WEBプログラミング/演習
第7回

スケジュール

- 第1回：Webプログラミングの基本
- 第2回：JavaScriptの基本
- 第3回：オブジェクト
- 第4回：関数
- 第5回：オブジェクト指向
- 第6回：クライアントサイドJavaScript
- 第7回：公開APIの利用(1)
- 第8回：公開APIの利用(2)
- 第9回：Pythonプログラミングの基礎(1)
- 第10回：　Pythonプログラミングの基礎(2)
- 第11回：サーバサイドプログラミング(1)
- 第12回：サーバサイドプログラミング(2)
- 第13回：サーバサイドプログラミング(3)
- 第14回：学習内容の振り返り

公開APIの利用（1）

CoursePower（第7回講義>第7回プログラム一式）からダウンロードください．これらはexercise7（フォルダ名）として扱いましょう．

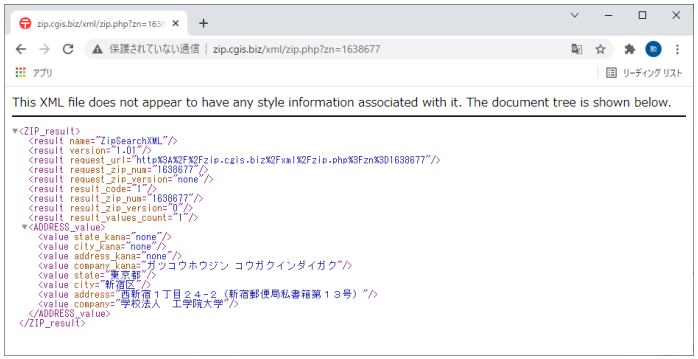
今日の課題について

- 今日はGCPを使います！！！
  - 使い方は第1回目の資料もしくは本資料最後の付録を参照のこと.
- まずは，exercise7のフォルダの内容を，GCPにアップロードください.

- GCPが必須なわけではなく，Webサーバが必須になります.
  - ローカルで，webServer.pyを動かしても実施可能だとは思います.
  - この意味がわかる人のみローカルで実施して構いません.

WebAPI

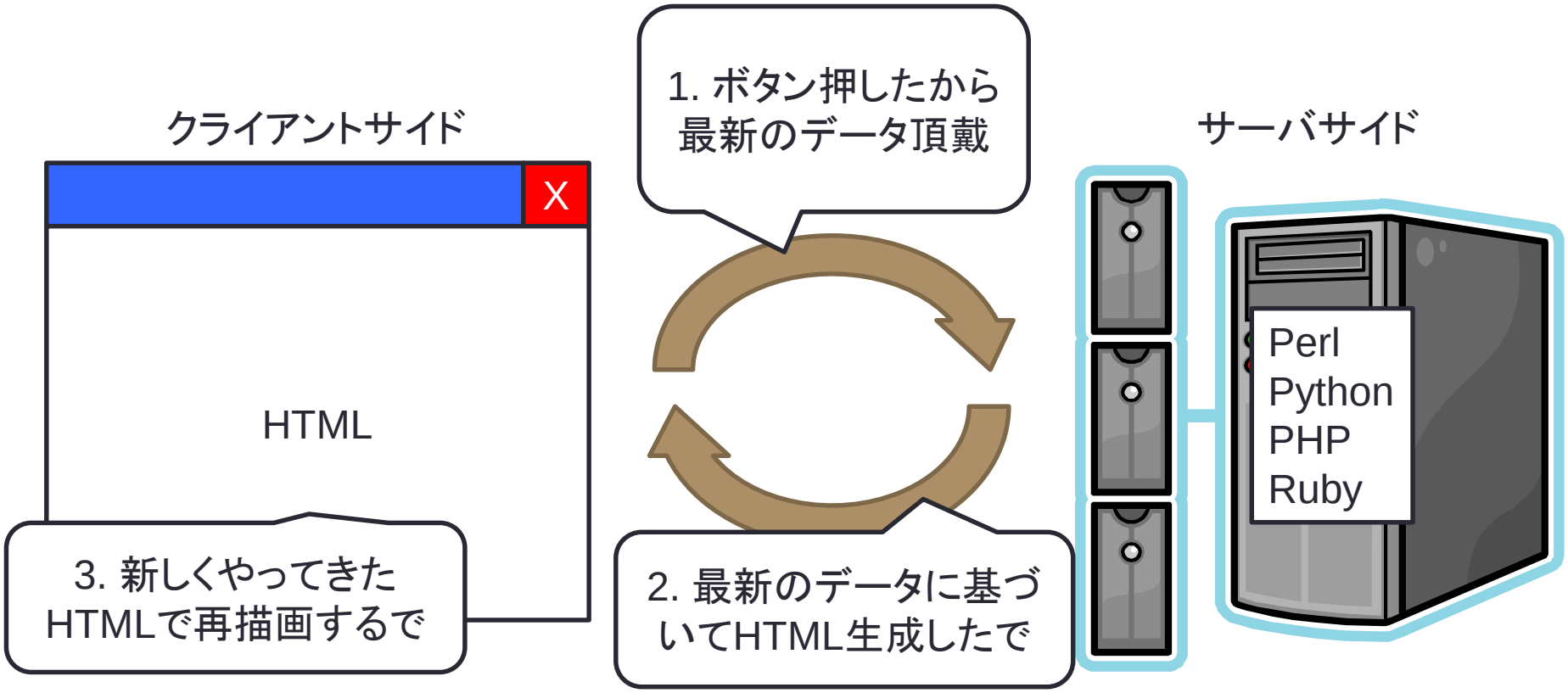
- WebAPIはインターネット経由で使える関数のようなもの
  - 引数...　　URLに指定して渡す
  - 戻り値...　　XML もしくは json で返す
    - 複雑なデータを汎用的な半構造表現で返すことができる
- 例：郵便番号検索API
- http://zip.cgis.biz/xml/zip.php?zn=1638677
  - zn に 郵便番号を指定すると，住所が返ってくる（下図参照）．ブラウザによっては一見何も表示されないこともある．
  - 適当に郵便番号部分（自宅とか）を入れ替えてアクセスしてみよう
- 利用例
  - 郵便番号を入れたら住所を自動的に補完してくれるプログラム



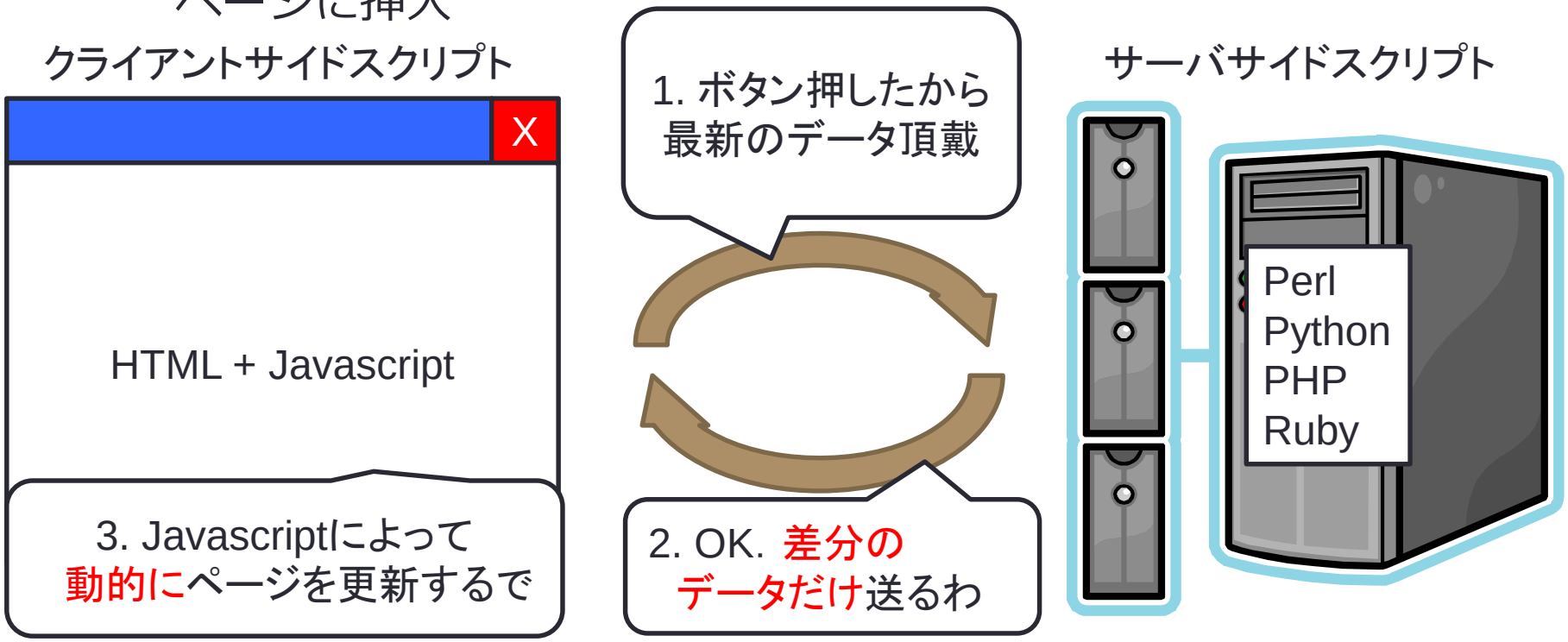
Chromeでの実施例

通常のWebシステム

- HTMLの更新依頼をサーバに送り一括更新
  - 掲示板の例：
    - 掲示板の更新ボタンを押したら，現在までの記載内容をすべて反映させた新たなWebページが送られてくる



- 必要なデータだけ取り寄せて，ページの切り替えなく更新（Webの非同期更新技術）
  - 掲示板の例：
    - 掲示板の更新ボタンを押したら，最終取得と現在時刻までの差分データをサーバから取り寄せて，JavaScriptで現在のWebページに挿入



AJAXの流れ

1. XMLHttpRequestオブジェクトを生成する
   - JavaScriptでサーバと通信するためのオブジェクト
2. サーバ通信時の処理を定義
   - 通信が成功した時の処理，失敗したときの処理を記述
3. 非同期通信を開始
   - 実際に通信を実行

- これを生のJavaScriptで書くのは結構大変なので，簡便に行えるライブラリが存在する
  - jQuery，React，etc，etc...
- この講義では特定のライブラリに依存せずに，何をしているのか理解するために「生のJavaScript」で頑張る

XMLHttpRequestオブジェクト

- 主要なプロパティとメソッド

response	応答本体を取得
readyState	HTTP通信の状態を取得
responseText	応答本体をプレーンテキストとして取得
status	HTTPステータスコードを取得
onreadystatechange	通信の状態が変化したタイミングで動作するイベントハンドラー
abort()	現在の非同期通信を中断
open(...)	HTTPリクエストを初期化
send(body)	HTTPリクエストを送信（bodyには要求本体が入る）
setRequestHeader(...)	リクエスト時に送信するヘッダーを追加

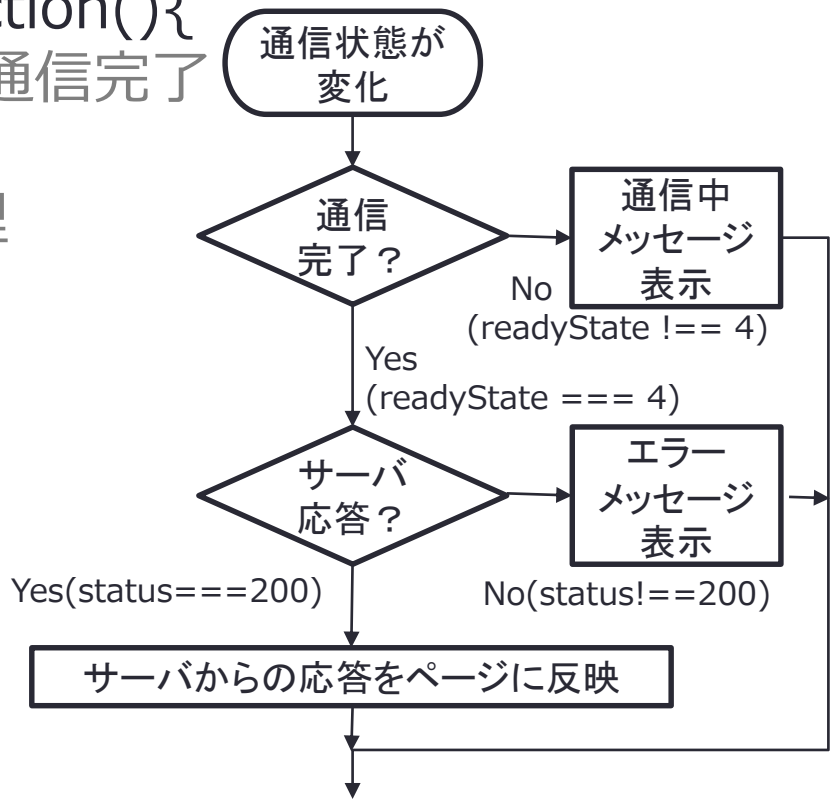
- XMLHttpRequestオブジェクトの作成

```
let xhr = new XMLHttpRequest();
```

サーバ通信時の処理を定義する

- onreadystatechangeのイベントに，通信開始から終了までに実行する処理を書く
  - サーバから正常な応答を受け取ったらページの更新処理をする
  - サーバがエラーを返したら，エラーメッセージを表示する
  - サーバへの通信を開始したタイミングで「通信中」と表示する

```
xhr.onreadystatechange = function(){
  if(xhr.readyState === 4){ //通信完了
    if(xhr.status === 200){
      ...          //通信成功時処理
    } else {
      ...          //通信失敗時処理
    }
  } else {
    ...          //通信開始時の処理
  }
}
```



通信状態の値

- readyStateの値とその意味

0	未初期化（openが呼ばれてない）
1	ロード中（sendが呼ばれてない）
2	ロード済み（応答が返ってきてない）
3	一部の応答を取得（ステータス/ヘッダーのみ）
4	すべての応答を取得（本体も含めて完了）

- statusの値とその意味

200	OK（成功）
401	Unauthorized（認証が必要）
403	Forbidden（アクセス拒否）
404	Not Found（リクエスト先が見つからない）
500	Internal Server Error（内部サーバエラー）
503	Service Unavaliable（サーバが利用不可）

onreadystatechange以外のイベント

- onreadystatechange 以外の便利なイベントないの？

loadstart	リクエスト送信時に発動
progress	データ送信中に発動
timeout	リクエストがタイムアウトしたら発動
abort	リクエストがキャンセルされたら発動
load	リクエストが成功したら発動
error	リクエストが失敗したら発動
loadend	正常/異状に関わらず，リクエストが完了したら発動

- 2ページ前のコードは以下のように書き換え可能（readyState/statusによる判定が不要になるのでコードが書きやすくなる）．

```
xhr.addEventListener('loadstart',function(){...
        },false); //通信開始時の処理
xhr.addEventListener('load',function(){...
        },false); //通信成功時処理
xhr.addEventListener('error',function(){...
        },false); //通信失敗時処理
```

オブジェクトができたので通信開始

- openメソッドでリクエストを初期化

  xhr.open(method, url [, async [, user [, passwd]]]);
  - method：GET/POST/PUT/DELETEなどのHTTPリクエストの種類
  - url：接続先のURL，殆どの場合，Web APIを提供するURL
  - [] はオプションで，省略可能な引数
  - async：非同期通信かの真偽値（デフォルトはtrue）
  - user, passwd：認証時のユーザ名，パスワード
- sendメソッドで通信をスタート

  xhr.send()

- GET method
  - データを取得することを目的とした方式
  - URL中にサーバに渡すデータを埋め込む（限界がある）
- POST method
  - データを送信することを目的とした方式
  - リクエストヘッダにデータを埋め込む

GETメソッド

例：
http://pikachu.com/pika?*p1*=abc&*p2*=123

- 上記のURLがあった時
  - ?の左側がURLの本体である
  - ?の右側がGETで渡されるパラメータである

- パラメータについて
  - &で複数のパラメータが接続されている
  - xx=yyの形で渡され，xxが ***ラベル***，yyが値となる
  - この例の場合，*p1*と*p2*というデータがサーバに送られ，それぞれの値は，’abc’と’123’である

[課題1] GETメソッドのイメージをつかもう

- googleのURLは，実はGETメソッドでgoogleのサーバにリクエストを投げ，その結果がHTMLとして返ってきてブラウザに表示されているだけである
  - そのため，検索キーワード入力欄にキーワードを入力しなくても，URL中のキーワードを意味するパラメータを変更して送信すれば，その検索結果が得られる

- googleで university を検索し，検索結果が表示されたときのURLを見よ
  - そのurl中で， university が値として与えられているラベルを特定せよ（xx=universityとなっているxxの部分のこと）
  - URL本体と，そのラベルだけ残し，ラベルの値として，kogakuin を与えよ
  - その結果，kogakuinの検索結果が表示されることを確認せよ

[課題2] とりあえず動かそう

- Web_API1.htmlを開いて，課題2用のボタンをクリックすると，一通り動く状態になっている.
- 2つのアクセス先（urlとurl2）があり，１つは成功し，もう1つは失敗する.
- 表示内容を確認し，どちらのURLが失敗しているか突き止めよ
  - また，デベロッパーツールのコンソールにて，エラー内容も確認せよ.
  - このエラーは，[課題3]以降の話に関係する

JSON

- AjaxはAsynchronous JavaScript + XMLの略ですが，XML以外も扱えます
  - XMLはExtensible Markup Languageで，タグでデータ構造を表現するものです
    - よくわからない人はHTMLのようなものと思ってOKです
  - XMLは以下の欠点によりajaxではあまり使われない
    - 開始タグ，閉じタグで表現するのでデータが冗長になる
    - DOM操作でアクセスするのでコードが冗長になる

- XMLのかわりに，JSON（JavaScript Object Notation）という形式が一般的に使われる
  - JavaScriptのオブジェクトリテラル表現をもとにしたデータ形式で，そのままオブジェクトとして取り込める

JSON形式のデータの例

- ラベル:値 の形式で格納されている
  - オブジェクトリテラル表現と同じ

```
{
  “isbn”: “978-4-7741-8030-4”,
  “title”: “Javaポケットリファレンス”,
  “publish”: “技術評論社”,
  “price”: 2680
}
```

オブジェクトリテラル例:

```
let obj = {
  name : ‘はにゃらら’,
  age: 18
};
```

全体を{}で囲み，その中で”:”で区切ったキーと値のペアを記述する．キーと値のペアが複数ある場合は”,”で区切る．

- 多くのWebAPIでは，このようなJSON形式でデータを返してくる

はてなブックマークAPI

- はてなブックマークとは？
  - ソーシャルブックマークサービスの１つ
    - Webページに対して独自にキーワード付け（タグ付け）して整理しようってサービスだと思えばよい
    - Webページに対してコメントを付けることもできる
  - つけられたキーワードやコメントは，「Webページ内容を補足する情報」と捉えることもできるので，Webからの知識抽出をするときにデータソースとして使われることも多い
- はてなブックマークAPI
  - http://developer.hatena.ne.jp/ja/documents/bookmark/apis/getinfo
  - URLをパラメータとして渡したら，それに対してつけられたブックマーク情報（ユーザ，コメント，タグ）を返してくれるAPI

[課題3] はてなブックマークAPIを使おう

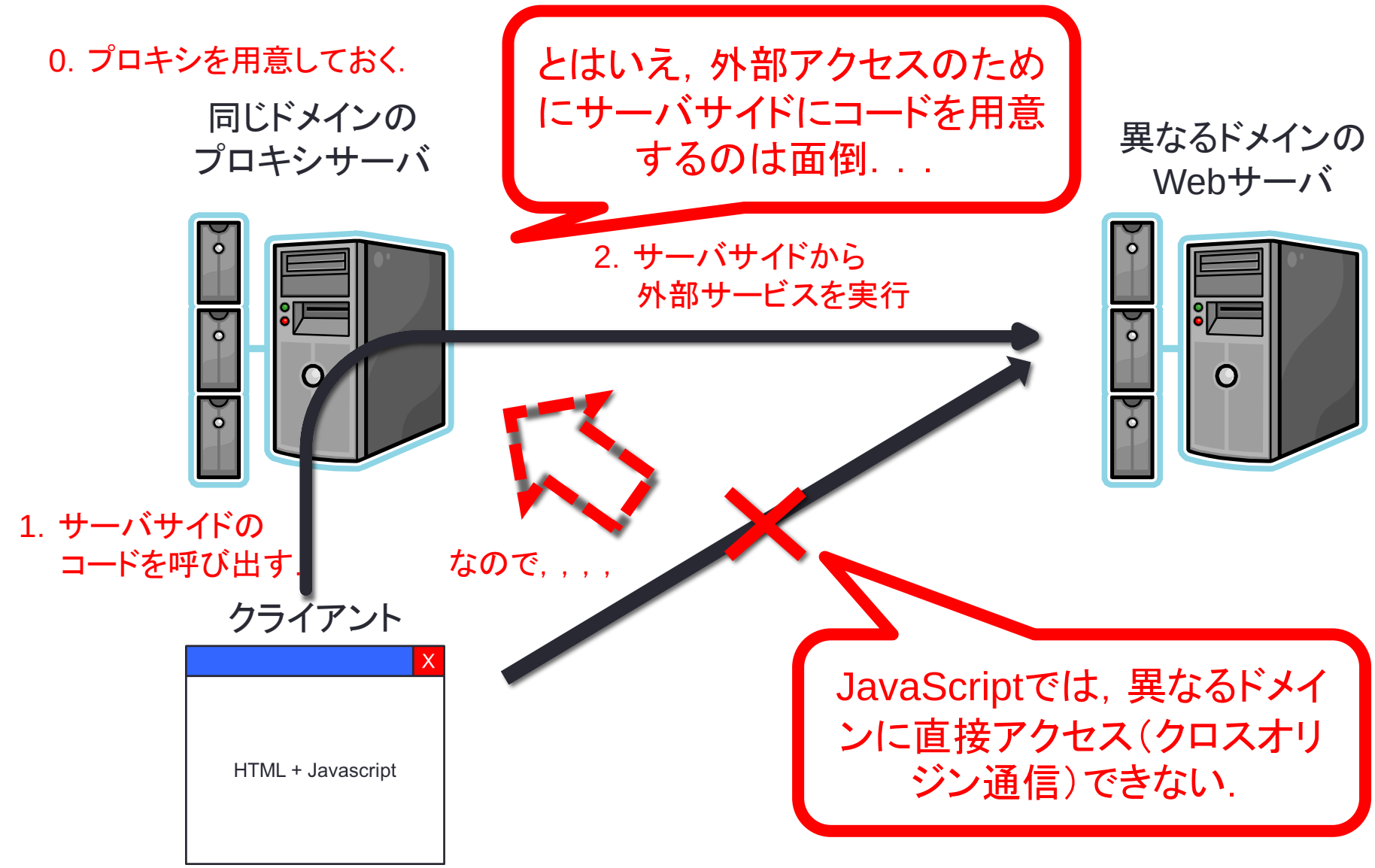
- Webブラウザで，以下のURLにアクセスしよう
  - http://b.hatena.ne.jp/entry/jsonlite/?url=http://www.wings.msn.to
- よくわからないものが表示されるけど落ち着いて観察しよう
  - 以下のような構造のJSONが返ってきているはず（順不同）（ブラウザFirefoxだと見やすいと思う．下図）
  - よくわからないのはURLエンコードされた日本語部分

```
{
        “url”: ...,
        “count”: ...,
        “screenshot”: ...,
        “title”: ...,
            :
        “bookmarks”: [{
                “timestamp”: ...,
                “tags”: [...]
                “comment”: ...,
                “user”: ...,
        },{...},...],
}
```

JSONPによる外部サーバからのデータ取得

- JavaScriptでは，安全性のためにクロスオリジン通信ができないようになっている（次頁図参照）
  - クロスオリジン通信
    - 現在接続中のサーバと異なるサーバとの通信
    - [課題2]で失敗していたのは，これが原因
- 外部のサーバのWebAPIからデータを取得するには，この制限を乗り越える必要がある
  - はてなブックマークAPIも当然外部サーバなので，この制限に引っかかる
  - <script>タグのsrc属性は，この制限に引っかからない
  - 動的にこのタグ経由で通信して，結果を処理する
  - これをやるのがJSON with Padding（JSONP）（次々頁図参照）

クロスオリジン通信

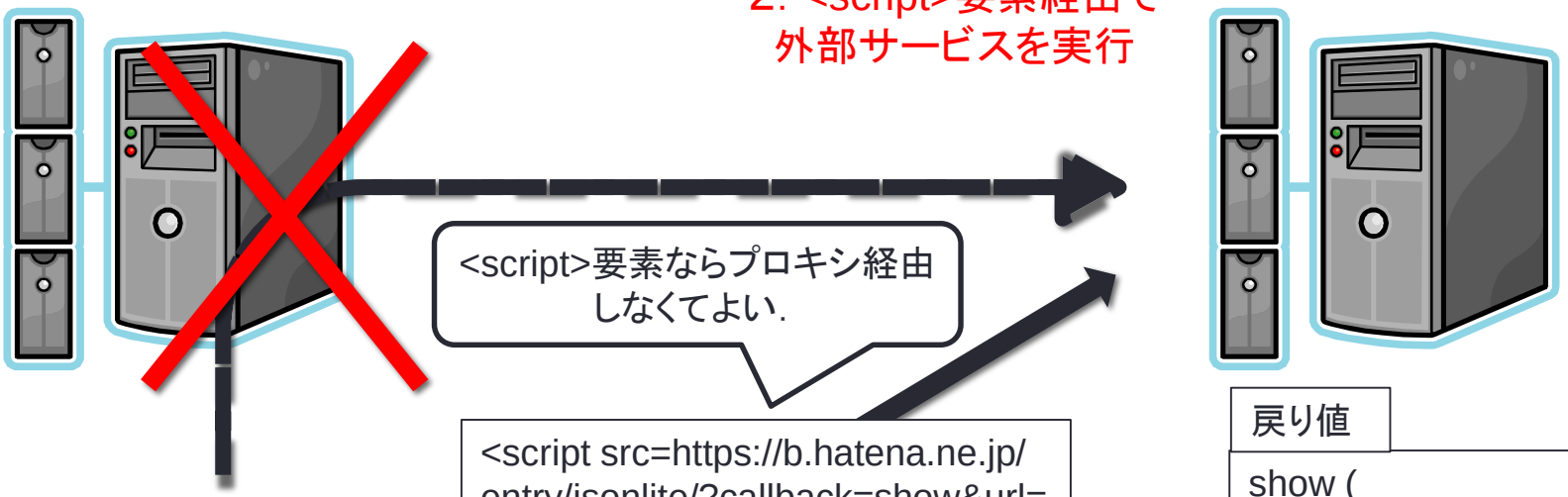

0. プロキシを用意しておく.
同じドメインの
プロキシサーバ
とはいえ，外部アクセスのため
にサーバサイドにコードを用意
するのは面倒. . .
異なるドメインの
Webサーバ
2. サーバサイドから
外部サービスを実行
1. サーバサイドの
コードを呼び出す.
なので，，，，
クライアント
X
HTML + Javascript
JavaScriptでは，異なるドメイ
ンに直接アクセス（クロスオリ
ジン通信）できない.

JSONP

同じドメインの
プロキシサーバ（不要）

異なるドメインの
Webサーバ

2．<script>要素経由で
外部サービスを実行



<script>要素ならプロキシ経由
しなくてよい．

クライアント

HTML + Javascript

```
<script src=https://b.hatena.ne.jp/
entry/jsonlite/?callback=show&url=
http://www.wings.msn.to/></script>
```

戻り値

```
show (
 {“count”:”100”,
  “bookmark”: […]
  }
)
```

戻り値は「コールバック関数名（JSONデータ）」の
JavaScriptコード

```
function show (data) {
  結果を処理するコード
}
```

1．イベント発生時に，外部からスクリプトを
ダウンロードする<script>要素を作成

3．戻り値（関数呼び出しコード）によって
クライアント側の関数を呼び出す．

JSONPをやってみる その1

- JSONPのためにはcallback関数（通信結果を受け取る関数）を作り，そこで結果を処理する
- まずはイベントリスナーから

```
document.addEventListener('DOMContentLoaded', function () {
    let result = document.getElementById('result');

    document.getElementById('btn').addEventListener('click', function () {
        let target_url = document.getElementById('url').value; //HTML上で入力されたURLを受け取る
        let url = 'http://b.hatena.ne.jp/entry/jsonlite/?callback=show&url=' + encodeURIComponent(target_url);
        //callbackパラメータには，結果を処理する関数を指定する
        //urlパラメータには，ブックマーク情報を受け取りたいURLを指定する
        //urlはencodeURIComponentを使って，getのパラメータとして適切な文字になるように変換する（変な記号とかを変換）
        let scr = document.createElement('script');
        scr.src = url; //ややこしいけど，scriptタグなので，scrって変数名，sorceを指定するので，src属性です．
        document.getElementsByTagName('body').item(0).appendChild(scr);
    }, false);
}, false);
```

今回はshow()がcallback関数として指定

- ボタンがクリックされたら，WebAPI用のURLと，それを読み込むscriptタグを作っている
- WebAPI用のURLにcallbackという処理する関数を渡している

[課題4] 任意の形式で表示しよう

- はてなブックマークAPIの結果において，タグを集計して表示したい．すでにある「課題4のためのコード」を参考に作成せよ.

課題4-1：まずは，取得したタグを表示するcallback関数 showTagsを作成せよ

- これは，nameとcommen同様にただ列挙するだけでよい
- 作成したら，callback関数をこちらに切り替えよう(show -> showTags)
- 注：課題3で見たように，空タグも多いので<li>だけ表示される行も多い.

課題4-2：次に，取得したタグの出現数を集計して表示する callback関数 countTagsを作成せよ

- タグ名:出現数 の形式で出力・表示せよ
- 可能であれば，出現数の降順になるようにしたい
- 作成したら，callback関数をこちらに切り替えよう(showTags -> countTags)
- ヒント：連想配列（Mapオブジェクト）を利用するとよい.

- かなり難解だと思うので，自信のない人は何がわからなかったのかを整理しておくこと.

成果物の提出（〆切：11/12中）

- 作成および編集した以下のファイルを提出せよ
  - WebAPI_1.js
  - 今回は課題2と4だけでよいです．1と3は任意.
  - 課題2に関しては，失敗した方のURLとどのようなエラーかを報告してください．WebAPI_1.jsの中にコメントアウト形式で書き込んでもらってもよいですし，WebAPI_1.jsとは別ファイルにして提出してもらってもよいです.
- 提出はCoursePowerで行うこと

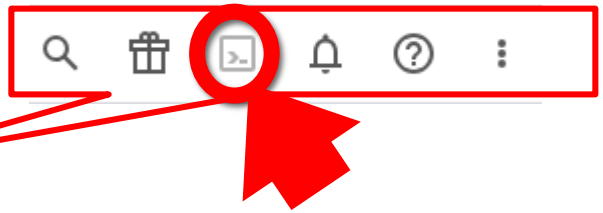
付録
GCPの動かし方

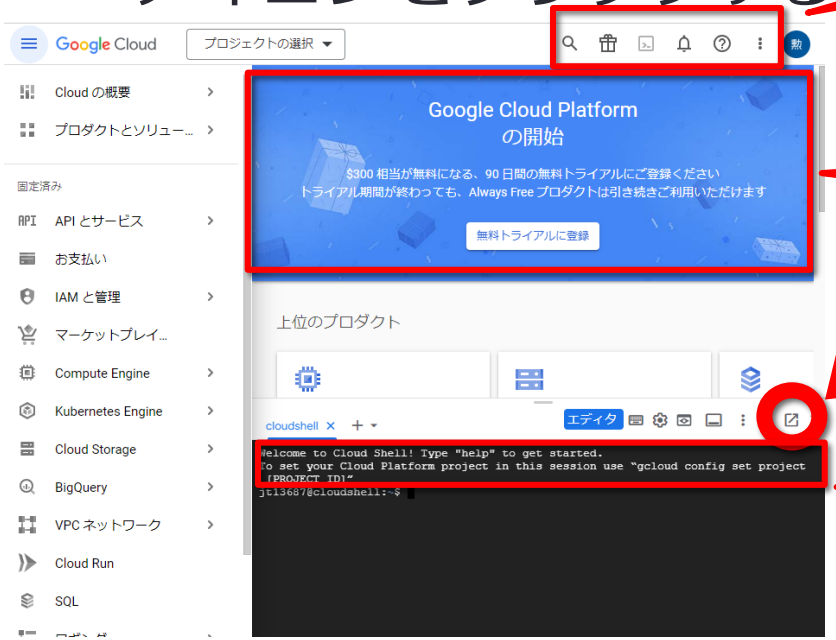
第1回資料の再掲

[課題1] GCPの利用

- プログラムの実行環境としては，Google Cloud Pratform（GCP）を利用する.

1. 以下に各自のアカウント(j3*****@g.kogakuin.jpなど)でアクセスする
   - https://console.cloud.google.com/?hl=ja
2. 右上のCloud Shellのアイコンをクリックする





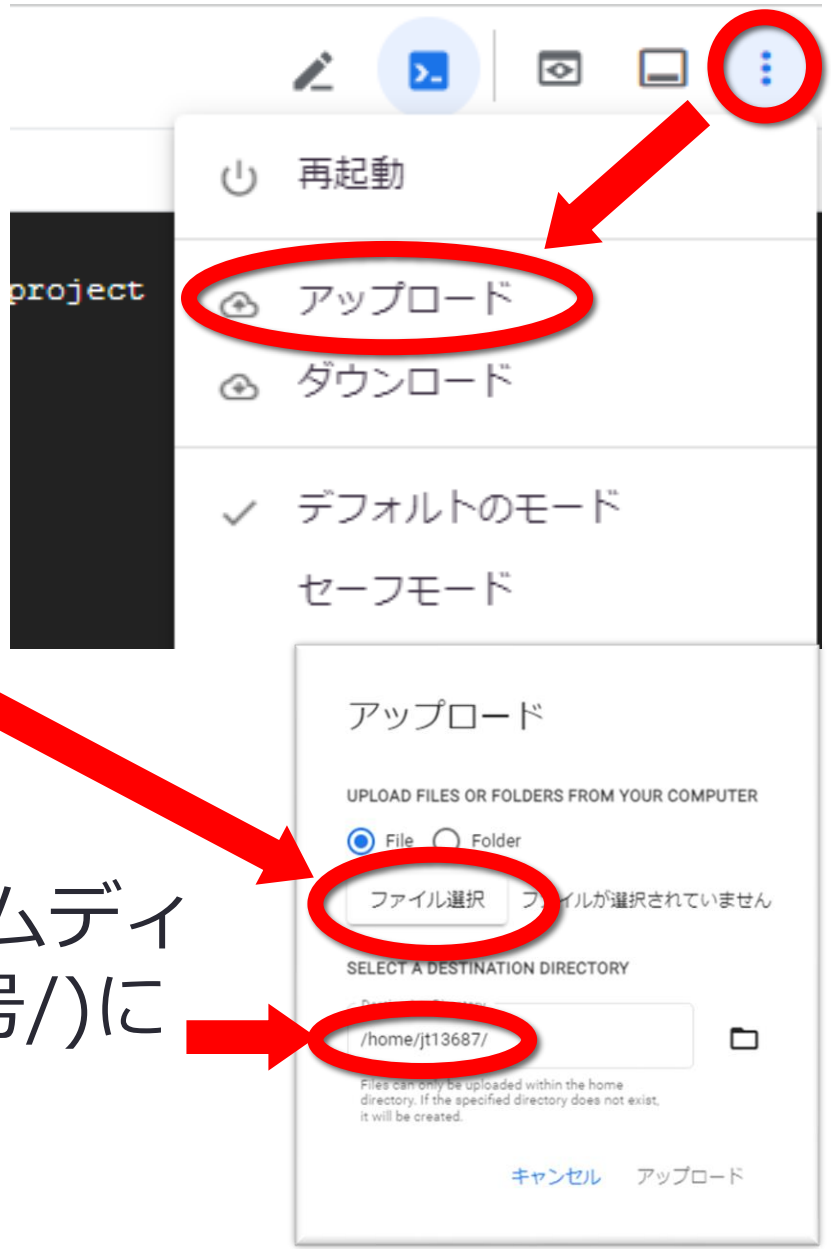
これは気にしない

3. 「新しいウインドウで開く」をクリックする
4. これで準備完了.

この文章も気にしない

GCPへのファイルアップロード

- GCP Cloud Shellの画面のメニューにある「ファイルをアップロード 」を選択
- ファイル選択のダイアログが出現するので，アップロードしたいファイルを1つ選択する
- ファイルは，GCP上のホームディレクトリ（/home/学籍番号/)にアップロードされる

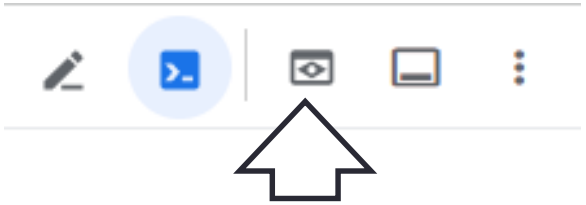


[課題3] PythonでWebサーバを立ち上げよう

- ~~webServer.pyとtest.htmlをアップロードせよ~~
  - Cloudshellでlsコマンドを打てばできているかどうかわかる.
- GCP上で exercise1 のフォルダを作成せよ
  - mkdir exercise1 で作成できる
- exercise1のフォルダに先程のファイルを移動せよ
  - mv [移動させたいファイル名] [移動先フォルダ名]
- 現在地を exercise1 にせよ
  - cd exercise1 で移動できる
  - 現在地が ~/exercise1 になったことを確認せよ
- webServer.pyを実行せよ
  - python3 webServer.py で実行できる

[課題3]つづき

- サーバ上のWebページが見れることを確認せよ



  - GCP Cloud Shellの右上「ウェブでプレビュー」をクリック
  - 「ポート8080でプレビュー」をクリック

  - 右のリストが見えるので，test.htmlをクリック

  - 下記のような画面が見えれば成功