JAVASCRIPT 3

CoderDojo

CONFIGURATIE

Kaart 1 van 6

Ik leer: JavaScript

Geproduceerd door CoderDojo[kennemerwaard]

1 Deze set Sushi Kaarten leert je hoe je een web-app maakt met een nog-te-doen lijst. Als je op enig moment wilt zien hoe de web-app kan worden, kijk dan op dojo.soy/js-todo. Deze app kun je gebruiken om te volgen wat je maar wilt: gave programmeertrucs, plaatsen om te bezoeken, liedjes om te beluisteren (of leren te spelen!), of zoiets simpels als een boodschappenlijstje.

2 Om te beginnen download je het zip bestand van dojo.soy/a-js-files en plaats je de bestanden in een map op je computer. Hier zul je al het werk voor je web-app uitvoeren.

3 Je hoeft je maar met twee bestanden bezig te houden:

- **index.html** - de webpagina waarop jouw JavaScript uitgevoerd wordt, die je open wilt hebben staan in je browser zodat je de wijzigingen kunt zien terwijl je werkt. Let op - als je je werk hebt opgeslagen, ververs je je browser!
- **js/to-do.js** - Het JavaScript bestand waarin je al je code schrijft

Er zit ook wat **CSS** in, om de pagina er mooi uit te laten zien (en als je de **HTML/CSS** Sushi Kaarten hebt gevolgd weet je hiermee te spelen), maar je **hoeft** het niet te veranderen, ook niet de **jQuery** versie die erbij zit.

4 Dit zijn de **Gevorderden Sushi Kaarten**, dus soms zal ik je dingen vertellen zonder te vertellen hoe je ze moet doen. In die gevallen kun je het namelijk terugvinden bij de **Beginner** of **Tussen** JavaScript Sushi Kaarten.

JAVASCRIPT 3

CoderDojo

CONFIGURATIE

Kaart 1 van 6

Ik leer: JavaScript

Geproduceerd door CoderDojo[kennemerwaard]

5 Je bent inmiddels ook zo ver dat je genoeg JavaScript kent om antwoorden online op te zoeken en daar naar voorbeeldcode te kijken. Ik zal dat nergens in deze kaarten aangeven, maar ik zal wel op wat dingen wijzen die gaaf zijn om eens op te zoeken!

6 Als je klaar bent met deze Sushi Kaarten, werk dan samen met je mentoren om nieuwe projecten te verzinnen. Als je nog niet goed weet wat je wilt maken, kun je ook naar de **HTML/CSS Sushi Kaarten** kijken om nog twee talen te leren die heel goed samengaan met JavaScript.

7 Succes! Veel plezier!

JAVASCRIPT 3

CoderDojo

PROJECT OUTLINE & WIRING BUTTONS

Kaart 2 van 6

Ik leer: JavaScript

Geproduceerd door CoderDojo[kennemerwaard]

1 Voordat we heel diep ingaan op hoe je applicatie gaat werken, moet je eerst alle knoppen verbinden met "dummy" functies, om het testen makkelijker te maken. Hopelijk herinner je je van de **Intermediate JavaScript Sushi Kaarten** dat een JavaScript **functie** er zo uit ziet:

```
function newToDoItem(itemText, completed) {
    console.log("New item created: "+itemText+"
Completed state: "+completed);
}
```

2 Bijna alle **functies** die je nodig zult hebben staan al in de map, maar ze doen nog niets bijzonders. Ze sturen allen **console.log** berichten die je vertellen hoe de **functie** heette en wat de waardes daarbinnen waren. Bekijk ze om te ontdekken wat er allemaal is.

3 Er ontbreekt er één die je zelf zult moeten toevoegen. Maak een nieuwe functie in **to-do.js** die:

- de naam **toggleToDoItemState** heet
- de volgende parameter/waarde accepteert: **listItem**
- Wanneer deze aangeroepen wordt, het bericht **"Toggling state of item "+itemId** opslaat

CoderDojo

PROJECT OUTLINE & WIRING BUTTONS

Kaart 2 van 6

Ik leer: JavaScript

Geproduceerd door CoderDojo[kennemerwaard]

4 Om dingen te verbinden heb je **jQuery** nodig (ook op de Tussen Kaarten). Voeg de volgende code toe aan het eind van **to-do.js**:

```
$(document).ready(function(){
    alert("Ready to go!");
});
```

Laad nu **index.html** in je browser. Je krijgt een signaal als de pagina laadt. Dit komt omdat toen het **document** (index.html) **klaar** was (geladen), de **functie** daarboven (die **alert** bevat) uitgevoerd werd.

5 Nu moet je JavaScript vertellen dat het moet **luisteren** naar een **klik** op de knoppen, om vervolgens iets te doen. Elk van de knoppen op de pagina heeft een unieke **identiteit**. Je kunt dat **ID** gebruiken om **JavaScript** te vertellen waar het naar moet luisteren. Voeg de volgende code toe in plaats van het **alert** in de **document.ready functie**:

```
$("#add-button").click(function(){
    addToDoItem();
});
```

Laad **index.html** opnieuw en klik op de knop **Add** (Toevoegen). Je moet nu een bericht op de **scherm** zien. Doe hetzelfde voor de twee andere knoppen:

- Verbind **#clear-butoon** naar **clearCompleteToDoItems**
- Verbind **#empty-button** naar **emptyList**

JAVASCRIPT 3

CoderDojo

EEN LIJST MAKEN

Kaart 3 van 6

Ik leer: JavaScript

Geproduceerd door CoderDojo[kennemerwaard]

1 Op dit moment is de to-do lijst op de pagina alleen maar wat HTML die ik daar geplaatst heb zodat je iets hebt om naar te kijken! Tijd om het van jou te maken: Ga naar **index.html** en verwijder de vier regels die beginnen met **li** tags. Als er niets veranderd is, zijn dat de regels 23-26. Laadt dan opnieuw **index.html** in je browser. Het ziet er een beetje kaal uit, maar dat ga je straks aanpassen!

2 Roep binnen de **document.ready** functie, de **loadList** functie op. Laadt **index.html** opnieuw om dat te checken. Als je het log ziet op je **scherm** dan ben je klaar om te beginnnen!

3 Nu moet je ervoor zorgan dat de **loadList** functie ook echt een lijst laadt! Nu is het nog een soort "demo list"". Later laat ik je zien hoe je een to-do lijst opslaat en laadt als je de pagina heropent (daar moet je dan wel online voor zijn). Dus verander de inhoud van de **loadList functie** naar:

```
toDoItems = [
  {
    "text": "My",
    "completed": false
  },
  {
    "text": "to-do",
    "completed": true
  },
  {
    "text": "list",
    "completed": true
  }
];
```

CoderDojo

EEN LIJST MAKEN

Kaart 3 van 6

Ik leer: JavaScript

Geproduceerd door CoderDojo[kennemerwaard]

4 Je ziet dat deze drie items elk hun **text** en **complete** status hebben en dat ze in een **array** staan. Je moet ze echter nog op de pagina zetten. Binnen de **getToDoItemHTML** functie voeg je dit stukje code toe:

```
var itemHTML = $("<li>"+toDoItem.text+"</li>");
    if(toDoItem.completed === true){
    $(itemHTML).addClass("completed");
}
return itemHTML;
```

Het gebruikt een **if** uitdrukking, die een **jQuery** object maakt van HTML en **als** de **toDoItem** compleet is, de "compleet" **class** toevoegt voor de opmaak.

5 Nu kun je HTML maken voor een lijstitem en moet je het aan de lijst toevoegen. Voeg dit toe aan de **loadList functie**, achter de code van stap 3:

```
toDoItems.forEach(function(toDoItem){
    var itemHTML = getToDoItemHTML(toDoItem);
    $(".todo-list").append(itemHTML);
});
```

Deze code gebruikt **forEach**, dat elk item in de **array** doorloopt en het **appends** (toevoegt) aan de to-do lijst, welke je selecteert via de **class** naam.

JAVASCRIPT 3

CoderDojo

ITEM AAN LIJST TOEVOEGEN

Kaart 4 van 6

Ik leer: JavaScript

Geproduceerd door CoderDojo[kennemerwaard]

1 Dus nu werkt je to-do lijst, maar de inhoud blijft hetzelfde. Dat kun je aanpassen! De "Add" knop roept al een **functie** aan, je moet alleen die **functie** nog iets laten doen. Ga naar je JavaScript bestand en verander de code in **addToDoItem** om de tekst uit het tekstveld te halen (genaamd **"new-todo"**) en update het log om zeker te weten dat je het snapt:

```
var itemText = $("input[name=new-todo]").val();
console.log("Item added: "+itemText);
```

val() staat voor value (waarde). Typ iets in het tekstveld en klik op de "Add" knop. Je moet wat je typt op je **scherm** zien.

2 De volgende stap is om die tekst op de to-do lijst te krijgen. Daarvoor heb je een **functie** nodig die de HTML pakt voor het nieuwe to-do item en het toevoegt aan het eind van de to-do lijst Dus verander **newToDoItem** om deze code erin te krijgen (de opmerkingen heb je niet nodig!):

```
var newItem = {
        "text": itemText,
        "completed": completed
}; // maak van de parameters een toDoItem
toDoItems.push(newItem); //zet newItem aan het eind
van toDoItems
var itemHTML = getToDoItemHTML(newItem);
$(".todo-list").append(itemHTML);
```

CoderDojo

ITEM AAN LIJST TOEVOEGEN

Kaart 4 van 6

Ik leer: JavaScript

Geproduceerd door CoderDojo[kennemerwaard]

3 Nu ga je terug naar **addToDoItem** en verandert het weer door het log te verwijderen (als je wilt) en het **newToDoItm** zo aan te roepen:

```
newToDoItem(itemText,false);
```

Val niet in herhaling!

Je kunt ook een **nieuwToDoItem** in **loadList** gebruiken om de hoeveelheid code waar je mee werkt te verminderen. Je kunt alle code in de functie hierdoor vervangen:

```
newToDoItem("My",false);
newToDoItem("to-do",true);
newToDoItem("list",false);
```

Je bent nu van 17 regels code naar 3 regels gegaan en dat is de kracht van functies! Probeer te begrijpen hoe dit werkt en hoe het exact hetzelfde doet als de oude versie van **loadList**.

CoderDojo

ITEM ALS COMPLEET MARKEREN

Kaart 5 van 6

Ik leer: JavaScript

Geproduceerd door CoderDojo[kennemerwaard]

1

Nu kun je je to-do lijst laden en items toevoegen, maar wat heeft dat voor zin als je ze niet kunt afvinken als je ze gedaan hebt? Dat ga je nu leren, en het is vrij makkelijk. Eerst moet je dubbelkikken op een lijst item (**li** tag) verbinden aan je **toggleToDoItemState** functie. Dit kan door **jQuery** te laten luisteren naar dubbelkliks op **li** tags en een functie uit laten voeren die je**toggleToDoItemState** functie aanroept. Het ziet er zo uit:

```
$(document).on("dblclick","li", function(){
    toggleToDoItemState(this);
});
```

Wat hier gebeurt is dit:

- jQuery luistert in de hele pagina naar **dblclick** (dubbelklik) op **li** (lijst item) tags.
- Als jQuery een dubbelklik hoort op een lijst item, dan voert het **toggleToDoItemState** uit en geeft het **this**, een speciaal **sleutelwoord**

Dus waar gaat this nou over?

In JavaScript is **this** een **sleutelwoord**, een woord met speciale betekenis. Hoe dat precies werkt hangt af van waar het gebruikt wordt (je kunt meer hierover online vinden - dojo.soy/a-js-this) maar in dit geval betekent **this** lijst item, degene waarop dubbelgeklikt werd.

CoderDojo

ITEM ALS COMPLEET MARKEREN

Kaart 5 van 6
Ik leer: JavaScript

Geproduceerd door CoderDojo[kennemerwaard]

2 Nu moet je de **toggleToDoItemState** functie updaten zodat deze twee dingen doet:

- het vernieuwt de array van to-do items.
- het voegt **completed** class toe aan het item waarop geklikt is. Eerst update je de array door de code in **toggleToDoItemState** hiermee te vervangen:

```
var itemId = $(listItem).index();
toDoItems[itemId].completed =
!toDoItems[itemId].completed;
```

De eerste regel gebruikt jQuery om de **index** van de **li** binnen de geordende lijst (**ol**) te krijgen. De **index** is het nummer dat de positie aangeeft en, aangezien computers vanaf 0 tellen, het één lager is dan het nummer dat te zien is op de webpagina.

De tweede regel zoekt dat item op in de array (de **index** is dezelfde als op de lijst, omdat we geen items rondzeulen) en maakt de **completed property** het tegenovergestelde van de huidige waarde. Door **!** voor een **true**/**false** (waar/niet waar) waarde te zetten verandert het in het tegenovergestelde.

3 Om de class te veranderen en een doorgestreept/afgevinkt item op de lijst te krijgen, hoef je maar één regel toe te voegen aan het eind van **toggleToDoItemState**:

```
$(listItem).toggleClass("completed");
```

CoderDojo

DE LIJST OPSCHONEN

Kaart 6 van 6

Ik leer: JavaScript

Geproduceerd door CoderDojo[kennemerwaard]

1 Als je gebruiker de to-do lijst wil opschonen (stel dat ze ermee gespeeld hebben en een hoop nep dingen hebben toegevoegd), dan kun je ze dat laten doen. Dat is zelfs vrij eenvoudig! Je hoeft alleen de to-do array in JavaScript en de geordende lijst in de HTML te legen. Je kunt de array legen door de **emptyList** functie het volgende te laten zijn:

```
toDoItems = [];
```

Dat is het! **toDoItems** is nu een lege array!

2 Om de **ol** in de HTML te legen is niet veel lastiger. Voeg de volgende code toe aan het eind van je **emptyList** functie:

```
$(".todo-list").children().remove();
```

Dit vertelt jQuery om:

- alle tags te selecteren met de class **todo-list**. In jouw geval is er maar één, de **ol**. Wees hier voorzichtig mee als je de class op meerdere plekken hebt gebruikt.
- Selecteer nu alle **kinderen** van die tag. Dat betekent: elke tag daarbinnen, in dit geval al je **li** tags.
- **Verwijder** alle tags die het geselecteerd heeft van de pagina.

CoderDojo

DE LIJST OPSCHONEN

Kaart 6 van 6

Ik leer: JavaScript

Geproduceerd door CoderDojo[kennemerwaard]

3 Hoe zit het nu met de afgevinkte items? Het wordt een zooitje als je de gebruikers niet dat ook laat opschonen. Begin met het verwijderen van afgevinkte items van de array. Update **cleanupCompletedToDoItems** als volgt:

```
toDoItems = toDoItems.filter(function(toDoItem){
    return !toDoItem.completed;
});
```

Deze code neemt de **toDoItems** en haalt er een **filter** overheen. Alleen **toDoItems** die een **waar** waarde aangeven via de filterfunctie zullen doorgaan en in de nieuwe **array** zitten die gemaakt wordt (welke jij weer terugzet in **toDoItems**). Door de **afgevinkte** waarde van elk **toDoItem** te nemen en **waar** naar **niet waar** te veranderen en **niet waar** in **waar** door **!**, krijg je alleen die items die nog niet af zijn.

4 Tenslotte verwijder je afgevinkte items van de lijst op de HTML pagina door één regel code toe te voegen aan je **cleanupCompleteToDoItems** functie:

```
$(".todo-list").find(".completed").remove();
```

Dit doet iets soortgelijks als de code in punt 2, behalve dat je niet alle kinderen van de **.todo-list** selecteert, je **vind** slechts degene met de **completed** class en **verwijdert** ze.