11장. 원시 값과 객체의 비교

▼ 목차

원시 타입과 객체 타입

원시 타입

- `number`
- `bigint`
- `string`
- `boolean`
- `null`
- `undefined`
- `Symbol`

객체 타입 (참조 타입)

원시 타입을 제외한 모든 타입

- `객체 {}`
- `배열 []`

- `function`

변경 불가능한 값과 변경 가능한 값

변경 불가능한 값 - 원시 타입

한 번 생성된 원시 값은 읽기 전용 값으로서 변경할 수 없습니다.

값을 변경할 수 없다는 것이 구체적으로 무엇을 말하는지 알아봅시다. 먼저 변수와 값을 구분해야합니다.

- 변수 : 하나의 값을 저장하기 위해 확보한 메모리 공간 자체 또는 그 메모리 공간을 식별하기 위해 붙인 이름
- 값 : 변수에 저장된 데이터로서 표현식이 평가되어 생성된 결과

```
var something = 100;
       변수       값

something = 22; // 재할당
```

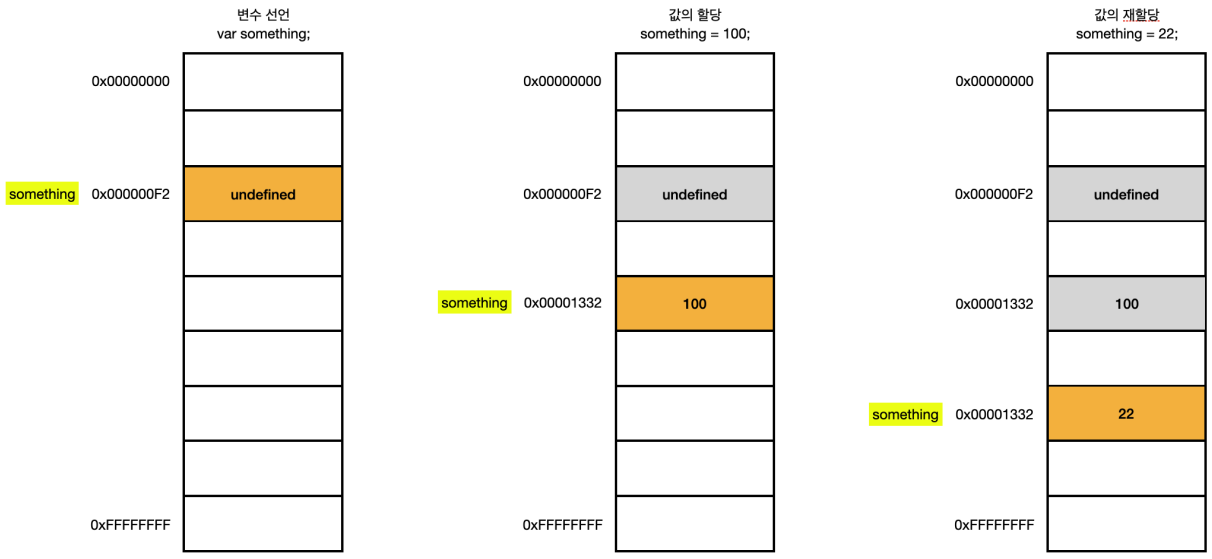
위 상황에서 `100` 이라는 원시값 자체가 `22`로 변경되는 것이 아니라 `something` 변수에 저장된 값이 `100`에서 `22`로 변경된 것입니다.

변경 불가능하다는 것은 **원시 값 자체를 변경할 수 없다는 것**이지 변수에 값을 재할당할 수 없다는 것이 아닙니다.

변수의 상대 개념으로 상수가 있습니다. 상수는 **재할당이 금지된 변수**를 말합니다.

```
const something = 100;

something = 22; // Cannot assign to 'something' because it is a constant.
```

변수에 값을 재할당하면 원시 값이 변경되는 것처럼 보이지만, 사실은 변수가 가리키는 메모리 공간의 주소가 바뀌는 것입니다.



변수에 값을 할당할 때마다 메모리 공간의 주소가 변경되는 이유는 변수에 할당된 원시 값이 변경 불가능한 값이기 때문입니다. 원시 값이 변경 가능한 값이라면 변수에 새로운 원시 값을 재할당했을 때 변수가 가리키던 메모리 공간의 주소를 바꿀 필요없이 원시 값 자체를 변경하면 되고, 그러면 변수가 가리키는 메모리 공간의 주소는 바뀌지 않습니다.

하지만 자바스크립트에서 원시 값은 변경 불가능한 값이기 때문에 값을 직접 변경할 수 없습니다. 따라서 변수의 값을 변경하기 위해 원시 값을 재할당하면 새로운 메모리 공간을 확보하고 재할당한 값을 저장한 후, 변수가 가리키던 메모리 공간의 주소를 변경합니다. 값의 이러한 특성을 불변성이라 하고, 변경 불가능한 값을 불변값이라고 부르기도 합니다.

변경 가능한 값 - 객체 타입

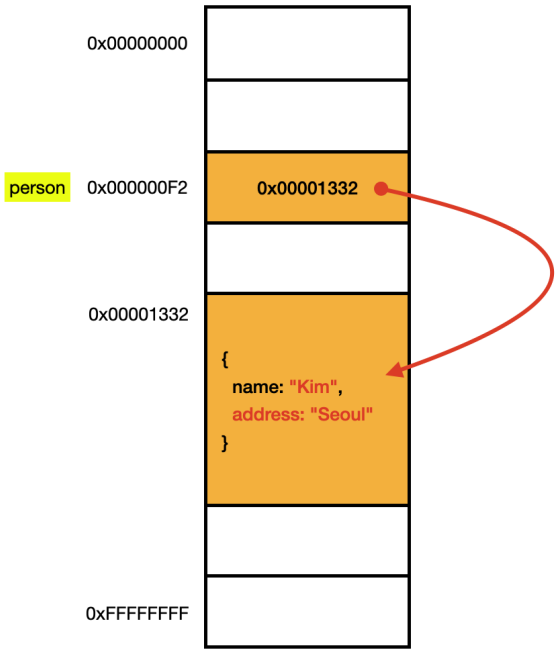
변수에 원시 값을 할당하면 변수가 가리키는 메모리 주소를 통해 메모리 공간에 접근하여 원시 값에 접근합니다. 즉, 원시 값을 할당한 변수는 원시 값 자체를 값으로 갖습니다. 하지만 변수에 객체를 할당하면 변수가 가리키는 메모리 주소를 통해 메모리 공간에 접근하여 **참조 값**에 접근합니다. 참조 값이란 생성된 객체가 저장된 메모리 공간의 주소, 그 자체를 의미합니다.

```
var something = 100;
var person = {
  name: 'Lee',
};
```

원시 값을 할당한 변수를 참조하면 메모리에 저장되어 있는 원시 값에 접근합니다. 하지만 객체를 할당한 변수를 참조하면 메모리에 저장되어 있는 참조 값을 통해서 실제 객체에 접근합니다.

원시 값은 변경 불가능한 값이므로 원시 값을 갖는 변수의 값을 변경하려면 재할당 외에는 방법이 없습니다. 하지만 객체는 변경 가능한 값이기 때문에 객체를 할당한 변수는 재할당없이 객체를 직접 변경할 수 있습니다.

```
var person = {
  name: 'Lee',
};

person.name = "Kim";
person.address = "Seoul";

console.log(person); // { name: "Kim", address: "Seoul" }
```



객체는 프로퍼티의 개수가 정해져있지 않고, 동적으로 추가되고 삭제할 수 있습니다. 그리고 프로퍼티의 값에도 제약이 없습니다. 따라서 객체는 원시 값과 같이 확보해야 할 메모리 공간의 크기를 사전에 정해 둘 수 없습니다. 그리고 객체를 생성하고 관리하는 방식이 매우 복잡하고 비용이 많이 드는 작업입니다. 객체를 변경할 때마다 원시 값처럼 이전 값을 복사해서 새롭게 생성한다면 명확하고 신뢰성이 확보되겠지만 객체는 크기가 매우 클 수도 있고, 원시 값처럼 크기가 일정하지도 않으며, 프로퍼티 값이 객체일 수도 있어서 복사해서 생성하는 비용이 많이 듭니다. 다시 말해, 메모리의 효율적 소비가 어렵고 성능이 나빠집니다. 따라서 성능을 효율적으로 사용하기 위해, 그리고 객체를 복사해 생성하는 비용을 절약하여 성능을 향상시키기 위해 객체는 변경 가능한 값으로 설계되어있습니다. 매모리 사용의 효율성과 성능을 위해 어느 정도의 구조적인 단점을 감안한 설계라고 할 수 있습니다.

객체는 이러한 구조적 단점에 따른 부작용이 있습니다. 그것은 원시 값과 다르게 여러 개의 식별자가 하나의 객체를 공유할 수 있다는 것입니다.

자바스크립트가 값을 전달하는 방법

모던 자바스크립트 Deep Dive에서는 값에 의한 전달, 참조에 의한 전달이라는 표현을 사용했습니다. 저자도 이러한 용어들이 자바스크립트를 위한 용어가 아니라고 언급을 했기 때문에 혼란을 방지하기 위해서 해당 용어들을 사용하지 않겠습니다.

원시 타입의 값을 전달하는 방법

```
var score = 80;
var copy = score;

console.log(score); // 80
console.log(copy); // 80

score = 100;

console.log(score); // 100
console.log(copy); // ?
```

`score` 변수에 숫자 `80`을 할당하고, `score` 변수를 `copy` 변수에 할당했습니다. 이 때 `score` 변수에 숫자 `100`을 재할당하면 `copy` 변수의 값은 어떻게 될까요?

이 질문의 핵심은 **"변수에 변수를 할당했을 때 무엇이 어떻게 전달되는가?"**입니다.

`copy = score`에서 `score`는 숫자 `80`으로 평가되므로 `copy` 변수에도 `80`이 할당됩니다.

이처럼 변수에 원시 값을 갖는 변수를 할당하면 할당받는 변수(`copy`)에는 할당되는 변수(`score`)의 원시 값이 복사되어 전달됩니다.

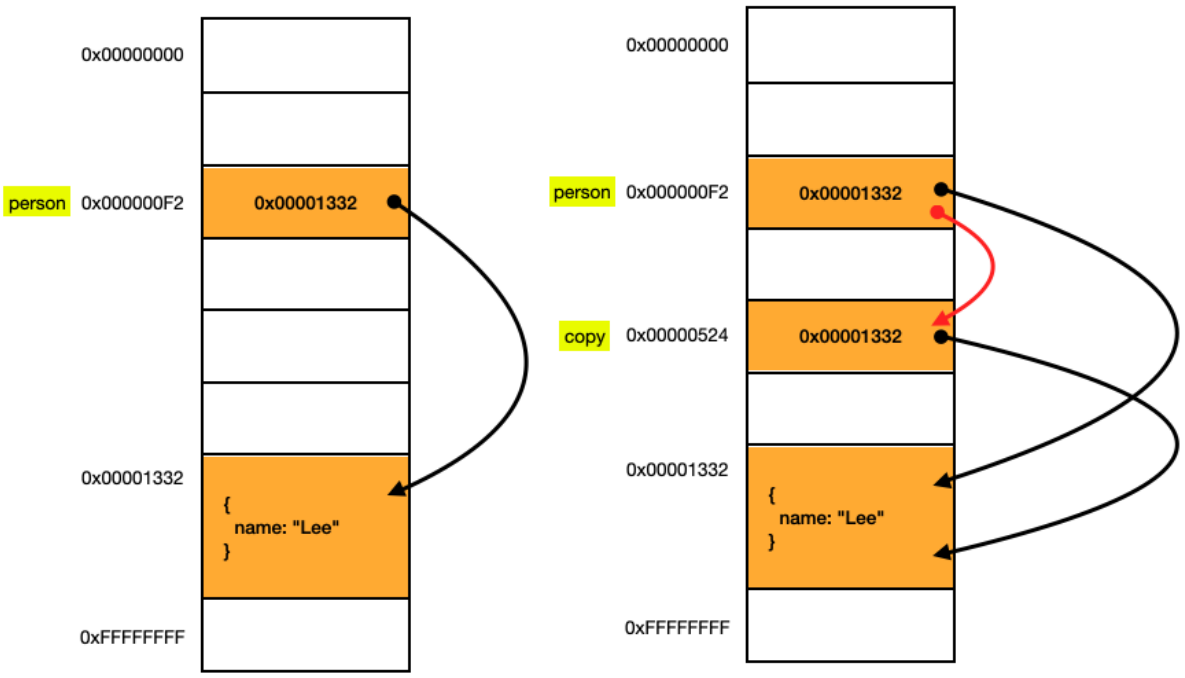
이 때 `score` 변수와 `copy` 변수는 숫자 `80`을 갖는다는 점에서 동일합니다. 하지만 `score` 변수와 `copy` 변수의 값 `80`은 다른 메모리 공간에 저장된 별개의 값입니다.

객체 타입의 값을 전달하는 방법

```
var person = {
  name: "Lee"
};

var copy = person;

console.log(copy === person); // true

copy.name = "Kim";

person.address = "Seoul";

console.log(copy); // { name: "Kim", address: "Seoul" }
console.log(person); // { name: "Kim", address: "Seoul" }
```

변수에 객체를 갖는 변수를 할당하면 할당받는 변수(`copy`)에는 할당되는 변수(`person`)가 가지고 있는 참조 값이 복사되어 전달됩니다.

앞에서 언급했듯이 객체는 구조적 단점에 따른 부작용이 있습니다. 참조 값이 복사되어 전달되기 때문에 원시 값과 다르게 여러 개의 식별자가 하나의 객체를 공유할 수 있습니다.



정리

자바스크립트에서 변수에 변수를 할당하면 원시 값이든 참조 값이든 복사해서 전달합니다.

의문점

> 엄격하게 표현하면 변수에는 값이 전달되는 것이 아니라 **메모리 주소가 전달되기** 때문이다. 이는 변수와 같은 식별자는 값이 아니라 메모리 주소를 기억하고 있기 때문이다. (p. 145)

> 이처럼 **"값에 의한 전달(원시값이 전달되는 상황)"도 사실은 값을 전달하는 것이 아니라 메모리 주소를 전달**한다. 단, 전달된 메모리 주소를 통해 메모리 공간에 접근하면 값을 참조할 수 있다. (p. 145)

해당 문장을 보니까 이 책에서 주로 설명하는 내용보다는 코어 자바스크립트에서 설명하는 게 더 타당하다는 생각이 들었습니다.

코어자바스크립트에서는 원시 타입이든 객체 타입이든 변수는 메모리 주소를 저장하고 있습니다.

```
var a = "abc";
```

`a`변수는 값을 아래처럼 저장하고 있습니다.

주소	...	1002	1003	1004	1005	...
데이터			식별자: a 값: @5004			
주소	...	5002	5003	5004	5005	...
데이터				"abc"		

```
var a = "abc";

a = "abcdef";
```

`a` 변수에 값을 재할당하면 아래와 같이 변경됩니다.

주소	...	1002	1003	1004	1005	...
데이터			식별자: a 값: @5005			
주소	...	5002	5003	5004	5005	...
데이터				"abc"	"abcdef"	

```
var a = "abc";
var copy = a;
```

변수에 변수를 할당하면 원시 값을 가지고 있는 변수라도 참조 값을 복사해서 전달합니다.

주소	...	1002	1003	1004	1005	...
데이터			식별자: a 값: @5004		식별자: copy 값: @5004	
주소	...	5002	5003	5004	5005	...
데이터				"abc"		