بسم الله الرحمن الرحیم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

فراخوانی تابع (function invocation)

مدرس : مهندس افشین رفوآ

دوره آموزش Jquery

دوره آموزش JavaScript

فراخوانی تابع (function invocation)

چهار روش مختلف برای فراخوانی تابع وجود دارد.

هر روش از نظر شیوه ی مقداردهی اولیه ی کلیدواژه ی *this* از روش های دیگر متمایز است.

کلیدواژه ی *this*

This یک شی است که مالک کد جاری محسوب می شود.

مقدار کلیدواژه ی this، هنگامی که در یک تابع بکار می رود، اشاره به object ای دارد که function متعلق به آن است.

توجه: لازم است دقت داشته باشید که this یک متغیر نیست، بلکه یک واژه ی کلیدی محسوب می شود.

برخلاف متغیر، شما نمی توانید مقدار کلیدواژه ی ذکر شده را تغییر دهید.

فراخوانی یک تابع جاوا اسکریپت

آموختیم که کد (دستور) موجود در یک تابع، زمانی اجرا می شود که " چیزی آن را صدا بزند ".

کد درون تابع زمانی که تابع اعلان (تعریف) می شود، بلافاصله اجرا نمی گردد، بلکه تنها زمانی اجرا می گردد که تابع فراخوانی (invoke) شود.

برخی از برنامه نویس ها به جای " صدا زدن تابع = calling the function " از " فراخوانی تابع = function invocation " استفاده می کنند.

اصطلاحات دیگری نیز برای فراخوانی تابع بکار می رود که عبارتند از " اجرای تابع " ، " راه اندازی تابع ".

در این مقاله ی آموزشی بجای صدا زدن (call) از واژه ی فراخوانی (invoke) استفاده می کنیم، زیرا که یک تابع جاوا اسکریپت می تواند بدون اینکه صدا زده شود، فراخوانده (invoke) شود.

نحوه ی فراخوانی تابع (به عنوان function)

مثال:

```
<!DOCTYPE html>
<html>
<body>
  <p>
    The global function (myFunction) returns the product of the arguments (a ,b):
  </p>
  <p id="demo"></p>
  <script>
    function myFunction(a, b) {
      return a * b;
    }
    document.getElementById("demo").innerHTML = myFunction(10, 2);
  </script>
</body>
</html>
```

تابع مثال فوق به هیچ شی ای تعلق ندارد. اما در جاوا اسکریپت همیشه یک شی سراسری (global object) دارد که تابع متعلق به آن است.

در زبان طراحی وب HTML، شی سراسری پیش فرض خود صفحه ی HTML است. از این رو تابع بکار رفته در مثال بالا به صفحه ی HTML تعلق دارد.

پنجره ی مرورگر خود یک شی است که به آن page object می گویند، بنابراین تابع یاد شده به صورت خودکار تابعی محسوب می شود که به شی پنجره (window object) تعلق دارد. به عبارتی دیگر، myFunction() و window.myFunction() هر دو در اصل یک تابع هستند.

مثال:

```
<!DOCTYPE html>
<html>
<body>
  <p>A global function like myFunction() automatically becomes a window method.</p>
  <p>myFunction() is the same as window.myFunction().</p>
  <p id="demo"></p>
  <script>
    function myFunction(a, b) {
      return a * b;
    }
    document.getElementById("demo").innerHTML = window.myFunction(10, 2);
  </script>
</body>
</html>
```

این روش، اگرچه یک روش متداول فراخوانی تابع در زبان جاوا اسکریپت محسوب می شود، اما به طور کلی در برنامه نویسی استفاده از آن توصیه نمی شود.

متدها، توابع و متغیرهای سراسری می توانند به راحتی منجر به ایجاد تداخل اسمی یا خطا در object سراسری شوند.

شی سراسری (Global object)

هنگامی که یک تابع (که متعلق به شی خاصی نیست) فراخوانی می شود، مقدار this شی سراسری درنظر گرفته می شود و تابع نام برده به شی سراسری تعلق پیدا می کند.

به عنوان مثال می توان به یک مرورگر وب اشاره کرد که پنجره ی آن، شی سراسری تلقی می گردد (در مرورگر وب، پنجره ی مرورگر همان شی سراسری است).

مثال زیر شی window را به عنوان مقدار کلیدواژه ی *this* بازیابی می کند:

```
<!DOCTYPE html>
<html>
<body>
  <p>In HTML the value of <b>this</b>, in a global function, is the window object.</p>
  <p id="demo"></p>
  <script>
    function myFunction() {
      return this;
    }
    document.getElementById("demo").innerHTML = myFunction();
  </script>
</body>
</html>
```

فراخوانی یک function به عنوان تابع سراسری باعث می شود مقدار this، شی سراسری درنظر گرفته شود.

استفاده از شی window به عنوان یک متغیر، ممکن است باعث از کار افتادن برنامه شود.

فراخوانی تابع (به عنوان متد)

در زبان جاوا اسکریپت می توان تابع را به عنوان متدهای یک شی تعریف کرد.

مثال زیر یک شی به نام myObject ایجاد کرده، سپس دو خاصیت firstName و lastName و یک متد به نام fullName به آن اضافه می کند.

```
<!DOCTYPE html>
<html>
<body>
  <p>myObject.fullName() will return John Doe:</p>
  <p id="demo"></p>
  <script>
    var myObject = {
      firstName: "John",
      lastName: "Doe",
      fullName: function () {
        return this.firstName + " " + this.lastName;
      }
    }
    document.getElementById("demo").innerHTML = myObject.fullName();
  </script>
</body>
</html>
```

fullName یک تابع است و این تابع به شی myObject تعلق دارد. در واقع می توان گفت که شی مذکور مالک تابع fullname است.

کلیدواژه ی this، شی ای است که کد جاوا اسکریپت به آن تعلق دارد. در این مثال، شی myObject مقدار this محسوب می شود.

می توانید خودتان امتحان کنید. متد fullName را گونه ای اصلاح کنید که مقدار this را برگرداند:

```
<!DOCTYPE html>
<html>
<body>
  <p>The value of <b>this</b>, in an object method, is the owner object.</p>
  <p id="demo"></p>
  <script>
    var myObject = {
      firstName: "John",
      lastName: "Doe",
      fullName: function () {
        return this;
      }
    }
    document.getElementById("demo").innerHTML = myObject.fullName();
  </script>
</body>
</html>
```

توجه: فراخوانی تابع به عنوان متد شی، باعث می شود مقدار this خود شی محسوب شود.

فراخوانی تابع با function constructor

اگر کلیدواژه ی new پیش از دستور فراخوانی تابع قرار گیرد، حتم بدانید که آن دستور فراخوانی یک سازنده (constructor) است. اینگونه بنظر می رسد که یک تابع جدید ایجاد می کنید، اما از آن جایی که توابع جاوا اسکریپت خود شی محسوب می شوند، شما در واقع دارید که شی جدید ایجاد می کنید:

```
<!DOCTYPE html>
<html>
<body>
  <p>In this example, myFunction is a function constructor:</p>
  <p id="demo"></p>
  <script>
    function myFunction(arg1, arg2) {
      this.firstName = arg1;
```

```
    this.lastName = arg2;
  }
  var x = new myFunction("John", "Doe")
  document.getElementById("demo").innerHTML = x.firstName;
</script>
</body>
</html>
```

سازنده (کلیدواژه ی new) یک شی جدید ایجاد می کند. شی جدید خاصیت ها (property) و متدهای خود را از سازنده ی خود (constructor) به ارث می برد.

توجه: کلیدواژه ی this در سازنده دارای مقدار نیست. مقدار this شی جدیدی است که به هنگام فراخوانی تابع ایجاد می گردد.

فراخوانی تابع با متد function

همان طور پیش تر توضیح دادیم، توابع در جاوا اسکریپت، خود شی هستند. همچنین آموختیم که توابع جاوا اسکریپت دارای خاصیت ها و توابعی هستند.

call() و apply() هر دو متدهای function از پیش تعریف شده در زبان جاوا اسکریپت هستند. از هر دو متد ذکر شده می توان برای فراخوانی تابع استفاده کرد، همچنین هر دو متد باید شی پدر (owner object) را به عنوان اولین پارامتر خود داشته باشند.

مثال:

```
<!DOCTYPE html>
<html>
<body>
  <p id="demo"></p>
  <script>
    var myObject;
    function myFunction(a, b) {
      return a * b;
    }
    myObject = myFunction.call(myObject, 10, 2);   // Will return 20
    document.getElementById("demo").innerHTML = myObject;
  </script>
</body>
</html>
```

مثال:

```
<!DOCTYPE html>
<html>
<body>
  <p id="demo"></p>
  <script>
    var myObject, myArray;
    function myFunction(a, b) {
      return a * b;
    }
    myArray = [10, 2]
    myObject = myFunction.apply(myObject, myArray);     // Will return 20
    document.getElementById("demo").innerHTML = myObject;
  </script>
</body>
</html>
```

هر دو متد شی پدر را به عنوان اولین آرگومان می پذیرند. تنها تفاوتی که وجود دارد این است که call() آرگومان های تابع (function argument) را به صورت جداگانه می گیرد، در حالی که apply() آرگومان های تابع را به صورت آرایه دریافت می کند.

در حالت کد نویسی دقیق/strict mode، اولین آرگومان در تابع فراخوانده شده مقدار this محسوب می شود، حتی اگر هم آرگومان یک شی باشد.

در حالت عادی (non-strict mode)، اگر مقدار اولین آرگومان null یا undefined باشد، این مقدار با شی سراسری (global object) جایگزین می شود.

توجه: با apply() و call() می توان this را مقداردهی کرده و یک تابع را به عنوان متد جدیدی از شی موجود فراخوانی کرد.