What is an API?

Intro to API’s

Assembler
School of Software Engineering

Assembler
School of Software Engineering

Contents

What is an API?

What is a Web API?

Client-Server Architecture

Data transfer formats

AJAX

REST

How to communicate with an API using JavaScript

API

Application Programming Interface

What is an API?

An API is an interface that we can use to interact with a computer, a web page, a software program or a web server.

It specifies the types of interactions that we can make, the types of data that we can send or receive or the URL we need to call to communicate with the API.

Assembler
School of Software Engineering

Examples of API’s

An example of a hardware API is the thermostat in your home which you can use to control the temperature.

Browser API’s

API’s can also be software based, like the browser based API’s you have used so far to interact with the storage API or the timers API.

```
// Browser Storage API
let storage = localStorage.getItem("storage");

// JSON API
let data = JSON.stringify(storage);

// Browser Timers API
setTimeout(function () {
 console.log(data);
}, 2000);

// Browser Console API
console.log(data);
```

Web API's

A Web API is an API that can be used to allow communication between devices or programs so that they can transfer data and information using the HTTP protocol.

Communication is usually made using HTTP endpoints which are URL's that the programs can send requests to and wait for a response once the request has been processed.

Example of a request made to a Web API endpoint

CLIENT

```
axios.get("https://www.recipes.io/api");
```

SERVER

```
"Content-Type": "application/json"
...
[
 {
   "name": "Fruit salad",
   "description": "Great recipe...",
   "difficulty": "Easy"
 }
]
```

Why do we need API’s?

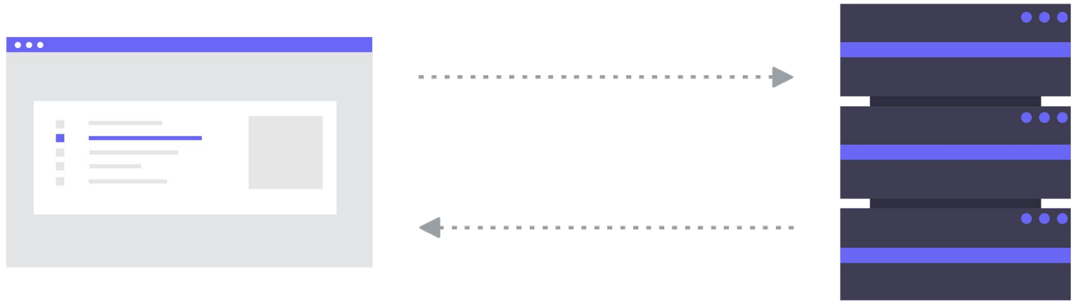
- Store information permanently in a database
- Recover stored information from a database
- Load stored information in different devices
- Cloud storage and servers
- Get data dynamically from a server
- Get data from multiple servers
- Authentication and user profiles
- Shopping carts
- ...

How do API's communicate with each other

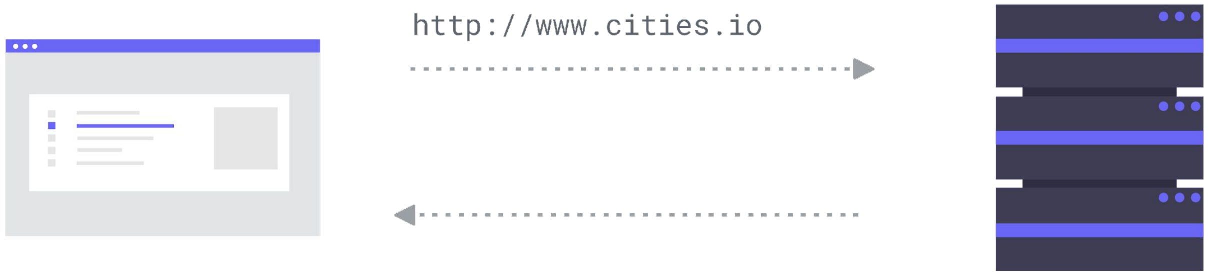
Client-Server Architecture

Client-Server Architecture

```
GET /recipes
```



Requesting an HTML page



```
content-type: text/html;
charset=UTF-8
<!DOCTYPE html>
<html>
    <head>
        <meta charSet="utf-8"/>
    </head>
. . .
<body>
  <p>. . .
```

Data transfer formats

Assembler
School of Software Engineering

MIME Types

A media type (also known as a Multipurpose Internet Mail Extensions or MIME type) is a standard that indicates the nature and format of a document or file.

It helps browsers understand the type of data they receive to know how to interpret the information.

content-type header

Browsers use the MIME type, not the file extension, to determine how to process an URL, so it's important that web servers send the correct MIME type in the response's Content-Type header.

If this is not correctly configured, browsers are likely to misinterpret the contents of files and sites will not work correctly, and downloaded files may be mishandled.

Structure of a MIME type

The ***type*** represents the general category of content such as text or video, and the ***subtype*** represents the exact type of data.

For sending JSON data, the MIME type would be: `application/json`

And for sending an HTML document it would be: `text/html`

type/subtype

text / html

text / plain

font / woff

image / jpeg

image / png

application / json

video / mp4

Requesting JSON data

CLIENT

```
axios.get("https://www.cities.io/api");
```

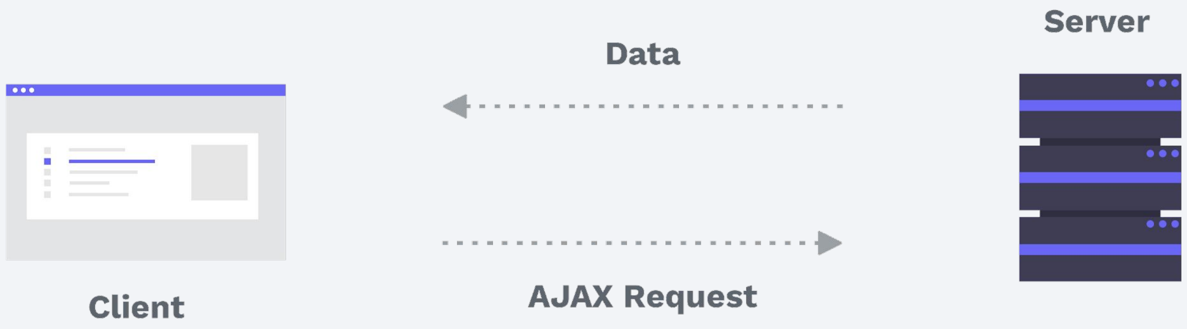
SERVER

```
"Content-Type": "application/json"
...
[
 {
   "name": "Barcelona",
   "population": 1620343,
   "postalCode": 08000
 }
]
```

Asynchronous JavaScript and XML

AJAX

What is AJAX



A set of web development techniques using web technologies on the client side to create asynchronous web applications.

With Ajax, web applications can send and retrieve data from a server asynchronously (in the background) without interfering with the display and behavior of the existing page.

Live Demo

Communicating with a Server

HTTP request example to API's endpoint

https://jsonplaceholder.typicode.com/users

https://reqres.in/api/users?page=2

Assembler
School of Software Engineering

AJAX Methods

The webpage can then be modified by JavaScript to dynamically display the new information. The data is usually fetched using the built-in `XMLHttpRequest` object, the more modern `fetch()` function or by other libraries like jQuery or Axios.

Representational State Transfer

REST

REST Endpoint

REST Endpoints are the URL we need to connect to in order to communicate with an API.

```
// Get all the posts
https://jsonplaceholder.typicode.com/api/posts
```

REST API Calls

In this example we are making an HTTP REST request to an URL to fetch all the posts from the server which will return a JSON array of data.

```
// Get all the posts
https://jsonplaceholder.typicode.com/posts
[
 {
   "userId": 1,
   "id": 1,
   "title": "sunt aut facere repellat provident",
   "body": "quia et recusandae consequuntur"
 },
 {
   "userId": 1,
   "id": 2,
   "title": "qui est esse",
   "body": "est rerum tempore nihil reprehenderit"
 }
]
```

Endpoints HTTP

```
https://www.recipes.io
```

Get all the recipes

GET /recipes

```
https://www.recipes.io/recipes
```

Get the user cart

GET /user/cart

```
https://www.recipes.io/user/cart
```

Delete the comment of a recipe

DELETE /recipes/:id/comments/:id

```
https://www.recipes.io/recipes/12/comments/23
```

Assembler
School of Software Engineering

HTTP Verbs

HTTP Verbs are used to indicate the type of action we want to perform at a particular endpoint.

`GET` `POST` `PUT` `PATCH` `DELETE`

HTTP Verbs

Get all the recipes

```
GET /recipes
```

Create a new recipe

```
POST /recipes
```

Update the recipe that has the id

```
PATCH /recipes/:id
```

Delete the recipe that has the id

```
DELETE /recipes/:id
```

Create/Update a new recipe at this location

```
PUT /recipes/recipe_1
```

Status Codes

Status codes are used to indicate the result of the operation and whether it was ok or it failed.

https://http.cat/

200 - OK
201 - Created
400 - Bad Request
401 - Unauthorized
403 - Forbidden
404 - Not Found
500 - Internal Server Error

Status Codes

A request to the following REST endpoint returns a status code of 200 when making a GET request to fetch all the posts.

```
Request URL: https://jsonplaceholder.typicode.com/posts
Request Method: GET
Status Code: 200 OK
```

Assembler
School of Software Engineering

ASYNCHRONOUS COMMUNICATION IN JAVASCRIPT

How to communicate with an API using JavaScript

`$.ajax(), fetch(), axios()`

Assembler
School of Software Engineering

$.ajax()

Assembler
School of Software Engineering

$.ajax()

The jQuery ajax() allows us to make networks requests and to send data to a server in a way that is much simpler than using the previous XMLHttpRequest object.

This method also handles the conversion by default of the response we get from the API by reading the MIME type in the response and converting the data.

```
$.ajax("https://www.blog.com/posts", {
   method: "GET"
 }).then(
   function success(data, statusText, jqXHR) {
     console.log(statusText);
     console.log(data);
   },
   function failed(jqXHR, errorStatusText, errorMessage) {
     console.log(jqXHR);
     console.log(errorStatusText);
     console.log(errorMessage);
   }
 );
```

Assembler
School of Software Engineering

$.ajax()

The jQuery ajax() method can be used in multiple ways which allows the developer to choose the shape that is more convenient.

```
$.ajax({
   url: "https://www.blog.com/posts",
   method: "GET",
   success: function (data, statusText, jqXHR) {
     console.log(data);
     console.log(statusText);
     console.log(jqXHR);
   },
   error: function (jqXHR, errorStatusText, errorMessage) {
     console.log(jqXHR);
     console.log(errorStatusText);
     console.log(errorMessage);
   },
 });
```

Assembler
School of Software Engineering

Sending data to a server

The jQuery ajax() method can be used to send data to a server by adding the data we need to send to the data property and by specifying the appropriate content type type.

```
$.ajax("https://www.blog.com/posts", {
   method: "POST",
   contentType: "application/json",
   data: JSON.stringify({
     title: "My first blog post",
     body: "This is my first blog post"
   }),
 }).then(
   function success(data, statusText, jqXHR) {
     console.log(statusText);
     console.log(data);
   },
   function failed(jqXHR, errorStatusText, errorMessage) {
     console.log(jqXHR);
     console.log(errorStatusText);
     console.log(errorMessage);
   }
 );
```

Assembler
School of Software Engineering

Sending an update request

The jQuery ajax() method can also be used to send PATCH, DELETE or PUT requests by specifying the id of the content that we need to update or remove in the request url and the data in the request body (data property).

```
$.ajax("https://www.blog.com/posts/1", {
   method: "PATCH",
   contentType: "application/json",
   data: JSON.stringify({
     body: "How to send data with jQuery",
   }),
 }).then(
   function success(data, statusText, jqXHR) {
     console.log(statusText);
     console.log(data);
   },
   function failed(jqXHR, errorStatusText, errorMessage) {
     console.log(jqXHR);
     console.log(errorStatusText);
     console.log(errorMessage);
   }
 );
```

Assembler
School of Software Engineering

fetch()

Assembler
School of Software Engineering

fetch()

Alternatively the more modern fetch() method can be used to make networks requests.

It can be used similarly to the $.ajax() method but without having to use an external library since it is part of almost all modern browsers.

```
fetch("https://www.blog.com/posts", {
   method: "GET",
 })
   .then(function (response) {
     // JSON data returned from the server
     // The response object has several properties
     // we can use to see the status of the request
     // response.ok: true;
     // response.status: 200;
     // response.statusText: "";
     return response.json();
   })
   .then(function (json) {
    // Data returned from the server
     console.log(json);
   });
```

Sending data using fetch()

The fetch() method can also be used to make networks requests that update or remove content in an API.

```
fetch("https://www.blog.com/posts/1", {
   method: "PATCH",
   headers: {
     "Content-type": "application/json; charset=UTF-8",
   },
   body: JSON.stringify({
     title: "How to send data with jQuery",
   }),
 })
   .then(function (response) {
     return response.json();
   })
   .then(function (json) {
     console.log(json);
   });
```

axios()

Assembler
School of Software Engineering

axios()

Another modern alternative is the axios library which can be used similarly to the $.ajax() and fetch() methods but it automatically converts JSON data and has more browser support that the native fetch() method.

```
// Load the axios library in the browser
<script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>
```

Assembler
School of Software Engineering

Using axios()

Example of using the library to make a GET request to an API.

```
axios({
  url: "http://www.restapiexample.com/api",
  method: "GET", // default
}).then(function (response) {
  // The response is already converted from JSON
  console.log(response);

  // The response object has the following properties:
  // The response that was provided by the server
  // data: {},

  // The HTTP status code from the server response
  // status: 200,

  // The HTTP status message from the server response
  // statusText: 'OK',
  // ...
});
```

Assembler
School of Software Engineering

Sending data with axios()

Example of using the library to make a POST request to an API to create content.

```
axios({
   url: "http://dummy.restapiexample.com/api/v1/create",
   method: "POST",
   body: JSON.stringify({
     firstname: "Alex",
     lastname: "Mark",
     salary: "123",
     age: "23",
   }),
 }).then(function (response) {
   console.log(response);
 });
```