Fetch API - Credentials

In Fetch API the cookies, authorization headers and TLS client certificates are known as the credentials. We can also say as credentials are the digital documents(like passwords, usernames, certificates, etc) that confirm the identity of the user or client while making a request to the server.

Let's understand these credentials in more detail below −

Cookies − They are the small chunks of data stored by the web browser and sent with all the same origin requests. It is used to store session information, frequently used data, etc. They also control their session, scope and accessibility. Cookies are also sent by the server with the help of the Set-Cookie header.

Authorization Headers − These include those HTTP headers that contain authentication information like password, username, key, etc. Authorization headers are used to implement various authentication schemas and are also validated by the server using various methods like hashing, encryption, etc.

TLS Client Certificates − They are digital certificates that are provided by certified authorities and also installed on the client's device. They are used to provide identity proof of the client while creating a secure connection with the server with the help of Transport layer security.

Credentials Property

The credentials property controls whether the cookies and other credential certificates will be sent in the cross-origin request or not. It is an optional property in the fetch() function.

Syntax

```
fetch(resourceURL, {credentials:"include"})
```

This property can have one value out of the following three values −

omit − When the value of the credentials property is set to omit that means the browser will remove all the credentials from the request and also ignore the credentials sent in the response.

same-origin − When the value of the credentials property is set to same-origin that means the browser will include credentials in those requests that are made to the same origin as the requesting page. And use only those credentials that are from the same origin URLs. It is the default value of this property.

include − When the value of the credentials property is set to include that means the browser will include credentials in both same-origin and cross-origin requests and always use those credentials that are sent in the response.

Example 1

In the following program, we use the fetch() function to make a request to the given URL. Here we set the credentials property to "include" value which represents both the cross-origin and same-origin credentials included in the request. After sending the request to the server now we use the then() function to handle the response. If we encounter an error, then that error is handled by the catch() function.

</> Open Compiler

```
<!DOCTYPE html>
<html>
<body>
<script>
   // Retrieving data from the URL using a GET request
   fetch("https://jsonplaceholder.typicode.com/todos/21", {
      // Sending both same-origin and
      // cross-origin credentials
      credentials: "include"
   })
   // Converting received data into text
   .then(response => response.text())
   .then(myData => {
      // Display the retrieve Data
      console.log(myData);
   })
   .catch(err=>{
      // Cach error if occur
      console.log("Found Error:", err)
   });
</script>
<h2>Fetch API Example</h2>
```

```
</body>
</html>
```

Output

Fetch API Example

Example 2

In the following program, we use the fetch() function to make a request to the given URL. Here we set the credentials property to the "same-oigin" value which means the credentials are only included in those requests that are made to the same origin or domain. After sending the request to the server now we use the then() function to handle the response. If we encounter an error, then that error is handled by the catch() function.

Open Compiler

```
<!DOCTYPE html>
<html>
<body>
<script>
   // Retrieving data from the URL using a GET request
   fetch("https://jsonplaceholder.typicode.com/todos/21", {
      // Sending credentials only for the same domain request
      credentials: "same-origin"
   })

   // Converting received data into text
   .then(response => response.text())
   .then(myData => {
      // Display the retrieve Data
      console.log(myData);
   })
   .catch(err=>{
      // Cach error if occur
      console.log("Found Error:", err)
   });
</script>
```

```
<h2>Fetch API Example</h2>
</body>
</html>
```

Output

Fetch API Example

Conclusion

Hence using credentials we can control how the credentials are sent in the request or how to handle the credentials sent back in the response. The credentials property only affects cross-origin requests and for the same-origin request, the browser will automatically add credentials to the request. Now in the next article, we will learn how to send a GET request in Fetch API.