Romancing the Java Script

- Objects
- Functions

Objects


Langham Hotel
LIPPO SUN PLAZA
GOLDEN FLOWER
金菊花
尊牆面漆
Supreme Wall Finish
5

```
var foobar = {
    foo:100,
    bar:20
};
```

- `document.writeln(foobar['bar'])`

Functions

```
function testfunction(){
    return "this is a test function"
};
```

Functions ..

- var values = [12,23,45,34];

  values.sort(function(num1, num2){return num1>num2})

Method invocation pattern

```
var multiply= {
    multi_number:10,
    multi:function(num){
        return num* this.multi_number;
    }
};

document.write(multiply.multi(200));
```

```
var multiply= {
    multi_number:10,
    multi:function(num){
        return num* this.multi_number;
    },
    change_no:function(new_no){
        this.multi_number = new_no;
    }
 };
```

```
multiply.change_no(3)
document.write(multiply.multi(200));
```

Object Oriented Way..

```
var multiply = function(){
    var multi_number=10;
     return{
         multi:function(num){
             return num* multi_number;
         },
         change_no:function(new_no){
             multi_number = new_no;
         }
     };
 };

 var obj = multiply()
 document.writeln(obj.multi_number);
 document.write(obj.multi(200));
```

Function Expression

```
var multiply = ( function(){
    var multi_number=10;
    return{
        multi:function(num){
            return num* multi_number;
        },
        change_no:function(new_no){
            multi_number = new_no;
        }
    };
 } ());
•
•
•
multiply.change_no(5);
document.write(multiply.multi(200));
```

Function Invocation

```
var multiply= {
    multi_number:10,
    multi:function(num){
        return num* this.multi_number;
    }
};

multiply.add = function(num){
    return num+this.multi_number;
};
```

- document.writeln(multiply.add(200));

Inner functions??

```
var multiply= {
    multi_number:10,
   multi:function(num){
       return num* this.multi_number;
    }
 };

 multiply.add = function(num){
   var inner_add = function(){
       return num+this.multi_number;
    };
    return inner_add();
 };

document.writeln(multiply.add(200));
```

Work around

```
var multiply= {
    multi_number:10,
    multi:function(num){
        return num* this.multi_number;
    }
};

multiply.add = function(num){
    var that = this;
    var local_num = num;
    var inner_add = function(){
        return local_num+that.multi_number;
    };
    return inner_add();
};

document.writeln(multiply.add(200));
```

Scoping

```
var testfunct= function(){
    inner();
    function inner(){
        function innermost(){
            alert("Innermost function is called");
        };
        innermost();
    };
};

testfunct();
```

Variables

```
var testfunct= function(){
    alert(x);
    var x = 10;
    {
        x = 30;
    }
    alert(x);
};

testfunct();
```

Closures

- var later;

- function testfunction(){
    function outfunction(){
        function innerfunction(){
            alert(late_variable);
        };
        later = innerfunction;
    };
    outfunction();
};
alert(late_variable);
var late_variable = 100;
testfunction();
later();

Constructor Invocation

```
var Flight = function(str){
    this.status = str;
};

Flight.prototype.get_status = function(){
    return this.status;
};

var f = new Flight('ontime');

document.writeln(f.get_status());
```

Shadow

```
var Flight = function(str){
    this.status = str;
};

Flight.prototype.get_status = function(){
    return this.status;
};

Flight.prototype.status = function(){
    return "status using prototype";
};


var f = new Flight('ontime');

document.writeln(f.get_status());
```

Inheritance

```
var Flight = function(str){
    this.status = str;
};

Flight.prototype.get_status = function(){
    return this.status;
};

var BritishAir = function() {};

BritishAir.prototype = new Flight('ontime');

var ba = new BritishAir();

document.writeln(ba.get_status());
```

Apply Invocation

```
var Flight = function(str){
    this.status = str;
};

Flight.prototype.get_status = function(){
    return this.status;
};

var BritishAir = function(str) {
    this.status = str;
};

var ba = new BritishAir("onborading");
Flight.prototype.get_status.apply(BritishAir);

document.writeln(ba.status);
```

Recusion

```
var sum = {
    rec_add :function(num){
        return num> 1 ? sum.rec_add(num-1)+num :num;
    }
};
document.writeln(sum.rec_add(10));
```