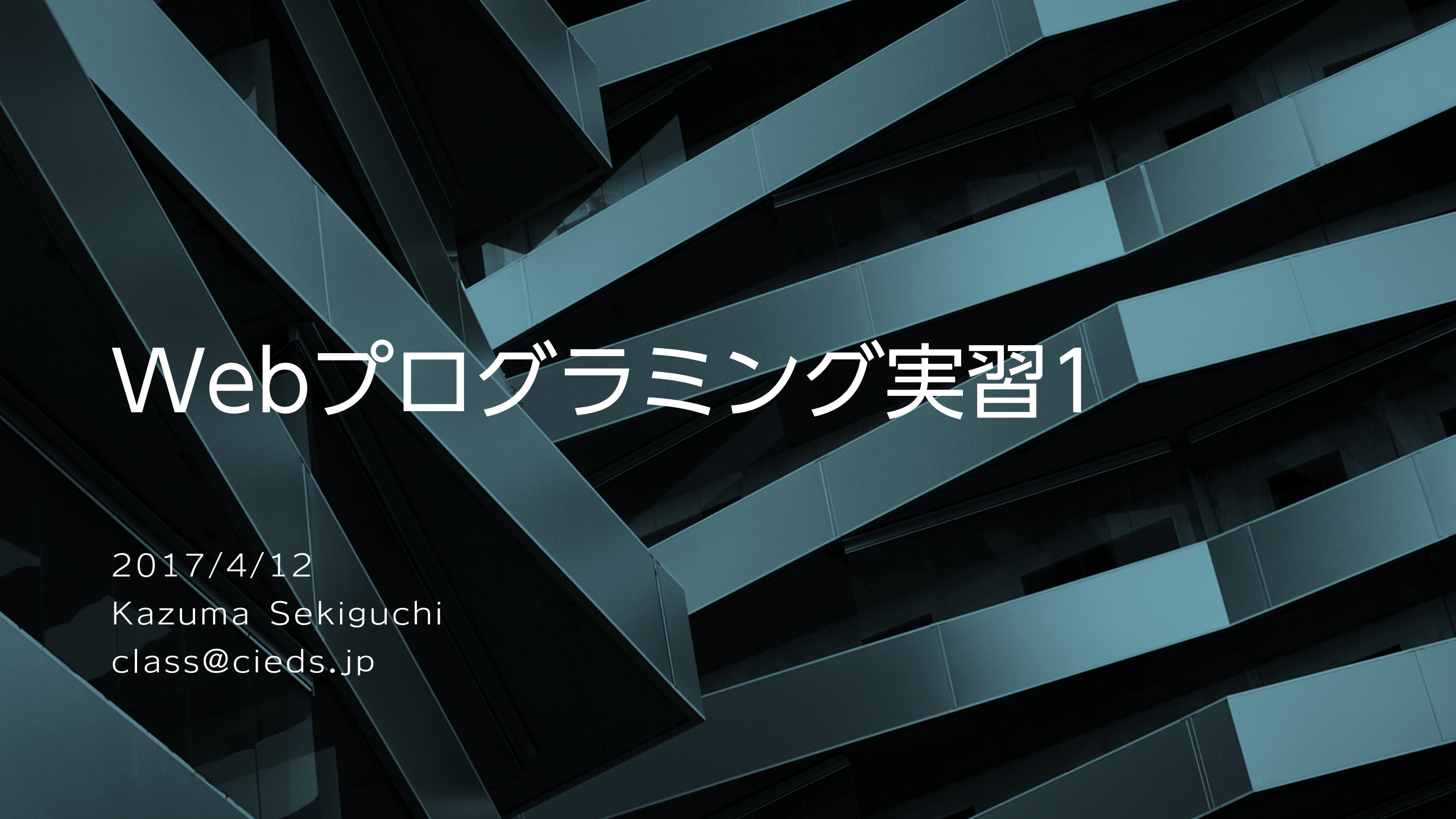


Webプログラミング実習1

2017/4/12

Kazuma Sekiguchi
class@cieds.jp

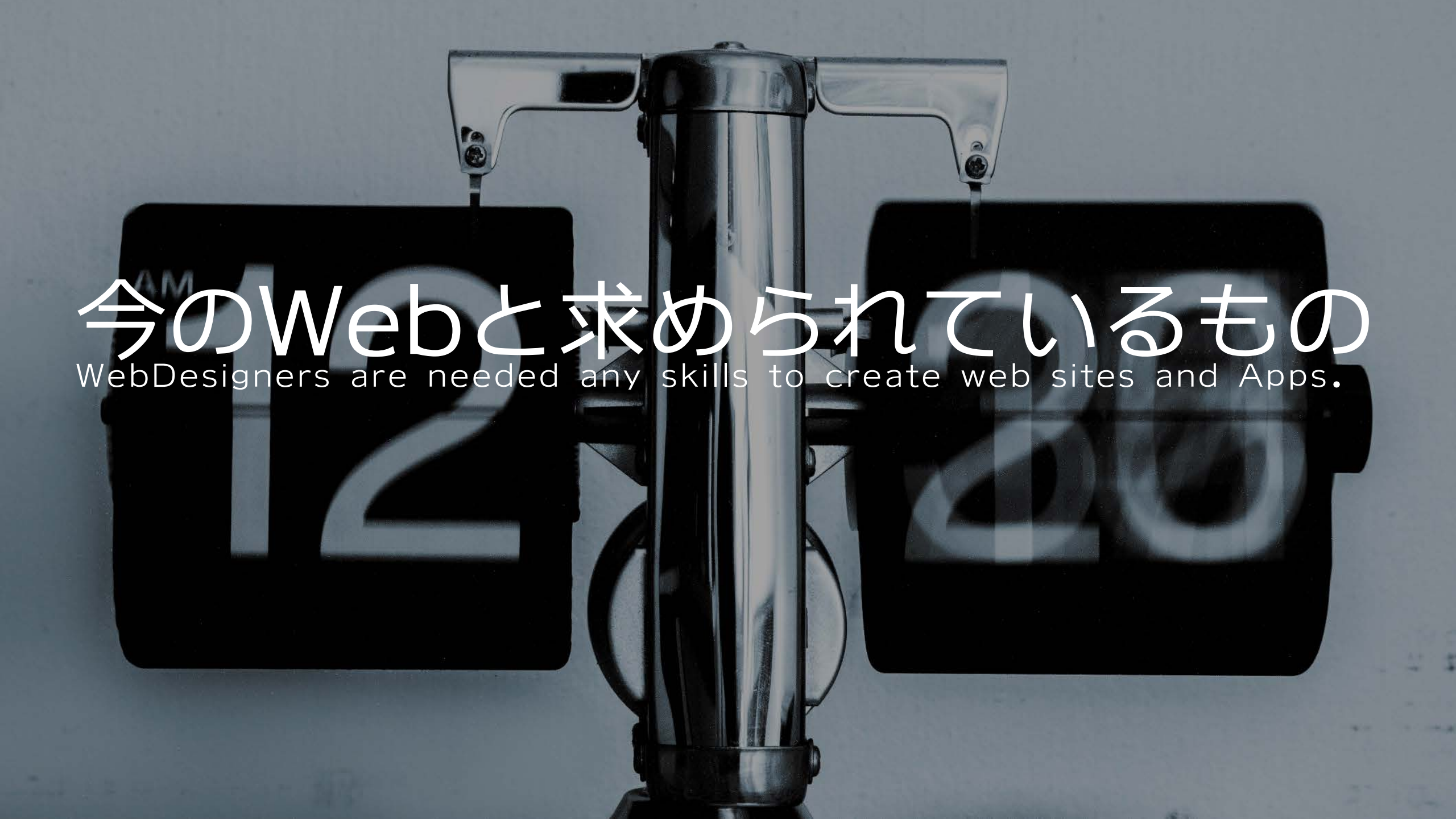
担当

- 関口　和真（Kazuma Sekiguchi）
  - Comcent,Inc
  - Webプログラマ兼コーダーとかいろいろ
  - デザインもごく希にやります
  - たまにWebに記事を書いていたり本を書いていたりします
  - Adobeの多少なりとも関係者
  - 絵は全く描けないので聞かないように

- 質問事項は、メールに所属、学年、氏名等を記載の上、class@cieds.jpに送付

Agenda

- JSの基本知識
- インタラクション性のあるWebサイトの作成
- スマートフォンでのインタラクティブ性
- Canvasを利用した表現技法



今のWebと求められているもの

WebDesigners are needed any skills to create web sites and Apps.

今のWeb

- 見る物から使う物への変化
  - 広告媒体の一種
  - 使ってもらう
- SNSでのシェア
- いつでもどこでも情報を手に入れられる
  - スマートフォンなどのデバイス
  - 複数のサービスを組み合わせて利用
  - スマホアプリ

重要性の変化

- 見せる物から使えるものへ
  - マウスクリックさえできればOK
  - 指でタップできる必要
  - 連携できる必要
- スマートフォンアプリとして展開
  - 他のサービスと連携して利用する
- オフラインでも使える、見られる
  - いつでも同じ状況でネットが使えるとは限らない

使い方の変化

- キーボードはより使いづらい
  - 入力を極力減らす工夫
  - ジェスチャーでの操作が可能なように
  - 入力させる場面を減らす
- 写真の方が楽
  - Instagram、Facebookの発展
- キーボード、マウスという当たり前デバイスからの思考脱却
- 大量のデバイスの登場
  - 性能がより向上

重要度の変化

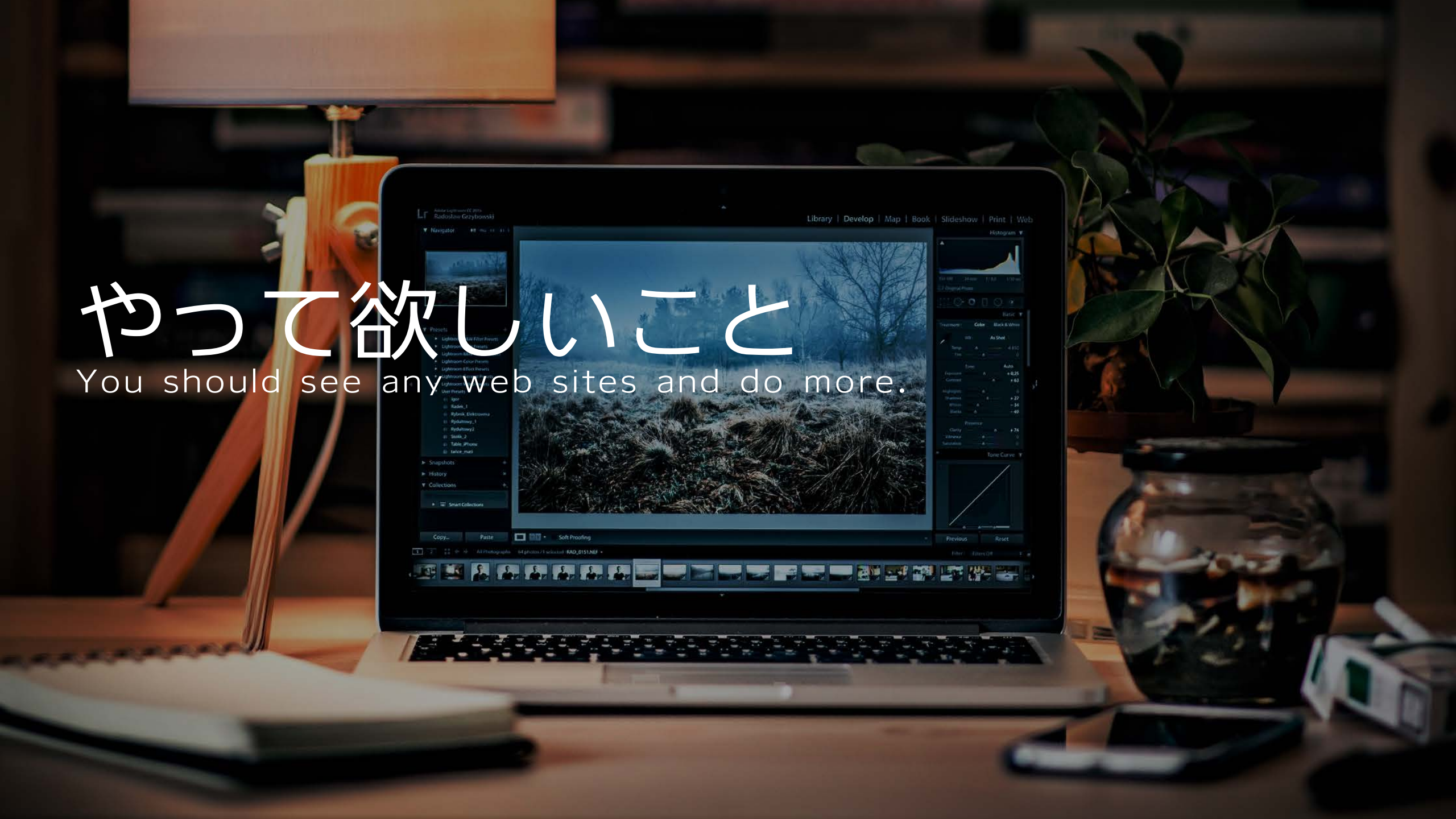
- サーバーサイドでデータを生成してクライアントに渡す
  - Googleでの検索など
- 検索や掲示板などでは未だに重要な仕組み
- データを更新、読み込むときにはリロードが掛かる
  - 思考が連続的に行えない欠点
- GoogleMapなどはデータが追加されてロードされる
  - 操作性の向上
  - ほかのWebサイトでも当たり前のように実現
  - 通常のアプリと同じ感覚でウェブを使えるように

ノンリロードコンテンツ

- リロード無しにWebを使うための仕組み
  - JSを最大限に利用し、クライアント側でデータを保持、更新、取得する方法で実現
  - SPA＝SinglePageApplication（URLが変わらないままさまざまな機能を利用することが可能）
    - Ex:TweetDeck、YouTubeなどなど
  - 最近のWebアプリ（使うWebサイト）では当たり前
- フロントエンドの重要性
  - 使い勝手、マルチデバイス対応、状況による変化への対応

使う方へのシフト

- Webデザイナー、コーダーは変化に耐える知識と技能が必要
  - デバイスの大量出現
    - スマートフォンだけで300種類以上
    - スマートデバイスも登場
  - 使い勝手の良いUI/UXの必要性
    - 使い勝手が利用率に直結
  - 効率的な開発、集団開発時にどのようにするのが良いか
    - 現状のCSSとJSの限界
    - PostCSS、JSNext


やって欲しいこと
You should see any web sites and do more.

やって欲しいこと（1）

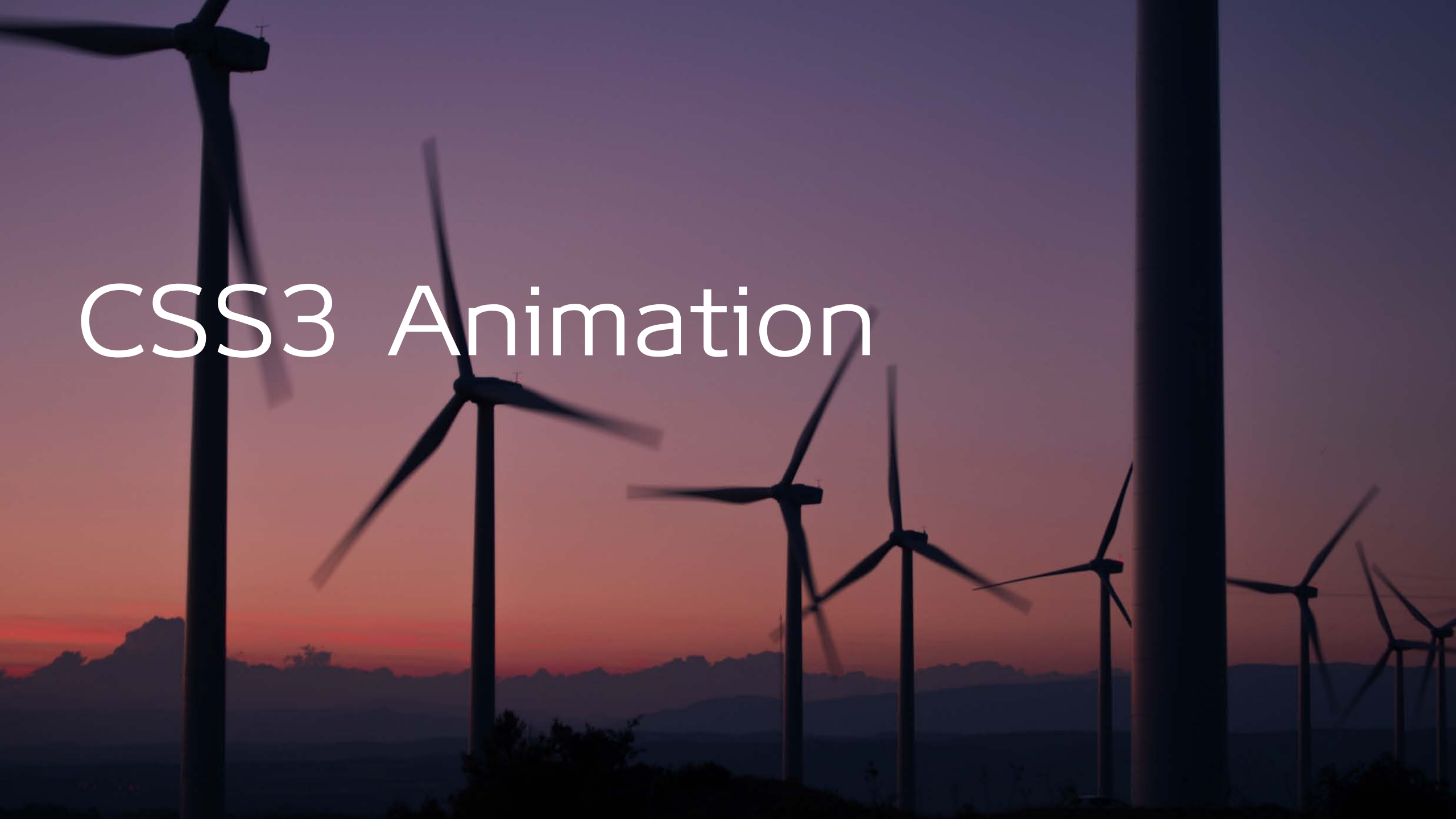
- 様々なウェブサイトを見る
  - 趣味のサイトは意識せずに見るはずなので、趣味以外のサイトを見る
  - インタラクション、インターフェイスを良く観察する
  - Dribble、Behance辺りはオススメ
- スマートデバイスを意識したUI/UXの作成
  - PC向けサイトの重要性は低い
  - スマートフォン向けにどういうメリットを出せるかを考える

やって欲しいこと（2）

- Webは進化が早い
  - 技術的なことは直ぐに陳腐化していく
    - 貪欲に調べる、やってみる
  - 基本原理は継続しているため、基本原理をきちんと把握する
    - 何かを解消するために新しい技術が出てくる
    - 何を解消しているのかを把握する
  - 技術が伴うことで表現できるものも多い
    - 最近のWeb表現はWeb技術と上手く融合している例が多い

やって欲しいこと（3）

- 勉強会などに出席してみる
  - 大体無料であちこちにて開催されている
  - 話が分からなくても刺激になることも多い
  - ヒントが得られることも
  - 何となく知っているようなものへの出席がオススメ
- 好奇心
  - ちょっとでも知っていると全く知らないでは大違い
  - 意外な知識が意外なところで役立つことも


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

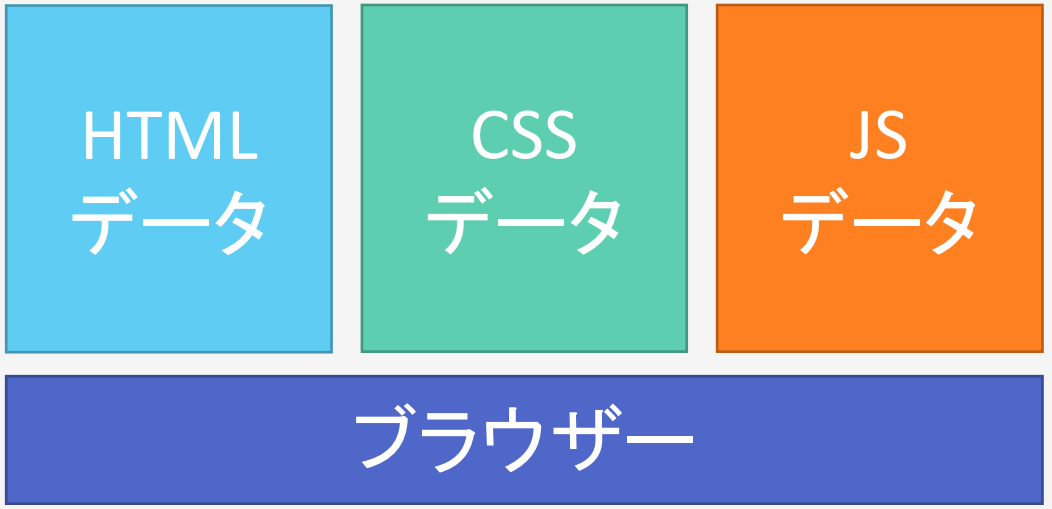
CSS3　animation

- @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
- 現時点ではブラウザー接頭辞が（一応）必要

```
.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;}
}
```

ブラウザーの仕事



- ブラウザーが表示するのはHTMLデータなど
  - HTMLはファイルじゃ無くてデータでも良い
- サーバーからデータを送ってきてそれを展開するのが本来のブラウザーの仕事

クライアント側での処理

- ブラウザーはHTMLとCSSを表示
  - ユーザーの操作に合わせて表示を変える、HTML、CSSだけでは表現できない、利用できないものがどうしても発生してきた
  - CSSを動的に書き換えることで、アニメーションを実現する
  - HTMLを動的に書き換えることで、マウスオーバー時に画像を変える
  - ユーザーの動きにあわせてCSSを書き換えて、インタラクションを行う
- これらはCSSだけでは不可能なことが多い

プログラミング言語

- ブラウザー側で利用できるプログラミング言語はJavaScript（JS）
  - 間違えてもJavaとは略さないこと（別な言語になる）
- JavaScriptを利用することで、ブラウザー上でさまざまなことが実現することが可能
  - HTMLの書き換え、CSSの書き換え、ブラウザー上でユーザーが実施した操作の取得などなど
  - 最近ではJSを利用して、デスクトップアプリやスマホアプリを作成することも可能
  - 極めて汎用性が高いが、習得が少し難しい言語

プログラミング言語とは

- 何らかのデータを与えると処理をして、結果を返す
  - この集合体（だけじゃないけど）
- 例：自販機
    - お金を投入→金額と偽物じゃ無いか確認→問題ない
    - 商品選択→売り切れじゃ無いか確認→問題ない
    - 商品落としまーす
- いくつかの処理を組み合わせて目的を達成する
  - 重要なのは、データという概念、処理という概念

プログラミング言語とは

- JavaScript
  - プロトタイプ指向のオブジェクト指向言語（覚えなくて良い）
  - ブラウザー上で動作して、HTMLやCSSを変化させて表示するための処理を行うことに特化したプログラミング言語
  - 書き方が結構独特（他の言語を知らなければ気にならない）
  - 複雑なことをしようとすると、書き方が途端に煩雑になる
  - 参考、サンプルが非常に豊富。探せば直ぐに求めているものが見つかる可能性が高い
  - 結果が直ぐ分かる

プログラミング言語とは

- 1文字でも間違えると動作しなくなる
  - HTMLやCSSは間違えても表示はされる
  - JSは1文字でも間違えると動作しない、最悪ブラウザーが真っ白になる
    - どのプログラミング言語も同じではあるが、入力時注意
    - 打ち間違いはどうしても出るが、できるだけ減らす工夫をする
    - フォントを変更したり、文字を大きくしたり、誤字と思われるところにアンダーラインが引かれるようなエディタを利用してみる
- 1度では大体動作しない
  - 慣れても何度書いても間違いは無くならない
  - エラーがあったときに自分で回復する能力を

JavaScriptを書いてみる

- <script></script>の間に記述する
  - 別ファイルに記述してもOK
  - 別ファイルの場合は、拡張子をjsとして<script src=“〇〇〇.js”></script>として読み込ませる
- 実行されるタイミングは基本的にブラウザーでHTMLが表示された後

プログラミング言語

- JSでは変数、メソッド、イベントなどが用意されている
  - 変数＝一時的に保存しておくための場所。名前を付けることでいろいろな内容を保持可能
  - メソッド＝JavaScriptで利用することができる機能のこと。1つ1つでは大したことはできない
    - 組み合わせることで処理をしていくことが可能
  - イベント＝ユーザーのインタラクション
    - 後日説明

JSの基本的なもの

- 変数
  - 何かの値やオブジェクトを格納しておくもの
  - JSの場合、いろいろなものを格納可能
    - オブジェクトも格納可能
  - 先頭にvarを付ける（変数宣言）
    - スペースを開けて変数名を指定（通常1文字目は英小文字）
  - 数値と文字が違うので注意（” ” で括れば文字）

```
var a = “JavaScript”;//aに”JavaScript”という文字列を格納
var b = 123;//bに123という数値を格納
```

変数

- 定義関数を格納できるところはJS独特
- 通常は変数宣言を行って利用する（宣言しなくてもOKだが、最近は警告が出ることが多い）
  - 先頭にvarを付ける（変数宣言）

```
var a;//aという変数を利用する（宣言のみ）
var b = 1;//bという変数を利用すると宣言し、1という数値を格納
var c = “js”//cという変数を宣言し、jsという文字列を格納

a = 4;//宣言済みの変数aに4を格納
```

変数宣言

- プログラムの一番最初にまとめて記述することが多い
  - ほかの人が見たときにどういう変数を利用しているか把握しやすい
  - 変数に設定値（リンク先のURLや何らかのパラメータ）などを格納した場合、変更しやすい

JSの基本的なもの

- 何らかの動作をまとめたもの＝関数
- 引数は複数の値を指定可能。返り値は1つだけ

```
function 関数名（引数）{
何らかの処理処理
return 返り値
}
```

```
function keisan（a,b）{
        var c = a + b;
        return c;
}
```

```
var kotae = keisan(2,3);
```

kotaeには5が入ってくる

ユーザ定義関数

- 何らかの処理をまとめたもの
- 同じような処理を繰り返す際に利用する
  - イチイチ同じ処理を何度も書くのは面倒。間違えも増える
- ()内に引数を指定可能
  - 引数はカンマで区切ることで複数指定可能
  - 引数に値を与えて関数を呼び出せば、その引数を利用して処理してくれる
- ユーザ定義関数から更にほかのユーザ定義関数を呼び出すことも可能
- 記述場所は自由
  - 通常まとめて書いておく
  - 関数だけのJSファイルを作ることも多い

ユーザ定義関数

```
function 関数名(引数1,引数2・・・){
    処理の内容
    return 処理の結果
}
```

基本的な記述方法

```
function Calc(a,b){
    var c = a + b;
    return c;
}
```

作成（Calcという関数を作成）

```
var g = Calc(4,6);//gには10が格納される
function Calc(a,b){
    var c = a + b;
    return c;
}
```

Calcを利用

```
function caution(){
    alert(“注意！ ！”);
}
```

returnの無いものも可能

変数宣言の場所による違い

- ユーザ定義関数内で宣言した変数
  - その関数内でのみ利用可能＝ローカル変数
- 関数外で宣言した変数
  - ユーザ定義関数内でも利用可能＝グローバル変数
- グローバル変数の場合、意図しないところで上書きすることがあるため、できるだけ避ける
  - 無理なことも多い

```
var a = 1;
function ShowAlert(){
    alert(a);//1が表示される
}
```

```
function SetA(){
    var a = 1;
}
function ShowAlert(){
    alert(a);//何も表示されない
}
```

変数宣言の場所による違い

```
var a = 1;
function ShowAlert(){
    alert(a);//1が表示される
}
```

グローバル変数としてaを利用

```
function SetA(){
    var a = 1;
}
function ShowAlert(){
    alert(a);//何も表示されない
}
```

aはローカル変数のため、ShowAlertでは利用できない

```
var a;//グローバル変数としてaを宣言
SetA();//関数呼び出し
ShowAlert();//関数呼び出し
function SetA(){
    a = 4;
}
function ShowAlert(){
    alert(a);//4が表示される
}
```

aはグローバル変数。ほかの関数で変更されるとそのまま変数の中身も変化する

条件

- 条件に応じて、処理を分けるケースは多い
  - クリックされてもこれ以上右に動かせない→左に動かす
  - いわゆる判断をさせる場合に利用
  - 条件式を記述して判断させる
  - ifのみ必須
    - 他は任意
  - else ifは複数回利用可能

```
if(条件式1){
        条件式1に当てはまるならここが動作
}
else if(条件式2){
        条件式2に当てはまるならここが動作
}
else{
        条件式に当てはまらない場合に動作
}
```

条件式

- 0以外の数、true ，何らかの文字列は全て条件式で当てはまる
  - 0,falseだと条件式に当てはまらない

- 比較して判断することも
  - if(a > 3)→aが3よりも大きければtrueになる
  - if(a == 3)→aが3のときだけtrueになる
  - if((a >= 3) && (a < 100))→aが3以上で、100未満のときだけtrueになる
  - if((a >= 100) || (a < 10))→aが100以上または10未満のときだけtrueになる

条件式の成立

- 成立する場合
  - if(true)
  - if(1以上の数値)
  - if(‘何らかの文字’)
  - 条件式が成立しない場合
- 不成立の場合
  - if(false)
  - if(null)
  - if(undefined)
  - if(0)
  - 条件式が成立しない場合

ループ

- 同じ処理を繰り返す際に利用
  - いくつか方法がある
  - プログラムは楽するために利用する＝同じ処理はできるだけループを使って処理する
  - 条件が成立している限りループする
    - 初期値に0を入れて1回回るごとに1を足していくと100回回った段階で条件を満たさなくなる

```
for(var i =0; i< 100; i++){
    var a = i;
    document.write(a);//aを表示。0〜99まで表示される
}
```

```
for(初期値; 条件; カウントアップ){
    繰り返す内容
}
```