dnd-kit Animation Fix Guide: Preventing Bounce-Back

Table of Contents

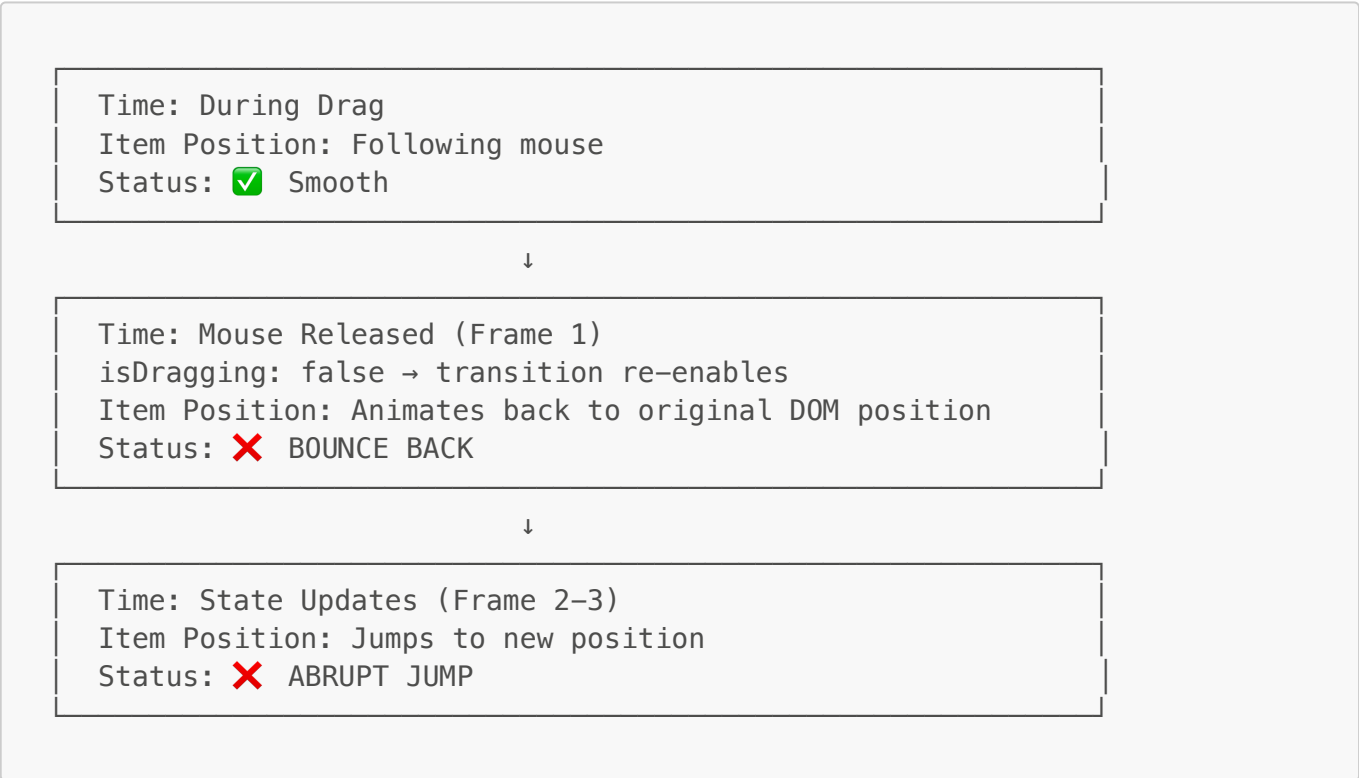
The Problem {#the-problem}

When dragging and dropping items (scripts or folders), after releasing the mouse, the item would:

1. ✅ Move to the new position instantly (optimistic update)
2. ❌ **Bounce back** to the original position
3. ❌ **Jump forward** to the new position (after state updates)

This created a jarring, unprofessional user experience.

Visual Timeline of the Bug



Root Cause Analysis {#root-cause-analysis}

The Original (Broken) Code

```
// ❌ BROKEN: Only checks wasDragging
animateLayoutChanges: (args) => {
    const { wasDragging } = args;
    if (wasDragging) return false;
    return defaultAnimateLayoutChanges(args);
},

const style: React.CSSProperties = {
    transform: CSS.Transform.toString(transform),
    transition: isDragging ? "none" : transition, // ❌ Problem here!
    opacity: isDragging ? 0 : 1,
};
```

Why It Broke

The issue is in the **transition timing**:

```
transition: isDragging ? "none" : transition;
```

Timeline:

1. **While dragging**: `isDragging = true` → `transition = "none"` ✅
2. **Mouse released**: `isDragging = false` → `transition = <enabled>` ❌
3. **Item still has transform** → Animates back to old DOM position
4. **State updates** → Item jumps to new position

The problem: **Transition re-enables too early** (as soon as mouse is released), while the item still has a transform applied and the DOM hasn't updated yet.

The Solution {#the-solution}

We need **two fixes**:

Fix 1: Check Both `isSorting` AND `wasDragging`

```
animateLayoutChanges: (args) => {
    const { isSorting, wasDragging } = args;
    // Disable all animations when actively sorting or just finished
dragging
    if (isSorting || wasDragging) return false;
    return defaultAnimateLayoutChanges(args);
},
```

Fix 2: Keep Transition Disabled While Transform Exists

```
const style: React.CSSProperties = {
    transform: CSS.Transform.toString(transform),
    transition: transform ? "none" : transition, // ← Changed from
isDragging
    opacity: isDragging ? 0 : 1,
};
```

Implementation Details {#implementation-details}

Understanding the dnd-kit Flags

`isDragging`

- **True**: While actively holding mouse and dragging
- **False**: As soon as mouse is released
- **Duration**: Entire drag operation

```
Mouse Down → isDragging = true
   ↓
Dragging around → isDragging = true
   ↓
Mouse Up → isDragging = false (immediate)
```

`isSorting`

- **True**: During the entire drag operation
- **False**: When drag operation ends
- **Duration**: Entire drag operation + slight delay

```
Mouse Down → isSorting = true
   ↓
Dragging around → isSorting = true
   ↓
Mouse Up → isSorting = false (in next render)
```

`wasDragging`

- **True**: For **ONE render cycle only** after releasing mouse
- **False**: After that single render cycle
- **Duration**: One frame after drag ends

```
Mouse Up → wasDragging = true (1 frame)
   ↓
```

```
Next Render → wasDragging = false
```

Why Check Both?

```
if (isSorting || wasDragging) return false;
```

- `isSorting`: Prevents animations **during** the drag
- `wasDragging`: Prevents animations **immediately after** release (the critical moment!)

The bounce was happening because we only checked `wasDragging` in `animateLayoutChanges`, but the **transition was re-enabling based on** `isDragging` in the style object.

How It Works {#how-it-works}

The Fixed Timeline

```
┌─────────────────────────────────────────────────────┐
│ Time: During Drag                                   │
│ isSorting: true                                     │
│ wasDragging: false                                  │
│ transform: exists                                   │
│ transition: "none" (transform exists)               │
│ Animation: Disabled ✅                              │
└─────────────────────────────────────────────────────┘
                          ↓
┌─────────────────────────────────────────────────────┐
│ Time: Mouse Released (Frame 1)                      │
│ isSorting: false (will be false next frame)         │
│ wasDragging: true (for this frame only!)            │
│ transform: still exists                             │
│ transition: "none" (transform still exists!)        │
│ Animation: Disabled ✅                              │
│ Result: No bounce back!                             │
└─────────────────────────────────────────────────────┘
                          ↓
┌─────────────────────────────────────────────────────┐
│ Time: State Updates (Frame 2)                       │
│ wasDragging: false                                  │
│ transform: null (cleared)                           │
│ transition: enabled (no transform anymore)          │
│ Animation: Re-enabled for future drags ✅           │
│ Item Position: Already at correct position!         │
└─────────────────────────────────────────────────────┘
```

The Key Insight

By using `transform ? "none" : transition` instead of `isDragging ? "none" : transition`:

- Transition stays disabled as long as **any transform exists**
- Transform clears **after** the DOM updates
- When transition re-enables, the item is **already in the correct position**
- No bounce! 🎉

Applied To All Components {#applied-to-all-components}

This fix was applied to **all sortable components** in the project:

1. `SortableScriptItem.tsx` (Scripts)

```
const {
    // ... other useSortable returns
} = useSortable({
    id: script.id || 0,
    data: {
        type: "script",
        script: script,
    },
    animateLayoutChanges: (args) => {
        const { isSorting, wasDragging } = args;
        if (isSorting || wasDragging) return false;
        return defaultAnimateLayoutChanges(args);
    },
});

const style: React.CSSProperties = {
    transform: CSS.Transform.toString(transform),
    transition: transform ? "none" : transition,
    opacity: isDragging ? 0 : 1,
    width: "100%",
    touchAction: "none",
};
```

2. `SortatbleCollapsableFolder.tsx` (Folders in Scripts Column)

```
const {
    // ... other useSortable returns
} = useSortable({
    id: folder.id!!,
    data: {
        type: "folder",
        folderId: folder.id,
        folder: folder,
    },
    animateLayoutChanges: (args) => {
```

```
        const { isSorting, wasDragging } = args;
        if (isSorting || wasDragging) return false;
        return defaultAnimateLayoutChanges(args);
    },
});

const style: React.CSSProperties = {
    transform: CSS.Transform.toString(transform),
    transition: transform ? "none" : transition,
    opacity: isDragging ? 0 : 1,
    width: "100%",
    height: "auto",
    minHeight: "fit-content",
    touchAction: "none",
};
```

3. `SortableFolderItem.tsx` (Folders in Folder Column)

```
const {
    // ... other useSortable returns
} = useSortable({
    id: folder.id,
    animateLayoutChanges: (args) => {
        const { isSorting, wasDragging } = args;
        if (isSorting || wasDragging) return false;
        return defaultAnimateLayoutChanges(args);
    },
});

const style: React.CSSProperties = {
    transform: CSS.Transform.toString(transform),
    transition: transform ? "none" : transition,
    opacity: isDragging ? 0.5 : 1,
    width: "100%",
    height: "auto",
    minHeight: "fit-content",
    touchAction: "none",
};
```

Before and After Comparison

Before (Broken)

```
// Only checks wasDragging (incomplete)
animateLayoutChanges: (args) => {
    const { wasDragging } = args;
    if (wasDragging) return false;
    return defaultAnimateLayoutChanges(args);
```

```
},

// Transition tied to isDragging (re-enables too early)
transition: isDragging ? "none" : transition,
```

Result: Bounce-back animation ❌

After (Fixed)

```
// Checks BOTH isSorting AND wasDragging
animateLayoutChanges: (args) => {
    const { isSorting, wasDragging } = args;
    if (isSorting || wasDragging) return false;
    return defaultAnimateLayoutChanges(args);
},

// Transition tied to transform existence (stays disabled until safe)
transition: transform ? "none" : transition,
```

Result: Smooth, bounce-free animations ✅

Key Takeaways

1. `isDragging` **alone is insufficient** for animation control because it changes too early
2. **Check both** `isSorting` **and** `wasDragging` in `animateLayoutChanges` to cover all phases of the drag operation
3. **Tie transition to** `transform` **existence**, not `isDragging` state, to prevent premature re-enabling
4. **Optimistic updates** combined with proper animation control create the smoothest UX
5. **Apply consistently** across all sortable components in your app

---

Related Issues Solved

By implementing these fixes, we also solved:

- ✅ Scripts bouncing when reordered within same folder
- ✅ Scripts bouncing when moved between folders
- ✅ Folders bouncing when reordered
- ✅ Scripts bouncing when dragged into/out of folders
- ✅ All abrupt "jump" transitions after drag operations

Debugging Tips

If you still see bouncing:

1. Check if optimistic updates are working

```
onQueryStarted: async (args, { dispatch, queryFulfilled }) => {
    console.log("Optimistic update happening...");
    const patchResult = dispatch(/* ... */);

    try {
        await queryFulfilled;
        console.log("✅ Mutation succeeded");
    } catch {
        console.log("❌ Mutation failed, rolling back");
        patchResult.undo();
    }
};
```

2. Check if `invalidatesTags` is causing refetch

```
// ❌ BAD: Refetch overwrites optimistic update
invalidatesTags: [{ type: "FolderContent" }],

// ✅ GOOD: Remove invalidatesTags, rely on optimistic updates
// Removed invalidatesTags - optimistic update handles the UI update
```

3. Add debug logging

```
const style: React.CSSProperties = {
    transform: CSS.Transform.toString(transform),
    transition: transform ? "none" : transition,
    opacity: isDragging ? 0 : 1,
};

console.log({
    transform: transform,
    transition: transform ? "none" : transition,
    isDragging,
});
```

4. Verify animation flags

```
animateLayoutChanges: (args) => {
    const { isSorting, wasDragging } = args;
    console.log({ isSorting, wasDragging });
    if (isSorting || wasDragging) return false;
    return defaultAnimateLayoutChanges(args);
},
```

Further Reading

- dnd-kit Animation Documentation
- Understanding useSortable Lifecycle
- CSS Transform and Transition
- React State Updates and Rendering

Summary

The bounce-back animation bug was caused by **premature transition re-enabling** after drag release. The fix requires:

1. **Checking both `isSorting` AND `wasDragging`** in `animateLayoutChanges`
2. **Tying transition to `transform` existence** instead of `isDragging` state
3. **Removing `invalidatesTags`** from mutations with optimistic updates

This creates a smooth, professional drag-and-drop experience with no visual glitches. 🎉