Fetch JSON

Obyektif:

- Menggunakan metode *fetch*() untuk membuat network request
- Mengambil (extract) data dari jawaban *fetch*
- Membuat JSON Format Untuk File Input
- Membaca File JSON dan mengaplikasikan data json pada program

Daftar Isi

Fungsi fetch()

Metode **fetch()** API merupakan interface penting untuk membuat request ke Server melalui jaringan. Fungsi fetch() tidak menggunakan API XMLHttpRequest, oleh karena itu fetch menjadi pengganti AJAX.

Metode fetch() didefinisikan pada obyek *window* , request dilakukan dari browser. Metode ini memberikan **Promise** sebagai nilai balik.

fetch() tidak digunakan pada Server-Site program seperti nodejs. Fetch adalah request data dari Client (Browser) ke Server dan Server memberikan jawaban atas request tersebut.

Penggunaan Dasar *fetch*

Perhatikan bagaimana metode fetch() digunakan untuk melakukan network request secara sederhana:

Sebuah file teks pada folder **data/** di localhost berisi teks seperti berikut:

```
File: data/catatan.txt
PWA progressive Web Application training di INIXINDO

Program js/readnote.js:

var URL="data/catatan.txt";
fetch(URL)
    .then(function(response){
       if (response.status !== 200) {  //HTTP Status
          console.log('Ada masalah. Status Code: ' +
                                      response.status);
          return;
       }
       return response.text()
    })
    .then( text => console.log(text) )
    .catch( err => console.log(err) );

File: readnote.html

<html>
<head>
</head>
<body>
<script src="js/readnote.js"></script>
```

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

Jawaban di console:

```
PWA progressive Web Application training di INIXINDO
```

Parameter response mempunyai properti status (HTTP-status) dan text(), berisi jawaban dari Server dalam format text.

Bila jawaban tersebut akan ditampilkan pada DOM, maka program berikut dapat melakukan hal tersebut.

```
<html>
<head>
</head>
<body>
<div id='hasil>
</div>
<script src="js/readnote.js"></script>
</body>
</html>
```

Program js/readnote.js:

```
var URL="data/catatan.txt";
fetch(URL)
    .then(function(response){
       if (response.status !== 200) {  //HTTP Status
          console.log('Ada masalah. Status Code: ' +
                                        response.status);
          throw response.statusText;
       }
       return response.text()
    })
    .then( text => {
       let t= document.getElementById('hasil');
       t.textContent = text;
    })
    .catch( err => console.log(err) );
```

Fetch JSON

Selanjutnya fetch mengganti file teks dengan file JSON, ditulis dalam folder data/ *catatan.json.*

File: data/catatan.json

```
{
   "judul":"PWA progressive Web Application",
   "lokasi": "INIXINDO"
}
```

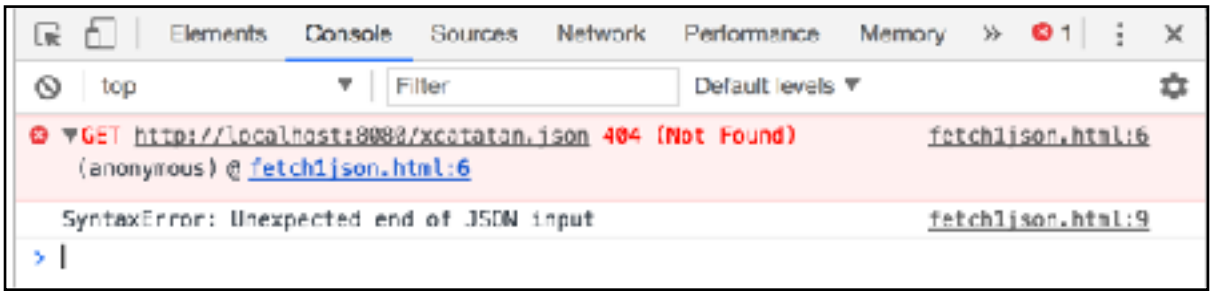
```
var URL="data/catatan.json";
fetch(URL)
    .then(function(response){
       if (response.status !== 200) {  //HTTP Status
          console.log('Ada masalah. Status Code: ' +
                                      response.status);
          throw response.statusText;
       }
       return response.json()
    })
    .then(function(res){
       console.log(res.judul);
       console.log(res.lokasi);
    })
    .catch(function(err){
        console.log(err);
    });
```

Reponse pertama adalah obyek yang dikonversi ke format JSON, kemudian response kedua ditampilkan sebagai json format.

Program disederhanakan dengan menggunakan notasi => .

```
var URL="data/catatan.json";
fetch(URL)
    .then(response=> response.json())
    .then(rsp => {
       console.log(rsp.judul);
       console.log(rsp.lokasi)
    })
    .catch((err)=> console.log(err));
```

```
Simulasi Error: URL="data/xcatatan.json"; // non existence
```



fetch(url)

Metode *fetch()* dengan argumen URL memberikan nilai balik sebuah Promise yang berisi obyek Response dengan rincian sebagai berikut:

```
fetch(URL)
.then (function(response){ // berisi Obyek Response

});
```

Atau dengan fat-arrow:

```
fetch(URL)
.then (response => { // berisi Obyek Response

});
```

Ekstraksi data dari Obyek Response:

```
fetch('catatan.json').then( response= > {
    console.log(response.headers.get('Content-Type'));
    console.log(response.headers.get('Date'));
    console.log(response.status);
    console.log(response.statusText);
    console.log(response.type);
    console.log(response.url);
});
```

Metode pada Obyek Response untuk mengambil data (fetching):

json()	extract json object
text()	extract text string
blob()	extract file-like object
formData()	extract elemen dari form
arrayBuffer()	extract data dari arrayBuffer

Periksa Status Response

Obyek Response hasil fetch dapat diperiksa statusnya sebelum diolah lebih lanjut. Status antara 200-299 berarti bahwa request tersebut sukses, tapi antara 400an atau 500an berarti terjadi masalah.

```
fetch(URL="234%241762") // bad URL
.then(function(response){  //berisi Obyek Response
    console.log(response);
    if (reponse.ok){
        return result.text();    // proses normal
    }
    else { // ada masalah dengan fetch
        console.log(response.status) //status 400an
        return Promise.reject(response.status);
        //returns promise yang gagal, masuk ke catch
    }

})
.then(function(result){
    console.log(result);   //normal case tanpa error
.catch(function(err){      // program lanjut di catch
    console.log("Error: " +  err);
})
```

Melempar Eksepsi

Secara default fetch tidak memberikan Eksepsi jika terjadi error dari jaringan, tapi memberikan response yang normal. Oleh karena itu pada saat response diterima, status **response.ok** akan true bila tidak ada error, dan false bila terjadi error. Oleh karena itu program bereaksi dengan memberikan Promise.reject(response.status) atau melempar ekspsi untuk ditangkap oleh catch.

```
fetch(URL="234%241762") // bad URL
.then(function(response){  //berisi Obyek Response
    console.log(response);
    if (reponse.ok){
        return result.text();    // proses normal
    }
    else { // ada masalah dengan fetch, lempar ke catch
        throw response.statusText;
    }

})
.then(function(result){
    console.log(result);   //normal case tanpa error
.catch(function(err){      // program lanjut di catch
    console.log("Error: " +  err);
})
```

fetch dengan async await

Modern JS menggunakan fat-arrow dan **async** untuk operasi asynchronous. Kendala format ini adalah "bahwa belum semua browser memiliki kapabiltas ini.

```
fetch("catatan.json")
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(e => console.log("Error"))
```

Atau dengan IIFE menggunakan async-await:

```
(async () => {
  try {
    var response = await fetch("catatan.json");
    var data = await response.json();
    console.log(data);
  } catch (e) {
    console.log("Error")
  }
})();
```

Dengan menggunakan fungsi biasa:

```
namafile= "data/catatan.json;
async function f ( namafile) {
  try {
    var response = await fetch( namafile );
    var data = await response.json();
    console.log(data);
  } catch (e) {
    console.log("Error")
  }
};
```

Tugas 3: Fetch-JSON (masukan sebagai Konten 3 di Menu Utama!)

Langkah 1: Persiapan

Salin semua program wisata kuliner sebelumnya, tulis sebagai konten 3. Berbasis dengan aplikasi kuliner tesrebut, maka program berikut adalah pengembangan fetch-JSON.

```
<html>
<head>
<meta name="viewport" content="width=device-width, initial-
scale=1.0, user-scalable=no" />

<link rel="stylesheet" type="text/css" href="css/peta.css">

<link rel="stylesheet" href="https://unpkg.com/leaflet@1.3.4/dist/
leaflet.css" integrity="sha512-puBpdR0798OZvTTbP4A8Ix/
l+A4dHDD0DGqYW6RQ+9jxkRFclaxxQb/SJAWZfWAkuyeQUytO7+7N4QKrDh+drA=="
crossorigin=""/>

<script src="https://unpkg.com/leaflet@1.3.4/dist/leaflet.js"
integrity="sha512-
nMMmRyTVoLYqjP9hrbed9S+FzjZHW5gY1TWCHA5ckwXZBadntCNs8kEqAWdrb9O7rxbC
aA4lKTIWjDXZxflOcA==" crossorigin=""></script>

</head>
<body>
    <div class='container'>
        <div class='peta' id='mapid'> </div>
        <div class='gambar' id='gmb'>
                Gambar disini
        </div>
        <div class='review' id='review'>
                Tulis review disini
        </div>
    </div>
<script>
var mymap = L.map('mapid').setView([-6.221028, 106.791434], 16);

L.tileLayer('https://api.tiles.mapbox.com/v4/{id}/{z}/{x}/{y}.png?
access_token={accessToken}', {
    attribution: 'Map data &copy; <a href="https://
www.openstreetmap.org/">OpenStreetMap</a> contributors, <a
href="https://creativecommons.org/licenses/by-sa/2.0/">CC-BY-SA</a>,
Imagery © <a href="https://www.mapbox.com/">Mapbox</a>',
    maxZoom: 20,
    id: 'mapbox.streets',
    accessToken:
'pk.eyJ1IjoiaWZpa2FyaWZpbiIsImEiOiJjamxndm83cTQwZjYwM3BvMHBma3dndGJq
In0.6LyWJiWv-yvp7mNPby'
}).addTo(mymap);
```

```
</script>
<script src="js/peta.js"></script>
</body>
</html>
```

Program ini tidak diubah. Yang dimodifikasi adalah javascript berikut:

File: js/peta.js

```
function findLocation(x,y) {
    // console.log(x,y);
    for (var i=0; i< places.length;i++) {
        if (places[i].lokasi[0]==x &&
            places[i].lokasi[1]==y) {
            return i;
        }
    }
    return -1;
}
function showLocation(e) {
    //console.log("you clicked " + e.latlng.lat + " dan "+
e.latlng.lng);
    let ix= findLocation(e.latlng.lat,e.latlng.lng);
    if (ix >=0) {
        img.src= places[ix].gambar;
        par.textContent=places[ix].review;
    }
}

let gmb= document.getElementById("gmb");
let rev= document.getElementById("review");
let img= document.createElement('img');
let par= document.createElement('p');
gmb.appendChild(img);
rev.appendChild(par);

let r0="restoran spanyol di jakarta yang dekat dengan kantor saya";
let r1="warung kopi cita rasa yang sangat tinggi dengan harga yang
murah";
let r2="Ikan bakar kualitas tinggi, hampir gosong tapi belum";
let r3="Steak lokal harga impor, 200gr dan 300gr mentah ";
let r4="seafood international lobster, king crabs, cumi, kerang,
semua ada";
let places= [
    {"lokasi": [-6.221028, 106.791434], "sponsor" : "Resto Spanyol",
"gambar":"images/planB.jpg","review": r0},
    {"lokasi": [-6.219912, 106.791239], "sponsor" : "Warung Kopi",
"gambar":"images/warkop.jpg","review": r1},
    {"lokasi": [-6.220529, 106.789848], "sponsor" : "Pondok Ikan
Bakar", "gambar":"images/ikan_bakar.jpg","review": r2},
    {"lokasi": [-6.222977, 106.789152], "sponsor" : "STEAK cow",
"gambar":"images/steak.jpg","review": r3},
```

```
    {"lokasi": [-6.222043, 106.791070], "sponsor" : "Rupa-rupa
Seafood!!", "gambar":"images/seafood.jpg","review": r4}
];

for (var p of places) {
    var marker= L.marker(p.lokasi).addTo(mymap)
     .bindPopup(p.sponsor);
    marker.on('click', showLocation);
}
```

Pastikan program tersebut berjalan dengan baik, sebelum modifikasi dibuat

Langkah 2: data/peta.json

Program ini menggunakan Array "places" yang memuat lokasi, sponsor, gambar dan review. Struktur data keempat elemen ini akan dijadikan JSON dan menyimpannya sebagai file **peta.json pada folder data/ .**

JSON selalu dimulai dengan obyek { } kemudian ditandai dengan nama elemen atau langsung daftar elemen. Pada contoh berikut places adalah elemen pertama.

File: data/peta.json

```
{
  "places" : [    ]
}
```

places adalah array dari obyek (ada 5 obyek);

```
{
   "places" : [
      {
         "lokasi": [-6.221028, 106.791434],
         "sponsor" : "Resto Spanyol",
         "gambar":"images/planB.jpg",
         "review": r0
     }
  ]
}
```

Review tidak memerlukan variable r0 karena teks dapat ditulis langsung pada kolom tersebut.

```
{
   "places" : [
    {
         "lokasi": [-6.221028, 106.791434],
         "sponsor" : "Resto Spanyol",
```

```
          "gambar":"images/planB.jpg",
          "review": "restoran spanyol terbaik di Jakarta"
    }
  ]
}
```

Selanjutnya dilengkapi elemen dari JSON file tersebut.

```
{
  "places":[
  {
    "lokasi": [-6.221028, 106.791434],
    "sponsor" : "Resto Spanyol",
    "gambar":"images/planB.jpg",
    "review": "restoran spanyol terbaik di Jakarta"
  },
  {
    "lokasi": [-6.219912, 106.791239],
    "sponsor" : "Warung Kopi",
    "gambar":"images/warkop.jpg",
    "review": "warung kopi cita rasa yang sangat tinggi dengan harga
murah"
  }
]}
```

Langkah 3: Persiapan Fetch

Untuk mengakomodasi program baru, maka hapus array dan review di peta.js.

File: js/peta.js

```
function findLocation(x,y) {
    // console.log(x,y);
    for (var i=0; i< places.length;i++) {
        if (places[i].lokasi[0]==x &&
            places[i].lokasi[1]==y) {
            return i;
        }
    }
    return -1;
}
function showLocation(e) {
    //console.log("you clicked " + e.latlng.lat + " dan "+
e.latlng.lng);
    let ix= findLocation(e.latlng.lat,e.latlng.lng);
    if (ix >=0) {
        img.src= places[ix].gambar;
        par.textContent=places[ix].review;
```

```
    }
}

let gmb= document.getElementById("gmb");
let rev= document.getElementById("review");
let img= document.createElement('img');
let par= document.createElement('p');
gmb.appendChild(img);
rev.appendChild(par);

const URL="data/peta.json";

........ // disini ambil file peta.json dengan fetch

for (var p of places) {
    var marker= L.marker(p.lokasi).addTo(mymap)
     .bindPopup(p.sponsor);
    marker.on('click', showLocation);
}
```

Langkah 4: Buat fetch

Pada tempat program ..... masukan program membaca json dengan fetch.

```
const URL="data/peta.json";
fetch(URL)
    .then(function(response){
       if (response.status !== 200) {  //HTTP Status
          console.log('Ada masalah. Status Code: ' +
                                      response.status);
          throw response.statusText;
       }
       return response.json()
    })
    .then ( resp => {
        let places= resp.places;

    })
    .catch(function(err){
        console.log(err);
});
```

Namun program ini bermasalah, karena asynchronous fetch tidak memberi nilai ke program utama. Places tersebut tidak dapat diberikan sebagai parameter untuk digunakan lebih lanjut sebagai variable.

Bila usaha berikut dicoba:

```
let places=[ ];
fetch (URL)
  .then ( resp => return resp.json() )
  .then (
      ....
      places = resp.places;
  )
```

Setelah program fetch aktif, namun nilai variable places tetap kosong. Oleh karena itu diperlukan mekanisme lain, yaitu penyimpanan melalui **localStorage**.

localStorage menyimpan data dengan **localStorage.setItem**(nama-data, nilai-data), dan mengambil nilainya dengan **nilai= localStorage.getItem(nama-data).**

Namun kendala baru, bahwa localStorage hanya menyimpan 1 obyek string saja. Oleh karena itu, digunakan **JSON.stringify()** untuk mengubah format JSON menjadi string, dan kemudian program utama menggunakan **JSON.parse()** untuk mengembalikan format JSON tersebut.

```
localStorage.setItem('places', JSON.stringify(resp.places));
let places= JSON.parse( localStorage.getItem('places'));
```

Program: peta.js (versi 1)

```
function findLocation(x,y) {
    // console.log(x,y);
    for (var i=0; i< places.length;i++) {
        if (places[i].lokasi[0]==x &&
            places[i].lokasi[1]==y) {
            return i;
        }
    }
    return -1;
}
function showLocation(e) {
    //console.log("you clicked " + e.latlng.lat + " dan "+
e.latlng.lng);
    let ix= findLocation(e.latlng.lat,e.latlng.lng);
    if (ix >=0) {
        img.src= places[ix].gambar;
        par.textContent=places[ix].review;
    }
}

let gmb= document.getElementById("gmb");
let rev= document.getElementById("review");
let img= document.createElement('img');
let par= document.createElement('p');
gmb.appendChild(img);
rev.appendChild(par);
```

```
const URL="data/peta.json";
fetch(URL)
    .then(function(response){
       if (response.status !== 200) {  //HTTP Status
          console.log('Ada masalah. Status Code: ' +
                                        response.status);
          return;
       }
       return response.json()
    })
    .then ( resp => {
        let places= resp.places;
        localStorage.setItem(...)
    })
    .catch(function(err){
        console.log(err);
});


let places= JSON.parse(....) ;
for (var p of places) {
    var marker= L.marker(p.lokasi).addTo(mymap)
     .bindPopup(p.sponsor);
    marker.on('click', showLocation);
}
```

Setelah **JSON.parse()** dilakukan, maka variable **places** berisi array dengan lokasi, sponsor, gambar dan review.

Tugas 4:

```
Gunakan async ...wait sebagai pengganti fetch().then().then( ).

async function f(url){
    try {
        const resp= await(fetch(url));
        const resp2= await resp.json();
        localStorage.setItem(....)

    }
    catch(err){
        console.log(err);
    }
}

const URL="data/peta.json";
f(URL);
let places = JSON.parse(localStorage.getItem (.......));
```