Event Handler Tutorial

In this tutorial you are going to take what you learned from the reading material and build on the Todo application. The starting code for this tutorial is located in this directory, in the `todo` folder. The Todo app should look familiar because it is 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 just sitting there 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 are going to do is add an event listener and move that code into the event handler.

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

Marking tasks Complete & Incomplete

Now that your application is running you are going to add some new features to it. You have been asked to allow users to click on a task and mark it completed. You have also been given a requirement to allow users the ability to double click on a task a task and mark it incomplete. All of the markup and styles are in place, you just 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 will first need to get a reference to each of the list items.

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

Next you will loop over each of the tasks and add a click event listener to each task (`<li></li>`). To mark a task complete you're basically adding the class `.completed` to both the list item and the icon. Before you add the completed class it would be good practice to first make sure that it already isn't complete.

```
tasks.forEach((task) => {
  task.addEventListener('click', () => {
    if( !task.hasAttribute('class', 'completed') ) {

    }
  })
}
```

If it's currently incomplete you can add the class `.completed` to the task (`<li></li>`) and the icon.

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

And that is all there is to it, you should be able to click on a task and mark it complete.

Marking tasks incomplete

To mark a task incomplete you are going to 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.hasAttribute('class', 'completed') ) {
        task.classList.add('completed')
        task.querySelector('i').classList.add('completed')
    }
  })
  // add double click event listener here
}
```

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

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

You should be able to double click on 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 on the button below the tasks to mark all of them completed. The first thing you will need to do 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 just 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 should be able to click on the mark all completed.

Tutorial Solution

If you followed everything correctly you should 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. What you need to do is 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.hasAttribute('class', '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.hasAttribute('class','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')
    })
  })

});
```