Dependency Chain Visualization

```
Step 1: Create DOM
    ↓
Step 2: Register Services
    ↓
Step 3: Get eventBus, shapeRegistry
    ↓
Step 4: Load Shapes into shapeRegistry
    ↓
Step 5: Setup UI (LeftPalette uses shapeRegistry)
    ↓
Step 6: Get editor, managers
    ↓
Step 7: Setup Event Handlers (uses all managers)
```

Technically valid alternative order

```
// Option A: Current order (shown in code)
this.createDOMStructure();
ServiceProvider.register(this.container);
this.eventBus = this.container.get("eventBus");
this.shapeRegistry = this.container.get("shapeRegistry");
ShapeLoader.loadBuiltInShapes(this.shapeRegistry);
this.setupUI();

// Option B: Register services first
ServiceProvider.register(this.container);
this.createDOMStructure(); // ← moved here
this.eventBus = this.container.get("eventBus");
this.shapeRegistry = this.container.get("shapeRegistry");
ShapeLoader.loadBuiltInShapes(this.shapeRegistry);
this.setupUI();
```

Best Practice

```
// Step 1: Create DOM structure first
this.createDOMStructure();

// Step 2: Register all services
ServiceProvider.register(this.container);
```

Reasons:

1. **Clarity** — You prepare the canvas (DOM) first, then populate it with services
2. **Psychology** — DOM structure first makes logical sense (prepare containers before populating)
3. **Debugging** — If DOM creation fails, you catch it early before setting up complex services
4. **Consistency** — Creates a clear mental model: "prepare, then populate"

What Really Matters

The **true dependencies** are:

```
Step 4: Load Shapes
↑ Requires shapeRegistry (from Step 3)

Step 5: Setup UI
↑ Requires DOM (from Step 1)
↑ Requires shapeRegistry (from Step 4)
↑ Requires services like eventBus

Step 7: Setup Event Handlers
↑ Requires DOM (from Step 1)
↑ Requires all managers (from Step 6)
```