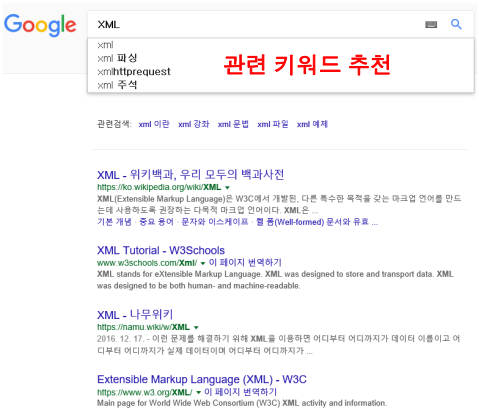
9. Ajax

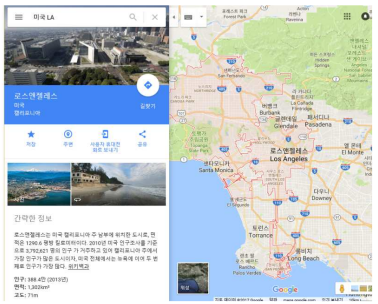
Ajax 개요

- Ajax(Asynchronous JavaScript And XML)
  - 비동기적인 웹 애플리케이션을 생성하기 위한 클라이언트 측의 웹 기술 집합
    - HTML and CSS for presentation
    - **DOM** for dynamic display of and interaction with data
    - **XML** or **JSON** for the interchange of data
    - **XMLHttpRequest** object for asynchronous communication
    - **JavaScript** to bring these technologies together
  - Ajax는 웹 서버와 비동기적으로 (백그라운드에서) 데이터를 교환함으로써 웹 페이지가 비동기적으로 갱신되도록 함
    - 전체 페이지를 다시 reload하지 않고 일부 내용을 동적으로 갱신 가능

Ajax 개요

- 사용 예
  - Google 키워드 추천, Google Maps, Yahoo! Stock

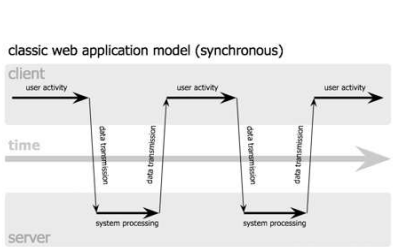


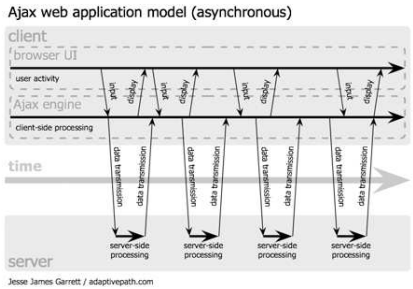


- 지도 상의 위치를 클릭하면 관련 정보를 출력
- 지도를 이동하거나 확대/축소 가능

Ajax 개요

- Synchronous Vs. Asynchronous



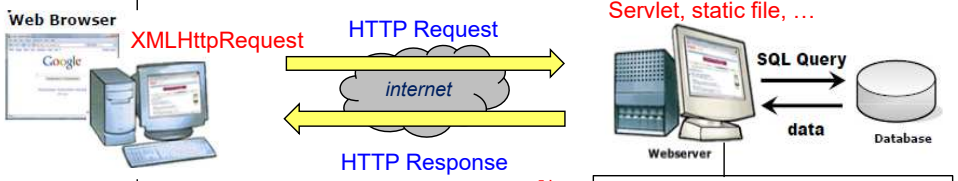


- 동기적 요청은 서버에서 처리가 완료되고 응답이 올 때까지 클라이언트(browser)의 실행이 중단됨
- 비동기적 요청은 클라이언트가 서버의 응답을 기다리지 않고 계속 진행 가능
  - ✓ 다른 JavaScript 코드 실행
  - ✓ Form에 data 입력
  - ✓ 서버로부터 응답이 오면 미리 등록된 callback 함수를 실행하여 처리

Ajax 개요

- 동작 과정

1. 웹 페이지에서 이벤트 발생 (예: 페이지 로딩, 버튼 클릭)
2. JavaScript 프로그램에서 XMLHttpRequest 객체 생성
3. XMLHttpRequest 객체가 웹 서버로 요청(request) 전송



4. 웹 애플리케이션 서버가 요청을 처리
5. 서버가 웹 브라우저로 응답 (response) 전송

6. JavaScript 코드가 응답에 포함된 데이터를 읽어서 처리
7. JavaScript 코드가 적절한 동작(예: 페이지 내용 갱신)을 수행

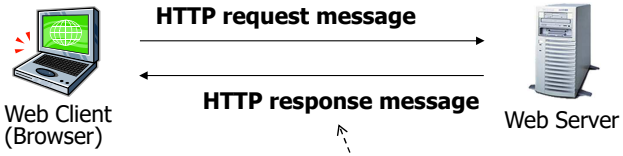
Ajax 개요

- 동작 방식
  - 웹 브라우저에서 실행되는 JavaScript 코드가 XMLHttpRequest 객체를 사용하여 비동기적으로 웹 서버 프로그램 호출 및 데이터 교환
  - 대부분의 브라우저들이 내부적으로 XMLHttpRequest 객체를 지원 (ex. Edge, Chrome, Firefox, Safari, Opera 등)
  - 브라우저가 비동기 요청에 대한 응답(response)을 받으면 미리 정의된 JavaScript 함수(callback function)를 호출하여 응답 데이터를 이용 (ex. 요청 처리 결과를 화면 출력)

HTTP Messages

```
METHOD /path-to-resource HTTP/version-number <CR><LF>
Header-Name-1: value <CR><LF>
Header-Name-2: value <CR><LF>
…
<CR><LF>          (empty line)
[ message body ]  (optional)
```

request line

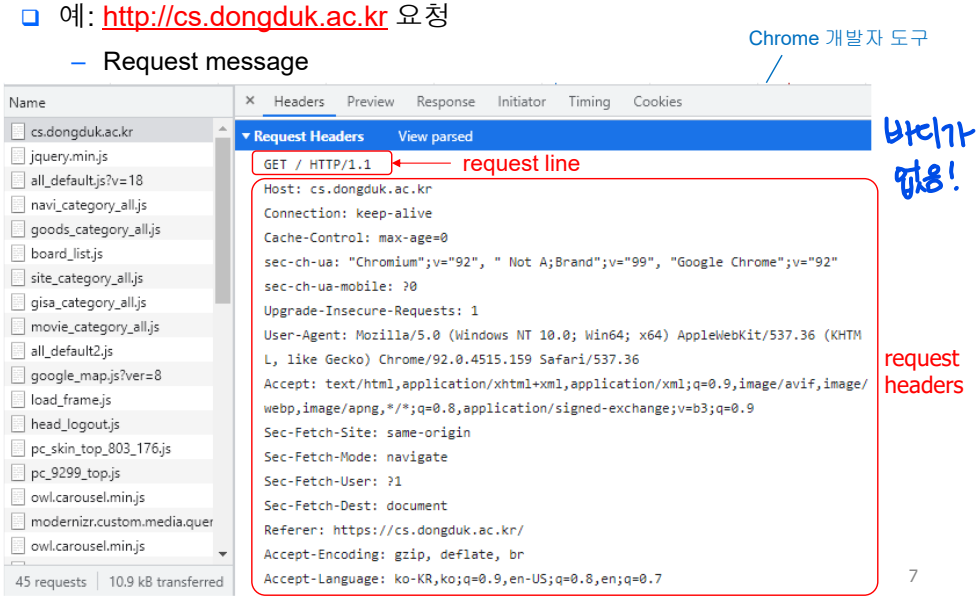



```
HTTP/version-number status-code message <CR><LF>
Header-Name-1: value <CR><LF>
Header-Name-2: value <CR><LF>
…
<CR><LF>          (empty line)
[ message body ]  (optional)
```

status line

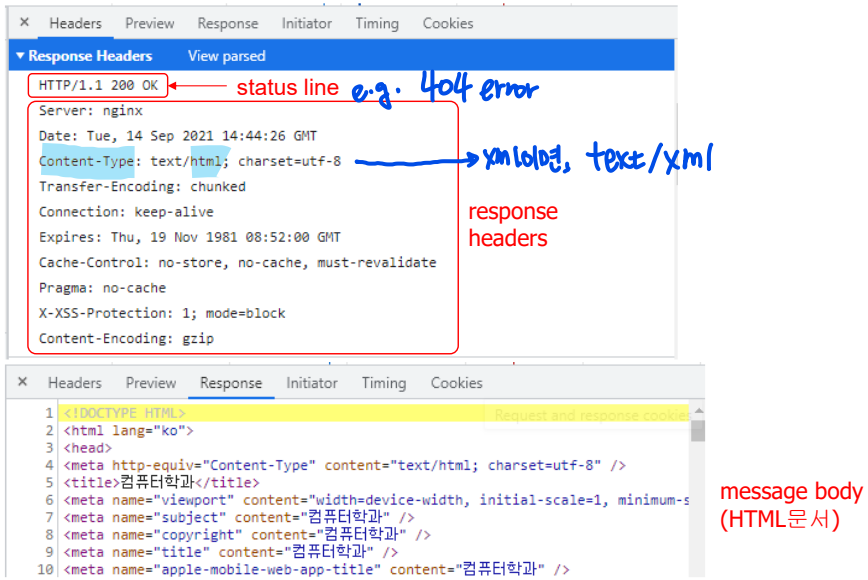
HTTP Messages

- 예: http://cs.dongduk.ac.kr 요청
  - Request message



HTTP Messages

– Response message



XMLHttpRequest 객체

– XMLHttpRequest의 함수

Function	Description
new XMLHttpRequest()	XMLHttpRequest 객체 생성
open(*method,url,async,user,psw*)	요청(request) 정의 (필요한 정보 설정) - *method*: request type (GET / POST) - *url*: file (program) location - *async*: true (asynchronous) / false (synchronous) - *user*: optional user name - *psw*: optional password
setRequestHeader(*header, value*)	요청 헤더를 설정 (헤더 이름-값 쌍을 지정)
send()	요청을 서버에 전송
send(*string*)	데이터를 포함한 요청을 서버에 전송 (used for POST requests)
abort()	현재의 요청을 취소
getAllResponseHeaders()	모든 응답 헤더 정보를 반환
getResponseHeader(*string*)	특정 응답 헤더 정보를 반환 (헤더 이름 지정)

XMLHttpRequest 객체

– XMLHttpRequest의 속성

Property	Description
readyState	XMLHttpRequest 객체의 상태 정보 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
onreadystatechange	readyState 속성 값이 변경될 때 호출될 함수를 정의
responseText	응답 데이터를 문자열 형태로 반환
responseXML	응답 데이터를 XML 형식으로 반환
status	요청 처리에 대한 status-code (예: 200: "OK", 403: "Forbidden", 404: "Not Found", 500: "Internal Server Error" …)
statusText	요청 처리에 대한 status-message (예: "OK", "Not Found")

Ajax Request

❑ 요청 정의 및 전송

```
var xhttp = new XMLHttpRequest();          // XMLHttpRequest 객체 생성
xhttp.open("GET", "ajax-info.txt", true);  // 요청 정의 (method, url, async)
xhttp.send();                              // 서버로 요청 전송
```

– method

- GET
  - ✓ 일반적인 resource 요청
  - ✓ 데이터 전송 시 URL에 query string(name-value 쌍들) 포함 가능
    - 길이의 제한이 있으며 URL 상에 노출됨
- POST
  - ✓ 데이터 전송, 서버 측의 파일이나 데이터베이스 갱신 요청
  - ✓ 데이터를 request message의 body에 포함시켜 전달
    - 길이의 제한이 없으며 URL에 노출되지 않음
    - 대량의 데이터(ex. 파일)나 사용자 정보 전송에 적합

Ajax Request

- GET 요청 예
  - xhttp.open("GET", "demo-get.jsp", true); xhttp.send();
    - ✓ 주의: 재호출 시 브라우저에 cache된 결과가 반환될 수 있음
  - xhttp.open("GET", "demo-get.jsp?t=" + Math.random(), true);
    xhttp.open("GET", "demo-get.jsp?t=" + new Date().getTime(), true);
    - ✓ query string이 다르므로 브라우저에 cache되지 않은 새로운 처리 결과를 프로그램에 요청
  - xhttp.open("GET", "demo-get.php?fname=Steve&lname=Jobs", true);
    - ✓ URL을 통해 데이터 전송
- POST 요청 예

```
xhttp.open("POST", "demo-post.php", true);
xhttp.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
xhttp.send("fname=Steve&lname=Jobs");
```

  - Message body를 통해 데이터 전송

form을 통해 입력받은 값들을 이름 = 값 형태로 보냄

Ajax Request

- url parameter
  - 서버 내 프로그램/파일 주소
    - ✓ 예: Script(JSP, PHP, ASP 등), Servlet, XML, JSON, plain text 등
- async parameter
  - *true* : 요청과 응답이 비동기적으로 전송 및 처리됨
    - ✓ 서버로부터 응답 메시지가 도착 및 사용가능 할 때 실행될 callback 함수를 *onreadystatechange* 속성의 값으로 미리 지정해야 함
  - *false* : 요청과 응답이 동기적으로 전송 및 처리됨
    - ✓ callback 함수를 지정할 필요 없음

```
var xhttp = new XMLHttpRequest();                    // XMLHttpRequest 객체 생성
xhttp.open("GET", "ajax-info.txt", true);            // 비동기적인 GET 요청 정의
xhttp.onreadystatechange = processResponse;          // callback function 지정
xhttp.send();                                        // request 전송

function processResponse() {                         // callback function
  if (xhttp.readyState == 4 && xhttp.status == 200) {
    …                                                // 응답으로 반환된 요청 처리 결과를 이용
  }
};
```

함수 (callback)

서버에서 성공적으로 처리

4 여야지 제대로 도착

Ajax Response 처리

- XMLHttpRequest 객체의 속성
  - *readyState*: XMLHttpRequest 객체의 상태 정보 (p.10의 표 참조)
  - *onreadystatechange*: *readyState* 속성 값이 변경될 때 호출될 callback function을 지정
    - callback function은 *readyState* 값이 변경될 때마다 호출됨
  - *status*: 응답 메시지의 상태 코드

→ *readyState* == 4 이고 *status* == 200일 때 응답 데이터 사용 가능

```
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {          // anonymous function 정의
    if (this.readyState == 4 && this.status == 200) {      // 응답 사용 가능
        // 응답 메시지의 내용을 <demo> 엘리먼트의 content로 사용
        document.getElementById("demo").innerHTML = this.responseText;
    }
};
xhttp.open("GET", "ajax-info.txt", true);
xhttp.send();
// 주의: 위 함수에서 this는 함수의 소유자인 XMLHttpRequest 객체를 가리킴
```

xhttp

같음!! id="demo"인 값!

해당 응답받은 내용을 text값 읽어온다.

Ajax Response 처리

- *responseText* 속성
  - 서버의 응답 데이터를 문자열로 반환
- *responseXML* 속성
  - 응답 데이터가 XML 데이터일 경우 DOM tree의 root node (Document 객체)를 반환
    - ✓ XMLHttpRequest는 DOM parser를 이용하여 XML 데이터를 parsing함

```
xhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
    txt = "";   xmlDoc = this.responseXML          // DOM tree의 document 객체
    x = xmlDoc.getElementsByTagName("ARTIST");     // DOM API 사용
    for (i = 0; i < x.length; i++) {
      txt += x[i].firstChild.nodeValue + "<br>";   // 첫 번째 자식 노드의 값
    }
    document.getElementById("demo").innerHTML = txt;   // <demo>의 content 변경
  }
};
xhttp.open("GET", "cd_catalog.xml", true);  xhttp.send();          // XML 파일 요청
```

XHR

Ajax Response 처리

- getAllResponseHeaders(), getResponseHeader(*string*)
  - 응답 메시지에 포함된 모든 header 정보들 / 특정 header 값을 반환

```
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
   if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo1").innerHTML = this.getAllResponseHeaders();
                                                    // 모든 header 정보 출력

      document.getElementById("demo2").innerHTML =
         = this.getResponseHeader("Last-Modified");     // Last-Modified header 값 출력
   }
};
```


Example 1

- XML 파일 검색 및 출력

```
<html><body><h1>The XMLHttpRequest Object</h1>
<form><input type="button" onclick="loadDoc()" value="Get my CD collection"/></form><br>
<table id="demo"></table>                                   // 검색 결과를 출력할 테이블
<script>
function loadDoc() {                                        // onclick 이벤트 핸들러
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) { myFunction(this); }
  };
  xhttp.open("GET", "cd_catalog.xml", true);  xhttp.send();      // XML file 요청
}
function myFunction(xhttp) {                                // callback function
  var xmlDoc = xhttp.responseXML;                  // 전송된 XML file에 대한 Document 객체
  var table = "<tr><th>Artist</th><th>Title</th></tr>";
  var cds = xmlDoc.getElementsByTagName("CD");
  for (var i = 0; i < cds.length; i++)                      // XML data 검색
    table += `<tr><td>${cds[i].getElementsByTagName("ARTIST")[0].firstChild.nodeValue}
              </td><td>${cds[i].getElementsByTagName("TITLE")[0].textContent}</td></tr>`;
    document.getElementById("demo").innerHTML = table;    // demo 테이블의 내용으로 출력
}
</script></body></html>
```

Example 1

- cd_catalog.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<CATALOG>
  <CD>
    <TITLE>Empire Burlesque</TITLE>
    <ARTIST>Bob Dylan</ARTIST>
    <COUNTRY>USA</COUNTRY>
    <COMPANY>Columbia</COMPANY>
    <PRICE>10.90</PRICE>
    <YEAR>1985</YEAR>
</CD>
<CD>
    <TITLE>Hide your heart</TITLE>
    <ARTIST>Bonnie Tyler</ARTIST>
    <COUNTRY>UK</COUNTRY>
    <COMPANY>CBS Records</COMPANY>
    <PRICE>9.90</PRICE>
    <YEAR>1988</YEAR>
</CD>
…
</CATALOG>
```

실행 결과:

The XMLHttpRequest Object

Get my CD collection click! → Ajax 호출

⇩

The XMLHttpRequest Object

Get my CD collection

Artist	Title
Bob Dylan	Empire Burlesque
Bonnie Tyler	Hide your heart
Dolly Parton	Greatest Hits
Gary Moore	Still got the blues
Eros Ramazzotti	Eros

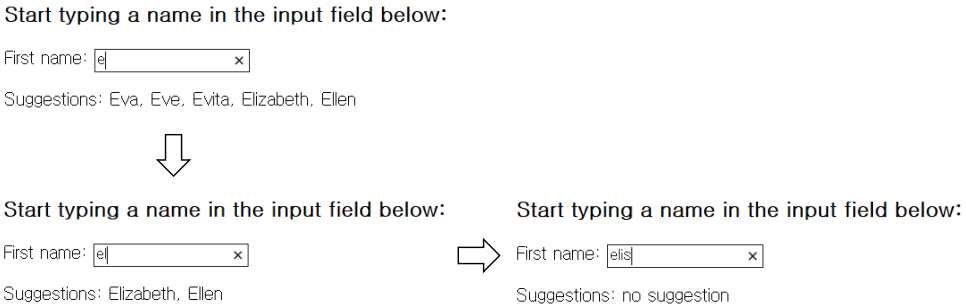
Example 1

- jQuery.ajax() 를 이용한 Ajax 구현

```
<form><input type="button" value="Get my CD collection"/></form><br>
<table id="demo"></table>                                   // 검색 결과를 출력할 테이블
<script src="scripts/jquery-1.12.4.min.js"></script>
<script>
$(document).ready(function() {
  $(':button').click(loadDoc);                              // 버튼 클릭 시 loadDoc() 호출 설정
});
function loadDoc() {
  $.ajax({ type: "GET",                                     // Ajax 실행 함수
       url: "cd_catalog.xml",
       datatype: "xml",
       success:  function(xmlDoc) {                         // callback function 설정
          var table = "<tr><th>Artist</th><th>Title</th></tr>";
          $(xmlDoc).find("CD").each( function() {
             table += "<tr><td>" + $(this).find("ARTIST").first().text()
                   + "</td><td>" + $(this).find("TITLE").first().text() + "</td></tr>";
          } );
          $('#demo').empty().append(table);
       } });
} </script></body></html>
```

Example 2

- 추천 키워드
  - 입력 창에 글자를 입력할 때마다 서버에 추천 키워드(hint)를 요청
  - 서버 프로그램은 입력 값에 대한 hint를 검색하여 반환



Example 2

- Web page (suggest.html) : HTML form & JavaScript code

```
<html><head><script>
function showHint(str) {                                    // str: 입력 필드의 값
 if (str.length == 0) {                                     // 입력 값이 없으면
    document.getElementById("txtHint").innerHTML = "";      // txtHint의 내용을 clear
    return;
 } else {
    $.ajax({ type: "GET",
              url: "gethint.jsp?q=" + str,                  // 요청 URL
              dataType: "text",                             // 응답 데이터의 형식: plain text
              success: function(response) {                 // callback function
                  document.getElementById("txtHint").innerHTML = response;
              },
              error: function(jqXHR) { alert("ERROR: " + JSON.stringify(jqXHR)); }
    });
}
</script></head>
<body>
<p><b>Start typing a name in the input field below:</b></p>
<form> First name: <input type="text" onkeyup="showHint(this.value)" /></form>
<p>Suggestions: <span id="txtHint"></span></p> …     // 키워드 hint 출력 장소
```

이벤트 핸들러

키보드로 한 문자 입력 시 showHint() 함수를 호출하고 입력된 문자열을 전송

Example 2

- JSP(Java)로 구현한 서버 프로그램 (gethint.jsp)

```
<%
   // 이름들의 배열 정의 (→ 또는 데이터베이스 검색)
   String[] names = { "Anna", "Brittany", "Cinderella", "Diana", "Eva", "Fiona", "Gunda", … };
   String q = request.getParameter("q");          // URL에서부터 q 파라미터 값 (입력 값) 추출
   String hint = null;

   // q가 빈문자열이 아니면 배열에서 모든 이름 값 조회
   if (q != null) {
       q = q.toLowerCase();                                     // 소문자로 변환
       for (int i = 0; i < names.length; i++) {                 // 각 name 조사
           if (names[i].toLowerCase().startsWith(q) == true) {  // i번째 이름이 q로 시작하면
               if (hint == null)  { hint = names[i]; }          // hint 문자열에 저장
               else  { hint = hint + ", " + names[i]; }
           }
       }
   }
   if (hint != null)  out.print(hint);          // 응답 메시지로 결과 출력 (text string)
   else  out.print("no suggestion");
%>
```


Example 2

- 참고: PHP로 구현한 프로그램 (gethint.php)

```
<?php
   // 이름들의 배열 정의 (→ 또는 데이터베이스 검색 실시)
   $names = array("Anna", "Brittany", "Cinderella", "Diana", "Eva", "Fiona", "Gunda", …);
   $q = $_REQUEST["q"];                  // URL에서부터 q 파라미터(전송된 입력 값) 추출
   $hint = "";

   // $q가 빈문자열이 아니면 배열에서 모든 이름 값 조회
   if ($q !== "") {
       $q = strtolower($q);              // q를 소문자로 변환
       $len=strlen($q);
       foreach($names as $name) {        // 각 이름 값에 대해
           if (stristr($q, substr($name, 0, $len))) {  // q가 name의 접두사인지 검사(대소문자 무시)
               if ($hint === "")  { $hint = $name; }
               else  { $hint .= ", $name"; }        // 문자열 접합
           }
       }

   // Output "no suggestion" if no hint was found or output correct values
   echo $hint === "" ? "no suggestion" : $hint;
?>
```

Example 3

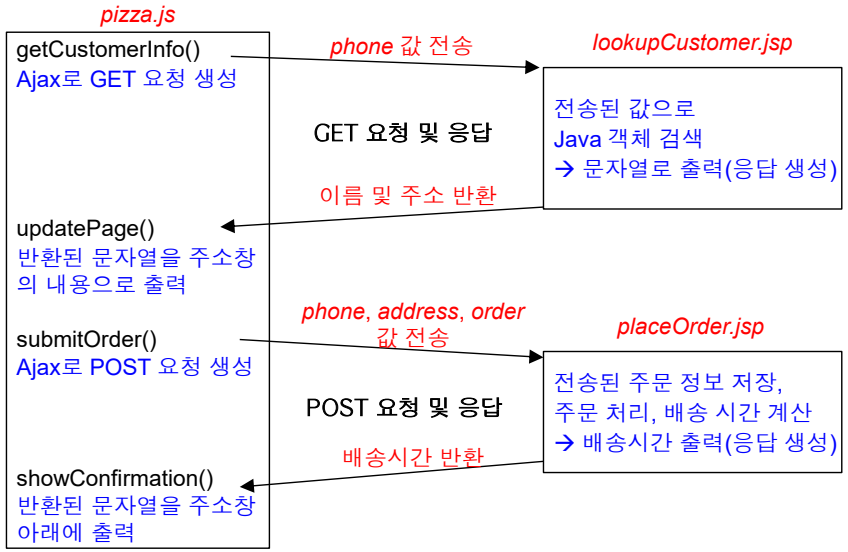
- 비동기적인 form field 값 입력 및 데이터 전송
  - 전화번호 입력 시 이름과 주소 검색 요청 (**비동기적인 GET 요청**)
  - 주문 정보를 비동기적으로 전송 (**비동기적인 POST 요청**)

Example 3

- 서버로부터 주문 처리 결과가 오면 화면에 출력 (**기존 페이지 갱신**)

Example 3

- 프로그램 실행 과정



Example 3

- Web page (pizza.html) : 입력 form

```
<html>
<head><title>Break Neck Pizza Delivery</title> ...
   <script type="text/javascript" src="pizza.js"> </script>
</head>
<body onLoad="document.forms[0].reset();">
   <div id="main-page">
      <p><img src="breakneck-logo_new.gif" /></p>
      <form id="order-form" action="">
         <p>Enter your phone number:
            <input type="text" name="phone" id="phone" onChange="getCustomerInfo();" />
         </p>
         <p>Type your order in here:</p>
         <p><textarea name="order" id="order" rows="6" cols="50"></textarea></p>
         <p>Your order will be delivered to:</p>
         <p><textarea name="address" id="address" rows="4" cols="50"></textarea></p>
         <p><input type="button" value="Order Pizza" onClick="submitOrder();" /></p>
      </form>
   </div>
</body>
</html>
```

다음 슬라이드 참조

입력 창에 내용 입력 시 getCustomerInfo() 함수 호출

focus가 바뀌었을 때 이벤트 처리

submit시 파라미터

주문 버튼 클릭 시 submitOrder() 함수 호출 → 요청 전송

Example 3

– JavaScript code (pizza.js)

```
var request = new XMLHttpRequest();                    // 편의 상 전역 변수로 선언
function getCustomerInfo() {                           // 고객 정보 검색 요청 (GET)
    var phone = document.getElementById("phone").value;
    var url = "lookupCustomer.jsp?phone=" +  phone;    // 호출할 서버 프로그램:
    request.open("GET", url, true);                    // 전화번호로 고객 정보 검색
    request.onreadystatechange = updatePage;
    request.send();
}

function updatePage() {                  // 고객 정보가 반환되면 실행되는 callback function
    if (request.readyState == 4) {
        if (request.status == 200) {
            var customerAddress = request.responseText;      // 응답 메시지 (고객 정보)
            document.getElementById("address").value = customerAddress;   // 주소 창에 출력
        } else {
            alert("Error! Request status is " + request.status);  }
    }
}
```


Example 3

– JavaScript code (pizza.js) (계속)

```
// jQuery 이용 시 아래와 같이 구현 가능

function getCustomerInfo() {                                  // 고객 정보 검색 요청 (GET)
    var phone = $("#phone").val();
    $.ajax({
            type: "GET",
            url: "lookupCustomer.jsp?phone=" +  phone;
            dataType: "text",
            success: function(address) {                      // callback function
                $("#address").val(address);                   // 응답 데이터를 주소 창에 출력
            },
            error: function(jqXHR) { alert("ERROR: " + JSON.stringify(jqXHR)); }
    });
}
```


Example 3

– JavaScript code (pizza.js) (계속)

```
function submitOrder() {
  // 입력된 주문 정보를 비동기적인 POST 요청으로 서버 프로그램(placeOrder.jsp)에 전송
  var phone = document.getElementById("phone").value;              // form에 입력된 값 추출
  var address = document.getElementById("address").value;
  var order = document.getElementById("order").value;
  request.open("POST", "placeOrder.jsp", true);                    // 호출할 서버 프로그램
  request.onreadystatechange = showConfirmation;
  request.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
  request.send("phone=" + phone + "&address=" + escape(address) +
    "&order=" + escape(order));                                    // form 입력 값 전송
}
function showConfirmation() {                                      // callback function
  if (request.readyState == 4) {
    if (request.status == 200) {     // 주문 처리 성공
      // 확인 메시지를 생성하여 기존 form 다음에 삽입(출력)
      var delTime = request.responseText;              // 응답 메시지: 배송시간
      pElement = document.createElement("p");          // DOM API 사용
      txtNode = document.createTextNode("Your order should arrive within "
                    + delTime + " minutes. Enjoy your pizza!");
      pElement.appendChild(txtNode);
      document.getElementById("main-page").appendChild(pElement);
    } else {  …  }  }  }             // 주문 처리 실패
```


Example 3

– JavaScript code (pizza.js) (계속)

```
// jQuery 이용 시 아래와 같이 구현 가능
function submitOrder() {
  var data = {
    "phone": $("#phone").val(),
    "address": $("#address").val(),
    "order": $("#order").val()
  };
  $.ajax({ type: "POST",
    url: "placeOrder.jsp",
    contentType: "application/x-www-form-urlencoded; charset=UTF-8", // default(생략 가능)
    data: data,
    dataType: "text",
    success: showConfirmation,
    error: function(jqXHR) { alert("ERROR: " + JSON.stringify(jqXHR)); }
  });
}
function showConfirmation() {                                  // callback function
  var p = document.createElement("p");
  p.innerHTML = `Your order should arrive within ${response} minutes. Enjoy your pizza!`;
  $("#main-page").append(p);
}
```

Example 3

– 서버 프로그램 (lookupCustomer.jsp) : 비동기적인 GET 요청 처리

```
<%@ page import="model.Customer" contentType="text/html; charset=utf-8" %>
<%
Customer[] customers = new Customer[4];
                    // Customer: name, street, city, state, zipCode 필드를 포함하는 도메인 클래스
customers[0] = new Customer("Doug Henderson",
    "7804 Jumping Hill Lane", "Dallas", "Texas", "75218");
customers[1] = new Customer("Mary Jenkins",
    "7081 Teakwood #24C", "Dallas", "Texas", "75182");
customers[2] = new Customer("John Jacobs",
    "234 East Rutherford Drive", "Topeka", "Kansas", "66608");
customers[3] = new Customer("Happy Traum",
    "876 Links Circle", "Topeka", "Kansas", "66608");          // Set up some customers

int i = (int)(Math.random()*4);
Customer c = customers[i];     // 고객 한 명을 임의로 선택
out.println(c.getName());      // 응답 메시지에 이름 출력 (name)
out.println(c.getAddress());   // 주소 정보 출력 (street + "\n" + city + ", " + state + " " + zipCode)
%>
```


Example 3

– 서버 프로그램 (placeOrder.jsp) : 비동기적인 POST 요청 처리

```
<%@ page contentType="text/html; charset=utf-8" %>
<%
String order = request.getParameter("order");              // order 및 address 파라미터 추출
String address = request.getParameter("address");
if (order.length() <= 0) {
    response.setStatus(400);              // bad request
    response.addHeader("Status", "No order was received");
    out.print(" ");
} else if (address.length() <= 0) {
    response.setStatus(400);              // bad request
    response.addHeader("Status", "No address was received");
    out.print(" ");
} else {
    // 주문 정보 저장, 피자 생성, 배송 예상 시간 계산 ...
    int deliveryTime = (int)(Math.random()*8 + 2);
    out.print(deliveryTime);                          // 응답 메시지에 배송 예상 시간 출력
}
%>
```


JSON

- JSON(JavaScript Object Notation)
  - 속성 이름-값 쌍으로 이루어진 데이터 객체들을 전달하기 위한 텍스트 형식의 표준 데이터 포맷(RFC 7159, ECMA-404)
    - 텍스트로 이루어져 있으므로 사람과 기계 모두 읽고 쓰기 쉬움
  - JavaScript 언어로부터 파생되어 JavaScript의 구문 형식을 따르지만 프로그래밍 언어나 플랫폼에 독립적
    - 많은 언어에서 JSON 데이터의 생성이나 해석(parsing)을 위한 수단(library)이 제공됨
    - 서로 다른 시스템 사이에 데이터를 교환하기에 적합
  - 비동기적인 브라우저-서버 통신(Ajax)에서 전송 데이터 포맷으로 사용됨
  - 인터넷 미디어 타입(MIME): application/json

JSON

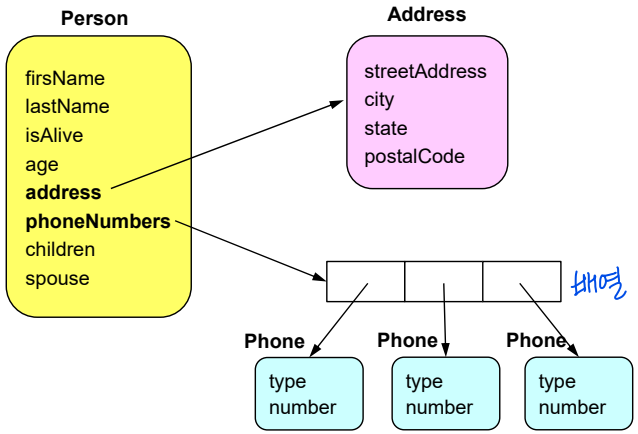
- Data types
  - 수(Number): 정수, 실수
  - 문자열(String): 0개 이상의 유니코드 문자들의 나열. 문자열은 큰 따옴표(")로 구분하며, 역슬래시(\)는 제어문자를 표현하기 위해 사용됨 (예: \n)
  - Boolean: true 또는 false 값
  - 배열(Array): 0개 이상의 임의의 타입의 요소들로 구성된 순서 있는 리스트
    - 요소는 JSON data type (배열, 객체 포함)
    - 대괄호 []로 나타내며, 각 요소는 쉼표로 구분
  - 객체(Object): 순서가 없는 이름-값 쌍의 집합 (associative array; map과 유사)
    - 이름(키)은 문자열(반드시 큰 따옴표 사용), 값은 JSON data type (배열, 객체 포함)
    - 중괄호 {}로 나타내며, 각 이름-값 쌍들은 쉼표로 구분
    - 이름과 값은 콜론(:)으로 구분
  - null: 값이 없음을 뜻함

JSON

- 예

```
{   -- person object
    "firstName": "John",                        -- string
    "lastName": "Smith",
    "isAlive": true,                            -- Boolean
    "age": 25,                                  -- number
    "address": {                                -- object
        "streetAddress": "21 2nd Street",
        "city": "New York",
        "state": "NY",
        "postalCode": "10021-3100"
    },
    "phoneNumbers": [                           -- array of 3 phone objects
        { "type": "home",  "number": "212-555-1234“  },
        { "type": "office",  "number": "646-555-4567“  },
        { "type": "mobile", "number": "123-456-7890“  }
    ],
    "children": [],                             -- empty array
    "spouse": null                              -- null value
}
```


JSON



JSON

- JSON vs. XML
  - XML과 JSON은 모두 구조적인 데이터를 표현하고 공유(전송)하기 위함
    - 자기 기술적(self-describing)이고 계층적인 구조
  - XML에서 엘리먼트와 tag 쌍을 주로 사용하여 데이터를 표현할 경우 JSON 형식의 데이터가 크기가 훨씬 작음 (→ 전송에 유리)
    - XML에서 엘리먼트 대신 속성을, tag 쌍 대신 축약형 태그('/>')를 활용할 경우 크기를 줄일 수 있음
    - 대량의 데이터는 압축을 실행하면 크기 상에 큰 차이가 없음
  - 일반적으로 JSON이 XML보다 parsing이 간단하고 빠름
  - XML은 XML Schema를 통해 XML 데이터에 대한 구조(스키마) 정의 가능
    - 태그 정의, 타입 제한, 사용자 정의 타입 지원, validation 가능
    - JSON은 현재 IETF에서 JSON Schema에 대한 표준안을 개발 중임
  - XML은 다양한 처리 기술(DOM, XPath, XSLT, CSS 등)이 존재함

JSON

- 예

```
{
  "firstName": "John",
  "lastName": "Smith",
  "age": 25,
  "address": {
      "streetAddress": "21 2nd Street",
      "city": "New York",
      "state": "NY",
      "postalCode": "10021-3100“
  },
  "phoneNumbers": [
      { "type": "home",
        "number": "212-555-1234" },
      { "type": "office",
        "number": "646-555-4567" } ],
  "gender" : {
      "type": "male“
  }
}
```

```
<person>          <!-- root element 필요 -->
  <firstName>John</firstName>
  <lastName>Smith</lastName>
  <age>25</age>
  <address>
     <streetAddr>21 2nd Street</streetAddr>
     <city>New York</city>
     <state>NY</state>
     <postalCode>10021</postalCode>
  </address>
  <phoneNumbers>
     <phone>
       <type>home</type>
       <number>212 555-1234</number>
     </phone>
     <phone>
       <type>office</type>
       <number>646 555-4567</number>
     </phone>
  </phoneNumbers>
  <gender><type>male</type></gender>
</person>
```

JSON

– 예: XML 속성 및 축약형 사용

```
{
  "firstName": "John",
  "lastName": "Smith",
  "age": 25,
  "address": {
     "streetAddress": "21 2nd Street",
     "city": "New York",
     "state": "NY",
     "postalCode": "10021-3100“
  },
  "phoneNumbers": [
     { "type": "home",
       "number": "212-555-1234" },
     { "type": "office",
       "number": "646-555-4567" } ],
  "gender" : {
     "type": "male“
  }
}
```

phone이라는 이름을 붙여줄 필요(X)

person이라는 이름을 붙여줄 필요(X)

```
<person
   firstName="John"
   lastName="Smith"
   age="25">
   <address
         streetAddress="21 2nd Street"
         city="New York"
         state="NY"
         postalCode="10021" />
   <phoneNumbers>
      <phone type="home"
             number="212 555-1234"/>
      <phone type=“office"
             number="646 555-4567"/>
   </phoneNumbers>
   <gender type="male"/>
</person>
```


Parsing JSON Data

❑ Parsing JSON text into JavaScript object (JSON text → JS Object)

– JavaScript의 eval() 함수를 사용하여 JSON text를 JavaScript 객체로 변환 가능

- var p = eval('(' + json_string + ')');
- 보안 취약성(code injection attack)과 Unicode 행종결문자 처리에 주의해야 함

– 브라우저에서 지원하는 native JSON parser(library)를 사용하는 것이 안전하고 효율적

- var obj = **JSON.parse(**json_string**);**
- json_string이 올바른 JSON text가 아닐 경우 오류(예외) 발생
  - ✓ try { … } catch (ex) { … } 구문을 이용하여 예외 처리
- Edge, Chrome, Opera 10+, Firefox 3.5+, Safari 4+ 등에서 지원

Parsing JSON Data

– 예:

JSON text를 JavaScript 객체로 변환

```
var obj = JSON.parse('{ "name":"John", "age":23, "city":"New York"}');
document.getElementById("demo").innerHTML = obj.name + ", " + obj.age;
                                  // 또는 obj[name] + ", " + obj[age]
```

– 날짜 String → Date type 변환

```
var text = '{ "name":"John", "birth":"2001-12-14", "city":"New York"}';
var obj = JSON.parse(text);
obj.birth = new Date(obj.birth);       // "2001-12-14"에 대한 Date 객체 생성
```

또는

```
var obj = JSON.parse(text, function(key, value) {     // reviver function
    if (key == "birth") { return new Date(value); }      // Date 객체 생성
       else { return value; }
    });
```

- 변환 결과: obj.birth == Sun Dec 14 2001 09:00:00 GMT+0900 (대한민국 표준시)

Parsing JSON Data

– Ajax 응답으로 JSON 데이터 활용

```
var xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
   if (this.readyState == 4 && this.status == 200) {
      var person = JSON.parse(this.responseText);     // JSON text → JS 객체 생성
      document.getElementById("demo").innerHTML = person.name;    // “John”
   }
};
xmlhttp.open("GET", "person.json", true);  // 또는 JSON text를 반환하는 프로그램 호출
xmlhttp.send();
```

```
{                                  -- person.json
   "name": "John",
   "age": 23,
   "pets": [
      { "animal": "dog", "name": "Fido" },
      { "animal": "cat", "name": "Felix" },
      { "animal": "hamster", "name": "Lightning" } ]
}
```

Parsing JSON Data

– 비교: Ajax의 응답으로 XML 데이터 사용

```
var xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
   if (this.readyState == 4 && this.status == 200) {
      var xmlDoc = this.responseXML;              // DOM tree의 Document 객체
      document.getElementById("demo").innerHTML =
        xmlDoc.getElementsByTagName("name")[0].textContent;    // DOM API 사용
   }
};
xmlhttp.open("GET", "person.xml", true); // 또는 XML data를 반환하는 프로그램 호출
xmlhttp.send();
```

```
<person>              <-- person.xml -->
   <name>John</name>
   <age>23</age>
   <pets>
      <pet animal="dog", name="Fido" />
      <pet animal="cat", name="Felix" />
      <pet animal="hamster", name="Lightning" />
   </pets>
</person>
```


Parsing JSON Data

- Parsing JSON text into Java object (JSON text → Java Object)
  - Java 용 JSON library 활용 (ex. JSON-simple, Jackson)

```
import org.json.simple.parser.*;                    // JSON-simple library

JSONParser parser = new JSONParser();      // parser 객체 생성
try {
   JSONObject jsonObj                          // parsing 후 결과를 HashMap으로 저장
      = (JSONObject) parser.parse(new FileReader("person.json"));

   String name = (String) jsonObj.get("name");              // 문자열
   int age = (Integer) jsonObj.get("age");                  // 숫자
   JSONArray msg = (JSONArray) jsonObj.get("pets");   // 배열(ArrayList) 생성
   Iterator<String> iterator = msg.iterator();
   while (iterator.hasNext()) {              // 배열의 원소를 하나씩 추출하여 사용
      String pet = iterator.next();
      …
   }
} catch (ParseException e) { e.printStackTrace(); }
```


Parsing JSON Data

– Jackson library 활용

- JSON text를 미리 정의된 클래스의 객체(POJO)로 자동 변환 가능
- https://mvnrepository.com/artifact/com.fasterxml.jackson.core에서 jackson-core, jackson-databind, jackson-annotation 파일을 다운로드

```
public class Person {
   private String name;
   private int age;
   private List<Pet> pets;

   // 기본 생성자와 setters & getters 정의
   public Person() {}
   public void setName(String name) {
      this.name = name;
   }
   public String getName() {
      return this.name;
   }
   …
}
```

```
import com.fasterxml.jackson.databind.*;

// Object mapper 객체 생성
ObjectMapper mapper = new ObjectMapper();

// Convert JSON text to POJO
Person person = mapper.readValue(
    new File("person.json"), Person.class);
```

→ Person 객체가 생성되고 person.json (JSON text)에 포함된 데이터를 가짐

Creating JSON Data

- JavaScript 객체로부터 JSON text 생성 (JS Object → JSON text)
  - **JSON.stringify()** 사용
    - var obj = { "name":"John", "age":23, "city":"New York"}; // JS 객체
      var jsonText = JSON.stringify(obj); // JSON text 생성
    - var arr = ["John", "Peter", "Sally", "Jane"]; // 배열
      var jsonText = JSON.stringify(arr);
    - var obj = { "name":"John", "today":new Date(), "city":"New York"};
      var jsonText = JSON.stringify(obj);
      - ✓ Date 타입을 String으로 변환 수행

Creating JSON Data

- Java (또는 JSP) 프로그램에서 JSON text 생성 (Java Object → JSON text)
  - JSON library 활용 (ex. JSON-simple)

```
var xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
        var person = JSON.parse(this.responseText);      // JSON text → JS 객체
        document.getElementById("demo").innerHTML = person.name;
    }
};
xmlhttp.open("GET", "demo.jsp", true);   xmlhttp.send();
```

```
<%@page contentType="text/html; charset=UTF-8"%>          <!-- demo.jsp -->
<%@page import="org.json.simple.JSONObject"%>
<%
    JSONObject obj = new JSONObject();                    // HashMap 생성
    obj.put("name", "John");
    obj.put("age", new Integer(23));
    obj.put("city", "New York");
    out.print(obj);                // { "name":"John", "age":23, "city":"New York"} 출력
%>
```


Creating JSON Data

  - 배열을 포함한 JSON text 생성

```
JSONObject obj = new JSONObject();
obj.put("name", "Peter");
obj.put("age", new Integer(21));
JSONArray list = new JSONArray();              // 배열을 위한 ArrayList 객체 생성
list.add("dog");    list.add("cat");    list.add("hamster ");
obj.put("pets", list);

try {
    FileWriter file = new FileWriter("person.json");
    file.write(obj.toJSONString());
    file.close();
} catch (IOException e) {
    e.printStackTrace();  }
```

```
{        -- person.json
   "age" : 21,   "name" : "Peter",   "pets" : ["dog", "cat", "hamster"]
}
```


Creating JSON Data

  - Jackson library 활용
    - POJO 객체를 JSON text로 변환

```
import com.fasterxml.jackson.databind.*;

// Object mapper 객체 생성
ObjectMapper mapper = new ObjectMapper();

// POJO 객체 생성
Person person = new Person("John", 23, "New York");

// Convert POJO to JSON text
String json = mapper.writeValueAsString(person);
```

→ 문자열 "{\"name\":\"John\", \"age\":23, \"city\":\"New York\"}" 생성

References

- Ajax
  - Brett McLaughlin, 박영록 역, Head Rush Ajax, O'Reilly, 2006.
  - https://en.wikipedia.org/wiki/Ajax_(programming) (Wikipedia)
  - *"Ajax: A New Approach to Web Applications" – by Jesse James Garrett*
  - https://www.w3schools.com/xml/ajax_intro.asp (W3Schools)

- JSON
  - http://www.json.org/ (JSON Homepage)
    - 각 언어별 JSON library 목록 제공
  - https://en.wikipedia.org/wiki/JSON (Wikipedia)
  - https://www.w3schools.com/js/js_json_intro.asp (W3Schools)
  - http://convertjson.com: JSON과 타 형식 간의 변환
  - https://github.com/fangyidong/json-simple: JSON-simple
    - http://juliusdavies.ca/json-simple-1.1.1-javadocs/ (Javadoc)
  - https://github.com/FasterXML/jackson: Jackson
    - https://www.baeldung.com/jackson (tutorial)