ソフトウェア開発
第 10 回目授業

平野 照比古

2015/11/27

イベントとは

- イベントとはプログラムに対して働きかける動作
- マウスボタンが押された (クリック) などのユーザからの要求
- 一定時間経ったことをシステムから通知される (タイマーイベント)
- プログラムが開始されるということ自体イベントである。このイベントは初期化をするために利用される。

イベントドリブンなプログラム

- イベントの発生する順序はあらかじめ決まっていない
- それぞれのイベントを処理するプログラムは独立している必要

イベントの発生を順次処理していくプログラムのモデルをイベントドリブンなプログラムという。

イベントハンドラー

- イベントを処理する関数 (イベントハンドラー) を適当な要素に登録
- 属性に関数を登録する方法と、メソッドを用いて要素に登録する方法がある。
- この授業ではメソッドを通じて登録する方法だけを紹介

イベントハンドラーを登録するメソッド

addEventListener(string *type* ,functiion *listener* ,[boolean *useCapture*])
引数の意味は次のとおり

- *type* はイベントの種類を示す文字列であり、イベントの属性名から on を取り除いたもの
- *listener* はイベントを処理する関数 (イベントハンドラー)
- *useCapture* はイベントの処理順序 (詳しくは後述)

イベントハンドラーの関数は通常、イベントオブジェクトを引数に取るように定義する。

イベントハンドラーの削除

`removeEventListener(string` *type* `,functiion` *listener* `,[boolean` *useCapture* `])`
イベントハンドラーが削除されるためには登録したときと同じ条件であることが必要

ドキュメントの onload イベント

Web ブラウザは HTML 文書を次のような順序で解釈し、実行する

1. 初めに document オブジェクトを作成し、Web ページの解釈を行う。
2. HTML 要素やテキストコンテンツを解釈しながら要素やテキスト (ノード) を追加する。この時点では、document.readystate プロパティは"loading"という値を持つ。
3. <script>要素が来ると、このこの要素を document に追加し、スクリプトを実行する。したがって、この時点で存在しない要素に対してアクセスすることはできない。つまり、<script>要素より後に書かれている要素のはアクセスできない。
4. したがって、この段階では後で利用するための関数の定義やイベントハンドラーの登録だけをするのがふつうである。

ドキュメントの onload イベント

6. ドキュメントが完全に解釈できたら、document.readystate プロパティは"interactive"という値に変わる。
7. ブラウザはドキュメントオブジェクトに対し DOMContentLoaded イベントを発生させる。
8. この時点では画像などの追加コンテンツの読み込みは終了していない可能性がある。
9. それらのことが終了すると、document.readyState プロパティの値が"complete"となり、window オブジェクトに対して load イベントを発生させる。

マウスイベントのプロパティ

イベントの発生条件	イベントの属性名
ボタンがクリックされた	onclick
ボタンが押された	onmousedown
マウスカーソルが移動した	onmousemove
マウスボタンが離された	onmouseup
マウスカーソルが範囲に入った	onmouseover
マウスカーソルが範囲から出た	onmouseout

マウスイベントのプロパティ(1)

プロパティ	型	意味
target	EventTarget	イベントが発生したオブジェクト
currentTarget	EventTarget	イベントハンドラーが登録されているオブジェクト

マウスイベントのプロパティ(2)

screenX	long	マウスポインタのディスプレイ画面における x 座標
screenY	long	マウスポインタのディスプレイ画面における y 座標
clientX	long	マウスポインタのクライアント領域における相対的な x 座標 (スクロールしている場合には window.pageXOffset を加える)
clientY	long	マウスポインタのクライアント領域における相対的な y 座標 (スクロールしている場合には window.pageYOffset を加える)
pageX	long	マウスポインタの document 領域における相対的な x 座標
pageY	long	マウスポインタの document 領域における相対的な y 座標

マウスイベントのプロパティ(3)

ctrlKey	boolean	cntrl キーが押されているか
shiftKey	boolean	shift キーが押されているか
altKey	boolean	alt キーが押されているか
metaKey	boolean	meta キーが押されているか
button	unsigned short	マウスボタンの種類、0 は左ボタン、1 は中ボタン、2 は右ボタンを表す。
eventPhase	unsigned short	イベント伝播の現在の段階を表す。Event.CAPUTURING_PHASE(1)、Event.AT_TARGET(2)、Event.BUBBLING_PHASE(3) の値をとる

DOM Level 2 のイベント処理モデル

あるオブジェクトの上でイベントが発生すると次の順序で各オブジェクトにイベントの発生が伝えられる。

1. 発生したオブジェクトを含む最上位のオブジェクトにイベントの発生が伝えられる。このオブジェクトにイベント処理関数が定義されていなければなにも起きない。
2. 以下順に DOM ツリーに沿ってイベントが発生したオブジェクトの途中にあるオブジェクトにイベントの発生が伝えられる。(イベントキャプチャリング)
3. イベントが発生したオブジェクトまでイベントの発生が伝えられる。
4. その後、DOM ツリーに沿ってこのオブジェクトを含む最上位のオブジェクトまで再びイベントの発生が伝えられる (イベントバブリング)。

- addEventListner の 3 番目の引数で false にするとイベント処理はイベントバブリングの段階で呼び出される。
- true にするとイベント処理はイベントキャプチャリングの段階で呼び出される。
- イベントの伝播を途中で中断させるメソッドが stopPropagation()
- ブラウザの画面で右クリックをすると、ページのコンテキストメニュが表示される。このようなデフォルトの操作を行わせないようにする方法が preventDefault()

ユーザの入力を取り扱う例

大きさが指定された div 要素が 3 つ並び、その横にプルダウンメニュー、ラジオボタン、テキスト入力エリアと設定ボタンが並んでいる。このページでは次のことができる。

- 左にある 3 つの領域でマウスをクリックすると、クリックした位置の情報が右側下方の 8 つのテキストボックスに表示
- 上の 6 つはイベントオブジェクトのプロパティをそのまま表示
- 最下部の値はそれぞれの領域から見た相対位置を表示
- 右最上のプルダウンメニューからは一番左の領域の背景色を変えることが可能
- ラジオボタンで選択された色が中央の領域の背景色になる。
- テキストボックスには CSS3 で定義された色の指定方法で右の領域の背景色を設定できる。設定するためには隣の「設定」ボタンを押す必要

ファイルのリスト (1)

```
<!DOCTYPE html>
<head>
<meta charset="UTF-8"/>
<title>イベント処理</title>
<link rel="stylesheet" type="text/css" href="event.css"/>
<script type="text/ecmascript" src="event.js"></script>
</head>
```

CSS ファイル、JavaScript ファイルが外部ファイル

ファイルのリスト (2)

```
<body>
  <div class="block" id="Squares">
    <div></div><div></div><div></div>
  </div>
```

ファイルのリスト (3)

```
<form>
  <select id="select">
    <option value="red">赤</option>
    <option value="yellow">黄色</option>
    <option value="blue">青</option>
  </select>
  <div id="radio">
    <div><input type="radio" value="red" name="color">赤</input></div>
    <div><input type="radio" value="yellow" name="color">黄</input></div>
    <div><input type="radio" value="blue" name="color">青</input></div>
  </div>
  <div>
    色名<input type="text" size="10" id="colorName"></input>
    <input type="button" value="設定" id="Set"></input>
  </div>
```

ファイルのリスト (4)

```
  <div id="position">
    <div>クリック位置</div>
    <div><div>clientX</div><input type="text" size="3" class="click">
      <div>clientY</div><input type="text" size="3" class="click"></div>
    <div><div>pageX</div><input type="text" size="3" class="click">
      <div>pageY</div><input type="text" size="3" class="click"></div>
    <div><div>pageXOffset</div><input type="text" size="3" class="click">
      <div>pageYOffset</div><input type="text" size="3" class="click"></div
   <div>from Boundary</div>
    <div><div> Left</div><input type="text" size="3" class="click">
      <div>Top</div><input type="text" size="3" class="click"></div>
   </div>
 </form>
</body>
</html>
```

CSS ファイル (1)—左部分

```
.block {
  display : inline-block;
  vertical-align : middle;
}
.block > div {
  width : 200px;
  height : 200px;
  display : inline-block;
}
```

CSS ファイルの解説 (1)—左部分

- 左側の 3 つの領域は id が block である要素の直下の div 要素であることから 5 行目のセレクタが適用される。
  - これらの 3 つの部分は大きさが 200px の正方形となっている (6 行目と 7 行目)。
  - 横に並べるようにするため、display を inline-block に指定している。
  - 背景色 (background) はプログラムから設定される。
- これらの領域全体を囲む div の垂直方向の配置の位置が 3 行目で定められている。

CSS ファイル (2)—右部分

```
#radio {
  width:80px;
}
.click {
  text-align:right;
}
form {
  display : inline-block;
  vertical-align : middle;
}
input, select{
  font-size:25px;
}
```

CSS ファイル (3)—右部分 (続き)

```
#position > div {
  text-align:center;
  font-size: 20px;
}
#position > div > div {
  display:inline-block;
  width: 90px;
  text-align: right;
  font-size: 18px;
}
```

CSS ファイルの解説 (2)—右部分

- 右上方のプルダウンメニューのフォントの大きさは 20px に定義されているが、文書のロード時に 30px に変更している。
- ラジオボタンの要素の幅は#radio のセレクタで定められている。
- クリックしたときの位置情報を示す部分の CSS は#position で始まるセレクタで定められている。
  - テキストボックスがない行は#position > div が適用されている。
  - テキストボックスがある行は、テキストボックスの位置をそろえるために、その前の文字列を div 要素の中に入れ、幅、テキストの配置位置とフォントの大きさを指定 (#position > div >div)。

JavaScript(1)

```
1   window.onload = function() {
2     var Squares   = document.getElementById("Squares");
3     var Select    = document.getElementById("select");
4     var ColorName = document.getElementById("colorName");
5     var Radio     = document.getElementById("radio");
6     var Set       = document.getElementById("Set");
7     var Inputs    = document.getElementsByClassName("click");
8     var lastClicked = Squares.children[1];
```

ここでは id 属性を持つ要素すべてを得ている。

JavaScript(2)

```
 9      Squares.children[0].style.background = "red";
10      Squares.children[1].style.background = "yellow";
11      Squares.children[2].style.background = "blue";
12      Select.style.fontSize = "30px";
13
```

- id が Squares である div 要素の下には 3 つの div 要素がある。これらの要素の背景色を指定するために、children プロパティを用いて参照
- スタイルを変更するためには style プロパティの後に属性を付ける。
- 9 行目から 11 行目では左の 3 つの領域の背景色 (background) を設定
- フォントの大きさを指定する CSS 属性は font-size であるが、これは-を含んでいる。このような属性は-を省き、次の単語を大文字で始める。この場合には fontSize となる。

```
14      Squares.addEventListener("click",function(E){
15        Inputs[0].value = E.clientX;
16        Inputs[1].value = E.clientY;
17        Inputs[2].value = E.pageX;
18        Inputs[3].value = E.pageY;
19        Inputs[4].value = window.pageXOffset;
20        Inputs[5].value = window.pageYOffset;
21        var R = E.target.getBoundingClientRect();
22        Inputs[6].value = E.pageX-R.left;
23        Inputs[7].value = E.pageY-R.top;
24          },false);
```

この領域がクリックされたときのイベントハンドラーを定義している。

- 表示するためのテキストボックスのリスト得る (7 行目)
- テキストボックスに該当する値を代入 (14 行目から 19 行目)
- 21 行目から 23 行目では該当する領域からの相対位置を求めている。それぞれの領域がページの先頭からどれだけ離れているかを求めるメソッドが getBoundingClientRect()

getBoundingClientRect() の戻り値

getBoundingClientRect() メソッドは ClientRect オブジェクトを返す。

プロパティ	解説
top	領域の上端の Y 座標
bottom	領域の下端の Y 座標
left	領域の左端の X 座標
right	領域の右端の X 座標
width	領域の幅
height	領域の高さ

これらの値とイベントが起きた位置の情報から領域内での位置が計算できる。

イベントハンドラーの定義–プルダウンメニュー

ここでは 3 つの入力方法に対して、イベントハンドラーを定義している。

```
25    Select.addEventListener("change", function(){
26      lastClicked.style.background = Select.value; },false);
```

プルダウンメニューに change イベントのハンドラーを定義している。左側の部分の背景色を変えている。

イベントハンドラーの定義–ラジオボタン

```
27    Radio.addEventListener("click", function(E){
28      alert(E.target.tagName);
29      if(E.target.tagName === "DIV") {
30        E.target.firstChild.checked = true;
31      }
32      lastClicked.style.background = Radio.querySelector("input
33    },false);
```

25 行目から 31 行目はラジオボタンのイベントハンドラーを設定。
ラジオボタンは name 属性が同じグループとして扱われる (どこか一つだけオンになる)。

- 27 行目ではクリックされた場所の要素名を表示 (通常は必要ない)。HTML の要素名は大文字に変換されることの確認。
- ここでは HTML のリストの 28 行目にある `div` 要素にクリックのイベントハンドラーを登録するので、クリックされた場所がラジオボタンの上でなくても、この範囲内であればイベントは発生
- クリックされた場所がラジオボタンの上でなければ (このときは、ラジオボタンの親要素の `div` 要素が `E.target` となる) のでその `firstChild` が値を変えるラジオボタンになる (28 行目)。
- ラジオボタンの集まりには `value` プロパティがないので、チェックされている場所を探すために `querySelector("input:checked")` を用いる (31 行目の右辺)。

イベントハンドラーの定義–テキストボックス

```
34    Set.addEventListener("click", function(){
35      lastClicked.style.background = ColorName.value; }, false)
36    }
```

テキストボックスの値を設定するためには記入されたテキストをそのまま設定すればよい。

日付

3 つのプルダウンメニューが順に年、月、日を選択できる
年や月が変化 (change イベントが発生) すると日のプルダウンメニューの日付のメニューがその年月にある日までに変わるようになっている。

リスト (1)

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
5  <title>日付</title>
6  <script type="text/ecmascript">
7  //<![CDATA[
8    window.onload = function(){
9      function makeSelectNumber(from, to, prefix, suffix, id, parent){
10       var i, option;
11       var Select = makeElm("select", {"id":id}, parent);
12       for(i=from; i<=to; i++) {
13         option = makeElm("option",{"value":i}, Select);
14         makeTextNode(prefix+i+suffix,option);
15       }
16       return Select;
17     };
```

この JavaScript のプログラムは 8 行目で定義されている window.onload のイベントハンドラーの中に入っている。

リスト (1)—解説

9 行目から 17 行目では指定された範囲の値を選択できるプルダウンメニューを作成する関数 makeSelectNumber() を定義している。

- 引数は順に、下限値 (from)、上限値 (to)、数字の前後に付ける文字列 (prefix と suffix)、プルダウンメニュー の id 属性の属性値 (id) と親要素 (parent) である。
- 11 行目で select 要素を作成している。作成のために makeElm 関数 (18 行目以降で定義) を呼び出している。
- 12 行目から 15 行目で option 要素を順に作成している。
- 14 行目でプルダウンメニューに表示される文字列をテキストノードを作成している (関数 makeTextNode())。

リスト (2)

```
18      function makeElm(name, attribs, parent) {
19        var elm = document.createElement(name);
20        var attrib;
21        for(attrib in attribs) {
22          elm.setAttribute(attrib,attribs[attrib]);
23        };
24        if(parent) parent.appendChild(elm);
25        return elm;
26      }
```

リスト (2)—解説

与えられた要素を作成する関数 `makeElm()` を定義

- 引数は順に、要素名 (`name`)、属性値のリスト (`attribs`) と親要素の指定
- 19 行目で指定された要素を作成
- 21 行目から 23 行目で与えられた属性とその属性値がメンバーになっているオブジェクトの値を順に選んで属性値を設定
- 24 行目では指定された親オブジェクトが有効な場合には、作成した要素を子要素に付け加えている。
- 25 行目で作成した要素を戻り値としている。

リスト (3)

```
27          function makeTextNode(text,parent) {
28              parent.appendChild(document.createTextNode(text));
29          };
```

関数 makeTextNode() は指定された文字列を基にテキストノードを作成し、指定された親要素の子要素に設定している。

リスト (4)

```
30      var Form  = document.getElementById("menu");
31      var Year  = makeSelectNumber(2000,2020,"","年","year", Form);
32      var Month = makeSelectNumber(1,12,"","月","month", Form);
33      var Days = [];
34      for(i=28; i<=31; i++) {
35        Days[i] = makeSelectNumber(1,i,"","日","day");
36      }
```

59 行目の form 要素内にプルダウンメニューを作成する。プルダウンメニューで表示される日付は実行当日になるように設定される。

リスト (4)—解説

- 30 行目では form 要素を得ている。
- 38 行目と 39 行目ではそれぞれ年、月のプルダウンメニューを作成して form 要素の子要素にしている。
- 日については 28 日から 31 日まであるプルダウンメニューを 4 つ作成して配列に格納している。
- ここでの関数呼び出しは最後の親要素がないので、いずれも子要素としては登録されない。

リスト (5)

```
37        var i, today = new Date();
38        var y = today.getFullYear();
39        var m = today.getMonth();
40        var d;
41        Year.value = y;
42        Month.value = m+1;
43        d = new Date(y, m+1,0).getDate();
44        Form.appendChild(Days[d]);
45        Form.children[2].value = today.getDate();
```

ここでは、実行時の日付がプルダウンメニューの初期値として表示されるようにしている。

リスト (5)—解説

- Date() オブジェクトを引数なしでよぶと、実行時の時間が得られる。この時間は 1970 年 1 月 1 日午前 0 時 (GMT) からの経過時間であり、単位はミリ秒。
- 38 行目で年を得ている。Date オブジェクトには getYear() というメソッドも存在するが、これは西暦の下 2 桁しか返さないので使わないこと。
- 39 行目で月を得ている。getMonth() メソッドの戻り値は 0(1 月) から 11(12 月) であることに注意
- 41 行目と 42 行目では、得られた年と月をそれぞれのプルダウンメニューに設定している。月の値を 1 増やしていることに注意

リスト (5)—解説 (続き)

- Date() オブジェクトに年、月、日 (さらに、時、分、秒もオプションの引数として与えられる) を与えて、経過時間を得るとができる。ここここでは実行当日の翌月の 0 日を指定することで翌月の 1 日の 1 日前の日付に設定できる。その日付から日を得る (getDate() メソッド) ことで当月の日数がわかる (43 行目)。
- Date オブジェクトには Day() というメソッドも存在する。これは曜日を得るメソッドで 0(日曜日) から 6(土曜日) の値が返る。
- 44 行目ではこの日数を持つプルダウンメニューを子要素として追加している。
- 45 行目で、日の選択する値を設定している。

リスト (6)

```
46      Form.addEventListener("change", function(){
47        var d2 = Form.children[2].value
48        d = new Date(Year.value, Month.value, 0).getDate();
49        if( d != Form.children[2].children.length) {
50          Form.replaceChild(Days[d],Form.children[2]);
51          Form.children[2].value = Math.min(Form.children[2].length, d2);
52        }
53      },false);
54    }
55    //]]>
56    </script>
57    </head>
```

リスト (6)—解説

プルダウンメニューの値が変化したイベント (change) ハンドラーを登録している。

- イベントハンドラーは `form` 要素につけている。
- 47 行目で、現在の日を保存している。
- 48 行目で、現在、プルダウンメニューで設定されている年月の日数を求めている。
- この日数と現在表示されている日数のプルダウンメニューの日数が異なる場合には (49 行目) 子ノードを入れ替える (50 行目)。さらに、初期値を現在のままか、月の最終日の小さいほうに設定する (51 行目)。

リスト (7)

```
58    <body>
59      <form id="menu"></form>
60    </body>
61    </html>
```

ここでは HTML 文書内で表示される要素を記述している。ここでは body 要素内に form 要素があるだけである。