

# Programmerings Eksamen

## Kilder:

ChatGPT, Jeg samarbeidet også sammen med en medstudent KUN på angitt område, Copilot(Visual Studio Code AI)

## ChatGPT: DEL 2

BRuk det ved hjelp av de her: `const randomHex = ["1","2","3","4","5","6","7","8","9","A","B","C","D","E","F"]`  
//Lager en variabel som inneholder #  
`let hexColor = "#";`

## Kopiert i fra ChatGPT:

```
57 | //Lager en loop som tar max 6 tegn fra arrayen over og legger til hexColor (Kopiert fra ChatGPT "linje 58-61")
58 | for (let i = 0; i < 6; i++){
59 |     hexColor += arrayForRandomHex[Math.floor(Math.random() * arrayForRandomHex.length)]
60 | }
61 | return hexColor
62 | }
```



Hvis du ønsker å bruke den samme funksjonen, men generere en tilfeldig fargekode i hex-format (som #RRGGBB), kan du gjøre det ved hjelp av den arrayen du har oppgitt (`randomHex`). Her er hvordan du kan implementere dette:

### Fullstendig kode for å lage en tilfeldig hex-farge:

```
javascript Kopier kode

const randomHex = [
  "1", "2", "3", "4", "5", "6", "7", "8", "9", "A", "B", "C", "D", "E", "F"
];

// Funksjon for å generere en tilfeldig hex-farge
function generateRandomHexColor() {
  let hexColor = "#"; // Start med #
  for (let i = 0; i < 6; i++) { // Hex-kode har 6 tegn (2 tegn for hver av R, G, B)
    hexColor += randomHex[Math.floor(Math.random() * randomHex.length)];
  }
  return hexColor;
}

// Eksempel på bruk:
for (let i = 0; i < 5; i++) { // Looper 5 ganger
  let randomColor = generateRandomHexColor(); // Generer en tilfeldig hex-farge
  console.log(randomColor); // Skriv ut fargen
}
```

### Forklaring på koden:

- `randomHex` array: Dette er arrayen som inneholder alle mulige tegn som kan brukes i en hex-fargekode. Hex-koder består av tallene 0-9 og bokstavene A-F.
- `generateRandomHexColor()`:
  - Jeg har laget en funksjon som starter med "#", som er nødvendig for å definere en hex-farge.
  - Funksjonen går gjennom en `for`-sløyfe 6 ganger, fordi en hex-fargekode består av 6 tegn (2 for rød, 2 for grønn og 2 for blå).
  - Hver gang velger den et tilfeldig tegn fra `randomHex`-arrayen og legger det til `hexColor`.
- `Math.floor(Math.random() * randomHex.length)`:
  - `Math.random()` genererer et tilfeldig tall mellom 0 og 1.
  - Ved å multiplisere det med `randomHex.length` (som er 16, antall elementer i arrayen), får vi et tall mellom 0 og 15.
  - `Math.floor()` runder det ned til nærmeste heltall, som vi deretter bruker som indeks for å hente et tilfeldig tegn fra `randomHex`-arrayen.

### Eksempel på utdata:

Hvis du kjører koden, vil den skrive ut 5 tilfeldige hex-farger, for eksempel:

3. `Math.floor(Math.random() * randomHex.length)`:

- `Math.random()` genererer et tilfeldig tall mellom 0 og 1.
- Ved å multiplisere det med `randomHex.length` (som er 16, antall elementer i arrayen), får vi et tall mellom 0 og 15.
- `Math.floor()` runder det ned til nærmeste heltall, som vi deretter bruker som indeks for å hente et tilfeldig tegn fra `randomHex`-arrayen.

### Eksempel på utdata:

Hvis du kjører koden, vil den skrive ut 5 tilfeldige hex-farger, for eksempel:

```
bash Kopier kode

#5A9D3F
#1E4C2A
#2F6D7A
#D56F9C
#3A8BFF
```

Hver gang du kjører funksjonen, vil du få forskjellige hex-farger.

DEL 2, Jeg skrev inn:

```
//Arbeidskrav D, oppgave 1:  
const btn = document.getElementById("savetask")
```

Og fikk deretter et forslag av Copilot (Visual Studio Code-AI):

```
//Lager en eventlistener som lytter etter klikk på knappen "Lagre" og  
kjører funksjonen "saveTask" når den blir klikket på.  
btn.addEventListener("click", saveTask)  
//Lager en funksjon som lagrer oppgaven som blir skrevet inn i  
inputfeltet "task" i todos arrayen.  
function saveTask(){  
    //Lager en variabel som henter verdien fra inputfeltet  
    "task"  
    const task = document.getElementById("task").value  
    //Legger til en ny oppgave i todos arrayen med teksten fra  
    inputfeltet "task" og status "To do"  
    todos.push({text: task, status: "To do"})  
    //Oppdaterer listen med oppgaver  
    updateTasks()  
}
```

Endret noen av ID-ene og fikk enda et forslag ved de endringene:

```
//Oppdaterer oppgavelisten i HTML:  
let oppgaver = ""  
todos.map((item, index) => {  
    oppgaver += `<li>  
        <span class="task-text">${item.text}</span>  
        <span class="task-status ${item.status === "Done" ?  
"task-status-done": "task-status-todo"}">${item.status}</span>  
    </li>`  
    })  
document.getElementById("todolist").innerHTML = oppgaver
```

Sluttresultatet ble da:

```
//Arbeidskrav D, oppgave 1:
    const btn = document.getElementById("savetask")
    //Lager en eventlistener som lytter etter klikk på knappen
    "Lagre" og kjører funksjonen "taskSaved" når den blir klikket på
    btn.addEventListener("click", taskSaved)
    //Lager en funksjon som lagrer oppgaven som blir skrevet inn i
    inputfeltet "task" i todos arrayen:
    function taskSaved(){
        //Lager en variabel som henter verdien fra inputfeltet
        "task"

        const task = document.getElementById("task").value
        //Legger til en ny oppgave i todos arrayen, med statusen
        "To do"

        todos.push({text: task, status: "To do"})
        //Oppdaterer oppgavelisten i HTML:
        let oppgaver = ""
        //Går igjennom alle objektene i todos arrayen og legger de
        til listen i HTML
        todos.map((item, index) => {
            oppgaver += `<li>
                <span class="task-text">${item.text}</span>
                <span class="task-status ${item.status === "Done" ?
                "task-status-done": "task-status-todo"}">${item.status}</span>
            </li>`
        })
        document.getElementById("todolist").innerHTML = oppgaver
    }
```

## DEL 2

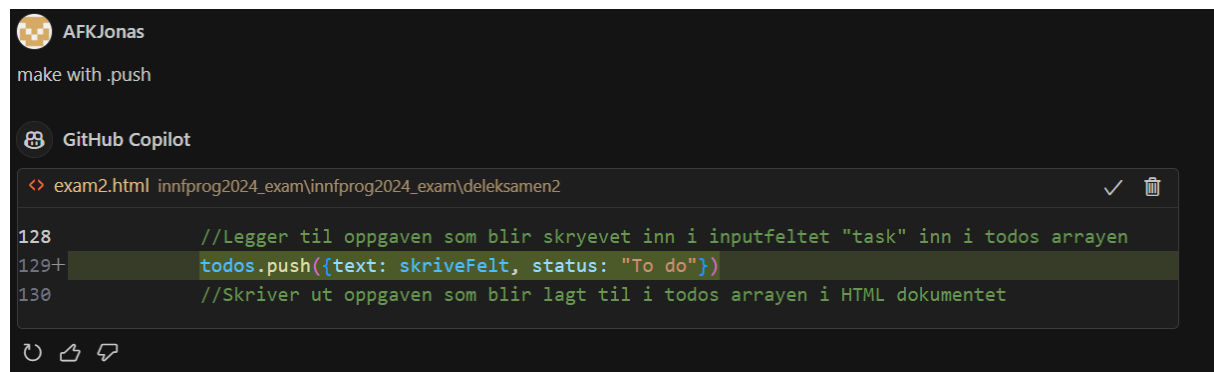
Diskuterte med en med student KUN om denne koden (som er bilde rett under), og han hjalp meg slik at den ble endret til slik:

```
120 //Arbeidskrav D, oppgave 1:
121 const btn = document.getElementById("savetask")
122 //Lager en eventlistener som lytter etter klikk på knappen "Lagre" og kjører funksjonen "leggeTilArray()" når den blir klikket på
123 btn.addEventListener("click", leggeTilArray)
124 //Lager en funksjon som lagrer oppgaven som blir skrevet inn i inputfeltet "task" i todos arrayen:
125 function leggeTilArray(){
126     //Laget en variabel som henter verdien som blir skrevet inn i inputfeltet "task"
127     const skriveFelt = document.getElementById("task").value
128     //Legger til oppgaven som blir skrevet inn i inputfeltet "task" inn i todos arrayen
129     todos.push({text: skriveFelt, status: "To do"})
130     //Skriver ut oppgaven som blir lagt til i todos arrayen i HTML dokumentet
131     oppgaver += `<li>
132         <span class="task-text">${skriveFelt}</span>
133         <span class="task-status task-status-todo">To do</span>
134     </li>`
135     document.getElementById("todolist").innerHTML = oppgaver
136 }
137
138
```

Men jeg ville heller ha med en “.push()” da dette legger til elementer i arrayen, slik som vi har blitt lært.

Da slettet jeg “todos.text = skriveFelt”

Og tastet “CTRL + I” slik at jeg fikk opp Copilot og skrev “make with .push”, tastet det slik at det kom i chat, og da ble det seende ut slik:



trykket på “✓” symbolet og fikk det ut til kode (på linje 129):

```

125 function leggeTilArray(){
126     //Laget en variabel som henter verdien som blir skrevet inn i inputfeltet "task"
127     const skriveFelt = document.getElementById("task").value
128     //Legger til oppgaven som blir skrevet inn i inputfeltet "task" inn i todos arrayen
129     todos.push({text: skriveFelt, status: "To do"})
130     //Skriver ut oppgaven som blir lagt til i todos arrayen i HTML dokumentet
131     oppgaver += `<li>
132         <span class="task-text">${skriveFelt}</span>
133         <span class="task-status task-status-todo">To do</span>
134     </li>`
135     document.getElementById("todolist").innerHTML = oppgaver
136 }

```

Da ble det lagret i arrayen på nettsiden:

### Oppgaver

Les eksamensteksten	Done
Løs Innføring i programmerings-eksamen	To do
Lag funksjonalitet for å registrere nye oppgaver	To do
Lag funksjonalitet for å sette en eksisterende oppgave til utført ved å endre status	To do
3e4r2qwe4r	To do
3e4r2qwe4r	To do
3e4r2qwe4r	To do
3e4r2qwe4r	To do
3e4r2qwe4r	To do
3e4r2qwe4r	To do
3e4r2qwe4r	To do
3e4r2qwe4r	To do
3e4r2qwe4r	To do
3e4r2qwe4r	To do
3e4r2qwe4r	To do

Filter output

Errors Warnings Logs Info Debug CSS XHR Requests

exam2.html:111

Array(4) [ {...}, {...}, {...}, {...} ]

- 0: Object { text: "Les eksamensteksten", status: "Done" }
- 1: Object { text: "Løs Innføring i programmerings-eksamen", status: "To do" }
- 2: Object { text: "Lag funksjonalitet for å registrere nye oppgaver", status: "To do" }
- 3: Object { text: "Lag funksjonalitet for å sette en eksisterende oppgave til utført ved å endre status", status: "To do" }
- 4: Object { text: "3e4r2qwe4r", status: "To do" }
- 5: Object { text: "3e4r2qwe4r", status: "To do" }
- 6: Object { text: "3e4r2qwe4r", status: "To do" }
- 7: Object { text: "3e4r2qwe4r", status: "To do" }
- 8: Object { text: "3e4r2qwe4r", status: "To do" }
- 9: Object { text: "3e4r2qwe4r", status: "To do" }
- 10: Object { text: "3e4r2qwe4r", status: "To do" }
- 11: Object { text: "3e4r2qwe4r", status: "To do" }
- 12: Object { text: "3e4r2qwe4r", status: "To do" }

length: 13

<prototype>: Array []

For klokken kopierte jeg ifra

[https://www.w3schools.com/js/tryit.asp?filename=tryjs\\_timing\\_clock](https://www.w3schools.com/js/tryit.asp?filename=tryjs_timing_clock)



Endret litt på ID-en slik at den ble skrevet ut i riktig sted.

```
147 //CLOCK
148 //Kopierte funksjonen (linje 150 - 164) direkte fra: https://www.w3schools.com/js/tryit.asp?filename=tryjs_timing_clock
149 //Endret KUN hvor klokken skal skrives ut, fra "txt" til "clock" på linje 157, og la til "startTime()" slik at funksjonen kjører
150 //Deklarerer funksjonen
151 function startTime() {
152     //Deklarerer variabelen "today" som henter dagens dato
153     const today = new Date();
154     //Deklarerer variablene "h", "m", "s" som henter både timer, minutter og sekunder fra "today"
155     let h = today.getHours();
156     let m = today.getMinutes();
157     let s = today.getSeconds();
158     //Legger til en null foran tallene hvis de er mindre enn 10
159     m = checkTime(m);
160     s = checkTime(s);
161     //Skriver ut klokkeslettet i HTML dokumentet ved at den henter iden "clock" og skriver ut "h:m:s" / timer, minutter og sekunder
162     document.getElementById("clock").innerHTML = h + ":" + m + ":" + s;
163     //Kjører funksjonen hvert 1 000 millisekund eller hvert sekund
164     setTimeout(startTime, 1000);
165 }
166
167 //Denne funksjonen legger til en null foran tallene hvis de er mindre enn 10, slik at de blir skrevet ut som "01" istedenfor "1"
168 function checkTime(i) {
169     if (i < 10) {i = "0" + i}; // add zero in front of numbers < 10 (- kopierte direkte notat fra w3schools)
170     //Returnerer verdien i (med null foran hvis verdien er mindre enn 10) tilbake til funksjonen
171     return i;
172 }
173 //Starter funksjonen
174 startTime();
175 </script>
176 </body>
177 </html>
```

Tok ikke med html-en da ID-en for plasseringen var allerede laget.

## Spurte ChatGPT:

problemet mitt med min kode er at det er kun farge boksen som flytter seg og ikke både det andre



Problemet i koden din er at posisjonen til de andre elementene (som små rektangler og tekst) ikke er relatert til `index`. De er alltid plassert på faste posisjoner på canvaset (f.eks. `(0, 0)` eller `(80, 80)`) i stedet for å bevege seg sammen med fargeboksene.

### Løsning:

For at alle elementer skal bevege seg sammen med fargeboksene, må du gjøre posisjonene for små rektangler og tekst dynamiske ved å bruke `index`. Dette knytter dem til den horisontale posisjonen til fargeboksen (`index * 100`).

### Oppdatert Kode:

```
javascript
// Kopier kode

randomHex.forEach((item, index) => {
  // Tegner fargeboksen
  ctx.fillStyle = item;
  ctx.fillRect(index * 100, 0, 100, 100);

  // Rektangel oppe til venstre i fargeboksen
  ctx.beginPath();
  ctx.fillStyle = "white";
  ctx.fillRect(index * 100 + 5, 5, 70, 15); // Flytter med fargeboksen
  ctx.closePath();

  // Tekst inni rektangelet
  ctx.beginPath();
  ctx.font = "11px Arial";
  ctx.fillStyle = "black";
  ctx.fillText(item, index * 100 + 10, 15); // Flytter med fargeboksen
  ctx.closePath();

  // Hvit firkant nede til høyre i fargeboksen
  ctx.beginPath();
  ctx.fillStyle = "white";
  ctx.fillRect(index * 100 + 80, 80, 20, 20); // Flytter med fargeboksen
  ctx.closePath();

  // Tekst inni den hvite firkanten
  ctx.beginPath();
  ctx.font = "10px Arial";
  ctx.fillStyle = "black";
  ctx.fillText("1", index * 100 + 85, 95); // Flytter med fargeboksen
  ctx.closePath();
});
```

### Hva er nytt?

#### 1. Dynamisk plassering:

- Alle koordinater er justert ved å bruke `index * 100`. Dette sikrer at små rektangler og tekst alltid er i forhold til fargeboksens posisjon.
- Eksempler:
  - Rektangelet oppe til venstre: `(index * 100 + 5, 5)`

### Hva er nytt?

#### 1. Dynamisk plassering:

- Alle koordinater er justert ved å bruke `index * 100`. Dette sikrer at små rektangler og tekst alltid er i forhold til fargeboksens posisjon.
- Eksempler:
  - Rektangelet oppe til venstre: `(index * 100 + 5, 5)`
  - Tekst oppe til venstre: `(index * 100 + 10, 15)`
  - Hvit firkant nede til høyre: `(index * 100 + 80, 80)`
  - Tekst nede til høyre: `(index * 100 + 85, 95)`

#### 2. Alt er knyttet til boksen:

- Nå beveger både små rektangler og tekst seg sammen med hver fargeboks når den plasseres i en ny posisjon.

### Resultat:

Med den oppdaterte koden vil:

- Hver fargeboks (fra `randomHex`) tegnes i en horisontal rekke.
- Små detaljer som rektangler og tekst følger fargeboksen, og alt holder seg innenfor riktig boks.

Dette løser problemet der bare fargeboksen beveger seg, mens andre elementer forble stasjonære!

🔊 📄 📌 🔄 🏠

Melding ChatGPT



Koden min før:

```
78 //Arbeidskrav D, oppgave 1:
79 //requestAnimationFrame
80 const requestAnimationFrame = window.requestAnimationFrame || window.msRequestAnimationFrame || window.mozRequestAnimationFrame || window.webkitRequestAnimationFrame
81 //Funksjon som tegner fargene i canvaset
82 function renderCanvas() {
83     //Tømmer canvaset
84     ctx.clearRect(0, 0, canvas.width, canvas.height)
85     //Lager en forEach løkke som går igjennom arrayen randomHex og skriver ut fargene i canvaset
86     randomHex.forEach((item, index) => {
87         //Klassifiserer fillStyle som fargene i arrayen
88         ctx.fillStyle = item
89         //Tar fargene ifra fillStyle og legger de i fargefirkanter i canvaset
90         ctx.fillRect(index * 100, 0, 100, 100)
91
92         //Rektangel inni fargeboksen oppe til venstre
93         ctx.beginPath()
94         ctx.fillStyle = "white"
95         ctx.fill()
96         ctx.fillRect(0, 0, 70, 15)
97         ctx.closePath()
98
99         //Tekst inni rektangelet
100         ctx.beginPath()
101         ctx.font = "11px Arial"
102         ctx.fillStyle = "black"
103         ctx.fillText(item, index + 5, 10)
104         ctx.closePath()
105
106         //Hvit firkant nede i høyre hjørne inni fargeboksen
107         ctx.beginPath()
108         ctx.fillStyle = "white"
109         ctx.fill()
110         ctx.fillRect(80, 80, 20, 20)
111         ctx.closePath()
112
113         //Tekst inni hvit firkant
114         ctx.beginPath()
115         ctx.font = "10px Arial"
116         ctx.fillStyle = "black"
117         ctx.fillText("1", 85, 95)
118         ctx.closePath()
119
120     });
121
122     requestAnimationFrame(renderCanvas);
123 }
124
125 //Initial running of drawing canvas
126 renderCanvas()
127
128
129
130
131
```

Koden min etter:

```
78 //Arbeidskrav D, oppgave 1:
79 //requestAnimationFrame
80 const requestAnimationFrame = window.requestAnimationFrame || window.msRequestAnimationFrame || window.mozRequestAnimationFrame || window.webkitRequestAnimationFrame
81 //Funksjon som tegner fargene i canvaset
82 function renderCanvas() {
83     //Tømmer canvaset
84     ctx.clearRect(0, 0, canvas.width, canvas.height)
85     //Lager en forEach løkke som går igjennom arrayen randomHex og skriver ut fargene i canvaset
86     randomHex.forEach((item, index) => {
87         //Klassifiserer fillStyle som fargene i arrayen
88         ctx.fillStyle = item
89         //Tar fargene ifra fillStyle og legger de i fargefirkanter i canvaset
90         ctx.fillRect(index * 100, 0, 100, 100)
91
92         //Hvit rektangel inni fargeboksen oppe til venstre
93         //begynner å tegne
94         ctx.beginPath()
95         //Klassifiserer hvilken farge rektangelet skal ha
96         ctx.fillStyle = "white"
97         //Fyller rektangelet med den angitte fargen og tegner rektangelet i den valgte posisjonen
98         ctx.fillRect(index * 100, 0, 70, 15)
99         //Ferdiggjør tegningen
100         ctx.closePath()
101
102         //Tekst inni rektangelet
103         //Begynner å tegne
104         ctx.beginPath()
105         //Velger tekst-type og størrelse
106         ctx.font = "12px Arial"
107         //Velger farge på teksten
108         ctx.fillStyle = "black"
109         //Skriver ut teksten(og fargen på teksten) i rektangelet på den valgte posisjonen
110         ctx.fillText(item.toLowerCase(), index * 100 + 5, 10)
111         //Ferdiggjør tegningen
112         ctx.closePath()
113
114         //Hvit firkant nede i høyre hjørne inni fargeboksen
115         //Begynner å tegne
116         ctx.beginPath()
117         //Klassifiserer hvilken farge rektangelet skal ha
118         ctx.fillStyle = "white"
119         //Fyller rektangelet med den angitte fargen og tegner rektangelet i den valgte posisjonen
120         ctx.fillRect(index * 100 + 80, 80, 20, 20)
121         //Ferdiggjør tegningen
122         ctx.closePath()
123
124         //Tekst inni hvit firkant
125         //Begynner å tegne
126         ctx.beginPath()
127         //Velger tekst-type og størrelse
128         ctx.font = "12px Arial "
129         //Velger farge på teksten
130         ctx.fillStyle = "black"
131         //Skriver ut teksten(og fargen på teksten) i rektangelet på den valgte posisjonen
132         ctx.fillText(index + 1, index * 100 + 85, 95)
133         //Ferdiggjør tegningen
134         ctx.closePath()
135         //Ferdiggjør tegningen
136     });
137 }
```



Ble bedt om å google på del 2 arbeidskrav C oppg. 1, så istedenfor brukte jeg ChatGPT, for å spare tid, jeg la inn “document.getElementById("contactlist").innerHTML = contactlist.sort()” fordi jeg viste den var feil og ChatGPT ville respondere på det og gi meg forklaringen.

```
document.getElementById("contactlist").innerHTML = contactlist.sort()
```



Koden `document.getElementById("contactlist").innerHTML = contactlist.sort()` vil ikke fungere som forventet. Her er hvorfor, og hvordan du kan løse det:

### Problem med koden:

- `contactlist` er en streng, ikke en array: I tidligere eksempler er `contactlist` en streng som inneholder HTML-koden for å vise kontaktene. Funksjonen `sort()` fungerer bare på arrays, ikke på strenger.
- Hvis du prøver å sortere kontaktene: Hvis du ønsker å sortere `contacts`-arrayen (ikke `contactlist`), må du sortere den før du genererer HTML-koden.

### Hvordan fikse det:

Hvis du vil vise kontaktene i sortert rekkefølge etter et felt, for eksempel etter etternavn, gjør dette:

#### Steg 1: Sorter `contacts`-arrayen

Bruk `sort()` på arrayen før du bygger HTML. Eksempel:

```
javascript
contacts.sort((a, b) => a.lastname.localeCompare(b.lastname));
```

Dette sorterer kontaktene alfabetisk etter `lastname`.

#### Steg 2: Generer HTML-koden

Bygg `contactlist` basert på den sorterte arrayen, slik:

```
javascript
let contactlist = "";
contacts.forEach(contact => {
  contactlist += `<article>
    <p><strong>${contact.lastname}, ${contact.firstname}</strong><br />
    Kontor: ${contact.office} | E-post: <a href="mailto:${contact.mail}">${contact.mail}</a>
  </article>`;
});
```

#### Steg 3: Oppdater DOM

Sett den sorterte HTML-en i `contactlist`-elementet:

```
javascript
document.getElementById("contactlist").innerHTML = contactlist;
```



### Fullstendig kode:

```
javascript
contacts.sort((a, b) => a.lastname.localeCompare(b.lastname)); // Sorterer etter etternavn

let contactlist = "";
contacts.forEach(contact => {
  contactlist += `<article>
    <p><strong>${contact.lastname}, ${contact.firstname}</strong><br />
    Kontor: ${contact.office} | E-post: <a href="mailto:${contact.mail}">${contact.mail}</a>
  </article>`;
});

document.getElementById("contactlist").innerHTML = contactlist; // Oppdaterer HTML
```

### Forklaring:

- `localeCompare`: En metode som brukes for å sammenligne strenger på en måte som respekterer alfabetisk rekkefølge.
- `contacts.sort`: Sorterer arrayen.
- `Oppdatering av DOM`: Den sorterte listen legges til i HTML-elementet.

Når denne koden kjører, vises kontaktene i alfabetisk rekkefølge etter etternavn i nettleseren.



Kode før:

```
57 //CONTACTS
58 const contacts = [
59   {lastname: "Hansen", firstname: "Stine", office: "D2", mail: "s.hansen@company.com"},
60   {lastname: "Andersen", firstname: "Willy", office: "D1", mail: "w.andersen@company.com"},
61   {lastname: "Danilo", firstname: "Luciano", office: "E2", mail: "l.danilo@company.com"},
62   {lastname: "Wester", firstname: "Zato", office: "C3", mail: "z.wester@company.com"},
63   {lastname: "Jansson", firstname: "Sigried", office: "B1", mail: "s.jansson@company.com"}
64 ]
65
66 let contactlist = ""
67
68 //Oppgave 2. minstekrav: Åpner en ny e-post når du klikker på en e-postadressene:
69 //Lagt til en <a> tag rundt e-postadressen (linje-73) slik at e-postene blir klikkbare
70 contacts.map((contact, index) => {
71   contactlist += `<article>
72     <p><strong>${contact.lastname}, ${contact.firstname}</strong><br />
73     Kontor: ${contact.office} | E-post: <a href="mailto:${contact.mail}"> ${contact.mail}</a></p>
74   </article>`
75 })
76
77 document.getElementById("contactlist").innerHTML = contactlist
78
```

Kode etter:

```
57 //CONTACTS
58 const contacts = [
59   {lastname: "Hansen", firstname: "Stine", office: "D2", mail: "s.hansen@company.com"},
60   {lastname: "Andersen", firstname: "Willy", office: "D1", mail: "w.andersen@company.com"},
61   {lastname: "Danilo", firstname: "Luciano", office: "E2", mail: "l.danilo@company.com"},
62   {lastname: "Wester", firstname: "Zato", office: "C3", mail: "z.wester@company.com"},
63   {lastname: "Jansson", firstname: "Sigried", office: "B1", mail: "s.jansson@company.com"}
64 ]
65
66 let contactlist = ""
67
68 //Arbeidskrav C, oppgave 1:
69 //Linje 71 er kopiert direkte ifra ChatGPT:
70 //Linjen under sorterer kontaktene etter etternavn FØR de blir skrevet ut i HTML:
71 contacts.sort((a, b) => a.lastname.localeCompare(b.lastname))
72 //Linjen over er kopiert direkte ifra ChatGPT
73
74 //Oppgave 2. minstekrav: Åpner en ny e-post når du klikker på en e-postadressene:
75 //Lagt til en <a> tag rundt e-postadressen (linje-73) slik at e-postene blir klikkbare
76 contacts.map((contact, index) => {
77   contactlist += `<article>
78     <p><strong>${contact.lastname}, ${contact.firstname}</strong><br />
79     Kontor: ${contact.office} | E-post: <a href="mailto:${contact.mail}"> ${contact.mail}</a></p>
80   </article>`
81 })
82
83 document.getElementById("contactlist").innerHTML = contactlist
```

Kopierte Direkte inn på linje 71:

"contacts.sort((a, b) => a.lastname.localeCompare(b.lastname));"

Fikk et problem på arbeidskrav C-oppgave 3, når det ikke var noe i inputfeltet kom det en advarsel da jeg trykket på “Lagre”(som det skal), men allikevel ville det “tomme” inputfeltet bli lagt til. Fant ikke ut problemet, men vet at det ligger inni funksjonen “leggeTilArray()”, så for å spare tid, plasserte jeg (hele koden min) i ChatGPT:

[illegible]

```

</strong><br />
    Kontor: ${contact.office} | E-post: <a
href="mailto:${contact.mail}"> ${contact.mail}</a> </p>
</article>
    })

document.getElementById("contactlist").innerHTML = contactlist

//BOOKS
const books = [
    {text: "Printerlekk", date: "02.11.24", amount: -750},
    {text: "Printerpapir", date: "02.11.24", amount: -439},
    {text: "Nettbutikkalsg, uke 44", date: "06.11.24", amount: 3428},
    {text: "Produksjonsmateriale", date: "07.11.24", amount: -1299},
    {text: "Kontorrekvisita", date: "10.11.24", amount: -644},
    {text: "Nettbutikkalsg, uke 45", date: "13.11.24", amount: 2988}
]

let equity = 23782; //Penger på konto ved inngangen til november
document.getElementById("incomingequity").innerHTML = equity
let balance = 0; //Startbalanse for utregning av utgifter og inntekter i november

//Arbeidskrav C, oppgave 2:
//Lager en funksjon som skriver ut bokføringene i books, i listen med bokføringer i HTML dokumentet, egentlig det samme som oppgave 1 minstekrav
function skriverUtBooks() {
    //Lager en tom variabel for at bokføringene blir lagret i denne
    let booklines = ""
    //Lager en .map() funksjon som går gjennom objektene i books og legger de inn i variabelen "booklines"
    books.map((book, index) => {
        //Bruker template literals for skrive ut bokføringene direkte
        booklines += <div class="bookline">
            <span class="booktext">${book.text}</span>
            <span class="bookdate">${book.date}</span>
            <span class="bookamount">${book.amount}</span>
        </div>
    });
    //Det som er lagret i variabelen "booklines" blir skrevet ut i HTML
    document.getElementById("booklines").innerHTML = booklines;
}

//Kjører funksjonen skriverUtBooks()

//TODOS
const todos = [
    {text: "Les eksamensteksten", status: "Done"},
    {text: "Les Innføring i programmerings-eksamen", status: "To do"},
    {text: "Lag funksjonalitet for å registrere nye oppgaver", status: "To do"},
    {text: "Lag funksjonalitet for å sette en eksisterende oppgave til utført ved å endre status", status: "To do"}
]

//Oppgave 1. minstekrav: Bytt ut de hardkodede oppgavene (linje 42-49 i exam2.html) med en dynamisk utskrift av oppgavene som ligger lagret i todos:

//Lager først en tom variabel som får verdiene ifra todos
let oppgaver = ""
//.map() funksjonen går gjennom alle objektene i todos
todos.map((item, index) => {
    //For hvert objekt lagret i todos, legger den til et nytt <li> element "$item.text" i "todolist" slik at det dannes seg en liste
    //Inni template literals er det en if/else som sjekker om $item.status) er satt til "Done" hvis den er "task-status-done", og hvis ikke, er den satt til "To do" ved at det står "task-status-todo"
    oppgaver += <li>
        <span class="task-text">${item.text}</span>
        <span class="task-status">${item.status} === "Done" ? "task-status-done": "task-status-todo"></span>
    </li>
})
//Linjen under gjør at oppgaver blir skrevet ut i HTML dokumentet
document.getElementById("todolist").innerHTML = oppgaver
console.log(todos) //Skriver det ut i konsollen

//Arbeidskrav D, oppgave 1:
const btn = dc <button id="savetask">
//Lager en eventlisterener som lytter etter klikk på knappen "Lag"

```

```

document.getElementById("todolist").innerHTML = oppgaver
console.log(todos) //Skriver det ut i konsollen

//Arbeidskrav D, oppgave 1:
const btn = document.getElementById("savetask")
//Lager en eventlisterene som lytter etter klikk på knappen "Lagre"
og kjerer funksjonen "leggeTilArray()" når den blir klikket på
btn.addEventListener("click", leggeTilArray)
//Lager en funksjon som lagrer oppgaven som blir skrevet inn i
inputfeltet "task" i todos arrayen:
function leggeTilArray(){

    function inputWarning(){
        if(document.getElementById("task").value === ""){
            alert("Du må skrive inn en oppgave for å lagre en
oppgave!")
        } return
    }
    inputWarning()

    //Laget en variabel som henter verdien som blir skrevet inn i
inputfeltet "task"
    const skriveFelt = document.getElementById("task").value
    //Legger til oppgaven som blir skrevet inn i inputfeltet "task"
inn i todos arrayen
    todos.push({text: skriveFelt, status: "To do"})
    //Skriver ut oppgaven som blir lagt til i todos arrayen i HTML
dokumentet
    oppgaver += <li>
    <span class="task-text">${skriveFelt}</span>
    <span class="task-status task-status-todo">To do</span>
    </li>
    document.getElementById("todolist").innerHTML = oppgaver

}

//Arbeidskrav C, oppgave 3:
//Lager en funksjon som varsler brukeren, dersom det er
ingenting i inputfeltet

//Arbeidskrav D, oppgave 2:
//CLOCK
//Kopierte funksjonen (linje 150 - 164)direkte fra:
https://www.w3schools.com/js/tryit.asp?filename=tryjs_timing_clock
//Endret KUN hvor klokken skal skrives ut, fra "txt" til "clock" på
linje 157, og la til "startTime()" slik at funksjonen kjerer
//Deklarerer funksjonen
function startTime() {
    //Deklarerer variabelen "today" som henter dagens dato
    const today = new Date();
    //Deklarerer variablene "h", "m", "s" som henter både timer,
minutter og sekunder fra "today"
    let h = today.getHours();
    let m = today.getMinutes();
    let s = today.getSeconds();
    //Legger til en null foran tallene hvis de er mindre enn 10
    m = checkTime(m);
    s = checkTime(s);
    //Skriver ut klokkeslettet i HTML dokumentet ved at den
henter iden "clock" og skriver ut "h:ms" / timer, minutter og sekunder
    document.getElementById("clock").innerHTML = h + ":" + m +
":" + s;
    //Kjerer funksjonen hvert 1 000 millisekund eller hvert sekund
    setTimeout(startTime, 1000);
}
//Denne funksjonen legger til en null foran tallene hvis de er
mindre enn 10, slik at de blir skrevet ut som "01" istedenfor "1"
function checkTime(i) {
    if (i < 10) {i = "0" + i}; // add zero in front of numbers < 10 (<-
kopiert direkte notat fra w3schools)
    //Returnerer verdien i (med null foran hvis verdien er mindre
enn 10) tilbake til funksjonen
    return i;
}
//Starter funksjonen
startTime();
</script>
</body>
</html>

```

## Svar ifra ChatGPT:



Takk for at du delte hele HTML- og JavaScript-koden din! Det ser ut som du har implementert flere funksjoner for å håndtere kontakter, bokføring, oppgaver, og en klokke. Nå ser jeg at du har et problem med å hindre at tomme oppgaver blir lagt til i `todos`-arrayen.

### Feilen:

I din `inputWarning()` funksjon er logikken riktig for å sjekke om input-feltet er tomt, men problemet er at funksjonen ikke stopper utførelsen av den etterfølgende koden i `leggeTilArray()`. Du kaller `inputWarning()` inne i `leggeTilArray()`, men den stopper ikke funksjonen hvis input er tomt.

### Løsning:

For at oppgavene ikke skal legges til når input-feltet er tomt, må vi stoppe videre kjøring av funksjonen `leggeTilArray()` hvis input er tomt. Dette kan vi gjøre ved å bruke `return` etter `alert()`-meldingen i `inputWarning()` eller rett i `leggeTilArray()`-funksjonen.

Her er den korrigerte koden:

```
javascript
// 7000: Legg til en funksjon som håndterer Input-feil
function leggeTilArray() {
  // Hent input verdien
  const inputField = document.getElementById("task").value;

  // Sjekk om input er tomt
  if (inputField === "") {
    // Hvis tomt, vis en alert og stopp videre utførelse av funksjonen
    alert("Du må skrive inn en oppgave for å lagre en oppgave!");
    return; // Stopp funksjonen her hvis input er tomt
  }

  // Legg oppgaven til arrayen "todos" og oppdater listen i HTML
  todos.push({ text: inputField, status: "To do" });

  // Oppdater oppgavene i HTML
  let oppgaver = "";
  todos.map((item) => {
    oppgaver += `<li>
      <span class="task-text">${item.text}</span>
      <span class="task-status ${item.status === "Done" ? "task-status-done" : "task-status-pending"}>${item.status}</span>
    </li>`;
  });

  // Skriv ut de oppdaterte oppgavene
  document.getElementById("todolist").innerHTML = oppgaver;

  // Tøm input-feltet etter at oppgaven er lagret
  document.getElementById("task").value = "";
}
```

### Hva har blitt endret:

- Input-sjekk:** Nå sjekkes input-feltet for å sikre at det ikke er tomt før vi legger til oppgaven. Hvis det er tomt, vises en alert og funksjonen stopper ved `return`.
- Funksjonen stopper:** Når `return` kalles etter `alert()`, stopper funksjonen `leggeTilArray()` umiddelbart, og ingen oppgave blir lagt til i `todos` eller vises i HTML.
- Tømmer input-feltet:** Etter at en oppgave er lagt til, blir input-feltet tømt, så brukeren kan legge til en ny oppgave umiddelbart.

### Fullstendig kode med rettelser:

```
javascript
const btn = document.getElementById("saveTask");
btn.addEventListener("click", leggeTilArray);

function leggeTilArray() {
  // Hent input verdien
  const inputField = document.getElementById("task").value;

  // Sjekk om input er tomt
  if (inputField === "") {
    // Hvis tomt, vis en alert og stopp videre utførelse av funksjonen
    alert("Du må skrive inn en oppgave for å lagre en oppgave!");
    return; // Stopp funksjonen her hvis input er tomt
  }

  // Legg oppgaven til arrayen "todos" og oppdater listen i HTML
  todos.push({ text: inputField, status: "To do" });

  // Oppdater oppgavene i HTML
  let oppgaver = "";
  todos.map((item) => {
    oppgaver += `<li>
      <span class="task-text">${item.text}</span>
      <span class="task-status ${item.status === "Done" ? "task-status-done" : "task-status-pending"}>${item.status}</span>
    </li>`;
  });

  // Skriv ut de oppdaterte oppgavene
  document.getElementById("todolist").innerHTML = oppgaver;

  // Tøm input-feltet etter at oppgaven er lagret
  document.getElementById("task").value = "";
}
```

```
// Skriv ut de oppdaterte oppgavene
document.getElementById("todolist").innerHTML = oppgaver;

// Tøm input-feltet etter at oppgaven er lagret
document.getElementById("task").value = "";
}
```

Nå skal det fungere som forventet: Hvis brukeren prøver å legge til en tom oppgave, vises en alert, og ingen oppgave legges til i listen.

👁️ 🗑️ 🔄 📄 📌 📎

## Koden min før:

```
121 //TODOS
122 const todos = [
123   {text: "Les eksamensteksten", status: "Done"},
124   {text: "Les Innføring i programmerings-eksamen", status: "To do"},
125   {text: "Lag funksjonalitet for å registrere nye oppgaver", status: "To do"},
126   {text: "Lag funksjonalitet for å sette en eksisterende oppgave til utført ved å endre status", status: "To do"}
127 ]
128
129
130 //Oppgave 1. minstekrav: Bytt ut de hardkodede oppgavene (linje 42-49 i exam2.html) med en dynamisk utskrift av oppgavene som ligger lagret i todos:
131
132 //Lager først en tom variabel som får verdiene ifra todos
133 let oppgaver = ""
134 //map() funksjonen går gjennom alle objektene i todos
135 todos.map((item, index) => {
136   //For hvert objekt lagret i todos, legger den til et nytt <li> element "${item.text}" i "todolist" slik at det dannes seg en liste
137   //Inni template literals er det en if/else som sjekker om ${item.status} er satt til "Done" hvis den er "task-status-done", og hvis ikke, er den satt til "To do"
138   oppgaver += `<li>
139     <span class="task-text">${item.text}</span>
140     <span class="task-status ${item.status === "Done" ? "task-status-done" : "task-status-todo"}>${item.status}</span>
141   </li>`
142 })
143 //Linjen under gjør at oppgaven blir skrevet ut i HTML dokumentet
144 document.getElementById("todolist").innerHTML = oppgaver
145 console.log(todos) //Skriver det ut i konsollen
146
147 //Arbeidskrav D, oppgave 1:
148 const btn = document.getElementById("savetask")
149 //Lager en eventlister som lytter etter klikk på knappen "Lagre" og kjører funksjonen "leggeTilArray()" når den blir klikket på
150 btn.addEventListener("click", leggeTilArray)
151 //Lager en funksjon som lagrer oppgaven som blir skrevet inn i inputfeltet "task" i todos arrayen:
152 function leggeTilArray(){
153
154   function inputWarning(){
155     if(document.getElementById("task").value === ""){
156       alert("Du må skrive inn en oppgave for å lagre en oppgave!")
157     } return
158   }
159   inputWarning()
160
161   //Laget en variabel som henter verdien som blir skrevet inn i inputfeltet "task"
162   const skriveFelt = document.getElementById("task").value
163   //Legger til oppgaven som blir skrevet inn i inputfeltet "task" inn i todos arrayen
164   todos.push({text: skriveFelt, status: "To do"})
165   //Skriver ut oppgaven som blir lagt til i todos arrayen i HTML dokumentet
166   oppgaver += `<li>
167     <span class="task-text">${skriveFelt}</span>
168     <span class="task-status task-status-todo">To do</span>
169   </li>`
170   document.getElementById("todolist").innerHTML = oppgaver
171
172
173
174
175
```

## Koden min etter:

```
121 //TODOS
122 const todos = [
123   {text: "Les eksamensteksten", status: "Done"},
124   {text: "Les Innføring i programmerings-eksamen", status: "To do"},
125   {text: "Lag funksjonalitet for å registrere nye oppgaver", status: "To do"},
126   {text: "Lag funksjonalitet for å sette en eksisterende oppgave til utført ved å endre status", status: "To do"}
127 ]
128
129
130 //Oppgave 1. minstekrav: Bytt ut de hardkodede oppgavene (linje 42-49 i exam2.html) med en dynamisk utskrift av oppgavene som ligger lagret i todos:
131
132 //Lager først en tom variabel som får verdiene ifra todos
133 let oppgaver = ""
134 //map() funksjonen går gjennom alle objektene i todos
135 todos.map((item, index) => {
136   //For hvert objekt lagret i todos, legger den til et nytt <li> element "${item.text}" i "todolist" slik at det dannes seg en liste
137   //Inni template literals er det en if/else som sjekker om ${item.status} er satt til "Done" hvis den er "task-status-done", og hvis ikke, er den satt til "To do" ved at det står "task-status-todo"
138   oppgaver += `<li>
139     <span class="task-text">${item.text}</span>
140     <span class="task-status ${item.status === "Done" ? "task-status-done" : "task-status-todo"}>${item.status}</span>
141   </li>`
142 })
143 //Linjen under gjør at oppgaver blir skrevet ut i HTML dokumentet
144 document.getElementById("todolist").innerHTML = oppgaver
145 console.log(todos) //Skriver det ut i konsollen
146
147 //Arbeidskrav D, oppgave 1 og Arbeidskrav C, oppgave 3:
148
149 const btn = document.getElementById("savetask")
150 //Lager en eventlister som lytter etter klikk på knappen "Lagre" og kjører funksjonen "leggeTilArray()" når den blir klikket på
151 btn.addEventListener("click", leggeTilArray)
152 //Lager en funksjon som lagrer oppgaven(e) som blir skrevet inn i inputfeltet "task" i todos arrayen:
153 function leggeTilArray(){
154
155   //Kopiert ifra ChatGPT med noen endringer (fra linje 156 - 180):
156
157   //Deklarerer variabelen "oppgaveTekstFelt" som henter verdien fra inputfeltet "task" og lagrer det.
158   const oppgaveTekstFelt = document.getElementById("task").value
159   //Lager en if-test som varsler brukeren hvis inputfeltet er tomt, og stopper funksjonen i å kjøre, men hvis det er skrevet noe i inputfeltet, kjører den.
160   if(oppgaveTekstFelt === ""){
161     //Her er varselet som blir vist hvis inputfeltet er tomt:
162     alert("Du må skrive inn en oppgave for å lagre en oppgave!")
163     //Her stopper funksjonen i å kjøre videre:
164     return
165   }
166   //pusher inn det som er skrevet i inputfeltet/"oppgaveTekstFelt" inn i todos arrayen, og setter statusen til "To do"
167   todos.push({text: oppgaveTekstFelt, status: "To do"})
168   //legger til en tom variabel som lagrer oppgavene
169   let oppgaver = ""
170   //map går gjennom alle objektene/item i todos arrayen og legger de inn i variabelen "oppgaver"
171   todos.map((item, index) => {
172     oppgaver += `<li>
173       <span class="task-text">${item.text}</span>
174       <span class="task-status ${item.status === "Done" ? "task-status-done" : "task-status-todo"}>${item.status}</span>
175     </li>`
176   })
177   //Skriver ut oppgavene i HTML todolist
178   document.getElementById("todolist").innerHTML = oppgaver
179   //Her blir "oppgaveTekstFelt" / inputfeltet tomt etter at oppgaven er lagret:
180   document.getElementById("task").value = ""
181
182 }
183 //Kopiert ifra ChatGPT med noen endringer, i linjene over (fra linje 156 - 180)
184
```

DEL 3:

Fikk auto fyll av kode i fra Copilot som fungerte ganske bra, men måtte justere den slik at den skulle fungere optimalt:

Koden min før

```
168 //Arbeidskrav C, oppgave 2:
169 //Lager en if test som gjør at "Legg til en farge" knappen blir disabled når det er 10 farger i arrayen og enabled når det er mindre enn 10 farger i arrayen
170 function maxTiFarger(){
171     document.getElementById("")
172     if (randomHex.length >= 10){
173         document.getElementById("addColor").disabled = true
174     } else if (randomHex.length < 10){
175         document.getElementById("addColor").disabled = false
176     }
177 }
178 maxTiFarger()
179
180
181
182
183
184 </script>
185 </body>
186 </html>
187
188
```

Copilot forslag:

```
168 //Arbeidskrav C, oppgave 2:
169 //Lager en if test som gjør at "Legg til en farge" knappen blir disabled når det er 10 farger i arrayen, og enabled når det er mindre enn 10 farger i arrayen
170 function maxTiFarger(){
171     if (randomHex.length >= 10){
172         document.getElementById("addColor").disabled = true
173     } else {
174         document.getElementById("addColor").disabled = false
175     }
176 }
177 document.getElementById("addColor").addEventListener("click", () => {
178     saveHex();
179     maxTiFarger();
180 });
181 document.getElementById("generateAll").addEventListener("click", () => {
182     genererTiNyeFarger();
183     maxTiFarger();
184 });
185 document.getElementById("emptyArray").addEventListener("click", () => {
186     tomFargeArray();
187     maxTiFarger();
188 });
189 maxTiFarger();
190
```

Så at det fungerte, bare at ulempen var at saveHex() var med og det gjør at hver gang jeg trykket "legg till en farge", så la den til to om gangen, da tok jeg vekk saveHex() og koden min ble slik:

```
168 //Arbeidskrav C, oppgave 2:
169 //Lager en if test som gjør at "Legg til en farge" knappen blir disabled når det er 10 farger i arrayen, og enabled når det er mindre enn 10 farger i arrayen
170 function maxTiFarger(){
171     if (randomHex.length >= 10){
172         document.getElementById("addColor").disabled = true
173     } else {
174         document.getElementById("addColor").disabled = false
175     }
176 }
177 document.getElementById("addColor").addEventListener("click", () => {
178     maxTiFarger();
179 });
180 document.getElementById("generateAll").addEventListener("click", () => {
181     genererTiNyeFarger();
182     maxTiFarger();
183 });
184 document.getElementById("emptyArray").addEventListener("click", () => {
185     tomFargeArray();
186     maxTiFarger();
187 });
188 maxTiFarger();
189
```