4.5.5 Save Tasks to localStorage

Well, the task data is all prepared and ready to be saved, so let's save it! Although this task is relatively small, we still have to consider one thing: how do we know when to save the data?

The answer is, any time the data changes.

We'll start by creating a function for saving tasks to `localStorage`, then we'll execute that function every time we add, update, or delete any tasks. Right above the `addEventListener()` methods at the bottom of `script.js`, create a function called `saveTasks`:

```
var saveTasks = function() {

}
```

The content of this function will be light—very light, actually. Only one method will be executed. Which one do you think it is?

Here's a hint: it's a `localStorage` method.

PAUSE

Which method saves data to localStorage—`setItem()` or `getItem()`?

`localStorage.setItem()` saves data to `localStorage`.

Hide Answer

Let's add the following code to the `saveTasks()` function:

```
localStorage.setItem("tasks", tasks);
```

That's all we need to do! Simple, right? Let's add the function call to the create, update, and delete task functions:

Add `saveTasks()` in the following functions:

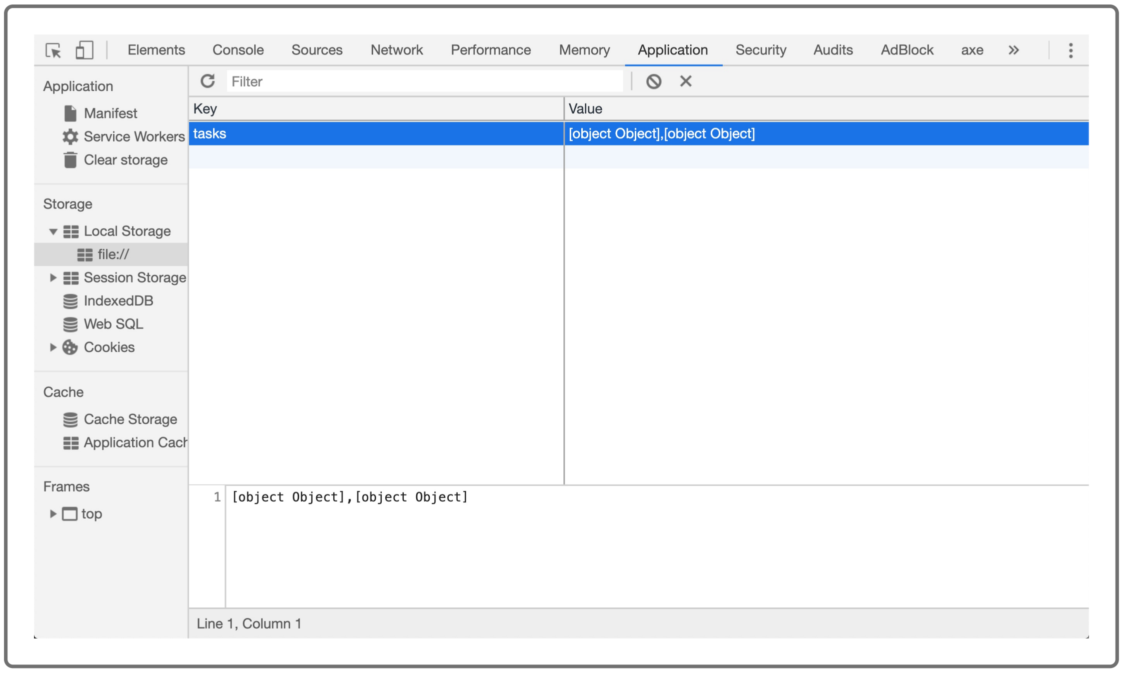
- In `createTaskEl()` anywhere after the `tasks.push(taskDataObj)` method
- In `completeEditTask()` anywhere after the `for` loop
- In `taskStatusChangeHandler()` at the end of the function
- In `deleteTask()` at the end of the function
- In `dropTaskHandler()`, right after the `for` loop

Now save script.js. Before we can test the application, we need to do some clean-up work in the local storage of Chrome.

You likely have local storage left over from the past project. Let's clear it out. Open DevTools and click on the Application tab. Then click on "Clear storage" next to the trash can icon, and a dialogue window will open to the right. Click on the "Clear site data" button in the dialogue window and you're good to go.

Now refresh the browser, create a task or two, and open DevTools again. Click on the Application tab, click on the Local Storage drop down list, and then click the `file://` option within it.

Right now, the `localStorage` for this application should look something like this image in DevTools:



Key	Value
tasks	[object Object],[object Object]

```
[object Object],[object Object]
```

We weren't expecting this, were we? We packaged up the data so neatly—we even logged it to the console to make sure it looked right. So why is `localStorage` storing it as `[object Object]`?

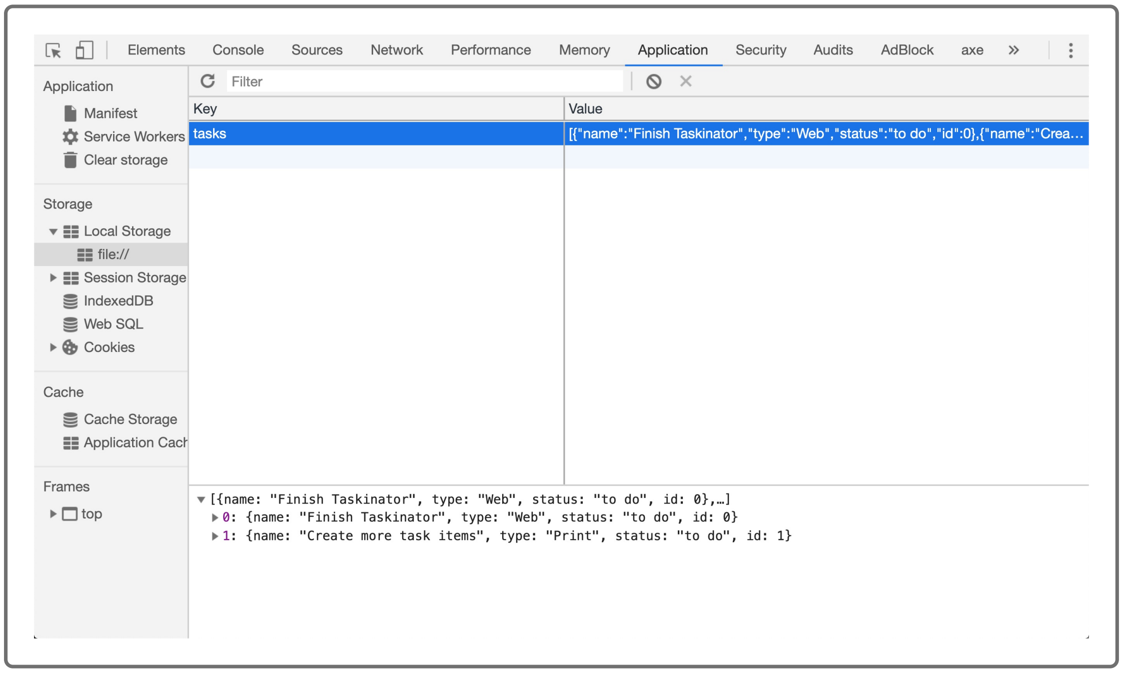
Unfortunately, `localStorage` can only store one type of data: strings. If we store a number in `localStorage`, it will turn into a string. If we store the Boolean `true`, it will end up as "true" instead.

Because objects and arrays aren't simple data values and can comprise multiple data types, a computer can't easily convert them to strings the same way it can numbers and Booleans. However, a couple of built-in JavaScript tools allow us to do this conversion ourselves.

Let's edit the `saveTasks()` function to look like this instead:

```
var saveTasks = function() {
  localStorage.setItem("tasks", JSON.stringify(tasks));
}
```

Try saving a task or two again, then check what gets stored in `localStorage`. The result should look like this image:



Key	Value
tasks	[{"name":"Finish Taskinator","type":"Web","status":"to do","id":0},{"name":"Crea…

```
▼ [{name: "Finish Taskinator", type: "Web", status: "to do", id: 0},…]
  ▶ 0: {name: "Finish Taskinator", type: "Web", status: "to do", id: 0}
  ▶ 1: {name: "Create more task items", type: "Print", status: "to do", id: 1}
```

Suddenly, `localStorage` understands what we're saving! As the method name `stringify()` implies, we just converted the `tasks` array into a string for saving in `localStorage`. The quotation marks wrapping the code tell us it's a string. But what is this whole JSON thing that `stringify()` is chained off of?

JSON stands for **JavaScript Object Notation**, a means of organizing and structuring data that's transferred from one place to another. We'll explore JSON in depth when we get further into transferring data from one place to another, but because the `stringify()` method worked for us, we can move on.

DEEP DIVE

DEEP DIVE

To learn more, see the **MDN web docs for JSON.stringify() (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON/stringify)**.

Okay, one half of this functionality is complete, as we can now save the tasks. Next, we'll discuss retrieving the tasks on page load.

Don't forget to add, commit, and push the updated code to GitHub.