ななちゃんのIT教室

エラーは投げるもの？の巻

by nara.yasuhiro@gmail.com

ななちゃんが、JavaScript の
エラー処理の方法を学ぶという お話

第 0.1 版　2017 年 6 月 18 日

フリー素材
http://freeillustration.net

いらすとやフリー素材
http://www.irasutoya.com/

もくじ

第1回　エラーをキャッチする
第2回　エラーを投げる
第3回　エラーを拡張する

第1回　エラーをキャッチする

なな：JavaScript のプログラムの実行時にエラーが発生すると止まってしまうけど、どうしようもないの？

先生：コンソールを開けばエラーメッセージを見ることができるかも知れないけど、一般のユーザには困難ね。たしかに、ウェブページ中に表示されたり、アラートが出るわけではないわね。

なな：プログラムでエラーを検出して、一般ユーザにエラーの情報を伝えることはできないの？

先生：できます。通常の状態では、エラーが発生すると、プログラムはそこで停止、終了してしまうけど、エラーを拾うようにプログラムしておけば、エラー発生時用のプログラムを実行することができの。まずは、エラーの種類にどんなのがあるのか整理しておきましょう。エラーのことを、JavaScript では、「例外」（exception）ということがあります。

Error	下記にあてはまらない、その他のエラー
RangeError	数値をあらわす変数又は引数が、その有効範囲外である場合に発生するエラー
ReferenceError	不正な参照を読み出した場合に発生するエラー
SyntaxError	eval() 内のコードを評価中に発生する構文エラー
TypeError	変数又は引数の型が有効でない場合に発生するエラー
URIError	encodeURI() 又は decodeURI() が不正な引数を渡された場合に発生するエラー

以下の2つは、仕様には規定されているが、現在のシステムでは発生しない。

EvalError	eval() に関して発生するエラー
InternalError	JavaScript エンジンで内部エラーが投げられた時に発生するエラー

これらのエラー情報それぞれに対する「エラーコンストラクタ」があり、プログラマがエラーオブジェクトを作って例外を生成（throw) することができます。

なな：例外を投げる（throw）の？

先生：プログラムで、このましくない状態、たとえば、ユーザ入力が不適切だったために処理続行が無意味になったと判断した場合に、システムやプログラムにエラー処理を依頼することを「例外を throw する」といいます。上記のような、既定のエラー種別以外に、プログラマが、独自のエラーオブジェクトを設計したり、規定のエラーオブジェクトを拡張することもできます

なな：まだ、エラーを検出して、対応する話になっていないみたい。

先生：エラーの捕獲（catch）の問題ね。下記のような枠組みになります。

```
try {
    // テストする文（多数行あってかまわない）
}
catch (e) {
    // エラー処理文。e が、エラーオブジェクト
}
```

これで、エラーを捉える（catch すること）が可能になります。

第2回　エラーを投げる

なな：これで、エラーが起きたときに捕捉できるわけね。プログラムでエラーを発見して throw する方法は？

先生：上記の「テストする文（多数行あってかまわない）」の中に、下記のような文を含めます。

```
throw new Error('不正な郵便番号表記です');
```

なな：「throw '不正な郵便番号表記です';」 ではダメなの？

先生：プログラム次第になります。ここで、「Error コンストラクター」の使い方を説明しておきましょう。

```
new Error([message[, fileName[, lineNumber]]])
```

- message 人間が読めるエラーの説明。文字列型。

以下は、標準化されていない。

- fileName Error オブジェクト上の fileName プロパティに設定される値。文字列型。
  デフォルトでは、Error() コンストラクターを呼び出したコードを含むファイルの名前。
- lineNumber Error オブジェクト上の lineNumber プロパティに設定される値。数値型。
  デフォルトでは、Error() コンストラクターの呼び出しを含む行番号。

先生：「Error」 以外にも、RangeError、ReferenceError、SyntaxErrorなどに対応したコンストラクターもあります。
Error インスタンスには、独自のプロパティやメソッドはないけど、派生元のクラスから継承しているものがあります。

message	エラーメッセージ。コンストラクタで設定したものと同じ文字列。
name	エラーの名称。「RangeError」などの、エラー種別を表す文字列。

標準化されていない、ブラウザ依存のプロパティに、lineNumber や、fileName がある。

なな：やっぱりプログラム例が欲しいな。

先生：どうぞ。

```
var s = "";
try           { x; }
catch (e) { s += "<" + e.name + "><" + e.message + ">" + e; }
s;        // String string "<ReferenceError><'x' is undefined>ReferenceError: 'x' is undefined"
```

```
var s = "";
try           { throw new Error('Whoops!'); }
catch (e) { s += "<" + e.name + "><" + e.message + ">" + e; }
s;        // String string "<Error><Whoops!>Error: Whoops!"
```

なな：それぞれの型のエラーを故意に発生させるプログラムがほしいな。

先生：どうぞ。

```
function f() { f(); } f()     // ! Error: Out of stack space
x                             // ! ReferenceError: 'x' is undefined
{ []:1 }                      // ! SyntaxError: Expected ';'
decodeURIComponent('%');      // ! URIError: The URI to be decoded is not a valid encoding
null.f();                     // ! TypeError: Unable to get property 'f' of undefined or null reference
new Array(-1);                // ! RangeError: Array length must be a finite positive integer
```

第3回 エラーを拡張する

なな：さっき、Error クラスを拡張するような話もあったような。

先生：独自のエラー型のことね。Error クラスを継承した例外オブジェクトの定義例です。

```
var MyError = function(message) {
      this.name = "myError";
      this.message = message || "my error";
};
MyError.prototype = new Error();

var s = "";
try {
      throw new MyError();
} catch (e) {
      if (e instanceof MyError) {
            s += e.message;
      }
}
s;        // String string "my error"
```

Error クラスを継承する

先生：try/catch/finally というのもあります。try { ... } の中でエラー（例外）が発生した場合、それが catch { ... } で拾われないと、そこで処理がストップします。しかし、finally { ... } は、それでも実行されます。try の入れ子も可能です。この場合、catch で捕捉されなかった例外は、外側の try に対する catch で捕捉が可能です。

```
var s = "";
try {
    try      { x; }
    finally { s += "<finally>" }     // 実行される
    s += "<next>";                   // 実行されない
}
catch (e) {
    s += "<" + e + ">";
}
s;        // String string "<finally><ReferenceError: 'x' is undefined>"
```

先生：エラーを捕捉する別の方法もあります

```
window.addEventListener('error', function (e) {
      var error = e.error;
      alert(error);
});
x
```

先生：ここで、error は、エラークラスのインスタンスなので、error.name、error.message などのプロパティを参照できます。

なな：エラーの扱い方が分かりました！