

</> JS: Grunnleggende JavaScript

Introduksjon

I denne oppgaven skal du lære helt enkle og grunnleggende elementer i JavaScript: **setninger**, **funksjoner** og **løkker**.

For å gjøre denne oppgaven trenger du ingen forkunnskaper innenfor HTML og CSS når man skal programmere JavaScript.

Hvis HTML er en bil, så er CSS utseendet og designet på bilen, mens JavaScript er motoren, få bilen til å kjøre osv. Hvis du har programmert et tekstlig program vil nok mye i denne oppgaven være kjent.

La oss begynne!

Steg 1: Variabler

Dersom du gjorde oppgaven [Hei JavaScript](#) så er nok variabler kjent. Men litt repetisjon skader ikke!

En variabel i JavaScript ser slik ut:

```
var variabelNavn = "verdi";
```

var forteller JavaScript at det er en variabel. En variabel kan være en **string**, et **tall**, eller en **objekt**.

```
var tall = 9;  
var tekst = "Dette er en string (tekst på engelsk)";  
var variabel = tall; // "variabel" vil nå ha samme verdi som tall
```

La oss prøve oss litt frem!

- ☐ Gå inn på [JSbin.com](https://jsbin.com) og sørg for at fanene `JavaScript` og `Console`
- ☐ I `JavaScript`-vindu skriver du følgende:

```
var tall = 9;  
var tekst = "Hei på deg!";
```

Forklaring

Husk at hver linje i JavaScript avsluttes med enten `;` eller `}`, avhengig av hva du skriver. Når vi bruker variabler avslutter vi med `;`, med funksjoner avslutter vi med `}`.

For at vi skal kunne skrive ut noe til `Console` bruker vi `console.log()`.

```
var tekst = "Hei på deg!";  
console.log(tekst);
```

- ☐ Trykk på `Run`
- ☐ Vises teskten?
- ☐ La oss prøve oss på litt variabel-morro. Lag følgende variabler, c

```
var tall1 = 4;  
var tall2 = 7;
```

- ☐ Nå skal vi ta de to variablene og plusse dem sammen:

```
console.log(tall1 + tall2);
```

- ☐ Trykk `Run` . Fikk du 11?
- ☐ Nå som vi vet at vi kan få JavaScript til å regne for oss, ta å bytt JavaScript klarer å regne ut med dem.

La oss nå se på hvordan vi kan la en variabel være en annen:

- ☐ Legg til enda en variabel:

```
var tall3 = tall2;
```

- ☐ Hva blir skriver `console.log(tall3 * tall2)` ut?

Svar

- ☐ Skriv ut `tall3` og `tall2` til `console`
- ☐ Prøv å endre på `tall2` , hva skjer med `tall3` ?

`tall3` er jo satt til å være det samme som `tall2` og derfor blir den e
Vi kan også legge til ekstra verdier til en variabel:

```
var tall4 = tall3 + tall1 + 5;
```

- ☐ Bruk `console.log` til å skrive ut `tall4` .

Nå har vi fått prøvd ut litt forskjellige variabler, da skal vi bruke dette sant eller ikke.

Steg 2: If-setninger

En `if/else`-setning ser slik ut i JavaScript:

```
if(betingelse) {  
    // Kjør koden som blir skrevet her  
} else {  
    // Kjør koden som blir skrevet her istedet  
}
```

Når vi bruker `if/else` sjekker vi en betingelse og basert på om betingelsen er sann eller falsk, vil en eller flere koder bli kjørt. La oss se på et eksempel med tall.

☐ Skriv dette inn i JSBin:

```
var tall = 5;  
  
if(tall === 5) {  
    console.log("Tallet er midt mellom 1 og 10");  
} else {  
    console.log("Tallet er enten over eller under 5");  
}
```

Forklaring

Dersom `tall` har verdien `5` vil den første meldingen skrives ut, ellers vil den andre meldingen skrives ut. For å sjekke om en variabel er lik noe så brukes operatoren `===`. Det er viktig å merke seg at datatypen (nummer, tekst osv) er like.

☐ Endre variabelen `tall` til andre verdier og se hvilke melding som skrives ut.

☐ La oss lage sjekk på om du kan ta sertifikatet til bil eller moped:

```
var alder = 0;

if(alder >= 18) {
    console.log("Du er gammel nok til å kjøre bil");
} else if(alder >= 16) {
    console.log("Du er gammel nok til å kjøre moped");
} else {
    console.log("Du er dessverre ikke gammel nok");
}
```

Forklaring

- ☐ `if(alder >= 18)` betyr: hvis alder er større eller lik 18. Altså sann og du er gammel nok til å kjøre bil.
- ☐ `else if(alder >= 16)` betyr: dersom `if`-testen var usann, hvis denne betingelsen er sann så sier den at du er gammel nok til å kjøre moped.
- ☐ `else` betyr `ellers` og vil si at koden til denne kjører dersom alder er under 16 år så får du beskjed om at du ikke er gammel nok.

- ☐ Prøv å endre `alder` slik at du får testet om `if/else`-setningene fungerer.
- ☐ La oss legge til noe som heter `prompt`, dette gjør at du kan ta inn input fra brukeren.

```
var alder = prompt("Hvor gammel er du?");
```

Nå skal vi ta dette opp ett hakk hvor vi skal sjekke hva klokken er og hva datoen er. I JavaScript heter `Date`, denne inneholder informasjon om dagen i dag og klokken på nå.

- ☐ Vi lager to variabler, en som er en `Date`-klasse og en annen som

```
var dato = new Date(); // Henter informasjon om dagen i dag
var tid = dato.getHours(); // Henter timen (klokka) vi er i nå
```

- ☐ Bruk `console.log` til å sjekke hva variabelen `tid` inneholder.

Før du skal få en oppgave må vi gå igjennom noen verktøy vi kan bruke

Forklaring

- ☐ I en `if(betingelse)` kan vi sjekke flere ting samtidig ved å bruke `&&`.
Finner du ikke disse på tastaturet, så spør en voksen om å hjelpe deg.

```
var date = new Date();
var mnd = date.getMonth();
var dato = date.getDate();

if(navn === "Lars" && mnd === 7 && dato === 10) { // mnd = 7 er August
  console.log("Gratulerer med navnedagen, Lars!");
}
```

For at denne koden skal være sann må navnet være `Lars` og dato leverer ut et tall fra 0-11, derfor er August 7 og ikke 8. Les mer om [Date](#).

Samme kan du gjøre med `||` hvis du heller vil sjekke `eller`:

```
if(mnd === 7 && dato === 10) {
  if(navn === "Lars" || navn === "Lasse") {
    console.log("Gratulerer med navnedagen!");
  } else {
    console.log("Du har dessverre ikke navnedag i dag.");
  }
}
```

```
}
```

Legg merke til her at hvis datoen er `10.8` så hopper du inn til en `Lars` eller `Lasse`. Hvis det er feil dato skjer det ingenting.

Nå håper jeg du har fått litt mer dreisen på `if/else`, hvis ikke kommer `if/else` er det mye å utforske, så det tar litt tid og erfaring å bli vant

Nå skal vi se på `funksjoner`!

Steg 3: Funksjoner

Til nå har vi bare skrevet linjer med kode i `JSBin`. Når koden kjøres, så vil kjøre i den rekkefølgen den står i. Men noen ganger ønsker vi at noe annet skjer. Ved hjelp av `funksjoner` kan vi selv bestemme når koder

Oppsettet ser slik ut:

```
function funksjonsNavn(parameter1, parameter2) {  
    // Kode som utføres når funksjonen kalles  
}  
  
funksjonsNavn(paramenter1, parameter2); // Gjør at funksjonen
```

En funksjon tar noen ganger inn noen variabler den ønsker å bruke ved `parameter` og da ser en funksjon sånn ut:

```
function navn() {  
    // Kode  
}  
  
navn(); //for å kjøre funksjonen
```

En funksjon kalles ofte for en `metode`. Videre kommer vi til å bruke `fu`

- ☐ Ta nå koden som har med alder å gjøre, og legg den inn i en fun

Hint

- ☐ Legg på kode for at `sjekkAlder` skal kjøre.

Hint

La oss nå se på et annet eksempel på bruk av funksjoner. Vi skal nå la `returnere` en verdi.

Vi lager en funksjon som konverterer `fahrenheit` (temperatur mål i US.

- ☐ Slett det du har i JSBin eller åpne et nytt vindu i JSBin (File -> Ne
- ☐ Lag følgende funksjon:

```
function fahrenheitTilCelsius(fahrenheit) {  
  return (5/9) * (fahrenheit - 32);  
}
```

Forklaring

- ☐ `fahrenheitTilCelsius(fahrenheit)` betyr at vi tar inn en va innenfor `{ }` i funksjonen vår.
- ☐ `return (5/9) * (fahrenheit - 32)` bruker variabelen `fahr` der.
- ☐ Men hva betyr `return`?
- ☐ Return betyr at funksjonen returnerer en verdi til brukeren se på et eksempel:


```
var grader = fahrenheitTilCelsius(77);
```

- ☐ Koden over kjører funksjonen `fahrenheitTilCelsius` med parameteren `77` fahrenheit er og lagrer denne verdien i variabelen `grader`.

Nå som vi har fått litt kontroll på hva en `funksjon` eller `metode` er, skal vi se på løkker.

Steg 4: Løkker

Vi som programmerer er ganske late og derfor bruker vi verktøy til å gjøre repetisjoner. Løkker kan vi gjenta kode istedet for å skrive mange linjer med kode. Det finnes to typer løkker: `for`-løkker og `while`-løkker.

For-løkke

```
for(var i = 0; i < 10; i++) {  
    // Kode som kjøres ett gitt antall ganger  
}
```

- ☐ `var i = 0;` dette er en variabel som fungerer som en teller, den starter på 0 og øker med 1 for hver runde løkken kjører.
- ☐ `i < 10;` dette er en betingelse som forteller løkken om den skal kjøre så lenge variabelen `i` er mindre enn 10.
- ☐ `i++;` øker variabelen `i` med 1 for hver runde løkken kjører.

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

Dette skrives ut i konsollen:

```
0  
1  
2  
3  
4  
5  
6  
7  
8  
9
```

Man kan gjøre det samme hvis man for eksempel skal skrive ut en handle listen din:

```
var handleliste = ["melk", "egg", "smør", "toalettpapir",  
  
for(var i = 0; i < handleliste.length; i++) {  
    console.log(handleliste[i]);  
}
```

- ☐ `[]` betyr liste, og alle elementene som er i listen blir separert.
- ☐ `handleliste.length()` er et tall på hvor stor listen er. Skriv hva slags tall du får ut.
- ☐ Når vi skal skrive ut et gitt element fra en liste skriver vi `handleliste[0]` for det første elementet, `handleliste[1]` for det andre elementet, osv.

 Prøv selv!

- ☐ Åpne ny side i JSBin
- ☐ Prøv å skriv ut tallene fra 1 - 100 ved hjelp av en `for`-løkke
- ☐ Bytt på telleren `i++` slik at alle partall mellom 1 og 100 skrives

Hint

- ☐ Klarer du å skrive ut alle oddetallene også?

Bra! La oss se på lister.

- ☐ Lag nå en liste over dine favoritt spill
- ☐ Skriv ut alle ved hjelp av en `for`-løkke
- ☐ Skriv ut annen hvert element i lista (hint: bruk samme metode som

While-løkke

```
while(betingelse) {  
    // Kjør kode til betingelsen er usann  
}
```

En `while`-løkke kjører en blokk med kode helt til `betingelsen` blir falsk. I eksemplet skal lage spill:

```
while(!gameOver) {  
    // Kjør spill  
}
```

`!` betyr `not`, altså betyr `!gameOver` at løkken kjører så lenge spill ikke er over. Hvis løkken kjøre for alltid kan vi sette betingelsen til `True`.

Vi kan også gjøre det samme som vi gjorde med `for`-løkken:

```
var i = 0;
while(i < 10) {
  console.log(i);
  i++;
}
```

Forskjellen her er at vi må lage en tellende variabel som vi definerer (`i++`) legger vi etter all koden vi har løkken sånn at den teller opp

✓ Prøv selv

- ☐ Skriv om `while`-løkken til å skrive ut alle tallene fra 1-100
- ☐ Skriv om løkken slik at du skriver ut alle partallene fra 1-100
- ☐ Bruk listen din over favorittspill og skriv ut alle elementene ved

La oss nå lage et enkelt `Kron eller mynt`-spill ved hjelp av `whi`

- ☐ Åpne en ny [JSbin.com](https://jsbin.com)
- ☐ Lag en variabel, `krone`, som skal være enten 0 eller 1, dette sk

```
var krone = Math.floor(Math.random()*2);
```

`Math.floor()` runder ned tallet du får fra `Math.random()`. `Math.rand` for at tallet blir mellom 0-2.

- ☐ Lag en `while`-løkke som skal kjøre helt til du får `mynt`. Vi sier n

```
while(krone === 0) {  
  console.log("Kron! Vi flipper igjen...");  
  var krone = Math.floor(Math.random()*2);  
}  
console.log("Mynt! Gratulerer!");
```

Bra jobba! Nå har du lært masse om JavaScript!

Utfordring

- ☐ Bruk `prompt` til å ta inn et valg fra brukeren sånn at du selv
- ☐ Klarer du å gjøre det samme med terninger? Da må du ha ta

Lisens: CC BY-SA 4.0