# 情報科学実験II WA
# JavaScript その2

2022/11/26

海谷 治彦

# 目次

- BOM, DOM
- イベント駆動型プログラム
- 対話処理
  - ブラウザ内で完結している対話処理
  - サーバーまでリクエストがいく対話処理
- まとめ

- 演習

# JavaScriptでページを操作

- JavaScript(以降，長いのでJSと略)は単なるプログラム言語という側面より，ウエブページを操作する言語という側面が大きい．
- 本日は，JSからウエブページがどのようにみえているか，どのように操作するかについて，まず紹介する．

- 一言でいうと，JSからは，ブラウザ内の情報が，オブジェクトとして見えていて，それに対して操作すると，実際，ブラウザ内の見えや動作が変わる．

# BOM: Browser Object Model

- ブラウザ内にあるオブジェクトの構造に相当.
- JSからは window という名前のオブジェクトとしてアクセス可能.
- 多数の変数と関数を持つが主なものは後述.
- ブラウザの種類によって，上記の変数や関数が異なる場合がある. (所謂，クロスブラウザ問題)

- 前回，スクリプト内に直接定義した変数や関数は実はwindowオブジェクトの属性やメソッドになっている.
- windowオブジェクトの変数や関数は window. を付記しなくてもよい．通常は省略する.

# BOM自体の主な変数・関数

- innerHeight
  - 画面内側の高さ
- innerWidth
  - 画面内側の幅

サンプル
windowOwnProps.html

- onclick
  - 画面がクリックされた際に呼び出される関数を保持する変数.
  - 一般に Callback 関数と呼ばれる.

- alert()
  - 画面に文字列を含む小さいウインドウを表示して警告を利用者に示すことができる.
- open()
  - 新規windowをあける, ブラウザによってはタブか.
- close()
  - windowを閉じる. ブラウザによってはタブか.
  - たいてい, 設定で禁止されている場合が多い.

# BOMの主な変数

- document
  - 現在表示している文書をあらわすオブジェクト
  - document.write 等もこのオブジェクトのメソッド(関数)
  - たぶん，もっともよくお世話になるオブジェクト
- location
  - 現在表示しているURLを表すオブジェクト
- navigator
  - ブラウザの情報
- history
  - ブラウザの表示履歴

サンプル
(document以外の動作テスト)
windowProps.html

# DOM: Document Object Model

- 表示されているページのデータにアクセスするためのオブジェクト.
- 本来，window.document. と書くが，window. は略せるので，document. と書くのが普通.
- ある意味，もっともJSを特徴的に使う際に重要となるデータ構造.

- 構造は無論，個々のページに依存する.

# できること

- ページ中の要素を変更できます.
  - スタイルも含めて変更できます.
  - 要素の中身(innerHTML)が変更できます.
  - 要素の属性も変更できます.
    - <img src="xxx" alt="image"> の src alt 等が属性
    - かなり後の formHiden.html 参照
- ページ中に要素の追加・削除ができます.
- ページ中の要素に対するCallbackを設定できます.
  - 例えば, の要素を押したら, 警告がでるとか.

# DOM操作の基本1 要素の指定

- 特に修正を行うためには，ページ中の要素が指定できないといけません.
- 以下の四通りの方法して指定するのが普通です.

1. idを利用
   - document.getElementById();
2. classを利用
   - document.getElementsByClass();
3. タグ名を利用
   - document.getElementsByTagName();
4. 最上位要素 document.body から階層的に辿る
   - 子要素は children[]という配列として取り出せる.

- それぞれサンプルにて使い方は紹介.

# DOM操作の基本2 要素へのアクセス

- 前述のgetElementなんとか によって指定された要素を読んだり，更新したりできます.
- 特に ByClass と ByTagName は複数の要素を同時にアクセスできます.
- アクセスした要素の中身にアクセスできます.
- 要素の属性にもアクセスできます. (後述)

- accessElements.html

# DOM操作の基本3 追加・削除

- HTMLは構造的(ぶっちゃけ箇条書き)な文書なので，構造に注目して要素を追加削除できます．
- 例題からわかると思いますが，ある要素の子要素は children[] という配列に見えます．
  - 配列に対する操作がそのまま有効です．

- updateElements.html 参照

# DOMの基本4 イベント駆動

- すでに出てきちゃってますが，ブラウザ上でマウスやキーボードを使ったりしたら何か関数を呼び出すことを仕掛けられます．
- これは一般にイベント駆動型プログラムと呼びます．
- マウスを押すとかキーを押すとかをイベントと呼びます．
  - どんなイベントを感知できるかは，事前に決められてます．
- イベントに対応して呼び出される処理(通常，関数)を Callback と呼びます．

# DOMの主なイベント

- onclick クリックする
- onkeypress キーを押す
- onmousemove マウスが動く
- onmouseover マウスが通過する

- onload ページがロードされる
- onplay 再生される

- はっきりいって沢山ある.
- 要素毎にも定義できる.
- domEvent.html

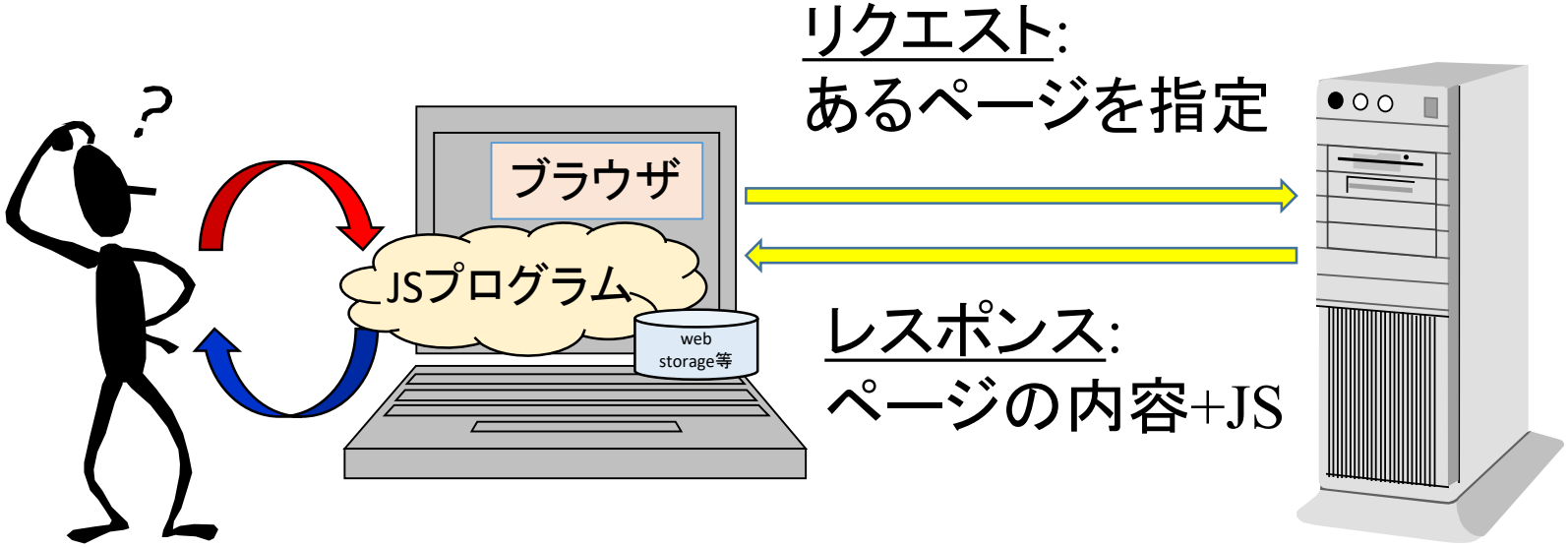
# 対話処理

- ウエブアプリに限らず普通目にするアプリは対話的な物が多い.
  - 一方，プログラムを一方的に動かす「バッチ処理」も技術者にとっては健在．ccコマンド，latex等.
- ウエブアプリでは技術的に以下の二種類の対話がある.

1. クライアント上のJSで対話をする.
   - ブラウザにある資源で可能な計算ならこのほうがリーゾナブル.
2. サーバー上の処理を行ったうえで対話する.
   - サーバー側が保持するデータベース等にアクセスしながらの対話の場合，こちらを行うしかない.

# クライアントのみで対話

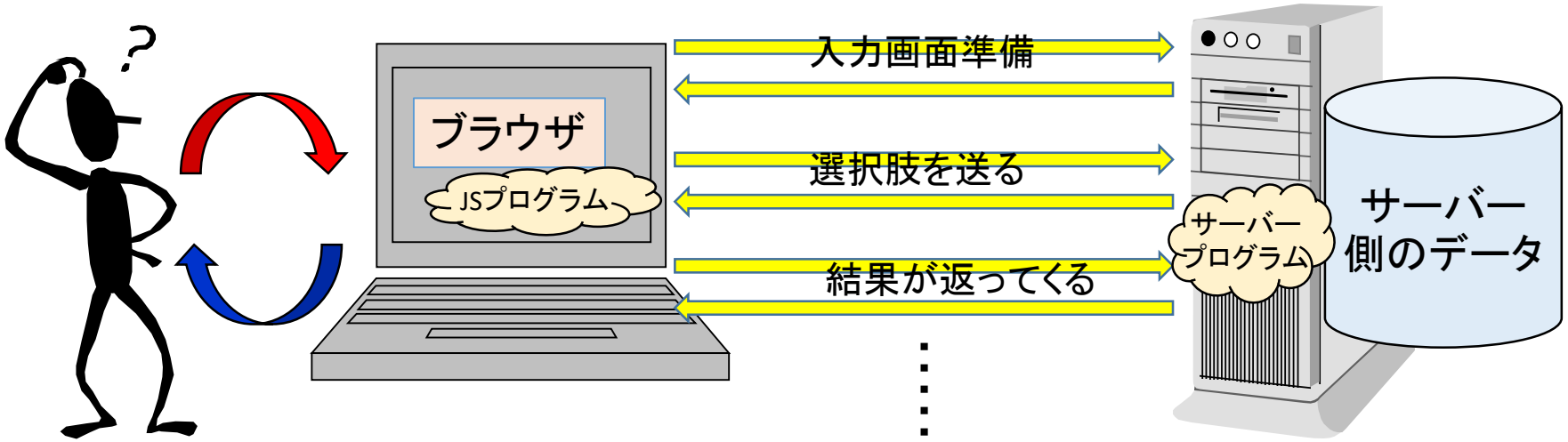
- HTTPリクエスト/レスポンスが一発で済むので通信コストは安い.
- レスポンスで送られてきたデータおよびクライアントにあるデータしか用いることはできない.




※ Ajax等，JSが独自に通信を行うことは当面考えない.

# サーバーとも対話

- HTTPリクエスト/レスポンスは複数回行われる.
  - 無論，JSも使えるが.
- サーバー側のデータが使えるので，例えば，端末を変えても処理が継続できる.
  - 途中まで大学でやって，あとは家のPCでやるとか.



# クライアントのみの対話

- ユーザーの入力手段
  - 前述のイベント駆動を利用
    - HTMLの buttonタグを使うのもあり
  - 入力ウインドウの表示
    - prompt 文字列等のデータを入れる
    - confirm Yes/No選択
  - canvasを用いれば描画情報も入力可能 (参考)
- 出力手段
  - alertによる警告windowの表示
  - HTMLに結果を直に埋め込む document.write 等

# サンプル

- button.html buttonli.html
- prompt.html
  - 後述の formText.htmlと比較
- confirm.html

- canvas.html
  - 参考．よくある落書きプログラム
  - Androidではうまく動かなかった・・・

# サーバーとの対話

- ユーザーの入力手段
  - form というタグを用いて，受け取りページを指定して，自身の入力情報とともにHTTPリクエストを送る.
  - 無論，受け取りページは処理能力がなければいけない. (通常，PHPやServletで書かれる)
- 出力手段
  - 受け取りページに結果が埋め込まれた形で，HTTPレスポンスが返ってくる.
  - その結果をブラウザが表示する.
  - 入力したページとは別のページに移動していることを注意. (ページ遷移と呼ばれる)
    - サーバーもクライアントも一連のページであるという情報を(細工無しでは)もっていない.

# Formの入力タイプ

- 以下に示すような多様？な入力タイプがある.
  - text テキストボックス
  - textarea テキストエリア
  - checkbox チェックボックス
  - radio ラジオボタン
  - select セレクション
  - hidden 隠しフィールド
- 上記はHTML要素として書かれるので，JSのCallbackを仕込むことも当然可能.

# Formの典型様式 (form.html)

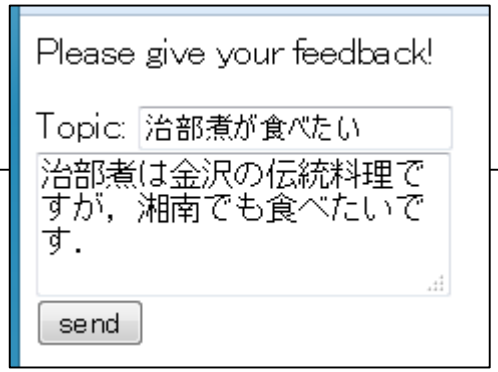
メッセージを受け取るページを指定．このページはリクエスト処理の能力がないといけない．

get もしくは post を指定．違いは後述．

```
<form action="http://www.example.com/form.php" method="post">
Topic:
 <input type="text" name="topic" size="20">
<br>
 <textarea rows="5" cols="22" name="feedback">
 </textarea>
<br>
 <input type="submit" value="send">
</form>
```

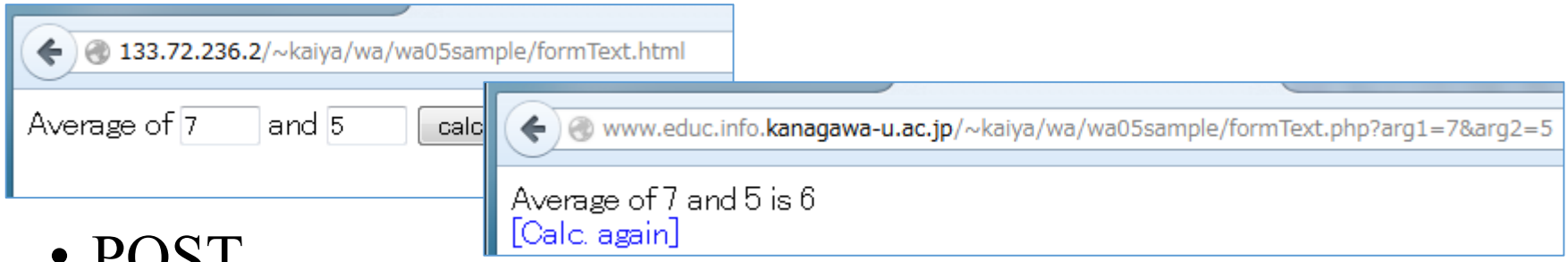
ユーザーの入力値を最終的に受け取りページに送るための呪文．

ユーザー入力場所の指定．詳細は後述．



# GETとPOST

- GET
  - 本来はサーバーからのデータ取得リクエストに用いる.
  - よって，クライアント側から原則データを送ることはできない.
  - しかし，URL自体にデータを付記することで，データを無理やり送ることが可能となっている.
  - 送ったデータはURLを見ただけで丸見え.




- POST
  - サーバーにデータを送る本来のリクエスト.
  - リクエストの本体(ヘッダーより下)に送るデータが付記される.
  - 一応，ぱっと見にはデータは見えないが，暗号化されていないHTTPは簡単に傍受できるので，実際にはデータは丸見え.

# サンプル

- ※ 受け側ページは全て学内でのみ有効.
- formText.html formTextp.html
- formTextarea.html
- formCheckbox.html
- formRadio.html
- formSelect.html
- formHidden.html
  - JSとの連携をしています.

# 参考Androidでも動きます



Canvasの例題は動かなかった

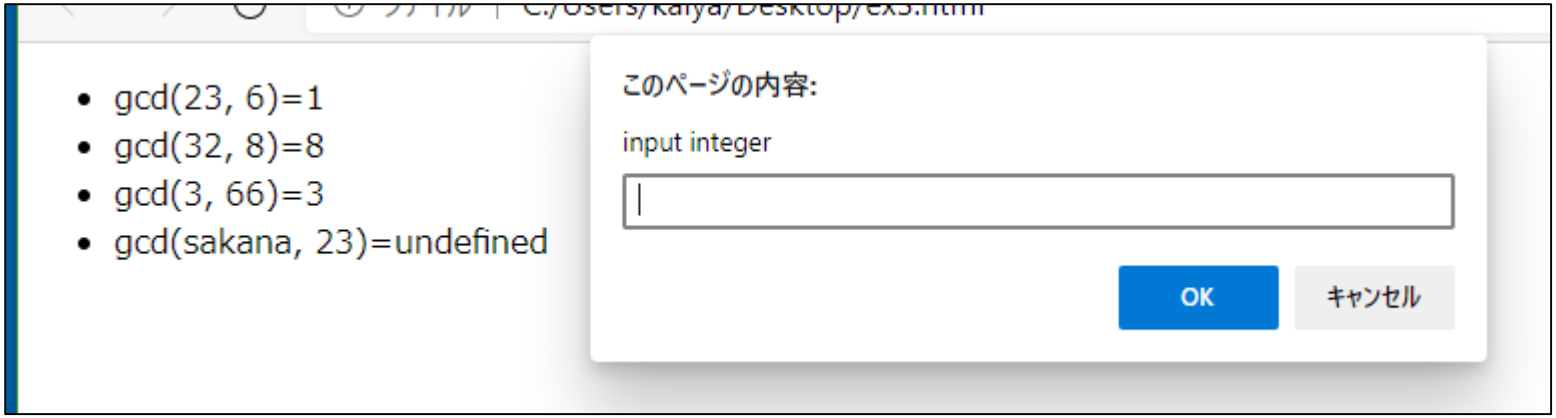
# 本日の演習3 (WAのFinal)

- 次項で指定するJavaScriptのプログラムを作成してください.
  - 複数ファイルになってもかまいません.
- 締切は月曜日とします.
- 提出先はWebClassの WA演習3 です.
  - 3a じゃないですよ.
- ファイル名は ex3.html 必要ならば ex3.js も.
  ex3.zip にまとめてください.

# 問題 (WA演習3)

- 二つの1以上の整数に関する最大公約数 (GCD) を繰り返し求めるJavaScriptのプログラムを作成せよ.
- 最大公約数を求める関数 `function gcd(x,y)` は必ず作成せよ.
  - `x`もしくは`y`が1以上の整数で無い場合, `undefined` の結果を返すものとせよ.
- 二つの整数は利用者がブラウザをクリックするとキーボードから入力できるものとせよ.
  - 入力の際にキャンセルを押した場合の挙動は考慮しなくてもよい.
- 結果の履歴は次のページに示す例のように, 箇条書きとして表示せよ.

# 画面例 (Edge)


gcd(23, 6)=1
gcd(32, 8)=8
gcd(3, 66)=3
gcd(sakana, 23)=undefined
このページの内容:
input integer
OK
キャンセル

# 最終レポート WA演習4

- 問題はwebclassのWA演習4をみてください.
- 原則，1/21にQ/Aを受けます.
- ∠切は1月中でお願いします.

以上