NTNU

Versjon: 2.1
Dato: 15. mai 2020

IT6204 – Anvendt programmering for lærere

Prosjektoppgave:
Undervisningsopplegg & progresjonsplan

Utført av:

Sune Magns Fjell

Tid og sted:

10/05-2020 Gjettum

Innhold

Undervisningsopplegg og progresjonsplan:

Det foregår undervisning i 2 økter i uken, begge øktene har 3 ganger 45 minutters økter.

Økt 1: Introduksjon til oppgaven

Undervisning:

I den første økten går jeg gjennom selve oppgaven. Hvilke krav som stilles til elevene, hva som er oppgaven, og hvilke utfordringer de vil møte på veien. Dette ligger under delen som heter prosjektbeskrivelse i oppgaven. Jeg ønsker å repetere en del sentrale kunnkaper de har lært tidligere gjennom skoleåret med IT1, og sentrale begreper de har lært gjennom å kode i HTML og CSS. Dette ligger under overskriften Forkunnskaper i selve prosjektoppgaven. Det gå gjennom spillets regeler. Til slutt ønsker jeg å vise progresjonsplanen på min undervisning de kommende ukene.

Elevarbeid:

I denne første økten er det viktig at elevene får frisket opp litt eldre kunnskaper og kommer i gang med prosjektoppgaven. De må velge seg selv hvilke medelever de ønsker å samarbeide med, og hvordan de fordeler abeidsmengden, slik at de kan gjøre det de selv synes er mest morsomt og spennende å jobbe med.

Økt 2: Lage rammeverket

Hvordan sette opp spillets grunnleggende variabler:

For å lage spillet må vi definere enkelte globale variabler som er nødvendig i spillet.
Det viktigste av alt poengsummen for hver spiller. Vi kan lage en variabel for hver spiller som vi setter til verdien 0. Fremfor å gjøre dette ønsker jeg å utfordre elevene til å ta i bruk arrays til dette formålet. Vi lager derfor en array som helter *poengene* med to verdier som begge settes til 0. Vi trenger også en variabel til å ta vare på poengene som hver spiller oppnår i hver runde, denne kaller vi for *rundePoeng*.

Vi trenger også en variabel som vil fortelle oss hvilken som er den aktive aktive spilleren i hver runde. La oss kalle denne variabelen for *aktivSpiller*, og sette verdien til denne variabelen til 0. Tallet 0 tilsvarer den første spilleren og tallet 1 er spiller number 2. Dette gjør det enkelt senere å hente riktig verdi ut av av arrayen over poengene fra den aktive spilleren, ved å ta i bruk verdien i *aktivSpiller* som indexnummeret i arrayen.

Koden i begynnelsen vil se slik ut:

```
var poengsum, rundePoeng, aktivSpiller;
poengsum = [0, 0];
aktivSpiller = 0;
rundePoeng = 0;
```

Hvordan lager vi tilfeldige tall:

For å kunne kaste terninger i spillet vårt må koden kunne generere tilfedige tall. Vi må derfor først konstruere en variabel som inneholder et tilfeldig heltall mellom 1 og 6. Vi må først lage metoden for å få et tilfeldig tall, som vi får til ved å bruke *Math.random()*. Vi må først gange

dette med 6, noe som gir verdier mellom 0 og 5 med desimaler. For å fjerne desimalene må vi legge hele denne metoden inn i Math.floor() funksjonen. Da vil vi få heltall mellom 0 og 5.

For å få riktige terningkast mellom 1 og 6, må vi til slutt addere tallet 1 til slutt. Da vil terninggeneratoren fungere slik den skal.
Hele koden for dette ser slik ut:

```
var terning = Math.floor(Math.random() * 6) + 1;
```

Elevene kan selv teste terningverdiene ved å skrive `console.log(terning);` inn på neste linje i koden. Dette ved å starte opp programmet i Google Chrome, og åpne konsollen. Hver gang de trykker refresh i nettleseren vi variabelen *terning* få en ny verdi mellom 1 og 6.

Skrive til DOM (Document Object Model):

For å lage kode som kommunserer med HTML dokumetet må vi bruke DOM (Document Object Model) og ta i bruk en queryselektor. Syntaksen ser slik ut: `document.querySelector`. Måten å velge på er ganske lik som du bruker selektorer i CSS kode. Vi kan endre innholdet på nettsiden vår ved å referere til en Id i HTML koden, og så lese innholdet i fra dokumetet ved å skrive `.textContent`.

Nå skal vi forsøke å hente det tilfeldige terningkastet, og sette det inn i nettsiden vår. Hver spiller har hver sin side av GUI på nettsiden. På linje 19 i HTML koden finner du en Id for feltet som inneholder *sum* verdien. Spiller 1 har en Id som heter sum-0, og på linje 28 ser du at spiller to har Id sum-1.

Vi har fra før definert variabelen *aktivSpiller*, og satt denne verdien til 0. Dette kan vi utnytte til å lage mer effektiv kode, som lettere kan gjenbrukes ved å bare endre verdien til *aktivSpiller* mellom 0 og 1, avhengig av hvilken spiller sin tur det er til å kaste terning. Da snakker programmet vårt med riktig id avhenging av hvilke spiller spiller som aktivt kaster terninger.
Koden for å oppnå dette ser slik ut:

```
document.querySelector('#sum-' + aktivSpiller).textContent =
terning;
```

Lese fra DOM (Document Object Model):

Tilsvarende metode kan brukes til å lese innholdet fra nettsiden. Dersom vi ønsker å hente en verdi fra HTML dokumentet kan det være lurt å legge denne verdien inn i en variabel. Om vi for eksempel vil hente verdien som står på nettsiden når den lastes inn første gang kan vi skrive:

```
var poengsum = document.querySelector('#sum-' +
aktivSpiller).textContent;
```

Dette kan du teste ut i konsollen ved å skrive `console.log(poengsum);`

Modifisering av CSS stiler:

Vi ønsker å skjule terningen når spillet starter, og la den dukke opp først etter at spillet begynner. Dette kan vi oppnå ved å manipulere CSS filen ved å bruke *style* metoden, og ved å sette *display* til *none*.

Koden for dette er ganske så lik som i ekempelet ovenfor:
```
document.querySelector('#sum-' + aktivSpiller).textContent =
terning;
```

Vi kan også bruke innerHTML for å oppnå det samme, men da er det viktig å legge inn HTML elementene med anførselstegn, og bruke + for å veksle mellom HTML elementer og variabler i koden vår. Dette vil se slik ut i vår kode:
```
document.querySelector('#sum-' + aktivSpiller).innerHTML = '<em>' +
terning + '</em>';
```

Vi får da samme resultat, men teksten vil nå dukke opp som kursiv på nettsiden. Fordelen er her at ikke bare teksten kan vises, men vi kan manupulerer ved hjelp av HTML.

Økt 3: Begivenheter

Før vi kan gå videre med å programmere spillet vårt, er det nødvendig å friske opp kunnskapen om begivenheter eller events i JavaScript. Events er en metode for hvordan koden kommuniserer med brukergenererte begivenheter, og hvordan koden responderer på disse begivenhetene. For eksempel når brukeren av nettsiden klikker med musen eller beveger seg over et bestemt element. Koden trenger også en såkalt *event listener*, som er en funksjon som utfører en bestemt handling, når en spesifikk begivenhet skjer.

Lage en metode for å få inn begivenheter:

Den første begivenheten vi skal jobbe med er hvordan vi kaster terningen. Vi må først finne ut hvilken id eller klasse dette har i HTML dokumentet vårt. I HTML filen finner du en knapp med klassen *btn-kast*. Denne knappen skal vi nå velge i koden vår. Dette gjøres ved hjelp av samme metode som før.

Vi må også legge til en *event listener* til det valgte elementet. En event listener skal ha to argumenter. Først må vi oppgi hva slagt type begivenhet som skal skje. Vi ønkser i vår kode at noe skal skje når brukeren klikker på knappen, derfor oppgir vi *click* som vårt første argument. Det andre argumentet vi må oppgi er navnet på funksjonen vi skal kalle opp, når vi klikker på knappen. Denne funksjonen skal vi lage senere i prosjektet, så la oss inntil videre kalle denne for *kastTerning*. Koden for dette ser slik ut:
```
document.querySelector('.btn-kast').addEventListener('click',
kastTerning);
```

Callback funksjon:

Når vi ønsker å kalle opp en funksjon ved hjelp av en begivenhet trenger vi ikke lage en egen funksjon for dette, men vi kan lage selve funksjonen direkte inn i vår event listener. Denne koden egner seg ikke for gjenbruk andre steder i koden, men er direkte knyttet til begiveheten som skal skje, når noen kilkker på knappen. Dette kalles en callback funksjon.

Anonyme funksjoner:

Vi kan også bruke en anonym funksjon, som er en funksjon som sitter i en bit med kode, som heller ikke kan brukes andre steder enn i akkurat der formålet er.

Programmet kan nå modifiseres, slik at alt som skal skje når brukeren kaster en terning skjer kun lokalt inne i vår event listener. Koden ser nå slik ut:

```
document.querySelector('.btn-kast').addEventListener('click',
function(){
      //Her kommer koden for hva funksjonen skal utføre
});
```

Metoden for å kaste en tilfeldig terning trenger ikke stå globalt i koden vår, så den kan vi klippe ut og lime inn i vår nye anonyme funksjon. Funksjonen må også kunne vise oss resultatet av den kastende terningen. Sist men ikke minst må vi oppdatere variabelen rundePoeng, bare dersom det tilfeldige tallet vi fikk på terningen er høyere enn 1.

Vi strukturer funskjonen i vår event listener slik:

1. Tilfeldig nummer
2. Vis resultatet
3. Oppdatere rundePoeng bare hvis kastet terning er mellom 2 og 6, og IKKE 1.

Først klipper vi ut og limer inn metoden for å kaste en tilfeldig terning. Vi må deretter kunne vise hva slags type terning vi har kastet, noe som gjøres ved å igjen endre på CSS stilen ved å sette *display* tilbake til *block,* slik det står skrevet i CSS filen.

Endre et bilde:

Vi kan endre bilde som dukker på nettsiden ved å bruker samme selektor som ovenfor, men fremfor å endre stilen skriver vi i stedenfor .src, da kan vi enkelt endre bildefilen som brukes. Det er en smart strategi å gi bildefilene navn etter variabelen vi bruker, og legge på nummeret på terningen som skal vises. Tekst må stå med anførselstegn og slår sammen med variableverdien ved hjelp av + tegnet. Husk også filendelsen til slutt.

Vi skriver følgende linje i koden vår:

```
document.querySelector('.terning').src = 'terning-' + terning +
'.png';
```

Flere måter å selektere elementer:

Vi har frem til nå brukt querySelector for å selektere elementer fra HTML filen vår. Vi kan også bruke *getElementById* for å selektere en id selektor direkte.
Vi må uansett sette alle poengene til tallet 0 på nettsiden vår, når den startes første gang. Vi bruker denne gangen *document.getElementById* til å sette alle verdiene til 0.

Finn selv ut hvilke id i HTML dokumentet som inneholder de aktive poengene til hver spiller, og samtidig id navnet på den totale summen til hver spiller. Lag koden for alle fire ved å bruke følgende kode:

```
document.getElementById(idnavn').textContent = '0';
```

Økt 4: Lagre poengene og bytte mellom spillerne

Legge til poeng:

Spillet vårt trenger nå en metode for å legge til poengene den aktive spilleren får, når terningen blir kastet. Dette skal bare skje dersom terningen som blir kastet er høyere enn 1. Vi

kan skrive `if(terning > 1 {gjør noe}`, men det er bedre å skrive om terningen *ikke* er talllet 1. Dette gjør vi ved å bytte ut > tegnet med `!==` i JavaScript.

For å legge til poengene fra den kastede terningen skriver vi på neste linje i koden `rundePoeng = rundePoeng + terning;` En mye bedre metode i JavaScript er å bruke += operatoren, da slipper vi å gjenta variabelen *rundePoeng*. Vi skriver derfor `rundePoeng += terning;`

Nå kan vi oppdaterer poengene på nettsiden, noe vi enkelt kan gjøre ved å bruke et triks vi så på tidligere. Vi skriver bare `document.querySelector('#sum-' + aktivSpiller).textContent = rundePoeng;` Da vil vi legge den oppdaterte variabelen *rundePoeng* inn i klassen *sum*, som tilhører den aktive spilleren.

Nå er det på tide å teste ut koden for å se om alt virker så langt. Hver gang du trykker på kast terning, så skal du få en tilfeldig terning, og poengene du får skal bli lagt til og oppdatert i feltet for poengsummene for den aktive spilleren. Det vi si spiller 1, siden vi enda ikke har lagd metoden for å endre spilleren basert på spillets regler. Test ut at dette bare skjer når du kaster 2 eller høyere, for å se om if setningen virker som den skal.

Ternæroperator:

Foreløpig fungerer ikke spillet helt, siden det bare legger til poeng for spiller 1. Vi trenger nå en metode for å bytte mellom spillerene, dersom den aktive spilleren kaster 1 på terningen. Nå har vi sett på en if setningen i praksis, og vi kan selvfølgelig bruke denne på nytt for å bytte spilleren, men dette er en tungvint måte å kode på for helt enkle ting. I stedenfor kan vi bruke noe som kalles ternæroperator (på engelsk *ternary operator*).

En ternæroperator er det samme som en forenklet if/else setning, men den skrives på en annen måte. Formulaen for denne ser slik ut: *`betingelse ? verdi hvis sant : verdi hvis usant`*. I vårt tilfelle er betingelsen om variabelen *aktivSpiller* har verdien 0. I JavaScript må vi bruke tre = tegn for å sammenligne verdier. Betingelsen vår ser da slik ut `aktivSpiller === 0`. Hvis dette er sant skal aktivSpiller settes til verdien 1, og verdi hvis usant (det vil si at aktiv spiller er 1) skal settes til 0. På denne måten bytter vi enkelt mellom aktive spillere. Forsøk nå å skrive denne setningen ved hjelp av ternæroperatorer.

Nullstille poeng:

Vi må også nullstille poengene, slik at neste spiller ikke får poengene som forrige spiller har opparbeidet seg i spillet. Dette gjøres enkelt ved å skrive `rundePoeng = 0;` på neste linje i koden. Problemet er nå at poengene blir nullstilt hver gang det er neste spiller sin tur. Det hjelper ikke bare nullstille poengene i koden, dette må også gjøres i brukergrensesnittet på nettsiden. Dette gjør du ved å lage selektor som peker til riktig id, der summen skal dukke opp og så endre dette tekstinnholdet. Dette må gjøres hos begge spillerne. Husker du hvordan du gjør det? Hint: `document.getElementById('id navn').textContent = '0';`

Oppdatere brukergrensesnittet for å vise aktiv spiller:

Om du ser etter i brukergrensesnittet vises det hvem som er aktiv spiller på flere måter. Dette er gjort i CSS ved å bruke klassen *aktiv*. Denne klassen setter navnet på spiller til fet tekst, en grønn prikk vises etter spillernavnet og bakgrunnen på hele panelet er grått. I HTML koden på

linje 15 ser du at spiller 1 har klassen aktiv. Denne klassen har ikke spiller 2, dette fordi spillet begynner alltid med at det er spiller 1 sin tur.

Det vi må gjøre nå er å bytte mellom hvilken spiller som er den aktive spilleren i brukergrensesnittet vårt. Dette kan vi gjøre ved å bruke classList egenskapen i JavaScript. ClassList returnerer klassens navn på et element. Denne egenskapen er nyttig å legge til, fjerne og veksle CSS-klasser på et element.

I koden kan vi skrive følgende for å fjerne klassen *aktiv* fra spiller 1, og for å legge til samme klasse til spiller 2:

```
document.querySelector('.spiller-0-
panel').classList.remove('aktiv');
document.querySelector('.spiller-1-panel').classList.add('aktiv');
```

Dersom vi tester programmet i nettleser ser vi nå at dersom spiller 1 kaster 1 på terningen, byttes aktiv spiller til spiller 2. Problemet er nå at det byttes ikke tilbake til spiller 1, når spiller 2 kaster 1 på din terning. For å løse dette kan vi bruke egenskapen *toggle*, noe som gjør at vi kan bytte frem og tilbake hvem som er aktiv spiller. Toggle fungerer som en av og på bryter. Forsøk dette selv ved å endre koden ovenfor ved å bruke *toggle* i stedenfor *remove* og *add* og test det ut i nettleseren.

Økt 5: Programmere hold funksjonen:

Det neste trinnet vi må gjøre for å utvikle spillet i henhold til reglene er å programmere knappen som skal gjøre det mulig for spillerne å boholde poengene sine. For å få til dette må vi først lage en event listener som venter på at spilleren klikker på hold knappen. På linje 36 i HTML dokumtentet finner du en klasse som heter *btn-hold*, det er på denne vi må sette opp vår event listener.

```
document.querySelector('.btn-hold').addEventListener('click',
function(){koden som skal utføres});
```

Koden som skal utføres i denne funksjonen er følgende:

1. Legg til poengsum til den globale variabelen *poengsum*
2. Oppdatere brukergrensesnittet
3. Sjekk om den aktive spilleren har vunnet spillet

Legg til poengsum til en global array:

Variabelen *poengsum* er en array, det vil si en variabel som kan innholde flere verdier. Siden variablen *aktivSpiller* enten har verdien 0 eller 1, kan vi bruke dette når vi skal lagre poengene i den globale arrayen *poengsum*. Dette ved å bruke *aktivSpiller* sim index i arrayen. Dette gjør vi slik:

```
poengsum[aktivSpiller] += rundePoeng;
```

Vi legger da inn poengene som ble oppsamlet i den aktive runden sammen med poengene som spilleren hadde oppsamlet fra tidligere runder i spilet.

Oppdatere brukergrensesnittet:

Før vi kan bygge ut denne funsjonen må vi bestemme oss for hva vi ønsker å oppdatere. I HTML dokumentet på linje 17 finner du en id med navn *poeng-0*, som skal inneholde

poengene til spiller 1. Spiller 2 har tilsvarende en id med navn *poeng-1*. Funksjonen vår må referere til korrekt id for at programmet skal oppdatere riktig element i HTML filen.
Vi har tidligere gjort mye av det samme, og koden her er veldig lik:

```
document.getElementById('poeng-' + aktivSpiller).textContent =
poengsum[aktivSpiller];
```

Test ut om dette gir ønsket resultat i nettleseren din, og om alt virker som det skal.

Bruk av funksjoner for å effektivisere koden:

Dersom du har testet ut spillet i nettleser, vil du oppdage at hold knappen frem til nå ikke har innebygd en funksjon som gjør neste spiller aktiv. Når vi konstruerte knappen for å kaste terning lagde vi nettopp denne funksjonaliteten i programmet. Vi kan enkelt kopiere denne koden og lime den inn i koden etter at vi oppdaterer brukergrensesnittet med riktig poengsum. Problemet med denne løsningen er at vi gjentar akkurat det samme, og dersom vi senere skal gjøre endringer så må vi gjøre nøyaktig samme endringen to ulike steder i koden vår.

Dersom vi følger reglene for god programmering, så bør vi ikke gjenta noe i koden, både for å effektivisere programmet og for å skape større grad av logikk. Løsningen på dette er å lage en egen funksjon som vi kaller opp når det er behov for at dette utføres. Vi lager derfor en ny funksjon som vi gir navnet *nesteSpiller*. Alt som er skrevet inn i *else* utførelsen i forrige if setning kan vi kopiere inn i vår nye funksjon, og heller kalle opp denne funksjonen når vi trenger å bruke den. Funksjonen skriver vi nederst i koden vår, og den vil se slik ut:

```
function nesteSpiller(){
    //Neste spiller
    aktivSpiller === 0 ? aktivSpiller = 1 : aktivSpiller = 0;
    rundePoeng = 0;
    //Oppdatere GUI
    document.getElementById('sum-0').textContent = '0';
    document.getElementById('sum-1').textContent = '0';
    //Endre på aktiv spiller i CSS
    document.querySelector('.spiller-0-
panel').classList.toggle('aktiv');
    document.querySelector('.spiller-1-
panel').classList.toggle('aktiv');
     }
```

Vi kaller enkelt opp denne funksjonen når vi trenger det ved å referere til den funksjonsnavn med parentes og semikolon slik: `nesteSpiller();`.

Spillets vinner:

Vi kan nå begynne å programmere vinneren av spillet. Vi må derfor først sjekke om den aktive spilleren har vunnet spillet. Dette kan vi gjøre ved hjelp av en enkel if setning. Vi tester først om den aktive spilleren sin poengsum er lik 100 eller høyere ved å skrive `if (poengsum[aktivSpiller] >= 100{koden som skal utføres};.`

Først ønsker vi å bytte ut spillernavnet med teksten *Vinner!*. Dette gjør vi enkelt på samme måte som tidligere ved hjelp av en selektor og *textContent* attributten. Dette ser slik ut:

```
document.querySelector('#navn-' + aktivSpiller).textContent =
'Vinner!';
```

Deretter ønsker vi å skjule den siste terningen vi kastet, noe vi også har gjort tidligere i oppgaven. Kan du huske hvordan dette gjøres ved å endre CSS stilen i JavaScript?

Nederst i CSS filen ligger det en egen klasse som heter *vinner*. Det vi nå skal gjøre er å aktivere denne klassen på panelet til den spilleren som har vunnet spillet. Dette gjøres ved å først legge til vinner klassen, og så fjerne aktiv klassen fra spilleren som ble vinner av spillet. Slik ser da koden ut:

```
document.querySelector('.spiller-' + aktivSpiller + '-
panel').classList.add('vinner');
document.querySelector('.spiller-' + aktivSpiller + '-
panel').classList.remove('aktiv');
```

Prøv ut spillet i nettleser og test at alt virker som det skal. Bruk gjerne konsollen og console.log(), for å finne feilene som måtte oppstå.

Økt 6: Start spillet

Programmering av knappen som starter et nytt spill:

Det neste trinnet i oppgaven er å få knappen som begynner ett nytt spill til å fungere. Dette gjør vi som tidligere ved å først lage en event listener som referer til riktig knapp i HTML dokumentet vårt. På linje 34 finner du denne knappen, som har tilknyttet klassen *btn-ny*. Vi skriver en ny linje i koden vår som referer til *btn-ny* slik:

```
document.querySelector('.btn-ny').addEventListener('click',
function{koden som skal kjøres});
```

Koden som må utføres er å nullstille spillernes poeng, poengene for hver runde og sette spiller 1 som aktiv spiller. Vi må også skjule terningen og sette alle poeng til 0 på selve nettsiden. Med andre ord det samme som vi gjorde i begynnelsen av programmet.

Funksjonen som begynner spillet:

Vi har lært å bruke funksjoner for å effektivisere koden i programmene våre, og nettopp dette kan vi fint gjøre her også. Vi lager derfor en ny funksjon med navnet *startSpill()*. Denne funksjonen skal ha samme innhold som i begynnelsen av koden vår, så kan vi heller kalle opp denne funksjonen når programmet starter. Forsøk selv å konstruere denne funksjonen, og husk å kalle den opp etter å ha definert de globale variablene (*poengsum*, *rundePoeng*, *aktivSpiller*) helt i starten av koden.

Vi må også huske å kalle opp denne funksjonen når vi klikker på knappen som starter spillet. Da er det ikke nødvendig å bruke parenteser etter funksjonsnavnet, siden dette ville blitt utført umiddelbart etter at programmet starter. Vi ønsker at dette først skal utføres etter at en av spillerne har trykket på knappen. Koden for knappen vil da se slik ut:

```
document.querySelector('.btn-ny').addEventListener('click',
startSpill);
```

Nullstilling av alle verdier:

Vi må også nullstille spillernavnet slik at det ikke står vinner, en navnet på spilleren. Dette gjør vi enkelt ved å legge til to linjer til i funksjonen som begynner spillet. Slik vi har gjort det mange ganger før må vi først selektere HTML elementet vi skal endre på, legge til *.textContent* og til slutt legge inn navnet på spiller 1. Vi må naturlignok gjøre det samme for spiller 2, slik at begge spillerne får riktig navn. Koden vil se slik ut:

```
document.getElementById('navn-0').textContent = 'Spiller 1';
```

```
document.getElementById('navn-1').textContent = 'Spiller 2';
```

Til slutt må vi fjerne vinner klassen i *startSpill()* funksjonen, slik at CSS formateringen blir fjernet igjen, uavhengig av hvem som vant spillet i forrige runde, når vi velger å starte ett nytt spill. Dette gjør vi enkelt slik vi har gjort det flere ganger tidligere i denne koden, om du husker hvordan det gjøres. Et hint er å bruke *.classList.remove* og navnet på klassen som skal fjernes *(vinner)*.

På samme måte som ovenfor må vi nullstille CSS stilen for aktiv klasse, uavhengig av vinneren i forrige spill. Når dette er gjort for begge spillerne, må du huske å reaktive *aktiv* klassen for spiller 1, siden det alltid er denne spilleren som begynner å kaste terning når du starter et nytt spill.

Økt 7: De siste justeringene

Spillet fungere nå slik det skal, men det gjenstår noen mindre justeringer for å gjøre det helt perfekt. Problemet er at det fortsatt er mulig å kaste terninger etter at vi har en vinner, helt frem til vi trykker på Nytt spill knappen. Dette må gjøres bedre for å få et vanntett spill.

Tilstandsvariabel:

Dette kan vi løse ved å bruke en tilstandsvariabel i koden vår. En tilstandsvariabel brukes for å sjekke tilstanden til noe i koden. Denne variabelen bør være globalt tilgjengelig i koden, slik at alle funksjoner kan få tilgang til den. Vi må derfor deklarere den i begynnelsen av programmet. Vi bruker navnet *spill* på denne variabelen, og legger den til i listen sammen med de andre globale variablene i begynnelsen av programmet.

Når *spill* variabelen er deklarert i begynnelsen av koden, kan vi sette boolean verdien *true* som verdi i *spill* variabelen, like etter å ha satt verdiene til *poengsum*, *rundePoeng* og *aktivSpiller* i begynnelsen av funksjonen *startSpill()*.

Variabelen *spill* skal i koden alltid være satt til *true*, bortsett i fra når en av spillerne vinner spillet. Vi trenger en måte å teste dette på, og det er da naturlig å bruke en if setning.

Legg til if setninger til knappenes funksjoner:

Når en av spillerne trykker knappen for å kaste terning, bør vi først teste om spillet er aktivt. Kun om dette er sant skal det være mulig å kaste terning på nytt. Vi må legge denne if testen i den anonyme funksjonene som tilhører vår event listener til knappen *btn-kast*. If testen trenger ikke noen operatorer for å teste siden den bruker boolean, som enten er sant eller usant, og den vil bare kjøre kodeblokken dersom noe er sant. Vi må deretter legge all koden som tilhører denne funksjonen inn i if setningens *kjør denne koden* del. Koden for dette vil se slik ut: `if (spill) { kjør denne koden }`.

For å forenkle dette vil jeg nå vise hele koden for denne knappen slik det ser ut etter denne oppdateringen:

```
document.querySelector('.btn-kast').addEventListener('click',
function(){
    if (spill) {
        //1. Tilfeldig nummer
        var terning = Math.floor(Math.random() * 6) + 1;
```

```
        //2. Vis resultatet
        document.querySelector('.terning').style.display =
'block';
        document.querySelector('.terning').src = 'img/terning-
' + terning + '.png';
        //3. Oppdatere rundePoeng bare hvis kastet terning er
mellom 2 og 6.
        if (terning !== 1) { //Dersom terningen er noe annet
enn 1
            //Legg til poeng
            rundePoeng += terning;
            //Oppdaterer poengene på nettsiden
            document.querySelector('#sum-' +
aktivSpiller).textContent = rundePoeng;
        } else {
            //Neste spiller
            nesteSpiller();

        }
    }

});
```

Avslutning:

Til slutt trenger vi å sette variabelen *spill* til å være usann (false) i programmet vårt. Husk at det eneste stedet vi skal stoppe spillet er når en av spillerne vinner spillet. Vi kan derfor bare legge til en ny linje i denne delen av programmet for å oppnå dette slik; `spill = false;`.

Vi må helt til slutt gjøre samme prosedyre til funksjonen som tilhører knappen *btn-hold*, på samme måte som vi gjorde på knappen *btn-kast*. Det vil si at vi må lage en if setning, for å se om *spill* er satt til verdien *true*. Du må huske på å flytte kodeblokken også denne gangen. Se koden i eksempelet på forrige side for veiledning.

Om du har har fulgt oppskriften og gjort alt riktig bør nå spillet fungere som det skal. Forsøk å spille terningspiller flere ganger i nettleseren din, for å sjekke at alt fungerer som det skal. Om du opplever problemer med spiller, kan du alltids teste ut variablene som er i bruk ved å bruke `console.log(variabelnavn);`. Se også etter feilmeldinger i konsollen til din valgte nettleser. Her vil det stå hvilken linje i koden din hvor feilen oppstår. Forsøk da å rette opp i feilen, og test på nytt i nettleserens konsoll. En annen metode som er praktisk å bruke, kan være å kommentere ut deler av koden, og kjøre mindre deler av koden separat.