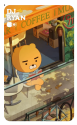

⭐Kakao Cloud(Day 3)⭐

Primitive Value VS Reference(Object)

→ immutable mutable

⭐function → 객체⭐

⭐paramter (지역변수, formal parameter)

⭐argument (실인자, 인수, actual parmeter)

return value (

⭐객체⭐

1. 확장금지(Object.preventExtensions()) → property 추가 불가능❗
2. 밀봉(Object.seal()) → property의 추가, 삭제 불가능❗
3. 얼린다(Object.freeze()) → property의 변경 불가능❗

```
'use strict'; // -> 문제가 되는 코드가 추가되었을 때 Error를 발생시킨다.
const person = {
  name: "Lee",
};

// 객체가 확장이 가능한지? -> property 추가가 가능한지!
console.log(Object.isExtensible(person));

person.age = 20;
console.log(person); // { name: 'Lee', age: 20 }

Object.preventExtensions(person); // 확장을 막는다.

person.addres = "서울"; // 추가 불가능!!

console.log(person); // { name: 'Lee', age: 20 }

Object.seal(person); // 밀봉 시킨다
delete person.name; // 삭제 불가능!!
console.log(person); // { name: 'Lee', age: 20 }

Object.freeze(person); // 얼린다
person.name = "아이유"; // 변경 불가능!!
console.log(person); // { name: 'Lee', age: 20 }
```

⭐생성자 함수에 의한 객체 생성⭐

1. Object 생성자 함수를 이용한 객체 생성.

→ **“new” keyword를 이용해서 생성자 함수 호출 ⇒ instance**

→ return 구문 사용 X

→ 묵시적으로 생성된 instance reference인 **this**가 리턴된다.

→ 객체를 리턴하게되면 망한다. but primitive value를 리턴하게 되면 무시한다.

this ⇒ 1. 일반함수 - window

2. 생성자함수 → 생성된 instance

3. 메소드에서 사용 → 현재 사용하는 객체

⭐함수는 그 자체로 객체⭐

```
const person1 = {};
console.dir(person1);
// 객체 literal을 이용한 객체 생성

const person2 = new Object();
console.dir(person2);
// 생성자 함수를 이용한 객체 생성
// => instance
```

⭐Java Script 대표적인 생성자 함수(built-in)⭐

⇒ **Objcet(),String(),Number(),Boolean(),Function(),Array()... 등등**

⭐Java Script Engine이 코드를 실행하기 위해 기동하면?⭐

1. built-in 객체(생성자 함수 포함)생성
2. 실행환경에 맞는 global객체를 생성 → browser 환경 ⇒ window
3. →

⭐함수객체의 property⭐

→ 일반객체에 비해 더 가지고 있는 Property

1. arguments property는 arguments 유사배열 객체를 property의 value 로 가져요

⇒ 함수내에서 지역변수처럼 사용

2. caller property(비표준) ⇒ 함수 자신을 호출한 함수에 대한 reference
3. length property ⇒ parameter의 개수
4. name property ⇒ 함수의 이름
5. prototype property ⇒ 해당 함수(생성자 함수)가 생성하는 instance가 내부슬롯으로 참조하는 객체를 가리킨다.

⇒ Constructor만 가지고 있다.

⭐prototype chain⭐

→ **객체의 property를 찾을려는 메커니즘**

모든객체는 [[prototype]] 이라는 내부슬롯을 가진다.

→ 저걸 따라가면 상위 prototype 객체가 있다.

→ 객체 생성 방법에 따라 종류가 달라진다.

⭐Rest Parameter(ES6)⭐

arguments 유사배열객체 대신 사용

Rest parameter는 Array로 사용

⇒ 둘 다 가변인자함수를 만들 때 사용, 함수 내에서 이용가능!!

but, arrow function에서는 arguments를 못쓴다. rest parameter만 이용가능

→ Rest parameter ⇒ “. . .”세개로 표현

→ 두 개 사용 불가능, 제일 마지막에 나와야한다

```
function foo(...args) {
  console.log(arguments);
  // [Arguments] { '0': 1, '1': 2, '2': 3, '3': 4, '4': 5 }
  console.log(args);
  // [ 1, 2, 3, 4, 5 ]
  return args.pop(); // restparameter는 배열이기 때문에 pop도 사용가능
}
var result = foo(1, 2, 3, 4, 5);
console.log(result); // 5
```