Fetch API - Status Codes

Fetch API provide a special property which is used to find the status of the request and the name of this property is the status property. It is a read-only property of the Response interface which return the HTTP status code of the response sent by the server of the given request. For example, 404 - resource were not found, 200 - success, 400 - bad request, etc. It is supported by all modern web browsers.

Syntax

```
response.status
```

The value returned by the status property is an unsigned short number which represents the status of the current request.

Status Codes

The status codes that HTTP status returned are as follows –

Successful

The successful status codes are those status codes which will return by the server when the request is fulfilled successfully. Some of the commonly used successful status codes with their meanings are as follows –

Status	Message	Description
200	OK	If the request is OK.
201	Created	When the request is complete and a new resource is created.
202	Accepted	When the request is accepted by the server.
204	No Content	When there is no data in the response body.
205	Reset Content	For additional inputs, the browser clears the form used f transaction.

206	Partial Content	When the server returns the partial data of the specified size.

Redirection

The redirection status codes are those status codes which represent the status of a redirect response. Some of the commonly used redirection status codes with their descriptions are as follows –

Status	Message	Description
300	Multiple Choices	It is used to represent a link list. So that user can select any one link and go to that location. It allows only five locations.
301	Moved Permanently	When the requested page is moved to the new URL.
302	Found	When the requested page is found in a different URL.
304	Not modified	URL is not modified.

Client Error

The Client error status codes represent an error that occurs on the client side during the request. Or we can say that they inform the client that due to an error, the request was unsuccessful. Some of the commonly used client error status codes with their descriptions are as follows –

Status	Message	Description
400	Bad Request	The server cannot fulfil the request because the request was malformed or has an invalid syntax.
401	Unauthorised	The request needs authentication and the user does not provide valid credentials.
403	Forbidden	The server understood the request but does not fulfil it.
404	Not Found	The requested page is not found.
405	Method Not Allowed	The method through which the request is made is not supported by the page.
406	Not Acceptable	The response generated by the server cannot be accepted by the client.

408	Request Timeout	Server timeout
409	Conflict	The request does not fulfil due to a conflict in the request.
410	Gone	The requested page is not available.
417	Exception Failed	The server does not match the requirement of the Expect request header field.

Server Error

The server error status codes represent an error that occurs on the server side during the request. Or we can say that they inform the client that due to an error with the server, the request was unsuccessful. Some of the commonly used server error status codes with their descriptions are as follows –

Status	Message	Description
500	Internal Server Error	When the server encounter error while processing the request.
501	Not Implemented	When the server does not recognise the request method or lacks the ability to fulfil the request.
502	Bad Gateway	When the server acts like a gateway and recovers an invalid response from another server(upstream).
503	Service Unavailable	When the server is not available or down.
504	Gateway Timeout	When the server acts like a gateway and does not receive a response from the other server(upstream) on time.
505	HTTP Version Not Supported	When the server does not support the version of the HTTP protocol.
511	Network Authentication Required	When the client needs to authenticate to gain access to the network.

Example 1: Finding status code using fetch() function

In the following program, we find the status code of the current request. So for that, we fetch the data from the given URL. If the response returned by the server is OK, then

display the status code. If not, then display the request failed status. If we get an error, then this code uses the catch() function to handle the error.

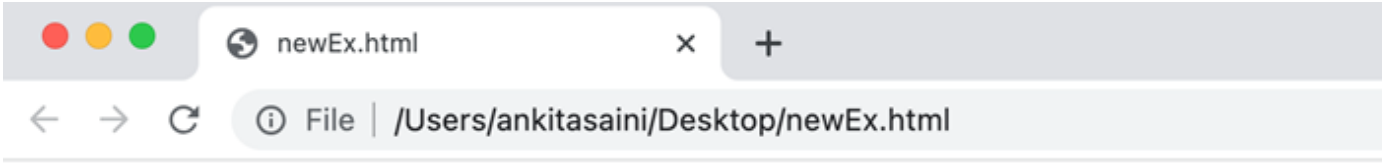
</> Open Compiler

```
<!DOCTYPE html>
<html>
<body>
<script>
   fetch("https://jsonplaceholder.typicode.com/todos")
   .then(response=>{
      if (response.ok){

      const mystatus = response.status;

      // Display output in HTML page
      document.getElementById("sendData").innerHTML = JSON.stringify(mystatus);
      }else{
         console.log("Request Fail:", mystatus);
      }
   })
   // Handling error
   .catch(err =>{
      console.log("Error is:", err)
   });
</script>
<h2>Status code of request</h2>
<div>
   <p>Status of the current request is </p>
   <!-- Displaying data-->
   <p id = "sendData"></p>
</div>
</body>
</html>
```

Output



Status code of request

Status of the current request is

200

Example 2: Finding status code using fetch() function with async/await

In the following program, we find the status code of the current request. So for that, we create an async function. In this function, we fetch the data from the given URL using the fetch() function. If the response returned by the server is OK, then display the status code in the console log. If not, then display the request failed status. If we get an error, then this code uses the catch() function to handle that error.

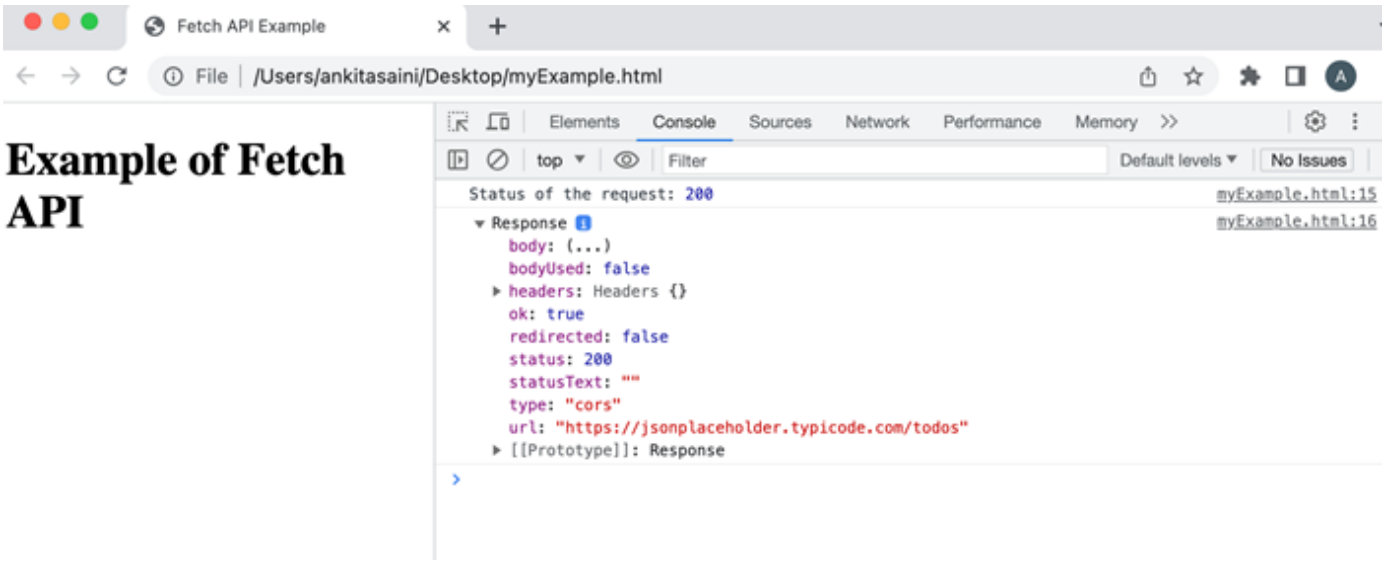
</> Open Compiler

```
<!DOCTYPE html>
<html>
<head>
<title>Fetch API Example</title>
</head>
<body>
<h1>Example of Fetch API</h1>
<script>
   async function getStatus() {
      try {
         const myResponse = await fetch("https://jsonplaceholder.typicode.com/todo

         // Finding the status of the request
         console.log("Status of the request:", myResponse.status);
         console.log(myResponse);
      } catch (err) {
         console.log("Error is:", err);
      }
   }
```

```
    getStatus();
</script>
</body>
</html>
```

Output



Conclusion

So this is how we can find the status code of the current request returned by the server. Using these status codes we can perform various operations such as checking if the request is successful or not, handling the specified error, or performing the appropriate action on the response returned by the server. Now in the next article, we will see how Fetch API handles errors.