SeSAC 용산 1기, 

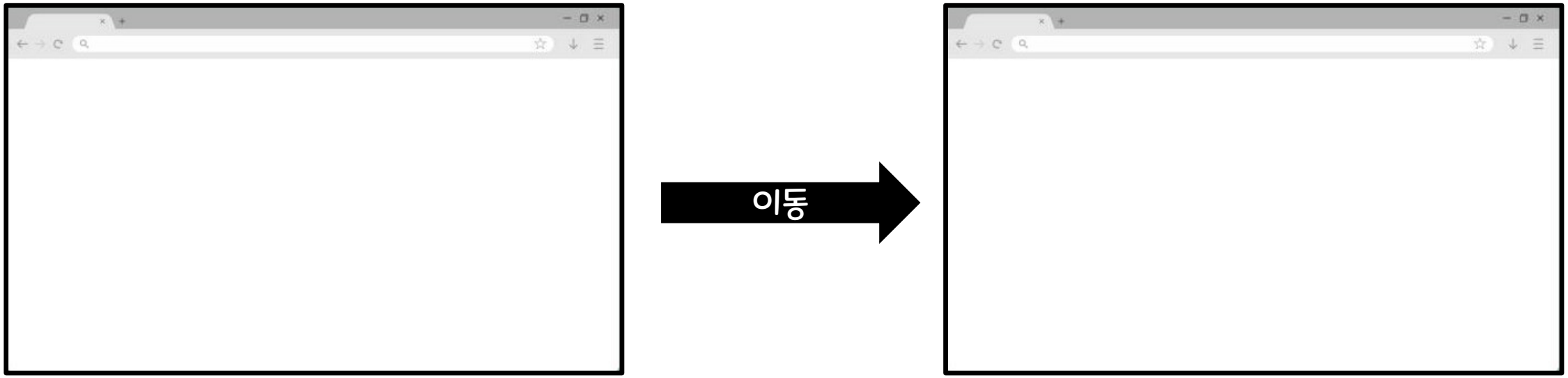
동적 form 전송 수업

WITH 팀 뤼쳐드 

동적폼전송

form 전송

- 〈input type="submit"〉 이나 〈button type="submit"〉 을 이용해 전송
- 전송 시 페이지 이동



form 전송

왼쪽처럼 보이게 하기 위해선?

비동기
HTTP 통신

비동기 HTTP 통신

- 동기 방식
  - 한 번에 하나만 처리 -> 페이지를 아예 이동해 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 변경되는 것
- 비동기 HTTP 통신 : 폼의 데이터를 서버와 dynamic하게 송수신 하는 것

비동기 HTTP 통신 방법

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.

1. Ajax

```
$.ajax({
    url: "/ajax",
    type: "POST", // get을 사용해도 url의 변화는 없음
    data: data,
    success: function(data){
        console.log(data);
    }
})
```

2. Axios

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

2. Axios

- 장점
  - Timeout 기능이 존재한다.
  - **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

```
axios({
  method: 'post',
  url: '/user',
  data: {
    userName: 'Cocoon',
    userId: 'co1234'
  }
}).then((response) => console.log(response));
```

3. Fetch

- 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

```
fetch("/fetch", {
    method: 'post',
    headers: {
        "Content-Type": "application/json",
    },
    body: JSON.stringify(data)
})
.then((response) => response.json())
.then((data)=>{
    console.log(data);
})
```

3. Fetch - get

```
var urlQuery = `?name=${form.name.value}&gender=${form.gender.value}`;

fetch("/fetch"+urlQuery, {
    method: 'get',
})
.then((response) => response.json())
.then((data)=>{
    console.log(data);
})
```

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);
})
```

(추가) JSON이란?

JSON이란

- JavaScript Object Notation라는 의미의 축약어로 데이터를 저장하거나 전송할 때 많이 사용되는 **경량의 DATA 교환 형식**
- Javascript에서 객체를 만들 때 사용하는 표현식
- JSON은 데이터 포맷일 뿐이며 어떠한 통신 방법도, 프로그래밍 문법도 아닌 단순히 데이터를 표시하는 표현 방법

JSON 특징

- 서버와 클라이언트 간의 교류에서 일반적으로 많이 사용된다.
- 자바스크립트 객체 표기법과 아주 유사하다.
- 자바스크립트를 이용하여 JSON 형식의 문서를 쉽게 자바스크립트 객체로 변환할 수 있는 이점이 있다.
- **JSON 문서 형식은 자바스크립트 객체의 형식을 기반으로 만들어졌다.**
- 자바스크립트의 문법과 굉장히 유사하지만 **텍스트 형식일 뿐**이다.
- 특정 언어에 종속되지 않으며, 대부분의 프로그래밍 언어에서 JSON 포맷의 데이터를 핸들링할 수 있는 라이브러리를 제공한다

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

```
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: "get", // 통신 방식
    url: "www.naver.com", // 서버
})
.then(function(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: "get", // 통신 방식
    url: "www.naver.com", // 서버
})
.then(function(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

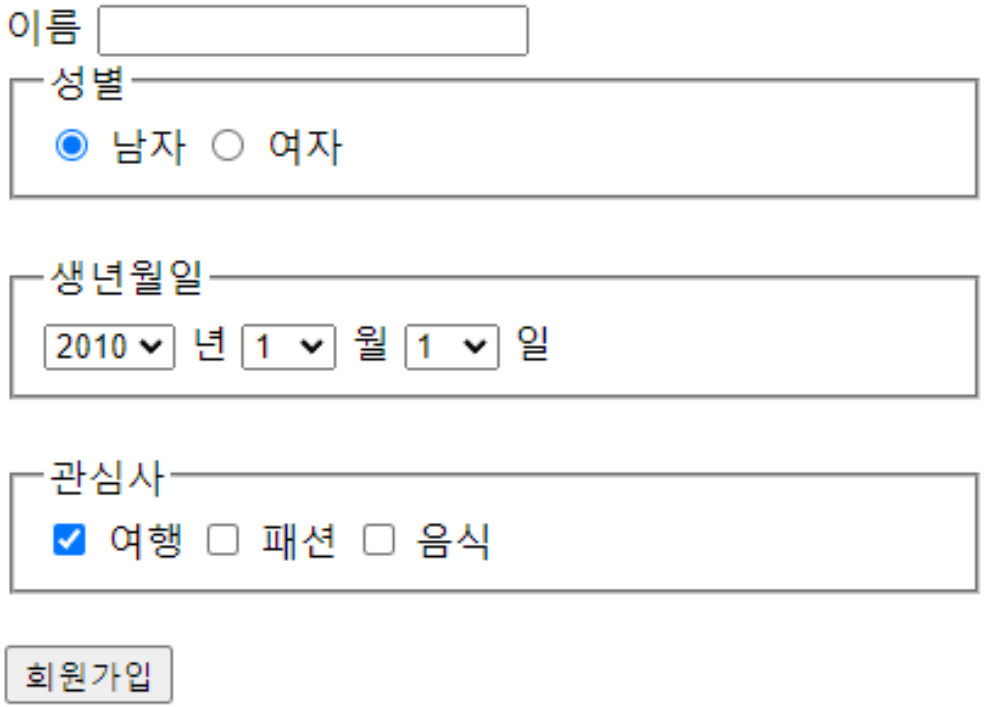
서버가 응답한 헤더

Axios를 백에서는?

- **Res.send()** 를 이용해 데이터를 보낸다.

- Res.send를 이용하면 **데이터를 클라이언트로 다시 보낼 수 있다.**

실습 32. 30번 -> axios로 보내기



앞에서 진행한 실습 “회원가입 " 을 **axios**의 **get 메소드**를 이용해 받게끔 작업하기

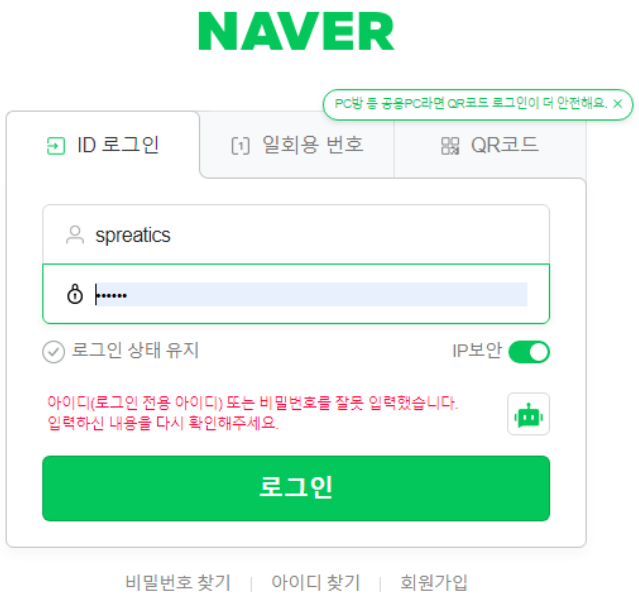
실습 33. 회원가입 -> 로그인

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

이때, 로그인은 **axios**의 **post**를 이용하기

Axios 의 결과를 받아와 “로그인“ 버튼 아래에 메시지로 보여주기

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



실습34. axios -> ajax, fetch

Axios 로 오늘 실습했던 모든 코드를

Ajax와 Fetch로도 동작 가능하게 바꿔보기