

Kanban Local-Storage.md

• Reading from Local Storage

```
const ticketsFromLS = JSON.parse(localStorage.getItem("My Tickets"))
let ticketsArr = ticketsFromLS
```

String → array
if present
// [];
if nothing exists
return null

• Loading Stored Tickets into UI

```
function init() { import stored tickets into UI
    ticketArr.forEach(function(ticket) {
        createTicket(ticket.ticketTask, ticket.ticketId, ticket.ticketColor);
    });
}
```

loops through each ticket Recreates UI
init() calls init() immediately when page loads

• Saving a newly Added Ticket

```
ticketsArr.push({
    ticketTask: task,
    ticketId: id,
    ticketColor: color,
});
```

localStorage.setItem("myTickets", JSON.stringify(ticketsArr));

writes updated task Array → string
because LS stores only strings

- ↳ Kanban Q.11 → Add a new Ticket to ticketsDB in local storage
- ↳ Kanban Q.13 → Create ticket UI from ticketsDB in local storage

Kanban
Q. 12 Deleting a ticket from ticketsDB in local Storage.

Sol.

```
let ticketArr = JSON.parse(localStorage.getItem("ticketsDB"))
let ticketID = e.currentTarget.querySelector("#ticket-id").innerHTML;
function getTicketIdn(id) {
    for (let i = 0; i < ticketArr.length, i++) {
        if (ticketArr[i].ticketId === id) {
            return i;
        }
    }
    return -1; → if no ticket matches
}

let idn = getTicketIdn(ticketID);
if (idn !== -1) {
    ticketArr.splice(idn, 1); → remove item and rebuilds array
}
localStorage.setItem("tickets DB", JSON.stringify(ticketArr));
```

if id matches,
we return the index
where it found

remove item
and rebuilds
array

Kanban Summary

- Kanban Q4. Viewing Input Modal on (+).

```
let modalFlag = false
{
    let key (selection)
}

addBlk.addEventListener('click', function() {
    if (modalFlag == false) {
        modalCont.style.display = 'flex';
        modalFlag = true;
    }
});
```

- Kanban Q5. Automate Ticket Creation from input.

```
modalCont.addEventListener('keydown', function(e) {
    let key = e.key;

    if (key == "Shift") {
        createTicket(TaskArea.value, modalPriorityColor);
        modalCont.style.display = "none";
        addFlag = false;
        TaskArea.value = "";
    }
});

TaskArea.value → prewritten value
$ { value } ↳ input value

// adding tickets to DOM
function createTicket(ticketTask, ticketColor) {
    ticketId++;
    let ticketCont = document.createElement("div");
    ticketCont.setAttribute("class", "ticket-cont");
    ticketCont.innerHTML =
        <div class="ticket-color ${ticketColor}"> </div>
        <div class="ticket-id"> ${ticketId} </div>
        <div class="task-area"> ${ticketTask} </div>
    mainCont.appendChild(ticketCont);
}
```

input values

Q.6 Kanban

Filter with Colors

```
let filterButtons = document.querySelectorAll('.color');
```

```
filterButtons.forEach(function(btn) {
```

```
  btn.addEventListener('click', function() {  
    let selectedColor = btn.classList[0];
```

// 1. Remove all existing tickets from DOM

```
let allTickets = document.querySelectorAll('.ticket-object');
```

```
allTickets.forEach(function(t) {
```

```
  t.remove();
```

```
});
```

~~let filteredTickets = filter(filteredTickets, selectedColor);
return filteredTickets;~~

// 2. filter ticketsArr

```
let filteredTickets = ticketsArr.filter(function(ticketObj) {  
  return ticketObj.ticketColor === selectedColor;
```

```
});
```

// 3. Add filtered tickets back to DOM

```
filteredTickets.forEach(function(ticketObj) {
```

```
  createTicket(ticketObj.ticketText, ticketObj.ticketColor,  
               ticketObj.ticketID);
```

```
});
```

```
});
```

```
});
```

• Kanban Q.7 : Locking Mechanism

```
let ticketContArr = document.querySelectorAll('.ticket-cont');
for(let i=0; i < ticketContArr.length; i++) {
    ticketContArr[i].querySelector('button').addEventListener('click', function(e) {
        let ticket = ticketContArr[i];
        let buttonDiv = ticketContArr[i].children[3];
        let button = buttonDiv.querySelector('button');
        if(buttonDiv.classList.contains('ticket-lock')) {
            button.innerHTML = 'Unlocked';
            buttonDiv.classList.remove('ticket-lock');
            buttonDiv.classList.add('ticket-unlock');
            ticketTaskArea.setAttribute("Contenteditable", "true");
        } else {
            button.innerHTML = 'Locked';
            buttonDiv.classList.remove('ticket-unlock');
            buttonDiv.classList.add('ticket-lock');
            ticketTaskArea.setAttribute("Contenteditable", "false");
        }
    });
}
```

• Kanban Q.8 Delete Button

```
let delCont = document.querySelector('.remove-bth');
let AllTickets = document.querySelectorAll('.ticket-cont');
let deleteMode = false; // toggle flag
delCont.addEventListener('click', function() {
    deleteMode = !deleteMode; // toggle
    if(deleteMode) {
        alert("activated");
        delCont.style.backgroundColor = "red";
        AllTickets.forEach(function(ticket) {
            ticket.addEventListener('click', deleteTicket);
        });
    }
});
```

```

else {
    alert("de-activated");
    delCont.style.backgroundColor = "inherit";
}

All Tickets. forEach(ticket) {
    ticket.removeEventListener('click', deleteTicket);
}
}
}

```

function to deleteTicket

```

function deleteTicket() {
    if (deleteMode) {
        this.remove();
    }
}

```

remove event target task

// Kanban Q.9 : Clicking and Changing Ticket Colors

```

let ticket = document.querySelector('.ticket-cont');
let colors = ["lightpink", "lightgreen", "lightblue", "black"];
handleColor(ticket);

function handleColor(ticket) {
    let ticketColorBand = ticket.querySelector('.ticket-color');
    ticketColorBand.addEventListener("click", function() {
        let currentColor = ticketColorBand.classList[1];
        let currentIndex = colors.indexOf(currentColor);
        let nextColorIndex = (currentIndex + 1) % colors.length;
        let nextColor = colors[nextColorIndex];
        ticketColorBand.classList.remove(currentColor);
        ticketColorBand.classList.add(nextColor);
    });
}

```