



# PIGGYBANK

## Simple PiggyBank



```

X balance = 0;
let lt = 0;

function deposit(v)
{
    if(v > 0)
    {
        balance += v;
        lt = v;
    }
}

function withdraw(v)
{
    if(v > 0 && v <= balance)
    {
        balance -= v;
        lt = -v;
    }
}

function statement()
{
    console.log("Balance: " + balance);
    console.log("Last Transaction: " + lt);
}

deposit(100);
statement();
withdraw(50);
statement();
```

## PiggyBank in an Object



```
var pg1 = {  
  balance:0,  
  lt:0  
};  
  
function deposit(v)  
{  
  if(v > 0)  
  {  
    pg1.balance += v;  
    pg1.lt = v;  
  }  
}  
  
function withdraw(v)  
{  
  if(v > 0 && v <= pg1.balance)  
  {  
    pg1.balance -= v;  
    pg1.lt = -v;  
  }  
}  
  
function statement()  
{  
  console.log("Balance: " + pg1.balance);  
  console.log("Last Transaction: " + pg1.lt);  
}  
  
deposit(100);  
statement();  
withdraw(50);  
statement();
```

## Multiple PiggyBank

```
var pg1 = {  
  balance: 0,
```







```
lt: 0

var pg2 = {
  balance: 0,
  lt: 0
};

function deposit1(v)
{
  pg1.balance = pg1.balance + v;
  pg1.lt = v;
}

function withdraw1(v)
{
  if (pg1.balance >= v)
  {
    pg1.balance = pg1.balance - v;
    pg1.lt = -v;
  }
}

function statement1()
{
  console.log("Balance = " + pg1.balance);
  console.log("Last Transction = " + pg1.lt);
}


function deposit2(v)
{
  pg2.balance = pg2.balance + v;
  pg2.lt = v;
}

function withdraw2(v)
{
  if (pg2.balance >= v) ⓘ
  {
```

```
pg2.balance = pg2.balance - v;
pg2.lt = -v;
}
}
function statement2()
{
    console.log("Balance = " + pg2.balance);
    console.log("Last Transction = " + pg2.lt);
}

deposit1(100);
statement1();
withdraw1(50);
statement1();
withdraw1(10);
statement1();

deposit2(200);
statement2();
withdraw2(100);
statement2();
withdraw2(50);
statement2();
```

## Object Modelling - Message Passing

```
var pg1 = {
    balance: 0,
    lt: 0,
    deposit : deposit1,
    withdraw : withdraw1,
    statement : statement1
};

var pg2 = {
    balance: 0,
    lt: 0,
```





```
    deposit : deposit2,  
    withdraw : withdraw2,  
    statement : statement2  
};  
  
function deposit1(v)  
{  
    pg1.balance = pg1.balance + v;  
    pg1.lt = v;  
}  
  
function withdraw1(v)  
{  
    if (pg1.balance >= v)  
    {  
        pg1.balance = pg1.balance - v;  
        pg1.lt = -v;  
    }  
}  
  
function statement1()  
{  
    console.log("Balance = " + pg1.balance);  
    console.log("Last Transction = " + pg1.lt);  
}  
  
function deposit2(v)  
{  
    pg2.balance = pg2.balance + v;  
    pg2.lt = v;  
}  
  
function withdraw2(v)  
{  
    if (pg2.balance >= v)  
    {  
        pg2.balance = pg2.balance - v;  
        pg2.lt = -v;  
    }  
}  
  
function statement2()  
{
```





```
console.log("Balance = " + pg2.balance);  
console.log("Last Transaction = " + pg2.lt);  
}
```



```
pg1.deposit(100);  
pg1.statement();  
pg1.withdraw(50);  
pg1.statement();  
pg1.withdraw(10);  
pg1.statement();
```

```
pg2.deposit(200);  
pg2.statement();  
pg2.withdraw(100);  
pg2.statement();  
pg2.withdraw(50);  
pg2.statement();
```

## Procedural Modelling - Reuse

```
var pg1 = {  
  balance: 0,  
  lt: 0,  
};  
  
var pg2 = {  
  balance: 0,  
  lt: 0,  
};  
  
function deposit(pg,v)  
{  
  pg.balance = pg.balance + v;  
  pg.lt = v;  
}  
  
function withdraw(pg,v) ⓘ  
{
```



```
if (pg.balance >= v)
{
```



```
    pg.balance = pg.balance - v;
```

```
    pg.lt = -v;
```

```
}
```

```
}
```

```
function statement(pg)
```

```
{
```

```
    console.log("Balance = " + pg.balance);
```

```
    console.log("Last Transction = " + pg.lt);
```

```
}
```

```
deposit(pg1,100);
```

```
statement(pg1);
```

```
withdraw(pg1,50);
```

```
statement(pg1);
```

```
withdraw(pg1,10);
```

```
statement(pg1);
```

```
deposit(pg2,200);
```

```
statement(pg2);
```

```
withdraw(pg2,100);
```

```
statement(pg2);
```

```
withdraw(pg2,50);
```

```
statement(pg2);
```

## Object Modelling - this

```
var pg1 = {
```

```
    balance: 0,
```

```
    lt: 0,
```

```
    deposit : deposit,
```

```
    withdraw : withdraw,
```

```
    statement : statement
```

```
};
```



```
var pg2 = {
```





```
    balance: 0,  
    lt: 0,  
    deposit : deposit,  
    withdraw : withdraw,  
    statement : statement  
};  
  
function deposit(v)  
{  
    this.balance = this.balance + v;  
    this.lt = v;  
}  
  
function withdraw(v)  
{  
    if (this.balance >= v)  
    {  
        this.balance = this.balance - v;  
        this.lt = -v;  
    }  
}  
  
function statement()  
{  
    console.log("Balance = " + this.balance);  
    console.log("Last Transction = " + this.lt);  
}  
  
pg1.deposit(100);  
pg1.statement();  
pg1.withdraw(50);  
pg1.statement();  
pg1.withdraw(10);  
pg1.statement();  
  
pg2.deposit(200);  
pg2.statement();  
pg2.withdraw(100);  
pg2.statement();
```



```
.statement();  
pg2.withdraw(50);  
pg2.statement();
```



## PiggyBank with multiple Transactions

```
function deposit(v)  
{  
  if(v > 0)  
  {  
    this.balance += v;  
    this.transactions.push(v);  
  }  
}  
  
function withdraw(v)  
{  
  if(v > 0 && v <= this.balance)  
  {  
    this.balance -= v;  
    this.transactions.push(-v);  
  }  
}  
  
function statement()  
{  
  console.log("Balance: " + this.balance);  
  for(let i of this.transactions)  
  {  
    console.log("Transaction: " + i);  
  }  
}  
  
let pg1 = {  
  balance:0,  
  transactions:[],  
  deposit:deposit,  
  withdraw:withdraw,  
  statement:statement,
```





```
pg1.deposit(100);  
pg1.deposit(30);  
pg1.deposit(40);  
pg1.statement();  
  
pg1.withdraw(35);  
pg1.withdraw(10);  
pg1.statement();
```

## Object Oriented Modelling - Inheritance

```
function deposit(v)  
{  
    this.balance = this.balance + v;  
    this.lt = v;  
}  
function withdraw(v)  
{  
    if (this.balance >= v)  
    {  
        this.balance = this.balance - v;  
        this.lt = -v;  
    }  
}  
function statement()  
{  
    console.log("Balance = " + this.balance);  
    console.log("Last Transction = " + this.lt);  
}  
  
var base = {  
    deposit: deposit,  
    withdraw: withdraw,  
    statement: statement  
};
```



```
pg1 = {  
  balance: 0,  
  lt: 0,  
  __proto__: base  
};  
  
var pg2 = {  
  balance: 0,  
  lt: 0,  
  __proto__: base  
};  
  
pg1.deposit(100);  
pg1.statement();  
pg1.withdraw(50);  
pg1.statement();  
pg1.withdraw(10);  
pg1.statement();  
  
pg2.deposit(200);  
pg2.statement();  
pg2.withdraw(100);  
pg2.statement();  
pg2.withdraw(50);  
pg2.statement();
```

## Steps Involved in creating a new Object

- Creating a new variable
- Creating a new Object literal with entries for balance and lt.
- Linking the \_\_proto\_\_ property of new object ( child ) to base object ( parent ).

## Object Modelling - Factory

```
function deposit(v)  
{
```





```
this.balance = this.balance + v;  
this.lt = v;
```



```
}
```

```
function withdraw(v)
```

```
{
```

```
    if (this.balance >= v)
```

```
    {
```

```
        this.balance = this.balance - v;
```

```
        this.lt = -v;
```

```
    }
```

```
}
```

```
function statement()
```

```
{
```

```
    console.log("Balance = " + this.balance);
```

```
    console.log("Last Transction = " + this.lt);
```

```
}
```

```
function Piggybank()
```

```
{
```

```
    var obj = {
```

```
        balance: 0,
```

```
        lt: 0,
```

```
        deposit: deposit,
```

```
        withdraw: withdraw,
```

```
        statement: statement
```

```
    };
```

```
    return obj;
```

```
}
```

```
var pg1 = Piggybank();
```

```
pg1.deposit(100);
```

```
pg1.statement();
```

```
pg1.withdraw(50);
```

```
pg1.statement();
```



```
pg1.withdraw(10);
```

 `.statement();` `var pg2 = Piggybank();``pg2.deposit(200);``pg2.statement();``pg2.withdraw(100);``pg2.statement();``pg2.withdraw(50);``pg2.statement();``var pg3 = Piggybank();``pg3.deposit(300);``pg3.statement();``pg3.withdraw(200);``pg3.statement();``pg3.withdraw(100);``pg3.statement();`

## Object Modelling - new

`function deposit(v)``{` `this.balance = this.balance + v;` `this.lt = v;``}``function withdraw(v)``{` `if (this.balance >= v)` `{` `this.balance = this.balance - v;` `this.lt = -v;` `}``}``function statement()`



```
console.log("Balance = " + this.balance);
console.log("Last Transction = " + this.lt);
}

function PiggyBank()
{

    this.balance = 0;
    this.lt = 0;
    this.deposit = deposit;
    this.withdraw = withdraw;
    this.statement = statement;
}

var pg1 = new PiggyBank();

pg1.deposit(100);
pg1.statement();
pg1.withdraw(50);
pg1.statement();
pg1.withdraw(10);
pg1.statement();


var pg2 = new PiggyBank();

pg2.deposit(200);
pg2.statement();
pg2.withdraw(100);
pg2.statement();
pg2.withdraw(50);
pg2.statement();

var pg3 = new PiggyBank();

pg3.deposit(300);
pg3.statement();
```





```

    .statement(),
    .withdraw(200);
pg3.statement();
pg3.withdraw(100);
pg3.statement();

```

## Object Oriented Modelling - Prototype

```

function deposit(v)
{
    this.balance = this.balance + v;
    this.lt = v;
}

function withdraw(v)
{
    if (this.balance >= v)
    {
        this.balance = this.balance - v;
        this.lt = -v;
    }
}

function statement()
{
    console.log("Balance = " + this.balance);
    console.log("Last Transction = " + this.lt);
}

function PiggyBank()
{
    this.balance = 0;
    this.lt = 0;
}

PiggyBank.prototype.deposit = deposit;
PiggyBank.prototype.withdraw = withdraw;
PiggyBank.prototype.statement = statement;

```



```
✕ pg1 = new PiggyBank();
```



```
pg1.deposit(100);
```

```
pg1.statement();
```

```
pg1.withdraw(50);
```

```
pg1.statement();
```

```
pg1.withdraw(10);
```

```
pg1.statement();
```

```
var pg2 = new PiggyBank();
```

```
pg2.deposit(200);
```

```
pg2.statement();
```

```
pg2.withdraw(100);
```

```
pg2.statement();
```

```
pg2.withdraw(50);
```

```
pg2.statement();
```

```
var pg3 = new PiggyBank();
```

```
pg3.deposit(300);
```

```
pg3.statement();
```

```
pg3.withdraw(200);
```

```
pg3.statement();
```

```
pg3.withdraw(100);
```

```
pg3.statement();
```

## Method Chaining - Fluent Interface

```
function deposit(v)
```

```
{
```

```
    this.balance = this.balance + v;
```

```
    this.lt = v;
```

```
    return this;
```

```
}
```

```
function withdraw(v)
```





```
if (this.balance >= v)
```

```
{
```

```
    this.balance = this.balance - v;
```

```
    this.lt = -v;
```

```
}
```

```
return this;
```

```
}
```

```
function statement()
```

```
{
```

```
    console.log("Balance = " + this.balance);
```

```
    console.log("Last Transction = " + this.lt);
```

```
    return this;
```

```
}
```

```
function PiggyBank()
```

```
{
```

```
    this.balance = 0;
```

```
    this.lt = 0;
```

```
}
```

```
PiggyBank.prototype.deposit = deposit;
```

```
PiggyBank.prototype.withdraw = withdraw;
```

```
PiggyBank.prototype.statement = statement;
```

```
console.log("Behold the chaining");
```

```
var pg1 = new PiggyBank();
```

```
pg1.deposit(100).statement().withdraw(50).statement().withdraw(100).
```


```
statement();
```

```
new
```

```
PiggyBank().deposit(200).statement().withdraw(100).statement().withdraw(50).statement();
```



```
var pg3 = new PiggyBank();
```

```
 .deposit(300).statement().withdraw(200).statement().withdraw(1  
.  
statement();
```

Add : 25, Patel Shopping Center, Sai Nath Road, Malad (west) ,Opp malad subway , Mumbai 64 Contact : 9820396074, 022-28809398, 9820860292  
Copyright © 2011-2020 Rajesh Patkar, All rights reserved.

