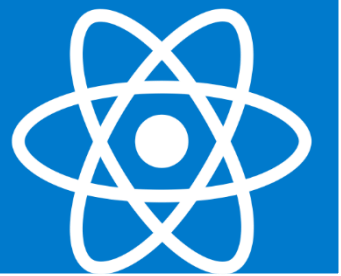
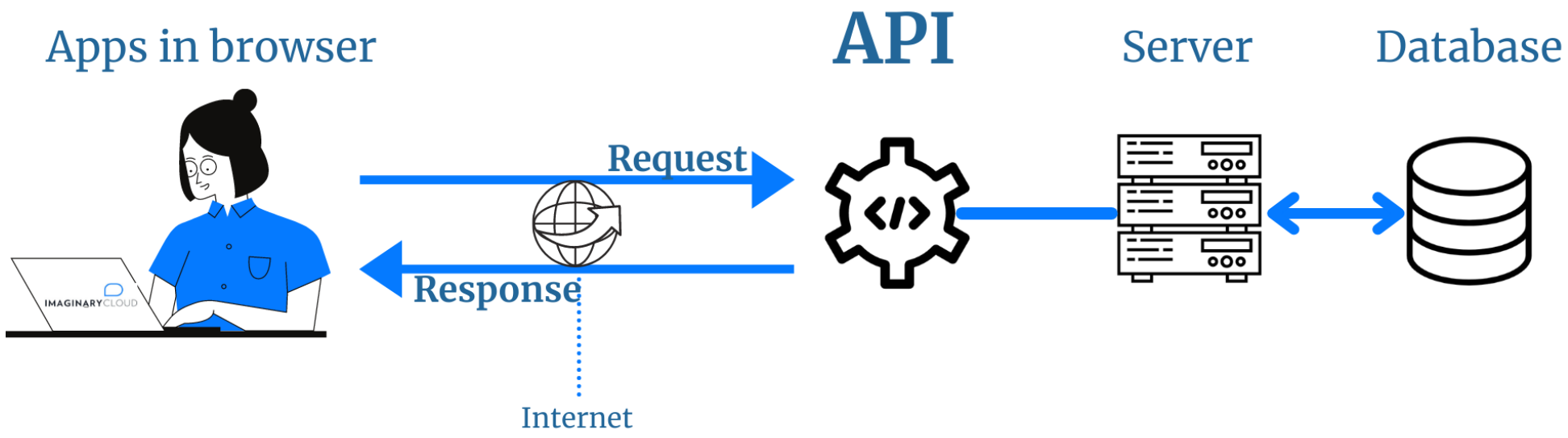
React JS

#React Notes

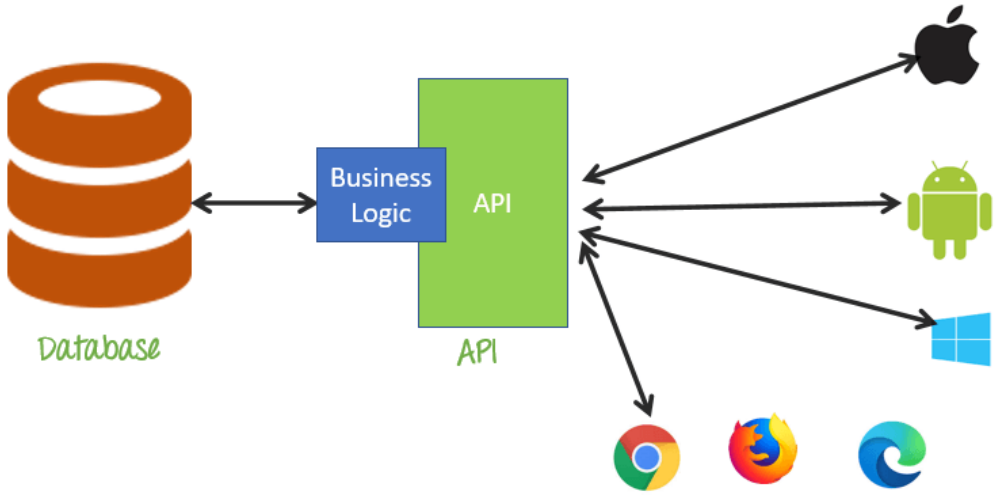
API

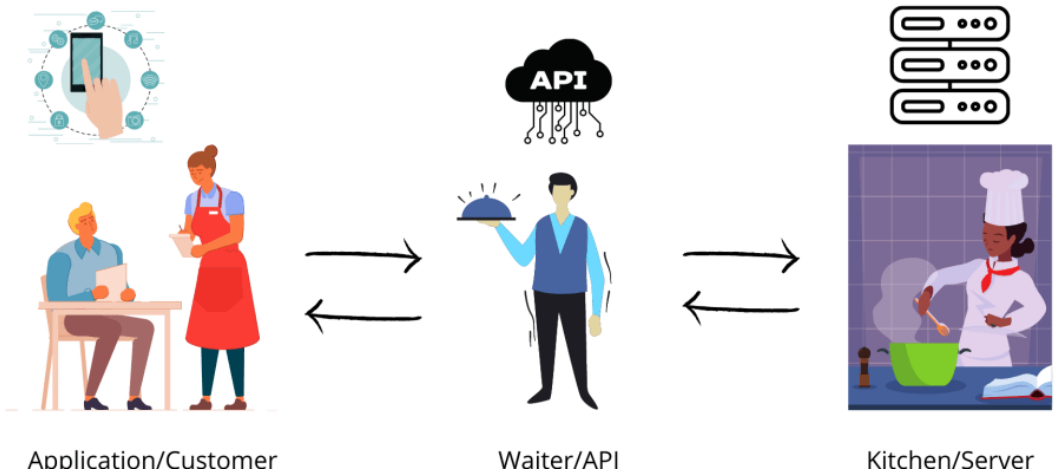


What is API ?


Apps in browser
API
Server
Database
Request
Response
Internet
IMAGINARYCLOUD

How API Works



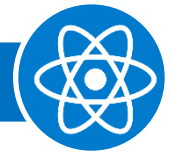


Tools to Test API

- PostMan
  - https://www.postman.com/ (Desktop)
  - https://web.postman.co/ (Web)
- ThunderClient
  - https://www.thunderclient.com/ (VS Code Extension)
- insomnia
  - https://insomnia.rest/
- Firecamp
  - https://firecamp.io/

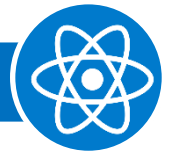
Mock API

- https://jsonplaceholder.typicode.com/
- https://fakestoreapi.com/
- http://dummyjson.com/
- https://fakejson.com/
- https://reqres.in/
- https://dummyapi.io/explorer
- https://gorest.co.in/
- https://www.mockaroo.com/
- http://dummy.restapiexample.com/

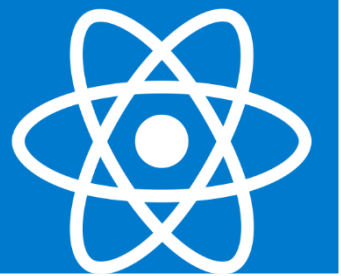


Partner API

- https://newsapi.org/
- https://openweathermap.org/api
- https://spoonacular.com/food-api



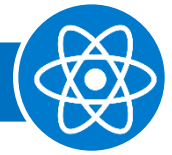
API Testing



IMP

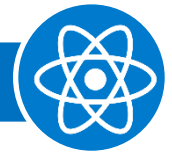
- Ask Question ?
  - Where is API URL ?
  - Which method is used in API ? (GET/POST)
  - Is there any Parameter ?
  - Is there any Authtentication ?

- CheckList?
  - 1-Test API in PostMan
  - 2-Check Status Code
  - 3-Check Response in Postman as JSON
  - 4-Copy paste response in jsonlint and validate it. (Optional)
    - https://jsonlint.com/

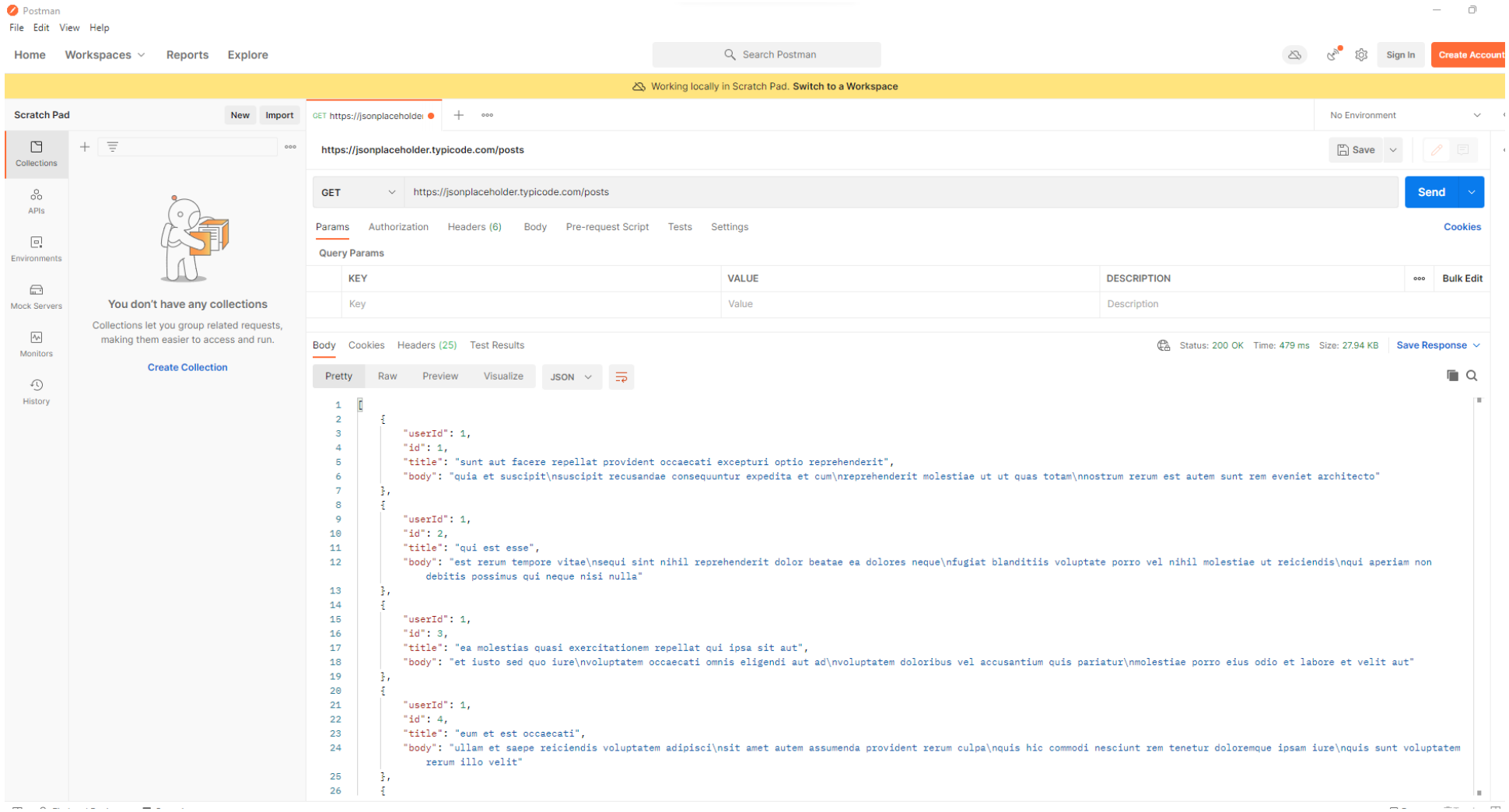


API URL

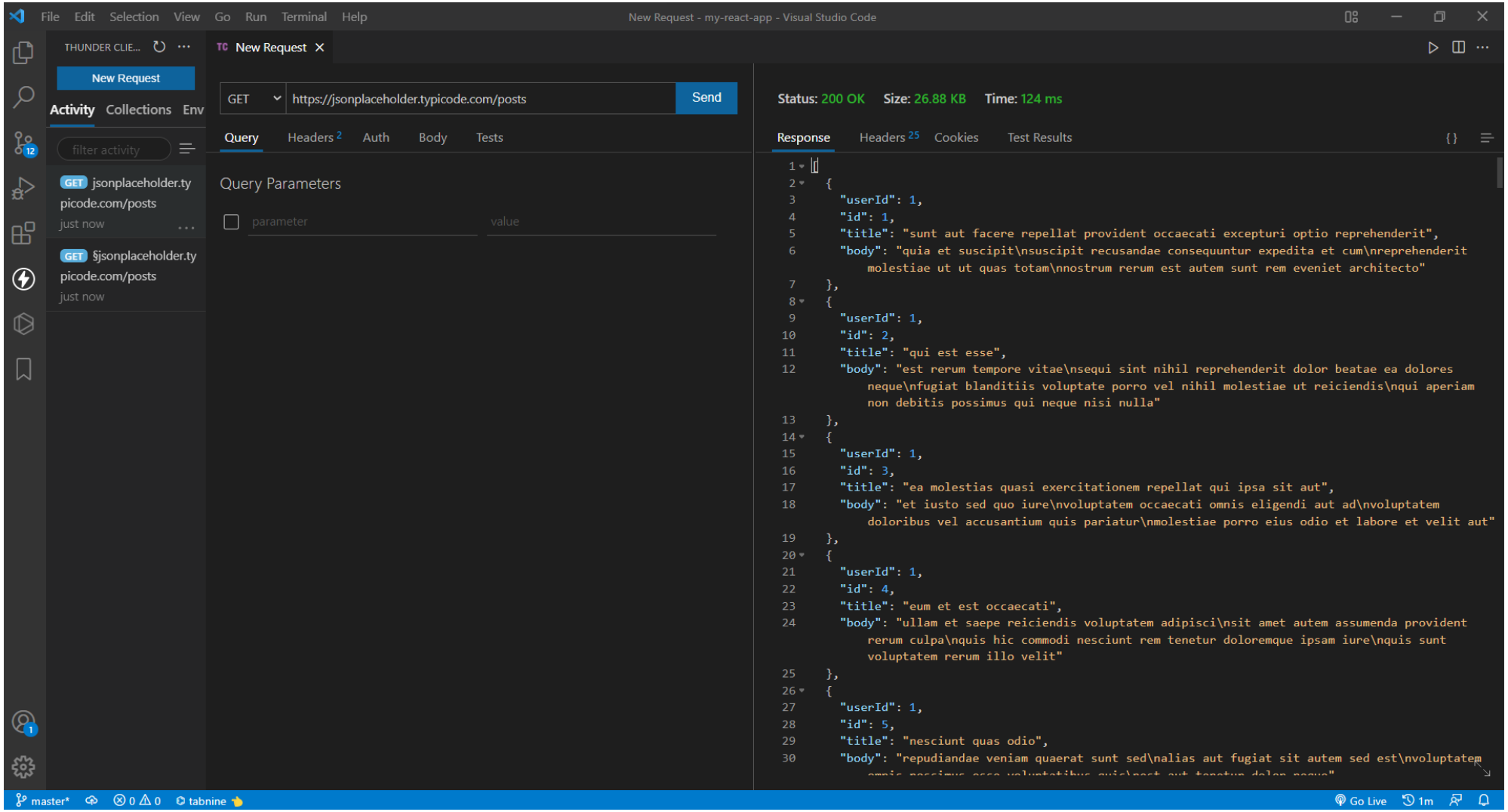
- Method : Get
- Use : List Details
- URL : https://jsonplaceholder.typicode.com/posts



Post Man

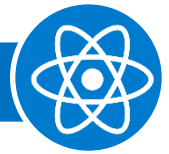

Postman
File Edit View Help
Home Workspaces Reports Explore
Search Postman
Sign In
Create Account
Working locally in Scratch Pad. Switch to a Workspace
Scratch Pad
New
Import
Collections
APIs
Environments
Mock Servers
Monitors
History
You don't have any collections
Collections let you group related requests, making them easier to access and run.
Create Collection
GET https://jsonplacholder
No Environment
https://jsonplaceholder.typicode.com/posts
Save
GET
https://jsonplaceholder.typicode.com/posts
Send
Params Authorization Headers (6) Body Pre-request Script Tests Settings
Cookies
Query Params
KEY VALUE DESCRIPTION Bulk Edit
Key Value Description
Body Cookies Headers (25) Test Results
Status: 200 OK Time: 479 ms Size: 27.94 KB
Save Response
Pretty Raw Preview Visualize JSON
[
{
"userId": 1,
"id": 1,
"title": "sunt aut facere repellat provident occaecati excepturi optio reprehenderit",
"body": "quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit molestiae ut ut quas totam\nnostrum rerum est autem sunt rem eveniet architecto"
},
{
"userId": 1,
"id": 2,
"title": "qui est esse",
"body": "est rerum tempore vitae\nsequi sint nihil reprehenderit dolor beatae ea dolores neque\nfugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis\nqui aperiam non debitis possimus qui neque nisi nulla"
},
{
"userId": 1,
"id": 3,
"title": "ea molestias quasi exercitationem repellat qui ipsa sit aut",
"body": "et iusto sed quo iure\nvoluptatem occaecati omnis eligendi aut ad\nvoluptatem doloribus vel accusantium quis pariatur\nmolestiae porro eius odio et labore et velit aut"
},
{
"userId": 1,
"id": 4,
"title": "eum et est occaecati",
"body": "ullam et saepe reiciendis voluptatem adipisci\nsit amet autem assumenda provident rerum culpa\nquis hic commodi nesciunt rem tenetur doloremque ipsam iure\nquis sunt voluptatem rerum illo velit"
},
{

Thunder Client

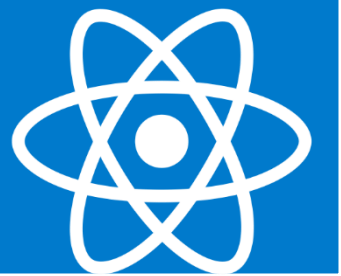


API Call

- We can call API Using 2 Methods
  - JS Fetch Method
  - Axios NPM

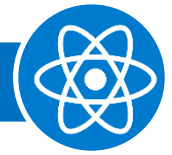


Fetch



Fetch

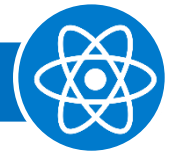
- The Fetch API is a promise-based API for making HTTP requests, Browsers come with a global fetch() function that we can use to make requests.
- The Fetch API can make GET, POST, PUT, PATCH, DELETE and other kinds of HTTP requests. We'll focus on two of the most common methods used in HTTP requests: GET and POST.



Get Syntax :

- In the first line we use the global fetch() function to send a GET request to our API. The argument of fetch() is the URL with the server-side resource.
- We then chain the promise with the then() method, which captures the HTTP response in the response argument and call its json() method. The json() method parses the response body to a JSON object. However, it returns that as a promise.
- That's we why use then() once again to chain another promise, which logs the parsed JSON to the console.

```
fetch('https://jsonplaceholder.typicode.com/posts/1')
   .then(response => response.json())
   .then(json => console.log(json));
```



https://jsonplaceholder.typicode

jsonplaceholder.typicode.com/users


```
// 20220315083309
// https://jsonplaceholder.typicode.com/users

[
  {
    "id": 1,
    "name": "Leanne Graham",
    "username": "Bret",
    "email": "Sincere@april.biz",
    "address": {
      "street": "Kulas Light",
      "suite": "Apt. 556",
      "city": "Gwenborough",
      "zipcode": "92998-3874",
      "geo": {
        "lat": "-37.3159",
        "lng": "81.1496"
      }
    },
    "phone": "1-770-736-8031 x56442",
    "website": "hildegard.org",
    "company": {
      "name": "Romaguera-Crona",
      "catchPhrase": "Multi-layered client-server neural-net",
      "bs": "harness real-time e-markets"
    }
  },
  {
    "id": 2,
    "name": "Ervin Howell",
    "username": "Antonette",
    "email": "Shanna@melissa.tv",
    "address": {
      "street": "Victor Plains",
      "suite": "Suite 879",
      "city": "Wisokyburgh",
      "zipcode": "90566-7771",
      "geo": {
        "lat": "-43.9509",
        "lng": "-34.4618"
```

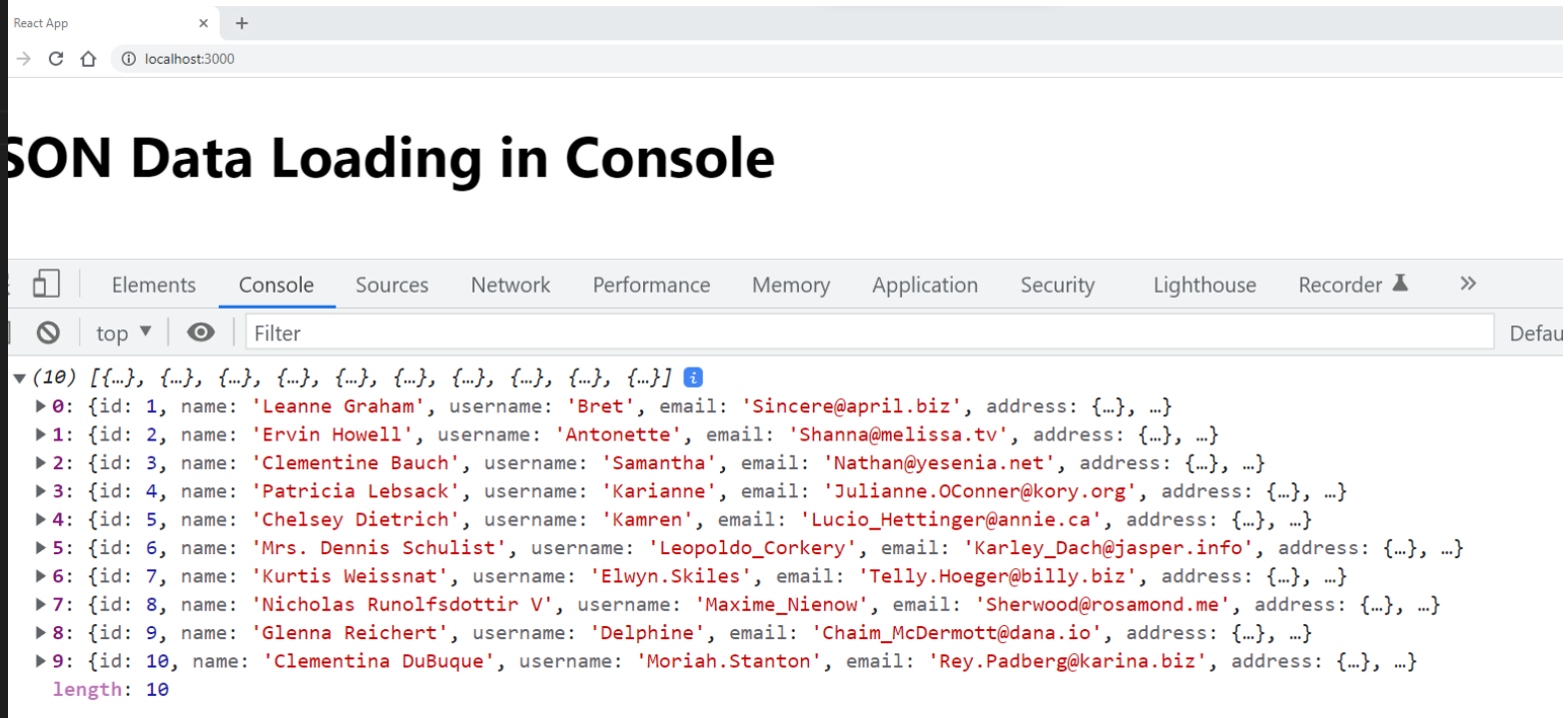
Fetch API Call & Print Response in Console.

```
componentDidMount() {
  fetch('https://jsonplaceholder.typicode.com/users')
    .then(response => response.json())
    .then(response => {
      console.log(response);
    })
}
```

App.js M

src > App.js > App > componentDidMount

```
1  import React from 'react';
2  class App extends React.Component {
3    constructor(props) {
4      super(props);
5      this.state = {  };
6    }
7
8    componentDidMount() {
9      fetch('https://jsonplaceholder.typicode.com/users')
10       .then(response => response.json())
11       .then(response => {
12         console.log(response);
13       })
14   }
15
16   render() {
17     return (
18       <>
19       <h1>JSON Data Loading in Console</h1>
20       </>
21     );
22   }
23 }
24
```



Console Table

```
componentDidMount(){
    fetch('https://jsonplaceholder.typicode.com/posts')
    .then(response => response.json())
    .then(response => {
        console.table(response);
    })
}
```

Elements | Console | Sources | Network | Performance | Memory | Application | Security | Lighthouse | Recorder | Components | Profiler

top | Filter | Default levels | 1 Issue: 1

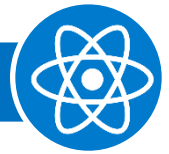
[webpack-dev-server] Server started: Hot Module Replacement enabled, Live Reloading enabled, Progress disabled, Overlay enabled. index.js:551

Display.js:12

(index)	userId	id	title	body
0	1	1	'sunt aut facere repellat provident occaecati excepturi optio reprehenderit'	'quia et suscipit\nsuscipit recusandae consequuntur …strum …
1	1	2	'qui est esse'	'est rerum tempore vitae\nsequi sint nihil reprehend…aperia…
2	1	3	'ea molestias quasi exercitationem repellat qui ipsa sit aut'	'et iusto sed quo iure\nvoluptatem occaecati omnis e…\nmole…
3	1	4	'eum et est occaecati'	'ullam et saepe reiciendis voluptatem adipisci\nsit … ipsam…
4	1	5	'nesciunt quas odio'	'repudiandae veniam quaerat sunt sed\nalias aut fugi…sse vo…
5	1	6	'dolorem eum magni eos aperiam quia'	'ut aspernatur corporis harum nihil quis provident …s\nvolu…
6	1	7	'magnam facilis autem'	'dolore placeat quibusdam ea quo vitae\nmagni quis e…t exce…
7	1	8	'dolorem dolore est ipsam'	'dignissimos aperiam dolorem qui eum\nfacilis quibus…\nipsa…
8	1	9	'nesciunt iure omnis dolorem tempora et accusantium'	'consectetur animi nesciunt iure dolore\nenim quia a…st aut…
9	1	10	'optio molestias id quia eum'	'quo et expedita modi cum officia vel magni\ndoloribus…it\nqu…
10	2	11	'et ea vero quia laudantium autem'	'delectus reiciendis molestiae occaecati non minima…luptate…
11	2	12	'in quibusdam tempore odit est dolorem'	'itaque id aut magnam\npraesentium quia et ea odit e…uia id…
12	2	13	'dolorum ut in voluptas mollitia et saepe quo animi'	'aut dicta possimus sint mollitia voluptas commodi …ssumend…
13	2	14	'voluptatem eligendi optio'	'fuga et accusamus dolorum perferendis illo volupta…molesti…
14	2	15	'eveniet quod temporibus'	'reprehenderit quos placeat\nvelit minima officia do…usanda…
15	2	16	'sint suscipit perspiciatis velit dolorum rerum ipsa laboriosam odio'	'suscipit nam nisi quo aperiam aut\nasperiores eos f… magni…

Print Response in React

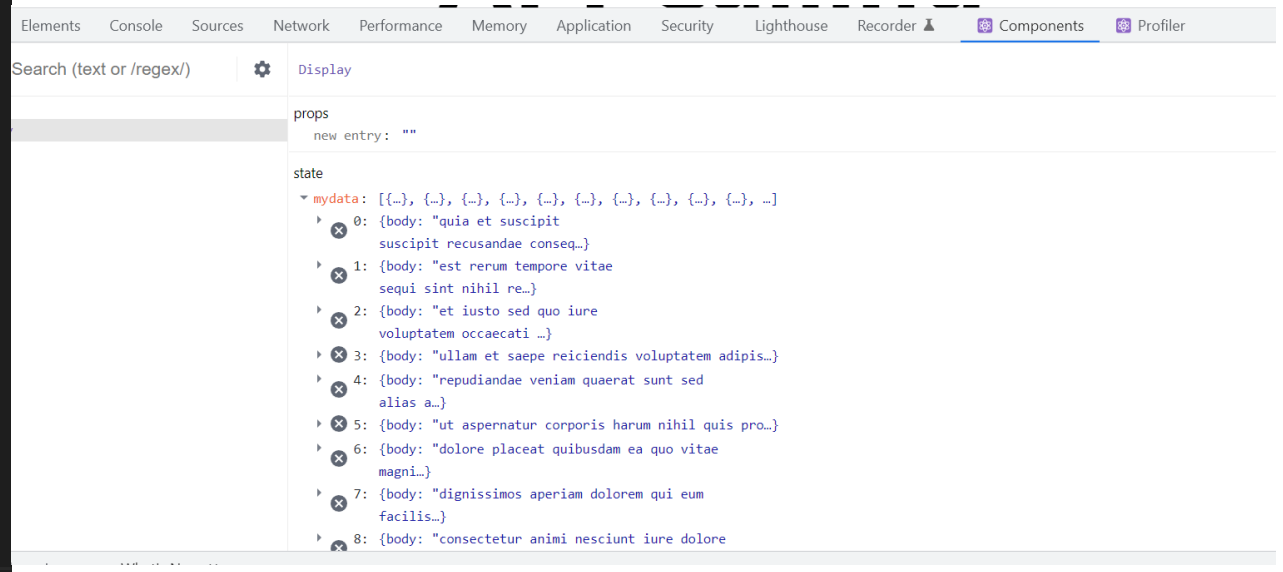
- Convert Data into JSON to String
- {JSON.stringify(this.state.map.arrayname)}
- {console.log(this.state.map.arrayname)}



Update Data in State

- Define Blank Array in State
- After Loading JSON Data Assign it to Array using State.

```
import React from 'react';
class Display extends React.Component {
    constructor(props) {
        super(props);
        this.state = {
            mydata : [],
        };
    }
    componentDidMount(){
        fetch('https://jsonplaceholder.typicode.com/posts')
        .then(response => response.json())
        .then(response => {
            console.table(response);
            this.setState({mydata: response});
        })
    }
```



JSON Stringify

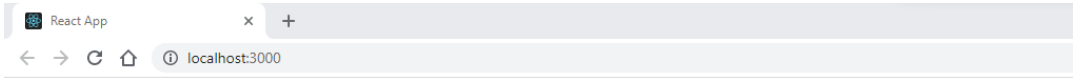
App.js M

src > App.js > default

```
1  import React from 'react'
2  class App extends React.Component {
3      constructor(props) {
4          super(props);
5          this.state = { mydata:[]  };
6      }
7      componentDidMount() {
8          fetch('https://jsonplaceholder.typicode.com/users')
9            .then(response => response.json())
10           .then(response => {
11             console.log(response);
12             this.setState({mydata:response});
13           })
14       }
15     render() {
16         return (
17             <div>
18                 {JSON.stringify(this.state.mydata)}
19             </div>
20         );
21     }
22 }
23 export default App;
```

JSON Api Parsing

```
import React from 'react'
class App extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      mydata : [] //Blank Array
    };
  }
  componentDidMount(){
    fetch('https://jsonplaceholder.typicode.com/users')
    .then(response => response.json())
    .then(response =>{
      this.setState({mydata: response}); //JSON Response assign in Array
    });
  }
  render() {
    return (
      <div>
        Api Call <br/>
        {/* Print Array Data using Map */}
        {this.state.mydata.map(values => (
          <li>Name : {values.name} | Email :{values.email} </li>
        ))}
      </div>
    );
  }
}
export default App;
```



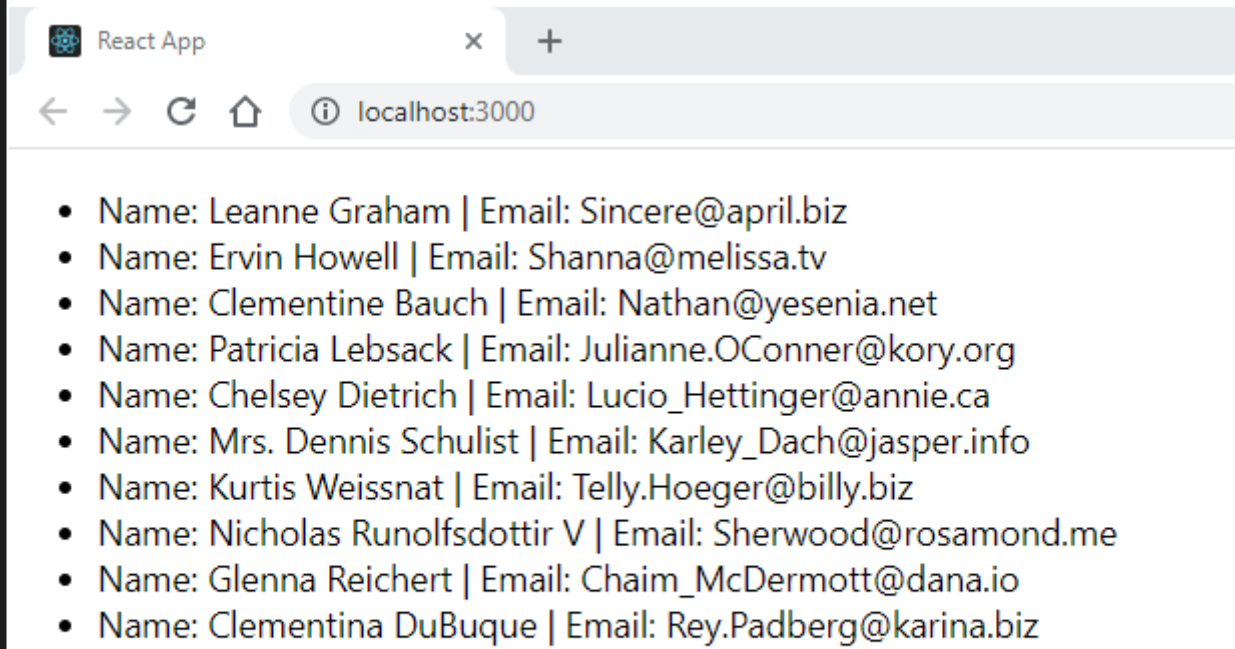
Api Call

- Name : Leanne Graham | Email :Sincere@april.biz
- Name : Ervin Howell | Email :Shanna@melissa.tv
- Name : Clementine Bauch | Email :Nathan@yesenia.net
- Name : Patricia Lebsack | Email :Julianne.OConner@kory.org
- Name : Chelsey Dietrich | Email :Lucio_Hettinger@annie.ca
- Name : Mrs. Dennis Schulist | Email :Karley_Dach@jasper.info
- Name : Kurtis Weissnat | Email :Telly.Hoeger@billy.biz
- Name : Nicholas Runolfsdottir V | Email :Sherwood@rosamond.me
- Name : Glenna Reichert | Email :Chaim_McDermott@dana.io
- Name : Clementina DuBuque | Email :Rey.Padberg@karina.biz

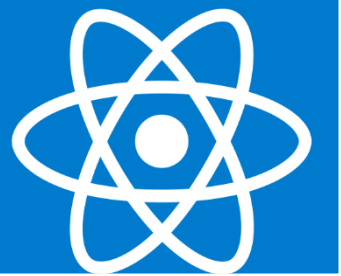
App.js M

src > App.js > App > constructor

```
1  import React from 'react';
2  class App extends React.Component {
3    constructor(props) {
4      super(props);
5      this.state = {
6        items: [],
7        isLoaded: false
8      }
9    }
10   componentDidMount() {
11     fetch('https://jsonplaceholder.typicode.com/users')
12       .then(res => res.json())
13       .then(json => {
14         this.setState({
15           isLoaded: true,
16           items: json
17         })
18       });
19   }
20   render() {
21     var { isLoaded, items } = this.state;
22     if (!isLoaded) {
23       return <div>Loading...</div>;
24     }
25     return (
26       <div classname="App">
27
28         <ul>
29           {items.map(item => (
30             <li key="{item.id}">
31               Name: {item.name} | Email: {item.email}
32             </li>
33           ))}
34         </ul>
35       </div>
36     );
37   }
38 }
39 export default App;
```



Hooks

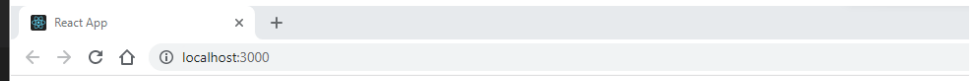


Function Component

App.js M

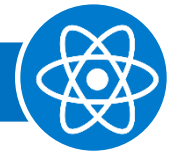
src > App.js > DataLoader

```
1  import React, { useState, useEffect } from "react";
2
3  export default function DataLoader() {
4    const [data, setData] = useState([]);
5
6    useEffect(() => {
7      fetch("https://jsonplaceholder.typicode.com/posts")
8        .then(response => response.json())
9        .then(data => setData(data));
10
11   });
12
13   return (
14     <div>
15       <ul>
16         {data.map(mydata => (
17           <li key={mydata.id}>{mydata.title}</li>
18         ))}
19       </ul>
20     </div>
21   );
22 }
```



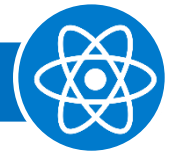
- sunt aut facere repellat provident occaecati excepturi optio repr
- qui est esse
- ea molestias quasi exercitationem repellat qui ipsa sit aut
- eum et est occaecati
- nesciunt quas odio
- dolorem eum magni eos aperiam quia
- magnam facilis autem
- dolorem dolore est ipsam

```
import React, { useState, useEffect } from "react";

export default function DataLoader() {
  const [data, setData] = useState([]);

  useEffect(() => {
    fetch("https://jsonplaceholder.typicode.com/posts")
      .then(response => response.json())
      .then(data => setData(data));

  });

  return (
    <div>
      <ul>
        {data.map(mydata => (
          <li key={mydata.id}>{mydata.title}</li>
        ))}
      </ul>
    </div>
  );
}
```

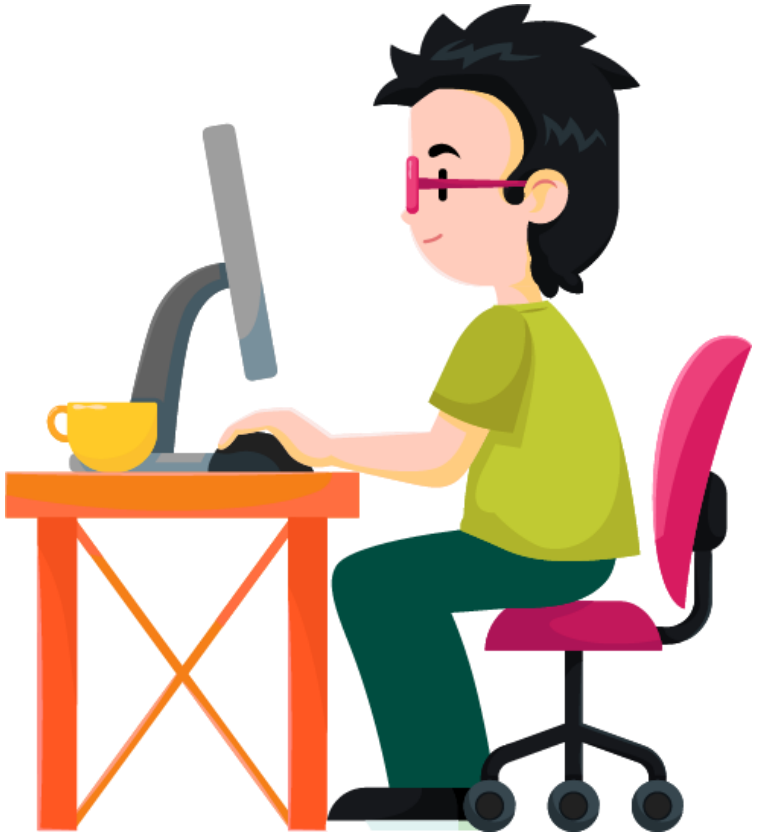


Task

- Print API Response in HTML Table.
  - https://jsonplaceholder.typicode.com/users
  - https://fakestoreapi.com/products (With Image)
  - https://reqres.in/api/users
  - https://dummyjson.com/quotes
  - https://dummyjson.com/products (With Image)
  - https://gorest.co.in/public/v2/users



Get Exclusive Video Tutorials



www.aptutorials.com

https://www.youtube.com/user/Akashtips

Get More Details

www.akashsir.com

If You Liked It !

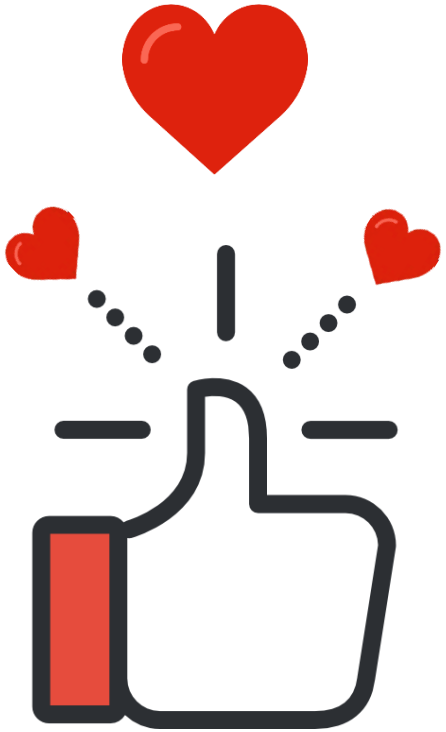
Rating Us Now

Just Dial

https://www.justdial.com/Ahmedabad/Akash-Technolabs-Navrangpura-Bus-Stop-Navrangpura/079PXX79-XX79-170615221520-S5C4_BZDET

Sulekha

https://www.sulekha.com/akash-technolabs-navrangpura-ahmedabad-contact-address/ahmedabad



Connect With Me

Akash Padhiyar
#AkashSir

www.akashsir.com
www.akashtechnolabs.com
www.akashpadhiyar.com
www.aptutorials.com

Social Info

Akash.padhiyar

Akashpadhiyar

Akash_padhiyar

+91 99786-21654

#Akashpadhiyar
#aptutorials