K-Digital Training 웹 풀스택 과정

Node.js 활용

Express 모듈

Express

codingOn

- **웹 서버를 생성**하는 것과 관련된 기능을 담당하는 프레임워크
- 웹 애플리케이션을 만들기 위한 각종 메소드와 미들웨어 등이 내장되어 있다.

- http 모듈 이용 시 코드의 가독성 ↓ 확장성 ↓

  → 이를 해결하기 위해 만들어진 것이 **Express 프레임워크**

Express 설치

```
> npm install express
```

- **npm_modules** 가 만들어지며 express에 관련된 폴더가 생성
- **package.json의 dependencies 에 express** 기록

```
> node_modules
```

```
"dependencies": {
  "express": "^4.18.1"
}
```

Express 사용

```
const express = require('express');
const app = express();
const PORT = 8000;

app.get('/', function (req, res) {
  res.send('hello express');
});

app.listen(PORT, function () {
  console.log(`Listening on port ${PORT}! http://localhost:${PORT}`);
});
```

app.js

Express 사용

- **express()**
  - Express 모듈이 export 하는 최상위 함수로, **express application을 만듦**
- **app 객체**
  - Express() 함수를 호출함으로써 만들어진 **express application**

```
1   const express = require('express');
2   const app = express();
```

Express 사용

codingON

- **응답 메서드**
- res.send() : 다양한 유형의 응답(문자열, 버퍼, 객체 등)을 클라이언트에게 전송하는 데 사용되며 Express가 적절한 Content-Type 헤더를 자동으로 설정
- res.render() : 템플릿 엔진을 사용하여 뷰 파일을 렌더링하고 클라이언트에게 HTML을 전송하는 데 사용
- res.json() : JSON 응답을 클라이언트에게 전송하는 데 사용. Content-Type 헤더를 application/json으로 설정

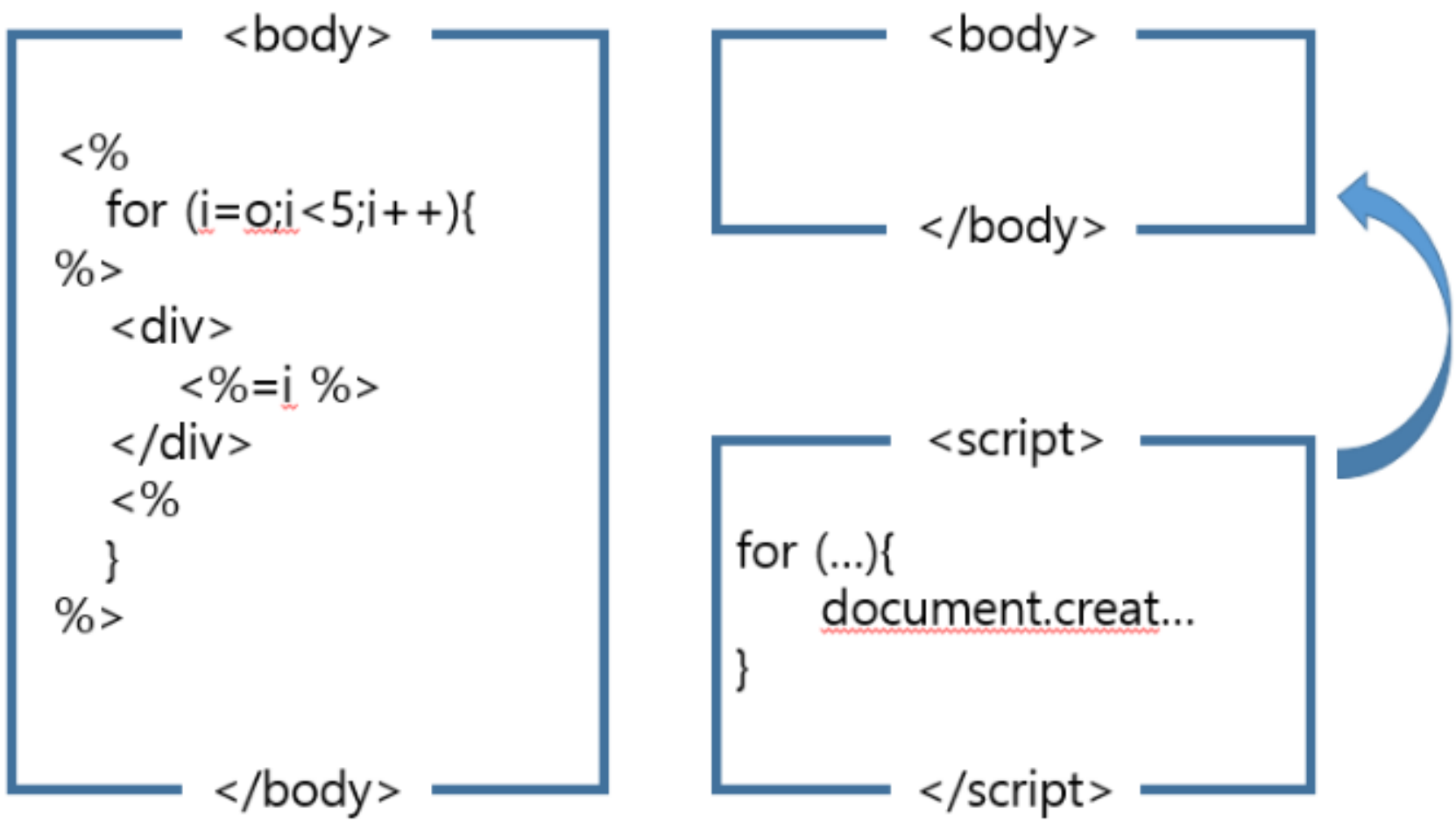
codingOn

템플릿 엔진

EJS 템플릿

- 템플릿 엔진
  - 문법과 설정에 따라 **파일을 html 형식으로 변환**시키는 모듈
- ejs
  - **Embedded Javascript** 의 약자로, **자바스크립트가 내장되어 있는 html 파일**
  - 확장자는 .ejs

ejs 템플릿


<body>
<%
for (i=o;i<5;i++){
%>

<%=i %>

<%
}
%>
</body>
<body>
</body>
<script>
for (...){
document.creat...
}
</script>

ejs 템플릿

```
$ npm install ejs
```

```
app.set('view engine', 'ejs');
app.set('views', './views');
```

ejs 템플릿

```
const express = require("express");
const app = express();
const PORT = 8000;

app.set("view engine", "ejs");
app.set("views", "./views");

app.get("/", (req, res) => {
    res.send("Hello Express");
});

app.get("/test", (req, res) => {
    res.render("test");
});

app.listen(PORT, () => {
    console.log(`http://localhost:${PORT}`);
});
```

ejs 템플릿 설정

ejs 템플릿 렌더링

ejs 템플릿

```
<html>
    <head>
        <title>EJS TEST</title>
    </head>
    <body>
        <% for (var i = 0; i < 5; i++) { %>
            <h1>안녕</h1>
        <% } %>
    </body>
</html>
```

ejs 문법 사용하기

```
<% %>
```

- 무조건 자바스크립트 코드가 들어가야 하고, 줄바꿈을 할 경우에는 새로운 〈% %〉 를 이용해야 한다.

```
<%= %>
```

- 값을 템플릿에 출력할 때 사용

```
<%- include('view의 상대주소') %>
```

- 다른 view 파일을 불러올 때 사용

미들웨어

- 요청과 응답사이에서 **실행되는 함수로**
- **서버와 클라이언트를 이어주는** 중간 작업
- use() 를 이용해 **등록**할 수 있다.
- next()는 다음 미들웨어 함수를 호출하기 위해 사용

```
app.use((req, res, next) => {
    console.log("Time:", Date.now());
    next();
});
```

미들웨어 - static

codingON

- 이미지, **css** 파일 및 **Javascript** 파일(front)과 같은 **정적 파일 제공**
- Express 에 있는 static 메소드를 이용해 미들웨어로 로드
- 등록 방법

```
//정적 파일 불러오기
//방법1
app.use("/public", express.static(__dirname + "/public"));

//방법2
const path = require('path');
app.use("/pulbic", express.static(path.join(__dirname, 'public')));
```

앞에 “/public”은 url경로, 뒤에는 폴더 경로

ejs 템플릿

```
const express = require("express");
const app = express();
const PORT = 8000;

app.set("view engine", "ejs");
app.set("views", "./views");
app.use("/public", express.static(__dirname + "/public"));

app.get("/", (req, res) => {
    res.send("Hello Express");
});

app.get("/test", (req, res) => {
    res.render("test");
});

app.listen(PORT, () => {
    console.log(`http://localhost:${PORT}`);
});
```

정적 파일 로드 코드

[keyword] 미들웨어, static