JS: Grunnleggende JavaScript

Skrevet av: Lars Klingenberg

Kurs: Web
Tema: Tekstbasert, Nettside
Fag: Matematikk, Programmering, Teknologi
Klassetrinn: 5.-7. klasse, 8.-10. klasse, Videregående skole

Introduksjon

I denne oppgaven skal du lære helt enkle og grunnleggende elementer av `JavaScript`. Du vil lære om `variabler`, `if-setninger`, `funksjoner` og `løkker`.

For å gjøre denne oppgaven trenger du ingen forkunnskaper innenfor HTML eller CSS, men det er veldig lurt å kunne HTML og CSS når man skal programmere JavaScript.

Hvis HTML er en bil, så er CSS utseendet og designet på bilen, mens JavaScript står for all funksjonalitet som å skru på motoren, få bilen til å kjøre osv. Hvis du har programmert et tekstlig programmeringsspråk som for eksempel `Python` før, så vil nok mye i denne oppgaven være kjent.

La oss begynne!

Steg 1: Variabler

Dersom du gjorde oppgaven Hei JavaScript (../hei_js/hei_js.html) så er nok variabler kjent. Samme gjelder hvis du har programmert litt fra før. 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 inneholder en verdi, som for eksempel `tall` eller `tekst`:

```
var tall = 9;
var tekst = "Dette er en tekst";
```

La oss prøve oss litt frem!

- [ ] Gå inn på JSbin.com (https://jsbin.com/?js,console) og sørg for at fanene `JavaScript` og `Console` er markert. Dette gjør du ved å trykke på dem

- [ ] 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 om du lager en variabel eller en funksjon. Når vi bruker variabeler 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 teksten

- [ ] La oss prøve oss på litt variabel-morro. Lag følgende variabler, du kan godt slette det du allerede har:

```
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 ut `+` med de andre regneartene vi har og se om 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 `console.log(tall3 * tall2)`?

Svar

- [ ] Skriv ut `tall3` og `tall2` til `console

- [ ] Prøv å endre på `tall2`, hva skjer med `tall3`?

Vi satt jo `tall3` til å være lik `tall2`? `tall3` blir ikke endret selvom vi endrer på `tall2`. Dette er fordi tall-variabler i JavaScript inneholder kun verdier, ikke referanser.

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 variabeler, da skal vi bruke dette sammen med `if-setninger` for å sjekke om noe er 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 usann, så kjører vi en gitt kode. La oss se på et eksempel med tall.

- [ ] Skriv dette inn i JSBin:

```
var tall = 5;

if(tall === 5) {
    console.log("Tallet er 5");
} else {
    console.log("Tallet er ikke 5");
}
```

Forklaring

Dersom `tall` har verdien `5` vil den første meldingen skrives ut, dersom `tall` har en annen verdi, så vil den andre meldingen skrives ut. For å sjekke om en variabel er lik noe så bruker vi `===` . Da sjekker JavaScript om verdien og datatypen (nummer, tekst osv) er like.

- [ ] Endre variabelen `tall` til andre verdier og se hvilke melding du får 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 er du 18 år eller eldre vil denne setningen være sann og du er gammel nok til å kjøre bil.
- `else if(alder >= 16)` betyr: dersom `if` -testen var usann, så sjekker den om alder er større eller lik 16, og 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 de andre testene blir usanne. Altså hvis du 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 av nettsiden:

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

Nå skal vi ta dette opp ett hakk hvor vi skal sjekke hva klokken er og skrive en melding ut i fra det. Da bruker vi noe som heter `Date` i JavaScript, denne inneholder informasjon om dagen i dag. Dette er en klasse, hva det vil si skal vi ikke fokusere på nå.

- [ ] Vi lager to variabler, en som er en `Date` -klasse og en annen som henter timen vi er i nå:

```
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 i `if/else`:

Forklaring

I en `if(betingelse)` kan vi sjekke flere ting samtidig ved å bruke `&&` eller `||`. `&&` betyr `og`, `||` betyr `eller`. Finner du ikke disse på tastaturet, så spør en voksen om å hjelpe deg. Eksempel:

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

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

For at denne koden skal være sann må navnet være `Lars` og datoen må være `10.8`, altså 10. August. `getMonth()` leverer ut et tall fra 0-11, derfor er August 7 og ikke 8. Les mer om `Date` her (http://www.w3schools.com/jsref/jsref_obj_date.asp).

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 ny `if/else` som sjekker om navnet ditt enten er `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 vi til å jobbe mer med dette senere. Innen for `if/else` er det mye å utforske, så det tar litt tid og erfaring å bli vant til alt.

Nå skal vi se på `funksjoner`!

Steg 3: Funksjoner

Til nå har vi bare skrevet linjer med kode i `JSBin`. Når koden kjøres, så blir den lest fra topp til bunn, linje for linje, så koden vil kjøre i den rekkefølgen den står i. Men noen ganger ønsker vi at koden skal kjøre når en hendelse inntreffer eller noe annet skjer. Ved hjelp av `funksjoner` kan vi selv bestemme når koden skal kjøre eller om den ikke skal kjøre i det hele tatt.

Oppsettet ser slik ut:

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

funksjonsNavn(paramenter1, parameter2); // Gjør at funksjonen blir kjør
t
```

En funksjon tar noen ganger inn noen variabler den ønsker å bruk ved hjelp av `parameter`, men ofte tar man ikke inn noen `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 `funksjon` og `metode` litt om hverandre.

- [ ] Ta nå koden som har med alder å gjøre, og legg den inn i en funksjon. Kall funksjonen for `sjekkAlder`

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 lage en funksjon som skal ta inn et `paramenter` og `returnere` en verdi.

Vi lager en funksjon som konverterer `fahrenheit` (temperatur mål i USA) til `Celsius` .

- [ ] Slett det du har i JSBin eller åpne et nytt vindu i JSBin (File -> New

- [ ] Lag følgende funksjon:

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

Forklaring

- `fahrenheitTilCelsius(fahrenheit)` betyr at vi tar inn en variabel som heter `fahrenheit` . Den kan kun brukes innenfor `{ }` i funksjonen vår.

- `return (5/9) * (fahrenheit - 32)` bruker variabelen `fahrenheit` og regner sammen mattestykket som står der.

- Men hva betyr `return` ?

- Return betyr at funksjonen returnerer en verdi til brukeren som brukeren kan se, lagre eller bruke videre. La oss se på et eksempel:

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

- Koden over kjører funksjonen `fahrenheitTilCelsius` med paramenter `77` . Funksjonen regner da ut hvor mange `celsius` 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 bevege oss over til `løkker`.

Steg 4: Løkker

Vi som programmerer er ganske late og derfor bruker vi verktøy til å gjenta kode. Løkker gjør akkurat dette. Ved hjelp av løkker kan vi gjenta kode istedet for å skrive mange linjer med kode. Så vi skal i denne seksjonen se på to type 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, derfor er det et tall og det starter gjerne på `0`.
- `i < 10;` dette er en betingelse som forteller løkken om den skal fortsette eller av sluttes. Her sier vi at løkken skal kjøre så lenge variabelen `i` er mindre enn 10.
- `i++;` øker variabelen `i` med `1` for hver runde løkken kjører.

Eksempel:

```
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 melding mange ganger eller skrive ut hva du har i handlelisten din:

```
var handleliste = ["melk", "egg", "smør", "toalettpapir", "brus",
"epler", "bananer"]  //[] betyr liste

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

- `[]` betyr liste, og alle elementene som er i listen blir separert med `,`.
- `handleliste.length()` er et tall på hvor stor listen er. Skriv `console.log(handleliste.length);` i JSBin og se hva slags tall du får ut.
- Når vi skal skrive ut et gitt element fra en liste skriver vi `handleliste[index]`. `index` her er hvilken plass i lista elementet har. Skal du skrive ut det første elementet skriver du `handleliste[0]`. `handleliste[1]` er det andre elementet, osv.

Prøv selv!

- [ ] Åpne ny side i JSBi
- [ ] Prøv å skriv ut tallene fra 1 - 100 ved hjelp av en `for` -løkk

- [ ] Bytt på telleren `i++` slik at alle partall mellom 1 og 100 skrives ut

Hint

- [ ] Klarer du å skrive ut alle oddetallene også?

Bra! La oss se på lister.

- [ ] Lag nå en liste over dine favoritt spil

- [ ] Skriv ut alle ved hjelp av en `for` -løkk

- [ ] Skriv ut annen hvert element i lista (hint: bruk samme metode som med partall)

While-løkke

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

En `while` -løkke kjører en blokk med kode helt til `betingelsen` blir sann. `While` -løkker er fine å bruke om man for eksempel skal lage spill:

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

`!` betyr `not` , altså betyr `!gameOver` at løkken kjører så lenge spillet `ikke er slutt` . Dersom vi vil at en løkke skal 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 definere (`var i = 0;`) utenfor selve løkken. Selve telleren (`i++`) legger vi etter all koden vi har løkken sånn at den teller opp etter vi har utført det vi skal.

Prøv selv

- [ ] Skriv om `while` -løkken til å skrive ut alle tallene fra 1-10
- [ ] Skriv om løkken slik at du skriver ut alle partallene fra 1-100
- [ ] Bruk listen din over favorittspill og skriv ut alle elementene ved hjelp av `while` -løkken

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

- [ ] Åpne en ny [JSbin.com](https://jsbin.com/?js,console
- [ ] Lag en variabel, `krone` , som skal være enten 0 eller 1, dette skal være tilfeldig:

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

`Math.floor()` runder ned tallet du får fra `Math.random()` . `Math.random()` henter et tall mellom 0 og 1, derfor bruker vi `*2` for at tallet blir mellom 0-2.

- [ ] Lag en `while` -løkke som skal kjøre helt til du får `mynt` . Vi sier nå at `kron` er `1` og `mynt` er `0` :

```
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 kan bestemme om du vil ha `Kron` eller `Mynt`

- [ ] Klarer du å gjøre det samme med terninger? Da må du ha tallene fra 0-6 og ikke bare fra 0-2.