자바스크립트 메서드 함수

산술 연산자의 종류와 기본형

종류	기본형	설명
+	A+B	더하기
-	A-B	빼기
*	A*B	곱하기
/	A/B	나누기
%	A%B	나머지

대입 연산자의 종류

종 류	풀 이
A = B	A = B
A += B	A = A+B
A *= B	A = A*B
A /= B	A = A/B
A %= B	A = A%B

비교 연산자의 종류

종류	설명	비고
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 자료형(type)은 숫자형이든 문자형이든 상관하지 않고 표기된 숫자만 일치하면 true를 반환합니다. 가령, 숫자형 10과 문자형 "10"은 같은 것으로 인식되어 true를 반환합니다.
A != B	A와 B는 다르다.	숫자를 비교할 경우 자료형은 숫자형이든 문자형이든 상관하지 않고 표기된 숫자만 다르면 true를 반환합니다. 가령, 숫자형 10과 문자형 "10"은 같은 것으로 인식되어 A!=B에 대해 false를 반환합니다.

논리 연산자의 종류

종류	설명
||	or 연산자라 부르며, 피연산자 중 값이 하나라도 true가 존재하면 true로 결괏값을 반환합니다.
&&	and 연산자라 부르며, 피연산자 중 값이 하나라도 false가 존재하면 false로 결괏값을 반환합니다.
!	not 연산자라 부르며, 단항 연산자입니다. 피연산자의 값이 true이면 반대로 false로 결괏값을 반환합니다.

if 문

if 문은 조건식을 만족(true)할 경우에만 코드를 실행합니다. 다음은 if 문의 기본형입니다. 조건식은 앞에서 배웠던 Boolean( ) 내장 메서드와 마찬가지로 그 어떤 데이터를 입력해도 true 또는 false를 반환합니다. 이 내용은 적용 예제를 보며 자세히 살펴보겠습니다.

기본형

```
if(조건식){
 자바스크립트 코드;
}
```

else 문

else 문은 조건식을 만족할(true) 경우와 만족하지 않을(false) 경우에 따라 실행되는 코드가 달라집니다. 즉, 두 가지 결과가 나올 수 있습니다.
다음은 조건식의 만족 여부에 따라 실행되는 코드가 달라지는 else 문의 기본형입니다.

기본형

```
if(조건식){
   자바스크립트 코드1;
}else{
   자바스크립트 코드2;
}
```

else if 문

else if 문은 두 가지 이상의 조건식과 정해 놓은 조건을 만족하지 않았을 때 실행되는 코드로 이루어져 있습니다.

다음은 else if 문의 기본형입니다. 가장 위에 있는 조건식1부터 5까지 차례로 조건 검사를 하면서 만족(true)하는 값이 나오면 그에 해당하는 코드를 실행하고 조건문을 종료합니다. 조건식 중 만족(true)하는 값이 하나도 없으면 else 문의 중괄호{...}에 있는 코드를 실행합니다.

기본형

```
if(조건식1){
 코드1;
}else if(조건식2){
 코드2;
}else if(조건식3){
 코드3;
}else if(조건식4){
 코드4;
}else if(조건식5){
 코드5;
}else{
 코드6;
}
```

switch 문

선택문인 switch 문은 변수에 저장된 값과 switch 문에 있는 경우(case)의 값을 검사하여 변수와 경우의 값에서 일치하는 값이 있을 때 그에 해당하는 코드를 실행합니다. if 문과 용도는 비슷하나 if 문은 만족하는 데이터가 여러 개일 경우에 주로 사용하고, switch 문은 여러 경우의 값 중 일치하는 데이터를 찾아 그에 해당하는 코드를 실행시킬 때 사용합니다.

다음은 switch 문의 기본형입니다.

기본형

```
var 변수=초깃값;
switch(변수){
  case 값1: 코드1;
  break;
  case 값2: 코드2;
  break;
  case 값3: 코드3;
  break;
  case 값4: 코드4;
  break;

  default: 코드5;
}
```

while 문

while 문은 조건식을 만족할 때까지 코드를 여러 회 반복하여 실행할 수 있습니다.

다음은 while 문의 기본형입니다. while 문은 조건식을 만족할 때까지 중괄호({...}) 안에 있는 코드를 반복하여 실행합니다. while 문의 실행 순서는 ❶ 조건식을 검사하고, 만족하면 ❷ 중괄호 안에 있는 코드와 증감식을 실행합니다. 그리고 ❸ 다시 조건식을 검사합니다.

기본형

```
var 변수=초깃값;
while(❶❸조건식){
  ❷
  자바스크립트 코드;
  증감식;
}
```

do while 문

while 문의 경우에는 조건식의 만족 여부를 먼저 검사한 후 중괄호에 있는 코드의 실행 여부를 결정했습니다. 하지만 do while 문은 반드시 한 번은 코드를 실행하고 조건식을 검사합니다.

다음은 do while 문의 기본형입니다. 먼저 중괄호({...})에 있는 코드를 실행하고 조건식을 검사합니다.

기본형

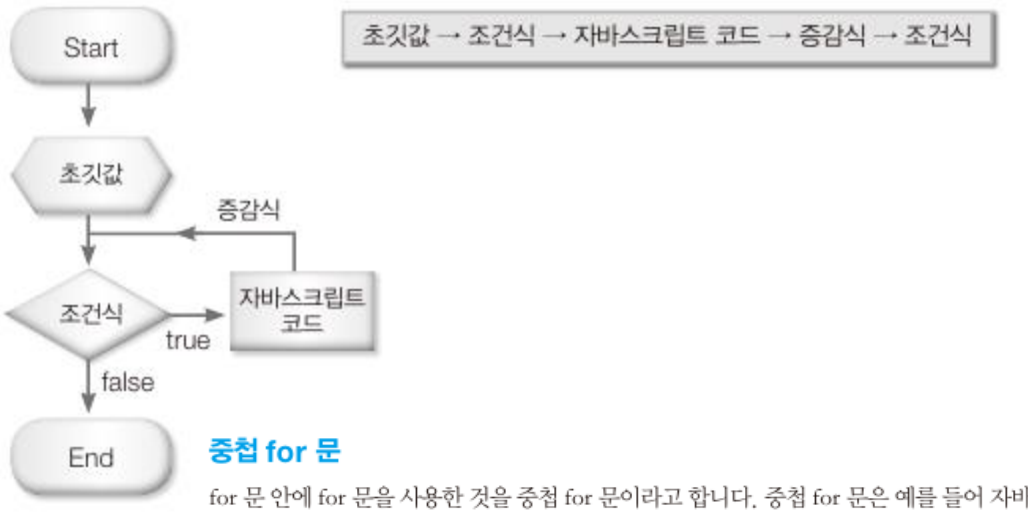
```
var 변수=초깃값;

do{
   자바스크립트 코드;
   증감식;
}while(조건식)
```

for 문

for 문은 조건식을 만족할 때까지 특정 코드를 반복하여 실행합니다. 사용 방법은 while 문과 같지만 while 문보다 사용하기 편해 사용 빈도가 높은 편입니다.

기본형
```
for(초깃값; 조건식; 증감식){
  자바스크립트 코드;
}
```

다음은 for 문의 실행 순서를 그림으로 나타낸 것입니다.



중첩 for 문

for 문 안에 for 문을 사용한 것을 중첩 for 문이라고 합니다. 중첩 for 문은 예를 들어 자바스크립트를 이용해 3행 5열의 표를 만든다고 할 때 1행씩 행이 만들어질 때마다 5개의 열을 만들어야 할 경우에 사용합니다. 중첩 for 문의 기본형은 다음과 같습니다.

기본형
```
for(초깃값; 조건식; 증감식){      //바깥쪽 for 문
  for(초깃값; 조건식; 증감식){    //안쪽 for 문
    자바스크립트 코드;
  }
}
```

break 문

반복문인 while 문 또는 for 문에서 break 문을 실행하면 조건식과 상관없이 강제로 반복문을 종료합니다. 즉, break 문은 반복문을 강제로 종료할 때 사용합니다.

다음은 for 문과 while 문에서 break 문이 사용된 기본형입니다. break 문이 코드보다 앞에 있으므로 코드는 실행되지 않고 for 문과 while 문이 바로 종료됩니다.

기본형
```
for(초깃값; 조건식; 증감식){
  break;  //반복문을 강제로 종료합니다.
  자바스크립트 코드;
}
```

```
var 변수=초깃값;
while(조건식){
  break;  //반복문을 강제로 종료합니다.
  자바스크립트 코드;
  증감식;
}
```

continue 문

continue 문은 반복문에서만 사용할 수 있습니다. while 문에 사용할 경우 continue 문 다음에 오는 코드는 무시하고 바로 조건식으로 이동해 조건 검사를 합니다. 즉, while 문 안에 있는 continue 문은 "다음에 오는 코드는 무시하고 조건식에서 조건 검사를 실행해!"라고 말하는 것이죠.

for 문에서 continue 문을 실행할 경우에는 continue 문 다음에 오는 코드는 무시하고 바로 증감식으로 이동하여 증감 연산을 실행합니다. 즉, for 문 안에 있는 continue 문은 "다음에 오는 코드는 무시하고 증감식을 바로 실행해!"라고 말하는 것이죠.

continue 문의 기본형은 다음과 같습니다.

기본형
```
for(초깃값; 조건식; 증감식){
  continue;
  자바스크립트 코드;
}
```

```
var 변수=초깃값;
while(조건식){
  증감식;
  continue;
  자바스크립트 코드;
}
```

날짜 관련 메서드

날짜 정보를 가져올 때(GET)		날짜 정보를 수정할 때(SET)	
getFullYear( )	연도 정보를 가져옴	setFullYear( )	연도 정보만 수정함
getMonth( )	월 정보를 가져옴(현재 월 – 1)	setMonth( )	월 정보만 수정함(월 – 1)
getDate( )	일 정보를 가져옴	setDate( )	일 정보만 수정함
getDay( )	요일 정보를 가져옴(일: 0 ~ 토: 6)	'요일'은 날짜를 바꾸면 자동으로 바뀌므로 setDay( )는 없음	
getHours( )	시 정보를 가져옴	setHours( )	시 정보만 수정함
getMinutes( )	분 정보를 가져옴	setMinutes( )	분 정보만 수정함
getSeconds( )	초 정보를 가져옴	setSeconds( )	초 정보만 수정함
getMilliseconds( )	밀리초 정보를 가져옴(1/1,000초 단위)	setMilliseconds( )	밀리초 정보만 수정함
getTime( )	1970년 1월 1일부터 경과된 시간을 밀리초로 표기함	setTime( )	1970년 1월 1일부터 경과된 시간을 밀리초로 수정함
toGMTString( )	GMT 표준 표기 방식으로 문자형 데이터로 반환함	toLocaleString( )	운영 시스템 표기 방식으로 문자형 데이터로 반환함

```
1초 = 1,000(msc)
1분(60초) = 1,000 * 60                  //60,000(msc)
1시간(60분) = 1,000 * 60 * 60           //3,600,000(msc)
1일(60분*24) = 1,000 * 60 * 60 * 24     //86,400,000(msc)
```

수학 객체의 메서드 및 상수

종류	설명
Math.abs(숫자)	숫자의 절댓값을 반환합니다.
Math.max(숫자 1, 숫자 2 , 숫자 3, 숫자 4)	숫자 중 가장 큰 값을 반환합니다.
Math.min(숫자 1, 숫자 2, 숫자 3, 숫자 4)	숫자 중 가장 작은 값을 반환합니다.
Math.pow(숫자, 제곱값)	숫자의 거듭제곱값을 반환합니다.
Math.random( )	0~1 사이의 난수를 반환합니다.
Math.round(숫자)	소수점 첫째 자리에서 반올림하여 정수를 반환합니다.
Math.ceil(숫자)	소수점 첫째 자리에서 무조건 올림하여 정수를 반환합니다.
Math.floor(숫자)	소수점 첫째 자리에서 무조건 내림하여 정수를 반환합니다.
Math.sqrt(숫자)	숫자의 제곱근값을 반환합니다.
Math.PI	원주율 상수를 반환합니다.

배열 객체의 메서드 및 속성

종류	설명
join(연결 문자)	배열 객체의 데이터를 연결 문자 기준으로 1개의 문자형 데이터로 반환합니다.
reverse( )	배열 객체의 데이터 순서를 거꾸로 바꾼 후 반환합니다.
sort( )	배열 객체의 데이터를 오름차순으로 정렬합니다.
slice(index1, index2)	배열 객체의 데이터 중 원하는 인덱스 구간만큼 잘라서 배열 객체로 가져옵니다.
splice( )	배열 객체의 지정 데이터를 삭제하고 그 구간에 새 데이터를 삽입할 수 있습니다.
concat( )	2개의 배열 객체를 하나로 결합합니다.
pop( )	배열에 저장된 데이터 중 마지막 인덱스에 저장된 데이터를 삭제합니다.
push(new data)	배열 객체의 마지막 인덱스에 새 데이터를 삽입합니다.
shift( )	배열 객체에 저장된 데이터 중 첫 번째 인덱스에 저장된 데이터를 삭제합니다.
unshift(new data)	배열 객체의 가장 앞의 인덱스에 새 데이터를 삽입합니다.
length	배열에 저장된 총 데이터의 개수를 반환합니다.

문자열 객체의 메서드 및 속성

종류	설명
charAt(index)	문자열에서 인덱스 번호에 해당하는 문자를 반환합니다. 예 var str="web he she"; str.charAt(2); ← "b"를 반환함
indexOf("찾을 문자")	문자열에서 왼쪽부터 찾을 문자와 일치하는 문자를 찾아 제일 먼저 일치하는 문자의 인덱스 번호를 반환합니다. 만일 찾는 문자가 없으면 –1을 반환합니다. 예 var str="web he she he"; str.indexOf("he"); ← 4를 반환함
lastIndexOf("찾을 문자")	문자열에서 오른쪽부터 찾을 문자와 일치하는 문자를 찾아 제일 먼저 일치하는 문자의 인덱스 번호를 반환합니다. 만일 찾는 문자가 없으면 –1을 반환합니다. 예 var str="web he she he"; str.lastIndexOf("he"); ← 11을 반환함
match("찾을 문자")	문자열에서 왼쪽부터 찾을 문자와 일치하는 문자를 찾아 제일 먼저 찾은 문자를 반환합니다. 만일 찾는 문자가 없으면 null을 반환합니다. 예 var str="web he she he"; str.match("boy"); ← null을 반환함
replace("바꿀 문자", "새 문자")	문자열에서 왼쪽부터 바꿀 문자와 일치하는 문자를 찾아 제일 먼저 찾은 문자를 새 문자로 치환합니다. 예 var str="web he she"; str.replace("web","html"); ← "html he she"를 반환함
search("찾을 문자")	문자열에서 왼쪽부터 찾을 문자와 일치하는 문자를 찾아 제일 먼저 일치하는 인덱스 번호를 반환합니다. 예 var str="web he she"; str.search("he"); ← 4를 반환함
slice(a, b)	a개의 문자를 자르고 b번째 이후에 문자를 자른 후 남은 문자를 반환합니다. 예 var str="hello javascript" str.slice(3, 7); ← "lo j"를 반환함 str.slice(3, 7);은 문자열에서 3글자 "hel"까지 자르고 7번째 이후 글자인 "j" 이후부터 "avascript"를 자른 후 나머지 "lo j"를 반환합니다. 예 var str="hello javascript" str.slice(3, –3); ← "llo javascr"를 반환함 여기에서 –1은 뒤에서부터 첫 번째 글자를 가리킵니다.
substring(a, b)	a 인덱스부터 b 인덱스 이전 구간의 문자를 반환합니다. 예 var str="hello javascript" str.substring(3, 7); ← "lo j"를 반환함 예 var str="hello javascript" str.substring(3, –3); ← "hel"을 반환함. 여기에서 –3은 인덱스 0을 가리킵니다. 그러므로 인덱스 0부터 인덱스 3 이전 구간의 문자를 반환합니다.
substr(a, 문자 개수)	문자열에 a 인덱스부터 지정한 문자 개수만큼 문자열을 반환합니다. 예 var str="hello javascript" str.substring(3, 2); ← "lo"를 반환함 str.str(3, 2);은 3 문자열에서 인덱스 3인 "l"부터 2글자를 가져와서 "lo"를 반환합니다.
split("문자")	지정한 문자를 기준으로 문자 데이터를 나누어 배열에 저장하여 반환합니다. 예 var str="webkmobilek2002"; var arr=str.split("k"); ← arr= (Index) (0) (1) (2) "web" "mobile" "2002"
toLowerCase( )	문자열에서 영문 대문자를 모두 소문자로 바꿉니다. 예 var str="ABC"; str.toLowerCase( ); ← "abc"를 반환함
toUpperCase( )	문자열에서 영문 소문자를 모두 대문자로 바꿉니다. 예 var str="abc"; str.toUpeerCase( ); ← "ABC"를 반환함
length	문자열에서 문자의 개수를 반환합니다. 예 var str="hello welcome"; str.length ← 13을 반환함
concat("새로운 문자")	문자열에 새로운 문자열을 결합합니다. 예 var str="hello"; str.concat("web"); ← "hello web"을 반환함
charCodeAt(index)	문자열 index에 해당 문자의 아스키 코드값을 반환합니다. 예 var str = "ABC", str.charCodeAt(0) ← "A"의 아스키 코드값 65를 반환함 좀 더 자세한 아스키 코드값을 확인하고 싶다면 다음 주소를 참고하세요. http://office.microsoft.com/ko-kr/word-help/HA010167539.aspx
fromCharCode(아스키 코드 값)	아스키 코드값에 해당하는 문자를 반환합니다. 예 String.fromCharCode(65); ← "A"를 반환함
trim( )	문자의 앞 또는 뒤에 공백 문자열을 삭제합니다(" hello " → "hello"). 예 str="hello"; str.trim( ); ← 공백이 제거된 "hello"를 반환함

window 객체의 메서드 종류

종류	설명
open("URL", "새 창 이름", "새 창 옵션")	URL 페이지를 새 창으로 나타냅니다.
alert(data)	경고 창을 나타내고 데이터를 보여줍니다. 방문자가 [확인] 버튼을 누르면 alert( )를 사용한 다음 위치의 코드를 수행합니다.
prompt("질문", "답변")	질문과 답변으로 질의응답 창을 나타냅니다.
confirm("질문 내용")	질문 내용으로 확인이나 취소 창을 나타냅니다. [확인] 버튼을 누르면 true를 반환하고, [취소] 버튼을 누르면 false를 반환합니다.
moveTo(x, y)	지정한 새 창의 위치를 이동합니다.
resizeTo(width, height)	지정한 새 창의 크기를 변경합니다.
setInterval(function( ) { 자바스크립트 코드 }, 일정 시간 간격)	지속적으로 일정한 시간 간격으로 함수를 호출하여 코드를 실행합니다.
setTimeout(function( ) { 자바스크립트 코드 }, 일정 시간 간격)	단 한 번 일정한 시간 간격으로 함수를 호출하여 코드를 실행합니다.

새 창의 옵션 종류

속성	설명
❶ width	새 창의 너비를 설정합니다.
❷ height	새 창의 높이를 설정합니다.
❸ left	새 창의 수평(X축) 위치를 설정합니다.
❹ top	새 창의 수직(Y축) 위치를 설정합니다.
❺ scrollbars	새 창의 스크롤바의 숨김/노출을 설정합니다(숨김 = no, 노출 = yes).
location	새 창의 URL 주소 입력 영역의 숨김/노출을 설정합니다(숨김 = no, 노출 = yes).
status	새 창의 상태 표시줄 영역의 숨김/노출을 설정합니다(숨김 = no, 노출 = yes).
toolbars	새 창의 도구 상자 영역의 숨김/노출을 설정합니다(숨김 = no, 노출 = yes).

screen 객체의 속성 종류

종류	설명
screen.width	화면의 너빗값을 반환합니다.
screen.height	화면의 높잇값을 반환합니다.
screen.availWidth	작업 표시줄을 제외한 화면의 너빗값을 반환합니다.
screen.availHeight	작업 표시줄을 제외한 화면의 높잇값을 반환합니다.
screen.colorDepth	사용자 모니터가 표현 가능한 컬러 bit를 반환합니다.

location 객체의 속성 종류

종류	설명
location.href	주소 영역의 참조 주소를 설정하거나 URL을 반환합니다. 예 http://www.easyspub.co.kr:80/Main/pub#view
location.hash	URL의 해시값(#에 명시된 값)을 반환합니다. 예 http://www.easyspub.co.kr/Main/pub#view
location.hostname	URL의 호스트 이름을 설정하거나 반환합니다. 예 http://www.easyspub.co.kr:80/
location.host	URL의 호스트 이름과 포트 번호를 반환합니다. 예 http://www.easyspub.co.kr:80/
location.protocol	URL의 프로토콜을 반환합니다. 예 http://www.easyspub.co.kr:80/
location.search	URL의 쿼리(요청값)를 반환합니다. 예 http://www.easyspub.com?pageNum=1&sort=DESC
location.reload( )	마치 브라우저에서 F5 키를 누른 것처럼 새로 고침합니다.

history 객체의 속성 종류

종류	설 명
history.back( )	이전 방문 사이트로 이동합니다.
history.forward( )	다음 방문 사이트로 이동합니다.
history.go(이동 숫자)	이동 숫자에 –2를 입력하면 2단계 이전의 방문 사이트로 이동합니다.
history.length	방문 기록에 저장된 목록의 개수를 반환합니다.

navigator 객체의 속성 종류

종류	설명
navigator.appCodeName	현재 브라우저의 코드명을 반환합니다. 현 시점의 모든 브라우저는 "Mozilla"를 반환합니다.
navigator.appName	현재 브라우저의 이름을 반환합니다. 현 시점의 모든 브라우저는 "Netscape"를 반환합니다.
navigator.appVersion	현재 브라우저의 버전 정보를 반환합니다. 현 시점의 모든 브라우저는 "5.0(Windows)"을 반환합니다.
navigator.language	현재 브라우저가 사용하고 있는 언어를 반환합니다. 한국어를 사용할 경우에는 "ko"를 반환합니다.
navigator.product	현재 브라우저의 엔진 이름을 반환합니다. 크롬의 경우는"Gecko"를 반환합니다.
navigator.platform	현재 컴퓨터의 운영체제 정보를 제공합니다. 운영체제가 윈도우이고 시스템 종류가 64비트(bit)라도 브라우저가 32비트로 설치되었다면 "Win32"라고 나타납니다.
navigator.onLine	온라인 상태 여부에 대한 정보를 제공합니다. 만일 인터넷이 정상적으로 연결되어 있는 상태라면 true 값을 반환합니다.
navigator.userAgent	브라우저와 운영체제의 종합 정보를 제공합니다.

내장 함수

내장 함수란 자바스크립트 엔진에 내장된 함수를 말합니다. 지금까지는 개발자가 함수를 정의하고 호출문을 사용해 함수 안에 있는 코드를 실행했습니다. 하지만 내장 함수는 개발자가 함수를 직접 선언하지 않아도 됩니다. 즉, 자바스크립트에 이미 내장된 함수는 바로 호출할 수 있습니다.

자바스크립트는 다음과 같이 다양한 내장 함수를 제공합니다.

내장 함수의 종류

종류	설명	사용 예
encodeURI()	문자를 유니 코드값으로 인코딩합니다. (영문, 숫자, 일부 기호(; , / ? : @ & = + $)는 제외)	encodeURI("?query=값"); → "?query=%EA%B0%91"
encodeURIComponent()	문자를 유니 코드값으로 인코딩합니다(영문, 숫자 제외).	encodeURIComponent("?query=값") → "%3Fquery%3D%EA%B0%91"
decodeURI()	유니 코드값을 디코딩해 다시 문자화합니다.	decodeURI("?query=%EA%B0%91") → "?query=값"
decodeURIComponent()	유니 코드값을 디코딩해 다시 문자화합니다.	decodeURIComponent("%3Fquery%3D%EA%B0%91") → "?query=값"
parseInt()	문자열 데이터를 정수형 데이터로 반환합니다.	parseInt("5.12") → 5 parseInt("15px") → 15
parseFloat()	문자열 데이터를 실수형 데이터로 반환합니다.	parseInt("5.12") → 5.12 parseInt("65.5%") → 65.5
String()	문자형 데이터로 반환합니다.	String(5) → "5"
Number()	숫자형 데이터로 반환합니다.	Number("5") → 5
Boolean()	논리형 데이터로 반환합니다.	Boolean(5) → true Boolean(null) → false
isNaN()	is Not a Number의 약자이며 숫자가 아닌 문자가 포함되어 있으면 true를 반환합니다.	isNaN("5-3") → true isNaN("53") → false
eval()	문자형 데이터를 따옴표가 없는 자바스크립트 코드로 처리합니다	eval("15 + 5") → 20

제이쿼리 메서드 함수

기본 선택자 종류

구분	종류	사용법	설명
직접 선택자	전체 선택자	$("*")	모든 요소를 선택합니다.
	아이디 선택자	$("#아이디명")	id 속성에 지정한 값을 가진 요소를 선택합니다.
	클래스 선택자	$(".클래스명")	class 속성에 지정한 값을 가진 요소를 선택합니다.
	요소 선택자	$("요소명")	지정한 요소명과 일치하는 요소들만 선택합니다.
	그룹 선택자	$("선택 1, 선택 2, 선택 3, ...선택 n")	선택 1, 선택 2, 선택 3, ...선택 n에 지정된 요소들을 한 번에 선택합니다.
	종속 선택자	$("p.txt_1") $("p#txt_1")	<p> 요소 중 class 값이 txt_1인 요소 또는 id 값이 txt_1인 요소를 선택합니다.

❶ 인자값을 사용해 CSS 속성과 값을 전달하는 방식
$("요소 선택").css("속성명1", "값1").css("속성명2", "값2");

❷ 객체를 사용해 CSS 속성과 값을 전달하는 방식
$("요소 선택").css({"속성명1" : "값1", "속성명2" : "값2"..., "속성명n" : "값n"});

인접 관계 선택자	부모 요소 선택자	$("요소 선택").parent( )	선택한 요소의 부모 요소를 선택합니다.
	상위 요소 선택자	$("요소 선택").parents( )	선택한 요소의 상위 요소를 모두 선택합니다.
	가장 가까운 상위 요소 선택자	$("요소 선택").closest("div")	선택한 요소의 상위 요소 중 가장 가까운 만 선택합니다.
	하위 요소 선택자	$("요소 선택 하위 요소")	선택한 요소에 지정한 하위 요소를 선택합니다.
	자식 요소 선택자	$("요소 선택>자식 요소")	선택한 요소를 기준으로 자식 관계에 지정한 요소만 선택합니다.
	자식 요소들 선택자	$("요소 선택").children( )	선택한 요소의 모든 자식 요소를 선택합니다.
	형(이전) 요소 선택자	$("요소 선택").prev( )	선택한 요소의 바로 이전 요소를 선택합니다.
	형(이전) 요소들 선택자	$("요소 선택").prevAll( )	선택한 요소의 이전 요소 모두를 선택합니다.
	지정 형(이전) 요소들 선택자	$("요소 선택").prevUntil ("요소명")	선택한 요소부터 지정한 요소의 이전 요소까지 모두 선택합니다.
	동생(다음) 요소 선택자	$("요소 선택").next( ) $("요소 선택+다음 요소")	선택한 요소의 다음 요소를 선택합니다.
	동생(다음) 요소들 선택자	$("요소 선택").nextAll( )	선택한 요소의 다음 요소 모두를 선택합니다.
	지정 동생(다음) 요소들 선택자	$("요소 선택").nextUntil("h2")	선택한 요소부터 지정한 요소의 다음 요소까지 모두 선택합니다.
	전체 형제 요소 선택자	$(".box_1").siblings( )	class 값이 box_1인 요소의 형제 요소 전체를 선택합니다.

위치 탐색 선택자 종류

종류	사용법	설명
$("요소 선택:first") $("요소 선택").first( )	$("li:first") $("li").first( )	전체 〈li〉 요소 중 첫 번째 요소만 선택합니다.
$("요소 선택:last") $("요소 선택").last( )	$("li:last") $("li").last( )	전체 〈li〉 요소 중 마지막 요소만 선택합니다.
$("요소 선택:odd")	$("li:odd")	〈li〉 요소 무리 중 짝수 번째(홀수 인덱스) 요소만 선택합니다.
$("요소 선택:even")	$("li:even")	〈li〉 요소 무리 중 홀수 번째(짝수 인덱스) 요소만 선택합니다.
$("요소 선택:first-of-type")	$("li:first-of-type")	〈li〉 요소 무리 중 첫 번째 요소만 선택합니다.
$("요소 선택:last-of-type")	$("li:last-of-type")	〈li〉 요소 무리 중 마지막 요소만 선택합니다.
$("요소 선택:nth-child(숫자)")	$("li:nth-child(3)")	〈li〉 요소 무리 중 세 번째 요소만 선택합니다.
$("요소 선택:nth-child(숫자n)")	$("li:nth-child(3n)")	〈li〉 요소 무리 중 3의 배수 번째에 있는 요소만 선택합니다.
$("요소 선택:nth-last-of-type(숫자)")	$("li:nth-last-of-type(2)")	〈li〉 요소 무리 중 마지막 위치로부터 두 번째에 있는 요소만 선택합니다.
$("요소 선택:only-child")	$("li:only-child")	부모 요소 내에 〈li〉 요소가 1개뿐인 〈li〉 요소만 선택합니다.

$("요소 선택:eq(index)") $("요소 선택").eq(index)	$("li:eq(2)") $("li").eq(2)	〈li〉 요소 중 인덱스 2가 참조하는 요소를 불러옵니다.
$("요소 선택:gt(index)")	$("li:gt(1)")	〈li〉 요소 중 인덱스 1보다 큰 인덱스가 참조하는 요소를 불러옵니다.
$("요소 선택:lt(index)")	$("li:lt(1)")	〈li〉 요소 중 인덱스 1보다 작은 인덱스가 참조하는 요소를 불러옵니다.
$("요소 선택").slice(index)	$("li").slice(2)	〈li〉 요소 중 인덱스 2부터 참조하는 요소를 불러옵니다.

배열 관련 메서드의 종류

종류	사용법	설명
each( ) / $.each( )	$("요소 선택").each(function) $.each($("요소 선택"), function)	배열에 저장된 문서 객체만큼 메서드가 반복 실행됩니다. 배열에 저장된 객체의 인덱스 순서대로 하나씩 접근하여 객체를 선택하고 인덱스를 구합니다.
$.map( )	$.map(Array, function)	배열에 저장된 데이터 수만큼 메서드가 반복 실행됩니다. 함수에서 반환된 데이터는 새 배열에 순서대로 저장됩니다. 새로 저장된 배열 객체를 반환합니다.
$.grep( )	$.grep(Array, function)	배열에 저장된 데이터 수만큼 메서드가 반복 실행됩니다. 반환값이 true인 경우에만 배열의 데이터가 인덱스 오름차순으로 새 배열에 저장되며 그 배열을 반환합니다.
$.inArray( )	$.inArray(data, Array, start index)	배열 안에서 데이터를 찾습니다. 데이터를 찾으면 가장 맨 앞 데이터의 인덱스를 반환하고, 찾지 못하면 −1을 반환합니다. start index의 값을 지정하면 해당 위치부터 데이터를 찾습니다.
$.isArray( )	$.isArray(object)	입력한 객체가 배열 객체라면 true를, 아니면 false를 반환합니다.
$.merge( )	$.merge(Array1, Array2)	인자값으로 입력한 2개의 배열 객체를 하나로 그룹화합니다.
index( )	$("요소 선택").index("지정 요소 선택")	선택자로 요소를 먼저 선택합니다. 그런 다음 지정한 요소의 인덱스 정보를 가져옵니다.

속성 탐색 선택자의 종류

종류	사용법	설명
$("요소 선택[속성]")	$("li[title]")	〈li〉 요소 중 title 속성이 포함된 요소만 선택합니다.
$("요소 선택[속성=값]")	$("li[title='리스트']")	〈li〉 요소 중 title 속성값이 '리스트'인 요소만 선택합니다.
$("요소 선택[속성^=텍스트]")	$("a[href^='http://']")	〈li〉 요소 중 href 속성값이 'http://'로 시작하는 요소만 선택합니다.
$("요소 선택[속성$=텍스트]")	$("a[href$='.com']")	〈li〉 요소 중 href 속성값이 '.com'으로 끝나는 요소만 선택합니다.
$("요소 선택[href*=텍스트]")	$("a[href*='easyspub']")	〈li〉 요소 중 href 속성값 중에서 'easyspub'을 포함하는 요소만 선택합니다.
$("요소 선택:hidden")	$("li:hidden")	〈li〉 요소 중 숨겨져 있는 요소만 선택합니다.
$("요소 선택:visible")	$("li:visible")	〈li〉 요소 중 보이는 요소만 선택합니다.
$(":text")	$(":text")	〈input〉 요소 중 type 속성값이 "text"인 요소만 선택합니다.
$(":selected")	$(":selected")	selected 속성이 적용된 요소만 선택합니다.
$(":checked")	$(":checked")	checked 속성이 적용된 요소만 선택합니다.

콘텐츠 탐색 선택자의 종류

종류	사용법	설명
$("요소 선택:contains(텍스트)")	$("li:contains('내용2')")	〈li〉 요소 중 '내용2'라는 텍스트를 포함하는 요소만 선택합니다.
$("요소 선택").contents( )	$("p").contents( )	선택한 요소의 하위 요소 중 가장 가까운 하위 요소를 선택합니다.
$("요소 선택:has(요소명)") $("요소 선택").has(요소명)	$("li:has('span')") $("li").has('span')	〈li〉 요소 중 〈span〉을 포함하는 요소만 선택합니다.
$("요소 선택:not(:제외 요소)") $("요소 선택").not(:제외 요소)	$("li:not(:first)") $("li")	〈li〉 요소 중 첫 번째 요소만 제외하고 선택합니다.
$(요소 선택).filter(필터 요소)	$("li").filter(".list2")	〈li〉 요소 중 class 값이 "list2"인 요소만 선택합니다.
$(요소 선택1).find(요소 선택2)	$("li").find("strong")	〈li〉 요소의 하위 요소인 〈strong〉만 선택합니다.
$("요소 선택1").closest("요소 선택2")	$("strong").closest("div")	〈strong〉을 감싸는 상위 〈div〉 요소 중 가장 가까운 상위 요소를 선택합니다.
end( )	$("li").children("a").end( )	필터링이 실행되기 이전의 요소인 〈li〉가 선택됩니다.

메서드 종류

종류	사용법	설명
is(":요소 상태")	$(".txt1").is(":visible")	선택한 요소가 보이면 true를 반환합니다.
$.noConflict( )	var 변수 = $.noConflict( ); 변수("요소 선택")	$.noConflict( ) 함수를 이용하면 현재 제이쿼리에서 사용 중인 $ 메서드 사용을 중단하고 새로 지정한 변수명 메서드를 사용합니다.
get( )	$("요소 선택").get(0). style.color = "#f00"	선택자에 get(0)을 적용하면 자바스크립트 DOM 방식의 스타일을 사용할 수 있습니다.

▶ 제이쿼리로 선택한 요소는 자바스크립트 DOM 방식의 스타일을 사용할 수 없습니다.

속성 조작 메서드

종류	사용법	설명
html( ) html("새 요소")	$("요소 선택").html( ); $("요소 선택").html("새 요소");	선택한 요소의 하위 요소를 가져옵니다. 선택한 요소의 하위 요소를 모두 제거하고, 그 위치에 지정한 새 요소를 생성합니다.
text( ) text("새 텍스트")	$("요소 선택").text( ); $("요소 선택").text("새 텍스트");	선택한 요소가 감싸는 모든 텍스트를 가져옵니다. 선택한 요소의 하위 요소를 모두 제거하고, 그 위치에 지정한 새 텍스트를 생성합니다.
attr("속성명") attr("속성명","새 값")	$("요소 선택").attr("속성명"); $("요소 선택").attr("속성명","새 값");	선택한 요소의 지정한 속성(attribute)값을 가져옵니다. 요소를 선택하여 지정한 속성에 새 값을 적용합니다.
removeAttr("속성명")	$("요소 선택").removeAttr("속성명");	선택한 요소의 지정한 속성만 제거합니다.
prop("상태 속성명") prop("상태 속성명", 새 값)	$("요소 선택").prop("상태 속성명"); $("요소 선택").prop("상태 속성명","새 값");	선택한 요소의 상태 속성값을 가져옵니다. 요소를 선택하여 상태 속성에 새 값을 적용합니다.
val( ) val("새 값")	$("요소 선택").val( ); $("요소 선택").val("새 값");	선택한 폼 요소의 value 값을 가져옵니다. 폼 요소를 선택하여 value 속성에 새 값을 적용합니다.
css("속성명") css("속성명","새 값")	$("요소 선택").css("속성명"); $("요소 선택").css("속성명","새 값");	선택한 요소의 지정한 스타일(CSS) 속성값을 가져옵니다. 요소를 선택하여 지정한 스타일(CSS) 속성에 새 값을 적용합니다.
addClass("class 값")	$("요소 선택").addClass("class 값");	선택한 요소의 class 속성에 새 값을 추가합니다.
removeClass("class 값")	$("요소 선택").removeClass("class 값");	선택한 요소의 class 속성에서 지정한 값만 제거합니다.
toggleClass("class 값")	$("요소 선택").toggleClass("class 값")	선택한 요소의 class 값에 지정한 값이 포함되어 있으면 제거하고 속성값이 없으면 추가합니다.
hasClass("class 값")	$("요소 선택").hasClass("class 값")	선택한 요소의 class 값에 지정한 클래스 값이 포함되어 있으면 true를, 없으면 false를 반환합니다.

수치 조작 메서드

수치 조작 메서드는 요소의 속성을 조작할 때 사용하는 메서드입니다. 다음은 수치 조작 메서드의 종류와 사용법을 정리한 표입니다.

수치 조작 메서드

종류	사용법	설명
height( )	$("요소 선택").height( ); $("요소 선택").height(100);	안쪽 여백과 선을 제외한 높잇값을 반환하거나 변환합니다.
width( )	$("요소 선택").width( ); $("요소 선택").width(100);	안쪽 여백과 선을 제외한 너빗값을 반환하거나 변환합니다.
innerHeight( )	$("요소 선택").innerHeight( ); $("요소 선택").innerHeight(300);	안쪽 여백을 포함한 높잇값을 반환하거나 변환합니다.
innerWidth( )	$("요소 선택").innerWidth( ); $("요소 선택").innerWidth(100);	안쪽 여백을 포함한 너빗값을 반환하거나 변환합니다.
outerHeight( )	$("요소 선택").outerHeight( ); $("요소 선택").outerHeight(100);	선과 안쪽 여백을 포함한 높잇값을 반환하거나 변환합니다.
outerWidth( )	$("요소 선택").outerWidth( ); $("요소 선택").outerWidth(100);	선과 안쪽 여백을 포함한 너빗값을 반환하거나 변환합니다.
position( )	$("요소 선택").position( ).left; $("요소 선택").position( ).top;	선택한 요소의 포지션 위칫값을 반환합니다.
offset( )	$("요소 선택").offset( ).left; $("요소 선택").offset( ).top;	선택한 요소가 문서에서 수평/수직으로 얼마나 떨어져 있는지에 대한 값을 반환합니다.
scrollLeft( )	$(window).scrollLeft( );	브라우저의 수평 스크롤 이동 높잇값을 반환합니다.
scrollTop( )	$(window).scrollTop( );	브라우저의 수직 스크롤 이동 너빗값을 반환합니다.

객체 편집 메서드의 종류

종류	사용법	설명
before( )	$("요소 선택").before("새 요소");	선택한 요소의 이전 위치에 새 요소를 추가합니다.
after( )	$("요소 선택").after("새 요소");	선택한 요소의 다음 위치에 새 요소를 추가합니다.
append( )	$("요소 선택").append("새 요소");	선택한 요소의 마지막 위치에 새 요소를 추가합니다.
appendTo( )	$("새 요소").appendTo("요소 선택");	선택한 요소의 마지막 위치에 새 요소를 추가합니다.
prepend( )	$("요소 선택").prepend("새 요소");	선택한 요소의 맨 앞 위치에 새 요소를 추가합니다.
prependTo( )	$("새 요소").prependTo("요소 선택");	선택한 요소의 맨 앞 위치에 새 요소를 추가합니다.
insertBefore( )	$(" 새 요소").insertBefore("요소 선택");	선택한 요소의 이전 위치에 새 요소를 추가합니다.
insertAfter( )	$("새 요소").insertAfter("요소 선택");	선택한 요소의 다음 위치에 새 요소를 추가합니다.
clone( )	$("요소 선택").clone(true or false);	선택한 문서 객체를 복제합니다. 이때 인자값이 true일 경우 하위 요소까지 모두 복제하고, false일 경우에는 선택한 요소만 복제합니다.
empty( )	$("요소 선택").empty( );	선택한 요소의 하위 내용들을 모두 삭제합니다.
remove( )	$("요소 선택").remove( );	선택한 요소를 삭제합니다.
replaceAll( ) / replaceWith( )	$("새 요소").replaceAll("요소 선택"); $("요소 선택").replaceWith("새 요소");	선택한 요소들을 새 요소로 교체합니다.
unwrap( )	$("요소 선택").unwrap( );	선택한 요소의 부모 요소를 삭제합니다.
wrap( )	$("요소 선택").wrap(새 요소);	선택한 요소를 새 요소로 각각 감쌉니다.
wrapAll( )	$("요소 선택").wrapAll( );	선택한 요소를 새 요소로 한꺼번에 감쌉니다.
wrapInner( )	$("요소 선택").wrapInner(새 요소);	선택한 요소의 내용을 새 요소로 각각 감쌉니다.

이벤트 등록 메서드의 종류

구분	종류	설명
로딩 이벤트	load( )	선택한 이미지 또는 프레임 요소에 연동된 소스의 로딩이 완료된 후 이벤트가 발생합니다.
	ready( )	지정한 HTML 문서 객체의 로딩이 완료된 후 이벤트가 발생합니다.
	error( )	이벤트 대상 요소에서 오류가 발생하면 이벤트가 발생합니다.
마우스 이벤트	click( )	선택한 요소를 클릭했을 때 이벤트가 발생합니다.
	dblclick( )	선택한 요소를 연속해서 두 번 클릭했을 때 이벤트가 발생합니다.
	mouseout( )	선택한 요소의 영역에서 마우스 포인터가 벗어났을 때 이벤트가 발생합니다. 이때 하위 요소의 영향을 받습니다.
	mouseover( )	선택한 요소의 영역에 마우스 포인터를 올렸을 때 이벤트가 발생됩니다.
	hover( )	선택한 요소에 마우스 포인터를 올렸을 때와 벗어났을 때 각각 이벤트가 발생합니다.
	mousedown( )	선택한 요소에서 마우스 버튼을 눌렀을 때 이벤트가 발생합니다.
	mouseup( )	선택한 요소에서 마우스 버튼을 눌렀다 떼었을 때 이벤트가 발생합니다.
	mouseenter( )	선택한 요소 범위에 마우스 포인터를 올렸을 때 이벤트가 발생합니다.
	mouseleave( )	선택한 요소 범위에서 마우스 포인터가 벗어났을 때 이벤트가 발생합니다.
	mousemove( )	선택한 요소 범위에서 마우스 포인터를 움직였을 때 이벤트가 발생합니다.
	scroll( )	가로, 세로 스크롤바를 움직일 때마다 이벤트가 발생합니다.
포커스 이벤트	focus( )	선택한 요소에 포커스가 생성되었을 때 이벤트를 발생하거나 선택한 요소에 강제로 포커스를 생성합니다.
	focusin( )	선택한 요소에 포커스가 생성되었을 때 이벤트가 발생합니다.
	focusout( )	포커스가 선택한 요소에서 다른 요소로 이동되었을 때 이벤트가 발생합니다.
	blur( )	포커스가 선택한 요소에서 다른 요소로 이동되었을 때 이벤트가 발생하거나 선택한 요소의 포커스가 강제로 사라지도록 합니다.
	change( )	이벤트 대상인 입력 요소의 값이 변경되고, 포커스가 이동하면 이벤트가 발생합니다. 그리고 강제로 change 이벤트를 발생시킬 때도 사용합니다.
키보드 이벤트	keypress( )	선택한 요소에서 키보드를 눌렀을 때 이벤트가 발생합니다. 그리고 문자 키를 제외한 키의 코드값을 반환합니다.
	keydown( )	선택한 요소에서 키보드를 눌렀을 때 이벤트가 발생합니다. 키보드의 모든 키의 코드값을 반환합니다.
	keyup( )	선택한 요소에서 키보드에서 손을 떼었을 때 이벤트가 발생합니다.

이벤트 객체의 속성 종류

구분	종류	설명
마우스 이벤트	clientX	마우스 포인터의 X 좌표값 반환(스크롤 이동 거리 무시)
	clientY	마우스 포인터의 Y 좌표값 반환(스크롤 이동 거리 무시)
	pageX	스크롤 X축의 이동한 거리를 계산하여 마우스 포인터의 X 좌표값을 반환
	pageY	스크롤 Y축의 이동한 거리를 계산하여 마우스 포인터의 Y 좌표값을 반환
	screenX	화면 모니터를 기준으로 마우스 포인터의 X 좌표값을 반환
	screenY	화면 모니터를 기준으로 마우스 포인터의 Y 좌표값을 반환
	layerX	position을 적용한 요소를 기준으로 마우스 포인터의 X 좌표값을 반환
	layerY	position을 적용한 요소를 기준으로 마우스 포인터의 Y 좌표값을 반환
	button	마우스 버튼의 종류에 따라 값을 반환 (왼쪽: 0, 휠: 1, 오른쪽: 2)
키보드 이벤트	keyCode	키보드의 아스키 코드값을 반환
	altKey	이벤트 발생 시 Alt 키가 눌렸으면 true를, 아니면 false를 반환
	ctrlKey	이벤트 발생 시 Ctrl 키가 눌렸으면 true를, 아니면 false를 반환
	shiftKey	이벤트 발생 시 Shift 키가 눌렸으면 true를, 아니면 false를 반환
전체 이벤트	target	이벤트가 전파된 마지막 요소를 가리킵니다.
	cancelBubble	이벤트의 전파를 차단하는 속성으로, 기본값은 false며, true로 설정하면 전파가 차단됩니다.
	stopPropagation( )	이벤트의 전파를 차단합니다.
	preventDefault( )	기본 이벤트를 차단합니다. 예를 들어 <a>에 클릭 이벤트를 적용하고 사용자가 이벤트를 발생시키면 기본 이벤트가 등록되어 있어 링크 주소로 이동하는데, 이런 기본 이벤트를 차단할 수 있습니다.

그룹 이벤트 등록 메서드의 종류

종류	설명
on( )	이벤트 대상 요소에 2개 이상의 이벤트를 등록합니다. 사용 방식에 따라 이벤트를 등록한 이후에도 동적으로 생성되거나 복제된 요소에도 이벤트가 적용됩니다.
bind( )	이벤트 대상 요소에 2개 이상의 이벤트를 등록합니다.
delegate( )	선택한 요소의 하위 요소에 이벤트를 등록합니다. 이벤트를 등록한 이후에도 동적으로 생성되거나 복제된 요소에도 이벤트가 적용됩니다.
one( )	이벤트 대상 요소에 1개 이상의 이벤트를 등록합니다. 지정한 이벤트가 1회 발생하고 자동으로 해제됩니다.

이벤트 제거 메서드의 종류

종류	설명
off( )	on( ) 메서드로 등록한 이벤트를 제거합니다.
unbind( )	bind( ) 메서드로 등록한 이벤트를 제거합니다.
undelegate( )	delegate( ) 메서드로 등록한 이벤트를 제거합니다.

효과 메서드 종류

구분	종류	설명
숨김	hide( )	요소를 숨깁니다.
	fadeOut( )	요소가 점점 투명해지면서 사라집니다.
	slideUp( )	요소가 위로 접히며 숨겨집니다.
노출	show( )	숨겨진 요소가 노출됩니다.
	fadeIn( )	숨겨진 요소가 점점 선명해집니다.
	slideDown( )	숨겨진 요소가 아래로 펼쳐집니다.
노출, 숨김	toggle( )	hide( ), show( ) 효과를 적용합니다.
	fadeToggle( )	fadeIn( ), fadeOut( ) 효과를 적용합니다.
	slideToggle( )	slideUp( ), slideDown( ) 효과를 적용합니다.
	fadeTo( )	지정한 투명도를 적용합니다.

애니메이션 효과 제어 메서드의 종류

종류	설명
stop( )	현재 실행 중인 효과를 모두 정지시킵니다.
delay( )	지정한 시간만큼 지연했다가 애니메이션을 진행합니다.
queue( )	큐에 사용자 정의 함수를 추가하거나 큐에 대기 중인 함수를 배열에 담아 반환합니다. 그리고 queue( ) 메서드 이후의 애니메이션 효과 메서드는 모두 취소합니다.
clearQueue( )	큐에서 처음으로 진행하고 있는 애니메이션만 제외하고 대기 중인 애니메이션은 모두 제거합니다.
dequeue( )	queue( ) 메서드를 이용하면 대기하고 있는 에니메이션 메서드는 제거됩니다. 하지만 dequeue( ) 메서드를 이용하면 메서드가 제거되는 것을 막을 수 있습니다.
finish( )	선택한 요소의 진행 중인 애니메이션을 강제로 완료 시점으로 보낸 후 종료합니다.