

# 네트워크서비스 프로토콜

## 6주차 1

» 소프트웨어학부

» 김형균 교수



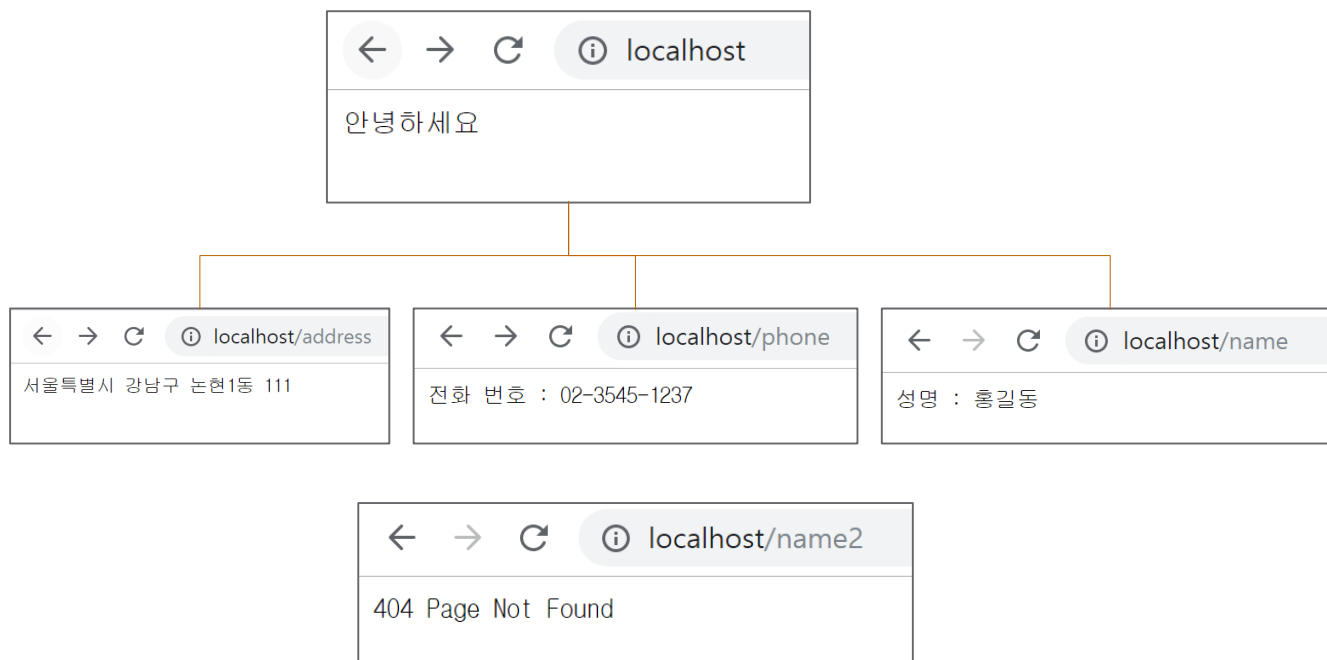
# 수업에 들어가며

- 복습
  - url, path
- 오늘 학습 내용
  - Url query의 값 가져오기
  - Url의 Query string 이용한 동적 웹페이지 만들기
  - 3.6 파일 시스템 접근하기
  - 본문 내용도 동적으로 변하는 웹페이지

# 복습



- » http 서버를 이용해 아래와 같이 브라우저에 접속한 상태에서 요청한 자원이 소스코드에서 정의한 `/`, `/address`, `/phone`, `/name` 에 메시지를 브라우저에 출력해 보자.(응답포트:80)
- » 주소를 요청시 콘솔에 요청된 url과 해당 path를 각각 출력한다.



```
Debugger attached.  
Server is running...  
/  
resource path=/  
/address  
resource path=/address  
/phone  
resource path=/phone  
/name  
resource path=/name
```



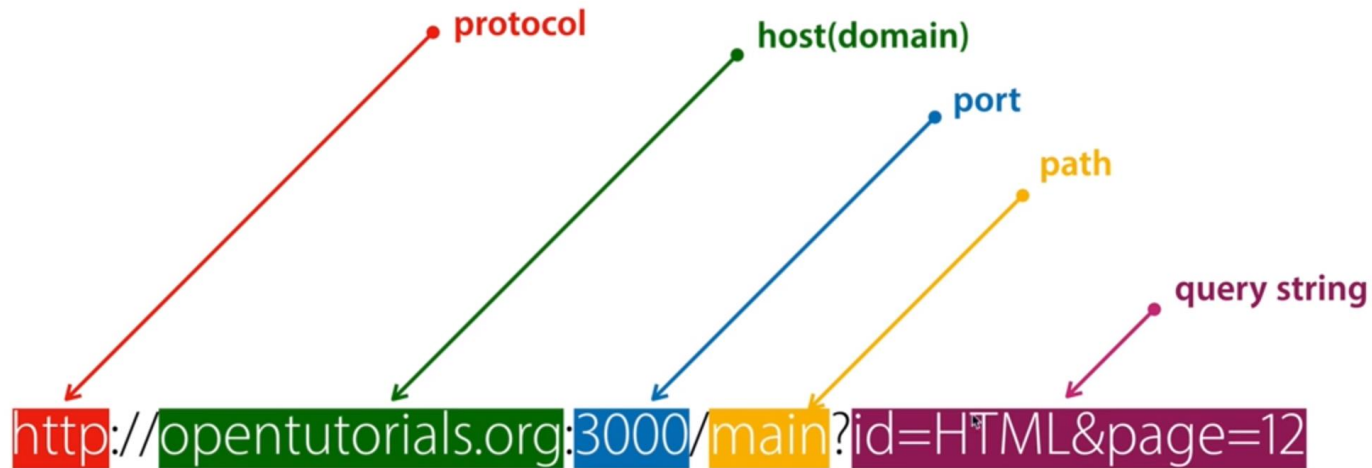
```
const http = require('http');
const url = require('url');

http.createServer((request, response) => {
  console.log(request.url); // 1. 실제 요청한 주소전체를 콘솔에 출력
  const parsedUrl = url.parse(request.url);
  const resource = parsedUrl.pathname;
  // 2. parsing 된 url 중에 서버URI에 해당하는 pathname 만 따로 저장
  console.log('resource path=%s', resource);

  response.writeHead(200, {'Content-Type': 'text/plain; charset=utf-8'});
  // 3. 리소스에 해당하는 문자열이 아래와 같으면 해당 메시지를 클라이언트에 전달
  if(resource == '/') response.end('안녕하세요');
  else if(resource == '/address') response.end('서울특별시 강남구 논현1동 111');
  else if(resource == '/phone') response.end('전화 번호 : 02-3545-1237');
  else if(resource == '/name') response.end('성명 : 홍길동');
  else response.end('404 Page Not Found');
}).listen(80, () => {
  console.log('Server is running...');
});
```



# Url의 구성



`http://localhost/?id=HTML`

**Query string**



# Url query의 값 가져오기

» Url의 Query string 에서 요청 파라미터 중 query의 값 가져오기

» **url.parse(urlStr, [parseQueryString], [slashesDenoteHost])**

- url 문자열(urlStr)을 url 객체로 변환하여 리턴합니다.
- parseQueryString과 slashesDenoteHost는 기본값으로 **false**입니다.
- **parseQueryString**
  - true : url 객체의 query 속성을 객체 형식으로 가져옵니다.(querystring 모듈을 사용합니다.)
  - false : url 객체의 query 속성을 문자열 형식으로 가져옵니다.
- **slashesDenoteHost**
  - true : urlStr이 '//foo/bar' 인 경우 foo는 host, /bar는 path로 인식합니다.
  - false : urlStr이 '//foo/bar' 인 경우 //foo/bar 전체를 path로 인식하고 host는 null 입니다.

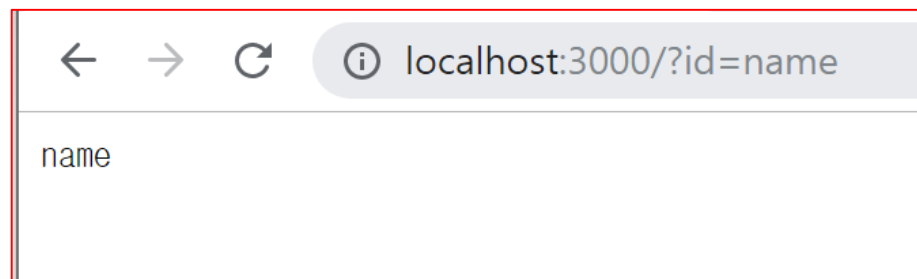
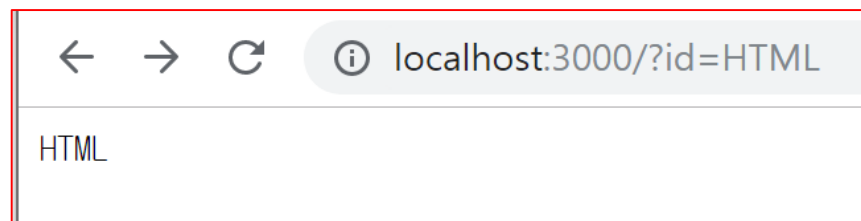
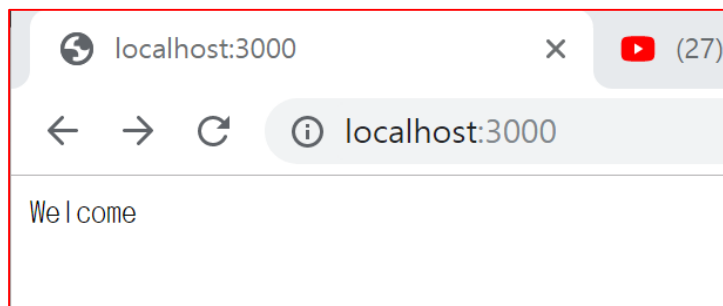
» Url의 query 값 가져오기

: url.parse(**urlStr**, **true**).**query**

: 쿼리스트링의 id 속성 값으로 가져오기

# 실습

» Url의 query 값을 이용해 브라우저에 다음과 같이 출력해보자.





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

var app = http.createServer(function(request, response){
  var _url = request.url;
  var queryData = url.parse(_url, true).query;

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





# Url의 Query string 이용한 동적 웹페이지 만들기

(27) Acoustic Songs 2019 - App - 동

localhost:8080

## WELCOME Home

- [HTML](#)
- [CSS](#)
- [JavaScript](#)

### Home

Node.js 웹 서버

만들 준비되셨나요?

(27) Acoustic Songs 2019 - App - 동적인 웹페이지

localhost:8080/?id=HTML

## WELCOME HTML

- [HTML](#)
- [CSS](#)
- [JavaScript](#)

### HTML

Node.js 웹 서버

만들 준비되셨나요?

(27) Acoustic Songs 2019 - App - 동적

localhost:8080/?id=CSS

## WELCOME CSS

- [HTML](#)
- [CSS](#)
- [JavaScript](#)

### CSS

Node.js 웹 서버

만들 준비되셨나요?

(27) Acoustic Songs 2019 - App - 동적인 웹페

localhost:8080/?id=JavaScript

## WELCOME JavaScript

- [HTML](#)
- [CSS](#)
- [JavaScript](#)

### JavaScript

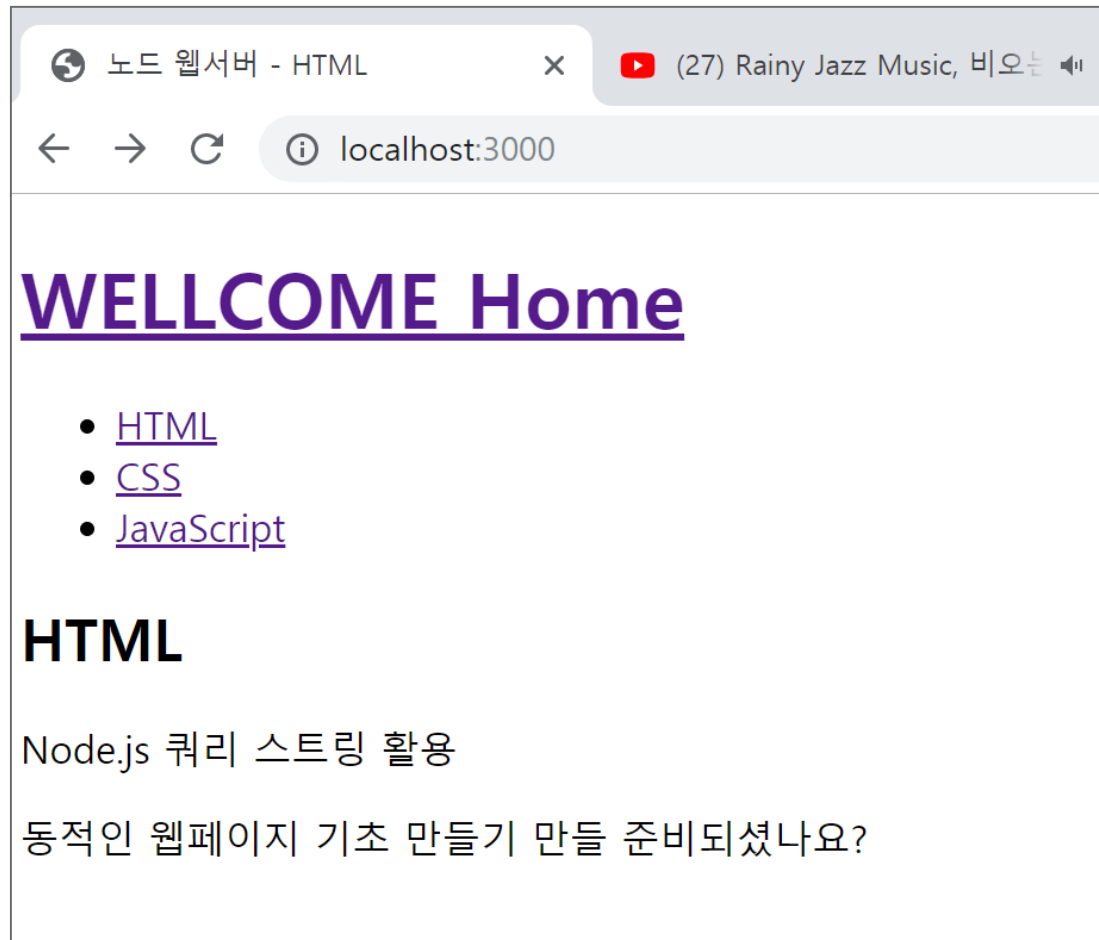
Node.js 웹 서버

만들 준비되셨나요?



# Url의 Query string 이용한 동적 웹페이지 만들기

## » Html 페이지 작성





# Url의 Query string 이용한 동적 웹페이지 만들기

```
<!doctype html>
<html>
  <head>
    <title>노드 웹서버 - HTML</title>
    <meta charset="utf-8">
  </head>
  <body>
    <h1><a href="/">WELLCOME Home</a></h1>
    <ul>
      <li><a href="/1.html">HTML</a></li>
      <li><a href="/2.html">CSS</a></li>
      <li><a href="/3.html">JavaScript</a></li>
    </ul>
    <h2>HTML</h2>
    <p>Node.js 쿼리 스트링 활용</p>
    <p>동적인 웹페이지 기초 만들기 만들 준비되셨나요?</p>
  </body>
</html>
```



# Url의 Query string 이용한 동적 웹페이지 만들기

» `` (백틱)을 이용해 html문서를 node.js 문서내에 포함

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

var app = http.createServer(function(request, response){
  var _url = request.url;
  var queryData = url.parse(_url, true).query;

  if(_url == '/'){
    var template = ``;
  }
  if(_url == '/favicon.ico'){
    return response.writeHead(404);
  }
  response.writeHead(200);
  response.end(template);
});
app.listen(3000);
```

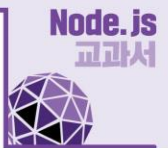
```

var http = require('http');
var fs = require('fs');
var url = require('url');

var app = http.createServer(function(request,response){
  var _url = request.url;
  var queryData = url.parse(_url, true).query;

  if(_url == '/'){
    var template = `
      <!doctype html>
      <html>
        <head>
          <title>노드 웹서버 - HTML</title>
          <meta charset="utf-8">
        </head>
        <body>
          <h1><a href="/">WELLCOME Home</a></h1>
          <ul>
            <li><a href="/1.html">HTML</a></li>
            <li><a href="/2.html">CSS</a></li>
            <li><a href="/3.html">JavaScript</a></li>
          </ul>
          <h2>HTML</h2>
          <p>Node.js 쿼리 스트링 활용</p>
          <p>동적인 웹페이지 기초 만들기 만들 준비되셨나요?</p>
        </body>
      </html>
    `;
  }
  if(_url == '/favicon.ico'){
    return response.writeHead(404);
  }
  response.writeHead(200);
  response.end(template);
});
app.listen(3000);

```





## Url의 Query string 이용한 동적 웹페이지 만들기

- » Url의 Query string id값을 이용해 문서 동적으로 변환하기
- » title변수에 Url의 Query string id값 저장

```
var app = http.createServer(function(request,response){  
  var _url = request.url;  
  var queryData = url.parse(_url, true).query;  
  var title = queryData.id;  
  if(_url == '/'){  
    title = 'Home';  
  }  
  if(_url == '/favicon.ico'){  
    return response.writeHead(404);  
  }  
}
```



# Url의 Query string 이용한 동적 웹페이지 만들기

» `` (백틱)을 이용해 포함시킨 html문서를 블록 밖으로 뺀다

```
var app = http.createServer(function(request,response){
  var _url = request.url;
  var queryData = url.parse(_url, true).query;
  var title = queryData.id;
  if(_url == '/'){
    title = 'Home';
  }
  if(_url == '/favicon.ico'){
    return response.writeHead(404);
  }
  response.writeHead(200);
  var template = `
    <!doctype html>
    <html>

    </html>
  `;
  response.end(template);
});
app.listen(8080);
```



# Url의 Query string 이용한 동적 웹페이지 만들기

» Html 문서에서 동적으로 변환 부분을 title변수로 대체

```
<!doctype html>
<html>
  <head>
    <title>노드 웹서버 -HTML</title>
    <meta charset="utf-8">
  </head>
  <body>
    <h1><a href="/">WELLCOME Home</a></h1>
    <ul>
      <li><a href="/1.html">HTML</a></li>
      <li><a href="/2.html">CSS</a></li>
      <li><a href="/3.html">JavaScript</a></li>
    </ul>
    <h2>HTML</h2>
    <p>Node.js 쿼리 스트링 활용</p>
    <p>동적인 웹페이지 기초 만들기 만들 준비되셨나요?</p>
  </body>
</html>
```





# Url의 Query string 이용한 동적 웹페이지 만들기

» <a href = > 링크 속성값을 url Query string 값으로 대체

```
<!doctype html>
<html>
  <head>
    <title>노드 웹서버 - HTML</title>
    <meta charset="utf-8">
  </head>
  <body>
    <h1><a href="/">WELLCOME Home</a></h1>
    <ul>
      <li><a href="/1.html">HTML</a></li>
      <li><a href="/2.html">CSS</a></li>
      <li><a href="/3.html">JavaScript</a></li>
    </ul>
    <h2>HTML</h2>
    <p>Node.js 쿼리 스트링 활용</p>
    <p>동적인 웹페이지 기초 만들기 만들 준비되셨나요?</p>
  </body>
</html>
```

```
response.writeHead(200);  
var template = `  
  <!doctype html>  
  <html>  
      
  </html>  
`;  
response.end(template);
```

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

```
var app = http.createServer(function(request, response){
```

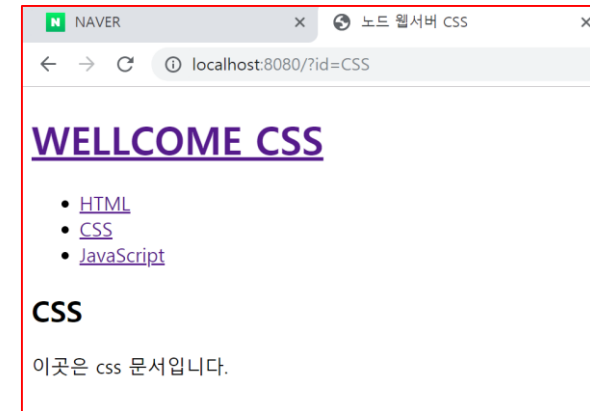
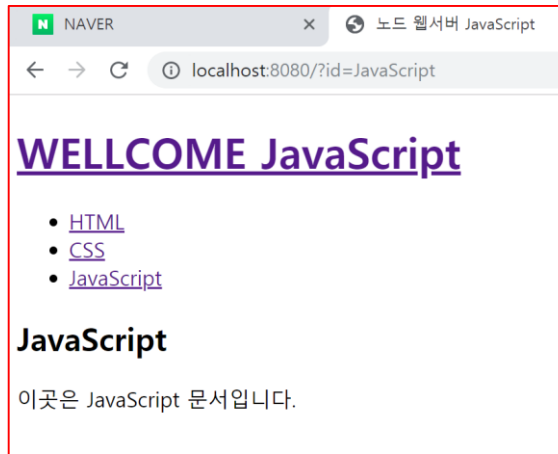
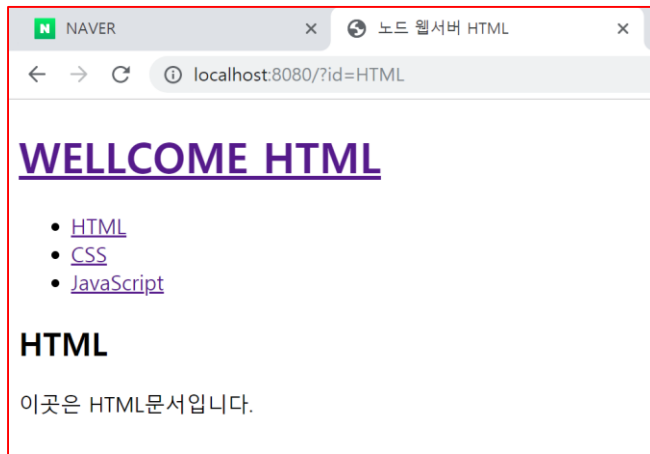
```
});  
app.listen(8080);
```



# 본문 내용도 동적으로 변하는 웹페이지



## » 파일 시스템 접근필요



## 3.6 파일 시스템 접근하기

---



# 1. fs

## » 파일 시스템에 접근하는 모듈

- 파일/폴더 생성, 삭제, 읽기, 쓰기 가능
- 웹 브라우저에서는 제한적이었으나 노드는 권한을 가지고 있음
- 파일 읽기 예제(결과의 버퍼는 뒤에서 설명함)

readme.txt

저를 읽어주세요.

readFile.js

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

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

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.



## 2. fs로 파일 만들기

### » 파일을 만드는 예제

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.
```



# 3. 동기 메서드와 비동기 메서드

» 노드는 대부분의 내장 모듈 메서드를 비동기 방식으로 처리

- 비동기는 코드의 순서와 실행 순서가 일치하지 않는 것을 의미
- 일부는 동기 방식으로 사용 가능
- 우측 코드 콘솔 예측해보기

readme2.txt

저를 여러 번 읽어보세요.

async.js

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

console.log('시작');
fs.readFile('./readme2.txt', (err, data) => {
  if (err) {
    throw err;
  }
  console.log('1번', data.toString());
});
fs.readFile('./readme2.txt', (err, data) => {
  if (err) {
    throw err;
  }
  console.log('2번', data.toString());
});
fs.readFile('./readme2.txt', (err, data) => {
  if (err) {
    throw err;
  }
  console.log('3번', data.toString());
});
console.log('끝');
```



## 4. 동기 메서드와 비동기 메서드

» 이전 예제를 여러 번 실행해보기

- 매 번 순서가 다르게 실행됨
- 순서에 맞게 실행하려면?

콘솔

\$ node async

시작

끝

2번 저를 여러 번 읽어보세요.

3번 저를 여러 번 읽어보세요.

1번 저를 여러 번 읽어보세요.

```
PS C:\test\nodejs-book-master\ch3\3.6> node async.js
시작
끝
2번 저를 여러 번 읽어보세요.
1번 저를 여러 번 읽어보세요.
3번 저를 여러 번 읽어보세요.
PS C:\test\nodejs-book-master\ch3\3.6> node async.js
시작
끝
1번 저를 여러 번 읽어보세요.
3번 저를 여러 번 읽어보세요.
2번 저를 여러 번 읽어보세요.
PS C:\test\nodejs-book-master\ch3\3.6> node async.js
시작
끝
1번 저를 여러 번 읽어보세요.
2번 저를 여러 번 읽어보세요.
3번 저를 여러 번 읽어보세요.
PS C:\test\nodejs-book-master\ch3\3.6> █
```



## 5. 동기 메서드 사용하기

sync.js

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

console.log('시작');
let data = fs.readFileSync('./readme2.txt');
console.log('1번', data.toString());
data = fs.readFileSync('./readme2.txt');
console.log('2번', data.toString());
data = fs.readFileSync('./readme2.txt');
console.log('3번', data.toString());
console.log('끝');
```

콘솔

```
$ node sync
```

시작

1번 저를 여러 번 읽어보세요.

2번 저를 여러 번 읽어보세요.

3번 저를 여러 번 읽어보세요.

끝



# 6. 비동기 메서드로 순서 유지하기

## » 콜백 형식 유지

- 코드가 우측으로 너무 들어가는 현상 발생(콜백 헬)

asyncOrder.js

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

console.log('시작');
fs.readFile('./readme2.txt', (err, data) => {
  if (err) {
    throw err;
  }
  console.log('1번', data.toString());
  fs.readFile('./readme2.txt', (err, data) => {
    if (err) {
      throw err;
    }
    console.log('2번', data.toString());
    fs.readFile('./readme2.txt', (err, data) => {
      if (err) {
        throw err;
      }
      console.log('3번', data.toString());
    });
  });
});
console.log('끝');
```

콘솔

\$ node asyncOrder

시작

끝

1번 저를 여러 번 읽어보세요.

2번 저를 여러 번 읽어보세요.

3번 저를 여러 번 읽어보세요.



# 본문 내용도 동적으로 변하는 웹페이지


## » 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);`
- Filename : 동적으로 변하도록 처리
- [options] : utf-8 : 한글처리

## » 본문의 내용을 다음과 같이 파일형태로 구분 저장

C > 로컬 디스크 (C:) > test > data

이름

 javascript

 css

 html



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

var app = http.createServer(function(request, response){
  var _url = request.url;
  var queryData = url.parse(_url, true).query;
  var pathname = url.parse(_url, true).pathname;
  var title = queryData.id;

  if(pathname == '/'){
    [REDACTED]
  } else {
    response.writeHead(404);
    response.end('Not found');
  }
});
app.listen(3000);
```

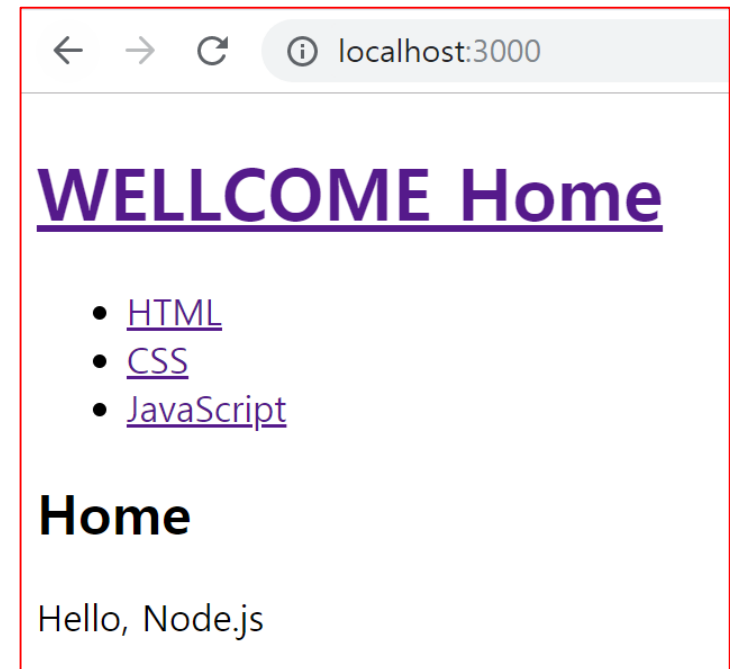
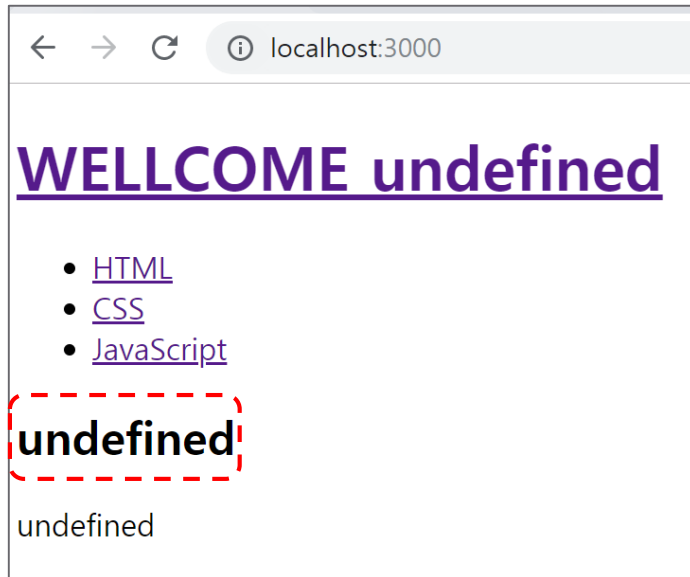
```
if(pathname == '/') {
```

```
});
```

# HomePage undefined 문제 해결


## » Hint

- 



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

var app = http.createServer(function(request, response){
  var _url = request.url;
  var queryData = url.parse(_url, true).query;
  var pathname = url.parse(_url, true).pathname;

  if(pathname == '/'){
    
  } else {
    response.writeHead(404);
    response.end('Not found');
  }
});
app.listen(3000);
```



