JavaScript 기본 문법

JavaScript의 객체란,

JS

- 객체(Object) = 속성(Property) + 기능(Method)

휴대폰을 객체에 비유한다면...

- 휴대폰의 기능 : 전화 걸기, 문자 보내기, 음악 듣기, 동영상 보기 등
- 휴대폰의 속성 : 크기, 색상, 카메라 화소수, 전원 (또는 음성) 버튼의 위치 등

- 객체.메서드(); //객체의 메서드를 실행한다.
- 객체.속성; //객체의 속성을 가져온다.
- 객체.속성=값 ; //객체의 속성값을 바꾼다.

JavaScript의 객체(Object)

JS

- 객체(Object) = 속성(Property) + 기능(Method)

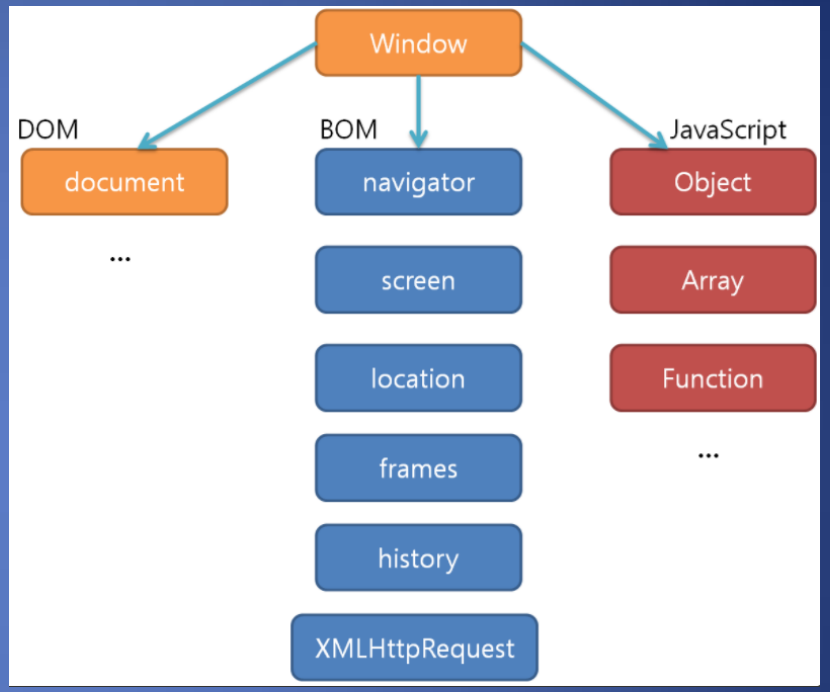
TV의 기능	TV의 속성	
TV.켜다( ); TV.끄다( ); TV.볼륨을 높이다( ); TV.볼륨을 낮추다( );	TV.높이; TV.컬러; TV.높이=30cm; TV.컬러=검정;	TV.height; TV.color; TV.height="30cm"; TV.color=black;

JavaScript의 객체(Object)의 종류

JS

1. 내장 객체 : 자바스크립트 엔진에 내장

- 문자(String)
- 날짜(Date)
- 배열(Array)
- 수학(Math)



2. 브라우저 객체 모델 (BOM)

- **window, screen, location, history, navigator**
- **window.location.href=** "싸이트 URL"
- **location.href=** "싸이트 URL"

3. 문서 객체 모델 (DOM)

- **<html><head><body><h1>...**

내장 객체

JS

- 내장객체 생성

```
참조 변수(인스턴트 네임) = new 생성함수( );
```

```
<script type="text/javascript">
//<![CDATA[
 var tv=new Object( ); //생성된 객체를 변수 tv가 참조

 //tv 객체의 속성 생성함.
tv.width="30cm";
tv.height="25cm";
tv.weight="20kg";
tv.color="white";

document.write(tv.width,"<br />");
document.write(tv.height,"<br />");
document.write(tv.weight,"<br />");
document.write(tv.color,"<br />");

//]]>
</script>
```

내장 객체 - 날짜 정보(Date) 객체

JS

- 현재 날짜 정보

```
참조 변수 = new Date( );
var d=new Date( ) ;
```

- 특정 날짜 정보

형태	예시
참조 변수 = new Date("연/월/일");	var d=new Date("2010/5/12") ;
참조 변수 = new Date(연, 월, 일);	var d=new Date(2010, 5, 12);

내장 객체 - 날짜 정보(Date) 객체

JS

- 날짜 관련 메서드

표기	설명
getFullYear( )	연도 정보
getMonth( )	월 정보(현재 월 -1, 0월부터 시작)
getDate( )	일 정보
getDay( )	요일 정보(일 : 0 ~ 토 : 6)
getHours( )	시 정보
getMinutes( )	분 정보
getSeconds( )	초 정보
getMilliseconds( )	밀리초 정보(1/1000초 단위)
getTime( )	1970년 1월 1일부터 경과된 시간을 밀리초로 표기

내장 객체 - 날짜 정보(Date) 객체

JS

- 날짜 관련 메서드

표기	설명
setFullYear( )	연도 정보만 수정
setMonth( )	월 정보만 수정 (현재 월 -1)
setDate( )	일 정보만 수정
setDay( )	요일 정보만 수정 (일 : 0 ~ 토 : 6)
setHours( )	시 정보만 수정
setMinutes( )	분 정보만 수정
setSeconds( )	초 정보만 수정
setMilliseconds( )	밀리초 정보만 수정 (1/1000초 단위)
setTime( )	1970년 1월 1일부터 경과된 시간을 밀리초로 수정

내장 객체 - 날짜 정보(Date) 객체

JS

- 날짜 관련 메서드

```
<script type="text/javascript">
//<![CDATA[
var t=new Date();
var nowMonth=t.getMonth();
var nowDate=t.getDate();
var nowDay=t.getDay();
document.write("현재 월:"+nowMonth,"<br />");
document.write("현재 일:"+nowDate,"<br />");
document.write("현재 요일:"+nowDay,"<br />");

var m=new Date('2018-06-29');  // 본인의 '생년 월 일'을 넣어 보세요.
var theYear=m.getFullYear();
var theMonth=m.getMonth()+1;
var theDate=m.getDate();
var theDay=m.getDay();
document.write("당신의 생일은 몇 년:"+theYear,"<br />");
if(theMonth==0){
        var twoelveMon=theMonth+12; /*월을 인식하는 부분은 0~11까지*/
        document.write("당신의 생일은 몇 월:"+twoelveMon,"<br />");
} else{
        document.write("당신의 생일은 몇 월:"+theMonth,"<br />");
}
document.write("당신의 생일은 몇 일:"+theDate,"<br />");
document.write("당신의 생일은 무슨 요일:"+theDay,"<br />");//]]>
</script>
```

내장 객체 - 날짜 정보(Date) 객체

JS

- 날짜 관련 메서드

```
<<결과>>

현재 월:2
현재 일:21
현재 요일:3
당신의 생일은 몇 월:4
당신의 생일은 몇 일:31
당신의 생일은 무슨 요일:1
```

내장 객체 - 날짜 정보(Date) 객체

JS

- 실습 – 01 오늘의 요일을 이미지로 표기하세요.

```
var t=new Date();
var nowDay=t.getDay();
console.log(nowDay);
if(nowDay==1){
        document.write("<img src='img-days/monday.gif'>");
}else if(nowDay==2){
        document.write("<img src='img-days/tuesday.gif'>");
}else if(nowDay==3){
        document.write("<img src='img-days/wednesday.gif'>");
}else if(nowDay==4){
        document.write("<img src='img-days/thursday.gif'>");
}else if(nowDay==5){
        document.write("<img src='img-days/friday.gif'>");
}else if(nowDay==6){
        document.write("<img src='img-days/saturday.gif'>");
}else{
        document.write("<img src='img-days/sunday.gif'>");
}
```

내장 객체 - 날짜 정보(Date) 객체

JS

- 실습 응용– 01 사계절의 이미지를 가져오고 오늘 일자의 계절 맞게 이미지를 표기하세요.

✓ 힌트 : 앞서 강의한 if 문과 else 문, else if 문을 활용하여 적용하시기 바랍니다.

내장 객체 - 날짜 정보(Date) 객체

JS

- 실습 응용– 01 사계절의 이미지를 가져오고 오늘 일자의 계절 맞게 이미지를 표기하세요.

```
var t=new Date();
var mon=t.getMonth()+1;

if(mon>=9 && mon<=11){
        document.write("<img src='img-season/img-f.jpg'/>");
}else if(mon>=6 && mon<=8){
        document.write("<img src='img-season/img-s.jpg'/>");
}else if(mon>=3 && mon<=5){
        document.write("<img src='img-season/img-m.jpg'/>");
}else{
        document.write("<img src='img-season/img-w.jpg'/>");
}
```

내장 객체 - 숫자(Number) 객체

JS

- 기본 형식

```
var 참조 변수 = new Number(값);
var 참조 변수= 값 ;
```

- <속성>의 형태

```
Number.속성;
```

- <기능>의 형태

```
객체.표기 메서드( ) ;
```

내장 객체 - 숫자(Number) 객체

JS

- Number 객체의 속성

속성	설명
MAX_VALUE	표현 가능한 가장 큰 수
MIN_VALUE	표현 가능한 가장 작은 수
POSIYIVE_INFINITY	무한대 수 표기
NEGATIVE_INFINITY	음의 무한대수 표기
NaN	숫자가 아닌 경우 표기

내장 객체 - 숫자(Number) 객체

JS

- Number 객체의 속성

속성	설명
toExponential(n)	지표기수법으로 소수점 n자리만큼 문자형 데이터로 반환
toFixed(n)	소수점 n 자리만큼 반올림하여 문자형 데이터로 반환
toPrecision(n)	유효 숫자 n의 개수만큼 반올림하여 문자형 데이터로 반환
toString( )	숫자형 데이터를 문자형 데이터로 반환
valueOf( )	객체의 원래 값을 반환
parseInt(값)	데이터를 정수로 변환하여 반환
parseFloat(값)	데이터를 실수로 변환하여 반환

- 지표기수란?

  $5470000000000000000 = 5.47 \times 10^{19}$

  54,700,000,000,000,000,000

내장 객체 - 숫자(Number) 객체

JS

- Number 객체 실습

```
<script type="text/javascript">
//<![CDATA[
var num1=3.456789;
var num2=700000;
var num3="30.5px";
var num4=40;

document.write(num2.toExponential(1), "<br />");
document.write(num1.toFixed(2), "<br />");
document.write(num1.toPrecision(2), "<br />");
document.write(num1.toString(), "<br />");
document.write(num4.valueOf(), "<br />");
document.write(parseInt(num3)+num4, "<br />");
document.write(parseFloat(num3)+num4, "<br />");
//]]>
</script>
```

내장 객체 - 숫자(Number) 객체

JS

- Number 객체 실습 - 결과

```
7.0e+5
3.46
3.5
3.456789
40
70
70.5
```

내장 객체 - 수학(Math) 객체

JS

- 산술 연산자가 할 수 없는 기능을 처리가능 (최대값, 최소값, 반올림값 등)

종류	설명
Math.abs(숫자)	숫자의 절대값을 반환
Math.max(숫자1, 숫자2, 숫자3, 숫자4)	숫자 중 가장 큰 값을 반환
Math.min(숫자1, 숫자2, 숫자3, 숫자4)	숫자 중 가장 작은 값을 반환
Math.pow(숫자, 제곱값)	숫자의 거듭제곱한 값을 반환
Math.random( )	0~1 사이의 난수를 반환
Math.round(숫자)	소수점 첫째 자리에서 반올림하여 정수를 반환
Math.ceil(숫자)	소수점 첫째 자리에서 무조건 올림해서 정수로 반환
Math.floor(숫자)	소수점 첫째 자리에서 무조건 내림해서 정수로 반환
Math.sqrt(숫자)	숫자의 제곱근 값을 반환
Math.PI	원주율 상수를 반환

내장 객체 - 수학(Math) 객체

JS

- 수학 객체 실습

```
<script type="text/javascript">
//<![CDATA[
var num=2.1234;

var maxNum=Math.max(10, 5, 8, 30);
var minNum=Math.min(10, 5, 8, 30);
var roundNum=Math.round(num);
var floorNum=Math.floor(num);
var floor_Num=Math.floor(Math.random( )*10);
var ceilNum=Math.ceil(num);
var rndNum=Math.random();
var piNum=Math.PI;

document.write(maxNum,"<br />");
document.write(minNum,"<br />");
document.write(roundNum,"<br />");
document.write(floorNum,"<br />");
document.write(floor_Num,"<br />");
document.write(ceilNum,"<br />");
document.write(rndNum,"<br />");
document.write(piNum,"<br />");
//]]>
</script>
```

내장 객체 - 수학(Math) 객체

JS

- 수학 객체 실습 - 결과

```
30
5
2
2
3
0.37493029172824066
3.141592653589793
```

내장 객체 - 수학(Math) 객체

JS

- 수학 객체 추가 확인

✓ 0~10까지 난수를 반환

```
Math.random( )*10 ;
```

✓ 0~10까지 정수로만 난수를 반환

```
Math.floor(Math.random( )*10) ;
```

※ 적용 예시 : 금융권에서 이자율 계산시 원단위 이하의 금액은 정수로만 적용함

내장 객체 - 수학(Math) 객체

JS

- 수학 객체 추가 실습

✓ 이미지 랜덤으로 보여주기
✓ (참조 이미지 slide1.jpg, slide2.jpg, slide3.jpg, slide4.jpg, slide5.jpg 활용)
✓ "<img src='img/slide"+(변수값)+".jpg'>" 를 활용해서 브라우저 화면에 출력

내장 객체 - 수학(Math) 객체

JS

- 수학 객체 추가 실습(결과)

✓ 이미지 랜덤으로 보여주기
✓ (참조 이미지 slide1.jpg, slide2.jpg, slide3.jpg, slide4.jpg, slide5.jpg 활용)
✓ "<img src='img/slide"+(변수값)+".jpg'>" 를 활용해서 브라우저 화면에 출력

```
var r_img = Math.ceil(Math.random()*5);
document.write("<img src='img/slide"+r_img+".jpg'>");
```

✓ 일단 random은 0~1 사이의 수를 가져올 수 있음. (예시 : 0.001, 0.999)
✓ 가져온 수에 이미지의 개수인 5를 곱하게 되면 0.005 또는 4.995라는 수로 변하게 됨
✓ ceil(올림) 대신 floor(내림)를 사용하면 파일명 중 없는 0이라는 것이 값으로 나올 수 있기 때문에 뒤에 반드시 +1이라는 숫자를 추가해줘야 함. 5번 이미지도 띄울 수 있음.

```
var r_img = Math.floor(Math.random()*5+1);
document.write("<img src='img/slide"+r_img+".jpg'>");
```

내장 객체 - 배열(Array) 객체

JS

- 배열 객체 기본 타입 및 적용

✓ 기본형 1

```
var 참조 변수 = new Array( ) ;
참조 변수[0]=값1 ;
참조 변수[1]=값1 ;
참조 변수[2]=값1 ;
```

✓ 기본형 2

```
var 참조 변수 = new Array(값1, 값2, 값3) ;
```

✓ 기본형 3

```
var 참조 변수 = [값1, 값2, 값3] ;
```

❖ 배열 객체 저장된 데이터 호출

```
참조 변수[인덱스 번호] ;
```

내장 객체 - 배열(Array) 객체

JS

- 배열 객체 기본 타입 및 적용 - 실습

✓ 특정 인덱스에 저장된 데이터 출력

```
<script type="text/javascript">
//<![CDATA[
var name=new Array();
name[0]="김원희";
name[1]="김민경";
name[2]="장동철";
name[3]="정주영";
name[4]="나원식";
document.write(name[3]+"<br/>"); //3번째 인덱스의 내용을 출력
//]]>
</script>
```

(정답 : 정주영)

내장 객체 - 배열(Array) 객체

JS

- 배열 객체 기본 타입 및 적용 - 실습

✓ 오늘의 요일 출력

```
<script type="text/javascript">
//<![CDATA[
var days=new Array("일", "월", "화",  "수", "목", "금", "토");
var today=new Date();
var yoil=days[today.getDay()];
document.write("오늘은 "+yoil+"요일 입니다.");
//]]>
</script>
```

내장 객체 - 배열(Array) 객체

JS

- 배열 객체 기본 타입 및 적용 - 응용문제

✓ 오늘의 요일 출력하여 이미지 불러오기

```
<script type="text/javascript">
//<![CDATA[
var days=new Array("sunday.gif", "monday.gif", "tuesday.gif", "wednesday.gif", "thursday.gif",
"friday.gif", "saturday.gif");
var t=new Date();
var nowDay=days[t.getDay()];
document.write("<img src='img-days/"+nowDay+"' />");
//]]>
</script>
```

✓ 배열을 응용하면 장문의 문장을 축약하여 사용 가능
날짜 정보를 받아와서 If 문을 사용하였을 때와 비교 하시기 바랍니다. (다음 페이지에서...)

내장 객체 - 배열(Array) 객체

JS

- 배열 객체 기본 타입 및 적용 - 응용문제

✓ 오늘의 요일 출력하여 이미지 불러오기
- if문 적용시 / 배열 적용시 비교

```
<script type="text/javascript">
//<![CDATA[
var days=new Array("sunday.gif", "monday.gif",
"tuesday.gif", "wednesday.gif", "thursday.gif", "friday.gif",
"saturday.gif");
var t=new Date();
var nowDay=days[t.getDay()];
document.write("<img src='img-days/"+nowDay+"' />");
//]]>
</script>
```

```
var t=new Date();
var nowDay=t.getDay();
if(nowDay==1){
        document.write("<img src='img-days/monday.gif'>");
}else if(nowDay==2){
        document.write("<img src='img-days/tuesday.gif'>");
}else if(nowDay==3){
        document.write("<img src='img-days/wednesday.gif'>");
}else if(nowDay==4){
        document.write("<img src='img-days/thursday.gif'>");
}else if(nowDay==5){
        document.write("<img src='img-days/friday.gif'>");
}else if(nowDay==6){
        document.write("<img src='img-days/saturday.gif'>");
}else{
        document.write("<img src='img-days/sunday.gif'>");
}
```

내장 객체 - 배열(Array) 객체

JS

- 배열 객체의 속성

종류	설명
join(연결문자)	배열 객체에 데이터 연결문자 기준으로 1개의 문자형 데이터로 반환
reverse( )	배열 객체에 데이터의 순서를 거꾸로 바꾼 후 반환
sort( )	배열 객체에 데이터를 오름차순으로 정렬
slice(index1, index2)	배열 객체에 데이터 중 원하는 인덱스 구간만큼 잘라서 배열 객체로 가져옴
splice( )	배열 객체에 지정 데이터를 삭제하고 그 구간에 새 데이터를 삽입
concat( )	2개의 배열 객체를 하나로 결합
pop( )	배열에 저장된 데이터 중 마지막 인덱스에 저장된 데이터를 삭제
push(new data)	배열 객체 마지막 인덱스에 새 데이터를 삽입
shift( )	배열 객체에 저장된 데이터 중 첫 번째 인덱스에 저장된 데이터 삭제
unshift(new data)	배열 객체의 가장 앞의 인덱스에 새 데이터를 삽입
length	배열 객체에 저장된 데이터의 개수를 반환

내장 객체 - 배열(Array) 객체

JS

- 배열 객체의 속성 적용

✓ 전체 합계 구하기

```
<script type="text/javascript">
//<![CDATA[
var money=[100,200,300,400];
var theLen=money.length;
var total=0;
for(var i=0; i<theLen; i++){
        total=total+money[i];
}
document.write(total);
//]]>
</script>
```

✓ 적용 예시 : 쇼핑몰에서 다수의 품목을 선택하여 결제할 때 적용

내장 객체 - 배열(Array) 객체

JS

- 배열 객체의 속성 적용 - 실습

✓ 배열 기본 실습 - 02

```
<script type="text/javascript">
//<![CDATA[
var num=["사당","교대","방배","강남"];
document.write(num,"<br />");
document.write(typeof num,"<br />");
document.write(num.join("-"),"<br />");
document.write(num.reverse(),"<br />");
document.write(num.sort(),"<br />");
//]]>
</script>
```

<결과>

사당,교대,방배,강남
object
사당-교대-방배-강남
강남,방배,교대,사당
강남,교대,방배,사당

- 추가 실습 과제
  - ✓ 내림차순으로 나올 수 있도록 하려면 어떻게 해야 할까요?

내장 객체 - 배열(Array) 객체

JS

- 배열 객체의 속성 적용 - 실습

✓ 배열 기본 실습 - 03

```
<script type="text/javascript">
//<![CDATA[
 var greenLine=["사당","교대","방배","강남"];
 var yellowLine=["미금","정자","모란","수서"];

 var twoLine=greenLine.concat(yellowLine);
document.write(twoLine,"<br />");

greenLine.pop();
document.write(greenLine,"<br />");

greenLine.push("삼성");
document.write(greenLine,"<br />");

greenLine.shift();
document.write(greenLine,"<br />");

greenLine.unshift("신도림");
document.write(greenLine,"<br />");
//]]>
</script>
```

<결과>

사당,교대,방배,강남,미금,정자,모란,수서
사당,교대,방배
사당,교대,방배,삼성
교대,방배,삼성
신도림,교대,방배,삼성

내장 객체 - 배열(Array) 객체

JS

- 배열 객체의 속성 적용 - 실습

✓ 배열 기본 실습 - 03

```
<script type="text/javascript">
//<![CDATA[
 var greenLine=["사당","교대","방배","강남"];
 var yellowLine=["미금","정자","모란","수서"];

 var twoLine=greenLine.concat(yellowLine);
document.write(twoLine,"<br />");

greenLine.pop();
document.write(greenLine,"<br />");

greenLine.push("삼성");
document.write(greenLine,"<br />");

greenLine.shift();
document.write(greenLine,"<br />");

greenLine.unshift("신도림");
document.write(greenLine,"<br />");
//]]>
</script>
```

<결과>

사당,교대,방배,강남,미금,정자,모란,수서
사당,교대,방배
사당,교대,방배,삼성
교대,방배,삼성
신도림,교대,방배,삼성

내장 객체 – 문자(String) 객체

JS

- 문자형 데이터를 객체로 취급

✓ 기본형 1

```
var 참조 변수 = new String(문자형 데이터);
```

✓ 기본형 1 - 예시

```
var t = new String("my name");
```

✓ 기본형 2

```
var 참조 변수 = 문자형 데이터;
```

✓ 기본형 2 - 예시

```
var t = "my name";
```

내장 객체 – 문자(String) 객체

JS

- 문자 객체의 속성 - #01

종류	설명
charAt(index)	문자열에서 인덱스 번호에 해당하는 문자를 반환 var str="web he she he"; str.charAt(2); →"b"를 반환

indexOf("찾을 문자")	문자열 왼쪽부터 찾을 문자와 일치하는 문자를 찾아 최초로 일치하는 문자의 인덱스 번호를 반환. 찾는 문자가 없을 경우 -1을 반환
	var str="web he she he";
	str.indexOf("he");
	→"4"를 반환
	var str="web he she he";
	str.indexOf("he", 5); //왼쪽부터 5번 인덱스 문항을 제외하고 그 다음에서 찾는다.
	→"11"를 반환

lastIndexOf("찾을 문자")	문자열 오른쪽부터 찾을 문자와 일치하는 문자를 찾아 최초로 일치하는 문자의 인덱스 번호를 반환. 찾는 문자가 없을 경우 -1을 반환 var str="web he she he"; str. lastIndexOf("he"); →"11"을 반환

내장 객체 – 문자(String) 객체

JS

- 문자 객체의 속성 - #02

종류	설명
match(“찾을 문자”)	문자열 왼쪽부터 찾을 문자와 일치하는 문자를 찾아 최초로 찾은 문자를 반환. 찾는 문자가 없을 경우 null을 반환 var str=”web he she he”; str. match(“man”); → null을 반환
replace(“바꿀 문자”, “새문자”)	문자열에서 왼쪽부터 바꿀 문자와 일치하는 문자를 찾아 최초로 찾은 문자를 새 문자로 치환 var str=”web he she he”; str. replace(“she”, “woman”); →”web he woman he”을 반환
search(“찾을 문자”)	문자열에서 왼쪽부터 찾을 문자와 일치하는 문자를 찾아 최초로 일치하는 인덱스 번호를 반환 var str=”web he she”; str. search(“he”); →”4”를 반환

내장 객체 – 문자(String) 객체

JS

- 문자 객체의 속성 - #03

종류	설명
slice(a, b)	a개의 문자를 자르고, b번째 이후에 문자를 자른 후 남은 문자를 반환 var str="webshow he she"; str. slice(3, 7); →"show"를 반환
substring(a, b)	a 인덱스부터 b 인덱스 이전 구간의 문자를 반환. var str="webshow he she"; str. substring(3, 7); →"show"를 반환
substr(a, 문자개수)	문자열에 a 인덱스부터 지정한 문자 개수만큼 문자열을 반환 var str="webshow he she"; str. substr (4, 3); →"how"를 반환
split("문자")	지정한 문자를 기준으로 문자 데이터를 나누어 배열에 저장하여 반환 var str="webkshowkhekshe"; var arr=str. split ("k"); → arr는 web, show, he, she 배열로 저장

내장 객체 – 문자(String) 객체

JS

- 문자 객체의 속성 - #04

종류	설명
toLowerCase( )	문자열에서 영문 대문자를 모두 소문자로 변경
toUpperCase( )	문자열에서 영문 소문자를 모두 대문자로 변경
length	문자열에서 문자의 개수를 반환
concat("새로운 문자")	문자열에 새로운 문자를 결합
charCodeAt(index)	찾을 문자의 아스키코드 값을 반환
fromCharCode(아스키 코드값)	아스키코드 값에 해당하는 문자를 반환
trim( )	문자의 앞 또는 위에 공백 문자열을 삭제 var str="hello "; str. trim( ); → 공백이 제거된"hello"를 반환

내장 객체 – 문자(String) 객체

JS

- 문자 객체의 속성 적용01 - 실습

```
<script type="text/javascript">
//<![CDATA[
var t="Hello Thank you good luck to you";

document.write(t.charAt(16),"<br />");
document.write(t.indexOf("you"),"<br />");
document.write(t.indexOf("you",16),"<br />");
document.write(t.lastIndexOf("you"),"<br />");
document.write(t.lastIndexOf("you",25),"<br />");
document.write(t.match("luck"),"<br />");
document.write(t.search("you"),"<br />");
document.write(t.substr(21,4),"<br />");
document.write(t.substring(6,12),"<br />");
document.write(t.replace("you","me"),"<br />");
document.write(t.toLowerCase(),"<br />");
document.write(t.toUpperCase(),"<br />");
document.write(t.length,"<br />");
var s=t.split(" ");
document.write(s[0],"<br />");
document.write(s[4],"<br />");
//]]>
</script>
```

내장 객체 – 문자(String) 객체

JS

- 문자 객체의 속성 적용01 – 실습(결과)

```
g
12
29
29
12
luck
12
luck
Thank
Hello Thank me good luck to you
hello thank you good luck to you
HELLO THANK YOU GOOD LUCK TO YOU
32
Hello
luck
```

내장 객체 – 문자(String) 객체

JS

- 문자 객체의 속성 적용02 - 실습

```
<script type="text/javascript">
//<![CDATA[
var userName=prompt("당신의 영문 이름은?","");
var upperName=userName.toUpperCase();
document.write(upperName,"<br />");

var userNum=prompt("당신의 연락처는?","");
var result=userNum.substring(0,7)+"****";
document.write(result,"<br />");
//]]>
</script>
```

내장 객체 – 문자(String) 객체

JS

- 문자 객체의 속성 적용02 - 실습

```
<script type="text/javascript">
//<![CDATA[
var userName=prompt("당신의 영문 이름은?","");
var upperName=userName.toUpperCase();
document.write(upperName,"<br />");

var userNum=prompt("당신의 연락처는?","");
var result=userNum.substring(0,7)+"****";
// var result=userNum.substring(0,userNum.length-4)+"****";
document.write(result,"<br />");
//]]>
</script>
```

내장 객체 – 문자(String) 객체

JS

- 문자 객체의 속성 적용02 – 응용 실습

✓ 은행 계좌번호 뒤쪽 5글자 가리기 (단, 조건은 은행계좌번호의 수는 은행마다 모두 다르다.)

✓ 힌트 : substring 을 이용하여 반영할 것

내장 객체 – 문자(String) 객체

JS

- 문자 객체의 속성 적용02 – 응용 실습(결과)

✓ 은행 계좌번호 뒤쪽 5글자 가리기(단, 조건은 은행계좌번호의 수는 은행마다 모두 다르다.)

```
<script type="text/javascript">
//<![CDATA[
var depositNumber=prompt("계좌번호를 입력하세요.", "");
var result=depositNumber.substring(0,depositNumber.length-5)+"*****"
document.write(result,"<br />");
//]]>
</script>
```

✓ 실무 적용 사례 : 오픈 마켓에서 판매자가 월별(또는 일정 주기별)로 수익을 돌려받을 때 본인의 은행 계좌(또는 가상 계좌)를 기재하게 되어 있음

객체 – 총정리 실습

JS

- 현재 날짜의 달력을 테이블로 출력하세요.

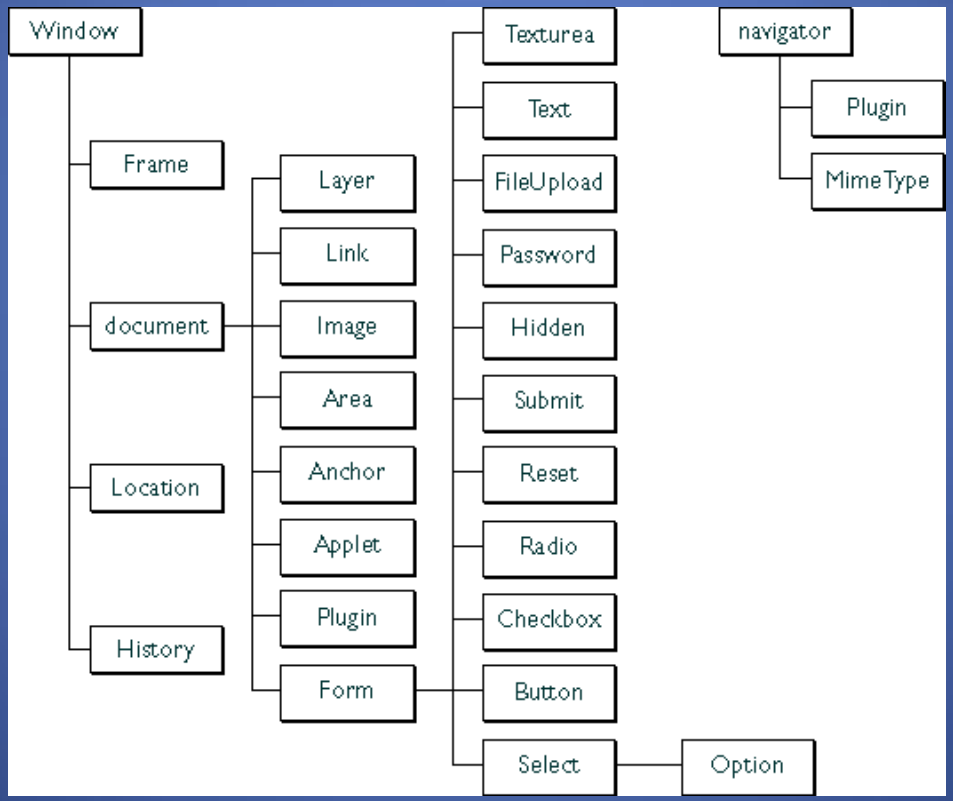
```
var date=new Date();
var y=date.getFullYear();
var m=date.getMonth();
var d=date.getDate();
var theDate=new Date("y/m/1");
var theDay=theDate.getDay();
var last=[31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31];
if(y%4==0 && y%100!=0 || y%400==0){
        lastDate=last[1]=29;
}
var lastDate=last[m];
var row=Math.ceil((theDay+lastDate)/7); //필요한 행수

document.write("<h1>"+y+"."+(m+1)+"</h1>");
var calendar="<table border='1'>";
calendar+="<tr>";
calendar+="<th>일</th>";
calendar+="<th>월</th>";
calendar+="<th>화</th>";
calendar+="<th>수</th>";
calendar+="<th>목</th>";
calendar+="<th>금</th>";
calendar+="<th>토</th>";
calendar+="</tr>";
var dNum=1; //달력에 표기되는 일의 초기 값
for(var i=1; i<=row; i++){
        calendar+="<tr>";
        for(var k=1; k<=7;k++){
                if(i==1 && k<theDay || dNum>lastDate){
        calendar+="<td> </td>";
                }else{
        calendar+="<td>"+dNum+"</td>";
                        dNum++;
                }
        }
        calendar+="</tr>";
}
document.write(calendar);
```

브라우저 객체 (BOM : Browser Object Model)

JS

- Window는 최상위 객체
- 브라우저에 계층적으로 내장되어 있는 객체들을 지칭
- 브라우저 객체 : window, screen, location, history, navigator 객체 등이 있음



브라우저 객체 (BOM : Browser Object Model)

JS

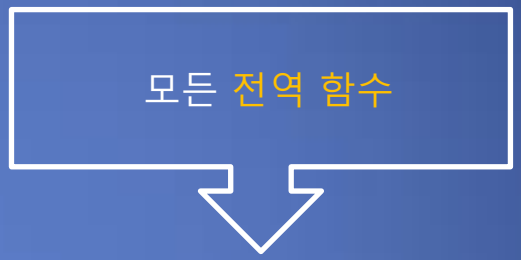
종류	설명
window 객체	브라우저의 내장 객체 중 최상위 객체
브라우저 객체 모델	문서를 제외한 주소, 화면 전체, 기록, 브라우저에 관련된 객체들의 집합을 의미
location 객체	현재 URL에 대한 정보를 가지고 있는 객체
navigator 객체	웹 문서를 실행하고 있는 브라우저에 대한 정보를 가지고 있는 객체
history 객체	브라우저 방문 기록에 대한 정보를 가지고 있는 객체
screen 객체	브라우저가 실행하고 있는 운영체제의 화면에 대한 정보를 가지고 있는 객체

브라우저 객체 (BOM : Browser Object Model)

JS

- **window 객체**
  - ✓ 브라우저의 내장 객체 중 최상위 객체
  - ✓ 모든 전역 객체, 함수, 변수는 자동적으로 window 객체에 속함

window 객체의 **속성**



window 객체의 **메서드**

- ✓ window를 생략한 형태로 window 객체와 메서드를 사용 가능
- ✓ 공식적인 표준은 없으나, 주요 브라우저들에서 지원되고 있음

브라우저 객체 (BOM : Browser Object Model)

JS

- window 객체 메서드 종류

종류	설명
open()	새 창을 열 때 사용
alert()	경고 창을 띄움
prompt()	질의응답 창을 띄움
confirm()	확인/취소 창을 띄움
moveTo()	창의 위치를 이동시킬 때 사용
resizeTo( )	창의 크기를 변경할 때 사용
setInterval( )	일정 간격으로 계속하여 실행문을 실행시킬 때 사용
setTimeout( )	일정 간격으로 한 번만 실행문을 실행시킬 때 사용

브라우저 객체 (BOM : Browser Object Model)

JS

- **open( )**

```
window.open("url 경로", "창 이름", "옵션설정")
```

<예시>

```
window.open("https://www.naver.com", "naver", "width=400, height=500, left=50, top=10,
scrollbars=no, toolbars=no, location=no");
```

브라우저 객체 (BOM : Browser Object Model)

JS

- open( ) 메서드 이용

  - 선택 사항인 4개의 매개 변수를 가짐
    - URL
      ① 새롭게 생성할 브라우저 창의 주소를 입력
      ② 입력하지 않으면 빈 브라우저 창이 생성
    - name
      ① 새로 생성될 창의 이름을 지정하는 것
      ② form 또는 anchor의 TARGET 속성을 위한 값
    - features
      ① 창의 위치와 크기 및 모양을 지정
    - replaces
      ① true와 false 중 선택
      ② 창의 히스토리 리스트를 새로운 것으로 대체할 것인지 현재의 것으로 유지할 것인지 결정
  - 주요 브라우저들에서 지원 가능

브라우저 객체 (BOM : Browser Object Model)

JS

- open( ) 의 속성

```
window.open("http://www.naver.com", "naver", "width=400, height=500, left=50, top=10,
scrollbars=no, toolbars=no, location=no");
```

속성	설명
width	창의 너비를 설정
height	창의 높이를 설정
left	창의 좌우 수평 위치를 설정
top	창의 상하 수직 위치를 설정
location	창의 url 주소 입력 영역 노출 여부 결정
status	창의 상태 영역 노출 여부를 결정
scrollbars	창의 스크롤 막대 노출 여부를 결정
toolbars	창의 도구상자 노출 여부를 결정

✓ location, status, scrollbars, toolbars의 상태값이 no 일 경우 노출, yes 일 경우 숨김

브라우저 객체 (BOM : Browser Object Model)

JS

- open( ) 의 속성 - 실습

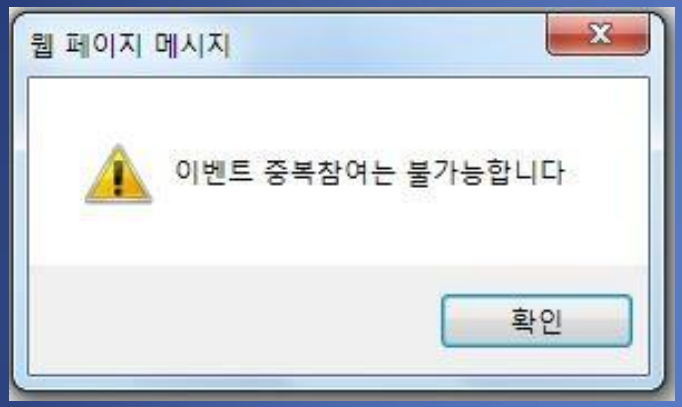
```
<script type="text/javascript">
//< ![CDATA[
var pop=window.open("http://www.naver.com", "pop1", "width=300, height=400, left=300,
top=10, scrollbars=no, toolbar=no, location=no");
//]]>
</script>
```

브라우저 객체 (BOM : Browser Object Model)

JS

- alert( )

```
alert("경고 메시지");
```



<예시>

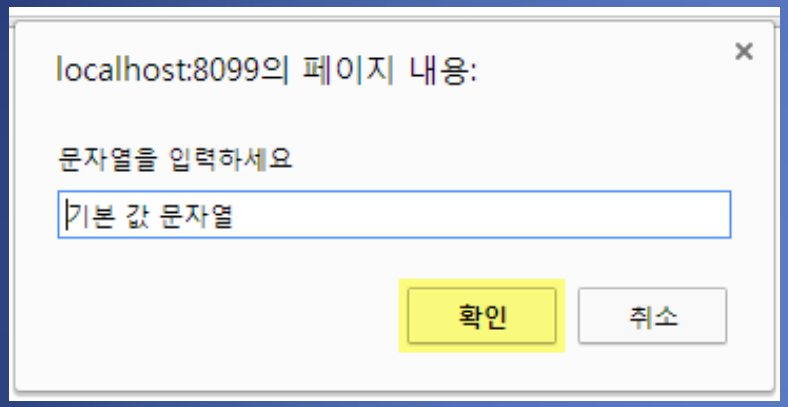
```
alert("이벤트 중복참여는 불가능합니다");
```

브라우저 객체 (BOM : Browser Object Model)

JS

- **prompt( )**

```
prompt("질의 내용", "기본값");
```



<예시>

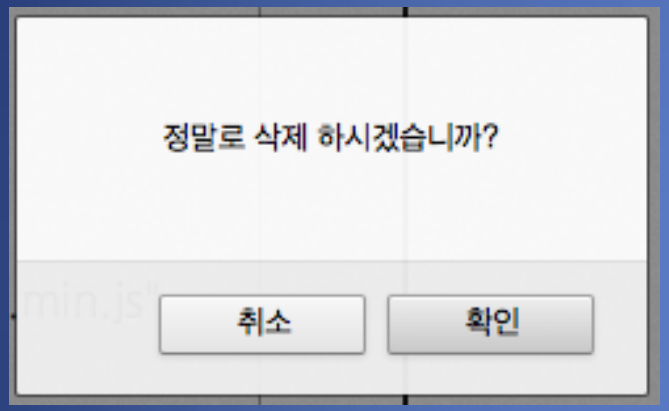
```
prompt("당신의 연령은", "0");
```

브라우저 객체 (BOM : Browser Object Model)

JS

- **confirm( )**

```
confirm("질의 내용");
```



<예시>

```
confirm("정말로 회원을 탈퇴하겠습니까?");
```

브라우저 객체 (BOM : Browser Object Model)

JS

- moveTo( )

```
moveTo(x의 위치값, y의 위치값);
```

<예시>

```
moveTo(100, 200);
```

- resizeTo( )

```
resizeTo(너비값, 높이값);
```

<예시>

```
resizeTo(100, 200);
```

브라우저 객체 (BOM : Browser Object Model)

JS

- moveTo( ), resizeTo( ) - 실습

```
<script type="text/javascript">
//< ![CDATA[
var pop=window.open("window.html", "pop", "width=300, height=400, left=300, top=10,
scrollbars=no, toolbar=no, location=no");
//]]>
</script>
<body>
        <p><button onclick="pop.moveTo(200,300); pop.focus();">팝업 위치 변경
</button></p>
        <p><button onclick="pop.resizeTo(200,300); pop.focus();">팝업 크기 변경
</button></p>
</body>
```

브라우저 객체 (BOM : Browser Object Model)

JS

- **setInterval( )**

```
var 참조변수=setIntreval('스크립트 실행문', 시간간격);
```

- 일정 시간 간격으로 스크립트 문을 반복하여 실행시킬 때 사용

<예시>

```
var t=setInterval('i++', 3000); //1초마다 변수 i 값이 1씩 증가
```

- **clearInterval( )**

```
clearIntreval(참조변수);
```

- setInterval( )을 적용하여 일정 시간 간격으로 반복하고 있는 스크립트 문을 중단시킬 때 사용

<예시>

```
clearIntreval(t); //t에 참조되어 적용한 setIntreval( )을 삭제
```

브라우저 객체 (BOM : Browser Object Model)

JS

- setInterval( ) / clearInterval( )

```
<p>현재 시각을 3초 간격으로 보여주기:</p>
<p id="presentTime"></p>

<script>
var myVar = setInterval(function(){ myTimer() }, 3000);

function myTimer() {
    var d = new Date();
    var t = d.toLocaleTimeString();
    document.getElementById("presentTime").innerHTML = t;
}
</script>
```

브라우저 객체 (BOM : Browser Object Model)

JS

- **setTimeout ( )**

```
var 참조변수=setTimeout('스크립트 실행문', 시간간격);
```

- 일정 시간 간격으로 스크립트 문을 한 번만 실행시킬 때 사용

<예시>

```
var t=setTimeout("console.log(++i)", 5000); //5초 간격 이후 단 한 번만 변수 i값을 1씩 증가
```

- **clearTimeout ( )**

```
clearTimeoout (참조변수);
```

<예시>

```
clearTimeout(t); //t에 참조되어 적용한 setTimeout( )을 삭제
```

브라우저 객체 (BOM : Browser Object Model)

JS

- **setTimeout ( ) / clearTimeout ( )**

```
<script>
  var i =0;
  var auto=setTimeout(console.log(++i), 3000);
</script>
```

✓ 단 한 번 3초 간격으로 실행문 console.log(++i) 를 실행한 후 마침

screen 객체

JS

- 모니터의 가용 너비 / 높이 / 컬러의 값을 반환 (확인용)
- 기본형

```
screen.속성;
```

```
screen.width; //모니터의 너비값을 반환
```

- screen 객체의 속성 종류

종류	설명
screen.width	화면의 너비값
screen.height	화면의 높이값
screen.availWidth	작업표시줄 제외한 화면 너비값
screen.availHeight	작업표시줄 제외한 화면 높이값
screen.colorDepth	사용자 모니터의 표현 가능한 컬러 bit를 반환

screen 객체

JS

- screen 객체 실습

```
<script type="text/javascript">
//<![CDATA[
document.write("width:" + screen.width + "<br/>");
document.write("height:" + screen.height + "<br/>");
document.write("availWidth:" + screen.availWidth + "<br/>");
document.write("availHeight:" + screen.availHeight + "<br/>");
//]]>
</script>
```

- availWidth 와 availHeight는 <레이어 팝업>을 모니터의 중앙에 위치해야 할 때, 주로 사용

location 객체

JS

- location 객체는 사용자 브라우저의 주소창에 url에 대한 정보(속성)와 새로고침 기능을 제공하는 객체
- 기본형

```
location.속성;  또는  location.메서드( );
```

location 객체

JS

- location 객체의 속성/메서드 종류

종류	설명
location.href	주소 영역에 참조 주소를 설정하거나 url을 반환
location.hash	url에 해시값(#에 명시된 값)을 반환 예)http://www.myhome.net#search
location.hostname	url에 호스트 이름을 설정하거나 반환
location.host	url에 호스트 이름과 포트 번호를 설정하거나 반환
location.port	url에 포트 번호를 반환
location.protocol	url에 프로토콜을 반환
location.search	url에 쿼리(요청값)를 반환
location.reload()	<새로 고침>이 발생

location 객체

JS

- location 객체의 실습

```
<script>
		document.write("href : "+location.href, "<br/>");
		document.write("hash : "+location.hash, "<br/>");
		document.write("hostname : "+location.hostname, "<br/>");
		document.write("host : "+location.host, "<br/>");
		document.write("protocol : "+location.protocol, "<br/>");
</script>
```

history 객체

JS

- 방문한 사이트 중 이전에 방문한 사이트와 다음 방문한 사이트로 돌아감
- 기본형

```
history.속성;  또는  history.메서드;
```

```
history.back();  // 바로 이전 페이지로 이동
```

- history 객체의 속성/메서드 종류

종류	설명
history.back();	이전 방문한 페이지로 이동
history.forward();	다음 방문한 페이지로 이동
history.go(이동 숫자);	이동숫자가 -2이면 2단계 이전 페이지으로 이동
length	방문기록에 저장된 목록의 개수를 반환

history 객체

JS

- 첫 번째 페이지 파일 (his_o_1.html)

```
<body>
        <h1>첫 번째 페이지</h1>
        <a href="his_o_2.html">2페이지로 이동</a>
</body>
```

- 두 번째 페이지 파일 (his_o_2.html)

```
<body>
        <h1>두 번째 페이지</h1>
        <a href="his_o_3.html">3페이지로 이동</a>
        <button onclick="history.back( );">이전 페이지</button>
        <button onclick="history.forward( );">다음 페이지</button>
</body>
```

- 세 번째 페이지 파일 (his_o_3.html)

```
<body>
        <h1>세 번째 페이지</h1>
        <button onclick="history.go(-1);">1단계 이전 페이지</button>
        <button onclick="history.go(-2);">2단계 이전 페이지</button>
</body>
```

navigator 객체

JS

- 현재 방문자가 사용하는 브라우저 정보와 운영체제의 정보를 제공하는 객체
- 기본형

```
navigator.속성;
```

```
navigator.userAgent;  // 방문자의 브라우저와 운영체제 정보를 제공
```

navigator 객체

JS

- navigator 객체의 속성

종류	설명
navigator.appCodeName	방문자의 브라우저 코드명을 반환
navigator. appName	방문자의 브라우저 이름을 반환
navigator.appVersion	방문자의 브라우저 버전 정보를 반환
navigator.language	방문자의 브라우저 사용 언어를 반환
navigator.product	방문자의 브라우저 사용 엔진을 반환
navigator.platform	방문자의 브라우저를 실행하는 운영체제를 반환
navigator.userAgent	방문자의 브라우저와 운영체제 종합 정보를 제공

navigator 객체

JS

- navigator 객체의 실습

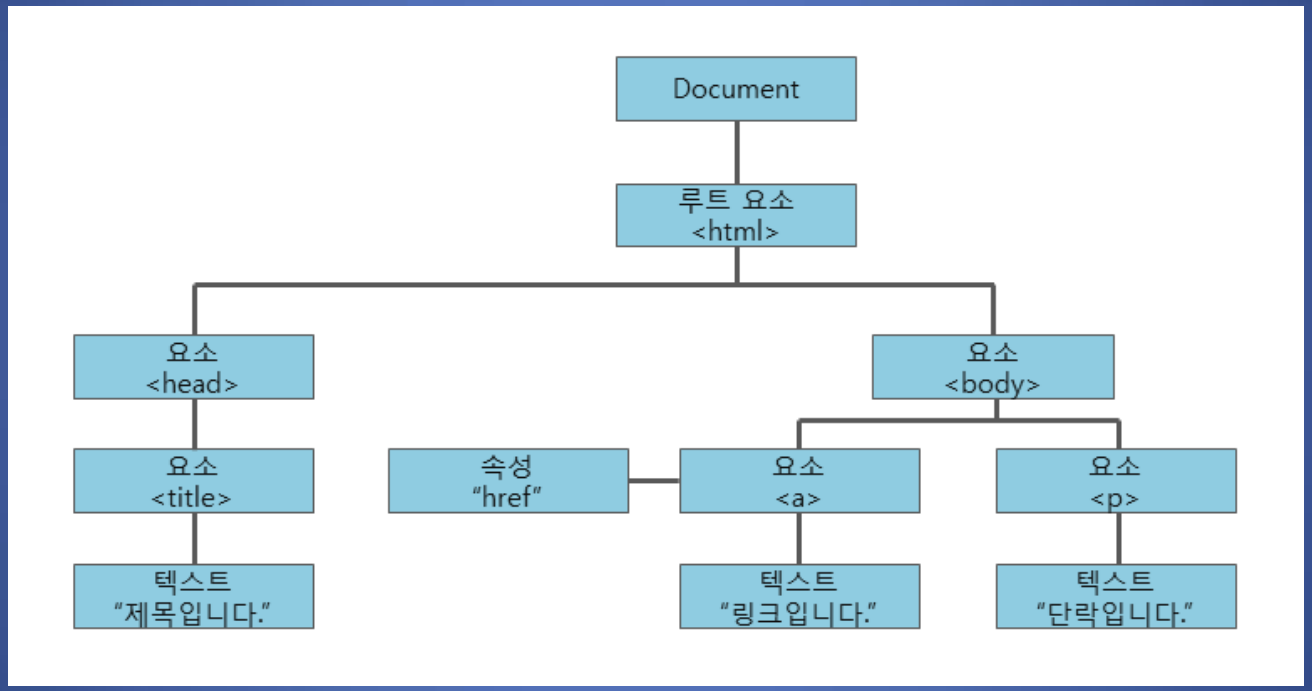
```
<script>
		document.write("appCodeName : "+navigator.appCodeName, "<br/>");
		document.write("appName : "+navigator.appName, "<br/>");
		document.write("appVersion : "+navigator.appVersion, "<br/>");
		document.write("language : "+navigator.language, "<br/>");
		document.write("product : "+navigator.product, "<br/>");
		document.write("platform : "+navigator.platform, "<br/>");
		document.write("userAgent : "+navigator.userAgent, "<br/>");
</script>
```

문서 객체 (DOM : Document Object Model)

JS

- 각 태그마다 기능과 속성을 갖고 있음.
  (예를 들어, <img> 태그는 이미지를 넣기 위한 태그이며, 속성으로는 src와 alt를 포함하고 있음)

- 문서 객체 모델을 사용하는 주요 이유는 자바스크립트로 문서 객체를 선택하고 속성 또는 스타일을 적용하기 위함



- 트리 구조로 구성
- 이 구조에서 HTML은 뿌리가 되며, 뿌리로부터 가지처럼 뻗어 나가는 것을 **노드(node)**라고 부름

문서 객체 (DOM : Document Object Model)

JS

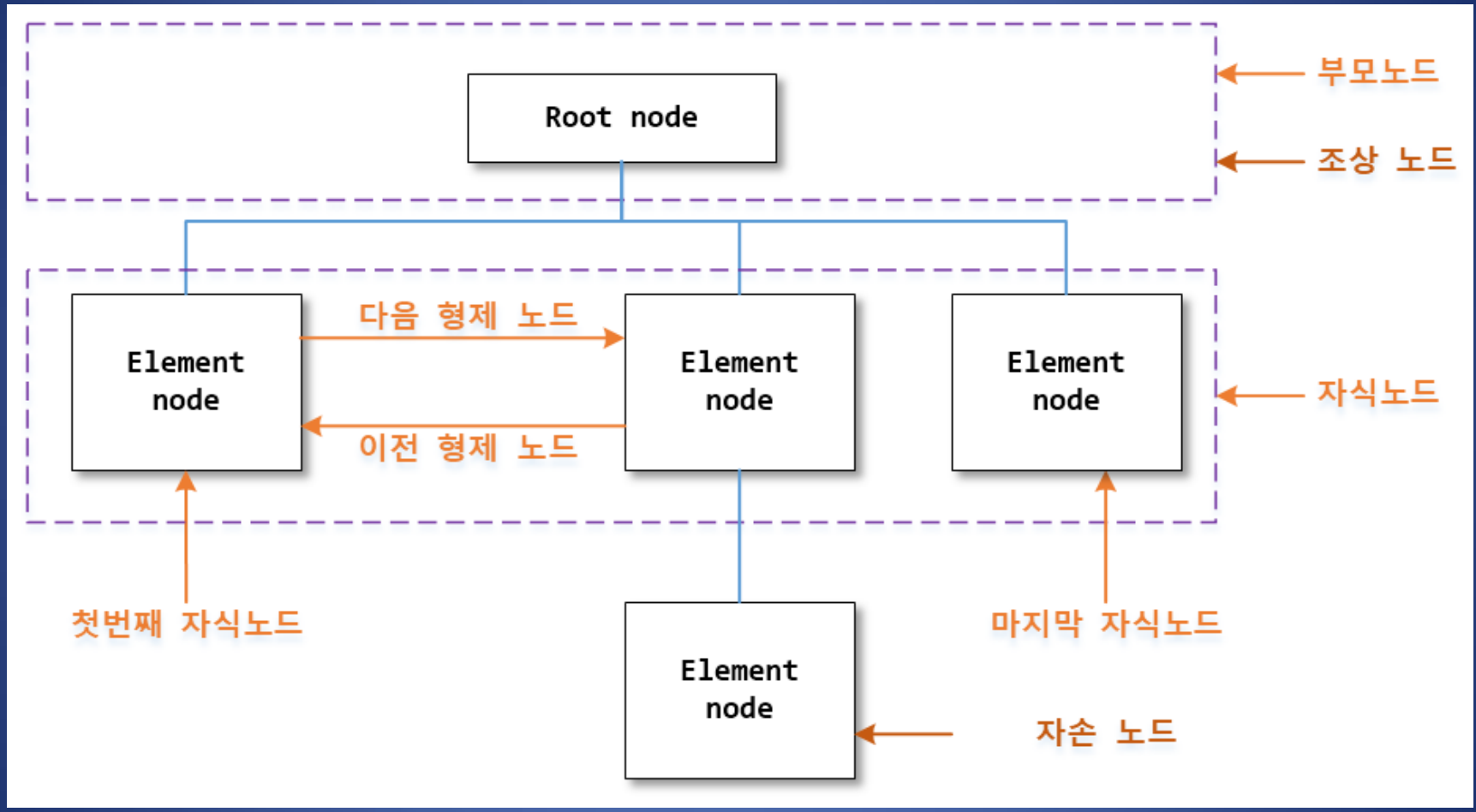
- 노드의 종류

종류	설명
요소노드 (Element node)	HTML 태그를 연결
텍스트 노드 (Text node)	텍스트를 연결
속성노드 (Attribute node)	HTML 태그에 속성을 연결

문서 객체 (DOM : Document Object Model)

JS

- 인접요소 관계파악



문서 객체 (DOM : Document Object Model)

JS

- 선택자
  - ✓ 일반적인 HTML의 내부 혹은 CSS에 작성한 스타일을 '정적 스타일'로 표현한다면 자바스크립트의 경우 사용자의 반응에 따라 다양하게 변화시킬 수 있기 때문에 '동적 스타일'이라고도 표현할 수 있음

- 선택자의 종류
  - ✓ 직접 선택자 - 원거리 선택자

종류	설명
document.getElementById("아이디명")	아이디를 이용해 요소를 선택
document.getElementsByTagName("요소(태그)명")	요소 이름을 통해 요소를 선택
document.getElementsByClassName("클래스명")[인덱스번호]	클래스명을 이용해 선택
document.formName.inputName	폼 요소에 name을 이용해 요소를 선택

문서 객체 (DOM : Document Object Model)

JS

- 선택자의 종류

✓ 간접(인접 관계) 선택자 - 근거리 선택자

종류	설명
parentNode	선택한 요소의 부모 요소를 선택 - 노드 접근 무시
childNodes	선택한 요소의 모든 자식 요소를 선택 선택한 모든 요소는 배열 객체로 저장 - (HTML 코딩상 엔터값)노드 접근
children	선택한 요소의 자식 요소인 태그만 선택 선택한 모든 요소는 배열 객체로 저장 - 노드 접근 무시
firstChild	선택한 요소의 첫 번째 자식 요소만 선택 - 노드 접근
lastChild	선택한 요소의 마지막 자식 요소만 선택 - 노드 접근
previousSibling	선택한 요소의 이전에 오는 형제 요소만 선택 - 노드 접근
nextSibling	선택한 요소의 다음에 오는 형제 요소만 선택 - 노드 접근

문서 객체 (DOM : Document Object Model)

JS

- 선택자의 적용과 위치 – 잘못된 사례
  - ✓ 선택자를 사용할 때, <head> 내부에 자바스크립트를 선언하게 되면 HTML의 요소를 읽어오는 것보다 먼저 실행되어 문서 객체를 선택할 수 없게 됨

```
<html>
<head>
        자바스크립트 선언
</head>
<body>
        HTML  문단 태그 선언
</body>
</html>
```

  - ✓ 대부분의 자바스크립트는 위와 같이 실행
  - ✓ 자바스크립트를 <head> 내부에 넣으면 HTML 문단 태그가 로딩되기 전에 실행되기 때문에 적용이 안됨

문서 객체 (DOM : Document Object Model)

JS

- 선택자의 적용과 위치 – 잘못된 사례

```
<html>
        <head>
                <script type="text/javascript">
                //<![CDATA[
                documen.getElementById("title").style.color="green"
                //]]>
                </script>
        </head>
        <body>
                <h1 id="title">선택자를 정합니다.</h1>
        </body>
</html>
```

문서 객체 (DOM : Document Object Model)

JS

- 선택자의 적용과 위치 – 올바른 사례 ①

```
<html>
        <head>
        </head>
        <body>
                HTML  문단 태그 선언
                <script type="text/javascript">
                        스크립트 실행문;
                </script>
        </body>
</html>
```

✓ HTML 문단 태그가 로딩된 후에 실행

문서 객체 (DOM : Document Object Model)

JS

- 선택자의 적용과 위치 – 올바른 사례 ①

```
<html>
        <head>
        </head>
        <body>
                <h1 id="title">선택자를 정합니다.</h1>
                <script type="text/javascript">
                //<![CDATA[
                document.getElementById("title").style.color="green";
                //]]>
                </script>
        </body>
</html>
```

문서 객체 (DOM : Document Object Model)

JS

- 선택자의 적용과 위치 – 올바른 사례 ②

```
<html>
<head>
        <script type="text/javascript">
                window.onload=function(){
                        스크립트 실행문;
                }
        </script>
</head>
<body>
        HTML  문단 태그 선언
</body>
</html>
```

- ✓ window.onload=function(){...}을 이용하여선택문 작성
- ✓ window.onload=function(){...}은 '모든 문서 객체를 로딩한 후에 중괄호{ } 내부에 있는 실행문을 실행하시오'라는 의미

문서 객체 (DOM : Document Object Model)

JS

- 선택자의 적용과 위치 – 올바른 사례 ②

```
<html>
<head>
        <script type="text/javascript">
                //<![CDATA[
                window.onload=function(){
                        document.getElementById("title").style.color="green";
                }
                //]]>
        </script>
</head>
<body>
        <h1 id="title">선택자를 정합니다.</h1>
</body>
</html>
```

문서 객체 (DOM : Document Object Model)

JS

- 직접 선택자 – id 선택
  - ✓ 태그에 지정한 id 속성값을 이용하여 특정 요소에 접근 가능한 선택자
  - ✓ 기본형식

```
선택대상.getElementById("아이디명");
```

  - ✓ 적용예시

```
<body>
        <p id="title">getElementById는 id의 속성값이 동일한 태그에 적용</p>
        <script>
                var myObj=document.getElementById("title");
                myObj.style.color="#ff0000“;
        </script>
</body>
```

문서 객체 (DOM : Document Object Model)

JS

- 직접 선택자 – 요소(태그) 선택
  - ✓ 태그로 접근할 수 있는 선택자이며, 이 때 같은 이름의 태그들이 여러 개가 존재하면 모두 배열로 저장됨. 따라서, getElements 처럼 맨 뒤에 's'가 붙음.
  - ✓ 기본형식

```
선택대상.getElementsByTagName("태그명")[인덱스번호];
```

  - ✓ 적용예시

```
<body>
        <p id="title">getElementsByTagName는 동일한 태그에 적용</p>
        <script>
                var myObj=document.getElementsByTagName("p")[0];
                myObj.style.color="#ff0000";
        </script>
</body>
```

문서 객체 (DOM : Document Object Model)

JS

- 직접 선택자 – 클래스 선택
  - ✓ 클래스명으로 접근할 수 있는 선택자이며, 이 때 같은 클래스 명이 여러 개가 존재하면 모두 배열로 저장됨. 따라서, getElements 처럼 맨 뒤에 's'가 붙음.
  - ✓ 기본형식

```
선택대상.getElementsByClassName("태그명")[인덱스번호];
```

  - ✓ 적용예시

```
<body>
        <p class="txt">getElementsByClassName은 지정한 클래스에 적용</p>
        <script>
                var myObj=document.getElementsByClassName("txt")[0];
                myObj.style.color="#ff0000";
        </script>
</body>
```

문서 객체 (DOM : Document Object Model)

JS

- 직접 선택자 – 요소(태그) 선택
  - ✓ 적용예시

```
<body>
        <div id="title">
                <p>회사소개</p>
                <p>회사연혁</p>
                <p>CEO소개</p>
                <p>회사위치</p>
        </div>
        <script type="text/javascript">
                var Obj=document.getElementById("title");
                Obj.getElementsByTagName("p")[0].style.color="#00ffff";
                Obj.getElementsByTagName("p")[1].style.color="#ff0000";
                Obj.getElementsByTagName("p")[2].style.color="#ff0077";
                Obj.getElementsByTagName("p")[3].style.color="#ff7700";
        </script>
</body>
```

문서 객체 (DOM : Document Object Model)

JS

- 직접 선택자를 사용한 문서 객체 CSS 적용

  ✓ 기본형

```
document.직접 선택자(선택자 메서드).style.속성=”값”;
```

  ✓ 적용 예시 - 01

```
<h1 id="title">선택자 CSS 기본형-01</h1>
<script type="text/javascript">
        document.getElementById("title").style.color="blue";
</script>
```

문서 객체 (DOM : Document Object Model)

JS

- 직접 선택자를 사용한 문서 객체 CSS 적용
  - ✓ 적용 예시 - 02

```
<p>회사소개</p>
<p>회사연혁</p>
<p>Ceo소개</p>
<p>회사위치</p>
<script type="text/javascript">
        document.getElementsByTagName("p")[3].style.color="blue";
</script>
```

- ✓ document.getElementsByTagName("p")[3] 는 배열을 적용하여 네 번째 <p>를 선택
- ✓ style.color는 스타일 중 color를 선언
- ✓ "blue" 또는 색상코드로 작성해도 무방함

문서 객체 (DOM : Document Object Model)

JS

- 직접 선택자를 사용한 문서 객체 CSS 적용
  - ✓ 적용 예시 - 03

```
<div id="title">
            <p>회사소개</p>
            <p>회사연혁</p>
            <p>CEO소개</p>
            <p>회사위치</p>
</div>
<script type="text/javascript">
            var Obj=document.getElementById("title");
            Obj.getElementsByTagName("p")[1].style.color="#ff0000";
</script>
```

- ✓ Obj 라는 변수명에 선언할 곳을 저장
- ✓ Obj 라는 변수 중에서 <p> 태그에 스타일 적용

문서 객체 (DOM : Document Object Model)

JS

- 직접 선택자를 사용한 문서 객체 CSS 적용
  - ✓ 자바스크립트 내부에서 CSS 사용시 중점사항

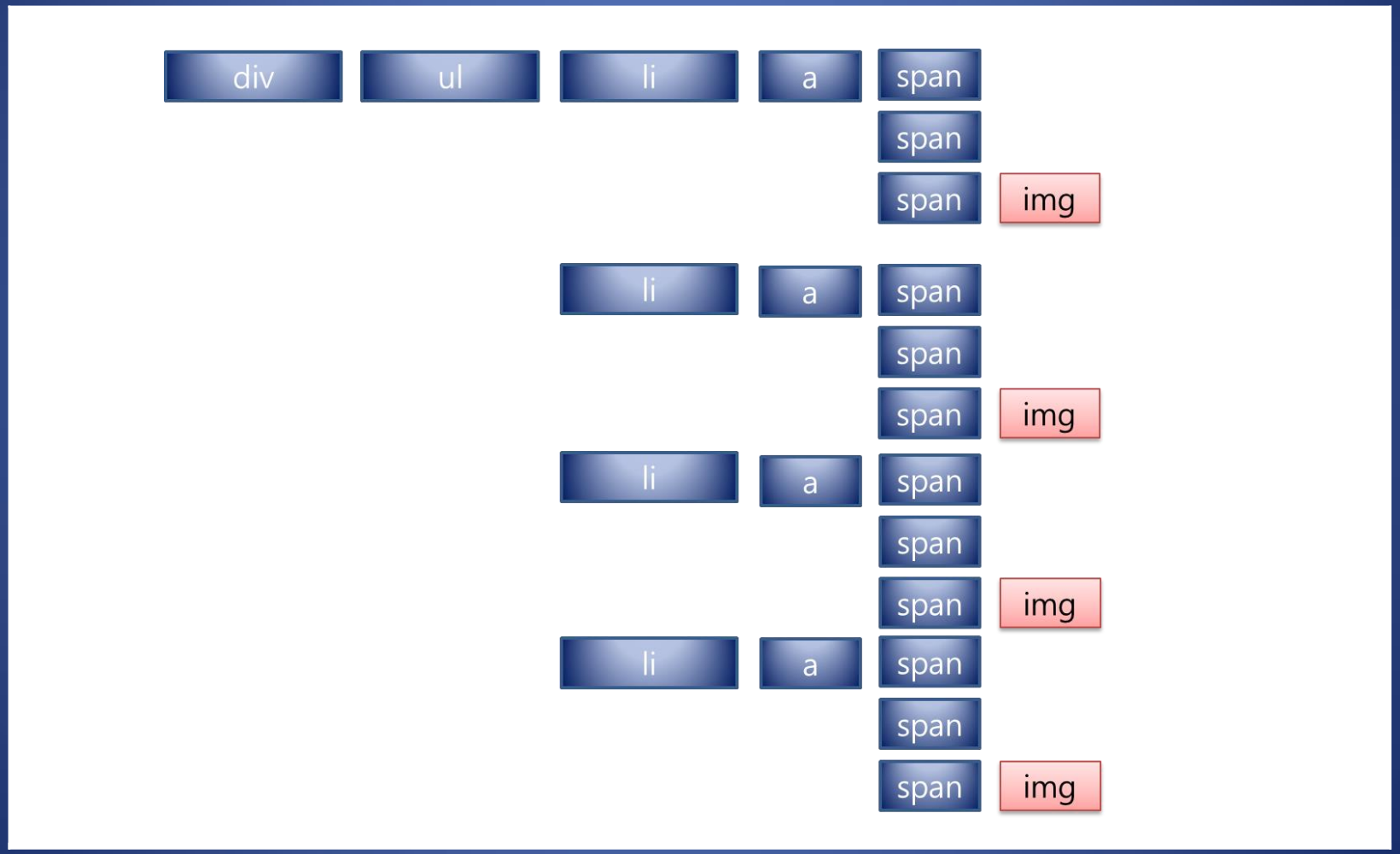
(X)HTML에서 CSS 사용시	**자바스크립트에서 CSS 사용시**
background-color	**backgroundColor**
font-size	**fontSize**

- ✓ 자바스크립트에서 스타일(CSS) 작성시 주의할 점, 배경색이나 글자 크기의 경우 하이픈('-')이 포함되어 있는데, 이는 산술 연산자로 인식 됨.

문서 객체 (DOM : Document Object Model)

JS

- 간접 선택자 (인접선택자)
  - ✓ 특정 요소를 기준으로 상대적인 다른 요소를 선택할 때 사용



문서 객체 (DOM : Document Object Model)

JS

- 간접 선택자 – parentNode(부모 요소 선택자)
  - ✓ id 값을 기준으로 상위의 요소에 접근 할 때 사용
  - ✓ 부모의 요소 전체가 아닌 바로 상위의 부모 요소만을 선택
  - ✓ HTML 코딩상 엔터값(공백)에 의한 빈 노드로 접근 없이 바로 부모의 영역으로 선택
  - ✓ 기본형식

```
document.getElementById("list").parentNode;
```

- 간접 선택자 – childNodes(자식 요소 선택자)
  - ✓ id 값을 기준으로 하위의 요소에 접근 할 때 사용
  - ✓ HTML 코딩상 엔터값(공백)이 존재할 경우 빈 노드로 접근
  - ✓ 기본형식

```
document.getElementById("list").childNodes;
```

문서 객체 (DOM : Document Object Model)

JS

- 간접 선택자 – children(자식 요소 선택자)
  - ✓ id 값을 기준으로 하위의 요소에 접근 할 때 사용
  - ✓ 기본형식

```
childObj = document.getElementById("list").children;
```

- 자식 요소 선택자인 childNodes과 children의 차이
  - ✓ childNodes 는 자식 요소 전체의 노드를 반환하기 때문에 코딩상의 공백문자에 대한 값도 반환 (IE 8 이하 버전에서는 공백 문자 무시)
  - ✓ Children은 자식 요소만을 반환하기 때문에 요소인 Element 만을 반환

문서 객체 (DOM : Document Object Model)

JS

- 간접 선택자 – firstChild (첫 번째 자식 요소 선택자)
  - ✓ id 값을 기준으로 하위의 요소 중 첫 번째 자식 요소에만 접근 할 때 사용
  - ✓ HTML 코딩상 엔터값(공백)이 존재할 경우 빈 노드로 접근
  - ✓ 기본형식

```
document.getElementById("list").firstChild;
```

- 간접 선택자 – lastChild (마지막 자식 요소 선택자)
  - ✓ id 값을 기준으로 하위의 요소 중 마지막 자식 요소에만 접근 할 때 사용
  - ✓ HTML 코딩상 엔터값(공백)이 존재할 경우 빈 노드로 접근
  - ✓ 기본형식

```
document.getElementById("list").lastChild;
```

문서 객체 (DOM : Document Object Model)

JS

- 간접 선택자 – previousSibling (이전 형제 요소 선택자)
  - ✓ id 값을 기준으로 이전 형제 요소에 접근 할 때 사용
  - ✓ HTML 코딩상 엔터값(공백)이 존재할 경우 빈 노드로 접근
  - ✓ 기본형식

```
var prev = document.getElementById("title").previousSibling;
```

- 간접 선택자 – nextSibling (다음 형제 요소 선택자)
  - ✓ id 값을 기준으로 다음 형제 요소에 접근 할 때 사용
  - ✓ HTML 코딩상 엔터값(공백)이 존재할 경우 빈 노드로 접근
  - ✓ 기본형식

```
var next= document.getElementById("title").nextSibling;
```

문서 객체 (DOM : Document Object Model)

JS

- 간접 선택자를 사용한 문서 객체 CSS 적용
  - ✓ 직접 선택자를 사용하여 원거리 요소까지 선택한 후 인접 관계 선택자(간접 선택자)를 사용
  - ✓ 기본형식

```
document.직접 선택자(선택 메서드).인접 관계 선택자.style.속성=값;
```

  - ✓ 적용예시

```
<h1 id="title"><a href="#">네이버 바로가기</a></h1>
```

```
document.getElementById("title").children[0].style.backgroundColor="#ff0000";
```

- 해석 : 아이디 값이 "title"인 태그에 첫 번째 자식 요소인 <a>에 스타일을 이용해 빨간 배경을 적용

문서 객체 (DOM : Document Object Model)

JS

✓ 적용예시

```
<div id="menu">
        <p>네이버 바로가기</p>
        <p>다음 바로가기</p>
        <p>구글 바로가기</p>
        <p>네이트 바로가기</p>
</div>
```

```
var ch = document.getElementsByTagName("p")[1];
ch.nextSibling.nextSibling.style.backgroundColor="#ffff00";
```

- 해석 1 : <p> 태그 중 두 번째 <p> 태그를 변수 ch로 저장
- 해석 2 : 두 번째 <p> 태그 다음에 오는 요소의 배경을 노란색으로 적용
- 해석 3 : nextSibling의 경우 노드를 기준으로 적용하기 때문에 공백문자에 적용됨. 그래서 한 번 더 적용해 주어야 함.

문서 객체 (DOM : Document Object Model)

JS

- 요소 속성값 변경, 생성 및 불러오는 방법

종류	설명
요소 선택.속성명	요소의 지정한 속성값을 불러옴
요소 선택.속성명=“새 값”	요소의 지정한 속성값을 새 값으로 변경 또는 생성
요소 선택.getAttribute(“속성”)	요소의 지정한 속성값을 불러옴
요소 선택.setAttribute(“속성”, “새값”)	요소의 지정한 속성값을 새 값으로 변경 또는 생성

문서 객체 (DOM : Document Object Model)

JS

- 요소 속성값 변경, 생성 및 불러오는 방법

  ✓ 속성 값 불러오기

```
<h1><img src="img/logo.png" id="main-logo" alt="메인 로고" /></h1>
<script>
            var m = document.getElementById("main-logo").src;
            var n = document.getElementById("main-logo").getAttribute("src");
            document.write("첫 번째 호출 : "+m +"<br/>"+"두 번째 호출 : "+n);
</script>
```

문서 객체 (DOM : Document Object Model)

JS

- 요소 속성값 변경, 생성 및 불러오는 방법

  ✓ 속성 값 변경하기

```
<h1><img src="img/logo.png" id="main-logo" alt="메인 로고" /></h1>
<script>
            var k = document.getElementById("main-logo").src="img/symbol.png";
            var j = document.getElementById("main-logo").setAttribute("src", "img/symbol.png");
            document.write("첫 번째 호출 변경 : "+k+"<br/>"+"두 번째 호출 변경 : "+j);
</script>
```

문서 객체 (DOM : Document Object Model)

JS

- 문서 객체의 하위 요소(innerHTML) 속성

종류	설명
요소 선택.innerHTML;	선택한 요소의 모든 하위 요소를 문자 데이터로 반환
요소 선택.innerHTML=새 요소;	선택한 요소의 전체 하위 요소를 새 요소로 변경 또는 생성

문서 객체 (DOM : Document Object Model)

JS

- 문서 객체의 하위 요소 (innerHTML) 속성

  ✓ 속성 변경하기

```
<h1 id="title"><a href="#">마이홈으로 바로가기</a></h1>
<script>
        var s=document.getElementById("title").innerHTML;
        var t=document.getElementById("title").innerHTML = "<strong>고객센터</strong>";
        document.write("첫 번째 호출 변경 : "+s +"<br/>"+"두 번째 호출 변경 : "+t);
</script>
```

문서 객체 (DOM : Document Object Model)

JS

- 문서 객체의 하위 요소 (innerHTML) 속성 – **실습 문제**

  ✓ id=context를 포함하고 있는 div 태그 내부에 img 태그를 활용한 이미지 넣기

```
<h4 id="context"></h4>
<script>
        ?
</script>
```

문서 객체 (DOM : Document Object Model)

JS

- 문서 객체 이벤트 핸들러 적용하기
  - ✓ 이벤트 : 방문자가 사이트에서 하는 행위

이벤트	설명
onclick	선택한 요소를 마우스로 클릭했을 때 이벤트 발생
onmouseover	선택한 요소에 마우스를 올렸을 때 이벤트가 발생
onmouseout	선택한 요소에 마우스가 벗어났을 때 이벤트가 발생
onsubmit	선택한 폼에 전송이 일어났을 때 이벤트가 발생

문서 객체 (DOM : Document Object Model)

JS

- 문서 객체 이벤트 핸들러 적용하기

  ✓ 기본형식

```
요소선택.이벤트 종류=function(){
            실행문
}
```

  ✓ 적용예시

```
<a href="#" id="btn_01">버튼</a>
<script>
        document.getElementById("btn_01").onclick=function(){
                alert("Welcome!");
        }
</script>
```

문서 객체 (DOM : Document Object Model)

JS

- 문서 객체 이벤트 핸들러 적용하기
  - ✓ 실습

```
<body>
<h1 id="title">마우스 아웃</h1>
<a href="#" id="btn"><img src="img-DOM/btn_out.gif" alt="버튼" /></a>
<script type="text/javascript">
//<![CDATA[
   var theBtn=document.getElementById("btn");
   theBtn.onmouseover=function(){
      document.getElementById("title").innerHTML="마우스 오버";

      theBtn.firstChild.src="img-DOM/btn_over.gif";
   }

   theBtn.onmouseout=function(){
      document.getElementById("title").innerHTML="마우스 아웃";

      theBtn.firstChild.src="img-DOM/btn_out.gif";
   }
//]]>
</script>
</body>
```

문서 객체 (DOM : Document Object Model)

JS

- 폼 요소 선택자
  - ✓ 회원가입 등의 양식을 적용할 때 약관 동의 및 가입한 양식이 부합한지에 대해 제어
  - ✓ 폼 요소 선택 방식

구분	종류	설명
입력요소 선택자	document.getElementById("아이디명")	폼 요소를 아이디로 선택할 때 사용
	document.폼 이름.입력 요소 이름	폼 요소를 이름으로 선택할 때 사용
Select option 선택자	document.폼 이름.입력 요소 이름.option[index]	<select>에 <option>을 선택할 때 사용
	var i=document.폼 이름.입력 요소 이름.selectedIndex; document.폼이름.입력 요소 이름.option[i];	<select>에 선택된 <option>을 선택할 때 사용

문서 객체 (DOM : Document Object Model)

JS

- 폼 요소 선택자
  - ✓ 적용예시

```
<body>
<form method="post" action="#" name="log_f">
        <fieldset>
                <legend>회원 로그인</legend>
                <p>
                        <label for="my_id">아이디</label>
                        <input type="text" name="user_id" id="my_id"/>
                        <label for="my_pw">비밀번호</label>
                        <input type="password" name="user_pw" id="my_pw"/>
                </p>
                <input type="submit" value="login"/>
        </fieldset>
</form>
<script>
        document.log_f.onsubmit=function(){
                var id_v=document.log_f.user_id.value;
                var pw_v=document.log_f.user_pw.value;
                alert(id_v);
                alert(pw_v);
        }
</script>
</body>
```

문서 객체 (DOM : Document Object Model)

JS

- 폼 요소 선택자
  - ✓ 회원가입 등의 양식을 적용할 때 약관 동의 및 가입한 양식이 부합한지에 대해 제어
  - ✓ 폼 요소 속성 종류

구분	종류	설명
전체	value	입력 요소의 value 속성을 변경하거나 값을 가져옴
	disable	입력 요소의 disable 속성을 변경하거나 현재 상태의 값을 가져옴
	defaultValue	입력 요소의 초기 value 값을 가져옴
선택박스	selected	<select> 태그에 <option> 선택의 상태를 가져옴. 선택시 true, 미선택시 false를 반환
체크박스 라디오버튼	checked	체크박스 또는 라디오버튼 태그에 체크 상태 값을 반환하거나 체크 유무를 제어. 체크시 true, 미체크시 false를 반환

문서 객체 (DOM : Document Object Model)

JS

- 폼 요소 선택자
  - 적용예시 - 01

```
<body>
<form method="post" action="#" name="log_s">
        <p><input type="text" name="user_name" id="user_name" value="강철수"/><p>
</form>
<script>
        document.log_s.user_name.defaultValue;
</script>
</body>
```

✓ 초기 value 값을 반환

문서 객체 (DOM : Document Object Model)

JS

- 폼 요소 선택자
  - 적용예시 - 02

```
<body>
<form method="post" action="#" name="log_f">
        <label><이용약관></label>
        <p>본 계약사항은 구입일로부터 1개월 내에 취소 및 반품처리가 가능합니다.</p>
        <input type="checkbox" name="chk1" value="c1" checked="checked"/> 약관 동의
</form>
<script>
        document.log_f.chk1.checked;
</script>
</body>
```

✓ 체크되어 있으므로 체크 값을 반환

문서 객체 (DOM : Document Object Model)

JS

- 폼 요소 선택자

```
<form action="#" method="get" name="f1">
<fieldset>
  <legend>폼 요소2</legend>
  <div id="terms">
    회사는 법령이 정하는 바에 따라 "회원"의 개인정보를 보호하기
    위해 노력합니다. 개인정보의 보호 및 사용에 대해서는 관련법
    및 회사의 개인정보취급방침이 적용됩니다.
  </div>
  <p>
    <input type="checkbox" name="accept" id="accept" />
    <label for="accept">약관 동의합니다.</label></p>
  <p>
    <input type="checkbox" name="allChk" id="allChk" />
    <label for="allChk">전체선택</label><br />
    <input type="checkbox" name="subject1" id="subject1" value="s1" />
    <label for="subject1">과목1</label><br />
    <input type="checkbox" name="subject2" id="subject2" value="s2" />
    <label for="subject2">과목2</label><br />
    <input type="checkbox" name="subject3" id="subject3" value="s3" />
    <label for="subject3">과목3</label>
  </p>
  <p>
    <input type="submit" value="등록 완료" />
    <input type="reset" value="등록 취소" />
  </p>
</fieldset>
</form>
```

문서 객체 (DOM : Document Object Model)

JS

- 최종 실습

```
<script>
//<![CDATA[
//전체선택 체크박스를 클릭했을 때
document.f1.allChk.onclick=function(){
   if(this.checked){ //체크되어 있으면 true 반환
      document.f1.subject1.checked=true; //첫 번째 체크박스 체크됨
      document.f1.subject2.checked=true; //두 번째 체크박스 체크됨
      document.f1.subject3.checked=true; //세 번째 체크박스 체크됨
   }else{
      document.f1.subject1.checked=false; //첫 번째 체크박스 체크 해제됨
      document.f1.subject2.checked=false; //두 번째 체크박스 체크 해제됨
      document.f1.subject3.checked=false; //세 번째 체크박스 체크 해제됨
   }
}
// 폼요소에 전송버튼 눌렀을 때 실행
document.f1.onsubmit=function(){
   var act=document.f1.accept;
   if(act.checked==false){ //체크되어 있지 않으면 false
      alert("약관 동의해야 합니다!");
      act.focus(); //약관 동의 체크박스에 포커스(라인 생성)
      return false; //함수 종료
   }
}
//]]>
</script>
```

문서 객체 (DOM : Document Object Model)

JS

- 최종 실습 – 자동차 옵션 견적 뽑기

```
<body>
  <div><img src="img/car-001.jpg" alt="" /></div>
  <table>
  <thead>
    <tr>
      <th>구분</th>
      <th>가격</th>
      <th>선택</th>
    </tr>
  </thead>
  <tfoot>
    <tr>
      <td><label for="total">총액</label></td>
      <td colspan="2"><input type="text"name="total" id="total" value="36000000"
readonly/></td>
    </tr>
  </tfoot>
  <tbody>
    <tr>
      <td><label for="option1">선루프</label></td>
      <td>250000원</td>
      <td><input type="checkbox" name="option1" id="option1" value="250000"
onclick="car();"/></td>
    </tr>
```

JS

```
      <tr>
         <td><label for="option2">후방 카메라</label></td>
         <td>200000원</td>
         <td><input type="checkbox" name="option2" id="option2" value="200000"
onclick="car();"/></td>
      </tr>
      <tr>
         <td><label for="option3">20인치 알루미늄 휠</label></td>
         <td>400000원</td>
         <td><input type="checkbox" name="option3" id="option3" value="400000"
onclick="car();"/></td>
      </tr>
   </tbody>
</table>
<script>
function car(){
   var car_price=Number(document.getElementById("total").defaultValue);
   for(i=1; i<=3; i++){
      var chkOption=document.getElementById("option"+i);
      if(chkOption.checked) car_price+=Number(chkOption.value);
   }
   document.getElementById("total").value=car_price;
}
</script>
</body>
```

문서 객체 (DOM : Document Object Model)

JS

- **문서 객체 조작에 관련한 메서드** : 문서에 새로운 요소를 직접 **생성, 추가, 삭제, 복제**

구분	메서드	상세설명
생성	createElement("요소명")	새 요소의 생성
	appendTextNode("요소명") createTextNode("요소명")	새 텍스트 생성
추가	선택요소.appendChild(새요소)	'선택요소'에 새로운 자식요소를 추가
	선택요소1.insertBefore(새요소, 선택요소2)	'선택요소1'의 자식인 '선택요소2' 앞에 새로운 자식요소를 추가
교체	선택요소1.replaceChild(새요소, 선택요소2)	'선택요소1'의 자식인 '선택요소2'를 새 요소로 교체
삭제	선택요소1.removeChild(선택요소2)	'선택요소1'의 자식인 '선택요소2'를 삭제
복제	선택요소.cloneNode(true or false)	'선택요소'를 복제하여 true인 경우에는 하위요소까지 모두 복제

문서 객체 (DOM : Document Object Model)

JS

요소를 생성하고 그 자식에 텍스트를 넣어 출력하기

- **createElement("요소명") :** 새 요소 생성
- **createTextNode("요소명") :** 새로운 텍스트 생성
- **선택요소.appendChild(새요소) :** '선택요소'에 새로운 자식요소를 추가

```
<h2>새요소 생성하기</h2>
<script>
        var $new_el = document.createElement("span");
        var $new_txt = document.createTextNode("새로운 텍스트");

        document.body.appendChild($new_el);
        $new_el.appendChild($new_txt);
</script>
```

문서 객체 (DOM : Document Object Model)

JS

요소를 복제하여 특정 대상 내부에 넣어 출력하기

- **선택요소.cloneNode(true)** : 선택요소를 복제
- **선택요소.appendChild(새요소)** : '선택요소'에 새로운 자식요소를 추가

```
<ul id="f_menu">
        <li>메뉴-01</li>
        <li>메뉴-02</li>
        <li>메뉴-03</li>
        <li>메뉴-04</li>
</ul>
<ul id="l_menu">
        <li>메뉴-05</li>
        <li>메뉴-06</li>
        <li>메뉴-07</li>
        <li>메뉴-08</li>
</ul>
<script>
        var $itm = document.getElementById("f_menu").children[3];
        console.log($itm);
        var $clone = $itm.cloneNode(true);
        document.getElementById("l_menu").appendChild($clone);
</script>
```

문서 객체 (DOM : Document Object Model)

JS

선택요소의 자식을 삭제하여 출력하기

- **선택요소.removeChild(자식요소) :** 선택요소의 자식요소를 삭제

```
<ul id=“r_menu">
            <li>메뉴-01</li>
            <li>메뉴-02</li>
            <li>메뉴-03</li>
            <li>메뉴-04</li>
</ul>
<script>
            var $itm_remove = document.getElementById(“r_menu");
            $itm_remove.removeChild($itm_remove.children[1]);
</script>
```

문서 객체 (DOM : Document Object Model)

JS

- **createElement("요소명") :** 새 요소 생성

구분	메서드	상세설명
생성	createElement("요소명")	새 요소의 생성
	appendTextNode("요소명") createTextNode("요소명")	새 텍스트 생성
추가	선택요소.appendChild(새요소)	'선택요소'에 새로운 자식요소를 추가
	선택요소1.insertBefore(새요소, 선택요소2)	'선택요소1'의 자식인 '선택요소2' 앞에 새로운 자식요소를 추가
교체	선택요소1.replaceChild(새요소, 선택요소2)	'선택요소1'의 자식인 '선택요소2'를 새 요소로 교체
삭제	선택요소1.removeChild(선택요소2)	'선택요소1'의 자식인 '선택요소2'를 삭제
복제	선택요소.cloneNode(true or false)	'선택요소'를 복제하여 true인 경우에는 하위요소까지 모두 복제

문서 객체 (DOM : Document Object Model)

JS

- **createElement("요소명") :** 새 요소 생성

구분	메서드	상세설명
생성	createElement("요소명")	새 요소의 생성
	appendTextNode("요소명") createTextNode("요소명")	새 텍스트 생성
추가	선택요소.appendChild(새요소)	'선택요소'에 새로운 자식요소를 추가
	선택요소1.insertBefore(새요소, 선택요소2)	'선택요소1'의 자식인 '선택요소2' 앞에 새로운 자식요소를 추가
교체	선택요소1.replaceChild(새요소, 선택요소2)	'선택요소1'의 자식인 '선택요소2'를 새 요소로 교체
삭제	선택요소1.removeChild(선택요소2)	'선택요소1'의 자식인 '선택요소2'를 삭제
복제	선택요소.cloneNode(true or false)	'선택요소'를 복제하여 true인 경우에는 하위요소까지 모두 복제

함수

JS

- 변수는 데이터 저장만 가능하다면 반대로 함수는 실행문을 메모리를 저장했다가 필요시마다 데이터를 실행할 수 있음

- 예를 들면, TV리모콘을 사용하여 각 버튼을 눌렀을 때 채널 또는 음량 등의 변경되는 현상
- 여기서 TV 리모콘의 버튼은 함수가 되며, 버튼을 누르는 것은 함수에 호출을 하는 행동

함수 - 함수의 정의

JS

- 함수는 작업을 수행하거나 값을 계산하는 문장 집합 같은 자바스크립트 절차
- 함수선언(기본형식)

```
function 함수명(){
  실행문;
}
```

```
참조변수=function( ) {
  실행문;
}
```

- "이 구역부터는 함수 정의문"이라고 선언

  •위에서 function은 함수에 대한 정의문으로서 { } 중괄호 에 입력하는 실행문을 지금 당장 실행하지 말고 함수명으로 저장하라는 의미

- 함수의 호출(기본형식)

```
함수명( ); 또는 참조변수( );
```

함수 - 함수의 정의

JS

- 함수의 사용 예시

```
<script>
  function greet( ){
    alert(“안녕히 가세요”);
  }
</script>
<button onclick=“greet( );”>탈퇴</buttton>
```

- 위에서 ‘greet’이라는 함수명을 사용자가 ‘탈퇴’라는 버튼을 눌렀을 때, 기존에 함수선언에서 저장한 “안녕히 가세요”라는 경고창으로 반환하는 방식

함수 - 함수의 정의

JS

- 함수의 사용 예시

```
<script type="text/javascript">
//<![CDATA[
var color=[“skyblue","yellow","aqua","purple"];
var i=0;
function colorBg(){
   i++;
   if(i>=color.length) i=0;
   var bodyTag=document.getElementsByTagName("body")[0];
   bodyTag.style.backgroundColor=color[i];
}
//]]>
</script>
<body>
  <button onclick="colorBg();">배경색 바꾸기</button>
</body>
```

함수 - 매개 변수가 존재하는 함수

JS

- 함수 명만으로 일방적인 호출이 아닌 입력 값 등의 전달된 값을 받아서 호출시 사용

✓ 기본형식

```
function 함수명 (매개변수1, 매개변수2, ..., 매개변수3) {
   스크립트 실행문;
}
함수명 (데이터1, 데이터2, ...,데이터3);
```

✓ 적용예시

```
<script type="text/javascript">
//<![CDATA[
function myFnc(name, area){
   document.write("만나서 반갑습니다. 저는 "+name+" 입니다.","<br />");
   document.write("저의 직장은 "+area+"입니다.","<br /><br />");
}
myFnc("정지우", "서울");
myFnc("우승민", "춘천");
//]]>
</script>
```

함수 - 매개 변수가 존재하는 함수

JS

✓ 실습

```
<script type="text/javascript">
//<![CDATA[

var user_id=prompt("아이디를 입력하세요.","");
var user_pw=prompt("패스워드를 입력하세요.","");

function login(id, pw){
   if(id=="본인의 아이디를 넣으세요"){
        if(pw=="본인의 패스워드를 넣으세요"){
          document.write(id+"님 방문을 환영합니다");
        }else{
          alert("잘못된 비밀번호입니다.")
        }
   }else{
      alert("존재하지 않는 아이디입니다.")
   }
}
login(user_id,user_pw);
//]]>
</script>
```

함수 - 내장함수

JS

- 자바스크립트 엔진에 내장된 함수 정의문
  - ✓ 내장함수 : 함수 정의문 선언 없이도 실행문의 실행이 가능

종류	설명	사용
parseInt()	문자형 데이터를 정수형 데이터로 변경	parseInt("8.123") → 8
parseFloat()	문자형 데이터를 실수형 데이터로 변경	parseFloat("8.123") → 8.123
String()	문자형 데이터로 변경	String(8) → "8"
Number()	숫자형 데이터로 변경	Number("8") → 8
Boolean()	논리형 데이터로 변경	Boolean("8.123") → true
isNaN()	데이터에 숫자가 아닌 문자를 포함하면 true를 반환	isNaN("8-3") → true isNaN("83") → false
eval()	문자형 데이터를 큰따옴표가 없는 스크립트 코드 처리	eval("8+12") → 20

함수 – return 문

JS

- return 문은 함수에서 결과값을 되돌려줄 때 사용

✓ 기본형식

```
function 함수명 (){
   스크립트 실행문;
   return 데이터(값);
}
var 변수 = 함수명 ();
```

✓ 적용예시

```
function calc(){
   var result=100+200;
   return result;
}
var num = calc();
document.write(num);
```

함수 – return 문

JS

- return 문은 함수에서 결과값을 되돌려줄 때 사용
  - ✓ 실습

```
<script type="text/javascript">
//<![CDATA[
function myFnc(a, b){  //2번, 5번
    var num=a*b;  //3번
    return num;  //4번
}
var result=myFnc(10,3);  //1번, 6번
document.write(result); //7번
//]]>
</script>
```

함수 – return 문

JS

- return 문이 실행되면 반복문에서 break 문처럼 실행문을 강제 종료

✓ 기본형식

```
function 함수명 (){
   스크립트 실행문;
   return;
   스크립트 실행문;
}
함수명 ();
```

✓ 적용예시

```
function testing(){
   document.write("저의 이름은 정찬우 입니다.");
   return;
   document.write("저의 나이는 27살 입니다."); //실행되지 않음
}
testing();
```

함수 – return 문

JS

- return 문이 실행되면 반복문에서 break 문처럼 실행문을 강제 종료

  ✓ 실습

```
<script type="text/javascript">
//<![CDATA[
   var myName=prompt("당신의 이름은 무엇입니까?", "");
   var myAge=prompt("당신의 나이는?", "");

   function myFunc(){
    document.write(myName);
    return;
    document.write(myAge);  //실행되지 않음
   }
   myFunc();
//]]>
</script>
```

함수 – 재귀함수

JS

- 재귀함수 란 : 최초 함수를 호출하면 정의문 내에서 실행문을 다시 호출하는 것
- 함수를 반복문처럼 여러 번 사용하기 위해서 사용

✓ 기본형식

```
function myFunc(){
   스크립트 실행문;
   myFunc();
}
myFunc();
```

함수 – 재귀함수

JS

- 재귀함수 란 : 최초 함수를 호출하면 정의문 내에서 실행문을 다시 호출하는 것
- 함수를 반복문처럼 여러 번 사용하기 위해서 사용

✓ 실습

```
<script type="text/javascript">
//<![CDATA[
var num=0;
function testFnc(){
    num++;
    document.write(num, "<br />");
    if(num==10) return;

    //재귀함수 호출
    testFnc();
}

testFnc();
//]]>
</script>
```

함수 – 지역변수, 전역변수

JS

- **전역변수 : 함수 정의문 내에서 var를 붙이지 않은 변수. 자바스크립트 내부라면 어디든 사용 가능**
- **지역변수 : 함수 정의문 내에서 var를 붙인 변수. 이 변수로 저장되는 데이터는 함수 정의문 내부에서만 사용 가능**

✓ 기본형식 - 전역변수

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

✓ 기본형식 - 지역변수

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

- 지역 변수와 전역변수로 나누는 이유 : 변수의 중복을 피하기 위함

함수 – 지역변수, 전역변수

JS

✓ 실습

```
<script type="text/javascript">
//<![CDATA[
        var num=200; //전역 변수
        function myFnc(){
                var num=500; //지역 변수
                document.write(num); //지역 변수 num의 500이 출력
        }
        myFnc(); //함수를 호출
        document.write(num); //전역 변수 num의 200이 출력
//]]>
</script>
```

이벤트(Event)

JS

<<개요>>

- 이벤트란
  - ✓ 웹브라우저나 사용자가 행하는 동작을 말함
  - ✓ 예를 들어 웹 문서에서 키보드를 누르는 것, 웹 페이지를 불러오는 것도 이벤트로 말함
  - ✓ 프로그램에 의해 감지되고 처리될 수 있는 동작이나 사건
  - ✓ 키보드나 마우스 동작처럼 브라우저에서 발생하는 사건(순간)들
  - ✓ 사용자가 발생시킬 수도 있고 응용 프로그램이 발생시킬 수도 있음

    → 사용자에 의해 발생하는 이벤트를 적절하게 처리
- 예를 들어 사용자가 버튼을 클릭했을 때, 웹페이지 또는 내부에 어떠한 반응이 나타나는 것을 일컬음. 상호간의 인터랙티브를 구성하기 위한 과정

이벤트(Event)

JS

- **마우스 이벤트** : 마우스에서 버튼이나 휠 버튼을 조작할 때 발생하는 이벤트

종류	사용
click	요소에 마우스를 클릭했을 때 이벤트 발생
dbclick	요소에 마우스를 더블 클릭했을 때 이벤트 발생
mouseover	요소에 마우스를 오버했을 때 이벤트 발생
mouseout	요소에 마우스를 아웃했을 때 이벤트 발생
mousedown	요소에 마우스를 눌었을 때 이벤트 발생
mouseup	요소에 마우스를 떼었을 때 이벤트 발생
mousemove	요소에서 마우스를 움직였을 때 이벤트 발생
contextmenu	Context menu(마우스 우클릭버튼을 눌렀을 때 나오는 메뉴)가 나오기 전에 이벤트 발생

이벤트(Event)

JS

- **키보드 이벤트** : 키보드의 특정 키를 조작할 때 발생하는 이벤트

종류	사용
keypress	지정한 요소에서 키보드를 누른 상태에서 이벤트 발생
keydown	지정한 요소에서 키보드를 눌렀을 때 이벤트 발생
keyup	지정한 요소에서 키보드를 눌렀다 떼었을 때 이벤트 발생

이벤트(Event)

JS

- **폼 이벤트** : 로그인, 검색/게시판/ 설문조사 처럼 사용자가 자료를 입력하는 모든 요소에서 이벤트 발생

종류	사용
focus	요소에 포커스가 이동되었을 때 이벤트 발생
blur	요소에 포커스가 벗어났을 때 이벤트 발생
change	요소에 값이 변경되었을 때 이벤트 발생
submit	submit 버튼을 눌렀을 때 이벤트 발생
reset	reset 버튼을 눌렀을 때 이벤트가 발생
select	input이나 textarea 요소 안의 텍스트를 드래그 하여 선택하였을 때 이벤트 발생

이벤트(Event)

JS

- **로드 및 기타 이벤트** : 서버에서 웹 문서를 가져오거나 문서를 상하로 스크롤 하는 등 웹 문서를 브라우저 창에 보여주는 것과 관련된 이벤트

종류	사용
load	페이지의 로딩이 완료되었을 때 이벤트 발생
abort	이미지의 로딩이 중단되었을 때 이벤트 발생
error	문서가 정확히 로딩되지 않았을 때 이벤트 발생
unload	페이지가 다른 곳으로 이동되었을 때 이벤트 발생
resize	요소에 사이즈가 변경되었을 때 이벤트 발생
scroll	스크롤 바를 움직였을 때 이벤트 발생

이벤트(Event)

JS

<<용어정리>>

- 이벤트 : 키보드나 마우스 동작처럼 브라우저에서 발생하는 사건(순간)들
- 이벤트 이름 : 이벤트 타입이라고도 하며, 이벤트 종류를 의미
- 이벤트 속성 : 이벤트 이름에 on이 붙은 형태, 문서 객체의 속성
- 이벤트 리스너 : 이벤트 발생시 실행할 코드를 나열할 함수
- 이벤트 연결 : 이벤트 속성에 이벤트 리스너를 지정
- 이벤트 모델 : 문서객체에 이벤트를 연결하는 방법
- 이벤트 객체 : 브라우저에서 발생한 이벤트에 관한 정보를 가지고 있는 객체

```
<script type="text/javascript">
   window.onload=function( ){
      document.write('놀러 가기 좋은 날');
   };
</script>
```

✓ 이벤트 이름 : load

✓ 이벤트 속성 : onload

✓ 이벤트 리스너 : 이벤트 핸들러

✓ 이벤트 속성 : 문서객체에 이벤트를 연결하는 방법. function 이하의 문서

이벤트(Event)

JS

- 이벤트 실습 – 클릭 이벤트 및 마우스오버 이벤트

```
<p><button onclick="colorBg();">클릭 버튼1</button></p>
<p><button id="btn2">클릭 버튼2</button></p>
<p><button onmouseover="colorBg();">마우스오버 버튼1</button></p>
<p><button id="btn4">마우스오버 버튼2</button></p>

<script type="text/javascript">
var color = ["white", "yellow", "aqua", "purple"];
var i = 0;
function colorBg(){  //onclick 또는 onmouseover 라는 이벤트가 걸려있는 부분 적용
    i++; //인덱스번호를 하나씩 늘려가겠다.
    if(i>=color.length){   //color.length:변수를 몇개를 가지고 있는지...4개 / i는 0부터 시작(=인덱스 번호)
        i=0;
    }
    var bodyColor = document.getElementsByTagName("body")[0];
    //getElementsByTagName이 배열로 저장되기 때문에 배열번호를 가져와야 됩니다.
    bodyColor.style.backgroundColor = color[i]; //color[0] : 색상 white, color[1] : 색상 yellow, ....
}
window.onload = function(){  //DOM 객체를 이용, 이벤트를 등록
    var $btn2 = document.getElementById("btn2");
    $btn2.onclick = function(){colorBg();}  //두번째 버튼을 클릭했을 때, colorBg();라는 함수를 호출
    var $btn4 = document.getElementById("btn4");
    $btn4.onmouseover = function(){colorBg();}  //네번째 버튼을 마우스오버 했을 때, colorBg();라는 함
수를 호출
}</script>
```

이벤트(Event)

JS

- 이벤트 중복 처리시 사용하는 addEventListener 또는 attachEvent

```
<button id="btn" onclick="alert('실행문 1');" onclick="alert('실행문 2');">버튼</button>
```

✓ 위와 같은 경우, 마지막 onclick 이벤트만 실행되기 때문에, 정확한 실행효과를 거둘 수 없음

✓ 이러한 경우 addEventListener 또는 attachEvent 를 사용하여야 한다.

브라우저 구분	방법
IE 8 이하 외 브라우저 (크롬, 사파리, 파이어폭스, 오페라)	요소 선택.addEventListenert("이벤트 종류", 함수명 또는 익명 함수, false(표준 캡처방식));
IE 8 이하	요소 선택.attachEvent ("**on**이벤트 종류", 함수명 또는 익명 함수);

이벤트(Event)

JS

- 이벤트 실습 – 이벤트 중복 처리시 사용하는 addEventListener 또는 attachEvent

```
<button id="btn_add">버튼-이벤트</button>

<script type="text/javascript">
window.onload = function(){
        function fnc1(){
                alert("자바스크립트 이벤트 1");
        }
        function fnc2(){
                alert("자바스크립트 이벤트 2");
        }
        var $btn_add = document.getElementById("btn_add");
        if(window.addEventListener){  //크롬, 파이어폭스 등의 브라우저 방식 addEventListener
                $btn_add.addEventListener("click", fnc1, false);
                $btn_add.addEventListener("click", fnc2, false);
        }else{  //IE8이하에서 실행되게 attachEvent
                $btn_add.attachEvent("onclick", fnc1);
                $btn_add.attachEvent("onclick", fnc2);
        }
}
</script>
```

이벤트(Event)

JS

- 이벤트 실습 – 이미지 태그 추가에서 내부 속성(src) 변경하는 클릭 이벤트

```
<script type="text/javascript">
window.onload=function(){
        var image=document.createElement('img');  // createElement : img 엘리먼트를 생성
        image.src= ' img/image1.jpg';
        image.width=300;
        image.height=300;

        document.body.appendChild(image);
        //appendChild : body를 기준으로 하위 자식요소에 맨 마지막 부분에 넣겠다는 의미
        function changeImage(){
                image.setAttribute('src', 'img/image2.jpg');
        }
        var $c_btn=document.getElementById('c_btn');
        $c_btn.onclick = function(){changeImage()};
        //DOM 방식으로 인한 태그 내부에 onclick이 적용 안되어 스크립트 문에서 직접 연결
};
</script>
<input type='button' id='c_btn' value=" 이미지 변경" />
```

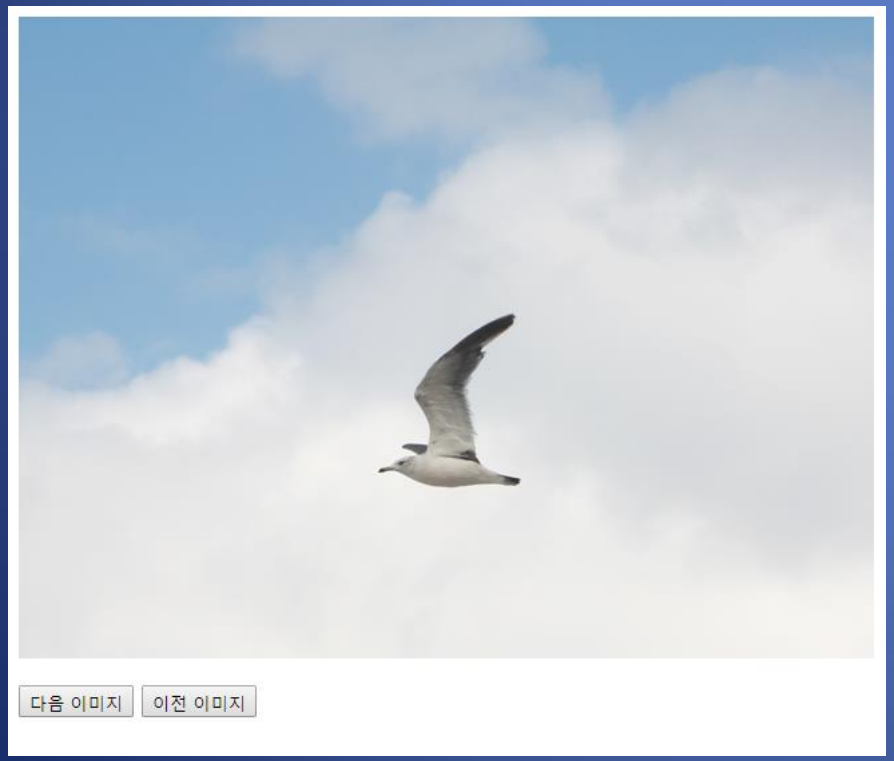
- ✓ 이벤트 이름 : click
- ✓ 이벤트 속성 : onclick
- ✓ 이벤트 리스너 : changeImage() 함수

종합 실습

JS

- onclick 이벤트를 통한 이미지 변경하기 - ① HTML

```
<body>
        <img id="slide_img" src="img/pic_1.jpg"/>
        <button id="next_btn">다음 이미지</button>
        <button id="before_btn">이전 이미지</button>
</body>
```



종합 실습

JS

- onclick 이벤트를 통한 이미지 변경하기 - ② Javascript

```
var $images = ["img/pic_1.jpg", "img/pic_2.jpg", "img/pic_3.jpg", "img/pic_4.jpg",
"img/pic_5.jpg"];
var i = 0;  //배열의 초기값을 설정
function slideImg_next(){  //"다음 이미지" 버튼을 클릭했을 때, 호출되는 함수문
        i++;  //배열의 인덱스 값을 증가
        if(i>=$images.length){  //배열이 마지막 인덱스를 넘을 경우
                i=0; //배열에 저장된 첫번째 이미지로 지정
        }
        var $slider = document.getElementById("slide_img");
        $slider.setAttribute("src", $images[i]);
}
function slideImg_before(){  //"이전 이미지" 버튼을 클릭했을 때, 호출되는 함수문
        i--; //배열의 인덱스 값을 감소
        if(i<0){  //배열이 첫번째 인덱스 아래로 떨어졌을 경우
                i=$images.length-1;  //배열에 저장된 마지막 이미지로 지정
        }
        var $slider = document.getElementById("slide_img");
        $slider.setAttribute("src", $images[i]);
}
window.onload = function(){
        var $btn_next = document.getElementById("next_btn");
        $btn_next.onclick = function(){slideImg_next();}
        var $btn_before = document.getElementById("before_btn");
        $btn_before.onclick = function(){slideImg_before();}
}
```