

Mini-Project 3 : Fetch 'Em



Let's build a mini PokéDex that fetches live Pokémon data over HTTP, parses JSON, handles timeouts and retries, and compares callbacks, promises, and async/await with a cancelable, responsive UI.

Async + HTTP + API

Learning objectives

- Understand HTTP requests/responses, status codes, and JSON payloads.
- Use fetch for GET requests, parse JSON, and handle network errors.
- Compare callback style, Promise .then, and async/await.
- Use AbortController to cancel in-flight requests.
- Combine results from multiple endpoints in parallel with Promise.all.
- Implement timeouts, retries with backoff, and client-side caching.

Project Brief

1) What to build

A tiny PokéDex that lets users:

- Search Pokémon by name/id or click “Surprise me” (random).
- See sprite, name, id, types, base stats, flavor text color (from species).
- Show loading / error states, cancel button, and last 5 searches.
- Demonstrate three async styles:
 1. Callback (simulated async via setTimeout)
 2. Promise chaining (fetch(...).then(...))
 3. async/await (final production handler)

API: <https://pokeapi.co/api/v2/>

2) UI requirements

Be creative !

Tasks

1. HTML skeleton & styles

- Create a nice UI:
- Input (#q), buttons: Search, Surprise me, Cancel
- Status area (#status) and results card (#card)
- History list (#history)

2. HTTP + JSON (Promise chain)

- Implement a fetchPokemonThenStyle(query) using .then:
- GET /pokemon/{query}
- Parse JSON, extract name/id/sprite/types/stats
- Update UI or show 404 nicely if not found

3. Parallel fetch with Promise.all

- Fetch species in parallel to get the color/flavor:
- /pokemon/{id} and /pokemon-species/{id} in parallel
- Render a colored card border using the species color

4. Async/Await production handler

- Write fetchPokemon(query) using async/await that:
- Wraps both endpoints via Promise.all
- Has try/catch for network and HTTP errors
- Sets loading state while fetching

5. Abort in-flight requests

- Wire AbortController:
- Cancel button aborts ongoing request
- New search cancels the previous

6. Timeout & Retry

- Create fetchWithTimeout(url, {signal, timeout}) that races a timeout.
- Create retry(fn, {retries, backoffMs}) to retry transient failures (≥500).

7. Callback style (demo only)

- Make fakeCallbackPipeline(query, cb) using setTimeout to simulate async (e.g., input normalization → “validation” → success/fail), then the Promise/async path does the real API call.

8. History & caching


- Cache responses in a Map by id, serve from cache instantly
- Keep a last-10 search history

PokeFetch+ HTTP • JSON • Callbacks • Promises • async/await • Abort • Retry

Type a Pokémon **name** (e.g., `pikachu`) or **id** (e.g., `25`). Try *Surprise me*. Inspect the **Console** for detailed async logs.

Search Surprise me Cancel

Fetches #494 Victini



Victini #494

types: psychic / fire

This Pokémon brings victory. It is said that Trainers with Victini always win, regardless of the type of encounter.

HP100base

ATK100base

DEF100base

SpA100base

SpD100base

SPE100base

Last 10 searches

#494 victini

Telemetry

HTTP Status—

Total Time (ms)—

From Cache—

Retries0

4