

FETCH API

The Fetch API is a promise-based API of JavaScript for making asynchronous HTTP requests in the browser similar to XMLHttpRequest (XHR). Unlike XHR, the fetch API is a simple and clean API that uses promises to provide more powerful features to fetch resources from the server. Fetch API is standardized now and is supported by all modern browsers except IE. The `FETCH()` method only has one mandatory argument: the URL of the resources that we want to fetch.

How to use fetch API:

To use a Fetch API, just pass the URL, the path to the resource we want to fetch, to `fetch()` method.

Syntax:

```
fetch('/js/users.json').then(response => {  
  // handle response data  
}).catch(err => {  
  // handle errors  
});
```

Page No. _____
Date _____

→ Pass the path of the resource that we want to retrieve as a parameter to `FETCH()`. We cannot block the user interface by waiting until the request finishes. That is why `FETCH()` returns a promise, an object which represents a future result. We are using the `then` method to wait for the server's response. The `catch()` method is optional. Its purpose is to intercept errors if the request fails to complete due to network failure or any other reason.

→ Now let us see how we can extract the JSON from that response once the request completes:

```
FETCH('URL HERE').THEN(  
  RES => RES.JSON().THEN(  
    JSON => CONSOLE.LOG(JSON));
```

→ We start the request by calling `FETCH()`. When the promise is fulfilled, it returns a response object, which exposes a `json` method. Within the first `THEN()`, we can call the `JSON` method to return the response body as JSON.

→ GET Request :

→ GET requests are widely used methods in APIs and websites. The purpose of this method is to retrieve data from the server at the specified resource. The Fetch API uses the GET method for asynchronous requests. Ex:

```
FETCH('HTTPS://API.GITHUB.COM/ORGS/NODEJS')  
THEN(RESPONSE => RESPONSE.JSON()). THEN(  
  DATA => {  
    console.log(DATA) // prints result from  
    'RESPONSE.JSON()' in get request  
  }) . CATCH (ERROR => console.error(ERROR))
```

→ As we know, the `FETCH()` method returns a promise. The response turned by the promise is a stream object, which means that it returns another promise when we call the `JSON()` method. Call to `JSON()` method indicates that we are expecting a JSON response. If we want an XML response, use the `XML()` method.

→ POST Request :

→ The purpose of the post request is to send the data to the server and create a new resource. The resource

post request creates subordinate to some other parent resource. When a new resource is posted to the parent, the API service will automatically associate the new resource by assigning it an ID. All we need to do is set the method and body parameters in the fetch() options:

```
LET DATA = {  
  FIRST-NAME: 'JOHN',  
  LAST-NAME: 'ADAMS',  
  JOB-TITLE: 'SOFTWARE ENGINEER'  
};  
  
CONST OPTIONS = {  
  METHOD: 'POST',  
  BODY: JSON.stringify(DATA),  
  HEADERS: {  
    'CONTENT-TYPE': 'APPLICATION/JSON'  
  }  
}  
  
FETCH('HTTPS://REARRES.IN/USERS', OPTIONS).  
  THEN (RES => RES.json()).  
  THEN (RES => console.log(RES));
```