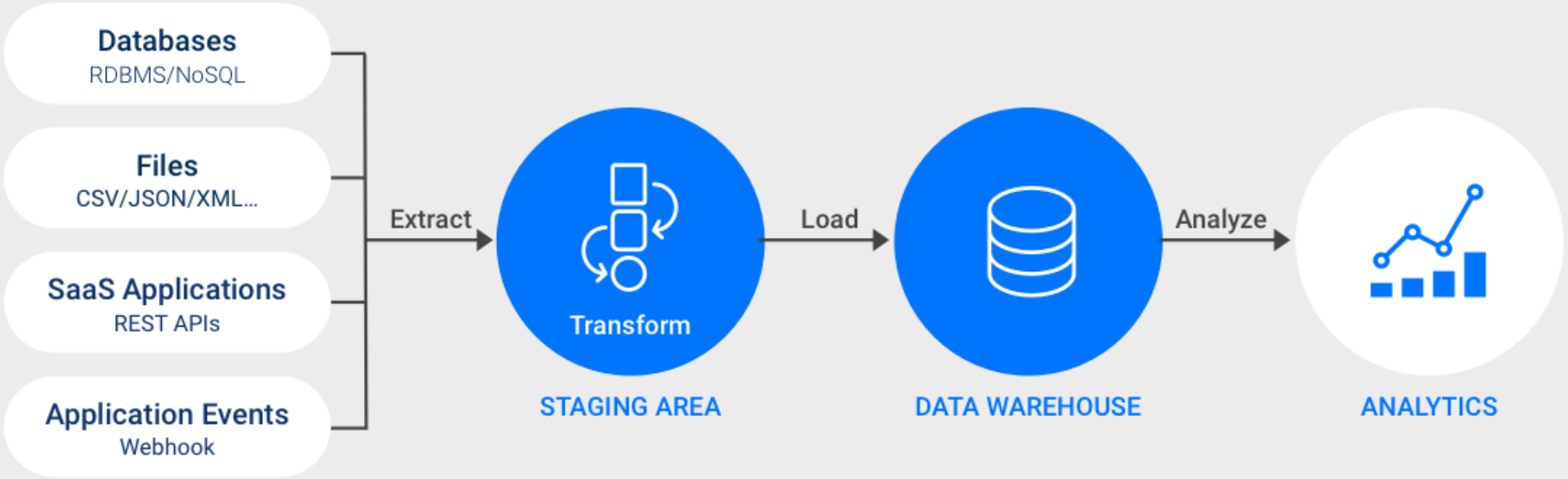
Extracting Data using APIs

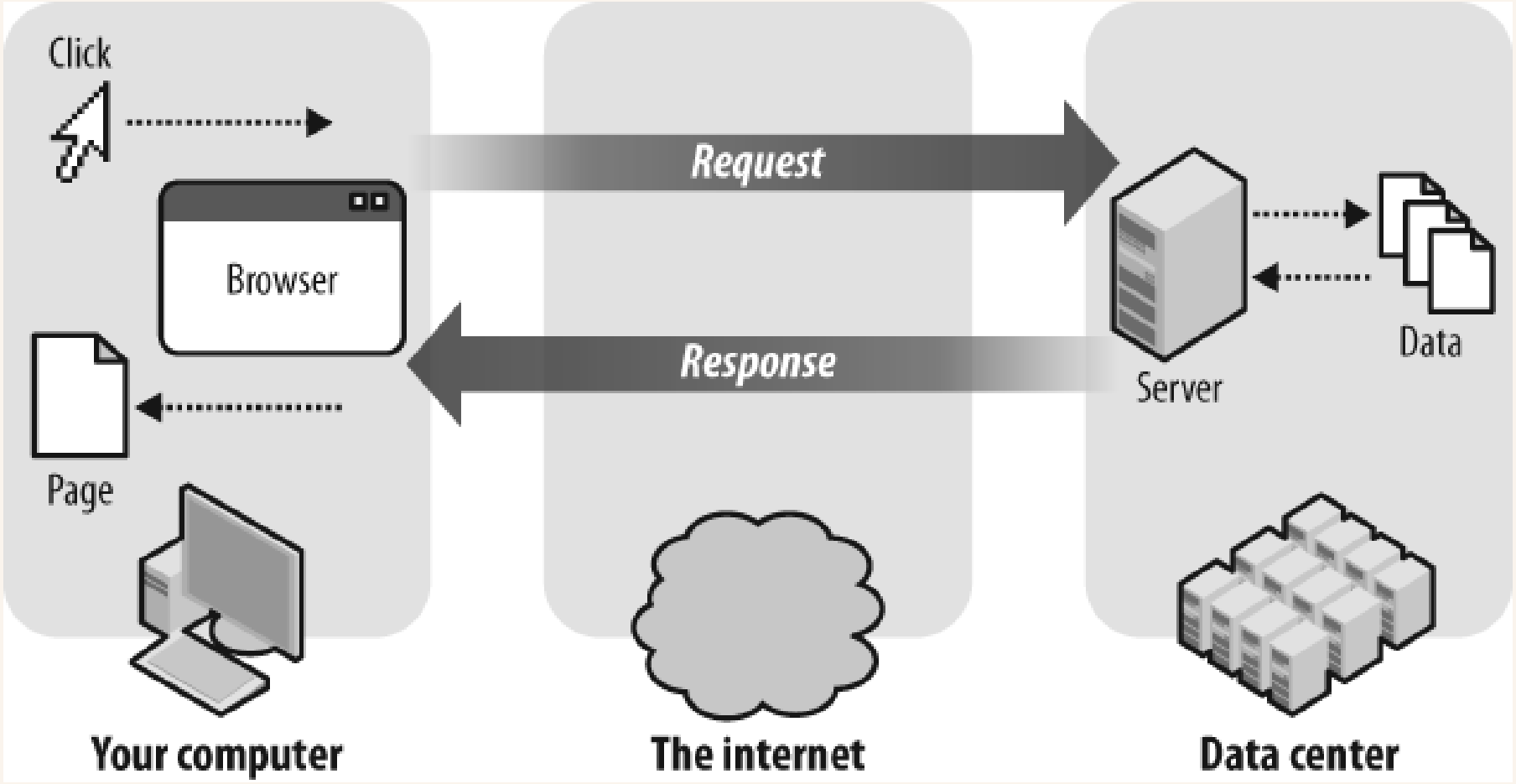
ETL PROCESS


Databases
RDBMS/NoSQL
Files
CSV/JSON/XML...
SaaS Applications
REST APIs
Application Events
Webhook
Extract
Transform
STAGING AREA
Load
DATA WAREHOUSE
Analyze
ANALYTICS

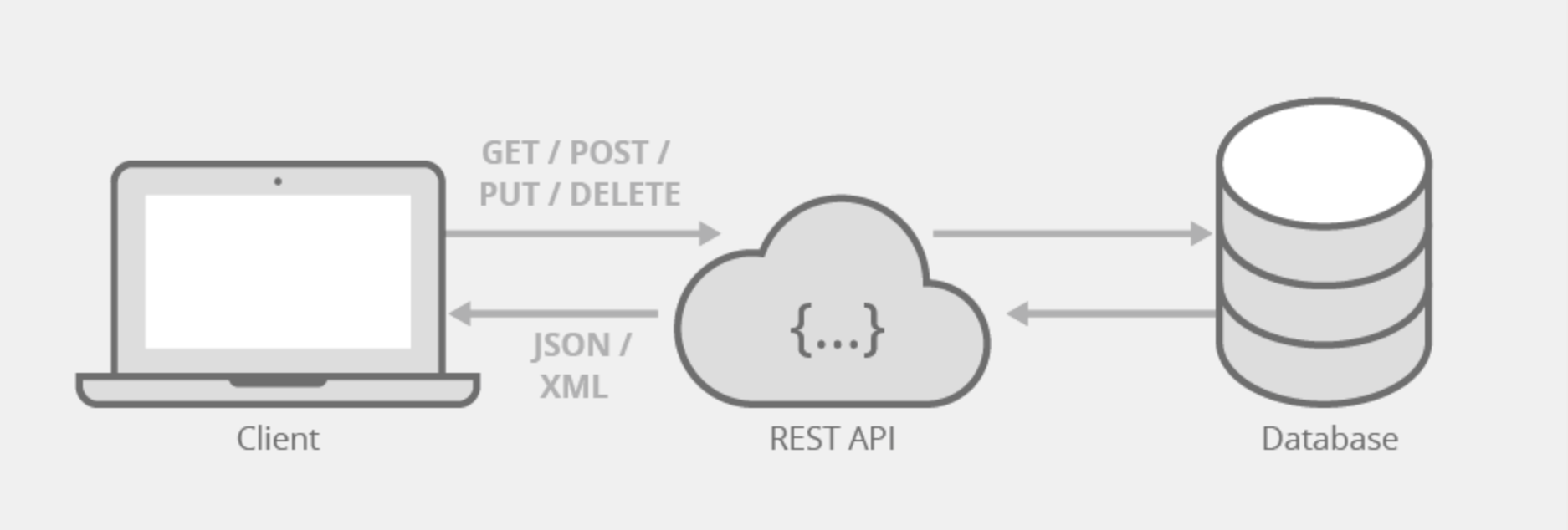
Extract data from source

- Manually download files
- Extract data using SQL
- **Extract data using APIs**
  - **A**pplication **P**rogramming **I**nterface

How does your browser work?



APIs to handle requests and responses



request: A request is made (over the internet) to the API endpoint with specific parameters.

response: The API processes the request and sends back a response, usually in JSON.

---

https://www.mongodb.com/resources/basics/what-is-an-api
https://factor-bytes.com/2023/03/11/request-response-in-web-applications-how-it-works/

API example: iTunes Search API

Web API that allows you to search for movies, music, apps, etc on iTunes.

---

https://developer.apple.com/library/archive/documentation/AudioVideo/Conceptual/iTuneSearchAPI/index.html

API request example

```
https://itunes.apple.com/search?entity=tvSeason&term=bigbang+theory&limit=1
```

API response example

In JSON (JavaScript Object Notation)


```
{
 "resultCount":1,
 "results": [
  {"wrapperType":"collection",
  "collectionType":"TV Season",
  "artistId":264032462,
  "collectionId":1271833178,
  "artistName":"The Big Bang Theory",
  "collectionName":"The Big Bang Theory, Season 11",
  ...
  }]
}
```

REST API syntax - API endpoint

API request:

`https://itunes.apple.com/search?entity=tvSeason&term=bigbang+theory&limit=1`

Endpoint: Base URL + Path

`https://itunes.apple.com/search`

- Base URL: `https://itunes.apple.com`
- Path: `/search`

---

https://www.ibm.com/docs/en/informix-servers/12.10?topic=api-rest-syntax

REST API syntax - Query parameters

API request:

`https://itunes.apple.com/search?entity=tvSeason&term=bigbang+theory&limit=1`

Query parameters:

`?entity=tvSeason&term=bigbang+theory&limit=1`

- `?` : indicates the start of query parameters
- `&` : separates each key-value pair
  - `entity=tvSeason`
  - `term=bigbang+theory`
  - `limit=1`

Understanding API documentation

- **API endpoint**
  - Base URL
  - Path
- **Query parameters**

iTunes Search API documentation

https://developer.apple.com/library/archive/documentation/AudioVideo/Conceptual/iTuneSearchAPI/Searching.html#//apple_ref/doc/uid/TP40017632-CH5-SW1

- Base URL: `https://itunes.apple.com`
- Path & Query parameters
  - `search`
    - `term`
    - `country`
    - `entity`
    - `attribute`

      ...

iTunes Search API documentation

- `https://itunes.apple.com/search?entity=musicArtist&term=billie+eilish&limit=1`
- `https://itunes.apple.com/search?entity=podcast&term=the+economist&limit=3`
- `https://itunes.apple.com/search?entity=tvSeason&term=Comedy&limit=5`

🖥️ Open Brewery DB

https://www.openbrewerydb.org

Read the documentation to identify:

- Base URL
- Path
- Query parameters

🖥️ Open Brewery DB

1. Find breweries in Detroit, Michigan
2. Find micro breweries in California
3. Search for breweries with "dog" in the name
4. Get five random breweries

Make `requests` programmatically

`requests` **library provides functions to make HTTP requests:**

- `requests.get()` : **get data from a URL**
- `requests.post()` : post data to a URL
- `requests.put()` : update data on a URL
- `requests.delete()` : delete data from a URL

`import` module (library/package)

`math`: a Python built-in module that provides mathematical functions and constants

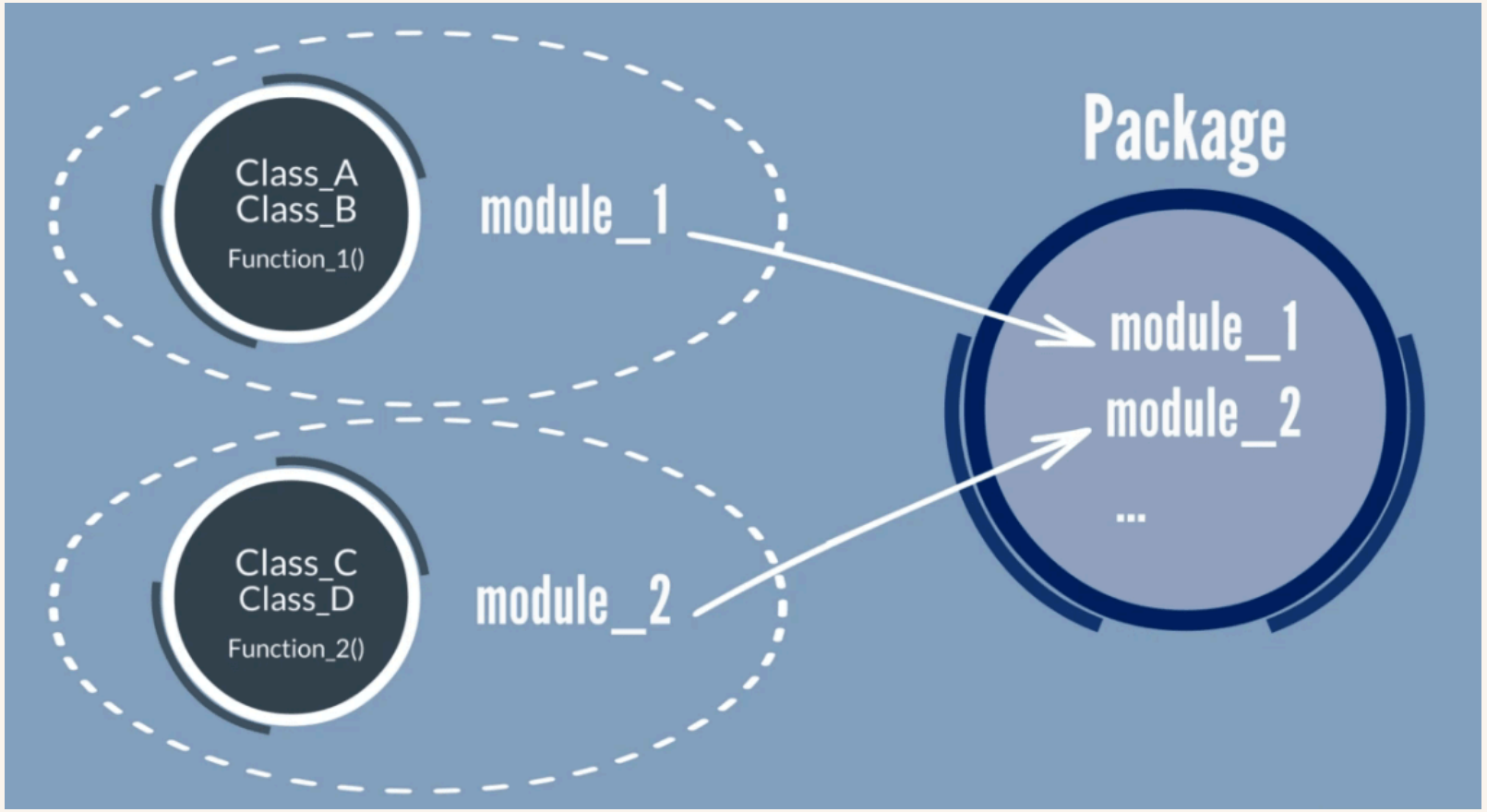
```
import math

print(math.pi)              # 3.141592653589793
print(math.sqrt(4))         # 2.0
print(math.pow(2, 3))       # 8.0
print(math.floor(3.14))     # 3
print(math.ceil(3.14))      # 4
print(math.factorial(5))    # 120
```

---

https://docs.python.org/3/library/math.html

functions $\subset$ modules $\subset$ packages $\approx$ libraries



Types of packages

Built-in functions: No need to `import`

- `print()`, `type()`, `len()`, `range()`, `input()`

Built-in modules: `import` required

- `math`, `random`, `datetime`, `os`, `sys`

Third-party packages: download and `import` required

- `requests`, `pandas`, `numpy`, `nltk`, `sqlalchemy`
- `pip install package_name` to download and install from PyPI (https://pypi.org/)

`requests.get()`

Makes a GET request to an API endpoint

Returns a `Response` object

```
# Must import requests before using it
import requests

# HTTP GET request to iTunes Search API
url = "itunes.apple.com/search?entity=tvSeason&term=bigbang+theory&limit=1"

# Make the GET request
response = requests.get(url)
```

`Response` object

Contains information about the response from the server.

```
print(response)                # <Response [200]>
print(response.status_code)    # 200
print(response.text)           # raw text of the response
print(response.json())         # convert raw text to python dictionary
```

`requests.get()` dynamically

Build URL with variables

- Store query parameter values in variables
- Concatenate variables to build URL

```
entity = "tvSeason"
term = "bigbang theory"
limit = 1

url = f"itunes.apple.com/search?entity={entity}&term={term}&limit={limit}"

response = requests.get(url)
```

`requests.get()` dynamically with `params`

Better way to build URL with parameters argument

- Store query parameters in a dictionary
- Pass dictionary to `requests.get()` using `params` argument

```
url = "https://itunes.apple.com/search"
params = {
    "entity": "tvSeason",
    "term": "bigbang theory",
    "limit": 1
}
# Pass params dictionary to requests.get()
response = requests.get(url, params=params)
```

💻 `requests.get()` dynamically with `params` - Open Brewery DB

Open Brewery DB Documentation: https://www.openbrewerydb.org

Make a request to the Open Brewery DB API

1. Find breweries in Detroit, Michigan
2. Find micro breweries in California
3. Search for breweries with "dog" in the name
4. Get five random breweries

Print each response in json format.

Accessing items in JSON response

JSON = nested structure of lists, tuples, and dicts

- list/tuple: `[index]`
- dict: `[key]`

🤔 Accessing nested data structures

```
cities = [
    {"name": "Montreal", "state": "QC", "country": "CA"},
    {"name": "Toronto", "state": "ON", "country": "CA"},
    {"name": "Vancouver", "state": "BC", "country": "CA"},
    {"name": "Detroit", "state": "MI", "country": "US"}
]
```

- `{"name": "Vancouver", "state": "BC", "country": "CA"}`
- `"Vancouver"`
- `"Montreal"`, `"Toronto"`, `"Vancouver"`, `"Detroit"`

```
cities = {
    "location": {
        "Montreal": {"state": "QC", "country": "CA"},
        "Toronto": {"state": "ON", "country": "CA"},
        ...
    },
    "stats": {
        "Montreal": [
            {"year": 2013, "population": 2000000, "area": 431.5},
            {"year": 2014, "population": 1980000, "area": 431.5},
            ...
            ],
        "Toronto": [
            {"year": 2013, "population": 2800000, "area": 630.2},
            ],
        ...
    }
}
```

- `{"year": 2013, "population": 2000000, "area": 431.5}`
- Montreal population data: `2000000`, `1980000`, ...

💻 `requests.get()` dynamically with `params` - Open Brewery DB

https://www.openbrewerydb.org

1. Make a request to the Open Brewery DB API to find micro breweries in California (Same as above)
2. Print the name and address (`address_1`) of each brewery.