11장
자바스크립트 객체 다루기

11.1 객체란

- 객체
  - 키와 값으로 구성된 속성들의 집합을 의미하는 자료형
  - {}를 이용해 생성
  - 키는 문자열로 작성(키에 공백이 들어갈 경우에는 반드시 따옴표 사용) ""
  - 모든 자료형의 데이터를 값으로 가질 수 있음
- 빈 객체 : 속성이 한 개도 없는 객체

```
const person = {};
```

```
const person = {"Hong Gildong"};
```

11.1 객체란

- 객체 사용 예시

```
const person = {
  name:["Hong", "Gildong"],
  age:20,
  isAdult:true
};
```

```
const person = {
  "phone number":"010-000-0000"
};
```

```
const person = {
  name:{
    firstName:"Gildong",
    lastName:"Hong"
  },
  age:20,
  isAdult:true,
  printInfo:function(){
    console.log('printInfo');
  }
};
```

11.2 객체 속성 다루기

1. 객체 속성에 접근하기

- 대괄호 연산자로 접근하기
  - []를 사용해 객체의 속성에 접근하는 방법
  - 배열에서도 사용 가능
  - 객체의 속성에 접근하려면 객체명 뒤에 []를 붙이고 [] 안에 키를 넣음(이때 키는 반드시 문자열 형태로 작성)
- 마침표 연산자로 접근하기
  - .를 이용해 객체 속성에 접근하는 방법
  - 객체 속성에 접근하려면 접근할 객체명과 객체 속성의 키를 마침표 연산자로 연결(이때 키를 큰따옴표나 작은따옴표로 감싸면 오류 발생)
  - 식별자에 공백이 있다면 마침표 연산자는 사용할 수 없음

11.2 객체 속성 다루기

1. 객체 속성에 접근하기 예제

- 대괄호 연산자로 접근하기

```
const person = {
  name:"Hong Gildong",
  age:20
};
console.log(person["name"]); // Hong GilDong
console.log(person["age"]);  // 20
```

올바른 예

```
const person = {
  name:"Hong Gildong"
};
console.log(person[name]); // ReferenceError: name is not defined
```

잘못된 예

11.2 객체 속성 다루기

2. 객체 속성 값 변경하기

- 객체로 정의된 값을 바꾸고 싶다면 키로 속성에 접근해서 값을 재할당

3. 객체 속성 동적으로 추가하기

- 속성을 동적으로 추가 : 이미 만들어진 객체에 나중에 속성을 추가하는 것
- 객체 속성에 값을 할당해 접근하면 해당 속성이 존재하는지 확인하고, 없는 속성이면 해당 키와 값으로 구성된 새로운 속성을 객체에 추가

4. 객체 속성 동적으로 삭제하기

- 객체 속성에 접근할 때 앞에 delete 키워드를 명시하면 해당 속성을 삭제

11.2 객체 속성 다루기

2. 객체 속성 값 변경하기

```
const person = {
  name:"Hong Gildong"
};
person.name = "Kim"; //  또는 person["name"] = "Kim";
console.log(person.name); // Kim
```

3. 객체 속성 동적으로 추가하기

```
const person = {};
console.log(person); // {}
person.name = "Hong Gildong";
console.log(person); // { name: 'Hong Gildong' }
```

11.2 객체 속성 다루기

11/02/dynamic_object.js

```
const person = {};
person.name = {
  firstName:"GilDong",
  lastName:"Hong"
};
person.likes = ["apple", "samsung"];
person.printHello = function(){
  return "hello";
}
```

4. 객체 속성 동적으로 삭제하기

```
const person = {
  name:"Hong Gildong"
};
delete person.name; // 또는 delete person["name"]
console.log(person); // {} 출력
```

11.2 객체 속성 다루기

11/02/primitive_copy.js

```
let num = 10;
let copyNum = num;
num = 20; // 변수 num을 재할당
console.log(num); // 20
console.log(copyNum); // 10
```

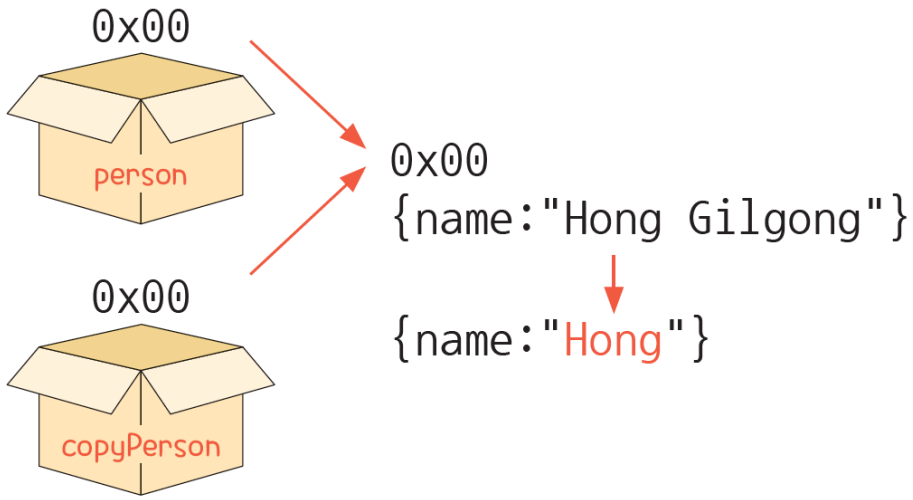
5. 객체의 데이터 관리 방법 이해하기

- 깊은 복사
  - 기본 자료형의 데이터 관리 방법
  - 복사한 값을 재할당할 때 한쪽 데이터가 변경되어도 서로 영향을 미치지 않게 복사되는 것

```
let num = 10;
let copyNum = num; // 변수 num의 데이터를 변수 copyNum에 할당
```

- 얕은 복사
  - 참조 자료형의 데이터 관리 방법
  - 한쪽 데이터가 변경되면 다른 쪽 데이터도 변경되어 서로 영향을 받는 것

그림 11-7 참조 데이터의 복사



11.3 표준 내장 객체 사용하기

- 표준 내장 객체 : 자바스크립트에 기본으로 내장된 객체

1. 문자열을 다루는 String 객체

- 기본 자료형 중 문자열과 관련 있는 속성과 메서드가 정의된 객체

2. 배열을 다루는 Array 객체

- 기본 자료형 중 배열과 관련 있는 속성과 메서드가 정의된 객체
- 파괴적 메서드 : 메서드를 사용했을 때 원본 데이터를 변경하는 메서드
- 비파괴적 메서드 : 원본을 변경하지 않는 메서드

11.3 표준 내장 객체 사용하기

표 11-1 string 객체의 주요 속성과 메서드

구분		설명
속성	length	문자열의 길이를 반환합니다.
메서드	includes()	메서드의 매개변수에 인자로 전달되는 문자열이 대상 문자열에 포함되어 있으면 true, 아니면 false를 반환합니다.
	replace()	대상 문자열에서 메서드의 매개변수에 인자로 전달되는 문자열과 일치하는 한 부분을 찾아서 다른 데이터로 변경한 새로운 문자열을 반환합니다.
	replaceAll()	대상 문자열에서 메서드의 매개변수에 인자로 전달되는 문자열과 일치하는 모든 부분을 찾아서 다른 데이터로 변경한 새로운 문자열을 반환합니다.
	split()	메서드의 매개변수에 인자로 전달되는 구분자를 이용해 대상 문자열을 여러 개의 문자열로 분리하고, 분리한 문자열을 새로운 배열로 반환합니다.
	toUpperCase()	대상 문자열을 대문자로 변경해 반환합니다.
	trim()	대상 문자열의 앞뒤 공백을 제거한 값을 반환합니다.
	indexOf()	대상 문자열과 일치하는 첫 번째 문자의 인덱스를 반환합니다.

11.3 표준 내장 객체 사용하기

11/03/string/pw_length.js

```
const pw = "124";
if(pw.length < 4){
  console.log("비밀번호는 최소 4자리 이상 입력해 주세요.");
}
```

11/03/string/includes.js

```
const email = "test!naver.com";
if(email.includes("@") === false){
  console.log("올바른 이메일 형식이 아닙니다.");
}
```

11/03/string/indexOf.js

```
const email = "test!naver.com";
if(email.indexOf("@") ===> -1){
  console.log("올바른 이메일 형식이 아닙니다.");
}
```

11.3 표준 내장 객체 사용하기

표 11-2 Array 객체의 주요 속성과 메서드

구분		설명
속성	length	배열의 요소 개수를 반환합니다.
파괴적 메서드	push()	배열의 맨 뒤에 데이터를 추가합니다.
	pop()	배열의 맨 뒤에서 데이터를 추출합니다.
	unshift()	배열의 맨 앞에 데이터를 추가합니다.
	shift()	배열의 맨 앞에서 데이터를 추출합니다.
	sort()	배열의 요소를 정렬합니다.
	reverse()	배열의 요소를 역순으로 정렬합니다.
비파괴적 메서드	forEach()	배열의 요소를 하나씩 순회하면서 요소마다 콜백(callback) 함수를 호출합니다.
	filter()	배열의 요소를 하나씩 순회하면서 요소마다 콜백 함수를 호출해 true를 반환하는 요소만 추출합니다. 추출한 요소로 새로운 배열을 만들고 만들어진 배열을 반환합니다.
비파괴적 메서드	find()	배열의 요소를 탐색하면서 주어진 판별 함수를 만족하는 첫 번째 값을 반환합니다.
	findIndex()	값 대신 인덱스 숫자를 반환한다는 것만 빼면 find() 메서드와 같습니다.
	includes()	배열에 특정 값이 포함되어 있는지 확인해서 포함됐으면 true, 아니면 false를 반환합니다.
	join()	배열의 모든 요소를 주어진 구분자로 합칩니다.

11.3 표준 내장 객체 사용하기

11/03/array/length_for.js

```
const arr = [10, 20, 30];
for(let i = 0; i < arr.length; i++){
  console.log(arr[i]);
}
```

```
const arr = [10, 20, 30, 40];
arr.push(50);     // 배열 맨 뒤에 50 추가
console.log(arr); // [10, 20, 30, 40, 50]
arr.pop();        // 배열 맨 뒤에서 요소 추출
console.log(arr); // [10, 20, 30, 40]
arr.unshift(0);   // 배열 맨 앞에 0 추가
console.log(arr); // [0, 10, 20, 30, 40]
arr.shift();      // 배열 맨 앞에서 요소 추출
console.log(arr); // [10, 20, 30, 40]
```

```
const arr = [10 , 20, 30, 40];
arr.forEach(function(v){
  console.log(v);
});
console.log(arr); // 10 20 30 40
```

11.3 표준 내장 객체 사용하기

3. 날짜와 시간을 다루는 Date 객체

- 날짜 및 시간과 관련 있는 속성과 메서드가 정의된 객체
- get 메서드 : 날짜와 시간 정보를 가져오는 메서드
- set 메서드 : 날짜와 시간 정보를 설정하는 메서드

4. 수학 연산을 다루는 Math 객체

- 수학 연산과 관련 있는 속성과 메서드가 정의된 객체
- random() 메서드 : 0 이상 1 미만의 난수를 반환하는 메서드

11.3 표준 내장 객체 사용하기

11/03/date/date_instance.js

```
const date = new Date(); // Wed Jan 19 2022 16:06:05 GMT+0900 (한국 표준시)
```

11/03/date/date_instance2.js

```
const date1 = new Date(2022, 11, 25); // Sun Dec 25 2022 00:00:00 GMT+0900 (한국 표준시)
const date2 = new Date(2022, 11, 25, 18, 30, 50); // Sun Dec 25 2022 18:30:50 GMT+0900 (한국 표준시)
```

11.3 표준 내장 객체 사용하기

표 11-3 Date 객체의 메서드

종류	설명
getFullYear()/setFullYear()	연도를 4자리 숫자로 표시합니다.
getMonth()/setMonth()	월을 0부터 11까지의 숫자로 표시합니다(1월 → 0, 12월 → 11).
getDate()/setDate()	일을 1부터 31까지의 숫자로 표시합니다.
getDay()	요일을 0부터 6까지의 숫자로 표시합니다(일요일 → 0, 토요일 → 6).
getTime()/setTime()	1970년 1월 1일 12:00 이후의 시간을 밀리초(1/1000초) 단위로 표시합니다.
getHours()/setHours()	시를 0부터 23까지의 숫자로 표시합니다.
getMinutes()/setMinutes()	분을 0부터 59까지의 숫자로 표시합니다.
getSeconds()/setSeconds()	초를 0부터 59까지의 숫자로 표시합니다.
getMilliseconds()/setMilliseconds()	밀리초를 0부터 999까지의 숫자로 표시합니다.

11.3 표준 내장 객체 사용하기

11/03/date_method.js

```
const date = new Date(2022, 11, 25, 18, 30, 50);
const dateFormat = `${date.getFullYear()}-${date.getMonth()+1}-${date.getDate()}
${date.getHours()}:${date.getMinutes()}:${date.getSeconds()}`;
console.log(dateFormat); // 2022-12-25 18:30:50
```

11/03/date/getDateDiff.js

```
const date1 = new Date('2022-12-23');
const date2 = new Date('2022-12-25');
const dateDiff = date2.getTime() - date1.getTime();
const interval = dateDiff / (24 * 60 * 60 * 1000);
console.log(`두 날짜의 차이는 ${interval}일입니다.`); // 두 날짜의 차이는 2일입니다.
```

11.3 표준 내장 객체 사용하기

표 11-4 Math 객체의 주요 메서드

종류	설명
Math.floor()	주어진 숫자와 같거나 작은 정수 중에서 가장 큰 수를 반환합니다(내림).
Math.ceil()	주어진 숫자와 같거나 큰 정수 중에서 가장 작은 수를 반환합니다(올림).
Math.round()	주어진 숫자를 반올림한 수와 가장 가까운 정수를 반환합니다(반올림).
Math.random()	0 이상 1 미만의 난수를 반환합니다.

```
const floatNum = 10.52;
Math.floor(floatNum); // 10
Math.ceil(floatNum);  // 11
Math.round(floatNum); // 11
```

11/03/math/random.js

```
const random = Math.random();
console.log(random); // 0.2982742766551536(실행할 때마다 달라짐)
```

11.3 표준 내장 객체 사용하기

11/03/math/getMaxRandom.js

```
function getMaxRandom(max){
  return Math.floor(Math.random() * max) + 1;
}
const maxRandom = getMaxRandom(20);
console.log(maxRandom); // 0 이상 20 이하의 무작위 정수
```

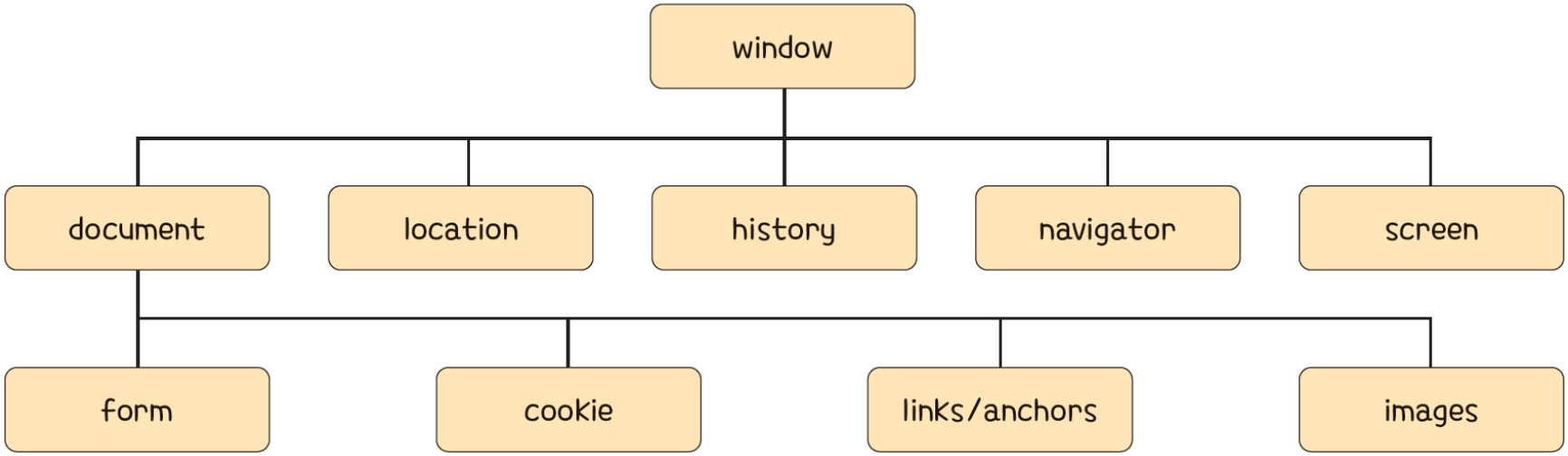
11/03/math/getMinMaxRandom.js

```
function getMinMaxRandom(min, max){
  return Math.floor(Math.random() * (max - min)) + 1 + min; // 20을 제외하고 싶으면 + 1 삭제
}
const maxRandom = getMinMaxRandom(10, 20);
console.log(maxRandom); // 10 이상 20 이하의 무작위 정수
```

11.4 브라우저 객체 모델 사용하기

- 브라우저 객체 모델 : 자바스크립트 언어 사양에 포함되지 않고 웹 브라우저에서 제공하는 객체

그림 11-8 브라우저 객체 모델의 계층도



11.4 브라우저 객체 모델 사용하기

- 브라우저 객체 모델의 종류
  - window : 웹 브라우저가 열릴 때마다 생성되는 최상위 관리 객체
  - document : 웹 브라우저에 표시되는 HTML 문서 정보가 포함된 객체
  - location : 웹 브라우저에 현재 표시된 페이지에 대한 URL 정보가 포함된 객체
  - history : 웹 브라우저에 저장된 방문 기록이 포함된 객체
  - navigator : 웹 브라우저 정보가 포함된 객체
  - screen : 웹 브라우저의 화면 정보가 포함된 객체

11.4 브라우저 객체 모델 사용하기

1. window 객체의 속성과 메서드

- 주요 속성(px 단위)
  - innerWidth : 웹 브라우저 화면의 너비
  - innerHeight : 웹 브라우저 화면의 높이
  - outerWidth : 웹 브라우저 창의 너비
  - outerHeight : 웹 브라우저 창의 높이
  - screenTop/screenY : 웹 브라우저 위쪽 면과 모니터의 간격
  - screenLeft/screenX : 웹 브라우저 왼쪽 면과 모니터의 간격
  - pageXOffset/scrollX : 웹 브라우저의 수평 스크롤 위치
  - pageYOffset/scrollY : 웹 브라우저의 수직 스크롤 위치

11.4 브라우저 객체 모델 사용하기

- 주요 메서드
  - alert() : 알림창 표시
  - confirm() : 확인창 표시
  - prompt() : 입력창 표시
  - open() : 새로운 웹 브라우저 창 열기
  - close() : 웹 브라우저 창 닫기
  - setTimeout() : 일정 시간(ms) 뒤에 콜백 함수를 한 번만 실행
  - setInterval() : 일정 시간(ms)마다 콜백 함수를 반복 실행
  - clearInterval : setInterval() 메서드로 반복 실행되는 함수를 중지
  - scrollTo() : 웹 브라우저의 스크롤을 특정 위치만큼 이동
  - scrollBy() : 웹 브라우저의 스크롤을 현재 위치에서 상대 위치로 이동

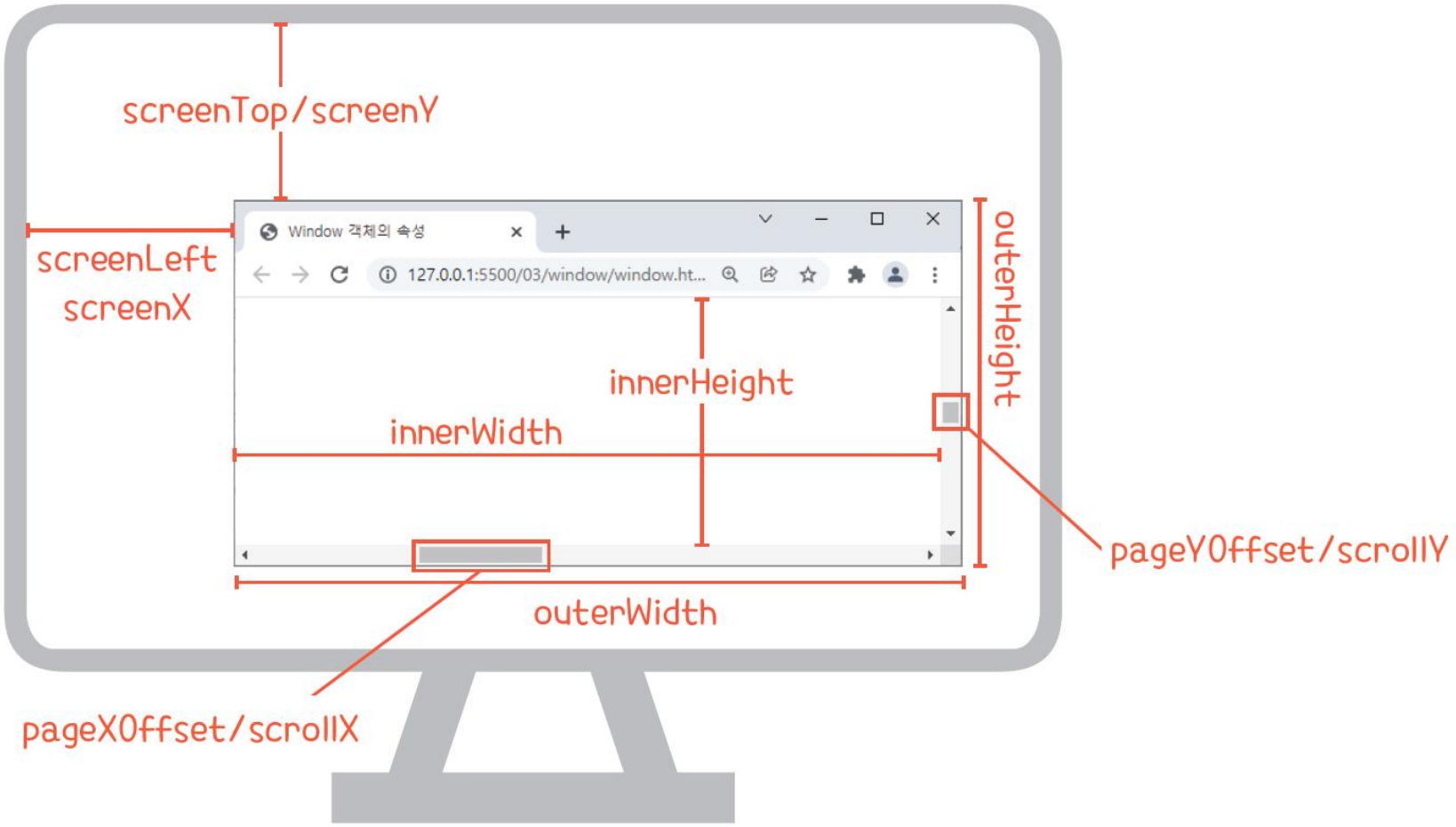
11.4 브라우저 객체 모델 사용하기

2. window 객체의 기본 속성 사용하기

- innerWidth : 웹 브라우저의 너비
- innerHeight : 웹 브라우저의 높이
- outerWidth : 웹 브라우저 창의 너비
- outerHeight : 웹 브라우저 창의 높이
- screenTop/screenY : 웹 브라우저 창 위쪽 면과 모니터 사이의 간격
- screenLeft/screenX : 웹 브라우저 창 왼쪽 면과 모니터 사이의 간격
- scroll : 웹 브라우저 창의 스크롤 가로 위치
- scrollY : 웹 브라우저 창의 스크롤 세로 위치

11.4 브라우저 객체 모델 사용하기

그림 11-9 window 객체 속성



11.4 브라우저 객체 모델 사용하기 과제

코드 캡쳐, window 객체 속성 버튼 클릭 후 콘솔창 출력내용 캡쳐

```html
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Window 객체의 속성</title>
  <style>
    /* 웹 브라우저에 스크롤을 생기게 하기 위해 body 태그에 너이, 높이 추가 */
    body{
      width:2500px;
      height:5000px;
    }
    /* 스크롤 되어도 버튼이 따라다니게 하기 위해 fixed 속성값 추가 */
    button{
      position:fixed;
      left:10px;
      top:10px;
    }
  </style>
</head>
<body>
  <button onclick="printInfo()">window 객체 속성</button>
  <script>
    function printInfo(){
      console.log(`웹 브라우저의 너비(innerWidth): ${window.innerWidth}`);
      console.log(`웹 브라우저의 높이(innerHeight): ${window.innerHeight}`);
      console.log(`웹 브라우저 창의 너비(outerWidth): ${window.outerWidth}`);
      console.log(`웹 브라우저 창의 높이(outerHeight): ${window.outerHeight}`);
      console.log(`웹 브라우저 창 위쪽 면과 모니터 사이의 간격(screenTop,screenY): ${window.screenTop}/${window.screenY}`);
      console.log(`웹 브라우저 창 왼쪽 면과 모니터 사이의 간격(screenLeft,screenX): ${window.screenLeft}/${window.screenX}`);
      console.log(`웹 브라우저 창의 스크롤 가로 위치(scrollX): ${window.scrollX}`);
      console.log(`웹 브라우저 창의 스크롤 세로 위치(scrollY): ${window.scrollY}`);
    }
  </script>
</body>
</html>
```

11.4 브라우저 객체 모델 사용하기

3. 웹 브라우저에서 새 창 제어하기

- open() : 웹 브라우저에서 새로운 창을 여는 데 사용
- 팝업창 : window.open() 메서드로 열리는 새 창

  형식 `window.open(경로, 이름, 속성);`

- 창 제어 속성
  - width : 웹 브라우저의 너비를 px 단위로 지정
  - height : 웹 브라우저의 높이를 px 단위로 지정
  - left : 웹 브라우저 왼쪽에서의 위치를 px 단위로 지정
  - top : 웹 브라우저 위쪽에서의 위치를 px 단위로 지정

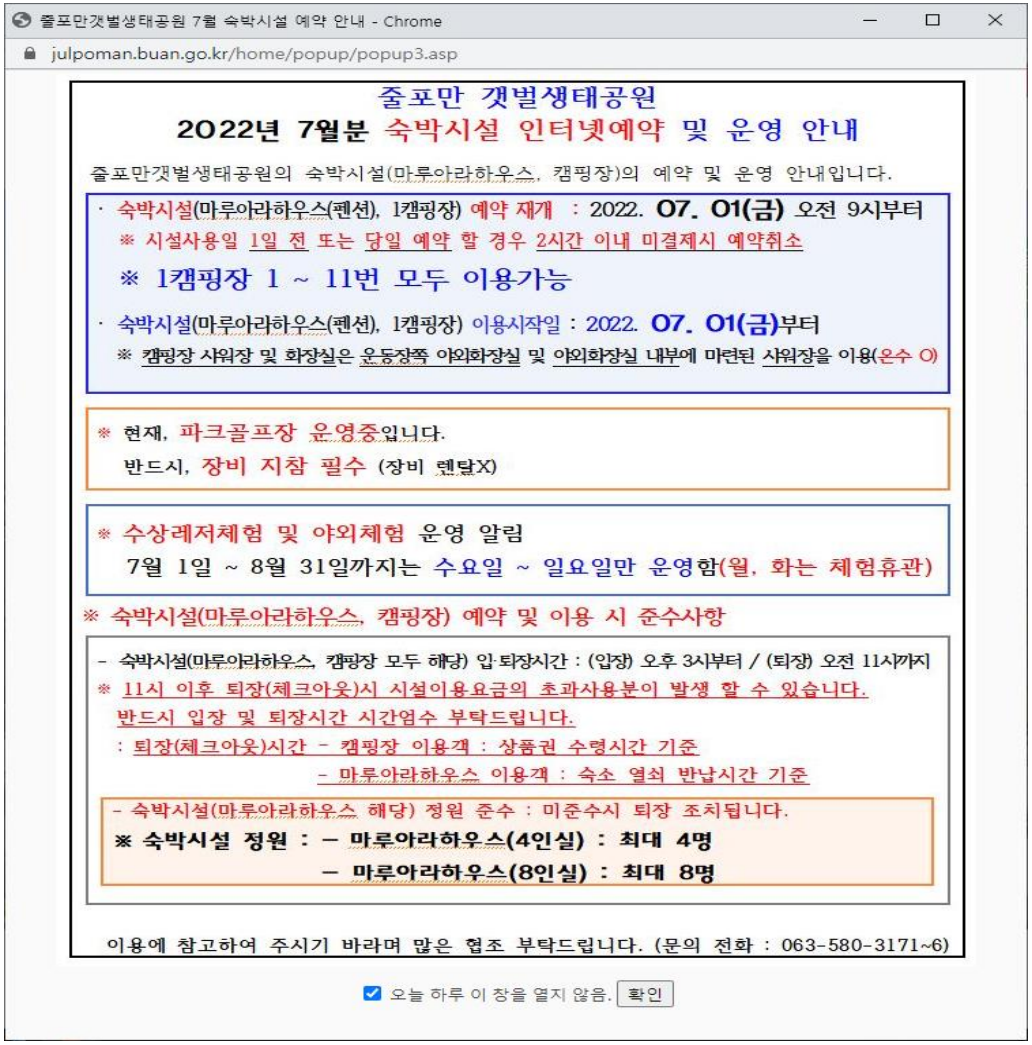
11.4 브라우저 객체 모델 사용하기

3. 웹 브라우저에서 새 창 제어하기 예제 실습

예제코드 : 080313-main₩11₩04₩open.html

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>open</title>
</head>
<body>
  <button onclick="popup()">팝업</button>
  <script>
    function popup(){
      window.open('popup.html', '팝업', 'width=200, height=100');
      // window.open('popup.html', '팝업', 'width=200, height=100, left=400, top=400, titlebar=no, resizable=yes');
    }
  </script>
</body>
</html>
```

11.4 브라우저 객체 모델 사용하기



11.4 브라우저 객체 모델 사용하기

4. 웹 브라우저의 스크롤 이동하기

- scrollTo() : 웹 브라우저의 스크롤 위치를 특정 좌표로 이동
- scrollBy() : 웹 브라우저의 스크롤을 현재 위치에서 상대 위치로 이동

형식 `window.scrollTo(x좌표, y좌표);`

`window.scrollBy(x좌표, y좌표);`

- behavior 속성(smooth로 값을 설정하면 부드럽게 이동)
  - scrollTo() 메서드나 scrollBy() 메서드의 매개변수에 객체 리터럴을 전달할 때, behavior 속성을 전달할 수 있음
  - smooth 값 : 웹 브라우저 스크롤이 해당 위치로 마우스 휠을 굴리듯이 부드럽게 이동(IE, 사파리 웹 브라우저에서는 지원하지 않음)

11.4 브라우저 객체 모델 사용하기

4. 웹 브라우저의 스크롤 이동하기 예제

```
<body>
  <div>
    <button onclick="sTo()">scrollTo(100, 200)</button>
    <button onclick="sBy()">scrollBy(100, 200)</button>
  </div>
  <script>
    function sTo(){
      window.scrollTo(100, 200);
      // window.scrollTo({left:100, top:200});
    }
    function sBy(){
      window.scrollBy(100, 200);
      // window.scrollBy({left:100, top:200});
    }
  </script>
</body>
```