HTML5

スプリングコース

2015年3月9日版

中島俊治

はじめに

HTML

おはようございます。講師の中島です。

みなさんは、普段PCやスマートフォンで**ブラウザを起動**してインターネット上にあるWebサイトを見ていますよね。

Webサイトの各ページ内には様々な文章、画像、音声、映像、情報などの**コンテンツ**が並んでいます。

そのコンテンツを表示するのに**HTML**(ハイパーテキスト・マークアップ・ランゲージ：Hypertext Markup Language)という言語を使って、コンテンツの構造や配置を決めます。

最近はHTMLの他に、**CSS**(Cascading Style Sheet)という言語も使い、Webページをとても綺麗なデザインにすることができるようになりました。

むずかしそう・・

なんだかHTMLって面白そうだけど、むずかしいのでは？と思う人もいるでしょう。

でも大丈夫。HTMLは、普段使っているブラウザと、文字を入力するテキストエディタがあれば、自宅でも喫茶店でも気軽に作りながら勉強することができます。英文字や数字、日本語の文字を入力するスキルで充分なのです。

私は、無料ホームページ・バーチャルコミュニティサービスの**GeoCities JAPAN**でコミュニティマネージャをつとめていました。私と同年代はもちろん、お年を召した方から学生さんまで、男女区別なくホームページを作り、自らの意見や知識を世界に発信されています。

毎日こつこつと

コンテンツを表現するためにHTMLやCSSには様々なツールが用意されています。それらを覚えるのは一苦労でしょう。本書ではサンプルソースを用意し実際に作って試しながら理解できるように工夫しています。一朝一夕には無理ですが、毎日コツコツと本書をすすんでいくうちに自然と理解できます。

HTML5

2015年10月28日、とても大きな出来事がありました。

Webに関する技術的な仕様はW3C（ワールド・ワイド・ウエブ・コンソーシアム：World Wide Consortium）という標準化団体が決めています。HTMLのバージョンは4.01でした。

そのW3Cが10月28日にHTMLのあたらしいバージョン「HTML5」の勧告があり、その時からHTML5は標準規格となりました。

参考URL： http://www.w3.org/2014/10/html5-rec.html.ja

HTML5は、従来のホームページを作るための規格から大きく進歩し、HTMLのタグが整理されてコンテンツの構造が書きやすくなり、検索エンジンにも理解しやすくなりました。これを狭義のHTML5といいます。

CSSもグループに加わって**CSS3**となりデザインがより豊かになり、都度画像を画像処理ソフトで加工しなくてもリッチなデザインやコンテンツの変形、アニメーションや三次元表示が可能になりました。

さらにWebのスクリプト（スクリプトはプログラム言語の一つ）である、JavaScriptもグループに入り、HTMLのタグやコンテンツ、CSSをコントロールしたり、計算も行い、さらにスマートフォンだとタッチセンサや重力センサ、GPSなどの端末固有の機能にまでアクセスして、利用者と端末の間で双方向性のある楽しいWebアプリケーションの製作ができるようになりました。

狭義HTML5、CSS3、JavaScriptなどを一つのグループとして**広義HTML5**といいます。通常HTML5と言えば、この広義HTML5を指し、とても広い技術が含まれています。

10月28日に標準規格になったとはいえ、実は標準規格以前から、確定した機能は各ブラウザは取り入れを行っており、PCブラウザ上だけではなく、スマートフォンアプリ、ゲーム製作、テレビのハイブリッドキャスト、車のナビ、そしてスマートフォンのOSをHTML5で作れるなど、HTML5は様々な分野へ広がりがすでに起こっていて、HTML5の可能性は無限大です。

HTML5プロフェッショナル認定試験

特定非営利活動法人エルピーアイジャパン（LPI-Japan）では、2014年１月から「HTML5プロフェッショナル認定試験（レベル１）」を９月からレベル２の実施を開始しました。

レベル1では、マルチデバイスに対応した静的なWebコンテンツをHTML5を使ってデザイン・作成できることを目指し、「HTML5の要素（要素は、開始タグと終了タグのセットになったひとつのページ上の部品と考えてください）」と「CSS3」をメインに、「レスポンシブWebデザイン」「オフラインWebアプリケーション」などが出題されます。

レベル2では、最新のマルチ・メディア技術に対応した動的なWebコンテンツを設計・作成できることを目指し「JavaScript」や「API（アプリケーション・プログラム・インターフェース：Application Program Interface：ここでは端末などの様々な機能や収集するデータをHTML5で利用できるようにするしくみ）」に関するものが出題されます。これらの試験は、HTML5の範囲をすべて網羅していますから、HTML5の全体像を知るにはとても役立ちます。

試験概要　http://html5exam.jp/outline/

オフィシャルページ　http://html5exam.jp

試験の内容

これが一番気になるところですね。会場には、各自のブースにPCが備え付けられて、PC上で解答をするCBT方式です。多肢選択の知識問題、コードリーディング問題や、記述問題があります。多肢選択は一つだけ選ぶだけではなく様々なバリエーションが考えられます。記述問題はキーボードでの打込みになるので、注意しておきましょう。

試験時間は90分、問題は約60問、合否結果は試験終了とともにすぐに分かります。問題がかなり多めですから、時間配分のことも考えて落ち着いて解いていきましょう。

なお、再認定ポリシーというのがあり、合格者には５年間のACTIVE認定ステイタスが維持されます。5年のうちに同等か上位資格を受けることで再認定されます。

試験概要　http://html5exam.jp/outline/

再認定ステイタス　http://html5exam.jp/register/recert.html

学習するために

必要な物は、テキストエディタ（UTF−８の文字コードが扱えて、行番号が表示されているもの）とブラウザ（google chrome推奨）です。なお、PCの設定で拡張子は表示設定にしておいてください。インターネットにアクセスしたり環境設定に時間を掛ける必要はありません。どこででも勉強することができます。

講師：中島俊治

HTML5プロフェッショナル認定資格取得。放送大学、八洲学園大学、秋草学園短期大学でHTML5の講師をつとめる傍ら、みずから「東京アプリ・ワークショップ」を運営し、HTML5の楽しさを広める。HTML5アカデミック認定校として運営。マイクロソフトMVPアワード(2014/2015)受賞。
97年上京、ソフトバンク入社。GeoCities JAPAN、Yahoo! JAPANなどのネットベンチャーを経験し個人事業主「春翠工房」独立。現在は講師業をメインに日々奮闘中。

著作権

目次

はじめに 2
1.HTML5マークアップ要素 8
2.アプリに便利な要素 12
2-1.input要素の属性 12
2-2.バリデート・値の制限など 13
2-3.マルチメディア 13
2-4.進捗・測定 13
2-5.擬似クラス・擬似要素 14
3.CSS3 16
3-1.角丸 16
3-2.影 17
3-3.変形 18
3-4.CSSトランジション 19
3-5.CSSアニメーション 20
3-6.３次元 21
3-7.3次元のアニメーション 21
4.JavaScript 23
4-1.変数 23
4-2.変数を作る 23
4-3.代入 24
4-4.変数の型 24
4-5.型変換 文字列を数値へ変換 25
4-6.型変換 数値から文字列への変換 26
4-7.数字に対する算術演算子 26
4-8.文字列の連結 27
4-9.等値演算子 28
4-10.演算子③　関係演算子 29
4-11.演算子④　論理演算子 30
4-12.for文 31
4-13.do-while文 31
5.オブジェクト 33
5-1.Date 33
5-2.Math 34
5-3.Array 34
5-4.配列と繰返し文 35
6.DOM 37

6-1.要素の取得 37
6-2.要素内の上書き(innerHTML) 37
6-3.スタイルの上書き(style) 38
7.Canvas 40
7-1.基本図形の描画 40
7-2.基本図形の描画 矩形 42
7-3.基本図形の描画 多角形 43
7-4.基本図形の描画 直線 43
7-5.基本図形の描画 円弧（線） 44
7-6.基本図形の描画 (塗)円 44
7-7.基本図形の描画 曲線 44
7-8.基本図形の描画 グラデーション 45
7-9.基本図形の描画 文字 46
7-10.基本図形の描画 画像 46
8.Canvas 変形 48
9.Canvas アニメーション 50
10.Geolocation API 52
11.イベント・センサ 55
11-1.マウス 55
11-2.タッチ 56
11-3.加速度 57
12.Web Storage 59
12-1.ローカルストレージ 59
13.Webアプリ制作実習 61
13-1.じゃんけん 61
13-2.ゲーム 62

1.HTML5マークアップ要素

HTML5には２つの意味があります。

狭義のHTML5は、簡単に言えばタグの仕様です。開始タグと終了タグで単語や文章、コンテンツを囲んで意味付けを行います。

広義のHTML5は、狭義のHTML5の他に、CSS3や、JavaScript、関連する多くのAPI（アプリケーション・プログラム・インターフェース）を含んだもので、一般的にHTML5というと広義のHTML5を指します。

この節では、狭義のHTML5の要素・タグについて説明します。

サンプル　index.html

```
<!DOCTYPE html>
<html>

 <head>
  <meta charset="utf-8">
  <title>フラワーショップわかば</title>
  <meta name="viewport" content="width=device-width,user-scalable=no">
  <link rel="stylesheet" href="style.css">
 </head>

 <body>
  <div id="wrap">
    <header>
      <h1>ようこそわかばへ</h1>
      <nav>
        <ul>
          <li><a href="index.html">トップ</a></li>
          <li><a href="trivia.html">知識</a></li>
          <li><a href="item.html">商品</a></li>
```

```
          <li><a href="toku.html">特商法</a></li>
          <li><a href="mngr.html">blog</a></li>
        </ul>
      </nav>
    </header>

    <main>
    <section>
      <h1>ご挨拶</h1>
      <p>「わかば」は幕張のちいさな花屋さん。あなたと、ともだちの記念日にお花をどうぞ！</p>
    </section>
    <section>
      <h1>おすすめ</h1>
      <p>きょうのおすすめの花はジンチョウゲ。2月末ないし3月に花を咲かせることから、春の季語としてよく歌われます。</p>
    </section>
    <section>
      <h1>ジンチョウゲの花言葉</h1>
      <p>花言葉は「栄光」「不死」「不滅」「歓楽」「永遠」です。</p>
    </section>
    </main>
    <footer>
      <small>
        Copyright(c) 2015 Wakaba.All Rights Reserved.
      </small>
    </footer>
  </div>
 </body>
</html>
```

このHTMLファイルでデザインを適用した、CSSは次のファイルです。

サンプル　style.css

```
*{
 margin:0;padding:0;
 text-decoration:none;
```

```
 list-style:none;
 font-family:serif;
 color:black;
 font-size:15px;
}
#wrap{
 width:320px;
 margin:0 auto;
}
header{
 background-image:url("img/header.jpg");
}
nav li{
 background-image:url("img/nav.gif");
 background-size:100% 100%;
}
header h1{
 text-align:center;
 font-size:2em;
 line-height:5em;
 text-shadow:0px 0px 10px white;
}
nav li{
 float:left;
 width:56px;
 height:56px;
 margin:3px 0 3px 5px;
 border:1px solid gray;
 border-radius:28px;
 text-align:center;
 font-weight:bold;
 box-shadow:2px 2px 2px gray;
}
nav ul:after{
 content:"";
 display:block;
 clear:both;
}
nav li a{
 display:block;
```

```
 line-height:56px;
}
main section h1{
 background:-webkit-gradient(linear,
  left top,left bottom,from(white),to(gray)
 );
 margin:5px;padding:5px;
 text-shadow:-1px -1px 1px white;
}
main section p{
 margin:5px;
 line-height:1.5em;
}
footer{
 background-color:gray;
 text-align:center;
}
footer small{
 font-size:0.7em;padding:5px;
 text-align:center;
 color:white;
}
```

2.アプリに便利な要素

HTML5の要素はまだ数多くあります。この章ではWebアプリケーションに便利な要素について説明します。

2-1.input要素の属性

サンプル index.html内に追加

```
<section>
 <h1>input要素の属性</h1>
 <ul>
  <li>
    <input type="text" id="txt1">
    <button onclick =
    'alert(document.getElementById("txt1").value);'>
      クリック</button>
  </li>
  <li>
    <input type="range" id="rng1">
    <button onclick =
    'alert(document.getElementById("rng1").value);'>
      クリック</button>
  </li>
  <li>
    <input type="date" id="dte1">
    <button onclick =
    'alert(document.getElementById("dte1").value);'>
      クリック</button>
  </li>
  <li>
    <input type="color" id="clr1">
    <button onclick =
    'alert(document.getElementById("clr1").value);'>
      クリック</button>
  </li>
 </ul>
</section>
```

2-2.バリデート・値の制限など

サンプル index.html内に追加

```
<section>
 <h1>バリデートと値の制限</h1>
 <form method="get" action="action.php">
  <p>数字入力(必須)：<input type="number" name="number" min="0" max="10" step="3" required></p>
  <p>メアド入力：<input type="email" name="email" placeholder="(記載例)aaa@ ***.com" ></p>
  <p><input type="submit" value="送信する"></p>
 </form>
</section>
```

2-3.マルチメディア

サンプル index.html内に追加

```
<section>
 <h1>音声</h1>
 <audio autoplay loop controls>
  <source src="media/audio.mp3">
  <source src="media/audio.ogg">
  <source src="media/audio.webm">
 </audio>
</section>
<section>
 <h1>ビデオ</h1>
 <video autoplay loop controls>
  <source src="media/video.m4v">
  <source src="media/video.ogg">
  <source src="media/video.mp4">
 </video>
</section>
```

2-4.進捗・測定

サンプル index.html内に追加

```
<section>
```

```
 <p>進捗状況：
  <progress value="40" max="100">40%</progress>
 </p>
 <p>測定状況：
  <meter value="60" min="0" max="100" optimum="0" low="30"
high="80">60%</meter>
</section>
```

2-5.擬似クラス・擬似要素

サンプル　style.cssに追加

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8">
 <title>フラワーショップわかば</title>
 <meta name="viewport" content="width=device-width,user-
scalable=no">
 <style>
 *{
    margin:0;padding:0;
    text-decoration:none;
    list-style:none;
    font-family:serif;color:black;font-size:15px;
 }
 #wrap{width:320px;margin:48px auto;}
 li{
    height:48px;
    line-height:48px;
    border:1px solid gray;
    border-bottom:none;
    text-indent:1em;
    position:relative;
 }
 ul li:first-of-type{
  border-radius:10px 10px 0 0;
 }
 ul li:last-of-type{
  border-bottom:1px solid gray;
  border-radius:0 0 10px 10px;
```

```
 }
 ul li:nth-of-type(odd){
  background-color:#eee;
 }
 ul li:after{
  content:"";
  display:block;
  position:absolute;
  top:4px;right:1em;
  width:40px;height:40px;
  background-image:url(arrow.png);
 }
 </style>
</head>

<body>
 <div id="wrap">
  <nav>
    <ul>
      <li><a href="index.html">トップ</a></li>
      <li><a href="trivia.html">豆知識</a></li>
      <li><a href="item.html">商品</a></li>
      <li><a href="tokusho.html">特商法</a></li>
      <li><a href="manager.html">店長</a></li>
    </ul>
  </nav>
 </div>
</body>
</html>
```

3.CSS3

3-1.角丸

サンプル　radius.html

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>角丸</title>
  <style>
    .box{
      width:300px;
      height:200px;
      line-height:200px;
      border:1px solid gray;
      text-align:center;
      margin:10px;
    }
    #css1{
      border-radius:15px;
    }
    #css2{
      border-radius:20px 40px 80px 0px;
    }
    #css3{
      border-radius:
      20px 40px 80px 100px/100px 80px 40px 20px;
    }
    #css4{
      border-radius:50%;
    }
  </style>
 </head>
 <body>
  <div id="css1" class="box">角を丸くします</div>
  <div id="css2" class="box">角を丸くします</div>
  <div id="css3" class="box">角を丸くします</div>
```

```
  <div id="css4" class="box">角を丸くします</div>
 </body>
</html>
```

3-2.影

サンプル　shadow.html

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>影</title>
   <style>
    .box{
      width:300px;height:200px;line-height:200px;
      border:1px solid gray;
      text-align:center;margin:100px;
    }
    #css1{
      box-shadow:3px 3px 3px gray;
    }
    #css2{
      box-shadow:3px 3px 0px gray;
    }
    #css3{
      box-shadow:-3px -3px 3px gray;
    }
    #css4{
      box-shadow:inset 3px 3px 30px gray;
    }
    #css5{
      width:48px;height:48px;
      background-color:red;
      box-shadow:
        50px 0px 0px pink,
        100px 0px 0px skyblue,
        150px 0px 0px red;
    }
  </style>
```

```
 </head>
 <body>
  <div id="css1" class="box">影を作ります</div>
  <div id="css2" class="box">影を作ります</div>
  <div id="css3" class="box">影を作ります</div>
  <div id="css4" class="box">影を作ります</div>
  <div id="css5" class="box"></div>
 </body>
</html>
```

3-3.変形

サンプル　transform.html

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>変形</title>
  <style>
    div{width:500px;height:50px;
      line-height:50px;border:1px solid gray;
      text-align:center;margin:50px;
    }
    .css1{-webkit-transform:rotate(45deg);}
    .css2{-webkit-transform:scale(1,2);}
    .css3{-webkit-transform:skew(10deg,0deg);}
    .css4{
      -webkit-transform:translate(100px,-140px);
    }
  </style>
 </head>
 <body>
  <div class="css1">回転します</div>
  <div class="css2">伸縮します</div>
  <div class="css3">歪みます</div>
  <div class="css4">移動します</div>
 </body>
```

```
</html>
```

3-4.CSSトランジション

サンプル　transition.html

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>CSSトランジション</title>
  <style>
    div{width:500px;height:50px;line-height:50px;
      border:1px solid gray;
      text-align:center;margin:50px;
    }
    .css{
      -webkit-transition:
        -webkit-transform 1s ease-in-out 0;
    }
    .css1:hover{
      -webkit-transform:rotate(45deg);
    }
    .css2:hover{
      -webkit-transform:scale(1,2);
    }
    .css3:hover{
      -webkit-transform:skew(10deg,0deg);
    }
    .css4:hover{
      -webkit-transform:translate(100px,100px);
    }
    .css5{
      -webkit-transition:
        color 1s ease-in-out 0,
        background-color 1s ease-in-out 0;
    }
    .css5:hover{
      color:white;background-color:red;
    }
  </style>
```

```
 </head>
 <body>
  <div class="css1 css">回転します</div>
  <div class="css2 css">伸縮します</div>
  <div class="css3 css">歪みます</div>
  <div class="css4 css">移動します</div>
  <div class="css5">色が変わります</div>
 </body>
</html>
```

3-5.CSSアニメーション

サンプル　animation.html

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>CSSアニメーション</title>
  <style>
    .css1{
      background-image:url(img/mL.png);
      background-size:100% 100%;
      width:100px;height:100px;margin:0 auto;
      -webkit-animation:
        anime 3s ease-in-out 0 infinite normal;
      animation-fill-mode:both;
    }
    @-webkit-keyframes anime{
      0%  {-webkit-transform:rotate(0deg);}
      100%{-webkit-transform:rotate(360deg);}
    }
  </style>
 </head>
 <body>
  <div class="css1">アニメ</div>
 </body>
</html>
```

3-6.３次元

サンプル　d3.html

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>３次元変形</title>
  <style>
    body{text-align:center;}
    .henkei{
      width:100px; height:100px;margin:50px auto;
      border:2px solid gray;
      font-size:100px;line-height:100px;
    }
    #wrap{
      -webkit-transform-style:preserve-3d;
      -webkit-perspective:105px;
    }
    .henkei{
      -webkit-transform:rotateY(30deg);
    }
  </style>
 </head>
 <body>
  <div id="wrap">
    <div class=henkei>♪</div>
  </div>
 </body>
</html>
```

3-7.3次元のアニメーション

サンプル　d3animathin.html

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>三次元アニメーション</title>
```

```
  <style>
    body{text-align:center;}
    .henkei{
      width:300px; margin:50px auto;
      border:2px solid gray;
      font-size:100px;line-height:100px;
    }
    #wrap{
      -webkit-transform-style:preserve-3d;
      -webkit-perspective:105px;
    }
    .henkei{
      -webkit-animation:
        animation3d 5s ease-in-out 0 infinite normal;
    }
    @-webkit-keyframes animation3d{
      0%{-webkit-transform:rotateX(0deg) rotateY(0deg);}
      100%{
        -webkit-transform:rotateX(360deg) rotateY(360deg);
      }
    }
  </style>
 </head>
 <body>
  <div id=wrap>
    <div class=henkei>三次元アニメ</div>
  </div>
 </body>
</html>
```

4.JavaScript

この章はJavaScriptの基本と文法について学びましょう。

狭義のHTML5は文書の構造、CSS3は文章のデザインでした。JavaScriptは、文書に動きをあたえます。

4-1.変数

サンプル var.html

```
<!DOCTYPE html>
<head>
 <meta charset="utf-8">
 <title>変数</title>
</head>
<body>
 <script>
  var a;var b;
  a=2;
  b="HTML5";
  document.write("a=" +a+ ",b=" +b);
 </script>
</body>
</html>
```

4-2.変数を作る

変数は値を入れる箱とよく例えられます。箱には「変数名」という名前がついています。箱の中には、文字や数字をいれることができます。

```
var a; var b;
```

aは変数名です。前にあるvarは変数(variable)であることを示し、これでaという箱ができあがります。

変数名は、大文字小文字のアルファベット(A〜Z,a〜z)もしくはアンダーバー(_)、数字(0〜9、ただし先頭はだめ)を使用することができます。

行末のセミコロン（;）はひとつの命令文（式）の終わりです。終止符みたいなものですね。JavaScriptでは、セミコロンを忘れても改行が続いていたら、そこが命令文の終わりと認識してくれますが、なるべく忘れないように心がけましょう。

また、いちいちvarと書くのは面倒なので、カンマ(,)で一括して設定することもあります。

```
var a,b;
```

document.write()は文字を表示します。以後数値の確認用に使用します。なお、後述のテスト・デバッグにあるconsole.log()のほうが、デバッグには便利なので、そちらも使って実際にサンプルを作ってみましょう。

4-3.代入

箱ができたので、数字や文字を入れてみましょう。入れることを「代入」といい、イコール(=)を使います。JavaScriptではイコール一つだけは等しいという意味はないので注意が必要です。

```
a = 2;
b = "HTML5";
```

次のように、以上をまとめて設定することもできます。

```
var a = 2,b = "HTML5"
```

4-4.変数の型

サンプル　format.html

```
<!DOCTYPE html>
<head>
 <meta charset="utf-8">
 <title>変数の型</title>
</head>
<body>
 <script>
  var n=3;
  var str="HTML5"
  var logic=true;
  document.write("数値=" +n+ ",文字列=" +str+ "論理値=" +logic);
 </script>
</body>
</html>
```

解説

変数には、数値・文字列、そして論理値というものも代入することができます。

文字列については、その値の前後を必ずダブルクオーテーション("〜")もしくはシングルクオーテーション('〜')で囲まなければなりません。ダブルもシングルもどちらでもかまいませんが、文字列の中にクオーテーションが含まれていた場合にはそれと同じものを使ってはいけません。

論理値はある条件が真か偽かを表現するのに用い、値はtrueとfalseになります。

4-5.型変換 文字列を数値へ変換

JavaScriptは他の言語に比べ、予め型を指定することもないのですが、例えば文字列の「3」と数字の「5」をプラス「+」しても8にはならず、「35」になります。文字列から数字に型を変換しないといけない場合があります。

文字列を数値へ変換

サンプル　parse.html

```
<!DOCTYPE html>
<head>
 <meta charset="utf-8">
 <title>変数の変換</title>
</head>
<body>
 <script>
  var str="3.14159";
  var n1=parseInt(str);
  var n2=parseFloat(str);
  var n3=Number(str);
  document.write(
    "<p>str⇒" +str+
    "<p>parseInt⇒" +n1+
    "<p>parseFloat⇒" +n2+
    "<p>Number()⇒" +n3
  );
 </script>
</body>
</html>
```

parseInt()関数は文字を整数値に変換します。小数点は切捨てになります。第2引数を追加することができ、10を追加すると10進数、2を追加すると2進数として変換します。また、第1引数が0xから始まっていると自動的に16進数に、また0から始まると8進数

に変換します。例えば parseInt("08") とすると、8進数で変換しようとするのですが8進数に8は存在しないため、0が返ってしまいます。"parseInt("08",10)としましょう。

parseFloat()関数は文字を小数点付きの数値型に変換します。

Number()関数でも変換できます。

さらに、文字列から0を引いても数字になります。

4-6.型変換 数値から文字列への変換

サンプル　string.html

```
<!DOCTYPE html>
<head>
 <meta charset="utf-8">
 <title>変数の変換</title>
</head>
<body>
 <script>
  var n=3.14159;
  var str1=String(n);
  var str2=""+n;
  document.write("<p>n⇒" +n+ "<p>str1⇒" +str1+ "<p>str2⇒"
+str2);
 </script>
</body>
</html>
```

String()関数で数値を文字に変換します。

空の文字列に数字を連結することでも文字列に変換できます。

4-7.数字に対する算術演算子

サンプル cal.html

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8">
 <title>算術演算子</title>
</head>
<body>
```

```
 <script>
  var n1=7;n2=3;
  var output1=n1+n2;
  document.write("<p>7+3="+output1);

  var output2=n1-n2;
  document.write("<p>7-3="+output2);

  var output3=n1*n2;
  document.write("<p>7×3="+output3);

  var output4=n1/n2;
  document.write("<p>7÷3="+output4);

  var output5=n1%n2;
  document.write("<p>7を3で割った余り="+output5);
 </script>
</body>
</html>
```

JavaScriptでは、四則演算と余り算ができます。

4-8.文字列の連結

サンプル join.html

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8">
 <title>文字列の連結</title>
</head>
<body>
<script>
 var name1 ="山田";
 var name2 = "太郎";
 var name = "<p>名前は" +name1+name2+ "です</p>";
 document.write(name);
</script>
</body>
```

```
</html>
```
文字列に対しては「+」は文字列の連結になります

4-9.等値演算子

サンプル equal.html

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8">
 <title>等値演算子</title>
</head>
<body>
<script>
 var n1=5;n2="5";

 if(n1==n2){
  document.write("<p>n1とn2は等しいです</p>");
 }else{
  document.write("<p>n1とn2は等しくありません</p>");
 }

 if(n1===n2){
  document.write("<p>n1とn2は厳密に等しいです</p>");
 }else{
  document.write("<p>n1とn2は厳密には等しくありません</p>");
}

 if(n1!=n2){
  document.write("<p>n1とn2は等しくありません</p>");
 }else{
  document.write("<p>n1とn2は等しいです</p>");
 }

 if(n1!==n2){
  document.write("<p>n1とn2は厳密には等しくありません</p>");
 }else{
```

```
  document.write("<p>n1とn2は厳密に等しいです</p>");
 }
</script>
</body>
</html>
```

この本の最初に「=」は代入と説明しました。左辺と右辺が等しいという時には、「=」ではなく「==」または「===」を使います。

a == b　　aとbは等しい

a === b　　aとｂはデータの型を含めて厳密に等しい

a != b　　aとbは等しくない

a !== b　　aとbはデータの型を含めて厳密に等しくない

4-10.演算子③　関係演算子

サンプル　morethan.html

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8">
 <title>関係演算子</title>
</head>
<body>
<script>
 var n1=3;n2="5";
 if(n1<n2){
  document.write("<p>n1はn2より小さいです</p>");
 }

 if(n1<=n2){
  document.write("<p>n1はn2以下です</p>");
 }

 if(n1>n2){
  document.write("<p>n1はn2より大きいです</p>");
 }

 if(n1>=n2){
```

```
  document.write("<p>n1はn2以上です</p>");
 }
</script>
</body>
</html>
```

左辺と右辺のどちらが大きいか、それ以上かという比較の演算子です。

>	(右辺は左辺より)大きい
>=	(右辺は左辺)以上
<	(右辺は左辺より)ちいさい
<=	(右辺は左辺より)以下

以上、以下はその値を含みますので注意しましょう

4-11.演算子④　論理演算子

サンプル　douji.html

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8">
 <title>論理演算子</title>
</head>
<body>
<ul>
<script>
 var n1=3;
 var n2=3;
 var n3=5;
 var n4=5;

 if(n1==n2 && n3==n4){
  document.write("<p>n1とn2は等しく、且つn3とn4は等しい。</p>");
 }

 n4=3;
 if(n1==n2 || n3==n4){
  document.write("<p>n1とn2は等しいか、またはn3とn4が等しい</p>");
 }
```

```
</script>
</ul>
</body>
</html>
```
複数の条件式が同時に成り立つときは 「&&」、どちらか一方が成り立つ時は 「||」をつかいます。|（縦線、バーチカルはシフトキーを押しながら¥マークを押すと入力できます。）

4-12.for文

サンプル for.html

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8">
 <title>for</title>
</head>
<body>
 <ol>
 <script>
  var max=10;
  for(var i=0;i<max;i++){
    document.write("<li>" +i+ "回目</li>");
  }
 </script>
 </ol>
</body>
</html>
```
ある程度決まった回数を繰り返して処理を行うのがfor構文です。

```
for(初期化式;条件式;更新式){繰り返しの処理内容}
```

4-13.do-while文

サンプル while.html

```
<!DOCTYPE html>
<html>
<head>
```

```
 <meta charset="utf-8">
 <title>1-20-2.do-while</title>
</head>
<body>
<ul>
<script>
 var i=50;
 do{
  document.write("<li>" +i+ "</li>");
  i-=Math.floor(Math.random()*3);
 }while(0<i);
</script>
</ul>
</body>
</html>
```

最初にdo{処理}の中を実行し、その後while(条件式)を評価して繰り返すかどうかを決めます。条件にかかわらず最初に一度実行させたい時にはdo-whileを使うといいでしょう。

5.オブジェクト

5-1.Date

サンプル date.html

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8">
 <title>日時の取得</title>
</head>
<body>
 <script>
  var now = new Date();
  var y = now.getFullYear();
  var m = now.getMonth()+1;
  var d = now.getDate();
  var w = now.getDay();
  var h = now.getHours();
  var i = now.getMinutes();
  var s = now.getSeconds();
  var ms= now.getMilliseconds();
  var week = new Array("日","月","火","水","木","金","土");
  document.write(y+"年"+m+"月"+d+"日"+week[w]+"曜日"+h+"時"+i+"分"+
(s+ms/1000)+"秒");
  document.write("<br>now=" +now);
 </script>
</body>
</html>
```

Date()はコンストラクタ。new演算子はそのコンストラクタ(Date())から、新しいオブジェクト(new)を生成しています。コンストラクタは設計図だと思うとわかりやすいかもしれません。設計図をそのまま使うのではなく、新しい複製を作りそれを利用する。そうすれば、その複製物は設計図の機能をそのまま引き継いでいますので、機能をわざわざ作らなくても使い勝手がとてもよくなります。;

5-2.Math

サンプル math.html

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8">
 <title>Math</title>
</head>
<body>
<script>
 var n1 = Math.random()*10;
 var n2 = Math.floor(n1);
 var n3 = Math.round(n1);
 var n4 = Math.ceil(n1);
 var n5 = Math.pow(2,3);
 var n6 = Math.sin(30/180*Math.PI);
 var n7 = Math.cos(45/180*Math.PI);
 var n8 = Math.tan(60/180*Math.PI);
 var pi = Math.PI;
 doocument.write("<p>乱数="+n1+"<p>切り捨て="+n2+"<p>四捨五入
="+n3+"<p>切り上げ="+n4+"<p>2の3乗="+n5+"<p>sin(30)="+n6
+"<p>cos(45)"+n7+"<p>tan(60)"+n6+"<p>円周率="+pi)
</script>
</body>
</html>
```

乱数で0~2までの整数の乱数を生成するには次のようなコードを記述します。これをつかえば占いアプリや、じゃんけんゲームができそうですね。

```
var max = 3;
n1 = Math.floor(Math.random()*max);
```

三角関数などでは、(〜)の中には0〜360度の角度ではなく、ラジアンという単位を用いる弧度を用い、0〜360度が0〜2πラジアンに相当します。

5-3.Array

サンプル array.html

```
<!DOCTYPE html>
<html>
```

```
<head>
 <meta charset="utf-8">
 <title>Array</title>
</head>
<body>
<script>
 var sweets = new Array();
 sweets[0] = "チョコ";
 sweets[1] = "ガム";
 sweets[2] = "アイス";
 sweets[3] = "ケーキ";
 document.write("<p>0番目=" +sweets[0]+ "<p>1番目=" +sweets[1]+
"<p>2番目=" +sweets[2]+ "<p>3番目=" +sweets[3]);
</script>
</body>
</html>
```

Arrayオブジェクトは配列を作ります。

配列は、ひとつの変数に複数のデータを格納するためのデータ構造です。

例えばお菓子の箱に、チョコ・ガム・アイス・ケーキが所属していることを表すには、Array()を使います。

次のように、配列をまとめて設定する書き方もあります。

```
var sweets =
new Array( "チョコ","ガム","アイス","ケーキ");
```

または、もっと簡単に記述することもできます。

```
var sweets = ["チョコ","ガム","アイス","ケーキ"];
```

5-4.配列と繰返し文

配列は繰り返し分で処理すると便利です

サンプル array-for.html

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8">
 <title>Array</title>
```

```
</head>
<body>
<script>
 var sweets = ["チョコ" , "ガム" , "アイス" , "ケーキ"];
 document.write("<ol>");
 for(var i=0;i<sweets.length;i++){
  document.write("<li>" +sweets[i]+ "</li>");
 }
 document.write("</ol>");

 sweets.reverse();
 document.write("<ol>");
 for(var i=0;i<sweets.length;i++){
  document.write("<li>" +sweets[i]+ "</li>");
 }
 document.write("</ol>");
</script>
</body>
</html>
```

配列の配列のデータを扱うには、繰り返し分で１つずつ抜き出して処理を行います。reverse()メソッドは、順番を逆にします。lengthプロパティは配列の個数になります。

6.DOM

DOMは「Document Object Model」。

DOMは、ブラウザ上のすべてのものをオブジェクトとみなし、値を変えたり、ソースを変えたり、デザインをかえることができます。

Webページ上に様々な部品が配置されていて、それらの内容物やデザインを変えることができると思っていただければいいでしょう。

6-1.要素の取得

「要素」とは開始タグと終了タグまでのひとつの固まりです。

DOMを扱う際は、まず、対象となる要素のID（例：elem1）を指定し、それに対して操作します。

```
var hoge = document.getElementById("elem1");
```

ただ、このメソッドは長いので、次のようなユーザ関数を作り、コードを簡素化すると便利です。

```
function $(id){
 return document.getElementById(id);
}
```

6-2.要素内の上書き(innerHTML)

サンプル inner.html

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8">
 <title>要素の上書き</title>
</head>
<body>
 <p>入力：<input id=a type=text></p>
 <div id=b>ここに出力</div>
 <button onclick="clk1();">押す</button>
 <script>
  function clk1(){
    document.getElementById("b").innerHTML
```

```
     = document.getElementById("a").value;
  }
 </script>
</body>
</html>
```

ページを再読込することなく、要素の中のコンテンツを書き換えることができます。サンプルは、ボタンを押すと、テキストボックスに入力した文字を「ここに出力」という部分(idは"b")を上書きするものです。

document.getElementById("〜")で要素を特定し、valueプロパティは、特定した要素内のvalueの値。innerHTMLプロパティは、特定した要素の内側のHTMLを上書きします。

6-3.スタイルの上書き(style)

サンプル style.html

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8">
 <title>スタイルの上書き</title>
</head>
<body>
 <div id="elem">Happy HTML5</div>
 <button onclick="clk2(1);">色の変更</button>
 <button onclick="clk2(2);">太さの変更</button>
 <button onclick="clk2(3);">大きさの変更</button>
 <button onclick="clk2(4);">左右揃えの変更</button>
 <button onclick="clk2(5);">visibility:hidden</button>
 <button onclick="clk2(6);">visibility:visible</button>
 <button onclick="clk2(7);">display:none</button>
 <button onclick="clk2(8);">display:block</button>
 <button onclick="clk2(9);">変形</button>
</body>
<script>
 function $(id){return document.getElementById(id);}
```

```
 function clk2(i){switch(i){
  case 1:$("elem").style.color="red";        break;
  case 2:$("elem").style.fontWeight="bold";    break;
  case 3:$("elem").style.fontSize="100px";   break;
  case 4:$("elem").style.textAlign="center"; break;
  case 5:$("elem").style.visibility="hidden";  break;
  case 6:$("elem").style.visibility="visible"; break;
  case 7:$("elem").style.display="none";     break;
  case 8:$("elem").style.display="block";      break;
  case 9:$("elem").style.webkitTransform="rotate(30deg)";
  break;
 }}
</script>
</html>
```

innerHTMLと同様に、ページを再読み込みすることなく、CSSデザインをJavaScriptで書き換えることができます。

要素を特定したら、styleプロパティにCSSのプロパティを記述しイコールの右辺にはCSSの値をダブルクオーテーション(" ")で挟んで記述してください。単位を伴うものは必ず書かなければなりません。なお、JavaScriptでは「-」は使えませんので、font-sizeをfontSizeのように、「-」を削除しその次の文字を大文字で記述します。

7.Canvas

7-1.基本図形の描画

サンプル canvas.html

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8">
 <title>canvas</title>
 <style>
  #mycanvas{border:1px solid gray;}
 </style>
</head>
<body>
 <canvas id="canvas" width="500" height="500">
  canvasは使用できません
 </canvas>

 <script>
 var canvas = document.getElementById("canvas");
 var ctx = canvas.getContext("2d");

//矩形
 ctx.fillStyle = "red";
 ctx.fillRect(50,50,400,400);

//多角形
 ctx.beginPath();
 ctx.moveTo(250,10);
 ctx.lineTo(10,490);
 ctx.lineTo(490,250);
 ctx.closePath();
 ctx.fillStyle = "green";
 ctx.fill();

//直線
```

```
 ctx.beginPath();
 ctx.moveTo(100,100);
 ctx.lineTo(400,400);
 ctx.lineWidth = 30;
 ctx.strokeStyle = "blue";
 ctx.stroke();

//円弧
 var angle1 = 40/180 * Math.PI ;
 var angle2 = 300/180 * Math.PI ;
 ctx.beginPath();
 ctx.arc(250, 250, 200, angle1, angle2, false);
 ctx.strokeStyle = "yellow";
 ctx.lineWidth = 10;
 ctx.stroke();

//円
 ctx.beginPath();
 ctx.arc(250, 250, 150, 0, 2*Math.PI, false);
 ctx.closePath()
 ctx.fillStyle = "pink";
 ctx.fill();

//曲線
 ctx.beginPath();
 ctx.moveTo(0,200);
 ctx.quadraticCurveTo(250,0,500,200);
 ctx.strokeStyle = "lightgreen";
 ctx.lineWidth = 10;
 ctx.stroke();

 ctx.beginPath();
 ctx.moveTo(0,400);
 ctx.bezierCurveTo(200, 0, 400, 500, 500, 400);
 ctx.stroke();

//グラデーション
 var g = ctx.createLinearGradient(0,0, 0,250);
```

```
 g.addColorStop(0,"red");
 g.addColorStop(0.5,"green");
 g.addColorStop(1,"blue");
 ctx.fillStyle = g;
 ctx.fillRect(0,0, 250,250);

//文字列
 ctx.fillStyle = "black";
 ctx.font = "100px serif";
 ctx.fillText("HTML5",50,450);
</script>
</body>
</html>
```

canvas要素は例えると透明の画用紙です。背景や枠線をCSSで設定することができます。

なおcanvasではcanvas要素の属性で縦横500px指定します。

タグは挟んだコンテンツに意味づけするのですが、canvasタグの間の部分は、canvasを扱えないブラウザで表示したときにメッセージとして表示されます。（IE8以下等のcanvasが扱えないブラウザへのメッセージ）

script内では、canvas要素を特定してJavaScriptで扱えるようcanvasの要素オブジェクトmycanvasを作ります。そのmycanvasから２次元の描画コンテキストを取得するため、getContext("2d")メソッドを登録します。するとctxは絵を書くための道具箱になります。

なお、canvasには座標があります。左上を(x,y)=(0,0)として右が正のx座標、下向きが正のy座標です。

7-2.基本図形の描画 矩形

矩形は四角形です。

fillStyleプロパティで記述した行から下の塗り色を指定します。

fillRect(x座標,y座標,幅,高さ)は矩形を描画するメソッドです。サンプルでは(50,50)から右横に400の幅、下に400の高さの四角形が描画されます。

```
ctx.fillStyle = "red";
ctx.fillRect(50,50,400,400);
```

また、枠線(stroke)の矩形も作ることができます。

```
ctx.strokeStyle = "blue";
ctx.strokeRect(80,80,400,400);
```
さらに、句形部分を消去することもできます。

```
ctx.crearRect(100,100,100,100);
```

7-3.基本図形の描画 多角形

矩形は描画が簡単ですが、次に多角形です。少々難しくなります。

beginPath()メソッドでパスを開始します。パスは図形の輪郭だと思うといいでしょう。

moveTo(x座標,y座標)メソッドは図形の描画の開始点です。これを書き漏らすと、前からの図形を引きずる可能性があるので、気をつけましょう。

lineTo(x座標,y座標)メソッドはその座標まで直線の輪郭を設定します。このメソッドを追加すればどのような図形も描けますね。

closePath()メソッドで、パスの終了し開始点に戻ります。

緑色の塗り色を設定した後、fill()メソッドで輪郭内を塗ります。

```
ctx.beginPath();
ctx.moveTo(250,10);
ctx.lineTo(10,490);
ctx.lineTo(490,250);
ctx.closePath();
ctx.fillStyle = "green";
ctx.fill();
```

7-4.基本図形の描画 直線

直線は、多角形と似ています。ただ、closePath()メソッドが開始点に戻らないのであれば必ずしも必要ではなく、塗りつぶしではなく枠線を引くメソッドを使います。

lineWidthプロパティは、記述行から下の線の幅を指定します。

strokeStyleは、記述した行から下の線の色を指定します。

最後に、stroke()で線を引きます。

```
ctx.beginPath();
ctx.moveTo(100,100);
ctx.lineTo(400,400);
ctx.lineWidth = 30;
ctx.strokeStyle = "blue";
```

```
ctx.stroke();
```

7-5.基本図形の描画 円弧（線）

```
var angle1 = 40/180 * Math.PI ;
var angle2 = 300/180 * Math.PI ;
ctx.beginPath();
ctx.arc(250, 250, 200, angle1, angle2, false);
ctx.strokeStyle = "yellow";
ctx.lineWidth = 10;
ctx.stroke();
```

円弧は、arc()メソッドを使います。このメソッドの中では、円弧の中心のx座標、y座標、半径と続き、開始の角度、終了の角度、回転方向を順番に指定します。Math.PIは円周率ですね。

上では開始の角度、終了の角度と書きましたが、正確には弧度といいます

JavaScriptでは弧度の単位ラジアンを使います。ざっくりと角度とラジアンとの関係は次のとおりです。またラジアンは座標上での向きが決まっています。

0度（時計の3時の位置）　0ラジアン

90度（時計の6時の位置）　0.5π(0.5*Math.PI)

180度（時計の9時の位置）　π(Math.PI)

270度（時計の12時の位置）　1.5π(1.5*Math.PI)

360度（時計の3時の位置）　2π(2*Math.PI)

このことから、例えば40度は一周360度で割ってラジアンの一周2πをかければ導かれます。つまり「ラジアン=角度/180*π」の関係になります。

7-6.基本図形の描画 (塗)円

円は、円弧を360度回転させたものですね。

```
ctx.beginPath();
ctx.arc(250, 250, 150, 0, 2*Math.PI, false);
ctx.closePath()
ctx.fillStyle = "pink";
ctx.fill();
```

7-7.基本図形の描画 曲線

曲線には2種類あります。

quadraticCurveTo()は2次曲線のメソッドになります。最初にmoveTo()メソッドで開始点を指定し、quadraticCurveTo()メソッドで、制御点のx座標、y座標、終了点のx座標、y座標を指定します。

bezierCurveTo()は3次曲線のメソッドになります。こちらも最初にmoveTo()メソッドで開始点を指定し、bezierCurveTo()メソッドで、1つめの制御点のx座標、y座標、そして2つ目の制御点のx座標、y座標、最後に終点のx座標とy座標を指定します。

```
//2次曲線
ctx.beginPath();
ctx.moveTo(0,200);
ctx.quadraticCurveTo(250,0,500,200);
ctx.strokeStyle = "lightgreen";
ctx.lineWidth = 10;
ctx.stroke();
//3次曲線
ctx.beginPath();
ctx.moveTo(0,400);
ctx.bezierCurveTo(200, 0, 400, 500, 500, 400);
ctx.stroke();
```

7-8.基本図形の描画 グラデーション

どの図形でもいいのですが、fillStyleプロパティにグラデーションを指定するとかんがえるとわかりやすいかもしれません。

グラデーションのオブジェクトをcreateLinearGradient()メソッドで作り、その際にグラデーションの範囲を設定します。サンプルでは(0,0)から(0,250)にかけてグラデーションの範囲が設定されます。

そのオブジェクトに色を登録するのがaddColorStop()メソッドです。このメソッドでは、開始点(0)から終了点(1)までと、その間(0〜1)の色も指定することができます。

```
var g = ctx.createLinearGradient(0,0, 0,250);
g.addColorStop(0,"red");
g.addColorStop(0.5,"green");
g.addColorStop(1,"blue");
ctx.fillStyle = g;
ctx.fillRect(0,0, 250,250);
```

7-9.基本図形の描画 文字

文字も描画することができます。

font プロパティで文字の大きさと文字フォントを指定します。

fillText()メソッドで、表示する文字列、x座標、y座標を指定して、文字を描画します。

x,y座標は文字の左下になることに注意しましょう。

```
ctx.fillStyle = "black";
ctx.font = "100px serif";
ctx.fillText("HTML5",50,450);
```

なお、ctx.textAlign="center" とすると、文字の左右真ん中がx,yの座標になるので、

試してみてください。

7-10.基本図形の描画 画像

サンプル drawimage.html

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8">
 <title>drawimage</title>
 <style>
  #mycanvas{border:1px solid gray;}
 </style>
</head>
<body>
 <canvas id="canvas" width="500" height="500">
  canvasは使用できません
 </canvas>
 <script>
 var canvas = document.getElementById("canvas");
 var ctx = canvas.getContext("2d");

 var img = new Image();
 img.src = "img/mL.png";
 img.onload = function(){
  ctx.drawImage(img,250,250);
  ctx.drawImage(img,100,100,100,150);
  ctx.drawImage(img,200,200,100,100,0,300,300,200);
```

```
 }
 </script>
</body>
</html>
```
画像をcanvasに表示することができます。画像の読込に時間がかかるため、onloadイベントで画像をすべて読み込んだ後に、drawImage()で描画を行っています。

drawImage(img, x座標,y座標)メソッドは、座標(x,y)にimgをオリジナルのサイズのまま表示させます。

```
 ctx.drawImage(img, 250,250);
```
画像の大きさを横100px 縦150px にしたければ、その値を後ろに付け足します。

```
 ctx.drawImage(img,100,100,100,150);
```
元画像の(50,50)の位置から横80px縦90pxの大きさの画像を切り抜いて表示したければ、その値を前に付け足します。

```
 ctx.drawImage(img,50,50,80,90,100,100,100,100);
```

8.Canvas 変形

基本的な図形の描画はできるようになりました。今度はそれらを変形しましょう。

サンプル canvas-transform.html

```
<!DOCTYPE html>
<head>
 <meta charset="utf-8">
 <title>変形</title>
</head>
<body>
 <canvas id="canvas" width="500" height="500"></canvas>
 <script>
  var canvas = document.getElementById('canvas');
  var ctx = canvas.getContext('2d');
  //変形した矩形の描画
  ctx.save();
  ctx.translate(100,100);
  ctx.rotate(30/180*Math.PI);
  ctx.scale(1,0.5);
  ctx.fillStyle="blue";
  ctx.fillRect(-50,-50,100,100);
  ctx.restore();

  ctx.save();
  ctx.translate(200,200);
  ctx.rotate(30/180*Math.PI);
  ctx.scale(1,0.5);
  ctx.fillStyle="red";
  ctx.fillRect(-50,-50,100,100);
  ctx.restore();
 </script>
</body>
</html>
```

rotate()メソッドは図形を回転させることができます。ただしラジアンの単位であることに注意しましょう。

```
 ctx.rotate(30/180*Math.PI);
```

変形の際は、常に原点(0,0)を中心に回転や伸縮する仕様になっていて自由な変形ができません。そのため、translate()メソッドで原点(0,0)を都度移動して、そこを中心(0,0)に変形の描画を行います。

```
 ctx.translate(100,100);
```

また、原点を移動したり、伸縮、回転を行っているので、別の描画にも影響してしまいます。そのため、save()メソッドで、その時点での描画のスタイルを一時保存し、restore()メソッドで、保存した描画スタイルを復元する工夫を行っています。

9.Canvas アニメーション

静止描画をつくることはできるようになりましたので、いよいよcanvasでアニメーションを制作しましょう。いわゆるパラパラ漫画を連想してください。

サンプル canvas-animation.html

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8">
 <title>canvasアニメーション</title>
 <style>
  #mycanvas{background-color:skyblue;}
 </style>
</head>

<body>
 <canvas id="canvas" width="600" height="600"></canvas>
 <script>
  var canvas = document.getElementById("canvas");
  var ctx = canvas.getContext("2d");

  //初期設定
  var x = 300;
  var dir = 1;
  var i=0;

  //アニメーション
  var mytimer = setInterval(anime, 50);

  function anime(){
    if(i==60*60*20){clearInterval(mytimer);}

    //1.画面を消去
    ctx.clearRect(0, 0, 600, 600);

    //2.図形を描画
    ctx.beginPath();
```

```
    ctx.arc(x, 300, 100, 0, 2*Math.PI, false);
    ctx.closePath();
    ctx.fillStyle = "blue";
    ctx.fill();

    ctx.beginPath();
    ctx.fillStyle = "blue";
    ctx.strokeRect(300-100, 300-100, 200, 200);

    //3.次の状態を計算しておく
    x += dir;
    if(x<100 || 500 < x){dir *= -1;};
  }
 </script>
</body>
</html>
```

初期設定の変数xは移動する描画図形の中心のx座標です。変数にして、刻々と値を変えます。

変数dirは、その図形の向きです。xに都度追加することで、プラスの時は右に進行、マイナスの時は左に進行。値は移動の距離になります。

setInterval()メソッドは、起動する関数と時間間隔を指定して、一定時間ごとに関数を起動するものです。ここでは関数をanimeとしました。関数名には丸括弧「()」はつけません。時間の単位はミリ秒になります。50ミリ秒だと1秒間に20コマのアニメーションになります。

このタイマに名前を付けてmytimeとします。そうすると、タイマを終了したい時にclearInterval(mytimer) と命令すれば終了します。

関数内はパラパラ漫画1枚の作業の内容を指定しておきます。具体的には、①画面を消去、②図形を描画、③次の状態を計算、になります。

10.Geolocation API

HTML5はGPSなどを利用して、位置情報を取得することができます。google mapなどの地図アプリを表示した時に、現在の位置情報を取得してもいいか許可を求められたことがあるでしょう。位置情報を取得して地図を表示しましょう。なお、使い方によっては有料になる場合もあります。

サンプル geo.html

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8">
 <title>位置情報</title>
 <style>
  *{margin:0; padding:0;}
 </style>
</head>
<body>
<main>
 <h1>位置情報</h1>
 <ul>
  <li id="ido">緯度</li>
  <li id="keido">経度</li>
  <li id="kodo">高度</li>
  <li id="hogaku">方角</li>
  <li id="sokudo">速度</li>
 </ul>
</main>

<script src="http://www.google.com/jsapi"></script>
<script>
function $(id){return document.getElementById(id);}

var geolocation;
if(typeof(navigator.geolocation)=='undefined'){
 geolocation=google.gears.factory.create('beta.geolocation');
}else{geolocation=navigator.geolocation;}
```

```
var option={enableHighAccuracy:true,timeout:10000,maximumAge:0};

geolocation.getCurrentPosition(seiko,shippai,option);

//成功
function seiko(position){
 $("ido").innerHTML = position.coords.latitude;
 $("keido").innerHTML = position.coords.longitude;
 $("kodo").innerHTML = position.coords.altitude;
 $("hogaku").innerHTML = position.coords.heading;
 $("sokudo").innerHTML = position.coords.speed;
}

//失敗
function shippai(err){alert("失敗"+err.code+err.message);}
</script>
</body>
</html>
```

次のコードはAndroid用スクリプト

```
<script src="http://www.google.com/jsapi"></script>
```

位置情報を取得するコード

```
var geolocation;
if(typeof(navigator.geolocation)=='undefined'){
 geolocation=google.gears.factory.create('beta.geolocation');
}else{
 geolocation=navigator.geolocation;
}
```

navigator.geolocation：このプロパティがあればHTML5の通常の処理を行います。端末によってはプロパティがない（android）ことがあり、その場合はgoogle のWeb拡張設定を使って値を取得します。

次のコードは、Android用のoptionを設定する

```
var option={
 enableHighAccuracy:true,
 timeout:10000,
 maximumAge:0
};
```

enableHighAccuracyは、正確さを増す、つまりGPSの使用。timeoutはデータ取得を待つミリ秒。maximumAgeはキャッシュを保持するミリ秒です。

データ取得形式を指定するには次のふたつのいずれかを使います。

①現在の一度きりのデータを取得する場合は

```
geolocation.getCurrentPosition(seiko,shippai,option);
```

②定期的にデータを取得する場合

```
geolocation.watchPosition(seiko,shippai,option);
```

成功したら次のコードで値を得ることができます。

```
function seiko(position){
 var ido = position.coords.latitude;
 var keido = position.coords.longitude;
 var kodo = position.coords.altitude;
 var hogaku = position.coords.heading;
 var sokudo = position.coords.speed;
}
```

次のコードは失敗したらアラートを表示します

```
function shippai(err){
 alert("失敗"+err.code+err.message);
}
```

11.イベント・センサ

11-1.マウス

サンプル　mouse.html

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8">
 <title>マウスイベント</title>
</head>
<body>
 <canvas id=mycanvas width="310" height="320"></canvas>
</body>
<script>
 var canvas=document.getElementById("mycanvas");
 var ctx=canvas.getContext("2d");
 var pos={x:150,y:150};
 canvas.addEventListener("mousemove",function(e){
  var rect = e.target.getBoundingClientRect();
  pos.x = e.clientX - rect.left;
  pos.y = e.clientY - rect.top;
  ctx.fillStyle="red";
  ctx.fillRect(pos.x-2,pos.y-2,4,4);
 });
</script>
</html>
```

マウスが動いたら、軌跡に四角形を描画していきます。

イベント

```
canvas.addEventListener("イベント",動作);
```

"イベント"をきっかけにして指定された動作が動作する「イベントリスナ」をcanvasの要素オブジェクトに登録しています。動作には関数名のほか、無名関数を使用することも多いようです。

マウスに関するイベントには

click	クリックした時
mousemove	マウスが動いている間

mousedown	クリックした瞬間
mouseup	クリックを離した瞬間

などがあります。

また、e.clientX、e.clientYは、マウスの x , y 座標の値が入っています。なお、この値はドキュメントを基準とした座標のため、canvas内の座標の基準に補正する必要があります。e.target.getBoundingClientRect(); に入っているcanvasの矩形状態の位置の値を元に、rect.left（ x 方向の補正値）、rect.top（ y 方向の補正値）で補正しています。

function(e){}：は無名関数です。わざわざ関数名をつけるまでもないときに処理内容を記述するもので、匿名関数、使い捨て関数ともいわれます。function(e){} の「e」についてはイベントが発生した時にイベントの情報はその関数の第1引数に入ることになっているので、今回は引数を任意に「e」としています。

11-2.タッチ

サンプル touch.html

```
<!DOCTYPE html>
<html>
<head>
 <meta charset=utf-8 >
 <title>お絵かきソフト</title>
 <meta name = "viewport" content="width=device-width,user-
scalable=no">
</head>
<body>
<section>
 <p id=info>座標</p>
 <canvas id=mycanvas width=310 height=320></canvas>
</section>
</body>
<script>
 var canvas=document.getElementById("mycanvas");
 var ctx=canvas.getContext("2d");
 var finger={x:150,y:150,z:0};
 canvas.addEventListener("touchmove",function(e){
  e.preventDefault();
```

```
  var rect = e.target.getBoundingClientRect();
  finger.x = e.touches[0].clientX-rect.left;
  finger.y = e.touches[0].clientY-rect.top;
  ctx.fillStyle="green";
  ctx.fillRect(finger.x-2,finger.y-2,4,4);
  document.getElementById("info").innerHTML
  ="x=" +finger.x+ ",y=" +finger.y+ "px";
});
</script>
</html>
```

タッチイベントにはつぎのようなものがあります。

touchmove　タッチした指が動いている間

touchstart　タッチした瞬間

touchend　タッチを離した瞬間

e.preventDefault();メソッドは、画面上を指でスライドさせてもブラウザが移動しないようにするあめの記述です。

タッチする指は複数あります。touches[0]は、最初にタッチした指のxy座標。

touches[1]とすれば２本目、touches[2]とすれば３本目の指の座標をそれぞれ取得できます。

11-3.加速度

サンプル　accel.html

```
<!DOCTYPE html>
<html>
<head>
 <meta charset=utf-8 >
 <title>お絵かきソフト</title>
 <meta name = "viewport" content="width=device-width,user-
scalable=no">
</head>
<body>
<section>
 <p id=info>加速度</p>
 <canvas id=mycanvas width=310 height=320></canvas>
</section>
</body>
```

```
<script>
 var canvas=document.getElementById("mycanvas");
 var ctx=canvas.getContext("2d");
 var pos={x:150,y:150,z:0};
 window.addEventListener("devicemotion",function(e) {
  pos.x += e.accelerationIncludingGravity.x;
  pos.y -= e.accelerationIncludingGravity.y;
  pos.z += e.accelerationIncludingGravity.z;
  ctx.fillStyle="red";
  ctx.fillRect(pos.x-2,pos.y-2,4,4);
  document.getElementById("info").innerHTML
  = "加速度：x="+pos.x+"px,y="+pos.y+"px,y="+pos.y+"px";
});
</script>
</html>
```

e.accelerationIncludingGravity：３方向のセンサの値。androidの場合、軸の方向が逆ですので補正が必要

window.addEventListener()：window（ブラウザ）に対してイベントが発生したら処理を行うようにしています

devicemotion：デバイスに動きがある間

12.Web Storage

ゲームで楽しんだあと、次回はその続きから始めたいもの。今までの得点やレベル、アイテムを保存しなければなりません。また体重とかの日々の値を記録できると便利ですね。

HTML5にはデータを保存する便利な機能があります。それがWebストレージ。Storageオブジェクトには「ローカルストレージ」と「セッションストレージ」の２種類があります。「セッションストレージ」はブラウザを閉じるとデータが消えてしまいます。「ローカルストレージ」はブラウザを閉じてもデータは残ります。今回は「ローカルストレージ」について説明します。

12-1.ローカルストレージ

サンプル storage.html

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8">
 <title>ローカルストレージ</title>
 <style>
  #memo{width:100%;height:300px;}
 </style>
</head>

<body>
 <h1>メモ帳</h1>
 <button onclick="clk(1);">保存</button>
 <button onclick="clk(2);">読出</button>
 <button onclick="clk(3);">削除</button>
 <button onclick="clk(4);">データを全で削除</button>
 <textarea id="memo"></textarea>

<script>
 var memo = document.getElementById("memo");
 function clk(i){switch(i){
  case 1:localStorage.setItem("str",memo.value);break;
```

```
  case 2:
    var str=localStorage.getItem("str");
    memo.value=str;
    break;
  case 3:localStorage.removeItem("str");break;
  case 4:localStorage.clear();break;
 }}
</script>
</body>
</html>
```

ローカルストレージは、ブラウザ内に、「キー」という荷札のようなものとデータをペアにして保存する仕組みです。ブラウザを閉じてもそのまま保存されます。

localStorageのgetItem(key)	キーのデータの読出
setItem(key , 値)	キーとデータを保存
removeItem(key)	キーのデータを削除
clear()メソッド	ローカルストレージにあるデータをすべて削除

13.Webアプリ制作実習

13-1.じゃんけん

サンプル　janken.html

```
<!DOCTYPE html>
<meta charset=utf-8>
<title>じゃんけんゲーム</title>

<section>
 <h1>ジャンケン</h1>
 <button onclick="janken(0);">ぐー</button>
 <button onclick="janken(1);">ちー</button>
 <button onclick="janken(2);">ぱー</button>
 <div id=output>じゃんけん・・・</div>
</section>

<script>
var point=0;                        //ポイントの初期化
var br="<br>";
var player=["ぐー","ちー","ぱー"];   //ぐーを0,ちーを1,ぱーを2
var pc=["ぐー","ちー","ぱー"];
var jadge=["まけ","あいこ","かち"]
var jadgeNo=new Array();            //勝ち負けの多次元配列表
jadgeNo[0]=[1,2,0];
jadgeNo[1]=[0,1,2];
jadgeNo[2]=[2,0,1];

function janken(playerNo){           //ボタンクリック時の動作
 var pcNo=Math.floor(Math.random()*3); //PCの手をかんがえる
 if(jadgeNo[playerNo][pcNo]==2)point++;      //勝ったら得点

 document.getElementById("output").innerHTML   //ページ表示
 ="PCは"+pc[pcNo]+br+"あなたは"+player[playerNo]+br+
```

```
jadge[jadgeNo[playerNo][pcNo]]+br+"得点"+point;
}
</script>
```

13-2.ゲーム

サンプル　game.html

```
<!DOCTYPE html>
<meta charset=utf-8>
<title>シューティングゲーム</title>
<meta name=viewport content="width=device-width,user-
scalable=no">

<style>
*{margin:0; padding:0;}
#open,#play,#finish{
 width:320px;height:480px;border:1px solid gray;
 text-align:center}
#play,#finish{display:none;}
button{font-size:50px;margin:100px auto;}
</style>

<section id=open>
 <h1>START</h1>
 <button onclick="start();">開始</button>
</section>
<section id=play>
 <canvas id="canvas" width="320" height="480"></canvas>
</section>
<section id=finish>
 <h1>END</h1>
 <p>あなたの得点は<span id=result></span>です</p>
</section>

<script>
var canvas=document.getElementById("canvas");
var c=canvas.getContext("2d");

//変数の設定と初期化
```

```
var open=document.getElementById("open");
var play=document.getElementById("play");
var finish=document.getElementById("finish");
var result=document.getElementById("result");

var frame=0;                    //全体のフレームカウント
var target={x:250,y:100,dir:1};   //ターゲット（x位置と移動方向）
var player={x:250,y:400};       //プレイヤー（x位置）
var ball={x:-10,y:-10};         //ボール（x位置とy位置）
var point=0;                    //得点
var sound=new Audio("media/pon.mp3");    //効果音読込

var limit=10;                     //制限時間
var timer;                      //タイマー

function anime(){//ルーチンのスクリプト――――――――――
 frame++;
 if(limit*50<=frame){end();}

 //画面をクリア
 c.clearRect(0,0,canvas.width,canvas.height);

 //targetと得点を描画
 c.save();
 c.translate(target.x,target.y);
 c.fillStyle="red";
 c.beginPath();
 c.moveTo(0,-10);
 c.lineTo(-25,10);
 c.lineTo(25,10);
 c.closePath();
 c.fill();

 c.font="20px sans-serif";
 c.textAlign="center";
 c.fillText(point,0,-10);
```

```
 c.restore();

 //playerを描画
 c.save();
 c.translate(player.x,player.y);
 c.fillStyle="blue";
 c.fillRect(-25,-5,50,10);
 c.restore();

 //ballを描画
 c.save();
 c.translate(ball.x,ball.y);
 c.fillStyle="green";
 c.beginPath();
 c.arc(0,0,10,0,2*Math.PI,false);
 c.fill();
 c.restore();

 //移動計算
 target.x+=target.dir;
 ball.y-=5;

 //壁衝突計算
 if(target.x<25||295<target.x){target.dir*=-1;}

 //得点ゲット
 if ((100 == ball.y) && (Math.abs(target.x - ball.x)<25)){
  point++;
  sound.play();
  ball.y=-100;
 }
}
//開始の処理 ――――――――――
function start(){
 sound.play();
 open.style.display="none";
 play.style.display="block";
 timer = setInterval(anime,50);
```

```
 anime();
}
//終了の処理――――――――――
function end(){
 clearInterval(timer);
 play.style.display="none";
 finish.style.display="block";
 result.innerHTML=point;
}
//シュートの処理――――――――――
canvas.addEventListener("mousedown",function(){
 ball.x = player.x;
 ball.y = player.y;
});
//マウスが動いているときの処理――――――――――
canvas.addEventListener("mousemove",function(e){
 e.preventDefault();
 var rect = e.target.getBoundingClientRect();
 player.x = e.clientX - rect.left;
});
</script>
```