

Javascript

Chapter 1. 자바스크립트 시작하기

- 1.1 자바스크립트란?
- 1.2 자바스크립트 기초문법
- 1.3 변수
- 1.4 연산자

Chapter 2. 제어문

- 2.1 조건문(if)
- 2.2 선택문(switch)
- 2.3 반복문(for)

Chapter 3. 객체

- 3.1 내장객체
- 3.2 브라우저객체모델(BOM)
- 3.3 문서객체모델(DOM)

Chapter 4. 함수

- 4.1 함수란?
- 4.2 return문
- 4.3 지역변수와 전역변수

Chapter 5. 이벤트

- 5.1 이벤트란?
- 5.2 이벤트객체

Chapter 1. 자바스크립트 시작하기

1.1 자바스크립트란?

자바스크립트란 개발자가 만든 문서에 방문자가 방문하여 어떤 동작을 취했을 때, 그 동작에 대응하여 반응이 일어날 수 있도록 해주는 언어이다. 즉, 우리가 흔히 보는 GNB(Global Navigation Bar)요소에 마우스를 올리면 그에 해당하는 서브메뉴가 펼쳐지는 것과 같다.

자바스크립트 언어의 특징

1) 자바스크립트는 인터프리터 언어이다.

자바스크립트는 코드가 작성된 순서대로 윗줄부터 순차대로 구문분석을 한다.

2) 자바스크립트는 클라이언트 스크립트 언어이다.

자바스크립트는 서버(서비스를 제공하는 컴퓨터)에서 실행되는 것이 아니라, 사용자(방문자) 컴퓨터에서 실행된다. 그래서 서버의 부하를 줄일 수 있다.

3) 객체 기반 언어이다.

자바스크립트는 객체를 기반으로 한 언어이다. 따라서 다양한 객체가 존재하며, 그에 해당하는 다양한 기능(메서드)들이 존재한다.

4) 공개된 언어이다.

웹문서에 완성된 스크립트는 외부로 분리할 수는 있으나 완벽히 숨길수는 없다.

5) 다양한 라이브러리를 활용할 수 있다.

자바스크립트이 대표적인 라이브러리 언어는 제이쿼리이다.

라이브러리란 자바스크립트를 이용하여 다양한 기능들을 쉽게 구현할 수 있도록 한 함수들이 집합이다.

1.2 자바스크립트 기초문법

자바스크립트 선언문

```
<script type="text/javascript">  
자바스크립트 실행문  
</script>
```

```
<script>  
    var age=prompt("당신의 나이는?", "0");  
    if(age>=20){  
        document.write("당신은 성인입니다");  
    }else{  
        document.write("당신은 미성년자입니다");  
    }  
</script>
```

자바스크립트 언어의 특징

- 1) 자바스크립트는 대소문자를 가려서 쓴다.
- 2) 실행문을 마치고 나서는 세미콜론(;)을 쓰는 것이 좋다.
- 3) 실행문을 작성할때는 한 줄에 한 문장씩 작성하는 것이 가독성을 위해 좋다.
- 4) 문자형 데이터를 작성할때는 큰따옴표(“)와 작은 따옴표(‘) 겹침 오류를 주의한다.
- 5) 실행문을 작성할 때 중괄호 {} 또는 소괄호() 의 짝이 맞아야 한다.

1.3 변수

변수란?

변수(Variables)는 변하는 데이터(값)를 저장할 수 있는 메모리 공간이다.

데이터를 담을수 있는 그릇이라고 할수 있다. 변수에는 데이터가 오직 한 개만 저장된다. 그래서 새로운 데이터가 들어오면 기존에 있던 데이터는 메모리 공간에서 지워지게 된다.

```
var 변수명; 또는 var 변수명=값;
```

*문자형 데이터

```
var 변수명="사용할 문자나 숫자";
```

```
var s="javascript"; //변수s에 문자데이터("javascript")를 저장한다.
```

```
var num="100"; //변수 num에 문자데이터("100")을 저장한다.
```

```
var tag("<h1>String</h1>"); //변수tag에 문자데이터("<h1>...</h1>")를 저장한다.
```

*숫자형 데이터

```
var 변수명=숫자; 또는 Number("숫자");
```

```
var s=100;
```

```
var t=Number("500"); //문자형 데이터를 숫자형으로 변환하여 500을 반환한다.
```

*논리형 데이터 : 주로 2개의 데이터를 비교할 때 나오는 결과

```
var 변수명=true or false; 또는 Boolean(데이터);
```

```
var s=true; //변수 s에 데이터 true를 저장한다.
```

```
var t=10>=100; //변수 t에 10>=100의 결과 데이터 false를 저장한다.
```

```
var k=Boolean("hello"); //변수 k에 반환된 데이터 true를 저장한다.
```

-->Boolean() 메서드에 입력하는 데이터 중에 숫자 0과 null,undefined를 제외한 모든 데이터는 true를 반환한다.

변수를 선언할 때 주의사항!

- 1) 변수명 첫글자로는 \$,_(언더바),영문자만 올수 있다.
- 2) 변수명 첫글자 다음은 영문자,숫자,\$,_,숫자만 포함할 수 있다.
- 3) 변수명으로는 예약어를 사용할 수 없다.(document,locatuon>window 등)
- 4) 변수명을 지을때는 되도록 의미를 부여해 작성하는 것이 좋다.
- 5) 변수명을 사용할때는 대소문자를 구분해야 한다.

1.4 연산자

연산자 우선순위

- | | | | | | |
|-------|----------|----------|----------|----------|----------|
| 1. () | 2. 단항연산자 | 3. 산술연산자 | 4. 비교연산자 | 5. 논리연산자 | 6. 대입연산자 |
|-------|----------|----------|----------|----------|----------|

산술 연산자

종 류	기본형	설 명
+	A+B	더하기
-	A-B	빼기
×	A*B	곱하기
÷	A/B	나누기
%	A%B	나머지

문자 결합 연산자

여러개의 문자를 하나의 문자형 데이터로 결합할 때 사용한다.

문자형데이터 + 문자형데이터 = 하나의 문자형 데이터
문자형데이터 + 숫자형데이터 = 하나의 문자형 데이터

대입 연산자

연산된 데이터를 최종적으로 변수에 저장할 때 사용한다.

A=B	-> A=B
A+=B	-> A=A+B
A*=B	-> A=A*B
A/=B	-> A=A/B
A%=B	-> A=A%B

증감 연산자

변수--; 또는 --변수 //1 감소시킨다.

변수++; 또는 ++변수 //1 증가시킨다.

var A=++B; //변수B의 데이터를 1증가시킨 후 변수A에 저장한다.

var A=B++; //변수A에 변수B의 데이터를 저장한 후 변수B이 데이터를 1증가시킨다.

비교 연산자

종 류	기본형	설 명
A>B	A가 B보다 크다	
A<B	A가 B보다 작다	
A>=B	A가 B보다 크거나 같다	
A<=B	A가 B보다 작거나 같다	
A==B	A와 B는 같다	숫자를 비교할 경우 데이터형은 숫자이든 문자이든 상관없이 표기된 숫자만 일치하면 true를 반환한다. 예)숫자형10과 문자형"10"은 같다 ->true
A!=B	A와 B는 다르다	숫자를 비교할 경우 데이터형은 숫자이든 문자이든 상관없이 표기된 숫자만 다르면 false를 반환한다. 예)숫자형10과 문자형"10"은 같다 ->>false
A===B	A와 B는 같다	숫자를 비교할 경우 반드시 표기된 숫자와 데이터형도 일치해야만 true를 반환한다. 예)10과 "10" ->>false
A!==B	A와 B는 다르다	숫자를 비교할 경우 반드시 표기된 숫자와 데이터형이 다를 때 true를 반환한다. 예)10과 "10" ->true

☑ 논리 연산자

|| or연산자, 피연산자중 값이 하나라도 true가 존재하면 true결과값을 반환
&& and연산자, 피연산자중 값이 하나라도 false가 존재하면 false결과값을 반환
! not연산자, 단항연산자, 피연산자의 값이 true면 반대로 false결과값을 반환

☑ 삼항 조건 연산자

조건식 ? 실행문1 : 실행문2;

조건식의 값이 true면 실행문1을 실행하고, false면 실행문2를 실행하게 된다.

Chapter 2. 제어문

제어문은 프로그램의 흐름을 제어할수 있도록 도와주는 실행문을 말한다.

제어문에는 조건만족 여부에 따라 실행문을 제어할수 있는 **조건문**과

변수에 일치하는 경우의 값에 따라 실행문을 제어할수 있는 **선택문**,

특정실행문을 여러번 반복 실행할수 있도록 하는 **반복문**이 있다.

2.1 조건문(if)


조건문은 조건식의 값이 참(true)인지 아니면 거짓(false)인지에 따라 실행문의 제어가 결정된다.

☑ **if 문** : 조건식을 만족할 경우에만 실행문을 실행함


```
if(조건식){  
    실행문;  
}
```

※ 다음값이 조건식에 입력되면 false 반환하지만, 그 밖에 모든 값은 true로 인식된다.


0, null, ""(빈문자), undefined

 **else 문** : 조건식을 만족할 경우와 만족하지 않았을 경우에 따라 실행문이 다름

```
if(조건식){
    실행문1;
}else{
    실행문2;
}
```

 **else if 문** : 두가지이상의 조건식과 정해놓은 조건식을 만족하지 않았을 때 실행

```
if(조건식1){
    실행문1;
}else if(조건식2){
    실행문2;
}else if(조건식3){
    실행문3;
}else if(조건식4){
    실행문4;
}else if(조건식5){
    실행문5;
}else{
    실행문6;
}
```

 **중첩 if 문** : 조건문 안에 조건문이 오는 것을 말함

```
if(조건식1){
    if(조건식2){
        실행문;
    }
}
```


2.2 선택문(switch)

변수에 지정된 값과 switch문에 있는 경우(case)의 값을 검사하여 변수와 경우의 값에서 일치하는 값이 있을 때 그에 해당하는 실행문을 실행한다. if문과 용도는 비슷하나 if문은 만족하는 데이터가 여러개일 경우 주로 사용하고, switch문은 여러 경우의 값중 일치하는 데이터를 찾아 그에 해당하는 실행문을 실행시킬 때 사용한다.

```
var 변수=초기값;
switch(변수){
    case 값1:실행문1;
    break;
    case 값2:실행문2;
    break;
    case 값3:실행문3;
    break;
    case 값4:실행문4;
    break;

    default:실행문6;
}
```

2.3 반복문(for)

반복문을 이용하면 실행문을 원하는 횟수만큼 반복하여 실행시킬수 있다.

while 문

```
var 변수=초기값;
while(조건식){
    실행문;
    증감씩;
}
```

do while 문

```
var 변수=초기값;
do{
    실행문;
    증감식;
}while(조건식)
```

for 문

```
for(초기값;조건식;증감식){
    실행문;
}
```

break 문 : 반복문을 강제로 끝낼 때 사용

```
for(초기값;조건식;증감식){
    break; //반복문을 강제로 종료한다.
    실행문;
}

var 변수=초기값;
while(조건식){
    break; //반복문을 강제로 종료한다.
    실행문;
    증감식;
}
```



continue 문 : 반복문에서만 사용, 실행문은 무시하고 바로 조건식으로 이동해서 실행

```
for(초기값;조건식;증감식){  
    continue;    //다음에 오는 실행문은 무시하고 바로 증감식으로 이동한다.  
    실행문;  
}  
  
var 변수=초기값;  
while(조건식){  
    증감식;  
    continue;    //다음에 오는 실행문은 무시하고 바로 조건식으로 이동한다.  
    실행문;  
}
```

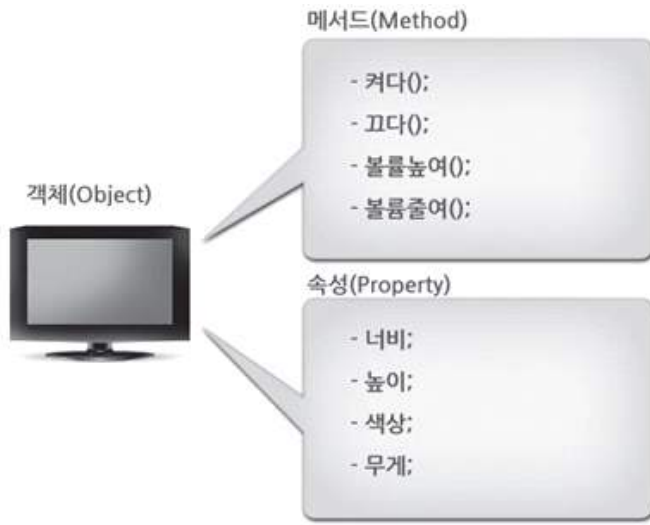


중첩 for 문

```
for(초기값;조건식;증감식){    //바깥쪽 for문  
    for(초기값;조건식;증감식){    //안쪽 for문  
        실행문;  
    }  
}
```

Chapter 3. 객체

자바스크립트는 객체(Object) 기반 언어이다. 객체는 기능 또는 속성으로 구성되어 있다.



```
TV.켜다( );  
TV.끄다( );  
  
TV.너비 = "30in"  
TV.색상 = "black"
```

✅ 다음은 자바스크립트 객체에 메서드와 속성 사용의 기본형

```
객체(Object). 메서드();    //객체에 구성된 기능을 사용할 때.  
객체(Object). 속성;    또는   객체(Object). 속성 = 값;
```

✅ 객체 생성법

객체를 생성하는 기본 형식은 다음과 같이 new 키워드와 생성함수를 이용한다.
이렇게 생성한 객체는 인스턴스네임(참조변수)으로 참조시켜 사용한다.

```
인스턴스네임(참조 변수)=new 생성 함수();  
var tv=new Object( ); //객체를 생성합니다.
```

내장객체란 브라우저의 자바스크립트 엔진에 내장된 객체를 말하며, 필요한 경우 객체를 생성해 사용할 수 있다. 내장객체로는 문자(String), 날짜(Date), 배열(Array), 수학(Math), 정규표현객체(RegExp Object) 등이 있다.



날짜 정보 객체

날짜 객체는 현재 날짜 또는 특정 날짜 관련 다양한 정보를 제공한다.

가령, 2020. 10. 20이 무슨 요일인지 정보를 알 수 있다.

```
var 인스턴스네임(참조변수) = new Date( ); //현재 날짜 객체 생성  
var 인스턴스네임(참조변수) = new Date( 년,월,일); //특정 날짜의 객체를 생성한다.
```

- 현재 날짜 객체를 생성

```
var t1=new Date();
```

- 현재 요일 값을 구해옵니다.

```
t1.getDay( );
```

- 월드컵(2002.05.31) 객체를 생성한다.

```
var t2=new Date(2002.4.31);
```

- 월드컵 요일 값을 구해옵니다.

```
t2.getDay( );
```

수학 객체

앞서 수를 더하거나 빼는 작업은 산술 연산자를 사용하였다. 하지만 최대 값, 반올림 등 수학 관련 기능을 사용하기 위해서는 수학객체를 사용해야 한다.

```
Math.메서드( );
```

- 최대 값을 구합니다.

```
var num=Math.max(1, 25, 10);
```

- 반올림 값을 구합니다.

```
var num=Math.round(2.4); //2
```

배열 객체

앞서 변수에는 한 개의 데이터만 저장할 수 있고, 만일 변수에 새 값이 저장되면 기존에 저장된 값은 제거되었다. 하지만 이번에 배열 객체를 이용하면 여러 개의 데이터를 한 변수에 저장할 수 있다. 배열 객체를 생성하고, 데이터를 저장하는 방법은 다음과 같다.

```
→ 참조변수 = new Array 생성 함수(값1, 값2, 값3);
```

```
→ 참조변수 = new Array 생성 함수( );
```

```
    참조변수[0]=값1;    참조변수[1]=값2;    참조변수[2]=값3;
```

```
→ 참조변수 = [값1, 값2, 값3];
```

*배열에 저장되어 있는 값을 가져오는 방법은 다음과 같습니다.

```
→ 참조변수[ 인덱스 값]
```

- 배열 객체를 생성

```
var arrData= [ 1, true, “값” ];
```

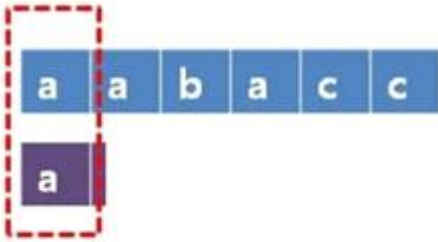
- 배열에 인덱스 1에 저장된 값을 불러옵니다.

```
var t=arrData[ 1 ]; // true
```

☑ 문자 객체

문자 객체는 지정한 문자열의 다양한 정보를 제공하는 메서드(기능)와 속성으로 구성

```
var 인스턴스네임(참조변수) = new 생성함수( "문자열" )
```



```
var t=new String("Good Web Site") or var t="Good Web Site";
```

문자열에 “W”에 인덱스 값 5를 반환한다.

```
t.indexOf("W");
```

문자열에 문자 총 개수 13을 반환한다.

```
t.length;
```

3.2 브라우저 객체 모델(BOM)

브라우저에 내장된 객체를 “브라우저객체”라고 한다. window는 브라우저 객체의 최상위 객체가 된다. window객체에는 하위객체를 포함하고 있고, 계층적 구조로 이루어져있다.

window 객체

window객체는 브라우저 객체의 최상위 객체이며 다음 메서드를 내포하고 있다.

`open(), alert(), prompt(), confirm(), moveTo(), resizeTo()
setInterval(), clearInterval(), setTimeout(), clearTimeout()`

■ window 객체 메서드 종류

종 류	설 명
open()	새창을 열 때 사용한다.
alert()	경고창을 띄운다.
prompt()	질의응답창을 띄운다.
confirm()	확인/취소 창을 띄운다.
moveTo()	창의 위치를 이동시킬 때 사용한다.
resizeTo()	창의 크기를 변경시킬 때 사용한다.
setInterval()	일정간격으로 지속적으로 실행문을 실행시킬 때 사용한다.
setTimeout()	일정간격으로 한번만 실행문을 실행시킬 때 사용한다.

screen

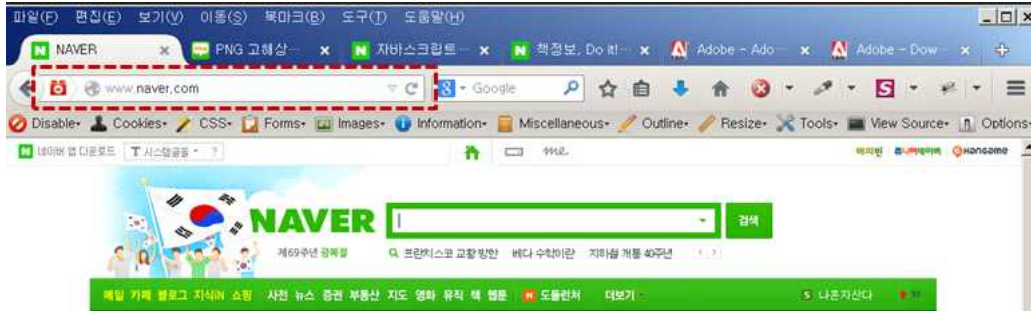
screen객체는 스크린(모니터) 너비, 높이 등 정보(속성)를 제공한다.

→ 스크린에 너비 속성 값을 반환한다.

screen.width

✓ location

location객체는 브라우저에 url 관련 정보(속성) 및 메서드를 제공한다.



→ 브라우저에 URL에 주소를 가져온다.

location.href

✓ history

history객체는 브라우저에 방문 기록에 대한 정보(속성) 및 메서드를 제공합니다. 마치 브라우저에 이전, 다음버튼과 같은 기능을 제공한다.



→ 브라우저에 이전에 방문한 사이트로 이동한다.

history.back();

✓ navigator

navigator객체는 방문자의 브라우저 정보와 운영체제 정보를 제공한다.

→ 방문자의 브라우저와 운영체제 정보를 제공합니다.

navigator.userAgent;

3.3 문서 객체 모델(DOM)

문서객체모델이란 HTML 문서의 구조를 가리킨다. 태그에는 src라는 속성과 그림을 표현하는 기능을 가지고 있다. 그러므로 HTML 태그는 다른 말로 문서객체라 부른다.

선택자

선택자는 자바스크립트를 이용해 HTML에 문서 객체를 선택할 때 사용한다.
스타일(CSS)에 선택자와 비슷하다고 생각하면 된다.

CSS 선택자 = javascript 선택자

→ CSS: #box{ color:red; }

→ javascript: document.getElementById("box").style.color=red;

CSS -> #box + p { color:red; }

javascript -> document.getElementById("box").nextSibling.style.color=red;

Chapter 4. 함수

4.1 함수란?

함수를 사용하면 일련의 실행문을 메모리에 저장했다가 필요할때마다 호출하여 사용할수 있다.

기본함수 정의문

```
function 함수명(){  
    실행문;  
}  
  
예)  
function test( ){  
    document.write("hello")  
}  
test ( );
```

```
참조변수 = function(){  
    실행문;  
}
```

매개변수가 있는 함수

매개 변수란 함수를 호출하였을 때 전달한 값을 저장하는 변수를 가리킨다. 이렇게 전달된 값은 함수에 일련에 실행문에 사용할 수 있다.

```
function 함수명(매개변수1, 매개변수2, ... 매개변수n ){ //함수 정의문  
    일련의 실행문;  
}  
함수명 (값1, 값2, ... 값n); //함수 호출문  
  
function test(num1, num2){  
    document.write(num1 + num2 )  
}  
test ( 100, 300 );
```

☑ 내장함수의 종류

내장함수는 자바스크립트 엔진에 내장된 함수정의문을 말한다.

내장함수는 함수 정의문의 선언없이 단지 함수 호출만으로 사용할 수 있다.

종 류	기본형	설 명
parseInt()	문자형 데이터를 정수형 데이터로 바꾼다.	parseInt("5.12") -> 5
parseFloat()	문자형 데이터를 실수형 데이터로 바꾼다.	parseFloat("5.12") -> 5.12
String()	문자형 데이터로 바꾼다.	String(5) -> "5"
Number()	숫자형 데이터로 바꾼다.	Number("5") -> 5
Boolean()	논리형 데이터로 바꾼다.	Boolean(5) -> true
isNaN()	데이터에 숫자가 아닌 문자를 포함하여 true를 반환한다.	isNaN("5-3") -> true isNaN("53") -> false
eval()	문자형 데이터를 큰따옴표가 없는 스크립트코드로 처리한다.	eval("15+ 5") -> 20

4.2 return 문

return문이란 함수에서 결과값을 되돌려 줄 때 사용된다.

☑ 데이터를 반환하는 return 문

```
function 함수명(){  
    실행문;  
    return 데이터(값); (2)  
}  
var 변수=함수명(); //함수호출(1)
```

->(1)함수호출 -> (2)실행후 "데이터" ->return

강제 종료 역할을 하는 return 문

```
function 함수명(){  
    실행문; (1)  
    return;  
    실행문; (2)  
}  
함수명();
```

->함수에 실행문을 실행하다가, return 문을 만나면 다음에 오는 실행문들은 무시하고 강제로 종료된다.

4.3 지역변수와 전역변수

전역과 지역 변수를 나누는 이유는 변수의 중복 사용을 피하기 위해서이다. 즉, 함수정의문 내에서 var를 붙여 지역변수를 이용하게 되면 여러 함수에 같은 변수가 있어도 중복될 일이 없다. ->한 개의 변수를 여러 함수에 공용해 사용하고 싶다면 전역변수를 사용해야 한다.

전역 변수

전역변수란 함수정의문 내에서 var를 붙이지 않는 변수이며, 이 변수에 저장된 데이터는 자바스크립트가 선언된 곳이라면 어디든 사용할수 있다.

```
var 변수;  
function 함수명(){  
    변수=값;  
}
```

지역 변수

지역변수는 함수 정의문 내에서 var를 붙인 변수로, 이 변수에 오는 데이터는 함수 정의문 내에서만 사용할수 있다. 즉, 이 변수에 저장된 데이터는 함수 내에서만 불러올수 있다.

```
function 함수명(){  
    var 변수;  
}
```

Chapter 5. 이벤트

5.1 이벤트란?

이벤트란 방문객이 취하는 모든 동작을 이벤트라 한다.

이벤트 종류

종 류		설 명
마우스 이벤트	onmouseover	마우스가 지정한 요소에 올라갔을 때 발생한다.
	onmouseout	마우스가 지정한 요소에 벗어났을 때 발생한다.
	onmousemove	마우스가 지정한 요소 영역에서 움직일때 발생한다.
	onclick	마우스가 지정한 요소를 클릭했을때 발생한다.
	ondblclick	마우스가 지정한 요소를 연속두번 클릭했을때 발생한다.
키보드 이벤트	onkeypress	지정한 요소에서 키보드가 눌렸을 때 발생한다.
	onkeydown	지정한 요소에서 키보드를 눌렸을 때 발생한다.
	onkeyup	지정한 요소에서 키보드를 눌렀다 떼었을때 발생한다.
기타 이벤트	onfocus	지정한 요소에 포커스가 갔을 때 발생한다.
	onblur	지정한 요소에 포커스가 다른 요소로 이동되어 잃었을 때 발생한다.
	onchange	지정한 요소에 value속성값이 바뀌고 포커스가 이동될 때 발생한다.
	onload	지정한 요소의 하위요소를 모두 로딩했을 때 발생한다.
	onunload	문서를 닫거나 다른 문서로 이동했을 때 발생한다.
	onsubmit	폼요소에 전송버튼을 눌렀을 때 발생한다.
	onreset	폼요소에 취소버튼을 눌렀을 때 발생한다.
	onresize	지정된 요소의 크기가 변경되었을 때 발생한다.
	onerror	문서객체가 로드되는 동안 문제가 발생되었을 때 발생한다.

키보드 접근성

마우스가 없다고 가정하면 onmouseover 이벤트를 사용할수 없다.

요소에 마우스 이벤트를 등록할 경우에는 마우스가 없어도 접근(작동)할수 있도록 해야 하는데 이것을 키보드 접근성이라 한다. 다음과 같이 마우스 이벤트가 등록되었을때는 반드시 키보드로 작동할수 있게 대응 이벤트를 함께 작성해야 한다.

마우스 이벤트	키보드 대응 이벤트
onmouseover	onfocus
onmouseout	onblur

JQuery

Chapter 1. jQuery 사용하기

Chapter 2. jQuery 효과

2.1 jQuery 이벤트

2.2 jQuery 애니메이션효과

2.3 탄력적인 애니메이션

Chapter 3. jQuery 기본문법

3.1 조건문(IF)

3.2 반복문(FOR) • 151

3.3 사용자정의함수 • 152

3.4 jQuery 셀렉터 • 153

3.5 HTML/CSS를 조작하는 jQuery명령어 • 155

Chapter 1. jQuery 사용하기

jQuery 구동파일 다운받기

<http://jquery.com>

위의 공식사이트에서 "Download(jQuery)" 를 클릭하여 다운로드한다.
2번째 파일 1.11.1....."minified"방식으로 압축되어 있다.

■ 다운로드한 jQuery를 웹서버에 업로드하고 jQuery를 읽어들일 html 페이지의
<head>~</head>사이에 삽입한다.

```
<script src="./jquery-1.11.3.min.js"></script>
```

```
<script src="./script.js"></script>
```

```
/* jQuery로 처리할 내용을 기술한 자바스크립트파일을 불러냄*/
```

■ <head> ~ </head> 또는 </body> 바로 전에 <script>~</script>문으로 선언

■ - 단일주석문 : //설명글 - 다중주석문 : /*설명글*/

■ jQuery 로 스크립트를 작성할때는 대부분 우선 ready함수를 작성하고,
그 안에 실행될 코드를 구현함.

```
$(document).ready(function(){    → $(function(){    (생략시)
    $("선택터").jQuery 의 명령
});
```

예)

```
<script>
```

```
$(document).ready(function(){
```

```
    $("li").css("color","red");
```

```
/* li 태그에 있는것을 css명령인 색상을 빨강으로 만들어라 */
```

```
});
```

```
</script>
```

=>여기서 위의 script 태그를 head 안에 기술할 경우 브라우저는 페이지로딩이 아직 끝나지 않았기 때문에 button 존재를 발견할수 없다.

아래와 같이 스크립트 전체를 \$(function(){....}) 안에 기술해야 한다.

Chapter 2. jQuery 효과

2.1 jQuery 이벤트

jQuery 이벤트 정리

```
$(선택터).이벤트(function(){
    $(선택터).명령
});
```

예)

```
<script type="text/javascript">
    $(document).ready(function(){
        $("button").click(function(){
            button 태그가 클릭되었을때 실행하는 처리
        });
    });
</script>
```

자주 사용되는 이벤트

- 클릭 : **click**
- 마우스를 올리면 : **mouseenter**
- 마우스를 올렸다떼면 : **mouseleave**
- 마우스 올리고 떼면 : **hover** (mouseenter + mouseleave)

예)

```
$("선택자").hover(function(){
    마우스를 올렸을 때의 명령문;
}, function(){
    마우스를 뗄 때의 명령문;
});
```

2.2 jQuery 애니메이션

애니메이션효과와 관련된 명령정리

■ 자주 사용되는 함수

- 숨김 / 표시 : `show() / hide() / toggle()`
- 점점사라짐 / 점점나타남 : `fadeIn() / fadeOut() / fadeToggle()`
- 아래로펼쳐짐 / 위로접힘 : `slideDown() / slideUp() / slideToggle()`
- 특정 속성의 애니메이션 : `animate ({ "속성" : "값" }, [시간])`
- css 수정 : `css ({ "속성" : "값", "속성" : "값"..... })`

명 령	의 미	내 용
<code>show(...)</code>	태그를 서서히 표시한다.	
<code>hide(...)</code>	태그를 서서히 비표시한다.(숨김)	
<code>toggle(...)</code>	태그의 표시/비표시를 반복 변경하면서 서서히 표시한다.	
<code>slideDown(...)</code>	태그를 슬라이딩 애니메이션 효과로 표시한다.	
<code>slideUp(...)</code>	태그를 슬라이딩 애니메이션 효과로 비표시한다.	
<code>slideToggle(...)</code>	태그를 슬라이딩 애니메이션 효과로 표시/비표시를 반복 변경한다.	
<code>fadeIn(...)</code>	태그를 페이드인 효과로 표시한다.	
<code>fadeOut(...)</code>	태그를 페이드아웃 효과로 표시한다.	
<code>fadeTo(...)</code>	태그의 투명도를 지정한 값으로 서서히 변경한다.	투명도는 0~1(0.5->50%)
<code>animate()</code>	임의의 CSS속성값을 서서히 변경한다.	

모든 코드 호출 후 명령문 수행

```
$(document).ready(function(){
    명령문;
    명령문;
    .....
});
```

시간제어

- 대 부분의 언어에서 '밀리초(1/1000)' 단위로 계산됨
예) show(2000) : 2초동안 나타나는 애니메이션
기본값 : 400

CSS 제어

```
$("#선택자").css( {"속성" : "값", "속성" : "값".....} );
```

예)

```
$("#header").css( {"margin" : "10px 20px", "color" : "#000"} );
```

```
$("#header").css( {
    "margin" : "10px 20px",
    "color" : "#000"
} );
```

// 아이디가 헤더인 선택자에 마진과 글자색 변경

this : 이벤트의 실질적인 주제를 가리킨다.

예)

```
$( "div ul li" ).click(function(){
    $(this).hide(); // this는 li를 가리킨다
});
```

✓ 애니메이션 : animate

```
$("#선택자").css( {"속성" : "값", "속성" : "값".....} );
```

예)

```
$("#div").animate( { "margin-left" : "-20px" }, 1200 );  
// div 요소는 왼쪽으로 20px 이동되는 애니메이션이 수행됨
```

✓ 시간차 명령문 수행

```
명령문1( );  
명령문2( );  
// 마치 동시에 진행된 것 처럼 보임  
명령문1( function( ) {  
    명령문2( );  
});  
// 명령문1이 완료된 후 명령문2가 수행됨
```

✓ 클래스 추가제거

```
- 클래스 추가 : addClass( )  
    ex) $("#선택자").addClass("클래스명");  
- 클래스 제거 : removeClass( )  
    ex) $("#선택자").removeClass("클래스명");
```

- 제이쿼리에서 직접 제어하지 않고 css 파일에 미리 선언해둔다.
(단, 우선순위를 고려한다 ★★)

예)

```
$("#ul").addClass("clear");  
//모든 ul에 clear 클래스 추가
```

예)

```
<p class="memo active">.....</p>  
$("#p.memo").removeClass("active");  
//memo라는 클래스를 갖고 있는 p요소에서 active 클래스를 제거
```

선택자 응용(찾기)

- 부모요소 : parent()
- 자식요소 : find()
- 형제요소
 - 이전 : prev() 다음 : next() 순서 : eq() 제외 : not()

사용자의 중복반응 제거

- 1) 문제제기 : 사용자 마우스의 중복(빠른) 반응 때문에 애니메이션 겹침현상이 생긴다
- 2) 해결방법 : stop() -> stop(효과누적, 진행과정)
 - 기본값 : stop() == stop(false, false)
 - 변경해야될 값 : stop(true, true)

예)

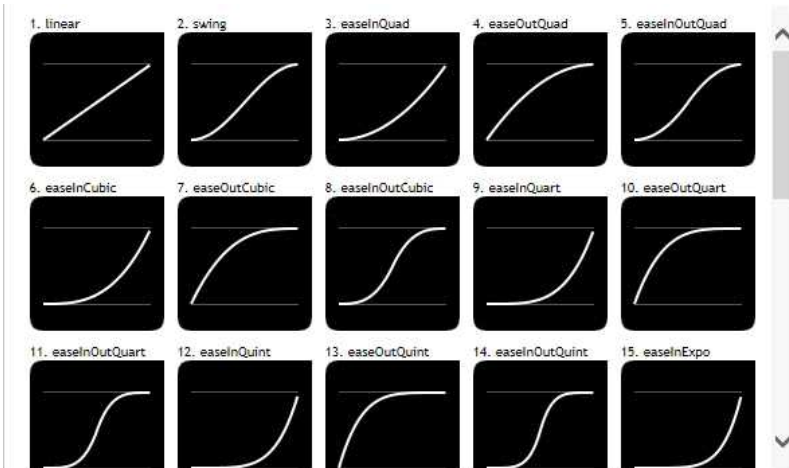
```
$(선택자).stop(true, true).animate({.....});  
$(선택자).stop(true, true).slideDown( );
```

.....

// 중간과정이 발생하는 애니메이션 효과의 모든 곳에 적용

2.3 탄력적인 애니메이션

✓ 탄력적인 애니메이션



- 1) 기존 애니메이션의 단점 : 움직임(변화)의 평균속도가 모두 같아 동적인 느낌이 적다
- 2) jqueryui.com에서 제이쿼리 UI 파일을 다운로드 받아 웹문서에 삽입(script)
- 3) 제이쿼리 UI의 장점
 - 기존 애니메이션으로 불가능한 배경색, 글자색, 패딩, 보더 등 대부분의 css 애니메이션 지원
 - 탄력적인 움직임 가능easeOutExpo / easeOutBack / easeOutElastic 등

4) 적용방법
`$(선택자).애니메이션({.....}, 시간, "easeOutExpo");`

기존 애니메이션을 탄력있는 (바운딩, 증감속도 등) 애니메이션(fade, slide, animate 등)으로 확장할 수 있습니다.

관련링크
<http://jqueryui.com/>
<http://jqueryui.com/download/>

■ 상단 링크(또는 첨부된 파일)를 다운로드 받은 후 아래와 같이 기본 제이쿼리 라이브러리 코드 아래에 ui 파일을 삽입합니다.

```
<script src="../js/jquery-1.8.3.min.js"></script>
<script src="../js/jquery-ui-1.10.4.custom.min.js"></script>
```

■ 사용방법

```
$(selector).slideDown(500, "easeOutExpo");
$(selector).animate({"height":"400px"}, 500, "easeOutExpo");
```

■ 탄력적인 움직임의 종류

jQuery 사이트에서 시각적인 형태로 확인가능

<http://jqueryui.com/effect/#easing>

<http://easings.net/>

easeInQuad / easeOutQuad / easeInOutQuad
easeInCubic / easeOutCubic / easeInOutCubic
easeOutCubic / easeInOutCubic / easeInQuart
easeOutQuart / easeInOutQuart / easeInQuint
easeOutQuint / easeInOutQuint / easeInSine
easeOutSine / easeInOutSine
easeInExpo / easeOutExpo / easeInOutExpo ::추천해요!
easeInCirc / easeOutCirc / easeInOutCirc
easeInElastic / easeOutElastic / easeInOutElastic
easeInBack / easeOutBack / easeInOutBack ::추천해요!
easeInBounce / easeOutBounce / easeInOutBounce ::통통튀는효과

자동화 : setInterval()

- 1) 의미 : 특정 사용자정의 함수를 원하는 시간마다 호출해서 실행
- 2) 작성형식

```
setInterval("함수이름()", 시간);
```

예)

```
setInterval("test()", 3000);  
//test() 함수를 3초에 한 번씩 수행하시오
```

```
# 자동화 해제 : clearInterval( )  
clearInterval(해제시킬 setInterval 값);
```

예)

```
name = setInterval("test()", 3000);  
clearInterval(name);  
  
# 특정 요소의 콘텐츠(내용) 삽입 / 수정  
  
$(선택자).text("삽입 또는 수정 할 내용....");
```

특정값을 검증하는 방법

01. 경고창 띄우기 : alert(값);
02. 콘솔창에서 확인 : console.log(값);

결합연산자 : +

01. 숫자 + 숫자 = 숫자
02. 숫자 + 문자 = 문자
03. 문자 + 문자 = 문자

Chapter 3. jQuery 기본문법

3.1 조건문(if)

특정 조건식에 대한 참, 거짓을 판단하여 명령문을 수행

■ 단일조건문

```
if (조건식) {  
    참일 때 수행할 명령문;  
}  
else {  
    거짓일 때 수행할 명령문;  
}
```

■ 다중조건문 (결과값이 3가지 이상인 경우)

```
if (조건식A) {  
    조건식 A가 참일 때 수행할 명령문;  
}  
else if (조건식B) {  
    조건식 A가 거짓이고 조건식 B가 참일 때 수행할 명령문;  
}  
.....  
else {  
    조건식 A, B가 모두 거짓일 때 수행할 명령문;  
}
```

비교연산자

크다(초과)	>	작다(미만)	<
크거나같다(이상)	>=	작거나같다(이하)	<=
같다	==	다르다 !=	(부정 !)
대입연산자	=		

- 변수 : 변화될 값을 보관하는 모든 대상
- 변수선언 : 영어소문자, 언더바(_), 숫자 **//단, 숫자로 시작 불가능**
예)

count = 10; //개수를 세는 변수 선언

cur_height = 300; **//현재 높이값을 기억하는 변수 선언**

3.2 반복문(for)

특정 요소의 명령문 제어시 반복 상황이 발생하는 것을 순환구조로 만들어 간단하게 처리되도록 하는 구조

```
for(초기값; 조건식; 증감식){
    반복하여 수행할 명령문;
}
```

예)

```
for( i=0; i<10; i++ ){ // i = i + 1 (== i++ ) i값이 1씩 증가
    $(li).eq(i).hide( );
}
```

태그의 속성제어

- 형식 (css, animate와 동일)

```
$(선택자).attr( { "속성명" : "값" } );
```

예)

```
$("li a img").attr( {"src" : "../images/img_over.png"} );
$("p input").attr( {"type" : "button"} );
```

순서(배열) 반환 : index()

- 형식 : \$(선택자).index();

예)

```
num = $("ul li").index( );
alert(num); // li의 개수만큼 번호가 추출이 됨
```

3.3 사용자정의함수

1) 의미 : 여러번 반복되는 명령문(비슷한 형식)을 수행할 경우 선언 및 호출

2) 선언

```
function 함수이름( ) {  
    반복될 명령문;  
}
```

- 주의 할 점 : 함수를 선언한 것은 "생성"만 했을 뿐 수행되지 않음 ★★

3) 호출

선언된 함수를 호출하는 것으로 수행하고 싶은 위치에서 함수이름을 적어주면 됨

예)

```
//선언  
function test( ){  
    a = 10;  
    b = 20;  
}  
//호출  
test( );
```

3.4 jQuery 선택터

명칭	서식	지정대상	사용예
★ CSS에서 자주 사용되는 선택터			
태그선택터	\$("태그명")	특정태그	\$("li").css("color","red");
ID선택터	\$("#ID명")	특정ID를 속성으로 갖는태그	\$("#first").css("color","red");
클래스선택터	\$(".클래스명")	특정클래스를 속성으로 갖는태그	\$(".second").css("color","red");
자손선택터	\$("태그1 태그2")	특정태그의 안쪽에 있는 태그	\$(".first strong").css("color","red");
유니버설선택터	\$("*")	전체태그	\$("li *").css("color","red");
그룹선택터	\$("선택터1, 선택터2")	복수의 선택터	\$("#first , #third").css("color","red");
※ CSS2 선택터			
자식선택터	\$("부모태그명>자식태그명")	특정태그의 바로 밑에 자식태그	\$("li > strong").css("color","red");
인접선택터	\$("태그1+ 태그2")	특정태그의 다음 태그	\$("#second + li").css("color","red");
first-child 유사클래스	\$("태그:first-child")	동일태그안의 첫 태그	\$("li:first-child").css("color","red");
※ CSS3 선택터			
간접선택터	\$("태그1~태그2")	특정태그의 뒤에 나타나는 태그	\$("#second ~ li").css("color","red");
부정유사클래스	\$("태그1:not(태그2)")	특정태그 안에서 태그2를 제외한 태그	\$("li:not(:first-child)").css("color","red");
empty 클래스	\$("태그:empty")	자식태그or문자열을 포함하지않는 태그	\$("li:empty").css("color","red");
nth-child 유사클래스	\$("태그:nth-child(번호)")	특정태그 안에서 지정한 번호의 태그	\$("li:nth-child(3)").css("color","red");
last-child 유사클래스	\$("태그:last-child")	동일태그안의 가장 마지막태그	\$("li:last-child").css("color","red");
only-child 유사클래스	\$("태그:only-child")	특정태그가 하나만 포함된 태그	\$("li span:only-child").css("color","red");

※ CSS의 속성선택터			
[attribute]\$("[속성명"])		특정속성을 가진 태그	\$("[id]").css("color","red");
[attribute='vlaue']	\$("[속성명='값']")	특정속성이 지정된 값을 가진태그	\$("[title='second']").css("color","red");
[attribute!='vlaue']	\$("[속성명!='값']")	특정속성이 지정된 값을 갖지않은 태그	\$("li[title!='first']").css("color","red");
[attribute^='vlaue']	\$("[속성명^='값']")	특정속성이 지정된 값으로 시작하는 태그	\$("[title^='f']").css("color","red");
[attribute\$='vlaue']	\$("[속성명\$='값']")	특정속성이 지정된 값으로 끝나는 태그	\$("[title\$='d']").css("color","red");
[attribute*='vlaue']	\$("[속성명*='값']")	특정속성이 지정된 값을 포함하는 태그	\$("[title*='ir']").css("color","red");
[attributeFilter1] attributeFilter2]	\$("[태그선택터1] [태그선택터2]")	복수속성 선택터에 해당하는 태그	\$("[title^='f'] [title*='th']").css("color","red");
※ jQuery 독자필터			
first 필터	\$("태그:first")	지정한 태그의 첫 태그	\$("li:first").css("color","red");
last 필터	\$("태그:last")	지정한 태그의 마지막태그	\$("li:last").css("color","red");
even 필터	\$("태그:even")	지정한 태그의 짝수번째 태그	\$("li:even").css("color","blue");
odd 필터	\$("태그:odd")	지정한 태그의 홀수번째 태그	\$("li:odd").css("color","red");
eq 필터	\$("태그:eq(번호)")	지정한 번호의 태그(0부터시작)	\$("li:eq(2)").css("color","blue");
gt 필터	\$("태그:gt(번호)")	지정한 번호보다 뒤의 태그	\$("li:gt(2)").css("color","green");
lt 필터	\$("태그:lt(번호)")	지정한 번호보다 앞의 태그	\$("li:lt(2)").css("color","red");
header 필터	\$("태그:header")	h 1 ~ h 6 까 지 의 heading 태그	\$(":header").css("color","red");
contain 필터	\$("태그:contain(문자열)")	특정문자열을 포함하는 태그	\$("li:contains('샘플)').css("color","red");
has 필터	\$("태그1:has(태그2)")	특정태그가 포함하고 있는 태그	\$("li:has(strong)").css("color","red");
parent 필터	\$("태그:parent")	자식태그or문자열을 포함하고 있는 태그	\$("li:parent").css("color","red");

3.5 HTML/CSS를 조작하는 jQuery 명령어

명칭	의미	사용예
※ 텍스트 변경과 가져오기		
text(...)	텍스트를 변경한다.	<code>\$("p#first").text("변경 후");</code>
text()	텍스트를 가져온다.	<code>\$("p#second").text(\$("p#first").text());</code>
※ HTML 변경과 취득		
html(...)	HTML을 변경한다.	<code>\$("p#first").html(" 변경 후");</code>
html()	HTML을 가져온다	<code>\$("p#second").html(\$("p#first").html());</code>
※ HTML 삽입(기존에 존재하던 태그의 내용을 남긴채...)		
prepend(...)	태그안의 맨앞에 HTML을 삽입한다.	<code>\$("p#first").prepend(" 앞부분에 삽입");</code>
append(...)	태그안의 맨뒤에 HTML을 삽입한다.	<code>\$("p#first").append(" 뒷부분에 삽입");</code>
before(...)	태그 앞에 HTML을 삽입한다.	<code>\$("p#first").before("<h1> 태그 앞에 삽입</h1>");</code>
after(...)	태그 뒤에 HTML을 삽입한다.	<code>\$("p#first").after("<h1> 태그 뒤에 삽입</h1>");</code>
※ HTML 의 이동		
prependTo(...)	다른 태그 안의 맨앞으로 이동한다.	<code>\$("strong").prependTo("p");</code>
appendTo(...)	다른 태그 안의 맨뒤로 이동한다.	<code>\$("strong").appendTo("p");</code>
insertBefore(...)	다른 태그의 앞으로 이동한다.	<code>\$("h1").insertBefore("p");</code>
insertAfter(...)	다른 태그의 뒤로 이동한다	<code>\$("h1").insertAfter("p");</code>
※ 다른 태그로 묶음		
wrap(...)	태그를 다른 태그로 묶는다.	<code>\$("strong").wrap("<h1></h1>");</code>
wrapAll(...)	태그 전체를 모아서 다른 태그로 묶는다.	<code>\$("strong").wrapAll("<h1></h1>");</code>
wrapInner(...)	자식 태그/텍스트를 다른 태그로 묶는다.	<code>\$("p").wrapInner("");</code>

※ 태그 변경		
replaceWidth()	태그를 다른 태그로 변경한다.	<code>\$("p").replaceWith("<h1> 변경 후</h1>");</code>
※ 태그 제거		
remove()	태그를 제거한다.	<code>\$("p strong").remove();</code>
※ 속성값 변경과 취득		
<code>attr("속성명","속성값")</code>	지정한 속성값을 변경한다.	<code>\$("a").attr("href","http://ascii.jp/");</code>
<code>attr(...)</code>	지정한 속성값을 가져온다.	<code>\$("a").text(\$("a").attr("href"));</code>
<code>removeAttr(...)</code>	지정한 속성을 제거한다.	<code>\$("a").removeAttr("target");</code>
※ class 속성 추가와 제거		
<code>addClass(...)</code>	class 속성을 추가한다.	<code>\$("p").addClass("red");</code>
<code>removeClass(...)</code>	class 속성을 제거한다.	<code>\$("p").removeClass("red");</code>
※ CSS 제어		
<code>css(... , ...)</code>	지정한 속성값을 설정한다.	<code>\$("p").css("color","red");</code>
<code>css({ 속성명 : "속성값", 속성명 : "속성값", (중략) 속성명 : "속성값" });</code>	복수의 css 속성 값을 동시에 설정	<code>\$("p").css({ background-color : "yellow", fontWeight : "bold", color : "red" });</code>
<code>css(...)</code>	지정한 속성값을 가져온다.	<code>\$("p").text(\$("p").css("color"));</code>