クレジット：
UTokyo Online Education Education コンピュータシステム概論 2018 小林克志

コンピュータシステム概論
第11回

小林克志

- 事務連絡
- 先週の課題、レビュー（振り返り）
- まくら
- ミニプロジェクトに向けて
- 演習:Flask サンプルを動かす
- HTTP, URI / URL DOM (Document Object Model)
- JavaScript
- 課題: Flask に対する機能追加

課題：https://www.kantei.go.jp のサーバ証明書

- 表記サイトのサーバ証明書、ルート証明局からのトラストチェインについて説明せよ。
- 日本政府は公的個人認証サービス（JPKI）を運営、ルート証明局も稼働している。しかしながら、官邸の Web サイトではこの証明局は利用されていない。官邸が JPKI を利用しない理由を考察せよ。

- 講義 Web より回答すること。

ミニプロジェクト

- 目標：Web サービスをつくる
  - 前半で作った可視化をもとに、対話インターフェースつき Web サービスをつくる
  - 他のサービスでも良いが、対話インターフェースをもつこと
- 評価：
  - 関連ファイルを GitHub にアップロード
  - 最終日、5 分 / 人でプレゼン or アップロード
    - プログラムの内容 / 工夫したところ / 苦労した点 / やりたりなかったこと
    - Jupyter-notebook 形式で作成すること
- 道具：
  - HTTP
  - HTML
    - DOM
  - JavaScript
  - Python
  - Jupyter Notebook
  - Matplotlib, Numpy, Pandas
  - Flask(Web フレームワーク)

演習:Flask サンプルを動かす
Git レポジトリのコピー

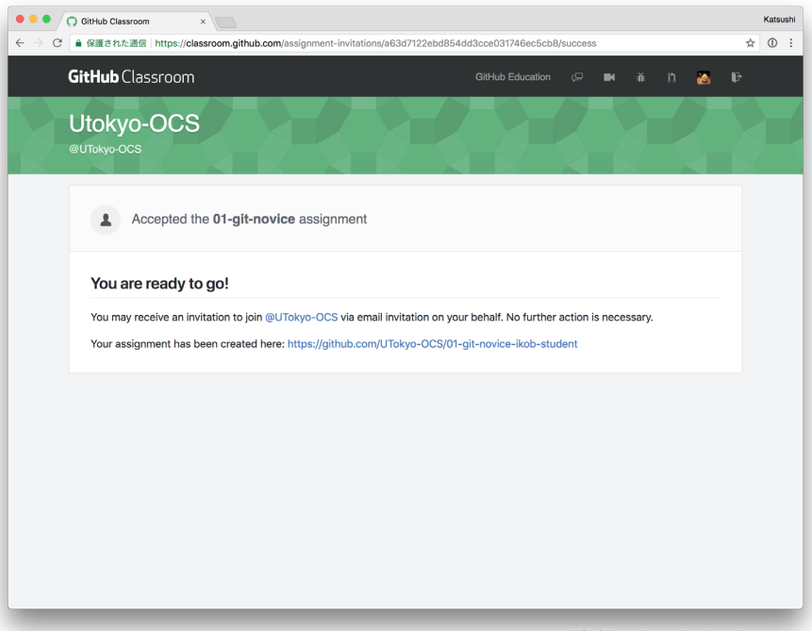
1.講義ページのお知らせページ
『第11回の課題でアクセスする GitHub Classroom の URL』のリンクをクリック

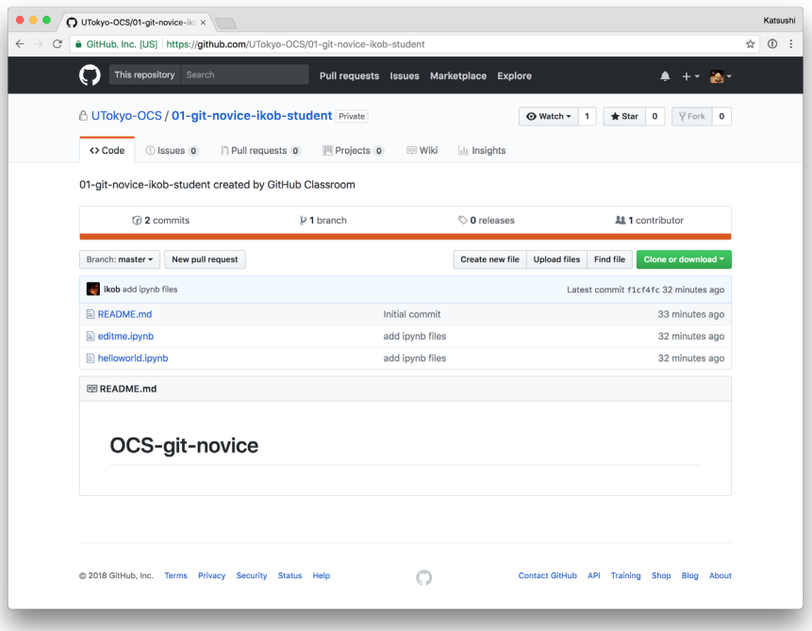
2.課題を Accept するかどうか聞かれるので、Accept する（右上）

3.“Your assignment has been created here: ”以降にアクセスしてみる

4.自身の GitHub レポジトリにアクセスできる。（右下）

5.メールが飛ぶので Accept するようにしてください。





演習:Flask サンプルを動かす
URL 取得と git clone

1.GitHub Classroom 登録で作られた GitHub レポジトリページに移動

2.“Clone or download” ボタンをクリック、URL を表示

3.URL をコピーする(右端のコピーボタンをクリック)

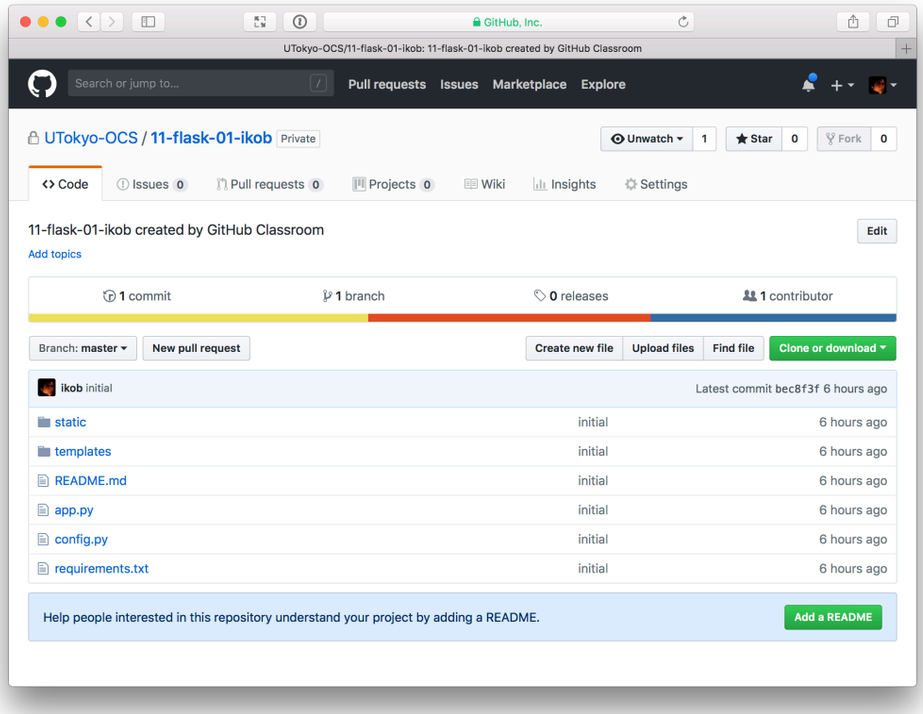
4.CLI で以下の git clone コマンドを実行する。途中で GitHub のユーザ名・パスワードを聞かれる:

```
$ git clone <コピーした URL をここにペーストする>
Cloning into '11-flask-01-ユーザ名'...
remote: Counting objects: 9, done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 9 (delta 0), reused 9 (delta 0), pack-reused 0
$
```

5.ディレクトリ”11-flask-01-ユーザ名”(以降作業ディレクトリ)が確認できれば複製は成功。

6.作業ディレクトリに移動、状態・ログを表示してみる

```
$ cd 11-flask-01-ユーザ名
$ git status
On branch master
Your branch is up to date with ‘origin/master’.
$ git log
コミットログの表示
```

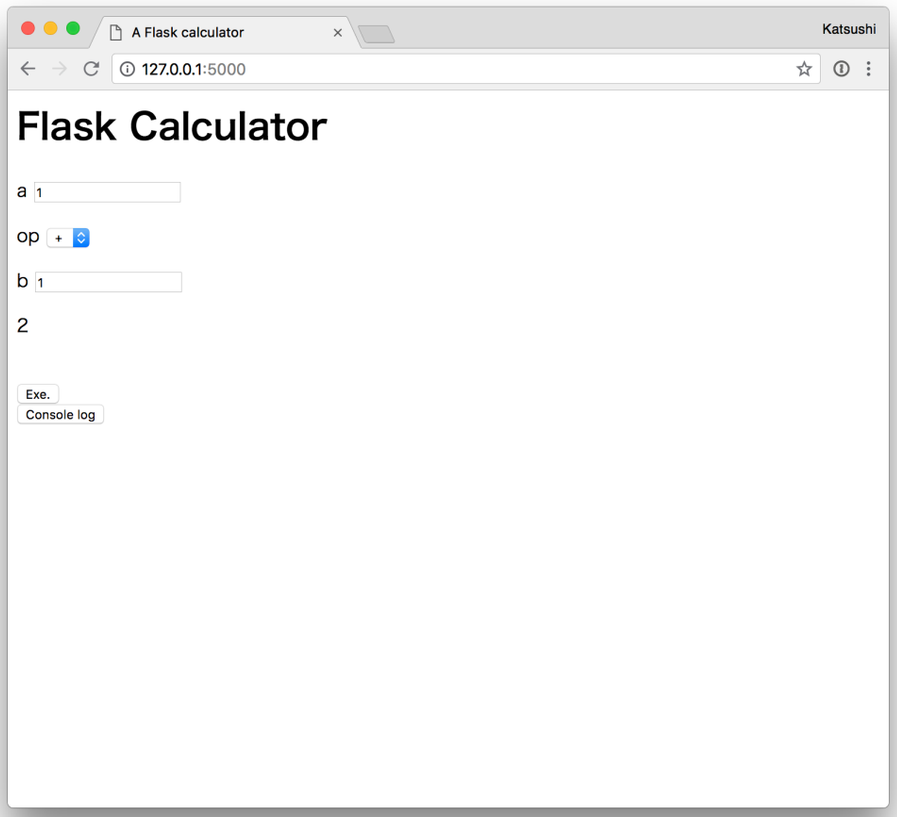


演習:Flask サンプルを動かす
Flask を起動する

1.python インタプリタから app.py を起動する

```
$ python app.py
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 634-123-011
$
```

2.GoogleChrome から http://127.0.0.1:5000 にアクセス、動作を確認する。



演習:Flask サンプルを動かす
レポジトリの構成

```
11-flask-01-ikob/
├──── README.md
├──── app.py
├──── config.py
├──── requirements.txt
├──── static
│     └──── calculator.js
└──── templates
      └──── index.html
```

- app.py : Python プログラム本体
- config.py : 設定ファイル
- static : 静的コンテンツ、
  - calculator.js : JavaScript プログラム
- templates : テンプレート
  - index.html : HTML テンプレート

Flask

- Python 向けマイクロ Web フレームワーク
  - werkzeug, jinja2, good intentions を活用
- 概観

1.デコレータによる URL ルーティングと関数定義で Web サービスを構築できる

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello, World!'
```

2.URL の一部を変数として関数に渡せるため、REST (Representational State Transfer (REST))との相性が良い

```
@app.route('/user/<username>')
def show_user_profile(username):
    # show the user profile for that user
    return 'User %s' % username
```

3.HTTP メソッドも URL ルーティングで対応

```
from flask import request

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        return do_the_login()
    else:
        return show_the_login_form()
```

4.静的ファイル、テンプレートへの対応

5.クライアントからサーバに送られるリクエストデータへのアクセスが容易

6.Cookie, ユーザセッションの取り扱い

- 参考文献

Robert Picard, "Explore Flask Documentation", https://exploreflask.com/en/latest/index.html, 2017
Robert Picard, 濱野司(訳),探検! Python Flask, 達人出版協会

app.py

```
1  #app.py
2  from flask import Flask, request, render_template
3  import urllib
4  import numpy as np
5
6  app = Flask(__name__)
7
8  @app.route("/")
9  def index():
10     return render_template("index.html")
11
12 @app.route("/func/<func>")
13 def exec_calculate(func):
14     if func != "calc" :
15         return "error"
16
17     # Obtain query parameters
18     a = request.args.get("a", default=0, type=int)
19     b = request.args.get("b", default=0, type=int)
20     op = request.args.get("op", "+")
21
22     # Calculator
23     if op == "+" or op == "add":
24         ret = str(a + b)
25     elif op == "-" or op == "sub":
26         ret = str(a - b)
27     elif op == "*" or op == "mul":
28         ret = str(a * b)
29     elif op == "/" or op == "div":
30         if b == 0 :
31             ret = "error(0 div)"
32         else:
33             ret = str(a // b)
34     else:
35             ret = "error(unknown)"
36
37     return ret
38
39 if __name__ == "__main__":
40     app.run(debug=False, port=5000)
```

- 8 – 9行: ルート "/" への　URL ルーティングと処理関数
- 10 行目: template の指定
- 12 – 13行: "/func" への URL ルーティングと処理関数、path の最も右のブロックを変数 func として渡す
- 18 – 20 行: URL の　query ブロックを変数に代入
- 23 – 35 行: 計算機
- 39 – 40 行: アプリケーション実行

int -> float に書き換える

- 2 行目: flask モジュールの読み込み
- 6 行目: Flask オブジェクトの生成

templates/index.html

```
 1 <html>
 2
 3 <head>
 4     <title>A Flask calculator</title>
 5     <script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"></script>
 6 </head>
 7 <body>
 8     <h1>Flask Calculator</h1>
 9     <div>
10         <p>a
11         <input type=number id=freq value=1>
12         </p>
13
14         <p>op
15         <select id=op>
16             <option value="+" selected>+</option>
17             <option value="-">-</option>
18             <option value="*">*</option>
19             <option value="/">/</option>
20         </select>
21         </p>
22
23         <p>b
24         <input type=number id=phase value=1>
25         </p>
26
27         <p id=result>
28             0
29         </p>
30         <br/>
31         <!-- The event handler is defined in the calculator.js. -->
32         <input type=button id=exec value="Exe.">
33     </div>
34     <div>
35         <!-- The event handler is described on an event attribute. -->
36         <input type=button id=log value="Console log" onclick="console.log('Button pushed!')">
37     </div>
38 </body>
39 <script type="text/javascript" src="/static/calculator.js"> </script>
40
41 </html>
```

- 14 – 20 行: 演算子(+-*/)の選択メニュー
- 23 – 25 行: 変数 b の入力フィールド
- 27 – 29 行: 結果出力
- 31 – 32 行: 実行ボタン
- 35 – 36 行: ログ出力ボタン
- 39 行目 : /static/calculator.js スクリプト読み込み

- 3 – 6 行: ヘッダ
- 5 行目: jQuery の読み込み
- 7 – 38 行: ボディ
- 10 – 12 行: 変数 a の入力フィールド

static/calculator.js

```
 1 function execCalculator() {
 2 // Operator Selector
 3     var op_sel = document.getElementById("op");
 4     var idx = op_sel.selectedIndex;
 5     var op = op_sel.options[idx].value;
 6
 7 // Buid query parameter
 8     var param = {};
 9     param["a"] = document.getElementById("freq").value;
10     param["b"] = document.getElementById("phase").value;
11     param["op"] = op;
12     var query = jQuery.param(param);
13
14 // Query with a new parameter
15     $.get("/func/calc" + "?" + query, function(data) {
16         document.getElementById("result").textContent = data;
17     });
18 };
19 // Color changer demo
20 function changeButtonColor(obj) {
21     var color = obj.getAttribute("style");
22     switch(color){
23         case "color:blue":
24             newcolor = "color:red";
25             break;
26         case "color:red":
27             newcolor = "color:blue";
28             break;
29         default:
30             newcolor = "color:red";
31             break;
32     }
33     obj.setAttribute("style", newcolor);
34 };
35 // Register Event handler
36 document.getElementById("exec").addEventListener("click", function(){
37     execCalculator();
38     changeButtonColor(document.getElementById("exec"));
39 }, false);
40 // run once
41 execCalculator();
```

- 3 – 5 行: 演算子を id = op で指定されるセレクタから取得、変数　op に代入
- 8 – 12 行: 変数 a. b を id = a, b からそれぞれ取得、query パラメータ生成
- 15 行目: query パラメータ付きの HTTP GET リクエストを "/func/calc" に送る
- 16 行目 : GET リクエストの結果で id = result のテキストフィールドを書き換える
- 20 – 34 行: obj で渡される要素の style 属性を、赤 -> 青 -> 赤 … と変える。
- 36 行目: id=exec で指定されるボタンクリック時の挙動を設定する（イベントハンドラの登録）

- 1 – 18 行: 計算機

Hyper Text Transfer Protocol (HTTP) RFC7230 他(再掲)

- Web サービスの核となるステートレスなアプリケーションレベルの通信方式
- ８種のメソッドが規定されている、太字がコンテンツ操作:
  - **GET** / HEAD / **POST** / **PUT** / **DELETE** / CONNECT / OPTIONS / TRACE

	Safe / Cacheble	Idempotent	機能 (CRUD)
GET	✔	✔	Read
POST	✖	✖	Create
PUT (PATCH)	✖	✔	Create / Update
DELETE	✖	✔	Delete

URI (Uniform Resource Identifier) / URL (Uniform Resource Locator) RFC 3986

- URI: 抽象的・物理的なリソースを識別する文字列
- URL: URI のサブセット、リソースへのアクセス方法（ネットワーク上の場所）を記述
- URI の構文と例

```
URI         = scheme ":" hier-part [ "?" query ] [ "#" fragment ]

hier-part   = "//" authority path-abempty
            / path-absolute
            / path-rootless
            / path-empty
```

```
  foo://example.com:8042/over/there?name=ferret#nose
   ¥_/   ¥______________/¥_________/ ¥_________/ ¥__/
     |           |            |            |        |
  scheme     authority       path        query   fragment
```

  - scheme: アクセス仕様、e.g., http, https
  - authority: 一般に、サーバアドレス、ポート番号、ユーザ情報
    [userinfo "@"] host [":" port]
  - path: Authority 範囲内の階層的なリソース識別子
  - query : Authority 範囲内の非階層的なリソース識別子、key, value ペアで表現することが多い
    ?key1=value1;key2=value2, ...
  - fragment: リソースサブセットの識別子
- 特殊な文字はエスケープが必要なことに注意

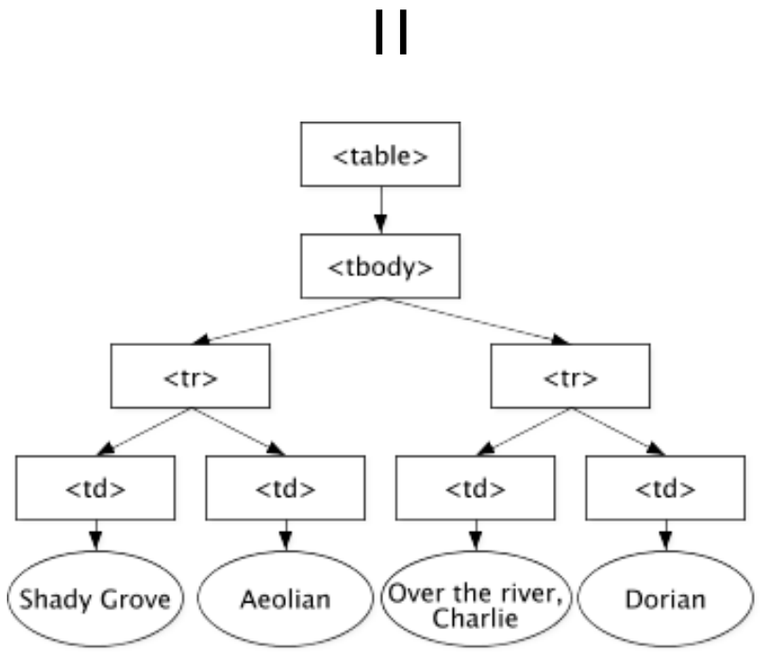
DOM (Document Object Model)

- XML, HTML 文書に対するプログラミングインターフェース。すなわち JavaScript のようなスクリプト言語から文書の構造、スタイル、コンテンツの変更を可能にする。DOM では文書をノードとオブジェクト からなる木構造として表現する。
- プログラミング言語ではない、言語中立。
  API (HTML or XML page) = DOM + scripting language
- 重要なデータ型
  - document: ルート文書オブジェクト
  - element: ドキュメント要素（ノード）
  - attribute: 属性（ノード）
- 重要な DOM インターフェース
  - document.getElementById(id) : 指定された ID を持つ要素を返す（多くの場合この方法で要素を指定する）
  - element.setAttribute(name,value): 要素に name 属性を追加 or 変更する
  - element.getAttribute(name): 要素の name 属性を返す

"Introduction to the DOM",
https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction

```
<table border='1'
summary="'Shady Grove' in cell one row one, Aeolian in cell two row one, 'Over the River, Charlie' in cell one row two, and Dorian in cell two row two">
<tbody>
<tr>
<td>Shady Grove</td>
<td>Aeolian</td>
</tr>

<tr>
<td>Over the River, Charlie</td>
<td>Dorian</td>
</tr>
</tbody>
</table>
```

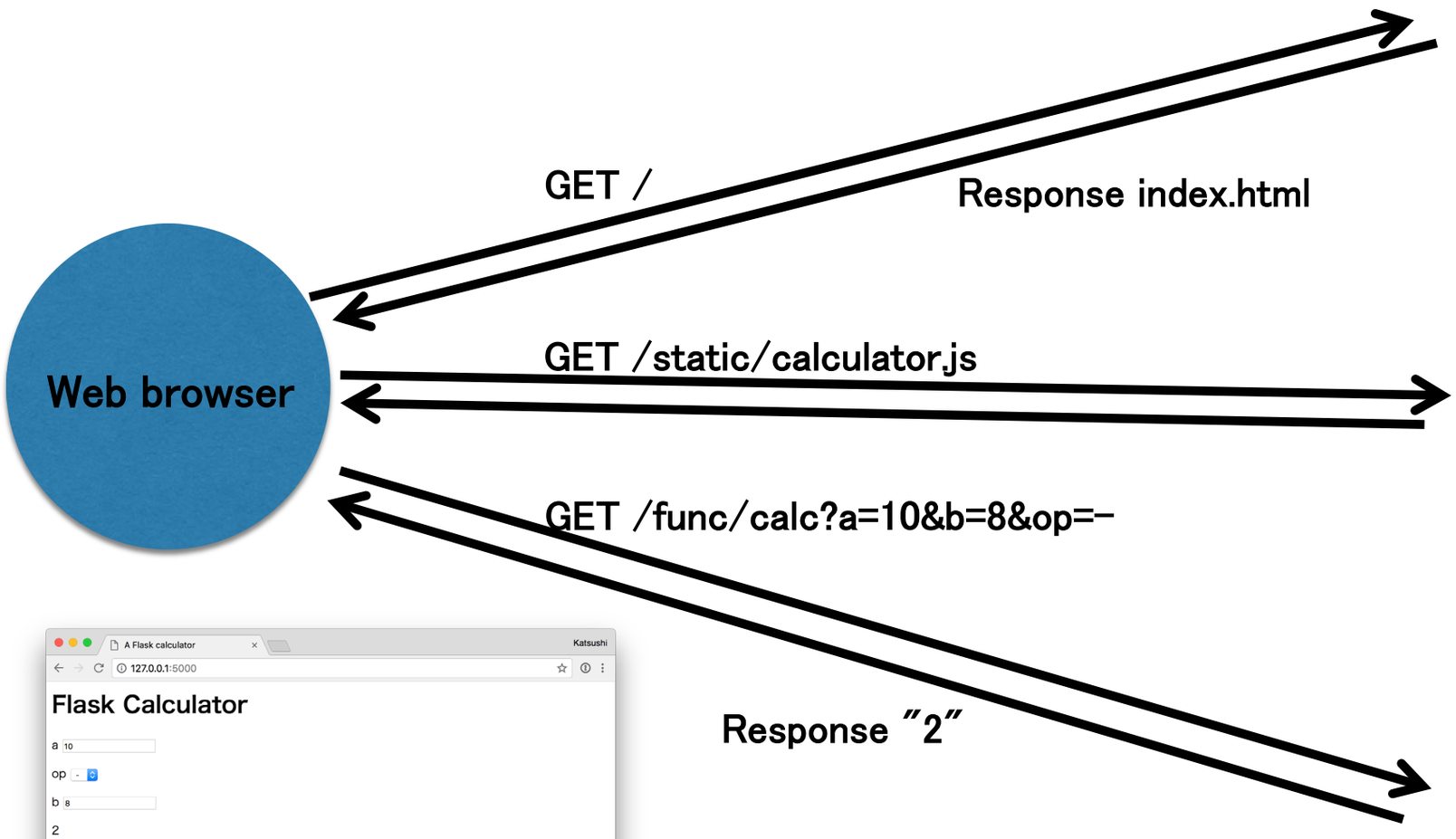


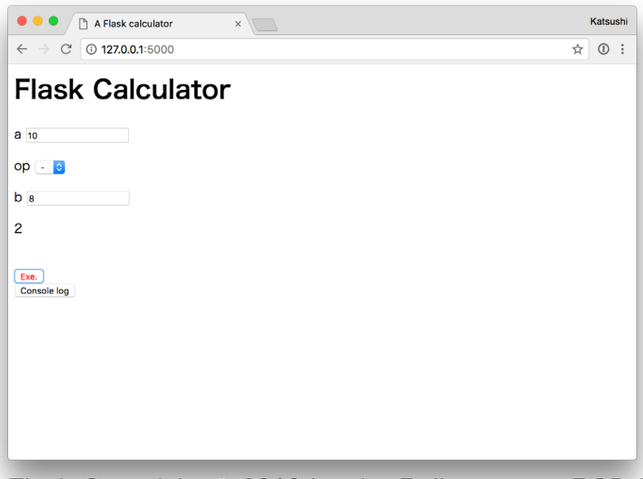
Philippe Le Hégaret, W3C "The W3C Document Object Model (DOM)", W3C, 2002
Copyright ©2002 W3C®(MIT, INRIA, Keiko), All Rights Reserved.
https://www.w3.org/2002/07/26-dom-article.html

JavaScript

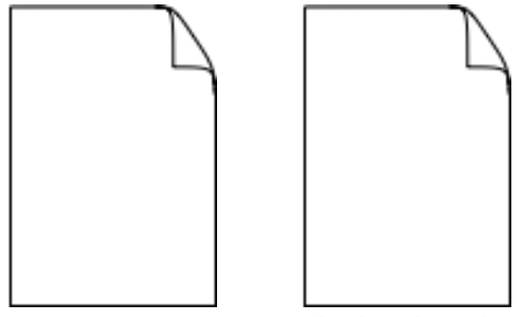
- Web ブラウザのスクリプト言語として普及（事実上の標準）
- 動的・対話的な Web 環境の実現
  - 1995 年 Netscape Navigator (Web ブラウザ)から採用
  - Ajax (Asynchronous JavaScript + XML)
- DOM による HTML / XML ドキュメント要素へのアクセス
  （作成、検索、更新、削除）
- jQuery など拡張ライブラリを介して利用することが一般的
- イベント駆動：多くの場合イベント駆動として実装される。
  - 振る舞いはコールバック関数として記述する
- サーバ側のプログラミング環境として node.js も利用されている。
- Java とは言語仕様は似ているが直接の関係はない。
- 利用方法：
  - HTML に埋め込む
    - ヘッダ、ボディ、
- 便利な関数：
  - console.log("文字列")：標準出力（コンソール）への出力
  - $(document).ready(handler)：DOM(HTML) が全て読み込まれた際に実行する関数（handler）を記述（jQuery）

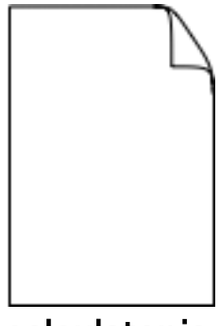
Flask サンプルの構成





Flask Copyright © 2010 by the Pallets team, BSD License





Note: 配布した Flask サンプルはかならずしも index.html はテンプレートである必要はない。
理由は app.py で書き換えないので。

課題: Flask に対する機能追加

- 配布した Flask サンプルをもとに以下を実装する。

1.http://127.0.0.1:5000/static/hello.html でアクセスすると、文字列 "Hellow World" を返すページを作る。

2.配布したオリジナルでは a, b には整数のみを入力できる、これらを小数点以下 1 桁まで対応させる。

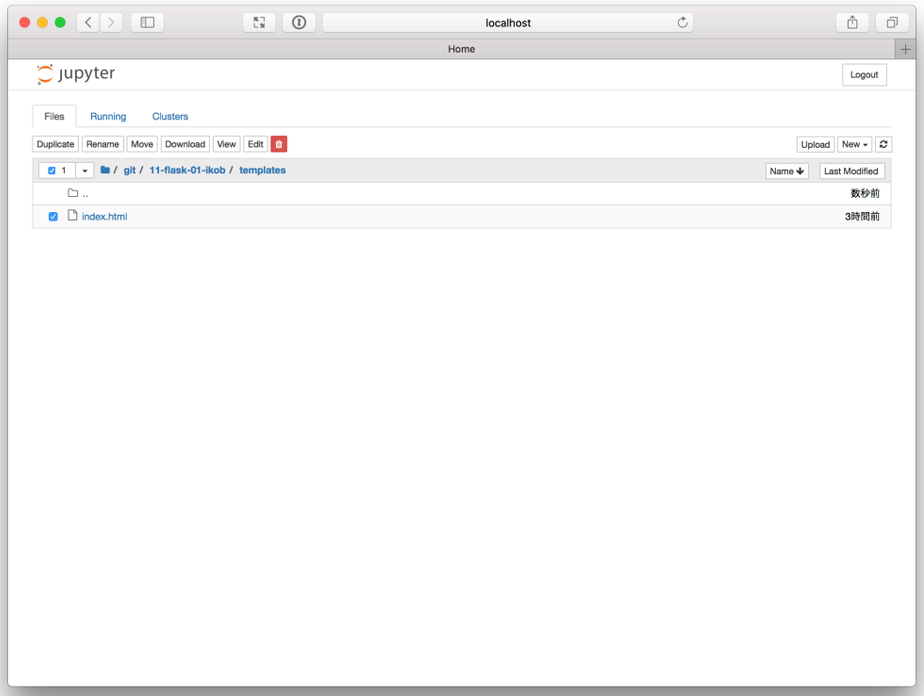
- オリジナルの HTML ファイルは template/index.html。
- HTML input 要素の step 属性を設定すればよい。

3.べき乗 a^b の計算に対応させる。

- JavaScript ファイルは static/calculator.js。HTML, JavaScript プログラム両者を矛盾なく変更する。
- a が負数の場合の処理も検討すること。

4.(選択課題)数値が変更されれば直ちに、すなわち Exec ボタンを押すことなく、結果が返される

Tips



- 編集は jupyter-notebook で可能
  - html ファイルの編集は、チェックボックスを on にし、メニューから Edit を選択
- デバグには Web ブラウザのデベロパーツールを使う。
  - Google Chrome の場合：
    表示 -> 開発/管理 -> デベロパーツール
- Web コンテンツを修正した場合は再読み込みが必要（コンテンツはキャッシュされている）
  - Google Chrome の場合：
    デベロパーツールを開いた状態で、リロードボタンを長押しする