Big Project Part 2

Some background info...

JavaScript

- Programming language made for developing websites
- Most know for building frontend web applications
- Used in combination with HTML and CSS

JavaScript Functions

```
function function_name
(parameter_name) {
   body_of_the_function
}
```

{.js}

JavaScript

What are we going to do today?

Walk through of JS code

Updating the DOM

Local Storage

Drag & Drop API

We hope you are having fun! ;)

Let's get started!

If you were not here last week …

Fork the following repository: https://github.com/infpals/ip2022-big-project-template-updated

This repo contains implemented parts from last session

Walk through of the JS code

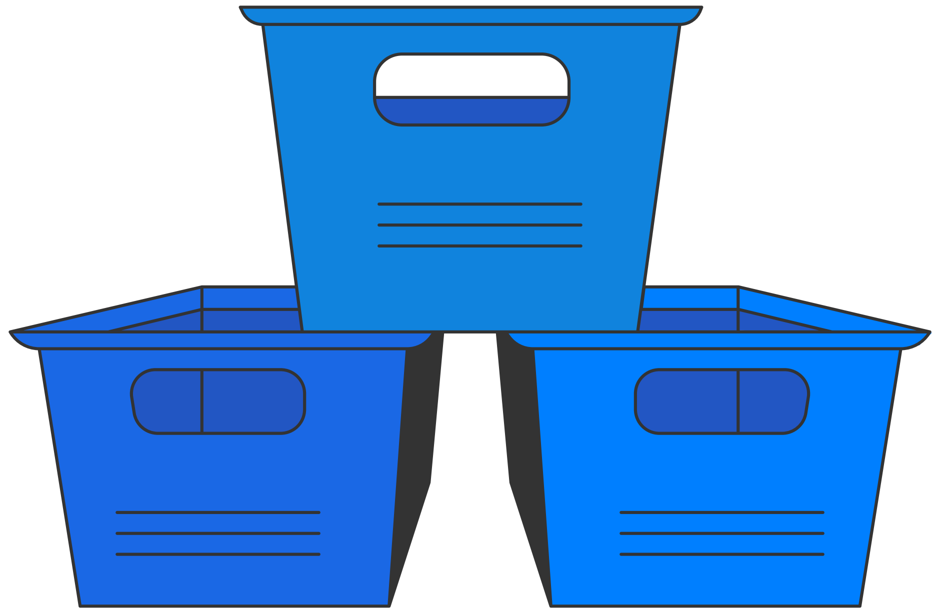
Open your script.js file

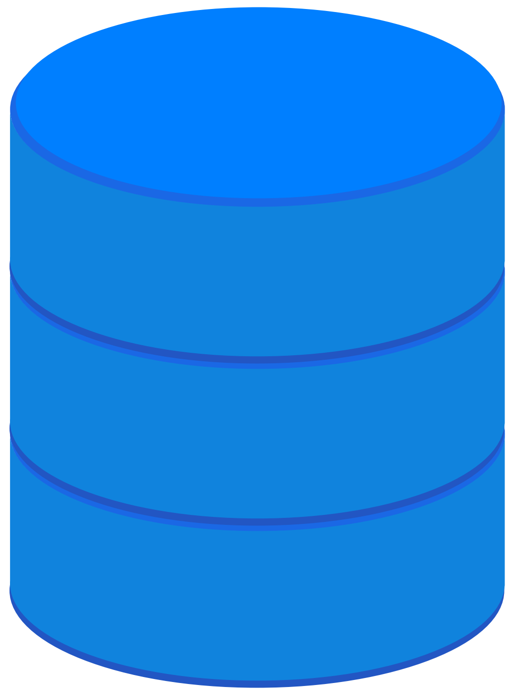
Local Storage

Local Storage

- Storage which stores information on your local machine
- Purely used by js no php/databases needed
- Uses key - values pairs
- localStorage - Storage object which Stores Data from the browser

Where can you see localStorage object being used and do you remember why?

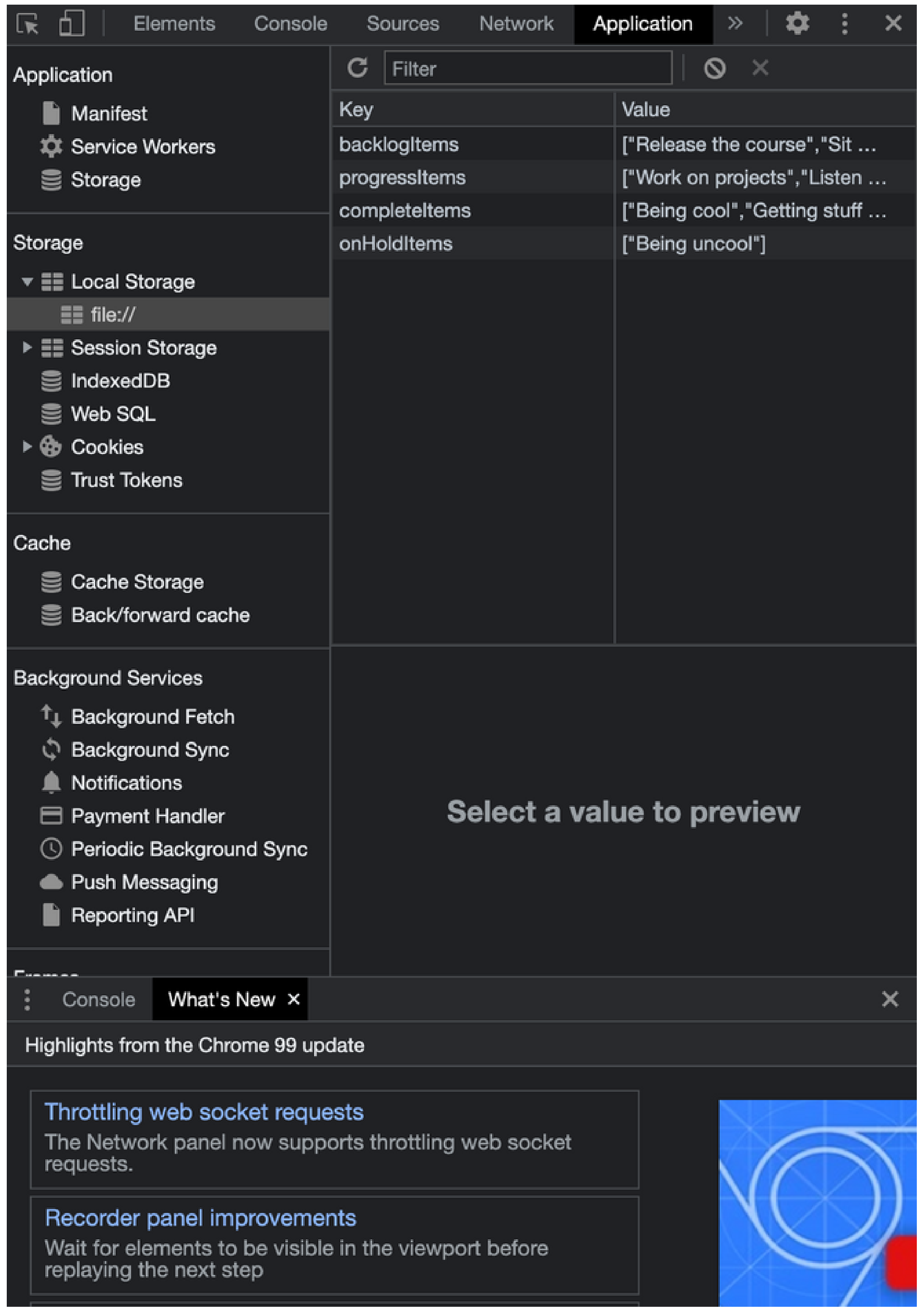




Local Storage

What can you now see in your local Storage? What is stored in key and values? What part of certain function now sets these values?

1. Right click anywhere on your Kanban Boards
2. Choose Inspect
3. Choose Application



Key	Value
backlogItems	["Release the course","Sit ...
progressItems	["Work on projects","Listen ...
completeItems	["Being cool","Getting stuff ...
onHoldItems	["Being uncool"]

Highlights from the Chrome 99 update

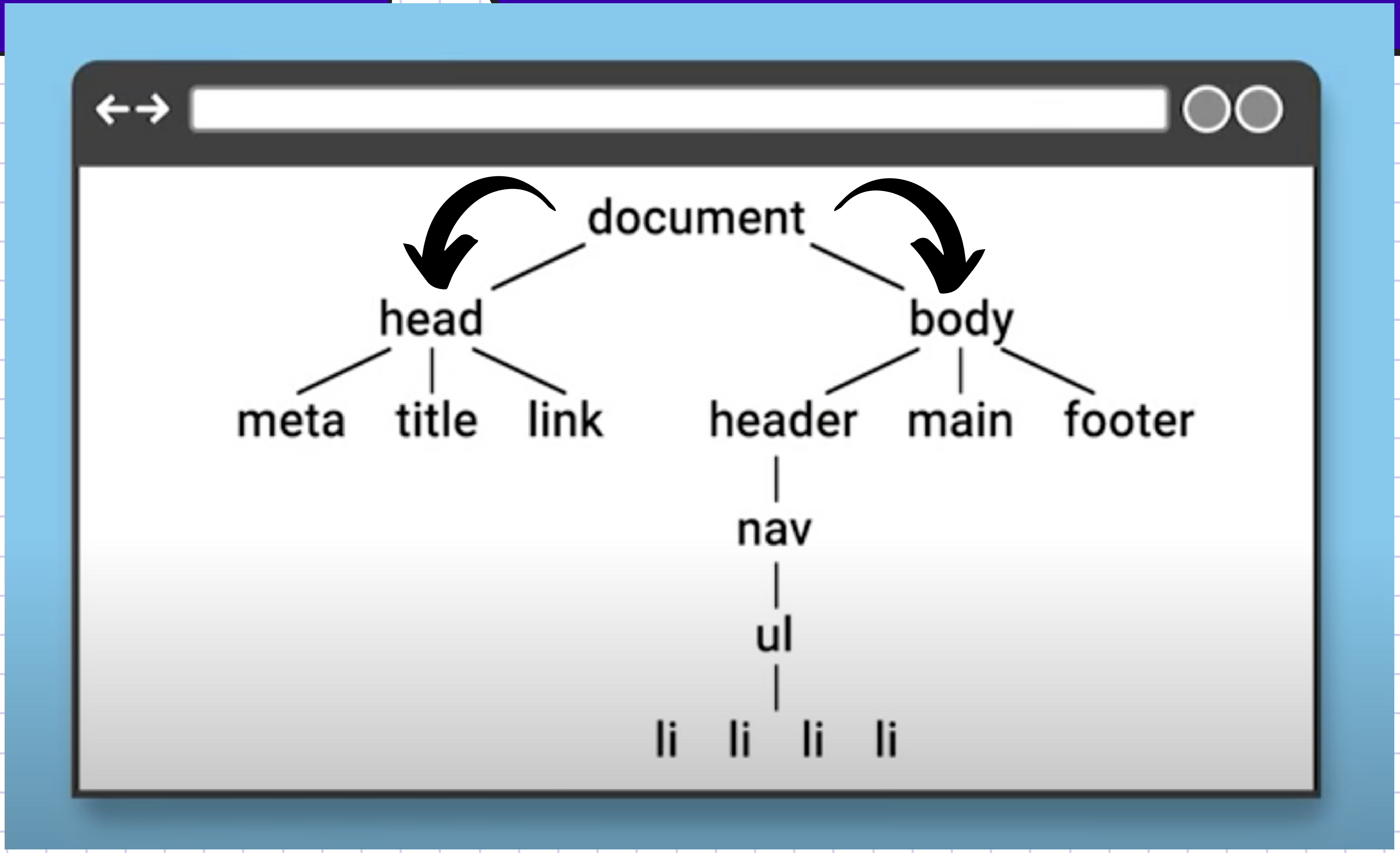
Throttling web socket requests
The Network panel now supports throttling web socket requests.

Recorder panel improvements
Wait for elements to be visible in the viewport before replaying the next step

Updating our DOM

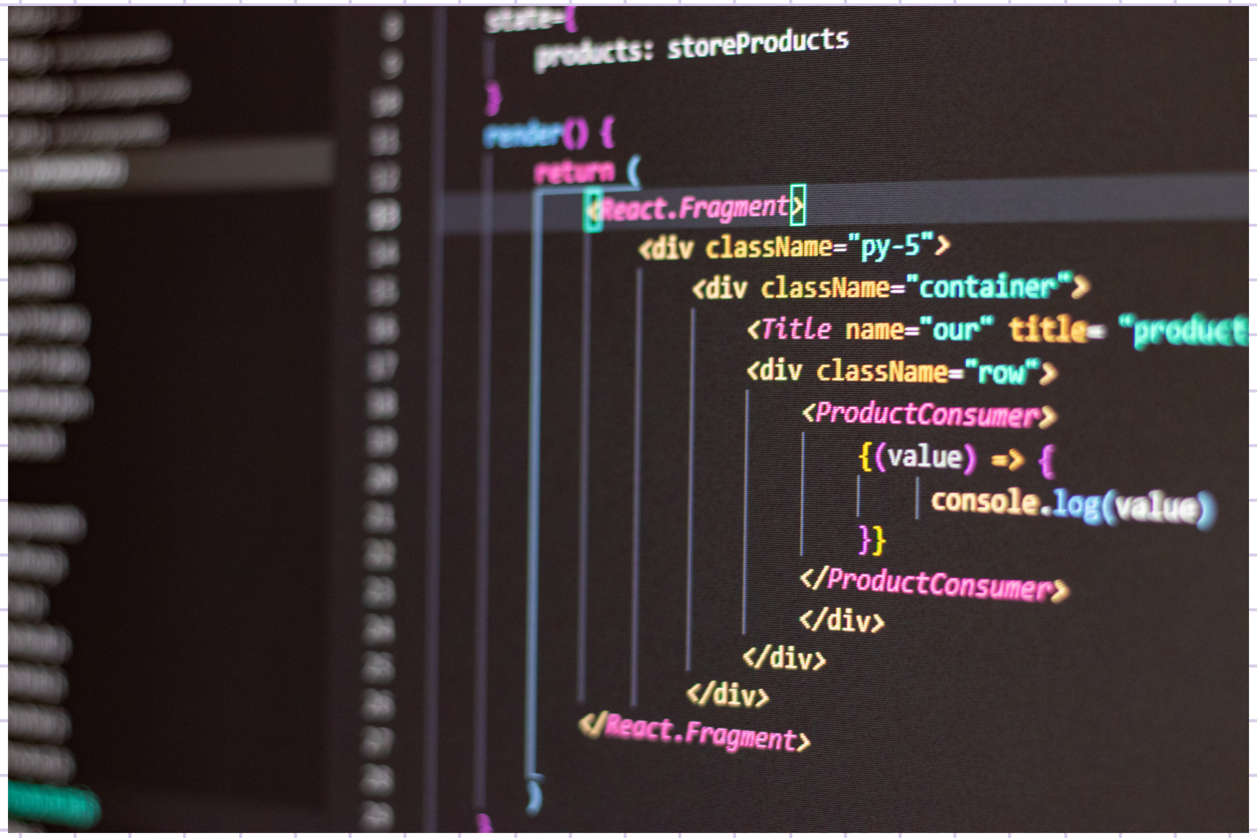
DOM

- Document Object Model
- Javascript object which is used in order to access content on the website
- Constructed of Nodes



*document is parent Node of head and body
*Creates a tree structure

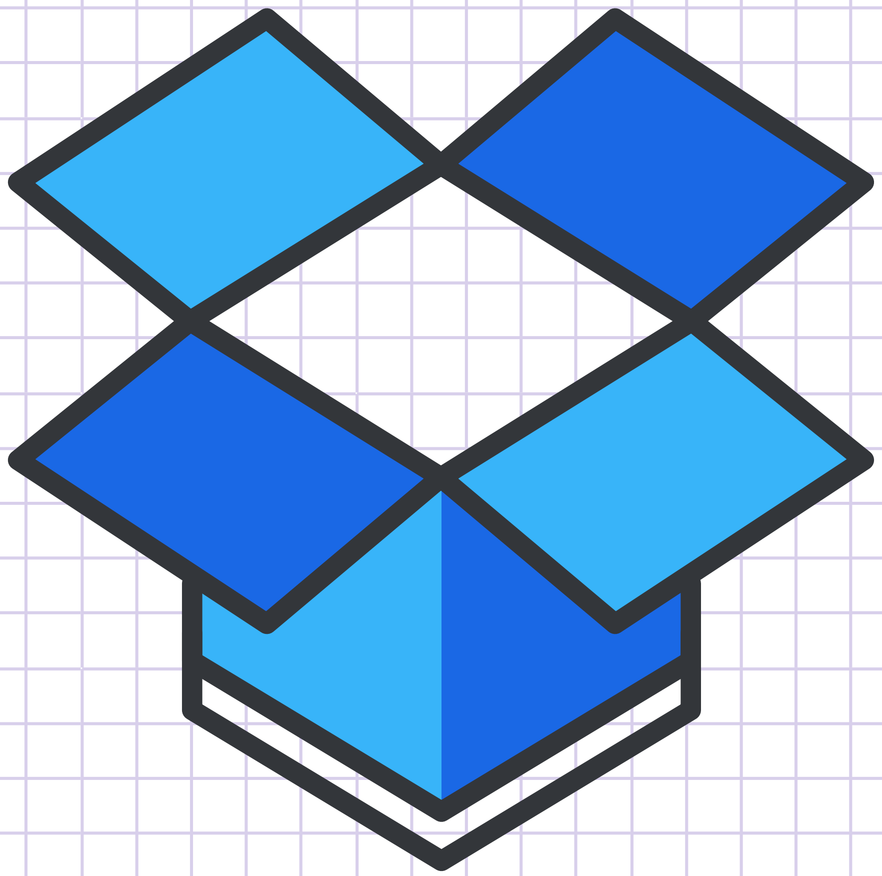
Update DOM



```
state={
  products: storeProducts
}
render() {
  return (
    <React.Fragment>
      <div className="py-5">
        <div className="container">
          <Title name="our" title= "product
          <div className="row">
            <ProductConsumer>
              {(value) => {
                console.log(value)
              }}
            </ProductConsumer>
          </div>
        </div>
      </div>
    </React.Fragment>
  }
```

- We want to check local storage, but only once
    - Create a global variable called updatedOnLoad with let under the comment //Items and set it to false
    - Inside our updateDOM function, we want to check if updateOnLoad is false, then, we want to call our saved columns using getSavedColumns();

Update DOM



- We first want to reset the textContent in our backlogList with setting the content to an empty string
  - backlogList.textContent = "";
- We then want to iterate over our backlogListArray and create new items!
  - In JS, we use forEach((backlogItem, index) => { }) on the variable backlogListArray
  - Inside our curly brackets, we want to call a function we has already been implemented called createItemEl(backlogList, 0, backlogItem, index);
- Question: Do you know why we use 0? Is it going to be the same for the other columns?

We hope you are having fun! ;)

Now try and do the same for the other three columns: Progress, Complete and On Hold!

Naming conventions: progress, complete, onHold

- **Remember that you have to change the "0" to match the number of each column!**
- **When you're done, call the function updateDOM();**
- **Inspect > Console what do you see?**

Update DOM

Adding Text to Items

- We want to update our createItemEl function!
- We are already creating an element 'li' which can be seen in HTML. Now we need to:
  - Call function textContent on var listEl and set it to equal item (don't forget ;)
  - Append our item listEl to our columnEl input using: .appendChild(listEl);
- Do you remember our placeholders in HTML for "Testing"? Let's go and remove those!

We hope you are having fun! ;)

Open your index.html in browser. What changes do you see? Where does the text in each col come from?

Drag & Drop

Drag & Drop

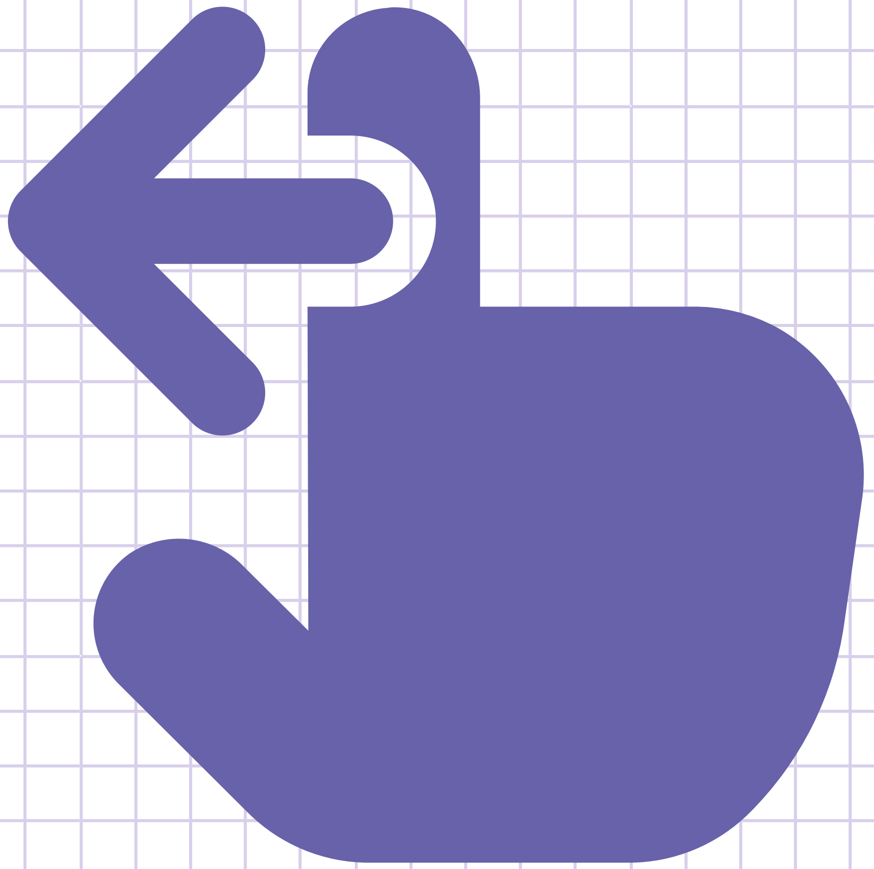
1. Make an element dragabble
    - In our function createItemEl, we want to set our listEl.draggable = true;

Drag & Drop

2. We want to set an event to know that we. have started dragging

- In our function createItemEl we want to call setAttribute('ondragstart', 'drag(event)'); on variable listEl

Drag & Drop



3. Create a function called drag with parameter e which stands for event
    - We need to create our drag function at the bottom of our JS file
    - For that, let's first create two global variables called draggedItem and currentColumn with let
    - Then, create a function drag(e)
        - Inside set draggedItem = e.target;

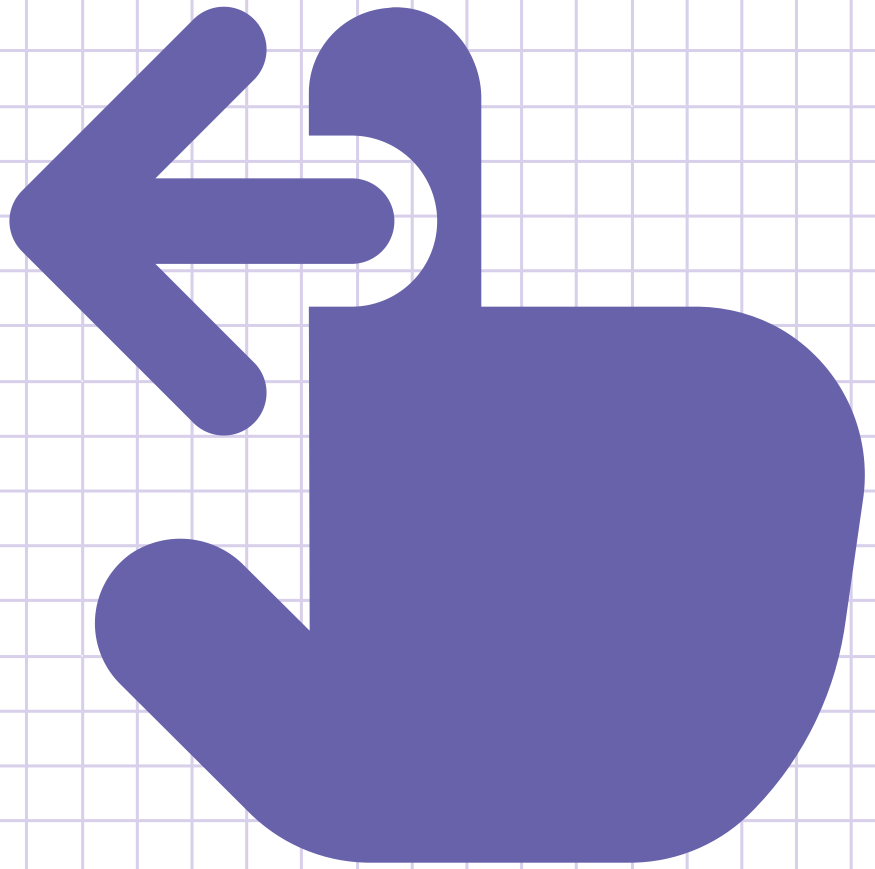
Drag & Drop

4. Now we want functions which allows items to be dropped into columns so we can create a function allowDrop with parameter e
   - On parameter e call function preventDefault() which allows items to be dropped

Drag & Drop

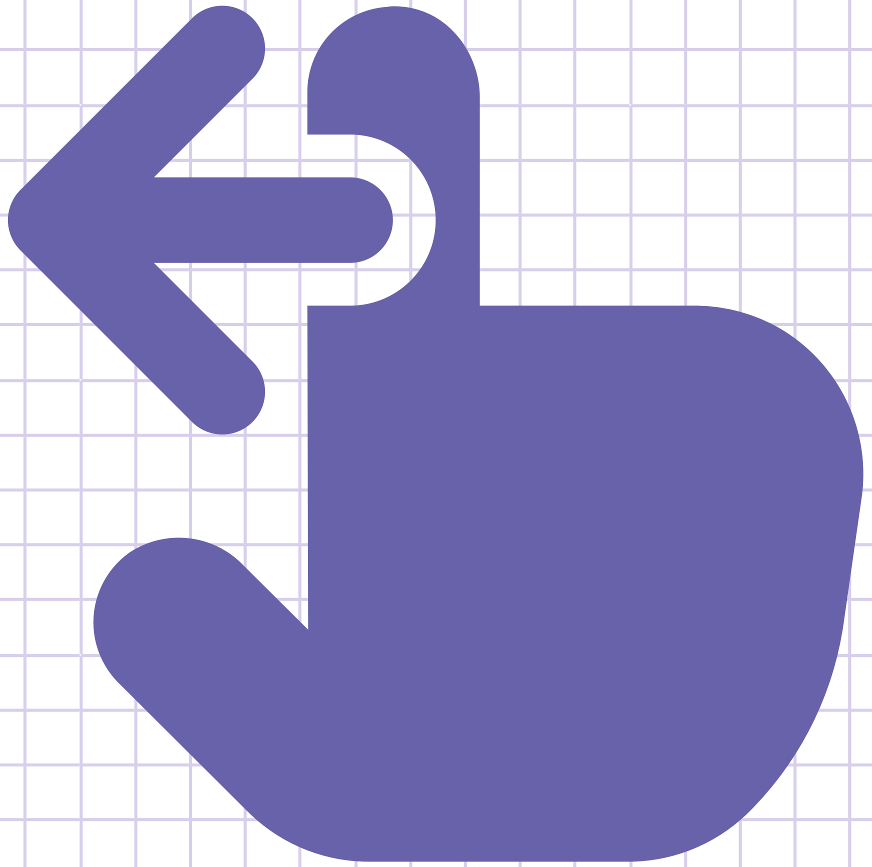
5. We want another function to actually drop our items into the columns
   - Let's create a function drop with parameter e (event)
     - call preventDefault() on e inside this function too!

Drag & Drop



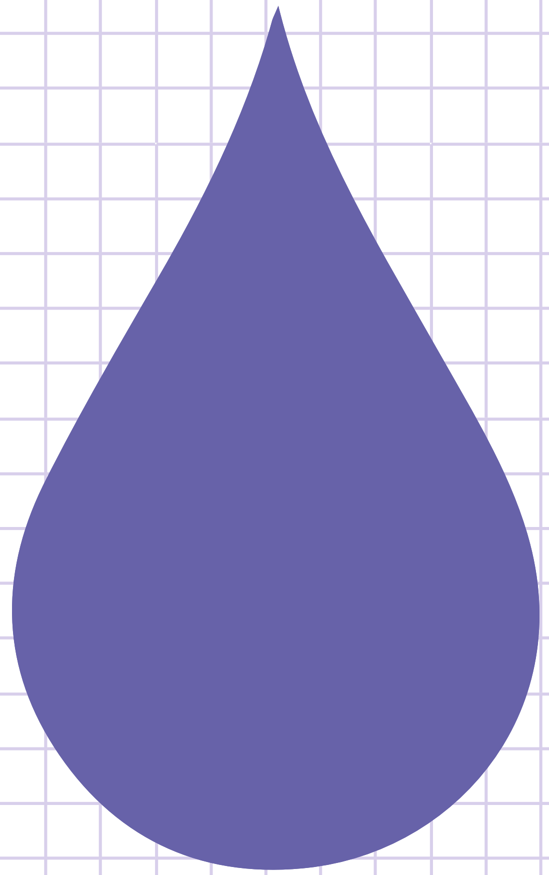
6. Create a function to change the color of the column when we enter a new item into it! (this function detects when new items enters the column area)
   - This function should be called dragEnter and we should pass it the parameter column
   - We will make use of .over in our CSS file!
   - Inside our function:
     - Let's Connect the css and js with itemLists[column].classList.add('over')
     - (itemLists is defined at the top of our JS file!)
     - Assign the global variable currentColumn the parameter column you are passing into the function

Drag & Drop



7. We now need to add our new event listeners to our HTML
   - Find the backlog column content division in our HTML
   - Find the unordered list with class="drag-item-list"
   - Here, add:
     - ondrop = "drop (event)"
     - ondragover = "allowDrop(event)"
     - ondragenter = "dragEnter(0)"
   - Copy these into each of our unordered lists! (Remember to switch the 0 depending on which column you're on!)
   - Open index.html in your browser and try to drag an item into different columns what happens?

Drop function



8. Now we need to make sure we can drop our items into column and they get updated with our item
   - First we want to remove the background color if an items in no longer in the column area
     - In the drop function itemLists.forEach((column) => {column.classList.remove('over');});
   - Now we want to add dropped items into the column:
     - const parent = itemLists[currentColumn];
     - parent.appendChild(draggedItem);

We hope you are having fun! ;)

Check your Kanban Board in the browser

Try and reload the page! What happens?

Apply to be InfPals Leader in 2022/2023