# Webデザイン実習2C

2017/11/01

Kazuma Sekiguchi
class@cieds.jp

# Fluent Design System

- https://fluent.microsoft.com/
- MSがWindows10 Fall Creators Updateから導入したデザイン概念
  - マテリアルデザインと似ているが、微妙に異なる
  - Windowsに搭載されたことから今後広がっていく可能性がある
- 奥行きを重視していく方向が見える

# 2018年のトレンド（？）

- 本当に流行るかは知りませんが、いくつか出始めてます
  - レインボーカラー
    - 比較的派手なグラデーションなども
  - 境界線をはみ出したテキスト
    - 写真画像からわざとテキストをはみ出させる
    - テキストに背景色を引いて、画像に載せる
  - ホローシェイプ
    - 枠線だけで構成されているシェイプ
    - 幾何学的な模様みたいなものを表現

# 2018年のトレンド（？）

- 本当に流行るかは知りませんが、いくつか出始めてます
  - スプリットスクリーン
    - 画面全体を2枚以上の画像で分割し表示
    - アンシンメトリーなものが流行るかも
  - マスク表現
    - CSSでマスクが表現が可能
    - 他の画像にマスクで形を作り、表現するなど
    - マスクの中を動かすなどの表現が考えられる

# 動きを付ける

- いくつかの方法を組み合わせる
  - CSS Transition
  - CSS Animation
  - SVG Animation
  - JavaScriptで動かす
- 一番良く利用されるのがCSS Animation
  - CSSがどうしてもサイズが増えるため、アニメーションだけ別ファイルにするなどの工夫は必要
- SVG Animationは今後普及していくはず

# CSS3 Animation

# CSS3を利用した動き

- CSS3からtransformプロパティが利用可能
  - 移動
  - 拡大、縮小
  - 回転
  - 傾斜変形

# CSS3　移動

- transform:translate(X方向の距離、Y方向の距離)
- transform:translateX()
  - X方向の距離
- transform:translateY()
  - Y方向の距離
- transform:translateZ()
  - Z方向の距離
- transform:translate3d(X方向の距離、Y方向の距離、Z方向の距離)

# CSS3　拡大・縮小

- transform:scale（X方向の比率、Y方向の比率）
- transform:scaleX（X方向の比率）
- transform:scaleY（Y方向の比率）
- transform:scaleZ（Z方向の比率）
- transform:scale3d（X方向の比率、Y方向の比率、Z方向の比率）
  - Z方向の比率は機能しない

# CSS3　回転

- transform:rotate(回転角度)
- transform:rotateX(回転角度)
  - X軸を軸とする角度によって時計回りの回転
- transform:rotateY(回転角度)
  - Y軸を軸とする角度によって時計回りの回転
- transform:rotateZ(回転角度)
  - Z軸を軸とする角度によって時計回りの回転
- transform:rotate3d(方向ベクトル、方向ベクトル、方向ベクトル、回転角度)

- 角度の単位はdeg
  - 10deg＝10度

# CSS3　傾斜変形

- transform:skewX（X軸の傾斜角度）
- transform:skewY（Y軸の傾斜角度）
- transform:skew（X軸の傾斜角度，Y軸の傾斜角度）

- 角度を指定するので、degで指定する

# CSS3　アニメーション

- transitionで指定するとアニメーション可能

```
.prefix_sample {
background-color:blue;
width:200px;
height:50px;
transition: background-color 1s, width 1s, height 1s;
}
.prefix_sample:hover {
background-color:aqua; width:400px; height:100px;
}
```

- :hoverなどの疑似イベントなどを利用してアニメーションさせる

# CSS3　animation

- @keyframes 名前でアニメーションを定義する
- CSSのanimationプロパティで@keyframesで指定したアニメーションを指定すると動作する
- infiniteを指定すると永久ループする

```
.sample {
animation: anime1 5s ease infinite alternate;
}

@keyframes anime1 {
0% {width: 50px; height: 50px; background-color: #F90;}
100% {width: 200px; height: 50px; background-color: #09F;}
}
```

# SVG

- ScalableVectorGraphicsの略
  - 画像形式の1つと考えて良い
  - 内部はXML形式で表現されるため、テキストエディタで変更が可能
  - 以前はサポートされていなかったが、現在はほとんどのブラウザーでサポートされている
    - IEは除く（もう、どうでも良い）
- パスをそのまま書いたものになっているため、拡大縮小してもジャギーが出ない
  - 解像度に依存しない

# SVG

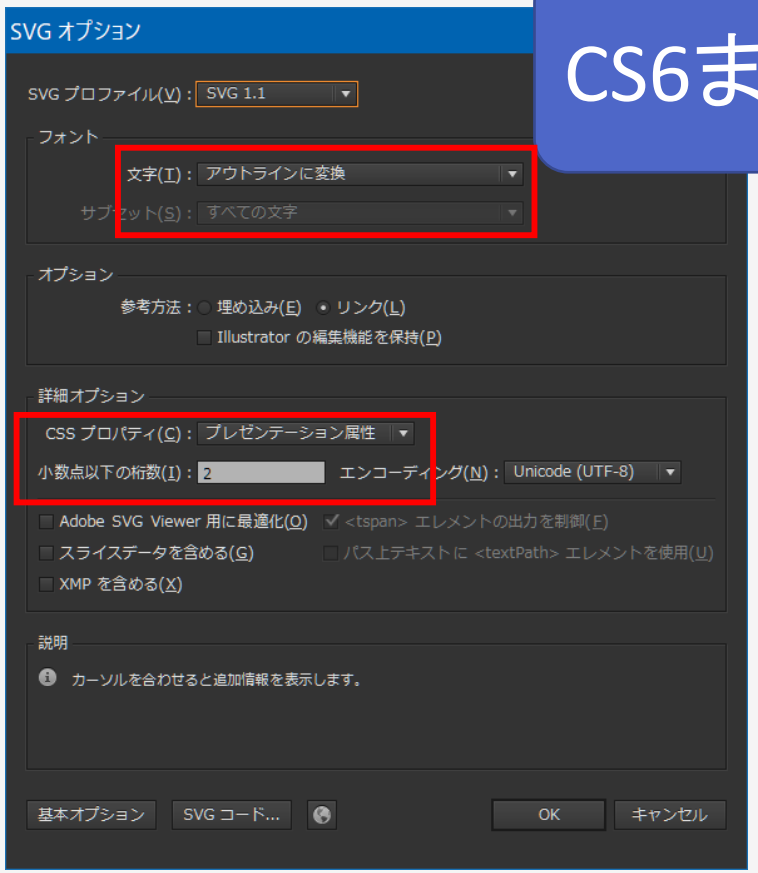
- ビットマップ画像とは異なる
  - ファイルサイズがビットマップ画像よりも軽い
  - 基本的にはイラスト、ロゴなどに利用するのがベスト
- JPEG画像を含めることが可能
  - 可能ではあるが、画像を呼び出しているだけなので、含めない方がベスト
  - JPEG画像などと組み合わせたい場合は、HTML上で組み合わせた方が扱いやすい
- アニメーションができる

# SVG作成

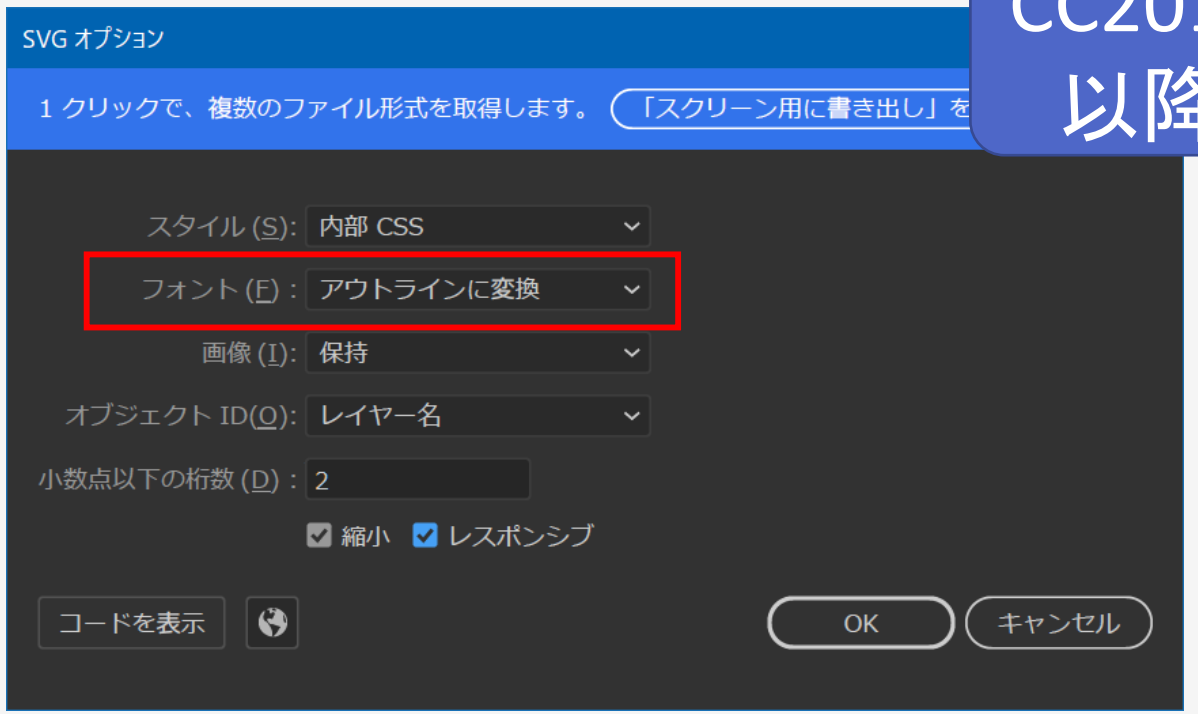
- Illustratorなどで作成可能
  - パスで描くものなので、ベクターツールであれば、大体対応している（が、結構おかしなところもあるので、Aiが一番良さげ）
  - テキストはそのままデバイスフォントを使うように指定されるので、アウトライン化した方が良い
    - アウトライン化しないと、フォントが変わる可能性大
  - 「ファイル」→「書き出し」で書き出せる（CC2015以降。それ以前は「別名で保存」だったり「Web用に保存」だったり）
  - 書き出す際は、SVG1.1形式を選択。圧縮はしないほうが良い
- レイヤー名、パス名などは英語化しておく
  - 書き出した際にそのまま使われるが、後からSVGを利用する際に変更するはめになるため

# SVG作成

CS6まで




CC2015以降




- 吐き出されるSVGのコードも違う
  - CCの方がかなり品質が良い

# SVGの利用

- 画像形式ではあるが、XML形式なので、埋め込むタグがimgタグではない
  - imgタグでも埋め込めるようだが、imgタグで埋め込むとJSなどから操作ができなくなるため、SVGとしての意味がイマイチ出てこない
  - SVGのインライン形式で埋め込むのがベスト
  - 行が長くなるが仕方ない
  - SVG内に出てくる<g>タグは削除可能
    - 但し、属性が付与されているものは削除しないこと
    - Aiの作りによっては、gタグは出てこない

# SVGアニメーション

- SVG自体は、XMLであるため、HTMLとの相性が良い
  - SVGの内部でアニメーションするための仕組みが備わっているほか、CSSも使ってスタイルを変えることが可能
  - CSSアニメーションも利用可能
- JSのイベントを利用してアニメーションを変化刺せることも可能
- アニメーションは利用可能だが、結構作成が面倒なところもある
  - 直感的ではない

# SVGアニメーションを作成できるツール

- Snap.svg拡張機能をAnimate CCに追加してSVGを書き出す
  - snap.jsを併用利用することで、Animate CCで作成したSVGアニメーションを利用可能
  - まだ完全互換性を有していないため、微妙なところがある
  - Flashでは導入できない
- svgator
  - https://app.svgator.com/
  - オンラインのFlashみたいなSVGアニメーション作成支援ツール
  - 比較的簡単に利用が可能

# SVGアニメーションをタグでがんばる

- SVG内にコードを書くことでアニメーションさせることが可能
  - 但し、非常にコードが面倒・・・
- SVGでは図形はタグで記述される
  - <circle>,<text>,<path>,<rect>などのタグが存在
  - タグの開始タグと終了タグの間にアニメーションタグを記述する
  - タグは大体省略表記になっているため、省略表記を展開する必要がある
- アニメーションするためのタグは3種類
  - <animate>,<animateTransform>,<animateMotion>
  - animateが基本的なアニメーション

# animateタグの利用

- <animate>の属性に値を指定してアニメーションさせる
  - attributeType:普通はXMLを指定
  - attributeName:アニメーションさせる属性
  - dur:アニメーションの秒数。sとかmsの単位を付与して指定
  - repeatCount：アニメーションの繰り返し回数。indefinite指定で永久ループ
  - from：初期値
  - to:終了値の値
  - value:from/toを指定せずにkeyframeを打っていくときに使用。セミコロンで区切って利用
  - begin:開始タイミング（sとかmsとか指定可能）

# animateTransformの利用

- <animateTransform>の属性に値を指定してアニメーションさせる
- 主にCSS3のtransformに該当する指定が可能
  - attributeNameがtransformのみ指定可能
  - typeにscale ， rotate ， translate ， skewなどが指定可能
  - 値はvaluesで指定することが可能
  - 他の属性はanimateタグに指定できるものと同じ
  - オブジェクトの中心にして動作する

# animateMotionの利用

- <animateMotion>の属性に値を指定してアニメーションさせる
- animateMotionタグはモーションパスに従って動かす場合に利用する
  - 予めパスを描画しておき、id属性を付与しておく
    - このパスは非表示にしておく
  - <mpath xlink:href=“#id名” />でパスを指定する
  - animateMotionで動かすオブジェクトは中心座標を0,0にして保存しておく

# SVG内のCSSで指定できる主なもの

- SVG内でもCSSが使われている
  - 普通のCSSの書き方と同じだが、使えるプロパティが違う

    - fillで塗りつぶしの色を指定
    - strokeで線の色を指定可能
    - stroke-widthで線の幅を指定可能
    - stroke-opacityで線の透明度
    - stroke-dasharrayで破線などのスタイルを指定。間隔も指定可能
    - cx,cyは座標、rは円の半径、rxは横方向の円の半径、ryは縦方向の円の半径

# SVG内のCSSで指定できる主なもの

- SVG内で指定できるCSSはSVGの外でも指定可能
  - classやidがSVG内部では使われているため、それらを利用することでSVGの見た目をSVGの外部CSSから変更可能
  - @keyframesによるアニメーションも利用可能

- SVGでは内部へのanimateタグなどによるアニメーション記述と、外部の@keyframesによるアニメーションが利用可能
  - 外部が利用できるため、JSで動的にスタイルを変えることも可能