Webデザイン実習4B

2019/10/01

Kazuma Sekiguchi
class@cieds.jp

Agenda

- せっかくやったPHPをもう少し
- JS（避けては通れません）
- HTML5 API
- WebService連携（JSON）
- Canvasの利用
- スマートフォンNativeアプリの作成（一応Monacoを使う予定）
  - HTML+CSS+JSで作成

最近のWeb業界

- デザイナーの終焉、ディレクター化
- デザイナーにはコーディング知識が必須
  - JSもできたら、くらいにはなっている
  - あと2年くらいでJSは必須
- 高速にインタラクション性を持つデザインの提示
  - AiとかPsではのデザインはあんまりない
  - できるだけ早くプロトタイプを作成できるか、という点
- Nativeだからできること、Webだからこそできることの識別を付ける
- 相も変わらず技術進歩は高速

さて就活ですね

- 採用する側からすると
  - 何がしたいのかが明確
    - したいことに関する知識は必要
  - ここで（この会社）何がしたいのか
  - どのくらいできるようになるか

  だけ知りたい
- ぶっちゃけ今何ができるかはどーでも良い
  - 会社によってやり方は違うので、変化に合わせることができるかどうかが重要
  - 説明能力は必要

就活

- 学生時代に何をしたか、語れるかどうかは結構アドバンテージ
  - バイトも1つの手ではある
  - 何でも良いが、興味をもって行動しているということは人より意味を持つ
- 意外とWebはコネでいける
  - 小さい会社であればあるほど採用は難しいのは事実
  - 直接アプローチする、勉強会などで話を聞いてみる
    - 名刺作っておこう（社会人は渡されると返すという残念な習性がある）
  - 勉強会とかイベントでスタッフ側に回る手はある

業種別（この辺意識しておこうか）

- デザイナー
  - 問題意識、解決策の模索、完成デザイン、プロセスデザイン
    バリエーション、ラフなど
- UI/UX
  - 問題意識、仮説（使われる場所や状況の把握）、プロトタイプ
    バリエーション、フィードバックによる改善
- コーダー
  - CSSの命名方法への工夫、集団開発時の対応、JSなどへの対応
    新しい技術への対応、GitHubなどへの貢献
- バックエンド系
  - 特にポートフォリオは重視されない（というか要らない）
  - 何を作ったかをまとめておけばOK

今からできること

- 何をしたいかをちゃんと考える、調べる
  - デザイナー？　（仕事無いけど）映像？フロントエンジニア？バックエンドエンジニア？
  - 個人的には最初からディレクターというのはお薦めしない
- 途中成果物を残す
  - ラフ、ワイヤーフレーム、パターンなど
  - デザイナーは特に過程は大事
- バズワードに踊らされない
  - でも語句をきちんと把握して自分なりに考えておく

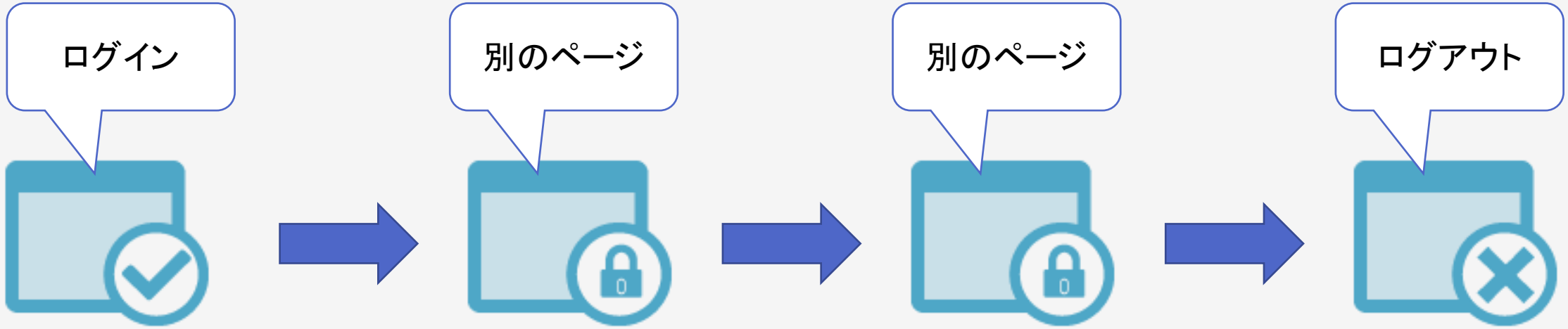
セッション

- 通常のプログラムでは、状態遷移を把握することが必要
  - 今ユーザはどこを見ているのか？
  - どんな状態か？

- 状態を把握できないと、プログラムを制御できない
  - プログラムを終了したのにいつまでも実行しているといったことが起こる
  - 通常状態遷移はアプリケーション側で実装する（イベントはOSから送られる）
  - 管理画面など認証が必要な画面では必須な要素

セッション

- ユーザがログインして、ページを利用した後でログアウトするまでの一連の流れ
  - ユーザ別に違う機能やページを表示する場合、認証が必要
  - 認証してその情報に基づいてページを表示する

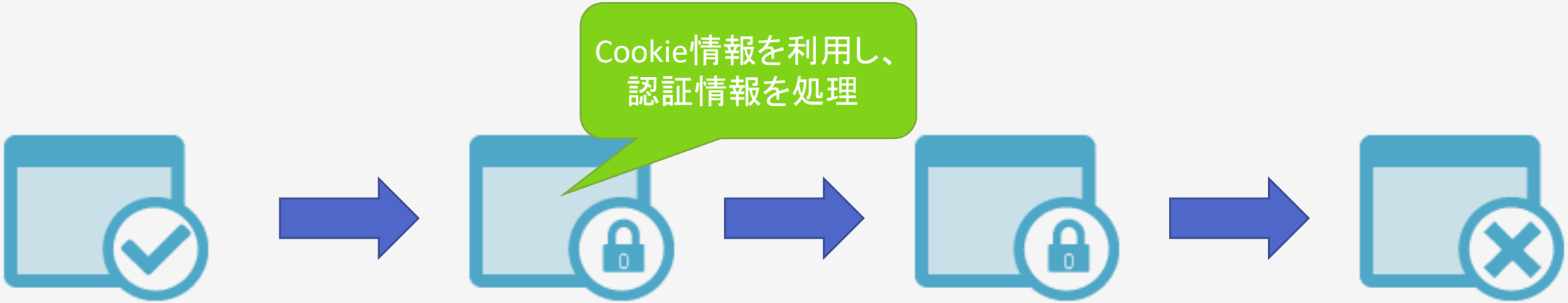


セッション

- 認証したら、きちんと認証したという状態を保持しておく必要がある
  - 保持していないと毎回認証していない、ということで認証させる必要が生じる
  - HTTPはステートレス＝認証状態なんて保持してくれない
  - ページが変わった瞬間に認証した、という事実も失われる

セッション管理

- 認証したら認証した事実を保持する仕組み＝セッション管理
  - 通常はCookieを用いて、クライアント側にあるデータを保持させる
    - Cookieは小さいデータを保持することが可能
    - GET送信する際にCookieのデータを同時に送信する
  - サーバはCookieのデータを判断し、状態を判断することが可能
    - 例えば、cookieから所定のデータが来ていればログインしている状態、と判断する（プログラムによっても異なる）



ハッシュ（1）

- プログラムや文字列を数学的に変換し、不可逆な文字にしたもの
- フィンガプリント（指紋）ともいう
- 同じ文字列ならばいつでも同じハッシュ値が計算される
- 1つでも文字が違う、またはプログラムが1カ所でも違えば、ハッシュ値も異なる

「php」のハッシュ値
e1bfd762321e409cee4ac0b6e841963c

「PHP」のハッシュ値
2fec392304a5c23ac138da22847f9b7c

ハッシュ（2）

- ハッシュ値を求めるのは簡単。但し、逆は不可能に近い（実際には計算できるが、数十年掛かる）→不可逆
- 何に使うか？
  - 妥当性チェック
    - 少しでも手を加えたらハッシュ値が変わる→偽造チェックになる
    - ダウンロードしたソフトが正しくダウンロード出来ているか？
  - パスワードチェック

ハッシュを使ったパスワードチェック

- 予めユーザの入れたパスワードをハッシュ値にして保存
  - ユーザの入れた平文のパスワードは破棄する
- ログインする際にユーザがパスワードを入力
  - ハッシュ値にして、保存しているハッシュと照合→同じならパスワードが正しい
  - ハッシュ値を入力されても入力されたハッシュ値のハッシュ値を比較に使うため、必ず違うものになる
- パスワードが万が一漏れてもハッシュならログイン出来ない
  - クレジットカード番号など元の文字列が必要なデータには使えない
  - ハッシュではなく、可逆暗号（元に戻せる暗号）を使うのが一般的

ハッシュアルゴリズムの種類

- MD5
  - もっとも使われているハッシュアルゴリズム。理論的な弱点があるため、最近ではあまり推奨されない
  - 違うファイルで同じハッシュ値になる可能性があることが証明されたため、使わない方が良い
  - ファイルの真贋判定レベルなら、高速なため意味はある
- SHA1
  - MD5よりも攻撃に強いとされるハッシュアルゴリズム。様々なセキュティ関連のプロトコルに採用されているため、今後はこちらの使用を推奨されている。SHA1,SHA-224,SHA-256,SHA-384,SHA-512などがある
  - 現在はSHA1も危険と言われている（SHA-256とかを利用）

管理画面の作成

- 管理画面はセッションでログインをしているかどうかを判定する
  - ログインしていない場合はログイン画面に遷移させる
  - ログインしている場合は、必要な画面を表示させる
- セッションの管理が重要
  - ログアウト時はセッションを破壊してログアウトとする

管理画面の作成

- 通常の機能
  - 登録（Create）
  - 読み出し（Read）
  - 更新（Update）
  - 削除（Delete）

  の各機能が必要

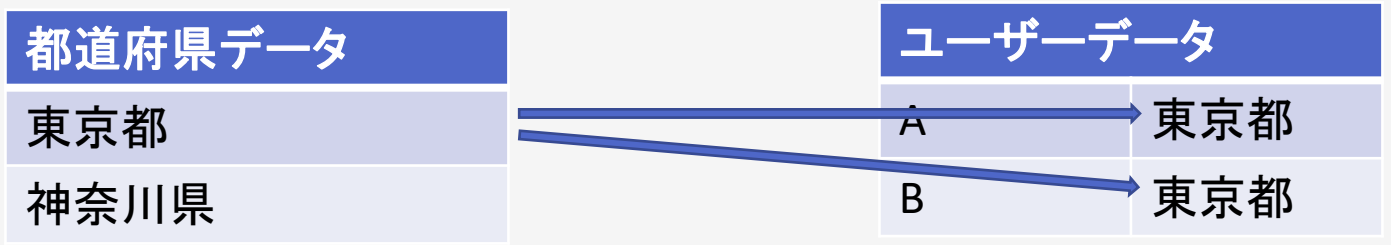
  1つの機能を作るために4つの機能が必要
- 実際には削除はしないことが多い（論理削除）
  - 削除フラグなどを立てて、削除したと見なすようにする
  - 万が一の際にDBにアクセスして復旧させることが可能

管理画面の作成

- 更新
  - 作成するのが面倒なものの1つ
  - 既に登録されているデータを取得して適切な場所に表示させる
  - アップデート時に新規登録時と同様の値のチェックが必要
  - どれを更新させるかキーとなる値を付与して、更新をするプログラムにデータを渡す
    - キーとなるもの＝ほとんどはID
  - キーを利用してDatabase内のデータを更新する
    - 更新時にキーを利用する点に注意

データベースの構造

- 基本的にデータベースは１対１の関係にあるデータを保存する
  - 例：Aという項目のデータがZ
- １対Nの関係にあるデータは工夫しないと保存できない
  - 複数のテーブルを組み合わせて表現を行っていく



  - この関連性をきちんと表現していくことが必要
    - 正規化