jQueryを使用した実装

作業を始める前の準備と注意事項

まずGitHubから作業環境をクローンして来てください。

GitHub

それができたらブランチを自分の名前で切ってください。

Q1～Q2までフォルダーがあります。それぞれの中に`sample`と`work`にフォルダーが別れています。

`sample`は動作確認用で`work`は実際に作業してもらうフォルダーです。

今回課題で共通なのは、HTML/CSSファイルは`sample`の中に入っているものとまったく同じなので編集する必要なありません。

コードの理解度も確認したいので1行ずつ何をしているのかコメントを書いてください。

```
$(function () {
  〇〇を〇〇した時に〇〇
  $("#sample").on('click', function () {
    変数に文字列代入
    const result = "ボタンが押されました！";
    ブラウザのコンソールに変数の中身を出力
    console.log(result);
  });
});
```

また、`sample`のjQueryのコードはコピペ防止の為にPC、ブラウザが処理しやすいよう変換していますのでそれをそのまま使っていると判断した場合、やり直しまたは、別課題をやっていただきます。

提出方法

完成したものをGitHubへプッシュしていただきレビュー依頼を出してください。

また、質問する際も途中のもので良いのでプッシュして質問してください。

課題について

まずはサンプルで動作確認を行い、どの要素がイベント対象にするべきなのかHTMLの観察をしてください。

ネットで調べるもしくは本を購入してそれで調べながら実装を進めてください。

オススメの本のリンクも貼っておきます。

jQuery最高の教科書

実装のコツですがいきなり全部を作るろうとするのではなく、少しずつ確認しながら作業していくと良いです。（クリックしてボタンを押した時に動くか確認、変数に正しく値が入っているか確認、ifの処理が正しく動いているかなど）

いきなり全部実装して動作確認を行なった際、もし動かなかった時にどこが間違えているの原因を特定するのに時間を取られてしまうのでそう言った実装はしないようにしましょう！

Q1

作業項目が10個あります。

どうするのかは問題ごとのタイトルとサンプルを見ながら作業してください。

- Q1-3

フェードアウトする時間は3秒でお願いします。

- Q1-6

アニメーションの動作内容ですが`margin-top: 100`と`margin-left: 100`を指定し、2秒かけて動くようにしてください。

- Q1-9

対象の要素のindex番号を取得する処理を作成するとできます。

Q2

モーダルウィンドウの実装です。

サンプルを見て何をフェードイン、フェードアウトさせているのかを観察して見てください。

Q3

ハンバーガーメニューを押した時にハンバーガーメニューが×印に変わり、メニューが出てきます。

要素に対してクラス名の追加はここまでの問題でやったかと思います。

まずはそれで作成して見てください、

正しく動作した場合は次に進んでも良いですが動かない場合は、他にもクラスを追加する方法がないか考えて見てください。

Q4

タブメニューの実装です。

ホーム、ページA～Eのどれかを押すと下の文字の表示が変わります。

本来はすべて表示されますがCSSで隠しています。

その役割をになっているものから見つけてください。

あとはこれまでの応用でできます。

Q5

ドロップダウンメニューの実装です。

イベント発火の対象とDOMの操作を行う対象の見極めが大事です。

完成したら動作確認してサンプル通りできているか確認してください。

Q6

セレクトメニューでカテゴリーを選択するとそれにあった項目へ絞られます。

HTMLを観察して見ると共通しているものがあると思います。

それを使用すると絞り込みが行えます。また、「全て」を選択した時は全項目が表示されるよう実装してください。

ループを作る必要がありますが、素のJavaScriptぽく書くなら`for`でも良いですし、`each`を使用したループでも大丈夫です。

Q7

以前作成した入力フォームを使用しています。

今回はそれぞれの項目の値をコンソールログに出力させてください。

inputフォームで作られたものは入力した値、セレクトメニュー、ラジオボタン、チェックボックスなどは選択した項目のvalueの値を表示させてください。

どの項目なのか分かりやすよう項目名もコンソールに表示させてください。

誕生日はそれぞれ結合させて表示させてください。（サンプル参照）

Q8

APIを使用した実装をしてもらいます。

APIとは自己のソフトウェアを一部公開して、他のソフトウェアと機能を共有できるようにしたものです。

ソフトウェアの一部を公開し誰にでも使用できます。

なので自作アプリなどでもAPIを利用した機能実装することも可能です。

たとえば会員登録が必要なサービスなどで新規登録ではなく、TwitterやLINEの情報と紐付けて新規登録の手間を省くことができます。

そう言ったものもAPIを利用して実装されてます。

今回は国立情報学研究所（NII）が提供しているAPIを使用して、全国各地の大学図書館等約1200館が所有する、図書情報の検索機能を実装してもらいます。

APIの実行には`Ajax`を使用します。

説明すると長くなるので解説している記事を貼っておきます。

初心者目線でAjaxの説明

「分かりそう」で「分からない」でも「分かった」気になれるIT用語辞典

記事を読んでいる上で説明します。

2つともJavaScriptで利用するみたいな書き方をしていますが今回の書き方はjQueryを使用した場合の実装なのでJavaScriptで書く場合にまったく同じ書き方はできないことを覚えておいてください。

```
// 変数settingsに設定情報などを格納
const settings = {
  // 実行するURL。実行するURLのことをエンドポイントと言います。
  "url": `https://ci.nii.ac.jp/books/opensearch/search?
title=${searchWord}&format=json&p=${pageCount}&count=20`,
  // サーバーに送るメソッド
  "method": "GET",
}
// Ajaxの実行
//.doneが通信成功した時の処理、”response”が引数となっていて通信した結果を受け取っている
$.ajax(settings).done(function (response) {
//.failが通信に失敗した時の処理、”err”にサーバーから送られてきたエラー内容を受け取っている。
}).fail(function (err) {
});
```

上記のコードはworkのJSファイルにコメントで書いているのでそれを元に実装してください。

実装の準備

VSCodeとChromeにそれぞれ拡張機能を追加してもらいます。

VSCode

APIの通信はできますが、失敗した時のレスポンスが返ってこないのでそれを受け取る為の簡易的なサーバーを立ち上げるものです。

更新した内容も自動で反映してくれます。

今回の作業ではこれを使用して作業してください。

【Live Serverの使い方】自動でブラウザが更新される便利なVSCodeの拡張機能

たったの3ステップでライブリロードが可能になるVisual Studio Codeの拡張機能「Live Server」がすごい！

Chrome

こちらはサーバーの実装を行なっていないのでAPI通信をした際に「あなたのサイトからのリクエストは許可していません。」みたいな内容のエラーが返ってきます。

開発上なんの問題もないのですが、都度エラーが表示されるのはあまり良い気もしないので下記の拡張機能を入れると表示されなくなります。ただ、一時的な解決であって根本的な解決ではないのです。

CORS Unblock

APIの仕様

下記のURLからAPI仕様が記載されています。

学術コンテンツサービス サポート

APIの使用も開発元よって違うのでまずどのような仕様になっているかを確認する必要があります。

実務だと仕様書が何かしらの形で確認できるようになっているのでAPIだけでなく何か実装、作成する時は仕様書などと言ったドキュメントを確認するようにしてください・

実装ポイント

- 入力した内容を`searchWord`に代入
- `pageCount`の初期値は1、同じ検索ワードで検索を行う場合は、pageCountに+1する。違う検索ワードの場合は1に戻す
- 通信成功、失敗の処理をdone、failの中に書いていくと見辛いコードになってしまうので関数化させてそれを呼ぶだけの処理とする
- レスポンス内容から値を取得する必要があるが、「@、:」などが使われているのでそう言った項目から値を取り出す時は、`〇〇['@や:が使われているプロパティ名'];`とすれば取り出せる（〇〇は引数、変数を想定）
- 検索結果がなかった場合でも検索結果があった場合と同じレスポンスを返す為doneの処理に入っていくで成功した時の処理で条件分岐を作る必要がある
- どう言った場合にエラーが起きるか想定してエラー処理の実装、念の為それ以外のエラーがおきた時の場合も考慮して実装
- リセットボタンの機能実装

開発時の注意点

一部のAPIは有料なものや利用料に応じて請求金額が変わるものもあります。

なのでむやみやたらとGitHubに上げるのは気をけてください。

会社で契約しているものを無断利用しプライベートのGitHubアカウントで上げた場合、損害賠償が発生するのでAPIの取り扱いには気をつけてください。