MONTANA
STATE UNIVERSITY

AJAX

Adding Interactivity To Web Applications

AJAX

- AJAX stands for Asynchronous Javascript and XML
- Technologies for issuing requests *asynchronously* outside of the normal HTML mechanisms
  - Links
  - Forms

AJAX

- Early 1990s - Websites were done with plain HTML pages
    - Each request caused a full page refresh
    - Clunky, but effective and *simple*
- 1998/1999 - Microsoft develops the XMLHttpRequest object (IE5)
- Allowed out of band HTTP requests in Javascript

AJAX

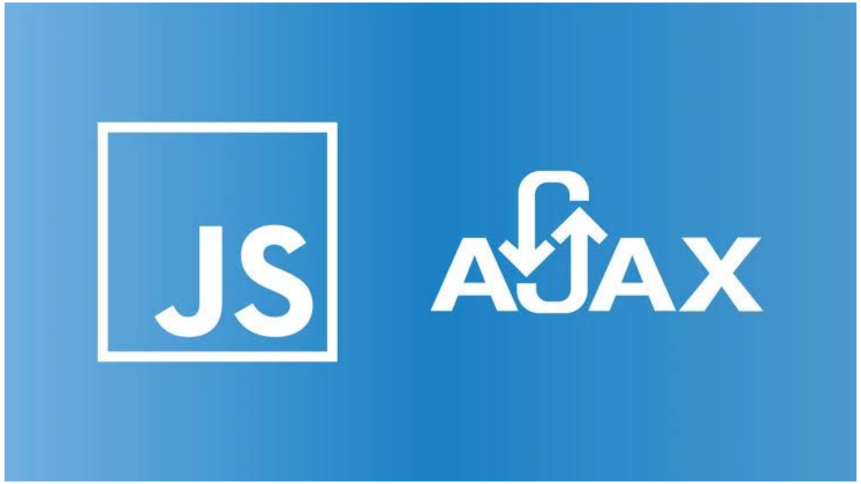
- API was Adopted by other browsers in early 2000s
- Originally XML was used as the data
- Early adopters (2004/2005)
  - GMail
  - Google Maps
  - Kayak
- 2005 - first use of the term AJAX

AJAX

- 2006 W3C standardizes the XMLHttpRequest object
- JSON, rather than XML, becomes the dominant way to transfer data
    - Easy to parse, easy to produce
    - https://www.json.org/

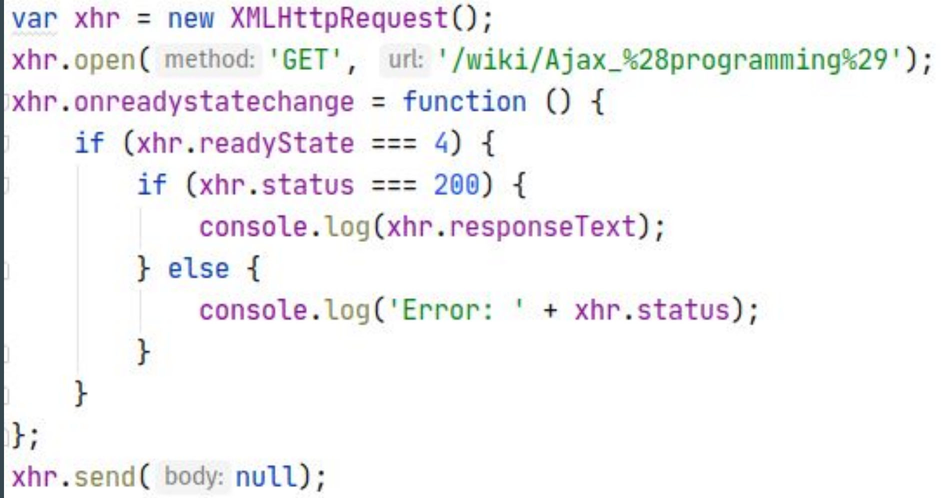
AJAX

- 2006 W3C standardizes the XMLHttpRequest object
- JSON, rather than XML, becomes the dominant way to transfer data
  - Easy to parse, easy to produce
  - https://www.json.org/



AJAX

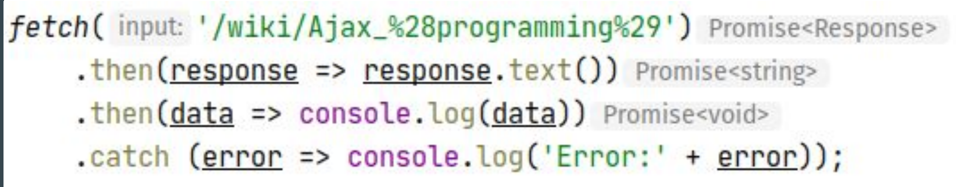
- Example working with the raw XMLHttpRequest API
- Pretty ugly, isn't it?



```
var xhr = new XMLHttpRequest();
xhr.open( method: 'GET',  url: '/wiki/Ajax_%28programming%29');
xhr.onreadystatechange = function () {
    if (xhr.readyState === 4) {
        if (xhr.status === 200) {
            console.log(xhr.responseText);
        } else {
            console.log('Error: ' + xhr.status);
        }
    }
};
xhr.send( body: null);
```

AJAX

- More modern API: *fetch()*
- I mean, I guess it's better…
- I'm not a big fan of promises and then chaining...



```
fetch( input: '/wiki/Ajax_%28programming%29') Promise<Response>
    .then(response => response.text()) Promise<string>
    .then(data => console.log(data)) Promise<void>
    .catch (error => console.log('Error:' + error));
```

SPA

- AJAX technology has been widely adopted
- The rise of the SPA - Single Page Application
    - No full page refreshes
    - All navigation and actions are handled with AJAX
    - Typically client side templates are used to render new UI

React JS

Vue.js

ANGULAR JS

SPA

- SPA Frameworks
  - Angular - From google, widely used in enterprise
  - React - From facebook, widely used by young people
  - Vue.js - Open source, gaining in popularity

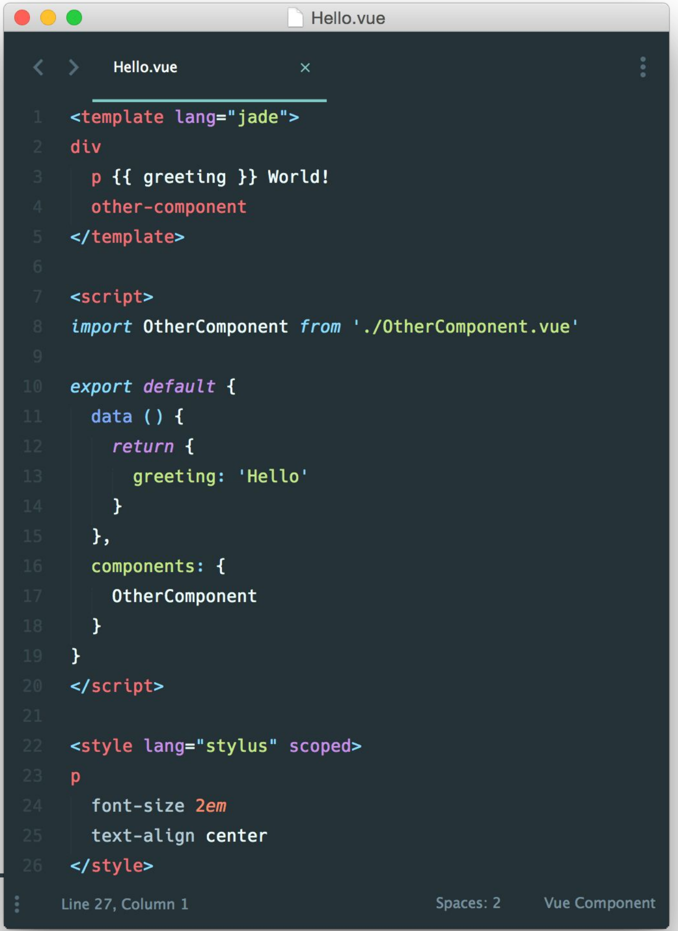
React JS

Vue.js

ANGULAR JS

SPA

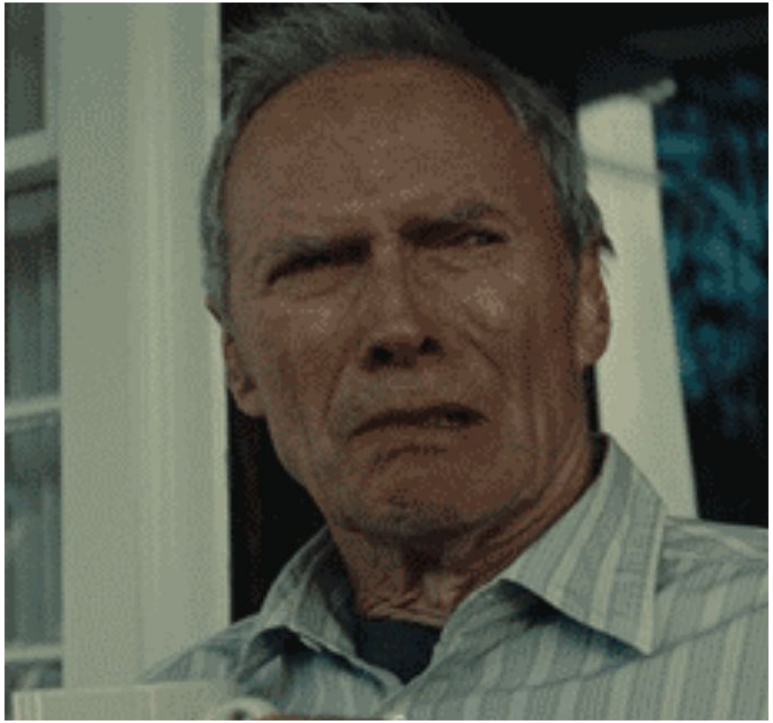
- Vue.js
  - Component oriented
  - Components encapsulate HTML and scripting behavior
  - Components are reactive to changes in an underlying model
    - Change the greeting var and this component will re-render
  - Includes a client-side routing mechanism



```
<template lang="jade">
div
  p {{ greeting }} World!
  other-component
</template>

<script>
import OtherComponent from './OtherComponent.vue'

export default {
  data () {
    return {
      greeting: 'Hello'
    }
  },
  components: {
    OtherComponent
  }
}
</script>

<style lang="stylus" scoped>
p
  font-size 2em
  text-align center
</style>
```

SPA

- > mfw
- If you want to get a job, you can go ahead and learn React or Vue
- But not in this class
    - Not on my watch

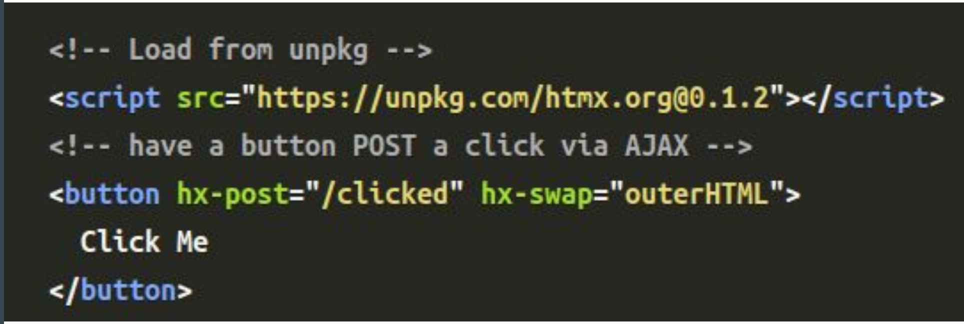


HTMX

- Let me tell you about a little library that I like to call htmx
  - allows you to add AJAX functionality to HTML directly in HTML
  - no javascript required
  - https://htmx.org

HTMX

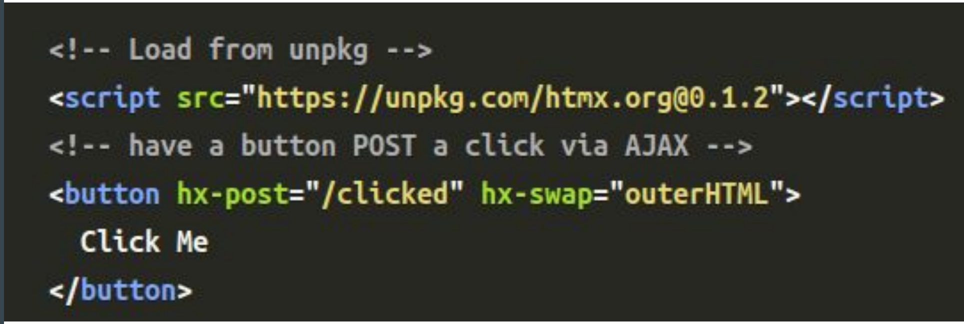
- History
  - Began life as intercooler.js
    - Based on jQuery
    - 4.6k stars on github
    - Part of the github arctic archive: https://archiveprogram.github.com/
  - Rewrote intercooler.js w/no jQuery dependency this summer
  - Renamed it to htmx



```
<!-- Load from unpkg -->
<script src="https://unpkg.com/htmx.org@0.1.2"></script>
<!-- have a button POST a click via AJAX -->
<button hx-post="/clicked" hx-swap="outerHTML">
  Click Me
</button>
```

HTMX

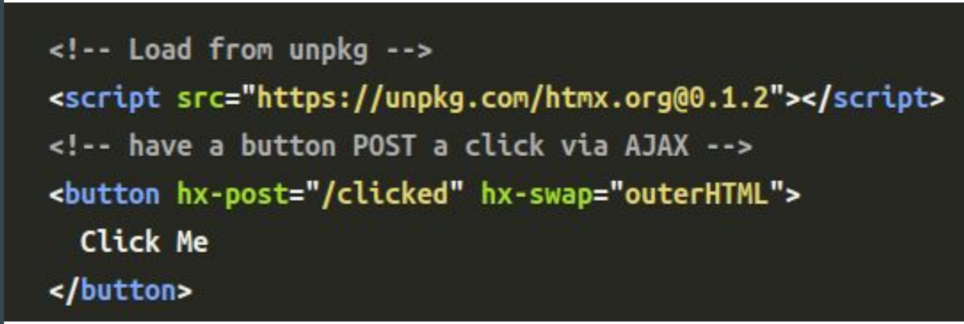
- Consider this button
  - When a click occurs
  - Issue a POST to /clicked
  - Take the content that is returned and swap it into the outerHTML of this button
- Response is expected to be HTML
  - Not JSON, not XML



```html
<!-- Load from unpkg -->
<script src="https://unpkg.com/htmx.org@0.1.2"></script>
<!-- have a button POST a click via AJAX -->
<button hx-post="/clicked" hx-swap="outerHTML">
  Click Me
</button>
```

HTMX

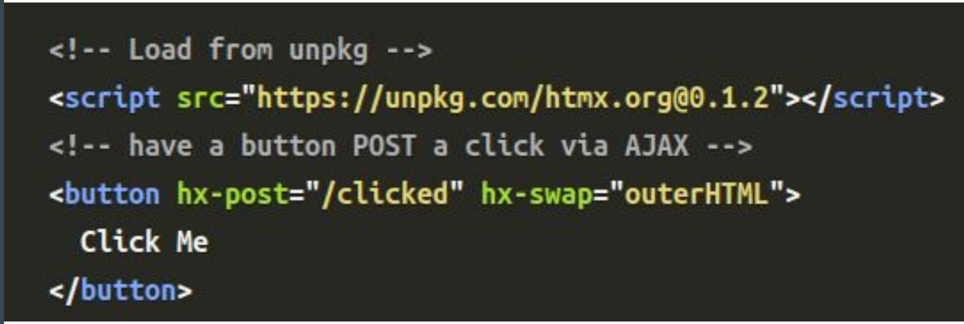
- Request is still issued asynchronously, using the standard XMLHttpRequest object
- Behavior is attribute driven
- There is a scripting layer as well, but not used extensively



```html
<!-- Load from unpkg -->
<script src="https://unpkg.com/htmx.org@0.1.2"></script>
<!-- have a button POST a click via AJAX -->
<button hx-post="/clicked" hx-swap="outerHTML">
  Click Me
</button>
```

HTMX

- Core attributes:
    - hx-get, hx-post, hx-put, hx-delete (!!!)
    - hx-target - target another element in the DOM
    - hx-swap - specifies how to swap (default: innerHtml)
    - hx-trigger - specify the event that causes a request to be issued

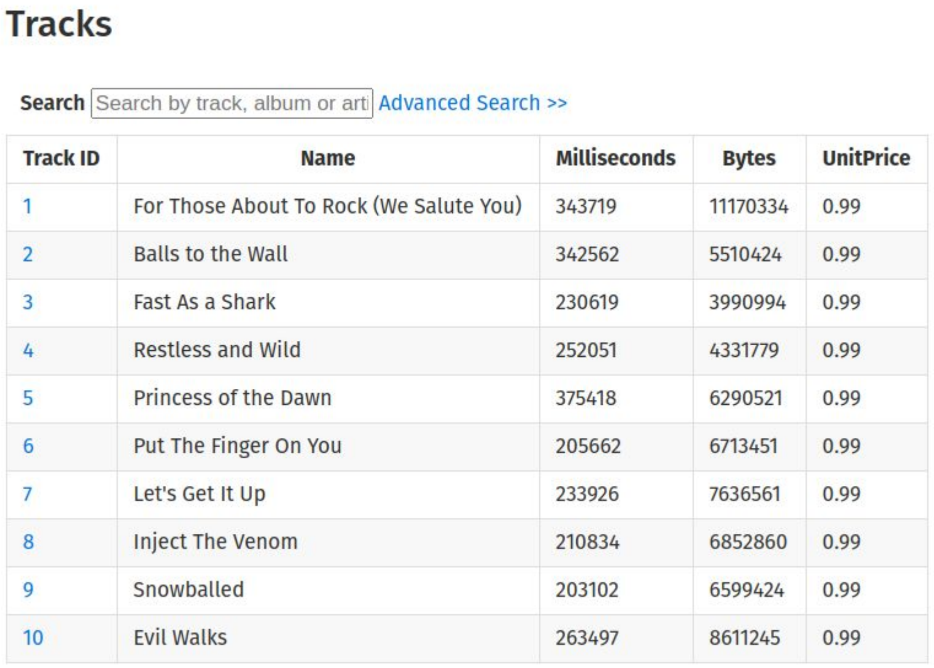


```html
<!-- Load from unpkg -->
<script src="https://unpkg.com/htmx.org@0.1.2"></script>
<!-- have a button POST a click via AJAX -->
<button hx-post="/clicked" hx-swap="outerHTML">
  Click Me
</button>
```

Next Level Search w/ HTMX

- Let's take our track search and next-level it by implementing *Active Search*
- As the user types, we filter the results
- Example code:

  https://htmx.org/examples/active-search/



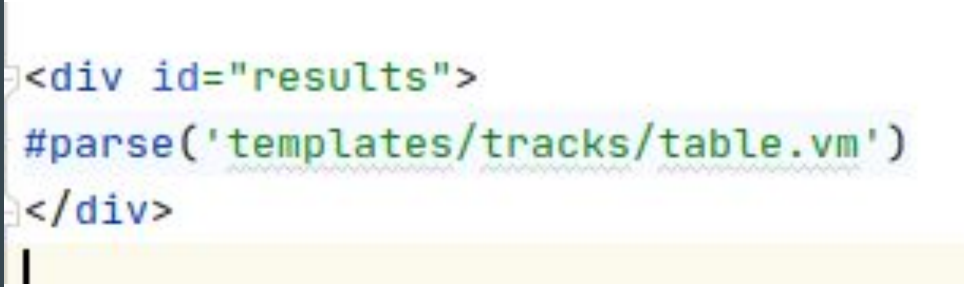
Tracks

Search Search by track, album or arti Advanced Search >>

Track ID	Name	Milliseconds	Bytes	UnitPrice
1	For Those About To Rock (We Salute You)	343719	11170334	0.99
2	Balls to the Wall	342562	5510424	0.99
3	Fast As a Shark	230619	3990994	0.99
4	Restless and Wild	252051	4331779	0.99
5	Princess of the Dawn	375418	6290521	0.99
6	Put The Finger On You	205662	6713451	0.99
7	Let's Get It Up	233926	7636561	0.99
8	Inject The Venom	210834	6852860	0.99
9	Snowballed	203102	6599424	0.99
10	Evil Walks	263497	8611245	0.99

Next Level Search w/ HTMX

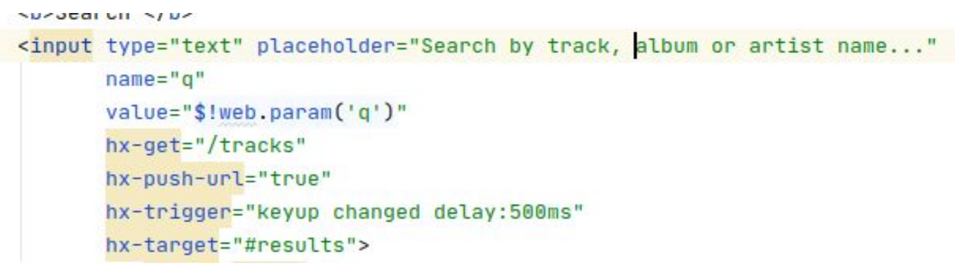
- Step 1: separate the table of results out to a new, independent template
  - Include it in the original index.vm file
  - Surround it with a div tag



```
<div id="results">
#parse('templates/tracks/table.vm')
</div>
```

Next Level Search w/ HTMX

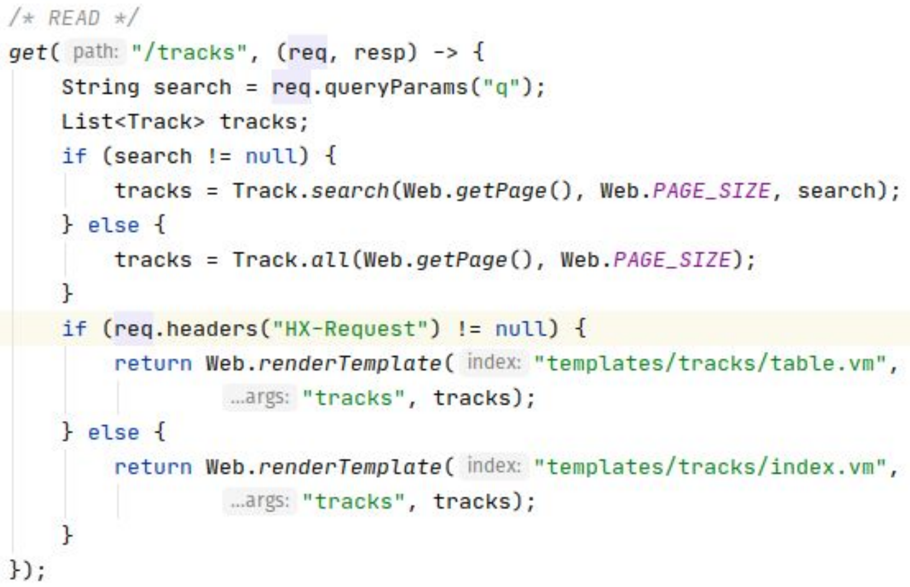
- Step 2: Add htmx attributes to our search input
  - hx-get - issue a request to /tracks
  - hx-push-url - you'll see
  - hx-trigger - trigger on a key up event, but only if it has been 500ms and the value of the input has changed
  - hx-target - put the response in the new div we created



```
<input type="text" placeholder="Search by track, album or artist name..."
       name="q"
       value="$!web.param('q')"
       hx-get="/tracks"
       hx-push-url="true"
       hx-trigger="keyup changed delay:500ms"
       hx-target="#results">
```

Next Level Search w/ HTMX

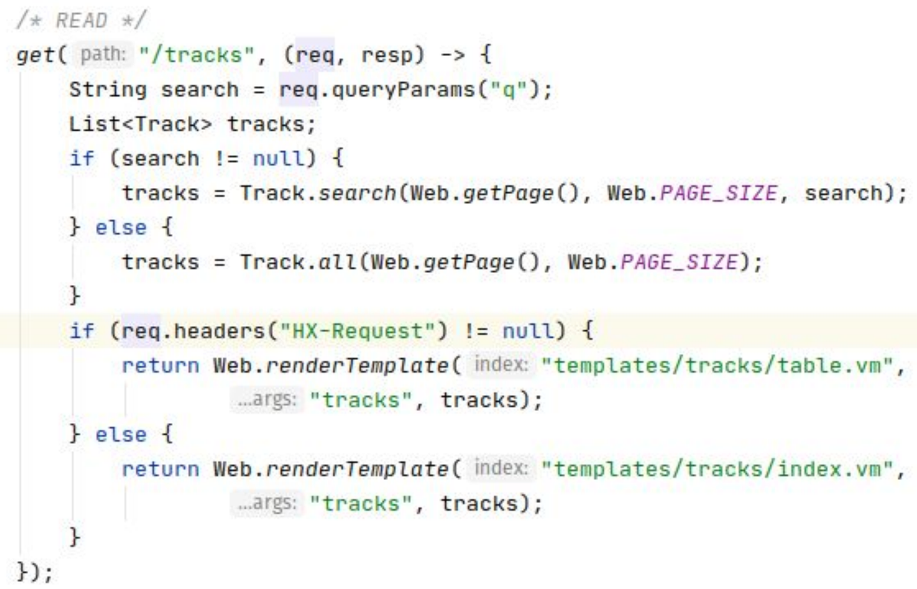
- Step 3: fix up the controller code
  - If this is an htmx request (indicated by the HX-Request header) render only the table
  - Otherwise, render the whole page



```
/* READ */
get( path: "/tracks", (req, resp) -> {
    String search = req.queryParams("q");
    List<Track> tracks;
    if (search != null) {
        tracks = Track.search(Web.getPage(), Web.PAGE_SIZE, search);
    } else {
        tracks = Track.all(Web.getPage(), Web.PAGE_SIZE);
    }
    if (req.headers("HX-Request") != null) {
        return Web.renderTemplate( index: "templates/tracks/table.vm",
                 ...args: "tracks", tracks);
    } else {
        return Web.renderTemplate( index: "templates/tracks/index.vm",
                 ...args: "tracks", tracks);
    }
});
```

Next Level Search w/ HTMX

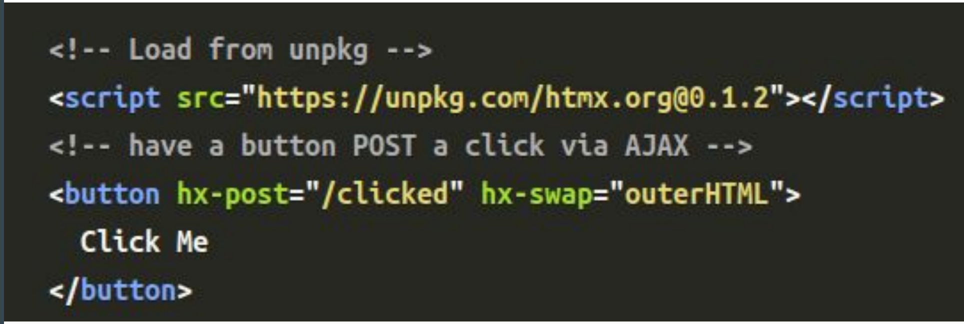
- That’s it!
- We are rockin’ and rollin’ with active search for tracks
- I *assure* you, this is impressive



```
/* READ */
get( path: "/tracks", (req, resp) -> {
    String search = req.queryParams("q");
    List<Track> tracks;
    if (search != null) {
        tracks = Track.search(Web.getPage(), Web.PAGE_SIZE, search);
    } else {
        tracks = Track.all(Web.getPage(), Web.PAGE_SIZE);
    }
    if (req.headers("HX-Request") != null) {
        return Web.renderTemplate( index: "templates/tracks/table.vm",
                ...args: "tracks", tracks);
    } else {
        return Web.renderTemplate( index: "templates/tracks/index.vm",
                ...args: "tracks", tracks);
    }
});
```

HTMX

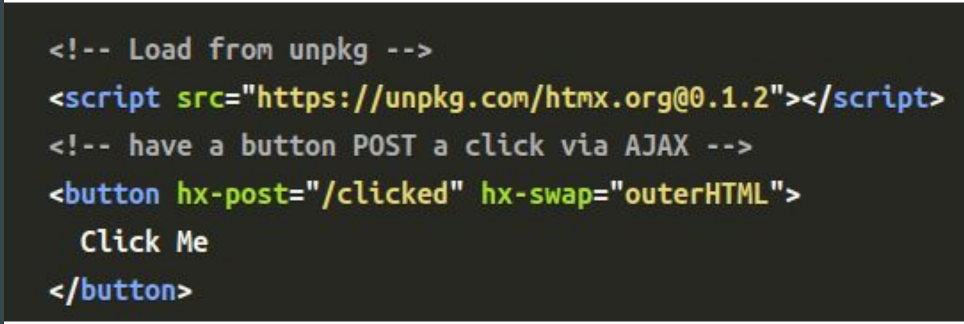
- There are lots of other examples and demos on the htmx website
- There is actually a pretty good theoretical basis for htmx based on REST



```html
<!-- Load from unpkg -->
<script src="https://unpkg.com/htmx.org@0.1.2"></script>
<!-- have a button POST a click via AJAX -->
<button hx-post="/clicked" hx-swap="outerHTML">
  Click Me
</button>
```

HTMX

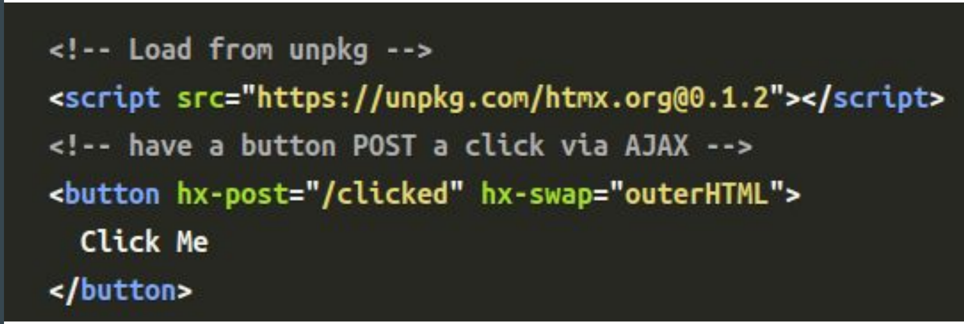
- REST: REpresentational State Transfer
  - The original theoretical basis for the web
  - Codified the notion of *hypermedia*
  - The stateless exchange of hypermedia *representations* of server resources



```
<!-- Load from unpkg -->
<script src="https://unpkg.com/htmx.org@0.1.2"></script>
<!-- have a button POST a click via AJAX -->
<button hx-post="/clicked" hx-swap="outerHTML">
  Click Me
</button>
```

HTMX

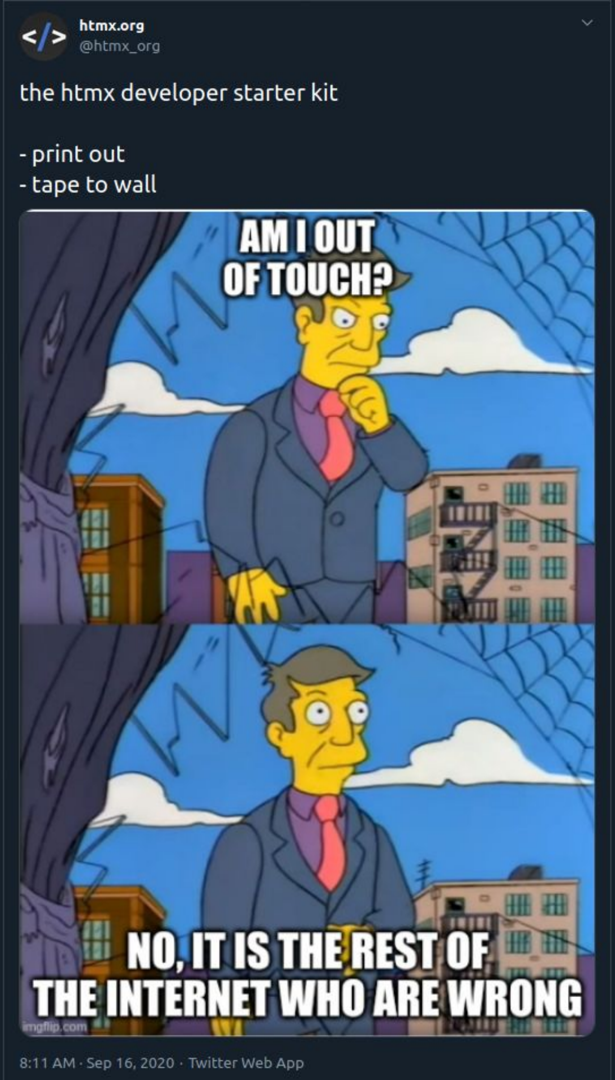
- A radical departure from older networking models
- One of the reasons why the web is so resilient is this new concept
- Htmx is firmly based on REST-ful principles
    - Most other front end libraries today are not



```html
<!-- Load from unpkg -->
<script src="https://unpkg.com/htmx.org@0.1.2"></script>
<!-- have a button POST a click via AJAX -->
<button hx-post="/clicked" hx-swap="outerHTML">
  Click Me
</button>
```

Ajax Summary

- Htmx is really, really rad
    - Active search in a few attributes!?!
- You should star it on Github
    - And tell all your friends about it
    - Also: htmx is rad



htmx.org
@htmx_org

the htmx developer starter kit

- print out
- tape to wall

8:11 AM · Sep 16, 2020 · Twitter Web App

MONTANA
STATE UNIVERSITY