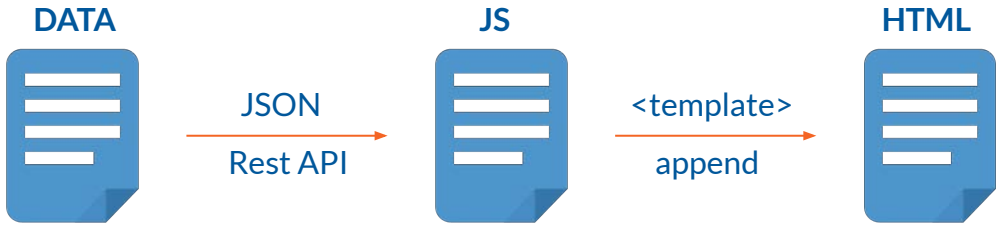
JSON

Introduktion til JSON og fetch

Helt overordnet struktur



1. Data hentes udefra i JSON format via Rest API
2. Javascript styrer kommunikationen
3. HTML modtager data som kopier af en html-skabelon

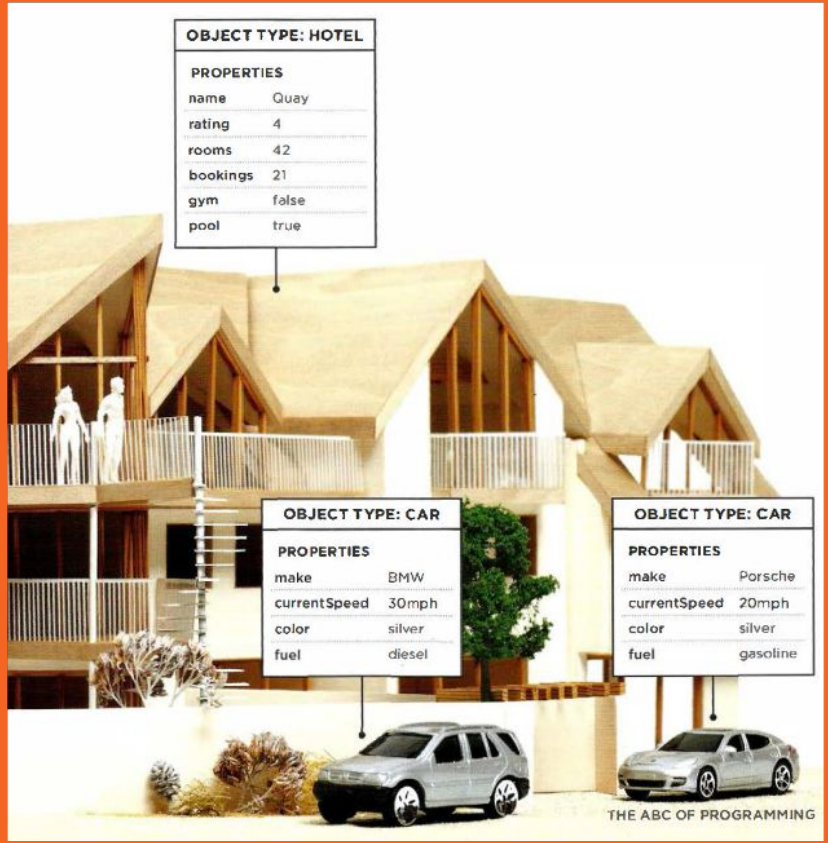
JSON

- JSON er en **syntaks** til **lagring** og **udveksling af data**.
- JSON er **tekst** skrevet med **J**ava**S**cript **O**bject **N**otation.
- Når man udveksler data mellem en browser og en server, kan data kun være som **tekst**.
- Med JSON kan vi sende og modtage ren tekst fra en server og bruge den som et JavaScript-objekt.
- Vi kan arbejde med data som JavaScript-objekter uden kompliceret parsing og oversættelser.
- Kode til læsning og generering af JSON findes i mange programmeringssprog.

https://www.w3schools.com/js/js_json_intro.asp

Hvad er det nu et objekt er...?

- En repræsentation af noget,
  - ofte et objekt i den "rigtige" verden som en person, et produkt, et køretøj el.lign. med en række fælles egenskaber.
- En abstraktion.
- En logisk gruppering.
- En datastruktur.



Hvorfor JSON?

Når data gemmes i særskilte datafiler, opnår vi at adskille html-kode mv. fra data:

- Struktur i html
- Layout i css
- Handling i JavaScript
- **Data** i JSON

Men først og fremmest er det smart til at udveksle data over nettet

JSON eksempel

JSON **datatyper**

- Strenge
- Tal
- Objekter
- Arrays
- Booleans
- Null

• biler.json (T7)

```
[
    {
        "mærke": "Volvo",
        "model": "Amazon",
        "motor": "Benzin",
        "km": 500000
            },
    {
        "mærke": "VW",
        "model": "Polo",
        "motor": "Diesel",
        "km": 40500
            },
    {
        "mærke": "Tesla",
        "model": "S",
        "motor": "El",
        "km": 1000
            }
]
```

JSON Syntaks { “ ” : ” ” }

- JSON-syntaks stammer som sagt fra JavaScript Objekt Notation:
  - Data er organiseret i nøgle / værdi-par {“agent” : “007”}
  - Data adskilles med kommaer
  - Krøllede parenteser { } omkranser objekter
  - Firkantede parenteser [ ] indeholder arrays
- Et nøgle/værdi-par består af et feltnavn i dobbelt citationstegn, efterfulgt af et kolon, efterfulgt af en værdi: **{”navn” : ”Klaus”}**
- I JSON skal nøgler (feltnavne) være en streng i dobbelt citationstegn!
- I JavaScript *kan* nøgler være *uden* citationstegn: {navn: ”Martin”}

Næsten lig med JavaScript objekt syntaks - men ikke helt.

Objekter i arrays, arryas i objekter...

NB! Sæt IKKE komma efter sidste egenskab og efter sidste objekt!

undervisere.json (T7) -

```
[
    {
        "fornavn": "Martin",
        "efternavn": "Bregnhøi",
        "mail": "mabe@kea.dk",
        "emner": ["JavaScript", "CSS", "Projektstyring"]
        },
    {
        "fornavn": "Klaus",
        "efternavn": "Mandal Hansen",
        "mail": "klmh@kea.dk",
        "emner": ["JavaScript", "HTML", "Tøjmode"]
        },
    {
        "fornavn": "Louise Ea",
        "efternavn": "Holbek",
        "mail": "loeh@kea.dk",
        "emner": ["SoMe", "SEO", "BMC"]
        }
]
```

Adgang til værdierne i JSON

- Objekters egenskaber nås ved hjælp af punktum .
- Arrays tilgås ved hjælp af firkantede parenteser [ ]

undervisere.json (T7)

```
[
    {
        "fornavn": "Martin",
        "efternavn": "Bregnhøi",
        "mail": "mabe@kea.dk",
        "emner": ["JavaScript", "CSS", "Projektstyring"]
            },
    {
        "fornavn": "Klaus",
        "efternavn": "Mandal Hansen",
        "mail": "klmh@kea.dk",
        "emner": ["JavaScript", "HTML", "Tøjmode"]
            },
    {
        "fornavn": "Louise Ea",
        "efternavn": "Holbek",
        "mail": "loeh@kea.dk",
        "emner": ["SoMe", "SEO", "BMC"]
            }
]
```

```
const person = {
    "navn": "Martin",
    "titel": "Lektor"
}
```

Præcis på samme måde som med JavaScript objekter

```
let navn = person.navn;
let fornavn = undervisere[2].fornavn;
let emne = undervisere[0].emner[2];
```

Hente og vise JSON data

Med fetch()

For at hente data fra en JSON ind i vores HTML, skal vi bruge en *asynkron* metode ved navn “fetch”

Hvad er en asynkron metode?

Også kaldet AJAX
(Asynkron JavaScript og XML)

Med **AJAX** kan du:

- Opdatere en webside uden at indlæse siden igen.
- Anmode om data fra en server, efter siden er indlæst.
- Modtage data fra en server, efter siden er indlæst.
- Sende data til en server i baggrunden.

---

https://www.w3schools.com/xml/ajax_intro.asp

fetch() er en asynkron metode

- Fetch er et API til hentning af ressourcer på tværs af netværket.
- Fetch(arg) kræver et obligatorisk argument: **stien** til den **ressource**, du vil hente, f.eks. en **json** fil.
- Fetch returnerer et *løfte* (**promise**).
- Et **promise** er et objekt, der repræsenterer den eventuelle færdiggørelse eller fiasko af en asynkron operation.
- **Promise tillader to eller flere asynkrone handlinger at køre parallelt, hvor hver efterfølgende operation starter, når den forrige operation lykkes, med resultatet fra det forrige trin.**
- Udføres ved hjælp af en "promise chain".

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Using_promises

Hente JSON med fetch()

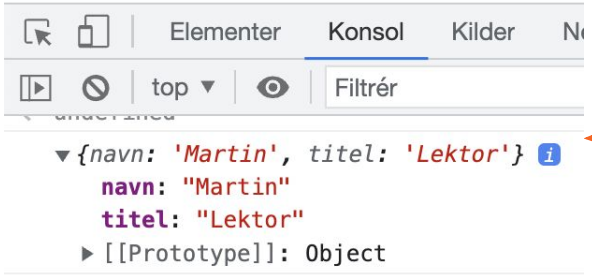

```
const fil = "person.json";

function hentData() {
  fetch(fil).then((resp) => resp.json()).then(vis);
}

function vis(data) {
  console.log(data);
}

hentData();
```

promise chain



To ligeværdige fetch syntakser

```
function getData() {
  fetch(url)
  .then((res) => res.json())
  .then(visProdukt);
}
```

```
async function getData() {
  const resp = await fetch(url);
  const data = await resp.json();
  visProdukt(data);
}
```

JSON array

Brug <template> og forEach

Som vi har set giver det ofte mening at gemme flere objekter i et array

For at få fat i disse, skal vi loop'e igennem array'et med forEach

Og for at vise dem i DOM'en skal vi klone <template> og tilføje data

(Præcis som vi gjorde med JS objekter lokalt tidligere)

Fra JSON til DOM

```html
<body>
  <h1>Undervisere</h1>
   <main></main>

   <template>
       <article>
           <h2 class="fornavn">NAVN</h2>
           <h3 class="efternavn">EFTERNAVN</h3>
           <p class="mail">MAIL</p>
       </article>
   </template>

    <script>
       "use strict";

       const fil = "undervisere.json";

       async function hentdata(fil) {
           const resultat = await fetch(fil);
           const json = await resultat.json();
           vis(json);
       }

       function vis(json) {
           const beholder = document.querySelector("main");
           const skabelon = document.querySelector("template");
           json.forEach(underviser => {
               const klon = skabelon.cloneNode(true).content;
               klon.querySelector(".fornavn").textContent=underviser.fornavn;
               klon.querySelector(".efternavn").textContent=underviser.efternavn;
               klon.querySelector(".mail").textContent=underviser.mail;
               beholder.appendChild(klon);
           });
       }

       hentdata(fil);
   </script>
</body>
```

Alternativ syntax

undervisere.json (T7) — Brackets

```json
[
    {
        "fornavn": "Martin",
        "efternavn": "Bregnhøi",
        "mail": "mabe@kea.dk",
        "emner": ["JavaScript", "CSS", "Projektstyring"]
            },
    {
        "fornavn": "Klaus",
        "efternavn": "Mandal Hansen",
        "mail": "klmh@kea.dk",
        "emner": ["JavaScript", "HTML", "Tøjmode"]
            },
    {
        "fornavn": "Louise Ea",
        "efternavn": "Holbek",
        "mail": "loeh@kea.dk",
        "emner": ["SoMe", "SEO", "BMC"]
            }
]
```

Undervisere

Martin
Bregnhøi
mabe@kea.dk

Klaus
Mandal Hansen
klmh@kea.dk

Louise Ea
Holbek
loeh@kea.dk

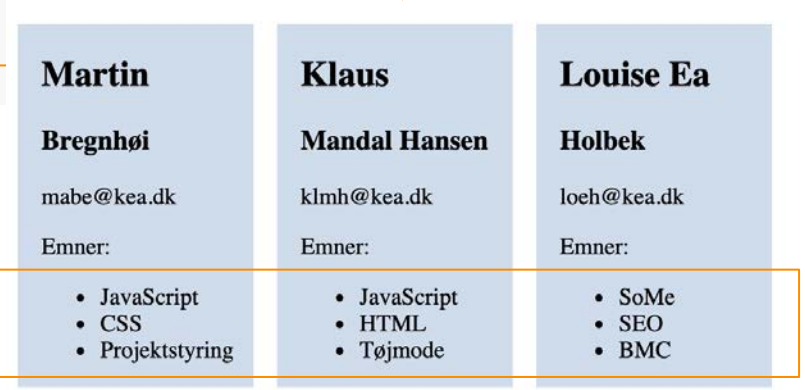
Arryas i objekter i arrays = loop i loop

```
function vis(json) {
    const beholder = document.querySelector("main");
    const skabelon = document.querySelector("template");
    json.forEach(underviser => {
        const klon = skabelon.cloneNode(true).content;
        klon.querySelector(".fornavn").textContent = underviser.fornavn;
        klon.querySelector(".efternavn").textContent = underviser.efternavn;
        klon.querySelector(".mail").textContent = underviser.mail;

        underviser.emner.forEach(emne => {
            klon.querySelector(".emneliste").innerHTML += "<li>" + emne + "</li>"
        })

        beholder.appendChild(klon);
    });
}
```

```
<template>
    <article>
        <h2 class="fornavn">NAVN</h2>
        <h3 class="efternavn">EFTERNAVN</h3>
        <p class="mail">MAIL</p>
        <p>Emner:</p>
        <ul class="emneliste"></ul>
    </article>
</template>
```



Øvelse: json array

1. Lav tre nye filer: en json fil, en js fil og en html fil (med et link til js-filen)
2. Lav et array med tre objekter i json-filen, som hver indeholder et array, f.eks. musik-albums el.lign. (eller genbrug dit array med superhelte)
3. Fetch json-filen via script-filen og vis alle data i DOM'en (html-filen)

Tip: Se de foregående to slides til inspiration

```
[
    {
        "mærke": "Volvo",
        "model": "Amazon",
        "motor": "Benzin",
        "km": 500000,
        "udstyr": ["blinklys", "læderrat"]
            },
    {
        "mærke": "VW",
        "model": "Polo",
        "motor": "Diesel",
        "km": 40500,
        "udstyr": ["sædevarme", "gps"]
            },
    {
        "mærke": "Tesla",
        "model": "S",
        "motor": "El",
        "km": 1000,
        "udstyr": ["gps", "aircon", "soltag"]
            }
```

JSON udefra Rest API

Jonas' API

Data kan komme udefra

- I har nu set, at json kan hentes ind i vores script fra en anden fil
- Json kan på samme måde hentes fra en helt anden lokation på internettet
- Dette gøres via et **Rest API**

Rest API

- **API**: Application Program Interface
- Specificerer hvordan software-komponenter (programmer) skal interagere.
- **REST** api: **RE**presentational **S**tate **T**ransfer api (se evt. video)
- Bruges især til web-applikationer.
- Rest API'er bruger http-requests (dvs. url-adresser) til kommunikationen mellem software-komponenterne.

https://searchmicroservices.techtarget.com/definition/RESTful-API

Jonas' API

- Vi bruger Jonas' API til vores øvelses-site
- Lad os kigge på det: https://kea-alt-del.dk/t7/api/
- Kopier nogle af de tilgængelige endpoints til din browsers (Firefox) adressefelt fx https://kea-alt-del.dk/t7/api/products
- Kig på hvad der kommer retur…

Forskellige endpoints

Visning af data

For at få vist de data vi får retur fra Jonas' API til brugerne, skal vi igen kombinere de teknikker vi har gennemgået:

- **Fetch** til at hente data
- **Manipulere** DOM (ændre indhold til det vi modtager)
- **forEach** til at loop'e igennem json array
- **HTML <template>** til at placere vores data i DOM'en

  (de sidste to kun når der er flere produkter)

Hent data fra API og vis i DOM (enkelt produkt)

```
const id = 1651;
const url = `https://kea-alt-del.dk/t7/api/products/${id}`;

function getProduct() {
  fetch(url)
    .then((res) => res.json())
    .then(visProdukt);
}

function visProdukt(produkt) {
  document.querySelector(".purchaseBox h3").textContent = produkt.productdisplayname;
  document.querySelector("img").src = `https://kea-alt-del.dk/t7/images/webp/640/${id}.webp`;
  document.querySelector("img").alt = produkt.productdisplayname;
  // etc. med de øvrige data
}

getProduct();
```

Hent data fra API og vis i DOM (liste visning)

```
const url = "https://kea-alt-del.dk/t7/api/products?start=100"

fetch(url).then((response) => response.json()).then(visData);

function visData(data) {
    console.log(data);
    const beholder = document.querySelector("main");
    const temp = document.querySelector("template").content;
    data.forEach(element => {
        const klon = temp.cloneNode(true);
        klon.querySelector("h1").textContent = element.brandname;
        klon.querySelector("p").textContent = element.productdisplayname;
        klon.querySelector("img").src = `https://kea-alt-del.dk/t7/images/webp/640/${element.id}.webp`;
        beholder.appendChild(klon);
    });
}
```

```
<main></main>

<template>
    <article>
        <h1></h1>
        <p></p>
        <img src="" alt="">
    </article>
</template>
<script>
```