[Javascript] Ajax/JSON/XMLHttpRequest

1 Ajax란?

Ajax(Asynchronous Javascript and XML) 자바스크립트를 사용하여
브라우저가 서버에게 비동기 방식으로 데이터를 요청하고,
서버가 응답한 데이터를 수신하여 웹페이지를 동적으로 갱신하는 프로그래밍 방식.

브라우저에서 제공하는 Web API인 XMLHttpRequest 객체를 기반으로 동작한다.
XMLHttpRequest는 HTTP 비동기 통신을 위한 메서드와 프로퍼티를 제공한다.

Ajax의 장점

- 필요한 데이터만 서버로부터 전송받기 때문에 불필요한 데이터 통신이 발생하지 않는다.
- 변경할 필요가 없는 부분은 다시 렌더링하지 않기 때문에 화면이 순간적으로 깜빡이는 현상이 발생하지 않는다.
- 클라이언트와 서버와의 통신이 비동기 방식으로 동작하기 때문에 서버에게 요청을 보낸 이후 블로킹이 발생하지 않는다.

2 JSON

JSON(Javascript Object Notation)은 클라이언트와 서버 간의 HTTP 통신을 위한 텍스트 데이터 포맷이다.
언어 독립형 데이터 포맷으로, 대부분의 프로그래밍 언어에서 사용할 수 있다.

JSON 표기 방식

```
{
    "name" : "Lee",
    "age" : 20,
    "alive" : true,
    "hobby" : ["traveling", "tennis"]
}
```

※ 키는 반드시 큰따옴표로 묶어야 한다.
※ 값에서는 문자열은 반드시 큰따옴표로 묶어야 한다.

JSON.stringify

JSON.stringify 메서드는 객체를 JSON 포맷의 문자열로 변환한다.
클라이언트가 서버로 객체를 전송하려면
객체를 문자열화해야 하는데 이를 **직렬화**라 한다

```
const obj = {
    name : "Lee",
    age : 20,
    alive : true,
    hobby : ['traveling', 'tennis']
};

//객체를 JSON 포맷의 문자열로 변환한다.
const json = JSON.stringify(obj);
console.log(typeof json, json);
//string {"name":"Lee","age":20,"alive":true,"hobby":["traveling","tennis"]}
```

```
//객체를 JSON 포맷의 문자열로 변환하면서 들여쓰기 한다.
const prettyJson = JSON.stringify(obj, null, 2);
console.log(typeof prettyJson, prettyJson);
// string {
//     "name": "Lee",
//     "age": 20,
//     "alive": true,
//     "hobby": [
//       "traveling",
//       "tennis"
//     ]
//   }

//replacer 함수. 값의 타입이 Number이면 필터링되어 반환되지 않는다.
function filter(key, value){
    return typeof value === 'number' ? undefined : value;

}

//JSON.stringify 메서드에 두 번째 인수로 replacer 함수를 전달한다.

const strFilterObject = JSON.stringify(obj, filter, 2);
console.log(typeof strFilterObject, strFilterObject);

//string {
//  "name": "Lee",
//  "alive": true,
//  "hobby": [
//    "traveling",
//   "tennis"
//  ]
//}
```

JSON.stringify 메서드는 객체뿐만 아니라 배열도 JSON 포맷의 문자열로 변환한다.

```
const todos = [
    {id : 1, content: 'HTML', completed:false},
    {id : 2, content: 'CSS', completed:true},
    {id : 3, content: 'Javascript', completed:false}
];

//배열을 JSON 포맷의 문자열로 변환한다.
```

```
const json2 = JSON.stringify(todos, null, 2);
console.log(typeof json2, json2);

// string [
//     {
//       "id": 1,
//       "content": "HTML",
//       "completed": false
//     },
//     {
//       "id": 2,
//       "content": "CSS",
//       "completed": true
//     },
//     {
//       "id": 3,
//       "content": "Javascript",
//       "completed": false
//     }
//   ]
```

JSON.parse

JSON.parse 메서드는 **JSON 포맷의 문자열을 객체로 변환**한다.
서버로부터 클라이언트에게 전송된 JSON 데이터는 문자열이다. 이 문자열을 객체로서 사용하려면 JSON 포맷의 문자열을 객체화해야 하는데 이를 역직렬화라 한다.

```
const obj2 = {
    name : "Lee",
    age : 20,
    alive : true,
    hobby : ['traveling', 'tennis']
};

//객체를 JSON 포맷의 문자열로 변환한다.
const json3 = JSON.stringify(obj);

//JSON 포맷의 문자열을 객체로 변환한다.
const parsed = JSON.parse(json);
console.log(typeof parsed, parsed);
```

배열이 JSON 포맷의 문자열로 변환되어 있는 경우 JSON.parse는 문자열을 배열 객체로 변환한다.

```
const todos = [
    {id : 1, content: 'HTML', completed:false},
    {id : 2, content: 'CSS', completed:true},
    {id : 3, content: 'Javascript', completed:false}
];

//배열을 JSON 포맷의 문자열로 변환한다.
const json4 = JSON.stringify(todos);

//JSON 포맷의 문자열을 배열로 변환한다. 배열의 요소까지 객체로 변화된다.
const parsed2 = JSON.parse(json4);
console.log(typeof parsed2, parsed2);
```

3 XMLHttpRequest

브라우저는 주소창이나 HTML의 form 태그 또는 a 태그를 통해 HTTP 요청 전송 기능을 기본 제공한다.
자바스크립트를 사용하여 HTTP 요청을 전송하려면 XMLHttpRequest 객체를 사용한다.
Web API인 XMLHRequest 객체는 **HTTP 요청 전송과 HTTP 응답 수신을 위한 다양한 메서드와 프로퍼티를 제공**한다.

XMLHttpRequest 객체 생성

XMLHttpRequest 생성자 함수를 호출하여 생성.

```
const xhr = new XMLHttpRequest();
```

XMLHttpRequest 객체의 프로퍼티와 메서드

- **XMLHttpRequest 객체의 프로토타입 프로퍼티**

프로토타입 프로퍼티	설명
readyState	HTTP 요청의 현재 상태를 나타내는 정수. 다음과 같은 XMLHttpRequest의 정적 프로퍼티를 값으로 갖는다. - UNSENT : 0

	- OPENED : 1 - HEADERS_RECEIVED: 2 - LOADING :3 - DONE: 4
status	HTTP 요청에 대한 응답 상태(HTTP 상태 코드)를 나타내는 정수 ex) 200
statusText	HTTP 요청에 대한 응답 메시지를 나타내는 문자열 ex) "OK"
responseType	HTTP 응답 타입 ex) document, json, text, blob, arraybuffer
response	HTTP 요청에 대한 응답 몸체. responseType에 따라 타입이 다르다.

- **XMLHttpRequest 객체의 이벤트 핸들러 프로퍼티**

이벤트 핸들러 프로퍼티	설명
onreadystatechange	readyState 프로퍼티 값이 변경된 경우
onerror	HTTP 요청에 에러가 발생한 경우
onload	HTTP 요청이 성공적으로 완료한 경우

- **XMLHttpRequest 객체의 메서드**

메서드	설명
open	HTTP 요청 초기화
send	HTTP 요청 전송
abort	이미 전송된 HTTP 요청 중단

HTTP 요청 전송

http 요청을 전송하는 경우 다음 순서를 따른다.

1. XMLHttpRequest.prototype.open 메서드로 HTTP 요청을 초기화한다.
2. 필요에 따라 XMLHttpRequest.prototype.setRequestHeader 메서드로 특정 HTTP 요청의 헤더 값을 설정한다.

3. XMLHttpRequest.prototype.send 메서드로 HTTP 요청을 전송한다.

```
//XMLHttpRequest 객체 생성
const xhr = new XMLHttpRequest();

//HTTP 요청 초기화
xhr.open('GET', '/users');

//HTTP 요청 헤더 설정
// 클라이언트가 서버로 전송할 데이터의 MIME 타입 지정 : json
xhr.setRequestHeader('content-type', 'application/json');

//HTTP 요청 전송
xhr.send();
```

XMLHttepRequest.prototype.open

open 메서드는 서버에 전송할 HTTP 요청을 초기화한다.

```
xhr.open(method, url[, async])
```

매개변수	설명
method	HTTP 요청 메서드("GET", "POST", "PUT", "DELETE" 등
url	HTTP 요청을 전송할 URL
async	비동기 요청 여부, 옵션으로 기본값은 true이며, 비동기 방식으로 동작한다.

HTTP 요청 메서드는 클라이언트가 서버에게 요청의 종류와 목적(리소스에 대한 행위)을 알리는 방법이다.

HTTP 요청 메서드	종류	목적	페이로드
GET	index/retrieve	모든/ 특정 리소스 취득	X
POST	create	리소스 생성	O
PUT	replace	리소스 전체 교체	O
PATCH	modify	리소스의 일부 수정	O

DELETE	delete	모든/특정 리소스 삭제	X

XMLHttepRequest.prototype.send

send 메서드는 open 메서드로 초기화된 HTTP 요청을 서버에 전송한다. 기본적으로 서버로 전송하는 데이터는 GET, POST 요청 메서드에 따라 전송 방식에 차이가 있다.

- GET 요청 메서드의 경우 데이터를 URL의 일부분인 쿼리 문자열로 서버에 전송한다.
- POST 요청 메서드의 경우 데이터(페이로드)를 요청 몸체에 담아 전송한다.

XMLHttepRequest.prototype.setRequestHeader

setRequestHeader 메서드는 특정 HTTP 요청의 헤더 값을 설정한다.
이 메소드는 반드시 open 메서드를 호출한 이후에 호출해야 한다.

MIME 타입	서브타입
text	text/plain, text/html, text/css, text/javascript
application	application/json, application/x-www-form-urlencode
multiple	multiple/formed-data

```
//XMLHttpRequest 객체 생성
const xhr = new XMLHttpRequest();

//HTTP 요청 초기화
xhr.open('POST','/users');

//HTTP 요청 헤더 설정
//클라이언트가 서버로 전송할 데이터의 MIME 타입 지정 : json
xhr.setRequestHeader('content-type', 'application/json');

//HTTP 요청 전송
xhr.send(JSON.stringify({id:1, content:'HTML', completed:false}));
```

4 HTTP 응답 처리

서버가 전송한 응답을 처리하려면 XMLHttpRequest 객체가 발생시키는 이벤트를 개치해야 한다.

```
//XMLHttpRequest 객체 생성
const xhr = new XMLHttpRequest();

//HTTP 요청 초기화
xhr.open('GET', 'https://jsonplaceholder.typicode/todos/1');

//HTTP 요청 전송
xhr.send();

xhr.onreadystatechange = () =>{
    if(xhr.readyState !== XMLHttpRequest.DONE) return;

    if(xhr.status === 200){
        console.log(JSON.parse(xhr.response));
    } else {
        console.log('Error', xhr.status, xhr.statusText);
    }
}
```