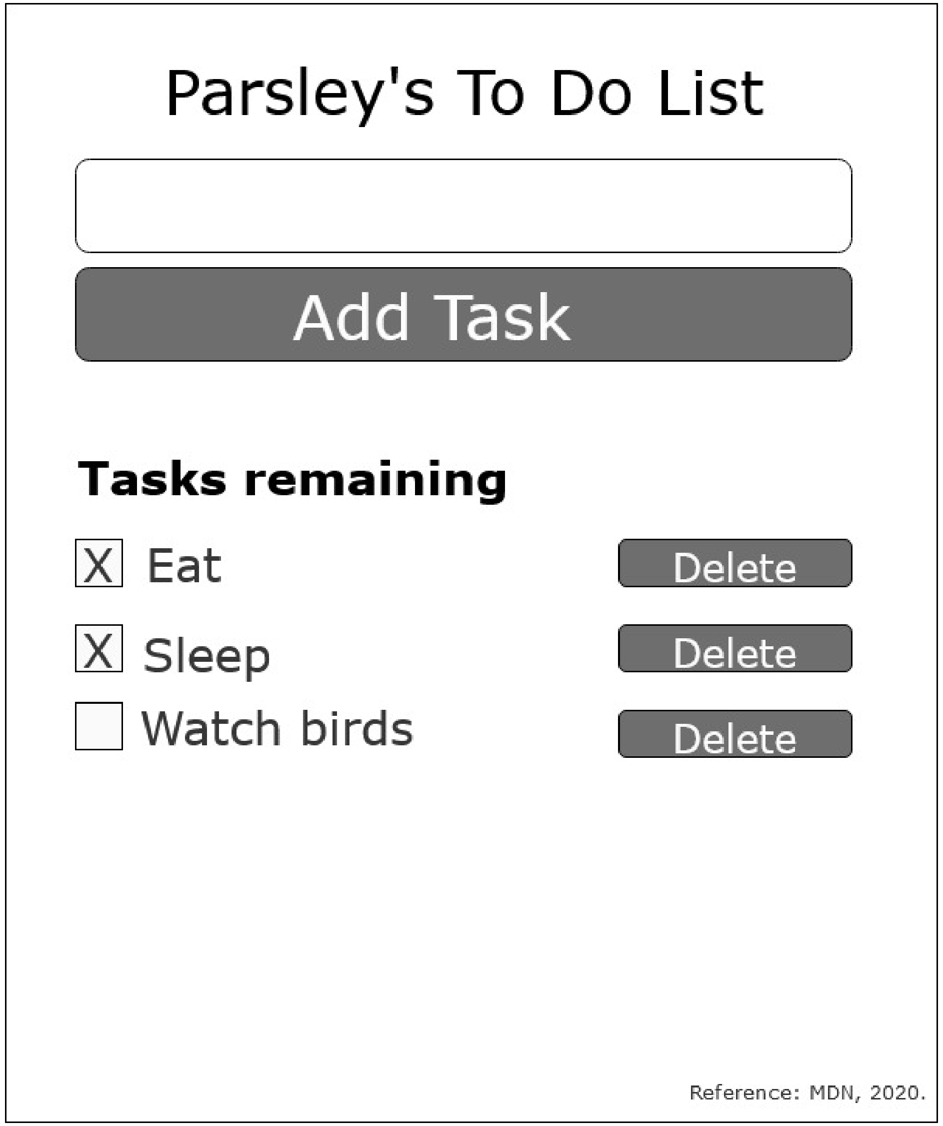
React To-Do Planning Assignment



Parsley's To Do List

Tasks remaining

☒ Eat

☒ Sleep

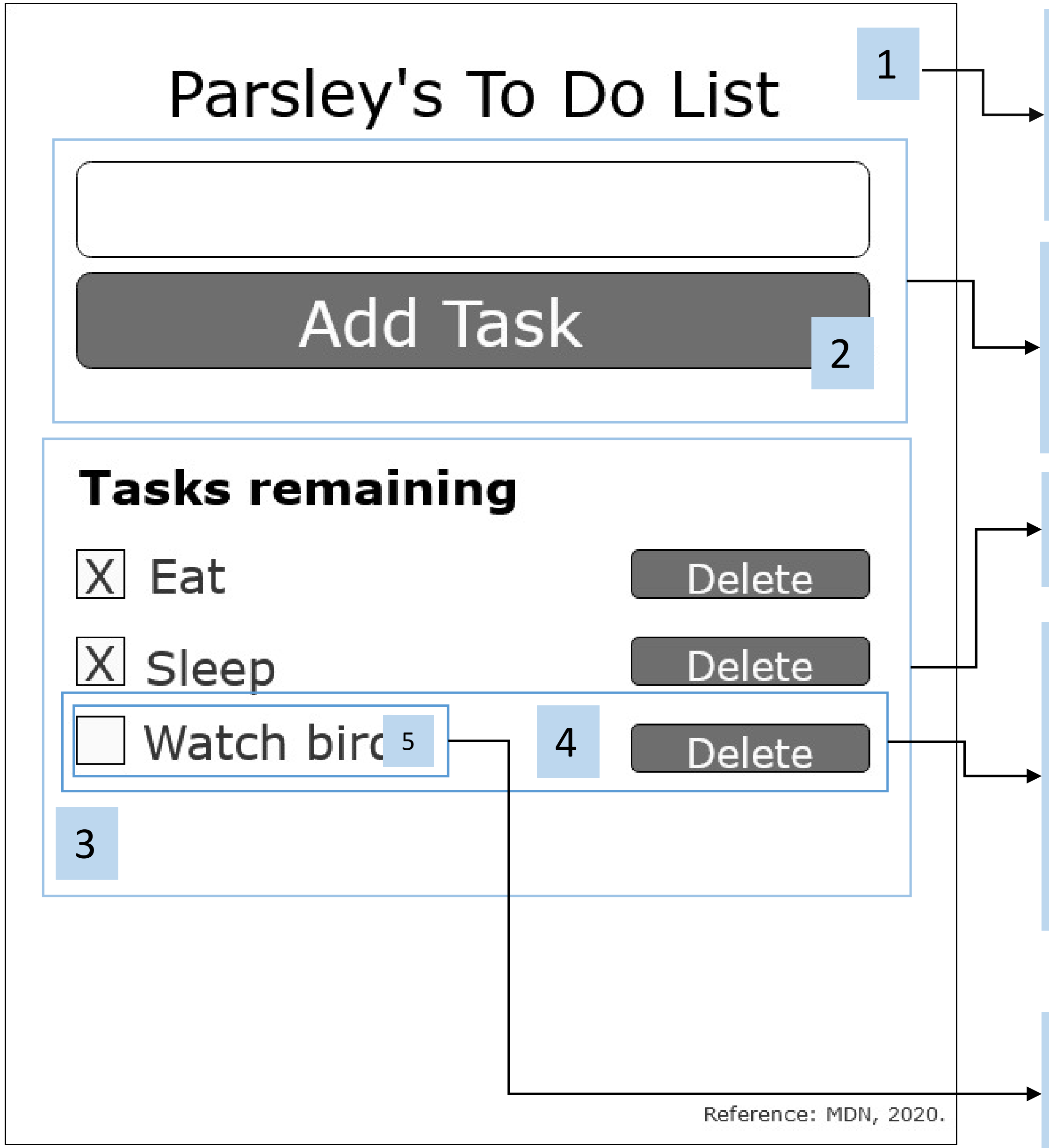
□ Watch birds

Reference: MDN, 2020.

- In React, a **component** is a reusable module that renders a part of our app (MDN, 2020b).
- A **prop** is any data passed into a React component (MDN, 2020b).
  - **Props** are written inside component calls, and use the same syntax as HTML attributes – `prop=“value”.`
- Consider Parsley’s user choice (MDN, 2020b):
  - read a lists of tasks.
  - add a tasks using the mouse/keyboard.
  - mark any task as completed using mouse/keyboard.
  - delete any task using mouse/keyboard.
- It is important to consider that *React is beginning to move away from classes, instead using functions to construct components* (Banks, 2020).
  - Therefore for this assignment, I will only focus on classes based on Ulrich teaching (Ulrich, 2020).

Initializing my To-Do app (MDN, 2020b).

1. Install `npm` packages such as `npx create-react-app moz-todo-react`.
2. Review app.js and understand `import` statements, app component in the middle, `export` statement at the bottom.
3. Pre-project house keeping. Remove `App.test.js logo.svg serviceWorker.js setupTests.js` because we are not doing any testing. Additionally, we also removed the unrelated links to styles sheet and script sheet.
4. Assume the design department has handed in their stylesheet index.css sheet in time for me to append the `index.css` to `index.js`.
5. Make component folder to store all the components we are making: **1. The Main Container of To-Do Task Component, 2. Input Form Component, 3. To-Do Component, 4. To-Do List Component,** and **5. Completed Component.**
6. All components are required to use `import React from “react”` in the global or at the top most line in the code.



Reference: MDN, 2020.

1. The Main Container of To-Do Task Component – This container component will show page for the user. This container include components: input task form, add task button, task list, delete buttons (Fesehatsion, 2019).

2. Input Form Component – This should allow user to input task. This contain class component to just receive props and return JSX to be rendered. This component will only have one prop to handle the click event for adding a new to-do (Nwamba, 2016).

3. To-Do Component – This component should include <ul> that contains a loop of to-do components made of <li> (Nwamba, 2016).

4. To-Do List Component:

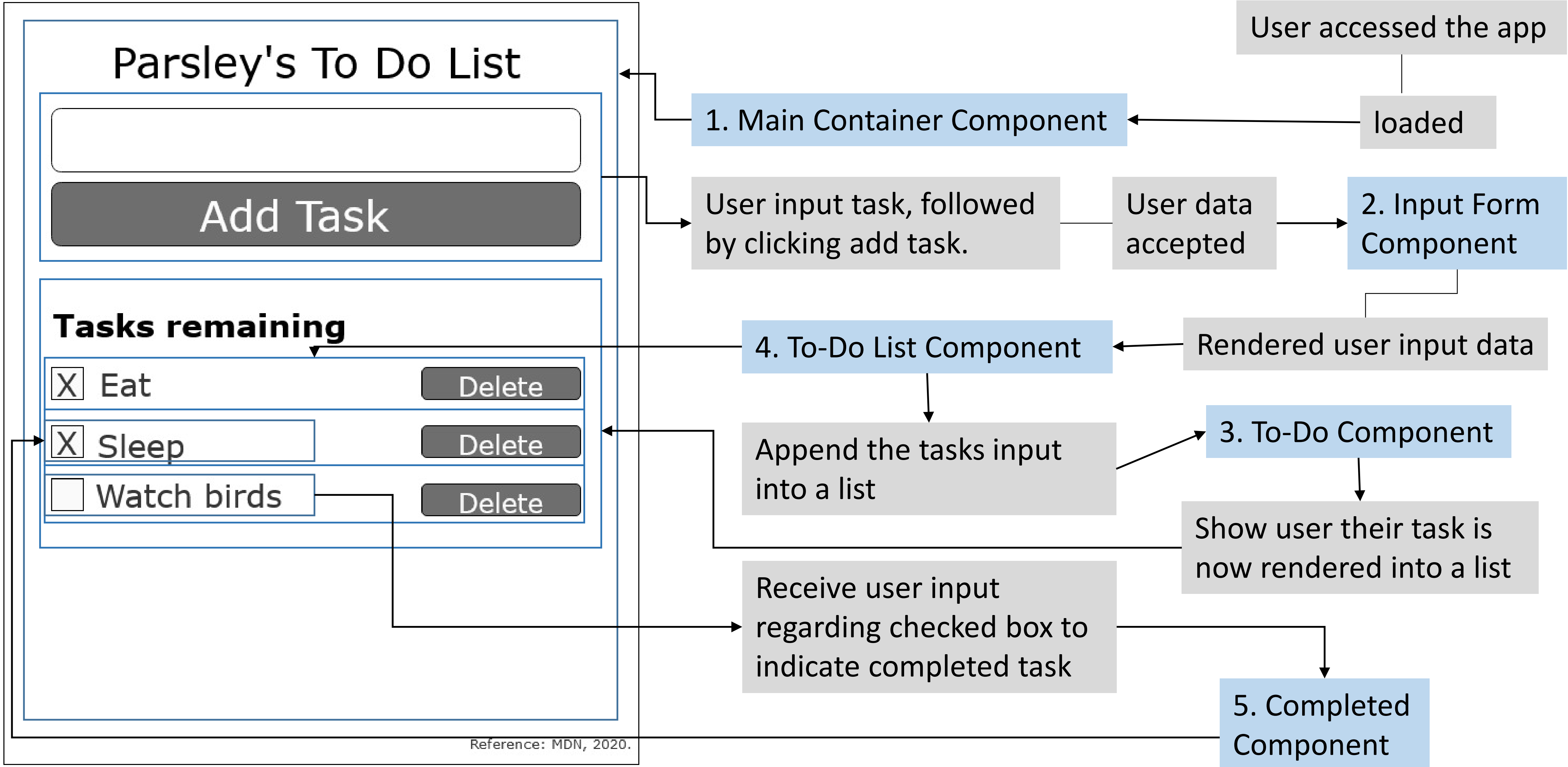
- The “Delete” property is an event handler that will be called when the item is clicked. The purpose is to delete a task when it is click. (Nwamba, 2016).
- The only the “delete” property can be passed in to it’s to the **To-Do Component** via it’s parent (Nwamba 2016).

5. Completed Component – This should be a simple component to hold true or false to determine whether it has been completed or not (MDN, 2020a).

How Each Components will Function

A visual description of a functioning To-Do application using arrows and pseudocode.

How Components Relates: A Visual Guide



Reference: MDN, 2020.

User accessed the app

loaded

1. Main Container Component

User input task, followed by clicking add task.

User data accepted

2. Input Form Component

Rendered user input data

4. To-Do List Component

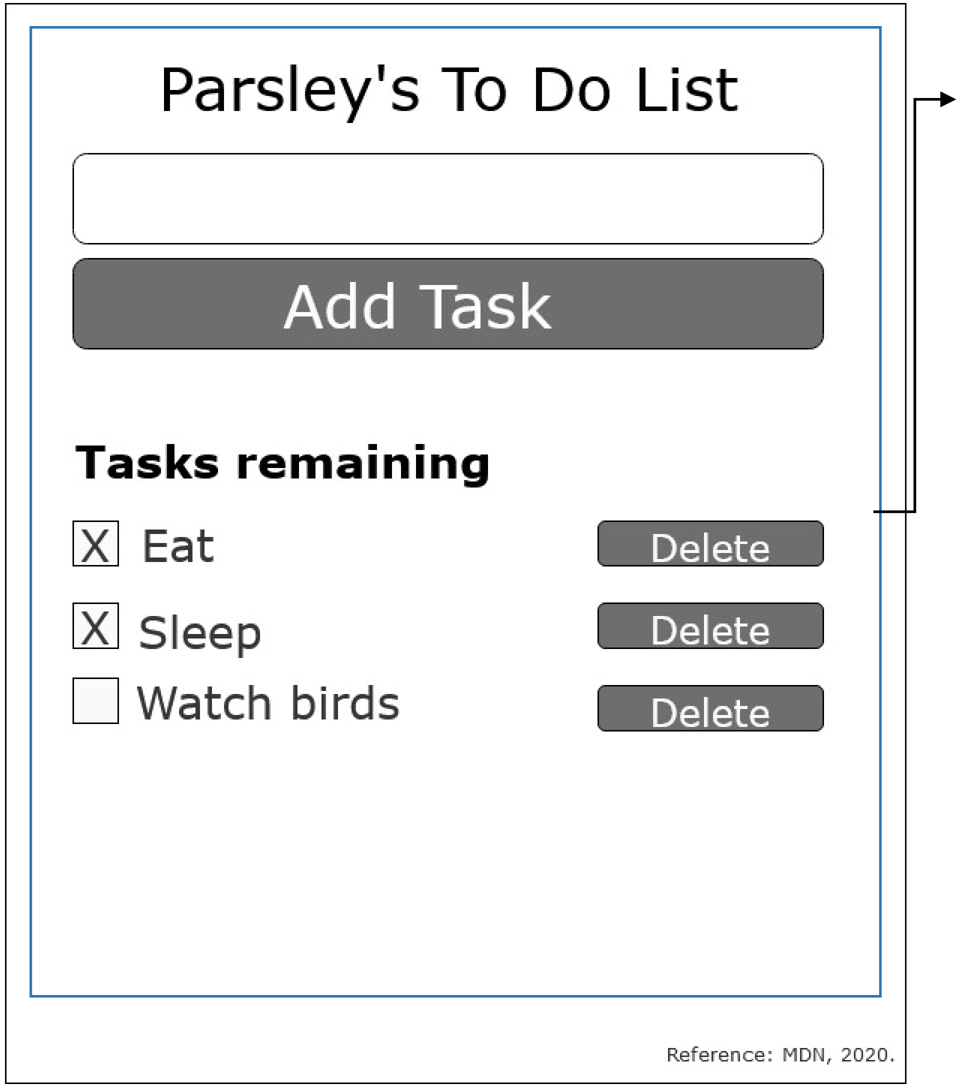
Append the tasks input into a list

3. To-Do Component

Show user their task is now rendered into a list

Receive user input regarding checked box to indicate completed task

5. Completed Component



Parsley's To Do List

Add Task

Tasks remaining

☒ Eat Delete

☒ Sleep Delete

☐ Watch birds Delete

Reference: MDN, 2020.

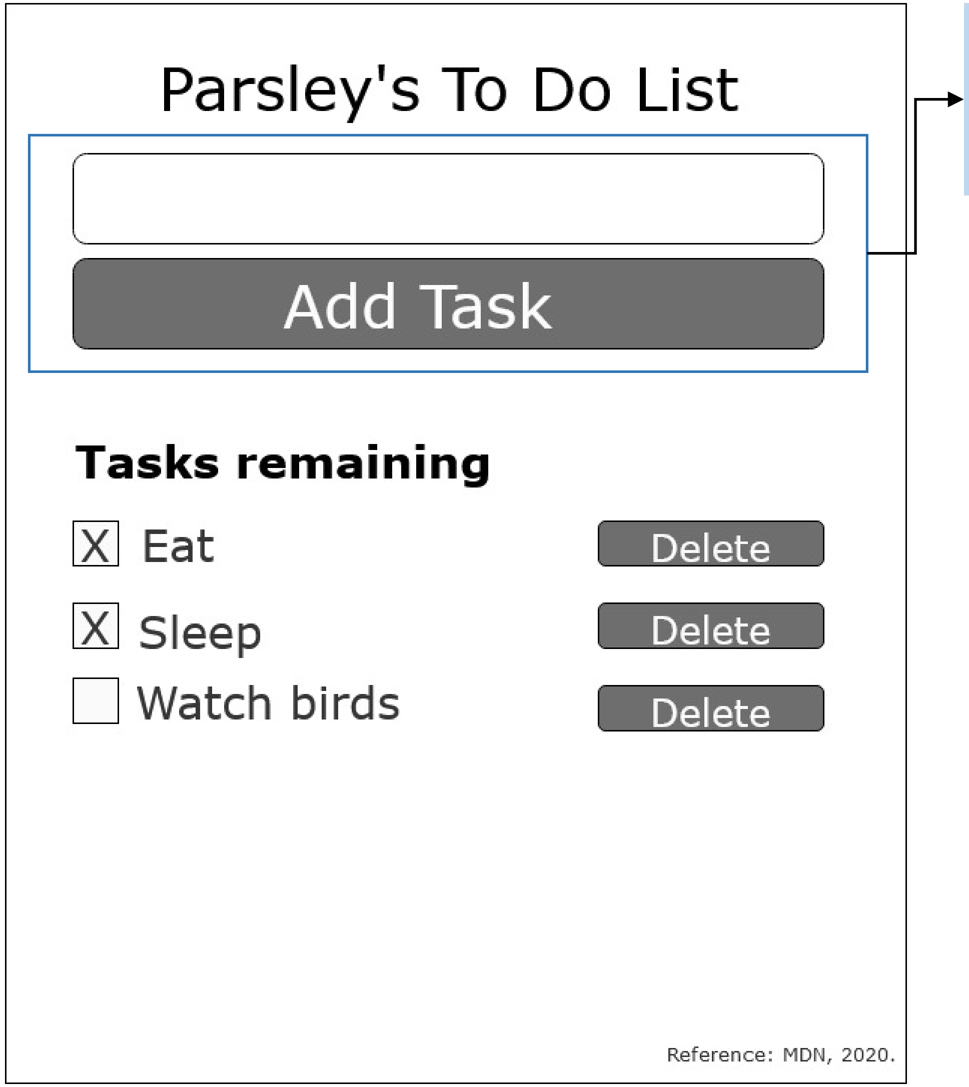
Step 1: The Main Container of To-Do Task (MDN, 2020b).

Create a skeleton of an app and paste to the App.js that creates the:

a) Title: Parsley’s To Do List
b) Input Form: create `<form>` element to make a skeleton of an input form user can type.
c) Button: create a button `type=“submit”.`
d) Create three unordered list `<ul>` and `<li>` as shown as eat, sleep, and watch birds.
e) Check box is represented by `<input type="checkbox"/>`
f) Delete button is represented with a button element to remove the specific listed task.

Now App.js component is completed and can be re-used by changing the JSX.

**Special note, there is accessibility features available for JSX and it starts with* `aria` *(MDN, 2020b)*



Parsley's To Do List

Tasks remaining

☒ Eat

☒ Sleep

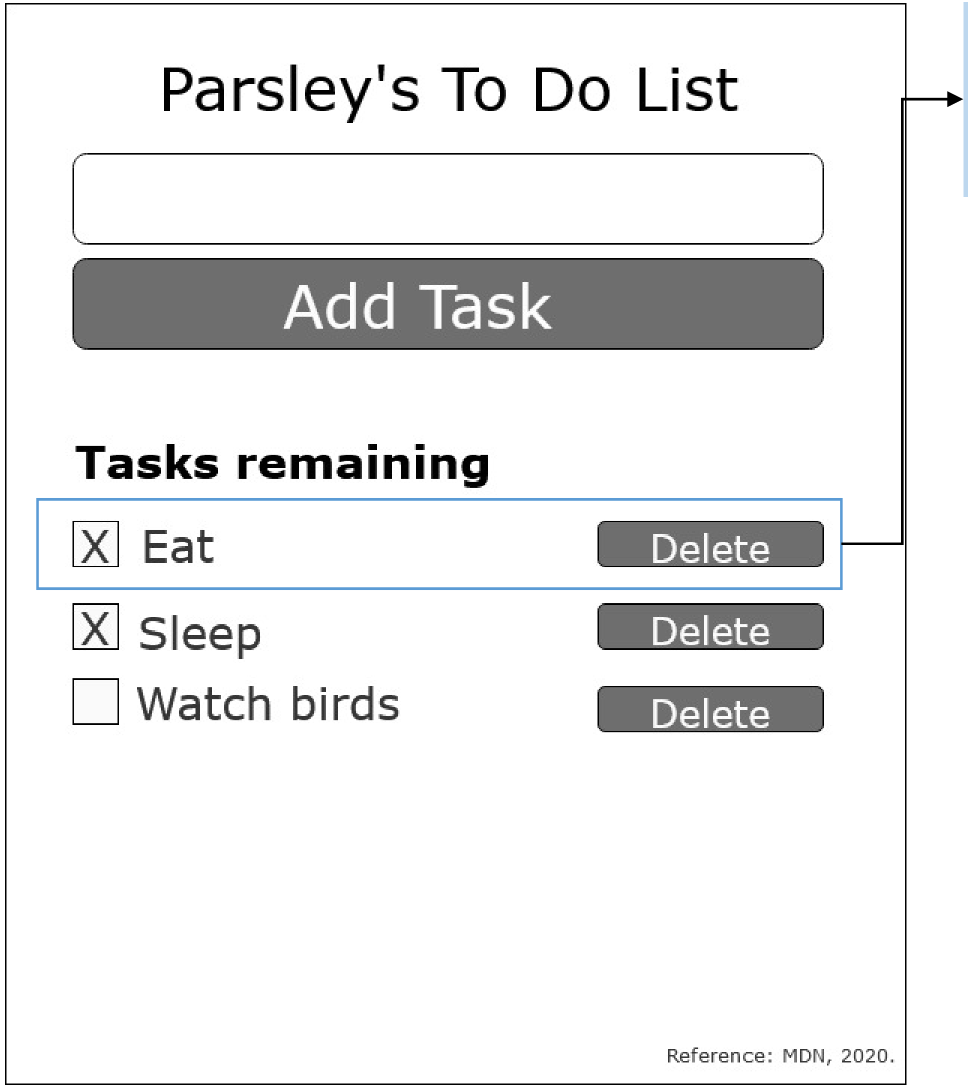
□ Watch birds

Reference: MDN, 2020.

Step 2: Input Form Component (MDN, 2020b).

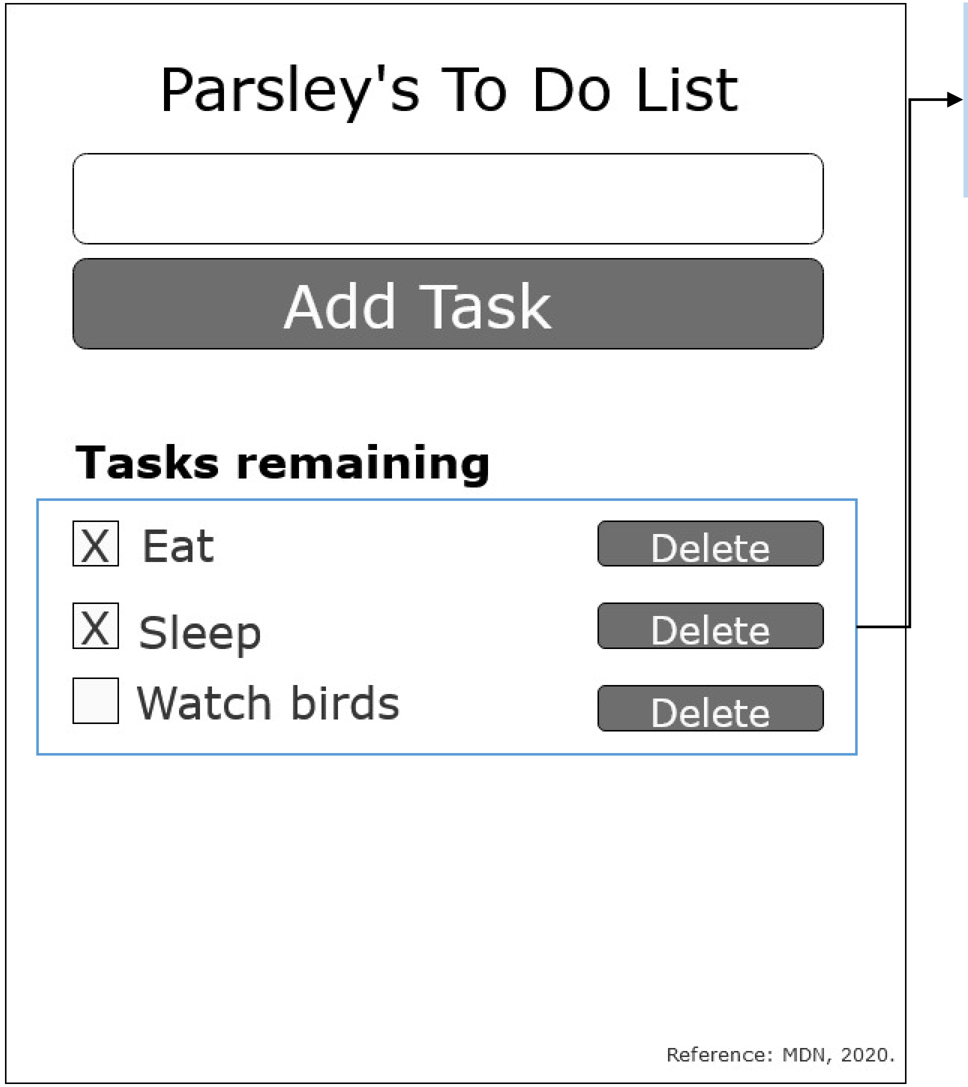
1. Input form is created to accept user input typed into the blank text area of the form.
2. “Add Task” is an event listener with a onClick that will trigger components **Step 3: To-Do Component, Step 4: To-Do List Component, and Step 5: Completed Component** sequentially.
3. Pseudocode for class in the **Step 2: Input Form Component** includes:
   a) Create `class Form extends React.Component {}.`
   b) Within the class mentioned in a), the button “Add Task” will have an event listener that prompt the input text in `<form>` to be rendered in **Step 3: To-Do Component** and **Step 4: To-Do List.**

Step 3: To-Do Component (MDN, 2020b)



Reference: MDN, 2020.

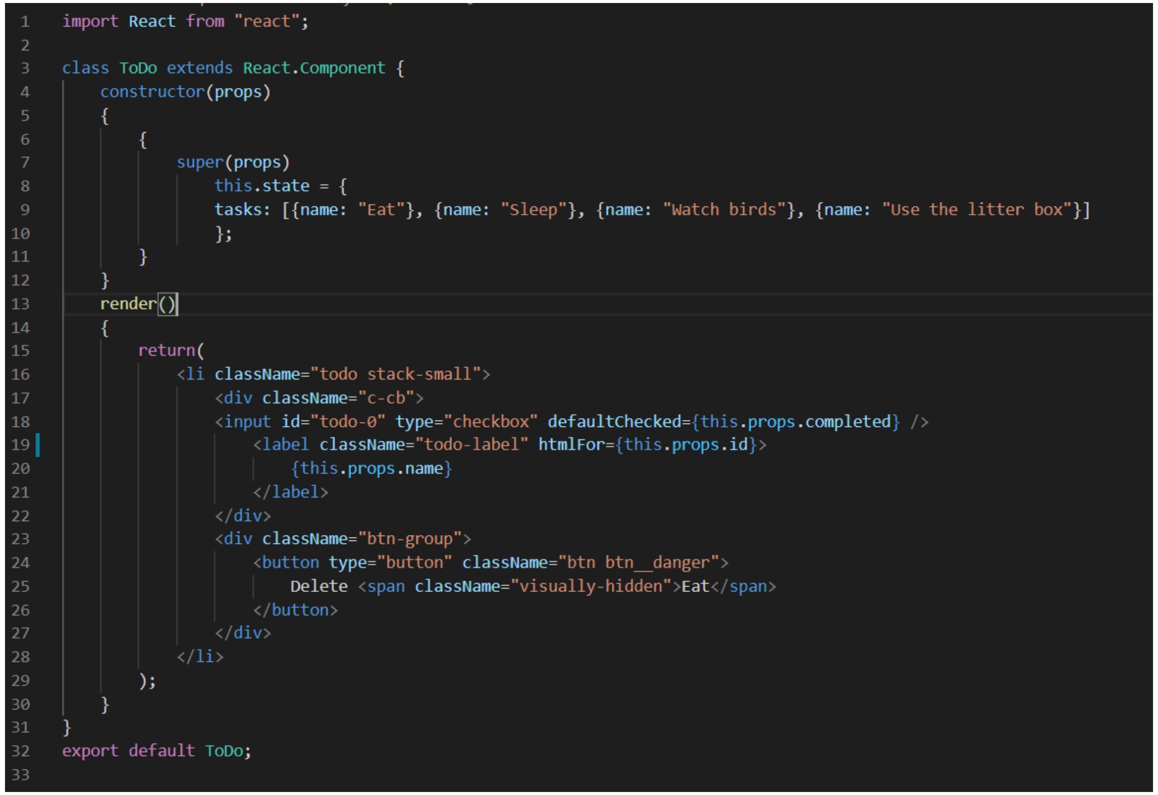
1. For this component we need to render iterations to produce a list of tasks.
2. Once a new “task” is received, we will render the data with a component in **Step 4: To-Do List Component**.
3. Pseudocode described below:
   a) Now we have to store data called “tasks” entered by user into an array.
   b) Create an array to store “tasks” **.**
   c) The “tasks” entered by user will be stored in the above array described in b).
4. Rendering an iteration of the array will produce the list that can be appended to the Apps.js.
5. “Delete” button is an event listener button, when it is click it will remove the “task” from array within ToDo.js.



Reference: MDN, 2020.

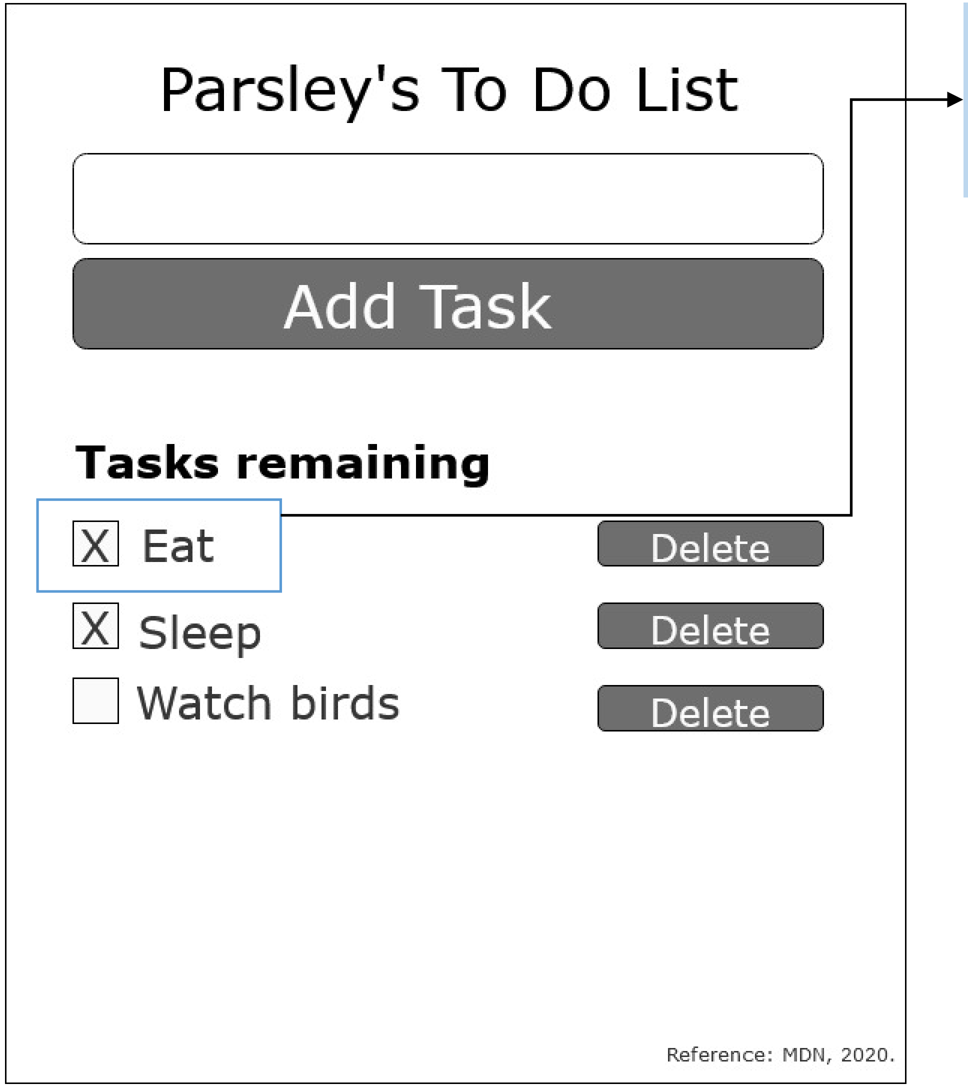
Step 4: To-Do List Component (MDN, 2020b)

1. Create a component called ToDo.js. Within ToDo.js, we will create a class to generate a To-Do list component.
2. Pseudocode as follows from `./moz-todo-react/src/components/ToDo.js` which can be found in my assignment:



```
import React from "react";

class ToDo extends React.Component {
    constructor(props)
    {
        {
            super(props)
                this.state = {
                tasks: [{name: "Eat"}, {name: "Sleep"}, {name: "Watch birds"}, {name: "Use the litter box"}]
                };
        }
    }
    render()
    {
        return(
            <li className="todo stack-small">
                <div className="c-cb">
                <input id="todo-0" type="checkbox" defaultChecked={this.props.completed} />
                    <label className="todo-label" htmlFor={this.props.id}>
                        {this.props.name}
                    </label>
                </div>
                <div className="btn-group">
                    <button type="button" className="btn btn__danger">
                        Delete <span className="visually-hidden">Eat</span>
                    </button>
                </div>
            </li>
        );
    }
}
export default ToDo;
```



Reference: MDN, 2020.

Step 5: Completed Component (MDN, 2020b)

1. The JSX check box was created in ToDo.js and currently only work for the browser.
2. This will be based on Boolean true or false statement
   a) If (checkbox == true) {task completed} else {task not completed.

Discussion/Thoughts

- I had learnt a lot, my assignment is made up of a bunch of pseudocodes for **Step 1, Step 2, Step 3 and Step 5 components.** As these components I feel requires additional teachings on `map` from the class or further reading about React.
- I have managed to do a JSX code for **Step 4** components because it only requires concepts taught by Ulrich, 2020.
- I realized while pseudocoding, both functions and classes can be used together and simultaneously, however it is the best to stick to one based on Ulrich, 2020 at this time.