サイト内検索

キーワードを入

Canvas

- Canvasとは
- Canvasの使い方
- いろいろな図形を描く
- 色を指定する
- 線形グラデーションを指定する
- 円形グラデーションを指定する
- **画像を組み込む**
- Canvasリファレンス

画像を組み込む

Canvasでは、JPEGやPNGといったブラウザに対応した画像であれば、それを組み込むことができます。このコーナーでは、画像の組み込み方法と、いくつかのオプション機能について紹介していきます。

画像組み込みの基本

Canvasで画像を組み込むためには、drawImageメソッドを使います。与える引数に応じて、描画する画像の扱い方が違ってきます。

***ctx*.drawImage(*image*, *dx*, *dy*)**

***ctx*.drawImage(*image*, *dx*, *dy*, *dw*, *dh*)**

Canvasの座標 (*dx*, *dy*) を左上端として、*image* に格納されたImageオブジェクトを描画します。*dw* と *dh* が指定されると、画像は、*dw* を横幅、*dh* を縦幅としたサイズに伸縮されます。

***ctx*.drawImage(*image*, *sx*, *sy*, *sw*, *sh*, *dx*, *dy*, *dw*, *dh*)**

元の画像をトリミングしたものをCanvas上に描画します。元の画像の (*sx*, *sy*) から、横幅 *sw*、縦幅 *sh* の領域をトリミングします。トリミングされた部分画像を、Canvasの座標 (*dx*, *dy*) に、横幅 *dw*、縦幅 *dh* のサイズに伸縮して描画します。

では、早速、画像を組み込んだコードを見てみましょう。

```
onload = function() {
  draw();
};
function draw() {
  var canvas = document.getElementById('c1');
  if ( ! canvas || ! canvas.getContext ) { return fals
  var ctx = canvas.getContext('2d');
  /* Imageオブジェクトを生成 */
  var img = new Image();
  img.src = "image1.gif";
  /* 画像を描画 */
  ctx.drawImage(img, 0, 0);
```

```
}
```

このサンプルコードは、image1.gifという画像ファイルを読み込んで、Canvasの左端上に描画しようとしたものです。期待する結果は、次の通りです。

では、このコードを実際にご覧ください。[実際のサンプルを見る]

いかがでしたか？ 恐らく何も表示されなかったのではないでしょうか。コードそのものには、文法エラーがあるわけではありませんが、ブラウザではJavaScriptエラーが発生しているのです。

Imageオブジェクトは、画像が読み込まれない限り、Imageオブジェクトとしての機能を果たすことができません。にも関わらず、それを`drawImage`メソッドに引数と与えてしまったため、エラーになっているのです。

先ほどのサンプルをもう一度ブラウザで表示してみてください。おそらくちゃんと表示されたのではないでしょうか。もし画像が表示されなければ、ブラウザで再読み込みしてみてください。画像がブラウザにロードされ、それがキャッシュされれば、その後は、何度アクセスしてもエラーにはなりません。

しかし、これはでは、実際のサイトでは採用できません。次は、この解決方法をご紹介します。

画像のプリロード

先ほどの問題を解決するためには、`drawImage`メソッドを呼び出す前に、画像のロードが完了するようにしなければいけません。先ほどのサンプルを次のように改良します。

```
onload = function() {
  draw();
};
function draw() {
  var canvas = document.getElementById('c1');
  if ( ! canvas || ! canvas.getContext ) { return fals
  var ctx = canvas.getContext('2d');
  /* Imageオブジェクトを生成 */
  var img = new Image();
  img.src = "image2.gif?" + new Date().getTime();
  /* 画像が読み込まれるのを待ってから処理を続行 */
  img.onload = function() {
    ctx.drawImage(img, 0, 0);
  }
}
```

この改良版のコードをブラウザでご覧ください。見た目は同じ画像ですが、効果がはっきりとわかるように、別の画像ファイル (image2.gif) をロードするようにしました。[実際のサンプルを見る]

このコードには2つのポイントがあります。まず、img.onloadイベントハンドラを使って、画像のロードが完了してからdrawImageメソッドが

呼び出されるようにしてあります。一見、これだけで問題ないような気がしますが、実は、Internet Explorerでは問題が起こります。

Internet Explorerの場合は、初めて画像がロードされた時であればloadイベントが発生しますが、いったん画像がロードされキャッシュされてしまうと、それ以降、loadイベントが発生しません。例えば、Canvasのページを表示した後、別のページに移動したとしましょう。その後、再度、Canvasのページに戻ると、画像に対してloadイベントが発生しませんので、Canvasの描画処理が実行されず、画像が表示されないという現象が起こります。

これを解決するために、呼び込む画像のURLの最後に、`?`と`new Date().getTime()`で取り出した値を付け加えます。今回の例でいえば、`img.src` には、`image2.gif?1187146057389` という値がセットされることになります。数字の部分は、アクセスの都度、変化します。こうすることで、ブラウザに別のURLと認識させることで、キャッシュを使わず、強制的に画像をロードさせます。結果的に、ページを表示する都度、Imageオブジェクトにセットされたonloadイベントハンドラが確実に実行されるようになります。

トリミング

画像をCanvasにそのまま組み込むだけでは芸がありません。そもそも、ただ単に画像を表示させたいだけなのであれば、img要素を使えば良く、わざわざCanvasを使う必要がありません。せいぜい、Canvasの背景として使うくらいでしょう。Canvasで画像を効果的に使えるシーンとは、画像をトリミングしたい場合でしょう。ある画像の一部分を切り取り、それを好きな場所に表示することができます。

では、これまで使ってきた画像をトリミングしてみましょう。[実際のサンプルを見る]

```
onload = function() {
  draw();
};
function draw() {
  var canvas = document.getElementById('c1');
  if ( ! canvas || ! canvas.getContext ) { return fals
  var ctx = canvas.getContext('2d');
  /* Imageオブジェクトを生成 */
  var img = new Image();
  img.src = "image1.gif?" + new Date().getTime();
  /* 画像が読み込まれるのを待ってから処理を続行 */
  img.onload = function() {
    ctx.drawImage(img, 100, 80, 50, 40, 80, 60, 150, 1
  }
}
```

このスクリプトが実行されると、下図のような結果となります。

これは、元の画像の左上端から見て (100, 80) の地点から、横幅50ピクセル、縦幅40ピクセル分を切り抜いています。そして、切り抜かれた部分画像を、Canvasの (80, 60) の地点から横幅150ピクセル、縦幅120ピクセルに拡大して表示しています。

このサンプルでは、あえて切り抜いた部分画像を拡大表示していますが、もちろん、縮小表示することも可能です。ただし、ご覧のように、画像を拡大したり縮小表示すると、見た目が荒くなりますので注意してください。

　drawImageメソッドでは、このようなトリミングする場合においては9つもの引数を要求します。どの順番でどんな値をセットしなければいけないかを間違えないようにして下さい。