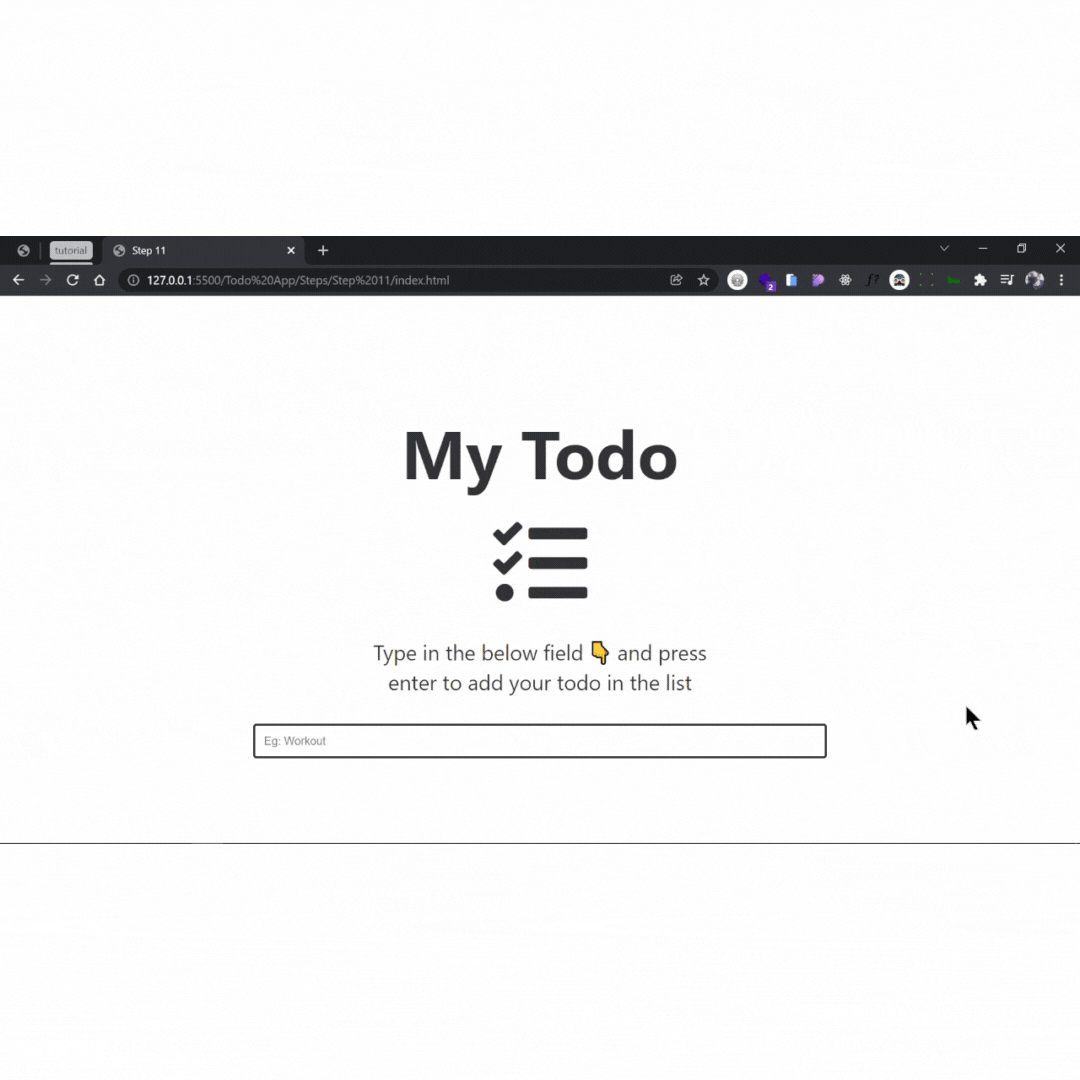
**TODO APP USING HTML**

**CSS & JAVASCRIPT**



**Table of contents**

[**Introduction**](#_otbqekhtbgfw) **3**

[**Prerequisites**](#_iuyilzf31cqd) **3**

[**Get started ✌**](#_m7sx9e88ridd) **3**

[**First let’s create the skeleton of the ui**](#_yl8sgls94iyh) **4**

[Add font awesome cdn 👇](#_fpwdx8vc8u8z) 4

[Markup of todo list , default todo , add todo form 👇](#_5bn6csgzdifu) 4

[Output of above codes 👉](#_4nnfqvq5rhyn) 4

[**Let’s add styling into ui 🎨**](#_bxmja4t4zog6) **5**

[Basic styling](#_wv1w9y9pt6et) 5

[Adding styling into container, svg button , todo list , todo item,](#_7pgzwfdtw1oz) 6

[todo item span , checked button](#_hbgrin50s988) 6

[Adding styling into delete todo button](#_n8hlpfcpd2od) 7

[Adding styling into add todo form](#_leguliqiwjw) 8

[Styling for todo complete effect checked ( we gonna use this in javascript )](#_kptg0vll0wo5) 9

[Styling done 🎨](#_26fd69ype4mf) 9

[**Let’s start building feature for add new todo**](#_rkfr339i61x0) **10**

[**and render todo in the list , store todo as a**](#_6a6zre73e26j) **10**

[**array object to localstorage**](#_nkun91gnzzsx) **10**

[Function for adding new todo](#_6st1hzrkw2s8) 10

[Add a event listener submit to the form](#_34qnbqnm1kns) 10

[Function for rendering new todo in display](#_9vw3ln4ztu4z) 11

[Select todo list for appending all todos in list](#_tasb6mzc112j) 12

[Select the current todo item in the DOM ( We’ll use](#_5kxbg7vhhqa8) 12

[item for deleting todo from todo list later in this tutorial )](#_hi87z8jav6f7) 12

[Check if checked is true add done class effect otherwise as it is](#_l7xx54a8cxch) 12

[Create new element li for todo ui](#_g8rr51wqjhm6) 12

[Save todo data in browser local storage](#_s42vynv1ezfk) 15

[Creating empty todoArray array object](#_4qh4g28c2207) 15

[Set todoArray to localstorage](#_lg2wqvt2y2l6) 16

[Push new todoobject to todoArray array object](#_26mz9k691kj9) 16

[Get todoArray from localstorage to display todos in todo list](#_1mcgzpx1wm3f) 16

[Let’s build functionality for marking todo done and delete todo](#_6jpi037sifxx) 17

[Function for marking todo complete ( in toggleDone function we’re going to )](#_91qdvkil4nix) 17

[Call toggleDone function](#_pkb1afbfjsta) 18

[Function for delete todo ( in deleteTodo function we’re going to )](#_9zgik6uxjes9) 20

[Call deleteTodo function ( Similarly how we called toggleDone() )](#_6eksj11y2w6z) 21

[Let’s add empty state when there is no todo to display ( optional )](#_s4n0ib868bbf) 22

[Markup of empty state ( add this markup below todo list markup in html )](#_to4ot8pmv8ys) 23

[when there is no todo in todo list then add .empty-state and make display to flex ( add this below the other styles )](#_pmbyxqcv1r0h) 23

[Styling for empty state .empty-state by default is none](#_fgyob0201r10) 23

[( add this below the other styles )](#_wvtzgbfrfkt0) 23

[**Congratulation we have**](#_8ev7eztplod2) **25**

[**successfully created todo app 🥳**](#_tda0ti3fpqgo) **25**

[**— The End —**](#_m867phng5i1a) **25**

[Was it useful 🤔 let me know on instagram](#_oz6ikekx08j5) 25

[— Get Source Code Here —](#_baaiaxup9zvb) 25

[— Get Source with code comment —](#_a9pk67wm6ojd) 25

# 

# **Introduction**

In this tutorial we’re going to learn how to create a cool **todo app**

using *html css and javascript* with the *dark light mode* functionality

# **Prerequisites**

# 

# 

# **Get started** ✌

The **todo app** we’ll build in this tutorial will be pretty *cool*.

a user can **add or remove and todo** by entering enter after filling todo form and delete todo by clicking delete icon , and mark todo complete by clicking circle

I’ll explain how to build each **feature**, but you must follow along by

**typing** the code and running it on your end to get the most out of this tutorial

*👇* ***Click here to get starter files***

[**](https://codesandbox.io/s/todo-app-starter-ve307?file=/index.html)



Just in case if your code didn't **work** as expected 😟

you can find the **working** code 😃 by ***clicking*** 👉

on **every end of steps**

# **First let’s create the skeleton of the *ui***

## Add font awesome cdn 👇

We’ll use **font awesome icons** in this project for that we have to include font

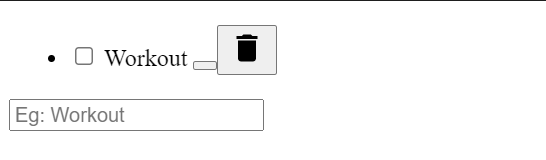
awesome cdn in body tag you can create your own font awesome kit

| <script src="https://kit.fontawesome.com/dd8c49730d.js" crossorigin="anonymous"></script> |
| --- |

## Markup of todo list , default todo , add todo form 👇

***Note:- We creating this todo just for demo purpose we gonna use this markup in javascript while we creating functionality for rendering new todo in todo list with the same ui then we’ll remove this default todo from the todo list***

| <div class="container"> <ul class="todo-list js-todo-list"> <li class="todo-item" data-key="123456"> <input id="123456" type="checkbox" /> <label for="123456" class="tick js-tick"></label> <span>Workout</span> <button class="delete-todo js-delete-todo"> <button class="delete-todo js-delete-todo"> <svg fill="var(--svgcolor)" xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24"> <path d="M6 19c0 1.1.9 2 2 2h8c1.1 0 2-.9 2-2V7H6v12zM19 4h-3.5l-1-1h-5l-1 1H5v2h14V4z" /> </svg> </button> </li> </ul>  <form class="formselect"> <input autofocus type="text" class="inputselect" placeholder="Eg: Workout"> </form> </div> |
| --- |



## Output of above codes 👉

[](https://codesandbox.io/s/step-1-dwssr)

# **Let’s add styling into ui** 🎨

## Basic styling

| html {  box-sizing: border-box; }  \*, \*::before, \*::after { box-sizing: border-box; margin: 0; padding: 0; }  body { font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, Oxygen-Sans, Ubuntu, Cantarell, "Helvetica Neue", sans-serif; line-height: 1.4; background-color: #fff; transition: all 0.5s; } |
| --- |

## 

## 

## Adding styling into container, svg button , todo list , todo item,

## todo item span , checked button

| .container {  width: 100%;  max-width: 700px;  margin: 0 auto;  padding-left: 10px;  padding-right: 10px;  color: #333;  margin: auto;  margin-top: 10%; }  .todo-list {  list-style: none;  margin-bottom: 20px; }  .todo-item {  margin-bottom: 10px;  width: 100%;  display: flex;  align-items: center;  justify-content: space-between; }  .todo-item span {  flex-grow: 1;  margin-left: 10px;  margin-right: 10px;  font-size: 22px; }  input[type="checkbox"] {  display: none; } .tick {  width: 30px;  height: 30px;  border: 3px solid #333;  border-radius: 50%;  display: inline-flex;  justify-content: center;  align-items: center;  cursor: pointer; } |
| --- |

## 

## 

## 

## Adding styling into delete todo button

.

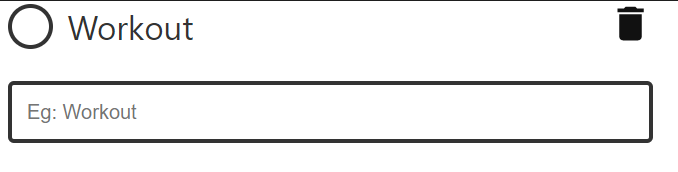
| delete-todo {  border: none;  background-color: transparent;  outline: none;  cursor: pointer; }  .delete-todo svg {  width: 30px;  height: 30px;  pointer-events: none; } |
| --- |



## 

## Adding styling into add todo form

| form { width: 100%; display: flex; justify-content: space-between; margin-bottom: 2rem; }  input[type="text"] { width: 100%; padding: 10px; border-radius: 4px; border: 3px solid #333; }  input:focus { outline: none; background-color: #d4e7d5; } |
| --- |

[](https://codesandbox.io/s/step-5-1t1ov)

## 

## Styling for todo complete effect checked ( we gonna use this in javascript )

| .done span { text-decoration: line-through; user-select: none; }  .tick::before { content: "✔️"; font-size: 20px; display: none; }  .done .tick::before { display: inline; } |
| --- |

## 

## Styling done 🎨



Take a breather, and see the [complete code](https://codesandbox.io/s/step-5-1t1ov) at the end of this step

# **Let’s start building feature for add new todo**

# **and render todo in the list , store todo as a**

# **array object to localstorage**

## Function for adding new todo

| function addTodo(todoText, id) {  const todoobject = {  todoText: document.querySelector(".inputselect").value,  checked: false;  id: Date.now(),  };  console.log(todoobject); } |
| --- |

In above code we create a function for adding a new todo

* We created todoobject to store all todo detail as an object property
* In todoobject we getting a value of entered input values with their id by

using .value method , creating checked property and giving boolean

value false and creating a id in todoobject using date.now method

* Console logged for demo purpose

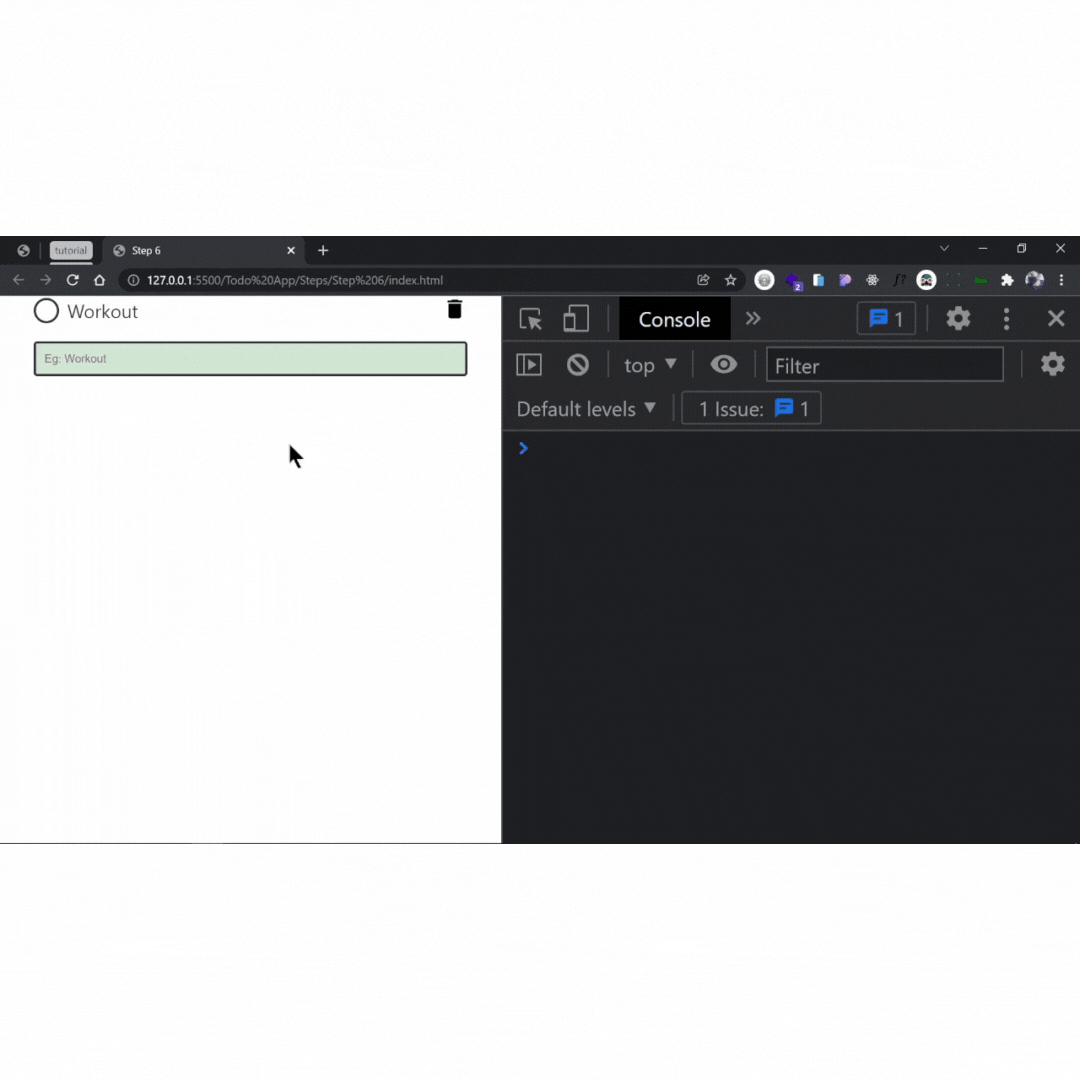
### Add a event listener submit to the form

| const form = document.querySelector(".formselect"); form.addEventListener("submit", (event) => {  event.preventDefault();  addTodo();  form.reset(); }); |
| --- |

In above code we added a event listener Submit to the form when form is submitted

* We will call our addTodo() function for adding todo
* We using reset() method to reset form after submitting a todo

See the output in console log 👇



As you can see above gif after submitting a form by pressing enter we getting

a new todo as an object in the console log

But we want to render a todo in todo-list to display in page to achieve that we have

to create a renderTodo() function above addTodo() function

## 

## Function for rendering new todo in display

In this function we’re going to pass todo as a parameter to access todo details from todoobject eg: todo.todoText , todo.id , todo.checked

* We’re going to select list where we will appending a all new todo items
* Select the current todo item in the DOM
* Check if checked is true add done class effect otherwise as it is
* Creating new element list (we’re going to use markup of todo what we created at beginning for demo purpose ) and append node in list

| function renderTodo(todo) { } |
| --- |

### 

### Select todo list for appending all todos in list

| function renderTodo(todo) {  const list = document.querySelector(".todo-list");  } |
| --- |

### 

### Select the current todo item in the DOM ( We’ll use

### item for deleting todo from todo list later in this tutorial )

| function renderTodo(todo) {  const list = document.querySelector(".todo-list");  const item = document.querySelector(`[data-key='${todo.id}']`);  } |
| --- |

### 

### Check if checked is true add done class effect otherwise as it is

| function renderTodo(todo) {  const list = document.querySelector(".todo-list");  const item = document.querySelector(`[data-key='${todo.id}']`);  const isChecked = todo.checked ? "done" : "";  } |
| --- |

### 

### Create new element li for todo ui

1. Set the class and data-key attribute
2. Add html markup of todo

| function renderTodo(todo) {  const list = document.querySelector(".todo-list");  const item = document.querySelector(`[data-key='${todo.id}']`);  const isChecked = todo.checked ? "done" : "";  const newlist= document.createElement("li");  newlist.setAttribute("class", `todo-item ${isChecked}`);  newlist.setAttribute("data-key", todo.id);  } |
| --- |

### 

We have to just cut and copy markup of todo ui to set the newList.innerHTML

what we create at beginning for demo purpose

| function renderTodo(todo) {  const list = document.querySelector(".todo-list");  const item = document.querySelector(`[data-key='${todo.id}']`);  const isChecked = todo.checked ? "done" : "";  const newlist= document.createElement("li");  newlist.setAttribute("class", `todo-item ${isChecked}`);   newlist.setAttribute("data-key", todo.id);  newlist.innerHTML = ` <input id="${todo.id}" type="checkbox"/> <label for "${todo.id}" class="tick js-tick"></label> <span>${todo.todoText}</span> <button class="delete-todo js-delete-todo"> <button class="delete-todo js-delete-todo"> <svg fill="var(--svgcolor)" xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24">  <path d="M6 19c0 1.1.9 2 2 2h8c1.1 0 2-.9 2-2V7H6v12zM19 4h-3.5l-1-1h-5l-1 1H5v2h14V4z" /> </svg> </button> `;   // appending a newList in list  list.append(newlist);  } |
| --- |

Yes here we successfully created our function for rendering a new todo in todo list

But as you can see we passed todo as a parameter of function to access the property

of todoobject for displaying in todo ui

To archive that we have to call renderTodo() function in addTodo function and pass todoobject object as an argument in renderTodo function

Updated addTodo as follows

| function addTodo(todoText, id) {  const todoobject = {  todoText: document.querySelector(".inputselect").value,  checked: false,  id: Date.now(),  };  console.log(todoobject); // console log for demo purpose   renderTodo(todoobject); } |
| --- |

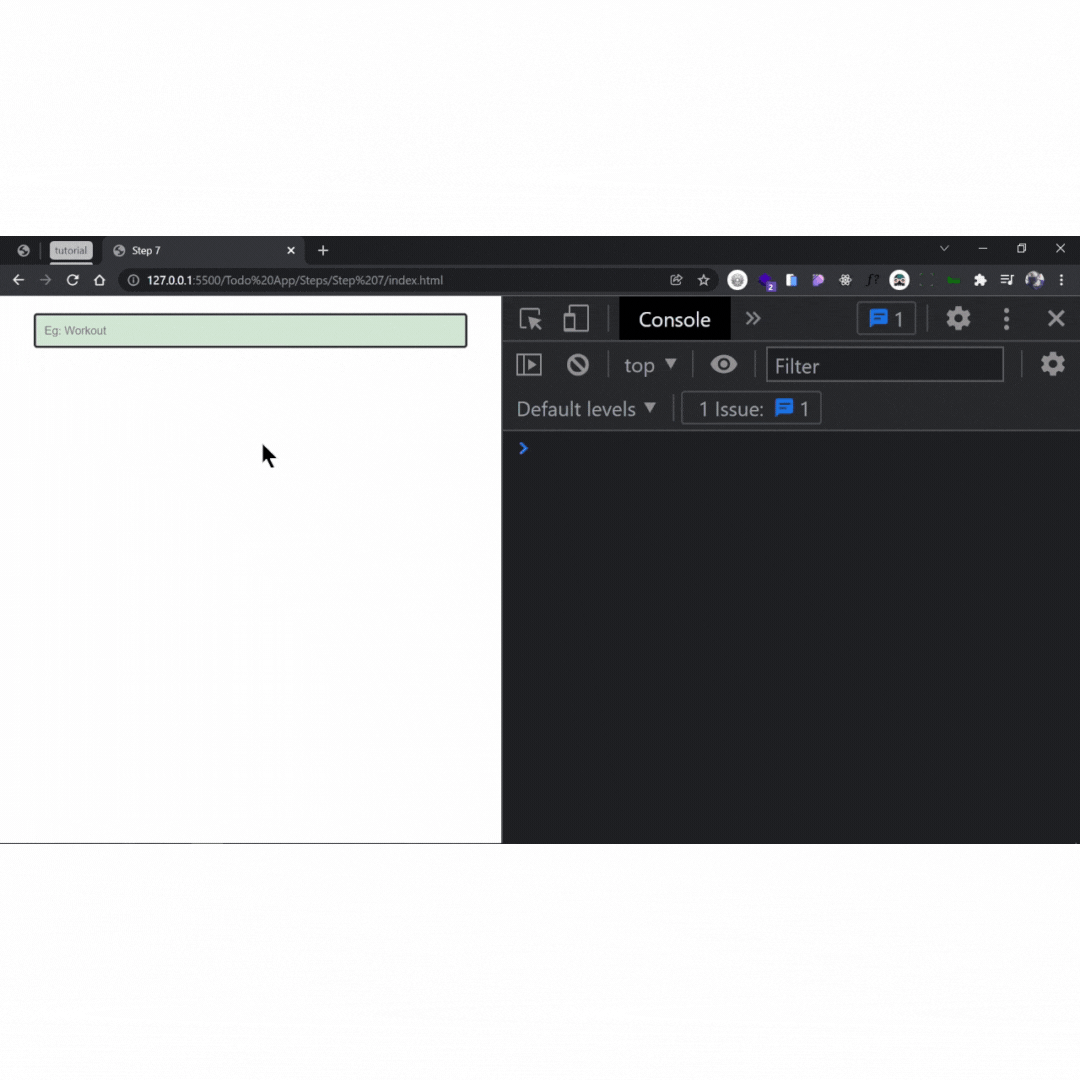
In above code we called renderTodo function and passed todoobject as an

argument now we can use property of todoobject in renderTodo() function

Now we can remove a default todo from list what we created at beginning

for demo purpose update markup of todo list as follows

| <**div** class="container"> <**ul** class="todo-list js-todo-list"> <!-- default todo removed --> <!-- default todo removed --> </**ul**>  <**form** class="formselect"> <**input** autofocus type="text" class="inputselect" placeholder="Eg: Workout"> </**form**> </**div**> |
| --- |

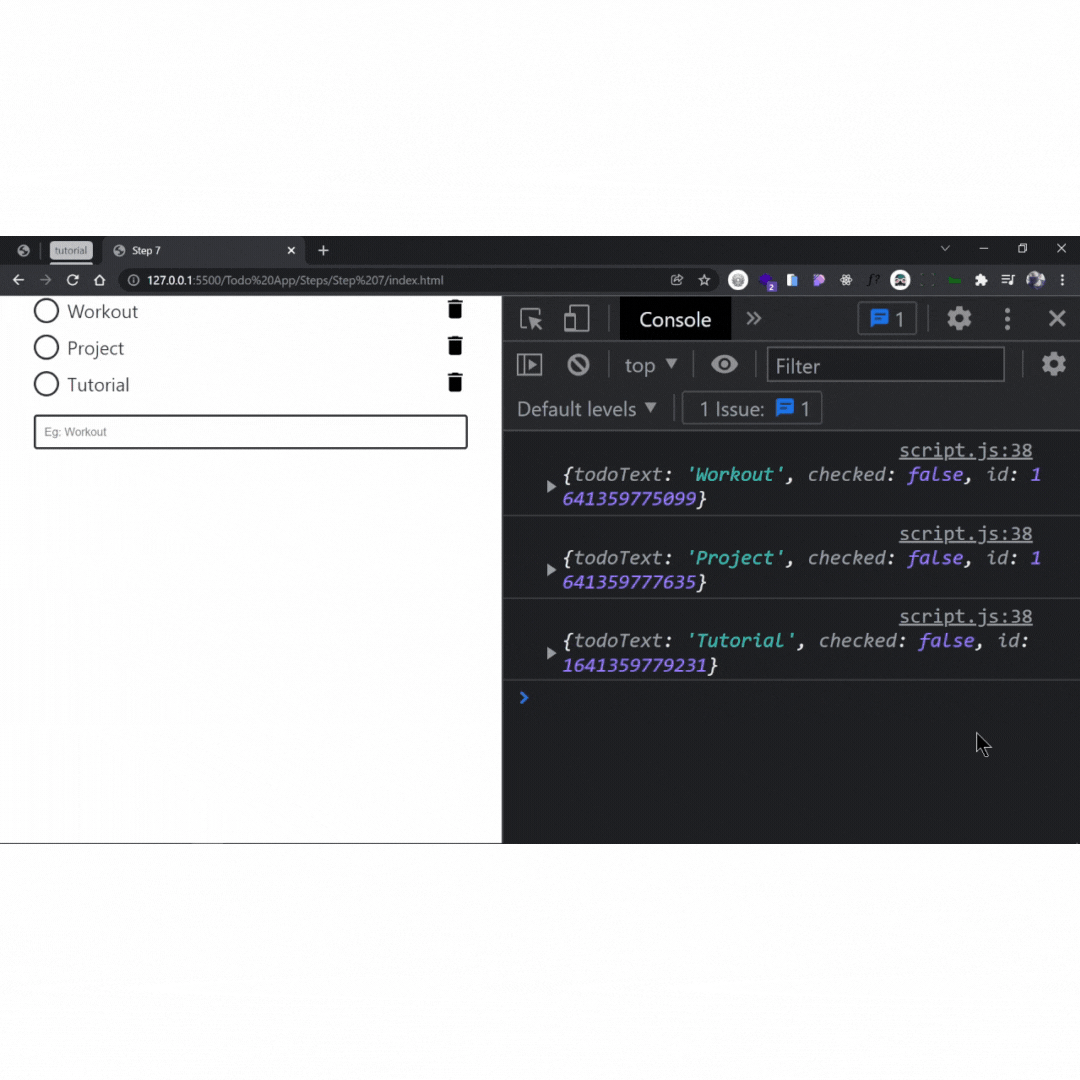


As you can see above it’s work as expected

## 

But here we have one problem when we refresh the page all todos cleared

See below gif



To keep it display we have to store data in localstorage for that we can

use javascript localStorage object

## Save todo data in browser local storage

1. Create empty todo array for storing all todos as a array object in localstorage
2. To store todo in localstorage we have to set todo array to localstorage
3. Push new todoobject to todoArray after calling addTodo()
4. Get todo array from localstorage to display todos in todo list

### Creating empty todoArray array object

| let todoArray = []; |
| --- |

### 

### Set todoArray to localstorage

We can set item to localstorage using localStorage.setItem method in renderTodo()

Add below code line in above renderTodo function

| localStorage.setItem("todoArray", JSON.stringify(todoArray)); |
| --- |

### Push new todoobject to todoArray array object

We have to push every new todoobject to todoArray array object after adding todo

Add below code line within addTodo

| todoArray.push(todoobject); |
| --- |

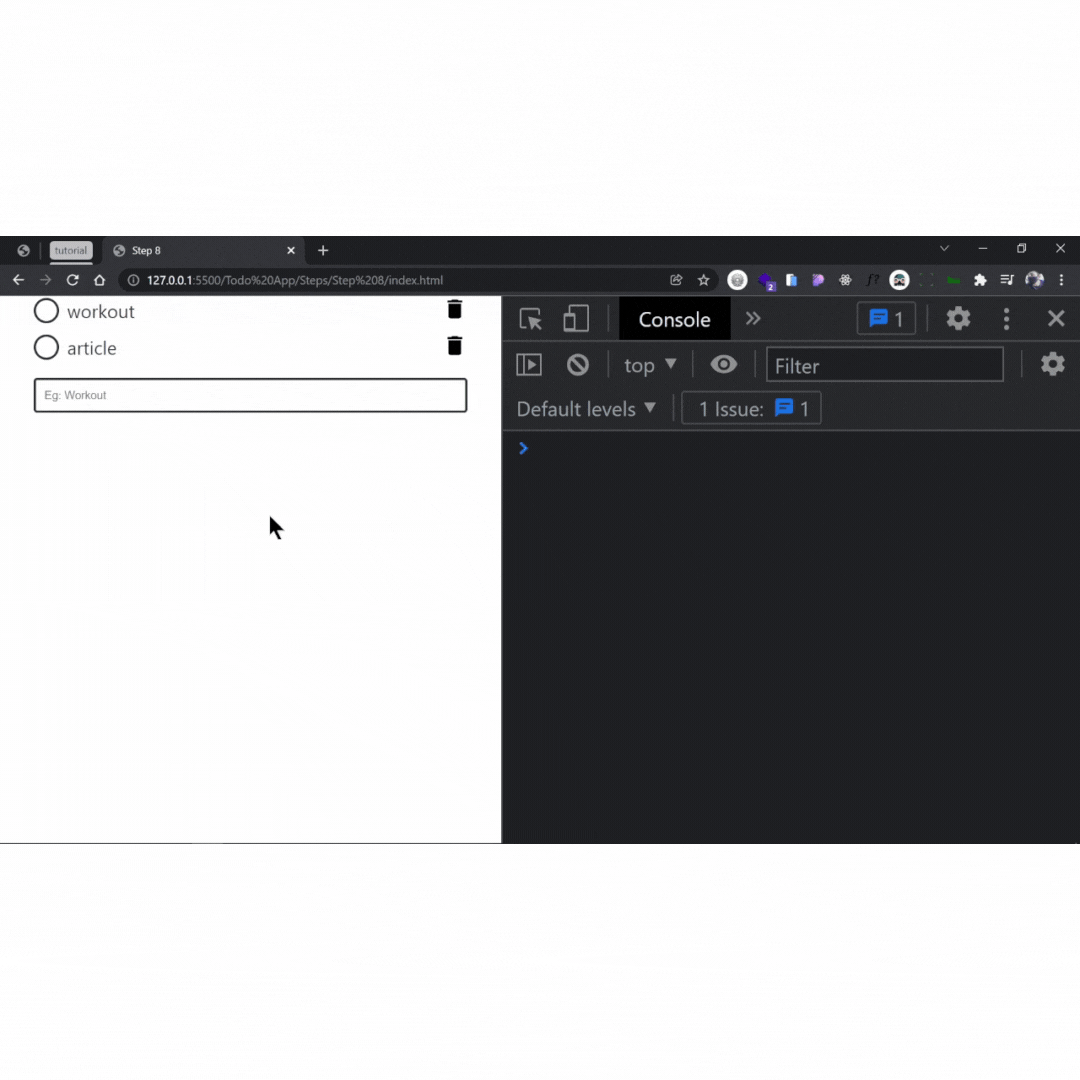
### Get todoArray from localstorage to display todos in todo list

To display todo in todo list we have to get the todo from todoArray after refreshing page to archive that we have to add event listener DOMContentLoaded and get the todoArray from localStorage

| document.addEventListener("DOMContentLoaded", () => {  const ref = localStorage.getItem("todoArray");  if (ref) {  todoArray = JSON.parse(ref);  todoArray.forEach((t) => {  renderTodo(t);  });  } }); |
| --- |

Now if we refresh page it’s keep the todos in todo list

See below gif



## 

Take a breather, and see the [complete code](https://codesandbox.io/s/step-8-unfuo) at the end of this step

## Let’s build functionality for marking todo done and delete todo

Do you remember we created id property in todoobject and set data-key attribute

to item in renderTodo function now we’re going to use these property and item to

Mark todo complete and delete todo from todo list

### Function for marking todo complete ( in toggleDone function we’re going to )

* Find the corresponding todoobject in the todoArray object
* Set checked property
* Call renderTodo function with todoArray index

| function toggleDone(key) { const index = todoArray.findIndex((myitem) => myitem.id === Number(key)); todoArray[index].checked = !todoArray[index].checked; renderTodo(todoArray[index]); console.log(todoArray[index]); // for demo purpose  console.log(index); // for demo purpose  } |
| --- |

In above code we created function toggleDone with key parameter , to mark done on specific

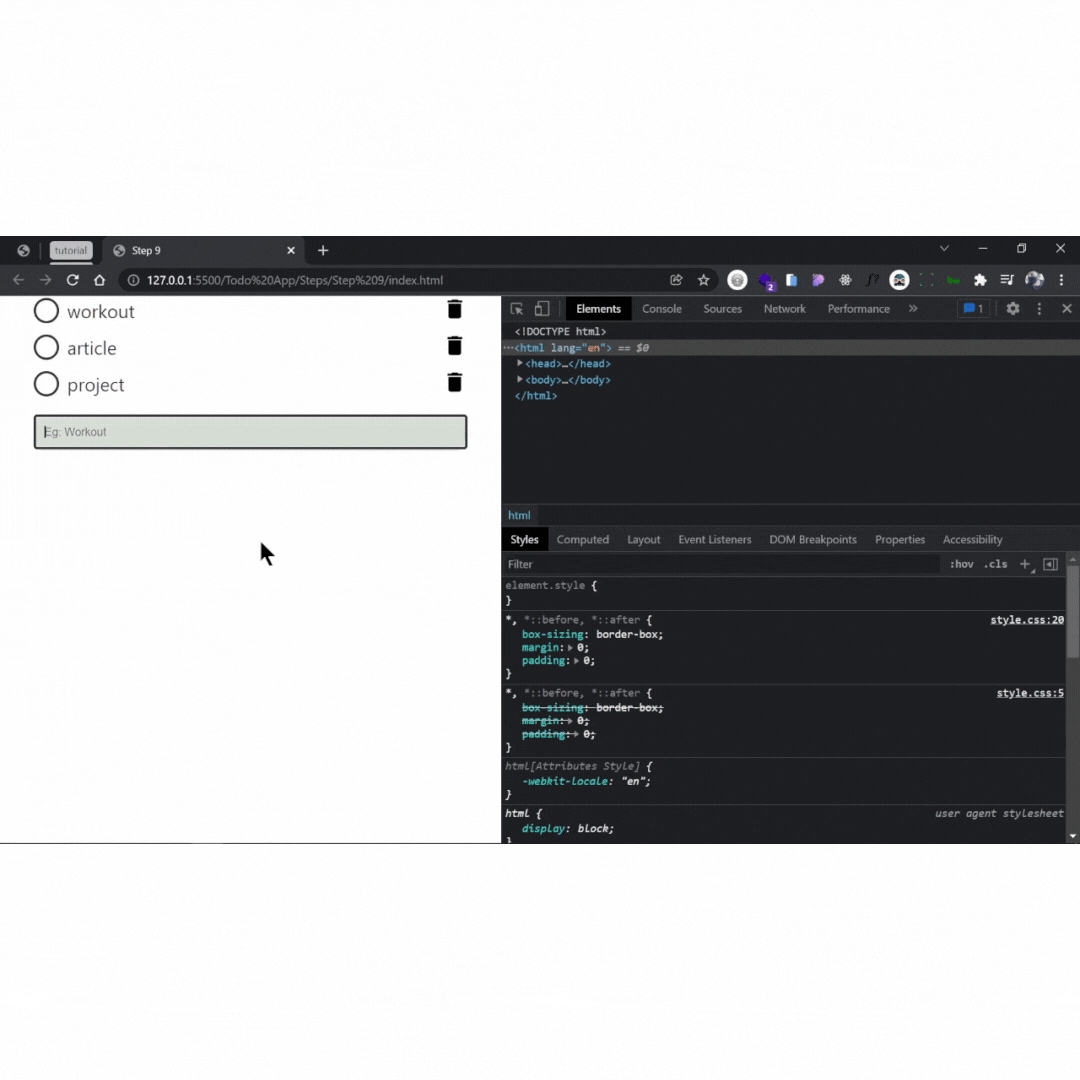
todo from the todo list we have to know which todo we wanted to mark done for that we find the index of corresponding todoobject from the todoArray object using findIndex method and set its checked property to the opposite. That means, `true` will become `false` and vice versa then called renderTodo()

### Call toggleDone function

| const list = document.querySelector(".js-todo-list"); list.addEventListener("click", (event) => {  if (event.target.classList.contains("js-tick")) {  const itemKey = event.target.parentElement.dataset.key;  toggleDone(itemKey);  } }); |
| --- |

In above code we added click event listener to todo list if targeted event class list contains js-tick then get the dataset.key of their parent Element and assign to itemkey and call toggleDone() function and passed itemKey as an argument

Let’s run above code and see what’s happening



As you can see here we have problem when we mark todo done it will duplicate the todo items instead of marking todo done

To solve this we have to check if the current todo item exists in the DOM first then and replace it with the updated newlist item

Update renderTodo() function as shown below :

| function renderTodo(todo) { localStorage.setItem("todoArray", JSON.stringify(todoArray)); const list = document.querySelector(".todo-list"); const item = document.querySelector(`[data-key='${todo.id}']`); const isChecked = todo.checked ? "done" : ""; const newlist = document.createElement("li"); newlist.setAttribute("class", `todo-item ${isChecked}`); newlist.setAttribute("data-key", todo.id); newlist.innerHTML = ` <input id="${todo.id}" type="checkbox"/> <label for "${todo.id}" class="tick js-tick"></label> <span>${todo.todoText}</span> <button class="delete-todo js-delete-todo"> <button class="delete-todo js-delete-todo"> <svg fill="var(--svgcolor)" xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24"> <path d="M6 19c0 1.1.9 2 2 2h8c1.1 0 2-.9 2-2V7H6v12zM19 4h-3.5l-1-1h-5l-1 1H5v2h14V4z" /> </svg> </button> `;  // If the item already exists in the DOM if (item) { // replace it list.replaceChild(newlist, item); } else { // otherwise append it to the end of the list list.append(newlist); } } |
| --- |

Now let’s see how it’s works



As you can see above gif it’s works as expected

## 

Take a breather, and see the [complete code](https://codesandbox.io/s/step-9-dwtws?file=/script.js) at the end of this step

### Function for delete todo ( in deleteTodo function we’re going to )

* Find the corresponding todoobject from the todoArray object
* Create a new object with properties of the current todoobject item and a `deleted` property which is set to true
* Remove the todoobject item from the array
* Render emptytodo

| function deleteTodo(key) {  const index = todoArray.findIndex((myitem) => myitem.id === Number(key));  const emptytodo = {  deleted: true,  ...todoArray[index], };  todoArray = todoArray.filter((myitem) => myitem.id !== Number(c));  renderTodo(emptytodo); } |
| --- |

In above code we created function deleteTodo with key parameter , to delete specific todo from the todo list we have to know which todo we wanted to delete for that we find the index of corresponding todoobject from the todoArray object using findIndex method

Created a new object with properties of the current todoobject item by using spread operatror and array[index] and a `deleted` property which is set to true

Filter outed deleted todo object from todoArray object using filter method

When deleteTodo() called it will remove the todoobject item from the array

otherwise it will not delete todo item by click on delete button from localstorage

### Call deleteTodo function ( Similarly how we called toggleDone() )

| const list = document.querySelector(".js-todo-list"); list.addEventListener("click", (event) => {  if (event.target.classList.contains("js-tick")) {  const itemKey = event.target.parentElement.dataset.key;  toggleDone(itemKey);  }   // delete todo  if (event.target.classList.contains("js-delete-todo")) {  const itemKey = event.target.parentElement.dataset.key;  deleteTodo(itemKey);  } }); |
| --- |

Finally let’s add remove method if emptytodo delete property true it will remove todo

Add below code in renderTodo function

| if (todo.deleted) {  item.remove();  return;  } |
| --- |

Now as you can see below gif we’re able to delete todo from todo list

## 

## 

## 

## 

## 

Here we almost completed todo app project

## Let’s add empty state when there is no todo to display ( optional )

Something like that

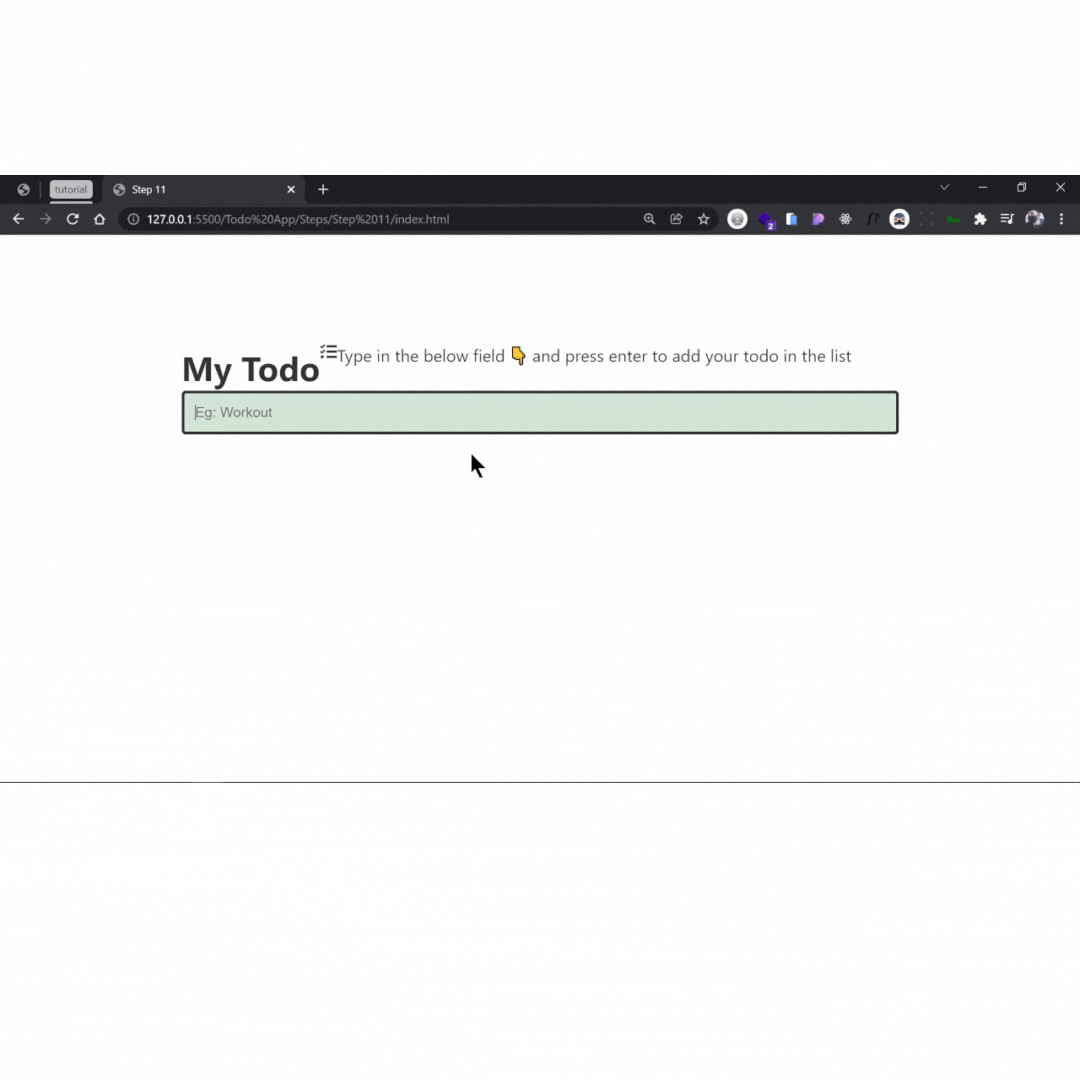


### Markup of empty state ( add this markup below todo list markup in html )

| <**div** class="empty-state"> <**h1** class="app-title">My Todo</**h1**> <**i** class="fas fa-tasks icon"></**i**> <**p** class="empty-state\_\_description"> Type in the below field 👇 and press enter to add your todo in the list </**p**> </**div**> |
| --- |

### when there is no todo in todo list then add .empty-state and make display to flex ( add this below the other styles )

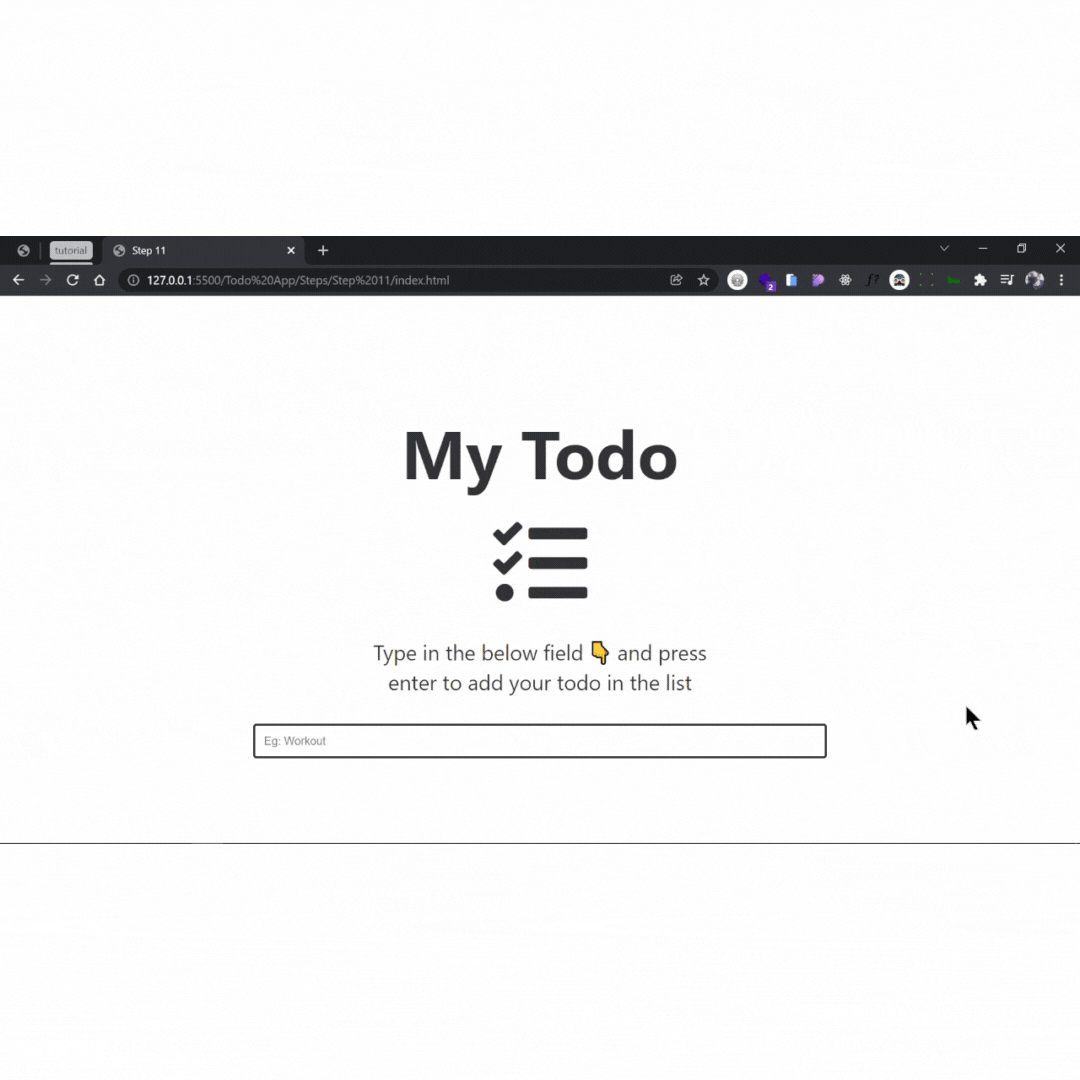
| .todo-list:empty + .empty-state {  display: flex; } |
| --- |



### Styling for empty state .empty-state by default is none

### ( add this below the other styles )

| .empty-state { flex-direction: column; align-items: center; justify-content: center; display: none; }  .app-title { text-align: center; margin-bottom: 20px; font-size: 80px; }  .checklist-icon { margin-bottom: 20px; }  .icon { font-size: 7rem; }  .empty-state\_\_title, .empty-state\_\_description { font-size: 25px; text-align: center; margin: 2rem; width: 25rem; } |
| --- |



[](https://codesandbox.io/s/step-11-d3b4x)

Tutorial written by [sumit](https://www.instagram.com/codingsumit/) Creator of [coding.batch](https://www.instagram.com/coding.batch/)

# 

# Congratulation we have

# successfully created todo app 🥳

# — The End —

## Was it useful 🤔 let me know on instagram

## — Get Source Code Here —

## — Get Source with code comment —