

# Webデザイン実習4B

2020/01/14

Kazuma Sekiguchi

class@cieds.jp

<html5 APIs Canvas>

# Canvasでのイベント

- JSで使用できるほとんどのイベントが使用可能
  - click , double click , mouseover/mouseout , mousedown/mouseup,mousemoveなど
  - イベントを利用する場合は、 canvasタグにイベントを紐付ける

```
window.onload = function(){
  canvas = document.getElementById("stage");
  ctx = canvas.getContext("2d");
  canvas.addEventListener("click",draw, false);
}
function draw(){
  ctx.fillStyle = "#F90";
  ctx.fillRect(100,100,20,20);
}
```

## クリックした場所の取得

- クリックした座標の取得は`e.clientX`,`e.clientY`で取得可能
- `getBoundingClientRect()`でCanvasの位置を取得可能
- Canvasの位置を考慮しないため、クリック座標である`e.clientX`,`e.clientY`から`getBoundingClientRect()`でその分を引く

```
function onClick(e){  
  var rect = e.target.getBoundingClientRect();  
  x = e.clientX - rect.left;  
  y = e.clientY - rect.top;  
}
```

# ローカルファイルへのアクセス (FileAPI)

- これまでブラウザーからローカルPC上にあるファイルにアクセスすることはできなかった
  - セキュリティ上の問題
- ローカルファイルをそのまま扱いたい場面も多々
  - ローカルPC上に保存している画像ファイルをプレビューしてからアップロードする
    - これまではサーバーにアップロードして初めて画像を確認出来た
    - サーバーにアップしなくても画像を確認可能

# DataURL

- 画像などのバイナリデータをテキストデータに変換したものの
  - Base64形式に変換することで、テキストデータに変換する
  - <img>などのsrc属性にそのテキストデータを指定することで、そのまま表示することが可能

# DataURLで取得したデータを表示

- canvasに取得したデータを表示可能
- FileAPIを使うことで、プレビュー画像を表示させることが可能
- canvasに書き出すことで効果を与えたりすることが可能
  - WebStorageに保存も可能
    - ブラウザー上に保存することができる仕組み
    - 保存できるのはプロパティ：値の形式でテキストのみ
    - シリアライズ（配列を単一行のテキストにする方法）することで結構な情報を入れることも可能

# JSでの画像の扱い

- JSでは画像の読み込みなどは非同期で行われる
  - 画像の読み込みには時間を要する
  - 画像の読み込みが終わっていないのにも関わらずその画像に対して処理を行うとすると確実にエラーが発生する
  - 画像の読み込み、処理をJSだけで実行する際は要注意
    - また画像を遅延読み込みするときも同様の問題が発生する
- JSで画像が読み込まれたことを確認
  - 予めnew Image();でイメージオブジェクトを作成しておく
  - onloadイベントで読み込まれたとき、というイベントになる

```
var img = new Image();  
img.src = "sample.jpg";  
img.onload = function(){  
//ここに画像が読み込まれたときの処理を記述  
}
```

# 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つ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

# 画像のトリミング

- 画像を読み込んでおけばOK
- `drawImage( "ファイル名" ,X0,Y0,W0,H0,X1,Y1,W1,H1);`
  - X0,Y0にはトリミング開始位置を指定
  - W0,H0にはトリミングサイズを指定
  - X1,Y1はトリミングした画像の描画位置
  - W1,H1はトリミングした画像の描画サイズ

# Canvas画像への画像処理

- Canvasではすべてのピクセルを制御することが可能
  - 任意のピクセルの色を変更することができる
  - ピクセルの色をシフトさせることで、色を変更することが可能
  - がんばればレタッチなども可能・・・、なはず
- Canvasでは全てのピクセルが4つの色情報を持つ
  - RGBaの4つ
  - `getImageData()`で全てのピクセル情報を取得し、処理を行って`putImageData()`で書き戻すことで色を変えたり画像処理をおこなうことが可能

# Canvasへの画像

- Canvasへ画像を読み込む際には、ローカル環境であるとCrossOriginセキュリティが働くため正常に動作しない
  - Webサーバー（SSL必須）か、ローカルサーバーを立ち上げて確認すること

# Canvasでペイントソフト的なもの

- Canvas上でマウスをクリックした時に描画すれば、ペイントソフトみたいなものを作成可能
  - ドットとしては描画できないので、線として描画する

```
ctx.beginPath();  
ctx.moveTo(mouseX,mouseY);//描き始めの位置  
ctx.lineTo(x,y);//描き終わりの位置まで線を引く  
ctx.lineWidth = 4;  
ctx.strokeStyle = "#F90";  
ctx.stroke();//実際の線の描画  
mouseX = x;  
mouseY = y;
```