CONTENTS

"웹 개발자를 위한 체계적 개발 입문서"

03 Ajax

3.1 Ajax 개념

- Ajax(Asynchronous JavaScript and XML)의 약자로 비동기식 자바스크립트와 XML을 의미한다.
- Asynchronous + JavaScript + CSS + DOM + XML + XMLHttpRequest
- Ajax는 프로그래밍 언어가 아니다.
- Ajax는 다음을 조합하여 사용한다.
  ① 브라우저 내장 XMLHttpRequest 객체(웹 서버에서 데이터를 요청)
  ② JavaScript 및 HTML DOM(데이터 표시 또는 사용)

3.1.1 Ajax의 특징

- 웹 페이지 내에서 자바스크립트와 CSS, XHTML 등을 이용하여 XML로 데이터를 교환하고, 제어함으로써 사용자들이 웹 페이지를 '새로 고침' 하지 않고도 대화형의 웹 페이지 기능을 이용할 수 있게 하는 기술이다.
- 사용자의 PC에서 실행되는 리치 인터넷 애플리케이션(RIA) 기술의 하나이다.
- 웹 서버는 XML 기반의 웹 서비스 언어, 클라이언트는 자바스크립트를 사용하므로 브라우저와 웹 서버 간의 데이터 량이 줄어 응답성이 향상되고 웹 서버의 부담이 줄어들며, 웹에 별도로 프로그램을 설치할 필요 없이 일반 브라우저 화면에서 그대로 이용할 수 있다.
- 사용자가 직접 웹 상의 자료 위치를 편집하는 등 원하는 대로 제작할 수 있는 특징이 있다.
- 구글 맵이나 구글 서제스트 같은 기능이 에이잭스(AJAX)를 이용하는 대표적인 예. ActiveX나 Flash 등에 비해 가볍고 속도가 빠르다.
- 개발자 중심의 패러다임
  ① 새로운 기술이 아니라 기존의 기술 활용
  ② 기존 클라이언트/서버 환경 → 웹 개발 시 새로운 언어를 공부해야 함
  ③ Ajax → 기존의 노하우/기술력을 더욱 발전 : 개발자가 주도적

※ Ajax를 한마디로 정의하면 "JavaScript를 사용한 비동기 통신, 클라이언트와 서버간에 XML 데이터를 주고받는 기술" 이라고 할 수 있다

3.1 Ajax 개념

3.1.2 Ajax 장·단점

- 장점
  ① 웹 페이지의 속도 향상
  ② 서버의 처리가 완료될 때까지 기다리지 않고 처리 가능
  ③ 서버에서 Data만 전송하면 되므로 전체적인 코딩 양이 줄어듦
  ④ 기존 웹에서는 불가능했던 다양한 UI가 가능(예, 사진 공유 사이트 Flickr의 경우 사진의 제목이나 태그를 페이지 리로드 없이 수정가능)

- 단점
  ① 히스토리 관리가 안됨(브라우저의 Back 버튼을 효과적을 사용할 수 없음 - 보안에 신경 써야 함)
  ② 연속으로 데이터를 요청하면 서버 부하가 증가할 수 있음
  ③ XMLHttpRequest를 통해 통신하는 경우 사용자에게 아무런 진행 정보가 주어지지 않기 때문에, 요청이 완료되지 않았음에도 사용자가 페이지를 떠나거나 오작동할 우려가 발생

3.1 Ajax 개념

3.1.3 Ajax 기술

- Ajax 구현 기술

XHTML, CSS	Standards-based presentation using XHTML & CSS;
DOM	Dynamic display and interaction using the Document Object Model;
XML, XSLT	Data interchange and manipulation using XML and XSLT;
XMLHttpRequest	Asynchronous data retrieval using XMLHttpRequest;
JavaScript	And JavaScript binding everything together

- Ajax 요소 기술 : 브라우저가 요소 기술을 내장

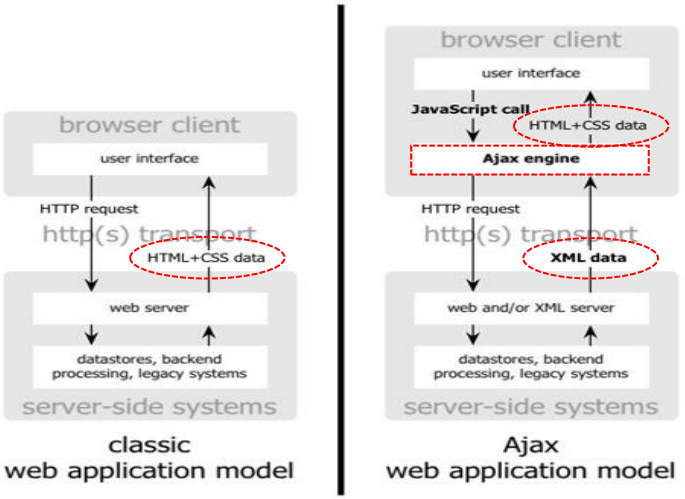
유저인터페이스	XHTML, DHTML, CSS, XSLT, DOM
통합제어/통신	JavaScript, XMLHttpRequest
데이터 처리	XML, JSON, CSV

3.1 Ajax 개념

3.1.4 Ajax 웹 애플리케이션 모델

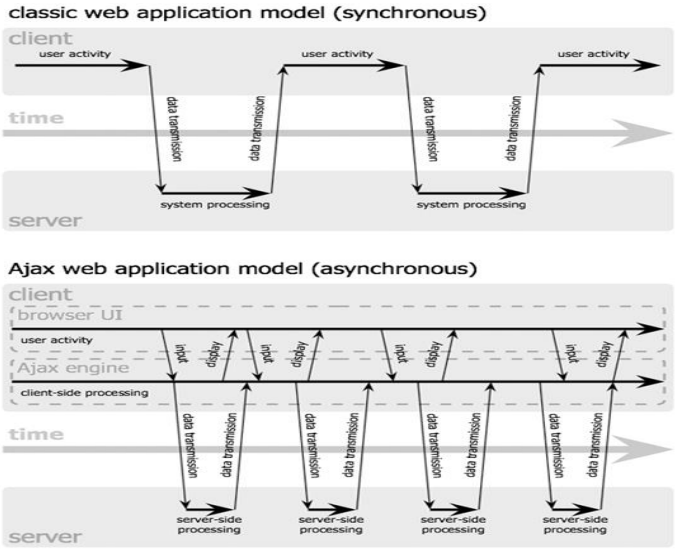
- Ajax 모델의 형태
  ① 전통적인 웹
  - 클릭할 때 마다 HTML 문서 요청하여 해석

  ② Ajax
  - 데이터 처리를 위해 서버와 통신
  - 비동기 방식으로 데이터만 전송 받음
  - 필요 시 동기통신 : JSON, CSN, Text
  - 클라이언트에서 HTML+CSS 해석



  ③ Ajax 엔진
  - HTML 문서처리 부분을 제외한 모든 엔진(추상적)

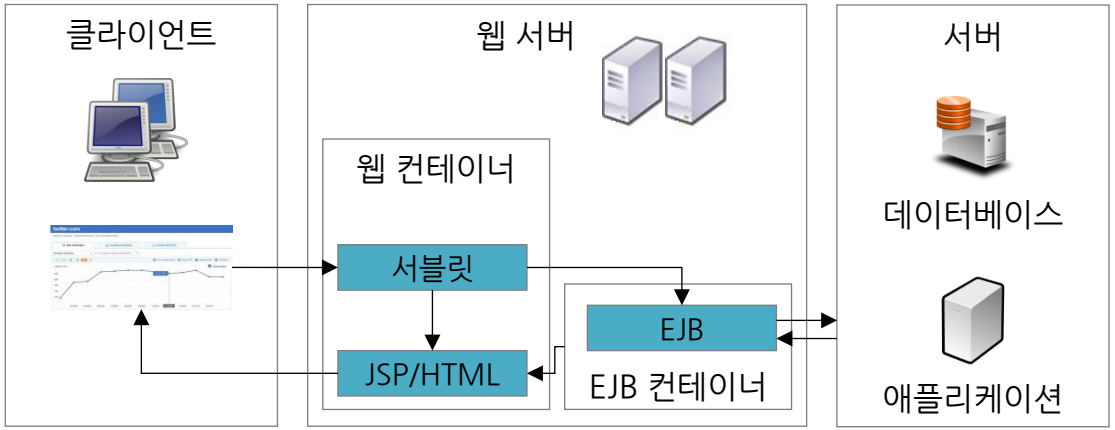
  ④ Asynchronous
  - XMLHttpRequest 객체
    * 서버처리 진행상태를 기록
  - Ajax 엔진이 서버와 통신
  - 브라우저는 서버와 무관
    * 유저 인터페이스만 수행
  - 데이터처리 등 비즈니스 로직은 서버가 담당
    * C#, Java 등으로 개발



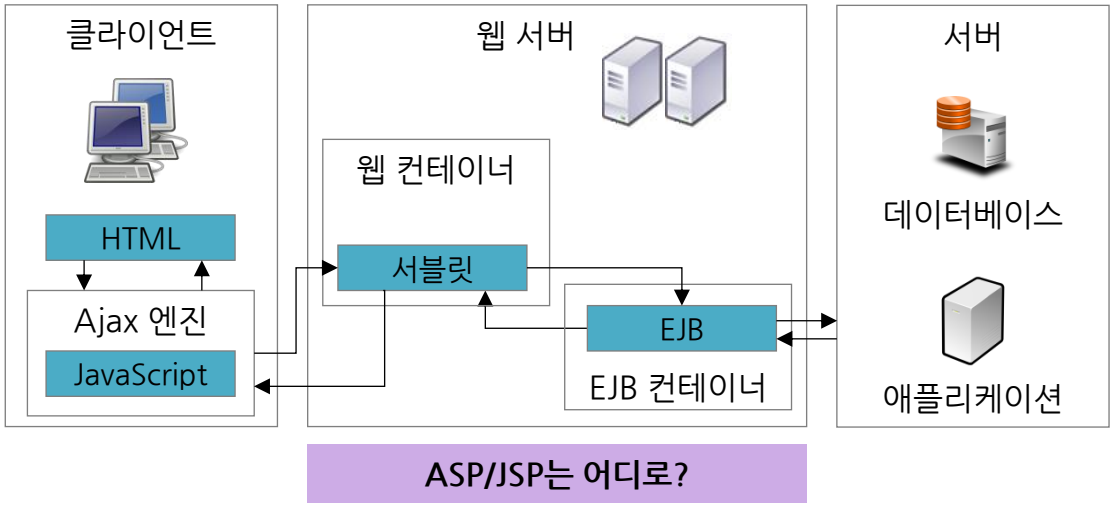
3.1 Ajax 개념

3.1.4 Ajax 웹 애플리케이션 모델

⑤ Ajax 모델과 MVC 패턴
- MVC(Model-View-Control) : 객체지향 컴포넌트 기반 개발방법
  * View : Presentation
  * Model : Business Logic
  * Control : View와 Model을 제어
- 전통적인 MVC 패턴 흐름 예(실행주체는 JSP)
  * View : JSP(사용자 확인 버튼)
  * Control : 서블릿(EJB 호출)
  * Model : EJB(회원정보 검색)
  * View : JSP(화면 표시)



- Ajax MVC 패턴
  * 클라이언트에서 View와 Control 수행
  * JSP 필요 없음
- Ajax MVC 패턴 흐름 예
  * View : 브라우저(사용자)
  * Control : Ajax엔진(서블릿 호출)
  * Control : 서블릿(EJB 호출)
  * Model : EJB(회원정보 검색)
  * Control : 서블릿 → Ajax엔진
  * View : 브라우저(화면 표시)



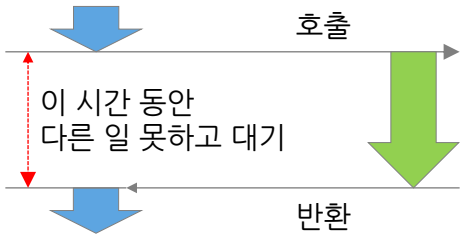
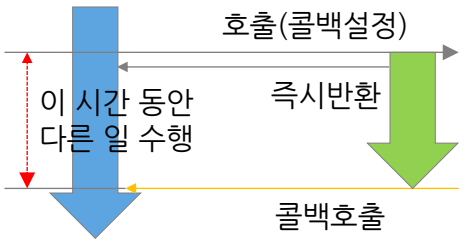
3.1 Ajax 개념

3.1.4 Ajax 웹 애플리케이션 모델(입출력 요청방식)

- 개요

	Blocking ← 함수의 즉시 리턴 여부 → Non-Blocking	
Synchronous ↕ 함수의 작업 완료 여부 Asynchronous	Read/Write	Read/Write (O-NONBLOCK)
	I/O Multiplexing (select, poll)	Async. I/O (return)

- 동작방식

Blocking I/O	Non-Blocking I/O
	
I/O 종료까지 Process 대기	I/O 종료와 관계없이 Process 수행

3.1 Ajax 개념

3.1.4 Ajax 웹 애플리케이션 모델(참고자료)

- 설명
  ① Blocking, Non-Blocking (구분 기준 : 통지)
  - Blocking : 애플리케이션 실행 시 운영체제 대기 큐에 들어가면서 요청에 대한 시스템 콜이 완료된 후에 응답을 보내는 방식
  - Non-Blocking : 애플리케이션 실행 시 운영체제 대기 큐에 들어가지 않고, 실행 여부와 관계없이 바로 응답을 보내는 방식

  ② Non-Blocking, Asynchronous
  - Non-Blocking
    * 시스템 콜이 반환될 때 실행된 결과와 함께 반환되는 방식
    * 요청에 대해 바로 응답할 수 있는 경우 응답, 바로 응답하기 힘든 경우 에러를 반환(203 : 요청이 성공적으로 처리되었지만, 다른 소스의 정보를 반환 중)
    * 에러를 받을 경우 데이터를 정상적으로 받을 때까지 계속해서 요청을 다시 전송
  - Asynchronous
    * 시스템 콜이 반환될 때 실행된 결과와 함께 반환되지 않는 방식
    * 요청에 대해 처리 완료의 여부와 관계없이 바로 응답
    * 이후 운영체제에서 응답할 준비가 완료되는 시점(예를 들어 네트워크로부터 데이터 받는 요청의 경우 데이터가 준비되는 경우)에 응답

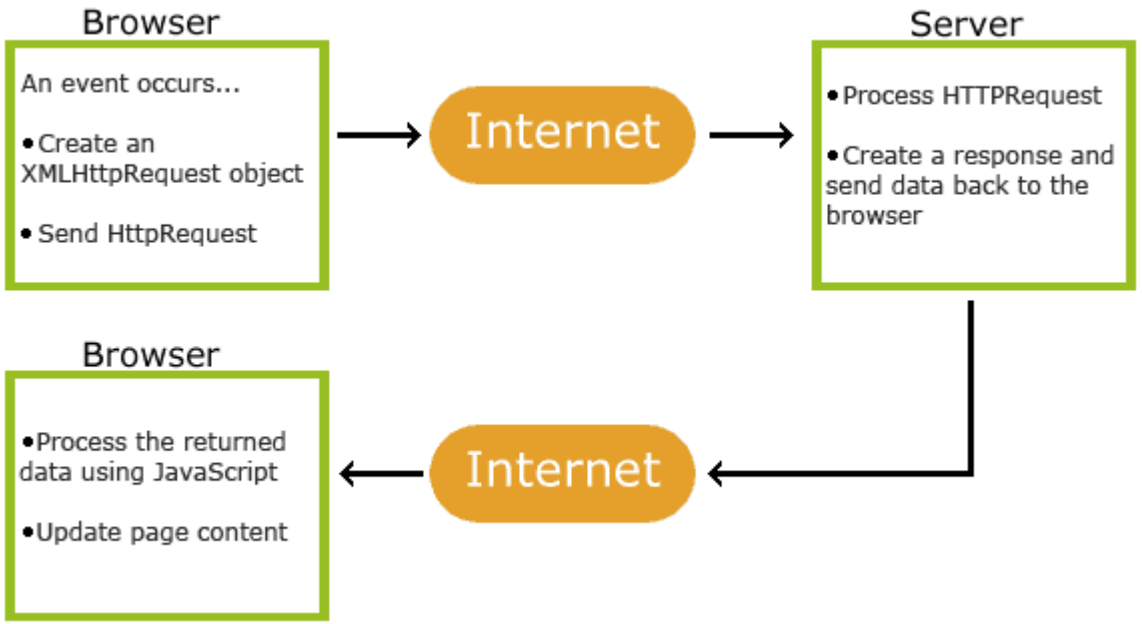
  ③ Synchronous, Asynchronous(구분 기준 : 작업순서)
  - Synchronous : 시스템 콜의 완료를 기다리는 방식
  - Asynchronous : 시스템 콜의 완료를 기다리지 않는 방식

  ④ Synchronous, Blocking
  - Synchronous : 시스템의 반환을 기다리는 동안 대기 큐에 머무는 것이 필수가 아닌 방식
  - Blocking : 시스템의 반환을 기다리는 동안 대기 큐에 머무는 것이 필수인 방식

- 요약
  ① Blocking : I/O 작업 상황(결과) 기다림(O), 대기 큐 필수 머무름(O)
  ② Non-Blocking : I/O 작업 상황(결과) 기다림(O), 대기 큐 필수 머무름(X)
  ③ Synchronous : I/O 작업 상황(결과) 기다림(O), 대기 큐 필수 머무름(X)
  ④ Asynchronous : I/O 작업 상황(결과) 기다림(X), 대기 큐 필수 머무름(X)

3.1 Ajax 개념

3.1.5 Ajax 작동 방식



- 작동 순서
  ① 웹 페이지에서 이벤트가 발생(페이지가 로드 되고 버튼이 클릭 됨)
  ② XMLHttpRequest 객체는 JavaScript에 의해 생성
  ③ XMLHttpRequest 객체는 웹 서버에 요청을 보냄
  ④ 서버가 요청을 처리
  ⑤ 서버가 웹 페이지에 응답을 보냄
  ⑥ 응답은 JavaScript에 의해 읽혀짐
  ⑦ 적절한 조치 (예 : 페이지 업데이트)가 JavaScript에 의해 수행

3.1 Ajax 개념

3.1.5 Ajax 작동 방식

1.ajax_info.html

```
<!DOCTYPE html>
<html>
<body>

<div id="demo">   -→ div 섹션 : 서버의 정보를 표시하는데 사용
  <h2>Let AJAX change this text</h2>
  <button type="button" onclick="loadDoc()">Change Content</button>   -→ button은 함수 호출
</div>

<script>
function loadDoc() {                -→ 이 함수는 웹 서버에 데이터를 요청하고 이를 표시
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
     document.getElementById("demo").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "https://www.w3schools.com/js/ajax_info.txt", true);
  xhttp.send();
}
</script>
</body>
</html>
```

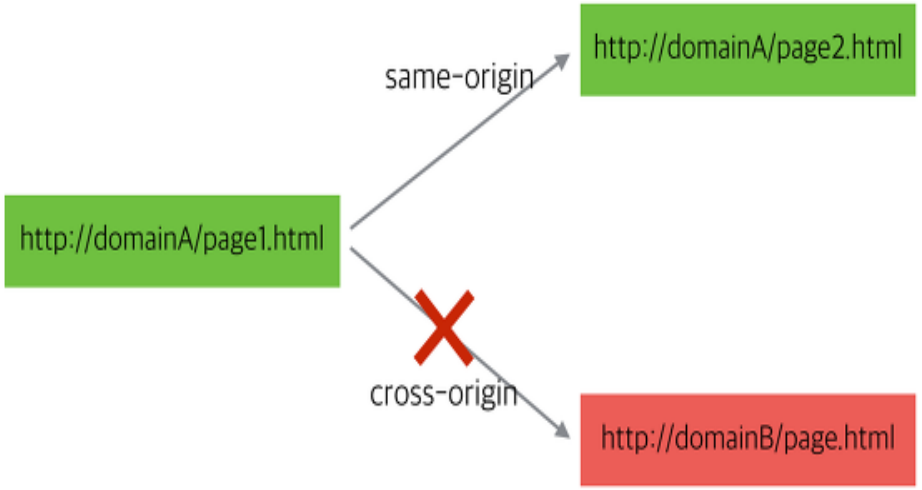
ajax_info.txt

```
<h1>AJAX</h1>
<p>AJAX is not a programming language.</p>
<p>AJAX is a technique for accessing web servers from a web page.</p>
<p>AJAX stands for Asynchronous JavaScript And XML.</p>
```

3.1 Ajax 개념

3.1.6 CORS(Cross Origin Resource Sharing)

- 웹 개발 시 자바스크립트로 외부 서버의 경로로 Ajax 요청을 보내면 에러가 나며 요청이 실패한다.
- 이는 외부로 요청이 되지 않는 자바스크립트 엔진 표준 스펙에 동일 출처 정책이라는 보안규칙 때문이다.
  ① 크롬 : No 'Access-Control-Allow-Origin' header is present on the requested resource. Origin '[요청한 도메인]' is therefore not allowed access.
  ② 파이어폭스 : 교차원본요청차단 : 동일 출처 정책으로 인해 [요청한 도메인]에 있는 원격 자원을 읽을 수 없다. 자원을 같은 도메인으로 이동시키거나 CORS를 활성화하여 해결할 수 있다.
- 동일출처정책(Same-Origin Polocy)란?
  ① 웹 애플리케이션 보안 모델에서 중요한 개념 중 하나
  ② 자바스크립트(XMLHttpRequest)로 다른 웹 페이지에 접근할 때는 같은 출처의 페이지에만 접근 가능
  ③ 같은 출처란, 프로토콜(URI Schema), 호스트명(host name), 포트(port number)가 같다는 의미로 웹페이지의 스크립트는 같은 서버에 있는 주소만 ajax 요청이 가능

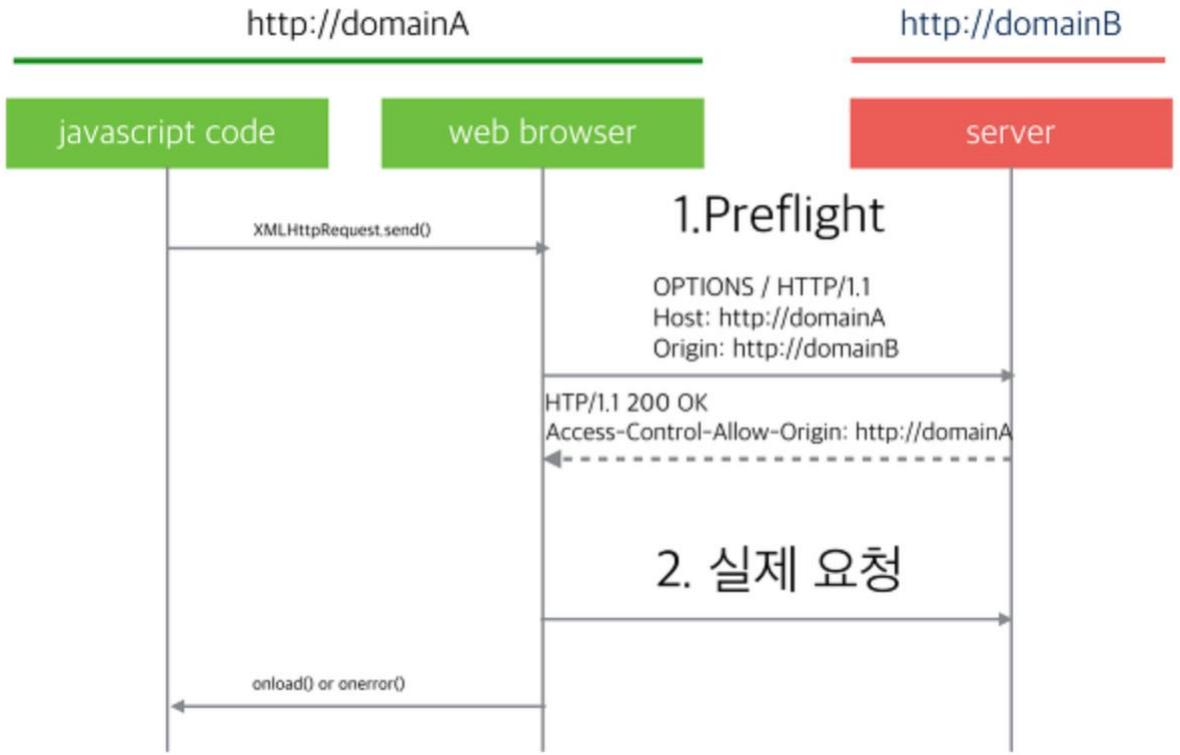


3.1 Ajax 개념

3.1.6 CORS(Cross Origin Resource Sharing)

- 문제 해결 방법
  ① 프런트에서 직접 다른 출처에 요청하지 않고, 같은 출처의 백앤드에 요청한 뒤, 백앤드에서 다른 출처로 요청(즉, 요청을 서버에 위임)
  ② 서버 측 요청 헤더에 Access-Control 관련 내용을 삽입

- CORS작동 방식



3.1 Ajax 개념

3.1.6 CORS(Cross Origin Resource Sharing)

- Ajax 에서 사용법(예시)

```
$.ajax({
  type: 'GET',
  url: My Url,
  contentType: 'application/json',
  dataTyep: 'jsonp',
  responseType: 'application/json',
  xhrFields: {
     withCredentials: false
  },

  headers: {
     'Access-Control-Allow-Credentials': true,
     'Access-Control-Allow-Origin': '*',
     'Access-Control-Allow-Methods': 'GET',
     'Access-Control-Allow-Headers':
'application/json',
  }
  success: function(data) {
      console.log(data);
  },
  error: function(error) {
      console.log("FAIL..==========");
  }
});
```

- Node.js(예시)

```
$npm install cors        -→ cors 라이브러리 설치

var cors = require('cors')

app.use(cors())

app.get('/products/:id', function (req, res, next) {
  res.json({msg: 'This is CORS-enabled for all origins!'})
})
```

3.1 Ajax 개념

3.1.6 CORS(Cross Origin Resource Sharing)

- Spring(Java Back-end(예시)) - 컨트롤러 단에서 헤더를 추가

```
response.setHeader("Access-Control-Allow-Origin", request.getHeader("Origin"))
response.setHeader("Access-Control-Allow-Credentials", "true");
response.setHeader("Access-Control-Allow-Methods", "POST, GET, OPTIONS, DELETE");
response.setHeader("Access-Control-Max-Age", "3600");
response.setHeader("Access-Control-Allow-Headers", "Content-Type, Accept, X-Requested-With, remember-me");
response.setHeader("Content-Type", "application/json");
response.setHeader("Accept", "application/json");
```

3.2 객체

3.2.1 XMLHttpRequest 객체

- 웹 브라우저와 웹 서버 간에 메소드가 데이터를 전송하는 객체 폼의 API이다.
- 브라우저의 자바스크립트 환경에 의해 제공된다.

① XMLHttpRequest 객체
- 최신의 모든 브라우저는 XMLHttpRequest 객체 지원
- XMLHttpRequest 객체는 백그라운드에서 웹 서버와 데이터를 교환하는 데 사용할 수 있음

② XMLHttpRequest 객체 만들기
- 모든 최신 브라우저 (Chrome, Firefox, IE7 +, Edge, Safari, Opera)에는 기본 제공 XMLHttpRequest 객체가 있음
- XMLHttpRequest 객체를 만들기 위한 구문

```
variable = new XMLHttpRequest();
```

2.ajax_xmlhttprequest.html

```
<!DOCTYPE html>
<html>
<body>

<h2>The XMLHttpRequest Object</h2>
<p id="demo">Let AJAX change this text.</p>
<button type="button" onclick="loadDoc()">Change Content</button>

<script>
function loadDoc() {
  Access-Control-Allow-Origin: *     -→ 있을 때와 없을 때
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "https://www.w3schools.com/js/ajax_info.txt", true);
  xhttp.send();
}
</script>
</body>
</html>
```

3.2 객체

3.2.1 XMLHttpRequest 객체

③ 도메인 간 액세스
- 보안상의 이유로 최신 브라우저는 도메인 전체에 대한 액세스를 허용하지 않음
- 즉,로드하려는 웹 페이지와 XML 파일은 모두 동일한 서버에 있어야 함

④ 이전 브라우저(IE5 및 IE6)
- 이전 버전의 Internet Explorer (5/6)는 XMLHttpRequest 객체 대신 ActiveX 객체를 사용
- IE5 및 IE6을 처리하려면 브라우저에서 XMLHttpRequest 객체를 지원하는지 확인하거나 ActiveX 객체를 만듦

```
variable = new ActiveXObject("Microsoft.XMLHTTP");
```

3.ajax_xmlhttp.html

```
<!DOCTYPE html>
<html>
<body>

<h2>The XMLHttpRequest Object</h2>
<p id="demo">Let AJAX change this text.</p>
<button type="button" onclick="loadDoc()">Change Content</button>

<script>
function loadDoc() {
  var xhttp;
  if (window.XMLHttpRequest) {
    // code for modern browsers
    xhttp = new XMLHttpRequest();
    } else {
    // code for IE6, IE5
    xhttp = new ActiveXObject("Microsoft.XMLHTTP");
  }
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "https://www.w3schools.com/js/ajax_info.txt", true);
  xhttp.send();
}
</script>
</body>
</html>
```

3.2 객체

3.2.1 XMLHttpRequest 객체

⑤ XMLHttpRequest 객체 메소드

메소드	설명
new XMLHttpRequest()	새 XMLHttpRequest 개체를 만듦
abort()	현재의 요청을 취소
getAllResponseHeaders()	헤더 정보를 반환
getResponseHeaders()	특정 헤더 정보를 반환
open(*method, url, async, user, psw*)	요청을 지정 메소드 : 요청 유형 GET 또는 POST URL : 파일 위치 비동기 : 참(비동기) 또는 거짓(동기) 사용자 : 사용자 이름(선택 사항) 암호 : 선택적 암호
send()	서버에 요청을 보냄 GET 요청에 사용
send(*string*)	서버에 요청을 보냄 POST 요청에 사용
setRequestHeader()	전송할 헤더에 레이블/값 쌍을 추가

3.2 객체

3.2.1 XMLHttpRequest 객체

⑥ XMLHttpRequest 객체 속성

속성	설명
onreadystatechange	readyState 속성이 변경될 때 호출할 기능을 정의
readyState	XMLHttpRequest의 상태를 유지 0 : 요청이 초기화 되지 않았음 1 : 서버 연결 설정 2 : 요청 수신 3 : 처리 요청 4 : 요청 완료 및 응답 준비
responseText	응답 데이터를 문자열로 반환
responseXML	응답 데이터를 XML 데이터로 반환
status(HTTP 상태 메시지)	요청의 상태 번호를 반환 200 : "확인" 403 : "금지" 404 : "찾을 수 없음"
statusText	상태 텍스트(예 : "확인" 또는 "찾아지지 않음") 을 반환

3.2 객체

3.2.1 XMLHttpRequest 객체

⑦ HTTP 상태 메시지

목록	메시지	설명
1xx : 정보	100 Continue	서버가 요청 헤더를 수신했으며, 클라이언트가 요청 본문을 계속 전송해야 함
	101 Switching Protocols	요청자가 서버에 프로토콜을 전환하도록 요청했음
	103 Checkpoint	다시 누적 가능한 요청 제안에 사용되어 중단된 PUT 또는 POST 요청을 재개함
2xx : 성공	200 OK	요청이 정상(HTTP 요청에 대한 표준 응답)
	201 Created	요청이 이행되고 새 리소스가 생성
	202 Accepted	요청이 처리용으로 승인되었지만, 처리가 완료되지 않았음
	203 Non-Authoritative Information	요청이 성공적으로 처리되었지만, 다른 소스의 정보를 반환하고 있음
	204 No Content	요청이 성공적으로 처리되었지만, 콘텐츠를 반환하지 않았음
	205 Reset Content	요청이 성공적으로 처리되었지만, 내용을 반환하지 않아 요청자가 문서 보기를 재 설정해야
	206 Partial Content	클라이언트가 보낸 범위 헤더로 인해 서버가 리소스의 일부만 배달하고 있음
3xx : 리디렉션	300 Multiple Choices	링크 목록. 사용자는 링크를 선택하고 해당 위치로 이동할 수 있습니다. 최대 5개의 주소
	301 Moved Permanently	요청한 페이지가 새 URL로 이동되었음
	302 Found	요청한 페이지가 일시적으로 새 URL로 이동되었음
	303 See Other	요청한 페이지는 다른 URL에서 찾을 수 있음
	304 Not Modified	요청한 페이지가 마지막으로 요청한 이후 수정되지 않았음을 나타냄
	306 Switch Proxy	더 이상 사용되지 않음
	307 Temporary Redirect	요청한 페이지가 일시적으로 새 URL로 이동되었음
	308 Resume Incomplete	다시 누적 가능한 요청 제안에 사용되어 중단된 PUT 또는 POST 요청을 재개함

3.2 객체

3.2.2 요청

- XMLHttpRequest 객체는 서버와 데이터를 교환하는데 사용한다.

① 서버에 요청 보내기
- 서버에 요청을 보내려면 XMLHttpRequest 객체의 open () 및 send () 메서드를 사용

```
xhttp.open("GET", "ajax_info.txt", true);
xhttp.send();
```

메소드	설명
open(*method, url, async*)	요청을 지정 메소드 : 요청 유형 GET 또는 POST URL : 서버(파일) 위치 비동기 : 참(비동기) 또는 거짓(동기)
send()	서버에 요청을 보냄 GET 요청에 사용
send(*string*)	서버에 요청을 보냄 POST 요청에 사용

② GET 또는 POST
- GET은 POST보다 간단하고 빠르며 대부분의 경우 사용할 수 있음
- 하지만, 다음의 경우 항상 POST 요청을 사용하여야 함
  * 캐시된 파일을 옵션으로 사용해야 할 때(서버에서 파일 또는 데이터베이스를 업데이트 함)
  * 서버에 대량의 데이터를 보낼 때(POST에는 크기 제한이 없음)
  * 사용자 입력(알 수 없는 문자를 포함할 수 있음)을 보내는 POST는 GET보다 견고하고 안전

3.2 객체

3.2.2 요청

③ GET 요청
- 간단한 GET 요청

```
xhttp.open("GET", "demo_get.asp", true);
xhttp.send();
```

4.ajax_demoget.html

```
<!DOCTYPE html>
<html>
<body>

<h2>The XMLHttpRequest Object</h2>

<button type="button" onclick="loadDoc()">Request data</button>

<p id="demo"></p>


<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "https://www.w3schools.com/js/demo_get.asp", true);
  xhttp.send();
}
</script>

</body>
</html>
```

3.2 객체

3.2.2 요청

③ GET 요청
- 캐시된 결과를 얻는 것을 방지하기 위해 URL에 고유한 ID를 추가

```
xhttp.open("GET", "demo_get.asp?t=" + Math.random(), true);
xhttp.send();
```

5.ajax_demogeturlID.html

```
<!DOCTYPE html>
<html>
<body>

<h2>The XMLHttpRequest Object</h2>

<button type="button" onclick="loadDoc()">Request data</button>

<p>Click the button several times to see if the time changes, or if the file is cached.</p>

<p id="demo"></p>

<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange=function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "https://www.w3schools.com/js/demo_get.asp?t=" + Math.random(), true);
  xhttp.send();
}
</script>

</body>
</html>
```

3.2 객체

3.2.2 요청

③ GET 요청

- GET 메소드를 사용하여 정보를 보내려면 URL에 정보를 추가

```
xhttp.open("GET", "demo_get2.asp?fname=Henry&lname=Ford", true);
xhttp.send();
```

6.ajax_demogeturl.html

```
<!DOCTYPE html>
<html>
<body>

<h2>The XMLHttpRequest Object</h2>

<button type="button" onclick="loadDoc()">Request data</button>

<p id="demo"></p>

<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "https://www.w3schools.com/js/demo_get2.asp?fname=Henry&lname=Ford", true);
  xhttp.send();
}
</script>

</body>
</html>
```

3.2 객체

3.2.2 요청

④ POST 요청
- 간단한 POST 요청

```
xhttp.open("POST", "demo_post.asp", true);
xhttp.send();
```

7.ajax_demopost.html

```
<!DOCTYPE html>
<html>
<body>

<h2>The XMLHttpRequest Object</h2>

<button type="button" onclick="loadDoc()">Request data</button>

<p id="demo"></p>

<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML = this.responseText;
    }
  };
  xhttp.open("POST", "https://www.w3schools.com/js/demo_post.asp", true);
  xhttp.send();
}
</script>

</body>
</html>
```

3.2 객체

3.2.2 요청

④ POST 요청

- HTML 폼처럼 POST 데이터를 보내려면 setRequestHeader ()로 HTTP 헤더를 추가. send () 메소드에서 전송할 데이터를 지정

```
xhttp.open("POST", "ajax_test.asp", true);
xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
xhttp.send("fname=Henry&lname=Ford");
```

8.ajax_demoposturl.html

```
<!DOCTYPE html>
<html>
<body>

<h2>The XMLHttpRequest Object</h2>

<button type="button" onclick="loadDoc()">Request data</button>

<p id="demo"></p>

<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML = this.responseText;
    }
  };
  xhttp.open("POST", "https://www.w3schools.com/js/demo_post2.asp", true);
  xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
  xhttp.send("fname=Henry&lname=Ford");
}
</script>

</body>
</html>
```

3.2 객체

3.2.2 요청

④ POST 요청
- 메소드

메소드	설명
setRequestHeader(header, *value*)	요청에 HTTP 헤더 추가 헤더 : 헤더 이름 지정 Value : 헤더 값 지정

⑤ URL - 서버상의 파일
- open() 메소드의 url 매개 변수는 서버의 파일에 대한 주소

```
xhttp.open("GET", "ajax_test.asp", true);
```

- 이 파일은 .txt 및 .xml과 같은 모든 종류의 파일이거나 .asp 및 .php와 같은 서버 스크립팅 파일일 수 있음
- 응답을 보내기 전에 서버에서 작업을 수행할 수 있음

⑥ 비동기 - True 또는 False
- 서버 요청은 비동기적으로 보내야 함
- open() 메소드의 async 매개 변수는 true로 설정해야 함

```
xhttp.open("GET", "ajax_test.asp", true);
```

- 비동기적으로 전송함으로써 JavaScript는 서버 응답을 기다리지 않아도 되지만 대신 다음을 수행 할 수 있음
  * 서버 응답을 기다리는 동안 다른 스크립트를 실행
  * 응답이 준비되면 응답을 처리

3.2 객체

3.2.2 요청

⑦ onreadystaechange 속성(비동기 처리 시 처리가 끝나면 응답을 받을 수 있음)
- XMLHttpRequest 객체를 사용하면 요청이 응답 받을 때 실행될 함수를 정의 할 수 있음
- 이 함수는 XMLHttpResponse 객체의 onreadystatechange 속성에 정의되어 있음

```
xhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
    document.getElementById("demo").innerHTML = this.responseText;
  }
};
xhttp.open("GET", "ajax_info.txt", true);
xhttp.send();
```

9.ajax_onreadystaechange.html

```
<!DOCTYPE html>
<html>
<body>

<div id="demo">
<h2>The XMLHttpRequest Object</h2>
<button type="button" onclick="loadDoc()">Change Content</button>
</div>

<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML =
      this.responseText;
    }
  };
  xhttp.open("GET", "https://www.w3schools.com/js/ajax_info.txt", true);
  xhttp.send();
}
</script>

</body>
</html>
```

3.2 객체

3.2.2 요청

⑧ 동기식 요청
- 동기식 요청을 실행하려면 open() 메소드의 세 번째 매개 변수를 false로 변경
- 때때로 async = false가 빠른 테스트에 사용되기도 함. 또한 이전 JavaScript 코드에서 동기식 요청을 찾을 수 있음
- 코드가 서버 완료를 기다리므로 onreadystatechange 함수가 필요하지 않음

```
xhttp.open("GET", "ajax_info.txt", false);
```

10.ajax_getasync.html

```
<!DOCTYPE html>
<html>
<body>

<h2>The XMLHttpRequest Object</h2>

<p id="demo">Let AJAX change this text.</p>

<button type="button" onclick="loadDoc()">Change Content</button>

<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.open("GET", "https://www.w3schools.com/js/ajax_info.txt", false);
  xhttp.send();
  document.getElementById("demo").innerHTML = xhttp.responseText;
}
</script>

</body>
</html>
```

※ 서버 응답이 준비 될 때까지 JavaScript가 실행을 중지하기 때문에 동기 XMLHttpRequest (async = false)는 권장되지 않음.
서버가 사용 중이거나 느린 경우 응용 프로그램이 중지되거나 중단됨.
동기 XMLHttpRequest가 웹 표준에서 제거되는 과정에 있지만 이 프로세스는 수년이 걸릴 수 있음

3.2 객체

3.2.3 응답

- XMLHttpRequest 객체는 서버와 데이터를 교환하는데 사용한다.

① onreadystaechange 속성
- readyState 속성은 XMLHttpRequest의 상태를 유지함
- onreadstatechange 속성은 readyState가 변경될 때 실행할 기능을 정의함
- status 속성 및 statusText 속성은 XMLHttpRequest 개체의 상태를 유지함

속성	설명
onreadystatechange	readyState 속성이 변경될 때 호출할 기능을 정의
readyState	XMLHttpRequest의 상태를 유지 0 : 요청이 초기화되지 않았음 1 : 서버 연결 설정 2 : 요청 수신 3 : 처리 요청 4 : 요청 완료 및 응답 준비
status	200: "OK" 403: "Forbidden" 404: "Page not found"
statusText	상태 텍스트(예: "확인" 또는 "찾아지지 않음")를 반환

3.2 객체

3.2.3 응답

① onreadystaechange 속성
- onreadystatechange 함수는 readyState가 변경 될 때마다 호출
- readyState가 4이고 상태가 200이면 응답이 준비됨(예시)

11.ajax_onreadystaechange2.html

```
<!DOCTYPE html>
<html>
<body>

<div id="demo">
<h2>The XMLHttpRequest Object</h2>
<button type="button" onclick="loadDoc()">Change Content</button>
</div>

<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML =
      this.responseText;
    }
  };
  xhttp.open("GET", "https://www.w3schools.com/js/ajax_info.txt", true);
  xhttp.send();
}
</script>

</body>
</html>
```

3.2 객체

3.2.3 응답

② 콜백 함수 사용
- 콜백 함수는 다른 함수에 매개 변수로 전달되는 함수
- 웹 사이트에 두 개 이상의 Ajax 작업이 있는 경우 XMLHttpRequest 객체를 실행하기 위한 함수 하나와 Ajax 작업에 대해 하나의 콜백 함수를 만들어야 함
- 함수 호출에는 URL과 응답 준비가 되었을 때 호출할 함수가 있어야 함

12.ajax_callback.html

```
<!DOCTYPE html>
<html>
<body>

<div id="demo">

<h2>The XMLHttpRequest Object</h2>

<button type="button"
onclick="loadDoc('https://www.w3schools.com/js/ajax_info.txt', myFunction)">Change Content
</button>
</div>

<script>
function loadDoc(url, cFunction) {
  var xhttp;
  xhttp=new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      cFunction(this);
    }
  };
  xhttp.open("GET", url, true);
  xhttp.send();
}
function myFunction(xhttp) {
  document.getElementById("demo").innerHTML =
  xhttp.responseText;
}
</script>
</body>
</html>
```

3.2 객체

3.2.3 응답

③ 서버 응답 특성

속성	설명
responseText	응답 데이터를 문자열로 가져오기
responseXML	응답 데이터를 XML 데이터로 가져오기

④ 서버 응답 메소드

메소드	설명
getResponseHeader()	서버 리소스에서 특정 헤더 정보를 반환
getAllResponseHeader()	서버 리소스에서 모든 헤더 정보를 반환

3.2 객체

3.2.3 응답

⑤ responseText 속성
- response 속성은 자바 스크립트 문자열로 서버 응답을 반환

```
document.getElementById("demo").innerHTML = xhttp.responseText;
```

13.ajax_responseText.html

```
<!DOCTYPE html>
<html>
<body>

<div id="demo">
<h2>The XMLHttpRequest Object</h2>
<button type="button" onclick="loadDoc()">Change Content</button>
</div>

<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML =
      this.responseText;
    }
  };
  xhttp.open("GET", "https://www.w3schools.com/js/ajax_info.txt", true);
  xhttp.send();
}
</script>

</body>
</html>
```

3.2 객체

3.2.3 응답

⑥ responseXML 속성
- XMLHttpRequest 객체에는 내장된 XML 파서가 있음
- responseXML 속성은 XML의 DOM 객체로 서버 응답을 반환
- 이 속성을 사용하면 응답을 XML DOM 객체로 파싱할 수 있음

14.ajax_responseXML.html

```
<!DOCTYPE html>
<html>
<body>

<h2>The XMLHttpRequest Object</h2>

<p id="demo"></p>

<script>
var xhttp, xmlDoc, txt, x, i;
xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
if (this.readyState == 4 && this.status == 200) {
  xmlDoc = this.responseXML;
  txt = "";
  x = xmlDoc.getElementsByTagName("ARTIST");
  for (i = 0; i < x.length; i++) {
    txt = txt + x[i].childNodes[0].nodeValue + "<br>";
  }
  document.getElementById("demo").innerHTML = txt;
  }
};
xhttp.open("GET", "https://www.w3schools.com/js/cd_catalog.xml", true);
xhttp.send();
</script>

</body>
</html>
```

3.2 객체

3.2.3 응답

⑦ getAllResponseHeaders() 메소드
- getAllResponseHeaders() 메소드는 서버 응답에서 모든 헤더 정보를 반환

15.ajax_getAllResponseHeaders.html

```
<!DOCTYPE html>
<html>
<body>

<h2>The XMLHttpRequest Object</h2>

<p>The getAllResponseHeaders() function returns all the header information of a resource, like length,
server-type, content-type, last-modified, etc:</p>

<p id="demo"></p>

<script>
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
    document.getElementById("demo").innerHTML =
    this.getAllResponseHeaders();
  }
};
xhttp.open("GET", "https://www.w3schools.com/js/ajax_info.txt", true);
xhttp.send();
</script>

</body>
</html>
```

3.2 객체

3.2.3 응답

⑧ getResponseHeaders() 메소드
- getResponseHeaders() 메소드는 서버 응답에서 특정 헤더 정보를 반환

16.ajax_getResponseHeaders.html

```
<!DOCTYPE html>
<html>
<body>

<h2>The XMLHttpRequest Object</h2>

<p>The getResponseHeader() function is used to return specific header information from a resource, like length, server-type, content-type, last-modified, etc:</p>

<p>Last modified: <span id="demo"></span></p>

<script>
var xhttp=new XMLHttpRequest();
xhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
    document.getElementById("demo").innerHTML =
    this.getResponseHeader("Last-Modified");
  }
};
xhttp.open("GET", "https://www.w3schools.com/js/ajax_info.txt", true);
xhttp.send();
</script>

</body>
</html>
```

3.3 적용 예

3.3.1 Ajax 적용

① XML
- 웹 페이지가 Ajax가 있는 XML 파일에서 정보를 가져올 수 있는 방법

17.ajax_xml.html

```
<!DOCTYPE html>
<html>
<style>
table,th,td {
  border : 1px solid black;
  border-collapse: collapse;
}
th,td {
  padding: 5px;
}
</style>
<body>

<h2>The XMLHttpRequest Object</h2>

<button type="button" onclick="loadDoc()">Get my CD collection</button>
<br><br>
<table id="demo"></table>

<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      myFunction(this);
    }
  };
  xhttp.open("GET", "https://www.w3schools.com/js/cd_catalog.xml", true);
  xhttp.send();
}
```

3.3 적용 예

3.3.1 Ajax 적용

① XML
- 웹 페이지가 Ajax가 있는 XML 파일에서 정보를 가져올 수 있는 방법

17.ajax_xml.html

```
function myFunction(xml) {
  var i;
  var xmlDoc = xml.responseXML;
  var table="<tr><th>Artist</th><th>Title</th></tr>";
  var x = xmlDoc.getElementsByTagName("CD");
  for (i = 0; i <x.length; i++) {
    table += "<tr><td>" +
    x[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue +
    "</td><td>" +
    x[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue +
    "</td></tr>";
  }
  document.getElementById("demo").innerHTML = table;
}
</script>

</body>
</html>
```

* 사용자가 "CD 정보 얻기" 버튼을 클릭하면 loadDoc() 함수가 실행
* loadDoc() 함수는 XMLHttpRequest 객체를 만들고 서버 응답이 준비될 때 실행될 추가 한 다음 서버에 요청
* 서버 응답이 준비되면 HTML 테이블이 작성되고 XML 파일에서 nodes(요소)가 추출되고 마지막으로 XML 데이터로 채워진 HTML 테이블로 "demo" 요소가 업데이트

3.3 적용 예

3.3.1 Ajax 적용

② PHP
- 사용자가 입력 필드에 문자를 입력하는 동안 웹 페이지가 웹 서버와 통신하는 방법

18.ajax_php.html

```
<!DOCTYPE html>
<html>
<body>

<h2>The XMLHttpRequest Object</h2>

<h3>Start typing a name in the input field below:</h3>

<p>Suggestions: <span id="txtHint"></span></p>

<p>First name: <input type="text" id="txt1" onkeyup="showHint(this.value)"></p>

<script>
function showHint(str) {
  var xhttp;
  if (str.length == 0) {
    document.getElementById("txtHint").innerHTML = "";
    return;
  }
  xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("txtHint").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "https://www.w3schools.com/js/gethint.php?q="+str, true);
  xhttp.send();
}
</script>

</body>
</html>
```

3.3 적용 예

3.3.1 Ajax 적용

② PHP

- 사용자가 입력 필드에 문자를 입력하는 동안 웹 페이지가 웹 서버와 통신하는 방법
  * 사용자가 입력 필드에 문자를 입력하면 "showHint()"라는 함수가 실행
  * 이 함수는 onkeyup 이벤트에 의해 트리거 됨
  * 입력 필드에 Data가 입력될 때 수행 절차
    . XMLHttpRequest 객체 만들기
    . 서버 응답이 준비되었을 때 실행될 함수를 생성
    . 요청을 서버의 PHP파일(gethint.php)로 보냄
    . q 매개 변수가 "gethint.php? q = " + str
    . str 변수는 입력 필드의 내용을 보유

3.3 적용 예

3.3.1 Ajax 적용

③ ASP

- 사용자가 입력 필드에 문자를 입력하는 동안 웹 페이지가 웹 서버와 통신하는 방법

19.ajax_asp.html

```
<!DOCTYPE html>
<html>
<body>

<h2>The XMLHttpRequest Object</h2>

<h3>Start typing a name in the input field below:</h3>

<p>Suggestions: <span id="txtHint"></span></p>

<p>First name: <input type="text" id="txt1" onkeyup="showHint(this.value)"></p>

<script>
function showHint(str) {
  var xhttp;
  if (str.length == 0) {
    document.getElementById("txtHint").innerHTML = "";
    return;
  }
  xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("txtHint").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "https://www.w3schools.com/js/gethint.asp?q="+str, true);
  xhttp.send();
}
</script>

</body>
</html>
```

3.3 적용 예

3.3.1 Ajax 적용

③ ASP

- 사용자가 입력 필드에 문자를 입력하는 동안 웹 페이지가 웹 서버와 통신하는 방법
  * 사용자가 입력 필드에 문자를 입력하면 "showHint()"라는 함수가 실행
  * 이 함수는 onkeyup 이벤트에 의해 트리거 됨
  * 입력 필드에 Data가 입력될 때 수행 절차
    . XMLHttpRequest 객체 만들기
    . 서버 응답이 준비되었을 때 실행될 함수를 생성
    . 요청을 서버의 ASP파일(gethint.asp)로 보냄
    . q 매개 변수가 "gethint.asp? q = " + str
    . str 변수는 입력 필드의 내용을 보유

3.3 적용 예

3.3.1 Ajax 적용

④ Database
- 웹 페이지가 Ajax를 사용하여 데이터베이스에서 정보를 가져올 수 있는 방법

20.ajax_database.html

```
<!DOCTYPE html>
<html>
<style>
table,th,td {
  border : 1px solid black;
  border-collapse: collapse;
}
th,td {
  padding: 5px;
}
</style>
<body>

<h2>The XMLHttpRequest Object</h2>

<form action="">
<select name="customers" onchange="showCustomer(this.value)">
<option value="">Select a customer:</option>
<option value="ALFKI">Alfreds Futterkiste</option>
<option value="NORTS ">North/South</option>
<option value="WOLZA">Wolski Zajazd</option>
</select>
</form>
<br>
<div id="txtHint">Customer info will be listed here...</div>
```

3.3 적용 예

3.3.1 Ajax 적용

④ Database
- 웹 페이지가 Ajax를 사용하여 데이터베이스에서 정보를 가져올 수 있는 방법

20.ajax_database.html

```
<script>
function showCustomer(str) {
  var xhttp;
  if (str == "") {
    document.getElementById("txtHint").innerHTML = "";
    return;
  }
  xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("txtHint").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", " https://www.w3schools.com/js/getcustomer.asp?q="+str, true);
  xhttp.send();
}
</script>

</body>
</html>
```

* 사용자가 Drop Down List에서 고객을 선택하면 "showCustomer()"라는 함수를 실행
* 함수는 "onchange" 이벤트에 의해 트리거 됨
* showCustomer() 함수는 다음을 수행
  . 고객이 선택되었는지 확인
  . XMLHttpRequest 객체 만들기
  . 서버 응답이 준비되었을 때 실행될 함수를 생성
  . 요청을 서버의 파일로 전송
  . 매개 변수 (q)가 URL에 추가

3.3 적용 예

3.3.2 Ajax 애플리케이션(JavaScript)

① HTML 테이블에 XML 데이터 표시하기
- 각 <CD> 요소를 반복하며 HTML 테이블에 <ARTIST> 및 <TITLE> 요소의 값 표시

21.ajax_apphtmltable.html

```
<!DOCTYPE html>
<html>
<style>
table,th,td {
  border : 1px solid black;
  border-collapse: collapse;
}
th,td {
  padding: 5px;
}
</style>
<body>

<button type="button" onclick="loadXMLDoc()">Get my CD collection</button>
<br><br>
<table id="demo"></table>

<script>
function loadXMLDoc() {
  var xmlhttp = new XMLHttpRequest();
  xmlhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      myFunction(this);
    }
  };
  xmlhttp.open("GET", "https://www.w3schools.com/js/cd_catalog.xml", true);
  xmlhttp.send();
}
```

3.3 적용 예

3.3.2 Ajax 애플리케이션(JavaScript)

① HTML 테이블에 XML 데이터 표시하기
- 각 <CD> 요소를 반복하며 HTML 테이블에 <ARTIST> 및 <TITLE> 요소의 값 표시

21.ajax_apphtmltable.html

```
function myFunction(xml) {
  var i;
  var xmlDoc = xml.responseXML;
  var table="<tr><th>Artist</th><th>Title</th></tr>";
  var x = xmlDoc.getElementsByTagName("CD");
  for (i = 0; i <x.length; i++) {
    table += "<tr><td>" +
    x[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue +
    "</td><td>" +
    x[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue +
    "</td></tr>";
  }
  document.getElementById("demo").innerHTML = table;
}
</script>

</body>
</html>
```

3.3 적용 예

3.3.2 Ajax 애플리케이션(JavaScript)

② 첫 번째 CD를 HTML div 요소에 표시하기
- 함수를 사용하여 id = "showCD"인 HTML 요소에 첫 번째 CD 요소를 표시

22.ajax_appcdhtml.html

```
<!DOCTYPE html>
<html>
<body>

<div id='showCD'></div>

<script>
displayCD(0);

function displayCD(i) {
   var xmlhttp = new XMLHttpRequest();
   xmlhttp.onreadystatechange = function() {
      if (this.readyState == 4 && this.status == 200) {
         myFunction(this, i);
      }
   };
   xmlhttp.open("GET", "https://www.w3schools.com/js/cd_catalog.xml", true);
   xmlhttp.send();
}

function myFunction(xml, i) {
   var xmlDoc = xml.responseXML;
   x = xmlDoc.getElementsByTagName("CD");
   document.getElementById("showCD").innerHTML =
   "Artist: " +
   x[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue +
   "<br>Title: " +
   x[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue +
   "<br>Year: " +
   x[i].getElementsByTagName("YEAR")[0].childNodes[0].nodeValue;
}
</script>

</body>
</html>
}
```

3.3 적용 예

3.3.2 Ajax 애플리케이션(JavaScript)

③ CD 간 이동하기
- CD 사이를 탐색하기 위해 next() 및 previous() 함수를 추가

23.ajax_appcdmove.html

```
<!DOCTYPE html>
<html>
<body>

<div id='showCD'></div><br>
<input type="button" onclick="previous()" value="<<">
<input type="button" onclick="next()" value=">>">

<script>
var i = 0, len;
displayCD(i);

function displayCD(i) {
    var xmlhttp = new XMLHttpRequest();
    xmlhttp.onreadystatechange = function() {
        if (this.readyState == 4 && this.status == 200) {
            myFunction(this, i);
        }
    };
    xmlhttp.open("GET", "https://www.w3schools.com/js/cd_catalog.xml", true);
    xmlhttp.send();
}
```

3.3 적용 예

3.3.2 Ajax 애플리케이션(JavaScript)

③ CD 간 이동하기
- CD 사이를 탐색하기 위해 next() 및 previous() 함수를 추가

23.ajax_appcdmove.html

```
function myFunction(xml, i) {
    var xmlDoc = xml.responseXML;
    x = xmlDoc.getElementsByTagName("CD");
    len = x.length;
    document.getElementById("showCD").innerHTML =
    "Artist: " +
    x[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue +
    "<br>Title: " +
    x[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue +
    "<br>Year: " +
    x[i].getElementsByTagName("YEAR")[0].childNodes[0].nodeValue;
}

function next() {
if (i < len-1) {
  i++;
  displayCD(i);
  }
}

function previous() {
if (i > 0) {
  i--;
  displayCD(i);
  }
}
</script>

</body>
</html>
```

3.3 적용 예

3.3.2 Ajax 애플리케이션(JavaScript)

④ CD를 클릭하면 앨범 정보를 표시하기
- 사용자가 CD를 클릭할 때 앨범 정보를 표시하는 방법

24.ajax_appcdclick.html

```
<!DOCTYPE html>
<html>
<style>
table,th,td {
  border : 1px solid black;
  border-collapse: collapse;
}
th,td {
  padding: 5px;
}
</style>
<body>

<p>Click on a CD to display album information.</p>
<p id='showCD'></p>
<table id="demo"></table>

<script>
var x,xmlhttp,xmlDoc
xmlhttp = new XMLHttpRequest();
xmlhttp.open("GET", "https://www.w3schools.com/js/cd_catalog.xml", false);
xmlhttp.send();
xmlDoc = xmlhttp.responseXML;
x = xmlDoc.getElementsByTagName("CD");
table="<tr><th>Artist</th><th>Title</th></tr>";
```

3.3 적용 예

3.3.2 Ajax 애플리케이션(JavaScript)

④ CD를 클릭하면 앨범 정보를 표시하기
- 사용자가 CD를 클릭할 때 앨범 정보를 표시하는 방법

24.ajax_appcdclick.html

```
for (i = 0; i <x.length; i++) {
  table += "<tr onclick='displayCD(" + i + ")'><td>";
  table += x[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue;
  table += "</td><td>";
  table +=  x[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue;
  table += "</td></tr>";
}
document.getElementById("demo").innerHTML = table;

function displayCD(i) {
  document.getElementById("showCD").innerHTML =
  "Artist: " +
  x[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue +
  "<br>Title: " +
  x[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue +
  "<br>Year: " +
  x[i].getElementsByTagName("YEAR")[0].childNodes[0].nodeValue;
}
</script>

</body>
</html>
```

3.3 적용 예

3.3.3 Ajax 애플리케이션(PHP)

① RSS 리더

- RSS 리더는 다시 로드 하지 않고, 웹 페이지에 로드 되는 RSS 리더를 보여줌 - showRSS() 함수

25.ajax_apprssreader.html

```
<!DOCTYPE html>
<html>
<head>
<script>
function showRSS(str) {
  if (str.length==0) {
    document.getElementById("rssOutput").innerHTML="";
    return;
  }
  if (window.XMLHttpRequest) {
    // code for IE7+, Firefox, Chrome, Opera, Safari
    xmlhttp=new XMLHttpRequest();
  } else {  // code for IE6, IE5
    xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
  xmlhttp.onreadystatechange=function() {
    if (this.readyState==4 && this.status==200) {
      document.getElementById("rssOutput").innerHTML=this.responseText;
    }
  }
  xmlhttp.open("GET","https://www.w3schools.com/php/getrss.php?q="+str,true);
  xmlhttp.send();
}
</script>
</head>
<body>

<form>
<select onchange="showRSS(this.value)">
<option value="">Select an RSS-feed:</option>
<option value="Google">Google News</option>
<option value="NBC">NBC News</option>
</select>
</form>
<br>
<div id="rssOutput">RSS-feed will be listed here...</div>
</body>
</html>
```

3.3 적용 예

3.3.3 Ajax 애플리케이션(PHP)

② 설문조사
- 다시 로드 하지 않고, 결과가 표시되는 폴링을 보여줌 - getVote() 함수

26.ajax_appgetvote.html

```
<!DOCTYPE html>
<html>
<head>
<script>
function getVote(int) {
  if (window.XMLHttpRequest) {
    // code for IE7+, Firefox, Chrome, Opera, Safari
    xmlhttp=new XMLHttpRequest();
  } else {  // code for IE6, IE5
    xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
  xmlhttp.onreadystatechange=function() {
    if (this.readyState==4 && this.status==200) {
      document.getElementById("poll").innerHTML=this.responseText;
    }
  }
  xmlhttp.open("GET","https://www.w3schools.com/php/poll_vote.php?vote="+int,true);
  xmlhttp.send();
}
</script>
</head>
<body>

<div id="poll">
<h3>Do you like PHP and AJAX so far?</h3>
<form>
Yes:
<input type="radio" name="vote" value="0" onclick="getVote(this.value)">
<br>No:
<input type="radio" name="vote" value="1" onclick="getVote(this.value)">
</form>
</div>

</body>
</html>
```

3.3 적용 예

3.3.4 Ajax 애플리케이션(jQuery)

① Load() 메서드
- 서버에서 데이터를 로드하고 반환된 데이터를 선택한 요소에 저장

27.ajax_appload.html

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("#div1").load("https://www.w3schools.com/jquery/demo_test.txt");
  });
});
</script>
</head>
<body>

<div id="div1"><h2>Let jQuery AJAX Change This Text</h2></div>

<button>Get External Content</button>

</body>
</html>
```

3.3 적용 예

3.3.4 Ajax 애플리케이션(jQuery)

① Load() 메서드
- 서버에서 데이터를 로드하고 반환된 데이터를 선택한 요소에 저장 - 특정 요소만 로드

28.ajax_appload2.html

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("#div1").load("https://www.w3schools.com/jquery/demo_test.txt #p1");
  });
});
</script>
</head>
<body>

<div id="div1"><h2>Let jQuery AJAX Change This Text</h2></div>

<button>Get External Content</button>

</body>
</html>
```

3.3 적용 예

3.3.4 Ajax 애플리케이션(jQuery)

① Load() 메서드
- 서버에서 데이터를 로드하고 반환된 데이터를 선택한 요소에 저장 - 메서드 완료 후 경고 상자 표시

29.ajax_appload3.html

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("#div1").load("https://www.w3schools.com/jquery/demo_test.txt", function(responseTxt, statusTxt, xhr){
      if(statusTxt == "success")
        alert("External content loaded successfully!");
      if(statusTxt == "error")
        alert("Error: " + xhr.status + ": " + xhr.statusText);
    });
  });
});
</script>
</head>
<body>

<div id="div1"><h2>Let jQuery AJAX Change This Text</h2></div>

<button>Get External Content</button>

</body>
</html>
```

3.3 적용 예

3.3.4 Ajax 애플리케이션(jQuery)

② get() 메서드
- 서버에 데이터를 요청

30.ajax_appget.html

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $.get("https://www.w3schools.com/jquery/demo_test.asp", function(data, status){
      alert("Data: " + data + "₩nStatus: " + status);
    });
  });
});
</script>
</head>
<body>

<button>Send an HTTP GET request to a page and get the result back</button>

</body>
</html>
```

3.3 적용 예

3.3.4 Ajax 애플리케이션(jQuery)

③ post() 메서드
- 서버의 데이터를 요청

31.ajax_apppost.html

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $.post("https://www.w3schools.com/jquery/demo_test_post.asp",
    {
      name: "Donald Duck",
      city: "Duckburg"
    },
    function(data,status){
      alert("Data: " + data + "₩nStatus: " + status);
    });
  });
});
</script>
</head>
<body>

<button>Send an HTTP POST request to a page and get the result back</button>

</body>
</html>
```

Front-end&RESTful
Chapter 03-1.
Ajax