[스파르타코딩클럽] Node.js 기초반 - 2주차

매 주차 강의자료 시작에 PDF파일을 올려두었어요!

▼ PDF 파일

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/7ec9fe82-5066-4a28-8ad7-e68b290c9fb9/_Nodejs_2.pdf

[수업 목표]

1. Express의 Routing이 무엇인지 알게 된다!
2. Express의 Middleware가 무엇인지 알게 된다!
3. 템플릿 엔진이 무엇인지 알게되고 실제 페이지를 만들어봅니다!

[목차]

모든 토글을 열고 닫는 단축키
Windows : `Ctrl` + `alt` + `t`
Mac : `⌘` + `⌥` + `t`

---

01. 2주차 오늘 배울 것

▼ 1) Express 에 대한 기본적인 지식을 배웁니다.

라우팅에 대해서 배웁니다. 이를 이용해 우리는 각각의 url에 따라 다른 결과물을 보여줄 수 있습니다.

미들웨어에 대해서 배웁니다. 이전에 배운 웹 프레임워크의 특징. 반복해서 자주 해야되는 동작들을 아주 간단하게 미리 처리 할 수 있습니다.

▼ 2) EJS를 사용해서 View를 생성합니다.

👉 html 로 응답을 보내주면 상대방의 웹 브라우저에 이쁜 결과를 보여줄 수 있습니다. 하지만 사용자마다 혹은 로그인/로그아웃 상태에따라 조금씩 다른 화면을 보여줘야 한다면 템플릿 엔진을 사용해서 그런 문제를 해결할 수 있습니다

02. Express - Routing

▼ 3) 라우팅

👉 이전 수업에서 우리가 웹브라우저에 google.com 이라고 입력하면 해당 구글 웹서버에서 응답을 받는 것이라고 배웠습니다.

```
https://spartacodingclub.kr/online/webplus
https://spartacodingclub.kr/portfolio
```

👉 위와 같은 두 개의 주소가 있습니다. 두 개는 같은 도메인 주소인데 서로 보여주는 내용이 다릅니다. url 주소의 뒷부분인 `path` 정보에 따라 다른 결과가 보여진 것입니다.

실제 express 에서 router는 어떤 것을 가지고 경로를 구분할까요? 예시를 한번 보면서 이야기하겠습니다.

```
app.get('/', (req, res) => {
  res.send('Hello World!')
})
```

👉 이것은 path 가 `'/'` 일 때, 이건 뒤에 아무것도 안붙었을때를 의미하는데요, 그냥 도메인 주소만 입력하고 들어왔을 때가 됩니다. 그런 경우에 `Hello World!` 를 응답한다는 내용입니다.

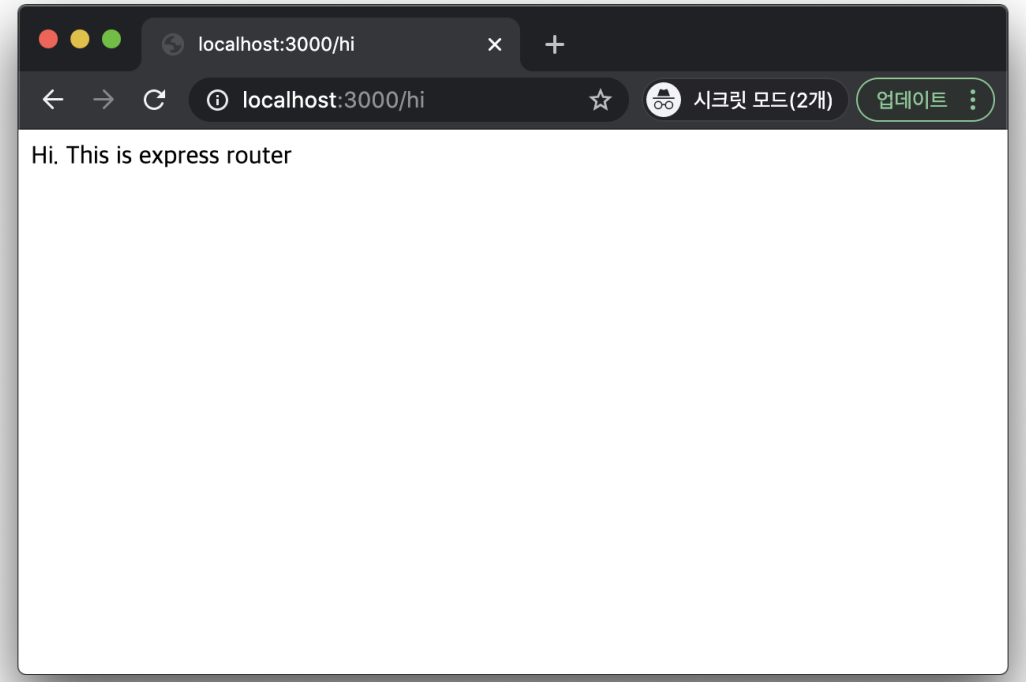
그럼 위의 예시처럼 다른 경로일 때 다른 내용을 보여줘 볼까요?

▼ [코드스니펫] index.js(1)

```
const express = require('express')
const app = express()
const port = 3000

app.get('/hi', (req, res) => {
    res.send('Hi. This is express router')
  })

app.get('/', (req, res) => {
  res.send('Hello World!')
})

app.listen(port, () => {
  console.log(`listening at http://localhost:${port}`)
})
```

```
app.get('/hi', (req, res) => {
  res.send('Hi. This is express router')
})
```

👉 위 코드를 실행하고 node index.js 를 실행해 봅시다. 그리고 브라우저로 http://localhost:3000/hi 에 접속합시다. 그럼 어떤 화면이 나오나요?



👉 `Hello World!` 가 아닌 다른 내용이 나오죠? 앞으로 우리는 이것을 이용해서 상품 목록을 보여주는 화면, 상품 상세 내용을 보여주는 화면, 장바구니에 담은 상품들을 보여주는 화면 등등을 만들 예정입니다!

▼ 4) http method

- router가 경로를 구분하는 요소로 url 주소 외에 `http method` 라는 부분이 있습니다. 우리가 해당 주소를 요청하는 방식에 대한 것인데요. 같은 주소라도 어떤 방식으로 요청하는지에 따라 응답을 다르게 할 수 있습니다. (참고)
- 우리 수업에서는 `GET`, `POST` , `DELETE` 정도만 쓰게 될 것입니다. 이중에 `GET` 은 우리가 그동안 몰랐지만 일반 브라우저에서 요청하는 행위가 다 `GET` 요청으로 되어있습니다. 우리는 알게 모르게 브라우저를 쓰면서 요청을 다 `GET`으로 보내고 있었던 것이지요.

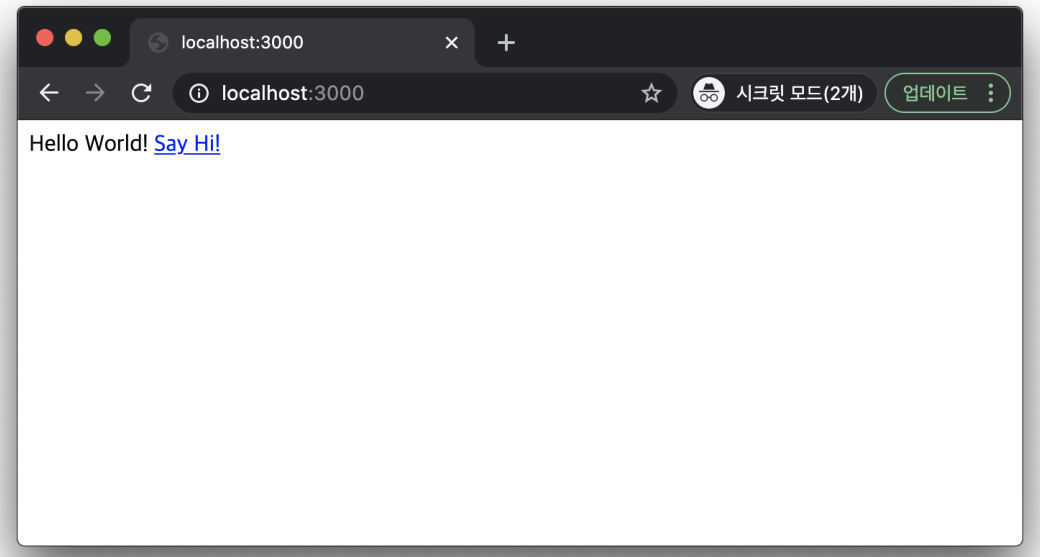
▼ 5) 페이지 이동을 새롭게 추가해봅시다!

👉 자 그럼 실제로 어떻게 하면 되는지 실습으로 진행해봅시다. 아까 위에서 `/hi` 로 진입할 수 있는 route를 만들었는데요. 이제 우리는 두 개의 페이지를 가지고 있다가 생각하시면 됩니다. 각각의 페이지는 루트 `/` 혹은 `/hi` 로 되어있고요. 그럼 루트페이지에서 hi페이지로 이동하려면 어떻게 해야할까요? html에서 배웠던 `a 태그` 를 이용해서 화면 이동하는것을 한번 해보도록 하겠습니다.

▼ [코드스니펫] index.js(2)

```
app.get('/', (req, res) => {
  res.send('Hello World! <a href="/hi">Say Hi!</a>')
})
```

👉 코드를 수정하고나서 기존에 실행중이던 `node index.js` 가 있다면 반드시 종료하고 (`ctrl + c`) 다시 노드를 실행하도록 합니다.



👉 그럼 이제 저 부분을 클릭을 하면 url이 변경이되고 우리가 만들어놓은 페이지로 이동을 할 수 있습니다!

03. Express - Router 객체 사용하기

▼ 6) router 사용하기

👉 현재 우리는 `/`, `/hi` 두가지 경로를 가지고 있습니다. 나중에 우리가 쇼핑몰을 만든다면 페이지가 어느정도 필요할까요? 미리 한번 페이지를 보면서 계산해 봅시다. http://3.36.86.60:8888/

👉 1) 첫 화면은 로그인 화면입니다. → 2) 그리고 회원가입화면이 있겠죠. → 3) 로그인을 하고나면 상품 목록 페이지가 있습니다. → 4) 상품을 하나 클릭하면 상품 상세페이지가 있구요. → 5) 장바구니에 담고나면 장바구니 페이지도 있습니다. 구매하기를 눌러볼까요? → 6) 결제를 하기 위한 배송정보를 입력하는 페이지가 있습니다.

지금 이 서비스만 해도 페이지가 적어도 6개는 필요합니다. 실제로 서비스를 만들다보면 이런 페이지의 숫자는 엄청나게 늘어납니다. 그럼 이렇게 페이지가 하나씩 늘어날 때마다 index.js 파일에 `app.~~` 가 계속 추가가 되는데요. 이러다보면 파일 하나에 너무 많은 양의 코드가 있고 점점 관리하기가 힘들어지는 문제가 생깁니다. Express 에서는 이를 방지하기 위해 Router 객체를 제공해서 비슷한 route끼리 묶어서 파일로 분리 시킬 수 있게 해줍니다.

▼ [코드스니펫] index.js

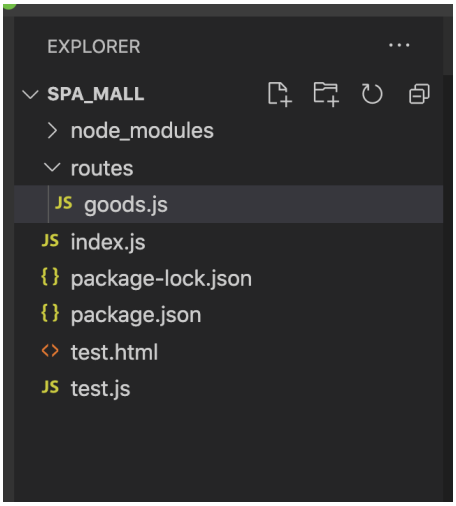
```
app.get('/goods/list', (req, res) => {
  res.send('상품 목록 페이지')
})

app.get('/goods/detail', (req, res) => {
  res.send('상품 상세 페이지')
})

app.get('/user/login', (req, res) => {
  res.send('로그인 페이지')
})

app.get('/user/register', (req, res) => {
```

```
  res.send('회원가입 페이지')
})
```

👉 자 위와 같이 route 가 4개가 있는 경우를 보겠습니다. 먼저 상품 목록과 상품 상세 url 을 볼게요. 두개는 다 상품에 관련된 페이지이다 보니 url 에 goods 라는 공통된 부분을 가지고 있습니다. 이런 비슷한 종류의 페이지끼리는 같은 Router 로 묶어주면 편한데요. 한번 해보겠습니다. 먼저 현재 프로젝트에서 새로운 파일을 추가해보겠습니다

▼ **[코드스니펫] goods.js**

```
var express = require('express');
var router = express.Router();

router.get('/list', function(req, res, next) {
  res.send('Router 상품 목록 페이지')
});

router.get('/detail', function(req, res, next) {
  res.send('Router 상품 상세 페이지')
});

module.exports = router;
```



👉 위와 같이 routes 폴더를 만들고 그 안에 goods.js 파일을 생성했습니다. 그리고 기존 index.js 파일을 수정합니다.

```
// 기존
// app.get('/goods/list'...
// app.get('/goods/detail'...
// 두가지 route 는 삭제합니다.

const goodsRouter = require('./routes/goods');

app.use('/goods',goodsRouter);
```

👉 자 여기까지 하고 이제 서버를 재시작 해봅니다. 그리고 `/goods/list` 경로로 접속해봅시다. 그럼 변경된 내용으로 새롭게 보이는 것을 확인할 수 있습니다. 이런 방식으로 Router는 비슷한 route 들의 집합으로 모아 모듈식으로 처리를 할 수 있습니다. 관리하기도 쉬워집니다. 그래서 Router 객체를 express 에서 "mini-app"이라고 부르기도 한답니다 😎

▼ 7) user router 분리

👉 위에서 Router객체를 이용해서 goods 관련된 route들을 별도의 파일로 분리했는데요. 그럼 아래 있었던 유저의 로그인/회원가입부분도 한번 별도의 Router 객체로 분리해 볼까요? `routes/user.js` 라는 파일을 만들고 한번 분리를 해보도록 합시다

▼ routes/user.js

```
var express = require('express');
var router = express.Router();

router.get('/login', function(req, res, next) {
  res.send('로그인 페이지')
});

router.get('/register', function(req, res, next) {
  res.send('회원가입 페이지')
});

module.exports = router;
```

▼ index.js 파일 수정

```
const userRouter = require('./routes/user');

app.use('/user',userRouter);
```

04. Express - Middleware

▼ 8) 미들웨어란?

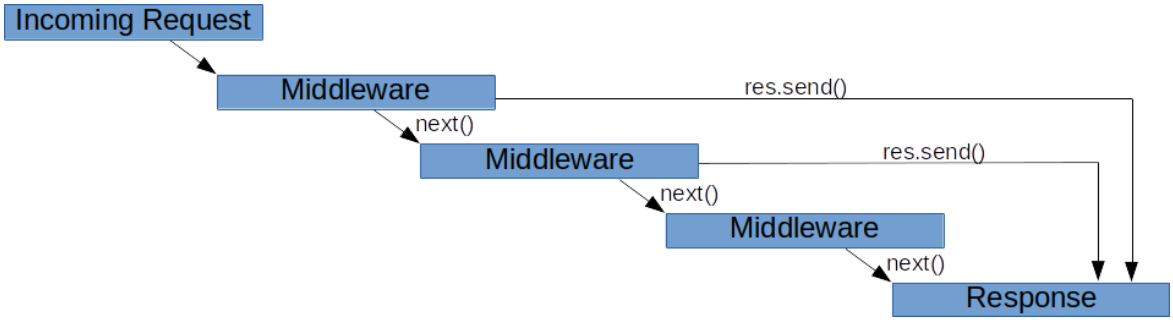
👉 이전 시간에 router 를 사용하면서 잠깐 미들웨어란 말을 들어보았을 겁니다. 이 미들웨어란 무엇일까요? 먼저 예를 하나 들어보겠습니다. 우리가 매 route 를 생성할 때마다 만약에 공통된 처리를 하고 싶은 경우를 생각해볼게요. 요청이 들어올 때마다 이 요청이 어떤 유저의 요청인지 인증검사를 해야할 수 있어요. 그런 경우에 코드를 작성하게 되면 route 를 하나 만들 때마다 매번 유저 인증을 확인하는 코드를 넣어야 할 것입니다. 이런 반복되는 작업들, 혹은 우리가 정의한 route에 오기 전에 중간에서 미리 처리해야할 것들을 정의해둔 것을 미들웨어라고 합니다.

▼ **[코드스니펫] 미들웨어**

```
app.use((req, res, next) => {
  console.log(req);
  next();
});

app.get('/', (req, res, next) => {
  res.send('Welcome Home');
});
```

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

app.use((req, res, next) => {
  console.log(req);
  next();
});

app.get('/', (req, res, next) => {
  res.send('Welcome Home');
});

app.listen(3000);
```

- 이런 예시코드를 한번 보겠습니다. 중간에 `app.use` 라고 하는 것이 미들웨어를 쓰겠다고 하는 부분입니다. 저 코드를 사용하면 어떤 요청이 들어오던간에 `req` 를 로그로 찍고 나서 실제 route 로 전달되게 됩니다.



- 이런식으로 들어온 요청은 우리가 설정한 미들웨어를 통과하고 중간에 응답을 해서 종료가 되거나 아니면 다음 미들웨어가 있는 경우에는 다음 미들웨어로 넘어갑니다.

▼ 9) 데이터 가공 용도 `express.json()` `express.urlencoded()`

> 👉 나중에 배우겠지만 우리는 웹서버에 요청할 때 단순히 url 외에 추가적인 정보를 전달 할 수 있습니다. `POST` 메서드의 `body` 정보가 그것인데요. 이것을 express 에서 사용하려면 복잡한 절차가 필요합니다. 이 데이터를 바로 사용하기 쉽게 가공해주는 미들웨어가 바로 이 `express.json()` 입니다.

▼ [코드스니펫] index.js 파일 수정

```
app.use(express.urlencoded({extended: false}))
app.use(express.json())
```

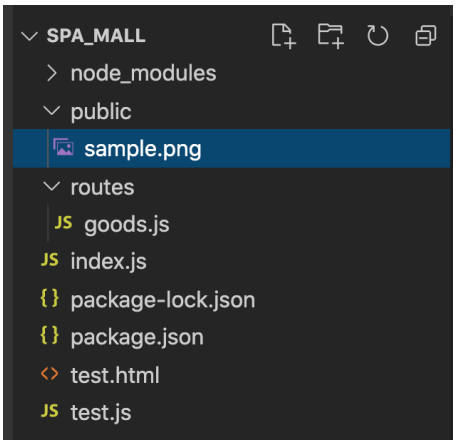
- 이렇게 코드를 추가하면 미들웨어를 쓸 준비는 다 완성된 것입니다. 실제 사용되는 것은 이후 4주차에 보게 될 거예요! (그때까지 꾸준히 배워봐요! 🐓)

▼ 10) Express의 기본 제공 미들웨어 - static

> 👉 Express 애플리케이션의 정적 자산을 제공하는 역할을 하는 static 입니다. 이것을 이용하면 우리의 express 에서도 이제 이쁜 이미지, 신나는 동영상 같은 정적 파일을 제공할 수 있습니다. 사용법은 간단합니다.

```
app.use(express.static('public'));
```

> 👉 index.js 파일 중간에 위 코드 한줄만 추가해주세요. 그리고 이제 프로젝트에서 public 폴더를 생성하고 이미지 하나를 해당 폴더에 넣어줍니다.



👉 저곳에 가지고 있는 이쁜 이미지를 하나 추가하고 서버를 재시작하고 `/sample.png` 경로를 입력해보면 내가 올린 이미지를 웹브라우저에서 보실 수 있습니다.

👉 **[혹시나 헷갈리실 분들을 위해]**
Q: app.use(express.static('public')); 같은 코드를 중간쯤에 넣으라고 했는데 아무데나 막 넣어도 되나요?
A: 네! 보통 파일 최상단의 const ~ 하는 부분들을 제외하고 그 아래 부분들 (대부분 app..이 모여있는곳들)에서 어디에 두셔도 큰 문제가 당장은 되지 않습니다. 현재는 서로 route가 겹치는 경우가 없기 때문인데요. 정확하게는 위에서부터 순차적으로 체크하면서 조건이 맞는 경우로 가기 때문에 실수로 동일한 route 를 두개 만들었다면 그중 위에것을 보게 됩니다.

05. 템플릿 엔진(Template Engine)이란?

▼ 11) 템플릿 엔진을 사용하기 전

👉 이제 실제 홈페이지처럼 화면을 구성할 것인데요. 여태까지 저희가 만든 페이지는 너무 썰~렁했죠. 흰 화면에 글자만 있구요. 일반적인 홈페이지는 어떻게 구성이 되어 있나요? 알록달록하고 이미지도 있고 예쁘게 레이아웃도 잡혀있죠. 그렇다면 우리도 그렇게 할것인데.. 필요한 것이 바로 HTML과 CSS입니다. 먼저 HTML 구성을 하려고 하는데요, 앞선 예제를 따라서 해볼까요?

▼ **[코드스니펫] index.js**

```
app.get('/', (req, res) => {
  res.send('<!DOCTYPE html>\
  <html lang="en">\
  <head>\
      <meta charset="UTF-8">\
      <meta http-equiv="X-UA-Compatible" content="IE=edge">\
      <meta name="viewport" content="width=device-width, initial-scale=1.0">\
      <title>Document</title>\
  </head>\
  <body>\
      Hi. I am with html<br>\
      <a href="/hi">Say Hi!</a>\
  </body>\
  </html>')
})

# 여기서 중요한점은 send() 안에 전체 내용이 ''작은 따옴표로 묶여있다는점
# 마지막줄을 제외한 매 줄마다 마지막에 \ 문자가 들어가 있는점
# 내부에서 쓰이는 문자들은 다 ""쌍 따옴표를 쓰고 있다는점입니다.
```

👉 이렇게 하고 다시 서버를 띄워볼까요? 아직 눈에 띄게 바뀐부분은 없지만 괜찮습니다. 앞으로 차차 추가하게 될 것입니다.
우리가 지금 한 페이지를 이렇게 추가했는데요. 앞으로 그럼 매 페이지를 생성할 때마다 이런 식으로 추가를 해야할까요? HTML 코드를 추가해야하는데 JavaScript 언어 위에서 작성하려니 불편한점이 한두가지가 아닙니다. 그럼 어떻게 하면 조금 더 편하게 작업 할 수 있을까요?
바로 템플릿 엔진을 사용하면 됩니다. 템플릿 엔진은 사용하면 일관된 양식의 HTML 파일에다가 그때그때 원하는 데이터를 동적으로 삽입해서 우리가 원하는 형태의 홈페이지를 구성할 수 있게 됩니다.

▼ 12) 템플릿 엔진의 장점

1. 많은 코드를 줄일 수 있다.
   → 대부분의 Template Engine은 기존의 HTML에 비해서 간단한 문법을 사용한다.
2. 재사용성이 높다.
   → 웹페이지 혹은 웹앱을 만들 때 똑같은 디자인의 페이지에 보이는 데이터만 바뀌는 경우가 굉장히 많다.
3. 유지보수에 용이하다.
   → 하나의 Template을 만들어 여러 페이지를 렌더링하는 작업에는 또 다른 이점이 있다.

06. 템플릿 엔진 설치하기

▼ 13) 템플릿 엔진 - ejs 설치

👉 위와 같은 이유로 템플릿 엔진이란 것을 이제 사용할 것인데요. express 에서 사용가능한 템플릿 엔진은 종류가 많이 있는데요, 우리는 그중에 `EJS`란 것을 사용해보도록 하겠습니다!

```
$ npm install ejs
```

👉 **[혹시나 헷갈리시는 분들을 위해]**
커맨드 실행시에 앞에 저 $ 부분은 복사하지 않습니다. `npm install ejs` 부분만 복사하셔서 터미널에 붙여넣기 하고 엔터 치시면 되어요~! 😉

👉 `package.json` 파일을 열었을 때 `dependencies` 부분에 `ejs` 가 들어가 있으면 정상적으로 설치가 된 것입니다. 설치하셨을 당시에 따라 뒤의 버전(`"^3.1.6"` 같은 숫자 부분)은 다를 수 있으니 이부분은 달라도 크게 신경쓰지 않으셔도 괜찮습니다 🙂

▼ 14) ejs 파일 추가하기

1. 설치를 하고 index.js 파일을 수정합니다

   ▼ **[코드스니펫] index.js**

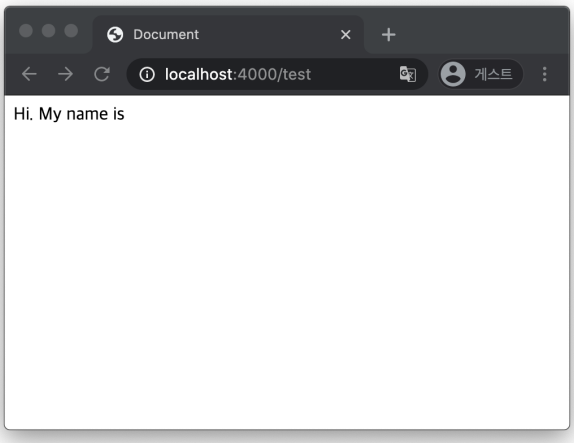
```
app.set('views', __dirname + '/views');
app.set('view engine', 'ejs');

app.get('/test', (req, res) => {
  let name = req.query.name;
  res.render('test', {name});
})
```

2. 그리고 `views` 폴더를 생성하고 `test.ejs` 파일을 추가해줍니다

   ▼ **[코드스니펫] test.ejs**

```
<!DOCTYPE html>
<html lang="en">
  <head>
      <meta charset="UTF-8">
      <meta http-equiv="X-UA-Compatible" content="IE=edge">
      <meta name="viewport" content="width=device-width, initial-scale=1.0">
      <title>Document</title>
  </head>
  <body>
      Hi. My name is <%= name %><br>
  </body>
</html>
```

3. 자 그럼 새로 생성한 페이지로 한번 접근해 볼까요?



4. 잘 뜨긴 하는데요 우리가 만든 파일과 조금 다른 거 같습니다. 우리가 만든 ejs 파일을 먼저 한번 볼까요?

```
Hi. My name is <%= name %><br>
```

- 중간에 이 부분이 있는데요 저 뒷부분에 `<%= name %>` 이게 있지만 실제 화면에서는 아무것도 보이지 않습니다. 이부분이 ejs 템플릿 문법으로 만약에 name 이라는 변수가 넘어온 것이 있으면 그것을 출력하는 것인데 지금 우리가 넘겨준 것이 없으므로 아무것도 출력하지 않습니다.

5. 그럼 그 변수를 넘겨준다는 표현은 무엇일까요? 앞서 index.js 파일 부분을 다시 보겠습니다.

```
app.get('/test', (req, res) => {
  let name = req.query.name;
  res.render('test', {name});
})
```

- 여기서 보면 render를 하는데 `test` 와 `{name}` 두 개의 인자를 넘겨줍니다. 이는 test.ejs 파일을 그리란 뜻이고 그 ejs 파일에 name 값을 객체로 넘겨준다는 의미입니다. 그렇다면 우리는 name 값을 안 넘겨줘서 아무것도 안 뜬 것은 아닙니다. 분명 여기서 name 이라는 변수를 위에서 생성해서 넘겨주고 있습니다.
- 브라우저에서 url 을 변경해 봅시다. http://localhost:3000/test?name=sparta

6. 그러면 이제 화면에 이름이 정상적으로 나오는 것을 확인할 수 있습니다!

👉 [회사에서도 템플릿 엔진 쓰나요?]
회바회(회사 by 회사)입니다. 장고, 노드, 라라벨같은 웹 프레임워크에서 템플릿엔진을 사용해서 홈페이지를 만드는 경우도 있지만 최근 프론트/백엔드로 나뉘어진 개발 문화에서는 프론트 개발은 별도의 프레임워크를 사용하기 때문에 템플릿 엔진을 쓰지 않는 곳도 많이 있습니다. React, Vue.js 등의 프론트엔드 프레임워크를 사용하면 이 자체로 화면을 구성하는 기능을 포함하고 있습니다. 따라서 프론트엔드가 화면을 보여주는 view 부분은 대부분 맡게 되고 백엔드 서버는 데이터를 가공/제공 하는 역할만 하게 됩니다.

07. 템플릿 엔진 사용해서 쇼핑몰 목록페이지 만들기

▼ 14) 쇼핑몰 페이지 맛보기

👉 템플릿 엔진을 사용해서 앞으로 만들 쇼핑몰의 첫페이지를 한번 그려보도록 해보겠습니다

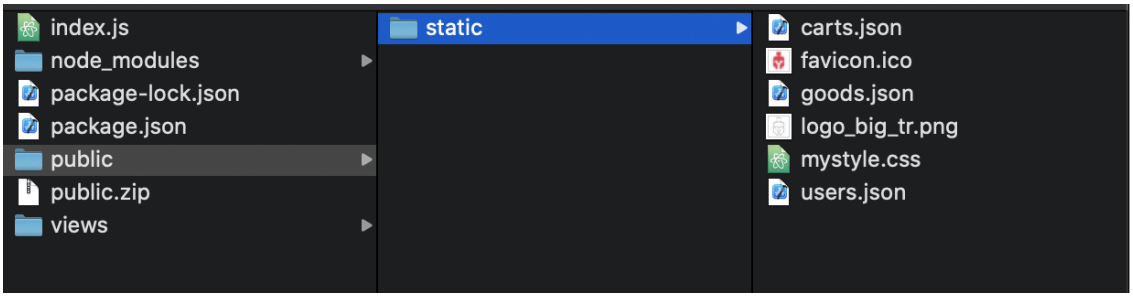
▼ 스태틱 파일 추가

▼ **[코드스니펫] public/static**

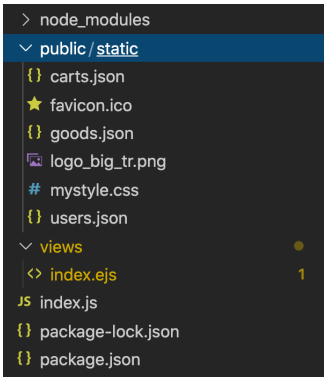
```
https://s3.ap-northeast-2.amazonaws.com/materials.spartacodingclub.kr/node.js/week03/public.zip
```

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/9192902f-b33c-43bf-8128-b4efc948c5ef/public.zip

이 파일들도 프로젝트 폴더에서 압축을 풀어주세요



이런식으로!



vscode에서 이렇게 보이면 성공입니다

▼ **[코드스니펫] views/index.ejs**

```
<!doctype html>
<html lang="en">

<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <link rel="icon" href="/static/favicon.ico" type="image/x-icon">
    <link rel="shortcut icon" href="/static/favicon.ico" type="image/x-icon">
    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css"
        integrity="sha384-TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkgIXeMed4M0jlfIDPvg6uqKI2xXr2" crossorigin="anonymous">

    <!-- Font Awesome CSS -->
    <link href="//maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css" rel="stylesheet">
    <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS -->
    <script src="https://code.jquery.com/jquery-3.5.1.js"
        integrity="sha256-QWo7LDvxbWT2tbbQ97B53yJnYU3WhH/C8ycbRAkjPDc=" crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"
        integrity="sha384-9/reFTGAW83EW2RDu2S0VKaIzap3H66lZH81PoYlFhbGU+6BZp6G7niu735Sk7lN"
        crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/js/bootstrap.min.js"
        integrity="sha384-w1Q4orYjBQndcko6MimVbzY0tgp4pWB4lZ7lr30WKz0vr/aWKhXdBNmNb5D92v7s"
        crossorigin="anonymous"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery-cookie/1.4.1/jquery.cookie.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/2.4.0/socket.io.js"></script>

    <link href="/static/mystyle.css" rel="stylesheet">
```

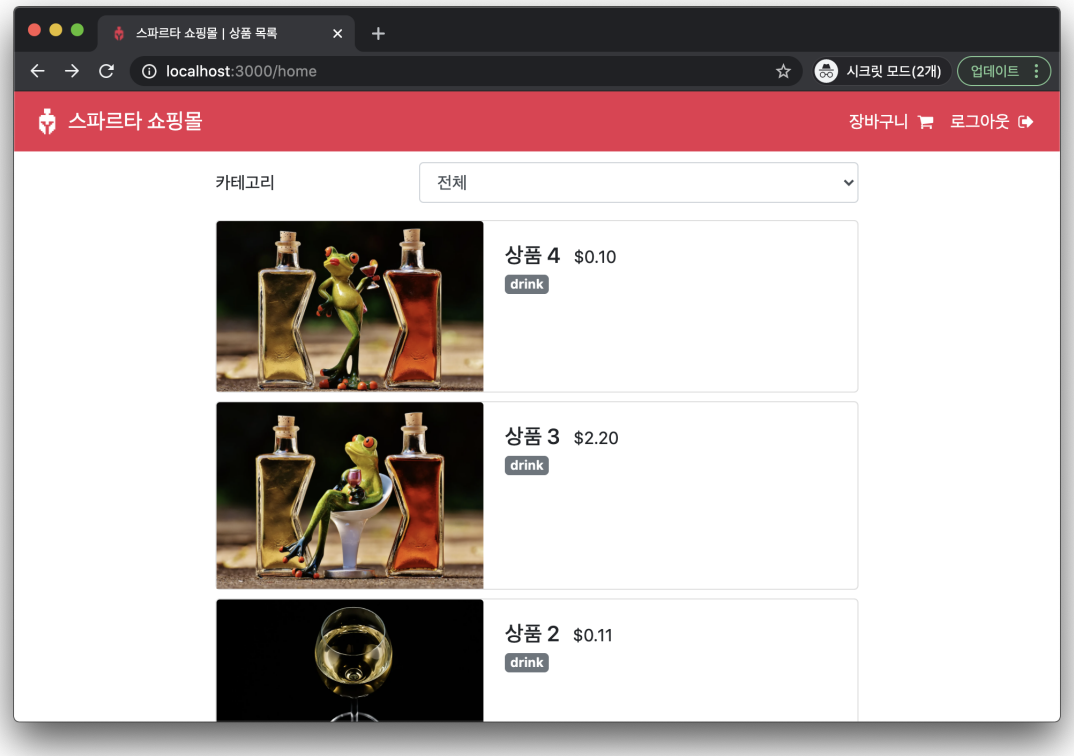
```
    <title>스파르타 쇼핑몰 | 상품 목록</title>
    <script>

        $(document).ready(function () {
            get_goods()
            $("#categorySelect").on("change", function () {
                get_goods($(this).val())
            })
        })

        function sign_out() {
            $.removeCookie('mytoken', { path: '/' });
            $.removeCookie('userName', { path: '/' });
            window.location.href = "/"
        }

        function get_goods(category) {
            $("#goodsList").empty()
            console.log(category)
            let goods = [
                {
                    "_id": "600fa6e49539b288e3c5a2cf",
                    "goodsId": 4,
                    "name": "상품 4",
                    "thumbnailUrl": "https://cdn.pixabay.com/photo/2016/09/07/02/11/frogs-1650657_1280.jpg",
                    "category": "drink",
                    "price": 0.1
                },
                {
                    "_id": "600fa6e49539b288e3c5a2ce",
                    "goodsId": 3,
                    "name": "상품 3",
                    "thumbnailUrl": "https://cdn.pixabay.com/photo/2016/09/07/02/12/frogs-1650658_1280.jpg",
                    "category": "drink",
                    "price": 2.2
                },
                {
                    "_id": "600fa6e49539b288e3c5a2cd",
                    "goodsId": 2,
                    "name": "상품 2",
                    "thumbnailUrl": "https://cdn.pixabay.com/photo/2014/08/26/19/19/wine-428316_1280.jpg",
                    "category": "drink",
                    "price": 0.11
                },
                {
                    "_id": "600fa6e49539b288e3c5a2cc",
                    "goodsId": 1,
                    "name": "상품 1",
                    "thumbnailUrl": "https://cdn.pixabay.com/photo/2016/09/07/19/54/wines-1652455_1280.jpg",
                    "category": "drink",
                    "price": 6.2
                }
            ]
            for (let i = 0; i < goods.length; i++) {
                make_card(goods[i])
            }
        }

        function make_card(item) {
            let htmlTemp = `<div>
                                    <div class="card mb-2" onclick="location.href='/detail?goodsId=${item["goodsId"]}'">
                                        <div class="row no-gutters">
                                            <div class="col-sm-5" style="background: #868e96;">
                                                <img src="${item["thumbnailUrl"]}"
                                                     class="card-img-top h-100" alt="...">
                                            </div>
                                            <div class="col-sm-7 d-flex">
                                                <div class="card-body flex-fill">
                                                    <div class="card-title mb-auto">
                                                        <h5 style="display: inline">${item["name"]}</h5>
                                                        <span class="card-price ml-2">$${number2decimals(item["price"])}</spa
                                                    </div>
                                                    <span class="badge badge-secondary">${item["category"]}</span>
                                                    <!--                                    <p class="card-text"><small class="te
                                                </div>
                                            </div>
                                        </div>
                                    </div>
                                </div>`
            $("#goodsList").append(htmlTemp)
        }

        function makeNoti(data) {
            let htmlTemp = `<div class="alert alert-sparta alert-dismissible show fade" role="alert" id="customerAlert">
                                    ${data["userName"]}님이 방금 <a href="#" class="alert-link">${data["goodsName"]}</a>을 구매했
                                    <button type="button" class="close" data-dismiss="alert" aria-label="Close">
                                        <span aria-hidden="true">&times;</span>
```

```
                                </button>
                            </div>`
            $("body").append(htmlTemp)
        }

        function number2decimals(num) {
            return (Math.round(num * 100) / 100).toFixed(2);
        }
    </script>
    <style>
        .card {
            cursor: pointer;
        }

        html {
            overflow: auto;
        }
    </style>
</head>

<body>
    <nav class="navbar navbar-expand-sm navbar-dark bg-sparta justify-content-end">
        <a class="navbar-brand" href="/goods">
            <img src="/static/logo_big_tr.png" width="30" height="30" class="d-inline-block align-top" alt="">
            스파르타 쇼핑몰
        </a>
        <button class="navbar-toggler ml-auto" type="button" data-toggle="collapse"
            data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="true"
            aria-label="Toggle navigation"><span class="navbar-toggler-icon"></span></button>
        <div class="navbar-collapse collapse flex-grow-0 ml-auto" id="navbarSupportedContent" style="">
            <ul class="navbar-nav mr-auto text-right">
                <li class="nav-item" id="link-cart">
                    <a class="nav-link" href="/cart">
                        장바구니<i class="fa fa-shopping-cart ml-2" aria-hidden="true"></i>
                    </a>
                </li>
                <li class="nav-item" id="link-logout">
                    <a class="nav-link" data-toggle="modal" data-target="#signOutModal">
                        로그아웃<i class="fa fa-sign-out ml-2" aria-hidden="true"></i>
                    </a>
                    <div class="modal text-left" id="signOutModal" tabindex="-1" role="dialog"
                        aria-labelledby="signOutModalLabel" aria-hidden="true">
                        <div class="modal-dialog" role="document">
                            <div class="modal-content">
                                <div class="modal-header">
                                    <h5 class="modal-title" id="signOutModalLabel">로그아웃</h5>
                                    <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                                        <span aria-hidden="true">&times;</span>
                                    </button>
                                </div>
                                <div class="modal-body">
                                    로그아웃하시면 장바구니가 사라져요!
                                </div>
                                <div class="modal-footer">
                                    <button type="button" class="btn btn-outline-sparta" data-dismiss="modal">취소
                                    </button>
                                    <button type="button" class="btn btn-sparta" onclick="sign_out()">로그아웃하기
                                    </button>
                                </div>
                            </div>
                        </div>
                    </div>
                </li>
            </ul>
        </div>

    </nav>
    <div class="wrap">
        <div>
            <div class="form-group row mr-0">
                <label for="categorySelect" class="col-4 col-form-label">카테고리</label>
                <select class="form-control col-8" id="categorySelect">
                    <option value="" selected>전체</option>
                    <option value="drink">음료</option>
                    <option value="food">음식</option>
                </select>
            </div>
        </div>
        <div id="goodsList" class="mb-5">
            <div>
                <div class="card mb-2" onclick="location.href='#'">
                    <div class="row no-gutters">
                        <div class="col-sm-5" style="background: #868e96;">
                            <img src="https://cdn.pixabay.com/photo/2016/09/07/19/54/wines-1652455_1280.jpg"
                                class="card-img-top h-100" alt="...">
                        </div>
                        <div class="col-sm-7 d-flex">
```

```
                            <div class="card-body flex-fill">
                                <div class="card-title mb-auto">
                                    <h5 style="display: inline">상품 1</h5>
                                    <span class="card-price ml-2">$6.20</span>

                                </div>
                                <span class="badge badge-secondary">drink</span>
                                <!--                                <p class="card-text"><small class="text-muted">drink</sma
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>


</body>

</html>
```

👉 다음에 ejs 파일을 추가했으니 그곳으로 연결하는 router를 추가해볼게요! 그리고 추가된 route 로 접속을 해보겠습니다.

```
app.get('/home', (req, res) => {
    res.render('index');
})
```



🎉 축하드려요!!! express 와 ejs를 이용해서 이제 쇼핑몰의 목록 페이지를 만들었네요. 벌써 2주차까지 완주하신 여러분 너무너무 대단합니다. 그동안 쇼핑몰을 만들기위해서 기반을 다지는 일은 이제 거의 끝입니다. 앞으로는 이제 실제 쇼핑몰에 데이터를 추가하고 api 와 페이지들을 만들면서 바로바로 눈에 보이는 재미있는 실습들을 진행할거에요. 남은 3주도 더 힘내서 힘차게 진행해보도록 하겠습니다. 화이팅입니다!! 🙌🙌🙌🙌🙌

08. 2주차 끝 & 숙제 설명

▼ 15) 목록 상품 선택하면 **상세 페이지 추가해보기**

```
http://3.36.86.60:3001/detail?goodsId=4
```

상품 목록페이지에서 상품을 클릭하면 위와같이 상세 페이지가 나옵니다. 우리도 아래 코드스니펫 파일을 추가해서 상품 상세 페이지를 추가해보도록 하겠습니다.

▼ **[코드스니펫] detail.ejs**

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8" />
    <meta
      name="viewport"
      content="width=device-width, initial-scale=1, shrink-to-fit=no"
    />
    <link rel="icon" href="/static/favicon.ico" type="image/x-icon" />
    <link rel="shortcut icon" href="/static/favicon.ico" type="image/x-icon" />
    <!-- Bootstrap CSS -->
    <link
      rel="stylesheet"
      href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css"
      integrity="sha384-TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkgIXeMed4M0jlfIDPvg6uqKI2xXr2"
      crossorigin="anonymous"
    />

    <!-- Font Awesome CSS -->
    <link
      href="//maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css"
      rel="stylesheet"
    />
    <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS -->
    <script
      src="https://code.jquery.com/jquery-3.5.1.js"
      integrity="sha256-QWo7LDvxbWT2tbbQ97B53yJnYU3WhH/C8ycbRAkjPDc="
      crossorigin="anonymous"
    ></script>
    <script
      src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"
      integrity="sha384-9/reFTGAW83EW2RDu2S0VKaIzap3H66lZH81PoYlFhbGU+6BZp6G7niu735Sk7lN"
      crossorigin="anonymous"
    ></script>
    <script
      src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/js/bootstrap.min.js"
      integrity="sha384-w1Q4orYjBQndcko6MimVbzY0tgp4pWB4lZ7lr30WKz0vr/aWKhXdBNmNb5D92v7s"
      crossorigin="anonymous"
    ></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery-cookie/1.4.1/jquery.cookie.js"></script>

    <link href="/static/mystyle.css" rel="stylesheet" />
    <script>
      const queryString = window.location.search;
      const urlParams = new URLSearchParams(queryString);
      const goodsId = urlParams.get("goodsId");

      $(document).ready(function() {
        get_detail();
        $("#numberSelect").on("change", function() {
          let orderNum = parseInt($(this).val());
          $("#orderNumber").html(
            `<small class="mr-2 text-muted">총 수량 ${orderNum}개</small>${number2decimals(
              orderNum * sessionStorage.getItem("goodsPrice")
            )}`
          );
          sessionStorage.setItem("orderNum", orderNum);
        });
      });

      function sign_out() {
        $.removeCookie("mytoken", { path: "/" });
        $.removeCookie("userName", { path: "/" });
        window.location.href = "/";
      }

      function get_detail() {
        let goods = [
```

```
            {
                "_id": "600fa6e49539b288e3c5a2cf",
                "goodsId": 4,
                "name": "상품 4",
                "thumbnailUrl": "https://cdn.pixabay.com/photo/2016/09/07/02/11/frogs-1650657_1280.jpg",
                "category": "drink",
                "price": 0.1
            },
            {
                "_id": "600fa6e49539b288e3c5a2ce",
                "goodsId": 3,
                "name": "상품 3",
                "thumbnailUrl": "https://cdn.pixabay.com/photo/2016/09/07/02/12/frogs-1650658_1280.jpg",
                "category": "drink",
                "price": 2.2
            },
            {
                "_id": "600fa6e49539b288e3c5a2cd",
                "goodsId": 2,
                "name": "상품 2",
                "thumbnailUrl": "https://cdn.pixabay.com/photo/2014/08/26/19/19/wine-428316_1280.jpg",
                "category": "drink",
                "price": 0.11
            },
            {
                "_id": "600fa6e49539b288e3c5a2cc",
                "goodsId": 1,
                "name": "상품 1",
                "thumbnailUrl": "https://cdn.pixabay.com/photo/2016/09/07/19/54/wines-1652455_1280.jpg",
                "category": "drink",
                "price": 6.2
            }
        ]

  let goodsDetail = goods.find((v) => v.goodsId == goodsId);
  if (!goodsDetail) {
      goodsDetail = goods[0]
  }

  $("#goodsUrl").attr("src", goodsDetail["thumbnailUrl"]);
  $("#goodsName").text(goodsDetail["name"]);
  $("#goodsPrice").text("$" + number2decimals(goodsDetail["price"]));

  sessionStorage.setItem("goodsId", goodsId);
  sessionStorage.setItem("goodsName", goodsDetail["name"]);
  sessionStorage.setItem("goodsPrice", goodsDetail["price"]);
  sessionStorage.setItem("orderNum", 1);
}

function addCart() {
  $.ajax({
    type: "POST",
    url: `/api/goods/${goodsId}/cart`,
    data: {
      quantity: sessionStorage.getItem("orderNum")
    },
    error: function(xhr, status, error) {
      if (status == 400) {
        alert("존재하지 않는 상품입니다.");
      }
      window.location.href = "/goods";
    },
    success: function(response) {
      if (response["result"] == "success") {
        $("#cartModal").modal("show");
      }
    }
  });
}

function buyNow() {
  sessionStorage.setItem(
    "priceSum",
    sessionStorage.getItem("goodsPrice")
  );
  sessionStorage.setItem(
    "cart",
    JSON.stringify([
      {
        goodsName: sessionStorage.getItem("goodsName"),
        quantity: sessionStorage.getItem("orderNum")
      }
    ])
  );
  window.location.href = "/order";
}
```

```
      function number2decimals(num) {
        return (Math.round(num * 100) / 100).toFixed(2);
      }
    </script>
    <title>스파르타 쇼핑몰 | 상품 상세</title>

    <style></style>
  </head>

  <body>
    <nav
      class="navbar navbar-expand-sm navbar-dark bg-sparta justify-content-end"
    >
      <a class="navbar-brand" href="/goods">
        <img
          src="/static/logo_big_tr.png"
          width="30"
          height="30"
          class="d-inline-block align-top"
          alt=""
        />
        스파르타 쇼핑몰
      </a>
      <button
        class="navbar-toggler ml-auto"
        type="button"
        data-toggle="collapse"
        data-target="#navbarSupportedContent"
        aria-controls="navbarSupportedContent"
        aria-expanded="true"
        aria-label="Toggle navigation"
      >
        <span class="navbar-toggler-icon"></span>
      </button>
      <div
        class="navbar-collapse collapse flex-grow-0 ml-auto"
        id="navbarSupportedContent"
        style=""
      >
        <ul class="navbar-nav mr-auto text-right">
          <li class="nav-item" id="link-cart">
            <a class="nav-link" href="/cart">
              장바구니<i
                class="fa fa-shopping-cart ml-2"
                aria-hidden="true"
              ></i>
            </a>
          </li>
          <li class="nav-item" id="link-logout">
            <a class="nav-link" data-toggle="modal" data-target="#signOutModal">
              로그아웃<i class="fa fa-sign-out ml-2" aria-hidden="true"></i>
            </a>
            <div
              class="modal text-left"
              id="signOutModal"
              tabindex="-1"
              role="dialog"
              aria-labelledby="signOutModalLabel"
              aria-hidden="true"
            >
              <div class="modal-dialog" role="document">
                <div class="modal-content">
                  <div class="modal-header">
                    <h5 class="modal-title" id="signOutModalLabel">로그아웃</h5>
                    <button
                      type="button"
                      class="close"
                      data-dismiss="modal"
                      aria-label="Close"
                    >
                      <span aria-hidden="true">&times;</span>
                    </button>
                  </div>
                  <div class="modal-body">
                    로그아웃하시면 장바구니가 사라져요!
                  </div>
                  <div class="modal-footer">
                    <button
                      type="button"
                      class="btn btn-outline-sparta"
                      data-dismiss="modal"
                    >
                      취소
                    </button>
                    <button
                      type="button"
                      class="btn btn-sparta"
```

```
                  onclick="sign_out()"
                >
                  로그아웃하기
                </button>
              </div>
            </div>
          </div>
        </div>
      </li>
    </ul>
  </div>
</nav>
<div class="wrap">
  <div class="row no-gutters">
    <div class="col-sm-5">
      <img
        src="https://cdn.pixabay.com/photo/2016/09/07/19/54/wines-1652455_1280.jpg"
        class="card-img-top h-100"
        alt="..."
        id="goodsUrl"
      />
    </div>
    <div class="col-sm-7 card-body px-3">
      <div class="flex-fill mt-3">
        <div class="d-flex justify-content-between mb-3">
          <h5 style="display: inline" id="goodsName">상품 1</h5>
          <span class="card-price" id="goodsPrice">$6.20</span>
        </div>

        <div class="form-group row mr-0">
          <label for="numberSelect" class="col-4 col-form-label"
            >수량</label
          >
          <select class="custom-select col-8" id="numberSelect">
            <option selected value="1">1개</option>
            <option value="2">2개</option>
            <option value="3">3개</option>
            <option value="4">4개</option>
            <option value="5">5개</option>
          </select>
        </div>
        <hr />
        <div class="row mb-3">
          <div class="col-5">총 상품금액</div>
          <div class="col-7 text-right" id="orderNumber">
            <small class="mr-2 text-muted">총 수량 1개</small>$6.20
          </div>
        </div>
        <div class="row d-flex justify-content-around">
          <div class="col-6 pr-2">
            <button
              type="button"
              class="btn btn-outline-sparta btn-block"
              onclick="addCart()"
            >
              장바구니
            </button>
          </div>
          <div class="col-6 pl-2">
            <button
              type="button"
              class="btn btn-sparta btn-block"
              onclick="buyNow()"
            >
              바로 구매
            </button>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
<div
  class="modal text-left"
  id="cartModal"
  tabindex="-1"
  role="dialog"
  aria-labelledby="cartModalLabel"
  aria-hidden="true"
>
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="cartModalLabel">알림</h5>
        <button
          type="button"
          class="close"
```

```
              data-dismiss="modal"
              aria-label="Close"
            >
              <span aria-hidden="true">&times;</span>
            </button>
          </div>
          <div class="modal-body">
            장바구니에 담았습니다! 장바구니로 갈까요?
          </div>
          <div class="modal-footer">
            <button
              type="button"
              class="btn btn-outline-sparta"
              data-dismiss="modal"
            >
              취소
            </button>
            <button
              type="button"
              class="btn btn-sparta"
              onclick='window.location.href="/cart"'
            >
              장바구니
            </button>
          </div>
        </div>
      </div>
    </div>
  </body>
</html>
```

09. 2주차 숙제 답안 코드

▼ **[코드스니펫] - 2주차 숙제 답안 코드**

전체 코드

▼ index.js

```
app.get('/detail', (req, res) => {
  res.render('detail')
})
```