عنوان: **CoffeeScript #12**
نویسنده: وحید محمّدطاهری
تاریخ: ۱۳۹۴/۰۵/۰۸ ۱۲:۳۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, CoffeeScript

بخش‌های بد

جاوااسکریپت یک زبان پیچیده است که شما برای کار با آن، نیاز است قسمت‌هایی را که **باید** از آن‌ها دوری کنید و قسمت‌های مهمی را که **باید** استفاده کنید، بشناسید. همانطور که Sun Tzu گفته "دشمن خود را بشناس"، ما نیز در این قسمت می‌خواهیم برای شناخت بیشتر قسمت‌های تاریک و روشن جاوااسکریپت به آن بپردازیم.

همانطور که در قسمت‌های قبل گفته شد، CoffeeScript تنها به یک syntax محدود نمی‌شود و توانایی برطرف کردن برخی از مشکلات جاوااسکریپت را نیز دارد. با این حال، با توجه به این واقعیت که کدهای CoffeeScript به صورت مستقیم به جاوااسکریپت تبدیل می‌شوند و نمی‌توانند تمامی مشکلاتی را که در جاوااسکریپت وجود دارند، حل کنند، پس برخی از مسائل وجود دارند که شما باید از آنها آگاهی داشته باشید.

اول از قسمت‌هایی که توسط CoffeeScript حل شده‌اند شروع می‌کنیم.

A JavaScript Subset

with یک دستور بسیار زمانبر است و **مضر** شناخته شده است و نباید از آن استفاده کنید. with با ایجاد یک ساختار خلاصه نویسی، برای جستجو بر روی خصوصیات اشیاء در نظر گرفته شده بود. برای نمونه به جای نوشتن:

```
dataObj.users.vahid.email = "info@vmt.ir";
```

می‌توانید به این صورت این کار را انجام دهید:

```
with(dataObj.users.vahid) {
  email = "info@vmt.ir";
}
```

مفسر جاوااسکریپت دقیقا نمی‌داند که شما می‌خواهید چه کاری را با with انجام دهید، و به شیء مشخص شده فشار می‌آورد تا اول اسم همه مراجعه شده‌ها را جستجو کند. این عمل واقعا به عملکرد و کارآیی لطمه می‌زند. یعنی مترجم، تمام انواع بهینه سازی‌های JIT را خاموش می‌کند. همچنین پیشنهادهایی مبنی بر حذف کامل آن از نسخه‌های بعدی جاوااسکریپت نیز مطرح شده است.
همه چیز برای عدم استفاده از with در نظر گرفته شده است. CoffeeScript یک قدم جلوتر از همه برداشته و with را از syntax خود حذف کرده است. به عبارت دیگر در صورتیکه شما از آن استفاده کنید، کامپایلر CoffeeScript خطا صادر می‌کند.

Global variables

به طور پیش فرض تمامی برنامه‌های جاوااسکریپت در دامنه global اجرا می‌شوند و تمامی متغیرهایی که ساخته می‌شوند به طور پیش فرض در ناحیه‌ی global قرار می‌گیرند. اگر شما بخواهید متغیری را در ناحیه‌ی local ایجاد کنید، باید از کلمه کلیدی *var* استفاده کنید.

```
usersCount = 1;         // Global
var groupsCount = 2;    // Global

(function(){
  pagesCount = 3;       // Global
  var postsCount = 4;   // Local
})()
```

اکثر اوقات شما می‌خواهید متغیر local ایی را ایجاد کنید و نه global. توسعه دهندگان باید همیشه به یاد داشته باشند که قبل از مقداردهی اولیه‌ی هر متغیری، کلمه‌ی کلیدی var را قرار دهند یا با انواع و اقسام مشکلات، هنگامی که متغیرها به طور تصادفی با یکدیگر برخورد و یا بازنویسی بر روی یکدیگر انجام می‌دهند، روبرو شوند.
خوشبختانه CoffeeScript به کمک شما می‌آید و به طور کامل انتساب متغیرهای global را به طور ضمنی از بین می‌برد. به عبارت دیگر کلمه کلیدی var در CoffeeScript رزرو شده است و در صورت استفاده خطا صادر می‌شود.
به صورت پیش فرض به طور ضمنی متغیرها local ایجاد می‌شوند و خیلی سخت می‌شود متغیر global ایی را بدون انتساب آن به عنوان خصوصیتی از شیء window ایجاد کرد.

```
outerScope = true
do ->
  innerScope = true
```

نتیجه‌ی کامپایل آن می‌شود:

```
var outerScope;
outerScope = true;
(function() {
  var innerScope;
  return innerScope = true;
})();
```

همانطور که مشاهده می‌کنید CoffeeScript مقداردهی اولیه متغیر را (با استفاده از var) به صورت خودکار در context ایی که برای اولین بار استفاده شده است انجام می‌دهد. باید مواظب باشید تا از نام متغیر خارجی مجددا استفاده نکنید که این اتفاق ممکن است در کلاس یا تابع با عمق زیاد ایجاد شود. برای مثال، در اینجا به صورت تصادفی متغیر package در یک تابع کلاس بازنویسی شده است:

```
package = require('./package')

class Test
  build: ->
    # Overwrites outer variable!
    package = @testPackage.compile()

  testPackage: ->
    package.create()
```

برای ایجاد متغیرهای global باید از انتساب آنها به عنوان خصوصیتی از شیء window استفاده کرد.

```
class window.Asset
    constructor: ->
```

با تضمین متغیرهای global به صورت صریح و روشن به جای به طور ضمنی بودن آنها، CoffeeScript یکی از منابع اصلی ایجاد مشکلات در جاوااسکریپت را حذف کرده‌است.

Semicolons

جاوااسکریپت اجباری برای نوشتن " **;** " ندارد، بنابراین ممکن است یک سری از دستورات از قلم بیافتند. با این حال در پشت صحنه‌ی کامپایلر جاوااسکریپت به " **;** " احتیاج دارد. به طوری که parser جاوااسکریپت به صورت خودکار هر زمانی که نتواند ارزیابی از دستورات داشته باشد، یک بار دیگر با " **;** " این کار را انجام می‌دهد و درصورت موفقیت، پیام خطایی مبنی بر نبود " **;** " را صادر می‌کند.
متاسفانه این یک ایده **بد** است. چرا که ممکن است تغییر رفتاری در کد نوشته شده به وجود آید. به مثال زیر توجه کنید. به نظر کد نوشته شده صحیح است؛ درسته؟

```
function() {}
```

```
(window.options || {}).property
```

اشتباه است، حداقل با توجه به parser، یک خطای syntax صادر می‌شود. در مورد دوم نیز parser، " ; " اضافه نمی‌کند و کد نوشته شده به کد یک خطی تبدیل می‌شوند.

```
function() {}(window.options || {}).property
```

حالا شما می‌توانید این موضوع را ببینید که چرا parser خطا داده‌است. وقتی شما در حال نوشتن کد جاوااسکریپتی هستید، باید بعد از هر دستور از " ; " استفاده کنید. خوشبختانه در تمام زمانیکه درحال نوشتن کد CoffeeScript هستید، نیازی به نوشتن " ; " ندارید. در زمانیکه کد CoffeeScript نوشته شده کامپایل می‌شود، به صورت خودکار " ; " را در جای مناسبی قرار می‌دهد.