

# Ajax

## ajax란?

서버로부터 데이터를 가져와 전체 페이지를 새로 고치지 않고 일부만 로드할 수 있게 하는 기법으로 비동기식 요청을 보내는데 필요한 기술을 말함

👉 AJAX (Asynchronous JavaScript AND XML)

## 장단점

장점 : 비동기식 방식으로 웹서버의 응답을 기다리지 않고 데이터를 빠르게 처리하는 개발기법, 페이지 리로딩 없이 처리됨

예) 실시간 검색어, 검색어자동 완성

단점 : 한 페이지에 지속적으로 사용시 리소스가 계속 쌓여 페이지가 느려짐, 스크립트로 되어 있어 에러 발생시 디버깅이 어려움

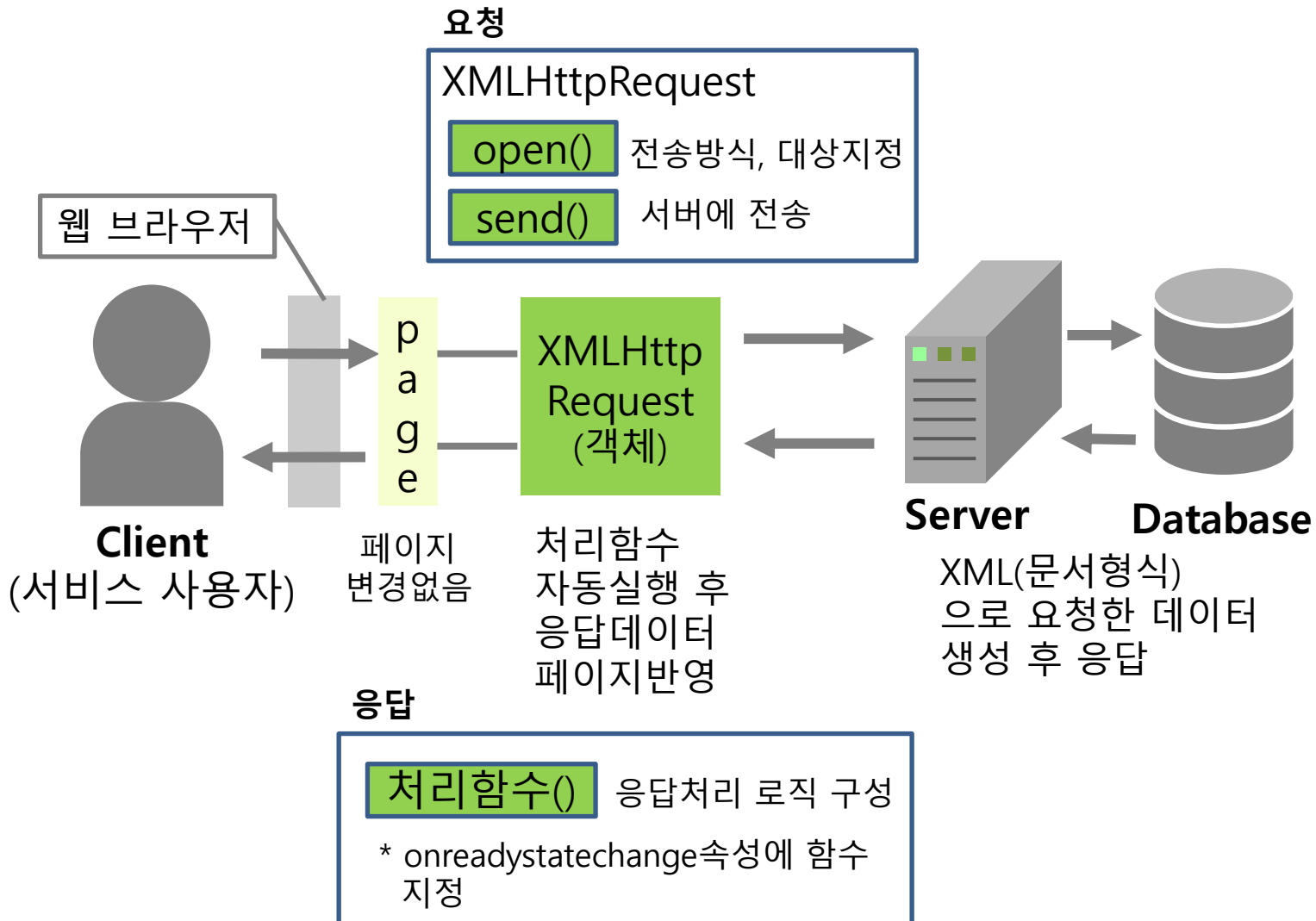
## 비동기식 처리모델

페이지가 로드 되는 동안 브라우저는 먼저 서버에 데이터를 요청 script문 실행한 후 페이지의 나머지를 계속 로드하고 페이지와 사용자의 상호작용을 처리하며 브라우저는 요청한 데이터를 기다리지 않는다. 그리고 요청한 데이터가 도착을 하면 그때 이벤트가 발생하면서 지정된 함수가 호출되어 실행되는 방식. (넌블로킹 모델)

## 동기식 처리모델

페이지가 로드 되는 동안 브라우저는 script문을 실행되면 그 실행이 종료될때 까지 나머지 페이지를 로드하지 않고 기다리고 있다가 그 script문이 처리가 종료가 되면 페이지의 나머지 부분을 로드하는 방식

# Javascript(Ajax)



1. script문에 요청을 위한 XMLHttpRequest객체 생성
2. 서버의 응답을 처리할 함수 생성 및 지정
  - ☞ onreadystatechange에 함수지정
3. open메소드로 요청할 방법 및 요청할 대상(Server)선정
  - ☞ 요청메소드, 요청주소, 동기/비동기 설정, 아이디, 패스워드 설정
3. send메소드로 대상(Server)에 전송
  - ☞ post일때 파라미터값 설정/ get일때는 매개변수 없음
3. 응답상태에 따라 상태확인
  - ☞ readyState(데이터응답) / status(처리결과) 값을 이용
6. 응답완료 reponseText / reponseXML이용 응답처리

## XMLHttpRequest란?

비동기식으로 서버에 요청(Request)을 보내기 위한 객체로 요청 및 응답을 처리함

## 속성

속성명	내 용
onreadystatechange	readyState속성이 변경될때 호출되는 매소드를 저장 하는 변수
readyState	객체의 상태를 저장하는 변수
responseText	응답(response)결과를 문자열로 저장하는 변수
responseXML	응답(response)결과를 XML data로 저장하는 변수
status	전송/응답 결과를 저장하는 변수(코드값)
statusText	전송/응답 결과를 저장하는 변수(문자열)

## 참고

### readyState속성 값

속성값	내 용
0	요청이 시작되지 않은 상태 / open메서드가 호출되지 않은 상태
1 (loading)	서버와 접속된 상태 / send메서드가 호출되지 않은 상태
2 (loaded)	send메서드 호출되고 headers는 도착하지 않은상태
3 (interactive)	일부 데이터를 받은 상태
4 (completed)	요청을 완료하고 응답하는 상태

### status속성 값

200 (OK) : 요청성공

404 (Not Found) : 페이지 없음

500 (Internal Server Error) : 서버오류발생 등

추가 참조

[https://www.w3schools.com/tags/ref\\_httpmessages.asp](https://www.w3schools.com/tags/ref_httpmessages.asp)



## 매소드

매소드명	내 용
abort()	요청을 취소하는 매소드
getAllResponseHeaders()	응답해더 정보를 가져오는 매소드
getResponseHeader("해더명")	특정 해더정보만 가져오는 매소드
open(method,url,async,user,pw)	요청정보를 설정하는 매소드
send()	서버로 요청을 보내는 매소드(get방식)
send(String)	서버로 요청을 보내는 매소드(post방식)
setRequestHeader()	해더에 이름/값 방식으로 정보를 추가하는 매소드

## 참고- Header속성값

<b>text</b>	<b>server(servlet)</b>
ALL date	ALL date
last-modified	content-length
server	server
etag	
content-length	
content-type	

## 1. 객체생성

### 생성방법

IE7이상, safari, firefox, opera, chrome

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

IE5이나 6 이하의 버전

```
var httpRequest=new ActiveXObject(Microsoft.XMLHTTP);
```

## 예시

```
<script>
var httpRequest;
function getHttpRequest(){
    //브라우저가 IE일 경우
    if(window.ActiveXObject){
        try{ //신버전 9버전 이후
            return new ActiveXObject("Msxml2.XMLHTTP");
        } catch(e) {
            try{ //그 이전 버전
                return new ActiveXObject("Microsoft.XMLHTTP");
            } catch (ex) {
                return null
            }
        }
    }
    //그 외의 브라우저일 경우
    } else if(window.XMLHttpRequest) return new XMLHttpRequest();
    else return null;

}
</script>
```

## 2. 응답처리 함수 설정

### 객체생성 후 속성값에 함수를 설정

```
var httpRequest= getHttpRequest();  
  
httpRequest.onreadystatechange = 실행할 함수명;  
또는  
httpRequest.onreadystatechange = function(){  
    // 처리 로직  
}
```

## 예시

```
<script>
var httpRequest= getHttpRequest();
httpRequest.onreadystatechange=test;
function test()
{
    if(httpRequest.readyState===4) //전송상태확인
    {
        if(httpRequest.status===200) //완료상태확인
        {
            document.getElementById("ex").innerHTML
            =httpRequest.responseText; //응답값처리
        }
    }
}
</script>
```

응답상태를 기준으로 처리 로직을 구성

## 3. 요청대상설정 / 요청처리

### 객체생성 후 매서드이용 요청대상 설정/요청처리

```
var httpRequest= getHttpRequest();
```

```
httpRequest.open(전송방법, 요청페이지 또는 파일,  
비동기식/동기식 설정, id값, password값)
```

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

또는

```
httpRequest.send("param값");
```

☞ param설정 : 명칭=값 형식으로 표현하고 여러 값일 경우 &를 사용

☞ 전송방법 및 페이지 외 생략 가능 / 비동기 설정이 default값

## 예시

```
<script>
```

```
var httpRequest= getHttpRequest();
```

```
httpRequest.onreadystatechange=test;
```

```
httpRequest.open("get","test?name=hongGD&age=15",true);
```

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

또는

```
httpRequest.open("post","test",true);
```

```
httpRequest.send("name=hongGD&age=15");
```

```
</script>
```



## 4. 응답처리

### 객체생성 후 속성값으로 응답처리(text)

```
var httpRequest= getHttpRequest();
```

```
var value=httpRequest.responseText;
```

또는

```
document.getElementById("test").innerHTML  
=httpRequest.responseText;
```

\* 문자 인코딩에 주의(servlet/jsp에서 응답시 코딩처리)

## 예시

```
<script>
  var httpRequest= getHttpRequest();
  httpRequest.onreadystatechange=test;
  function test()
  {
    if(httpRequest.readyState===4)
    {
      if(httpRequest.status===200)
      {
        document.getElementById("ex").innerHTML
          =httpRequest.responseText; //특정 태그에
                                     응답 받은 데이터 처리
      }
    }
  }
</script>
```

## 4. 응답처리

### 객체생성 후 속성값으로 응답처리(xml)

```
var httpRequest= getHttpRequest();

var xmlDoc=httpRequest.responseXML;
var xml=xmlDoc.getElementsByTagName("태그명칭");
var text="";
for(i=0;i<xml.length;i++)
{
    text+=xml[i].childNodes[0].nodeValue+"<br>";//노드값불러오기
}
document.getElementById("명칭").innerHTML=text;
```

\* 문자 인코딩에 주의(servlet / jsp에서 응답시 코딩처리)

## 예시

```
var httpRequest= getHttpRequest();
httpRequest.onreadystatechange=test;
function test()
{
    if(httpRequest.readyState===4)
    {
        if(httpRequest.status===200)
        {
            var xmlDoc=httpRequest.responseXML;
            var xml=xmlDoc.getElementsByTagName("태그명칭");
            var text="";
            for(i=0;i<xml.length;i++)
            { text+=xml[i].childNodes[0].nodeValue+"<br>"; }
            document.getElementById("test").innerHTML=text;
        }
    }
}
```

# jQuery(Ajax)

## 파라미터

명칭	내 용
url	요청할 페이지 주소 설정 / servlet, 문서 둘다 가능
data	요청시 보낼 데이터 설정 예) { 명칭:값, 명칭:값 }
function(data,status)	응답처리할 함수 지정, data : 응답값, status:처리상태

## 사용법

```
<script>
```

```
  $("선택자").load(url,[data, function]);
```

```
</script>
```

\* url은 필수 입력, 나머지 매개변수 생략가능

url제외한 나머지를 생략할 경우 servlet이나 document에서 응답한 값을  
선택자로 선택된 태그의 텍스트 노드에 출력(get방식) / 페이지 변경  
data입력시(post방식)

# \$('.선택자').load()

## 예시

```
<script>
  function test(url)
  {
    $('.pp').load(url); //get방식으로 전달하고 class pp에 응답한
                        값을 출력 쿼리 스트링으로 값 전달
    $('.pp').load(url,{id:'홍길동',pw:0000}); //post방식전달
    $('.pp').load(url,{id:'홍길동',pw:0000}, responseFunc());
  }
  function responseFunc(data, status)
  {
    var test=data+status; $('.pp').text(test);
  }
</script>

<button onclick="test('ajaxtest?id=??&pw=00')"> 데이터전송 </button>
<div class='pp'> </div>
<p class='pp'> </p>
<span class='pp'> </span>
```

## 파라미터

명칭	내 용
url	요청할 페이지 주소 설정 / servlet, 문서 둘다 가능
data *post에 사용	요청시 보낼 데이터 설정 예) { 명칭:값, 명칭:값 }
function(data,status)	응답처리할 함수 지정, data : 응답값, status:처리상태

## 사용법

```
<script>
```

```
$.get(url, function); / $.post(url,data,function);
```

```
</script>
```

\* 응답처리에 관한 페이지 변경은 function에서 수행

\*예시는 load()메소드와 유사하여 생략



## 처리절차

1. url 속성을 통해 전송할 URL 주소 선언
2. data 속성을 통해 전달할 데이터 설정
3. 성공, 실패 시 처리할 로직을 함수로 선언
4. 반드시 처리할 로직을 선언



## \$.ajax() 주요 속성 정보

속성 명	설 명
url	데이터를 전송할 URL의 주소 설정
data	서버에 전송할 데이터를 key : value 형식으로 설정
dataType	서버가 리턴하는 데이터의 타입 설정(text, xml, html)
type	서버로 전송하는 형식 지정 (GET, POST)
success	통신에 성공했을 때 처리할 로직을 함수로 작성
error	통신에 실패했을 때 처리할 로직을 함수로 작성
complete	통신 시 반드시 실행할 로직을 함수로 작성

# \$.ajax()

## 예시

```
//서블릿을 호출하여 String 값을 전달
$.ajax({
    url : "test",      // 1. 전달할 servlet url mapping
    data : {name : "hongGD"}, // 2. 전달할 데이터
    type : "get",      // 전달할 방식 지정
    success : function(data){ // 3-1. 성공 시 처리할 절차
        console.log("성공 "+data);
    },
    error : function(request,status,error) {
        // 3-2. 에러 발생 시 처리할 절차
        alert("code:"+request.status+"\n"+
            "message:"+request.responseText+"\n"+"error:"+error);
    },
    complete : function(data){ // 4. 반드시 처리할 절차
        console.log("무조건 호출 되는 로직");
    }
});
```