بسم الله الرحمن الرحیم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

اشیا در جاوا اسکریپت

مدرس : مهندس افشین رفوآ

دوره آموزش Jquery

دوره آموزش JavaScript

اشیا در جاوا اسکریپت

جاوا اسکریپت یک زبان برنامه نویسی مبتنی بر شی است، بدین معنا که در صورت یادگیری کامل مفهوم آن، فراگیری زبان مزبور بسیار سهل می گردد.

در این زبان محور همه چیز یک شی است:

متغیرهای بولی می توانند شی باشند (یا داده های اولیه ای که به صورت شی مورد استفاده قرار می گیرند)

متغیرهای عددی می توانند شی باشند (یا داده های اولیه ای که به صورت شی مورد استفاده قرار می گیرند)

متغیرهای رشته ای می توانند شی باشند (یا داده های اولیه ای که به صورت شی مورد استفاده قرار می گیرند)

متغیر تاریخ (Date) همیشه شی (object) است

متغیرهای ریاضی (Math) همیشه شی محسوب می شوند

عبارات باقاعده (regular expression) همیشه شی است

آرایه ها همیشه شی محسوب می شوند

توابع همیشه شی تلقی می گردند

خود Objectها نیز شی هستند

در زبان جاوا اسکریپت تمامی مقادیر، به استثنای مقادیر اولیه، شی محسوب می شوند.

مقادیر اولیه (primitive values) عبارتند از: رشته یا مقدار رشته ای ("John Doe")، اعداد یا مقدار عددی (3.14)، true، false، null و undefined.

اشیا عملا متغیرهایی هستند که خود دربردارنده ی چندین متغیر می باشند!

متغیرهای جاوا اسکریپت قادراند تنها یک مقدار را در خود ذخیره کنند.

مثال:

```
<!DOCTYPE html>
<html>
<body>
  <p>Creating a JavaScript Variable.</p>
  <p id="demo"></p>
  <script>
    var person = "John Doe";
    document.getElementById("demo").innerHTML = person;
  </script>
</body>
</html>
```

اشیا نیز نوعی متغیر هستند، با این وجود که شی می تواند چندین مقدار متعدد را در خود نگه دارد.

مقادیر به صورت جفت های name : value (اسم : مقدار) نوشته می شوند.

مثال:

```
<!DOCTYPE html>
<html>
<body>
  <p>Creating a JavaScript Object.</p>
  <p id="demo"></p>
  <script>
    var person = {
      firstName: "John",
      lastName: "Doe",
```

```
    age: 50,
    eyeColor: "blue"
  };
  document.getElementById("demo").innerHTML =
  person.firstName + " " + person.lastName;
</script>
</body>
</html>
```

نکته: شی عملا یک مجموعه ی نامرتب از متغیرهاست که named values یا مقادیر نام گذاری شده خوانده می شوند.

خواص مربوط به object

اسمی که به عنوان یک ظرف برای ذخیره ی مقدار عمل می کند، property (خاصیت) نامیده می شود.

Property	Value
firstName	John
lastName	Doe
age	50
eyeColor	blue

اشیایی که به صورت جفت های اسم : مقدار نوشته می شوند مشابه:

آرایه های شرکت پذیر (associative arrays) در PHP

Dictionary ها در Python

Hash table در C

Hash map در Java

Hash ها در زبان های Ruby و Perl هستند.

توابع مربوط به object

توابع (method) به معنای دقیق کلمه عملیاتی هستند که می توان روی اشیا اجرا کرد.

خاصیت های مربوطه به object هم می توانند مقادیر اولیه، دیگر اشیا و هم دیگر توابع باشند.

object method (تابع شی) یک object property (خصوصیت شی) است که دربردارنده ی تعریف تابع می باشد.

Property	Value
firstName	John
lastName	Doe
age	50
eyeColor	blue
fullName	function() {return this.firstName + " " + this.lastName;}

توجه: اشیا (در زبان جاوا اسکریپت) در واقع ظرف هایی برای نگهداری مقادیر نام گذاری شده به نام property ها و method ها هستند.

نحوه ی ایجاد شی در جاوا اسکریپت

زبان جاوا اسکریپت این امکان را به شما می دهد که اشیا خود را ایجاد و تعریف کنید.

راه های مختلفی برای ایجاد اشیا در زبان مذکور وجود دارد که زیر شرح داده شده:

با استفاده از یک object literal، یک شی جدید تعریف و ایجاد کنید.

با استفاده از کلیدواژه ی new، یک شی جدید ایجاد کنید.

ابتدا یک سازنده ی شی (object constructer) تعریف کرده، سپس اشیا جدیدی از نوع شی ساخته شده ایجاد کنید (تعریف شی با استفاده از یک تابع به عنوان سازنده (Constractor) سپس ایجاد یک نمونه ی جدید از شی مذکور).

نکته: در ویرایش 5 زبان جاوا اسکریپت (ECMAScript 5)، می توان با بهره گیری از تابع Object.create() یک شی جدید ایجاد کرد.

ایجاد شی با استفاده از object literal

این روش ساده ترین راه برای ایجاد یک شی جدید محسوب می شود.

با استفاده از object literal، شی ای جدید در تنها یک دستور تعریف و ایجاد می کنید.

Object literal در حقیقت یک لیست متشکل از جفت های اسم : مقدار مانند age:50 است که داخل براکت " {} " باز و بسته محصور می شود.

مثال زیر یک شی جدید که دارای چهار خاصیت است، ایجاد می کند.

مثال:

```
<!DOCTYPE html>
<html>
<body>
  <p>Creating a JavaScript Object.</p>
  <p id="demo"></p>
  <script>
    var person = { firstName: "John", lastName: "Doe", age: 50, eyeColor: "blue" };
    document.getElementById("demo").innerHTML =
    person.firstName + " is " + person.age + " years old.";
  </script>
</body>
</html>
```

خطوط فاصله و اینکه خط کجا پایان می یابد، از اهمیت چندانی برخوردار نیست. تعریف یک شی می تواند چندین خط ادامه داشته باشد.

مثال:

```
<!DOCTYPE html>
<html>
<body>
  <p>Creating a JavaScript Object.</p>
  <p id="demo"></p>
  <script>
    var person = {
      firstName: "John",
      lastName: "Doe",
      age: 50,
      eyeColor: "blue"
    };
    document.getElementById("demo").innerHTML =
    person.firstName + " is " + person.age + " years old.";
  </script>
</body>
</html>
```

ایجاد شی به وسیله ی کلیدواژه ی new

مثال زیر نیز همانند مثال قبلی شی ای جدید با چهار خاصیت ایجاد می کند:

```
<!DOCTYPE html>
<html>
<body>
  <p id="demo"></p>
  <script>
    var person = new Object();
    person.firstName = "John";
    person.lastName = "Doe";
    person.age = 50;
    person.eyeColor = "blue";
    document.getElementById("demo").innerHTML =
    person.firstName + " is " + person.age + " years old.";
  </script>
</body>
</html>
```

هر دو روش فوق، در اصل یک کار را انجام می دهند و نیازی به استفاده از تابع new Object() برای ایجاد شی جدید نیست.

به منظور حفظ سادگی، بهبود خوانایی و همچنین افزایش سرعت اجرا، روش اول بهترین گزینه برای ایجاد شی جدید می باشد.

استفاده از object constructor (سازنده ی شی) برای ایجاد شی جدید

دو روش نام برده از این جهت که تنها قادر به ایجاد یک شی هستند، دارای محدودیت هایی می باشند.

گاهی نیاز به یک " object type " داریم تا بتوانیم با استفاده از آن اشیا متعددی از یک نوع ایجاد کنیم.

روش استاندارد ایجاد یک " object type "، استفاده از یک تابع به عنوان سازنده شی است.

مثال:

```
<!DOCTYPE html>
<html>
<body>
  <p id="demo"></p>
  <script>
    function person(first, last, age, eye) {
      this.firstName = first;
      this.lastName = last;
      this.age = age;
      this.eyeColor = eye;
    }
    var myFather = new person("John", "Doe", 50, "blue");
    var myMother = new person("Sally", "Rally", 48, "green");
    document.getElementById("demo").innerHTML =
    "My father is " + myFather.age + ". My mother is " + myMother.age;
  </script>
</body>
</html>
```

تابع بکار رفته در مثال بالا (person)، یک object constructor (سازنده ی شی) است.

پس از اینکه object constructor فراهم شد، می توانید اشیا متعددی از نوع یکسان ایجاد کنید:

```
var myFather = new person("John", "Doe", 50, "blue");
var myMother = new person("Sally", "Rally", 48, "green");
```

کلیدواژه ی *this*

این کلمه ی کلیدی در هر لحظه به شی جاری از کلاس که در آن لحظه در حال ساخت (یا ویرایش) است، اشاره می کند.

سازنده های توکار (javascript built-in constructor)

جاوا اسکریپت برای اشیا محلی (native)، سازنده هایی درون خود دارد:

```
<!DOCTYPE html>
<html>
<body>
<p id="demo"></p>
<script>
var x1 = new Object();  // A new Object object
var x2 = new String();  // A new String object
var x3 = new Number();  // A new Number object
var x4 = new Boolean(); // A new Boolean object
var x5 = new Array();   // A new Array object
var x6 = new RegExp();  // A new RegExp object
var x7 = new Function(); // A new Function object
var x8 = new Date();    // A new Date object
document.getElementById("demo").innerHTML =
"x1: " + typeof x1 + "<br>" +
"x2: " + typeof x2 + "<br>" +
"x3: " + typeof x3 + "<br>" +
"x4: " + typeof x4 + "<br>" +
"x5: " + typeof x5 + "<br>" +
"x6: " + typeof x6 + "<br>" +
"x7: " + typeof x7 + "<br>" +
"x8: " + typeof x8 + "<br>";
</script>
<p>There is no need to use String(), Number(), Boolean(), Array(), and RegExp()</p>
<p>Read the JavaScript tutorials.</p>
</body>
</html>
```

همان طور که مشاهده می کنید، شی Math() در لیست بالا وجود ندارد. Math یک شی سراسری است و امکان استفاده ی کلیدواژه ی new برای آن وجود ندارد.

جاوا اسکریپت نسخه های به صورت شی از داده های اولیه دارد،مانند شی String، Number و Boolean .

از این رو نیازی به ایجاد اشیا پیچیده، نیست. مقادیر اولیه (primitive values) به مراتب سریع تر اجرا می شوند.

همچنین احتیاجی به استفاده از تابع new Array() نیست. در عوض توصیه می کنیم از روش " [] " (array literal) استفاده کنید.

دلیلی نیز برای استفاده از متد new RegExp() وجود ندارد. به عنوان جایگزینی بهتر می توان از pattern literal مانند /()/ استفاده کنید.

لزومی ندارد از new Function() استفاده کنید و بجای آن بهتر است function expression هایی مثل : function () {} را مورد استفاده قرار دهید.

بجای تابع new Object() نیز توصیه می شود از object literal ها " {} " استفاده کنید.

مثال:

```
<!DOCTYPE html>
<html>
<body>
  <p id="demo"></p>
  <script>
    var x1 = {};
    var x2 = "";
    var x3 = 0;
    var x4 = false;
    var x5 = [];
    var x6 = /()/;
    var x7 = function () { };
    document.getElementById("demo").innerHTML =
    "x1: " + typeof x1 + "<br>" +
    "x2: " + typeof x2 + "<br>" +
    "x3: " + typeof x3 + "<br>" +
    "x4: " + typeof x4 + "<br>" +
    "x5: " + typeof x5 + "<br>" +
    "x6: " + typeof x6 + "<br>" +
    "x7: " + typeof x7 + "<br>";
  </script>
</body>
</html>
```

اشیا جاوا اسکریپت ناپایدار هستند!

اشیا ناپایدار (mutable) هستند، آدرس دهی و اشاره به آن ها باید با ارجاع (reference) صورت گیرد و نه با مقدار.

فرض بگیرید y یک شی است، دستوری که به دنبال آن می آید یک کپی یا نسخه ی عینی از شی y ایجاد نمی کند:

```
var x = y;  // This will not create a copy of y.
```

شی x یک کپی از y محسوب نمی شود. در واقع x و y هر دو به یک شی اشاره می کنند.

هر تغییری که به y وارد می شود، شی x را نیز تحت تاثیر قرار می دهد زیرا که x و y هر دو یک شی هستند.

مثال:

```
<!DOCTYPE html>
<html>
<body>
  <p>JavaScript objects are mutable.</p>
  <p>Any changes to a copy of an object will also change the original.</p>
  <p id="demo"></p>
  <script>
    var person = { firstName: "John", lastName: "Doe", age: 50, eyeColor: "blue" }
    var x = person;
    x.age = 10;
    document.getElementById("demo").innerHTML =
    person.firstName + " is " + person.age + " years old.";
  </script>
</body>
</html>
```