Type System

03 February 2014 15:50

```
var i = 10;
var j = "Sony";
var k = true;

console.log(typeof i);
console.log('Value of j is : ' + j);
console.log(typeof k == 'boolean');

var m = '10';
var n = m + 10;
var o = 10 + parseInt(m);

console.log(n);
console.log(o);

var p = parseInt('10is my age') * 10;

console.log(p);

var q = 'ABC' * 100;

console.log(q);
console.log(typeof q);

console.log(typeof z);

var z = null;

console.log(typeof z);
```

Arrays

03 February 2014 15:50

```
var values = ['Narendran', 'Shankaran', 'Amit', 'Ajit', 'Srini'];

console.log('Totally ' + values.length + ' Participant(s) there in the class!');

values[10] = 'Abhinav';

console.log('Totally ' + values.length + ' Participant(s) there in the class!');

console.log(values);

values.push('Parvez');

console.log(values.pop());

console.log(values);

values.forEach(function(name) {
   console.log(name);
});

var employees = [ [1,'Ramkumar','Bangalore'], [2,'Rajesh','Mysore'] ];

console.log(employees[0][1]);
console.log(employees[1][1]);
```

Handling Errors and Exceptions

03 February 2014 15:50

```
try {
   var i = 10;
   var j = 10;

   if(i == j) {
      throw new Error('Invalid Operation Performed!');
   }
} catch(err) {
   console.log('Error Occurred, Details : ' + err.message);
} finally {
   console.log('Operation Completed Successfully!');
}
```

Comparison Operators

03 February 2014 14:44

```
console.log('Testing 1 ...');

var i = 10;
var j = '10';

console.log(i == j);
console.log(i === j);

console.log('Testing 2 ...');

var k = 20;
var l = '20';

console.log(k != l);
console.log(k !== l);
```

Functions - Basic

03 February 2014 14:54

```
function calculate(a, b) {
var output = a + b;

return output;
};

var result = calculate(10, 90);

console.log(result);

result = calculate(10, 90, 100, 110);

console.log(result);

function add() {
var parameters = arguments;
var result = 0;

for(var parameterIndex in parameters) {
result = parameters[parameterIndex];
}

return result;
};

var output = add(10, 20);
console.log(output);

output = add(10,20,30,40,50);
console.log(output);

function A() {
var callerName = A.caller.name;
console.log(A.caller.toString());
console.log(callerName + ' Called this function A');
}

function B() {
console.log('Working on ... B ...');

A();
}

B();

console.log();
console.log(Math.random.toString());

eval('function f() { console.log("Called!"); } f();');
```

Functions - Adventures

03 February 2014 14:55

Self-Invoking Functions

```
(function() {
console.log('Called myself!');
})();
```

Inner Junctions

```
function processor() {
function innerProcessing() {
console.log('Processing Internally Completed ...');

return 10;
};

var result = innerProcessing();

console.log('Processing Completed Successfully, Result : ' + result);
};

processor();
```

Functions that accept Functions

```
function processOrder(orderId, orderDate, orderValue, callback) {
var message = 'Order Details : ' + orderId + ', ' +
orderDate.toString() + ', ' + orderValue;

console.log('Processing Started ...');
console.log(message);

var status = 'INVALID';

if(orderValue >= 10000 && orderValue <= 50000)
status = 'VALID';

if(callback && typeof callback == 'function')
callback(status);
};

processOrder('ORD10001', new Date(), 12300,
function(status) {
console.log('Callback Called, Status : ' + status);
});
```

Functions that Accept and Return Functions

03 February 2014 15:25

```
function Execute(callback) {
var result = callback(10, 20);

return function(x, y) {
return result + x + y;
};
};

var output = Execute(function(a, b) {
return a*b;
});

console.log(output(10,20));

var output2 = output;

delete output;

console.log(output2(100,200));
console.log(output(100,200));
```

Iterators using Closures

```
function SetupIterator(values) {
var index = 0;

return function() {
if(index >= 0 && index < values.length)
return values[index++];

throw new Error('Invalid Iteration Value Specified!');
};
};

var next = SetupIterator([10,20,30,40,50]);

try {
while(true) {
console.log(next());
}
} catch(error) {
console.log('Iteration Completed!');
}
```

Objects

03 February 2014 15:56

```
var customer = {
id: 10,
name: 'Rajesh',
address: 'Bangalore',
creditLimit: 23000,
activeStatus: true,
format: function() {
var formattedMessage = this.id + ', ' +
this.name + ', ' +
this.address + ', ' +
this.creditLimit + ', ' +
this.activeStatus + ', ' +
this.activationKey;

return formattedMessage;
}
};

console.log(customer.format());

customer.activationKey = 'AC73498343';

console.log(customer.format());
```

Construction Functions (Classes)

03 February 2014 16:04

```
function Customer(id, name, address) {
this.id = id;
this.name = name;
this.address = address;
};

Customer.prototype.format = function() {
var message = this.id + ', ' +
this.name + ', ' + this.address;

return message;
};

var customer = new Customer(1, 'Shankaran', 'Bangalore' );

console.log(customer.format());

console.log('Iterating all properties ...');

for(var property in customer)
if(typeof customer[property] != 'function')
console.log(property + ' Value is : (Type is : ' +
typeof customer[property] + ') ' + customer[property]);

console.log(typeof customer);
console.log(typeof Customer);
console.log(typeof Customer.prototype);
```

Inheritance

03 February 2014 16:15

```
function Customer(id, name, address) {
this.id = id;
this.name = name;
this.address = address;
};

Customer.prototype.format = function() {
var message = this.id + ', ' +
this.name + ', ' + this.address;

return message;
};

function InternetCustomer(id, name, address, blogUrl) {
this.id = id;
this.name = name;
this.address = address;
this.blogUrl = blogUrl;
};

InternetCustomer.prototype = new Customer();
InternetCustomer.prototype.constructor = InternetCustomer;

var internetCustomer = new InternetCustomer(
1, 'Ashok', 'Bangalore', 'http://blogs.sony.com/ashok');

console.log(internetCustomer.format());
console.log(internetCustomer instanceof Customer);
console.log(internetCustomer instanceof InternetCustomer);

console.log(InternetCustomer.prototype.constructor.toString());
```

Call and Apply

03 February 2014 16:22

```
function Work(message) {
console.log(message + ', Details : ' +
this.id + ', ' + this.name);
};

var obj = {id: 10, name: 'Srinivasan' };

Work.call(obj, 'Welcome');

console.log('Using Apply ...');

Work.apply(obj, ['Welcome']);

console.log();
```

Inheritance and Polymorphic Features

03 February 2014 16:30

```
function Customer(id, name, address) {
this.id = id;
this.name = name;
this.address = address;
};

Customer.prototype.format = function() {
var message = this.id + ', ' +
this.name + ', ' + this.address;

return message;
};

function InternetCustomer(id, name, address, blogUrl) {
Customer.call(this, id, name, address);

this.blogUrl = blogUrl;
};


InternetCustomer.prototype = new Customer();
InternetCustomer.prototype.constructor = InternetCustomer;

InternetCustomer.prototype.format=function() {
var formattedMessage =
Customer.prototype.format.call(this) + ', ' + this.blogUrl;

return formattedMessage;
};

var internetCustomer = new InternetCustomer(
1, 'Ashok', 'Bangalore', 'http://blogs.sony.com/ashok');

console.log(internetCustomer.format());
```

JSON parsing and stringifying

03 February 2014 16:40

```
function Customer(id, name, address) {
this.id = id;
this.name = name;
this.address = address;
};

Customer.prototype.checkIfExists = function() {
var existingCustomerIds = ['C1', 'C2', 'C3', 'C4', 'C5'];
var isFound=false;

for each(var customerId in existingCustomerIds) {
if(customerId == this.id) {
isFound=true;
break;
}
};

return isFound;
};

var customer = new Customer('C1', 'Nagendran', 'Bangalore');
var customerJsonString = JSON.stringify(customer);

console.log(customerJsonString);

var parsedCustomerObject = JSON.parse(customerJsonString);

try {
console.log("Status : " +
parsedCustomerObject.checkIfExists()); // Error
} catch (error) {}

console.log('Status : (Working) ... ' +
Customer.prototype.checkIfExists.call(parsedCustomerObject));
```

AJAX Calls

03 February 2014 16:59

```
function Customer(id, name, creditLimit) {
    this.Id = id;
    this.Name = name;
    this.CreditLimit = creditLimit;
}

Customer.prototype.toRow = function () {
    var row =
        "<tr>" +
            "<td>" + this.Id + "</td>" +
            "<td>" + this.Name + "</td>" +
            "<td>" + this.CreditLimit + "</td>" +
        "</tr>";

    return row;
};

function prepareTable(customers) {
    var table =
        "<table border='1'>" +
            "<thead>" +
                "<tr>" +
                    "<th> Customer Id </th>" +
                    "<th> Name </th>" +
                    "<th> Credit Limit </th>" +
                "</tr>" +
            "</thead>";

    table += "<tbody>";

    for (var index in customers) {
        var customer = customers[index];

        table += customer.toRow();
    }

    table += "</tbody>";
    table += "</table>";

    return table;
}

function prepareCustomers(customers) {
    var parsedCustomers = [];

    $.each(customers, function (index, item) {
        var customer = $.extend(new Customer(), item);

        parsedCustomers.push(customer);
    });

    return parsedCustomers;
}
```

```
function loadCustomers() {
    var customersJsonUrl = "/Customers.json";
    var httpRequest = new XMLHttpRequest();

    httpRequest.onreadystatechange = function () {
        if (httpRequest.readyState == 4 &&
            httpRequest.status == 200) {
            var responseText = httpRequest.responseText;
            var parsedCustomers = prepareCustomers(JSON.parse(responseText));
            var table = prepareTable(parsedCustomers);
            var contentElement = document.getElementById("content-panel");

            contentElement.innerHTML = table;
        }
    };

    httpRequest.open('GET', customersJsonUrl, true);
    httpRequest.send();
}
```