JavaScript

◆ 객체

정수아

Contents

01 객체

02 내장 객체

01 객체

객체

❖ **현실 세계는 객체들의 집합**

- 사람, 책상, 자동차, TV 등
- 객체는 자신만의 고유한 구성 속성을 가진다.
  - 자동차 : <색상:오렌지, 배기량:3000CC, 제조사:한성, 번호:1234>
  - 사람 : <이름:정수아, 나이:20, 성별:여, 주소:서울>
  - 은행 계좌 : <소유자:정수아, 계좌번호:111, 잔액:35000원>

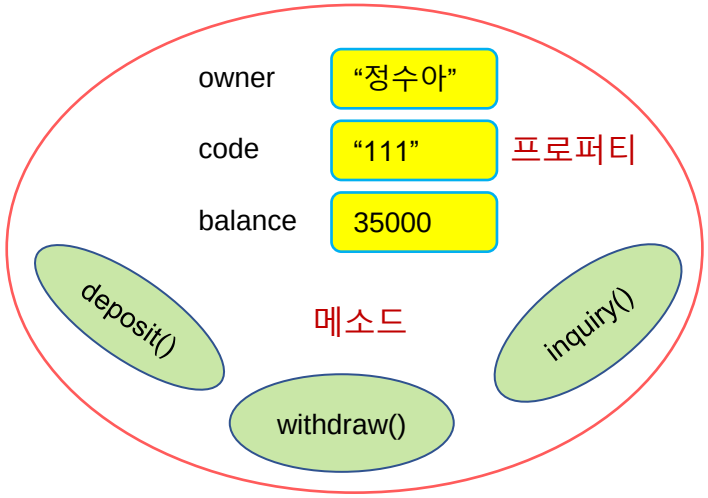
자동차 객체(car)

은행 계좌(account)

자바스크립트 객체

❖ 자바스크립트 객체 구성

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



자바스크립트 객체 account

```
var account = {
   owner      :  " 정수아",
   code       : "111",
   balance    : 35000,
   deposit    : function() { … },
   withdraw   : function() { … },
   inquiry    : function() { … }
};
```

account 객체를 만드는 자바스크립트 코드

객체

❖ 배열과 차이점

- 배열은 원소에 접근 할 때, 숫자를 사용
- 객체는 원소에 접근 할 때, 문자열을 사용
  - 문자열은 키(key) 또는 프로퍼티(property)라 부름
  - 문자열이 가리키는 원소를 값이라 함

- 배열은 여러 개체를 표현할 때 사용
  - 예) 여러 동물의 이름이 나열된 배열

- 객체는 다양한 특성이나 속성이 있는 하나의 개체를 표현할 때 사용
  - 예) 한 동물과 관련된 여러 정보를 저장

객체 생성

❖ **한 가지 동물에 관한 여러 정보를 저장**

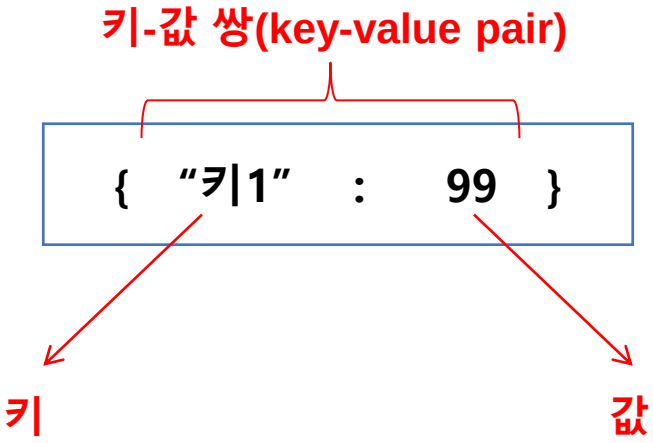
- 다리가 두 개인 고양이 '야옹이'에 대한 정보가 저장된 객체
  - 변수 생성
  - 세 개의 키-값 쌍이 들어있는 객체를 할당
  - 객체를 생성 시에는 중괄호({ }) 사용

```
var 고양이 = {
      "다리" : 2,
      "이름" : "야옹이",
      "색깔" : "얼룩무늬"
};
```

객체 리터럴(object literal)

- 리터럴 표기법
  - 중괄호를 이용하여 프로퍼티와 메서드를 한 번에 작성

객체 생성



구분	설명
키	• 문자열만 가능
값	• 모든 타입의 자료형 사용 가능
키-값 쌍 구분	• 쉼표로 구분 • 마지막 키-값 쌍에는 쉼표를 붙이지 않음

따옴표가 없는 키

❖ **각 키의 따옴표 생략**
- 자바스크립트는 키가 문자열이어야 한다는 것을 알고 있음
- 따라서 따옴표는 생략 가능

❖ **따옴표 사용 시**
- 키 이름에 공백 포함 가능
- 따옴표 생략 시, 불가능

```
var 고양이 = {
        다리 : 2,
        이름 : "야옹이",
        색깔 : "얼룩무늬"
};
```

객체 안의 값 접근

❖ 따옴표를 사용하여 접근하는 경우

- 배열처럼 대괄호를 사용해서 접근

```
>> console.log(고양이["이름"]);
<< "야옹이"
```

❖ 따옴표를 생략하고 접근하는 경우(점 표기법)

```
>> console.log(고양이.이름);
<< "야옹이"
```

❖ 객체내의 모든 키 목록

- Object.keys()

```
>> var 강아지 = {이름: "멍멍이", 나이: 6, 색깔: "흰색", 울음소리: "멍멍"};
>> var 고양이 = {이름: "야옹이", 나이: 8, 색깔: "얼룩무늬"};
>> console.log(Object.keys(강아지));
<< ["이름", "나이", "색깔", "울음소리"]
```

객체에 값 추가

❖ 빈 객체

```
var 객체 = { };
```

❖ 객체에 원소 추가

```
var 고양이 = {};
고양이[“다리”] = 2;
고양이[“이름”] = “야옹이”;
고양이[“색깔”] = “얼룩무늬”;

console.log(고양이);      //  {다리: 2, 이름: "야옹이", 색깔: "얼룩무늬"}
```

❖ 객체의 키는 순서가 없음

- 따라서, 키의 순서가 중요할 때는 객체를 사용하지 않음
- 배열에는 순서가 있음

객체에 값 추가

❖ 점 표기법으로 키 추가하기

```
var 고양이 = {};

고양이.다리 = 2;
고양이.이름 = "야옹이";
고양이.색깔 = "얼룩무늬";

// {다리: 2, 이름: "야옹이", 색깔: "얼룩무늬"}
document.write(JSON.stringify(고양이));
```

❖ JSON.stringify()

- 자바스크립트의 값을 JSON 문자열로 변환

객체와 배열 결합하기

❖ 객체/배열의 값으로 객체/배열을 사용

- 예) 공룡 객체로 만든 배열

```
var 공룡 = [
      { 이름 : "티라노사우루스 렉스", 연대 : "백악기 후기"},
      { 이름 : "스테고사우루스", 연대 : "쥐라기 후기"},
      { 이름 : "플라테오사우루스", 연대 : "트라이아스기"}
];
```

- 공룡에 대한 정보를 얻을 때

```
>> 공룡[0];
<< {이름: "티라노사우루스 렉스", 연대: "백악기 후기"}
```

대괄호 사용

```
>> 공룡[0]["이름"];
<< "티라노사우루스 렉스"
```

점표기법 사용

```
>> 공룡[1].연대;
<< "쥐라기 후기"
```

배열과 객체 결합하기

❖ 친구 배열

- 3개의 객체 생성
- 각 객체에 '이름, 나이, 행운의숫자'라는 키 삽입
- 이름(문자열), 나이(숫자), 행운의숫자(숫자 배열)

```
var 민지 = { 이름 : "민지", 나이 : 11, 행운의숫자 : [2, 4, 8, 16] };
var 지훈 = { 이름 : "지훈", 나이 : 15, 행운의숫자 : [3, 9, 40] };
var 서연 = { 이름 : "서연", 나이 : 19, 행운의숫자 : [1, 2, 30] };
```

- 친구 배열 생성
  - 각 원소는 해당하는 친구 객체를 참조

```
var 친구 = [민지, 지훈, 서연];
```

배열과 객체 결합하기

❖ 정보 가져오기

- 배열의 인덱스를 사용

```
console.log(친구[1]);
// 결과 : {"이름":"지훈","나이":15,"행운의숫자":[3,9,40]}
```

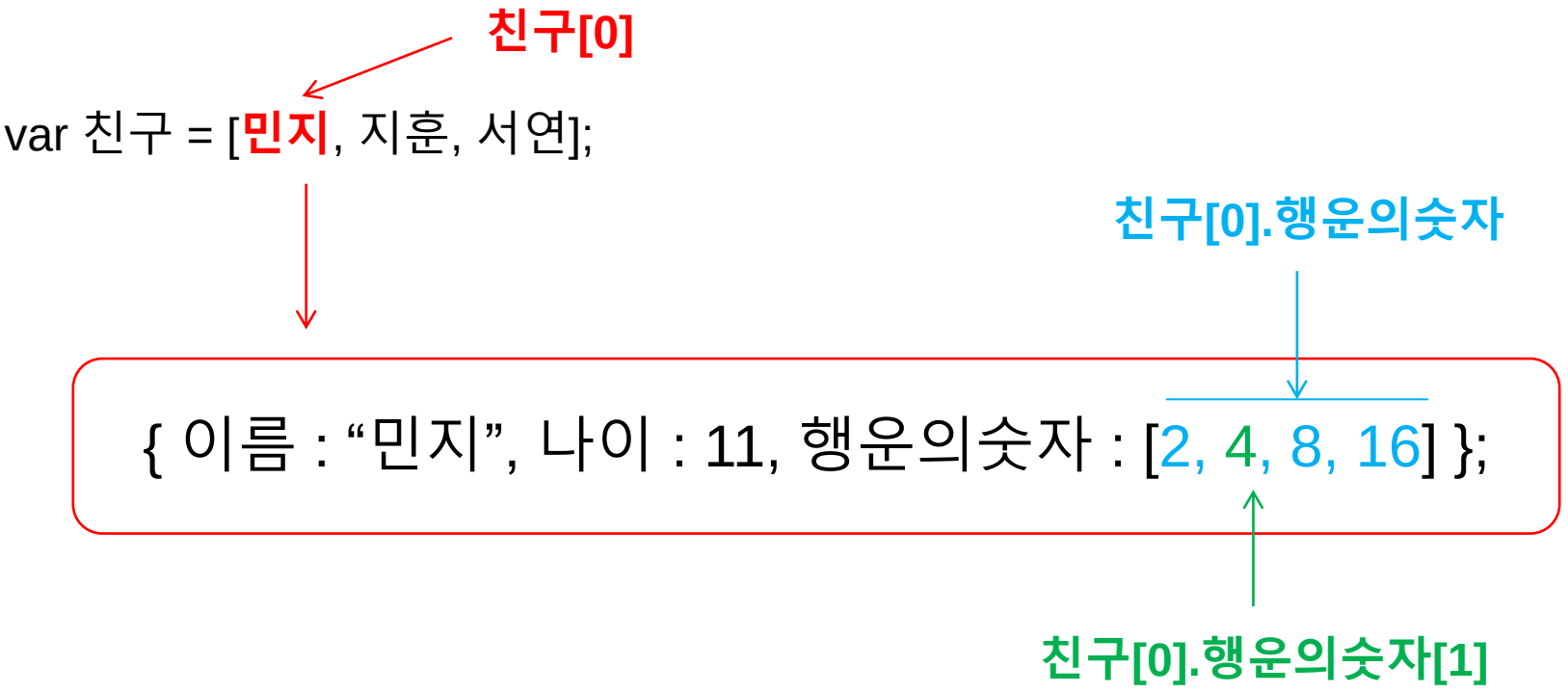
- 객체의 값 얻기

```
console.log(친구[2].이름);
// 결과 : "서연"
```

- 내부 배열의 값 얻기

```
console.log(친구[0].행운의숫자[1]);
// 결과 : 4
```

배열과 객체 결합하기



객체 활용

❖ 빌려준 돈 기록하기

- 객체를 활용하여 문자열과 값을 연결
- 키(친구 이름), 값(빌린 금액)

```
var 빌려준돈 = {};

빌려준돈["지훈"] = 5000;
빌려준돈["민지"] = 7000;

console.log(빌려준돈["지훈"]);     // 5000
```

```
console.log(빌려준돈["진영"]);     // undefined
```

설정된 값이 없으므로 undefined

객체 활용

❖ 빌려준 돈 기록하기

- 지훈이가 3000원을 더 빌림

```
빌려준돈["지훈"] += 3000;
console.log(빌려준돈["지훈"]);

// 결과 : 8000
```

- 각 친구가 빌린 돈 확인(전체 객체 보기)

```
console.log(빌려준돈);

// 결과 : {지훈: 8000, 민지: 7000}
```

객체 활용

❖ 영화 관련 정보 저장하기

- 키(영화제목), 값(영화 정보를 담은 객체)

```
var 영화 = {
        "니모를 찾아서" : {
                개봉연도 : 2003,
                상영시간 : 100,
                출연진 : ["앨버트 브룩스", "엘런 드제너러스", "알렉산더 굴드"],
                형식 : "DVD"
        },
        "스타워즈에피소드6-제다이의귀환" : {
                개봉연도 : 1983,
                상영시간 : 134,
                출연진 : ["마크 해밀", "해리슨 포드", "캐리 피셔"],
                형식 : "DVD"
        },
        "해리포터와 불의 잔" : {
                개봉연도 : 2005,
                상영시간 : 157,
                출연진 : ["다니엘 레드클리프", "엠마 왓슨", "루퍼트 그린트"],
                형식 : "블루레이"
        }
};
```

객체 활용

❖ 영화 관련 정보 저장하기

- 영화 정보 얻기

```
var 니모를찾아서 = 영화[“니모를 찾아서”];
console.log(니모를찾아서.상영시간);
// 결과 : 100
```

```
console.log(니모를찾아서.형식);
// 결과 : "DVD"
```

```
console.log(니모를찾아서.출연진);
// 결과 : ["앨버트 브룩스", "엘런 드제너러스", "알렉산더 굴드"]
```

객체 활용

❖ 영화 관련 정보 저장하기

- 새로운 영화 추가

```
var 카 = {
        개봉연도 : 2006,
        상영시간 : 117,
        출연진 : [“오웬 윌슨”, “보니 헌트”, “폴 뉴먼”],
        형식 : “블루레이”
};

// 영화 객체에 "카"라는 키를 생성 후, 카(객체) 저장
영화[“카”] = 카;
```

객체 활용

❖ 영화 관련 정보 저장하기

- 모든 영화 제목 얻기

```
console.log(Object.keys(영화));

// 결과 :
// 니모를 찾아서,스타워즈에피소드6-제다이의 귀환,해리포터와 불의 잔, 카
```

const로 선언된 객체 내부 값 변경하기

❖ const로 선언된 객체의 값 변경하기

```
const dog = {
    name: "바둑이",
    legs: 2,
    color: "brown",
};

console.log(dog.color);

dog.color = "white";

console.log(dog.color);
```

- 실행 결과

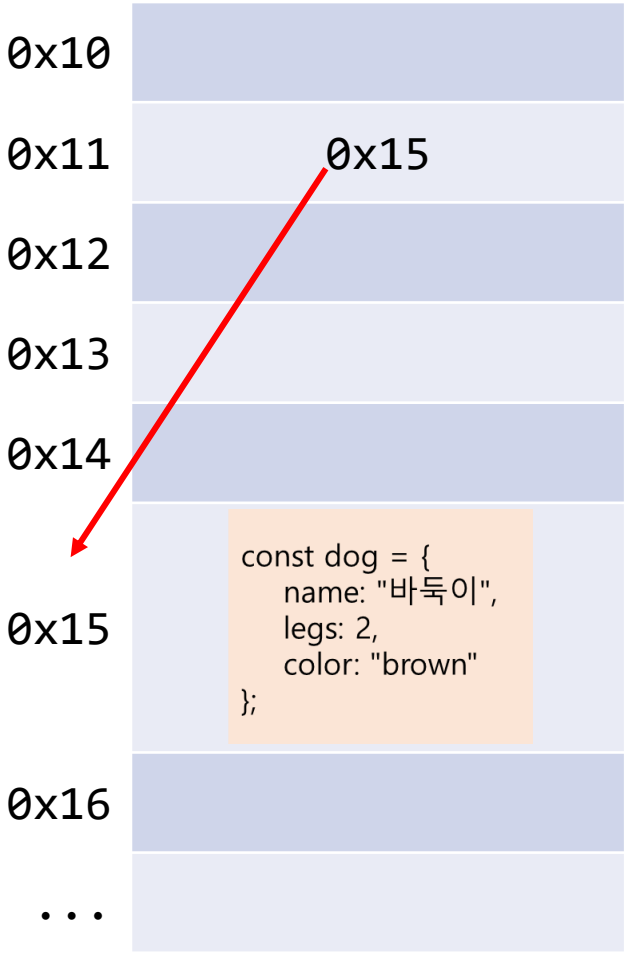
```
brown
white
```

const로 선언된 객체 내부 값 변경하기

❖ 값이 변경되는 이유

```
const dog = {
    name: "바둑이",
    legs: 2,
    color: "brown"
};
```

메모리 주소만 참조



객체 축약하기

❖ 객체 생성문 축약

- key와 변수의 이름이 같은 경우 축약 가능

```
let width = 100;
let height = 100;

let area = {
  width: width,
  height: height
};

console.log(area);
```

```
let width = 100;
let height = 100;

let area = { width, height };


console.log(area);
```

- 결과

```
{ width: 100, height: 100 }
```

객체 안에 함수

❖ 객체 안에 함수 넣기

```
const dog = {
  name: "바둑이",
  legs: 2,
  color: "brown",
  speak: function () {
      console.log("멍멍");
  }
};

// dog 객체 안 speak() 함수 호출
dog.speak();
```

- 실행 결과

```
멍멍
```

객체 안에 함수

❖ 객체 안에 함수 넣기

```
const dog = {
  name: "바둑이",
  legs: 2,
  color: "brown",
  speak: function () {
      console.log("멍멍");
      console.log(this.name);
  }
};

// dog 객체 안 speak() 함수 호출
dog.speak();
```

- 실행 결과

```
멍멍
바둑이
```

객체 안에 함수

❖ 객체 안에 함수 넣기 - 화살표 함수

```
const dog = {
  name: "바둑이",
  legs: 2,
  color: "brown",
  speak: () => {
      console.log("멍멍");
      console.log(this.name);
  }
};

// dog 객체 안 speak() 함수 호출
dog.speak();
```

- 실행 결과

```
멍멍
undefined
```

생성자 함수를 이용하여 객체 생성하기

❖ 생성자 함수

- 유사한 객체를 여러 개 생성 할 수 있음

- 특징
  - 함수 이름의 첫 글자는 대문자로 시작
  - new 연산자를 붙여서 실행

- 객체 생성 예시

```
function Animals(name, legs) {
    this.name = name;
    this.legs = legs;
}

let dog = new Animals("바둑이", 4);
let bird = new Animals("짹짹이", 2);
```

[실습] 생성자 함수

```
function Animals(name, legs) {
    this.name = name;
    this.legs = legs;
}

let dog = new Animals("바둑이", 4);
let bird = new Animals("짹짹이", 2);

console.log(dog.name);
console.log(bird.name);
```

❖ 실행 결과

```
바둑이
짹짹이
```

02

내장 객체

Date 객체

❖ Date 객체

- 시간 정보를 담는 객체
- 현재 시간 정보

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

- 예) 2017년 4월 1일의 날짜 정보를 기록하는 객체 생성

```
var startDay = new Date(2017, 3, 1);
```

- Date 객체에서 월(month) 값은 0부터 시작
  - 0은 1월, 11은 12월

Date 객체

객체 생성 방법	설명	
new Date()	현재 날짜와 시간 값으로 초기화된 객체 생성	
new Date(y,m,d)	y	년
	m	월(0~11)
	d	일(1~31)
	정보를 가진 객체 생성	
new Date(y,m,d,hour,min,sec)	y	년
	m	월(0~11)
	d	일(1~31)
	hour	시
	min	분
	sec	초
	정보를 가진 객체 생성	

Date 객체

객체 생성 방법	설명
getFullYear()	4자리 연도 리턴(2021)
getMonth()	0~11 사이의 정수 리턴
getDate()	한 달 내의 날짜 리턴(1~31)
getDay()	한 주 내 요일을 정수로 리턴 일요일=0, 월요일=1, 토요일=6
getHours()	0~23 사이의 정수 시간 리턴
getMinutes()	0~59 사이의 정수 분 리턴
getSeconds()	0~59 사이의 정수 초 리턴
getMiliseconds()	0~999 사이의 정수 밀리초 리턴
toLocaleString()	객체에 든 시간 정보를 로컬 표현의 문자열로 리턴
toUTCString()	객체에 든 시간 정보를 UTC 문자열로 리턴

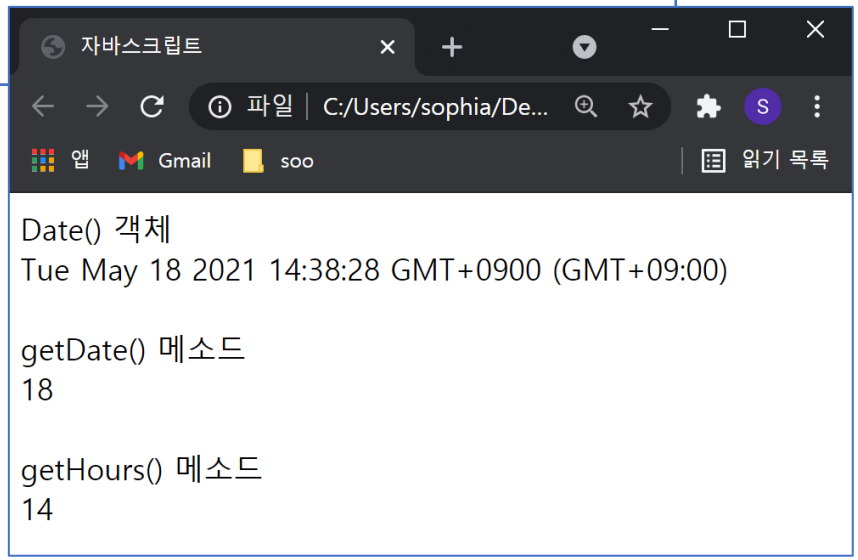
Date 객체

❖ Date 객체의 메소드

```
// 현재 날짜
var now = new Date();

// 날짜(date)
var date = now.getDate();

// 시간(hour)
var hour = now.getHours();
```



[실습] Date 객체

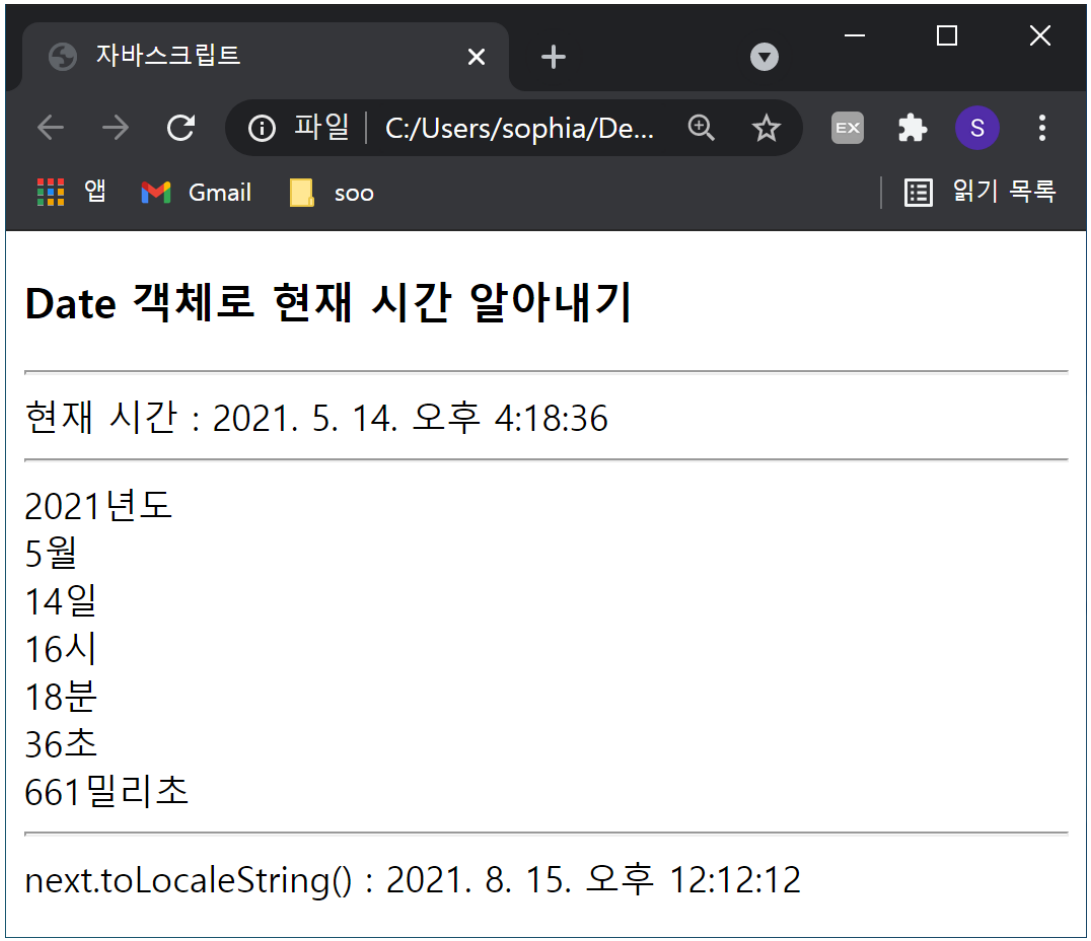
```
<body>
   <h3>Date 객체로 현재 시간 알아내기</h3>
   <hr>

   <script>
      var now = new Date(); // 현재 시간 값을 가진 Date 객체 생성
      document.write("현재 시간 : " + now.toLocaleString() + "<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>");

      var next = new Date(2021, 7, 15, 12, 12, 12); // 7은 8월
      document.write("next.toLocaleString() : "+ next.toLocaleString() + "<br>");
   </script>
</body>
```

[실습] Date 객체

❖ 실행 결과



Math 객체

❖ Math 객체

- 수학 계산을 위한 객체
- new Math()로 객체를 생성하지 않음

```
Math.프로퍼티   또는   Math.메소드()
```

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

Math 객체

❖난수 발생

- Math.random()
  - 1보다 작은 0~1 사이의 실수를 리턴
- Math.floor(m)
  - m의 소수점 이하를 제거한 정수 리턴

- 예) 0~100 보다 작은 정수 10개를 랜덤하게 생성

```
for(i=0; i<10; i++) {
   var m = Math.random()*100;  // m은 0~99.99... 보다 작은 실수
   var n = Math.floor(m);      // m은 정수(0~99사이)

   document.write(n + " ");
}
```


THANK YOU