DOM Manipulation

Live Coding:

The title would be to do list. At the core we are creating a list, an unordered list we can do this with ordered list as well and the unordered list will have the list of things that we want to do. Furthermore, we will have an input type below that we will have a button. We will give a certain logic in the input type such that when we click the button then it will execute the logic that we have mentioned in the input tag.

TO-Do List:

In html file;

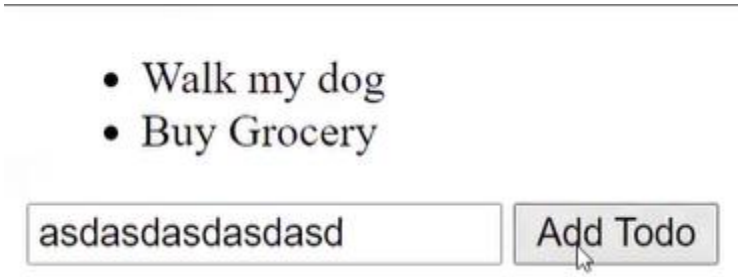
```
<ul>
    <li>Walk my Dog</li>
    <li>By Grocery</li>
</ul>

<input type="text" id="input-field">
<!-- // do something -->
<button id="add-todo">ADD To-Do</button>
```

In js file;

```
var todoList = document.getElementById('todo-list')
console.log(todoList)
```

Output:



When you enter something in the input and click on add to do it should get added to the list.

For now, let us give an id to the ul tag as todo-list and another id to the input tag as input-field and the last one to the button tag as add-todo. Now back in my script right first I have to include the script file. So, <script src="script.js"></script>. In the script.js file, we need get the reference of all the id's from the HTML file.

todoList, this will be my main list item, the unordered list using document.getElementById('todo-list'). To verify if the script is linked properly and if the reference is getting in properly.

Now when we refresh the page, we will be getting a null here which should not be the case because we won't be able to do anything. We will not be able to attach an event listener to it. What to do?

The reason is, when the HTML document is parsing from top to bottom, at first it comes across the javascript tag and then later on it will move forward to the body tag so the moment it gets to the script tag it with immediately execute everything that is written in the script tag and then moved to the body tag. This is the reason why we are getting a null.

A quick way to solve this is to put the script tag at the end of the body tag or we can specify an attribute – defer to it, a Boolean attribute. Defer, when present, it specifies that the script is executed when the page has finished parsing. The defer attribute is only for external scripts (should only be used if the src attribute is present).

So, we are done with getting the reference for the todo-list. Now we need to get a reference for the input tag as well.

```
var inputField = document.getElementById('input-field')
```

Similarly, we can get reference to the button tag as well.

```
var addTodoButton = document.getElementById('add-todo')
```

The better you name your variable, the better the chance of someone newcoming in on your code to understand it. Always give descriptive name for elements and functions.

Every time We enter something in input and click on the button, it should get added to the to do list. So, for that we would add that an event listener. we need to attach event listener to the add-todo button.

To make it look clean instead of writing and animals function and anonymous function and the event listener butte create a separate function as addItemToList(). irrespective of where you declare your function, above or below the addEventListener, JavaScript will host all the function definition and variable declaration in the global context.

So, using onclick, when we click the button the function has to be executed. So, the very first parameter to get executed would be the event object. Meaning when we click on the button, information such as where was the mouse, when did we click the button will be recorded in the event object.

Now what I need to do now can you to get whatever is type here right that is my and not the end but temporary end goal. So, how can we get the value of that input field. We have the reference of the input field.

Basically, everything related to input when you put in an input tag and refer that in your javascript it will have a '.value' parameter. So, when the button is clicked addItemToList is executed we want to get the input.

```
    var newTodoItem = inputField.value
    console.log(newTodoItem)
```

We did not put anything in the input box and I click on Add Todo, something is getting console.logged. We are calling the function addItemToList the moment when a user clicks the button. When the function is getting executed, we get the value from the input field. Whatever text the user has provided we are console logging it.

When the user is actually clicking the button the addEventListener function is taking the addItemToList function and attaching that to the event of clicking. We will click the button, only then it will execute this function addItemToList and doing so it will also pass and another object called as event object.

We need to add text to the unordered list. If we console.log('todoList'), we will be getting the ul tag.

```
addTodoButton.addEventListener('click', addItemToList)

var todoList = document.getElementById('todo-list')
var inputField = document.getElementById('input-field')
var addTodoButton = document.getElementById('add-todo')

function addItemToList(e){
    var newTodoItem = inputField.value
    console.log(newTodoItem)

}
addTodoButton.addEventListener('click', addItemToList)
```

Anything we write in the input box and click on the Add Todo button will be logged in the console.

```
<li>Walk my dog</li>
<li>Buy Grocery</li>
```

Since this is coming as a string, we can concatenate it.

```
var newTodoListHTML = todoListHTML + '<li>' + inputTest + '</li>'
```

OR

```
var newTodoListHTML = `${todoListHTML}<li>${inputText}</li>`
```

Now its just a matter of resetting.

```
var todoList = document.getElementById('todo-list')
var inputField = document.getElementById('input-field')
var addTodoButton = document.getElementById('add-todo')

function addItemToList(e) {
    var inputText = inputField.value
    var todoListHTML = todoList.innerHTML

    var newTodoListHTML = `${todoListHTML}<li>${inputText}</li>`

    todoList.innerHTML = newTodoListHTML
}
addTodoButton.addEventListener('click', addItemToList)
```

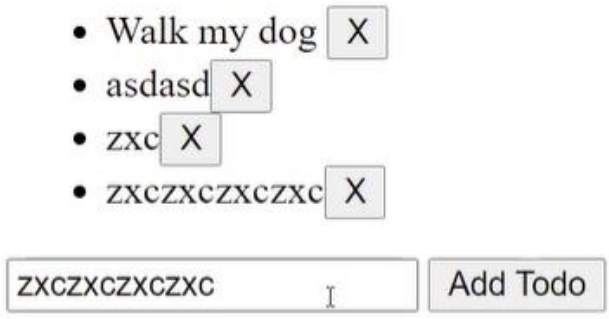
This code will add to the to-do list. We keep adding whatever we want to add. This is a very basic to-do list.

To remove from the to-do list:

We have two li tags. So, to remove it, we need to have a button on which I can click and it gets removed. So, we can have a button next to with ‘x’. When we click on it, the li tag should be removed. By this we can say that for every list item that we add, we also need to add a button element so that we can remove it. if we needed to such behavior instead of just aligning my text, we need to add the button in the newTodoListHTML.

```
var newTodoListHTML = `${todoListHTML}<li>${inputText}</li> <button>x</b
utton>`
```

Output:



We are getting the button, but the issue by clicking the button we need to remove the li tag. We need to get the reference of the x button, so that we can attach it to the javascript file and add logic to it.

To do something like this there is a limitation that I am not able to attach an event listener for this button. As this button is not getting appended when I click the button, I cannot get reference of that button inside my function directly. in order to get that reference, we need to learn few more extra functions that we will be able to utilize.

But now there is something called document.create element. Inside this element we can pass-in any valid HTML tag name like paragraph p, H1, H2, H3, form anything basically what we're trying to do here is we are trying to create a button element.

We will pass-in a button and this will return me removeButton.

It will create a button and return it and put it in my variable. do notice that createElement will not actually append my button. It will not put it in my document by default. It will create an element in the memory and then returned it so that you can do something out of it. kya ho raha hai kya hai Iske andar etc etc.,

Remove button refresh, we get a button element

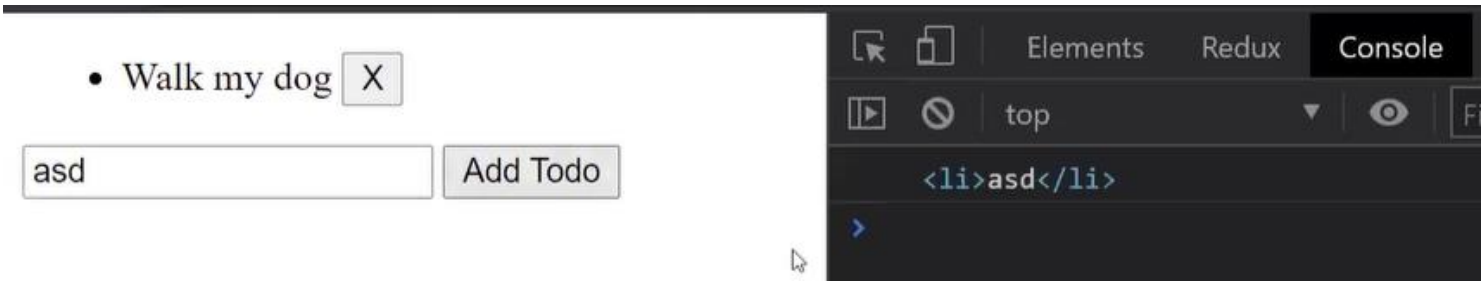
Output:

We got the button element in the console.

Now, we can do anything we want with this removeButton element. I will do removeButton.innerText = 'x'; referring to the x that we want from the HTML page. Now when I console I get text along with my button. I have now lost my ability to properly appended to the element. There is another way for appending to my list. like I have did this "removeButton", can we not create an li item also.

```
    var newLiTag = document.createElement('li')
    console.log(newLiTag)
```

Since the newLiTag is empty, we will add the inputText. We will get the list item along with the input text.

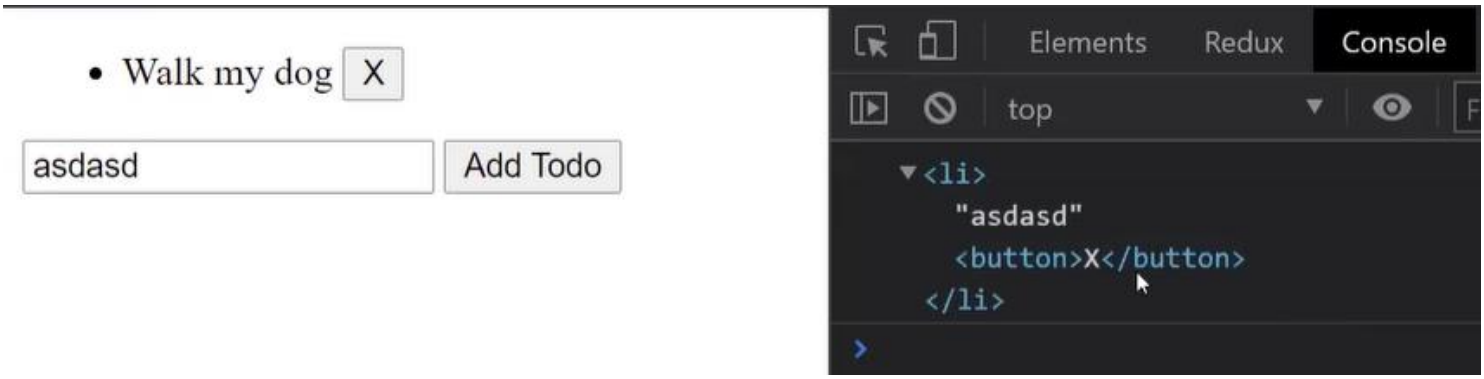


All we need to do now is append it to the list.

We should have the text and the button element in it. There is a method called append and appendChild. Inside this using appendChild() appendchild inside this function which is called upon the newLiTag we'll pass in this new removeButton.

```
    newLiTag.appendChild(removeButton)
```

And now if you look I have the text as well as the button.



This is a better way but for simple use cases string contcatenation will also work. Now, we want to append this newLiTag into toDoListHTML. Similarly, what we will be doing,

```
    todoList.appendChild(newLiTag)
```

What we are doing, just for brief revision we are creating an element 'removeButton' and we are setting inside the button a text as 'x'. A newLiTag is created and we are putting whatever text the user inputs. li tag has only text now I want to add this button also inside it. it already has the text so we will call the appendChild which already has a HTML reference

Now, all we have to do is to append the newLiTag into the todoList. So, whatever is there in my newLiTag will get appended when you call the function.

```
function addItemToList(){
    var inputText = inputField.value
    var todoListHTML = todoList.innerHTML

    // var newTodoListHTML = todoListHTML + '<li>'+inputText+'</li>'
```

```
    var newTodoListHTML = `${todoListHTML}<li>${inputText}</li> <button>
x</button>`
    todoList.innerHTML = newTodoListHTML

    // creating remove button.
    var removeButton = document.createElement('button')
    removeButton.innerText = 'x'

    removeButton.addEventListener('click', removeItem)

    // created a new li tag
    var newLiTag = document.createElement('li')
    newLiTag.innerText = inputText

    //appending remove button element o the li that was created right no
w
    newLiTag.appendChild(removeButton)

    // appending the new <li> with the button in it to the todo list
    todoList.appendChild(newLiTag)

    console.log(newLiTag)
```

We were creating the input text and button and all through the textual way, string concatenation way. Now we are doing it through document.createElement. So, just as we have all the reference to the html, same way when I hover over, it is also says as HTML button element. It gives me the ability to attach it to the eventListener the textual representation of the string concatenation so in a textual way of adding the element we are limited and we cannot add event listener to the 'x' button.

I need an eventListener so that I can tell you what should happen if I click or if the user Clicks on this button. I need to it that for all the elements that are going to be present in my in the list; already present as well new item added.

If we click on an element only it has to be removed. Since we have created this removeButton dynamically here, I can also do something such as event attachment or just event to the removeButton.

```
removeButton.addEventListener()
```

Now that I've created my button and I have taken the text inside it know what I am doing, I am going on click of this removeButton and every time it will be referring to that list item only in that list item inside that on click and I will pass the function for removeItem.

```
removeButton.addEventListener('click', removeItem)
```

We will remove it and will have empty unordered list. We add it and click on ‘x’, we can see in the console log ‘trying to remove item’. Now remove item you what this is 'e'. We probably know this is an event object but I want to show something more inside that event object.

We get the event logged out. It has many properties; one of the properties is called as target When we expand upon it, are hover upon this, you will notice the same item is getting highlighted.

Now if we create another item, "pasd" and added to the list. Now when we click on the X button, we will get another event. If we expand it and hover over the target: button, it gets highlighted. Target also has many properties, one of them being parentElement: li, when we hover over li element, the entire element gets highlighted. we are going to use this or capitalize upon it to achieve our goal.

Each of the event has the target property and this target property will always refer on which it was fired. The target property in our case is always the button. Target element will always have a parent element refer to the first level element. The element is button, the parent of the button is li tag. Similarly, for li, the parent element would be ul.

All I have to do now is to call a function remove on that parent element. That will get removed from the html document itself.

```
function removeItem(e) {
    e.target.parentElement.remove()
```

So, the functionality of what we are trying is achieved.