



SINGLE VIEW

på separat side

...

Loop-view kontra detail-view (single-view)

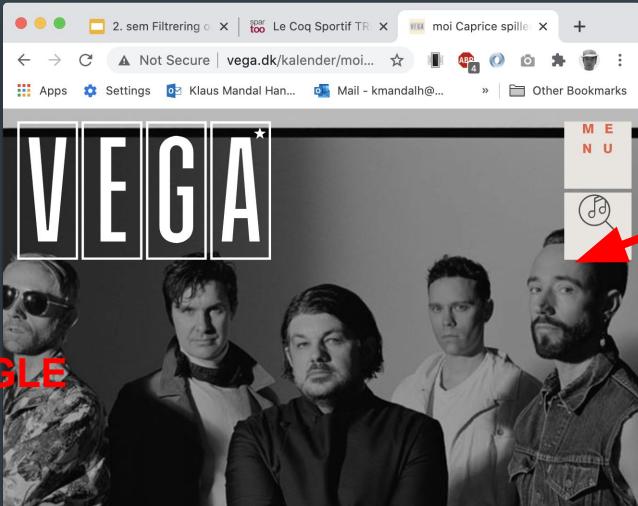
Loop-view er en oversigtsvisning af mange objekter
(en liste)

Single-view er en detaljeret visning af ét objekt

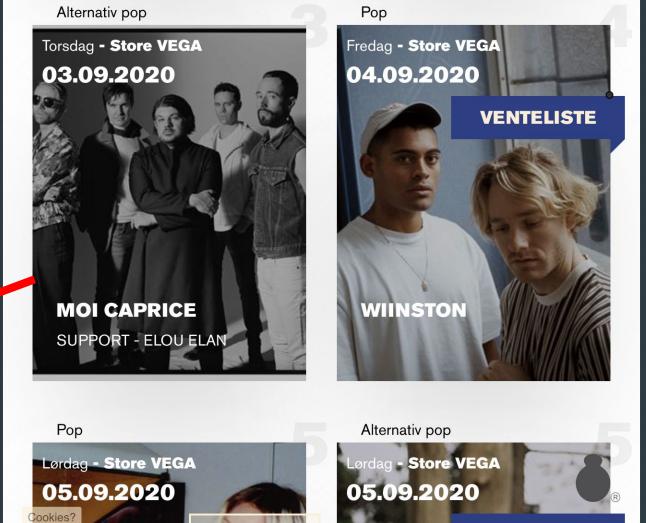
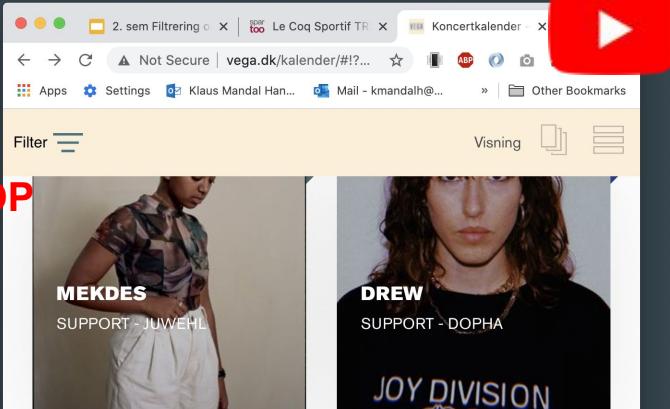
I går arbejdede vi
med loop-views

I dag arbejder vi med
single-view

SINGLE



LOOP





1. Man kan lave detail-view på to måder. Som en popup der lægger sig oven på den side man er på:

Fordele:

Giver en god brugeroplevelse, fordi der ikke skiftes side

Kan styles meget elegant med fx lightbox-effekter

Ulemper:

Single-viewet har ikke sin egen url-adresse

kan ikke deles

kan ikke findes af søgemaskiner



2. Man kan lave detail-view på to måder. Som en separat side:

Fordele:

Single-viewet får sin egen url-adresse

Siden kan nemmere deles

Siden kan findes af Google

Ulemper:

Der skiftes side, når der klikkes på en person - en underside skal altså laves



Strategi for detail-view på separat side

1. Når der klikkes på et element i liste-viewet, skal browseren hoppe over til en ny side, detalje.html
En variabel skal medbringes der indeholder ID på det element (person) der er klikket på.
2. I detalje.html: hent url-variablen med ID'et ind i scriptet
3. Kald restdb med ID'et og fyld siden op med det content der kommer tilbage fra restdb.
4. Der skal være en tilbage-knap på detail-view siden der tager os tilbage til loop-view siden



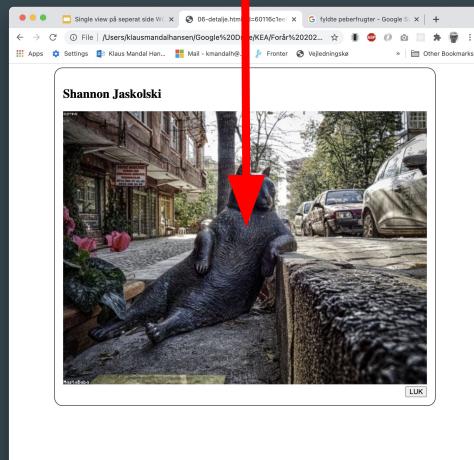
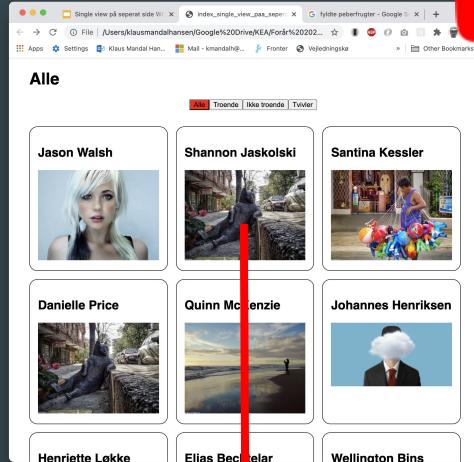
Kald med url-variabel

I loop-view skal der være eventlisteners på alle elementer.
Eventhandleren skal gå til en ny side, detail.html

I javascript kan man hoppe til en ny adresse med
location.href

Medbring en variabel, som kan fortælle hvilket element,
der blev klikket på, altså elementets ID.

Variablen kan sendes som **url-variabel**





window.location

I javascript har vi et indbygget objekt, `window.location`
`window.location` har en lang række egenskaber

En af dem er `href`:

```
console.log(window.location.href)
```

vil vise den aktuelle sides url i inspectoren

Mens:

```
window.location.href = "http://kea.dk"
```

får browseren til at springe over til en helt anden side: keas webside.

“window” kan udelades, så vi skriver: `location.href = "http://kea.dk";`



url-variabler

Når en side kaldes med en url-adresse, kan man sende variabler med til siden.

Sætter jeg denne url-adresse ind i browserens adressefelt:

<https://mandalskeaweb.space.dk/tema7urlparametertest/hej.html?navn=Alan>

vil browseren hente siden <https://mandalskeaweb.space.dk/tema7urlparametertest/hej.html>, og den vil desuden give siden en variabel, i dette tilfælde *navn*, med værdien “Alan”.

Først kommer url'en så kommer der et spørgsmåltegn, efter spørgsmåltegnet kommer url-parametrene, der kan sagtens være flere, så skal de adskilles af &.

Vi kommer senere til, hvad modtagersiden gør, for at få fat i variablen. (Hvis du er nysgerrig så kik i sidens kode: højreklik: view source)



Hvad skal vi gemme i vores url-variabel?

Det vi skal gemme er id'et der er tilknyttet den post vi har klikket på. Alle posts i restdb har en id. Den kan vi fx se hvis vi logger vores JSON eller kigger på JSON' en direkte i browseren. Fordelen ved et ID er at det er unikt ligesom fx vores CPR-nummer. Så der er ikke nogen anden post med dette id.

```
personer = await JSONData.json();
console.log("Personer", personer);
```

```
Personer
▼ (13) [{}]
  ▼ 0:
    ► billede: ["601b9d7fdc06ba22000364ca"]
      efternavn: "Walsh"
      fornavn: "Jason"
      fødselsdag: "2011-06-07T00:00:00.000Z"
      hobby: "cumque atque aperiam ad quos tempore odio"
      titel: "Et adipisci"
      troende: "nej"
      _id: "60116c1eef2e8a1b00036d3a"
      _mock: true
    ► __proto__: Object
  ▶ 1: {_id: "60116c1eef2e8a1b00036d39", efternavn: "Jaskolski", hobby: "est n
```



```
[{"_id": "60116c1eef2e8a1b00036d3a",
  "eternavn": "Walsh",
  "hobby": "cumque atque aperiam ad quos tempore odio",
  "titel": "Et adipisci",
  "fornavn": "Jason",
  "troende": "nej"}]
```



I vores JavaScript tilføjer vi en eventListener og en eventHandler

```
//funktion der viser personer i liste view
function visPersoner() {
  const dest = document.querySelector("#liste"); // container til listen
  const skabelon = document.querySelector("template").content;
  dest.textContent = ""; // ryd container inden ny loop
  personer.forEach(person => {
    console.log("Troende", person.troende);
    // loop igennem json (personer)
    // tjek hvilken tro personen har og sammenlign med aktuelle
    // værdien "alle"
    if (person.troende == filter || filter == "alle") {
      const klon = skabelon.cloneNode(true);
      klon.querySelector(".navn").textContent = person.fornavn + " " + person.efternavn;
      klon.querySelector(".profil-billede").src = medieurl + person.billeder;
      klon.querySelector(".person").addEventListener("click", () => visDetaljer(person));
      dest.appendChild(klon);
    }
  })
}

function visDetaljer(hvem) {
  location.href = `02-detalje.html?id=${hvem._id}`;
}
```

Event listener der gør at når man klikker på person-articlen aktiverer den en anonym funktion der kalder visDetaljer funktionen med `person` som parameter. Grunden til at vi bruger en anonym funktion istedet for at skrive navnet på eventHandleren som vi plejer, er at vi skal have sendt et parameter med.

`() => visDetaljer(person)` er det samme som at skrive: `function(){visDetaljer(person)}`

Her navigerer vi med `location.href` hen til en side vi ikke har lavet endnu.

Vi sender urlParametret `id` med, og det får værdien af `hvem._id`.

Bliver fx til: `location.href = "02-detalje.html?id=60116c1eef2e8a1b00036d3a";`



Øvelse 1

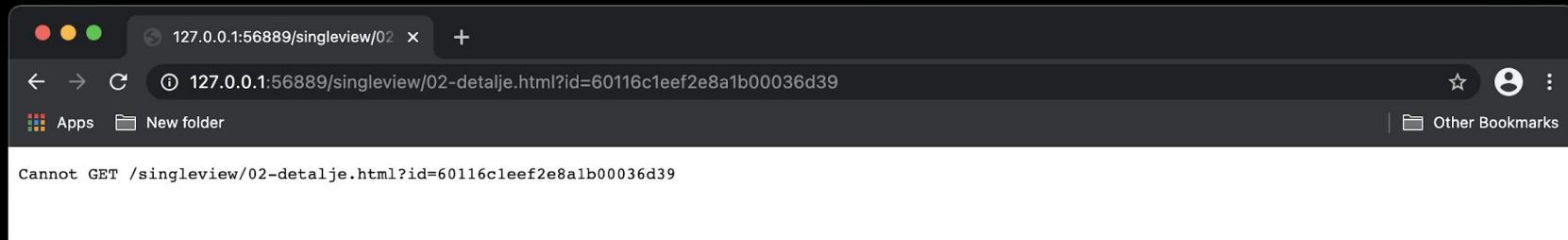
Opret en ny mappe, *single*, i tema7-mappen.

Gem en kopi af din slutfil fra i går i single-mappen under navnet 01-kald..html.

Læg en eventlistener på alle personer, som får browseren til at hoppe til 02-detalje.html medbringende personens id som et url-parameter.

Test, at det virker: url'en i 02-detalje.html skal have en variabel med.

Vi laver 02-detalje.html om lidt, men test at browseren **prøver** OG medbringer den rigtige url-variabel.

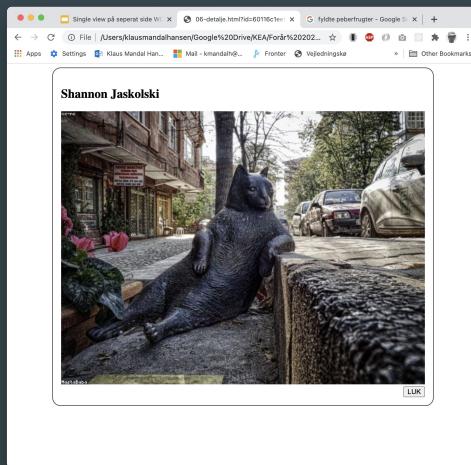




Detalje siden

Vi skal lave en side der samler URL-parametret op, og viser et detaljeview

Nu er der ikke noget loop, der skal kun vises en person ad gangen, single view





Øvelse 2

Lav en ny html-side der hedder 02-detalje.html

Lav et html-skelet

Tag indholdet af den HTML-template du bruger i 01-kald.html og sæt den ind i dit body tag i 02-detalje.html. Det er et godt udgangspunkt for vores detalje view. Husk det er **indholdet** af templetten, så template-tags skal ikke med. Vi får ikke brug for at klone den, vi bruger den som den er.

Den skal selvfølgelig styles som du synes den skal se ud. Hvis du styler den nu, skal du skrive noget tekst ind i HTML'en ellers kan du ikke se hvad du styler. Du kan også vente til at der kommer noget content ind.



Få fat i URL-variablen på detalje siden

Så nu skal vi hent url-parametret, vi har sendt fra vores 01-kald.html, ind i scriptet i vores 02-detalje.html.

Øverst i scriptet kan vi hente alle URL-parametrene ud (vi har kun et), og ved hjælp af `get("id")` hente vores enlige parameter ud, der jo hedder *id*. Det sætter vi så lig med en const vi kalder *id*, og på den måde har vi fået overført parameteret fra url-strengen og over i en variabel vi kan bruge i vores script

```
<script>
  const urlParams = new URLSearchParams(window.location.search);
  const id = urlParams.get("id");
```

DISCLAIMER

Hvis der ikke er noget url-parameter i adresselinien på browseren, er der selvfølgelig heller ikke noget id at hente! Så start altid dit arbejde på 02-detalje.html med at kalde den fra 01-kald.html. Du kan selvfølgelig også skrive URL-parametrene ind manuelt hvis du kan huske id'et du vil bruge 😊



Find personen ud fra id'et og fyld indholdet ind i HTML'en

Nu har vi vores id og det kan vi så bruge til at få fat i kun den person fra restdb. Restdb har lavet et *cheat-sheet* og hvis man kigger i det kan man se at man kan få fat i en bestemt post på denne her måde:

GET

/rest/{collection}/ID

Get one document from a collection. ID must be a valid ObjectId.



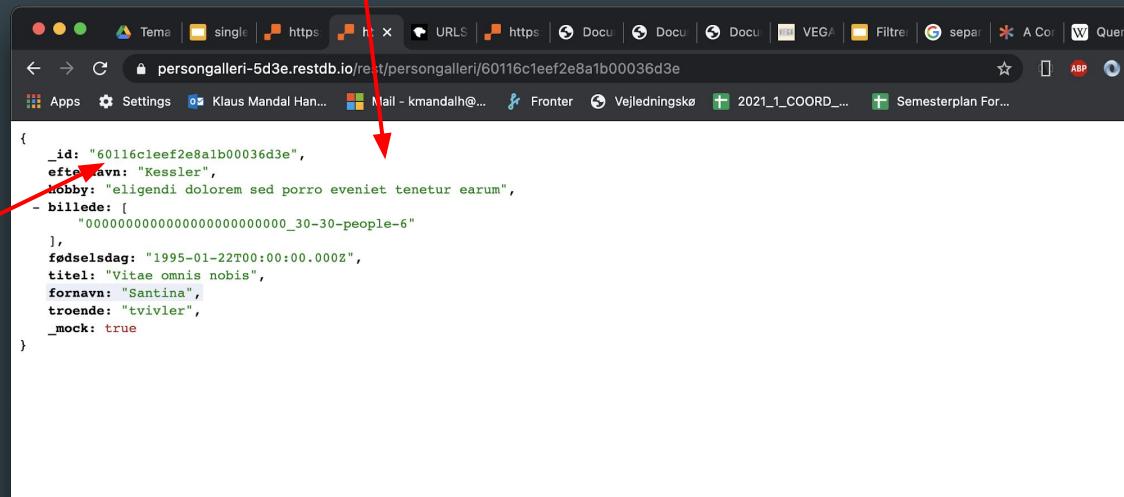
Kalde med id

Hvis jeg kalder min gode gamle <https://persongalleri-5d3e.restdb.io/rest/persongalleri> får jeg jo alle posts i databasen, men hvis jeg gør som der står i cheat-sheetet får jeg kun en bestemt post, hvis jeg altså har et gyldigt id på den post:

<https://persongalleri-5d3e.restdb.io/rest/persongalleri/60116c1eef2e8a1b00036d39>

Jeg kan ikke linke direkte så I må kikke herunder. I kan kikke på det når I opretter jeres egen restdb.

```
- {
  "_id": "60116c1eef2e8a1b00036d3a",
  "efternavn: "Walsh",
  hobby: "cumque atque aperiam ad quos tempore odio",
  - billede: [
    "601b9d7fdc06ba22000364ca"
  ],
  fødselsdag: "2011-06-07T00:00:00.000Z",
  titel: "Et adipisci",
  fornavn: "Jason",
  troende: "nej",
  _mock: true
},
- {
  "_id": "60116c1eef2e8a1b00036d39",
  "efternavn: "Jaskolski",
  hobby: "est non qui omnis provident sed aspernatur",
  - billede: [
    "00000000000000000000000000000000_30-30-people-8"
  ],
  fødselsdag: "2006-08-17T00:00:00.000Z",
  titel: "Inventore sunt",
  fornavn: "Shannon",
  troende: "ja",
  _mock: true
},
- {
  "_id": "60116c1eef2e8a1b00036d3e",
  "efternavn: "Kessler",
  hobby: "eligendi dolorem sed porro eveniet tenetur earum",
  - billede: [
    "00000000000000000000000000000000_30-30-people-6"
  ],
  fødselsdag: "1995-01-22T00:00:00.000Z",
  titel: "Vitae omnis nobis",
  fornavn: "Santina",
  troende: "tvivler",
  _mock: true
}
```



The screenshot shows a browser window with the URL <https://persongalleri-5d3e.restdb.io/rest/persongalleri/60116c1eef2e8a1b00036d3e>. The page displays the following JSON data:

```
{
  "_id": "60116c1eef2e8a1b00036d3e",
  "efternavn: "Kessler",
  hobby: "eligendi dolorem sed porro eveniet tenetur earum",
  - billede: [
    "00000000000000000000000000000000_30-30-people-6"
  ],
  fødselsdag: "1995-01-22T00:00:00.000Z",
  titel: "Vitae omnis nobis",
  fornavn: "Santina",
  troende: "tvivler",
  _mock: true
}
```

02-detalje



```
<script>  
const urlParams = new URLSearchParams(window.location.search);  
const id = urlParams.get("id");
```

```
const medieurl = "https://persongalleri-5d3e.restdb.io/media/";  
let person;  
const myHeaders = {  
    "x-apikey": "600fe9211346a1524ff12e31"  
}  
console.log("ID", id);  
document.addEventListener("DOMContentLoaded", loadJSON);
```

```
async function loadJSON() {  
    const JSONData = await fetch(`https://persongalleri-5d3e.restdb.io/rest/persongalleri/${id}`, {  
        headers: myHeaders  
    });  
    person = await JSONData.json();
```

```
    console.log("Personer", person);  
    visPerson(person);  
}
```

```
function visPerson() {  
    document.querySelector(".navn").textContent = person.fornavn + " " + person.efternavn;  
    document.querySelector(".profil-billede").src = medieurl + person.billed;
```

Nu kan vi så sætte id'et vi har hentet fra url-parametrene ind i vores kald til restdb og derved modtage en og kun en post. *fetch*-delen ligner scriptet i vores 01-kald.html ...

.... men da vi kun får en post er det jo bare at hente den og overføre det content til vores html. Ikke noget med at loope.

```
</script>
```



Eventlistener og evnhandler til Tilbage-knap

Men vi skal jo også kunne komme **tilbage** til 01-kald.html fra 02-detalje.html.

Til det formål skal der bruges en knap **<button></button>** der skal skrues ind i 02-detalje.html'en et passende sted.

Og så skal vi bruge den knap til at komme hjem igen. Så der skal en eventlistener på og en eventhandler. Og i event handleren bruger vi history web API'et til at komme et hak baglæns i browserhistorien og dermed komme tilbage til 01-kald.html. *The History interface allows manipulation of the browser session history, that is the pages visited in the tab or frame that the current page is loaded in*

```
function visPerson() {
    document.querySelector(".navn").textContent = person.fornavn + " " + person.efternavn;
    document.querySelector(".profil-billede").src = medieurl + person.billede;
    document.querySelector("button").addEventListener("click", tilbageTilPersonGalleri);
}

function tilbageTilPersonGalleri() {
    history.back();
}
```



Øvelse 3

Arbejd videre med 02-detalje.html

Du skal:

- hente url-variablen ved hjælp af URLSearchParams.
- Lav en fetch der henter den post ind der svarer til id'et i URL-variablen
- Overfør indholdet fra post'en til html'en
- Lave en tilbage-knap, som kan springe tilbage til 01-kald.html
- Når det tekniske kører så stejl det så det ser mamselækkert ud