HENDELSER

Lyttere (event listeners) og hendelser

Hovedoppgaven til JavaScript er å gi interaksjon til nettsider. Foreløpig har vi brukt JavaScript til å endre innhold på nettsider, men med *hendelser* kan vi begynne å bruke JavaScript til å reagere på det brukeren gjør på en nettside.

Når vi er inne på en nettside, registrerer nettleseren ting som hvor vi klikker med musa, og om vi trykker på tastaturet. Det er dette vi kaller hendelser. Det finnes hendelser for nesten alt vi kan foreta oss på en nettside, fra skalering av nettleservinduet til markering av tekst på nettsiden. I tabellen nedenfor kan du se noen eksempler på hendelser. Ved å bruke disse hendelsene på en fornuftig måte kan vi gi brukeren en god opplevelse av en nettside, og vi kan skape avansert interaktivitet som åpner for mange muligheter.

Hendelse	Fyres av når ...
`blur`	et element mister fokus (f.eks. når et tekstfelt forlates)
`click`	et element blir klikket på
`dblclick`	et element blir dobbeltklikket på
`focus`	et element får fokus (f.eks. når et tekstfelt velges)
`input`	et `<input>` - eller `<textarea>` - element endres
`change`	et `<select>` -element endres

Hendelse	Fyres av når ...
`keydown`	en tast trykkes
`keyup`	en tast slippes
`mousemove`	musepekeren beveges mens den er over et element
`mouseout`	musepekeren forlater et element
`mouseover`	musepekeren føres over et element

Hendelsene `input` og `change` gjelder for elementer som gjennomgås i læringsløpet *Brukerinput*.

Snakk!

Hendelser

Se gjennom tabellen ovenfor. Når tror du det er vanlig å bruke de ulike hendelsene? Kan du tenke deg situasjoner på nettsider du har vært på, der en eller flere av disse hendelsene ble brukt?

Lyttere

Når en hendelse oppstår i nettleseren, sier vi at hendelsen *fyres av*. Det fyres av hendelser hele tiden, for eksempel når vi flytter på musepekeren eller trykker på en knapp. For å kunne bruke en slik hendelse i koden vår må vi lage en *lytter*. En lytter følger med på en spesifikk hendelse, og hvis hendelsen inntreffer, gir den oss muligheten til å la noe skje.

addEventListener()

Metoden vi bruker for å lytte etter hendelser, heter `addEventListener()` . Denne metoden bruker vi på et element i DOM-treet, og når den ønskede hendelsen fyres av på elementet vi har valgt, vil lytteren reagere. Vi kan for eksempel gjøre om teksten i en overskrift når vi trykker på den (prøv selv!):

HTML CSS JS Re

LIVE

```
1  let overskriftEl = document.querySelector("h1");
2  overskriftEl.addEventListener("click", skrivAntall);
3
4  let antallTrykk = 0;
5
6  function skrivAntall() {
7    antallTrykk++;
8    overskriftEl.innerHTML = "Du har trykket på meg " + antallTrykk +
   " ganger.";
9  }
```

Resources 1×

Her henter vi først ut et overskriftselement. Husk at vi bare får det første `<h1>` -elementet med denne metoden. Hvis nettsiden inneholder flere overskrifter, bør du i stedet bruke en id for å være sikker på at du velger riktig overskrift.

Når vi har hentet overskriftselementet, kan vi plassere en lytter på det. Her lytter vi etter hendelsen `"click"` . Når noen klikker på overskriften, vil hendelsen `"click"` fyres av på elementet. Ved å lytte etter denne hendelsen kan vi la noe skje når den inntreffer. For at noe skal skje, må vi også angi en funksjon når vi bruker `addEventListener()` . I funksjonen plasserer vi koden som skal utføres når noen klikker på overskriften. Legg merke til at vi skriver funksjonsnavnet *uten* parenteser i `addEventListener()` .

Legg merke til at vi her også har lagt til CSS-egenskapen `cursor: pointer;` på `<h1>` -elementet. Denne egenskapen gjør at musepekerens utseende blir til en hånd i stedet for en pil, noe som gir opplevelsen av at det er mulig å trykke på elementet. Slike små detaljer er viktige for å gi gode og oversiktlige brukergrensesnitt.

Den generelle bruken av `addEventListener()` kan oppsummeres slik:

```
element.addEventListener("hendelse", funksjon);
```

Vi må ha et *element* som vi lytter på, og en spesifikk *hendelse* som vi lytter etter. Når hendelsen fyres av på elementet vårt, vil funksjonen utføres. Hvis du leser spesifikasjonen til metoden `addEventListener()` , vil du se at den også kan bruke et tredje argument. Det tredje argument sier noe om hvordan nettleseren skal behandle hendelsen, men dette har ikke noen viktig betydning for vår bruk; vi har derfor valgt å la være å fokusere på dette argumentet.

removeEventListener()

I enkelte situasjoner vil det være ønskelig å ta bort en lytter. Det er for eksempel tilfellet for de fleste spill. Når et spill er ferdig, bør det ikke lenger være mulig å gjøre noe for brukeren. Dette er et annet eksempel på små detaljer som forbedrer

et brukergrensesnitt. For å fjerne en lytter bruker vi metoden `removeEventListener()`. Vi kan for eksempel ta bort lytteren på `<h1>`-elementet i eksemplet ovenfor:

```
overskriftEl.removeEventListener("click", skrivAntall);
```

Her angir vi både hendelsen det er snakk om (`"click"`), og hvilken funksjon som skal fjernes (`skrivAntall`). Vi må spesifisere funksjonen fordi et element kan ha flere funksjoner koblet til den samme hendelsen.

Oppgaver

1. Lag en kopi av koden ovenfor og legg til en CSS-klasse som gir en farget kantlinje.
2. Lag et `<p>`-element med litt tekst nedenfor overskriften og legg til en lytter på `<p>`-elementet slik at avsnittet får den nye CSS-klassen når du klikker på det.
3. Legg til enda en CSS-klasse som gir en kantlinje med en annen farge.
4. Utvid hendelsesfunksjonen slik at `<p>`-elementet veksler mellom de to CSS-klassene hver gang du trykker på det. For å sjekke hvilken klasse som er satt, kan du bruke `avsnittEl.className` (bytt ut `avsnittEl` med ditt variabelnavn).
5. Gjør om funksjonen `skrivAntall()` slik at lytteren fjernes etter 10 klikk. Gjør også om teksten i `<h1>`-elementet til «Nå har du klikket 10 ganger, nå får du ikke klikke mer!».
6. Lag tre `<p>`-elementer med fargene rød, gul og grønn. De tre elementene skal ligge ved siden av hverandre. Lag også et `<p>`-element med teksten «Klikk på den røde firkanten». Du skal nå lage et enkelt klikkespill som skal fungere slik:
   a. Når brukeren klikker på det røde elementet, skal teksten endres til «Før musepekeren over den gule firkanten». Fjern lytteren fra den røde firkanten når hendelsen inntreffer.
   b. Når brukeren fører musepekeren over den gule firkanten, skal teksten endres til «Dobbeltklikk på den grønne firkanten». Fjern lytteren fra den gule firkanten når

hendelsen inntreffer.

c Når brukeren dobbeltklikker på den grønne firkanten, skal teksten endres til en hendelse du velger selv. Fjern lytteren fra den grønne firkanten når hendelsen inntreffer.

d Når den siste oppgaven utføres, skal den tilhørende lytteren fjernes, og teksten skal igjen bli «Klikk på den røde firkanten». Da er vi tilbake til start, og spillet kan spilles på nytt.

I dybden

Andre metoder for håndtering av hendelser

I *Informasjonsteknologi 1–2* kommer vi bare til å bruke `addEventListener()` . Den metoden lar oss legge til flere lyttere på samme hendelse, i tillegg til at vi også lett kan fjerne lyttere. Du bør likevel kjenne til de to andre metodene som er vanlige å bruke, fordi du vil møte dem i kode du finner på internett.

Den ene metoden, som *ikke* anbefales å bruke, legger til lyttere direkte i HTML-attributter:

```
<p id="mittAvsnitt" onclick="minFunksjon()"> Trykk på meg </p>
```

Her legger vi til lytteren i attributtet `onclick="minFunksjon()"` . Denne metoden anbefales ikke, først og fremst fordi den blander HTML-kode (struktur) og JavaScript-kode (interaktivitet). Koden blir mer oversiktlig om vi holder språkene adskilt.

Den andre metoden er grei å bruke, men lar oss bare legge til én funksjon til hver lytter:

```
mittAvsnittEl = document.querySelector("#mittAvsnitt");
mittAvsnittEl.onclick = minFunksjon;
```

Her bruker vi egenskapen `onclick` på elementet vi ønsker å gi en lytter. Denne framgangsmåten er ikke veldig ulik `addEventListener()` . Vi har valgt å bruke sistnevnte fordi den gir flest muligheter, og fordi det er den anbefalte framgangsmåten.