# 01. Active Minutes

**Environment Specifics**

Please be aware that every JS environment may **behave differently** when executing code. Certain things that work in the browser are not supported in **Node.js**, which is the environment used by **Judge**.

The following actions are **NOT** supported:

* **.forEach()** with **NodeList** (returned by **querySelector()** and **querySelectorAll()**)
* **.forEach()** with **HTMLCollection** (returned by **getElementsByClassName()** and **element.children**)
* Using the **spread-operator** (**...**) to convert a **NodeList** into an array
* **append()** in Judge (use only **appendChild()**)
* **prepend()**
* **replaceWith()**
* **replaceAll()**
* **closest()**
* **replaceChildren()**
* Always turn the collection into a **JS array** (forEach, forOf, et.)

If you want to perform these operations, you may use **Array.from()** to first convert the collection into an array.

**Use the provided skeleton to solve this problem.**

**Write the missing functionality** of this user interface. The functionality is divided in the following steps:

**Your Task**

**Write the missing JavaScript code** to make the **Active Minutes** work as expected:

All fields **(Activity Type, Intensity, Calories Burned, Duration in Minutes** and **Date)** are **filled with the correct input**

* **Activity Type, Intensity, Calories Burned, Duration in Minutes** and **Date** are **non**-**empty** **strings**. If any of them is empty, the program should not do anything.

## 1.Getting the information from the form

• When the **["Add Activity"]** button is clicked, the information from the input fields is listed in the unordered list. A **list item** is added to the **"preview-activity"** unordered list.

* The list item should follow the same text format and order as shown in the provided picture.
* When the button is clicked, the **input** fields must be **cleared**, and the **["Add Activity"]** button should be **disabled**. Additionally, the **"Edit"** and **"Next"** buttons need to be added.

The HTML structure looks like this:



**2.Edit Activity Info**

**The functionality here is the following:**

* **When the "**Edit" **button is clicked, all of the information is loaded in to the input fields from step 1 and the list item is removed, while the** ["**Add Activity**"] **button is enabled again.**

****

* **The list item must be removed from the** "preview-activity"

****

**3.Next**

* **When the** "Next" **button is clicked,** **the information from** "preview-activity" unordered list must be transferred to "activities-table". **For you, this means removing everything inside of the <ul> with class =** "preview-activity" **and adding the info in** "activities-table" **table body**.

**For each peace of information, you must create a cell in the table, and a Delete button in cell. The** ["**Add Activity**"] **button shoud be enabled again, so that the user can add another activity.**

****

* **This is the HTML** structure **of the** "activities-table" table body **when it has one activity:**



**4.Delete**

* When you click the "**Delete**" button, you must remove the **respective** table row from the **"activities-table"** table body.****
* **This is the resulting HTML** structure when there was a single activity in ther table**:**



# Submission

Submit only yours **solve()** function.

*GOOD LUCK… ☺*