

2023-04-23

JavaScript는 Prototype 기반 언어이다?

: 프로토타입에 대해서

김도경

JavaScript는 Prototype 기반 언어이다.

이 말이 무슨 뜻일까요?

설명하실 수 있나요?

프로토타입이 무엇일까요?

프로토타입 기반 프로그래밍이란?

객체의 원형인 **프로토타입**을 이용하여 새로운 객체를 만들어내는 프로그래밍 기법이다.

이렇게 만들어진 객체 역시 자기자신의 **프로토타입**을 갖는다.

이 새로운 객체의 원형을 이용하면 또 다른 새로운 객체를 만들어 낼수도 있으며,
이런 구조로 객체를 확장하는 방식을 **프로토타입 기반 프로그래밍**이라고 한다.

그러니까 이게 무슨 말이야

Prototype VS Class

다른 객체지향 언어들과 다르게 JavaScript는 클래스 기반이 아니라 프로토타입 기반의 언어입니다.

프로토타입 기반 OOP(객체 지향 프로그래밍)는

클래스 기반의 OOP와 완전히 상반되는 방식입니다.

그래서 프로토타입을 이해하려면 클래스에 대해서 알아야 한다.

Q. 그럼 우리가 JavaScript 강의때 배웠던 Class는 무엇인가요?



프로토타입 기반의 JavaScript에는 상속 개념이 존재하지 않습니다.

클래스 기반의 다른 언어에 익숙한 많은 개발자들을 혼란스럽게 했고,
따라서 클래스와 비슷하게 동작하게끔 흉내 내는 여러 기법들이 탄생했는데,

이러한 니즈에 따라 결국 ES6에 클래스 문법이 추가되었습니다.

클래스 문법이 추가됐다는 것이지, 자바스크립트가 클래스 기반으로 바뀌었다는 것은 아닙니다.

사실 클래스도 함수이며, 기존 프로토타입 기반 패턴의 Syntactic sugar라고 볼 수 있습니다.

(Deep Dive 등의 서적에서 더 자세하게 다루고 있습니다.)

Prototype VS Class

플라톤 이데아 이론 - 아리스토텔레스

분류 개념(Classification)이 등장합니다.

(Class 키워드는 분류(Classification)에서 왔습니다.)

-> 개체의 속성이 동일한 경우 개체 그룹이 같은 범주에 속한다.
범주는 정의 구별의 합이다.

=> 즉, 공통된 속성으로 분류하고, 일반화 과정을 통해 클래스로 추상화됩니다.

Prototype VS Class

꺾

사과

바나나

Prototype VS Class



```
class Fruit {  
  ...  
}  
Fruit apple = new Fruit();
```

Class Fruit

추상적인 개념.

코드상으로만 존재하지 실제 메모리상으로는 존재하지 않습니다.

(Heap 메모리 기준)

new 키워드를 사용, new Fruit 하는 순간

구체적으로 존재(인스턴스화)하게 된다.

Prototype VS Class

비트겐슈타인

분류(Classification)?

NONO

공유 속성의 관점에서 정의하기 어려운 개념이 있다.

(사실상 올바른 분류란 없다)

Prototype VS Class

비트겐슈타인 - 의미 사용 이론

사용(Use)에 의해 의미(Meaning)가 결정된다.

-> 단어의 쓰임새가 곧 의미가 된다.

=> 즉, 단어의 '진정한 본래의 의미'란 존재하지 않고

‘상황과 맥락(Context)에 의해서 결정된다’ 라고 주장

Prototype VS Class

비트겐슈타인 - 의미 사용 이론

EX) 누군가 **벽돌!** 이라 외쳤을 때 상황마다 그 의미가 달라진다.

1. (벽돌이 필요할 때) : 벽돌을 달라
2. (벽돌로 보수해야 할 때) : 벽돌을 채우라
3. (벽돌이 떨어질 때) : 벽돌을 피해라

Prototype VS Class

비트겐슈타인 - 가족 유사성

가족 유사성

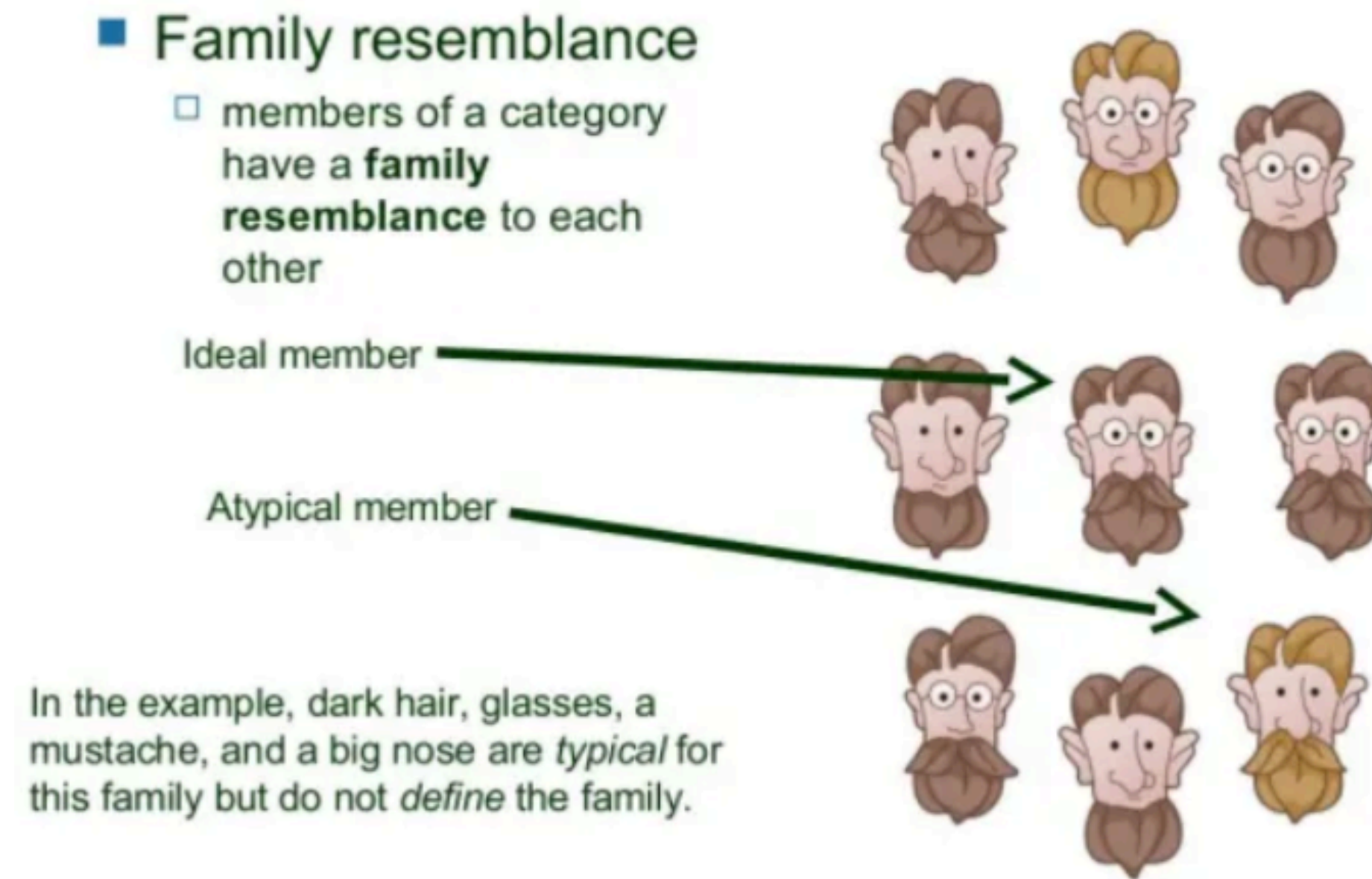
: 가족 구성원 모두에게 공통적인 특성은 없지만,

부분적으로 유사한 특성들이 연결되어 있어

가족 구성원들을 하나의 가족으로 인정하게 만드는 유사성

Prototype VS Class

비트겐슈타인 - 가족 유사성



머리색이 다르고, 코의 크기가 다르고, 안경을 썼는지, 수염의 정도, 귀의 크기, 등 다른 부분이 많지만
그런데도 저 사람들을 보면 “아하, 가족이구나” 하며 인정한다.

Prototype VS Class

Eleanor Rosch

1970년경 철학자 Eleanor Rosch에 의해

프로토타입 이론(Prototype theory)로 정리된다.

Prototype VS Class

Eleanor Rosch의 실험

- 실험 참가자들에게 여러 범주 구성원(사과, 코코넛, 오렌지)의 속성을 적어보라고 함
- 각 범주 구성원에 대해 범주의 다른 구성원과 공유하는 속성의 개수를 도출함
- 사과, 오렌지 : 2점(둥글다. 즈이 있다.)
- 코코넛 : 1점(둥글다)

점수가 높을수록 **가족 유사성**이 높다고 볼 수 있다.

사과, 오렌지 등은 전형적인 것으로 보되,
코코넛 같은 것은 그 중 가장 비전형적인 것으로 볼 수 있다.

Prototype VS Class

Eleanor Rosch의 실험

인간은 사물을 분류할 때 자연스럽게 가장 유사성 높은 것 순서대로 등급을 매긴다.

(유사성 높은 것을 기준으로 등급을 매기는 것)

이렇게 분류했을 때 가장 높은 등급을 가진 것

=> 원형(Prototype)

Prototype VS Class

Eleanor Rosch의 실험

원형 (Prototype)

객체는 ‘정의’로부터 분류되는 것이 아니라
가장 좋은 보기(Prototype)로 부터 범주화 된다.

원형에서 가장 먼, 비전형적인 새

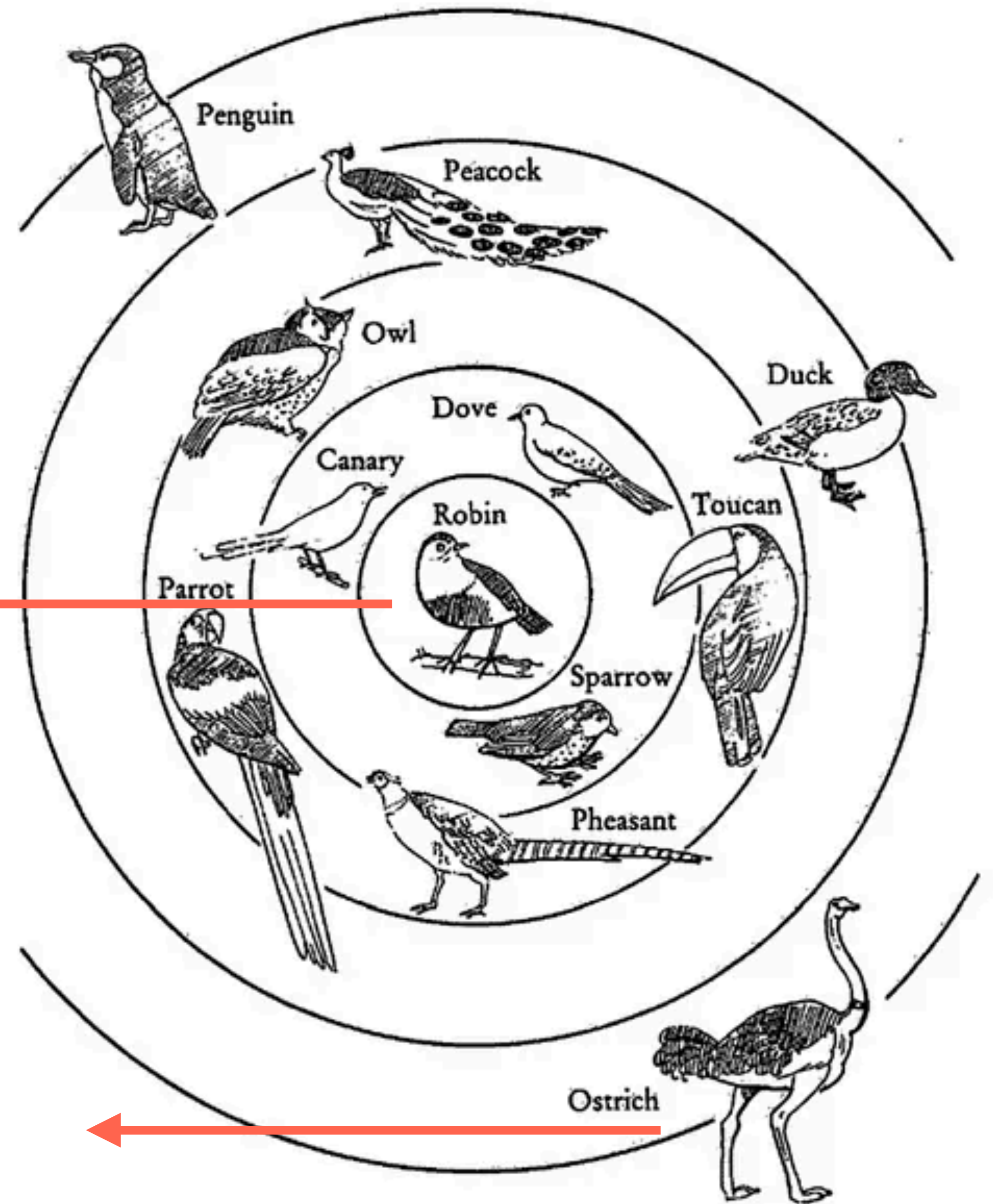


Figure . 1 Birdiness rankings

Prototype VS Class

중간 정리

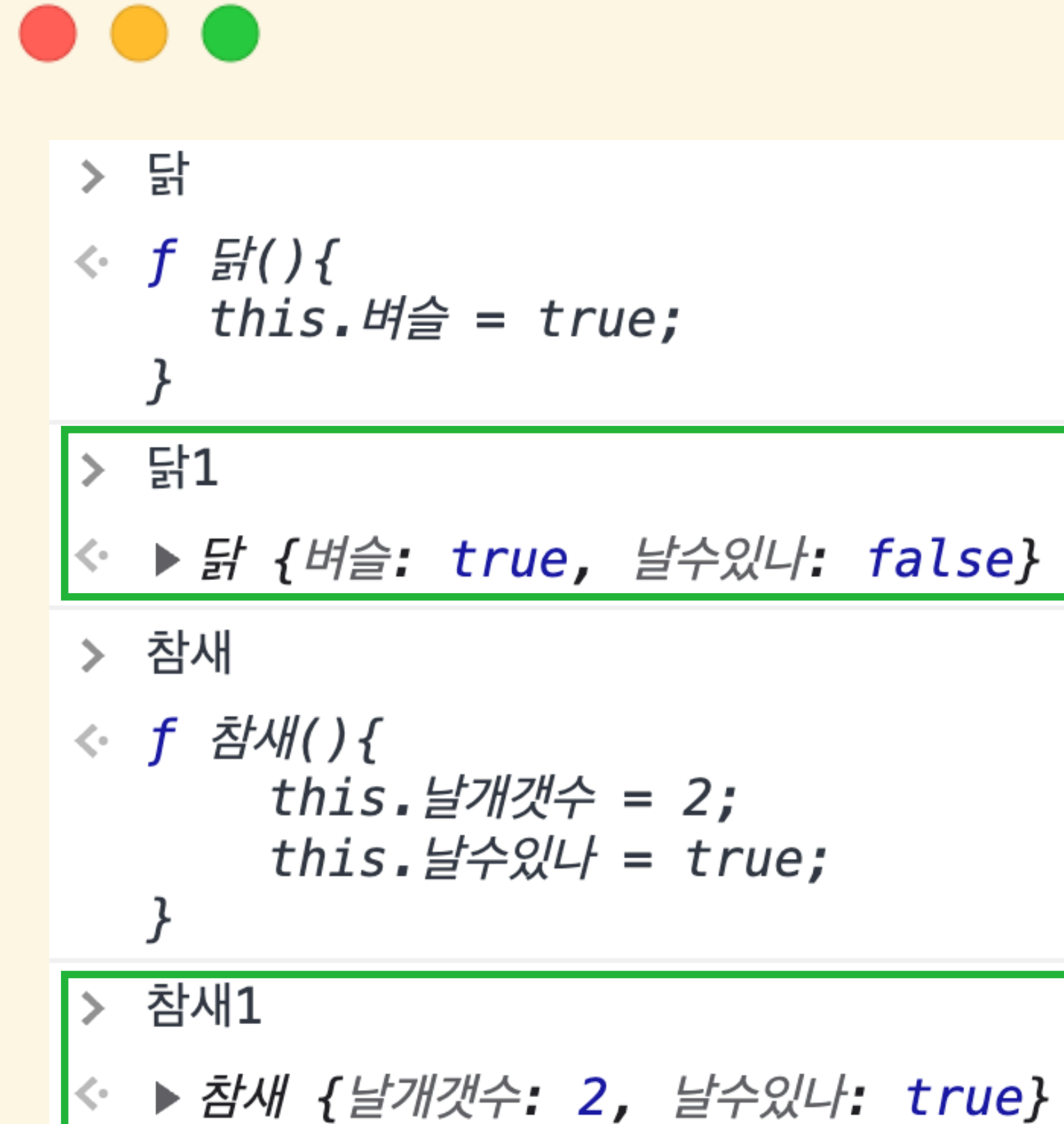
의미사용이론 -> 같은 단어라 할지라도 누가 어떤 상황에 접했나에 따라 의미가 달라진다.

=> 상황, 문맥(Context)에 따라 '범주', 즉 '의미'가 달라진다.

가족 유사성 -> 현실에 존재하는 것 중 가장 좋은 본보기를 원형(Prototype)으로 선택한다.

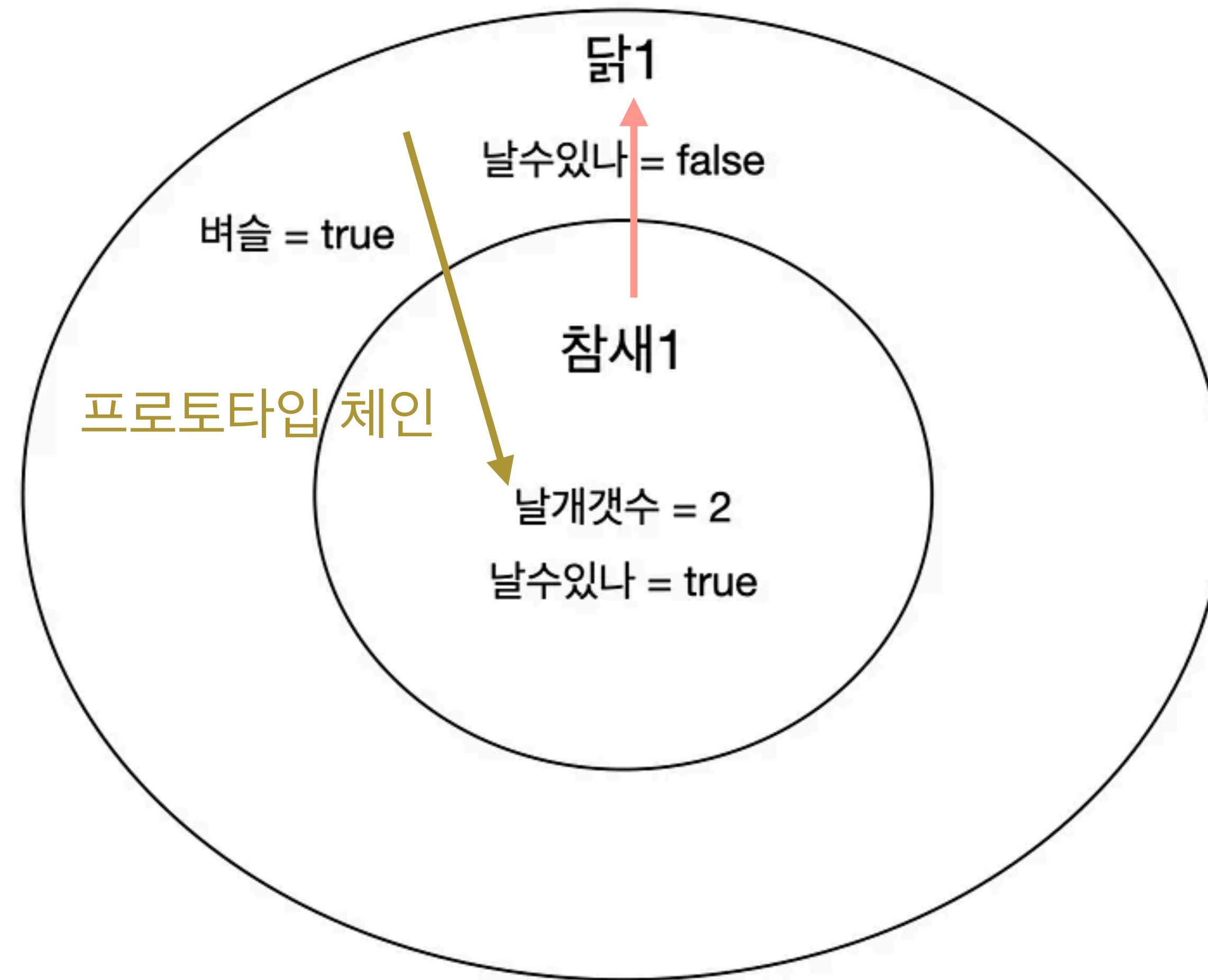
Prototype VS Class

```
function 참새(){  
  this.날개갯수 = 2;  
  this.날수있나 = true;  
}  
  
const 참새1 = new 참새();  
  
console.log("참새의 날개 갯수 : ", 참새1.날개갯수); // 2  
  
function 닭(){  
  this.벼슬 = true;  
}  
  
닭.prototype = 참새1; // reference(오른쪽이 인스턴스인 점 주목)  
const 닭1 = new 닭();  
console.log("닭1 날개 : ", 닭1.날개갯수, ", 날수있나? ", 닭1.날수있나); // 2, true  
닭1.날수있나 = false;  
console.log("다시 물어본다. 닭1은 날 수 있나? :", 닭1.날수있나); // false
```



```
> 닭  
< f 닭(){  
  this.벼슬 = true;  
}  
  
> 닭1  
< ▶ 닭 {벼슬: true, 날수있나: false}  
  
> 참새  
< f 참새(){  
  this.날개갯수 = 2;  
  this.날수있나 = true;  
}  
  
> 참새1  
< ▶ 참새 {날개갯수: 2, 날수있나: true}
```

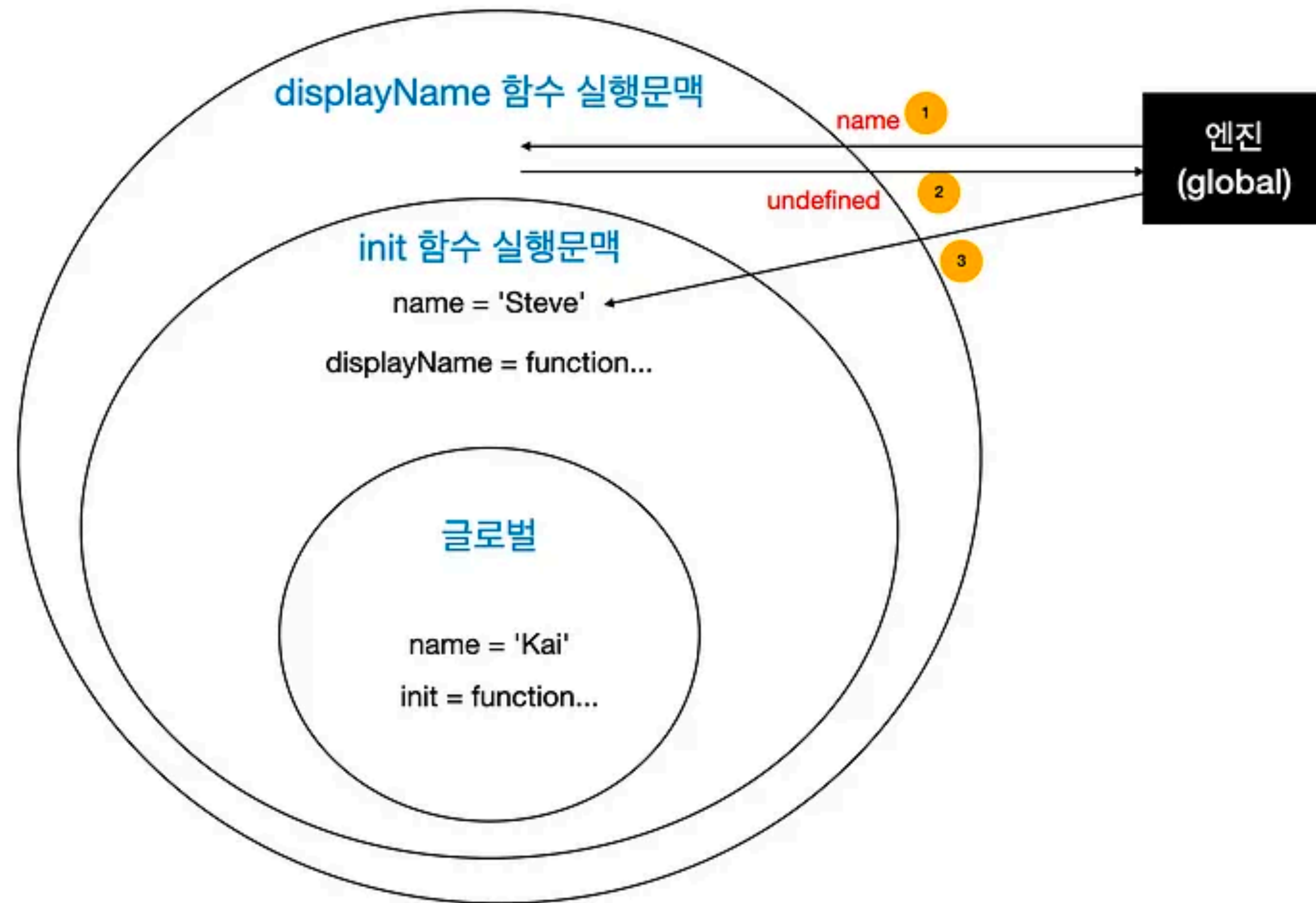
Prototype VS Class



Prototype VS Class

```
// 전역 실행문맥 생성. 전체 정의(name, init) 호이스팅
var name = 'Kai';
init(); // init 실행문맥 생성. 내부 정의(name, displayName) 호이스팅
function init() {
  var name = "Steve";
  function displayName() {
    console.log(name); // 현재 실행문맥 내에 정의된게 없으니 outer 로 chain
    // var name = 'troll?'; // 주석 해제되면 호이스팅
  }
  displayName(); // displayName 실행문맥 생성. 내부 정의 호이스팅.
}
```

Prototype VS Class



Prototype VS Class



의미사용이론

-> 같은 단어라 할지라도 누가 어떤 상황에 접했나에 따라 의미가 달라진다.

=> 상황, 문맥(Context)에 따라 '범주', 즉 '의미'가 달라진다.

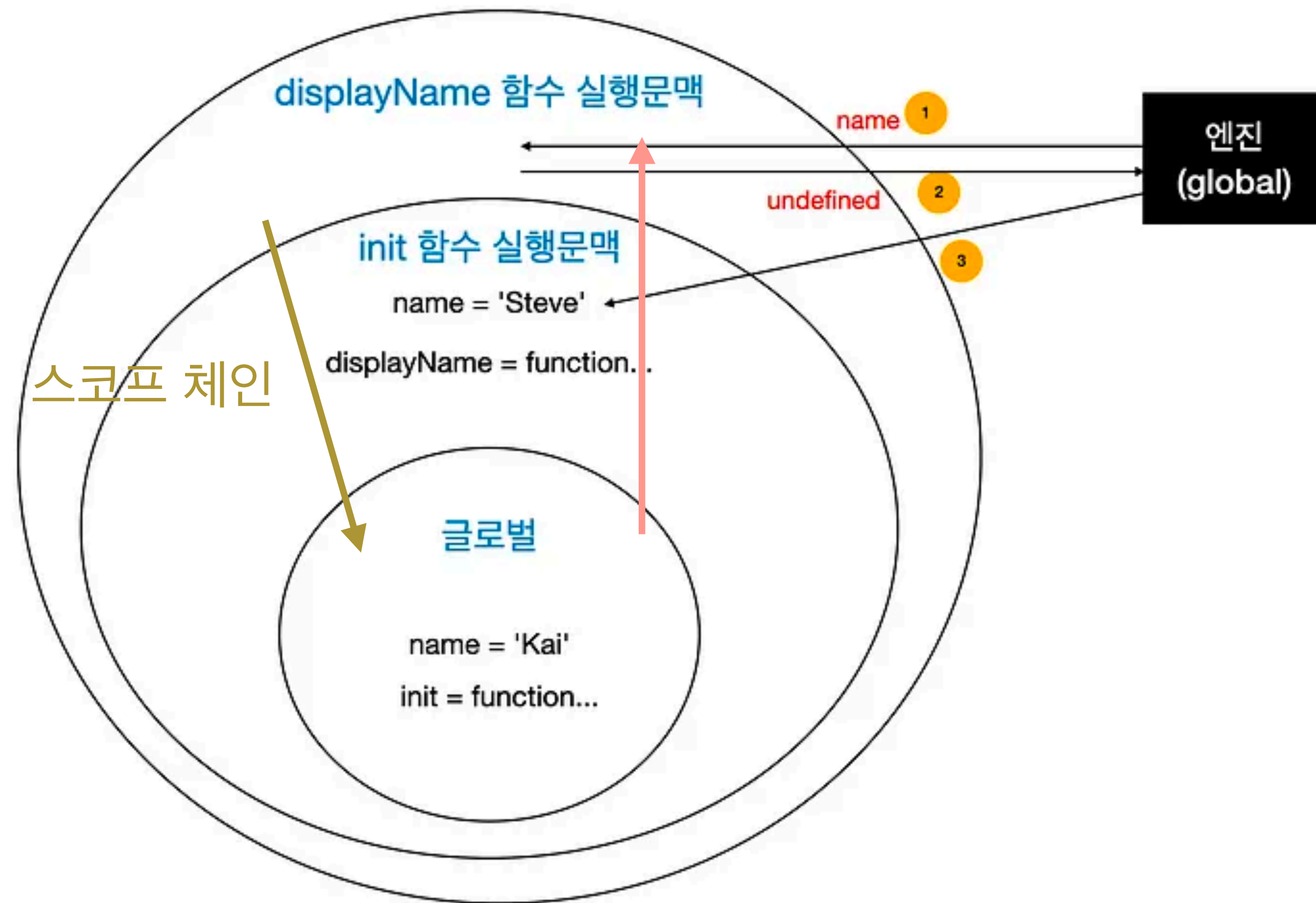
변수의 의미는 그 어휘적인(Lexical),

실행 문맥(Execution Context)에서의 의미가 된다.

-> 그렇기 때문에

동일 범위(실행 문맥)의 모든 선언을 참고(호이스팅)해 의미를 정의한다.

Prototype VS Class



Prototype 사용

```
function Person() {  
  this.eyes = 2;  
  this.nose = 1;  
}  
  
var kim = new Person();  
var park = new Person();  
  
console.log(kim.eyes); // => 2  
console.log(kim.nose); // => 1  
  
console.log(park.eyes); // => 2  
console.log(park.nose); // => 1
```

kim과 park은 eyes와 nose를 공통적으로 가지고 있는데,

메모리에는 eyes와 nose가 **두 개씩 총 4개** 할당됩니다.

객체를 100개 만들면 **200개의 변수**가 메모리에 할당됩니다.

바로 이런 문제를 프로토타입으로 해결할 수 있습니다.

Prototype 사용

```
function Person() {}  
  
Person.prototype.eyes = 2;  
Person.prototype.nose = 1;  
  
var kim = new Person();  
var park = new Person();  
  
console.log(kim.eyes); // => 2  
...
```

Person.prototype이라는 빈 Object가 어딘가에 존재하고,

Person 함수로부터 생성된 객체(kim, park)들은

어딘가에 존재하는 Object에 들어있는 값을 모두 갖다 쓸 수 있습니다.

Prototype 사용

```
function Person() {} // => 함수  
var personObject = new Person(); // => 함수로 객체를 생성
```

```
var obj = {};
```

==

```
var obj = new Object();
```

```
> Object  
◁ function Object() { [native code] }
```

Object도 함수다!

Prototype 사용

1. 해당 함수

관련 내용은

-> **네이버**

Core JavaScript, 모던 자바스크립트 Deep Dive 등에

자세히 (생각보다) 이해하기 쉽게 나와있습니다.

2. 해당 함수



마무리

프로토타입 기반 프로그래밍이란?

객체의 원형인 **프로토타입**을 이용하여 **새로운 객체**를 만들어내는 프로그래밍 기법이다.

이렇게 만들어진 객체 역시 자기자신의 프로토타입을 갖는다.

이 새로운 객체의 원형을 이용하면 또 다른 새로운 객체를 만들어 낼수도 있으며,
이런 구조로 객체를 확장하는 방식을 프로토타입 기반 프로그래밍이라고 한다.

이제는 좀 이해되시나요?

2023-04-23

Q & A

김도경

2023-04-23

감사합니다.

김도경

Reference.

- <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.56.4713&rep=rep1&type=pdf>
- <https://m.blog.naver.com/ms2948/220999019306>
- <https://imnt.tistory.com/207>
- <http://m.blog.daum.net/wise1004-1/6985545>
- <http://zolaist.org/category/과학철학/page/2>
- http://www.aistudy.co.kr/paper/pdf/scientific_lee.pdf
- <https://m.blog.naver.com/PostView.nhn?blogId=linigy&logNo=80002151688&proxyReferer=https:%2F%2Fwww.google.co.kr%2F>
- <https://www.koreascience.or.kr/article/JAKO200916263468706.pdf>
- <https://www.netinbag.com/ko/science/what-is-prototype-theory.html>
- <https://skytreesea.tistory.com/11>
- <https://m.blog.naver.com/PostView.nhn?blogId=ilikepeace8&logNo=221004215876&proxyReferer=&proxyReferer=https:%2F%2Fwww.google.co.kr%2F>
- <https://kaspyx.tistory.com/93>
- <https://namu.wiki/w/루트비히요제프요한비트겐슈타인#s-1>
- <https://www.slideshare.net/vcmlab/cog5-lecppt-chapter08>
- <https://laurabecker.gitlab.io/classes/as/08-semantics.pdf>
- <https://m.blog.naver.com/PostView.nhn?blogId=starsailoris&logNo=220530218923&proxyReferer=https:%2F%2Fwww.google.com%2F>
- https://expertiza.csc.ncsu.edu/index.php/CSC/ECE_517_Fall_2010/ch4_4e_ms

=> <https://medium.com/@limsungmook/%EC%9E%90%EB%B0%94%EC%8A%A4%ED%81%AC%EB%A6%BD%ED%8A%B8%EB%8A%94-%EC%99%9C-%ED%94%84%EB%A1%9C%ED%86%A0%ED%83%80%EC%9E%85%EC%9D%84-%EC%84%A0%ED%83%9D%ED%96%88%EC%9D%84%EA%B9%8C-997f985adb42>

<https://medium.com/@bluesh55/javascript-prototype-%EC%9D%B4%ED%95%B4%ED%95%98%EA%B8%B0-f8e67c286b67>
://insanenhong.kr/post/javascript-prototype/
Book, Core JavaScript