

Event Handler Tutorial

In this tutorial, you'll take what you learned from the reading material and build on the Todo application. You can find the starting code for this tutorial in the `todo` folder. The Todo application might look familiar because it's where you finished up in the previous tutorial.

DOM content loaded

In the reading, you learned about an event called `DOMContentLoaded`. The `DOMContentLoaded` event fires when the initial HTML document has been completely loaded and parsed, without waiting for stylesheets, images, and subframes to finish loading.

In the current application, you have the following code at the bottom of the application:

```
init();
addPageTitle();
addTodos();
```

This code seems like a good candidate to run only once the `DOMContentLoaded` has fired. The first thing you'll do is add an event listener and move that code into the event handler.

```
document.addEventListener("DOMContentLoaded", () => {
  init();
  addPageTitle();
  addTodos();
});
```

Marking tasks complete and incomplete

Now that your application is running, you'll add some new features to it. The first thing you'll do is allow users to click a task to mark it completed. You must also give users the ability to double-click a task and mark it incomplete. All of the markup and styles are in place, so now you need to add the appropriate JavaScript.

Marking tasks complete

You want to add a `click` event listener to each of the tasks. To accomplish this, you need to get a reference to each of the list items:

```
const tasks = document.querySelectorAll('li');
```

Next, you'll loop over each of the tasks and add a click event listener to each task (``). To mark a task complete, you'll add the class `completed` to both the list item and the icon. Before you add the completed class, check to make sure that it already isn't complete:

```
tasks.forEach((task) => {  
  task.addEventListener('click', () => {  
    if (!task.classList.contains('completed')) {  
  
    }  
  });  
});
```

If it's currently incomplete, you can add the class `completed` to the task (``) and the icon:

```
tasks.forEach((task) => {  
  task.addEventListener('click', () => {  
    if (!task.classList.contains('completed')) {  
      task.classList.add('completed');  
      task.querySelector('i').classList.add('completed');  
    }  
  });  
});
```

Now, you can click a task and mark it complete.

Marking tasks incomplete

To mark a task incomplete you'll do something very similar. Since you already have a list of all the tasks and are already looping over them, you can add your code there:

```
tasks.forEach((task) => {  
  task.addEventListener('click', () => {  
    if (!task.classList.contains('completed')) {  
      task.classList.add('completed');  
      task.querySelector('i').classList.add('completed');  
    }  
  });  
  // add double click event listener here  
});
```

You'll use the same approach as you did before, but this time you'll listen for the `dblclick` event. You only want to run the event handler logic if the task is already marked complete:

```
task.addEventListener('dblclick', () => {  
  if (task.classList.contains('completed')) {  
    task.classList.remove('completed');  
    task.querySelector('i').classList.remove('completed');  
  }  
});
```

You can double-click a task and mark it incomplete as long as it was already marked completed.

Mark all completed

Finally, your users want the ability to click the button following the tasks to mark all of them completed. The first step is to get a reference to the button and add a `click` event listener to it:

```
const completeAll = document.getElementById('btnCompleteAll');
completeAll.addEventListener('click', () => {

});
```

You already have a reference to all the tasks, so loop over them and add the `completed` class to the list item and the icon:

```
const completeAll = document.getElementById('btnCompleteAll');
completeAll.addEventListener('click', () => {
  tasks.forEach((task) => {
    task.classList.add('completed');
    task.querySelector('i').classList.add('completed');
  });
});
```

If you run the application, you can now click the button to mark all tasks complete.

Tutorial solution

If you followed everything correctly, you have something that looks like this:

```
/*
 * When the DOM is fully loaded into a browser, the browser itself will trigger an
 * event called
 * DOMContentLoaded on the document object. You'll need to add all of your event
 * listeners inside
 * of an anonymous function that only runs once the DOMContentLoaded event is
 * fired.
 */
document.addEventListener('DOMContentLoaded', () => {
  init();
  addPageTitle();
  addTodos();

  const tasks = document.querySelectorAll('li');

  tasks.forEach((task) => {
    // when you click on a task, mark it completed
    task.addEventListener('click', () => {
```

```
    if (!task.classList.contains('completed')) {
      task.classList.add('completed');
      task.querySelector('i').classList.add('completed');
    }
  });

  // when you double click a task, remove the completed class
  task.addEventListener('dblclick', () => {
    if (task.classList.contains('completed')) {
      task.classList.remove('completed');
      task.querySelector('i').classList.remove('completed');
    }
  });
});

// mark all tasks as completed
const completeAll = document.getElementById('btnCompleteAll');
completeAll.addEventListener('click', () => {
  tasks.forEach((task) => {
    task.classList.add('completed');
    task.querySelector('i').classList.add('completed');
  });
});
});
```