

PLP - Práctica 5: Programación Orientada a Objetos

Zamboni, Gianfranco

8 de marzo de 2018

Programación en JS

5.1. Ejercicio 1

a)

```
c1i = {  
  r: 1,  
  i: 1  
}
```

b)

```
c1i.sumar = function(complejo){  
  this.r += complejo.r  
  this.i += complejo.i  
}
```

c)

```
c1i.sumar = function(complejo){  
  return {  
    r: this.r + complejo.r  
    i: this.i + complejo.i  
  }  
}
```

d) Asi como están implementadas, funciona.

e)

```
let c = c1i.sumar(c1i)  
  
c1i.sumar = function(complejo){  
  return {  
    r: this.r - complejo.r  
    i: this.i - complejo.i  
  }  
}
```

f)

```
c1i.mostrar = function () {  
  if(this.i !== 1){  
    console.log(  
      `${this.r} + ${this.i}i`);  
  }else{  
    console.log(`${this.r} + i");  
  }  
}
```

c1i no tiene definida la función restar, por lo que c1i.restar(_) nos dará un error.

En este caso, c no podrá mostrar sus elementos, ya que cuando lo creamos lo hacemos como un objeto nuevo que no está relacionado con c.

5.2. Ejercicio 2

```
// a)
let t = {}

t.ite = function(a,b) {
  return a;
}

// b)
t.mostrar = function() {
  return "Verdadero";
}

// c)
t.not = function(){
  return f;
}

t.and = function(otroValor) {
  return otroValor;
}
```

```
// a)
let f = {}

f.ite = function(a,b) {
  return b;
}

// b)
f.mostrar = function(){
  return "Falso";
}

// c)
f.not = function(){
  return t;
}

f.and = function(element) {
  return this;
}
```

5.3. Ejercicio 3

```
// a)
let Cero = {};

Cero.esCero = function(){
    return true;
};

Cero.succ = function(){
    return new Sucesor(this);
}

function Sucesor(pred){
    this.predecesor = pred;
};

Sucesor.prototype.__proto__ = Cero;

Sucesor.prototype.pred = function(){
    return this.predecesor;
}

// b)
Cero.toNumber = function(){
    return 0;
}

Sucesor.prototype.toNumber = function(){
    return this.pred().toNumber() + 1;
}

// c)
Cero.for = function(f){}

Sucesor.prototype.for = function(f){
    this.pred().for(f);
    f.eval(this);
}
```

5.4. Ejercicio 4

a)

```
var Punto = {};  
Punto.new = function(x,y) {  
    var p = {};  
    p.x=x;  
    p.y=y;  
    p.mostrar = function () {  
        return Punto.mostrar(this);  
    }  
    return p;  
}  
  
Punto.mostrar = function(o) {  
    return 'Punto({o.x},{o.y})'  
}
```

b)

```
var PuntoColoreado = {};  
PuntoColoreado.new = function(x,y){  
    var p = Punto.new(x,y);  
    p.color = "rojo";  
    p.mostrar = function(){  
        return PuntoColoreado.mostrar(this);  
    };  
    return p;  
};  
  
PuntoColoreado.mostrar = function(o) {  
    return Punto.mostrar(o);  
};
```

c)

```
var PuntoColoreado = {};  
PuntoColoreado.cons = function (x,y,color) {  
    var nuevo = this.new(x,y);  
    nuevo.color = color;  
    return nuevo;  
}
```

d)

5.5. Ejercicio 5

```
function Punto(x, y){
    this.x = x;
    this.y = y;
}

Punto.prototype.mostrar = function(){
    return `Punto(${this.x},${this.y})`
}

function PuntoColoreado(x,y, color) {
    this.x = x
    this.y = y
    this.color = color
}
PuntoColoreado.prototype.__proto__ = Punto.prototype

Punto.prototype.moverX = function(x){
    this.x += x
}
```

5.6. Ejercicio 6

a) En el primer caso, una vez creado el objeto `a`, el prototipo de este objeto ya queda fijo. Cuando realizamos la asignación `C1.prototype = C2.prototype`; la variable `C1.prototype` deje de referenciar al objeto prototipo de `a` y referencie al objeto que referencia `C2.prototype`. Provocando, esto, que el constructor `C1()` lo asigne como prototipo de `b`.

Entonces `a` sigue teniendo el mismo prototipo y `b` tiene como prototipo a `C2` por lo que los resultados de evaluar sus atributos son:

```
a.g // 'Hola'
b.g // 'Mundo'
```

b) En el segundo caso, estamos modificando el atributo `g` del objeto que es referenciado por `C1.prototype`. La asignación `C1.prototype.g = C2.prototype.g`; reemplaza la función `g` original de `C1.prototype` por la función `g` de `C2.prototype`. Entonces, las soluciones quedan:

```
a.g // 'Mundo'
b.g // 'Mundo'
```

5.7. Ejercicio 7

a) `a` será un array con todas las claves de `o1` y `b` será el array con todos sus valores en el mismo orden, es decir, si en `a[0]` se encuentra la clave `'a'`, entonces en `b[0]` se encuentra el valor `1`.

b)

```
function extender(o1, o2) {
    for(let key in o1) {
        if(o2[key] == undefined){
            o2[key] = o1[key];
        }
    }
}
```

- c) Hay que hacer dos modificaciones: Definir en B el método y eliminarlo de A.

```
B.presentar = A.presentar
delete A.presentar
```

- d) Es lo mismo que en el anterior, pero usando `prototype`.

```
B7d.prototype.presentar = A7d.prototype.presentar
delete A7d.prototype.presentar
```

Cálculo de Objetos

5.8. Ejercicio 8

- a) Ambos objetos tienen los mismos atributos y los métodos asociados a cada tributo son equivalentes (son los mismos, salvo renombre de variables).
- b) En este caso, los objetos tienen distintas etiquetas por lo que no pueden ser considerados iguales. Representan a distintos objetos porque responden a los distintos mensajes.

5.9. Ejercicio 9

- a)

$$\frac{\frac{\frac{}{o \longrightarrow o} [\text{Obj}]}{\frac{\frac{\frac{}{o \longrightarrow o} [\text{Obj}]}{o \longrightarrow o} [\text{Obj}]}{(x.arg)\{x \leftarrow o\} \longrightarrow o} [\text{Sel}]} \quad \frac{\frac{\frac{}{o \longrightarrow o} [\text{Obj}]}{o \longrightarrow o} [\text{Obj}]}{(x)\{x \leftarrow o\} \longrightarrow o} [\text{Obj}]}{(x)\{x \leftarrow o\} \longrightarrow o} [\text{Sel}]}{o.val \longrightarrow o} [\text{Sel}]$$

- b)

$$\frac{\text{Por 9.a)} \quad \frac{\frac{\frac{}{o \longrightarrow o} [\text{Obj}]}{o \longrightarrow o} [\text{Obj}]}{o.val \longrightarrow o} [\text{Obj}]}{o.val \longrightarrow o} \quad \frac{\frac{\frac{\frac{}{o \longrightarrow o} [\text{Obj}]}{o \longrightarrow o} [\text{Obj}]}{(x.arg)\{x \leftarrow o\} \longrightarrow o} [\text{Sel}]}{\frac{\frac{\frac{}{o \longrightarrow o} [\text{Obj}]}{o \longrightarrow o} [\text{Obj}]}{(x)\{x \leftarrow o\} \longrightarrow o} [\text{Obj}]}{(x)\{x \leftarrow o\} \longrightarrow o} [\text{Sel}]}{o.val.arg \longrightarrow o} [\text{Sel}]$$

- c)

$$\frac{\frac{\frac{}{o \longrightarrow o} [\text{Obj}]}{o \longrightarrow o} [\text{Obj}]}{o.arg \Leftarrow \varsigma(z)0 \longrightarrow o'} [\text{Upd}] \quad \frac{o' \equiv [arg = \varsigma(z)0, val = \varsigma(x)x.arg] \quad \frac{o' \longrightarrow o' \quad 0\{x \leftarrow o'\} \longrightarrow 0}{x.arg\{x \leftarrow o'\} \longrightarrow 0} [\text{Sel}]}{(o.arg \Leftarrow \varsigma(z)0).val \longrightarrow 0} [\text{Sel}]$$

5.10. Ejercicio 10

$$\frac{(1) \quad \frac{o.a \longrightarrow o' \quad o' \equiv [a = \varsigma(y)y := []]}{o.a.a \longrightarrow [a = []]} \quad \frac{o' \longrightarrow o' \quad (y.a := [])\{y \leftarrow o'\} \longrightarrow [a = []]}{[\text{Upd}]} \quad [\text{Sel}]}{o.a.a \longrightarrow [a = []]}$$

$$(1) \quad \frac{o \longrightarrow o \quad \frac{o \longrightarrow o \quad (x.a \Leftarrow \varsigma(y)(y := []))\{x \leftarrow o\} \longrightarrow [a = \varsigma(y)y := []]}{[\text{Upd}]}}{o.a \longrightarrow [a = \varsigma(y)y := []]} [\text{Sel}]$$

5.11. Ejercicio 11

a)

$$\begin{aligned} \text{true} &= [\text{not} = \text{false}, & \text{false} &= [\text{not} = \text{true}, \\ & \text{if} = \lambda(y)\lambda(z)y, & & \text{if} = \lambda(y)\lambda(z)z, \\ & \text{ifnot} = \lambda(y)\lambda(z)f.\text{if}(y,z) & & \text{ifnot} = \lambda(y)\lambda(z)t.\text{if}(y,z) \\ &] & &] \end{aligned}$$

b)

$$\begin{aligned} \text{true} &= [\text{not} = \text{false}, & \text{false} &= [\text{not} = \text{true}, \\ & \text{if} = \lambda(y)\lambda(z)y, & & \text{if} = \lambda(y)\lambda(z)z, \\ &] & &] \end{aligned}$$

5.12. Ejercicio 12

a)

$$\text{origen} \stackrel{\text{def}}{=} [x = 0, \\ y = 0, \\ mv = \varsigma(p)\lambda(w)\lambda(z)(p.x := p.x + w).y := p.y + z]$$

b)

$$\text{Punto} \stackrel{\text{def}}{=} [\text{new} = \varsigma(z)[x = 0, y = 0, mv = \varsigma(p)\lambda(x_1)\lambda(x_2)z.mv(p, x_1, x_2)], \\ mv = \varsigma(c)\lambda(p)\lambda(w)\lambda(z)(p.x := p.x + w).y := p.y + z]$$

c)

$$\frac{\text{Punto} \longrightarrow \text{Punto} \quad [x = 0, y = 0, mv = \varsigma(p)\lambda(x_1)\lambda(x_2)z.mv(p, x_1, x_2)]\{z \leftarrow \text{Punto}\} \longrightarrow p}{\text{Punto.new} \longrightarrow p} [\text{Sel}]$$

$$\text{con } p \stackrel{\text{def}}{=} [x = 0, y = 0, mv = \varsigma(p)\lambda(x_1)\lambda(x_2)\text{Punto}.mv(p, x_1, x_2)]$$

d)

$$PuntoColoreado \stackrel{def}{=} [new = \varsigma(z)[x = 0, y = 0, color = \mathbf{rojo}, mv = \varsigma(p)\lambda(x_1)\lambda(x_2)z.mv(p, x_1, x_2)], \\ mv = Punto.mv]$$

5.13. Ejercicio 13

a)

$$plantaClass = [new = \varsigma(c)[altura = c.altura, crecer = \varsigma(t)c.crecer t], \\ altura = 10, \\ crecer = \varsigma(c)\lambda(t)t.altura := (t.altura + 10) \\]$$

b)

$$\frac{(1) \quad \frac{plantaClass.new \longrightarrow p' \quad (plantaClass.crecer t)\{t \leftarrow p'\} \longrightarrow p}{plantaClass.new.crecer \longrightarrow [altura = 20, crecer = \varsigma(t)plantaClass.crecer t]} [Sel]}{con p \equiv [altura = 20, crecer = \varsigma(t)plantaClass.crecer t]}$$

$$(1) \quad \frac{\frac{plantaClass \longrightarrow plantaClass}{[Obj]} \quad [altura = c.altura, crecer = \varsigma(t)c.crecer t]\{c \leftarrow plantaClass\} \longrightarrow p'}{plantaClass.new \longrightarrow p'} [Sel]$$

$$con p' \equiv [altura = plantaClass.altura, crecer = \varsigma(t)plantaClass.crecer t]$$

c)

$$broteClass = [new = \varsigma(c)[altura = c.altura, crecer = \varsigma(t)c.crecer t], \\ altura = 1, \\ crecer = \lambda(t)plantaClass.crecer t \\]$$

d)

$$malezaClass = [new = \varsigma(c)[altura = c.altura, crecer = \varsigma(t)c.crecer t], \\ altura = plantaClass.altura, \\ crecer = \lambda(t)t.altura := (t.altura * 2) \\]$$

e)

```
frutalClass = [ new =  $\varsigma(c)[altura = c.altura, cantFrutos = c.cantFrutos$   
                $crecer = \varsigma(t)c.crecer\ t]$ ,  
                $cantFrutos = 0$ ,  
                $altura = plantaClass.altura$ ,  
                $crecer = \lambda(t)((plantaClass.crecert).cantFrutos := t.cantFrutos + 1)$   
               ]
```

f)