بسم اللّه الرحمن الرحیم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

اشتباهات معمولی که در کد نویسی تکرار می شود

مدرس: مهندس افشین رفوآ

آدرس آموزشگاه. تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 561 - واحد 7
88146323 - 88446780 - 88146330

اشتباهات معمولی که در کد نویسی تکرار می شود

استفاده ی نادرست از عملگر تخصیص (assignment operator)

جاوا اسکریپت ممکن است در صورت استفاده ی نادرست برنامه نویس از عملگر تخصیص (=) بجای عملگر مقایسه (==) (در یک دستور) نتایج ناصحیح یا غیر قابل پیش بینی ارائه دهد.

دستور if که در مثال زیر بکابرده شده، به این خاطر که x مساوی 10 نیست، مقدار false بازمی گرداند.

```
var x = 0;
        if (x == 10)
```

دستور if مثال زیر، مقدار true بازگردانی می کند، زیرا که 10 صحیح می باشد.

```
var x = 0;
        if (x = 10)
```

مثال زیر مقدار false برمی گرداند زیرا که 0 ناصحیح (false) است.

```
var x = 0;
        if (x = 0)
```

در مقایسه های معمولی، نوع داده از اهمیت خاصی برخوردار نیست. برای مثال، دستور if بکار رفته در مثال زیر مقدار true بازیابی می کند.

```
var x = 10;
        var y = "10";
        if (x == y)
```

اما در مقایسه های دقیق، علاوه بر مقدار، نوع داده نیز حائز اهیمت می باشد. دستور if مثال زیر، مقدار false برمی گرداند.

```
var x = 10;
        var y = "10";
        if (x === y)
```

برنامه نویس ها اغلب فراموش می کنند که دستور switch از مقایسه های دقیق استفاده می کند (نوع داده و مقدار را در مقایسه درنظر می گیرد).

این case switch یک پنجره ی هشدار نمایش می دهد که دربردانده ی پیغام Hello می باشد.

```
var x = 10;
        switch(x) {
            case 10: alert("Hello");
        }
```

مثال زیر پیغام Hello را نمایش نمی دهد.

```
var x = 10;
        switch(x) {
            case "10": alert("Hello");
        }
```

اشتباه گرفتن عملیات جمع (addition) و اتصال (concatenation)

در عملیات جمع، اعداد را به هم اضافه می کنیم.

در عملیات اتصال (concatenation)، دو رشته را به هم متصل می کنیم.

در زبان جاوا اسکریپت برای هر دو منظور از عملگر (+) استفاده می کنیم.

بنابراین افزودن دو عدد به یکدیگر و یک عدد به یک رشته که در خود عدد دارد، نتایج کاملا متفاوتی را بدست می دهد.

```
var x = 10 + 5;           // the result in x is 15
        var x = 10 + "5";         // the result in x is "105"
```

مشکل جاوا اسکریپت با اعداد با ممیز شناور

تمامی اعداد در جاوا اسکریپت به صورت اعداد با ممیز شناور 64 بیتی ذخیره می گردند.

کلیه ی زبان های برنامه نویسی از جمله جاوا اسکریپت در برخورد با اعداد با ممیز شناور و محاسبه ی دقیق آن ها با مشکل مواجه می شوند.

مثال

```
var x = 0.1;
        var y = 0.2;
        var z = x + y            // the result in z will not be
0.3
        if (z == 0.3)            // this if test will fail
```

جهت رفع این مشکل توصیه می شود از ضرب و تقسیم کمک بگیرید.

```
var z = (x * 10 + y * 10) / 10;       // z will be 0.3
```

نصف کردن رشته (تقسیم یک رشته به دو خط)

جاوا اسکریپت به شما این امکان را می دهد که یک دستور را به دو بخش تقسیم کرده و بخش دوم را درون خط بعدی قرار دهید.

مثال 1

```
var x =
        "Hello World!";
```

اما باید توجه داشته باشید که دو نیم کردن یک دستور درست از وسط یک رشته امکان پذیر نمی باشد.

مثال 2

```
var x = "Hello
        World!";
```

در صورت نیاز به نصف کردن یک دستور در وسط یک رشته، لازم است از کاراکتر "\" استفاده کنید.

مثال 3

```
var x = "Hello \
        World!";
```

جای گذاری نقطه ویرگول در مکان اشتباه

در مثال زیر به دلیل قرار دادن نقطه ویرگول پس از دستور if، دستورات داخل قطعه کد (code block) مورد نظر بدون در نظر گرفتن مقدار x اجرا می شود.

```
if (x == 19);
        {
            // code block
        }
```

دو نیم کردن دستور Return

جاوا اسکریپت به صورت پیش فرض دستورات خود را در انتهای جمله و با بهره گیری از کاراکتر نقطه ویرگول می بندد. به همین دلیل نیز دو نمونه ی زیر، نتیجه ای یکسان برمی گردانند.

مثال 1

```
function myFunction(a) {
            var power = 10
            return a * power
        }
```

مثال 2

```
function myFunction(a) {
            var power = 10;
            return a * power;
        }
```

همچنین جاوا اسکریپت به شما امکان می دهد یک دستور را به دو بخش تقسیم کنید.

به همین خاطر مثال سوم نیز نتیجه ای یکسان را بدست می دهد.

```
function myFunction(a) {
            var
            power = 10;
            return a * power;
        }
```

اینجا با این سوال مواجه می شویم که، چه رخ می دهد اگر دستور return را مانند نمونه ی زیر به دو بخش تقسیم (break) کنیم؟

مثال 4

```
function myFunction(a) {
            var
            power = 10;
```

```
        return a * power;
    }
```

function (تابع) مقدار undefined را برمی گرداند. آیا می دانید علت آن چیست؟

علت آن این است که جاوا اسکریپت دستور شما را اینگونه تفسیر می کند.

مثال 5

```
function myFunction(a) {
        var
        power = 10;
        return;
        a * power;
    }
```

توضیح

چنانچه دستوری مانند مثال زیر، اینگونه ناقص بود.

```
Var
```

جاوا اسکریپت سعی می کند با خواندن خط بعدی، دستور را کامل کند.

```
power = 10;
```

اما از آنجایی که دستور کامل می باشد.

```
Return
```

جاوا اسکریپت به صورت خودکار، همانند مثال زیر آن را می بندد.

```
return;
```

این امر به این دلیل اتفاق می دهد که بستن (خاتمه دادن) دستورات در جاوا اسکریپت با نقطه ویرگول، اختیاری می باشد.

جاوا اسکریپت return statement را به این خاطر می بندد که این دستور یک دستور کامل محسوب می شود.

دسترسی به (المان های) آرایه با استفاده از اندیس های نام گذای شده

بسیاری از زبان های برنامه نویسی از آرایه هایی که دارای اندیس های نام گذاری شده (named index) هستند، پشتیبانی می کنند.

به چنین آرایه هایی در اصطلاح، آرایه ی شرکت پذیر (associative arrays) می گویند.

همان طور که در مباحث پیشین شرح داده شد، زبان برنامه نویسی تحت وب جاوا اسکریپت از آرایه های شرکت پذیر (انجمنی) پشتیبانی نمی کند.

زبان جاوا اسکریپت در عوض از آرایه هایی که المان های آن شماره گذاری شده است (numbered index) پشتیبانی می کند.

مثال

```
var person = [];
        person[0] = "John";
        person[1] = "Doe";
        person[2] = 46;
        var x = person.length;          // person.length will
return 3
        var y = person[0];              // person[0] will return
"John"
```

باید توجه داشته باشید که در زبان جاوااسکریپت، objects (اشیا) از اندیس های نامگذاری شده استفاده می کنند.

چنانچه به هنگام دسترسی به یک آرایه از اندیس نام گذاری شده استفاده کنید، در آن صورت جاوا اسکریپت آرایه را به یک شی استاندارد تبدیل (redefine) می کند.

پس از تعریف مجدد خودکار (تبدیل پیش فرض آرایه به شی حین دسترسی به یک آرایه توسط اندیس نام گذاری شده)، توابع و خواص (property) مربوط به آرایه به طور قطع نتیجه ی نادرست یا تعریف نشده بازمی گرداند.

مثال

```
var person = [];
        person["firstName"] = "John";
        person["lastName"] = "Doe";
        person["age"] = 46;
        var x = person.length;          // person.length will
return 0
        var y = person[0];              // person[0] will return
undefined
```

قرار دادن کاراکتر ویرگول در انتهای تعریف یک آرایه

نمونه ی زیر بدلیل قرار دادن کاراکتر ویرگول در انتهای تعریف آرایه کاملا نادرست می باشد.

```
points = [40, 100, 1, 5, 25, 10,];
```

در نتیجه، برخی از موتورهای جاوا اسکریپت و JSON از کار افتاده و یا به گونه ای غیر منتظره عمل می کنند.

صحیح

```
points = [40, 100, 1, 5, 25, 10];
```

قرار دادن کاراکتر ویرگول در انتهای تعریف یک شی

مثال زیر تعریف یک شی را به طور نادرست، با ویرگول خاتمه داده

```
person = {firstName:"John", lastName:"Doe", age:46,}
```

همان گونه که در مورد آرایه توضیح داده شد، استفاده از ویرگول در انتهای تعریف شی باعث از کار افتادن موتور جاوا اسکریپت و JSON شده یا نتیجه ی کاملا پیش بینی نشده ارائه می هد.

نمونه ی صحیح

```
person = {firstName:"John", lastName:"Doe", age:46}
```

undefined با null یکسان نیست.

لازم به ذکر است که در زبان جاوا اسکریپت، مقدار null ویژه ی اشیا بوده و undefined مختص متغیرها، خاصیت ها و همچنین توابع می باشد.

برای اینکه یک شی null باشد، لازم است آن شی تعریف شده باشد و در غیر این صورت تعریف نشده محسوب می شود.

اگر می خواهید بررسی کنید که آیا یک شی موجود می باشد یا خیر، آن شی باید حتما تعریف شده باشد، در غیر این صورت یک پیغام خطا ایجاد می شود (یا به عبارتی دیگر یک خطا پرتاب می شود).

نمونه ی ناصحیح

```
if (myObj !== null && typeof myObj !== "undefined")
```

برای این منظور لازم است ابتدا تابع typeof () را تایپ کنید.

نمونه ی صحیح

```
if (typeof myObj !== "undefined" && myObj !== null)
```

جاوا اسکریپت ویژه ی هر قطعه کد (code block)، یک scope (دامنه) جدید ایجاد نمی کند.

اگرچه بسیاری از زبان های برنامه نویسی از این امکان پشتیبانی می کنند، اما جاوا اسکریپت قطعا یکی از آن زبان ها نیست.

اغلب برنامه نویسان زبان جاوا اسکریپت به طور غلط فرض می گیرند که کد زیر undefined بازمی گرداند.

```
for (var i = 0; i < 10; i++) {
            // some code
        }
        return i;
```