Building a To-Do List

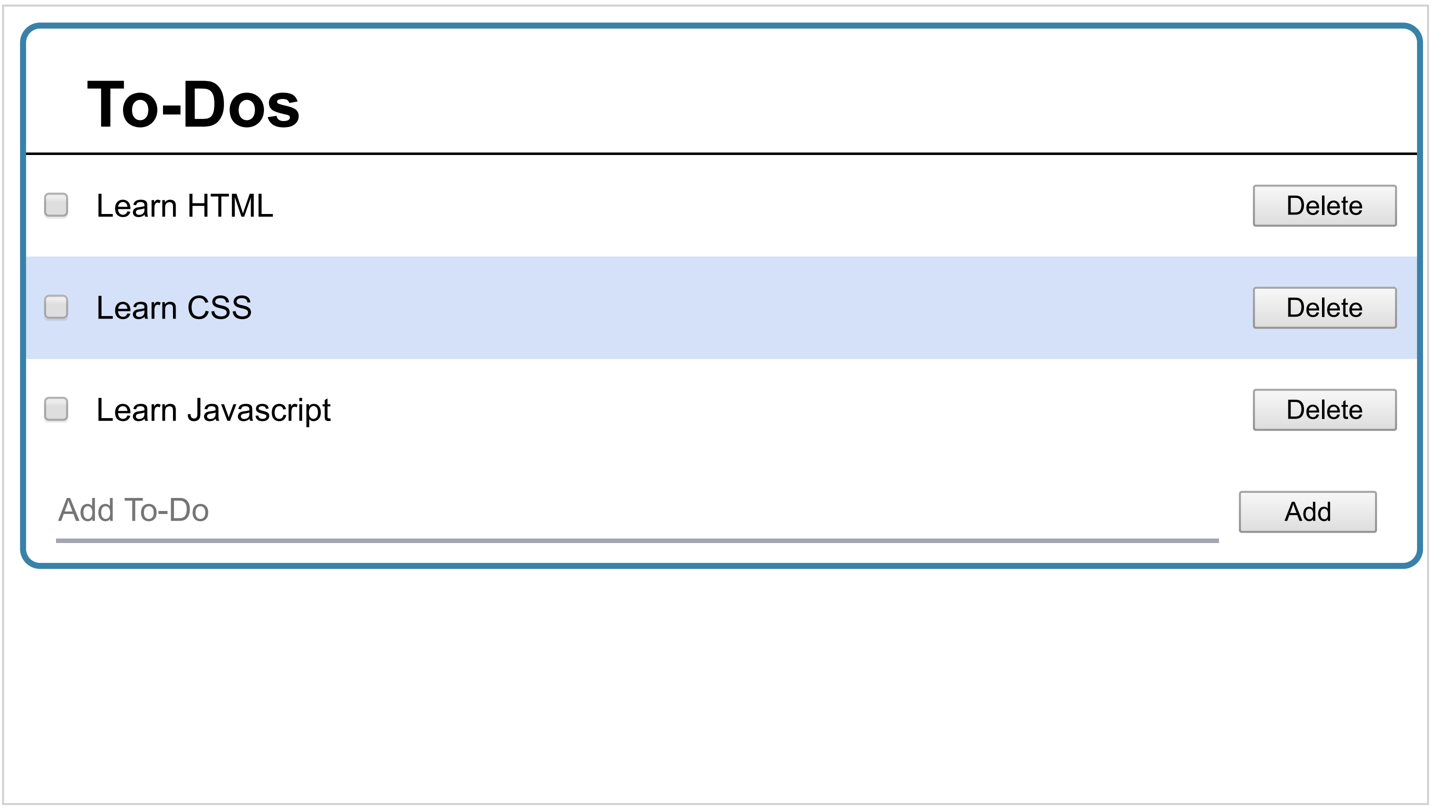
Creating a Fully Functional Application Using HTML, CSS, and Javascript

We'll cover the following

Overview

In this final lesson, we’ll be using the DOM manipulation techniques you’ve learned to implement a fully-functional To-Do list application.

Like the previous lesson, we’ll start by setting up the to-do list with HTML and CSS and then use Javascript to implement the functionality through interactive exercises. Hope you enjoy!


To-Dos

Learn HTML

Learn CSS

Learn Javascript

Add To-Do

Setup

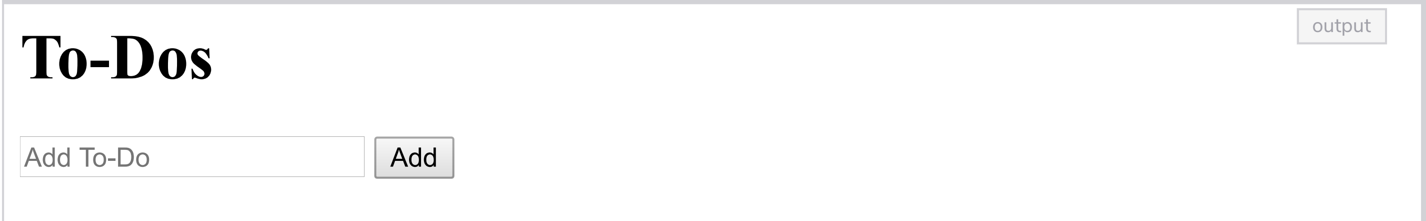
Let's first give some structure to the To-Do list with HTML:

HTML

html

```html
<html>
 <head>
   <title>Todo List</title>
 </head>
 <body>
   <div id="todos">
     <h1>To-Dos</h1>
     <ul id="todoList">
     </ul>
     <div class="addTodo">
       <input type=text placeholder="Add To-Do" id="newTodo">
       <button id="add">Add</button>
     </div>
   </div>
 </body>
</html>
```

output


To-Dos

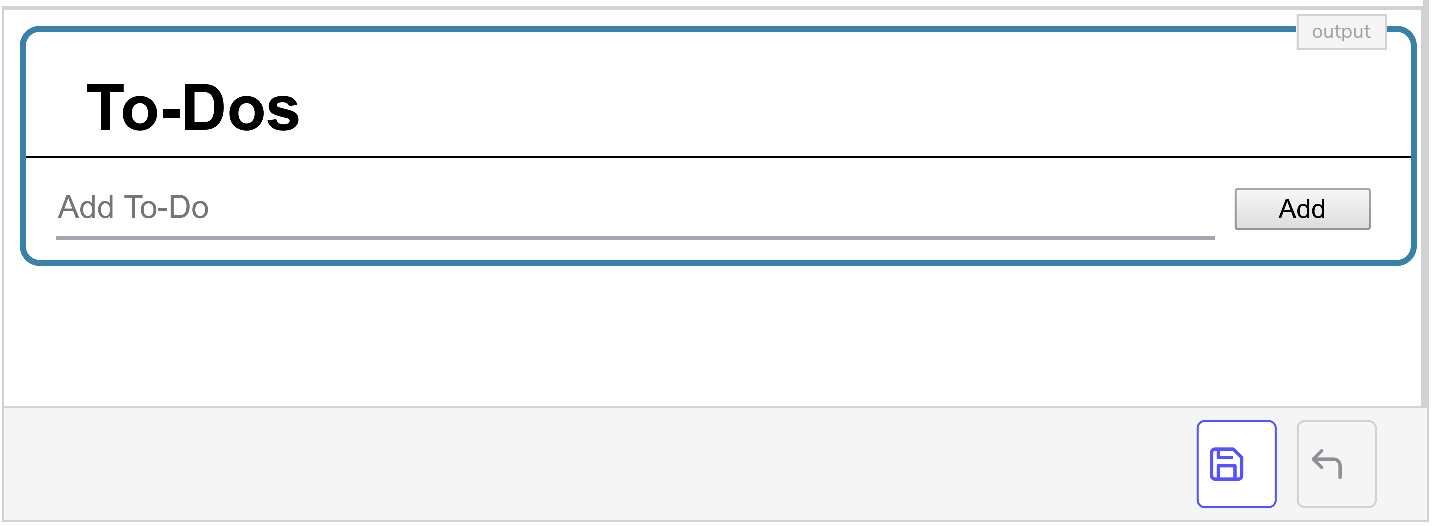
The contents of the To-Do application are wrapped in a `<div>` **container**, with the id `todos`. The actual list is stored in an **unordered list** with the id `todoList`.

Additionally, we create another `<div>` container that will store a text input that will allow us to add new to-do items to our list.

We'll also go ahead and add some CSS to style the list. Don't worry too much about understanding every aspect of the CSS, as we're going to be more focused on making the list functional through Javascript.

CSS

```css
#todos {
  border: 3px solid rgba(57,129,169,1);
  border-radius: 10px;
  width: 100%;

  font-family: "Helvetica Neue", Helvetica, Arial, sans-serif;
}

#todos > h1 {
  margin: 10px 0px 0px 0px;
  padding: 10px 20px 5px 30px;
  border-bottom: 1px solid black;
}

.addTodo {
  padding: 10px 15px;
}

#todoList {
  padding: 0px;
  margin: 0px;
}

#todoList > li {
  display: flex;
  align-items: center;

  padding: 10px 5px;
  margin: 0px;
  list-style: none;
}
```



Exercise: Implement `createTodo(todo)`

Let’s first start by writing a Javascript function that **returns** a `<li>` element housing the contents of a To-Do item.

The `<li>` should be structured like so:

```
<li>
  <input type="checkbox">
  <label>Learn CSS</label>
  <button class="delete">Delete</button>
</li>
```

The `<li>`'s child elements should have the following characteristics:

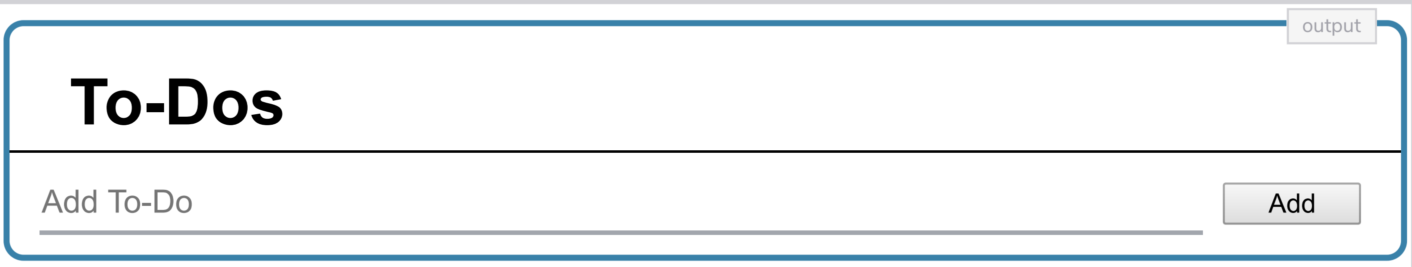
- a checkbox `<input>`
- a `<label>` element that **stores the to-do** *passed in as an argument* to the function
- a “Delete” `<button>` with the class `delete`

Hide Hint

Use the `innerHTML` property to pass the `todo` argument to the appropriate element. You can create an element using `document.createElement()` and append elements to another element using `Element.appendChild()`.

HTML CSS JavaScript

```javascript
var createTodo = function(todo) {
  // write your code here;
}
```



If you passed the tests, you now have a function that creates a new To-Do item! Now, let’s go ahead and tie this function to an event listener.

Exercise

Use your `createTodo` function within an **event listener** to create a new To-Do item using the “Add Todo” text input.

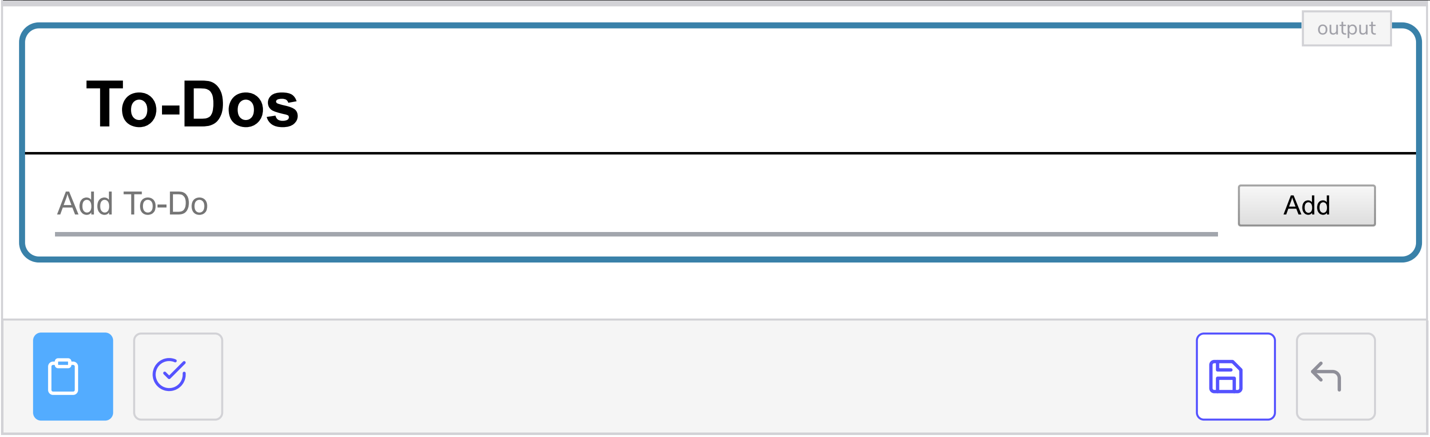
- the event listener should only create a new To-Do if the input’s value **is not empty**
- the event listener should place the newly created to-do within the **unordered list** with the id `todoList`

```html
<html>
 <head>
   <title>Todo List</title>
 </head>
 <body>
   <div id="todos">
     <h1>To-Dos</h1>
     <ul id="todoList">
     </ul>
     <div class="addTodo">
       <input type=text placeholder="Add To-Do" id="newTodo">
       <button id="add">Add</button>
     </div>
   </div>
 </body>
</html>
```

HTML CSS JavaScript

```javascript
var createTodo = function(todo) {
  // copy your createTodo() code here
}

```

```
var addButton = document.getElementById('add');

addButton.onclick = function() {
  // store the button's parent element (.addTodo <div>) in a variable
  var parent = this.parentNode;
  // store the input, which is the *first* child element of the .addTodo <div>
  var input = parent.children[0];

  // write your event listener code here
    // get the input's value
    // if input isn't empty, create a new element and add it to the unordered list
}
```



If you passed the tests, your to-do list should now add new items! Now let's implement a way to indicate which To-Do items have been completed.

Exercise: Implement to-do completion

If you take a closer look at the CSS, you'll notice there is a class, `checked`, that **places a line through** a label's text.

```
.checked > label {
  text-decoration: line-through;
}
```

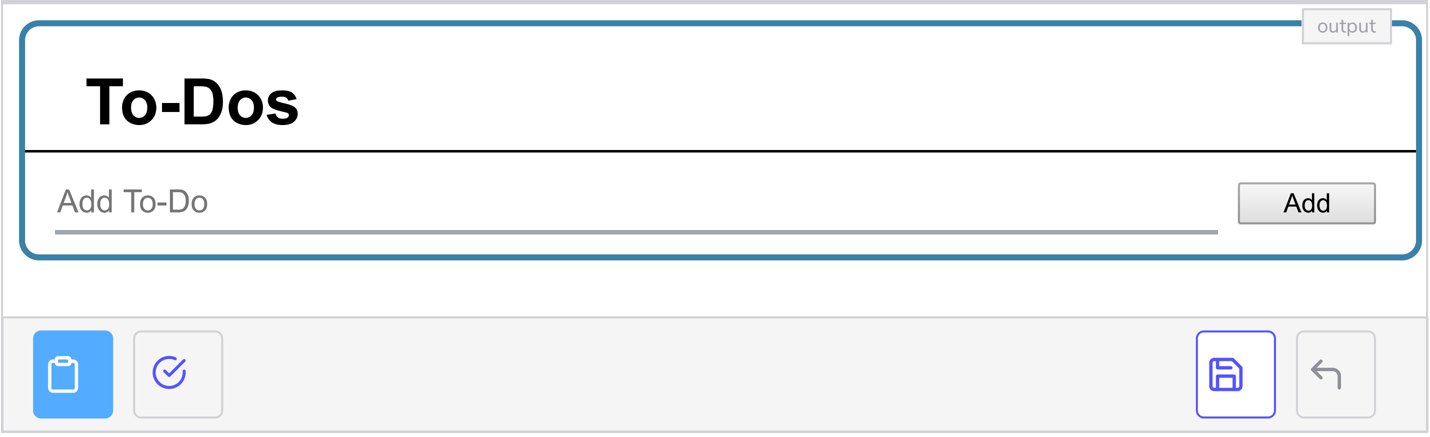
Within your `createTodo` function, **add an `onchange` event listener** to the checkbox `<input>` element:

- when an input changes, use the `classList` property to **toggle the class `checked`** on the **`<li>` parent element**

By adding the event listener **within the `createTodo()`** function, each newly added item will have the same functionality. Good luck!

HTML CSS JavaScript

```javascript
var createTodo = function(todo) {
  // copy your createTodo() code here

  // create an event listener on <input> elements
}

document.getElementById('add').onclick = function() {
  // copy your add button event listener code here
}
```



The To-Do list should now strike a line through items when they are checked! Finally, let’s add the ability to delete To-Do items we no longer want in our list.

Exercise: Implement deleting functionality

Within your `createTodo` function, **add an `onclick` event listener** to the “Delete” `<button>` element:

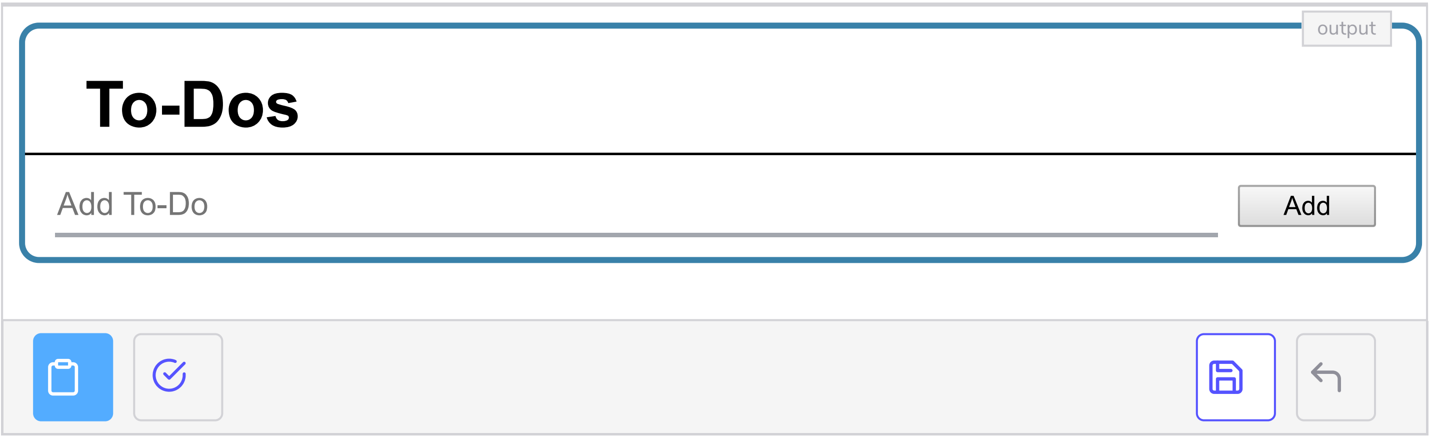
- when a “Delete” `<button>` is clicked, remove **the entire list item** associated with that button.

Hint: the list item is the **parent element** of the `<button>`

Like with the previous exercise, by adding the event listener **within the `createTodo()`** function, each newly added item will have the same functionality. Good luck!

HTML CSS JavaScript

```javascript
var createTodo = function(todo) {
  // copy your previous createTodo() code here

  // create an event listener for the "Delete" <button> element that removes the entire lis
}

document.getElementById('add').onclick = function() {
  // copy your add button event listener code here
}
```



Congratulations!

If you’ve made it this far, you’ve successfully implemented a To-Do list application in the browser.

You’ve also reached the very end of this course! You should now feel ready to pick up more challenging concepts related to developing applications for the web.

We hope you had an engaging experience and are feeling more comfortable with introductory web development concepts. The goal of this course was to focus on **actually using** the concepts we discussed and we sincerely hope you received value from putting things into practice through the exercises.