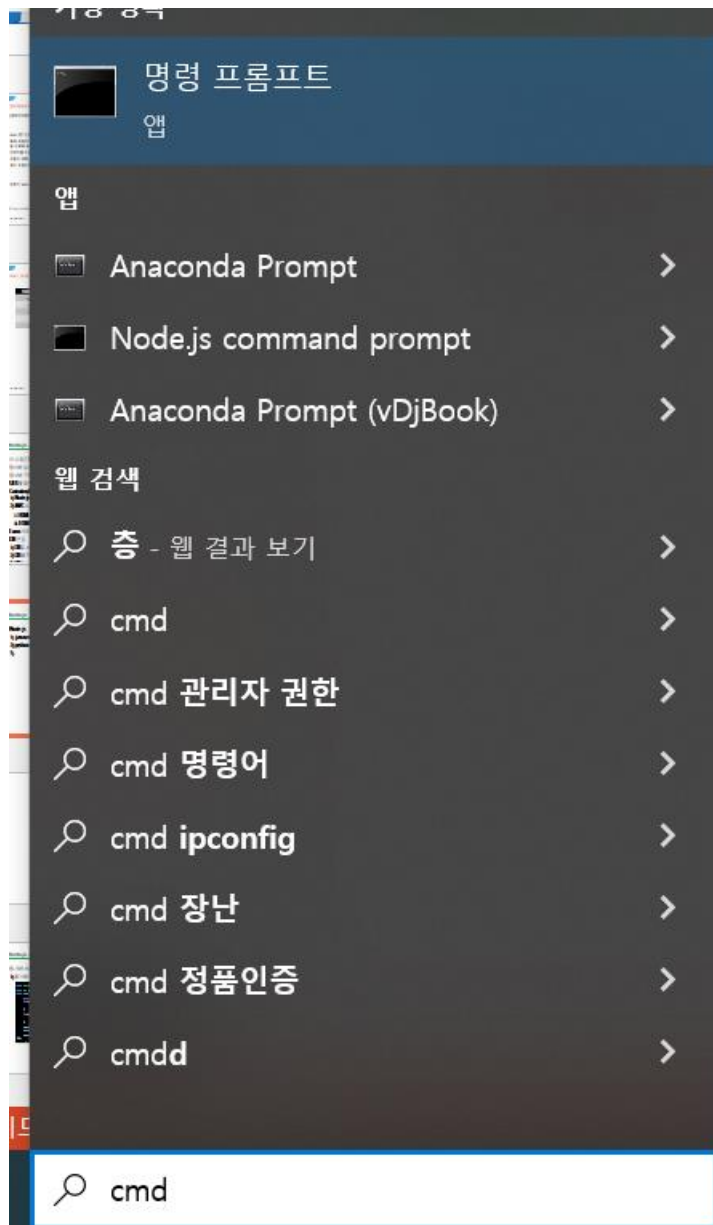
4) 다운로드 받은 파일 실행



0. Node.js 설치

5) 설치 확인

- * **REPL**: 인터프리터 기반의 프로그래밍 언어에서 사용되는 대화식 환경을 제공하는 도구. 셸과 유사
- * **run time**: 운영체제 위 또는 운영체제 자체에서 실행되면서 특정 프로그래밍 언어가 구동될 수 있는 환경
- * **interpreter**: 즉시 번역



명령 프롬프트 - node

```
Microsoft Windows [Version 10.0.18363.1082]
(c) 2019 Microsoft Corporation. All rights reserved.

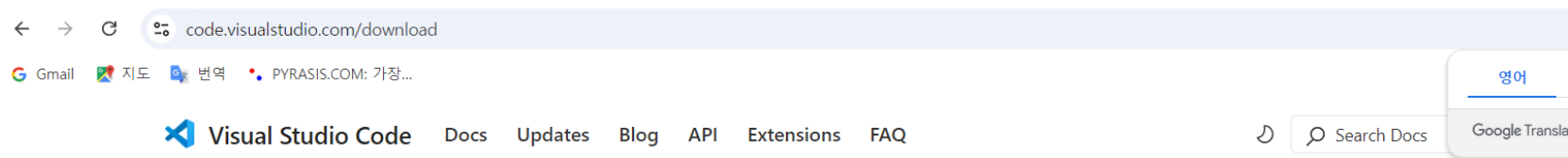
C:\Users\WBH>node
Welcome to Node.js v16.17.0.
Type ".help" for more information.
>
```

```
> console.log("hello")
hello
undefined
>
```

```
C:\Users\WBH\nodejs>node
Welcome to Node.js v16.17.0.
Type ".help" for more information.
> console.log("Hello ")
Hello
```

0. Visual Studio 설치

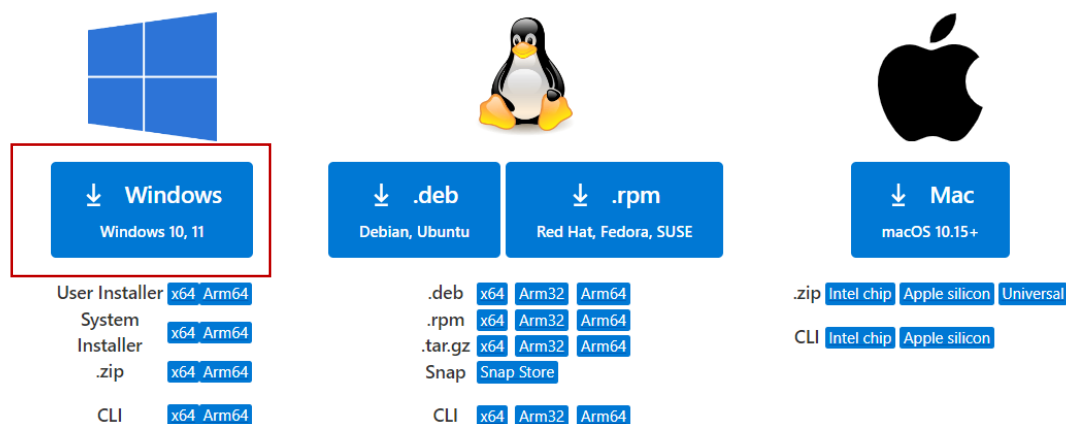
1) <https://code.visualstudio.com/download> : 사이트에서 다운로드



[Version 1.92](#) is now available! Read about the new features and fixes from July.

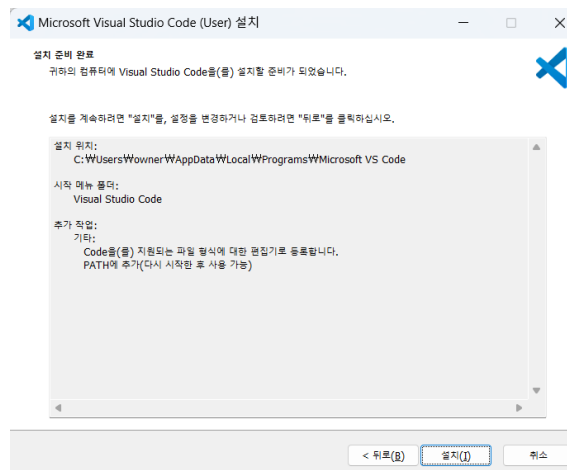
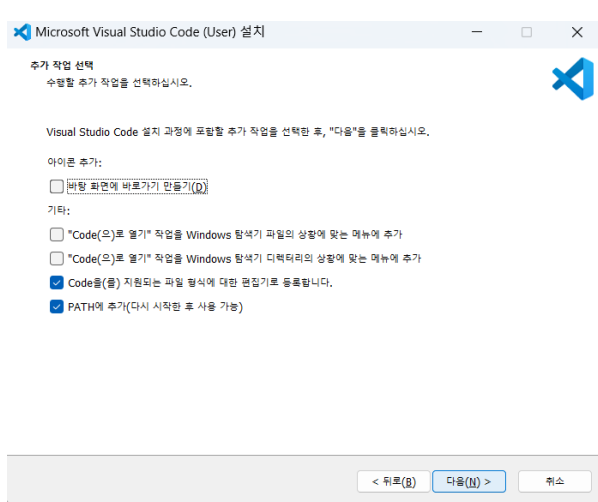
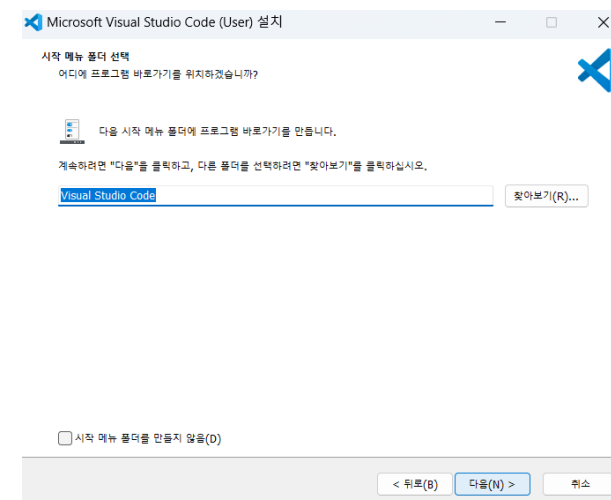
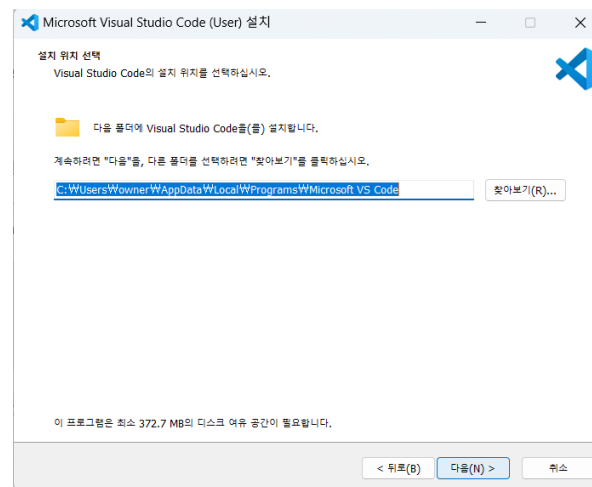
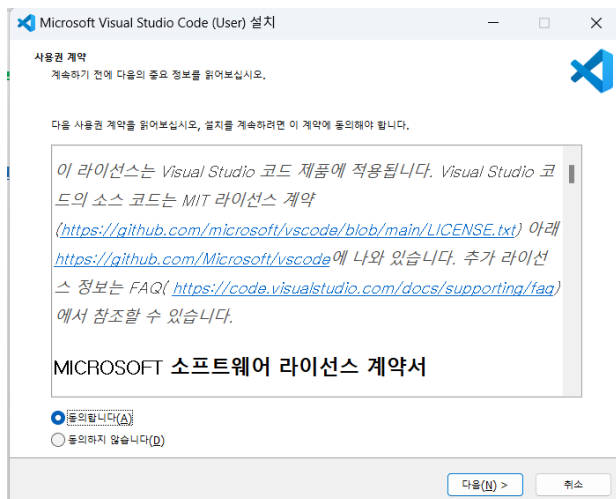
Download Visual Studio Code

Free and built on open source. Integrated Git, debugging and extensions.



0. Visual Studio 설치

1) <https://code.visualstudio.com/download> : 사이트에서 다운로드



1. Node.js의 탄생 배경

1) 1990년 웹이 처음 등장 → 초기 **HTML**만으로 웹페이지 구성 → 동적 **HTML** 작성

2) 자바스크립트의 변신

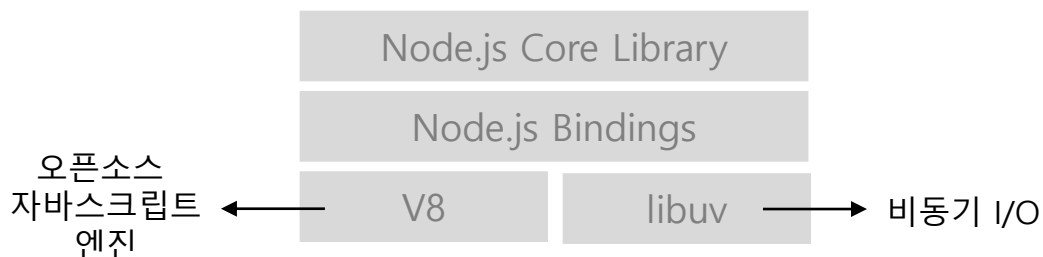
- * 넷스케이프가 브랜든 아이크에게 자바스크립트 제작의뢰(1995)
- * 초기 자바스크립트 : 웹 브라우저에 갇힌 편파적인 언어라는 혹평
- * 2008년 구글이 **V8** 엔진을 개발하고 오픈소스로 공개
- * 라이언 달(Ryan Dahl)이 **V8**엔진에 기반을 두고 **Node.js**를 제작 → 프로그래밍 언어로의 재탄생

2. Node.js의 특징

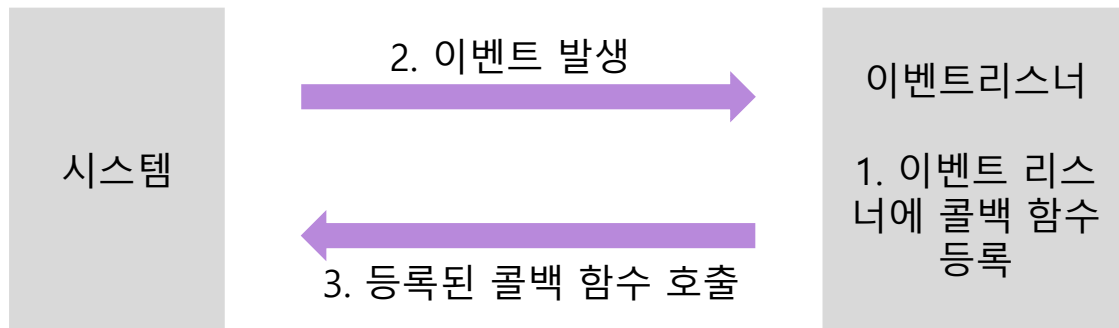
1) 서버의 기능

2) 런타임 기능 : 특정 언어로 만든 프로그램들을 실행할 수 있는 환경. 자바스크립트 실행기

3) **V8**과 더불어 **libuv**라는 라이브러리를 사용



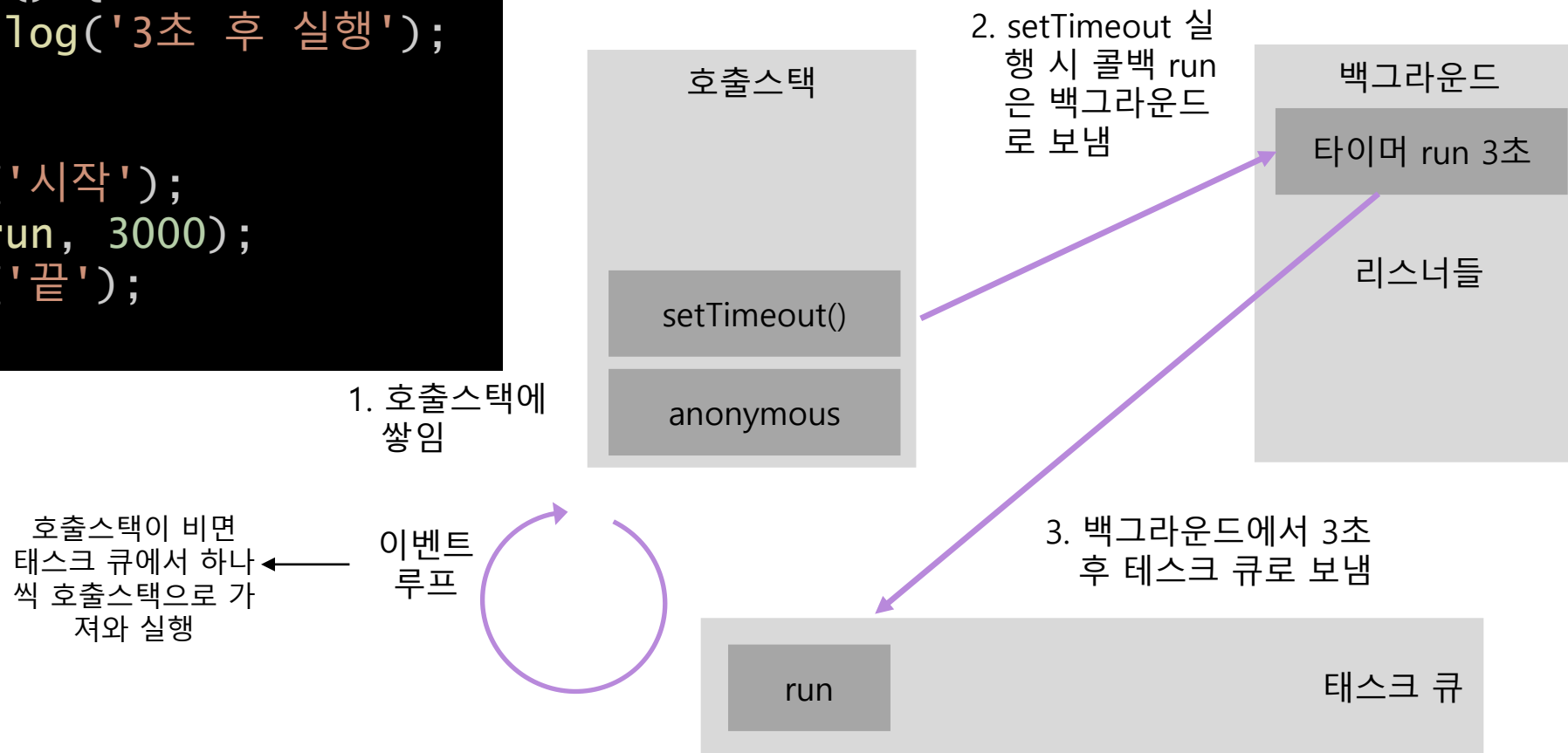
4) 이벤트 기반 : 이벤트 리스너에 콜백 함수 등록. 이벤트 루프 사용



- * 이벤트 루프 : 이벤트 발생 시 호출할 콜백 함수들을 관리하고, 호출된 콜백 함수의 실행 순서를 결정하는 역할을 담당. 노드가 종료될 때까지 이벤트 처리를 위한 작업을 반복하므로 루프라고 부름.
- * 백그라운드 : **setTimeout** 같은 타이머나 이벤트 리스너들이 대기하는 곳. 자바스크립트가 아닌 다른 언어로 작성된 프로그램이라고 봐도 됨.
- * 태스크 큐 : 이벤트 발생 후, 백그라운드에서는 태스크 큐로 타이머나 이벤트 리스너의 콜백 함수를 보냄. 정해진 순서대로 콜백들이 줄을 서 있으므로 콜백 큐라고도 부름.

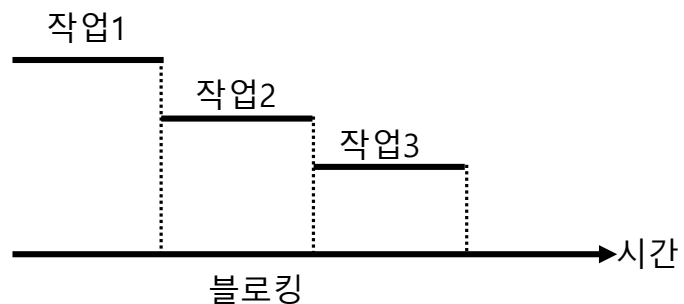
4) 이벤트 기반 : 이벤트 리스너에 콜백 함수 등록. 이벤트 루프 사용

```
function run() {  
  console.log('3초 후 실행');  
}  
  
console.log('시작');  
setTimeout(run, 3000);  
console.log('끝');
```

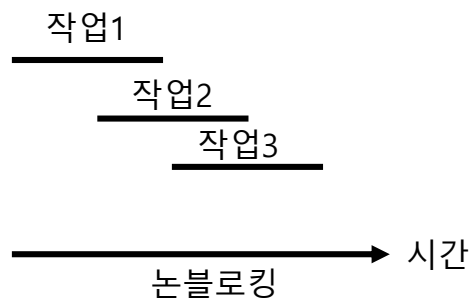


5) 논 블로킹

- * 논블로킹: 이전 작업이 완료될 때까지 대기하지 않고 다음 작업 수행. I/O와 JS 코드는 동시 실행 가능 → 논블로킹
- * 블로킹: 이전 작업이 끝나야만 다음 작업을 수행. JS 코드는 동시에 실행 불가 → 블로킹



```
function longRunningTask()  
{  
  console.log('작업끝');  
}  
  
console.log('시작');  
longRunningTask();  
console.log('다음작업');
```



```
function longRunningTask() {  
  console.log('작업끝');  
}  
  
console.log('시작');  
setTimeout(longRunningTask, 0);  
console.log('다음작업');
```

6) 싱글 스레드

- * 프로세스 : 운영체제에서 할당하는 작업의 단위. 노드나 웹 브라우저 같은 프로그램은 개별적인 프로세스
- * 스레드 : 프로세스 내에서 실행되는 흐름의 단위. 하나의 프로세스가 스레드를 여러 개 생성해 여러 작업을 동시에 처리 가능.

스레드는 부모 프로세스의 자원을 공유. 같은 주소의 메모리에 접근 가능하므로 데이터를 공유

- * 노드는 싱글 스레드 ➔ 엄밀히 말하면 멀티 스레드

노드 실행 ➔ 프로세스 생성 ➔ 스레드 여러 개 생성 ➔ 프로그래머가 직접 제어할 수 있는 스레드는 하나뿐

- * 요청이 여러 개 들어오면 한 번에 하나씩 요청을 처리하지만 블로킹이 발생할 것 같은 경우에는 논 블로킹 방법으로 대기 시작 절약

※ 스레드풀 : 노드가 특정 동작을 수행할 때 스스로 멀티 스레드를 사용. 대표적인 예로 암호화, 파일 입출력, 압축 등

※ 워커 스레드 : 노드 **12**버전에서 안정화된 기능으로 멀티 스레드를 사용할 수 있는 도구

3. 서버로서의 노드

1) Node의 장단점

장점	단점
멀티 스레드 방식에 비해 적은 컴퓨터 자원 사용	기본적으로 싱글 스레드라서 CPU 코어를 하나만 사용
I/O 작업이 많은 서버로 적합	CPU 작업이 많은 서버로는 부적합
멀티 스레드 방식보다 쉬움	하나뿐인 스레드가 멈추기 않도록 관리가 필요함
웹 서버가 내장되어 있음	서버 규모가 커졌을 때 서버를 관리하기 어려움
자바스크립트를 사용함	어중간한 성능
JSON 형식과 쉽게 호환됨	

2) 사용 예

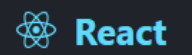
* **NASA**, 에어비앤비, 우버, 넷플릭스, 링크드인,페이팔, 월마트, 이베이, 네이버, 카카오, 위메프, 야놀자

4. 서버 외의 노드

1) 웹, 모바일, 애플리케이션 개발에 사용

2) 노드 기반으로 돌아가는 대표적인 웹 프레임 워크

* 앵귤러(Angular) : <https://angular.kr/>,  , 구글 진영에서 프론트엔드 앱을 만들 때 주로 사용

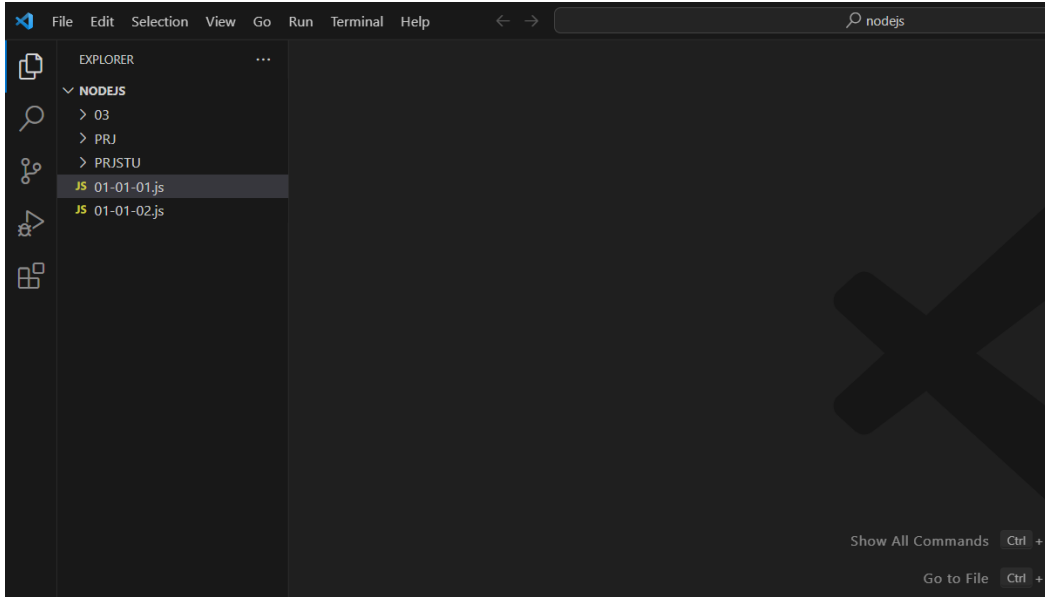
* 리액트(React) :  , 페이스북 진영에서 주로 사용

* 리액트 네이티브 : 모바일 개발 도구, 페이스북/인스타그램/핀터레스트/월마트/테슬라 등이 사용

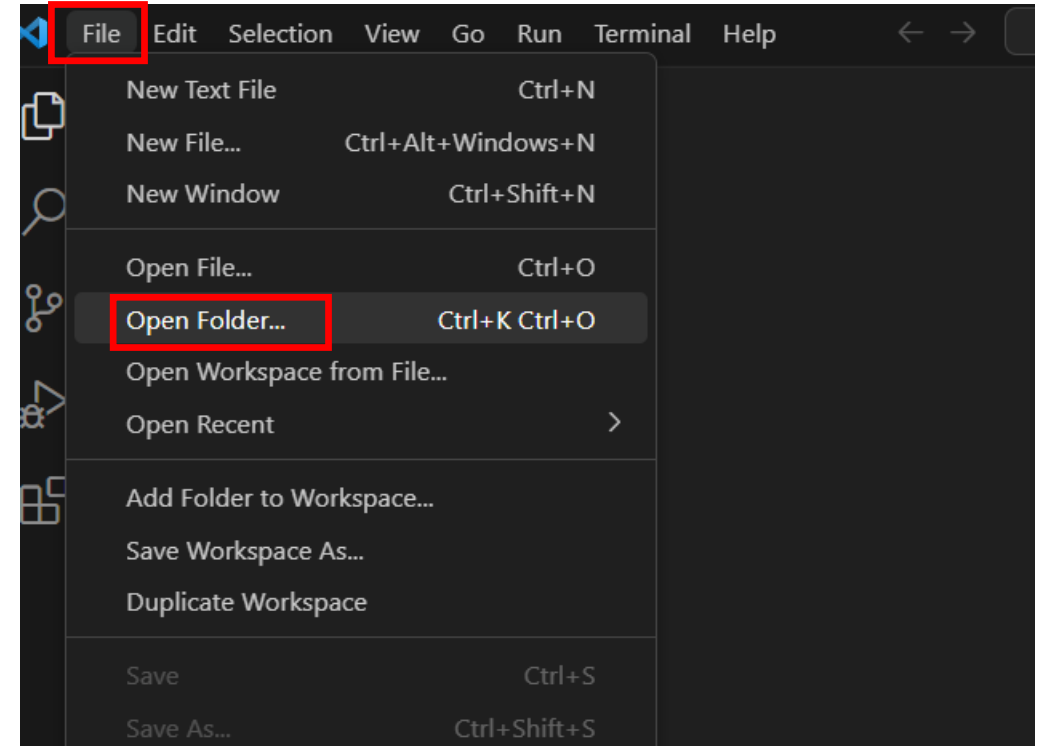
* 뷰(Vue)

* 일렉트론(Electron) : 데스크 톱 개발 도구, Atom/Stack/Discord/VSCode 등의 프로그램

① Visual Studio Code 실행

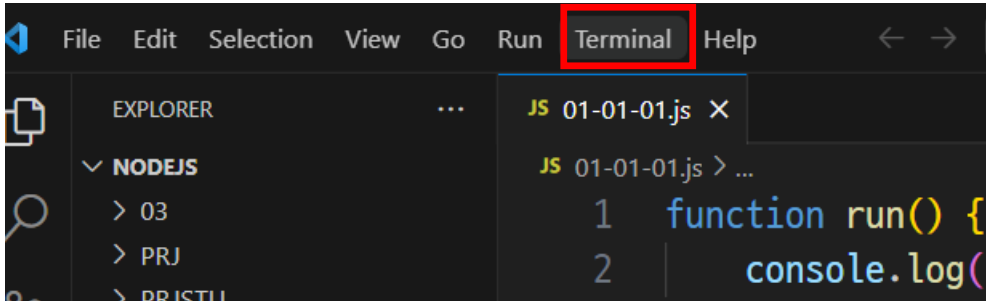


② File 메뉴 → OpenFolder → 작업 폴더 선택

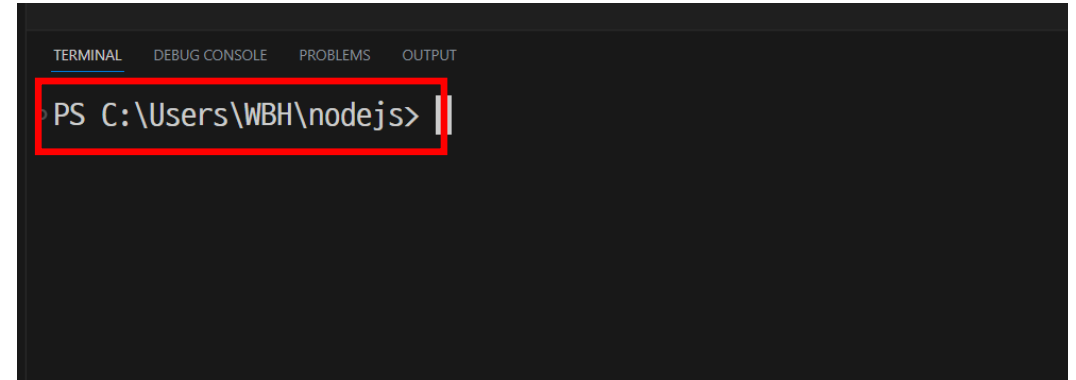


③ 파일 생성하여 코드 작성 후 확장자 js로 저장

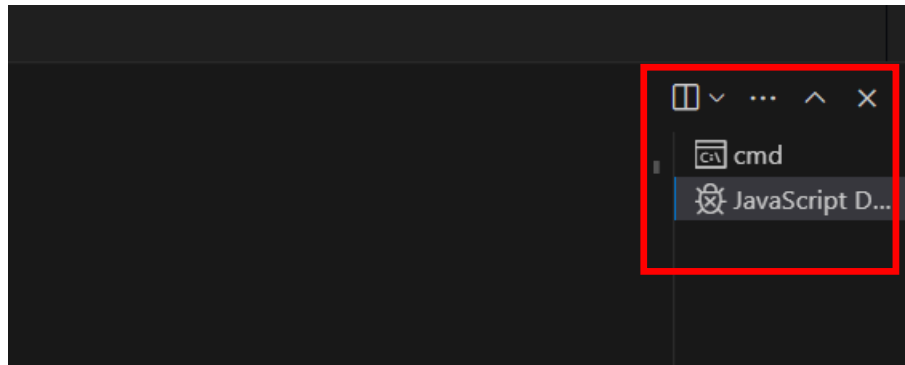
④ Terminal 메뉴 선택 ➔ New Terminal 선택



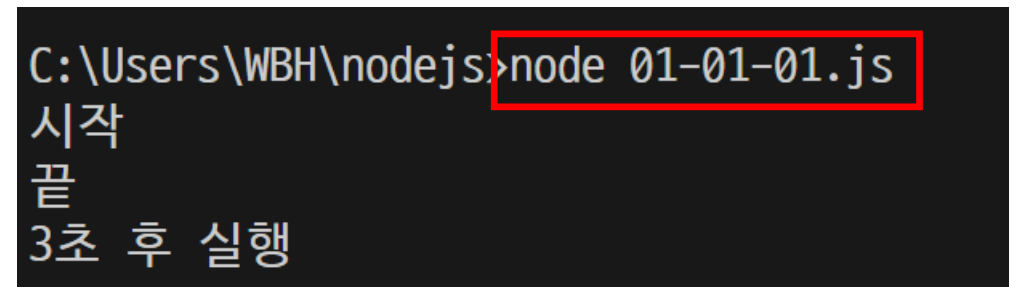
⑤ 하단에 실행 터미널 생성 ➔ 작업 폴더를 현재 폴더로



⑥ 터미널 우측에서 원하는 터미널 유형 선택



⑦ 노드 프로그램 실행 방법 : 커맨드 라인에 node 쓰고 한 칸 띄고 js 파일명



알아두어야 할 자바스크립트 – const, let

① 블록 스코프

```
if(true) {  
    var x = 3;  
}  
console.log(x);  
  
if(true) {  
    const y = 3;  
}  
console.log(y)
```

var : 함수 scope
const, let : block scope

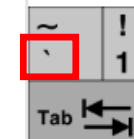
block이란 중괄호 { } 내의 영역

② let과 const의 차이

```
const a = 0;  
a = 1;  
let b = 0;  
b = 1;
```

const로 정의된 변수의 값은 변경 불가 ➔ 상수 취급

- ① 백틱 `` (틸트 ~ 아래의 문자) 으로 감싸는 문자열을 템플릿 문자열이라고 함
백틱 안에 \${변수명}과 같이 변수를 불러올 수 있음



```
var num1 = 1;
var num2 = 2;
var result = 3;
var string1 = num1 + ' 더하기 ' + num2 + '는 \'' + result + '\';
console.log(string1)
var string2 = `${num1} 더하기 ${num2} 는 '${result}'`
console.log(string2)
```

① 화살표 함수 표현, 기존 함수도 사용 가능

```
function add1(x, y) {  
    return x + y;  
}  
  
const add2 = (x, y) => {  
    return x + y;  
}  
  
const add3 = (x,y) => x + y;  
  
const add4 = (x,y) => (x + y);  
  
function not1(x){  
    return !x;  
}  
  
const not1 = x => !x;
```

① 일반적인 코드

```
var candyMachine = {  
  status : {  
    name: 'node',  
    count: 5,  
  },  
  getCandy: function(){  
    this.status.count--;  
    return this.status.count;  
  },  
};  
var getCandy = candyMachine.getCandy;  
var count = candyMachine.status.count;  
  
console.log(count)  
t = getCandy ()  
console.log(count)  
console.log(t)
```

② 구조 분해 할당

```
var candyMachine = {  
  status : {  
    name: 'node',  
    count: 5,  
  },  
  getCandy: function(){  
    this.status.count--;  
    return this.status.count;  
  },  
};  
  
const {getCandy, status : {count}} = candyMachine;  
console.log(count)  
t = getCandy()  
console.log(count)  
console.log(t)
```

③ this 오류 해결 → bind사용

```
var candyMachine = {
  status : {
    name: 'node',
    count: 5,
  },
  getCandy: function(){
    this.status.count--;
    return this.status.count;
  },
};

const {getCandy, status : {count}} = candyMachine;
getCandy1 = getCandy.bind(candyMachine)
console.log(count)
t = getCandy1()
console.log(count)
console.log(t)
```

① 모듈 : 특정한 기능을 하는 함수나 변수들의 집합

W01-2.js

```
const { odd, even } = require('./w01-2-var'); //require 함수 : 외부 모듈을 객체로 반환
const mtest = require('./w01-2-var');
function checkOddOrEven(num){
  if (num%2) {
    return odd;
  }
  return even;
}

console.log(checkOddOrEven(5));

function checkOddOrEven2(num){
  if (num%2) {
    return mtest.odd;
  }
  return mtest.even;
}

console.log(checkOddOrEven2(8));
```

W01-2-var.js


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

module.exports = { //module 객체의 exports 속성에 외부에 사용가능
  하게 할
  odd, //변수, 또는 함수를 객체 형태로 전달
  even,
};
```


1. 웹 서버 프로그램 구동

1) 웹 서버 구동 파일 만들기 - main.js

```
var http = require('http');
var fs = require('fs');
var app = http.createServer(function(request, response) {
  var url = request.url;
  if(request.url == '/') {
    url = '/05-index.html';
  }
  if(request.url == '/favicon.ico') {
    return response.writeHead(404);
  }
  response.writeHead(200);
  console.log(__dirname + url); // ①웹브라우저가 요청한 파일의 경로를 콘솔에 출력
  response.end(fs.readFileSync(__dirname + url));
});
app.listen(3000);
```



```
code: 'ENOENT',
path: 'C:\\작업들\\5_강의자료\\2022-2학기\\웹DB프로그래밍\\programs\\01\\coding.jpg'
}
PS C:\작업들\5_강의자료\2022-2학기\웹DB프로그래밍\programs\01> node main.js
PS C:\작업들\5_강의자료\2022-2학기\웹DB프로그래밍\programs\01> node main.js
C:\작업들\5_강의자료\2022-2학기\웹DB프로그래밍\programs\01\05-index.html
```

1. 웹 서버 프로그램 구동

1) 웹 서버 구동 파일 만들기 - main.js

```
var http = require('http');
```

- **require** 함수 : 외부 모듈을 객체화 해서 가져오는 함수
- **http** 모듈 : **http** 웹 서버와 관련된 모든 기능을 담은 모듈
- 서버 생성을 위해서 **createServer()**와 **listen()** 메소드가 필요함

```
var http = require('http');
var app = http.createServer( function(request,response){// request,response : 43~46p 참조

    // 여기에 클라이언트의 요청을 받아서 URL을 분류하고
    // URL에 따른 controller에 해당하는 로직을 작성

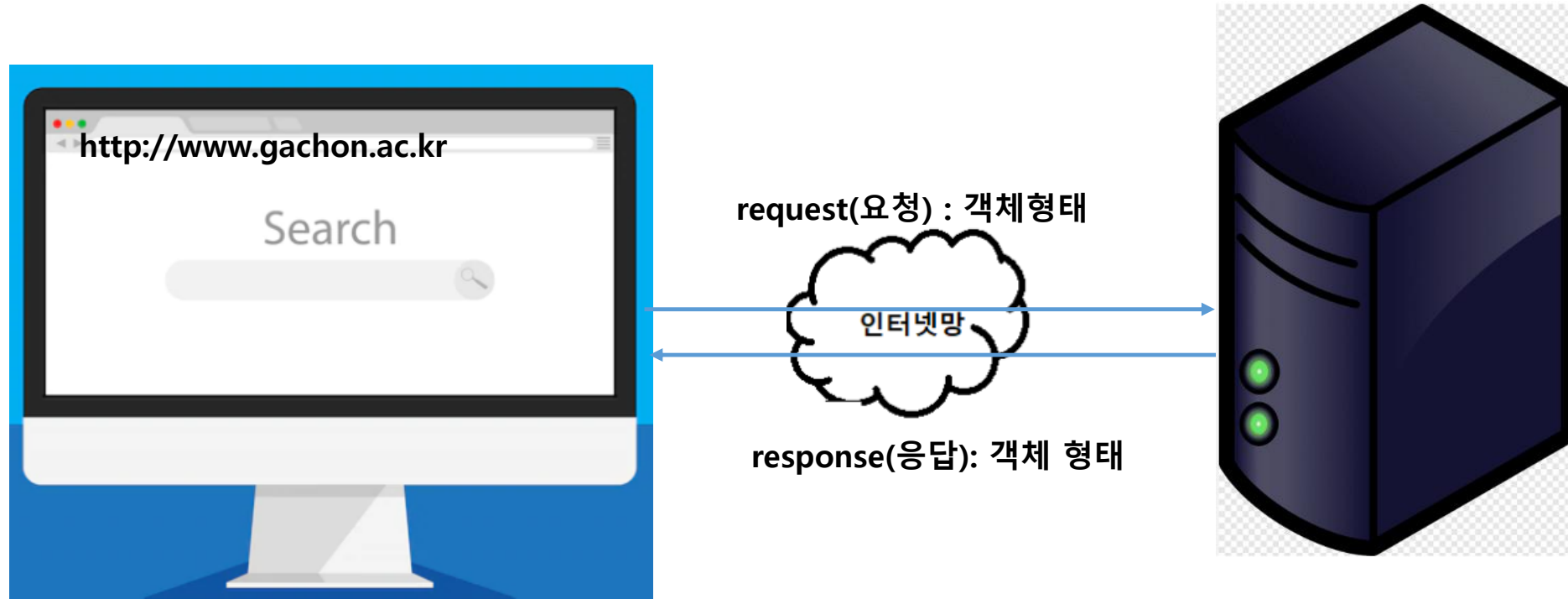
});
app.listen(3000);
```

- **createServer()** 메소드 : 웹서버 객체를 생성
- 웹서버 객체의 **listen** 메소드 : 클라이언트의 웹 요청을 기다리고 있음.

웹 요청이 들어오면 **createServer**의 **callback**함수인 **function (req, res) { }** 함수를 실행

1. 웹 서버 프로그램 구동

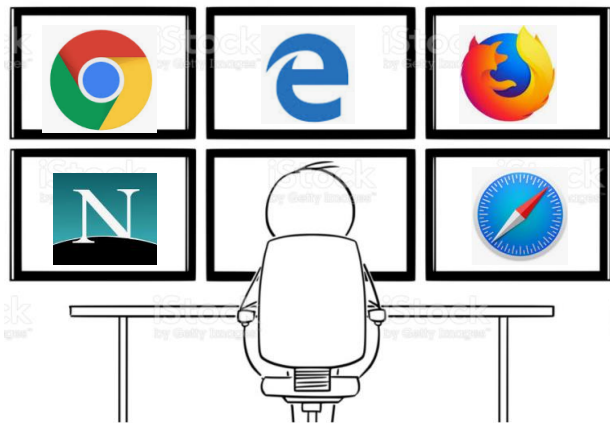
1) 웹 서버 구동 파일 만들기 - **main.js**





HTTP 프로토콜의 Request Header와 Response Header

HTTP 기본적으로 **request**(요청)/**response**(응답) 구조로 되어있다.



Request

GET /test.html HTTP/1.1

Host: google.com

Accept: text/html

Accept-Encoding: gzip, deflate

Connection: keep-alive

hl=ko&ogbl=0&page=99

start line

headers

blank line

body

HTTP Request Message

웹서버

Request 객체로 받음

Response 객체로 보냄

HTTP/1.1 200 OK

Date: Sun, 26 June ...

Server: Apache

Content-Length: 35

Content-Type: text/html

<h1>Hello World</h1>

status line

headers

blank line

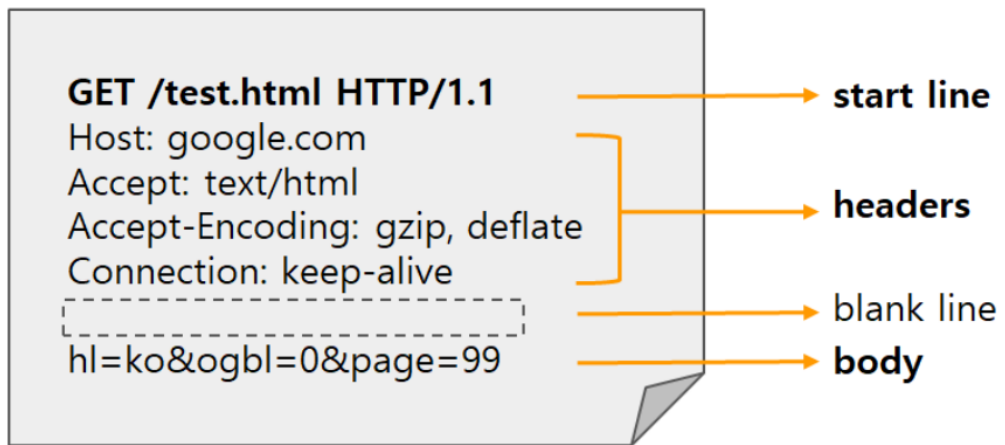
body

HTTP Response Message



HTTP 프로토콜의 Request Header와 Response Header

Request Header



HTTP Request Message

- * start line

HTTP method GET
Request target /test.html
HTTP version HTTP/1.1

- * headers : request에 대한 추가 정보를 담고 있는 부분

request 메시지 body의 총 길이 등 Key:Value 형태로 구성

Host: google.com 요청하려는 서버 호스트 이름과 포트 번호
Accept 클라이언트가 처리 가능한 미디어 타입 종류 나열

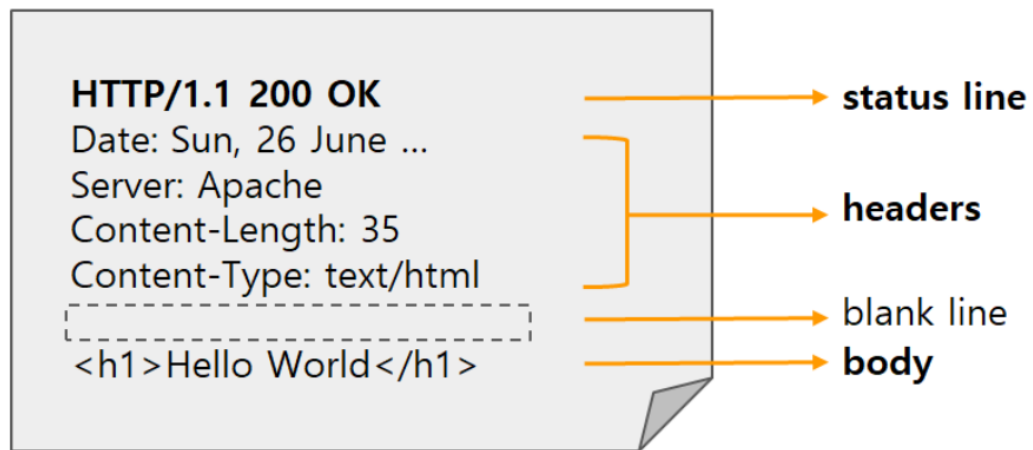
- * body : HTTP request가 전송하는 데이터를 담고 있는 부분

post 요청일 경우, HTML form 데이터가 포함되어 있음



HTTP 프로토콜의 Request Header와 Response Header

Response Header



HTTP Response Message

- * **status line**

HTTP version HTTP/1.1

Status Code 200

Status Text OK

- * **headers** : 서버의 종류등의 정보 포함

- * **body** : 보내려고 하는 문서 내용. 비어 있는 경우도 있음.

1. 웹 서버 프로그램 구동

1) 웹 서버 구동 파일 만들기 - **main.js**, 외부ip

```
var http = require('http');
const host = '192.168.0.5';
var app = http.createServer( function(req,res){

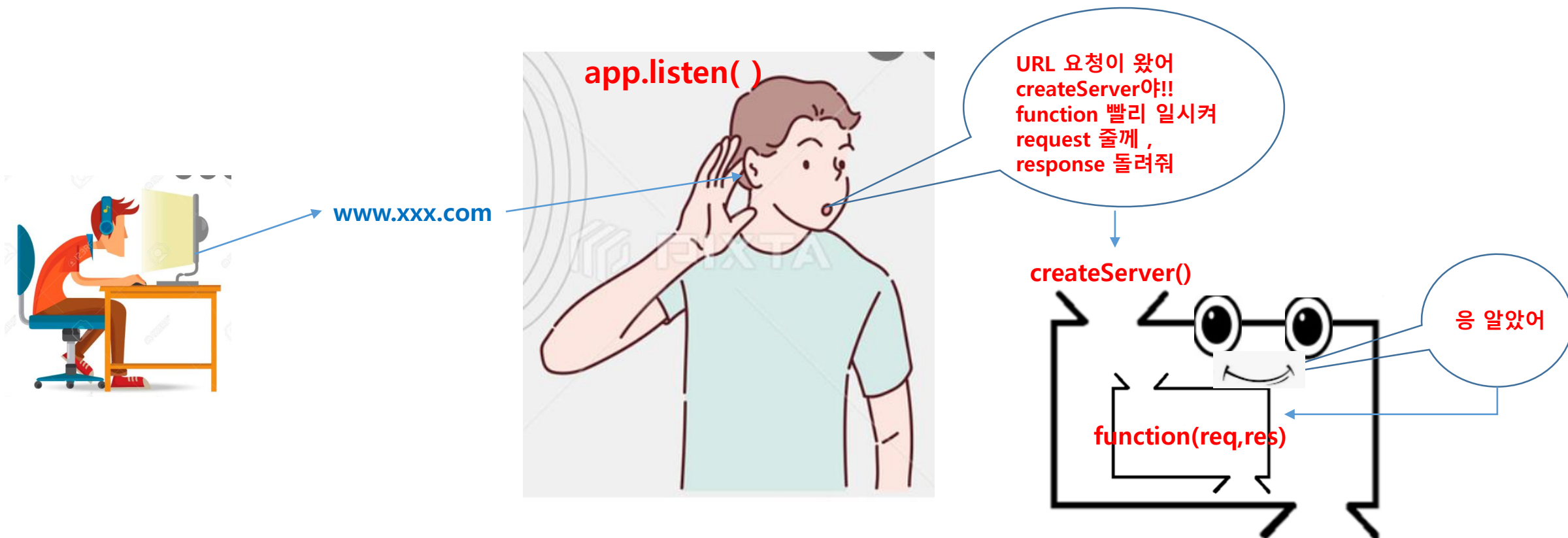
    // 여기에 클라이언트의 요청을 받아서 URL을 분류하고
    // URL에 따른 controller에 해당하는 로직을 작성

    res.writeHead(200);
    res.end("Hello. My response, Node.js !!!")

});
app.listen(3000,host);
```

1. 웹 서버 프로그램 구동

1) 웹 서버 구동 파일 만들기 - **node** 웹 서비스 제공 메커니즘



1. 웹 서버 프로그램 구동

1) 웹 서버 구동 파일 만들기 - **main.js** 의 또 다른 예

```
var http = require('http');
var app = http.createServer( function(req,res){

    // 여기에 클라이언트의 요청을 받아서 URL을 분류하고
    // URL에 따른 controller에 해당하는 로직을 작성

    res.writeHead(200);
    res.end("Hello. My response, Node.js !!!")

});
app.listen(3000);
```

- **function(req, res) { }**

첫번째 매개변수 **req** : **request** 객체로써 웹서버에 클라이언트의 요청이 들어오면 요청 관련된 정보가 **request** 객체에 저장

두번째 매개변수 **res** : **response** 객체로써 클라이언트에(웹브라우저) 응답하기 위한 메소드 및 정보 저장

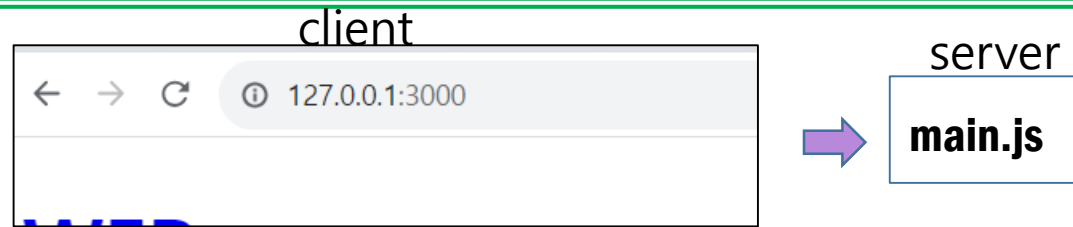
- **response** 객체의 메소드

writeHead(statusCode, Object) : 응답헤더작성, 예) **writeHead(200, {'Content-Type': 'text/html'})**

end([data]) : 응답 본문 작성

1. 웹 서버 프로그램 구동

1) 웹 서버 구동 파일 만들기 - main.js



```
var http = require('http');
var fs = require('fs'); // ① node.js의 file system 모듈을 객체화 해서 fs 변수에 저장
var app = http.createServer(function(request, response) {
  var url = request.url;
  if(request.url == '/') { // ② url 분류기 : 요청이 들어온 url 에 따라 처리 url
    url = '/index.html'; // Controller
  }
  if(request.url == '/favicon.ico') {
    return response.writeHead(404);
  }
  response.writeHead(200);
  console.log(__dirname + url); // ③ 웹브라우저가 요청한 파일의 경로를 콘솔에 출력
  response.end(fs.readFileSync(__dirname + url)); // ④ Template
});
app.listen(3000);
```

② request 객체의 url 속성에는 client가 요청한 url이 저장되어 있음.

④ fs.readFileSync : 웹브라우저가 요청한 파일 (__dirname+url)을 읽어서 응답, 동기식





__dirname : __ 는 JS에서 기본적으로 정의된 변수 앞에 연결 현재 파일이 위치한 폴더의 절대경로를 저장

1. 웹 서버 프로그램 구동

1) 웹 서버 구동 파일 만들기 - main.js

```
var http = require('http');
var fs = require('fs');
var app = http.createServer(function(request, response) {
  var url = request.url;
  if(request.url == '/') {
    url = '/index.html';
  }
  if(request.url == '/favicon.ico') {
    return response.writeHead(404);
  }
  response.writeHead(200);
  console.log(__dirname + url); // ①웹브라우저가 요청한 파일의 경로를 콘솔에 출력
  response.end('egoing : ' + url); // ③ 화면에 출력
});
app.listen(3000);
```

← → ↻ ⓘ 127.0.0.1:3000

 Gmail  지도  번역  PYR

egoing : /index.html

1. 웹 서버 프로그램 구동

>**node main.js** → 웹 서버 구동

2. URL 과 홈디렉토리 개념

URL에 매핑 되는 **root** 디렉토리가 홈 디렉토리

3. 웹 브라우저에서 **html** 실행

localhost:3000//

4. 웹 서버 끄기 : **ctrl + c**