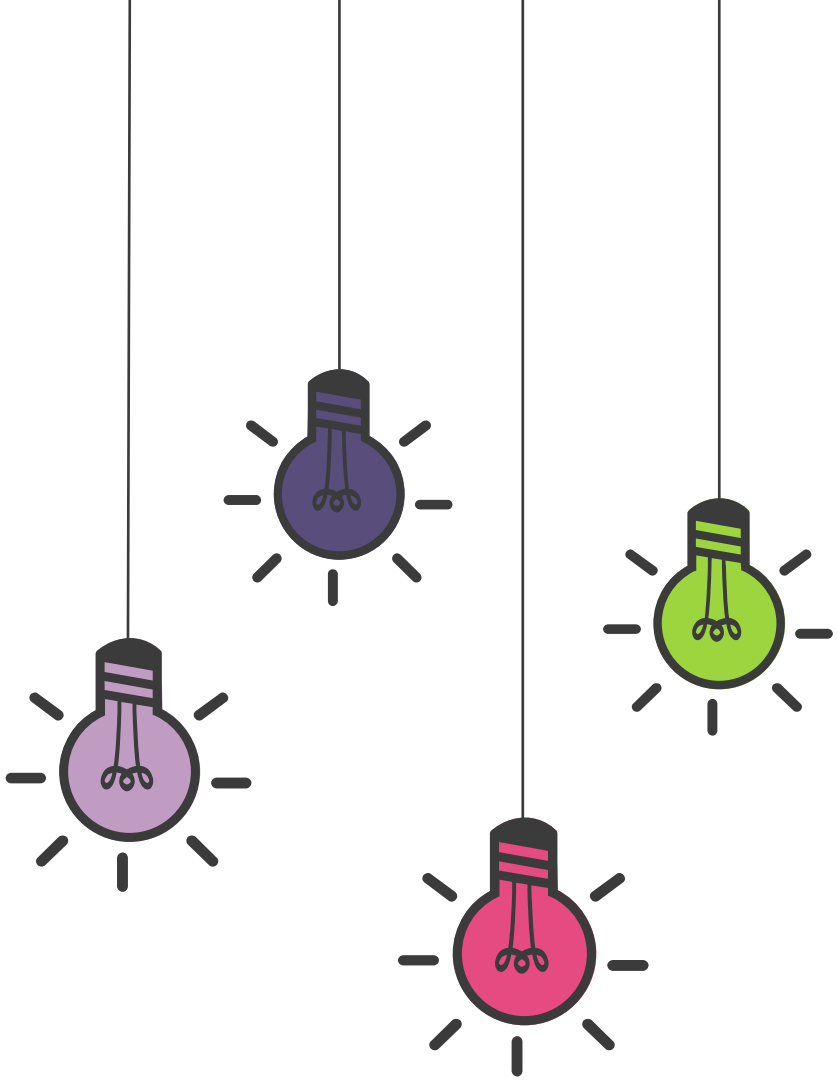
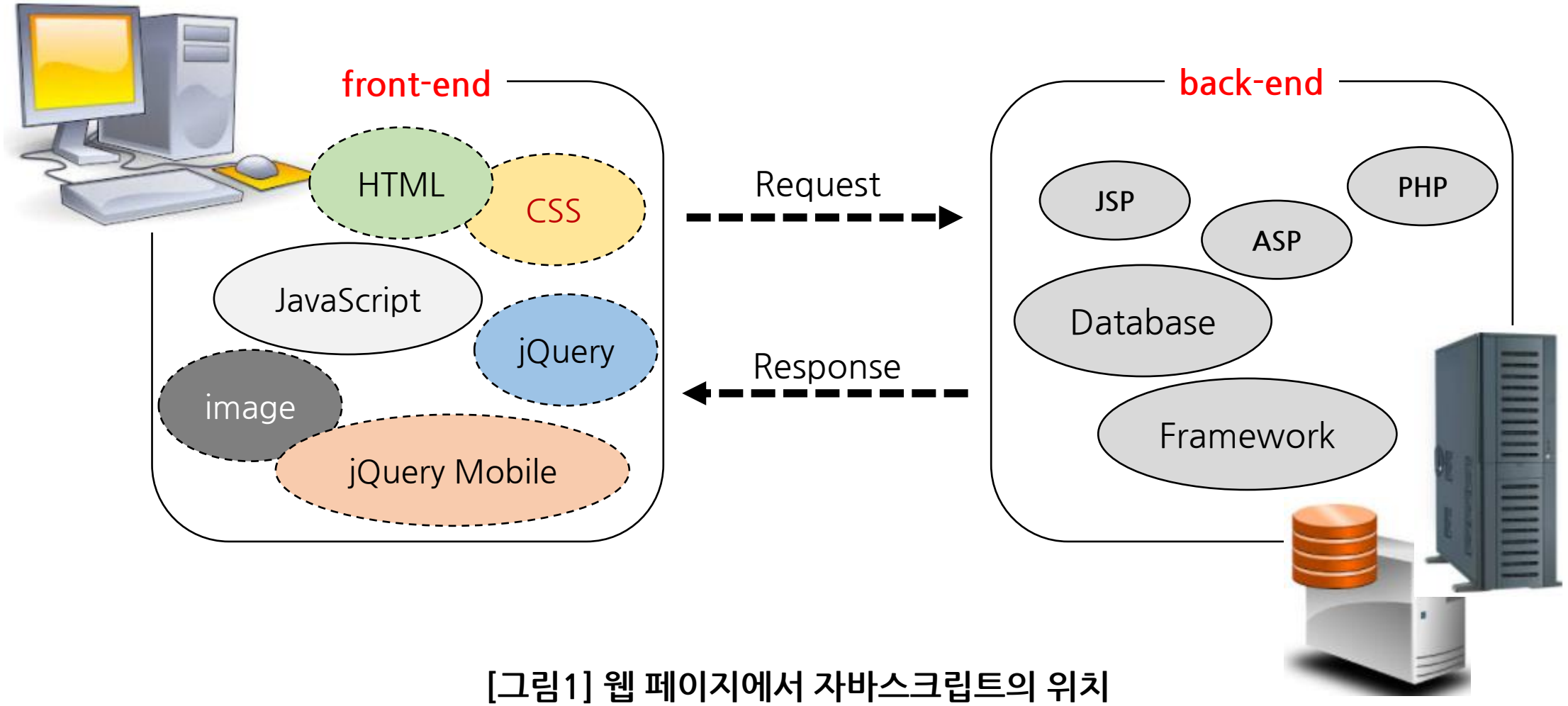


JavaScript/ES6

Javascript/ES6

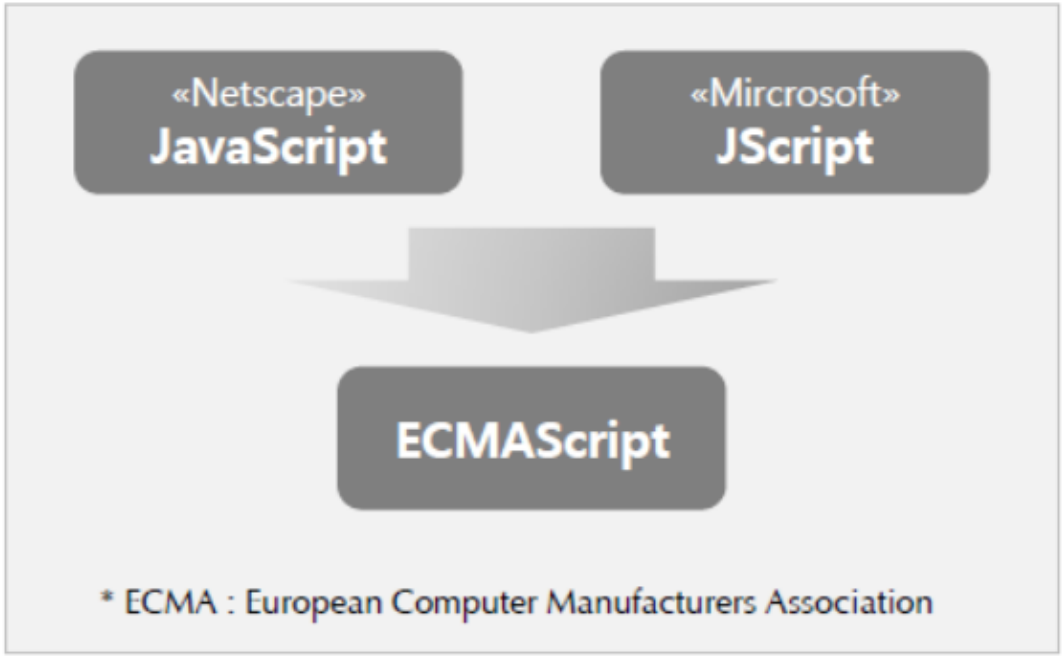
1. 자바스크립트(JavaScript) 란?

- 웹 브라우저 내에서 사용하는 객체 기반의 스크립트 프로그래밍 언어이다.
- JavaScript는 정적인 웹페이지에 애니메이션 처리, 계산 처리, 데이터 처리 등을 가능하게 한다.



[그림1] 웹 페이지에서 자바스크립트의 위치

- 1996년 3월 Netscape는 ‘Netscape Navigator 2.0’으로 JavaScript를 지원하기 시작했다.
- Microsoft사는 웹에 호환되는 JScript를 개발해서 1996년 8월에 출시하였다.
- Netscape는 JavaScript 표준화를 위해 기술 규격을 ECMA에 제출했다.



2. 자바스크립트 문법

1) 자바스크립트 선언

➢ 자바스크립트의 선언은 일반적으로 <head>와 <head> 사이에 작성한다.

[형식]

<script type="text/javascript">

실행문;

</script>

[작성 규칙]

1. 자바스크립트는 대소문자를 구분한다.
2. 한 라인을 끝내려면 원칙적으로 세미콜론(;)으로 닫아준다.

   그러나, Enter로 라인을 구분하기 때문에 생략이 가능하다.
3. 한 번에 2개 이상의 문장이 기술되면 세미콜론을 생략할 수 없다.
4. /* */ 또는 // 로 주석처리 한다.

[사용 예 1]

```
var str1="very"
var str2="good"
```

[사용 예 2]

```
var str1="very"; var str2="good";
```

- JavaScript를 삽입하는 방법은 3가지가 있다.
- 각각의 방법은 CSS와 마찬가지로 HTML 작성 상황에 따라 적절하게 선택하여 사용하면 된다.

1. HTML문서 안의 <SCRIPT>과 </SCRIPT> 안에 JavaScript 코드를 추가하는 방법이다.

<head> 태그 안에서 작성하는 방법

HTML

```
<html>
<head>
   <title>Insert title here</title>
   <script>
      document.write("Hello World");
   </style>
</head>
<body>
</body>
</html>
```

2. <BODY></BODY>안에서 작성할 수 있다.

<body> 태그 안에서 작성하는 방법

HTML

```
<html>
<head>
   <title>Insert title here</title>
</head>
<body>
   <script>
      document.write("Hello World");
   </style>
</body>
</html>
```

3. 확장자를 js로 하는 외부 파일을 불러서 사용할 수 있다.

외부 파일로 작성하는 방법

HTML

```
<html>
<head>
   <title>Insert title here</title>
   <script src="script.js"></script>
</head>
<body>
</body>
</html>
```

script.js

```
document.write("Hello World");
```

2) 자바스크립트 작성 방법

- var는 variable의 약자이며 생략이 가능하다.
- 변수명은 대. 소문자를 구별하고 영문, $, _ (밑줄 문자), 숫자를 포함할 수 있다.
- 하지만 변수명의 맨 앞에는 숫자가 올 수 없고,영문, _(밑줄 문자), $만 가능하다.

❖ 변수 선언 방법

	변수 선언	설명

맞음	var num=10;
	var num10=100;
	var $num=100;
	var _num=100;
		영문자와 숫자의 조합은 가능하다.
		변수명 맨 앞에 $나 _(밑줄문자)로 시작할 수 있다.

틀림	var 2num=100; var num%=100;	변수명 맨 앞에는 숫자가 올 수 없다. %(특수문자)는 포함할 수 없다.

❖ 변수에 저장되는 데이터 타입

데이터 타입	사용 방법
문자형(string)	var str="hello"; 또는 var str='hello';
숫자형(number)	var num=10; 또는 var num=50.4;
논리형(boolean)	var check=true; 또는 var check=false;
널형(null)	var str=null; 또는 var str=" ";

3) 연산자

❖ 산술 연산자

연산자	설명	연산자	설명
+	더하기	-	빼기
*	곱하기	/	나누기
%	나머지		

❖ 관계 연산자

연산자	설명	연산자	설명
A > B	A가 B보다 크다.	A >= B	A가 B보다 크거나 같다.
A < B	A가 B보다 작다.	A <= B	A가 B보다 작거나 같다.
A == B	A와 B는 같다.	A != B	A와 B는 같지 않다.
A === B	A와 B는 같다. (데이터 형의 비교)	A !== B	A와 B는 같지 않다. (데이터 형의 비교)

❖ 대입(할당) 연산자

연산자	사용 예	의미	설명
+	A=10	A=10	A에 10을 저장한다.
+=	A+=10	A=A+10	A에 10을 더해서 다시 A에 저장한다.
-=	A-=10	A=A-10	A에서 10을 빼서 다시 A에 저장한다.
*=	A*=10	A=A*10	A에 10을 곱해서 다시 A에 저장한다.
/=	A/=10	A=A/10	A를 10으로 나누어서 다시 A에 저장한다.
%=	A%=10	A=A*10	A를 10으로 나눈 나머지를 다시 A에 저장한다.

```
<script>
	var  a=10, b=3, c,d,e,f;
	c=a+b;
	d=a-b;
	e=a*b;
	f=a/b;
	document.write("두수의 합 : " + (a + b) + "<br>");
	document.write("두수의 차 : " + d + "<br>");
	document.write("두수의 곱 : " + e + "<br>");
	document.write("두수의 몫 : " + f + "<br><br>");
</script>
```

두수의 합 : 13
두수의 차 : 7
두수의 곱 : 30
두수의 몫 : 3.3333333333333335

[ex01.html] 산술연산자

```
<script>
	var a=5;
	rs=a==5;	document.write(rs + "<br>");
	rs=a>=5;	document.write(rs + "<br>");
	rs=a!=5;	document.write(rs + "<br>");
</script>
```

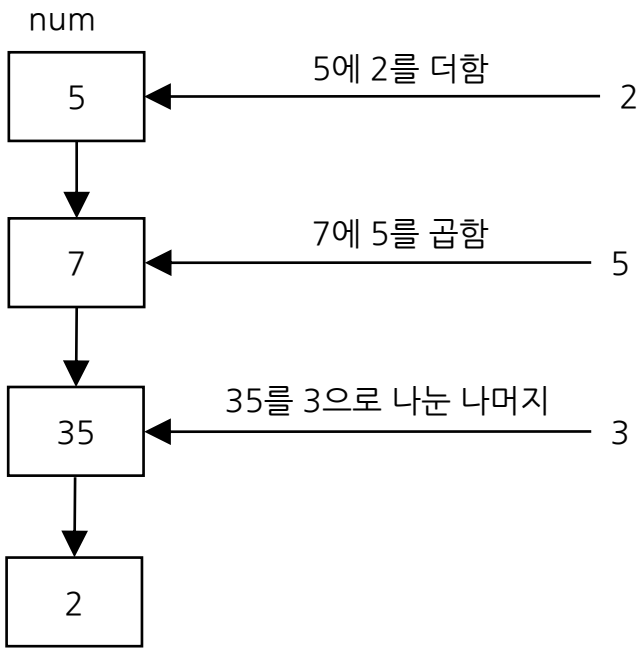
true
true
false

[ex02.html] 관계연산자

```
<script>
      var num=5;
      num+=2;              document.write(num + "<br>");
      num*=5;              document.write(num + "<br>");
     num%=3;               document.write(num + "<br>");
</script>
```

```
7
35
2
```

[ex03.html] 대입(할당)연산자



❖ 논리 연산자

연산자	의미	설명
A && B	AND(그리고)	A, B 둘 다 만족해야 참
A || B	OR(또는)	A, B 둘 중 하나만 만족해도 참
!A	NOT(논리부정)	A의 반대

❖ 증감 연산자

연산자	사용 예	설명
++	++A	A의 값을 1증가 시킨 후에 사용
	A++	A의 값을 사용한 후 1증가
--	--A	A의 값을 1감소 시킨 후에 사용
	A--	A의 값을 사용한 후 1 감소

❖ 삼항 연산자

연산자	설명
(조건식)? A:B;	조건식 결과가 참이면 A를 수행하고 거짓이면 B를 수행한다.

```
<script>
	var str="korea";
	document.write(str=="korea" || str=="KOREA");
	document.write("<br>");

	var score=57;
	document.write(score>=70 && score<90);
	document.write("<br>");

	var sign="true";
	document.write(!sign);
</script>
```

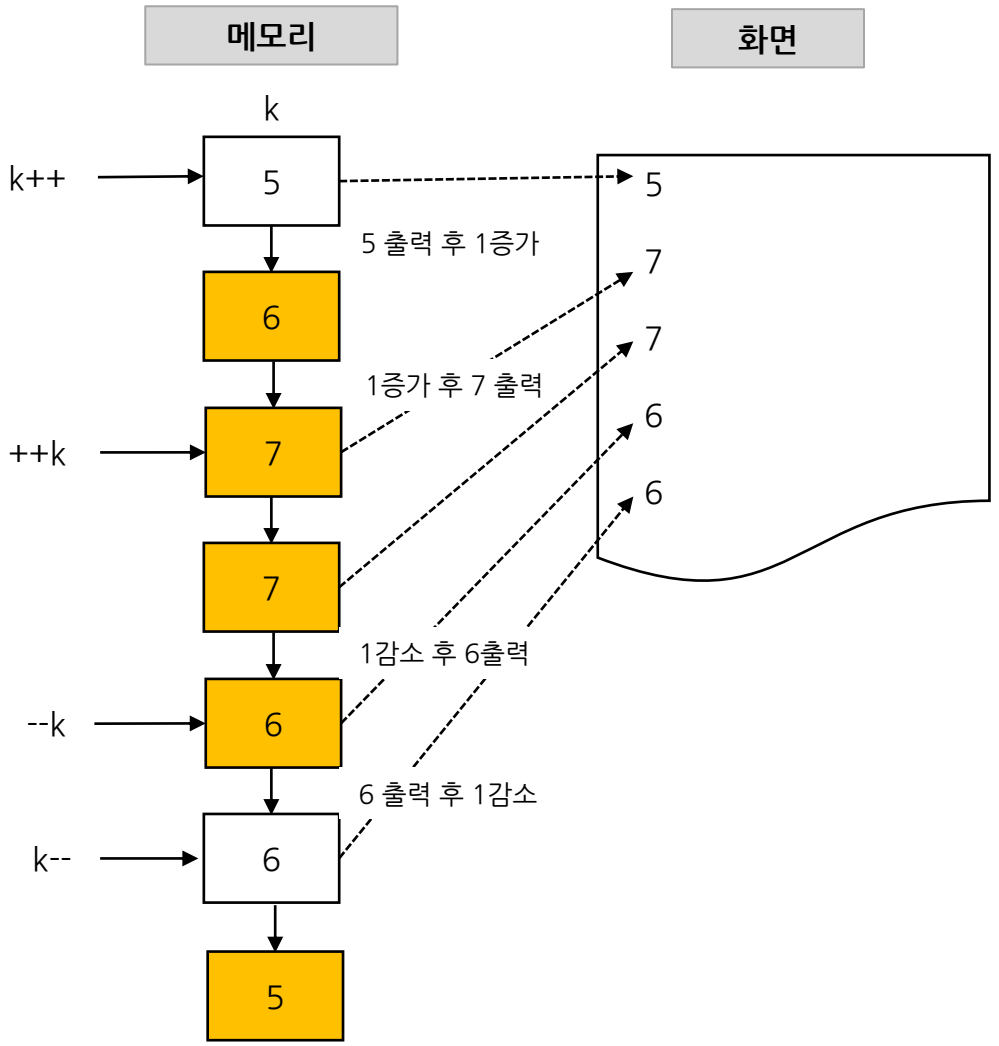
```
true
false
false
```

[ex04.html] 논리연산자

```
<script>
    var k=5;
    document.write(k++ + "<br>");
    document.write(++k + "<br>");
    document.write(k + "<br>");
    document.write(--k + "<br>");
    document.write(k-- + "<br>");
</script>
```

5
7
7
6
6

[ex05.html] 증감연산자



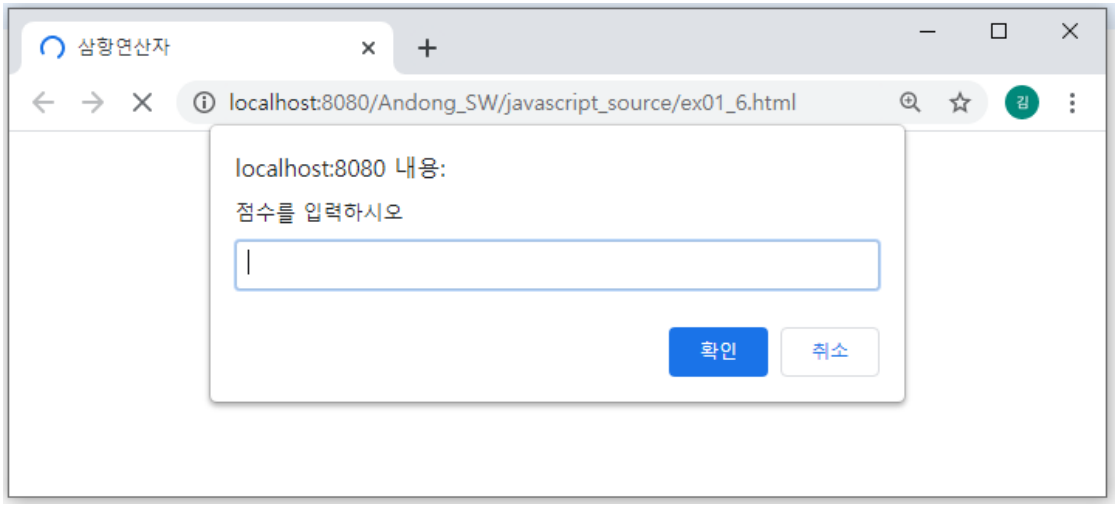
```
<script>
    var str1="korea";   var str2=" fighting";
    document.write(str1 + str2 + "<br><br>");

    var str="abc";  var num="123";
    document.write(str + num);
</script>
```

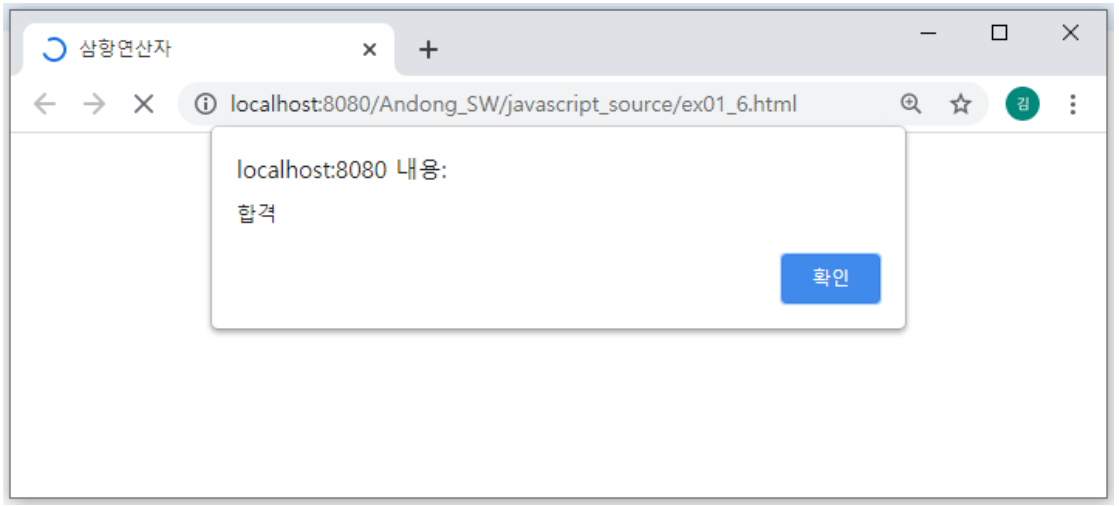
korea fighting

abc123

[ex06.html] 문자결합연산자

```
<script>
     var score=prompt("점수를 입력하시오","");
    (score>=60)?alert("합격") : alert("불합격");
</script>
```





[ex07.html] 삼항연산자

4) 제어문

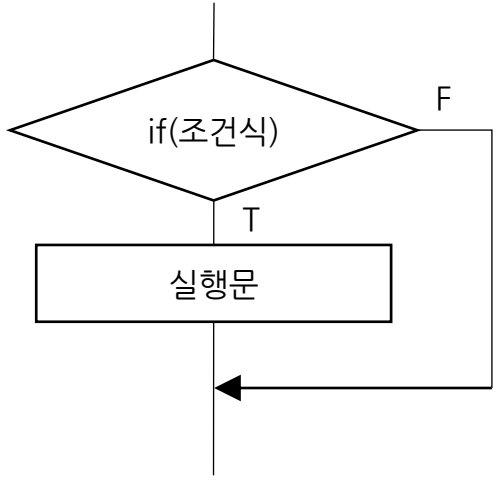
- 일반적인 프로그램의 진행 순서는 위에서 아래로 순차적으로 실행한다.
- 그러나, 상황에 따라서 조건이 만족할 경우에만 처리해야 하는 일이 생긴다.
- 또한 코드를 반복적으로 처리해야 하는 경우도 생긴다.
- 제어문의 종류에는 조건문, 선택문, 반복문이 있다.

❖ 조건문

- if(조건식)은 조건식이 참일 때 { } 안의 실행문을 수행한다.
- { } 안의 실행문이 한 문장인 경우에는 { }는 생략이 가능하다.
- 하지만, 두 문장 이상이면 { }은 생략할 수 없다.

(1) if

```
if(조건식)  {
    실행문;
}
```



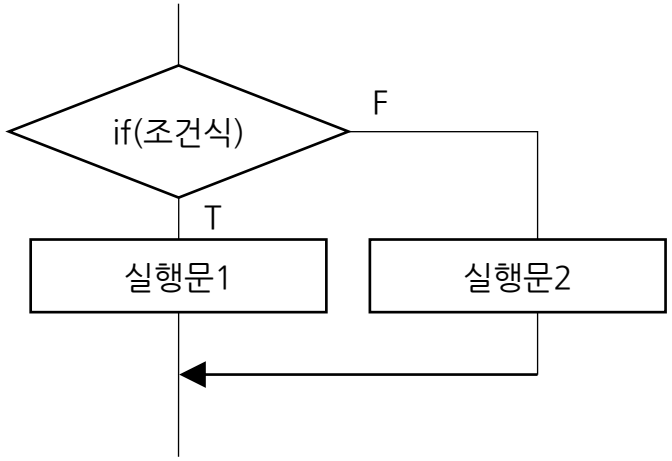
```
<script>
    var num=35;
    if(num%7==0)
    {
        document.write(num+"은(는) 7의 배수 입니다");
    }
</script>
```

35은(는) 7의 배수 입니다

[ex08.html] if 문

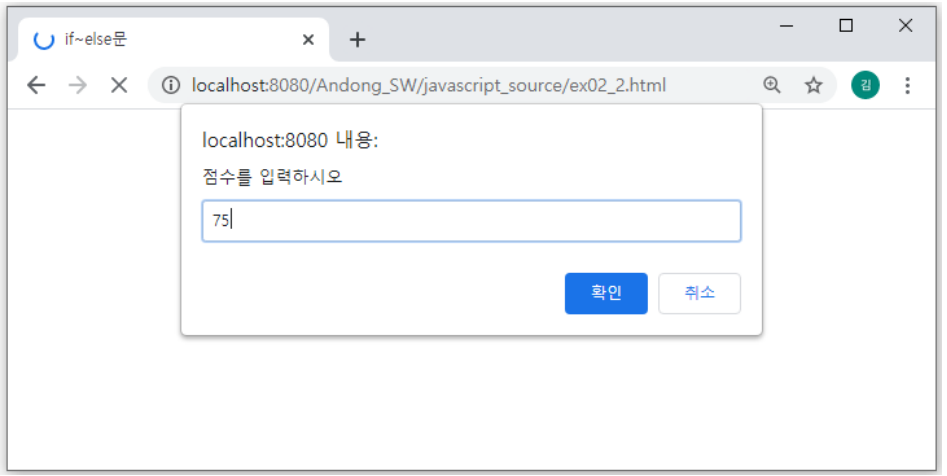
(2) if~else

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



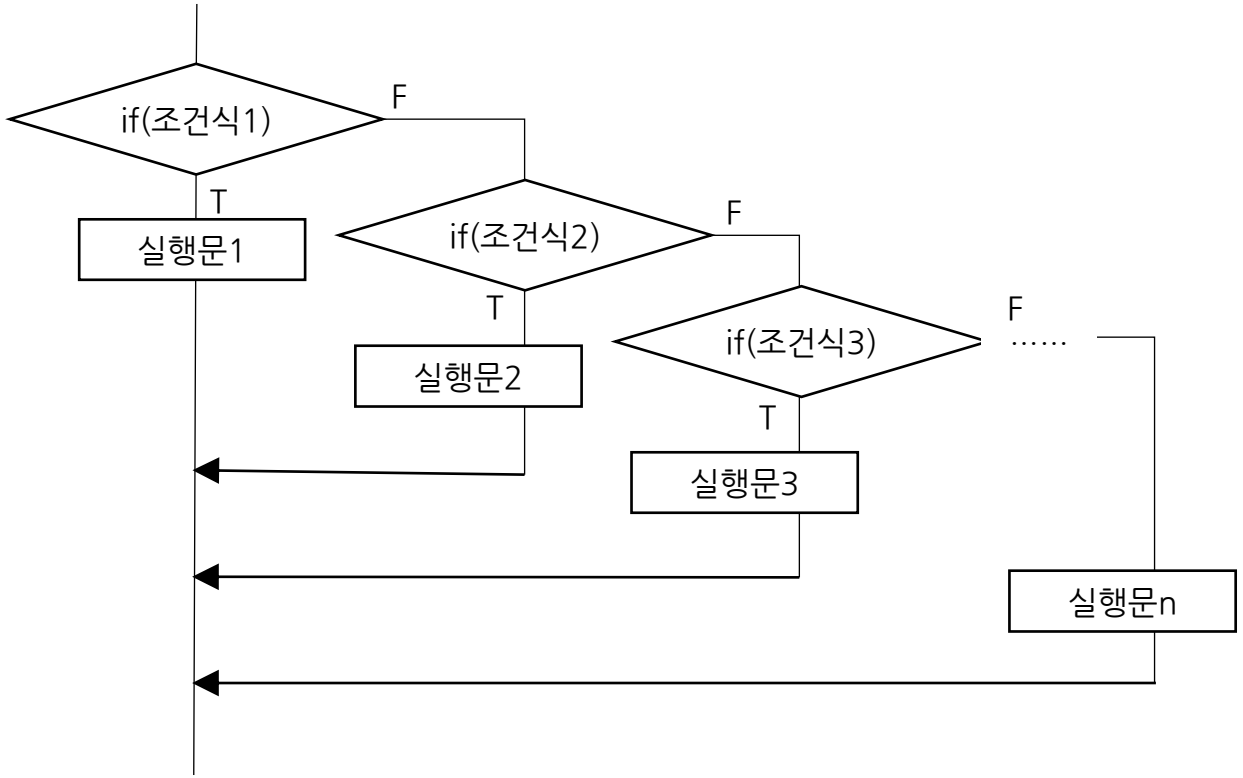

```
<script>
    var score=prompt("점수를 입력하시오","");
    if(score>=60){
        document.write("합격<br>");
        document.write("당신의 점수는 "+score+"점 입니다");
    }else{
        document.write("불합격<br>");
        document.write("당신의 점수는 "+score+"점 이며, 다음 기회에~");
    }
</script>
```

[ex09.html] if ~ else 문



(3) if~else if

```
if(조건식1){
    실행문1;
} else if(조건식2) {
    실행문2;
} else {
    실행문n;
}
```

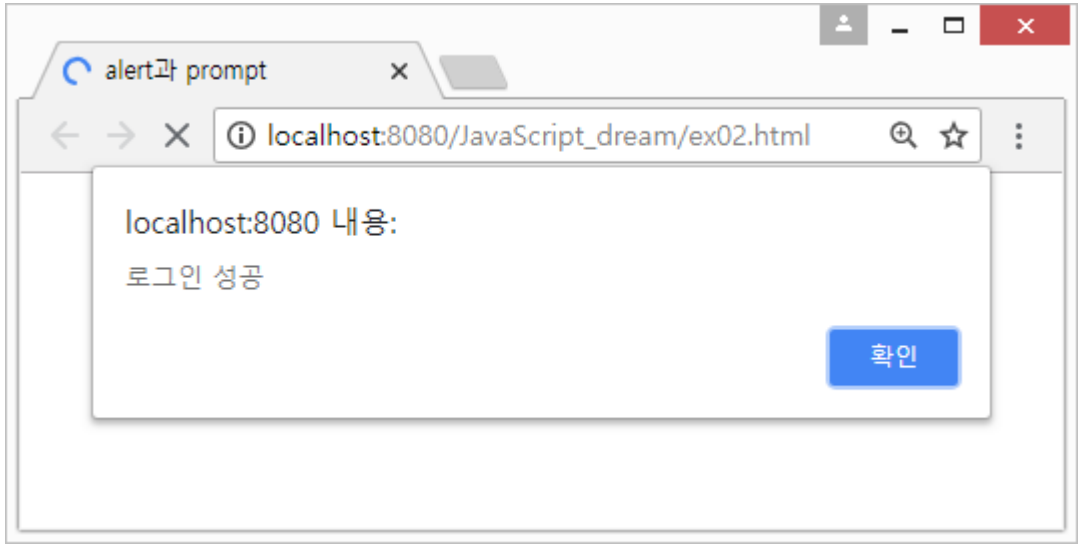


```
<script>
    var score=95;
    var grade;

    if(score>=90)           grade='A';
    else if(score>=80)      grade='B';
    else if(score>=70)      grade='C';
    else if(score>=60)      grade='D';
    else                    grade='F';

   document.write("나의 점수는 "+score+"점이고 학점은 " + grade +" 입니다");
</script>
```

나의 점수는 95점이고 학점은 A 입니다

[ex10.html] if ~ else if 문

```
<script>
    var id="admin";  var pw="1234";
    if(id=="admin" && pw=="1234")
        alert("로그인 성공");
    else
        alert("ID 또는 PW가 일치하지 않습니다");
</script>
```



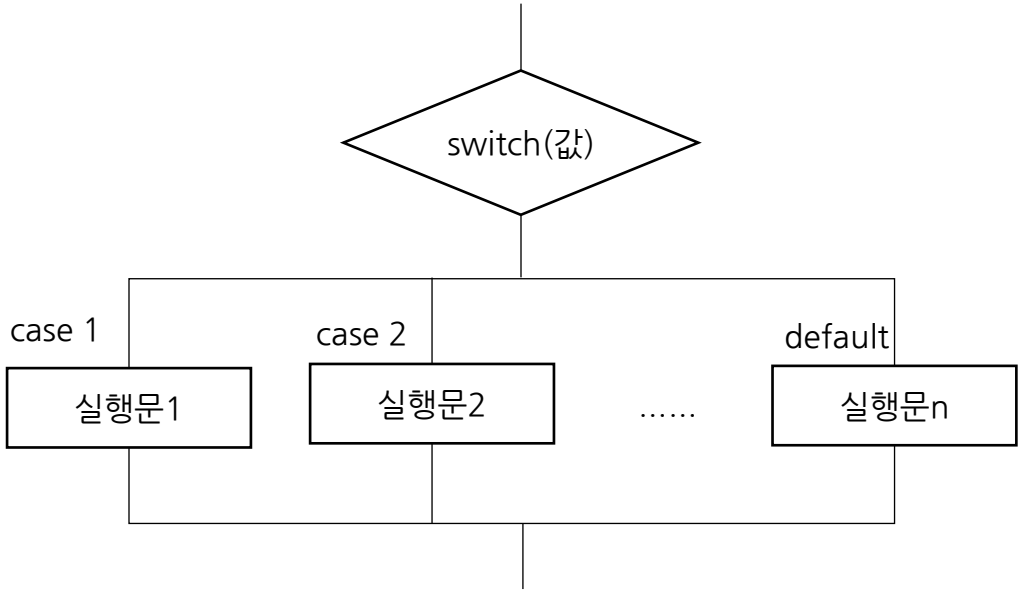
[ex11.html] alert

❖ 선택문

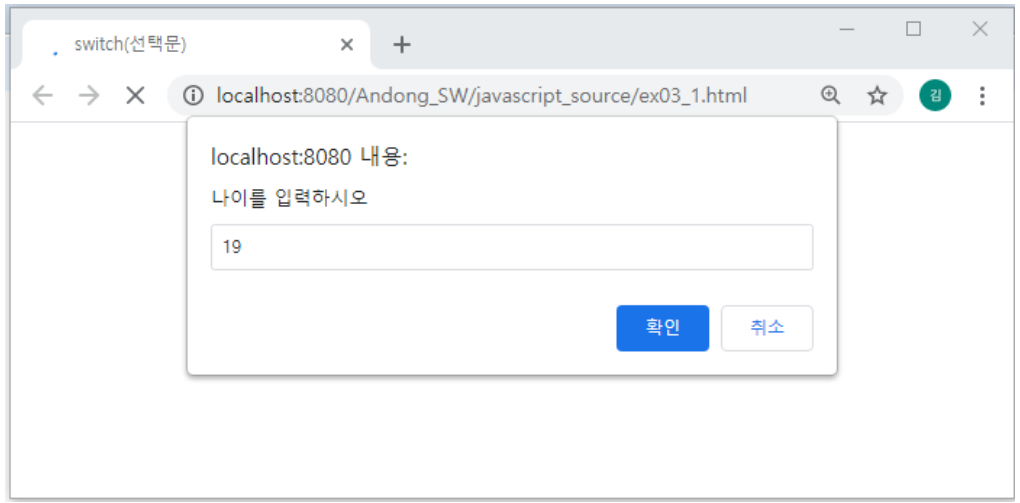
- 선택문은 저장된 데이터의 값이 일치하는 경우에 해당 실행문을 수행하는 제어문이다.
- 선택문에는 switch~case가 있다.

(4) switch

```
switch(값){
    case n1: 실행문1; break;
    case n2: 실행문2; break;
    case n3: 실행문3; break;
        :
    [default:] 실행문n;
}
```

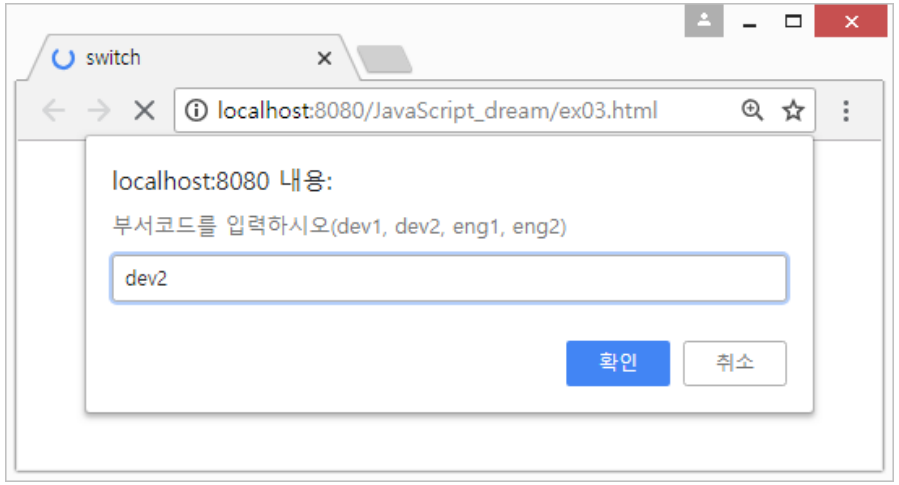


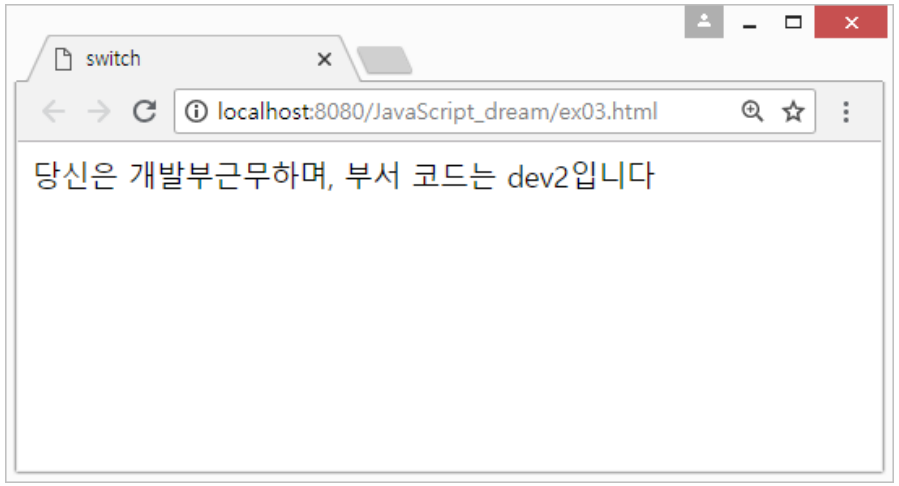
```
<script>
    var age=prompt("나이를 입력하시오","");
    switch(true)
    {
                case age>=60: alert("노년입니다");  break;
                case age>=35: alert("중년입니다");  break;
                case age>=20: alert("청년입니다");  break;
                default : alert("어린이 또는 청소년입니다");
     }
</script>
```



[ex12.html] switch 문

```
<script>
   var code=prompt("부서코드를 입력하시오(dev1, dev2, eng1, eng2)","");
   var dept;
   switch(code)
   {
      case "dev1":
      case "dev2": dept="개발부"; break;
      case "eng1":
      case "eng2": dept="기술부"; break;
   }
   document.write("당신은 " + dept+ "근무하며, 부서 코드는 "
    + code +"입니다");
</script>
```





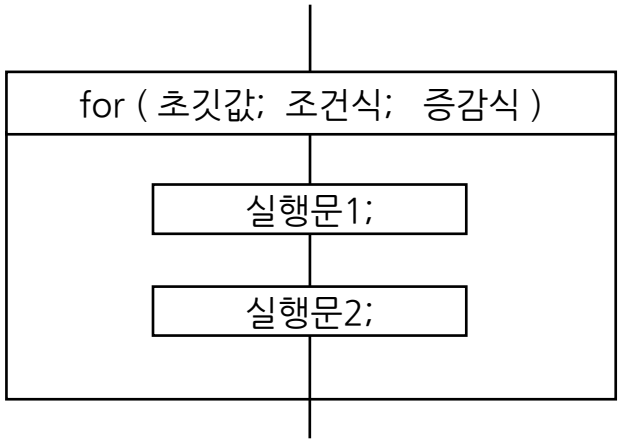
[ex13.html] switch 문

❖ 반복문

- 반복문은 조건식이 참(true)인 경우에 실행문을 반복적으로 수행하며 for, while이 있다.
- for문은 정해진 구간만큼 반복할 때 사용한다

(5) for

```
for(초깃값; 조건식; 증감식){
    실행문;
}
```



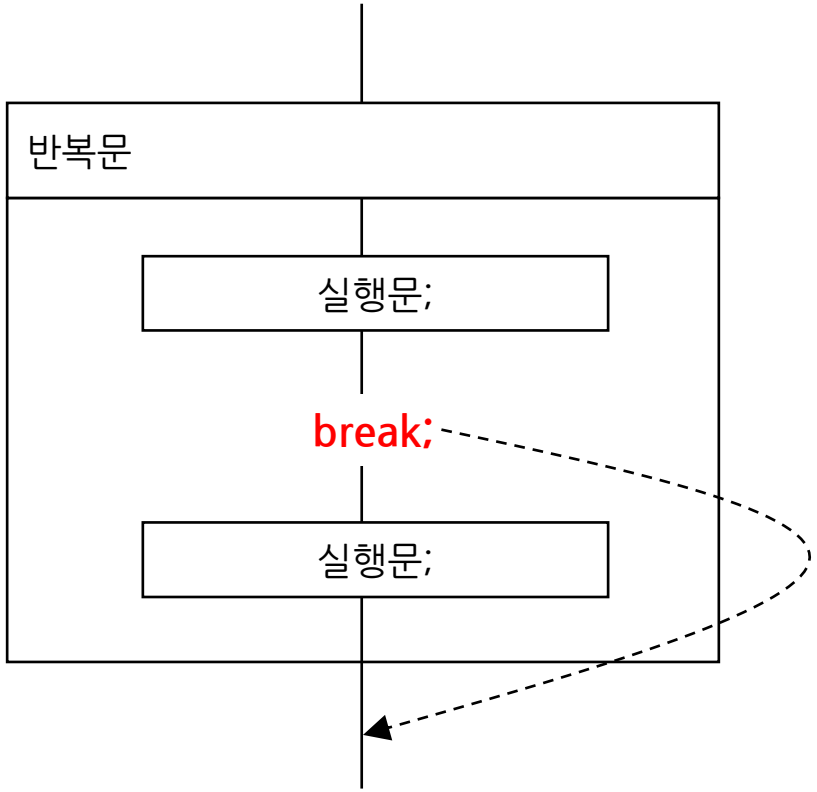

```
<script>
for(var a=1; a<=10; a++){
     document.write(a+" & nbsp; ");
}

for(var b=1; b<=6; b++){
    document.write("<h"+b+">Hello World<"+"/h"+b+">");
}
</script>
```

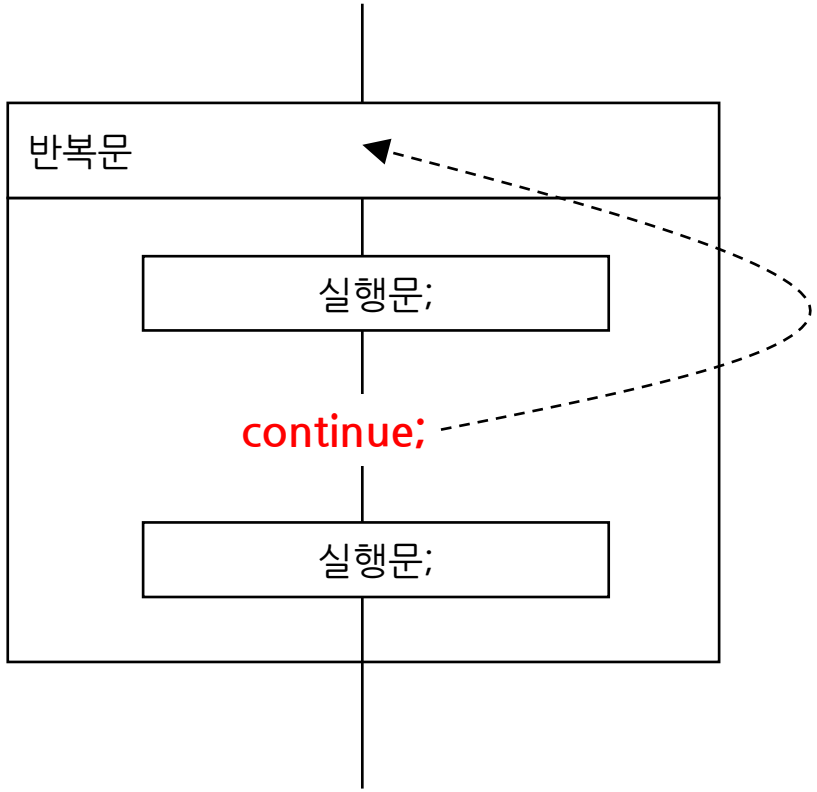
[ex14.html] for 문

(6) 블록 탈출

```
반복문 {
    break; 또는 continue;
}
```

- 반복문에서 break를 만나면 해당 반복문의 { }에서 빠져나와 반복 작업을 종료한다.
- continue를 만나면 다음 실행문을 수행하지 않고 다음 loop를 진행한다.





```
<script>
    for(var i=1; i<=10; i++)
    {
        if(i==5)
            break;
        document.write(i+"  ")
    }
    document.write("<br><br>");

    for(var j=1; j<=10; j++)
    {
        if(j==5)
            continue;
        document.write(j+"  ")
    }
</script>
```

```
1-100까지의 전체합: 5050
1-100까지의 홀수합: 2500
1-100까지의 짝수합: 2550
```

[ex15.html] for 문을 이용한 1-100까지의 합

```
<script>
    for(var i=1; i<=10; i++)
    {
        if(i==5)
            break;
        document.write(i+"  ")
    }
    document.write("<br><br>");

    for(var j=1; j<=10; j++)
    {
        if(j==5)
            continue;
        document.write(j+"  ")
    }
</script>
```

```
1  2  3  4

1  2  3  4  6  7  8  9  10
```

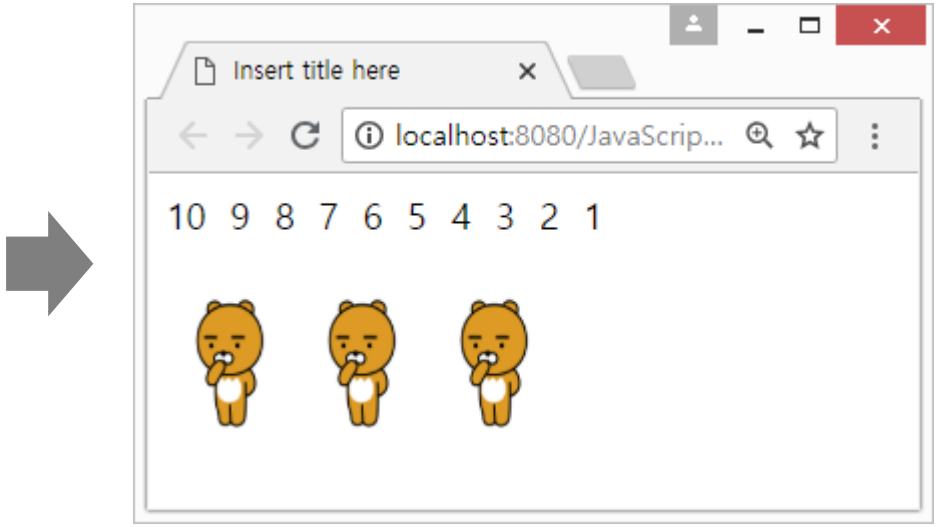
[ex16.html] continue와 break

(7) while

```
while(조건식){
    실행문;
}
```

- while문은 조건 반복문이다. for문과의 차이점은 while은 정해진 구간이 없다.
- 조건이 맞으면 무한 루프(loop)를 수행하기도 한다.
- 조건 반복문은 while, do~while 등 2가지 타입이 있다.

```
<script>
   var n=10;
   while(n >= 1)
   {
       document.write(n+"  ");
       --n;
   }
   document.write("<br><br>");
   var m=0;
   while(m < 3)
   {
       document.write("<img src='image/bear.gif' width='60px' height='70px'>");
       ++m;
   }
</script>
```



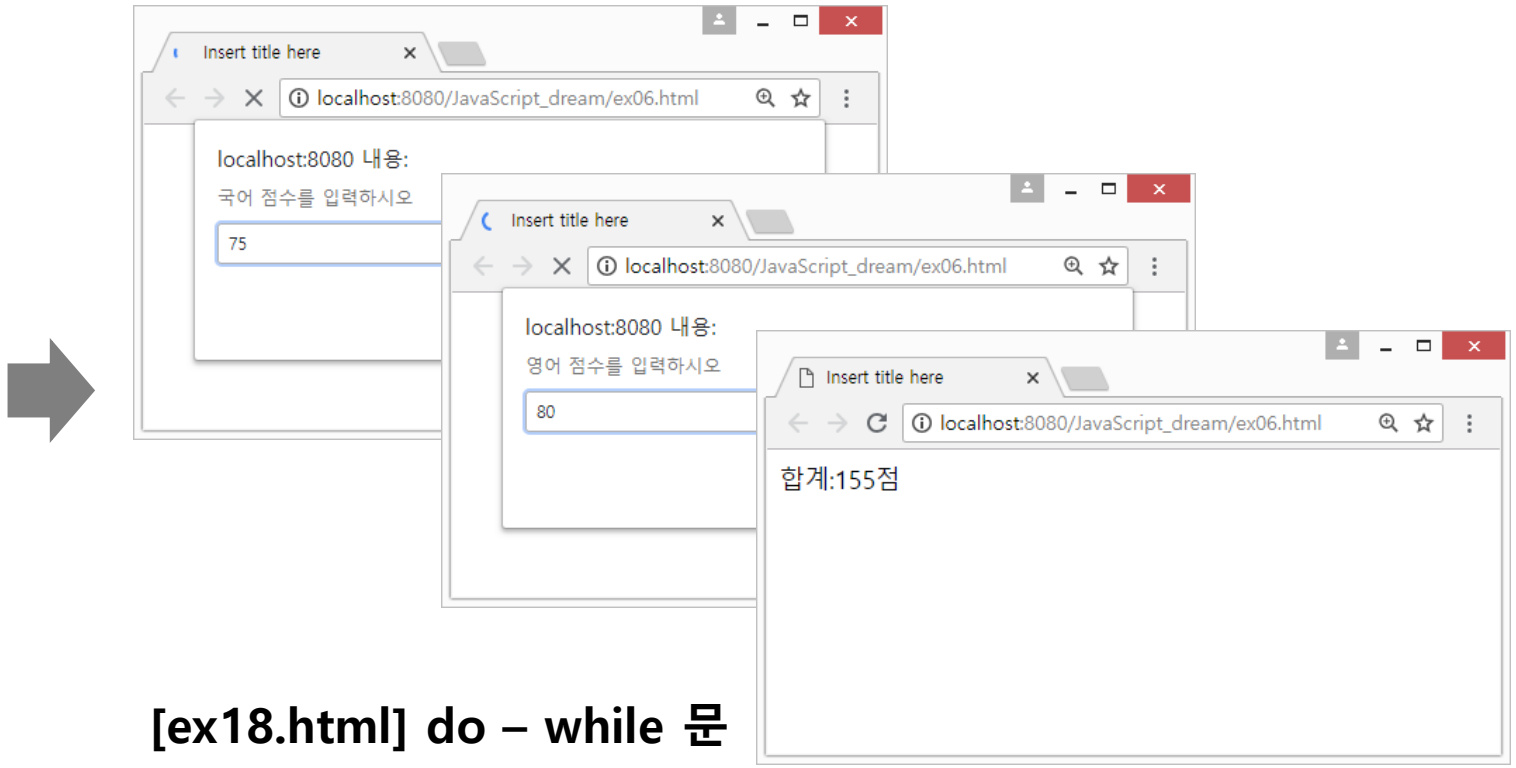
[ex17.html] while 문

(8) do~while

```
do{
    실행문;
} while(조건식);
```

- do~while문은 while문과 크게 다르지 않다.
- 먼저 do{ 안으로 들어가서 실행문을 수행한 후 조건을 비교한다.
- 조건이 참이면 다시 do{ 안으로 이동하여 반복처리합니다.
- while문과의 차이점은 do~while문은 조건식이 맞지 않아도 무조건 1번은 실행한다.

```
<script>
   do{
      kor=prompt("국어 점수를 입력하시오","");
      if(kor<0 || kor>100)
         alert("0-100사이의 숫자를 입력하시오");
   }while(kor<0 || kor>100);

   do{
      eng=prompt("영어 점수를 입력하시오","");
      if(eng<0 || eng>100)
         alert("0-100사이의 숫자를 입력하시오");
   }while(eng<0 || eng>100);

   sum=parseInt(kor)+parseInt(eng);
   document.write("합계:" +sum+"점");
</script>
```



[ex18.html] do – while 문

5) 함수(function)

관련 코드끼리 묶어서 분리하는 구조적 프로그램이다.

(1) 선언적 함수

[형식 1]

```
function 함수명(){
   실행문;
}
```

호출 함수명();

[형식 2]

```
function 함수명(매개 변수, 매개 변수, ….){
   return 리턴값;
}
```

호출 var 변수 = 함수명(매개 변수, 매개 변수, …);

(2) 익명 함수

[형식 1]

```
var 변수명 = function(){
   실행문;
}
```

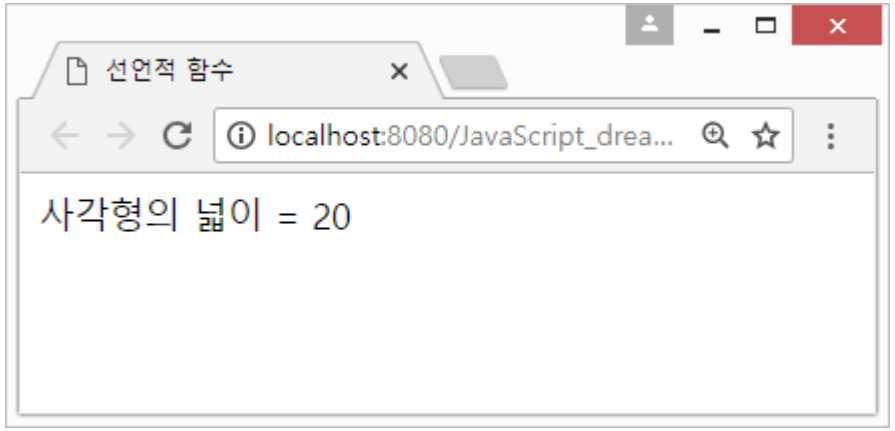
호출 변수명();

[형식 2]

```
var 변수명 = function(매개 변수, 매개 변수, ….){
   return 리턴값;
}
```

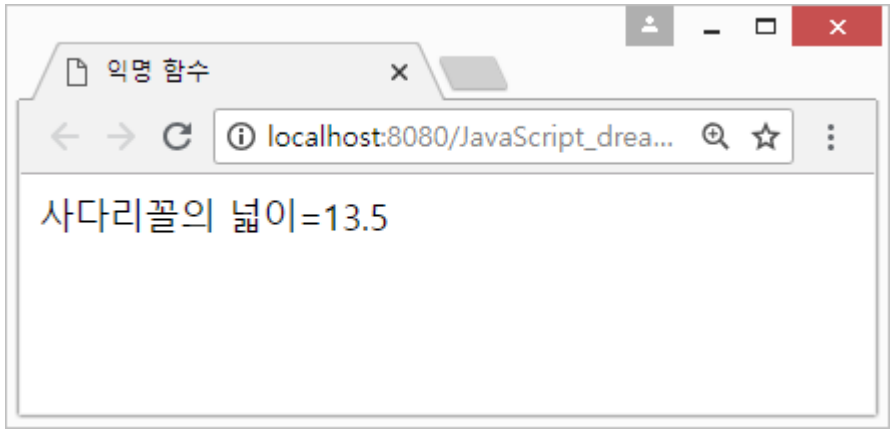
호출 var 변수 = 변수명(매개 변수, 매개 변수, …);

```
<script>
function getSquareSize(bottom, height){
    return bottom * height;
}
document.write("사각형의 넓이 = " + getSquareSize(4,5));
</script>
```



[ex19.html] 선언적인 함수

```
<script>
var getTrapezoid = function(bottom, top, height){
      return (bottom + top)* height / 2.0;
}
document.write("사다리꼴의 넓이=" + getTrapezoid(5,4,3));
</script>
```



[ex20.html] 익명함수

6) 내장함수

- 내장 함수는 JavaScript가 자체적으로 가지고 있는 함수이다.
- 네트워크 통신용 함수와 숫자와 관련된 함수가 있다.

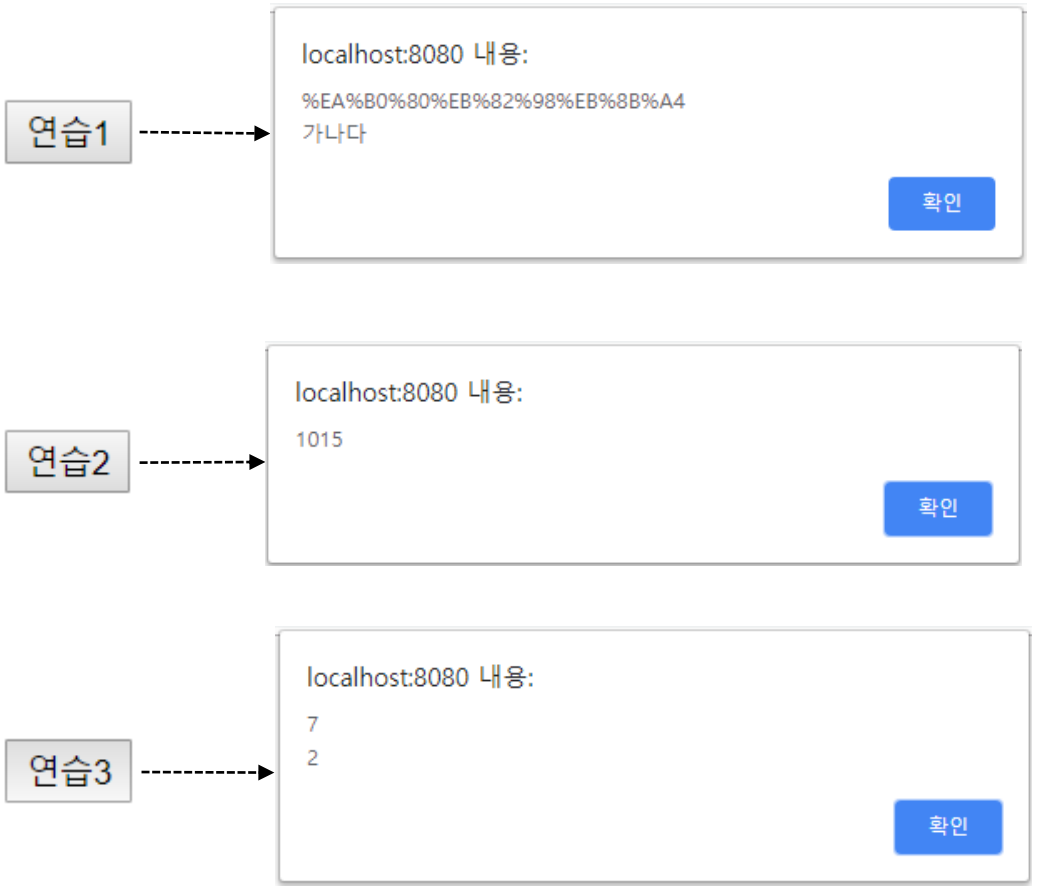
❖ 인코딩과 디코딩

내장 함수	설명
encodeURI( )	파라미터를 전달하는 URI 전체를 인코딩 할 때 사용하며 특수문자(:, ;, /, =, ?, & 등)를 제외한 문자만 인코딩합니다. 주로 인터넷 주소를 인코딩할 때 사용한다 .
decodeURI( )	encodeURI( )로 인코딩된 데이터를 다시 되돌린다.

❖ 기타 내장 함수

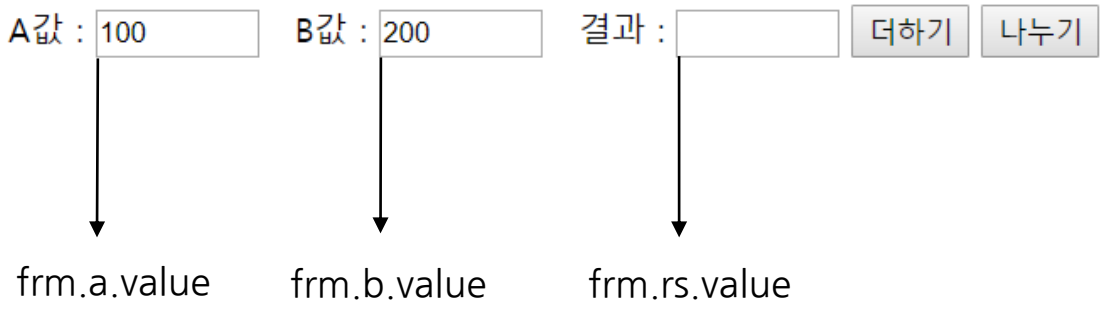
내장 함수	설명
isNaN( )	NaN은 "Not a Number"의 약자이다. 숫자가 아닌 문자가 포함되면 true를 반환한다.
isFinite( )	값이 유한수인지를 판단을 한다.
parseInt( )	문자를 정수형으로 변환한다.
parseFloat( )	문자를 실수형으로 변환한다.
eval()	문자로 된 수식을 JavaScript의 수식으로 인식하여 실행시키고 결과를 변환한다.
Number( )	문자를 숫자형으로 변환한다.
String( )	숫자를 문자형으로 변환한다.

```
<script>
    function ex1(){
        alert(encodeURI('가나다')+"₩n“
            + decodeURI('%EA%B0%80%EB%82%98%EB%8B%A4'));
    }
    function ex2(){
        var x=10, y=15;
        alert(String(x)+String(y));
    }
    function ex3(){
        var ob1=eval("var num=5+2");
        var ob2=eval("({'a':1, 'b':2, 'c':3})");
        alert(num +"₩n" + ob2.b);
    }
</script>
```



[ex21.html] encode와 decode

```
<script>
function plusfunc(){
		var a=parseInt(frm.a.value);
		var b=parseInt(frm.b.value);

		if(isNaN(a) || isNaN(b)){
			alert("a값 또는 b값은 숫자가 아닙니다");
			frm.a.value="";
			frm.b.value="";
			frm.a.focus();
			return;
		}
		frm.rs.value = a+b;
}
function divifunc(){
		var a=Number(frm.a.value);
		var b=Number(frm.b.value);
		rs=a/b;

		if(isFinite(rs)==false){
			alert("0으로 나눌수 없습니다");
			frm.a.value="";
			frm.b.value="";
			frm.a.focus();
			return;
		}
		frm.rs.value=rs;
}
</script>
```



[ex22.html] 숫자로 변환한 두수의 연산

1) 표준 객체

- 객체 란, 데이터와 데이터를 처리하는 모든 기술의 조합이다
- 표준 객체(Standard Built-in Object)란 JavaScript에서 자주 사용하는 Number, String, Array, Date, Math 등의 내장 객체를 말하며, 기본 자료형을 제외하고는 배열, 함수 등 모두가 객체이다. .

❖ Number 객체

- Number 객체는 숫자를 다루는 객체이다.
- Number 객체는 new 연산자로 객체를 생성할 수도 있다.

[형식]

var 변수명 = new Number(숫자); 또는 var 변수명=숫자;

```
<script>
    var a=15;
    var b=new Number(15);
    document.write("a와 b를 더하면: " + (a.toString()+b.toString()));
    document.write("<br>15의 2진수 표현: " + a.toString(2));
    document.write("<br>a와 b의 타입: " + typeof a+"  "+typeof b);
    document.write("<br>a와 b의 값 비교: " + (a==b));
    document.write("<br>a와 b의 타입 비교: "+ (a===b));
</script>
```

```
a와 b를 더하면: 1515
15의 2진수 표현: 1111
a와 b의 타입: number  object
a와 b의 값 비교: true
a와 b의 타입 비교: false
```

[ex23.html] Number객체

❖ String 객체

- String 객체는 문자열을 다루는 객체이다.
- 문자열은 단일 따옴표(' ') 또는 이중 따옴표(" ") 안에 작성한다.

[String 속성과 문자 관련 메서드]

메서드		설명
속성	length	문자열의 길이를 구한다.
문자	big( )/small( )	글자를 한 단계 크게/작게 설정한다.
	blink( )	글자를 깜박이게 설정한다.
	bold( )	글자를 굵게 설정한다.
	sub( )/sup( )	아래 첨자 / 위 첨자로 설정한다.
	fontsize(크기)	글자의 크기를 설정한다.(1-7까지)
	fontcolor(색상)	글자의 색상을 설정한다.
	toLowerCase( )	글자를 소문자로 변경한다.
	toUpperCase( )	글자를 대문자로 변경한다.

[String 문자열 관련 메서드]

메서드	설명
charAt(n)	인덱스 값에 해당 n의 위치를 반환한다.
indexOf("문자열")	처음부터 시작해서 최초로 만나는 "문자열"의 위치를 반환한다.
lastIndexOf("문자열")	끝에서부터 시작해서 최초로 만나는 "문자열"의 위치를 반환한다.
substring(n1, n2)	n1에서 (n2-1) 사이의 문자열을 반환한다.
slice(s, e)	s부터 (e-1)의 문자열을 분리한다.
substr(s, 길이)	s부터 길이만큼 문자열을 추출한다.
concat("문자열")	두 개의 문자열을 연결한다.
split("문자", 개수)	문자를 기준으로 개수만큼 분리한다.
replace(s1, s2)	문자열 중 s1을 s2로 치환한다.

```
<script>
    var data=new String("javascript test");
    document.write(data+"<br>");

    document.write(data.length+"<br>");
    document.write(data.toLowerCase()+"<br>");
    document.write(data.bold()+"<br>");
    document.write("국립안동대학교".link('http://http://www.andong.ac.kr/main/index.do')+"<br><br>");

    document.write(data.charAt(0)+"<br>");
    document.write(data.substring(1,3)+"<br>");
    document.write(data.substring(4)+"<br>");
    document.write(data.replace("test", "sample")+"<br>");
</script>
```

javascript test
15
javascript test
javascript test
국립안동대학교

j
av
script test
javascript sample

[ex24.html] 문자열 함수

❖ Array 객체

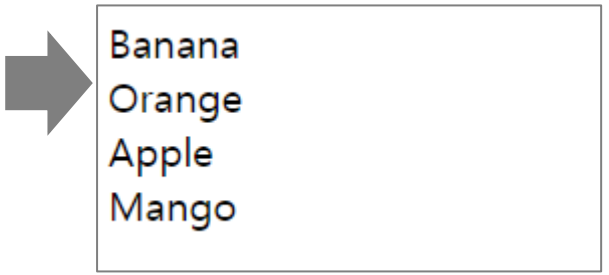
- 배열은 하나의 객체에 여러 개의 데이터 값을 저장할 때 사용한다.

[형식]

var 변수명 = [값1, 값2, 값3……값n];

메서드	설명
sort( )	배열 값들을 오름차순으로 정렬한.
reverse( )	배열 값들을 역순으로 바꾼다.
concat(array)	두 개의 배열을 합하여 하나의 배열로 만든다.

```
<script>
    var fruits=["Banana", "Orange", "Apple", "Mango"];

    var len=fruits.length;
    var text="";
    for (i = 0; i < len; i++) {
                text += fruits[i] + "<br>";
    }
   document.write(text);
</script>
```



[ex25.html] 배열객체

- 배열 객체를 선언하는 방식은 new 키워드를 사용해서 객체 선언과 동시에 값을 인자로 추가한다.

[형식]

var 변수명 = new Array (값1, 값2, 값3……값n);

```
<script>
    var fruits=new Array("Banana", "Orange", "Apple", "Mango");
    fruits.sort();
    document.write(fruits);
    document.write("<br>");

    fruits.reverse();
    document.write(fruits);
</script>
```

Apple,Banana,Mango,Orange
Orange,Mango,Banana,Apple

[ex26.html] 배열객체

❖ Date 객체

- Date 객체는 날짜와 시간에 대한 정보를 제공한다.
- 객체 생성 시 날짜를 지정하지 않으면 시스템에 설정된 날짜와 시간을 제공한다.

[형식]

var 변수명 = new Date(); 또는 var 변수=new Date(년, 월, 일, 시, 분, 초);

메서드	설명
setFullYear( ) / getFullYear( )	연도만 설정하거나 반환한다.
setMonth( ) / getMonth( )	월만 설정하거나 반환한다
setDate( ) / getDate( )	일(월 기준)을 설정하거나 반환한다.
setDay( ) / getDay( )	일(주 기준)을 설정하거나 반환한다.
setHour( ) / getHour( )	시간을 설정하거나 반환한다.

날짜	범위	시간	범위
연도	1900년(00) ~ 1999년(99)	시	0시(0) ~ 23시(23)
월	1월(0) ~ 12월(11)	본	0분(0) ~ 59분(59)
일	1일(1) ~ 31일(31)	초	0초(0) ~ 59초(59)

```
<script>
   var days=new Array("일","월","화","수","목","금","토");
   var date=new Date();
   document.write("오늘은 " +date.getFullYear()+"년"
                           +(date.getMonth()+1)+"월"
                           +date.getDate()+"일 "
                           +days[date.getDay()]+"요일 입니다.<br>");

   document.write("현재시간은 "+date.getHours()+":"
                              +date.getMinutes()+":"
                              +date.getSeconds()+"입니다.");
</script>
```

오늘은 2021년4월1일 목요일 입니다.
현재시간은 10:21:0입니다.

[ex27.html] 날짜객체

❖ Math 객체

- Math 객체는 수학에서 자주 사용하는 메서드를 미리 정의해 놓은 객체이다.
- 최댓값/최솟값을 구하거나 거듭제곱, 제곱근 등의 특정 공식을 사용할 수 있다.

메서드	설명
max(n1, n2, …)	가장 큰 값을 반환한다.
min(n1, n2, …)	가장 작은 값을 반환한다.
round(n)	반올림 값을 반환한다.
ceil(n)	올림 값을 반환한다.
floor(n)	내림 값을 반환한다.
abs(n)	절댓값을 반환한다.
random(n)	0~1사이의 임의의 수를 반환한다.
pow()	숫자의 거듭제곱을 계산하여 반환한다.

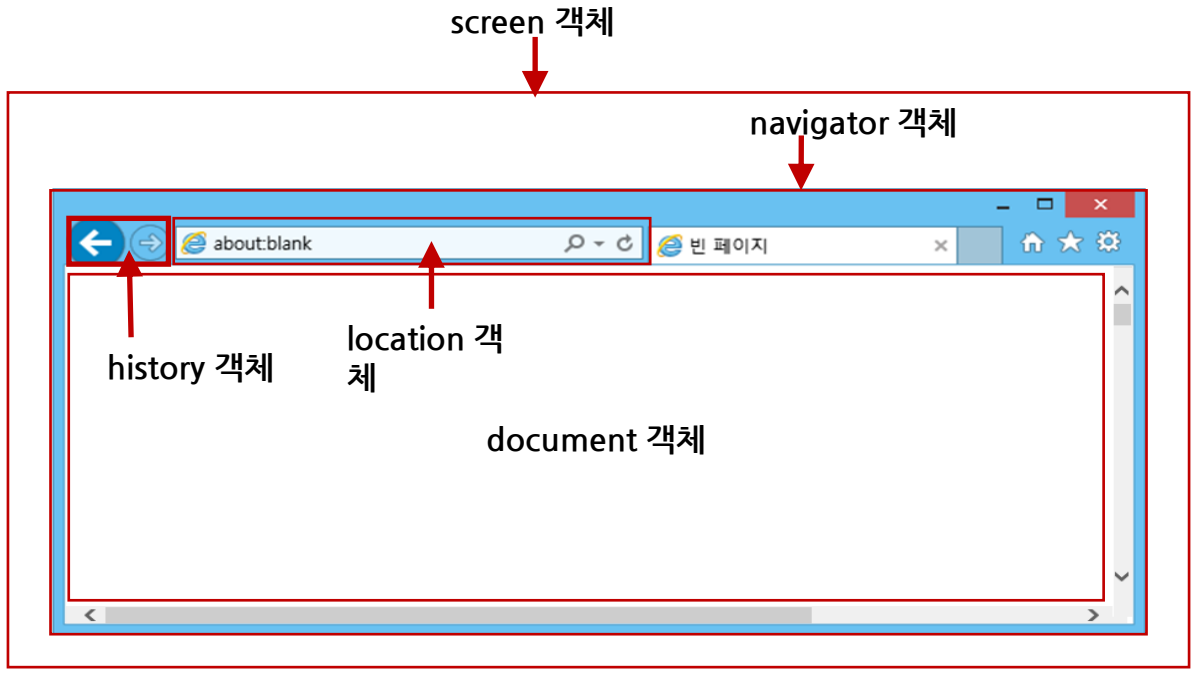
```
<script>
    document.write(Math.pow(8, 2)+"<br>");
    document.write(Math.random()+"<br>");
    document.write(Math.floor(4.7)+"<br>");
    document.write(Math.ceil(4.4)+"<br>");
    document.write(Math.round(4.7)+"<br>");
    document.write(Math.abs(-4.4)+"<br>");
    document.write(Math.sqrt(64)+"<br>");
</script>
```

```
64
0.20956530818219044
4
5
5
4.4
8
```

[ex28.html] 수학객체

2) 브라우저 객체모델 (BOM)

브라우저 객체 모델(Browser Object Model)은 웹 브라우저와 관련된 모든 객체들의 집합이다.



객체	설명
location 객체	브라우저 객체 모델의 종류로는 URL의 정보를 제공
navigator 객체	현재 실행 중인 브라우저의 정보를 제공
history 객체	방문 기록에 대한 정보를 제공
screen 객체	모니터의 정보를 제공
document 객체	문서에 대한 정보를 제공

❖ Window 객체

- window 객체는 브라우저 객체 모델의 최상위 객체이다.
- window의 형태와 위치를 변경할 수 있는 메서드이다.
- window는 생략이 가능하다.

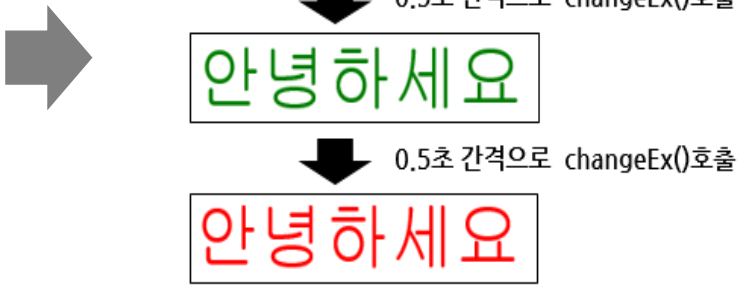
[window객체 메서드]

메서드	설명
open( )	새로운 윈도우 객체를 생성한다.
alert( )	경고창을 띄운다.
prompt( )	입력창을 띄운다.
confirm( )	확인/취소 창을 띄운다.
setInterval( )	일정시간 마다 반복 실행한다.
setTimeout( )	시간을 지연시킨 후 실행문을 수행한다.

[window객체 메서드]

메서드	설명
`moveBy(x, y)`	윈도우를 현재 위치에서 상대적 위치로 이동한다.
`moveTo(x, y)`	윈도우를 현재 위치와 상관없이 절대적 위치로 이동한다.
`resizeBy(x, y)`	윈도우의 화면을 현재 크기에서 (x, y) 만큼 증가시킨다. (상대적)
`resizeTo(x, y)`	윈도우의 화면을 (x, y) 크기로 한다. (절대적)
`scrollBy(x, y)`	윈도우 스크롤의 위치를 상대 위치로 이동한다.
`scrollTo(x, y)`	윈도우 스크롤의 위치를 절대 위치로 이동한다.
`focus( )`	윈도우에 초점을 맞춘다.
`blur( )`	윈도우에 초점을 제거한다.
`close( )`	윈도우를 닫는다.

```
<script>
   var n=0;
   function changeEx()
   {
                f=document.getElementById("fm");
                n++;
                switch(n)
                {
                                case 1: f.style.color="red";            break;
                                case 2: f.style.color="green";          break;
                                case 3: f.style.color="blue";  n=0;    break;
                }
                window.setTimeout("changeEx()", 500) //호출함수, 호출주기
   }
</script>

<body onload="changeEx()">
                <p id="fm">안녕하세요</p>
</body>
```



[ex29.html] 5초 간격으로 색상변경

❖ location 객체

- location 객체는 URL과 관련된 인터넷의 주소와 관련된 속성이다.
- 프로토콜, 호스트 이름, 문서의 주소 등의 정보가 있다.

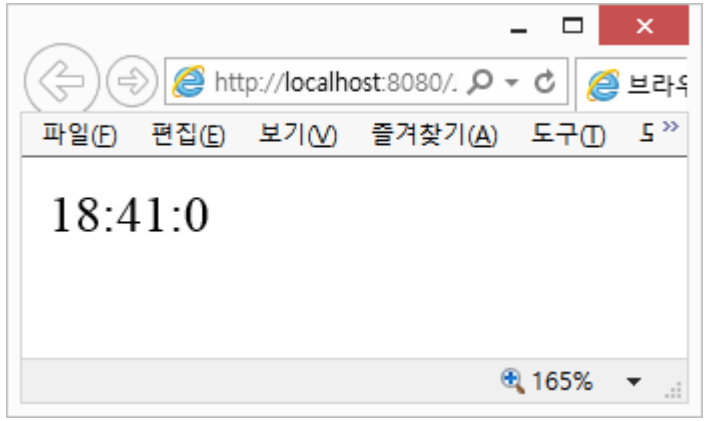
[location 객체 속성]

속성	사용 방법	설명
href	localhost:80/index.html	문서의 url 주소를 반환한다.
host	localhost:80	호스트 이름과 포트 번호를 반환한다.
hostname	localhost	호스트 이름을 반환한다.
port	80	포트 번호를 반환한다.
pathname	/docs/index.html	파일 경로를 반환한다.
hash	http://localhost:80/#test	앵커의 이름이다.
search	?param=100	요청 값(쿼리)을 반환한다.
protocol	http://	프로토콜을 반환한다.

[location 객체 메서드]

메서드	설명
reload( )	화면을 "새로 고침" 한다.
replace(link)	현재 페이지를 새로운 페이지로 바꾼다.

```
<script>
    var t=new Date();
    document.write( t.getHours()+":"+ t.getMinutes()+":"+ t.getSeconds());
    setInterval(function(){
            location.reload();
    },1000);
</script>
```



[ex30.html] 디지털 시계

❖ navigator 객체

- navigator 객체는 웹 브라우저에 대한 정보를 제공하는 객체이다.
- 브라우저의 종류나 버전 정보를 파악하기 위해 사용한다.

[navigator 객체 속성]

메서드	설명
appCodeName	브라우저의 코드명을 반환한다.
appName	브라우저의 이름을 반환한다.
language	브라우저의 사용 언어를 반환한다.
product	브라우저의 사용 엔진이름을 반환한다.
platform	사용중인 운영체제의 시스템 환경 정보를 반환한다.

❖ history 객체

- 인터넷 방문 기록에 대한 정보를 제공하는 객체이다.

[history 객체 속성]

메서드	설명
go(숫자)	숫자만큼 다음 또는 이전 페이지로 이동한다.
back()	이전 페이지로 이동한다.
forward()	다음 페이지로 이동한다.
length	방문 기록에 저장된 목록의 개수를 반환한다.

❖ screen객체

- screen 객체는 현재 화면의 해상도나 색상, 화면의 크기 정보 등의 속성을 제공하는 객체이다.

[screen 객체 속성]

속성	설명
width	전체 화면의 너비이다.
height	전체 화면의 높이이다.
availWidth	실제 화면에서 사용 가능한 화면의 너비이다.
availHeight	실제 화면에서 사용 가능한 화면의 높이이다.
colorDepth	사용 가능한 색상 수 이다.
pixelDepth	한 픽셀당 비트 수 이다.

3) 문서 객체모델 (DOM)

- 문서 객체 모델(Document Object Model)은 document와 관련된 집합이다.
- 문서 객체 모델을 사용해서 HTML 페이지에 태그를 추가, 제거, 수정할 수 있다.
- DOM은 반드시 HTML만 가능한 것은 아니다.

[HTML 문서의 DOM 트리 구조]

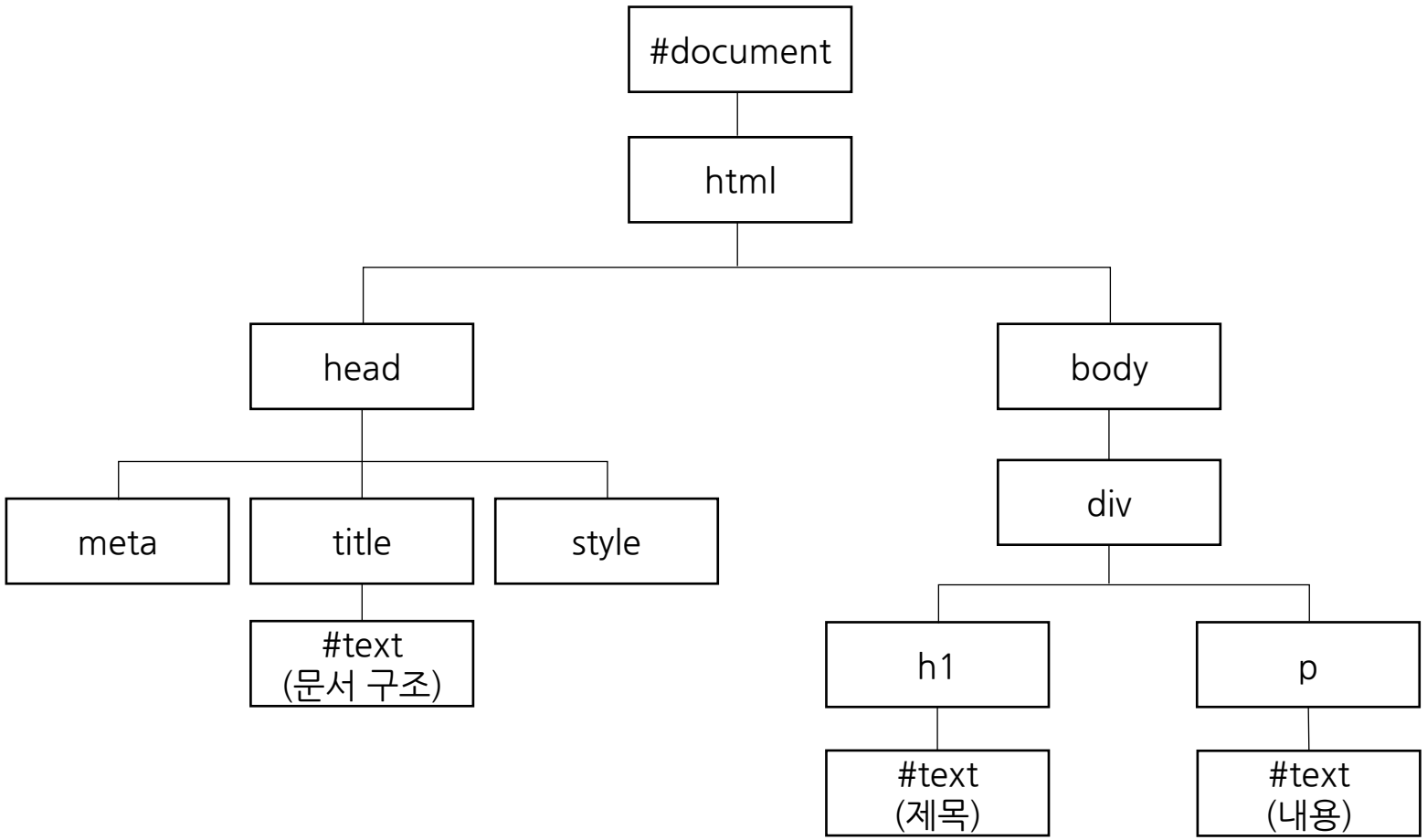
```
<html>
  <head>
      <title>문서 구조</title>
  </head>
  <body>
      <h1>제목</h1>
  </body>
</html>
```

[XML 문서의 DOM 트리 구조]

```
<books>
  <book>
     <title>HTML5</title>
     <author>김계희</author>
  </book>
  <book>
     <title>CSS3</title>
     <author>정은미</author>
  </book>
  <book>
     <title>JavaScript</title>
     <author>박은주</author>
  </book>
</books>
```

➢ **HTML의 DOM 구조는 다음과 같다.**

- 최상위는 #document이고 마지막은 #text로 구성되어 있다.
- #document는 HTML보다 상위에 있다.



[그림1] HTML의 DOM 구조

- document 객체의 요소를 추출하는 메서드 선택자는 크게 두 가지로 구분할 수 있다.
- 요소의 속성명으로 선택해 오는 원거리 선택자와 가까이에 있는 요소를 선택하는 근거리 선택자가 있다.

❖ 원거리 선택자

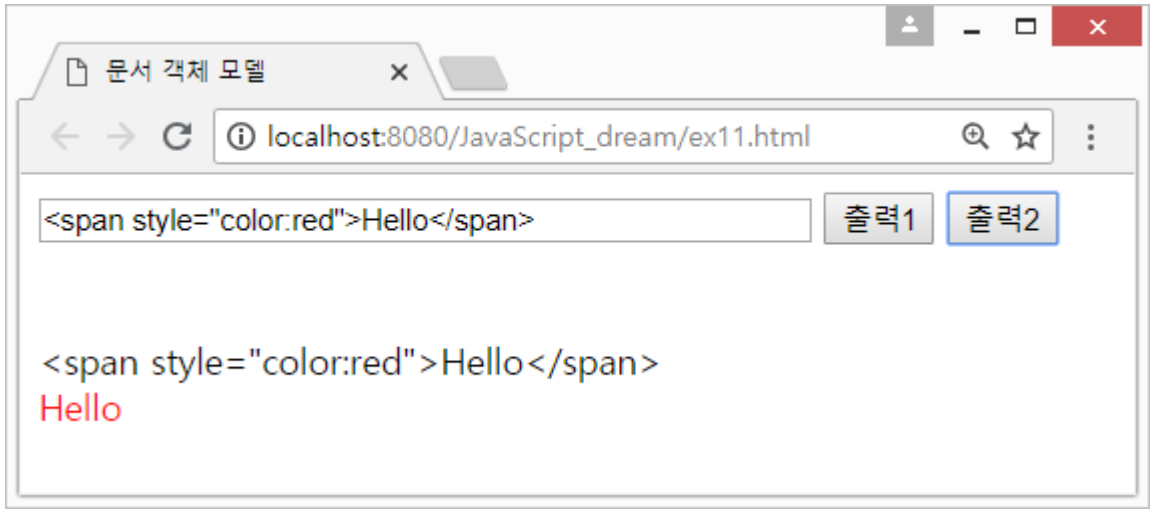
[원거리 선택자의 종류]

메서드	설명
getElementById(' id명 ')	태그의 id 속성이 getElementById(' id 명 ')과 일치하는 문서 객체를 가져오는 선택자이다.
getElementsByName(' name명 ')	태그의 name 속성이 getElementsByName(' name 명 ')과 일치하는 문서 객체를 가져오는 선택자이다.
getElementsByTagName(' tag명 ')	' tag 명 ' 과 일치하는 문서 객체를 가져오는 선택자이다.

[속성]

속성	설명
innerText	요소의 내용을Text로 설정하거나 반환한다.
innerHTML	요소의 내용을 HTML로 설정하거나 반환한다

```
<script>
    function process1(){
        var ob=document.getElementById("txt").value;
        document.getElementById("view1").innerText=ob;
    }
    function process2(){
        var ob=document.getElementById("txt").value;
        document.getElementById("view2").innerHTML=ob;
    }
</script>

<body>
    <input type="text" id="txt" size="50" name="text">
    <input type="button" value="출력1" onclick="process1()">
    <input type="button" value="출력2" onclick="process2()">
    <br><br><br>
    <div id="view1"></div>
    <div id="view2"></div>
</body>
```



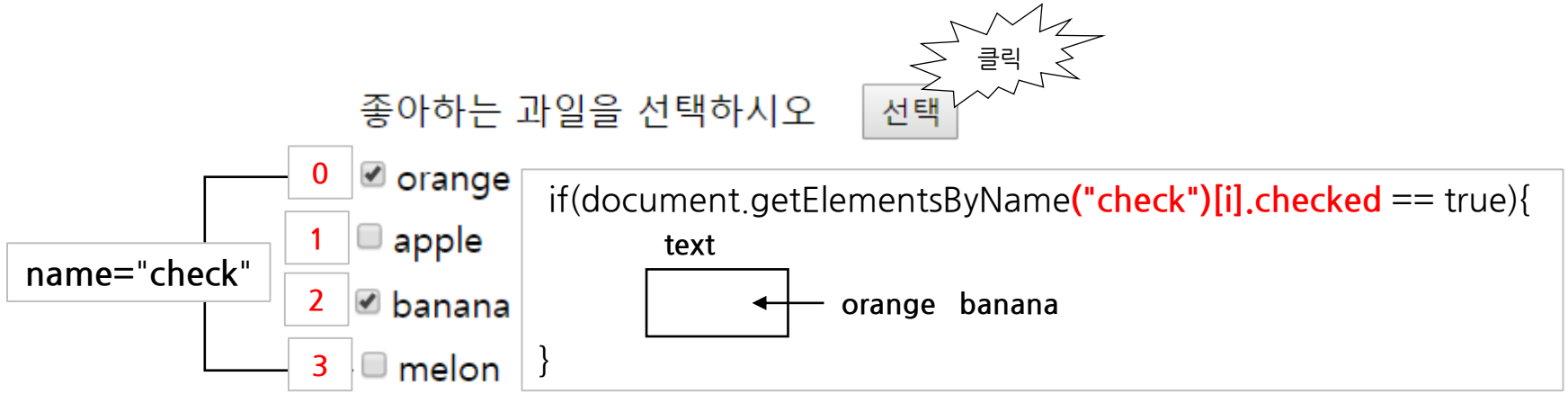
[ex31.html] getElementById

```
<script>
function choice(){
    var size = document.getElementsByName("check").length;
    var text="내가 좋아하는 과일은 ";
    for(var i = 0; i < size; i++){
        if(document.getElementsByName("check")[i].checked == true){
            text += (document.getElementsByName("check")[i].value);
            text += "  ";
        }
    }
    text += "입니다";
    document.getElementById("demo").innerHTML=text;
}
</script>
```

좋아하는 과일을 선택하시오 선택

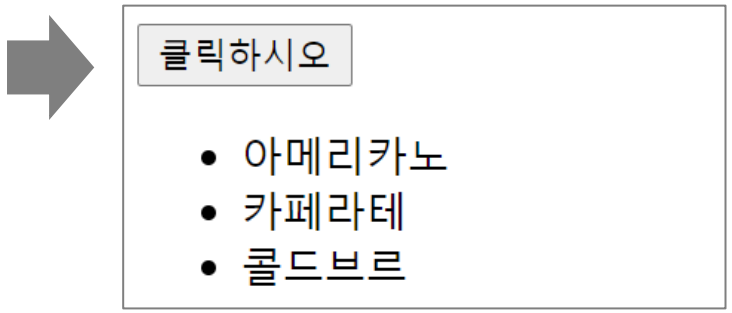
- orange
- apple
- banana
- melon

[ex32.html] getElementsByName

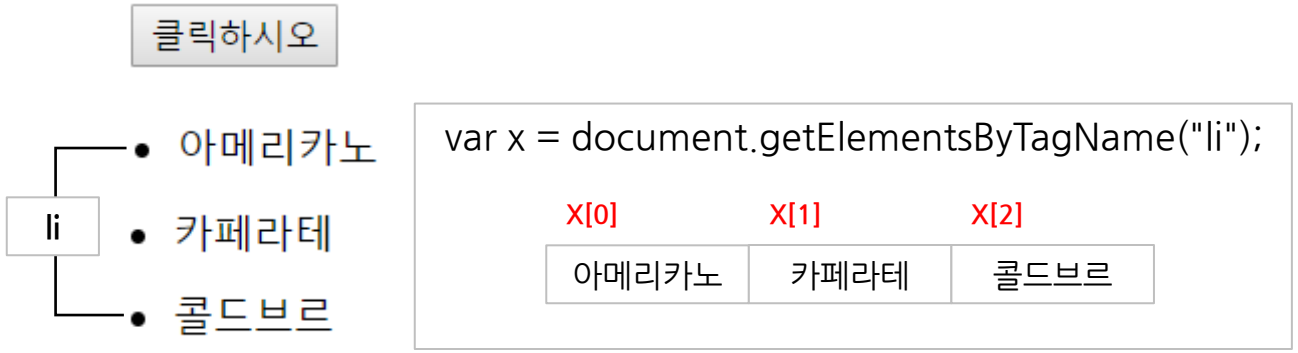


[그림2] fruit 체크박스의 선택 여부 확인

```
<script>
function choice() {
    var x = document.getElementsByTagName("li");
    var text="";
    for(i=0; i<x.length; i++){
        text += x[i].innerHTML;
        text += "  "
    }
    document.getElementById("demo").innerHTML = text;
}
</script>


<body>
<button onclick="choice()">클릭하시오</button><br>
<ul>
  <li>아메리카노</li>
  <li>카페라테</li>
  <li>콜드브르</li>
</ul>
<div id="demo"></div>
</body>
```



[ex33.html] getElementsByTagName



[그림3] li 로 만들어진 x 배열

❖ 근거리 선택자

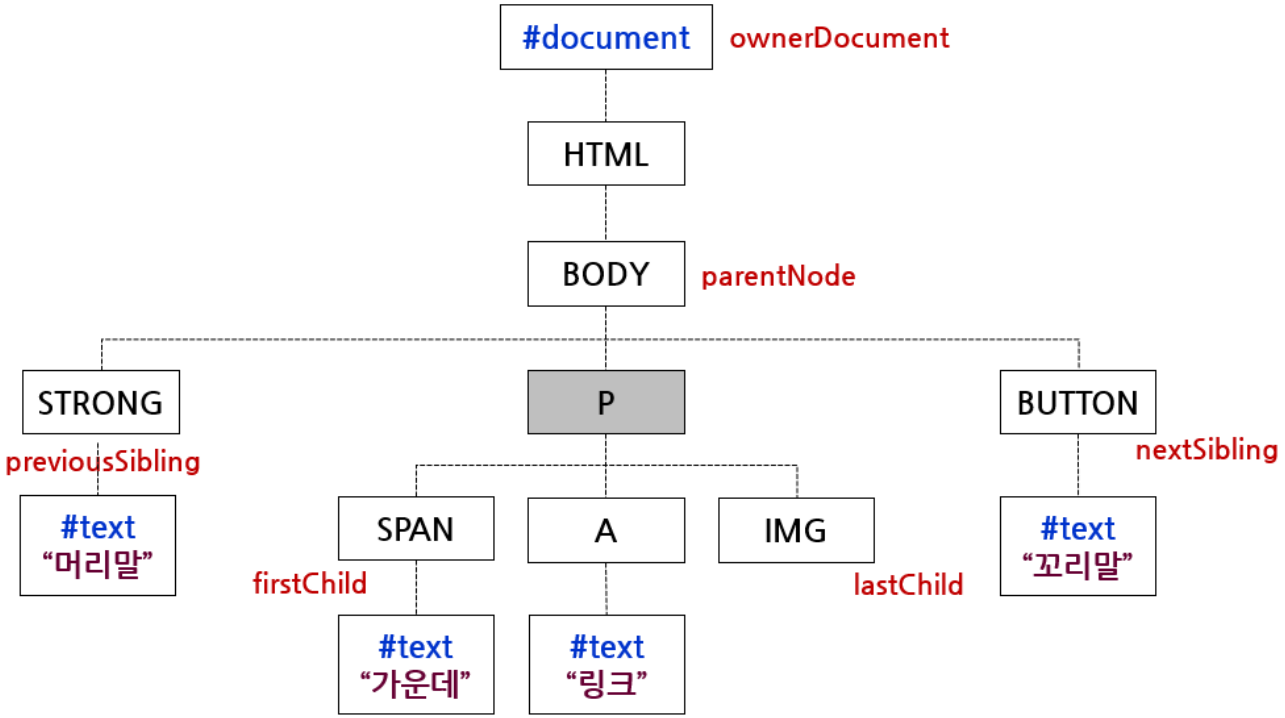
[근거리 선택자의 종류]

속성	설명
ownerDocument	최상위 노드를 선택한다.
parentNode	선택요소의 부모 노드를 선택한다.
childNodes	선택요소의 모든 자식 노드(배열객체로 저장)를 선택한다.
children	선택요소의 자식 노드(배열객체로 저장)만 선택한다.
firstChild	선택요소의 첫 번째 자식 노드만 선택한다.
lastChild	선택요소의 마지막 자식 노드만 선택한다.
previousSibling	선택요소의 이전에 오는 형제 노드를 선택한다.
nextSibling	선택요소의 다음에 오는 형제 노드를 선택한다.

[DOM 객체 메서드]

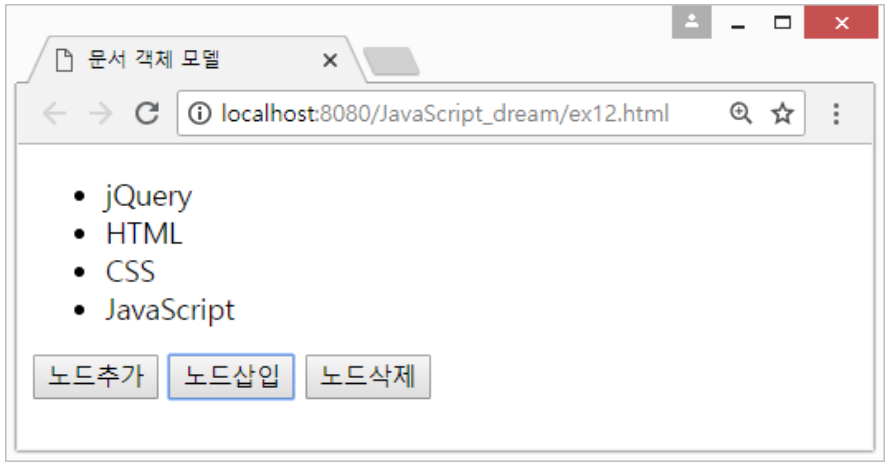
메서드	설명
`createElement(tagName)`	요소 노드를 생성한다.
`createTextNode(text)`	텍스트 노드를 생성한다.
`removeChild(child)`	자식 노드를 삭제한다.
`appendChild(node)`	자식 노드를 추가한다.
`setAttribute(name, value)`	객체의 속성을 설정한다.
`getAttribute(name)`	객체의 속성을 가져온다.

```
<body>
<strong>머리말</strong><p id="me"><span>가운데</span><a href="">링크</a><img src="image/cat.png"/></p><button>꼬리말</button>
<hr>
<script>
    var me = document.getElementById("me");
    document.write("topNode : " + me.ownerDocument.nodeName + "<br>");
    document.write("firstChild : " + me.firstChild.nodeName + "<br>");
    document.write("lastChild : " + me.lastChild.nodeName + "<br>");
    document.write("parentNode : " + me.parentNode.nodeName + "<br>");
    document.write("previousSibling : " + me.previousSibling.nodeName + "<br>");
    document.write("nextSibling : " + me.nextSibling.nodeName + "<br>");
</script>
</body>
```



[ex34.html] 노드

```
<script>
function appendNode(){
		var target=document.getElementById("target");
		var li=document.createElement("li");
		var text=document.createTextNode("JavaScript");
		li.appendChild(text);
		target.appendChild(li);
}
function insertNode(){
		var target=document.getElementById("target");
		var li=document.createElement("li");
		var text=document.createTextNode("jQuery");
		li.appendChild(text);
		target.insertBefore(li, target.firstChild);
}
function removeNode(){
		var target=document.getElementById("target");
		target.parentNode.removeChild(target);
}
</script>

<body>
<ul id="target">
   <li>HTML</li>
   <li>CSS</li>
</ul>

<input type="button" onclick="appendNode()" value="노드추가">
<input type="button" onclick="insertNode()" value="노드삽입">
<input type="button" onclick="removeNode()" value="노드삭제">
</body>
```

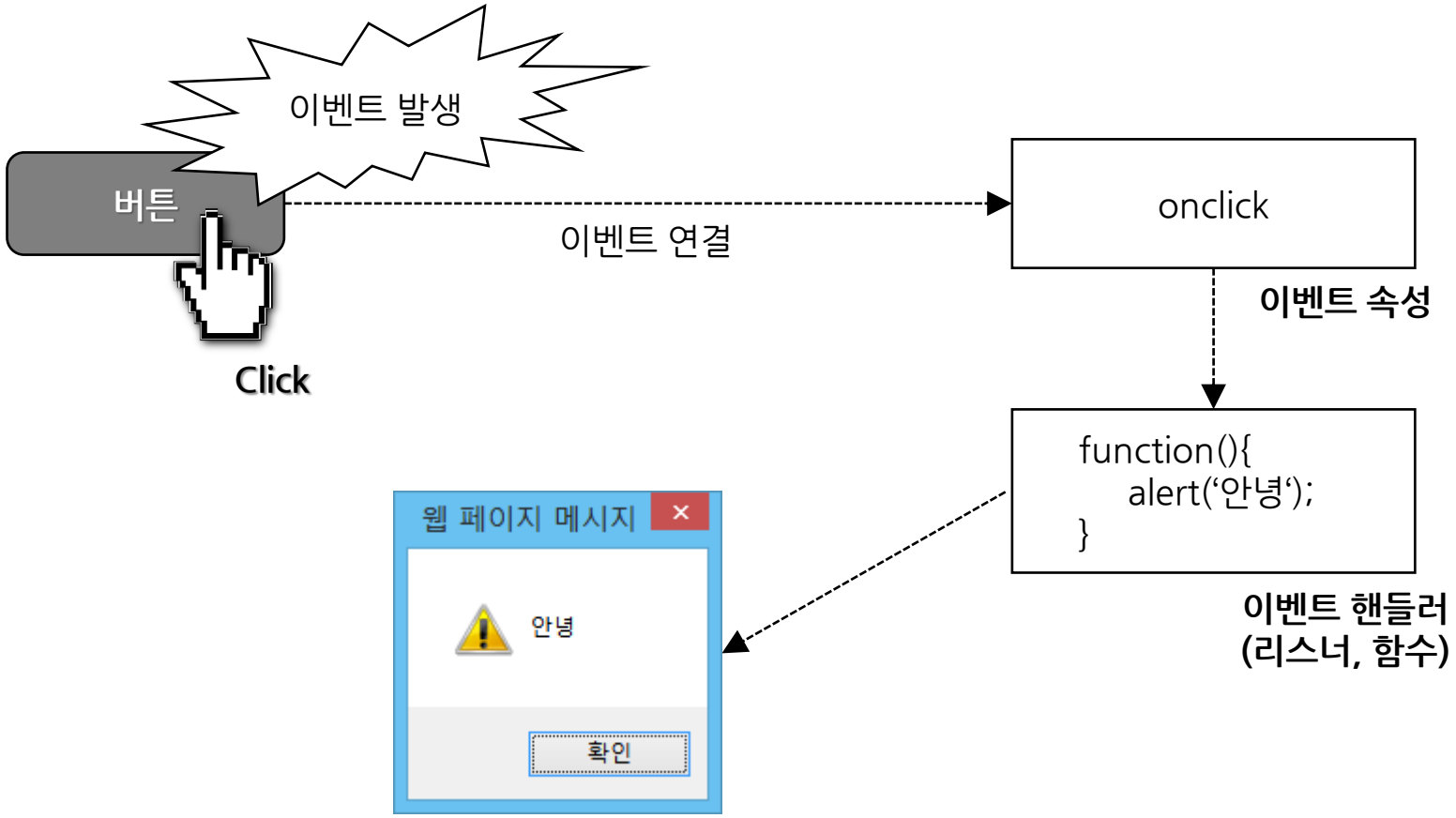


[ex35.html] 노드 추가 하기

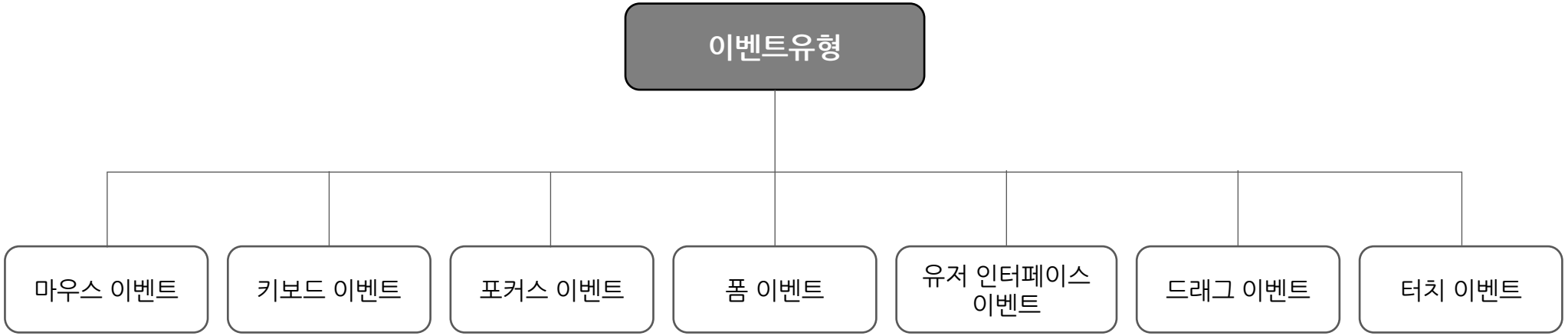
4. 이벤트 모델

1) 이벤트 모델이란?

이벤트란 키보드 입력, 마우스 클릭과 같이 사용자가 특정 행위를 했을 때 행위에 대한 결과 동작을 보여주는 것이다.



2) 이벤트 유형



❖ 마우스 이벤트

이벤트	이벤트 핸들러	이벤트 발생 시기
click	onclick()	마우스가 요소를 클릭할 때 발생한다.
mouseup	onmouseup()	마우스를 요소 위에 놓을 때 발생한다.
mousedown	onmousedown()	마우스로 요소를 누를 때 발생합니다.
mousemove	onmousemove()	마우스가 요소 위에서 움직일 때 발생한다.
mouseenter	onmouseenter()	포인터가 요소 위로 이동할 때 발생한다.
mouseleave	onmouseleave ()	포인터가 요소 밖으로 이동할 때 발생한다.
mouseover	onmouseover()	포인터를 요소 위로 이동하거나 자식 요소 중 하나 위로 이동할 때 발생한다.
mouseout	onmouseout()	포인터를 요소 밖으로 이동하거나 자식 요소 중 하나에서 벗어날 때 발생한다.

❖ 키보드 이벤트

이벤트	이벤트 핸들러	이벤트 발생 시기
keydown	onkeydown()	키보드를 처음 누를 때 발생한다.
keyup	onkeyup()	키보드에서 손을 땔 때 발생한다.
keypress	onkeypress()	키보드에 문자가 입력될 때 발생한다.

❖ 폼 이벤트

이벤트	이벤트 핸들러	이벤트 발생 시기
change	onchange( )	내용을 변경할 때 발생한다.
submit	onsubmit( )	입력 내용을 전송할 때 발생한다.
reset	onreset( )	입력 내용을 초기화할 때 발생한다.

❖ UI 이벤트

이벤트	이벤트 핸들러	이벤트 발생 시기
load	onload( )	웹페이지의 로드가 완료되었을 때 발생한다
resize	onresize()	브라우저의 창 크기를 변경할 때 발생한다.

❖ 터치 이벤트

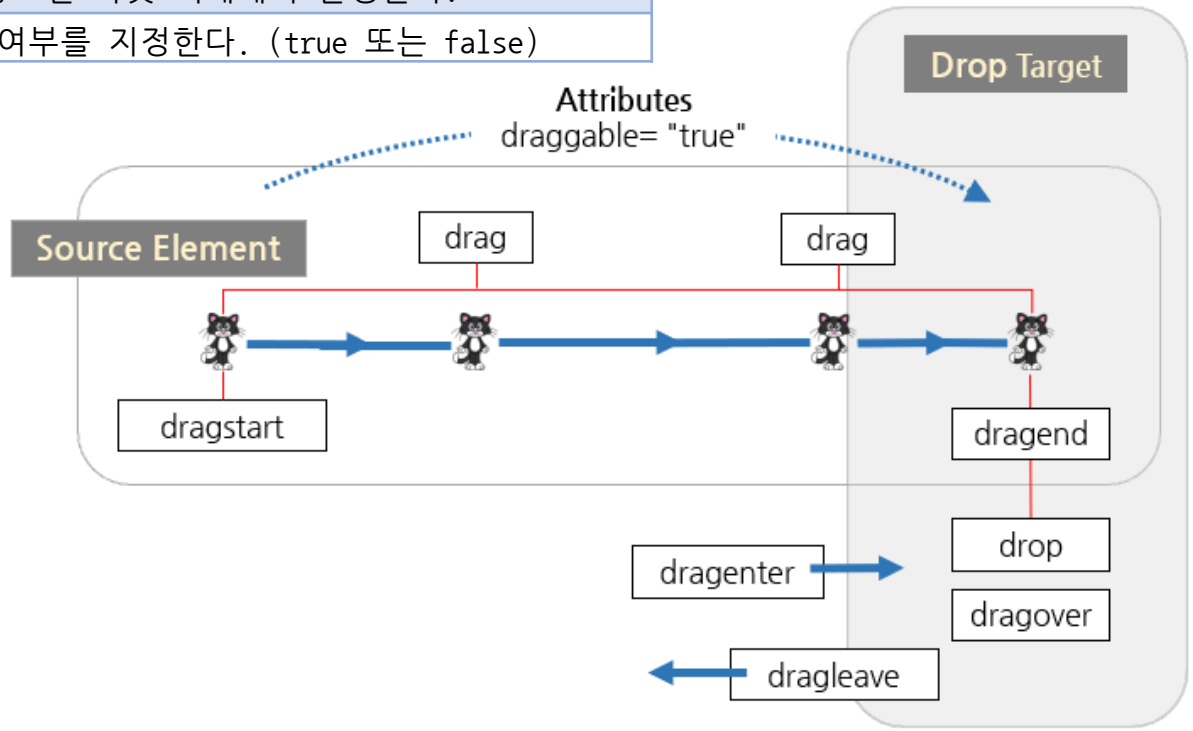
이벤트	이벤트 핸들러	이벤트 발생 시기
touchcancel	ontouchcancel( )	터치가 중단되면 발생한다.
touchend	ontouchend( )	스크린에서 터치가 끝나면 발생한다.
touchmove	ontouchmove( )	스크린에 드래그하면 발생한다.
touchstart	ontouchstart( )	스크린에 터치하면 발생한다.

❖ 포커스 이벤트

이벤트	이벤트 속성	이벤트 발생 시기
focus	onfocus( )	포커스를 얻었을 때 발생한다.
blur	onblur( )	포커스를 잃었을 때 발생한다.
focusin	onfocusin( )	포커스를 얻기 직전에 발생한다.
focusout	onfocusout( )	포커스를 잃기 직전에 발생한다.

❖ 드래그 이벤트

이벤트	이벤트 핸들러	이벤트 발생 시기
drag	ondrag( )	요소가 드래그 되고 있는 동안 소스 객체에서 발생한다.
dragend	ondragend( )	요소의 드래그가 끝나면 소스 객체에서 발생한다.
drangenter	ondrangenter( )	요소가 타겟 객체의 태그 영역에 들어갈 때 타겟 객체에서 발생한다.
dragleave	ondragleave( )	요소가 타겟 객체의 태그 영역에 나갈 때 타겟 객체에서 발생한다.
dragover	ondragover( )	요소를 타겟 객체 영역에서 드래그 하는 동안 타겟 객체에서 발생한다.
dragstart	ondragstart( )	요소를 드래그하기 시작할 때 소스 객체에서 발생한다.
drop	ondrop( )	요소를 타겟 객체 영역에서 놓으면 타겟 객체에서 발생한다.
draggable		요소를 드래그 할 수 있는지 여부를 지정한다. (true 또는 false)



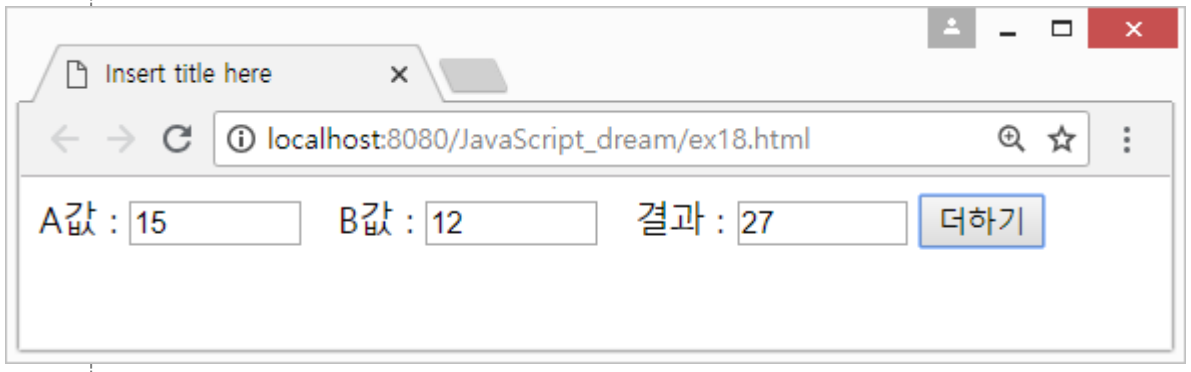
3) 이벤트 모델 종류

	DOM Level 0	DOM Level 2
모델 종류	• 인라인 이벤트 모델 • 고전 이벤트 모델	• 인터넷 익스플로러 이벤트 모델 • 표준 이벤트 모델
이벤트 연결	하나의 이벤트에 하나의 리스너만 연결 가능하다.	하나의 이벤트에 여러 개의 리스너를 연결할 수 있다.

❖ 인라인 이벤트모델

- 인라인 이벤트 모델은 가장 기본적인 이벤트 연결 방식이다.
- HTML 태그에 JavaScript 코드를 추가해서 이벤트 핸들러를 연결한다.
- 하나의 이벤트에 하나의 핸들러만 연결이 가능하다.

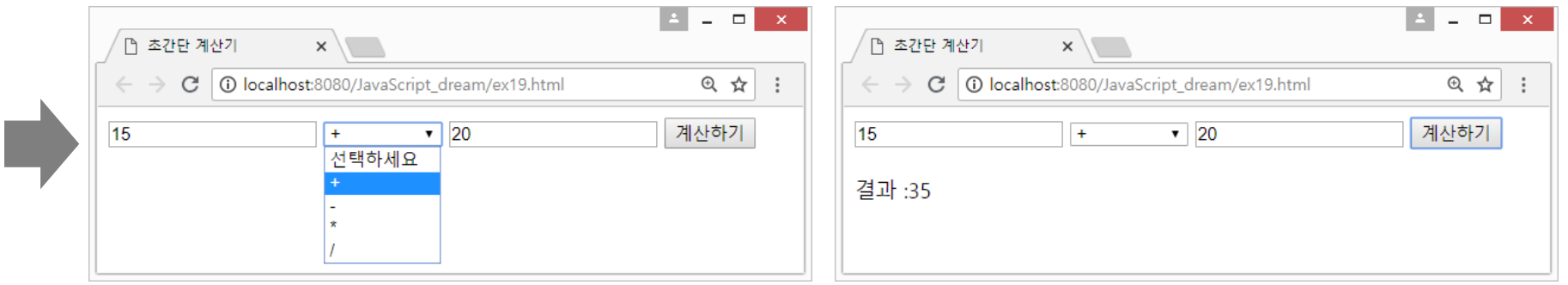
```
<script>
function plusfunc(){
        // "10" --> 10 변환
        //frm.rs.value=parseInt(frm.a.value) + parseInt(frm.b.value);
        frm.rs.value=Number(frm.a.value) + Number(frm.b.value);
}
</script>

<body>
<form name="frm">
        A값 : <input type="text" size="7" name="a">  
        B값 : <input type="text" size="7" name="b">  
        결과 : <input type="text" size="7" name="rs">
        <input type="button" name="plus" value="더하기" onclick="plusfunc()">
</form>
```



[ex36.html] 인라인 이벤트 모델

```
<script>
function process()
{
		var x=parseInt(document.getElementById('x').value);
		var y=parseInt(document.getElementById('y').value);
		var op=document.getElementById('ope').value;
		var solValue;

		switch(op)
		{
			case "+":solValue=x+y;	break;
			case "-":solValue=x-y;	break;
			case "*":solValue=x*y;	break;
			case "/":solValue=x/y;	break;
			default:alert("연산자 오류");
		}
		document.getElementById('sol').innerText="결과 :" + solValue;
}
</script>
```

```
<body>
	<input type="text" name="x" id="x" />

	<select id="ope">
		<option >선택하세요</option>
		<option value="+">+</option>
		<option value="-">-</option>
		<option value="*">*</option>
		<option value="/">/</option>
	</select>

	<input type="text" name="y" id="y" />
	<input type="button" value="계산하기" onclick="process()"><br><br>

	<span id="sol"></span>
</body>
```

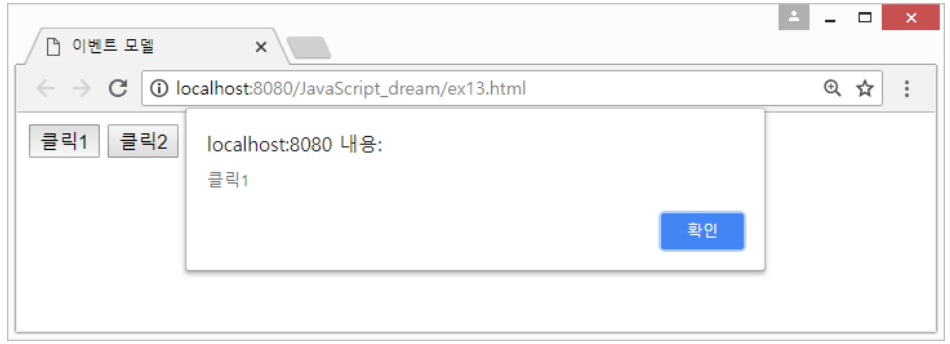


[ex37.html] 인라인 이벤트 모델

❖ 고전 이벤트모델

- 고전 이벤트 모델은 문서 객체의 이벤트를 사용해서 이벤트 핸들러에 연결하는 방법이다.
- 이전에도 사용했지만 지금도 많이 사용하는 방식이다.
- 하나의 이벤트에 하나의 핸들러만 연결할 수 있다.

```
<script>
    window.onload = function() {
        var btn = document.getElementById('btn1');
        btn1.onclick = function() {
            alert('클릭1');
            btn1.onclick = null;
        }
    };
</script>

<body>
<input type="button" id="btn1" value="클릭1">
<input type="button" id="btn2" value="클릭2" onclick="alert('클릭2')">
</body>
```



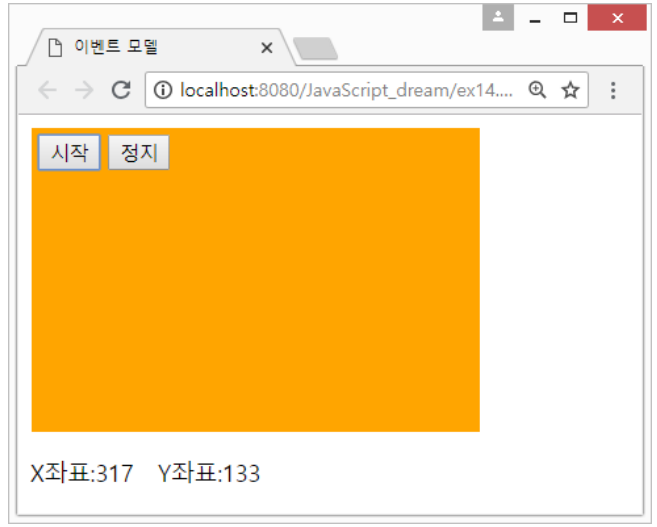
[ex38.html] 고전 이벤트 모델

❖ 표준 이벤트모델

- 객체나 요소의 메서드로 이벤트 핸들러를 전달하는 방식이다.
- 한 번에 여러 가지 이벤트 핸들러를 연결할 수 있다.

메서드	설명
addEventListener(eventName, handler, [useCapture])	특정 이벤트를 등록하고 해당 함수를 실행한다.
removeEventListener(eventName, handler)	등록한 이벤트 리스너를 제거한다.

```
<body>
  <div id="box">
      <button onclick="addEventHandler()" id="startbtn">시작</button>
      <button onclick="removeHandler()" id="stopbtn">정지</button>
  </div>
  <p id="demo"></p>

  <script>
    function myFunction() {
        document.getElementById("demo").innerHTML = "  X좌표:"
        + event.x + "   Y좌표:" + event.y;
    }
    function addEventHandler() {
       document.getElementById("box").addEventListener("mousemove", myFunction);
    }
    function removeHandler() {
        document.getElementById("box").removeEventListener("mousemove", myFunction);
    }
  </script>
</body>
```



[ex39.html] 표준 이벤트 모델

5. JSON과 정규표현식

1) JSON

➢ JSON은 저중량(lightweight) 데이터를 교환하는 형식으로 컴퓨터 시스템이 파싱하고 생성하기 쉽다.

[형식 1] 객체인 경우

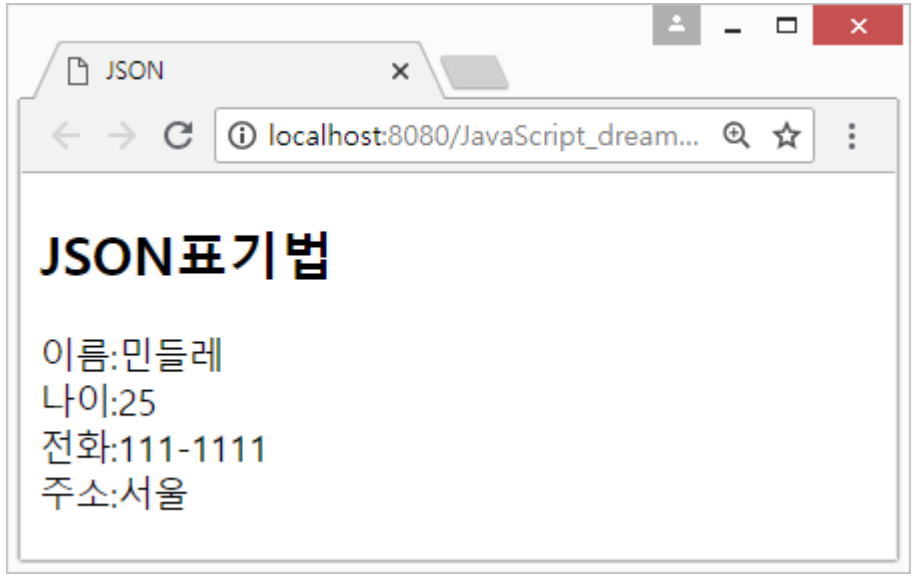
```
{"이름1":"값1", "이름2":"값2"}
```

[형식 2] 배열인 경우

```
[{"이름1":"값1", "이름2":"값2"}, {"이름1":"값1", "이름2":"값2"}]
```

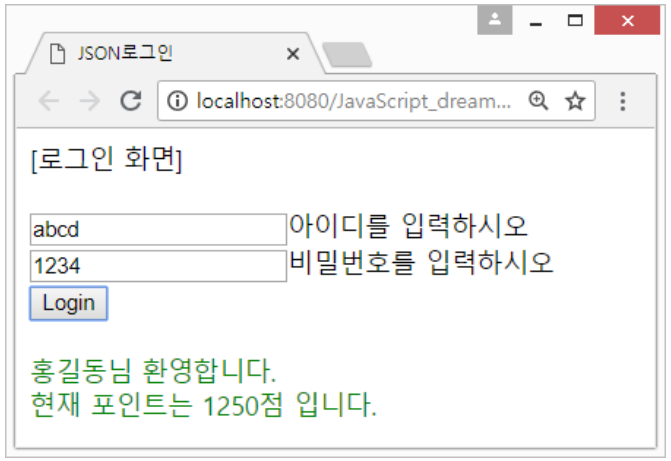
사용 → 객체명.이름 또는 객체명['이름']

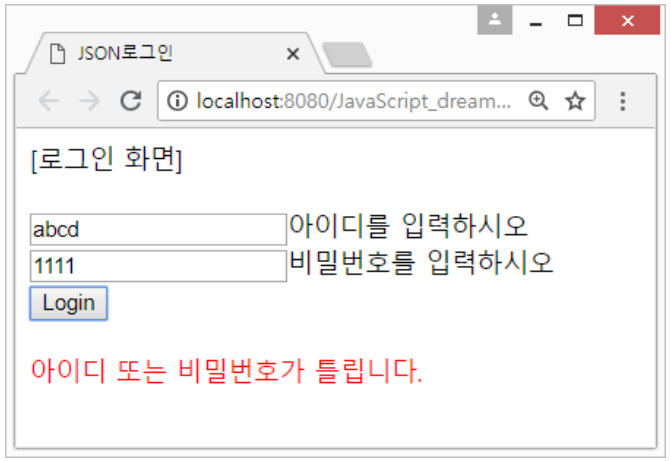
```
<script>
   function init(){
      var emp={"name":"민들레",
                  "age":"25",
                  "tel":"111-1111",
                  "address":"서울"};
      var msg="";
      msg += "이름:" + emp.name +"<br>"
               + "나이:" + emp.age +"<br>"
               + "전화:" + emp['tel'] +"<br>"
               + "주소:" + emp['address'] +"<br>"
      document.getElementById("result").innerHTML=msg;
   }
</script>

<body onload="init()">
   <h2>JSON표기법</h2>
   <div id="result"></div>
</body>
```



[ex40.html] JSON 표기법

```
<script>
   var emp = {
                "id" : "abcd",
                "pw" : "1234",
                "name" : "홍길동",
                "point" : "1250"
   };
   function login(){
      var a = document.getElementById("id").value;
      var b = document.getElementById("pw").value;

      if(emp.id == a && emp.pw == b)
                document.getElementById("confirm").innerHTML
                      ="<span style='color:green'>"+emp.name
                      + "님 환영합니다.<br>현재 포인트는 "+emp.point + "점 입니다.</span>";
      else
                document.getElementById("confirm").innerHTML
                      ="<span style='color:red'>아이디 또는 비밀번호가 틀립니다.</span>";
}
</script>

<body>
   <label for="input">[로그인 화면]</label><br><br>
   <input type="text" id="id"><span>아이디를 입력하시오</span><br>
   <input type="text" id="pw"><span>비밀번호를 입력하시오</span><br>
   <button id="log" onclick="login()">Login</button><br><br>
   <span id="confirm"></span>
</body>
```



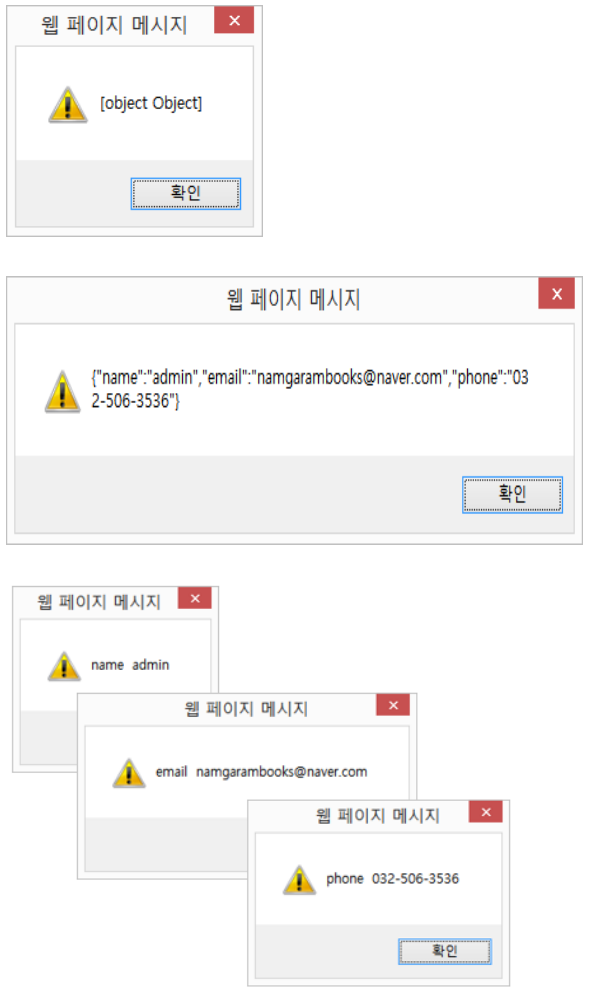


[ex41.html] 로그인 화면

[JSON 메서드]

메서드	설명
parse( )	string 객체를 JSON 객체로 변환해 준다.
stringify( )	JSON 객체를 String 객체로 변환해 준다.

```
<script>
        user={
          name:'admin',
          email:'namgarambooks@naver.com',
          phone:'032-506-3536'};

        alert(user);

        userStr = JSON.stringify(user);
        alert(userStr);

        JSON.parse(userStr, function(key,value){
                alert(key+"  "+value);
        });
</script>
```



[ex42.html] JSON 메서드

2) 정규 표현식

정규 표현식은 조합된 문자열이 특정 규칙에 맞게 작성되었는지 찾아내기 위한 검색 패턴이다.

[형식]

var 객체명 = new RegExp('정규표현식',['Flag']);

var 객체명 = /정규표현식/[Flag];

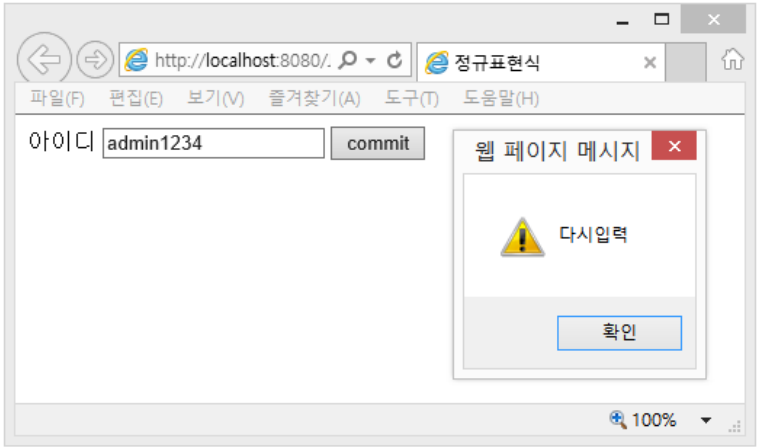
[정규 표현식]

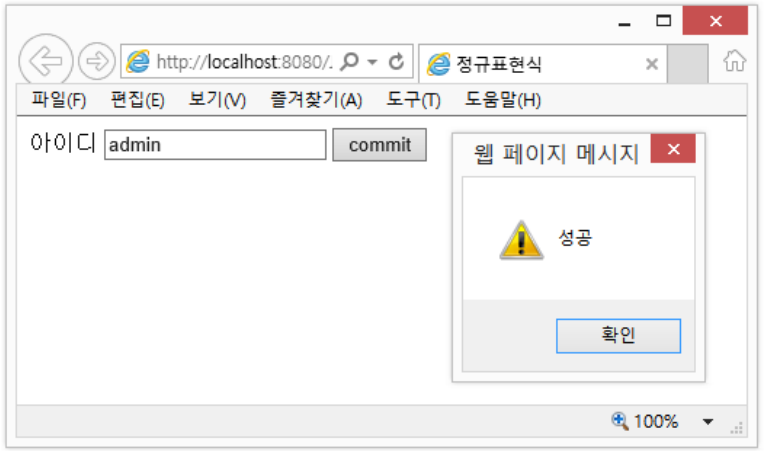
표현식	설명
^	문자열의 시작한다.
$	문자열의 종료한다.
.	임의의 한 문자이다. (단, ₩ 는 제외)
*	앞 문자가 없을 수도 무한정 많을 수도 있다.
+	앞 문자가 하나 이상입니다
?	앞 문자가 없거나 하나 있습니다.
{ }	횟수 또는 범위를 나타냅니다.

사용 예) ^[0-9]*$

^ 으로 우선 패턴의 시작을 알린다.
[0-9] 괄호 사이에 두 숫자를 넣어 범위를 지정해줄 수 있다.
* 를 넣으면 글자 수를 상관하지 않고 검사한다.
$ 로 패턴의 종료를 알린다.

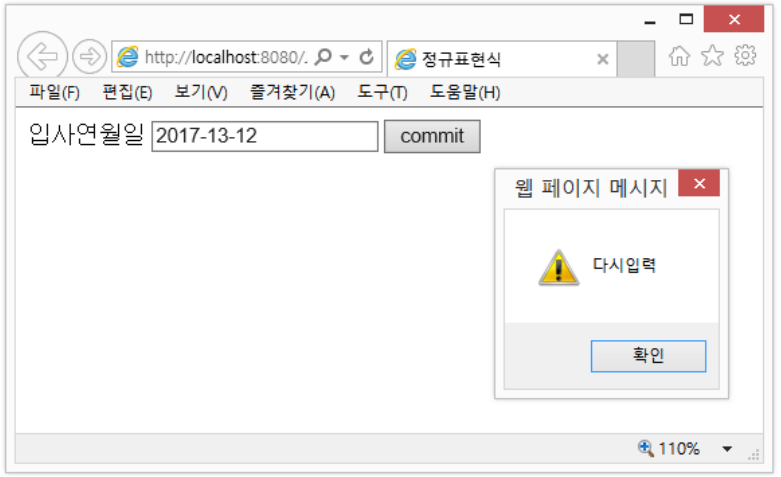
[아이디 입력 조건]
1. 입력되는 전체 길이는 5-8자이어야 한다.
2. 아이디는 숫자와 영문자로만 사용 가능하다.

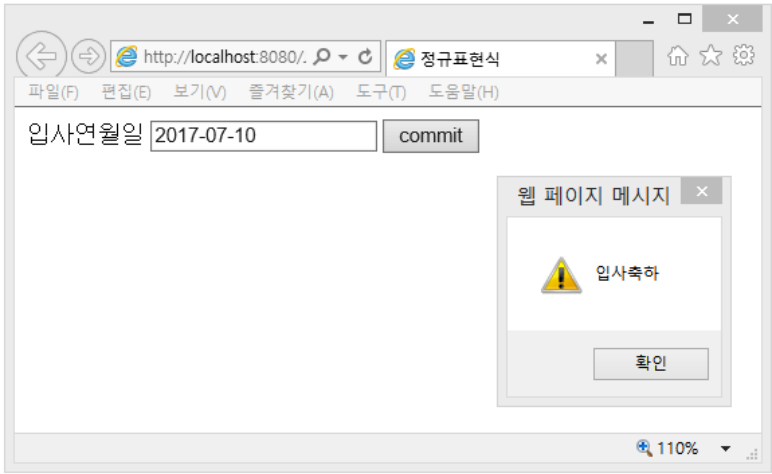




[ex43.html] 정규표현식_아이디확인

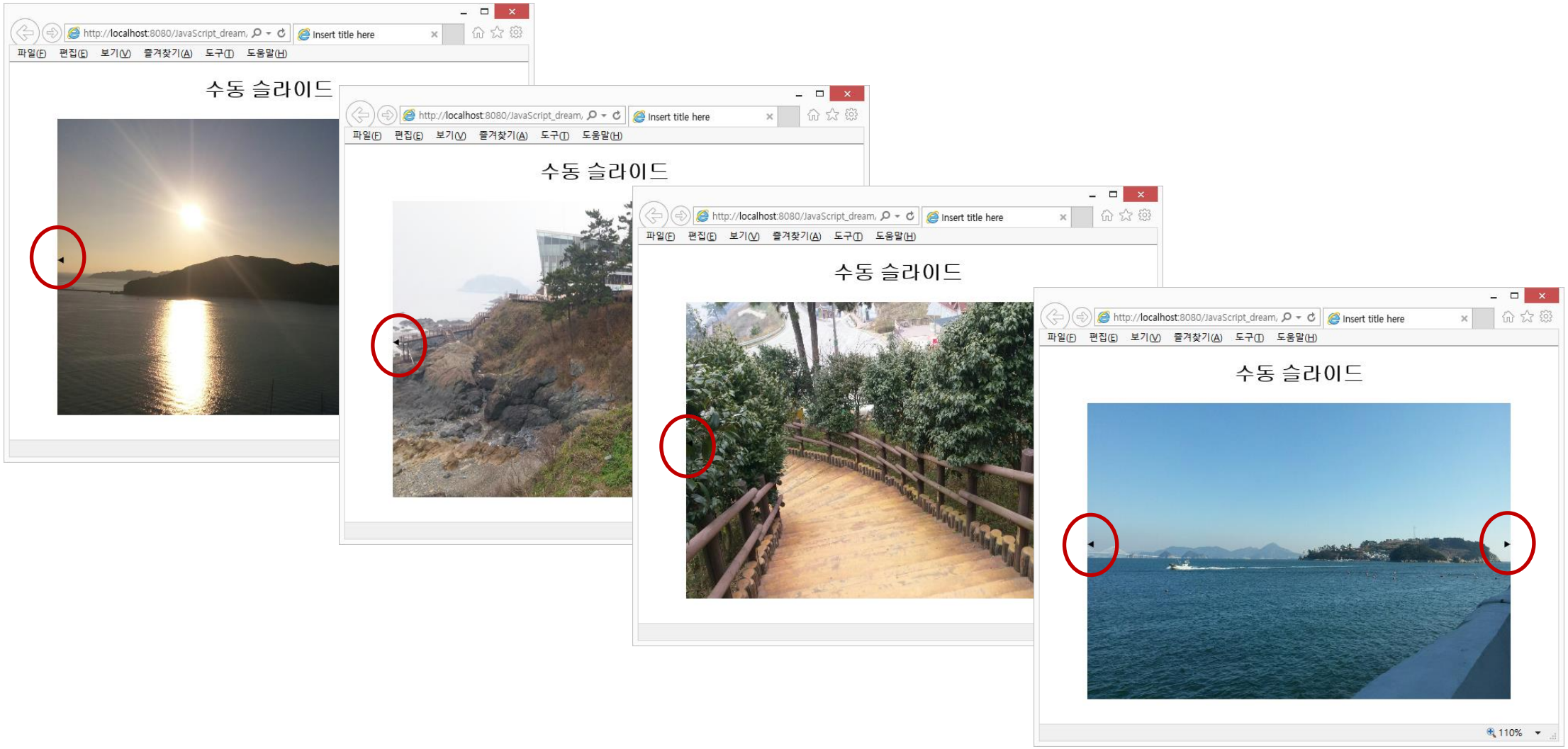
[날짜 입력 조건]
연도 4자리, 월이나 일은 1자리이거나 2자리로 합니다.
예시: 2018-05-24





[ex44.html] 정규표현식_입사 연월일

3) 자바스크립트를 이용한 수동 슬라이드 만들기



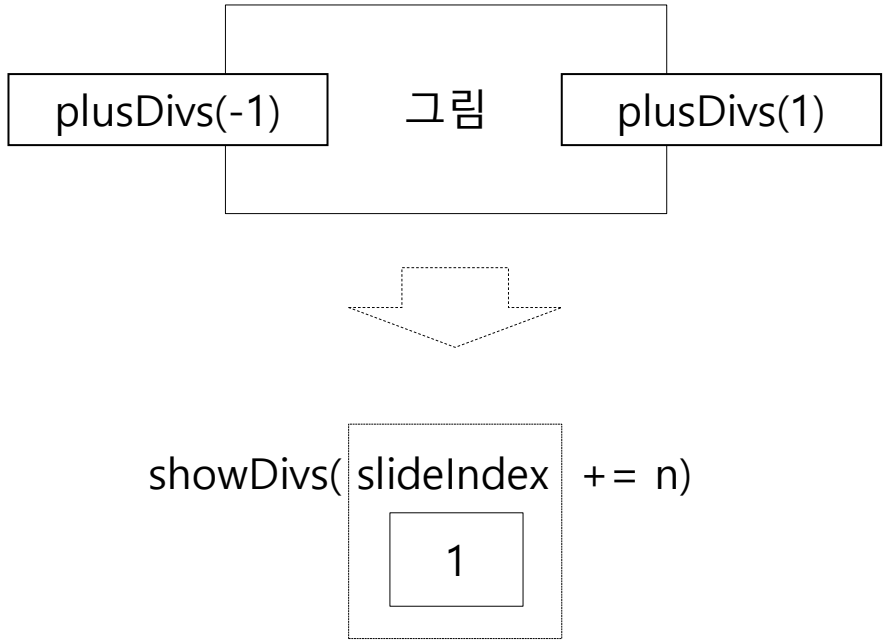
[ex45.html] 수동 슬라이드

수동 슬라이드 코딩 분석

```
<body onload="showDivs(1)">
<div align="center" class="large">
          <h2 align="center">수동 슬라이드</h2>
          <div class="small">
                    <img class="mySlides" src="image/01.jpg">
                    <img class="mySlides" src="image/02.jpg">
                    <img class="mySlides" src="image/03.jpg">
                    <img class="mySlides" src="image/04.jpg">

                    <a class="leftbtn" onclick="plusDivs(-1)">◀</a>
                    <a class="rightbtn" onclick="plusDivs(1)">▶</a>
          </div>
</div>
```

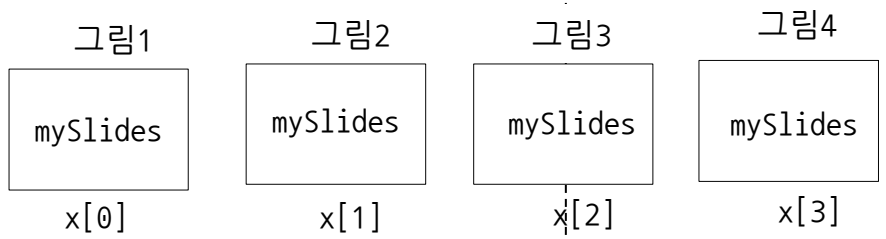
수동 슬라이드 코딩 분석

```
<script>
var slideIndex=1;
function plusDivs(n) {
  showDivs(slideIndex += n);
}
function showDivs(n) {
  var i;
  var x = document.getElementsByClassName("mySlides");
  if (n > x.length) {slideIndex = 1}
  if (n < 1) {slideIndex = x.length} ;
  for (i = 0; i < x.length; i++) {
     x[i].style.display = "none";
  }
  x[slideIndex-1].style.display = "block";
}
```

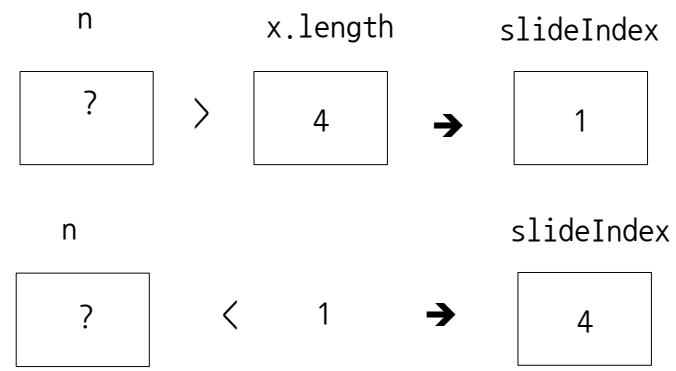


수동 슬라이드 코딩 분석

```
<script>
var slideIndex=1;
function plusDivs(n) {
  showDivs(slideIndex += n);
}
function showDivs(n) {
  var i;
  var x = document.getElementsByClassName("mySlides");
  if (n > x.length) {slideIndex = 1}
  if (n < 1) {slideIndex = x.length} ;
  for (i = 0; i < x.length; i++) {
     x[i].style.display = "none";
  }
  x[slideIndex-1].style.display = "block";
}
```



x.length는 그림의 갯수 ➔ 4



i

0 ➔ 1 ➔ 2 ➔ 3

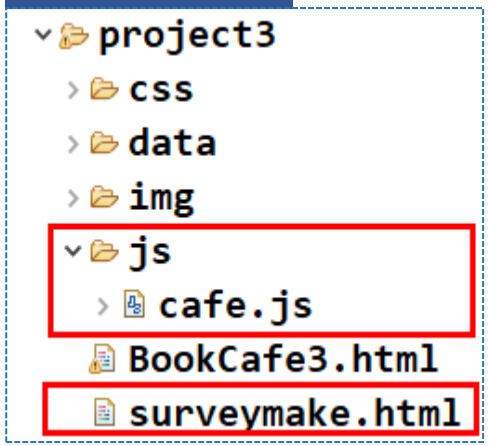
x[i].style.display= “ none ” ➔ 4개의 이미지를 안보이게 함

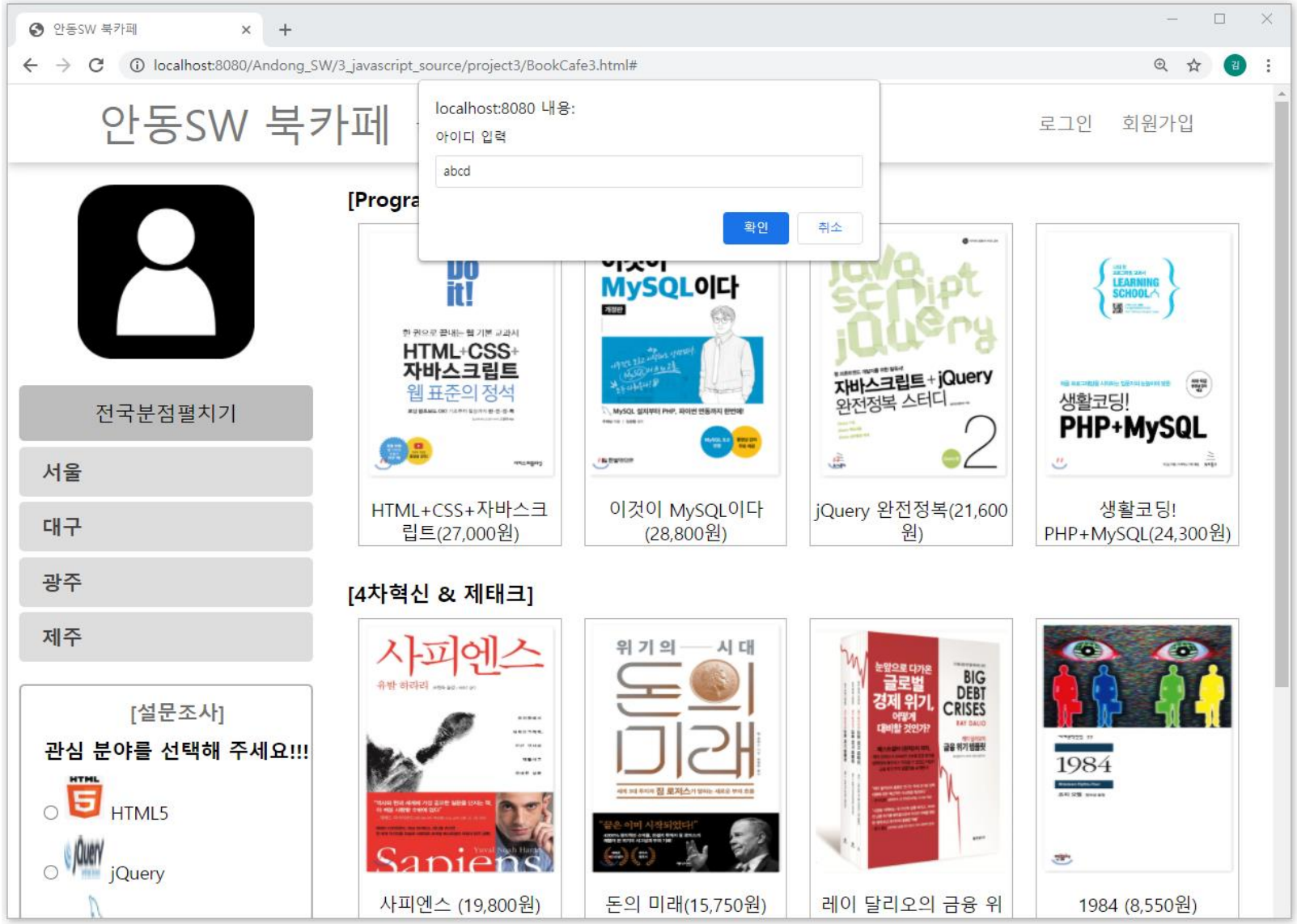
x[slideIndex-1].style.display= “ block ” ➔ 이미지를 보이게 함

4) 자바스크립트를 이용한 종합실습

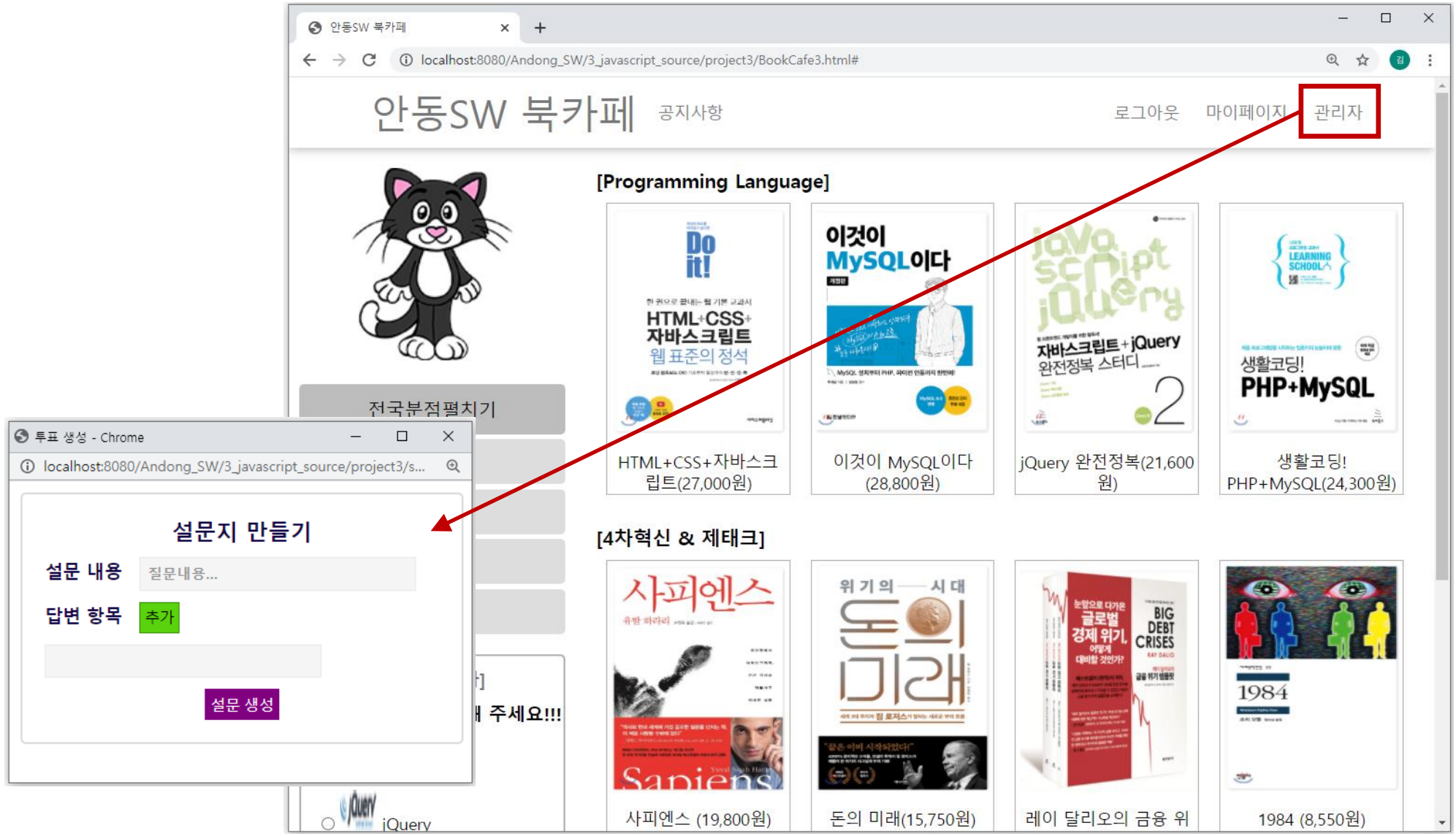
❖ 미니프로젝트 – 3 (JavaScript)

프로젝트구조



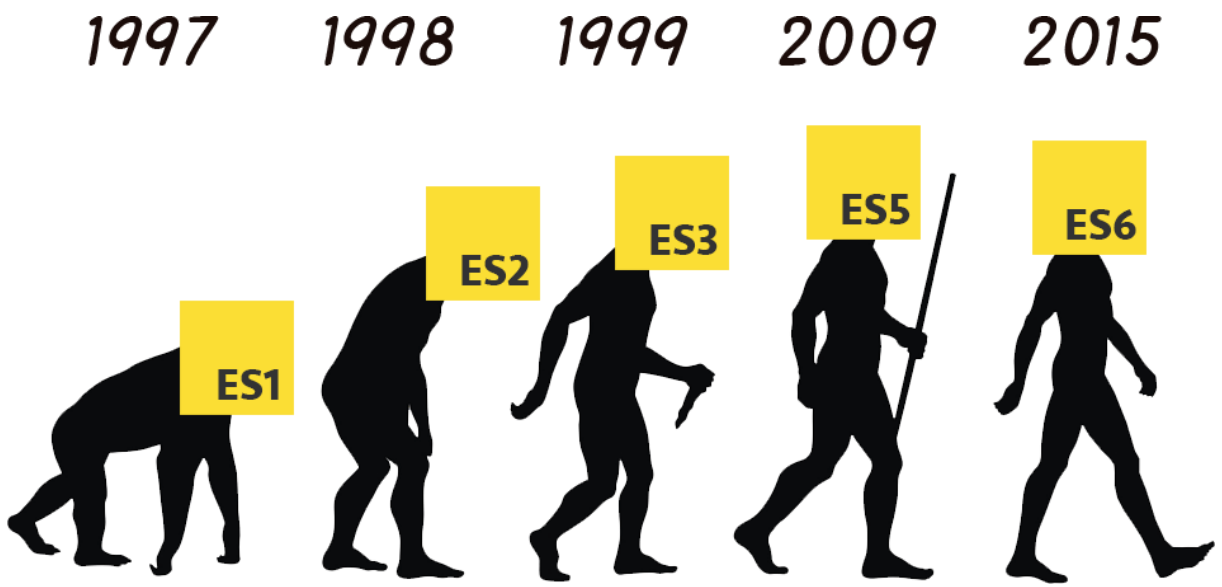


[BookCafe3.html] JavaScript를 추가한『안동SW 북카페』홈페이지


안동SW 북카페
localhost:8080/Andong_SW/3_javascript_source/project3/BookCafe3.html#
안동SW 북카페
공지사항
로그아웃
마이페이지
관리자
[Programming Language]
HTML+CSS+자바스크립트(27,000원)
이것이 MySQL이다 (28,800원)
jQuery 완전정복(21,600원)
생활코딩! PHP+MySQL(24,300원)
[4차혁신 & 제태크]
사피엔스 (19,800원)
돈의 미래(15,750원)
레이 달리오의 금융 위
1984 (8,550원)
전국분점펼치기
투표 생성 - Chrome
localhost:8080/Andong_SW/3_javascript_source/project3/s...
설문지 만들기
설문 내용
질문내용...
답변 항목
추가
설문 생성
주세요!!!
jQuery

6. ES6(ECMA Script)

- **표준화된 자바스크립트**
- ES6(ECMAScript 6)는 ECMAScript 표준의 가장 최신 버전이다.
- 현대적인 코드를 사용하면, 코드가 간결해지고 생산성이 향상된다.



1) 변수의 선언방식과 차이

- var은 재정의와 재 선언 모두 가능하다.
- let은 가변변수로 재정의가 가능하지만 재선언은 불가능하다.
- const는 불변변수로 재선언과 재정의 모두 불가능하다.

* 재선언: 똑같은 이름의 변수를 다시 만드는 것
* 재정의: 값이 지정된 변수에 값을 바꾸려는 것
* 스코프(scope) : 식별자(ex. 변수명, 함수명, 클래스명 등)의 유효범위.

```
//변수 선언
var x = 2;
// 재정의
x = 4;
// 재선언
var x = 4;
```

[es6_100.html]

❖ var 의 문제점

① 변수 선언이 유연하기 때문에 예기치 못한 값을 반환할 수 있다.

② 코드가 길어진다면 어디에서 어떻게 사용될지 파악하기 힘들다.

③ 함수 레벨 스코프로 인해 함수 외부에서 선언한 변수는 모두 전역 변수로 된다.

④ 변수 선언문 이전에 변수를 참조하면 언제나 undefined를 반환한다. (호이스팅 발생)

➡ 따라서, var 보다는 let과 const 키워드를 사용하는 것을 권장한다.

2) 템플릿 리터럴(Template literals)

➢ 템플릿 리터럴은 ES6부터 새로 도입된 문자열 표기법으로, 문자열 생성시 따옴표 대신, 백틱(`)을 사용한다. 따옴표와 달리 백틱 안에서는 줄바꿈이 반영된다. 또한, 문자열 사이에 변수나 연산을 넣을 때는 ${} 사이에 표현식을 삽입한다.

```
var js = "자바스크립트";
// 기존 코드
console.log("이건 " + js + "입니다.");

// 템플릿 리터럴 방식
console.log(`이건 ${js}입니다.`);

// 출력 결과 -> 이건 자바스크립트입니다.
```

[es6_101.html]

3) 화살표 함수(Arrow function)

- 화살표 함수는 함수 표현식을 보다 단순하고 간결한 작성하는 문법이다.

```
// 기본 함수 형식
let sum = function(a, b) {
  return a + b;
};

// 화살표 함수 형식
let sum = (a, b) => a + b;
```

[es6_102.html]

- 인수가 하나밖에 없다면 인수를 감싸는 괄호를 생략할 수 있다.
- 인수가 하나도 없을 땐 괄호를 비워 놓으면 된다. 이 때 괄호는 생략할 수 없다.
- 본문이 한 줄 밖에 없다면 중괄호를 생략할 수 있다.
- **중괄호는 본문 여러 줄로 구성**되어 있음을 알려주고, **중괄호를 사용했다면 return으로 결과값을 반환**해주어야 한다.

4) 모듈 내보내고 가져오기(export / import)

- 모듈을 내보내는 방법으로는 named export와 default export가 있다.

```
// named export 기본 형식
export { 모듈명1, 모듈명2 };
import { 모듈명1, 모듈명2 } from 'js 파일 경로';

// default export 기본 형식
export default 모듈명;
import 모듈명 from 'js 파일 경로';
```

- named export는 한 파일에서 여러 개를 export할 때 사용 가능하다.
- export한 이름과 동일한 이름으로 import해야 하며, 중괄호에 묶어서 import 해야 한다.
- 다른 이름으로 import 하려면 as를 사용하고, 한 파일에 있는 클래스나 변수들을 한 번에 import 하려면 * as를 사용한다.

```
// named export는 중괄호 포함 import
import { named1, named2 } from './example.js';

// named export에서 as를 사용하여 다른 이름으로 import
import { named1 as myExport, named2 } from './example.js';

// 한 파일에 있는 모든 클래스나 변수를 * as를 사용하여 한 번에 import
import * as Hello from './example.js'
```

- default export는 하나의 파일에서 단 하나의 변수 또는 클래스 등만 export 할 수 있다.
- 또한 import 할 때 아무 이름으로나 자유롭게 import 가능하며, 중괄호에 묶지 않아도 된다.

```
// default export 는 중괄호 없이 import
import default1 from './example.js';
```

5) 클래스(class)

➢ Class는 객체를 생성하기 위한 템플릿으로, 틀과 같은 역할을 한다.

- 클래스를 선언하려면 class 키워드와 함께 클래스의 이름을 작성한다.
- 클래스는 함수로 호출될 수 없다.
- 클래스 선언은 let과 const처럼 블록 스코프에 선언되며, 호이스팅(hoisting)이 일어나지 않는다.
  클래스는 반드시 정의한 뒤에 사용한다.
- 클래스의 메소드 안에서 super 키워드를 사용할 수 있다.
- static 키워드를 메소드 이름 앞에 붙여주면 해당 메소드는 정적 메소드가 됩니다.
- Getter 혹은 Setter를 정의하고 싶을 때는 메소드 이름 앞에 get 또는 set을 붙여주면 된다.
- extends 키워드를 사용하여 클래스에서 다른 클래스로 상속하면서 클래스의 기능을 확장해 나갈수 있다.
- 클래스에서 일반적인 방식으로 프로퍼티를 선언하고 할당하면 Public Property(공개 프로퍼티)이다.
  ⮕ 외부에서 프로퍼티에 접근하여 값을 사용하거나 수정이 가능
- 클래스에서 프로퍼티 앞에 # 키워드를 작성하여 선언하면 Private Property (비공개 프로퍼티)가 된다.
  ⮕ 오직 클래스 안에서만 사용, 변경이 가능. 외부 접근 불가.

```
class Person {
  constructor(name, age) {
    this.name = name;
    this.age = age;
  }
  nextYearAge() {  // 메서드 생성
    return Number(this.age) + 1;
  }
}
```

[es6_103.html]

```
// 클래스 상속
class introducePerson extends Person {
  constructor(name, age, city, futureHope) {
    // super 키워드를 이용해서 자식 class에서 부모 메서드를 호출
    super(name, age, city);
    this.city = city
    this.futureHope = futureHope
  }
  introduce () {
    return `저는 ${this.city}에 사는 ${this.name} 입니다.
            내년엔 ${super.nextYearAge()}살이며,
            장래희망은 ${this.futureHope} 입니다.`
  }
}

let kim = new introducePerson('kim','23','seoul', '개발자');
console.log(kim.introduce())
```

```
저는 seoul에 사는 kim 입니다.
        내년엔 24살이며,
        장래희망은 개발자 입니다.
>
```

6) 구조 분해 할당(destructing)

- 객체와 배열의 값을 쉽게 변수로 저장할 수 잇다.
- 객체에서 값을 꺼낼 때는 중괄호를 사용해서 key와 같은 이름으로 꺼내 올 수 있고, key와 다른 이름으로 꺼낼 때는 변수이름: 키 값으로 꺼내 올 수 있다.

```
const introduce = {name: 'candy', age: 23};
// key와 같은 이름으로 변수 선언
const { name, age } = introduce;
// 다른 이름으로 변수 선언 -> 변수이름: 키값
const { name : myName, age : myAge } = introduce;
console.log(myName) // candy
console.log(myAge) // 23
//배열에서 값을 꺼낼 때는 대괄호를 사용해서 앞에서부터 순차적으로 꺼내올 수 있다.
const fruits = ['apple', 'mango', 'grape'];
// 앞에서부터 순차적으로 변수 선언 가능
const [zero, one, two] = fruits;
console.log(zero) // apple
```

[es6_104.html]

```
candy
23
apple
>
```

7) Rest Operator / Spread Operator

- **Rest Operator(나머지 매개변수)**는 나머지 후속 매개변수들을 묶어 하나의 배열에 저장해서 사용하는 것이다. 묶어줄 매개변수 앞에 …을 붙여서 작성하면 된다.
- 즉, Rest Operator는 배열과 함수의 인자 중 나머지를 가리키며, 객체의 나머지 필드를 가리킨다.

```
// args에 1,2,3,4,5가 한꺼번에 배열로 담겨 인자로 넘겨진다.
function func1(...args) {
        console.log(`args: [${args}]`)
  // args: [1,2,3,4,5]
}
func1(1,2,3,4,5);
// arg1에는 1, arg2에는 2, arg3에는 나머지 3,4,5가 배열로 담겨 인자로 넘겨진다.
function func2(arg1, arg2, ...arg3) {
        console.log(`arg1: ${arg1}, arg2: ${arg2}, arg3: [${arg3}]`)
  // arg1: 1, arg2: 2, arg3: [3,4,5]
}
func2(1,2,3,4,5);
```

[es6_105.html]

```
args: [1,2,3,4,5]
arg1: 1, arg2: 2, arg3: [3,4,5]
>
```

- func(인자1, 인자2, …인자들)로 넘겨주게 되면 인자1, 인자2처럼 지정된 인자는 앞에서부터 각각의 값을 넣어주고 그 뒤의 나머지 후속 인자들을 배열에 인자들로 묶어서 보내주는 것이다.
- **Rest Operator**는 함수 정의에는 하나의 …만 존재할 수 있으며, 반드시 마지막 매개변수여야 한다.

```
func(...wrong, arg2, arg3)
// 틀린 예. ...wrong이 마지막으로 와야 한다.
```

- Spread Operator(전개 구문)는 묶인 배열 혹은 객체를 개별적인 요소로 분리한다. 즉, Rest와 반대 개념이라고 생각하면 되고, 마찬가지로 전개할 매개변수 앞에 ...을 붙여서 작성하면 된다.
- 따라서, 배열과 함수에선 또 다른 배열과 함수의 인자로의 전개를, 객체에선 또 다른 객체로의 전개를 한다.

```
let arr = [1, 2, 3, 4, 5];
console.log(...arr);
// 1 2 3 4 5
var str = 'javascript';
console.log(...str);
// "j" "a" "v" "a" "s" "c" "r" "i" "p" "t"
```

[es6_106.html]

- Spread Operator도 Rest Operator와 마찬가지로 ...의 작성 순서에 주의해야 한다. 등장 순서에 따라, 덮어씌워 질 수 있기 때문이다.

```
var obj = { name: '짱구', species: 'human'};
obj = { ...obj, age: 5};
console.log(obj)
// {name: "짱구", species: "human", age: 5}

obj = { ...obj, name: '짱아', age: 1};
console.log(obj);
// {name: "짱아", species: "human", age: 11}
```

[es6_106.html]

- 위 예제와 같이 ...obj가 먼저 나오고 name과 age가 나중에 등장함으로써 덮어씌워져 값이 변경된 것을 확인할 수 있다.
- 요약하면, Rest Operator(나머지 매개변수)는 배열로 묶는 역할을, Spread Operator(전개 구문)는 개별적인 요소로 분리하는 역할을 한다.
- 둘 다 "..." 을 붙여서 사용하며, 작성 순서에 주의해야 한다.

8) forEach() / map() / reduce()

- forEach()와 map()은 반복문을 돌며 배열 안의 요소들을 1대1로 짝지어 주는 역할을 한다.

forEach() : 배열 요소마다 한 번씩 주어진 함수(콜백)를 실행

```
배열.forEach((요소, 인덱스, 배열) => { return 요소 });
```

map() : 배열 내의 모든 요소 각각에 대하여 주어진 함수(콜백)를 호출한 결과를 모아 새로운 배열을 반환

```
배열.map((요소, 인덱스, 배열) => { return 요소 });
```

- 하지만 forEach()와 map()은 역할은 같지만, 리턴 값의 차이가 있다.
- forEach()는 기존의 배열을 변경하는 반면, map()은 결과값으로 새로운 배열을 반환한다.

```
var arr = [1,2,3,4,5];
// forEach()
var newArr = arr.forEach(function(e, i) {
  return e;
});
// return -> undefined

// map()
var newArr = arr.map(function(v, i, arr) {
  return v + 1;
});
// return -> 2, 3, 4, 5, 6
```

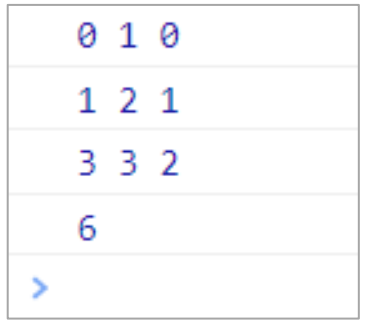
[es6_107.html]

reduce() : 배열의 각 요소를 순회하며 callback함수의 실행 값을 누적하여 하나의 결과값을 반환

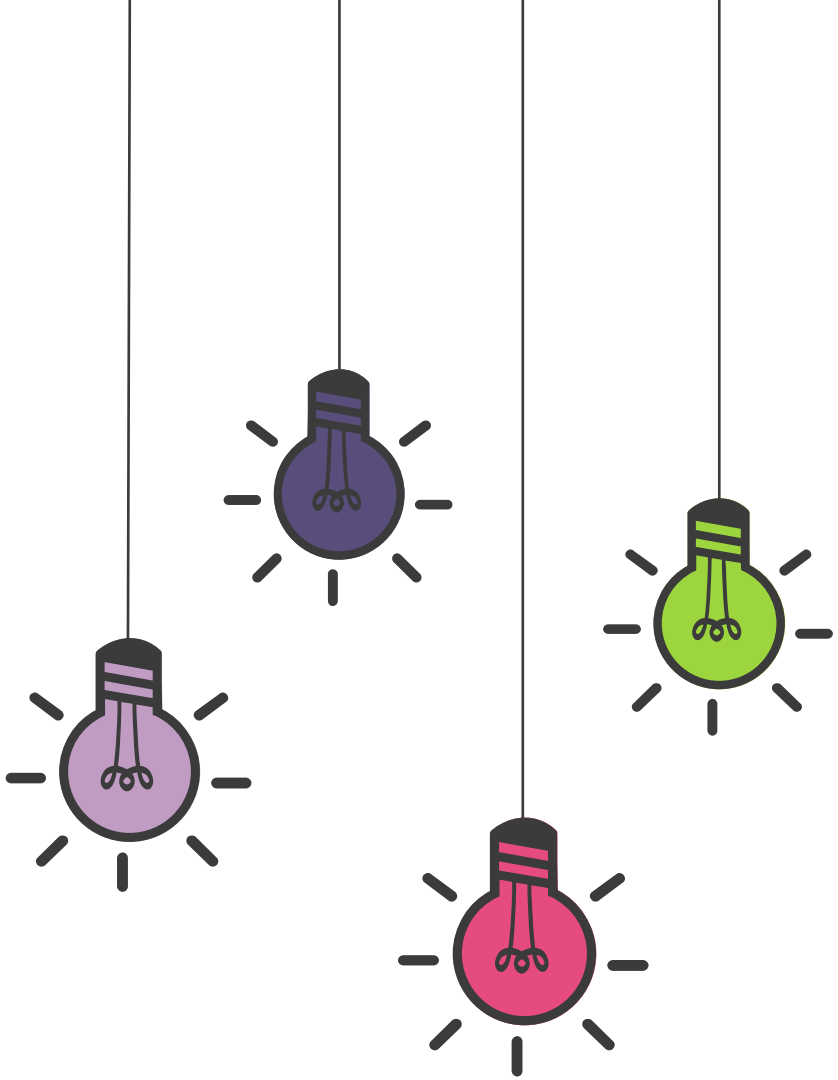
배열.reduce((누적값, 현잿값, 인덱스, 요소) => { return 결과 }, 초깃값);

```
sum = [1, 2, 3];
result = sum.reduce((prev, curr, i) => {
  console.log(prev, curr, i);
  return prev + curr;
}, 0);
// 0 1 0
// 1 2 1
// 3 3 2
console.log(result);  // 6
```

[es6_108.html]



- 초깃값을 적어주지 않으면 자동으로 초깃값이 0번째 인덱스의 값이 된다.
- reduce()는 초깃값을 배열로 만들고, 배열에 값들을 push하면 map과 같아진다.



감사합니다

THANK YOU FOR WATCHING