**Local Storage with Context API**

1. Setup the folder structure with tailwind CSS.

**Context API Concept**

1. Create **toDoContext.js** context file inside the **context folder**.
2. In **context file** we will import create context and use context, then we will create a To Do context which have some objects inside it.
   1. In each **1st object** we have **arrays of To-Do:**
      1. Inside this array we have some objects like: title, completed and id of every to-do.
   2. In **2nd object** we will have some **methods/function** of our to-do app:
      1. **Add** to-do in which we will pass the title of our to-do.
      2. **Update/Edit** to-do in which we will pass id and title of our to-do.
      3. **Delete** to-do in which we will pass the id only.
      4. **Toggle complete** in which we will pass the id only.
3. We will create a **index.js** file in the same folder for adding all the providers in the single file for the cleaner and less line of code.
   1. In which we will import **to do context, to do provider, use to do** form to do context
4. Now, we will create a state for changing to-do UI.
5. And we will wrap the whole code with to do provider with some values.
6. After providing values we have design the functionality of each values like add, delete, update/edit and toggle.
   1. By adding title to-dos like this, **our old to-do will be deleted and only new one will be added**.



* 1. **To avoid this we have to pass the previous to-dos also.**

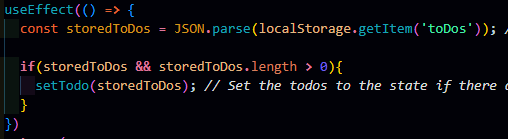


* + 1. Now, both old and new to-dos can be added together, we can move title in starting to add new to-dos at first.
  1. **To add IDs with each to-do we can use date function.**

1. For editing/updating to-do we have use loop and then we find the ID.

**Local Storage Concept**

1. In local storage we have only 4 concepts:
   1. get items
   2. set items
   3. Data stored in the key value pair
   4. Data always stored in the string format, so we have convert the data into JSON format.
2. When we load our app, all the to-dos are available so to do this we have to use a hook called **use effect** which is used to update the DOM, fetch the data, etc.
3. After this, first we have get all the to-dos, in JSON format.



1. We will again create a use effect hook, which will be used to save the data into the local storage whenever the to-Do changes the state.
2. **Conclusion, we have 2 use effect hook, 1st is used to load all the previous to-dos and 2nd is used to save all the to-dos.**
3. Now, we will create components:
   1. Form component
   2. To-Do item component
4. To add items we will use loop.