```elm
module Oppgave exposing (main)

import Browser
import Html exposing (..)
import Html.Attributes exposing (..)
import Html.Events exposing (..)



--- MODEL ---


type alias Model =
    { inputText : String
    , todos : List Todo
    , nextId : Int
    }


type alias Todo =
    { text : String
    , id : Int
    }


initialModel : Model
initialModel =
    { inputText = ""
    , todos = []
    , nextId = 1
    }



--- UPDATE ---


type Msg
    = TextUpdated String
    | TodoAdded


update : Msg -> Model -> Model
update msg model =
    case msg of
        TextUpdated string ->
            { model | inputText = string }

        TodoAdded ->
```

```
50              { model
51                  | inputText = ""
52                  , todos = createTodo model.inputText model.
   nextId :: model.todos
53                  , nextId = model.nextId + 1
54              }
55
56
57 createTodo : String -> Int -> Todo
58 createTodo string id =
59     { text = string
60     , id = id
61     }
62
63
64
65 --- VIEW ---
66
67
68 view : Model -> Html Msg
69 view model =
70     section [ class "todoapp" ]
71         [ header [ class "header" ]
72             [ h1 [] [ text "todos" ]
73             , div [ class "new-todo-row" ]
74                 [ input
75                     [ class "new-todo"
76                     , placeholder "What needs to be done?"
77                     , autofocus True
78                     , autocomplete False
79                     , name "newTodo"
80                     , onInput TextUpdated
81                     , value model.inputText
82                     ]
83                     []
84                 , button
85                     [ class "add-todo"
86                     , onClick TodoAdded
87                     ]
88                     [ text "Add" ]
89                 ]
90             ]
91         , section [ class "main" ]
92             [ ul [ class "todo-list" ]
93                 (List.map viewTodo model.todos)
94             ]
95         ]
96
97
```

```
 98 viewTodo : Todo -> Html Msg
 99 viewTodo todo =
100     li [ classList [ ( "completed", False ) ] ]
101         [ div [ class "view" ]
102             [ input [ class "toggle", type_ "checkbox",
    checked False ] []
103             , label [] [ text todo.text ]
104             , button [ class "destroy" ] []
105             ]
106         ]
107
108
109
110 {--
111 Oppgave 1: Slett todo
112 1. Legg til en Msg som heter TodoDeleted, som tar en Int
    som argument (akkurat som TextUpdated tar en String som
    argument).
113    Int'en skal være id'en til todoen som blir slettet.
114 2. Kompiler koden og les feilmeldingen
115 3. Legg til TodoDeleted i update. I første omgang kan du
    returnere modellen uten å gjøre noen endringer. Sørg for
    at koden kompilerer her.
116    Hint: Pass på at alle casene er like mye indentert, og
    formatter koden med formatteringsknappen i Ellie
117 4. Legg til onClick med en message som argument i button
    med klassen "destroy". Se at meldingen blir sendt, ved å
118    trykke på DEBUG i menyen øverst til høyre.
119    Hint: Du kommer til å måtte bruke parenteser rundt
    argumentet til onClick
120 5. Implementer fjerning av todoen i update.
121    Hint: Du må bruke List.filter og en anonym funksjon.
122    Eksempel: Dette er koden for å filtrere bort 1-tall fra
     en liste med tall: List.filter (\number -> number /= 1) [
    1,2,3,1]
123
124 Oppgave 2: Marker todo som fullført
125 1. Legg til et felt completed som har typen Bool i type
    aliaset Todo.
126 2. Kompiler koden, og les feilmeldingen (les hele
    feilmeldingen, inkludert siste linje)
127 3. Legg til et felt for completed i createTodo. Sett det
    til å være True (vi endrer dette senere)
128 4. Erstatt de to stedene hvor det står False i viewTodo
    med feltet completed til todo. Test at view-koden fungerer
     ved
129    å legge til en todo i appen (ikke i koden) og se at den
     er fullført med en gang man legger til todoen.
130 5. Endre til at completed er False i createTodo
```

```
131 6. Legg til en Msg som heter CompleteToggled som også tar
    en Int som argument
132 7. Legg til CompleteToggled i update. Også her kan du
    først returnere bare modellen uendret, og sjekke at det
    kompilerer.
133 8. Legg til en onClick på input'en som har class "toggle
    ". Sjekk at det blir sendt en message i DEBUG når
    checkboxen blir trykket på.
134 9. Implementer å sette completed på todoen i update.
135    Hint: Du må bruke List.map og en anonym funksjon.
136    Tips: Start med å toggle alle todoene i første omgang,
    og sjekk at det fungerer. Prøv så å endre koden til at kun
137    den todoen som har blitt togglet blir endret.
138    Hint2: Husk at den anonyme funksjonen alltid må
    returnere en todo, men at den ikke nødvendigvis behøver å
    endre hver gang.
139    Hint3: Syntaksen for if else i Elm er sånn her:
140    if a == 1 then
141        "a var visst 1"
142    else
143        "a var noe annet enn 1, jeg gjetter 7"
144
145 --}
146 --- INIT ---
147
148
149 main =
150     Browser.sandbox
151         { init = initialModel
152         , view = view
153         , update = update
154         }
155
```