عنوان: CoffeeScript #10
نویسنده: وحید محمّدطاهری
تاریخ: ۱۳۹۴/۰۴/۲۲ ۲۰:۱۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, CoffeeScript

اصطلاحات عمومی CoffeeScript

Destructuring Assignments

با استفاده از Destructuring assignments می‌توانید خصوصیات را از آرایه‌ها یا اشیاء، با هر میزان عمقی استخراج کنید.

```
someObject = { a: 'value for a', b: 'value for b' }
{ a, b } = someObject
console.log "a is '#{a}', b is '#{b}'"
```

نتیجه‌ی کامپایل آن می‌شود:

```
var a, b, someObject;

someObject = {
  a: 'value for a',
  b: 'value for b'
};

a = someObject.a, b = someObject.b;

console.log("a is '" + a + "', b is '" + b + "'");
```

این موضوع به خصوص در برنامه‌های کاربردی، وقتی نیاز به ماژول‌های دیگر است، مفید خواهد بود.

```
{join, resolve} = require('path')

join('/Users', 'Vahid')
```

External libraries

استفاده از کتابخانه‌های خارجی دقیقا مانند فراخوانی توابع CoffeeScript است. در پایان نوشتن کدهای CoffeeScript، همه به جاوااسکریپت تبدیل می‌شوند:

```
# Use local alias
$ = jQuery

$ ->
  # DOMContentLoaded
  $(".el").click ->
    alert("Clicked!")
```

نتیجه‌ی کامپایل آن می‌شود:

```
var $;
$ = jQuery;
$(function() {
  return $(".el").click(function() {
    return alert("Clicked!");
  });
});
```

از آنجاییکه خروجی همه کدهای CoffeeScript در داخل یک تابع بدون نام قرار می‌گیرد، می‌توانیم از یک متغیر محلی به نام $ به عنوان نام مستعار jQuery استفاده کنیم. این کار به شما کمک می‌کند تا حتی وقتی که حالت jQuery.noConflict نیز فراخوانی

شده باشد، $ مجدد تعریف شده و اسکریپت ما به طور کامل اجرا شود.

Private variables

کلمه‌ی کلید **do** در CoffeeScript به ما اجازه می‌دهد تا توابع را مستقیما اجرا کنیم و این روش یک راه خوب برای کپسوله سازی و حفاظت از متغیرهاست. در مثال زیر متغیر classToType را در context یک تابع بدون نام که به وسیله‌ی do فراخوانی می‌شود، تعریف کرده‌ایم. تابع بدون نام دوم، مقدار نهایی از type است را برمی گرداند. از آنجایی که classToType در context تعریف شده است و هیچ ارجایی به آن نگهداری نمی‌شود، پس امکان دسترسی به آن خارج از این scope وجود ندارد.

```
# Execute function immediately
type = do ->
  classToType = {}
  for name in "Boolean Number String Function Array Date RegExp Undefined Null".split(" ")
    classToType["[object " + name + "]"] = name.toLowerCase()

  # Return a function
  (obj) ->
    strType = Object::toString.call(obj)
    classToType[strType] or "object"
```

نتیجه‌ی کامپایل آن می‌شود:

```
var type;

type = (function() {
  var classToType, i, len, name, ref;
  classToType = {};
  ref = "Boolean Number String Function Array Date RegExp Undefined Null".split(" ");
  for (i = 0, len = ref.length; i < len; i++) {
    name = ref[i];
    classToType["[object " + name + "]"] = name.toLowerCase();
  }
  return function(obj) {
    var strType;
    strType = Object.prototype.toString.call(obj);
    return classToType[strType] || "object";
  };
})();
```

به بیان دیگر classToType به طور کامل private است و امکان دسترسی به آن از طریق تابع بدون نام اجرا کننده وجود ندارد. این الگو راه بسیار خوب و مناسبی برای کپسوله سازی scope و مخفی سازی متغیرها است.