



## Contents

- [1 Introduction](#)
- [2 Install prototype.js](#)
- [3 Object Creation](#)
  - ◆ [3.1 Define classes in new syntax](#)
  - ◆ [3.2 Define classes in old syntax](#)
  - ◆ [3.3 Add new methods for an existing class temporarily](#)
  - ◆ [3.4 Use a mixin module when defining a new class](#)
  - ◆ [3.5 Add class methods](#)
- [4 Download Sample Application](#)
- [5 Related topics](#)
- [6 References](#)

## Introduction

As you may know, Prototype is a famous cross-web-browser javascript library. It supports almost all current popular web browsers, such as FireFox, Safari, IE, Opera, etc. With the latest version(Prototype 1.6.0), it also supports AppleWebKit, an open source web engine provided by Apple Inc.

The Nokia Web Browser is built upon S60WebKit, a port of the open source WebKit project to the S60 platform. Nokia WRT(Web-RunTime) is based on it.

In this topic, we will see how to define a class and its subclass using prototype.js library.

## Install prototype.js

Please go to [here](#) for how to install prototype javascript library.

## Object Creation

Classes and subclasses can be defined in new syntax or old syntax, we will define both for a comparison. We will also learn how to add an instance method and class method.

### Define classes in new syntax

To define a class, for example, Animal class, we can call Class.create method and pass properties into this method as follows:

```
// properties are directly passed to `create` method
var Animal = Class.create({
  initialize: function(name) {
```

## Use\_prototype\_javascript\_library:\_Object\_Creation\_in\_WRT\_application

```
    this.name = name;
  },
  eat: function(food) {
    return this.name + ' is eating ' + food;
  }
});
```

To derive a sub-class from the Animal class -- Cat class, we call the Class.create function again and use Animal class as the first parameter and a method collection for the new class as the second parameter. Note: Class.create can take in an arbitrary number of arguments.

```
// when subclassing, specify the class you want to inherit from
var Cat = Class.create(Animal, {
  // redefine the eat method
  eat: function($super, food) {
    return $super(food) + ' or sth.!!';
  }
});
```

In Cat class, the eat method overrides the same one in parent class.

The source code for the "new syntax" test case:

```
function testNewSyntax()
{
  var cat1 = new Cat('Mimi');
  alert(cat1.eat('fish'));
  // ->; "Mini is eating fish or sth!"
}
```

## Define classes in old syntax

Define a class -- Animal0 class. (We must use a different class name in a global namespace)

```
/** obsolete syntax */
var Animal0 = Class.create();
Animal0.prototype = {
  initialize: function(name) {
    this.name = name;
  },
  eat: function(food) {
    return this.name + ' is eating ' + food;
  }
};
```

Define a subclass -- Cat0 class. (Again, we use a different name.)

```
var Cat0 = Class.create();
// inherit from Person class:
Cat0.prototype = Object.extend(new Animal0(), {
  // redefine the eat method
  eat: function(food) {
    return this.name + ' is eating ' + food + ' or sth.!!';
  }
});
```

Define classes in new syntax

The source code for the "Old syntax" test case:

```
function testOldSyntax()
{
  var cat1 = new Cat0('Mimi');
  alert(cat1.eat('fish'));
  // -> "Mini is eating fish and sth.!"
}
```

### Add new methods for an existing class temporarily

Add a new method for Animal class, so an instance of a derived class can also call that method.

```
function testAddMethods()
{
  var cat1 = new Cat('Mimi');

  // every animal should be able to drink. But here we just let it "eat water"
  Animal.addMethods({
    drink: function() {
      return this.eat('water');
    }
  });

  alert(cat1.drink()); // -> "Mimi is eating water or sth.!"
}
```

### Use a mixin module when defining a new class

First, define a mixin module -- Tunable.

```
// define a module
var Tunable = {
  down: function(hp) {
    this.volume -= hp;
    if (this.volume < 0) this.mute();
  },
  mute: function() {
    this.isMute = true;
  }
};
```

Second, create a class based on the mixin.

```
var Tuner = Class.create(Tunable, {
  initialize: function() {
    this.volume = 100;
    this.isMute = false;
  }
});
```

And then test it. Here is the source code for this test case.

```
function testMixin()
{
```

Define classes in old syntax



## References

- Prototype Homepage[[1](#)]
- Sample WRT applications Download [[2](#)]