عنوان: **CoffeeScript #4**
نویسنده: وحید محمّدطاهری
تاریخ: ۱۳۹۴/۰۳/۳۱ ۱۴:۳۰
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, CoffeeScript

Syntax

Loops

```
for name in ["Vahid", "Hamid", "Saeid"]
  alert "Hi #{name}"
```

نتیجه‌ی کامپایل کد بالا می‌شود:

```
var i, len, name, ref;

ref = ["Vahid", "Hamid", "Saeid"];
for (i = 0, len = ref.length; i < len; i++) {
  name = ref[i];
  alert("Hi " + name);
}
```

درصورتیکه نیاز به شمارنده‌ی حلقه داشته باشید، کافیست یک آرگومان اضافه را ارسال کنید. برای نمونه:

```
for name, i in ["Vahid", "Hamid", "Saeid"]
  alert "#{i} - Hi #{name}"
```

همچنین می‌توانید حلقه را به صورت یک خطی نیز بنویسید:

```
alert name for name in ["Vahid", "Hamid", "Saeid"]
```

همچنین مانند Python نیز می‌توانید از فیلتر کردن در حلقه، استفاده کنید.

```
names = ["Vahid", "Hamid", "Saeid"]
alert name for name in names when name[0] is "V"
```

و نتیجه کامپایل کد بالا می‌شود:

```
var i, len, name, names;

names = ["Vahid", "Hamid", "Saeid"];

for (i = 0, len = names.length; i < len; i++) {
  name = names[i];
  if (name[0] === "V") {
    alert(name);
  }
}
```

شما می‌توانید حلقه را برای یک object نیز استفاده کنید. به جای استفاده از کلمه‌ی کلیدی *in* ، از کلمه کلیدی *of* استفاده کنید.

```
names = "Vahid": "Mohammad Taheri", "Ali": "Ahmadi"
alert("#{first} #{last}") for first, last of names
```

پس از کامپایل نتیجه می‌شود:

```
var first, last, names;

names = {
  "Vahid": "Mohammad Taheri",
  "Ali": "Ahmadi"
};

for (first in names) {
  last = names[first];
  alert(first + " " + last);
}
```

حلقه while در CoffeeScript به مانند جاوااسکریپت عمل می‌کند؛ ولی مزیتی نیز به آن اضافه شده است که آرایه‌ای از نتایج را بر می‌گرداند. به عنوان مثال مانند تابع Array.prototype.map() .

```
num = 6
minstrel = while num -= 1
  num + " Hi"
```

نتیجه‌ی کامپایل آن می‌شود:

```
var minstrel, num;
num = 6;
minstrel = (function() {
  var _results;
  _results = [];
  while (num -= 1) {
    _results.push(num + " Hi");
  }
  return _results;
})();
```

CoffeeScript **Arrays** با الهام گرفتن از Ruby، به وسیله تعیین محدوده، آرایه را ایجاد می‌کند. محدوده آرایه به وسیله دو عدد تعیین می‌شوند که با .. یا ... از هم جدا می‌شوند.

```
range = [1..5]
```

نتیجه‌ی کامپایل می‌شود:

```
var range;
range = [1, 2, 3, 4, 5];
```

در صورتی که محدوده‌ی آرایه **بلافاصله** بعد از یک متغیر بیاید CoffeeScript، کد نوشته شده را به تابع slice() تبدیل می‌کند.

```
firstTwo = ["one", "two", "three"][0..1]
```

نتیجه کامپایل می‌شود:

```
var firstTwo;
firstTwo = ["one", "two", "three"].slice(0, 2);
```

در مثال بالا محدوده تعیین شده سبب می‌شود که یک آرایه جدید با دو عنصر "one" و "two" ایجاد شود. همچنین می‌توانید برای جایگزینی مقادیر جدید، در یک آرایه از قبل تعریف شده نیز از روش زیر استفاده کنید.

```
numbers = [0..9]
numbers[3..5] = [-3, -4, -5]
```

نکته: در صورتیکه متغیری قبل از تعریف محدوده آرایه قرار گیرد، اگر رشته باشد، نتیجه‌ی خروجی، آرایه‌ای از کاراکترهای آن می‌شود.

```
my = "my string"[0..2]
```

چک کردن وجود یک مقدار در آرایه، یکی از مشکلاتی است که در جاوااسکریپت وجود دارد (عدم پشتیبانی از ()indexOf در IE کمتر از 9). CoffeeScript با استفاده از کلمه‌ی کلیدی *in* این مشکل را برطرف کرده است.

```
words = ["Vahid", "Hamid", "Saeid", "Ali"]
alert "Stop" if "Hamid" in words
```

نکات مهم

در صورت تعریف محدوده آرایه به صورت numbers[3..] (که آرایه numbers از قبل تعریف شده باشد)، خروجی، آرایه‌ای از مقادیر موجود در numbers را از خانه شماره 4 تا انتهای آن برمی گرداند.
در صورت تعریف محدوده آرایه به صورت numbers[-3..] (که آرایه numbers از قبل تعریف شده باشد)، خروجی، آرایه‌ای از مقادیر موجود در numbers را از خانه انتهایی به میزان 3 خانه به سمت ابتدای آرایه برمیگرداند.
در صورت عدم تعریف محدوده آرایه و فقط استفاده از [..] یا [...] (یک شکل عمل می‌کنند)، کل مقادیر آرایه اصلی (که از قبل تعریف شده باشد)، برگردانده می‌شود.

تفاوت .. و ... در حالتی که دو عدد برای محدوده تعریف شود، در این است که ... آرایه به صورت عدد انتهایی - 1 تعریف می‌شود. مثلا [0...3] یعنی خانه‌های آرایه از 0 تا 2 را به عنوان خروجی برگردان.

CoffeeScript **Aliases** شامل یک سری نام‌های مستعار است که برای خلاصه نویسی بیشتر بسیار مفید هستند. یکی از آن نام ها، @ است که به جای نوشتن *this* به کار می‌رود.

```
@name = "Vahid"
```

نتیجه کامپایل آن می‌شود:

```
this.name = "Vahid";
```

یکی دیگر از این نام ها، :: می‌باشد که به جای نوشتن prototype به کار می‌رود.

```
User::first = -> @records[0]
```

نتیجه کامپایل آن می‌شود:

```
User.prototype.first = function() {
  return this.records[0];
};
```

یکی از عمومی‌ترین شرط هایی که در جاوااسکریپت استفاده می‌شود، شرط چک کردن not null است. CoffeeScript این کار را با استفاده از ? انجام می‌دهد و در صورتی که متغیر برابر با null یا undefined نباشد، مقدار true را برمی گرداند. این ویژگی همانند ?nil در Ruby است.

```
alert "OK" if name?
```

نتیجه‌ی کامپایل آن می‌شود:

```
if (typeof name !== "undefined" && name !== null) {
  alert("OK");
}
```

از ? به جای || نیز می‌توانید استفاده کنید.

```
name = myName ? "-"
```

نتیجه‌ی کامپایل آن می‌شود:

```
var name;

name = typeof myName !== "undefined" && myName !== null ? myName : "-";
```

در صورتیکه بخواهید به یک property از یک شیء دسترسی داشته باشید و بخواهید null نبودن آن را قبل از دسترسی به آن چک کنید، می‌توانید از ? استفاده کنید.

```
user.getAddress()?.getStreetName()
```

نتیجه‌ی کامپایل آن می‌شود:

```
var ref;

if ((ref = user.getAddress()) != null) {
  ref.getStreetName();
}
```

همچنین در صورتیکه بخواهید چک کنید یک property در واقع یک تابع است یا نه (مثلا برای مواقعی که می‌خواهید callback بسازید) و سپس آن را فراخوانی کنید، نیز از ? می‌توانید استفاده کنید.

```
user.getAddress().getStreetName?()
```

و نتیجه‌ی کامپایل آن می‌شود:

```
var base;

if (typeof (base = user.getAddress()).getStreetName === "function") {
  base.getStreetName();
}
```