9장 객체

9.1 객체 생성하기

객체란? { key_propertyName: propertyValue }

9.1.1 객체의 생성

객체를 생성하는 3가지 방법

1 객체 리터럴

```
var card = { suite:'하트', rank: "A"  }
```

2 생성자

```
function Card(suit, rank) {
    this.suite = suit;
    this.rank = rank;
}
var card = new Card("하트", "A");
```

3 Object.create

```
function Card(suit, rank) {
    this.suite = suit;
    this.rank = rank;
}
var card = Object.create(Object.prototype, {
    suit: {
        value: "하트",
        writable: true,
        enumerable: true,
        configurable: true,
    },
    rank: {
        value: "A",
        writable: true,
        enumerable: true,
        configurable: true,
    },
});
```

9.1.2 프로토타입

프로토타입 객체 === 함수.prototype (함수의 prototype 프로퍼티가 가리키는 객체를 함수의 프로토타입 객체라고 한다.) 9-1.js

프로토타입 체인 메소드 체인 9-1.html

9.2 프로토타입 상속

9.2.1 상속

Java, C++: 클래스 상속 Javascript: 프로토타입 상속

프로토타입 상속이란? **프로토타입 체인** 자료구조를 통해 클래스가 아닌 객체를 상속

함수형 언어에서 상속과 비슷해지고 싶어서 흉내낸 것. 결과적으로 ES6 부터는 그냥 class가 추가됨.

9.2.2 상속을 하는 이유

유지 보수

9.2.3 프로토타입 체인

`obj.prototype`: prototype 프로퍼티 `obj.__proto__`: 내부 프로퍼티 **프로토타입**: `__proto__`가 가리키는 객체

결국 프로토타입 체인이란? `__proto__`를 통해 부모 프로토타입들이 연결된것. List 자료 구조의 next를 생각해보자.
p.345 그림 참조

`obj.__proto__`로 사용하기 보다는`Object.getPrototypeOf(obj)`를 통해 readOnly로 사용하기를 권장한다.

상속관계에서 자식이 부모를 변경한다는 것은 말이 안되지만, 정 바꾸고 싶다면 `Object.setPrototypeOf`를 활용하자.

9.2.4 new 연산자의 역할

1. new Object() 생성
2. `__proto__`설정
3. constructor 실행
4. return

9.2.5 프로토타입 객체의 프로퍼티

```
Function = {
    ...
    prototype: {
        ...
        constructor: 'constructor 프로퍼티',
        __proto__: '내부 프로퍼티',
    }
}
```

constructor 프로퍼티

Card 함수가 constructor 프로퍼티

```
function Card(suit, rank) {
    this.suite = suit;
    this.rank = rank;
}
var card = new Card("하트", "A");
```

내부 프로퍼티

프로토타입 체인을 통해 연결된 부모 객체

```
var person1 = {
    name: "Tom",
    sayHello: function () { console.log(`Hello!: ${this.name}`); }
};
var person2 = {
    name: "Huck",
};
person2.__proto__ = person1;
person2.name = "Huck";
person2.sayHello(); // Hello! Huck
```

프로토타입 객체의 교체 및 constructor 프로퍼티

인스턴스의 프로퍼티는 생성 시점의 프로토타입에서 상속받는다. => 인스턴스 생성 전 프로토타입 수정 인스턴스 생성 후 프로토타입 수정 위의 두개는 다른 결과를 낸다. 9-2-5.js

9.2.6 프로토타입의 확인

```
function F() {};
var obj = new F();
```

1 instanceof

```
obj instance of F
```

2 isPrototypeOf

```
F.prototype.isPrototypeOf(obj)
```

9.2.7 Object.prototype

- Object는 최상위 부모 객체이다.
    - Object의 부모(`__proto__`)는 null이다.
- Object객체는 Object 생성자(`var obj = Object()`)로 만들수 있다.
    - 원래 new를 붙여야 하지만, 생략 가능하다.
- Object 생성자의 프로퍼티와 메소드가 있다.
- 자식들은 부모인 Object.prototype 안에 정의된 메소드를 사용할 수 있다.
    - 예시: `isPrototypeOf`

9.2.8 Object.create로 객체 생성하기

```
var person1 = {
    name: "Tom",
    sayHello: function () { console.log(`Hello!: ${this.name}`); }
};
var person2 = Object.create(person1);
person2.name = "Huck";
person2.sayHello(); // Hello! Huck
```

위에것과 비교

```
var person1 = {
    name: "Tom",
    sayHello: function () { console.log(`Hello!: ${this.name}`); }
};
var person2 = {
    name: "Huck",
};
person2.__proto__ = person1;
person2.name = "Huck";
person2.sayHello(); // Hello! Huck
```