Architecture of Hybrid Languages

Dblab141-F

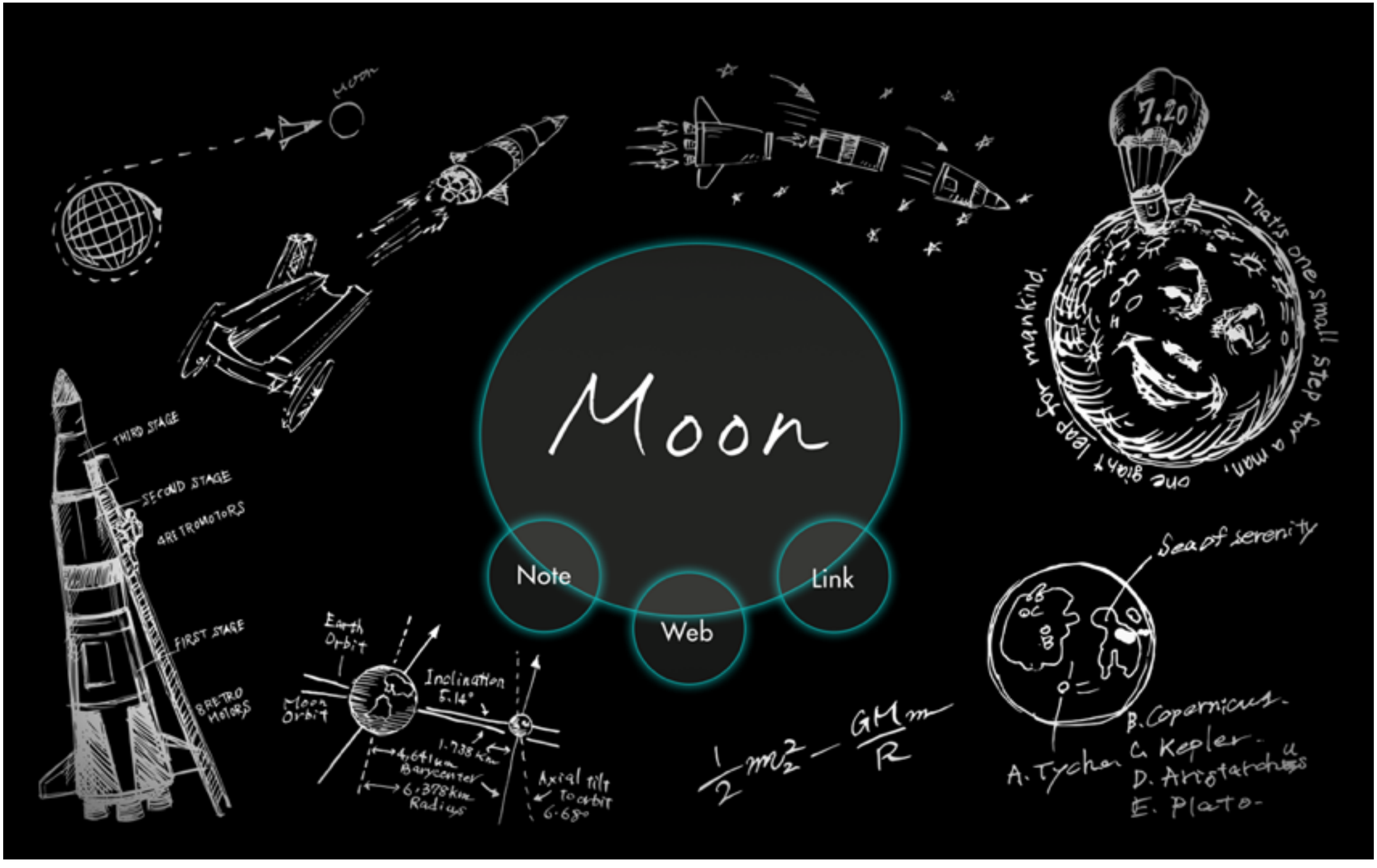
d8161105 MiraiWatanabe

Background - Visual Languages

- Education
- Algorithm
- Network
- Software Engineering
- Infrastructure

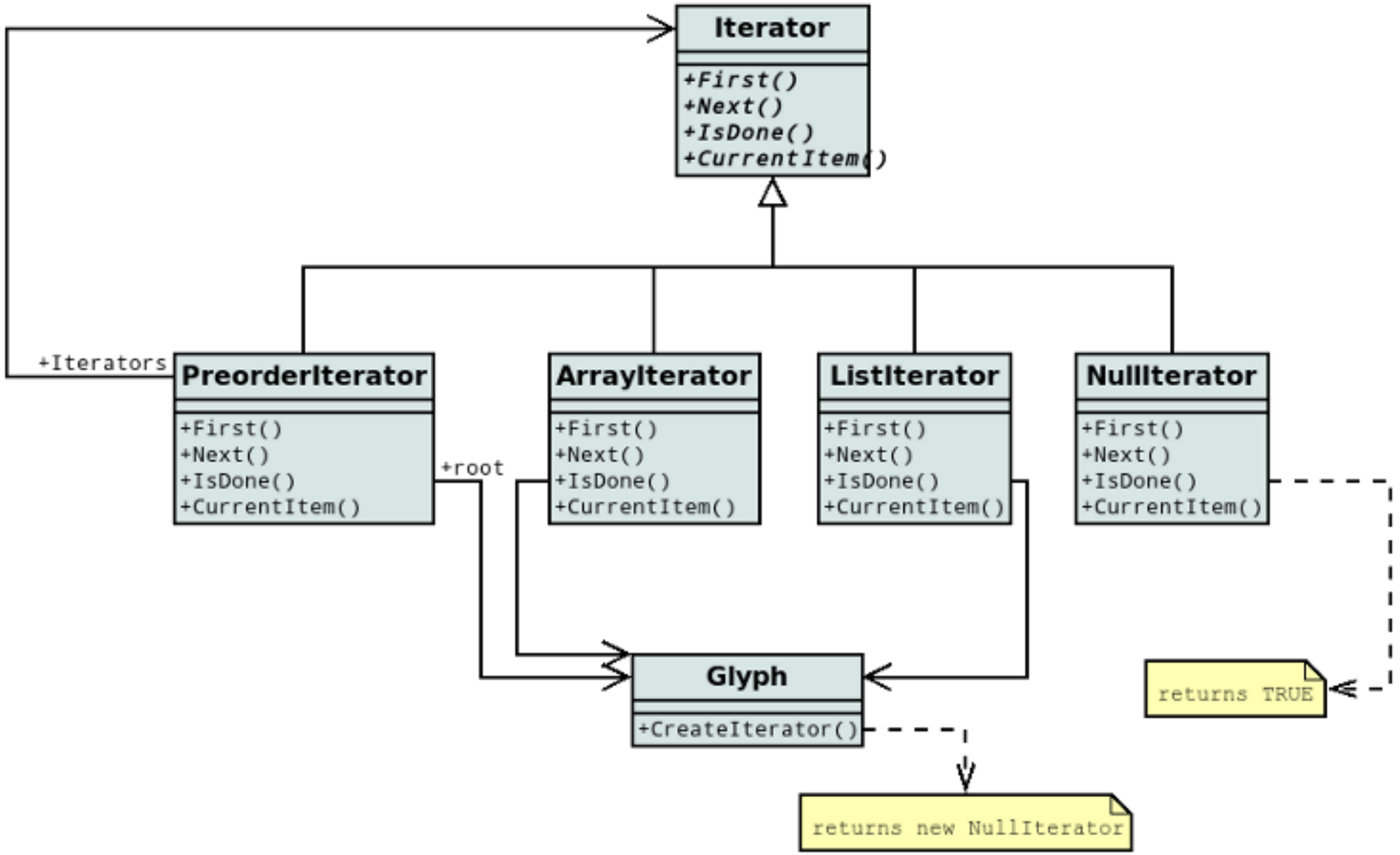
Background - Visual Languages - Scratch

Background - Visual Languages - enchantMoon

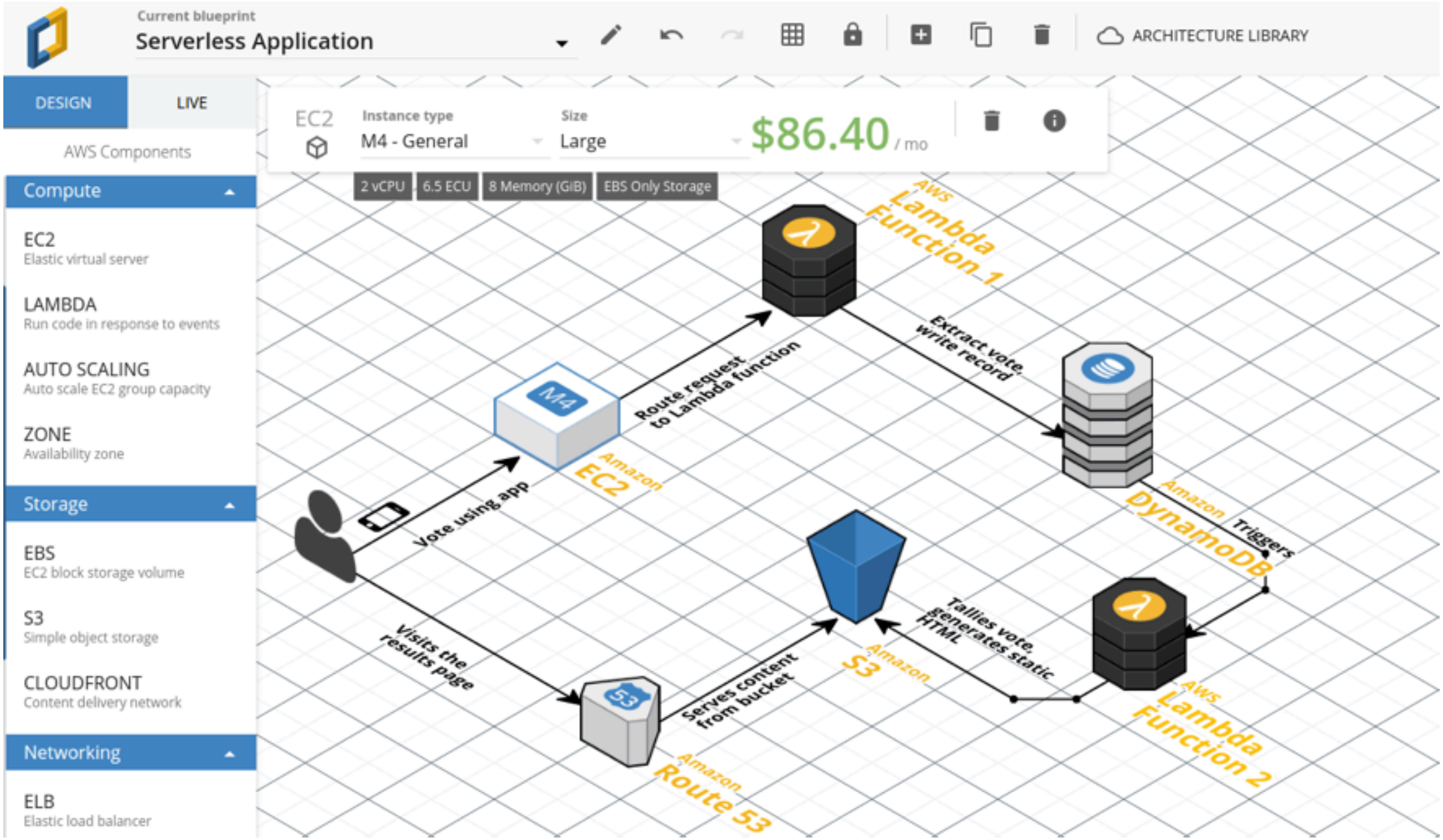


実は開発者は布留川さん 会津大２期生

Background - Visual Languages - UML

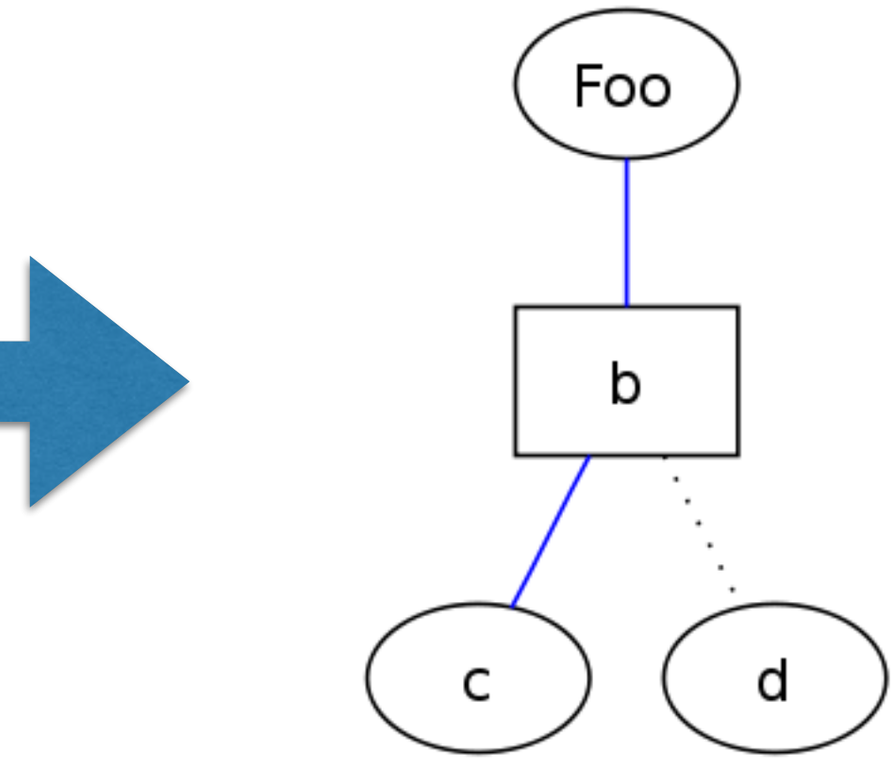


Background - Visual Languages - AWS CloudCraft



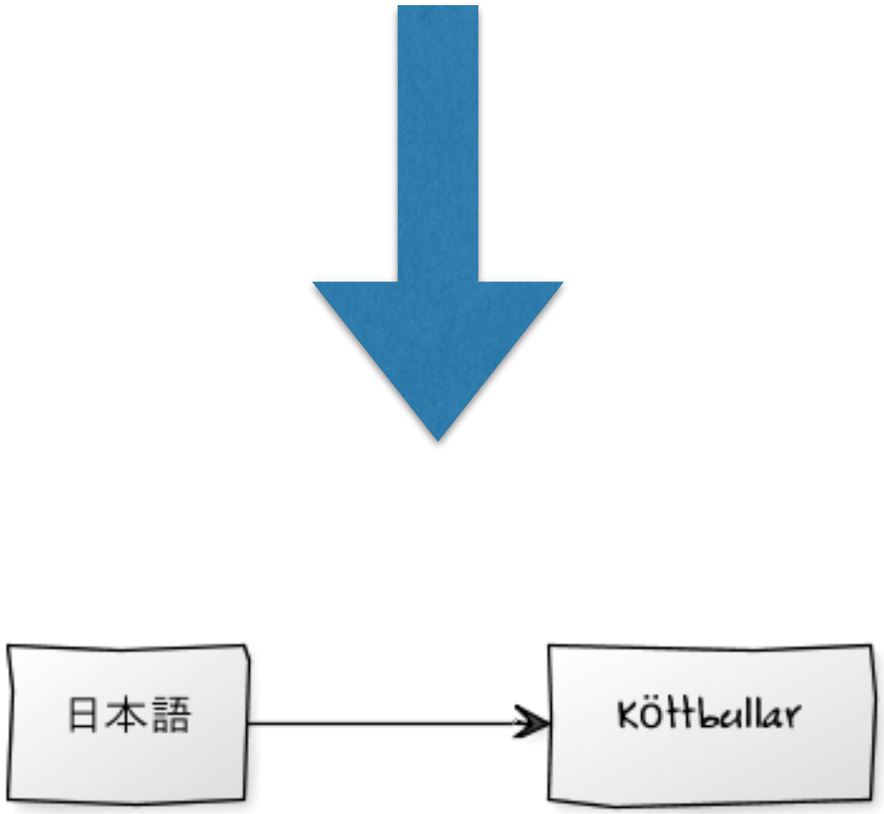
Background - Textual Language to Visual Language - Dot Language

```
graph graphname {
    a [label="Foo"];
    b [shape=box];
    a -- b -- c [color=blue];
    b -- d [style=dotted];
}
```

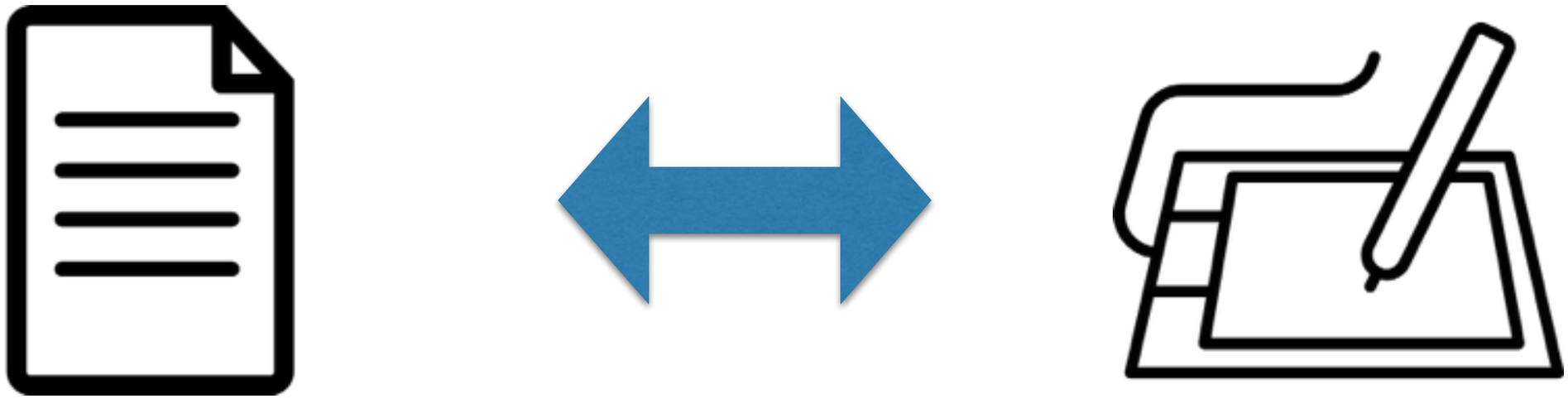


Background - Textual Language to Visual Language - yUML

```
<img src="http://yuml.me/diagram/scruffy/class/[日本語]->[Köttbullar]" >
```



Hybrid Language



Textual Language

Visual Language

Objective - Propose Architecture of Hybrid Languages

- Structured
- Reusability
- Performance
- Cross-Platform

React

- JavascriptScript library for creating user interfaces by **Facebook** and **Instagram**
- **Virtual DOM**
- **Component based**
- **JSX (Declarative programming)**

Comments

Pete Hunt

Hey there!

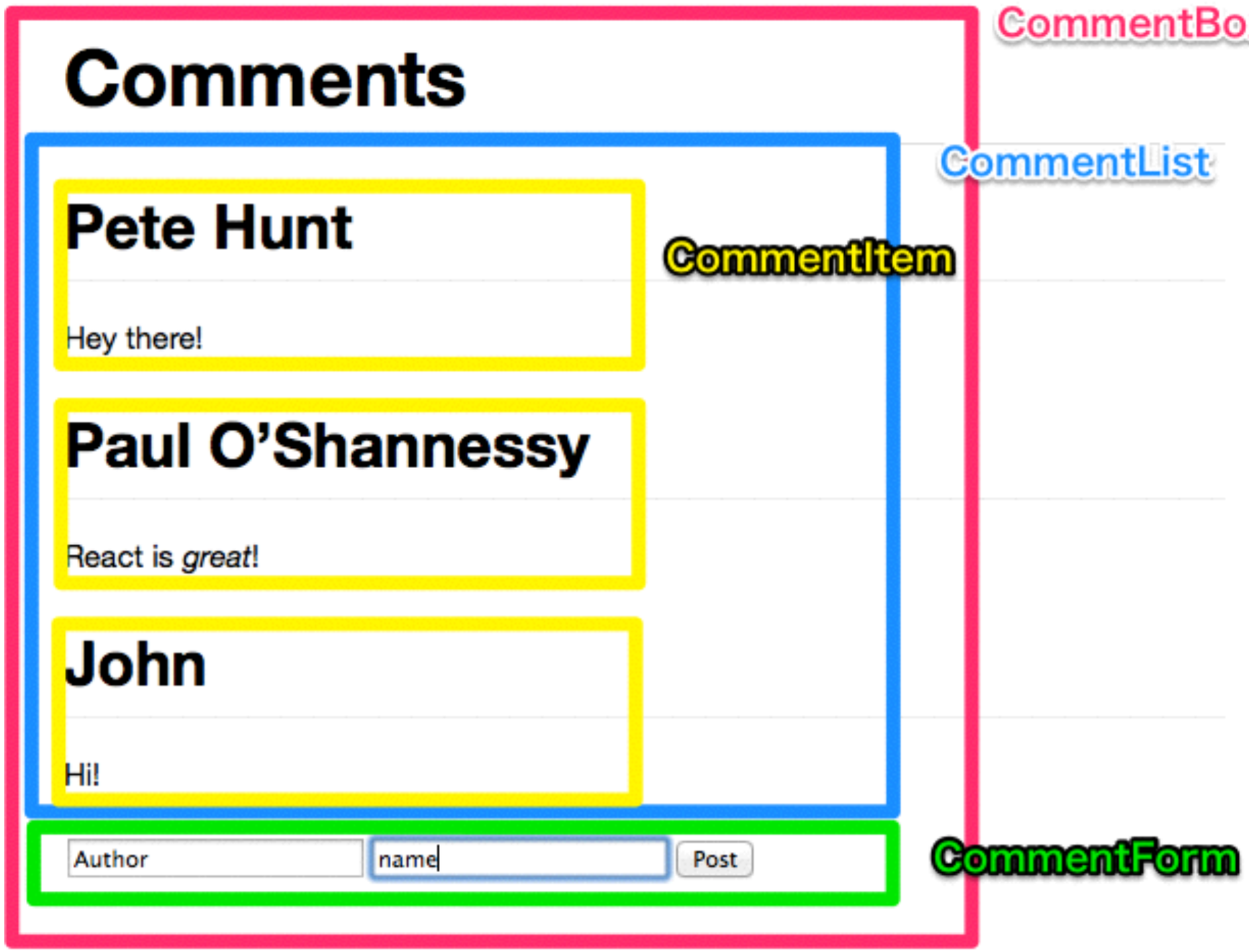
Paul O'Shannessy

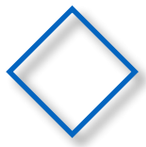
React is *great*!

John

Hi!

Author name Post

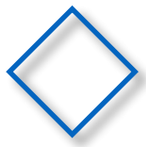

CommentBox
Comments
CommentList
Pete Hunt
CommentItem
Hey there!
Paul O'Shannessy
React is great!
John
Hi!
Author
name
Post
CommentForm



React

```
interface Data {
    author: string;
    text: string;
}

interface CommentBoxState {
    data: Data[];
}
```

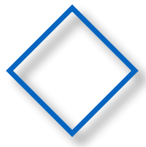
```
class CommentBox extends React.Component<CommentBoxProps, CommentBoxState> {
render() {
    return <div className="commentBox">
            <h1>Comments</h1>
        <CommentList data={this.state.data} />
        <CommentForm onCommentSubmit={this.handleCommentSubmit.bind(this)} />
    </div>;
    }
}
```



React

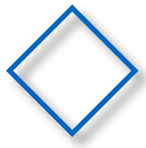
```
class CommentList extends React.Component<CommentListProps, any> {
    render() {
        var commentNodes = this.props.data.map(x => <CommentItem
author={x.author}>{x.text}</CommentItem>);
        return <div className="commentList">
            {commentNodes}
        </div>;
    }
}
```

```
class CommentForm extends React.Component<CommentFormProps, any> {

render() {
        return <form className= “commentForm"
onSubmit={this.handleSubmit.bind(this) }>
                <input type="text" placeholder="your name" ref="author" />
                <input type="text" placeholder="Say something..." ref="text" />
                <input type="submit" value="Post" />
               </form>;
    }
}
```



React

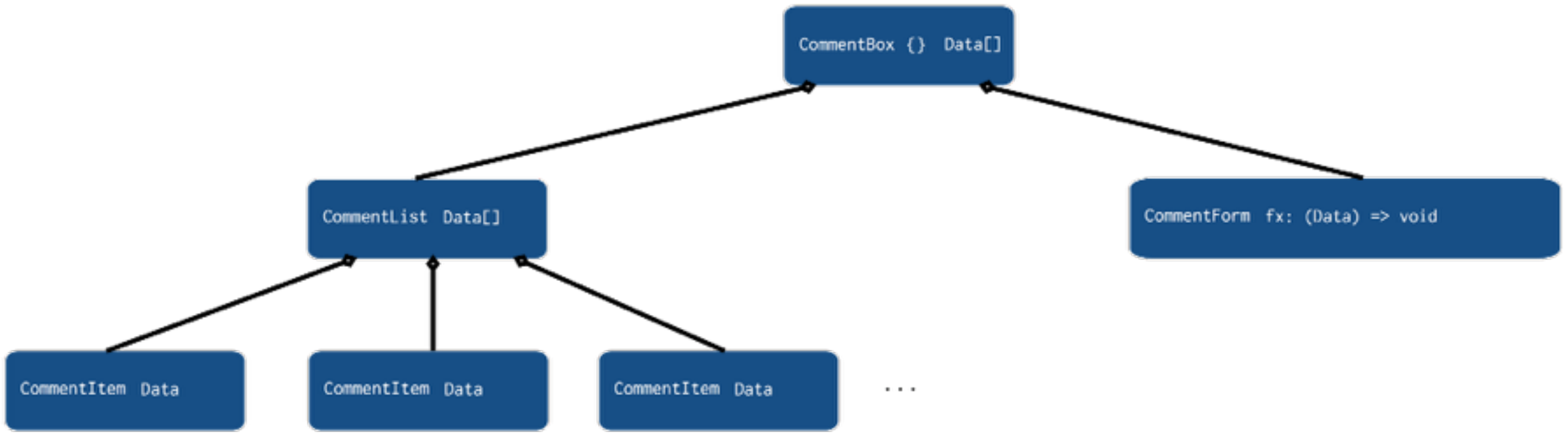
```
class CommentItem extends React.Component<CommentItemProps, any> {
    render() {
        var rawMarkup = marked(this.props.children.toString());
        return <div className="commentItem">
            <h2 className="commentAuthor">{this.props.author}</h2>
            <span dangerouslySetInnerHTML={{ __html: rawMarkup }}></span>
        </div>;
    }
}
```



React

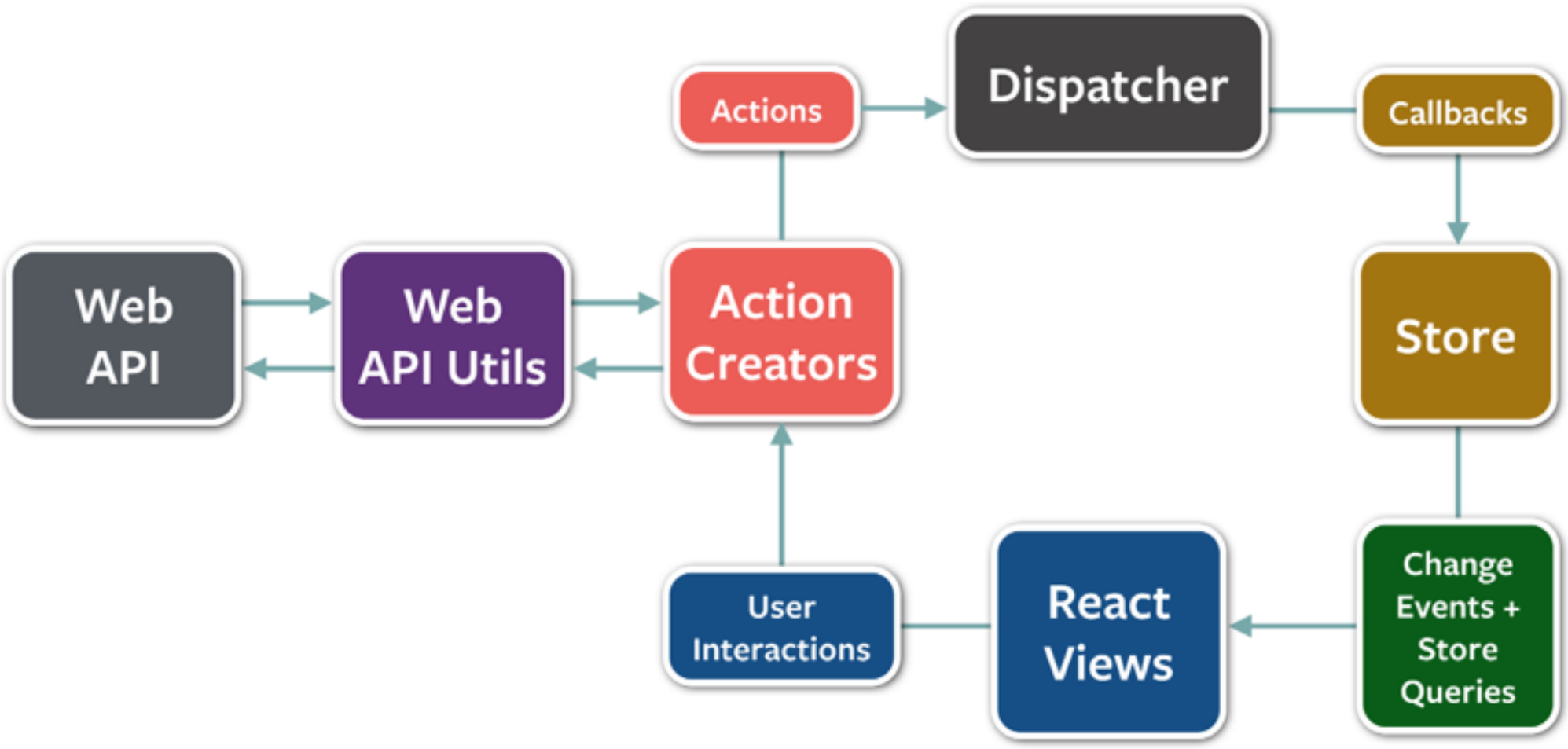
```
ReactDOM.render(
    <CommentBox url="api/comments.json"
poolInterval={2000} />,
    document.getElementById("content")
);
```

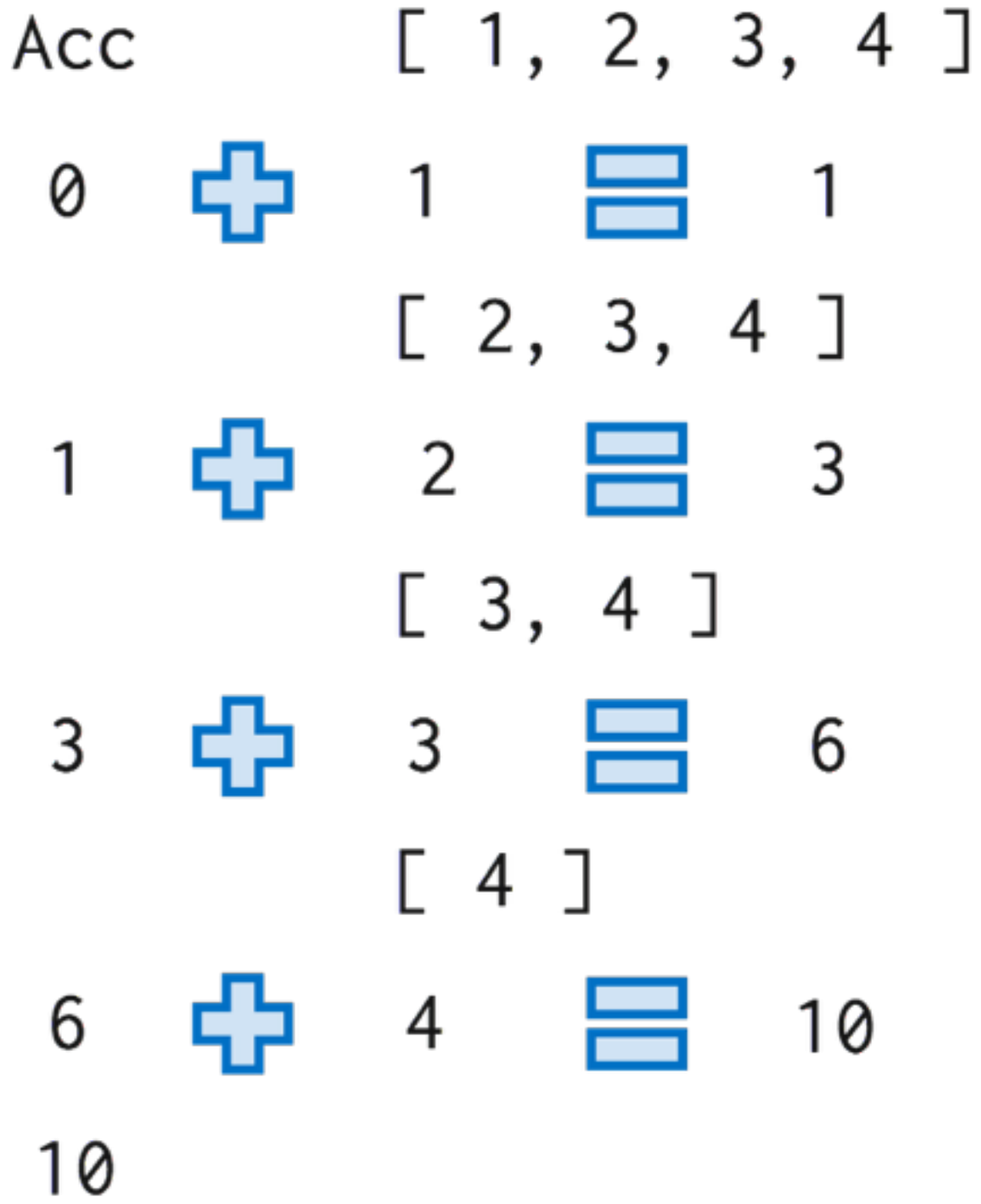
React



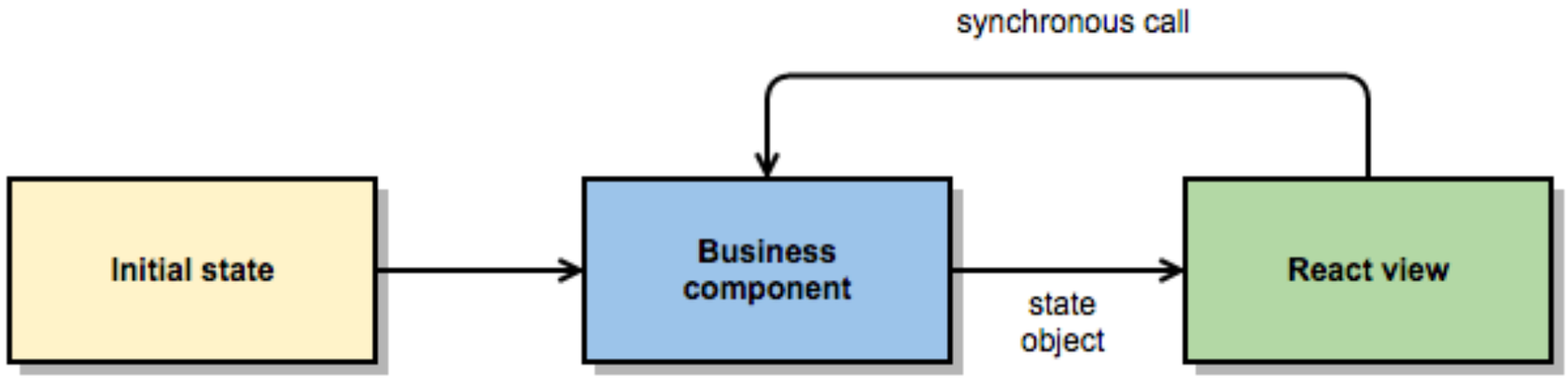

```
Data = {author: string, comment: string}
```

Flux

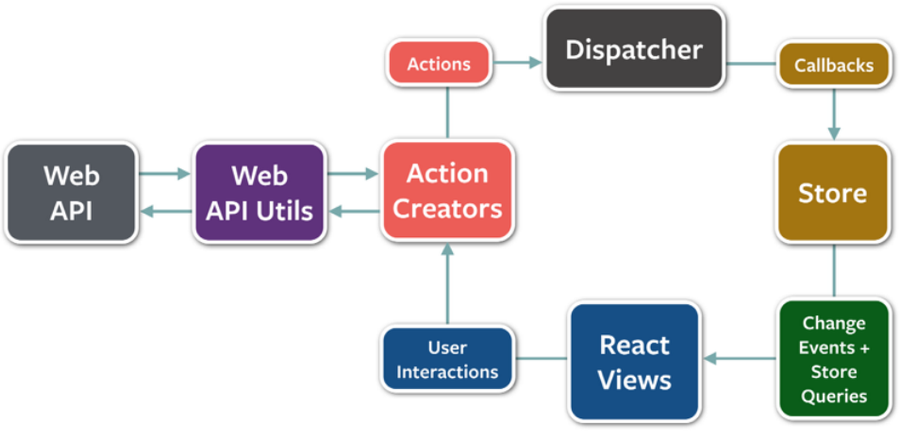

Actions
Dispatcher
Callbacks
Web API
Web API Utils
Action Creators
Store
User Interactions
React Views
Change Events + Store Queries

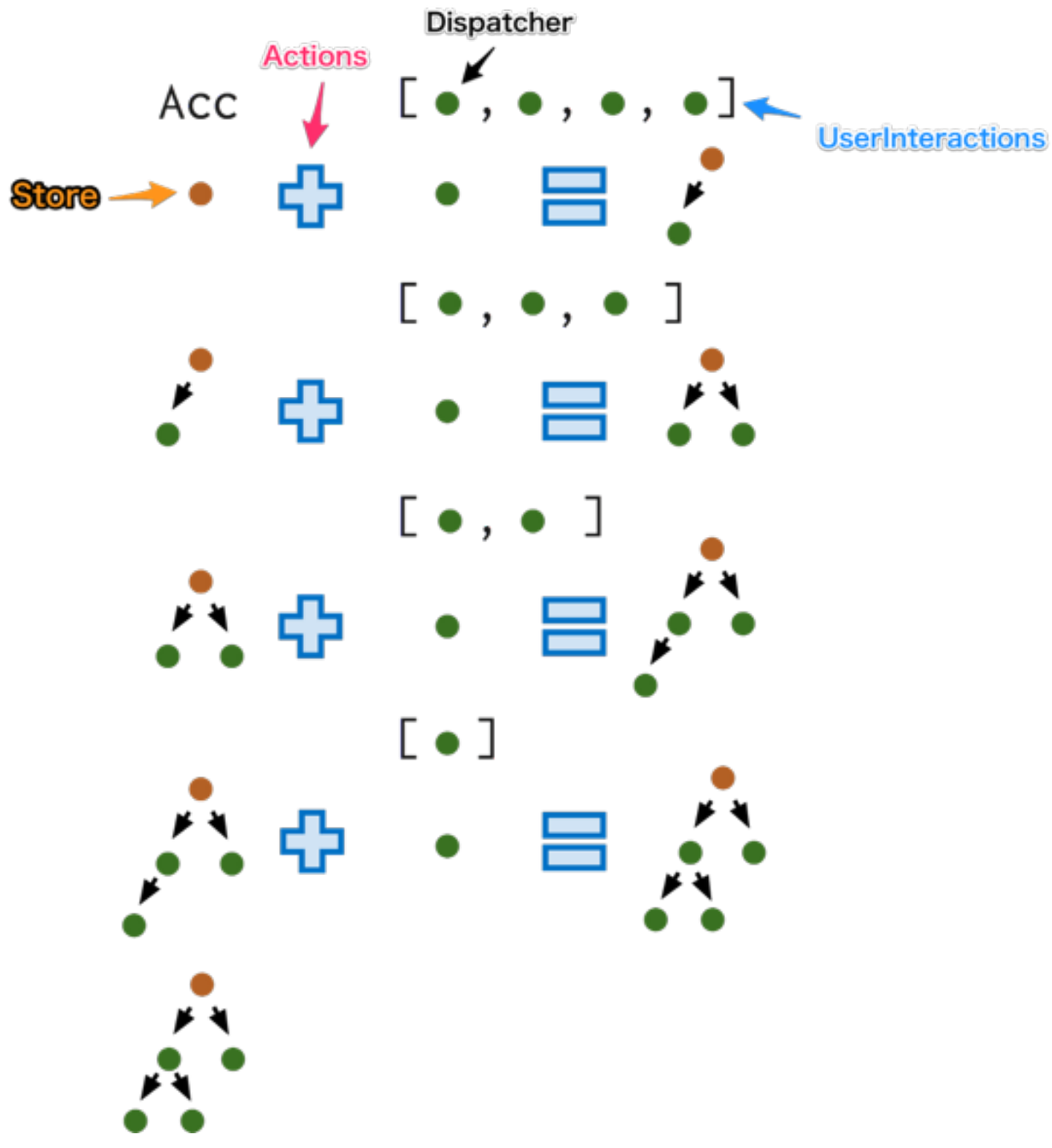

Acc
[1, 2, 3, 4]
0
1
1
[2, 3, 4]
1
2
3
[3, 4]
3
3
6
[4]
6
4
10
10

Flux with FRP (Bacon.js)

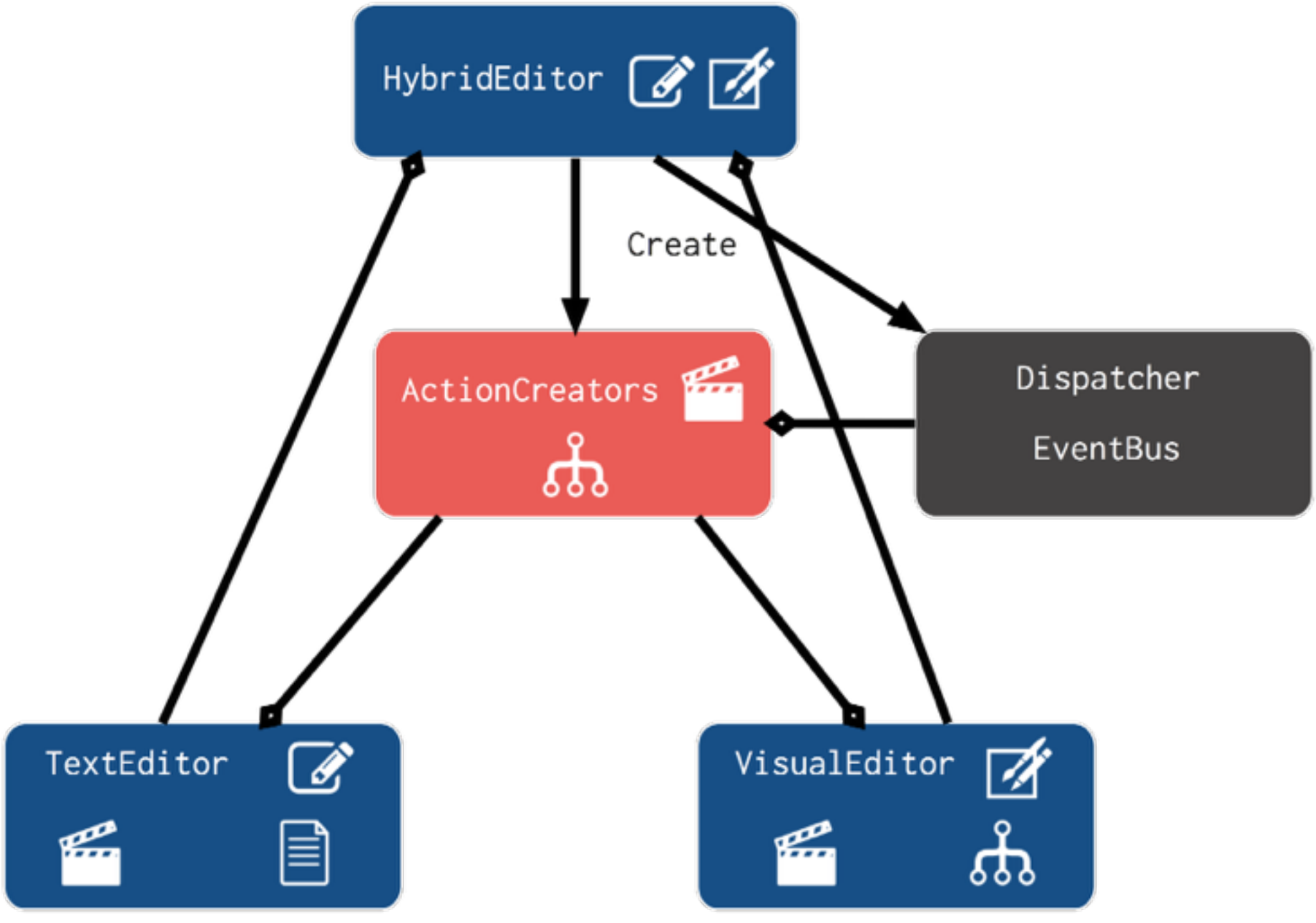


Flux with FRP (Bacon.js)

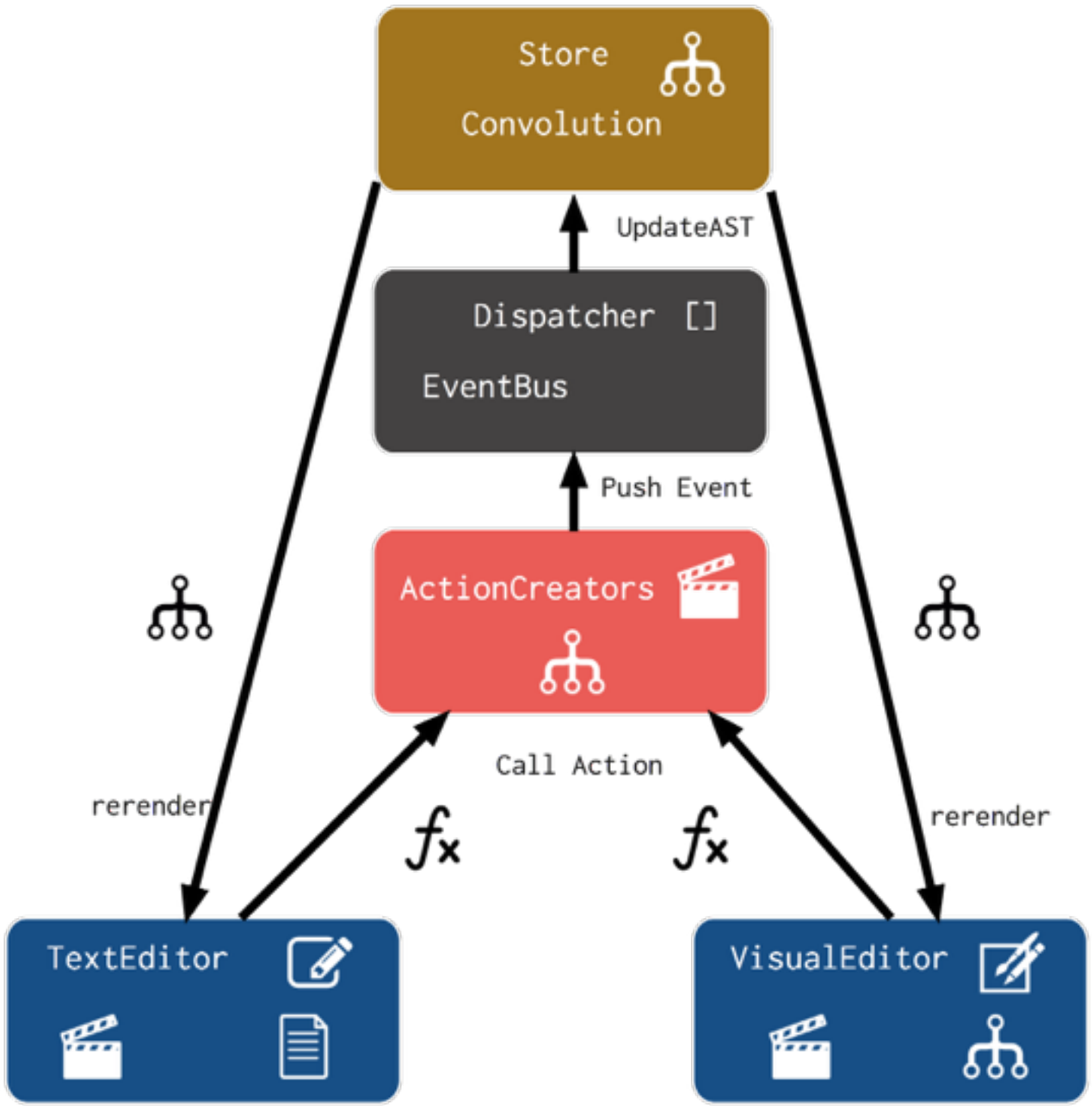




Architecture of Hybrid Languages



Architecture of Hybrid Languages



Case study - JSON Hybrid Language (Editor)

JSON (JavaScript Object Notation)

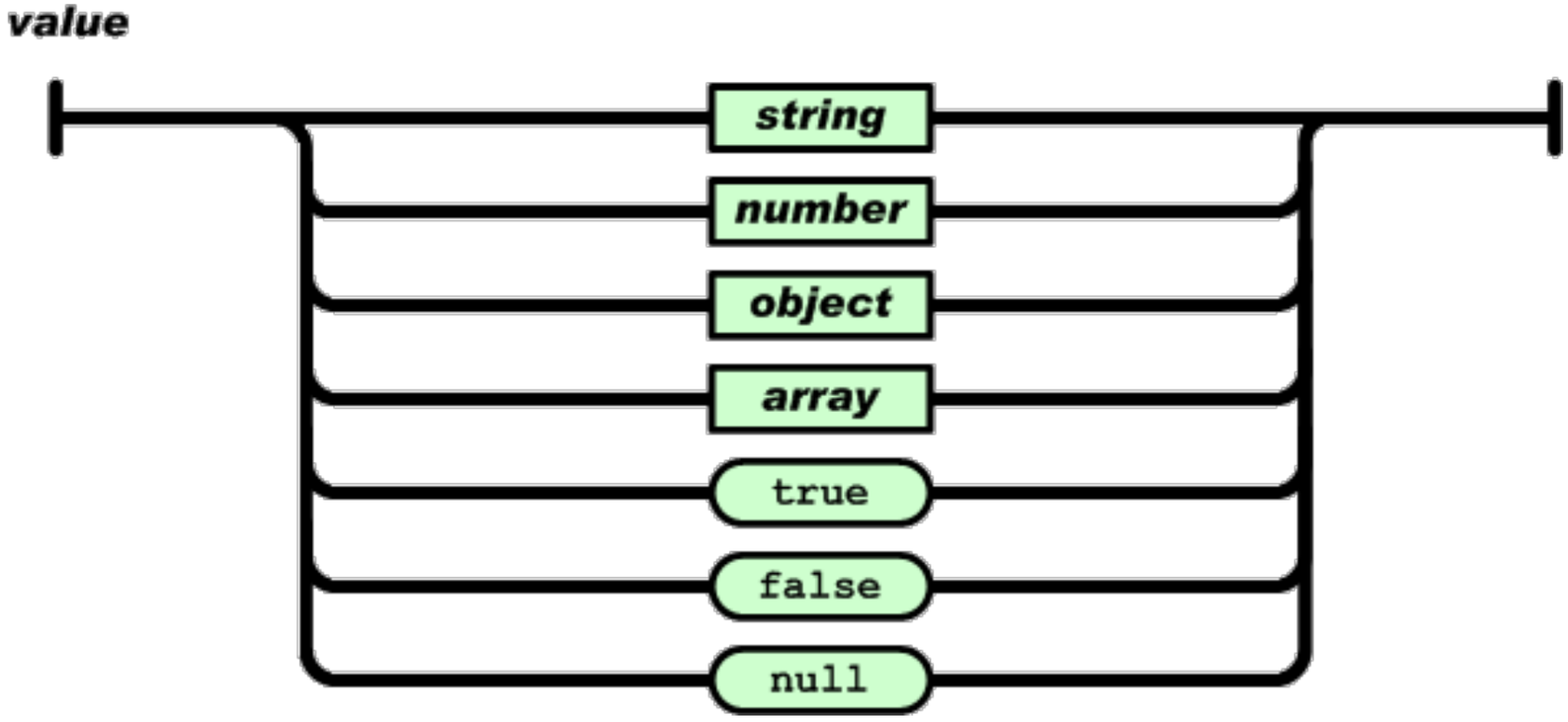
```
{"foo": [1, null], "baz": {"foo": [true, "bar"], "baz": "qux"}}
```

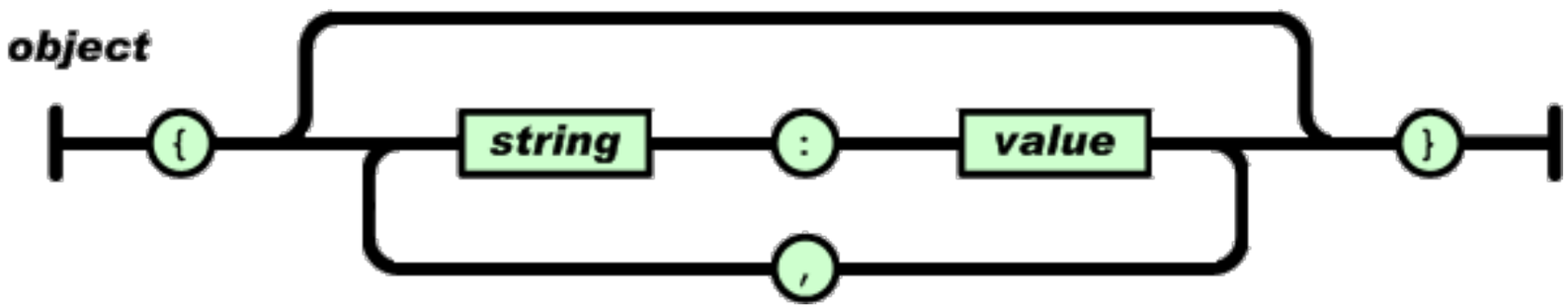
Case study - JSON Hybrid Language (Editor)

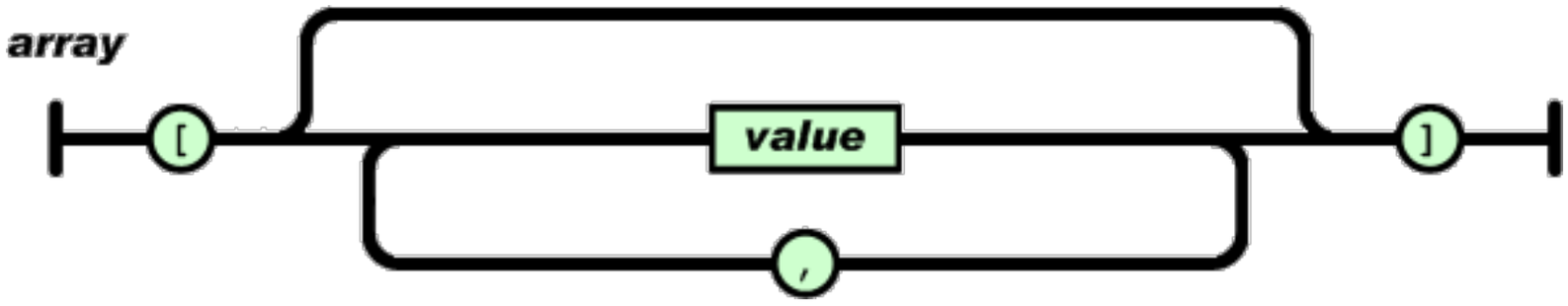
JSON - PEG

```
JSON ← S? ( Object / Array / String / True / False / Null /
Number ) S?

Object ← "{"
            ( String ":" JSON ( "," String ":" JSON )*
            / S? )
         "}"

Array ← "["
            ( JSON ( "," JSON )*
            / S? )
         "]"

String ← S? ["] ( [^ " \ U+0000-U+001F ] / Escape )* ["] S?

True ← "true"

False ← "false"

Null ←  “null”

Number ← Minus? IntegralPart FractionalPart? ExponentPart?
```

Case study - JSON Hybrid Language (Editor)







Case study - JSON Hybrid Language (Editor)

JsonText 

```
{"foo": [1, null],
"baz": {"foo": [true,
"bar"], "baz": "qux"}}
```

AST 

```
JsValue
JsNumber
JsString
JsNull
JsArray
JsField
JsObject
```

JSX </>

```
<JsValueComponent />
<JsNumberComponent />
<JsStringComponent />
 <JsNullComponent />
<JsArrayComponent />
<JsFieldComponent />
<JsObjectComponent />
```

Case study - JSON Hybrid Language (Editor)

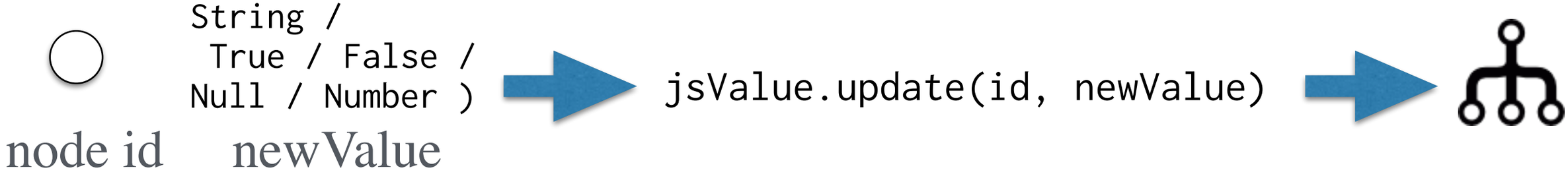
Action

```
{"foo": [1, null],
"baz": {"foo":
[true,
"bar"], "baz":
"qux"}}
```

updateJsonText

`JsAST.parse(text)`

AST



updateJsonAst

Case study - JSON Hybrid Language (Editor)

Demo

Conclusion

- Structured
- Reusability
- Performance
- Cross-Platform

Future work

- Usability Test
- Create Hybrid Languages by Architecture
- Architecture to Framework(Library)
- UI component