عنوان: **CoffeeScript #1**
نویسنده: وحید محمّدطاهری
تاریخ: ۱۳۹۴/۰۳/۲۷ ۹:۲۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, CoffeeScript

مقدمه CoffeeScript یک زبان برنامه نویسی برای تولید کدهای جاوااسکریپت است که Syntax آن الهام گرفته از Ruby و Python است و بسیاری از ویژگی‌هایش، از این دو زبان پیاده سازی شده است.
سوالی که ممکن است برای هر کسی پیش بیاید این است که چرا باید از CoffeeScript استفاده کرد و یا چرا نوشتن CoffeeScript بهتر از نوشتن مستقیم جاوااسکریپت است؟

از جمله دلایلی که می‌شود عنوان کرد:
حجم کد کمتری نوشته می‌شود (تجربه شخصی من: تقریبا کدنویسی شما به یک سوم تا نصف تبدیل می‌شود)، بسیار مختصر است و پیاده سازی aliases prototype و classes به سادگی و با حداقل کدنویسی انجام می‌گیرد.

CoffeeScript زیرمجموعه‌ای از جاوااسکریپت نیست، اگرچه از کتابخانه‌های خارجی جاوااسکریپت می‌توان در کدهای CoffeeScript استفاده کرد، اما برای اینکار باید کدهای مورد نیاز را به CoffeeScript تبدیل کرد تا از خطای زمان کامپایل جلوگیری شود.

پیش نیاز نوشتن کد به زبان CoffeeScript، شناخت جاوااسکریپت است تا بتوان خطاهای زمان اجرا را اصلاح کرد.

CoffeeScript محدودیتی در مرورگر ندارد و می‌توان در برنامه‌های جاوااسکریپتی تحت سرور مانند Node.js با کیفیت بالا نیز از آن استفاده کرد.
زمانی را که برای یادگیری CoffeeScript صرف می‌کنید در زمان نوشتن پروژه، نتیجه‌ی آن‌را متوجه خواهید شد.

راه اندازی اولیه
یکی از ساده‌ترین راه‌های نوشتن CoffeeScript استفاده از نسخه‌ی مرورگر این زبان است و برای اینکار باید وارد سایت CoffeeScript.Org شده و بر روی تب *Try CoffeeScript* کلیک کنید. این سایت از نسخه‌ی مرورگر CoffeeScript Compiler استفاده می‌کند و هر کدی CoffeeScript ایی که در پنل سمت چپ سایت بنویسید، تبدیل به جاوااسکریپت می‌شود و در پنل راست سایت، نمایش داده می‌شود.

همچنین می‌توانید با استفاده از پروژه‌ی js2coffee کدهای جاوااسکریپت را به کدهای CoffeeScript تبدیل کنید.

در صورتیکه بخواهید از نسخه‌ی درون مرورگری CoffeeScript Compiler استفاده کنید، باید یک تگ اسکریپت لینک به این اسکریپت و با اضافه کردن تگ اسکریپت با type coffeescript این کار را انجام دهید. برای نمونه:

```
<script src="http://jashkenas.github.com/coffee-script/extras/coffee-script.js" type="text/javascript"
charset="utf-8"></script>
<script type="text/coffeescript">
  # Some CoffeeScript
</script>
```

بدیهی است که استفاده از چنین روشی برای تحویل پروژه به مشتری صحیح نیست چرا که به خاطر تفسیر کدهای CoffeeScript در زمان اجرا، سرعت اجرایی پایین خواهد بود. به جای این روش CoffeeScript پیشنهاد می‌کند که از compiler Node.js و تبدیل آن به فایل‌های coffeescript pre-process استفاده کنید.
برای نصب باید آخرین نسخه‌ی Node.js و (Node Package Manager) npm را نصب کرده باشید. برای نصب CoffeeScript با استفاده از npm از دستور زیر استفاده کنید.

```
npm install -g coffee-script
```

پس از نصب می‌توانید با استفاده از دستور coffee فایل‌های CoffeeScript خود را (بدون پارامتر) اجرا کنید و در صورتیکه بخواهید خروجی جاوااسکریپت داشته باشید، از پارامتر *compile*-- استفاده کنید.

```
coffee --compile my-script.coffee
```

در صورتیکه پارامتر *output*-- تعریف نشود CoffeeScript فایل خروجی را هم نام با فایل اصلی قرار می‌دهد که در مثال بالا فایل خروجی می‌شود *my-script.js* . در صورتیکه فایلی از قبل موجود باشد، بازنویسی انجام می‌شود.

عنوان: **CoffeeScript #2**
نویسنده: وحید محمّدطاهری
تاریخ: ۱۳۹۴/۰۳/۲۷ ۱۹:۵۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, CoffeeScript

Syntax

برای کار با CoffeeScript، ابتدا باید با ساختار Syntax آن آشنا شد. CoffeeScript در بسیاری از موارد با جاوااسکریپت یکسان است در حالیکه در قسمت قبل گفته شد که CoffeeScript زیر مجموعه‌ای جاوااسکریپت نیست؛ بنابراین برخی از کلمات کلیدی مانند *function* و *var* در آن مجاز نیست و سبب بروز خطا در زمان کامپایل می‌شوند. وقتی شما شروع به نوشتن فایل CoffeeScript می‌کنید، باید تمام کدهایی را که می‌نویسید، با Syntax کامل CoffeeScript بنویسید و نمی‌توانید قسمتی را با جاوااسکریپت و قسمتی را با CoffeeScript بنویسید.

برای نوشتن توضیحات در فایل CoffeeScript باید از علامت # استفاده کنید که این قسمت را از زبان Ruby گرفته است.

```
# A comment
```

در صورتیکه نیاز به نوشتن توضیحات را در چندین خط داشته باشید نیز این امکان دیده شده است:

```
###
  A multiline comment
###
```

نکته: تفاوتی که در توضیح یک خطی و چند خطی وجود دارد این است که توضیحات چند خطی پس از کامپایل، در فایل جاوااسکریپت خروجی نوشته می‌شوند، ولی توضیحات یک خطی در فایل خروجی تولید می‌شود.

در زبان CoffeeScript فاصله (space) بسیار مهم است؛ چرا که زبان Python براساس میزان تو رفتگی کدها، بدنه‌ی شرط‌ها و حلقه‌ها را تشخیص می‌دهد و CoffeeScript نیز از این ویژگی استفاده می‌کند. هرگاه بخواهید از {} استفاده کنید فقط کافی است از کلید Tab استفاده کنید تا پس از کامپایل به صورت {} تبدیل شود.

Variables & Scope

CoffeeScript یکی از باگهایی را که در نوشتن جاوااسکریپت وجود دارد (متغیرهای سراسری) حل کرده است. در جاوااسکریپت درصورتیکه هنگام تعریف متغیری از کلمه‌ی کلیدی *var* در پشت اسم متغیر استفاده نشود، به صورت سراسری تعریف می‌شود. CoffeeScript به سادگی متغیرهای سراسری را حذف می‌کند. در پشت صحنه‌ی این حذف، اسکریپت نوشته شده را درون یک تابع بدون نام قرار می‌دهد و با این کار تمامی متغیرها در ناحیه‌ی محلی قرار می‌گیرند و سپس قبل از نام هر متغیری، کلمه‌ی کلیدی *var* را قرار می‌دهد. برای مثال:

```
myVariable = "vahid"
```

که نتیجه کامپایل آن می‌شود:

```
var myVariable;
myVariable = "vahid";
```

همان طور که مشاهده می‌کنید، متغیر تعریف شده به صورت محلی تعریف شده و با این روش تعریف متغیر سراسری را به صورت اشتباهی، غیرممکن می‌کند. این روش استفاده شده در CoffeeScript جلوی بسیاری از اشتباهات معمول توسعه دهندگان وب را می‌گیرد.
با این حال گاهی اوقات نیاز است که متغیر سراسری تعریف کنید. برای اینکار باید از شیء سراسری موجود در مرورگر (window)

یا از روش زیر استفاده کنید:

```
exports = this
exports.MyVariable = "vahid"
```

Functions

CoffeeScript برای راحتی در نوشتن توابع، کلمه کلیدی function را حذف کرده و به جای آن از <- استفاده می‌کند. توابع در CoffeeScript می‌توانند در یک خط یا به صورت تورفته در چندین خط نوشته شده باشند. آخرین عبارتی که در یک تابع نوشته می‌شود به صورت ضمنی بازگشت داده می‌شود. در صورتیکه نیاز به بازگرداندن مقداری در تابع ندارید، از کلمه‌ی *return* به تنهایی استفاده کنید.

```
func = -> "vahid"
```

نتیجه‌ی کامپایل آن می‌شود:

```
var func;
func = function() {
  return "vahid";
};
```

همان طور که در بالا گفته شده، در صورتیکه بخواهید تابعی با چندین خط دستور داشته باشید، باید ساختار تو رفتگی را حفظ کرد. برای مثال:

```
func = ->
  # An extra line
  "vahid"
```

نتیجه کامپایل کد بالا نیز همانند کد قبلی می‌باشد.

Function arguments

برای تعریف آرگومان در توابع باید قبل از <- از () استفاده کرد و آرگومان هایی را که نیاز است، در داخل آن تعریف کرد. برای مثال:

```
func = (a, b) -> a * b
```

نتیجه‌ی کامپایل آن می‌شود:

```
var func;
func = function(a, b) {
  return a * b;
};
```

CoffeeScript از مقدار پیش فرض برای آرگومان‌های توابع نیز پشتیبانی می‌کند:

```
func = (a = 1, b = 2) -> a * b
```

همچنین در صورتیکه تعداد آرگومان‌های یک تابع برای شما مشخص نبود، می‌توانید از " ... " استفاده کنید. مثلا وقتی می‌خواهید جمع n عدد را بدست آورید که n عدد به صورت آرگومان به تابع ارسال می‌شوند:

```
sum = (nums...) ->
  result = 0
```

```
  nums.forEach (n) -> result += n
  result
```

در مثال فوق آرگومان nums آرایه‌ای از تمام آرگومان‌های ارسال شده به تابع است و نتیجه‌ی کامپایل آن می‌شود:

```
var sum,
  slice = [].slice;

sum = function() {
  var nums, result;
  nums = 1 <= arguments.length ? slice.call(arguments, 0) : [];
  result = 0;
  nums.forEach(function(n) {
    return result += n;
  });
  return result;
};
```

فراخوانی توابع

برای فراخوانی توابع می‌توانید به مانند جاوااسکریپت از با پرانتز () یا ()apply و یا ()call صدا زده شوند. اگرچه مانند Ruby، کامپایلر CoffeeScript می‌توانند به صورت اتوماتیک توابعی با حداقل یک آرگومان را فراخوانی کند.

```
a = "Vahid!"

alert a
# برابر است با
alert(a)

alert inspect a
# برابر است با
alert(inspect(a))
```

اگرچه استفاده از پرانتز اختیاری است اما توصیه می‌شود در مواقعی که آرگومان‌های ارسالی بیش از یک مورد باشد توصیه می‌شود از پرانتز استفاده کنید.
در صورتی که تابعی بدون آرگومان باشد، برای فراخوانی آن بدون نوشتن پرانتز بعد از نام تابع، CoffeeScript نمی‌تواند تشخیص دهند که این یک تابع است و مانند یک متغیر با آن برخورد می‌کند. دراین رابطه، رفتار CoffeeScript بسیار شبیه به Python می‌باشد.

عنوان: **CoffeeScript #3**
نویسنده: وحید محمّدطاهری
تاریخ: ۱۳۹۴/۰۳/۲۸ ۲۱:۴۰
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, CoffeeScript

Syntax

Object & Array

برای تعریف Object در CoffeeScript می‌توان دقیقا مانند جاوااسکریپت عمل کرد؛ با یک جفت براکت و ساختار کلید / مقدار. البته همانند تابع، نوشتن براکت اختیاری است. در واقع، شما می‌توانید از تورفتگی و هر کلید/مقدار، در خط جدید به جای کاما استفاده کنید:

```
object1 = {one: 1, two: 2}

# Without braces
object2 = one: 1, two: 2

# Using new lines instead of commas
object3 =
  one: 1
  two: 2

User.create(name: "Vahid Mohammad Taheri")
```

به همین ترتیب، برای تعریف آرایه‌ها می‌توانید از کاما به عنوان جدا کننده و یا هر مقدار آرایه را در یک خط جدید وارد کنید؛ هر چند براکت [] هنوز هم مورد نیاز است.

```
array1 = [1, 2, 3]

array2 = [
  1
  2
  3
]

array3 = [1,2,3,]
```

Flow control

طبق قاعده‌ای که برای نوشتن پرانتز در قبل گفته شد (پرانتز اختیاری است)، در دستورات if و else نیز چنین است:

```
if true == true
  "We're ok"

if true != true then "Vahid"

# برابر است با:
#  (1 > 0) ? "Yes" : "No!"
if 1 > 0 then "Yes" else "No!"
```

همانطوری که در مثال بالا مشاهده می‌کنید، در صورتی که از if در یک خط استفاده شود باید پس از شرط، کلمه کلیدی then را بنویسید.
CoffeeScript از اپراتورهای شرطی (:?) پشتیبانی **نمی کند** و به جای آن از if / else استفاده کنید.
CoffeeScript نیز همانند Ruby امکان نوشتن بدنه شرط را به صورت پسوندی ایجاد کرده است.

```
alert "It's cold!" if 1 < 5
```

به جای استفاده از علامت ! برای منفی سازی شرط، می‌توانید از کلمه‌ی کلیدی *not* استفاده کنید که سبب خوانایی بیشتر کد نوشته شده می‌شود:

```
if not true then "Vahid"
```

CoffeeScript امکان نوشتن خلاصه‌تر if not را نیز ایجاد کرده است؛ برای این کار از کلمه‌ی کلیدی *unless* استفاده کنید. معادل مثال بالا:

```
unless true
  "Vahid"
```

همانند not که برای خوانایی بالاتر کد به کار می‌رود، CoffeeScript کلمه کلیدی is را مطرح کرده‌است که پس از کامپایل به === ترجمه می‌شود.

```
if true is 1
  "OK!"
```

برای نوشتن ==! نیز می‌توان از is not استفاده کرد، که شکل خلاصه‌تر آن *isnt* است.

```
if true isnt true
  alert "OK!"
```

همانطوری که در بالا گفته شد، CoffeeScript عملگر == را به === و =! به ==! تبدیل می‌کند. دلیلی که CoffeeScript این عمل را انجام می‌دهد این است که جاوااسکریپت عمل مقایسه را بر روی نوع و سپس مقدار آن انجام می‌دهد و سبب پیشگیری از باگ در کد نوشته شده می‌شود.

الحاق رشته ها CoffeeScript امکان الحاق رشته‌ها را با استفاده از روش الحاق رشته‌ها در Ruby فراهم کرده است. برای انجام این عمل از #{} در داخل " " استفاده کنید که در داخل براکت می‌توانید از دستورات مختلف استفاده کنید. برای مثال:

```
favorite_color = "Blue. No, yel..."
question = "Sam: What... is your favorite color?
            Ben: #{favorite_color}
            Sam: Wrong!
            "
```

نتیجه‌ی کامپایل کد بالا می‌شود:

```
var favorite_color, question;
favorite_color = "Blue. No, yel...";
question = "Sam: What... is your favorite color?           Ben: " + favorite_color + "           Sam:
Wrong!";
```