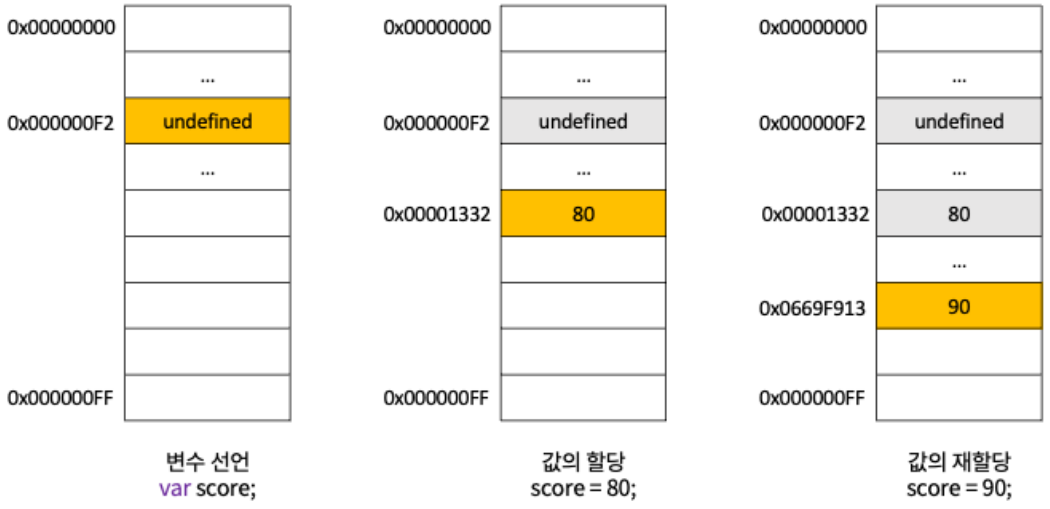
Chap11 원시 값과 객체의 비교

- 원시 타입의 값 : 변경 불가능한 값
  - 원시 값을 변수에 할당하면 변수(확보된 메모리 공간)에는 실제 값이 저장된다.
  - 원시 값을 변수를 다른 변수에 할당하면 원본의 원시 값이 복사되어 전달된다. `= 값의 의한 전달`
- 객체 타입의 값 : 변경 가능한 값
  - 변수 확보된 메모리 공간에는 참조 값이 저장된다.
  - 객체 값을 다른 변수에 할당하면 원본의 참조 값이 복사되어 전달 `= 참조에 의한 전달`

11-1 원시값

▼ 변경 불가능한 값

변경 불가능하다는 것은 변수가 아닐 값이다. 즉 원시 값 자체를 변경할 수 없다는 것이지 변수 값을 변경할 수 없다는 것이 아니다. 변수는 언제든지 재할당을 통해 변수 값을 변경(교체)할 수 있다.



즉 재할당하여 변수 값이 바뀌는 것이지 원시 값이 바뀌는 것이 아니다. 만약 원시값이 바뀐다고 하면 값을 재할당 할 때 변수가 가리키던 메모리 공간의 주소를 바꿀 필요없이 원시 값만 변경될 것이다.

따라서 원시값은 변경 불가능한 값이기 때문에 **불변성**이란 특징을 갖는다.

▼ 문자열과 불변성

```
var str='Hello';
str='world';
```

문자열도 원시값이기 때문에 변수값은 재할당할 수 있어도 원시값은 바꿀 수 없다.

하지만 여기서 헷갈리는 개념이 있다. 바로 문자열은 유사 배열 객체라는 점이다.

▼ 갑자기 문자열이 객체??

유사 배열 객체란 마치 배열처럼 인덱스로 프로퍼티 값에 접그할 수 있고 length 프로퍼티를 갖는 객체를 말하다.

따라서 **문자열**은 마치 배열처럼 인덱스ㅡㄹ 통해 각 문제 접근할 수 있으며 length 프로퍼티를 갖기 때문에 유사 뱅려 객체이고 for문으로 순회할 수도 있다.

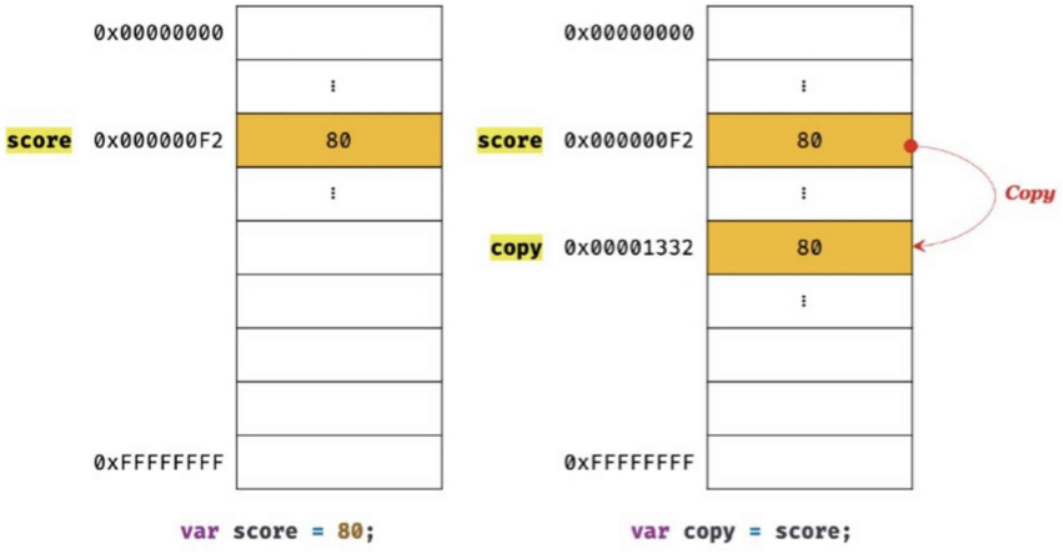
그러나 아무튼 문자열은 유사 배열 객체이기도 하지만 원시값이기때문에 원시 값을 변경하려고 했을 때 변경되지 않는다.

```
var str='string';
str[0]='S'; //원시값 바꾸려고 시도함. -> 결론은 바뀌지 않음.

console.log(str); // string -> 값이 바뀌지 않음
```

▼ 값에 의한 전달

값에 의한 전달이란 변수에 원시 값을 갖는 변수를 할당하면 할당 받는 변수에는 할당되는 변수의 원시 값이 복사되어 전달되는 것

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

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



여기서 주의할 점은 copy변수는 scroe 변수의 값을 할당받아 같은 값을 갖는다고 같은 메모리 공간에 저장되었다고 착각하면 안된다.

말그대로 복사를 해서 다른 메모리 공간에 값을 저장한 것이다.

따라서 score의 값을 변경하면 copy의 값은 변경되지 않는다.

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

score=100;
console.log(score); //100
console.log(copy); //80
```

그렇다면 의문점이 드는데 변수에는 값이 전달이 아니고 메모리 주소가 전달되는 거 아닌가?

그렇다 메모리 주소를 전달해주는 것이다. 사실 값에 의한 전달이라는 용어를 쓰는 이유가 타 언어에서 자주 사용해서 이지만 그렇게 썩 맞는 말은 아니다. 그래서 공유에 의한 전달이라고 표현하는 경우도 있다.

결론적으로 두 변수의 원시 값은 서로 다른 메모리 공간에 저장된 별개의 값이 되어 어느 한쪽에서 재할당을 통해 값을 변경해도 서로 간섭할 수 없다.

11-2 객체

▼ 변경 가능한 값

객체를 할당한 변수에 참조(메모리 공간의 주소에 접근하는)하면 메모리에 저장되어 있는 참조 값을 통해 객체에 접근한다

- 복사했을 때

사진

참조 값이 복사된다.

객체는 크기가 크고 프로퍼티로 또 객체를 갖고있을 수 있어서 비용이 많이 들기 때문에 복사해서 값을 변경하게 안하고 바로 값을 변경할 수 있도록 설정했다

▼ 얕은 복사 깊은 복사

얕은 복사는 객체의 한 단계까지만 복사하는 것을 말하고,

깊은 복사는 객체에 중첩되어 있는 객체까지 모두 복사하는 것을 말한다.

```
const o = { x: { y: 1 } };

// 얕은 복사
const c1 = { ...o }; // 35장 "스프레드 문법" 참고
console.log(c1 === o); // false
console.log(c1.x === o.x); // true

// lodash의 cloneDeep을 사용한 깊은 복사
// "npm install lodash"로 lodash를 설치한 후, Node.js 환경
const _ = require('lodash');
// 깊은 복사
const c2 = _.cloneDeep(o);
console.log(c2 === o); // false
console.log(c2.x === o.x); // false
```

▼ 참조에 의한 전달

객체를 가리키는 변수를 다른 변수에 할당하면 원본의 참조 값이 복사되어 전달되는 것

▼ 결론

값에 의한 전달과 참조에 의한 전달은 식별자가 기억하는

메모리 공간에 저장되어 있는 값을 복사해서 전달한다는 면에서 동일하다.

다만, 식별자가 기억하는 메모리 공간, 즉 **변수에 저장되어 있는 값이 원시 값이냐, 참조 값이냐의 차이**로 동작이 다른 것이다.