제이쿼리 AJAX 요청

AJAX 란?

AJAX란 **asynchronous Javascript and XML**입니다. 요즘은 XML보다 **HTML**이나 **JSON**을 더 많이 쓰지만, 개발 당시 xml이 주류라서 이름이 그렇게 된 것입니다.

기존의 웹에서는 한 번 페이지를 로딩하면 다른 페이지를 로딩하기 위해서 링크를 타고 넘어가야 했습니다. 그렇게 되면 흔히 말하는 페이지 깜빡임이 발생합니다. 구글 검색을 해보면, 페이지를 전환하지 않고도 예상 검색어와 결과를 보여줍니다. Gmail도 마찬가지죠. 바로 AJAX 기술을 사용하였기 때문입니다.

이처럼 비동기적으로 서버에 요청을 하여 페이지 전환 없이도 새로운 데이터를 가져올 수 있습니다. 대표적으로 ajax를 이용해서 브라우저에 요청을 보내는 방식이 두가지 있는데, get과 post 방식이 있습니다.

특징	GET 방식	POST 방식
캐시화(cached)	캐시될 수 있음.	캐시되지 않음.
브라우저 히스토리	히스토리에 쿼리 문자열이 기록됨.	히스토리에 기록되지 않음.
데이터 길이	데이터의 길이가 URL 주소의 길이 이내로 제한됨. (익스플로러에서 URL 주소가 가질 수 있는 최대 길이는 2,083자이며, 이 중에서 순수 경로 길이는 2,048자까지만 허용됨)	제한 없음.
데이터 타입	오직 ASCII 문자 타입의 데이터만 전송할 수 있음.	제한 없음.
보안성	데이터가 URL 주소에 포함되어 전송되므로, 아무나 볼 수 있어 보안에 매우 취약함.	브라우저 히스토리에도 기록되지 않고, 데이터가 따로 전송되므로, GET 방식보다 보안성이 높음.

제이쿼리 ajax 메소드

요즘 대부분은 ajax 요청을 보낼때 fetch() 나 axios 라이브러리를 사용하지만, 몇몇 전통이 있는 기업에선 레거시 프로젝트 때문에 jQuery AJAX를 사용하곤 합니다.

메소드	설명
$.ajax()	비동기식 Ajax를 이용하여 HTTP 요청을 전송함.
$.get()	전달받은 주소로 GET 방식의 HTTP 요청을 전송함.
$.post()	전달받은 주소로 POST 방식의 HTTP 요청을 전송함.
$.getScript()	웹 페이지에 스크립트를 추가함.
$.getJSON()	전달받은 주소로 GET 방식의 HTTP 요청을 전송하여, 응답으로 JSON 파일을 전송받음.
.load()	서버에서 데이터를 읽은 후, 읽어 들인 HTML 코드를 선택한 요소에 배치함.

그리고 POST와 GET 타입 외에도 PUT과 DELETE도 있는 만큼, PUT과 DELETE를 사용하고 싶다면 다른 라이브러리를 사용하는 것도 고려해보아야 합니다.

$.get()

제이쿼리에서는 Ajax를 이용하여 GET 방식의 HTTP 요청을 구현한 `$.get()` 메소드를 제공합니다.
URL 주소는 클라이언트가 HTTP 요청을 보낼 서버의 주소입니다. 그리고 HTTP 요청이 성공했을 때 콜백 함수가 실행되게 됩니다.

JAVASCRIPT

```
$.get(URL주소, 콜백함수]);
```

JAVASCRIPT

```
// GET 방식으로 서버에 HTTP Request를 보냄.
$.get("/examples/media/request_ajax.php",
    // 서버가 필요한 정보를 같이 보냄.
    {
       species: "고양이",
       name: "나비",
       age: 3,
    },
```

```
    // 응답이 오면 호출되는 콜백 함수
    function(data, status) {
        $("#text").html(data + "<br>" + status); // 전송받은 데이터와 전송 성공 여부를 보여줌.
    }
);
```

$.post()

제이쿼리에서는 Ajax를 이용하여 POST 방식의 HTTP 요청을 구현한 `$.post()` 메소드를 제공합니다.

JAVASCRIPT

```
$.post(URL주소, 데이터, 콜백함수);
```

JAVASCRIPT

```
// POST 방식으로 서버에 HTTP Request를 보냄.
$.post("/examples/media/request_ajax.php",
    {
       name: "홍길동",
       grade: "A"
    },              // 서버가 필요한 정보를 같이 보냄.
    function(data, status) {
       $("#text").html(data + "<br>" + status); // 전송받은 데이터와 전송 성공 여부를 보여줌.
    }
);
```

$.ajax()

`$.ajax()` 메소드는 위에서 배운 제이쿼리 get, post 메소드를 하나로 합친 한방 메소드라고 보면 됩니다. 이 메서드 하나로 다양한 요청을 서버에 보낼 수 있습니다. (type 속성이나 method 속성으로 http 메서드를 지정하여 보낼 수 있습니다)

JAVASCRIPT

```
$.ajax(URL주소, 옵션)
```

JAVASCRIPT

```
$.ajax({
    url: "/examples/media/request_ajax.php", // 클라이언트가 HTTP 요청을 보낼 서버의 URL 주소
    data: { name: "홍길동" },                // HTTP 요청과 함께 서버로 보낼 데이터
    method: "GET",                           // HTTP 요청 방식(GET, POST)
    dataType: "json"                         // 서버에서 보내줄 데이터의 타입
})
    // HTTP 요청이 성공하면 요청한 데이터가 done() 메소드로 전달됨.
    .done(function(json) {
    })
    // HTTP 요청이 실패하면 오류와 상태에 관한 정보가 fail() 메소드로 전달됨.
    .fail(function(xhr, status, errorThrown) {
    })
    // HTTP 요청이 성공하거나 실패하는 것에 상관없이 언제나 always() 메소드가 실행됨.
    .always(function(xhr, status) {
    });
```

JAVASCRIPT

```
$.ajax({
    url: 'http://example.com/api/users/1',
    method: 'DELETE', // delete 요청
    success: function(response) {
        console.log('User deleted successfully!');
    },
    error: function(error) {
        console.error('Error deleting user:', error);
```

```
    }
});
```

.load()

load() 메소드는 서버에서 데이터를 읽은 후, 읽어 들인 HTML 코드를 선택한 요소에 표시하는 일종의 한방 단축 메서드라 보면 된다.

JAVASCRIPT

```
// demo_test.txt 내용을 응답받은 뒤에 id가 list인 요소에 표시함
$("#list").load("/demo_test.txt");
```

제이쿼리 ajax 다양한 문법

제이쿼리 ajax 메서드를 이용할때 성공, 에러 처리를 위한 굉장히 다양한 문법 방식이 존재합니다. 자신이 편한 것을 골라 사용하면 됩니다.

콜백 방식

심플하게 하나의 응답에 대한 콜백 함수를 실행에 콜백 내부에서 에러나 다양한 처리를 하는 방법입니다.

JAVASCRIPT

```
// 콜백 방식
$.get('url', function(data) {
  console.log(data);
});
```

프로퍼티 방식

콜백 함수 대신 함수를 포함한 자바스크립트 객체 자체를 인자로 줄수도 있습니다.

JAVASCRIPT

```
$.get('url', {
  success: function(data) {
    console.log(data);
  },
  error: function() {
    console.log('요청이 실패하였습니다.');
  },
  always: function() {
    console.log('요청이 완료되었습니다.'); // always 프로퍼티는 요청이 성공하든 실패하든 항상 실
  }
});
```

프로미스 방식

프로미스 객체를 통한 메서드 체이닝 문법입니다.

JAVASCRIPT

```
$.get('url')
    .done(function() {
      console.log('요청이 성공하였습니다.');
    })
    .fail(function() {
      console.log('요청이 실패하였습니다.');
```

```
    })
    .always(function() {
      console.log('요청이 완료되었습니다.');
    });
```

헤더 설정하기

만일 추가적은 http 헤더를 설정하고 요청을 보내야 할 경우 아래와 같이 사용합니다.

JAVASCRIPT

```
$.ajax({
    type: "GET",
    url: "/api/me",
    beforeSend: function (xhr) { // 서버에 요청 보내기전 헤더 설정
        xhr.setRequestHeader("Content-type", "application/json");
        xhr.setRequestHeader("ApiKey", "asdfasxdfasdfasdf");
    }
});
```