# 応用情報処理 No. 9

# JavaScriptの関数

プログラムのなかでひとまとまりの処理の部分を抜き出して名前をつけ、その処理が必要なときにこのひとまとまりの処理にジャンプするようにすると、プログラムの全体の見通しがよくなる。また、このひとまとめの処理を何度も実行する必要がある場合は、同じコードを何度もソース・プログラムに記述する必要がないので経済的であり、記述ミスの確率を減らすことができる。JavaScriptでは、このひとまとめの処理は関数(function)と呼ばれる。他のプログラミング言語[[1]](#footnote-1)では、関数と呼ばずにサブルーチン(subroutine)と呼ぶものもある。これは、プログラムの最初に実行される部分をメインルーチン(main routine)と呼ぶことに対して、メインルーチンから呼び出されるひとまとまりの処理である関数をサブルーチン（部分ルーチン）と呼ぶためである。

関数はメインルーチンからだけではなく、サブルーチンである関数の処理の途中からそのサブ・サブルーチンとして呼び出すこともできる。

関数を呼び出すときに、パラメータを引数として渡すこともできる。また、関数の実行結果として得られた値を返り値（/戻り値）として、返すこともできる。

JavaScriptでの関数の定義は、function文で行う。

function 関数名(引数1, 引数2, ・・・){

 JavaScript文1;

 JavaScript文2;

 JavaScript文3;

 ・・・

 return 返り値;

}

## 関数の考え方

次のsum.htmlは、1から引数xまでの数の和を計算するプログラムである。

[sum.html]

<html>

<head>

<meta http-equiv="content-type" content="text/html;charset=Shift\_JIS">

<title>関数にしない場合の例</title>

</head>

<body>

<h1 style="color: #E00;">簡単な関数の例</h1>

<p>1からxまでの数の和を計算します。</p>

<script type="text/javascript">

 var x, sum, i;

 // １から4まで足す

 x = 4;

 sum = 0;

 i = 1;

 while( i <= x ){

 sum = sum + i;

 ++i;

 }

 document.write("1から4まで足すと、" + sum + "です。<br>");

 // 1から10まで足す

 x = 10;

 sum = 0;

 i = 1;

 while( i <= x ){

 sum = sum + i;

 ++i;

 }

 document.write("1から10まで足すと、" + sum + "です。<br>");

</script>

</body>

</html>

これをadd\_from\_one(x)は、1から引数xまでの数の和を戻り値とする関数を使って書き直すと次のようになる。

[function.html]

<!DOCTYPE html>

<html lang="ja">

<head>

<meta charset="Shift\_JIS">

<meta http-equiv="Content-Type" content="text/html; charset=Shift\_JIS">

<title>簡単な関数の例</title>

<script type="text/javascript">

function add\_from\_one( final\_num )

{

 var sum = 0;

 var i = 1;

 while( i <= final\_num ){

 sum = sum + i;

 ++i;

 }

 return sum;

}

</script>

</head>

<body>

<h1 style="color: #E00;">簡単な関数の例</h1>

<p>add\_from\_one(x)は、1からxまでの数の和を求める関数です。</p>

<script type="text/javascript">

 var sum;

 // １から4まで足す

 sum = add\_from\_one(4); //引数を4としてadd\_from\_one()関数を呼び出す

 document.write("1から4まで足すと、" + sum + "です。<br>");

 // １から10まで足す

 sum = add\_from\_one(10); //引数を10としてadd\_from\_one()関数を呼び出す

 document.write("1から10まで足すと、" + sum + "です。<br>");

</script>

</body>

</html>

### Do! sum.html（応用情報処理のホームページから入手可能）をコピーしてfunction.htmlという名前にし、上のようにadd\_from\_one()関数を定義してソース・コードを書き換えなさい。そして、add\_from\_one()関数を使うことの利点は何か考察しなさい。

### Do! 上で作成したfunction.htmlに、1から100までの和と、1から1000までの和の計算結果も表示されるように、コードを追加しなさい。

## オブジェクト

JavaScriptでは、オブジェクト指向のプログラミング言語のように、関連のある変数や関数を一まとめにして一つのオブジェクト（もの）として扱うことができる。オブジェクト指向のプログミングでは、オブジェクトの変数はプロパティ、関数はメソッドと呼ばれる。

例えば、猫のタマ、トラ、クロの名前、色、鳴き方という情報をプログラムで扱う時、

var tama\_name = "タマ";

var tama\_color = "白";

function tama\_mew() { alert( "みゃー" ); }

var tora\_name = "トラ";

var tora\_color = "茶";

function tora\_mew() { alert( "にゃー" ); }

var kuro\_name = "クロ";

var kuro\_color = "黒";

function kuro\_mew() { alert( "なー" ); }

のように変数や関数を定義するであろう。tama\_、tora\_、kuro\_という接頭辞をつけて区別しているが、name、color、mew()を一まとめにして一つのオブジェクトとして扱うのが自然であろう。それらは、tamaオブジェクト、toraオブジェクト、kuroオブジェクトというもので表現することができる。

tamaオブジェクトの定義は、

var tama = {

 name: "タマ",

 color: "白",

 mew: function() { alert( "みゃー" ); }

}

のように書くことができる。tama.name、tama.colorが変数名となり、各々の値は”タマ”、”白”となる。tama.mew()は関数名となる。これらのJavaScriptのオブジェクトは、プロパティ名やメソッド名がキーとなる連想配列（ハッシュ）[[2]](#footnote-2)と呼ばれるものと同じデータの構造を持っている。

下に、オブジェクトを使った例を示そう。

[object.html]

<!DOCTYPE html>

<html lang="ja">

<head>

<meta charset="Shift\_JIS">

<title>オブジェクト</title>

[ここに、tama、kuro、toraの定義が必要だが、省略]

</head>

<body>

<h1>オブジェクト</h1>

<script type="text/javascript">

tama.mew();

document.write('<p>吾輩は猫である。名前は' + tama.name + 'である。毛の色は' + tama.color + '色である。</p>')

tora.mew();

document.write('<p>吾輩は猫である。名前は' + tora.name + 'である。毛の色は' + tora.color + '色である。</p>')

kuro.mew();

document.write('<p>吾輩は猫である。名前は' + kuro.name + 'である。毛の色は' + kuro.color + '色である。</p>')

</script>

</body>

</html>

### Do! object0.html（応用情報処理のホームページから入手可能）に、kuroとtoraのオブジェクトの定義を追加して、上のobject.htmlを完成させ、ブラウザで動作を確かめなさい。

### クラス

JavaScriptでは、変数名がオブジェクトを指すと解釈された場合、そのオブジェクトのプロパティやメソッドを使用することができる。オブジェクトを定義するときは、あらかじめプロパティとメソッドが定められたクラスと呼ばれるものから生成すると、便利である。

　例として、猫のオブジェクトを生成するCatクラスを以下のように定義してみる。

///////////// Cat クラス 始まり /////////////////

// 猫を表すクラス。名前、毛色、鳴き声

//// コンストラクタは、プロパティのみ ////

function Cat( name, color, voice ) {

 if (!(this instanceof Cat)) {

 return Cat.help;

 }

 if( name === undefined ){ name = "名無し"; }

 if( color === undefined ){ color = "不明"; }

 if( voice === undefined ){ voice = "..."; }

 this.name = name;

 this.color = color;

 this.voice = voice;

}

Cat.help =

 "\n"

 + "new Cat(name, color, voice) -- 新しい猫オブジェクトを生成\n";

//// メソッドはprotoypeで参照 ////

Cat.prototype.mew = function( ){

 alert( this.voice );

}

///////////// Cat クラス 終わり /////////////////

関数 Cat()は、Catクラスのコンストラクタと呼ばれ、猫のオブジェクトを定義するのに必要な情報を引数として、thisという特殊変数を使って、プロパティを定義している。メソッドについては、Cat.prototype.メソッド名という名前の関数として定義している。

クラスに属する具体的なモノ（オブジェクト）はインスタンスと呼ばれる。Catクラスのインスタンスとして、tamaオブジェクトを生成（定義）するには、次のようにする。

var tama = new Cat( "タマ", "白", "みゃー" );

### Do! class0.html（応用情報処理のホームページから入手可能）に、kuroとtoraのオブジェクトの生成を追加して、上のclass.htmlを完成させ、ブラウザで動作を確かめなさい。

### Stringクラス

Javascriptには最初から定義されているクラスがあり、それらを利用することができる。例えば、文字列のクラスはStringクラスとよばれ、そのオブジェクト[[3]](#footnote-3)mojiretuは、

var mojiretu = new String(“あいうえお”);

のようにして生成することができる。var mojiretu = “あいうえお”;とした場合は、これの省略形と解釈される。Stringクラスのオブジェクトのlengthプロパティは文字列の文字数を格納する変数、charAt(i)メソッドは(i+1)番目の文字を返す関数である。これらの動作は次の例をみるとわかるであろう。

 [string.html]

<!DOCTYPE html>

<html lang="ja">

<head>

<meta charset="Shift\_JIS">

<title>Stringオブジェクト</title>

</head>

<body>

<h1>Stringオブジェクト</h1>

<script type="text/javascript">

var mojiretu = new String("あいうえお");

alert(mojiretu.length);

for(var i = 0; i< mojiretu.length; i++ ){

 document.write('<p>' + i + ' : ' + mojiretu.charAt(i) + '</p>');

}

</script>

</body>

</html>

### Do! string.html（応用情報処理のホームページから入手可能）の動作をブラウザで確認しなさい。更に、mojiretuを“あいうえお”とは異なる別の文字列で初期化するようにソース・コードを変更して、同様にブラウザで動作を確かめなさい。

## 関数の再帰呼び出し（recursive call）の例

関数の定義の中で、自分自身の関数を呼び出す方法を関数の再帰呼び出しという。似たような処理を繰り返す回数が固定されていない場合に関数の再帰呼び出しを使って定義すると、直観的にわかり易いプログラムにすることができる。例えば、生年月日の数字占いのとき、生年月日の8ケタの数値を桁ごとに合計することを繰り返して1ケタの数字にするという計算をするが、関数の再帰呼び出しを使うことで、プログラムをすっきりさせることができる。

[recursive.html]

<!DOCTYPE html>

<html lang=”ja”>

<head>

<meta charset=”Shift\_JIS">

<title>再帰的関数の例</title>

<script type="text/javascript">

function add\_digits( number )

{

// console.log("引数:" + number);

 var mojiretu = new String(number); // numberの値を文字列とする

 var length = mojiretu.length; // mojiretuの文字数

 var sum = 0;

 var i = 0;

 for(i = 0; i < length; i++){

 sum += eval(mojiretu.charAt(i)); //mojiertu.charAt(i)はmojiretuのi番目の文字

 }

 if( sum >= 10 ) {

 sum = add\_digits( sum ); // 再帰呼び出し（recursive call）

 }

 return sum;

}

</script>

</head>

<body>

<h1 style="color: #E00;">再帰的関数の例</h1>

<p>関数の再帰的利用の例です。</p>

<script type="text/javascript">

 var input = Number( prompt("生年月日を8ケタの半角数字で入力してください。例）19401009", "19401009") );

 var sum = add\_digits(input);

 document.write(input + "生まれの数字は、" + sum + "です。<br>");

</script>

</body>

</html>

この例では、add\_digits( number )という関数の主な処理は、引数のnumberの各桁の和を計算してその結果をsumに格納する。もしこのsumが10以上であれば、1ケタになっていないので、add\_digits( sum )として、関数add\_digitsの中からadd\_digitsを呼び出すということを繰り返す。関数の中からその関数自身を呼び出すことを再帰呼び出しと言う。

### 【重要】console.logの利用

コメントになっている//console.log("引数:" + number);の行の//をとると、FirefoxのWebコンソールのロギングで、add\_digits()関数が呼び出されるときの引数である変数numberの値を表示させることができる。引数:194001009、引数:24というように2回呼び出されていることがわかる。console.logはデバッグのときに役立つ。

Do! recursive.htmlを作成して（応用情報処理のホームページから入手可能）、自分の生年月日を入力して、ブラウザで動作を確かめなさい。更に、console.log()を有効にして実行（再読み込み）し、Webコンソールでロギングを有効にして、その表示結果を確かめなさい。最後に、以下のコードを追加して、計算結果（変数sumの値）の意味の解説サイトにジャンプできるようにしなさい。

document.write( 'この数字の意味を知りたい方は、<a href="http://www.etwa-ru.com/uranai/cabara/cabara0' + sum + '.html" target="\_blank">ここ</a>をクリックしてください。');

## 相性占い

誕生月で相性占いを作成して、関数、フォーム（<form>～</form>）、ボタン（<input type=”button”>）の使い方を学ぼう。

## uranai1.html

<!DOCTYPE html>

<html lang="ja">

<head>

<meta charset="Shift\_JIS">

<title>相性占い</title>

</head>

<body bgcolor="white">

<h1>インド人もビックリ！</h1>

誕生月で、二人の相性がわかります。ホンマかいな？

<p>

<script type="text/javascript">

<!--

var yourBirth = Number( prompt("あなたの誕生月を半角数字で入力してください。", ""));

var otherBirth = Number( prompt("相手の誕生月を半角数字で入力してください。", "") );

var num = Math.abs(yourBirth - otherBirth);

if(num == 4 || num == 8) {

 alert("相性は最高です。");

} else if(num == 2 || num == 6 || num == 10) {

 alert("相性は最悪です。");

} else {

 alert("相性はまあまあです。");

}

//-->

</script>

</p>

</body>

</html>

【解説】

(1) Math.abs()関数

引数の絶対値を返す関数。

Do! uranai1.htmlを作成し、ブラウザで動作を確かめなさい。相性が、最高、最悪、まあまあとなる誕生月の組み合わせの例を見つけなさい。

## サブルーチン（関数）化

次は、uranai1.htmlを、関数（または、サブルーチン）を使って書き直しものである。これにより、メインルーチンはたったの3行となる。

## uranai2.html

<!DOCTYPE html>

<html lang="ja">

<head>

<meta charset="Shift\_JIS">

<title>相性占い</title>

<script type="text/javascript">

<!--

// aisyouという名前でひとかたまりの処理をまとめた。

function aisyou(birth1, birth2){

 var num = Math.abs(birth1 - birth2);

 if(num == 4 || num == 8) {

 alert("相性は最高です。");

 } else if(num == 2 || num == 6 || num == 10) {

 alert("相性は最悪です。");

 } else {

 alert("相性はまあまあです。");

 }

}

//-->

</script>

</head>

<body bgcolor="white">

<h1>インド人もビックリ！</h1>

誕生月で、二人の相性がわかります。ホンマかいな？

<p>

<script type="text/javascript">

<!--

// メインルーチン

var yourBirth = Number( prompt("あなたの誕生月を半角英数字で入力してください。", "") );

var otherBirth = Number( prompt("相手の誕生月を半角英数字で入力してください。", "") );

aisyou(yourBirth, otherBirth); // 関数 aisyouをyourBirth, otherBirthを引数にして呼び出す

//-->

</script>

</p>

</body>

</html>

【解説】

aisyouという名前の関数を定義して、ひとかたまりの処理をまとめた。この関数の定義function aisyou()を記述する場所はどこでもよいので、HTMLのヘッダー部（<head>～</head>）に記述した。

このひとかたまりの処理は、関数名aisyouで呼び出すことができるようになる。これで、無駄な繰り返しが減ったり、プログラムの構造（特にメインルーチン）が見やすくなったりする。

Do! uranai1.htmlをコピーしてuranai2.htmlという名前にし、上のように関数aisyou()を使って書き直しなさい。ブラウザで動作を確かめなさい。3行に短くなるメインルーチンの部分はどの部分か答えなさい。

## ボタンの使用

JavaScriptはソースがブラウザに読み込まれたときに実行される。ボタン（<input type=”button”>）を作成して、ボタンをクリックすることで、占いが実行されるように書き換えてみよう。ボタンを使用するには、フォーム（<form>～</form>）を定義して、そのなかにボタンを配置する必要がある。

## uranai3.html

<!DOCTYPE html>

<html lang="ja">

<head>

<meta charset="Shift\_JIS">

<title>相性占い</title>

<script type="text/javascript">

<!--

// promptを表示する部分をstart\_uranaiというfunctionにまとめた

function start\_uranai(){

 var yourBirth = Number( prompt("あなたの誕生月を半角数字で入力してください。", "") );

 var otherBirth = Number( prompt("相手の誕生月を半角数字で入力してください。", "") );

 aisyou(yourBirth, otherBirth); // 関数 aisyouをyourBirth, otherBirthを引数にして呼び出す

}

// aisyouという名前でひとかたまりの処理をまとめた。

function aisyou(birth1, birth2){

 var num = Math.abs(birth1 - birth2);

 if(num == 4 || num == 8) {

 alert("相性は最高です。");

 } else if(num == 2 || num == 6 || num == 10) {

 alert("相性は最悪です。");

 } else {

 alert("相性はまあまあです。");

 }

}

//-->

</script>

</head>

<body bgcolor="white">

<h1>インド人もビックリ！</h1>

誕生月で、二人の相性がわかります。ホンマかいな？

<p>

<form>

<!-- ボタンをクリックして関数start\_uranai()を呼び出すには、onClick属性に呼び出す関数を指定します -->

<input type="button" value="占い開始" onClick="start\_uranai()">

</form>

</p>

</body>

</html>

【解説】

(1) JavaScriptでのボタンの使用法

<input>タグのtype属性の値として”button”を指定するとボタンを表示する（後述するように、type=”text”とすると、テキストボックスを表示する）。また、value属性の値には、そのボタンに表示する文字列を指定する。この例ではvalue="占い開始"である。

ボタンを表示する<input type=”button”>タグのonClick属性の値として、このボタンがクリックされたときに呼び出したいJavaScriptの関数名を指定する。

Do! uranai2.htmlをコピーしてuranai3.htmlという名前にし、上のように書き直しなさい。ブラウザで動作を確かめなさい。

## テキストボックスの使用

prompt()関数の代わりに、フォームにテキストボックス（<input type="text">）を配置して、誕生月の入力ができるようにしよう。

## uranai4.html

<!DOCTYPE html>

<html lang="ja">

<head>

<meta charset="Shift\_JIS">

<title>相性占い</title>

<script type="text/javascript">

<!--

// promptを表示する部分をstart\_uranaiというfunctionにまとめた

function start\_uranai(form){

 // 関数 aisyouをyourBirth, otherBirthを引数にして呼び出す

 aisyou( Number( form.YourBirth.value ), Number( form.OtherBirth.value ) );

}

// aisyouという名前でひとかたまりの処理をまとめた。

function aisyou(birth1, birth2) {

 var num = Math.abs(birth1 - birth2);

 if(num == 4 || num == 8) {

 alert("相性は最高です。");

 } else if(num == 2 || num == 6 || num == 10) {

 alert("相性は最悪です。");

 } else {

 alert("相性はまあまあです。");

 }

}

//-->

</script>

</head>

<body bgcolor="white">

<h1>インド人もビックリ！</h1>

誕生月で、二人の相性がわかります。ホンマかいな？

<form>

<p>

あなたの誕生月を半角英数字で入力してください:<input type="text" name="YourBirth" size=4 maxlength=2><br>

お相手の誕生月を半角英数字で入力してください:<input type="text" name="OtherBirth" size=4 maxlength=2><br>

</p><p>

<input type="button" value="占い開始" onClick="start\_uranai(this.form)">

<input type=”reset" value="リセット">

</p>

</form>

</body>

</html>

【解説】

(1)テキストボックスを使用する方法

テキストボックスに<input type=”text” name=”YourBirth”>のように名前YourBirthを付けると、このテキストボックスにJavaScriptからアクセスするときは、テキストボックスが定義されているフォームの名前をformとすると、form.YourBirthで制御することができる。このときテキストボックスの値を表す変数は、form.YourBirth.valueとなり、値の読み書きができるようになる。

(2)フォームの参照の仕方

この例では、フォームには名前を特に付けていないが、start\_uranai(this.form)というように、this.formでフォームを参照している。thisというのは、その処理が行われているオブジェクトを表す記号なので、特にフォームに名前を付けなくても、参照することができる[[4]](#footnote-4)。

Do! uranai3.htmlをコピーしてuranai4.htmlという名前にし、上のように書き直しなさい。ブラウザで動作を確かめなさい。

# 8. 課題9

uranai4.htmlを画像などを追加したり、判定条件を変更して改造し、uranai.htmlとして占いゲームを作成しなさい。

（参考：http://www.ipc.hokusei.ac.jp/~z00328/adjyou/ex/uranai.html）

# 9. Check List

* Javascriptで、関数（サブルーチン）を使用することの利点を理解したか？
* Javascriptで、関数（サブルーチン）を定義する仕方を理解したか？
* Javascriptで、関数（サブルーチン）を呼び出す仕方を理解したか？
* オブジェクト指向プログラミングで、オブジェクトのプロパティとは何か理解したか？
* オブジェクト指向プログラミングで、オブジェクトのメソッドとは何か理解したか？
* Javascriptでオブジェクトを定義したいとき、どのように書けば良いか。
* console.log()の使い方を理解したか？
* Math.abs()はどのような関数か？
* HTMLのform要素の使い方を理解したか？
* HTMLの<input type=”button” value=”ほげほげ”>はどのような要素を定義するか理解したか？
* HTMLの<input type=”button” ・・・>で定義される要素のonClick属性は、どのような役割をするか理解したか？
* HTMLの<input type=”text” ・・・>はどのような要素を定義するか理解したか？
* HTMLの<input type=”reset" value="リセット">はどのような要素を定義するか理解したか？
1. 例えば、Perlがあります。ユーザが定義した関数をサブルーチンと呼んでいます。因みに、C言語ではサブルーチンと言わずに、メインルーチンも含めて、すべて関数と呼びます。その意味でJavaScriptはC言語の習慣に従っています。 [↑](#footnote-ref-1)
2. Perlに実装されている連想配列、またはハッシュ変数で上の例に対応したものを書くと、メッソッドを除いて、%tama={ “name”=>”タマ”, “color”=>”白”}という連想配列になる。Perlの場合の$tama{name}、$tama{color}は、それぞれJavaScript場合のtama.name、tama.colorに対応する。 [↑](#footnote-ref-2)
3. オブジェクト指向では、オブジェクトというより、そのクラスのインスタンスということもある。 [↑](#footnote-ref-3)
4. この場合、thisは具体的にはINPUT要素(type=”button”、value=”占い開始”)を指す。this.formはその親要素であるフォームを指すという仕様である。 [↑](#footnote-ref-4)