عنوان: **CoffeeScript #11**
نویسنده: وحید محمّدطاهری
تاریخ: ۱۳۹۴/۰۵/۰۳ ۱۰:۴۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, CoffeeScript

کامپایل خودکار CoffeeScript

همانطور که گفته شده CoffeeScript یک لایه میان شما و جاوااسکریپت است و هر زمان که فایل CoffeeScript تغییر کرد، باید به صورت دستی آن را کامپایل کرد. خوشبختانه CoffeeScript روش‌های دیگری را برای کامپایل کردن دارد که به وسیله آن می‌توان چرخه‌ی توسعه را بسیار ساده‌تر نمود.

در قسمت اول گفته شد، برای کامپایل فایل CoffeeScript با استفاده از *coffee* به صورت زیر عمل می‌کردیم:

```
coffee --compile --output lib src
```

همانطور که در مثال بالا مشاهده می‌کنید، تمامی فایل‌های *coffee*. در داخل پوشه *src* را کامپایل می‌کنید و فایل‌های جاوااسکریپت تولید شده را در پوشه lib ذخیره می‌کنید.
حال به کامپایل خودکار CoffeeScript توجه کنید.

Cake

Cake یک سیستم فوق العاده ساده برای کامپایل خودکار است که مانند Make و Rake عمل می‌کند. این کتابخانه همراه پکیج npm *coffee-script* نصب می‌شود و برای استفاده با فراخوانی *cake* اجرا می‌شود.

برای ایجاد فایل tasks در cake که Cakefile نامیده می‌شود، می‌توان از خود CoffeeScript استفاده کرد. برای اجرای cake با استفاده از دستور cake [options] [task] می‌توان عمل کرد. برای اطلاع از لیست امکانات cake کافی است دستور cake را به تنهایی اجرا کنید.

وظایف را می‌توان با استفاده از تابع task، با ارسال نام و توضیحات (اختیاری) و تابع callback، تعریف کرد. به مثال زیر توجه کنید:

```
fs = require 'fs'

{print} = require 'sys'
{spawn} = require 'child_process'

build = (callback) ->
  coffee = spawn 'coffee', ['-c', '-o', 'lib', 'src']
  coffee.stderr.on 'data', (data) ->
    process.stderr.write data.toString()
  coffee.stdout.on 'data', (data) ->
    print data.toString()
  coffee.on 'exit', (code) ->
    callback?() if code is 0

task 'build', 'Build lib/ from src/', ->
  build()
```

همانطور که در مثال بالا مشاهده می‌کنید، تابع task را با نام **build** تعریف کردیم و با استفاده از دستور cake build می‌توان آن را اجرا نمود. پس از اجرا همانند مثال قبل تمامی فایل‌های CoffeeScript در پوشه‌ی src به فایل‌های جاوااسکریپت در پوشه lib تبدیل می‌شوند.
همان طور که مشاهده می‌کنید پس از تغییر در فایل CoffeeScript باید به صورت دستی cake build را فراخوانی کنیم که این دور از حالت ایده آل است.
خوشبختانه دستور coffee پارامتر دیگری به نام watch-- دارد که به وسیله آن می‌توان تمامی تغییرات یک پوشه را زیر نظر گرفت

و در صورت نیاز دوباره کامپایل انجام شود. به مثال زیر توجه کنید:

```
task 'watch', 'Watch src/ for changes', ->
    coffee = spawn 'coffee', ['-w', '-c', '-o', 'lib', 'src']
    coffee.stderr.on 'data', (data) ->
      process.stderr.write data.toString()
    coffee.stdout.on 'data', (data) ->
      print data.toString()
```

در صورتی که task ایی وابسته به task دیگری باشد، می‌توانید برای اجرای taskهای دیگر از دستور *invoke(name)* استفاده کنید. برای مثال یک task را به فایل Cakefile اضافه می‌کنیم که در آن ابتدا فایل index.html را باز کرده و سپس شروع به زیر نظر گرفتن پوشه src می‌کنیم.

```
task 'open', 'Open index.html', ->
  # First open, then watch
  spawn 'open', 'index.html'
  invoke 'watch'
```

همچنین می‌توانید با استفاده از تابع ()option، options را برای taskها تعریف کنید.

```
option '-o', '--output [DIR]', 'output dir'

task 'build', 'Build lib/ from src/', ->
  # Now we have access to a `options` object
  coffee = spawn 'coffee', ['-c', '-o', options.output or 'lib', 'src']
  coffee.stderr.on 'data', (data) ->
    process.stderr.write data.toString()
  coffee.stdout.on 'data', (data) ->
    print data.toString()
```

Cake یک روش عالی برای انجام وظایف معمول به صورت خودکار است، مانند کامپایل فایل‌های CoffeeScript است. همچنین برای آشنایی بیشتر می‌توانید به سورس cake نگاهی کنید.