

Web Services

Practical Definition:

- **Web Service**
 - Web call returning data for programmatic use
- **Endpoint**
 - The URL for that call

Services aren't "pages"

The difference is in how it is used

- Not a code difference (mostly)

Pages return HTML intended for the browser

- Other endpoints return CSS, JS, images, media

Service endpoints can return

- HTML fragments, text, JSON, XML, YAML, etc

Services used by frontend **and/or** backend

Service Conventions

- What do you send?
 - Url
 - Parameters
 - Headers
- How do you send it?
 - HTTP Method
 - Body/URL Parameters
- What do you get back?
 - Body
 - Status
 - Headers

One way: Anything goes

- Have some endpoint
- Send data
- Return a default 200 status code (probably)
- Return a body with results or error message

Newer devs write this a lot

- It "works"
- Isn't a good choice

Why?

Another way: SOAP

Anything goes, but with a lot of XML

- Benefits of XML
 - Schemas
- Drawbacks of XML
 - Verbosity
 - Parsing complexity
- Control at the data assembly/parsing
 - (server) Deciding action, identifiers
 - (client) Reading results, errors
 - Control not at the HTTP layer

Another way: GraphQL

- Send a "query" using a **query language**
 - Conditions that need to be true
 - Or changes to stored data
- List the fields you want in the response
- Query Language is not JSON, but similar
 - Kind of a pain to read/write w/o libraries
- Most commonly everything is POST
- Errors reported in response body
- "Heavy"
 - Almost always using libraries/frameworks

REST

Nuanced system, often done partially

Three Basic rules (my summary)

- URL represents a "resource" (to interact with)
- HTTP Methods are the interaction
 - GET: Read
 - POST: Create
 - PUT: Replace
 - DELETE: Delete
 - PATCH: Update
- HTTP Status code
 - Result of resource interaction

There are other paradigms, more coming

- All are just conventions on
 - What to send
 - How to send it
 - What you get back
- **Conventions** means no purity police
 - Some definitions are strict, some loose
 - No HTTP-based technical enforcement

Modern Day

- SOAP - older, now rare
- GraphQL - newer but established
- REST - older, still most common
- Others (tRPC, etc) - newer, growing?
- JSON most common data format

For this course:

- Use REST services
- Use cookies for service authentication
- Send services data as JSON in the body
- Services return JSON data as the body