Fetch API - Handling Binary Data

Binary data is data that is in the binary format not in the text format e.g. new Uint8Array([0x43, 0x21]). It includes images, audio, videos, and other files that are not in plain text. We can send and receive binary data in the Fetch API. While working with binary data in Fetch API it is important to set proper headers and response types. For binary data, we use "Content-Type": "application/octet-stream" and the "responseType" property to "arraybuffer" or "blob" which indicates that binary data is received.

Let us understand how to Send and Receive Binary Data in Fetch API using the following examples.

Sending Binary Data

To send binary data we use send() method of XMLHttpRequest which can easily transmit binary data using ArrayBuffer, Blob or File object.

Example

In the following program, we create a program which will send binary data to the server. So first we create binary data, and then we convert the binary data into Blob using Blob() constructor. Here this constructor takes two arguments: binary data and headers for the binary data. Then we create a FormData object and add Blob to the FormData object. Then we use the fetch() function to send the request to the server and then handle the response returned by the server and handle the error if occurs.

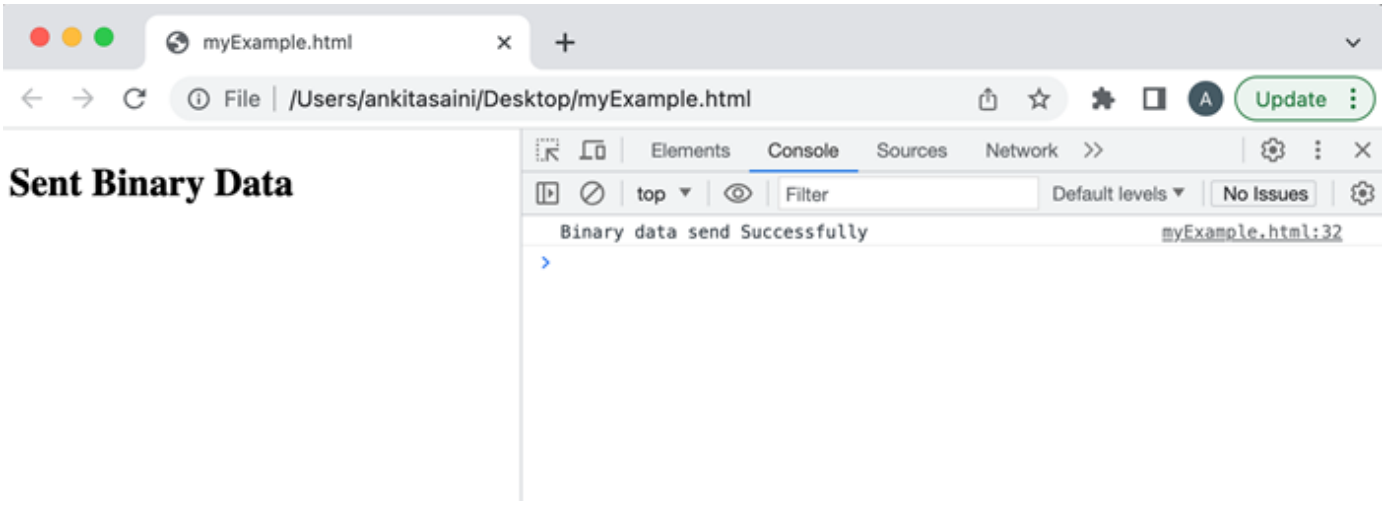
</> Open Compiler

```
<!DOCTYPE html>
<html>
<body>
<script>
   // Binary data
   var BinaryData = new Uint8Array([0x32, 0x21, 0x45, 0x67]);

   // Converting binary data into Blob
   var blobObj = new Blob([BinaryData], {type: 'application/octet-stream'});

   // Creating FormData object
   var obj = new FormData();
```

```
    // Add data to the object
    // Here myfile is the name of the form field
    obj.append("myfile", blobObj);

    // Sending data using POST request
    fetch("https://jsonplaceholder.typicode.com/todos", {
       // Adding POST request
       method: "POST",

       // Adding body which we want to send
       body: obj
    })
    // Handling the response
    .then(response =>{
       if (response.ok){
          console.log("Binary data send Successfully");
       }
    })
    // Handling the error
    .catch(err=>{
       console.log("Found error:", err)
    });
</script>
   <h2>Sent Binary Data</h2>
</body>
</html>
```

Explore our latest online courses and learn new skills at your own pace. Enroll and become a certified expert to boost your career.

Output



Receiving Binary Data

In Fetch API, receiving binary data means retrieving the response data from the server after making the request. To receive binary data we have set the correct value of the responseType either ArrayBuffer(), or Blob().

Example

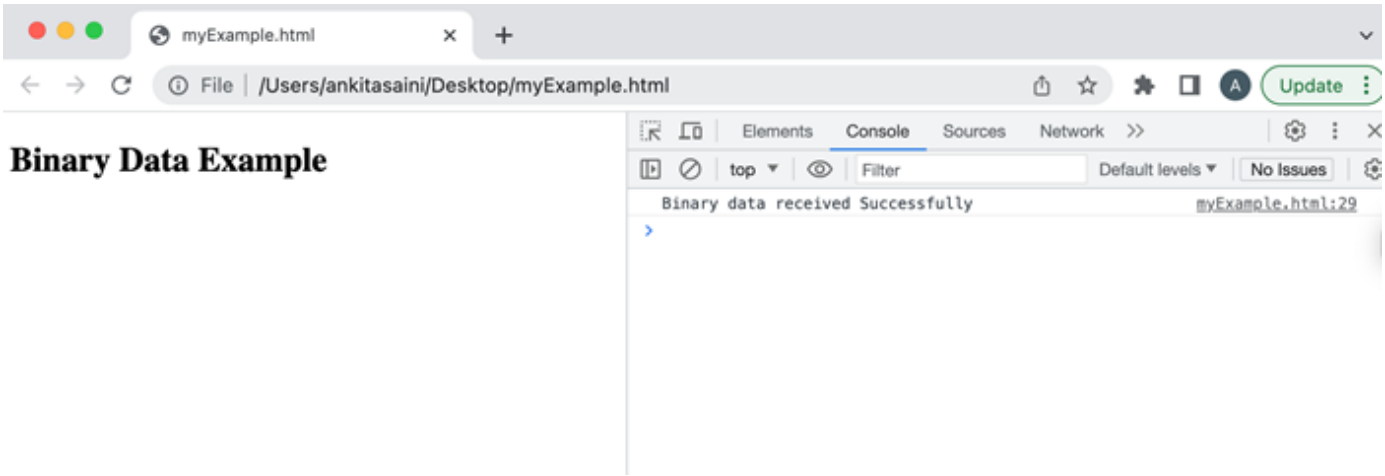
In the following program, we create a program which will receive binary data from the server. So we use the fetch() function to get binary data from the given URL. In the fetch() we define headers which tell the browser we are expecting a binary response and set responseType to arraybuffer so that it tells the browser that you are receiving a response as an ArrayBuffer. Then we receive the promise in the.then() block and check the status is OK. If the status is OK, then convert the response to ArrayBuffer with the help of the arrayBuffer() function. The next .then() processes the return binary data and displays the data accordingly. The .catch() function handles the error if occurs.

</> Open Compiler

```
<!DOCTYPE html>
<html>
<body>
<script>
   // Receiving data using GET request
   fetch("https://jsonplaceholder.typicode.com/todos", {
      // Adding Get request
      method: "GET",

      // Setting headers
      headers: {
         'Content-Type': 'application/octet-stream',
```

```
      },
      // Setting response type to arraybuffer
      responseType: "arraybuffer"
   })

   // Handling the received binary data
   .then(response =>{
      if (response.ok){
         return response.arrayBuffer();
      }
      console.log("Binary data send Successfully");
   })
   .then(arraybuffer => console.log("Binary data received Successfully"))

   // Handling the error
   .catch(err=>{
      console.log("Found error:", err)
   });
</script>
   <h2>Binary Data Example</h2>
</body>
</html>
```

Output



Conclusion

So this is how we can handle binary data using Fetch API. To handle binary data we need to convert binary data to an appropriate data format. We can also send binary data in the

file, string, ArrayBuffer, and Blob. Now in the next chapter, we will learn how to find status codes using Fetch API.