



jQuery Ajax

jQuery Ajax



■ "Asynchronous Javascript And XML"

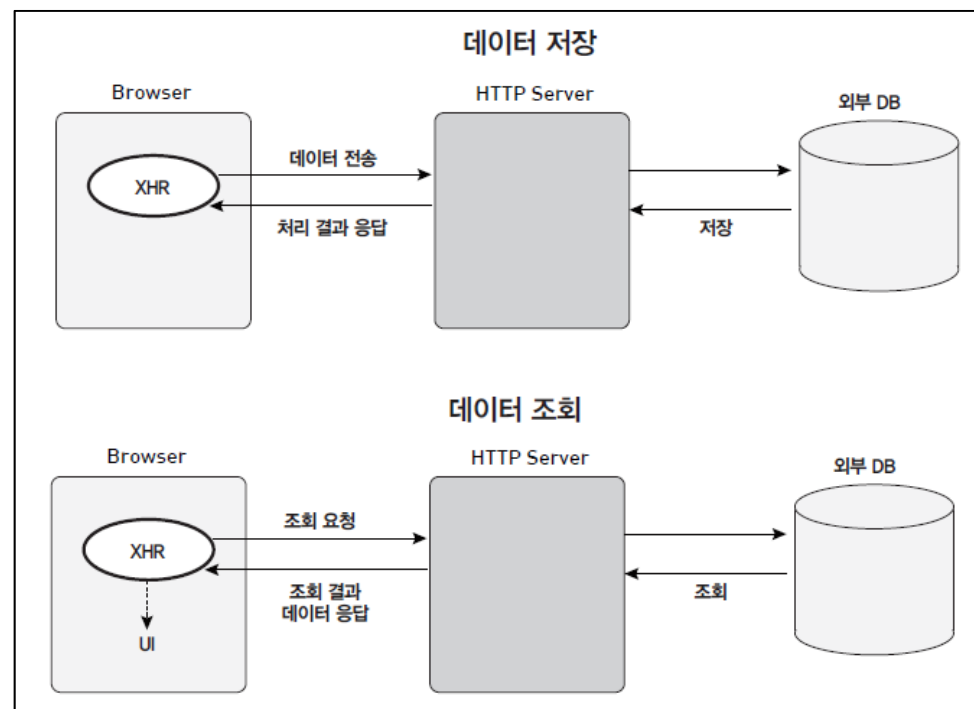
■ Javascript And XML의 의미

- 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 객체를 이용해 외부와 통신할 수 있는 기능을 제공

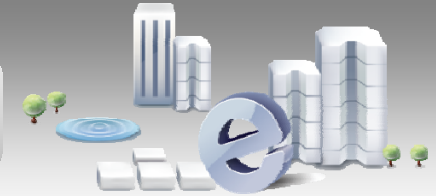
jQuery Ajax



- 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
 - 서버에 HTTP 요청을 보낸 뒤 XML, JSON, TEXT 형식 등으로 응답을 받아 페이지의 일부만을 변경



jQuery Ajax



- 웹 페이지에서 Ajax를 이용 예



[골든쿠폰]대박난박양 HOT한 트레이닝

쿠팡가 9,800원

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.

구매할 상품 옵션을 선택 후, 구매하기 버튼을 클릭해주세요.

총 상품금액 : 0원
최대 20,000원 쿠폰할인

바로구매 > 장바구니담기

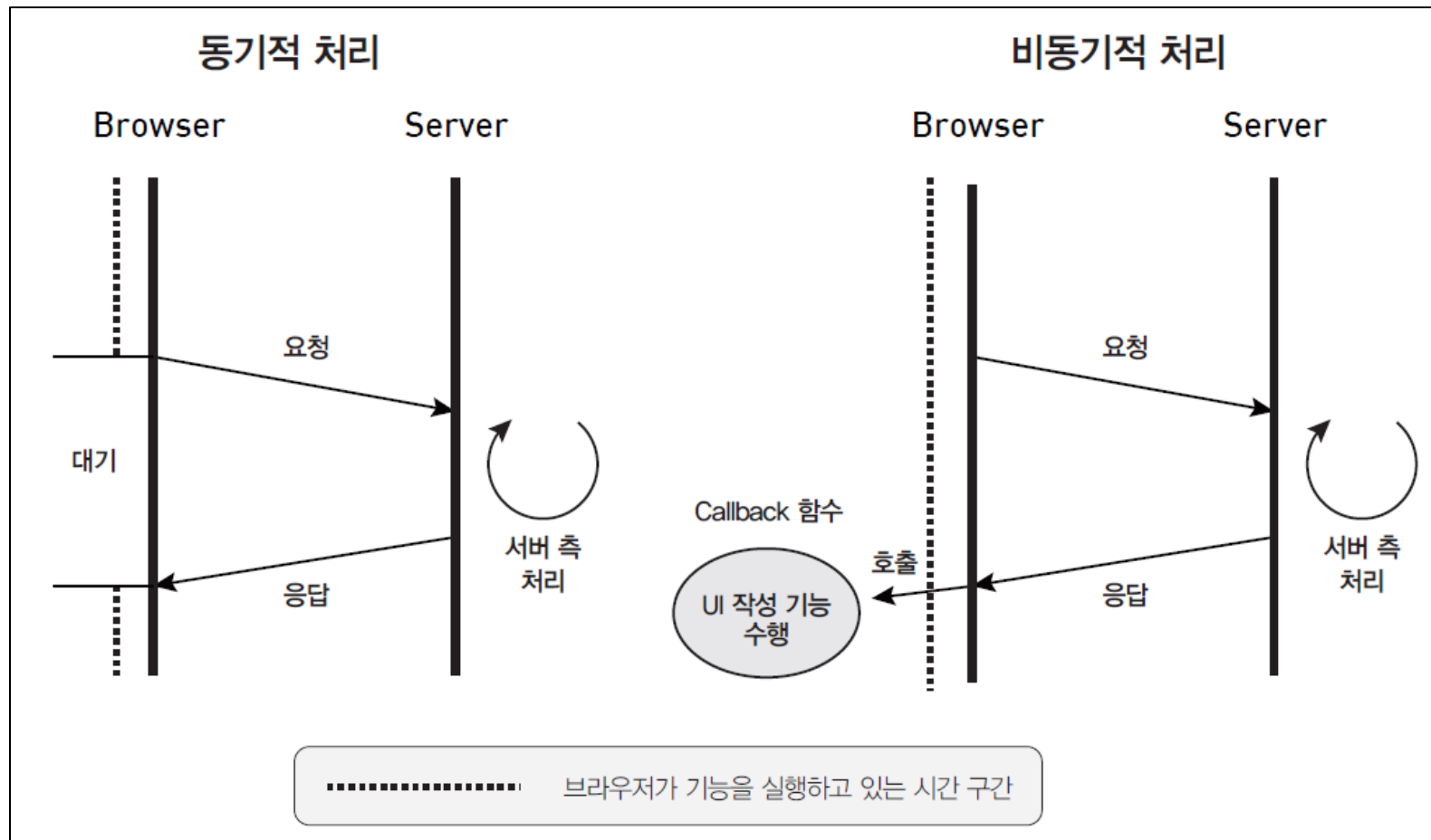
상품 설명 상품 필수 표기정보 쉬운반품 상품 문의 (26)

jQuery Ajax



■ Asynchronous의 의미

- 동기적 vs 비동기적

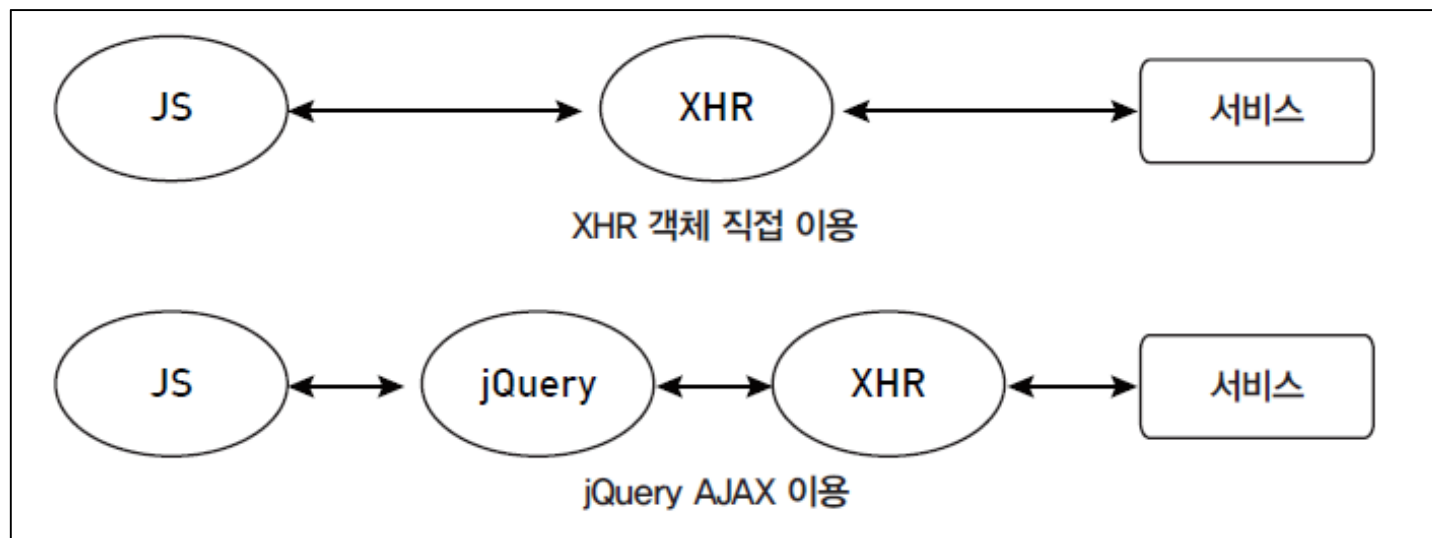


jQuery Ajax



■ XHR(XMLHttpRequest) 객체를 이용한 Ajax 통신

- Ajax는 비동기 통신을 위해 XHR 객체를 이용
- 그러나 XHR 객체를 직접 이용하여 Ajax 통신을 하는 것보다는 jQuery 라이브러리를 이용하는 것이 훨씬 간결하고 편리
- jQuery는 웹 브라우저 종류에 상관없이 같은 방식으로 Ajax 기능을 구현하도록 다양한 메소드를 제공



jQuery Ajax



jQuery Ajax 주요 메소드

Ajax 메소드	기능/예
\$.ajax()	<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 - \$.get(), \$.post(), \$.getJSON() 기능을 결합한 메소드
\$.get()	<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
\$.post()	<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
\$.getJSON()	<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
load()	<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
\$.getScript()	<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
\$.ajaxSetup()	<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:      "데이터를 전송할 url"(액션 페이지),  
    type :    "서버로 전송하는 전송방식"(get, post 방식),  
    data:      "서버에 전송할 데이터",  
    timeout:   "응답 제한시간",  
    datatype: "서버가 리턴하는 데이터타입"  
              (HTML, XML, JSON, TEXT 등 리턴타입),  
    async:     "비동기여부"(default: true),  
    success:   ajax 통신 성공시 호출할 이벤트 핸들러  
              function(data)(성공콜백함수(data:서버의 반환값) {  },  
    error:     function()(실패 콜백함수) {  }  
    complete: function()(요청이 완료되었을 때 실행 콜백함수) {  }  
});
```

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="https://code.jquery.com/jquery-3.4.1.min.js"></script>
8
9         <script>
10             $(document).ready(function() {
11                 //코드 추가
12             });
13         </script>
14
15 </head>
16
17 <body>
18     <div>
19         <div>
20             <h1>Ajax 활용</h1>
21         </div>
22         <div>
23             <button id="btn1">XML</button>
24             <button id="btn2">JSON</button>
25             <button id="btn3">HTML</button>
26         </div>
27         <hr>
28         <div>
29             <ul id="listArea">
30                 <li id="item">데이터 로드 하기 전 . . .</li>
31             </ul>
32         </div>
33     </div>
34 </body>
35 </html>
```

jquery_ajax01.html

jQuery Ajax



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

```
12      // XML 파일 처리
13      $('#btn1').click(function() {
14          $.ajax({
15              url: 'xml-data.xml', //요청하고자하는 서비스 url
16              type: 'get', //전송방식
17              dataType: 'xml', //서버로부터 반환 받아올 데이터 형식
18              timeout: 10000, //응답 제한시간
19              success: function(xmlData) { // 성공시 실행할 콜백함수
20                  console.log(xmlData);
21                  var tagList = ""; // <li> 항목 저장 변수 선언
22                  $(xmlData).find('player').each(function() { //반환데이터에서 <li> 항목 생성
23                      tagList += "<li>" + $(this).find('name').text() + "</li>";
24                  });
25                  $('#listArea').empty();
26                  $('#listArea').append(tagList);
27                  $('#listArea').show();
28              },
29              error: function() { //전송 및 요청 실패 시 리스트뷰에 error 표시
30                  $("#listArea").html("<p>Error!!</p>");
31              }
32          });
33      });
```

jquery_ajax01.html - 스크립트

xml-data.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <soccerplayer>
3     <player><name>손흥민</name></player>
4     <player><name>이강인</name></player>
5     <player><name>이승우</name></player>
6     <player><name>황의조</name></player>
7 </soccerplayer>
```

jQuery Ajax



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

```
35 // JSON 파일 처리
36 $('#btn2').click(function() {
37     $.getJSON('json-data.json', function(jsonData) {
38         console.log(jsonData);
39         var tagList = "";
40         $.each(jsonData.stuinfo, function() {
41             tagList += "<li>" + this.name + "</li>";
42         });
43         $('#listArea').html(tagList);
44     });
45 });
46
47 // HTML 파일 처리
48 $('#btn3').click(function() {
49     $('#listArea').empty();
50     //데이터를 load하여 listArea에 추가
51     $('#listArea').load('html-data.html li', function(htmlData) {
52         console.log(htmlData);
53         $('#listArea').show();
54     });
55 });
```

jquery_ajax01.html - 스크립트

json-data.json

```
1 {
2   "stuinfo" : [
3     {"name": "폴킴"},
4     {"name": "장범준"},
5     {"name": "거미"},
6     {"name": "태연"},
7     {"name": "아이유"},
8     {"name": "박정현"},
9     {"name": "윤립"}
10  ]
11 }
```

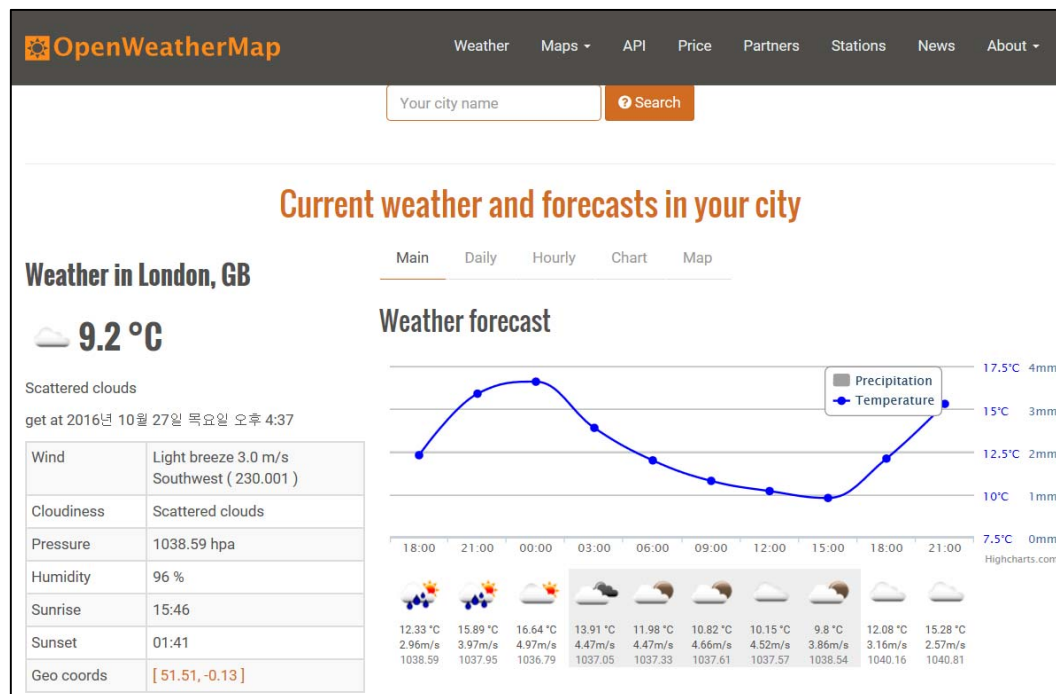
html-data.html

```
3 <body>
4   <ul>
5     <li>Javascript</li>
6     <li>jQuery</li>
7     <li>Ajax</li>
8     <li>Node.js</li>
9   </ul>
10 </body>
```

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

Weather in your city Sign In **Sign Up**

OpenWeatherMap

Weather Maps API Price Partners Stations News About

회원가입

Create New Account

Username

Enter email

Password Repeat Password

☐ I agree to the [Terms of Service](#) and [Privacy Policy](#)

Create Account

Weather in your city **Sign In** Sign Up

OpenWeatherMap

Weather Maps API Price Partners Stations News About

로그인

Sign In To Your Account

ugkang@gachon.ac.kr

.....

☐ Remember me

Submit

Not registered? [Create an Account](#).

Lost your password?? [Click here to recover](#).

jQuery Ajax 실습



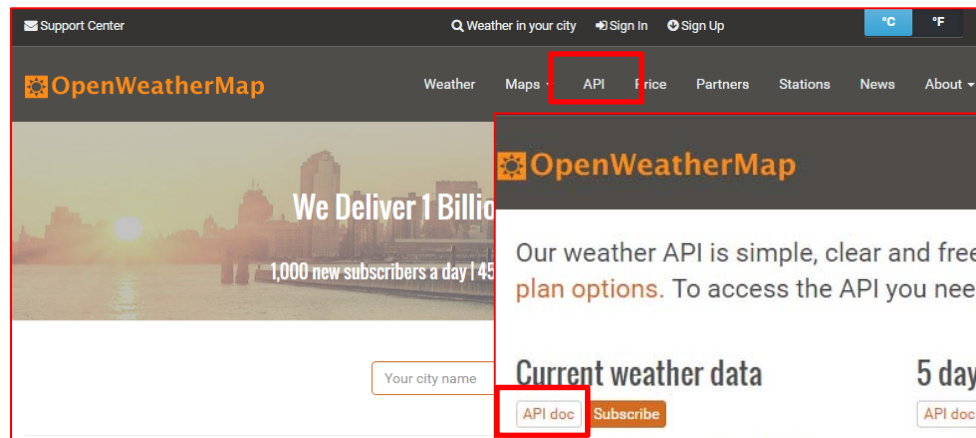
- Openweathermap API를 활용한 원하는 도시의 날씨 정보 가져오기
 - 회원 가입 후 로그인하여 API 키 발급

The screenshot displays the OpenWeatherMap website interface. On the left, the '로그인 화면' (Login screen) shows a 'Signed in successfully' notice and the 'API keys' tab selected in the navigation menu, marked with a circled '1'. The user profile information is visible: Username: ugkang, Email: ugkang@gachon.ac.kr, Full name: un-gu, kang. On the right, the 'API keys' page shows a 'Notice' that 'API key was created successfully'. Below this, a table lists the generated API key: 'c2bb263a34...179af1878' with the name 'ugkang', marked with a circled '2'. A blue box labeled 'API Key 생성됨' (API Key generated) points to the key. The 'Create key' section shows the 'Name' input field and the 'Generate' button, both marked with a circled '3'.

jQuery Ajax 실습



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



Current weather data

Access current weather data for any location on Earth including is frequently updated based on global models and data from mo Data is available in JSON, XML, or HTML format.

Call current weather data for one location

By city name

Description:
You can call by city name or city name and country code. API responds with a list of results that match a searching word.

API call:
`api.openweathermap.org/data/2.5/weather?q={city name}`

Parameters:
city name and country code divided by comma, use ISO 3166 country codes

복사

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

1

도시명

2

전송모드

3

표시타입(섭씨)

4

API Key

jQuery Ajax 실습



- Openweathermap API를 활용한 원하는 도시의 날씨 정보 가져오기
 - API response(json)

```
{
  "coord": {
    "lon": -122.08,
    "lat": 37.39
  },
  "weather": [
    {
      "id": 800,
      "main": "Clear",
      "description": "clear sky",
      "icon": "01d"
    }
  ],
  "base": "stations",
  "main": {
    "temp": 296.71,
    "pressure": 1013,
    "humidity": 53,
    "temp_min": 294.82,
    "temp_max": 298.71
  },
  "visibility": 16093,
  "wind": {
    "speed": 1.5,
    "deg": 350
  },
  "clouds": {
    "all": 1
  },
  "dt": 1560350645,
  "sys": {
    "type": 1,
    "id": 5122,
    "message": 0.0139,
    "country": "US",
    "sunrise": 1560343627,
    "sunset": 1560396563
  },
  "timezone": -25200,
  "id": 420006353,
  "name": "Mountain View",
  "cod": 200
}
```

jQuery Ajax 실습



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

Icon url : "http://openweathermap.org/img/wn/" + icon

How to get icon URL
For code 501 - moderate rain icon = "10d"
URL is
<http://openweathermap.org/img/wn/10d@2x.png>

Icon list

Day icon	Night icon	Description
<u>01d.png</u> 	<u>01n.png</u> 	clear sky
02d.png 	02n.png 	few clouds
03d.png 	03n.png 	scattered clouds
04d.png 	04n.png 	broken clouds

```
▼ weather: Array(1)  
  ▼ 0:  
    description: "haze"  
    icon: "50d"  
    id: 721  
    main: "Haze"
```


jQuery Ajax 실습



- Openweathermap API를 활용한 원하는 도시의 날씨 정보 가져오기
 - 실행화면

날씨검색 웹


날씨 확인



도시를 입력해주세요.

날씨검색 웹

날씨 확인



도시를 입력해주세요.

Country: Seoul, KR
Current Temperature: 26.01 °C
Current Humidity: 78 %
Current Wind Speed: 1 m/s
Weather Conditions: haze
Clouds: 20%
Sunrise: Fri Oct 04 2019 06:29:22 GMT+0900 (한국 표준시)
Sunset: Fri Oct 04 2019 18:12:29 GMT+0900 (한국 표준시)

jQuery Ajax 실습



- Openweathermap API를 활용한 원하는 도시의 날씨 정보 가져오기

```
<!DOCTYPE html>
<html lang="ko">
  <head>
    <meta charset="utf-8">
    <title>jQuery Ajax</title>
    <link type="text/css" rel="stylesheet" href="./css/custom.css">
    <script src="https://code.jquery.com/jquery-3.4.1.min.js"></script>

  </head>
  <body>
    <section class="content-wrapper">
      <h2>날씨검색 웹</h2>
      <div class="input-wrapper">
        <input type="text"><!-- 검색할 도시 입력창 -->
        <button class="confirm">날씨 확인</button>
      </div>

      <div class="result-wrapper">
        
      </div>

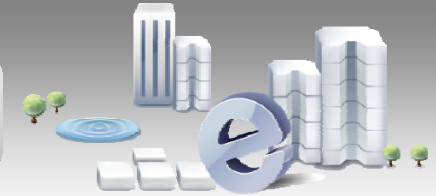
      <h2 id="result-text">도시를 입력해주세요.</h2>

      <div id="weather-info">
        <!-- 날씨 정보 표시 -->
      </div>
    </section>

    <!-- Script -->
    <script type="text/javascript" src="./js/ajax_api.js"></script>
  </body>
</html>
```

weather-info.html

jQuery Ajax 실습



- Openweathermap API를 활용한 원하는 도시의 날씨 정보 가져오기
- jQuery Ajax 비동기 통신(1) ajax.js

```
$(document).ready(function () {
    $(".confirm").click(function () {
        var city = "검색 도시 저장 변수";
        var appid = " API Key ";
        var url = "http:// " + city + " " + appid;

        $.ajax({
            url: url,
            dataType: "jsonp",
            type: "GET",
            success: function (data) {
                console.log(data);
            },
            error: function (error) {
                alert("Error!");
            },
            complete: function () {
                alert("Complete!");
            }
        });

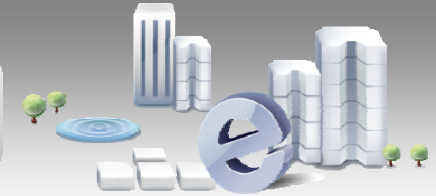
        //날씨 정보 표시 함수
        function viewMapping(data) {

        } //end of viewMapping
    });
});
```

API response

```
▼ Object 1
  base: "stations"
  ► clouds: {all: 20}
  cod: 200
  ► coord: {lon: 126.98, lat: 37.57}
  dt: 1570174914
  id: 1835848
  ▼ main:
    humidity: 78
    pressure: 1011
    temp: 26.01
    temp_max: 27
    temp_min: 25
    ► __proto__: Object
    name: "Seoul"
  ▼ sys:
    country: "KR"
    id: 5509
    message: 0.0065
    sunrise: 1570138162
    sunset: 1570180349
    type: 1
    ► __proto__: Object
    timezone: 32400
    visibility: 10000
  ▼ weather: Array(1)
    ▼ 0:
      description: "haze"
      icon: "50d"
      id: 721
      main: "Haze"
      ► __proto__: Object
      length: 1
      ► __proto__: Array(0)
  ▼ wind:
    deg: 210
    speed: 1
```

jQuery Ajax 실습



- Openweathermap API를 활용한 원하는 도시의 날씨 정보 가져오기
- jQuery Ajax 비동기 통신(2)

fetch.js

```
$(document).ready(function () {
    $(".confirm").click(function () {
        var city = "검색 도시 저장 변수";
        var appid = " API Key ";
        var url = "http:// " + city + " " + appid;

        //fetch API를 이용한 비동기 통신
        fetch(url).then(function (response) {
            response.json().then((data) => {
                console.log(data);
            });
        });
    });

    //날씨 정보 표시 함수
    function viewMapping(data) {

    } //end of viewMapping
});
```

jQuery Ajax 실습

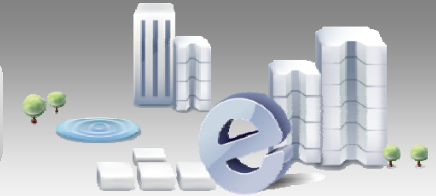


- Openweathermap API를 활용한 원하는 도시의 날씨 정보 가져오기
- jQuery Ajax 비동기 통신(3)

promise.js

```
$(document).ready(function () {  
    // async-await을 이용한 비동기 통신  
    $(".confirm").click(async function () {  
        var city = "검색 도시 저장 변수";  
        var appid = " API Key ";  
        var url = "http:// " + city + " " + appid;  
  
        var response = await fetch(url);  
        var data = await response.json();  
        console.log(data)  
  
    });  
  
    //날씨 정보 표시 함수  
    function viewMapping(data) {  
  
    } //end of viewMapping  
});
```

jQuery Ajax 실습



- Openweathermap API를 활용한 원하는 도시의 날씨 정보 가져오기
- jQuery Ajax 비동기 통신(4)

pure.js

```
$(document).ready(function () {
    $(".confirm").click(async function () {
        var city = "검색 도시 저장 변수";
        var appid = " API Key ";
        var url = "http:// " + city + " " + appid;

        //XMLHttpRequest를 이용한 비동기 통신
        var xhr = new XMLHttpRequest();
        xhr.onreadystatechange = function () { // 요청에 대한 콜백
            if (xhr.readyState === xhr.DONE) { // 요청이 완료되면
                if (xhr.status === 200 || xhr.status === 201) {
                    data = JSON.parse(xhr.responseText)
                    console.log(data);

                } else {
                    console.error(xhr.responseText)
                }
            }
        }
        xhr.open('GET', url) // 메소드와 주소 설정
        xhr.send() // 요청 전송

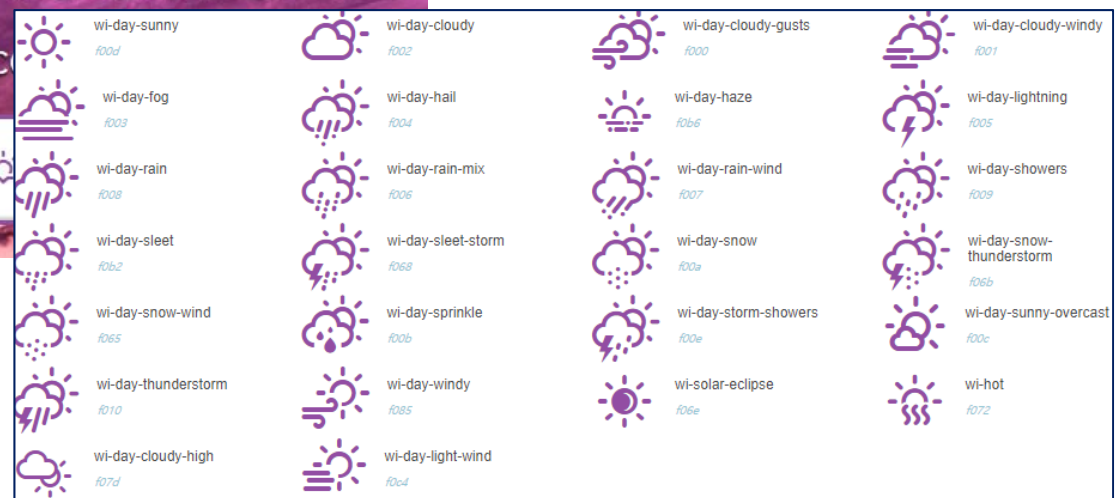
    });

    //날씨 정보 표시 함수
    function viewMapping(data) {

    } //end of viewMapping
});
```


jQuery Ajax 실습



- Weather Icons을 사용하여 날씨 아이콘 표시
- url : <https://erikflowers.github.io/weather-icons/>



jQuery Ajax 실습



- Weather Icons을 사용하여 날씨 아이콘 표시
- Icon list
- url: <https://erikflowers.github.io/weather-icons/api-list.html>
 - icon은 “wi-owm-상태 코드” 형식의 클래스를 선언함으로써, font awesome과 동일한 형식으로 사용 가능

  https://erikflowers.github.io/weather-icons/api-list.html	
Open Weather Map	World Meteorological Organization
wi-owm-200: thunderstorm	wi-wmo4680-0, wi-wmo4680-00: thermometer
wi-owm-201: thunderstorm	wi-wmo4680-1, wi-wmo4680-01: cloudy
wi-owm-202: thunderstorm	wi-wmo4680-2, wi-wmo4680-02: thermometer
wi-owm-210: lightning	wi-wmo4680-3, wi-wmo4680-03: cloudy
wi-owm-211: lightning	wi-wmo4680-4, wi-wmo4680-04: fog
wi-owm-212: lightning	wi-wmo4680-5, wi-wmo4680-05: fog
wi-owm-221: lightning	wi-wmo4680-10: fog
wi-owm-230: thunderstorm	wi-wmo4680-11: fog
wi-owm-231: thunderstorm	wi-wmo4680-12: lightning
wi-owm-232: thunderstorm	wi-wmo4680-18: strong-wind
wi-owm-300: sprinkle	wi-wmo4680-20: fog
wi-owm-301: sprinkle	wi-wmo4680-21: rain-mix
wi-owm-302: rain	wi-wmo4680-22: rain-mix
wi-owm-310: rain-mix	wi-wmo4680-23: rain
wi-owm-311: rain	wi-wmo4680-24: snow
wi-owm-312: rain	wi-wmo4680-25: hail
wi-owm-313: showers	wi-wmo4680-26: thunderstorm

jQuery Ajax 실습



- Weather Icons을 사용하여 날씨 아이콘 표시
- 사용 방법

1. Weather icon CSS CDN 추가

```
<link rel="stylesheet" type="text/css"
      href="https://cdnjs.cloudflare.com/ajax/libs/weather-icons/2.0.9/css/weather-
      icons.min.css">
```

2. Icon 스타일 설정(필요시)

```
/* Weather Icons */
.result-wrapper i {
    font-size: 70px;
    color: white;
}
```

3. Icon 선언 및 DOM 객체에 icon 추가

```
var icon = `<i class="wi wi-owm-${data.weather[0].id}"></i>`;
$(".result-wrapper").html(icon);
```

` : Grave Accent - html 코드를 위치시키거나 특정 변수를 삽입할 때 사용