# 話題のhtmxで始める シンプルなWeb開発入門

2025/08/19

Kazuma Sekiguchi

# 自己紹介

関口和真

株式会社コムセントCTO
Webシステム開発、スマートフォンアプリ制作、
サーバー構築、運用など

スマートフォンを使った動画配信アプリの制作
サーバーサイドシステムの作成
フロントエンド部分の作成
AI周りのいろいろ

# 今回のAgenda

- htmxとは何か？
- 基本構文の仕組み
- 主要属性の詳細
- イベントフック
- Alpine.jsとの併用


# htmxとは

- HTMLの属性を利用し、ページ全体をリロードせずに
  部分的な更新や動的なコンテンツ取得を可能とする
  JavaScriptライブラリー
  - JavaScriptをほとんど書かずに利用することが可能
  - ただし、ベースはJavaScriptではある

- 簡易的にSPA（SinglePageApplication）の実装も可能

- JavaScriptが要らなくなる、ということではない
  - やはり複雑なことをするとなると、JSは必要

# htmxとは

- Webサイトを動的にしたいが、JavaScriptのコードは良く分からないという人には向いている
  - 細かいロードの部分などの処理はhtmx自体が担ってくれる
  - プロトタイプ作成時などに利用すると効率的

- DOM要素自体を置換えるように作用するため、JSONなどを吐き出すサーバーシステムでは改変が必要
  - HTML自体を出力してくれるシステムなら問題無い

# htmxとは

- DOMベースで改変されるため、jQueryなどとの相性は問題無い
  - もちろん動的にDOMが変更されるため、それを考慮してjQueryなどを組む必要はある
  - 他のJSライブラリーとの相性も特に問題は起きづらいが、公式では問題無いと言っているが、React、Vueなどの仮想DOM系のものとの併用は避けた方が無難

# htmxとは

- 通信はバックエンドの通信（Ajax通信）として行われるため、ユーザーの待ち時間は最小限にすることができる
  - 極端な話、枠と後から内容を入れ込むdivタグなどを用意しておけば、ほぼ空のHTMLだけユーザーにロードさせて、中身を後から渡すことが可能
- バックエンドの通信は非同期であるため、いつ反映されるかは補償されない
  - 同期通信も可能ではある

# 基本の構文

- HTMLにおいて、<a href="item.html">はitem.htmlというファイルデータをGETして、レスポンスデータをブラウザー上に表示せよ、という命令
- htmxではどのタグに対してもデータをGETやPOSTするように指示を出すことが可能
  - レスポンスでブラウザー全体を置換えるのでは無く、一部分だけ置換えることが可能
  - HTMLを受け取る必要がある、JSONデータなどを受け取る場合は整形するなりして、HTMLにする必要がある

# 基本の構文

```
<button hx-get="click.php"
    hx-trigger="click"
    hx-target="#maindiv"
    hx-swap="outerHTML"
>
    クリック
</button>
<div id="maindiv">もともと存在するデータ</div>
```

- ボタンをクリックすると、click.phpにアクセスして、データを取得し、maindivの中身を置換える

# HTTPメソッド

- hx-で示される属性を各タグに付与することで動作を決定づけることが可能
  - 組み合わせることで複合的な動作を行うこともできる
    - データを取得してきて、他のタグの中身を置換える、など

- hx-get:GETで取得
- hx-post:POSTで送信取得
- hx-put:PUTで置換え
- hx-patch:PATCHで更新
- hx-delete:DELETEで削除

動作自体はサーバー側の処理方法に依存する

クリックされると、ここの文字がdata.phpの内容に置換わる

```
<button hx-get="data.php">ボタン</button>
```

# 他のサーバーからデータを取得

- 通常はhtmxを動かしているHTMLと後からデータを取得する先のサーバーは同一である
  - 一応、別のサーバーからデータ取得することも可能
  - ただし、CORS（クロスオリジン対策）に抵触するため、サーバー側が対応しておく必要がある
  - htmxを動かしているHTML側でもmetaタグに以下を記述する

```
<meta name="htmx-config" content='{"selfRequestsOnly": false}'>
```

- 同じサーバーから取得する場合は、気にしなくて良い

# 拡張CSSセレクタ

- hx-target:指定したターゲットのデータを置換える
  - 拡張CSSセレクターを利用可能

| 拡張CSSセレクタ | 意味 | 構文例 |
|---|---|---|
| this | 自分自身を指定（デフォルト） | hx-target="this" |
| closest <CSSセレクタ> | CSSセレクタに一致する一番近い親要素を指定 | hx-target="closest ul" |
| find <CSSセレクタ> | CSSセレクタに一致する一番近い子要素を指定 | hx-target="find div" |
| next <CSSセレクタ> | CSSセレクタに一致する自分から見て後方にある要素を指定 | hx-target="next .text" |
| previous <CSSセレクタ> | CSSセレクタに一致する自分から見て前方にある要素を指定 | hx-target="previous .text" |

# イベントの指定

- 一番単純なのは、id属性を付与しておいて、hx-target="#id属性値"とするのが分かりやすいはず
  - クラス属性も利用可能だが、混乱を招きやすいこともあるため、id属性を推奨
- hx-triggerでイベントを指定することが可能
  - 指定されたイベントが発生したときに通信が生じる

| イベント名 | 対象タグ |
| --- | --- |
| change | input , textarea , select |
| submit | form |
| click | ほとんどのタグで利用可能 |
| keyup | input,textarea |

# イベント修飾子

- 標準イベント修飾子というものが利用可能
  - イベントにオプション的に組み合わせて利用することが可能
  - イベントを拡張し、複雑なトリガーとして動作させることが可能

```
hx-trigger="keyup changed delay:1s"
```

  - 1秒間何も入力が無ければ、トリガーが動作する

| オプション名 | 意味 |
| --- | --- |
| once | 1回のみ利用可能 |
| changed | 要素が変更したときだけ発生 |
| delay:<時間> | 発火する前に遅延が発生する |
| throttle:<時間> | 発火後に遅延が発生する |

# 特別なイベント

- 特別なイベントとしてロードとスクロール、ポーリングに関係するイベントが定義されている

| イベント | 説明 |
| --- | --- |
| load | ロード時（要素が全てロード完了したとき）に発火する |
| revealed | この要素までスクロールしたときに発火する |
| every <時間> | everyの後にスペースを空けて時間を指定することで、指定した時間毎に要求を出す（ポーリング）ことが可能 |

- 複数のトリガーを指定するときはカンマで区切る

# リクエストインジケーター

- リクエストインジケーターを表示させることも可能
  - htmx-indicatorクラスをインジケーターを表示するための要素に指定する
  - htmxが通信を行うと、htmx-requestクラスが付与されて、htmx-indicatorクラスの不透明度が1になり表示される
  - htmx-requestクラスを他の要素に追加する場合は、hx-indicator属性を利用する

```
<button hx-get="click.php" hx-indicator="#indicator">
クリック
</button>
<img src="spinner.gif" class="htmx-indicator" id="indicator">
```

# HTMLの変更方法

- HTMLの変更方法を変える場合は、hx-swapを利用して、値を指定する

```
<button hx-get="click.php" hx-swap="innerHTML" hx-target="#target">
クリック
</button>
<div class="target">ここの中身だけ変わる</div>
```

# hx-swapの値

| 設定値 | 説明 |
| --- | --- |
| innerHTML | ターゲット要素の中身だけ置換える（ターゲット要素のタグは残る） |
| outerHTML | ターゲット要素全体を置換える（ターゲット要素のタグが消える） |
| afterbegin | ターゲット内の最初の子要素の前にコンテンツを追加する |
| beforebegin | ターゲットの親要素の前にコンテンツを追加する |
| beforeend | ターゲット内の最後の子要素の後にコンテンツを追加する |
| afterend | ターゲットの親要素の後にコンテンツを追加する |
| delete | レスポンスに関係なくターゲット要素を削除する |
| none | レスポンスがあってもコンテンツを追加しない |

# OOBスワップ（Out of Band Swap）

- サーバー側のHTMLレスポンス内にhx-swap-oob属性を付けておくことで、ブラウザーの任意の要素を更新できる仕組み
  - true以外にbeforebegin,afterend,innerHTMLなども指定できる

```
<div id="target">ここがターゲット</div>
<div id="oobtarget"></div>
<button hx-get="list" hx-target="#target">
更新</button>
```

HTMLの内容

```
<p>メインエリア更新内容</p>

<!-- OOBスワップ部分 -->
<div id="oobtarget" hx-swap-oob="true">
更新完了しました
</div>
```

サーバーからのHTML

# POSTパラメータ

- Form内に入力した値はパラメータとして渡される
  - hx-parmsを指定することで、パラメータを制限することができる

| 設定値 | 説明 |
| --- | --- |
| * | 全てのパラメータを許可する |
| none | 全てのパラメータを渡さない |
| not <param-list> | 指定したパラメータを渡さない（カンマ区切り） |
| <param-list> | 指定したパラメータだけ渡す（カンマ区切り） |

- hx-includeを使うと他の要素の値を含むことが可能

```
<button hx-post="register.php" hx-include="[name='email']">登録</button>
メールアドレス <input name="email" type="email"/>
```

# イベントフック

- htmxが内部で発火するカスタムイベントに対して処理を割り込ませるための仕組み
  - リクエスト送信前やレスポンス処理後などに独自のJSを実行できる
  - JSを利用して処理を追加することができる

| イベント名 | タイミング |
| --- | --- |
| htmx:configRequest | リクエスト送信直前（パラメータやヘッダーの追加・変更） |
| htmx:beforeRequest | リクエスト送信直前（DOM変更やローディング表示などに利用） |
| htmx:beforeSwap | レスポンスをDOMに反映する直前 |
| htmx:afterSwap | DOMに反映した直後 |
| htmx:responseError | リクエスト失敗時 |
| htmx:afterRequest | リクエスト完了後（成功・失敗は問わない） |

# イベントフック

- レスポンス反映後にアニメーションを起こしたりすることが可能

```
document.body.addEventListener('htmx:afterSwap,function(event){
    event.target.classList.add('fade-in');
});
```

fade-inクラスを適用する

DOMに反映した直後

- イベントは伝播するので、document.bodyで受けることが可能
  - jQueryのon()でも対応可能

# Alpine.jsとの組み合わせ

- 軽量でHTML属性ベースのJSフレームワーク
  - htmxと結構似ている位置づけだが、JSがベース
  - ReactやVueのような機能を極めてコンパクトに使える
- HTML属性にx-から始まる属性や@から始まるイベントなどを記述することで、動作をさせることができる
- htmxと組み合わせることで、部分更新とインタラクションを実現することが容易にできる

# Alpine.jsとの組み合わせ

```
<div x-data="{open:false, msg:''}">
  <button @click="open = !open">検索条件</button>
  <section x-show="open" class="border p-2">
    <form hx-post="/search" hx-target="#result" hx-indicator="#loading">
      <input name="q" class="border" />
      <button type="submit">検索</button>
    </form>
  </section>
  <div id="loading" class="hidden">Loading...</div>
  <div id="result"></div>
</div>
```

初期状態は open:false

クリックでopenは逆転する

openがtrueになれば表示される

htmxでsearchに対してPOSTでパラメータを送信

htmxで結果が返ってくるまで表示する

htmxで結果が返ってきたら表示する

- 開閉部分はAlpine.jsを利用する

# Alpine.jsとの組み合わせ

- htmx部分は外部との通信と部分更新だけ担当させて、UI部分の状態などはAlpine.jsに任せる、などの責務分担を行う使い方がベター

# 参考類

- 『htmxを使ってみた』
  - https://qiita.com/Final1900/items/49a8ac6f27a2427f6306
  - 13回に渡り記載されているので参考になる
- Carson Gross（著), Adam Stepinski（著), Deniz Akşimşek（著)、『ハイパーメディアシステム――htmxとRESTによるシンプルで軽やかなウェブ開発』,2025,技術評論社
  - htmxの作者らが書いた本
  - ただし、htmxだけを書いている本ではない

ありがとうございました。

状況やスキルによってはhtmxは非常に力強い
味方になってくれるはずです