# </> htmx

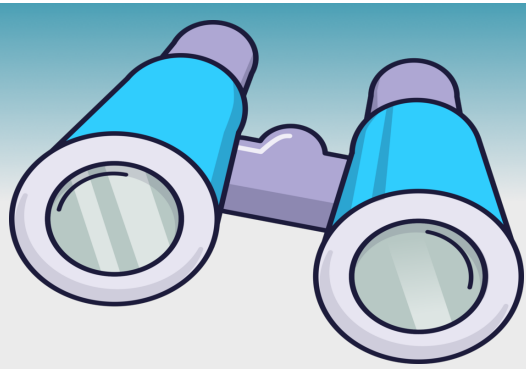

# htmx - past is now future

R. Mark Volkmann
Object Computing, Inc.
https://objectcomputing.com
mark@objectcomputing.com
@mark_volkmann

OBJECT COMPUTING
YOUR OUTCOMES ENGINEERED

Slides at **https://github.com/mvolkmann/talks/**

# htmx Overview



- Client-side JavaScript library for implementing hypermedia-driven applications (HDAs)
  - adds support for new HTML attributes that make HTML more expressive
  - uses endpoints that return HTML rather than JSON
  - free and open-source
- Sponsored by 19 companies (as of Feb. 2024)
  - including GitHub and JetBrains
- https://htmx.org/ Zero-Clause BSD license

# htmx Improves HTML

| HTML | htmx |
| --- | --- |
| only anchor and `form` elements trigger HTTP requests | any element can trigger HTTP requests |
| only triggered by clicking an anchor or submitting a form | any event can trigger |
| only GET and POST requests are sent | any verb can be used, including PUT, PATCH, and DELETE |
| response causes a full page refresh | response can be inserted into current page |

# SPA Frameworks vs htmx

- **SPA approach**
  - HTTP responses contain JSON
  - client-side code parses JSON
  - client-side code uses result to build DOM
  - framework code inserts new DOM into current page
- **htmx approach**
  - HTTP responses contain HTML
  - browser automatically builds DOM from HTML
  - htmx inserts new DOM into current page

# HATEOAS

- Stands for Hypermedia As The Engine Of Application State
- Major focus of htmx
- **Hypermedia**: any data format that can describe branching from one “media” (ex. a document) to another
- HTML is a kind of hypermedia, but JSON is not
- **Hypermedia client**: software that understands and renders a hypermedia format, such as web browsers with HTML
  - no custom client-side code is required

# htmx Tech Stacks ...

- Can use any programming language and framework that can implement an **HTTP server** whose endpoints return **HTML responses**
- Referred to as “Hypermedia On Whatever you’d Like” (**HOWL**)
- **Good choices make it easy to**
  - create new endpoints for any HTTP verb
  - specify type checking and validation of request data
  - get request data from headers, path parameters, query parameters, and bodies
  - send HTTP responses that include headers and bodies that contain text or HTML

# ... Tech Stacks

- **Good choices have tooling that supports**
  - **fast server startup** with no build process or a simple one
  - **automatic server restarts** after source code changes are detected
  - **good HTML templating** support (such as JSX) without relying on string concatenation

or my npm package js2htmlstr

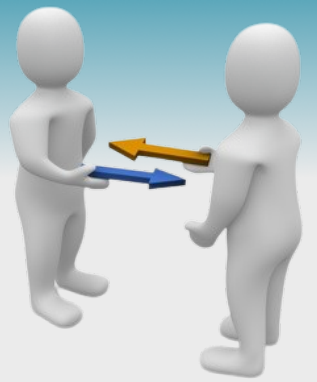
  - **syntax highlighting** of HTML in code editors

# htmx Basics


- Add htmx attributes to elements that trigger HTTP requests
- Specify events that trigger the request
  - **`hx-trigger`** comma-separated list of event names with optional modifiers
- Specify HTTP verb to use and endpoint URL
  - **`hx-get`**, **`hx-post`**, **`hx-put`**, **`hx-patch`**, and **`hx-delete`**
- Specify element where response HTML will go
  - **`hx-target`** can be CSS selector and/or use several keywords; defaults to **`this`**
- Specify where to place HTML relative to target
  - **`hx-swap`** see next slide

All elements have a **default trigger**.
**`form`** elements trigger on **`submit`**.
**`input`**, **`textarea`**, and **`select`** elements trigger on **`change`**.
All other elements trigger on **`click`**.

# hx-swap



Assume **`hx-target`** refers to the **`ul`** element.

Options to insert content

**`beforebegin`**
**`afterbegin`**
**`beforeend`**
**`afterend`**

```
<p>before list</p>
<ul>
  <li>Red</li>
  <li>Green</li>
  <li>Blue</li>
</ul>
<p>after list</p>
```

Options to replace content

**`outerHTML`**
**`innerHTML`** (default)

Options that do not use response HTML

**`delete`** removes target element
**`none`** leaves target element as-is

# Demo Time!

**Very fast server startup**

Automatically **restarts server** when server code is modified

Automatically **refreshes browser** when server is restarted or client code is modified

- These and more are at https://github.com/mvolkmann/**htmx-examples**
- **htmx-demo** - very basic — review this code
- **htmx-crud** - create, retrieve, update, and delete dogs — displays confirmation dialog for deletes
- ★ **lazy-load** - wait to build table until it scrolls into view — open DevTools Network tab
- ★ **active-search** - type-ahead retrieval of matching names

```
hx-trigger="keyup changed delay:200ms"
```

- **infinite-scroll** - table of over 1000 Pokemon — fetches 10 at a time
- ★ **todo-hono** - classic todo app — persists in SQLite database

★ **sleeps** to simulate long-running requests in order to demonstrate a **loading spinner**

# Resources


- **htmx home page** - https://htmx.org
  - see docs, reference, examples, talk, and essays
- **My blog** - https://mvolkmann.github.io/blog/ (select htmx)
- **My htmx example code** - https://github.com/mvolkmann/htmx-examples/
- **htmx Discord server** - https://htmx.org/discord
- **“Hypermedia Systems” book** - https://hypermedia.systems/

# Wrap Up


- **htmx** provides a new way of implementing web applications that has many benefits
  - HTML becomes more expressive
  - code is easier to understand
  - state management is simplified
  - can implement with any programming language
  - faster app startup
    - due to downloading much less client-side JavaScript
  - faster client/server interactions
    - due to removal of JSON generation and parsing