





Search any topic...

Follow Us  

To Do List Javascript

Javascript to do list app is a **beginner level project** that you must create to learn basic methods to control the webpage on the user's action and make it interactive.

In this article, we have explained how to make your first javascript project in detail with source code. This is an editable to do list using javascript.

We are going to create a basic to do list app in javascript with the following features:

1. Add new task
2. Delete task
3. Mark task as completed
4. Edit task
5. Sort tasks by date
6. Search tasks



repor

- ▶ Tic Tac Toe JavaScript
- ▶ JavaScript Calculator
- ▶ Age calculator in JavaScript
- ▶ JavaScript BMI Calculator
- ▶ Javascript Countdown Timer
- ▶ Get Value From JSON Object In Jav
- ▶ HTML5 Digital Clock with JavaScript
- ▶ Password Generator In JavaScript

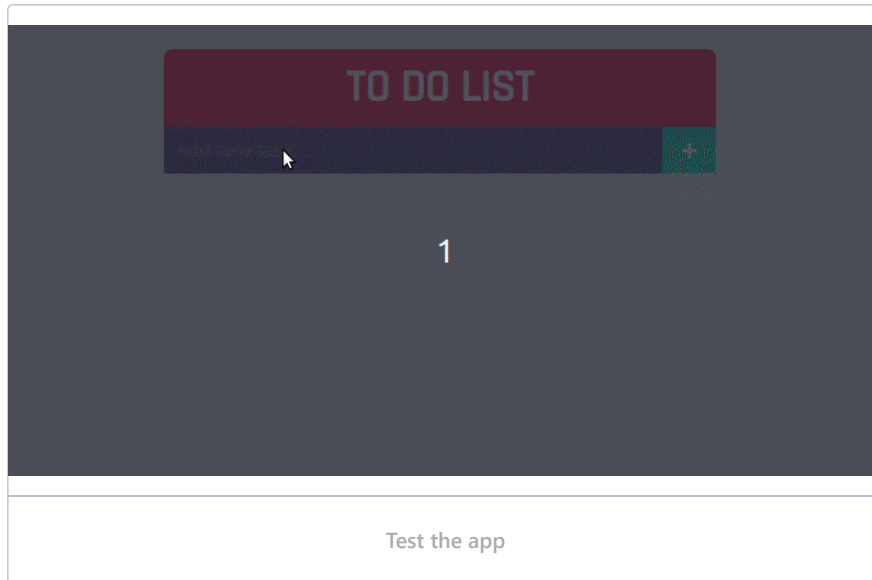


r



Here is the [final To Do list app](#) you are going to make.

Prerequisites: Before proceeding with this section you need to have a basic understanding of [HTML](#), [CSS](#), and [Javascript](#).



Making To Do List Javascript

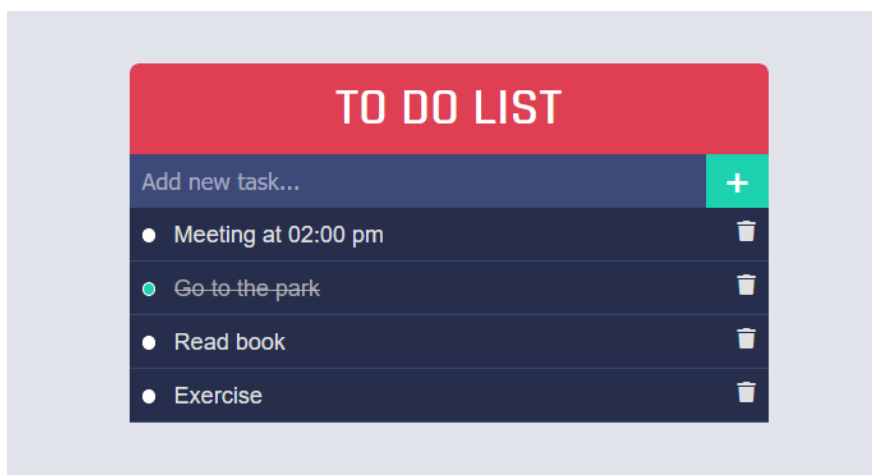
Creating this app will be a small step to learn javascript and create projects in javascript.

We will create this app by working in 3 different section first we will work on



▼ Category

Our complete app will look like something as shown in the picture below.



To Do List HTML Template

First, we will create the HTML structure of our app.

Create a container for our app. We will put all our elements inside this container and this container will be the parent of our app.

Give this container a class name of **"container"**. We will use this class name to style our container.

Creating outline

Then create an [<div> element](#) with class name 'app' for the app, this will help us to style the app elements using CSS.

Create a header for the app inside an element with the class 'app'.

```
1 <div class="container">
2   <div class="app">
3     <h2>To DO List</h2>
4   </div>
5 </div>
```

Form creation for writing task

To take tasks as input from the user you need to create a [form](#) with a text input and a submit button.

On the submit button will later be attached a click event listener to add the task to the list.

```
1 <form>
2   <input type="text" placeholder="Add new task...">
3   <button type="submit">&plus;</button>
4 </form>
```

Create task list

Now we need to create a list of tasks. To do this we will create an `` element inside element with class 'app'.

Inside ```` element we will have our list of tasks.

Each list of task contains 3 sections:

1. **Checkbox:** To mark the completed task. Create a checkbox
2. **Input element:** To show and edit the task
3. **Delete button:** To delete the task. Create a trash icon using font-awesome.

To check if the task is complete create an ``onclick`` event which will listen to whether the task is complete or not, also add another ``onclick`` event to delete the task.

The following code represent the task list template (we will not insert this in HTML but its is just to visualise the task list template):

```
1 <ul>
2   <li>
3     <input type="checkbox" onclick="taskComplete(this)" class="checkbox">
4     <input type="text" value="My Task" class="task" onfocus="getCurTask(this)" />
5     <i class="fa fa-trash" onclick="removeTask(this)"></i>
6   </li>
7 </ul>
```

We have added different event listeners to the task list to use it when different action occurs.

We will use the ``taskComplete()`` method to mark the task as complete, ``removeTask()`` method to remove the task, ``editTask()`` method to edit the task, and ``getCurrentTask()`` method to get the current task value when the user is editing the task.

Complete HTML Code

Here is the complete HTML code of our app.

```

1  <div class="container">
2    <div class="app">
3      <h1>TO DO LIST</h1>
4      <form>
5        <input type="text" placeholder="Add new task...">
6        <button type="submit">&plus;</button>
7      </form>
8      <ul>
9        <li>
10         <input type="checkbox" onclick="taskComplete(this)" class="task">
11         <input type="text" value="Exercise" class="task" onfocus="getFocus">
12         <i class="fa fa-trash" onclick="removeTask(this)"></i>
13       </li>
14     </ul>
15   </div>
16 </div>

```

HTML Output:

TO DO LIST

- ☐ Exercise

Note: The dummy data (Exercise) will be removed from the code later.

CSS Code For TO DO List App

Now let's style the To Do list app using CSS.

First, select the 'container' class, it is the first element we had created and it holds our app. Use [flexbox](#) to centre its child horizontally, give some padding and background color.



```
1  .container {  
2    display: flex;  
3    flex-direction: column;  
4    align-items: center;  
5    padding-top: 50px;  
6    background-color: #e1e3ea;  
7    height: 100vh;  
8    font-family: sans-serif;  
9  }
```

Style 'app' element:

Create a CSS class with the name 'app' and give it a width of **60%** for device size more than 768px and **90%** for device size less than 768px using [media query](#).

This will help the app to resize itself according to a different device.

```
1  .app {  
2    width: 60%;  
3  }  
4  
5  @media (max-width:768px) {  
6    .app {  
7      width: 90%;  
8    }  
9  }
```

Style the heading

To make the To Do app you should make the heading attractive.

You can use a different font for it. For this, you can import a google font as shown in the code below.

```

1  @import url('https://fonts.googleapis.com/css2?family=Rajdhani:wght@
2
3  .app h1 {
4      color: white;
5      font-size: 40px;
6      padding: 10px 0;
7      text-align: center;
8      border-radius: .5rem .5rem 0 0;
9      background-color: #de3f53;
10     font-family: 'Rajdhani', sans-serif;
11 }

```

Styling form and input

Use flexbox to flow the form elements in columns and give 10% width to it.

Provide basic color, padding, and other properties to input and submit button and give 90% width to input and 10% width to submit button.

```

1  .app form {
2      display: flex;
3      width: 100%;
4      background-color: #262e4c;
5  }
6
7  .app form input {
8      border: none;
9      background-color: #3c4979;
10     font-size: 18px;
11     color: white;
12     padding: 10px;
13     width: 90%;
14 }
15
16 .app form button {
17     border: none;
18     color: white;
19     background-color: #1dd2af;
20     font-size: 25px;
21     font-weight: 600;
22     height: 42px;
23     width: 10%;
24 }

```

Styling list elements

Now style the list elements that are our task. Each task element contains 3 different elements inside it.

Select ```` tag and remove the default bullets to it by using ``list-style: none`` .

Select the checkbox by `.check` CSS selector and remove the original look and style it to look round using the given code below.

For checked box give different background color. i.e. `background-color: #1dd2af;`

Push the trash button to the right of the task using `float: right` property.

```
1  .app ul {
2    list-style: none;
3    color: #e0e0e0;
4    padding: 0;
5  }
6
7  .app ul li {
8    padding: 10px;
9    background-color: #262e4c;
10   border-bottom: 1px solid #3a4674;
11  }
12
13  .check {
14    width: 0.8em;
15    height: 0.8em;
16    cursor: pointer;
17    border-radius: 50%;
18    background-color: white;
19    border: 1px solid rgb(255, 255, 255);
20    -webkit-appearance: none;
21  }
22
23  .check:checked {
24    background-color: #1dd2af;
25  }
26
27  .task {
28    font-size: 18px;
29    padding: 0 10px;
30  }
31
32  .task:focus {
33    outline: none;
34  }
35
36  .app ul li i {
37    float: right;
38    cursor: pointer;
39  }
40
41  .app ul li i:hover {
42    color: rgb(255, 82, 82);
43  }
```


Style the completed task



For the task which is completed, we have to make it look different. So create a class 'complete' and add necessary CSS properties, which we will add to the task which is completed.

```
1 | .complete {  
2 |     color: rgba(192, 192, 192, 0.8);  
3 |     text-decoration: line-through;  
4 | }
```

To Do List JavaScript Code

Now let's create the logic and wrote to do list Javascript code.

We are going to use **localStorage** to store our tasks so that we can access them later.

Let's take a brief introduction to localStorage.

localStorage

localStorage is a global object that is used to store data locally on a user's computer.

It is used to store data in a way that is accessible by the user's web browser.

localStorage is a key-value storage system.

It has a limit of 5MB.

The methods of localStorage are:

1. **setItem(key, value)** - to set a new key-value pair

```
1 | `localStorage.setItem('name', 'John');`
```

2. **getItem(key)** - to get the value of a key

```
1 | `localStorage.getItem('name');`
```

3. **removeItem(key)** - to remove a key-value pair

```
1 | `localStorage.removeItem('name');`
```

4. **clear()** - to clear all the key-value pairs

```
1 | `localStorage.clear();`
```

The javascript code for our To Do list app is mainly divided into a different function for different tasks.

Our app is going to work in the following steps:

1. **Load tasks:** When the page is loaded, the app will check if there is any task in localStorage. If there is any task, it will display them.
2. **Add task:** When the user clicks on the add button, the app will add the task to the list and store it in localStorage.
3. **Edit task:** When user click on the task itself then they can edit the task and when they remove focus from the task, the app will update the task in localStorage.
4. **Mark complete:** When the user clicks on the checkbox, the app will mark the task as completed and store it in localStorage.
5. **Remove task:** When the user clicks on the trash button, the app will remove the task from the list and also remove it from localStorage.

For each of the above mentioned tasks, we have to create a function and call it when the user take the action.

1. Function to load the tasks from localStorage

When the user first loads the page, then we need to check if there is any task in localStorage. If there is any task, then we need to display it.

We are using the [ternary operator](#) to determine if the task is complete or not and then add the class 'complete' to the task.

```
1 | // On app load, get all tasks from localStorage
2 | window.onload = loadTasks;
3 |
4 | function loadTasks() {
5 |     // Get the tasks from localStorage and convert it to an array
6 |     let tasks = Array.from(JSON.parse(localStorage.getItem("tasks")));
7 |
8 |     // Loop through the tasks and add them to the list
9 |     tasks.forEach(task => {
10 |         const list = document.querySelector("ul");
11 |         const li = document.createElement("li");
12 |         li.innerHTML = `<input type="checkbox" onclick="taskComplete(thi
13 |             <input type="text" value="${task.task}" class="task ${task
14 |             <i class="fa fa-trash" onclick="removeTask(this)"></i>`;
```

```

15     list.insertBefore(li, list.children[0]);
16   });
17 }

```

Function to add the task

To add a new task first create a javascript function with the name 'addTask'.

Within the function first, **check if the task is empty** or not. If it is empty then return and alert the user.

Then **check if the task is already present** in the list or not. If it is present then return again and alert the user.

If the task is not present in the list then add the task to the list and store it in localStorage.

If the task is not empty then create a new ``li`` element and set its ``innerHTML`` with the HTML code for list element and put the task value in it using [backticks](#).

Now append the created list element at the start of the list by using the ``insertBefore`` method.

Finally, clear the input section for the next task to be added.

```

1  function addTask() {
2    const task = document.querySelector("form input");
3    const list = document.querySelector("ul");
4    // return if task is empty
5    if (task.value === "") {
6      alert("Please add some task!");
7      return false;
8    }
9    // check is task already exist
10   let tasks = Array.from(JSON.parse(localStorage.getItem("tasks")));
11   // task already exist
12   tasks.forEach(todo => {
13     if (todo.task === task.value) {
14       alert("Task already exist!");
15       task.value = "";
16       return;

```

```

17     }
18   });
19
20   // add task to local storage
21   localStorage.setItem("tasks", JSON.stringify([...JSON.parse(localStorage.getItem("tasks"))], ...[task]));
22
23   // create list item, add innerHTML and append to ul
24   const li = document.createElement("li");
25   li.innerHTML = `<input type="checkbox" onclick="taskComplete(this)" /> <input type="text" value="${task.value}" class="task" onfocus="this.focus()" /> <i class="fa fa-trash" onclick="removeTask(this)"></i>`;
26   list.insertBefore(li, list.children[0]);
27   // clear input
28   task.value = "";
29 }

```

Also add a submit event listener to the form to call the `addTask` function.

```

1 // Add submit event listener to form
2 document.querySelector("form").addEventListener("submit", e => {
3   e.preventDefault();
4   addTask();
5 });

```

Function to edit the task

First store the current task which is being edited to track change. So when the user focuses on the input field, the current task is stored in the variable ``currentTask``.

Now check if the task is already there in the list. If it is there then return again and alert the user.

If the task is not present in the list then edit the task and store it in `localStorage`.

```

1 // store current task to track changes
2 var currentTask = null;
3
4 // get current task
5 function getCurrentTask(event) {
6   currentTask = event.value;
7 }

```

```

8
9 // edit the task and update local storage
10 function editTask(event) {
11     let tasks = Array.from(JSON.parse(localStorage.getItem("tasks")));
12     // check if task is empty
13     if (event.value === "") {
14         alert("Task is empty!");
15         event.value = currentTask;
16         return;
17     }
18     // task already exist
19     tasks.forEach(task => {
20         if (task.task === event.value) {
21             alert("Task already exist!");
22             event.value = currentTask;
23             return;
24         }
25     });
26     // update task
27     tasks.forEach(task => {
28         if (task.task === currentTask) {
29             task.task = event.value;
30         }
31     });
32     // update local storage
33     localStorage.setItem("tasks", JSON.stringify(tasks));
34 }

```

function to mark the complete task

The `taskComplete` function marks the task as a completed task.

The function basically toggles a CSS class 'complete' on the `` element of the list using the `toggle` method via the `classList` property of the element.

The function also updates the local storage by updating the `completed` property of the task.

```

1 function taskComplete(event) {
2     let tasks = Array.from(JSON.parse(localStorage.getItem("tasks")));
3     tasks.forEach(task => {
4         if (task.task === event.nextElementSibling.value) {

```

```

5     task.completed = !task.completed;
6   }
7   });
8   localStorage.setItem("tasks", JSON.stringify(tasks));
9   event.nextElementSibling.classList.toggle("completed");
10  }

```

function to delete the complete task

The trash button is used here to delete the task. Same as the previous function the `event` parameter point to the trash button.

Finally, update the local storage by removing the task from the list.

```

1  function removeTask(event) {
2    let tasks = Array.from(JSON.parse(localStorage.getItem(""tasks
3    tasks.forEach(task => {
4      if (task.task === event.parentNode.children[1].value) {
5        // delete task
6        tasks.splice(tasks.indexOf(task), 1);
7      }
8    });
9    localStorage.setItem(""tasks";, JSON.stringify(tasks));
10   event.parentElement.remove();
11  }

```

Source code of To Do List App

Here is the complete source code of To Do list app.

```

1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5    <meta charset="UTF-8">
6    <meta name="viewport" content="width=device-width, initial-scale=1
7    <title>To DO List</title>
8    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/lib
9    <style>
10     @import url("https://fonts.googleapis.com/css2?family=Rajdhani:w
11
12     * {

```



```
13     margin: 0;
14 }
15
16 .container {
17     display: flex;
18     flex-direction: column;
19     align-items: center;
20     padding-top: 50px;
21     background-color: #e1e3ea;
22     height: 100vh;
23     font-family: sans-serif;
24 }
25
26 .app {
27     width: 60%;
28 }
29
30 @media (max-width:768px) {
31     .app {
32         width: 90%;
33     }
34 }
35
36 .app h1 {
37     color: white;
38     font-size: 40px;
39     padding: 10px 0;
40     text-align: center;
41     border-radius: .5rem .5rem 0 0;
42     background-color: #de3f53;
43     font-family: "Rajdhani", sans-serif;
44 }
45
46 .app form {
47     display: flex;
48     width: 100%;
49     background-color: #262e4c;
50 }
51
52 .app form input {
53     border: none;
54     background-color: #3c4979;
55     font-size: 18px;
56     color: white;
57     padding: 10px;
58     width: 90%;
59 }
60
61 .app form button {
62     border: none;
63     color: white;
64     background-color: #1dd2af;
65     font-size: 25px;
```



```
66     font-weight: 600;
67     height: 42px;
68     width: 10%;
69 }
70
71 .app ul {
72     list-style: none;
73     color: #e0e0e0;
74     padding: 0;
75 }
76
77 .app ul li {
78     padding: 10px;
79     background-color: #262e4c;
80     border-bottom: 1px solid #3a4674;
81 }
82
83 .check {
84     width: 0.8em;
85     height: 0.8em;
86     cursor: pointer;
87     border-radius: 50%;
88     background-color: white;
89     border: 1px solid rgb(255, 255, 255);
90     -webkit-appearance: none;
91 }
92
93 .check:checked {
94     background-color: #1dd2af;
95 }
96
97 .task {
98     font-size: 18px;
99     padding: 0 10px;
100    width: fit-content;
101    background: transparent;
102    border: none;
103    color: #fff;
104 }
105
106 .task:focus {
107     outline: none;
108 }
109
110 .app ul li i {
111     float: right;
112     cursor: pointer;
113 }
114
115 .app ul li i:hover {
116     color: rgb(255, 82, 82);
117 }
118
```



```

119     .completed {
120         color: rgba(192, 192, 192, 0.8);
121         text-decoration: line-through;
122     }
123 </style>
124 </head>
125
126 <body>
127     <div class="container">
128         <div class="app">
129             <h1>TO DO LIST</h1>
130             <form>
131                 <input type="text" placeholder="Add new task...">
132                 <button type="submit">&plus;</button>
133             </form>
134             <ul></ul>
135         </div>
136     </div>
137     <script>
138         // On app load, get all tasks from localStorage
139         window.onload = loadTasks;
140
141         // On form submit add task
142         document.querySelector("form").addEventListener("submit", e => {
143             e.preventDefault();
144             addTask();
145         });
146
147         function loadTasks() {
148             // check if localStorage has any tasks
149             // if not then return
150             if (localStorage.getItem("tasks") == null) return;
151
152             // Get the tasks from localStorage and convert it to an array
153             let tasks = Array.from(JSON.parse(localStorage.getItem("tasks")));
154
155             // Loop through the tasks and add them to the list
156             tasks.forEach(task => {
157                 const list = document.querySelector("ul");
158                 const li = document.createElement("li");
159                 li.innerHTML = `<input type="checkbox" onclick="taskComplete">
160                     <input type="text" value="${task.task}" class="task ${task.completed}"/>
161                     <i class="fa fa-trash" onclick="removeTask(this)"></i>`;
162                 list.insertBefore(li, list.children[0]);
163             });
164         }
165
166         function addTask() {
167             const task = document.querySelector("form input");
168             const list = document.querySelector("ul");
169             // return if task is empty
170             if (task.value === "") {
171                 alert("Please add some task!");

```

```

172     return false;
173 }
174 // check is task already exist
175 if (document.querySelector(`input[value="${task.value}"]`)) {
176     alert("Task already exist!");
177     return false;
178 }
179
180 // add task to local storage
181 localStorage.setItem("tasks", JSON.stringify([...JSON.parse(lo
182
183 // create list item, add innerHTML and append to ul
184 const li = document.createElement("li");
185 li.innerHTML = `<input type="checkbox" onclick="taskComplete(t
186 <input type="text" value="${task.value}" class="task" onfocus=
187 <i class="fa fa-trash" onclick="removeTask(this)"></i>`;
188 list.insertBefore(li, list.children[0]);
189 // clear input
190 task.value = "";
191 }
192
193 function taskComplete(event) {
194     let tasks = Array.from(JSON.parse(localStorage.getItem("tasks"
195     tasks.forEach(task => {
196         if (task.task === event.nextElementSibling.value) {
197             task.completed = !task.completed;
198         }
199     });
200     localStorage.setItem("tasks", JSON.stringify(tasks));
201     event.nextElementSibling.classList.toggle("completed");
202 }
203
204 function removeTask(event) {
205     let tasks = Array.from(JSON.parse(localStorage.getItem("tasks"
206     tasks.forEach(task => {
207         if (task.task === event.parentNode.children[1].value) {
208             // delete task
209             tasks.splice(tasks.indexOf(task), 1);
210         }
211     });
212     localStorage.setItem("tasks", JSON.stringify(tasks));
213     event.parentElement.remove();
214 }
215
216 // store current task to track changes
217 var currentTask = null;
218
219 // get current task
220 function getCurrentTask(event) {
221     currentTask = event.value;
222 }
223
224 // edit the task and update local storage

```

```
225     function editTask(event) {
226         let tasks = Array.from(JSON.parse(localStorage.getItem("tasks"))
227         // check if task is empty
228         if (event.value === "") {
229             alert("Task is empty!");
230             event.value = currentTask;
231             return;
232         }
233         // task already exist
234         tasks.forEach(task => {
235             if (task.task === event.value) {
236                 alert("Task already exist!");
237                 event.value = currentTask;
238                 return;
239             }
240         });
241         // update task
242         tasks.forEach(task => {
243             if (task.task === currentTask) {
244                 task.task = event.value;
245             }
246         });
247         // update local storage
248         localStorage.setItem("tasks", JSON.stringify(tasks));
249     }
250 </script>
251 </body>
252
253 </html>
```

Conclusion

Finally, we have a working to-do list app. It is very simple to use and it is very easy to add new tasks

You can add, edit, remove and mark tasks as completed. You can also save your tasks to local storage and load them when you open the app again.

About Us

[About us](#)[Contact us](#)[Privacy Policy](#)[Disclaimer](#)

Tutorials

[HTML5](#)[CSS3](#)[JavaScript](#)[Bootstrap 4](#)[Python](#)[Practice Problems](#)

Tools

[HTML Editor](#)[Advance HTML Editor](#)[JavaScript Compiler](#)

Follow Us



Copyright © 2022 Tutorials Tonight

