8253jang@daum.net
장희정

React Axios 들어가기에 앞서

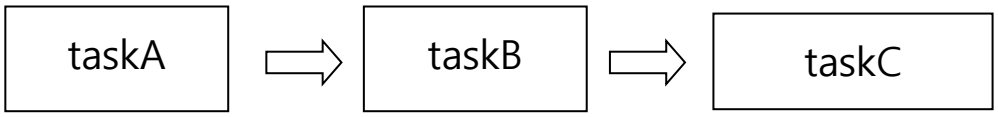
동기화(Synchronous) vs 비동기화(Asynchronous)

: 순서대로 실행하는 것과 그렇지 않은 것들

1) 동기화(Synchronous)

동기는 동시에 일어난다는 뜻으로 요청(request)을 하면 요청에 대한 결과 즉,응답(response)이 올 때 까지 기다린다는 뜻이다.

:다른 작업이 실행되는 되는 동안 다른 작업을 할수없는것을 **블로킹방식**이라한다.



만약, taskA(0.2초) , taskB(10초) , taskC(0.4초) 걸리는 작업이라면 taskB작업시간이 너무 길어 성능이 저하된다.

이럴때, 비동기화가 필요하다.

2) 비동기화(Asynchronous)

비동기는 "동시에 일어나지 않는다"는 뜻으로 요청(request)를 하고 응답(response)이 올 때까지 기다리지 않고 다른 작업을 할 수 있도록 하는 것이다.

여러개의 작업을 동시에 실행시킨다. 즉 먼저 작성한 코드의 결과를 기다리지 않고 다음 코드를 바로 실행한다.–**논블로킹방식**

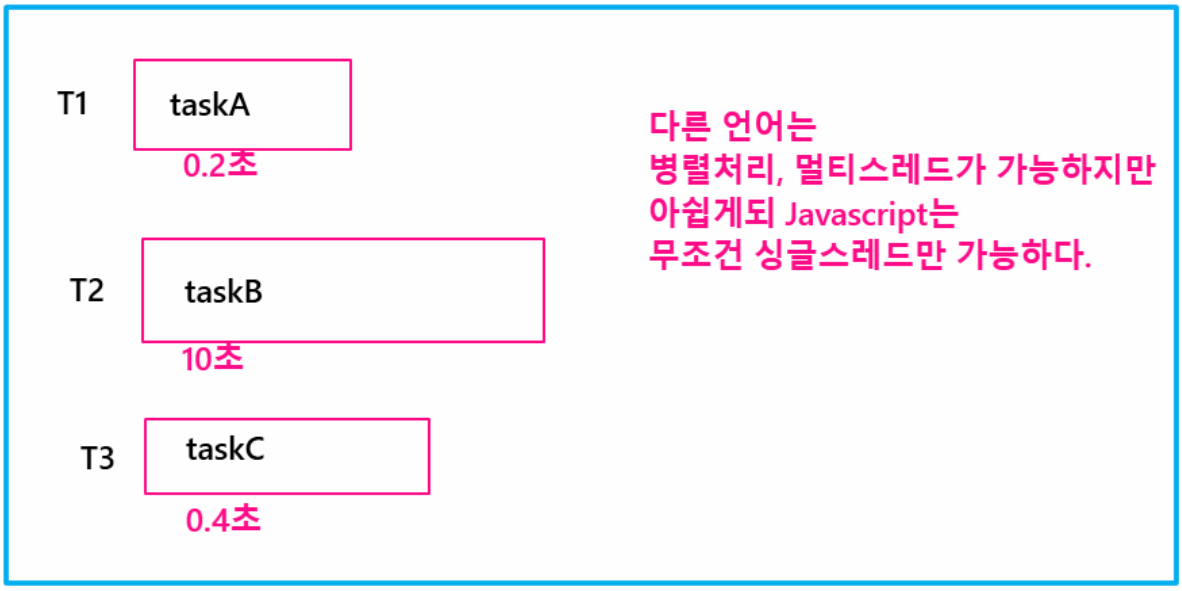
동기방식은 설계가 매우 간단하고 직관적이지만 결과가 주어 질 때까지 아무것도 못하고 대기해야하는단점이 있고,

비동기방식은 동기보다 복잡하지만 결과가 주어지는데 시간이 걸리더라도 그 시간 동안 다른 작업을 할 수 있으므로 자원을 효율적으로 사용할 수 있는 장점이 있다

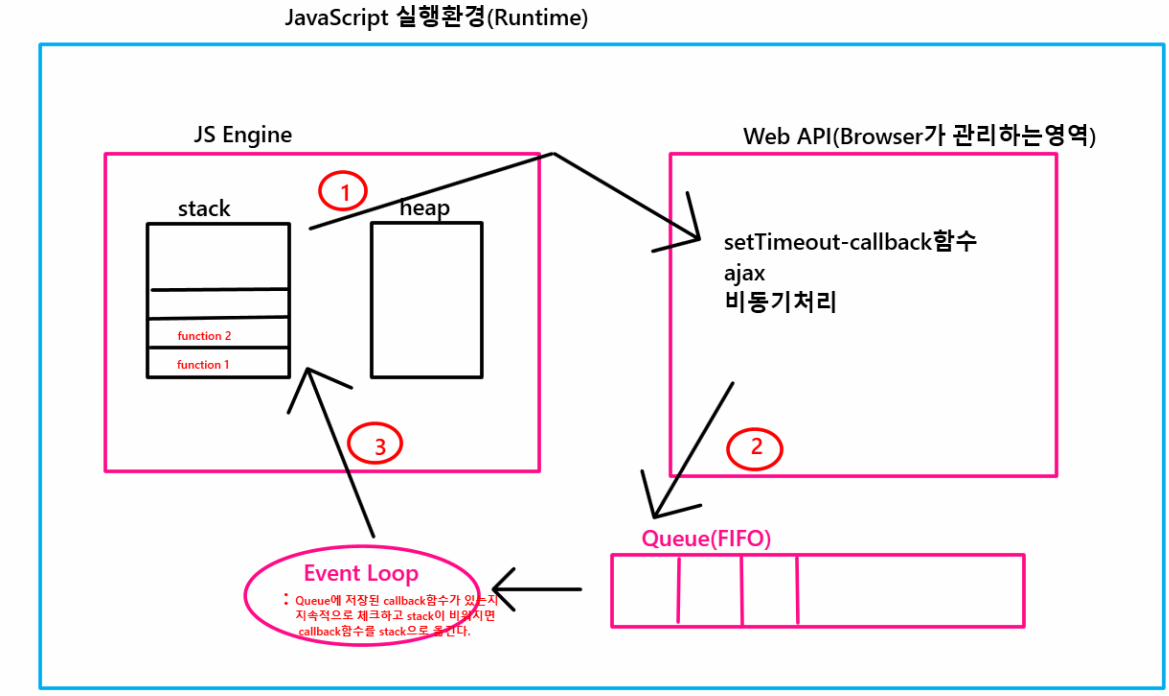
"자바스크립트는 싱글스레드로 동작한다"

: 멀티스레드가 불가하기 때문에 비동기작업이필요.

8253jang@daum.net
장희정



그렇다면, JavaScript가비동기처리를어떻게?



비동기화(Asynchronous)처리 방법

1) 순수 JavaScript 기반

:소스가 길어지고 특정 작업이 끝난 후에 다른 작업(콜백함수)을 실행하고 또 그 작업이 끝난 후에 또 다른 작업(콜백함수)을 실행 해야 하는 경우에 콜백지옥의 문제점이 발생한다. 가독성이 떨어지고 깔끔하지 못한 코드의 문제점.

2) **Promise**

: 비동기 처리를 효율적으로 작업 할 수 있도록 도와주는 자바스크립트 내장객체

:순수 자바스크립기반의 비동기화 처리의 문제점 Callback hell에서 탈출 하기 위한 JS 비동기를 돕는 함수이다.

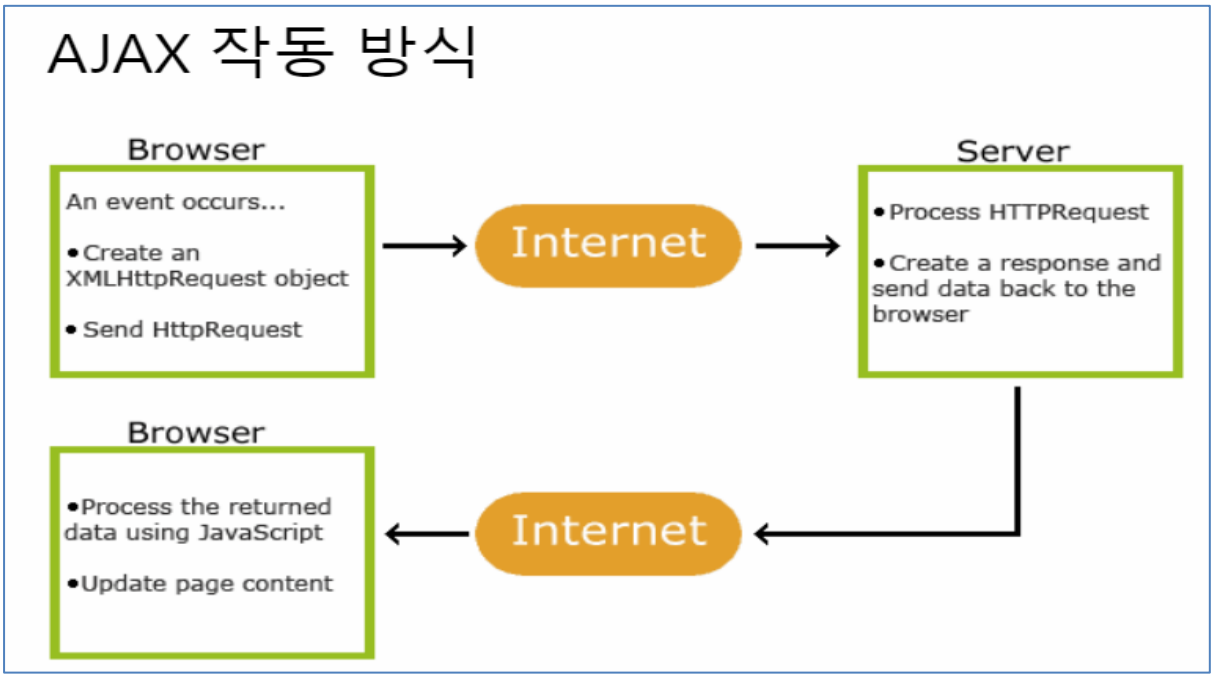
3) **Async& await**

:: Promise를 좀더 세련되고 직관적으로 처리 할수 있도록 돕는다.

Ajax(Asynchronous **J**avaScript **A**nd **X**ML**)란?**

Ajax 는 프로그래밍 언어가 아니라 기술적 용어로, 웹 애플리케이션과 서버 간에 데이터를 주고받는 데 사용된다. Ajax 는 클라이언트와 서버 사이에서 비동기적으로 데이터를 교환하여 웹 페이지를 새로고침하지 않고도 일부 내용을 업데이트할 수 있게 해준다.

즉, 페이지가 로드된 후 페이지 전체를 새로고침하지 않고도 일부만 갱신할 수 있는 기술로, 브라우저에 내장된 XMLHttpRequest 객체를 이용해 웹 서버에 데이터를 요청하고, 그 결과를 XML 또는 JSON 형태로 받아 JavaScript 기반으로 화면에 반영한다.



8253jang@daum.net
장희정

요즘, 웹은 한 페이지에서 다양한 요청을 보내고 그 결과를 받아야 하는 경우가 많기 때문에 특정 요청이 올 때까지 다른 요청을 할 수 없으면 사용자 입장에서는 로딩이 느리고 화면이 제대로 랜더링되지 않는다.

그러므로, Ajax 기술을 이용하면 한 페이지에서 전체 새로고침 없이 화면의 일부분만 데이터를 갱신할 수 있기 때문에 훨씬 효율적이다.

비동기통신

애플리케이션의 사용자 경험을 향상시키며 서버로부터 응답이 오는 동안 사용자가 다른 작업을 계속할 수 있다.

그렇다면,
JavaScript에서 비동기 통신을 어떻게?

1) **Fetch 함수 – IE에서 지원이잘 안됨.**
   : 원격 API를 간편하게 호출 할 수 있도록 브라우저에서 제공하는 fetch()함수 - window.fetch(url, option);

2) **외부 lib 사용(Promise기반의 Http 비동기 통신라이브러리)**
   : 원격 API를 호출 하는 라이브러리
   - jQuerylib
   - Axios lib

   이를 통해 개발자들은 비동기 작업의 성공 또는 실패에 따라 콜백 함수를 체인으로 연결할 수 있다.

여기서 잠깐,

콜백함수(Callback Function)란?

: 함수를 파라미터(매개변수)로 전달받는 함수를 뜻함.
: 함수에서 특정 작업이 완료되었을 때,추가로 실행 해야 하는 작업을 기술 한 함수를 만들 때 많이 사용한다.

ex) **setTimeout(콜백함수 ,ms); / Ajax와같은Network작업**

8253jang@daum.net
장희정

-콜백함수를 이용하지 않은 코드

```
13  function add(a, b){
14     return a + b;
15  }
16
17 function minus(a,b){
18     return a-b;
19 }
20
21 function printAdd(a, b){
22     let result = add(a, b);
23     console.log(a+" + " +b + " = " + result);
24 }
25
26 function printMinus(a, b){
27     let result = minus(a, b);
28     console.log(a+" -" +b + " = " + result);
29 }
30
31 printAdd(5, 3);
32 printMinus(5, 3);
33
```

-콜백함수 이용한 코드

```
13  function add(a, b){
14     return a + b;
15  }
16
17 function minus(a,b){
18     return a-b;
19 }
20
21 function print(a, b, callback){
22     let result = callback(a, b);
23     console.log(a+" - " +b + " = " + result);
24 }
25
26 print(5, 3 , add);
27 print(5, 3, minus);
28
```

☞콜백함수의 장점

1) 함수를 인자로 받기 때문에 필요에 따라 함수의 정의를 달리해 전달할 수 있다.
2) 함수를 굳이 정의하지 않고 익명 함수로도 전달 할 수 있다.
3) 비동기(Asynchronous) 처리 방식을 좀더 간결하게 할 수 있다.

☞콜백함수의 단점

1) 콜백함수를 너무 남용하면 코드의 가독성이 떨어진다. **(콜백 지옥)**
2) 에러 처리가 어렵다.

8253jang@daum.net
장희정

콜백지옥란?

콜백지옥(callback hell)이란 콜백함수를 익명함수로 전달하는 과정에서 또 다시 콜백안에 함수 호출이 반복되어 코드의 들여쓰기 수준이 감당하기 힘들 정도로 깊어지는 현상을 말한다.

콜백지옥 코드 예

```
function test() {
    setTimeout(() => {
        console.log('하나');
        setTimeout(() => {
            console.log('둘');
            setTimeout(() => {
                console.log('셋');
            }, 10);
        }, 10);
    }, 10);

    console.log("시작~~");
}

test();
```

Ex) 비동기 코드를 만들어보자.

ex01.html

```
<script type="text/javascript">
function taskA() {
      setTimeout(() => {
        console.log("taskA End");
      }, 2000);
    }

    taskA(); //이작업을 끝날때까지 기다리지 않고 아래 실행후 2초후에 taskA end 나옴 (비동기방식)
    console.log("code end");
</script>
```

실행결과

```
code end                    ex01.html:16
taskA End                   ex01.html:11
>
```

8253jang@daum.net
장희정

Ex) 비동기처리의 결과 값을 확인하기 위한 콜백함수 전달해보자.

ex02.html

```
<script type="text/javascript">
   function taskA(a, b, cb) {
      //매개변수로 함수를 전달하는것을 콜백함수로 함.
      setTimeout(() => {
        const res = a + b;
        console.log("taskA End = res = " + res);
        cb(res); //콜백함수 호출
      }, 3000);
    }

    taskA(3, 4, (result) => {
      console.log("result = " + result);
    }); //이작업을 끝날때까지 기다리지 않고 아래 실행후 2초후에 taskA end 나옴 (비동기방식)
    console.log("code end");

</script>
```

-실행결과

```
code end                                    ex02.html:21
taskA End = res = 7                         ex02.html:13
result = 7                                  ex02.html:19
>
```

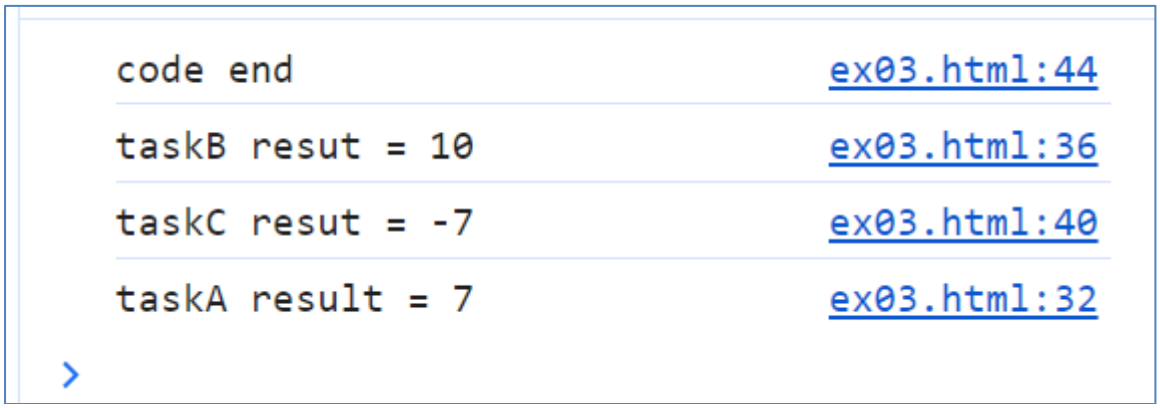
Ex) 작업 여러 개를 만들어서 A->B->C 순으로 실행 해보자.

ex03.html

```
<script type="text/javascript">
  function taskA(a, b, cb) {
    setTimeout(() => {
        const res = a + b;
        cb(res); //콜백함수 호출
      }, 3000);
  }

  function taskB(a, cb) {
      setTimeout(() => {
        const res = a * 2;
        cb(res);
      }, 1000);
  }

  function taskC(a, cb) {
      setTimeout(() => {
        const res = a * -1;
        cb(res);
      }, 2000);
  }

   //함수 호출  taskA -> taskB -> taskC
    taskA(3, 4, (result) => {
      console.log("taskA result = " + result);
    });

    taskB(5, (result) => {
      console.log("taskB resut = " + result);
    });

    taskC(7, (result) => {
      console.log("taskC resut = " + result);
    });

    console.log("code end");

</script>
```

- 실행결과

8253jang@daum.net
장희정



Ex) 위 코드는 동기적 작업이 안된다.콜백 함수의 결과를 받아서 다른 작업의 인자로 전달하여 호출 해보자.

Ex04.html

```
<script type="text/javascript">
  //콜백함수의 결과를 받아서 다른 작업의 인자로 전달하여 호출해보자
   function taskA(a, b, cb) {
      //매개변수로 함수를 전달하는것을 콜백함수로 함.
      setTimeout(() => {
        const res = a + b;
        cb(res); //콜백함수 호출
      }, 3000);
    }

    function taskB(a, cb) {
      setTimeout(() => {
        const res = a * 2;
        cb(res);
      }, 1000);
    }

    function taskC(a, cb) {
      setTimeout(() => {
        const res = a * -1;
        cb(res);
      }, 2000);
    }

    //함수 호출  taskA -> taskB -> taskC
    taskA(4, 5, (res) => {
      console.log("taskA result = " + res);
      //res의 결과를 받아서 taskB의 인자로 전달
      taskB(res, (b_res) => {
        console.log("taskB result = " + b_res);

        taskC(b_res, (c_res) => {
          console.log("taskC result = " + c_res);
        });
      });
    });

    console.log("code end");
</script>
```

-실행결과

```
code end                                    ex04.html:44
taskA result = 9                            ex04.html:33
taskB result = 18                           ex04.html:36
taskC result = -18                          ex04.html:39
>
```

위 코드가 복잡하다 – 콜백지옥(callback hell)
: JS에서 콜백지옥을 벗어나기 위해서 promise를 지원한다.

Ex)위 코드를 Promise로 변경해보자.
ex04.html

<Promise문법>

Promise Syntax

```
let myPromise = new Promise(function(myResolve, myReject) {
// "Producing Code" (May take some time)

  myResolve(); // when successful
  myReject();  // when error
});

// "Consuming Code" (Must wait for a fulfilled Promise)
myPromise.then(
  function(value) { /* code if successful */ },
  function(error) { /* code if some error */ }
);
```

Axios란?

- **https://axios-http.com/docs/api_intro**
- Axios는 node.js와 브라우저를 **위한 Promise 기반 HTTP 비동기 통신 라이브러리이다**.
- 백엔드 서버와 프론트엔드 서버간의 통신을 위해 사용한다.
- 서버 사이드에서는 네이티브 node.js의 http 모듈을 사용하고, 클라이언트(브라우저)에서는

XMLHttpRequests를 사용한다.

- **HTTP 요청과 응답을 JSON 형태로 반환해준다.**
- 주요 HTTP 요청 유형(GET, POST, PUT, DELETE 등)을 지원이를 통해 웹 애플리케이션은 서버와 다양한 방식으로 통신할 수 있다.
- **Axios는 더 간결하고 명확한 오류 처리 방식을 제공한다.**
- Promise 기반 구조를 활용하여 .catch 메서드를 사용할 수 있다.
- 일부 오래된 브라우져에서는 작동하지 않을 수 있다.

Axios(config)

```
axios({
        method:,
      url   :
        data:
    })
    .then((result)=>{
       //성공
  })
  .catch((error)=>{
       //실패
  });
}
```

메소드	설명
axios.get(url[, config])	HTTP GET 요청을 보내 서버로부터 데이터를 가져옴
axios.post(url, data[, config])	HTTP POST 요청을 보내 서버에 새로운 데이터를 생성
axios.put(url, data[, config])	HTTP PUT 요청을 보내 서버의 기존 데이터를 갱신
axios.delete(url[, config])	HTTP DELETE 요청을 보내 서버의 데이터를 삭제
axios.request(config)	사용자 정의 HTTP 요청을 보냄 메소드, 헤더, 본문 등을 지정
axios.head(url[, config])	HTTP HEAD 요청을 보내 서버의 헤더 정보만을 가져옴
axios.options(url[, config])	HTTP OPTIONS 요청을 보내 서버가 지원하는 HTTP 메소드를 조회
axios.patch(url, data[, config])	HTTP PATCH 요청을 보내 서버의 기존 데이터 중 일부를 수정

이제 ReactAxios라이브러를 사용하여 비동기통신 해보자.

1) VS_CODE tool을 열고 Terminal을 open한다.

2) npm create vite@latest my-axios 프로젝트 생성

3) cd my-axios 폴더 이동

4) npm i

5) npm install react-axios설치

6) package.json 확인

```
"dependencies": {
  "@testing-library/jest-dom": "^5.17.0",
  "@testing-library/react": "^13.4.0",
  "@testing-library/user-event": "^13.5.0",
  "react": "^18.2.0",
  "react-axios": "^2.0.6",
  "react-dom": "^18.2.0",
  "react-scripts": "5.0.1",
  "web-vitals": "^2.1.4"
},
```

코드예시)

8253jang@daum.net
장희정

```
3  import axios from 'axios';
4
5  function App() {
6    axios.get("https://jsonplaceholder.typicode.com/users")
7    .then(function (result) {
8      console.log(result)
9    })
10   .catch(function (error) {
11     console.error(error);
12   });
```

실행 결과)

```
{data: Array(10), status: 200, statusText: '', headers: {…}, config: {…}, …}
  config: {transitional: {…}, transformRequest: Array(1), transformResponse: Array(1), timeout: 0, adapter: ƒ, …}
  data: Array(10)
    0: {id: 1, name: 'Leanne Graham', username: 'Bret', email: 'Sincere@april.biz', address: {…}, …}
    1: {id: 2, name: 'Ervin Howell', username: 'Antonette', email: 'Shanna@melissa.tv', address: {…}, …}
    2: {id: 3, name: 'Clementine Bauch', username: 'Samantha', email: 'Nathan@yesenia.net', address: {…}, …}
    3: {id: 4, name: 'Patricia Lebsack', username: 'Karianne', email: 'Julianne.OConner@kory.org', address: {…}, …}
    4: {id: 5, name: 'Chelsey Dietrich', username: 'Kamren', email: 'Lucio_Hettinger@annie.ca', address: {…}, …}
    5: {id: 6, name: 'Mrs. Dennis Schulist', username: 'Leopoldo_Corkery', email: 'Karley_Dach@jasper.info', address: {…}, …}
    6: {id: 7, name: 'Kurtis Weissnat', username: 'Elwyn.Skiles', email: 'Telly.Hoeger@billy.biz', address: {…}, …}
    7: {id: 8, name: 'Nicholas Runolfsdottir V', username: 'Maxime_Nienow', email: 'Sherwood@rosamond.me', address: {…}, …}
    8: {id: 9, name: 'Glenna Reichert', username: 'Delphine', email: 'Chaim_McDermott@dana.io', address: {…}, …}
    9: {id: 10, name: 'Clementina DuBuque', username: 'Moriah.Stanton', email: 'Rey.Padberg@karina.biz', address: {…}, …}
    length: 10
    [[Prototype]]: Array(0)
  headers: {cache-control: 'max-age=43200', content-type: 'application/json; charset=utf-8', expires: '-1', pragma: 'no-cache'}
  request: XMLHttpRequest {onreadystatechange: null, readyState: 4, timeout: 0, withCredentials: false, upload: XMLHttpRequestUpload, …}
  status: 200
  statusText: ""
  [[Prototype]]: Object
```

☞ **샘플 데이터(가상데이터)를 제공하는 무료 온라인 REST API서비스**

https://jsonplaceholder.typicode.com/

실습해보기

App.jsx파일

```
123   return (
124     <>
125     <h3>React Axios </h3>
126     <button onClick={selectAll}>get-selectAll</button>
127     <button onClick={selectById}>get-selectById</button>
128
129      <hr/>
130
131     <h3>Spring Boot 연동하기(CRUD) </h3>
132     <button onClick={insertUser}>Post - user등록 </button>
133     <button onClick={deleteUser}>Delete - user삭제 </button>
134     <button onClick={putUser}>Put - user수정 </button>
135     <button onClick={getById}>Get - 부분조회 </button>
136     <button onClick={getUsers}>Get - 전체검색 </button>
137
138     </>
139   );
```

8253jang@daum.net
장희정

```
function App() {
  //전체조회
  const selectAll = ()=>{
    axios
    .get("https://jsonplaceholder.typicode.com/users")
    .then((result)=>{
        console.log(result);
        result.data.map((user)=> console.log(user.id +" | " + user.name ));

    })
    .catch((error)=>{
        console.log(error);
    });
  }

  const selectById = ()=>{
    axios({
      url:"https://jsonplaceholder.typicode.com/users/2" ,
      method:"get" ,
      //data : ,
    })
    .then((result)=>{
      console.log(result.data)
    })
    .then(()=>{
      console.log("성공이후 해야할 작업입니다.~~")
    })
    .catch((err)=>{
      console.log(err);
    });
  }
```

```
36  /////Spring Boot//////////////////////////////////////
37  const insertUser =()=>{
38   axios({
39     //url:"http://localhost:9000/a" ,
40     url:"http://localhost:9000/user" ,
41     method:"post" ,
42     data :{name:"jang" , email :"8253jang@daum.net", age:20} ,
43   })
44   .then((result)=>{
45     console.log(result.data)
46      //화면 출력...
47
48   })
49   .catch((err)=>{
50      console.log(err);
51   });
52  }
53
54  const deleteUser =()=>{
55   axios({
56     //url:"http://localhost:9000/user/5" ,
57     url:"http://localhost:9000/user/3" ,
58     method:"delete" ,
59   })
60   .then((result)=>{
61     console.log(result)
62      //화면 출력...
63
64   })
65   .catch((err)=>{
66      console.log(err);
67      alert(err.response.data.title +" | " +err.response.data.detail  )
68   });
69
70  }
71
```

```
71
72    //수정
73    const putUser = ()=>{
74     axios({
75       method:"PUT",
76       url:"http://localhost:9000/user/2",
77       data:{name:"gahyun",email:"kkk@daum.net", age :20 },
78     })
79     .then((result)=>{
80       console.log( result);
81     })
82     .catch((err)=>{
83       console.log(err);
84     });
85   };
86
87   //조회
88   const getById = ()=>{
89    alert(1)
90    axios({
91      method:"get",
92      url:"http://localhost:9000/user/7",
93    })
94    .then((result)=>{
95      console.log( "result =", result);
96    })
97    .catch((err)=>{
98      console.log(err.response.data)
99      console.log("detail = " , err.response.data.detail);
100     console.log("title = " , err.response.data.title);
101     console.log("timestamp = " , err.response.data.timestamp);
102     console.log("status = " , err.response.data.status);
103   });
104  };
105
```

8253jang@daum.net
장희정

```
107  //전체조회
108  const getUsers = ()=>{
109   axios({
110     method:"get",
111     url:"http://localhost:9000/users"
112   })
113   .then((result)=>{
114     console.log( result);
115   })
116   .catch((err)=>{
117     console.log(err);
118   });
119  };
```

Spring boot 프로젝트 SOP정책을 풀어주기 위한 CORS설정 Configuration

```
1   package web.mvc.config;
2
3   import org.springframework.context.annotation.Configuration;
4   import org.springframework.web.servlet.config.annotation.CorsRegistry;
5   import org.springframework.web.servlet.config.annotation.EnableWebMvc;
6   import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
7
8   @Configuration
9    @EnableWebMvc
10   public class WebMvcConfig implements WebMvcConfigurer {
11      @Override  no usages
12      public void addCorsMappings(CorsRegistry registry) {
13          registry.addMapping( pathPattern: "/**")
14                  .allowedOrigins("http://localhost:5173")
15                  .allowedMethods("OPTIONS", "GET", "POST", "PUT", "DELETE");
16      }
17
18  }
```

Spring boot Controller 소스

8253jang@daum.net
장희정

```java
@RestController
@Slf4j
//@CrossOrigin
public class AxiosTestController {

    @GetMapping("/a")
    public String test(){
        log.info("여기 오니??");
        return "OK";
    }

    /**
     * 등록하기
     * */
    @PostMapping("/user")
    public String insert(@RequestBody  UserReq userReq){
        System.out.println("userReq = " + userReq);
        return "ok";
    }

```

```java
    /**
     * 삭제하기
     * */
    @DeleteMapping("/user/{id}")
    public ResponseEntity<?> delete(@PathVariable Integer id){
        System.out.println("id = " + id);

        if(id !=5){
            throw new MemberAuthenticationException("아아디값 유효하지 않아요.","ID Bad", HttpStatus.BAD_REQUEST);
        }

        return  ResponseEntity.status(HttpStatus.OK).body("OK");
    }

```

8253jang@daum.net
장희정

```java
/**
 * 수정하기
 * */
@PutMapping("/user/{id}")
public ResponseEntity<?> update(@PathVariable Integer id , @RequestBody UserReq userReq){
    System.out.println("id = " + id);

    System.out.println("userReq = " + userReq);
    return ResponseEntity.status(HttpStatus.OK).body("1");
}

/**
 * 부분검색
 * */
@GetMapping("/user/{id}")
public ResponseEntity<?> select(@PathVariable String id){
    System.out.println("id = " + id);
    UserReq userReq = new UserReq( name: "Chan", email: "chan@naver.com", age: 50);

    return ResponseEntity.status(HttpStatus.OK).body(userReq);
}
```

```java
/**
 * 전체검색
 * */
@GetMapping("/users")
public ResponseEntity<?> selectAll(){
    List<UserReq> list = new ArrayList<>();
    list.add(new UserReq( name: "A", email: "a@daum.net", age: 10));
    list.add(new UserReq( name: "B", email: "b@daum.net", age: 20));
    list.add(new UserReq( name: "C", email: "c@daum.net", age: 30));

    return ResponseEntity.status(HttpStatus.OK).body(list);
}
```