

# ES6

## Slideserie: ES6

DS3103 Webutvikling

Høst 2022

Rolando Gonzalez

# Temaer for slideserien

- Arrow functions
- forEach()
- filter()
- map()
- Spread operator
- sort()
- reduce()

# Arrow functions

- Arrow functions gjør at vi kan få mer korte og konsise funksjoner.
- I kodeeksempelet nedenfor vises en funksjon for å multiplisere to tall, i tre versjoner.
- Man kaller på arrow-funksjoner ved å referere til navn og sende inn argumenter: multiply3(7, 9)

```
//regular function
function multiply(number1, number2){
  return number1 * number2;
}

//arrow function v.1
let multiply2 = (number1, number2) => {
  return number1 * number2;
}

//arrow function v.2
let multiply3 = (number1, number2) => number1 * number2;
```

# Arrow functions

- `let multiply3` er deklarasjonen av funksjonen.
- Parantesen med `number1` og `number2` er parameterne; angir hva slags informasjon som sendes inn til funksjonen.
- Fra og med `=>` og utover kommer delen som skal utføres ved funksjonskall og returneres

```
let multiply3 = (number1, number2) => number1 * number2;
```

# Arrow functions

- Hvis det bare er 1 parameter trenger man ikke bruke parantes.

```
let priceWithTaxes = price => price + ( price * 0.25);  
  
alert(priceWithTaxes(100));
```

# Arrow functions

- Hvis funksjonen ikke skal ta imot argumenter bruker man en tom parantes.

```
let getVersion = () => "4.5";  
alert( getVersion() );
```

# Selvkallende arrow-funksjon

- Det kan være tilfeller hvor en ønsker å ha en arrow-funksjon som kjøres umiddelbart når siden laster

```
(  
    () => {  
        // Denne koden kjøres automatisk når siden laster*  
    }  
)();
```

\*med betingelse om at den er utenfor en funksjon

# forEach

- Foreach-løkken kan brukes for å gå gjennom objekt-lister eller lister med primitive verdier og modifisere.

## Syntax

```
arr.forEach(callback(currentValue [, index [, array]]), thisArg);
```

## Parameters

### callback

Function to execute on each element, taking three arguments:

### currentValue

The current element being processed in the array.

### index

Optional

The index of the current element being processed in the array.

### array

Optional

The array `forEach()` was called upon.

### thisArg

Optional

Value to use as `this` when executing `callback`.



# forEach

- Koden nedenfor viser bruk av forEach-løkken på en liste med string-verdier, med bruk av standard funksjon og arrow-funksjon.

```
let listOfCapitals = ["Oslo", "Stockholm", "Helsinki", "Paris", "London"];

listOfCapitals.forEach(function(capital){
    console.log(capital);
});

listOfCapitals.forEach( capital => console.log(capital) );
```

# forEach

- I en foreach kan man også få tak i indexen til elementet i listen, samt hele listen.

```
let listOfCapitals = ["Oslo", "Stockholm", "Helsinki", "Paris", "London"];  
listOfCapitals.forEach( (capital, index, listOfCapitals) => console.log(`Index: ${index}, ${capital}`) );
```

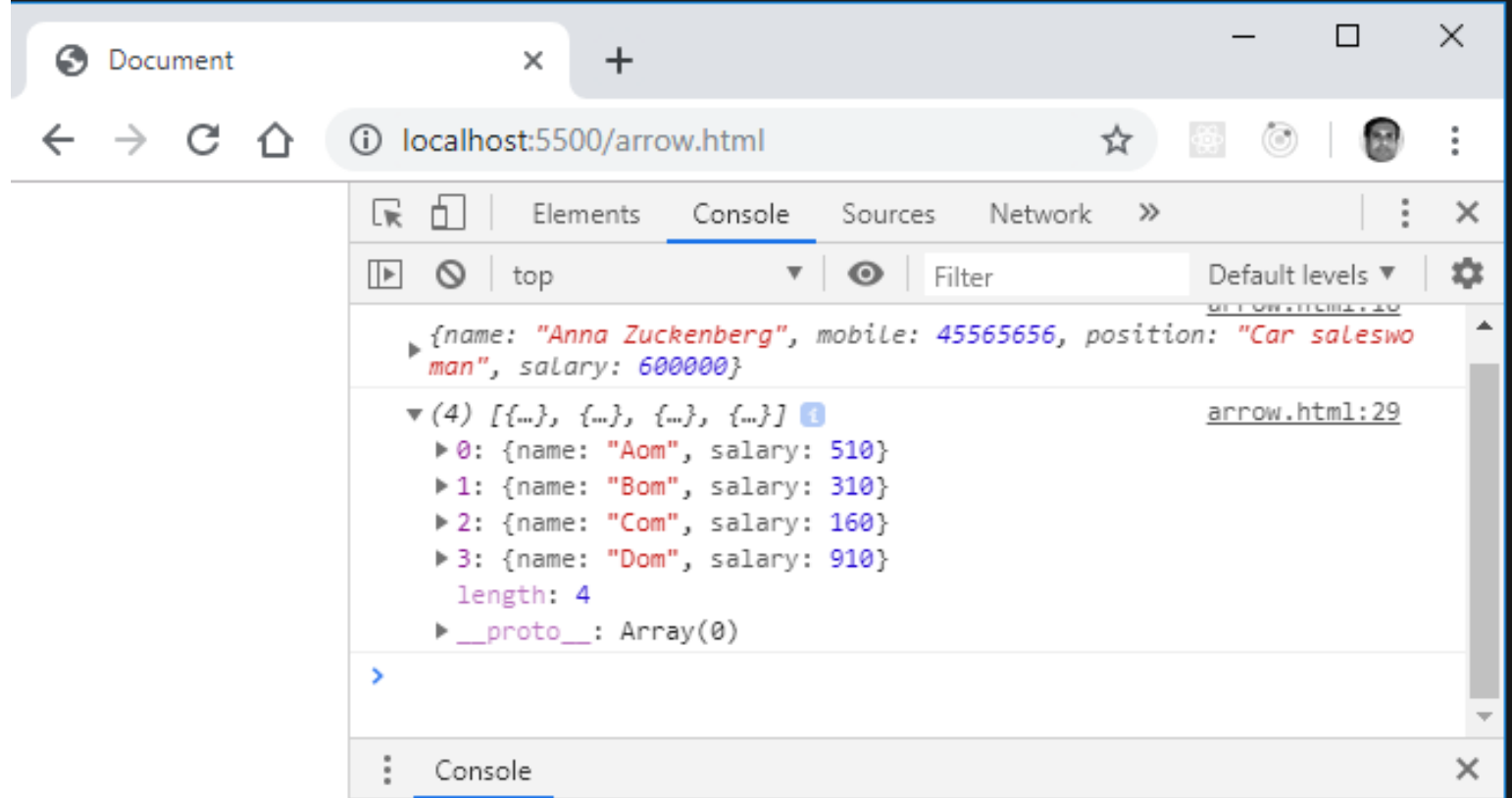
# forEach

- I dette eksempelet ønsker vi å modifisere alle objektene i listen ved å øke lønnen med 10.

```
let objects = [
  { name: "Aom", salary: 500 },
  { name: "Bom", salary: 300 },
  { name: "Com", salary: 150 },
  { name: "Dom", salary: 900 }
];

objects.forEach( object => object.salary = ( object.salary += 10 ) );

console.log( objects );
```

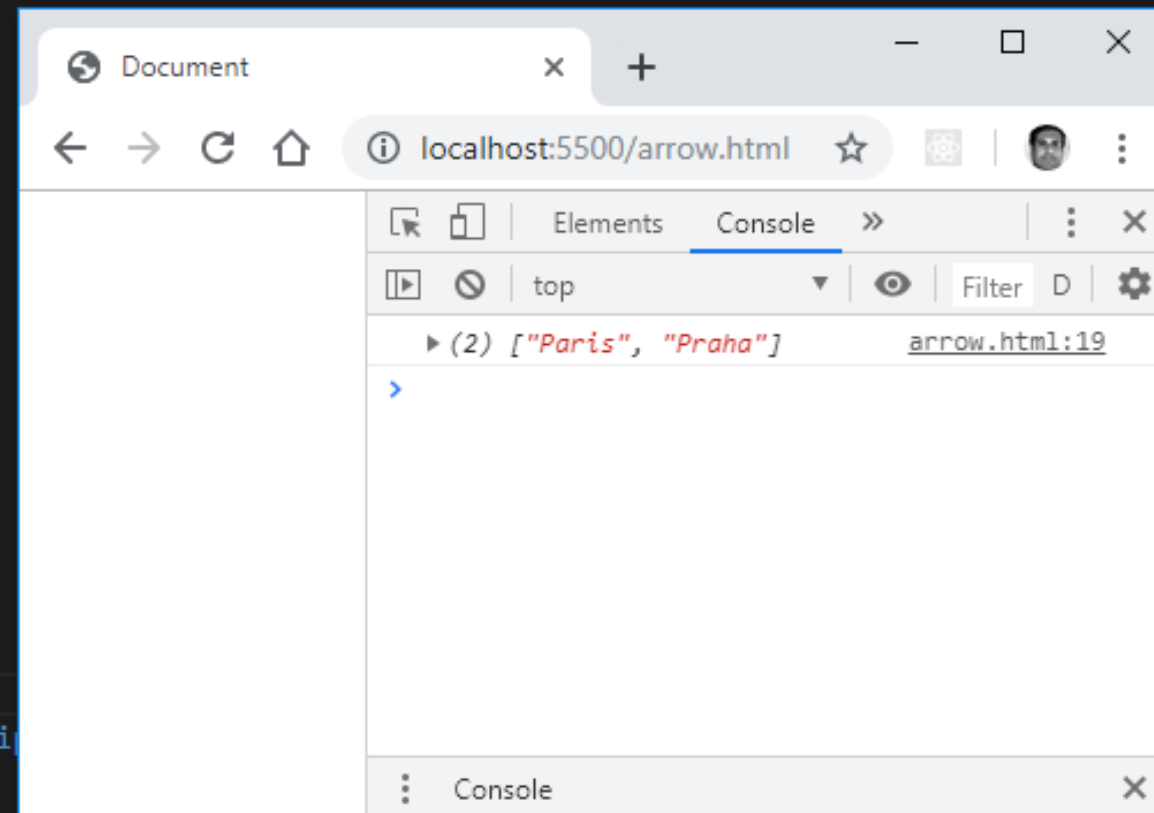


The screenshot shows a web browser window with the address bar displaying 'localhost:5500/arrow.html'. The console is open, showing the output of the JavaScript code. The console displays the modified array of objects, where each salary has been increased by 10. The objects are: {name: 'Anna Zuckenber...', mobile: 45565656, position: 'Car saleswo...', salary: 600000}, {name: 'Aom', salary: 510}, {name: 'Bom', salary: 310}, and {name: 'Com', salary: 160}. The array length is 4.

# filter

- Filter dreier seg om å få tak i et utvalg av elementer fra en liste basert på en betingelse.
- I eksempelet under blir variabelen capitalSearch satt til en liste som inneholder alle hovedstader som begynner med «P»; Paris og Praha her.

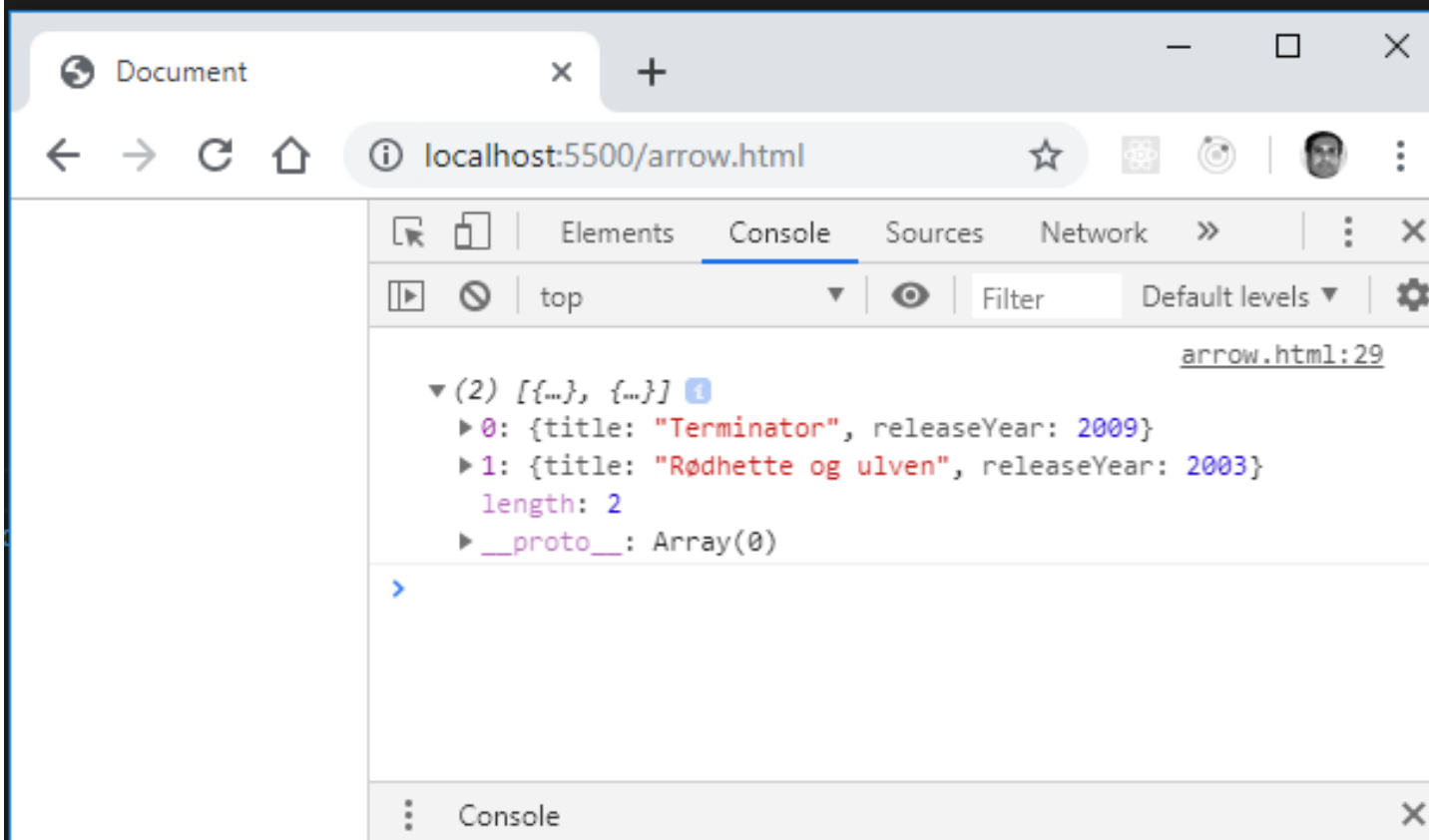
```
let listOfCapitals = ["Oslo", "Stockholm", "Helsinki", "Paris", "London", "Praha"];  
  
let searchLetter = "P";  
let capitalSearch = listOfCapitals.filter( capital => capital.charAt(0) == searchLetter );  
console.log(capitalSearch);
```



# filter

- I kodeeksempelet nedenfor jobber vi mot en JSON-objekt-liste og bruker dobbel betingelse.

```
let listOfMovies = [  
  { title: "Terminator", releaseYear: 2009 },  
  { title: "Predator", releaseYear: 2000 },  
  { title: "Rødhetten og ulven", releaseYear: 2003 },  
  { title: "La la land", releaseYear: 2015 }  
];  
  
let filteredMovies = listOfMovies.filter( movie => movie.releaseYear > 2000 && movie.releaseYear < 2015);  
  
console.log(filteredMovies);
```



# map

- Målet med map er å lage en ny liste (array) basert på en allerede eksisterende liste. Det kan for eksempel være fordi at vi kun trenger en liste med et utvalg av tilgjengelig informasjon, eller at vi ønsker å gjøre modifikasjoner på listen, eller at vi ønsker å formatere en tekst per element i listen.

# map

- I eksempelet nedenfor er målet å bruke map for å kun få tak i titlene på filmene.

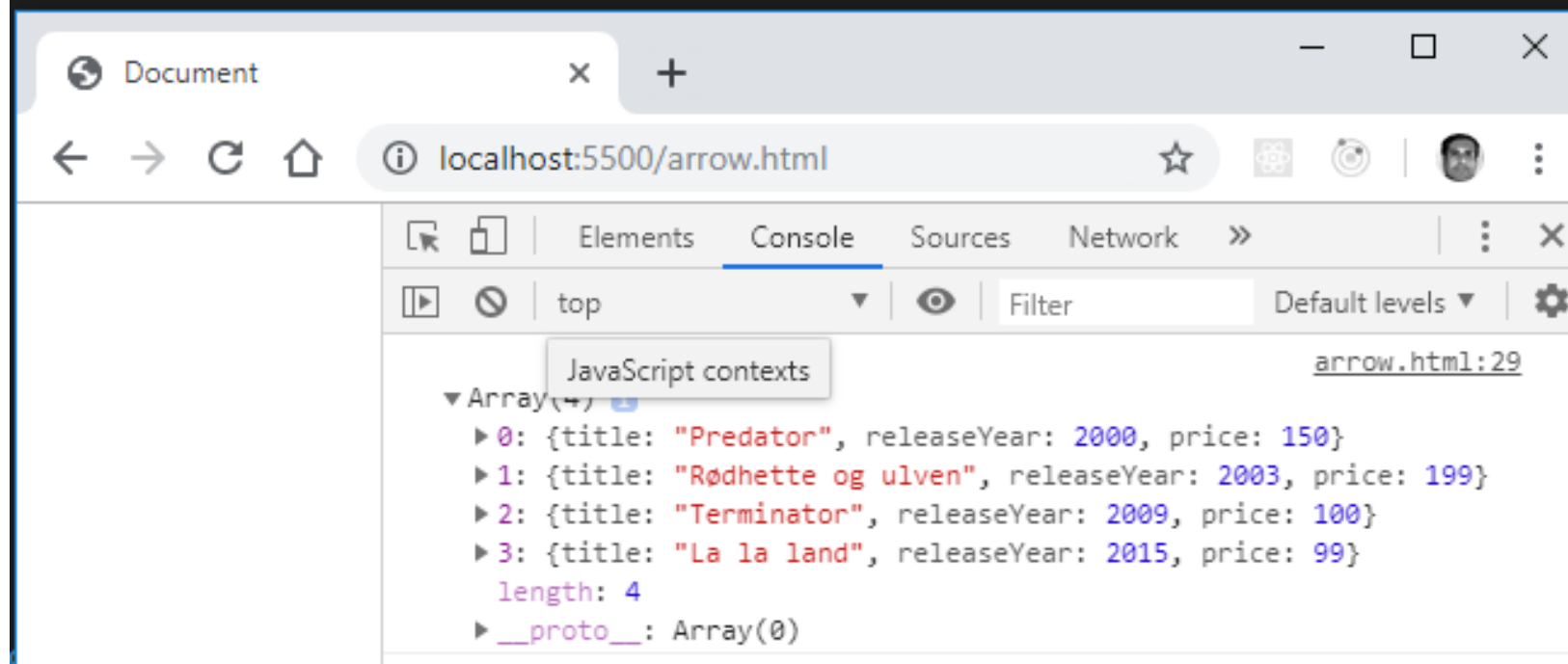
```
let listOfMovies = [  
  { title: "Terminator", releaseYear: 2009, price: 100 },  
  { title: "Predator", releaseYear: 2000, price: 150 },  
  { title: "Rødhette og ulven", releaseYear: 2003, price: 199 },  
  { title: "La la land", releaseYear: 2015, price: 99 }  
];
```

```
let listOfMovieTitles = listOfMovies.map( movie => movie.title );  
  
console.log(listOfMovieTitles);
```

# sort

- Sort brukes for å sortere lister etter en betingelse. Det er et krav om at man må returnere 1 eller -1 ved sammenligning.

```
let listOfMovies = [  
  { title: "Terminator", releaseYear: 2009, price: 100 },  
  { title: "Predator", releaseYear: 2000, price: 150 },  
  { title: "Rødhette og ulven", releaseYear: 2003, price: 199 },  
  { title: "La la land", releaseYear: 2015, price: 99 }  
];  
  
let sortedMoviesByYear = listOfMovies.sort( (movie1, movie2) => movie1.releaseYear > movie2.releaseYear ? 1 : -1 );  
  
console.log(sortedMoviesByYear);
```

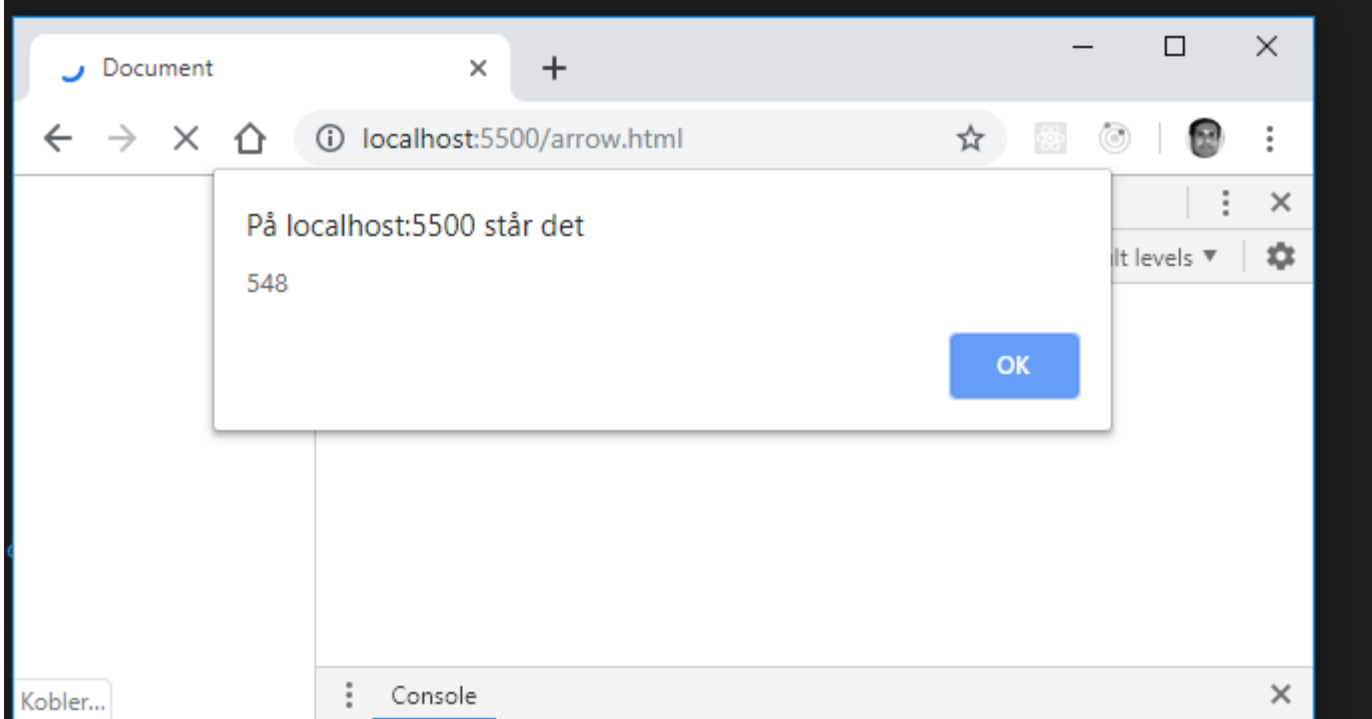




# Reduce

- Reduce dreier seg om å akkumulere, slå sammen, verdier i en liste for et sluttresultat. Reduce krever en «total»-verdi samt en oppstartsverdi; her satt til 0 helt til høyre.

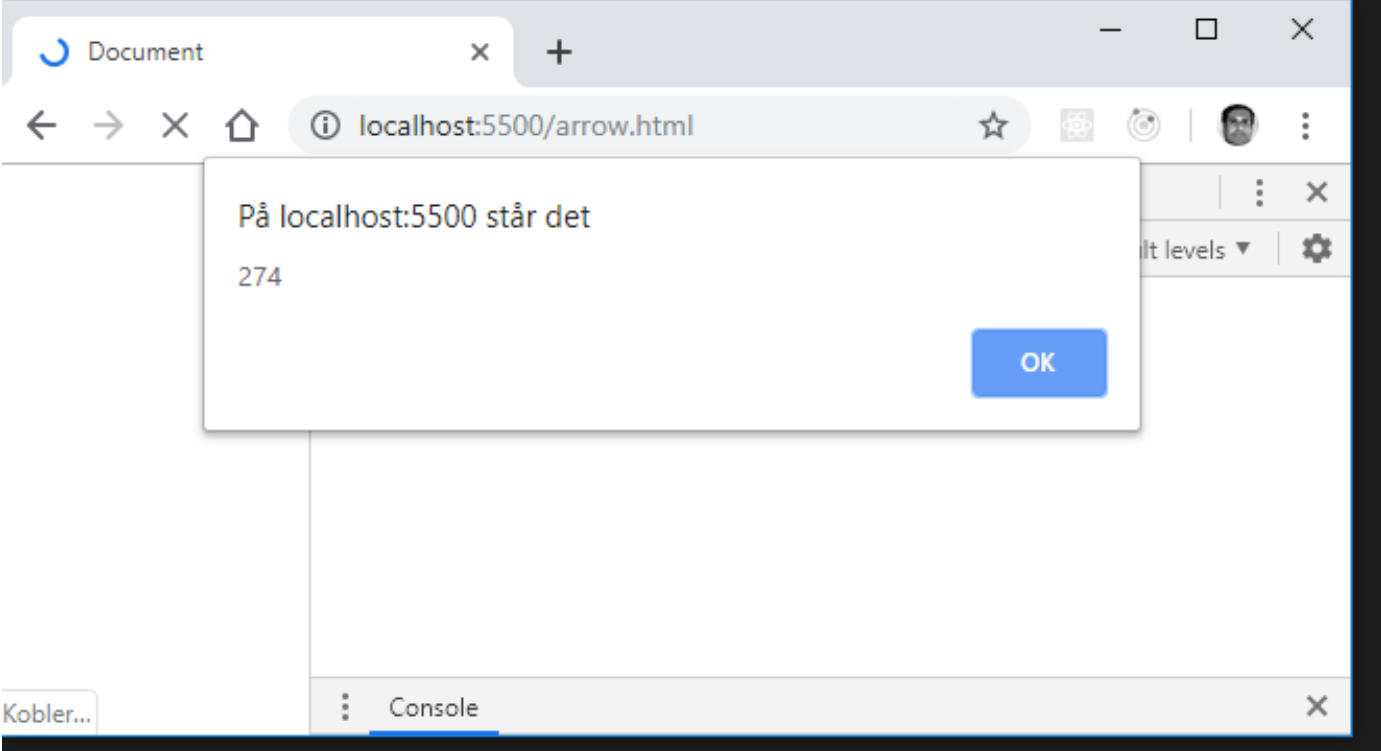
```
let shoppingCart = [  
  { title: "Terminator", releaseYear: 2009, price: 100 },  
  { title: "Predator", releaseYear: 2000, price: 150 },  
  { title: "Rødhetten og ulven", releaseYear: 2003, price: 199 },  
  { title: "La la land", releaseYear: 2015, price: 99 }  
];  
  
let totalPrice = shoppingCart.reduce( (total, product) => total + product.price, 0 );  
  
alert(totalPrice);
```



# Kombinere funksjoner

- Merk at man kan kombinere flere av de nevnte funksjonene etter hverandre.
- I eksempelet her kan vi tenke oss at alle produkter er nedsatt med 50%, slik at vi først får en liste med halverte priser for deretter å slå dem sammen.

```
let shoppingCart = [  
  { title: "Terminator", releaseYear: 2009, price: 100 },  
  { title: "Predator", releaseYear: 2000, price: 150 },  
  { title: "Rødhetten og ulven", releaseYear: 2003, price: 199 },  
  { title: "La la land", releaseYear: 2015, price: 99 }  
];  
  
let totalPrice = shoppingCart  
  .map( product => (product.price / 2) )  
  .reduce( (total, price) => total + price, 0 );  
  
alert(totalPrice);
```



På localhost:5500 står det  
274

OK

Kobler... Console

# Default parameter

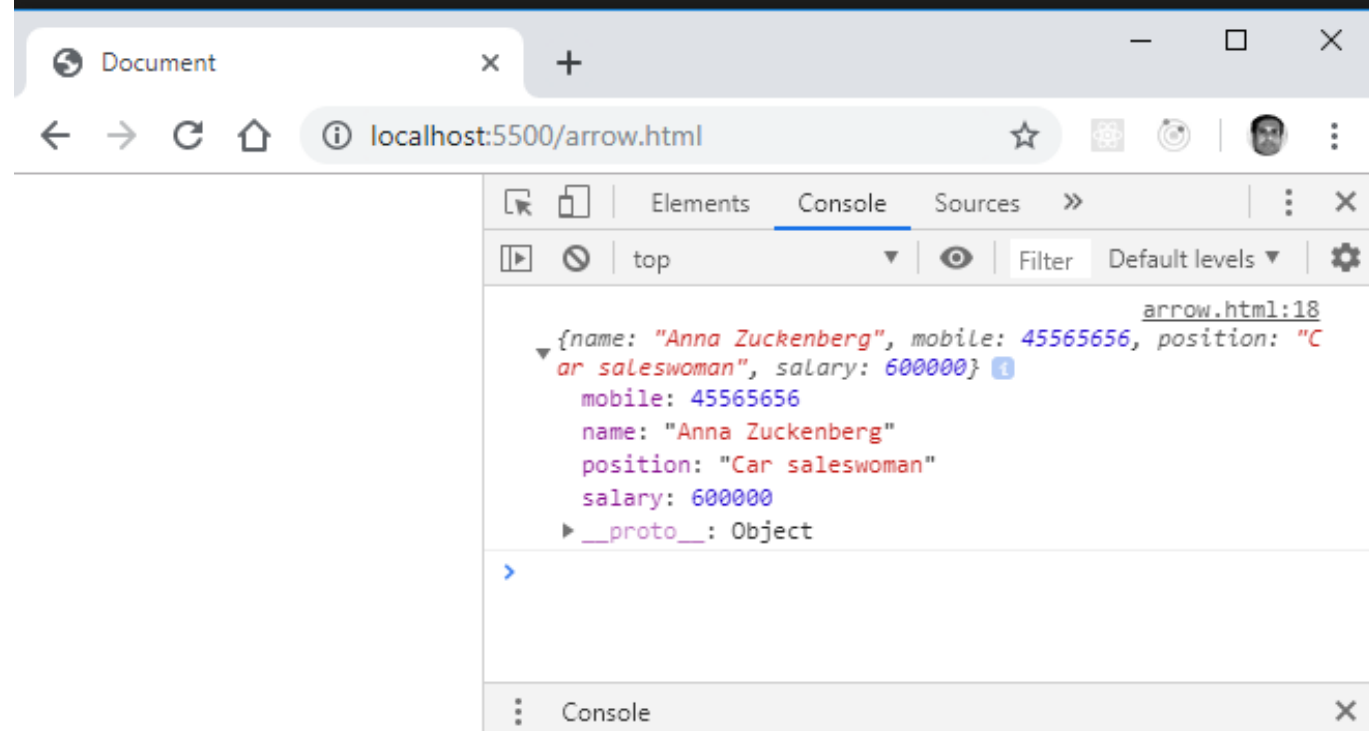
- Man kan sette verdier på variabler som vil gjelde, så sant de ikke settes gjennom funksjonskallet.

```
/*let alertTitle = (title) => alert(title);  
alertTitle("Flying in space!");*/  
  
let alertTitle = (title = "Title not set") => alert(title);  
alertTitle();
```

# Spread operator

- Spread operator dreier seg om muligheten for å kunne sende inn flere verdier i noe.
- I eksempelet nedenfor tenker vi at informasjonen om Anna Zuckenberg ligger i to separate funksjoner. Ved å bruke syntaks med ... får vi «spredd» attributtene inn i et felles nytt objekt.

```
let personInfo = { name: "Anna Zuckenberg", mobile: 45565656 };  
let ansattInfo = { position: "Car saleswoman", salary: 600000 };  
  
let mergedInformation = { ...personInfo, ...ansattInfo };  
  
console.log(mergedInformation);
```



# Primitive verdier

- Boolean
- Null
- Undefined
- Number
- BigInt
- String
- Symbol (new in ECMAScript 6)

[https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data\\_structures](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures)

# Litteratur / referanse

- Mozilla (2022): JavaScript Reference, <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference>, [last accessed 07.09.22]