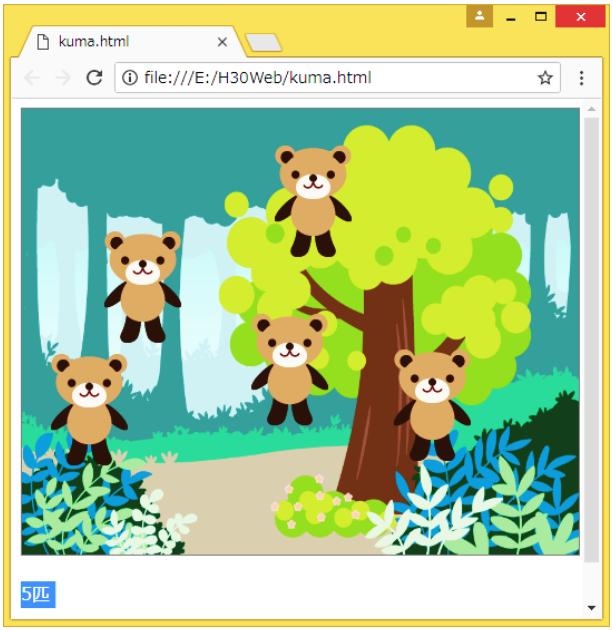
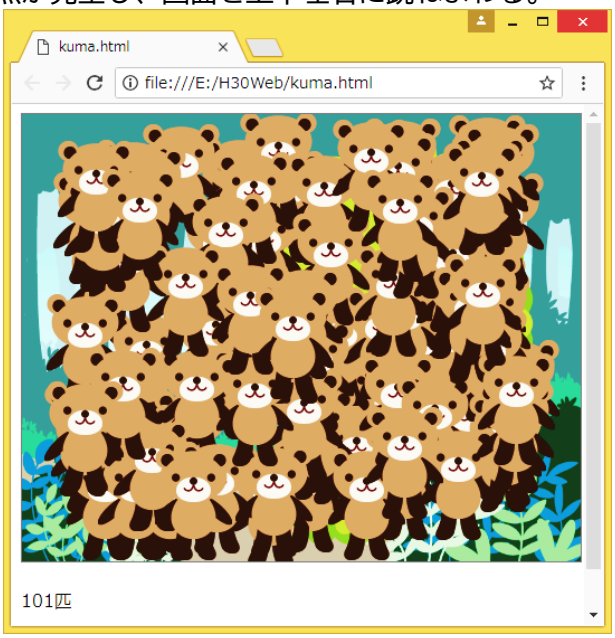
●オブジェクトの大量発生

canvas 内で任意の場所をｸﾘｯｸするとその地点で熊が発生し、画面を上下左右に跳ねまわる。



↑５回クリックした場合は５匹



↑101 回クリックしたら 101 匹クマちゃん

≪メモ≫
- 実際はわずかながら遅くなっている。
- よほど多くないと人間には感じない。
- 一つのタイマーですべての熊を動かしている。
- ブラウザが進歩したおかげ。
- ブラウザ内だけで処理しているので速い。
  └→クライアントサイドでの処理と言う。

←1000 匹発生させても重くならない！（やってみよう）

●html 文

```
<!DOCTYPE html>
<html lang="ja">
<head>
        <meta charset="UTF-8">
        <style>
        #canvas {                                          ←CANVAS の見た目
                background-color:white;                    ←背景色：白
                border: 1px solid gray;                    ←境界線：グレー
        }
        </style>
        <script src="objkuma.js"></script>                 ←動かすスクリプト
</head>
<body onload="init();">                                    ←最初に init();を実行する
        <canvas id="canvas" width="500" height="400"></canvas>  ←CANVAS のサイズ
        <p id="num">0 匹</p>                               ←最初の表示
</body>
</html>
```

●JavaScript 文

```
var canvas;                                                    ←CANVAS 要素
var img,bkimg;                                                 ←イメージ格納場所
var anim;                                                      ←anim(ｱﾆﾒ)ｵﾌﾞｼﾞｪｸﾄ
var timer;                                                     ←タイマー用
var count=0;                                                   ←熊：計数用
//初期化
function init(){                                               ←最初に実行される関数
        canvas = document.querySelector('#canvas');            ←CANVAS 要素の取得
        num = document.querySelector('#num');                  ←回数の部分の取得
        img = new Image();                                     ←Image を New する
        img.src = "character.png";                             ←その img の画像
        bkimg = new Image();                                   ←Image を New する
        bkimg.src = "background.png";                          ←その bkimg の画像
        bkimg.onload = function(){                             ←bkimg が読み込まれたら
                anim = new Anim();                             ←Anim ｵﾌﾞｼﾞｪｸﾄを new する
                canvas.onclick = makeCharacter;                ←CANVAS がｸﾘｯｸされたら
                timer = setInterval(doTimer,50);               ←doTimer()を 50 ﾐﾘ秒ごとに
        }
}
//ｸﾘｯｸしてｷｬﾗｸﾀを追加
function makeCharacter(event){                                 ←キャラ作成関数
        anim.click(event);                                     ←anim がｸﾘｯｸされたから
        count++;                                               ←熊のカウントアップ
        num.textContent = count + "匹";                        ←表示用に細工
}
//ﾀｲﾏｰで実行する処理
function doTimer(){                                            ←50 ﾐﾘ秒ごとに実行
        anim.draw();                                           ←anim を draw する
}
//ｱﾆﾒｰｼｮﾝを管理するｵﾌﾞｼﾞｪｸﾄ
function Anim(){                                               ←anim の元になるｵﾌﾞｼﾞｪｸﾄ
        this.imgs = [];                                        ←img 配列を全部カラにする
        //ｸﾘｯｸした地点に AnimImage を作成
        this.click = function(event){                          ←これ(this)がｸﾘｯｸされたら
                var x = event.clientX - canvas.offsetLeft - img.width/2;  ←座標計算
                var y = event.clientY - canvas.offsetTop - img.height/2;  ←座標計算
                this.imgs.push(new AnimImage(img,x,y));        ←img を imgs 配列に追加する
        }                    └→配列の最後に値を追加するﾒｿｯﾄﾞ
        //ｲﾒｰｼﾞの描画
        this.draw = function(){                                ←これ(this)が描画されたら
                this.drawBackground();                         ←背景を描画する
                for(var n in this.imgs){                       ←imgs 配列の中身全部
                        this.imgs[n].drawImage();              ←該当番号の img 描画関数
                }
        }
        //背景の描画
        this.drawBackground = function(){                      ←背景の描画関数
                var context = canvas.getContext('2d');         ←ｺﾝﾃｷｽﾄを取得
                context.clearRect(0,0,500,400);                ←ｴﾘｱをｸﾘｱ
                context.drawImage(bkimg,0,0,500,400);          ←bkimg をｺﾝﾃｷｽﾄに描画する
        }       └→実際の描画はここでしている
}
```

```
//ｱﾆﾒｰｼｮﾝｷｬﾗｸﾀｰのｵﾌﾞｼﾞｪｸﾄ
function AnimImage(img,x,y){                                    ←描画する関数
        this.img = img;                                         ←img をこの(this)img にする
        this.x = x;                                             ←x 座標をこの(this)x にする
        this.y = y;                                             ←y 座標をこの(this)y にする
        this.dx = Math.floor(Math.random()*10)+1;               ←横変化量を乱数で求める
        this.dy = Math.floor(Math.random()*10)+1;               ←縦変化量を乱数で求める
        //ｲﾒｰｼﾞの描画
        this.drawImage = function(){                            ←これ(this)を描画する関数
                var context = canvas.getContext('2d');          ←ｺﾝﾃｷｽﾄ取得
                this.x += this.dx;                              ←横変化量を加算
                this.y += this.dy;                              ←縦変化量を加算
                if(this.x < 0){ this.dx *= -1; }                ←画面左端に来たら逆走
                if(this.y < 0){ this.dy *= -1; }                ←画面上端に来たら逆走
                if(this.x + this.img.width > 500){ this.dx *= -1; }  ←画面右端で逆走
                if(this.y + this.img.height > 400){ this.dy *= -1; }  ←画面下端で逆走
                context.drawImage(this.img,this.x,this.y);      ←img をｺﾝﾃｷｽﾄに描画する
        }        └→実際の描画はここでしている
}
```

●処理の流れのイメージ図