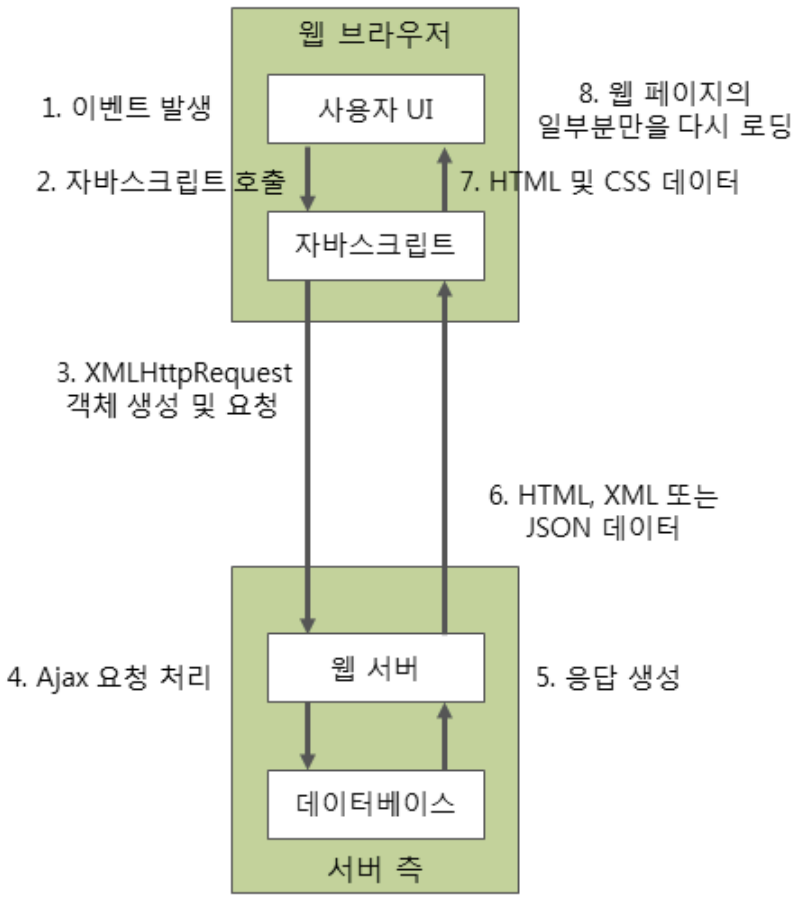
# AJAX (Asynchronous Javascript And XML)

Ajax는 웹 페이지 전체를 **다시 로딩하지 않고도, 웹 페이지의 일부분만을 갱신**할 수 있게 해준다. Ajax를 이용하면 백그라운드 영역에서 서버와 통신하여, 그 결과를 웹 페이지의 일부분에만 표시할 수 있다.

ajax를 통한 웹 브라우저와 웹 서버 간의 통신 절차는 다음과 같다.

< Ajax를 이용한 웹 응용 프로그램의 동작 원리 >




사용자에 의한 요청 이벤트가 발생

요청 이벤트가 발생하면 이벤트 핸들러에 의해 자바스크립트가 호출

자바스크립트는 XMLHttpRequest 객체를 사용하여 서버로 요청을 보냄

서버는 전달받은 XMLHttpRequest 객체를 가지고 요청을 처리

서버는 처리한 결과를 HTML, XML 또는 JSON 형태의 응답 데이터를 생성 웹 브라우저에 전달

이때 전달되는 응답은 새로운 페이지를 전부 보내는 것이 아니라 필요한 데이터만을 전달한다.

서버로부터 전달받은 데이터를 가지고 웹 페이지의 일부분만을 갱신하는 자바스크립트를 호출. 결과적으로 웹 페이지의 일부분만이 다시 로딩되어 표시된다.

# XMLHttpRequest 객체

XMLHttpRequest 객체는 **웹 브라우저가 서버와 데이터를 교환**할 때 사용된다. 웹 브라우저가 백그라운드에서 계속해서 서버와 통신할 수 있는 것은 바로 이 객체를 사용하기 때문이다.

JAVASCRIPT

```
// 1. XMLHttpRequest객체 생성
var httpRequest = new XMLHttpRequest();
// 2. onreadystatechange 등록
httpRequest.onreadystatechange = function() {
    // XMLXttpRequest 객체의 현재 상태와 서버 상의 문서 존재 유무를 확인
    if (httpRequest.readyState == XMLHttpRequest.DONE && httpRequest.status == 200 ) {
        console.log(httpRequest.responseText); // 서버에 요청하여 응답으로 받은 데이터를 문자열로 반환
    }
};
// 3. GET 방식으로 요청을 보내면서 데이터를 동시에 전달함
```

```
httpRequest.open("GET", "서버URL", true);
httpRequest.send();
```

## XMLHttpRequest 요청 보내기

XMLHttpRequest 기본 사용법 설명은 다음과 같다.

### 1. XMLHttpRequest 객체 생성

JAVASCRIPT

```
var httpRequest = new XMLHttpRequest();
```

### 2. onreadystatechange 등록

JAVASCRIPT

```
// 2. onreadystatechange 등록
httpRequest.onreadystatechange = function() {
};
```

### 3. 서버로 보낼 Ajax 요청의 형식을 설정

`httpRequest.open(전달방식, URL주소, 동기여부)` 메소드를 등록한다.

- 전달 방식은 요청을 전달할 방식으로 GET 방식과 POST 방식 중 하나를 선택할 수 있다.
- URL 주소는 요청을 처리할 서버의 파일 주소를 전달한다.
- 동기 여부는 요청을 동기식으로 전달할지 비동기식으로 전달할지를 전달한다. (true : 비동기 / false : 동기)

JAVASCRIPT

```
httpRequest.open("GET", "서버URL", true);
```

### 4. 작성된 Ajax 요청을 서버로 전달

`httpRequest.send()` 메서드를 통해 서버로 객체를 전달한다. 이때 `send()` 메서드 매개변수를 쓰냐 안쓰냐에 따라 GET / POST 방식으로 다르게 보내게 된다.

- `send()` - GET 방식
- `send(문자열)` - POST 방식

JAVASCRIPT

```
httpRequest.send();
```

## POST 방식 서버 요청

Aiax에서는 서버에 POST 방식의 요청을 보내기 위해서 다음과 같이 작성 한다. 이때 서버로 전송하고자 하는 데이터는 HTTP 헤더에 포함되어 전송된다. 따라서 `setRequestHeader()` 메소드를 이용하여 먼저 헤더를 작성한 후에, `send()` 메소드로 데이터를 전송하면 된다.

JAVASCRIPT

```
var httpRequest = new XMLHttpRequest(); // 객체 생성
httpRequest.onreadystatechange = function() {
    if (httpRequest.readyState == XMLHttpRequest.DONE && httpRequest.status == 200 ) {
        console.log(httpRequest.responseText);
    }
};
// POST 방식의 요청은 데이터를 Http 헤더에 포함시켜 전송함.
httpRequest.open("POST", "/examples/media/request_ajax.php", true);
```

```
httpRequest.setRequestHeader("Content-Type", "application/x-www-form-urlencoded"); // 보낼 데이터 헤더를 지정
httpRequest.send("city=Seoul&zipcode=06141"); // 보낼 데이터 지정
```

---

# XMLHttpRequest 응답 받기

## readyState 프로퍼티

Ajax에서 서버로부터의 응답을 확인하기 위해 사용하는 XMLHttpRequest 객체의 readyState 프로퍼티는 XMLHttpRequest 객체의 현재 상태를 나타낸다. 이 프로퍼티의 값은 객체의 현재 상태에 따라 다음과 같은 주기로 순서대로 변화한다.

UNSENT (숫자 0) : XMLHttpRequest 객체가 생성됨.

OPENED (숫자 1) : open() 메소드가 성공적으로 실행됨.

HEADERS_RECEIVED (숫자 2) : 모든 요청에 대한 응답이 도착함.

LOADING (숫자 3) : 요청한 데이터를 처리 중임.

DONE (숫자 4) : 요청한 데이터의 처리가 완료되어 응답할 준비가 완료됨.

JAVASCRIPT

```
var httpRequest = new XMLHttpRequest();
var currentState = "";
httpRequest.onreadystatechange = function () {
    switch (httpRequest.readyState) {
        case XMLHttpRequest.UNSET:
            currentState += "XMLHttpRequest 객체의 현재 상태는 UNSET 입니다.<br>";
            break;
        case XMLHttpRequest.OPENED:
            currentState += "XMLHttpRequest 객체의 현재 상태는 OPENED 입니다.<br>";
            break;
        case XMLHttpRequest.HEADERS_RECEIVED:
            currentState += "XMLHttpRequest 객체의 현재 상태는 HEADERS_RECEIVED 입니다.<br>";
            break;
        case XMLHttpRequest.LOADING:
            currentState += "XMLHttpRequest 객체의 현재 상태는 LOADING 입니다.<br>";
            break;
        case XMLHttpRequest.DONE:
            currentState += "XMLHttpRequest 객체의 현재 상태는 DONE 입니다.<br>";
            break;
    }
    console.log(currentState);
    if (httpRequest.readyState == XMLHttpRequest.DONE && httpRequest.status == 200) {
        console.log(httpRequest.responseText);
    }
};
httpRequest.open("GET", "/examples/media/ajax_intro_data.txt", true);
httpRequest.send();
```

## status 프로퍼티

status 프로퍼티는 서버의 문서 상태를 나타낸다.

- 200 : 서버에 문서가 존재함.
- 404 : 서버에 문서가 존재하지 않음.

## 응답 데이터 프로퍼티

서버로부터 응답 받은 데이터는 아래 프로퍼티에 들어 있다. 이를 이용하여 클라이언트에서 적절히 처리하면 된다.

- responseText 프로퍼티 : 서버에 요청하여 응답으로 받은 데이터를 문자열로 반환
- responseXML 프로퍼티 : 서버에 요청하여 응답으로 받은 데이터를 XML DOM 객체로 반환

```
httpRequest.onreadystatechange = function() {
    // XMLXttpRequest 객체의 현재 상태와 서버 상의 문서 존재 유무를 확인
    if (httpRequest.readyState == XMLHttpRequest.DONE && httpRequest.status == 200 ) {
        console.log(httpRequest.responseText); // 서버에 요청하여 응답으로 받은 데이터를 문자열로 반환
    }
};
```