

tt()

Schedule

1. Review topics (q&A)
2. Fetch
3. Dataset research
4. All together!



Schedule

1. Review topics (q&A)

2. Fetch

3. Dataset research

4. All together!



Review

- Objecten en arrays (structuur, dot notation)
- Functional programming (return, puur / impuur)
- Higher-order functions (map, filter reduce)
- API workshop (libraries, abstractie)

Review

- ~~Objecten en arrays (structuur, dot notation)~~
- ~~Functional programming (return, puur / impuur)~~
- ~~Higher-order functions (map, filter reduce)~~
- ~~API workshop (libraries, abstractie)~~
- Fetchen (promises, async)
- Dataset research

Schedule

1. Review topics (q&A)

2. Fetch

3. Dataset research

4. All together!



APIs

An API is an application programming interface. **It is a set of rules that allow programs to talk to each other.** The developer creates the API on the server and allows the client to talk to it.

Smashing – Understanding REST API's

Sync vs async

why?

Getting data from a resource (API) takes time. It needs to fetch the resource, parse it etc. But also, what if the data isn't available (no internet connection e.g.) how should *errors* be handled?

Sync vs async

Many Web API features now use **asynchronous code to run**, especially those that access or *fetch some kind of resource from an external device*, such as fetching a file from the network, accessing a database and returning data from it.

[MDN – Introducing asynchronous JavaScript](#)

SYNCHRONOUS LOAD



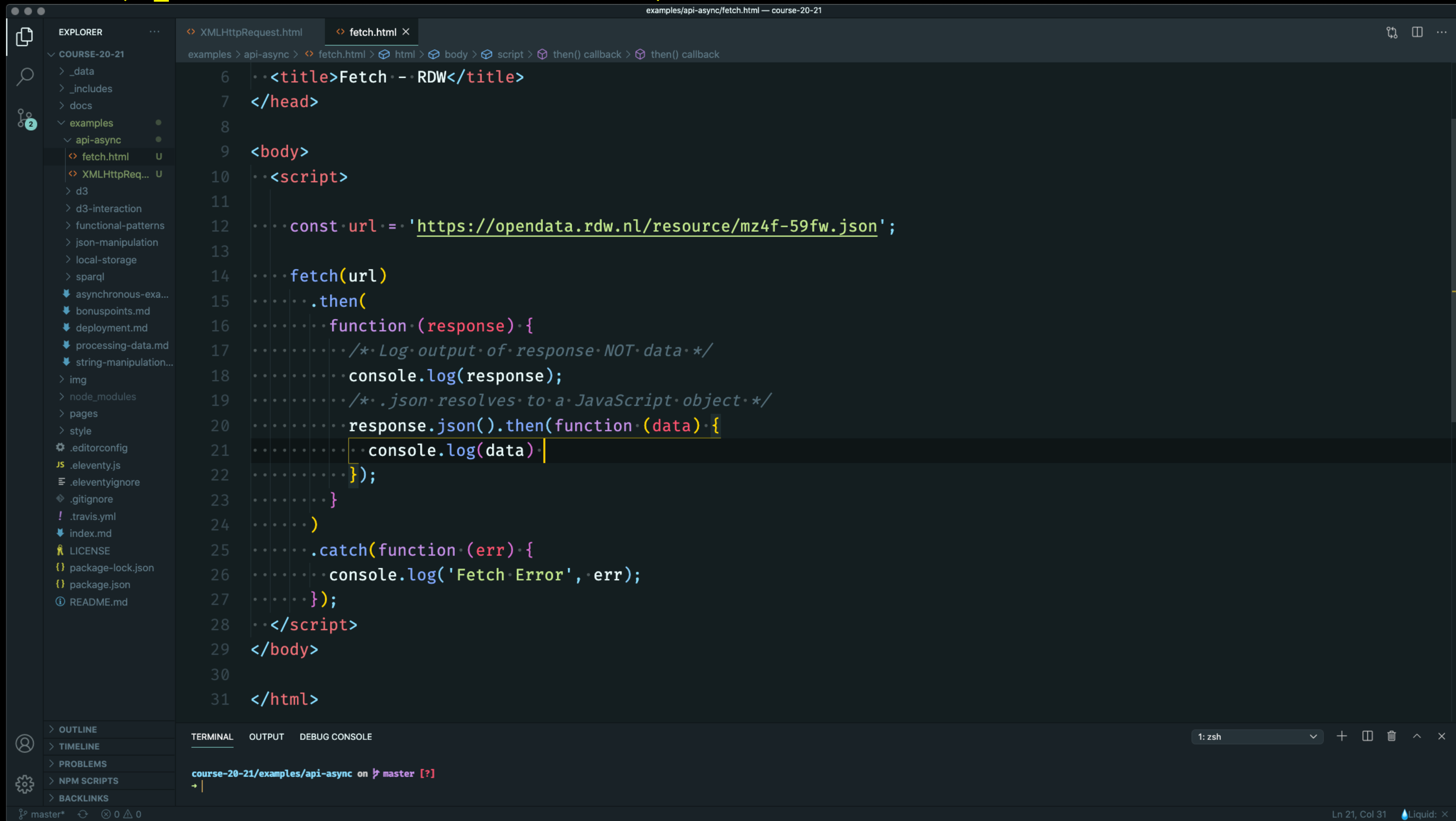
ASYNCHRONOUS LOAD



Fetching in JavaScript

- ❖ **Callbacks (XMLHttpRequest)**
- ❖ **Promises (Fetch)**
- ❖ **Async / Await (Fetch)**

Fetch (promises)



The screenshot shows a code editor with a file explorer on the left and a terminal at the bottom. The file explorer shows a project structure with a folder named 'examples' containing a subfolder 'api-async' which has a file 'fetch.html'. The code editor displays the content of 'fetch.html', which is an HTML document with a title 'Fetch - RDW' and a script that uses the Fetch API to retrieve data from a JSON endpoint. The script logs the response to the console and handles errors. The terminal shows the current directory and the git status.

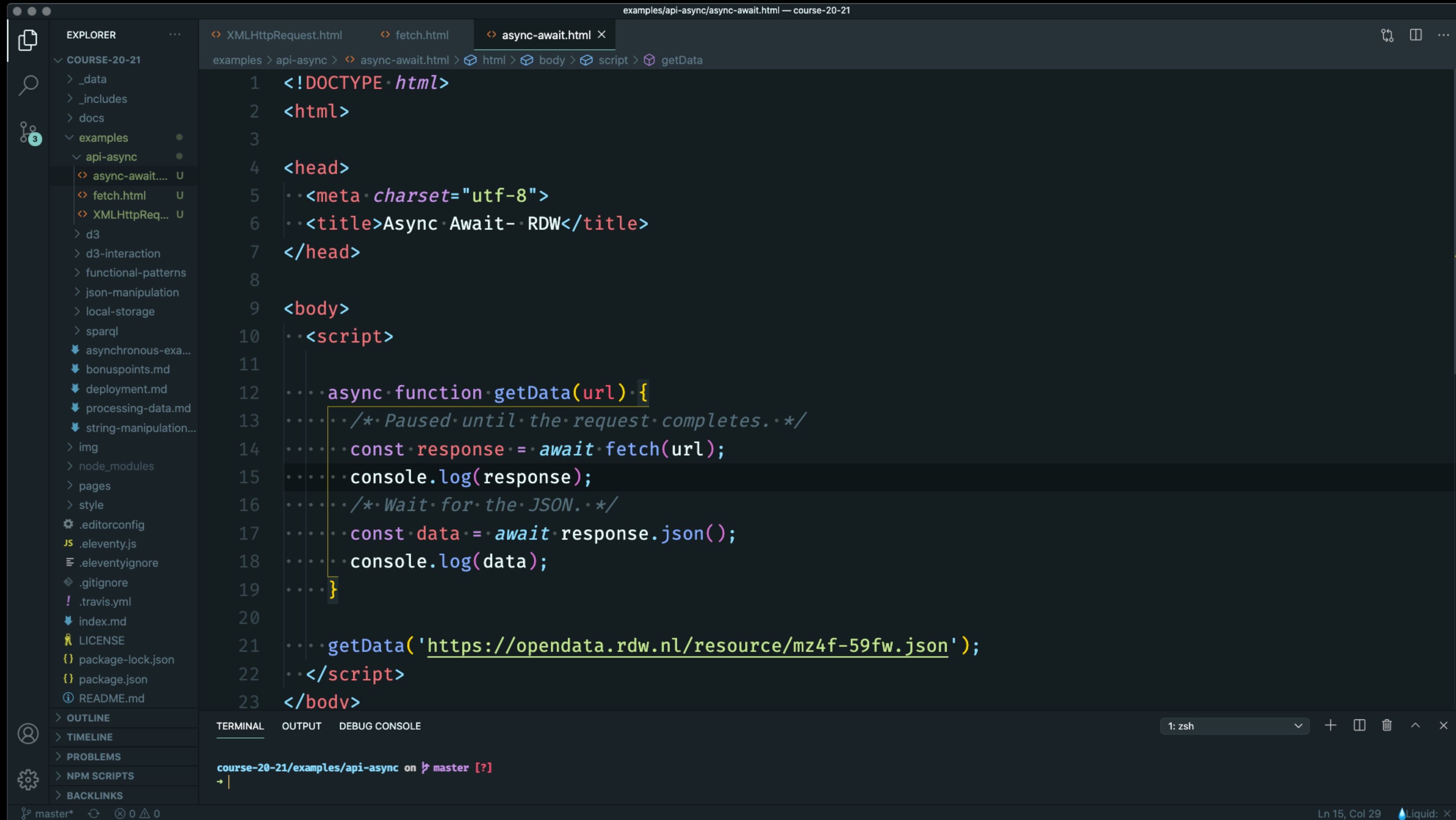
```
6  <title>Fetch - RDW</title>
7  </head>
8
9  <body>
10 <script>
11
12   const url = 'https://opendata.rdw.nl/resource/mz4f-59fw.json';
13
14   fetch(url)
15     .then(
16       function(response) {
17         /* Log output of response NOT data */
18         console.log(response);
19         /* .json resolves to a JavaScript object */
20         response.json().then(function(data) {
21           console.log(data);
22         });
23       }
24     )
25     .catch(function(err) {
26       console.log('Fetch Error', err);
27     });
28 </script>
29 </body>
30
31 </html>
```

TERMINAL OUTPUT:

```
course-20-21/examples/api-async on master [?]  
→
```

Fetch Example

Fetch (async / await)



The screenshot shows a VS Code editor window with the file `examples/api-async/async-await.html` open. The Explorer sidebar on the left shows the project structure, including a folder `examples` with subfolders `api-async`, `d3`, `d3-interaction`, `functional-patterns`, `json-manipulation`, `local-storage`, `sparql`, and `asynchronous-exa...`. The main editor area displays the following HTML code:

```
1 <!DOCTYPE html>
2 <html>
3
4 <head>
5   <meta charset="utf-8">
6   <title>Async Await - RDW</title>
7 </head>
8
9 <body>
10   <script>
11     async function getData(url) {
12       /* Paused until the request completes. */
13       const response = await fetch(url);
14       console.log(response);
15       /* Wait for the JSON. */
16       const data = await response.json();
17       console.log(data);
18     }
19
20     getData('https://opendata.rdw.nl/resource/mz4f-59fw.json');
21   </script>
22 </body>
```

The bottom of the editor shows a terminal window with the command prompt `course-20-21/examples/api-async on master [?]` and a cursor.

Async Await Example

Schedule

1. Review topics (q&A)
2. Fetch
- 3. Dataset research**
4. All together!



Dataset

We gaan aan de slag met het zoeken van een dataset die je wilt gaan visualiseren voor je individuele eindopdracht.

Kies een onderwerp wat je zelf interessant lijkt. Na de vakantie een **show en tell**.

Donderdag lever je een **concept idee** in.

Dataset kiezen

Maar belangrijker; een dataset waar je een **onderzoeksvraag uit kan halen.**

e.g. een patroon dat je kan visualiseren, een data vraag die je kan beantwoorden door vergelijken etc.

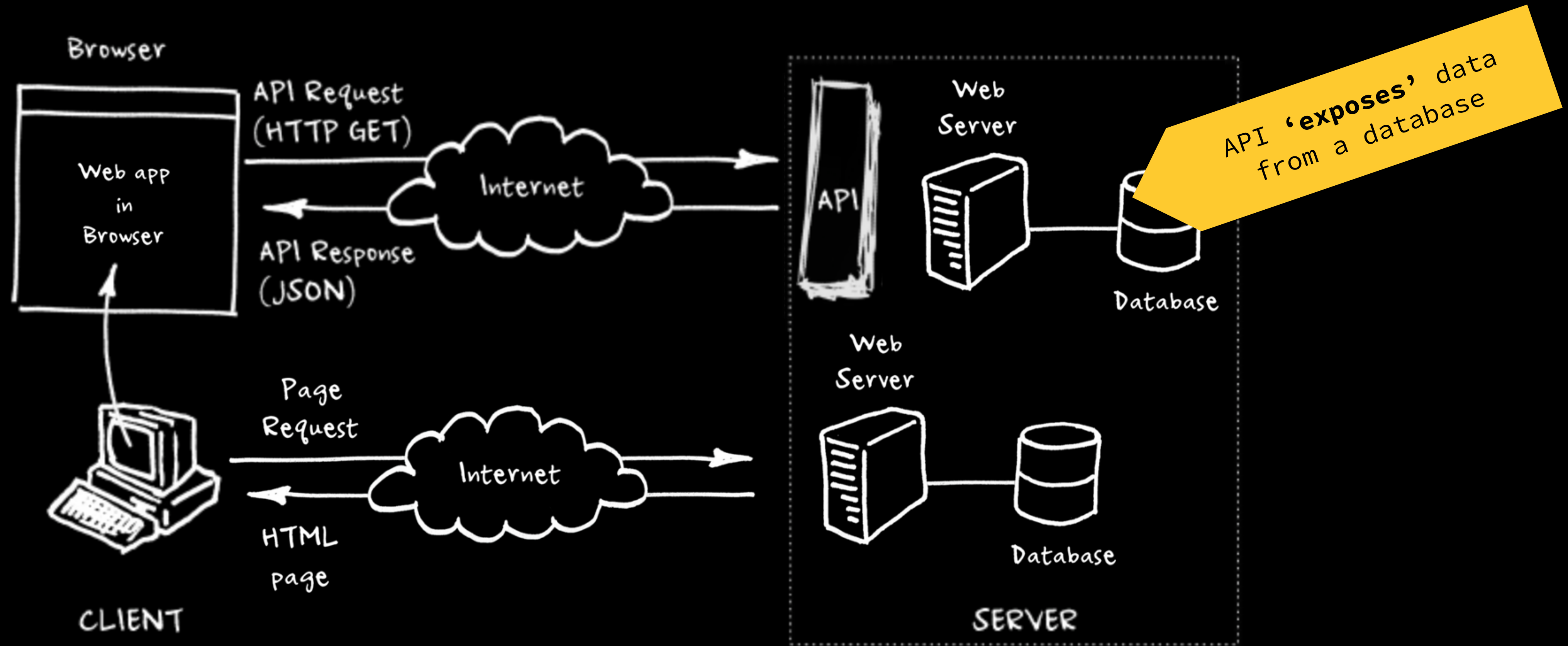
API

REST determines how the API looks like. It stands for “Representational State Transfer”. It is a set of rules that developers follow when they create their API.

Smashing – Understanding REST API's

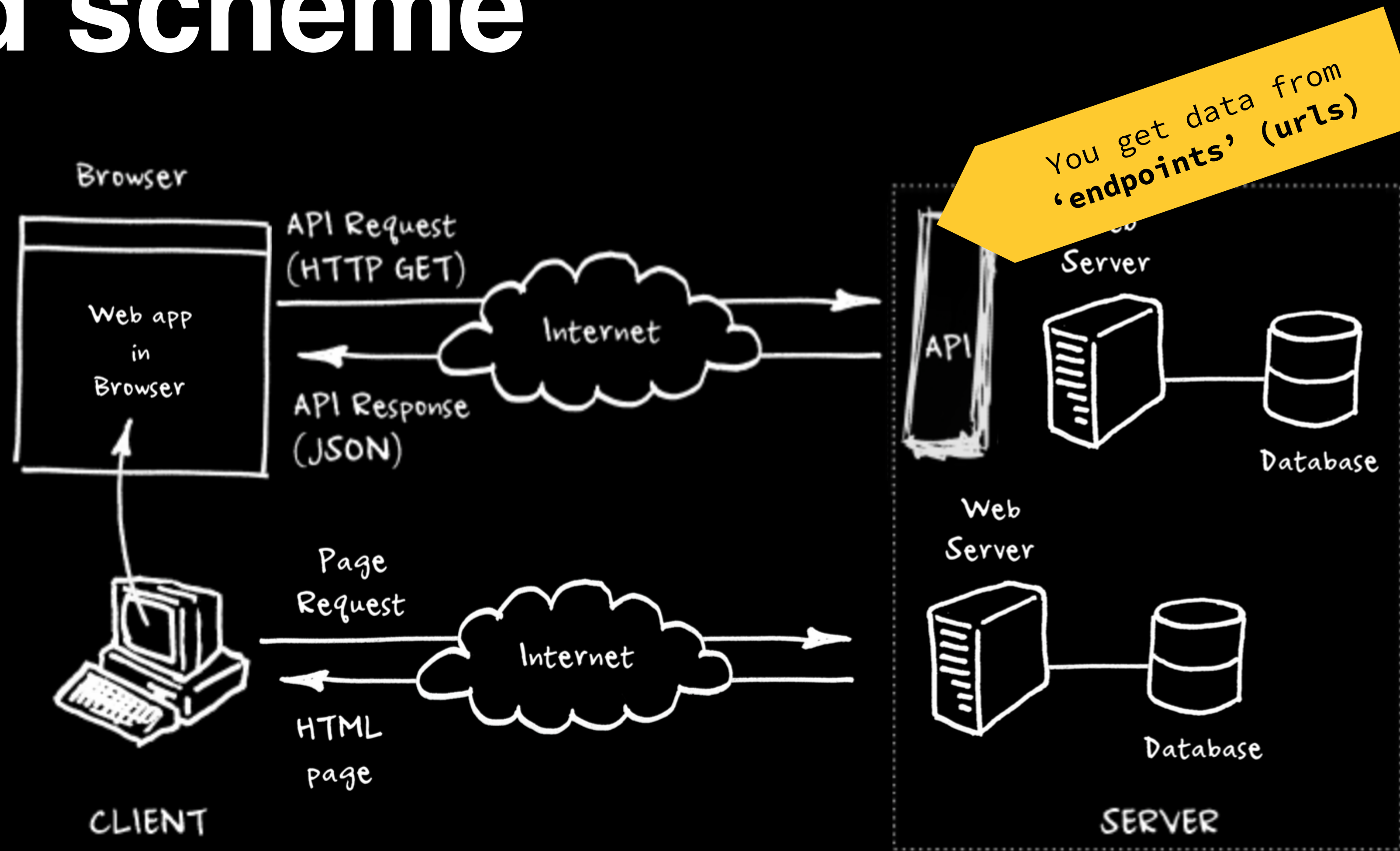
grand scheme

API



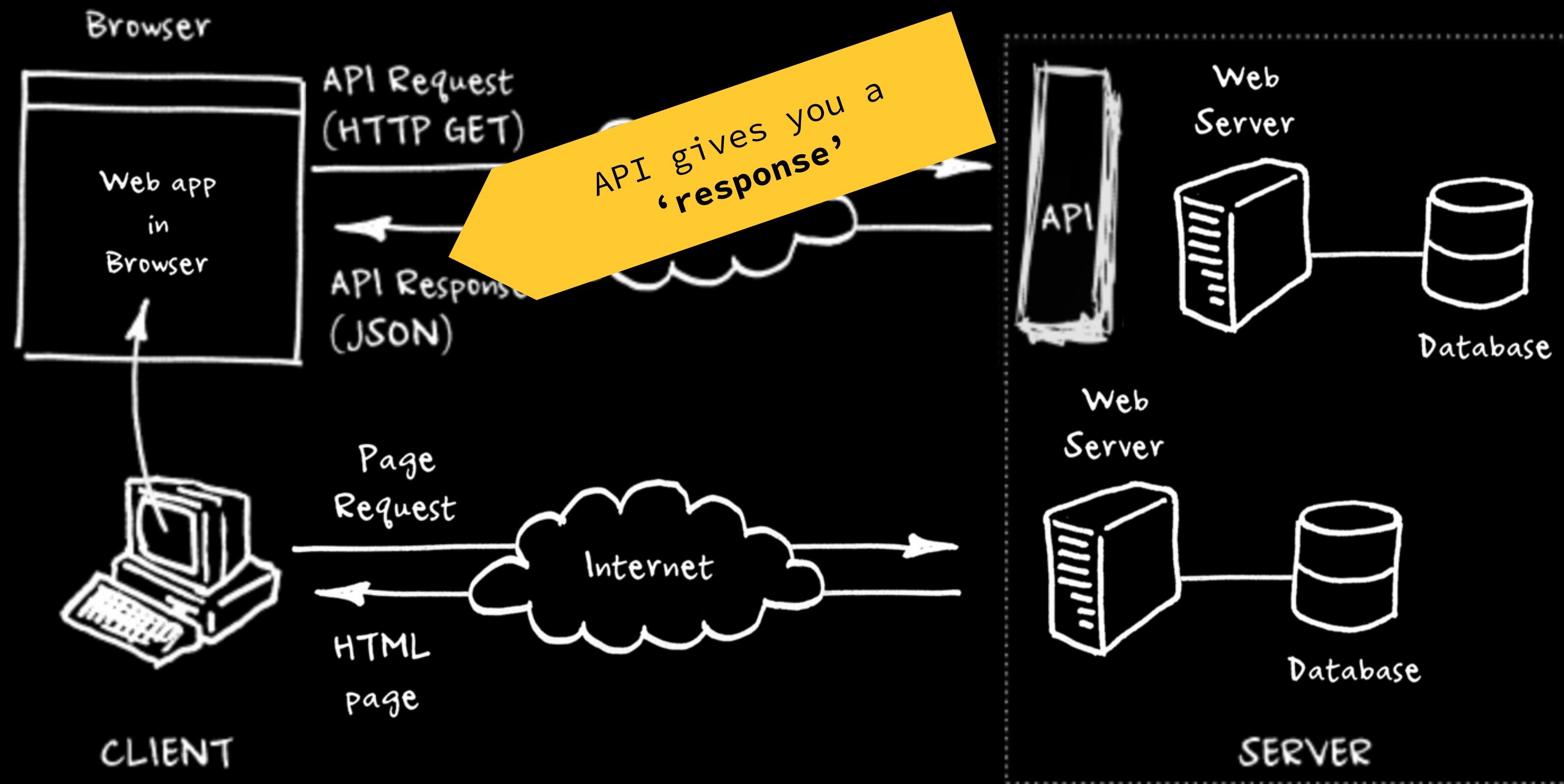
grand scheme

API



grand scheme

API



Open Data Parkeren: PARKEERGEBIED

Parkeren

Data bekijken

Visualiseren en ontdekken ▾

Exporteer

API

...

Deze tabel legt een koppeling tussen de gebieden zoals de de gebieden zoals deze voor Open Data Parkeren volgens gepubliceerd worden.

Betreffende deze dataset

Bijgewerkt

25 oktober 2020

Laatst bijgewerkt op
25 oktober 2020

Metadata laatst bijgewerkt op
25 oktober 2020

Gemaakt op
28 oktober 2014

Weergaves
4.800

Downloads
6.776

Data voorzien door
(geen)

Eigenaar dataset
Open data team RDW

Licentie

Licentie

Onderwerpen

Categorie

Parkeren

Tags

parkeergebied, parkeren

Krijg toegang tot deze Dataset via SODA API

De Socrata Open Data API (SODA) voorziet een programmatische toegang tot deze dataset en de mogelijkheid om gegevens te filteren, op te vragen en te combineren.

[API-documenten](#)

[Ontwikkelingsport](#)

Endpoint and data format

API-eindpunt

<https://opendata.rdw.nl/resource/mz4f-59>

JSON

Kopiëren

JSONRaw DataHeaders

SaveCopyCollapse AllExpand AllFilter JSON

▼ 0:

areamanagerid:"344"

areaid:"3300"

uuid:"57d6f361-ffea-4186-a5a0-80a122c06fc3"

▼ 1:

areamanagerid:"796"

areaid:"020411"

uuid:"b6a5905f-c79f-4669-a43a-68eccef1f3c3"

▼ 2:

areamanagerid:"202"

areaid:"BC2"

uuid:"58de2c3d-2c38-4bb4-b653-30e2c9b90363"

▼ 3:

areamanagerid:"344"

areaid:"B7200"

uuid:"eab21b04-bcea-4ff4-858e-eedd6a24e5b7"

▼ 4:

areamanagerid:"262"

areaid:"1"

uuid:"f51fa39b-dda9-4e06-8374-1f6791c8be7c"

▼ 5:

areamanagerid:"599"

areaid:"599_10"

uuid:"dca24e79-98d6-46e7-980d-b14b461cfdd8"

▼ 6:

areamanagerid:"826"

areaid:"2"

uuid:"63f645d1-c2c0-4042-809e-4a217a6f2445"

▼ 7:

areamanagerid:"200"

areaid:"PAS2"

uuid:"5c89ef00-b5a2-4601-82f5-38090bea1351"

▼ 8:

areamanagerid:"141"

Get data (.json) back
(body, headers etc.)

API (auths)

- ❖ **Public;** *open url* endpoint which can be freely ‘fetched’ (rate-limit).



[chucknorris.io](#) is a free JSON API for hand curated Chuck Norris facts. [Read more](#)

Subscribe for new Chuck Facts

USAGE

Retrieve a random chuck joke in JSON format.

```
GET https://api.chucknorris.io/jokes/random
```

Example response:

```
{
  "icon_url" : "https://assets.chucknorris.host/img/avatar/chuck-norris.png",
  "id" : "LBpfYFjlSfSfmw4cpJS6IA",
  "url" : "",
  "value" : "Chuck Norris can play slide guitar with a beer bottle. Or,
```


API (auths)

- ❖ **Public;** *open url* endpoint which can be freely ‘fetched’ (rate-limit).
- ❖ **apiKey;** *open url* endpoint which needs a ‘identifier key’ in the header or uri

OMDb API

The Open Movie Database

The OMDb API is a RESTful web service to obtain movie information, all content and images on the site are contributed and maintained by our users.

If you find this service useful, please consider making a [one-time donation](#) or [become a patron](#).



Poster API

The Poster API is only available to patrons.

Currently over 280,000 posters, updated daily with resolutions up to 2000x3000.

Attention Users

04/08/19 - Added support for eight digit IMDb IDs.

01/20/19 - Supressed adult content from search results.

01/20/19 - Added Swagger files ([YAML](#), [JSON](#)) to expose current API abilities and upcoming REST functions.

[Become a Patron](#)

Sponsors

[Emby](#), [Trakt](#), [FileBot](#), [Reelgood](#), [Xirvik Servers](#), [Yidio](#), [mi.tv](#), [Couchpop](#), [What's on Netflix](#), [Edu Reviewer](#), [Flixboss](#), [StreamingMoviesRight](#), [Scripts on Screen](#), [Writers Per Hour](#), [Medium.com](#), [Write my paper](#), [Ramotion.com](#), [Phone Trackers](#), [Property for sale in Lake Como](#), [iStarTips](#), [What A Room](#), [Vibelovely](#), [StreamToday](#), [Property for sale in Spain](#), [Top Casinos Reviews](#), [Classics on DVD](#), [Streaming App](#), [How to make a FinTech app](#), [Vid2 - Create movie lists with AI](#)

Usage

Send all data requests to:

`http://www.omdbapi.com/?apikey=[yourkey]&`

Poster API requests:

`http://img.omdbapi.com/?apikey=[yourkey]&`

`http://www.omdbapi.com/?apikey=[yourkey]`

Parameter

Unique Identifier

`http://www.omdbapi.com/?apikey=[yourkey]`

API (auths)

- ❖ **Public**; *open url* endpoint which can be freely ‘fetched’ (rate-limit).
- ❖ **apiKey**; *open url* endpoint which needs a ‘identifier key’ in the header or uri
- ❖ **oAuth**; authentication protocol with user identification before fetching

Web API

- Overview
- Getting started

Concepts

Access Token

API calls

Apps

Authorization

Playlists

Quota modes

Rate limits

Scopes

Spotify URIs and IDs

Track Relinking

Tutorials

Authorization code

Authorization code PKCE

Client credentials

Implicit grant

Refreshing tokens

How-Tos

REFERENCE

Albums

Artists

Audiobooks

Categories

Chapters

Episodes

Genres

Access Token

The *access token* is a string which contains the credentials and permissions that can be used to access a given resource (e.g artists, albums or tracks) or user's data (e.g your profile or your playlists).

To use the *access token* you must include the following header in your API calls:

Header Parameter	Value
Authorization	Valid access token following the format: Bearer <Access Token>

Note that the *access token* is valid for 1 hour (3600 seconds). After that time, the token expires and you need to request a new one.

Examples

The following example uses cURL to retrieve information about a track using the [Get a track](#) endpoint:

```
1 curl --request GET \
2   'https://api.spotify.com/v1/tracks/2TpxZ7JUBn3uw46aR7qd6V' \
3   --header "Authorization: Bearer NgCXRK...MzYjw"
```

The following code implements the `getProfile()` function which performs the API call to the [Get Current User's Profile](#) endpoint to retrieve the user profile related information:

```
1 async function getProfile(accessToken) {
2   let accessToken = localStorage.getItem('access_token');
3
4   const response = await fetch('https://api.spotify.com/v1/me', {
5     headers: {
6       Authorization: 'Bearer ' + accessToken
7     }
8   });
9
10  const data = await response.json();
11 }
```

Multi-fetch

Often API's uses **multiple endpoints** for different types of data (collections vs individual items) or **limit the amount of items** that get returned in each request (pagination).

Home

Object metadata

API

Harvest

Download

Bibliographic data

Controlled vocabularies

User-generated content

Object metadata / API

Object metadata APIs

The object metadata APIs make the power of the award-winning [Rijksmuseum website](#) directly accessible to developers. [Searching the collection](#) through the API offers a wide range of interesting possibilities, as do the [tiled images](#) used to zoom in to close-ups of objects. The JSON-based service is so easy to use that you can create an application using the Rijksmuseum's rich and [freely accessible content](#) in no time.

Access to APIs

To start using the data and images, you first need to obtain an API key by registering for a [Rijksstudio account](#). You will be given a key instantly upon request, which you can find at the advanced settings of your Rijksstudio account.

Collection API

`GET /api/[culture]/collection` gives access to the collection with brief information about each object. The results are split up in result pages. By using the `p` and `ps` parameters you can fetch more results, up to a total of 10,000. All of the other parameters are identical to the [advanced search page](#) on the Rijksmuseum website. You can use that page to find out what's the best query to use.

Parameter	Format	Default	Notes
<code>key</code>	<code>a-z 0-9</code>		Your API-key , mandatory for every request.
<code>format</code>	<code>json</code> / <code>jsonp</code> / <code>xml</code>	<code>json</code>	The format of the result.
<code>culture</code>	<code>nl</code> / <code>en</code>		The language to search in (and of the results).
<code>p</code>	<code>0-n</code>	<code>0</code>	The result page. Note that <code>p * ps</code> cannot exceed 10,000.
<code>ps</code>	<code>1-100</code>	<code>10</code>	The number of results per page.
<code>q</code>	<code>a-z</code>		The search terms that need to occur in one of the fields of the object data.
<code>involvedMaker</code>	<code>a-z</code>		Object needs to be made by this agent.
<code>type</code>	<code>a-z</code>		The type of the object.
<code>material</code>	<code>a-z</code>		The material of the object.
<code>technique</code>	<code>a-z</code>		The technique used to make the object.
<code>f.dating.period</code>	<code>0-21</code>		The century in which the object is made.

https://data.rijksmuseum.nl/object-metadata/api/

Dataset eisen

- Je hebt een dataset die **enigszins dynamisch** is
- Je hebt een dataset die **enigszins groot** is (+10.000 punten)
- Je hebt een dataset met **query parameters en filter opties**
- Je hebt een dataset die met **gangbare formaten (.json)** werkt
- Je hebt een dataset die **blijft werken tot het einde** 🙄

API overzichten

- <https://rapidapi.com/>
- <https://publicapis.dev/>
- <https://github.com/public-apis/public-apis>
- <https://www.kaggle.com/>

API instances

- Gemeente: <https://data.amsterdam.nl/>
- Musea: <http://data.rijksmuseum.nl/>
- Overheden: <https://data.gov/>

Schedule

1. Review topics (q&A)
2. Fetch
3. Dataset research
- 4. All together!**



Beschrijf in je wiki

- Ga opzoek naar een goede dataset.
- Waarom deze dataset? Waarom dit onderwerp?
- Wat is je onderzoeksvraag bij de dataset?
- Hoe is de documentatie van de API?
- Is de dataset dynamisch en up-to-date?
- Welke formaten geeft de API terug qua data?
- Welke visualisatie zie je voor je?

**Uncaught SyntaxError
Unexpected end of input**