**\*\*\*\*HTML\*\*\*\*\*\*\*\*Java Script Functions\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>

<script language=*"javascript"* type=*"text/javascript"*>

**function** sayHello(message)

{

alert(message+" Everyone....");

}

**var** sayNewYearWish = **function**(message)

{

alert(message +" EveryOne ");

};

**var** sayHappyBirthDay = **new** Function("message", "alert(message +' To You ');");

</script>

<title>Insert title here</title>

</head>

<body bgcolor=*"cyan"*>

<input type=*"button"* name=*"buttoenSayHi"* value=*"Say Hello"*

onclick="sayHello('Hello');"/>

<input type=*"button"* name=*"buttonNewYearWish"* value=*"Happy New Year"*

onclick="sayNewYearWish('Happy New Year');"/>

<input type=*"button"* name=*"buttoenHappyBirthDay"* value=*"BirthDay Wish"*

onclick="sayHappyBirthDay('Happy Birth Day');"/>

</body>

</html>

**\*\*\*\*HTML\*\*\*\*Object Creation Using Constructor Function Syntax[1]\*\*\***

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>

<title>Insert title here</title>

<script language=*"javascript"* type=*"text/javascript"*>

**function** Person(personName)

{

**this**.personName = personName;

**var** personAge;

**this**.talk = **function**()

{

alert("Hello EveryOne : My Name Is : " + personName+

" And I Am "+personAge+ " Year Old" );

}

**this**.setAge = **function**(tempAge)

{personAge = tempAge; }

**this**.getAge = **function**(){ **return** personAge; }

}

**function** createEmp()

{

**var** p1=**new** Person("Samir");

p1.setAge(30);

p1.talk();

**var** p2=**new** Person("Punit");

p2.setAge(34);

p2.talk();

}

</script>

</head>

<body bgcolor=*"cyan"*>

<input type=*"button"* name=*"buttoenShowEmpInfo"* value=*"Show Emp Info"*

onclick="createEmp();" />

</body>

</html>

**Ex-2…………………………………………………………………………….**

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>

<script language=*"javascript"* type=*"text/javascript"*>

**function** displayQuote()

{ alert(**this**.memorableQuote); }

**var** williamShakespeare =

{

"memorableQuote": "It is a wise father that knows his own child.","sayIt": displayQuote

};

**var** markTwain =

{"memorableQuote":"Golf is a good walk spoiled.","sayIt" : displayQuote};

**var** oscarWilde =

{"memorableQuote": "True friends stab you in the front.","sayIt" : displayQuote };

// ouch!

</script>

<title>Insert title here</title>

</head>

<body bgcolor=*"cyan"*>

<input type=*"button"* name=*"buttoenSay1"* value=*"What Shakespeare Says"*

onclick="williamShakespeare.sayIt();"/>

<input type=*"button"* name=*"buttonNewYearWish"* value=*"What markTwain Says"*

onclick="markTwain.sayIt();"/>

<input type=*"button"* name=*"buttoenHappyBirthDay"* value=*"Call displayQuote.call(oscarWilde)"*

onclick="displayQuote.call(oscarWilde);"/>

</body>

**\*\*\*\*\*\*\*CHROME\*\* Object Creation Using Constructor Function Syntax[1]\*\*\*\*\*\***

function Employee(empId,empName,empSal)

{

this.empId=empId,

this.empName=empName,

this.empSal=empSal,

this.calcEmpSal=function()

{

return (empSal+2000);

}

}

**<<<undefined>>>**

...........................................................................................................

var e1=new Employee(111,"Vaishali",5000);

**<<<undefined>>>**

..........................................................................................................

e1.empId

**<<<111>>>**

............................................................................................................

e1.empName

**<<<"Vaishali">>>**

.............................................................................................................

e1.empSal

**<<<5000>>>**

...........................................................................................................

e1.hasOwnProperty('empId')

**<<<true>>>**

.............................................................................................................

e1.calcEmpSal

**<<<function ()**

**{**

**return (empSal+2000);**

**}**

**>>>**

...........................................................................................................

e1.calcEmpSal()

**<<<7000>>>**

.........................................................................................................

e1.hasOwnProperty("calcEmpSal");

**<<true>>**

........................................................................................................

typeof Employee.prototype

**<<<"object">>>**

.....................................................................................................

Employee.prototype.gender="Female"

**<<<"Female">>>**

..................................................................................................................

Employee.prototype.sayHello=function(){return "Happy New Year"}

**<<<function (){return "Happy New Year"}>>>**

.................................................................................................................

e1.hasOwnProperty("sayHello");

**<<<false>>>**

............................................................................................................

e1.gender

**<<<"Female">>>**

.................................................................................................................

e1.sayHello()

**<<<"Happy New Year">>>**

**\*HTML\*\*\* Empty Object Creation Using new Object() Syntax[2]\*\*\*\***

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>

<title>Insert title here</title>

<script language=*"javascript"* type=*"text/javascript"*>

**function** createEmp()

{

//empty Object

**var** employee1=**new** Object();

employee1.empId="777";

employee1.empName="Vaishali";

employee1.empSal=**function**(empSal)

{**return** empSal; };

employee1.calcEmpSal = **function**()

{**return** (**this**.empSal(5000) + 2000); }

alert (employee1.empId+" : "+employee1.empName+ " : "+

employee1.empSal(5000)+" Total Sal : "+employee1.calcEmpSal())

//another syntax

**var** employee2 = {}; // equivalent to new Object()

employee2["empId"] = 888;

employee2["empName"] = "Samir";

employee2["empSal"]=**function**(empSal)

{ **return** empSal; };

employee2["calcEmpSal"] = **function**()

{**return** (**this**.empSal(15000) + 2000); }

alert (employee2.empId+" : "+employee2.empName+ " : "+

employee2.empSal(15000)+" Total Sal : "+employee2.calcEmpSal())

}

</script>

</head>

<body bgcolor=*"cyan"*>

<input type=*"button"* name=*"buttoenShowEmpInfo"* value=*"Show Emp Info"*

onclick="createEmp();" />

</body>

</html>

**\*\*\*\*CHROME\*\* Empty Object Creation Using new Object() \*\*\*\*\*\*\*\***

**\*\*\*\*RunTime Object In Chrome\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

var employee2={} - This is same as - var emp2=new Object();

<<undefined>> - -<<<undefine>>>

………………………………………………………………………….

employee2.empId=821219; - This is same as- emp.ID=821230

<<<821219>>> -- <<<821230>>>

……………………………………………………………………………..

employee2.empName="vaishali"

<<"vaishali">>>

………………………………………………………………………………….

employee2.empSal=5000;

<<<5000>>>

………………………………………………………………..

employee2.empSal=**function**(empSal)

{ **return** empSal; };

**………………………………………………………………………………**

employee2.calcEmpSal= **function**()

{ **return** (**this**.empSal(15000) + 2000); }

<<<17000>>>

…………………………………………………

Employee2

<<<Object {ID: 821219, name: "vaishali"}>>>

**.................................................................................................................**

**F12 for Developer window and ALL tab for enable Console**

for(key in employee2)

{

console.log(" KEY : "+key+ " Value :"+employee2[key]);

}

KEY : empId Value :821219 VM178:4

KEY : empName Value :vaishali VM178:4

KEY : empSal Value :function (empSal)

{return empSal; }

KEY : calcEmpSal Value :function ()

{

return (this.empSal(15000) + 2000);

} VM178:4

<<<undefined>>>

........................................................................................................

delete employee2["empName"];

<<<true>>>

…………………………………………………………………………………

employee2

<<Object {empId: 821219, empSal: function, calcEmpSal: function}>>>

**\*\*\*\*\*\*\*\*\*\*\*\*\*Check for Non Existing property\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

employee2.gender;

<<<undefined>>>

…………………………………………………………………………………

employee2.gender ==undefined

<<<true>>>

……………………………………………………………………………..

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*check for existence of key\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

"gender" in employee2;

<<<false>>>

…………………………………………………………………………..

employee2.gender="Female"

<<<"Female>>>

………………………………………………………………………………..

employee.gender ==undefined

<<<false>>>

………………………………………………………………………………

"gender" in employee;

<<<true>>>

……………………………………………………………………………….

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Object References\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

var emp=employee2

<<<undefined>>>

………………………………………………………………………………

emp.ID="vs821219"

<<<"vs821219">>>

………………………………………………………………………

Employee2.ID

<<<"vs821219">>>

................................................................................................

**HTML\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Object Creation Syntax[3]\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**CHROME \*\*\*\*\*\*\*\*\*Object Creation Syntax[3]\*\*\*\*\*\*\*\***

**Ex-1**

var Person=

{

panNo : 0,

personName : "Guest",

setName : function()

{

this.personName=prompt("Enter UR Name ");

this.panNo=prompt("enter UR PAN No ");

},

getName : function()

{ console.log(" Welcome "+this.personName+" UR Pan No is :"+this.panNo); }

}

<<<undefined>>>

………………………………………………………………………………………

Person.setName();

<<<undefined>>>

…………………………………………………………………………………..

Person.getName();

<<<Welcome vvv UR Pan No is :99 >>>

**Ex-2……………………………………………………………………………**

var foo=

{

name:"Guest",

setName: function()

{

this.name=prompt("Enter UR name ");

},

getName:function()

{console.log(" Hello : "+this.name); }

};

<<<Undefine>>>

...........................................................................................

foo.setName();

<<<undefined>>>

........................................................................

foo.getName();

<<<Hello : Vaishali >>>

<<<undefined>>>

……………………..**Constructor Function…………….**

function foo(empName)

{

this.empName=empName;

this.getName=function()

{

return " Hello : "+this.empName;

}

}

<<<Undefine>>>

....................................................................................................

var f1=new foo("vaishali");

<<<undefined>>>

...............................................................................................................

f1.getName();

<<<" Hello : vaishali">>>

................................................................................................................

**\*\*\*\*\*\*Prototype Inheritance\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

In most languages, there are classes and objects. Classes inherit from other classes.

In JavaScript, the inheritance is prototype-based. That means that there are no classes.

Instead, an object inherits from another object Smile

Every JavaScript object has a prototype. The prototype is also an object.

All JavaScript objects inherit their properties and methods from their prototype.

JavaScript Prototypes

All JavaScript objects inherit the properties and methods from their prototype.

Objects created using an object literal, or with new Object(), inherit from a prototype called Object.prototype.

Objects created with new Date() inherit the Date.prototype.

The Object.prototype is on the top of the prototype chain.

All JavaScript objects (Date, Array, RegExp, Function, ....) inherit from the Object.prototype

**\*\*\*\*\*\*\*\*\*Object Inheritance Using \_\_Proto\_\_\*\*\*Foo\*\*\*\*\*\*\*\*\*\*\*\***

var foo=

{

name:"Guest",

setName: function()

{

this.name=prompt("Enter UR name ");

},

getName:function()

{

console.log(" Hello : "+this.name);

}

};

...................................................................................................

var bar=

{

empId : " Your Emp ID",

setEmpId : function()

{

this.empId=prompt("Enter UR ID : ");

},

getAllInfo :function()

{

return this.empId +" : "+ this.name;

}

}

<<<Undefine>>

..................................................................................................

bar.\_\_proto\_\_=foo;

<<<Object {name: "Vaishali", setName: function, getName: function}>>>

..................................................................................................................

bar.setName();

<<<undefined>>>

......................................................................................................................

bar.getName();

<<<Hello : Samir >>>

.........................................................................................................................

bar.setEmpId();

<<<undefined>>>

.....................................................................................................................

bar.getAllInfo();

<<<"666 : Samir">>>

.....................................................................................................................

for(property in bar)

{

if(bar.hasOwnProperty(property))

{

console.log("Own Property :"+property);

}

else

{

console.log("Inherited Property :"+property);

}

}

<<<

Own Property :empId VM191:6

Own Property :setEmpId VM191:6

Own Property :getAllInfo VM191:6

Inherited Property :name VM191:10

Inherited Property :setName VM191:10

Inherited Property :getName VM191:10

Undefined>>>

……………………………………………………………………

**\*\*\*\*\*\*\*\*\*\*\*Object Inheritance Using Object.create() method\*\*\*\*\*\***

var foo2=

{

name:"Guest",

setName: function()

{

this.name=prompt("Enter UR name ");

},

getName:function()

{

console.log(" Hello : "+this.name);

}

};

<<<Undefine>>>

..............................................................................................................

var bar2=Object.create(foo2);

<<<undefined>>>

................................................................................................................

bar2.empId=999;

<<<999>>>

..................................................................................................................

bar2.setName();

<<<undefined>>>

...........................................................................................................................

bar2.getName();

Hello : Anju

<<<undefined>>>

.............................................................................................................................

bar2.setEmpId=function()

{ return prompt("Enter UR ID : ");}

<<<function ()

{ return prompt("Enter UR ID : ");}

>>>

........................................................................................................................... .

bar2.setEmpId();

<<<"555">>>

..............................................................................................................................

bar2.empId;

<<<999>>>

............................................................................................................................

bar2.getAllInfo=function(){ return this.empId +" : "+ this.name;}

<<<function (){ return this.empId +" : "+ this.name;}>>>

.......................................................................................................................

bar2.getAllInfo();

<<<"999 : Anju">>>

…………………………………………………………………..

Object.getPrototypeOf(bar2);

<<<Object {name: "Guest", setName: function, getName: function}

…………………………………………………………………………….

Object.getPrototypeOf(bar2)==foo2;

<<<true>>

……………………………………………………………………………..

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*Object Inheritance Using Prototype\*\*\*\*\*\*\*\*\*\*\*\*\*\***

function Employee(empId,empName,empSal)

{

this.empId=empId;

this.empName=empName;

this.empSal=empSal;

this.calcEmpSal=function()

{

return (this.empSal+2000);

}

};

<<<undefined>>>

......................................................................................................

var amitE1=new Employee(111,"Amit",10000);

<<<undefined>>>

...........................................................................................................

amitE1;

<<<Employee {empId: 111, empName: "Amit", empSal: 10000, calcEmpSal: function}>>>

...........................................................................................................................

function WageEmp(noOfHrs,ratePerHrs)

{

this.noOfHrs=noOfHrs;

this.ratePerHrs=ratePerHrs;

this.calcWageEmpSal=function()

{

return (this.calcEmpSal()+(this.ratePerHrs\*this.noOfHrs));

}

}

<<<undefined>>>

…………………………………………………………………..

WageEmp.prototype=new Employee(821219,"vaishali",5000);

<<<Employee {empId: 821219, empName: "vaishali", empSal: 5000, calcEmpSal: function}>>>

…………………………………………………………………………

var vaiObj=new WageEmp(5,200);

<<<undefined>>>

…………………………………………………………………………………………………..

vaiObj.empId;

<<<821219>>>

………………………………………………………………………………………..

vaiObj.empName

<<<"vaishali">>>

………………………………………………………………………………………………

vaiObj.empSal;

<<<5000>>

……………………………………………………………………………..

vaiObj.calcEmpSal();

<<<7000>>>

……………………………………………………………………………………

vaiObj.noOfHrs;

<<<5>>>

………………………………………………………………………….

vaiObj.ratePerHrs

<<<200>>>

……………………………………………………………………….

vaiObj.calcWageEmpSal();

<<<8000>>>

\*\*\*\*\*\*\*\*\*\*\*\*Closure Function\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Variables created **without** the keyword **var**, are always global, even if they are created inside a function.

The variable add is assigned the return value of a self-invoking function.

The self-invoking function only runs once. It sets the counter to zero (0), and returns a function expression.

This way add becomes a function. The "wonderful" part is that it can access the counter in the parent scope.

This is called a JavaScript closure. It makes it possible for a function to have "private" variables.

The counter is protected by the scope of the anonymous function, and can only be changed using the add function.

|  |  |
| --- | --- |
| Note | A closure is a function having access to the parent scope, even after the parent function has closed. |

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>

<script language=*"javascript"* type=*"text/javascript"*>

function sayHello(message)

{

alert(message+" Everyone....");

}

var sayNewYearWish = function(message)

{

alert(message +" EveryOne ");

};

var sayHappyBirthDay = new Function("message", "alert(message +' To You ');");

</script>

<title>Insert title here</title>

</head>

<body bgcolor=*"cyan"*>

<input type=*"button"* name=*"buttoenSayHi"* value=*"Say Hello"*

onclick="sayHello('Hello');"/>

<input type=*"button"* name=*"buttonNewYearWish"* value=*"Happy New Year"*

onclick="sayNewYearWish('Happy New Year');"/>

<input type=*"button"* name=*"buttoenHappyBirthDay"* value=*"BirthDay Wish"*

onclick="sayHappyBirthDay('Happy Birth Day');"/>

<button type=*"button"* onclick="myFunction()">Count!</button>

<p id=*"demo"*>0</p>

</br>

<script>

var add = (

function ()

{

var counter = 0;

return function ()

{return counter += 1; }

}

)

();

function myFunction()

{

document.getElementById("demo").innerHTML = add();

}

</script></body></html>

**Syntax[2\*\*\*\*\*\*\*Another Syntax [2]\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

var Dept=function()

{

}

<<<undefined>>>

...............................................................................................................

Dept.prototype.deptCode=12

..................................................................................................................

var d1=new Dept();

<<undefined>>

..................................................................................................................

d1.deptCode

<<"12">>

........................................................................................................

Dept.prototype.getDeptLocation=function(){return "Pune Hinjwadi"}

<<<function (){return "Pune Hinjwadi"}>>>

.............................................................................................

d1.getDeptLocation()

<<<"Pune Hinjwadi">>>

............................................................................................

Dept.prototype={deptCode:"10",deptName:"Corporate University"}

<<<Object {deptCode: "10", deptName: "Corporate University"}>>>

............................................................................................................

d1.deptName

<<"Corporate University">>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

If you take a look at the source code of the Microsoft AJAX Library, you’ll see that the library’s authors use a similar technique to implement namespaces (take a look at the implementation of the static method Type.registerNamespace). See the sidebar "OOP and ASP.NET AJAX" for more information.

**Should You Code JavaScript This Way?**

You’ve seen that JavaScript supports object-oriented programming just fine. Although it was designed as a prototype-based language, it is flexible and powerful enough to accommodate the class-based programming style that is typically found in other popular languages. But the question is: should you code JavaScript this way? Should you code in JavaScript the way you code in C# or C++, coming up with clever ways to simulate features that aren’t there? Each programming language is different, and the best practices for one language may not be the best practices for another.

In JavaScript, you’ve seen that objects inherit from objects (as opposed to classes inheriting from classes). So it is possible that making a lot of classes using a static inheritance hierarchy is not the JavaScript way. Maybe, as Douglas Crockford says in his article "[Prototypal Inheritance in JavaScript](http://javascript.crockford.com/prototypal.html)", the JavaScript way of programming is to make prototype objects, and use the simple object function below to make new objects, which inherit from that original object:

function object(o) {

function F() {}

F.prototype = o;

return new F();

}

Then, since objects in JavaScript are malleable, you can easily augment the object after its creation with new fields and new methods as necessary.

This is all good, but it is undeniable that the majority of developers worldwide are more familiar with class-based programming. Class-based programming is here to stay, in fact. According to the upcoming edition 4 of ECMA-262 specification (ECMA-262 is the official specification for JavaScript), JavaScript 2.0 will have true classes. So JavaScript is moving towards being a class-based language. However, it will probably take years for JavaScript 2.0 to reach widespread use. In the meantime, it’s important to know the current JavaScript well enough to read and write JavaScript code in both prototype-based style and class-based style.

**Putting It into Perspective**

With the proliferation of interactive, client-heavy AJAX applications, JavaScript is quickly becoming one of the most useful tools in a .NET developer’s arsenal. However, its prototypal nature may initially surprise developers who are more used to languages such as C++, C#, or Visual Basic. I have found my JavaScript journey a rewarding experience, although not entirely without frustration along the way. If this article can help make your experience smoother, then I’m happy, for that’s my goal.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Closure\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

<!DOCTYPE html>

<html>

<head>

</head>

<body>

<input type="button" value="Print" onclick="printYear()"/>

<input type="button" value="Increment" onclick="incrementYear()"/>

<input type="button" value="Set" onclick="setYear(2015)"/>

</body>

<script>

(

function()

{

var year = 2015;

printYear = function()

{

alert(year);

}

incrementYear = function()

{

year++;

}

setYear = function(yr){ year = yr; }} )();

</script></html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

<title>Insert title here</title>

<script language="javascript" type="text/javascript">

function increaseCount()

{

document.getElementById("divId").innerHTML=plus();

}

/\* \*

\* var count=0;

function add()

{

return count +=1;

}\*/

/\*

function add()

{

var count = 0;

function plus()

{

count += 1;

}

plus();

return count;

}\*/

/\*var add =

(

function ()

{

var count = 0;

return function ()

{

return count += 1;

}

}

)();

\*/

/\*(

function()

{

var count = 0;

plus=function()

{

count += 1;

return count;

}

}

) ();

\*/

var add=

(

function()

{

var count = 0;

plus=function()

{

count += 1;

return count;

}

}

) ();

</script>

</head>

<body bgcolor="pink">

<input type="button" name="countButton"

value="Increase Count " onClick="increaseCount();"/>

</br>Count:<div id="divId">0</div>

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*