



*Last class this year.
Wish a merry Christmas
& happy new year.*

Webデザイン実習4B

2019/12/17

Kazuma Sekiguchi
class@cieds.jp

課題

- スマートフォンに最適化したウェブサイトの構築
 - RWDにする必要は無い（してももちろんOK）
- レストランの紹介および予約サイト
 - 名前や場所などは適当に
 - 画像は商用利用可のもの、または自分で撮影、作成したもののみ許可
 - キャラクターは自分で作成したもの以外使用禁止
- トップページ、紹介部分、予約ページが必要
- 効果的にインタラクションを用いること

課題

- 幅414px 高さ896pxのiPhone11をターゲット
 - Chromeのモバイルモードで表示可能であること
 - 確認はChrome78のモバイルモードで確認します
- jQueryはバージョン3以上のみ使用許可
 - 動かないプラグインが多数出てくるので、注意すること
- 動画の利用はもちろん可
- 各ページにリンクを張ること
 - 1ページサイトもOK。ただし、ページ内リンクは用意すること
- 12月17日の授業時に提出

課題提出物

- HTML
- CSS（HTML埋め込みでもOK）
- JS（HTML埋め込みでもOK）
- 画像ファイル群、動画ファイル群
- デザインファイル（psd , ai , sketch , xd , figmaなどなど）、バージョンは最新版でOK
 - デザインファイルのPDFファイル

<html5 APIs Canvas>

Canvas

- Canvasタグを利用する際は、width値とheight値、id名を付与する
 - width値とheight値はそのままCanvasの大きさになる
 - width属性とheight属性を記入しないとブラウザによってはまともに表示されない
 - RWDで利用する場合は、JSで動的に画面サイズに合わせてwidth属性の値を変更するなどの対応が必要
 - id名はJSからcanvasを識別するために必要
 - canvas自体は複数設置が可能
- JSからcanvasを利用する場合は、id名をキーにしてCanvasオブジェクトを取得し、そこに対して、描画を行っていく

図形の描画

- 矩形
 - `ctx.fillRect(x, y, width, height)`
 - 塗りつぶした矩形の描画
 - `x,y`：座標位置、`width`：幅、`height`:高さ
 - `ctx.strokeRect(x, y, width, height)`
 - 枠線だけの矩形の描画
- 円
 - `ctx.arc(x, y, radius, startAngle, endAngle, anticlockwise)`
 - 円や円弧の描画
 - `x,y`：座標位置（中心）
 - `radius`：半径
 - `startAngle`：円弧を書き始める角度（ラジアン指定）
 - `endAngle`：円弧を書き終える角度（ラジアン指定）
 - `anticlockwise`：円弧を描く向きを真偽値で指定。`true`で反時計周り
 - ラジアン：角度*Math.PI/180

色の指定

- `ctx.strokeStyle = color`
 - 枠線の色
 - `color`はCSS色指定が可能（rgba指定で半透明も可能）
- `ctx.fillStyle = color`
 - 塗りつぶしの色
- グラデーションの指定の仕方は異なる

画像の読み込み

- 任意の画像ファイルを読み込み可能
 - ブラウザーが扱うことができるファイルであることは最低条件
 - 実のところ映像も扱う事が可能
 - iOSで動画再生がインラインでできなかったときに代用された
- `ctx.drawImage(“ファイル名” ,貼り付ける場所 (X座標) ,貼り付ける場所 (Y座標) ,貼り付けるサイズ (幅) 、貼り付けるサイズ (高さ));`
- 縮小したり拡大して貼り付けることが可能
 - 貼り付けた後は、普通のピクセルデータの扱いになる

Canvasでアニメーション

- 実のところ、結構大変なので、あまりオススメしない
 - CSSアニメーションの方が速度が出ることもあり、それほど利用されている場面も少ない
- Canvas自体にはアニメーション機能は備わっていない
 - JSでCanvasにオブジェクトを描画する
 - JSでCanvasに描画したオブジェクトを消去する
 - JSでCanvasにオブジェクトを少し動かして描画する
の繰り返し
 - setIntervalやrequestAnimationFrameを多用する

setInterval, setTimeout, requestAnimationFrame

- setIntervalは指定した時間が経過することに関数を呼び出す機能
 - setTimeoutは指定した時間後に関数を呼び出す機能
 - setTimeoutで呼び出した関数内で更にsetTimeoutを呼び出せば、永久ループが可能
- requestAnimationFrameはアニメーションが可能になったら実行される関数
 - どのような間隔で実行されるかはブラウザーや環境などに依存する
 - (ある程度環境が整っていれば) スムーズな動きが実現可能
 - Canvasアニメーションでは最近はこれを利用する率が高い (ブラウザーがすべて対応していない点注意)

requestAnimationFrame

- requestAnimationFrameはアニメーションが可能になったら実行される関数
- アニメーションを終了させるときは、cancelAnimationFrame()を利用する
 - 予めrequestAnimationFrameを実行したときに値を変数に格納しておく必要がある→終了時に使用する

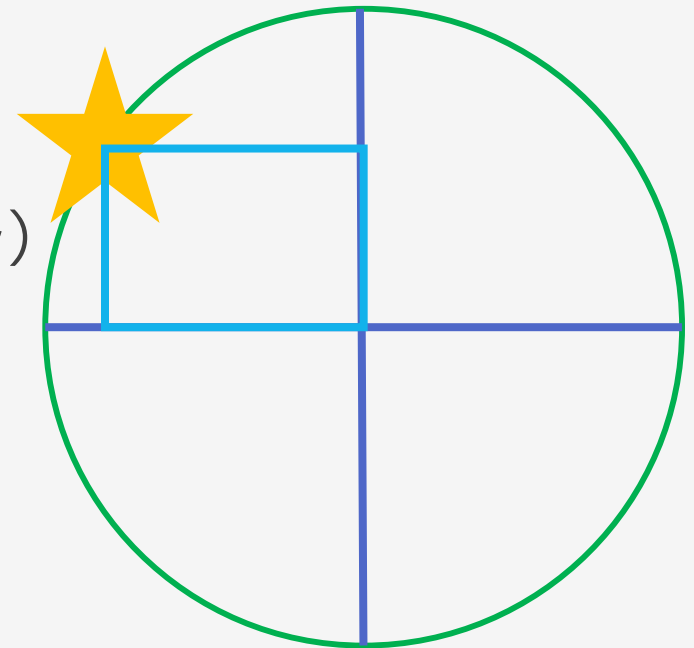
```
var req = requestAnimationFrame(anime);  
function anime(){  
  //アニメーション内容  
}  
cancelAnimationFrame(req); //アニメーションの終了
```

アニメーション

- 描画自体は比較的簡単
 - 上手くランダム関数などを利用すると毎回違った画像になるので、変化を与えることが可能
- アニメーション
 - 一度描画したものを消去して新しく描画し直す
 - これを高速に繰り返せば、アニメーションしているようになる
 - 実際のところ、これはものすごく大変
 - 同じ図形を動かすだけならそれほどでもないが、ループで複数作成した場合は難しい

直線運動は簡単

- 直線的に動くのであれば、最終地点のX座標、Y座標が分かれば、途中の座標は算出可能
- 問題は曲線、円運動
 - Sin、Cosを利用する
 - SinとCosで座標を算出することが可能
 - sin、cosを使うことで円運動は表現可能
 - X座標：中心のX座標 + 半径 * $\text{Math.cos}(\text{ラジアン})$
 - Y座標：中心のY座標 + 半径 * $\text{Math.sin}(\text{ラジアン})$
 - X座標とY座標が決まれば配置が可能



波表現

- `Math.sin`(ラジアン) はどんな数を入れても-1~1までの値を返す
 - それをY座標に入れば、波みたいな形を実現することが可能
 - sin波とも呼ぶ
- sin波を利用することで、波みたいな表現が可能
 - 複数を組み合わせると結構キレイ
 - 実際には細かい点を打ちながら直線を繋げて表現する

複数の要素を動かす

- 複数の要素を動かす場面が多い
 - パーティクルを利用する場合など
 - すべての要素を管理する配列またはオブジェクトを利用する
 - オブジェクトなどに現在値、移動値などを格納しておいて、取り出して利用する（場合によっては書き戻す）
 - オブジェクトに格納しておくことで、ループ処理が可能
 - 番号で指し示すことが可能であるため、それぞれの要素に名前を付けたのと同じ状態になる

オブジェクト (復習)

```
var obj = {};  
obj.x = 200;  
obj.y = 400;  
obj.name = "object Element";  
var arr = [];//配列を作成  
arr.push = obj;//配列内にオ  
ブジェクトを格納  
arr[0].x;//→200が取得できる
```

```
var arr[];  
arr.push({  
  x = 200;  
  y = 400;  
  name = "object Element";  
});  
arr[0].x;//→200が取得できる
```

まとめて記述すると右のようにも記述可能

pushを上手く利用

- pushメソッドを使うと配列に要素を追加可能
 - pushメソッドをループさせれば、必要な数だけ配列内に要素を追加することができる
 - インデックス（配列に付けられる番号）を指定しても格納可能だが、pushの方が手軽なことが多い

移動

- 移動とはベクトルである
 - ベクトル=XとY座標における移動量（厳密にはだいぶ違う）
 - どのときに使うか→移動のときに利用する
 - 計算：元のX座標とY座標と移動後のX座標とY座標の差
 - XとY座標の差が移動量となる（これを何秒掛けて移動するか）
 - 通常この値をオブジェクトに持たせておくことで、ループ処理し移動させる

