بسم الله الرحمن الرحیم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

آموزش بستارها (closure function)

مدرس : مهندس افشین رفوآ

دوره آموزش Jquery

دوره آموزش JavaScript

آموزش بستارها (closure function)

متغیرهای جاوا اسکریپت می توانند محلی یا سراسری می باشند.

می توان با استفاده از closure (بستار) یک متغیر را خصوصی (private) کرد.

Closure (یا به فارسی بستار) تابعی است که شامل body تابع و محیطی می شود که تابع در آن معرفی شده است.

متغیرهای سراسری (Global variable)

یک تابع عملا به تمام متغیرهای تعریف شده درون آن دسترسی دارد:

```
<!DOCTYPE html>
<html>
<body>
  <p>A function can access variables defined inside the function:</p>
  <button type="button" onclick="myFunction()">Click Me!</button>
  <p id="demo"></p>
  <script>
    function myFunction() {
```

```
    var a = 4;
    document.getElementById("demo").innerHTML = a * a;
  }
 </script>
</body>
</html>
```

اما این امکان برای تابع وجود دارد که به متغیرهای تعریف شده بیرون از خود نیز دستیابی پیدا کند:

```
<!DOCTYPE html>
<html>
<body>
 <p>A function can access variables defined outside the function:</p>
 <button type="button" onclick="myFunction()">Click Me!</button>
 <p id="demo"></p>
 <script>
  var a = 4;
  function myFunction() {
   document.getElementById("demo").innerHTML = a * a;
  }
 </script>
</body>
</html>
```

در این مثال، a یک متغیر سراسری است.

در یک صفحه ی وب، تمامی متغیرهای سراسری به شی window تعلق دارند.

متغیرهای سراسری توسط تمامی اسکریپت های شی page (و window) می توانند مورد استفاده قرار بگیرند و حتی تغییر داده شوند.

در مثال اول، a یک متغیر محلی (local) محسوب می شود.

یک متغیر محلی تنها می تواند داخل تابعی که در آن تعریف شده، مورد استفاده قرار گیرد و از این رو توابع و کدهای اسکرپتی دیگر قابلیت دسترسی به آن را ندراند.

دو متغیر با اسمی یکسان که یکی محلی و دیگری سراسری است، دو متغیر کاملا متفاوت با هم در نظر گرفته می شوند، به گونه ای که اصلاح یکی تاثیری بر دیگری ندارد.

نکته: متغیری که بدون استفاده از کلیدواژه ی var اعلان شده باشد، همیشه یک متغیر سراسری تلقی می گردد، صرف نظر اینکه داخل تابع تعریف شده باشد یا بیرون از آن.

چرخه ی حیات متغیر (variable lifetime)

متغیرهای سراسری مادام اینکه application (صفحه یا پنجره ی وب) پا بر جاست، از بین نمی روند.

بر خلاف متغیرهای سراسری، متغیرهای محلی عمر کوتاهی دارند. متغیرهای محلی زمانی که تابع فراخوانی می شود ایجاد شده، و زمانی که تابع پایان می یابد، عمر آن ها به پایان می رسد.

Counter (شمارنده)

فرض بگیرید می خواهیم با استفاده از یک متغیر عملیات شمارشی انجام داده و نیز این شمارنده را در دسترس تمامی توابع قرار دهیم.

برای این منظور می توان از یک متغیر سراسری و یک تابع برای افزایش مقدار شمارنده (counter) بهره گرفت:

```
<!DOCTYPE html>
<html>
<body>
  <p>Counting with a global variable.</p>
  <button type="button" onclick="myFunction()">Count!</button>
  <p id="demo">0</p>
  <script>
    var counter = 0;
    function add() {
      return counter += 1;
    }
    function myFunction() {
      document.getElementById("demo").innerHTML = add();
    }
  </script>
</body>
</html>
```

شمارنده را باید فقط به وسیله ی تابع add() تغییر داد. تنها مشکل این است که هر اسکریپتی (موجود در صفحه) می تواند آن را، حتی بدون فراخوانی add() تغییر دهد. اگر شمارند را داخل تابع تعریف کنیم، هیچ کس قادر نخواهد بود بدون فراخوانی تابع call() آن را اصلاح کند.

مثال:

```
<!DOCTYPE html>
<html>
```

```
<body>
  <p>Counting with a local variable.</p>
  <button type="button" onclick="myFunction()">Count!</button>
  <p id="demo">0</p>
  <script>
    function add() {
      var counter = 0;
      return counter += 1;
    }
    function myFunction() {
      document.getElementById("demo").innerHTML = add();
    }
  </script>
</body>
</html>
```

همان طور که مشاهده می کنید، نتیجه ی دلخواه حاصل نمی گردد. هر بار که تابع add() را صدا می زنید، شمارنده روی 1 تنظیم می شود.

جهت رفع این مشکل می توان از تابع درونی/تابع تودرتو (inner function) جاوا اسکریپت استفاده کرد.

توابع تودرتو جاوا اسکریپت (nested functions)

تمامی توابع دارای حوزه ی دسترسی سراسری هستند (به صورت سراسری به تمام scope دسترسی دارند).

در حقیقت، تمامی توابع به توابع بالا دستی خود دسترسی دارند (توابع زیرین به توابعی که بالای آن تعریف شده اند قابلیت دسترسی دارند).

جاوا اسکریپت از توابع تودرتو (nested) پشتیبانی می کند. توابع تودرتو به scope بالای (توابع بالا دستی) خود دسترسی دارند.

در این مثال، تابع تودرتو plus() به متغیر counter (در تابع پدر) دسترسی دارد:

```
<!DOCTYPE html>
<html>
<body>
  <p>Counting with a local variable.</p>
  <p id="demo">0</p>
  <script>
    document.getElementById("demo").innerHTML = add();
```

```
    function add() {
      var counter = 0;
      function plus() { counter += 1; }
      plus();
      return counter;
    }
  </script>
</body>
</html>
```

حال اگر بتوانیم به تابع plus() از بیرون دسترسی پیدا کنیم، مشکلی که در مثال پیش با آن برخورد کردیم برطرف می شود. همچنین لازم است راهی پیدا کنیم که در آن counter = 0 را تنها یکبار اجرا کنید.

به عبارتی دیگر، به یک closure (بستار) نیاز داریم.

Closure ها در جاوا اسکریپت

توابع خود فراخوان را به یاد دارید؟ به نظر شما این تابع چه نقشی ایفا می کند؟

```
<!DOCTYPE html>
<html>
<body>
  <p>Counting with a local variable.</p>
  <button type="button" onclick="myFunction()">Count!</button>
  <p id="demo">0</p>
  <script>
    var add = (function () {
      var counter = 0;
      return function () { return counter += 1; }
    })();
    function myFunction() {
      document.getElementById("demo").innerHTML = add();
    }
  </script>
</body>
</html>
```

تشریح مثال:

مقدار بازگشتی تابع خود فراخوان به متغیر add تخصیص داده شده است.

تابع خود فراخوان تنها یکبار اجرا می شود، counter را روی 0 تنظیم کرده و به دنبال آن یک عبارت تابع (function expression) باز می گرداند.

از این طریق add به یک تابع تبدیل می شود. مسئله قابل توجه این است که این تابع می تواند به counter (در scope یا حوزه ی دسترسی پدر) دستیابی پیدا کند.

به کمک این روش یک closure ایجاد می شود. closure (بستار) تابعی است که شامل body تابع و محیطی می شود که تابع در آن معرفی شده است.

Closure به تابع این امکان را می دهد که متغیرهای " private " یا " خصوصی " داشته باشد.

Counter توسط حوزه ی دسترسی (scope) تابع بی نام (anonymous function) محافظت می شود، بدین معنا که فقط و فقط تابع add اجازه ی اصلاح و تغییر آن را دارد.