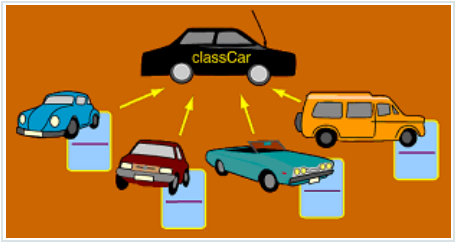
Object-Oriented Programming With JavaScript

By Viral Patel on July 16, 2009



Object-Oriented programming is one of the widely used programming paradigm that uses abstraction to create model based on real world. It is a model organized around "objects" rather than "actions" and data rather than logic. Historically, a program has been viewed as a logical procedure that takes input data, processes it, and produces output data. Object-oriented programming (OOP) uses **"objects"** – data structures consisting of **datafields** and **methods** – and their interactions to design applications and computer programs. Each object can be seen as a tiny machine which is responsible for the set of task assign to it.

Today, many popular programming languages (such as Java, JavaScript, C#, C++, Python, PHP etc) support **object-oriented programming** (OOP).

JavaScript has strong object-oriented programming capabilities. Although there is differences in object-oriented capability of javascript compared to other languages.

Terminology

First let us see few terminologies that we use in object-oriented programming.

Class: Defines the characteristics of the Object.
Constructor: A method called at the moment of instantiation.
Object: An Instance of a Class.
Method: An Object capability as walk.
Property: An Object characteristic, such as color.
Inheritance: A Class can inherit characteristics from another Class.
Encapsulation: A Class defines only the characteristics of the Object, a method defines only how the method executes.
Abstraction: The conjunction of complex inheritance, methods, properties of an Object must be able to simulate a reality model.
Polymorphism: Different Classes might define the same method or property.

The Class in JavaScript

Unlike Java, C++ etc, JavaScript does not contains class statement. JavaScript is a prototype-based language. JavaScript uses functions as classes. Defining a class is as easy as defining a function. In the example below we define a new class called Car.

```
//Define the class Car
function Car() { }
```

The Object (Class Instance) in JavaScript

For creating instances of any class i.e. objects use **new** keyword. For example, in below code snippet we created two instances of class Car.

```
//Define the class Car
function Car() { }

var car1 = new Car();
var car2 = new Car();
```

The Constructor in JavaScript

In object oriented methodology, a Constructor is a method that is used to initiate the properties of any class instance. Thus the constructor gets called when the class is instantiated.
As in JavaScript there is no class keyword and the function serves as class definition, there is no need of defining constructor explicitly. The function defined for the class acts as constructor. For example, in

Latest Posts

1. HTML5 DataList Example
2. Auditing DML changes in Oracle
3. Java 8 Lambda Expressions Tutorial with

following code snippet we have called an alert statement when class Car is instantiated.

```
//Define the class Car
function Car() {
    alert("Class CAR Instantiated");
}

var car1 = new Car();
var car2 = new Car();
```

The Property (object attribute) in JavaScript

Properties are the variable that are member of an object and can define the state of any instance of Class. Property of a class can be accessed within the class using **this** keyword. For example, in following code snippet we have assign a property *speed* to Car.

```
//Define the class Car
function Car(speed) {
    alert("Class CAR Instantiated");
    this.speed = speed;
}

var car1 = new Car(40);
var car2 = new Car(60);

alert("Car1 Speed: " + car1.speed);
alert("Car2 Speed: " + car2.speed);
```

One thing we should note here is that for inheritance works correctly, the Properties should be set in the prototype property of the class (function). The above example becomes:

```
//Define the class Car
function Car(speed) {
    alert("Class CAR Instantiated");
    this.speed = speed;
}
Car.prototype.speed= 'Car Speed';

var car1 = new Car(40);
var car2 = new Car(60);

alert("Car1 Speed: " + car1.speed);
alert("Car2 Speed: " + car2.speed);
```

The methods in JavaScript

To define methods in a Class, all you have to do is just define a attribute and assign an function to it. For example, in below code snippet we defined a method `setSpeed()` for class `Car`.

```
//Define the class Car
function Car(speed) {
    alert("Class CAR Instantiated");
    this.speed = speed;
}
Car.prototype.speed= 'Car Speed';
Car.prototype.setSpeed = function(speed) {
    this.speed = speed;
    alert("Car speed changed");
}

var car1 = new Car(40);
var car2 = new Car(60);

car1.setSpeed(120);
car2.setSpeed(140);
```

Inheritance in JavaScript

JavaScript supports single class inheritance. To create a child class that inherit parent class, we create a Parent class object and assign it to the Child class. In following example we created a child class Ferrari from parent class Car.

```
//Define the Car class
function Car() { }
Car.prototype.speed= 'Car Speed';
Car.prototype.setSpeed = function(speed) {
```

Examples
4. Java 8 Default Methods Tutorial
5. Compound Triggers in Oracle 11g - Tutorial with example
6. How to pass CLOB argument in EXECUTE IMMEDIATE
7. How to get Eclipse current workspace path
8. jQuery window height is not correct
9. 45 Useful Oracle Queries
10. JavaScript Singleton Design Pattern

```
    this.speed = speed;
    alert("Car speed changed");
}


//Define the Ferrari class
function Ferrari() { }
Ferrari.prototype = new Car();


// correct the constructor pointer because it points to Car
Ferrari.prototype.constructor = Ferrari;

// replace the setSpeed method
Ferrari.prototype.setSpeed = function(speed) {
    this.speed = speed;
    alert("Ferrari speed changed");
}

var car = new Ferrari();
car.setSpeed();
```

Encapsulation in JavaScript

In JavaScript, encapsulation is achieved by the inheritance. The child class inherit all the properties and methods of parent class and needs to override the method only that needs to be changed. This encapsulation by which every class inherits the methods of its parent and only needs to define things it wishes to change.

Update: A new tutorial cover these Javascript OOPs topics in details. Please refer: **JavaScript 101: Objects and Functions**

Related Articles

1. JavaScript 101: Objects and Functions
2. JavaScript Array Remove an Element
3. 21 JavaScript Tips and Tricks for JavaScript Developers
4. RPC in Javascript using JSON-RPC-Java
5. JavaScript Singleton Design Pattern
6. How to apply HTML User Interface Effects using jQuery.
7. Calling JavaScript function from String

Tags: JAVASCRIPT, OBJECT ORIENTED PROGRAMMING, OO CONCEPT, TUTORIAL

6 Comments

cancel bubble 17 July, 2009, 7:51

\"In JavaScript, encapsulation is achieved by the inheritance. The child class inherit all the properties and methods of parent class\"
In JavaScript, when you create a subclass from an existing class, only the *public* and *privileged* members are passed on (public/privileged members are created using the this keyword). You can also implement encapsulation with private members using closures.
In your example, this.speed is public and accessible, you can change it after instantiating the object. To truly protect speed, you\'d want to do something like this (I hope the spacing will be preserved):

```
var Car = function(mph) {
                //private attribute
                var speed;

                //privileged methods
                this.getSpeed = function() {
                return speed;
        }
```

```
        this.setSpeed = function(mph) {
                speed = mph;
        }

        this.setSpeed(mph);
} ;
Car.prototype = {
        //public, non-privileged methods go here
        whatever: function () {}
};
```

speed is private, you can only access it using the public getter and setter.
You can subclass Car and still access speed, but only through the privileged methods (getSpeed and setSpeed) because the privileged methods will be passed on (they are publicly accessible via the this keyword). No instance methods in the subclass will have *direct* access to speed, though – you have to go through the existing privileged methods (getSpeed and setSpeed).
\"JavaScript supports single class inheritance.\"
JavaScript also has prototypal inheritance which uses objects instead of defining a class structure and technically, you can have multiple inheritance via augmentation/mixins.

Reply

fernando trasviña 17 July, 2009, 22:33

Well i just want to point that JavaScript instantiation method does not mean copy public methods from the prototype, this process is an implementation of decoration pattern. Every instance creates an empty object that decorates the constructor.prototype object, so actually instance or inheritance (actually done by instance), its a decoration process, this is the process by which JavaScript allows to have dynamic classes, and when you augment the class all previous instances of that class get the method.
i wrote an article on the same topic of Object Oriented JavaScript in Mozilla
https://developer.mozilla.org/en/Introduction_to_Object-Oriented_JavaScript

Reply

Light up 10 May, 2010, 13:40

Thanks for the posting

Reply

Marimuthukumar 25 May, 2013, 18:12

Excellent article. Clearly understandable. I have got confused with class and function in oop concept in javascript. After I read this, I clearly understood. Thanks for posting.

Reply

Gosel 13 September, 2013, 11:49

really awesome, ThanQ .

Reply

dinker 19 December, 2013, 15:13

Hi Viral ,
Really awesome post. I was confused with classes, functions, objects, instance of class which is same as class object etc. But thanks a lot for such a nice post.
Many thanks,
Dinker.

Reply

Leave a Reply

Your email address will not be published. Required fields are marked *

Name *

Email *

Website

Comment

Note
To post source code in comment, use `[code language] [/code]` tag, for example:

- `[code java] Java source code here [/code]`
- `[code html] HTML here [/code]`

Post Comment

Categories

Home page Struts Spring AJAX PHP Java JavaEE JavaScript CSS Database Web 2.0 News Fun General Featured Play Framework Android FreeMarker Template HTML5

Site Pages

About viralpatel.net Join Us Search Advertise Posts Feed Comments Feed

Back to top