

인터넷활용

1. HTML5+CSS3+Javascript 웹프로그래밍
2. 10주차

강의 목표

1. 객체의 기본 개념을 간단히 이해한다.
2. 브라우저가 제공하는 기본 객체(코어 객체)들의 종류를 알고 사용할 수 있다.
3. Date 객체를 활용할 수 있다.
4. String 객체를 활용할 수 있다.
5. 자바스크립트 배열을 만들 수 있다.
6. Array 객체를 이용하여 배열을 만들고 활용할 수 있다.
7. Math 객체를 활용할 수 있다.

객체 개념

□ 현실 세계는 객체들의 집합

- 사람, 책상, 자동차, TV 등

- 객체는 자신만의 고유한 구성 속성

- 자동차: <색상:오렌지, 배기량:3000CC, 제조사:한성, 번호:서울1-1>
- 사람: <이름:이재문, 나이:20, 성별:남, 주소:서울>
- 은행계좌: <소유자:황기태, 계좌번호:111, 잔액:35000원>



자동차 객체(car)



은행 계좌(account)

객체 개념 (사칙연산)

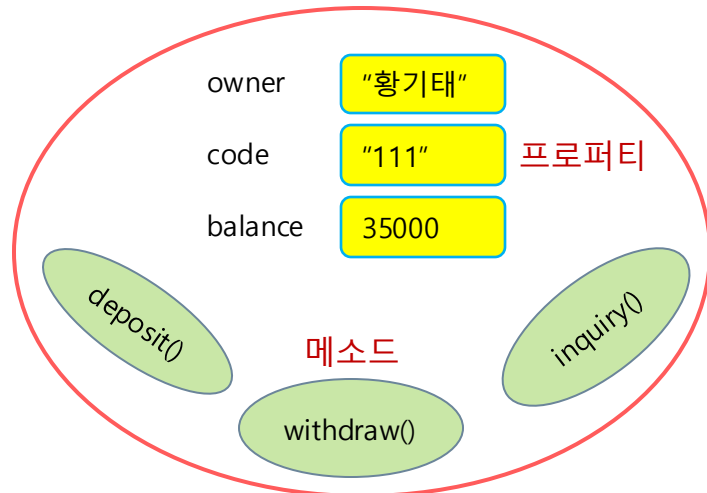
4

항목	절차 지향 언어 (예: C)	객체 지향 언어 (예: Python, Java)
덧셈, 뺄셈 등 기능	add(), sub() 같은 함수만 따로 존재	Calculator라는 객체 안에 기능이 들어있음
계산기 만들기	덧셈, 뺄셈 함수를 하나하나 따로 작성	계산기라는 하나의 객체로 묶어서 처리
코드 구조	모든 계산을 순서대로 처리	계산기 객체가 스스로 동작함
사용 방법	add(a, b) 처럼 함수 이름으로 호출	calc.add(a, b)처럼 객체 이름으로 호출
확장성 (기능 추가)	곱셈, 나눗셈 추가하려면 새 함수 작성 + 수정 많음	Calculator 클래스에 기능만 추가하면 끝

자바스크립트 객체

□ 자바스크립트 객체 구성

- ▣ 여러 개의 프로퍼티(property)와 메소드로 구성
 - 프로퍼티 : 객체의 고유한 속성(변수)
 - 메소드(method) : 함수



자바스크립트 객체 account

```
let account = {  
  owner    : "황기태",  
  code     : "111",  
  balance  : 35000,  
  deposit  : function() { ... },  
  withdraw : function() { ... },  
  inquiry  : function() { ... },  
};
```

account 객체를 만드는 자바스크립트 코드

자바스크립트 객체 종류

- 자바스크립트는 객체 기반 언어
 - ▣ 자바스크립트는 객체 지향 언어 아님 -> 2015년 이후 class 지원. 객체 지향 언어라고 볼 수 있음.
- 자바스크립트 객체의 유형
 1. 코어 객체
 - 자바스크립트 언어가 실행되는 어디서나 사용 가능한 기본 객체
 - 기본 객체로 표준 객체
 - Array, Date, String, Math 타입 등
 - 웹 페이지 자바스크립트 코드에서 혹은 서버에서 사용 가능
 2. HTML DOM 객체
 - HTML 문서에 작성된 각 HTML 태그들을 객체화한 것들
 - HTML 문서의 내용과 모양을 제어하기 위한 목적
 - W3C의 표준 객체
 3. 브라우저 객체
 - 자바스크립트로 브라우저를 제어하기 위해 제공되는 객체
 - BOM(Browser Object Model)에 따르는 객체들
 - 비표준 객체

코어 객체

□ 코어 객체 종류

- ▣ Array, Date, String, Math 등

□ 코어 객체 생성

- ▣ new 키워드 이용

```
let today = new Date();      // 시간 정보를 다루는 Date 타입의 객체 생성  
let msg = new String("Hello"); // "Hello" 문자열을 담은 String 타입의 객체 생성
```

- ▣ 객체가 생성되면 객체 내부에 프로퍼티와 메소드들 존재

□ 객체 접근

- ▣ 객체와 멤버 사이에 점(.) 연산자 이용

```
obj.프로퍼티 = 값;           // 객체 obj의 프로퍼티 값 변경  
변수 = obj.프로퍼티;        // 객체 obj의 프로퍼티 값 알아내기  
obj.메소드(매개변수 값들);   // 객체 obj의 메소드 호출
```

예제 7-1 자바스크립트 객체 생성 및 활용

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>객체 생성 및 활용</title>
</head>
<body>
<h3>객체 생성 및 활용</h3>
<hr>
<script>
  // Date 객체 생성
  let today = new Date();

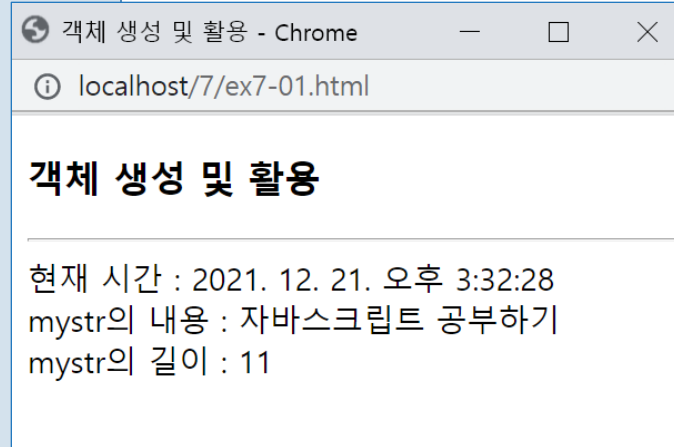
  // Date 객체의 toLocaleString() 메소드 호출
  document.write("현재 시간 : " + today.toLocaleString()
    + "<br>");

  // String 객체 생성
  let mystr= new String("자바스크립트 공부하기");
  document.write("mystr의 내용 : " + mystr + "<br>");
  document.write("mystr의 길이 : " + mystr.length + "<br>");
  // mystr.length=10; // 이 문장은 오류이다.
</script>
</body>
</html>
```

객체 생성

메소드 호출

프로퍼티 읽기



자바스크립트 배열

□ 배열

- 여러 개의 원소들을 연속적으로 저장
- 전체를 하나의 단위로 다루는 데이터 구조

□ 배열 생성 사례

```
let cities = ["Seoul", "New York", "Paris"];
```

cities	"Seoul"	<i>cities[0]</i>
	"New York"	<i>cities[1]</i>
	"Paris"	<i>cities[2]</i>

```
let n = [4, 5, -2, 28, 33];
```

n	4	5	-2	28	33
	<i>n[0]</i>	<i>n[1]</i>	<i>n[2]</i>	<i>n[3]</i>	<i>n[4]</i>

- 0에서 시작하는 인덱스를 이용하여 배열의 각 원소 접근

```
let name = cities[0];           // name은 "Seoul"  
cities[1] = "Gainesville";      // "New York" 자리에 "Gainesville" 저장
```

자바스크립트에서 배열을 만드는 방법

10

□ 배열 만드는 2가지 방법

- ▣ []로 배열 만들기
- ▣ Array 객체로 배열 만들기

□ []로 배열 만들기

- ▣ [] 안에는 원소들의 초기 값 나열

```
let week = ["월", "화", "수", "목", "금", "토", "일"];  
let plots = [-20, -5, 0, 15, 20];
```

- ▣ 배열 크기 : 배열의 크기는 고정되지 않고 원소 추가 시 늘어남

- 배열의 끝에 원소 추가

```
plots[5] = 33; // plots 배열에 6번째 원소 추가. 배열 크기는 6이 됨  
plots[6] = 22; // plots 배열에 7번째 원소 추가. 배열 크기는 7이 됨
```

- **주의** : 현재 배열보다 큰 인덱스에 원소를 추가하면 값이 비어 있는 중간의 원소들도 생기는 문제 발생

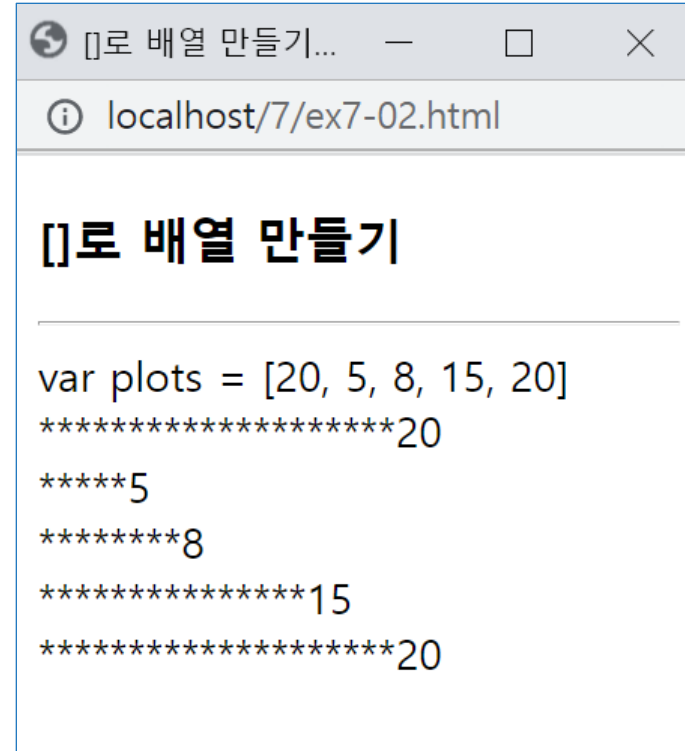
```
plots[10] = 33; // 주의. plots 배열의 크기는 11개가 되고,  
// plots[7], plots[8], plots[9]의 값은 모두 undefined 값
```

예제 7-2 []로 배열 만들기

[]로 정수 5를 저장할 배열을 만들고, 원소의 값만큼 '*'를 출력하는 프로그램을 작성하라.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>[]로 배열 만들기</title>
</head>
<body>
<h3>[]로 배열 만들기</h3>
<hr>
<script>
    let plots = [20, 5, 8, 15, 20]; // 원소 5개의 배열 생성
    document.write("var plots = [20, 5, 8, 15, 20]<br>");

    for(let i=0; i<5; i++) {
        let size = plots[i]; // plots 배열의 i번째 원소
        while(size>0) {
            document.write("*");
            size--;
        }
        document.write(plots[i] + "<br>");
    }
</script>
</body>
</html>
```



Array로 배열 만들기

12

□ 초기 값을 가진 배열 생성

```
let week = new Array("월", "화", "수", "목", "금", "토", "일");
```

□ 초기화되지 않은 배열 생성

▣ 일정 크기의 배열 생성 후 나중에 원소 값 저장

```
let week = new Array(7); // 7개의 원소를 가진 배열 생성
```

```
week[0] = "월";  
week[1] = "화";  
...  
week[6] = "일";
```

□ 빈 배열 생성

▣ 원소 개수를 예상할 수 없는 경우

```
let week = new Array(); // 빈 배열 생성
```

```
week[0] = "월"; // 배열 week 크기 자동으로 1  
week[1] = "화"; // 배열 week 크기 자동으로 2
```

배열의 원소 개수, length 프로퍼티

13

□ 배열의 크기 : Array 객체의 length 프로퍼티

```
let plots = [-20, -5, 0, 15, 20];  
let week = new Array("월", "화", "수", "목", "금", "토", "일");  
let m = plots.length; // m은 5  
let n = week.length; // n은 7
```

▣ length 프로퍼티는 사용자가 임의로 값 변경 가능

- length 프로퍼티는 Array 객체에 의해 자동 관리
- 사용자가 임의로 값 변경 가능

- 배열의 크기를 줄이거나 늘릴 수 있음
- 예

```
plots.length = 10; // plots의 크기는 5에서 10으로 늘어남  
plots.length = 2; // plots의 크기는 2로 줄어 들어,  
// 처음 2개의 원소 외에는 모두 삭제 됨
```

예제 7-3 Array 객체로 배열 만들기

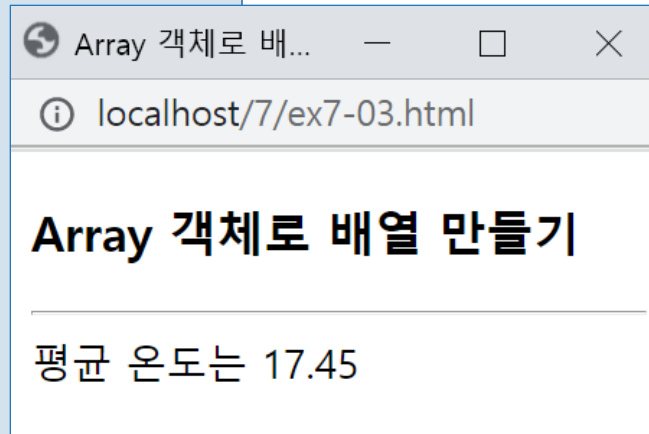
```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Array 객체로 배열 만들기</title>
</head>
<body>
<h3>Array 객체로 배열 만들기</h3>
<hr>
<script>
  let degrees = new Array(); // 빈 배열 생성
  degrees[0] = 15.1;
  degrees[1] = 15.4;
  degrees[2] = 16.1;
  degrees[3] = 17.5;
  degrees[4] = 19.2;
  degrees[5] = 21.4;

  let sum = 0;
  for(let i=0; i<degrees.length; i++)
    sum += degrees[i];

  document.write("평균 온도는 " + sum/degrees.length + "<br>");
</script>
</body>
</html>
```

배열 크기만큼 루프

배열 degrees의 크기, 6



배열의 특징

15

- 배열은 Array 객체
 - ▣ []로 생성해도 Array 객체로 다루어짐
- 배열에 여러 타입의 데이터 섞여 저장 가능

```
let any = new Array(5);    // 5개의 원소를 가진 배열 생성
any[0] = 0;
any[1] = 5.5;
any[2] = "이미지 벡터";    // 문자열 저장
any[3] = new Date();       // Date 객체 저장
any[4] = convertFunction;  // function convertFunction()의 주소 저장
```

표 7-2: Array 객체의 주요 메서드

16

메서드	설명
concat(arr)	현재 배열에 배열 arr의 요소를 덧붙여 새 배열을 생성하여 반환함 . 원본 배열은 변경되지 않음
join([separator])	배열의 모든 요소를 하나의 문자열로 만들어 반환함. 각 요소 사이는 separator로 구분하며, 생략하면 ","가 기본값
reverse()	배열의 요소를 역순으로 정렬하며 원본 배열을 직접 변경함 . 반환값도 변경된 배열
slice(idxA[, idxB])	인덱스 idxA부터 idxB 직전까지의 요소를 추출하여 새 배열을 반환함 . 원본 배열은 변경되지 않음
sort()	배열의 요소들을 문자열 형태로 정렬하며 원본 배열을 변경함 . 숫자 정렬 시 비교 함수를 직접 작성해야 정확한 정렬 가능
toString()	배열의 모든 요소를 문자열로 변환하여 ","로 연결된 하나의 문자열로 반환함. join()과 유사하지만 separator를 지정할 수 없음

예제 7-4 Array 객체의 메소드 활용

```
<!DOCTYPE html>
<html><head><meta charset="utf-8"> <title>Array 객체의 메소드 활용</title>
<script>
    function pr(msg, arr) { document.write(msg + arr.toString() + "<br>"); }
</script>
</head>
<body>
<h3>Array 객체의 메소드 활용</h3>
<hr>
<script>
    let a = new Array("황", "김", "이");
    let b = new Array("박");
    let c;

    pr("배열 a = ", a);
    pr("배열 b = ", b);
    document.write("<hr>");

    c = a.concat(b); // c는 a와 b를 연결한 새 배열
    pr("c = a.concat(b)후 c = ", c);
    pr("c = a.concat(b)후 a = ", a);

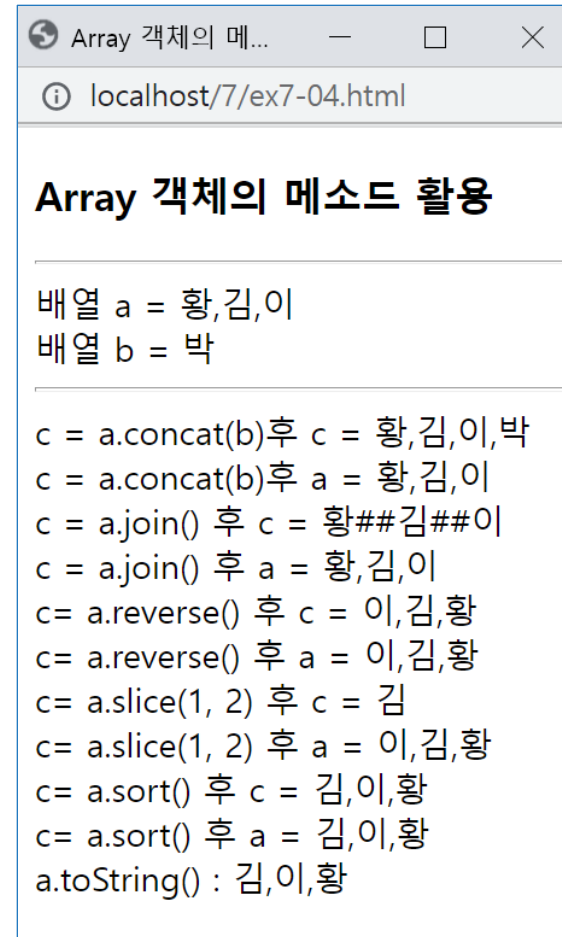
    c = a.join("##"); // c는 배열 a를 연결한 문자열
    pr("c = a.join(\"##\") 후 c = ", c);
    pr("c = a.join(\"##\") 후 a = ", a);

    c = a.reverse(); // a.reverse()로 a 자체 변경. c는 배열
    pr("c= a.reverse() 후 c = ", c);
    pr("c= a.reverse() 후 a = ", a);

    c = a.slice(1, 2); // c는 새 배열
    pr("c= a.slice(1, 2) 후 c = ", c);
    pr("c= a.slice(1, 2) 후 a = ", a);

    c = a.sort(); // a.sort()는 a 자체 변경. c는 배열
    pr("c= a.sort() 후 c = ", c);
    pr("c= a.sort() 후 a = ", a);

    c = a.toString(); // toString()은 원소 사이에 ","를 넣어 문자열 생성
    document.write("a.toString() : " + c); // c 는 문자열
</script></body></html>
```



Date 객체

18

□ Date 객체

- 시간 정보를 담는 객체
- 현재 시간 정보

```
let now = new Date(); // 현재 날짜와 시간(시, 분, 초) 값으로 초기화된 객체 생성
```

- 학기 시작일 2017년 3월 1일의 날짜 기억

```
let startDay = new Date(2017, 2, 1); // 2017년 3월 1일(2는 3월을 뜻함)
```

□ Date 객체 활용

```
let now = new Date(); // 현재 2017년 5월 15일 저녁 8시 48분이라면  
let date = now.getDate(); // 오늘 날짜. date = 15  
let hour = now.getHours(); // 지금 시간. hour = 20
```

표 7-3: Date 객체 생성 방법

19

생성 방법	설명
<code>new Date()</code>	현재 날짜와 시간 값으로 초기화된 Date 객체를 생성함
<code>new Date(y, m, d)</code>	y: 연도, m: 월(011), d: 일(131). 월은 0부터 시작하므로 1월은 0, 12월은 11
<code>new Date(y, m, d, hour, min, sec)</code>	연, 월, 일, 시, 분, 초 값을 이용하여 Date 객체 생성. 이 외에도 밀리초(ms)까지 포함한 생성도 가능함

표 7-4: Date 객체의 주요 메서드

20

1. 날짜/시간 정보 읽기 (get 계열)

메서드	설명
getFullYear()	4자리 연도 반환 (예: 2025). getYear()는 폐기됨
getMonth()	월 반환 (0~11): 0 = 1월, 11 = 12월
getDate()	해당 월의 일(day) 반환 (1~31)
getDay()	요일 반환 (0~6): 0 = 일요일, 6 = 토요일
getHours()	시(hour) 반환 (0~23)
getMinutes()	분(minute) 반환 (0~59)
getSeconds()	초(second) 반환 (0~59)
getMilliseconds()	밀리초(ms) 반환 (0~999)
getTime()	1970년 1월 1일 00:00:00 UTC부터 현재까지 경과된 밀리초를 정수로 반환

표 7-4: Date 객체의 주요 메서드

2. 날짜/시간 설정 (set 계열)

메서드	설명
setFullYear(year)	연도를 year 값으로 설정
setMonth(month)	월을 month 값(0~11)으로 설정
setDate(date)	일을 date 값(1~31)으로 설정
setHours(hour)	시(hour)를 hour 값(0~23)으로 설정
setMinutes(minute)	분(minute)을 minute 값(0~59)으로 설정
setSeconds(second)	초(second)를 second 값(0~59)으로 설정
setMilliseconds(ms)	밀리초(ms)를 ms 값(0~999)으로 설정
setTime(t)	t 밀리초(1970년 1월 1일 이후 경과 시간)로 전체 시간 설정

표 7-4: Date 객체의 주요 메서드

22

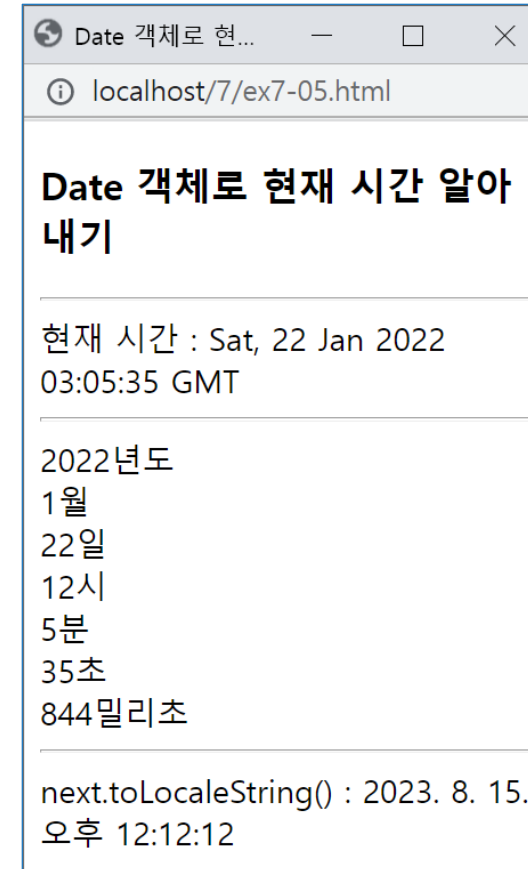
3. 출력용 메서드

메서드	설명
toUTCString()	UTC(협정 세계시) 기준 날짜와 시간 정보를 문자열로 반환
toLocaleString()	브라우저 또는 시스템 로케일 기준 날짜와 시간 정보를 문자열로 반환
toLocaleTimeString()	로케일 기준의 시간 정보만 문자열로 반환 (예: "10:12:32")

예제 7-5 Date 객체 생성 및 활용

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Date 객체로 현재 시간 알아내기</title>
</head>
<body>
<h3>Date 객체로 현재 시간 알아내기</h3>
<hr>
<script>
let now = new Date(); // 현재 시간 값을 가진 Date 객체 생성
document.write("현재 시간 : " + now.toUTCString()
    + "<br><hr>");
document.write(now.getFullYear() + "년도<br>");
document.write(now.getMonth() + 1 + "월<br>");
document.write(now.getDate() + "일<br>");
document.write(now.getHours() + "시<br>");
document.write(now.getMinutes() + "분<br>");
document.write(now.getSeconds() + "초<br>");
document.write(now.getMilliseconds() + "밀리초<br><hr>");

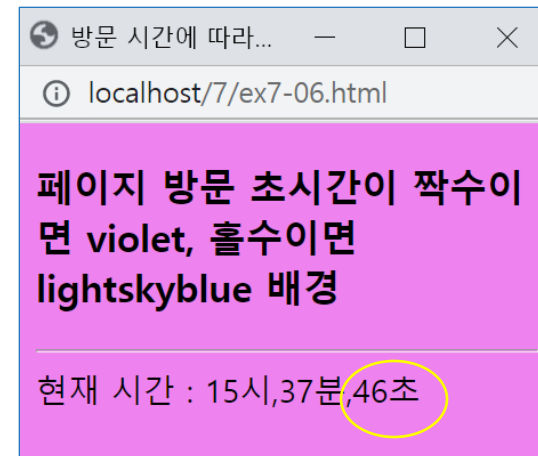
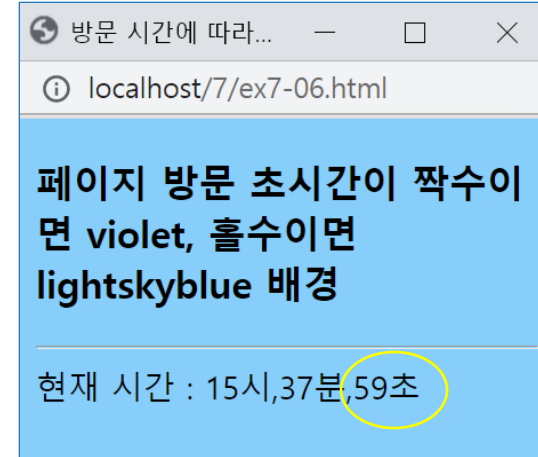
let next = new Date(2023, 7, 15, 12, 12, 12); // 7은 8월
document.write("next.toLocaleString() : "
    + next.toLocaleString() + "<br>");
</script>
</body>
</html>
```



예제 7-6 방문 시간에 따라 변하는 배경색 만들기

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>방문 시간에 따라 변하는 배경색</title>
</head>
<body>
<h3>페이지 방문 초시간이 짝수이면 violet, 홀수이면 lightskyblue 배경</h3>
<hr>
<script>
    let current = new Date(); // 현재 시간을 가진 Date 객체 생성
    if(current.getSeconds() % 2 == 0)
        document.body.style.backgroundColor = "violet";
    else
        document.body.style.backgroundColor = "lightskyblue";

    document.write("현재 시간 : ");
    document.write(current.getHours(), "시,");
    document.write(current.getMinutes(), "분,");
    document.write(current.getSeconds(), "초<br>");
</script>
</body>
</html>
```



String 객체

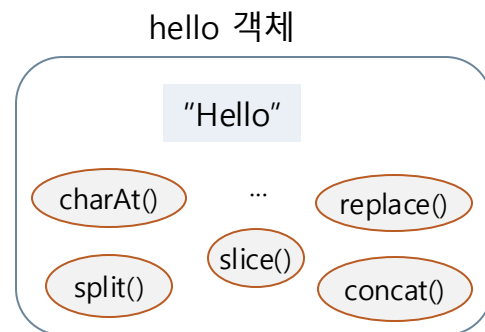
25

□ String

▣ 문자열을 담기 위한 객체

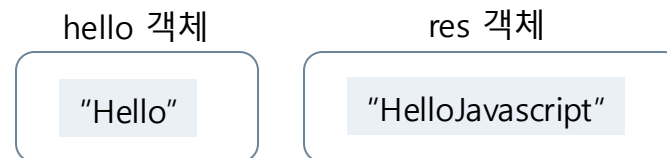
// 2 경우 모두 오른쪽 String 객체 생성

```
let hello = new String("Hello");  
let hello = "Hello";
```



▣ String 객체는 일단 생성되면 수정 불가능

```
let hello = new String("Hello");  
let res = hello.concat("Javascript");  
  
// concat() 후 hello의 문자열 변화 없음
```



String 객체의 특징

26

□ 문자열 길이

- ▣ String 객체의 length 프로퍼티 : 읽기 전용

```
let hello = new String("Hello");  
let every = "Boy and Girl";  
let m = hello.length;           // m은 5  
let n = every.length;           // n은 12
```

```
let n = "Thank you".length;      // n은 9
```

□ 문자열을 배열처럼 사용

- ▣ [] 연산자를 사용하여 각 문자 접근

```
let hello = new String(Hello");  
let c = hello[0];      // c = "H". 문자 H가 아니라 문자열 "H"
```

표 7-5: String 객체의 주요 메서드

문자 단위 메서드

메서드	설명
charAt(index)	인덱스 index에 위치한 문자를 반환 (문자열로 반환됨)
charCodeAt(index)	인덱스 index에 위치한 문자의 유니코드(16비트 정수)를 반환
fromCharCode(char)	주어진 유니코드 값(char)에 해당하는 문자로 된 문자열을 반환 (static 메서드)

항목 (자바스크립트)	문자 (char)	문자열 (string)
정의	단일 문자	하나 이상의 문자로 이루어진 텍스트
예시	'A' (길이 1)	'Apple', 'A' (길이 1 이상도 문자열)
구분 여부	별도의 자료형 없음	모든 문자는 문자열로 취급됨
타입	모두 "string"	"string"
길이	.length === 1	.length >= 1

표 7-5: String 객체의 주요 메서드

28

문자열 조작 메서드

메서드	설명
concat(s1, s2, ... sN)	현재 문자열에 s1, s2, ...를 순서대로 연결하여 새 문자열 반환
replace(strA, strB)	문자열 중 strA를 찾아 첫 번째 일치 항목만 strB로 바꾼 새 문자열 반환
slice(idxA[, idxB])	인덱스 idxA부터 idxB 바로 전까지 의 부분 문자열 반환 idxB 생략 시 끝까지
substr(idxA[, len])	인덱스 idxA에서 시작해 len 개수만큼의 문자를 잘라낸 부분 문자열 반환 len 생략 시 끝까지
substring(idxA, idxB)	인덱스 idxA에서 idxB 직전까지 의 문자열 반환. 두 값 중 작은 값을 시작점으로 자동 보정됨

표 7-5: String 객체의 주요 메서드

29

문자열 탐색 및 분할

메서드	설명
<code>indexOf(s[, idx])</code>	문자열 <code>s</code> 가 처음 나타나는 인덱스를 반환 . <code>idx</code> 지정 시 해당 위치부터 검색. 없으면 <code>-1</code> 반환
<code>split([separator[, limit]])</code>	문자열을 <code>separator</code> 기준으로 나누어 배열로 반환. <code>limit</code> 은 최대 분할 개수 제한

공백 처리 및 대소문자 변환

메서드	설명
<code>trim()</code>	문자열의 양쪽 공백 문자 (스페이스, 탭, 개행 등)를 제거한 새 문자열 반환
<code>toLowerCase()</code>	모든 문자를 소문자로 변환한 새 문자열 반환
<code>toUpperCase()</code>	모든 문자를 대문자로 변환한 새 문자열 반환

예제 7-7 String 객체의 메소드 활용

```
<!DOCTYPE html>
<html> <head> <meta charset="utf-8">
<title>String 객체의 메소드 활용</title> </head>
<body>
<h3>String 객체의 메소드 활용</h3>
<hr>
<script>
let a = new String("Boys and Girls");
let b = "!!";
document.write("a : " + a + "<br>");
document.write("b : " + b + "<br><hr>");
```

```
document.write(a.charAt(0) + "<br>");
document.write(a.concat(b, "입니다") + "<br>");
document.write(a.indexOf("s") + "<br>");
document.write(a.indexOf("And") + "<br>");
document.write(a.slice(5, 8) + "<br>");
document.write(a.substr(5, 3) + "<br>");
document.write(a.toUpperCase() + "<br>");
document.write(a.replace("and", "or") + "<br>");
document.write("  kitae  ".trim() + "<br><hr>");
```

```
let sub = a.split(" ");
document.write("a를 빈칸으로 분리<br>");
for(let i=0; i<sub.length; i++)
    document.write("sub" + i + "=" + sub[i] + "<br>");
```

```
document.write("<hr>String 메소드를 실행 후 a와 b 변함 없음<br>");
document.write("a : " + a + "<br>");
document.write("b : " + b + "<br>");
</script>
</body> </html>
```

a.charAt(0)

a.indexOf("s")

a.slice(5,8)

String 객체의 메소드 활용

a : Boys and Girls
b : !!

B
Boys and Girls!!입니다
3
-1
and
and
BOYS AND GIRLS
Boys or Girls
kitae

a를 빈칸으로 분리
sub0=Boys
sub1=and
sub2=Girls

String 메소드를 실행 후 a와 b 변함 없음
a : Boys and Girls
b : !!

Math 객체

31

□ Math

- ▣ 수학 계산을 위한 프로퍼티와 메소드 제공
- ▣ new Math()로 객체 생성하지 않고 사용

```
let sq = Math.sqrt(4);    // 4의 제곱근을 구하면 2
let area = Math.PI*2*2;    // 반지름이 2인 원의 면적
```

▣ 난수 발생

- Math.random() : 0~1보다 작은 랜덤한 실수 리턴
- Math.floor(m)은 m의 소수점 이하를 제거한 정수 리턴

```
// 0~99까지 랜덤한 정수를 10개 만드는 코드
for(let i=0; i<10; i++) {
  let m = Math.random()*100; // m은 0~99.999... 보다 작은 실수 난수
  let n = Math.floor(m);      // m에서 소수점 이하를 제거한 정수(0~99사이)
  document.write(n + " ");
}
```

표 7-6: Math 객체의 주요 메서드 (x와 y는 정수 또는 실수 가능)

32

메서드	설명
abs(x)	x의 절댓값을 반환
sin(x)	x의 사인 값을 반환 (단위: 라디안)
exp(x)	e^x 값을 반환 (자연상수 e의 거듭제곱)
pow(x, y)	x의 y 제곱 값을 반환 (x^y)
random()	0 이상 1 미만의 난수 (무작위 실수)를 반환
floor(x)	x보다 작지 않은 가장 큰 정수 (내림)를 반환예: Math.floor(3.9) → 3
round(x)	x를 반올림 한 정수를 반환예: Math.round(3.7) → 4, Math.round(3.2) → 3
sqrt(x)	x의 제곱근 을 반환

표 7-7: Math 객체의 주요 프로퍼티

33

프로퍼티	설명
Math.E	자연상수 e (약 2.718...)
Math.PI	원주율 π (약 3.141592...)

예제 7-8 Math를 이용한 구구단 연습

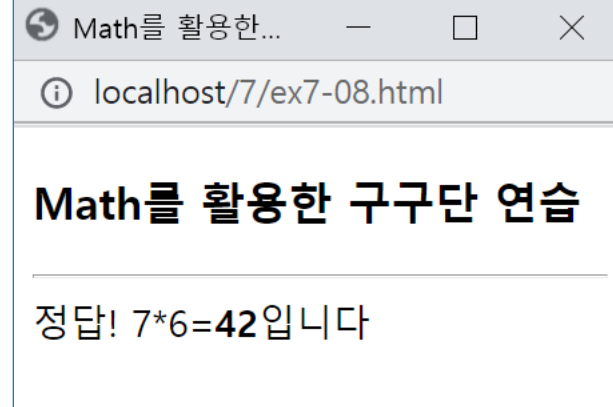
```
<!DOCTYPE html>
<html>
<head> <meta charset="utf-8">
<title>Math를 활용한 구구단 연습</title>
<script>
    function randomInt() { // 1~9의 십진 난수 리턴
        return Math.floor(Math.random()*9) + 1;
    }
</script>
</head>
<body>
<h3>Math를 활용한 구구단 연습</h3>
<hr>
<script>
    // 구구단 문제 생성
    let ques = randomInt() + "*" + randomInt();
    // 사용자로부터 답 입력
    let user = prompt(ques + " 값은 얼마입니까?", 0);
    if(user == null) { // 취소 버튼이 클릭된 경우
        document.write("구구단 연습을 종료합니다");
    }
    else {
        let ans = eval(ques); // 구구단 정답 계산
        if(ans == user) // 정답과 사용자 입력 비교
            document.write("정답! ");
        else
            document.write("아니오! ");
        document.write(ques + "=" + "<strong>" + ans
            + "</strong> 입니다<br>");
    }
</script>
</body> </html>
```

localhost 내용:

7*6 값은 얼마입니까?

확인

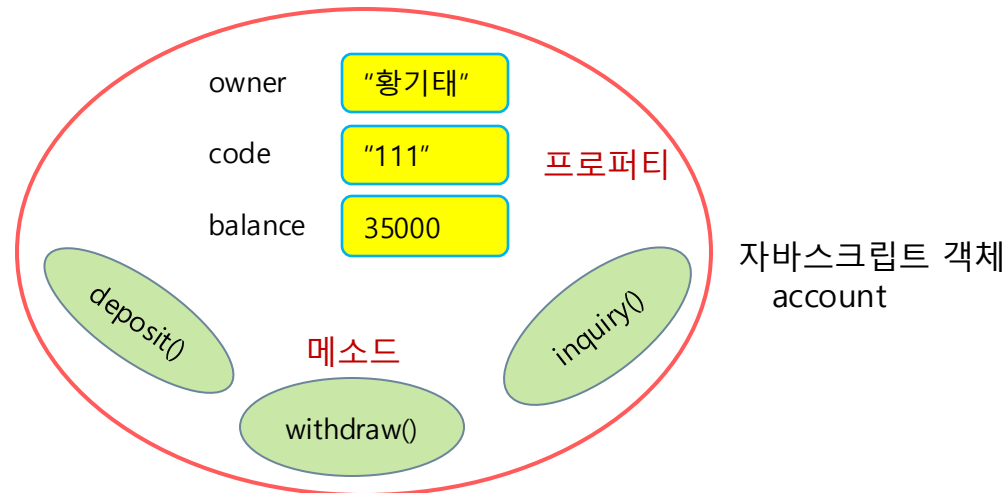
취소



사용자 객체 만들기

35

- 사용자가 새로운 타입의 객체 작성 가능 : 3 가지 방법
 - ▣ 1. 직접 객체 만들기
 - new Object() 이용
 - 리터럴 표기법 이용
 - ▣ 2. 객체의 틀(프로토타입)을 만들고 객체 생성하기
- 샘플
 - ▣ 은행 계좌를 표현하는 account 객체



new Object()로 객체 만들기

36

□ 과정

- ▣ 1. new Object()로 빈 객체 생성
- ▣ 2. 빈 객체에 프로퍼티 추가
 - 새로운 프로퍼티 추가(프로퍼티 이름과 초기값 지정)
- ▣ 3. 빈 객체에 메소드 추가
 - 메소드로 사용할 함수 미리 작성
 - 새 메소드 추가(메소드 이름에 함수 지정)

```
let account = new Object();  
account.owner = "황기태"; // 계좌 주인 프로퍼티 생성 및 초기화  
account.code = "111"; // 코드 프로퍼티 생성 및 초기화  
account.balance = 35000; // 잔액 프로퍼티 생성 및 초기화  
account.inquiry = inquiry; // 메소드 작성  
account.deposit = deposit; // 메소드 작성  
account.withdraw = withdraw; // 메소드 작성
```

예제 7-9 new Object()로 계좌를 표현하는 account 객체 만들기

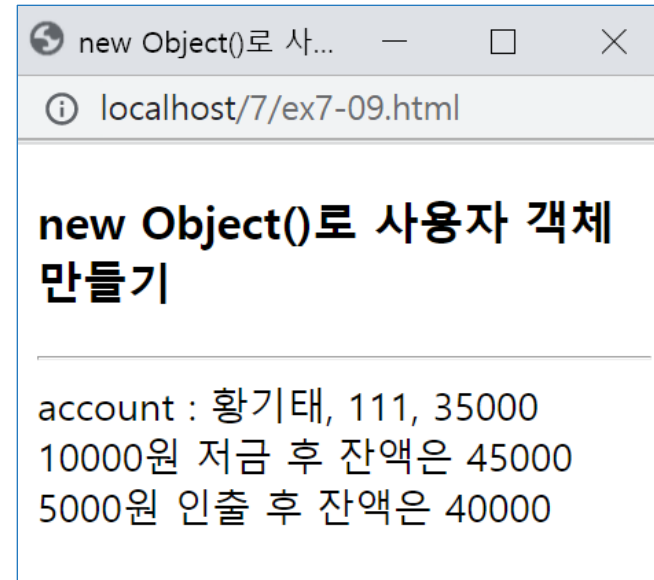
```
<!DOCTYPE html>
<html><head><meta charset="utf-8"> <title>new Object()로 사용자 객체 만들기</title>
<script>
  //메소드로 사용할 3 개의 함수 작성
  function inquiry() { return this.balance; } // 잔금 조회
  function deposit(money) { this.balance += money; } // money 만큼 저금
  function withdraw(money) { // 예금 인출, money는 인출하고자 하는 액수
    // money가 balance보다 작다고 가정

    this.balance -= money;
    return money;
  }

  // 사용자 객체 만들기
  let account = new Object();
  account.owner = "황기태"; // 계좌 주인 프로퍼티 생성 및 초기화
  account.code = "111"; // 코드 프로퍼티 생성 및 초기화
  account.balance = 35000; // 잔액 프로퍼티 생성 및 초기화
  account.inquiry = inquiry; // 메소드 작성
  account.deposit = deposit; // 메소드 작성
  account.withdraw = withdraw; // 메소드 작성
</script></head>
<body>
<h3>new Object()로 사용자 객체 만들기</h3>
<hr>
<script>
  // 객체 활용
  document.write("account : ");
  document.write(account.owner + ", ");
  document.write(account.code + ", ");
  document.write(account.balance + "<br>");

  account.deposit(10000); // 10000원 저금
  document.write("10000원 저금 후 잔액은 " + account.inquiry() + "<br>");
  account.withdraw(5000); // 5000원 인출
  document.write("5000원 인출 후 잔액은 " + account.inquiry() + "<br>");
</script>
</body></html>
```

this.balance는 객체의
balance 프로퍼티



리터럴 표기법으로 만들기

38

□ 과정

- ▣ 중괄호를 이용하여 객체의 프로퍼티와 메소드 지정
- ▣ 가장 많이 사용하는 방법

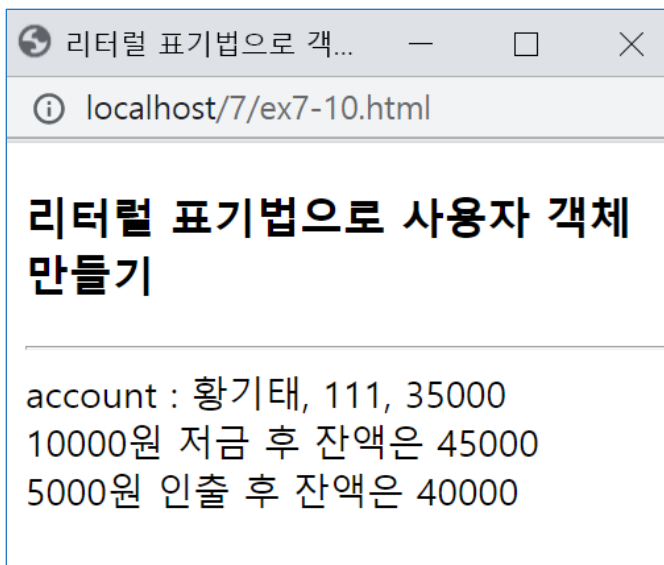
```
let account = {  
  // 프로퍼티 생성 및 초기화  
  owner : "황기태",    // 계좌 주인 프로퍼티 추가  
  code : "111",        // 계좌 코드 프로퍼티 추가  
  balance : 35000,     // 잔액 프로퍼티 추가  
  
  // 메소드 작성  
  inquiry : function () { return this.balance; }, // 잔금 조회  
  deposit : function(money) { this.balance += money; }, // 저금. money 만큼 저금  
  withdraw : function (money) { // 예금 인출, money는 인출하고자 하는 액수  
    // money가 balance보다 작다고 가정  
    this.balance -= money;  
    return money;  
  }  
};
```

예제 7-10 리터럴 표기법으로 계좌를 표현하는 account 객체 만들기

```
<!DOCTYPE html>
<html>
<head> <meta charset="utf-8">
<title>리터럴 표기법으로 사용자 객체 만들기</title>
<script>
//사용자 객체 만들기
let account = {
  // 프로퍼티 생성 및 초기화
  owner : "황기태", // 계좌 주인
  code : "111", // 계좌 코드
  balance : 35000, // 잔액 프로퍼티

  // 메소드 작성
  inquiry : function () { return this.balance; }, // 잔금 조회
  deposit : function(money) { this.balance += money; }, // 저금. money 만큼 저금
  withdraw : function (money) { // 예금 인출, money는 인출하고자 하는 액수
    // money가 balance보다 작다고 가정
    this.balance -= money;
    return money;
  }
};
</script></head>
<body>
<h3>리터럴 표기법으로 사용자 객체 만들기</h3>
<hr>
<script>
  document.write("account : ");
  document.write(account.owner + ", ");
  document.write(account.code + ", ");
  document.write(account.balance + "<br>");

  account.deposit(10000); // 10000원 저금
  document.write("10000원 저금 후 잔액은 " + account.inquiry() + "<br>");
  account.withdraw(5000); // 5000원 인출
  document.write("5000원 인출 후 잔액은 " + account.inquiry() + "<br>");
</script>
</body></html>
```



프로토타입

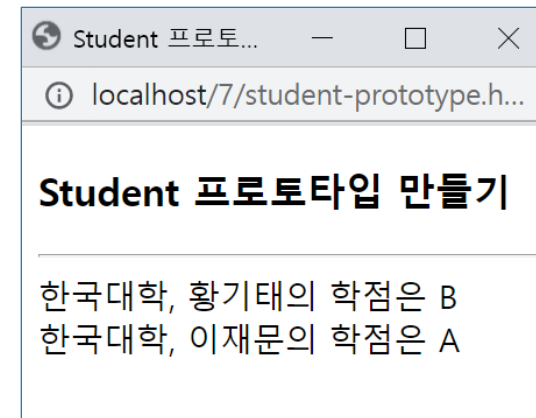
40

- 프로토타입(prototype)이란?
 - ▣ 객체의 모양을 가진 틀
 - 붕어빵은 객체이고, 붕어빵을 찍어내는 틀은 프로토타입
 - ▣ C++, Java에서는 프로토타입을 클래스라고 부름
 - ▣ Array, Date, String : 자바스크립트에서 제공하는 프로토타입
 - ▣ 객체 생성시 'new 프로토타입' 이용
 - let week = **new Array**(7); // Array는 프로토타입임
 - let hello = **new String**("hello"); // String은 프로토타입임

프로토타입 만드는 사례 : Student 프로토타입

- 프로토타입은 함수로 만든다
 - 프로토타입 함수를 생성자 함수라고도 함

```
// 프로토타입 Student 작성
function Student(name, score) {
  this.univ = "한국대학"; // this.univ을 이용하여 univ 프로퍼티 작성
  this.name = name; // this.name을 이용하여 name 프로퍼티 작성
  this.score = score; // this.score를 이용하여 score 프로퍼티 작성
  this.getGrade = function () { // getGrade() 메소드 작성
    if(this.score > 80) return "A";
    else if(this.score > 60) return "B";
    else return "F";
  }
}
```



- new 연산자로 객체를 생성한다

```
let kitae = new Student("황기태", 75); // Student 객체 생성
let jaemoon = new Student("이재문", 93); // Student 객체 생성
document.write(kitae.univ + ", " + kitae.name + "의 학점은 " + kitae.getGrade() + "<br>");
document.write(jaemoon.univ + ", " + jaemoon.name + "의 학점은 " + jaemoon.getGrade() + "<br>")
```

예제 7-11 프로토타입으로 객체 만들기

42

```
<!DOCTYPE html>
<html><head><meta charset="utf-8"> <title>프로토타입으로 객체 만들기</title>
<script>
  // 프로토타입 만들기 : 생성자 함수 작성
  function Account(owner, code, balance) {
    // 프로퍼티 만들기
    this.owner = owner; // 계좌 주인 프로퍼티 만들기
    this.code = code; // 계좌 코드 프로퍼티 만들기
    this.balance = balance; // 잔액 프로퍼티 만들기

    // 메소드 만들기
    this.inquiry = function () { return this.balance; }
    this.deposit = function (money) { this.balance += money; }
    this.withdraw = function (money) { // 예금 인출, money는 인출하는 액수
      // money가 balance보다 작다고 가정
      this.balance -= money;
      return money;
    }
  }
</script></head>
<body>
<h3>Account 프로토타입 만들기</h3>
<hr>
<script>
  // new 연산자 이용하여 계좌 객체 생성
  let account = new Account("황기태", "111", 35000);

  // 객체 활용
  document.write("account : ");
  document.write(account.owner + ", ");
  document.write(account.code + ", ");
  document.write(account.balance + "<br>");

  account.deposit(10000); // 10000원 저금
  document.write("10000원 저금 후 잔액은 " + account.inquiry() + "<br>");
  account.withdraw(5000); // 5000원 인출
  document.write("5000원 인출 후 잔액은 " + account.inquiry() + "<br>");
</script>
</body></html>
```

