

네트워크서비스 프로토콜

9주차 1

» 소프트웨어학부

» 김형균 교수



수업에 들어가며

- 중간고사 풀이
- 복습
 - 파일생성과 리다이렉션
- 오늘 학습할 내용
 - 글수정(update) - 수정링크생성
 - 글수정(update) - 수정할 정보 전송
 - 글수정(update) - 수정된 내용으로 변경
 - 글수정(update) - fs.rename()
 - 글 삭제 - 삭제버튼 구현
 - 삭제버튼 위치 지정
 - 글 삭제 - 처리 프로세스
 - 객체 메서드를 이용해서 템플릿 기능 정리 하기



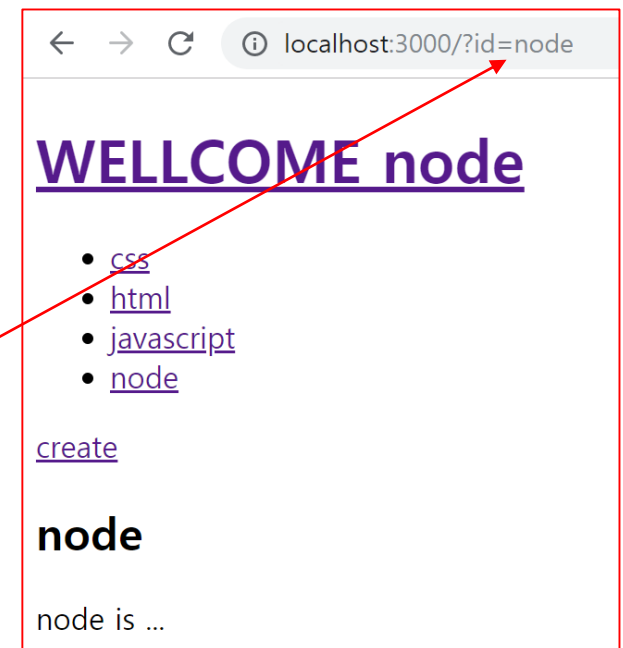
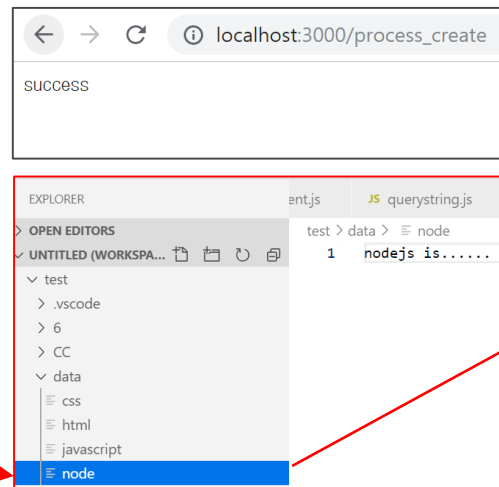
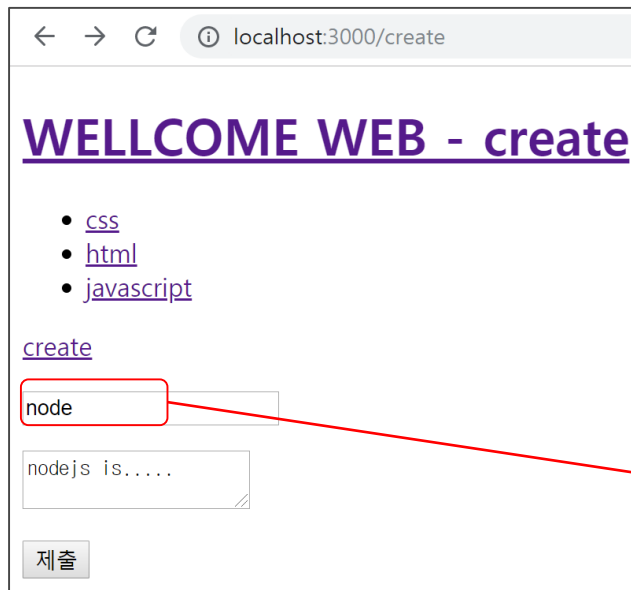
복습 : 파일생성과 리다이렉션

» 전송된 POST 데이터를 받아서 파일에 저장

- fs.writeFile()

» 그 결과 페이지로 리다이렉션하기

- 특정 페이지로 강제로 보내고자 할 때 응답 헤더의 Location 속성을 사용





복습 : 파일생성과 리다이렉션

```
1  var http = require('http');
2  var fs = require('fs');
3  var url = require('url');
4  var qs = require('querystring');
5
6  > function templateHTML(title, list, body){ ...
22  }
23
24  > function templateList(filelist){ ...
33  }
34
35  var app = http.createServer(function(request, response){
36      var _url = request.url;
37      var queryData = url.parse(_url, true).query;
38      var pathname = url.parse(_url, true).pathname;
39      if(pathname == '/'){
40  >         if(queryData.id == undefined){ ...
49  >         } else { ...
59         }
60  >     } else if(pathname == '/create'){ ...
78  >     } else if(pathname == '/process_create'){ ...
92     } else {
93         response.writeHead(404);
94         response.end('Not found');
95     }
96  });
97  app.listen(3000);
```

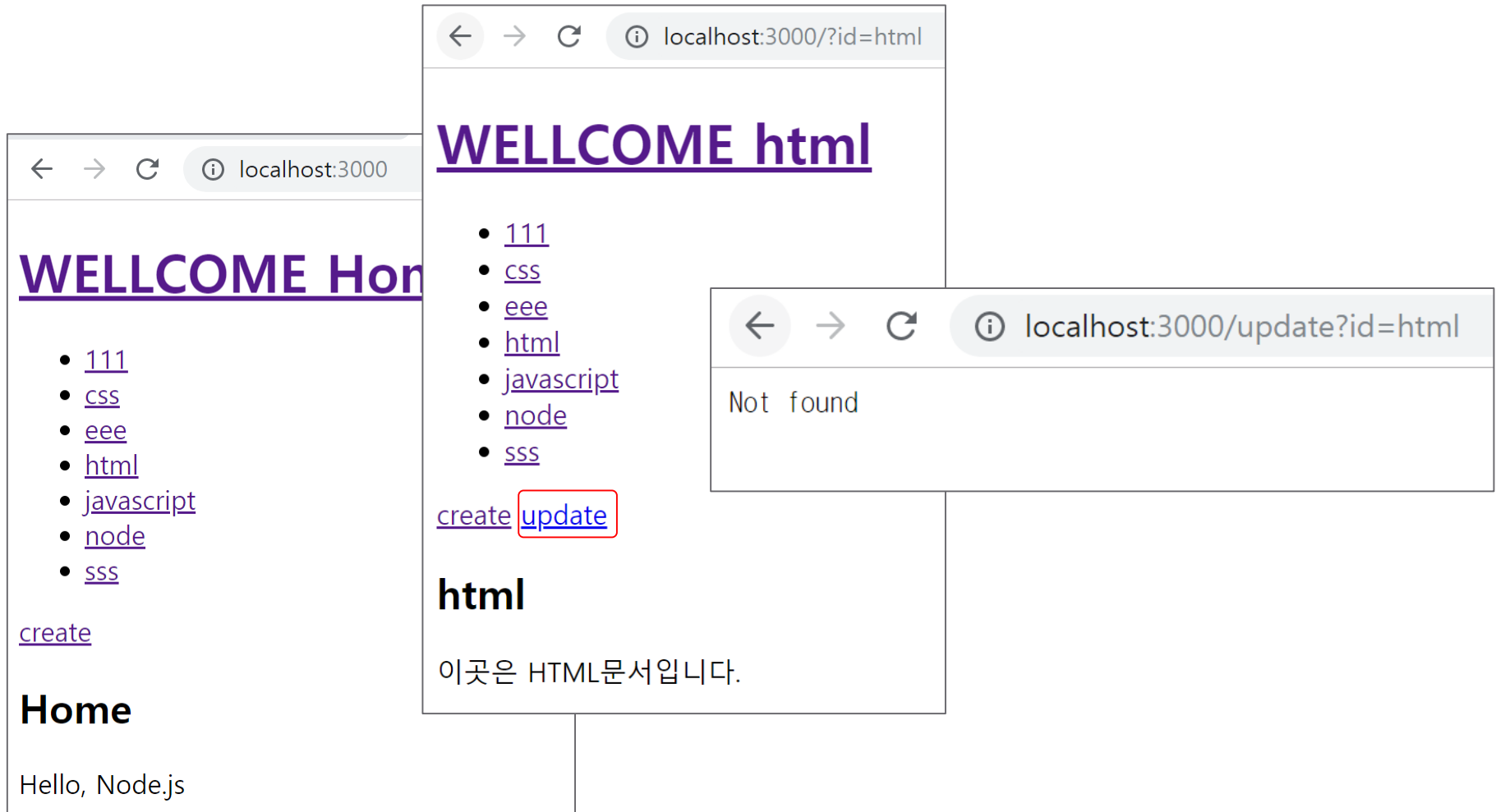


복습 : 파일생성과 리다이렉션

```
} else if(pathname == '/process_create'){
  var body = '';
  request.on('data', function(data){
    body = body + data;
  });
  request.on('end', function(){
    var post = qs.parse(body);
    var title = post.title;
    var description = post.description;
    fs.writeFile(`data/${title}`, description, 'utf8', function(err){
      response.writeHead(302, {Location: `/?id=${title}`});
      response.end();
    })
  });
} else {
  response.writeHead(404);
  response.end('Not found');
}
});
app.listen(3000);
```



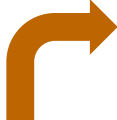
글수정(update) - 수정링크생성





```
function templateHTML(title, list, body){  
  return `  
    <!doctype html>  
    <html>  
    <head>  
    <title>노드 웹서버 ${title}</title>  
    <meta charset="utf-8">  
    </head>  
    <body>  
    <h1><a href="/">WELLCOME ${title}</a></h1>  
    ${list}  
    <a href="/create">create</a>  
    ${body}  
    </body>  
    </html>  
  `;  
}
```

```
function templateHTML(title, list, body, control){  
  return `  
    <!doctype html>  
    <html>  
    <head>  
    <title>노드 웹서버 ${title}</title>  
    <meta charset="utf-8">  
    </head>  
    <body>  
    <h1><a href="/">WELLCOME ${title}</a></h1>  
    ${list}  
    ${control}  
    ${body}  
    </body>  
    </html>  
  `;  
}
```



```
if(pathname == '/'){
  if(queryData.id == undefined){
    fs.readdir('./data', function(error, filelist){
      var title = 'Home';
      var description = 'Hello, Node.js';
      var list = templateList(filelist);
      var template = templateHTML(title, list, `

## ${title}</h2>${description}`, `


```

```
if(pathname == '/'){
  if(queryData.id == undefined){
    fs.readdir('./data', function(error, filelist){
      var title = 'Home';
      var description = 'Hello, Node.js';
      var list = templateList(filelist);
      var template = templateHTML(title, list, `

## ${title}</h2>${description}`); response.writeHead(200); response.end(template); }) } }



```




```
} else {  
  fs.readdir('./data', function(error, filelist){  
    fs.readFile(`data/${queryData.id}`, 'utf8', function(err, description){  
      var title = queryData.id;  
      var list = templateList(filelist);  
      var template = templateHTML(title, list,  
        `

## ${title}</h2>${description}`, ` ); response.writeHead(200); response.end(template); }); }); }


```



```
} else {  
  fs.readdir('./data', function(error, filelist){  
    fs.readFile(`data/${queryData.id}`, 'utf8', function(err, description){  
      var title = queryData.id;  
      var list = templateList(filelist);  
      var template = templateHTML(title, list, `

## ${title}</h2>${description}`); response.writeHead(200); response.end(template); }); }); }

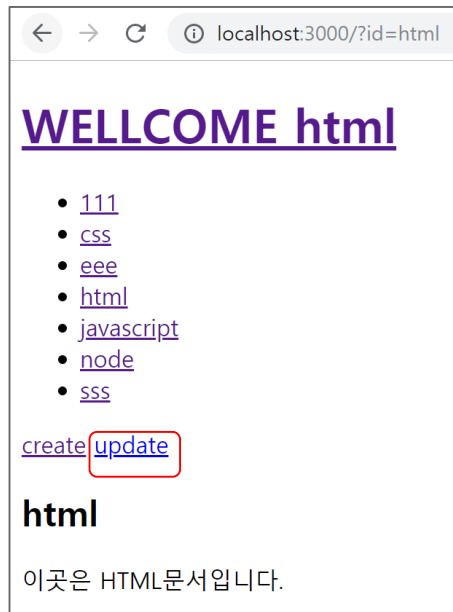

```



글수정(update) - 수정할 정보 전송

» 주의점

- 현재 문서파일을 열어 새로 입력한 내용으로 파일명과 파일내용을 변경해야 함





글수정(update) - 수정할 정보 전송



```
} else if(pathname == '/process_create'){  
  var body = '';  
  request.on('data', function(data){  
    body = body + data;  
  });  
  request.on('end', function(){  
    var post = qs.parse(body);  
    var title = post.title;  
    var description = post.description;  
    fs.writeFile(`data/${title}`, description, 'utf8', function(err){  
      response.writeHead(302, {Location: `/?id=${title}`});  
      response.end();  
    });  
  });  
} else {  
  response.writeHead(404);  
  response.end('Not found');  
}  
});  
app.listen(3000);
```

```
} else if(pathname == '/process_create'){  
  var body = '';  
  request.on('data', function(data){  
    body = body + data;  
  });  
  request.on('end', function(){  
    var post = qs.parse(body);  
    var title = post.title;  
    var description = post.description;  
    fs.writeFile(`data/${title}`, description, 'utf8', function(err){  
      response.writeHead(302, {Location: `/?id=${title}`});  
      response.end();  
    });  
  });  
} else if(pathname == '/update'){ ...  
} else {  
  response.writeHead(404);  
  response.end('Not found');  
}  
});  
app.listen(3000);
```



글수정(update) - 수정할 정보 전송

» <input>

- 웹에서의 폼은 사용자 입력부분과 서버 전송 버튼으로 나뉨
- 사용자 입력 부분은 거의 <input> 태그를 이용해 처리 함
- <input>태그의 type=""속성을 통해 입력하는 내용이 어떤 정보인지를 지정 함
- type=""속성 값에 따라 사용할 수 있는 속성들도 다름
- <input type="유형" 속성="속성 값">

» type="hidden"

- 숨겨진 입력필드를 정의
- 즉, 화면상에 폼에는 보이지 않지만, 폼을 서버로 전송할 때 함께 전송되는 요소
- 예를 들어, 회원가입 폼에서 가입 경로, 날짜와 같이 굳이 사용자가 입력하지 않아도 알 수 있는 정보들을 서버로 넓길때 사용
- <input type="hidden" name="..." value="서버 전송 값">



글수정(update) - 수정할 정보 전송

<form action="/update_process" method="post">

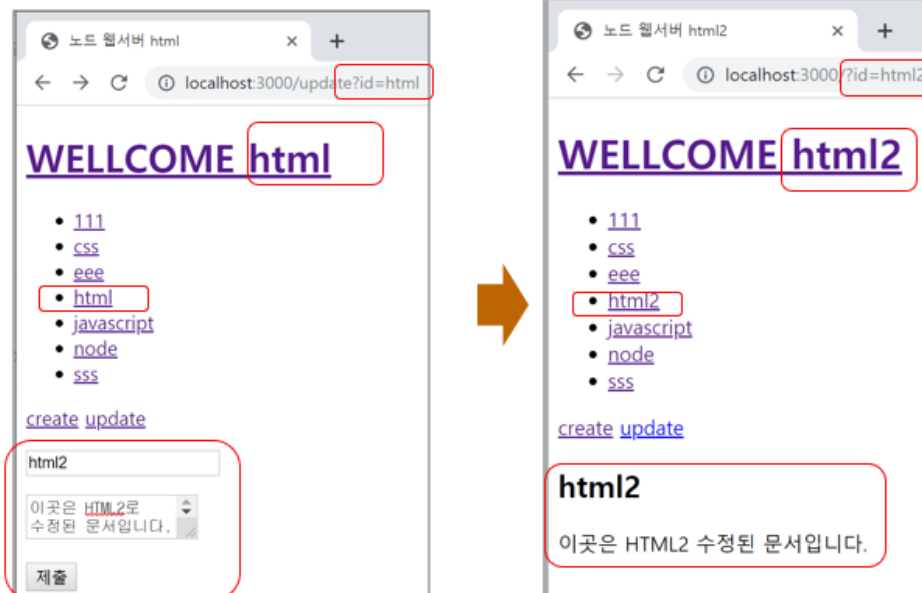
- 현재 문서파일을 열어 새로 입력한 내용으로 파일명과 파일내용을 변경해야 함
- 현재 선택된 파일의 정보를 히든으로 서버에 전송

name= "id" value= "\${title}"

- 입력한 정보(변경할 파일명)는

name="title" value="\${title}"

- 입력한 정보(변경할 파일내용)는 **name="description"** 으로 전송





글수정(update) - 수정할 정보 전송

```
} else if(pathname == '/update'){
  fs.readdir('./data', function(error, filelist){
    fs.readFile(`data/${queryData.id}`, 'utf8', function(err, description){
      var title = queryData.id;
      var list = templateList(filelist);
      var template = templateHTML(title, list,
      

A large, solid light green rectangular area that appears to be a placeholder or a redacted section of the code.


      );
      response.writeHead(200);
      response.end(template);
    });
  });
} else {
```

글수정(update) – 수정된 내용으로 변경 파일명, 본문내용 수정



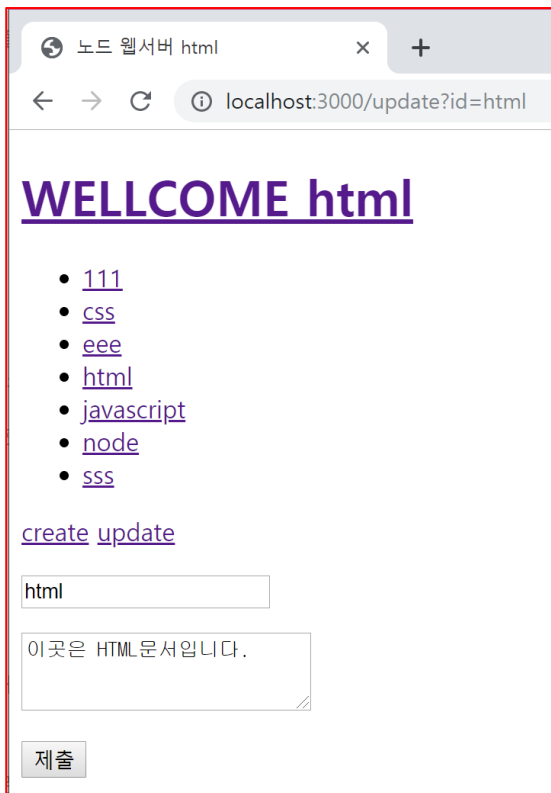
```
var app = http.createServer(function(request,response){
  var _url = request.url;
  var queryData = url.parse(_url, true).query;
  var pathname = url.parse(_url, true).pathname;
  if(pathname == '/'){...
} else if(pathname == '/create'){...
} else if(pathname == '/process_create'){...
} else if(pathname == '/update'){...
} else if(pathname === '/update_process'){...
} else {
  response.writeHead(404);
  response.end('Not found');
}
});
app.listen(3000);
```



fs.rename()

» fs.rename(기존 경로, 새 경로, 콜백)

- 파일의 이름을 바꾸는 메서드
- 기존 파일 위치와 새로운 파일 위치 작성
- 반드시 같은 폴더를 지정할 필요는 없으므로 잘라내기 같은 기능도 가능





```
} else if(pathname === '/update_process'){
```

```
} else {
```

글삭제 - 삭제버튼 구현

» fs.unlink(경로, 콜백)

- 파일을 지울 수 있습니다.
- 파일이 없다면 에러가 발생하므로 먼저 파일이 있는지를 꼭 확인해야 합니다.

» 삭제는 하이퍼링크로 구현하지 않는 것이 좋다

The screenshot illustrates the implementation of a delete button for a file named 'html2'. It shows two browser windows and a VS Code editor.

Left Browser Window (localhost:3000/?id=html2):

- Page Title: **WELLCOME html2**
- Links: [111](#), [css](#), [eee](#), [html2](#) (highlighted), [javascript](#), [node](#), [sss](#)
- Buttons: [create](#), [update](#), [delete](#) (highlighted)
- Text: **html2**
- Text: 이곳은 HTML2 수정된 문서입니다.

Right Browser Window (localhost:3000):

- Page Title: **WELLCOME Home**
- Links: [111](#), [css](#), [eee](#), [javascript](#), [node](#), [sss](#)
- Buttons: [create](#)
- Text: **Home**
- Text: Hello, Node.js

VS Code Editor:

- File Explorer: Shows a list of files including [data](#), [111](#), [css](#), [eee](#), [javascript](#), [node](#), [sss](#), [fast_web-recipe](#), [javascript](#), [js-basic-book-master](#), [learn-express](#), [learn-sequelize](#), [nodebook-master](#), and [nodejs](#). The file [html2](#) is highlighted.



삭제버튼 위치 지정

```
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 == '/'){
    if(queryData.id == undefined){...
    } else {
      fs.readdir('./data', function(error, filelist){
        fs.readFile(`data/${queryData.id}`, 'utf8', function(err, description){
          var title = queryData.id;
          var list = templateList(filelist);
          var template = templateHTML(title, list,
            `

## ${title}</h2>${description}`, `


```

```
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 == '/'){
    if(queryData.id == undefined){...
    } else {
      fs.readdir('./data', function(error, filelist){
        fs.readFile(`data/${queryData.id}`, 'utf8', function(err, description){
          var title = queryData.id;
          var list = templateList(filelist);
          var template = templateHTML(title, list,
            `

## ${title}</h2>${description}`, `


```



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

2



객체의 메서드-3주차 수업내용 복습

- 메서드는 객체가 가지고 있는 동작
- 메서드를 수행하기 위해서는 객체를 통해서 해당 메서드를 수행
- 함수와 메서드의 차이 : 함수는 그 동작을 수행하기 위해 객체에게 어떤을 동작을 수행하라고 명령하지 않아도 된다. 그이유는 함수자체가 그 동작을 정의한 함수객체이기 때문에 자기 자신을 수행하는 것
- 형식)

```
메서드명 : function( ) {  
    실행명령 1;  
    :  
    return 리턴값;  
}
```



```
var circle = {  
  center : {x:1.0, y:2.0} ,  
  radius : 2.5,  
  area : function(){  
    return Math.PI * this.radius * this.radius;  
  },  
  round : function(){  
    return 2 * Math.PI * this.radius;  
  }  
};  
  
console.log("원의 중심좌표는 (" + circle.center.x + "," + circle.center.y + ")");  
console.log("원의 반지름은 " + circle.radius );  
console.log("원의 면적은 " + circle.area().toFixed(2) + "입니다.");  
console.log("원의 둘레는 " + circle.round().toFixed(2) + "입니다.");
```



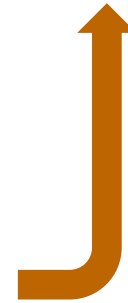

객체 메서드를 이용해서 템플릿 기능 정리 하기

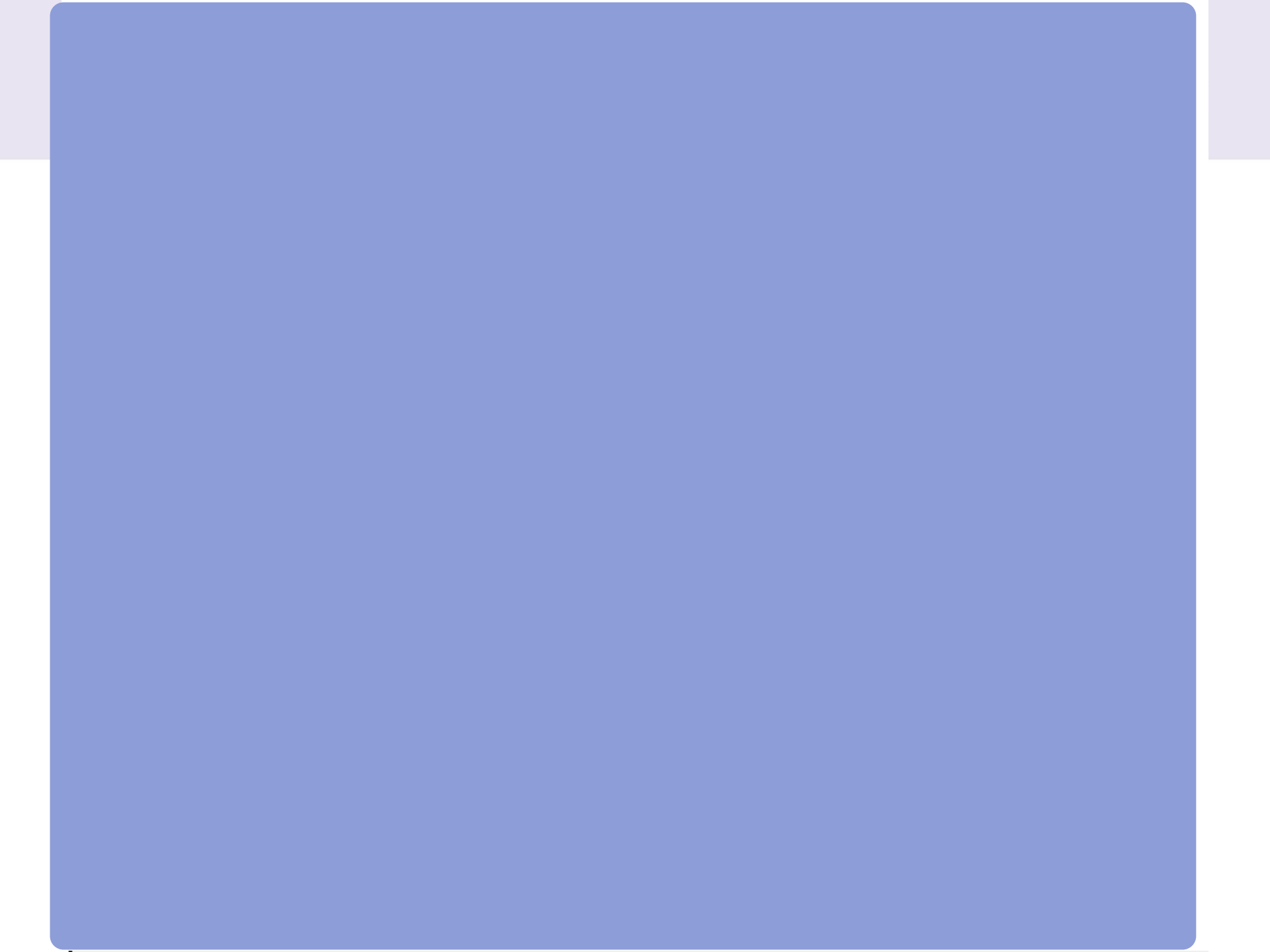
```
function templateHTML(title, list, body, control){  
  return `  
    <!doctype html>  
    <html>  
    <head>  
    <title>노드 웹서버 ${title}</title>  
    <meta charset="utf-8">  
    </head>  
    <body>  
    <h1><a href="/">WELLCOME ${title}</a></h1>  
    ${list}  
    ${control}  
    ${body}  
    </body>  
    </html>  
  `;  
}
```



```
var template = {  
    
}
```

```
function templateList(filelist){  
  var list = '<ul>';  
  var i = 0;  
  while(i < filelist.length){  
    list = list + `<li><a href="/?id=${filelist[i]}">${filelist[i]}</a></li>`;  
    i = i + 1;  
  }  
  list = list+'</ul>';  
  return list;  
}
```







객체 메서드 호출후 변수에 저장하기

```
var template = {  
    
}
```