

오늘은 종합수업이다. 여러분들이 이제까지 배운 기술로 경대도시락이라는 프로그램의 “일부만” 만들어볼 것이다.

경대도시락은 다음과 같은 프로그램이다.

직접 보고싶다면 다음 링크로 접속하자.

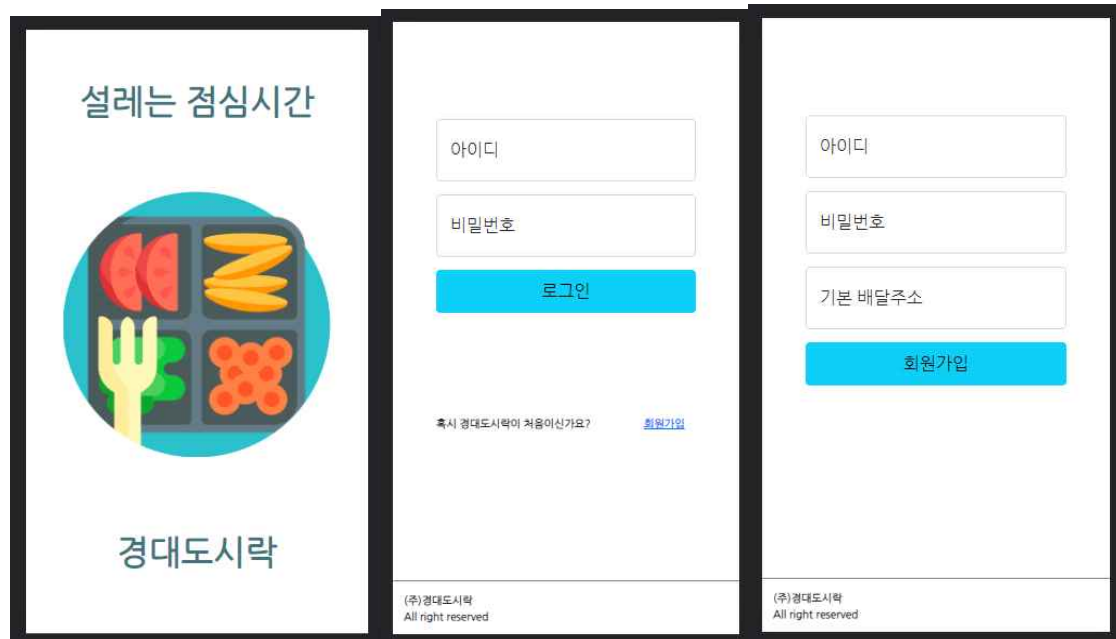
<http://203.255.3.144:1993>

모바일에 최적화되어있다. 크롬 개발자도구에서 iPhone5 사이즈에 맞춰주자.

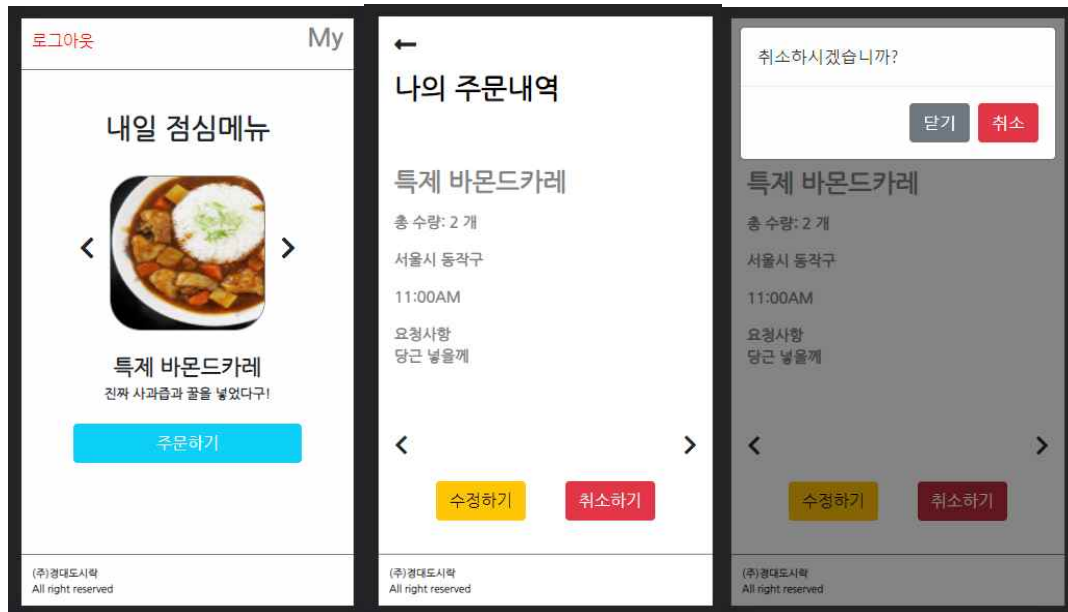
소스코드는 다음 깃허브 링크에 제공된다. 도전하고 싶은 학생은, 깃허브를 보고 직접 모든 것을 만들어보자. 지금까지 배운 지식이면 도전할만하다.

<https://github.com/JonJaryongLee/lunchbox-static>

모르는 코드가 나온다면, 조교 이자룡에게 직접 카톡해도 되고, 구글링을 하면 더 좋다.



인트로, 로그인창, 회원가입창이다.



## 메인화면

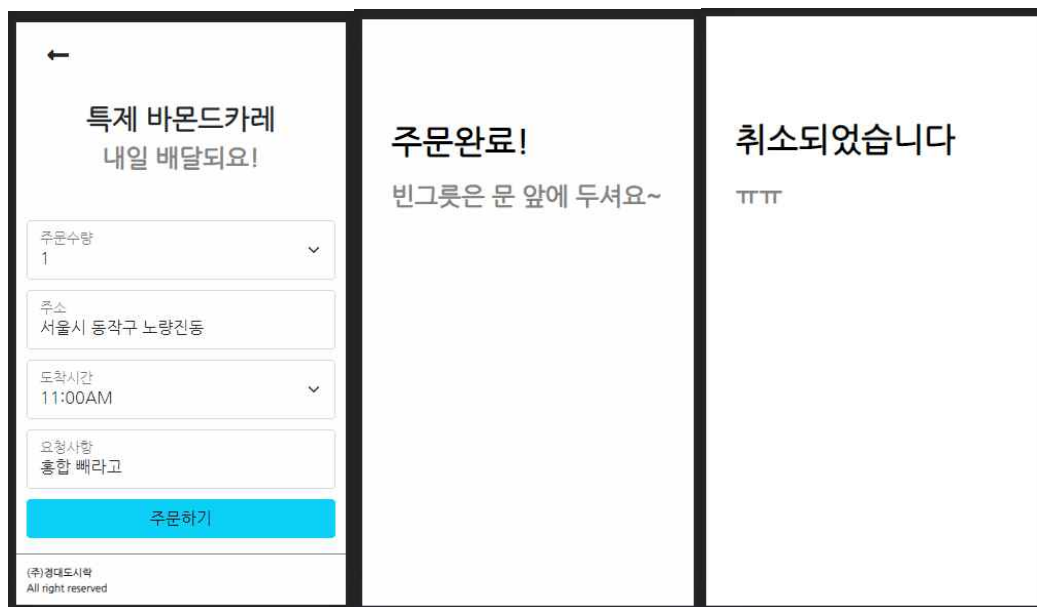
내일, 모레, 3일 뒤 메뉴를 골라 선택할 수 있다.

주문하기 버튼을 누르면 주문하기로 넘어간다.

My 버튼을 누르면 내 주문내역을 확인할 수 있다.

## 나의 주문내역

주문내역을 확인하고, 수정, 취소할 수 있다.



주문하기를 눌렀거나, 주문수정을 눌렀을 때 나오는 주문창이다.

주문이 완료되면 주문완료 안내가 뜨고,

My 를 클릭할 때 나오는 나의 주문내역에서 특정한 주문을 취소하면 취소안내가

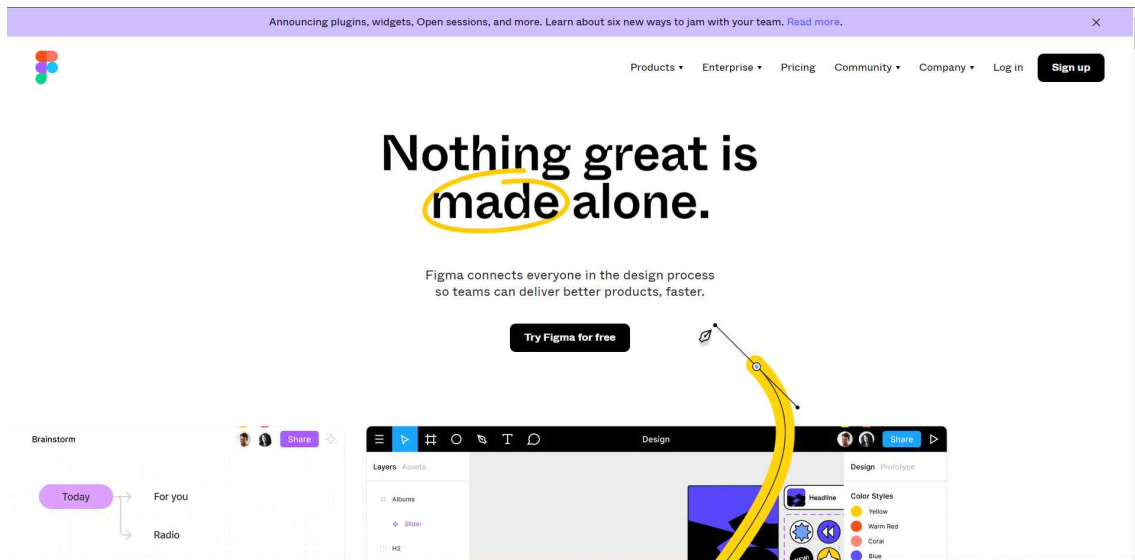
뜯다.

일단, 전체적인 개발 과정을 생각해보자.

## 1. UI

경대도시락을 템플로 만들었다고 생각하면, 여러분은 가장 먼저 이 프로그램의 UI 부터 뜯 것이다.

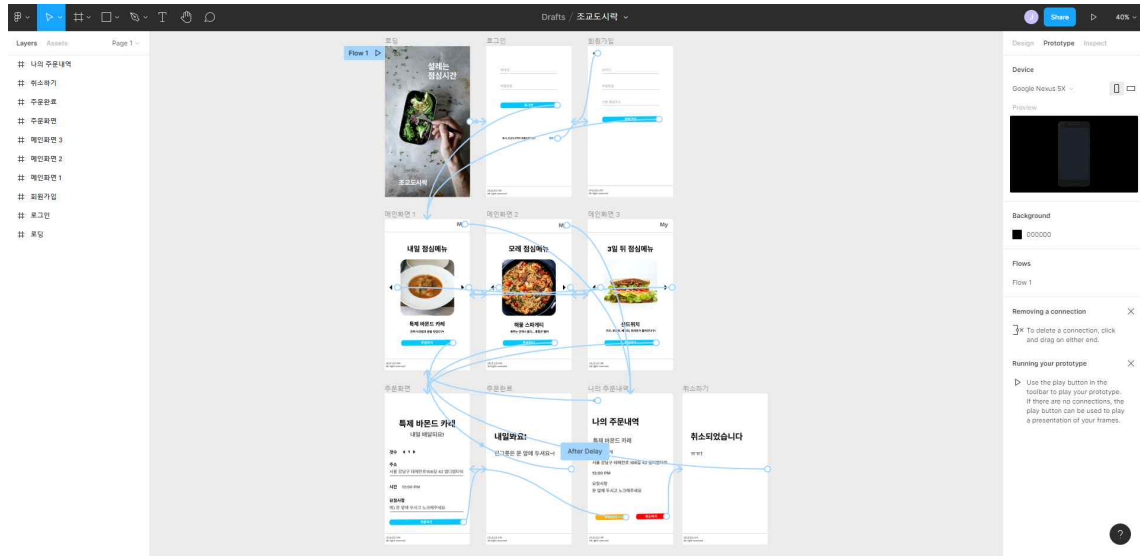
UI 를 예쁘게 짜는 웹사이트로 추천하는 건 Figma 다.



Figma 는 현 시점에서 UI 를 만들 때 쓰는 업계 표준이다.

무료이며, 프로토타입이라는 것을 제공하는데, 자세하게 배워보고싶은 학생은 유튜브의 피그마 강좌를 찾아서 보도록 하자.

다음은 경대도시락의 Figma 이다. 곡선은 프로토타입을 연결해놓은것이다.



물론, UI 는 프로그램을 만들기위한 초안밖에 안 되기 때문에 UI에만 집중하는것은 아무 의미가 없으며, 본 조교도 UI를 얼마나 예쁘게 짰는가는 평가하지 않는다. UI 에 힘빼지 말자.

가장 편한 건 사실 펜과 종이다. Figma 를 꼭 배울 필요도 없고, 펜과 종이만 있으면 UI 정도는 금방 만들 수 있다.

## 2. 프로그램의 기능

우리 프로그램이 어떤 기능을 가질지 생각해봐야한다.

경대도시락

로그인

회원가입

도시락 주문

주문수정

주문취소

이 기능에 맞추어서 어떤 html 파일을 만들어야될지, 그 html 과 연관된 어떤 파일들이 있을지 생각나는대로 적어보자.

## 3. DB 설계

이건 제대로 하려면 JOIN 과 ERD 를 반드시 알아야하지만 우린 배운적이 없다. 따라서, 여러분이 DB와 백엔드 코드(PHP등) 없는 프로그램을 만들어도 이해하겠다.

DB 연결과 백엔드는 평가대상이 아니다. 정 DB를 만들고 싶으면 테이블은 하나만 해라. 나처럼 3개 다루려면 primary key 뿐만 아니라 foreign key 도 다뤄야하기 때문에 알아야할 지식이 많다.

그러면 멀쩡한 프로그램 못 만들지 않나요?

난 여러분들이 멀쩡한 프로그램 만들거라고 기대 안했다. 프로토타입이라고 들어봤는가? 돌아가는 “척” 하는 프로그램을 만드는 게 이번 수업 목표다.

그래도 경대도시락은 DB가 있으니깐, 설명해보겠다.

총 세 개의 테이블로 구성되어 있다.

## menu 테이블

menu_id	menu_day	menu_name	description	image_link
0	내일	특제 바론드카레	진짜 사과즙과 꿀을 넣었다구!	/images/menu/japanese-curry.jpg
1	모레	해물 스파게티	새우는 언제나 좋다... 홍합은 뺐어.	/images/menu/shrimp-spaghetti.jpg
2	3일 뒤	샌드위치	치즈, 양상추, 베이컨, 토마토가 들어갔다구!	/images/menu/sandwich.jpg

사용자가 주문할 메뉴의 정보. image\_link 는 서버에서 접근할 이미지 경로다.

참고로, DB 엔 원래 사진, 음악같은거 못 담는다. 이런식으로 경로로만 저장한다.

## order 테이블

order_id ↑	user_id	menu_id	quantity	address	what_time	etc	order_time
4	joayo55	0	1	제주시 애월읍	11:00AM	소스 넣지마세요	2021-09-03 13:57:25
5	joayo55	2	2	제주시 애월읍	12:00PM	맛있게 해주세요	2021-09-05 13:54:58
6	joayo55	1	3	인천광역시 옹진군	01:00PM	EMPTY	2021-09-05 13:55:37
14	jaeseuk97	1	1	진주대로 501 경상대학교 30동 310호	12:00PM	EMPTY	2021-09-05 19:06:49
15	wjs5025	1	3	경상대 기숙사 13	11:00AM	ㅇㅇㄹ	2021-09-05 19:06:55
17	wjs5025	2	3	경상대 기숙사 8동 ㄴㅇ	12:30PM	매름	2021-09-05 19:08:59
18	wjs5025	1	1	경상대 기숙사 8동	11:00AM	EMPTY	2021-09-05 19:09:16
21	wjs5025	1	3	경상대 기숙사 8동	11:00AM	EMPTY	2021-09-05 19:09:44
22	jaeseuk97	2	2	EMPTY	11:00AM	EMPTY	2021-09-05 19:09:49
24	jaeseuk97	0	1	EMPTY	11:00AM	EMPTY	2021-09-05 19:10:26
27	ghkd3531	2	1	가좌동	01:30PM	EMPTY	2021-09-05 19:13:45
28	ghkd3531	0	1	가좌동	11:00AM	console.log("테스트")	2021-09-05 19:14:33
32	ghkd3531	1	1	가좌동	11:00AM	EMPTY	2021-09-05 19:20:10
33	circlezero	0	1	30동	11:00AM	EMPTY	2021-09-05 19:21:38

사용자의 주문내역이다. 다양한 데이터타입이 쓰였다.

## person 테이블

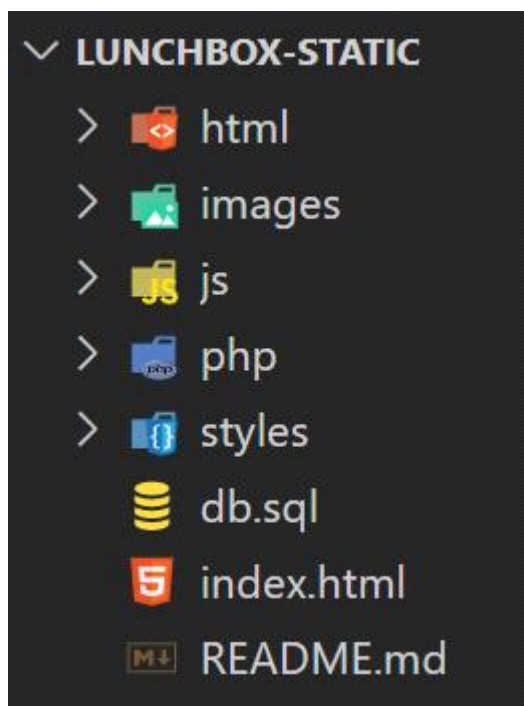
user_id	password	default_address	default_request
assistant0603	\$2y\$10\$888xsp3j7E/JES8sbJaupQxwAJNKg5nnIU8ASUv9jtDFQemziPy	서울시 동작구 노랑진동	홈 배라고
circlezero	\$2y\$10\$QDYpCOpwFXe9CqiMFi/p8.eehD7J8W4cC8R4CHopHDi7sA2a4ouY.	30동	NULL
circlezero	\$2y\$10\$xEHxSkThiFp8jxs/p57/SOlpuvdFgZnmiWV0skYb8tOxurj8w34Ym	30동	NULL
dfsdfsdfs	\$2y\$10\$y1bRejbPtVNe/vNh/ZSh0OGGA1dXexjq2Pn9I4vapBfgtdjJ4Fdbe	sdfsfsdf	NULL
eooomji	\$2y\$10\$795XnnCvYbJOlgLwpGSY9j.48GgtAooxldztuGrhaSLbP7eSq21u3K	컴퓨터과학관 415호	EMPTY
gdgd	\$2y\$10\$AohTy7.wmz3moFvumKwJfehco/pkcO3vak6.OxVolsD3evp8GCcla	ㅎㅎㅎ	NULL
ghkd3531	\$2y\$10\$2iqdX27akcyhfWwxESNWnu3PHB1GrdWxa5f60/Mjuknksr6vUHZT2	가좌동	NULL
hihi	\$2y\$10\$3fRjkkGzAaib8JjV7iqJcOtMNYiuLMIXzzQS84jb5diw0mud0xlEG	경대 30동 409호	NULL
hyunhoo	\$2y\$10\$iUR20VAvkSrrer0Be6ZquVwlfzvhGh/Sny7bVSGaQmM7OiviiKce	집	EMPTY

사용자 가입정보다. 패스워드는 해시 암호처리되서 관리자인 나도 각자의 암호가 뭔지 모른다.

## 4. 파일 구조 작성

트리 형태로 그리는 게 가장 깔끔하다. 각 화면이 무엇을 하고, 다른 어떤 파일과 관련이 있는지 작성하는게 좋다.

경대도시락의 파일 구성은 다음과 같다.



html 디렉터리

images 디렉터리

js 디렉터리

php 디렉터리

styles 디렉터리

html 모음이다. index.html 은 제외한다.

프로그램에서 쓸 이미지 모음이다.

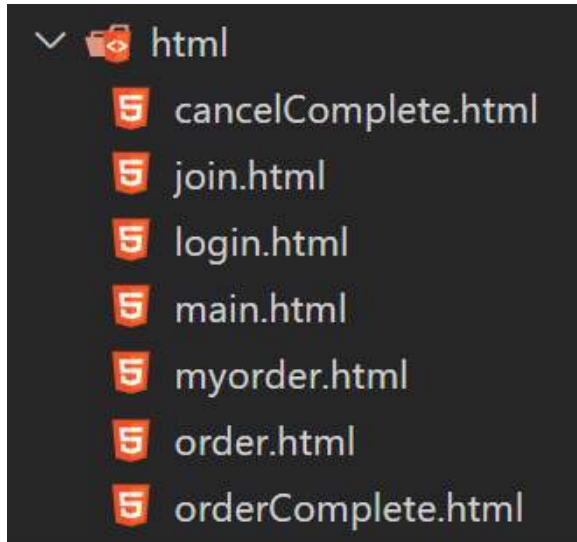
JavaScript 파일 모음이다.

php 파일 모음, 즉 백엔드 코드다.

css 파일 모음이다. 디렉터리 이름을 css 로 지정해도 된다

db.sql                      초기 데이터베이스 세팅 파일  
README.md                리드미는 이 프로그램의 사용설명서다.  
여러분이 취업을 하려면 가장 공들여서 작성해야되는 파일  
마크다운 문법을 따른다. 작성법은 구글링해서 알아보자

본 조교는 html 파일 이름을 css 와 js 에 그대로 쓰는 습관을 가지고 있으므로,  
html 디렉터리만 보도록 하자. 가장 처음 마주하는 index.html 은 제외다.



하는 역할만 빠르게 알아보자.

login.html	로그인
join.html	회원가입
main.html	메인
myorder.html	내 주문 확인
order.html	주문하기
cancelComplete.html	취소완료 안내페이지
orderComplete.html	주문완료 안내페이지

읽어보면 알겠지만, 사실 한국어로 굳이 바뀌서 설명할 필요도 없다.  
개발자는 작명가다. 이름을 잘 지어야한다. 변수, 함수, 파일, 객체 등등, 이름을 그  
목적에 맞게 지을 줄 알아야한다.

여기에 확장자만 js, css 로 바뀌서 js와 styles 디렉터리에 싹 다 넣으면 일차적인  
세팅은 끝난 것이다.

Q. 그냥 만들어가면서 하나하나 넣으면 안되요?

A. 그래도 되고, 나도 그런 식으로 개발했었다. 하지만, 미리 큰 틀을 설계해놓고 세부적인 것을 개발했을때와, 작은것부터 생각나는대로 개발했을때의 깔끔함은 천지차이다. 생각나는대로 개발하면, 변경사항이 있을 때 이것저것 엉켜있어서 바꾸기도 힘든데, 이것을 개발자들은 스파게티 코드 라고 한다.

## 5. 도전해볼만한 코드

경대도시락의 모든 것을 다 개발하진 않는다고 했다.

가장 좋은 학습법은, 여러분들이 이제까지 배운 지식으로 내 코드를 이해하려고 노력하는 것이다.

이 중에서 해볼만한 후보군들을 골라보면,

### index.html

이 파일같은 경우는 html 디렉터리 안에 있지 않지만, 맨 첫페이지에 나타나는 웰컴페이지다.

3초 동안 등장했다가 로그인으로 넘어가는데,

이를 위해 setTimeout() 과 location 객체를 쓸 것이라 자동으로 페이지 전환을 연습하는 좋은 예제다.

### 로그인 및 회원가입

**하지 말자.** 여러분이 실수하는 것 중 대표적인것은, 로그인 기능 구현한다고 모든 힘을 빼는 것이다.

\* 로그인 및 회원가입은 굉장히 어려운 기능이다. 여러분들이 사이트를 만들 때, 당연히 로그인부터 만들어야겠다 생각하는데, 로그인 만들다가 다른 건 만들지도 못하는 경우가 많을 정도로 어렵다.

로그인에 필요한 기술들을 기본적으로 생각해보자.

당연히, 서버로 입력값을 JSON 으로 보낸 후, DB에 접근해서 결과를 받아야한다.

이것을 가능하게하는 서버코드를 작성해야한다.

받은 결과로 로그인 성공여부를 판단해, 실패하면 경고창을 띄워야한다.

성공하면 화면을 전환하는데, 로그인 성공한 아이디를 다른 파일에서 사용할 수 있도록 세션변수를 등록해야한다.

추가적으로,

해당 세션변수는 일정 시간이 지나면 해제되도록 조치해야한다.

비밀번호는 절대 string 이 되서는 안되며, hash 암호화

처리해야한다.

읽어보면 알겠지만 절대 만만한 게 아니다. 진짜, 로그인 회원가입만 잘해도 웬만한 건 다 할 수 있을 것이다.

특히, 세션은 가르치지도 않았기 때문에 로그인과 회원가입은 생략한다.



\* db는 안 만들어도 좋다고 계속 말했다. db 없이, db 와 비슷하게 로그인을 구현하려면 어떻게 하면 될까? JavaScript 의 객체와 변수를 활용해보자. 페이지가 넘어가서 데이터를 보존할 수 없으면, 새 페이지에 샘플아이디를 넣어두면 될 것이다. 완벽하게 작동되는 프로그램이 아닌, 프로토타입을 만든다는 생각으로 프로젝트에 도전하자.

다른 후보군을 살펴보자.

**메인화면**                      경대도시락의 메인페이지다.  
버튼을 클릭하면 이미지 및 음식정보가 바뀔 것이다. 훌륭한  
돔 조작 예제다. 도전해볼만하지만, 아래의 예제가 더 좋다.

**내 주문내역 확인**            이 페이지는 돔 조작이 포함되어있을뿐만 아니라,  
db 데이터 유무에 따라 같은 페이지에서 보이는 화면이  
다르다. 게다가, 취소 모달도 연습해 볼 수 있다.  
메인화면과 비교해봤을 때, 오히려 더 좋은 예제다.

**주문하기**                      이 프로그램 전체에서 로그인, 회원가입을 제외하고 가장  
난이도가 높다. 시간변환을 위해 객체를 두 개 정의해서  
데이터를 맞춰준다던지, 유저의 기본배달지와 요청사항을  
받아오기도 하고, 이 페이지에서 “주문수정” 기능도  
담당하며, db에 insert 요청도 보내야한다. 하지만 코드  
안에서 복잡하기만 하지 딱히 크게 새로운 건 없으므로,  
여러분들이 직접 도전해 볼 수 있도록 하겠다.

그래서 내린 결론은, 2천줄 가까이 되는 이 거대한 프로그램 중에 딱 두가지 페이지,  
**웰컴페이지**와 **내 주문 확인 페이지**만 구현하는 게 이번시간 목표다.

그럼 db 테이블은 딱 하나만 있으면 된다. 사용자 주문내역을 담당하는 order  
테이블이다.

\* 여러분들이 구상한 프로그램의 “특정 기능” 만 제대로 구현해도 프로젝트에서 좋은  
점수를 받을 수 있다. 제발, 작은 것을 만들어라. 여러분은 대학생할 4년동안  
컴퓨터과학과에서 경험할 3가지 프로젝트 과목중에 첫번째 수업 수강생이며,  
2학년이다. 최대한 작은 것을 만들고, 시간이 남으면 다른 기능들을 만들어보라.

## db 테이블 만들기

다음과 같은 샘플 테이블을 만들 것이다.

테이블 이름은 ordered 이다. 왜 order 가 아니냐면, MySQL과 MariaDB 에서 order 는 키워드이기 때문이다.

order_id	menu_name	quantity	address	what_time	etc
1	특제 바몬드 카레	2	제주시 애월읍	11:00AM	안 맵게 해주세요
2	샌드위치	1	경남 함양군	12:00PM	오이 빼주세요
3	해물스파게티	3	경상대 기숙사 8동	01:00PM	

타입 먼저 생각해보자.

order_id	int (primary key auto_increment)
menu_name	text
quantity	int
address	text
what_time	text (date 로 쓸수도 있는데 편의상 text 로 지정)
etc	text

\* 꼭 date 타입을 써야 할 상황은 무엇일까? 만약 해당 시간으로 계산할 일이 있다면 그렇게 해야한다. 그런데 이 프로그램에선 시간에 따라 계산할만한 일이 없기 때문에 text 로 쓴 것이다.

먼저, PowerShell 을 키고 DB 부터 만들자.

```

* 임혜경 C:\Bitnami\wampstack-8.0.12-0\mariadb\bin> .\mysql.exe -u root -p
Enter password: *****
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 14
Server version: 10.4.21-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| test |
+-----+
4 rows in set (0.008 sec)

MariaDB [(none)]> CREATE DATABASE lunchbox;
Query OK, 1 row affected (0.008 sec)

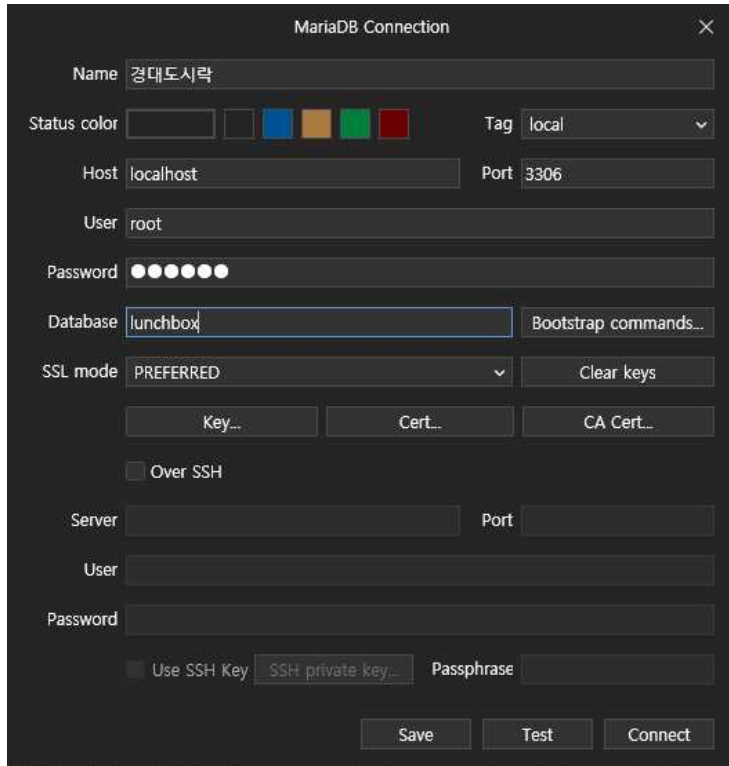
MariaDB [(none)]> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| lunchbox |
| mysql |
| performance_schema |
| test |
+-----+
5 rows in set (0.000 sec)

MariaDB [(none)]>

```

lunchbox 라는 DB 를 만들었다. 이제 TablePlus 를 키자.

접속정보는 다음과 같다.



The image shows a 'MariaDB Connection' dialog box with the following fields and options:

- Name:** 경대도시락
- Status color:** A row of five colored squares (white, blue, orange, green, red).
- Tag:** local (dropdown menu)
- Host:** localhost
- Port:** 3306
- User:** root
- Password:** A password field with six dots.
- Database:** lunchbox
- Bootstrap commands:** A button labeled 'Bootstrap commands...'.
- SSL mode:** PREFERRED (dropdown menu)
- Clear keys:** A button.
- Key...:** A button.
- Cert...:** A button.
- CA Cert...:** A button.
- Over SSH:** An unchecked checkbox.
- Server:** A text field.
- Port:** A text field.
- User:** A text field.
- Password:** A text field.
- Use SSH Key:** An unchecked checkbox.
- SSH private key...:** A button.
- Passphrase:** A text field.
- Buttons at the bottom:** Save, Test, Connect.

이제 쿼리문을 작성하는데, 먼저 VSCode 에 db.sql 로 프로젝트 디렉터리에 저장해놓고 붙여넣는것이 좋다. 프로그램이 잘못되었을 경우 DB 를 처음부터 다시 만들어야 하기 때문이다.

Bitnami htdocs 를 우리 프로젝트 디렉터리로 삼을 것이다. 여기에 작성하자.

```

1  CREATE TABLE `ordered` (
2      `order_id` INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
3      `menu_name` TEXT NOT NULL,
4      `quantity` INT NOT NULL,
5      `address` TEXT NOT NULL,
6      `what_time` TEXT NOT NULL,
7      `etc` TEXT
8  );
9
10 INSERT INTO `ordered`
11 (`menu_name`, `quantity`, `address`, `what_time`, `etc`)
12 VALUES
13 ('특제 바몬드 카레', 2, '제주시 애월읍', '11:00AM', '안 맵게 해주세요'),
14 ('샌드위치', 1, '경남 함양군', '12:00PM', '오이 빼주세요'),
15 ('해물 스파게티', 3, '경상대 기숙사 8동', '01:00PM', NULL);

```

TablePlus 에 입력한 결과는 다음과 같다.

order_id	menu_name	quantity	address	what_time	etc
1	특제 바몬드 카레	2	제주시 애월읍	11:00AM	안 맵게 해주세요
2	샌드위치	1	경남 함양군	12:00PM	오이 빼주세요
3	해물 스파게티	3	경상대 기숙사 8동	01:00PM	NULL

## index.html

만들어볼 목표



3초 뒤, myorder.html

```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8" />
5      <meta http-equiv="X-UA-Compatible" content="IE=edge" />
6      <meta name="viewport" content="width=device-width, initial-scale=1.0" />
7      <title>경대도시락</title>
8  </head>
9  <body>
10 <div class="loading">
11     <h1>설레는 점심시간</h1>
12 <div class="lunchboxIconContainer">
13     <img src="" alt="메인이미지" />
14 </div>
15     <h1>경대도시락</h1>
16 </div>
17 </body>
18 </html>

```

<title> 을 경대도시락으로 변경한다.

먼저, 모든 것을 감싸는 <div> 태그를 하나 만들고, class 를 loading 이라고 하자.


<h1> 은 총 2개다. “설레는 점심시간”, “경대도시락”

그 사이에 들어가는 <img> 태그는 바깥에 <div> 하나로 감싸고, <div> 의 클래스 이름을 lunchboxIconContainer 로 하겠다.

이미지는 아직 경로를 넣지 말고 alt 만 설정해주자.

결과:

## 설레는 점심시간

메인이미지

## 경대도시락

사이트에서 적당한 이미지를 찾아 images 디렉터리 하나 만들어서 담고, 이미지 경로를 설정해주고, 높이를 알맞게 지정하자.

결과:

# 설레는 점심시간



## 경대도시락

그리고 이제 css 로 넘어갈건데, 프로젝트에 css 라는 디렉터리를 하나 달아서, index.css 라고 이름짓고, html 에서 <link> 연결해주자

index.css 에서 디자인해보자. 제일 중요한 건, 모든 것을 가운데로 배치하는 것이다.

```
1  .loading {  
2  |      text-align: center;  
3  |  }
```

loading 클래스는 <body> 바로 아래에서 모든 것을 감싼다. 이렇게, 모든 것을 가운데로 이동시킨다.

우리가 원하는 건, 이것이 3초 정도 화면에 보였다가 myorder.html 로 넘어가는 것이다. 일단, myorder.html 을 프로젝트 디렉터리에 미리 만들어놓고, 기본적인 내용들만 작성해두자.

```

myorder.html > ...
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4  |   <meta charset="UTF-8">
5  |   <meta http-equiv="X-UA-Compatible"
6  |   <meta name="viewport" content="wid
7  |   <title>내 주문</title>
8  </head>
9  <body>
10 |   <h1>내 주문</h1>
11 </body>
12 </html>

```

그럼, 어떻게 “자동으로” 이 페이지로 넘어갈까? 물론 JavaScript 를 사용해야 한다. js 라는 이름의 디렉터리를 하나 만들고, index.js 라는 파일 하나를 그 안에 담고, index.html 에서 연결해주자.

index.js 에 다음과 같이 작성한다.

```

1  setTimeout(() => {
2  |    location.href = "../myorder.html";
3  }, 3000);

```

setTimeout() 을 사용하는데, 3초 지연시키고, location.href 라는 걸 사용했다. 정확하게는, window.location.href 이다. window 는 우리 브라우저 “창” 의 모든 정보를 담은 “객체” 다. 마치 document 와 비슷하지만, document 는 “문서”의 모든 정보고, window 는 “창” 에 대한 모든 정보다. window 는 생략 가능하다.

결과는 다음과 같다.

결과:



설레는 점심시간



경대도시락

3초 후, myorder.html 로 이동

결과:

# 내 주문

성공이다. 이제 myorder.html 파일을 만들어보자

## myorder - DOM 조작으로 css display 변경

myorder.html 을 만들자.

기본 템플릿을 만들고, myorder.js 와 myorder.css 를 각각 만들어 임포트하자.

여기서도 axios 를 사용하니 CDN 을 추가한다.

맨 처음 할 일은, loading 화면을 띄우는 것이다.

다음과 같이 작성하자.

```

1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8" />
5     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6     <meta http-equiv="X-UA-Compatible" content="ie=edge" />
7     <!-- axios -->
8     <script src="https://cdn.jsdelivr.net/npm/axios@0.21.1/dist/axios.min.js"></script>
9     <!-- js -->
10    <script src="./js/myorder.js"></script>
11    <!-- css -->
12    <link rel="stylesheet" href="./css/myorder.css" />
13    <title>경대도시락</title>
14  <body>
15    <div class="loading">loading...</div>
16    <!-- 만약 사용자가 주문을 안 했을 경우 -->
17    <div class="ifNoData">주문이 없네요</div>
18    <!-- 주문 내역이 있을 경우 -->
19    <div class="myOrderMain">주문이 있네요</div>
20  </body>
21 </html>

```

결과:

loading...

주문이 없네요

주문이 있네요

여기서, 로딩 후 상황은 둘로 나뉜다.

주문내역이 하나도 없을 경우엔 ifNoData 클래스가 보여지고,

주문내역이 뭐라도 있으면 myOrderMain 클래스가 보여진다.

이것은 display: none 으로 제어할 수 있다.

db 를 다시 만들긴 번거로우므로, JavaScript 객체 데이터를 사용해 DOM 조작을 연습해보자.

### myorder.css

먼저, loading 클래스를 제외한 나머지 두 개 클래스, ifNoData, myOrderMain 클래스가 “안 보이도록” 설정한다.

```

; > myorder.css > ...
1  .ifNoData, .myOrderMain {
2    display: none;
3  }

```

결과:

# loading...

myorder.js

```
js > JS myorder.js > ...
1  const dataTrue = [
2    {
3      order_id: 1,
4      menu_name: "특제 바몬드 카레",
5      quantity: 2,
6      address: "제주시 애월읍",
7      what_time: "11:00AM",
8      etc: "안 맵게 해주세요",
9    },
10   {
11     order_id: 2,
12     menu_name: "샌드위치",
13     quantity: 1,
14     address: "경남 함양군",
15     what_time: "12:00PM",
16     etc: "오이 빼주세요",
17   },
18   {
19     order_id: 3,
20     menu_name: "해물 스파게티",
21     quantity: 3,
22     address: "경상대 기숙사 8동",
23     what_time: "01:00PM",
24     etc: null,
25   },
26 ];
27 const dataFalse = null;
```

다음과 같은, dataTrue 객체와 dataFalse 변수를 만들어서 테스트를 해볼 것이다.  
서버로부터 이러한 형태의 데이터가 왔다고 “가정” 하는 것이다.  
데이터가 없게 하려면 dataFalse 가 왔다고 “가정”,  
데이터가 있게 하려면 dataTrue 가 왔다고 “가정” 하자.

그리고, 다음과 같이 아래에 쓴다.

```
onload = () => {  
  if(dataTrue) {  
    document.querySelector(".loading").style.display = "none";  
    document.querySelector(".myOrderMain").style.display = "block";  
  }  
};
```

dataTrue 변수는 데이터가 “있다”.

그래서, 로딩을 “꺼버리고” (none)

페이지를 보여주는 클래스인 myOrderMain 을 “컨다”. (block)

결과:

## 주문이 있네요

로딩이 재빠르게 사라지고 “주문이 있네요” 글자가 뜬다.

그렇다. 데이터가 있냐 없냐를 기준으로, 같은 화면에서 여러분이 보여주고자 하는 정보를 다르게 보여줄 수 있다.

코드를 다음과 같이 바꿔보자.

```
onload = () => {  
  if (dataFalse === null) {  
    document.querySelector(".loading").style.display = "none";  
    document.querySelector(".ifNoData").style.display = "block";  
  }  
};
```

dataFalse 변수는 데이터가 “없다”. null 이기 때문이다.

이 경우엔, !dataFalse 라고 써도 똑같은 뜻이지만, 위와 같이 쓰는게 훨씬 이해하기 쉽다.

그래서, 로딩을 “꺼버리고” (none)

주문이 없다는 안내를 담당하는 ifNoData 를 “컨다”. (block)

결과:

## 주문이 없네요

지금부터 비록 테스트데이터로 연습했지만, 실전에선 서버코드를 사용해 실제 서버에서 데이터를 받아와서 이 기능을 구현할 수 있을 것이다.

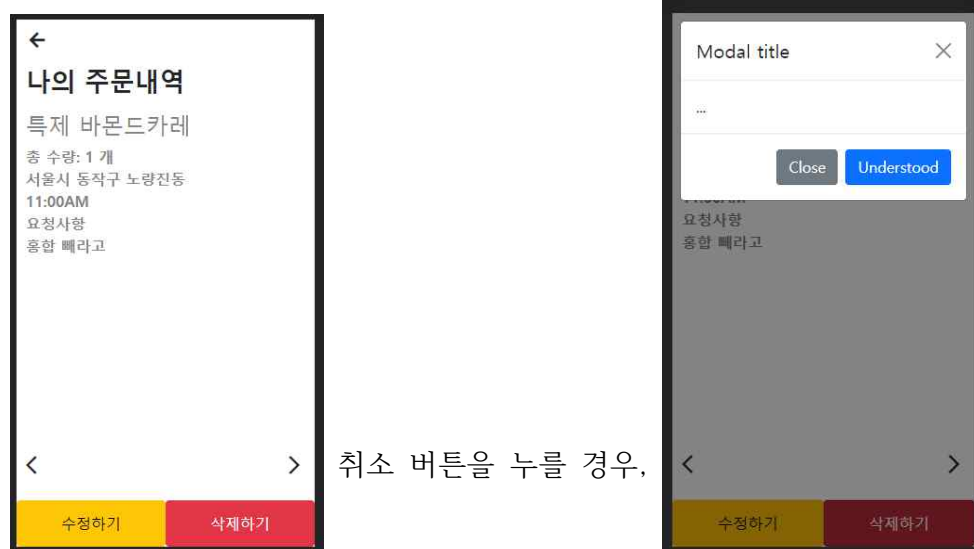
\* 여기서 배울 수 있는 게 한 가지 더 있다. 프론트엔드 작업자의 경우, 백엔드 작업자가 특정 코드를 완성할 때까지 기다리는것은 매우 나쁜 습관이다. 이런 식으로 JavaScript 안에서 테스트 변수를 만들어 화면을 최대한 미리 만들어놓고 백엔드와 연결을 시도해야 한다.

일단, 잠시 멈춰 생각을 해보자. JavaScript 코드는 당연히 바뀌야 한다. 그런데, JavaScript 실행결과를 제대로 받아오려면 PHP 코드를 짜야한다. 이 일을 먼저 할 수도 있다.

아니면 다른 쉬운 걸 먼저 할 수 있다. 화면부터 만들어두는 것이다. 데이터가 없을 때의 화면과, 데이터가 있을 때의 화면을 미리 만든다면, 즉 HTML 과 CSS 부터 만든다면 훨씬 쉽다. 왜냐, HTML 과 CSS의 코드는 웬만해선 안 바뀔 것이기 때문이다.

쉬운것부터 해보자. 먼저, 주문이 없을 때는 더이상 건드릴 게 없다. 만약 메인화면을 작업해두었다면, 뒤로가기 링크 정도 추가할 수 있을 것이다.

우리가 작업할 화면은 데이터가 있을 때의 화면이다. 아직 서버로부터 데이터가 들어오지 않았다면, 다음 형태의 화면을 미리 만들 수 있을 것이다.



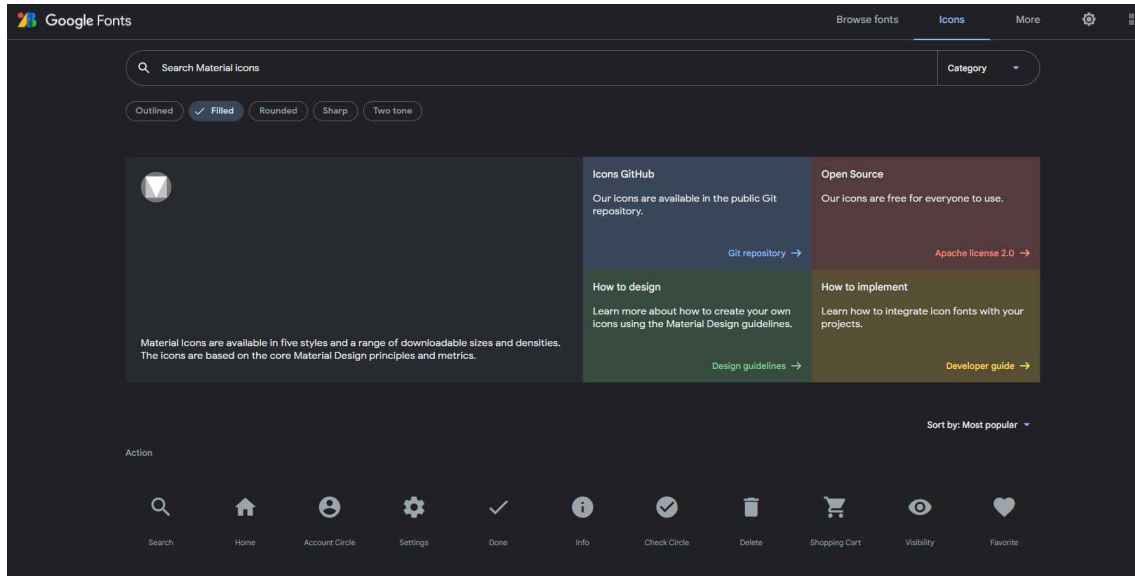
일단, 필요한것들을 생각해보면, 예쁜 버튼과 모달 (알림창) 을 직접 만드려면 매우 고생할 것이기에 Bootstrap 에 미리 만들어진 코드를 사용할 것이다.

아이콘을 가져오기 위한 가장 대표적인 사이트는 Google에서 제공하는 Material icons 다.

레이아웃은 우리가 배운 grid 로 만들어본다.

Bootstrap 홈페이지에 들어가서 CDN 부터 받아오자. Bootstrap 을 잘 모르겠다면, 자료실에 올려둔 Bootstrap 강의를 들어보길 바란다.

다음, 구글에 material icon 이라고 치고, 다음 사이트에 접속하자.



여기서, 네 개의 색 상자 중 git repository 버튼을 클릭

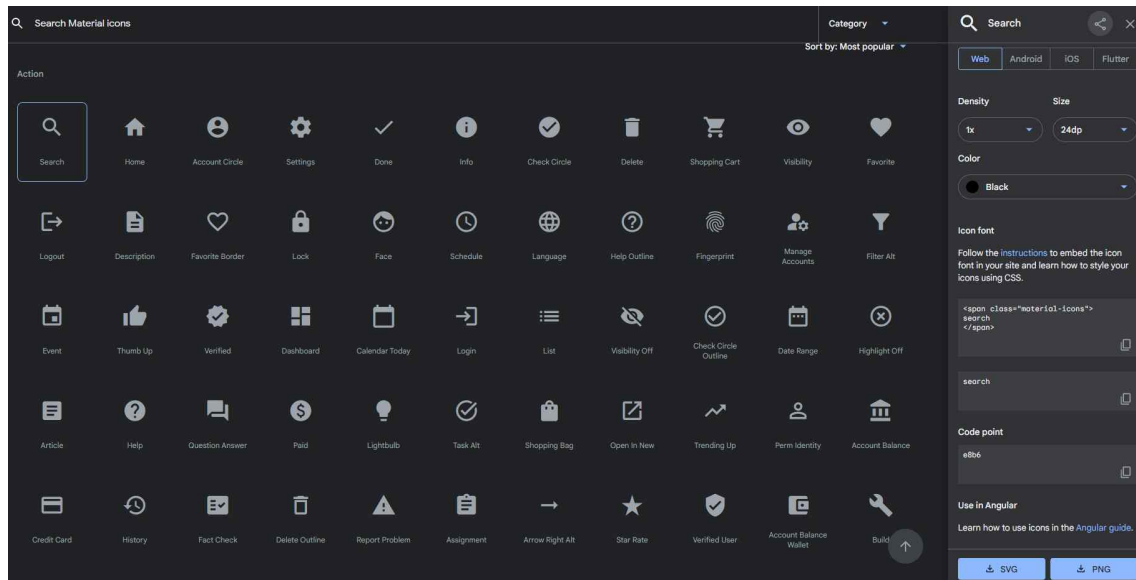
## Using a font

The `font` folder contains pre-generated font files that can be included in a project. This is especially convenient for the web; however, it is generally better to link to the web font hosted on Google Fonts:

```
<link href="https://fonts.googleapis.com/css2?family=Material+Icons"
      rel="stylesheet">
```

Read more in the [font portion](#) of our full developer guide.

이것이 cdn 이다. 복사해서 붙여넣자.



아무 아이콘이나 클릭할 경우 다음 창이 옆에 뜬다. 오른쪽을 좀 더 자세히 보면,



이 부분을 붙여넣으면 된다.

결과:

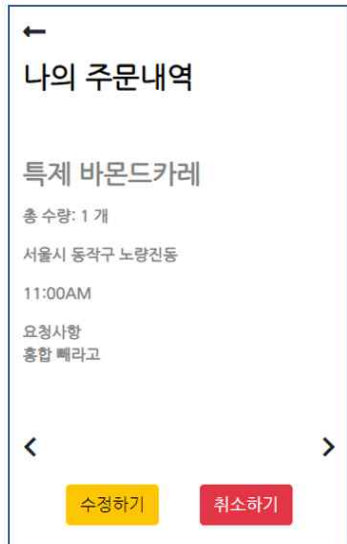


주문이 없네요

아이콘은 정확하게 말하자면 웹 폰트(web font) 이기 때문에, rem 으로 크기조절이 가능하고, color 로 색깔도 바꿀 수 있다. 말하자면, 글자처럼 다룰 수 있다.

이제 세팅은 끝났다. grid 부터 짜보자.

## grid



크게는 총 네 개의 부분으로 나눌 수 있다.

1. 아이콘과 나의 주문내역 글자
2. 실제 주문내역
3. 화살표 아이콘
4. 버튼

그리고, 밑에 있는 화살표 아이콘과 버튼은 그리드를 좌우로 다시 나눠야 할 것이다.  
myOrderMain 클래스의 내용을 지우고, 다음과 같이 작성한다.



```

<!-- 주문 내역이 있을 경우 -->
<div class="myOrderMain">
  <div>
    <div>뒤로가는화살표</div>
    <div>나의 주문내역</div>
  </div>
  <div>주문내용</div>
  <div>
    <div>왼쪽화살표</div>
    <div>오른쪽화살표</div>
  </div>
  <div>
    <div>수정하기버튼</div>
    <div>취소하기버튼</div>
  </div>
</div>

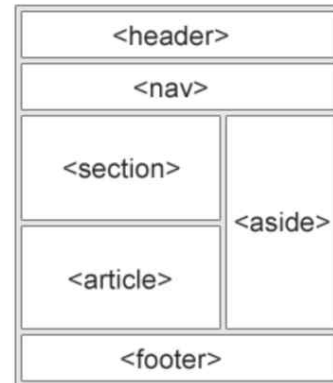
```

여러분이 실전개발할 때 하는 대표적인 실수 중 하나가, 다음과 같이 모든 것을 <div> 로 처리하고 각각의 <div> 에 class 를 부여하는 것이다.

이런 습관은 코드를 매우 읽기 힘들게 한다. <div> 를 줄이는 두 가지 해결책이 있다.

1. display: inline 이라면 span 사용
  - 지금 상황에선 해결책이 아니다. inline 요소가 보이지 않는다.
2. 시맨틱 태그(semantic tag) 사용
  - <div> 와 똑같이, block 인데 아무 역할을 하지 않는 태그들이다. 차이점은, 특별한 이름이 있다는 것.

- <header> 제목. <head> 와 헷갈리지 말자
- <nav> 메뉴 바 또는 사이드바
- <section> 특정한 영역
- <article> 본문
- <aside> 광고 등
- <main> 주요 부분
- <footer> 사이트 하단 저작권정보 등



위에 있는 것들이 시맨틱 태그다. 이름이 다른 <div> 라고 기억해두자. 쓰는 목적에 따라서 이름을 구분해놓았을 뿐이다.

우리가 쓴 코드를 시맨틱태그로 바꿔보자.

```
<!-- 주문 내역이 있을 경우 -->
<div class="myOrderMain">
  <header>
    <div>뒤로가는화살표</div>
    <div>나의 주문내역</div>
  </header>
  <article>주문내용</article>
  <section>
    <div>왼쪽화살표</div>
    <div>오른쪽화살표</div>
  </section>
  <section>
    <div>수정하기버튼</div>
    <div>취소하기버튼</div>
  </section>
</div>
```

읽기는 훨씬 수월해졌다. 클래스가 필요하면 그때그때 만들어서 쓸 예정이다. myOrderMain 안에 큰 태그는 총 4개다. 접어보면 확실히 보인다.

```

<div class="myOrderMain">
  <header> ...
</header>
<article>주문내용</article>
<section> ...
</section>
<section> ...
</section>
</div>

```

grid 를 적용해보자. 당연히, JavaScript 는 테스트하고자 하는 화면, myOrderMain 클래스를 켜둔 상태로 두고 작업해야한다.

그리고, JavaScript 에서 조심해야 할 게 하나 더 있다. querySelector() 를 사용해 display 값을 block 으로 바꾸는데, 이러면 grid 가 작동하지 않기 때문에 block 이 아니라 grid 로 바꿔주자.

css는 다음과 같이 작성한다.

```

.myOrderMain {
  grid-template-rows: 100px 1fr 50px 50px;
  min-height: 100vh;
}

.myOrderMain header .arrow_back {
  margin-top: 10px;
  margin-left: 10px;
  font-weight: bold;
}

```

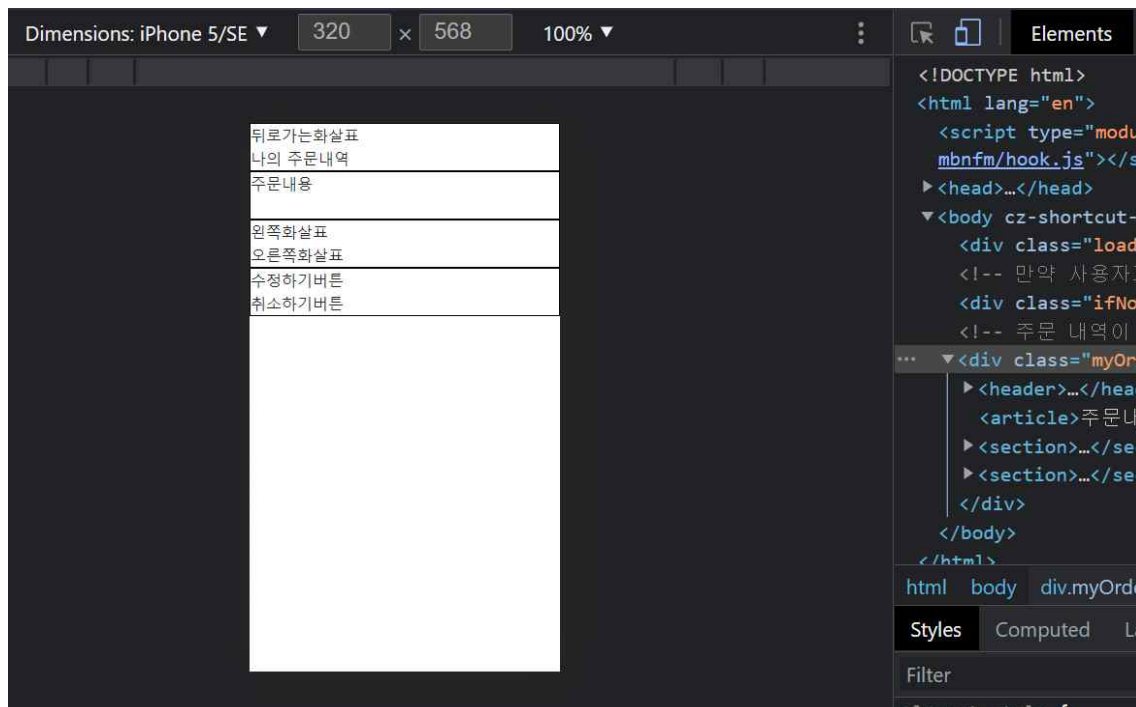
여기서, grid-template-rows 를 써봤지만 display 는 grid 로 지정 안해놓았다. 왜냐면, 이것은 맨 처음에 none 으로 지정되어 있다가, JavaScript 가 실행되면서 grid 로 바뀔 것이기 때문에, 여기서 display:grid 로 지정한다면 원하는 결과가 나오지 않기 때문이다.

선택자 한칸 띄우고 선택자를 쓰면, 그 선택자 안에 있는 선택자를 말한다.

myOrderMain 안에 있는 header

myOrderMain 안에 있는 article

myOrderMain 안에 있는 section      이 경우엔 두 개의 section 이다.



개발자도구는 모바일 화면을 지원한다. Elements 탭 옆에 모바일과 태블릿이 같이 있는 버튼을 클릭하고 원하는 모바일 기종을 선택하면 된다.

나는 주로 개발할 땐 iPhone 5/SE 로 맞추는 편인데, 여러분이 쓰고 싶은 기종이 있다면 거기에 맞춰서 개발해도 좋다.

\* 주의사항, 모든 화면에 맞춰서 개발하려는 욕심을 버려라. 미디어쿼리를 써서 하는 작업인데, 해보면 알겠지만 심각한 노가다작업이라 정신적으로 힘들 것이다. 하나의 화면만이라도 정해서 제대로 만드는 데 집중하자.

근데, 이 화면은 굉장히 마음에 안 든다. 대부분의 개발자는 이 화면 전체를 꽉 채우고 싶은 상태에서 1대 1대 1대 1의 비율로 나누고 싶을 것이다. 그럼 어떻게 할까?

다음 코드를 css myOrderMain 클래스에 추가해주면 된다.

```
min-height: 100vh;
```

이것은, 해당 태그, 즉 myOrderMain 이라는 클래스 이름을 가진 <div> 태그의 높이를 화면 전체로 지정한다는 것이다.

뒤로가는화살표 나의 주문내역
주문내용
왼쪽화살표 오른쪽화살표
수정하기버튼 취소하기버튼

지금부터 여러분들이 화면 크기를 바꿔도 똑같이, 화면 전체를 쓰게 될 것이다.  
 1대 1대 1대 1의 비율은 아니므로, 비율을 바꿔주도록 하겠다.

`grid-template-rows: 100px 1fr 50px 50px;`

이 비율은 확정된 수치가 아니기 때문에, 상황에 따라 변경해서 쓰도록 하자.

뒤로가는화살표 나의 주문내역
주문내용
왼쪽화살표 오른쪽화살표
수정하기버튼 취소하기버튼

이제 아래 두 영역의 grid 를 나눠야한다.  
html 에서 class 부터 지정해주자

```
<section class="arrows">
  <div>왼쪽화살표</div>
  <div>오른쪽화살표</div>
</section>
<section class="buttons">
  <div>수정하기버튼</div>
  <div>취소하기버튼</div>
</section>
```

그리고 css 를 다음과 같이 작성한다.

```
.arrows, .buttons {
  display: grid;
  grid-template-columns: 1fr 1fr;
}

.arrows div, .buttons div {
  border: 1px solid black;
}
```

결과:

뒤로가는화살표	
나의 주문내역	
주문내용	
왼쪽화살표	오른쪽화살표
수정하기버튼	취소하기버튼

## 기본 css 스타일링

←

나의 주문내역

특제 바몬드카레

총 수량: 1 개

서울시 동작구 노량진동

11:00AM

요청사항

홍합 빼라고

<

>

수정하기

취소하기

이제 이것을 하나하나 만들어나가면 된다. 대신, 아이콘과 버튼은 제외하고 나머지것들만 해주도록 하자.

먼저 <header> 는 나의 주문내역만 <div> 에서 <h1> 으로 바꾸고, margin 만 조금 주면 된다. font-weight 도 bold 로 주자.

html

```
<header>
  <div>뒤로가는화살표</div>
  <h1>나의 주문내역</h1>
</header>
```

CSS

```
.myOrderMain header h1 {
  font-weight: bold;
  margin-top: 10px;
  margin-left: 10px;
}
```

결과:



다음은 주문내용이다. 역시 크게 어려울 건 없는데, 일단 코드부터 보고 설명하겠다.



```

<article>
  <h2 class="menu_name">특제 바몬드카레</h2>
  <div class="quantity_container">
    <span class="quantity_tag">총 수량:</span>
    <span class="quantity">1</span>
    <span class="quantity_measures">개</span>
  </div>
  <div class="address_container">
    <span class="address">서울시 동작구 노량진동</span>
  </div>
  <div class="what_time_container">
    <span class="what_time">11:00AM</span>
  </div>
  <div class="etc_container">
    <div class="etc_tag">요청사항</div>
    <div class="etc">홍합 빼라고</div>
  </div>
</article>

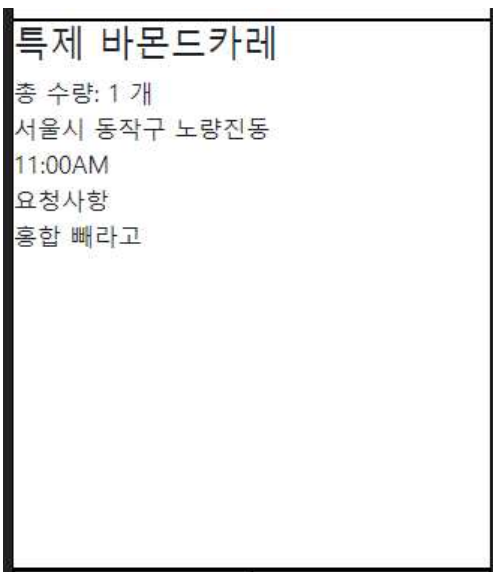
```

이 경우, <div> 를 줄이기 위해 시맨틱태그를 쓰기 힘든 경우다. 거대한 영역이 아니기 때문이다.

그러면, <div> 가 꼭 필요하지 않은 곳에 <span> 을 써줄 필요는 있다. 대표적으로, quantity\_container 클래스가 그렇다.

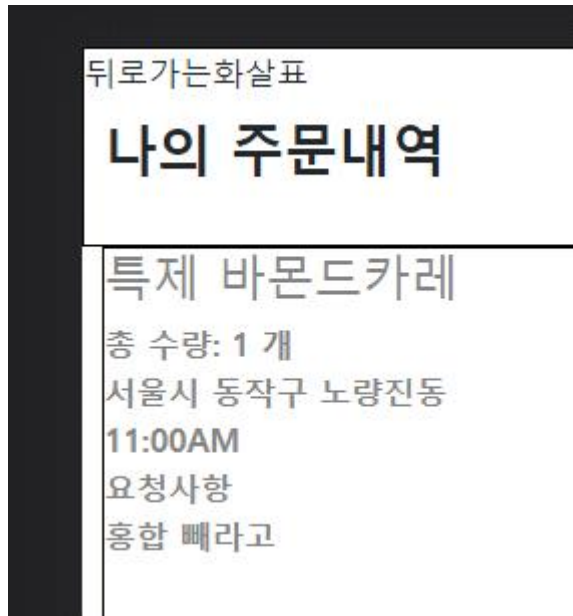
각각이 무슨 역할을 하는지 이름을 하나하나 지정해두었고, 해당 이름은 DB의 내용에 따라 다를 것이기에, 이름을 DB 의 컬럼이름과 동일하게 해주었다.

결과:



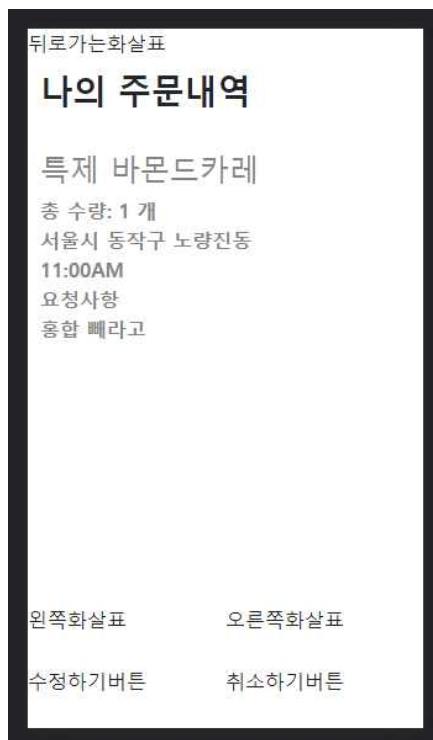
css 의 경우, 마진 주고, 글자 회색으로 바꾸고, 볼드체로 지정해주자.

결과:



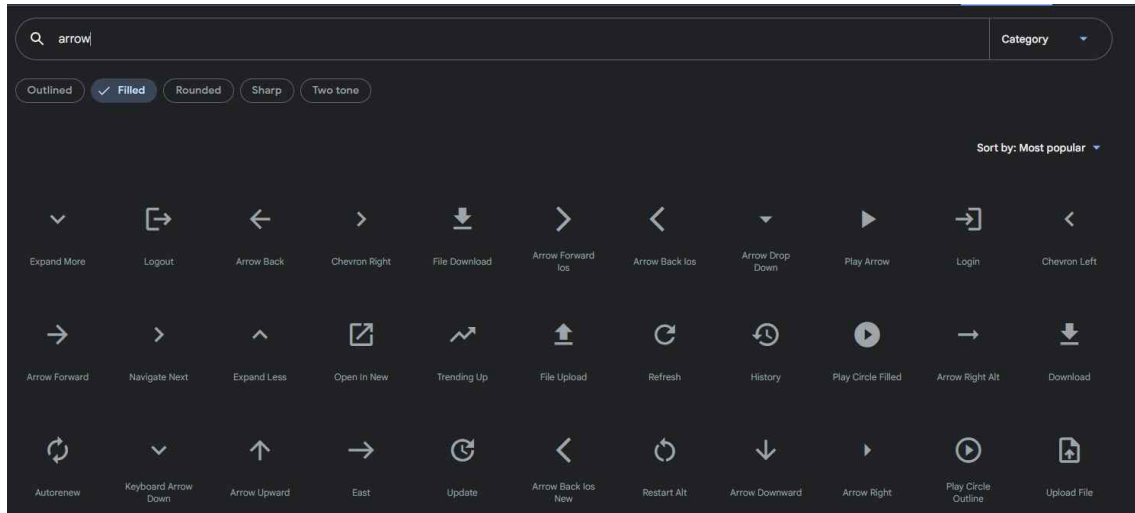
이것은 grid 지정 시 보기 위한 border 가 그대로 남아있어서 그렇다. grid 를 잘 맞추었기 때문에, 해제해주도록 하자.

결과:



## Material icons, Bootstrap

아이콘부터 해보자. 우리가 필요한 아이콘은 뒤로가기, 그리고 주문내역을 전환하는 화살표다.



화살표를 의미하는 arrow 로 쳤을 때, 다음 결과들이 나왔다. 우리가 쓰기엔 충분하다.

해당 부분의 div 를 제거하고 적용해보자.

```
<header>
  <span class="material-icons arrow_back"> arrow_back </span>
  <h1>나의 주문내역</h1>
</header>
```

위치와 크기를 변경해야 하기 때문에, arrow\_back 이라는 클래스를 하나 더 지정해주었다.

...

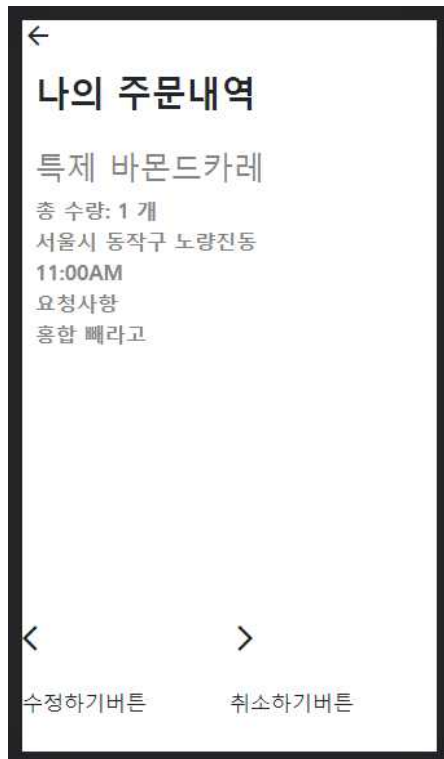
...

...

```
<section class="arrows">
  <span class="material-icons left"> arrow_back_ios </span>
  <span class="material-icons right"> arrow_forward_ios </span>
</section>
```

여기서는, left, right 라는 클래스를 하나 더 지정해주었다.

결과:

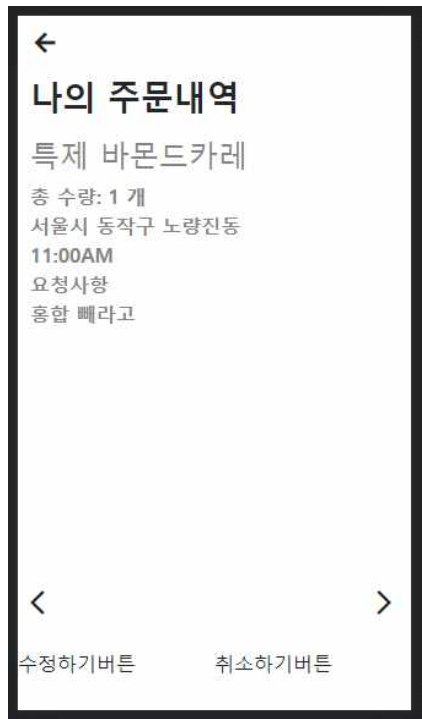


그리고 css로 위치를 적절히 조절하고 크기를 키워보자.

```
.arrow_back {  
  margin-top: 10px;  
  margin-left: 10px;  
  font-weight: bold;  
}  
  
.arrows .left {  
  margin-left: 10px;  
}  
  
.arrows .right {  
  text-align: right;  
  margin-right: 10px;  
}
```

text-align 을 통해, 오른쪽 화살표 버튼을 오른쪽으로 땡겼다.

결과:



이제, 수정하기 취소하기 버튼을 가져오기 위해 Bootstrap 으로 가서, 노란버튼 빨간버튼을 가져오자.

```
<button type="button" class="btn btn-warning fix_button">
|   수정하기
</button>
<button type="button" class="btn btn-danger delete_button">
|   삭제하기
</button>
```

둘 다 클래스를 하나씩 더 달아줬다. fix\_button, delete\_button

결과:



grid 때문에 버튼이 꽉 채워서 나오긴 하는데, 이것도 나름 맘에 듭므로 그냥 이대로 두도록 하겠다.

## 삭제 경고창: modal

삭제 버튼을 사용자가 실수로 눌러버릴수도 있다. 그래서 보통의 프로그램은 한번 더 확인하는 절차를 거친다.

삭제하기 버튼을 클릭했을 때, 모달창이 나오도록 해보자. Bootstrap 에서 제공한다. Bootstrap 사이트 검색창에 modal 이라고 쳐보자.

## Static backdrop

When backdrop is set to static, the modal will not close when clicking outside it. Click the button below to try it.

Launch static backdrop modal

```
<!-- Button trigger modal -->
<button type="button" class="btn btn-primary" data-bs-toggle="modal" data-bs-target="#staticBackdrop">
  Launch static backdrop modal
</button>

<!-- Modal -->
<div class="modal fade" id="staticBackdrop" data-bs-backdrop="static" data-bs-keyboard="false" tab
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="staticBackdropLabel">Modal title</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
      </div>
      <div class="modal-body">
        ...
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Close</button>
        <button type="button" class="btn btn-primary">Understood</button>
      </div>
    </div>
  </div>
</div>
```

나는 여러 모달 중에서, static backdrop 을 사용해 볼 것이다.

이 모달은, 바깥 부분을 클릭해도 자동으로 안 닫히는, 이 창을 종료하던지 삭제할 진행하던지 둘 중 하나를 해야만 하도록 사용자에게 요구하는 모달이다.

일단, 모달이 들어갈 자리부터 생각해보자. 모달은 특별한 존재이기 때문에, myOrderMain 안에 있으면 안된다. myOrderMain 바로 아래에 독립적으로 복사 붙여넣기 할 것이다.

그런데, 우리가 단순히 복붙하더라도 코드를 잘 읽어야하는건 당연하다. 저 예제를 자세히 보면, 모달을 컨트롤하는 것은 하나의 버튼이다.

우리에게 그 버튼은 무엇인가? 이미 만들어둔 취소 버튼이다.

예제의 버튼을 자세히 보면, 이 모달을 컨트롤하는 애트리뷰트 두 개를 넣어주어야만 모달이 작동되는데, data-bs-toggle 과 data-bs-target 이 바로 그것이다.

이걸 기존의 버튼에 넣고, 예제의 나머지 부분을 myOrderMain 아래에 넣는 것이다.

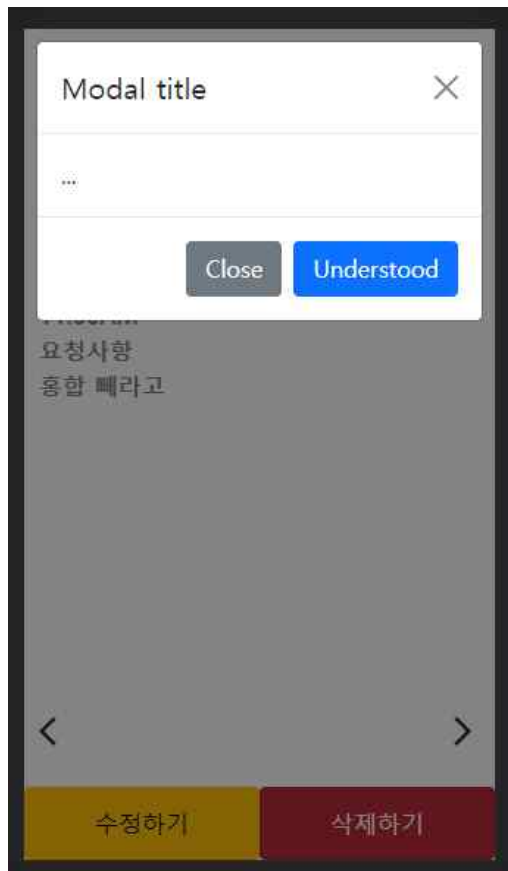
```

<button
  type="button"
  class="btn btn-danger delete_button"
  data-bs-toggle="modal"
  data-bs-target="#staticBackdrop"
>
  삭제하기
</button>
</section>
</div>
<!-- Modal -->
<div ...
</div>

```

모달 주석 밑은 일부러 접어놨다. 위의 예제에서 버튼만 제외하고 그대로 가져온 내용이 들어있다.

결과:





잘 작동됨을 확인할 수 있다.

여기서 더 진행할 수도 있지만, 나머지는 어려운 부분이 아니라서 여기까지만 HTML과 CSS를 작성하고 JavaScript로 넘어가자.

## myorder.js - onload()

이 화면이 동작하는 과정을 생각해보자.

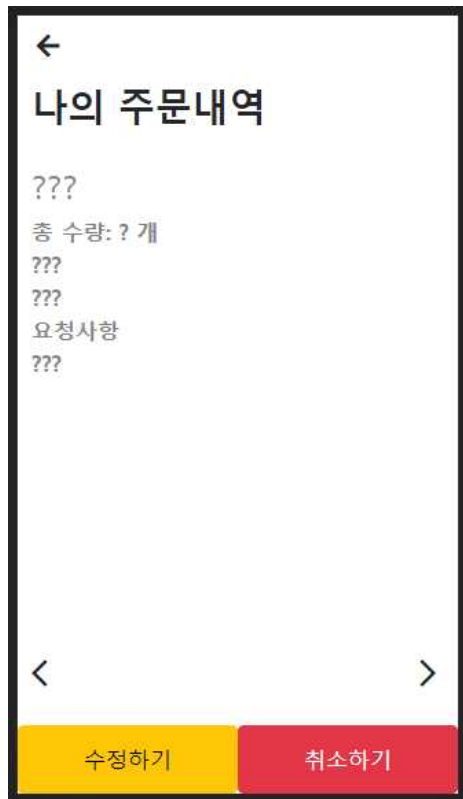
1. 서버로부터 데이터를 받아온다.
2. 만약 데이터가 있으면, 나의 주문내역을 보여준다.
3. 데이터가 없으면, 주문내역이 없다는 안내를 보여준다.
4. 아래 화살표 두 개를 누르면, 여러 개 주문내역이 있을 시 화면이 바뀐다.

이것들을 구현해 보는 것이 우리의 목표다.

먼저, 화면은 우리가 원하는대로 만들어졌으니, 가짜로 입력해놓은 데이터를 물음표 처리해놓자.

```
<article>
  <h2 class="menu_name">???</h2>
  <div class="quantity_container">
    <span class="quantity_tag">총 수량:</span>
    <span class="quantity">?</span>
    <span class="quantity_measures">개</span>
  </div>
  <div class="address_container">
    <span class="address">???</span>
  </div>
  <div class="what_time_container">
    <span class="what_time">???</span>
  </div>
  <div class="etc_container">
    <div class="etc_tag">요청사항</div>
    <div class="etc">???</div>
  </div>
</article>
```

결과:



그리고, 서버로부터 데이터를 제대로 가져오기 위해, 샘플데이터를 날려버리고  
onload 를 수정하자.

```

s > JS myorder.js > ...
1   let datas;
2
3   onload = async () => {
4       try {
5           const response = await axios.get("../php/getOrder.php");
6           datas = response.data;
7           if (datas) {
8               console.log(datas);
9               // getFirstOrder();
10              document.querySelector(".loading").style.display = "none";
11              document.querySelector(".myOrderMain").style.display = "grid";
12          } else {
13              document.querySelector(".loading").style.display = "none";
14              document.querySelector(".ifNoData").style.display = "block";
15          }
16      } catch (error) {
17          console.log(error);
18      }
19  };

```

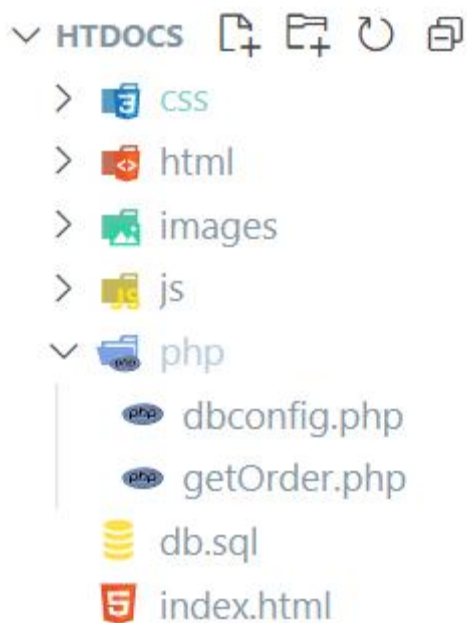
일단, 왜 `datas` 라는 변수를 `onload()` 함수 안에 두지 않고 전역변수로 사용하는걸까? 만약 `datas` 변수를 안에서 정의했다면, `onload()` 를 벗어나는순간 사용하지 못하기 때문이다.

그리고 왜 `let` 일까? line7 을 보면, 이 전역변수에 새로 “집어넣어야” 되기 때문에 `const` 로 하면 에러를 일으킨다.

`onload()` 이후에, 클릭을 하면 화면에 있는 데이터가 바뀌는 돔 조작 함수에서 이 `datas` 에 접근할 것이다.

그리고, REST API 를 `get` 방식으로 호출하며, 이름은 로컬에 위치한 `getOrder.php` 다. 빈 파일로 하나 만들어두고, `dbconfig.php` 파일도 미리 만들어두자.

그것을 `datas` 에 담아 전역변수로 활용할 수 있도록 한다.



만약, 서버에서 전달한 데이터가 있으면,

`datas` 를 `console.log()` 로 찍어내서 잘 들어오는지부터 확인한다.

`getFirstOrder()` 는 아직 만들지 않은 함수라서 에러 방지용으로 주석처리했다. 이 함수에서 `datas` 에 접근해 "0번째 인덱스" 를 화면에 붙일 것이다.

그리고 `loading` 을 끄고, `myOrderMain` 을 키는데, "grid" 로 켜야 화면이 잘 나온다.

만약, 서버로부터 아무 데이터를 받지 못했을 경우, 즉 주문이 하나도 없을 경우엔 우리가 연습했던 화면, 주문이 없네요 라는 안내페이지가 뜰 것이다.

Q. `myOrderMain` 은 아직 켜지지 않았는데, 이런식으로 짜면 `undefined` 에러가 나서 화면에 못 붙이는거 아닌가요?

A. 좋은 질문이다. `display: none` 에 대해 알아두어야 할 것이 있는데, 이것은 `css` 가 화면에서 "숨기는" 것이지, "없애는" 것이 아니다. `onload()` 가 실행되는 시점에서 이미 `html` 의 모든 태그는 화면에 붙어있는데, "안보일" 뿐이다.

이제 `dbconfig.php` 로 가서, 접속정보부터 입력해주자.

**dbconfig.php**

php >  dbconfig.php > ...

```
1  <?php
2  error_reporting(E_ALL);
3  ini_set("display_errors", 1);
4
5  header("Content-Type:application/json");
6
7  $host = 'localhost';
8  $user = 'root';
9  $pw = '111111';
10 $dbName = 'lunchbox';
11 $db = new mysqli($host, $user, $pw, $dbName);
12
13 mysqli_set_charset($db, "utf8");
```

다음과 같이 작성한다. 이 파일은 반드시 시험문제에 출제할 것이니, 무슨 뜻인지 모르는 사람은 앞의 영상으로 돌아가서 공부하길 바란다.

getOrder.php

php >  getOrder.php > ...

```
1  <?php
2  require_once("dbconfig.php");
3  $sql = "
4      SELECT
5          order_id,
6          menu_name,
7          quantity,
8          address,
9          what_time,
10         etc
11     FROM ordered
12 ";
13
14 $data = array();
15
16 $res = $db->query($sql);
17
18 for ($i = 0 ; $i < $res->num_rows ; $i++) {
19     $row = $res->fetch_array(MYSQLI_ASSOC);
20     array_push($data, $row);
21 }
22
23 if ($data != null) {
24     echo json_encode($data, JSON_UNESCAPED_UNICODE|JSON_NUMERIC_CHECK);
25 } else { // 사용자 주문 내역이 없을 경우
26     echo json_encode(false, JSON_UNESCAPED_UNICODE|JSON_NUMERIC_CHECK);
27 }
28
29 mysqli_close($db);
```

코드는 지난 번에 설명한것에서 크게 벗어나지 않는다.

dbconfig.php импорт하고

sql 문 미리 넣어두고

빈 배열 변수 \$data 만들고

\$res 로 sql 실행결과를 받고

여러줄이기 때문에 for 문 써주고

데이터가 있는지 아닌지에 따라서 보내는 결과가 다르고

db 닫아준다.

결과:

```

▼ (3) [{...}, {...}, {...}] ⓘ
  ▶ 0: {order_id: 1, menu_name: '특제 바몬드 카레', quantity: 1, etc: '해물 스파게티'}
  ▶ 1: {order_id: 2, menu_name: '샌드위치', quantity: 1, etc: '해물 스파게티'}
  ▶ 2: {order_id: 3, menu_name: '해물 스파게티', quantity: 1, etc: '해물 스파게티'}
    length: 3
  ▶ [[Prototype]]: Array(0)

```

데이터가 성공적으로 datas 라는, 우리가 만든 배열에 들어왔음을 확인할 수 있다. 이제 우리가 할 일은, 맨 처음 인덱스를 화면에 붙이는 것이다. JavaScript의 `getFirstOrder()` 함수를 작성해보자.

## getFirstOrder()

```

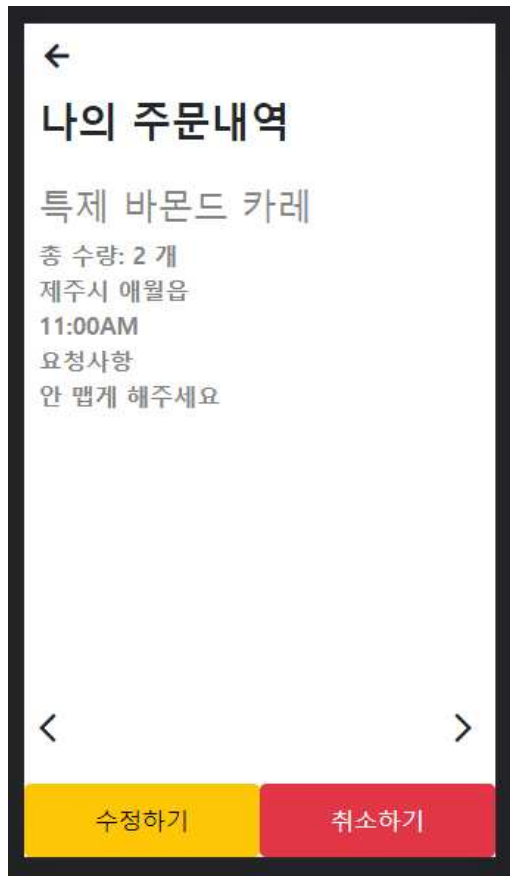
const getFirstOrder = () => {
  document.querySelector(".menu_name").textContent = datas[0].menu_name;
  document.querySelector(".quantity").textContent = datas[0].quantity;
  document.querySelector(".address").textContent = datas[0].address;
  document.querySelector(".what_time").textContent = datas[0].what_time;
  document.querySelector(".etc").textContent = datas[0].etc;
};

```

태그 안의 텍스트에 접근할 땐, `textContent` 로 접근한다. 왜 0번인가? 그게 첫번째 데이터이기 때문이다.

결과:





이제 마지막작업, 버튼을 눌렀을 때 datas 의 다른 인덱스에 접근하여 화면을 바꾸는 연습을 해보자.

## changeIndex()

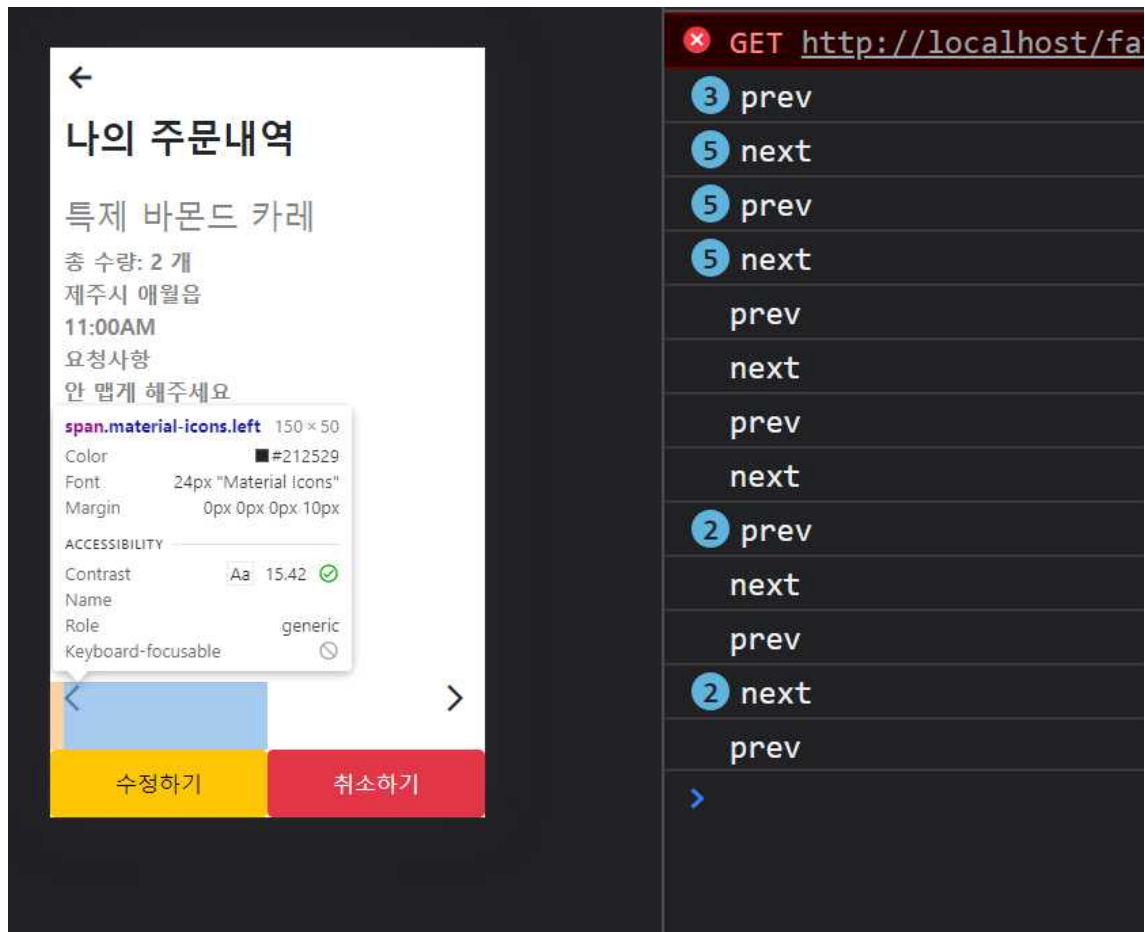
먼저, 이벤트핸들러부터 달자. 함수 이름은 changeIndex() 라고 하고, 파라미터를 넣어주도록 하겠다.

```
<section class="arrows">
  <span class="material-icons left" onclick="changeIndex('prev')"> arrow_back_ios </span>
  <span class="material-icons right" onclick="changeIndex('next')"> arrow_forward_ios </span>
</section>
```

두 버튼은 각각 prev, next 라는 파라미터를 클릭할때마다 넘긴다.  
우선 함수로 콘솔로그부터 찍어보자.

```
const changeIndex = (mode) => {
  console.log(mode);
};
```

결과:



사용자 입장에서 불편한 점이 있는데, 정확하게 화살표를 눌러야 작동하는것이 아니라 저 파란 영역 안에, 그러니깐 grid 로 잡혀있는 영역을 클릭하면 함수가 작동된다.

이것은 여러분들 수준이면 충분히 해결해볼 수 있는 현상이다. 해결은 여러분들에게 남겨두도록 하고, 함수부터 작성해보도록 하자.

```

const changeIndex = (mode) => {
  if (mode === "prev") {
    if (index !== 0) {
      index = index - 1;
    } else {
      index = datas.length - 1;
    }
  } else {
    if (index !== datas.length - 1) {
      index = index + 1;
    } else {
      index = 0;
    }
  }
  console.log(index);
  // changeDOM();
};

```

인덱스라는 "전역변수" 를 쓸 것이다. 우린 만든 적 없으니 let 으로 하나 만들어두자. 이 인덱스는 계속 변할 것이기 때문에 const 로 지정할 순 없다. 그리고 index 는 반드시 0으로 초기화해줘야 한다.

왜? 화면에 붙은 맨 처음 인덱스는 0이기 때문이다.

만약, 왼쪽 버튼(prev) 를 클릭하면 어떤 식으로 작동해야 하는가?

현재 인덱스가 0이 아니라면, 인덱스를 하나 빼면 될 것이다.

만약, 현재 인덱스가 0이라면 어떠한가?

이 경우엔, datas 배열의 전체 크기에서 하나를 빼야한다. 맨 마지막 데이터를 보여줄 것이기 때문이다.

그러니깐, 끝과 끝으로 가면 반대쪽 끝으로 돌아가게 만들 것이다.

데이터가 3개 있다면?

0 부터 시작하므로 클릭했을 땐 2가 될 것이다.

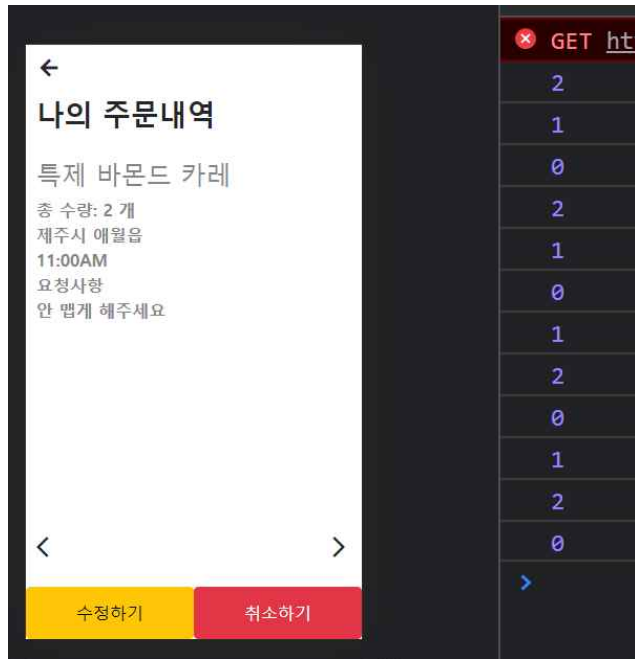
2 인 경우에 다시 클릭하면 1이 되고

1 인 경우에 다시 클릭하면 0이 된다.

0 인 경우에 다시 클릭하면, 다시 맨 뒤로 가야되니깐 2다.

오른쪽 버튼을 눌렀을때에도 마찬가지로 코딩했다.  
그리고 콘솔로그로 찍어서 테스트할 것이고,  
changeDOM() 이라는 함수를 만들 것이기에 주석처리해둔다.

결과:



## changeDOM()

진짜 마지막. 인덱스를 기준으로 화면을 바꿔보자.

changeDOM() 은 changeIndex() 의 가장 마지막에 실행된다.

```
const changeDOM = () => {  
  document.querySelector(".menu_name").textContent = datas[index].menu_name;  
  document.querySelector(".quantity").textContent = datas[index].quantity;  
  document.querySelector(".address").textContent = datas[index].address;  
  document.querySelector(".what_time").textContent = datas[index].what_time;  
  document.querySelector(".etc").textContent = datas[index].etc;  
};
```

datas 에 현재 인덱스로 바꿔준다.

결과는 확인해보면 될 것이다.

이것으로 실전개발 강좌를 마치도록 하겠다.