

# قواعد کد نویسی در فلش (اکشن اسکریپت)

## Code Style in FLASH (ActionScript)

گردآوری، تالیف و ترجمه:

امین کریمی

[www.amindesign.ir](http://www.amindesign.ir)

صاحب امتیاز:

[www.amindesign.ir](http://www.amindesign.ir)

## الف) قواعد نام گذاری

### ۱) متغیرها:

- Actionscript به بزرگی و کوچکی حروف حساس است بطور مثال myBall و MyBall رو متغیر متفاوت هستند ولی در عین حال دقت کنید برای جلوگیری از گیج شدن برنامه نویس برای متغیرهای خود نام یکسان با اندازه حروف متفاوت انتخاب نکنید.
  - کلمات را درست اختصار کنید که مفهوم چند چیز را نرساند، به عنوان مثال sec هم نماد section است و هم نماد second.
  - از کوتاه سازی نامها پرهیز شود.  
مثال:  

```
computeAverage(); // NOT: compAvg();
```

واژه های رایج هیچگاه نباید بصورت کوتاه شده نوشته شوند. کوتاه سازی های زیر غیر مجاز می باشند:

    - cmp بجای command
    - cp بجای copy
    - pt بجای point
    - comp بجای compute
    - init بجای initialize

همچنین کوتاه نامها یا سرنامهای معروف نباید به صورت کامل نوشته شوند. مثلاً هرگز نباید:

    - HypertextMarkupLanguage را بجای html،
    - CentralProcessingUnit را بجای cpu و یا
    - PriceEarningRatio را بجای pe نوشت.
  - نام متغیر را با حرف کوچک آغاز کنید.
  - برای نام های چندکلمه ای، کلمات را به هم چسبیده بنویسید بطوریکه حرف اول تمام کلمات بجز کلمه اول بزرگ باشد.
- myFirstVar
- نام متغیرها تنها می تواند شامل حروف، اعداد و علامت دلار (\$) باشد.

- نام متغیر را با عدد آغاز نکنید.

مثال غلط:

my var , my/var , 5 var , 25var

مثال درست:

myVar, var25

- کلماتی را که جزئی از زبان اکشن اسکریپت هستند (keywords) را برای نام متغیر انتخاب نکنید.

همینطور در مورد کلمات مربوط به ساختارها و اجزای فلش

مثال غلط:

Var = "2"

Return = "flash"

Interface = 10

TextField = "my var"

- از کدهای رزوز شده بپرهیزید!

|             |            |               |            |
|-------------|------------|---------------|------------|
| add         | and        | break         | case       |
| catch       | class      | continue      | default    |
| delete      | do         | dynamic       | else       |
| eq          | extends    | false         | finally    |
| for         | function   | ge            | get        |
| gt          | if         | ifFrameLoaded | implements |
| import      | in         | instanceof    | interface  |
| intrinsic   | le         | it            | ne         |
| new         | not        | null          | on         |
| onClipEvent | or         | private       | public     |
| return      | set        | static        | super      |
| switch      | tellTarget | this          | throw      |
| try         | typeof     | var           | void       |
| while       | with       |               |            |

|              |          |           |           |
|--------------|----------|-----------|-----------|
| as           | abstract | Boolean   | bytes     |
| char         | const    | debugger  | double    |
| enum         | export   | final     | float     |
| goto         | is       | long      | namespace |
| native       | package  | protected | short     |
| synchronized | throws   | transient | use       |
| volatile     |          |           |           |

- در اکشن اسکریپت نیازی به گذاشتن کلمه var قبل از نام متغیر و تعیین نوع آن بعد از آن نیست ولی ترجیحا برای خوانا تر بودن و سریع تر اجرا شدن کدها این کار را بکنید.

```
var font_array:Array = TextField.getFontList()
```

## ۲) مقادیر ثابت:

- برای نام گذاری مقادیر ثابت از حروف بزرگ استفاده کنید.
- در نامهای چندکلمه ای مقادیر ثابت (که تماماً باید با حروف بزرگ نوشته شوند) کلمات را با `under` score ( \_ ) از هم جدا کنید.

`BASE_NUMBER = 10`

### ۳) توابع:

- نام توابع باید با حروف کوچک آغاز شود.
- در صورت چند کلمه ای بودن حرف اول تمام کلمات آن غیر از کلمه اول بزرگ نوشته می شود.  
`gerCurrendSound()`
- نام گذاري توابع و متغیر ها ظاهرا یکسان می باشد با این تفاوت که در پایان نام توابع ( ) قرار می گیرد.

- برای توابعی که یک مقدار را محاسبه می کنند می توان از واژه `compute` بهره جست.  
مثال:

```
valueSet->computeAverage();  
matrix->computeInverse();
```

- برای توابعی که یک مقدار را جستجو می کنند می توان از واژه `find` بهره جست.  
مثال:

```
vertex.findNearestVertex();  
matrix.findMinElement();
```

- برای توابعی که برای مقداردهی اولیه استفاده می شوند، می توان از واژه `initialize` بهره گرفت.  
مثال:

```
printer.initializeFontSet();
```

نباید از گونه کوتاه شده ی "init" بجای "initialize" استفاده کرد.

#### ۴) متدها:

- نام گذاري متدها همانند توابع است که بايد شامل فعل متد باشد.

Sing()

runFast()

- براي متدهايي که عملياتهاي مکمل يا ضد يکديگر را انجام مي دهند بايد نامهاي متقارن و مناسب انتخاب نمود.

مثال:

get/set, add/remove, create/destroy, start/stop, insert/delete, increment/decrement,  
old/new, begin/end, first/last, up/down, min/max, next/previous, old/new,  
open/close, show/hide, suspend/resume

## ۵) حلقه ها:

- برای افزایش سرعت و حجم کم ساختار حلقه، برای متغیر های دستور حلقه از نام های یک حرفی ( i, j, k, m, n ) برای integer و c, d برای کاراکتر استفاده کنید.

```
for (int i = 0; i < nTables); i++){  
    .... }
```



## ۶) کلاس ها و اشیا:

- نام کلاس ها را با حروف بزرگ آغاز کنید.
- در نام هاي چند کلمه اي حرف اول تمام کلمات بزرگ باشد.
- نام کلاس باید ساده و توصيفي از عملکرد کلاس باشد.

class Birds

class StreamingVideo

- نام کلاس نباید زیرمجموعه اي از ویژگی هاي آن باشد.
  - قبل از نام متغيرهاي داخل کلاس از پیشوند m\_ استفاده کنید.
  - هنگامی که قصد دستیابی به یکی از متغيرها یا مشخصههاي کلاس را داشته باشیم باید از عبارات get/set استفاده شود.
- مثال:

employee.getName();

employee.setName(name);

matrix.getElement(2, 4);

matrix.setElement(2, 4, value);

## ٧) بسته ها (Packages):

- نام بسته ها را با mx يا com.macromedia آغاز كنيد. قسمت بعدي شامل نام رويه است. براي مثال:

mx.containers.ScrollPan

- اگر بسته، ساخته خود شماست نام آن را با com.macromedia آغاز كنيد:
- com.macromedia.Shapes

## ٨) کامپوننت ها (Components):

- نام کامپوننت ها با حروف بزرگ آغاز مي شود.
- در نام هاي چندکلمه اي حرف اول تمام کلمات بزرگ نوشته مي شود.

CheckBox

DataGrid

DataField

## Interfaces (۹)

- نام Interface ها با حروف بزرگ آغاز می شود.
  - در نام های چندکلمه ای حرف اول تمام کلمات بزرگ نوشته می شود.
  - نام Interface ها معمولاً بصورت صفت بیان می شود.
- Interface EmployeeRecords

## ۱۰) قواعد عمومي

- نامها بايد به زبان انگليسي باشد. از نوشتن به "PINGLISH" يا زبانهاي ديگر پرهيز شود.

- براي متغيرهاي ک محدوده گسترده‌اي دارند بايد نام بلند انتخاب شود.

- نامهاي که بيانگر مجموعه‌اي از اشياء مي باشند بايد بصورت جمع ظاهر شوند.

- براي نام هايي که نشاندهنده تعداد مي باشند بايد از پيشوند n استفاده نمود.

مثال:

nPoints, nLines

- براي متغيرهاي نشاندهنده عدد (شماره يک شيء) از پسوند No استفاده شود.

مثال:

tableNo, employeeNo

يک جاگزين براي اين روش استفاده پيشوند i مي باشد؛ مانند: iTable و iEmployee . اين روش

بخصوص در حلقه ها و براي متغيرهاي شمارنده مفيد است.

- براي نام متغيرهاي boolean بايد از پيشوندهاي مناسب مانند: is, has, can, should و ... استفاده

نمود.

مثال:

isSet, isVisible, isFinished, isFound, isOpen, hasLicense, canEvaluate, shouldSort

- از نام گذاري متغيرهاي boolean بصورت منفي خودداري شود.

مثال:

isError; // NOT: isNoError

isFound; // NOT: isNotFound

نام گذاريهاي منفي در هنگام اعمال عملگر منطقي "!", باعث کاهش خوانايي کد مي گردند.

## ب) Include

- در برنامه نویسی با فلش (اکشن اسکریپت) می توان کدهای خارجی را با اشتراک گذاشتی این کار با دستور `#include` و به این صورت انجام می شود:  
`#include "codes.as"`
- فایل کد خود را باید با پسوند `.as` ذخیره سازی کنید.
- تمام فایل های مطلوب را در ابتدای کد اصلی و در خطوط پشت سر هم `include` کنید.

## پ) انواع (Types)

- در یک کلاس ابتدا `public` سپس `private` می آید.
  - برای تعریف یک متد یا متغیر به شکل `public` یا `private` این کلمات را قبل از آن قرار دهید.
- ```
public function useClass(username,password){  
:  
}  
private var m_username,m_password
```

- تبدیل انواع باید بصورت صریح انجام شود (نه ضمنی)

## ت) متغیر ها

- متغیرها به هنگام اعلان باید مقدار دهی اولیه شوند. با این کار اطمینان حاصل می شود که متغیرها همواره معتبرند. گاهی اوقات مقدار دهی اولیه متغیرها در هنگام اعلان غیر ممکن می باشد که در این صورت بهتر است دستنخورده باقی بماند.
- متغیرها نباید به گونه ای باشند که چند معنا از آنها برداشت شود. این کار باعث می شود تا همه مفاهیم بصورت یکتا معرفی شده باشند که خود سبب خوانایی کد می گردد.
- استفاده از متغیرها، توابع سرتاسری (global) باید به کمترین حد خود برسد.
- متغیرهای خاص کلاس نباید به صورت public معرفی شوند.
- متغیرها باید در کوتاهترین دامنه دید تعریف گردند. این کار باعث می شود که تأثیرات جانبی متغیر در طول کد بهتر و ساده تر کنترل گردد.
- برای اینکه صفر بودن یا نبودن یک متغیر را بسنجیم نباید از صورت ضمنی عبارت شرطی استفاده کرد. مگر در مواقعی که متغیر از نوع boolean باشد.  
مثال:

```
if (nLines != 0) // NOT: if (nLines)
if (value != 0.0) // NOT: if (value)
```

## ث) کدگذاری روی سمبل ها و فریم ها

- کدهای برنامه های ساده را حتی الامکان روی سمبل ها بنویسید.
- در کد نوشتن روی button ها یا movie clip ها به روابط پدر-فرزندی و آدرس دهی صحیح دقت داشته باشید.
- در پروژه های نسبتاً طولانی کل کد خود را در یک فریم و روی خط زمان اصلی بنویسید. این کار سبب خوانایی بیشتر برنامه و تغییر ساده آن می شود.
- کد گذاری روی یک سمبل با این ساختار انجام می شود.

```
on (press) {  
:  
}
```

- کدگذاری یک سمبل روی فریم با این ساختار انجام می شود.

```
M1.onPress = function () {  
:  
}
```



### ج) حلقه ها

- تنها متغیرهای حلقه می توانند در داخل دستور for() استفاده گردند.

مثال:

```
sum = 0;           // NOT: for (i = 0, sum = 0; i < 100; i++)
for (i = 0; i < 100; i++) // sum += value[i];
    sum += value[i];
```

این کار سبب افزایش خوانایی و سادگی نگهداری و گسترش کد می گردد.

- متغیرهای حلقه باید دقیقاً قبل از حلقه مشخص شوند

مثال:

```
isDone = false;    // NOT: bool isDone = false;
while (!isDone) {  // :
    :              // while (!isDone) {
}                  // :
                  // }
```

- می توان از حلقه های do-while استفاده نکرد.

به سبب قرار داشتن شرط حلقه های do-while در انتهای آنها، خوانایی این حلقه ها از حلقه های while و for کمتر است. همچنین می توان هر حلقه do-while را به حلقه های while یا for تبدیل نمود. استفاده از تعداد ساختارهای کمتر در داخل برنامه سبب افزایش خوانایی آن می گردد.

- از استفاده از break و continue در داخل حلقه ها پرهیز شود.

تنها در صورتی استفاده از break و continue مجاز است که ثابت شود بکار بردن آنها باعث افزایش خوانایی برنامه خواهد شد.

- برای نشان دادن حلقه های بی پایان از حلقه while(true) استفاده شود.

مثال:

```
while (true) {
:
}
```

*for (;;) { // NO!*

*:*

*}*

*while (1) { // NO!*

*:*

*}*

## چ) دستورها و عبارات شرطی

- از استفاده از عبارات شرطی در هم و پیچیده خودداری شود در عوض می توان از متغیرهای موقتی و کمکی boolean یاری گرفت. با این کار کد برنامه، خود- مستند می گردد و همچنین خواندن، اشکالزدایی و نگهداری کد آسانتر می گردد.

- در دستورهایی if-else ، بخشی که انتظار اتفاق افتادنش وجود دارد (بخش مثبت) در قسمت if و بخش استثناء (منفی) شرط در قسمت else آورده می شود.

مثال:

```
if (isOk) {  
    ..  
}  
else {  
    :  
}
```

- دستورات شرطی باید در خطوط مجزا نگاشته شوند.

مثال:

```
if (isDone)      // NOT: if (isDone) doCleanup();  
doCleanup();
```

این کار در هنگام اشکال زدایی مفید واقع می شود چرا که اگر مقدم و تالی در یک خط واقع شده باشند، نمی توان تشخیص داد که شرط درست بوده یا نادرست.

- از آوردن دستورات و عبارات اجرایی در دستورات شرطی باید خودداری شود. خواندن دستورات شرطی شامل دستورات اجرایی مشکل می باشد؛ بویژه برای تازه کاران.

### ج) موارد دیگر:

- در برنامه نویسی با فلش نیازی به گذاشتن سمی کالن (;) در پایان دستورات نمی باشد و نوشتن هر دستور در خط مجزا کفایت می کند ولی برای رعایت استاندارد و خوانایی بیشتر از آن استفاده کنید.
- پنل Action فلش گزینه ای به نام Auto format وجود دارد. می توانید با زدن این دکمه کد خود را بصورت استاندارد در آورید. برای درست کردن استاندارد مطلوب خود از منوی preferences در پنل اکشن تب مربوط به Auto format را انتخاب کرده و style موردنظر خود را بسازید.
- استفاده صریح از اعداد و ارقام – به غیر از ارقام 0 و 1 -- در داخل کد مجاز نمی باشد و بجای آنها باید از ثابتهایی با نام معنادار بهره گرفت.  
اگر عددی به خودی خود معنایی نداشته باشد، استفاده از ثابتی با نام با معنادار می تواند سبب خوانایی کد گردد.
- اعداد از نوع اعشاری باید شامل حداقل یک رقم صحیح و ممیز (نقطه اعشاری) باشند.  
با این روش متغیرهای از نوع اعشاری از متغیرهای صحیح باز شناخته می شوند.
- در اعداد اعشاری حتماً باید دست کم یک رقم صحیح پیش از ممیز وجود داشته باشد.  
مثال:

`double total = 0.5; // NOT: double total = .5;`

## خ) بلاک ها

- مقدار پایه فرو رفتگی برابر دو ستون می باشد.

مثال:

```
for (i = 0; i < nElements; i++)  
a[i] = 0;
```

مقدار کمتر از دو ستون سبب ناتوانی در شناسایی بلاکها می شود. همچنین مقدار بیشتر از چهار ستون نیز باعث دشواری خواندن و همچنین افزایش احتمال گسسته شدن خطوط در بلاکهای تودرتو می گردد. بهترین مقادیر برای انتخاب دو، سه یا چهار ستون می باشد.

- نحوه نوشتن بلاکها باید مطابق دو نمونه زیر باشد. (که نمونه اولی بهتر می باشد)

```
while (!done) {  
doSomething();  
done = moreToDo();  
}
```

یا

```
while (!done) {  
doSomething();  
done = moreToDo();  
}
```

همچنین یک بلاک نباید به گونه زیر نوشته شود:

```
while (!done)  
{  
doSomething();  
done = moreToDo();  
}
```

- نحوه نوشتن بلاک کلاس باید به شکل زیر باشد:

```
class SomeClass {  
:
```

```
}
```

- نحوه نوشتن بلاک توابع باید به شکل زیر باشد:

```
void someMethod(){  
:  
}
```

- نحوه نوشتن بلاک دستور if-else باید به شکل زیر باشد:

```
if (condition) {  
    statements;  
}
```

```
if (condition) {  
    statements;  
} else {  
    statements;  
}
```

```
if (condition) {  
    statements;  
} else if (condition) {  
    statements;  
} else {  
    statements;  
}
```

- نحوه نوشتن بلاک دستور for باید به شکل زیر باشد:

```
for (initialization; condition; update) {  
    statements;  
}
```

- نحوه نوشتن دستور for تهی، باید به شکل زیر باشد:

```
for (initialization; condition; update)
;
```

- نحوه نوشتن بلاک دستور while باید به شکل زیر باشد:

```
while (condition) {
    statements;
}
```

- نحوه نوشتن بلاک دستور do-while باید به شکل زیر باشد:

```
do {
    statements;
} while (condition);
```

- نحوه نوشتن بلاک دستور switch باید به شکل زیر باشد:

```
switch (condition) {
    case ABC :
        statements;
        break;

    case DEF :
        statements;
        break;

    default :
        statements;
        break;
}
```

- ساختارهای تک دستوری if-else، while یا for می توانند بدون آکولاد، "{}"، نوشته شوند.

مثال:

```
if (condition)  
    statement;
```

```
while (condition)  
    statement;
```

```
for (initialization; condition; update)  
    statement;
```



## د) فضاهای خالی

- پیش و پس از عملگرها باید یک فضای خالی باشد.
- پس از واژگان پس‌نهاد (reserved words) C++ باید یک فاصله قرار گیرد.
- پس از هر کاما باید یک فضای خالی قرار گیرد.
- پیش و پس از علامت دو نقطه، ":", باید یک فاصله قرار گیرد.
- پس از علامت ";" باید یک فضای خالی قرار گیرد.

مثال:

```
a = (b + c) * d; // NOT: a=(b+c)*d
```

```
while (true) // NOT: while(true)
```

```
doSomething(a, b, c, d); // NOT: doSomething(a,b,c,d);
```

```
case 100 : // NOT: case 100:
```

```
for (i = 0; i < 10; i++) { // NOT: for(i=0;i<10;i++){
```

...

این روش با برجسته ساختن مؤلفه های منفرد دستورات، سبب خواناتر شدن کد می گردد.

- اگر پس از نام توابع، یک نام واقع شود، می توان بعد از نام تابع یک فاصله خالی قرار داد.

مثال:

```
doSomething (currentFile);
```

این کار با واضح و مشخص ساختن هر نام، خوانایی کد را موجب می شود. اگر هیچ نامی پس از نام تابع نیامده باشد می توان فاصله خالی را حذف نمود.

```
doSomething()
```

راه دیگر این است که فاصله را پس از پرانتز باز قرار داد که بعضی از برنامه نویسان برای وضوح و همچنین تقارن بیشتر، پیش از پرانتز بسته نیز یک فاصله خالی قرار می دهند.

```
doSomething( currentFile );
```

- درون یک بلاک را باید از نظر منطقی گروه بندی کرد و پس از هر گروه یک خط خالی قرار داد.  
مثال:

```
matrix = new Matrix4x4();

cosAngle = Math.cos(angle);
sinAngle = Math.sin(angle);

matrix.setElement(1, 1, cosAngle);
matrix.setElement(1, 2, sinAngle);
matrix.setElement(2, 1, -sinAngle);
matrix.setElement(2, 2, cosAngle);

multiply(matrix);
```

- متغیرها به هنگام اعلان باید چپ-تراز گردند.

- تراز کردن در هر جایی که موجب افزایش خوانایی گردد توصیه می شود.  
مثال:

```
if (a == lowValue) computeSomething();
else if (a == mediumValue) computeSomethingElse();
else if (a == highValue) computeSomethingElseYet();

value = (potential * oilDensity) / constant1 +
        (depth * waterDensity) / constant2 +
        (zCoordinateValue * gasDensity) / constant3;

minPosition = computeDistance(min, x, y, z);
averagePosition = computeDistance(average, x, y, z);

switch (value) {
```

```
case PHASE_OIL : strcpy(phase, "Oil"); break;  
case PHASE_WATER : strcpy(phase, "Water"); break;  
case PHASE_GAS : strcpy(phase, "Gas"); break;  
}
```

در بسياري از موارد، فضاهاي خالي مي تواند سبب افزايش خوانايي كد گردد حتي اگر قواعد گفته شده را نقص نمايد. بايد توجه كرد كه در صورت استفاده از فضاهاي خالي بايد تراز منديها رعايت شود.

## (ذ) توضیحات (Comments)

- کد باید خود-مستند باشد.
- در حالت کلی کد باید بقدری خوانا و قابل فهم باشد که نیازی به استفاده از توضیحات نداشته باشد. با نام گذاریهایی شایسته و رعایت ساختارهای مناسب این خواسته تحقق می یابد.
- تمامی توضیحات باید به زبان انگلیسی نگاشته گردند.
- برای اینکه در ساختار منطقی کد خلل وارد نشود، توضیحات درون بلاک باید مطابق با تورفتگیها نوشته شوند.

**گردآوری و ترجمه:**

امین کریمی

[www.amindesign.ir](http://www.amindesign.ir)