

# json-filer

...

# Agenda

1. Opfølgning på torsdagens opgaver
2. JSON-filer
3. Hent JSON-data ind fra fil
4. Asynkrone events i javascript (promises)
5. Data fra google sheets til json
6. Plugin til visning af json-filer i Chrome
7. Opgave: persondata
8. Intro til temaopgaven

# JSON-filer

**JSON** = **J**ava**S**cript**O**bject**N**otation

Et objekt, eller et array af objekter kan gemmes som en **json-fil**. En fil med filtypen **.json**

F.eks. kan man tage dette array:

```
let undervisere=[
  {"navn":"Alan Engelhardt","email":"ale","github": "alan-engelhardt"},
  {"navn":"Klaus Mandal","email":"klmh", "github": "MondaleMondale"},
  {"navn":"Martin Bregnhøi", "email":"mabe", "github": "martinbregnhoi"},
  {"navn":"Kamilla Viktor", "email":"kvi", "github": "kvikea"}
];
```

...og paste dets værdi ind i en ny fil og gemme den under navnet ex. **undervisere.json**.

json-filer kan hentes ind i html-dokumenter med javascript

JSON is based on JavaScript object literals The literal value of an object exposes the properties or attributes in a way which we can see (and read). By Lindsay Bassett: February 18, 2016: <https://www.oreilly.com/ideas/json-is-based-on-javascript-object-literals>  
What is JSON? Introduction (Part 1/4): [https://www.youtube.com/watch?v=BGfmpvM4Zp0&list=PLfdiltiRHWKQOby9HEyYtB\\_Y9g7z6yL](https://www.youtube.com/watch?v=BGfmpvM4Zp0&list=PLfdiltiRHWKQOby9HEyYtB_Y9g7z6yL)

# JSON-filer - hvorfor??

Når data gemmes i datafiler, opnår vi at **adskille** html-kode og data:

- Dokumentets struktur i **html-fil**
- Layout i **css-fil**
- Js i **js-fil**
- **Data** i **json-fil**

Json-filer bruges til **udveksling af data**

# Læg i json:

undervisere.json

```
{
  "fornavn": "Alan",
  "efternavn": "Engelhardt",
  "email": "ale@kea.dk",
  "github": "alan-engelhardt"
},
{
  "fornavn": "Klaus",
  "efternavn": "Mandal Hansen",
  "email": "klmh@kea.dk",
  "github": "mondalemondale"
}, {
  "fornavn": "Kamilla",
  "efternavn": "Viktor",
  "email": "kvi@kea.dk",
  "github": "kvikea"
}, {
  "fornavn": "Martin",
  "efternavn": "Bregnhøi",
  "email": "mabe@kea.dk",
  "github": "martinbregnhøi"
}
}
```

```
<body>
```

```
<section class="data-container"></section>
```

```
<script>
```

```
document.addEventListener("DOMContentLoaded", hentJson);
```

```
let undervisere;
```

```
//Hent JSON fil i asynkron funktion
```

```
async function hentJson() {
```

```
  //henter data filen
```

```
  const myJson = await fetch("undervisere.json");
```

```
  // den hentede data skal tolkes som json
```

```
  undervisere = await myJson.json();
```

```
  //kald funktion der viser data i DOM
```

```
  visUndervisere();
```

```
}
```

Her hentes json filen.  
Det kommer vi til om  
lidt...

Om json:

<https://www.oreilly.com/ideas/json-is-based-on-javascript-object-literals>

[https://www.youtube.com/watch?v=BGfmpvM4Zp0&list=PLfdiltiRHWKQOby9HEyYtB\\_Y9g7z6yL](https://www.youtube.com/watch?v=BGfmpvM4Zp0&list=PLfdiltiRHWKQOby9HEyYtB_Y9g7z6yL)

# Øvelse 1: Opret json-fil

1. I mappen **undervisningsopgaver**, laver du en ny undermappe, **json** til dagens opgaver
2. Tag en kopi af **06-visMedTemplate.html** fra i går, og kald kopien: **01-dyrIjson.html**
3. Tag værdien af arrayet **alleDyr**, og læg det ud i en json-fil, **dyr.json**

**Nu fungerer siden ikke længere - vi har brug for at hente data ind igen fra json-filen - det kommer vi til nu!**

# Hent og vis json-fil

...

# Hent JSON-fil

Javascript kan hente en json-fil ind vha en særlig **async function**

I funktionen hentes filmen med **fetch**

man kan få js til at vente med at gå videre, til filen er indlæst med **await**

Erklæring af variabel  
(ingen værdi endnu)

**async**-function

```
let undervisere;  
//Hent JSON fil i asynkron function  
async function hentJson() {  
  //henter data filen  
  const myJson = await fetch("undervisere.json");  
  // den hentede data skal tolkes som json  
  undervisere = await myJson.json();  
  //kald funktion der viser data i DOM  
  visUndervisere();  
}
```

**fetch**: hent filen  
**await**: vent med at  
fortsætte, til den er hentet

Når alle personer ligger i arrayet, kan  
vi kalde en funktion, der viser arrayet

Variablen **undervisere** gemmer  
json-filens data-indhold i json  
format.  
**await**: vent med at fortsætte, til  
indholdet er hentet



# Hele koden:

## undervisere.json

```
{
  "fornavn": "Alan",
  "efternavn": "Engelhardt",
  "email": "ale@kea.dk",
  "github": "alan-engelhardt"
},
{
  "fornavn": "Klaus",
  "efternavn": "Mandal Hansen",
  "email": "klmh@kea.dk",
  "github": "mondalemondale"
}, {
  "fornavn": "Kamilla",
  "efternavn": "Viktor",
  "email": "kvi@kea.dk",
  "github": "kvikea"
}
, {
  "fornavn": "Martin",
  "efternavn": "Bregnhøi",
  "email": "mabe@kea.dk",
  "github": "martinbregnhøi"
}
}
```

```
<div id="liste"></div>
<template>
  <div class="underviser">
    <h2></h2>|
    <p>email: <span></span>@kea.dk</p>
    <p><a href="http://github.com/" ></a></p>
  </div>
</template>
<script>
  let undervisere=[];
  document.addEventListener("DOMContentLoaded", start);
  function start(){
    const dest=document.querySelector("#liste");
    const temp=document.querySelector("template");
    async function getJson(){
      let jsonData = await fetch("undervisere.json");
      undervisere=await jsonData.json();
      visUndervisere();
    }
    function visUndervisere(){
      undervisere.forEach(underviser =>{
        const klon=temp.cloneNode(true).content;
        klon.querySelector("h2").textContent = underviser.fornavn;
        klon.querySelector("span").textContent = underviser.email;
        klon.querySelector("a").textContent = underviser.github;
        klon.querySelector("a").href += underviser.github;
        dest.appendChild(klon);
      })
    }
    getJson();
  }
</script>
```

# Asynkrone events / promises

...

# Ajax og promises

**Fetch** bruges til at hente en fil ind i js.

Det er en operation, som tager laaang tid i js's målestok.

Derfor vil programmet bare drøne videre - hvis man ikke får det til at vente.

**Await** får funktionen der henter filen til at tage en pause, indtil filen er hentet, hvis den er erklæret som en async function.

Det kaldes et **promise**, når en js-funktion er blevet sat på pause

Et asynkront kald kaldes også et **AJAX -kald**(Asynkron JavaScript XML)

# Øvelse 2: Indlæs og vis json

Arbejd videre med [01-dyrljson.html](#) :

- dyrene skal indlæses fra json-filen og vises på siden

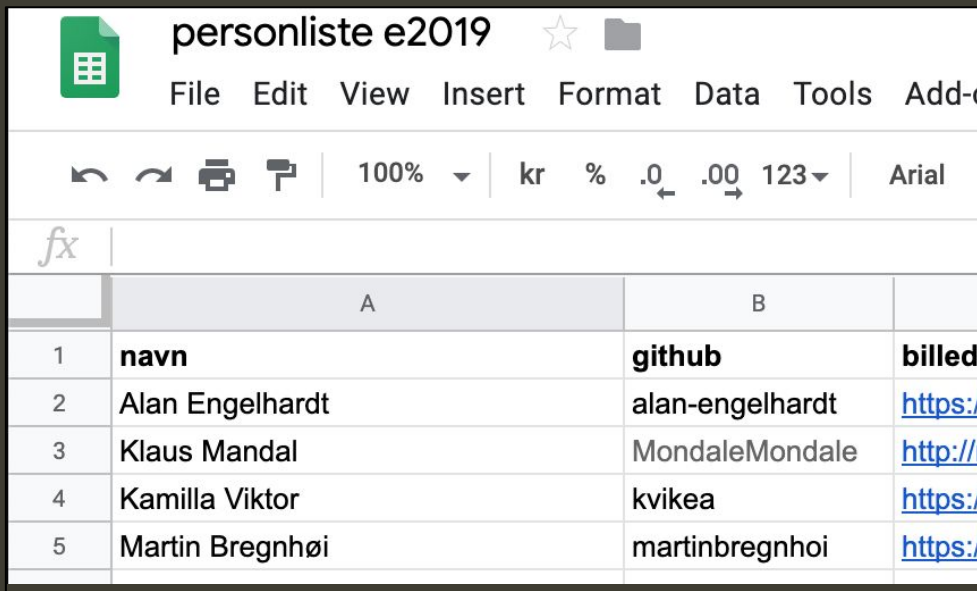
<b>frø</b>  Type: amfibie Levested: vandhullet
<b>gris</b>  Type: pattedyr Levested: grisefarmen
<b>ræv</b>  Type: pattedyr Levested: skoven
<b>sild</b>  Type: fisk Levested: havet
<b>krocodile</b>  Type: krybdyr Levested: floden

# Fra google sheets til json-fil

...

# Google sheet

skærbilleder: sheet og json



The screenshot shows a Google Sheet titled "personliste e2019". The sheet has three columns: "navn", "github", and "billed". The data is as follows:

	A	B	
1	navn	github	billed
2	Alan Engelhardt	alan-engelhardt	<a href="https://avatars0.githubusercontent.com/u/58">https://</a>
3	Klaus Mandal	MondaleMondale	<a href="http://">http://</a>
4	Kamilla Viktor	kvikea	<a href="https://">https://</a>
5	Martin Bregnhøi	martinbregnhoi	<a href="https://">https://</a>

```
    },
    - content: {
      type: "text",
      $t: "github: alan-engelhardt, billede: https://a
        historie: Til september går jeg på pension!! :))"
    },
    - link: [
      - {
        rel: "self",
        type: "application/atom+xml",
        href: "https://spreadsheets.google.com/feeds/

      ],
    - gsx$navn: {
      $t: "Alan Engelhardt"
    },
    - gsx$github: {
      $t: "alan-engelhardt"
    },
    - gsx$billede: {
      $t: "https://avatars0.githubusercontent.com/u/58"
    },
    - gsx$alder: {
      $t: "272"
    },
    - gsx$skøn: {
      $t: "m"
    },
    - gsx$email: {
      $t: "ale@kea.dk"
    },
    - gsx$hold: {
      $t: "b"
    },
    - gsx$historie: {
      $t: "Til september går jeg på pension!! :))"
    },
  },
  - {
    - id: {
```

# Fra sheet til json

sheetet skal være **public**: vælg **file > publish to the web**

url-adresse: <https://spreadsheets.google.com/feeds/list/her sættes ID'et ind/od6/public/values?alt=json>

sheetets id tages fra dets url:



[https://spreadsheets.google.com/feeds/list/1Xge7slZ9dEOTCn1Yxl3OE4xgvrXOL8Y\\_iu3WN1yGB1U/od6/public/](https://spreadsheets.google.com/feeds/list/1Xge7slZ9dEOTCn1Yxl3OE4xgvrXOL8Y_iu3WN1yGB1U/od6/public/)

hvor skal url-adressen ind i javascript:

```
let jsonData = await
```

```
fetch("https://spreadsheets.google.com/feeds/list/1Xge7slZ9dEOTCn1Yxl3OE4xgvrXOL8Y_iu3WN1yGB1U/od6/public/values?alt=json");
```

# Plugin til visning af json-filer

På [Chrome webstore](#), skal du hente en extension, som formatterer jsonfiler pænt.

Udvid fx med denne extension: [JSONView](#)

## TEST

[https://spreadsheets.google.com/feeds/list/1Xge7slZ9dEOTCn1Yxl3OE4xgvrXOL8Y\\_iu3WN1yGB1U/od6/public/values?alt=json](https://spreadsheets.google.com/feeds/list/1Xge7slZ9dEOTCn1Yxl3OE4xgvrXOL8Y_iu3WN1yGB1U/od6/public/values?alt=json)



# Den søde JSON vi selv har lavet.

Hvis vi kigger på vores dyre json fil er den i et niveau og de 5 elementer vi forEach'er (itererer) over er tydelige.

```
minJson.forEach((dyr) => {
```

```
    dyr.navn;  
    dyr.type;  
    dyr.billede;  
    dyr.levested;
```

```
}
```

```
[  
  {  
    "navn": "frø",  
    "type": "amfibie",  
    "billede": "http://helf-kea.dk/test/frog.png",  
    "levested": "vandhullet"  
  },  
  {  
    "navn": "gris",  
    "type": "pattedyr",  
    "billede": "http://helf-kea.dk/test/gris.png",  
    "levested": "grisefarmen"  
  },  
  {  
    "navn": "ræv",  
    "type": "pattedyr",  
    "billede": "http://helf-kea.dk/test/fox.png",  
    "levested": "skoven"  
  },  
  {  
    "navn": "sild",  
    "type": "fisk",  
    "billede": "http://helf-kea.dk/test/sild.jpg",  
    "levested": "havet"  
  },  
  {  
    "navn": "krokodille",  
    "type": "krybdyr",  
    "billede": "http://helf-kea.dk/test/kroko.jpg",  
    "levested": "floden"  
  }  
]
```

# 1 Den nasty rå JSON fra Google sheet'et!

Det kan være svært at presse JSON'en her ind et slide, så kik [her](#) også. Vi skal ned til det array af objekter vi kan foreache over og de gemmer sig i objektet `entry`. `Entry` ligger i `feed` der ligger i vores JSON objekt:

```
minJson.feed.entry.forEach((person) => {
```

```
}
```

```
version: "1.0",
encoding: "UTF-8",
feed: {
  xmlns: "http://www.w3.org/2005/Atom",
  xmlnsOpenSearch: "http://a9.com/~spec/opensearchrss/1.0/",
  xmlnsGsx: "http://schemas.google.com/spreadsheets/2006/extended",
  - id: {
    $t: "https://spreadsheets.google.com/feeds/list/1Xqe7slZ9dEOTCn1Yx13OE4xgvrXOL8Y_iu3WNlyGB1U/od6/public/values"
  },
  - updated: {
    $t: "2019-08-22T10:39:40.332Z"
  },
  - category: [
    - {
      scheme: "http://schemas.google.com/spreadsheets/2006",
      term: "http://schemas.google.com/spreadsheets/2006#list"
    }
  ],
  - title: {
    type: "text",
    $t: "Ark1"
  },
  - link: [
    - {
      rel: "alternate",
      type: "application/atom+xml",
      href: "https://docs.google.com/spreadsheets/d/1Xqe7slZ9dEOTCn1Yx13OE4xgvrXOL8Y_iu3WNlyGB1U/pubhtml"
    },
    - {
      rel: "http://schemas.google.com/g/2005#feed",
      type: "application/atom+xml",
      href: "https://spreadsheets.google.com/feeds/list/1Xqe7slZ9dEOTCn1Yx13OE4xgvrXOL8Y_iu3WNlyGB1U/od6/public/values"
    },
    - {
      rel: "http://schemas.google.com/g/2005#post",
      type: "application/atom+xml",
      href: "https://spreadsheets.google.com/feeds/list/1Xqe7slZ9dEOTCn1Yx13OE4xgvrXOL8Y_iu3WNlyGB1U/od6/public/values"
    },
    - {
      rel: "self",
      type: "application/atom+xml",
      href: "https://spreadsheets.google.com/feeds/list/1Xqe7slZ9dEOTCn1Yx13OE4xgvrXOL8Y_iu3WNlyGB1U/od6/public/values?alt=json"
    }
  ],
  - author: [
    - {
      - name: {
        $t: "klausmandal"
      },
      - email: {
        $t: "klausmandal@gmail.com"
      }
    }
  ],
  - openSearch$totalResults: {
    $t: "4"
  },
  - openSearch$startIndex: {
    $t: "1"
  },
  - entry: [
    - {
      - id: {
        $t: "https://spreadsheets.google.com/feeds/list/1Xqe7slZ9dEOTCn1Yx13OE4xgvrXOL8Y_iu3WNlyGB1U/od6/public/values"
      },
      - updated: {
        $t: "2019-08-22T10:39:40.332Z"
      },
      - category: [

```

# 2 Den nasty rå JSON fra Google sheet'et!

Nu er vi nede ved entry. Navnene på JSON-objekterne er sammensat af navnene på vores kolonner i sheetet og nogle bogstaver og \$ tegn som google bruger til at strukturere med. Men ellers er det som vores dyre JSON:

```
minJson.feed.entry.forEach((person) => {
```

```
    person.gsx$navn.$t;  
    person.gsx$github.$t;  
    person.gsx$billede.$t;  
    person.gsx$alder.$t;
```

```
}
```

```
- gsx$navn: {  
  $t: "Alan Engelhardt"  
},  
- gsx$github: {  
  $t: "alan-engelhardt"  
},  
- gsx$billede: {  
  $t: "https://avatars0.githubusercontent.com/u/5858672?s=400&v=4"  
},  
- gsx$alder: {  
  $t: "272"  
},  
- gsx$ken: {  
  $t: "m"  
},  
- gsx$email: {  
  $t: "ale@kea.dk"  
},  
- gsx$hold: {  
  $t: "b"  
},  
- gsx$historie: {  
  $t: "Til september går jeg på pension!! :)"  
},  
},  
- id: {  
  $t: "https://spreadsheets.google.com/feeds/list/1Xqe7slZ9dEOTCnYx130E4xqvrXOL8Y_iu3WNlyGB1U/od6/public/val"  
},  
- updated: {  
  $t: "2019-08-22T10:39:40.332Z"  
},  
- category: [  
  {  
    scheme: "http://schemas.google.com/spreadsheets/2006",  
    term: "http://schemas.google.com/spreadsheets/2006#list"  
  }  
],  
- title: {  
  type: "text",  
  $t: "Klaus Mandal"  
},  
- content: {  
  type: "text",  
  $t: "github: MondaleMondale, billede: http://mandalskeawebspace.dk/mitfoto/elvis.jpg, alder: 68, køn: m, em"  
},  
- link: [  
  {  
    rel: "self",  
    type: "application/atom+xml",  
    href: "https://spreadsheets.google.com/feeds/list/1Xqe7slZ9dEOTCnYx130E4xqvrXOL8Y_iu3WNlyGB1U/od6/public/val"  
  }  
],  
- gsx$navn: {  
  $t: "Klaus Mandal"  
},  
- gsx$github: {  
  $t: "MondaleMondale"  
},  
- gsx$billede: {  
  $t: "http://mandalskeawebspace.dk/mitfoto/elvis.jpg"  
},  
- gsx$alder: {  
  $t: "68"  
},  
- gsx$ken: {  
  $t: "m"
```

# Personøvelse

...

# Data til personliste-opgaven

Personlisteopgaven går ud på at hente data fra et regneark, og vi tager udgangspunkt i jeres egne oplysninger - udfyld en række på:

[https://docs.google.com/spreadsheets/d/1Xge7slZ9dEOTCn1Yxl3OE4xgvrXOL8Y\\_iu3WN1yGB1U/edit#gid=0](https://docs.google.com/spreadsheets/d/1Xge7slZ9dEOTCn1Yxl3OE4xgvrXOL8Y_iu3WN1yGB1U/edit#gid=0)

I skal udfylde: **navn**, **github**, **billede**, **alder**, **køn**, **email**, **hold**, **historie**

**github**: kun navnet på jeres konto - ikke en url-adresse til den

**billede**: Det skal være url'en på et billede, du har liggende af dig selv på nettet - fx. dit profilbillede på Facebook.

**email**: brug din KEA-mailadresse.

**alder**: et tal (ikke bogstaver, og skriv ikke "år" bagefter)

**køn**: m eller k (eller noget andet, som du selv formulerer, men ikke M, K eller F)

**hold**: a eller b (ikke A eller B)

**historie**: En linje om dig selv.

Sådan henter du adressen på et billede af dig selv, som ligger på nettet: højreklik på billedet og vælg **Copy Image Address**

# Personlisteopgave

Personlistens data ligger nu på et regneark, og vi kan hente data fra regnearket

Regnearket key-værdi er: **1Xge7slZ9dEOTCn1Yxl3OE4xgvrXOL8Y\_iu3WN1yGB1U**

Lav en webside, **personListe.html**, som for hver person viser:

- navn i en h2-overskrift
- billede
- et fungerende link til github-kontoen.

Opsæt den dynamiske personliste, med css grid, så personerne står i 1 kolonne på mobil, 2 kolonner på tablet, og tre kolonner på laptop/desktop.

Du skal aflevere et link til din løsning på github senest kl. 23.59 i aften.

# Intro til modul-opgave (Babushka)

## **Modulopgave:**

Ligger på Fronter

## **Data til casen:**

<https://docs.google.com/spreadsheets/d/17Dd7DvkPaFamNUdUKlrFgnH6POvBJXac7qyiS6zNRw0/edit?usp=sharing>

(link på Fronter)

## **Billeder:**

[zip-fil på Fronter](#) - i regnearket ligger billedfilernes navne.