Big Project Part 2

Some background info...

JavaScript

•

What are we going to do today?

Refactoring

Updating the DOM

Drag & Drop API

Fixing up some bugs!

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/ip-2022-big-project-template

Updating our DOM

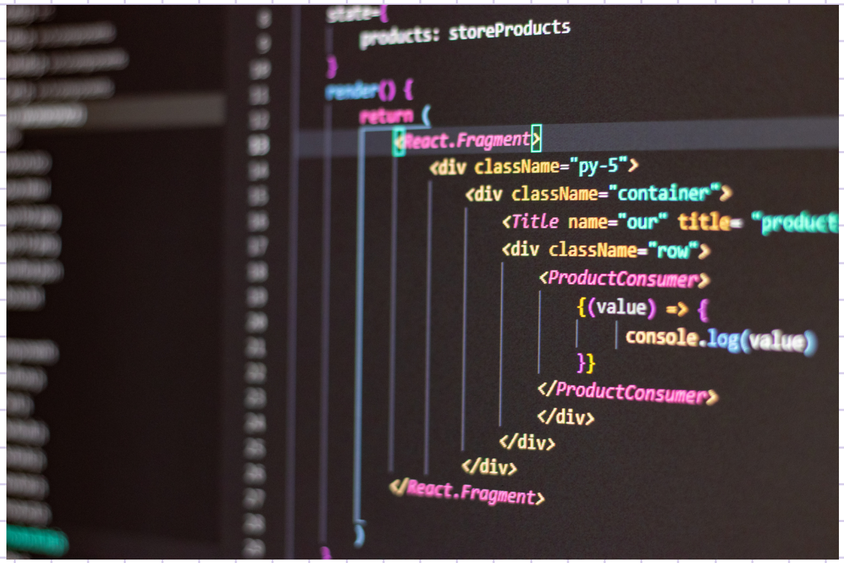
Copy our refactored function

- Create an empty array below the comment: Initialize arrays called listArrays

Let's make our function for ipdateSavedColumns() less repetitive:

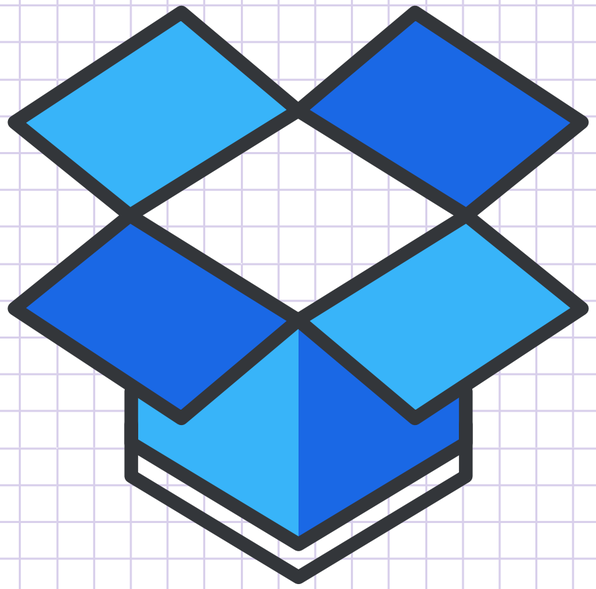
```
// Set localStorage Arrays
function updateSavedColumns() {
  listArrays = [backlogListArray, progressListArray, completeListArray, onHoldListArray];
  const arrayNames = ['backlog', 'progress', 'complete', 'onHold'];
  arrayNames.forEach((arrayName, index) => {
    localStorage.setItem(`${arrayName}Items`, JSON.stringify(listArrays[index]));
  });
}
```

Update DOM



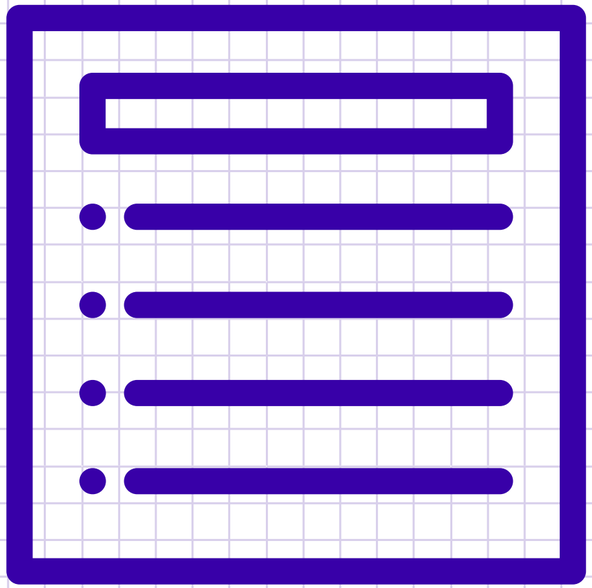
- We want to check local storage, but only once.
  - Create a variable called updatedOnLoad.
  - Inside our updateDOM function, we want to check if it is false. Then, we want to call our saved columns using getSavedColumns();

Update DOM



- We first want to reset the textContent in our backlogList,
  - backlogList.textContent = '';
- We then want to iterate over our backlogListArray and create new items!
  - In JS, we use forEach((backlogItem, index) => { ... });
  - Inside our brackets, we want to call a function we have already 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?

Creating Items



- We want to update our createItemEl function!
- We are already creating an element 'li'. Now we need to:
  - Set textContent = item
  - 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!
- Don't forget to set updatedOnLoad to true at the end of our updateDOM() function!

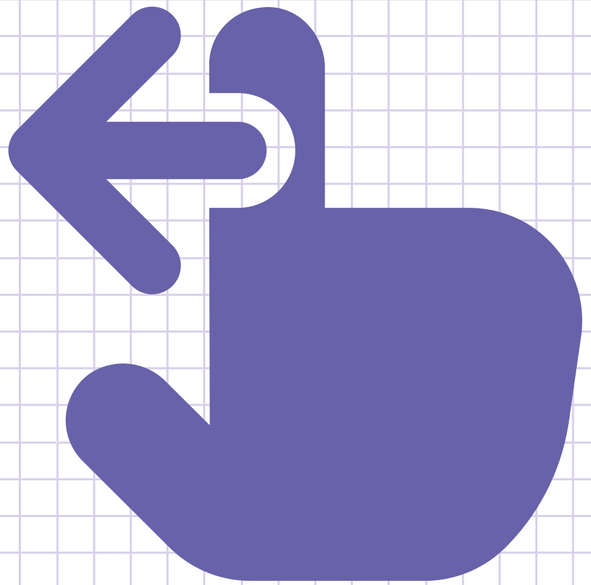
Now try and do the same for the other three columns: Progress, Complete and On Hold!

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();

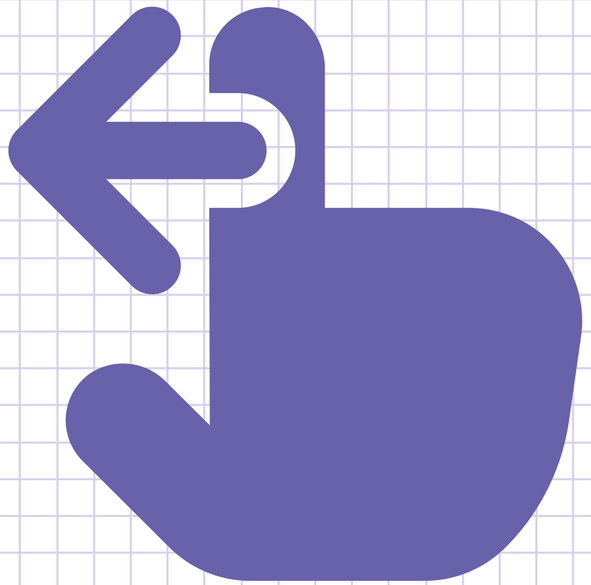
Drag & Drop

Drag & Drop



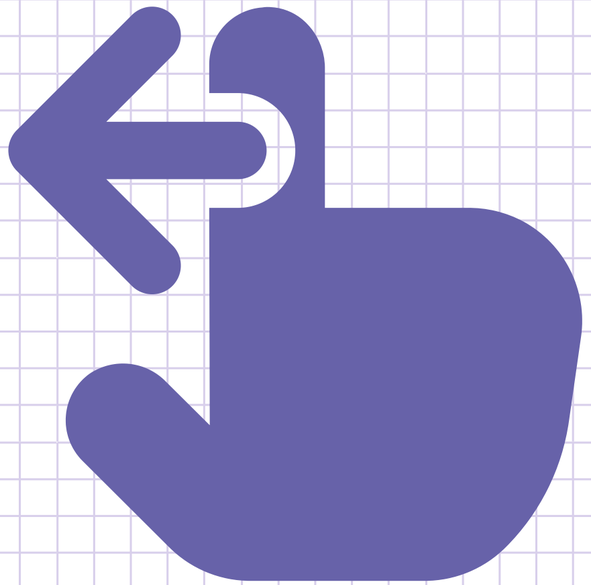
1. Make an element dragabble
    a. In our function createItemEl, we want to set our listEl.draggable = true;
2. We want to be able t
    a. We want to set an event to know that we have started dragging
        i. We can use setAttribute('ondragstart', 'drag(event)');

Drag & Drop



3. Create a drag function
   - 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;
   - Then, create a function drag(e)
     - Set draggedItem = e.target;
4. Now, we don't want to drop one item on top of another. So we can create a function allowDrop(e)
   - We want to add preventDefault() to our event so that it can be dropped!
5. We want another function to actually drop our item into a column.
   - Let's call this drop(e)
   - Set e.preventDefault() inside this function too!

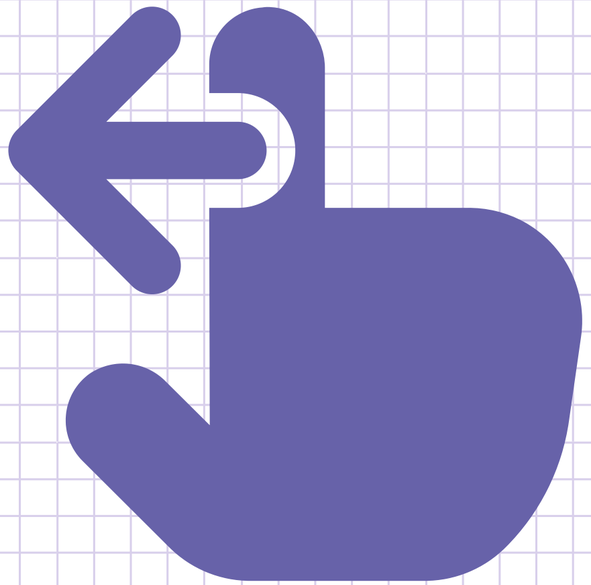
Drag & Drop



6. Create a function to change the color of the column when we enter a new item into it!

- This function should be called dragEnter(column)
- We will make use of .over in our CSS file!
- Inside our function:
  - get the column using itemLists[column] (this is defined at the top of our JS file!)
  - Add .classList.add('over')
  - Let's also set our currentColumn = column;

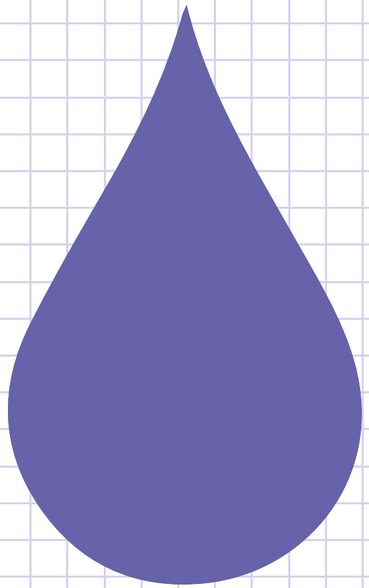
Drag & Drop



7. We now need to add our new event listeners to our HTML.

- Find the backlog content division in our HTML.
- Find the unordered list with id = "backlog-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!)

Drop function



8. Now we need to make sure we can drop our items!

- First we want to remove the background color:
  - itemLists.forEach((column) => {column.classList.remove('over');});
- Now we want to add item to column:
  - const parent = itemLists[currentColumn];
  - parent.appendChild(draggedItem);

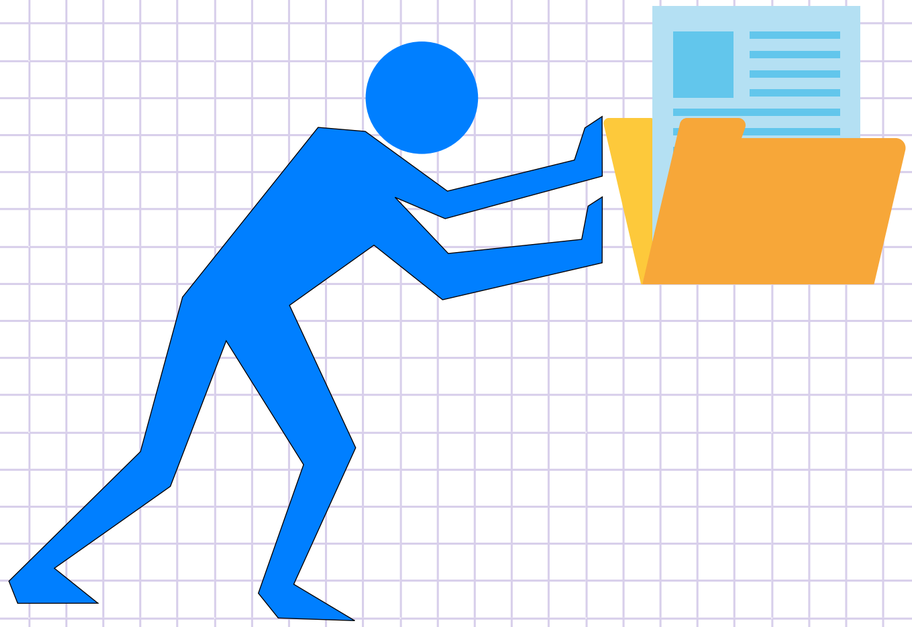
We hope you are having fun! ;)

Check it out using LiveServer!

Try and reload the page! What happens?

Rebuilding Arrays

Pushing items



1.We need a new function that allows our arrays to reflect drag and drop items!

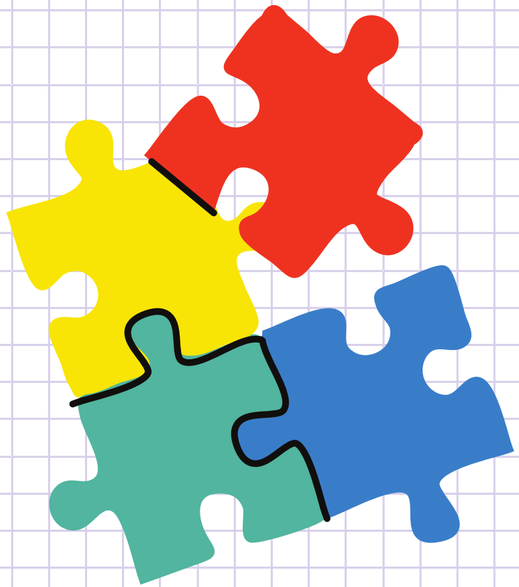
- Create a new function below our updatedDOM() called rebuildArrays()
- We need to call this function from within our drop function (do it!)!

2. Let's iterate over the children in our list:

- First, set backlogListArray = [];
- Then, set up a normal for loop (like in Java!) that loops while i < backlogList.children.length
- Inside thus loop, we want to add backlogListArray.push(backlogList.children[index].textContent);

3. Do the same for our other lists! (be mindful of the names!)

Put it together!



4. Lastly, at the end of our UpdateDOM function, let's call our updateSavedColumns() function!
5. Check it out!
   - Try and move items from one column to the next!
   - Reload the page to make sure everything is still working!

Apply to be InfPals Leader in 2022/2023