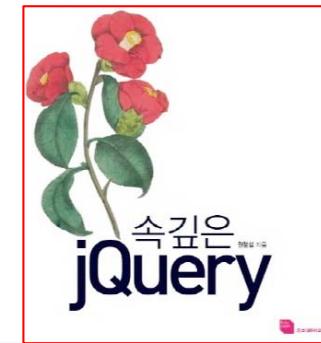


JQuery Ajax



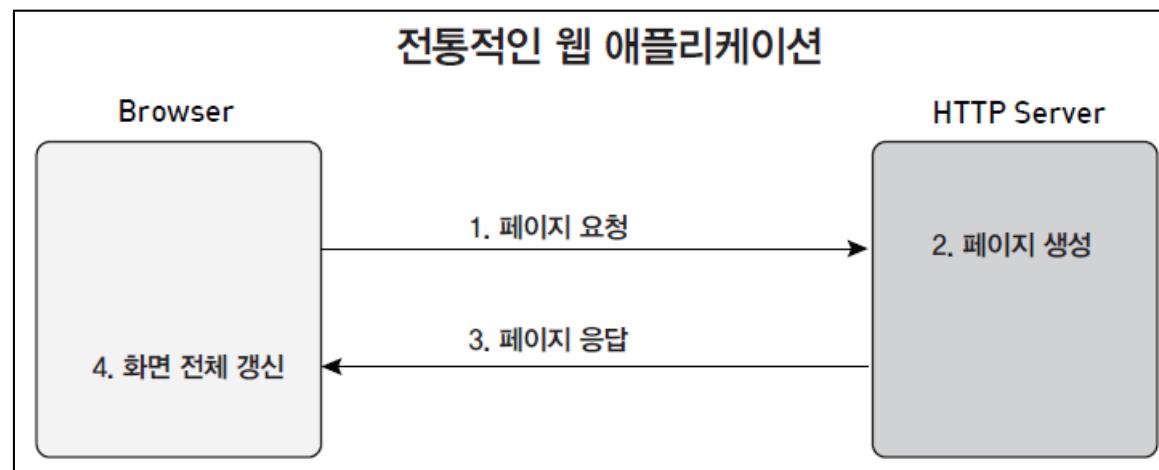
출처 : 속깊은 jQuery(루비페이지, 원형섭)

jQuery Ajax



■ 전통적인 웹 애플리케이션

- 요청의 단위가 페이지!
- 서버에서 화면(페이지)을 생성해서 응답함.(PHP, JSP, ASP, ASP.NET, Django, node.js+express 등 이용)
 - 페이지 단위의 요청과 응답의 문제점은 화면 일부분만 갱신하고 싶어도 페이지 전체를 HTTP 서버로부터 다시 받아와야 한다는 것을 의미
 - HTTP 서버는 브라우저의 요청이 있을 때마다 페이지 전체를 재 생성하여 전송
 - 전송 데이터가 증가하기 때문에 서버 부하 가중



jQuery Ajax



■ "Asynchronous Javascript And XML"

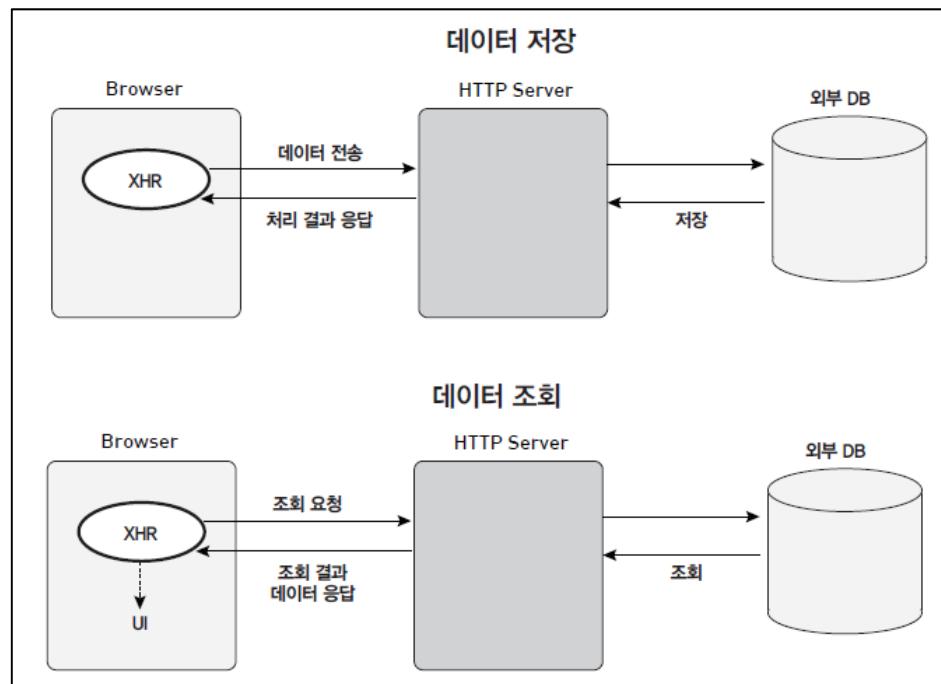
■ Javascript And XML의 의미

- jQuery는 기본적으로 브라우저에서 작동
 - 외부 데이터를 조회하고 싶거나 데이터를 외부 저장소에 저장하고 싶은 경우에는 jQuery만으로 해결할 수 없음
 - 브라우저에서 외부 서버와 통신할 수 있는 기능이 필요
- 브라우저 화면 일부분만 갱신할 수 있어야 함
 - 전통적인 웹 애플리케이션은 서버에서 HTML 문서 즉 화면 UI를 모두 생성해서 응답하기 때문에 요청 단위가 페이지가 될 수 밖에 없었고, 이로 인해 브라우저 화면의 일부분만 갱신하는 것은 사실상 불가능했음.
- 이런 문제 해결을 위해 브라우저는 XHR(XMLHttpRequest)이라는 객체 이용해 외부와 통신할 수 있는 기능을 제공

jQuery Ajax



- XHR 객체를 이용한 요청 단위는 페이지가 아니라 데이터.
 - 자바스크립트 코드를 작성해 XHR 객체를 통해 HTTP 서버와 데이터를 주고 받음
 - 서버로부터 가져온 데이터를 이용해 화면 UI를 생성하는 것은 자바스크립트 코드를 이용
 - 이와 같이 개발하면 브라우저 화면의 일부분만 갱신할 수 있음
 - 즉, 서버에 HTTP 요청을 보낸 뒤 XML, JSON, TEXT 형식 등으로 응답을 받아 페이지의 일부만을 변경



jQuery Ajax



- 웹 페이지에서 Ajax를 이용 예

상품 설명 상품 필수 표기정보 쉬운반품 상품 문의 (26)

[골든쿠폰]대박난박양 HOT한 트레이닝

쿠팡가? **9,800 원**

할인혜택 최대 20,000원 쿠폰할인
카드할인혜택▼ 무이자할부혜택▼

배송정보 무료배송

남은시간 05일 11:47:53 381 개 구매

구매 가능한 옵션만 보기

상품을 선택하세요 ▾

사이즈/색깔/날짜 등을 선택하세요. ▾

구매 할 상품 옵션을 선택 후, 구매하기 버튼을 클릭해주세요.

총 상품금액 : 0원
최대 20,000원 쿠폰할인

바로구매 > **장바구니 담기**

jQuery Ajax



■ "Javascript And XML"

- 자바스크립트 코드를 이용해서 서버와 데이터(XML)를 주고 받겠다는 의미
- XML, HTML, JSON, TEXT 형식 등 다양한 정보 교환 가능

[표 13-1 : JSON, XML 비교]

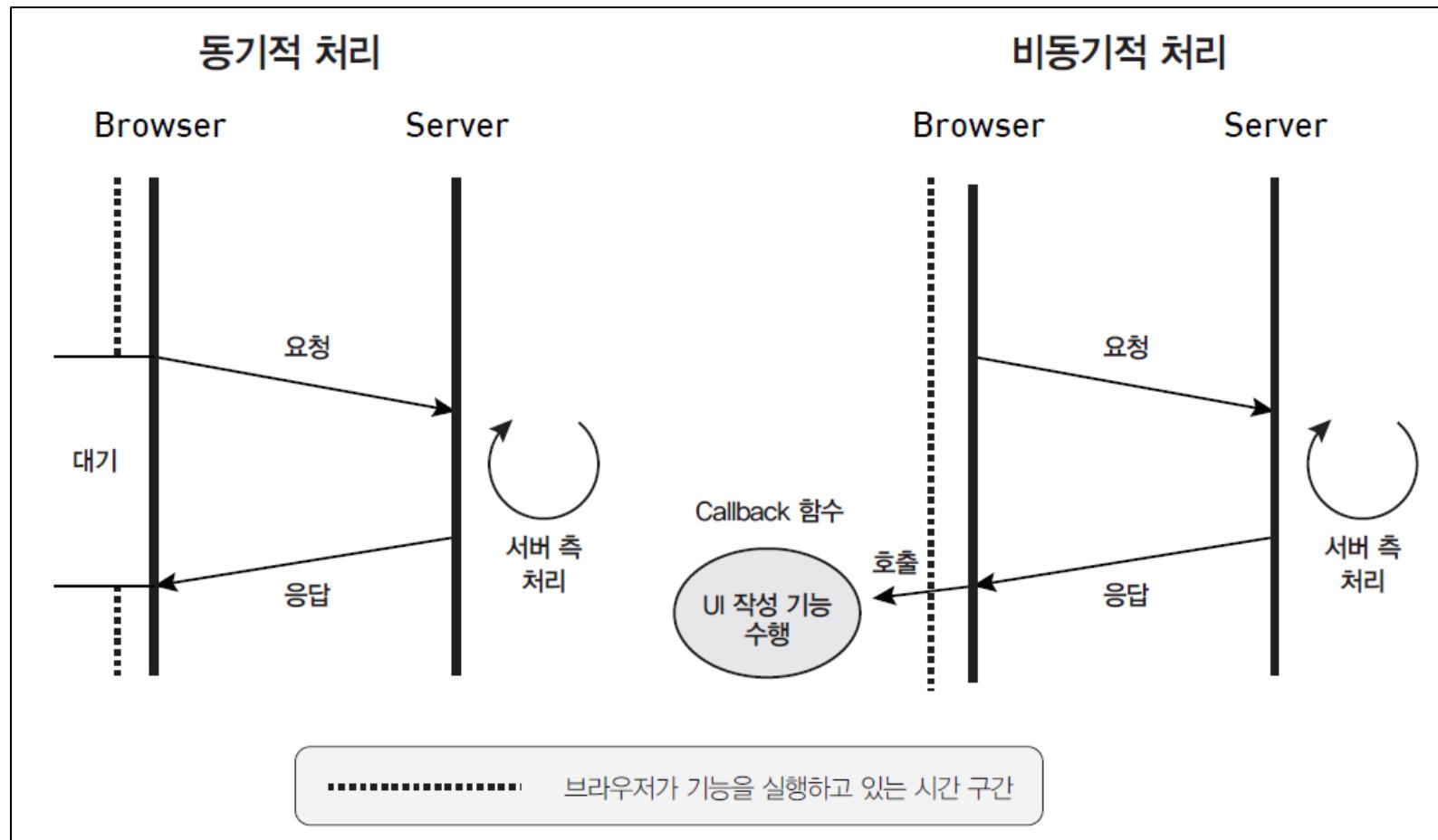
구분	JSON	XML
데이터 크기	상대적으로 작다	상대적으로 크다
가독성	좋다	좋다
웹앱 접근성	좋다	상대적으로 나쁘다
스키마 존재 여부	있지만 많이 쓰이지는 않는다. 표준화 작업이 진행 중이다.	존재하며 많이 쓰인다.

jQuery Ajax



■ Asynchronous의 의미

- 동기적 vs 비동기적

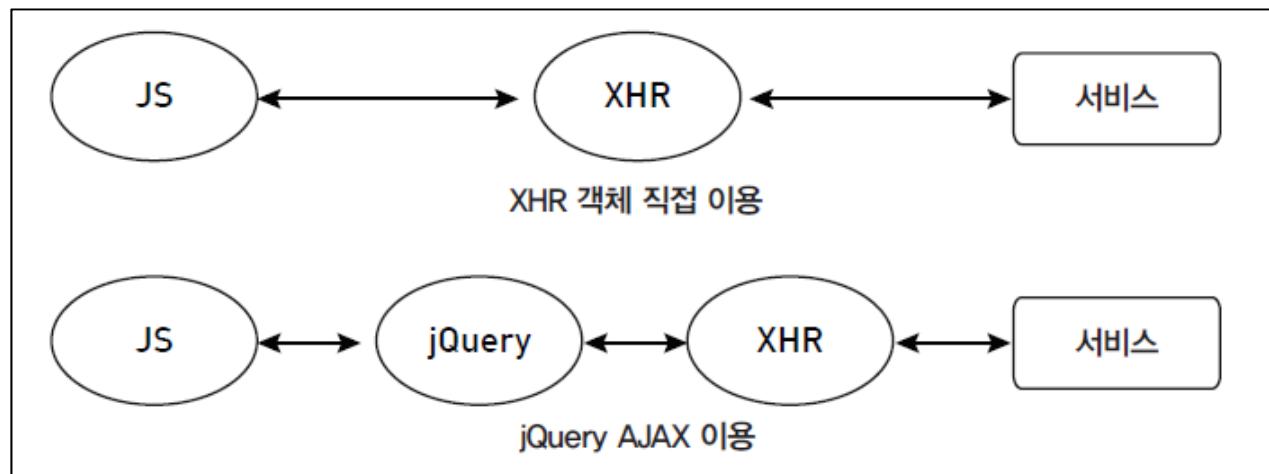


jQuery Ajax



■ XHR(XMLHttpRequest) 객체를 직접 이용하는 것은 비효율적.

- jQuery 라이브러리를 이용하는 것이 훨씬 간결하고 편리
- jQuery는 웹 브라우저 종류에 상관없이 같은 방식으로 Ajax 기능을 구현하도록 다양한 메소드를 제공(크로스 브라우징 문제 해결)
- jQuery mobile은 페이지 이동을 위해 Ajax 기술을 사용



jQuery Ajax



■ jQuery Ajax 주요 메소드

Ajax 메소드	기능/예
<code>\$.ajax()</code>	<p>모든 Ajax 메소드의 기본이 되는 메소드 예) <code>\$.ajax({ url: 'service.php', success: function(data) { \$('#area').html(data); } });</code></p> <ul style="list-style-type: none">- 데이터를 서버에 HTTP POST, GET 방식으로 전송- HTML, XML, JSON, TEXT 유형의 데이터를 요청할 수 있는 통합 메서드- <code>\$.get()</code>, <code>\$.post()</code>, <code>\$.getJSON()</code> 기능을 결합한 메소드
<code>\$.get()</code>	<p>GET 방식의 ajax() 메소드 예) <code>\$.get('sample.html', function(data) { \$('#area').html(data); });</code></p> <ul style="list-style-type: none">- 데이터를 서버에 HTTP GET 방식으로 전송한 후 서버 측에 응답 요청 받을 때 사용
<code>\$.post()</code>	<p>POST 방식의 ajax() 메소드 예) <code>\$.post('sample.html', function(data) { \$('#area').html(data); });</code></p> <ul style="list-style-type: none">- 데이터를 서버에 HTTP POST 방식으로 전송한 후 서버 측의 응답을 받을 때 사용
<code>\$.getJSON()</code>	<p>JSON 형식으로 응답 받는 ajax() 메소드 예) <code>\$.getJSON('sample.json', function(data) { \$('#area').html('<p>' + data.age + '</p>'); });</code></p> <ul style="list-style-type: none">- 데이터를 서버에 HTTP GET 방식으로 전송한 후 서버 측의 응답을 JSON 형식으로 받을 때 사용
<code>load()</code>	<p>서버로부터 데이터를 받아서 일치하는 요소 안에 HTML을 추가 예) <code>\$('#area').load('sample.html', function() { });</code></p> <ul style="list-style-type: none">- 외부 콘텐츠를 가져올 때 사용
<code>\$.getScript()</code>	<p>자바스크립트 형식으로 응답 받는 ajax() 메소드 예) <code>\$.getScript('sample.js', function() { });</code></p> <ul style="list-style-type: none">- Ajax를 이용하여 외부 자바스크립트를 불러올 때 사용
<code>\$.ajaxSetup()</code>	<p>ajax() 메소드의 선택 사항들에 대한 기본값 설정 예) <code>\$.ajaxSetup({ url: 'service.php' });</code></p>

jQuery Ajax



■ jQuery Ajax 주요 메소드

Ajax 메소드	기능
ajaxStart()	첫 번째 Ajax 요청이 시작될 때 호출되는 이벤트 메소드 예) \$('#img1').ajaxStart(function(){ \$(this).show(); });
ajaxStop()	모든 Ajax 요청이 끝날 때 호출되는 이벤트 메소드 예) \$('#img1').ajaxStop(function(){ \$(this).fadeOut(2000); });
ajaxSend()	특정 Ajax 요청을 보내기 전에 호출되는 이벤트 메소드 예) \$("#msg").ajaxSend(function(event, request, settings){ \$(this).append("<p>" + settings.url + "페이지 요청 시작</p>"); });
ajaxSuccess()	특정 Ajax 요청이 성공적으로 완료될 때마다 호출되는 이벤트 메소드 예) \$("#msg").ajaxSuccess(function(event, request, settings){ \$(this).append("<p>요청 성공</p>"); });
ajaxError()	Ajax 요청들에 대한 오류 발생시 호출되는 이벤트 메소드 예) \$("#msg").ajaxError(function(event, request, settings){ \$(this).append("<p>" + settings.url + "페이지 요청 실패</p>"); });
ajaxComplete()	Ajax 요청들이 완료되면(성공/실패 관련 없이) 호출되는 이벤트 메소드 예) \$("#msg").ajaxComplete(function(event,request, settings){ \$(this).append("<p>요청 완료</p>"); });

jQuery Ajax



■ \$.ajax() 메소드

- 제이쿼리에서 모든 Ajax 메소드는 내부적으로는 \$.ajax() 메소드를 사용(기본 메소드)
- \$.ajax() 메소드는 사용자가 지정한 URL 경로에 파일의 데이터를 전송하고, 입력한 URL 경로 파일로부터 요청한 데이터를 불러오는 기능을 수행
- 불러올 수 있는 외부 데이터로는 HTML, XML, JSON, TEXT 유형 등이 있음
- ```
$.ajax({
 url: "전송 페이지"(액션 페이지),
 type : "전송방식"(get, post 방식),
 data: "전송할 데이터",
 timeout: "응답 제한시간",
 datatype: "요청한 데이터 타입"(HTML, XML, JSON, TEXT 등 리턴타입),
 async: "비동기여부"(default: true),
 success: function(data)(성공콜백함수(data:서버의 반환값) { },
 error: function()(실패콜백함수) { }
 complete: function()(요청이 완료되었을 때 실행콜백함수) { }
});
```

# jQuery Ajax

## ■ \$.ajax() 메소드의 주요 옵션들

- **url** 요청하고자 하는 서비스 URL(기본값 : 현재 페이지).
- **type** 요청 메서드, GET, POST, PUT, DELETE 등을 지정한다(기본값 : GET).
- **async** 비동기 전송을 수행할지를 지정. 동기적인 처리를 원할 경우 이 값을 false로 지정한다(기본값 : true).
- **data** 서버로 전송하는 데이터. 문자열, 객체, 배열 형식을 지원한다. 문자열인 경우 값을 직접 URL 인코딩해야 한다.
  - ex) String : "keyword=%EC%98%A4&mode=2"
  - 객체 : { keyword:"오", mode:2 }
  - 배열 : [ { name:"keyword", value: "오" }, { name:"mode", value: "2" } ]
- **contentType** 요청 시 서버로 전달하는 데이터의 형식(기본값 : application/x-www-form-urlencoded)
- **dataType** 서버로부터 응답받기를 원하는 데이터 타입을 지정(기본값 : 자동 추정). xml, json, script, html 등을 지정할 수 있음.
- **statusCode** 서버로부터 응답의 status 코드에 따라 호출할 함수를 지정.
  - ex) \$.ajax({  
    ...  
    statusCode : {  
        404 : function() {  
            alert("요청하신 경로의 데이터가 존재하지 않습니다");  
        }  
    }  
})

■ \$.ajax(url, settings) → 리턴값 : jqXHR

■ \$.ajax(settings) → 리턴값 : jqXHR

url    요청하고자 하는 서비스의 주소

settings    자바스크립트 객체. 요청 정보, 옵션 정보 등을 포함한다.

- **timeout** 요청 제한 시간(ms). 지정한 시간 이내에 응답하지 않으면 요청을 종단시킨다.
- **headers** 요청 시에 전달할 헤더값을 객체 형태로 지정.
  - ex) \$.ajax({  
    headers : { Authorization:[credential], Accept : "application/json" }  
});
- **beforeSend** 요청을 서버로 전송하기 전에 실행될 콜백 함수를 지정. 콜백 함수의 두 번째 파라미터는 요청 시에 사용한 옵션 정보이다.
  - ex) beforeSend : function(xhr, settings) { }
- **success** 요청이 성공했을 때 실행될 콜백 함수를 지정. 콜백 함수의 첫 번째 파라미터는 수신한 데이터이다.
  - ex) success : function(data, status, xhr) { }
- **error** 요청이 실패했을 때 실행될 콜백 함수를 지정. 콜백 함수의 두 번째 파라미터는 상태 정보를 "timeout", "error", "abort"와 같이 문자열로 전달한다. 세 번째 파라미터는 에러 정보를 문자열로 전달한다.
  - ex) error : function(xhr, status, error) { }
- **complete** 요청이 완료되었을 때 실행될 콜백 함수를 지정. 요청의 성공/실패와 관계없이 항상 실행되는 콜백 함수다. 두 번째 파라미터는 상태 정보를 전달하며 "success", "error", "abort", "parseerror" 등의 문자열을 가질 수 있다.
  - ex) complete : function(xhr, status) { }

# jQuery Ajax



- `$.ajax()` 메소드를 이용하여 서버로부터 XML, JSON, HTML 형식의 데이터를 가져와서 웹 페이지의 일부를 동적으로 갱신

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="utf-8"/>
5 <meta name="viewport" content="width=device-width, initial-scale=1"/>
6 <title>jQuery Ajax</title>
7 <script src="js/jquery-3.1.1.min.js"></script>
8 <script>
9 $(document).ready(function() {
10
11 });
12
13 </script>
14 </head>
15 <body>
16 <div>
17 <div>
18 <h1>Ajax 활용 </h1>
19 </div>
20 <div>
21 <button id="btnLoad1">XML</button>
22 <button id="btnLoad2">JSON</button>
23 <button id="btnLoad3">HTML</button>
24 </div>
25 <hr>
26 <div>
27 <ul id="listArea">
28 <li id="item">데이터 로드 하기 전 ...
29
30 </div>
31 </div>
32 </body>
33 </html>
```

jqueryAjax01.html

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <stuinfo>
3 <student>
4 <studentName>이정민 </studentName>
5 </student>
6 <student>
7 <studentName>손미나 </studentName>
8 </student>
9 <student>
10 <studentName>차영호 </studentName>
11 </student>
12 <student>
13 <studentName>황민호 </studentName>
14 </student>
15 </stuinfo>
```

XML

```
1 {
2 "stuinfo" : [
3 {"grade":"1학년"},
4 {"grade":"2학년"},
5 {"grade":"3학년"},
6 {"grade":"4학년"}
7]
```

JSON

```
1 <!DOCTYPE html>
2 <html>
3 <body>
4
5 IT대학
6 경영대학
7 예술대학
8 공과대학
9
10 </body>
11 </html>
```

HTML

# jQuery Ajax



- `$.ajax()` 메소드를 이용하여 서버로부터 XML, JSON, HTML 형식의 데이터를 가져와서 웹 페이지의 일부를 동적으로 갱신

```
8⑨ <script>
9⑩ $(document).ready(function() {
10
11 // XML 파일 처리
12 $('#btnLoad1').click(function() {
13 $.ajax({
14 url: 'xml-data.xml', //전송페이지(액션페이지) - 서버 url
15 type: 'get', //전송방식
16 dataType: 'xml', //서버로부터 반환 받아올 데이터 형식
17 timeout: 10000, //응답 제한시간
18 success: function(xmlData) { //데이터 전송및요청 성공시 실행할 함수(반환데이터)
19 var tagList = ""; // 항목 저장 변수 선언
20 $(xmlData).find('student').each(function(){//반환데이터에서 항목 생성
21 tagList += "" + $(this).find('studentName').text() + "";
22 });
23 $('#listArea').empty();
24 $('#listArea').append(tagList);
25 $('#listArea').show();
26 },
27 error: function() {//전송 및 요청 실패 시 리스트뷰에 error 표시
28 $("#listArea").html("<p>Error!!</p>");
29 }
30 });
31 });
32 });

jqueryAjax01.htm-스크립트
```

# jQuery Ajax



- `$.ajax()` 메소드를 이용하여 서버로부터 XML, JSON, HTML 형식의 데이터를 가져와서 웹 페이지의 일부를 동적으로 갱신

```
33 // JSON 파일 처리
34 $('#btnLoad2').click(function() {
35 $.getJSON('json-data.json', function(jsonData) {
36 var tagList = "";
37 $.each(jsonData.stuinfo, function() {
38 tagList += "" + this.grade + "";
39 });
40 $('#listArea').empty();
41 $('#listArea').append(tagList);
42 $('#listArea').show();
43 });
44 });
45
46 // HTML 파일 처리
47 $('#btnLoad3').click(function() {
48 $('#listArea').empty();
49 //서버로부터 데이터를 받아(htmlData 저장), 메소드를 실행하는 대상 엘리먼트(#listArea)에 직접 추가
50 $('#listArea').load('html-data.html li', function(htmlData){
51 $('#listArea').append(htmlData);
52 });
53 });
54 });
55 </script>
```

jqueryAjax01.html-스크립트

- The first screenshot shows a list of student grades: 이정민, 순미나, 차영호, 황민호.
- The second screenshot shows a list of years: 1학년, 2학년, 3학년, 4학년.
- The third screenshot shows a list of university majors: IT대학, 경영대학, 예술대학, 공과대학.

# jQuery Ajax



- `$.ajax()` 메소드를 이용하여 서버로부터 XML, JSON, HTML 형식의 데이터를 가져와서 웹 페이지의 일부를 동적으로 갱신

```
33 // JSON 파일 처리
34 $('#btnLoad2').click(function() {
35 $.getJSON('json-data.json', function(jsonData) {
36 var tagList = "";
37 $.each(jsonData.stuinfo, function() {
38 tagList += "" + this.grade + "";
39 });
40 $('#listArea').empty();
41 $('#listArea').append(tagList);
42 $('#listArea').show();
43 });
44 });
45
46 // HTML 파일 처리
47 $('#btnLoad3').click(function() {
48 $('#listArea').empty();
49 //서버로부터 데이터를 받아(htmlData 저장), 메소드를 실행하는 대상 엘리먼트(#listArea)에 직접 추가
50 $('#listArea').load('html-data.html li', function(htmlData){
51 $('#listArea').append(htmlData);
52 });
53 });
54 });
55 </script>
```

jqueryAjax01.htm-스크립트

The three dialog boxes show the following data:

- Top box: "데이터 로드 하기 전 ..."
- Middle box: "이정민", "손미나", "차영호", "황민호"
- Bottom box: "1학년", "2학년", "3학년", "4학년"

//서버로부터 데이터를 받아(htmlData 저장), 메소드를 실행하는 대상 엘리먼트(#listArea)에 직접 추가

`$('#listArea').load('html-data.html li', function(htmlData){`

`$('#listArea').append(htmlData);`

`});`

`});`

`});`

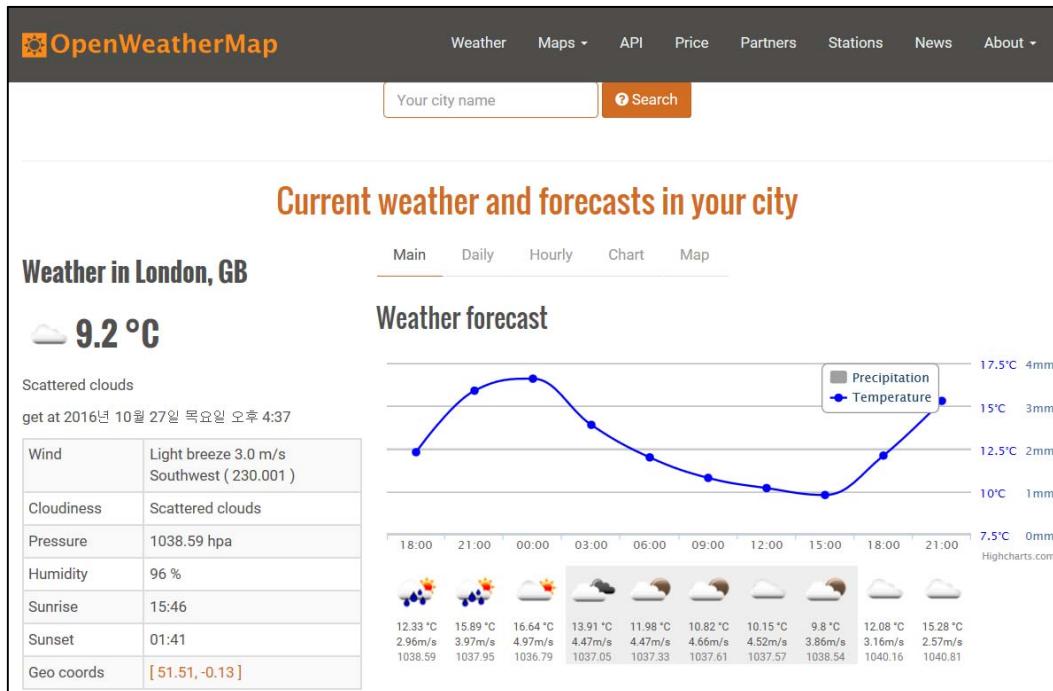
`});`

`</script>`

# jQuery Ajax 실습



- Openweathermap API를 활용한 원하는 도시의 날씨 정보 가져오기
  - Openweathermap은 전 세계 날씨 정보 관련 웹 API를 제공하며, API를 통해서 전 세계 도시의 날씨 정보를 XML이나 JSON 형태로 제공
  - URL : <http://openweathermap.org/>



# jQuery Ajax 실습



- Openweathermap API를 활용한 원하는 도시의 날씨 정보 가져오기
  - 회원 가입 후 로그인하여 API 키 발급

The screenshot shows the OpenWeatherMap homepage with a dark header. In the top right corner, there are three buttons: 'Sign In' (with a user icon), 'Sign Up' (with a user icon), and another 'Sign Up' button (with a user icon). The 'Sign Up' button in the middle is highlighted with a red box. Below the header, the main content area has a title 'OpenWeatherMap' with a sun icon. A large '회원가입' (Membership Registration) button is visible. The registration form is titled 'Create New Account' and contains fields for 'Username', 'Enter email', 'Password', and 'Repeat Password'. There is also a checkbox for agreeing to the 'Terms of Service' and 'Privacy Policy', and a 'Create Account' button.

The screenshot shows the OpenWeatherMap homepage with a dark header. In the top right corner, there are three buttons: 'Sign In' (with a user icon), 'Sign Up' (with a user icon), and another 'Sign Up' button (with a user icon). The 'Sign In' button in the middle is highlighted with a red box. Below the header, the main content area has a title 'OpenWeatherMap' with a sun icon. A large '로그인' (Login) button is visible. The login form is titled 'Sign In To Your Account' and contains fields for 'Username' (with a user icon) and 'Password' (with a lock icon). There is a 'Remember me' checkbox, a 'Submit' button, and links for 'Not registered? Create an Account.' and 'Lost your password?? Click here to recover.'

# jQuery Ajax 실습



- Openweathermap API를 활용한 원하는 도시의 날씨 정보 가져오기
  - 회원 가입 후 로그인하여 API 키 발급

The screenshot shows the OpenWeatherMap API keys generation interface. On the left, there's a sidebar with '로그인 화면' (Login screen) showing a 'Signed in successfully.' message. The main navigation bar includes 'Weather', 'Maps', 'API', 'Price', 'Partners', and 'Stations'. Below the navigation, there are tabs for 'Setup' (highlighted), 'API keys' (highlighted with a red box and circled 1), 'My Services', 'My Payments', 'Billing plans', and 'Map editor'. A 'Logout' button is on the far right.

The central area is titled 'API keys' and displays a green 'Notice' box stating 'API key was created successfully'. Below this, a message says 'NEW! You can generate as much API keys as needed for your subscription. We accumulate the total loading from all of them.' A table lists existing API keys:

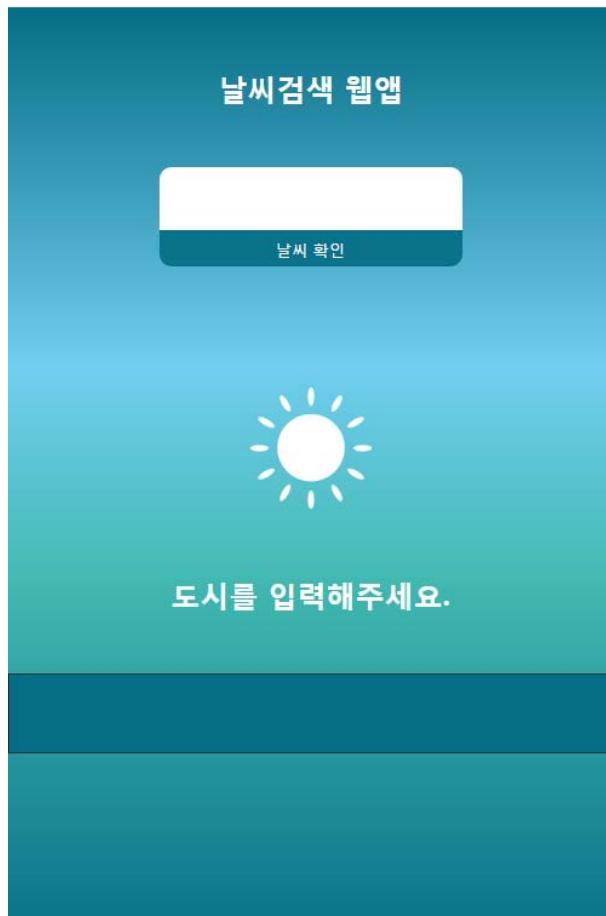
| Key                    | Name   | Actions                                                                   |
|------------------------|--------|---------------------------------------------------------------------------|
| c2bb263a34...379af1878 | ugkang | <input type="button" value="Edit"/> <input type="button" value="Delete"/> |

To the right, there's a 'Create key' form with fields for 'Name' (labeled 'name 입력') and a 'Generate' button (circled 3). A purple callout box labeled 'API Key 생성됨' points to the newly generated key in the table.

# jQuery Ajax 실습



- Openweathermap API를 활용한 원하는 도시의 날씨 정보 가져오기
  - 실행화면



# jQuery Ajax 실습



- Openweathermap API를 활용한 원하는 도시의 날씨 정보 가져오기

```
1 <!DOCTYPE html>
2 <html lang="ko">
3 <head>
4 <meta charset="utf-8">
5 <title>jQuery Ajax</title>
6 <link type="text/css" rel="stylesheet" href="css/custom.css"> </head>
7 <script src="js/jquery-3.1.1.min.js"></script>
8
9 </head>
10 <body>
11 <section class="content-wrapper">
12 <h2>날씨검색 웹앱</h2>
13 <div class="input-wrapper">
14 <input type="text">
15 <button>날씨 확인</button>
16 </div>
17
18 <div class="result-wrapper">
19
20 </div>
21
22 <h2 id="result-text">도시를 입력해주세요.</h2>
23
24 <div id="weather-info">
25 <!-- 날씨 정보 표시할 곳 -->
26 </div>
27 </section>
28
29 <!-- Script -->
30 <script type="text/javascript" src="js/custom.js"></script>
31 </body>
32 </html>
```

# jQuery Ajax 실습



- Openweathermap API를 활용한 원하는 도시의 날씨 정보 가져오기
  - URL 패턴

The screenshot shows the OpenWeatherMap website's API documentation page. The top navigation bar includes 'Support Center', a search bar ('Weather in your city'), and links for 'Sign In', 'Sign Up', and temperature units ('°C', '°F'). The 'API' tab is highlighted with a red box. Below the navigation, there's a banner with the text 'We Deliver 1 Billion' and '1,000 new subscribers a day'. A search input field 'Your city name' is also present. The main content area is titled 'OpenWeatherMap' and describes the API as simple, clear, and free. It offers plan options and requires API keys. A 'Current weather data' section is shown with a sub-section for 'By city name'. The 'Call current weather data for one location' section contains examples of API calls:

```
api.openweathermap.org/data/2.5/weather?q={city name}
api.openweathermap.org/data/2.5/weather?q={city name},{country code}
```

A red box highlights the first API call example. Below it, a gray button labeled '복사' (Copy) is shown. At the bottom of the page, a summary of the API call structure is provided:

api.openweathermap.org/data/2.5/weather?q= + city + &mode=json&units=metric + appid;

This summary is annotated with four numbered circles:

1. '도시명' (City Name)
2. '전송모드' (Transmission Mode)
3. '표시타입(섭씨)' (Display Type (Celsius))
4. 'API Key'

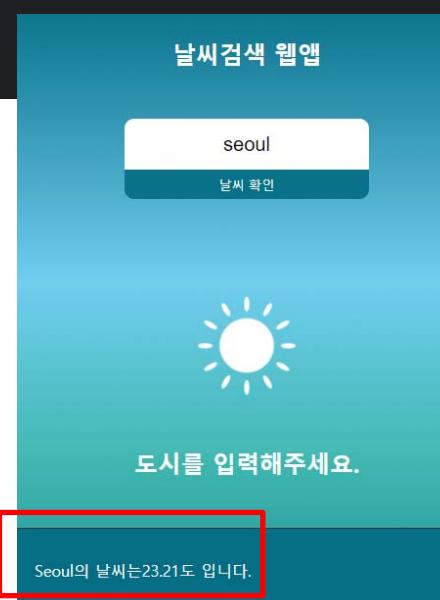
# jQuery Ajax 실습(1)



- Openweathermap API를 활용한 원하는 도시의 날씨 정보 가져오기

```
1 ▼ $(document).ready(function () {
2 ▼ $("button").click(function () {
3 var city = $("input").val();
4 var appid = "8b61cd7e8[REDACTED]lc65d3ec4";
5 var url = "http://api.openweathermap.org/data/2.5/weather?q=" +
6 city + "&mode=json&units=metric&appid=" + appid;
7
8
9 ▼ $.get(url, function(data, status) {
10 console.log(data);
11 var result = data.name + "의 날씨는" +
12 data.main.temp + "도 입니다.";
13 $("#weather-info").html(result);
14 });
15 });
16});
17});
```

```
24<div id="weather-info">
25 <!-- 날씨 정보 표시할 곳 -->
26 </div>
```



jqueryAjax02.html-스크립트

Elements Console Sources Network

Filter

날씨검색 웹앱

seoul

날씨 확인

도시를 입력해주세요.

날씨검색 웹앱

Object

- base: "stations"
- clouds: {all: 0}
- cod: 200
- coord: {lon: 126.98, lat: 37.57}
- dt: 1508565600
- id: 1835848
- main:
  - humidity: 38
  - pressure: 1017
  - temp: 23.21
  - temp\_max: 25
  - temp\_min: 22
- name: "Seoul"
- sys: {type: 1, id: 7676, message: 0.011, country: "KR", visibility: 10000}
- weather: Array(1)
  - 0:
    - description: "clear\_sky"
    - icon: "01d"
    - id: 800
    - main: "Clear"
- wind:
  - deg: 330
  - speed: 5.1

Elements Console Sources Network

Filter

날씨검색 웹앱

Object

- base: "stations"
- clouds: {all: 0}
- cod: 200
- coord: {lon: 126.98, lat: 37.57}
- dt: 1508565600
- id: 1835848
- main:
  - humidity: 38
  - pressure: 1017
  - temp: 23.21
  - temp\_max: 25
  - temp\_min: 22
- name: "Seoul"
- sys: {type: 1, id: 7676, message: 0.011, country: "KR", visibility: 10000}
- weather: Array(1)
  - 0:
    - description: "clear\_sky"
    - icon: "01d"
    - id: 800
    - main: "Clear"
- wind:
  - deg: 330
  - speed: 5.1

# jQuery Ajax 실습(2)



- Openweathermap API를 활용한 원하는 도시의 날씨 정보 가져오기
- 날씨 아이콘 정보(<https://openweathermap.org/weather-conditions>)

## How to get icon URL

For code 501 - moderate rain icon = "10d"  
URL is  
<http://openweathermap.org/img/w/10d.png>

## 날씨 아이콘 표시방법

"<http://openweathermap.org/img/w/>" + icon

### Icon list

| Day icon | Night icon | Description                      |
|----------|------------|----------------------------------|
| 01d.png  | 01n.png    | clear sky                        |
| 02d.png  | 02n.png    | few clouds                       |
| 03d.png  | 03n.png    | scattered clouds                 |
| 04d.png  | 04n.png    | broken clouds<br>overcast clouds |
| 09d.png  | 09n.png    | shower rain                      |
| 10d.png  | 10n.png    | rain                             |
| 11d.png  | 11n.png    | thunderstorm                     |
| 13d.png  | 13n.png    | snow                             |
| 50d.png  | 50n.png    | mist                             |

icon = "01d.png"



도시를 입력해주세요.

```
18<div class="result-wrapper">
19
20</div>
```

# jQuery Ajax 실습(2)



- Openweathermap API를 활용한 원하는 도시의 날씨 정보 가져오기

```
1 <!DOCTYPE html>
2 <html lang="ko">
3 <head>
4 <meta charset="utf-8">
5 <title>jQuery Ajax</title>
6 <link type="text/css" rel="stylesheet" href=".//css/custom.css">
7 <script src=".//js/jquery-3.2.1.min.js"></script>
8
9 </head>
10 <body>
11 <section class="content-wrapper">
12 <h2>날씨검색 웹앱</h2>
13 <div class="input-wrapper">
14 <input type="text">
15 <button>날씨 확인</button>
16 </div>
17
18 <div class="result-wrapper">
19
20 </div>
21
22 <h2 id="result-text">도시를 입력해주세요.</h2>
23
24 <div id="weather-info">
25 <!-- 날씨 정보 표시할 곳 -->
26 </div>
27 </section>
28
29 <!-- Script -->
30 <script type="text/javascript" src=".//js/custom2.js"></script>
31 </body>
32 </html>
```

jqueryAjax03.html

날씨검색 웹앱

seoul

날씨 확인

도시를 입력해주세요.

날씨 이미지 (Clear Sky)

Country: KR  
City: Seoul  
Current Temperature: 22.99 C  
Current Humidity: 29 %  
Current Wind Speed: 2.6 m/s  
Weather Conditions: clear sky

# jQuery Ajax 실습(2)



- Openweathermap API를 활용한 원하는 도시의 날씨 정보 가져오기

```
1 $(document).ready(function () {
2 $("button").click(function () {
3 var city = $("input").val();
4 var appid = "8b61cd7e86[REDACTED]5d3ec4";
5 var url = "http://api.openweathermap.org/data/2.5/weather?q=" +
6 city + "&mode=json&units=metric&appid=" + appid;
7
8 $.ajax({
9 url : url,
10 dataType : "jsonp",
11 type : "GET",
12 success : function(data) {
13 console.log(data);
14 var icon;
15 switch (data.weather[0].description) {
16 case "clear sky":
17 icon = "01d.png";
18 break;
19 case "few clouds":
20 icon = "02d.png";
21 break;
22 case "scattered clouds":
23 icon = "03d.png";
24 break;
25 case "broken clouds":
26 case "overcast clouds":
27 icon = "04d.png";
28 break;
29 case "shower rain":
30 case "drizzle":
31 case "drizzle rain":
32 case "shower drizzle":
33 icon = "09d.png";
34 break;
35 case "light rain":
36 case "moderate rain":
37 case "extreme rain":
38 icon = "10d.png";
39 break;
40 }
41 }
42 case "thunderstorm":
43 icon = "11d.png";
44 break;
45 case "snow":
46 case "freezing rain":
47 case "heavy snow":
48 case "shower snow":
49 icon = "13d.png";
50 break;
51 case "mist":
52 case "smoke":
53 case "haze":
54 case "fog":
55 icon = "50d.png";
56 break;
57 default:
58 icon = "01d.png";
59 break;
60 }
61 }
62 $("#result-image").attr("src", "http://openweathermap.org/img/w/" + icon);
63
64 var w_info = 'Country: ' + data.sys.country + '
';
65 w_info += 'City: ' + data.name + '
';
66 w_info += 'Current Temperature: ' + data.main.temp + ' C
';
67 w_info += 'Current Humidity: ' + data.main.humidity + ' %
';
68 w_info += 'Current Wind Speed: ' + data.wind.speed + ' m/s
';
69 w_info += 'Weather Conditions: ' + data.weather[0].description + '
';
70
71 $("#weather-info").html(w_info);
72
73 error : function(error) {
74 alert("Error!");
75 },
76 complete : function() {
77 alert("Complete!");
78 });
79 });

```

jqueryAjax03.html – 스크립트

날씨검색 웹앱

seoul

날씨 확인

도시를 입력해주세요.

Country: KR  
City: Seoul  
Current Temperature: 22.99 C  
Current Humidity: 29 %  
Current Wind Speed: 2.6 m/s  
Weather Conditions: clear sky

# jQuery Ajax 실습(3)



- Openweathermap API를 활용한 원하는 도시의 날씨 정보 가져오기

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="utf-8">
5 <title>jQuery Ajax</title>
6 <meta name="viewport" content="width=device-width,initial-scale=1.0,minimum-scale=1.0,maximum-
 scale=1.0,user-scalable=no">
7 <link type="text/css" rel="stylesheet" href=".//css/custom2.css">
8 <script src=".//js/jquery-3.2.1.min.js"></script>
9 </head>
10
11 <body>
12 <div id="weather_info">
13 <!-- 도시명 -->
14 <h1 class="city"></h1>
15
16 <section>
17 <p class="w_id"></p>
18 <figure class="icon"></figure>
19
20 <p class="temp"></p>
21 <aside>
22 <p class="temp_max">max</p>
23 <p class="temp_min">min</p>
24 </aside>
25 </section>
26
27
28 </div>
29
30 <!-- Script -->
31 <script type="text/javascript" src=".//js/custom3.js"></script>
32 </body>
33
34 </html>
```

jqueryAjax04.html



# jQuery Ajax 실습(3)



- Openweathermap API를 활용한 원하는 도시의 날씨 정보 가져오기

```
1 // API 요청 URL 변수
2 var appid = "8b61cd7e8[REDACTED]3ec4";
3 var url = "http://api.openweathermap.org/data/2.5/weather?q=seoul&appid=" + appid;
4
5 // 로딩 이미지 표시
6 $('#weather_info .load_img').show();
7
8 $.getJSON(url, function (data) {
9 console.log(data);
10
11 // 날씨 데이터 객체
12 var sys = data.sys; // 국가명, 일출/일몰
13 var city = data.name; // 도시명
14 var country = sys.country; // 국가명
15 var weather = data.weather; // 날씨 객체
16 var main = data.main; // 온도 기압 관련 객체
17
18 var wmain = weather[0].main; // 구름 상태(Cloudiness)
19 var w_id = weather[0].id; // 날씨상태 id 코드
20 var icon = weather[0].icon; // 날씨 아이콘 정보
21
22 var temp = main.temp; // 현재 온도
23 var temp_min = main.temp_min // 최저 온도
24 var temp_max = main.temp_max // 최고 온도
25
26 //일출시간
27 var sunrise = sys.sunrise;
28 var date = new Date(sunrise * 1000);
29 var sunrisetimestr = date.toLocaleDateString();
30 console.log('일출시간 : ' + date, sunrisetimestr);
31
32
```

jqueryAjax04.html – 스크립트

```
33 //일몰시간
34 var sunset = sys.sunset;
35 var date = new Date(sunset * 1000);
36 var sunsettimestr = date.toLocaleDateString();
37 console.log('일몰시간 : ' + date, sunsettimestr);
38
39 // 날씨 아이콘
40 var icon_url = 'http://openweathermap.org/img/w/' + icon;
41
42 // 날씨 정보 표시
43 $('#weather_info > .city').html(city + "/" + country);
44 $('#weather_info .w_id').html(wmain);
45 $('#weather_info .icon').html("");
46
47 //온도의 기본값은 국제단위인 캠빈값으로 반환됨으로 섭씨(-273.15)로 변환해야함
48 $('#weather_info .temp').html(parseInt(temp - 273.15) + '°');
49 $('#weather_info .temp_min').html(parseInt(temp_min - 273.15) + '°' + ' min');
50 $('#weather_info .temp_max').html(parseInt(temp_max - 273.15) + '°' + ' max');
51
52 // 데이터 로딩 후 로딩이미지 제거
53 $('#weather_info .load_img').hide();
54
55 } // end getJSON()
56
57 .fail(function () {
58 // 오류 메시지
59 alert("Loding error");
60
61});
```



# jQuery Ajax 실습(3)



- Openweathermap API를 활용한 원하는 도시의 날씨 정보 가져오기
- API respond

```
Seoul/KR
Clear
24° 24° max
24° min

Elements Console Sources
top Filter

{
 "coord": {...},
 "weather": [
 {
 "base": "stations",
 "clouds": {"all": 1},
 "cod": 200,
 "coord": {"lon": 126.98, "lat": 37.57},
 "dt": 1508568900,
 "id": 1835848,
 "main": {
 "humidity": 29,
 "pressure": 1017,
 "temp": 297.15,
 "temp_max": 297.15,
 "temp_min": 297.15
 },
 "sys": {
 "country": "KR",
 "id": 7673,
 "message": 0.0094,
 "sunrise": 1508535962,
 "sunset": 1508575604,
 "type": 1
 }
 }
],
 "base": "stations",
 "weather": [
 {
 "description": "clear sky",
 "icon": "01d",
 "id": 800,
 "main": "Clear"
 }
],
 "main": {
 "temp": 297.15,
 "temp_max": 297.15,
 "temp_min": 297.15
 },
 "wind": {
 "speed": 2.6,
 "deg": 20
 }
}
```

Parameters:

- coord
  - coord.lon City geo location, longitude
  - coord.lat City geo location, latitude
- weather (more info Weather condition codes)
  - weather.id Weather condition id
  - weather.main Group of weather parameters (Rain, Snow, Extreme etc.)
  - weather.description Weather condition within the group
  - weather.icon Weather icon id
- base Internal parameter
- main
  - main.temp Temperature. Unit Default: Kelvin, Metric: Celsius, Imperial: Fahrenheit.
  - main.pressure Atmospheric pressure (on the sea level, if there is no sea\_level or grnd\_level data), hPa
  - main.humidity Humidity, %
  - main.temp\_min Minimum temperature at the moment. This is deviation from current temp that is possible for large cities and megalopolises geographically expanded (use these parameter optionally). Unit Default: Kelvin, Metric: Celsius, Imperial: Fahrenheit.
  - main.temp\_max Maximum temperature at the moment. This is deviation from current temp that is possible for large cities and megalopolises geographically expanded (use these parameter optionally). Unit Default: Kelvin, Metric: Celsius, Imperial: Fahrenheit.
  - main.sea\_level Atmospheric pressure on the sea level, hPa
  - main.grnd\_level Atmospheric pressure on the ground level, hPa
- wind
  - wind.speed Wind speed. Unit Default: meter/sec, Metric: meter/sec, Imperial: miles/hour.
  - wind.deg Wind direction, degrees (meteorological)
- clouds
  - clouds.all Cloudiness, %
- rain
  - rain.3h Rain volume for the last 3 hours
- snow
  - snow.3h Snow volume for the last 3 hours
- dt Time of data calculation, unix, UTC
- sys
  - sys.type Internal parameter
  - sys.id Internal parameter
  - sys.message Internal parameter
  - sys.country Country code (GB, JP etc.)
  - sys.sunrise Sunrise time, unix, UTC
  - sys.sunset Sunset time, unix, UTC
- id City ID
- name City name
- cod Internal parameter

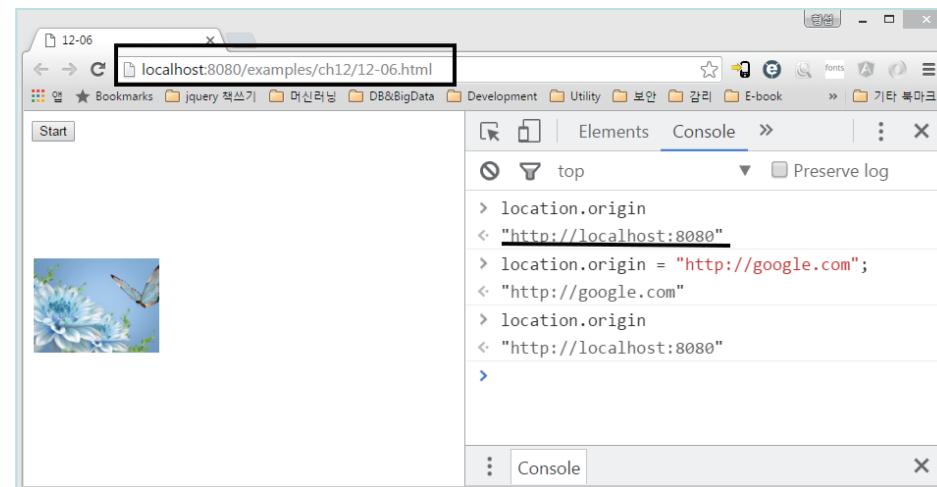
# 크로스 도메인 문제란?



## ■ 정의

- "동일 근원 정책(SOP : Same Origin Policy)이라는 웹브라우저의 보안 정책으로 인해, 현재 브라우저와 동일한 오리진(Origin)이 아닌 다른 오리진으로부터 AJAX를 이용해 데이터를 로드하지 못한다."

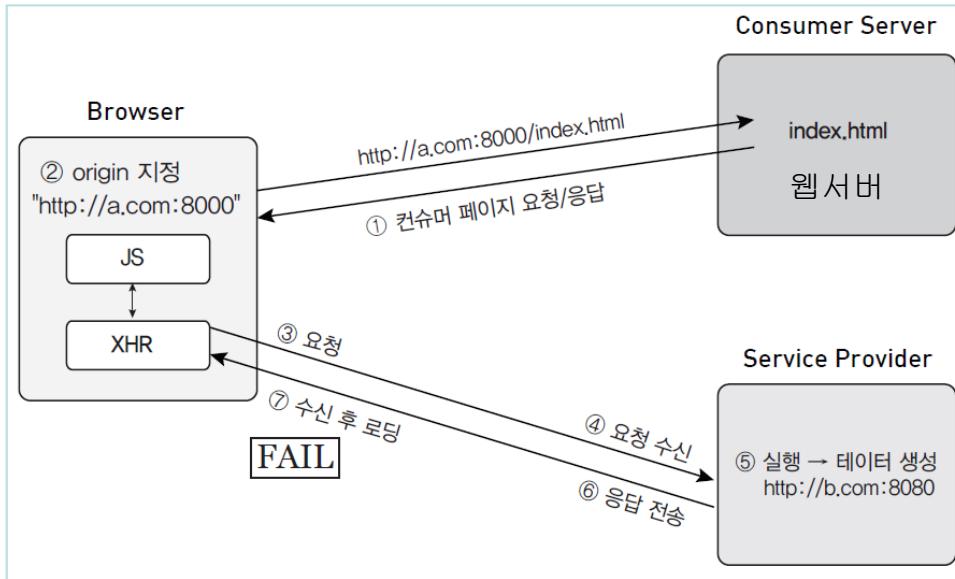
- 오리진이란?
- 브라우저상에 현재 보여지고 HTML 문서가 어디에서 가져온 것인지를 나타내는 문자열 정보.(브라우저의 콘솔에서 직접 확인 가능)
- 오리진은 직접 수정 불가



# 크로스 도메인 문제란?



## ■ 크로스 도메인 문제 개념도



- 1) 웹 브라우저에서 컨슈머 서버에 요청하여 HTML 페이지가 브라우저에 로딩되면,
- 2) 이때 브라우저의 오리진이 지정됨
- 3) 자바스크립트 코드와 XHR객체를 이용해 서비스 프로바이더에게 데이터 요청
- 4) 서비스 프로바이더는 요청을 수신 한 후,
- 5) 브라우저가 요청한 데이터를 생성하여,
- 6) 응답을 전송한다.
- 7) 그러나 웹 브라우저는 수신한 데이터를 브라우저에 로딩할 수 없다.(오리진이 다르기 때문)

- 크로스 오리진 상황이라도 요청~응답전송까지는 정상 수행
- 마지막 단계에 브라우저가 수신 데이터를 로드하지 않음
  - 통신할 수 없다(X) 통신은 하지만 데이터를 로드하지 않음(O)

# 크로스 도메인 문제란?



## ■ 예제 15-01 : 크로스 도메인 문제 발생

[예제 15-01]

```
01: <script src="http://code.jquery.com/jquery-3.1.0.js"></script>
02: <script type="text/javascript">
03: $(document).ready(function() {
04: var url = http://127.0.0.1:8080/contact/list.jsp;
05: $.get(url, function(data) {
06: console.log(data);
07: });
08: });
09: </script>
```

The screenshot shows a web browser window titled '15-01' with the URL 'localhost:8080/examples/ch15/15-01.html'. The page content says 'console을 확인합니다.' (Check the console). In the developer tools, the 'Console' tab is selected, showing the following error message:

```
XMLHttpRequest cannot load http://127.0.0.1:8080/contact/list.jsp. No 'Access-Control-Allow-Origin' header is present on the requested resource. Origin 'http://localhost:8080' is therefore not allowed access.
```

A red arrow points from the line 'var url = http://127.0.0.1:8080/contact/list.jsp;' in the code block above to the error message in the browser's console.

## 크로스 도메인 문제란?



### ■ 동일 근원 정책(SOP:Same Origin Policy)는 왜?

- 컨슈머로부터 내려받은 자바스크립트 코드가 사용자가 알지 못한 상태에서 다른 오리진으로부터 출처를 알 수 없는 데이터나 악성 코드를 내려받지 못하게 막기 위한 목적으로 적용된 것
- 하지만 지금은 크로스 도메인 간의 데이터 요청/응답이 꼭 필요함
- 크로스 도메인 문제를 극복할 수 있는 방법이 생겨났음

# 크로스 도메인 문제 해결 방법



## ■ 크로스 도메인 문제를 해결

- 문제 해결을 위해 브라우저의 옵션을 설정할 수 있지만 모든 사용자에게 설정을 강요할 수는 없음
- 브라우저의 설정을 변경하지 않고도 해결할 수 있는 방법이 필요

## ■ 크로스 도메인 문제 해결 방법

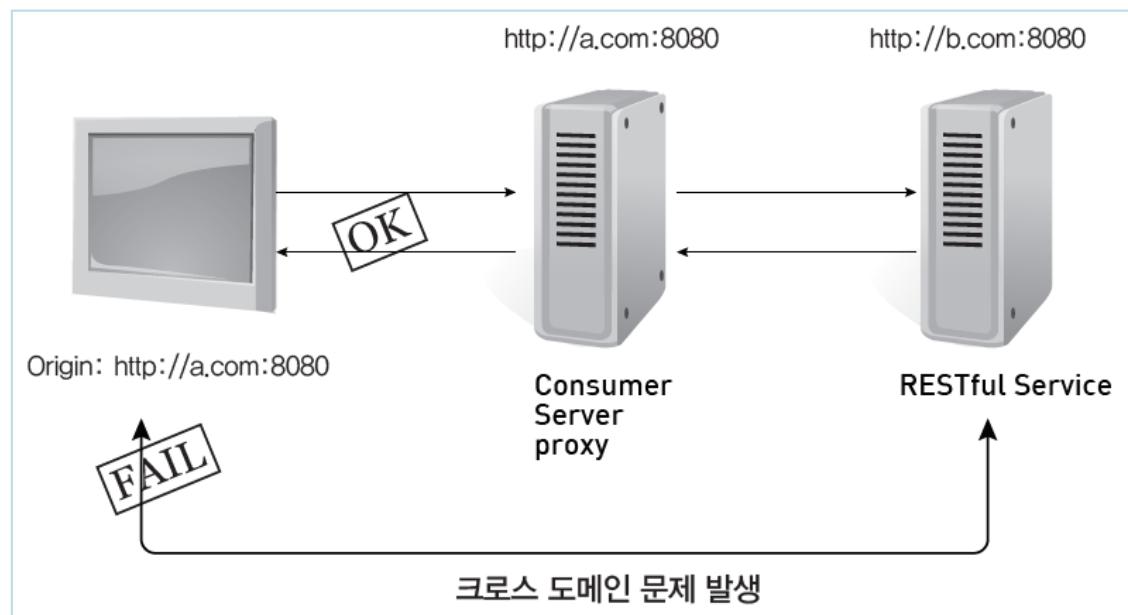
- 컨슈머측 해결 방법
  - 컨슈머 서버 프록시
- 프로바이더측 해결 방법
  - CORS(Cross Origin Resource Sharing)
  - JSONP(JSON Padding)

# 크로스 도메인 문제 해결 방법



## ■ 컨슈머 서버 프록시

- 웹 브라우저(클라이언트)가 자신을 통해서 다른 네트워크 서비스에 간접적으로 접속할 수 있게 해주는 프록시 서버를 이용하는 방법
  - 브라우저가 요청하는 경로를 컨슈머 서버의 프록시를 향하도록 하고 프록시가 서비스 쪽으로 대신 요청하고 응답 데이터가 수신되면 브라우저로 전달해주는 방법
  - 컨슈머 앱 개발자가 직접 프록시 요소를 작성해야 하는 불편함이 있음.

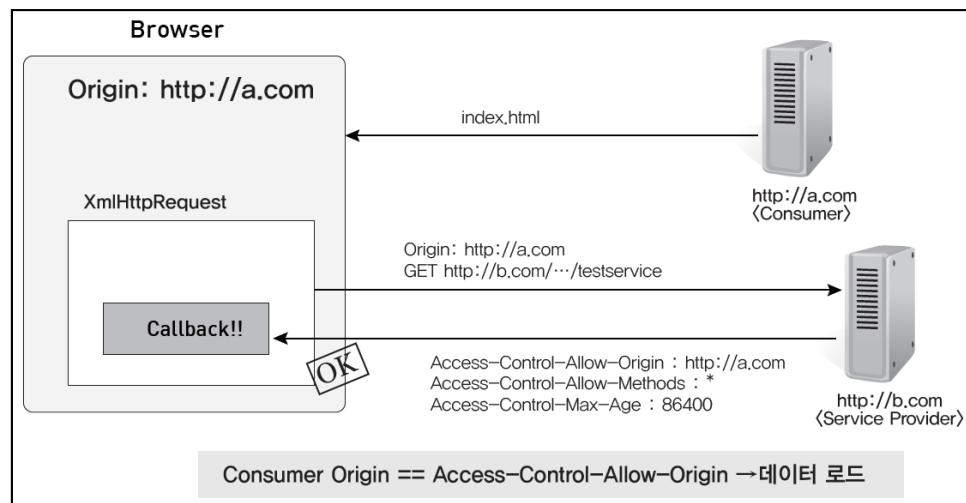


# 크로스 도메인 문제 해결 방법



## ■ CORS(Cross Origin Resource Sharing)

- "서비스 제공자(Service Provider)가 허락하는 경우에 브라우저가 데이터를 로드할 수 있도록 한다."
  - 서비스 제공자가 Access-Control-Allow-Origin 헤더로 브라우저의 오리진을 지정해 응답하면 브라우저가 거부하지 않고 데이터를 로드 할 수 있도록 하는 방법



- 1) 웹 브라우저는 http://a.com 서버에서 웹 페이지를 로딩(오리진: http://a.com)
- 2) 웹 브라우저는 Ajax로 http://b.com에 정보를 요청할 때, Origin HTTP 헤더를 추가하여 요청
- 3) http://b.com 서버는 요청 정보 중 Origin 헤더값을 알아 내어 등록된 오리진 인지를 확인 한 후,
- 4) 응답 헤더에 “Access-Control-Allow-Origin” 헤더 값으로 웹 브라우저의 오리진을 지정하여 응답
- 5) 웹 브라우저에서는 응답된 데이터를 확인하여 “Access-Control-Allow-Origin” 헤더 값이 자신의 오리진 일치하면 데이터를 로딩함

# 크로스 도메인 문제 해결 방법



## ■ CORS(Cross Origin Resource Sharing)

- 웹 브라우저의 요청을 받은 서버는 웹 브라우저에서 보내온 데이터에서 Origin 헤더 값을 읽어내어, 등록된 오리진 인지 확인하고,
- 일치하면 Access-Control-Allow-Origin 헤더를 응답 데이터에 실어 전송
- 웹 브라우저는 응답된 데이터를 확인하여, 자신의 오리진과 Access-Control-Allow-Origin 헤더 값을 비교해 일치하면 데이터를 로드 함

```
27: //사전에 등록된 오리진은 http://localhost:8080
28: String origin = request.getHeader("origin");
29: if (origin.equals("http://localhost:8080")) {
30: response.setHeader("Access-Control-Allow-Origin", origin);
31: }
```

# 크로스 도메인 문제 해결 방법



## ■ JSONP(JSON Padding)

- `$.ajax()`

- `dataType` 옵션을 `jsonp`로 지정
- `jsonp` 옵션을 `callback` 파라미터 이름으로 지정.
- `success` 콜백 함수를 지정하여 응답결과 수신

```
09: var url1 = "http://127.0.0.1:8080/contact/list_jsonp.jsp";
10: $.ajax({
11: url : url1,
12: type : "GET",
13: data : { pageno:1, pagesize:3 },
14: dataType : "jsonp",
15: jsonp : "callback",
16: success : function(data) {
17: console.log(data);
18: }
19: });
20:
21: var url2 = "http://127.0.0.1:8080/contact/list_jsonp.jsp?callback=?"
22: $.getJSON(url2, { pageno:2, pagesize:3 }, function(data) {
23: console.log(data)
24: });

```



## ■ jsRender 플러그인

- 최근 웹 개발에 있어서 Ajax 기술의 활용과 SPA(Single Page web Application)성격의 웹 사이트가 점점 더 많이 대두됨에 따라 HTML DOM의 조작과 로직을 구성하는데 많은 시간 할애
- 또한, 현재 화면의 데이터를 갱신하고 추가하는 작업에 있어 기존 서버단에서 구성하던 로직을 클라이언트에서 구성하는 것이 더 효과적인 경우가 많아, 데이터를 이용한 동적인 UI 작업이 많아짐
- 그러나 데이터를 이용해 UI로 나타내는 작업은 쉽지 않음
  - 문자열 이어붙이기 : 유지 보수성과 가독성이 나쁨

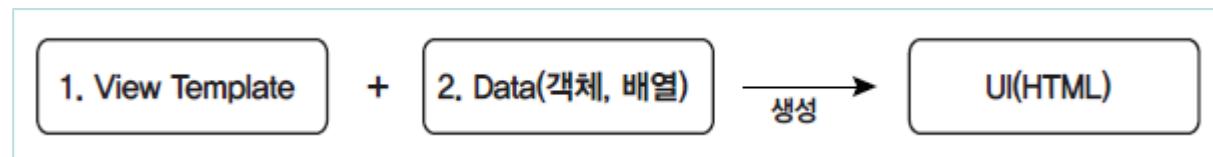
```
var str = "<div style='width:" + option.width +"px; height:" + option.height +
 "px; color:" + option.color + "; background-color:" + option.bgColor + "'>" +
 "이름 : " + data.name + "
" +
 "나이 : " + data.age + "
" +
 "이메일 : " + data.email + "
" +
 "</div>";
$("#result").html(str);
```

# jQuery 플러그인



## ▪ jsRender 플러그인

- 특히, 기존 테이블에 대량의 데이터를 추가/삭제/갱신하거나 테이블 자체를 삭제하고 새로운 테이블을 추가하는 작업 등 정형화된 템플릿 안에 데이터를 갱신/추가하는 작업이 많음
- 이런 작업인 경우 AJAX를 통해 데이터를 JSON이나 XML형식으로 서버로부터 데이터를 수신하고, 해당 데이터를 JsRender 템플릿을 이용하여 기존 HTML,DOM에 추가하는 방법으로 개발하는 것이 효과적임
- jsRender는 모델과 뷰를 분리하여 개발할 수 있는 기능 제공



# jQuery 플러그인



## ▪ jsRender 플러그인

- URL : <http://www.jsviews.com/#jsrender>
- jsRender.js 파일을 다운로드하여 폴더에 저장

The screenshot shows the JsViews website's download page. At the top, there are three main components: JsRender (with a red hexagon logo), JsViews (with a blue hexagon logo), and JsObservable (with a green hexagon logo). Below them is a search bar and a navigation menu with links like Home, Get Started, JsRender API, JsViews API, JsObservable API, Samples, Download, and Community.

In the center, there's a dark banner with the text "Best-of-breed templating" and "Simple and intuitive, powerful and extensible, lightning fast". To the right of this banner is a "Download" button, which is highlighted with a red box.

On the left, there's a code editor showing some JSON-like data:

```
Here's a first example of the power and
simplicity of JsRender templates:
Some data:
[{
 "name": "Robert",
 "nickname": "Bob",
 "showNickname": true
},
{
 "name": "Susan",
 "nickname": "Sue",
 "showNickname": false
}]
```

To the right of the code editor is a "Try it" button and a preview area. Below these are two buttons: "How It Works" and "Code".

On the right side of the page, there's a large box titled "JsRender and JsViews Downloads". It contains the following text and links:

**JsRender (jsrender.js) – rendering templates in the browser**

**Latest version** (To download, right-click and select “Save as...” from the menu):

- Uncompressed (for development): [jsrender.js](#)
- Compressed (for production): [jsrender.min.js](#). (Source map available [here](#))

JsRender is also available:

- on CDN at [cdnjs.com/libraries/jsrender](#)
- using [Bower](#) to install on the file system: `$ bower install jsrender`

41

# jQuery 플러그인



## ■ jsRender 플러그인

### ▪ JsRender 템플릿의 일반적인 사용 방법

1. 템플릿(HTML 과 JsRender에서 제공하는 템플릿 태그)을 정의
2. 템플릿에 사용할 데이터를 로드
3. JsRender의 제공 함수로 템플릿에 데이터를 렌더링 하여 HTML 을 출력
4. 렌더링에 의해 출력된 HTML을 해당 위치에 삽입

```
<!DOCTYPE html>
<html>
 <head>
 <meta charset = "utf-8" />
 <title> new document </title>
 <script src="http://code.jquery.com/jquery-3.1.1.js"></script>
 <script src="js/jsrender.js"></script>
 </head>
 <body>
 <div id="result"></div>
 <!-- (1) 템플릿(HTML 과 JsRender에서 제공하는 템플릿 태그)을 정의 -->
 <script id="theTmpl" type="text/x-jsrender">
 <div>
 Name: {{:name}}
 {{if showNickname && nickname}}
 (Goes by {{:nickname}})
 {{/if}}
 </div>
 </script>
```

Name: Robert (Goes by Bob)  
Name: Susan

```
<script>
 // (2) 템플릿에 사용할 데이터를 로드
 var data = [
 {
 "name": "Robert",
 "nickname": "Bob",
 "showNickname": true
 },
 {
 "name": "Susan",
 "nickname": "Sue",
 "showNickname": false
 }
];
 // (3) JsRender의 제공 함수로 템플릿에 데이터를 렌더링 하여 HTML 을 출력
 var template = $.templates("#theTmpl");
 var htmlOutput = template.render(data);

 // (4) 렌더링에 의해 출력된 HTML을 해당 위치에 삽입
 $("#result").html(htmlOutput);
</script>
</body>
</html>
```

# jQuery 플러그인



address.json

## jsRender 플러그인 예제

```
1 <!DOCTYPE html>
2 <html lang="ko">
3 <head>
4 <meta charset="utf-8">
5 <title>jQuery Plugin jsRender</title>
6 <style>
7 table { width: 400px; border:1px solid black; border-collapse:collapse; }
8 td, th { border:1px solid black; text-align:center; }
9 table > thead > tr { color:yellow; background-color: purple; }
10 </style>
11 <script type="text/javascript" src="https://code.jquery.com/jquery-3.1.0.js"></script>
12 <script src="https://www.jsviews.com/download/jsrender.js"></script>
13 <script type="text/javascript">
14 $(document).ready(function() {
15 //jsRender 플러그인을 사용한 방법
16 $.get("address.json", function(data) { // (2) 템플릿에 사용할 데이터 로드
17 var tmpl = $.templates("#contact_template"); // 정의한 템플릿을 변수에 저장
18 var str = tmpl.render(data.studentAddress); // (3) render() 메서드로 템플릿에 데이터 랜더링 (HTML DOM 객체)
19 console.log(str);
20 $("#container").html(str); // (4) 랜더링에 의해 생성한 HTML DOM 객체를 해당 위치에 삽입
21 });
22 });
23 </script>
24
25 <!--(1) jsRender 템플릿 정의 -->
26 <script id="contact_template" type="text/x-jsrender">
27 <tr>
28 <td>{:no}</td>
29 <td>{:name}</td>
30 <td>{:tel}</td>
31 <td>{:address}</td>
32 </tr>
33 </script>
34 </head>
35 <body>
36 <table id="list">
37 <thead>
38 <tr><th>번호</th><th>이름</th><th>전화번호</th><th>주소</th></tr>
39 </thead>
40 <!-- 학생 주소록 출력 요소 -->
41 <tbody id="container">
42
43 </tbody>
44 </table>
45 </body>
46 </html>
```

```
1 {"studentAddress": [
2 {"no":"1", "name":"손흥민", "tel":"010-1234-1224", "address":"서울"},
3 {"no":"2", "name":"김연아", "tel":"010-2222-2222", "address":"인천"},
4 {"no":"3", "name":"이영결", "tel":"010-3333-3333", "address":"부산"},
5 {"no":"4", "name":"최미나", "tel":"010-4444-4444", "address":"대구"},
6 {"no":"5", "name":"이하니", "tel":"010-5555-5555", "address":"광주"}
7]
8 }
```

번호	이름	전화번호	주소
1	손흥민	010-1234-1224	서울
2	김연아	010-2222-2222	인천
3	이영결	010-3333-3333	부산
4	최미나	010-4444-4444	대구
5	이하니	010-5555-5555	광주

```
/* //jsRender 플러그인을 사용하지 않은 경우
$.ajax({
 url:"address.json", //데이터를 요청할 URL 주소
 dataType:"json", //데이터 타입 설정
 success:function(data){ //데이터 성공적으로 요청되었을 때...
 var str = "<tr>";
 $.each(data.studentAddress, function(i,dat){
 //일반적인 문자열 결합 방법
 str += "<td>" + this.no + "</td>" +
 "<td>" + this.name + "</td>" +
 "<td>" + this.tel + "</td>" +
 "<td>" + this.address + "</td>" +
 "</tr>";

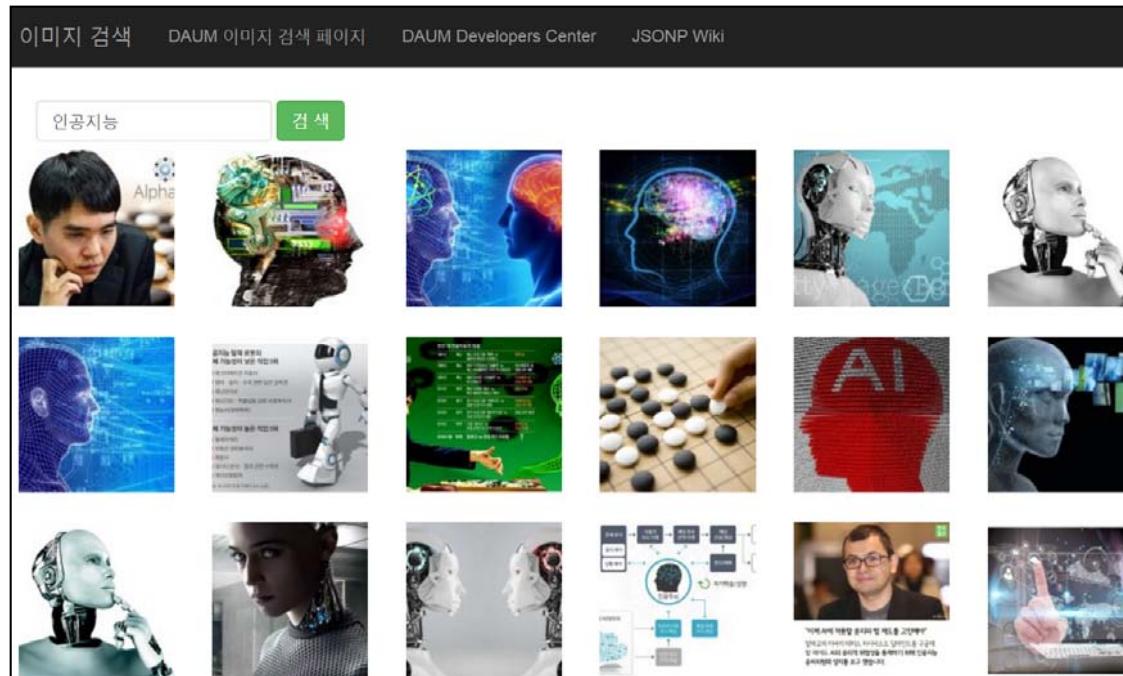
 });
 $("#container").html(str);
 });
});*/
*/
```

# jQuery 플러그인



## ▣ 다음 카카오의 이미지 검색 API를 이용한 이미지 뷰어 만들기

- 부트스트랩과 jQuery 결합
- 부트스트랩
- JSONP 기법 사용



# jQuery 플러그인



## ■ 이미지뷰어 애플리케이션 작성

- 다음 카카오의 이미지 검색 API를 이용한 이미지 뷰어 어플리케이션을 작성
- 다음 카카오 이미지 검색 Open API를 이용하려면 Daum 개발자 센터에서 API키를 받아야 함
- URL : <http://developers.daum.net/> → <https://developers.kakao.com/>

The screenshot shows the Daum Developers website. A modal window is open, stating: "앞으로 보다 나은 서비스 제공을 위해 2017년 7월 18일부터는 Kakao API 및 Daum API가 Kakao Developers로 통합되어 제공됩니다." Below the modal, there's a note: "추후 별도의 공지가 있을 때까지 Daum Developers의 기존 회원/기존 앱에 대한 구 Daum API 서비스는 계속 유지되지만, 신규 회원 및 신규 앱 생성은 이제 Kakao Developers를 이용하셔야 합니다." At the bottom of the modal, it says "하루 동안 다시 보지 않기 X".

Tip: You can use various functions such as API key management, API Quota settings, usage statistics, and API status checks across different devices.

The screenshot shows the Kakao Developers website. It features a dark background with several yellow icons arranged in a grid. From top-left to bottom-right, the icons are: 카카오개정 로그인 (KakaoOpen Login), 카카오팝크 (KakaoTalk), 나에게 보내기 (Send to Me), 카카오내비 (KakaoNavi), 푸시 알림 (Push Notification), 데브로그 분석 (DevBlog Analysis), 검색 (Search), and 음성 (Voice).

# jQuery 플러그인



## ■ 이미지 뷰어 어플리케이션

### ■ 실행 화면

The image displays two side-by-side screenshots of a web-based photo viewer application. Both screenshots show a dark header bar with navigation links: '이미지 검색' (Image Search), 'DAUM 이미지 검색 페이지' (DAUM Image Search Page), 'DAUM Developers Center', and 'JSONP Wiki'. The main content area contains a search input field containing '검색어' (Search term) and a green '검 색' (Search) button.

The left screenshot shows a large blue button labeled '다음 20개 가져오기' (Load next 20 items). The right screenshot shows a search input field containing '가을' (Autumn) and a green '검 색' (Search) button. Below the search input is a blue button labeled '다음 20개 가져오기' (Load next 20 items).

# jQuery 플러그인



## ■ 이미지 뷰어 어플리케이션

### ■ 실행 화면

이미지 검색 DAUM 이미지 검색 페이지 DAUM Developers Center JSONP Wiki

단풍 | 검색

가을 단풍 2

다음 20개 가져오기

# jQuery 플러그인



## ■ 이미지 뷰어 어플리케이션

jqueryplugin\_imgviewer.html

```
1 <!DOCTYPE html>
2 <html lang="">
3 <head>
4 <meta charset="utf-8">
5 <meta http-equiv="X-UA-Compatible" content="IE=edge">
6 <meta name="viewport" content="width=device-width, initial-scale=1">
7 <title>다음 카카오 이미지 검색 API를 이용한 이미지 뷰어</title>
8 <link rel="stylesheet" href="imgviewer/css/bootstrap.min.css">
9 <link rel="stylesheet" href="imgviewer/css/ekko-lightbox.min.css">
10 <style>
11 body { padding-top: 70px; }
12 .portfolio-item { margin-bottom: 25px; }
13 footer { margin: 50px 0; }
14 </style>
15 <script src="imgviewer/js/jquery.js"></script>
16 <script src="imgviewer/js/bootstrap.min.js"></script>
17 <script src="imgviewer/js/ekko-lightbox.min.js"></script>
18 <script src="https://www.jsviews.com/download/jsrender.js"></script>
19
20 </head>
21
```

# jQuery 플러그인



## ■ 이미지 뷰어 어플리케이션

jqueryplugin\_imgviewer.html

```
22 <body>
23 <!-- Navigation Bar -->
24 <nav class="navbar navbar-inverse navbar-fixed-top" role="navigation">
25 <div class="container">
26 <div class="navbar-header">
27 <button type="button" class="navbar-toggle" data-toggle="collapse"
28 data-target="#bs-example-navbar-collapse-1">
29 Toggle navigation
30
31
32
33 </button>
34 이미지 검색
35 </div>
36 <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
37 <ul class="nav navbar-nav">
38
39 DAUM 이미지 검색 페이지
40
41
42 DAUM Developers Center
43
44
45 JSONP Wiki
46
47
48 </div>
49 </div>
50 </nav>
```

# jQuery 플러그인



## ■ 이미지 뷰어 어플리케이션

```
52 <!-- Page Content -->
53 <div class="container">
54 <div class="row">
55 <!-- 검색창 -->
56 <div id="navbar" class="navbar-collapse navbar-form navbar-left">
57 <div class="form-group col-lg-12">
58 <input id="keyword" type="text" placeholder="검색어" class="form-control">
59 <button id="inquiry" class="btn btn-success form-control">검색</button>
60 </div>
61 </div>
62 </div>
63
64 <!-- 검색된 썸네일 이미지 출력 -->
65 <div id="imagelist" class="row">
66
67 </div>
68
69 <!-- 다음 가져오기 버튼 -->
70 <footer>
71 <div class="row">
72 <div class="navbar-collapse navbar-form navbar-left">
73 <div class="form-group col-lg-12">
74 <button type="submit" id="getNext" class="btn btn-primary">
75 다음 20개 가져오기</button>
76 </div>
77 </div>
78 </div>
79 </footer>
80 </div>
81 </body>
82 </html>
```

jqueryplugin\_imgviewer.html

## jQuery 플러그인



## ■ 이미지 뷰어 어플리케이션

### jqueryplugin\_imgviewer.html – 스크립트

```
20 <script type="text/javascript">
21 $(document).ready(function() {
22 /* 요청 URL 포맷
23 * https://apis.daum.net/search/image?apikey={apikey}&q=검색어&output=json
24 */
25 // JSONP 요청을 위한 문자열 지정
26 var url = "https://apis.daum.net/search/image?callback=?";
27
28 // getJSON()으로 요청을 위한 파라미터(param) 객체 생성
29 var param = {
30 pageno:1, result:20, output:"json", sort : "accu",
31 apikey : "d307e80495ec3495344edc086e532d1a"
32 };
33
34 //이미지 검색용 함수: JSONP 기법 이용
35 var searchImages= function() {
36 $.getJSON(url, param, function(data) {// getJSON()으로 이미지 검색
37 var tmpl = $.templates("#image_template");
38 console.log(data.channel.item);
39 //jsRender 템플릿에 데이터 텬더링
40 var str = tmpl.render(data.channel.item);
41 console.log(str);
42 // 검색 이미지 출력
43 $("#imagelist").append(str);
44 });
45 /*
46 $.ajax({
47 url : url,
48 dataType : "jsonp",
49 type : "GET",
50 data : param,
51 jsonp : "callback",
52 success : function(data) {
53 console.log(data);
54 var tmpl = $.templates("#image_template");
55 var str = tmpl.render(data.channel.item);
56 $("#imagelist").append(str);
57 }
58 });
59 };

```

# jQuery 플러그인



## ■ 이미지 뷰어 어플리케이션

jqueryplugin\_imgviewer.html-스크립트

```
61 //다음 20개 가져오기를 클릭했을 때의 이벤트 처리
62 $("#getNext").on("click", function() {
63 param.pageno++; //페이지 번호 증가
64 searchImages(); //이미지 검색 함수 호출
65 });
66
67 //'검색' 버튼을 클릭했을 때의 이벤트 처리
68 $("#inquery").on("click", function() {
69 $("#imagelist").empty();
70 param.pageno = 1;
71 //param 객체에 사용자가 입력한 검색 키워드를 q 속성으로 추가(q=검색어)
72 param.q = $("#keyword").val();
73 searchImages(); //이미지 검색 함수 호출
74 });
75
76 //검색어 입력을 위한 텍스트박스에서 엔터키를 눌렀을 때도 검색이 가능하도록 함.
77 $("#keyword").on("keyup", function(e) {
78 if (e.keyCode == 13) {
79 $("#inquery").trigger("click");
80 }
81 });
82
83 //이벤트 위임 처리를 통해 검색된 썸네일 이미지를 클릭했을 때 라이트박스 화면으로 보여주도록 처리
84 $("#imagelist").delegate("div.portfolio-item a", "click", function(e) {
85 e.preventDefault();
86 //썸네일 이미지를 클릭/터치했을 때 라이트박스(Modal)로 보여준다.
87 $(this).ekkoLightbox();
88 });
89 });
90
```

# jQuery 플러그인



## ■ 이미지 뷰어 어플리케이션

jqueryplugin\_imgviewer.html-스크립트

```
92 <!-- 검색된 이미지(썸네일)를 출력하기 위한 jsRender 템플릿 정의 -->
93 <script id="image_template" type="text/x-jsrender">
94 <div class="col-md-2 portfolio-item">
95 <a data-toggle="lightbox" data-gallery="multis"
96 data-title="{{:title}}" href="{{:image}}">
97
98
99 </div>
100 </script>
```