자바스크립트 문법

3/ 자바스크립트 문법

목차

01

호이스팅

변수, 표현식과 문

- 변수 선언은 소스코드가 한 줄씩 순차적으로 실행되는 시점(런타임)이 아니라 그 이전 단계에 실행된다.
  - 생성단계
  - 실행단계
- 변수 선언이 소스코드 어느 위치에 있든 상관없이 다른 코드보다 제일 먼저 실행됨
- 변수 선언문이 코드의 선두로 끌어 올려진 것처럼 동작하는 자바스크립트 고유의 특징을 **변수 호이스팅** 이라고 한다.

```
console.log(score); // undefined

var score; // 변수 선언문
```

```
var score = 80; // 변수 선언과 값의 할당
```

let, const, var

let 키워드 특징

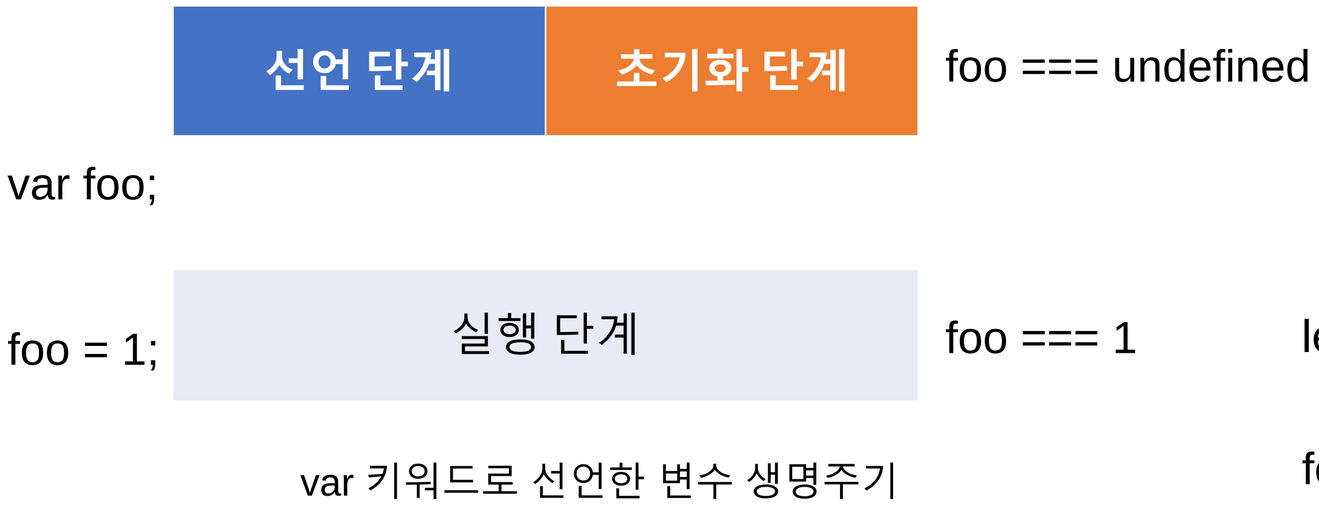
- 변수 중복 선언 금지
- 블록 레벨 스코프 가능
- **변수 호이스팅 발생하지 않도록 함**

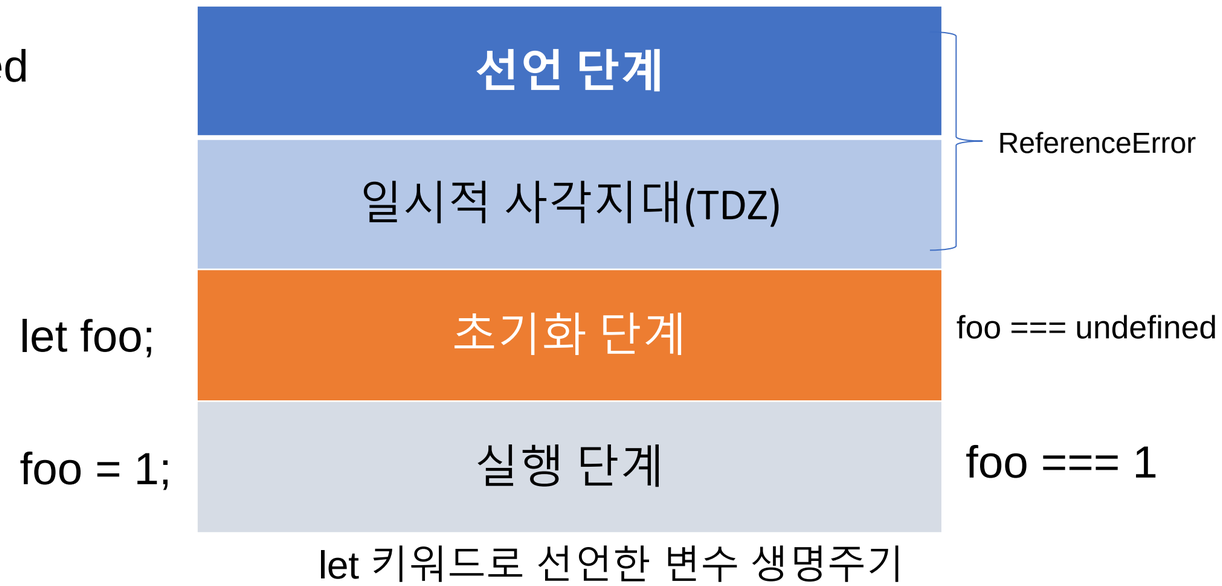
const 키워드 특징

- 선언과 초기화: 반드시 선언과 동시에 초기화 해야 함
  - 블록 레벨 스코프를 가짐
  - **변수 호이스팅 동작하지 않음**
- 재할당 금지

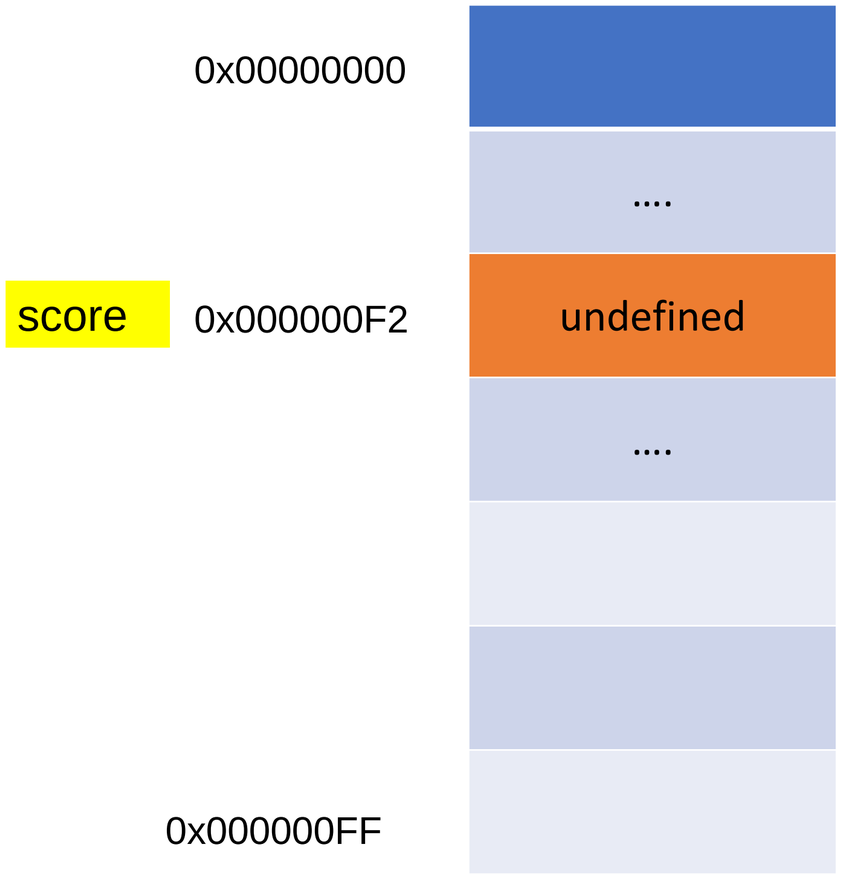
연산자와 제어문

- 자바스크립트 엔진은 변수 선언을 다음과 같은 2단계에 거쳐 수행한다.
  - 생성단계
    - 선언 단계
    - 초기화 단계
  - 실행단계

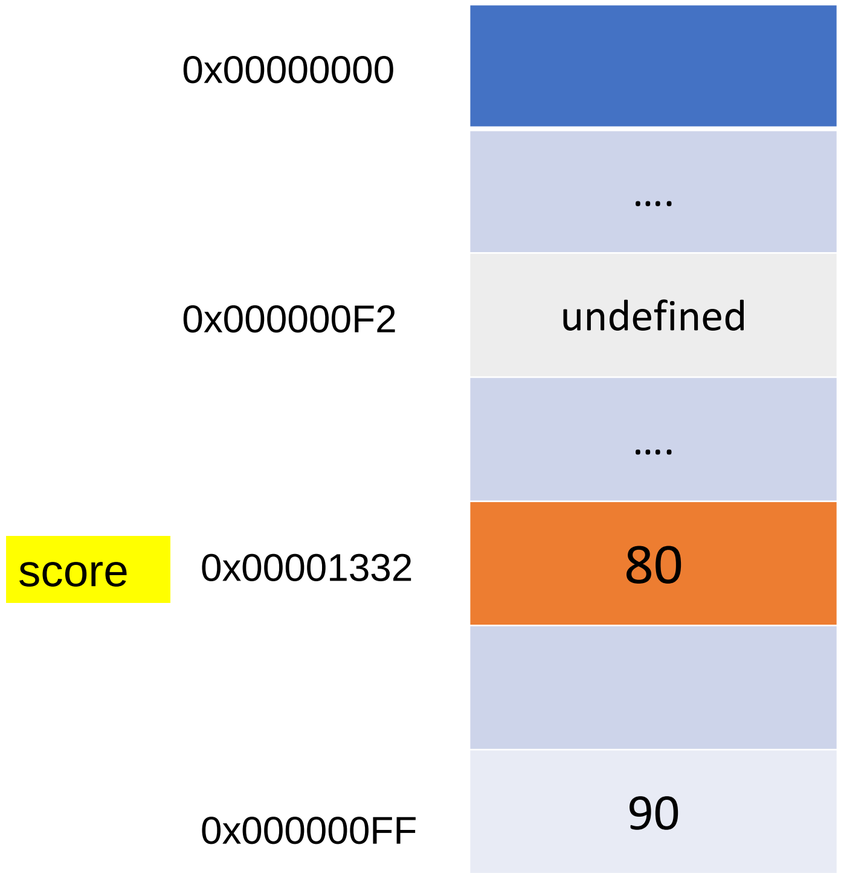




변수, 표현식과 문



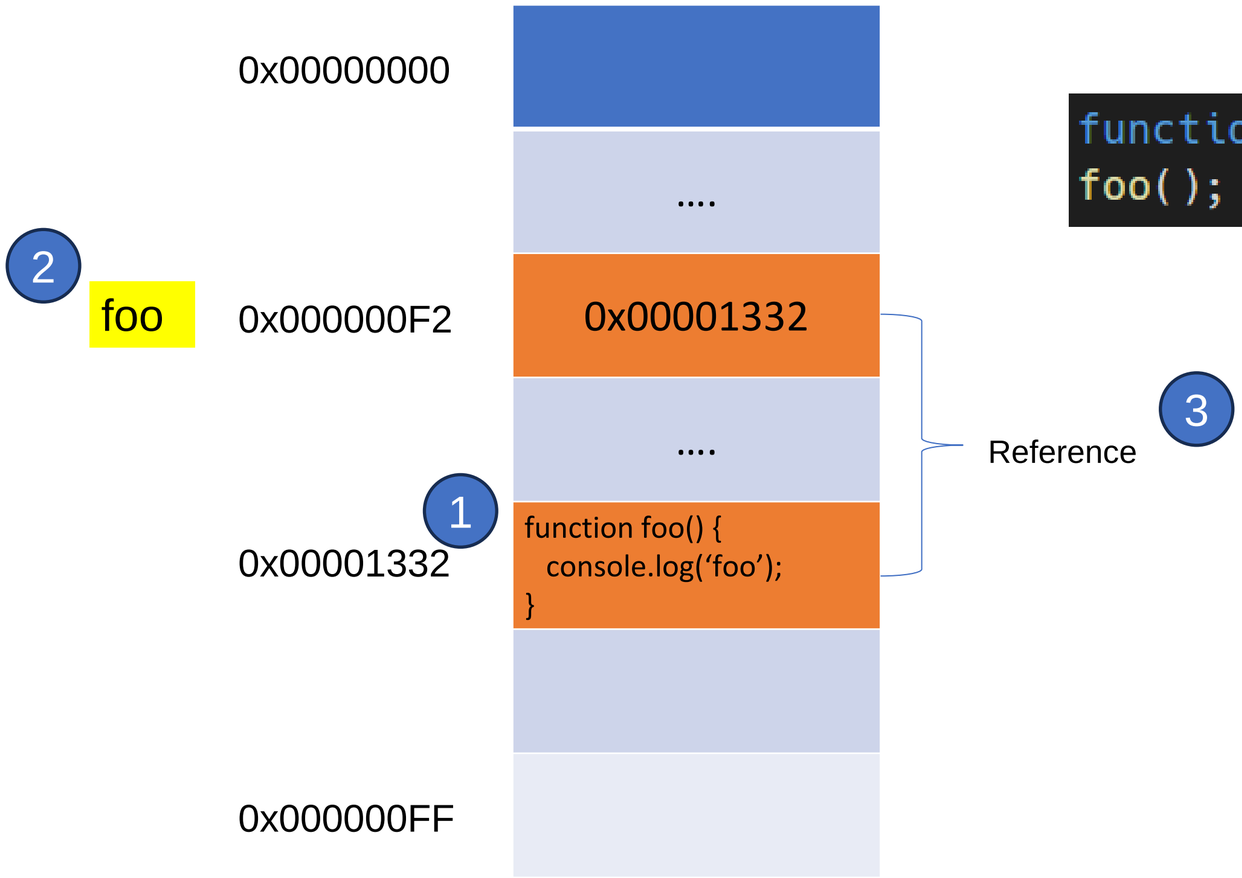

```
var score;  // ① 변수 선언
```




```
score = 80; // ② 값의 할당
```

자바스크립트 문법

함수




```
function foo() { console.log('foo'); }
foo(); // foo
```

함수

```
// 함수 참조
console.dir(add); // ƒ add(x, y)
console.dir(sub); // undefined

// 함수 호출
console.log(add(2, 5)); // 7
console.log(sub(2, 5)); // TypeError: sub is not a function

// 함수 선언문
function add(x, y) {
  return x + y;
}

// 함수 표현식
var sub = function (x, y) {
  return x - y;
};
```

함수 선언문이 코드의 선두로 끌어 올려진 것처럼 동작하는 것을 **함수 호이스팅** 이라고 한다.

변수 호이스팅: var 키워드로 선언된 변수는 undefine로 초기화 됨

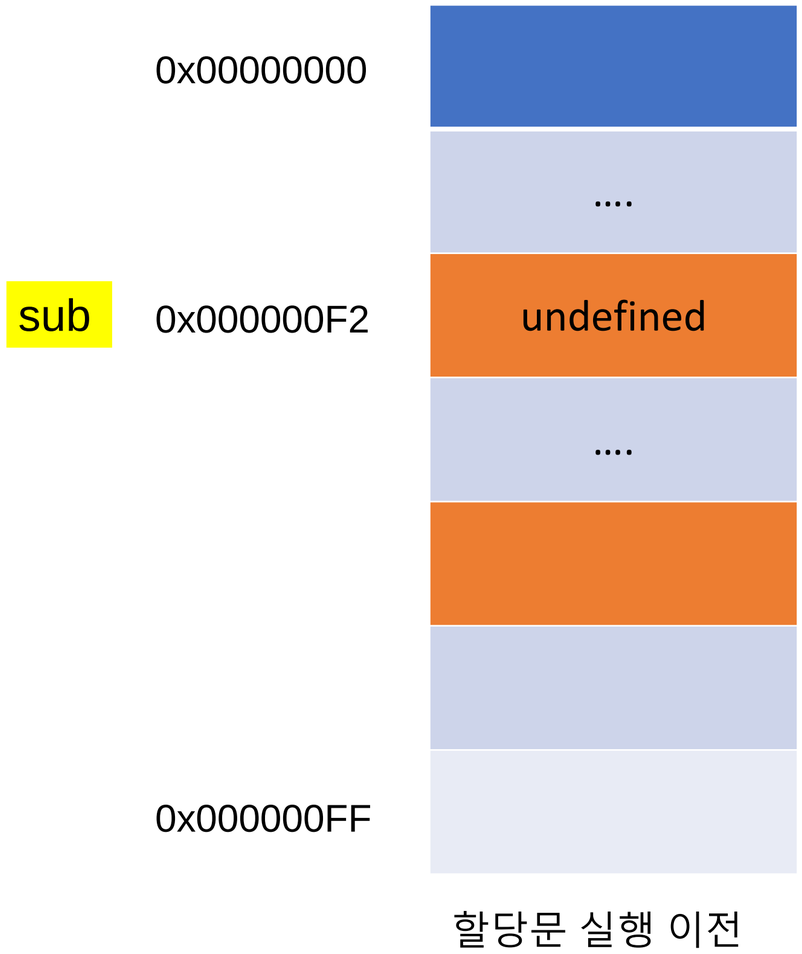
vs

함수 호이스팅: 함수 선언문으로 암묵적으로 생성된 식별자는 함수 객체로 초기화 됨

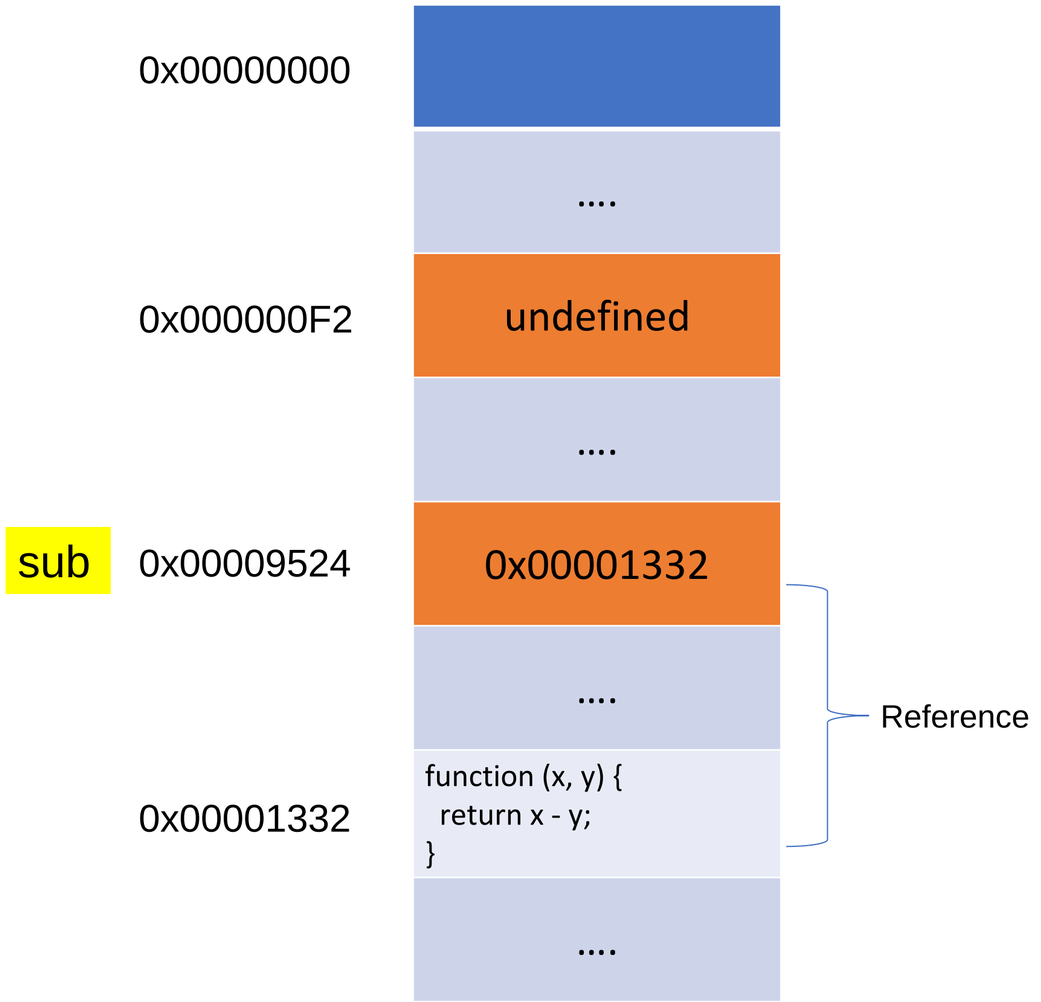
자바스크립트 문법

함수

```
// 함수 표현식
var sub = function (x, y) {
  return x - y;
};
```



할당문 실행 이전



할당문 실행 이후

02

타입 변환

타입 변환

모든 값은 타입이 있다!

값의 타입을 변경할 수 있음!

방법은 2가지

- 명시적 타입 변환
- 암묵적 타입 변환

```
//Explicit vs. Implicit Coercion in JS

let a = 100;

let b = a + '';  //IMPLICIT COERCION
console.log(b); //'100'

let c = String( a ); //EXPLICIT COERCION
console.log( c ); //'100'
```

타입 변환

변환 종류

- 문자열 타입으로 변환
- 숫자 타입으로 변환
- 불리언 타입으로 변환
  - false
  - undefined
  - null
  - 0, -0,
  - NaN
  - ‘’(빈문자열)

타입 변환

명시적 타입 변환 vs 암묵적 타입 변환

- 명시적 타입 변환
  - 타입을 변경하겠다는 개발자의 의지가 코드에 명백히 드러남
- 암묵적 타입 변환
  - 드러나지 않게 타입이 자동으로 변경됨
- 더 나은 것이나 나쁜 것은 없다, 둘 다 필요!
  - 중요한 것은 코드를 예측할 수 있어야 함

03

원시 값과 객체의 비교

데이터 타입

- 자바스크립트의 모든 값은 데이터 타입을 가짐, 총 7개의 데이터 타입 제공

구분	데이터 타입	설명
원시 타입	숫자 타입	숫자. 정수와 실수 구분 없이 하나의 숫자 타입만 존재
	문자열 타입	문자열
	불리언(Boolean) 타입	참(true)과 거짓(false)
	undefined 타입	var 키워드로 선언된 변수에 암묵적으로 할당되는 값
	null 타입	값이 없다는 것을 의도적으로 명시할 때 사용하는 값
	심벌(symbol) 타입	ES6에서 추가된 7번째 타입
객체 타입		객체, 함수, 배열 등

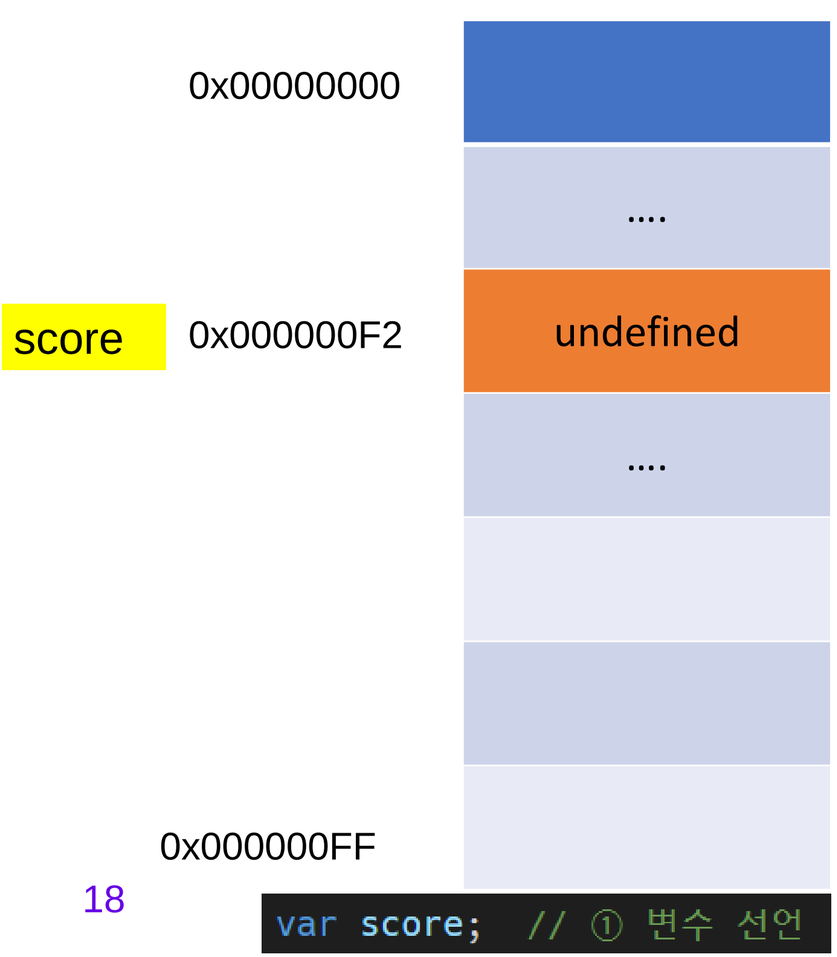
원시 값과 객체의 비교

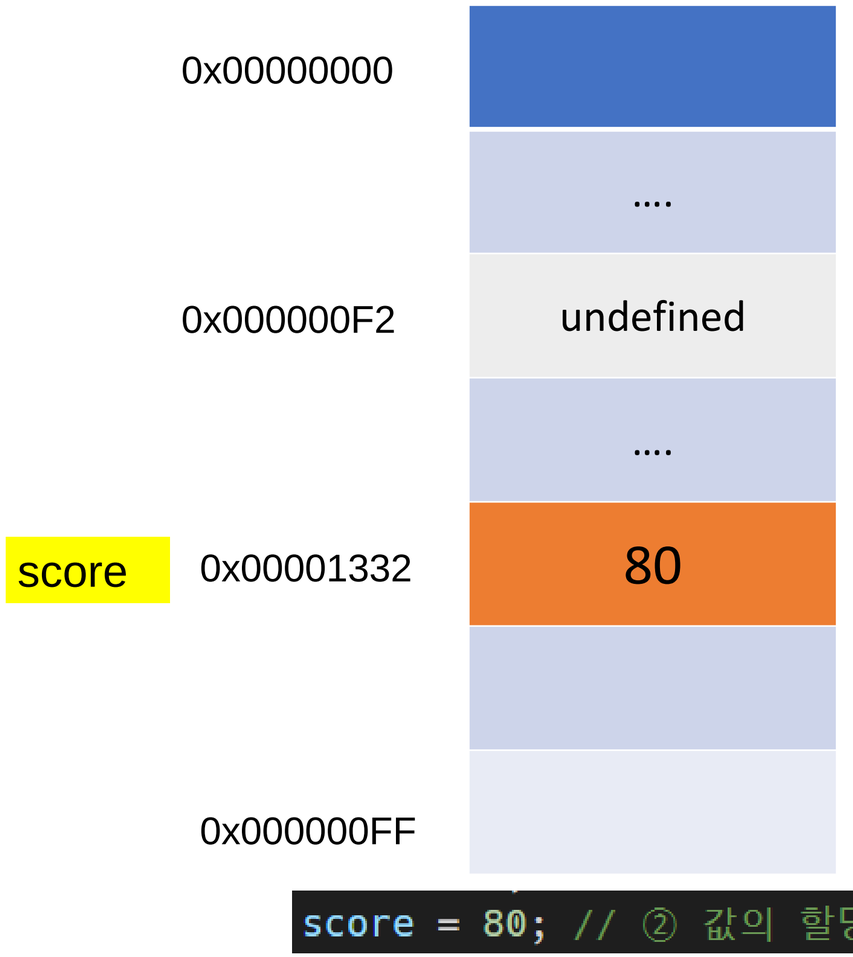
데이터 타입은 원시 타입과 객체 타입으로 구분 가능

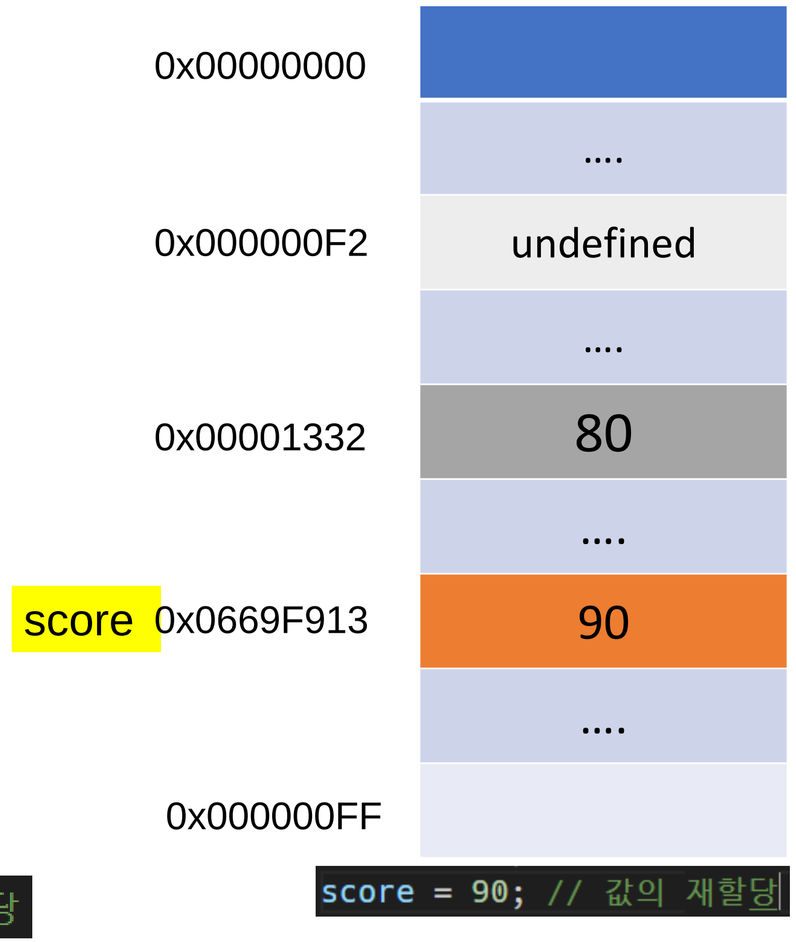
- 원시 타입의 값: 변경 불가능한 값
- 객체 타입의 값: 변경 가능한 값

원시 값과 객체의 비교

- 원시타입의 값: 변경 불가능한 값 (변수가 변경 불가능하다는게 아니다!!)

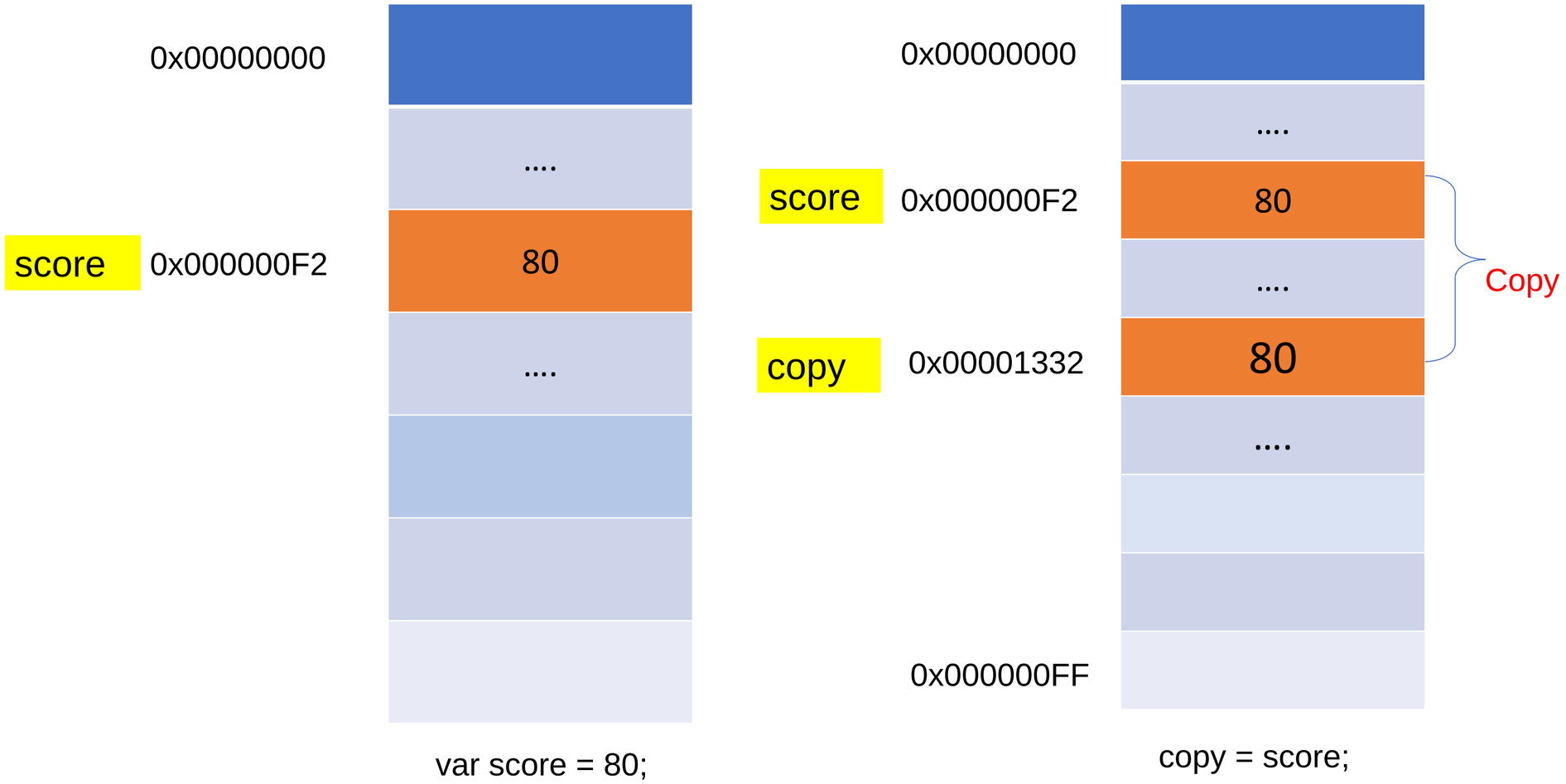






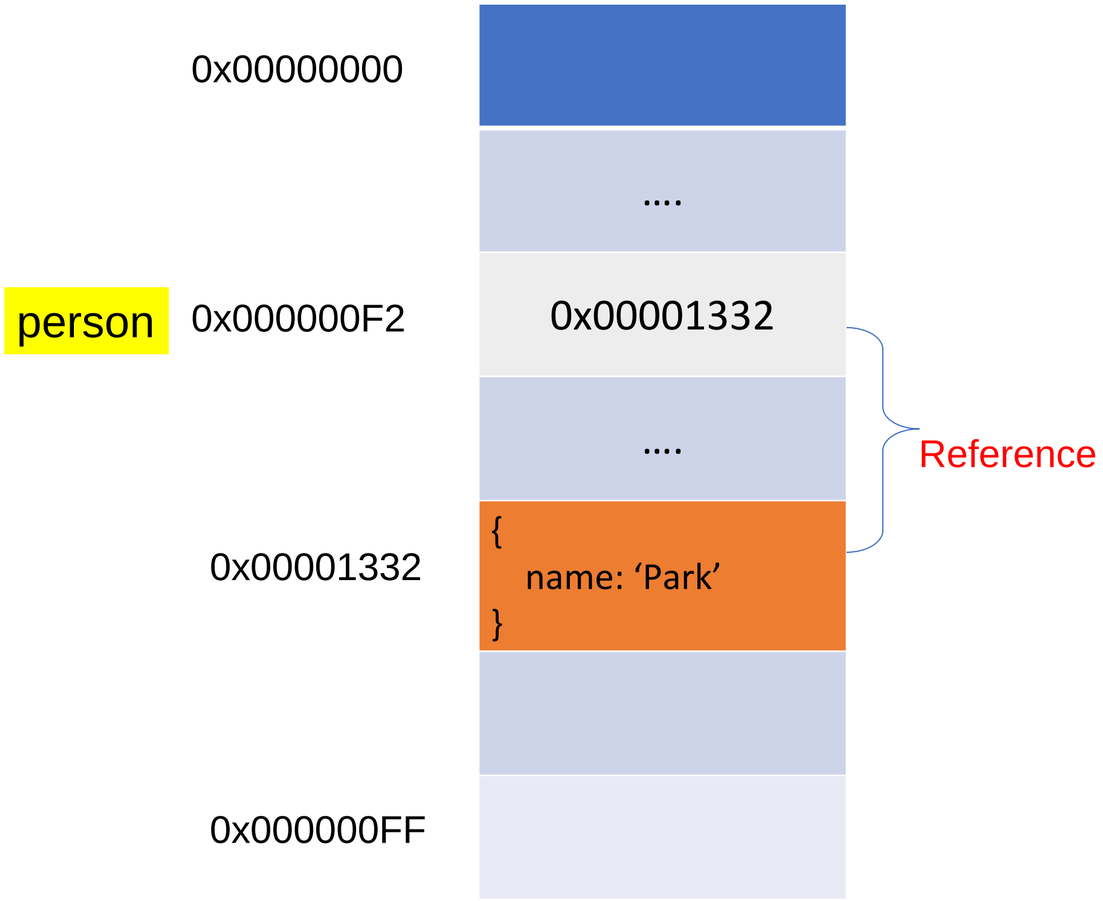
원시 값과 객체의 비교

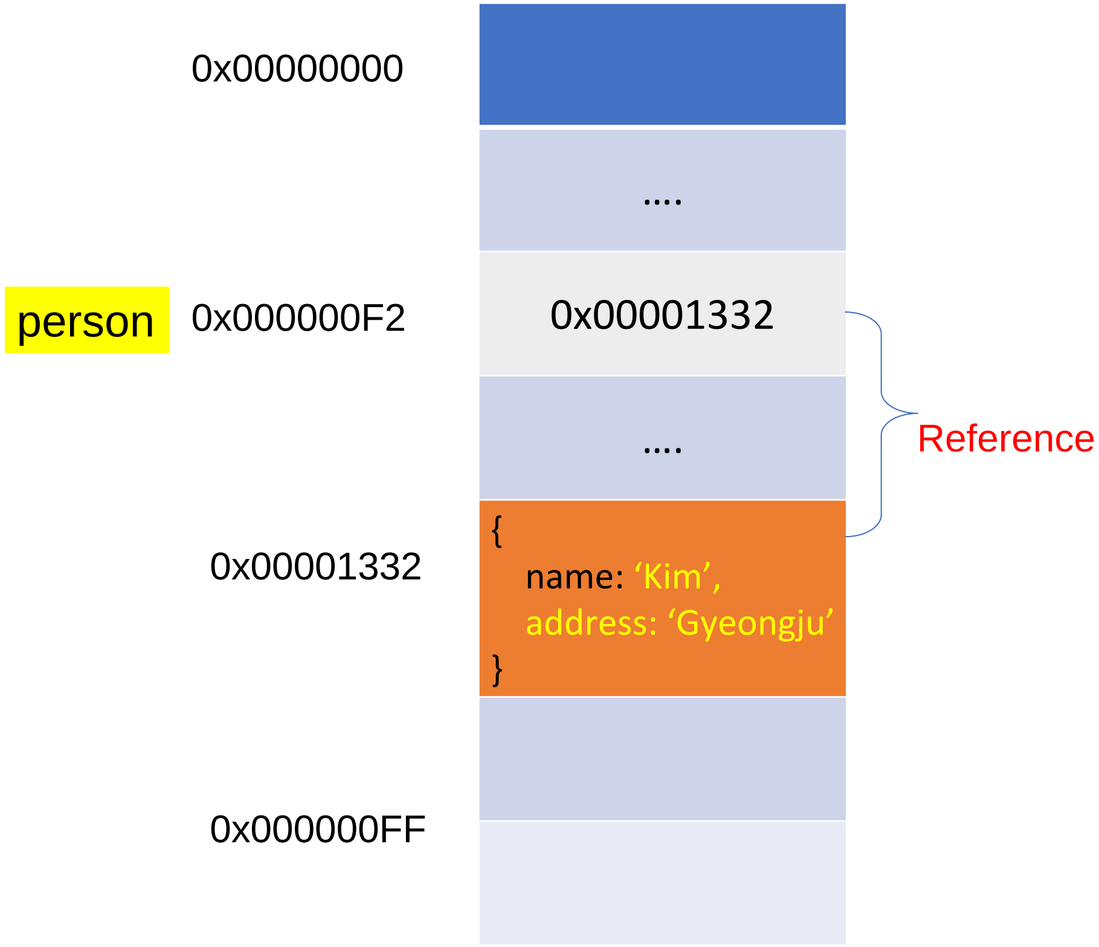
- 값에 의한 전달



원시 값과 객체의 비교

- 객체타입의 값: 변경 가능한 값

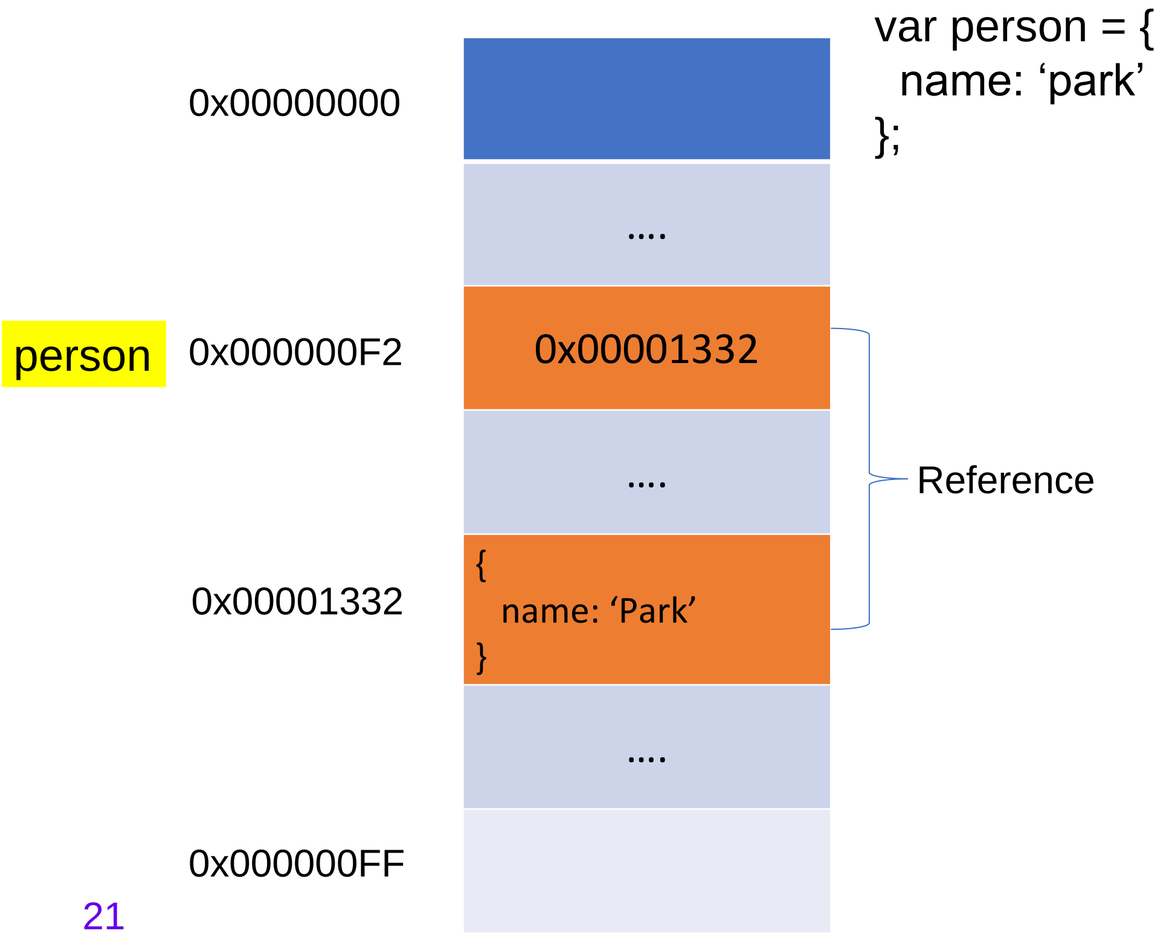


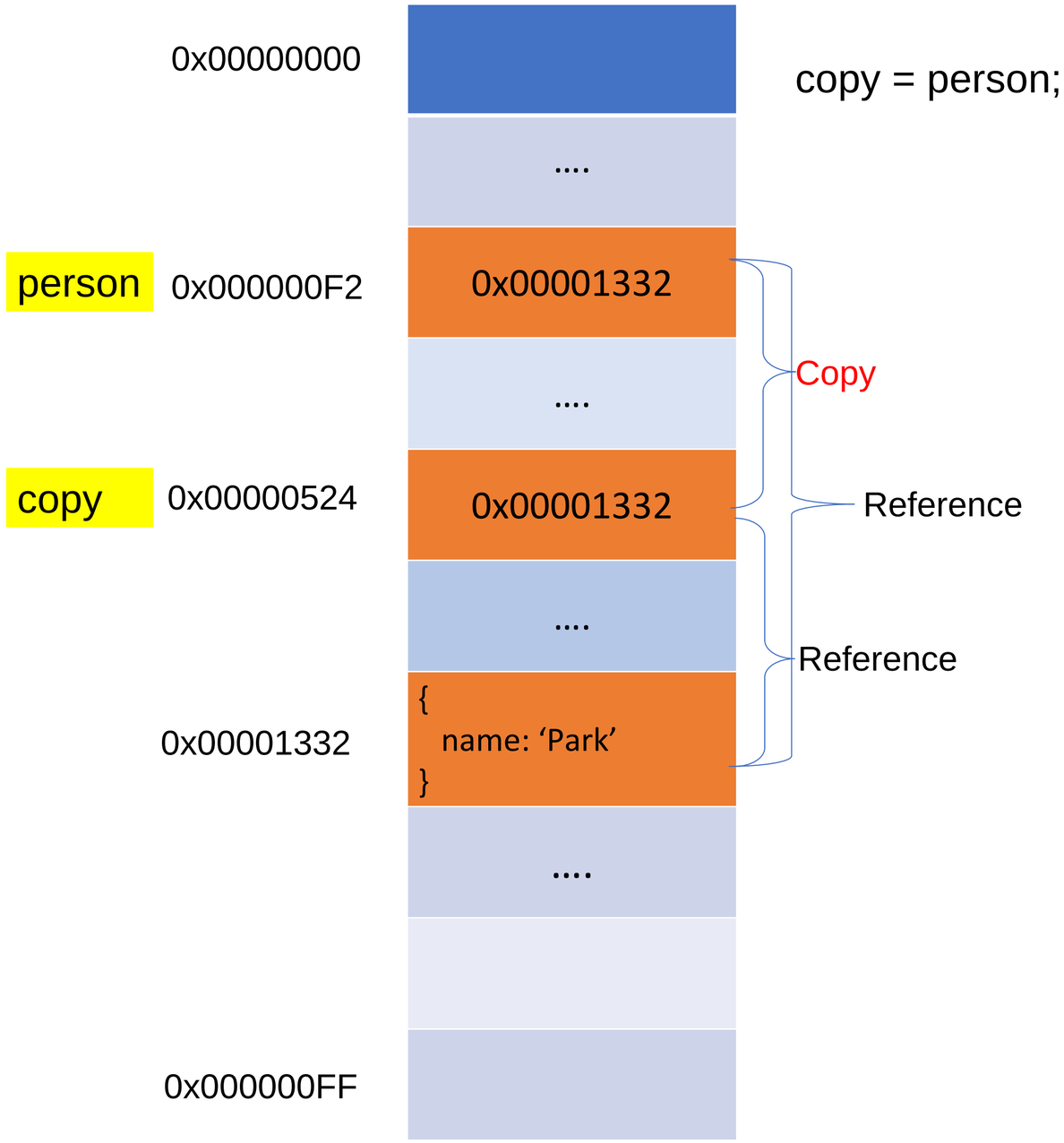


자바스크립트 문법

원시 값과 객체의 비교

- 참조에 의한 전달





04

함수 심화내용

함수 심화내용

- 함수는 일급 객체다
  - 다음 조건을 만족하면 일급 객체라고 한다.
    - 무명의 리터럴로 생성할 수 있다.
    - 변수나 자료구조(객체, 배열 등)에 저장할 수 있다.
    - 함수의 매개변수에 전달할 수 있다.
    - 함수의 반환값으로 사용할 수 있다.

함수 심화내용

- 화살표 함수
  - 기존의 함수 정의 보다 간략하게 함수를 정의하는 것
  - 함수 선언문으로 정의할 수 없고 함수 표현식으로 정의해야 한다.
- 재귀 함수
  - 자기 자신을 호출하는 함수
- 즉시 실행 함수
  - 함수 정의와 동시에 즉시 호출되는 함수

함수 심화내용

- 콜백 함수
  - 함수에 파라미터로 들어가는 함수
  - 콜백 함수는 순차적으로 실행하기 위한 용도로 많이 쓰인다.

05

배열 고차 함수(feat. reduce)

배열 고차 함수(feat. reduce)

- reduce
  - 자바스크립트는 다양한 고차함수를 지원한다, 특히 배열에서 사용할 수 있는 고차함수 중 위 3개는 특히 유용하므로 잘 알아두면 좋다
  - reduce
    - 자신을 호출한 배열의 모든 요소를 순회하며 인수로 전달받은 콜백 함수를 반복 호출한다. 그리고 콜백 함수의 반환값을 다음 순회 시 콜백 함수의 첫 번째 인수로 전달하면서 콜백 함수를 호출하여 하나의 결과값으로 만들어 반환한다.
    - arr.reduce((accumulator, currentValue, currentIndex, array) => { … } [, initialValue])

배열 고차 함수(feat. reduce)

`reduce()`

`reduce()` 메서드는 배열의 각 요소에 대해 주어진 리듀서(reducer) 함수를 실행하고, 하나의 결과값을 반환합니다.

리듀서 함수

리듀서 함수는 네 개의 매개변수를 갖습니다.

- 누적 값 (acc)
- 현재 값 (cur)
- 현재 인덱스 (idx)
- 원본 배열 (src)

리듀서 함수의 반환 값은 누적 값에 할당되고 최종 결과는 이 누적 값이 됩니다.

```
// 누적합 구하기
const numbers = [1, 2, 3, 4, 5, 6, 7];

const result = numbers.reduce((acc, cur) => acc + cur);

console.log(result);
```

배열 고차 함수(feat. reduce)

- reduce

  - reduce의 동작 방식을 표로 나타내면 아래와 같다.

구분	콜백 함수에 전달되는 인수				콜백함수의 반환값
	accumulator	currentValue	index	array	
첫 번째 순회	0 (초기값)	1	0	[1,2,3,4]	1 (accumulator + currentValue)
두 번째 순회	1	2	1	[1,2,3,4]	3 (accumulator + currentValue)
세 번째 순회	3	3	2	[1,2,3,4]	6 (accumulator + currentValue)
네 번째 순회	6	4	3	[1,2,3,4]	10 (accumulator + currentValue)

06

전개 문법

전개 문법

- 하나로 뭉쳐 있는 여러 값들의 집합을 펼쳐서 개별적인 값들의 목록으로 나타내줌
  - 전개문법 적용대상
    - Array
    - String
    - Map
    - DOM 컬렉션
      - NodeList
      - HTMLCollection
    - arguments

전개 문법

- 함수에서 사용하는 경우
- 배열에서 사용하는 경우
- 객체에서 사용하는 경우

전개 문법 (feat. Rest 파라미터)

- Rest 파라미터
  - 함수에 전달된 인수 목록을 배열로 전달받음
  - 단 하나만 선언 가능
  - 반드시 마지막 파라미터여야 함

07

this 란?

this

- this는 자신이 속한 객체 또는 자신이 생성할 인스턴스를 가리키는 자기 참조 변수
  - this를 통해 자신이 속한 객체 또는 자신이 생성할 인스턴스의 프로퍼티나 메서드를 참조할 수 있음
  - this가 가리키는 값은 함수 호출 방식에 의해 동적으로 결정된다

```
const circle = {
  radius: 5,
  getDiameter() {
    // 이 메서드가 자신이 속한 객체의 프로퍼티나 다른 메서드를 참조하려면
    // 자신이 속한 객체인 circle을 참조할 수 있어야 한다.
    // 하지만 자기를 재귀적으로 참조하는 방식은 바람직하지 않음
    return 2 * circle.radius;
  }
};

console.log(circle.getDiameter()); // 10
```

this

- this는 어디서 호출하느냐에 따라 값이 달라진다

함수 호출 방식	this가 가리키는 값(this 바인딩)
일반 함수로서 호출	전역 객체
메서드로서 호출	메서드를 호출한 객체(마침표 앞의 객체)
생성자 함수로서 호출	생성자 함수가 (미래에) 생성할 인스턴스
apply/call/bind 메서드에 의한 간접 호출	apply/call/bind 메서드에 첫번째 인수로 전달한 객체

this

- 화살표 함수 this
  - 화살표 함수의 this 에 대해서 알아보자
    - 화살표 함수의 this는 일반 함수의 this와 다르게 동작한다.
    - Lexical this
      - 화살표 함수 내부에서 this를 참조하면 상위 스코프의 this를 참조한다.

참고

- 모던 자바스크립트 Deep Dive
- 짐코딩 CODING GYM
- 코딩애플
- https://ko.javascript.info/dom-navigation