عنوان: **CoffeeScript #6**
نویسنده: وحید محمّدطاهری
تاریخ: ۱۳۹۴/۰۴/۰۲ ۲۰:۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, CoffeeScript

Classes

Inheritance & Super

شما می‌توانید به راحتی از کلاس‌های دیگری که نوشته‌اید، با استفاده از کلمه‌ی کلیدی extends، ارث بری کنید:

```
class Animal
  constructor: (@name) ->

  alive: ->
    true

class Parrot extends Animal
  constructor: ->
    super("Parrot")

  dead: ->
    not @alive()
```

در مثال بالا، Parrot (طوطی) از کلاس Animal ارث بری شده، که تمام خصوصیات آن را مانند ()alive، ارث برده است.
همانطوری که در مثال بالا مشاهده می‌کنید، در کلاس Parrot در تابع constructor، تابع super فراخوانی شده است. با استفاده از کلمه‌ی کلیدی super می‌توان تابع سازنده‌ی کلاس پدر را فراخوانی کرد. نتیجه‌ی کامپایل super در مثال بالا به این صورت می‌شود:

```
Parrot.__super__.constructor.call(this, "Parrot");
```

تابع super در CoffeeScript دقیقا مانند Ruby و Python عمل می‌کند.
در صورتیکه تابع constructor را در کلاس فرزند ننوشته باشید، به طور پیش فرض CoffeeScript سازنده کلاس پدر را فراخوانی می‌کند.

CoffeeScript با استفاده از prototypal inheritance، به صورت خودکار تمامی خصوصیات کلاس پدر، به فرزندان انتقال پیدا می‌کند. این ویژگی سبب داشتن کلاس‌های پویا می‌شود. برای درک بهتر این موضوع، فرض کنید که خصوصیتی را به کلاس پدر بعد از ارث بری کلاس فرزند اضافه می‌کنید. خصوصیت اضافه شده به تمامی فرزندان کلاس پدر به صورت خودکار اضافه می‌شود.

```
class Animal
  constructor: (@name) ->

class Parrot extends Animal

Animal::rip = true

parrot = new Parrot("Macaw")
alert("This parrot is no more") if parrot.rip
```

Mixins **Mixins** توسط CoffeeScript پشتیبانی نمی‌شود و برای همین نیاز است که این قابلیت را برای خودمان پیاده سازی کنیم، به مثال زیر توجه کنید.

```
extend = (obj, mixin) ->
  obj[name] = method for name, method of mixin
  obj
```

```
include = (klass, mixin) ->
  extend klass.prototype, mixin

# Usage
include Parrot,
  isDeceased: true

alert (new Parrot).isDeceased
```

نتیجه کامپایل آن می‌شود:

```
var extend, include;
extend = function(obj, mixin) {
  var method, name;
  for (name in mixin) {
    method = mixin[name];
    obj[name] = method;
  }
  return obj;
};
include = function(klass, mixin) {
  return extend(klass.prototype, mixin);
};
include(Parrot, {
  isDeceased: true
});
alert((new Parrot).isDeceased);
```

Mixins یک الگوی عالی برای به اشتراک گذاشتن خصوصیت‌های مشترک، در بین کلاس‌هایی است که امکان ارث بری در آنها وجود ندارد. مهمترین مزیت استفاده از Mixins این است که می‌توان چندین خصوصیت را به یک کلاس اضافه کرد؛ در حالیکه برای ارث بری فقط از یک کلاس می‌توان ارث بری داشت.

Extending classes

Mixins خیلی مرتب و خوب است اما خیلی شیء گرا نیست؛ در عوض امکان ادغام را در کلاس‌های CoffeeScript ایجاد می‌کند. برای اینکه اصول شیء گرایی را بخواهیم رعایت کنیم و ویژگی ادغام را نیز داشته باشیم، کلاسی با نام Module را پیاده سازی می‌کنیم و تمامی کلاس‌هایی را که می‌خواهیم ویژگی ادغام را داشته باشند، از آن ارث بری می‌کنیم.

```
moduleKeywords = ['extended', 'included']

class Module
  @extend: (obj) ->
    for key, value of obj when key not in moduleKeywords
      @[key] = value

    obj.extended?.apply(@)
    this

  @include: (obj) ->
    for key, value of obj when key not in moduleKeywords
      # Assign properties to the prototype
      @::[key] = value

    obj.included?.apply(@)
    this
```

برای استفاده از کلاس Module به مثال زیر توجه کنید:

```
classProperties =
  find: (id) ->
  create: (attrs) ->

instanceProperties =
  save: ->

class User extends Module
```

```
  @extend classProperties
  @include instanceProperties

# Usage:
user = User.find(1)

user = new User
user.save()
```

همانطور که مشاهده می‌کنید دو خصوصیت ثابت (static property)، را به کلاس User اضافه کردیم (find, create) و خصوصیت save.

همچنین برای خلاصه نویسی بیشتر می‌توان از این الگو استفاده کرد (ساده و زیبا).

```
ORM =
  find: (id) ->
  create: (attrs) ->
  extended: ->
    @include
      save: ->

class User extends Module
  @extend ORM
```