Object based programming in JavaScript

Some terms

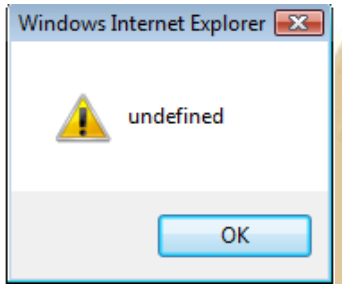
- Object
- Abstraction
- Class
- Encapsulation

Creating objects in JavaScript

- Using built-in data type called **`Object`**
- Any number of properties can be added to an object at any time.

```
<SCRIPT type="text/JavaScript">
obj = new Object;
obj.x = 1;
obj.y = 2;
alert(obj.x + " "+ obj.y);
</SCRIPT>
```

What will
`alert(obj.k);`
display?



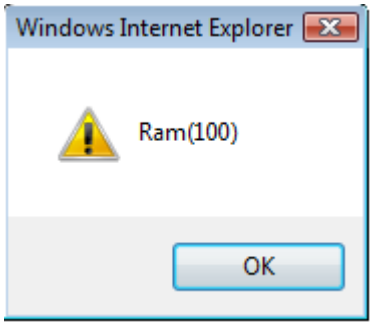
Class

- A class is defined using function.
- When a function is called with the **new** operator, the function is considered as the constructor for that class!

```
<SCRIPT type="text/JavaScript">
function Person(name,eid){
this.name=name;
this.eid =eid;
}
```

attributes

constructor




```
p=new Person("Ram",100);
alert(p.name+ "("+ p.eid+")");
</SCRIPT>
```

Methods

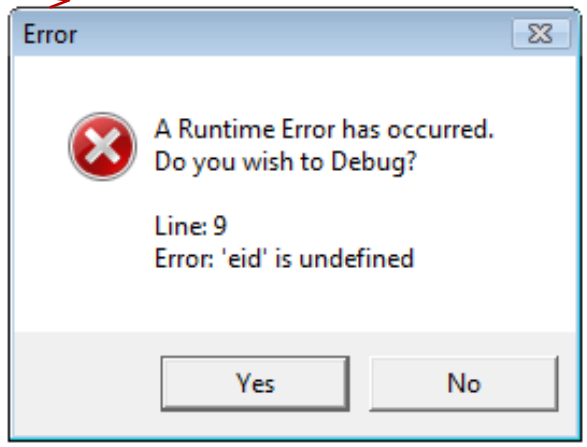
- To define a method
  <classname>.prototype.<methodName>=
  function syntax is used.

```
<SCRIPT type="text/JavaScript">
function Person(name,eid){
this.name=name;
this.eid=eid;}
Person.prototype.display = function(){
alert(this.name+ "("+ this.eid+")"); }
Person.prototype.change = function(name){
this.name=name;
}
p=new Person("Ram",100);
p.display();
p.change("Ramakrishna");
p.display();</SCRIPT>
```

This is used to distinguish between local variable and member variables in C++ and Java. Is it not the same here?

Yes that is true in JavaScript too. But here, once you have defined member variable using `this`, you must always access it using `this` (even if there are no clashing local variables.

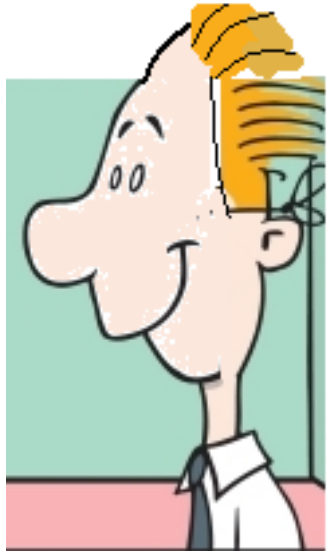
```
<HTML>
<BODY>
<SCRIPT type="text/JavaScript">
function Person(){
this.name="ss";
this.eid=22;
}
Person.prototype.display = function()
{
alert(name+ "("+ eid+")");
}
p=new Person("Ram",100);
p.display();
</SCRIPT>
</BODY></HTML>
```



Correct way is this:

```
alert(this.name+ "("+ this.eid+")");
```

What about access specifiers? Can I have private member



A local variable of the function is considered as private member.

But local variable is accessible only where function is defined. What if the member function wants to access the private member?

For this we need to make the method as **a Privileged Method**

Privileged and Private Methods

- A privileged method is a method having access to private members of a class.

- Accessible to the public methods and the outside

- Privileged methods are assigned with **this** within the constructor.

- Private methods are inner functions of the constructor

Privileged Method Sample

```
<HTML><BODY>
<SCRIPT type="text/JavaScript">
function Person(name,eid){
var fullid=name+eid;
this.name=name;
this.eid=eid;
this.display = function(){
alert(fullid);};
}
Person.prototype.change =
  function(name){
this.name=name;}
p=new Person("Ram",100);
p.display();</SCRIPT></BODY></HTML>
```

private member

Privileged Method

Private Method

```
<SCRIPT type="text/JavaScript">
function Person(name,eid){
var fullid=name+eid;
this.name=name;
this.eid=eid;
this.display = function() {
changeID(7);
alert(fullid);};
var changeID = function(eid)  {
this.eid=eid;
alert(this.eid);};
}
Person.prototype.change = function(name){
//changeID(7);
this.name=name;}
p=new Person("Ram",100);
p.change("Ramakrishna");
p.display();   </SCRIPT>
```

private member

Error- object expected

Arrays

- Declaration: arrays are objects in JavaScript

```
a= new Array(3);
a[0]=1;
a[1]=2;
a[2]=3;
```

- A single array can hold any kind of data

```
junk= new Array(3);
junk[0]="Sunday";
junk[1]=0    ;
junk[2]=true;
```

- Initialized array

```
week=new Array("sun","mon","tue","wed","thu","fri","sat");
alert(week[0]); →sun
alert(week); →sun,mon,tue,wed,thu,fri,sat
```

- Array length:

```
a.length→ 3
```

Adding elements to an array

- Add using subscript. Array size is incremented dynamically

```
a= new Array();
a[4]=4;
a.length is 5
```

- **push()** to automatically adds elements to the end of the array

```
week=new Array("sun","mon","tue","wed);
week.push('thu');
week.push('fri','sat');
alert(week);
```

→**sun,mon,tue,wed,thu,fri,sat**

•**unshift** to automatically adds elements to the beginning of the array

```
nums=new Array(3,4,5,6);
nums.unshift(0,1,2);
alert(nums);
```

Removing elements to an array

- **`pop()`** to remove element from the end of the array

```
week=new Array("sun","mon","tue","wed");
week.pop();
alert(week);→sun,mon,tue
```

- **`shift()`** to remove element from the beginning of the array

```
nums=new Array(3,4,5,6);
nums.shift()
alert(nums);→ 4,5,6
```

splice()

- Allows adding/removing from any index position
- **splice(indexposition, numberOfItemsToDelete,[item(s) to be added]**
- **Example 1:**

```
nums=new Array(3,4,5,6);
nums.splice(2,1,10);
alert(nums);→ 3,4,10,6
```

- **Example 2:**

```
nums=new Array(3,4,5,6);
nums.splice(2,1);
alert(nums);→ 3,4,6
```

- **Example 3:**

```
nums=new Array(3,4,5,6);
nums.splice(1,2,11,12,13);
alert(nums);→ 3,11,12,13,6
```

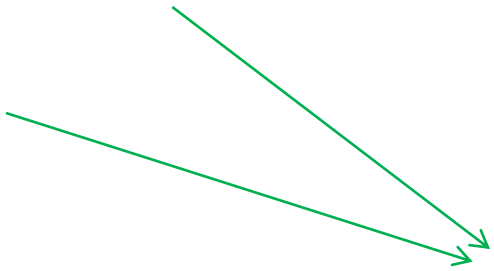
for..in statement

```
<html><body>
  <script type="text/javascript">
  var x;
  flowers=new Array(“rose”,”lilly”,”lotus”);
  for (x in flowers)  {
    alert(flowers[x]);
    }
  </script>
  </body>
  </html>
```

Accessing arguments of a function

- Any number of arguments can be passed to a JavaScript function.
- To access arguments inside the function, **`arguments`** member can be used.
- Functions in JavaScript is actually an object.

```
<script>
function sum(){
total=0
for(j=0;j<sum.arguments.length;j++){
total+=sum.arguments[j];}
alert(total);
}
</script>
```



Or just **`arguments`**