

# **AJAX**

## **(Asynchronous JavaScript And XML)**

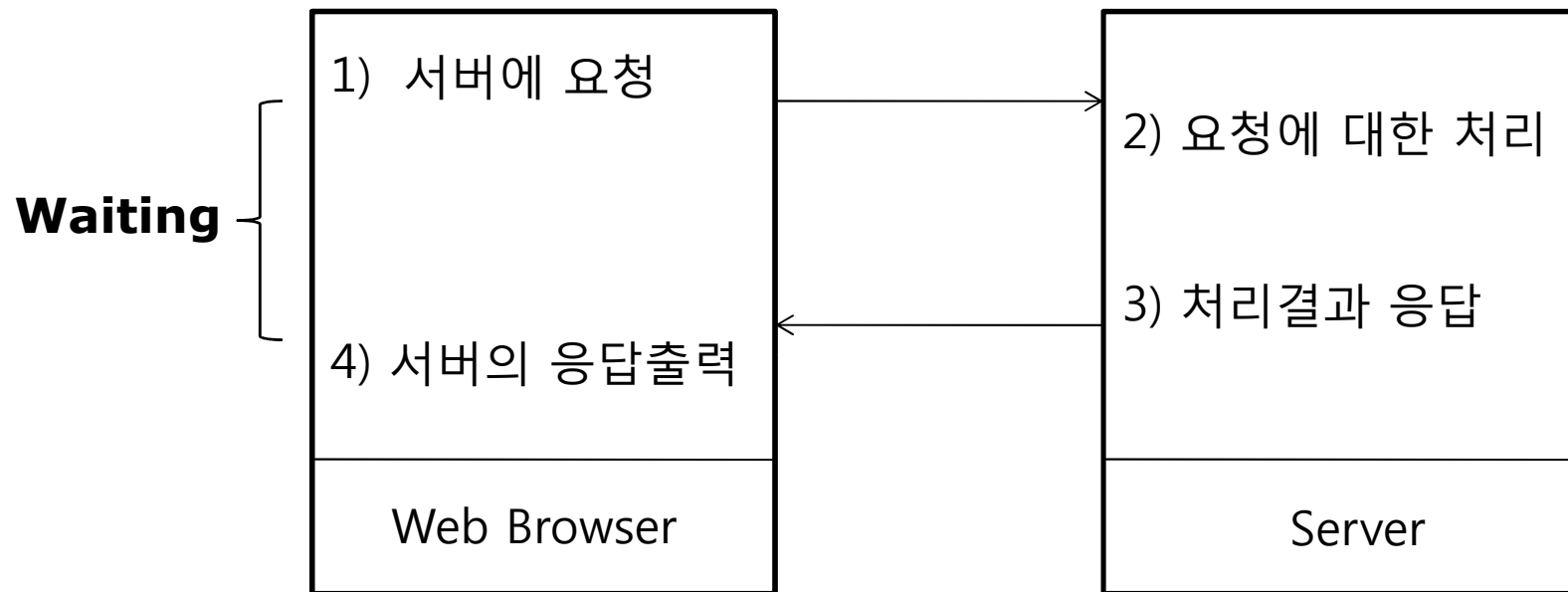
KOSTA

# AJAX 개요

- JavaScript와 XML을 이용하여 Web Client와 Server간의 비동기적 처리를 지원하는 프로그래밍 기법
  - Google의 Google Map이 AJAX를 이용하여 구축된 이후 각광 받음.
  - Web 2.0의 Rich Client 기술로 인식되어 Web의 많은 분야에서 적용되는 추세
- AJAX는 그 자체가 하나의 기술이라기 보다는 패턴을 의미한다.
  - 기존 Web Browser가 제공하는 기능을 활용하는 프로그래밍 패턴.
  - 기존 기술인 JavaScript와 XML등을 이용한 프로그래밍 패턴

# AJAX 이전의 WEB

- Client와 Server – 동기적 전송방식



한계 : 기존 Desktop Application과 같은 실행 능력을 보이지 못함

- 대안 기술

- ActiveX
- Applet
- ActionScript

} Web Browser 지원 기술이 아님 – 플러그인 형식

# AJAX 프로그래밍 패턴

1. XMLHttpRequest 객체 구하기
2. 서버에 Request 전달
3. 서버에서 응답한 Response 데이터 처리하기

# XMLHttpRequest 객체

- AJAX 의 비동기적 처리의 핵심 JavaScript 객체
- 서버와 클라이언트(Web Browser) 사이의 비동기적 데이터 송수신을 담당하는 객체
- 모든 Web Browser가 지원하나 사용 방법은 차이가 있다.
  - IE 6버전까지는 지원하는 MS 자체 객체로 지원

## XMLHttpRequest 객체 생성

```
<script language="javascript">
    var xhr;
    function createXMLHttpRequest(){
        if(window.ActiveXObject){
            xhr = new ActiveXObject("Microsoft.XMLHTTP");
        }else{
            xhr = new XMLHttpRequest();
        }
    }
</script>
```

# 서버에 Request 전달

- XMLHttpRequest 객체의 함수를 이용
  - open() : 요청 정보 설정. Http Method, URL
  - send() : 요청 전송
  - call back(event Handler) 함수 등록 : 서버와의 데이터 송수신 상태에 따른 처리를 담당할 call back 함수 등록
- open(http method, url, 동기화 방식);
  - http method : 전송 방식 – GET, POST
  - URL : 요청정보를 전송할 URL
  - 동기화 방식 : true – 비동기적 전송 처리  
false – 동기적 전송 처리  
xhr.open("GET", "process.jsp", true);
- send(전송데이터)
  - 전송데이터 : 요청 시 서버에 보낼 querystring
    - GET 방식의 경우 URL에 붙여 보내므로 null 값으로 처리
    - POST 방식은 데이터를 key-value로 저장하여 전달 할 수 있다.

# 서버에 Request 전달

- GET 방식 데이터 전달
  - open() 의 url 뒤에 query string으로 전달

```
var url = "send_data.jsp?name=홍길동"  
xhr.open("GET", url, true);  
xhr.send(null);
```

- POST 방식 데이터 전달
  - 요청 전에 헤더의 Content-Type을 지정해 준다.
  - send() 호출 시 argument로 데이터를 전송한다.

```
var url = "send_data.jsp"  
xhr.open("POST", url, true);  
xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded;charset=euc-kr');  
xhr.send("name=홍길동");
```

# 서버가 보낸 Response 데이터 처리

- Call Back 함수
  - XMLHttpRequest는 요청과 관련된 상태값을 저장하며 그 값이 변경시 마다 event를 발생시킨다. 그 event를 처리할 handler 함수.
  - 주로 서버가 보낸 응답을 처리
  - 개발자는 call back 함수를 정의 하고 XMLHttpRequest에 등록
- Call Back 함수 등록
  - `xhr.onreadystatechange` = call back 함수
  - `xhr.onreadystatechange=function(){ //코드 }`
  - `onreadystatechange` : 요청에 대한 상태가 변화할 때 마다 발생하는 event를 처리하기 위한 handler
- Call Back 함수 구현
  - 서버와의 연결 상태 변화 체크
    - `readyState == 4`
  - 응답 status 체크
    - `status == 200`
- 서버의 응답 데이터 읽기
  - TEXT 응답 – XMLHttpRequest객체.**responseText**
  - xml 형식 응답 – XMLHttpRequest객체.**responseXML**



# 서버가 보낸 Response 데이터 처리

readystatechange 값 – XMLHttpRequest객체의 상태 값

값	상태
0	unInitialized– XMLHttpRequest 객체만 생성되고 초기화 되지 않음
1	Loading – open() 호출 후 아직 서버에 요청하지 않은 상태
2	Loaded – send() 실행 된 상태. 아직 응답은 받지 않은 상태
3	Interactive – 응답 데이터를 받는 중인 상태
4	Completed – 응답 데이터를 받은 상태

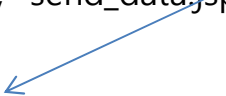
주요 응답 status – Http 응답 상태 코드

값	상태
200	OK : 성공
403	실행 거부
404	요청 페이지 없음
405	요청 방식 실행 불가
500	서버 CGI 실행 중 오류
<a href="http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html">http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html</a>	

# 서버가 보낸 Response 데이터 처리

callback 함수 예제

```
<script>
  var xhr;
  function createXMLHttpRequest(){
    if(window.ActiveXObject){
      xhr = new ActiveXObject("Microsoft.XMLHTTP");
    }else{
      xhr = new XMLHttpRequest();
    }
  }
  function openRequest(){
    createXMLHttpRequest();
    xhr.onreadystatechange = getdata;
    xhr.open("GET", "send_data.jsp?name=hong", true);
    xhr.send(null);
  }
  function getdata(){
    if(xhr.readyState==4){
      if(xhr.status==200){
        var txt = xhr.responseText;
        alert(txt);
      }
    }
  }
}
</script>
```



## AJAX와 서버 간의 데이터 송수신 기법

- XML을 이용한 데이터 송수신
- JSON을 이용한 데이터 송수신
- 기타 AJAX 기반 Framework을 이용한 데이터 송수신

## JSON을 이용한 데이터 송수신

- XML의 단점 : 속도, 데이터 크기
- JSON – JavaScript Object Notation
  - 경량의 데이터 교환 형식
  - 데이터 표현이 단순하고 직관적이다.
  - Java Script에서 eval() 를 통해 parsing 할 수 있다.
    - 라이브러리 필요 없음
    - 크로스 브라우저 호환 문제 없음
    - 악성코드 실행의 위험이 있음
  - json2.js 를 이용해 parsing
    - JSON 객체 형태의 문자열만 변경처리
    - 예) JSON.parse(txt)

# JSON을 이용한 데이터 송수신

- 기본문법

- 표현하고자 하는 데이터는 { } 로 감싼다.
- 데이터는 name:value 형식으로 표현
- 각각의 값들은 ' , ' 으로 구분한다.
- name : " " 로 감싸거나 그냥 써도 된다. "name":"홍길동" 또는 name:"홍길동"
- 값 : string, number, 배열, true, false, null
  - string은 " " 로 감싼다.
  - 배열은 [ ] 로 값들을 감싼다.

예

```
• { id:"abc", password:"1111", name:"홍길동", address:"서울", age:20}
• { names : ["이순신", "강감찬", "홍길동"]}
• { "member": [
    {id:"abc", password:"1111", name="홍길동", address:"서울", age:20},
    {id:"def", password:"2222", name:"이순신", address:"부산", age:25},
    {id:"ghi", password:"3333", name:"강감찬", address:"광주", age:22}
  ]}
• {
  member:[ { name:"홍길동", age:20, address:"서울" }, { name:"이순신", age:23, address:"인천" },
  admin:[ { name:"이철수", age:25, address:"마산" }, { name:"박영희", age:30, address:"대구" }, ]
}
```

# JSON을 이용한 데이터 송수신

- JavaScript에서 JSON 데이터 접근
  - 응답 문자열을 json object로 변경

```
var txt = xhr.responseText;  
var jsonData = eval('(' + txt + ')'); //또는  
var jsonData = JSON.parse(txt);
```

- value 는 .을 통해 접근 -key.subkey

```
{name:"홍길동", age:20}
```

```
var name = jsonData.name  
var age = jsonData.age
```

- 배열은 [index] 로 접근. index는 0부터 시작

```
{names:["홍길동", "이순신", "강감찬"]}
```

```
var name1 = jsonData.names[0];  
var name2 = jsonData.names[1];
```

# JSON을 이용한 데이터 송수신 – org.json

- JAVA에서 JSON 사용
  - 다양한 LIB 존재 : [www.json.org](http://www.json.org) 참조
- org.json : [www.json.org/java](http://www.json.org/java)
- JSONObject – JSON 형식의 String 값을 만들기 위한 객체
  - 내부적으로 HashMap 이용 : key-value 쌍으로 관리
  - 생성자
    - JSONObject(), JSONObject(Map map), JSONObject(Object dto)
  - 메소드
    - put(String key, XXXX value) : XXXX : primitive, Collection, Map
    - get(String key) : Object
    - getXXXX(String key) : XXXX
    - toString()
- JSONArray – JSON 배열 형태의 String 값을 만들기 위한 객체
  - 내부적으로 ArrayList 이용
  - 생성자
    - JSONArray(), JSONArray(Collection list), JSONArray(Object array)
  - 메소드
    - put(XXXX value) : XXXX : primitive, Collection, Map, DTO객체
    - get(int idx) : Object
    - getXXXX(int idx) : XXXX
    - toString()

**jQuery**



# jQuery 개요

- Javascript Library
  - 오픈소스 Javascript Library
  - 존 레시그, 2006년 발표
  - <http://jquery.com>
- 다운로드
  - [http://docs.jquery.com/Downloading\\_jQuery](http://docs.jquery.com/Downloading_jQuery)
  - jquery-X.X.X-min.js 다운
  - js 파일 다운 로드 후 사용하고자 하는 페이지에서 script 태그의 src 속성으로 등록해 사용
- 참고 사이트
  - <http://docs.jquery.com/>
  - <http://visualjquery.com/>

# jQuery?

- Rich Client UI를 개발하는 다양한 기능을 지원
  - HTML 문서 탐색 , 이벤트처리 , 애니메이션 , ajax 상호작용
- HTML에서 구조와 동작을 분리하여 작성
  - 구조 : HTML, 동작 : Javascript
  - 강력한 selector지원
- 크로스 브라우저 지원
- 메소드 체인
- 플러그인 지원

# 기본 문법

- jQuery 라이브러리 등록
  - <script>로 다운받은 파일 등록
  - URL로 등록
    - <script src="http://code.jquery.com/jquery-latest.js">
- Selector함수
  - jQuery 함수를 적용할 요소를 찾는 함수
  - jQuery("selector") 또는 \$("selector")
- Selector함수.jQuery커맨드
  - jQuery커맨드는 jQuery가 지원하는 메소드임
  - jQuery Chain
- 예
  - jQuery("#layer").html("안녕");
  - \$("p").css("color", "red").css("background", "blue");

# Selector를 이용해 HTML구조 접근

- 구조 태그 (HTML 태그) 와 Javascript 분리
  - 동작을 HTML에 적용하기 위해 HTML 태그에 접근할 필요
  - CSS기본 selector나 jQuery 정의 selector이용해 접근
- CSS selector 예

```
body{  
background-color:#D024FE;  
}  
div#message_layer{  
font-color:red;  
}
```

## jquery 가 지원하는 기본 CSS 셀렉터

셀렉터	설명
*	모든 엘리먼트와 일치
E	태그명이 E인 모든 엘리먼트
E F	E의 자손이면서 태그명이 F인 모든 엘리먼트와 일치
E>F	E의 바로 아래 자식이면서 태그명이 F인 모든 엘리먼트와 일치
E+F	E의 형제 엘리먼트로 바로 다음에 나오는 엘리먼트 F와 일치
E~F	E의 형제 엘리먼트로 다음에 나오는 모든 엘리먼트 F와 일치
E:has(F)	태그명이 F인 자손을 하나 이상 가지는 태그명이 E인 모든 엘리먼트와 일치
E.C	클래스명 C를 가지는 모든 엘리먼트 E와 일치. E의 생략은 *.C와 동일함.
E#I	아이디가 I인 엘리먼트 E와 일치. E의 생략은 *#I와 동일함.
E[A]	어트리뷰트 A를 가지는 모든 엘리먼트 E와 일치
E[A=V]	값이 V인 어트리뷰트 A를 가지는 모든 엘리먼트 E와 일치
E[A^=V]	값이 V로 시작하는 어트리뷰트 A를 가지는 모든 엘리먼트 E와 일치
E[A\$=V]	값이 V로 끝나는 어트리뷰트 A를 가지는 모든 엘리먼트 E와 일치
E[A*=V]	값에 V를 포함하는 어트리뷰트 A를 가지는 모든 엘리먼트 E와 일치

jquery 가 지원하는 고급 위치 기반 셀렉터.

셀렉터	설명
:first	페이지에서 처음으로 일치하는 엘리먼트. li a:first는 리스트 아이템의 첫 번째 링크를 반환한다.
:last	페이지에서 마지막으로 일치하는 엘리먼트. li a:last는 리스트 아이템의 마지막 링크를 반환한다.
:first-child	첫 번째 자식 엘리먼트. li:first-child는 각 리스트의 첫 번째 아이템을 반환한다.
:last-child	마지막 자식 엘리먼트. li:last-child는 각 리스트의 마지막 아이템을 반환한다.
:only-child	형제가 없는 모든 엘리먼트를 반환한다.
:nth-child(n)	n번째 자식 엘리먼트. li:nth-child(2)는 가 리스트의 두 번째 리스트 아이템을 반환한다.
:nth-child(even:odd)	짝수 또는 홀수 자식 엘리먼트. li:nth-child(even)은 각 목록의 짝수 번째 자식 엘리먼트를 반환한다.
:nth-child(Xn+Y)	전달된 공식에 따른 n번째 자식 엘리먼트. Y는 0인 경우 생략 가능하다. li:nth-child(3n)은 3의 배수 번째 아이템을 반환한다. li:nth-child(5n+1)은 5의 배수 +1 번째 아이템을 반환한다.
:even / odd	페이지 전체의 짝수/홀수 번째 엘리먼트. li:even은 모든 짝수 번째 아이템을 반환한다.
:eq(n)	n번째로 일치하는 엘리먼트
:gt(n)	n번째 엘리먼트(포함되지 않음) 이후의 엘리먼트와 일치
:lt(n)	n번째 엘리먼트(포함되지 않음) 이전의 엘리먼트와 일치

jQuery 정의 필터 셀렉터. 대상 엘리먼트를 식별해내는 데 강력한 기능을 준다.

셀렉터	설명
:animated	현재 애니메이션이 적용되고 있는 엘리먼트를 선택한다.
:button	모든 버튼을 선택한다 (input[type=submit], input[type=reset], input[type=button], button)
:checkbox	체크박스 엘리먼트만 선택한다.( input[type=checkbox])
:checked	선택된 체크박스나 라디오 버튼만 선택한다. (CSS에서 지원)
:contains(foo)	텍스트 foo를 포함하는 엘리먼트만 선택한다.
:disabled	인터페이스에서 비활성화 상태인 모든 폼 엘리먼트를 선택한다.(CSS에서 지원)
:enabled	인터페이스에서 활성화 상태인 모든 폼 엘리먼트를 선택한다.(CSS에서 지원)
:file	모든 파일 엘리먼트를 선택하다 (input[type=file])
:header	헤더 엘리먼트만 선택한다. (<h1> ~ <h6>)
:hidden	감춰진 엘리먼트만 선택한다.
:image	폼 이미지를 선택한다. (input[type=image])
:input	폼 엘리먼트만 선택한다. (input, select, textarea, button)
:not(fileter)	필터의 값을 반대로 변경한다.
:parent	빈 엘리먼트를 제외하고, 텍스트도 포함해서 자식 엘리먼트를 가지는 엘리먼트를 선택한다.
:password	패스워드 엘리먼트만 선택한다.(input[type=password])
:radio	라디오 버튼 엘리먼트만 선택한다.(input[type=button])
:selected	선택된 엘리먼트만 선택한다.
:submit	전송 버튼을 선택한다 (button[type=submit], input[type=submit])
:text	텍스트 엘리먼트만 선택한다.(input[type=text])
:visible	보이는(visible) 엘리먼트만 선택한다.

## jQuery 기본문법 – selector 예

- \$('a') – page 내 모든 <a>
- \$('div a') – page 내 <div> 하위에 있는 <a>
- \$('#test') – id가 test인 태그
- \$('.btn') – class가 btn인 모든 태그
- \$('tr:odd') - <tr> 태그들 중 홀수번째들
- \$('tr:last') – 문서내의 마지막 <tr>
- \$('b:contains('hi')) – hi를 content로 가진 b태그
- \$('div:has('ul')) - <ul>을 가진 <div> 태그
- \$('input:checkbox') – input태그중 checkbox
- \$('td:nth-child(2n)') – 2의 배수 번째 <td>



## **`$(document).ready(callback 함수)`**

- jQuery 이벤트 메서드 중 하나
- 문서의 DOM 요소들을 조작 할 수 있는 시점에서 호출
- javascript의 load 이벤트와 비슷
- 예

```
$(document).ready(function(){  
    alert('a');  
})  
);
```

# jQuery event메소드 - 1

- bind('event type',[data] , 처리 function)
  - Event 등록
- unbind('event type')
  - Event 해제
- one('event type', function)
  - 한번만 실행되는 event 처리
- live("event type", function)
  - 현재 조건이 맞거나 앞으로 조건이 맞을 모든 요소에 event 처리
  - Ajax연동 시 강력
- event 도우미 메소드
  - bind() 메소드를 wrapping한 메소드들로 event명을 메소드 명으로 사용
  - <http://api.jquery.com/category/events/> 참고
  - ex : click(function(){}), mouseover(function(){})

## jQuery event메소드 - 2

- 처리 function
  - 구문 : function([event]){처리 코드}
  - 인수 : event 객체 받을 수 있음.
  - event 객체 property
    - data – bind() 에서 넘긴 Data
    - target – event 소스 객체

```
$('#mybtn').bind('click', {name:'kim', age:20}, myFun);
```

```
function myFun(event){  
  alert(event.data.name);  
  alert(event.data.age);  
  event.target.css('color', 'red');  
}
```

# Content/DOM 변경 및 조회 메소드

- `html()`
  - 선택요소의 `html` 내용을 가져옴. `innerHTML`과 동일
- `html(value)`
  - 선택요소에 인수로 받은 `value`를 넣는다.
  - `value`내에 `html`태그가 있는 경우 태그로 들어간다.
- `text()`
  - 선택요소 내의 `html`태그 요소를 제외한 내용을 가져옴. `innerText`와 동일.
- `text(value)`
  - 선택요소 내에 인수로 받은 `value`를 넣는다.
  - `value`내의 `html`태그도 `text`로 들어간다.
- `val()`
  - `input` 태그의 `value` 값을 조회
- `val(value)`
  - 인수로 받은 `value`를 `input` 태그의 `value` 값을 설정

## Content/DOM 변경 및 조회 메소드

- append(value)
  - 선택요소내의 content 뒤에 value를 붙인다.
- prepend(value)
  - 선택요소내의 content 앞에 value를 붙인다.
- appendTo(selector)
  - 선택된 요소를 selector로 선택된 요소내 내용 뒤에 붙인다.
- prependTo(selector)
  - 선택된 요소를 selector로 선택된 요소내 내용 앞에 붙인다.

# Content/DOM 변경 및 조회 메소드

- after(value)
  - 선택된 요소 뒤에 value를 붙인다.
- before(value)
  - 선택된 요소 앞에 value를 붙인다.
- insertAfter(selector)
  - 선택된 요소를 selector 뒤에 붙인다.
- insertBefore(selector)
  - 선택된 요소를 selector 앞에 붙인다.
- remove()
  - 선택된 요소들을 삭제한다.
- empty()
  - 선택된 요소들의 자식 요소를 삭제 한다.

# Traversing(탐색) 메소드 - 추가 필터링

- 조회한 요소들의 집합에서 원하는 요소를 찾는 메소드들
- eq(index)
  - index와 일치한 요소 조회
- filter(표현식)
  - 지정된 표현식과 일치하는 요소들 조회
- is(표현식) :
  - 조회한 요소들 중 표현식을 만족하는 요소가 있으면 true(하나라도 있으면) 없으면 false
- not(표현식) :
  - 표현식과 일치하지 않는 요소들 조회
- end() :
  - 필터링 이전 단계로 돌리는 메소드
    - \$('div').eq(1).addClass('a').end() -> \$('div') 중 첫번째 것에 'a' 클래스를 붙인뒤 다시 \$('div") 선택상태로 돌린다.

# Traversing(탐색) 메소드 – 찾기1

- find(표현식)
  - 조회한 요소의 자식요소 중 표현식과 일치한 것을 검색
- add(표현식)
  - 조회한 요소에 표현식에 맞는 요소를 추가한다.
- next([표현식])
  - 조회한 요소 바로 다음에 오는 형제 요소 선택
- nextAll([표현식])
  - 조회한 요소 다음의 모든 형제 요소



# Traversing(탐색) 메소드 - 찾기2

- prev([표현식])
  - 조회한 요소 바로 전에 오는 형제요소 선택
- prevAll([표현식])
  - 조회한 요소 이전의 모든 형제 요소 선택
- siblings([표현식])
  - 조회한 요소의 모든 형제 요소(자신은 제외) 선택
- parent([표현식])
  - 조회한 요소의 부모요소
- children([표현식])
  - 조회한 요소의 자식 요소

# CSS관련 메소드

- `css(name)`
  - 조회된 요소들 중 첫 요소의 `name` 스타일 값을 조회
- `css(name, value)`
  - 조회된 모든 요소에 스타일 적용
- `css(properties)`
  - 조회된 모든 요소에 스타일 적용
  - 여러 개의 스타일을 한번에 설정 시 사용
  - `properties`는 `key:value` 쌍의 모음
- `addClass(class명)`
  - 조회된 모든 요소에 인수의 클래스 추가
- `removeClass(class명)`
  - 조회된 모든 요소에 인수의 클래스 제거
- `toggleClass(class명)`
  - 조회된 모든 요소에 인수의 `class`가 없으면 추가, 있으면 제거

# 태그의 Attribute 관련 메소드

- attr(name)
  - 조회된 요소의 name Attribute의 값을 조회
- attr(key, value)
  - 모든 조회된 요소들의 attribute를 설정
- attr(key, 함수)
  - 모든 조회된 요소들의 attribute를 설정. 값은 함수 실행 후 return 값으로 설정
- attr(properties)
  - 모든 조회된 요소들의 attribute를 설정.
  - 여러 개의 attribute 설정시 사용
  - properties는 key:value 쌍의 모음
- removeAttr(name)
  - 모든 조회된 요소들의 해당 attribute를 제거

# jQuery Ajax

- load('url')
  - url 의 응답 값을 가져와 지정된 위치에 뿌린다.
  - ex) \$('div:first').load('test.jsp')
- load('url', '요청 파라미터 data');
  - ex) \$('#mydiv').load('test.jsp', 'id=a&pwd=b');
  - ex) \$('#first').load('src.jsp', {id:'a',pwd:'b'});
- load('url', [data], function)
  - load ing 종료 후 호출할 function 등록

# jQuery Ajax

- \$.getJSON("url", data, callback함수)
  - 서버로 부터 비동기적으로 JSON 데이터를 받아온다.
  - 인수
    - 1번 : data를 받아올 url
    - 2번 : CGI로 보낼 요청파라미터값
    - 3번 : 서버에서 **정상** 응답을 받은 뒤 실행 될 함수. 실패시 호출 안됨.
  - ex  
\$.getJSON('TestServlet', function(data){});
- \$.get(url, data, callback함수), \$.post(url, data, callback함수)
  - 비동기적으로 GET/POST 방식 요청
  - 인수
    - 1번 : 요청 url
    - 2번 : CGI로 보낼 요청 파라미터 값
    - 3번 : 서버에서 **정상** 응답을 받은 뒤 실행 될 함수. 실패시 호출 안됨.

## jQuery Ajax

- \$.ajax({property})
  - 모든 ajax 함수의 basic 함수
  - property : ajax 호출 관련 설정을 넣어준다.
  - 주요 property
    - url – 호출 할 URL
    - type – 통신 방식 GET, POST
    - data – 요청파라미터
    - dataType – 응답 데이터 타입 : ex) text, json, xml
    - error – 에러 시 처리할 callback 함수
    - success – 통신 성공 시 처리할 callback 함수
    - beforeSend – 요청전송 전 호출 되는 callback 함수