HTTP Requests using JS | Cheat Sheet

1. Fetch

The

`fetch()` method is used to fetch resources across the Internet.

Syntax: fetch(URL, OPTIONS)

URL - URL of the resource
OPTIONS - Request Configuration

1.1 Request Configuration

We can configure the fetch request with many options like,

- Request Method
- Headers
- Body
- Credentials
- Cache, etc.

We can configure a request by passing an

`options` object with required properties and their values.

For example,

```
let options = {
  method: "GET",
  headers: {
    "Content-Type": "application/json"
  },
  body: JSON.stringify(data)
};
```

2. Making HTTP Requests using Fetch

The

`method` property value in the `options` object can be `GET`, `POST`, `PUT`, `DELETE`, etc. The default method is `GET`.

2.1 GET

The

`GET` method can be used to retrieve (get) data from a specified resource.

For example,

```
let options = {
  method: "GET"
};

fetch("https://gorest.co.in/public-api/users", options);
```

2.2 POST

The

`POST` method can be used to send data to the server.

```
let data = {
  name: "Rahul",
  gender: "Male",
  email: "rahul@gmail.com",
  status: "Active"
};

let options = {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
    Accept: "application/json",
    Authorization: "Bearer ACCESS-TOKEN"
  },
  body: JSON.stringify(data)
};

fetch("https://gorest.co.in/public-api/users", options)
  .then(function(response) {
    return response.json();
  })
  .then(function(jsonData) {
    console.log(jsonData);
  });
```

2.3 PUT

The

`PUT` method can be used to update the existing resource.

```
let data = {
  name: "Rahul Attuluri"
};

let options = {
  method: "PUT",
  headers: {
    "Content-Type": "application/json",
    Accept: "application/json",
    Authorization: "Bearer ACCESS-TOKEN"
  },
  body: JSON.stringify(data)
};

fetch("https://gorest.co.in/public-api/users/1359", options)
  .then(function(response) {
    return response.json();
  })
  .then(function(jsonData) {
    console.log(jsonData);
  });
```

2.4 DELETE

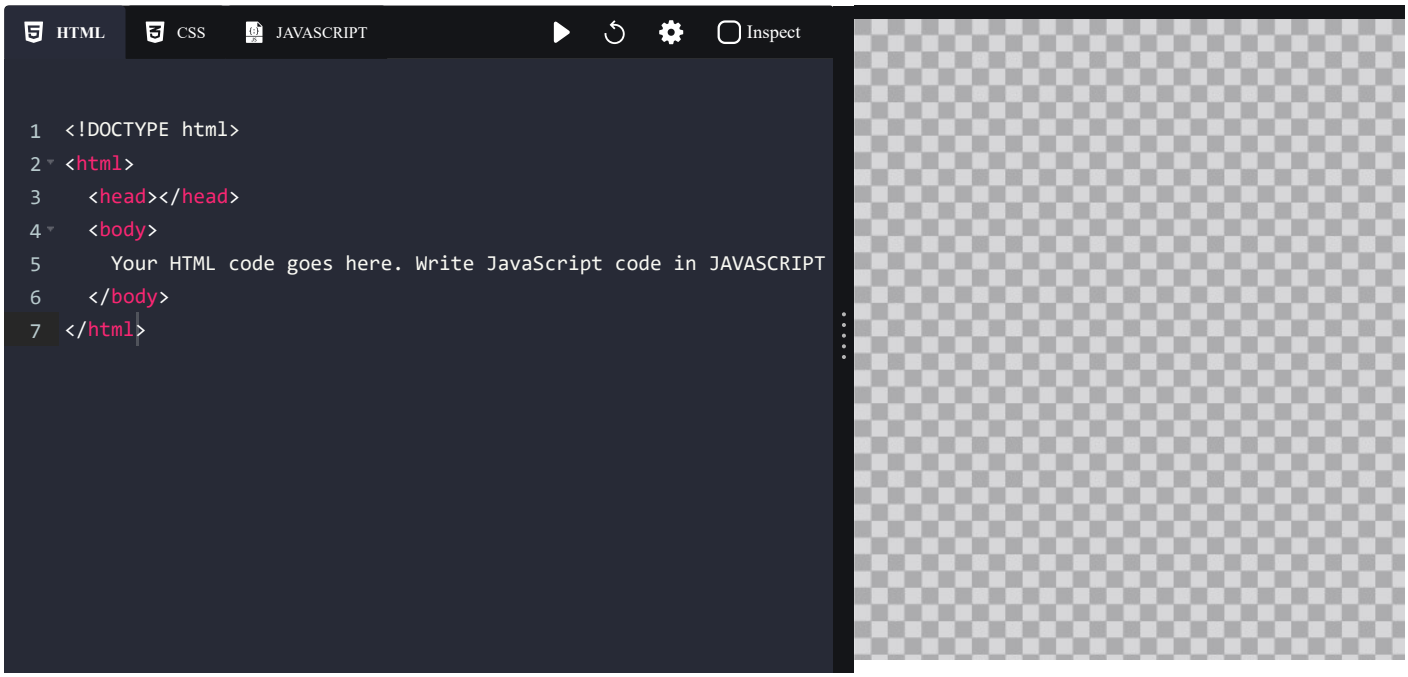
The

`DELETE` method can be used to delete the specified resource.

```
let options = {
  method: "DELETE",
  headers: {
```

```
    "Content-Type": "application/json",
    Accept: "application/json",
    Authorization: "Bearer ACCESS-TOKEN"
  }
};

fetch("https://gorest.co.in/public-api/users/1359", options)
  .then(function(response) {
    return response.json();
  })
  .then(function(jsonData) {
    console.log(jsonData);
  });
```

Try out the different HTTP Request Methods and check the output in the below Code Playground console.



3. HTTP Response Object Properties and Methods

Response Object provides multiple properties to give more information about the HTTP Response.

- status (number) - HTTP status code
- statusText (string) - Status text as reported by the server, e.g. "Unauthorized"
- headers
- url
- text() - Getting text from response
- json() - Parses the response as JSON

For example,

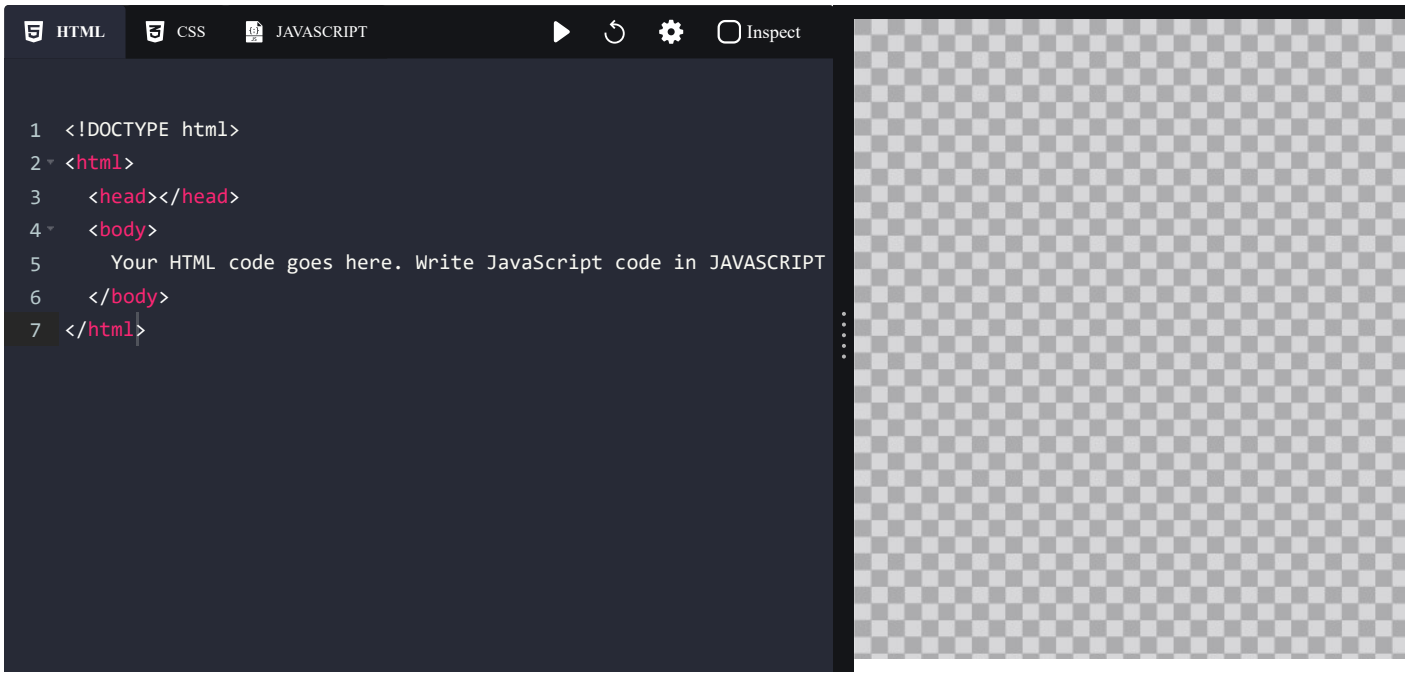
```
let options = {
  method: "GET"
};

fetch("https://gorest.co.in/public-api/users", options)
  .then(function(response) {
    return response.status;
  })
  .then(function(status) {
    console.log(status);  // 200
  });
```

In the above example, we can get the response status as

`200` when the request is success.

Try out accessing the different properties and methods of the HTTP Response Object like

`url` , `text()` , `json()` , etc. and check the output in the below Code Playground console.



Submit Feedback

Notes Discussions

Notes