ななちゃんのIT教室

デバッグ奥義の巻

by nara.yasuhiro@gmail.com

ななちゃんが
デバッグ奥義を教えてもらうという お話

第 1.0 版　2017 年 5 月 7 日

フリー素材
http://freeillustration.net

もくじ

第1回　デバッグとは

なな：これって、朝日新聞の「ののちゃんのDO科学」のパクリ？

先生：パロディって言ってちょうだい。家政婦のミタ（「家政婦は見た」のパロディ）、クレヨンしんちゃんのダズニーランド（「ディズニーランド」のパロディ）みたいなものよ。

なな：「デバッグ」ということばを聞いたけど、どういう意味なの？

先生：プログラムの間違いを見つけて、修正することよ。庭木の害虫（バグ）を見つけて駆除するというイメージね。**すぐれたプログラマは、間違いをしない人ではなくて、間違いを見つけることに慣れている人**ということなの。

なな：プロフェッショナルのプログラマは、一発で正しいプログラムを書けるんじゃないの？

先生：そんなことはないのよ。タイピングミスをしょっちゅうするわ。たとえば、「;」のかわりに「:」を入力してしまうこともよくあるわよ。なんしろ、キーボードでおとなりのキーだしね。

なな：なるほど。プログラムの知識と、タイピングミスは別物だからね。

先生：それに、だいたいプログラムを作ってから、仲間に使ってもらって、問題を発見することもよくやるのよ。「想定外の使い方」があることをみつけたり。

なな：そういえば、いろんなソフトで、「バージョンアップ」というのがよくあるわね。

先生：そうそう。バージョンアップは、機能追加だけではなくて、バグ部分を修正するという意味も大きいの。それから、プログラマは、みな、自分なりに失敗を重ねながら技術を身につけてきているの。プログラマの後輩指導では、後輩に同じ苦労をさせないように、「苦労から発見した要点は、ようするに〇〇だ」と説明するんだけど、後輩は、すぐに忘れてしまう。覚えられない。なぜだと思う？

なな：頭が悪いから？

先生：いいえ。それは、人間は「失敗を重ねることで覚える」からなの。言葉で表現された記憶を「意味記憶」といい、失敗を通じて、場面と一緒に覚えている記憶を「エピソード記憶」というの。両者は脳における記憶メカニズムが異なっていて、後者のほうが、圧倒的に忘れにくいことが脳科学の研究で分かっているの。ななちゃん、資料を読んで、それを覚えるのではなく、プログラムを作ってみて、なかなか動作しないで、苦労してデバッグしながら、「失敗を重ねる」ことで覚えてね。

「AはBである」ではなく、「AはCであると勘違いしてひどい目にあった。Bだったんだ！」という形で心にとどめて欲しいのね。

なな：ラジャー！

第2回　デバッグのやりかた

なな：デバッグは、どういうふうにやれば良いの？

先生：デバッグには、2つのステップがあるのよ。

なな：ひとつ目は？

先生：「文法エラー」の修正ね。英文の誤文訂正のたとえで説明するわね。「A cow fly.」という文には文法間違いが含まれているわね？

なな：三単現（三人称・単数・現在）の「s」が抜けているわ。「A cow **flies**.」が正解ね。

先生：そうそう。プログラムの場合は、「ぜんぜん動かない」とか、「エラーメッセージが出る」という症状になるの。そういう間違いをみつけて、プログラムを修正して、動かす、エラーメッセージが出ないようにする必要があるのね。これには、コンピュータのエラー検出機能が使えることが多いの。

なな：じゃあ、ふたつ目は？

先生：「意味上のエラー」の修正ね。「A cow flies.」は、文法的には正しいけど、「牛は空を飛ぶ」はずはないので、意味としておかしいわね。

なな：「牛が飛行機に乗っている」とか、「メルヘン」だとか？

先生：こらこら。たとえ話だと言ったでしょう。たとえば、郵便番号を入力すると、その地域の住所を表示するようなプログラムで、見かけ上動くけど、東京の郵便番号を入力しても、沖縄の地名が表示されるようなエラーね。これは、コンピュータが発見するのではなく、人間ががんばらないといけないの。まあ、入力と出力のデータの正しい組み合わせを前もって用意しておいて、その通りの出力になるか、自動チェックすることもできるけどね。

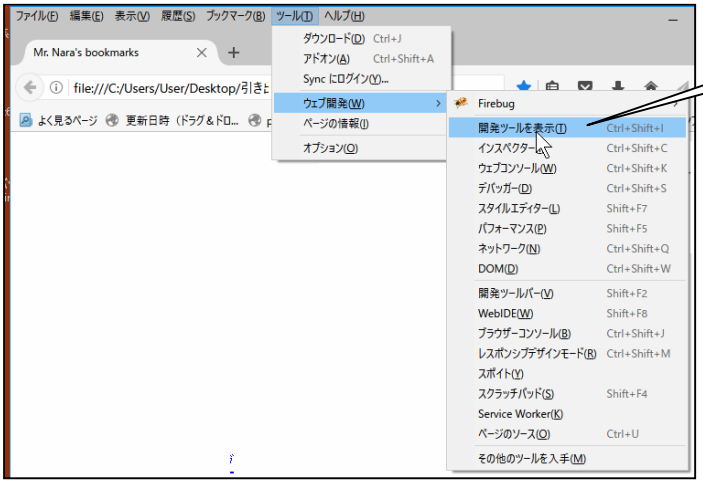
なな：そうか。

先生：それ以外にも、不適切な入力データが入力された時に、プログラムが暴走しないで、ちゃんと再入力を促すかといった、使い勝手のテストも必要な場合もあるわ。高額な商品としてのプログラムなんかだと、使いにくかったら、買ってもらえないからね。

第3回　デバッグのやりかた：実践編1（デバッガ）

なな：デバッグは、具体的には、どういうふうにやれば良いの？

先生：まずは、「デバッガ」というプログラムの使い方を説明するわね。ここでは、Firefox ブラウザに組み込まれているものを使って説明します。他のブラウザでも、使い方が少し違うけど、機能的には同じようなものがあります。Firefox で、「ツール」→「ウェブ開発」→「開発ツールを表示」というメニュー操作をして、デバッガの画面を表示せます。画面の下のほうに出したり、右のほうに出したり、別ウィンドウで出したりを切り替えできます。



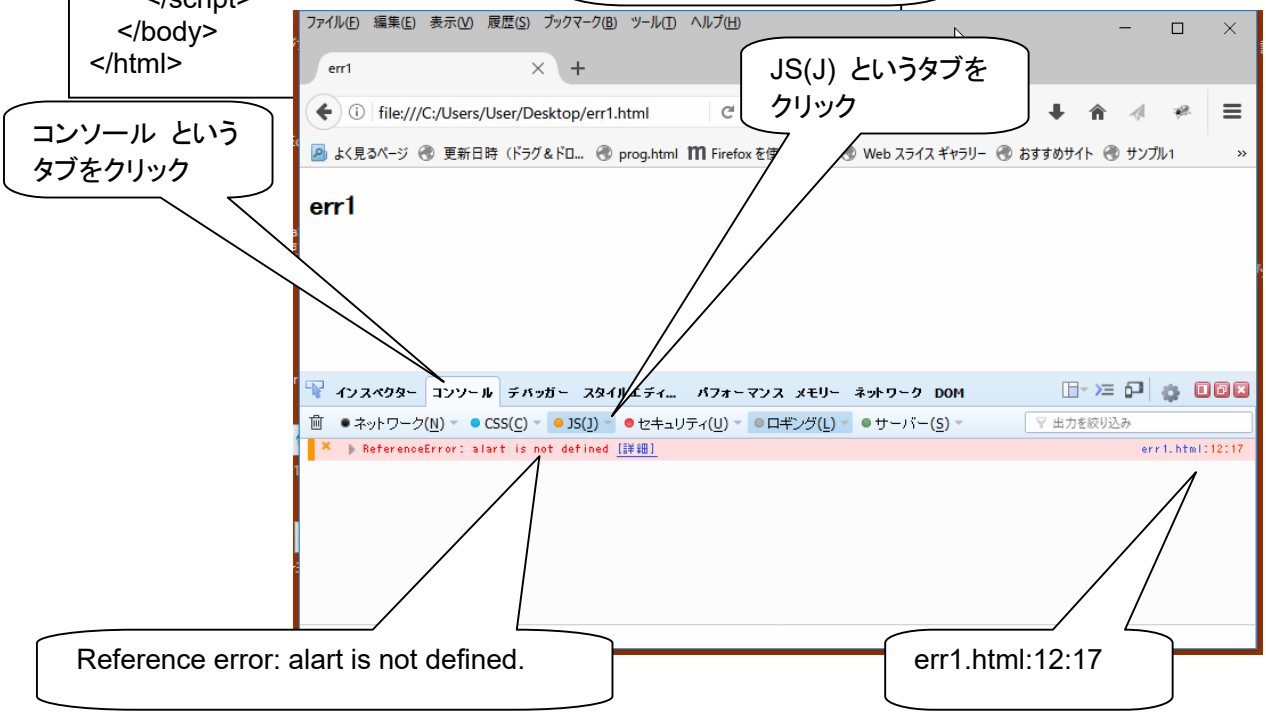
次に、そのブラウザで、プログラムを実行します。ここでは、わざとエラーを入れた、下記のプログラムを実行してみましょう。

err1.html

```
<!DOCTYPE html>
<html lang="ja">
  <head>
    <meta charset="utf-8">
    <title>err1</title>
  </head>
  <body>
    <h2>err1</h2>
    <script>
  var x;
  for(x=1; x<4; x++)
     if (x = 2) alart(x);
    </script>
  </body>
</html>
```

alert を alart と
ミスタイプしている

x == 2 を
x = 2 にミスタイプしている

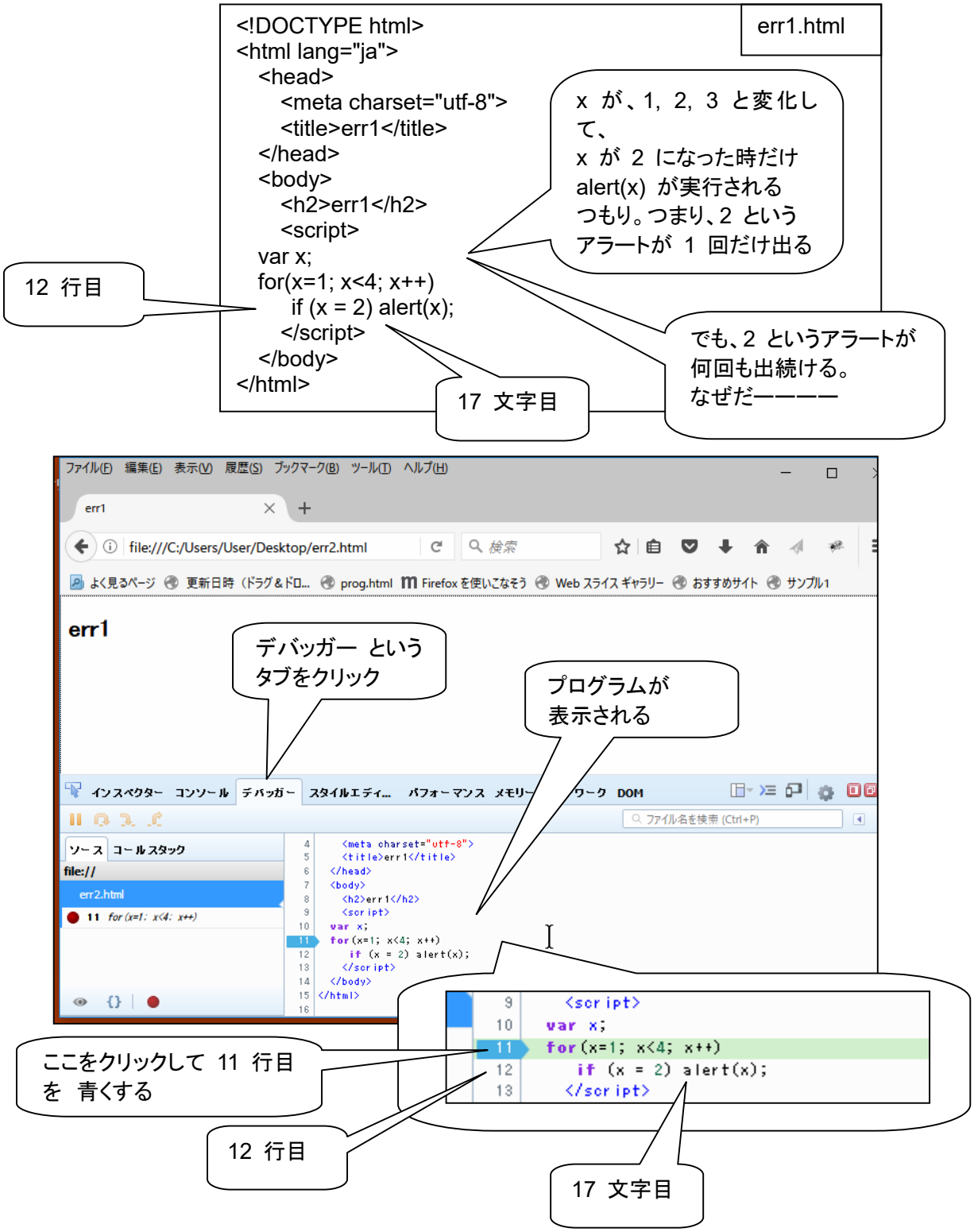


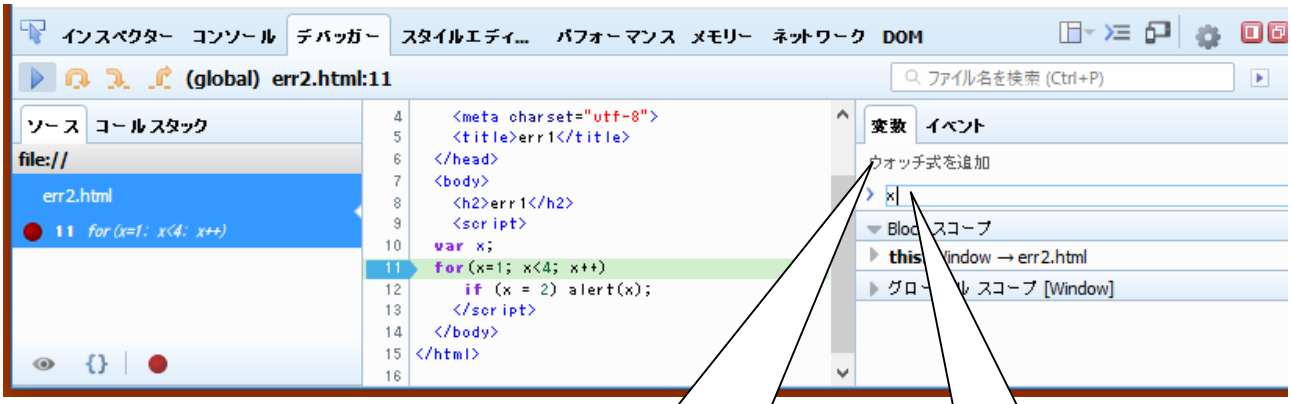
なな：エラーメッセージが出たけど、英語だあ。

先生：残念ながらそう。でも、難しい英文ではないから、すぐに慣れるわ。
「Reference error: alart is not defined.」は、「参照エラー： alart は定義されていません」という意味。これは、「alert」と書くべきだったのを「alart」と書き間違えたからなの。計算機は「alert と書くべきだった」とは判断できないので、「alart というのは知らない命令だよ」という形で警告しているわけ。画面の右のほうに「err1.html:12:17」と書いてあるけど、これは、「error1.html」ファイルの 12 行目、17 文字目。

なな：いわゆる「文法エラー」ということ？

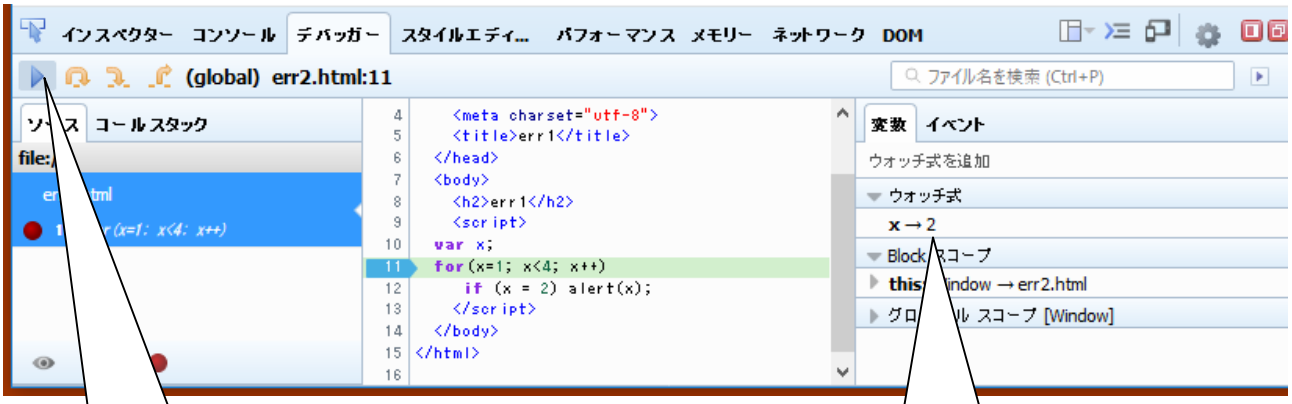
先生：そうね。次に、ファイルの「alart」を、「alert」に書き直して、デバッグの次の段階にすすみましょう。





ウォッチ式を追加　をクリック

ここに「x」と入力



ボタンをク ▶ して、
プログラムを実行する

x → 2 と表示されたまま

if は、false、""、0、
undefined、null 以外の
値を true とみなす

x = 2 全体の「値」は 2

if (2) は if (true) と同じ

if (0) は if (false) と同じ

だから alert(x) はいつも
実行されてしまう

```
var x;
for(x=1; x<4; x++)
    if (x = 2) alert(x);
```

x == 2 （ x が 2 に等しい ）
を
x = 2 （ x に 2 を記憶する ）
に書き間違えている！！！

だから、x が 2 のままで
増えない！！！

だから、「2」という
アラームが
何回も 出続ける！！！！

なな：ということで、「x == 2」を「x = 2」と書き間違えたという「意味上のエラー」を発見したということね！

第4回 デバッグのやりかた:実践編2(デバッグ奥儀)

デバッグは、海外旅行好きの人から聞いた話に似ている点がある。その人は、海外旅行が好きだから、旅行の申し込みをする。しかし、出発当日、成田空港にゆくと、憂鬱な気分になるという。言葉も通じないだろうし、自由も制限される。旅行に行くのをやめて、ここから引き返したくなるという。ひとこと、旅行会社に電話すればそれで済むこと。でも、なんとなく、がまんして、飛行機に乗る。しかし、いったん、旅行に出発していまえば、興奮の連続。帰ってきてから、「旅行に行ってよかった」 とつくづく思い、また、海外旅行に行きたいと思うという。

デバッグも、最初は気が重い。会社の上司に「プログラムは完成しませんでした」、と謝ってしまったらどんなに楽かと思う。しかし、はじめてしまうと、興奮の連続。プログラムが動くと、この上ないような喜びを感じる。また、デバッグしてみようかなと思ったりする。

先生から教えてもらったこと、本で読んだことは、しばらくすると、簡単に忘れてしまう。でも、**デバッグで、自分で見つけて、直したことのあるバグは、そうは簡単に忘れない**。心理学的にも証明されている。場面、情景を伴って覚えたことを「エピソード記憶」と言うが、簡単には忘れない。エストニア生まれのカナダ人心理学者エンデル・タルヴィング(Endel Tulving)が1972年に発表した。みなさんには、デバッグに励んでもらいたい。どんどんミスをして、どんどんバグを見つけて、直すことでプログラミングの達人になって欲しい。

デバッグ奥義を贈ろう。

```
---------- prog.html ----------
<!DOCTYPE html>
<html>
  <head>
    <script src="prog.js"></script>
  </head>
  <body>
      <button onclick="go()">実行</button>
  </body>
</html>
-------------------------------------------------
---------- prog.js ----------
function go() {
  alert("Hello!");
}
-------------------------------------------------
```

こんな簡単なプログラムでも「ReferenceError: go is not defined」というようなエラーになることがある。もちろん、Web Console などを使わないと、「何も反応しない」「動かない」だけであるが。

「go is not defined」のようなエラーの場合、私なら

```
---------- prog.js ----------
alert("!!!);
//function go() {
//   alert("Hello!");
//}
-------------------------------------------------
```

のように、1行追加する。これで、「!!!」を表示するアラートが出なければ、prog.js のファイル名が違っている(全角文字だったりする)か、prog.js が、エディタで、まだ保存していないとか、prog.js が prog.html と異なるフォルダに作ってあったりする。「// ～」 の行はコメント扱い。コンピュータは無視する。実行対象にならない。

「!!!」を表示するアラートが出れば、今度は、

```
---------- prog.js ----------
function go() { alert("!!!); }
//    alert("Hello!");
//}
-----------------------------------------------
```

に変えて、「!!!」を表示するアラートが出ることを確認する。

```
---------- prog.js ----------
function go() {
   alert("Hello!"); alert("!!!); }
//}
-----------------------------------------------
```

のように、「//」を消して「alert("!!!); }」を追加する行をひとつずつ下に移動してゆく。はじめてエラーが出たところで、その行をチェックする。全角の空白が入っていたりする。半角の空白やタブと、全角の空白は、画面で見た上では区別できない。全角の空白は、JavaScript としては使ってはならない。行末についていてもダメ。例外は、コメントの中と、「"～"」の、文字列の部分だけです。こんな感じで、間違いを見つけてください。

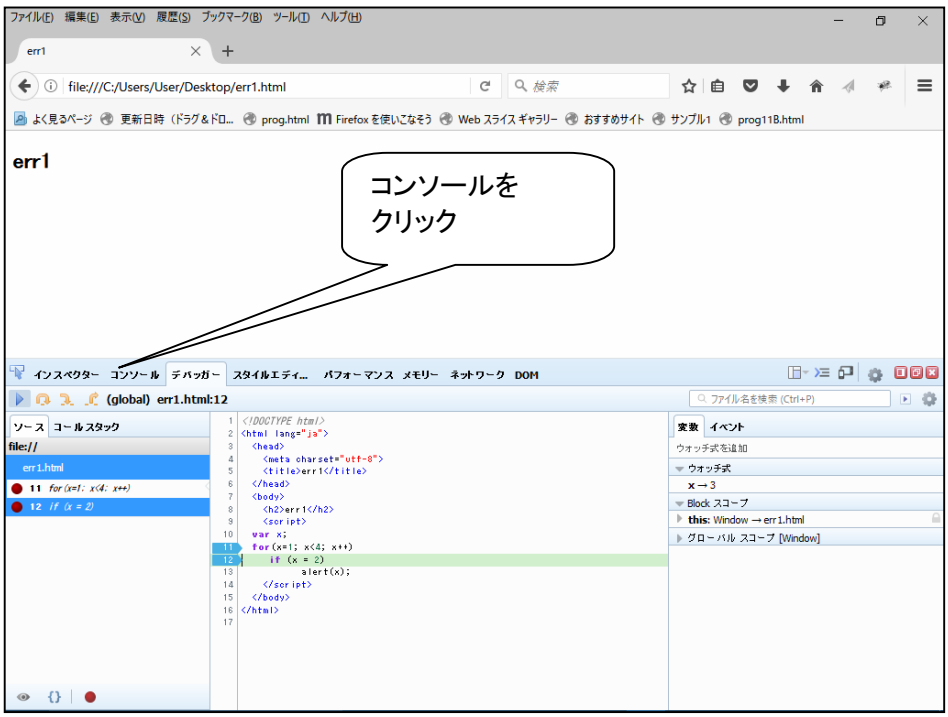
さいごに、プレゼント。答えは掲載しないので、自分でバグを見つけてください。

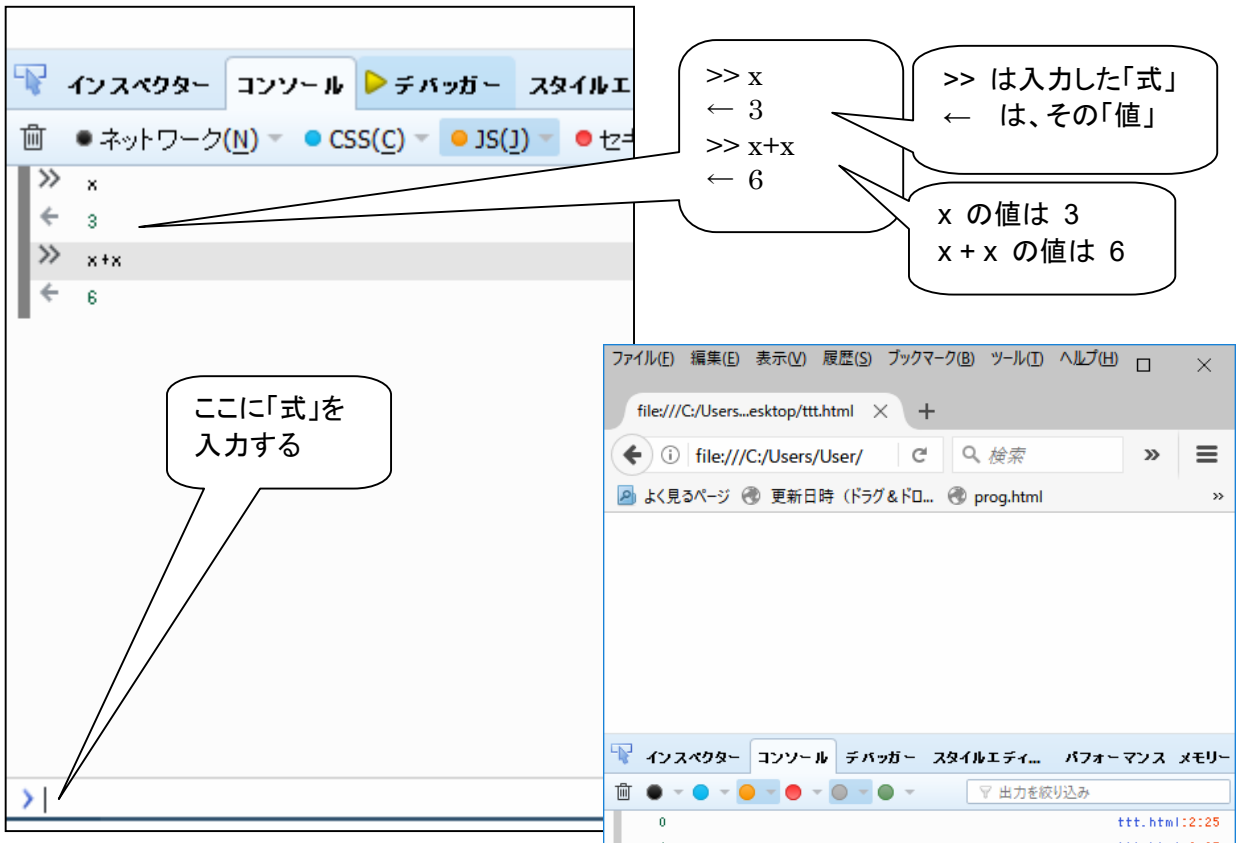
```
---------- prog.js ----------
function go() {
   var x, y;
   for(x=1; x<4; x++)
        y = x;
        alert(y);
}
-----------------------------------------------
```

1, 2, 3 のアラートが出るはずなのが、3 のアラートしか出ないのはなぜでしょう？

第5回 デバッグのやりかた：実践編3（コンソール）

デバッグ中、「コンソール」タグをクリックすると、JavaScript の「式」を入力して、その「値」を確認することができるようになります。





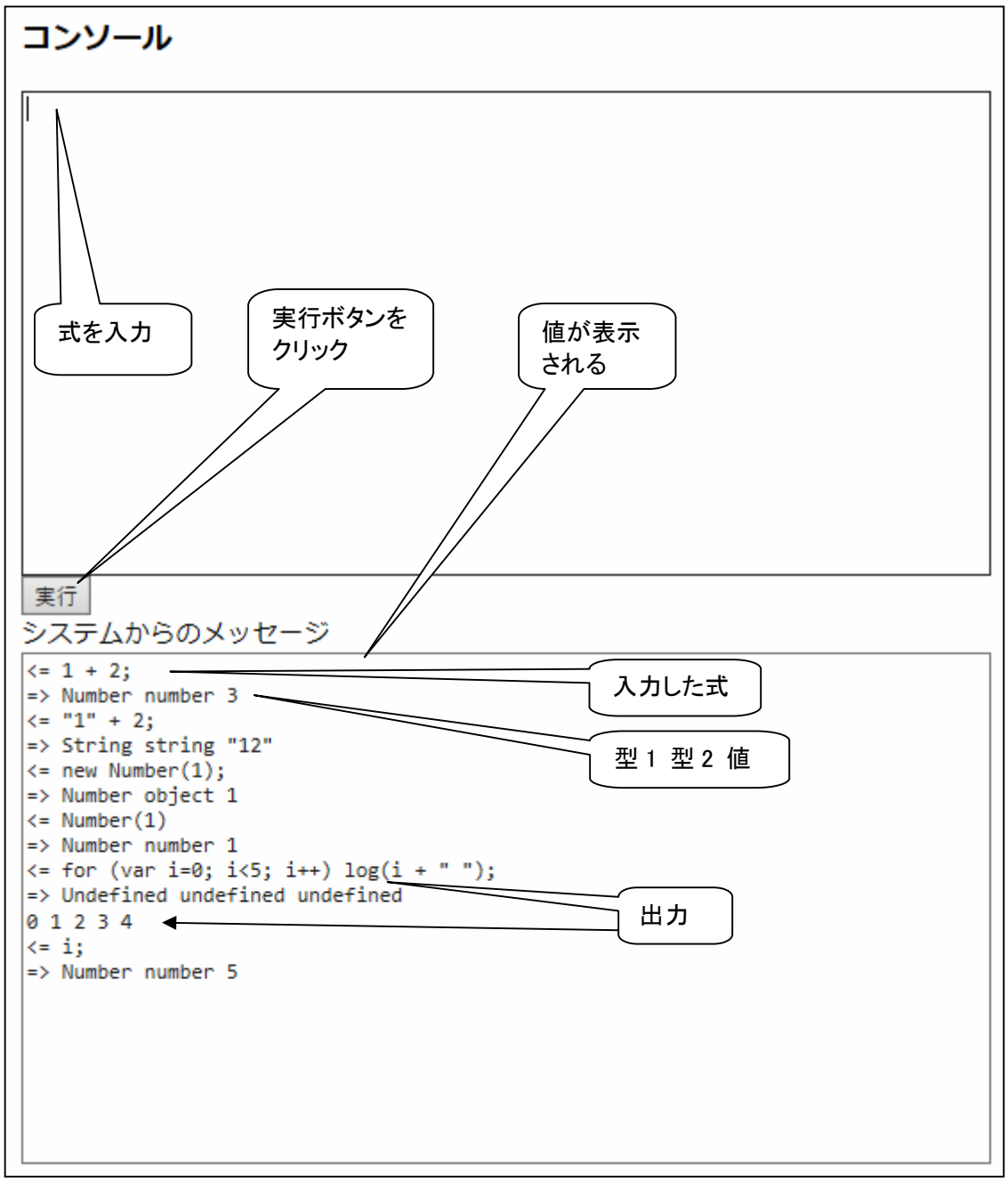
JavaScript プログラムから console.log(式) で
コンソール領域への出力が可能です。

```
for (var i=0; i<5; i++) console.log(i);
```

第6回　デバッグのやりかた：実践編4（便利ツールを作っておこう）

コンソールの入力は 1 行です。shift+enter を使うと、複数行にまたがる式も入力できますが、ちょっと不便です。プログラムのデバッグ中でなく、数行の JavaScript 文を入力して、その場で結果を見たい場合に便利なツールを紹介します。

画面上に、JavaScript の「式」を入力し、「実行」ボタンをクリックすると、その「値」（評価結果）を表示します。JavaScript では、「文」も、「式」の一種として扱われます。入力と実行は何回も繰り返すことができます。入力した式にエラーが含まれる場合は、エラーメッセージが表示されます。



```
<!DOCTYPE html>
<html>
 <head>
    <meta charset="utf-8">
    <title>コンソール</title>
 </head>
 <body>
    <h3>コンソール</h3>
    <textarea rows="19" cols="80" id=pg autofocus>1 + 2;</textarea>
    <br><input type=button onClick=go() value="実行">
    <br>システムからのメッセージ
    <br><textarea rows="20" cols="80" id=log></textarea>
    <script>
var geval = eval;

var logp = document.getElementById("log");
var pgp  = document.getElementById("pg");
var logd;

function clog(s) { logp.value += s; }

function log(s) { logd += s; }

function typeIs(obj) {
      return(Object.prototype.toString.call(obj).slice(8, -1)); }

function isPrimitive(x) {
    return (typeof x)!="object";
}

function toLiteral(x) {
    return JSON.stringify(x);
}

function type(x) { return "" + (typeof x); }

function isInteger(n) { return n%1 === 0; }

function keys(obj) { return Object.keys(obj); }

function go() {
 logd = "";
 try {
            var v = geval(pgp.value);
            clog("<= " + pgp.value + "¥n=> "
               + typeIs(v) + " " + type(v) + " " + toLiteral(v) + "¥n");
            pgp.value = "";
            logp.scrollTop = logp.scrollHeight;
            pgp.focus();
 }
 catch(e) { clog(pgp.value + "¥n! " + e + "¥n");
                pgp.value = "";
                logp.scrollTop = logp.scrollHeight;
                pgp.focus();
 }
 if (logd != "") clog(logd + "¥n");
}
   </script>
 </body>
</html>
```

eval 関数をグローバル化
（グローバル化しないと、
毎回、変数値がリセット
されてしまう）

エラー捕捉