SVG 資料第 7 回目 (その 2)

クロージャ

メディア専門ユニット I(SVG)

2017/6/6

クロージャ

- JavaScript では関数もオブジェクトなので関数内で定義された関数はローカルな関数
- ローカルに定義された関数が関数の外のローカルな変数を参照すると、そのローカルな関数を通じて参照、変更が可能
- 関数とその実行環境を取りまとめたものをクロージャという。
- 上記のローカルな関数のクロージャには関数から参照可能なローカルに定義された変数も含まれる
- クロージャを使うことでグローバル変数の使用を減少させることができる

アニメーション版サイクロイドを描く–クロージャ版


```
 6window.onload = function() {
 7  let C =document.getElementById("cycliod");
 8  let T =document.getElementById("translate");
 9  let Rot =document.getElementById("rotate");
10  let R = 50, Current = 1, Step=2, d ="M0,0 ";
11  let drawCurve = function (){
12      let Next = Current + Step, rad;
13      if(Current<=360) {
14        for( ; Current< Next; Current++) {
15          rad = Current/180*Math.PI;
16          d += `${R*(rad-Math.sin(rad))},${R*(-1+Math.cos(rad))} `;
17        }
18        C.setAttribute("d",d);
19        T.setAttribute("transform",`translate(${R*rad},-50)`);
20        Rot.setAttribute("transform", `rotate(${Next})`);
21        setTimeout(drawCurve,100);
22      }
23    }
24  drawCurve();
25}
```

アニメーション版サイクロイドを描く–クロージャ版解説

- 以前のものと異なるのはグローバル変数になっていたものがすべて `window.onload` の関数定義内に含まれていること
- 11 行目で関数 `drawCurve` を定義
- 関数の中身は以前とほとんど同じ
- 24 行目で初めの値でサイクロイドを描き始める

アニメーション版サイクロイドを描く クロージャ版 (1)

```
 6 window.onload = function() {
 7   let C =document.getElementById("cycloid");
 8   let T =document.getElementById("translate");
 9   let Rot =document.getElementById("rotate");
10   let R = 50, Step=2, d ="M0,0 ";
11   (function drawCurve(Current){
12       let Next = Current + Step, rad;
13       if(Current<=360) {
14         for( ; Current< Next; Current++) {
15           rad = Current/180*Math.PI;
16           d += `${R*(rad-Math.sin(rad))},${R*(-1+Math.cos(rad))} `;
17         }
18         C.setAttribute("d",d);
19         T.setAttribute("transform",`translate(${R*rad},-50)`);
20         Rot.setAttribute("transform", `rotate(${Next})`);
21         setTimeout(drawCurve,100,Next);
22       }
23     })(1);
24 }
25 //]]>
26 </script>
```

アニメーション版サイクロイドを描く クロージャ版 (2)

定義した関数をその場で実行

ここでは定義した関数をその場で実行するという技法を用いている

- 11 行目から 23 行目で関数 drawcurve が引数付きで定義
- その前後が () で囲まれていることに注意 (23 行目の前の) に対応)
- これによりこの関数が実行される。
- 23 行目の (1) はこの関数の引数
- したがって 11 行目の仮引数 Current ははじめに 1 で実行される
- setTimeout 関数の 3 番目の引数 Next はこの関数で呼び出される関数 (ここでは drawCurve) の引数となる

関数の arguments について

- 関数が定義されるとその中では仮引数のリストが格納されている arguments オブジェクトが作成される
- 最新の EcmaScript の仕様では strict mode ではエラーとなる
- strict mode とは Ecmascript 5 版で導入されたエラーチェックとセキュリティが強化された実行環境
- したがって、ここでは arguments オブジェクトについては解説をしない
- 新しい EcmaScript では arguments オブジェクトの代わりに、残余仮引数という概念が導入されている。

残余仮引数の使い方

```
function mulAll(v, ...Args) {
  for(let i=0; i<Args.length; i++) {
    v *= Args[i];
  }
  return v;
}
```

- この関数では第 1 の引数が v で、残りの引数は Args という配列に引き渡される。
- 残余変数の前にはピリオド (.) を 3 つ置く。
- 残余変数は配列なので通常の使い方ができる。

やってみよう

- クロージャ版のアニメーションサイクロイドを実行中に、コンソールで 7 行目や 8 行目で定義してある変数を参照できるか確認する。
- グローバル変数を減らすメリットは何か？
- 関数を定義してその場で実行することのメリットは何か
- 今まで作成したものからグローバル変数を減らしたものを作成する