Opdracht 1: Todo app

Vind jij het ook zo vervelend om te vergeten wat je allemaal nog moet doen? Wij mensen zijn heel goed in het bedenken en uitvoeren van creatieve ideeën, maar niet zo goed in het onthouden ervan. Gelukkig kunnen wij daarvoor de computer gebruiken. In deze opdracht gaan we een todo app bouwen!

Setup

- Als het goed is zit in het mapje wat je ontvangen hebt een html, css en js bestand. Het index.html bestand kun je openen in je browser. Hier gaan wij steeds naartoe om te kijken of ons programma het doet. Het script.js bestand is waarin wij aan het werk gaan.

Knoppen binnen halen

- Haal een knop uit de HTML op in je script.js bestand en sla deze op in een variabele.
- Koppel de knop aan een eventlistener, waardoor je programma een functie uitvoert wanneer je op de knop drukt.

```
addButton.addEventListener("click", addToDoItem);
```

- Maak vervolgens de functie die bij de knop hoort, addToDoItem. In de body van de functie zet je een console.log, zodat je kan zien of de functie werkt.

```
console.log("Het werkt!");
```

- Haal ook de andere twee knoppen binnen, verbind ze aan een eventlistener en maak de bijbehorende functie met een console.log.

Todo items maken

- Nu gaan we aan de slag met de lijst. Haal aller eerst de todo-entry-box en todo-list elementen binnen.
- We willen zo aan het todo-list element todo-items toevoegen. We hebben al een methode addToDoItem gemaakt. Om daadwerkelijk een todoitem te maken hebben we de teskt nodig die we in het vakje getypt hebben. We kunnen die text ophalen door het element aan te spreken. Vervolgens roepen we een nieuwe methode aan waarin we het todoitem toevoegen aan de lijst.

```
function addToDoItem() {
    var itemText = toDoEntryBox.value;
    newToDoItem(itemText);
}
```

- Allereerst moeten we ervoor zorgen dat we data kunnen ontvangen in de newToDoItem functie. Dit doe je door tussen de haken van de functie te zeggen wat je gaat ontvangen. In dit geval noem ik dat itemText.

```
function newToDoItem(itemText) {
}
```

- Wij kunnen in Javascript html elementen aanmaken. In dit geval gaan wij een items in een lijst maken, in html hebben die een li.

```
var toDoItem = document.createElement("li");
```

- Leuk, nu hebben wij een li, maar die is nog niet te zien en die heeft nog geen tekst. Wij gaan eerst een tekst element maken, zodat we die daarna aan de li toe kunnen voegen.

```
var toDoText = document.createTextNode(itemText);
```

- Vervolgens gaan wij dit text element toevoegen aan de li.

```
toDoItem.appendChild(toDoText);
```

- Voeg nu het toDoItem toe aan de toDoList
- Als laatste willen wij ervoor zorgen dat we zomenteen onze todoitems als klaar kunnen merken. Immers, leuk om taken te bij te houden, maar we willen ze ook weg kunnen strepen. Dit doen wij weer met een eventlistener, dit keer de dubbele klik.

```
toDoItem.addEventListener("dblclick", toggleToDoItemState);
```

- De volledige functie ziet er dan als volgt uit

```
function newToDoItem(itemText) {
    var toDoItem = document.createElement("li");
    var toDoText = document.createTextNode(itemText);
    toDoItem.appendChild(toDoText);
    toDoList.appendChild(toDoItem);
    toDoItem.addEventListener("dblclick", toggleToDoItemState);
}
```

Klaar markeren

- We hebben aan alle todoitems een eventlistener gehangen die de toggleToDoItemState functie aanroept. Laten we die functie declareren.

```
function toggleToDoItemState() {
  console.log("Het werkt!");
}
```

- Test de functie door een todo item toe te voegen en er dubbel op te klikken.
- In onze css staat een klasse 'completed'. Wij gaan in de javascript dynamisch deze klasse toevoegen aan de todoitems waarop we dubbel klikken.
- Als je een functie aanroept met een eventlistener heb je standaard toegang tot een argument, namelijk het this argument. In dezen verwijst this dan naar het element waarop de eventlistener wordt aangeroepen. Dat komt voor ons goed uit, want wij willen daar namelijk een nieuwe css klasse aan vast hangen. Dit doen wij met de classlist.add() functie.

```
this.classlist.add("completed");
```

- Nu lopen we echter tegen het probleem aan dat we nu alleen maar een item als klaar kunnen bestempelen, terwijl wij misschien ook een wel een todoitem als niet klaar willen kunnen bestempelen. Daarom moeten we eerst checken of een als klaar is bestempeld. In dat geval gaan wij de completed klasse weghalen. Anders krijgt dit element wel de completed klasse.

```
function toggleToDoItemState() {
    if (this.classList.contains("completed")) {
        this.classList.remove("completed");
    } else {
        this.classList.add("completed");
    }
}
```

Items weghalen

- Op den duur kan je todolijst best vol worden. Om die overzichtelijk te houden willen we graag de items die klaar zijn weggooien. Aan het begin van de tutorial heb je een eventlistener aan een knop gehangen waar clear op staat. We gaan nu die functie implementeren. Gelukkig hebben alle items die klaar zijn de css klasse completed. Dit kunnen we wederom goed gebruiken!
- Wij gaan uit het toDoList element alle items ophalen die de completed klasse hebben.

```
var completedItems = toDoText.getElementsByClassName("completed");
```

- Vervolgens loopen we door die items heen en verwijderen we iedere keer het eerste element. Wij blijven loopen totdat de completedItems lijst een lengte heeft van 0.

```
while (completedItems.length > 0) {
  completedItems.item(0).remove();
}
```

- De hele functie ziet er dan zo uit.

```
function clearCompletedToDoItems() {
    var completedItems = toDoList.getElementsByClassName("completed");

    while (completedItems.length > 0) {
        completedItems.item(0).remove();
    }
}
```

- We zijn er bijna! Soms wil je ook wel eens al je todoitems weggooien. We hebben hiervoor een stuk eerder al een knop gemaakt en hier een eventlistener aan gehangen. Nu gaan we die functie schrijven. Ik heb

deze functie emptyList genoemt. Eigenlijk lijkt deze functie heel erg op de vorige. Wij halen alle elemtenen op die onder de toDoList vallen.

```
var toDoItem = toDoList.children;
```

- Vervolgens loopen we hier doorheen en verwijderen we iedere keer het eerste element totdat de lijst een lengte van 0 heeft.

```
while (toDoItems.length > 0) {
  toDoItems.item(0).remove();
}
```

- De volledige functie is dan:

```
function emptyList() {
    var toDoItems = toDoList.children;
    while (toDoItems.length > 0) {
        toDoItems.item(0).remove();
    }
}
```

Challenge

- Bedenk een manier waarop je de todo items op kan slaan, zodat ze nog bestaan als je de browser nog een keer opent. Tip: Je browser heeft een localstorage waar je het een en ander in op kan slaan.