**How to Build a Basic To-Do List App Using JavaScript**

**BY**[**UNNATI BAMANIA**](https://www.makeuseof.com/author/unnati-bamania/)

PUBLISHED SEP 01, 2021

One of the better project ideas for JavaScript beginners is to create a simple to-do list app. Here's our own tutorial for JS newbies.

The Document Object Model (DOM) is the data representation of the objects that comprise the structure and content of a web page. You can access all the DOM elements on the website and dynamically create, read, update, and delete (CRUD) them using JavaScript.

This article will explain how you can perform CRUD operations on a to-do list using JavaScript and DOM manipulation. We expect you to know the basics of HTML and JavaScript before going through this article.

**Understanding Basic DOM Manipulation**

Let’s go through a simple example:

<button id=”submit”>Submit</button>  
<script>  
const submitButton = document.getElementById(“submit”);  
submitButton.addEventListener(“click”, ()=>{  
alert(“The form has been submitted”);  
});  
</script>

The **submitButton** variable has access to the HTML button in the above code. You have to add the **click** event listener on the button (by getting the element by its id of **submit**). When the button is clicked, the event is triggered,and the window displays a pop-up with the text: "The form has been submitted."

Now that we’ve covered the basic idea of [**DOM manipulation**](https://www.makeuseof.com/what-is-the-dom/), let’s proceed forward and dive into building the to-do app.

**Building the Layout Using HTML and TailwindCSS**

Let’s have a look at the HTML layout of this project. The input elements and the buttons have their respective ids in order to get access to these elements in the JavaScript file.

For the frontend design, this article uses **[TailwindCSS](https://tailwindcss.com/)**, a utility CSS framework. You can use TailwindCSS in your project by importing the CSS file from the CDN.

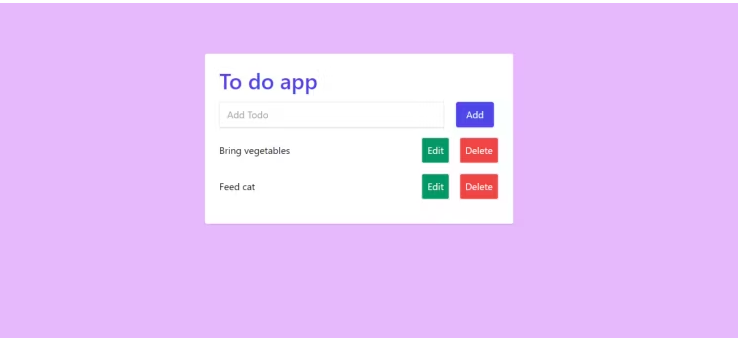
<link href="[https://unpkg.com/tailwindcss@^2/dist/tailwind.min.css](https://unpkg.com/tailwindcss@%5E2/dist/tailwind.min.css)" rel="stylesheet" />

**RELATED:**[**Tailwind CSS Vs. Bootstrap: Which Is A Better Framework?**](https://www.makeuseof.com/tailwind-css-vs-bootstrap-which-is-a-better-framework/)

**Code:**

<div class="h-100 w-full flex items-center justify-center bg-teal-lightest font-sans mt-20">  
 <div class="bg-white rounded shadow p-6 m-4 w-full lg:w-3/4 lg:max-w-lg">  
 <div class="mb-4">  
 <h1 class="text-3xl md:text-4xl text-indigo-600 font-medium mb-2">  
 To-Do List App  
 </h1>  
 <div class="flex mt-4">  
 <input class="shadow appearance-none border rounded w-full py-2 px-3 mr-4 text-grey-darker" name="text" id="text" placeholder="Add Todo" />  
 <input type="hidden" id="saveIndex" />  
 <button class="p-2 lg:px-4 md:mx-2 text-center border border-solid border-indigo-600 rounded text-white bg-indigo-600 transition-colors duration-300 mt-1 md:mt-0 md:ml-1" id="add-task-btn">Add</button>  
 <button class="p-2 lg:px-4 md:mx-2 text-center border border-solid border-indigo-600 rounded bg-indigo-600 text-white transition-colors duration-300 mt-1 md:mt-0 md:ml-1" style="display: none" id="save-todo-btn">Edit Todo</button>  
 </div>  
 </div>  
 <div id="listBox"></div>  
 </div>  
 </div>

This is how our app looks after designing:



**Adding Functionality With Javascript:**

The first step is getting access to the elements by their ids using the method **getElementById()**.

const text = document.getElementById("text");  
const addTaskButton = document.getElementById("add-task-btn");  
const saveTaskButton = document.getElementById("save-todo-btn");  
const listBox = document.getElementById("listBox");  
const saveInd = document.getElementById("saveIndex");

We need an array to store all the to-do tasks. Hence, we need to initialize one.

let todoArray = [];

**Adding Items to the To-Do List**

To add a task to the array, you need to push it to the **todoArray** and then display it on the webpage. For this to happen, a click event must be triggered on the add button.

addTaskButton.addEventListener("click", (e) => {  
 e.preventDefault();  
 let todo = localStorage.getItem("todo");  
 if (todo === null) {  
 todoArray = [];  
 } else {  
 todoArray = JSON.parse(todo);  
 }  
 todoArray.push(text.value);  
 text.value = "";  
 localStorage.setItem("todo", JSON.stringify(todoArray));  
 displayTodo();  
});

You have to store the **todoArray** to the **localStorage** on every change (i.e. whenever a task is added, updated, or deleted).

In the above code, you have to fetch the array from the **localStorage**; if no array exists, we create a blank one. Then we push the newly added task to the **todoArray** and store the whole array again in **localStorage**.

**Displaying the To-Do List Changes**

After appending the value to the **todoArray**, you need to display it on the web page. This is done by using **.innerHTML**attribute.

We put the HTML for the to-do list inside a variable named **htmlCode**. Then, we loop through the **todoArray** and add each item to the **htmlCode** variable.

Once you are done looping through all the items, you need to assign the whole HTML code to the **listBox** element using the **.innerHTML** attribute.

So after pushing the new to-do list item to the array, we call the **displayTodo()** function which handles all of that as described:

function displayTodo() {  
 let todo = localStorage.getItem("todo");  
 if (todo === null) {  
 todoArray = [];  
 } else {  
 todoArray = JSON.parse(todo);  
 }  
 let htmlCode = "";  
 todoArray.forEach((list, ind) => {  
 htmlCode += `<div class='flex mb-4 items-center'>  
 <p class='w-full text-grey-darkest'>${list}</p>  
 <button onclick='edit(${ind})' class='flex-no-shrink p-2 ml-4 mr-2 border-2 rounded text-white text-grey bg-green-600'>Edit</button>  
 <button onclick='deleteTodo(${ind})' class='flex-no-shrink p-2 ml-2 border-2 rounded text-white bg-red-500'>Delete</button>  
</div>`;  
 });  
 listBox.innerHTML = htmlCode;  
}

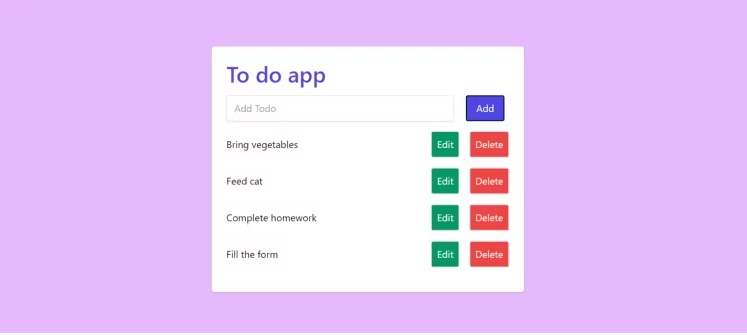
You have to add two buttons—update and delete—for each item while appending the todo items to the variable **htmlCode**.

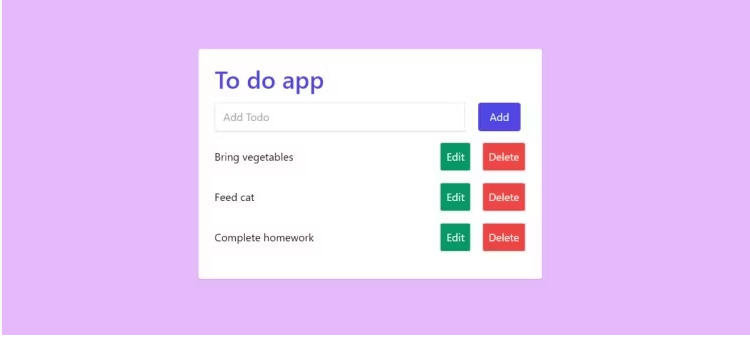
**Deleting Items From the To-Do List**

The delete button has an attribute method **onclick()** that passes the todo index as the parameter. On clicking the delete button, the **deleteTodo()**method will be executed.

In this method, you have to apply the **splice()** [**array method**](https://www.makeuseof.com/tag/javascript-array-methods/) on the **todoArray**. The **splice()** method helps to delete the item at the specified index. After deleting the item, you have to store the changes to the **localStorage** and call the **displayTodo()** function to reflect changes on the web page.

function deleteTodo(ind) {  
 let todo = localStorage.getItem("todo");  
 todoArray = JSON.parse(todo);  
 todoArray.splice(ind, 1);  
 localStorage.setItem("todo", JSON.stringify(todoArray));  
 displayTodo();  
}

****

****

**Updating Items in the To-Do List**

Each to-do list item has an edit button, just like the delete button. The edit button has an attribute method **onclick()**. On clicking the button, the edit method gets executed and passes the index as the parameter.

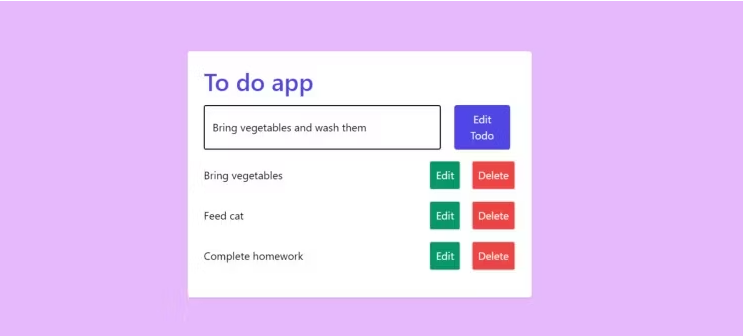
There are two HTML elements whose display properties are set to **none:**

1. Input element with id **saveIndex**
2. Button with the id **save-task-btn**

As soon as you click on the edit button, the input will have the text value that you want to update with. The **saveTaskButton** will be displayed instead of **addTaskButton**.

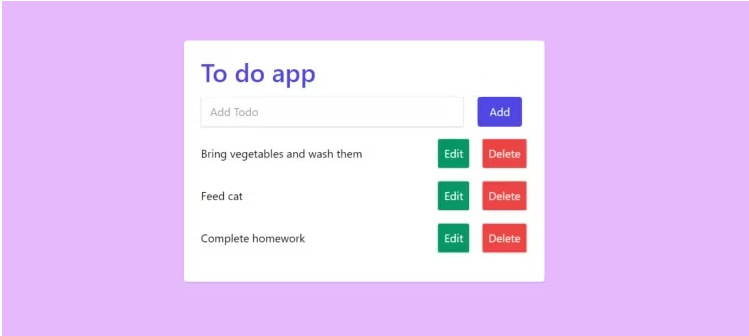
The HTML code consists of an input element with id **saveIndex**. You have to set its default style property of display as **none**. When the edit method is called, you set the value attribute of this element to the id, so you can reference it later when saving the updated task.

function edit(ind) {  
 saveInd.value = ind;  
 let todo = localStorage.getItem("todo");  
 todoArray = JSON.parse(todo);  
 text.value = todoArray[ind];  
 addTaskButton.style.display = "none";  
 saveTaskButton.style.display = "block";  
}



Once you are done editing the text, you click on the **saveTaskButton**. On clicking the button, you retrieve the id of the text using the **saveInd** input. After retrieving the id, you can update the **todoArray** at that index and push the changes to the **localStorage**. Finally, we called the **displayTodo()** function to reflect changes on the web page.

saveTaskButton.addEventListener("click", () => {  
 let todo = localStorage.getItem("todo");  
 todoArray = JSON.parse(todo);  
 let id = saveInd.value;  
 todoArray[id] = text.value;  
 addTaskButton.style.display = "block";  
 saveTaskButton.style.display = "none";  
 text.value = "";  
 localStorage.setItem("todo", JSON.stringify(todoArray));  
 displayTodo();  
});

****

**Check One Item Off Your To-Do List**

Now that you have completed the basic to-do list app, it's time for you to start building more exciting projects by yourself!

You can either build a game or a web application that you can use for your personal use. Building projects will help you build your skills and get a good grasp of JavaScript. Keep learning and building amazing projects as much as you can.

Want to build another JavaScript project? Here's a simple calculator web application that you can build with HTML, CSS, and JavaScript.