웹표준

jQuery

jQuery

first coding

jQuery

http://jquery.com

1.1 개요

- jQuery
  - 모든 브라우저에서 동작하는 클라이언트 자바스크립트 라이브러리
  - 2006년 1월, 존 레식John Resig이 BarCamp NYC에서 발표
  - 무료로 사용 가능한 오픈 소스 라이브러리
  - jQuery는 다음 기능을 위해 제작됨

    - **문서 객체 모델과 관련된 처리를 쉽게 구현**
    - **일관된 이벤트 연결을 쉽게 구현**
    - **시각적 효과를 쉽게 구현**
    - **Ajax 애플리케이션을 쉽게 개발**

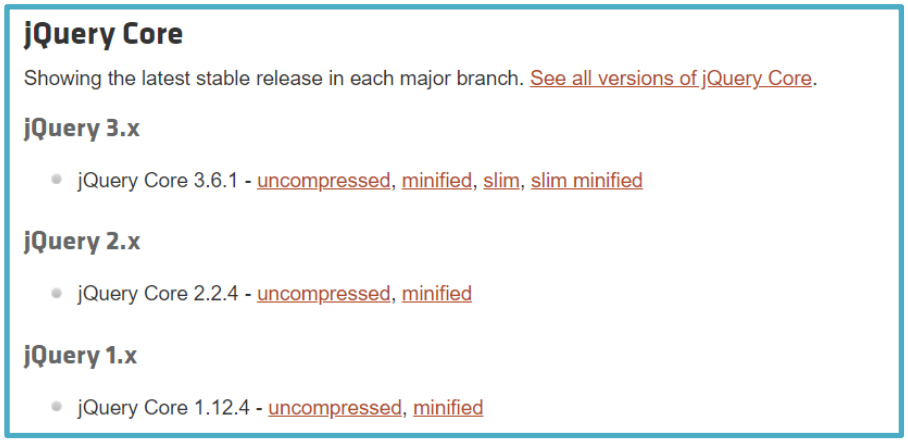
1.2 다운로드

- jQuery 다운로드
  - jQuery를 내려받으려면 http://jquery.com 에 접속
  - 메인 화면에서 다운로드 버튼을 누르면 다운로드 페이지로 이동

- jQuery 사용
  - 첫 번째 방법은 CDN 호스트를 사용하는 방법
  - 두 번째는 직접 다운받아 사용하는 방법
  - 구글과 마이크로소프트에서 CDN 호스트를 지원
  - CDNContent Delevery Network은 사용자에게 간편하게 콘텐츠를 제공하는 방식을

1.2 다운로드

- CDN 호스트 사용
  - https://releases.jquery.com/
  - https://releases.jquery.com/jquery/



  - CDN을 이용한 라이브러리 로드

```
<script src="https://code.jquery.com/jquery-1.12.4.js"
        integrity="sha256-Qw82+bXyGq6MydymqBxNPYTaUXXq7c8v3CwiYwLLNXU="
crossorigin="anonymous"></script>
```

1.2 다운로드

- CDN 호스트 사용
  - 그 밖의 CDN 호스트

Other CDNs

The following CDNs also host compressed and uncompressed versions of jQuery releases. Starting with jQuery 1.9 they may also host sourcemap files; check the site's documentation.

Note that there may be delays between a jQuery release and its availability there. Please be patient, they receive the files at the same time the blog post is made public. Beta and release candidates are not hosted by these CDNs.

- Google CDN
- Microsoft CDN
- CDNJS CDN
- jsDelivr CDN

1.3 $(document).ready( )

- $(document).ready( )
  - jQuery를 이용하여 문서객체 처리를 위해서는 다음 코드로 시작

```
// $() -> 팩토리 함수 : jquery 객체 생성 메소드
// document 객체를 -> jquery 객체로 변환 생성
$(document).ready(function () {

  // 페이지 로드 완료 후 실행
  // window.onload = ()=>{}

}
```

```
$().ready(() => { alert('onload') })
```

1.4 기본 선택자

- 팩토리함수를 이용한 Jquery 객체 생성
  - 기본 선택자
    - jQuery 메소드의 가장 기본적인 형태 이며 jQuery에서 가장 중요한 역할

```
$('*').css('color', 'red');
```

```
<h2>header - 0</h2>
<h2>header - 1</h2>
<h2>header - 2</h2>
<h2>header - 3</h2>
<h2>header - 4</h2>
<h2>header - 5</h2>
```

1.4 기본 선택자

- **전체 선택자**
  - 전체 선택자 : * 사용
  - 태그 선택자
  - 아이디 선택자
  - 클래스 선택자
  - 자손 선택자와 후손 선택자
  - 속성 선택자

1.5 필터 선택자

- 입력 양식 필터 선택자

```
<h2>header - 0</h2>
<h2>header - 1</h2>
<h2>header - 2</h2>
<h2>header - 3</h2>
<h2>header - 4</h2>
<h2>header - 5</h2>
```

```
$('input[type=text]').val('안녕') // set 메소드

var value = $('input[type=text]').val() // get 메소드
console.log(value)

let val2 = $('input:text').val()
console.log(val2)
```

1.5 필터 선택자

- 입력 양식 필터 선택자

```
<select>
    <option>2022</option>
    <option>2021</option>
    <option>2020</option>
    <option>2019</option>
</select>

<input id="userID" type="text" value="hello">
```

```
// 5초 후에 select의 선택 항목의 value 값을 확인
setTimeout(function () {
    var selVal = $('select>option:selected').val();
    console.log('selVal', selVal);

    var chk = $('input:checkbox:checked').val();
    console.log('chk',chk);

}, 5000);
```

1.6 배열 관리

- **배열 처리**
  - jQuery로 배열을 처리할 때는 each ( ) 메소드를 사용
    - each ( ) 메소드는 매개변수로 입력한 callback 함수로 반복문 처럼 객체나 배열의 요소를 처리
    - `$.each(배열원본, callback : 배열의 하나의 요소를 처리하는 함수, 매개변수 배열의 index 값, 요소 데이터)`

```
$.each()
```

```
$.each(
    collection,
    callback: (indexInArray: number, valueOfElement: any)
)
```

```
$(selector).each(
    collection,
    callback: (indexInArray: number, valueOfElement: any)
)
```

1.6 배열 관리

- 자바스크립트 배열 관리

```
// 배열을 다루는 .each() 메소드
// $.each(배열원본, callback : 배열의 하나의 요소를 처리하는 함수, 매개변수 배열의 index 값, 요소 데이터)

// 배열생성
let links = [
    { name: '네이버', url: 'http://www.naver.com' },
    { name: '다음', url: 'http://www.daum.net' },
    { name: '카카오', url: 'http://www.kakao.com' }
];

let output = '';

$.each(links, (index, item) => {
    console.log(index, item.name);
    output += '<a href="' + item.url + '">' + item.name + '</a>\n';
});

console.log(output);

//document.querySelector('#links').innerHTML=output;
$('#links').html(output);
```

1.6 배열 관리

- jQuery 배열 관리

```
<h1>header - 0</h1>
<h1>header - 1</h1>
<h1>header - 2</h1>
<h1>header - 3</h1>
<h1>header - 4</h1>
```

```
$('h1').each(function (index, item) {

    console.log(index, item.innerHTML);
    console.log(index, $(item).html());

});
```

first coding

문서 객체 선택과 탐색

2.1 기본 필터 메소드

- 필터 메소드
  - 기본 필터 메소드

```
$(selector).filter(selector)

$(selector).filter(() => {
    return true | false ;
})
```

```
$('table tr').filter(':even').css('background-color', '#AAA');
$('h2').css('background', 'orange').filter(':even').css('color', 'white')
$('h1').eq(3).css('background', 'orange');
$('h1').last().css('background', 'blue');
$('h1').first().css('background', 'green');
```

```
$('h3').filter(function (index) {
    return index % 3 == 0;
}).css({
    backgroundColor: 'black',
    color: 'white'
})
```

2.2 xml 조작

- xml 특정 태그 선택
  - XML 문자열 생성

```
// xml 문자열을 만든다.
var xml = '';
xml += '<links>';
xml += '    <link>';
xml += '        <name>네이버</name>';
xml += '        <url>http://www.naver.com</url>';
xml += '    </link>';
xml += '    <link>';
xml += '        <name>다음</name>';
xml += '        <url>http://www.daum.net</url>';
xml += '    </link>';
xml += '    <link>';
xml += '        <name>카카오</name>';
xml += '        <url>http://www.kakao.com</url>';
xml += '    </link>';
xml += '</links>';
```

2.2 xml 조작

- xml 특정 태그 선택
  - $.parseXML( ) 메소드와 each( ) 메소드

```
// 문자열 -> xml 문서객체로 변환
// $.parseXML(문자열)
var xmlDoc = $.parseXML(xml);

// xml 요소 엘리먼트를 찾아주는 메소드 -> find(엘리먼트 이름)
// xml 문서객체 -> $

var html = '';


$(xmlDoc).find('link').each(function(index){

    // 반복 처리 할 코드작성


});
```

2.2 xml 조작

- xml 특정 태그 선택
  - $.parseXML( ) 메소드와 each( ) 메소드

```
$(xmlDoc).find('link').each(function(index){

    var name = $(this).find('name').text();
    var url = $(this).find('url').text();
    console.log(name, url);

    html += '<div>';
    html += '    <h1>'+name+'</h1>';
    html += '    <p>'+url+'</p>';
    html += '</div>';


});

$(document).ready(function(){
    $('body').html(html);
});
```

first
coding

문서 객체 조작

3.1 문서 객체의 클래스 속성 추가

- 문서 객체의 클래스 속성 추가
  - addClass( )

    문서 객체에 클래스 속성을 추가할 때 사용하는 메소드

```
<style>

    .title {
        background-color: crimson;
        color: floralwhite;
        float: left;
    }

</style>
```

```
$('h2').addClass('title');
```

3.2 문서 객체의 클래스 속성 제거

- **문서 객체의 클래스 속성 제거**
  - removeClass()

    문서 객체에 클래스 속성을 제거할 때 사용하는 메소드

```
$('h1').removeClass('title')
```

3.3 문서 객체의 클래스 속성 검사

- **문서 객체의 클래스 속성 검사 및 속성 추가**
  - attr()

    속성 값을 처리하는 메소드

```
<img class="img" src='../images/starwars.jpg'>
```

    - 속성확인

```
var srcPath = $('.img').attr('src');
console.log('이미지 경로', srcPath);
```

    - 속성 추가

```
$('.img').attr('width', 200);
```

```
$('.img').attr('width', function(index){
    return (index+1)*100;
});
```

```
$('img').attr({
    width : 100,
    height: function(index){
        return (1+index)*100;
    }
});
```

3.4 문서 객체의 속성 제거

- 문서 객체의 속성 제거
  - removeAttr()

    문서 객체의 속성을 제거할 때 사용

```
$('.img').removeAttr('height');
```

3.5 문서 객체의 스타일 검사

- 문서 객체의 스타일 처리

```
<h1 class="title">header - 0</h1>
```

  - css()

    문서 객체의 스타일 값을 확인하거나 추가할 때 사용

    - 태그의 스타일 확인

```
// 태그의 css 속성값 확인
var color = $('h1.title').css('color');
console.log(color);
```

    - 태그의 스타일 추가

```
// css 적용
$('h1').css('background', 'red');
```

```
var colors = ['black', 'orange', 'yellow'];

$('h1').css('background', function(index){
    return colors[index];
});
```

```
$('h1').css({
    color : 'green',
    backgroundColor :
function(index){
        return colors[index];
    }
});
```

3.6 문서 객체의 내부 검사

- **문서 객체의 내부 처리**
  - 기존의 자바스크립트에서 문서 객체의 innerHTML, textContent 을 이용해서 처리
  - jQuery 에서는 html(), text() 를 사용
    - 내부 요소 확인

```
// 태그의 요소값 확인
console.log('html() : ', $('#menu1').html());
console.log('text() : ', $('#menu2').text());
```

```
<div id="menu1">
    <span>Menu Text1</span>
</div>
<div id="menu2">
    <span>Menu Text2</span>
</div>
<div id="menu3">
    <span>Menu Text3</span>
</div>
```

    - 내부 요소 변경

```
// 태그 요소 변경
$('#menu1').html('<h1>TITLE</h1>');
$('#menu2').text(<h1>TITLE</h1>');
```

```
$('div').html(function(index, html){

    console.log(index, html);

    $(this).html('<h1>'+ html + index + '</h1>');

});
```

3.7 문서 객체 제거

- **문서 객체 제거**
  - remove()
    - 문서 객체를 제거할 때 사용
  - empty()
    - 문서 객체 내부 요소를 모두 삭제 할 때 사용

```
<button id="btn1">메뉴 삭제</button>
<button id="btn2">내용 삭제</button>
```

```
$('#menu1').remove();
$('#menu2').empty();
```

3.8 문서 객체 생성(1)

- 문서 객체 생성(1)
  - $()
    - 문서 객체를 생성할 때 사용

3.8 문서 객체 생성(1)

- 문서 객체 생성(1)
  - $ ( ) 메소드의 매개변수에 HTML 태그를 문자열로 넣기만 하면 문서 객체가 생성

```
$('<h1></h1>');
```

  - 문서 객체 생성 및 텍스트 노드 추가

```
$('<h1></h1>').html('Hello~!!!');
```

3.8 문서 객체 생성(1)

- 문서 객체 생성(1)
  - 문서 객체 연결

```
$('<h1></h1>').html('Hello~!!!').appendTo('body')
```

  - 문서 객체 생성

```
$('<h1>Hello~!!! Javascript!!!</h1>').appendTo('body')
```

3.9 문서 객체 생성(2)

- 문서 객체 생성(2)
  - 텍스트 노드를 갖지 않는 문서 객체를 생성하는 방법
    - img 태그를 생성

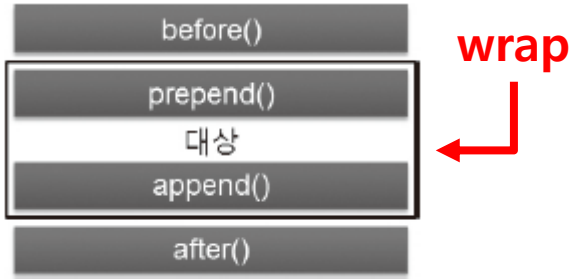
```
$('<img>').attr('src','../mini1.jpg').attr('width', 100).appendTo('body')
```

```
$('<img>').attr({
    src : '../mini1.jpg',
    width : 200
}).appendTo('body');
```

3.10 문서 객체 삽입

- 문서 객체 삽입
  - 문서 객체에 문서 객체를 추가하는 메소드

메서드 이름	설명
$(A).appendTo(B)	A를 B의 뒷부분에 추가합니다.
$(A).prependTo(B)	A를 B의 앞부분에 추가합니다.
$(A).insertAfter(B)	A를 B의 뒤에 추가합니다.
$(A).insertBefore(B)	A를 B의 앞에 추가합니다.
$(A).append(B)	B를 A의 뒷부분에 추가합니다.
$(A).prepend(B)	B를 A의 앞부분에 추가합니다.
$(A).after(B)	B를 A의 뒤에 추가합니다.
$(A).before(B)	B를 A의 앞에 추가합니다.



3.10 문서 객체 삽입

- 문서 객체 삽입
  - append ( ) 메소드의 사용 형태

```
$(selector).append(content, content, content, …)
```

```
$(selector).append(function(index){})
```

3.10 문서 객체 삽입

- 문서 객체 삽입
  - 여러 개의 문서 객체를 한꺼번에 입력 가능

```
var html1 = '<a href="#">test</a>';
var html2 = '<p>TEST</p>';
var html3 = '<div>DIV-0</div><div>DIV-1</div><div>DIV-2</div>';

$('body').append('<h3>Hello~!!! JS</h3>');
$('body').append(html1, html2, html3, '<hr>');
```

3.10 문서 객체 삽입

- 문서 객체 삽입
  - 함수를 이용하는 문서객체생성도 가능

```
// 배열생성
var links = [
    { name: '네이버', url: 'http://www.naver.com' },
    { name: '다음', url: 'http://www.daum.net' },
    { name: '카카오', url: 'http://www.kakao.com' }
];

$('div').append(function(index){
    return '<h3><a href="'+links[index].url+'">'+links[index].name+'</a></h3>';
});
```

first
coding

이벤트

4.1 이벤트 연결 기본

- 이벤트 연결 기본
  - jQuery로 이벤트를 연결하는 가장 기본적인 방법은 on ( ) 메소드 사용

```
$(selector).on (eventName, function(event) {})
```

```
$(selector).on (object : {})
```

4.1 이벤트 연결 기본

- 이벤트 연결 기본
  - h1 태그를 click 이벤트에 연결하고 이벤트 발생 시 이벤트 발생 객체에 '+' 글자 추가

```
<h1>header-0</h1>
<h1>header-1</h1>
<h1>header-2</h1>
```

```
// h1태그를 캐스팅 -> click 이벤트를 연결
$('h1').on('click',function(){
    // this -> 이벤트가 발생한 엘리먼트 객체를 가르킨다.
    $(this).html(function(index, html){
        return html+'+';
    });
});
```

4.1 이벤트 연결 기본

- 이벤트 연결 기본
  - on ( ) 메소드의 매개변수에 객체를 넣어 이벤트 연결

```
<h1>header-0</h1>
<h1>header-1</h1>
<h1>header-2</h1>
```

```
$('h1').on({
    //click : function(){},
    mouseenter : function(){
        $(this).addClass('reverse');
    },
    mouseleave : function(){
        $(this). removeClass('reverse');
    }
});
```

4.2 간단한 이벤트 연결

- 간단한 이벤트 연결

```
$(selector).method(function(event){})
```

  - 연결 이벤트 메소드

blur	focus	focusin	focusout	load
resize	scroll	unload	click	dblclick
mousedown	mouseup	mousemove	mouseover	mouseout
mouseenter	mouseleave	change	select	submit
keydown	keypress	keyup	error	ready

4.3 매개변수 context

- 매개변수 context
  - jQuery 메소드는 매개변수를 두 개 입력 가능
  - 특정 부분에 선택자를 적용하고 싶을 때 사용 → 매개변수 context

```
$: JQueryStatic
(selector: string, context?: Element | JQuery) => JQuery
```

  - 매개변수 context는 selector가 적용하는 범위를 한정

4.3 매개변수 context

- 매개변수 context
  - context 객체

```html
<div>
    <h3>header 1</h3>
    <p> paragraph1</p>
    <input type="hidden" value="1">
</div>
<div>
    <h3>header 2</h3>
    <p> paragraph2</p>
</div>
<div>
    <h3>header 3</h3>
    <p> paragraph3</p>
</div>
```

```javascript
$('div').click(function(e){

    // 이벤트가 발생한 div -> this
    // $('h3', this) -> 이벤트가 발생한 div안에 위치한 h3을 선택
    var h3 = $('h3', this).text();
    var p = $('p', this).text();

    alert(h3 + '\n' + p);

    console.log(e);

})
```

4.4 이벤트 객체

- **이벤트 객체**
  - 모든 이벤트 리스너는 이벤트 객체가 있음
    - jQuery의 이벤트 객체는 모든 브라우저가 같은 방법으로 사용하고 같은 속성을 가지고 있음

이벤트 객체 속성	설명
event.pageX	브라우저의 화면을 기준으로 한 마우스의 X 좌표 위치
event.pageY	브라우저의 화면을 기준으로 한 마우스의 Y 좌표 위치
event.preventDefault( )	기본 이벤트를 제거합니다.
event.stopPropagation( )	이벤트 전달을 제거합니다.

4.4 이벤트 객체

- 이벤트 객체
  - 모든 이벤트 리스너는 이벤트 객체가 있음

```
<a href="http://www.naver.com">click</a>
```

```
$('a').click(function(e){
    alert('a tag click');

    var e = e || window.event;

    // 이벤트 버블링 제거
    // if(e.stopPropagation) {
    //     e.stopPropagation();
    // }
    // 기본 이벤트 제거
    // e.preventDefault();

    return false;

});
```

4.5 이벤트 연결 범위 한정

- 이벤트 연결 범위 한정
  - 이벤트를 연결할 때 on ( ) 메소드를 사용

```
JQuery.on(events: string, data: any,
  handler: (eventObject: JQueryEventObject, ...args: any[]) => any
```

  - 동적으로 만들어지는 태그에는 이벤트 연결이 되지 않기 때문에 이벤트 연결 시 범위의 지정이 필요함.

4.5 이벤트 연결 범위 한정

- 이벤트 연결 범위 한정

```html
<div id="wrap">
    <h1>header</h1>
</div>
```

```
$('h1').click(function(){
    var length = $('h1').length;
    var oldHtml = $(this).html();
    $('#wrap').append('<h1>'+length+' - '+oldHtml+'</h1>');
});
```

```
$('#wrap').on('click' , 'h1', function () {

    var length = $('h1').length;
    var oldHtml = $(this).html();
    $('#wrap').append('<h1>' + length + ' - ' + oldHtml + '</h1>');
});
```

4.6 키보드 이벤트

- **키보드 이벤트**
  - textarea 태그에 keyup 이벤트를 연결
  - keyup 이벤트가 발생하면 글자의 개수를 받아 출력
  - keydown 이벤트 진행 순서

1 사용자가 키보드를 누릅니다.
2 keydown 이벤트가 발생합니다.
3 글자가 입력됩니다.
4 keypress 이벤트가 발생합니다.
5 사용자가 키보드에서 손을 뗍니다.
6 keyup 이벤트가 발생합니다.

  - 입력한 글자 수를 표시해야 하므로 keyup 이벤트를 사용

4.6 키보드 이벤트

- 키보드 이벤트

```
<textarea cols="50" rows="10"></textarea>
<span>0</span>/150
```

```
$('textarea').keyup(function () {

    var cnt = $(this).val().length;
    var remain = 150 - cnt;

    console.log(cnt, remain);

    if (remain < 0) {
        alert('소개서는 150자까지 작성이 가능합니다.');
        var str = $(this).val().substr(0, 150);
        $(this).val(str);
        $('span').text('150');
        return false;
    }
    $('span').text(cnt);
});
```

4.7 입력 양식 이벤트

- 입력 양식 이벤트

이벤트 이름	설명
change	입력 양식의 내용을 변경할 때 발생합니다.
focus	입력 양식에 초점을 맞추면 발생합니다.
focusin	입력 양식에 초점이 맞추어지기 바로 전에 발생합니다.
focusout	입력 양식에 초점이 사라지기 바로 전에 발생합니다.
blur	입력 양식에 초점이 사라지면 발생합니다.
select	입력 양식을 선택할 때 발생합니다(input[type="text"] 태그 및 textarea 태그 제외).
submit	submit 버튼을 누르면 발생합니다.
reset	reset 버튼을 누르면 발생합니다.

4.7 입력 양식 이벤트

- 입력 양식 이벤트
  - submit 이벤트

```
<form id="myForm">
    <input type="submit">
</form>
```

```
$('#myForm').submit(function () {
    alert('myForm submit !!');
    return false;
});
```

4.7 입력 양식 이벤트

- 입력 양식 이벤트
  - check 속성 변경
    - type 속성이 checkbox와 radio인 input 태그의 상태를 변경하는 이벤트는 click 이벤트가 아닌 change 이벤트

```
<input type="checkbox" id="allCheck">
<label for="allCheck">all 전체 동의</label>

<div id="checkItem">
    <input type="checkbox"> <label>A type</label>
    <input type="checkbox"> <label>B type</label>
    <input type="checkbox"> <label>C type</label>
</div>
```

```
$('#allCheck').change(function(){
    // 선택 상태이면 true, 미 선택 상태이면 false
    if(this.checked){
        // #checkItem 안에 있는 input type=checkbox -> checked=true
        $('#checkItem').children().prop('checked', true);
    } else {
        $('#checkItem').children().prop('checked', false);
    }
})
```

first
coding

AJAX

CONTENTS

- aJax API

  https://api.jquery.com/category/ajax/
- $.aJax()

  https://api.jquery.com/jQuery.ajax/
- 추가적인 aJax 메서드

  https://api.jquery.com/category/ajax/shorthand-methods/
- 폼 관련 보조 메서드

  https://api.jquery.com/category/ajax/helper-functions/

$ajax()

- 기본
  - jQuery를 활용한 Ajax 메서드

```
$.ajax(options);
```

```
$.ajax(url, option);
```

$ajax()

- 기본
  - $ajax() 메서드(2)

```
<script>
        $(document).ready(function () {
            $.ajax('data.html', {
                success: function (data) {
                    $('body').append(data);
                }
            });
        });
</script>
```

$ajax()

- 기본
  - $ajax() 메서드(2)

```
<script>
        $(document).ready(function () {
            $.ajax({
                url: 'data.html',
                success: function (data) {
                    $('body').append(data);
                }
            });
        });
</script>
```

$ajax()

- 기본
  - $ajax() 메서드(2)

```
<input type="text" id="txt" value="파라미터 데이터">
```

```
<script>
        $(document).ready(function () {
            $.ajax({
                url: 'parameter.jsp',
                type: 'GET',
                data: {
                    name: $('#txt').val(),
                    price: 'test'
                },
                success: function (data) {
                    $('body').append(data);
                }
            });
        });
</script>
```

$ajax()

```
$.ajax({
        url : requestUrl,
        type : 'DELETE',
        async : true,
        data : JSON.stringify(requestParam),
        dataType : "json",
        timeout : 10000,
        contentType : "application/json",
        beforeSend : function(){
            $('.wrap-loading').removeClass('display-none');
        },
        complete : function(){
            $('.wrap-loading').addClass('display-none');
        },
        success : function(data){
            alert(data);
        },
        error : function(request,status,error){
            alert("code:"+request.status +"\n"+"message:"+request.responseText+"\n"+"error:"+error);
            $('.wrap-loading').addClass('display-none');
        }
});
```

$ajax()

- $ajax() Option

옵션 속성 이름	설명	자료형
async	비동기 통신 플래그. true (비동기 통신)에서 요청 이후 응답입 없어도 비동기 처리를 계속한다. false로 설정 (동기 통신)하면 통신에 응답이 있을 때까지 브라우저는 잠겨 이벤트 실행이 대기한다.	**true** , false
complete	AJAX 통신 완료될 때 호출되는 함수 success이나 error가 호출된 후에 호출되는 Ajax Event	Function
data	요청 매개변수를 지정	Object, String
dataType	서버에서 반환되는 데이터 형식을 지정 생략했을 경우는, jQuery이 MIME 타입 등을 보면서 자동으로 결정 "json" : JSON 형식 데이터로 평가하고 JavaScript의 개체로 변환합니다.	"xml", "html", "script", "jsonp", "json", "text"
error	통신에 실패했을 때 호출되는 Ajax Event	Function
contentType	서버에 데이터를 보낼 때 사용 content - type 헤더의 값 "application / x - www - form - urlencoded"	
timeOut	제한 시간 (밀리초)을 설정합니다.	숫자
type	HTTP 통신의 종류를 설정합니다.	"**GET**", "POST", "PUT", "DELETE"
success	AJAX 통신이 성공하면 호출되는 Ajax Event 입니다. 돌아온 데이터와 dataType 지정한 값 2 개의 인수를받습니다.	
URL	대상 URL을 지정	

추가적인 ajax 메서드

- 추가적인 jQuery Ajax 메서드

메서드 이름	설명
jQuery.get()	Get방식으로 수행 jQuery.get(url [, data] [, success] [, dataType]) jQuery.get([settings])
jQuery.post()	Post방식으로 수행 jQuery.post(url [, data] [, success] [, dataType]) jQuery.post([settings])
jQuery.getJSON()	Get 방식으로 ajax를 수 행해 Json 데이터로 반환 jQuery.getJSON(url [, data] [, success])
.load()	Ajax를 수행하고 선택자로 선택한 문서 객체 안에 데이터 삽입 .load(url [, data] [, complete])

추가적인 ajax 메서드

- jQuery.get()

```
<script>
        $(document).ready(function () {
            $.get('data.html', function (data) {
                $('body').html(data);
            });
        });
</script>
```

추가적인 ajax 메서드

- jQuery.post()

```
<script>
        $(document).ready(function () {
            $.post('data.html', function (data) {
                $('body').html(data);
            });
        });
</script>
```

추가적인 ajax 메서드

- jQuery.getJSON()

```
<script>
    $(document).ready( function() {
        $.getJSON('data.json', function(data) {
            $.each(data, function(key, value) {
                $('body').append('<h1>'+value.name+':'+value.price+'</h1>');
            });
        });
    });
</script>
```

추가적인 ajax 메서드

- .load()

```
<script>
        $(document).ready(function () {
            $('body').load('data.html');
        });
</script>
```

추가적인 ajax 메서드

- XML 조작

```
<script>
   $(document).ready(function () {
      $.ajax({
          url: 'data.xml',
          success: function (data) {
             $(data).find('product').each(function () {
                // 변수를 선언합니다.
                var name = $(this).find('name').text();
                var price = $(this).find('price').text();

                // 출력합니다.
                $('<h1></h1>').text(name + ':' + price).appendTo('body');
             });
          }
      });
   });
</script>
```