Javascript - 클래스

이 부분의 내용은 OOP in JS 내용을 그대로 가져온 것이다. 파트 1의 번역본도 존재하지만 번역질이 좋지 않다. -_-;

Public/Private 변수, 함수

- **private 변수**는 객체 안에서 'var' 키워드로 선언한다, 그리고 private 함수와 privileged 메소드만이 접근 할수 있다.
- **private 함수**는 객체의 생성자 안에서 선언한다. (또는 `var functionName=function(){…}` 로 선언 가능하다.) 그리고 (생성자를 포함하여) privileged 메소드에 의해서만 호출이 가능하다.
- **privileged 메소드**는 다음과 같이 선언한다.

```
this.methodName=function(){...}
```

객체 바깥의 코드에 의해 호출이 가능하다.

- **public 속성**은 다음과 같이 선언한다.

```
this.variableName
```

객체 바깥에서 읽기와 쓰기가 가능하다.

- **public 메소드**는 다음과 같이 정의된다.

```
Classname.prototype.methodName = function(){...}
```

그리고 객체의 바깥에서 호출될수 있다.

- **prototype 속성**은 다음과 같이 정의된다.

```
Classname.prototype.propertyName = someValue
```

- **static 속성**은 다음과 같이 정의된다.

```
Classname.propertyName = someValue
```

Example

In this example, a person's name and race are set at birth and may never be changed. When created, a person starts out at year 1 and a hidden maximum age is determined for that person. The person has a weight which is modified by eating (tripling their weight) or exercising (halfing it). Every time the person eats or exercises, they grow a year older. The person object has a publicly accessible 'clothing' property which anyone can modify, as well as a dirtFactor which can be modified manually (throwing dirt on or scrubbing it off), but which increases every time the person eats or exercises, and is reduced by the use of the shower() method.

```
function Person(n,race){
    this.constructor.population++;

    //
*****************************************************************************
    // PRIVATE 변수와 함수
    // PRIVELEGED 메소드만 보기,수정,호출이 가능하다.
    //
*****************************************************************************
    var alive=true, age=1;
    var maxAge=70+Math.round(Math.random()*15)+Math.round(Math.random()*15);
    function makeOlder(){ return alive = (++age <= maxAge) }

    var myName=n?n:"John Doe";
    var weight=1;


    //
*****************************************************************************
    // PRIVILEGED 메소드
    // 어디서든 호출할수 있고 PRIVATE ITEMS에 접근이 가능하다
    // PUBLIC FLAVORS와 함께 변경될수 있다
    //
*****************************************************************************
    this.toString=this.getName=function(){ return myName }

    this.eat=function(){
        if (makeOlder()){
            this.dirtFactor++;
            return weight*=3;
        } else alert(myName+" 는 먹을수 없습니다, 그는 죽었습니다!");
    }
    this.exercise=function(){
        if (makeOlder()){
            this.dirtFactor++;
            return weight/=2;
        } else alert(myName+" 는 운동할수 없습니다, 그는 죽었습니다!");
    }
    this.weigh=function(){ return weight }
    this.getRace=function(){ return race }
    this.getAge=function(){ return age }
    this.muchTimePasses=function(){ age+=50; this.dirtFactor=10; }


    //
*****************************************************************************
    // PUBLIC 속성 -- 누구든 읽고 쓸수 있다
    //
*****************************************************************************
    this.clothing="아무것도입지않음";
    this.dirtFactor=0;
```

```
}


// ********************************************************************************
// PUBLIC 메소드 -- 누구든 읽고 쓸수 있다
// ********************************************************************************
Person.prototype.beCool = function(){ this.clothing="검은 셔츠" }
Person.prototype.shower = function(){ this.dirtFactor=2 }
Person.prototype.showLegs = function(){ alert(this+" 는 다리가 "+this.legs+" 개
입니다") }
Person.prototype.amputate = function(){ this.legs-- }


// ********************************************************************************
// PROTOTYOPE 속성 -- 누구든 읽고 쓸수 있다 (overridden 할수 없습니다)
// ********************************************************************************
Person.prototype.legs=2;


// ********************************************************************************
// STATIC 속성 -- 누구든 읽고 쓸수 있다
// ********************************************************************************
Person.population = 0;



// Person 클래스를 사용한 코드
function RunGavinsLife(){
    var gk=new Person("Gavin","caucasian");        // Person 객체의 인스턴스를
생성
    var lk=new Person("Lisa","caucasian");         // Person 객체의 인스턴스를
생성
    alert("현재 "+Person.population+" 명의 사람이 있습니다");

    gk.showLegs(); lk.showLegs();                  // 'this.legs'를 참조할때 모
두 'Person.prototype.legs' 변수를 공유한다

    gk.race = "hispanic";                          // 값을 변경하려고 해도 private
'race' 변수는 overwrite 되지 않는다.
    alert(gk+"'의 인종은 "+gk.getRace()+" 입니다");     // 인스턴스 생성시에 지정된
private 'race' 변수로부터 'caucasian'을 리턴한다
    gk.eat(); gk.eat(); gk.eat();                  // 몸무게는 3... 9... 27
    alert(gk+" 는 "+gk.weigh()+" 파운드의 무게가 나가며 dirt factor 는
"+gk.dirtFactor + "입니다");

    gk.exercise();                                 // 몸무게 13.5
    gk.beCool();                                   // 옷이 새로운 것으로 변경됩니다.
    gk.clothing="Pimp Outfit";                     // clothing 은 public 변수
고 다른 값으로 변경이 가능합니다.
    gk.shower();
    alert("현재의 샤워기술로 "+gk+" 의 dirt factor가 "+gk.dirtFactor + " 이 되었습니
```

```
다");

    gk.muchTimePasses();                          //50년이 지남
    Person.prototype.shower=function(){           //모두에게 샤워 기술이 향상됨
        this.dirtFactor=0;
    }
    gk.beCool=function(){                         //Gavin 은 혼자서 새로운 패션
아이디어를 얻었다.
        this.clothing="꽃남방";
    };

    gk.beCool(); gk.shower();
    alert("패셔너블한 "+gk+" 는 "
        +gk.getAge()+" 살이고 지금 "
        +gk.clothing+" 를 입고있으며 dirt factor 는 "
        +gk.dirtFactor+"이다");

    gk.amputate();                                //prototype 메소드를 사용하여
public 속성을 만듭니다
    gk.showLegs(); lk.showLegs();                 //Lisa는 여전히 prototype 속
성을 가지고 있다

    gk.muchTimePasses();                          //50년이 지나고... Gavin 은
현재 100살이 넘었다.
    gk.eat();                                     // 나이가 수명을 초과하였고 죽었
기 때문에 먹을수 없다.
}
```

Notes

- maxAge is a private variable with no privileged accessor method; as such, there is no way to publicly get or set it.
- race is a private variable defined only as an argument to the contructor. Variables passed into the constructor are available to the object as private variables.
- The 'tinfoil' beCool() fashion method was applied only to the gk object, not the entire Person class. Other people created and set to beCool() would still use the original 'khakis and black shirt' clothing that Gavin eschewed later in life.
- Note the implicit call to the gk.toString() method when using string concatenation. It is this which allows the code `alert(gk+' is so cool.')` to put the word 'Gavin' in there, and is equivalent to `alert(gk.toString()+' is so cool.')`. Every object of every type in JS has a .toString() method, but you can override it with your own.
- You cannot (to my knowledge) assign public methods of a class inside the main object constructor...you must use the prototype property externally, as above with the beCool() and shower() methods.
- As I attempted to show with the Person.prototype.legs property and the amputate() function, prototype properties are shared by all object instances. Asking for lk.legs yields '2' by looking at the single prototype property. However, attempting to change this value using either

`gk.legs=1` or (in the Person object) `this.legs=1` ends up making a new public property of the object specific to that instance. (This is why calling gk.amputate() only removed a leg from Gavin, but not Lisa.) To modify a prototype property, you must use `Person.prototype.legs=1` or something like `this.constructor.prototype.legs=1`. (I say 'something like' because I discovered that this.constructor is not available inside private functions of the object, since this refers to the window object in that scope.)

- 어디서든 이름이 없는 함수의 선언은 인라인으로 다음과 같이 선언한다.

```
foo = function(p1,p2){ some code }
```

new Function() 생성자와는 다르다. 예를 들어,

```
foo = new Function('p1','p2','code');
```

후자의 코드는 (생성자 함수의 scope을 상속받지 않고) 전역 scope에서 실행된다. 전자의 코드는 private 변수의 접근을 막아준다.

- As noted above in the code comments, the act of setting gk.race to some value did NOT overwrite the private race variable. Although it would be a dumb idea, you can have both private and public variables with the same name. For example, the yell() method in the following class will yield different values for foo and this.foo:

```
function StupidClass(){
  var foo = "internal";
  this.foo = "external";
  this.yell=function(){ alert("Internal foo is "+foo+"\nExternal foo is
"+this.foo) }
}
```

- Private functions and privileged methods, like private variables and public properties, are instantiated with each new object created. So each time new Person() is called, new copies of makeOlder(), toString(), getName(), eat(), exercise(), weigh(), getRace(), getAge(), and muchTimePasses() are created. For every Person, each time. Contrast this with public methods (only one copy of beCool() and shower() exist no matter how many Person objects are created) and you can see that for memory/performance reasons it can be preferable to give up some degree of object protection and instead use only public methods.
- Note that doing so requires making private variables public (since without privileged accessor methods there would be no way to use them) so the public methods can get at them...and which also allows external code to see/destroy these variables. The memory/performance optimization of using only public properties and methods has consequences which may make your code less robust.
- For example, in the above age and maxAge are private variables; age can only be accessed externally through getAge() (it cannot be set) and maxAge cannot be read or set externally. Changing those to be public properties would allow any code to do something like `gk.maxAge=1; gk.age=200;` which not only does it not make sense (you shouldn't be able to manipulate someone's age or lifespan directly), but by setting those values directly the alive variable wouldn't properly be updated, leaving your Person object in a broken state.

상속

- Javascript - 상속