11 PERCOBAAN 11: AJAX DAN FETCH API

11.1 TUJUAN PERCOBAAN

- Memahami bagaimana AJAX dan Fetch API bekerja
- Dapat menggunakan JavaScript untuk berinteraksi dengan Web Service
- Dapat memanipulasi halaman web secara parsial

11.2 TINJAUAN PUSTAKA

11.2.1 AJAX

AJAX merupakan singkatan dari Asynchronous JavaScript And XML merupakan salah satu teknik pada pengembangan aplikasi web asinkron. AJAX memungkinkan pemutakhiran sebagian halaman web saja, meminta dan menerima data baru sesudah halaman selesai dimuat, dan mengirim data ke server tanpa interaksi pengguna.

AJAX memanfaatkan obyek XMLHttpRequest yang tersedia di browser untuk membuat situs web memuat konten ke halaman tanpa harus melakukan refresh. Walau XMLHttpRequest sudah mulai dipakai semenjak tahun 2000, pada tahun 2004 pengembang web mulai tertarik untuk memakainya sebagai salah satu tool interaksi akibat sukses penggunaannya pada Gmail dan Kayak, aplikasi skala enterprise. Istilah AJAX pertama kali dipakai pada tahun 2005 pada artikel berjudul Ajax: A New Approach to Web Applications yang ditulis oleh Jesse James Garret.

Ajax mewakili kumpulan teknologi yang memungkinkan aplikasi web berkomunikasi dengan server di latar belakang, tanpa harus mengubah keadaan halaman saat ini. Teknologi yang dilibatkan untuk membangun Ajax adalah:

- HTML dan CSS untuk penyajian
- DOM untuk tampilan dinamis dan interaksi dengan data
- JSON atau XML untuk pertukaran data
- XMLHttpRequest untuk komunikasi asinkron
- JavaScript sebagai jembatan antara teknologi-teknologi tersebut

Pada praktiknya, JSON yang umumnya dipakai. Hal ini disebabkan JSON memiliki ukuran yang kecil dan format yang mudah dimengerti.

```
// Inisialisasi HTTP request.
var xhr = new XMLHttpRequest();
xhr.open('GET', 'send-ajax-data.php');

// Melacak perubahan state dari request
xhr.onreadystatechange = function () {
    var DONE = 4; // readyState 4 artinya request selesai
    var OK = 200; // status 200 adalah return berhasil
    if (xhr.readyState === DONE) {
        if (xhr.status === OK) {
            console.log(xhr.responseText); // 'Ini adalah keluaran.'
        } else {
```

```
            console.log('Error: ' + xhr.status); // Kesalahan terjadi selama
request.
        }
    }
};

// Kirim request ke send-ajax-data.php
xhr.send(null);
```

Gambar 11-1. Contoh Skrip AJAX menggunakan JavaScript murni

JavaScript Library yang sering dipakai untuk menggunakan AJAX adalah jQuery. Hal ini karena jQuery menyederhanakan kode yang harus ditulis dan dinyatakan dengan jelas.

```
$.ajax({
    type: 'GET',
    url: 'send-ajax-data.php',
    dataType: "JSON", // data type expected from server
    success: function (data) {
        console.log(data);
    },
    error: function() {
        console.log('Error: ' + data);
    }
});
```

Gambar 11-2. Contoh Skrip AJAX menggunakan JavaScript dengan library jQuery

Beberapa kelemahan AJAX diantaranya adalah CORS (cross-origin resource sharing) yang membatasi AJAX untuk mengambil data dari host yang berbeda dengan tidak terbacanya informasi yang dimuat dinamis oleh teknologi screen reading seperti WAI-ARIA dan juga Web Crawler yang tidak mengeksekusi kode JavaScript sehingga situs tidak diindeks.

11.2.2 Fetch API

Fetch merupakan standar terbaru untuk komunikasi data asinkron yang bertujuan menggantikan AJAX dengan teknik yang lebih jelas serta lebih tangguh terhadap kesalahan. Dengan menggunakan Fetch, pengembang web diharapkan dapat bekerja dengan lebih jelas dalam mendefinisikan komunikasi data asinkron dengan server dan mengurangi ketergantungan terhadap library tertentu.

Fetch menyediakan definisi umum dari obyek Request dan Response. Hal ini memungkin kedua obyek tersebut digunakan ketika dibutuhkan, bisa untuk Service Worker, Cache API, dan hal lainnya yang memodifikasi request dan response. Atau, kondisi dimana diperlukan menghasilkan respon yang dapat diprogram.

Selain itu, fetch juga menyediakan dukungan terhadap CORS (*Cross-Origin Resource Sharing*) yang memungkinkan komunikasi terjadi pada host/domain yang berbeda. CORS tidak didukung oleh Ajax. Agar bisa mengakses host/domain yang berbeda digunakan teknik yang disebut dengan JSONP, tetapi pendekatan ini kurang elegan dan tidak intuitif. Sehingga, dukungan CORS oleh fetch merupakan hal yang baik untuk pengembangan web, terutama untuk kode yang lebih mudah dibaca.

Cara kerja Fetch API adalah fetch() menggunakan sebuah argumen yang wajib, yaitu alamat ke sumberdaya yang mau diambil. Kemudian, Promise dikembalikan yang memberikan Response dari Request, bisa berhasil atau gagal. Promise merepresentasikan hasil akhir dari operasi asinkron dan nilai yang dikembalikan. Hal ini membuat aplikasi web tidak mengalami blocking dan tetap berjalan seperti biasa, dan ketika nilai dikembalikan, proses baru dijalankan.

Ketika response diterima, banyak metode yang dapat digunakan untuk menanganinya. Promise dapat digunakan berkali-kali untuk memproses hasil sampai dengan yang dibutuhkan.

11.3 PERCOBAAN

Percobaan ini memerlukan :

1. Web server yang menggunakan minimal PHP 5.1 dan MySQL 5.0.
2. Basis data bernama kuliah dengan collation:utf8mb4_general_ci di dalam server MySQL yang digunakan. Telah dibuat pada praktikum PHP dan MySQL.
3. Berkas php (dbconfig.php) pada praktikum PHP dan MySQL dipakai di semua kode php untuk dapat terkoneksi ke basis data.
4. Berkas api.php yang merupakan gabungan seluruh method GET, POST, PUT, dan DELETE sebagai web service yang diakses.

11.3.1 Percobaan 11-1: Ajax GET

Ketik kode program di bawah ini:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" />
    <title>AJAX GET</title>
</head>
<body>
    <h1>AJAX GET</h1>
    <button id="btnAmbilData">Ambil Data</button>
    <div id="tampilanData"></div>
    <script>
        var tampilanData = document.getElementById("tampilanData");

        function AjaxGET() {
            // Penambahan ?t=" + Math.random() adalah untuk menghindari
hasil cache
            var data_file = "api.php/mahasiswa?t=" + Math.random();
            var http_request = new XMLHttpRequest();
            try {
                // Opera 8.0+, Firefox, Chrome, Safari
                http_request = new XMLHttpRequest();
            } catch (e) {
                // Internet Explorer Browsers
                try {
                    http_request = new ActiveXObject("Msxml2.XMLHTTP");
                } catch (e) {
```

```
                    try {
                        http_request = new
ActiveXObject("Microsoft.XMLHTTP");
                    } catch (e) {
                        // Ada masalah
                        alert("Browser rusak!");
                        return false;
                    }
                }
            }
            http_request.onreadystatechange = function() {
                if (http_request.readyState == 4) {
                    // Fungsi JavaScript JSON.parse untuk parsing data JSON
                    var jsonObj = JSON.parse(http_request.responseText);
                    tampilkanData(jsonObj);
                }
            }
            http_request.open("GET", data_file, true);
            http_request.send();
        }

        //Fungsi untuk memudahkan buat Node
        function createNode(element) {
            // Membuat tipe elemen yang dilewatkan melalui parameter
            return document.createElement(element);
        }

        //Fungsi untuk menambahkan sub node di bawah Node
        function append(parent, el) {
            // Append parameter kedua ke yang pertama
            return parent.appendChild(el);
        }

        // Fungsi untuk menampilkan data
        function tampilkanData(dataRAW) {
            tampilanData.innerHTML = "Status: " + dataRAW.status;
            tampilanData.innerHTML = "Status: " + dataRAW.status;
            // memakai fungsi pembuat elemen
            let table = createNode('table'),
                thead = createNode('thead'),
                th1 = createNode('th'),
                th2 = createNode('th');
            // memasukkan judul
            th1.innerHTML = 'Nama';
            th2.innerHTML = 'NPM';
            // memakai fungsi append ke parameter pertama
            append(thead, th1);
            append(thead, th2);
            append(table, thead);

            for (i = 0; i < dataRAW.data.length; i++) {
                let tr = createNode('tr'),
                    td1 = createNode('td'),
                    td2 = createNode('td');
```

```
                // memasukkan data ke dalam data tabel
                td1.innerHTML = dataRAW.data[i].nama;
                td2.innerHTML = dataRAW.data[i].npm;
                // memakai fungsi append ke parameter pertama
                append(tr, td1);
                append(tr, td2);
                append(table, tr);
            }
            append(tampilanData, table);
        }

        document.getElementById("btnAmbilData").addEventListener("click",
AjaxGET);
    </script>
</body>
</html>
```

Kode 11-1. Ajax-get.html

- Jelaskan proses yang terjadi saat tombol Ambil Data ditekan!
- Kenapa perlu menghindari hasil cache?

11.3.2 Percobaan 11-2: Ajax POST

Ketik kode program di bawah ini:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" />
    <title>AJAX POST</title>
</head>
<body>
    <h1>AJAX POST</h1>
    <input type="text" name="nama" id="nama" placeholder="Ketik nama
disini">
    <input type="text" name="npm" id="npm" placeholder="Ketik NPM disini">
    <button id="btnKirimData">Kirim Data</button>
    <div id="tampilanData"></div>

    <script>
        var tampilanData = document.getElementById("tampilanData");

        function AjaxPOST() {
            var inputNama = document.getElementById("nama").value;
            var inputNPM = document.getElementById("npm").value;

            var data_file = "api.php/mahasiswa";
            var params = "nama=" + inputNama + "&npm=" + inputNPM;
            var http_request = new XMLHttpRequest();
            try {
                // Opera 8.0+, Firefox, Chrome, Safari
                http_request = new XMLHttpRequest();
            } catch (e) {
                // Internet Explorer Browsers
                try {
```

```
                    http_request = new ActiveXObject("Msxml2.XMLHTTP");
                } catch (e) {
                    try {
                        http_request = new
ActiveXObject("Microsoft.XMLHTTP");
                    } catch (e) {
                        // Ada masalah
                        alert("Browser rusak!");
                        return false;
                    }
                }
            }
            http_request.onreadystatechange = function() {
                if (http_request.readyState == 4) {
                    // Fungsi JavaScript JSON.parse untuk parsing data JSON
                    var jsonObj = JSON.parse(http_request.responseText);
                    tampilkanData(jsonObj);
                }
            }
            http_request.open("POST", data_file, true);
            http_request.setRequestHeader("Content-type", "application/x-
www-form-urlencoded");
            http_request.send(params);
        }

        // Fungsi untuk menampilkan data
        function tampilkanData(dataRAW) {
            tampilanData.innerHTML = "Status: " + dataRAW.status + "<br/>";
            tampilanData.innerHTML += "ID data: " + dataRAW.id;
        }

        document.getElementById("btnKirimData").addEventListener("click",
AjaxPOST);
    </script>
</body>
</html>
```

Kode 11-2. Ajax-post.html

- Kenapa variabel inputNama dan inputNPM berada di dalam fungsi AjaxPOST?
- Untuk apa `http_request.setRequestHeader`?

11.3.3 Percobaan 11-3: Ajax PUT

Ketik kode program di bawah ini:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" />
    <title>AJAX PUT</title>
</head>
<body>
    <h1>AJAX PUT</h1>
    <input type="text" name="id" id="id" placeholder="Ketik ID data yang mau
diganti disini">
```

```
    <input type="text" name="nama" id="nama" placeholder="Ketik nama
disini">
    <input type="text" name="npm" id="npm" placeholder="Ketik NPM disini">
    <button id="btnKirimData">Kirim Data</button>
    <div id="tampilanData"></div>

    <script>
        var tampilanData = document.getElementById("tampilanData");

        function AjaxPUT() {
            var inputID = document.getElementById("id").value;
            var inputNama = document.getElementById("nama").value;
            var inputNPM = document.getElementById("npm").value;

            var data_file = "api.php/mahasiswa/" + inputID;
            var params = "nama=" + inputNama + "&npm=" + inputNPM;
            var http_request = new XMLHttpRequest();
            try {
                // Opera 8.0+, Firefox, Chrome, Safari
                http_request = new XMLHttpRequest();
            } catch (e) {
                // Internet Explorer Browsers
                try {
                    http_request = new ActiveXObject("Msxml2.XMLHTTP");
                } catch (e) {
                    try {
                        http_request = new
ActiveXObject("Microsoft.XMLHTTP");
                    } catch (e) {
                        // Ada masalah
                        alert("Browser rusak!");
                        return false;
                    }
                }
            }
            http_request.onreadystatechange = function() {
                if (http_request.readyState == 4) {
                    // Fungsi JavaScript JSON.parse untuk parsing data JSON
                    var jsonObj = JSON.parse(http_request.responseText);
                    tampilkanData(jsonObj);
                }
            }
            http_request.open("PUT", data_file, true);
            http_request.setRequestHeader("Content-type", "application/x-
www-form-urlencoded");
            http_request.send(params);
        }

        // Fungsi untuk menampilkan data
        function tampilkanData(dataRAW) {
            tampilanData.innerHTML = "Status: " + dataRAW.status + "<br/>";
            tampilanData.innerHTML += "Data -> ID: " + dataRAW.id;
            tampilanData.innerHTML += ", Nama: " + dataRAW.nama;
            tampilanData.innerHTML += ", NPM: " + dataRAW.npm;
```

```
        }

        document.getElementById("btnKirimData").addEventListener("click",
AjaxPUT);
    </script>
</body>
</html>
```

Kode 11-3. Ajax-put.html

- Mengapa, pada var data_file, variabel inputID ditambahkan?
- Untuk apa variabel params dalam http_request.send?

11.3.4 Percobaan 11-4: Ajax DELETE

Ketik kode program di bawah ini:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" />
    <title>AJAX DELETE</title>
</head>
<body>
    <h1>AJAX DELETE</h1>
    <input type="text" name="id" id="id" placeholder="Ketik ID data yang mau
dihapus disini">
    <button id="btnHapusData">Hapus Data</button>
    <div id="tampilanData"></div>
    <script>
        var tampilanData = document.getElementById("tampilanData");

        function AjaxDELETE() {
            var inputID = document.getElementById("id").value;

            var data_file = "api.php/mahasiswa/" + inputID;
            var http_request = new XMLHttpRequest();
            try {
                // Opera 8.0+, Firefox, Chrome, Safari
                http_request = new XMLHttpRequest();
            } catch (e) {
                // Internet Explorer Browsers
                try {
                    http_request = new ActiveXObject("Msxml2.XMLHTTP");
                } catch (e) {
                    try {
                        http_request = new
ActiveXObject("Microsoft.XMLHTTP");
                    } catch (e) {
                        // Ada masalah
                        alert("Browser rusak!");
                        return false;
                    }
                }
            }
            http_request.onreadystatechange = function() {
```

```
                if (http_request.readyState == 4) {
                    // Fungsi JavaScript JSON.parse untuk parsing data JSON
                    var jsonObj = JSON.parse(http_request.responseText);
                    tampilkanData(jsonObj);
                }
            }
            http_request.open("DELETE", data_file, true);
            http_request.send();
        }

        // Fungsi untuk menampilkan data
        function tampilkanData(dataRAW) {
            tampilanData.innerHTML = "Status: " + dataRAW.status + "<br/>";
            tampilanData.innerHTML += "Data -> ID: " + dataRAW.id;
        }

        document.getElementById("btnHapusData").addEventListener("click",
AjaxDELETE);
    </script>
</body>
</html>
```

Kode 11-4. Ajax-delete.html

- Jelaskan arti kode lain selain 4 pada `http_request.readyState`!
- Mengapa try digunakan pada Ajax?

11.3.5 Percobaan 11-5: Fetch

Ketik kode program di bawah ini:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" />
    <title>Fetch API</title>
</head>
<body>
    <h1>Fetch API</h1>
    <button id="btnAmbilData">Ambil Data</button>
    <div id="tampilanData"></div>
    <script>
        var tampilanData = document.getElementById("tampilanData");

        function FetchAPI() {
            url = 'api.php/mahasiswa';
            fetch(url)
                .then((resp) => resp.json())
                .then(function(data) {
                    tampilkanData(data);
                })
                .catch(function(error) {
                    console.log(JSON.stringify(error));
                });
        }
```

```
        //Fungsi untuk memudahkan buat Node
        function createNode(element) {
            // Membuat tipe elemen yang dilewatkan melalui parameter
            return document.createElement(element);
        }

        //Fungsi untuk menambahkan sub node di bawah Node
        function append(parent, el) {
            // Append parameter kedua ke yang pertama
            return parent.appendChild(el);
        }

        // Fungsi untuk menampilkan data
        function tampilkanData(dataRAW) {
            tampilanData.innerHTML = "Status: " + dataRAW.status;
            // memakai fungsi pembuat elemen
            let table = createNode('table'),
                thead = createNode('thead'),
                th1 = createNode('th'),
                th2 = createNode('th');
            // memasukkan judul
            th1.innerHTML = 'Nama';
            th2.innerHTML = 'NPM';
            // memakai fungsi append ke parameter pertama
            append(thead, th1);
            append(thead, th2);
            append(table, thead);

            for (i = 0; i < dataRAW.data.length; i++) {
                let tr = createNode('tr'),
                    td1 = createNode('td'),
                    td2 = createNode('td');
                // memasukkan data ke dalam data tabel
                td1.innerHTML = dataRAW.data[i].nama;
                td2.innerHTML = dataRAW.data[i].npm;
                // memakai fungsi append ke parameter pertama
                append(tr, td1);
                append(tr, td2);
                append(table, tr);
            }
            append(tampilanData, table);
        }

        document.getElementById("btnAmbilData").addEventListener("click",
FetchAPI);
    </script>
</body>
</html>
```

Kode 11-5. fetch.html

- Apakah kegunaan => (fat arrow)?
- Darimana nilai data pada .then kedua?

11.3.6 Percobaan 11-6: Fetch GET

Ketik kode program di bawah ini:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" />
    <title>Fetch API: GET</title>
</head>
<body>
    <h1>Fetch API: GET</h1>
    <input type="text" name="id" id="id" placeholder="Ketik ID disini">
    <button id="btnAmbilData">Ambil Data</button>
    <div id="tampilanData"></div>
    <script>
        var tampilanData = document.getElementById("tampilanData");

        function FetchAPIGET() {
            var inputID = document.getElementById("id").value;
            url = 'api.php/mahasiswa/' + inputID;
            fetch(url,
                    {
                        method: 'GET',
                        cache: 'reload'
                    }
                )
                .then((resp) => resp.json())
                .then(function(data) {
                    tampilkanData(data);
                })
                .catch(function(error) {
                    console.log(JSON.stringify(error));
                });
        }

        //Fungsi untuk memudahkan buat Node
        function createNode(element) {
            // Membuat tipe elemen yang dilewatkan melalui parameter
            return document.createElement(element);
        }

        //Fungsi untuk menambahkan sub node di bawah Node
        function append(parent, el) {
            // Append parameter kedua ke yang pertama
            return parent.appendChild(el);
        }
```

```
        // Fungsi untuk menampilkan data
        function tampilkanData(dataRAW) {
            tampilanData.innerHTML = "Status: " + dataRAW.status;
            // memakai fungsi pembuat elemen
            let table = createNode('table'),
                thead = createNode('thead'),
                th1 = createNode('th'),
                th2 = createNode('th');
            // memasukkan judul
            th1.innerHTML = 'Nama';
            th2.innerHTML = 'NPM';
            // memakai fungsi append ke parameter pertama
            append(thead, th1);
            append(thead, th2);
            append(table, thead);

            for (i = 0; i < dataRAW.data.length; i++) {
                let tr = createNode('tr'),
                    td1 = createNode('td'),
                    td2 = createNode('td');
                // memasukkan data ke dalam data tabel
                td1.innerHTML = dataRAW.data[i].nama;
                td2.innerHTML = dataRAW.data[i].npm;
                // memakai fungsi append ke parameter pertama
                append(tr, td1);
                append(tr, td2);
                append(table, tr);
            }
            append(tampilanData, table);
        }

        document.getElementById("btnAmbilData").addEventListener("click",
FetchAPIGET);
    </script>
</body>
</html>
```

Kode 11-6. Fetch-GET.html

- Untuk apa cache: reload?
- Jelaskan `dataRAW.data[i].nama`!

11.3.7 Percobaan 11-7: Fetch POST

Ketik kode program di bawah ini:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" />
```

```
    <title>Fetch API: POST</title>
</head>
<body>
    <h1>Fetch API: POST</h1>
    <input type="text" name="nama" id="nama" placeholder="Ketik nama
disini">
    <input type="text" name="npm" id="npm" placeholder="Ketik NPM disini">
    <button id="btnKirimData">Kirim Data</button>
    <div id="tampilanData"></div>
    <script>
        var tampilanData = document.getElementById("tampilanData");

        function fetchAPIPOST() {
            var inputNama = document.getElementById("nama").value;
            var inputNPM = document.getElementById("npm").value;
            var dataInput = "nama=" + inputNama + "&npm=" + inputNPM;
            url = 'api.php/mahasiswa';
            fetch(url, {
                    method: 'POST',
                    headers: {
                        'Content-Type': 'application/x-www-form-urlencoded'
                    },
                    body: dataInput
                })
                .then((resp) => resp.json())
                .then(function (data) {
                    tampilkanData(data);
                })
                .catch(function (error) {
                    console.log(JSON.stringify(error));
                });
        }

        // Fungsi untuk menampilkan data
        function tampilkanData(dataRAW) {
            tampilanData.innerHTML = "Status: " + dataRAW.status + "<br/>";
            tampilanData.innerHTML += "ID data: " + dataRAW.id;
        }

        document.getElementById("btnKirimData").addEventListener("click",
fetchAPIPOST);
    </script>
</body>
</html>
```

Kode 11-7. Fetch-POST.html

- Mengapa harus ada headers?
- Apakah kegunaan body pada fetch?

11.3.8 Percobaan 11-8: Fetch PUT

Ketik kode program di bawah ini:

```
<!DOCTYPE html>
<html>
```

```
<head>
    <meta charset="UTF-8" />
    <title>Fetch API: PUT</title>
</head>
<body>
    <h1>Fetch API: PUT</h1>
    <input type="text" name="id" id="id" placeholder="Ketik ID data yang mau
diganti disini">
    <input type="text" name="nama" id="nama" placeholder="Ketik nama
disini">
    <input type="text" name="npm" id="npm" placeholder="Ketik NPM disini">
    <button id="btnKirimData">Kirim Data</button>
    <div id="tampilanData"></div>
    <script>
        var tampilanData = document.getElementById("tampilanData");

        function fetchAPIPUT() {
            var inputID = document.getElementById("id").value;
            var inputNama = document.getElementById("nama").value;
            var inputNPM = document.getElementById("npm").value;
            var dataInput = "nama=" + inputNama + "&npm=" + inputNPM;
            url = 'api.php/mahasiswa/' + inputID;
            header = {'Content-Type':'application/x-www-form-urlencoded'};
            fetch(url, {
                    method: 'PUT',
                    headers: header,
                    body: dataInput
                })
                .then((resp) => resp.json())
                .then(function (data) {
                    tampilkanData(data);
                })
                .catch(function (error) {
                    console.log(JSON.stringify(error));
                });
        }

        // Fungsi untuk menampilkan data
        function tampilkanData(dataRAW) {
            tampilanData.innerHTML = "Status: " + dataRAW.status + "<br/>";
            tampilanData.innerHTML += "Data -> ID: " + dataRAW.id;
            tampilanData.innerHTML += ", Nama: " + dataRAW.nama;
            tampilanData.innerHTML += ", NPM: " + dataRAW.npm;
        }

        document.getElementById("btnKirimData").addEventListener("click",
fetchAPIPUT);
    </script>
</body>
</html>
```

Kode 11-8. Fetch-PUT.html

- Kenapa headers pada fetchAPIPUT berbeda dengan fetchAPIPOST (kode sebelumnya)?

- Untuk apa catch?

11.3.9 Percobaan 11-9: Fetch DELETE

Ketik kode program di bawah ini:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" />
    <title>Fetch API: DELETE</title>
</head>
<body>
    <h1>Fetch API: DELETE</h1>
    <input type="text" name="id" id="id" placeholder="Ketik ID data yang mau
dihapus disini">
    <button id="btnHapusData">Hapus Data</button>
    <div id="tampilanData"></div>
    <script>
        var tampilanData = document.getElementById("tampilanData");

        function fetchAPIDELETE() {
            var inputID = document.getElementById("id").value;
            url = 'api.php/mahasiswa/' + inputID;
            fetch(url,
                    {
                        method: 'DELETE'
                    }
                )
                .then((resp) => resp.json())
                .then(function(data) {
                    tampilkanData(data);
                })
                .catch(function(error) {
                    console.log(JSON.stringify(error));
                });
        }

        // Fungsi untuk menampilkan data
        function tampilkanData(dataRAW) {
            tampilanData.innerHTML = "Status: " + dataRAW.status + "<br/>";
            tampilanData.innerHTML += "ID data: " + dataRAW.id;
        }

        document.getElementById("btnHapusData").addEventListener("click",
fetchAPIDELETE);
    </script>
</body>
</html>
```

Kode 11-9. Fetch-DELETE.html

- Apakah yang terjadi jika method: 'DELETE' dihapus?
- Mengapa error harus diproses oleh JSON.stringify?

11.4 TUGAS AKHIR

- Gabungkan seluruh kode Ajax dalam 1 berkas html. Lakukan modifikasi yang diperlukan agar PUT dan DELETE tidak lagi menerima input dari tag <input> tetapi harus menggunakan ID dari hidden input yang disematkan pada tabel yang dihasilkan oleh GET. Dan, pastikan berfungsi!
- Gabungkan seluruh kode Fetch API dalam 1 berkas html. Lakukan modifikasi yang diperlukan agar PUT dan DELETE tidak lagi menerima input dari tag <input> tetapi harus menggunakan ID dari hidden input yang disematkan pada tabel yang dihasilkan oleh GET. Dan, pastikan berfungsi!