</> Web

JS: Grunnleggjande JavaScript

Skrevet av: Lars Klingenberg

Oversatt av: Stein Olav Romslo

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

Introduksjon

I denne oppgåva skal du lære heilt enkle og grunnleggjande element av `JavaScript`. Du vil lære om `variablar`, `if-setningar`, `funksjonar` og `løkker`.

For å gjere oppgåva treng du ikkje forkunnskapar i HTML eller CSS, men det er veldig lurt å kunne HTML og CSS når du skal programmere JavaScript.

Viss HTML er ein bil, så er CSS utsjånaden og designet på bilen, medan JavaScript står for funksjonaliteten, til dømes å skru på motoren og å få bilen til å køyre. Viss du har programmert i eit tekst-programmeringsspråk før, til dømes `Python`, så vil nok mykje i denne oppgåva vere kjent.

La oss starte!

Steg 1: Variablar

Viss du gjorde oppgåva Hei, JavaScript (../hei_js/hei_js_nn.html), så er nok variablar kjent. Det same gjeld viss du har programmert noko frå før. Men litt repetisjon skadar ikkje!

Ein variabel i JavaScript ser slik ut:

```
var variabelNamn = "verdi";
```

`var` fortel JavaScript at det er ein variabel. Ein variabel inneheldt ein verdi, til dømes `tal` eller `tekst`:

```
var tal = 9;
var tekst = "Dette er ein tekst";
```

La oss prøve oss fram litt!

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

- [ ] I `JavaScript` -vindauget skriv du følgjande:

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

Forklaring

Hugs at kvar linje i JavaScript skal avsluttast med anten `;` eller `}`, avhengig av om du lagar ein variabel eller ein funksjon. Når me brukar variablar avsluttar me med `;`, med funksjonar avsluttar me med `}`.

For at me skal kunne skrive ut noko til `Console` brukar me `console.log()`:

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

- [ ] Trykk på `Run`.

- [ ] Ser du teksten?

- [ ] La oss prøve oss på litt variabel-moro. Lag følgjande variablar, du kan godt slette det du allereie har skrive:

```
var tal1 = 4;
var tal2 = 7;
```

- [ ] No skal me ta dei to variablane og leggje dei saman:

```
console.log(tal1 + tal2);
```

- [ ] Trykk `Run` . Fekk du 11?

- [ ] No som me veit at me kan få JavaScript til å reikne for oss, prøv å byte ut `+` med dei andre rekneartane me har, og sjå om JavaScript klarar å rekne med dei.

No skal me sjå nærare på korleis me kan la ein variabel vere ein annan:Q

- [ ] Legg til endå ein variabel:

```
var tal3 = tal2;
```

- [ ] Kva blir `console.log(tal3 * tal2)` ?

Svar

- [ ] Skriv ut `tal3` og `tal2` til `console` .

- [ ] Prøv å endre på `tal2` , kva skjer med `tal3` ?

Me sette jo `tal3` til å vere lik `tal2` ? Men `tal3` blir ikkje endra sjølv om me endrar på `tal2` . Det er fordi tal-variablar i JavaScript berre inneheldt verdiar, ikkje referansar.

Me kan òg leggje til ekstra verdiar til ein variabel:

```
var tal4 = tal3 + tal1 + 5;
```

- [ ] Bruk `console.log` til å skrive ut `tal4` .

No har me prøvd ut litt ulike variablar. Vidare skal me bruke dei saman med `if-setningar` for å sjekke om noko er sant eller ikkje.

Steg 2: If-setningar

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

```
if(vilkår) {
    // Køyr koden som blir skrive her
} else {
    // Køyr koden som blir skrive her i staden
}
```

Når me brukar `if/else` sjekkar me eit vilkår, og basert på om vilkåret er sant eller usant øyrer me ein gitt kode. La oss sjå på eit døme med tal.

- [ ] Skriv dette i JSBin:

```
var tal = 5;

if(tal === 5) {
    console.log("Talet er 5");
} else {
    console.log("Talet er ikkje 5");
}
```

Forklaring

Viss `tal` har verdien `5` vil den fyrste meldinga bli skrive ut. Viss `tal` har ein annan verdi, så er det den andre meldinga som blir skrive ut. For å sjekke om ein variabel er lik noko brukar me `===` . Då sjekkar JavaScript om verdien og datatypen (nummer, tekst osb.) er like.

- [ ] Endre variabelen `tal` til andre verdiar og sjå kva melding du får ut.

- [ ] La oss lage ein sjekk på om du kan ta sertifikatet til bil eller moped:

```
var alder = 0;

if(alder >= 18) {
    console.log("Du er gamal nok til å køyre bil");
} else if(alder >= 16) {
    console.log("Du er gamal nok til å køyre moped");
} else {
    console.log("Du er diverre ikkje gamal nok");
}
```

Forklaring

- `if(alder >= 18)` tyder: viss alder er større enn eller lik 18. Altså er setninga sann viss du er 18 år eller eldre, og då er du gamal nok til å køyre bil.
- `else if(alder >= 16)` tyder: viss `if` -testen var usann, så sjekkar den om alder er større enn eller lik 16. Viss dette vilkåret er sant, så seier den at du er gamal nok til å køyre moped.
- `else` tyder `elles` og vil seie at koden til denne blir køyrt viss dei andre testane er usanne. Altså vil du få beskjed om at du ikkje er gamal nok viss du er under 16 år.

- [ ] Prøv å endre `alder` slik at du får testa om `if/else` -setningane fungerer.
- [ ] No vil me leggje til noko som heiter `prompt` . Dette gjer at du kan få `input` frå brukaren av nettsida:

```
var alder = prompt("Kor gamal er du?");
```

No skal me gå eit steg vidare. Me skal sjekke kva klokka er og skrive ei melding ut frå det. Då brukar me noko som heiter `Date` i JavaScript. Denne inneheldt informasjon om dagen i dag. Dette kallast ei klasse, men det treng du ikkje tenke meir på akkurat no.

- [ ] Me lagar to variablar. Ein er ei `Date`-klasse og ein annan hentar tida. Me nøyer oss med kva time me er i:

```
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 kva variabelen `tid` inneheldt.

Før du skal få ei ny oppgåve må me gå gjennom nokre verktøy me kan bruke i `if/else`:

Forklaring

- I ein `if(vilkår)` kan me sjekke fleire ting samstundes ved å bruke `&&` eller `||`. `&&` tyder `og`, `||` tyder `eller`. Finn du ikkje desse teikna på tastaturen kan du spørje ein vaksen om å hjelpe deg. Døme:

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

if(namn === "Lars" && mnd === 7 && dato === 10) { // mnd = månad,
 dato = dagen i månaden

    console.log("Gratulerer med namnedagen, Lars!");
}
```

For at koden skal vere sann må namnet vere `Lars` og datoen må vere `10.8`, altså 10. august. `getMonth()` leverer ut eit tal frå 0-11, difor er August 7 og ikkje 8. Les meir om `Date` her (http://www.w3schools.com/jsref/jsref_obj_date.asp).

Det same kan du gjere med `||` viss du heller vil sjekke `eller`:

```
if(mnd === 7 && dato === 10) {
    if(namn === "Lars" || namn === "Lasse") {
        console.log("Gratulerer med namnedagen!");
    } else {
        console.log("Du har diverre ikkje namnedag i dag.");
    }
}
```

```
}
```

Legg merke til her at viss datoen er `10.8` så går du vidare til ein ny `if/else` som sjekkar om namnet ditt er anten `Lars` eller `Lasse`. Viss det er feil dato skjer det ingenting.

No håpar me at du har fått dreisen på `if/else`. Viss du vil kan du øve meir, men me kjem til å jobbe meir med dette seinare. Innanfor `if/else` er det mykje å utforske, så det tek litt tid og krev erfaring å bli vane med det.

No skal me sjå på `funksjonar`!

Steg 3: Funksjonar

Til no har me berre skrive linjer med kode i `JSBin`. Nor koden blir køyrt blir den lest frå topp til botn, linje for linje, så koden blir køyrt i den rekkefølgja den står i. Men nokre gonger ynskjer me at koden skal køyre når ei hending inntreff eller noko anna skjer. Ved hjelp av `funksjonar` kan me sjølv bestemme når koden skal køyre, eller om den ikkje skal køyre i det heile.

Oppsettet ser slik ut:

```
function funksjonsNamn(parameter1, parameter2) {
    // Kode som utførast når funksjonen blir kalla
}

funksjonsNamn(parameter1, parameter2); // Gjer at funksjonen blir køyrt
```

Ein funksjon tek nokre gonger inn variablar den skal ta i bruk ved hjelp av `parameter`, men ofte treng ein ikkje å ta inn noko `parameter`. Då ser funksjonen slik ut:

```
function namn() {
    // Kode
}

namn(); //for å køyre funksjonen
```

Ein funksjon kallast ofte for ein `metode`. Vidare kjem me til å bruke `funksjon` og `metode` litt om kvarandre.

- [ ] Kopier koden som har med alder å gjere, og legg den inn i ein funksjon. Kall funksjonen `sjekkAlder`

Hint

- [ ] Legg på kode for at `sjekkAlder` skal køyre.

Hint

No skal me sjå eit anna døme på bruk av funksjonar. Me skal lage ein funksjon som tek inn ein `parameter` og som skal `returnere` ein verdi.

Me lagar ein funksjon som konverterer `Fahrenheit` (temperaturskalaen dei brukar i USA) til `Celsius` .

- [ ] Slett det du har i JSBin eller åpne eit nytt vindauge i JSBin (File -> New).

- [ ] Lag følgjande funksjon:

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

Forklaring

- `fahrenheitTilCelsius(fahrenheit)` tyder at me tek inn ein variabel som me kallar `fahrenheit` . Den kan berre bli brukt innanfor `{ }` i funksjonen vår.

- `return (5/9) * (fahrenheit - 32)` brukar variabelen `fahrenheit` og reknar ut reknestykket som står der.

- Men kva tyder `return` ?

- Return tyder at funksjonen returnerer ein verdi til brukaren som brukaren kan sjå, lagre eller bruke vidare. La osss sjå på eit anna døme:

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

- Koden over køyrer funksjonen `fahrenheitTilCelsius` med parameter `77`. Då reknar funksjonen ut kor mange gradar `Celsius` 77 Fahrenheit er, og lagrar denne verdien i variabelen `gradar`.

No som me har fått kontroll på kva ein `funksjon` eller `metode` er skal me bevege oss over til `løkker`.

Steg 4: Løkker

Me som programmerer er ganske late, og difor brukar me verktøy til å gjenta kode. Lækker gjer akkurat det. Ved hjelp av løkker kan me gjenta kode i staden for å skrive koden mange gonger. Så i denne delen skal me sjå på to typer løkker: `for`-løkker og `while`-løkker.

For-løkke

```
for(var i = 0; i < 10; i++) {
    // Kode som køyrast eit gitt antal gonger
}
```

- `var i = 0;` dette er ein variabel som fungerer som ein teljar, difor er det eit tal, og me startar vanlegvis med `0` når me tel.
- `i < 10;` dette er eit vilkår som fortel løkka om den skal fortsetje eller avsluttast. Her seier me at løkka skal køyre så lenge variabelen `i` er mindre enn 10.
- `i++;` aukar variabelen `i` med `1` for kvar runde løkka køyrer.

Døme:

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

Dette blir skrive ut i konsollen:

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

Ein kan gjere det same for å til dømes skrive ut ei melding mange gonger, eller skrive ut kva du har i handlelista di:

```
var handleliste = ["mjølk", "egg", "smør", "toalettpapir", "brus",
"eple", "bananar"]  //[] tyder liste

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

- `[]` tyder liste, og alle elementa som er i lista blir separert med `,`.
- `handleliste.length()` seier kor stor lista er. Skriv `console.log(handleliste.length);` i JSBin og sjå kva tal du får ut.
- Når me skal skrive ut eit gitt element frå ei liste skriv me `handleliste[index]`. `index` er kva plass i lista elementet har. Hugs at me startar å telje på 0, så det fyrste elementet er `handleliste[0]`. Då blir `handleliste[1]` det andre elementet, osb.

Prøv sjølv!

- [ ] Åpne ei ny side i JSBin.
- [ ] Prøv å skrive ut tala frå 1 til 100 ved hjelp av ei `for`-løkke.

- [ ] Byt på teljaren `i++` slik at alle partal mellom 1 og 100 blir skrive ut

Hint

- [ ] Klarar du å skrive ut alle oddetala i staden?

Bra! La oss sjå på lister.

- [ ] Lag ei liste over alle favorittspela dine.

- [ ] Skriv ut alle ved hjelp av ei `for` -løkke.

- [ ] Skriv ut annakvart element i lista. **Hint:** bruk same metode som med partala.

While-løkke

```
while(vilkår) {
    // Køyre kode til vilkåret er usant
}
```

Ei `while` -løkke køyrer ei blokk med kode heilt til `vilkåret` blir usant. `While` -løkker er fine å bruke, til dømes når ein skal lage eit spel:

```
while(!gameOver) {
    // Køyr spel
}
```

`!` tyder `not` , altså tyder `!gameOver` at løkka køyrer så lenge spelet `ikkje er slutt` . Viss me vil at ei løkke skal køyre for alltid kan me setje vilkåret til `True` .

Me kan gjere det same som me gjorde med `for` -løkka:

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

Skilnaden er at her må me lagje ein teljande variabel som me definerer utanfor sjølve løkka (`var i = 0;`). Sjølve teljaren legg me etter all koden i løkka, slik at den tel opp etter me har utført det me skal.

Prøv sjølv

- [ ] Skriv om `while` -løkka til å skrive ut alle tala frå 1 til 10.
- [ ] Skriv om løkka slik at du skriv ut alle partalla frå 1 til 100.
- [ ] Bruk lista di over favorittspela dine og skriv ut alle elementa ved hjelp av `while` -løkka.

No skal me lage eit enkelt `kron eller mynt` -spel ved hjelp av `while` .

- [ ] Åpne ei ny fane med JSbin.com (https://jsbin.com/?js,console).
- [ ] Lag ein variabel, `krone` , som skal vere anten 0 eller 1, dette skal vere tilfeldig:

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

`Math.floor()` rundar ned talet du får frå `Math.random()` . `Math.random()` hentar eit tal mellom 0 og 1, difor brukar me `*2` for at talet skal vere mellom 0 og 2.

- [ ] Lag ei `while` -løkke som skal køyre heilt til du får `mynt` . Me vil at `1` tyder `kron` og `0` tyder `mynt` .

```
while(krone === 0) {
    console.log("Kron! Me flippar ein gong til...");
    var krone = Math.floor(Math.random()*2);
}
console.log("Mynt! Gratulerer!");
```

Bra jobba! No har du lært mykje om JavaScript!

Utfordring

- [ ] Bruk `prompt` til å ta inn eit val frå brukaren, slik at du sjølv kan bestemme om du vil ha `kron` eller `mynt`

- [ ] Klarar du å gjere det same med terningar? Då må du ha tala frå 0 til 6, ikkje berre 0-2.