

Webデザイン実習4B

2019/11/12

Kazuma Sekiguchi

class@cieds.jp

スマートフォンのUI

- ほとんどのUIに何らかの動きが必要
 - タップしたことは分かりづらいため、反応が必要
 - ページを遷移しないように作成することが多いため、余計反応が無いと、タップできたのかどうか困る
 - 少しのインタラクションを付与＝マイクロインタラクション
- hoverは機能しないため、タップされた後にどうするか、が鍵
 - あまり過激なインタラクションを付与するとウザい感じがするので、ほどほどに

スマートフォンでの動き

- いくつかの方法を組み合わせる
 - CSS Transition
 - CSS Animation
 - SVG Animation
 - JS
- 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

サイズ

- 基本的にリキッドデザインであるため、%を上手く利用する
 - calc()関数を上手く利用することで、%とpxを並列で利用することが可能
 - 縦方向はpx指定で問題ない
 - スマートフォンの幅がそれぞれ違う点に注意
- 端から端まで利用するのが基本だが、内部でドラッグする要素がある場合は、端まで利用するのを避けて、動かすための余白を用意する
 - GoogleMapなどは2本指でドラッグしないと動かないように変更されている

ボタン位置

- スマートフォンでのメニューは大体右上にあることが多い
 - 70%くらいが右上・・・、らしい
 - 右上か、左上が多いが、PCと同じようにグローバルメニューのようにボタンを配置する手もある（任天堂など）
 - 画面中央に遷移用のボタンを用意する手もある
- サイドメニューとしてボタンをクリックしたら表示する場合、大体アニメーションを付与
 - 画面全体をズラしてズラした領域にメニューを表示
 - 画面の上にメニューを重ねる

動き

- タップして何かをする場合は、JSでclassを動的に適用、外すことでアニメーションを発生させる
 - transitionを利用すると適用した時と外したときにアニメーションが実行される
 - animationと使い分けて対応

スマートフォンならではのUI

- できるだけ大きなUIを用いたり、直感的なUIが好まれる
 - ジェスチャーがあまり多用されていると逆に分かりづらくなることもあるため、代用手段も用意しておく
- ユーザーに入力させるときは、写真などの方が楽なことが多い
 - できるだけユーザーの入力（特に文字入力）を減らす
- 写真や動画をアップする方がスマートフォンの方が楽
- 次を想像できるようなアニメーションやインタラクションを利用することで、使いやすいUIとなる

スマートフォンのイベント

- マウスではなく操作するため、イベントなども異なる
 - clickはあるが使わない方が良い
 - clickだと大体300msくらい遅れる
 - tapイベントを利用する
 - 但しJSでは存在しないため、自分で作成するかライブラリを利用する
 - Hammer.jsなどを使うと楽（但し書き方が独特）
- RWDの場合、clickも使うため、両対応にしておく必要がある

スマートフォンのイベント

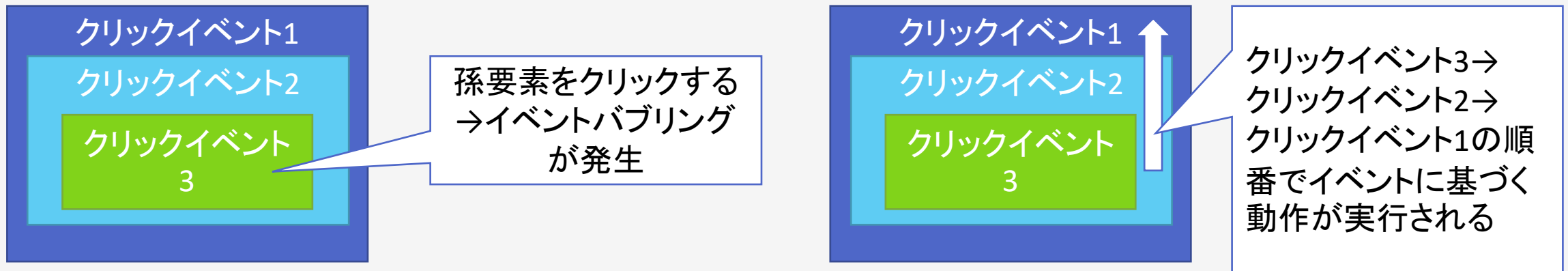
- 想像以上にiPhoneやAndroidは複雑
 - かなりの部分をJSで対応する必要がある
 - iPhoneの画面サイズ、positionの指定などでバグが結構ある
 - JSでサイズを取得したり、position:absoluteなどで対応する必要がある
 - iPhoneのMobileSafariはスクロールすると下に操作画面が出てくる→実質の画面サイズが変わる
 - 最近はAndroidもナビゲーションバーが消えるため同じような対応が必要なケースも
 - RWDの場合、JSを組み込む際に配慮しないといけない点

iOSにおけるタップ、クリックイベント

- iOS12までの要素はaタグのみクリックが可能
 - 他のタグはクリックしても無視される
 - JSなどでイベントを付与した場合に機能しなくなる
 - 解決策
 - aタグを使ってクリックされる要素を括る
 - CSSのcursorプロパティでpointerを指定する
- 後者の方が影響が少ない

イベントの伝播

- スマートフォンに限らずクリックなどのイベントは親要素に伝わる
 - 親要素にイベントを付与していた場合、そちらも動作してしまうことがある



- CSSの`pointer-event:none`でイベント発生させないことが可能
- JSの`stopPropagation()`を使うことで、親要素への伝播を防げる

hover

- スマートフォンでもhoverは存在
 - 但し、PCの場合はマウスが乗ったときにhoverが働く
 - スマートフォンでは指を離したときにhoverが働く
- 実行されるタイミングが違うので、使うときに注意が必要
- hoverはスマートフォンでも設定した方が分かりやすい
 - 指を離したときに何を押したのかを確認可能

PCとスマートフォンの違い

- ボタンの大きさ、UIの違いなど多岐に渡る
 - 一番大きいのが「肯定」と「NG」の指定がPCと逆
 - スマートフォンでは右側に「肯定」、PCは左側に「肯定」
- スクロールは意外にされない
 - スクロールバーが勝手に消えたりするからでは？と思ったり

画面位置で表示

- スマートフォンで多いのが、スクロールしたときに段々出てくる表現
 - lazyloadと組み合わせておくと読込が早くなる点は優位
 - 1ページサイトにしてしまった方がスマートフォンの場合は、操作がしやすいという点もある
 - 段々出てくる表現と同時に動きを付けてあげることで、画面に変化を付けやすい
 - アニメーションを付与して、印象づける
 - アニメーションを同じものを利用し続けると飽きられる、という指摘もあるため、画面画面で動きを変えるなどの対策は必要

MobileSafari

- MobileSafariでは画面のスクロールを停止した段階で下からメニューが出てくる
 - 上にスクロールしたときにも出てくる
 - 画面サイズがここで変化する点に注意（画面サイズを変更したことを検知して、レイアウトを変えるようにしている場合、実行される）
 - `$(window).height()`で取得しても、下側のボタン部分高さは取得されない
 - `window.innerHeight()`で下側のボタン部分を含んだ高さは取得可能
 - ボタンが表示されているときはOKだが、消えたときはぽっかり穴が開くなどの問題が生じる
 - 全画面表示をするときは要注意
 - 画面下部にメニューボタンなどを設置すると位置が変わるため、UI的に操作しやすいとは言えない

動かす

- 基本的にjQueryのanimateメソッドは使わない方が良い
 - 速度的な問題
 - CSSアニメーションを利用した方が現在では楽
 - 途中でアニメーションを強制的に止める必要がある場合などはanimateメソッドを利用する
- JSのclassList.addメソッドを利用して、クラスを付与してあげることによってタイミングを制御可能
 - 動かすときは、transform:translate(x,y)を利用すると便利

calc()を上手く利用する

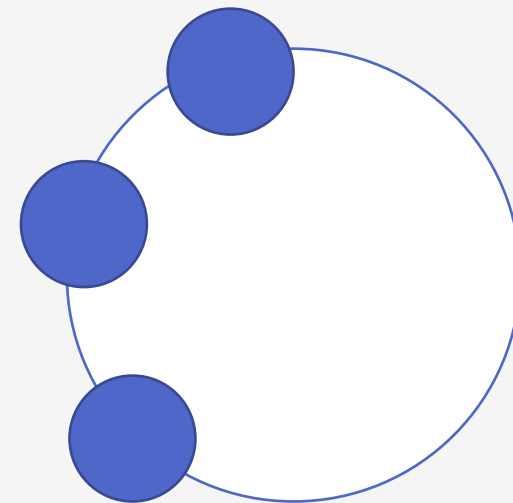
- calc()で計算が可能のため、これを上手く利用することで、JSの計算を省くことが可能
 - translate内などでも利用可能

```
transform: translate(calc(-2048px + 100vw),0);
```

- この場合、-2048pxに画面幅（100vw）を足して、そのサイズの位置に動かすようにしている
- @keyframesの定義はJSから書きづらいため、できるだけCSSだけで対応する

円周上への配置

- 要素を円周上に配置する場合は、sinとcosを使う
 - X軸の位置は Math.cos (ラジアン)で算出可能
 - Y軸の位置は Math.sin (ラジアン)で算出可能
- ラジアンとして角度を指定すればOK
 - $\text{ラジアン} = \text{角度} * \text{Math.PI} / 180$
 - 半径の値をさらにかけ算して上げれば位置が算出可能



スマートフォンで良くあるドロワー

- タップするとアイコンが変化して横からメニューが出てくる
 - アイコンをCSSで作成しておく
 - CSSで作成しておくことで、transitionによるアニメーションが可能になる
 - JSでクラスを付与する、外すとtransitionが発生する→アニメーションが動作する
 - メニューをwidthにアニメーションを付けるようにして展開させる
 - メインコンテンツ部分をtranslateX()を利用して横にスライドさせる
 - mainタグなどをスライドさせれば、全体をスライドさせることができる
- JSとCSSを上手く組み合わせるのがコツ
- viewportでinitial-scale=1を設定しておく
 - スライドしたときに全体が縮小されて表示されるのを防ぐ

スマートフォンで良くあるドロワー

