Bilde: simonkr / iStock

Løpe gjennom arrayer med løkker

Vi har nå sett at alle verdiene i en array har sin egen posisjon, med et eget indeks-nummer. Fordi verdiene kan finnes med deres unike nummer, kan vi bruke løkker til å gå gjennom alle verdiene i en array.

Husk at en `for`-løkke kan gå gjennom en serie med hele tall som vi bestemmer. Hvis en array inneholder 10 verdier, så vil vi at løkken skal gå gjennom tallene 0, 1, 2, 3, 4, 5, 6, 7, 8 og 9. Vi starter på 0 fordi den første verdien alltid har indeks-nummer 0. En passende løkke kan derfor se slik ut:

```
for (let i = 0; i < 10; i++) {
  console.log(i);
}
```

Som nevnt ovenfor: Legg merke til at vi starter på 0 og holder på så lenge `i < 10`. Altså så lenge `i` er mindre enn arrayens lengde. Det skyldes at indeksnumrene starter på 0, slik at den siste verdien har indeks-nummer én mindre enn arrayens lengde. Nå skal vi se på hvordan vi kan bruke en løkke som denne på arrayer.

Enkle arrayer

La oss si at vi har følgende array:

```
let tall = [1, 2, 3, 4, 6, 8, 12];
```

For å gå gjennom alle verdiene i denne arrayen må vi først kjenne arrayens lengde. Tidligere i kapitlet nevnte vi egenskapen `length`. Vi kan bruke den til å finne antall verdier i arrayen `tall`:

```
console.log(tall.length); // Skriver ut: 7
```

Nå vet vi nok til å lage en `for`-løkke. Vi skal starte på 0, øke med én for hver runde, og holde på så lenge tallet er mindre enn arrayens lengde (her 7):

```
for (let i = 0; i < tall.length; i++) {
  console.log(tall[i]);
}
```

I dybden

for...of

I tillegg til den vanlige `for`-løkken, finnes det annen skrivemåte som kan brukes for å gå gjennom arrayer, nemlig `for…of`:

```
let tall = [1, 2, 3, 4, 6, 8, 12];

for (let enkeltTall of tall) {
  console.log(enkeltTall);
}
```

Her opprettes variabelen `enkeltTall` for hvert av tallene i arrayen. Vi kan kalle den variabelen hva vi ønsker.

I selve løkken får vi dermed tilgang til hver av verdiene i arrayen, én om gangen.

Begge skrivemåtene fungerer fint, og du velger selv hvilken du ønsker å bruke.

Med løkken ovenfor blir alle verdiene i arrayen `tall` skrevet til konsollen. Det er ikke så veldig spennende; så la oss heller gjøre noe nyttig med disse tallene. Vi kan for eksempel finne summen av dem.

Vi møter da en veldig vanlig situasjon der vi må lage en egen variabel for å holde orden på summen. Det er viktig at vi lager den variabelen *utenfor* løkken, slik at den ikke blir laget på nytt hele tiden. Vi gir også variabelen verdien 0 for å forsikre oss om at den får riktig datatype. Summen av tallene finner vi derfor slik:

```
let sum = 0;

for (let i = 0; i < tall.length; i++) {
  sum += tall[i];
}

console.log(sum); // Skriver ut: 36
```

Dette konseptet er veldig viktig, så vi tar et eksempel til før vi går videre. Hvordan kan vi finne det største tallet i arrayen `tall`? Tenk litt på det selv før du fortsetter å lese.

Vi kan tenke på samme måte som med sum-eksemplet ovenfor. Vi lager en variabel som vi sammenligner med hvert av tallene i arrayen. Hvis et tall er større enn variabelen, endrer vi variabelens verdi. Men husk at vi ikke vet noe om tallene i arrayen; de kan være veldig små, veldig store, eller de kan være negative. Vi tar derfor utgangspunkt i den første verdien i arrayen, og sammenligner de andre verdiene med den:

```
// Lagrer det første tallet i arrayen i en variabel
let storst = tall[0];

for (let i = 0; i < tall.length; i++) {
  if (tall[i] > storst) {
    storst = tall[i];
  }
}

console.log(storst); // Skriver ut: 12
```

Snakk!

Hvorfor starte med et av tallene i arrayen?

Hvorfor tror du det er lurt å starte med et tall i arrayen når vi skal finne det største tallet ovenfor? Kunne vi ikke bare ha satt verdien til 0?

Det finnes også andre måter å gjøre dette på. Vi kan for eksempel sortere tallene i arrayen (se senere i læringsløpet) og velge den siste verdien, eller vi kan bruke innebygde funksjoner. Det er likevel mye å lære ved å gjøre slike ting «manuelt» når du skal lære å programmere.

Oppgave

Lag en array med 20 forskjellige tall. Velg noen negative og noen positive tall.

5

a Bruk programmet ovenfor for å finne den *største* verdien i arrayen. Sjekk at du finner riktig tall.

b Skriv et program som finner den *minste* verdien i arrayen. Sjekk at du finner riktig tall.

c Skriv et program som finner *variasjonsbredden* for verdiene i arrayen. Variasjonsbredde er forskjellen mellom den største og den minste verdien i et datasett.

d Skriv et program som finner gjennomsnittet av verdiene i arrayen.

Arrayer i funksjoner

Vi har tidligere sett på funksjoner og parametrene de kan motta. En array kan brukes som en parameter på samme måte som andre variabler. Vi kan for eksempel lage en funksjon som summerer tallene i en array:

```
function summer(tallArray) {
  let sum = 0;
  for (let i = 0; i < tallArray.length; i++) {
    sum += tallArray[i];
  }

  return sum;
}
```

Denne funksjonen bruker vi slik:

```
let mineTall = [2, 4, 6, 8];
console.log(summer(mineTall));
```

Her blir summen, som i dette tilfellet er 20, skrevet til konsollen.

I dybden

Kopiering av arrayer

Iblant kan du havne i en situasjon der du vil kopiere en eksisterende array. La oss prøve på det nå. Se på koden nedenfor. Hva tror du blir resultatet?

```
let a = [1, 2, 3];
let b = a;

b[1] = 4;

console.log(a);
console.log(b);
```

Vi får den same utskriften to ganger! Det er fordi `b` ikke er en egen array, men det vi kaller en *peker* til arrayen `a`. De to representerer altså *den samme arrayen*.

Det finnes heldigvis måter å kopiere en array på slik at vi får en ekte kopi. Du kan for eksempel bruke `Array.from()`:

```
let b = Array.from(a);
```

Oppgaver

6 Gjenta oppgave 5 ovenfor, men denne gangen skal du lage funksjoner som får en array som parameter. Du skal altså lage funksjoner som finner minste og største verdi, variasjonsbredden og gjennomsnittet. Husk at én funksjon kan bruke andre funksjoner.

7 Lag en yatzy-funksjon. Funksjonen skal motta to arrayer som parametre. Den ene skal inneholde fem tall (terningkast) og den andre skal inneholde de verdiene du ønsker å *ikke* spare på. Hvis tallene er `[1, 2, 1, 1, 1]`, så vil du kanskje spare på enerne og *ikke* spare på `[2]`. Alternativt kan array nummer to inneholde *posisjonene* til tallene du ikke vil spare på. Hvis array nummer to er helt tom, trilles alle fem terningene.

Todimensjonale arrayer

Som nevnt tidligere kan vi også ha arrayer inni arrayer. Hvis vi har to arrayer inni hverandre, kaller vi det en *todimensjonal array*. Vi kan for eksempel lage én hovedarray med tre verdier: forretter, hovedretter og desserter. Disse tre verdiene kan *også* være arrayer, slik at vi har én array med forretter, én med hovedretter og én med desserter. Arrayen vår kan for eksempel se slik ut:

```
let retter = [
  ["laksetartar", "ertesuppe"],
  ["biff med rødvinssaus", "lammeskank"],
  ["jordbæris", "eplekake"]
];
```

Her har vi skrevet arrayen over fem linjer. Det er ikke nødvendig, men det gjør koden mye lettere å lese. Her er det for eksempel lett å se hva som er forretter, hva som er hovedretter, og hva som er desserter.

Nå som vi har arrayen, må vi finne ut hvordan vi kan gå gjennom alle verdiene med en løkke. Prøv gjerne selv før du leser videre.

Vi vet allerede hvordan vi kan lage en løkke for å gå gjennom de tre verdiene i hovedarrayen:

```
for (let i = 0; i < retter.length; i++) {
  console.log(retter[i]);
}
```

Denne koden skriver ut de tre «indre» arrayene til konsollen, men hva om vi vil ha alle rettene skrevet ut? Vi vet at forrettene ligger i `retter[0]`, hovedrettene i `retter[1]` og dessertene i `retter[2]`. Vi vet også at den første desserten ligger i `retter[2][0]`, og at den andre forretten ligger i `retter[0][1]`. Vi må derfor gå gjennom både den første indeksen, for å finne forrett, hovedrett og dessert, *og* den andre indeksen for å finne enkeltrettene. Vi kan lage en ny løkke inni den første, slik at vi får gått gjennom hver av de indre arrayene også:

```
for (let i = 0; i < retter.length; i++) {
  for (let j = 0; j < retter[i].length; j++) {
    console.log(retter[i][j]);
  }
}
```

Her får vi tilgang til alle enkeltverdiene i de «indre» arrayene. Denne framgangsmåten gjør også at de «indre» arrayene kan ha ulik lengde, fordi vi sjekker lengden til hver av dem. Legg merke til at vi bruker ulike tellere (`i` og `j`). Vi kan ikke bruke `i` i begge løkkene fordi det vil endre

variabelen og påvirke den andre løkken.

Snakk!

Samme variabelnavn i begge løkkene

I eksemplet ovenfor brukte vi variabelnavnene `i` og `j`. Hvorfor tror du det går galt hvis vi bruker `i` i begge løkkene?

Finn ut hva som skjer med variabelen hvis vi gjør det slik.

Det er naturlig å bruke todimensjonale arrayer hvis vi skal representere et rutenett. Da kan vi la den første arrayen representere rader, og de indre arrayene kan representere kolonner. Da kan vi for eksempel lage en array som kan brukes til å lage et tre-på-rad-spill:

```
let trePaaRad = [
  [_,_,_],
  [_,_,_],
  [_,_,_]
];
```

Her brukes `_` for å angi at feltene er tomme. Vi kan bruke liten «x» og liten «o» for å representere de to spillerne.

Oppgaver

8 Lag en todimensjonal array som består av tre arrayer, hver av de tre arrayene skal inneholde tre tall. Tegn en tabell (for hånd) som viser indeksene til hver verdi i den todimensjonale arrayen.

9 Lag en todimensjonal array som inneholder en gangetabell. Du skal kunne angi faktorer som indekser og få ut produktet av dem som verdi. Det vil for eksempel bety at `gangetabell[3][7]` skal inneholde verdien 21.