Rejseplan API Guide

Contents

BaseURL

For at kunne få data fra rejseplanens API om lokationer og rejser, skal det først kaldes på. Rejseplanens API virker på den måde af man bruger hvad der er kaldt et Base URL, hvilket et den første del af en APIs URL, til at kunne få fat på deres data.

Base URL fra Rejseplanens URL kan findes i opgaven, på dette tidspunkt 15-01-2020 er det som følger:

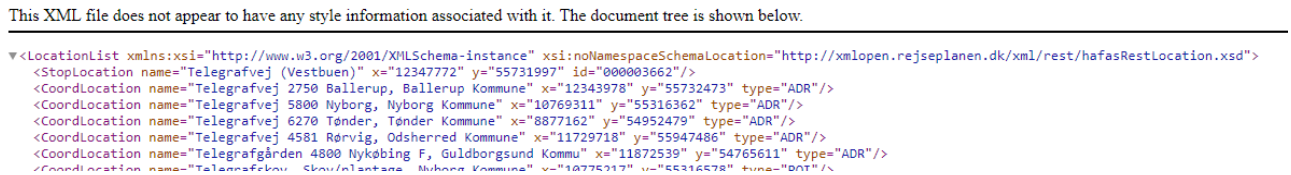
http://xmlopen.rejseplanen.dk/bin/rest.exe/

Dette kan normalt ikke findes uden at få kontakt til udviklerne af Rejseplanens API.

Dette URL bruges i kombination med nogen parametre til at få fat på, og returnere JSON eller XML data fra Rejseplanen. F.eks. for at få fat på en lokation bruger man:

http://<baseurl>/location?input=user%20input

Hvor ”user%20input” er brugerens input, dette kunne f.eks. være ”Telegrafvej” som ville få Rejseplanens API til at returnere XML data om lokationer der kunne matche ”Telegrafvej”. Man kan teste dette ved at bare skrive URL’et ind i et browser vindue, og se om det giver resultatet man søger.



For at få det returneret som JSON i stedet, tilføjer man argumentet:

format=json

Hvilke ville lave linket til dette:

http://<baseurl>/location?input=user%20input&format=json

Man tilføjer argumentet med ”&” tegnet. Dette fortæller Rejseplanen at vi gerne vil have data returneret i JSON format.


```
{
"LocationList":{
  "noNamespaceSchemaLocation":"http://xmlopen.rejseplanen.dk/xml/rest/hafasRestLocation.xsd",
  "StopLocation":[{
    "name":"Telegrafvej (Vestbuen)",
    "x":"12347772",
    "y":"55731997",
    "id":"000003662"
    },{
    "name":"Telehøjen (Odense Kommune)",
    "x":"10456405",
    "y":"55360688",
    "id":"461112900"
    },{
    "name":"Ud 212 Telebus zone (Fynbovej)",
    "x":"11217360",
    "y":"54927147",
    "id":"000052308"
    },{
```

Brug BaseURL med JavaScript

For at få fat i JSON data i JavaScript Koden, skal vi bruge en funktion kaldet ”fetch”, hvilket får fat i data asynkront, dette kan ikke gøres synkront siden at vi skal vente på et tredje parti til at få fat i Data, i dette tilfælde er det Rejseplanens Database. Her er et eksempel på hvordan ”fetch” bruges:

```
function getData() {
  let baseUrl = 'http://xmlopen.rejseplanen.dk/bin/rest.exe/';
  let argument = 'location?input=';
  let location = 'Telegrafvej'

  let apiURL = baseUrl.concat(argument, location, '&format=json')

  fetch(apiURL)
  .then (response => {
    return response.json();
  })
  .then (data => {
    console.log(data);
  })
}
```

Figur 1 JavaScript af en simpel fetch

Hvis vi kalder det script med en knap på i en normal Google Chrome browser vil du nok opdage en fejl.

Access-Control-Allow-Origin

Når dette stykke eksempel kode bliver kørt i en normal Google Chrome browser, vil du opdage at i konsollen bliver der vist en fejl.

```
Access to fetch at 'http://xmlopen.rejseplanen.dk/bin/rest.exe/locati  index.html:1
on?input=Telegrafvej&format=json' from origin 'null' has been blocked by CORS
policy: No 'Access-Control-Allow-Origin' header is present on the requested
resource. If an opaque response serves your needs, set the request's mode to 'no-
cors' to fetch the resource with CORS disabled.
▸Uncaught (in promise) TypeError: Failed to fetch                    index.html:1
▸Cross-Origin Read Blocking (CORB) blocked cross-origin response  rejseplan.js:266
http://xmlopen.rejseplanen.dk/bin/rest.exe/location?input=Telegrafvej&format=json
 with MIME type application/json. See https://www.chromestatus.com/feature/56297098
24032768 for more details.
```

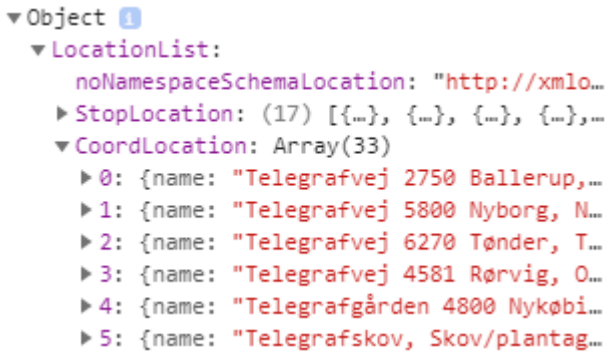
Figur 2 Access-Control-Allow-Origin Fejl

For at overkomme dette, uden at kontakte udviklerne af Rejseplanens API, kan man køre Google Chrome uden web security. Åben Kør eller ”Run” i Windows, ved at søge på den med søge funktionen, og indtast dette:

chrome.exe --user-data-dir=CChrome dev session --disable-web-security

Dette åbner et nyt Google Chrome Vindue, uden web sikkerhed i Developer mode. Brug helst kun dette vindue til at kode og teste hjemmesiden.

Kør koden igen, og den burde give en besked i konsollen med et objekt der indeholder JSON Data.



Figur 3 Console.log af JSON data fra Rejseplanens API

Dette objekt indeholder nogen arrays kaldt ”StopLocation” og ”CoordLocation”, baseret på hvad der er brugt som input, kan kun den ene eller anden dukke op, det skal der også tages forhold til i koden f.eks. sådan:

```
if (typeof data.LocationList.StopLocation !== 'undefined') {
  console.log(data.LocationList.StopLocation);
} else if (typeof data.LocationList.CoordLocation !== 'undefined') {
  console.log(data.LocationList.CoordLocation);
}
```

Figur 4 Se om et array findes

Her ser vi om ”StopLocation” eller ”CoordLocation” er undefined eller ej. Foroven kan der også ses hvordan man får fat i StopLocation arrayet, den ligger egentlig i et andet array kald LocationList, hvilket er et array af arrays.

Hvis vi nu ville have det første resultat i ”stopLocation” ville vi gøre sådanne:

```
console.log(data.LocationList.StopLocation[0]);
```

Figur 5 Første index af StopLocation arrayet

Console.log er en livreder hvis man skal finde ud af lige præcis hvad der bliver returneret af Rejseplanens API, og finde frem til hvad de forskellige arrays hedder, hvor de er i forhold til hindanden, og hvad de indeholder.

Async

Hvis vi kigger på kode eksemplet igen, ser vi noget kaldt ”then”:

```
8     fetch(apiURL)
9     .then (response => {
10      return response.json();
11    })
12    .then (data => {
13      console.log(data);
14    })
15  }
```

Figur 6 Fetch og .then eksempel

”then” venter på Data asynkront, fetch returnere noget kaldt et ”promise” hvilket løser til ”response” til den anmodning fra fetch. Ved den første ”then” tager vi fat i HTML responsen og tager fat i JSON data fra dens ”body” med ”.json()” vi tager så det returneret JSON objekt under den næste ”then” og skriver den i konsollen.

Hvis man vil tag den data og bruge den i en anden funktion kan man gøre så ledes:

```
let response = await fetch(apiURL);
let data = await response.json();
```

Figur 7 Et eksempel på Await

Det er vigtig at huske at fetch returnere asynkront, og vi kan kun få fat i dataen i en asynkron funktion, await bruger vi til at vente på at dataen bliver returneret asynkront, så vi kan bruge den andre steder.

Med then behøver man ikke putte koden i en asynkron funktion, til gengæld kan vi ikke sende data videre, vi skal behandle det hele inde i then.

En asynkron funktion erklæres sådanne:

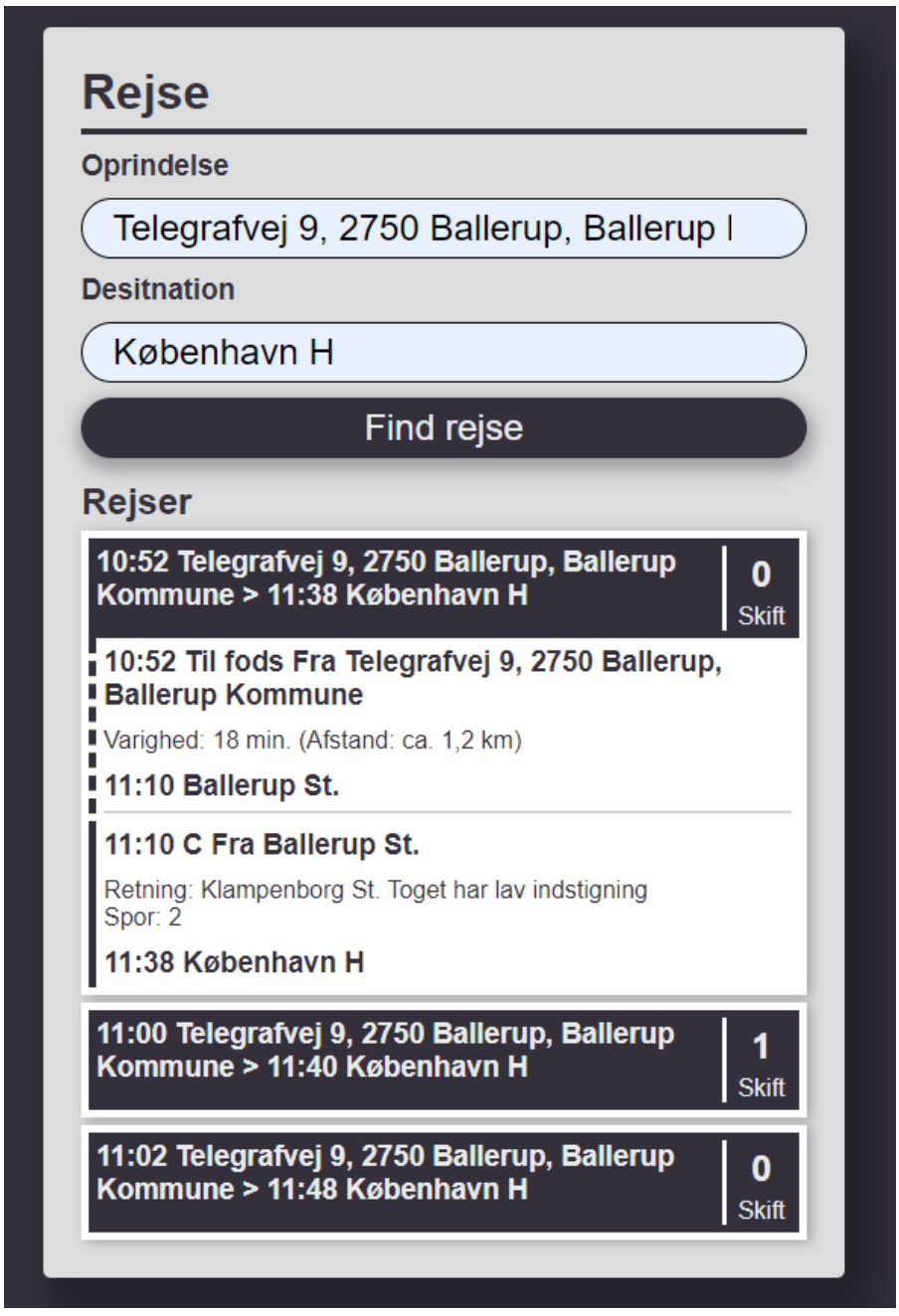
```
async function getData() {
```

Figur 8 Oprettelsen af en asynkron funktion

Og den er obligatorisk for at bruge await.

HTML med JavaScript

For en idé af hvordan siden kunne se ud:



Figur 9 Rejseplan Hjemmeside Eksempel

Her er Oprindelse, Destination og Find Rejse alle en del af en form, hvilke bliver brugt af JavaScript til at finde ud af hvor brugeren gerne vil hen, og hvor de er henne. Kasserne med oprindelse, destination og tid, er oprettet i JavaScript, der er flere måder at lave HTML og addere til en side i JavaScript, her er et eksempel på hvordan dette blev opnået:

Et meget simpelt stykke HTML der bliver brugt til eksemplet:

```html
<!DOCTYPE html>
<html lang="da" dir="ltr">
<head>
  <meta charset="utf-8">
  <title>Rejseplan Api</title>
  <link rel="stylesheet" href="stylesheet.css">
  <script src="htmlJS.js" charset="utf-8"></script>
</head>
<body>
  <div id="pageContainer">
    <script type="text/javascript">
      BuildPage();
    </script>
  </div>
</body>
```

Figur 10 Et eksempel på HTML til JavaScript

Og et eksempel på hvordan man kan bygge HTML med JavaScript:

```javascript
function BuildPage() {
  let container = document.getElementById('pageContainer');

  let box = document.createElement('div');
  container.appendChild(box);

  let boxHeader = document.createElement('h1');
  let boxHeaderText = document.createTextNode('Her er Titlen');
  boxHeader.appendChild(boxHeaderText);

  let boxParagraf = document.createElement('p');
  let boxParagrafText = document.createTextNode('Dette er en paragraf');
  boxParagraf.appendChild(boxParagrafText);

  box.appendChild(boxHeader);
  box.appendChild(boxParagraf);
}
```

Figur 11 Eksemplet på at oprette HTML med JavaScript

BuildPage bliver kaldt nederest på HTML siden. Koden foroven giver dette resultat:

Figur 12 Resultatet af Javascript (Det gule er fra et CSS stylsheet der styler pageContainer)

Princippet er at man opretter et Element af en type f.eks. ”div” og så sætter man den på som barn af et andet objekt, i dette tilfælde et div kaldt ”pageContainer”. Hvis elementet skal have tekst, skal man oprette en ”text Node” og sætte den som barn på elementet man vil give tekst.

Man kan også give dem klasser således:

```
function BuildPage() {
  let container = document.getElementById('pageContainer');

  let box = document.createElement('div');
  box.classList.add('box');
  container.appendChild(box);
```

Figur 13 Eksempel på hvordan man kan give et element en klasse

Dette giver elementet ”box” klassen ”box” så den kan blive stylet senere.

Ekstra Information

Mellemrum

Der kan ikke være mellemrum i et URL så for at undgå fejl, skal mellemrum i bruger input erstattes med ”%20” hvilket computeren læser som mellemrum, her er hvordan man kunne gøre det:

```
//Erstat mellemrum i location med %20 til URL
location = location.replace(/ /g, '%20');
let url = baseUrl.concat(location, '&format=json');
```

Figur 14 Eksempel på at erstatte mellemrum med %20

Finde rejse

For at finde en rejse skal man bruge BaseURL som normalt, og så tilføje ”trip?”, efter trip skal der enten et originID, hvilket kan findes i StopLocation arrayet, hvis man ikke kan få fat i det, skal man bruge originCoordX, originCoordY og originName, hvilket kan findes i CoordLocation arrayet.

Man skal også bruge destinationID eller som før skal man bruge destCoordX, destCoordY og destName, hvilket vil resultere i et URL lidt som dette:

<baseurl>trip?originID=12345&destCoordX=1234&destCoordY=1234&destName=Dest%20Navn&format=json

Dette returnere hvad en række rejser hvilke der kaldes ”Trip”, for hvert trip er der et ”Leg” hvilke er metoderne der skal benyttes for at brugeren kommer til deres endelige destination.

For at få fat i det kan man f.eks. gøre sådan med den data man får i returneret:

```
if (typeof data.TripList.Trip !== 'undefined') {
  let tripArray = data.TripList.Trip;
```

Figur 15 Eksempel på at få data fra TripList

Her bliver der også set om der er blevet fundet nogen rejser eller ej.

Bilag

Referencer og kilder

Rejseplan API: https://help.rejseplanen.dk/hc/da/articles/214174465-Rejseplanens-API

Rejseplan API Dokumentation: https://help.rejseplanen.dk/hc/da/article_attachments/115002672369/ReST_documentation_Rejseplanen_Latest.pdf

JS Async Function: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/async_function

JS Fetch: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch