

이미 공지한 대로,

이 수업은 이러닝에 업로드한 SQL 강의를 들었어야 진행이 된다
듣지 않은 학생은, 이 수업 녹화본 역시 자료실에 업로드할 예정이므로,
SQL 강의를 들은 후에 본 강의를 학습하라.

먼저 이러닝 자료실에 가서, 오늘 할 수업코드를 받아 bitnami htdocs 에 담는다.

당연히, 압축 풀어서 모든 파일을 htdocs 로 옮겨야한다.

21-11-03 ?요일 수업 코드

이 폴더 있는 상태로 두지 마라 ㅋㅋ

≡ 웹프로그래밍실습(039분반) > 웹프로그래밍실습(039분반)

2021년 2학기

홈

수업 계획서

공지

강의자료실

문의게시판

총 27 개

⋮



21-11-03 수요일 수업 코드

파일

가장 먼저, test 라는 이름의 db를 만든다.

경로는 알 것이다. (맥은 모른다.)

Path

```
C:\Bitnami\wampstack-8.0.9-0\mariadb\bin
```

여기까지 cd 명령어를 써서 간 다음,

```
조교 행님@DESKTOP-4S6IR37 bin
→ .\mysql.exe -u root -p
Enter password: *****
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 8
Server version: 10.4.20-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)]> |
```

다음 명령어로 db에 접속한다.

test 라는 이름의 db를 만들 것이므로, 일단 있는지부터 확인해보자.

```
MariaDB [(none)]> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| lunchbox |
| mysql |
| performance_schema |
| test |
+-----+
5 rows in set (0.001 sec)

MariaDB [(none)]> |
```

있으면 이 db를 날려버리도록 하자.

날리기 싫으면 다른 이름으로 지어도 된다.

```
MariaDB [(none)]> DROP DATABASE test;
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> |
```

그리고, 다음 명령어로 test 라는 이름의 db를 만든다.

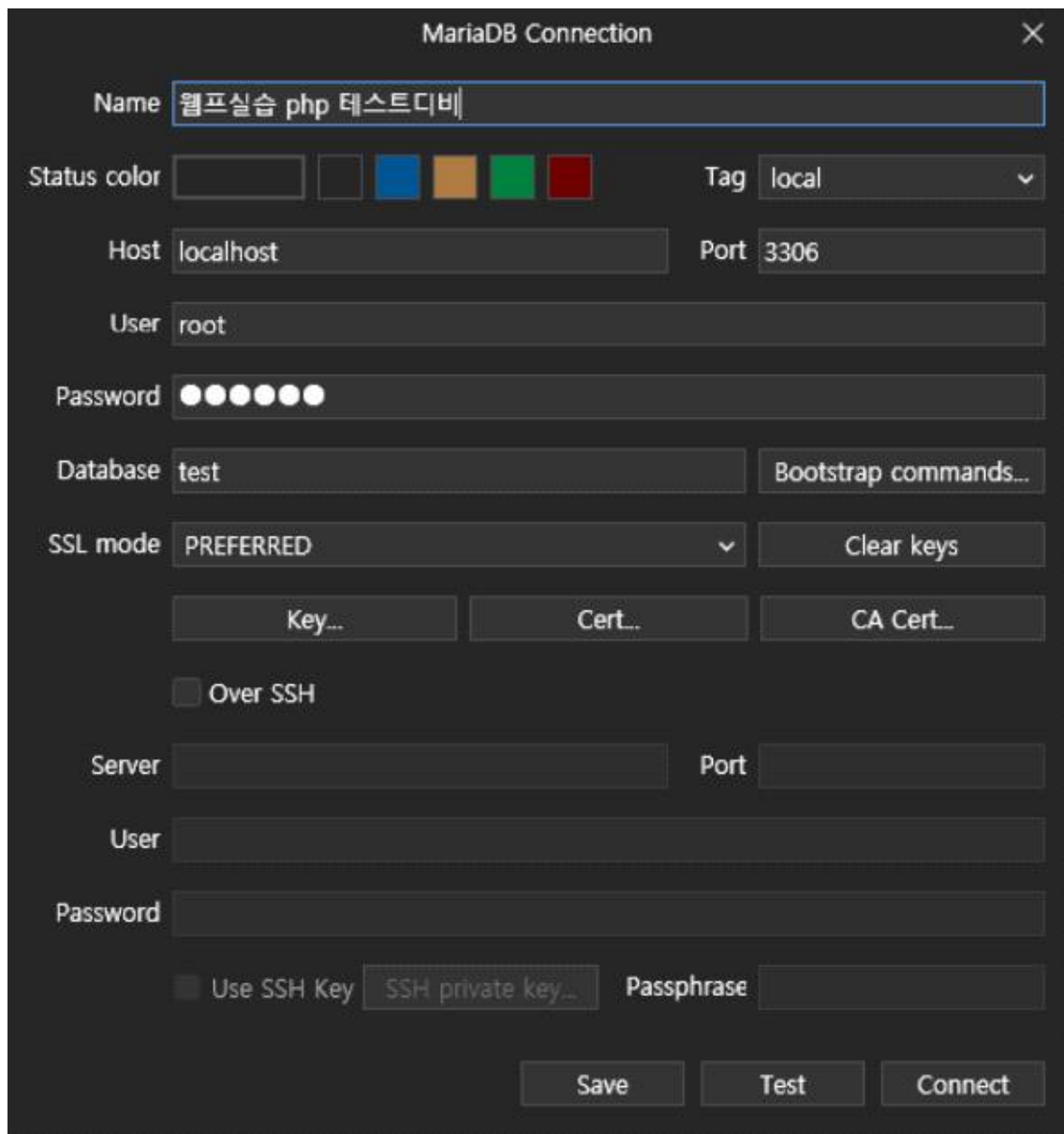
```
MariaDB [(none)]> CREATE DATABASE test;
Query OK, 1 row affected (0.001 sec)

MariaDB [(none)]> |
```

이제 TablePlus 에서 작업할 것이다.



중간 아래 Create a new connection 클릭

A screenshot of the MariaDB Connection dialog box. The dialog has a title bar 'MariaDB Connection' with a close button. It contains several input fields and buttons. The 'Name' field is filled with '웹프실습 php 테스트디베'. The 'Status color' field shows a row of six colored squares (white, blue, orange, green, red). The 'Tag' field is a dropdown menu set to 'local'. The 'Host' field is 'localhost' and the 'Port' field is '3306'. The 'User' field is 'root'. The 'Password' field is masked with six white dots. The 'Database' field is 'test' and there is a 'Bootstrap commands...' button next to it. The 'SSL mode' is a dropdown set to 'PREFERRED' with a 'Clear keys' button next to it. Below these are three buttons: 'Key...', 'Cert...', and 'CA Cert...'. There is a checkbox for 'Over SSH'. At the bottom, there are fields for 'Server', 'Port', 'User', and 'Password'. Below these is a checkbox for 'Use SSH Key' with a button 'SSH private key...' and a 'Passphrase' field. At the very bottom are three buttons: 'Save', 'Test', and 'Connect'.

다음 정보를 입력한다.

Name: 원하는 이름. 한국어도 가능

Host: localhost

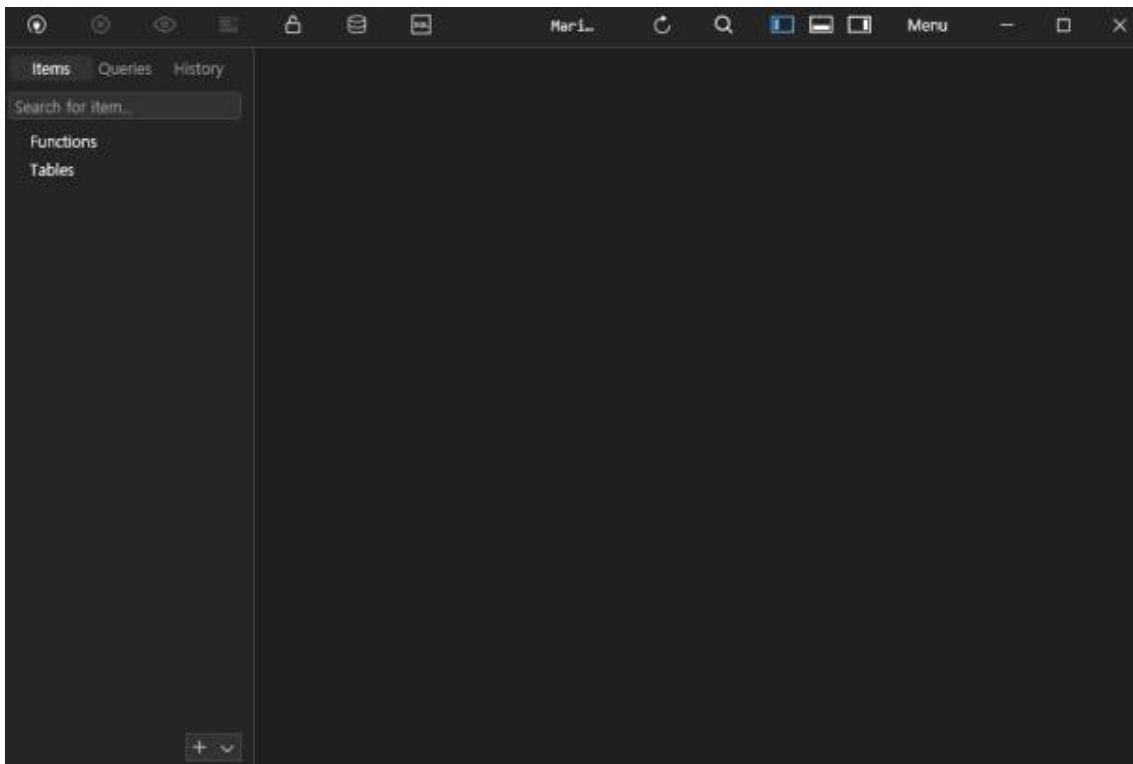
Port: 3306 - MySQL과 MariaDB의 기본 포트

User: root

Password: Bitnami 설치 시 초기 설정 비밀번호. 나는 111111 (1 여섯개) 였다.

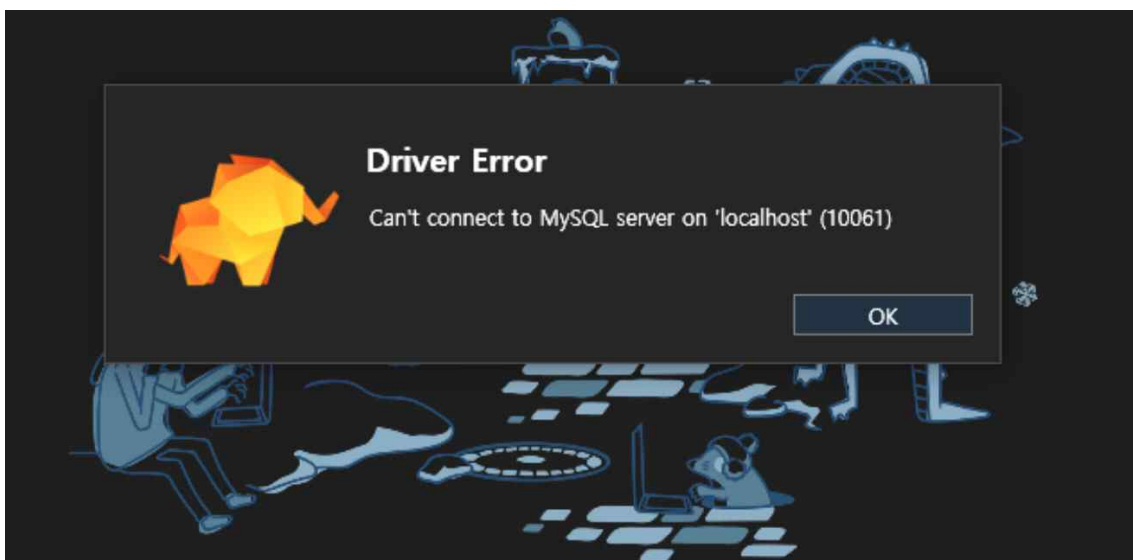
Database: test 우리가 test 라는 이름의 db를 만들었기에 여기에 접속

나머지 정보는 필요없고, 아래의 Connect 버튼을 누르자.

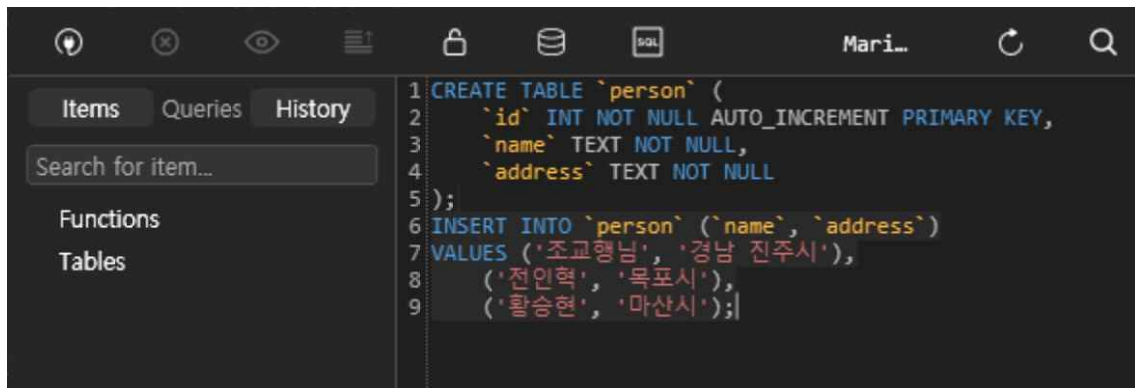


성공하면 다음과 같이, 로딩 후 빈 화면이 뜬다.

연결이 실패할 경우, 다음 형태의 에러메세지가 뜬다.



그리고 이러닝에서 받은 코드 중, db.sql 파일 안에 있는 코드를 그대로 복사해, TablePlus 의 SQL 창에 붙여넣는다.

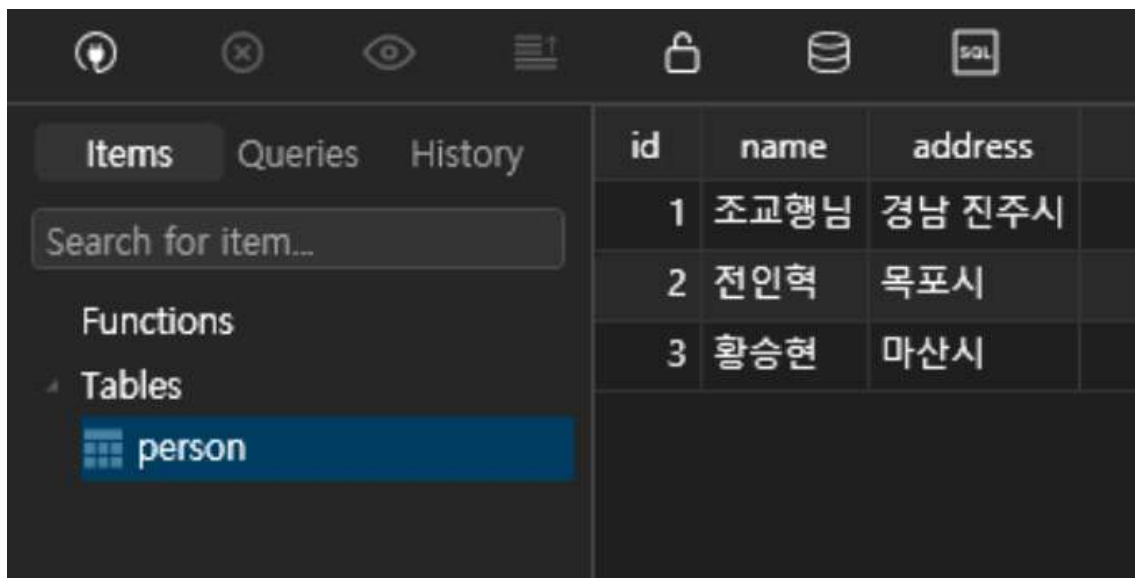


The screenshot shows a SQL IDE interface with a dark theme. On the left, there's a sidebar with tabs for 'Items', 'Queries', and 'History'. Below these is a search bar labeled 'Search for item...'. Further down are links for 'Functions' and 'Tables'. The main area displays a SQL query with line numbers 1 through 9. The query creates a table named 'person' with columns 'id' (integer, not null, auto-increment, primary key), 'name' (text, not null), and 'address' (text, not null). It then inserts three rows of data into the table.

```
1 CREATE TABLE `person` (  
2   `id` INT NOT NULL AUTO_INCREMENT PRIMARY KEY,  
3   `name` TEXT NOT NULL,  
4   `address` TEXT NOT NULL  
5 );  
6 INSERT INTO `person` (`name`, `address`)  
7 VALUES ('조교행님', '경남 진주시'),  
8         ('전인혁', '목포시'),  
9         ('황승현', '마산시');
```

이걸 “한 문단씩” 실행하자. (ctrl + enter)

결과:

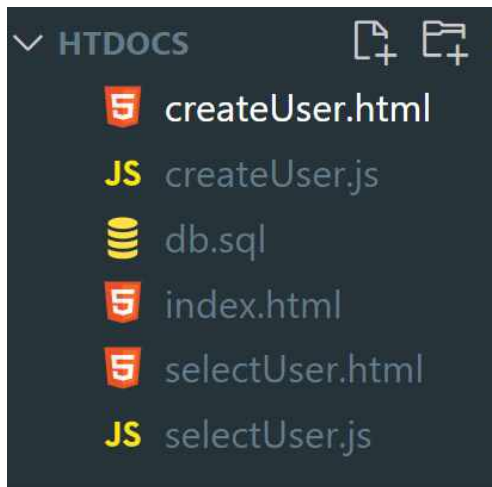


The screenshot shows the same SQL IDE interface, but now the 'Tables' tab in the sidebar is selected, and the 'person' table is highlighted. The main area displays a table with the results of the query. The table has four columns: 'id', 'name', 'address', and an empty column. The data is as follows:

id	name	address	
1	조교행님	경남 진주시	
2	전인혁	목포시	
3	황승현	마산시	

여기까지 했으면, 우리의 db가 준비된 것이다.

이러닝에서 받은 파일 목록은 다음과 같다.



세 개의 html 파일, 두 개의 js 파일, 한 개의 sql 파일이며, css 는 없다.
그 중, 한 개의 sql 파일은 db 만드는 데 이미 사용했다.

크롬에서 localhost 로 접속하면 다음 화면이 뜰 것이다.

index.html

php rest api 연습

[유저 생성](#)

[유저 조회](#)

createUser.html

유저 생성

이름
주소
유저 생성하기

selectUser.html

유저 조회

모든 유저의 정보를 가져옵니다.

모든 유저 가져오기

이름과 일치하는 유저의 주소를 가져옵니다.

이름

이름과 일치하는 유저 주소 가져오기

우리가 JavaScript와 PHP 를 코딩하지 않았기 때문에 아무것도 작동 안한다.
가장 먼저, 유저를 생성하는 createUser.html 의 JavaScript 를 담당하는
createUser.js 파일을 보면, 아무것도 없을 것이다.

selectUser.js

이 파일엔 두 개의 함수를 만들 것이다.

getAllUser() 모든 유저를 가져옴

getAddressByID() ID와 일치하는 유저의 주소를 가져옴

둘 다 onclick 이벤트핸들러에 의해 실행될 것이다.

selectUser.html

```
<body>
  <h1>유저 조회</h1>
  <h2>모든 유저의 정보를 가져옵니다.</h2>
  <button onclick="getAllUser()">모든 유저 가져오기</button>
  <h1>이름과 일치하는 유저의 주소를 가져옵니다.</h1>
  <input class="nameInput" type="text" placeholder="이름">
  <button onclick="getAddressByID()">이름과 일치하는 유저 주소 가져오기</button>
</body>
```

두 개의 <button> 태그만 보자. onclick 이벤트핸들러가 달려있다. 클릭을 하면 함수
getAllUser() 와 함수 getAddressByID() 가 실행될 것이다.

각각의 함수를 미리 만들어두고, 콘솔로그만 넣어두자.

selectUser.js

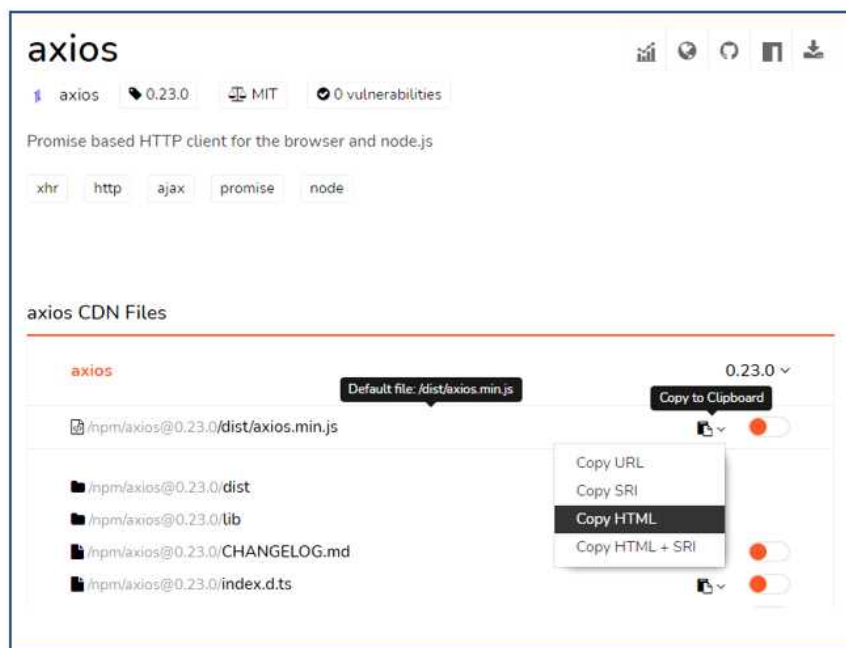
```
const getAllUser = () => {  
  console.log("getAllUser 함수 실행");  
};  
  
const getAddressByID = () => {  
  console.log("getAddressByID 함수 실행");  
};
```

우린 이 함수들에서, 서버코드인 PHP 를 실행할 것이다.
우리가 만들 PHP 파일들을 일단 API 라고 알아두자.
서버의 API를 실행하기위해 JavaScript 에선 fetch() 함수를 제공한다.
그러나, 현재 업계에서 표준은 axios 라는 패키지이다.
패키지란? 개발자 선배들이 “미리 만들어놓은” 코드이다.
개발할 때 우리가 모든 걸 다 만들어 쓰지는 않는다.
선배 개발자들이 잘 만들어놓은 코드가 있으면, 가져다 쓰는 것도 능력이다.
패키지는 라이브러리, API 와 비슷한 뜻인데, 세 가지 뜻을 정확하게 아는것보다
중요한건, 세 가지 모두 선배 개발자들이 미리 만들어둔 코드라는 것이다.

axios 를 사용해보기위해, jsdelivr 라는 사이트로 가보자.

<https://www.jsdelivr.com/>

이 사이트에선, 각종 패키지들의 CDN 을 제공해준다.
CDN 은, 패키지가 저장된 링크를 말한다.
보통 JavaScript 개발자들은 npm 이나 yarn 을 통해 패키지들을 개발자의 컴퓨터에
직접 설치해서 사용하는데, 그것까지 다루긴 이 수업이 너무 어려워진다.
CDN은 패키지의 코드가 저장된 서버의 링크를 말한다. 직접 설치하지 않고, 링크만
HTML 에 넣어서 가져온다.



jsdelivr 검색창에 axios 를 입력한 후, CDN Files 중에 가장 위에 있는 것을 복사하는 버튼을 클릭하면, Copy HTML 이라고 보일 것이다. 클릭하면 CDN 링크가 복사된다.

그걸 어디에 넣으면 될까? selectUser.html 의 <head> 태그 안에 다음과 같이 넣는다. 나는 axios 라고 주석을 하나 달아두고 아래위로 한칸씩 띄웠다.

```
<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">

  <!-- axios -->
  <script src="https://cdn.jsdelivr.net/npm/axios@0.23.0/dist/axios.min.js"></script>

  <script src="./selectUser.js"></script>
  <title>유저 조회</title>
</head>
```

그리고 다시 selectUser.js 파일로 돌아와서, 다음과 같이 작성한다.

```
JS selectUser.js ●
JS selectUser.js > [?] getAddressByID
1  const getAllUser = () => {
2    try {
3
4    } catch (error) {
5      console.log(error);
6    }
7  };
```

try catch 는 통신 에러 발생 시 실행되는 에러처리 구문이다.

접근하고자하는 서버의 트래픽이 과도하게 발생하거나, 셧다운되거나, 원하는 경로에 파일이 없거나 하는 문제가 발생하면, catch 에서 에러를 받아 적절한 일을 한다.

예를 들면, 서버와 통신 실패했을 때 경고창을 뜨게 만들 수 있다.

사이트에 문제가 있습니다. 잠시 후 다시 시도해주세요

우리가 실행시키고자 하는 구문은 try 에 작성한다.

다음과 같이 쪽 입력해보자.

```
JS selectUser.js X
JS selectUser.js > ...
1  const getAllUser = () => {
2    try {
3      const response = axios.get("http://203.255.3.246:9000/getAllUser.php");
4      if (response.data) {
5        console.log(response.data);
6      }
7    } catch (error) {
8      console.log(error);
9    }
10 };
```

특히, http 로 시작하는 주소를 잘 적어야한다.

http://203.255.3.246:9000/getAllUser.php

이건 조교가 미리 만들어둔 API 주소이다.

여기서, 당장 필요한 지식만 간단하게 설명하자면,

axios.get() 을 사용해 getAllUser.php 파일에 접근한다.

이 php 파일은 어떤 DB의 모든 유저데이터를 가져오는 일을 하며,
response 객체를 리턴한다.

response 객체 안에 data 가 바로 getAllUser.php 가 한 일의 결과다.
그걸 console.log() 로 찍어보는 것이다.

그러나, 모든 유저 가져오기 버튼을 눌러보면 아무것도 콘솔에 찍히지 않는것을 확인할 수 있다.

함수 이름을 잘못 썼나? 그렇 리 없다. 콘솔로그로 테스트했다.

심지어, 에러 로그도 뜨지 않는다.

당연히 아무것도 안 찍힐수밖에 없다. 여러분은 비동기의 개념을 모르기 때문이다.

비동기(Asynchronous)

JavaScript 개발을 하는 데 이 개념을 모르면 엄청나게 고생한다.

```
const getAllUser = () => {  
  try {  
    const response = axios.get("http://203.255.3.246:9000/getAllUser.php");  
    if (response.data) {  
      console.log(response.data);  
    }  
  } catch (error) {  
    console.log(error);  
  }  
};
```

파란색으로 박스 표시한 부분을 잘 보자.

이 부분이 실행되기 전에 바로 아래에 있는 if 가 실행된 것이다.

왜냐, JavaScript 는 건방지게도, 늦어질 것 같으면 다음 행을 먼저 실행시키기 때문이다.

```
if (response.data) {  
  console.log(response.data);  
}  
const response = axios.get("http://203.255.3.246:9000/getAllUser.php");
```

실제로 실행된 순서는 다음과 같다.

애초에 response.data 안에 아무것도 없으니깐 실행이 안 된 것이다.

그럼 undefined 에러가 터야하지 않나? 그건 또 아니다.

response를 const 변수로 선언하긴 했기 때문이다.

대체 왜 JavaScript 는 이런 끔찍한, 비동기방식을 쓸까?

만약 엄청나게 느린 서버가 있다고 가정하자.

비동기가 없다면, 서버로부터 데이터가 올때까지 계속 기다려야만 한다.
여러분은 배달의민족 앱에서 떡볶이 주문버튼을 클릭했는데 20분을 기다리고싶은가?
JavaScript 는 웹의 언어다. 따라서 이런 사태를 방지하기 위해, 서버와 통신하는 코드가 있을 땐 무조건 다음 코드를 먼저 실행시킨다.
JavaScript 개발자는 비동기 현상을 알고 있어야하며, 이를 제어할 수 있는 능력도 갖추어야 한다.
다음과 같은 방법들이 있다.

해결책1: 콜백함수

2015년 이전의 유일한 해결책이었다. 콜백이 무엇인가? 파라미터로 함수를 넣어서, 파라미터로 들어와 실행된 함수가 “끝나야만” 다음 로직을 진행하는 것이다.
이 경우, 다음과 같은 콜백 지옥(callback hell) 에 빠질 수 있다.

```
step1(function (value1) {  
  step2(function (value2) {  
    step3(function (value3) {  
      step4(function (value4) {  
        step5(function (value5) {  
          step6(function (value6) {  
            // Do something with value6  
          });  
        });  
      });  
    });  
  });  
});
```

콜백 지옥은 JavaScript 개발자들이 가장 싫어하는 현상이다. 여러분은 위와 같은 코딩을 하고싶은가?

해결책2: Promise 객체

2015 년 이후 콜백에 지친 JavaScript 개발자들을 덜 피로하게 만든 해결책이다.
여러분이 JavaScript 로 밥먹고 살려면 반드시 배워보길 바란다.
Promise 객체의 pending, fulfilled, rejected 를 사용해 비동기를 제어한다.
그러나, 우리 초보자 입장에선 여전히 어렵다.

해결책3: async await

2015년 ES6 당시엔 존재하지 않는 기술이었는데, 몇 년 지나고나서 나온

해결책이다.

일단, 매우 직관적이고 쉽다. 비동기를 동기적으로 제어하는 기술이다.

async 로 비동기 로직이 포함되어있음을 알리고,

await 로 끝날때까지 기다리라고 지시한다.

우리가 썼던 코드에서 이 두개만 추가해주면 된다.

async 비동기가 포함되어있음을 알림

```
const getAllUser = async () => {  
  try {  
    const response = await axios.get("http://203.255.3.246:9000/getAllUser.php");  
    if (response.data) {  
      console.log(response.data);  
    }  
  } catch (error) {  
    console.log(error);  
  }  
};
```

await 이게 끝나기전까진 다음줄로 넘어가지 마라!

결과:

```
▼ (3) [{...}, {...}, {...}]  
  ▶ 0: {id: 1, name: '조교행님', address: '경남 진주시'}  
  ▶ 1: {id: 2, name: '전인혁', address: '목포시'}  
  ▶ 2: {id: 3, name: '황승현', address: '마산시'}  
    length: 3  
  ▶ [[Prototype]]: Array(0)
```

async 와 await 를 추가해줬을 뿐인데, 드디어 데이터가 들어온 것을 확인할 수 있다.

여러분이 서버와 통신할 일이 있을 땐, 반드시 비동기를 고려해서 async await 를 활용하길 바란다.

본격적으로 우리가 할 일은 다음과 같다.

우리는 203.255.3.246 서버의, 9000 번 포트의, 조교가 직접 만든 getAllUser.php 를 사용했다.

이제 직접 PHP API 를 만들어볼 것이다.

왜 PHP를 쓰는가? JavaScript 가 가지고 있는 한계 때문이다.

1. DB CRUD 불가능

사실은 틀린 말이다. Node.js 등장 이후 가능하다.

그러나, DB CRUD 만 하기엔 Node.js 는 지나치게 어렵다.

해본 사람은 알겠지만, 아무것도 없는 산골짜기에 집을 짓는 느낌이다.

2. 페이지를 벗어나면 데이터 보존이 안 됨

로그인 페이지(login.html) 과 메인페이지 (main.html) 이 있다고 가정하자.

login.html 에서 로그인 성공해 main.html 로 넘어가는 순간,
login.html에서의 모든 데이터는 초기화되므로 main.html 로 데이터를 넘길
수가 없다.

이것을 해결하려면 세션(session) 변수를 브라우저가 아닌 서버에 저장해야
하는데, 브라우저 안에서 기본적으로 사용하는 JavaScript 로는 할 수가
없다.

여기서 PHP 가 나온다. 개인적으로, 2021년 웹 입문자라면 PHP 대신 Python 을
쓰는 Django 를 배우길 추천한다. DB CRUD 를 배우기엔 가장 쉽고, 가장 좋다.
PHP는 1995년에 등장해 인터넷 부흥기를 이끈 언어다. 무려 26년 역사를 가지고
있는데, 이 정도면 웹 시장에선 영감님 기술이다.

그러나, Django 와 비교해봤을 땐 DB CRUD 가 크게 어렵진 않다. 특히, 현업에서
쓰이는 JSP Spring 과 비교해보면 엄청나게 쉬운 언어다.

우린 PHP 를 사용해 DB CRUD 를 하는 REST API 를 만들 것이다. 이게 대체 무슨
소리인가? REST 는 뭐고 API 는 또 뭐가?

REST

통신의 규칙 중 하나다. 누구도 부정할 수 없는 업계 표준으로 자리잡았는데,
여러분이 어떤 회사를 가든지 REST 라는 “규칙”을 지킨다.

2000년 로이 필딩의 박사학위 논문에 등장해 지금까지 쓰이고 있으며, 기존 SOAP
라는 “규칙” 과 경쟁하다가 완전히 시장의 표준이 되었다.

REST 는 특정한 기술이 아니다. 통신을 하는 “규칙” 이다.

정확하게는 HTTP 를 사용해 데이터를 주고 받는 “규칙” 이다.

대한민국 KOREA
신원 및 배달안내
☎ 1588-1300
우체국 택배
www.ePOST.kr
우편번호가
2015년 8월1일부터 바뀝니다.
11110-11110 > 01311817

보내신 분	성명				내 용 물	
	전화				* 단상소포로 접수한 경우에는 표기하신 내용을 가역의 범위에서(300만원 이내) 실 손해액을 배상받을 수 있습니다.(우편법 제38조)	
	주소	□□□□□			<input type="checkbox"/> 착불소포	원
받으신 분	성명				<input type="checkbox"/> 안심(보험)	만원
	전화				<input type="checkbox"/> 파손주의	원
	주소	□□□□□			<input type="checkbox"/> 취급주의	만원

- 고객안내사항 -
* 부패, 변질, 파손(훼손)이 우려되는 내용물은 '특수포장' 할 경우에 한하여 접수 가능합니다. 책임 원인이 발송인에게 있는 경우 손해배상에서 제외됩니다. (우편법 제39,40조)
* 냉동·냉장물품은 보냉재(식음료를 또는 아이스팩 등)를 넣어 포장하되, ① 육류는 별도 진공포장 ② 부패, 변질가능 우편물은 스티로폼 박스 사용

증지라벨 붙이는 곳
(우체국사용)

개인정보 유출방지를 위하여 성명, 전화번호, 주소를 제거 바랍니다.

우체국 송장처럼, 정해진 하나의 양식(form) 이다.

보내는 분 주소는 받는분 성명에 쓰면 당연히 안된다.

착불로 보내고 싶으면 체크박스에 체크해야한다.

API 에서 데이터를 가져오고 싶으면 GET 방식, API 에 데이터를 보내고 싶으면 POST 방식을 사용한다.

근데 우리 REST API 를 만든다고 했다. API 는 또 무엇인가?

API

라이브러리, 패키지와 비슷한 뜻인데, 쉽게 생각하면 특정 일을 하는 코드다.

ctrl + c 를 누르면 복사가 되는 건 누구나 아는 사실이다.

그런데 이 일을 하는 코드가 없다면 ctrl + c 를 눌러도 당연히 아무 일도 안 일어날 것이다.

즉, ctrl + c 를 “입력” 받아서 클립보드에 복사하는 일을 하는 코드가 “호출(call)” 된다는 것이다.

우리가 만들어볼 현실적인 예로는,

getAllUserData 라는 PHP 파일을 “호출”하면,

DB에 접근해 모든 유저의 데이터를 가져올 수 있다.

REST API

REST 통신 규칙을 따르는 API 이다.

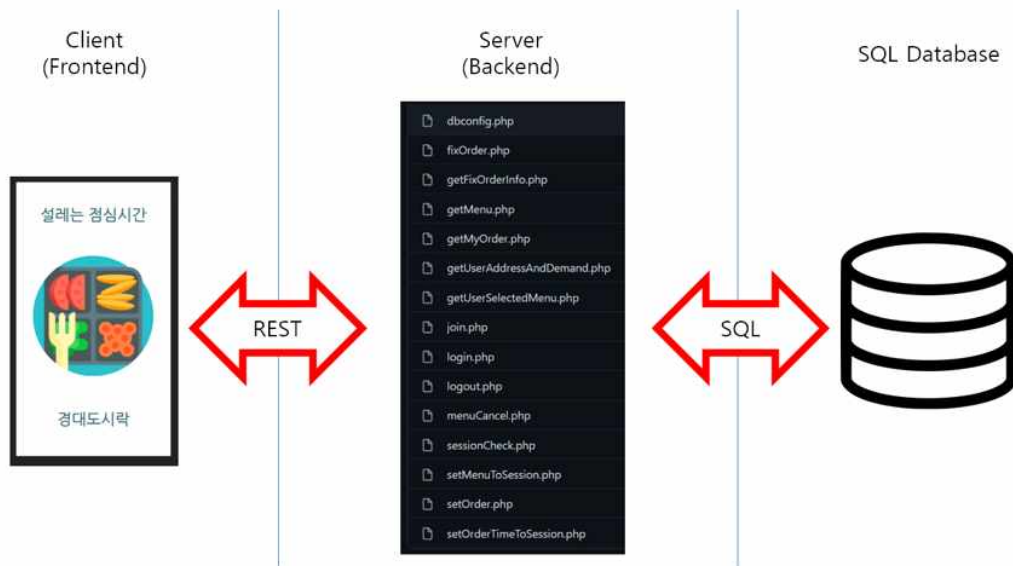
그래서 우리, REST API 를 PHP 로 만들 것이다.

이런 어려운 것까지 배워야하는가? 실제로, 2020년 카카오 개발형 코딩테스트에 다음과 같은 문제가 출제된적이 있다.

“오프라인 코딩 테스트에서는 JSON format 의 데이터를 응답하는 REST API 를 활용해야하니, REST 호출과 JSON format 데이터를 파싱해 활용할 수 있는 parser 코드를 미리 준비해 오시기 바랍니다.”

즉, 요즘 기업은 여러분이 웹개발을 했다고 하면, REST API 는 당연히 알고 있다고 생각한다.

경대도시락의 REST API 를 구경해보자.



경대도시락엔 총 15개의 PHP 파일로 구성된 REST API 를 사용한다. 정확하게는 14개인데, 하나는 DB에 접근하는 코드이기 때문이다.

두가지만 예를 들면,

`join.php` 프론트엔드에서 ID 와 비밀번호를 넘겨받고, DB에 접근해 이미 해당 아이디로 가입되어있는지를 확인한 다음, 가입이력이 없으면 가입시키고 성공시 `true` 를 프론트엔드로 보낸다.

`getMenu.php` 프론트엔드에서 아무것도 보내지 않는다. DB에 접근해 모든 도시락 메뉴를 가져와 프론트로 JSON format 으로 넘겨준다.

즉, REST API 를 사용해 프론트엔드와 백엔드의 경계가 완전히 구분된다. 예전엔 프론트코드에 백엔트코드가 섞여있거나 백엔트코드에 프론트엔트코드가 섞여있어서 뒤죽박죽이었는데, REST API 를 만들어둘 경우, 프론트엔드는 화면만, 백엔드는 서버에만 집중할 수 있는 것이다.

우리가 해볼 일을 살펴보자. 15개까지는 아니고, PHP 파일을 3개 만들 것이다.

유저 생성

`createUser.php`

조교행님
서울시 분당구
유저 생성하기

성공하면 1을 리턴, 동일한 아이디가 있어 실패하면 0을 리턴해 프론트엔드로

보내준다.

getAllUser.php

모든 유저의 정보를 가져옵니다.

모든 유저 가져오기

```
▼ (3) [{"-"}, {"-"}, {"-"}] ⓘ
  ▼ 0:
    address: "서울시 분당구"
    id: 29
    name: "조교행님"
    ▶ [[Prototype]]: Object
  ▼ 1:
    address: "전남 목포"
    id: 30
    name: "전인혁"
    ▶ [[Prototype]]: Object
  ▼ 2:
    address: "부산광역시"
    id: 31
    name: "황혁주"
    ▶ [[Prototype]]: Object
    length: 3
    ▶ [[Prototype]]: Array(0)
```

DB 에 등록된 모든 유저의 정보를 JSON format 으로 보낸다.

getUserById.php

이름과 일치하는
유저의 주소를
가져옵니다.

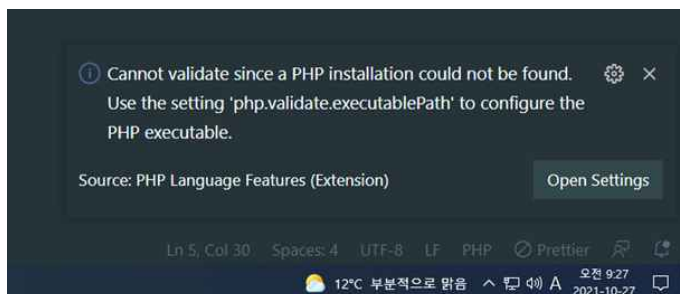
```
▶ {address: '서울시 분당구'}
```

조교행님

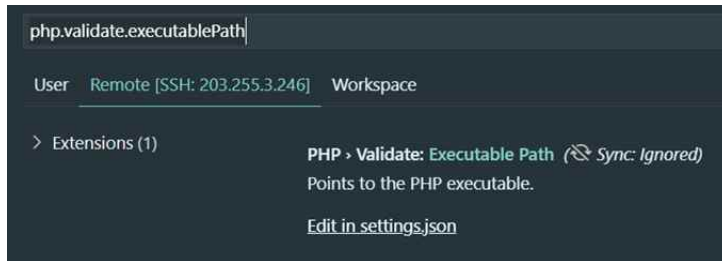
이름과 일치하는 유저 주소 가져오기

프론트엔드에서 입력한 값을 PHP 파일로 보내면, DB에 접근해 해당 유저가 존재하는지 확인 후, 존재한다면 유저의 주소를 JSON format 으로 프론트엔드로 보낸다.

다음 시간부터 본격적인 PHP 코드를 작성해볼 것인데, 여러분은 PHP 익스텐션 두 개를 깔아두었기 때문에 경고를 일으킬 것이다. 현재 VSCode 는 여러분의 컴퓨터에, 대체 어디에 PHP 가 설치되어있는지를 모른다. 다음과 같은 경고창이 뜰 것이다. 이 경고를 해결하는 건 필수는 아니나, 여러분들의 코딩 편의성을 위해 알아보자.



이 경우, 톱니바퀴 setting 을 클릭해 다음과 같이 입력해주고, 밑줄 처진 Edit in setting.json 을 클릭한다.



다음을 추가해주면 된다. 물론 이것은 조교의 경로지, 여러분의 경로와 wampstack 버전은 다를 수 있다.

```
"php.validate.executablePath": "C:/Bitnami/wampstack-8.0.9-0/php/php.exe",
```