Notes for submission .2

Please see below notes adressing the issues of submission 1.

1: Module imports

```
There should be only a single script tag in the file index.html
The modules should manage the dependecies themselves using import.
That is, only taskview.js should be loaded by index.html.
No need to use defer with script tag if type="module".
```

index.html now only fetches TaskView.js via script tag.
The index.html script tag no longer has a 'defer' attribute, since 'type = module' script tags are always deferred.

Other modules are imported in TaskView.js:

- ApiClient is exported as a default and imported as a named constant.
- TaskBox and TaskList are not exported and imported per se, but added as global custom elements when their respective files are imported.

2: Option element values

```
For the option elements, why not use the status name as the option value?
The values can be DONE, ACTIVE and WAITING, equal to the option text.
```

The code that was distributed with the assignment (TaskView) used '0' for the value of the 'MODIFY' option, so it seemed natural to just continue along that path.

```
taskrow.innerHTML = `
    <tr>
        <td></td>
        <td></td>
        <td>
            <select>
                <option value="0" selected>&lt;Modify&gt;</option>
            </select>
        </td>
        <td><button type="button">Remove</button></td>
    </tr>`;
```

The solution has been refactored to use the status text as suggested. This simplified the code, and fixed the bug mentioned in section 8 of this file.

3: querySelector / querySelectorAll usage

```
Properties like table.tBodies[0].rows are faster than using querySelector/
querySelectorAll.
```

Taskbox and TaskView:

- Removed all usages of querySelector / querySelectorAll

TaskList:

- querySelector is now only used to locate task tablerows for deletion or status updates
- these lookups are performed on the tbody of the table

4: Callback registration

```
The only functionality of addChangestatusCallback, addDeletetaskCallback
and addNewtaskCallback is to register a callback method, and nothing more.

E.g. for , with #newcallback a private field:
addNewtaskCallback(callback) {
    this.#newcallback;
}

or, with #newtaskCallbacks a private Map:
addNewtaskCallback(callback) {
    const callbackId = Symbol("addNewtaskCallback");
    this.#newtaskCallbacks.set(callbackId, callback);
    return callbackId;
}
```

- TaskView has no methods to register callbacks as it is the root component
- TaskList has two callback registrations:
  - changestatusCallback - was OK
  - deletetaskCallback - was OK
- TaskBox has one callback registration method:
  - newtaskCallback
    - Functionality associated with setting up the callback for the 'Add Task' button has been moved to method 'connectedCallback'

5: Boolean operators

```
Use operators in a boolean context to avoid problems.
Example on what to avoid:
    if (this.querySelector("table")) { ... }
Rather, use:
    if (this.querySelector("table") !== null) { ... }

In a boolean context, undefined, 0, Null, an empty text will all
evaluate to false. Without an operator, a misprint in the expression
can therefore be difficult to find, and can give unexpected results.
```

- showTask of Tasklist:
  - old: *if (!root.querySelector("table"))*
  - new: *if (this.#shadow.getElementById("table")===null)*

    - Two instances of *if (window.confirm ...){..}*:
        - *.confirm(...)* returns bool, so these remain as they were

We make use of the ternary operator in TaskList and TaskView.
Both predicates are explicit:

- *return tableEl === null ?*
- *numTasks>0?*

6: 'Await' usage

```
Why do you "await" a method that does not return a Promise?

Using "await" on something that is not a Promise is not necessary. If
"await", and no promise, the browser will encapsulate the return value
in a Promise that resolves immediately.
```

'await' is used in following classes:

- ApiClient
    - It it thought that the usage of await here is OK, as both *fetch()* and *.json()* return promises
- TaskBox
    - No 'await' usages
- TaskList
    - No 'await' usages
- TaskView
    - 'await' is only used as follows:
        - in Promise.All() of connectedCallback
            - all of the contained method calls are async and return promises
        - when registering callbacks for TaskList and TaskBox
            - these are usages of ApiClient methods which are all async and return promises

On review, we couldn't find any inappropriate usages of 'await' in the solution.

7: Taskbox close symbol

```
The TaskBox dialog window does not close when clicking on the close symbol.
```

fixed

8: Task status modification not working

```
Modifying task status does not work.
```

fixed

(this was OK at some point, but a last minute bugfix elsewhere introduced this bug. Changing the value of the option to just use the text simplified everything)

9: Input validation new tasks

```
A task with no title should be ignored and the data must not be sent to the
```

We set a requirement that the task title should be at least 1 character long,
allowing for tasks like 'A', '1' etc.

The 'Add Task' button is now inactive as long as this requirement is not met.

Also the default value of the select is now 'WAITING', rather than beeing blank.