WEB 2 - Node.js

시작

- Node.js: 백엔드 서버 구축하기

```
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);
  response.end(fs.readFileSync(__dirname + url));
});

app.listen(3000);
```

- fs: `file system`의 약자 😮
- 실행 방법

```
$ node main.js
```

- 잘 실행됨

WEB

1. HTML
2. CSS
3. JavaScript

CSS

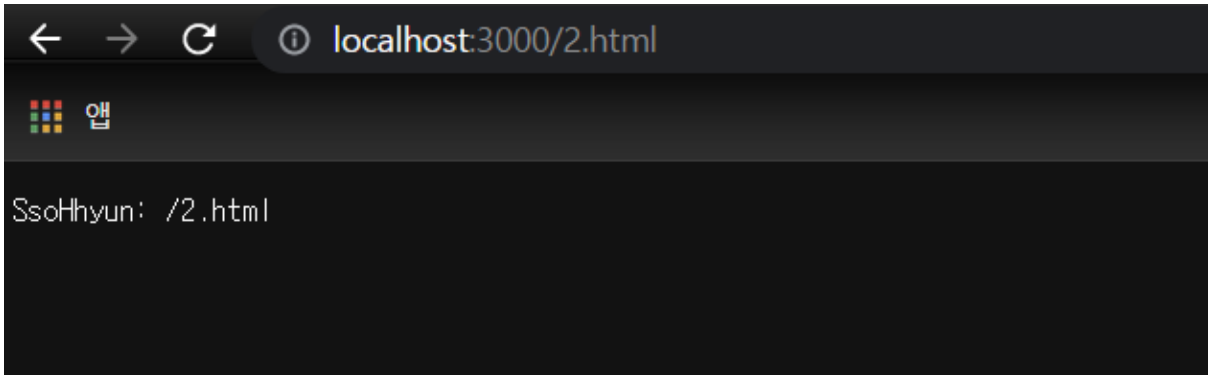
Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language. Although most often used to set the visual style of web pages and user interfaces written in HTML and XHTML, the language can be applied to any XML document, including plain XML, SVG and XUL, and is applicable to rendering in speech, or on other media. Along with HTML and JavaScript, CSS is a cornerstone technology used by most websites to create visually engaging webpages, user interfaces for web applications, and user interfaces for many mobile applications.

```
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);
  // response.end(fs.readFileSync(__dirname + url));
  response.end("SsoHhyun: " + url);
});

app.listen(3000);
```

- 콘솔로 찍으면 실행되고 있는 파일이 보인다

```
$ node main.js
E:\Studying\Node.js\WEB 2 - Node.js\Practice1/1.html
E:\Studying\Node.js\WEB 2 - Node.js\Practice1/coding.jpg
E:\Studying\Node.js\WEB 2 - Node.js\Practice1/3.html
E:\Studying\Node.js\WEB 2 - Node.js\Practice1/1.html
E:\Studying\Node.js\WEB 2 - Node.js\Practice1/coding.jpg
```

- 이름 바꿔 놓으면 내가 적은 이름이 표기된다



💡 **변수**
let: 재할당 가능, 재선언 불가능
const: 재할당, 재선언 불가능

Template Literal

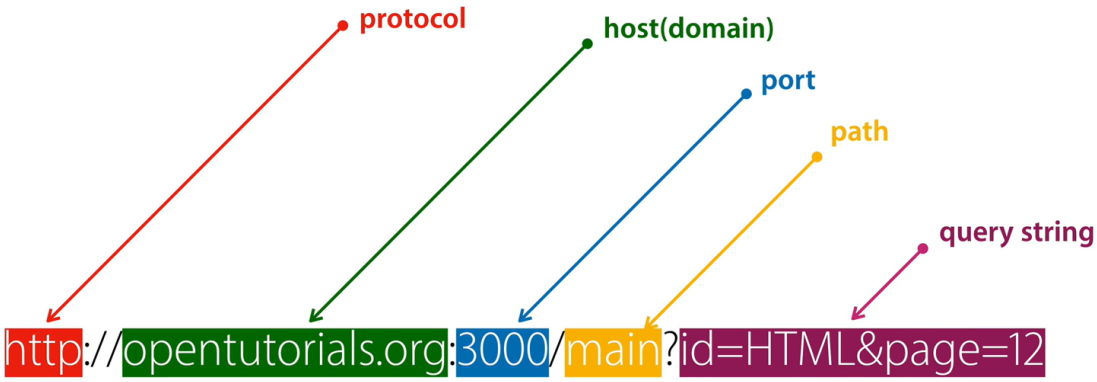
- 리터럴?
  - 어떤 정보를 표현하는 방법
  - 백슬래시(`) 사용. `${}` 쓰기

```
const name = 'SsoHhyun';
const letter = `Dear ${name}, nice to meet you.`;

console.log(letter);

// Dear SsoHhyun, nice to meet you.
```

- 문자열을 더해 줄 필요 없고, 줄 바꿈 시 \n 안 써도 된다는 장점

💡 **모듈(Module)**
기본적으로 제공하는 기능들을 그룹핑해 놓은 각각의 그룹들

URL



- protocol: 통신 규칙, 어떤 방식으로 통신할 것인가?
  - http: 웹 서버와 웹 브라우저가 서로 데이터를 주고받기 위해 만든 통신 규칙
- host(domain): 인터넷에 접속되어 있는 각각의 컴퓨터
  - 특정한 인터넷에 연결되어 있는 컴퓨터의 주소
- port: 3000번 포트에 연결되어 있는 서버와 통신
  - 포트 번호를 생략하면 기본 접속이 80
- path: 컴퓨터 디렉토리 안에 있는 어떤 파일인지
- `query string`: 변경하여 웹 서버에게 데이터 전달
  - `?`로 시작하기로 약속
  - `=`으로 값의 이름과 값 나누기
  - `&`으로 값과 값 사이 연결


```
const http = require("http");
const fs = require("fs");
const app = http.createServer(function (request, response) {
  let url = request.url;
  console.log(url);
  if (url == "/") {
    url = "/index.html";
```

```
  }
  if (url == "/favicon.ico") {
    return response.writeHead(404);
  }
  response.writeHead(200);
  response.end(fs.readFileSync(__dirname + url));
});

app.listen(3000);
```

- `url` 콘솔 찍어 보기

```
/favicon.ico
/index.html
/1.html
/coding.jpg
/2.html
/?id=html
```

- 검색: `nodejs url parse query string`

```
const http = require("http");
const fs = require("fs");
let url = require("url");

const app = http.createServer(function (request, response) {
  let _url = request.url;
  let queryData = url.parse(_url, true).query;
  console.log(queryData);
  if (_url == "/") {
    _url = "/index.html";
  }
  if (_url == "/favicon.ico") {
    return response.writeHead(404);
  }
  response.writeHead(200);
  response.end(fs.readFileSync(__dirname + _url));
});

app.listen(3000);
```

- 아래 url들 `_url`로 변경
- `queryData` 선언 후 console에 찍어 보기
- 결과

```
{ id: 'HTML' }
```

- id 찍어 보기

```
console.log(queryData.id);
```

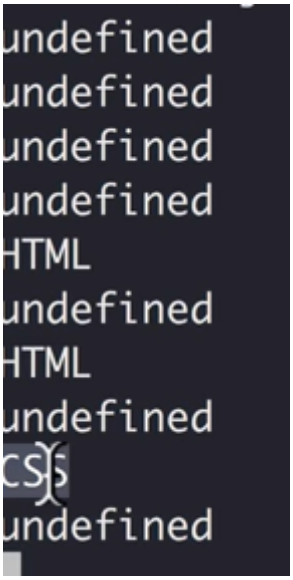
- 결과

```
HTML
```

- 아래를 `queryData`로 변경하면

```
response.end(queryData.id);
```

- `querystring` 값을 변경함에 따라서 찍히는 값이 바뀐다

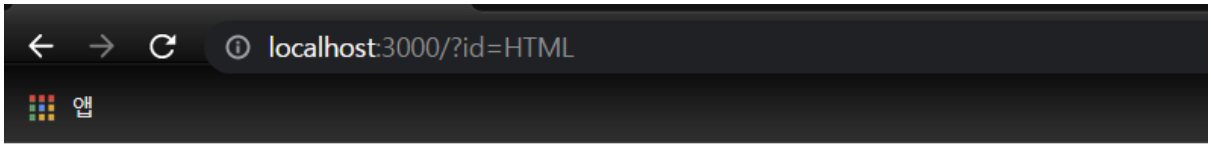


동적인 웹페이지 구현

```
const http = require("http");
const fs = require("fs");
let url = require("url");

const app = http.createServer(function (request, response) {
  let _url = request.url;
  let queryData = url.parse(_url, true).query;
  console.log(queryData.id);
  if (_url == "/") {
    _url = "/index.html";
  }
```

```
  if (_url == "/favicon.ico") {
    return response.writeHead(404);
  }
  response.writeHead(200);
  let template = `
  <!DOCTYPE html>
<html>
  <head>
    <title>WEB1 - ${queryData.id}</title>
    <meta charset="utf-8" />
  </head>
  <body>
    <h1><a href="index.html">WEB</a></h1>
    <ol>
      <li><a href="/?id=HTML">HTML</a></li>
      <li><a href="?/id=CSS">CSS</a></li>
      <li><a href="?/id=JavaScript">JavaScript</a></li>
    </ol>
    <h2>${queryData.id}</h2>
    <p>
      <a
        href="https://www.w3.org/TR/html5/"
        target="_blank"
        title="html5 speicification"
        >Hypertext Markup Language (HTML)</a
      >
      is the standard markup language for
      <strong>creating <u>web</u> pages</strong> and web applications.Web
      browsers receive HTML documents from a web server or from local storage
      and render them into multimedia web pages. HTML describes the structure of
      a web page semantically and originally included cues for the appearance of
      the document.
      <img src="coding.jpg" width="100%" />
    </p>
    <p style="margin-top: 45px">
      HTML elements are the building blocks of HTML pages. With HTML constructs,
      images and other objects, such as interactive forms, may be embedded into
      the rendered page. It provides a means to create structured documents by
      denoting structural semantics for text such as headings, paragraphs,
      lists, links, quotes and other items. HTML elements are delineated by
      tags, written using angle brackets.
    </p>
  </body>
</html>

  `;
  response.end(template);
});

app.listen(3000);
```

- 1.html 내용 복사 후, 제목을 `${queryData.id}`로 바꿔 주기



WEB

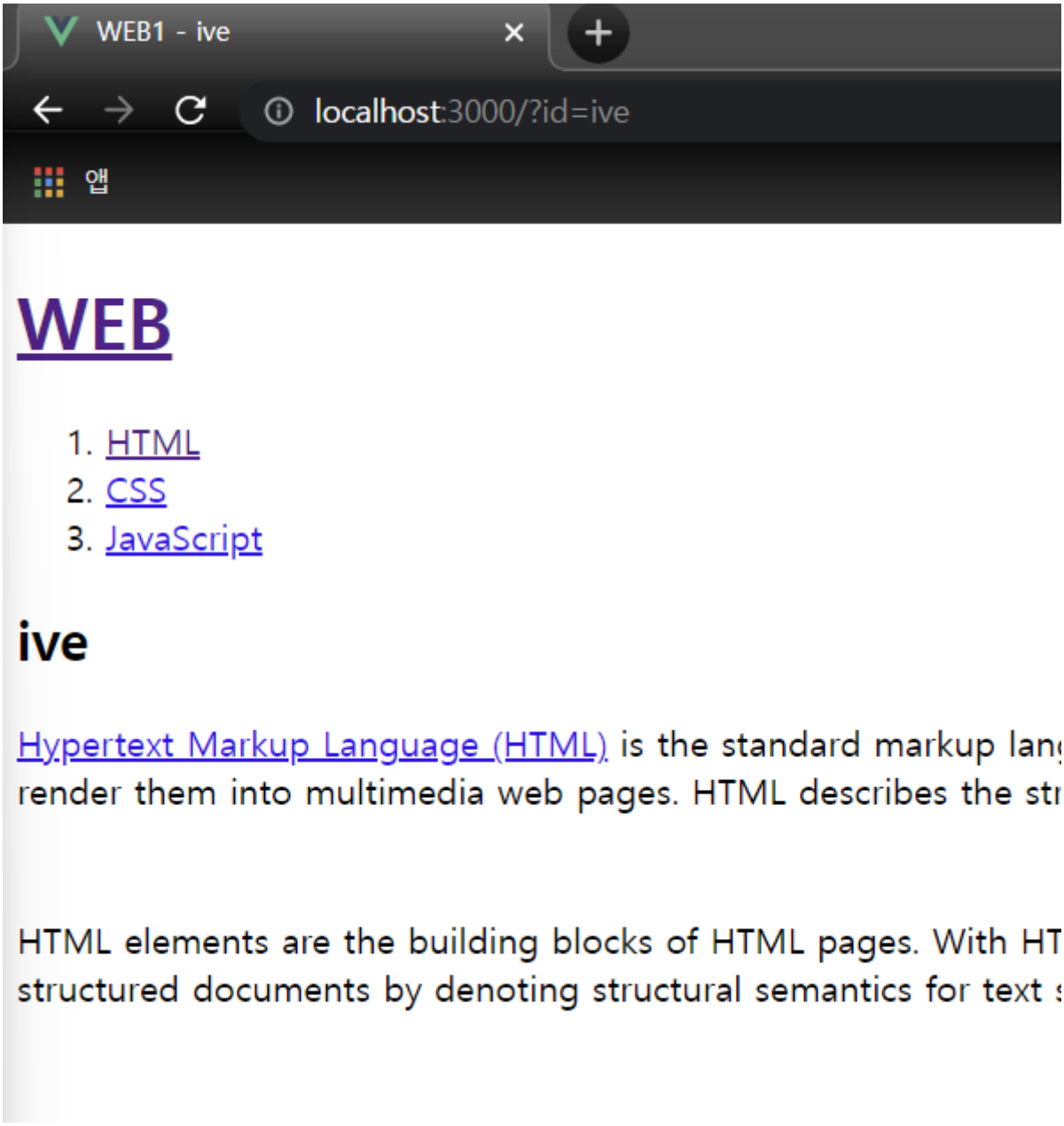
1. HTML
2. CSS
3. JavaScript

HTML

Hypertext Markup Language (HTML) is the standard markup language for **creating we**
render them into multimedia web pages. HTML describes the structure of a web page

HTML elements are the building blocks of HTML pages. With HTML constructs, image
structured documents by denoting structural semantics for text such as headings, para

- 잘 실행된다



- 주소를 바꾸면 제목도 동적으로 변경된다
- 콘솔 창

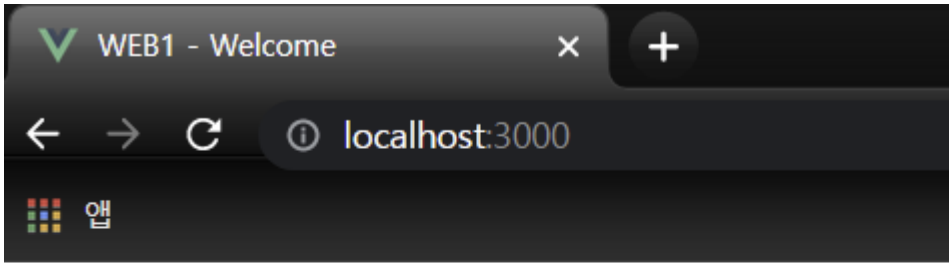
```
$ node main.js
HTML
undefined
undefined
ive
undefined
undefined
```

- queryData.id 눈에 잘 안 들어오니까 그냥 `title`로 선언

```
let title = queryData.id;
...
<title>WEB1 - ${title}</title>
...
<h2>${title}</h2>
```

- url 변경

```
if (_url == "/") {
    title = "Welcome";
  }
...
<h1><a href="/">WEB</a></h1>
```



WEB

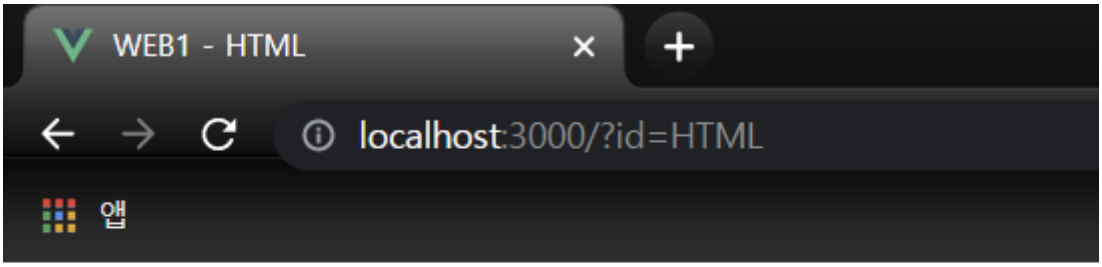
1. HTML
2. CSS
3. JavaScript

Welcome

Hypertext Markup Language (HTML) is the standarc
render them into multimedia web pages. HTML des

HTML elements are the building blocks of HTML pa
structured documents by denoting structural semar

- 첫 메인으로 접속 시 `Welcome`으로 잘 바뀌어 있다
  - `/`일 경우에만 Welcome인 거니까 나머지는 클릭할 때마다 그 제목별로 잘 바뀜



WEB

1. HTML
2. CSS
3. JavaScript

HTML

Hypertext Markup Language (HTML) is the standard marku
render them into multimedia web pages. HTML describes t

동작

- CRUD 중점적으로 알기

파일 읽기

- 검색: `nodejs file read`

fs.readFile(path[, options], callback)

▶ History

- `path` `<string>` | `<Buffer>` | `<URL>` | `<integer>` filename or file descriptor
- `options` `<Object>` | `<string>`
  - `encoding` `<string>` | `<null>` **Default:** `null`
  - `flag` `<string>` **Default:** `'r'`
- `callback` `<Function>`
  - `err` `<Error>`
  - `data` `<string>` | `<Buffer>`

Asynchronously reads the entire contents of a file. Example:

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

The callback is passed two arguments `(err, data)`, where `data` is the contents of the file.

If no encoding is specified, then the raw buffer is returned.

If `options` is a string, then it specifies the encoding. Example:

```
fs.readFile('/etc/passwd', 'utf8', callback);
```

Here is an example of the asynchronous version:

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

fs.unlink('/tmp/hello', (err) => {
  if (err) throw err;
  console.log('successfully deleted /tmp/hello');
});
```

- 코드 작성

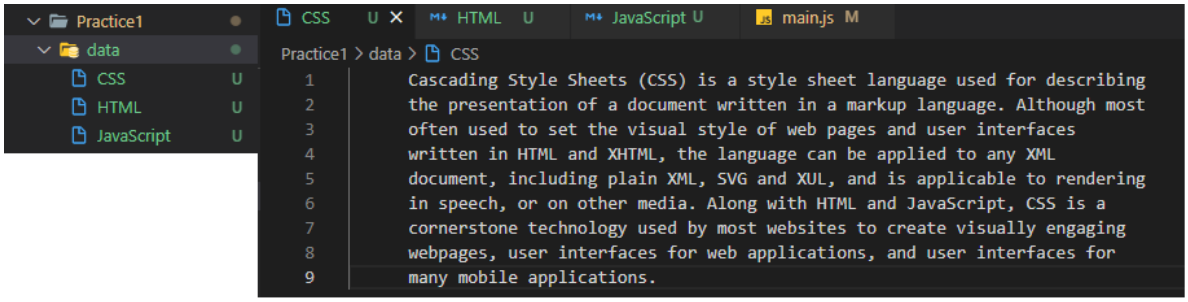
```
const fs = require("fs");

fs.readFile("sample.txt", "utf8", (err, data) => {
  if (err) throw err;
  console.log(data);
});
```

```
kiwio@SSAFY MINGW64 /e/Studying/Node.js/WEB 2 - Node.js/nodejs (main)
$ node fileread.js
그룹 아이브(IVE)가 세 번째 싱글 '애프터 라이크(After LIKE)' 발매 이후 10주 연속 빌보드
차트인에 성공했다.
미국 음악 전문 매체 빌보드가 1일(이하 현지 시각) 발표한 최신 차트(11월 5일 자)에 따르면
 아이브(안유진·가을·레이·장원영·리즈·이서)의 세 번째 싱글 타이틀곡 '애프터 라이크'
는 '빌보드 글로벌 200(Billboard Global 200 / 최고 순위 20위)' 차트 116위, '빌보드 글로
벌(미국 제외)(Billboard Global Excl. U.S. / 최고 순위 9위)' 차트 78위에 자리했다.
이로써 아이브는 지난 8월 22일 세 번째 싱글 '애프터 라이크' 발매 이후 10주째 '빌보드 글
로벌 200'과 '빌보드 글로벌(미국 제외)' 차트인을 이어가고 있다.
또 아이브는 10월 26일 자 'Billboard Japan Hot 100(빌보드 재팬 핫 100)' 차트에 첫 번째
싱글 타이틀곡 '일레븐(ELEVEN)'을 9위로 재진입시키는 저력을 과시했고, '애프터 라이크'도
43위로 안착해 9주째 차트인을 기록했다.
```

- 잘 찍힌당

본문 읽기

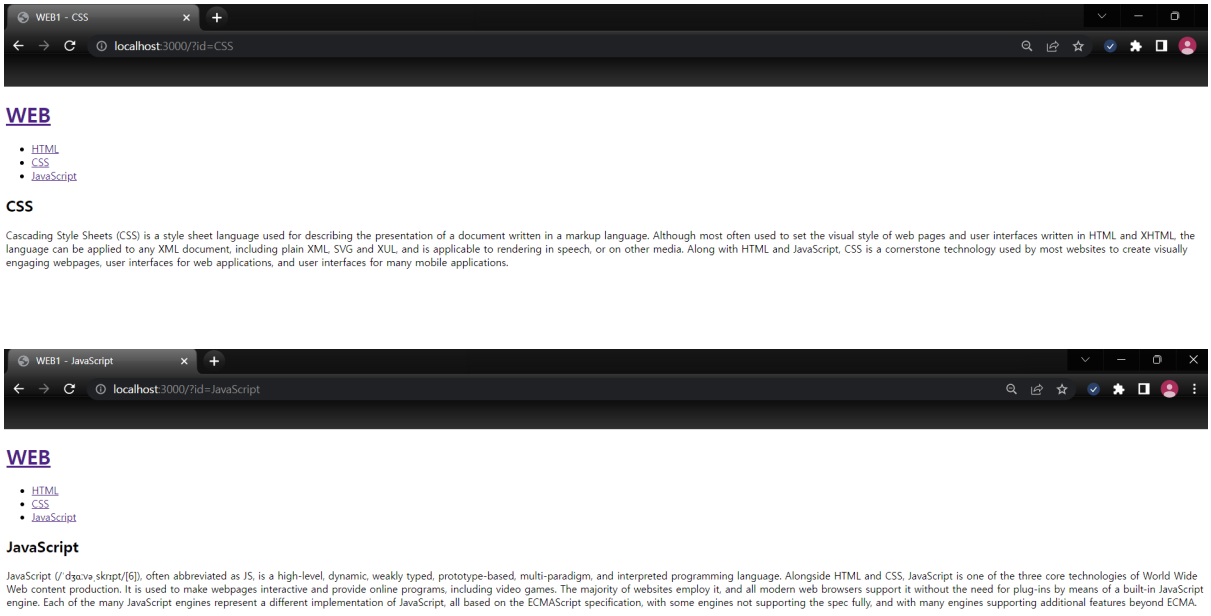
- data 파일 생성하여 본문만 가져오기




```
const http = require("http");
let fs = require("fs");
let url = require("url");

const app = http.createServer(function (request, response) {
  let _url = request.url;
  let queryData = url.parse(_url, true).query;
  let title = queryData.id;
  if (_url == "/") {
    title = "Welcome";
  }
  if (_url == "/favicon.ico") {
    return response.writeHead(404);
  }
  response.writeHead(200);
  fs.readFile(`data/${queryData.id}`, "utf8", function (err, description) {
    let template = `
  <!DOCTYPE html>
<html>
  <head>
    <title>WEB1 - ${title}</title>
    <meta charset="utf-8" />
```

```
  </head>
  <body>
    <h1><a href="/">WEB</a></h1>
    <ul>
      <li><a href="/?id=HTML">HTML</a></li>
      <li><a href="/?id=CSS">CSS</a></li>
      <li><a href="/?id=JavaScript">JavaScript</a></li>
    </ul>
    <h2>${title}</h2>
    <p>
      ${description}
    </p>
  </body>
</html>
  `;
    response.end(template);
  });
});

app.listen(3000);
```

- `fs.readFile` 코드 생성하여 그 안에 본문 집어넣기
  - 여기서 본문은 `${description}`으로 표현



- 본문 내용 잘 바뀐다 😉

입력값