分散システム 第13-14回
― Webアプリケーション ―

大連理工大学・立命館大学 国際情報ソフトウェア学部

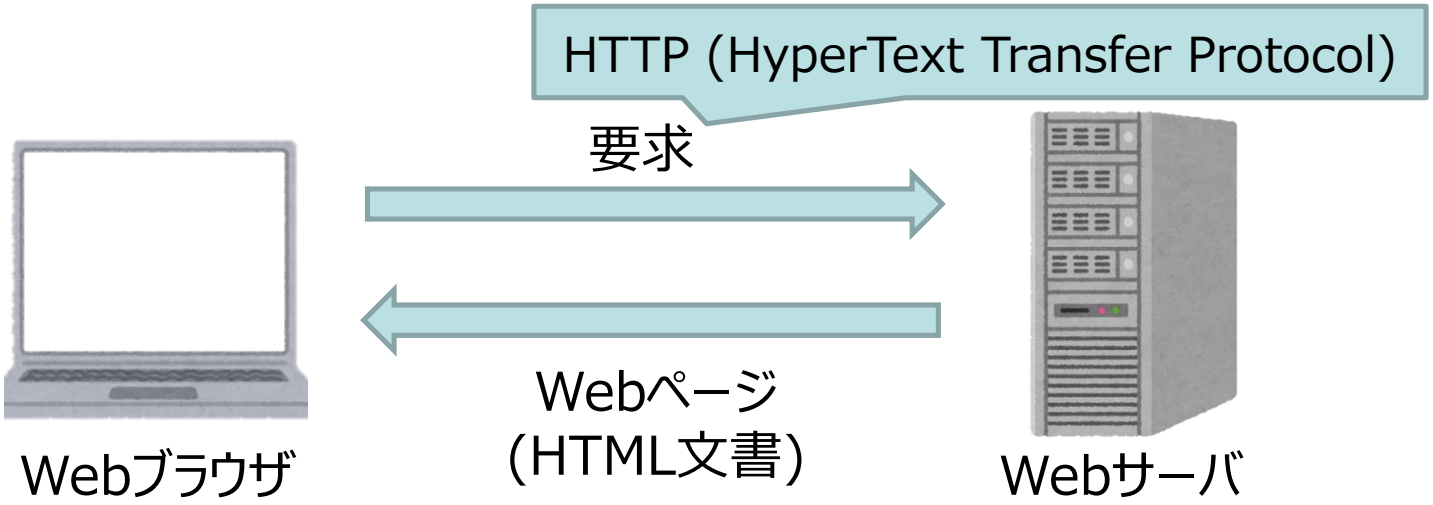
大森 隆行

講義内容

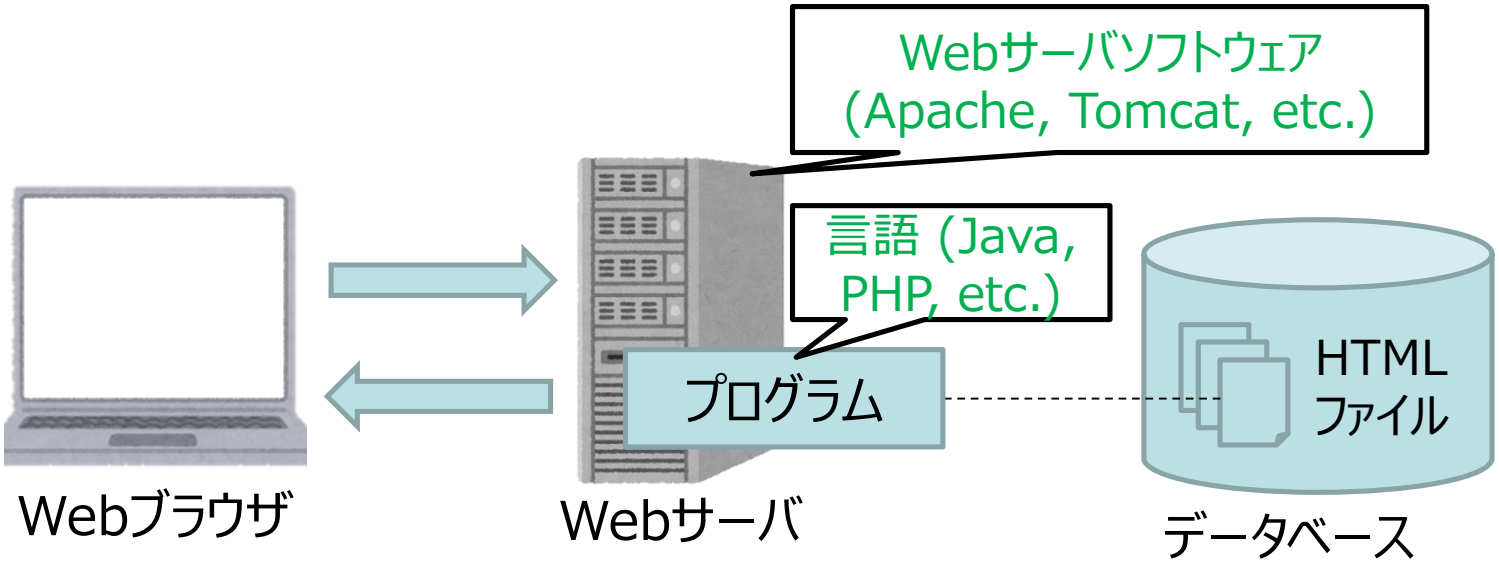
- Webアプリケーション
  - 概説
  - 要素技術
  - セキュリティ

WWW

- Web
  - 文書の公開・閲覧システム
    - ハイパーテキストによって文書を連結
  - → World Wide Webは世界規模のWeb
- 近年ではWeb上で動作するアプリケーション(Webアプリケーション)も多用



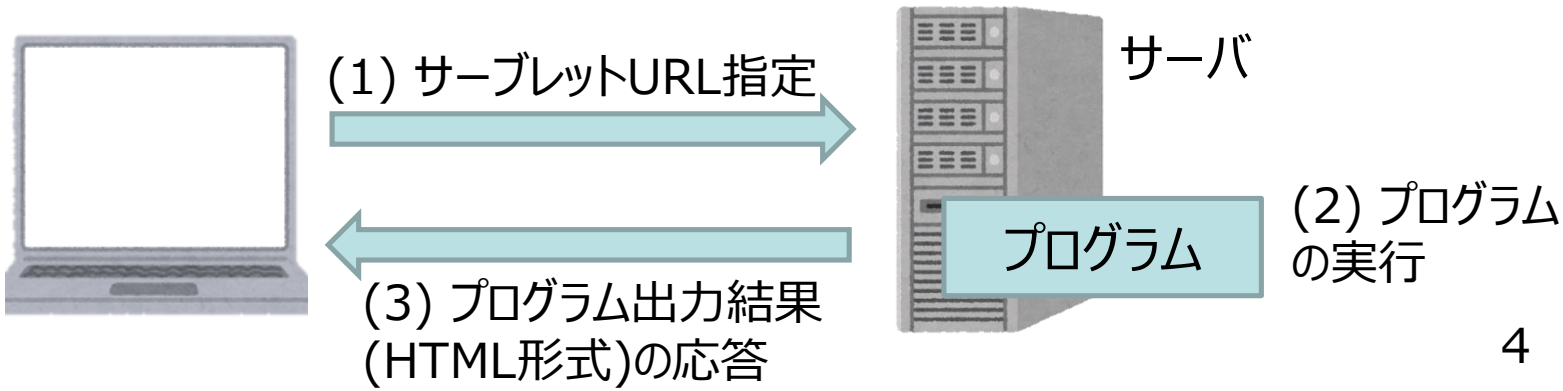
Webアプリケーション



- 静的なWebページ：常に同じ内容
- 動的なWebページ：その都度異なる内容
  - プログラム(Webアプリケーション)により生成
  - サーバ側、ブラウザ側両方で生成可

(例) Javaサーブレットプログラム

```
public void doGet(HttpServletRequest req,
    HttpServletResponse resp) throws IOException {
  resp.setContentType("text/html; charset=utf-8");
  PrintWriter writer = resp.getWriter();
  writer.println("<html>");
  writer.println("<body>");
  writer.println("<h1>Servletのテスト</h1>");
  writer.println("</body>");
  writer.println("</html>");
}
```



(例) Javaサーブレットプログラム

```
public void doGet(HttpServletRequest req,
      HttpServletResponse resp) throws IOException {
  resp.setContentType("text/html; charset=utf-8");
  PrintWriter writer = resp.getWriter();
  writer.println("<html>");
  writer.println("<body>");
  Date currentDate = new Date();
  String currentDateString = currentDate.toString();
  writer.println(currentDateString);
  writer.println("</body>");
  writer.println("</html>");
}
```

動的な内容を埋め込める

Webの設計思想

- RESTful
  - 下記4つの原則から成るシンプルな設計

1. 統一インタフェース
   あらかじめ決められた方法(e.g., HTTP)で情報がやりとりされる
2. アドレス可能性
   すべての情報が一意なURLで示される
3. 接続性
   やりとりされる情報にはリンクを含めることができる
4. ステートレス性
   やりとりは1回ごとに完結し、前の結果に影響を受けない

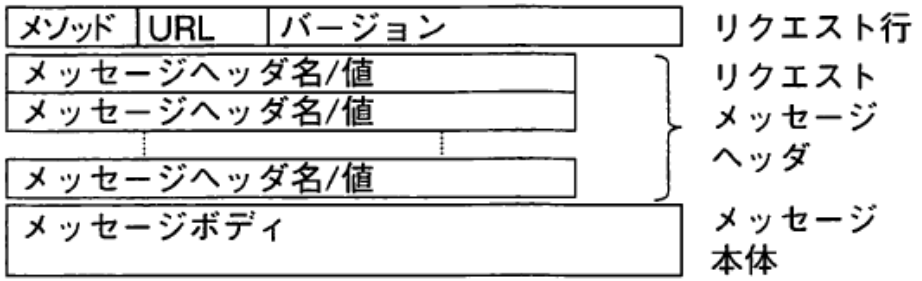
  - 利点
    - シンプルでわかりやすい
    - システムの相互連携も容易

講義内容

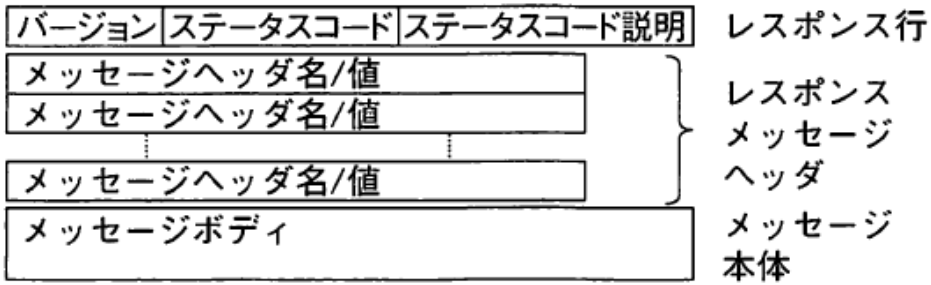
- Webアプリケーション
  - 概説
  - 要素技術
  - セキュリティ

HTTP

- WebブラウザとWebサーバの間のメッセージ交換プロトコル
  - 様々なデータを扱う可能性
  - 用途ごとに基本的なメソッドが存在
- 基本的なメソッド
  - head: ドキュメントのヘッダ(日付やサイズ)を要求
  - get: サーバからドキュメントを取得
  - put: サーバにドキュメントを送信 (ファイル送信等)
  - post: サーバにドキュメントを送信 (パスワード送信等の使用が多い)
  - delete: サーバ上のドキュメント削除を要求



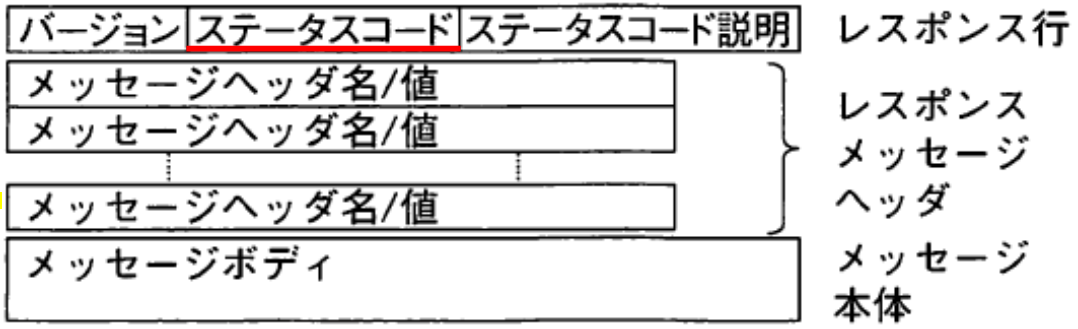
HTTP要求メッセージのフォーマット



HTTP応答メッセージのフォーマット

メッセージヘッダの例：
If-Modified-Since(指定の日付以降の情報を要求), If-None-Match(ドキュメントの内容が一致しない場合のみ実行)

HTTP

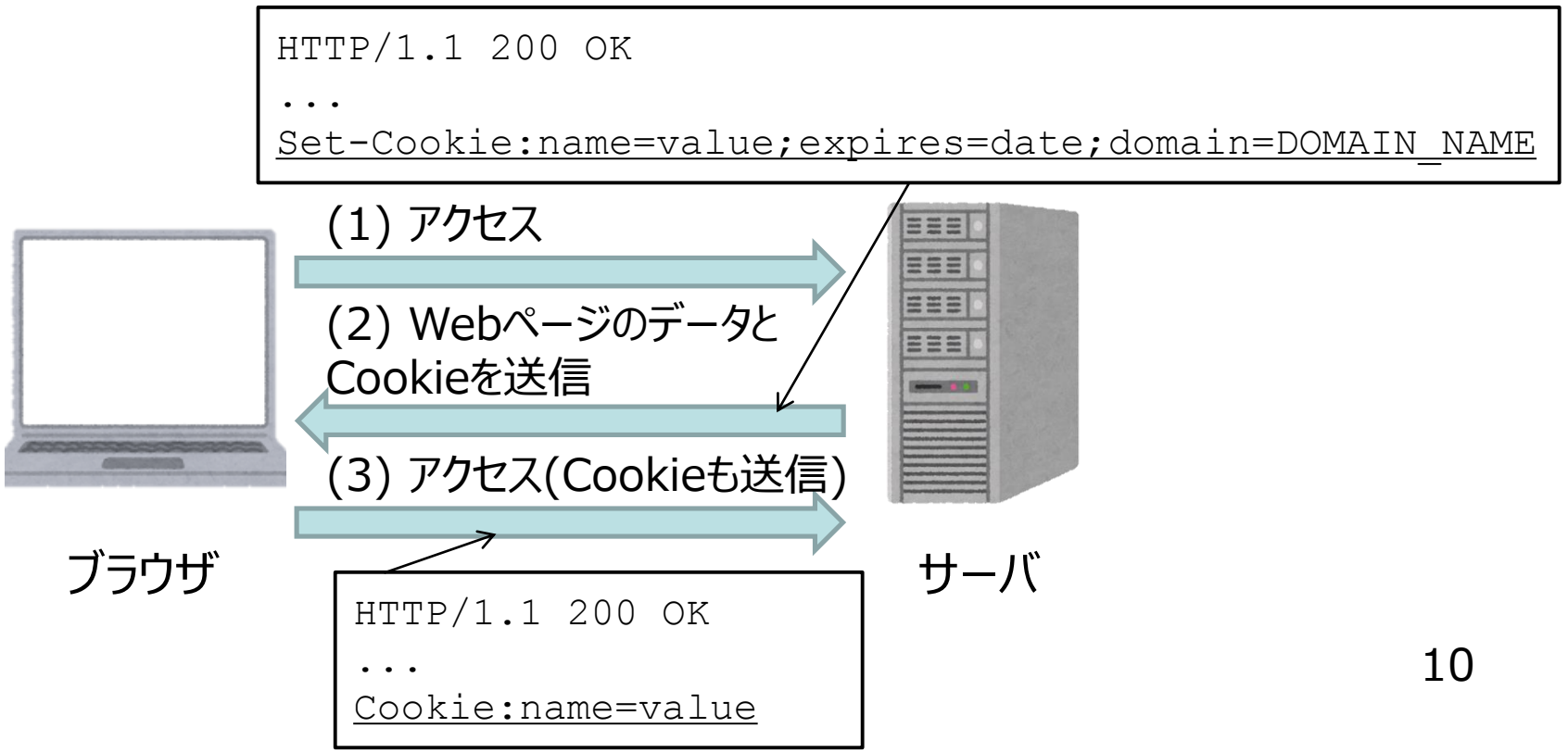


HTTP応答メッセージのフォーマット

- ステータスコード
  - 100番台: 続きの情報がある
  - 200番台: Webサーバが要求を処理できた
    - (例) 200 (OK) 正常に処理が完了した
  - 300番台: 別のURLへ要求を出し直すように要求
    - (例) 301 (Moved Permanently)
      Locationヘッダを使用して別のURLにリダイレクトする
  - 400番台: Webブラウザの要求に問題があり、処理できなかった
    - (例) 403 (Forbidden) アクセス拒否
      404 (Not Found) リソースが存在しない
  - 500番台: Webサーバに問題があり、処理できなかった
    - (例) 500 (Internal Server Error) 内部サーバエラー

Cookie

- HTTPはステートレスな(状態を保持しない)プロトコル
  → セッションを維持するための仕組みがなければ、
  ショッピングサイト等は作れない
- Cookie(クッキー)
  - Webブラウザ側で保持される状態管理のためのデータ



JavaScript

- 動的ページ作成のための代表的なスクリプト言語
  - サーバ側、クライアント側両方で使用
  - HTML文書に埋め込むことも可能だが、
    文書間で共用するため別ファイルにすることが多い
- ECMAScript: 標準化されたJavaScript
  - 元々MozillaやMicrosoftが独自に実装していた
  - Ecmaインターナショナルにより標準化
    - 情報通信システムの分野における国際的な標準化団体
  - 現在、多くのブラウザで処理可能
  - 毎年バージョンアップされている

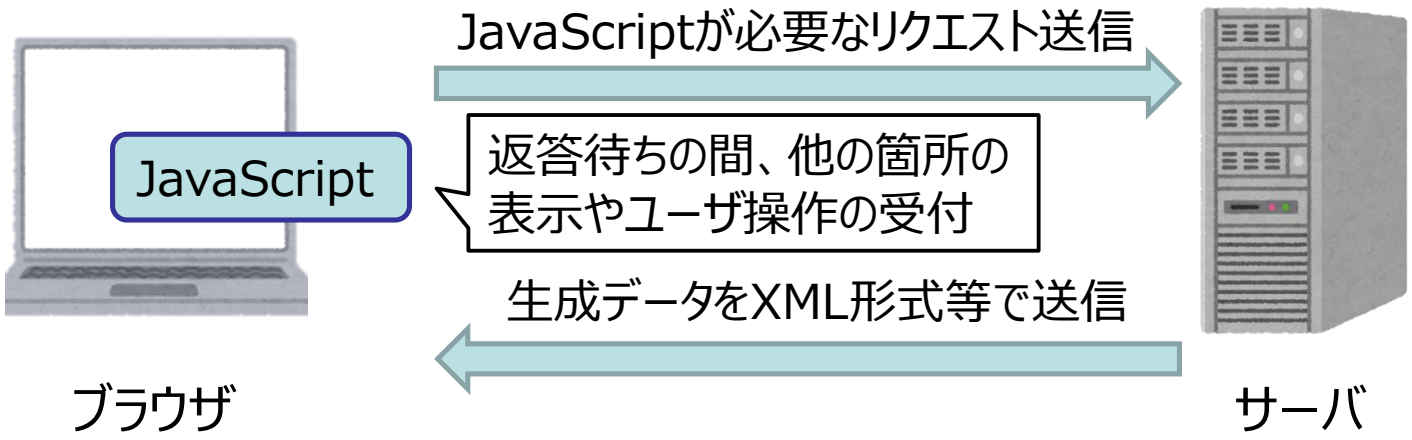
(例) JavaScriptプログラム

ブラウザ上(クライアント側)で実行するプログラム

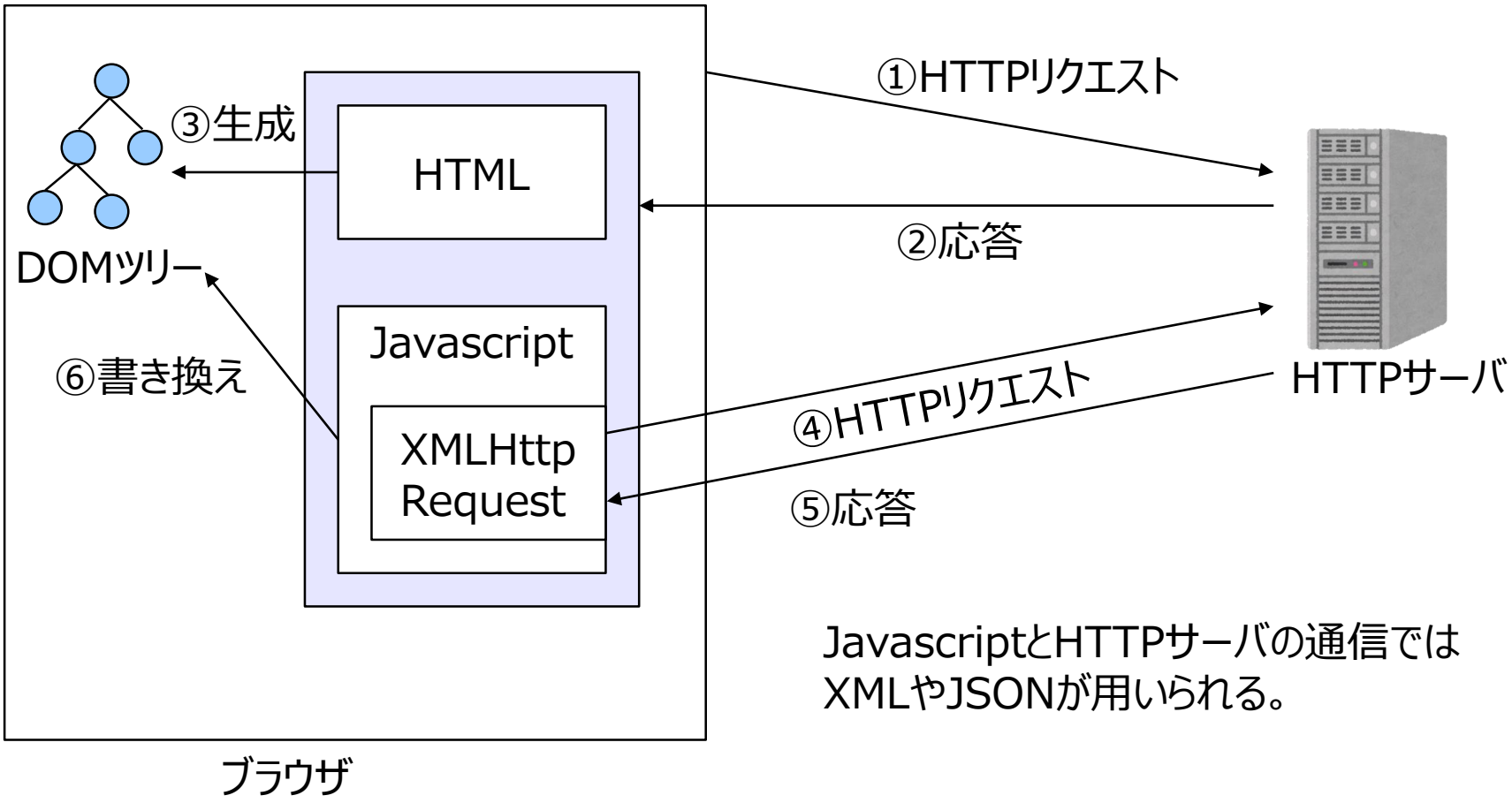
```
<script language="javascript">
  function check() {
    var param = document.form1.text1.value;
    if(param == '') {
      alert("値を入力してください。");
      return false;
    } else if(param.match( /[^0-9]+/ )) {
      alert("数字のみを入力してください。");
      return false;
    }
    return true;
  }
</script>

<input type="submit" value="Send" onclick="return check();">
```

Ajax

- Asynchronous JavaScript + XML
- 従来のWebページはページ単位で
  サーバで生成したものを受信
  → クライアント側では待ち時間が発生
  → 対話性の悪いUI (e.g., 昔の地図サイト)



Ajaxの動作



JavascriptとHTTPサーバの通信では
XMLやJSONが用いられる。

ブラウザ本体からのHTTPリクエストでページ全体（例えば地図を表示する枠など）を取得し，JavascriptからのHTTPリクエストでページ内のデータ（例えば地図の画像データなど）を取得。

Ajaxの要素技術

- ダイナミックHTML
  - HTMLをDOMにより表現し、JavaScriptにより動的書き換えを行う
  - サーバと通信せずにページの内容を変更できる
- XMLHttpRequest
  - HTTPで通信を行うためのライブラリクラス
  - Webページ内に記述されたスクリプトからHTTPリクエストを発する

両方ともJavaScriptのライブラリとして提供されている

(例) ダイナミックHTML

```
<html>
  <head>
    <title>Dynamic HTML Sample</title>
    <script type="text/javascript">
      var number = 0;
      function increment() {
        number = number + 1;
        var numElm = document.getElementById("number");
        numElm.removeChild(numElm.firstChild);
        numElm.appendChild(document.createTextNode(number));
      }
    </script>
  </head>
  <body>
    <div id="sample">
      <center>
        <font size="7"><span id="number">0</span></font><br/>
        <input type="button" value="Increment" onClick="increment();" />
      </center>
    </div>
  </body>
</html>
```

DOMノードの取得

DOMノードの操作

ボタンを押すとincrement関数を呼ぶ

(例) XMLHttpRequest

```
<html lang="ja">
 <head> ...
  <script type="text/javascript">
   const xhr = new XMLHttpRequest();
   window.onload = function(){
    xhr.addEventListener('load', function(){
     if(xhr.status != 200){
      window.alert("Error" + xhr.status + ":" + xhr.statusText);
     }
     const xmlDoc = xhr.responseXML;
     const date = xmlDoc.getElementsByTagName("date")[0].getAttribute("value");
     const weather = xmlDoc.getElementsByTagName("forecast")[0].textContent;
     document.getElementById('elem').textContent = date + " " + weather;
    }, false);
  }
  function buttonClick(){
   xhr.open('GET', 'https://api.xxx.com/tenki/week.php?fmt=xml&city=331');
   xhr.send(null);
  }
  </script>
 </head>
 <body><div id="sample">
  <center>
   <font size="7"><span id="elem">Push Button</span></font><br/>
   <input type="button" value="Get Server Data" onclick="buttonClick()" />
</center></div></body></html>
```

ステータス確認

応答XML文書から情報を取得して表示

XMLHttpRequest送信

JSON (JavaScript Object Notation)

- 構造化データを表すためのデータ記述言語
  - JavaScriptの書式に従っているが、JavaScript専用ではない

(例) JSONで記述した書籍データ

```
[
  "名前":"Webアプリケーションのすべて",
  "発行年":2016
  "著者":[
    "山田太郎","鈴木次郎"
  ]
  "目次":[
    "章":[
      …
    ]
  ]
]
```

- JavaScriptコードとして読み込むことができる
- XMLと比較すると軽量

- XMLと比較するとタグがない分読みにくい

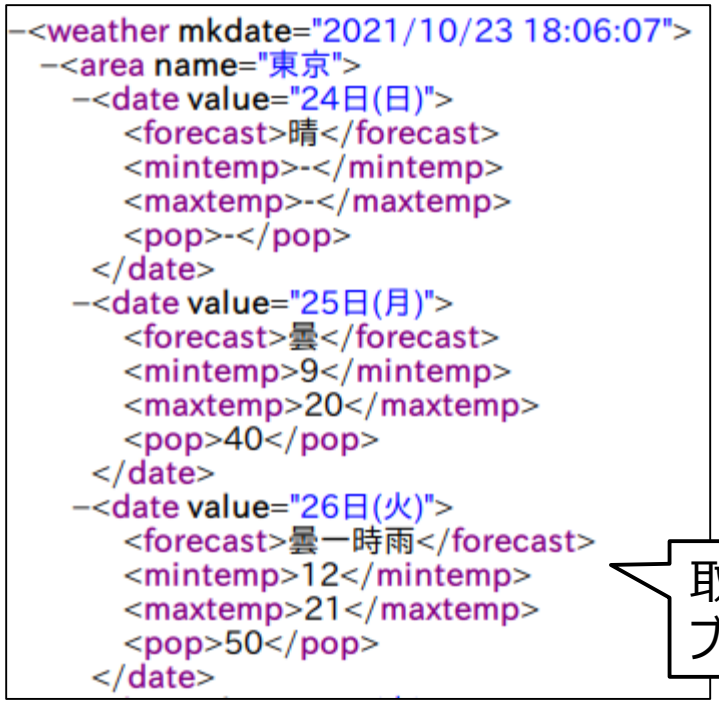
Web API

- クライアントとなるWebアプリケーションがWeb経由でサーバが提供するサービスを利用するためのインタフェース
  - XMLやJSON等のデータが返される

(例) 単純な無料Web API利用例

https://api.aoikujira.com/tenki/week.php?fmt=xml&city=319

URLによる引数渡し
(東京のID)



(1) HTTP GETメッセージで
URLを送信
(2) サーバ側で引数部分を
プログラムに渡す
(3) プログラムで処理
(プログラムの言語等は多種多様)

確認問題

- 以下の各文は正しいか。○か×で答えよ。
  - 動的なWebページを作成するにはサーバ側でページを生成するプログラムの稼働が必須である。
  - Cookieとは、セッション生成時にクライアントが生成し、サーバに送付する、セッション状態管理のためのデータである。
- 以下はHTTPの基本的なメソッドについての説明である。各説明に合うものを語群から選び答えよ。
  - ドキュメントのヘッダを要求する。
  - サーバからドキュメントを取得する。
  - サーバにドキュメントを送信する。

語群：post, delete, head, get, add, send

- HTTPでの通信の結果、ステータスコード400番台のメッセージが返ってきた。これを解消するためのクライアント側の対策として適切なものはどれか。
  - (1) すぐに同じ要求を再送する。
  - (2) サーバ側の問題が解決するまで待ってから再送する。
  - (3) アクセス先のURL指定など、要求の内容を見直す。

確認問題

```
<html lang="ja">
 <head> ...
  <script type="text/javascript">
   const xhr = new [          ]
   window.onload = function(){
    xhr.addEventListener('load', function(){
     if(xhr.status != 200){
      window.alert("Error" + xhr.status + ":" + xhr.statusText);
     }
     const xmlDoc = xhr.[          ]
     const date = xmlDoc.getElementsByTagName("date")[0].getAttribute("value");
     const weather = xmlDoc.getElementsByTagName("forecast")[0].textContent;
     document.getElementById('elem').textContent = date + " " + weather;
    }, false);
  }
  function buttonClick(){
   xhr.open('GET', 'https://api.aoikujira.com/tenki/week.php?fmt=xml&city=331');
   xhr.send(null);
  }
  </script>
 </head>
 <body><div id="sample">
  <center>
   <font size="7"><span id="elem">Push Button</span></font><br/>
   <input type="button" value="Get Server Date" onclick="buttonClick()" />
</center></div></body></html>
```

- Webページ内に記述されたスクリプトからHTTPリクエストを発してサーバからXML文書(等)を取得するために(　1　)クラスを使用する
- (　1　)によりサーバから取得したXML文書は(　2　)プロパティにより利用できる

講義内容

- Webアプリケーション
  - 概説
  - 要素技術
  - セキュリティ

Webアプリケーションのセキュリティ

- パスワードクラッキング
- DoS攻撃
- セッションハイジャック
- クロスサイトスクリプティング(XSS)
- SQLインジェクション
- クロスサイトリクエストフォージェリ(CSRF)

パスワードクラッキング

- ユーザのパスワードを不正に取得しようとする試み
- 代表的な手法
  - 辞書攻撃：よく使われる単語を単体あるいは組み合わせて、次々にログインを試す
  - ブルートフォースアタック：パスワードとして可能な文字の組み合わせをすべて試す
- 対策
  - 簡単なパスワードを設定できないようにしておく

DoS攻撃(Denial of Service)

- 短時間にサーバが処理しきれないような大量のアクセスをすることでサービス停止に陥らせる
- 代表的な手法
  - SYN Flood：TCP SYNパケットを大量に送りつける
  - F5攻撃：Webページの再読み込みを何度も繰り返す
- 対策
  - 不自然なアクセスを早期に検知し、当該IPからのアクセスを遮断する

セッションハイジャック

- Cookieの中身やセッションIDを取得し、正規のセッションを乗っ取る
- 代表的な手法
  - 盗聴
  - Webアプリケーションの脆弱性を突く
- 対策
  - 通信の暗号化
  - 急に異なるIPアドレスからアクセスした場合に強制ログアウトさせる

クロスサイトスクリプティング(XSS)

```
<?php
  session_start();
  // ログインチェック
?>
<body>
検索キーワード:<?php echo $_GET['keyword']; ?><br>
以下略
</body>
```

ログイン後に何かを検索することを想定したPHPスクリプト

```
http://.../sample.php?keyword=Hello
```

検索キーワード：Hello
以下略　　と表示

```
http://.../sample.php?keyword=
  <script>alert(document.cookie)</script>
```

CookieのセッションIDが表示されてしまう

クロスサイトスクリプティング(XSS)

- 実際には、攻撃対象者を罠サイトに誘導
  → 取得したCookie値を利用して正規サイトへのセッションを乗っ取る

- 対策
  - メタ文字(HTMLの文法上特別な意味を持つ記号)をエスケープ(escape)する
    - "<"を"<"に置換する等

```
keyword=&lt;script&gt;alert(document.cookie)&lt;/script&gt;
```

タグだとみなされないので無害

SQLインジェクション

injection[N]投入、注入

- XSSと同様にサーバにコードを送る。送る物が「SQL文」
  → データベース内の情報を不正に取得

```
$sql = "SELECT * FROM books WHERE author='$author'";
```

クエリ文字列が
' で開始

```
http://example.jp/ex.php?author='+AND+EXTRACTVALUE(0,(SELECT+
CONCAT('$',id,':',pwd)+FROM+users+LIMIT+0,1)+%23
```

表users 0行目ユーザのidとpwdを取得

- 対策
  - データベースの権限設定
  - SQL文の文字列リテラルの扱いに注意

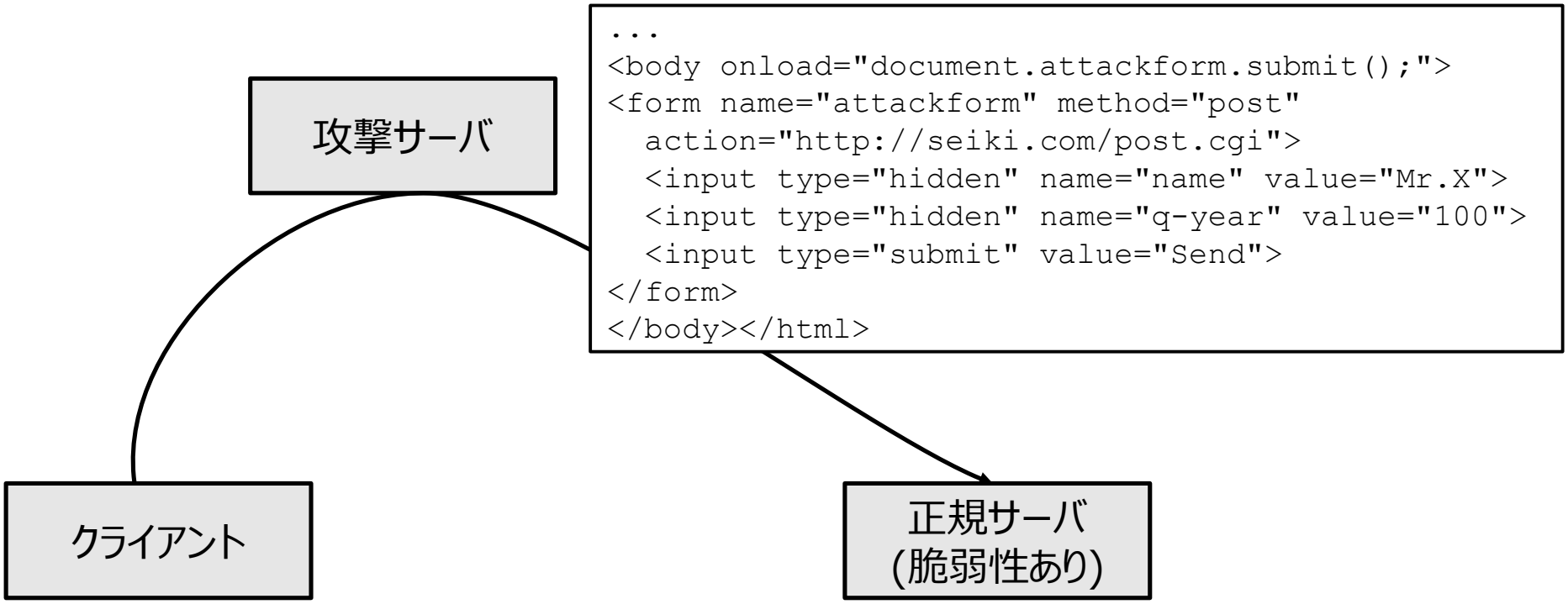
```
$sql = "SELECT * FROM books WHERE author=?";
...
(bindValueメソッドによる ? と $author の対応付け)
```

プレースホルダ
(?)を利用

クロスサイトリクエストフォージェリ (CSRF)

forgery [N] 偽造

- 意図しないHTTPリクエストの偽造



- 対策
  - 正規サーバにおけるセッション確認等

確認問題

- 各攻撃方法の名称を選択肢から選んで答えよ。
  - Webサイトの脆弱性を利用して、不正なSQL文を実行させる。
  - Webサイトの脆弱性を利用して、正規利用者に不正なスクリプトを含むWebアプリケーションを実行させる。
  - ユーザのパスワードを不正な手段で取得しようとする。
  - セッションIDを盗聴し、正規セッションを乗っ取る。
  - 短時間に大量の要求を送りつける。

選択肢：パスワードクラッキング、DoS攻撃、
セッションハイジャック、XSS、SQLインジェクション、

確認問題

- DoS攻撃を防止するために有効な手段は
  以下のうちどれか。
  - (1) 通信を暗号化する。
  - (2) サーバプログラムの保守性を改善する。
  - (3) 不自然なアクセスを早期に検知する。
  - (4) 簡単なパスワードを設定できないようにする。
- クロスサイトスクリプティングの防止のために、利用者から送信された文字列に含まれるメタ文字を無害な文字に置換する処理を何と呼ぶか。

参考文献

- 「この一冊で全部わかるWeb技術の基本」
  小林 恭平、坂本 陽 緒、佐々木 拓郎 監修、
  ソフトバンククリエイティブ、2017
- 「体系的に学ぶ安全なWebアプリケーションの作り方 第2版」徳丸 浩 著、ソフトバンククリエイティブ、2018
- 「分散システム」
  水野 忠則 監修、共立出版、2015
- 「分散システム 原理とパラダイム 第2版」
  アンドリュー・S・タネンバウム 他 著、
  ピアソン・エデュケーション、2009
- 「JavaScriptコードレシピ集」
  池田泰延、鹿野壮 著、技術評論社、2019
- 「ネットワークセキュリティ」
  菊地浩明、上原哲太郎 著、オーム社、2017