

アニメーションの描き方

-フレームを描く-

この場合フレームとは、HTML・ブラウザ・JavaScript の描画領域を指します。フレームは setFrame 関数によって管理します。

関数は setFrame(html,browser,js) と定義されています。

例えば、setFrame(1,1,0) と記述すると、HTML とブラウザの領域が描画され、JavaScript の領域は描画されません。

-フレームに名前をつける-

各フレームには名前をつけることができます。

フレーム名は setLabels 関数によって管理します。

関数は setLabels(html,browser,js) と定義されています。

例えば、setLabels("index.html","", "ex05-11.js") と記述すると、HTML フレームには「index.html」、JavaScript フレームには「ex05-11.js」という名前がつけます。

-ソースコードを記述する-

HTML や JavaScript のソースコードは、扱い易いように一行ずつ別の変数に格納します。格納を全て手作業で行うのは大変なので、parseText 関数を使います。parseText 関数は、parseText(no,text) と定義されています。

no に 0 を渡すと HTML、1 を渡すと JavaScript のコードとして認識されます。

text には、コードを文字列で渡します。この関数に渡す文字列の記法は少し特殊で、改行は改行文字、スペースは必ず全角、シングルクォートはエスケープします。手作業で書くのは面倒ですが、下記サイトで通常の文字列から変換できます。

<http://web.sfc.keio.ac.jp/~t13507rs/parser/>

コードをセットできたら、setText0 関数を使って実際にコードを SVG テキストとして描画します。

HTML のコードは `htmlText` 配列、JavaScript 配列に格納されます。

HTML コードの三行目を参照する場合は、`htmlText[2]` のように記述します。

-矩形を描画する-

矩形は `rectSoft` または `rectHard` 関数によって描画します。

関数は `rectSoft(x,y,width,height,color,opacity,stroke,strokeWidth)` と定義されています。

例えば、`rectSoft(630,150,100,50,"#ccc",1,"black",1)` と記述すると、(630,150) の座標に、幅 100px・高さ 50px の矩形を、色 #ccc、透明度 1(透明度は 0~1)、枠線を黒、枠線の太さを 1px で描画します。色指定は、16 進数か RGB 値、もしくは色名で行います。

また、`rectSoft` 関数は少し丸みを帯びた矩形を、`rectHard` 関数は角が直角な矩形を描画します。

これらの関数は SVG オブジェクトを返すので、それを変数に入れたり、後述する `addComponent` 関数の引数として使います。

-SVG テキストを描画する-

SVG テキストの描画は、`labelUI` 関数で行います。

関数は `labelUI(x,y,font-size,font-family,content,font-color)` と定義されています。

例えば、`addComponent(labelUI(635,180,13,"Helvetica","ここをクリック","black"))` と記述すると、座標 (635,180) にフォントサイズ 13px、書体を Helvetica、色を黒で、「ここをクリック」という SVG テキストを描画します。この関数は SVG オブジェクトを返すので、それを変数に入れたり、後述する `addComponent` 関数の引数として使います。

-コンポーネントを追加する-

ブラウザ領域に描画するオブジェクトをコンポーネントと呼びます。

コンポーネントは、`addComponent` 関数を用いて追加します。

関数に SVG オブジェクトを渡すと、そのオブジェクトがコンポーネントとして登録されます。コンポーネントは配列に格納されるので、`component[0]` のように参照できます。

例えば、`addComponent(labelUI(630,100,25,"","イベントハンドラの練習","black"))` と記述すると、「イベントハンドラの練習」という SVG テキストをコンポーネントに追加できます。

-SVG をハイライトする-

`highlight` 関数に SVG オブジェクトを渡すと、SVG オブジェクトを赤くハイライトします。

例えば、`highlight(htmlText[9])` と記述すると HTML コードの 10 行目がハイライトされます。

また、`defuse` 関数に渡すとハイライトを解いてもとの色に戻ります。

-コンポーネントを表示する-

コンポーネントはデフォルトで透明度が 0 になっています。そのため、必要に応じて `setVisible` 関数を用いて透明度を上げます。

関数は `setVisible(obj,rate)` と定義されています。

例えば、`setVisible(component[0],1)` と記述すると、一つ目のコンポーネントの透明度が 1 になります。

また、`setVisible(component[0],0)` と記述すると、一つ目のコンポーネントの透明度が 0 になります。

-矢印の管理-

矢印はデフォルトで透明度 0、つまり非表示になっています。

`showArrow()` で表示、`hideArrow()` で非表示にできます。

また、矢印の座標は `setArrow` 関数で行います。

関数は `setArrow(obj,obj2)` と定義されています。

例えば、`setArrow(html[9],component[0])` と記述すると、HTML コードの十行目から、一つ目のコンポーネントに向かって矢印がひかれます。

-フレームをスクロールさせる-

`scroll` 関数を使うと、各フレームの内容をスクロールすることができます。

`scroll(0)` で HTML、`scroll(1)` でブラウザ、`scroll(2)` で JavaScript のフレームをスクロールします。

一度の関数呼び出しにつき、HTML と JavaScript は一行分、ブラウザはコンポーネント約 1 個分スクロールします。コンポーネントひとつの高さを 50px としているので、コンポーネントは 50px おきに配置すると綺麗にスクロールできます。

-毎フレーム処理-

update 関数内に記述したコードは毎フレーム実行されます。アニメーションの自動再生などに活用できます。

なお、この関数がないとエラーが出るので、使わない場合も、必ずどこかに `update()` と書くようにしてください。