SVG 資料第 4 回目 (その 3)

DOM の操作とイベント (2)

メディア専門ユニット I(SVG)

2017/5/17

クリックした位置に円を移動

自分以外の要素の属性値を変える

```
 1<?xml version="1.0" encoding="UTF-8" ?>
 2<svg xmlns="http://www.w3.org/2000/svg"
 3     xmlns:xlink="http://www.w3.org/1999/xlink"
 4     height="100%" width="100%">
 5        <title>マウスをクリックした位置に円が移動</title>
 6  <script type="text/ecmascript">
 7//    <![CDATA[
 8  window.onload = function() {
 9    document.getElementById("Canvas").addEventListener("click",click, false);
10  }
11  function click(E) {
12  let T = document.getElementById("Circle");
13    T.setAttribute("cx",E.clientX);
14    T.setAttribute("cy",E.clientY);
15  }
16//    ]]></script>
17  <rect x="0" y="0" width="100%" height="100%" fill="white" id="Canvas" />
18  <circle id="Circle" cx="50" cy="50" r="20" fill="red"
19          stroke="black" stroke-width="2" />
20</svg>
```

ソースコードの解説

- 17 行目に画面全体を覆う長方形を定義
- この長方形の属性 id に Canvas が設定
- 9 行目でこの長方形にイベント処理関数を定義
- 18 行目から 19 行目に属性 id が Circle である円を定義
- イベント処理関数の処理手順はつぎのとおり
  - 12 行目で円の要素を得ている
  - その要素の属性 cx にイベントが発生したときのマウスカーソル位置 (E.clientX) を設定
  - その要素の属性 cy に対しても同様
- このコードでは円に対してクリックイベントの処理関数が定義されていないので円の上でクリックしても円は移動しない

やってみよう

- 円に対してもイベント処理関数を登録してその上をクリックしたときに移動するようにする
- 円の上をクリックしたら色が変化する
- 2 つ以上の要素を置く。要素以外のところでクリックしたら最後にクリックした要素がクリックした位置に移動する

イベント処理を登録する要素について (配布資料 144 ページ)

- イベントを処理関数は個々の要素につける必要はない
- ある要素上で起きたイベントは親要素にも伝えられる
  - イベントの発生は一番外の要素で捉えられる。
  - そこから順に子要素をたどって、イベントの発生が伝播する。
    この段階をイベントキャプチャリングという。
  - イベントが発生した要素で伝播が逆向きに親要素の方向に向かっ伝播する。
    この段階をイベントバブリングという。
  - `addEventListener`3 番目の引数が `true` のときはイベントキャプチャリングの段階で、`false` のときはイベントバブリングの段階で処理される。

円の上をクリックすると円の色を表示 (3)

```
 1<?xml version="1.0" encoding="UTF-8" ?>
 2<svg xmlns="http://www.w3.org/2000/svg"
 3      xmlns:xlink="http://www.w3.org/1999/xlink"
 4      height="100%" width="100%">
 5   <title>クリックするとメッセージボックスが表示 (改良版)</title>
 6   <script type="text/ecmascript">
 7//     <![CDATA[
 8   window.onload = function() {
 9     let Cs = document.getElementById("Canvas");
10     Cs.addEventListener("click",click, false);
11   }
12   function click(event) {
13     alert(‘Circle ${event.target.getAttribute("fill")} clicked.‘);
14   }
15//  ]]></script>
16   <g id="Canvas">
17     <circle cx="50" cy="50" r="20" fill="red"/>
18     <circle cx="100" cy="50" r="20" fill="blue"/>
19     <circle cx="150" cy="50" r="20" fill="green"/>
20   </g>
21</svg>
```

円の上をクリックすると円の色を表示する
イベント処理をつける要素について

- 3 つの円を子要素に持つ<g>要素を追加 (16 行目)
- <g>要素の属性 `id` が `Canvas`
- 9 行目で指定された属性 `id` を持つ要素を `getElementById` メソッドで得る
- 10 行目でその要素に対してイベントリスナーをつける

やってみよう

16 行目の<g>要素を取り除いて、<svg>要素に属性 id を Canvas にすると正しく動くか確認する

マウスのドラッグを処理

- マウスのイベントにドラッグというイベントはない
- マウスのドラッグを処理とはマウスのボタンが押されたままマウスカーソルを移動することなので次のように処理
  - マウスのボタンが押される (mousedown イベント)
  - マウスカーソルが移動している間は mousemove イベントが発生
  - マウスボタンを離す (mouseup イベント)
- ここでは mousedown イベントが発生したら要素に mousemove イベント処理関数を登録し、マウスボタンが離されたらその処理関数を取り除くという処理を行う

マウスのドラッグを処理

```
 1 <?xml version="1.0" encoding="UTF-8" ?>
 2 <svg xmlns="http://www.w3.org/2000/svg"
 3      xmlns:xlink="http://www.w3.org/1999/xlink"
 4      height="100%" width="100%">
 5   <title>円をドラッグで移動する</title>
 6   <script type="text/ecmascript">
 7 //    <![CDATA[
 8   let G, inDragging;
 9   window.onload = function() {
10     G = document.getElementsByTagName("svg")[0];
11     G.addEventListener("mousedown",mdown, false);
12     G.addEventListener("mouseup",mup, false);
13   }
14   function mdown(event) {
15     inDragging = event.target;
16     G.addEventListener("mousemove",mmove, false);
17   }
18   function mmove(event) {
19     inDragging.setAttribute("cx",event.clientX);
20     inDragging.setAttribute("cy",event.clientY);
21   }
22   function mup(event) {
23     G.removeEventListener("mousemove", mmove, false);
24   }
25 //    ]]></script>
26   <circle cx="50" cy="50" r="20" fill="red"/>
27   <circle cx="100" cy="50" r="20" fill="blue"/>
28   <circle cx="150" cy="50" r="20" fill="green"/>
29 </svg>
```

マウスのドラッグを処理–ソースコード解説 (1)

- ファイルがロードされた後で実行する関数が 9 行目から 13 行目で定義
  - 10 行目で<svg>要素のリストを getElementsByTagName を用いて得ている。この条件を満たすものはひとつしかないので [0] でその要素が得られる。
  - この要素に対して、11 行目で mousedown の、12 行目で mouseup のイベント処理関数を登録
- 14 行目から 18 行目で mousedown のイベント処理関数 mdown を定義
  - 15 行目でイベントが発生した要素を変数 inDragging に保存
  - 16 行目でイベントが捕らえられたオブジェクトをコンソールに出力
  - alert と異なり、実行がそこで中断しない
  - 出力結果は「開発者ツール」の Console で確認できる
  - 17 行目でこの要素に mousemove のイベント処理関数を定義

マウスのドラッグを処理–ソースコード解説 (2)

- 19 行目から 22 行目で mousemove のイベント処理関数を定義
  ここでは mousemove のイベントで得られたカーソル位置をドラッグ開始時に保存した円の中心座標に設定
- 23 行目から 25 行目で mouseup の処理関数を定義
  ここでは mousemove のイベント処理関数を取り除く

mousemove のイベント処理関数では event.target の要素に対して属性を定義していないことに注意

マウスのドラッグを処理 (改良版)

```
 1 <?xml version="1.0" encoding="UTF-8" ?>
 2 <svg xmlns="http://www.w3.org/2000/svg"
 3      xmlns:xlink="http://www.w3.org/1999/xlink"
 4      height="100%" width="100%">
 5   <title>円をドラッグで移動する--改良版</title>
 6   <script type="text/ecmascript">
 7 //    <![CDATA[
 8   let G, inDragging;
 9   window.onload = function() {
10     G = document.getElementsByTagName("svg")[0];
11     G.addEventListener("mousedown",mdown, false);
12     G.addEventListener("mouseup",mup, false);
13   }
14   function mdown(event) {
15     if(event.target.nodeName !== "svg") {
16       inDragging = event.target;
17       G.appendChild(inDragging)
18       G.addEventListener("mousemove",mmove, false);
19     }
20   }
21   function mmove(event) {
22     inDragging.setAttribute("cx",event.clientX);
23     inDragging.setAttribute("cy",event.clientY);
24     event.stopPropagation();
25   }
26   function mup(event) {
27     G.removeEventListener("mousemove",mmove, false);
28   }
29 //    ]]></script>
30     <circle cx="50" cy="50" r="20" fill="red"/>
31     <circle cx="100" cy="50" r="20" fill="blue"/>
32     <circle cx="150" cy="50" r="20" fill="green"/>
33 </svg>
```

マウスのドラッグを処理 (改良版)–ソースコード解説

以前のコードに対し、18 行目に appendChild メソッドを使用。

- appendChild はメソッドの要素に対して引数の要素を一番最後の子要素として追加する
- 引数の要素がすでにすでにどこかのの子要素となっていた場合にはそれがコピーされず、移動となる

確認しよう DOM ツリーを開いて、ドラッグした円が一番最後の要素になっているかどうか確認しよう

やってみよう

- 正方形をドラッグするように変える
- 文字をドラッグするように変える
- 通常、Windows 上でアイコンなどをドラッグするときと、ここでのドラッグするリストの相違点を述べ、それを改善する
- <path>要素で描かれた図形をドラッグするものを作成する
- 円と正方形の図形に対し、1 種類のイベント処理関数でする方法を考えよ。図形の種類が増えてもイベント処理関数に手を付けないようにするにはどのような方法があるか検討すること