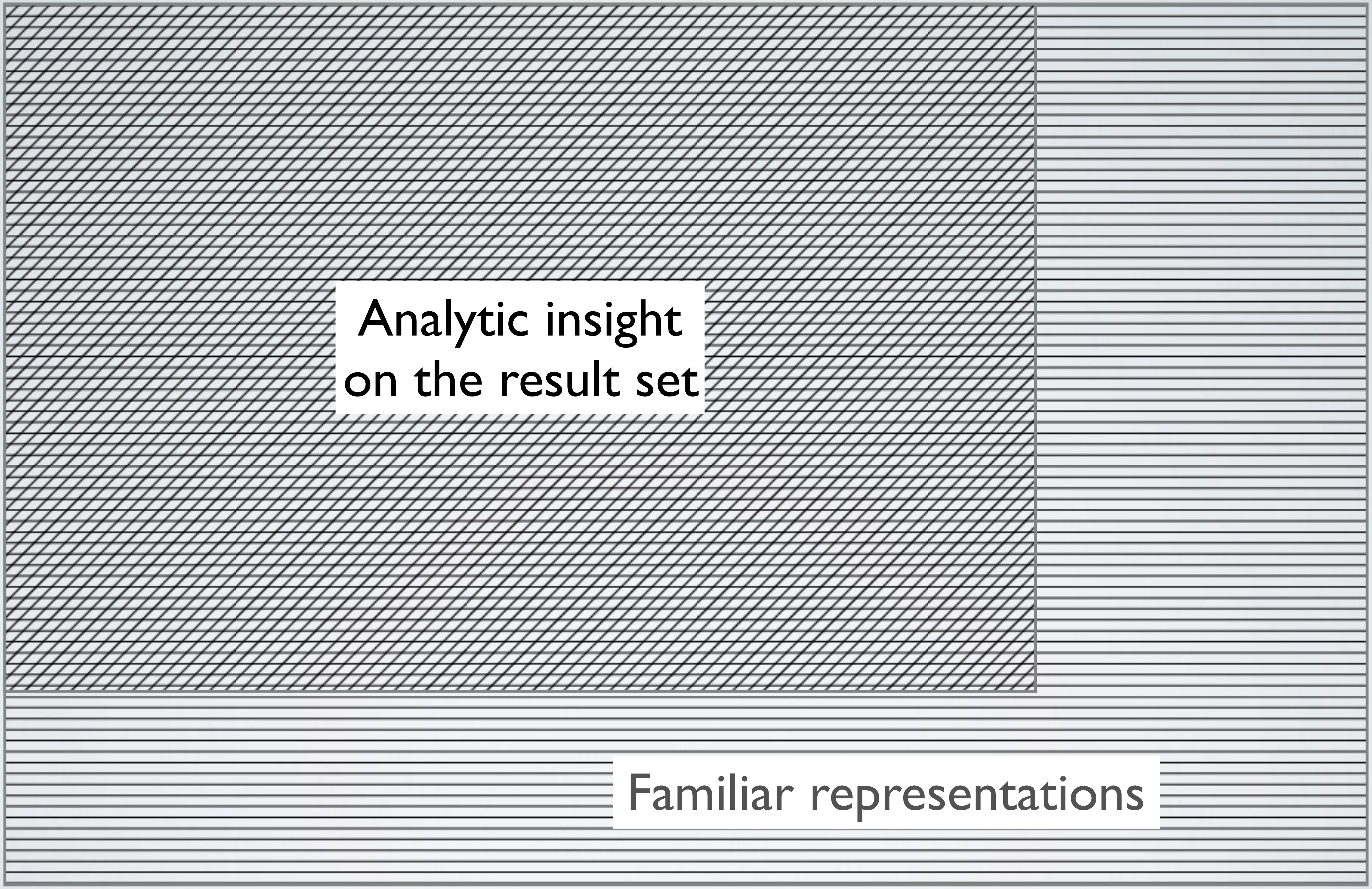
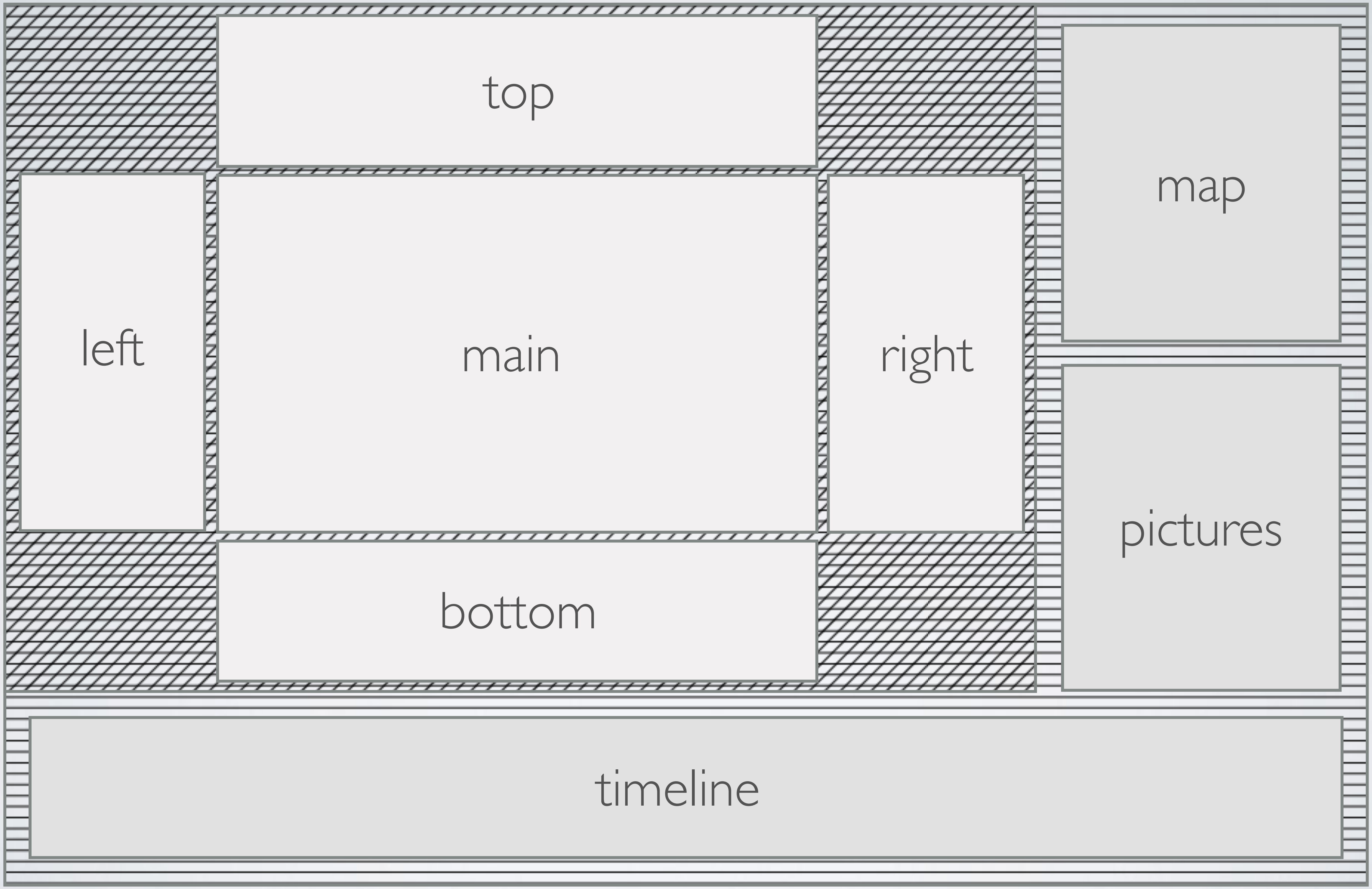
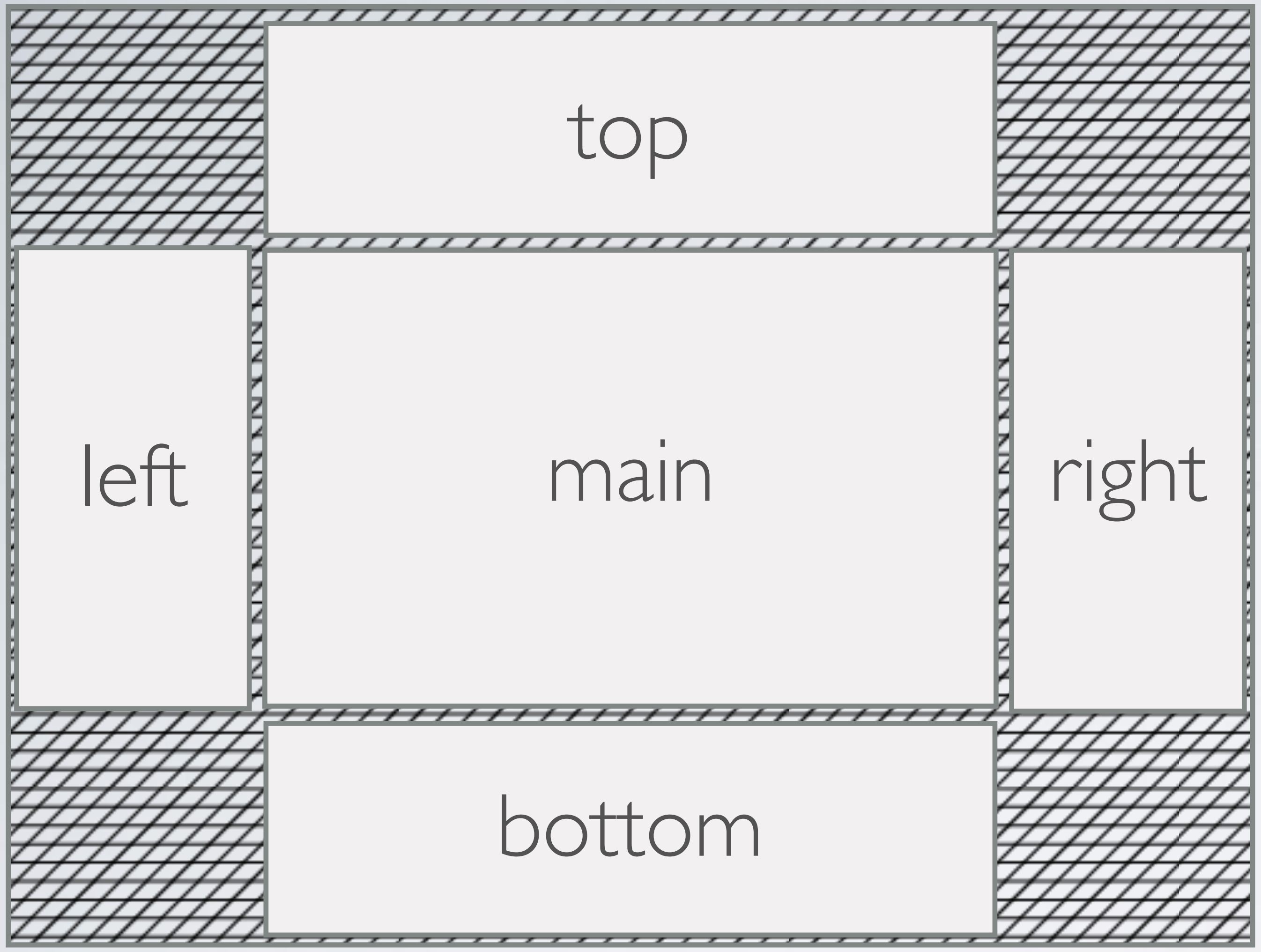
SEMANTIC LIBRARIES

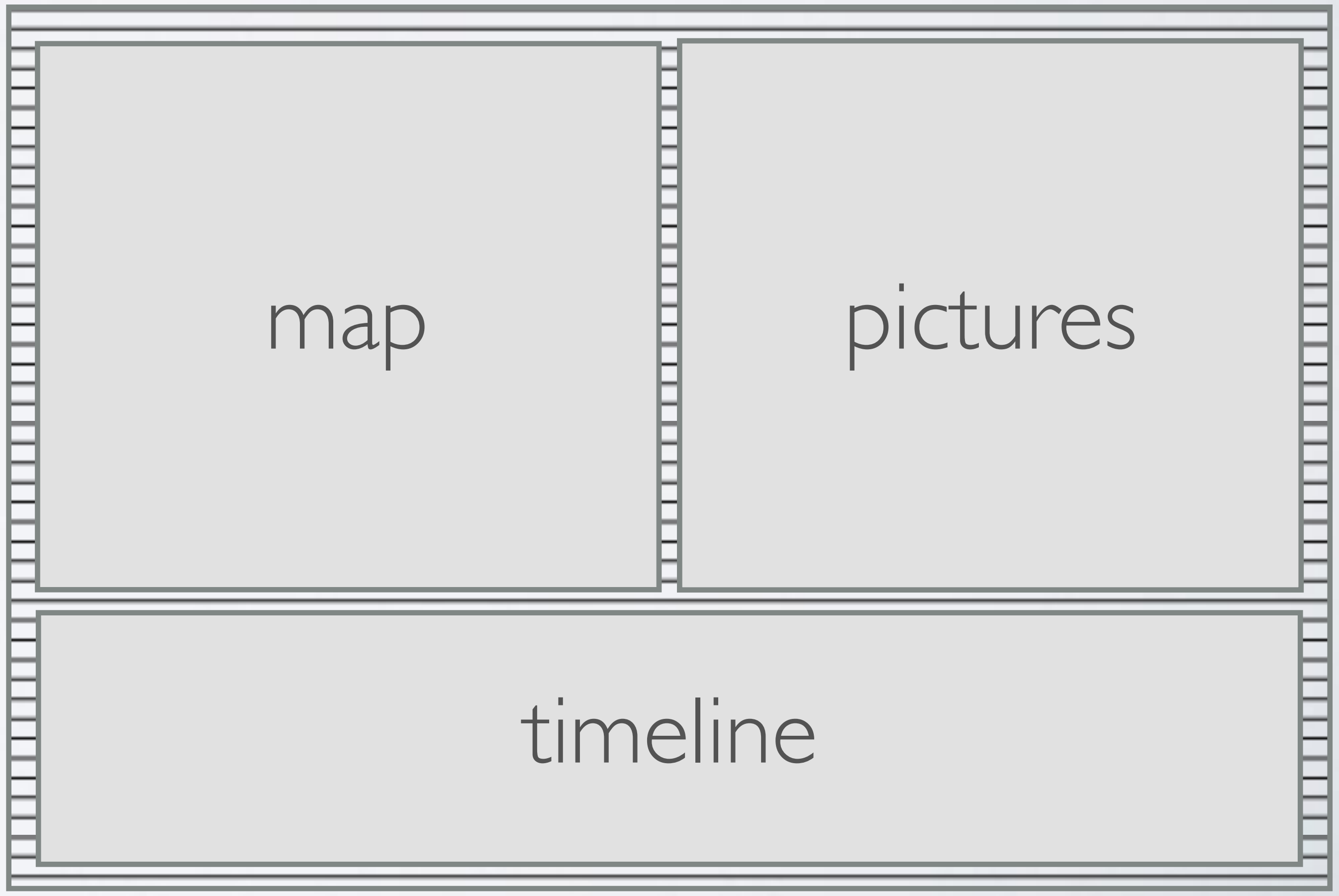
12/10/2017

INTERFACE

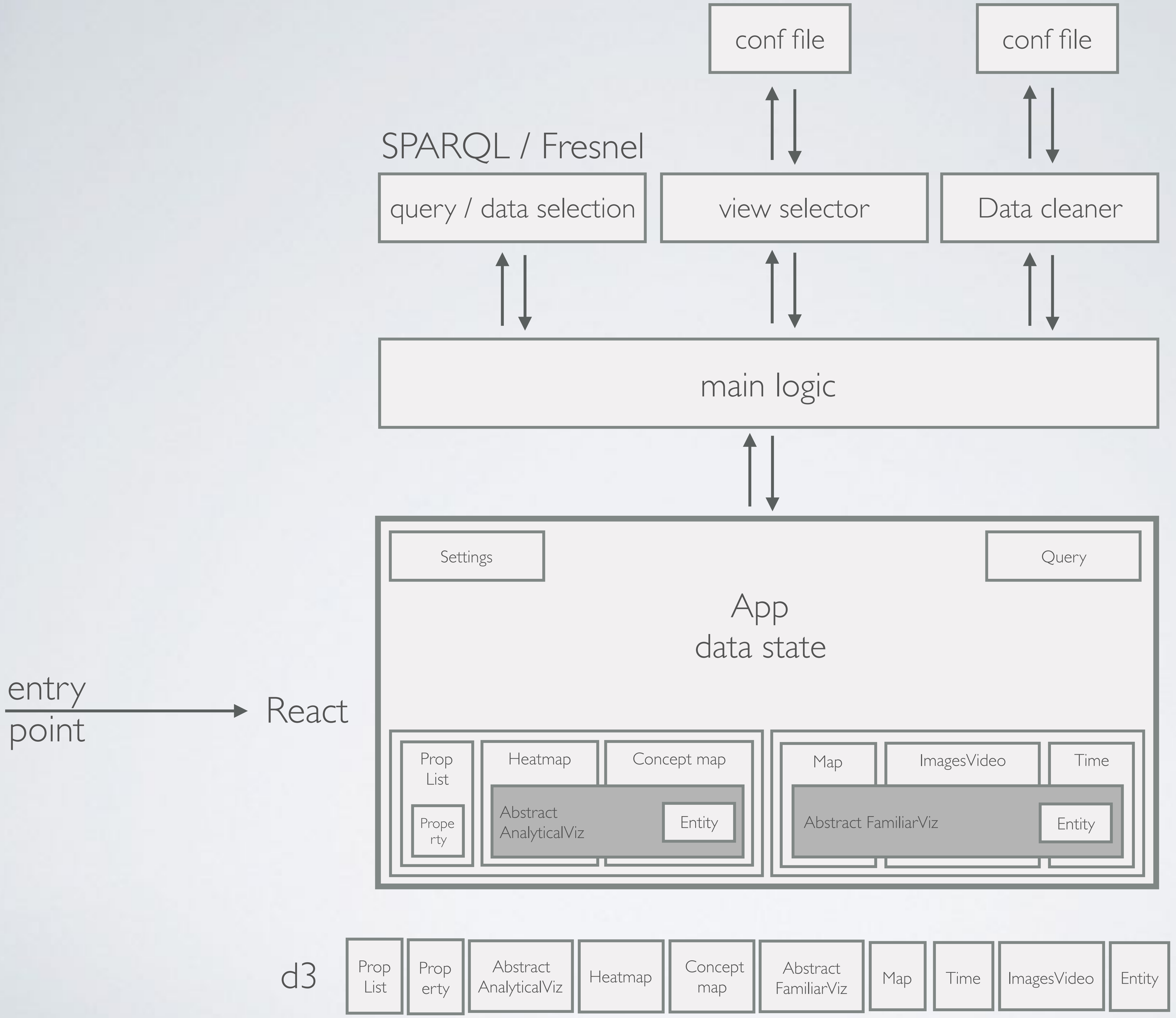

Analytic insight on the result set
Familiar representations


top
map
left
main
right
pictures
bottom
timeline


top
left
main
right
bottom


map
pictures
timeline

ARCHITECTURE


conf file
conf file
SPARQL / Fresnel
query / data selection
view selector
Data cleaner
Fresnel
main logic
Settings
Query
App
data state
entry
point
React
Prop List
Prope rty
Heatmap
Concept map
Abstract AnalyticalViz
Entity
Map
ImagesVideo
Time
Abstract FamiliarViz
Entity
d3
Prop List
Prop erty
Abstract AnalyticalViz
Heatmap
Concept map
Abstract FamiliarViz
Map
Time
ImagesVideo
Entity

query / data selection

INITIAL QUERY
constraint =
collection of entities
of the same type ?

SECONDARY QUERY
Retrieves related lenses

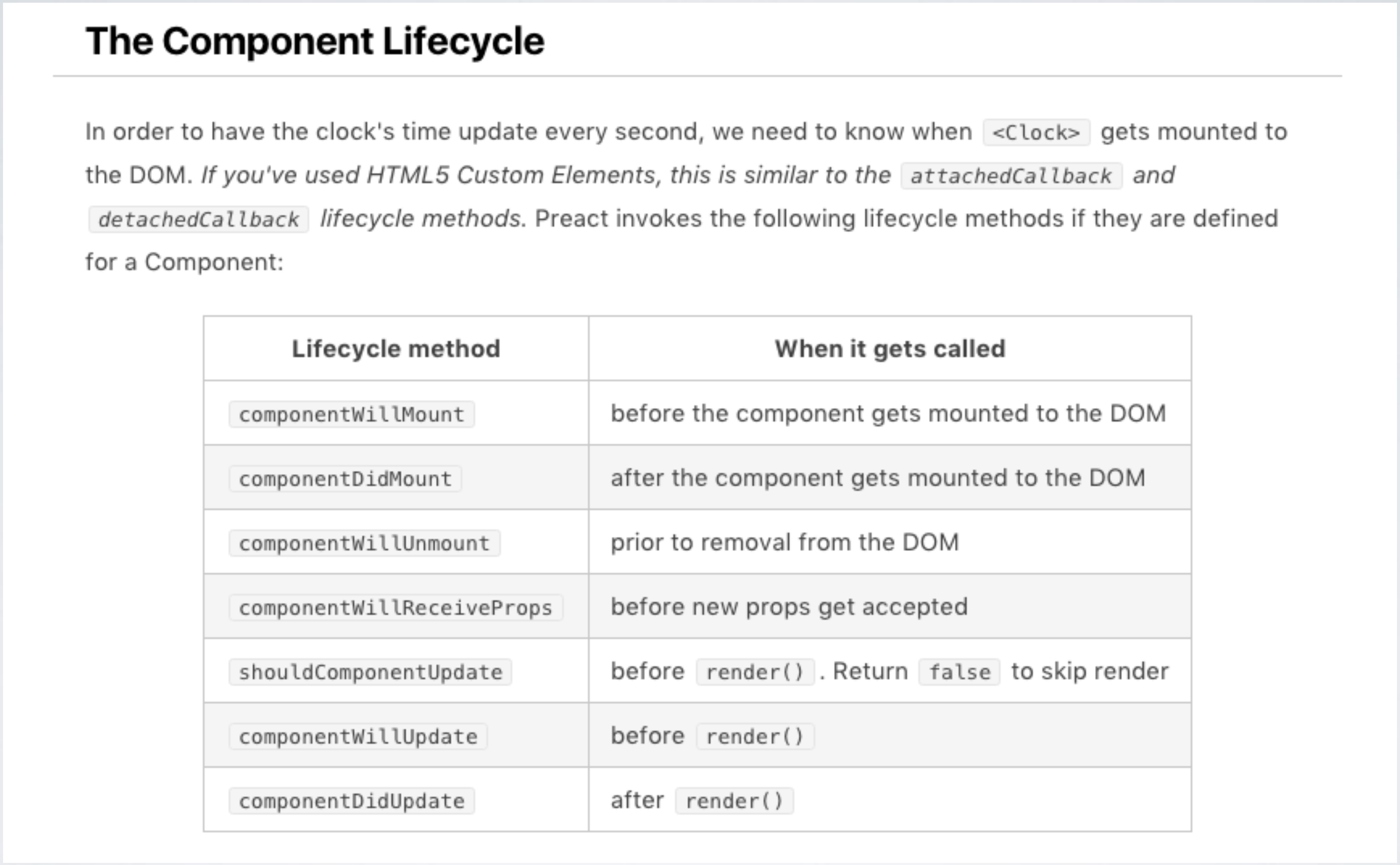
Data cleaner

adds a domain
specific layer
with grouping mechanisms
(fuzzy matching
on names for certain entities)

view selector

selects the best view
based on the distribution and volume
of the set of results

REACT

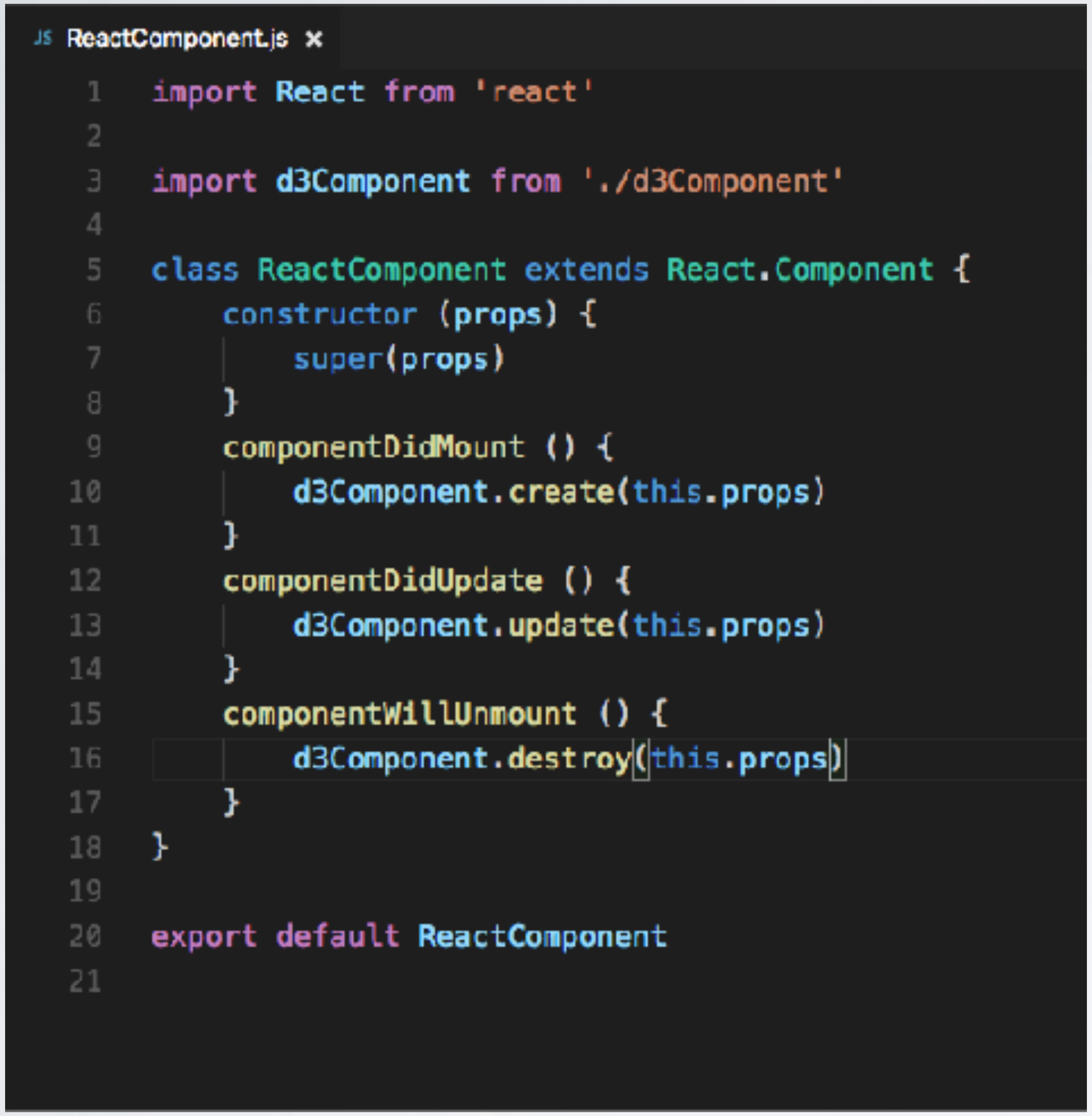


The Component Lifecycle

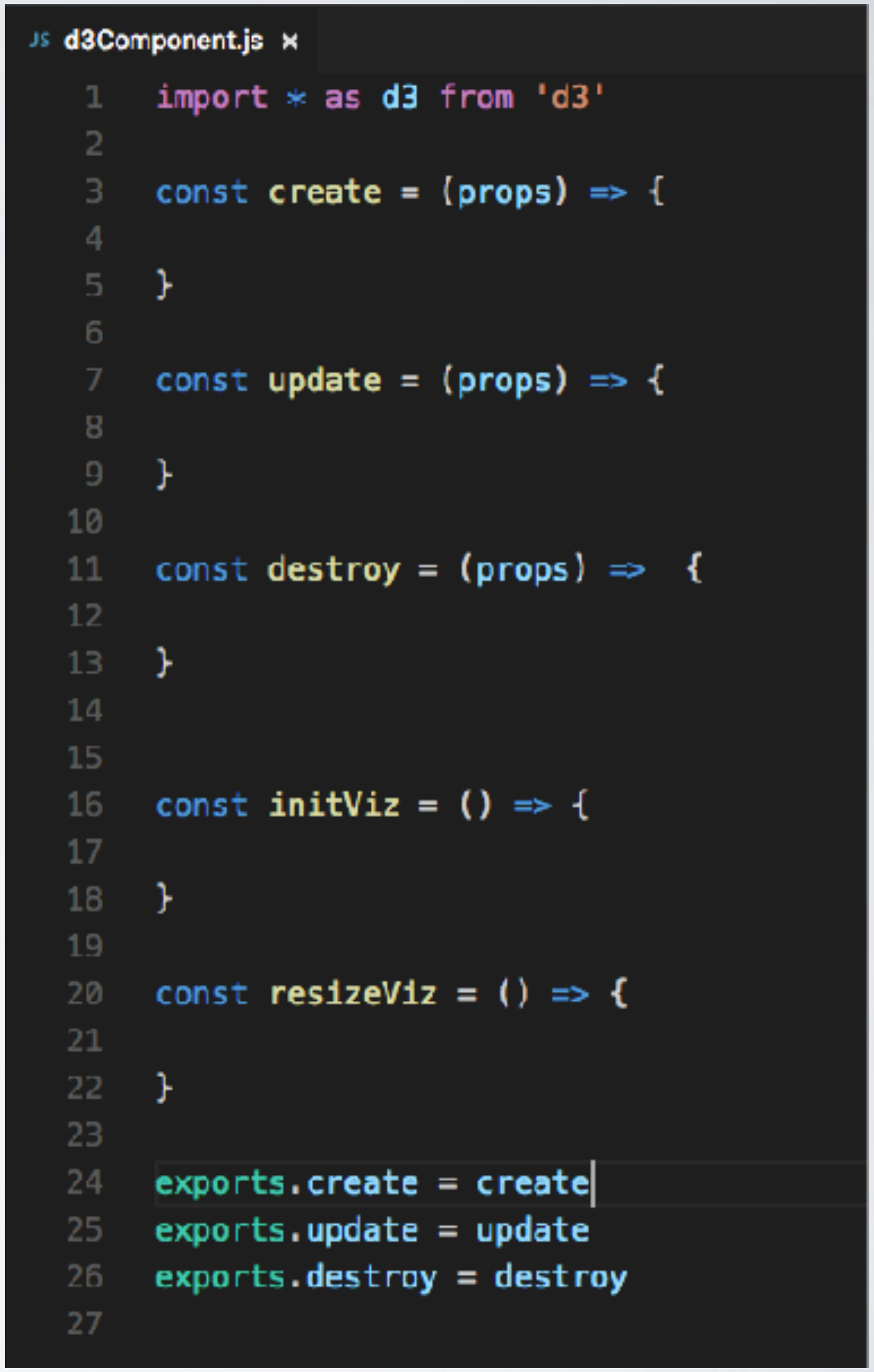
In order to have the clock's time update every second, we need to know when `<Clock>` gets mounted to the DOM. *If you've used HTML5 Custom Elements, this is similar to the* `attachedCallback` *and* `detachedCallback` *lifecycle methods.* Preact invokes the following lifecycle methods if they are defined for a Component:

Lifecycle method	When it gets called
`componentWillMount`	before the component gets mounted to the DOM
`componentDidMount`	after the component gets mounted to the DOM
`componentWillUnmount`	prior to removal from the DOM
`componentWillReceiveProps`	before new props get accepted
`shouldComponentUpdate`	before `render()` . Return `false` to skip render
`componentWillUpdate`	before `render()`
`componentDidUpdate`	after `render()`

&… Most popular framework : well maintained, lots of libraries, easy to find help & solutions, designed for tests


ReactComponent.js

```
import React from 'react'

import d3Component from './d3Component'

class ReactComponent extends React.Component {
    constructor (props) {
        super(props)
    }
    componentDidMount () {
        d3Component.create(this.props)
    }
    componentDidUpdate () {
        d3Component.update(this.props)
    }
    componentWillUnmount () {
        d3Component.destroy(this.props)
    }
}

export default ReactComponent

```

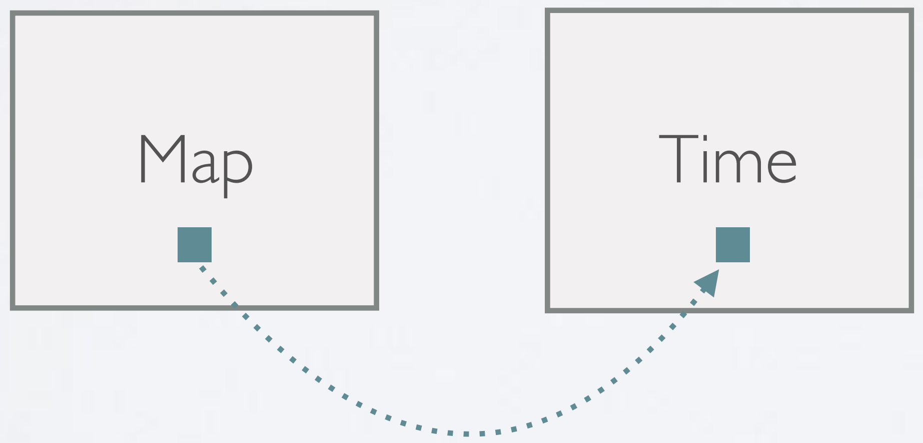

d3Component.js

```
import * as d3 from 'd3'

const create = (props) => {

}

const update = (props) => {

}

const destroy = (props) =>  {

}


const initViz = () => {

}

const resizeViz = () => {

}

exports.create = create
exports.update = update
exports.destroy = destroy

```

REACT

- Advanced transitions between components

E.g. in Dylan’s work: elements going from the timeline to the map -> which component is in charge of them ? Can we find a mechanism to let them “travel”easily ? an Entity component with shape and display attributes, that can be easily destroyed and replicated ?



- Frameworks evolves quickly -> PREACT !