codingOn

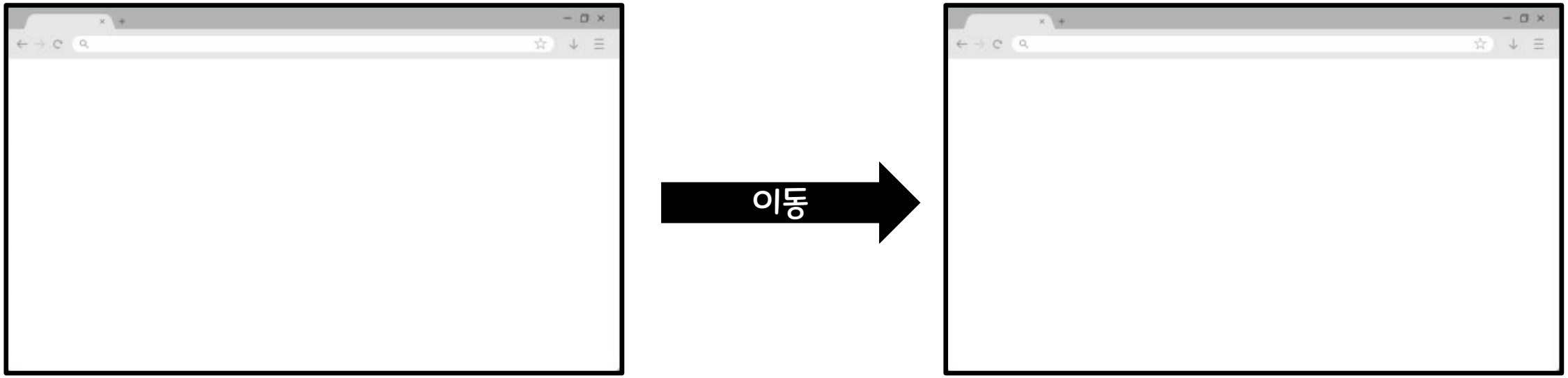
K-Digital Training 웹 풀스택 과정

동적 Form 전송

동적 폼 전송 이론

form 전송

- **<input type="submit">** 이나 **<button type="submit">** 을 이용해 전송
- 전송 시 페이지 이동이 일어난다!!!



form 전송

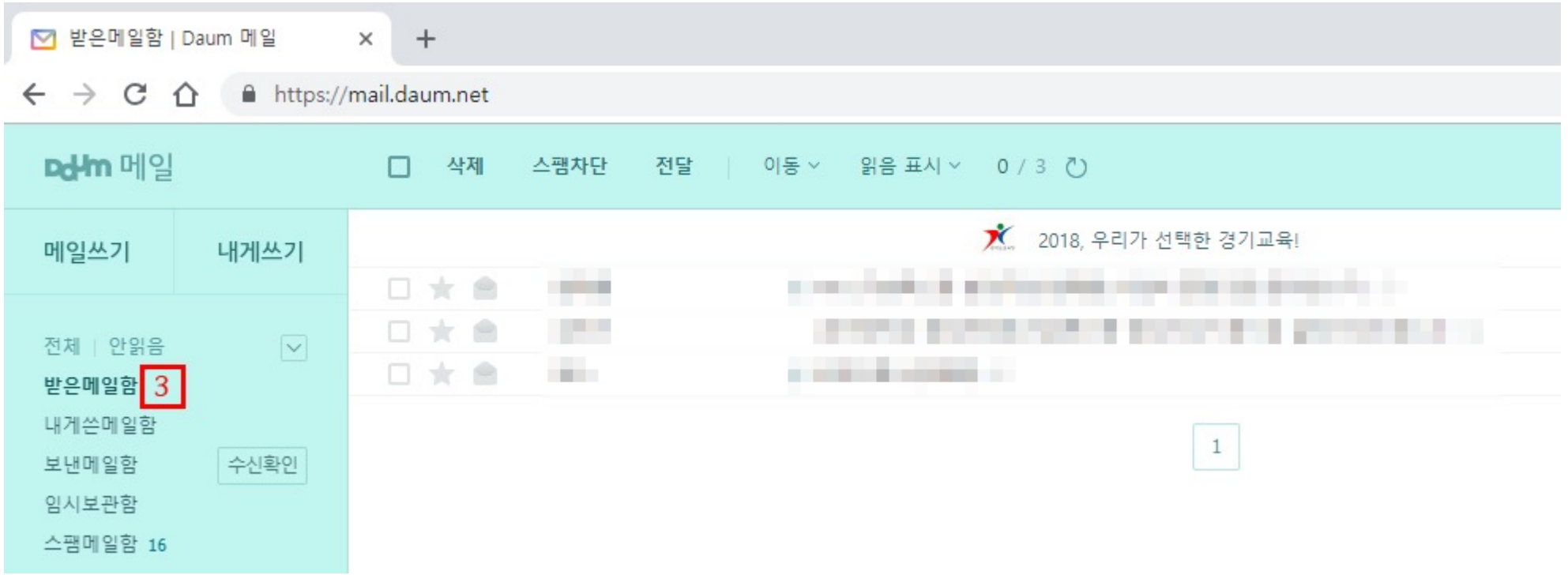
왼쪽처럼 보이게 하기 위해선?

비동기
HTTP 통신

비동기 HTTP 통신

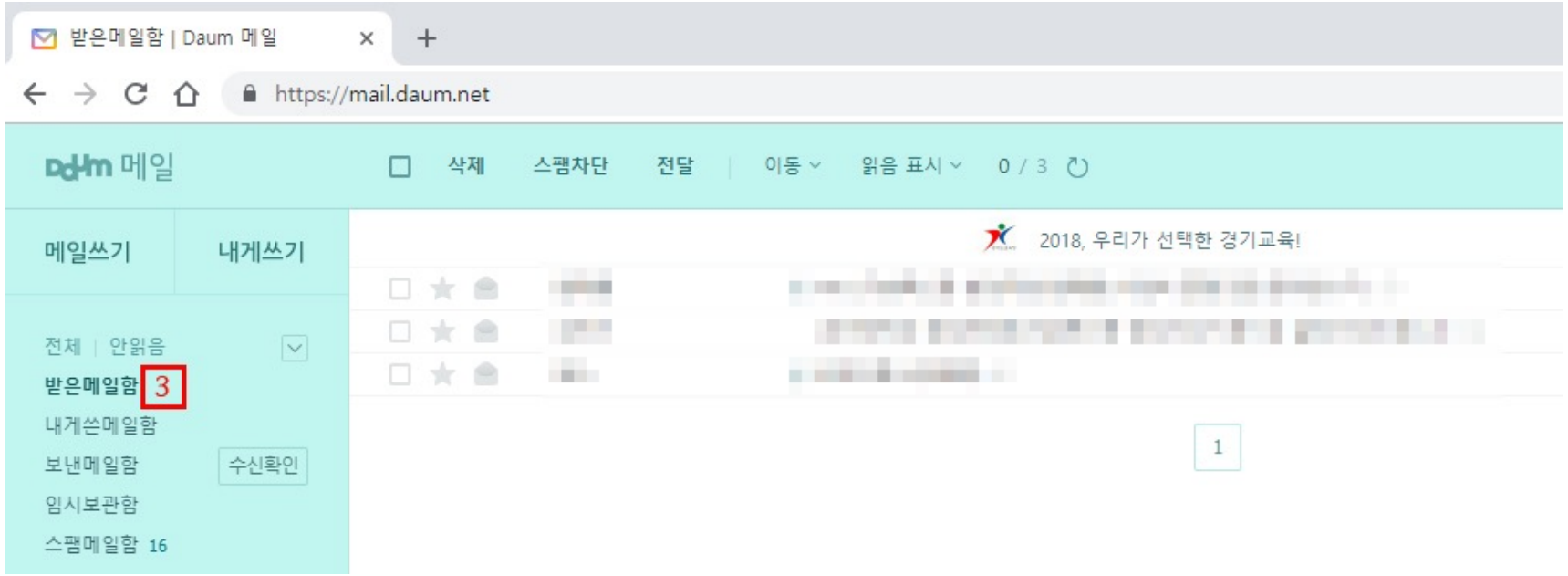
- 동기 방식
  - 한 번에 하나만 처리 -> **페이지를 아예 이동**해 서버가 데이터 처리
- **비동기 방식**
  - **서버에 데이터를 보내고 응답을 기다리는** 동안 다른 처리 가능!

비동기 HTTP 통신


받은메일함 | Daum 메일
https://mail.daum.net
Daum 메일
삭제
스팸차단
전달
이동
읽음 표시
0 / 3
메일쓰기
내게쓰기
2018, 우리가 선택한 경기교육!
전체 | 안읽음
받은메일함 3
내게쓴메일함
보낸메일함
수신확인
임시보관함
스팸메일함 16
1

비동기 HTTP 통신

- **dynamic**
  - 웹 문서가 정적으로 멈춰있는 것이 아니라 **일부 내용이 실시간으로 변경**되는 것
- 비동기 HTTP 통신 : 폼의 데이터를 서버와 dynamic하게 송수신 하는 것



클라이언트 방법1

codingOn

```
<form name="name-form" id="id-form">
    <input type="text" id="id" /><br />
    <input type="password" id="pw" /><br />
    <button type="button" onclick="login()">로그인</button>
</form>
<script>
    function login() {
        const form = document.forms["name-form"];
        const idform = document.getElementById("id-form");
    }
</script>
```

클라이언트 방법2

```
<form onsubmit="login(event)" name="name-form" id="id-form">
    <input type="text" id="id" /><br />
    <input type="password" id="pw" /><br />
    <button type="submit">로그인</button>
</form>
<script>
    function login(event) {
        event.preventDefault(); // 기본 제출 동작을 막음
        const form = document.forms["name-form"];
        const idform = document.getElementById("id-form");
    }
</script>
```

비동기 HTTP 통신 방법

codingOn

1. Ajax
2. Axios
3. Fetch

1. Ajax

- Asynchronous JavaScript And XML
- 자바스크립트를 이용해 클라이언트와 서버 간에 데이터를 주고 받는 비동기 HTTP 통신

- EXtensible Markup Language
- HTML과 비슷한 마크업 언어
- HTML와 달리 정해져 있는 것이 아니라 사용자가 정의해 사용 가능하다.

1. Ajax

- 장점
  - **JQuery를 통해 쉽게** 구현 가능
  - Error, Success, Complete의 상태를 통해 실행 흐름을 조절할 수 있다.

- 단점
  - Jquery를 사용해야만!! 간편하고 호환성이 보장된다. (xml 사용은 복잡)
  - Promise 기반이 아니다.

```
<script src="https://code.jquery.com/jquery-3.6.1.min.js"
  integrity="sha256-o88AwQnZB+VDvE9tvIXrMQaPlFFSUTR+nldQm1LuPXQ="
  crossorigin="anonymous"></script>
```

1. Ajax

codingOn

GET 요청

```
$.ajax({
  url: '/ajax',
  type: 'GET',
  data: data, // { name: form.name.value, gender: form.gender.value, }
  success: function (data) {
    console.log(data);
    resultBox.textContent = `GET /ajax 요청 완료!! ${data.name} 님은 ${data.gender}이시죠?ㅎㅎ`;
    resultBox.style.color = 'blue';
  },
});
```

POST 요청

```
$.ajax({
  url: '/ajax',
  type: 'POST',
  data: data, // { name: form.name.value, gender: form.gender.value, }
  success: function (data) {
    console.log(data);
    resultBox.textContent = `POST /ajax 요청 완료!! ${data.name} 님은 ${data.gender}이시죠?ㅎㅎ`;
    resultBox.style.color = 'red';
  },
});
```

2. Axios

codingOn

- **Node.js**와 브라우저를 위한 **Promise API**를 활용
- 비동기 HTTP 통신이 가능, **return**이 **Promise 객체**로 온다.

2. Axios

- 장점
  - **Promise 기반**으로 만들어졌다.
  - **브라우저 호환성이 뛰어나다.**
- 단점
  - 모듈 설치 or 호출을 해줘야 사용이 가능하다.

```
# 서버 (npm)
npm install axios

# 클라이언트 (cdn)
<script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>
<script src="https://unpkg.com/axios/dist/axios.min.js"></script>
```

2. Axios

GET 요청

```
axios({
    method: 'GET',
    url: '/axios',
    params: data, // GET 요청 => params에!!
}).then(function (response) {
    console.log(response);
    console.log(response.data);
    resultBox.textContent = `GET /axios 요청 완료!! ${response.data.name}님은 ${response.data.gender}이시죠?ㅎㅎ`;
    resultBox.style.color = 'green';
});
```

POST 요청

```
axios({
    method: 'post',
    url: '/axios',
    data: data, // POST 요청 => data에!!
}).then((response) => {
    console.log(response);
    console.log(response.data);
    resultBox.textContent = `POST /axios 요청 완료!! ${response.data.name}님은 ${response.data.gender}이시죠?ㅎㅎ`;
    resultBox.style.color = 'gold';
});
```

3. Fetch

codingOn

- ES6부터 들어온 JavaScript **내장 라이브러리**
- **Promise 기반**

3. Fetch

- 장점
  - JavaScript 내장 라이브러리이므로 **별도의 import 필요 X**
  - Promise 기반
- 단점
  - 최신 문법
  - Timeout 기능이 없다.
  - 상대적으로 Axios에 비해 **기능 부족**

3. Fetch - post

codingOn

GET 요청

```
fetch(`/fetch${urlQuery}`, {
  method: 'GET',
})
  .then(function (response) {
    return response.json(); // JSON 응답을 네이티브 JavaScript 객체로 파싱
  })
  .then(function (data) {
    console.log(data);
    resultBox.textContent = `GET /fetch 요청 완료!! ${data.name}님은 ${data.gender}이시죠?ㅎㅎ`;
    resultBox.style.color = 'limegreen';
  });
```

POST 요청

```
fetch('/fetch', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
  },
  body: JSON.stringify(data), // 데이터를 JSON 형식으로!
})
  .then(function (response) {
    return response.json(); // JSON 응답을 네이티브 JavaScript 객체로 파싱
  })
  .then(function (data) {
    console.log(data);
    resultBox.textContent = `POST /fetch 요청 완료!! ${data.name}님은 ${data.gender}이시죠?ㅎㅎ`;
    resultBox.style.color = 'hotpink';
  });
```

3. Fetch - response

response 에는 Promise를 기반으로 하는 다양한 메서드(함수) 존재. 이 메서드들을 사용하면 다양한 형태의 응답 처리 가능

`response.text()` – 응답을 읽고 텍스트를 반환

`response.json()` – 응답을 JSON 형태로 파싱(실제로 확인해보면 javascript 객체 형태로 반환)

```
app.post("/fetch", (req,res)=>{
    var data = {
        name: req.body.name
    }
    res.send(data);
});

app.post("/fetch", (req,res)=>{
    res.send("안녕");
});
```

```
fetch("/fetch", {
  "요청에 대한 설정"
})
.then((response) => response.json())
.then((data)=>{
    console.log(data);
})
```

```
})
.then((response) => response.text())
.then((data)=>{
    console.log(data);
})
```

codingOn

JSON

JSON이란?

codingON

- JavaScript Object Notation의 약자
- 데이터를 문자열의 형태로 나타내기 위해서 사용
  - 즉, 데이터를 표시하는 방법 중 하나
- 네트워크를 통해 다른 시스템들이 소통할 때 사용하는 경량의 DATA 교환 형식
- 자바스크립트에서 파생되었으나 현재는 다른 프로그래밍 언어에서도 지원하는 데이터 포맷

- 가독성이 뛰어나 컴퓨터와 사람 모두 해석하기 편함
- JavaScript의 Object를 기반으로 하는 텍스트 형식

JSON이란?

codingOn

- JavaScript Object Notation
- JavaScript 객체와 유사
- key 이름을 **큰 따옴표("key-name")로 감싸는 차이점**이 있음
- 문자열, 숫자(정수, 실수), 불리언(true, false), 중첩된 객체와 배열 저장도 가능함

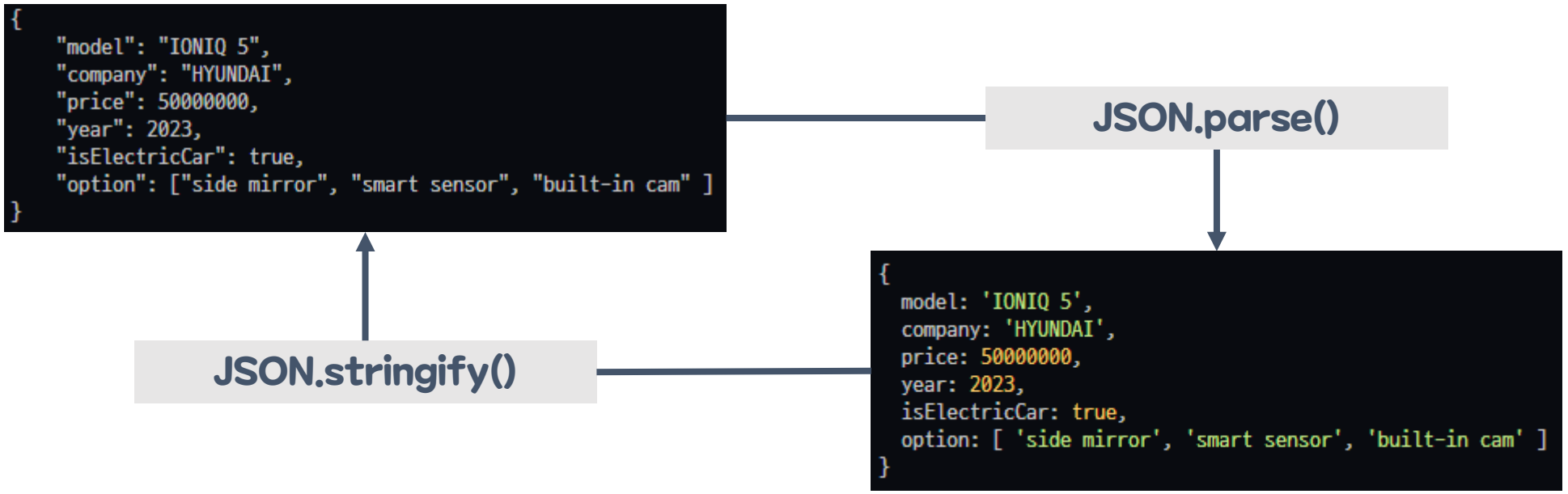
```
{
  "name": "Sean",
  "age": 20,
  "skills": ["JavaScript", "JAVA", "Python"],
  "family": {
    "father": "Jake",
    "mother": "Sunny",
  },
  "height": 120.9
  "isLogin": true,
}
```

JSON 예시

요약하자면 **Client와 Server가**
JSON 이라는 특별한 형식의 데이터로 정보를 주고 받는다!

JSON 내장 객체

- JavaScript에서는 JSON 데이터를 간편하게 다룰 수 있는 JSON 내장 객체 존재
- **JavaScript Object**와 **JSON 문자열**을 서로 변환할 수 있도록 메서드 제공
  - **JSON.parse()**: JSON 문자열을 JavaScript 객체로 변환
  - **JSON.stringify()**: JavaScript 객체를 JSON 문자열로 변환



Axios

Axios 문법 - 요청

```
axios({
  url: '통신하고자 하는 주소',
  method: '통신하고자 하는 방식',
  data: { json 형태의 보내고자 하는 데이터 }
});
```

- url : 서버 주소
  - form 에서의 action에 해당한다.
  - **내가 데이터를 보내고자 하는 주소**

Axios 문법 - 요청

```
axios({
  url: '통신하고자 하는 주소',
  method: '통신하고자 하는 방식',
  data: { json 형태의 보내고자 하는 데이터 }
});
```

- method : 요청방식 (default 값은 get)
  - get
  - post
  - patch
  - delete

Axios 문법 - 요청

```
axios({
  url: '통신하고자 하는 주소',
  method: '통신하고자 하는 방식',
  data: { json 형태의 보내고자 하는 데이터 }
});
```

- data : 보내고자 하는 데이터
  - { key: value, key: value }
  - 위와 같은 형태로 만들어 보낸다.
  - put, post, patch 일 때 사용
  - Request의 **body**로 데이터를 보낸다.

Axios 문법 - 요청

codingOn

```
axios({
  url: '통신하고자 하는 주소',
  method: 'get',
  params: { ? 뒤에 오는 쿼리 값들 }
```

- Params : URL 파라미터
  - GET 방식으로 보낼 때 ? 뒤에 객체로 보내는 것
  - { key: value, key: value } 로 작성한다.
  - Request의 **query** 가 받는다.

Axios 문법 - 요청

```
axios({
  url: '통신하고자 하는 주소',
  method: 'get',
```

- Params 값을 안 보낼거면 url 자체를
- http://~~~?key=value&key=value 라고 보내도 된다.

Axios 문법 – 응답

```
axios({
    method: 'post',
    url: '/axios',
    data: data
}).then((response) => {
    console.log(response.data);
    console.log(response.status)
    console.log(response.statusText)
    console.log(response.headers)
    console.log(response.config)
});
```

Axios 문법 – 응답

```
axios({
    method: 'post',
    url: '/axios',
    data: data
}).then((response) => {
    console.log(response.data)
    console.log(response.status)
    console.log(response.statusText)
    console.log(response.headers)
    console.log(response.config)
});
```

response.data

서버가 제공한 응답(데이터)

response.status

서버 응답의 HTTP 상태 코드

성공이면 200

response.headers

서버가 응답한 헤더

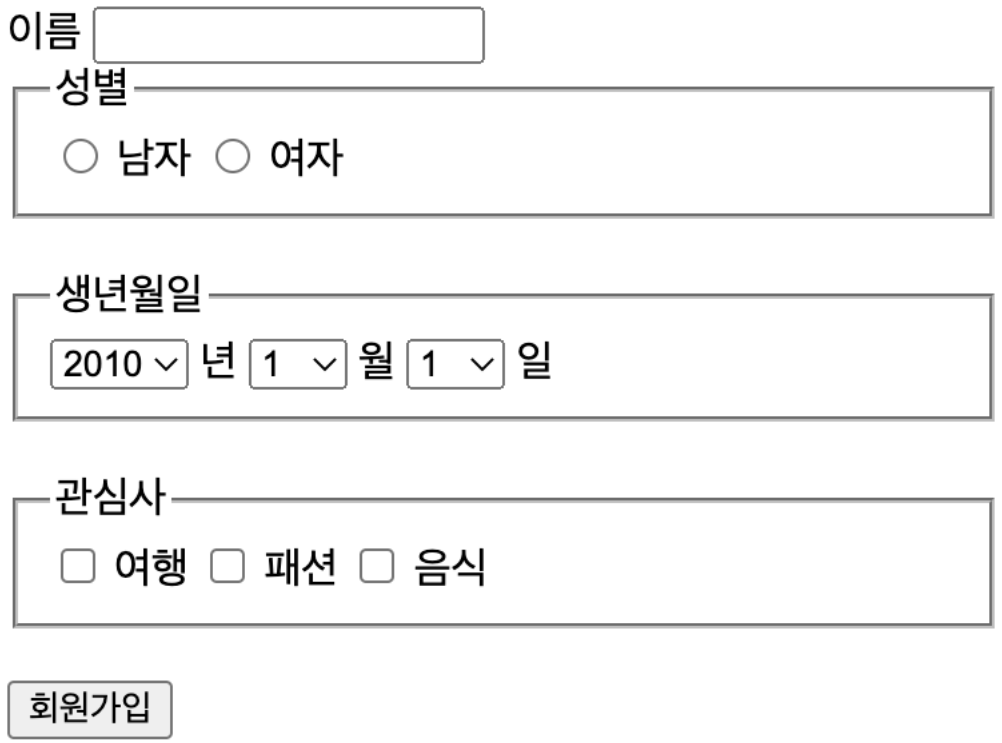
Axios를 백에서는?

- **Res.send()** 를 이용해 데이터를 보낸다.
- Res.send를 이용하면 **데이터를 클라이언트로 다시 보낼 수 있다.**
- **json응답시 res.json()을 이용한다.**

동적 폼 전송 실습

실습1. axios get 회원가입

codingOn



이전에 진행한 실습 “회원가입 '' 을

axios의 **get 메소드**를 이용해 받게끔 작업하기

실습2. axios post 로그인

index.js에서 id, pw를 변수로 저장해두고, 로그인 할 수 있게 만들기

이때, 로그인은 **axios**의 **post**를 이용하기

Axios의 결과를 받아와 “로그인” 버튼 아래에 메시지로 보여주기

- 실패 메시지는 빨간 글자
- 성공 메시지는 파란 글자
- Ex) 네이버 로그인 화면