عنوان: **مدیریت حافظه در JavaScript**
نویسنده: م منفرد
تاریخ: ۱۳۹۲/۰۵/۰۸ ۲۲:۰
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript, Memory Manegement, GC

مدیریت حافظه در JavaScript همانند مدل مدیریت حافظه در .NET می‌باشد. حافظه وقتی مورد نیاز است تخصیص پیدا می‌کند و وقتی دیگر مورد نیاز نیست آزاد می‌شود. این پروسه در CLR به نام جمع آوری زباله یا Garbage Collector یا GC مشهور است. تفاوت عمده فی مابین مدیریت حافظه در .NET با مدیریت حافظه در JavaScript این است که مدیریت حافظه در .NET توسط CLR واحد انجام می‌شود. یعنی پیاده سازی واحدی از GC وجود دارد و شما می‌توانید از نوع فعالیت آن اطمینان حاصل نمایید ولی در JavaScript با توجه به اینکه موتورهای اجرایی مختلفی برای اجرای آن وجود دارد، در سرورها و مرورگرهای مختلف پیاده سازی‌های متفاوتی برای آن وجود دارد. اطلاع از نحوه کار GC می‌تواند به درک ما از JavaScript کمک کرده تا بتوانیم کدهای بهتری در این زبان تولید کنیم.

در این مقاله به بررسی دو الگوریتم عمده GC در JavaScript می‌پردازیم.

1. مدل Reference Counting Garbage Collector

در این مدل از جمع آوری زباله، به ازای ایجاد هر آبجکت در حافظه و یا هر تخصیصی در حافظه، شمارشگری با عنوان reference counter در نظر گرفته می‌شود. هر زمان که به این آبجکت یا حافظه تخصیصی دسترسی ایجاد شود و یا reference داده شود، یک واحد به شمارشگر آن اضافه و هر وقت که رفرنس به حافظه یا آبجکت دیگر مورد استفاده نداشت یا از دسترس خارج شد، یک واحد از شمارشگر آن کاسته می‌شود. این مدل که سریعترین، ساده‌ترین و کم سربارترین مدل GC می‌باشد، وقتی شمارشگر رفرنس حافظه به صفر رسید، حافظه ومنابع سیستم تخصیصی به آن آبجکت آزاد شده و آماده استفاده مجدد می‌شودبه عنوان نمونه به کد زیر دقت کنید:

```
var object1='GC test object 1';
function Test1(){
   var object2='GC test object 2';
   alert (object1+'-' + object2);
}
alert (object1);
```

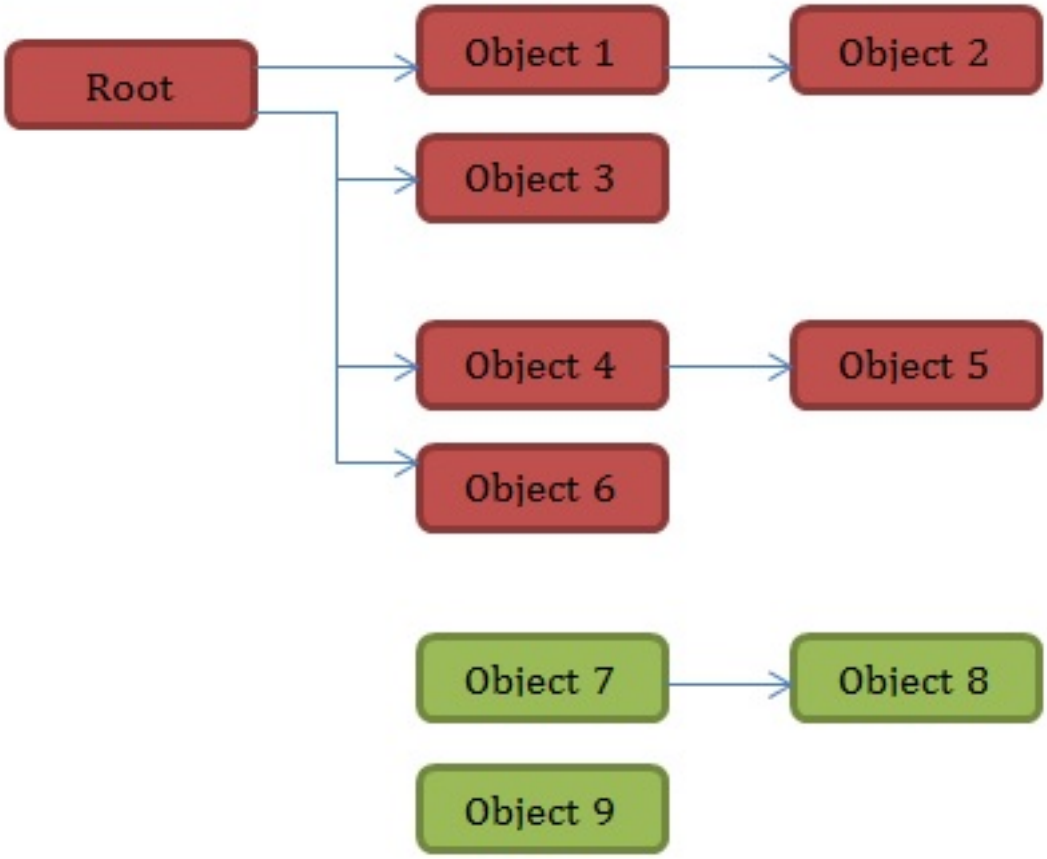
پس از اجرای این کد، جدولی مانند زیر در GC ایجاد می‌شود که به صورت زیر مقدار طی اجرای برنامه مقدار دهی می‌شود:

Reference Counter End Program	Reference Counter Line 6	Reference Counter Line 5	Reference Counter Line 3	Reference Counter Line 1	Object
0	1	1	2	1	object1
0	0	0	1	-	object2

همانطور که در جدول فوق مشخص است، وقتی از متقیر استفاده می‌شود، reference count آن زیاد و وقتی دیگر مورد استفاده ندارد یکی کم می‌شود. وقتی مقدار reference count به صفر برسد، متقیر از حافظه حذف شده و منابع سیستمی آزاد می‌شود. این مدل که در مرورگرهای قدیمی مورد استفاده قرار گرفته است، در صورتی که دو آبجکت به یکدیگر ارجاع داشته باشند، reference counter آن صفر نشده، حافظه و منابع تخصیصی آنها آزاد نمی‌شود و احتمال ایجاد نشت حافظه زیاد می‌شود.

2. مدل Mark-and-Sweep

در این مدل از مدیریت حافظه، برای آبجکت‌های ایجادی در حافظه،GC درخت ارجاعات ایجاد کرده و دقیقا مشخص می‌کند زمانی که یک آبجکت در دسترس نباشد و یا دیگر نیازی به آن نباشد، آن را از حافظه حذف می‌کند. مانند شکل زیر:



در این صورت هنگامی که آبجکت دیگر واقعا مورد نیاز نباشد از حافظه حذف می‌شود. یعنی اگر دو آبجکت به یکدیگر نیز ارجاع داشته باشند هنگامی که دیگر مورد استفاده قرار نگیرند حذف شده و امکان ایجاد نشت حافظه به حداقل می‌رسد.

تفاوت عمده بین GC در Javascript و GC در CLR این است که در زبان‌های مبتنی بر .NET شما می‌توانید به صورت مستقیم GC را صدا زده تا عمل جمع آوری زباله انجام پذیرد ولی در JavaScript هر زمان که نیاز به حافظه بیشتر باشد (و یا در یک زمانبندی مشخص) عمل جمع آوری زباله انجام شده و از طریق کد قابل فراخوانی نمی‌باشد.