

**javascript-objekter og  
array af objekter**

**...**

# Program

1. Ny datatype i javascript: **Object**
2. Vis object i DOM-elementer
3. Mange objects i array
4. Liste i DOM'en over mange objekter

# Objekter

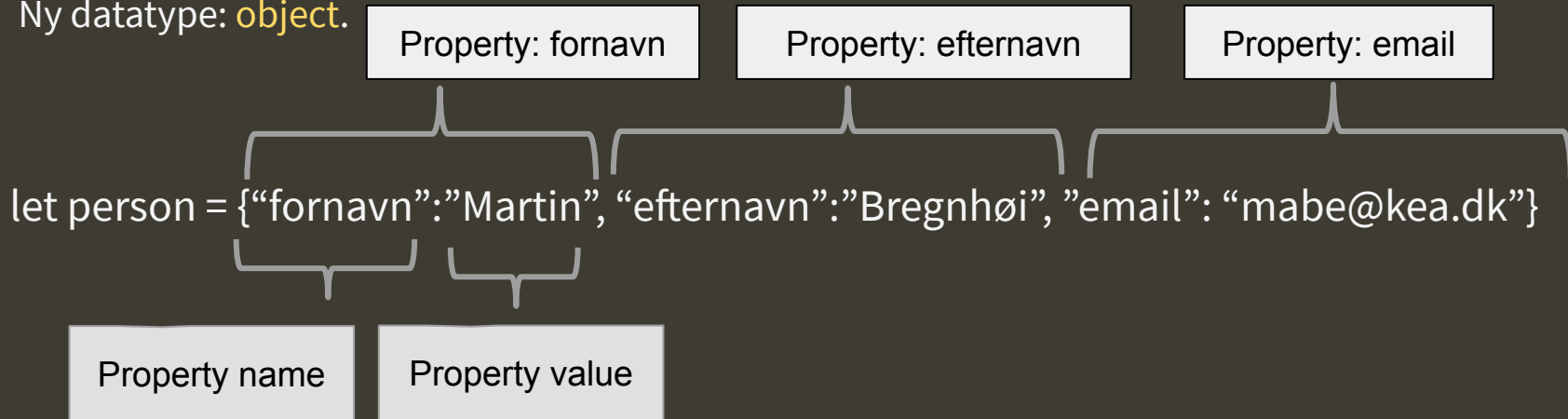
...

# Object: ny data-type, I skal lære

**Datatypes:** hvilken slags data kan vi tildele en variabel i js?

I kender: **number** (tal), **string** (tekst), **boolean** (sand eller falsk) og **array** (liste)

Ny datatype: **object**.



# Syntax-regler

Omkring property names og values:

"..."

```
let person = {"fornavn" : "Martin", "efternavn" : "Bregnhøj", "email" : "mabe@kea.dk"}
```

Start med {

mellem name og  
value: :

mellem properties: ,

Afslut med }

# Object-properties og dot-notation

```
let person = { "fornavn" : "Martin", "efternavn" : "Bregnhøj", "email" : "mabe@kea.dk" }
```

Property: fornavn

Property: efternavn

Property: email

person har tre properties: fornavn, efternavn og email

I js kan man få fat i property-values vha. **dot-notation**

```
let navn = `${person.fornavn} ${person.efternavn}`; //Martin Bregnhøj  
let navn = person.fornavn + " " + person.efternavn;
```

Dot-notation: .  
(punktum)

Dot-notation: .  
(punktum)

# Vis et objects værdier i DOM-element

I DOM'en har vi fx. et tomt element:

```
<article id="person">
  <h3></h3>
  <p></p>
  <img src="" alt="">
</article>
```

I js har vi erklæret et object, person:

```
const person = {
  "fornavn": "Martin",
  "efternavn": "Bregnhøi",
  "email": "mabe@kea.dk",
  "billede": "http://mabe-kea.dk/martin.jpg"
}
```

For at få personens data ind i html-elementet:

```
const destination = document.querySelector("#person");
const fuldeNavn = `${person.fornavn} ${person.efternavn}`;
destination.querySelector("h3").textContent = fuldeNavn;
destination.querySelector("p").textContent = person.email;
destination.querySelector("img").src = person.billede;
```

Martin Bregnhøi

mabe@kea.dk



# Øvelse 1: Opret et dyreobjekt

I mappen med øvelser, laver du en ny undermappe, **js-objekter**.

Åbn den i Brackets og lav en ny html-fil - kald den **01-objekt.html**

1. Erklær en variabel, **dyr**, som et **object**
2. **dyr** skal have fire properties: **navn**, **type**, **billede** og **levested** med værdierne: **gris**, **pattedyr**, "<http://mabe-kea.dk/E19/pics/pig.png>" og **svinefarme**
3. Udskriv objektet med `console.log` - se efter i console, hvordan det ser ud - prøv at folde det ud, så du kan se detaljerne
4. Udskriv dyrets navn, type og levested i console-vinduet på denne form:  
"**Dyr: gris, type: pattedyr, levested: svinefarme**".





# Øvelse 2: Vis dyret i DOM

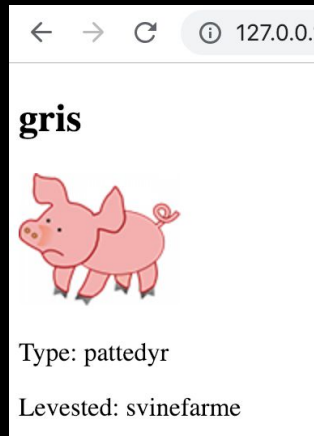
- Gem **01-objekt.html** i en ny kopi, **02-visDyr.html**.
- Lav et tomt article-tag i DOM'en, som du kan bruge til at vise informationerne om dyret.
- I scriptet skal du starte med at lave en eventlistener på, om DOM-indholdet er loaded (article-tagget skal være læst ind).
- Dét, du i **01-objekt.html** udskrev i console.log, skal du nu udskrive i article-tagget.
- Vis også billedet i article-tagget. Placer det under teksten.



# Øvelse 3: Udskrivning med billede

- Gem **02-visDyr.html** i en ny kopi, **03-vislelement.html**.
- Udbyg det tomme article tag, så det indeholder:
  1. En h3-overskrift til dyrets navn
  2. Et tomt img-tag til billedet
  3. Et p-tag til typen
  4. Et p-tag til levestedet
- Lav om på scriptet, så det indsætter de forskellige værdier i de rigtige tags
- commit og push til Github

```
<article id="dyr">  
  <h3></h3>  
  <img src="" alt="">  
  <p>Type: </p>  
  <p>Levested: </p>  
</article>
```



# Opsamling

javascript object

syntax for object

objekt-egenskab/property

dot-notation

# Objekter i array

...

# Mange ensartede objects

Tit arbejder man med mange ensartede objekter i javascript

De tilhører samme type eller **klasse** og har samme properties.

```
let person1={"fornavn":"Helle", "efternavn": "Frederiksen", "email": "helf@kea.dk" };  
let person2={"fornavn":"Martin", "efternavn": "Bregnhøj", "email": "mabe@kea.dk" };  
let person3={"fornavn":"Kamilla", "efternavn": "Victor", "email": "kvi@kea.dk" };
```

men... hvad gør man, hvis vi har 1000 personer???

# Objects i array

Start med [

```
let personer = [  
  {"fornavn": "Helle", "efternavn": "Frederiksen", "email": "helf@kea.dk"},  
  {"fornavn": "Martin", "efternavn": "Bregnhøi", "email": "mabe@kea.dk"},  
  {"fornavn": "Kamilla", "efternavn": "Victor", "email": "kvi@kea.dk"}  
];
```

, (komma) mellem  
objects - **IKKE** efter  
sidste object!

Afslut med ]

Hvis jeg nu skal have mit eget efternavn vist i console skriver jeg:

```
console.log(personer[1].efternavn);
```

eller Helles email:

```
console.log(personer[0].email);
```

OBS! første  
element i et array  
har altid index 0

# forEach loop til listevisning

En tom tag, som alle elementer skal sættes ind i

```
26 <section id="liste"></section>
27 <script>
28   document.addEventListener("DOMContentLoaded", start);
29   let personer = [
30     {"fornavn": "Helle", "efternavn": "Frederiksen", "email": "helf@kea.dk"},
31     {"fornavn": "Martin", "efternavn": "Bregnhøj", "email": "mabe@kea.dk"},
32     {"fornavn": "Kamilla", "efternavn": "Viktor", "email": "kvi@kea.dk"}
33   ];
34
35
36   function start() {
37     const listevisning = document.querySelector("#liste");
38     //løb personlisten igennem og indsæt i #liste
39     personer.forEach(person => {
40       //placer person i html
41       listevisning.innerHTML += `<p>${person.fornavn} ${person.efternavn} ${person.email}</p>`;
42     });
43   }
44 </script>
```

Variabel **listevisning** sættes lig den tomme liste

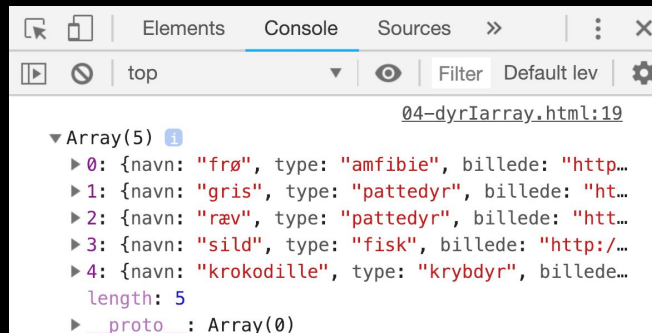
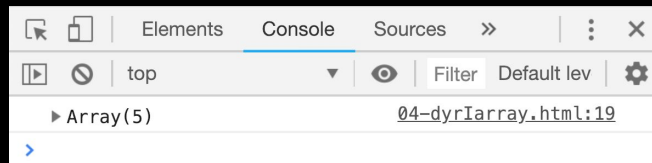
forEach-loopet sætter personerne ind i listen, **person** for person

← → ↻ ⓘ 127.0.0.1:57238/

Helle Frederiksen helf@kea.dk  
Martin Bregnhøj helf@kea.dk  
Kamilla Viktor helf@kea.dk

# Øvelse 4: Dyr i array

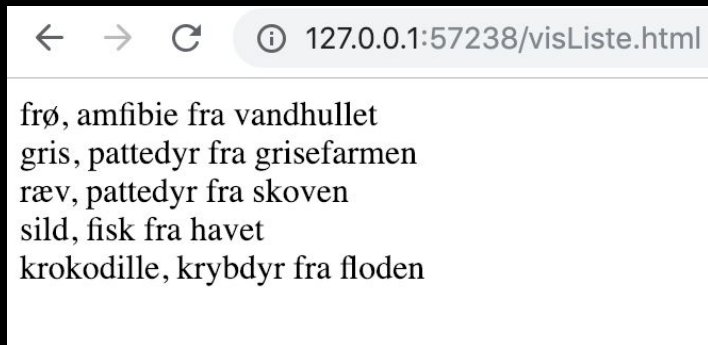
- Gem **02-visDyr.html** i en ny kopi, **04-dyrIarray.html**. Lav ændringer i scriptet:
- Erklær et array, med fem dyr: **gris**, **frø**, **ræv**, **sild** og **krokodille**  
Egenskaber: **navn**, **type**, **levested** og **billede**  
Billed-url'er:  
<http://mabe-kea.dk/E19/pics/frog.png>  
<http://mabe-kea.dk/E19/pics/pig.png>  
<http://mabe-kea.dk/E19/pics/fox.png>  
<http://mabe-kea.dk/E19/pics/sild.jpg>  
<http://mabe-kea.dk/E19/pics/kroko.jpg>
- Lav et script, som viser arrayet i Console.  
Kig i Console, og fold arrayet ud.





# Øvelse 5: Liste af dyr udskrives i DOM

- Gem **04-dyrlarray.html** i en ny kopi, **05-visAlleDyr.html** Lav ændringer i scriptet:
- Lav et script, som laver en liste over dyrene og deres typer og levested  
(husk at teste på om dokumentet er loaded ind før du arbejder med det.)



**Vis mange objekter i  
DOM-elementer**

...

# HTML <template> - hvad er det?

Når mange ens elementer skal indsættes dynamisk i vores HTML, kan vi bruge en **template**.

I tidligere slides brugte vi *innerHTML* += "...".

Nu skal vi bruge HTML template tag.

Et HTML template tag er en prædefineret HTML struktur der kan klones x antal gange, og fyldes med de data vi har i vores array.

tagget der definerer at det er et template element.

Indholdet i template tagget, der skal klones og fyldes med data.

```
<template>
  <section class="ret">
    <h2></h2>
    <div></div>
  </section>
</template>
```

OBS!

Template elementet er **ikke** synligt i DOM'en.

```
24 <section class="data-container"></section>
25 <template>
26   <article class="underviser">
27     <h3></h3>
28     <p class="email"></p>
29     <p class="github"></p>
30   </article>
31 </template>
```

**template-tag (usynligt i DOM)**

```
33 <script>
34   document.addEventListener("DOMContentLoaded", visUndervisere);
35   let undervisere = [
36     {"navn": "Alan Engelhart", "email": "ale@kea.dk", "github": "alan-engelhardt"},
37     {"navn": "Martin Bregnhøi", "email": "mabe@kea.dk", "github": "martinbregnhoi"},
38     {"navn": "Kamilla Viktor", "email": "kvi@kea.dk", "github": "kvikea"},
39     {"navn": "Klaus Mandal", "email": "klmh@kea.dk", "github": "MondaleMondale"}
40   ];
41
42   function visUndervisere() {
43
44     const container = document.querySelector(".data-container");
45     const underviserTemplate = document.querySelector("template");
46     //løb personlisten igennem og indsæt data i en template
47     undervisere.forEach(person => {
48       //placer person i html
49       let klon = underviserTemplate.cloneNode(true).content;
50       klon.querySelector("h3").textContent = person.navn;
51       klon.querySelector(".email").textContent = person.email;
52       klon.querySelector(".github").textContent = person.github;
53       container.appendChild(klon);
54     })
55   }
56
57 </script>
```

**kloning af template-taggets indhold**

**tilføj klonen til container (section tagget)**

# Øvelse 6: Dyreudskrivning med template

Gem **05-visAlleDyr.html** i en ny kopi - **06-visMedTemplate.html**.  
Lav følgende ændringer:

- Alle dyr skal vises i #liste, men denne gang skal de hver vises i et lidt mere komplekst element, som det du brugte til et enkelt dyr i øvelse 3 (03-vislelement.html).
- Lav om i HTML og script, så du bruger template tag til at generere opbygningen!
- Brug lidt styling - fx som her: en boks omkring dyret ;)
- Commit og push til GitHub

**frø**



Type: amfibie

Levested: vandhullet

**gris**



Type: pattedyr

Levested: grisehaven

**ræv**



Type: pattedyr

Levested: skoven

**sild**



Type: fisk

Levested: havet

**krokodille**

# Eftermiddags øvelse

...

# Øvelse 7: Events

På websiden:

<https://offstream.dk/archive>, ser man en liste - et “loop-view”, med forskellige events.

Du skal lave et site, [09-events.html](#), som viser de første 5 events:

1. Lav et array af objekter, som indeholder data for de 5 events (højreklik på billederne og “undersøg” for at få deres url-adresser).
2. Vis elementerne på din egen side efter inspiration fra originalen.
3. Udfra “mobile first” opsæt events’ne i et grid, så de vises i én kolonne på mobil, to kolonner på tablet( i portrait ) og tre kolonner på større skærme

