G's ACADEMY TOKYO

Ajax

1

Ajaxとは

Ajaxとは

- AjaxのAである「Asynchronous（非同期）」は、
  非同期でのクライアント・サーバ間の通信を指します

- JavaScriptによりクライアント上での動作が主で
  必要なデータのみ受信することで通信量の負担を軽減

- 特殊なサーバの設定等は必要としない

Ajaxのメリット

- Webページのリンクをクリックした時のレスポンス待ち時間の体感時間が少ない。

- 必要な部分の情報のみを取得変更し、必要なときに更新可能のため高速に動作する。

- 例）
  ５画面 → 5HTMLファイル（通常の方法）

  ５画面 → 1HTMLファイル（Ajaxだとこうできる！）

Ajaxのデメリット

- SEO対策には不向き
- スクリプトの知識が必要
- 作りが複雑になりがち
- 更新履歴が残りません、ブラウザの[戻る]ボタンでは1つ前の状態には戻りません。
- 更新後でも再表示すると初期状態に戻ってしまいます。

Ajaxとは（非同期通信）

あとでゆっくり読んでね！

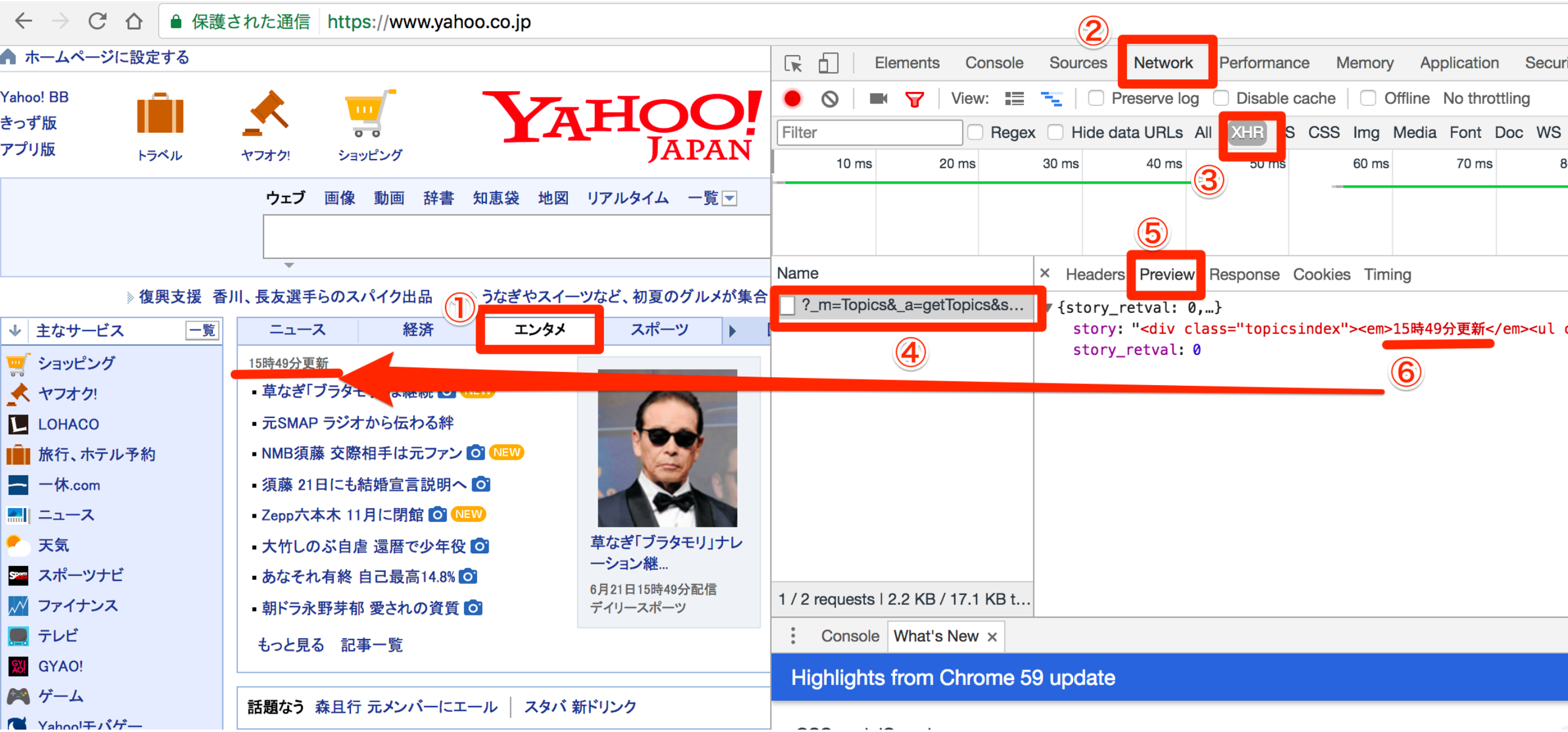
Ajaxとは「Asynchronous（非同期）＋　javascript ＋ XML」の略語。

「非同期」でのクライアント・サーバ間のデータ通信技術のことです。
反対語の「同期」は通常のWebページのリンク（aタグ）をクリックした場合の挙動がイメージしやすいでしょう。リンクをクリックするとページが切り替わりページ表示されるまで操作待ちとなります。

しかし「非同期（ajax）」を利用すると、必要なデータだけを取得して部分的にページ内容を更新することができます。そのためデータ取得、表示切り替え時も他の操作をすることも可能です。
ページの切り替えが必要なく、部分的なコンテンツのみ表示を更新することが可能になるのです。Webページでよくありがちなリンクをクリックして次のページを読み込むまで操作ができないようなことが防げます。また「ajaxのx」はXMLのXですが、最近ではXMLではなく「 JSON / JSONP 」を使用するケースが多くなりました。

Ajaxとは（Chrome:Yahooサイトで確認）

◇ChromeブラウザでYahooサイトを表示 → 右クリック(検証)



Ajaxの構文（全体像）

```
$.ajax({
    type: "get",    //default=get
    url: "*********",
    cache: false,
    datatype: "jsonp",
    success: function(data){
        alert("通信成功");
    },
    error: function() {
        alert("エラー");
    },
    complete: function() {
        alert("完了");
    }

});
```

Ajaxの構文１

```
$.ajax({        //関数START
```

引数を記述

引数を記述（ 複数ある場合は ”,” を間に記述 ）

```
});             //関数END
```

type,urlプロパティ

```
$.ajax({
    type: “GET”,    //POST or GET
    url: ”＊＊＊＊＊＊＊＊＊＊”
});
```

《引数》
type: GET or POST（受け取るときはGET, 送信するときはPOST）
url:　通信先アドレスを指定

※パラメータを続けて記述する場合は、”,”を間に記述する。

cacheプロパティ

```
$.ajax({
    type: “GET”,
    url: “**********”,
    cache: false   //chacheの有無をtrue,falseで指定。
});
```

《引数》

cache: true or false

JSONの場合はデフォルトでcache:true
JSONPの場合はcache:false
JSONをキャッシュを持たせない場合はcache:falseを指定する。

datatypeプロパティ

```
$.ajax({
    type: “GET”,
    url: “**********”,
    cache: false,
    datatype: “json”
});
```

《引数》

datatype: text, html, xml, json, jsonp, script

（データ形式を指定）

dataプロパティ

```
$.ajax({
    type: “GET”,
    url: “**********”,
    cache: false,
    datatype: “json”,
    data:{
        “id”: 1,
        “name”: ”yamazaki”
    }
});
```

asyncプロパティ

```
$.ajax({
    type: “GET”,
    url: “**********”,
    cache: false,
    datatype: “json”,
    data:{ “id”: 1, “name”:”yamazaki” },
    async: false
});
```

《引数》
asyncに” false”を設定すると同期処理に変わる。
記述してない場合には自動で ” true ” が設定されている。

Ajaxの構文２

```
$.ajax({
    type: “GET”,
    url: “**********”,
    cache: false,
    datatype: “jsonp”,
    success: function(data){
        alert("通信成功");
    },
    error: function() {
        alert(”エラー”);
    },
    complete: function() {
        alert(“完了”);
    }

});
```

通信が失敗したときの処理を記述

successメソッド

```
$.ajax({
    type: “GET”,
    url: “*********”,
    datatype: “json”,
    success: function(data){
        var len = data.length;
         for(var i=0; i<len; i++){
             $("body").append('<p>'+data[i].id + ':' + data[i].mode+ '</p>');
         }
    }
});
```

通信が成功したときの処理を記述。第一引数を指定してJSONのデータを取得。

解説：Ajaxの記述方法 (参考までに)

```
//A.通信プロパティの設定(datatype:JSON)
var request = $.ajax({
    type: "GET",                          //送信方法
    url: "demo_ajax_data.json",    //データを取得するURL
    datatype: "json",                     //データ形式(html,text,xml,json...)
    data:{"id": "1", "name":"yamazaki"},  //URL先にデータを送る場合に使用
    timeout: 3000                         //接続待ち時間(1000で1秒,3000で3秒)
});
//B. 通信成功の処理
request.done(function(data) {
    alert(“通信成功”);       //ここに通信成功時の処理を記述していく
});
//C. 通信エラーの処理
request.fail(function() {
    alert(“通信エラー”);   //ここに通信エラー時の処理を記述していく
});
//D. 通信処理完了後の処理
request.always(function() {
    alert(“通信完了”);    //通信完了時の処理を記述
});
```

Elements | Network | Sources | Timeline | Profiles | Resources | Audits | Console

Preserve log

Name
Path

method_jsonp.html
/jquery

jquery-1.11.0.min.js
code.jquery.com

method_jsonp.php?callb...
/jquery

3 requests | 1.1 KB transferred

Headers | Preview | Response | Timing

Remote Address: ::1:80
Request URL: http://localhost/jquery/method_jsonp.php?callback=json_data
Request Method: POST
Status Code: 200 OK

Request Headers view source

Accept: text/javascript, application/javascript, application/ecmascript, application/x-ecmascript, */*; q=0.01
Accept-Encoding: gzip,deflate,sdch
Accept-Language: ja,en-US;q=0.8,en;q=0.6
Cache-Control: max-age=0
Connection: keep-alive
Content-Length: 33
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Host: localhost
Origin: http://localhost
Referer: http://localhost/jquery/method_jsonp.html
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/35.0.1916.114 Safari/537.36
X-Requested-With: XMLHttpRequest

Query String Parameters view source view decoded

callback: json_data

Form Data view source view decoded

id: 1
mode: hoge
type: entry
sleep: 0

Response Headers view source

Console | Search | Emulation | Rendering

PHP側の受け方

```
//PHP側の処理の書き方：一例
if(
    isset($_SERVER['HTTP_REFERER']) &&
    $_SERVER['HTTP_REFERER']== http://www.gs.com/index.php
) {
    //上記アドレスのリンクからAjaxでページを参照した場合はここを処理

    /* ここに正常動作する処理を記述：
    戻したい値、text文字列、html文字列、JSON文字列などを
    echo 関数を使用して戻す！！ */
    echo "<div>テスト戻し文字列</div>";
    exit();


}else{
    //上記アドレスから参照されなかった場合Errorを返す。
     echo "false";   //文字列でfalseをAjaxに戻す
     exit();          //処理終了
}
```

Ajax＆Session

- サンプルファイル
  ajax_sessionフォルダを参照

G's ACADEMY TOKYO

Fileupload

Form・Camera

◇ Camera/写真選択

```
<form method=“post” action=“送信先" enctype="multipart/form-data">
   <input type=“file” accept=“image/*” capture=“camera” name="upfile">
   <input type="submit” value="Fileアップロード”>
 </form>
```

① FILE選択できるようにする

<input type=“file” ……>

※他の例<input type=“text” ……>

② File送信時はenctype属性を指定

enctype="multipart/form-data”

③POST送信（Action=“送信先”）

◇ Camera/写真選択

```
<form method=“post” action=“送信先" enctype="multipart/form-data">
   <input type=“file” accept=“image/*” capture=“camera” name="upfile">
   <input type="submit” value="Fileアップロード”>
 </form>
```

② カメラ起動＆画像選択可能

input accept=“image/*” capture=“camera”

※ 他の記述方法例）accept="image/jpeg, image/gif, image/png”

※ 他の記述方法例）accept="audio/*"

※ 他の記述方法例）accept=”video/*"

※ 他の記述方法例）accept=“text/comma-separated-values” //CSV

FileUpload

◇ $_FILES：パラメータチェック

ファイルアップロード パラメータ取得

- !isset(パラメータ名)　パラメータがセットされているか？
- !is_int(パラメータ名)　数値なのか？

※このチェック（IF分岐処理）を入れないとエラーがでたりします。

```
if ( !isset($_FILES[‘upfile’][‘error’] ) || !is_int( $_FILES[‘upfile’][‘error’] ) ) {
    // echo 'パラメータが不正です';
} else {
    // echo ‘アップロード完了’;
}
```

◇ $_FILES：ファイル名、アップロード先のPath取得

```
if ( !isset($_FILES[‘upfile’][‘error’] ) || !is_int( $_FILES[‘upfile’][‘error’] ) ) {
    // echo 'パラメータが不正です';
} else {
    $file_name = $_FILES["upfile"]["name"];
    $tmp_path  = $_FILES["upfile"]["tmp_name"];}
```

FileUpload

◇ $_FILES：パラメータチェック

- is_uploaded_file　　アップロードOK!?
- move_uploaded_file　Tempフォルダからimgフォルダへ移動
- chmod　　ファイルに権限を付与する

```
// FileUpload [--Start--]
  if ( is_uploaded_file( $tmp_path ) ) {
      if ( move_uploaded_file( $tmp_path, $file_dir_path . $file_name ) ) {
            chmod( $file_dir_path . $file_name, 0644 );
            //echo $file_name . "をアップロードしました。";
            $img = '<img src="'. $file_dir_path . $file_name . '" >';
    } else {
            echo "Error:アップロードできませんでした。";
    }
}
 // FileUpload [--End--]
```

◇ アップロードしたファイルを表示する

```
$img = '<img src="'. $file_dir_path . $file_name . '" >';  //Imgタグを作成
```

上記をHTMLのbody要素内に <?=$img?> と埋め込むことで画像を読み込める。

◇ fileupload処理の流れ（シンプルバージョン）

```
<?php
if ( !isset($_FILES[‘upfile’][‘error’] ) || !is_int( $_FILES[‘upfile’][‘error’] ) ) {
      // echo 'パラメータが不正です';
}else{
  $file_name = $_FILES[“upfile"]["name”];          //"1.jpg"ファイル名取得
  $tmp_path  = $_FILES["upfile"]["tmp_name"]; //"/usr/www/tmp/1.jpg"アップロード先のTempフォルダ
  $file_dir_path = “upload/”;  //画像ファイル保管先
  $img="";
  // FileUpload [--Start--]
  if ( is_uploaded_file( $tmp_path ) ) {
      if ( move_uploaded_file( $tmp_path, $file_dir_path . $file_name ) ) {
            chmod( $file_dir_path . $file_name, 0644 );
            //echo $file_name . "をアップロードしました。";
            $img = '<img src="'. $file_dir_path . $file_name . '" >';
      } else {
            echo "Error:アップロードできませんでした。";
      }
  }
  // FileUpload [--End--]
}
?>
```