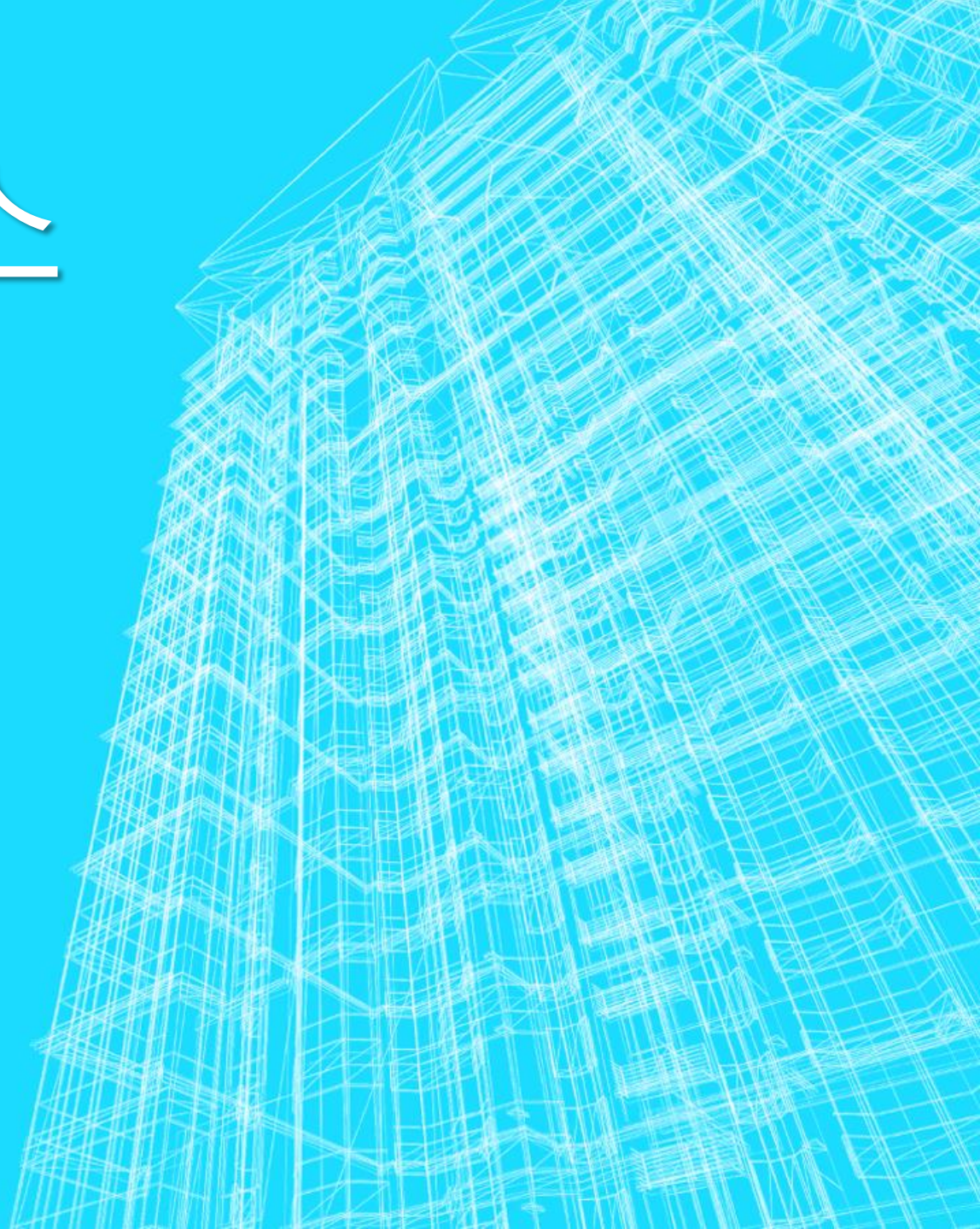


# 네트워크 서비스 프로토콜

## 4주차 1

소프트웨어학부  
김형균 교수



# 주차별 수업 계획

01주차	9월 3일	교과목 소개	09주차	10월 29일	몽고디비 1
	9월 5일	교과목 소개		10월 31일	몽고디비 2
02주차	9월 10일	노드 시작하기	10주차	11월 5일	익스프레스로 SNS 서비스 만들기 1
	9월 12일	추석 휴무		11월 7일	익스프레스로 SNS 서비스 만들기 2
03주차	9월 17일	알아두어야 할 자바스크립트 1	11주차	11월 12일	웹 API 서버 만들기 1
	9월 19일	알아두어야 할 자바스크립트 2		11월 14일	웹 API 서버 만들기 2
04주차	9월 24일	노드 기능 알아보기 1	12주차	11월 19일	웹 소켓으로 실시간 데이터 전송하기 1
	9월 26일	노드 기능 알아보기 2		11월 21일	웹 소켓으로 실시간 데이터 전송하기 2
05주차	10월 1일	개천절 휴무	13주차	11월 26일	실시간 경매 시스템 만들기 1
	10월 3일	패키지 매니저		11월 28일	실시간 경매 시스템 만들기 2
06주차	10월 8일	익스프레스 웹 서버 만들기 1	14주차	12월 3일	과제발표 1
	10월 10일	익스프레스 웹 서버 만들기 2		12월 5일	과제발표 2
07주차	10월 15일	MySQL 1	15주차	12월 10일	기말고사
	10월 17일	MySQL 2		12월 12일	보충수업
08주차	10월 22일	중간고사			
	10월 24일	보충수업			



# 수업에 들어가며

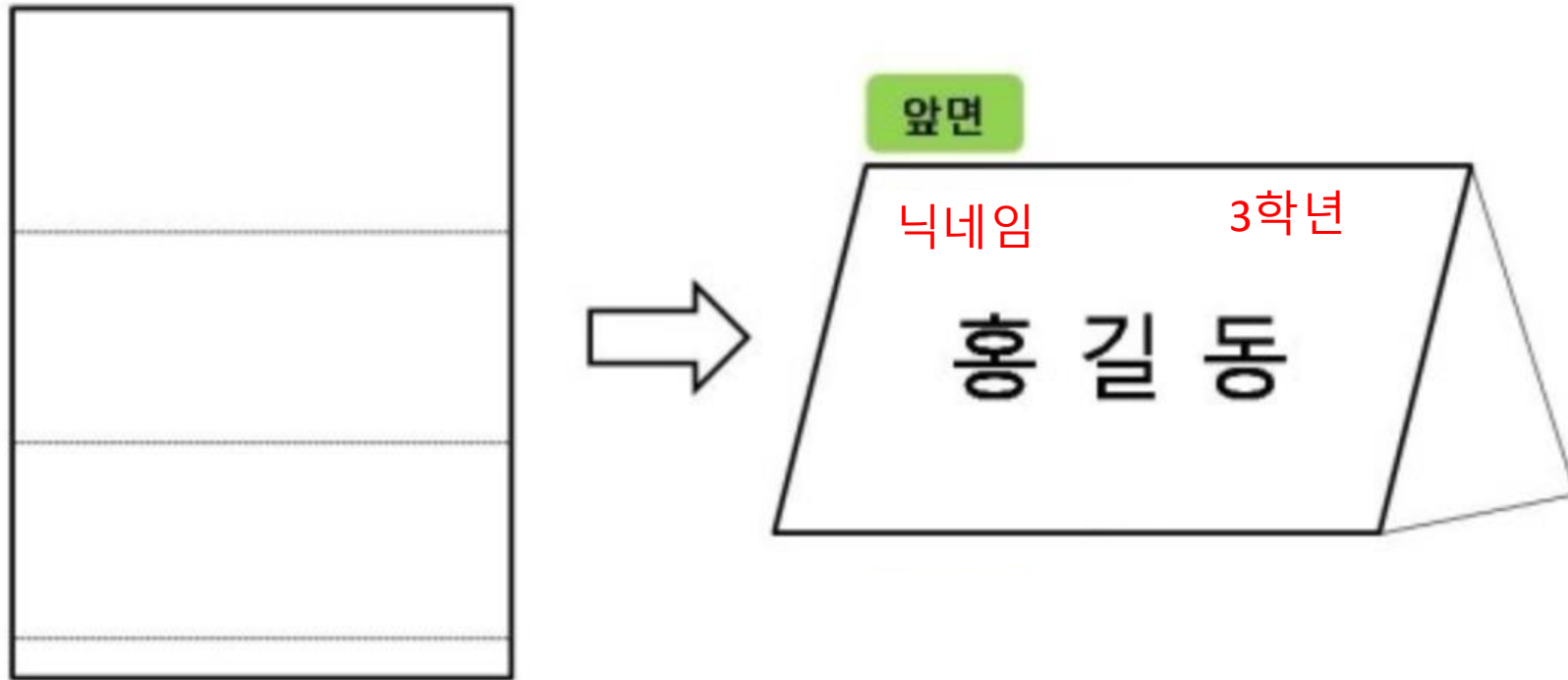
- 네임텐트 제작
- 복습문제
  - => 함수형태로 가장 최적화해 보기
- 이번 시간 학습목표
  - `require()`, `exports`, `module.exports`
  - 모듈로 만들기
  - Http모듈로 웹서버 만들기
    - : http 모듈
    - : 노드로 http 서버 만들기
    - : 파일 시스템 접근하기
    - : fs로 파일 만들기
    - : html 읽어서 전송하기



# 네임텐트 제작

네임텐트 만들어 책상위에 올려 놓기

맨 아래를 먼저 조금 접어둔 상태에서 A4 용지를 3등분해야 지지가 된다



# 복습문제

```
var circle = {  
  center : {x:1.0, y:2.0} ,  
  radius : 2.5,  
};
```

```
console.log("원의 중심좌표는 (" + circle.center.x + "," + circle.center.y + ")");
```

area, round, translate 함수를  
=> 함수형태로 가장 최적화해 정의해 보자.

```
console.log("원의 면적은 " + area(circle.radius).toFixed(2) + "입니다.");  
console.log("원의 둘레는 " + round(circle.radius).toFixed(2) + "입니다.");  
translate(1,2);  
console.log("(1,2)이동한 원의 중심좌표는 (" + circle.center.x + "," + circle.center.y + ")");
```

# REQUIRE(), EXPORTS

node.js에서는 모듈을 불러오기 위해 require()함수  
두개의 파일을 작성해보자. foo.js, bar.js

```
//foo.js  
const a = 10
```

```
//bar.js  
console.log(a)
```

실행

node bar.js  
ReferenceError: a is not defined

# REQUIRE(), EXPORTS

- 예러 해결방안
  - bar.js에 foo.js 모듈을 불러오자
  - 이때, require 함수를 사용
  - require함수로 foo.js를 bar.js로 가지고 온다.

```
//bar.js  
const foo = require('./foo')  
console.log(foo.a)
```

실행

node bar.js  
ReferenceError: a is not defined

# REQUIRE(), EXPORTS

- 에러 해결방안

- exports**는 require()함수로 연결하고자 하는 모듈의 특정 값을 다른 모듈로 넘겨주고 싶을 때 사용

```
//foo.js  
const a = 10
```



```
//foo.js  
exports.a=10
```

```
//bar.js  
const foo = require('./foo')  
console.log(foo.a)
```





# module.exports 사용

- 주로 객체, 함수형태의 모듈을 연계할 때 사용
- 먼저 모듈로 연계할 함수를 정의
- 파일 끝에 module.exports에 모듈로 만들 함수를 지정

```
//bar.js
const foo = require('./foo')
const checknum = require('./fun')

console.log(foo.a);
checknum(10);
```

```
//foo.js
exports.a=10
```

```
//fun.js
function check(num) {
  if (num % 2) { // 홀수면
    console.log('홀수');
  }
  console.log('짝수');
}

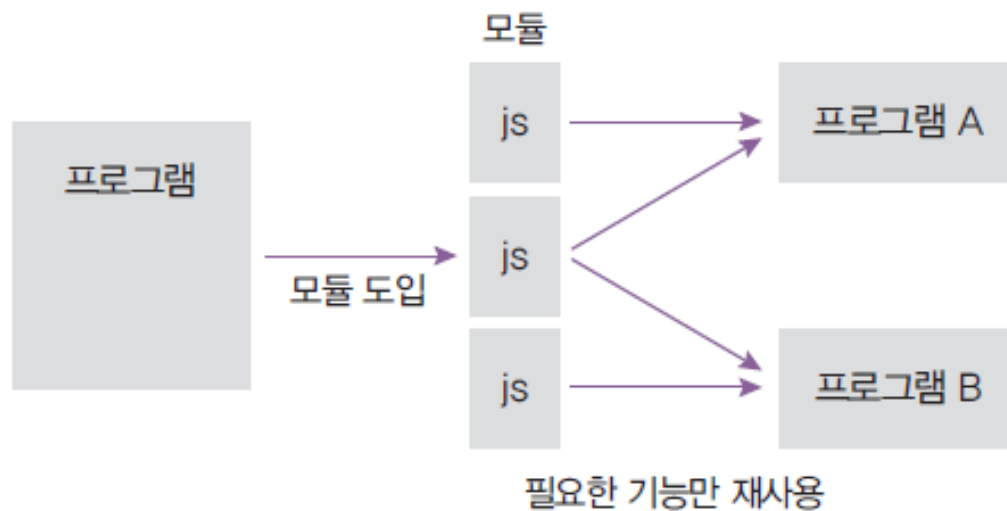
module.exports = check;
```

# 모듈로 만들기-P.78

노드는 자바스크립트 코드를 모듈로 만들 수 있음

- 모듈: 특정한 기능을 하는 함수나 변수들의 집합
- 모듈로 만들면 여러 프로그램에서 재사용 가능

▼ 그림 3-2 모듈과 프로그램



./index\_old.js

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

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

```
function checkStringOddOrEven(str) {  
  if (str.length % 2) { // 홀수면  
    return odd;  
  }  
  return even;  
}
```

```
console.log(checkOddOrEven(10));  
console.log(checkStringOddOrEven('hello'));
```

모듈화

./index.js

문자열수비교함수정의  
숫자 10 홀짝 결과 출력  
'hello' 홀짝 결과출력

./func.js

숫자홀짝비교함수 정의

./var.js

const odd  
const even 정의

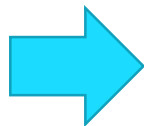
./index\_old.js

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

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

function checkStringOddOrEven(str) {
  if (str.length % 2) { // 홀수면
    return odd;
  }
  return even;
}

console.log(checkOddOrEven(10));
console.log(checkStringOddOrEven('hello'));
```



./index.js

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

function checkStringOddOrEven(str) {
  if (str.length % 2) { // 홀수면
    return odd;
  }
  return even;
}

console.log(checkNumber(10));
console.log(checkStringOddOrEven('hello'));
```



./index.js

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

function checkStringOddOrEven(str) {
  if (str.length % 2) { // 홀수면
    return odd;
  }
  return even;
}

console.log(checkNumber(10));
console.log(checkStringOddOrEven('hello'));
```



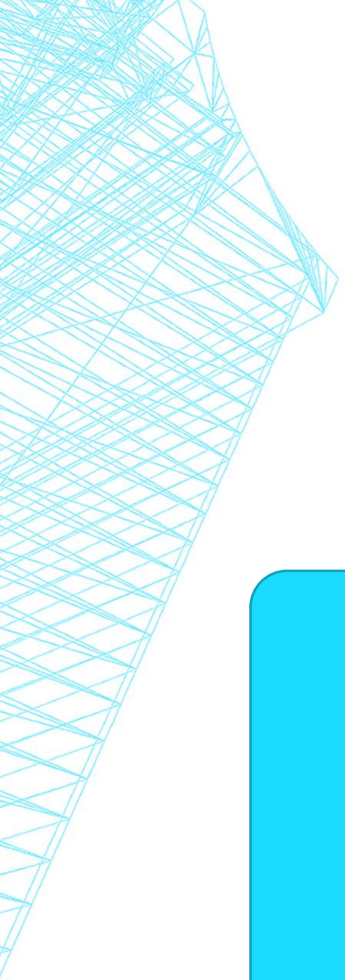
# 문제 다음 코드를 모듈화해보자.

```
var center = {x:1.0, y:2.0};  
var radius = 2.5;
```

```
console.log("원의 중심좌표는 (" + center.x + "," + center.y + ")");
```

```
const area = () => (Math.PI * radius * radius);  
const round = () => (2 * Math.PI * radius);  
const translate = (a,b) => {  
  center.x = center.x + a;  
  center.y = center.y + b;  
};
```

```
console.log("원의 면적은 " + area().toFixed(2) + "입니다.");  
console.log("원의 둘레는 " + round().toFixed(2) + "입니다.");  
translate(1,2);  
console.log("(1,2)이동한 원의 중심좌표는 (" + center.x + "," + center.y + ")");
```

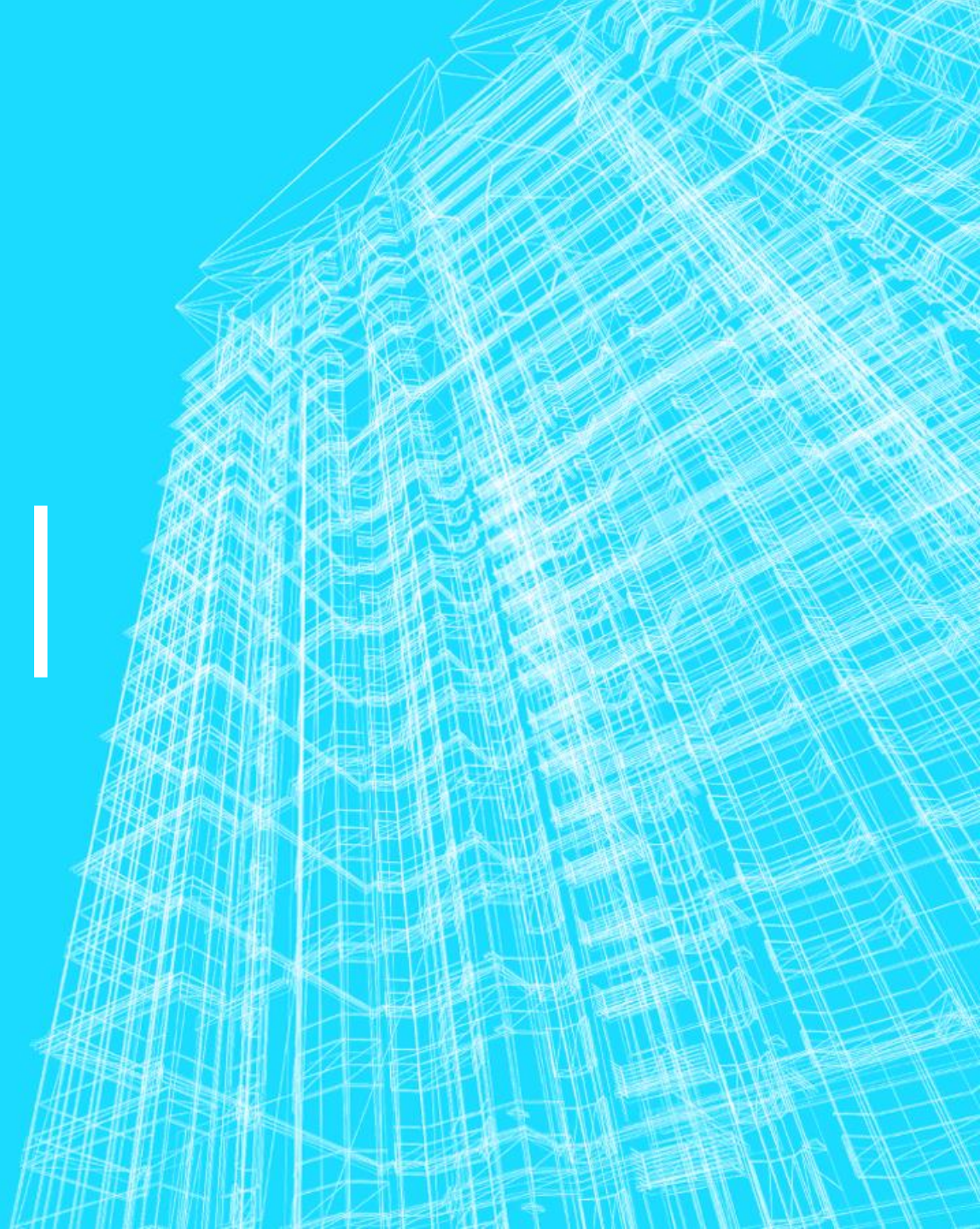


```
var { center, radius } = require('./cir1');
var area = require('./cir2');
var round = require('./cir3');
var translate = require('./cir4');

console.log("원의 중심좌표는 (" + center.x + "," + center.y + ")");

console.log("원의 면적은 " + area().toFixed(2) + "입니다.");
console.log("원의 둘레는 " + round().toFixed(2) + "입니다.");
translate(1,2);
console.log("(1,2)이동한 원의 중심좌표는 (" + center.x + "," + center.y + ")");
```

# 4장 HTTP 모듈로 웹서버 만들기





# HTTP 모듈

- Node.js에서 가장 기본적인 웹 모듈이며 HTTP 웹 서버를 생성하는 것과 관련된 모든 기능을 담당
- server 객체
  - http 모듈에서 가장 중요한 객체는 server 객체
  - http 모듈의 `createServer()` 메서드를 사용하여 **server 객체를 생성**
- Server 객체의 메서드
  - **listen**(port[, callback]) : 서버를 실행
  - `close()` : 서버를 종료





# HTTP 모듈

## Server 객체의 이벤트

- request : 클라이언트가 요청할 때 발생하는 이벤트
- connection : 클라이언트가 접속할 때 발생하는 이벤트
- close : 서버가 종료될 때 발생하는 이벤트
- checkContinue : 클라이언트가 지속적인 연결을 하고 있을 때 발생하는 이벤트
- upgrade : 클라이언트가 HTTP 업그레이드를 요청할 때 발생하는 이벤트
- clientError : 클라이언트에서 오류가 발생할 때 발생하는 이벤트

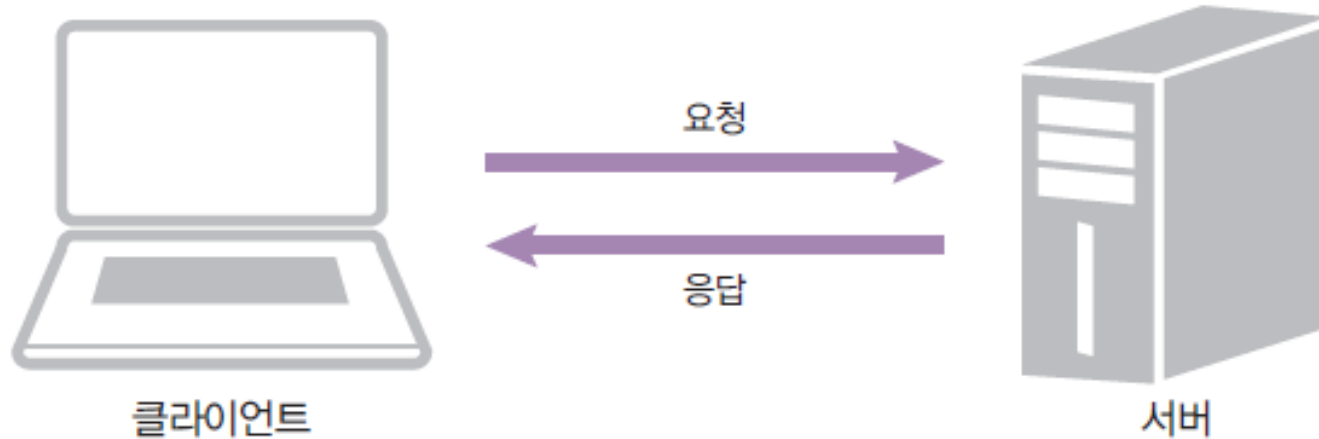


# 요청과 응답하기 – P.132

## 서버와 클라이언트의 관계

- 클라이언트가 서버로 요청(request)을 보냄
- 서버는 요청을 처리
- 처리 후 클라이언트로 응답(response)을 보냄

▼ 그림 4-1 클라이언트와 서버의 관계



# 노드로 HTTP 서버 만들기

## 1) **CREATESERVER()** 메서드

http 요청에 응답하는 노드 서버

- createServer로 요청 이벤트에 대기
- req 객체는 요청에 관한 정보가, res 객체는 응답에 관한 정보가 담겨 있음

createServer.js

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

http.createServer((req, res) => {
  // 여기에 어떻게 응답할지 적어줍니다.
});
```

---

# 노드로 HTTP 서버 만들기

## 2) 8080 포트에 연결하기

res 메서드로 응답 보냄

- write로 응답 내용을 적고
- end로 응답 마무리(내용을 넣어도 됨)

listen(포트) 메서드로 특정 포트에 연결

server1.js

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

http.createServer((req, res) => {
  res.write('<h1>Hello Node!</h1>');
  res.end('<p>Hello Server!</p>');
}).listen(8080, () => {
  console.log('8080번 포트에서 서버 대기 중입니다!');
});
```

# 노드로 HTTP 서버 만들기

## 3) 8080 포트로 접속하기

스크립트를 실행하면 8080 포트에 연결됨

콘솔

```
$ node server1
```

```
8080번 포트에서 서버 대기 중입니다!
```

localhost:8080 또는 <http://127.0.0.1:8080>에 접속

▼ 그림 4-2 서버 실행 화면

**Hello Node!**

Hello Server!

# 파일 시스템 접근하기- P109

fs 파일 시스템에 접근하는 모듈

- 파일/폴더 생성, 삭제, 읽기, 쓰기 가능
- 웹 브라우저에서는 제한적이었으나 노드는 권한을 가지고 있음
- 형식) fs.readFile(filename, [options], callback);

--[fs\_readFile.js]-----

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

fs.readFile('./fs_readFile.js', function (err, data) {
  if (err) throw err;
  console.log(data.toString());
});
```

readFile() 함수는 기본적으로 세가지 아규먼트 인자  
첫번째는 읽고자 하는 파일이름  
두번째는 읽을 때 옵션, 하지만 이것은 생략이 가능  
세번째는 파일이 읽혀진 후 호출될 함수..  
readFile() 함수는 읽은 후 호출하는 함수는  
err 과 data 라는 두가지 아규먼트 인자로 받게 됩니다.

```
function (err, data) {
}
```

이때 data 는 옵션에 특별한 형식을 지정하지 않았다면  
buffer 타입이 됩니다.



# 파일 시스템 접근하기- P109

fs 파일 시스템에 접근하는 모듈

- 파일/폴더 생성, 삭제, 읽기, 쓰기 가능
- 웹 브라우저에서는 제한적이었으나 노드는 권한을 가지고 있음

readFile.js

```
const fs = require('fs');

fs.readFile('./readme.txt', (err, data) => {
  if (err) {
    throw err;
  }
  console.log(data);
  console.log(data.toString());
});
```

readme.txt

저를 읽어주세요.

콘솔

```
$ node readFile
<Buffer ec a0 80 eb a5 bc 20 ec 9d bd ec 96 b4 ec a3 bc ec 84 b8 ec 9a 94 2e>
저를 읽어주세요.
```

# FS로 파일 만들기

writeFile.js

```
const fs = require('fs');

fs.writeFile('./writeme.txt', '글이 입력됩니다', (err) => {
  if (err) {
    throw err;
  }
  fs.readFile('./writeme.txt', (err, data) => {
    if (err) {
      throw err;
    }
    console.log(data.toString());
  });
});
```

콘솔

\$ node writeFile  
글이 입력됩니다.

# HTML 읽어서 전송하기

write와 end에 문자열을 넣는 것은 비효율적

- fs 모듈로 html을 읽어서 전송하자
- write가 버퍼도 전송 가능
- server2.html 문서 작성 후 fs모듈을 이용해 HTML문서를 웹서버로 전송해보자

server2.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>Node.js 웹 서버</title>
</head>
<body>
  <h1>Node.js 웹 서버</h1>
  <p>만들 준비되셨나요?</p>
</body>
</html>
```

# HTML 읽어서 전송하기

- SERVER2.HTML 문서 작성 후 FS모듈을 이용해 HTML문서를 웹서버로 전송해보자

기존 방식

server1.js

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

http.createServer((req, res) => {
  res.write('<h1>Hello Node!</h1>');
  res.end('<p>Hello Server!</p>');
}).listen(8080, () => {
  console.log('8080번 포트에서 서버 대기 중입니다!');
});
```