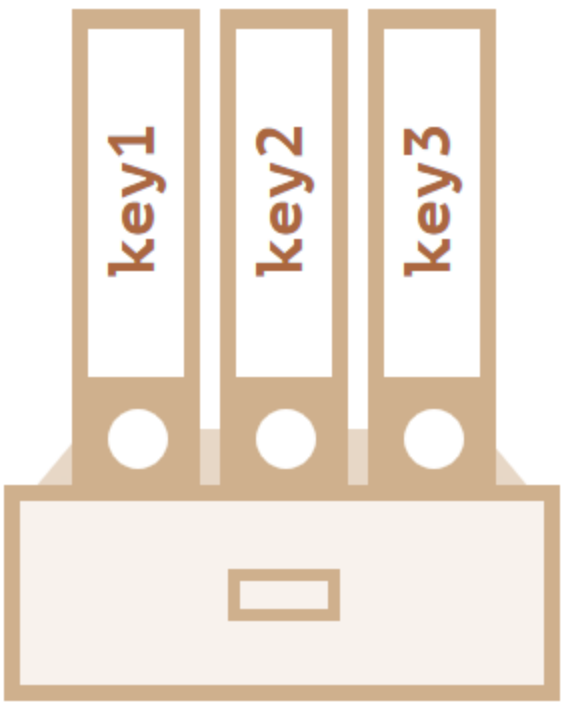
04-1. 객체: 기본

객체(Object)

- JS의 자료형은 8개
    - 그 중 7개는 하나의 데이터만 담을 수 있는 원시형
    - 나머지 하나가 다양한 데이터를 담을 수 있는 객체형
- 객체는 JS의 모든 면에 녹아있기에, 이해가 필수적
    - JS에서는 배열(Array)도 객체고, 함수(Function)도 객체다.

구성

- 중괄호 `{...}`로 표현된다.
    - '키(key): 값(value)' 쌍으로 구성
    - '키(key): 값(value)' 쌍을 *프로퍼티(property)*라고 하며, 이것을 객체에 여러 개 넣을 수 있다.
        - `키`엔 `문자형`과 `심볼형`만
        - `값`엔 `모든 자료형`(객체 포함)이 허용된다.
- 서랍장으로 생각하면 이해가 쉽다.



대괄호 표기법

- 대괄호 표기법은 이름과 값의 제약을 없애주기에, 유연하고 강력하다. 대신 작성이 번거롭다.
- 여러 단어를 key값으로 준다면, 이를 묶어줘야한다.
    - 선언할 때에는 따옴표로, 변수 식별자로 접근할 때에는 **대괄호 표기법**으로

```
// 선언 시
let user = {
  name: "John",
  age: 30,
  "likes birds": true  // 복수의 단어는 따옴표로 묶어야
```

```
};

alert(user.name); // John
alert(user["likes birds"]); // true

// 대괄호 표기법, 이 때도 따옴표로 묶어줘야 한다.
let user = {};

user["likes birds"] = true;
alert(user["likes birds"]); // true
delete user["likes birds"];
```

- 또한 대괄호 표기법을 사용하면 key값을 변수로 할당할 수 있다.
  - 아래와 같이 key값이 변동될 수 있는 상황에서 유용하다.

```
let user = {
  name: "John",
  age: 30
};

let key = prompt("사용자의 어떤 정보를 얻고 싶으신가요?", "name");

// 변수로 접근
alert( user[key] ); // John (프롬프트 창에 "name"을 입력한 경우)

let key = "name";
alert( user.key ) // undefined, 점 표기법은 변수 할당 불가
```

- 객체는 const로 선언하더라도 수정이 가능하다.
  - 단, 아예 `user = ...`처럼 전체를 설정하려고 하면 오류가 발생한다.

```
const user = {
  name: "John"
};

user.name = "Pete"; // (*)
alert(user.name); // Pete
```

‘in’ 연산자로 프로퍼티 존재 여부 확인하기

- JS는 존재하지 않는 프로퍼티에 접근 시, 에러가 발생하는 대신 `undefined`반환
- 이를 응용해 프로퍼티 존재 여부 확인 가능
- 그런데 만약 `user.name`의 값이 `undefined`였다면?
  - `"Key" in object`를 사용하면 확인 가능

```
let user = {
    name: undefined,
    age: 30,
};

// undefined로 확인
alert( user.name === undefined ); // 프로퍼티가 존재하지만 true 출력
alert( user.age === undefined ); // 프로퍼티가 존재하기에 false 출력
alert( user.blabla === undefined ); // 존재하지 않기 때문에 true 출력

// "key" in object
alert( "name" in user ); // user.name이 존재하므로 true 출력
```

```
alert( "age" in user ); // user.age가 존재하므로 true가 출력
alert( "blabla" in user ); // user.blabla는 존재하지 않기 때문에 false가 출력
```

‘for...in’ 반복문

- 객체의 모든 키를 순회하는 반복문

```
let user = {
  name: "John",
  age: 30,
  isAdmin: true
};

for (let key in user) {
  // 키
  alert( key );  // name, age, isAdmin
  // 키에 해당하는 값
  alert( user[key] ); // John, 30, true
}
```

정렬

- 보통은 작성된 순서대로 프로퍼티가 나열
- 하지만 만약 key값으로 정수를 사용한다면, 자동으로 정렬이 일어남
  - 이를 막기 위해서는 번호 앞에 “+”를 붙여서 정수로 취급되지 않도록 속여야

```
let codes = {
  "49": "독일",
  "41": "스위스",
  "44": "영국",
  // ..,
  "1": "미국"
};

for (let code in codes) {
  alert(code); // 1, 41, 44, 49
}

// 번호 앞에 "+" 붙이기
let codes = {
  "+49": "독일",
  "+41": "스위스",
  "+44": "영국",
  // ..,
  "+1": "미국"
};

for (let code in codes) {
  alert( +code ); // 49, 41, 44, 1
}
```

기타

- 단축 프로퍼티
  - `key : value`가 동일하다면 아래와 같이 단축된 형태로 표현 가능

```
function makeUser(name, age) {
  return {
```

```
    name, // name: name 과 같음
    age: 30,
  };
}
```

- 제약 사항
  - `key` 값에는 `for`, `let`, `return` 등 어떤 문자열을 사용해도 괜찮음
  - `__proto__` 는 사용불가! 프로토타입 상속 때문.
- 객체의 종류
  - 사실 지금까지 공부한 건 모두 일반 객체
  - `Array` – 정렬된 데이터 컬렉션을 저장할 때 쓰임
  - `Date` – 날짜와 시간 정보를 저장할 때 쓰임
  - `Error` – 에러 정보를 저장할 때 쓰임
  - 기타 등등 여러 객체가 있음 ㅋㅋ