عنوان: **CoffeeScript #7**
نویسنده: وحید محمّدطاهری
تاریخ: ۱۳۹۴/۰۴/۰۴ ۱۹:۴۰
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, CoffeeScript

اصطلاحات عمومی CoffeeScript

هر زبانی دارای مجموعه‌ای از اصطلاحات و روش هاست. CoffeeScript نیز از این قاعده مستثنی نیست. در این قسمت می‌خواهیم مقایسه‌ای بین جاوااسکریپت و CoffeeScript انجام دهیم تا به وسیله‌ی این مقایسه، مفهوم عملی این زبان را درک کنید.

Each

در جاوااسکریپت وقتی می‌خواهیم بر روی آرایه‌ای با بیش از یک خانه، کاری را چندین بار انجام دهیم، می‌توانیم از تابع forEach() یا از همان قالب حلقه‌ی for در زبان C استفاده کنیم:

```
for (var i=0; i < array.length; i++)
  myFunction(array[i]);

array.forEach(function(item, i){
  myFunction(item)
});
```

اگرچه تابع forEach() مختصر و خواناتر است ولی یک مشکل دارد؛ به دلیل فراخوانی تابع callback در هر بار اجرای حلقه، بسیار کندتر از حلقه for اجرا می‌شود.
حال به نحوه‌ی کارکرد CoffeeScript دقت کنید.

```
myFunction(item) for item in array
```

که پس از کامپایل می‌شود:

```
var i, item, len;

for (i = 0, len = array.length; i < len; i++) {
  item = array[i];
  myFunction(item);
}
```

همانطوری که مشاهده می‌کنید، از نظر syntax بسیار ساده و با خوانایی بالا است و مطمئن هستم شما هم با من موافق هستید و نکته‌ی مهمی که وجود دارد، کامپایل حلقه‌ی با ظاهر forEach به حلقه‌ی for، توسط CoffeeScript و حفظ سرعت اجرای آن است.

Map

همانند تابع forEach که در استاندارد ES5 قرار داشت، تابع دیگری به نام map() وجود دارد که از نظر syntax بسیار خلاصه‌تر از حلقه‌ی for می‌باشد. ولی متاسفانه همانند تابع forEach، این تابع نیز به دلیل فراخوانی تابع، بسیار کندتر از for اجرا می‌شود.

```
var result = []
for (var i=0; i < array.length; i++)
  result.push(array[i].name)

var result = array.map(function(item, i){
  return item.name;
});
```

همانطور که مشاهده می‌کنید در اینجا طریقه‌ی استفاده از تابع map و پیاده سازی آن بدون استفاده از تابع map نشان داده شده

است. حال به مثال زیر توجه کنید:

```
result = (item.name for item in array)
```

با استفاده از ساختار حلقه‌ها که در قسمت 4 گفتیم و تنها با قراردادن () در اطراف آن می‌توان تابع map را به راحتی پیاده سازی کرد.
نتیجه‌ی کامپایل مثال بالا می‌شود:

```
var item, result;

result = (function() {
  var i, len, results;
  results = [];
  for (i = 0, len = array.length; i < len; i++) {
    item = array[i];
    results.push(item.name);
  }
  return results;
})();
```

Select

یکی دیگر از توابع ES5، تابع filter() است که برای کاهش خانه‌های آرایه استفاده می‌شود.

```
var result = []
for (var i=0; i < array.length; i++)
  if (array[i].name == "test")
    result.push(array[i])

result = array.filter(function(item, i){
  return item.name == "test"
});
```

CoffeeScript با استفاده از کلمه‌ی کلیدی when، عمل فیلتر کردن آیتم‌هایی را که نمی‌خواهیم در آرایه باشند، انجام می‌دهد و در پشت صحنه، با استفاده از یک حلقه‌ی for این عمل را انجام می‌دهد.

```
result = (item for item in array when item.name is "test")
```

در اینجا نیز همانند تابع map برای جلوگیری از تداخل متغیرها از یک تابع بی‌نام استفاده می‌کند.

```
var item, result;

result = (function() {
  var i, len, results;
  results = [];
  for (i = 0, len = array.length; i < len; i++) {
    item = array[i];
    if (item.name === "test") {
      results.push(item);
    }
  }
  return results;
})();
```

نکته‌ی مهم: در صورت فراموشی اضافه کردن () در اطراف حلقه‌ی نوشته شده، نتیجه‌ی صحیحی تولید نخواهد شد و نتها آخرین عضو از خروجی را باز می‌گرداند.

```
var i, item, len, result;

for (i = 0, len = array.length; i < len; i++) {
  item = array[i];
  if (item.name === "test") {
```

```
    result = item;
  }
}
```

قوه‌ی درک CoffeeScript بسیار بالا و انعطاف پذیر است، به مثال زیر توجه کنید:

```
passed = []
failed = []
(if score > 60 then passed else failed).push score for score in [49, 58, 76, 82, 88, 90]

# Or
passed = (score for score in scores when score > 60)
```

و یا در صورتیکه طول خط نوشته شده زیاد باشد می‌توانید به صورت چند خطی آن را بنویسید:

```
passed = []
failed = []
for score in [49, 58, 76, 82, 88, 90]
  (if score > 60 then passed else failed).push score
```

و نتیجه‌ی کامپایل مثال آخر می‌شود:

```
var failed, i, len, passed, ref, score;

passed = [];

failed = [];

ref = [49, 58, 76, 82, 88, 90];
for (i = 0, len = ref.length; i < len; i++) {
  score = ref[i];
  (score > 60 ? passed : failed).push(score);
}
```