

# ▲ JS: Partikkel-fest

*Skrevet av: Lars Klingenberg*

*Kurs: Web*

*Tema: Tekstbasert, Nettside, Animasjon*

*Fag: Matematikk, Programmering, Kunst og håndverk*

*Klassetrinn: 5.-7. klasse, 8.-10. klasse, Videregående skole*

## Introduksjon

Denne oppgaven bygger på koden du skrev i oppgaven Partikkel-animasjon (`../partikkel_animasjon/partikkel_animasjon.html`). Så dersom du ikke har gjort den, så anbefaler vi å gjøre Partikkel-animasjon (`../partikkel_animasjon/partikkel_animasjon.html`) før du fortsetter på denne oppgaven.

Her skal vi videreutvikle `partikkel`-animasjonen vår slik at den ser slik ut:



Merk at i denne oppgaven vil du kun få beskrevet hva du skal gjøre med et par hint. Du vil ikke få presentert den ferdige koden.

# Steg 1: Hva må gjøres?

I denne oppgaven får du kun små eksempler på kode for å hjelpe deg til å komme frem til resultatet. Derfor skal vi gå gjennom tankemåten til å lage animasjonen over ved å presentere en liste over ting som må gjøres:

La oss studere animasjonen og analysere hva den inneholder:

- ☐ En partikkel i midten av skjermen som alltid er der. Hva kan være grunnen til det?
- ☐ Partiklene som går ut fra midten og blir mindre og mindre jo lengre ut de går
- ☐ Hastigheten til hvert partikkel varierer
- ☐ Retningen varierer, men et partikkel reiser i en rett linje
- ☐ Det er mange partikler som blir til hvert sekund

La oss analysere punktene over, og se hva på hva vi må programmere. Vi starter fra toppen

- ☐ Siden partiklene går ut fra midten må jo alle starte der, derfor må vi setter `x` - og `y` -posisjonen til å være det samme for hvert partikkel.
- ☐ Siden partiklene blir mindre og mindre, men starter med samme størrelse, må vi endre på `størrelser` -attributtet til partiklet på samme måte som vi gjør når vi skal flytte på det. Tips: bruk ganging ( `*` ) for å få en bedre minknings-effekt.
- ☐ Siden hastigheten varierer kan vi bruke `Math.random` til `xSpeed` og `ySpeed` , her er et forslag til hvordan det kan se ut:

```
xSpeed: Math.floor(Math.random()*20 - Math.random()*20));
```

Dette vil gjøre at du får et positivt eller negativt tall med varierende hastighet fra -20 til 20 i `x` -retning. Gjør det samme for `y` -retningen for å få partiklene til å bevege seg overalt på skjermen.

- For å få dem til å følge en rett linje bruker vi bare endringer i `x` - og `y` -retning fra forrige oppgave: `particle.x = particle.x + particle.xSpeed;`
- Siden det er mange som blir laget på engang må vi for hver gang `draw()` blir kalt, legge et nytt partikkel i en `liste` og bruke en `for` -løkke til å endre hvert partikkel sine attributter og gjenta dette for alle elementene i listen.

Prøv selv først! Dersom du ikke får det til kan du benytte deg av hintene under.

## Hint

### For-løkke

- En `for` -løkke som skal gå gjennom en liste vil se slik ut:

```
for(var i = 0; i<listeNavn.length; i++){  
    //kode  
    element = listeNavn[i] // element blir nå det i-te elementet i liste  
    n, "i" blir her et tall fra 0 til lengden av listen.  
}
```

### Oppbygning av koden

For at du skal kunne bygge opp koden slik at partiklene oppfører seg som den gjør i animasjonen må vi tenke over hvor vi putter koden vår.

- All endring på partikkel-objektet bør skje i `for` -løkken. På denne måten vil endringene skje gradvis som gjør at animasjonen blir finere.
- Når man bør legge elementer i `partikkel` -lista bør du eksperimentere litt med.
- Du bør også eksperimentere litt med når du bruker `clearRect()` , klarer du å se hva som er forskjellen på om du legger den i eller utenfor `for` -løkken?



Lisens: CC BY-SA 4.0 (<http://creativecommons.org/licenses/by-sa/4.0/deed>)