

JavaScript Workshop

Creating JavaScript Objects

JavaScript Object Basics

- JavaScript is object based not object oriented: JS lacks some of the features that are required in a fully object oriented language such as class-based inheritance and private members. JS does not support classes and hence does not support inheritance.
- JavaScript is a prototype based language that uses objects as prototypes for other objects: a new object can replicate the behavior of an existing object by cloning it.
- Functions (constructors) in JS are the blueprints for objects (instead of classes).
- Each JavaScript object is based on another object known as, appropriately enough, Object. So every new object you create is actually an “extension” of the base Object.

How are objects created?

- JS provides for a constructor (type of function), via `Object`, that allows developers to construct new objects. It is the `Object` constructor that then allocates the memory for the object, including all its properties and methods.
- The `Object` also provides a prototype property (and other methods), which enables you to extend any object, including the built-in ones such as `String` and `Number`. It is this prototype that's used to derive new object methods and properties --- not class inheritance.

```
Function Car() { }           // constructor: defines the object structure
```

```
var objCar = new Car();     // creates the object
```

How does Prototyping work?

- Taking an existing object (or a newly created one) and “extend” its functionality by adding properties and/or methods:

```
// All Strings objects would have a new method called trim  
String.prototype.trim = function() {  
    return (this.replace(/^[\\s\\xA0]+/, "").replace(/[\\s\\sA0]+$/, ""));  
}
```

```
var sObj = new String("  This is a string  ");  
sTxt = sObj.trim();
```

```
// Could have done this (would only apply to the one str instance):  
var str = "  This is a string  ";  
str.trim = function()...
```

Creating an object: new Object()

- Standard way of creating an object. Good as long as you do not need multiple instances of the object:

```
var employee = new Object();
```

```
employee.hours = 80;
```

```
employee.wages = 22.50;
```

```
employee.getSalary = function() {  
    return this.hours * this.wages;
```

```
};
```

```
alert(employee.hours + " " + employee.wages);
```

```
alert(employee.getSalary());
```

Creating an object: constructor

- The object is created by the developer and mimics how a class is instantiated in other programming languages though it is inefficient because the functions (methods) are duplicated every time constructor is created.

```
function employee (hours, wages) {      // object constructor
```

```
    this.hours = hours;
```

```
    this.wages= wages;
```

```
    this.getSalary = function() {  
        return this.hours * this.wages;
```

```
    }  
};
```

```
var emp = new employee(80, 22.50);    // instantiating object  
alert(emp.getSalary());
```

Creating an object: prototype

- The object is created by the developer as well and is more efficient, but the developer cannot pass initial values of properties by passing argument to the constructor.

```
function employee () { }
```

```
employee.prototype.hours = 80;
```

```
employee.prototype.wages= 22.50;
```

```
employee.prototype.getSalary = function() {
```

```
    return this.hours * this.wages;
```

```
}
```

```
var emp = new employee();
```

```
alert(emp.getSalary());
```

Creating an object: constructor and prototype

- This hybrid way uses the constructor to define all non-function properties and use the prototype to define the function properties (methods) of the object.

```
function employee (hours, wages) {
```

```
    this.hours = hours;
```

```
    this.wages = wages;
```

```
}
```

```
employee.prototype.getSalary = function () {
```

```
    return this.hours * this.wages;
```

```
}
```

```
var emp = new employee (80, 22.50);
```

```
alert(emp.getSalary());
```

Creating an object: object literal notation

- A shorthand of the new Object() syntax. Many JavaScript libraries are constructed this way:

```
var employee = {  
  
    hours: 80,  
    wages: 22.50,  
  
    getSalary: function() {  
        return this.hours * this.wages;  
    }  
};  
  
alert(employee.hours + " " + employee.wages);  
alert(employee.getSalary());
```

Why care about creating JavaScript objects?

- Allows for better reusability.
- Easier to edit and maintain.
- Necessary for larger multi-developer applications.
- JavaScript libraries use objects extensively.
- Ajax development (i.e. libraries) also use it.

- Non JS related: Every programming language uses objects.
Porting your skills to other technologies
improves marketability. \$\$\$

- JavaScript Workshop website:
<http://www.javascriptworkshop.com>
- Any questions?
frank@javascriptworkshop.com - Frank Stepanski
- Reference material:

[Learning JavaScript](#) (chapter 11)

[Pro JavaScript Techniques](#) (chapter 2)

[Professional JavaScript for Web Developers](#) (chapter 3)