Javascript

참조 사이트

- W3School Javascript 튜토리얼
  - https://www.w3schools.com/
- MDN (Mozilla Developer Network) Javascript 튜토리얼
  - https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference

웹 문서의 구성요소

- HTML
  - 웹 문서 내용의 구조를 정의
- CSS(Cascade Style Sheet)
  - 웹 문서의 디자인을 처리
- Javascript
  - 웹 문서 내에서 동적인 부분을 처리

Javascript 개요

- 웹 문서(HTML) 내에서 실행되는 프로그램을 구현하기 위한 언어
  - 초창기에 웹 문서 내에서 요청 파라미터 검증 등 사용자의 요청에 대한 동적 처리를 위해 만들어짐
- HTML 문서 내에 작성하며 Web Browser 에 내장되 있는 Javascript 엔진에 의해 실행된다.
- 특징
  - 객체지향 언어
  - 이벤트 중심 언어

HTML 페이지에 Javascript 코드 추가

- <script> 태그 내에 작성
  - script 태그는 HTML 문서 내 어디든지 올 수 있다.

```
<script>
    document.write("안녕하세요");
</script>
```

- 외부 Javascript 코드 추가
  - Javascript 코드를 HTML 외부에 작성 - **.js 로 저장**
  - <script **src="파일 URL"**></script>
  - 태그 내에 Javascript 코드를 넣어서는 안 된다.

```
<script src="scripts/myscript.js"></script>
<script>
    //코드
</scrip>
```

Data Type

- Javascript 데이터 타입 – 리터럴 타입과 , 객체(참조)타입이 있다.
- **리터럴 타입/값**
  - **number** : 정수, 실수
  - **string** : 문자열
    - 값은 따옴표로 감싼다. (큰따옴표, 작은 따옴표 상관없음)
  - **boolean** : 논리값 – true/false
  - **null** : 값이 없음
  - **undefined** : 값이 없음
- **객체(Object) 타입**
  - **object : 객체**
  - **array** : 배열
  - **function** : 함수
- **typeof** 연산자
  - 값의 Data Type을 문자열로 알려주는 연산자
  - 구문
    - **typeof 값; typeof 변수;**

```
typeof  "hello";   //string
typeof  20;        //number
typeof  true;      //boolean
```

변수

- 변수 선언
  - **var** 변수명 [= 값]
  - Javascript는 변수명에 Datatype을 붙이지 않는다.
  - 변수는 값을 저장하는 저장소 이며 모든 타입의 값을 다 대입할 수 있다.
  - 예

```
var num = 20;
var address = "경기도 성남시";
var num1, num2, num3;
var num1, num2=20, num3=40;
var ageList = [10, 20, 30, 40, 50];
```

  - 변수 명 규칙
    - Java와 동일
    - 사용할 수 있는 문자 : 글자, 숫자, 특수기호는 $와 _
    - 숫자는 두 번째 글자부터 사용가능
    - 키워드는 사용할 수 없다.

변수

- 변수에 값 대입하기
  - 대입 연산자 사용
  - 값 직접 대입: var y = 20;
  - 표현식(계산식)으로 대입
    - 숫자 표현식 : var num = 6+7; var num = (10/2) * 3; var num = 10.22-3.21 ;
    - 논리 표현식 : var flag = x >2; var flag = 10 == 15;
    - 문자열 표현식 : var str = "super"+"computer"; var date = "3월"+21+"일";
  - 다른 변수의 값 대입 : var x = y;
  - 다른 function 의 실행 결과 대입 : var k = abc();

조건문

- if – else 문

구문
if(조건){
  구문
}**else if**(조건){
  구문
}**else**{
  구문
}

조건 : 모든 타입의 값이 다 들어올 수 있다.

타입 별 false인 값

- string : 빈 문자열
- number : 0
- null, undefined

- switch case 문

switch(표현식){
  case 값 :
    구문
     break;
  case 값 : 구문
  default : 구문
}

표현식 : 문자열, 숫자, 논리 다 올 수 있다
break 키워드를 이용해 구문을 빠져 나온다.

반복문

- for, while, do while, for in
  - 반복문 조건식에는 모든 타입의 값이 다 들어 올 수 있다.

- for 구문

```
for([초기식]; [조건식]; [증감식]){
        반복할 구문
}
```

```
var sum = 0;
for(var i = 0; i < 10; i++){
  sum = sum +i;
}
```

- while 구문

```
[초기식]
while(조건식){
        반복구문
        [증감식]
}
```

```
var i =0;
var sum = 0;
while(i < 10){
        sum = sum +i;
        i++;
}
```

반복문

- do while문

```
[초기식]
do{
        반복구문
        [증감식]
}while(조건식);
```

```
var sum = 0;
var i = 0;
do{
        sum = sum +i;
        i++;
}while(i < 10);
```

- for in
  - 객체의 속성 명을 반복 조회 시 사용한다.
  - 배열의 index 들 조회

```
for(변수 in 객체||배열){
  반복 구문
}
```

```
var obj = {name:"김영수", age:20}
for(var property in obj){
    console.log(obj[property]);
}
```

function - 함수

- 프로그램 수행의 일련 과정을 하나의 단위로 묶어주는 수행 block
  - 구현된 script 실행 시 바로 실행되지 않고 호출 시 실행된다.
  - 반복 호출이 가능.
- **함수는 객체이다. ➔ 함수는 값이다.**
  - **Javascript에서는 함수가 객체로 취급된다.**
  - **전달인자**나 **리턴 값**으로 사용할 수도 있다.

함수 기본 구문

```
function 함수이름(arg [,arg]){
        //함수 코드
        [return value]
}
```

```
function test(name){
    alert(name);
    return name+"님";
}
```

함수 정의 표현식(함수 리터럴)

```
var 함수이름 = function(arg [,arg]){
        //함수코드
        [return value]
}
```

```
var test = function(name){
    alert(name);
    return name+"님";
}
```

전역변수와 지역변수

- 지역변수
  - 함수 내부에 선언된 변수
  - scope(사용범위) : 함수 내
- 전역변수
  - 함수 외부에 선언된 변수(<script> 내에 선언 된 변수)
  - window 객체의 속성으로 등록된다.
  - scope : 전체 페이지 내 (page 시작~page 종료시)
- this : 함수 내에서 전역변수 또는 객체의 변수에 접근할때 사용

```
<script type="text/javascript">
  var myvar = 10; //전역변수
  function abc(){
     var myvar = 20; //지역변수
     alert(myvar);
     alert(this.myvar);
  }
</script>
```

Event 처리

- 자바 스크립트는 사용자의 요청처리를 Event 모델을 통해 처리한다.
  - Event Source : Event가 발생 한 컴포넌트
  - Event
    - Event Source의 상태를 바꾸게 만드는 Action(동작).
    - Ex) 마우스로 버튼 클릭, Text 입력양식에 글 입력, 페이지 로딩 등..
  - Event Handler
    - Event 발생시 처리하는 코드를 등록하는 것.
  - Listener
    - Event Source에서 Event가 발생하는 것으 감시하다 발생시 Handler 호출
    - Web Browser

예

```
<input type="button" value="확인" onclick="alert('button클릭')"/>
<form action="a.jsp" onsubmit="alert('전송합니다.')">
```

Event 처리

- 주요 Event와 Handler

Event	Handler	설명
load	**onload**	해당 페이지가 로딩 되었을 때(처음 읽힐 때) 발생
focus	**onfocus**	입력 양식를 선택해서 포커스가 주어졌을 때
blur	**onblur**	포커스가 폼의 입력 양식을 벗어났을 때
change	**onchange**	입력 양식 select에서 선택 item이 바뀌었을때
mousemove	**onmousemove**	해당영역에 마우스를 움직였을 때 발생
mouseover	**onmouseover**	해당 영역에 마우스가 올라갔을 때 발생
mouseout	**onmouseout**	해당 영역에서 마우스가 나갔을 때 발생
mousedown	**onmousedown**	해당 영역에서 마우스버튼을 눌렀을 때 발생
mouseup	**onmouseup**	해당 영역에서 누르던 마우스버튼를 떼었을 때 발생
click	**onclick**	해당 영역에서 마우스를 클릭 했을 때 발생
keydown	**onkeydown**	해당 영역에서 키보드를 눌렀을 때 발생
keyup	**onkeyup**	해당 영역에서 누르고 있던 키보드를 떼었을 때 발생
keypress	**onkeypress**	해당 영역에서 키보드를 계속 누르고 있을 때 발생
submit	**onsubmit**	폼의 내용을 전송 할 때 발생
reset	**onreset**	폼의 내용을 초기화 시킬 때

Javascript와 객체

- 자바스크립트는 객체지향언어(Object Oriented Language)
  - Java Script가 지원하는 객체만 사용할 수 있었으나 1.2 버전부터는 개발자가 직접 객체를 만들 수 있다.
- 객체의 종류
  - 내장 객체(Built-In 객체) : 웹 브라우저에서 제공하는 객체
    - **자바스크립트 내장객체** : Javascript 코드 작성시 유용한 기능을 제공해 주는 객체들
      - Array, String, Date, Math 등
    - **DOM 객체** (브라우저 내장객체) : 웹 브라우저의 각 요소들을 객체로 제공
  - 사용자 정의 객체 : 사용자가 만드는 객체

Javascript 내장객체 - Array

- 배열 객체
  - 생성 구문
    - 변수 = [value, value, value…]
  - 값 접근
    - 변수[index]
  - 속성
    - length : 배열의 크기

```
var arr = [10, 20, 30, 40, 50];

for(var idx = 0; idx<arr.length; idx++){
    alert(arr[idx]);
}
```

Javascript 내장객체 - Array

- 주요 메소드
  - push(value[,value,...])
    - 배열에 값을 추가
  - pop()
    - 배열의 마지막 index의 값을 return하고 삭제한다.
  - shift()
    - 배열의 첫번째 index의 값을 return하고 삭제한다.
  - splice(start_idx [,length])
    - 배열의 일부값들을 삭제. 시작 idx와 삭제할 개수(개수 생략하면 나머지 다 삭제)
  - slice(start_idx[, end_idx]);
    - 배열의 일부 값들을 조회하여 새로운 배열로 return 한다.(시작 idx와 끝 idx. 끝 idx는 포함 안됨)
  - concat(배열객체[,배열객체,..])
    - 배열을 합쳐 새로운 배열을 생성
  - join([구분자])
    - 배열의 element들을 구분자로 이어진 문자열로 만들어 return. 기본구분자는 ' , ' 임.

Javascript 내장객체 - Date

- 날짜와 시간을 다루는 객체
  - 생성 구문
    - 변수 = new Date();
    - 변수 = new Date([년, 월, 일, 시, 분, 초, 1/1000]);
  - 주요 메소드

메 소 드	설 명
getFullYear()	1900년대 4자리
getYear()	1900년대 2자리(2000년대는 4자리로 나옴)
getMonth()	월(0~11) 예) 0:1월, 11:12월 …
getDate()	날짜
getDay()	요일(0~6) 예) 0:일요일, 6:토요일
getHours()	시(0~23)
getMinutes()	분(0~59)
getSeconds()	초(0~59)
getTime()	1970년 1월 1일 이후 시간을 1/1000초 단위로 나타낸 값

```
var today = new Date();
alert(today.getFullYear()+"."+(today.getMonth()+1)+"."+today.getDate());
```

Javascript 내장객체 - String

- 문자열을 관리하는 객체
  - 값은 따옴표로 감싸준다.(작은 따옴표, 큰 따옴표 상관없다.)
  - 생성구문
    - 변수 = "값";
    - 변수 = '값'
    - 변수 = new String("값");
  - 주요메소드

메 소 드	설 명
charAt(index)	지정된 위치에서 문자 찾기
indexOf(string)	지정된 문자의 위치를 왼쪽부터 찾기
lastIndexOf(string)	지정된 문자의 위치를 오른쪽부터 찾기
substring(index1, index2)	문자열의 일부를 추출하기 index1 ~ index2-1
toLowerCase()	소문자로 변환하기
toUpperCase()	대문자로 변환하기
concat(string)	두 문자열을 합치기
slice(start_index, end_index)	문자열의 일부를 추출하기
substr(start_index, length)	문자열을 length만큼 잘라내기

DOM(Document Object Model) Tree 구조

- 문서의 구성요소들(elements들)을 웹브라우저가 객체화 해서 관리하는 방식
  - 문서 구성요소를 객체화 한 것을 DOM이라 한다.
  - DOM 객체들을 Tree 구조로 계층화 해서 관리
- 동적으로 문서의 내용, 구조, 스타일에 접근하여 변경, 생성
- 구성
  - 노드 (Node) : Tree 구조를 구성하는 요소
    - Element(요소) 노드 : 태그
    - Text 노드
    - Attribute(속성) 노드 : 태그내 속성
- Web browser 가 HTML **문서를 로딩 시점 생성** 및 구성 한다.

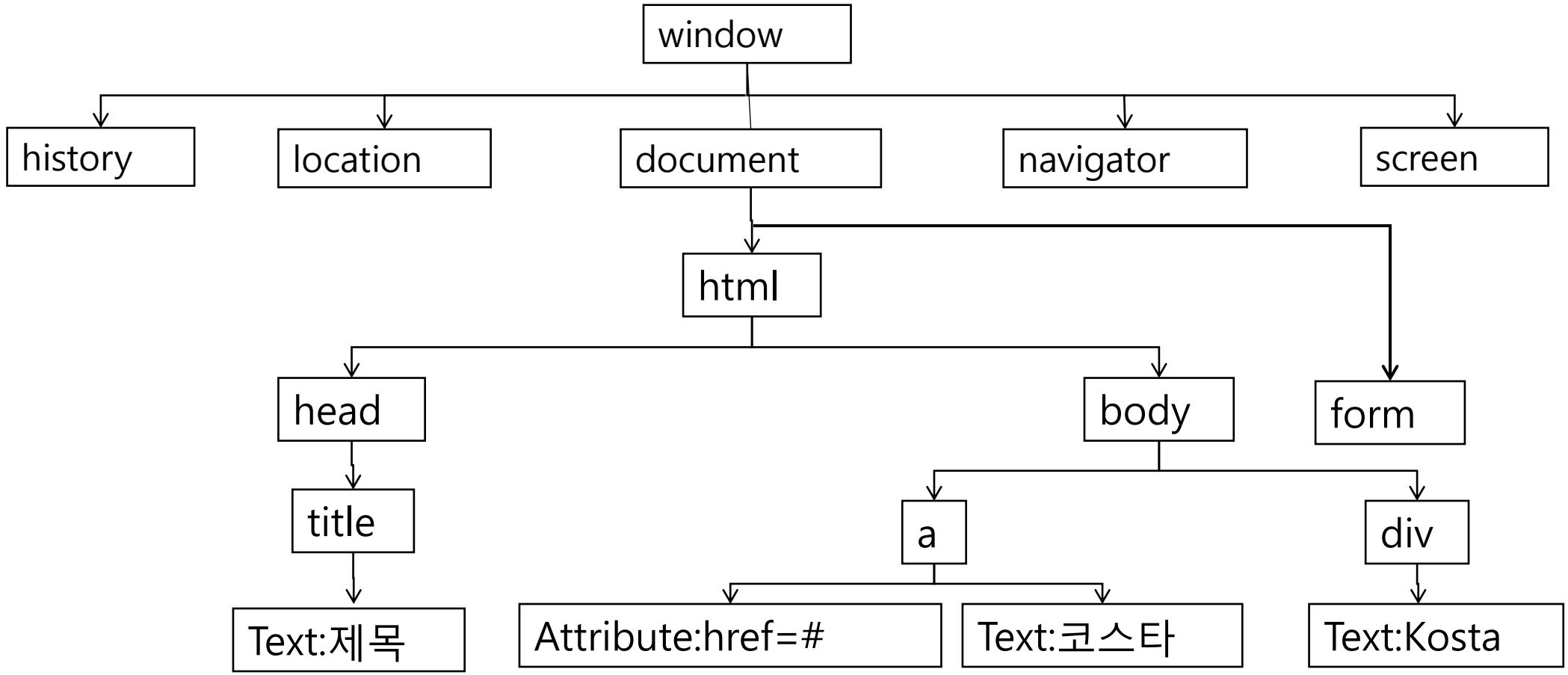
DOM(Document Object Model)

- HTML 문서 구조

```
<html>
   <head>
        <title>제목</title>
   </head>
   <body>
        <a href="#">코스타</a>
        <div>Kosta</div>
        <form> .. </form>
   <body>
</html>
```

DOM(Document Object Model)

- DOM Tree 구조



DOM 객체 (브라우저 내장객체)

- window 객체
  - 브라우저 내장객체 중 최상위 Node에 위치한 객체
    - 자바스크립트 동작에 필요한 전역 객체로 동작
    - 사용자가 보는 브라우저 창을 나타낸다.
    - 브라우저 창의 HTML 문서에 접근하는 시작객체로 사용된다.
  - 접근방법
    - window.속성, window.메소드
    - window 는 생략 가능
  - 주요 메소드

메소드	설 명
open()	새로운 윈도우(창) 열기. 열린 창에서 연 창을 **opener** 통해 접근할 수 있다.
close()	열려있는 브라우저(윈도우) 닫기
alert()	간단한 메시지를 보여주기 위한 dialog box
confirm()	사용자로부터 어떠한 작업을 확인 받기 위한 dialog box
back()	이전 페이지로 이동
parseInt(String) **parseFloat(String)**	인수의 String을 정수, 실수 형태로 변환
isNaN(String)	인수의 String이 숫자 형태가 아니면 true, 숫자형태이면 false리턴
eval(String)	String을 Javascript 코드로 변환 해서 실행

DOM 객체 (브라우저 내장객체)

- document객체
  - 브라우저에 보여지는 HTML 문서를 가리키는 객체
    - 문서의 구성요소에 접근
    - 문서의 내용을 변경
  - window의 하위 객체
  - 접근 방법
    - window.document 또는 document
  - 주요 메소드

메소드	설명
write("문자열")	문자열을 문서에 출력한다.
DOM 객체 접근 및 생성 메소드	

DOM 객체 (브라우저 내장객체)를 통한 Form 처리

- form 객체
  - <form> 객체
  - document의 하위 계층 객체
  - 접근 방법
    - window.document.form의 name속성값
  - 주요 속성

속성	설명
action	<form>의 action 속성
method	<form>의 method 속성
enctype	<form>의 encoding 속성
name	<form>의 name 속성. form 객체접근시사용

  - 주요 메소드

메소드	설명
reset()	입력양식 초기화. reset버튼 의 동작 처리
submit()	입력된 값을 action의 url로 전송. submit 버튼의 동작 처리

DOM 객체 (브라우저 내장객체)를 통한 Form 처리

- text, password객체
  - <input type="text">, <input type="password"> 객체
  - form의 하위 객체
  - 접근 방법
    - window.document.form_name.태그의name속성값
  - 주요 속성

속성	설명
name	태그의 name 속성
value	태그의 value 속성
기타 input type='text||password' 가지는 속성들	

  - 주요 메소드

메소드	설명
focus()	객체에 포커스를 준다.

DOM 객체 (브라우저 내장객체)를 통한 Form 처리

- textarea객체
  - <textarea> 객체
  - form의 하위 객체
  - 접근 방법
    - window.document.form_name.textarea_name 속성값
  - 주요 속성

속성	설명
name	태그의 name 속성
value	textarea에 입력된 값(value)
cols	열 속성값 – 한줄에 입력될 수 있는 최대 글자수
rows	행 속성값 – 보이는 총 행수

  - 주요 메소드

메소드	설명
focus()	객체에 포커스를 준다.

DOM 객체 (브라우저 내장객체)를 통한 Form 처리

- checkbox 객체
  - <input type="checkbox"> 객체
  - form의 하위 객체
  - 접근 방법
    - window.document.form_name.checkbox_name 속성값
  - 주요 속성

속성	설명
name	checkbox name 속성
value	checkbox value 속성
checked	checkbox checked 속성으로 선택상태. true/false

  - 주요 메소드

메소드	설명
click()	체크박스 클릭 처리

DOM 객체 (브라우저 내장객체)를 통한 Form 처리

- radio 객체
  - <input type="radio"> 객체
  - form의 하위 객체
  - 접근 방법
    - window.document.form_name.radio_name속성값
  - 주요 속성

속성	설명
name	radio name 속성
value	radio value 속성
checked	radio checked 속성으로 선택상태. true/false

  - 주요 메소드

메소드	설명
click()	라디오버튼 클릭 처리

DOM 객체 (브라우저 내장객체)를 통한 Form 처리

- select 객체
  - <select> 객체
  - form의 하위 객체
  - select 태그의 value(사용자가 선택하는 값)은 <option> 태그에 있다.
    값에 접근하기 위해서는 select를 통해 option 객체로 접근한다.
  - 접근 방법
    - window.document.form_name.select_name 속성값
      선택 Item(option) 접근 : window.document.form.select_name.**options - 배열**
  - 주요 속성

속성	설명
name	<select> name 속성
selectedIndex	선택된 item(option)의 index. (option의 index는 0부터 시작)
length	option의 총 개수
options	select내의 option객체들을 가진 배열(Node List)

DOM 객체 (브라우저 내장객체)를 통한 Form 처리

- option 객체
  - select 내의 <option> 태그 객체
  - select 객체의 options 속성을 통해 배열로 받아 접근한다.
  - 주요 속성

속성	설명
text	option의 내용
value	value 속성의 값 (value속성이 없으면 내용이 value의 역할을 한다.)
selected	option의 선택 여부. true/false

jQuery

jQuery 개요

- Javascript Library
  - 웹 브라우저 마다 다르게 작성해야 되는 크로스 브라우저 자바스크립트 코드를 최대한 공통된 방법으로 작성할 수 있도록 하는 것을 목표로 존 레식(John Resig) 이 2006년 발표 한 Javascript Library
  - http://jquery.com

John Resig - 위키백과

- 다운로드
  - http://jquery.com/download
  - jquery-X.X.X-min.js 다운(compressed 버전)
  - library 파일 다운 로드 후 사용하고자 하는 페이지에서 script 태그의 src 속성으로 등록해 사용

기본 문법

- jQuery 객체
  - jQuery 의 모든 기능을 제공하는 객체.
  - 구문 : jQuery 또는 $ 로 사용한다.
  - Ex) $.ajax(….), $.get(…)
- Selector함수
  - jQuery가 제공하는 기능을 적용할 DOM 객체를 찾아 jQuery 객체에 넣는 함수.
  - jQuery(“selector”) 또는 $(“selector”)
  - Ex) $(“div”), $(“#email”), $(“안녕”)
- jQuery 또는 $ 는 jQuery 객체를 호출하고 jQuery(“selector”) 또는 $(“selector”)는 jQuery Selector 함수를 호출한다.

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

태그명으로 조회	
CSS	jQuery

div{
background-color:#D024FE;
}
	$("**div**")

ID 속성값으로 조회	
CSS	jQuery

#address{
background-color:#D024FE;
}
	$("**#address**")

class 속성값으로 조회	
CSS	jQuery

.hobbyInput{
background-color:#D024FE;
}
	$("**.hobbyInput**")

태그기반 Selector (선택자) – 태그명이나 속성을 이용

셀렉터	설명	Ex)
*	모든 태그	$("*")
E	태그 명 E와 일치하는 모든 element node	$("div"), $("input")
E.C	E 요소들 중 class속성값이 C인 모든 element node	$("div.result"), $(".input_form")
E#I	E 요소들 중 id속성값이 I인 모든 element node	$("span#layer"), $("#message")
E F	E 로 선택된 요소의 자손인 모든 F element node	$("ul li"), $("#layer>li")
E>F	E 로 선택된 요소의 자식인 모든 element node	$("table > tr") $("#layer>li")
E+F	E 로 선택된 요소의 바로 다음 형제 element node	$("input+span")
E~F	E 로 선택된 요소 다음에 나오는 모든 형제 element node	$("div~span")
E:has(F)	자손 태그로 F를 가지는 태그명이 E인 element node	$("table:has(tbody)")
E[A]	태그명이 E인 것중 속성으로 A를 가지는 모든 element node	$("input[value]");
E[A=V]	태그명이 E인 것중 속성 A의 값이 V인 모든 element node	$("input[type=checkbox]")
E[A^=V]	태그명이 E인 것중 속성 A의 값이 V로 시작하는 모든 element node	$("a[href^=www]")
E[A$=V]	태그명이 E인 것중 속성 A의 값이 V로 끝나는 모든 element node	$("a[href$=net]")
E[A*=V]	태그명이 E인 것중 속성 A의 값에 V가 들어가는 모든 element node	$("a[href*=google]")

위치기반 Selector(선택자)

선택된 요소들 중 위치를 기준으로 걸러내는 Selector

E:first	E 들 중 첫번째 요소	$("div:first")
E:last	E 들 중 마지막 요소	$("table:last")
E:eq(정수)	E들 중 정수번째 요소 – **0부터 시작**	$("div:eq(5))
E:odd	E들 중 홀수번째 요소 – **0부터 시작**	$("tr:odd")
E:even	E들 중 짝수번째 요소 – **0부터 시작**	$("td:even")
E:gt(정수)	E들 중 정수 이후 번째 요소들 – **0부터 시작**	$("td:gt(3)")
E:lt(정수)	E들중 정수 이전 번째 요소들 – **0부터 시작**	$("td:lt(5)")
E:first-child	E 들 중 첫번째 자식 요소(태그)	$("td:first-child")
E:last-child	E 들 중 마지막 자식 요소(태그)	$("tr:last-child")
E:nth-child(정수)	E 들 중 정수번째 자식 요소(태그) – **1부터 시작**	$("ul li:nth-child(4))
E:nth-child(even||odd)	E 들 중 짝수||홀수번째 자식 요소(태그)	$("ul li:nth-child(even))
E:nth-child(정수n)	E 들 중 정수배수번째 자식 요소(태그)	$("td:nth-child(3n),$("td:nth-child(3n+2)

필터 기반 Selector(선택자)

선택된 요소들 중에서 원하는 요소를 걸러내기 위한 Selector

셀렉터	설명
E:button	버튼 요소(모든 버튼요소를 다 선택한다.)
E:checkbox	Input type이 checkbox
E:radio	Input type이 radio
E:checked	Radio와 checkbox중 체크된 모든 요소
E:text	input type이 text인 요소
E:password	Input type이 password인 요소
E:file	Input type이 file인 요소
E:selected	Select의 option 중 선택 되 있는 요소
E:hidden	화면상에서 안 보이는(감춰진)요소
E:visible	화면에 보이는 요소
E:enable	활성화된 입력 폼 요소
E:disable	비활성화된 입력 폼
E:header	<hX> 요소들

셀렉터	설명
E:input	모든 입력 양식 요소
E:submit	Submit 버튼요소
E:reset	Reset 버튼 요소
E:parent	Text를 포함해 자식 노드를 가진 요소
E:contains (문자열)	Text로 문자열을 가진 요소
E:not(필터)	필터의 반대가 되는 요소

예)
$("input:text")
$(":radio:checked")
$(":checkbox:checked")
$("#no:selected")
$("td:contains(만두)")
$(":checkbox:not(:checked)")

jQuery 기본문법 – selector 예

- $("selector") – selector는 따옴표(큰/작은 상관없다)로 감싼다.
  – selector로 조회된 dom 객체들은 jQuery객체에 Node List(배열)로 저장됨.
- $('a') – 모든 <a> node들.
- $('.layer:eq(3)') – class 속성값이 layer인 node들 중 3번째 위치한 node.
- $('#test') – id속성이 "test"인 node.
- $(':checked:first') – 모든 node중 checkbox인 것 중 첫번째 node(첫번째 체크박스)
- $('tr:odd:eq(3)') - <tr> 태그들 중 홀수번째 node 중 3번째 것.
- $('tr:last') – 문서내의 마지막 <tr> 요소.
- $('td:contains("hi")') – hi 가 들어간 문자열을 text로 가진 <b> 요소들
- $('div:has(ul)') - 을 자손으로 가진 node들.
- $(':checkbox:checked') –checkbox중 check된 것들.
- $('td nth-child(2n)') –전체 td들 중 2의 배수 번째 자식인 <td> node들.

$(document).ready(callback 함수)

- 문서 로딩 할 때 문서의 DOM Tree가 완성되는 시점에 매개변수의 함수를 호출한다.
- 역할 : Javascript의 load 이벤트
- Javascript와 HTML(구조) 를 분리하기 위해 주로 사용.
- 예

```
$(document).ready(function(){
        $(“#btn”).on(“click”, remove);
   }
);
```

jQuery event 관련 메소드

- DOM의 Element 객체(태그) 에 Event Handler를 추가하는 메소드들
  - 태그의 onXXX="" 를 처리하는 메소드들.
- on()
  - $(event src).on("event type", callback 함수)
    - on() 메소드 실행 시점에 있는 elements 들에 이벤트 핸들러 추가
  - $(event src의 parent요소).on("event type", "selector", callback 함수)
    - on() 메소드가 실행 시점 뿐 아니라 이후에 추가되는 elements 들에 이벤트 핸들러 추가
- $(event src).off("event type")
  - on을 통해 설정된 event 삭제
- $(event src).one('event type', function)
  - 한번만 실행되는 event 처리

jQuery event 관련 메소드

- Event Handling도우미 메소드
  - bind() 메소드를 wrapping한 메소드들로 event명을 메소드 명으로 사용
  - http://api.jquery.com/category/events/ 참고
  - event명(callback 함수) 형태를 가진다.
- ex
  - $(event src).click(function(){})
  - $(event src). mouseover(function(){})
  - $(event src). hover(function(){}, function(){})
    - event handling 도우미 메소드로 mouseover와 mouseout 동시 처리

Content/DOM 변경 및 조회 메소드

- html()
  - 선택요소의 하위 노드를 html 태그들을 포함한 문자열로 조회
- html(value)
  - 선택요소의 하위 노드를 인수로 받은 value를 변경한다.
  - value내에 html태그가 있는 경우 태그(DOM Node객체로 변환해서) 처리한다.
- text()
  - 선택요소의 하위 요소 중 Text 노드들만 을 문자열로 조회.(태그는 제외된다)
- text(value)
  - 선택요소의 하위 노드를 인수로 받은 value로 변경한다.
  - value는 텍스트 노드로 만들어져 들어간다. (태그도 text로 들어간다.)
- val()
  - 입력양식의 value 값을 조회.
- val(value)
  - 입력양식의 value 값을 인수로 받은 value로 변경.

Content/DOM 변경 및 조회 메소드

- append(value)
  - 선택요소의 마지막 자식 노드로 value를 붙인다.(추가한다.)
  - value는 DOM Node객체로 변환되어 추가된다.
- prepend(value)
  - 선택요소의 첫번째 자식 노드로 value를 붙인다.(추가한다.)
  - value는 DOM Node객체로 변환되어 추가된다.
- appendTo(selector)
  - 선택된 요소를 selector로 선택된 요소의 마지막 자식노드로 추가한다.
- prependTo(selector)
  - 선택된 요소를 selector로 선택된 요소의 첫번째 자식노드로 추가한다.

Content/DOM 변경 및 조회 메소드

- after(value)
  - 선택된 요소 다음 형제 노드로 value를 추가한다.(붙인다)
  - value는 DOM Node객체로 변환되어 추가된다.
- before(value)
  - 선택된 요소 이전 형제 노드로 value를 추가한다.(붙인다)
  - value는 DOM Node객체로 변환되어 추가된다.
- insertAfter(selector)
  - 선택된 요소를 selector로 선택된 요소의 다음 형제노드로 추가한다..
- insertBefore(selector)
  - 선택된 요소를 selector로 선택된 요소의 이전 형제노드로 추가한다.
- remove()
  - 선택된 요소들을 DOM Tree 구조에서 제거한다.
- empty()
  - 선택된 요소들의 **모든 자식 요소를** DOM Tree 구조에서 제거한다.

Traversing(탐색) 메소드 – 추가 필터링

- 조회한 DOM 요소 집합에서 추가적으로 원하는 요소를 찾는 메소드들
- eq(index)
  - index와 일치한 요소 조회
- filter(표현식)
  - 지정된 표현식과 일치하는 요소들 조회
- not(표현식) :
  - 표현식과 일치하지 않는 요소들 조회
- first()
  - 조회한 요소가 노드리스트일 때 그 중 첫 번째 요소 조회
- last()
  - 조회한 요소가 노드리스트일 때 그 중 마지막 요소 조회
- slice(start, [end])
  - 조회한 요소가 노드리스트일 때 특정 범위의 요소를 조회할 때

표현식 – selector 표현식
[표현식] – 표현식 생략가능

Traversing(탐색) 메소드 – 기타 탐색

- find(표현식)
  - 조회한 요소의 자손요소 중 표현식과 일치한 것을 검색
- children([표현식])
  - 조회한 요소의 자식 요소
- parent([표현식])
  - 조회한 요소의 부모요소
- add(표현식)
  - 조회한 요소에 표현식에 맞는 요소를 추가한다.
- is(표현식) :
  - 조회한 요소들 중 표현식을 만족하는 요소가 있으면 true(하나라도 있으면) 없으면 false
- end() :
  - 필터링 이전 단계로 돌리는 메소드
    - $('div').eq(1).addClass('a').end() -> $('div') 중 첫 번째 것에 'a' 클래스를 붙인 뒤 다시 $('div") 선택상태로 돌린다.

Traversing(탐색) 메소드 – 형제노드 탐색

- next([표현식])
  - 조회한 요소 바로 다음에 오는 형제 요소 선택
- nextAll([표현식])
  - 조회한 요소 다음의 모든 형제 요소
- 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

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
  - properties는 Javascript 객체(key:value 쌍의 모음) 로 설정.
- addClass(class명)
  - 조회된 모든 요소에 인수의 클래스 추가
- removeClass(class명)
  - 조회된 모든 요소에 인수의 클래스 제거
- toggleClass(class명)
  - 조회된 모든요소에 인수의 class가 없으면 추가, 있으면 제거

태그의 Attribute 관련 메소드

- prop(속성명)
  - 조회된 요소의 name Atribute의 값을 조회
- prop(속성명, 속성값)
  - 모든 조회된 요소들의 attribute를 설정
- prop(key, 함수)
  - 모든 조회된 요소들의 attribute를 설정. 값은 함수 실행 후 return 값으로 설정
- prop(properties)
  - 모든 조회된 요소들의 attribute를 설정.
  - 여러 개의 attribute 설정시 사용
  - properties는 Javascript 객체(key:value 쌍의 모음) 로 설정.
- removeProp(name)
  - 모든 조회된 요소들의 해당 attribute를 제거

AJAX 처리

Asynchronous **J**avaScript **a**nd **X**ML

Ajax 란

- Single-Page Application 제작을 위한 개발 기법
  - XMLHttpRequest 객체와 XML, JSON 데이터 포멧을 이용
- 기존 기술
  - Flash, Flex, ActiveX, Silver Light
  - 기존 기술은 브라우저 지원기술이 아니므로 따로 설치해야 한다.
- XMLHttpRequest
  - Javascript 에서 HTTP 요청을 지원하기 위한 객체
  - 비동기적 요청처리를 지원한다.
- 장점
  - 웹브라우저가 지원하는 기술들(Javascript, XML, JSON) 을 이용해 제작한다.
  - 페이지 변경 없이 서버와의 통신이 가능하다.(Single-Page Application)
  - 수신 데이터 양이 기존 웹 어플리케이션보다 적다.
  - 비동기적 처리가 가능하다

JSON

- **J**ava**S**cript **O**bject **N**otation (Javascript 객체 표기법)
- Text 기반으로 여러 개의 값을 name-value 쌍으로 표현하기 위한 기법
- 기존에 데이터 교환 시 사용하던 XML의 단점을 극복하기 위해 제안됨
- 장점
  - 데이터 표현이 단순하고 직관적이다.
  - Javascript 객체로 변환해서 사용이 가능하므로 데이터 처리가 간편하다.
- json.org

JSON 문법

- JSON 기본문법
  - 데이터는 { } 로 감싼다.
    - 데이터는 name : value 형식으로 표현하며 ' , ' 를 구분자로 사용해 나열한다.
  - 배열은 [ ] 로 감싼다.
    - 배열은 값들을 ' , ' 를 구분자로 값들을 나열한다.
  - name : " "로 감싸거나 생략 가능하다. 단 공백과 같은 특수문자가 들어간 경우 반드시 감싸야 한다.
  - value : 문자열, 숫자, 배열, true/false, null, JSON 데이터
    - string은 " "로 감싼다.

예

```
• { id:"abc",  password:"1111",  name:"홍길동",  address:"서울", age:20}
• ["이순신", "강감찬", "홍길동"]
• [
     {id:"id-1",  password:"1111",  name:"홍길동",  address:"서울", age:20, marriage:false},
     {id:"id-2",  password:"2222",  name:"이순신",  address:null, age:25, marriage:true},
     {id:"id-3",  password:"3333",  name:"강감찬",  address:"광주", age:22, marriage:false}
  ]
• {
    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" } ]
  }
```

jQuery Ajax 처리 – 기본 메소드 ajax()

- $.ajax({property})
  - XMLHttpRequest를 이용해 HTTP 요청을 처리하는 메소드.
    - property : ajax 호출 관련 설정을 넣어준다.
    - Javascript 객체로 속성을 설정해 전달.
  - 주요 property
    - url – 호출 할 URL : String
    - type – 통신 방식 GET, POST : String
    - data – 요청파라미터
      - 형식 : queryString 또는 javascript객체
    - dataType – 응답 데이터 타입. 값-text, json, jsonp, xml : String
    - beforeSend – 요청전송 전 호출 되는 함수 : 함수
    - success – 통신 성공 시 처리할 함수 (HTTP 응답 코드 : 200) : 함수
    - error – 통신 실패시 처리할 함수 (HTTP 응답 코드 : 200 이외): 함수
    - complete – success나 error에 등록된 함수 실행후 호출될 함수 : 함수

jQuery Ajax 처리 – Helper 메소드

- $.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. 서버에서 오류 전송시 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. 서버에서 오류 전송시 호출 안됨.