자료형

\* 기본형 자료형

-Boolean(논리적인 요소) : true, false

-null(빈 값) : js는 대소문자를 구분하므로 null, Null이 다름

-undefined(값을 할당하지 않는 변수가 가지는 값)

-Number 숫자형으로 정수와 부동 소수점, 무한대 및 NaN(숫자가 아님)값을 포함한다.

-String(JavaScript에서는 char형이 존재하지 않음): 문자열

-Symbol(EcmaScript 6에 도입) 인스턴스가 고유하고 불변인 데이터형

Ex) let Sym1 = Symbol(“Sym”)

let Sym2 = Symbol(“Sym”)

console.log(Sym1==Sym2) // return false; (타입과 값이 동일해도 Symbol을 서로 다른 객체로 취급)

\* Object(객체) : 객체는 식별자로 참조할 수 있는 메모리에 있는 값을 뜻한다.

Ex)

const junwan = {

    name : '김준완',

    age : 27,

    gender : '남자',

    location : '인천'

}

    console.log(junwan);

junwan = 식별자 / name,age,gender,location = key / ‘김준완’, 27, ‘남자’, ‘인천’ = value

Var : 변수를 선언. 추가로 동시에 값을 초기화, 변수 재선언 가능, 변수 재할당 가능, var의 경우 변수 선언과 초기화를 동시에 하기 때문에 호이스팅이 가능

Let : 블록 범위(scope) 지역 변수를 선언. 추가로 동시에 값을 초기화, 변수 재선언 불가능, 변수 재할당 가능, let과 const는 변수와 선언이 분리되어 있어서 호이스팅 불가능

Const : 블록 범위 읽기 전용 상수를 선언, 변수 재선언 불가능, 변수 재할당 불가능

Var의 단점:

var name='test1';

console.log(name); //test1

var name='test2';

console.log(name); //test2

이처럼 변수를 한번 더 선언했음에도 불구하고 에러가 아닌 각기 다른 값이 출력된다는 단점이 있다.

이를 방지하기 위해 나온게 let

let name='test1';

console.log(name);

let name='test2';

console.log(name);

SyntaxError: Identifier 'name' has already been declared

이 경우 name 변수를 이미 선언해서 에러가 난다. 재선언은 불가하지만 재할당은 가능

Const의 경우 재할당, 재선언 둘다 불가능

const name='test1';

const name='test2';

console.log(name);

재선언시 오류

SyntaxError: Identifier 'name' has already been declared

재할당시 오류

const name='test1';

name='test2';

console.log(name);

TypeError: Assignment to constant variable.