Webプログラミング実習1

2017/6/21

Kazuma Sekiguchi
class@cieds.jp

課題

- 今までにやったJSの技法を活かしてサイトを作成
  - インタラクションのあるサイトを作成
  - インタクションは効果的に利用すること
  - jQueryの使用OK、jQueryプラグインはそれだけを利用する訳では無いならOK（プラグイン以外も利用する）
- テーマ
  - 梅雨や夏に楽しめるもの、こと、場所の紹介サイト
  - 写真などの素材はネットから取得しても可（商用利用可のもの、CC0ライセンスのもののみ許可）
  - 自分で作成したキャラクターなどは許可
  - トップページのみ作成すればOK

課題

- 提出
  - デザインカンプ（Ai、Ps、Xd、Sketchなど）
    - AiやPsは画像を埋め込んでください
    - JPEG形式などにしないこと（aiとかpsdのまま）
  - HTML、CSS、JS、画像群

- 期限
  - 7月5日授業時間までに提出

Ajax

非同期通信による情報更新

Ajax = Asynchronous JavaScript + XML

- 非同期でサーバとデータをやり取りする方法
  - ウェブサーバからデータを貰う
    - ユーザがURLを入力、リンクをクリック
      - 同期
    - ユーザがクリックしない、リロードしない
      - 非同期
- Ajaxでデータをやり取りしている間も操作可能
- 非同期とか言っているが、実際にはイベントドリブン
  - 状態の変化でデータをやり取りする

どこで使われているか

- スマホのアプリ
  - バックエンドでサーバーから情報を取得して更新
    - データを更新している間も利用可能
- バックエンドシステムと結合しているフロントエンド
  - 投稿時に画面が遷移しないタイプのシステムは間違いなくこれ
    - Facebook、Twitterなど
  - 情報のやりとりをAjaxで行うことで、画面を真っ新にしないで更新する
- 外部APIを利用して表示する場合
  - GoogleMapなども外部API

Ajax実行タイミング

- タイミング
  - HTMLを読み込んだ後で大きなデータを後からダウンロード
    - ユーザの待ち時間減少
      - GoogleMapなど
  - ユーザがクリックしたときに必要なデータをダウンロード
    - ページ遷移無しにデータを表示
  - テキストフィールドの変化でデータをダウンロード
    - 送信しなくても保存、データの更新が可能
      - Googleの検索ボックスのサジェスチョン

Ajaxのメリット、デメリット

- メリット
  - データの読み込みなし（ユーザにはそう見える）にコンテンツが進行できる
    - スムーズに進行
- デメリット
  - 開発が少し面倒
  - サーバ側の処理に依存する部分も出てくる
  - ユーザが「更新」ボタンとか押してしまったら、アウト
    - 更新されても元に戻るようにしておくなどの対応が必要
    - 戻るボタンは、HTML5対応ブラウザであれば対応出来る

JSON

- JSでデータを受け取ったりする場合展開が容易なため良く使われているファイル形式
  - JSのオブジェクトと同じような形式で記述
  - jQueryで展開が容易
  - 旧来はXMLでデータを取得していたが、JSONで取得するのがほとんど
    - ex:Twitterのデータ、Amazonのデータなど

```
{
        "name":"Dreamweaver",
        "version":"CC",
        "price":"4,000/month"
}
```

```
[{
        "name":"Dreamweaver",
        "version":"CC",
        "price":4000
},
{
        "name":"Photoshop",
        "version":"CC",
        "price":4000
}]
```

クロスドメインセキュリティ

- Ajaxの場合、後からデータを取得可能なため、ドメインを跨がって取得することも可能

  なはずなのだが、ブラウザのセキュリティにより不可能にしているブラウザが多い

  - ユーザに見せているHTMLと後から取得するデータのドメインが同じでないとデータを取得出来ない、ことがある
    - Firefox , GoogleChromeなど
    - ドメインを跨ぐならJSONPを利用する

JSONP

- 基本的にはJSONと同じ
  - ブラウザーのセキュリティで後からデータを落とす場合（JSONは後からデータをJS側で取得させる）元のサーバと同じURLからしか落とせない
  - 別サービスにアクセスしてJSONデータを取得してごにょごにょするのが無理

JSONP

- <script>タグで呼び出せば、ほかのサーバからでも落とせる
- JSONではデータなので、落としても上手くいかない
  - 関数の形でJSONを呼び出す
    - 呼び出すURLの後にgeo.php?callback=?みたいにcallbackを付けてJSONを呼び出す
    - このタイプのJSONがJSONP
    - 処理は変わらない
      - jQueryなら$.getJSONで処理できる
- Ajaxでは当たり前のもの（Pjax）
  - 末尾はXMLなのにね

$.getJSON

- JSONを非同期的にダウンロードするjQueryの機能
  - JSONの形式を展開してくれるため、非常にあとの処理が楽
  - 現在Ajaxというと、データはJSONが多いので、良く使う機能

```
[{
“name”:”名前1ですよ”,
“address”:”住所ですよ”
},
{
“name”:”名前2ですよ”,
“address”:”住所ですよ”
}]
```

JSON

```
$.getJSON(“JSONまたはJSONPのURL",function(json){
        for(var i in json){
                $(“#contents”).append(“名前：”+json.name+”
住所”+json.address”);
        }
});
```

ノンリロードコンテンツ

- 通常のウェブページの場合、ページ遷移などを行った場合は、一度ブラウザーの画面を初期化（真っ白）にしてデータを読み込み、再度レンダリングを行って表示を行う
  - 非常に手間の掛かるプロセスを経ている
  - ステートレスなため、セッションなどを利用している場合は、変数などをCookieなどで保持しておく必要がある
- Ajaxなどの技術を使うと後から必要なデータを取得することが可能
  - この場合、JSでデータだけを取得してくるため、ブラウザーが初期化されることがない＝リロードがない＝ノンリロード

JSで取得

- JSでデータを取得したら、JSでデータをパース（使えるようにデータを加工する）して、適切な位置にデータを挿入または置換して入れ込む
  - 更新、変更のあった部分だけ修正、アップデートすることが可能
  - ユーザはその間も操作することが可能
    - ユーザが操作できる＝使えてしまう、という点は注意が必要
    - 読み込み完了前にも操作できてしまうため、誤作動の原因になることも
    - 読み込み完了したら画面表示するようにするなどの工夫も必要

データの送受信

- 昔はXML形式で実行
  - 現在はJSONまたはJSONPで行うのが通常
  - JSONであれば、JSでのパースが極めて容易かつ軽量
  - バイナリデータ（画像データなど）は含められないので、画像を使う場合は、URLを指定しておき、改めてJS側で取得し直すようにする

- やりとりは全てバックエンドで行われるため、どのURLにアクセスしてどういうデータを取得しているか、は開発者ツールなどで確認する必要がある
  - 正しいJSONが返ってきているかどうかなどを確認していく

タイムアウト処理を入れた場合

```
$.ajax({
        type:"GET",
        url: http://wbclass.net/json/json.php,
        cache: false,
        timeout: 10000,
        success: successHandler,
        error: errorHandler
});
```

```
function successHandler(data)
{
   alert("通信成功");
}

function errorHandler(XMLHttpRequest, textStatus,
errorThrown)
{
   alert("通信失敗");
}
```

- 10秒経ってもデータが来ない場合、エラーになる
  - 成功したときは、successHandler関数が呼ばれる
  - 失敗したときは、errorHandler関数が呼ばれる

何が面倒か

- HTMLとCSSの場合、全てDOMに依存して表示を行う
  - テンプレートエンジンを用いても最終的にはDOM依存
  - クラス名やID名、タグの順番などに依存してしまうため、ちょっとした変更に極めて弱い
  - 特にjQueryで作成した場合は、顕著
  - 各パーツをできるだけ分離することで、ほかの部分に影響を及ぼさないような設計を
    - CSSとの兼ね合いも気をつける

画像をAjaxで取得

- 画像ファイル自体を送受信することは比較的面倒
  - Ajaxのシステム自体がテキストファイルの送受信用
  - 画像ファイル自体を送る場合は、DataURI形式のテキストにして送り出す→Ajaxとしてデータを送ることが可能
- 通常画像ファイルを送受信したい場合は画像の表示目的
  - 画像を表示するためのURIを送信してあげればOK
  - ブラウザー側に動的にimgタグを追加してあげれば、src属性で指定されたファイルをブラウザーは取得しに行く
  - 画像ファイルの受信はバックグラウンドで行われる（Ajaxに限らず）

画像をAjaxで受信

- サーバー側のJSONでファイルパスを送り出す
  - 受信側はパースしてimgタグとして展開する
    - ブラウザーは受信した画像から表示していく（時間差が生じる）
  - タイムラグ無しに展開したい場合は、new Image();でイメージオブジェクトを作成して、JS側で処理をしてから展開を行うようにするとタイムラグ無く展開可能
- 画像ファイルの高さと幅は分からない
  - JSで判別することもできなくは無いが、面倒
    - 指定しなくても特に問題無いので無視でOK
  - サーバーサイドで送ってくれるならそれを利用する

スクロールバーを下にしたときに読み込み

```
$(window).scroll(function(){
	lastTop = $("#next").offset().top;
	scrollBottom = $(window).scrollTop() + $(window).height();
	if((scrollBottom >= lastTop) && (flg == 1)){
		flg = 0;
		setTimeout('next()',1000);
	}
});
```

- スクロールバーが所定の場所まできたら次のデータを読み込むようにすることも可能

スクロールバーを下にしたときに読み込み

- 所定の場所の位置を測定
  - $(“”).offset().topで取得できる
  - $(window).scrollTop() + $(window).height();でスクロールの下側位置を取得可能
  - 双方を比較すれば、所定位置に届いているかどうかを確認出来る
- 届いていたら、読み込む用のスクリプトを実行する
  - フラグを利用して一時的に次の読み込みをしないようにする
  - setTimeoutを利用すると、直ぐに読み込まれず、ワンクッション置くことが可能

jQueryの$.ajax

- 細かい制御を行うことのできるAjax通信用メソッド

```
$.ajax({
  type: "POST",
  url: "some.php",
  dataType:”json”,//JSONデータを受け取る場合指定
  timeout: 10000,//タイムアウトにする秒数（ミリ秒）
  data: { name: “John”, location: “Boston” },//送り出すデータ
  success:function(){
        //データを受信できた場合の処理
  },
  error:function(){
        //データを受信できなかった場合の処理
  }
});
```

Ajaxによるフォーム

- ユーザに入力してもらったデータをサーバーに送信
  - サーバーでバリデーションを実行
  - 問題があれば、JSONデータとしてエラーを返す
  - 問題なければ、受信したデータをJSONデータとして返す
  - 確認してもらって問題なければ、サーバーに登録
  - 画面がいちいちリロードしないので、スムーズな動きを実現
  - ユーザの入力してあるデータを再利用可能（セキュリティに注意）

- データのやりとりはJSON形式で行う
  - サーバー側から送り出すとき

送信ボタンを無効にする

- submitボタンなどを押されると、action属性で指定されているURLへと遷移する
  - Ajaxとしての意味がない
  - ユーザーのクリックなどを無効にして処理を奪う

```
$(“input”).submit(function(e)){
e.preventDefault();
}
```

  - preventDefaultを指定することで、ユーザーの動きをキャンセル可能（必ずイベント識別子を変数として渡すこと。渡さないと何を阻害してよいかわからないため、エラーになる）