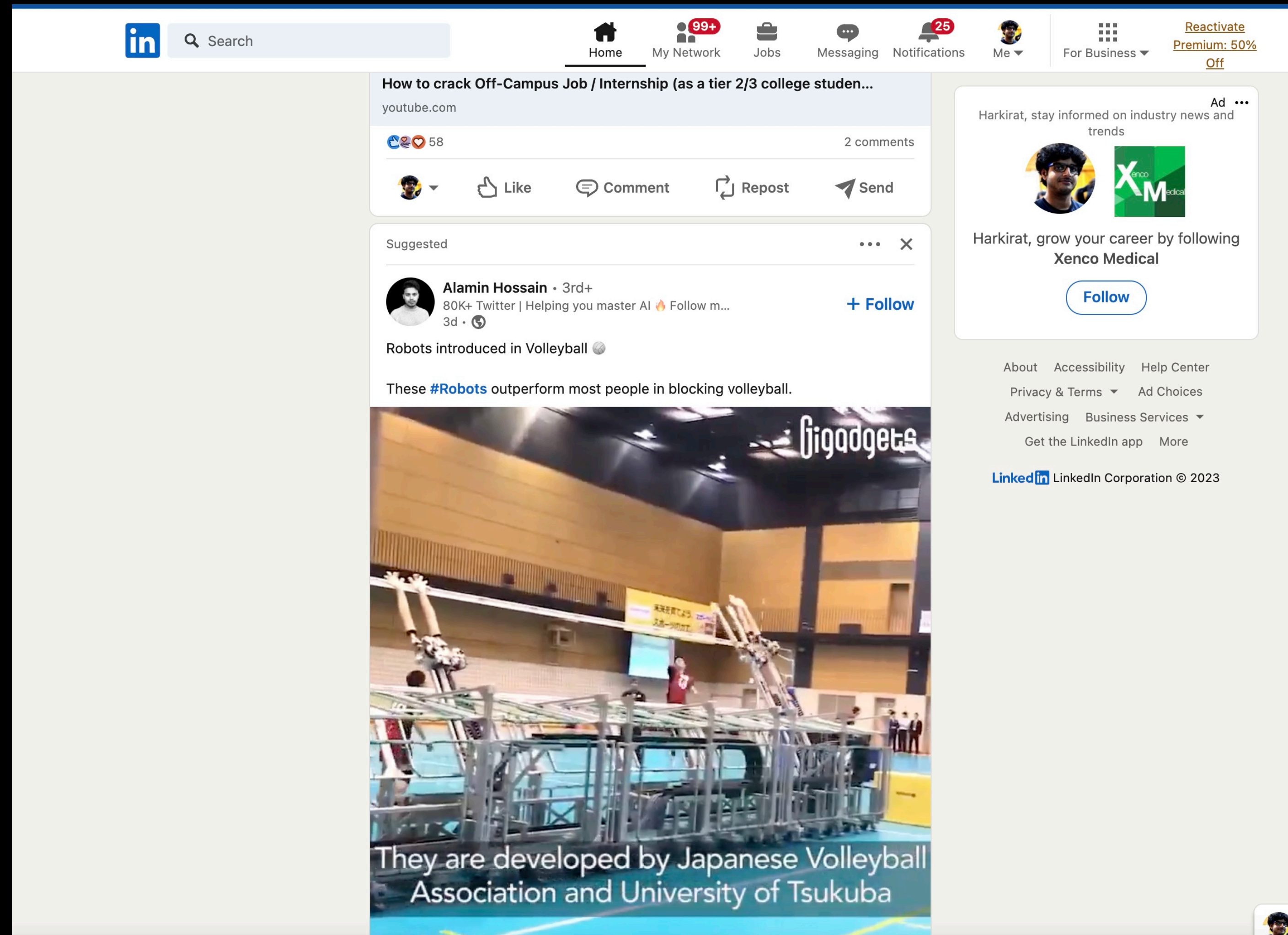


4.2 - Why frontend frameworks

Reconcilers and Intro to React

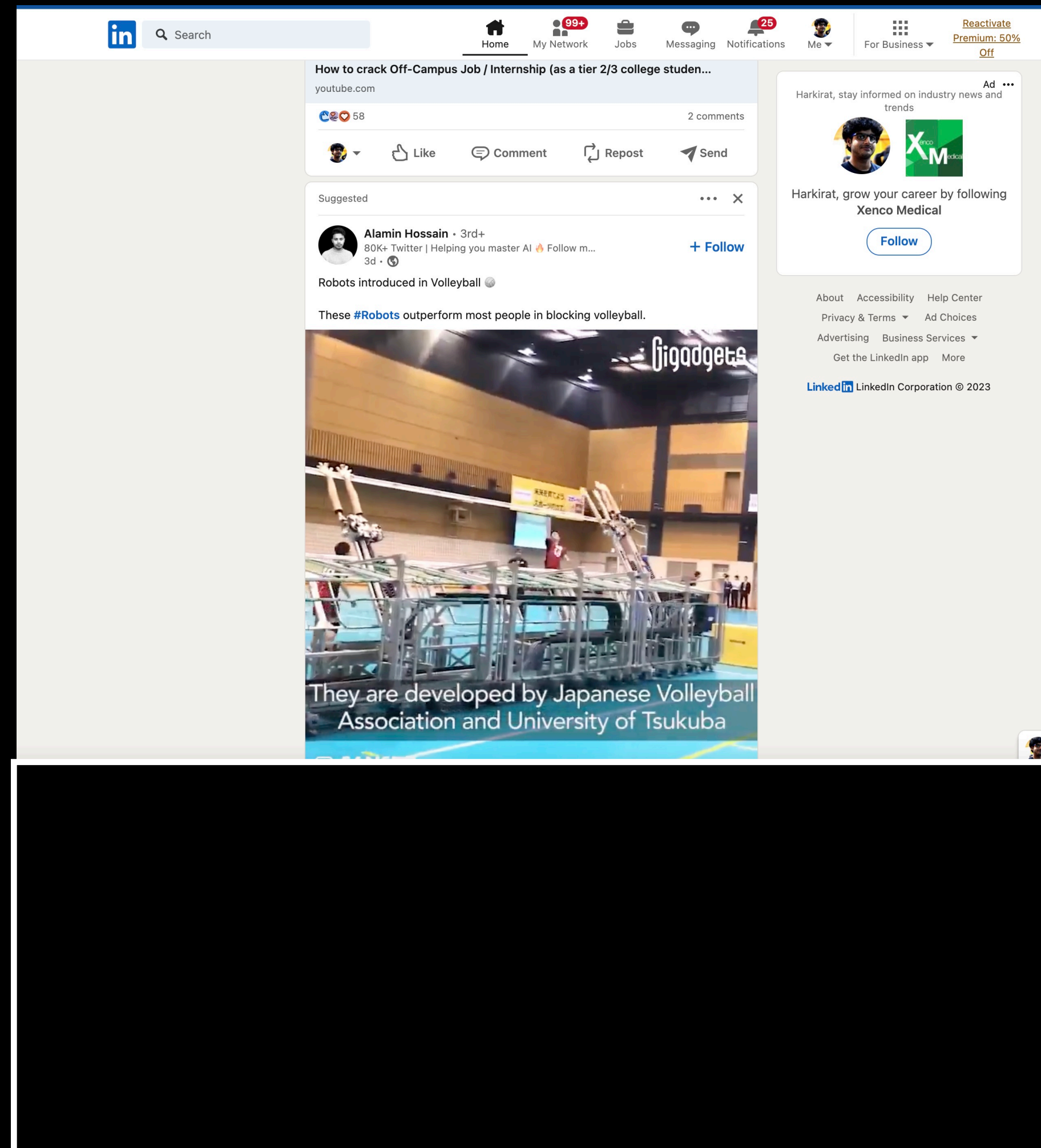
Jargon - DOM Manipulation

What happens when you scroll down



Jargon - DOM Manipulation

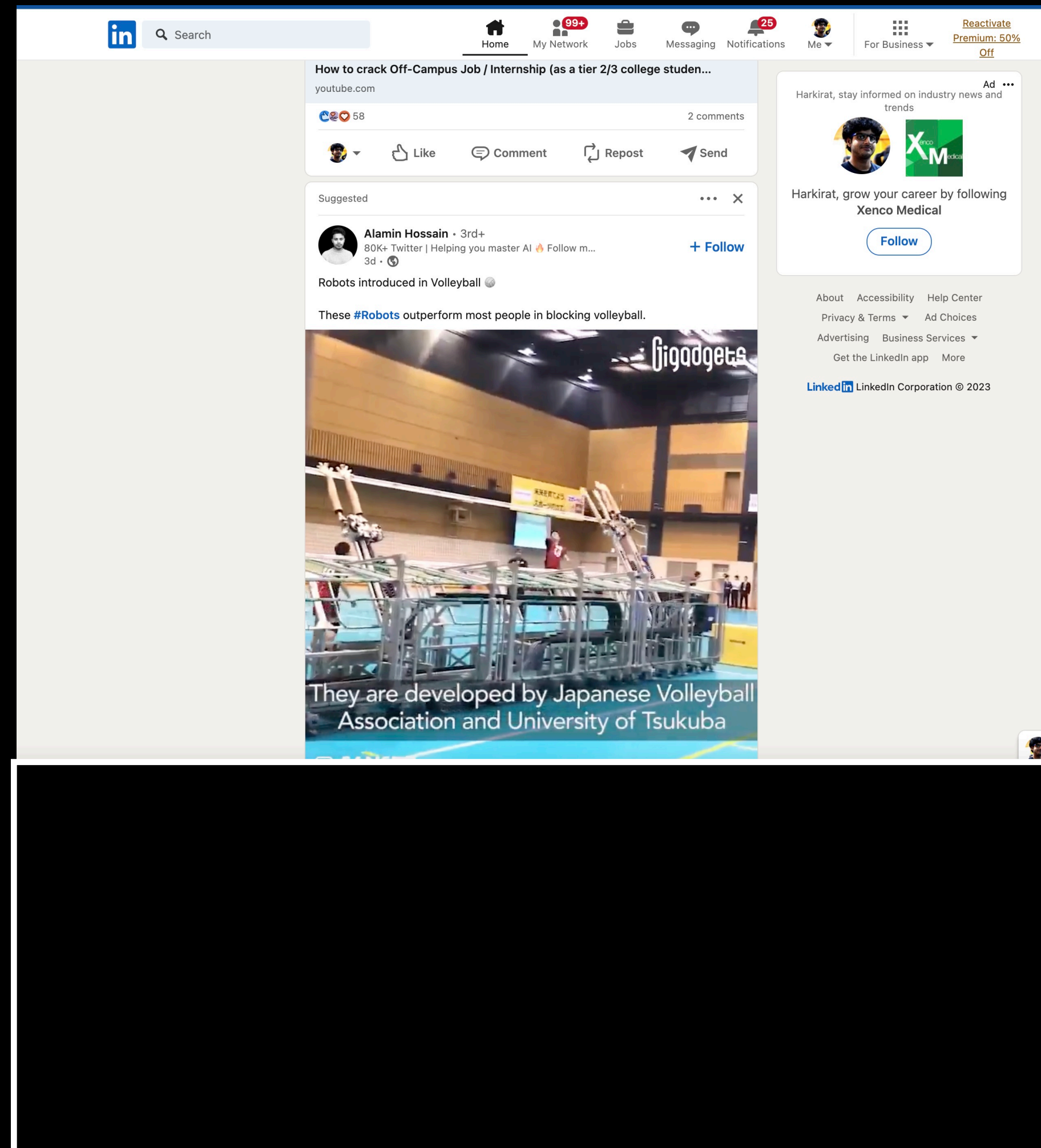
Jargon - DOM Manipulation



getNextPosts

New Items get pushed
to the DOM

Jargon - DOM Manipulation



getNextPosts

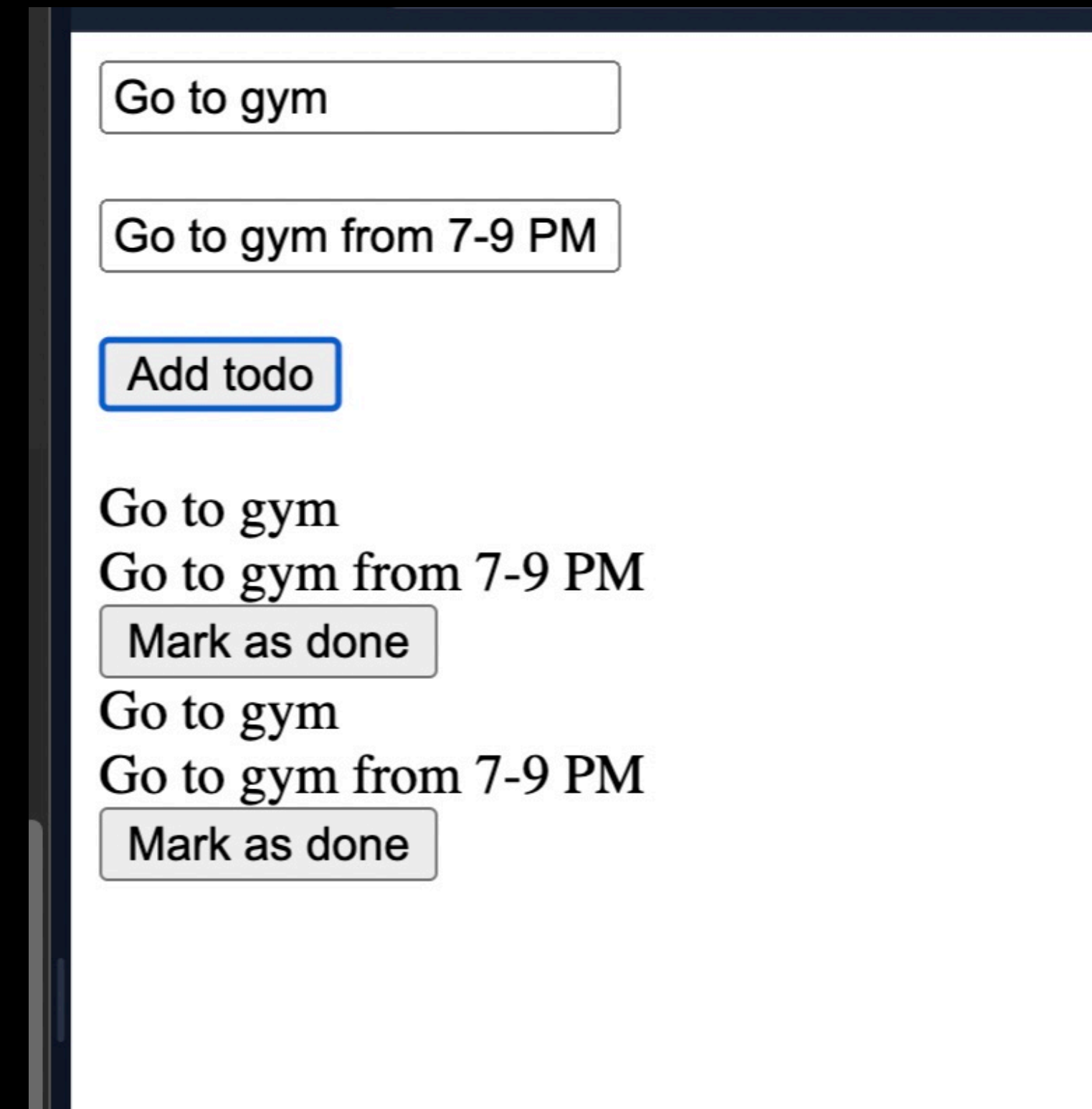
New Items get pushed
to the DOM

This is called **DOM manipulation**

DOM manipulation is very hard to write as a developer
Making dynamic websites, with the primitives that DOM provides you is very hard

Let's try to create a TODO application using DOM manipulation

```
document.createElement  
document.appendChild  
element.setAttribute  
element.children
```



<https://gist.github.com/hkirat/e61655a7d93ce06810488be402adebee>

Let's try to create a TODO application using DOM manipulation

JS

```
<script>
  let globalId = 1;

  function markAsDone(todoId) {
    const parent = document.getElementById(todoId);
    parent.children[2].innerHTML = "Done!"
  }

  function createChild(title, description, id) {
    const child = document.createElement("div");
    const firstGrandParent = document.createElement("div");
    firstGrandParent.innerHTML = title;
    const secondGrandParent = document.createElement("div");
    secondGrandParent.innerHTML = description;
    const thirdGrandParent = document.createElement("button");
    thirdGrandParent.innerHTML = "Mark as done";
    thirdGrandParent.setAttribute("onclick", `markAsDone(${id})`);
    child.appendChild(firstGrandParent);
    child.appendChild(secondGrandParent);
    child.appendChild(thirdGrandParent);
    child.setAttribute("id", id);
    return child;
  }

  function addTodo() {
    const title = document.getElementById("title").value;
    const description = document.getElementById("description").value;
    const parent = document.getElementById("todos");

    parent.appendChild(createChild(title, description, globalId++));
  }
</script>
```

HTML

```
<body>
  <input type="text" id="title" placeholder="Todo title"></input> <br /><br />
  <input type="text" id="description" placeholder="Todo description"></input> <br /><br />
  <button onclick="addTodo()">Add todo</button>
  <br /> <br />

  <div id="todos">

  </div>
</body>
```


Let's try to create a TODO application using DOM manipulation

Problem with this approach - Very hard to add and remove elements

No central **State**

What if there is a server where these todos are put

What if you update a TODO from your mobile app

You will get back the new array of TODOs on the frontend

How will you update the DOM then?

You only have a **addTodo** function

You don't have an **updateTodo** or **removeTodo** function yet

The screenshot displays a web application interface for a TODO list. It is divided into two horizontal sections. The top section represents the initial state, featuring two input fields with the text "Go to gym" and "Go to gym from 7-9 PM", followed by a blue-outlined "Add todo" button. The bottom section represents the state after an update, showing the same two TODO items, but each now has a grey "Mark as done" button next to it. This visualizes the problem of updating the DOM when only an "add" function is available.

Let's try to create a TODO application using DOM manipulation

What do I mean when I say **State**

```
[
  {
    Id: 1,
    title: "Go to Gym",
    description: "Go to Gym from 7-9 PM"
  },
  {
    id: 1,
    title: "Go to Gym",
    description: "Go to Gym from 7-9 PM"
  },
]
```

State

Go to gym

Go to gym from 7-9 PM

Add todo

Go to gym

Go to gym from 7-9 PM

Mark as done

Go to gym

Go to gym from 7-9 PM

Mark as done

Let's try to create a TODO application using DOM manipulation

What do I mean when I say **State**

If you can write a function, that takes this state as an input and creates the output on the right, that is much more powerful than our original approach

```
state
[
  {
    Id: 1,
    title: "Go to Gym",
    description: "Go to Gym from 7-9 PM"
  },
  {
    id: 1,
    title: "Go to Gym",
    description: "Go to Gym from 7-9 PM"
  },
]
```

Go to gym

Go to gym from 7-9 PM

Add todo

Go to gym

Go to gym from 7-9 PM

Mark as done

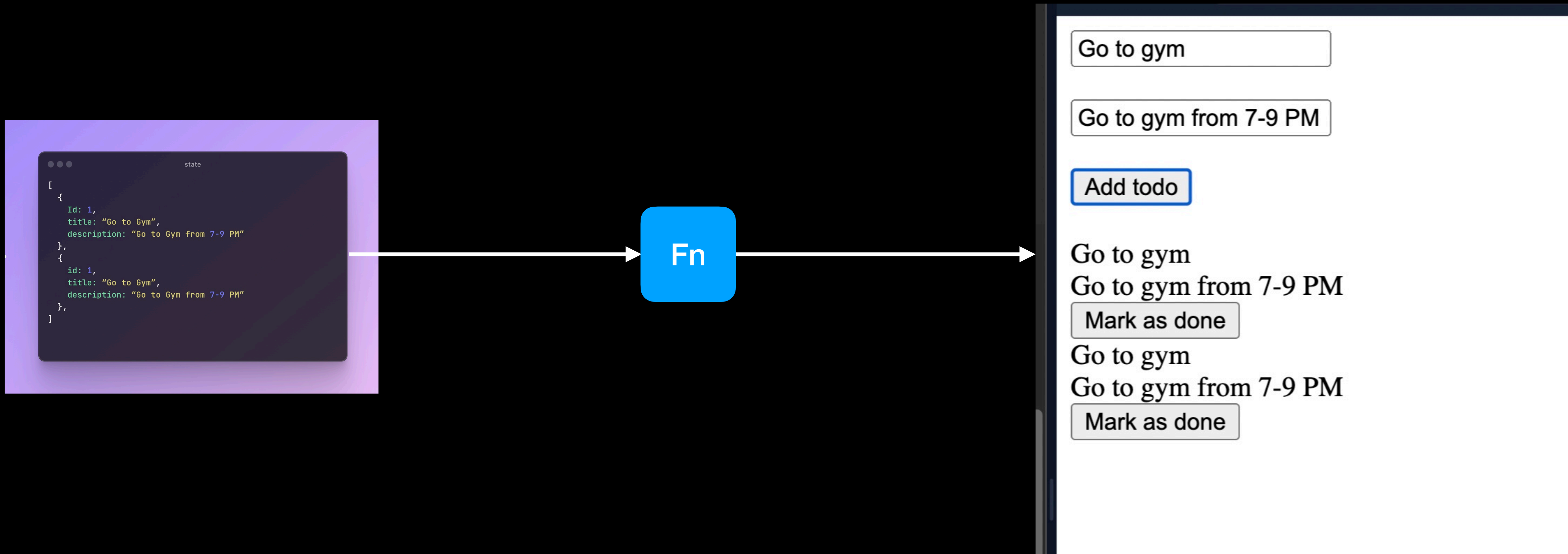
Go to gym

Go to gym from 7-9 PM

Mark as done

Let's try to create a TODO application using DOM manipulation

What do I mean when I say **State**



Let's try to create a TODO application using DOM manipulation

What do I mean when I say **State**

```
Untitled-1

<!DOCTYPE html>
<html>

<head>
  <script>
    let todoState = [];
    function addTodo() {
      const title = document.getElementById("title").value;
      const description = document.getElementById("description").value;
      todoState.push({
        title: title,
        description: description,
        id: globalId++
      })
      updateState(todoState);
    }
  </script>
</head>

<body>
  <input type="text" id="title" placeholder="Todo title"></input> <br /><br />
  <input type="text" id="description" placeholder="Todo description"></input> <br /><br />
  <button onclick="addTodo()">Add todo</button>
  <br /> <br />

  <div id="todos">

  </div>
</body>

</html>
```

Go to gym

Go to gym from 7-9 PM

Add todo

Go to gym

Go to gym from 7-9 PM

Mark as done

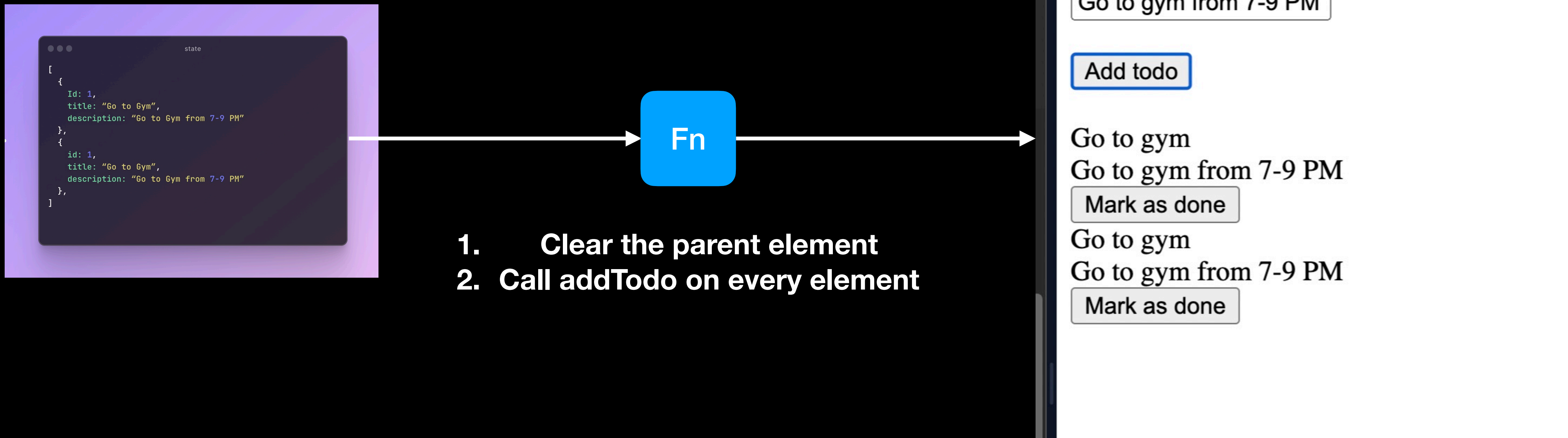
Go to gym

Go to gym from 7-9 PM

Mark as done

Let's try to create a TODO application using DOM manipulation

Dumb Solution



<https://gist.github.com/hkirat/cb9d7e2f75617ac281427276e20a691c>

Let's try to create a TODO application using DOM manipulation

Better Solution

Don't clear the DOM upfront, update it based on what has changed.

Question is, how does it calculate what all has changed?

Has a todo been marked as complete?

Has a todo been removed from the backend?

Let's try to create a TODO application using DOM manipulation

Better Solution

Don't clear the DOM upfront, update it based on what has changed.

Question is, how does it calculate what all has changed?

Has a todo been marked as complete?

Has a todo been removed from the backend?

By remembering the old todos in a variable (Virtual DOM)

Let's try to create a TODO application using DOM manipulation

Better Solution (Take home assignment)

```
<body>
  <input type="text" id="title" placeholder="Todo title"></input> <br /><br />
  <input type="text" id="description" placeholder="Todo description"></input> <br
/><br />
  <button onClick="addTodo()">Add todo</button>
  <br /> <br />

  <div id="todos">

</div>
</body>
```

<https://gist.github.com/hkirat/ed34df967f162d152e35537cb8215144>

```
<!DOCTYPE html>
<html>

<head>
  <script>
    let globalId = 1;
    let todoState = [];
    let oldTodoState = [];

    function addTodo() {
      // big function we wrote in the beginning
    }

    function removeTodo(todo) {
      const element = document.getElementById(todo.id);
      element.parent.removeChild(element);
    }

    function updateTodo(oldTodo, newTodo) {
      const element = document.getElementById(oldTodo.id);
      element.children[0].innerHTML = newTodo.title;
      element.children[1].innerHTML = newTodo.description;
      element.children[0].innerHTML = newTodo.completed ? "Mark as done" : "Done";
    }

    function updateState(newTodos) {
      // calculate the diff b/w newTodos and oldTodos.
      // More specifically, find out what todos are -
      // 1. added
      // 2. deleted
      // 3. updated
      const added = [];
      const deleted = [];
      const updated = [];
      // calculate these 3 arrays
      // call addTodo, removeTodo, updateTodo functions on each of the
      // elements
      oldTodoState = newTodos;
    }

    function addTodo() {
      const title = document.getElementById("title").value;
      const description = document.getElementById("description").value;
      todoState.push({
        title: title,
        description: description,
        id: globalId++,
      })
      updateState(todoState);
    }
  </script>

</html>
```


What is the easiest way to create a dynamic frontend website?

1. Update a state variable
2. Delegate the task of figuring out diff to a hefty function
3. Tell the hefty function how to add, update and remove elements

```
<!DOCTYPE html>
<html>

<head>
  <script>
    let globalId = 1;
    let todoState = [];
    let oldTodoState = [];

    function addTodo() {
      // big function we wrote in the beginning
    }

    function removeTodo(todo) {
      const element = document.getElementById(todo.id);
      element.parent.removeChild(element);
    }

    function updateTodo(oldTodo, newTodo) {
      const element = document.getElementById(oldTodo.id);
      element.children[0].innerHTML = newTodo.title;
      element.children[1].innerHTML = newTodo.description;
      element.children[0].innerHTML = newTodo.completed ? "Mark as done" : "Done";
    }

    function updateState(newTodos) {
      // calculate the diff b/w newTodos and oldTodos.
      // More specifically, find out what todos are -
      // 1. added
      // 2. deleted
      // 3. updated
      const added = [];
      const deleted = [];
      const updated = [];
      // calculate these 3 arrays
      // call addTodo, removeTodo, updateTodo functions on each of the
      // elements
      oldTodoState = newTodos;
    }

    function addTodo() {
      const title = document.getElementById("title").value;
      const description = document.getElementById("description").value;
      todoState.push({
        title: title,
        description: description,
        id: globalId++,
      })
      updateState(todoState);
    }
  </script>

</html>
```

What is the easiest way to create a dynamic frontend website?

1. **Update a state variable**
2. Delegate the task of figuring out diff to a hefty function
3. Tell the hefty function how to add, update and remove elements

```
<!DOCTYPE html>
<html>

<head>
  <script>
    let globalId = 1;
    let todoState = [];
    let oldTodoState = [];

    function addTodo() {
      // big function we wrote in the beginning
    }

    function removeTodo(todo) {
      const element = document.getElementById(todo.id);
      element.parent.removeChild(element);
    }

    function updateTodo(oldTodo, newTodo) {
      const element = document.getElementById(oldTodo.id);
      element.children[0].innerHTML = newTodo.title;
      element.children[1].innerHTML = newTodo.description;
      element.children[0].innerHTML = newTodo.completed ? "Mark as done" : "Done";
    }

    function updateState(newTodos) {
      // calculate the diff b/w newTodos and oldTodos.
      // More specifically, find out what todos are -
      // 1. added
      // 2. deleted
      // 3. updated
      const added = [];
      const deleted = [];
      const updated = [];
      // calculate these 3 arrays
      // call addTodo, removeTodo, updateTodo functions on each of the
      // elements
      oldTodoState = newTodos;
    }

    function addTodo() {
      const title = document.getElementById("title").value;
      const description = document.getElementById("description").value;
      todoState.push({
        title: title,
        description: description,
        id: globalId++,
      })
      updateState(todoState);
    }
  </script>

</html>
```

What is the easiest way to create a dynamic frontend website?

1. Update a state variable
2. Delegate the task of figuring out diff to a hefty function
3. Tell the hefty function how to add, update and remove elements

```
<!DOCTYPE html>
<html>

<head>
  <script>
    let globalId = 1;
    let todoState = [];
    let oldTodoState = [];

    function addTodo() {
      // big function we wrote in the beginning
    }

    function removeTodo(todo) {
      const element = document.getElementById(todo.id);
      element.parent.removeChild(element);
    }

    function updateTodo(oldTodo, newTodo) {
      const element = document.getElementById(oldTodo.id);
      element.children[0].innerHTML = newTodo.title;
      element.children[1].innerHTML = newTodo.description;
      element.children[0].innerHTML = newTodo.completed ? "Mark as done" : "Done";
    }

    function updateState(newTodos) {
      // calculate the diff b/w newTodos and oldTodos.
      // More specifically, find out what todos are -
      // 1. added
      // 2. deleted
      // 3. updated
      const added = [];
      const deleted = [];
      const updated = [];
      // calculate these 3 arrays
      // call addTodo, removeTodo, updateTodo functions on each of the
      // elements
      oldTodoState = newTodos;
    }

    function addTodo() {
      const title = document.getElementById("title").value;
      const description = document.getElementById("description").value;
      todoState.push({
        title: title,
        description: description,
        id: globalId++,
      })
      updateState(todoState);
    }
  </script>
</html>
```


What is the easiest way to create a dynamic frontend website?

1. Update a state variable
2. Delegate the task of figuring out diff to a hefty function
3. Tell the hefty function how to add, update and remove elements

```
<!DOCTYPE html>
<html>

<head>
  <script>
    let globalId = 1;
    let todoState = [];
    let oldTodoState = [];

    function addTodo() {
      // big function we wrote in the beginning
    }

    function removeTodo(todo) {
      const element = document.getElementById(todo.id);
      element.parent.removeChild(element);
    }

    function updateTodo(oldTodo, newTodo) {
      const element = document.getElementById(oldTodo.id);
      element.children[0].innerHTML = newTodo.title;
      element.children[1].innerHTML = newTodo.description;
      element.children[0].innerHTML = newTodo.completed ? "Mark as done" : "Done";
    }

    function updateState(newTodos) {
      // calculate the diff b/w newTodos and oldTodos.
      // More specifically, find out what todos are -
      // 1. added
      // 2. deleted
      // 3. updated
      const added = [];
      const deleted = [];
      const updated = [];
      // calculate these 3 arrays
      // call addTodo, removeTodo, updateTodo functions on each of the
      // elements
      oldTodoState = newTodos;
    }

    function addTodo() {
      const title = document.getElementById("title").value;
      const description = document.getElementById("description").value;
      todoState.push({
        title: title,
        description: description,
        id: globalId++,
      })
      updateState(todoState);
    }
  </script>
</html>
```

Usually done by the FE developer

What is the easiest way to create a dynamic frontend website?

1. **Update a state variable**
2. Delegate the task of figuring out diff to a hefty function
3. **Tell the hefty function how to add, update and remove elements**

```
<!DOCTYPE html>
<html>

<head>
  <script>
    let globalId = 1;
    let todoState = [];
    let oldTodoState = [];

    function addTodo() {
      // big function we wrote in the beginning
    }

    function removeTodo(todo) {
      const element = document.getElementById(todo.id);
      element.parent.removeChild(element);
    }

    function updateTodo(oldTodo, newTodo) {
      const element = document.getElementById(oldTodo.id);
      element.children[0].innerHTML = newTodo.title;
      element.children[1].innerHTML = newTodo.description;
      element.children[0].innerHTML = newTodo.completed ? "Mark as done" : "Done";
    }

    function updateState(newTodos) {
      // calculate the diff b/w newTodos and oldTodos.
      // More specifically, find out what todos are -
      // 1. added
      // 2. deleted
      // 3. updated
      const added = [];
      const deleted = [];
      const updated = [];
      // calculate these 3 arrays
      // call addTodo, removeTodo, updateTodo functions on each of the
      // elements
      oldTodoState = newTodos;
    }

    function addTodo() {
      const title = document.getElementById("title").value;
      const description = document.getElementById("description").value;
      todoState.push({
        title: title,
        description: description,
        id: globalId++,
      })
      updateState(todoState);
    }
  </script>

</html>
```

Usually done by the React

What is the easiest way to create a dynamic frontend website?

1. Update a state variable
2. Delegate the task of figuring out diff to a hefty function
3. Tell the hefty function how to add, update and remove elements

```
<!DOCTYPE html>
<html>

<head>
  <script>
    let globalId = 1;
    let todoState = [];
    let oldTodoState = [];

    function addTodo() {
      // big function we wrote in the beginning
    }

    function removeTodo(todo) {
      const element = document.getElementById(todo.id);
      element.parent.removeChild(element);
    }

    function updateTodo(oldTodo, newTodo) {
      const element = document.getElementById(oldTodo.id);
      element.children[0].innerHTML = newTodo.title;
      element.children[1].innerHTML = newTodo.description;
      element.children[0].innerHTML = newTodo.completed ? "Mark as done" : "Done";
    }

    function updateState(newTodos) {
      // calculate the diff b/w newTodos and oldTodos.
      // More specifically, find out what todos are -
      // 1. added
      // 2. deleted
      // 3. updated
      const added = [];
      const deleted = [];
      const updated = [];
      // calculate these 3 arrays
      // call addTodo, removeTodo, updateTodo functions on each of the
      // elements
      oldTodoState = newTodos;
    }

    function addTodo() {
      const title = document.getElementById("title").value;
      const description = document.getElementById("description").value;
      todoState.push({
        title: title,
        description: description,
        id: globalId++,
      })
      updateState(todoState);
    }
  </script>
</html>
```

Let's see how you would do the same thing in React

```
npm create vite@latest
```