



SECTION 1: SUMMARY

# JavaScriptとは

JavaScriptは、一言でいうと「オブジェクト指向スクリプト言語」です。しかし、この一言では、どんなものなのか、あまりピンとこないかもしれません。そこで、最初にJavaScriptの成り立ちから見ていくことにしましょう。まずは、その概要を理解してください。

## ■JavaScriptの成り立ち

JavaScriptは、Netscape Communications社(以下Netscape社)。現在のAOL Time Warner社)が「Netscape Navigator2.0(以下NN2)」のベータ版を公開したときに実装された「LiveScript」を起源にしています。そして、NN2が正式版となった際に、「JavaScript1.0」として発表されました。これがJavaScriptの始まりです。NN2がWebブラウザのシェアの大半を握るにしたがい、JavaScriptはWeb上でのスクリプト言語として重要な役割を担うようになりました。

この後しばらくして、JavaScriptにとって大きな出来事が起こります。それは1996年の8月に、Microsoft社から「Internet Explorer3.0(以下IE3)」が登場したことです。このWebブラウザは、Windowsに特化した「ActiveX」という技術を導入し、さらにスクリプトとしても「Visual Basic Scripting Edition(以下VBScript)」とともに、JavaScript1.0互換の「JScript1.0」を実装していました。そして、IE3は無償ということもあり、急速にシェアを伸ばしたのです。

Netscape社は、このIE3に対抗する形で「Netscape Navigator3.0(以下NN3)」を1996年10月に登場させました。

NN3には、大幅な機能強化を行った「JavaScript1.1」が実装されており、この機能強化によって、Webページを「インタラクティブなもの」(制作者とユーザーが相互に情報を送り合えるもの)にできるようになりました。しかし、その結果、IE3に実装されているJScript1.0とJavaScript1.1の間で、互換性の問題が発生しました。JavaScript1.1で強化された機能を使うと、IE3では正常に動作しないのです。これ以後、Webページ制作者は互換性の問題に悩まされるようになりました。

1997年になると、インターネットはより広く一般に使われるようになりました。Microsoft社、Netscape社はそれぞれ、機能を強化した「Internet Explorer4.0(以下IE4)」「Netscape Communicator4.0(ブラウザ部分はNetscape Navigator4.0:以下NN4)」を出荷し、IE4は「JScript3.0(JavaScript1.1互換)」、NN4は「JavaScript1.2」をそれぞれ搭載しました。ただ、JavaScript1.2の機能拡張はNN4専用のものが多くたため、一般にはあまり利用されませんでした。

## ▼JScript3.0を搭載したInternet Explorer4.0



## ▼JavaScript1.2を搭載したNetscape Navigator4.0



その後、1998年8月にリリースされたNN4.06では、「JavaScript1.3」が実装されています。このバージョンではUnicodeに対応し、さらに新しいメソッドやプロパティが追加されました。

そして、1999年の3月に、Webブラウザの位置付けを変える製品が登場します。それが「Internet Explorer5(以下IE5)」です。IE5は、見た目こそIE4から大きな変化はありませんが、「XML (eXtensible Markup Language)」や「XSL (eXtensible Stylesheet Language)」に対応することにより、Webブラウザの枠に収まらず、アプリケーションのフロントエンドを担える製品になってきました。スクリプト面ではJScriptが、アプリケーション作成に堪えられるまでに機能強化されました。

さらに、2001年9月に「Internet Explorer6(以下IE6)」が出荷されました。IE6では、「ECMA Script」(p.6参照)やW3C(World Wide Web Consortium)の各種仕様への対応がより進んでいます。また、Windows Media Playerとの統合も進むなど、よりリッチなコンテンツに対応し、ブロードバンドをはじめとするインターネット環境の進化に合わせたソフトウェアになっています。スクリプト面では、IE6ではJScriptに大きな進化はなく、基本的にIE5とほとんど同じです。

### ▼JScript5.5を搭載したInternet Explorer6



Netscape社からは、2000年12月に「Netscape6(以下NN6)」が公開されました。Netscape社は、NN6を開発するにあたり、Netscape Navigatorのソースコードを

公開して、「Mozillaプロジェクト」(<http://www.mozilla.org/>)を立ち上げ、ネット上での開発を試みました。全面的なソースコードの書き換えや、W3C準拠の各種機能サポートなど野心的なプランでしたが、その完成は遅れました。

NN6は、このMozillaプロジェクトで開発されたプログラムを基に、さらに改良を行ったものです。しかし、動作スピードが非常に遅く、多くのバグが存在したため、あまり使われませんでした。その後、いくつかの改良が加えられましたが、それほどシェアは伸びませんでした。そこで、さらに改良を加え、2002年8月にリリースされたのが、「Netscape7」です。Netscape7では、パフォーマンスが向上し、タブブラウジング機能を実装するなど使いやすくなっています。スクリプトとしては、「JavaScript1.5」が搭載されていますが、大きな機能向上はありません。

#### ▼JavaScript1.5を実装したNetscape7



この2大Webブラウザメーカー以外にも目を向けてみましょう。2003年6月にはApple Computer社から、Mac OS X用のWebブラウザ、「Safari」が公開されました。Mac OS Xは、それまでMicrosoft社のIEをデフォルトのWebブラウザとしていました。それが、今後のMac OS Xマシンでは、デフォルトで「Safari」が搭載されることになります。

## ▼Apple Computer社のSafari



以上のように発展してきたWebブラウザ(と実装されたJavaScript)ですが、今後は、そのあり方自体が大きく変わる可能性があります。

たとえば、Internet ExplorerというWebブラウザは、単体で提供されるのはIE6(Mac版はIE5)で終わりになります。これはMicrosoft社が、Webブラウザは次の主戦場ではないと見ている証拠でしょう。

また、これまでHTMLの表現力を高めるために、WebブラウザやJavaScriptの機能を更新してきました。しかし、現在のWebページ上の表現手法は、すでに一定のレベルに達したと考えられます。そのため、最近ではJavaScriptの機能追加もほとんど見られません。むしろ、これからのJavaScriptは、さまざまな要素(Webブラウザのオブジェクト、Macromedia Flash、XMLドキュメントの要素など)を操作するための基本的な技術として成熟していくに違いありません。

## ●ECMA Script

1996年にIE3とNN3が出荷され、それぞれに強力なJScript1.0、JavaScript1.1が実装されたことで、JavaScriptを使ったさまざまなWebページ上の表現が可能になりました。しかし、すでに述べたように、互換性という非常に大きな問題が起こりました。どちらもJavaScriptをベースにしていたのですが、独自の拡張が行われたり、実装の不備があるなどの理由から、片方のWebブラウザで動いたスクリプトが、もう一方のWebブラ

ウザでは動かないといったことがひんぱんに起こりました。そのため、Webページ制作者は、ページ制作時にそれぞれのWebブラウザで動作を確認したり、ひどいときにはそれぞれのWebブラウザ専用に別々のWebページを用意する必要があったのです。

この状況に困ったWebページ制作者は、標準規格を求めるようになりました。そこでNetscape社は、JavaScriptの標準化案を国際標準化団体のECMAに提出し、Microsoft社も対抗案を提示しました。ECMAは、両者の標準化案をうまく融合させ、1997年7月に標準規格「ECMA-262」として制定しています。これは、「ECMA Script」と呼ばれ、ベースとなったJavaScript1.1に似たものになっています。

その後、ECMA-262の機能拡張が行われ、現在は第3版(1999年12月制定)となっています。JavaScriptの実装にあたっては、この規格が標準となります。実際に、IE6やNN6などは、ECMA Scriptへの準拠度が高くなっています。

ECMAについては、次のURLを参照してください。

#### URL

<http://www.ecma-international.org/>

## ●JScript

ここで、JScriptについて簡単にまとめておきましょう。JScriptは、Microsoft社がJavaScript互換のスクリプト言語として実装したものです。Microsoft社は、スクリプト言語をIE3に実装するにあたって、念入りにアーキテクチャを検討し、「ActiveX Scripting」という非常にスマートな形で実装してきました。

この技術で重要なのは、Webブラウザからスクリプトの実行エンジンを切り離したことです。これにより、スクリプトエンジンのバージョンだけを上げる、といったことが可能になります。たとえば、IE5を使っていたとしても、スクリプトエンジンのバージョンをアップデートすれば、新しい機能が使えるようになります。さらに、複数のスクリプトを動作させるアプリケーション(ホストアプリケーションと呼ぶ)で、1つのスクリプトエンジンを共有することもできます。また、1つのOS上に、それぞれ異なるスクリプト言語を実行するスクリプトエンジンを用意することで、ホストアプリケーションに対して複数のスクリプト言語の使用を許可することもできます(IEで、VBScriptとJScriptが動作するのはこのためです)。

2003年8月現在、JScript実行エンジンの最新のバージョンは「JScript5.6」となっています。Windowsのスクリプトについては、次のURLを参照してください。

#### URL

<http://msdn.microsoft.com/scripting/>

## ■JavaScriptはスクリプト言語

JavaScriptは、名前の示すとおり、スクリプト言語です。スクリプト言語は、以前からあるもので、UNIX環境では非常によく使われてきました。反復的なシステム管理タスクをひとまとめにするプログラムを記述し、システム管理を簡単にする、といった目的に使われています。プログラミング言語ほど大がかりな処理には向きませんが、簡単なプログラムを手軽に書いたり、実行したりできるのがスクリプト言語の特徴です。代表的なスクリプト言語としては、「Bourne Shell」「Cshell」「Awk」「Perl」などがあります。これらのスクリプト言語とJavaScriptの大きな違いは、JavaScriptは主にWebブラウザ上で動作するものであるということです。

### ●JavaScriptとJavaの違い

JavaScriptには、「Java」という言葉が含まれています。しかし、実はJavaScriptとJavaは、まったく関係がありません。Netscape社はスクリプト言語の開発当初、LiveScriptという名前を使っていましたが、ちょうど同じ時期に発表されたJavaとのマーケティング的な関係から、NN2の出荷時に「JavaScript」という名前に変更したのです。そのため、名前に「Java」という言葉が含まれてはいますが、開発経緯、設計思想などが異なる、まったく別のものです。

ここで、Javaについても少しだけ触れておきましょう。Javaは、インターネットだけにとどまらず、昨今のコンピュータ業界の中で非常に重要な役割を占めるようになっています。そして、Javaには、プログラミング言語としての側面と、実行環境としての側面があります。

通常のプログラミング言語、たとえば、C言語、C++、Pascalなどといったものは、実行環境ごとに(要するに、それぞれのOSごとに)コンパイラが用意され、コンパイルした後の実行ファイルは、そのOSでしか動作しません。それに対してJavaは、「Java VM (Virtual Machine)」上で動作するという実装となっています。つまり、どのようなOS上であっても、Java VMさえ存在すれば、同じJavaのプログラムファイルを実行できるのです。

Javaの内容を大きく分類すると、「Webブラウザ上で動くJavaアプレット(Applet)」「クライアントJavaプログラム」「サーバサイドJavaプログラム」の3つになります。Webページ制作者にとって、JavaアプレットとサーバサイドJavaプログラムに注目する必要があるでしょう。

Javaアプレットは、Webサーバから「アプレット」と呼ばれる小さなJavaプログラムをダウンロードして、Webブラウザ上、またはOS上のJava VMの上で動作させるものです。ただし現状では、Java VMの起動が遅く、一方ではHTML(とスクリプト)の表現力が高くなっているため、Javaアプレットを使うことはほとんどありません。

サーバサイドJavaプログラムには、いくつかの種類がありますが、その中でもWebサーバーの機能拡張として使われている「Java Servlet」「EJB(Enterprise Java Beans)」「JSP(JavaServer Pages)」が注目されています。これらは、Javaの生産性の高さや安定性から、最近の電子商取引の基盤技術として使われ始めています。

これで、少なくともJavaとJavaScriptがまったく異なるものであるということが、おわかりいただけただでしょうか？

### ●JavaScriptの特徴

さまざまなインターネット関連技術の中でも、JavaScriptは特に重要な役割を担っています。まず、JavaScriptはWebブラウザの中で動作するため、WWW(World Wide Web)を使っているほとんどの人の環境で動作します。さらに、制作にあたっては単純なテキストエディタがあればよく、特別なツールを必要としません。また、スクリプト言語であるため、コンパイルの作業も不要です。JavaScriptは、数あるスクリプト言語の中でも、特に手軽に利用できるものであるといえます。

JavaScriptが、Webブラウザ側、つまりクライアント側で動作することも大きな特徴です。これは、Webサーバーの管理者でなくとも、Webページにさまざまな機能を簡単に追加できることを意味し、CGIなどと比較して、非常に大きなメリットとなっています。そのため、ほとんど同じことをするのであれば、CGIよりもJavaScriptで実装したほうがよいといえます。ただし、当然、JavaScriptにはできないことがあるので、実際にはCGIと上手に使い分けるのが最もよい制作スタイルであることは間違ひありません。

### ●JavaScriptではできないこと

JavaScriptではできないことを挙げておきましょう。大きく分けると2つあります。

1つ目は、セキュリティの関係から、Webブラウザの動作しているマシンから情報を取り出したり、書き込んだりできないことです(Cookieは除く)。これは、WebブラウザでWebページにアクセスしたときに、ハードディスク内のデータが勝手に取り出されたり、ハードディスク内にウイルスなどを書き込まれたりしないための仕様です。

2つ目は、Webページを見る人(ユーザー)からコードを隠せないことです。JavaScriptはHTMLファイル内にそのまま記述されているため、Webブラウザでページを表示した人は、簡単にコードを見ることができます。Webページ上で、ある機能をどのようにして実現しているかを知られたくないときには、JavaScriptではなく、ほかの方法を使うのがよいでしょう。

こういった制約があるため、JavaScriptは、クライアント側でのフォーム入力の際の補助的な処理や、ダウンロードしたWebページの要素をインタラクティブに動作させるといった用途に向いています。

# JavaScriptの展開

「1-1 JavaScriptとは」で紹介したように、JavaScriptには非常に簡単ながら、かなり強力なプログラムの作成能力があります。この点に注目して、さまざまな場面でJavaScriptを使おうという試みがなされています。

## ■DHTMLでのJavaScript

IE、NNの両者には、「Dynamic HTML(以下DHTML)」という機能が実装されています。これは名前こそ同じですが、IEとNNのDHTMLには、互換性はほとんどありません。

しかし、両者とも目指す方向性は同じです。それは、アニメーションGIFやJavaアブレットを使わずに、Webページ内の各HTML要素を、再読み込みなしに動かすというものです。

このDHTMLがどのように実装されているかというと、「Cascading Style Sheets(以下CSS)」と呼ばれるスタイルシートの各要素の属性値を、スクリプト言語によりダイナミックに変化させるという方法が取られています。そのため、スクリプト言語が重要なになっており、とりわけデファクトスタンダードであるJavaScriptの役割は重要なものとなっています。

さらに、最近ではDOM(Document Object Model)とJavaScriptを組み合わせることによって、DHTMLと同様の表現が得られるようになってきています。DOMによってWebページ中の要素を特定し、その要素のCSSの属性値をスクリプト言語によって変化させることで、Webページをダイナミックに変更することが可能です。DOMは、IE5以降、NN6以降、Safariなどの主要なWebブラウザが対応しており、実装上も違いがありません。そのため、今後はDOMを用いたWebページの実装が主流となるでしょう。

## ■WebサーバサイドでのJavaScript

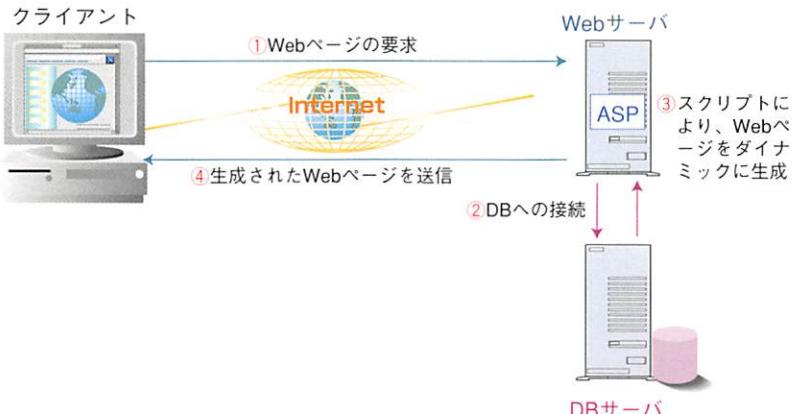
JavaScriptには、Webブラウザというクライアント環境で動作するイメージがありますが、実際にはWebサーバ側でもかなり使われています。Webサーバ側のスクリプト言語というと、すぐに思いつくのがCGIに使われているPerlです。Perlは、非常に強力なのですが、CGI上で使うという点と、さまざまな記述で同じ機能が実現できる柔軟性のため、書く人によってコードがバラバラになってしまう傾向があります。

こういった状況の中で、Webサーバ側でスクリプトを動作させてしまおうという製品がいくつか登場しています。たとえば、Microsoft社の「Active Server Pages(以下ASP)」といったものです。ASPIは、Webサーバである「Internet Information Server

「以下IIS」とともに無償で利用できるため、最近ではさまざまなWebページで使われるようになってきました。

ASPとは、HTMLファイル中にサーバサイドで動作するスクリプトを記述し、そのWebページの要求がきた時点でスクリプトを実行し、ダイナミックにページを生成して送り出すものです。このスクリプト記述用の言語として、VBScript、JScriptが標準で実装されています。

#### ▼ASPの動作概念図



ASPでは、データベースへの接続も非常に簡単に行うことができるので、インターネット構築などでは特に重要な役割を占めるようになっています。また、Microsoft社の.NET戦略の中のテクノロジーの1つである「ASP.NET」も、JScriptにより記述することができます。

## ■デスクトップへの展開

JavaScriptは、WWWの世界では、もはやなくてはならないものです。逆に、一般には、WWW上で使うものと認識されています。しかし、最近では、JavaScriptは思わずところで使われてたりします。たとえば、Windowsでは「Windows Scripting Host(以下WSH)」により、Windowsのスクリプト言語としてJavaScriptが使えるようになっています。

WSHとは、Windows上のスクリプト言語実行環境です。Webページ内でJavaScriptを用いるのと同様のことを、Windows上でも行うことができます。さらにWSHは、さまざまな組み込みオブジェクトを実装しており、Windows全体のコントロールも可能になっています。そのため、簡単なことならば、「Visual Basic(以下VB)」な

どでプログラムを作成しなくても、WSHで実現できます。

WSHでは、IE上でのスクリプト言語の実装状況と同じように、ActiveX Scriptingに対応するスクリプト実行エンジンが用意されているスクリプト言語ならば動作します。標準では、VBScript、JScriptが組み込まれています。なお、WSHは、Windows 98以降のバージョンに搭載されています。

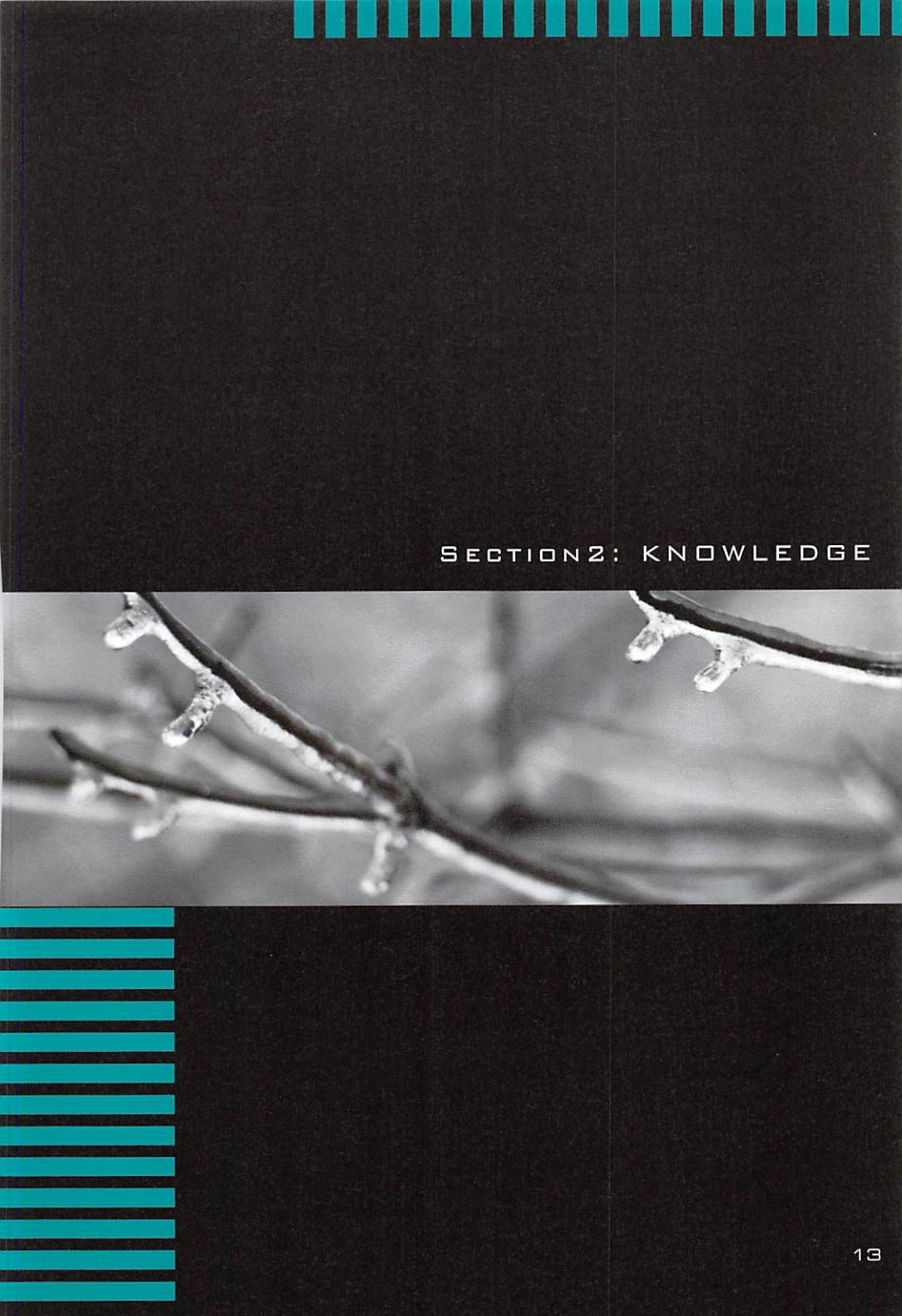
## ■そして、すべての環境へ

スクリプト言語は、Webブラウザからクライアント環境へと、使用範囲を広げてきました。そして今後、さらにその範囲を広げようとしています。Windowsの世界では、WSHにより、JScriptなどのスクリプト言語でWindowsのすべてをコントロールできるようになっています。たとえば、スクリプト言語で「ADSI(Active Directory Service Interface)」や「WMI(Windows Management Interface)」を操作すると、ユーザー管理、マシン管理、システム管理などといったシステム管理のほとんどが実現できるようになります。

そして、Microsoft SQL Server7.0/2000でも、データ操作言語などにスクリプト言語が使えるようになっているため、データベースの分野においてもJavaScript(正しくはJScript)が使われています。

さらにJavaScriptは、「Webサービス」を実現する際にも、重要な役割を担うことになるはずです。Webサービスとは、クライアントからの要求に対して、単にWebページを返すだけでなく、その問い合わせに対して何らかの処理をした結果を返したり、サービスを行ったりするものです。Webサービスは、XMLをメッセージングのフォーマットとして使い、システム間の連携を容易にします。Webサービスを実現する技術にはさまざまなものがありますが、先ほどのASPやASP.NETのように、JavaScriptで記述できるもののが多数あります。

このようにスクリプト言語、その中でもJavaScriptは重要な存在であり、いずれすべてのコンピューティング環境で使われることになるでしょう。



SECTION 2: KNOWLEDGE

# JavaScriptの組み込み

このセクションでは、JavaScriptの記述にあたって必要となる情報を、わかりやすく解説しています。まずは、JavaScriptの組み込みから始めましょう。

## ■JavaScriptを組み込む方法

Webページ(HTMLファイル)にJavaScriptを組み込む方法は、非常に簡単です。JavaScriptにかぎらず、HTMLファイルにスクリプトを組み込むときには、以下のように<script>タグを用いて記述します。

```
 <script type="使用する言語">
  スクリプト文
</script>
```

組み込むスクリプト言語がJavaScriptなら、次のように記述します。

```
<script type="text/javascript">
  スクリプト文
</script>
```

上記のように、type属性を用いてスクリプト言語を指定します。なお、以前よく用いられていたlanguage属性は、HTML4.01では非推奨属性となっており、将来はなくなる予定です。そのため本書では、特別な事情がないかぎり、type属性を用います。

参考までに、language属性を用いた記述方法も見ておきましょう。

```
 <script language="JavaScript">
  スクリプト文
</script>
```

language属性では、JavaScriptのバージョンを明示的に指定することもできます。たとえば、JavaScript1.3であることを示すには、以下のように記述します。

```
<script language="JavaScript1.3">
  スクリプト文
```





```
</script>
```

これに対してtype属性では、バージョンを指定することはできません。

## ■JavaScriptを書くときのルール

JavaScriptのスクリプトの記述には、いくつかのルールがあります。特に覚えておく必要があるのが、次の2つです。

- ・「;」が命令文の区切りとなる
- ・大文字と小文字を区別する

以下で、それについて解説します。

### ●JavaScriptでの1命令文

VB (Visual Basic)などでは、1行につき1つの命令文といった規則がありますが、JavaScriptにはそのような制限はありません。しかし、何の規則もない、どこからどこまでが1つの命令文なのか区別がつかなくなってしまいます。そこで、「;」(セミコロン)を命令文の区切り文字として使います。

JavaScriptでは、複数の命令文を1行に記述してもかまいません。逆に、1つの命令文を複数の行に分けて記述してもかまいません。ただし、このときに気をつけなくてはならないのは、メソッド名やプロパティ名、変数名などの単語の途中で改行をしてはいけないということです(これらがどんなものかは、後述します)。

以下、具体例を見てみましょう。

```
document.write("Hello");
alert("Hello");
```

上記の記述を1行に書き換えると、次のようにになります。

```
document.write("Hello"); alert("Hello");
```

逆に、1行で記述できる命令を、複数の行に分割することもできます。

```
win1>window.open('', 'newwin', 'toolbar=no');
```

SUMMARY

KNOWLEDGE

CHECK

WINDOW

DIALOG

PAGE

IMAGE

DATE/TIME

SUPPORT

FORM MAIL

TRICK

COOKIE

DATABASE

APPENDIX

上記の命令は、以下のように記述することもできます。

```
win1=window.open(' ', 'newwin',
'toolbar=no');
```

### ●大文字と小文字

JavaScriptで用いる変数名やキーワード、メソッド名やプロパティ名は、それぞれ大文字・小文字を区別します。そのため、大文字・小文字が間違っていると、エラーとなります。

Basic系の言語やPerlなどの、大文字と小文字の区別をしなくてもよい言語に慣れている人は注意してください。

## ■JavaScriptを記述する位置

JavaScriptを実装するための<script>タグは、HTMLファイルのどの位置に記述すればよいのでしょうか？ 結論からいうと、特に決まった位置はありません。

<script>タグは、HTMLファイル中に何回でも記述することができるので、好みに応じて使うことができます。多くのWebページでは、<head>タグの中に配置しているようです。しかし、フォームボタンなどに関連付けられたスクリプトは、それぞれの要素の近くに記述するとメンテナンスがしやすいといったこともあります。

ただし、1つ気をつけなくてはならないのは、スクリプトの呼び出しよりも先に、スクリプトの本体を記述しておく必要があるということです。

### ▼スクリプトの呼び出しよりも先に、スクリプトの本体を記述しておく

```
<html>
<title>alert setTimeout</title>
<script type="text/javascript">
<!--
    function alertTimer() {
        alert("Hello JavaScript!");
    }
//-->
</script>
<body onload="setTimeout('alertTimer()', 5000)">
```

呼び出しそれよりも前に  
記述しておく

呼び出し

たとえば、WebブラウザがWebページを読み込むと同時に動作させたいスクリプトは、<body>タグの onload 属性による呼び出しそりも先に記述しなくてはなりません。

## ■コメント

JavaScriptのスクリプトの中には、コメント(解説)を入れることができます。スクリプトのどこで何を行っているのか、スクリプト内にコメントを入れることにより、メンテナンスが容易になります。コメントを入れる方法は、次の2種類です。

- // コメント
- /\* コメント \*/

まず、「//」によりコメントを記述する方法を見てみましょう。この「//」は、記述した部分から行末までをコメントとして扱います。つまり、1行のコメントを入れるときに使います。C++やJavaなどで使われるコメントの実装と同じです。

```
<html>
<title>コメントの実装1</title>
<body>
<script type="text/javascript">
<!--
// この行はコメントとして扱われます。
document.write("Hello JavaScript World!");
//-->
</script>
</body>
</html>
```

次は、「/\*」と「\*/」でコメントを囲む方法です。囲んだ範囲がコメントとして扱われる所以、間に改行を入れることも可能です。これは、主にC言語などで使われるコメントの実装と同じです。

```
<html>
<title>コメントの実装2</title>
<body>
```

```
<script type="text/javascript">
<!--
/*複数行に
 わたるコメントにも
 対応します。 */
document.write("Hello JavaScript World!");
//-->
</script>
</body>
</html>
```



## ■外部ファイルを組み込む方法

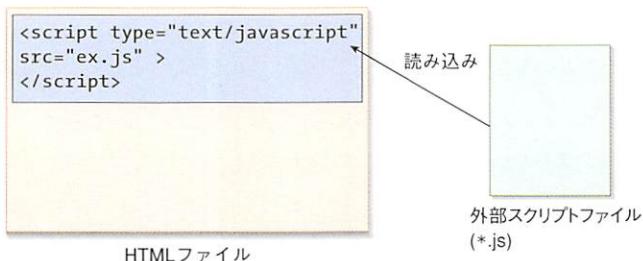
ここまで、HTMLファイルに直接JavaScriptを記述する方法について述べてきました。この方法は、記述性、メンテナンス性に優れているのですが、場合によっては不都合が生じこともあります。

たとえば、次のようなときは、できればHTMLファイルに直接JavaScriptを記述したくありません。

- ・非常に長いスクリプト
- ・JavaScriptのコードを簡単には見せたくない
- ・さまざまなHTMLファイルで同じスクリプトを使い回す

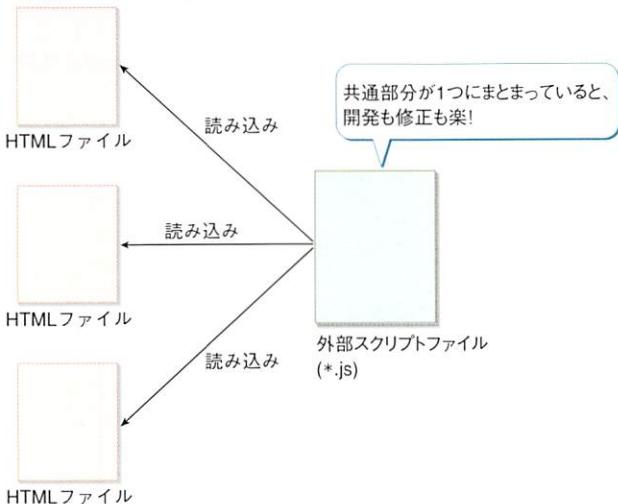
そのためJavaScriptでは、スクリプトだけを書いた外部ファイルを用意し、それを読み込んで実装する方法も用意されています。

### ▼外部ファイルでのJavaScriptの実装



外部ファイルとしてスクリプトを作成すると、1つのスクリプトをいくつものHTMLファイルで共有できるようになります。このようにしておくと、何度もスクリプトを作る必要もなく、また、後になって機能を変更した場合に、1つのファイルを修正するだけでよくなります。そのため、複数のHTMLファイルで共通で使うようなスクリプトは、外部ファイルとして実装するほうがよいでしょう。

▼複数のHTMLで共通のスクリプトを使うときなどに有効



外部ファイルの読み込みには、`<script>`タグの`src`属性を用います。値は相対パス、もしくは絶対パスで指定します。

```
 <script type="text/javascript" src="*****.js">
    </script>
```

なお、JavaScriptのコードを書いた外部ファイルの拡張子は、必ず「.js」としなくてはなりません。

### ●組み込みのサンプル

ここで、外部ファイルを組み込むサンプルを見てみましょう。外部スクリプトファイルを呼び出すHTMLファイルは、次のようにになります。

```
<html>
<title>外部ファイルでの実装</title>
```

```
<body bgcolor="#fff8dc">
<h3>外部ファイルを呼び出す</h3>
<hr />
<script type="text/javascript" src="ex_write.js"></script>
</body>
</html>
```

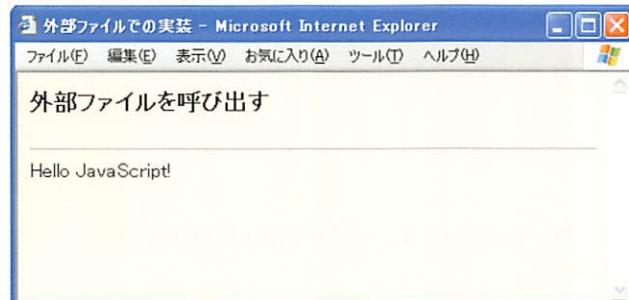


そして、呼び出されるJavaScriptのほうは、「ex\_write.js」という名前で、次のようなコードを記述したファイルを作ります。

```
document.write("Hello JavaScript!");
```

このサンプルを表示させると、次のようになります。

#### ▼外部ファイル組み込みのサンプル



#### ●外部ファイル組み込みの注意点

外部ファイルによるJavaScriptの実装は、JavaScriptをサポートしていないWebブラウザから、スクリプトを自動的に隠せるといったメリットがあります。また、HTMLファイルにはスクリプトを記述しないので、HTMLファイルのソースを見ただけではJavaScriptのコードがわからないようにすることができます。しかし、外部ファイルはいったんWebページをブラウジングしているユーザーのPC上にダウンロードされるため、コードを完全にユーザーから隠せるわけではありません。

#### ■エラーについて

JavaScriptを組み込んだWebページ内に、JavaScriptの文法のエラーがあったときはどうなるのでしょうか？ その場合、次のような警告ウィンドウが表示されます。

## ▼IEでのエラーメッセージ



エラーの表示は、Webブラウザによって異なります。Webブラウザによっては、エラーが存在すると思われる行が表示されることもありますし、ステータスバーにエラーがあるとだけ表示されるものもあります。エラーと思われる行が表示されているときは、その行までにエラーがあることを示しているので、その行までを注意深く確認することで、エラーを素早く発見できます。

## ■JavaScriptに対応していないWebブラウザのための記述

JavaScriptは、現在の主要なWebブラウザのほとんどで動作します。しかし、Webブラウザの中にはJavaScriptが動作しないものもありますし、家電やPDAなどに組み込まれているWebブラウザには、JavaScriptが動作しないものが多数存在します。Webページを作成するときには、そのようなJavaScriptが動作しないWebブラウザにも配慮してください。

JavaScriptが動作しないWebブラウザのためにできる対応は、次の2つがあります。

- ・スクリプトのコメント処理
- ・<noscript>タグの記述

### ●スクリプトのコメント処理

JavaScriptが動作しないWebブラウザには、「スクリプト行をコメントのように見せる」という方法を用います。サンプルは、次のとおりです。

```
<html>
<title>文字を表示する コメント対応</title>
<body>
<script type="text/javascript">
<!--
    document.write("Hello JavaScript World!");
//-->
</script>
</body>
</html>
```

HTML文では通常、注釈宣言(<!-- コメント -->)を用いてコメントを記述しますが、これを応用して、JavaScriptに対応していないWebブラウザではスクリプト全体をコメントとして認識させるようにします。HTML文のコメントの指定と異なるのは、コメントの終了部の前に「//」が入っていることです。これがないと、JavaScriptの実行エンジンは「-->」を解釈できない命令として認識し、エラーとなってしまいます。「//」をつけることで、この行はJavaScriptでのコメント行として認識されます。

以上をまとめると、JavaScriptをWebページに実装するときには、次のような形で組み込むことになります。

<script type="text/javascript">
<!--
 スクリプト
//-->
</script>

JavaScriptの動作するWebブラウザは、<script>タグを認識してスクリプトを実行しますが、<script>タグを認識できないWebブラウザは、コメントタグに囲まれた部分を解釈せず、スクリプトを実行しません。

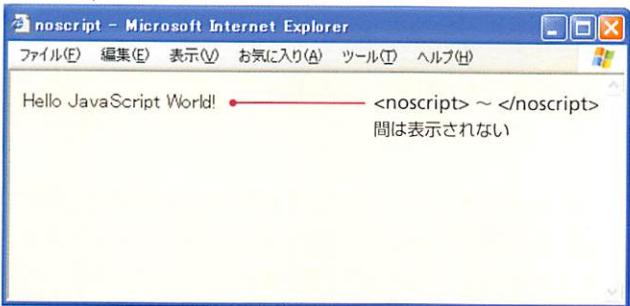
### ●<noscript>タグ

スクリプトをコメント行で囲むことにより、JavaScriptに対応していないWebブラウザがスクリプトの処理を行わないようになりました。しかし、あくまで何の処理も行わないようにしただけです。そこから一歩進んで、JavaScriptが動作しないWebブラウザに対しても、代わりのメッセージを表示するなど、何らかの配慮をしておく必要があります。

このときに用いるのが、`<noscript>`タグです。`<noscript> ~ </noscript>`の間に記述した内容は、JavaScriptに対応したWebブラウザでは無視されます。次の記述をJavaScript対応Webブラウザと非対応Webブラウザで表示してみましょう。

```
<script type="text/javascript">
<!--
    document.write("Hello JavaScript World!");
//-->
</script>
<noscript>
    Sorry... This page uses JavaScript.<br />
    <a href="../noscript.html">Please click here.</a>
</noscript>
```

#### ▼JavaScript対応ブラウザでの表示



#### ▼JavaScript非対応ブラウザでの表示



代替ページへのリンク

# JavaScriptで扱えるデータ

スクリプトでは、さまざまな情報(データ)を扱います。それはたとえば、文字列、数字といったものです。JavaScriptでは、情報を次の4つの形式に分類して扱います。

## ▼JavaScriptで扱えるデータ型

型	例
数値	10、0.1、10e3など
文字列	"Hello"、"Script"など
論理値	true(真)またはfalse(偽)
null	null値。何も入っていないことを表す特別なキーワード

ほかのプログラミング言語などに比べると、扱えるデータの型が少ないのですが、あまり高度で複雑な処理をしないかぎり、問題となることはないでしょう。

## ■数値

JavaScriptの数値型には、整数と浮動小数点数があります。

### ●整数

「整数」とは、小数点以下を持たない数値のことです。整数には正と負があり、表現できる最大の整数は、動作させるOSやWebブラウザによって異なります。また、JavaScriptでは、10進数、8進数、16進数の3種類の整数を扱うことができます。例としては、次のようなものがあります。

## ▼正数の例

例(整数)	種類
123	10進数(正)
-123	10進数(負)
056	8進数
0x7F	16進数

### ●浮動小数点数

「浮動小数点数」とは、小数点以下を持ち、小数の表現が浮動小数点により表されているものです。浮動小数点数にも正と負があり、さらに小数点以下の精度はOSやWebブラウザにより異なります。浮動小数点数では、小数点以下を持った10進数、またはべき乗

の識別子(E、e)と指数を持った10進数を取り扱えます。

▼浮動小数点数の例

例(浮動小数点数)	種類
1.23	10進数(正)
-1.23	10進数(負)
1e3	べき乗の識別子と指数を持った10進数

## ■文字列

「文字列」とは、0文字以上の文字のことをいいます。文字列は「"(ダブルクオーテーション)または「'」(シングルクオーテーション)で囲んで記述します。

▼文字列の例

例(文字列)	種類
"Hello"	ダブルクオーテーションで囲む
'JavaScript'	シングルクオーテーションで囲む
"234"	数字を囲む
""	空文字

ダブルクオーテーションやシングルクオーテーションで囲むと、数字であっても文字列として扱われます。また一番下の、囲まれた文字のない「""」は、「空文字」と呼ばれるものです。

## ■論理値とnull値

「論理値」は、「true(真)」または「false(偽)」のどちらかを持っている値のことをいいます。これは、データの比較や判定などに用います。JavaScriptでは、C言語と異なり、「1」「0」といったものは論理値として扱いません。

「null値」は、「何もない」ことを表す特別な値です。たとえば、定義されていない変数を参照しようとしたときなどに、null値が返されます。null値は「何もない」という値を持っているため、文字列の説明で出てきた空文字や、数値の0とは異なります。JavaScriptでは、null値は「null」と表現します。

# 変数

SUMMARY

KNOWLEDGE

CHECK

WINDOW

DIALOG

PAGE

IMAGE

DATE/TIME

SUPPORT

FORM MAIL

TRICK

COOKIE

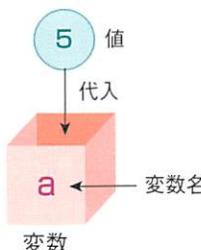
DATABASE

APPENDIX

## ■変数の生成

「2-2 JavaScriptで扱えるデータ」で解説したデータを扱うには、「変数」を用意し、それにデータを入れるという手段を取ります。変数には名前があり、どんなデータが入っているかは、その名前で参照できます。すなわち、変数は、「ある値を入れることのできる、名前のついた箱」と考えられます。変数に、ある値を入れることを「代入」といいます。

▼変数に、ある値を入れるのが代入



JavaScriptの変数は、どんな型のデータでも格納することができます。

変数を作ることを「変数の宣言」と呼びます。変数の宣言は、「var」キーワードの後に変数名を記述して行います。

var 変数名;

varは、「variable(変数)」という意味です。実際に用いると、次のようにになります。

```
var sample;
```

これで、まだ値を持っていない「sample」という名前の変数が作成されます。

変数の宣言と同時に値を代入するには、次のように記述します。

```
var sample="Hello";
```

上記のように記述すると、「sample」という名前の変数を作成し、その中に「Hello」という値(文字列)を代入します。

複数の変数をまとめて宣言することもできます。

var 変数名1, 変数名2, ……;

また、JavaScriptでは、ある変数を初めて使ったときに、その変数の宣言が行われたと解釈するようになっています。そのため、varによる変数の宣言を行わず、次のように記述したとしても、同様の結果が得られます。

sample="Hello";

変数に値を代入するには等号(=)を使います。これは、代入演算子と呼ばれます(p.30参照)。

## ■変数の名前

スクリプト中では、さまざまな処理を行うために、複数の変数を用います。そのため、JavaScriptでは、それぞれの変数を名前で区別します。変数に名前をつける際には、次の4つに気をつけなくてはなりません。

- ・大文字と小文字を区別する
- ・アルファベット、数字、アンダースコア(\_)で構成される
- ・1文字目はアルファベットか、アンダースコア(\_)
- ・予約語を使わない

### ●大文字・小文字の区別と文字構成

p.16で述べたとおり、JavaScriptでは、大文字と小文字を区別します。このルールは、UNIXの世界では当たり前のことですが、大文字と小文字の区別のないWindowsに慣れていると、ちょっと面倒に感じるかもしれません。

大文字と小文字が区別されるので、次の3つの変数名は、それぞれ別のものとして認識されます。

- ・SAMPLE
- ・Sample
- ・sample

変数は、アルファベット、数字、アンダースコア(\_)を用いて構成します。ただし、変

数の1文字目だけは、必ずアルファベットかアンダースコア(\_)を用いるという点に注意してください。

## ●予約語

JavaScriptでは、いくつかのキーワードを、JavaScriptそのものが使うために予約しています。これを「予約語」といいます。予約語は、変数名、関数名には使えません。

### ▼予約語一覧

break	case	catch	continue	default	delete	do
else	finally	for	function	if	in	instance
of	new	return	switch	this	throw	try
typeof	var	void	while	with		

また、ECMA-262では、将来予約語になる可能性のあるキーワードとして、次のものを挙げています。これらも、変数名や関数名に使わないほうがよいでしょう。

### ▼将来の予約語一覧

abstract	boolean	byte	char	class
const	debugger	double	enum	export
extends	final	float	goto	implements
int	interface	long	native	package
private	protected	public	short	static
super	synchronized	throws	transient	volatile

## ■配列

配列とは、複数個の変数をひとまとめとして扱えるようにしたものです。配列は、変数に添え字がついたものとして表されます。配列の宣言は、次のようにになります。

var 変数=new Array(配列の数);

たとえば、以下のように宣言すると、5個の値を代入することができる「a」という配列が生成されます。

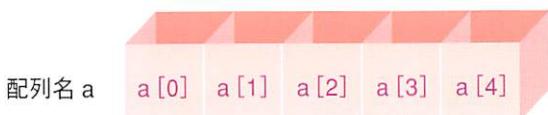
var a=new Array(5);

変数の宣言と同様に、varを省略してもかまいません。

```
a=new Array(5);
```

この配列の要素は、a[0]、a[1]、a[2]、a[3]、a[4]の5つです。p.26の変数の図で考えると、次のように5つの箱があることになります。

#### ▼配列のイメージ



気をつけなくてはならないのは、配列の一番最初の添え字は「0」から始まるということです。これは、非常に間違いやすいので注意してください。

なお、あらかじめ長さを指定せずに、配列を生成することもできます。

var 変数=new Array();

また、配列の宣言と同時に値を代入することもできます。

var 変数=new Array(配列の要素);

具体的には、次のように記述します。

```
var day=new Array("月", "火", "水", "木", "金", "土", "日");
```

配列は、複数の変数をまとめて処理する場合に便利です。また、配列をマスターすると、スクリプトの制作効率が上がります。ここでの説明だけではわかりにくいかかもしれません、SECTION3以降のサンプルの中で実際の利用方法と解説が登場しますので、楽しみにしていてください。

## ■変数の処理

変数に、ただ値を入れただけでは、あまり意味がありません。変数に対して何らかの処理を加えることで初めて、プログラムに役立つ結果が得られるのです。ここでいう処理とは、「演算子(Operator)」を用いて、何らかの計算を行うことです。演算子には、「代入

演算子」「算術演算子」「比較演算子」「論理演算子」「条件演算子」「文字列連結演算子」といったものがあります。JavaScriptの演算子の種類や使用方法は、JavaやC言語の演算子とほとんど同じです。

なお、「ビット演算子」というものもありますが、本書では扱いません。

## ●代入演算子

代入演算子は、ある値を変数に代入するために使います。基本的には、代入演算子の右側に記述されている値や式の結果などを、左側に記述されている変数に代入する処理を行います。以下に、主な代入演算子を示します。

### ▼代入演算子

例	内容
<code>a = b</code>	<code>a</code> に <code>b</code> を代入する
<code>a += b</code>	<code>a</code> に <code>a+b</code> を代入する
<code>a -= b</code>	<code>a</code> に <code>a-b</code> を代入する
<code>a *= b</code>	<code>a</code> に <code>a*b</code> を代入する
<code>a /= b</code>	<code>a</code> に <code>a/b</code> を代入する
<code>a %= b</code>	<code>a</code> に <code>a/b</code> の剰余を代入する

## ●算術演算子

数値計算を行う演算子です。和(+)、差(-)、積(\*)、商(/)を求める演算子のほかに、割り算の余りを求める剰余演算子(%)があります。

### ▼算術演算子

例	内容
<code>a + b</code>	<code>a</code> と <code>b</code> を足す
<code>a - b</code>	<code>a</code> から <code>b</code> を引く
<code>a * b</code>	<code>a</code> と <code>b</code> を掛ける
<code>a / b</code>	<code>a</code> を <code>b</code> で割る
<code>a % b</code>	<code>a</code> を <code>b</code> で割った余り

なお、商演算子で割る数(b)を「0」にすると、演算結果には無限大(Infinity)が返されます。さらに、剰余演算子で割る数(b)を「0」にすると、その剰余は計算できないため「NaN」(Not a Number)が返されます。これらは、計算上でトラブルを引き起こす可能性があるので、気をつけてください。

また、上記の演算子以外にも、インクリメント演算子(++)、デクリメント演算子(--)、単項マイナス(-)という演算子があります。

## ▼インクリメント演算子、デクリメント演算子、単項否定

例	内容
<code>a = b++</code>	aにbを代入してから、bの値を1増やす
<code>a = ++b</code>	bの値を1増やしてから、aに代入する
<code>a = b--</code>	aにbを代入してから、bの値を1減らす
<code>a = --b</code>	bの値を1減らしてから、aに代入する
<code>a = -b</code>	aにbの符号を反転させたものを代入する

## ●比較演算子

比較演算子は、右側と左側の値を比較するときに用います。比較演算子を使った式は、条件式として用います。主な比較演算子を以下に示します。

## ▼比較演算子

例	内容
<code>a == b</code>	aとbが等しいときに真
<code>a != b</code>	aとbが等しくないときに真
<code>a &gt; b</code>	aがbよりも大きいときに真
<code>a &gt;= b</code>	aがb以上とのときに真
<code>a &lt; b</code>	aがbよりも小さいときに真
<code>a &lt;= b</code>	aがb以下のときに真
<code>a === b</code>	aとbのデータ型と値がともに等しいときに真
<code>a !== b</code>	aとbのデータ型と値のどちらかが等しくないときに真

なお、上の表の中にある「`==`」と「`!=`」は、JavaScript1.3で追加された演算子です。これらは、比較を行う前に、それぞれのデータ型を自動的には変換しません。そのため、「`a==b`」が真となるのは、aとbのデータ型が同じで、さらにaとbの値が同じときとなります。また、「`a!=b`」は、aとbのデータ型または値のどちらかが異なっていても真となります。そのため、aとbの値が同じであっても、データ型が異なる場合には真となります。

JavaScript1.2では、比較演算子「`==`」と「`!=`」、「`==`」と「`!=`」が同様の働きをします。しかし、これ以外のバージョンのJavaScriptは、「`==`」と「`!=`」はデータ型の比較を行いません。これは、Netscape社がJavaScript1.2において変更した仕様を、その後に作成されたECMA Scriptの仕様に合わせて、元に戻したためです。なお、明示的にJavaScript1.2として動作させるには、language属性を用いて次のように指定する必要があります。

<script language="JavaScript1.2">  
 .....  
 </script>

現在配布されているWebブラウザでJavaScript1.2として動作するのは、Netscape Navigator4.x以上とMozillaです。これ以外のWebブラウザは、上記のような仕様にはなっていません。

### ●論理演算子

論理演算子は、右側と左側の条件式を評価して、それぞれ成り立っているかどうかで true(真)、false(偽)を返します。これは、複数の条件式を評価するために、よく用いられます。論理演算子を以下に示します。

#### ▼論理演算子

例	内容
条件式a && 条件式b	aとbが成り立っていたら真(AND)
条件式a    条件式b	aとbのどちらかが成り立っていたら真(OR)
!条件式a	aが真のときには偽、aが偽のときには真(NOT)

### ●条件演算子

条件演算子は、これまで見てきた演算子とは少し異なります。これは、条件式と2つの値から構成され、2つの値のうちのどちらかが返されます。書式は、以下のようになります。

(条件式) ? 値1 : 値2

条件式の部分は、評価すると論理値(trueまたはfalse)が返るような式にします。条件式が真ならば、この式全体が返す値が「値1」に、偽ならば「値2」になります。以下に具体的なコードを挙げます。

```
<html>
<title>true or false</title>
<body bgcolor="#ffff8dc">
<h3>条件演算子</h3>
<hr />
<script type="text/javascript">
```



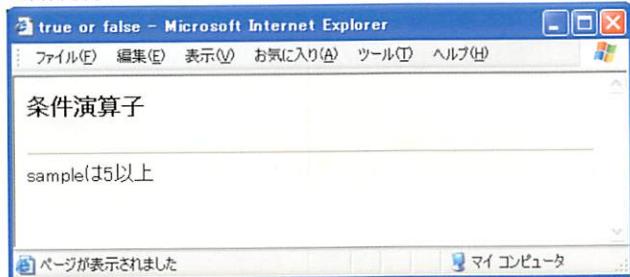
```

<!--
var sample=10, tmp;
tmp=(sample>=5) ? "sampleは5以上" : "sampleは5より小さい";
document.write(tmp);
//-->
</script>
</body>
</html>

```

上記のコードの場合、変数sampleの値を評価し、その結果が真であるため、「sampleは5以上」と表示されます。

#### ▼条件演算子のサンプル



### ●文字列連結演算子

文字列連結演算子(+)は、文字列同士を連結する演算子です。書式としては、次のようにになります。

文字列1 + 文字列2

上記のように記述すると「文字列1」と「文字列2」が連結されます。実際の記述は、以下のようにになります。

```
chars="Hello +"JavaScript";
```

上記のコードでは、変数charsに「Hello JavaScript」という連結された文字列が代入されます。

## ●演算子の優先順位

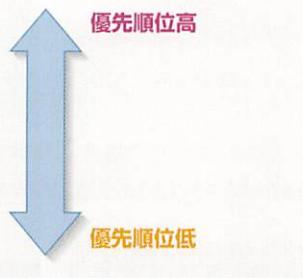
演算子は、同時に用いたときにどちらが先に実行されるかという順位が決まっています。

複雑な式を記述するときには、この優先順位を意識する必要があります。

演算子の優先順位は、以下のとおりです。

### ▼演算子の優先順位

1. 括弧 `()`
2. インクリメント `++`、デクリメント `--`
3. 積 `\*`、商 `/`、剰余 `%`
4. 和 `+`、差 `-`
5. 比較演算子 `<`、`<=`、`>`、`>=`
6. 等価演算子 `==`、`!=`、`====`、`!=!=`
7. 論理積 `&&`
8. 論理和 `||`
9. 条件演算子 `?`
10. 代入演算子 `=``+=`、`-=`、`\*=`、`/=`、`%=`



## ●演算の順序の制御

演算子が実行される順序は、通常の数学の計算と同様に、乗算、除算が加算、減算よりも優先されます。加算、減算を乗算、除算よりも先に計算したい場合には「`(`」(括弧)で囲みます。また、剰余演算は乗算、除算と同じ優先順位です。たとえば、次のような計算があったとしましょう。

```
result=3*(4+5);
```

この場合、括弧で囲んでいる「`4+5`」を先に計算して、その結果に「`3`」を掛けることになります。

また、複数の括弧があったときには、一番内側の括弧が一番優先され、順々に外側の括弧の処理が行われます。

## 条件によって処理を変える

p.29からの「変数の処理」において、ある式(比較式など)が条件式の条件部分になると説明しました。また、条件演算子を使えば、条件の真、偽という2つの値のどちらかが返されるという簡単な条件選択を実現できます。しかし、これらの方法では、上から下へと流れる直線的なプログラムの流れを変えることはできません。

そこで、ここでは条件によって処理を変えたり、繰り返しの処理を行ったりする方法を解説します。ここまで覚えたことと組み合わせて、よりプログラマらしいスクリプトを記述してみましょう。

### ■条件分岐(if～else文)

if～else文(条件文)を式と組み合わせて使うと、条件に応じて処理を分岐させ、プログラムの流れを変更することができます。if～else文には、何パターンかの記述方法があります。まず、一番簡単な、ifを用いた文の記述方法から説明しましょう。

if(条件式)

処理；

このように記述すると、条件が真ならば処理を行い、偽なら処理を実行せずに次の行へ移ります。また、処理の部分は、中括弧({ })を用いて複数記述することもできます。

if(条件式)

{

処理1；

処理2；

……；

}

処理が1つならば、中括弧は省略してかまいません。if文を用いたサンプルは、次のようにになります。このサンプルでは、条件が真となるので、次ページの画面のように中括弧内の処理が行われています。

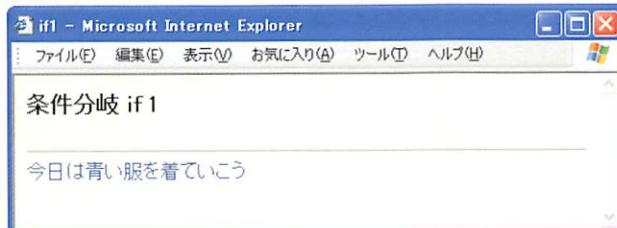
```
<html>
<title>if1</title>
```





```
<body bgcolor="#ffff8dc">
<h3>条件分岐 if 1</h3>
<hr />
<script type="text/javascript">
<!--
var myColor="blue";
if(myColor=="blue") {
    document.write("<font color='blue'>今日
    は青い服を着ていこう</font>");
}
//-->
</script>
</body>
</html>
```

#### ▼if文のサンプル



もしも、上記サンプルの8行目で変数myColorの値を「blue」ではなく「red」とすると、条件式が偽となってしまうため、何の処理も行われません。

このように、単純な分岐はifだけでも記述できますが、条件式が偽のときにも何らかの処理を行いたい場合は、次のようにelseも用います。

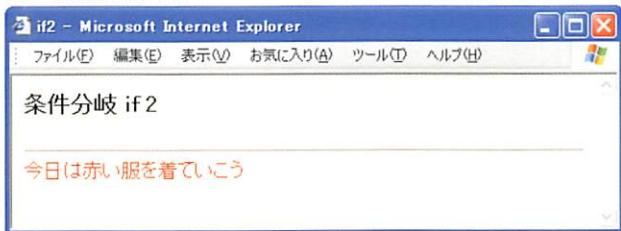
```
 if(条件式) {
    処理1; •———— 1つでも複数でもかまわない
}
else {
    処理2; •———— 1つでも複数でもかまわない
}
```

このように記述すると、条件が真のときには処理1が、偽のときには処理2が行われます。if～else文を用いたサンプルは、次のようにになります。

```
<html>
<title>if2</title>
<body bgcolor="#ffff8dc">
<h3>条件分岐 if 2</h3>
<hr />
<script type="text/javascript">
<!--
var myColor="red";
if(myColor=="blue") {
    document.write("<font color='blue'>今日
    は青い服を着ていこう</font>");
}
else {
    document.write("<font color='red'>今日
    は赤い服を着ていこう</font>");
}
//-->
</script>
</body>
</html>
```

このサンプルでは、条件式が偽となります。そのため、elseに記述されている処理が実行されます。実行結果は、次のようになります。

#### ▼if～else文のサンプル

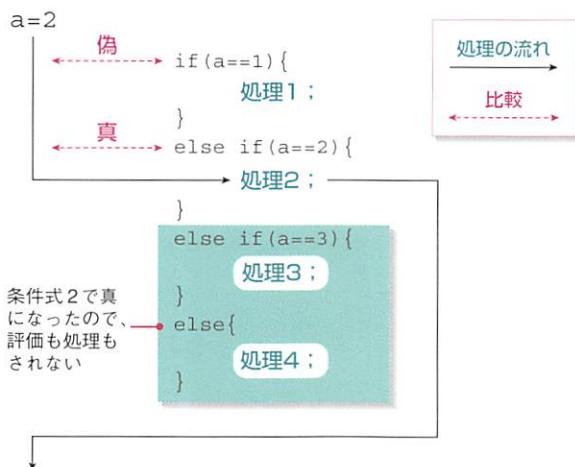


ただし現実には、もっとたくさんの分岐をしたいこともあります。その場合には、ifとelseを組み合わせて、さらに複数の条件式を評価できるようにします。

```
 if(条件式1) {
    処理1;
}
else if(条件式2) {
    処理2;
}
else if(条件式3) {
    処理3;
}
...
else {
    処理n;
}
```

このように記述すると、任意の数の条件を評価できるようになります。しかし、上記の記述を使う際に気をつけなくてはならないことがあります。それは、たとえば条件式1で真となると、処理1が実行された後、if～else文で書かれたコード全体を抜けてしまうことです。

#### ▼真になった時点で条件式を抜ける



この図でもわかるように、if～else文では、ある条件式で真になったら、その部分の処理を行い、それ以降の条件式は評価も実行もしません。

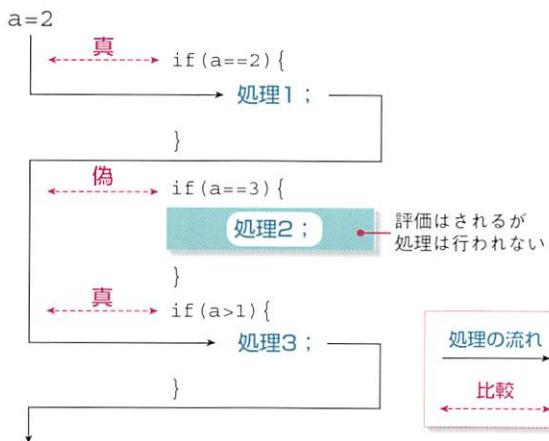
それでは、書いた条件式をすべて評価してほしい場合には、どうしたらよいのでしょうか？ 解決法は簡単です。次のように記述します。

```
 if(条件式1) {
    処理1;
}
if(条件式2) {
    処理2;
}
```

.....

このように記述すると、条件式をすべて評価し、真になった部分の処理をすべて行うことになります。

#### ▼記述する条件式をすべて評価する



## ■繰り返し処理

ここまでで、簡単なプログラミングを実現するif～else文を解説しました。しかし、これらはスクリプトの行が、それぞれ1回しか実行されませんでした(if～else文の場合には、実行されない行もあります)。しかし、プログラミングをしていると、同じ行(処理)を複数回繰り返したいという状況が発生します。これを実現するのが、繰り返し処理です。この処理を「ループ」と呼びます。また、ループは2種類あります。1つは、条件に基づいてその条件が満たされるまで繰り返すもの、もう1つは、設定された範囲を超えるまで繰り返すものです。

### ●for文を使ったループ

for文を使ったループは、プログラミングの世界では非常に基本的なもので、C言語をはじめ、Basic、Perl、Javaなどにも同様の仕組みが実装されています。

for文を使ったループは、基本的に指定した回数だけ処理を繰り返すために用いられることが多いです。

for文は、次のように記述します。

```
 for(初期状態; 終了条件; 繼続処理) {  
    処理;  
}
```

具体的には、次のような記述になります。

```
for(i=1; i<=10; i++) {  
    処理1;  
}
```

これは、初期状態は「iが1」、終了条件は「iが10になったとき」、繰り返しごとに「iに1を足していく」这样一个for文です。これにより、iが1から10になるまで「処理1」を10回繰り返すことができます。このとき、iのことを「カウンタ」と呼び、数を数えるのに用います。また、終了条件は、ループの最初に評価されます。継続処理は、ループ内の処理が終わって、次のループに入る前に実行されます。

では、for文を使ったサンプルを見てみましょう。

```

<html>
<title>for 1</title>
<body bgcolor="#fff8dc">
<h3>for文で繰り返し処理が行われます</h3>
<hr />
<script type="text/javascript">
<!--
for(i=1; i<=7; i++) {
    document.write("<font size="+i+">Hello
    JavaScript!</font><br />");
}
//--
</script>
</body>
</html>

```

このスクリプトを実行すると、次のように表示されます。

▼for文のサンプル



もしも、for文を使わずにこのWebページを書いたとしたら、非常に面倒なことになるのは容易に想像できると思います。ほかにもいくつか文例を挙げておきましょう。

▼iを0から始め、10より小さい間、iを1ずつ加えながら繰り返す

```
for(i=0; i<10; i++)
```

▼iを10から始め、20以下の間、iを1ずつ加えながら繰り返す

```
for(i=10; i<=20; i++)
```

▼iを10から始め、0より大きい間、iを1ずつ減らしながら繰り返す

```
for(i=10; i>0; i--)
```

▼iを0.0から始め、1.0以下の間、iを0.2ずつ加えながら繰り返す

```
for(i=0.0; i<=1.0; i=i+0.2)
```

## ●for文を使ったループでの注意

for文は、非常に便利なものですが、プログラミング次第では深刻なエラーを引き起こします。for文の後ろの「( )」の中に条件を書くのですが、ここで終了条件を誤ると、いつまでたってもループから抜けられず、スクリプトが終わらないということが起きます。これを「無限ループ」と呼びます。

無限ループは、プログラム上有効なこともあるのですが、意図しない場合には深刻なエラー以外の何ものでもありません。このようなエラーを避けるために気をつけなくてはならないのは、終了条件の記述の仕方です。終了条件で使える演算子は、次の4種類です。

▼終了条件で使える演算子

<	>	<=	>=
---	---	----	----

これ以外の演算子では、ほとんどエラーとなりますので、使わないようにしましょう。また、条件の設定は、必ず真になるものにしてください。

## ●for～in文を使ったループ

for文を使ったループは、一般的なものですが、JavaScriptではさらに拡張を行い、for～in文という仕組みを実装しています。for～in文を使ったループは、オブジェクトのすべてのプロパティに自動的にアクセスするために使われます（オブジェクトやプロパティについては、p.54以降で詳しく解説します）。

for～in文の記述は、以下のようになります。

```
 for(カウンタ in オブジェクト名) {  
    处理;  
}
```

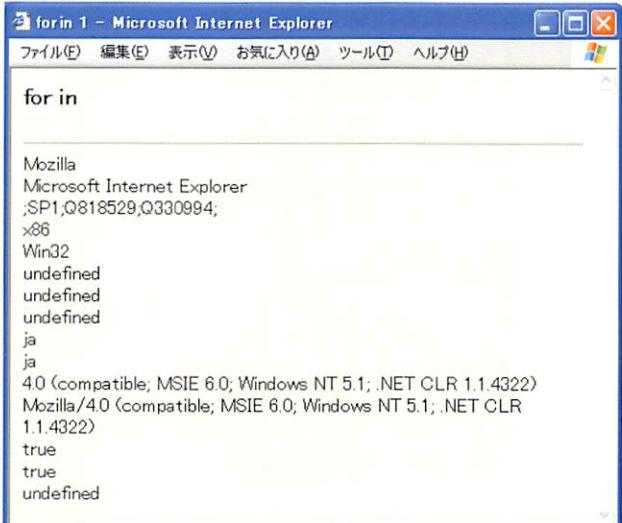
カウンタは、0からオブジェクトのプロパティがある間、1ずつ足されてループを行い

ます。サンプルは、次のようにになります。

```
<html>
<title>forin 1</title>
<body bgColor="#fff8dc">
<h3>for in</h3>
<hr />
<script type="text/javascript">
<!--
for(i in navigator) {
    document.write(navigator[i]);
    document.write("<br />");
}
//--
</script>
</body>
</html>
```

上記のスクリプトを実行すると、以下のようになります。このスクリプトでは、navigatorオブジェクトのプロパティ一覧を表示しています。

#### ▼IE6の表示



▼NN7での表示

for in

---

```
Win32
Mozilla
Netscape
5.0 (Windows; ja-JP)
ja-JP
[object MimeTypeArray]
Windows NT 5.1
Netscape
7.1
Gecko
20030624
[object PluginArray]

Mozilla/5.0 (Windows; U; Windows NT 5.1; ja-JP; rv:1.4)
Gecko/20030624 Netscape/7.1 (ax)
true
function javaEnabled() { [native code] }
function taintEnabled() { [native code] }
function preference() { [native code] }
```

▼Safariでの表示

for in

---

```
20030107
Gecko
[object PluginArray]
Netscape
Mozilla
[object MimeTypeArray]
[function]
5.0 (Macintosh; U; PPC Mac OS X; ja-jp) AppleWebKit/85 (KHTML, like Gecko) Safari/85
MacPPC
en
Mozilla/5.0 (Macintosh; U; PPC Mac OS X; ja-jp) AppleWebKit/85 (KHTML, like Gecko)
Safari/85
Apple Computer, Inc.
true
```

これは、スクリプト制作時にプロパティの一覧を表示したい場合や、プロパティの番号がわからないときなどに役に立ちます。なお、このスクリプトは、Webブラウザごとに表示結果が異なります。

### ●while文を使ったループ

繰り返し処理を実現するものとして、for文以外にも、while文によるループがあります。while文によるループは、ある条件が真である間、ループを繰り返すものです。while文の形式は、以下のようになります。

```
 while(条件) {
    処理;
}
```

条件には、論理値(trueまたはfalse)を返す条件式ならば、どのような式でも書くことができます。実際のサンプルは、次のようにになります。

```
<html>
<title>while 1</title>
<body>
<h3>while 1</h3>
<hr />
<script type="text/javascript">
<!--
var i=0;
while(i<=7) {
    document.write("<font size='"+i+">" +
    Hello JavaScript!</font><br />");
    i++;
}
//--
</script>
</body>
</html>
```

このサンプルは、「i」が0から7になるまで繰り返されます。「i=8」になった時点で

while文の条件式が偽となるので、ループから抜けます。これは、条件式だけで繰り返しをコントロールしたいときに役に立ちます。

### ●break文とcontinue文

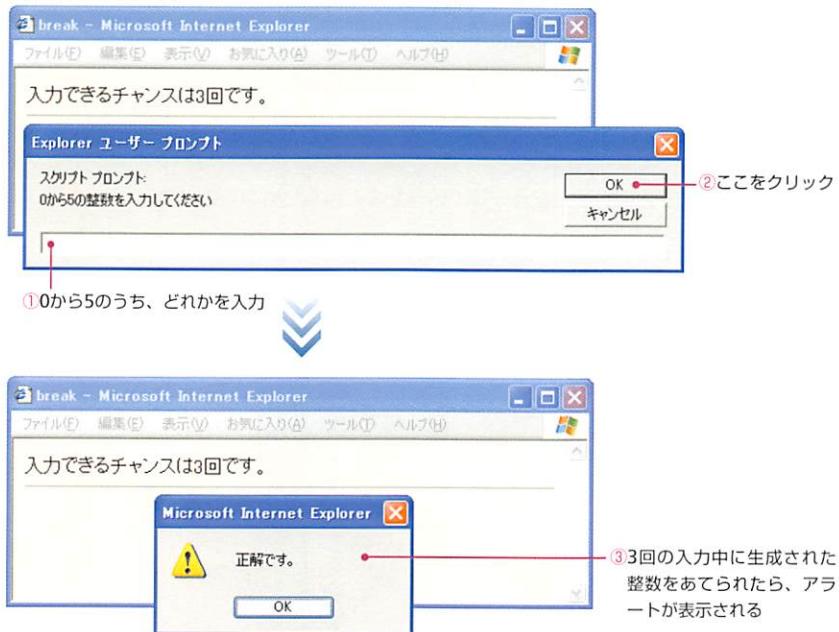
for文とwhile文によるループをより使いやすくするために、JavaScriptはbreak文とcontinue文を用意しています。これらの文は、for文やwhile文の持っている終了条件だけでなく、繰り返し処理中に何らかの条件が成立したらループの動作を変えるといった目的のために使われます。

break文は、ループが終了していないくとも、この命令が実行されるとループから抜け出すというものです。サンプルは、次のようにになります。

```
<html>
<title>break</title>
<body bgcolor="#ffff8dc">
 入力できるチャンスは3回です。
<hr />
<script type="text/javascript">
<!--
var correct=Math.round(5*Math.random());
for(i=0; i<3; i++) {
    answer=prompt("0から5の整数を入力してください","");
    if(answer==correct) {
        alert("正解です。");
        break;
    }
}
//-->
</script>
</body>
</html>
```

このスクリプトを実行した結果は、次のように表示されます。

## ▼break文のサンプル



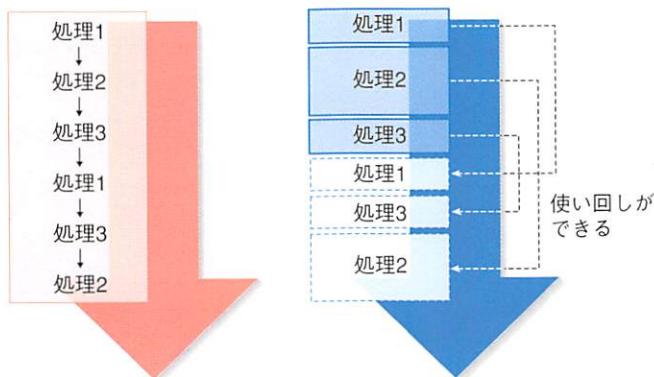
このスクリプトは、0から5の整数を生成し、その値は何かをあてるものです。整数の生成には乱数を使っているため、その値は毎回変わります。答えられる回数は、for文を使って3回に制限しています。プロンプトに入力した整数が正答だったときには、if文の条件が真となり、画面にアラートで「正解です。」と表示して、break文によってループを抜けて処理を終了します。

もう1つのcontinue文は、ループ中にこの命令を実行させると、そのループ処理を中断し、いったんループ処理の先頭に戻ります。そこで継続処理を行ってから、次のループに入ります。break文と異なるのは、ループを抜けてしまわないところです。

# 関数

プログラムやスクリプトを書いていくと、同じような処理が何度も出てくることがあります。同じ処理を何度も記述するのはとても面倒だし、効率も悪くなります。そんなときには、どうすればよいのでしょうか？この問題を解決するのが関数(ファンクション)と呼ばれる機能です。関数は、ある一連の処理をまとめたのです。

▼関数を使うと、同じ処理を何度も記述しなくても済む



たとえば、上の図のようにいくつかの処理が行われる中で、同じ処理が繰り返し出てくる場合、処理の流れに沿って毎回同じ処理を記述していると、非常に長いコードになり、コーディングにも時間がかかってしまいます。

そこで、処理単位ごとに関数を作つて呼び出せるようにしておくと、何度も同じ処理を記述する必要がなくなり、すっきりとしたスクリプトが作成できます。関数は、メソッド(p.62)などと同様に引数を取り、処理をした結果を返します。

## ■関数の定義

関数は「function」キーワードを用いて定義します。この定義には、関数の名前と関数に渡す引数リスト、関数内部で行われる処理を記述したスクリプトが必要となります。関数の定義は、次のように行います。

```
 function 関数名(引数リスト) {
    処理;
}
```