Lessons Learned: Simpson's Paradox Force Animation

This document captures pain points encountered while building the force-directed animation for the Simpson's Paradox visualization. These insights should inform library improvements to make similar visualizations easier to build.

Pain Points

1. D3 Force Caching Gotcha (CRITICAL)

The Problem: D3's `forceX().x(accessor)` **caches** the target values at initialization time. When we updated `node.gridX`, the force kept using the old cached values. The animation appeared broken - nodes wouldn't move to new positions.

What we tried:

```
Sim.updateGridXWithFn fourGroupsXFn sim
Sim.reheat sim  -- Nodes don't move!
```

What actually worked:

```
Sim.updateGridXWithFn fourGroupsXFn sim
currentNodes <- Sim.getNodes sim
_ <- Core.initializeForce forceXHandle currentNodes  -- Re-initialize!
Sim.reheat sim  -- Now nodes move
```

Library Improvement Ideas:

- Create `Sim.updateGridXAndReinit` that combines updating + re-initialization
- Document this caching behavior prominently
- Consider a force variant that always reads dynamically (at performance cost)
- Add a `reinitializePositionForces` helper

2. Debug Code Breaking Production

The Problem: Added debug logging with a `break` statement to only log first few nodes:

```
for (let i = 0; i < nodes.length; i++) {
  node.gridX = xFn(node);
  console.log("Node", i, "gridX:", node.gridX);
  if (i > 2) break;  // OOPS - stops the actual update loop!
}
```

Only 4 of 4526 nodes got updated.

Library Improvement Ideas:

- Keep debug utilities separate from core mutation functions
- Use a debug wrapper pattern instead of inline logging

3. Selection Type Mismatches (SEmpty vs SBoundOwns)

The Problem: `appendData` requires `SEmpty` selections, but `appendChild` returns `SBoundOwns`. Required awkward pattern:

```
_ <- appendChild Group [ class_ "dept-labels" ] g
deptLabelsParent <- select ".force-viz-svg .dept-labels"  -- Re-select!
renderDepartmentLabels config deptLabelsParent
```

Library Improvement Ideas:

- Make `appendData` polymorphic over selection state
- Provide `asEmpty :: Selection s e d -> Selection SEmpty e d` helper
- Or provide `appendDataTo` that works with any parent selection

4. FFI Location Policy

The Problem: Initially created local FFI (`ForceViz.js`) for `setGridX` mutation. User correctly noted FFI should live in the library (`psd3-simulation`) to keep demo code pure PureScript.

Library Improvement Ideas:

- Pre-build common node mutation patterns:
  - `updateGridX`, `updateGridY`, `updateGridXY`
  - `updateNodeProperty :: String -> (node -> a) -> Simulation -> Effect Unit`
- Document the "no FFI in app code" policy

5. Monad Context Mixing (D3v2M vs Effect)

The Problem: Inside `runD3v2M` block, calling `Sim.getNodes` (returns `Effect`) requires `liftEffect`:

```
nodeSel <- runD3v2M do
  -- ... D3v2M code ...
  currentNodes <- liftEffect $ Sim.getNodes sim  -- Easy to forget!
```

Library Improvement Ideas:

- Consider providing `D3v2M` wrappers for common simulation queries
- Or use `MonadEffect` constraint more consistently

6. Type Annotations for Data-Bound Attributes

The Problem: Type inference couldn't figure out datum types in attribute accessors:

```
-- Compile error without explicit annotation
y (\d -> genderY config d.gender)

-- Works with annotation
y (\(d :: GenderLabel) -> genderY config d.gender)
textContent (_.label :: GenderLabel -> String)
```

Library Improvement Ideas:

- Provide typed attribute helpers: `yTyped :: forall d. (d -> Number) -> Attribute d`
- Better type inference through functional dependencies
- Document this requirement prominently

7. Multi-Phase Animation Timing

The Problem: Needed `setTimeout` for multi-phase animation, but `Effect.Timer` wasn't available. Had to add local FFI for `setTimeout_`.

Library Improvement Ideas:

- Include timing utilities in the library
- Or provide a `Sim.schedulePhase :: Int -> Effect Unit -> Effect Unit`
- Consider an animation sequencing DSL

Summary

The biggest issue by far was **#1 (force caching)**. This was completely non-obvious and took significant debugging to identify. The force appeared to be running (simulation was active) but targets weren't updating.

Recommended Priority for Library Improvements:

1. Fix force caching issue (provide `updateAndReinit` or similar)
2. Document force caching behavior
3. Add common node mutation helpers to library
4. Improve selection type ergonomics