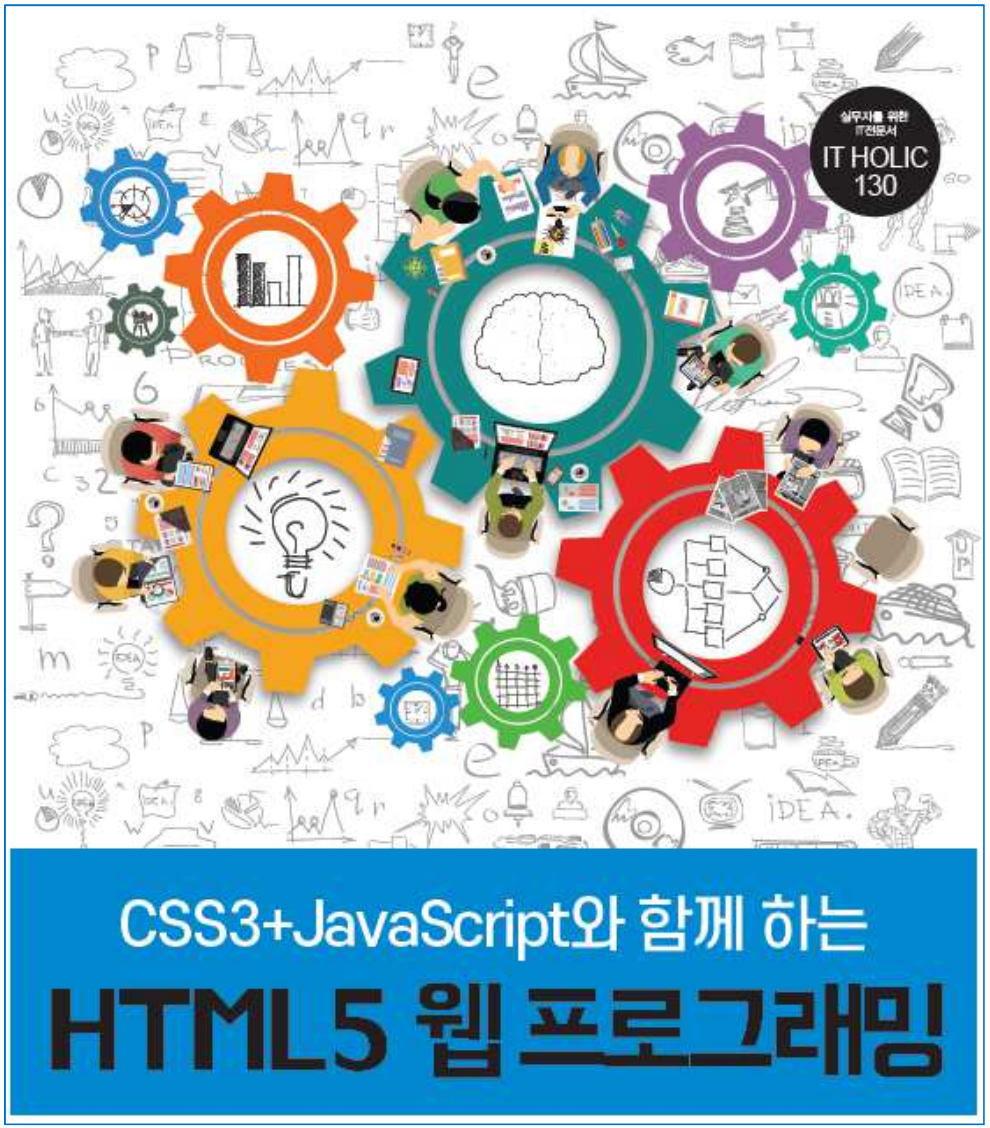


CSS3+JavaScript와 함께 하는

HTML5 웹 프로그래밍

7장 자바스크립트 객체와 브라우저 객체 모델

CHAPTER

07

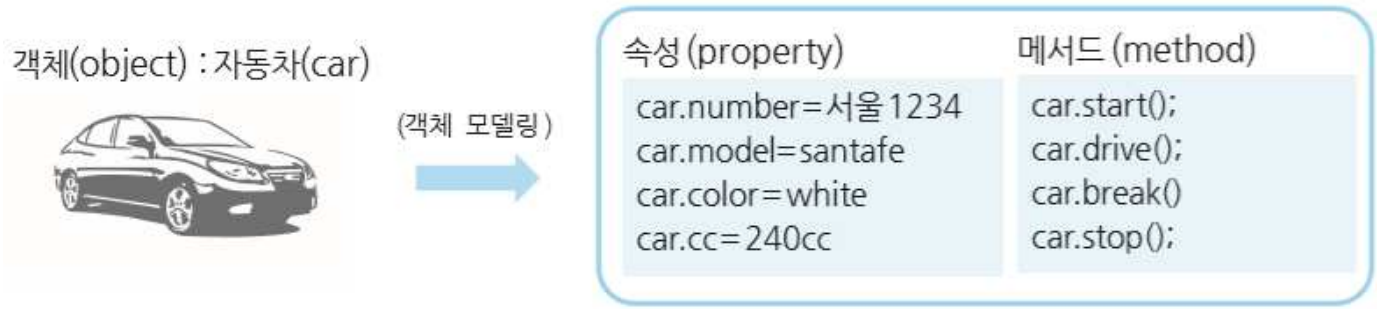
HTML5+CSS3+JavaScript

자바스크립트 객체와 브라우저 객체 모델

7.1 자바스크립트 객체

7.1.1 객체의 개념

- 컴퓨터 시스템에서 하나의 처리 대상으로 파악되는 모든 개념이나 실체 등
- 자바스크립트 객체형 :
  - ✓ 하나 이상의 특성들을 그룹핑해서 나타내는 것
  - ✓ 객체들의 특성은 여러 속성(property)들과 메서드(method)들로 모델링해서 표현함
  - ✓ 객체 속성 : 객체의 정적인 한 특성을 나타내는 것 예) 이름, 나이, 색상 등
    - 값으로 표현하고 변수를 이용해서 저장함
  - ✓ 객체 메서드 : 특정한 행동이나 처리 방법 등과 같은 동적인 특성을 나타내는 것
    - 함수를 이용해서 나타냄



[그림 7.1] 객체 모델링의 예(자동차)

객체(object) : 자동차(car)

(객체 모델링)

속성 (property)	메서드 (method)
car.number=서울1234	car.start();
car.model=santafe	car.drive();
car.color=white	car.break()
car.cc=240cc	car.stop();

- 자바스크립트의 객체 모델링 표현 방법

[예 1] 객체 상수 모델링

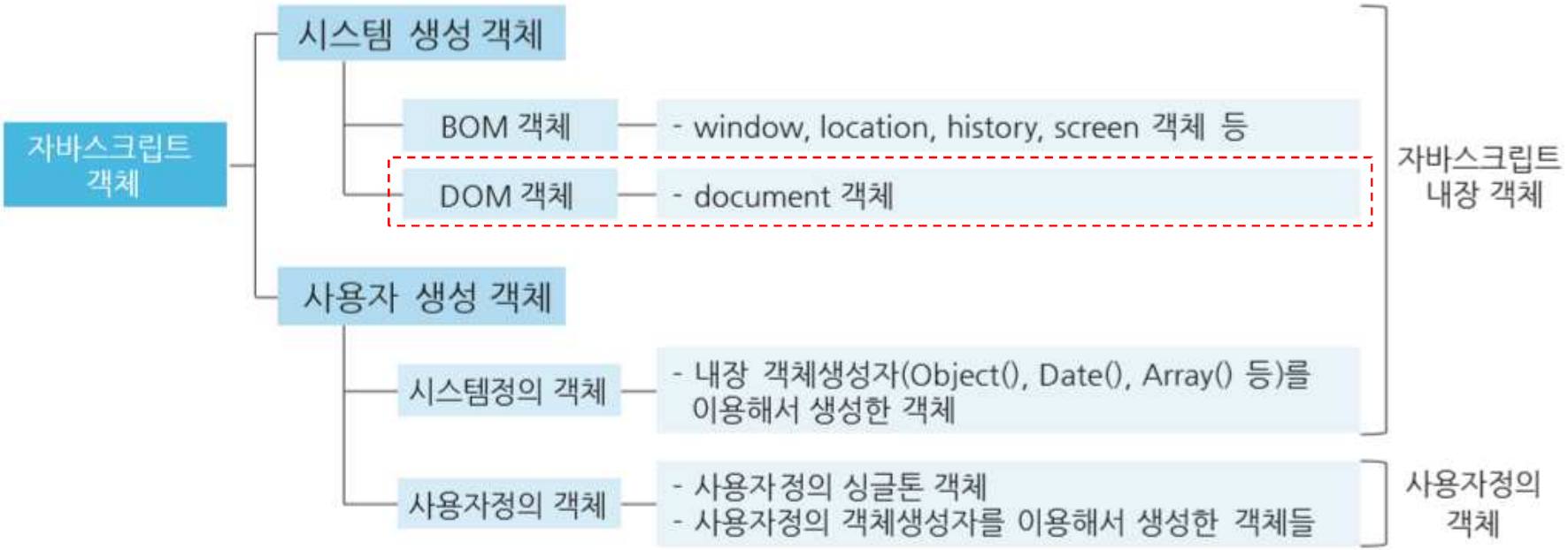
```
{ // 객체속성정의
    number : 1234,
    model : K5,
    color : black,
    cc : 2400,

// 객체메서드정의
    start : function(){ ... },
    drive : function(){ ... },
    break : function(){ ... },
    stop: f unction(){ ... }
}
```

[예 2] 객체생성자 함수 모델링

```
function Car(number, model, color, cc) {
    // 객체속성정의
    this.number = number;
    this.model = model;
    this.color = color;
    this.cc = cc;

    //객체메서드정의
    this.start = function(){ ... };
    this.drive = function(){ ... };
    this.break = function(){ ... };
    this.stop = function(){ ... };
}
```

7.1.2 객체의 생성과 사용

- 시스템 생성 객체 : 자바스크립트를 사용할 때 이미 존재하는 객체들
- 사용자 생성 객체 : 사용자들이 자바스크립트 코드로 생성하는 객체들



- 자바스크립트에서 사용자들이 객체를 생성하는 방법

1) 싱글톤 객체 생성

❶ 객체상수(객체 리터럴) 표현
```
var 객체변수명 = 객체상수;   // 싱글톤객체 생성
```

❷ 객체생성자 함수 Object() 이용
```
var 객체변수명 = new Object();  // 빈 객체 생성
```

2) 인스턴스 객체 생성

❶ 내장 객체생성자 함수 이용 :
```
var 객체변수명 = new 내장_객체생성자();
```

❷ 사용자 정의 객체생성자 함수이용 :
```
var 객체변수명 = new 사용자정의_객체생성자();
```

(1) 싱글톤 객체 생성

- 싱글톤 객체 : 유일하게 모델링되는 객체
- 싱글톤 객체 생성 방법

  ❶ 객체 상수로 직접 표현
  ❷ 객체생성자 함수 Object()를 이용해서 빈 객체 생성후, 객체 속성과 메서드들을 추가함

❶
```
var person = {
        name : "조우진",
        nickName : "X-man",
        major : "건축학",
        age : 24
};
```

❷
```
var  person = new Object();
     .....
person.name = "조우진";
person.nickName = "X-man";
person.age = 24;
person.major = "건축학";
```

- 메서드들이 필요없는 객체들을 나타낼 때에 한해 제한적으로 사용함

(2) 인스턴스 객체 생성

- 인스턴스 객체 : 객체생성자를 이용해서 생성한 객체
- 객체생성자 함수 : 같은 유형의 객체를 생성하는 함수, 특정한 객체 유형(object type)을 나타냄
  - ❶ 내장 객체 생성자 : Object(), Number(), Sting(), Array(), Date() 등
  - ❷ 사용자 정의 객체생성자
    - ✓ 일반 함수와 유사하지만, 예약어 this를 이용해서 속성과 메서드들을 나타냄
    - ✓ this : 현재 코드를 실행하는 객체를 나타내는 예약어
    - ✓ 객체생성자에서 this는 생성되는 객체 자신을 나타냄

❶ 내장 객체의 생성 예

```
var x1 = new Object();  //빈객체생성
var x2 = new String();  //문자열객체생성
var x3 = new Number();  //수치객체생성
var x4 = new Boolean();  //논리객체생성
var x5 = new Array();  //배열객체생성
var x6 = new Date();  //Date객체생성
```

❷ 사용자 정의 객체 생성 예

```
// 사용자 정의 객체생성자 함수 정의
function person(name, year, score1, score2){
        this.name = name;
        this.year = year;
        this.score1 = score1;
        this.score2 = score2;
}
var  guy = new person("조우진", 3, 80, 90);  // 객체 생성
var  lady = new person("신은수", 4, 92, 88);  // 객체 생성
```

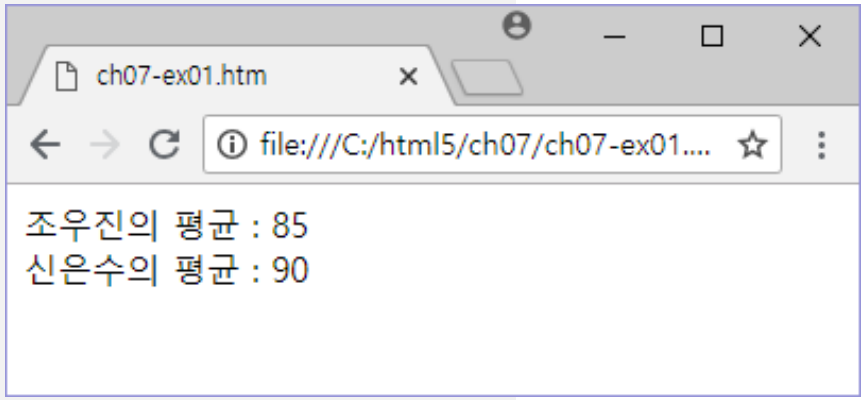
- 객체생성자를 이용해서 객체를 생성할 때는 반드시 예약어 new를 사용해야 함

(3) 객체의 참조

- 객체의 속성과 메서드들은 개별적으로 접근해서 참조할 수 있음
  - ✓ 객체 속성 참조 : **객체명.속성명** 또는 **객체명["속성명"]**
  - ✓ 객체 메서드 참조 : **객체명.메서드명()**

[예 7.1] 객체의 속성 참조 및 메서드 호출

```
<script>
   function person(name,year,score1,score2){
          this.name=name;
          this.year=year;
          this.score1=score1;
          this.score2=score2;
          this.average=function(){
                return(this.score1+this.score2)/2;
          }
   }

   var  guy = new  person("조우진",3,80,90);
   var  lady = new  person("신은수",4,92,88);
   document.write(guy.name+'의평균:'+guy.average()+'<br>');
   document.write(lady.name+'의평균:'+lady.average());
</script>
```



[참고] 자바스크립트의 객체 생성자 함수와 C/C++(또는 java)의 클래스(class) 개념 비교

- 자바스크립트의 객체생성자를 이용해서 생성된 각 객체는 속성값들과 메서드들을 저장하는 메모리들을 개별적으로 확보함
- C/C++(또는 java)의 클래스로 생성하는 객체들은 속성값들을 저장하는 메모리들은 개별적으로 확보지만 메서드들을 저장하는 메모리들을 공유함
- 객체의 메서드들은 함수 코드이므로, 객체들이 개별적으로 저장하는 것은 비효율적임
- 자바스크립트에서는 클래스와 유사한 개념으로 **프로토타입(prototype)** 개념 지원
- 자바스크립트는 생성된 객체를 소멸시키는 방법은 제공하지 않고 대신 사용되지 않는 객체들을 자동으로 수거하는 **가비지 컬렉션(garbage collection)** 기능을 제공함

7.1.3 자바스크립트의 내장 객체

- 대표적인 내장 객체들 : Date, Math, Array, String, Number, Object 객체 등

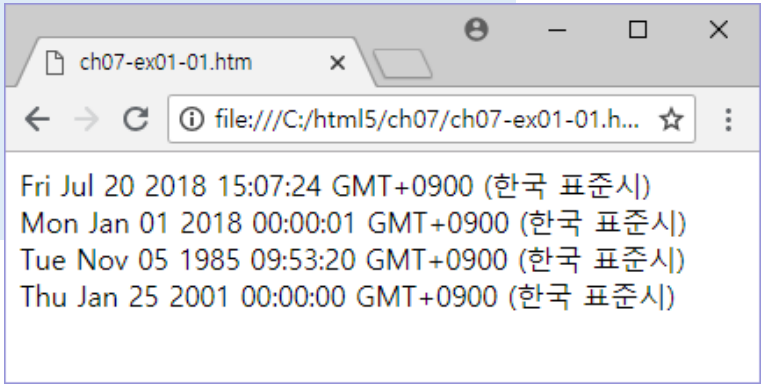
(1) Date 객체

- 년, 월, 일, 요일, 시간 등과 같은 날짜와 시간 정보를 나타내는 객체
- Date 객체의 주요 메서드들

속성 및 메서드	의미 또는 기능
getFullYear(), getMonth(), getDate(), getDay()	년도, 월(0~11), 날짜(1~31), 요일(0~6) 반환
getHours(), getMinutes(), getSeconds(), getMilliseconds()	시간(0~23), 분(0~59), 초(0~59), 밀리초(0~999) 반환
setHours(), setMinutes(), setSeconds(), setMilliseconds()	시간(0~23), 분(0~59), 초(0~59), 밀리초(0~999) 지정
getTime() / setTime()	1970-1-1 자정 이후 경과시간(밀리초) 반환, 지정
to ○○○○ String() 메서드 ○○○○ : Date, LocalDate, LocalTime, Local, Time, GMT 등	Date 객체를 다양한 형식으로 출력함 (Date: 표준시간, Local: 지역시간 등)

- Date 객체의 생성 예

```
var d1 = new Date();  // ❶ 현재 날짜 및 시간 표시
var d2 = new Date(2018, 0, 1, 0, 0, 1);  // ❷ 2018년 1월 1일 0시 0분 1초 표시
var d3 = new Date(500000000000);  // ❸ 1970년 1월 1일 이후의 경과시간(밀리초) 표시
var d4 = new Date(2000, 12, 25);  // ❹ 지정된 시간(dateString) 표시

document.write(d1+"<br>");
document.write(d2+"<br>");
document.write(d3+"<br>");
document.write(d4+"<br>");
```



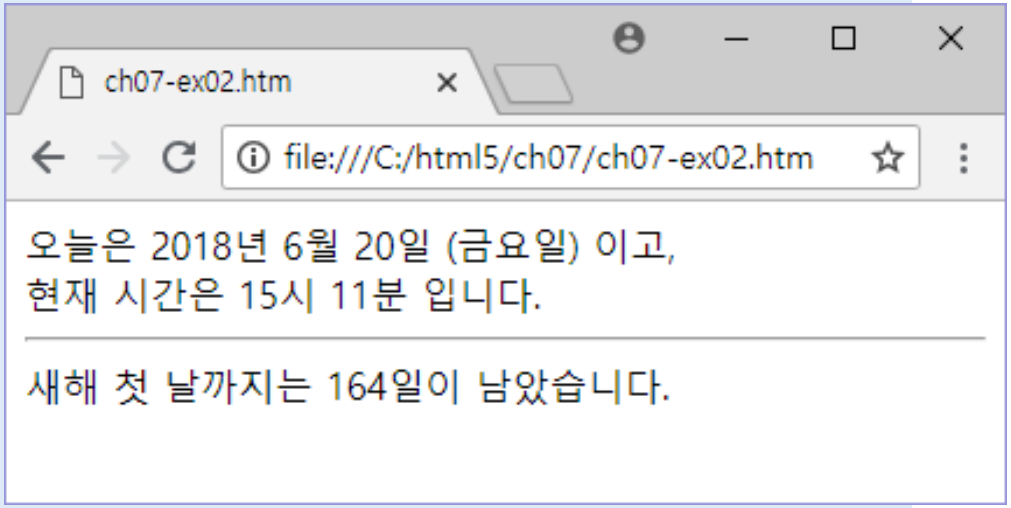
- getMonth() : Data 객체의 월(month, 1월~12월) 정보를 0~11 사이의 정수로 반환함
- getDate() : 날짜(date, 매월 1~31) 정보를 1~31 사이의 정수로 반환함
- getDay() : 요일(일~토) 정보를 0~6 사이의 정수로 반환함
- getTime() : Date 객체의 시간을 1970년 1월 1일 자정 이후의 경과시간(밀리초)으로 환산해서 반환함

[date1, date2 사이의 시간 차이 계산방법]

```
var difference = date1.getTime() - date2.getTime();   // date1이 date2 보다 최근시간으로 가정함
differenceDate=difference/1000*60*60*24;  // date1, date2 사이의 일수 차이
```

[예 7.2] Data 객체 이용 예

```
<script>
    var cYear = new Date();
    nYear = new Date(2019,0,1);
    year=cYear.getFullYear();
    month=cYear.getMonth();
    date=cYear.getDate();
    day=cYear.getDay();
    hour=cYear.getHours();
    minute=cYear.getMinutes();

    switch(day){
        case0: days="일";  break;
        case1: days="월";  break;
        case2: days="화";  break;
        case3: days="수";  break;
        case4: days="목";  break;
        case5: days="금";  break;
        case6: days="토";
    }
    leftedTime = nYear.getTime() - cYear.getTime();
    leftedDays = Math.floor( leftedTime / (24*60*60*1000) );
    document.write("오늘은 " + year + "년 " + month + "월 " + date + "일("+days+"요일)이고,<br>");
    document.write("현재시간은 " + hour + "시 " + minute + "분입니다.<hr>");
    document.write("새해첫날까지는 " + leftedDays + "일이남았습니다.<br>");
</script>
```



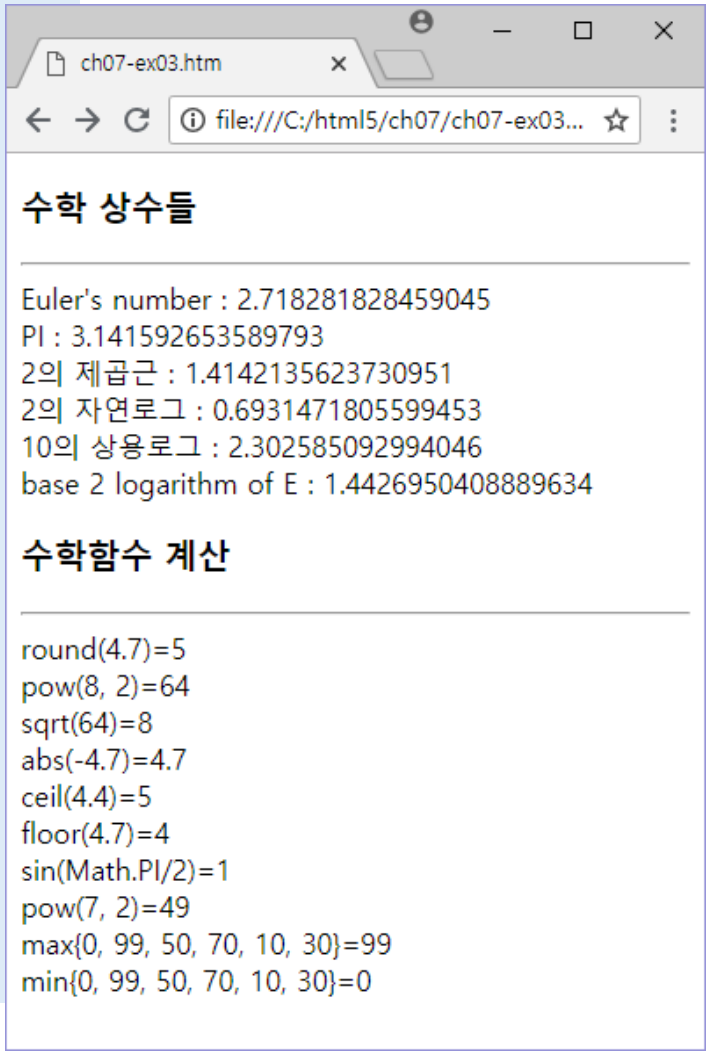
(2) Math 객체

- 수학의 상수와 함수들을 나타내는 객체
- Math는 그 자체가 객체임(Math() 같은 객체생성자 함수를 제공하지 않음)
- 즉, 직접 Math 객체의 속성과 메서드들을 참조해서 사용함

속성 및 메서드	의미 또는 기능
E, LN2, LN10, PI, SQRT1_2, SQRT2	오일러 상수, 자연로그($\log_2 1$), 자연로그($\log_{10} 1$), 파이(ø), 1/2의 제곱근($\sqrt{1/2}$), 2의 제곱근($\sqrt{2}$)
abs(*x*), sqrt(*x*), exp(*x*), log(*x*),	*x*의 절대값(|*x*|), 제곱근($\sqrt{x}$), 지수 함수값(e^x), 로그 함수값($\log x$)
round(*x*), ceil(*x*), floor(*x*)	x의 반올림, ⌊x⌋, ⌈x⌉를 반환함
sin(*x*), cos(*x*), tan(*x*)	삼각 함수값 sin(x), cos(x), tan(x)를 반환함
pow(*x*,*y*),	x^y 지수승값을 반환함 l
max(*x*,*y*, …,*z*) , min(*x*,*y*, …,*z*)	*max*{*x*,*y*,…,*z*}, *min*{*x*,*y*,…,*z*}을 반환함
random( )	0과 1 사이의 난수값을 반환함

[예 7.3] Math 객체의 수학 상수와 수학 함수들

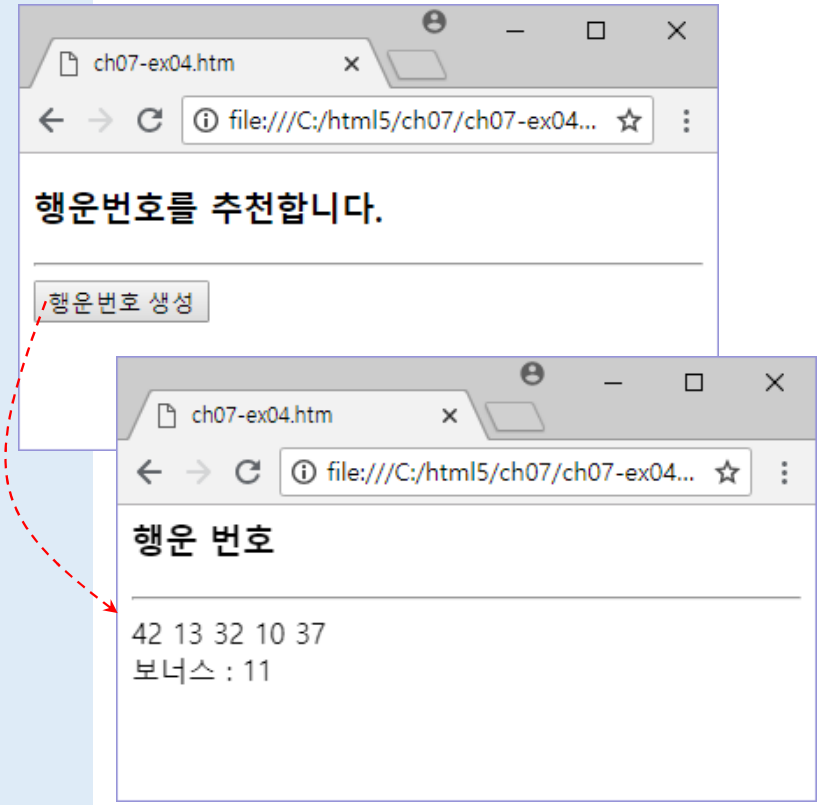
```
<script>
   document.write("<h3>수학상수들</h3><hr>");
   document.write("Euler'snumber: "  + Math.E+"<br>");
   document.write("PI: "  + Math.PI + "<br>");
   document.write("2의제곱근: "  + Math.SQRT2 + "<br>");
   document.write("2의자연로그: "  + Math.LN2 + "<br>");
   document.write("10의상용로그: "  + Math.LN10 + "<br>");
   document.write("base2logarithmofE: "  + Math.LOG2E + "<br>");
   document.write("<h3>수학함수계산</h3><hr>");
   document.write("round(4.7)= "  + Math.round(4.7) + "<br>");
   document.write("pow(8,2)= "  + Math.pow(8,2) + "<br>");
   document.write("sqrt(64)= "  + Math.sqrt(64) + "<br>");
   document.write("abs(-4.7)= "  + Math.abs(-4.7) + "<br>");
   document.write("ceil(4.4)=" + Math.ceil(4.4) + "<br>");
   document.write("floor(4.7)=" + Math.floor(4.7) + "<br>");
   document.write("sin(Math.PI/2)=" + Math.sin(Math.PI/2) + "<br>");
   document.write("pow(7,2)=" + Math.pow(7,2) + "<br>");
   max = Math.max(0, 99, 50, 70, 10, 30);
   min = Math.min(0, 99, 50, 70, 10, 30);
   document.write("max{0,99,50,70,10,30}=" + max + "<br>");
   document.write("min{0,99,50,70,10,30}=" + min + "<br>");
</script>
```



- Math.random() : 0과 1 사이의 임의의 실수를 난수로 생성함
- Math.random()*n : 0~n 사이의 난수 생성
- Math.floor(Math. random()*n)+1 : 1과 n 까지의 난수(정수) 생성

[예 7.4] Math.random()를 이용한 난수 생성

```
<h3>행운번호를추천합니다.</h3><hr>
<buttononclick="luckyNumber();">행운번호생성</button>
<script>
   function luckyNumber(){
      var num = new Array();
      for (i=1;i<=5;i++) {
         num[i]=Math.floor(Math.random()*44)+1;
         for (j=1;j<=i;j++)
            if (num[j]==num[i]) num[i]=Math.floor(Math.random()*44)+1;
      }
      num[6]=Math.floor(Math.random()*44)+1;
      for (i=1;j<=5;i++){
         if  (num[j]==num[i])num[6]=Math.floor(Math.random()*44)+1;
      }
      document.write("<h3>행운번호</h3><hr>");
      for (i=1;i<=5;i++)  document.write(num[i]+"");
      document.write("<br>보너스:"+num[6]);
  }
</script>
```



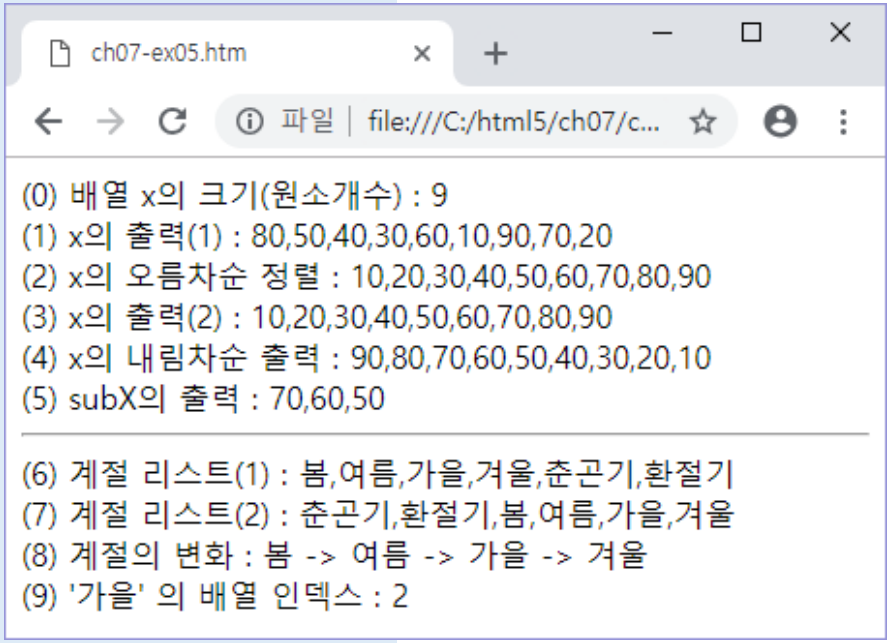
(3) Array 객체

- Array 객체 : 여러 데이터들의 순차적 모임을 나타내는 객체
- 배열 원소들은 정수인덱스 또는 문자열 인덱스를 이용해서 참조 가능함
- 주요 속성과 메서드 : 정수인덱스를 가진 배열 원소들에 대해서만 적용됨

속성 및 메서드	의미 또는 기능
length	배열 크기(즉, 배열원소 개수)
indesOf(x)	값이 x 인 배열원소의 인덱스 반환함
concat(x)	x를 배열 끝에 추가함, x는 배열일 수도 있음
slice(i, j)	배열의 일부분(인덱스 i부터 j까지의 배열원소들)을 복사해서 반환함
join(c)	문자 c를 분리자로해서 배열원소들을 연결해서 한 문자열로 반환함
reverse( )	배열의 원소들을 역순으로 변경함.
sort( )	배열을 오름차순으로 정렬함

[예 7.5] Array 객체의 특성

```
<script>
  var x = [ 80, 50, 40, 30, 60, 10, 90, 70, 20];
  document.write("(0) 배열 x의 크기(원소개수) : " + x.length + "<br>");
  document.write("(1) x의 출력(1) : " + x + "<br>");
  document.write("(2) x의 오름차순 정렬  : " + x.sort() + "<br>");
  document.write("(3) x의 출력(2) : " + x + "<br>");
  document.write("(4) x의 내림차순 출력 : " + x.reverse() + "<br>");

  subX = x.slice(2,5);
  document.write("(5) subX의 출력 : " + subX + "<hr>");

  var season1 = [ "봄", "여름", "가을", "겨울" ];
  var season2 = [ "춘곤기",  "환절기" ];
  seasons = season1.join(" -> ");
  season3 = season1.concat(season2);
  season4 = season2.concat(season1);
  document.write("(6) 계절 리스트(1) : " + season3 + "<br>");
  document.write("(7) 계절 리스트(2) : " + season4 + "<br>");
  document.write("(8) 계절의 변화 : " + seasons + "<br>");
  document.write("(9) '가을' 의 배열 인덱스 : "
              + season1.indexOf('가을') + "<br>");
</script>
```



(4) String 객체

- string 형(문자열) 자료들을 객체로 표현한 것
- 자바스크립트의 문자열 표현 방법
  ❶ 문자열 상수 표현 : string 자료형 ❷ String 객체 표현 : object 객체형

```
<script>
  var str1 = "javascript";   // ❶
  var str2 = new String("webprogramming");   // ❷
  document.write("typeof(str1):"+typeof(str1)+"<br>");
  document.write("typeof(str2):"+typeof(str2)+"<br>");   </script>
```

- String 객체들도 내부적으로 문자열 값을 변경할 수 없음

주요 속성과 메서드

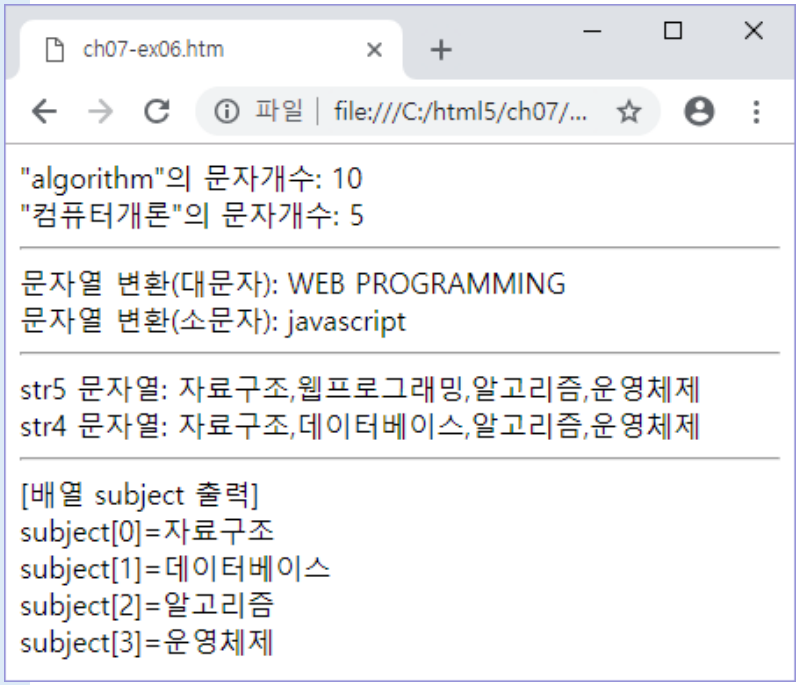
속성 및 메서드	의미 또는 기능
length	문자열 크기(길이)를 나타내는 속성
toUpperCase( ), toLowerCase( )	문자열의 영문자를 각각 대문자, 소문자로 변환함
concat(s)	문자열 끝에 문자열 s를 연결한 문자열을 반환함
indexOf(substring), match(s)	각각 부분 문자열 s를 시작 위치, 포함 여부를 반환함
replace(s1, s2)	부분 문자열 s1을 s2로 대체한 문자열을 반환함
split(s)	분리 문자열(s)를 기준으로 부분 문자열들로 분할해서 배열로 반환함
substr(i, n), substring(i, j)	각각 i부터 n개의, i부터 j 까지의 부분 문자열을 반환함

- 문자열 상수들도 이용할 수 있음(자동 형변환)

```
str1.Length=str1.length;   // ❸ str1=10
str3=str1.toUpperCase();   // ❹ str3="JAVASCRIPT"
str4=str1.concat("programming");   // ❺ str4="javascriptprogramming"
str5=str2.replace("web","HTML5");   // ❻ str5="HTML5programming"
str6=str2.split("");   // ❼ str6[0]="web", str6[1]="programming"
```

[예 7.6] String 객체의 속성과 메서드들

```
<script>
   str1 = "JAVASCRIPT";        str2 = "web programming";
   str3 = "컴퓨터개론";        str4 = "자료구조,데이터베이스,알고리즘,운영체제";

   document.write('"algorithm"의 문자개수: ' + str1.length + "<br>");
   document.write('"컴퓨터개론"의 문자개수: ' + str3.length + "<hr>");
   document.write("문자열 변환(대문자): " + str2.toUpperCase() + "<br>");
   document.write("문자열 변환(소문자): " + str1.toLowerCase() + "<hr>");
   str5 = str4.replace("데이터베이스", "웹프로그래밍");
   document.write("str5 문자열: " + str5 + "<br>");
   document.write("str4 문자열: " + str4 + "<hr>");
   document.write("[배열 subject 출력]<br>");
   subject = str4.split(',');
   for (i=0; i<subject.length; i++)
      document.write("subject[" + i + "]=" + subject[i] + "<br>");
</script>
```



(5) Number 객체

- number형 자료들을 객체로 표현한 것
- 자바스크립트의 수치 표현 방법
  ❶ 수치 상수 표현 : number 자료형 ❷ Number 객체 표현 : Number 객체형

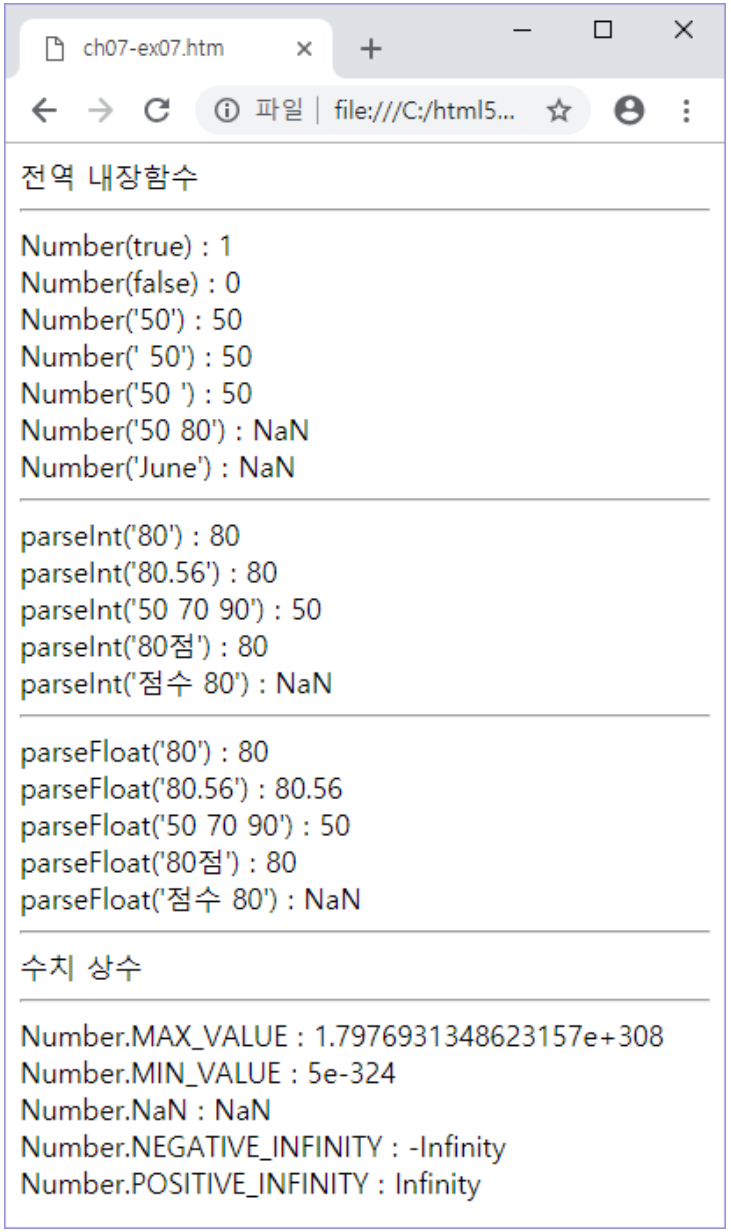
```
❶ var num1 = 92;  // 수치값 92(number형) 저장
❷ var num2 = new Number(256);   // Number 객체 256 생성후 저장
```

- number형 수치들과 Number 객체의 모든 속성과 메서드들을 이용할 수 있음
- Number 객체의 대문자 속성들은 반드시 "Number." 속성 형식으로만 사용해야 함
  (Number.MAXVALUE, Number. POSITIVE_INFINITY 등)

속성과 메서드	의미 또는 기능
MAX_VALUE, MIN_VALUE	표현할 수 있는 가장 큰 정수, 가장 작은 정수
POSITIVE_INFINITY, NEGATIVE_INFINITY	infinity, -infinity
NaN	jNaN(Not-a-Number)
toExponential(n)	수치를 소수점 이하 n 지수 표현 문자열로 변환함
toFixed(n), toPrecision(n)	소수점 이하 n의 고정소수, n의 유효자리 수치로 변환함
toString(n)	n진 수치 문자열로 변환함

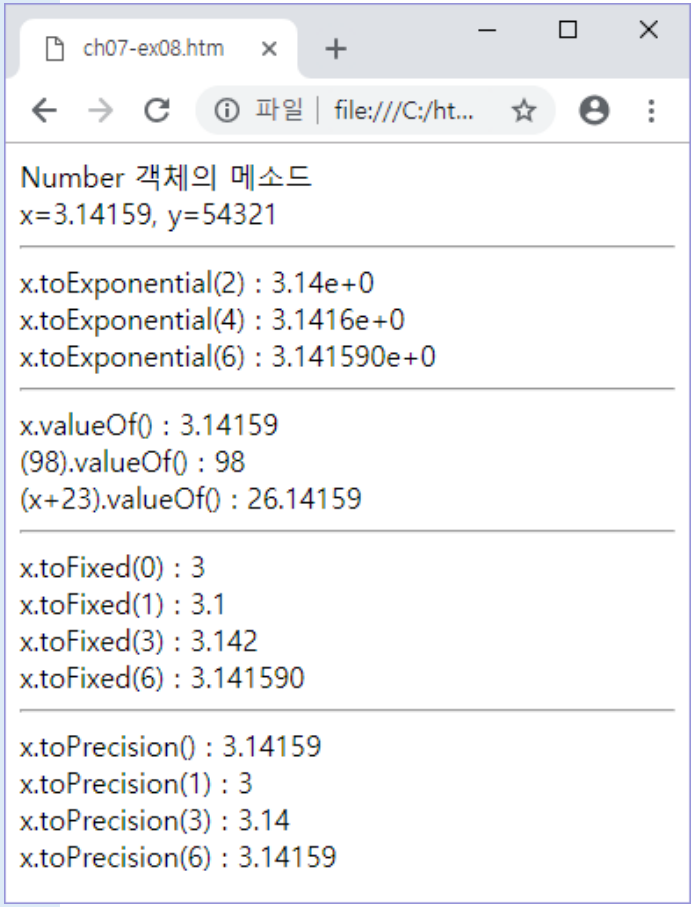
[예 7.7]

```
<script>
 document.write("전역 내장함수<hr>");
 document.write("Number(true) : " + Number(true) + "<br>");
 document.write("Number(false) : " + Number(false) + "<br>");
 document.write("Number('50') : " + Number("50") + "<br>");
 document.write("Number('  50') : " + Number("  50") + "<br>");
 document.write("Number('50  ') : " + Number("50  ") + "<br>");
 document.write("Number('50 80') : " + Number("50 80") + "<br>");
 document.write("Number('June') : " + Number("June")  + "<hr>");

 document.write("parseInt('80') : " + parseInt("80") + "<br>");
 document.write("parseInt('80.56') : " + parseInt("80.56") + "<br>");
 document.write("parseInt('50 70 90') : " + parseInt("50 70 90") + "<br>");
 document.write("parseInt('80점') : " + parseInt("80점") + "<br>");
 document.write("parseInt('점수 80') : " + parseInt("점수 80") + "<hr>");

 document.write("parseFloat('80') : " + parseFloat("80") + "<br>");
 document.write("parseFloat('80.56') : " + parseFloat("80.56") + "<br>");
 document.write("parseFloat('50 70 90') : " + parseFloat("50 70 90") + "<br>");
 document.write("parseFloat('80점') : " + parseFloat("80점")  + "<br>");
 document.write("parseFloat('점수 80') : " +parseFloat("점수 80")  + "<hr>");

 document.write("수치 상수<hr>");
 var x1 = Number.MAX_VALUE, x2 = Number.MIN_VALUE, x3 = Number.NaN;
 var x4 = Number.NEGATIVE_INFINITY, x5 = Number.POSITIVE_INFINITY;
 document.write("Number.MAX_VALUE : " + x1 + "<br>");
 document.write("Number.MIN_VALUE : " + x2 + "<br>");
 document.write("Number.NaN : " + x3 + "<br>");
 document.write("Number.NEGATIVE_INFINITY : " + x4 + "<br>");
 document.write("Number.POSITIVE_INFINITY : " + x5 + "<br>");
</script>
```



[예 7.8] 소수점 이하 세 자리까지 비교 예

```
<script>
  var x = 3.14159;  y = 54321;
  document.write("Number 객체의 메소드<br>");
  document.write("x=" + x + ",  y=" + y + "<hr>");
  document.write("x.toExponential(2) : " + x.toExponential(2) + "<br>");
  document.write("x.toExponential(4) : " + x.toExponential(4) + "<br>");
  document.write("x.toExponential(6) : " + x.toExponential(6) + "<hr>");

  document.write("x.valueOf() : " + x.valueOf() + "<br>");
  document.write("(98).valueOf() : " + (98).valueOf() + "<br>");
  document.write("(x+23).valueOf() : " + (x+23).valueOf() + "<hr>");

  document.write("x.toFixed(0) : " + x.toFixed(0) + "<br>");
  document.write("x.toFixed(1) : " + x.toFixed(1) + "<br>");
  document.write("x.toFixed(3) : " + x.toFixed(3) + "<br>");
  document.write("x.toFixed(6) : " + x.toFixed(6) + "<hr>");

  document.write("x.toPrecision() : " +  x.toPrecision() + "<br>");
  document.write("x.toPrecision(1) : " + x.toPrecision(1) + "<br>");
  document.write("x.toPrecision(3) : " + x.toPrecision(3) + "<br>");
  document.write("x.toPrecision(6) : " + x.toPrecision(6));
</script>
```

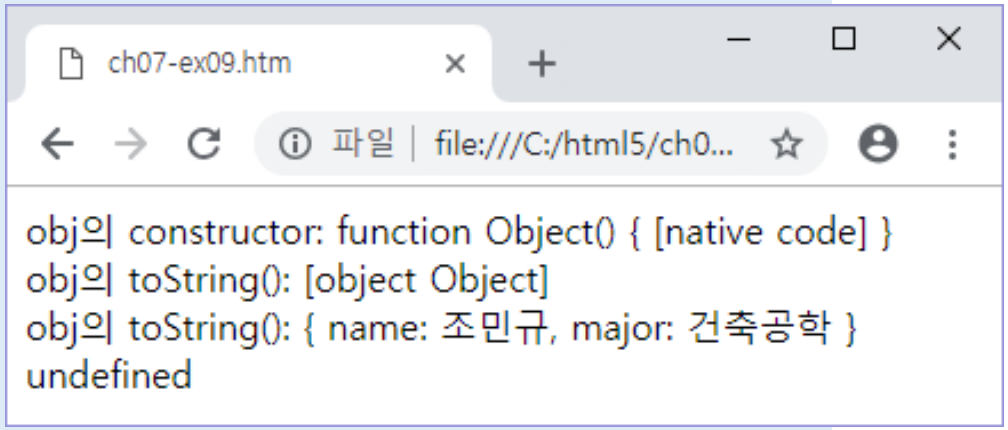


(6) Object 객체

- 모든 객체들의 최상위 객체
- 자바스크립트의 모든 객체들은 Object 객체의 모든 속성과 메모드를 상속받음

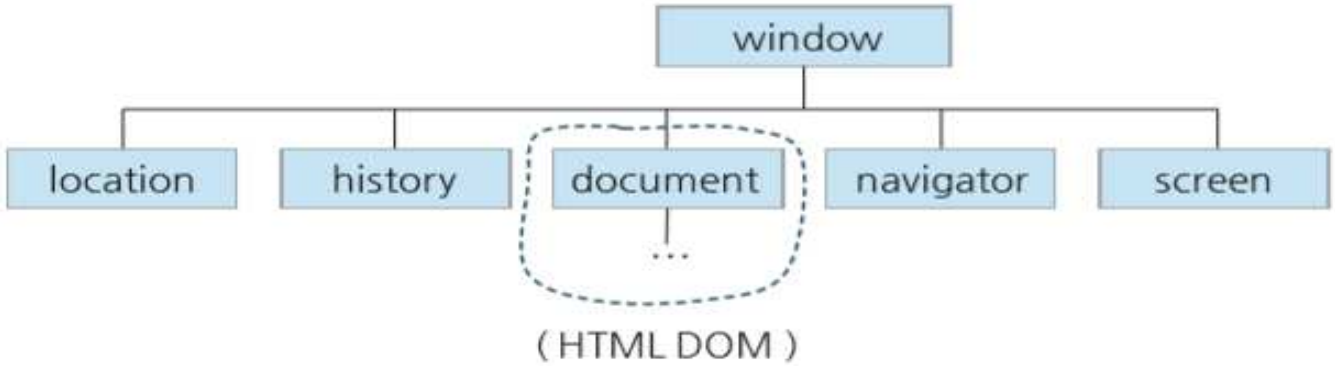
메서드	의미 또는 기능
constructor( )	객체의 생성자 함수
hasOwnProperty(name)	name을 속성으로 가지고 있는 여부를 판단함(true, false)
toString(n)	수치를 n진 수치 표현 문자열로 변환함

```
<script>
   var obj = new Object();
   obj.name = "조민규";  obj.major = "건축공학";

   document.write("obj의 constructor: ");
   document.write(obj.constructor + "<br>");
   document.write("obj의 toString():  ");
   document.write(obj.toString()); // 상속 메서드
   document.write("<br>");

   // toString() 재정의
   obj.toString   = function() {
      document.write("{ name: " + this.name   + ", major: " + this.major + " }<br>");
   };
   document.write("obj의 toString(): ");
   document.write(obj.toString());  // 재정의 메서드
</script>
```



7.2 브라우저 객체 모델

- 웹 브라우저는 웹문서를 로드하면, 현재 웹 브라우저와 관련된 정보들을 객체로 표현해서 제공함
- 브라우저 객체 모델(BOM, Brower Object Model)



  - window 객체 : 현재 열린 웹 브라우저 윈도우 관련 정보를 나타냄
  - document 객체 : 웹브라우저에 로드된 HTML 문서 정보를 나타냄 (HTML DOM)
  - location과 history 객체 : 현재 윈도우의 방문이력과 이동 관련 정보를 지원함
  - natvigator, screen 객체 : 웹 브라우저와 모니터 정보를 제공함

- BOM 객체들은 웹 브라우저를 실행하면 자동으로 생성되어 자바스크립트에게 제공됨

7.2.1 window 객체

- BOM의 최상위 객체로서 현재 실행되고 있는 웹 브라우저 창을 나타냄
- 자바스크립트 코드에서 정의하는 모든 전역 변수와 함수, 객체들은 window 객체의 멤버임
  - 예) alert(), conform(), prompt() 함수도 window 객체의 메서드들임
- window 객체는 새로운 윈도우를 열 때마다 생성되고 닫으면 소멸됨

속성 및 메서드	의미 또는 기능
onload	로드 이벤트 속성
open(url, name, winSpecs), close( )	각각 윈도우를 열고, 닫는다
setTimeout(f, t),	t 시간(millsec)이 경과된 후, 함수 f를 1회 실행함
setTimeInterval(f, t),	t 시간 간격(millsec)으로 함수 f를 반복 실행함
clearTimeout(id), clearInterval(id)	각각 setTimeout( ), setTimeInterval( )의 실행을 중지시킴
print( )	현재 윈도우 창에 표시된 페이지를 출력함

(1) 동적 윈도우 열기와 닫기 : window.open(), window.close()

- window.open() : 윈도우 생성해서 나타냄
- window.close() : 열린 윈도우를 닫고 삭제함

[window.open() 을 이용한 윈도우(웹브라우저 창) 생성방법]

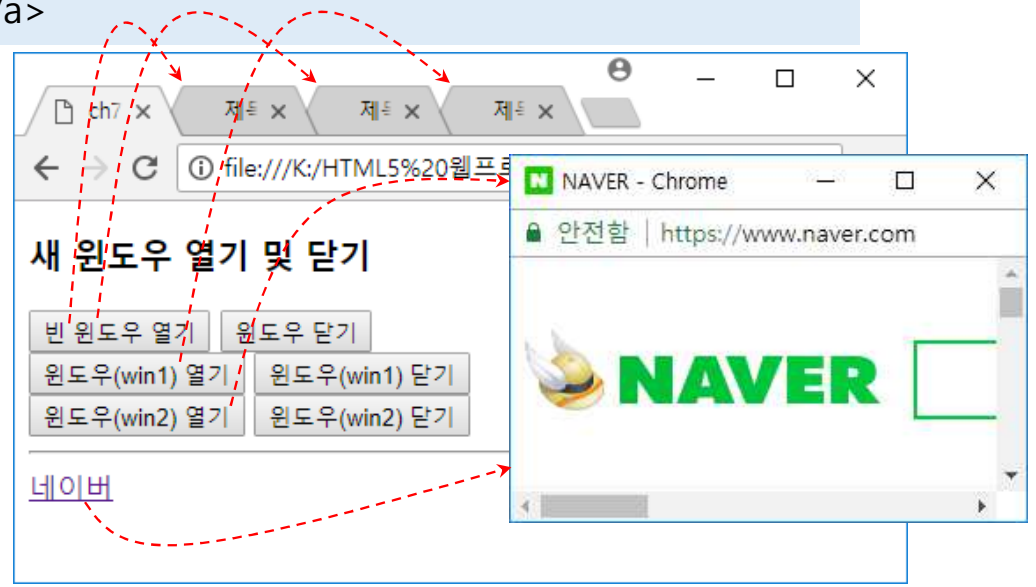
```
❶ window.open();   // 새윈도우를 생성해서 나타냄
❷ window.open('l', 'win1');   // 윈도우이름(win1)을 지정함
❸ window.open('url', '', "width=300, height=600");   // 로드할 문서(url)와 윈도우크기 등을 지정함
```

인수 속성	속성값	
width, height	정수(pixcels)	윈도우의 가로, 세로 크기 지정
left, top	정수(pixcels)	윈도우의 left, top 위치 지정
menubar, status, titlebar	yes, no, 1, 0	메뉴바, 상태바, 타이블바의 표시 여부
location, scrollbars, toolbar	yes, no, 1, 0	주소필드, 스크롤바, 툴바의 표시 여부
resizable, fullscreen	yes, no, 1, 0	윈도우의 크기 조정, 전체 화면 지원 여부

[예 7.10] window.open(), window.close()의 사용

```
<h3> 새 윈도우 열기 및 닫기 </h3>
 <button onclick="w1=window.open(); " >빈 윈도우 열기</button>
 <button onclick="w1.close();" >윈도우 닫기</button> <br>

 <button onclick="w2=window.open('','win1');" >윈도우(win1) 열기</button>
 <button onclick="w2.close();" >윈도우(win1) 닫기</button> <br>

 <button onclick="w3=window.open('','win2', 'width=250, height=150');" >윈도우(win2) 열기</button>
 <button onclick="w3.close();" > 윈도우(win2) 닫기 </button> <br><hr>
 <a href="http://naver.com" target="win2" > 네이버 </a>
```

- window.open()은 열린 윈도우의 식별값을 반환함
- 이 식별값과 window.close() 를 이용하면 현재 열린 윈도우를 동적으로 닫을 수 있음



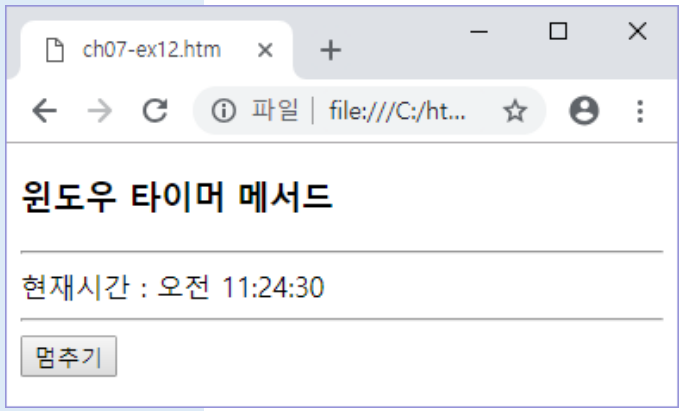
(2) 윈도우 타이머 메서드 : **setTimeout(), setTimeInterval()**

- 윈도우의 함수 타이머 기능 제공함
- window 객체의 메서드들 중에서 가장 많이 사용됨
- 특히, 시각효과나 애니메이션 효과를 나타낼 때 거의 필수적으로 사용함
  - setTimeout(f,t) : t 시간이 경과된 후에 함수 f를 실행함(함수 f는 한번만 실행됨)
  - setTimeInterval(f, t) : t 시간 간격으로 함수 f를 반복적으로 실행함

```
id1 = window.setTimeout(f1, 5000);    // ❶
id2 = window.setTimeInterval(f2, 2000);    // ❷
...
clearTimeout(id1);    // ❶의 setTimeout(f1,5000)의 실행을 중지함
clearInterval(id2);    // ❷의 setTimeInterval(f2,2000)의 실행을 중지함
```

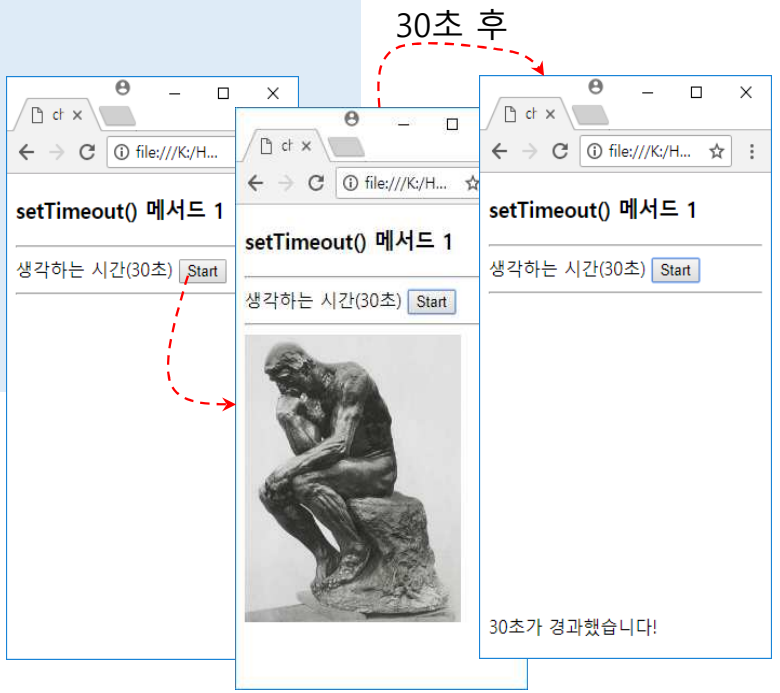
[예 7.12]

```
<h3> 윈도우 타이머 메서드 </h3><hr>
 현재시간 : <span id="clock"></span> <hr>
<button onclick="clearInterval(t1)">멈추기</button>
<script>
   var t1 = setInterval(myClock, 1000);
   function myClock() {
      var currentTime = new Date();
      document.getElementById("clock").innerHTML
               = currentTime.toLocaleTimeString();
   }
</script>
```



[예 7.11] setTimeout()의 활용

```
<h3>  setTimeout() 메서드 1</h3><hr>
  생각하는 시간(30초)
  <button onclick="thinkingStart();">Start</button><hr>
  <img id="img1" style="visibility:hidden", src="thinking.png" width=200  height=250 />
  <p id="noti"> </p>
  <script>
    function thinkingStart() {
       document.images[0].style.visibility="visible";
       setTimeout(myTimer, 30000);
    }
    function myTimer() {
       document.images[0].style.visibility= "hidden";
       document.getElementById("noti").innerHTML = "30초가 경과했습니다!" ;
    }
  </script>
```

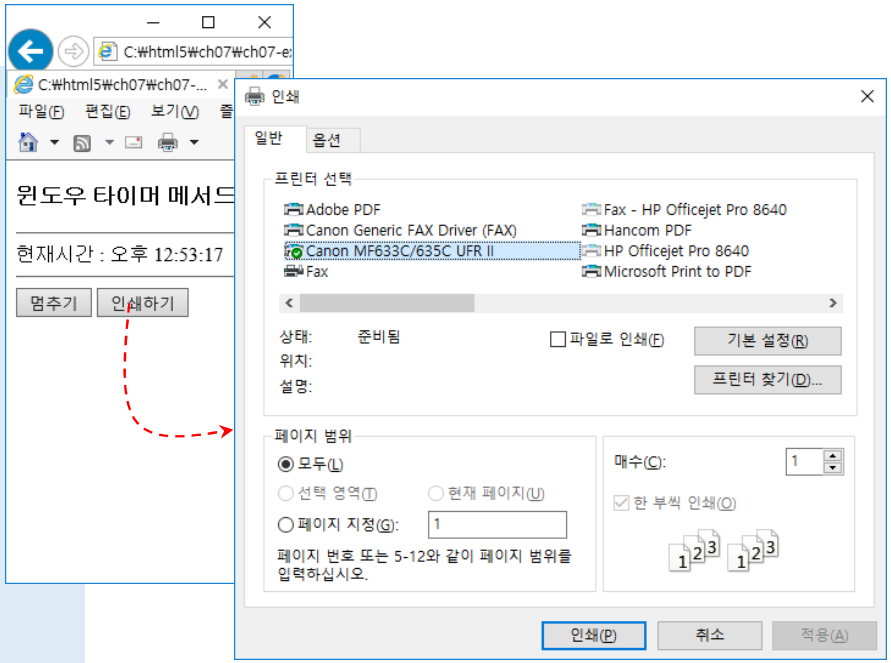


(3) 윈도우 표시 내용의 출력 : window.print()

- window.print() : 현재 윈도우에 표시된 문서 내용의 출력기능을 제공하는 메서드
- 프린터 대화상자를 제공함

[예 7.13] window.print()를 이용한 현재 화면 내용 출력

```
<h3> 윈도우 타이머 메서드 </h3><hr>
  현재시간 : <span id="clock"></span> <hr>
  <button onclick="clearInterval(t1)">멈추기</button>
  <button onclick="window.print()"> 인쇄하기 </button>

  <script>
    var t1 = setInterval(myClock, 1000);
    function myClock() {
      var currentTime = new Date();
      document.getElementById("clock").innerHTML
              = currentTime.toLocaleTimeString();
    }
  </script>
```



7.2.2 location, history 객체

(1) location 객체

- 웹 브라우저의 주소 표시 영역을 나타내는 객체
- location 객체를 이용하면, 현재 웹문서를 다른 문서들로 교체(이동)할 수 있음

속성 및 메서드	의미 또는 기능
url	현재 웹 브라우저에 로드된 문서의 URL
protocol, hostname, pathname, port	URL의 프로토콜, 호스트명, 패스명, 포트번호를 반환함
assign(url) , reload( )	새로운 페이지, 현재 페이지를 로드함
replace(url)	현재 웹 페이지를 url 페이지로 대체함

- location 객체를 이용한 동적 문서 이동 방법

```
❶ location='url';  또는 location.href='url';   // url로 지정한 문서를 로드함
❷ location.assign('url');   // url로 지정한 문서를 로드함
❸ location.replace();   // url로 지정한 문서를 로드함
❹ location.reload();  // 현재 문서를 다시 로드함
```

[예 7.14] location 객체를 이용한 문서 연결

page1.htm

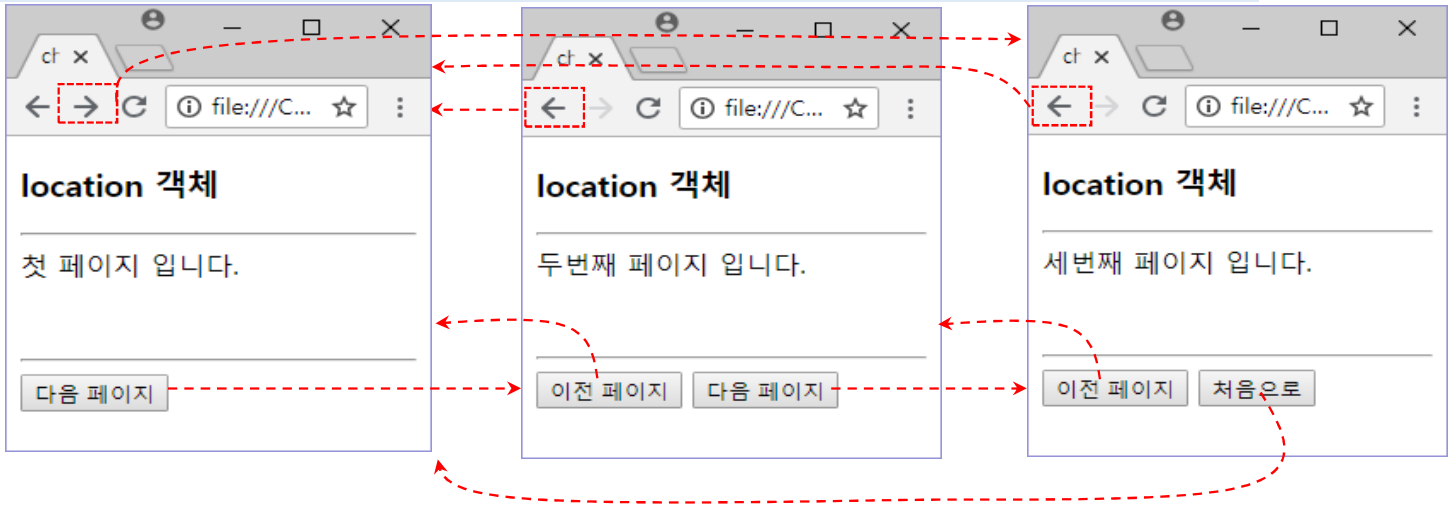
```
<h3> location 객체 </h3><hr>
 첫 페이지 입니다. <br> <br><br><hr>
 <button onclick="location.href='page2.htm';">다음 페이지</button>
```

page2.htm

```
<h3> location 객체 </h3><hr>
 두번째 페이지 입니다. <br> <br><br><hr>
 <button onclick="location.assign('page1.htm');">이전 페이지</button>
 <button onclick="location.replace('page3.htm');">다음 페이지</button>
```

page3.htm

```
<h3> location 객체 </h3><hr>
 세번째 페이지 입니다. <br> <br><br><hr>
 <button onclick="location.assign('page2.htm');">이전 페이지</button>
 <button onclick="location.assign('page1.htm');">처음으로</button>
```



(2) history 객체

- 현재 웹 브라우저의 히스토리 정보를 제공하는 객체
- 웹 브라우저의 히스토리 정보
  - ✓ 현재 실행된 웹 브라우저에서 사용자가 로드했던 웹 문서들에 대한 리스트

- history 객체를 이용하면, 사용자가 방문했던 웹 문서들로 신속하게 이동시킬 수 있음
- history 객체는 작성자가 제어하는 문서들의 이동에서 제한적으로 사용하는 것이 바람직함

속성 및 메서드	의미 또는 기능
length	히스토리에 등록된 웹 페이지(사이트) 개수
back( ), forward( )	각각 히스토리의 이전, 다음 페이지로 이동함
go(n)	현재 웹 페이지를 기준으로 히스토리의 n번째 페이지로 이동함

[예 7.15] history 객체를 이용한 문서 연결

page1.htm

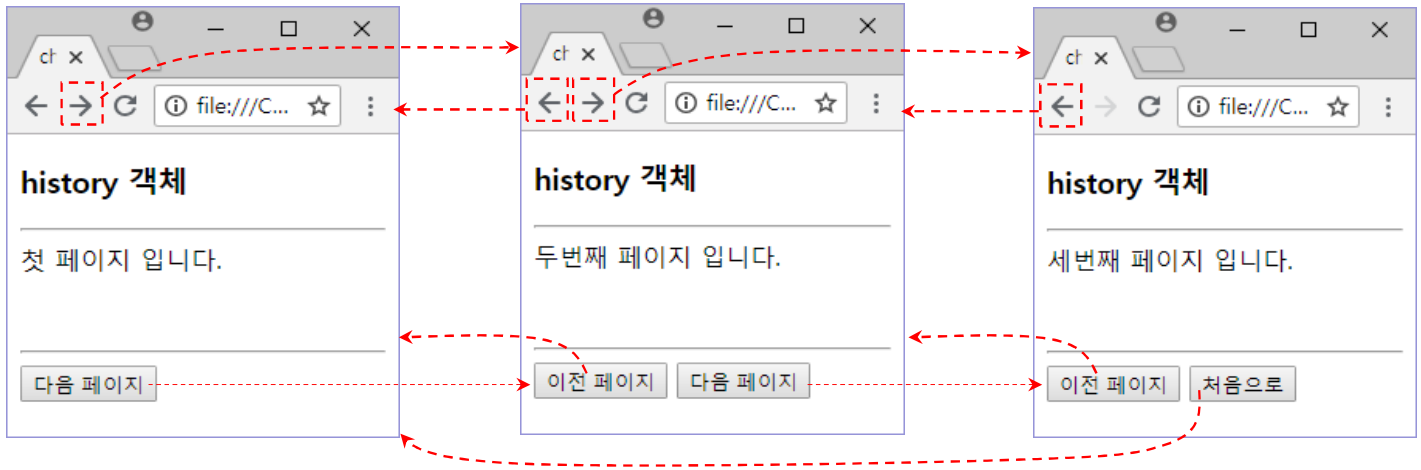
```
<h3> history 객체 </h3><hr>
 첫 페이지 입니다. <br> <br><br><hr>
 <button onclick="location.href='page2.htm';">다음 페이지</button>
```

page2.htm

```
<h3> history 객체 </h3><hr>
 두번째 페이지 입니다. <br> <br><br><hr>
 <button onclick="history.back();">이전 페이지</button>
 <button onclick="location.href='page3.htm';">다음 페이지</button>
```

page3.htm

```
<h3> history 객체 </h3><hr>
 세번째 페이지 입니다. <br> <br><br><hr>
 <button onclick="history.back();">이전 페이지</button>
 <button onclick="history.go(-2)">처음으로</button>
```



7.2.3 navigator, screen 객체

(1) navigator 객체

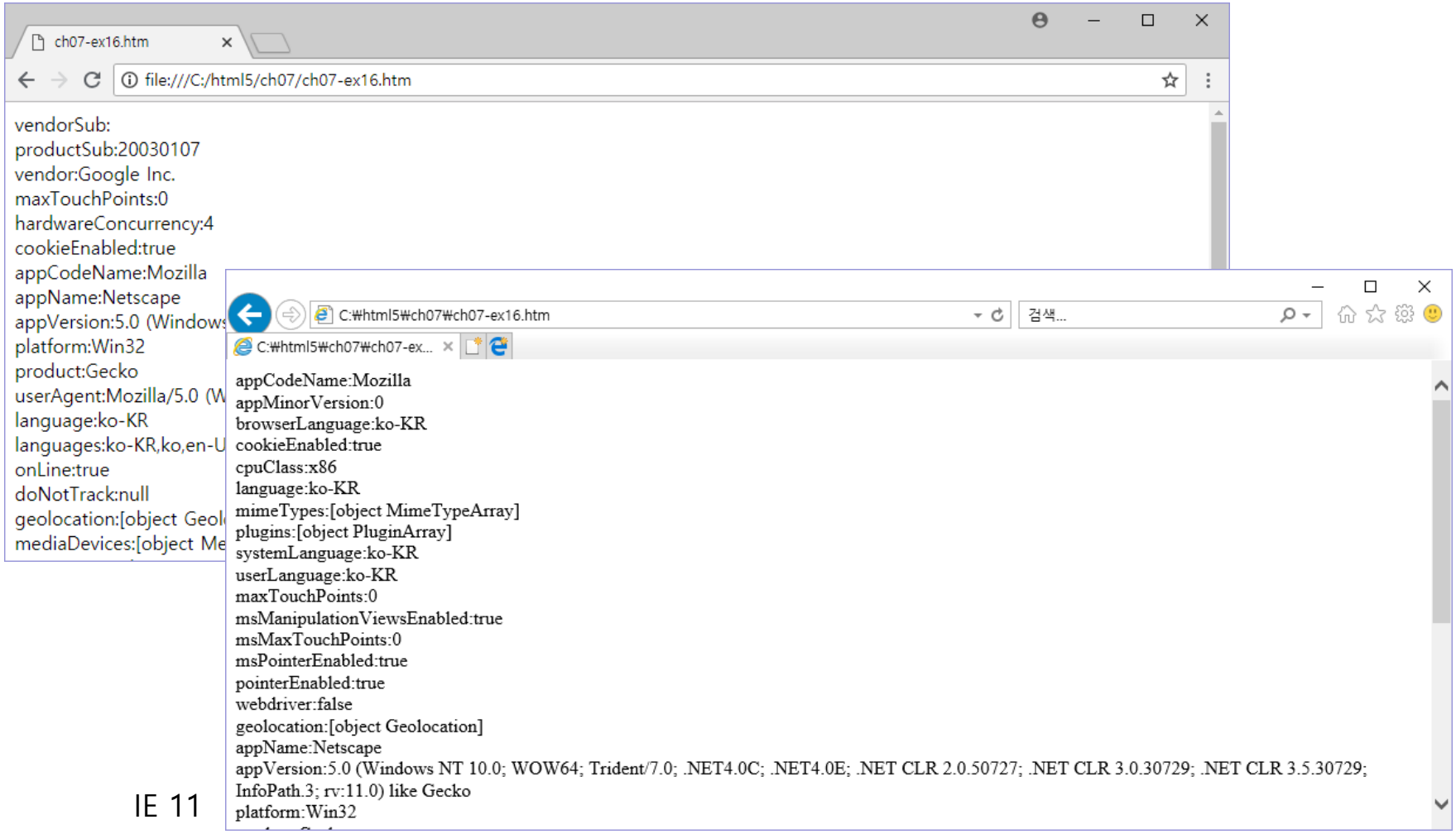
- 현재 실행중인 웹 브라우저에 대한 정보를 나타냄

속성 및 메서드	의미 또는 기능
appCodeName, appName, appVersion	각각 웹 브라우저의 코드명, 이름, 버전
platform	현재 브라우저가 실행되고 있는 운영체제
userAgent	브라우저 전체 정보

[예 7.16] 현재 실행중인 내비게이션 정보 확인

```
<script>
   var output = "";
   for (var info in navigator) {
      output = output + info + ':' + navigator[info] + '<br>';
   }
   document.write(output);
</script>
```

구글 크롬


ch07-ex16.htm
file:///C:/html5/ch07/ch07-ex16.htm
vendorSub:
productSub:20030107
vendor:Google Inc.
maxTouchPoints:0
hardwareConcurrency:4
cookieEnabled:true
appCodeName:Mozilla
appName:Netscape
appVersion:5.0 (Windows
platform:Win32
product:Gecko
userAgent:Mozilla/5.0 (W
language:ko-KR
languages:ko-KR,ko,en-U
onLine:true
doNotTrack:null
geolocation:[object Geol
mediaDevices:[object Me
C:₩html5₩ch07₩ch07-ex16.htm
검색...
C:₩html5₩ch07₩ch07-ex...
appCodeName:Mozilla
appMinorVersion:0
browserLanguage:ko-KR
cookieEnabled:true
cpuClass:x86
language:ko-KR
mimeTypes:[object MimeTypeArray]
plugins:[object PluginArray]
systemLanguage:ko-KR
userLanguage:ko-KR
maxTouchPoints:0
msManipulationViewsEnabled:true
msMaxTouchPoints:0
msPointerEnabled:true
pointerEnabled:true
webdriver:false
geolocation:[object Geolocation]
appName:Netscape
appVersion:5.0 (Windows NT 10.0; WOW64; Trident/7.0; .NET4.0C; .NET4.0E; .NET CLR 2.0.50727; .NET CLR 3.0.30729; .NET CLR 3.5.30729;
InfoPath.3; rv:11.0) like Gecko
platform:Win32

IE 11

(2) screen 객체

- 현재 웹 브라우저가 실행되는 운영체제 화면, 즉, 컴퓨터 모니터에 대한 정 보를 가지고 있는 객체

속성 및 메서드	의미 또는 기능
width, height	각각 컴퓨터 화면(모니터)의 너비, 높이
availWidth, availHeight	각각 컴퓨터 화면에서 실제로 사용 가능한 너비, 높이
colorDepth	사용 가능한 색상 수
pixelDepth	한 픽셀당 비트 수

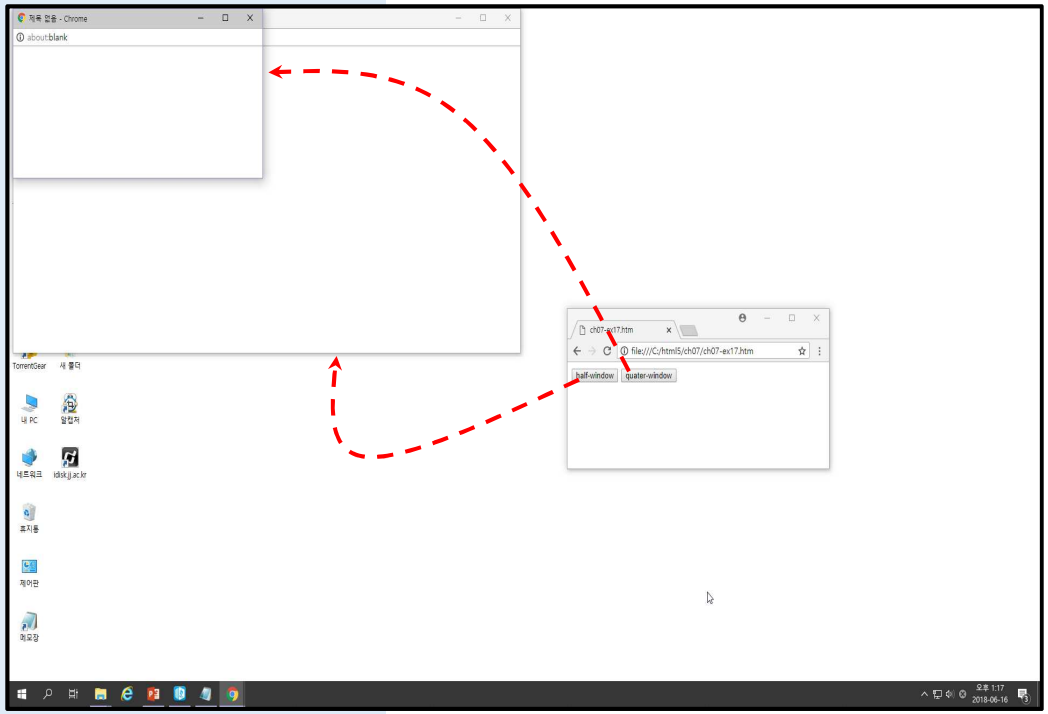
- screen 객체에서 많이 사용되는 속성 : width, height

```
예)  window.open('','','width=screen.width,height=screen.height');
```

[예 7.17] screen 객체를 이용한 윈도우 크기 지정

- 전체 스크린 크기의 ½, ¼ 크기의 윈도우를 생성함

```
<script>
  function halfWindow() {
    var w1=window.open('', '', 'width=200, height=150');
    var width = screen.width / 2;
    var height = screen.height / 2;
    w1.top += 30; w1.left += 30;
    w1.resizeTo(width, height);
  }

  function quaterWindow() {
    var w1=window.open('', '', 'width=200, height=150');
    var width = screen.width / 4;
    var height = screen.height / 4;
    w1.top=0; w1.left=0;
    w1.resizeTo(width, height);
  }
</script>
 <button onclick="halfWindow();"> half-window </button>
 <button onclick="quaterWindow();"> quater-window </button>
```



Next 8장

HTML DOM과 동적 문서 작성