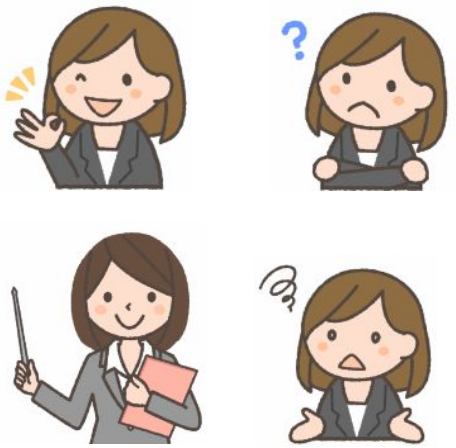
ななちゃんのIT教室

かっこいい生意気JavaScriptの巻

by nara.yasuhiro@gmail.com

ななちゃんが
禁断のプロテクニックを覚えるという お話

第 0.9 版　2017 年 7 月 11 日



フリー素材
http://freeillustration.net

もくじ

第1回　かっこいい生意気テクニックとは

なな：友達にえばれるような、プロのテクニックを覚えたいなあ。

先生：JavaScript の記述を短く、コンパクトにするテクニックはあるわ。でも、他の初心者には読みづらいプログラムになるので、使いすぎないようにね。たとえば、こんなのがあります。

```
var x = 1.5;
alert(Math.floor(x));
```

→

```
var x = 1.5;
alert(~~x);
```

これは、**切り捨てによる整数化**の例なの。「~」は、「整数部分を取り出して（小数以下切り捨て）、各ビットの 1/0 を反転する」という演算子なんだけど、それをふたつ重ねると、2 回反転するから、元に戻るわけ。結局、切り捨てによる整数化になるというわけ。

なな：ひゃあ、かっこいい！

先生：こういうのを機会に、「~」演算子を覚えたり、プログラムを楽しむのは良いかもね。

なな：「むむ、お主、できるな！」という感じね。**四捨五入**はできないの？

先生：こんな感じになります。0.5 を足してから切り捨てということ。

```
var x = 1.5;
alert(~~(x+0.5));
```

なな：もっと教えて！

先生：じゃあ、これなんかどうかしら。

```
var x, y;
x = 0;
y = 0;
```

→

```
var x, y;
x = y = 0;
```

◎

```
var x=0, y=0;
```

→

```
var x = y = 0;
```

△

なな：こういう書き方ができるの？

先生：実は、特別なことではないのよ。「y = 0」という「式」の「値」が「0」になるということによるの。つまり、計算順序を考えると、「x = (y = 0);」ということで、y に 0 を記憶させる「(y = 0)」自体の値が「0」だから、x にとっては、x = 0; と同じことになるわけね。

なな：りょうかい！

```
var x=1, y=2;
function fn() {
    var x = y = 0;
}
fn();
x;        // Number number 1
y;        // Number number 0
```

要注意！
y はグローバルになってしまいます。
var x, y;
x = y = 0;
または
var x, y = x = 0;
と書きましょう。

第2回　かっこいい生意気テクニック（続）

なな：かっこいいのをもっと教えて！　ワクワク！

先生：こんなのもあるわよ。

```
var x, y=4;
for (x=1; x<5; x++) { alert(y); y--; }
```

→

```
var x, y=4;
for (x=1; x<5; x++) alert(y--);
```

これは、4、3、2、1 という順にアラートが出るプログラム。ポイントは**「y--」ね。これは、y の値を 1 減らす命令だけど、全体としての値は、減らす前の y のになる**のよ。たとえば、y の値が 4 だった時、「y--」を実行すると、y の値は 3 になるけど、「y--」全体の値は 4 になるの。

なな：ちょっと難しいかな。

先生：**「--y」は、y の値が 1 減るけど、全体の値は、1 減った後の値になります**。y の値が 4 だったら、「--y」を実行すると、y の値は 3 になり、「--y」全体の値も 3 になります。

なな：そう言えば、x の値を 1 増やすのに、「x++」と「++x」のふたつの書き方があると聞いていたけど、これで違いが分かったわ！

先生：こういう、生意気テクニックを使うことで、JavaScript の理解が深まるという効果もあるわね。

なな：「{ alert(y); y--; }」は「{」、「}」がついているけど、「alert(y--);」にはついていないのね。

先生：**for や if の中身（実行文）がふたつ以上ある場合は、「 { 」、「 } 」で囲まないといけないけど、 中身がひとつの場合は、省略して良い**という文法規則があるの。初級者は、よく、中身がふたつ以上あるのに、「 { 」、「 } 」をつけ忘れて混乱するので、初級者には、「ひとつでも、二つ以上でも、必ず { } をつけましょう」と指導することが多いのね。

なな：だから、中身がひとつの時に、{ } を省略すると、「私は中級者」みたいでかっこいいのね！

先生：まあ、そういうことね。さっきの説明だけど、正確には、for、while、if、else なんかは、後ろに文をひとつだけ書けるという文法になっていて、「{ 文; 文; 文; }」のように、{ と } でくくると、「全体として ひとつの文として扱う」、、「{ 文; 文; 文; }」は、ひとつの文である、ということなの。

なな：ふうん。

第3回　かっこいい生意気テクニック（続々）

なな：もっともっと教えて！　ワクワク！

先生：じゃあ、こんなのは？

```
var x, y=4;
for (x=1; x<5; x++) { alert(y); y--; }
```

→

```
var x, y;
for (x=1,y=4; x<5; x++,y--) alert(y);
```

なな：え～～～！　for 文に、こんな書き方があるの？

先生：for 文の書き方というのと、ちょっと違うのよ。for 文は、「 for (文, 真理値式, 文) 文;」の形をしているんだけど、「x=1,y=4」、も、「x++,y--」も、それぞれ、ひとつの文の扱いになるのよ。そのポイントは、**「 , 」**にあるの。これは、「**コンマ演算子**」なの。「式1, 式2」は、「式1 を実行して、その結果の値は捨てて、次に、式2 を実行して、その値を全体の値とする」という意味になるの。だから、「1, 2」の値は「2」になるのね。しかも、「『文』は『式』の一種」ということになっているの。「 for (文, 真理値式, 文) ;」の部分の、ふたつの「文」は、結果としての「値」は利用しないので、、**「x=1,y=4」は**、「x=1」という文と、「y=4」という文を、**コンマ演算子でつないだ、全体としてひとつの式（文）**、ということなの。同じように、「x++,y--」は、「x++」という文と、「y--」という文を、コンマ演算子でつないだ、全体としてひとつの式（文）、ということなの。

なな：なんか、へりくつみたいだけど、「for (x=1,y=4; x<5; x++,y--)」は、「 for (文, 真理値式, 文)」の形になっているということなのね。

先生：それから、次に、これとは別の見方なんだけど、「var x=1,y=4;」も、全体として、ひとつの文なのね。だから、次のようなこともできるのよ。

```
var x, y=4;
for (x=1; x<5; x++) { alert(y); y--; }
```

→

```
for (var x=1,y=4; x<5; x++,y--) alert(y);
```

第4回 かっこいい生意気テクニック（続々々）

なな：もっともっともっと教えて！ ワクワク！

先生：今回のは、これ。

```
var x = 5;
while (x >= 0) { alert(x); x--; }
```

→

```
var x = 6;
while (x--) alert(x);
```

どちらも、アラートが、「5、4、3、2、1、0」の順に出るのよ。

なな：右のは、絶対におかしいわ。while の（）の中は、true/false になる「論理式」にしないといけないんでしょ？

先生：実は、JavaScript では、**while なんかの「条件」**のところは、「**false、""、0、undefined、null 以外の値を true とみなす**」ということになっているのよ。数字なら、「6、5、4、3、2、1」はすべて「true」扱いで、「 0 」だけ「false」扱いになるの。右のプログラムでは、alert は、「6、5、4、3、2、1」の各々をマイナス1 した「5、4、3、2、1、0」を表示するのね。0 をマイナス1 した「-1」は表示しないで終わるということなの。

なな：ふうん。「 0 」で終わるということだと、「0、1、2、3、4、5」みたいに表示する時には使えないということ？

先生：そこが、工夫しだい。

```
while (x--) alert(5-x);
```

のようにすれば良いのよ。

なな：むむ。お主、なかなかやるのう！

第5回　かっこいい生意気テクニック（続々々々）

先生：今回のプログラムは、これ。

```
for (x=1; x<3; x++) {
    for (y=1; y<3; y++) {
        alert(x*y);
    }
}
```

→

```
for (x=1; x<3; x++) {
    for (y=1; y<3; y++) {
        alert(x*y);
} }
```

①　②

なな：1 行減っているわね。

先生：これは、ちょっとした視覚的トリックになっているの。右のプログラムの、①の } は、「for (y=1; y<3; y++) {」に対応していて、②の } は、「for (x=1; x<3; x++)」に対応しているのね。でも、見かけ上、反対に対応しているようになっているの。} の数は 2 つで合っているし、ちゃんと動作するし、過不足チェックもしやすいので問題ないということ。

なな：だまされているみたい。

先生：でも、正しく動くプログラムだし、見やすいし、行数が少ないということ。
二重の for だけでなく、for の中に if があるとか、関数定義の最後に for がある時なのに利用できるテクニックです。

第6回 かっこいい生意気テクニック(続々々々々)

先生: もとになるプログラムは、これ。入門者向けの、合計を計算するプログラムです。1+3+5 で、9 のアラート。

```
var d = [ 1, 3, 5 ], sum=0;
for (var i=0; i<3; i++) sum += d [ i ] ;
alert(sum);
```

な な: めんどくさいやつね！

先生: この例では、配列に入っているすべてのデータの合計を計算するけど、「i=0; i<3; i++」の「 0 」や「 3 」の部分を書き換えると、データの一部分だけの合計を求めることもできるという点で、「万能」なので、入門者に覚えてもらいたいということで教えるわけね。でもね、実際には、「**すべてのデータ**を」という場合が圧倒的に多いのよね。 実は、**そういう場合は、便利な書き方、かっこいい書き方がある**のよ。

な な: じらさないで、早く教えて！

先生: まず、「i=0; i<3; i++」の「 3 」の部分だけど、データの数が多い時に、データ個数をプログラマが数えるのが面倒だし、プログラム実行中にデータ個数が増減する場合に面倒なのね。 そこで、「配列に現在入っているデータの個数」を数える「配列名.length」というのを教えるわね。

→

```
var d = [ 1, 3, 5 ], sum=0;
for (var i=0; i<d.length; i++) sum += d [ i ]
alert(sum);
```

な な: これ、聞いたことあるわ。

先生: 聞いたことがあるとは残念。じゃあ、これは？

→

```
var d = [ 1, 3, 5 ], sum=0;
for (var i in d) sum += d [ i ] ;
alert(sum);
```

な な: 聞いたころがあるかも。

先生: むむ。 でも、これは知らないはず！ どうだ！

→

```
var d = [ 1, 3, 5 ], sum=0;
for (var x of d) sum += x;
alert(sum);
```

な な: これは知らな～～い。

先生: おっほん。 これは、最新の JavaScript、つまり、EcmaScript 2015 という規格で使えるようになった機能なの。

な な: 「2015」？ 今は 2017 年なので、ちょっと古い？

先生: 規格が発表されたのは 2015年だけど、ブラウザで使えるようになったのは最近なの。Firefox や、Google Chrome では、結構前から「前倒し実装」されていたけど、
Internet Explorer 11 は未対応で、Windows 10 の、
Microsoft Edge から使えるようになったのよ。

な な: それなら、知らない人も多そうだから、いばれるかも。

第7回　かっこいい生意気テクニック（これでおしまい）

先生：かっこいいテクニックも、今回で最終回。

なな：ちょっとさみしいような。

先生：今回のプログラムは、これ。日曜日を 0、月曜日を 1、…、土曜日を 6 としてコード化しているデータを元に戻して表示するものです。「x = 3」は、テスト用データと考えてください。3 を元に戻して「水曜日」と表示します。データは、0 ～ 6 の値しかとらないという前提です。

```
var x = 3;
if      (x == 0) alert("日曜日");
else if (x == 1) alert("月曜日");
else if (x == 2) alert("火曜日");
else if (x == 3) alert("水曜日");
else if (x == 4) alert("木曜日");
else if (x == 5) alert("金曜日");
else             alert("土曜日");
```

→

```
var x = 3;
switch (x) {
    case 0: alert("日曜日"); break;
    case 1: alert("月曜日"); break;
    case 2: alert("火曜日"); break;
    case 3: alert("水曜日"); break;
    case 4: alert("木曜日"); break;
    case 5: alert("金曜日"); break;
    default: alert("土曜日");
}
```

なな：構造は単純になったけお、行数は かえって増えてしまっているような。

先生：かっこいいテクニックはこれからよ。

→

```
var x = 3;
var table = [ "日", "月", "火", "水", "木", "金", "土" ];
alert ( table [ x ] + "曜日" );
```

なな：配列を使うのね！

先生：最終回ということで、生意気というより、正統派のテクニックを紹介しました。

第8回　かっこいい生意気テクニック（まとめ）

先生：生意気テクニックは、プログラマの、ちょっとした「遊び心」です。プロフェッショナル同士のコミュニケーション、「にやっとして分かり合う」というようなものです。プログラミングを楽しむというニュアンスで使ってください。入門者をうならせて、優越感を味わう、自分のプログラミング技術の成長を実感するというものでもあります。

また、こういうことを通じて、JavaScript の文法を、より深く理解するきっかけになるという面もあります。

しかし、「（入門者に）読みにくいプログラム」、「人によっては、分からなくて不愉快になる」という面もあるので、使い過ぎには注意してください。 とくに、まじめな、JavaScript の試験答案で使うのはやめましょう。「分かってなくて、思いつき、当てずっぽうで回答している」と誤解されて減点されるかも。 こういう「生意気テクニック」に不慣れな先生、不快感を感じる先生もいるかも。