Lucky in the Cloud
With Diagrams

Jan Köhnlein
Miro Spönemann

@jankoehnlein
@sponemann

TypeFox
Sprotty
Live Diagrams in Your Web Application

Package Dependency Graph for npm

void-elements

html-parse-stringify2

snabbdom

snabbdom-virtualize

file-saver

snabbdom-jsx

inversify

sprotty

Clear  Resolve All  Filter dependencies

Brought to you by TypeFox  Powered by sprotty and ELK
Live Diagrams in Your Web IDE
Outline

• Base architecture
  ‣ Main components
  ‣ The model source
  ‣ Example: package dependency graph

• Layout
  ‣ Micro layout and macro layout
  ‣ Hidden rendering

• Configuration
Base Architecture
Base Architecture

- **Model Source**
- **Action Dispatcher**
- **Command Stack**
- **Viewer**
- **SModel**
- **View**
- **DOM**
- **Virtual DOM**

**Actions**:
- Model Source to Action Dispatcher
- Command to Command Stack
- Command Stack to Viewer
- Viewer to View
- View to DOM
- DOM to Virtual DOM

**Events**:
- Viewer to events
- View to create
- DOM to patch

**Calls**:
- Viewer to call
Example: Selection

SelectCommand
execute(context)

Command Stack
{
    type: "node"
    id: "node23"
    selected: false
}

Action Dispatcher

Viewer
mousedown

DOM

SelectAction
{
    kind: "elementSelected"
    selectedElementsIDs: ["node23"]
}
Example: Rendering

```
render(node, context)

Viewer ➔ call ➔ View ➔ create ➔ DOM ➔ patch ➔ Virtual DOM

<rect
  class-selected={node.selected}
  width={node.size.width}
  height={node.size.height}
/>
```

`npmjs.com/package/snabbdom`
The Model Source

DiagramServer (local proxy)

DiagramServer (remote)

LocalModelSource

Model Source

Action Dispatcher

WebSocket

TypeScript API

Java API

Domain-Specific Languages in the Cloud – With Eclipse Technologies
Wednesday 16:30 — 17:05
Local Model Source

LocalModelSource

- currentRoot: SModelRootSchema
- setModel(newRoot)
- updateModel()
- protected handleRequestModel(action)
- protected handleComputedBounds(action)
- protected handle...
Internal and External Model

SNode (internal)
- parent: SParentElement
- root: SModelRoot
- bounds: Bounds

getAnchor(referencePoint)
getTranslatedAnchor(refPoint, refContainer, edge)

SNodeSchema (external)
- type: string
- id: string
- children: SModelElementSchema[]
- position: Point
- size: Dimension
- selected: boolean

Model Source ➔ Action Dispatcher ➔ Command Stack ➔ Viewer ➔ SModelFactory
Model Updates

**UpdateModelCommand**

```javascript
execute(context)
```

**Model Source**

```javascript
UpdateModelAction
{
    kind: "updateModel"
    newRoot: { ... }
}
```

**Command Stack**

animate changes (fade in, fade out, move, etc.)

**Viewer**
Example: Package Dependency Graph

LocalModelSource

DepGraphModelSource

- createNode(name)
- resolveNodes(nodes)
- resolveGraph()
- filter(text)
- clear()

- protected handleSelect(action)
- protected handleSelectAll(action)
Resolve Dependencies on Selection

```javascript
handleSelect(action)

resolveNodes(nodes)

resolveNode(node)

https://registry.npmjs.org/sprotty

{
  type: "node",
  id: "sprotty",
  name: "sprotty"
}
```
Resolve Dependencies on Selection

handleSelect(action)

resolveNodes(nodes)

resolveNode(node)

addDependencies(node, dependencies)

updateModel()

https://registry.npmjs.org/sprotty

package metadata
Layout
Micro Layout and Macro Layout

**Micro Layout**
- Compartments, labels, icons, etc.
- Horizontal Box
- Vertical Box
- Stacked

**Macro Layout**
- Networks of nodes and edges
- Layer-based
- Force-based
- Planarization
Micro Layout and Macro Layout

- SVG has no means for automatic micro layout
- Text size depends on CSS
- Node size depends on text size
- Macro layout depends on node size
RequestBoundsCommand

execute(context)

render new model
in the hidden DOM

Model
Source

Action Dispatcher

Viewer

DOM

Virtual DOM

Virtual DOM

DOM

View
Hidden Rendering Cycle

Model Source

Action Dispatcher

Command Stack

Viewer

Virtual DOM

DOM

ComputedBoundsAction

{ 
  kind: "computedBounds"
  bounds: [ ... ]
}

compute macro layout

copy computed bounds into the model

compute micro layout

call

create

patch
Hidden Rendering Cycle

Model Source

UpdateModelAction

{  
  kind: "updateModel"
  newRoot: { ... }
}

Action Dispatcher

UpdateModelCommand

execute(context)

Command Stack

animate changes
(fade in, fade out, move, etc.)

Viewer

call

View

create

DOM

patch

Virtual DOM
Configuration
Configuration

Dependency Injection

bind(TYPES.ModelSource).to(DepGraphModelSource).inSingletonScope()

 npmjs.com/package/inversify

Model and View

configureModelElement(context, 'node', DependencyGraphNode, DependencyNodeView)

Model Class  View Class
Evaluate the Sessions

Sign in and vote at eclipsecon.org

-1  0  +1