# Consolidated TODO

This document consolidates all active TODOs from the project, organized by priority and area.

---

## High Priority

### Visualization Improvements

**Chord Diagram** (needs work)

- [ ] Add labels to arcs and chords
- [ ] Source interesting dataset (trade flows, migration patterns, music genre relationships)
- [ ] Respect container height
- [ ] Hover interactions: highlight connected arcs
- [ ] Accessible color scheme

**Bubble Chart** (needs work)

- [ ] Find own interesting dataset (not Flare)
- [ ] Click-to-zoom: click inner bubble to drill down, click outer to go back up
- [ ] Fix labeling: drop tiny labels, hide labels that don't fit
- [ ] Respect container height
- [ ] Stretch: Live transition to tree, treemap, or graph layout

**Sankey Diagram** (needs work)

- [ ] Add margins for proper spacing
- [ ] Hover highlighting for flow paths
- [ ] Find more up-to-date energy flow data
- [ ] "Primary energy fallacy" color toggle
- [ ] Readable labels
- [ ] Smooth transitions

**Hierarchical Data Displays** (needs comprehensive page)

- [ ] Horizontal tree (centering, zoom)
- [ ] Vertical tree (centering, zoom, 45° rotated labels)
- [ ] Radial tree (centering, zoom, click-to-rotate)
- [ ] Circle packing
- [ ] Treemap (new - needs implementation)
- [ ] Control panel to switch between layouts
- [ ] Code overlay panels
- [ ] Use our own codebase as data source
- [ ] Fix three-layout switcher example (currently not working)
- [ ] Stretch: Animate transitions between layouts
- [ ] Stretch - FINAL BOSS: Animate tree into force layout graph

### Anscombe's Quartet

- [ ] Show summary statistics alongside charts

---

## Medium Priority

### Code Atlas Enhancements

- [ ] Module cohesion metrics visualization
- [ ] Package coupling visualization
- [ ] Evolve "code-explorer" into package set explorer
- [ ] Evolve "code-atlas" into project explorer
- [ ] Mock up AWS or Kubernetes explorer
- [ ] Highlight "candidate for extraction" functions
- [ ] Demonstrate concrete insight about codebase
- [ ] "Blueprint" CSS style - architectural drawing aesthetic

### Understanding Section Completion

- [ ] Complete "Grammar of D3 in SelectionM" section in Patterns page
- [ ] Complete "From DOM to Visualization Elements" section in Patterns page
- [ ] Add more practical examples to Patterns page
- [ ] Create simpler diagrams for SelectionM grammar flow and General Update Pattern
- [ ] Add interactive code examples to Understanding pages
- [ ] Link Understanding pages to relevant Tutorial and How-to examples

### API Reference Section

- [ ] Add detailed explanations/prose for each module
- [ ] Add usage examples for key functions
- [ ] Add diagrams showing relationships between modules
- [ ] Add "See Also" links between related modules
- [ ] Generate API documentation from source code comments
- [ ] Add search functionality across all modules

### Tour/Interpreters Page

**Four-panel display of one PureScript code interpreted as:**

- [ ] English description
- [ ] D3.js JavaScript code
- [ ] Vega-Lite JSON specification
- [ ] AST visualization as tree

**Supporting content:**

- [ ] Diagram explaining Finally Tagless pattern
- [ ] EBNF-style grammar documentation for SelectionM
- [ ] Overhaul string interpreter output (newlines, indentation)

- [ ] Create Vega-Lite interpreter (new)
- [ ] Create English-language interpreter (new)
- [ ] Create AST-to-tree-data interpreter (new)
- [ ] Example selector feature with syntax highlighting

### Documentation Improvements

- [ ] Complete How-to guides for all common patterns
- [ ] Add module-level documentation to all core modules
- [ ] Document capability/interpreter relationship in more depth
- [ ] Add inline comments for complex FFI interactions
- [ ] Add more code examples throughout documentation
- [ ] Create "migration guide" for D3.js users

---

## Lower Priority

### Site Infrastructure

**Routing/Navigation:**

- [ ] Eliminate Gallery routing layer (if still present)
- [ ] Make routing fully idiomatic using `purescript-routing`

**CSS/Layout:**

- [ ] Good responsive CSS for readability across screen sizes
- [ ] Replace remaining Ocelot-derived components with modern CSS
- [ ] Responsive design improvements for all screen sizes
- [ ] Consider thumbnail navigation for examples
- [ ] Clean up and possibly delete Ocelot-derived code

**Accessibility:**

- [ ] Main text appears and is navigable for screen readers
- [ ] Handle no-JS case gracefully
- [ ] Explain what sighted users see/do for non-visual users
- [ ] Stretch: Sonic "visualizations" or audio descriptions

**Home Page:**

- [ ] "Why PS<$>D3?" section explaining Finally Tagless benefits
- [ ] Quick code example on home page
- [ ] Consider generating sections from markdown

### Library API Improvements

**SimulationM Refactoring:**

- [ ] SimpleSimulationM - easier API for straightforward force simulations
- [ ] AdvancedSimulationM - full-featured API with General Update Pattern support

- [ ] Rename SimulationM/SimulationM2 to clearer names
- [ ] Add "Common Patterns" cookbook to docs
- [ ] Helper utilities for simulation data manipulation
- [ ] Scaffolding for common event handler patterns
- [ ] Custom type error hints for common mistakes

**Testing:**

- [ ] Test suite for D3 API wrapper
- [ ] Unit tests for core Selection and Simulation operations
- [ ] Integration tests for common patterns
- [ ] Property-based testing where applicable

### Interactive Wizard

- [ ] Design wizard flow for beginners
- [ ] Step-by-step guided tour through library concepts
- [ ] Interactive code editing/playground
- [ ] "Choose Your Own Adventure" style branching
- [ ] Progress tracking
- [ ] Link wizard completions to relevant documentation sections

### Data and Build Pipeline

- [ ] Consider additional metrics beyond basic dependencies
- [ ] Set up PostgreSQL/SQLite for example datasets (future)
- [ ] Create HTTP API server (future)
- [ ] Migrate static data files to database (future)
- [ ] Update examples to fetch via `purescript-affjax` (future)

---

## Stretch Goals / Future

- [ ] Add e-charts support
- [ ] Parser to convert markdown to tree data
- [ ] Additional advanced visualizations (parallel coordinates, hexbin, etc.)
- [ ] Other interpreters beyond visualization
- [ ] Video tutorials for complex topics
- [ ] Interactive playground with live code editing
- [ ] Performance benchmarking suite

---

## Future Plans (Separate Efforts)

### Code Explorer Integration into Understanding Pages

Use Code Explorer as running example across all Understanding pages, making abstract concepts concrete. Per-page integration planned for:

- Grammar Page - show select, appendChild, joinData, setAttrs

- Attributes Page - static values, datum functions, contravariant
- Selections Page - state machine in action, enter/update/exit
- TreeAPI Page - layer cake approach
- Scenes Page - transition matrix

### MetaTree Editor Vision

WYSIWYG visualization coding platform:

- Interactive MetaTree Editor (visual tree, drag-and-drop, property editing)
- Live Preview pane
- Code Generation pane
- Data panel with type matching guidance
- try.purescript.org backend integration for advanced lambda expressions

---

## Additional How-Tos Needed

### Visualization Types (Basic Shells)

- Chord diagram
- Sankey diagram
- Treemap
- Icicle
- Bubblepack
- Tree

### Force Layout Additions

- Scene management
- Transition matrix
- Force library (turning forces on/off)
- Turning simulation on/off
- Use transitions within force layouts
- Create a custom force

### Other Topics

- "Bless JSON without parsing it"
- "Write a JSON parser with Argonaut"
- Event delegation with selections
- Debug.spy usage
- Adding tooltips
- Link generators (h, v, r, iso)
- Animated trees, update trees, expand/collapse trees
- Zooming treemap example

---

## Notes

- Priorities can be adjusted based on what's most impactful for demonstrating the library
- "Stretch goals" within sections can be deferred to later phases
- Focus on polish and education: each example should be both impressive and instructive